Міністерство освіти і науки України
Інститут спеціального зв'язку та захисту інформації
Національного технічного університету України
"Київський політехнічний інститут імені Ігоря Сікорського"

Д.В. Ланде, І.Ю. Субач, Ю.Є. Бояринова

ОСНОВИ ТЕОРІЇ І ПРАКТИКИ ІНТЕЛЕКТУАЛЬНОГО АНАЛІЗУ ДАНИХ У СФЕРІ КІБЕРБЕЗПЕКИ

Навчальний посібник

Київ – 2018

УДК 004.5
ББК 22.18, 32.81, 60.54
С 95

Рекомендовано до друку вченою радою Інституту спеціального зв'язку та захисту інформації Національного технічного університету України "Київський політехнічний інститут імені Ігоря Сікорського" (протокол № 13 от 26 липня 2018 р.)

Рецензенти:
д.т.н., професор С.В. Толюпа
д.т.н., професор В.В. Вишнівський

Л 95 **Ланде Д.В., Субач І.Ю., Бояринова Ю.Є. Основи теорії і практики інтелектуального аналізу даних у сфері кібербезпеки: навчальний посібник**. — К.: ІСЗЗІ КПІ ім. Ігоря Сікорського», 2018. — 297 с.

У навчальному посібнику розглядаються базові питання теорії і практики інтелектуального аналізу даних: алгоритми, моделі, задачі класифікації, кластерного аналізу, пошуку, глибинного аналізу даних (Data Mining), теорії складних мереж (Complex Networks), а також приводяться відомості, необхідні для математичного і комп'ютерного моделювання та аналізу складних систем і мереж в сфері кібербезпеки.

Інтелектуальний аналіз даних – це комплексний науковий напрям, що знаходиться на перетині таких наук, як дискретна математика, теорія штучного інтелекту, комп'ютерна лінгвістика, теорія графів, теорія алгоритмів тощо. Тому для його успішного проведення необхідні базові відомості із всіх цих областей, які частково викладені у цьому навчальному посібнику.

Видання призначено для студентів, курсантів і аспірантів закладів вищої освіти, які навчаються за спеціальністю 122 Комп'ютерні науки, а також дослідників і наукових співробітників, що працюють у сфері кібербезпеки.

УДК 004.5
ББК 22.18, 32.81, 60.54
ISBN 978-966-2577-12-9
Тираж 200 прим.

ЗМІСТ

ВСТУП

Згідно стратегії кібербезпеки України, одним з пріоритетних напрямків її забезпечення є створення системи своєчасного виявлення, запобігання та нейтралізації кіберзагроз і створення умов для впровадження в Україні сучасних технологій кіберзахисту.

Для цього необхідно вирішити ряд завдань, що полягають в: удосконаленні системи зберігання, передачі та обробки даних державних реєстрів і баз даних із застосуванням сучасних інформаційно-комунікаційних технологій (включаючи технології онлайн-доступу); розробленні нових методів запобігання кібератакам, кіберінцидентам та поширенню інформації про них; розробленні вимог (правил, настанов) щодо безпечного використання мережі Інтернет і надання електронних послуг державними органами та інші.

Ефективне вирішення поставлених завдань безпосередньо залежить від розробки та впровадження у діяльність суб'єктів забезпечення кібербезпеки (Державної служби спеціального зв'язку та захисту інформації України, Служби безпеки України, Міністерства оборони України та інших) новітніх інформаційних технологій, в основу яких, покладено результати наукової та науково-технічної діяльності вітчизняних та закордонних учених.

Яскравим представником даних технологій є інтелектуальний аналіз даних (Data Mining). Інтелектуальний аналіз даних (ІАД) – це сучасна концепція, яка дозволяє аналізувати дані різної природи, у тому числі величезних обсягів, що можуть бути неточними, неповними, суперечливими, різнорідними. Передбачається, що «розуміння» даних в конкретних програмних реалізаціях вимагає зусиль, що зазвичай притаманні людині, тобто зусиль інтелектуальних. Звідси і назва дисципліни. В інтелектуальному аналізі даних застосовуються математичні і програмні методи та засоби, що дозволяють виявляти невідомі раніше

закономірності і тенденцій в даних, які неможливо або важко виявити при традиційному аналізі через великі обсяги даних або їх складність.

З інтелектуальним аналізом даних тісно пов'язані два поняття – Knowledge Discovery in Databases (KDD) і Data Mining, що охоплюють такі задачі, як перетворення, зберігання, аналіз, моделювання та отримання інформації при прийнятті рішень на основі фактичних даних. Саме базовим поняттям дисципліни ІАД, класифікації методів і стадій глибинного аналізу даних присвячено перший розділ цього навчального посібника.

Як початкова задача ІАД розглядається обробка часових рядів, рядів вимірів, основам якої присвячено другий розділ цього навчального посібника. Вводиться поняття часового ряду, наведені основи дискримінантного аналізу, розглядаються практичні застосування. Відповідно, як узагальнення методів дискримінантного аналізу розглядається задача класифікації.

У третьому розділі надано формальний опис цієї задачі, розглянуто низку методів класифікації, починаючи від найпростіших лінійних класифікаторів, закінчуючи методом опорних векторів. Велику увагу приділено опису штучних нейронних мереж, принципам глибинного навчання. Четвертий розділ присвячено методам машинного навчання без вчителя – таким методам кластерного аналізу, як k-means, LSA/LSI, HAC, метод суфіксних дерев. Велику увагу приділено окремому виду нейронних мереж – самоорганізаційним картам Кохонена.

У п’ятому розділі розглядаються технології м'яких обчислень, орієнтовані на рішення задач управління слабко структурованими об'єктами, наведено класифікацію таких технологій, детально розглянуто концепцію клітинних автоматів, реалізацію мурашиного алгоритму. Моделюванню інформаційних потоків присвячено шостий розділ. Розглянуто лінійну, експонентійну, логістичну модель, модель дифузії інформації. Розглянуті кореляційні властивості інформаційних потоків,

наведено методологію їх розрахунків під час вирішення задач у сфері кібербезпеки.

Сьомий розділ присвячено фрактальному і мультифрактальному аналізу, його основам і практиці застосування щодо об'єктів інформаційного простору, зокрема, для задач прогнозування.

Восьмий розділ присвячено такій сучасній гілці кореляційного аналізу, як вейвлет-аналіз. Показано, що застосування вейвлет-аналізу надає можливості виявити відмінності в характеристиках процесів на різних шкалах, в різних точках на всьому досліджуваному інтервалі.

На цей час дані досліджуються не тільки у лінійному вимірі. Мережевий аналіз зв'язує ІАД з реальністю сучасного мережевого інформаційного середовища. Саме змістовному аналізу складних мереж, дослідженню його феноменів присвячено дев'ятий розділ навчального посібника.

У десятому розділі розглядаються класичні моделі інформаційного пошуку, що логічно поєднуються з елементами комп'ютерної лінгвістики (у тому числі мережевими), яким присвячено одинадцятий розділ.

І на останок, у Додатку наведено мінімально необхідну добірку відомостей щодо системи Matlab, яка розглядається як середовище виконання практичних завдань в рамках наведеного курсу.

1. БАЗОВІ ПОНЯТТЯ ГЛИБИННОГО АНАЛІЗУ ДАНИХ

1.1. Поняття інтелектуального аналізу інформації

Інтелектуальний аналіз даних – науковий і технологічний напрямок, пов'язаний з обробкою інформації та виявленням в ній закономірностей і тенденцій, які можуть застосовуватися при підтримці прийняття рішень. Можливість практичної реалізації інтелектуального аналізу даних виникла завдяки розвитку і повсюдному поширенню інформаційних технологій, виникненню і розвитку методів штучного інтелекту. Великі обсяги даних (Big Data), їх складність, мережева природа, динаміка і різноманітність інформації привели до вибухового зростання кількості і потужності методів і засобів інтелектуального аналізу даних.

Результат інтелектуального аналізу даних у загальному смислі – знання щодо закономірностей і тенденцій, що повинно мати наступні властивості:

- відображати результати дослідження системи (пізнання об'єктивної реальності);
- бути виражено певним, зрозумілим людині чином (використовує загальноприйняті символи, поняття, природну мову);
- бути компактним (за формою, опису), що робить його доступним до розуміння, інтерпретації і подальшого використання.

ІАД найчастіше вирішує чотири завдання – асоціація, кластеризація, класифікація і регресія. Коротко охарактеризуємо їх.

1. Асоціація – виявлення залежностей між пов'язаними подіями, що вказують, що з події *X* слідує подія *Y*. Такі правила називаються асоціативними. Вперше ця задача була запропонована для знаходження типових шаблонів покупок, що здійснюються в супермаркетах, тому іноді її ще називають аналізом споживчого кошика (market basket analysis). Якщо події можна впорядкувати за часом настання, то говорять про

послідовні шаблони – асоціативні правила, в яких важливий порядок проходження подій.

2. Кластеризація – це групування об'єктів (спостережень, подій) на основі даних (властивостей), що описують сутність об'єктів. Об'єкти усередині кластера повинні бути "схожими" один на одного і відрізнятися від об'єктів, що увійшли в інші кластери. Чим більше схожі об'єкти всередині кластера і чим більше відмінностей між кластерами, тим точніше кластеризація.

3. Класифікація – встановлення функціональної залежності між вхідними і дискретними вихідними змінними, що відповідають певним класам. За допомогою класифікації вирішується завдання віднесення об'єктів (спостережень, подій) до одного з заздалегідь відомих класів.

4. Регресія – встановлення функціональної залежності між вхідними і безперервними вихідними змінними. Прогнозування, зокрема, найчастіше зводиться до вирішення завдання регресії.

У сучасному ІАД прийнято виділяти два класи моделей Data Mining описові (дескриптивні), які необхідні для кращого розуміння досліджуваної системи, відомих фактів і спостережень, і передбачувальні, необхідні для розуміння нових фактів щодо системи.

Описова аналітика ближче до складної візуалізації та розвідувального аналізу даних в тому плані, що результат моделювання – компактне опис множини об'єктів у вигляді кластерів, правил, груп, а для побудови моделей непотрібен завдання цільової змінної. У першу чергу до описових моделей належать асоціативні правила і кластери. Основним недоліком описових моделей є їх відносна простота, що не дозволяє ефективно вирішувати завдання прогнозування.

Прогнозуюче моделювання дозволяє передбачати нові стани об'єктів, для чого використовуються алгоритми класифікації і регресії.

Крім того, концепція Data Mining дозволяє вирішувати такі завдання: аналіз відхилень – виявлення найбільш нехарактерних шаблонів; аналіз

зв'язків (link analysis) процес аналізу сукупності взаємовідносин між різними об'єктами для виявлення тенденцій і характеристик; аналіз виживання (survival analysis) – для оцінювання залежностей між характеристиками об'єкта з часом його життя. Нерідко ці завдання за допомогою спеціальних прийомів зводяться до перерахованих вище чотирьох основних завдань Data Mining.

Основу методів Data Mining складають методи класифікації, моделювання і прогнозування, засновані на застосуванні лісів і дерев рішень, штучних нейронних мереж, машин опорних векторів. До методів Data Mining нерідко відносять статистичні методи аналізу – дескриптивний, кореляційний і регресійний, факторний, дисперсійний, компонентний, дискримінантний, часових рядів. Такі методи, однак, припускають деякі апріорні уявлення про аналізованих даних, що дещо розходиться з цілями Data Mining (виявлення раніше невідомих нетривіальних і практично корисних знань).

Технологія ІАД є міждисциплінарною областю дослідження. Вона використовує методи таких дисциплін, як теорія інформації, системи штучного інтелекту, теорія ймовірностей, математична статистика, машинне навчання. Звідси велика кількість методів і алгоритмів, реалізованих у різних діючих системах Data Mining. Багато з таких систем інтегрують у собі відразу кілька підходів. Проте, як правило, в кожній системі є якась ключова компонента, на яку робиться головна ставка. При цьому основна увага приділяється обчислювальної ефективності алгоритмів, що застосовуються при обробці великих обсягів даних.

1.2. Поняття "інформація" і "знання"

До базових понять, які використовуються в інформатиці, відносяться дані, інформація і знання. Ці поняття часто використовуються як синоніми, проте між цими поняттями існують принципові відмінності.

У сучасних наукових концепціях трактування категорій "дані", "інформація" і "знання" залежить від області та ракурсу дослідження.

Розглянемо таку ієрархію понять «відомості – дані – інформація – знання».

Відомості є нижчим рівнем ієрархії і визначаються як частина знань, критерій істинності яких суб'єктивний у різних учасників пізнавального процесу. Під даними розуміються невпорядковані спостереження, числа, слова, звуки, зображення. Це набір дискретних, об'єктивних фактів. Термін дані походить від слова *data* – факт, а інформація (information) означає роз'яснення, виклад.

Дані – це сукупність відомостей, зафіксованих на певному носії в формі, придатній для постійного зберігання, передачі і обробки. Перетворення і обробка даних дозволяє отримати інформацію, тобто коли дані організовані, впорядковані, згруповані і категоризуються, вони стають інформацією.

Інформація – це сукупність даних, впорядкована з певною метою, що додає їм сенс. Інформація – це результат перетворення і аналізу даних. Відмінність інформації від даних полягає в тому, що дані – це фіксовані відомості, які зберігаються на певних носіях, а інформація з'являється в результаті обробки даних при вирішенні конкретних завдань. Наприклад, в базах даних зберігаються різні дані, а по певному запиту система управління базою даних видає необхідну інформацію.

Існують і інші визначення інформації, наприклад, інформація – це відомості про об'єкти і явища навколишнього середовища, їх параметри, властивості і стан, які зменшують наявну про них ступінь невизначеності, неповноти знань.

Знання трактується як інформація, готова до продуктивного застосування, дієва, змістовна. Знання – це зафіксована і перевірена практикою оброблена інформація, яка використовувалася і може багаторазово використовуватися для прийняття рішень.

Знання являють собою сукупність оформленого досвіду, цінностей, контекстуальної інформації, експертного розуміння, що складають основу для оцінки і інтеграції нових досвіду та інформації.

Взаємовідносини інформації і знань складні. Знання не тільки розташовуються на вищому ступені узагальнення, ніж дані та інформація, вони виконують також функцію структур, які систематизують і організують дані.

Існує декілька підходів до класифікацій знань. Наприклад, класифікація за ступенем науковості (наукове і ненаукове знання) або поділ на індивідуальне та групове знання. У 2003 р Європейська комісія представила класифікацію, яка розділяє знання наступним чином:

- наукові знання;
- технічні (технологічні) знання;
- інновації;
- людський капітал;
- кваліфікації (компетенції);
- інформаційно-комунікаційні технології.

Існує поділ на формалізовані і неформалізовані знання, певне Майклом Полані у 1966 році. Формалізовані знання – це знання, які точно визначені, а їх деталі можуть бути відтворені, передані та збережені в будь-якому вигляді: вербальному, письмовому, відео-, аудіо- або електронному носії, у формі текстів, графіків, формул, схем, цифр і т.п.

Неформалізовані знання знаходяться в пам'яті людей і організацій, доступні і можуть вільно передаватися іншим, але передача їх відбувається за допомогою безпосереднього спостереження, повторення.

Неформалізоване знання в значній мірі пов'язане з суб'єктивними властивостями системи (людини, колективу), і його важче передавати від системи до системи, а іноді й неможливо.

1.3. Глибинний аналіз даних (Data Mining)

Під глибинним аналізом даних (Data Mining) розуміють дослідження та виявлення комп'ютером (алгоритмами, засобами штучного інтелекту) у сирих даних прихованих структур і залежностей, які раніше не були відомі, нетривіальні, мають практичну цінність, доступні для інтерпретації людиною тощо.

Термін Data Mining було введено Григорієм Пятецкий-Шапіро в 1989 році. Працюючи в компанії GTE Labs, він зацікавився питанням: чи можна автоматично знаходити певні правила, щоб прискорити деякі запити до великих баз даних. Тоді ж було запропоновано два терміни – Data Mining і Knowledge Discovery In Data.

Англійське словосполучення Data Mining (DM) поки не має усталеного перекладу на українську мови. При перекладі використовуються наступні словосполучення: глибинний аналіз даних, видобуток даних, а також, саме, інтелектуальний аналіз даних, під яким розуміють "процес виявлення у початкових даних, раніше невідомих, нетривіальних, але корисних на практиці та доступних до інтерпретації знань, які є необхідними для прийняття рішень".

1.4. Класифікації методів Data Mining

Мета технології Data Mining – знаходження в даних таких закономірностей, які не можуть бути знайдені традиційними методами. Є два види моделей: предиктивні та описові.

Предиктивні моделі будуються на підставі набору даних з відомими результатами. Вони використовуються для прогнозу результатів на підставі інших наборів даних. Вимагається, щоб модель працювала максимально точно, була статистично значимою і виправданою. До них належать моделі класифікації, що описують правила або набір правил, відповідно до яких можна віднести опис будь-якого нового об'єкта до одного з класів. Такі правила будуються на підставі інформації про наявні

об'єкти шляхом поділу їх на класи; моделі послідовностей, що описують функції, що дають змогу прогнозувати зміну параметрів. Вони будуються на підставі даних про зміну певного параметра за минулий період часу.

Описові (descriptive) моделі пов'язані із залежностями в наборі даних, взаємного впливу різних чинників, тобто базуються на побудові емпіричних моделей різних систем. Ключовий момент у таких моделях – легкість і прозорість для сприйняття людиною. До них належать такі види моделей:

- кластеризації – описують групи (кластери), на які можна поділити об'єкти, дані щодо яких піддаються аналізу. Групуються об'єкти (спостереження, події) на основі даних (властивостей). Об'єкти усередині кластера мають бути подібними один до одного і відрізнятися від об'єктів, що ввійшли до складу інших кластерів;
- виключень – описують виняткові ситуації в записах, які різко відрізняються від основної множини записів;
- підсумкові (результатні) – виявлення обмежень на даних. Подібні обмеження важливі для розуміння даних масиву, тобто це нове знання, здобуте у результаті аналізу. Таким чином, Data Summarization – це знаходження яких-небудь фактів, які істинні для всіх або майже всіх записів у вибірці даних, що вивчається, але які досить рідко зустрічалися в усьому різноманітті записів;
- асоціації – виявлення закономірностей між пов'язаними подіями.

Для побудови розглянутих моделей використовуються різні методи й алгоритми Data Mining.

Більшість аналітичних методів, що використовуються у технології Data Mining – це математичні алгоритми і методи. Зокрема до методів і алгоритмів Data Mining відносять штучні нейронні мережі, дерева рішень, символьні правила, метод найближчого сусіда і *k*-найближчих сусідів, метод опорних векторів, байєсовські мережі, лінійну регресію,

кореляційно-регресійний аналіз, ієрархічні методи кластерного аналізу, неієрархічні методи кластерного аналізу, зокрема алгоритми *k*-середніх і *k*-медіа ми, методи пошуку асоціативних правил, метод обмеженого перебору, еволюційне програмування і генетичні алгоритми, різноманітні методи візуалізації даних тощо.

До базових методів Data Mining належать також підходи, що використовують елементи теорії статистики. Основна їх ідея зводиться до кореляційного, регресійного та інших видів статистичного аналізу. Основним недоліком їх є усереднювання значень, що призводить до втрати інформативності даних. Це у свою чергу спричинює зменшення кількості знань, що здобуваються.

Основним способом дослідження задач аналізу даних є їх представлення формалізованою мовою і подальший аналіз за допомогою моделі.

1.5. Огляд методів Data Mining

Вибір методів Data Mining часто залежить від типу наявних даних і від того, яку інформацію потрібно дістати. До найпоширеніших методів можна віднести:

- об'єднання (association; іноді вживають термін affinity, що означає подібність, структурну близькість) — виокремлення структур, що повторюються в часовій послідовності. Цей метод визначає правила, за якими можна встановити, що один набір елементів корелює з іншим. Користуючись ним, зокрема, аналізують ринковий кошик, розробляють каталоги, здійснюють перехресний маркетинг тощо;
- аналіз часових рядів (sequence-based analysis, або sequential association) , який дає змогу знаходити часові закономірності між даними (трансакціями). Наприклад, можна відповісти на запитання: купівля яких товарів передує купівлі даного виду

продукції? Метод застосовується, коли йдеться про аналіз цільових ринків, керування гнучкістю цін або циклом роботи із замовником (Customer Lifecycle Management);

- кластеризація (clustering) – групування записів, що мають однакові характеристики, наприклад за близькістю значень полів у базах даних. При цьому можуть залучатися статистичні методи або нейромережі. Кластеризація часто розглядається як перший необхідний крок для подальшого аналізу даних;
- класифікація (classification) — віднесення запису до одного із заздалегідь визначених класів;
- оцінювання (estimation);
- нечітка логіка (fuzzy logic);
- статистичні методи, що дають змогу знаходити криву, найближче розміщену до набору точок даних;
- генетичні алгоритми (genetic algorithms);
- фронтальні перетворення (fractal-based transforms);
- нейронні мережі (neural networks) — дані пропускаються через шари вузлів, «навчених» розпізнавати ті чи інші структури — використовуються для аналізу переваг і цільових ринків, а також для приваблювання замовників.

До Data Mining можна віднести ще візуалізацію даних — побудову графічного образу даних, що допомагає у процесі загального аналізу даних вбачати аномалії, структури, тренди. До Data Mining примикають дерева рішень і паралельні бази даних. Data Mining тісно пов'язана (інтегрована) зі сховищами даних (Data Warehousing, DW), що забезпечують роботу Data Mining.

1.6. Стадії глибинного аналізу даних

Data Mining складається з кількох фаз:

1) виявлення закономірностей (вільний пошук);

2) використання виявлених закономірностей для прогнозу невідомих значень (прогностичне моделювання);

3) аналіз виключень – стадія призначена для виявлення і пояснення аномалій, знайдених у закономірностях.

Виділяється типовий ряд етапів:

1. Формування гіпотези;

2. Збір даних;

3. Підготовка даних (фільтрація);

4. Вибір моделі;

5. Підбір параметрів моделі і алгоритму навчання;

6. Навчання моделі (автоматичний пошук інших параметрів моделі);

7. Аналіз якості навчання, якщо незадовільний перехід на п. 5 або п. 4;

8. Аналіз виявлених закономірностей, якщо незадовільний перехід на п. 1, 4 або 5.

Питання для самоперевірки

1. Що таке Data Mining?
2. Назвіть основні методи Data Mining.
3. В чому полягає інтелектуальний аналіз інформації?
4. Назвіть основні стадії Data Mining.

Література до розділу

1. *Ситюк В.Є.* Прогнозування. Моделі. Методи. Алгоритми: Навчальний посібник. – К.: «Маклаут», 2008. – 364 с. **2.** *Шумейко А.А., Сотник С.Л.* Интеллектуальный анализ данных (введение в Data Mining). – Днепр: Белая Е.А., 2012. – 212 с. **3.** *Гладун Ф.Я., Рогушина Ю.В.* Data Mining: пошук знань в даних. – К.: ВД «АДЕФ-Україна», 2016. – 452 с. **4.** *Дюк В*., *Самойленко А*. Data mining: учебный курс. – СПб: Питер, 2001. – 368 с. **5.** *Барсегян А.А.* Методы и модели анализа данных: OLAP и Data Mining / Барсегян А.А., Куприянов М.С., Степаненко В.В., Холод И.И. – Спб.: БХВ-Петербург, 2004. – 336 с.

2. СТАТИСТИЧНА ОБРОБКА ЧАСОВИХ РЯДІВ

2.1. Ведення в аналіз часових рядів

На цей час для вивчення властивостей складних систем, в тому числі і при експериментальних дослідженнях, широко використовується підхід, заснований на аналізі сигналів від системи. Це дуже актуально в тих випадках, коли математично описати досліджуваний процес практично неможливо, але в нашому розпорядженні є деяка характерних спостерігається величина. Тому аналіз систем, особливо при експериментальних дослідженнях, часто реалізується за допомогою обробки сигналів. Зазвичай метод дослідження називається реконструкцією динамічних систем. Цей розділ теорії динамічних систем називається аналізом часових рядів.

Спостережувана це послідовність значень деякої змінної (або змінних), що реєструються безперервно або через деякі проміжки часу. Часто замість терміна спостерігається використовується поняття часового ряду. Ясно, що наявність лише часового ряду замість повного вирішення рівнянь сильно обмежує знання про досліджувану систему. Це накладає великі обмеження на можливості методу реконструкції.

Скалярним часовим рядом $\{x_i\}$, $i=1,\ ...,\ N$ називається масив з N чисел, що представляють собою значення деякої динамічної змінної $x(t)$, що спостерігається, з деяким постійним кроком τ за часом, $t_i = t_0 + \tau(i-1)$: $x_i = x(t_i)$, $i=1,\ ...,\ N$. В аналізі часових рядів виділяються дві основні задачі: ідентифікації та прогнозу.

Задача ідентифікації при аналізі дає відповідь на питання щодо параметрів системи, що породили цей часовий ряд, а саме, щодо розмірності вкладення, кореляцій, ентропія тощо. Розмірність вкладення – це мінімальне число динамічних змінних, які однозначно описують процес.

Задача прогнозу має на меті за даними спостережень передбачити майбутні значення вимірюваних характеристик досліджуваного об'єкта, тобто скласти прогноз на певний відрізок часу вперед. Зараз розроблено і обґрунтовано декілька різних методів прогнозування. Всі вони поділяються на два основні класи: локальні і глобальні.

Історично першими були розроблені глобальні методи, в яких на основі статистичного аналізу пропонувалося використовувати процедури авторегресії, визначення ковзного середнього і ін. Пізніше в рамках нелінійної динаміки були розроблені нові практичні методики.

2.2. Класифікація часових рядів

Як і кожен вид аналізу, аналіз часових рядів передбачає вирішення конкретних завдань, таких як: вимірювання абсолютної і відносної швидкості росту або зниження рівня за окремі проміжки часу; визначення узагальнюючих характеристик рівня та швидкості його зміни за той чи інший період; виявлення і чисельна характеристика основних тенденцій розвитку явищ на окремих етапах; виявлення факторів, що обумовлюють зміну досліджуваного явища у часі; прогнозування розвитку явища в майбутньому (екстраполяція і інтерполяція).

Часові ряди розрізняються за такими ознаками:

1) за часом – моменті та інтервальні. Інтервальний ряд динаміки – послідовність, в якій рівень явища відноситься до результату, накопиченому або знову зробленому за певний інтервал часу. Якщо ж рівень ряду показує фактичну наявність досліджуваного явища в конкретний момент часу, то сукупність рівнів утворює моментний ряд динаміки. Основна відмінність моментних рядів від інтервальних: підсумовуючи сусідні рівні інтервального ряду ми щоразу отримуємо значущу величину – результат процесу за більш тривалий проміжок часу. Коли підсумовуємо сусідні рівні моментного часового ряду ми отримуємо

лише якусь умовну величину – необхідну, як правило, для подальших розрахунків і самостійно сенсу, як правило, не має.

2) за кількістю показників – комплексні ряди та ізольовані. Часовий ряд, ізольований в тому випадку, якщо рівень ряду представлений одним показником; якщо рівень ряду представлений системою узагальнюючих показників – комплексний;

3) залежно від відстані між рівнями ряду – повні і неповні. Повний часовий ряд мають у тому випадку, якщо відстань між датами на моментном ряду або інтервали в інтервальному ряду однакові. Неповний ряд буде в іншому випадку;

4) залежно від способу вираження рівнів – часові ряди абсолютних, відносних і середніх величин. При цьому часові ряди абсолютних величин є вихідними, а ряди відносних і середніх величин – як похідні. Часові ряди абсолютних величин більш повно характеризують розвиток процесу або явища. Ряди відносних величин можуть характеризувати в часі темпи зростання (або зниження) певного показника; зміна питомої ваги того чи іншого показника в сукупності; зміна показників інтенсивності окремих явищ.

2.3. Дискримінантний аналіз

Дискримінантний аналіз – розділ обчислювальної математики, багатовимірного статистичного аналізу, що представляє набір методів статистичного аналізу для вирішення задач розпізнавання образів, класифікації багатовимірних спостережень за принципом максимальної схожості при наявності навчальних ознак. Дискримінантний аналіз використовується для прийняття рішення про те, які змінні поділяють (тобто «дискримінують») вихідний набор даних, тобто виділяют так звані «групи». На відміну від кластерного аналізу в дискримінантному аналізі групи відомі апріорі.

Основні положення дискримінантного аналізу легко зрозуміти з уявлення щодо досліджуваної області, яка складається з окремих сукупностей, кожна з яких характеризується змінними з багатовимірним нормальним розподілом. Дискримінантний аналіз намагається знайти лінійні комбінації таких показників, які найкращим чином поділяють представлені сукупності.

При використанні методу дискримінантного аналізу головним показником є точність класифікації, і цей показник можна легко визначити, оцінивши частку правильно класифікованих елементів. Якщо дослідник працює з досить великою вибіркою, застосовується наступний підхід: виконується аналіз по частині даних (наприклад, по половині), а потім прогностичне рівняння застосовується для класифікації спостережень у другій половині даних. Далі оцінюється точність прогнозу, тобто відбувається перехресна верифікація. У дискримінантному аналізі існують методи покрокового відбору змінних, які допомагають здійснити вибір прогностичних змінних.

У дискримінантному аналізі формулюється правило, за яким об'єкти підмножини підлягає класифікації відносяться до одного з уже існуючих (навчальних) підмножин (класів). У загальному випадку задача розрізнення (дискримінації) формулюється таким чином. Нехай результатом спостереження над об'єктом є реалізація k-мірного випадкового вектора $\vec{x} = (x_1, x_2, ..., x_k)$. Потрібно встановити правило, згідно з яким за спостереженнями значенням вектора $\vec{x}$ об'єкт відносять до однієї з можливих сукупностей c_i, $i = 1,..,l$. Для побудови правила дискримінації простір R значень вектора $\vec{x}$ розбивається на області R_i, $i = 1,..,l$ так, що при попаданні $\vec{x}$ в R_i об'єкт відносять до сукупності X_i.

Правило дискримінації вибирається відповідно до визначеного принципом оптимальності на основі апріорної інформації щодо об'єкта з

X_i. При цьому слід враховувати розмір збитку від неправильної дискримінації. Апріорна інформація може бути представлена як у вигляді деяких відомостей щодо функції розподілу ознак у кожної сукупності, так і у вигляді вибірок з цих сукупностей. Апріорні ймовірності можуть бути задані, або ні. Очевидно, що рекомендації будуть тим точніше, чим повніше вихідна інформація. З точки зору застосування дискримінантного аналізу найважливішою є ситуація, коли вихідна інформація щодо розподілу представлена вибірками.

У цьому випадку завдання дискримінації ставиться таким чином.

Нехай $x_1^{(i)}$, ..., $x_{ni}^{(i)}$ вибірка із сукупності π_i, $i = 1, ..., l$, причому кожен *j*-й об'єкт вибірки представлений *k*-мірним вектором параметрів $x_j^{(i)} = \left(x_{j1}^{(i)}, ..., x_{jk}^{(i)}\right)$. Проведено додаткове спостереження $x = \left(x_1, ..., x_k\right)$ над об'єктом, що належить до однієї із сукупностей X_i.

Потрібно побудувати правило віднесення спостереження *x* до однієї з цих сукупностей.

Зазвичай у завданні розрізнення переходять від вектора ознак, що характеризують об'єкт, до лінійної функції від них, дискримінантної функції, гіперплощини, що найкращим чином розділяє сукупність вибіркових точок.

Найбільш вивчений випадок, коли відомо, що розподіл векторів ознак у кожній сукупності нормальний, але немає інформації про параметри цих розподілів. Тут природно замінити невідомі параметри розподілу в дискримінантній функції їх найкращими оцінками. Правило дискримінації можна засновувати на відношенні правдоподібності. Непараметричні методи дискримінації не вимагають знань щодо точного функціонального вигляду розподілів і дозволяють вирішувати завдання дискримінації на основі незначної апріорної інформації щодо сукупностей, що особливо цінно для практичних застосувань.

У параметричних методах ці точки використовуються для оцінки параметрів статистичних функцій розподілу. При цьому, як правило, використовується нормальний розподіл.

2.4. Лінійний дискримінантний аналіз (LDA)

Лінійний дискримінантний аналіз або LDA (Linear Discriminant Analysis) – це найстаріший з методів класифікації, розроблений Р. Фішером. Лінійний дискримінантний аналіз – це метод пошуку лінійної комбінації змінних, що найкращим чином розділяє деяку множину об'єктів на два або більше класів. Слід зазначити, що під загальною назвою LDA об'єднано декілька схожих методів, що розрізняються вимогами до властивостей вибірки. Нижче розглянуто один з цих методів, при якому висуваються лише дві вимоги до вибірки: 1) розмір вибірки повинен перевершувати число змінних; 2) класи можуть перетинатися, але їх центри повинні бути віддалені один від одного.

Висуваються припущення:

1. Є різні класи об'єктів.
2. Кожен клас має нормальну функцію щільності від *n* змінних

$$f_i(x)_{rs} = (2\pi)^{-1/2} |\Sigma_i|^{-1/2} \exp\left(-\frac{1}{2}(x-\mu^{(i)})\right)^T \Sigma_i^{-1}(x-\mu^{(i)}),$$

де μ (*i*) – вектор математичних очікувань змінних розмірності *k*;

Σ_i – коваріаційна матриця, позитивно визначена;

Σ_i^{-1} – обернена коваріаційна матриця.

Коваріаційна матриця – це матриця, утворена з попарних коваріацій кількох випадкових величин; точніше, для *k*-мірного випадкового вектора $X = (X_1,...,X_K)$ коваріаційна матриця – це квадратна матриця

$$\sum = E(X - EX)(X - EX)^T,$$

де $EX = (EX_1,...,EX_K)$ – вектор середніх значень.

Компоненти коваріаційної матриці дорівнюють:

$$\sigma_{ij} = E\left[(X_i - EX_i)(X_j - EX_j)\right] = cov(X_i, X_j), \quad i, j = 1 \dots k,$$

і при $i = j$ збігаються з DX_i (тобто на головній діагоналі знаходяться дисперсії величин X_i).

Головною проблемою в методі LDA є звернення матриці Σ. Якщо вона вироджена, то метод використовувати не можна.

У разі, якщо параметри відомі, дискримінацію можна провести наступним чином.

Нехай існують функції щільності $f_1(x), f_2(x), \dots, f_i(x)$ нормально розподілених класів. Задана точка x в просторі k вимірювань. Припускаючи, що точка x має найбільшу щільність, необхідно віднести цю точку до i-го класу. Існує теорема, яка доводить, що якщо апріорні ймовірності для визначених точок кожного класу однакові і втрати при неправильної класифікації i-ї групи в якості j-й не залежать від і та j, то вирішальна процедура мінімізує очікувані втрати при неправильної класифікації.

Нижче наведено приклад оцінки параметра багатовимірного нормального розподілу $\hat{\mu}^{(i)}$ і Σ, які можуть бути оцінені за вибірковими даними.

Нехай задано вибірку $\left(x_1^{(i)}, x_2^{(i)}, \dots, x_m^{(i)}\right) = x_i, (i = 1, \dots, l)$.

Математичні очікування $\mu_1, \mu_2, \dots, \mu_k$ можуть бути оцінені середніми значеннями

$$\hat{\mu}_q^{(i)} = \frac{1}{n_i} \sum_{j=1}^{ni} x_{jq}^{(i)}, \; q = 1 \dots k.$$

Незміщені оцінки елементів коваріаційної матриці Σ є

$$\left(\hat{\sum_i} \right)_{rs} = \frac{1}{n_l - 1} \sum_{j=1}^{ni} \left(x_{jr}^{(i)} - \hat{\mu}_r^{(i)} \right) \left(x_{js}^{(i)} - \hat{\mu}_s^{(i)} \right); \quad r, s = 1, \dots, k$$

Отже, можна визначити $\hat{\mu}^{(i)}$ і $\hat{\sum_i}$ за вибірками у кожному класі, отримавши оцінки, точку *x* необхідно віднести до класу, для якої функція *f* максимальна.

Вводиться припущення, що всі класи, серед яких повинна проводитися дискримінація, мають нормальний розподіл з однаковою коваріаційною матрицею Σ. У результаті істотно спрощується вираз для дискримінантної функції. Клас, до якого повинна належати точка *x*, можна визначити на основі нерівності

$$f_i(x) > f_{j(x)}$$

Необхідно скористатися наведеною вище формулою для випадку, коли їх коваріаційні матриці рівні: $\Sigma_i = \Sigma_j = \Sigma$, а $\mu^{(i)}$ – це вектор математичних очікувань класу і. Тоді наведену умову-нервіність можна представити у вигляді:

$$-\left[(x-\mu^{(i)})^T \Sigma^{-1}(x-\mu^{(i)})\right] > -\left[(x-\mu^{(j)})^T \Sigma^{-1}(x-\mu^{(j)})\right].$$

Зауважимо, що якщо є два вектори ***M*** і ***W***, то скалярний добуток можна записати у вигляді $Z^T W = W^T Z$ = (Z,W). У наведеному виразі–нерівності необхідно виключити $x^T \Sigma^{-1} x$ справа і зліва, поміняти у всіх членів суми знаки. Тоді вираз виглядає наступним чином:

$$(x, \Sigma^{-1}\mu^{(i)} - \tfrac{1}{2}(\mu^{(i)}, \Sigma^{-1}\mu^{(i)}) > (x, \Sigma^{-1}\mu^{(j)} - \tfrac{1}{2}(\mu^{(j)}, \Sigma^{-1}\mu^{(j)}) .$$

Внесемо позначення:

$$v^{(i)} = \Sigma^{-1}\mu^{(i)}, \quad i = 1,...,m,$$

$$\lambda_i = \frac{1}{2}\left(\mu^{(i)}, \Sigma^{-1}\mu^{(i)}\right), \quad i = 1,...,m.$$

Тоді нерівність прийме остаточний вигляд:

$$\left(x, v^{(i)}\right) - \lambda_i > \left(x, v^{(j)}\right) - \lambda_j.$$

Наслідок: точка *x* належить до класу *i*, для якого лінійна функція

$$h_i(x) = \left(x, v^{(i)}\right) - \lambda_i = max.$$

Перевага методу лінійної дискримінації Фішера полягає в лінійності дискримінантної функції і надійності оцінок коваріаційних матриць класів. Розглянемо частковий випадок, коли навчальний набір складається з двох матриць X_1 і X_2, в яких є по I_1 та I_2 рядків (зразків). Число змінних (стовпців) однаково – *J*. Вихідні припущення полягають у наступному:

1.Кожен клас ($k = 1$ або 2) має нормальний розподіл $N\left(\mu_k, \Sigma_k\right)$.

2. Коваріаційні матриці цих класів однакові: $\Sigma_i = \Sigma_j = \Sigma$

Класифікаційне правило в LDA дуже просте – новий зразок *x* відноситься до того класу, до якого він ближче по метриці Махаланобіса:

$$d_k = \left(x - \mu_k\right)\Sigma^{-1}\left(x - \mu_k\right)^T, \quad k = 1,2.$$

Як вже було показано, невідомі математичні очікування і коваріаційна матриця замінюються їхніми оцінками:

$$m_k = \frac{1}{I_k}\sum_{i=1}^{I_k} x_i, \quad S = \frac{1}{I_1 + I_2 - 2}\left(\tilde{X}_1^T \tilde{X}_1 + \tilde{X}_2^T \tilde{X}_2\right).$$

У цих формулах $\tilde{X}_k$ позначає центровану матрицю X_k. Якщо прирівняти відстані $d_1 = d_2$, то можна знайти рівняння кривої, що розділяє класи. При цьому квадратичні члени $xS^{-1}x^t$ скорочуються, і рівняння стає лінійним

$$xw_1^t - v_1 = xw_2^t - v_2,$$

де

$$w_k = m_k S^{-1}, v_k = 0.5 m_k S^{-1} m_k^t.$$

Величини, що стоять в різних частинах рівняння називаються LDA-рахунками, f_1 і f_2. Зразок відноситься до класу 1, якщо $f_1 > f_2$, і, навпаки, до класу 2, якщо $f_1 < f_2$.

2.5. Алгоритм дискримінантного аналізу

Нехай є дві генеральні сукупності X і Y, що мають нормальний закон розподілу з невідомими, але рівними коваріаційними матрицями.

Алгоритм виконання дискримінантного аналізу включає основні етапи:

Вихідні дані представляються або в табличній формі у вигляді q підмножин (навчальних вибірок) M_k і підмножини M_0 об'єктів, які підлягають дискримінації, або відразу у вигляді матриць $X^{(1)}$, $X^{(2)}$, ..., $X^{(q)}$, розміром ($n_k \times p$).

$$X^{(1)} = \begin{pmatrix} x_{1,1}^{(1)} & x_{1,2}^{(1)} & \dots & x_{1,p}^{(1)} \\ x_{2,1}^{(1)} & x_{2,2}^{(1)} & \dots & x_{2,p}^{(1)} \\ \dots & \dots & \dots & \dots \\ x_{n_1,1}^{(1)} & x_{n_1,2}^{(1)} & \dots & x_{n_1,p}^{(1)} \end{pmatrix} \quad X^{(2)} = \begin{pmatrix} x_{1,1}^{(2)} & x_{1,2}^{(2)} & \dots & x_{1,p}^{(2)} \\ x_{2,1}^{(2)} & x_{2,2}^{(2)} & \dots & x_{2,p}^{(2)} \\ \dots & \dots & \dots & \dots \\ x_{n_2,1}^{(2)} & x_{n_2,2}^{(2)} & \dots & x_{n_2,p}^{(2)} \end{pmatrix}$$

$$X^{(q)} = \begin{pmatrix} x_{1,1}^{(q)} & x_{1,2}^{(q)} & \dots & x_{1,p}^{(q)} \\ x_{2,1}^{(q)} & x_{2,2}^{(q)} & \dots & x_{2,p}^{(q)} \\ \dots & \dots & \dots & \dots \\ x_{n_q,1}^{(q)} & x_{n_q,2}^{(q)} & \dots & x_{n_q,p}^{(q)} \end{pmatrix} \quad X^{(0)} = \begin{pmatrix} x_{1,1}^{(0)} & x_{1,2}^{(0)} & \dots & x_{1,p}^{(0)} \\ x_{2,1}^{(0)} & x_{2,2}^{(0)} & \dots & x_{2,p}^{(0)} \\ \dots & \dots & \dots & \dots \\ x_{m,1}^{(0)} & x_{m,2}^{(0)} & \dots & x_{m,p}^{(0)} \end{pmatrix}$$

де $X^{(k)}$ – матриці з навчальними ознаками (k = 1, 2, ..., q); $X^{(0)}$ – матриця нових m-об’єктів, що підлягають дискримінації (розміром $m \times p$); p — кількість властивостей, якими характеризується кожен i-й об’єкт.

Тут повинна виконуватися умова: загальна кількість об'єктів N множини M має дорівнювати сумі кількості об'єктів m (в підмножині M_0), що підлягають дискримінації, і загальної кількості об’єктів $\sum_{k=1}^{q} n_k$ у навчальних підмножинах:

$$N = m + \sum_{k=1}^{q} n_k,$$

де q – кількість навчальних підмножин ($q \geq 2$). У реальній практиці найбільш часто реалізується випадок $q = 2$, тому і алгоритм дискримінантного аналізу наведемо для даного варіанту:

1. Визначаються $\bar{X}_j^k$ елементи векторів $\bar{X}^k$ середніх значень за кожної j-ї ознаки для i об'єктів всередині k–ї підмножини ($k = 1, 2$):

$$\bar{X}_j^{(k)} = \frac{\sum_{i=1}^{n} x_{ij}^{(k)}}{n_k}, \quad j = 1 \ldots p \quad .$$

2. Результати розрахунку представляються у вигляді векторів стовпців $\bar{X}^k$.

3. Для кожної навчальної підмножини розраховуються коваріаційні матриці $S(k)$ (розміром $p \times p$):

$$S^{(k)} = \left| \frac{1}{n_k} \sum_{k=1}^{n} (X_{ik}^{(k)} - \bar{X}_i^{(k)} \right| (X_{jk}^{(k)} - \bar{X}_j^{(k)})_{p \times p}$$

4. Розраховується об'єднана коваріаційна матриця $\check{S}$ за формулою:

$$\check{S} = \frac{1}{n_1 + n_2 - 2} (n_1 \times S^{(1)} + n_2 \times S^{(2)})$$

5. Розраховується матриця $\check{S}^{-1}$ обернена до $\check{S}$:

$$\check{S}^{-1} = \frac{1}{|\check{S}|} \times \hat{\check{S}}$$

де $|\check{S}|$— визначник матриці $\check{S}$, (причому $|\check{S}| \neq 0$), $\hat{\check{S}}$- приєднувальна матриця, елементи якої є алгебраїчними доповненнями елементів матриці.

6. Розраховується вектор-стовпець

$$A = \begin{vmatrix} A_1 \\ A_2 \\ \dots \\ A_p \end{vmatrix}$$

дискримінантних множників з урахуванням всіх елементів навчальних підмножин за формулою:

$$A = \hat{S}^{-1}(\bar{X}^{(1)} - \bar{X}^{(2)}).$$

Наведена розрахункова формула отримана за допомогою методу найменших квадратів за умови забезпечення найбільшого відмінності між дискримінантному функціями. Найкращий поділ двох навчальних підмножин забезпечується поєднанням мінімальної внутрішньогрупової варіації і максимальної груповий варіації.

7. По кожному *i*-му об'єкту (i = 1, 2, ..., N) множини M визначається відповідне значення дискримінантної функції:

$$F^{(k)} = A_1 x_{i,1}^{(k)} + A_2 x_{i,2}^{(k)} + \dots + A_p x_{i,p}^{(k)}$$

8. За сукупністю знайдених значень $F^{(k)}$ розраховуються середні значення для кожної підмножини M_k:

$$\bar{F}^{(k)} = \frac{\sum_{i=1}^{n_i} F_i^{(k)}}{n_k}, \quad k = 1,2,\dots$$

9. Визначається загальна середня (константа дискримінації) для дискримінантних функцій.

10. Виконується розподіл

$$\bar{F} = \frac{\sum_{k=1}^{q} \bar{F}^{(k)}}{q}$$

(дискримінація) об'єктів підмножини M_0 підлягають дискримінації за навчальними вибірками M_1 і M_2. З цією метою розраховані за п. 7 по кожному *i*-му об'єкту значення дискримінантних функцій

$$F_i^{(0)} = A_1 X_{i,1}^{(0)} + A_2 X_{i,2}^{(0)} + \cdots + A_p X_{i,p}^{(0)}, \quad i = 1,2,\ldots,m$$

порівнюються з величиною $\bar{F}$ загальної середньої. На основі порівняння даний об'єкт відносять до однієї з навчальних підмножин.

11. Далі робиться оцінка якості розподілу нових об'єктів, для чого оцінюється вклад змінних в дискримінантному функцію.

Вплив ознак на значення дискримінантної функції і результати класифікації може оцінюватися за дискримінантними множниками (коефіцієнтам дискримінації), по дискримінантному навантаженню ознак або по дискримінантної матриці. Дискримінантні множники залежать від масштабів одиниць виміру ознак, тому вони не завжди зручні для оцінки.

Дискримінантні навантаження більш надійні в оцінці ознак, вони обчислюються як парні лінійні коефіцієнти кореляції між розрахованими рівнями дискримінантної функції *F* і ознаками, взятими для її побудови.

Дискримінантна матриця характеризує міру відповідності результатів класифікації фактичному розподілу об'єктів на підмножини і використовується для оцінки якості аналізу. В цьому випадку дискримінантна функція *F* формується за даними об'єктів (з вимірюваними ознаками) навчальних підмножин, а потім перевіряється якість цієї функції шляхом зіставлення фактичної приналежності об'єктів до класів з тією, що отримана в результаті формальної дискримінації.

2.6. Квадратичний дискрімінатний аналіз (QDA)

Квадратичний дискримінантний аналіз, QDA (Quadratic Discriminant Analysis) є природним узагальненням методу LDA. QDA – багатокласовий метод і він може використовуватися для одночасної класифікації декількох класів $k = 1,\ldots, K$.

Нехай навчальний набір складається з *K* матриць $X_1, \ldots, X_k$, у яких є I_1 ,…, I_K рядків (зразків). Кількість змінних (стовпців) однакова.

Зберігаючи перше припущення LDA, відмовимося від другого, тобто допустимо, що коваріаційні матриці в кожному класі різні. Тоді QDA-рахунки обчислюються за формулою

$$f_k = (x - \mu_k)\Sigma_k^{-1}(x - \mu_k)^t + \log(\det(\Sigma_k^{-1})), \quad k = 1 \dots K$$

Класифікаційне правило QDA таке – новий зразок x відноситься до того класу, для якого значення критерію QDA найменше. На практиці, так само, як і в LDA, невідомі математичні очікування і коваріаційні матриці замінюються їх оцінками:

$$m_k = \frac{1}{I_k}\sum_{i=1}^{I_k} x_i, \quad S_k = \frac{1}{I_k}\tilde{X}_k^t\tilde{X}_k$$

У цих формулах $\tilde{X}_k$ позначає центровану матрицю X_k. Поверхня, що розділяє класи k і l визначається квадратним рівнянням $f_k = f_l$ тому метод і називається квадратичним.

2.7. Практичне застосування. Реалізація методів LDA і QDA

Для реалізації методів LDA і QDA в середовищі Matlab використовують функції gscatter і classify з Statistics Toolbox.

Функція gscatter призначена для побудови графіка розподілу двох змінних, що групуються за значеннями третьої змінної.

Функція classify забезпечує класифікацію методома лінійного або квадратичного дискримінантого аналізу.

Наведемо деякі синтаксичні реалізації функції gscatter:

gscatter(x,y,group);

gscatter(x,y,group,clr,sym),

де gscatter(x,y,group) реалізує побудову графіка від x і y, згрупованих за group. x і y – вектори того ж розміру, що й group – вектор або масив рядків із значеннями категорій, що відповідають певним значенням x і y.

gscatter(x, y, group, clr, sym, siz) – це узагальнення попереднього виразу, що визначає крім того колір, тип маркерів і розмір для кожної

групи. clr – масссив кольорів, відповідних функції plot, за умовчанням clr рівний 'bgrcmyk', sym – масив символів, також соответсвующих команді plot (за замовчанням – '.'), siz – вектор розмірів символів (за умовчанням – значення властивості 'DefaultLineMarkerSize').

Функція classify використовуватиметься в подальшому викладі в таких форматах:

class = classify(sample,training,group);

class = classify(sample,training,group,'type'),

де class = classify(sample, training, group) дозволяє класифікувати кожен рядок вхідних даних з sample за навчальною множиною (training), ставлячи їм у відповідність групу значень класифікатора (group). Тут sample і training – матриці з однаковим числом стовпців. Вихідний клас (class) вказує на групу, в якій кожен рядок визначений і має той самий тип, що і група (group).

Приклад: Необхідно побудувати лінійний і квадратичний класифікатори на підставі масиву точок, визначених координатами:

(8,3); (6,4); (1,7); (4,5);(2,1); (5,8); (7,3); (8,2); (5,5); (2,3); (6,0); (3,7); (0,6); (5,4); (4,3); (4,1); (7,2).

Попередня класифікація для цих точок задана вектором:

['a';'a';'a';'a';'b';'b';'b';'b';'c';'c';'c';'c';'c';'c';'c';'c';'c';'c'].

Визначити якість класифікації.

Для вирішення завдання у вікні Command Window вводиться:

```
>>a=[8;6;1;4;2;5;7;8;5;2;6;3;0;5;4;4;7]
>>b=[3;4;7;5;1;8;3;2;5;3;0;7;6;4;3;1;2]
>> g=[a,b]
>> species=['a';'a';'a';'a';'b';'b';'b';'b';'c';'c';'c';'c';'c';'c';'c';'c';'c';'c']
```

після чого будуємо графік (рис. 10) за допомогою функції gscatter :

```
>> gscatter(g(:,1), g(:,2), species,'rgb','osd');
```

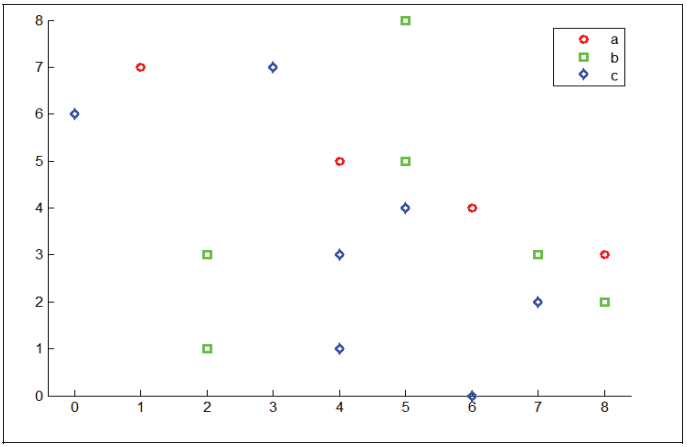


Рис. 2.1. Відображення результату тренувальної (попередньої) класифікації

Далі методом LDA будується класифікатор і за його допомогою виконується класифікація. Неспівпадаючі результати початкової класифікації і класифікації методом LDA називатимемо "помилками" і відмітимо закресленням (рис. 2.2).

Для виконання цього завдання у вікні Command Window вводяться команди:

```
>> hold on
>> linclass = classify(g(:,1:2),g(:,1:2),species,'quadratic');
>> N=size(a)
>> n=N(:,1)
>> k=0;
>> for i=1:n
>> if species(i)~=linclass(i)
>> k=k+1;
>> plot(g(i,1),g(i,2),'kx');
>> end
>> end
```

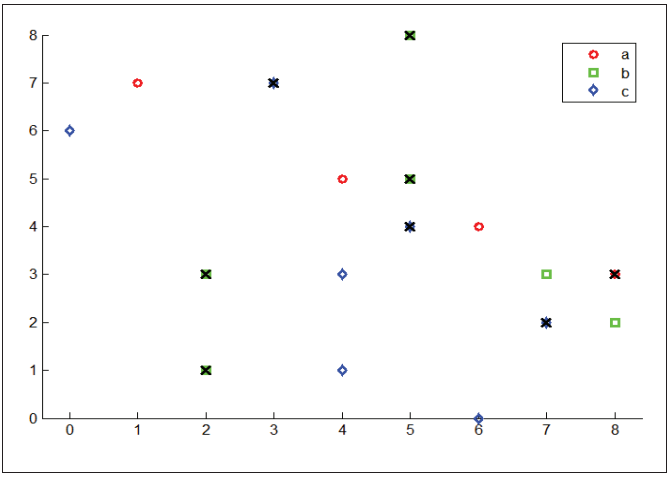


Рис. 2.2. Класифікація, уточнена методом LDA

Помилка класифікації обчислюється шляхом введення команди:

>> ldaResubErr = k / n

ldaResubErr =

0.4706

Як бачимо, помилка всього одна – це точка з координатами (1,3).

Для наочної демонстрації ліній LDA-класифікатора (рис. 2.3) класифікуємо точки, розташовані на площині.

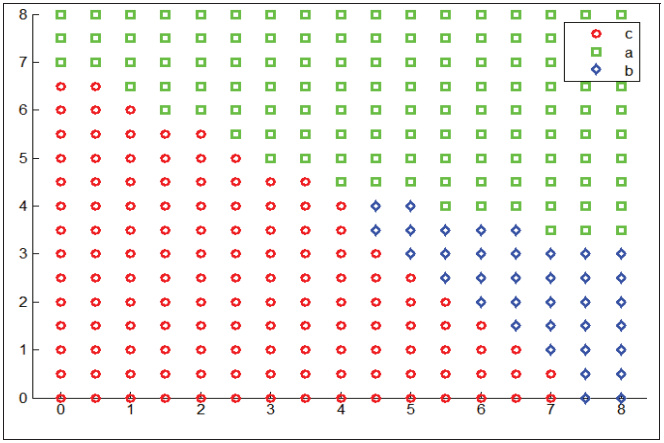


Рис. 2.3. Області LDA-класифікатора

Для цього введемо команди:

```
>> [x,y]=meshgrid(0:0.5:8,0:0.5:8)
>> x=x(:);
>> y=y(:);
>> j=classify([x y],g,species)
>> hold off
>>gscatter(x, y, j,'rgb','osd');
```

Для порівняння далі побудуємо класифікатор і виконаємо власну класифікацію за допомогою методу QDA, для чого введемо команди:

```
>> hold on
>> quaclass = classify(g(:,1:2),g(:,1:2),species,'quadratic');
>> N=size(a)
>> n=N(:,1)
>> k=0;
>> for i=1:n
>> if species(i)~=quaclass(i)
>> k=k+1;
>> plot(g(i,1),g(i,2),'kx');
>> end
>> end
```

Помилка класифікації також обчислюється шляхом введення команди:

```
>> qdaResubErr = k / n
qdaResubErr =
   0.0588
```

Питання для самоперевірки

1. За якими ознаками розрізняються часові ряди?
2. Що таке коваріаційна матриця?
3. В чому полягає лінійний дискримінантний аналіз?
4. В чому відмінність між лінійним та квадратичним дискримінантним аналізом?

5. Вивчити основні можливості лінійного і квадратичного дискримінантного аналізу.
6. Вивчити основні можливості пакету Statistics Toolbox.
7. Яке призначення і синтаксис функцій gscatter і classify?

Література до розділу

1. *Ситюк В.Є.* Прогнозування. Моделі. Методи. Алгоритми: Навчальний посібник. – К.: «Маклаут», 2008. – 364 с. **2.** *Дуброва Т.А., Архипова М.Ю.* Статистические методы прогнозирования в экономике. Учебно-методический комплекс. – М.: Изд. Центр ЕАОИ, 2008. – 136 с. **3.** *Дубров А.М.* Многомерные статистические методы и основы эконометрики. [Текст]: Учебное пособие. – М.: МЭСИ, 2008. – 79 с. **4.** *Калинина В.Н.* Введение в многомерный статистический анализ [Текст]: Учебное пособие. – ГУУ. – М., 2010. – 66 с.

3. МЕТОДИ КЛАСИФІКАЦІЇ ІНФОРМАЦІЇ

3.1. Визначення класифікації

Для групування об'єктів (документів) або їх ознак використовуються два основні прийоми – класифікація і кластеризація.

Класифікація та кластеризація це дві протилежні сторони стосовно участі користувача в процесі групування. Здійснення на практиці можливості автоматичного групування тематично близьких документів дозволяє побудувати тематичні каталоги. Механізм класифікації зазвичай встановлюється на відібраних документах тільки після завершення стадії автоматичного виявлення згрупованих даних (кластерів).

При класифікації використовуються статистичні кореляції для побудови правил розміщення об'єктів у певні категорії. Під класифікацією (Categorization) розуміють розподіл об'єктів за заздалегідь визначеними категоріями (на противагу кластеризації, де множина категорій заздалегідь невідома).

Методи класифікації текстів лежать на стику двох областей – машинного навчання (machine learning, ML) і інформаційного пошуку (information retrieval, IR).

Відповідно, автоматична класифікація може здійснюватися: на основі заздалегідь заданої схеми класифікації і вже наявного множини класифікованих документів; повністю автоматизовано.

Завдання класифікації полягає у визначенні приналежності об'єкту, що розглядається, одному або декільком класам. Класифікація текстів, зокрема, може визначатися загальною тематикою, наявністю певних дескрипторів або виконанням певних умов, іноді досить складних.

Для кожного класу експерти відбирають масиви типових об'єктів (документів), які використовуються системою класифікації в режимі навчання. Після того, як навчання закінчено, система за допомогою

спеціальних алгоритмів зможе розподіляти вхідні потоки документальної інформації за класами.

Класифікацію можна розглядати як задачу розпізнавання образів, при такому підході для кожного об'єкта виділяються набори ознак. У разі текстів ознаками є слова і взаємозалежні набори слів – терми, які містяться в текстах. Для формування набору ознак для кожного документа використовуються лінгвістичні і статистичні методи. Ознаки групуються в спеціальну таблицю – інформаційну матрицю. Кожен рядок інформаційної матриці відповідає одному з класів, кожен елемент рядка – одному з ознак; чисельне значення цього елемента визначається в процесі навчання системи класифікації. Коли навчання завершується, приналежність нового тексту до одного з класів встановлюється шляхом аналізу ознак цього тексту з урахуванням відповідних вагових значень. Існуючі алгоритми дозволяють здійснювати класифікацію з досить високою точністю, проте результати досягаються за рахунок великих розмірів інформаційної матриці, що визначаються загальним числом дескрипторів – термів.

Автоматична класифікація, зокрема, може застосовуватися в таких процедурах інформаційного пошуку, як:

- фільтрація (виборчий відбір) інформації;
- формування тематичних каталогів;
- пошук по класах;
- реалізація зворотного зв'язку за релевантністю шляхом класифікації результатів пошуку і вибору користувачем релевантних класів;
- розширення запитів за рахунок термів, що характеризують тематику класу;
- зняття омонімії (тобто облік тих випадків, коли один і той же слово може мати різне значення);
- автоматичне реферування.

3.2. Формальний опис класифікації

Нехай $D = \{d_1, ..., d_{|D|}\}$ – множина об‘єктів (вузлів мережі або, наприклад, документів), $C = \{c_1, ..., c_{|C|}\}$ – множина категорій, Ф – цільова функція, яка по парі $< d_i, c_j >$ визначає, чи відноситься документ d_i до категорії c_j (1 або True) або ні (0 або False). Задача класифікації полягає в побудові функції Ф', максимально близької до Ф.

Методи машинного навчання, які застосовуються для класифікації, передбачають наявність колекції заздалегідь класифікованих експертами об'єктів, тобто таких, для яких вже точно відомо значення цільової функції. Для того щоб після побудови класифікатора можна було оцінити його ефективність, ця колекція розбивається на дві частини, не обов'язково рівного розміру:

1. Навчальна (training-and-validation, TV) колекція. Класифікатор Ф' будується на основі характеристик цих об'єктів.

2. Тестова (test) колекція. На ній перевіряється якість класифікації. Об'єкти з tst не повинні використовуватися в процесі побудови класифікатора.

Вже згадана класифікація називається чіткою бінарною, тобто мається на увазі, що існують тільки дві категорії, які не перетинаються. До такої класифікації зводиться багато завдань, наприклад, класифікація по множині категорій $C = \{c_1, ..., c_{|C|}\}$ розбивається на $|C|$ бінарних класифікацій по множинам $\{c_i, \overline{c}_i\}$.

Часто використовується ранжирування, при якому множина значень цільової функції – це відрізок [0, 1]. Об'єкт при ранжируванні може ставитися не тільки до однієї, а відразу до декількох категорій з різним ступенем приналежності, тобто категорії можуть перетинатися між собою.

3.3. Ранжирування і чітка класифікація

Припустимо, що для кожної категорії побудована функція ранжирування. Розглянемо задачу, яка полягає в тому, щоб від функції ранжирування перейти до точної класифікації. Найбільш простий спосіб – для кожної категорії вибрати граничне значення (поріг). Якщо значення функції перевищує поріг, то документ відповідає категорії. Інший підхід: для кожного документа вибирати найближчі категорії, тобто категорії, на яких функція ранжирування приймає максимальні значення.

Припустимо, що для кожної категорії c_i побудована функція ранжирування (статусу приналежності до цієї категорії) CSV_i. Розглянемо задачу, яка полягає в тому, щоб від функції ранжирування перейти до точної класифікації. Найбільш простий спосіб – для кожної категорії c_i обрати граничне значення (поріг). Якщо $CSV_i(d) > \tau_i$, то документ d відповідає категорії c_i. Інший підхід: для кожного документа d обрати k найближчих категорій, тобто k категорій, на яких $CSV_i(d)$ приймає максимальні значення.

Обрати порогове значення можна декількома способами:

- Пропорційний метод. Навчальна колекція розбивається на дві частини. Для кожної категорії c_i на одній частині навчальної колекції обчислюється, яка частина документів їй належить. Граничні значення вибираються так, щоб на іншій частині навчальної колекції кількість документів, що залишились, віднесених до c_i, була такою ж.
- Метод k найближчих категорій. Кожен документ d_i вважається таким, що належить до k найближчих категорій, і відповідно до цього вибирається порогове значення.

3.4. Міра близькості об‘єкта та категорії

Нехай кожній категорії C_i відповідає вектор $\boldsymbol{C_i} = (c_{i1}, \ ..., \ c_{iN})$, де N – розмірність простору термів. За правило класифікатору можна використовувати скалярний добуток:

$$CSV_i(d) = \boldsymbol{d} \cdot \boldsymbol{C}_i = \sum_{j=1}^{N} c_{ij} d_j .$$

Нормалізація здійснюється таким чином, щоб кінцева формула для $CSV_i(d)$ представляла собою скалярний добуток – косинус кута між вектором категорії $\boldsymbol{C_i}$ та вектором із вагових значень термів, що входять до документу d – $\boldsymbol{d} = (d_1, \ ..., \ d_N)$:

$$CSV_i(d) = \frac{\boldsymbol{d} \cdot \boldsymbol{C_i}}{|\boldsymbol{d}| \cdot |\boldsymbol{C_i}|}.$$

Координати вектора $\boldsymbol{c_i}$ визначаються в процесі навчання, яке проводиться по кожній категорії незалежно від інших.

3.5. Метод Rocchio

Деякі класифікатори використовують так званий профайл для визначення категорії. Профайл – це список зважених термів, присутність або відсутність яких дозволяє найбільш точно відрізняти конкретну категорію від інших категорій. До таких методів класифікації відноситься і метод Rocchio, який відноситься до лінійних класифікаторів, в яких кожен документ представляється у вигляді вектора вагових значень термів. Профайл категорії i будемо розглядати як вектор $\boldsymbol{C_i} = (c_{1i}, \ ..., \ c_{Ni})$ (N – кількість термів в словнику), значення елементів якого c_{ki} в рамках методу Rocchio розраховується за формулою:

$$c_{ki} = \frac{\alpha}{|POS_i|} \cdot \sum_{d_j \in POS_i} w_{kj} - \frac{\beta}{|NEG_i|} \cdot \sum_{d_j \in NEG_i} w_{kj},$$

де w_{kj} – це вага терма t_k в документі d_j (розраховується, наприклад, по принципу *TF IDF*), $POS_i = \{d_j \mid \Phi(d_j, c_i) = 1\}$ і $NEG_i = \{d_j \mid \Phi(d_j, c_i) = 0\}$. В цій формулі α і β – контрольні параметри, які характеризують значимість позитивних і негативних прикладів. Наприклад, якщо $\alpha = 1$ і $\beta = 1$, то $\boldsymbol{C}_i$ буде центром мас всіх документів, що відносяться до відповідної категорії. Функція $CSV_i(d)$ визначається як величина, обернена до відстані від вектора з вагових значень термів, що входять в документ d, до профайла категорії i – $\boldsymbol{C}_i$, або як скалярний добуток цих векторів. Метод Rocchio дає задовільні результати в разі, коли документи з однієї категорії близькі один до одного по відстані.

3.6. Метод лінійної регресії

Регресійний аналіз використовується в випадках, коли ознаки категорій можуть бути виражені кількісно у вигляді деякої комбінації векторів вагових значень термів, що входять в документи з навчальної колекції. Отримана комбінація може використовуватися для визначення категорії, до якої буде ставитися новий документ. У найпростішому випадку для вирішення цього завдання використовуються стандартні статистичні методи, такі як лінійна регресія.

Метод регресії є варіантом лінійної класифікації, навченою відразу на всій колекції. При застосуванні регресійного аналізу для класифікації текстів розглядається множина термів (F) і множина категорій (C). У цьому випадку навчальній колекції документів ставляться у відповідність дві матриці:

- матриця документів $\boldsymbol{D}$ в навчальній колекції, в якій кожен рядок – це документ, а стовпець – терм, кількість рядків N – кількість документів в навчальній колекції;

- матриця відповідей $\boldsymbol{O} = \left\| o_{i,j} \right\|$, в якій рядок i відповідає документу $i = 1, ..., N$, стовпець j – категорії ($j = 1, ..., K$), а $o_{i,j}$ – значення $CSV_j(d_i)$.

Метод регресії базується на алгоритмі знаходження матриці правил $\boldsymbol{M}$, яка мінімізує значення норми матриці $\left\| \boldsymbol{MD - O} \right\|_F$, тобто:

$$M = \arg \min_M \left\| \boldsymbol{MD - O} \right\|_F .$$

В лінійній алгебрі під нормою матриці розуміється функція, яка ставить у відповідність матриці числову характеристику. Норма матриці відображає порядок величини матричних елементів. У даному випадку рекомендується використовувати норму Фробеніуса $\left\| \, . \, \right\|_F$, що дорівнює кореню квадратному із суми квадратів всіх елементів відповідної матриці:

$$\left\| A \right\|_F = \sqrt{\sum_{i,j} a_{ij}^{\ 2}}.$$

Елемент m_{ij} початкової матриці $\boldsymbol{M}$ буде відображати ступінь належності i–го терму j–й категорії.

3.7. Байєсовська логістична регресія

В моделі байєсовської логістичної регресії розглядається умовна ймовірність належності документу D класу C: $p(C \mid D)$.

Передбачається, що документ визначається термами, що входять в нього, тобто в рамках даної моделі документ – це вектор $D = (w_1, ..., w_N)$, де w_i – вага терма i, а N – розмір словника.

Модель байєсовської логістичної регресії задається формулою:

$$p(C \mid D) = \varphi(\beta \cdot D) = \varphi(\sum_{i=1}^{N} \beta_i \cdot w_i),$$

де $C \in \{0,1\}$, $\beta = \{\beta_1, ..., \beta_N\}$ - вектор параметрів моделі, а φ – логістична функція, в якості якої можна використовувати, наприклад:

$$\varphi(x) = \frac{1}{1+\exp(-x)}.$$

Основна ідея підходу полягає в тому, щоб використовувати попередній розподіл вектора параметрів β, в якому кожне конкретне значення β_i з великою ймовірністю може приймати значення, близьке до 0. При реальних розрахунках приймаються гіпотези про Гаусовський або Лапласів розподіл значень β_i, а також те, що всі величини β_i взаємно незалежні.

3.8. Класифікація на основі штучних нейронних мереж

Штучними нейронними мережами називаються обчислювальні структури, що моделюють процеси, які зазвичай відбуваються в мозку людини. Мозок людини містить біля 10^{11} нейронів. Прийнято вважати, що кожний нейрон в мозку людини містить тіло, аксон, 10 000 дендритів і синапси.

На цей час штучні нейронні мережі вирішують завдання класифікації, кластеризації, розпізнавання образів, апроксимації функцій, прогнозу тощо.

Модель окремого нейрона можна розглядати як комп'ютер, дійсно, потенціал нейрона (аксона) – це функція від потенціалу дендритів. По стану нейрона – визначаються величиною цього потенціалу.

Формальний нейрон

Розглянемо формальну модель нейрона, наведену на рис. 3.1. Вхідні сигнали через синапси і дендрити надходять в тіло нейрона, при цьому їх значення множаться на вагові коефіцієнти, які відповідають певним синапсах (які можуть змінюватися під час навчання), потім результати сумуються. На основі отриманої суми (NET), до якої застосовується деяка функція F, що зветься активізаційною, формується вихідний сигнал нейрона OUT.

Сигнал суматора визначається за формулою:

$$NET = \sum_{i=1}^{n} x_i w_i,$$

де n – кількість синапсів, x_i – вхідний сигнал i-го дендриту, w_i – вагові показники, NET – сигнал суматора.

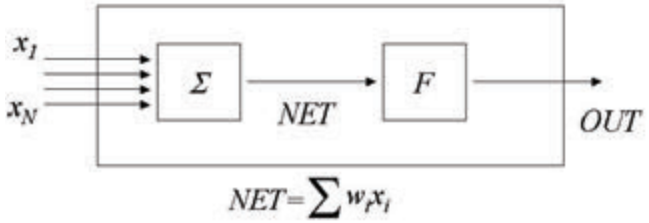


Рис. 3.1 – Формальний нейрон

Сигнал на виході нейрона має наступний вигляд:

$$OUT = F(NET).$$

Найчастіше застосовуються такі активізаційні функції:

$$OUT = K \cdot NET;$$

$$OUT = \begin{cases} 1, & NET > T, \\ -1, & NET \le T; \end{cases}$$

$$OUT = \frac{1}{1 + e^{-NET}};$$

$$OUT = \text{th}(NET).$$

Штучна нейрона мережа

Штучна нейронна мережа – це орієнтований граф, вершини якого - нейрони, розподілені по верствам, а ребра – синапси. Кожному ребру приписані свою вагу і функція провідності.

За архітектурою зв'язків можна виділити два класи нейронних мереж:

- мережі прямого поширення (Feed Forward) – односпрямовані з послідовними зв'язками, де нейрони не мають зворотних зв'язків;

- нейронні мережі із зворотними зв'язками, де вихід нейронів наступного шару прямує до нейронам попереднього шару.

Елементарну нейронну мережу прямого поширення прийнято називати перцептроном.

Перша версія перцептрону представляла собою одношарову нейронну мережу (Рис. 3.2), запропоновану в 1958 році нейрофізіологом Ф. Розенблаттом (F. Rosenblatt). У перцептроні кожен нейрон пов'язаний через синаптичний контакт з усіма рецепторами попереднього шару. Перцептрон Розенблатта був здатний розпізнавати найпростіші образи. Нейрон у цій моделі обчислює зважену суму елементів вхідного сигналу і пропускає результат через жорстку порогову функцію, вихід якої дорівнює +1 або –1.

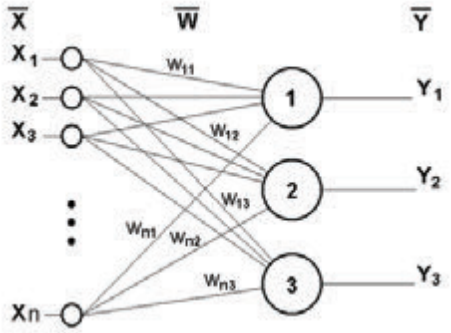


Рис. 3.2 – Одношаровий перцептрон:

$\bar{X}$ - вхідні сигнали; $\bar{W}$ - вагові значення синаптичних контактів;

(1), (2), (3) – нейрони; $\bar{Y}$ - вихідні сигнали

Типовий алгоритм навчання однокрокового перцептрону має вигляд:

1. Ініціалізація значень синаптичної ваги w_i $(i = 1, ..., N)$ і зсуву b: значення синаптичної ваги – випадкові значення.
2. Пред'явлення нейрону вхідного сигналу x_i $(i = 1, ..., N)$ і бажаного вихідного сигналу d.
3. Обчислення вихідного сигналу нейрона:

$$y(t) = sign\left(\sum_{i=1}^{N} w_i(t) x_i(t) - b \right),$$

де t – шаг ітерації, b – зсув.

4. Налаштування вагових значень:

$w_i(t+1) = w_i(t) + r[d(t) - y(t)]x_i(t), \ \ i = 1, \ ..., \ N,$

де $w_i(t)$ – вага зв'язку i-го елемента вхідного сигналу нейрона в момент t; r – швидкість навчання; $d(t)$ – бажаний вихідний сигнал. Якщо мережа приймає правильні рішення, то синаптичні ваги не змінюються.

5. Перехід до кроку 2.

Перцептрон становить інтерес завдяки його адаптивності, яка використовується в задачах розпізнавання образів. Однак було доведено, що одношарові нейронні мережі не здатні вирішувати багатьох завдань (наприклад, реалізувати логічну операцію виключне АБО). Для вирішення цих проблем використовуються багатошарові нейронні мережі. Багатошарові мережі можуть утворюватися каскадами шарів, в яких вихід одного шару є входом для наступного. Багатошарова нейронна мережа містить нейрони, які розподілені по шарам. У найпростішому випадку в мережі існує вхідний і вихідний шари, і мережа називається одношаровою. У загальному випадку мережа може містити багато проміжних шарів, так званих прихованих шарів, тобто бути багатошаровою. Відповідно нейрони, розташовані на проміжних шарах, називаються прихованими нейронами.

Нейронна мережа навчається, щоб по деякій множині вхідних сигналів видавати необхідну множину вихідних сигналів. Кожна множина сигналів при цьому розглядається як вектор. Навчання здійснюється шляхом послідовного пред'явлення вхідних векторів з одночасним коригуванням ваги. В процесі ітеративного навчання за визначеними правилами вагові значення стають такими, що кожен вхідний вектор породжує необхідний вихідний вектор.

Правила навчання перцептрона

Існує багато методик навчання нейронних мереж, наведемо одне з правил навчання одношарового перцептрона – дельта-правило (або правило Відрова-Хофа). Це правило базується на простій ідеї безперервної зміни синаптичних ваг для зменшення різниці («дельти») між значеннями бажаного і поточного вихідного сигналу нейрона. Алгоритм дельта-правила наступний:

1. Подати на шар перцептрону сигнал $X = \{x_1, ..., x_N\}$ і обчислити $OUT = \{OUT_1, ..., OUT_M\}$.
2. Для всіх $j = 1, ..., |X|$ обчислити $\delta_j = T_j - OUT_j$, де необхідне (бажане) значення T_j задає вчитель.
3. Якщо для всіх j виконується: $\delta_j = 0$, то навчання закінчується.
4. Якщо існує таке j, що $\delta_j \neq 0$, то відбувається коригування w_{ij} наступним чином: $w_{ij}(s+1) = w_{ij}(s) + \Delta_{ij}$, де $\Delta_{ij} = \gamma \delta_j x_i$, j – номер нейрона, i - номер синапсу, γ – коефіцієнт швидкості навчання.
5. Перехід до кроку 1.

Існує багато доступних програмних реалізацій штучних нейронних мереж, наприклад Toolbooks в такому пакеті, як Matlab.

Нейрона мережа як класифікатор

Класифікатор може являти собою нейронну мережу, входи якої відповідають термам, а виходи – категоріям. Для того щоб класифікувати документ $d^{(j)}$, вагові значення його термів $w_k^{(j)}$ подаються на відповідні входи мережі; активація поширюється по мережі, і значення, які отримані на виходах, є результатами класифікації. Типовий метод навчання такої мережі – зворотне поширення помилки (Back Propagation). Якщо на одному з тренувальних документів отримано неправильну відповідь на

одному з виходів, то помилка поширюється назад по мережі, і вагові значення ребер коригуються так, щоб цю помилку зменшити.

Глибоке навчання

Глибоке навчання (Deep Learning) – це сукупність методів машинного навчання (Machine Learning), заснованих на навчанні (Feature / Representation Learning), а не на спеціалізованих алгоритмах, розроблених для конкретних завдань. Багато методів глибокого навчання були відомі ще в минулому столітті, але результати були не вражаючими, поки просування в теорії штучних нейронних мереж і обчислювальні потужності не дозволили створювати складні технологічні архітектури нейронних мереж, що мають достатню продуктивність і дозволяють вирішувати широкий спектр завдань, які не піддавалися ефективному вирішенню раніше.

Глибоке навчання розглядає методи вирішення завдань штучного інтелекту (Artificial Intelligence) з використанням глибоких нейронних мереж. На цей час глибоке навчання дозволяє вирішувати велике коло складних завдань від розпізнавання мови і пошуку об'єктів на зображеннях до машинного перекладу текстів і автоматичного управління.

Системи глибокого навчання знайшли застосування в таких областях, як комп'ютерний зір, розпізнавання мови, обробка природної мови, аудіорозпізнавання, біоінформатика, де для ряду завдань були продемонстровані істотно кращі результати, ніж раніше.

Дослідження в цій галузі дозволили вдосконалити моделі роботи з великими обсягами немаркованих даних. Деякі підходи виникли в результаті досягнень в області нейронаук, успіхів інтерпретації обробки інформації, побудови комунікаційних моделей в нервовій системі, таких як нейронне кодування, пов'язане з визначенням відносини між стимулом і нейронними реакціями і взаємозв'язку електричної активності між нейронами в головному мозку людини.

Глибоке навчання характеризується, як клас алгоритмів машинного навчання, який:

- використовує багатошарову систему нелінійних фільтрів для вилучення ознак з перетвореннями. Кожен наступний шар отримує на вході вихідні дані попереднього шару. Система глибокого навчання може поєднувати алгоритми навчання з вчителем і без вчителя, при цьому аналіз зразка являє собою навчання без учителя, а класифікація – навчання з учителем;
- має декілька шарів виявлення ознак або параметрів представлення даних (навчання без вчителя). При цьому ознаки організовані ієрархічно, ознаки більш високого рівня є похідними від ознак нижчого рівня;
- є частиною більш широкої області машинного навчання;
- формує в процесі навчання шари на декількох рівнях представлення, які відповідають різним рівням абстракції; шари утворюють ієрархію понять.

При цьому для всіх систем глибокого навчання характерно:

- наявність декількох шарів нелінійної обробки;
- навчання (з учителем або без вчителя) ознак кожного шару, формуючи ієрархію від низького до високого рівня.

Склад конкретних нелінійних шарів залежить від розв'язуваної проблеми, що розв’язується. Використовуються як приховані шари нейронної мережі, так і шари складних логічних перетворень.

Глибоке навчання виражається набором алгоритмів машинного навчання для моделювання високорівневих абстракцій, застосовуючи архітектури, що включають численні нелінійні перетворення.

У першу чергу до глибинного навчання відносяться такі методи і їх варіації:

- системи навчання без учителя, такі як обмежена машина Больцмана для попереднього навчання, автокодувальник, глибока мережа довіри, генеративно-змагальна мережу,
- системи навчання з учителем, такі як згортаюча нейронна мережа, яка вивела на новий рівень технології розпізнавання образів,
- рекурентні нейронні мережі, що дозволяють навчатися на процесах в часі,
- рекурсивні нейронні мережі, що дозволяють включати зворотний зв'язок між елементами схеми і ланцюжками.

Комбінуючи ці методи, створюються складні системи, які відповідають різним завданням штучного інтелекту.

Рекомендації щодо підвищення продуктивності нейронної мережі

Розглянемо деякі евристичні рекомендації, що дозволяють підвищити швидкість навчання нейронної мережі.

- Послідовний режим навчання є кращим з точки зору застосування в процесах реального часу. Даний факт пояснюється меншими вимогами до обсягів пам'яті, необхідних для зберігання синаптичних ваг. Стохастичний пошук скорочує можливість зупинки пошуку в точці локального мінімуму.
- Максимізація інформативності. Цілком природна рекомендація для всіх методів машинного навчання, яка передбачає наявність кардинально різних прикладів в навчальній вибірці. Під різницею розуміються як відмінності щодо зовнішнього вигляду прикладів, що пред'являються, так і відмінності в значеннях функції помилки.
- Функція активації. Багатошарова повнозв’язна мережа навчається швидше, якщо функція активації є антисиметричною (тобто $\varphi(-x) = -\varphi(x)$). Зазначену властивість, наприклад, має функція гіперболічного тангенса.

- Нормалізація входів. Всі вхідні змінні повинні бути попередньо нормалізовані по всій навчальній множині, щоб середнє значення було близько до нуля.
- Початкова ініціалізація синаптичних ваг. Якщо синаптичні ваги приймають великі початкові значення, то нейрони з великою ймовірністю досягнуто режиму насичення. Як наслідок, локальні градієнти прийматимуть малі значення, і станеться гальмування процесу навчання. Якщо синаптичні ваги приймають занадто маленькі початкові значення, то метод буде повільно працювати в околі початку координат поверхні помилок.
- Швидкість навчання синаптичних ваг на різних шарах нейронної мережі. В ідеальному випадку всі нейрони повинні навчатися з однаковою швидкістю. Однак останні шари зазвичай мають більш високі значення градієнтів, ніж початкові. Тому має сенс останнім шарам призначати менші значення параметра швидкості навчання η в порівнянні з першими шарами.

3.9. Метод опорних векторів

Метод опорних векторів (Support Vector Mashine, SVM), що запропонований В.Н. Вапніком, відноситься до групи граничних методів класифікації. Він визначає приналежність об'єктів до класів за допомогою визначення границь областей.

Розглядається бінарна класифікація, тобто тільки за двома категоріями c і $\bar{c}$ (береться до уваги те, що цей підхід може бути розширений на будь-яке кінцеве число категорій). Крім того передбачається, що кожен об'єкт класифікації є вектором в N-вимірному просторі. Кожна координата вектора – це деяка ознака, кількісно тим більша, чим більше ця ознака виражена в даному об'єкті.

Передбачається, що існує навчальна колекція – множина векторів $\{x_1, ..., x_n\} \in R^N$ і чисел $\{y_1, ..., y_n\} \in \{-1, 1\}$. Число y_i дорівнює 1 в разі

належності відповідного вектора x_i категорії c, і -1 – в іншому випадку. Лінійний класифікатор – це один з найпростіших способів вирішення завдання класифікації. В цьому випадку шукається пряма (гіперплощина у N-мірному просторі), що відокремлює всі елементи одного класу від елементів іншого класу. Якщо вдається знайти таку пряму, то задача класифікації зводиться до визначення взаємного розташування точки і прямої: якщо нова точка лежить з одного боку прямої (гіперплощини), то вона належить класу c, якщо з іншого боку – класу $\overline{c}$.

Формалізуємо цю класифікацію: необхідно знайти вектор $\boldsymbol{w}$ такий, що для деякого граничного значення b і нової точки x_i виконується:

$$y_i = \begin{cases} +1, \text{ якщо } \boldsymbol{w} \cdot \boldsymbol{x_i} \geq b, \\ -1, \text{ якщо } \boldsymbol{w} \cdot \boldsymbol{x_i} < \boldsymbol{b}, \end{cases}$$

де $\boldsymbol{w} \cdot \boldsymbol{x}_i$ – скалярний добуток векторів $\boldsymbol{w}$ і $\boldsymbol{x}_i$:

$$\boldsymbol{w} \cdot \boldsymbol{x}_i = \sum_{j=1}^{N} w_j x_{i,j}.$$

Рівняння $\boldsymbol{w} \cdot \boldsymbol{x}_i = b$ описує гіперплощину, що розділяє класи. Тобто, якщо скалярний добуток векторів w і x_i не менш значення b, то нова точка належить до першого класу, якщо менше – до другого. Відомо, що вектор $\boldsymbol{w}$ перпендикулярний до розділяючої прямої, яка має бути знайдена, а значення b залежить від найкоротшої відстані між розділяючою прямою і початком координат. Таким чином, якщо існує розділяюча пряма, то вона не єдина. Виникає питання, яка з прямих розділяє класи найкраще?

Метод SVM базується на такому постулаті: найкраща розділяє пряма – це та, яка знаходиться максимально далеко від найближчих до неї точок обох класів. Тобто завдання методу SVM полягає в тому, щоб знайти такі вектор w і число b, щоб для деякого $\varepsilon > 0$ (половини ширини розділяючої поверхні) виконувалося:

$$\begin{cases} \boldsymbol{w} \cdot \boldsymbol{x}_i \geq b + \varepsilon \;\Rightarrow\; y_i = +1, \\ \boldsymbol{w} \cdot \boldsymbol{x}_i \leq b - \varepsilon \;\Rightarrow\; y_i = -1. \end{cases}$$

Помножимо після цього обидві частини нерівності на $1/\varepsilon$ і, не обмежуючи спільності, виберемо ε рівним одиниці. Таким чином, для всіх векторів x_i з навчальної колекції буде справедливо:

$$\begin{cases} w \cdot x_i - b \geq +1, \text{ якщо } y_i = +1, \\ w \cdot x_i - b \leq -1, \text{ якщо } y_i = -1. \end{cases}$$

Умова $-1 < w \cdot x_i - b < 1$ задає смугу, яка розділяє класи. Межами смуги є дві паралельні гіперплощини з направляючим вектором $\boldsymbol{w}$. Точки, найближчі до розділяючої гіперплощини, розташовані на межах смуги.

Чим ширше смуга, тим впевненіше можна класифікувати документи; відповідно, в методі SVM передбачається, що найширша смуга є найкращою.

Сформулюємо умови задачі оптимальної розділяючої смуги, що визначається нерівністю: $y_i(w \cdot x_i - b) \geq 1$ (таким чином переписується система рівнянь, виходячи з того, що $y_i \in \{-1, 1\}$). Жодна з точок навчальної вибірки не може лежати всередині цієї розділяє смуги. При цих обмеженнях x_i і y_i – постійні, як елементи навчальної колекції, а w і b – змінні.

З геометричних міркувань відомо, що ширина розділяючої смуги дорівнює $2/\|w\|$. Тому необхідно знайти такі значення w і b, щоб виконувалися наведені лінійні обмеження, і при цьому якомога менше була норма вектора, тобто необхідно мінімізувати:

$$\|w\|^2 = w \cdot w.$$

Це відома задача квадратичної оптимізації при лінійних обмеженнях.

Якщо припустити, що на навчальних документах можливо були допущені помилки експертами при класифікації, то необхідно ввести набір додаткових змінних $\xi_i \geq 0$, що характеризують величину помилок на об'єктах $\{x_1, ..., x_n\}$. Це дозволяє пом'якшити обмеження: $y_i(w \cdot x_i - b) \geq 1 - \xi_i$.

Передбачається, що якщо $\xi_i = 0$, то на документі x_i помилки немає. Якщо $\xi_i > 1$, то на документі x_i допускається помилка. Якщо $0 < \xi_i < 1$, то об'єкт потрапляє всередину розділяє смуги, але відноситься алгоритмом до свого класу.

Завдання пошуку оптимальної розділяючої смуги можна в цьому випадку переформулювати наступним чином: при певних обмеженнях мінімізувати суму:

$$\|w\|^2 + C\sum_i \xi_i.$$

Коефіцієнт C – це параметр настройки методу, який дозволяє регулювати співвідношення між максимізацією ширини розділяючої смуги і мінімізацією сумарної помилки. Наведена задача залишилася задачею квадратичного програмування, яку можна переписати у наступному вигляді:

$$\begin{cases} \dfrac{\|w\|^2}{2} + C\sum_i \xi_i \to \min; \\ y_i(w \cdot x_i - b) + \xi_i \geq 1, \quad i = 1, \ ..., \ n. \end{cases}$$

За відомою теоремою Куна-Такера така задача еквівалентна двоїстій задачі пошуку сідлової точки функції Лагранжа:

$$\begin{cases} \dfrac{1}{2} w \cdot w + C\sum_i \xi_i - \sum_i \lambda_i(\xi_i + y_i(w \cdot x_i - b) - 1) \to \min\limits_{w,b} \ \max\limits_{\lambda}; \\ \xi_i \geq 0, \quad \lambda_i \geq 0, \quad i = 1, \ ..., \ n. \end{cases}$$

Необхідною умовою методу Лагранжа є рівність нулю похідних лагранжіана за змінними w і b, звідки отримуємо:

$$w = \sum_{i=1} \lambda_i y_i x_i,$$

тобто вектор, який необхідно визначити, – це лінійна комбінація навчальних векторів, для яких $\lambda_i \neq 0$. Якщо $\lambda_i > 0$, то документ навчальної колекції називається опорним вектором.

Таким чином, рівняння розділяючої гіперплощини має вигляд:

$$\sum_{i=1} \lambda_i y_i x_i \cdot x - b = 0.$$

Прирівнявши похідну лагранжіана по b нулю, отримаємо:

$$\sum_{i=1} \lambda_i y_i = 0.$$

Підставляючи останній вираз і вираз для w у лагранжіан отримаємо еквівалентну задачу квадратичного програмування, що містить тільки двоїсті змінні:

$$\begin{cases} \sum_i \lambda_i - \frac{1}{2} \sum_{i,j} \lambda_i \lambda_j y_i y_j (x_i \cdot x_j) \to \min\limits_{\lambda}; \\ \sum_{i=1} \lambda_i y_i = 0; \\ C \geq \lambda_i \geq 0, \quad i = 1, \ ..., \ n. \end{cases}$$

При цьому дуже важливо, що цільова функція залежить не від конкретних значень x_i, а від їх скалярних добутків. Слід зауважити, що цільова функція є опуклою, тому будь-який її локальний мінімум є глобальним.

Метод класифікації розділяючою смугою має два недоліки:

• при пошуку розділяючої смуги важливе значення мають лише прикордонні точки;

• у багатьох випадках знайти оптимальну розділяючу смугу неможливо.

Для поліпшення методу застосовується ідея розширеного простору, для чого:

- Обирається відображення $\phi(x)$ векторів $\boldsymbol{x}$ в новий, розширений простір.
- Автоматично застосовується нова функція скалярного добутку, яка застосовується при вирішенні задачі квадратичного програмування, так звана функція ядра (Kernel Function): $K(x,y)=\phi(x)\cdot\phi(y)$. На практиці зазвичай вибирають не відображення $\phi(x)$, а відразу функцію $K(x,y)$, яка могла б бути скалярним твором при деякому відображенні $\phi(x)$. Функція ядра – головний параметр настроювання машини опорних векторів.
- Знаходимо розділяючу гіперплощину в новому просторі: за допомогою функції $K(x,y)$ встановлюється нова матриця коефіцієнтів для задачі оптимізації. При цьому замість $x_i \cdot x_j$ підставляються значення $K(x_i,x_j)$ і вирішується нове завдання оптимізації.
- Знайшовши $\boldsymbol{w}$ і b, отримуємо поверхню, яка класифікує $w\cdot\phi(x)-b$ в новому, розширеному просторі.

Як приклад, в системі класифікації новинного контенту з застосуванням відомого пакету LibSVM (http://www.csie.ntu.edu.tw/~cjlin/libsvm) в якості опції ядра рекомендується використовувати радіальну базисну функцію:

$$K(x,y)=\exp(-\gamma\|x-y\|^2),$$

де γ - параметр, що може бути налагоджений.

Питання для самоперевірки

1. В чому полягає задача класифікації?
2. Наведіть формальний опис класифікації.
3. В чому полягає метод Rocchio?
4. В чому полягає метод лінійної регресії?
5. На чому базується Байєсовська логістична регресія?
6. В чому полягає метод опорних векторів?

Література до розділу

1. *Потапов А.С.* Технологии искусственного интеллекта – СПб: СПбГУ ИТМО, 2010. – 218 с. **2.** *Рассел С., Норвиг П.* Искусственный интеллект. Современный подход. – М.: Вильямс, – 2006. – 1408 с. **3.** *Ландэ Д.В., Снарский А.А., Безсуднов И.В.* Интернетика: Навигация в сложных сетях: модели и алгоритмы. – М.: Либроком (Editorial URSS), 2009. – 264 с. **4.** *Додонов А.Г., Ландэ Д.В., Путятин В.Г.* Компьютерные сети и аналитические исследования. – К.: ИПРИ НАН Украины, 2014. – 486 с. **5**. *Барсегян А.А.* Методы и модели анализа данных: OLAP и Data Mining / Барсегян А.А., Куприянов М.С., Степаненко В.В., Холод И.И. – Спб.: БХВ-Петербург, 2004. – 336 с.

4. ЕЛЕМЕНТИ КЛАСТЕРНОГО АНАЛІЗУ

4.1. Визначення кластерного аналізу

Механізм класифікації зазвичай навчається на відібраних об'єктах (надалі, як синонім буде застосовуватися термін «документ») тільки після того, як закінчується стадія навчання шляхом автоматичної кластеризації – розбиття множини документів на класи (кластери), смислові параметри яких заздалегідь невідомі. Кількість кластерів може бути довільним або фіксованим. Якщо класифікація допускає приписування об'єктам певних, відомих заздалегідь ознак, то кластеризація складніший процес, який допускає не тільки приписування документам деяких ознак, але і виявлення цих ознак, як основ формування класів. Мета методів кластеризації полягає в тому, щоб подібність об'єктів, які потрапляють в кластер, було максимальним. Тому методи кластерного аналізу базуються на таких визначеннях кластера, як множини об'єктів, значення деякої міри близькості (зокрема, семантичної) між будь-якими двома елементами яких (або значення близькості між будь-яким документом цієї множини і центром кластера) не менше певного порогу.

Для чисельного визначення значення близькості між об'єктами у кластерному аналізі використовуються такі основні правила визначення відстані (метрики), як метрика Мінковського:

$$D_p(\vec{x},\vec{y}) = \left(\sum_{k=1}^{N} (x_k - y_k)^p \right)^{1/p},$$

окремим випадком для $p = 2$ є Евклідова метрика:

$$D_p(\vec{x},\vec{y}) = \sqrt{\sum_{k=1}^{N} (x_k - y_k)^2}.$$

Для групування документів, представлених у вигляді векторів вагових значень що входять в них термів, часто використовується скалярний добуток вагових векторів:

$$Sim(\vec{x}, \vec{y}) = \vec{x} \cdot \vec{y},$$

де $\vec{x}$, $\vec{y}$ – вектори, що відповідають документам, елементи яких – вагові значення термів, що визначаються в результаті аналізу великого масиву документів. Для проведення такого аналізу використовуються різні підходи – ваговий, ймовірнісний, семантичний і т. д.

В області інформаційного пошуку кластерний аналіз найчастіше застосовується для вирішення двох завдань – групування документів у базах даних (інформаційних масивах) і групування результатів пошуку.

Для статичних масивів документів методи кластерного аналізу на цей час набули значного розвитку. Разом з тим відкритим залишається питання застосування цих методів до інформаційних потоків, яким притаманні великі обсяги і динаміка.

Методи кластерного аналізу знаходять широке застосування в процедурах ранжирування відгуків інформаційно-пошукових систем, при побудові персоналізованих пошукових інтерфейсів і папок пошуку.

4.2. Методи латентного семантичного індексування (LSA)

Метод кластерного аналізу LSA/LSI (Latent Semantic Analysis / Indexing – метод латентно-семантичного аналізу / індексування) базується на сингулярному розкладанні матриць (SVD).

Нехай масиву документів $D = \{d^{(j)} \mid j = 1, \ ..., \ n\}$ ставиться у відповідність матриця A, рядки якої відповідають документам, а стовпці – ваговим значенням умов (розмір словника термів - m). Сингулярним розкладом матриці A рангу r розмірності $n \times m$ називається її розкладання виду $A = USV^T$, де U і V ортогональні матриці розмірності $m \times r$ і $r \times n$, відповідно, а S діагональна матриця, діагональні елементи якої невід'ємні ($s_{i,i} \geq 0$). Діагональні елементи матриці називають сингулярними значеннями матриці.

Наведене вище розбиття матриці A має ту властивість, що якщо в матриці S залишити тільки k найбільших сингулярних значень (позначимо таку матрицю як S_k), а в матрицях U і V тільки відповідні цим значенням колонки (відповідно, матриці U_k, V_k), то матриця $A_k = U_k S_k V_k^T$ буде найкращою за Фробеніусом апроксимацією вихідної матриці $\mathbf{A}$ матрицею з рангом, що не перевищує k. Нагадаємо, що норма матриці X розмірності $N \times M$ за Фробеніусом визначається виразом:

$$\|X\|_F = \sqrt{\sum_{i=1}^{M}\sum_{j=1}^{N} x_{ij}^2}.$$

Вказану вище властивість можно сформулювати таким чином: A_k буде той матрицєй рангу k, яка мінімізує норму матриці $\|A - A_k\|_F$, що можна записати в визначеннях, які застосовуються в методах оптимизації:

$$A_k = \underset{X:rank(X)=k}{\arg\min} \|A - X\|_F.$$

Відповідно до методу LSA в розгляд беруться не всі, а лише k найбільших сингулярних значень матриці A, і кожному такому значенню ставиться у відповідність один кластер.

A_k визначає k- мірний факторний простір, на який проектуються як документи (за допомогою матриці V), так і терміни (за допомогою матриці U). В отриманому факторному просторі документи і терміни групуються в масиви (кластери), які мають деякий загальний зміст, що не заданий в явному вигляді, тобто латентний.

Вибір найкращого значення k для LSA – це проблема окремих досліджень. В ідеалі, k має бути досить велике для відображення всієї реально існуючої структури даних, але в той же час досить мало, щоб не враховувати випадкових залежностей.

У практиці інформаційного пошуку особливе значення відводиться матрицями U_k і V_k^T. Як зазначалося раніше, рядки матриці U_k

розглядаються як образи термів в $k-$вимірному дійсному просторі. Аналогічно, стовпці матриці V_k^T розглядаються як образи документів в тому ж $k-$вимірному просторі. Іншими словами, ці вектори задають вихідне подання термів і документів в $k-$вимірному просторі прихованих факторів.

Існують також методи інкрементного оновлення всіх значень, які використовуються в LSA. При поповненні новим документом d (наприклад, новим результатом пошуку за запитом) інформаційного масиву, для якого вже проведено сингулярне розкладання, можна не виконувати розкладання заново. Досить апроксимувати його, обчислюючи образ нового документа на основі раніше обчислених образів термів і ваг факторів. Нехай d – вектор ваг термів нового документа (новий стовпець матриці A), тоді його образ можна обчислити за формулою:

$$d' = S_k^{-1} U_k^T d.$$

Якщо q – вектор запиту, i -й елемент якого дорівнює 1, коли терм з номером i входить в запит, і 0 – в іншому випадку, то образ запиту q в просторі латентних факторів буде мати вигляд:

$$q' = q^T U_k \, S_k^{-1}.$$

В цьому випадку міра близькості запиту q і документа d оцінюється величиною скалярного добутку векторів q'і $V_k^T\{d\}$ ($V_k^T\{d\}$ тут позначає d-й стовпець матриці V_k^T).

При інформаційному пошуку, в результаті того, що відкидаються найменш значущі сингулярні значення, формується простір ортогональних факторів, що грають роль узагальнених термів. В результаті відбувається «зближення» документів з близьких за змістом предметних областей, частково вирішуються проблеми синонімії і омонімії термів.

Метод LSA широко застосовується при ранжируванні видачі інформаційно-пошукових систем, заснованих на цитуванні. Це алгоритм

HITS (Hyperlink Induced Topic Search) – один з двох найвідоміших в області інформаційного пошуку. Метод LSA не потребує попереднього налаштування на специфічний набір документів, разом з тим дозволяє якісно виявляти приховані фактори.

До недоліків методу можна віднести невисоку продуктивність. Швидкість обчислення SVD відповідає порядку $O\left(N^2 \times k\right)$, де $N = |D| + |T|$, D – множина документів, T – множина термів, k – розмірність простору факторів.

LSA також не передбачає можливість перетину кластерів, що суперечить практиці. Крім того, зважаючи на свою обчислювальну трудомісткість метод LSA застосовується тільки для відносно невеликих матриць.

4.3. Ймовірнісний латентно-семантичний аналіз

Ймовірнісний латентно-семантичний аналіз (Probabilitstic Latent Semantic Analysis, PLSA) – це модифікація LSA, побудована на використанні імовірнісного підходу. Метод PLSA також призначений для виявлення прихованих чинників, присутніх в інформаційному масиві і пов'язаних з ними документів і слів.

Як і в попередньому випадку, передбачається, що існує k прихованих факторів $z_1, \ldots, z_k$ (число k задається заздалегідь). Фактору z_i зіставляється ймовірність $P(z_i)$ того, що випадково обраний з даної колекції документ найбільш точно характеризується даним фактором $\sum_{i=1}^{k} P(z_i) = 1$.

Позначимо $P(d|z_i)$ ймовірність того, що для обраного фактору z_i з множини фактів Z, саме документ d зі всієї множини документів D найкраще характеризується цим фактором. Тоді $\sum_{d \in D} P(d|z_i) = 1$. Аналогічно позначимо $P(t|z_i)-$ ймовірність того, що для обраного

фактора z_i, з усіх термів саме терм t зі словника системи T найкраще характеризується цим фактором z_i. Тоді $\sum_{t \in T} P(t|z_i) = 1$.

Імовірність того, що при випадковому виборі документа d і терма t, терм t зустрінеться в документі d, можна визначити з одного боку (рис. 4.1 а, асиметрична параметризація), як:

$$P(d,t) = P(d)P(t \mid d),$$

$$P(t \mid d) = \sum_{i=1}^{k} P(t \mid z_i) P(z_i \mid d).$$

З іншого боку, ця ж ймовірність при симетричній параметризації представляється (рис. 5.1 б) як:

$$P(d,t) = \sum_{i=1}^{k} P(z_i) P(d \mid z_i) P(t \mid z_i).$$

а) *б*)

Рис. 4.1. Графічне представлення моделі: а) – асиметрична; б) – симетрична параметризація

Зафіксувавши число прихованих факторів k, за допомогою методу PLSA можна оцінити такі величини:

$P(z_i)$ ймовірність того, що випадково обраний з колекції документ в відповідає фактору z_i;

$P(d_j \mid z_i)$ – ймовірність того, що документ d_j потрапить в групу документів z_i,

$P(t_j \mid z_i)$ – ймовірність того, що терм t_j потрапить в групу слів, зв'язаних з фактором z_i .

Для оцінки наведених вище ймовірностей на контрольному масиві документів визначається спостережувана частота входження терма t в документ d, традиційно позначається як $tf(d,t)$.

Згадані вище ймовірності визначаються виходячи з умови максимізації функції максимальної правдоподібності:

$$L = \sum_{d \in D} \sum_{t \in T} tf(d,t) \log P(d,t),$$

де зовнішня сума береться по всім документам, а внутрішня по всьому термам словника.

У PLSA використовується алгоритм EM (Expectation Maximization – оціночної максимізації), в якому на кожному кроці виконуються два кроки – 1-й оцінювання, при якому обчислюються і оцінюються післядосвідні ймовірності латентних змінних, і 2-й максимізація, в результаті якої параметри змінюються.

На першому кроці оцінюється:

$$P(z \mid d,t) = \frac{P(z)P(d \mid z)P(t \mid z)}{\sum_{z' \in Z} P(z')P(d \mid z')P(t \mid z')},$$

Після чого виконується крок максимізації L на базі обчислень:

$$P(t \mid z) \propto \sum_{d \in D} tf(d,t)P(z \mid d,t),$$
$$P(d \mid z) \propto \sum_{t \in T} tf(d,t)P(z \mid d,t),$$
$$P(z) \propto \sum_{d \in D} \sum_{t \in T} tf(d,t)P(z \mid d,t).$$

Даний алгоритм забезпечує збіжність функції L до деякого локального максимуму. Експерименти показують, що збіжність досягається після декількох десятків ітерацій.

Покажемо, як представити PLSA у вигляді матричної записи.

Визначимо матриці:

1) $\hat{U}$, елементами якої $\hat{u}_{i,k}$ будуть умовні ймовірності $P(d^{(i)} \mid z_k)$,

2) $\hat{V}$, елементами якої $\hat{v}_{j,k}$ будуть умовні ймовірності $P(t_j \mid z_k)$,

3) $\hat{S}$ – діагональну матрицю рангу k, на діагоналі якої будуть розміщені значення ймовірностей $P(z_i)$. Об'єднана імовірнісна модель $P(z_i)$ апроксимується виразом $\hat{U}\hat{S}\hat{V}^T$. Порівнюючи це розкладання з SVD, можна помітити, що $\hat{U}$ і $\hat{V}$ в PLSA також незалежні зважаючи припущення незалежності термів і документів.

Хоча наведене розкладання не є сингулярним, разом з тим, k найбільших компонент $\hat{S}$ визначають правила кластеризації PLSA. Основна відмінність PLSA від LSA полягає у виборі цільової функції L.

4.4. Метод k-means

Ітеративний алгоритм кластерного аналізу k-means (k-середніх) групування документів $\{\vec{d}^{(1)}, ..., \vec{d}^{(N)}\}$ (як і в векторно-просторової моделі інформаційного пошуку документи являють собою вектори, координати яких – вагові значення термів) за фіксованою кількістю кластерів полягає у наступному: випадковим чином вибирається k документів, які визначаються як центроїди (найбільш типові представники) кластерів. Тобто кожен кластер C_j ($j = 1, ..., k$) представлений вектором $\vec{C}_j$, відповідним центроїду. Близькість документів до центроїду може визначатися різними способами, наприклад, як нормований скалярний добуток:

$$Sim(\vec{d}, \vec{C}_j) = \frac{\vec{d} \cdot \vec{C}_j}{|\vec{d}||\vec{C}_j|}.$$

Після цього документ приписується до того кластеру, значення $Sim(\vec{d}, \vec{C}_j)$ для якого виявляється найбільшим. Далі для кожного з нових

кластерів заново визначається центроїд $\vec{C}_j$ $(j = 1, ..., k)$ – вектор, координати якого визначаються, наприклад, як середнє арифметичне відповідних вагових документів, що входять в даний кластер.

Після цього знову здійснюється процес наповнення кластерів, потім обчислення нових центроїдів і т.д., поки процес формування кластерів не стабілізується (або, якщо зменшення суми відстані від кожного елемента до центру його кластера не стане менше деякого заданого порогового значення).

Алгоритм *k*-means максимізує функцію якості кластеризації Q:

$$Q(C_1, ..., C_k) = \sum_{j=1}^{k} \sum_{\vec{d} \in C_j} Sim(\vec{d}, \vec{C}_j).$$

На відміну від методу LSI, ***k*** -means може використовуватися для групування динамічних інформаційних потоків завдяки своїй обчислювальної простоті – $O(kn)$, де n – кількість об'єктів групування (документів). Недоліком методу є те, що кожен документ може потрапити лише в один кластер.

Практичне застосування методу k-means

Існують дві реалізації алгоритму *k*-means, «жорсткий», коли число k фіксоване та «м'який», що дозволяє на підставі деяких критеріїв оцінити значення k. Сутність жорсткого алгоритму *k*-means визначається таким чином: випадковим образом вибирається k векторів-рядків, які визначаються як центроїди (найбільш типові представники) кластерів. Потім k кластерів наповнюються – для кожного з векторів-рядків, що залишилися, визначається близькість до центроїду відповідного кластера. Після цього вектор-рядок приписується до того кластера, до якого він найбільш близький. Далі рядки-вектори групуються та перенумеровуються відповідно до отриманого групування. Потім для кожного з нових кластерів заново визначається центроїд – вектор-рядок, найбільш близький до всіх векторів з даного кластера (наприклад, той,

сума скалярних добутків якого з кожним з векторів кластера – максимальна). Після цього заново здійснюється процес наповнення кластерів, потім обчислення нових центроїдів і т.д., поки процес формування кластерів не стабілізується (або набір центроїдів не повториться).

У системі Matlab існує спеціальна процедура, що реалізує метод *k*-means – kmeans. Нижче наведено фрагмент програмного коду, де здійснюється генерація випадкових точок на площині, їх кластеризація і відображення кластерів на площині.

```
clear
hold off
X=rand(1,20)
X=X'
Y=rand(1,20)
Y=Y'
Z=[X,Y]
plot(X,Y,'o')
[IDX,C]=kmeans(Z,3)
hold on
for i=1:20
   if  IDX(i)==1
      plot(X(i),Y(i),'r+')
   end
   if IDX(i)==2
      plot(X(i),Y(i),'g+')
   end
end
for i=1:3
   plot(C(i,1),C(i,2),'x')
end
```

4.5. Ієрархічне групування-об'єднання

Ієрархічне групування-об'єднання (Hierarchical Agglomerative Clustering, HAC) починається з того, що кожному об'єкту у відповідність ставиться окремий кластер, а потім відбувається об'єднання кластерів, які найбільш близькі один до одного, відповідно до обраного критерію. Алгоритм завершується, коли всі об'єкти групуються в єдиний кластер. Історія об'єднань утворює бінарне дерево ієрархії кластерів.

Різновиди алгоритму HAC розрізняються вибором критеріїв близькості між кластерами. Наприклад, близькість між двома кластерами може обчислюватися як максимальна близькість між об'єктами з цих кластерів.

Можуть використовуватися також і інші міри близькості, наприклад, близькість «центрів мас», середня близькість між усіма парами об'єктів в об'єднаних кластерах і т.п. Міра близькості між двома кластерами C_i і C_j у останньому випадку обчислюється за формулою:

$$Sim(C_i,\ C_j) = \frac{1}{\left|C_i \bigcup C_j\right|(\left|C_i \bigcup C_j\right| - 1)} \sum_{x,y \in C_i \bigcup C_j, x \neq y} Sim(x,\ y).$$

У виразі $|C_i \bigcup C_j|$ – кількість об‘єктів в множині $C_i \bigcup C_j$, а x і y – об‘єкти, що належать $C_i \bigcup C_j$.

Складність алгоритму HAC становить $O(n^2 s)$, де n – кількість об'єктів, а s – складність обчислення близькості між кластерами.

Для практичного вивчення методів кластеризації, реалізованих у системі Matlab, зупинимося на таких функціях із Statistics Toolbox, як pdist, linkage, cluster, dendrogram.

Функція pdist – розрахунок парних відстаней.

Розглянемо основні формати функції pdist:

Y = pdist(X)

Y = pdist(X,'metric')

Функція pdist(X) дозволяє розрахувати вектор відстаней (за замовченням Евкліда) Y між парами об'єктів початкової великої кількості даних, заданих матрицею X. Розмірність матриці X дорівнює $m \times n$, де m – число спостережень n-вимірної випадкової величини. Кількість пар відстаней для m спостережень багатовимірної випадкової величини – параметр Y є вектором, що містить значення парних відстаней між об'єктами. Число елементів Y дорівнює $\frac{m(m-1)}{2}$.

За допомогою функції squareform вектор Y можна конвертувати у квадратну матрицю. Елемент (i, j) отриманої матриці, при $i < j$, відповідатиме відстані між i-м та j-м об'єктами початкової множини даних.

У виразі Y = pdist(X', metric') вхідний параметр 'metric' визначає вид відстані між об'єктами початкової множини даних. 'metric' задається як строкова змінна.

Передбачені наступні види відстані між об'єктами:

Значення параметра 'metric'	Метод розрахунку відстані між об'єктами
'euclidean'	Відстань Евкліда. Значення за умовчанням
'minkowski'	Метрика Мінковського
'cosine'	Косинусна відстань. Визначається як одиниця мінус косинус від кута між об'єктами. Об'єкти в багатовимірному просторі розглядаються як вектори
'hamming'	Відстань Хеммінга. Визначається як відсоток відмінних координат від їх загального числа

Наведемо деякі команди Matlab, які будемо застосовувати для вирішення завдання кластеризації:

Генерування матриці нормально розподілених випадкових чисел:

```
>> X=normrnd(0,1,5,3)
```

Результат:

```
X =
   0.1139    0.2944    0.8580
   1.0668   -1.3362    1.2540
   0.0593    0.7143   -1.5937
  -0.0956    1.6236   -1.4410
  -0.8323   -0.6918    0.5711
```

Розрахунок Евклідової відстані:

```
>> Y = pdist(X)
```

Результат:

```
Y =
  Columns 1 through 8
    1.9297    2.4880    2.6638    1.3965    3.6509    4.1682    2.1185    0.9349
  Columns 9 through 10
    2.7311    3.1547
```

Формування квадратної (5x5) матриці зв'язків:

```
>> S = squareform(Y)
```

Результат:

```
S =
   0         1.9297    2.4880    2.6638    1.3965
   1.9297         0    3.6509    4.1682    2.1185
   2.4880    3.6509         0    0.9349    2.7311
   2.6638    4.1682    0.9349         0    3.1547
   1.3965    2.1185    2.7311    3.1547         0
```

Функція linkage – формування ієрархічного дерева кластерів

Основні формати:

```
Z = linkage(Y)
Z = linkage(Y,'method')
```

Ця функція дозволяє сформувати ієрархічне дерево бінарних кластерів з використанням алгоритму "найближчого сусіда". Вхідний аргумент Y є вектором відстаней між парами об'єктів початкової множини даних. Число елементів вектора Y дорівнює $\frac{m(m-1)}{2}$, де m – кількість об'єктів в початковій множині даних. Y може бути отриманий як вихідний параметр функції pdist.

Вихідний параметр Z є матрицею, що містить інформацію про дерево кластерів. Розмірність Z дорівнює $3(m-1)$.

Кінцеві вузли дерева кластерів – це об'єкти початкової множини спостережень багатовимірної випадкової величини Y, пронумерованих від 1 до m. Кінцеві вузли є одиничними кластерами. Вони об'єднуються в кластери, що відповідають розміщеним вище вузлам дерева.

Стовпці 1 і 2 матриці Z містять індекси об'єктів, пов'язаних в новий кластер. Кількість сформованих бінарних кластерів буде дорівнювати (m – 1). 3-й стовпець матриці Z містить значення відстаней між парами об'єктів, об'єднаних в кластери.

Z = linkage(Y', method') вхідний аргумент 'method' дозволяє задати алгоритм кластеризації. Значення вхідного аргументу 'method' задається як тестовий рядок. Передбачені наступні алгоритми кластеризації:

Метод	Назва алгоритму
'single'	Алгоритм "найближчого сусіда". Заснований на визначенні найменшої відстані між об'єктами у двох групах. Значення за умовчанням
'complete'	Алгоритм "далекого сусіда". Заснований на визначенні найбільшої відстані між об'єктами у двох групах
'average'	Алгоритм "середнього зв'язку". Заснований на розрахунку відстаней між усіма можливими парами об'єктів в кластерах
'centroid'	Центроїдний алгоритм, що використовує відстань за "центрам тяжіння" вибірок. Відстань між центроїдами кластерів
'ward'	Покроковий алгоритм. Відстань між кластерами, визначається за центроїдним алгоритмом. Алгоритм заснований на збільшенні загальної внутрішньогрупової суми квадратів в

	результаті приєднання груп

Формування дерева кластерів засноване на послідовному об'єднанні пар вузлів нижчого рівня у вузли вищого рівня.

Формування ієрархічних дерев

Приклад 1. Формування ієрархічного дерева бінарних кластерів для тривимірної нормально розподіленої випадкової величини. Кількість об'єктів у множині початкових даних дорівнює 5. Графічне представлення дерева бінарних кластерів виконується за допомогою функції dendrogram (рис. 4.2).

```
>> X=normrnd(0,1,5,3);
>> Y = pdist(X);
>> Z = linkage(Y)
  Z =
   3.0000    4.0000    0.9349
   1.0000    5.0000    1.3965
   2.0000    7.0000    1.9297
   6.0000    8.0000    2.4880
>> H = dendrogram(Z);
```

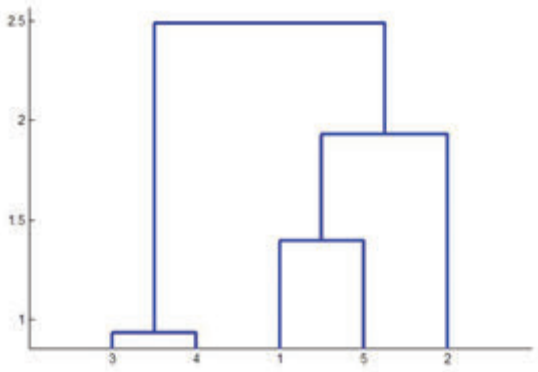


Рис. 4.2 – Дерево бінарних кластерів

Приклад 2. Формування ієрархічного дерева бінарних кластерів для двомірної матриці. Кількість об'єктів у множині початкових даних дорівнює 7. Порівнюються різні алгоритми кластеризації. Графічне представлення результатів кластеризації (рис. 4.3) виконується за допомогою функції dendrogram.

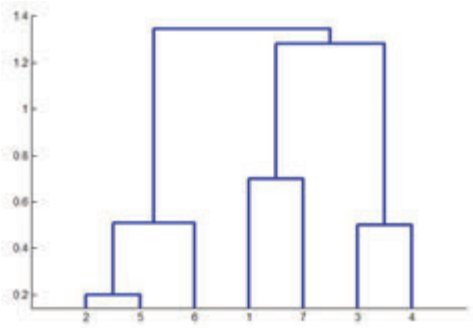


Рис. 4.3 – Дерево бінарних кластерів для двовимірної випадкової величини

Початкові дані:

```
>> X = [3 1.7; 1 1; 2 3; 2 2.5; 1.2 1; 1.1 1.5; 3 1]

X =

   3.0000    1.7000
   1.0000    1.0000
   2.0000    3.0000
   2.0000    2.5000
   1.2000    1.0000
   1.1000    1.5000
   3.0000    1.0000

>> Y = pdist(X);

Y =

  Columns 1 through 11
   2.1190    1.6401    1.2806    1.9313    1.9105    0.7000    2.2361    1.8028
0.2000    0.5099    2.0000
  Columns 12 through 21
   0.5000    2.1541    1.7493    2.2361    1.7000    1.3454    1.8028    0.5099
1.8000    1.9647
>> Z = linkage(Y, 'single')

Z =
```

```
    2.0000    5.0000    0.2000
    3.0000    4.0000    0.5000
    6.0000    8.0000    0.5099
    1.0000    7.0000    0.7000
    9.0000   11.0000    1.2806
   10.0000   12.0000    1.3454
```

```
>> dendrogram(Z);
```

Кластеризація за допомогою центроїдного алгоритму (Рис. 4.4)

```
>> Z = linkage(Y, 'centroid')

Z =

    2.0000    5.0000    0.2000
    6.0000    8.0000    0.5000
    3.0000    4.0000    0.5000
    1.0000    7.0000    0.7000
   10.0000   11.0000    1.7205
    9.0000   12.0000    1.6554

>> H = dendrogram(Z)

H =

  158.0043
  160.0043
  161.0043
  162.0043
  163.0043
  164.0043
```

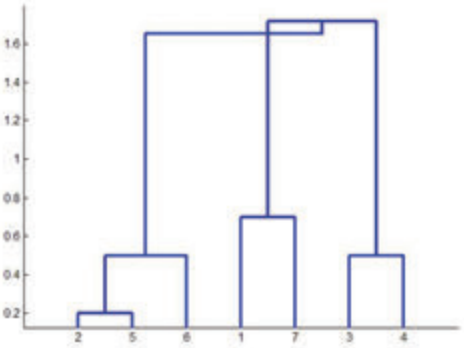


Рис. 4.4 – Результати застосування центроїдного алгоритму

Функція cluster – кластеризація по виходу функції linkage

Розглядаються формати:

T = cluster(Z,'cutoff',c)

T = cluster(Z,'maxclust',n)

Ця функція призначена для розподілу на окремі кластери T ієрархічного дерева кластерів Z, отриманого за допомогою функції linkage. Вхідний параметр Z є матрицею розмірності $3(m-1)$, де m – кількість спостережень багатовимірної випадкової величини початкової множини даних. Вхідний параметр C – порогова величина призначена для розбиття ієрархічного дерева Z на кластери. Кластер формується шляхом виключення зв'язків ієрархічного дерева кластерів для яких значення коефіцієнтів несумісності менше величини C. Вихідний параметр T є вектором номерів кластерів, до яких віднесені об'єкти початкової множини даних. Число елементів вектора T дорівнює m.

T = cluster(Z', maxclust ', n): вхідний параметр n визначає максимальну кількість кластерів, на яку ділиться ієрархічне дерево кластерів Z.

Деякі приклади застосування функції кластеризації об'єктів

Приклад 1. Як початкові дані використовується 5-ти вимірна випадкова величина X. Розподіл X є композицією нормального закону і закону Вейбулла. Порівнюється розподіл об'єктів за кластерами для порогової величини, що дорівнює 0.1*(max(Z(:,3))), 0.3*(max(Z(:,3))), 0.4*(max(Z(:,3))). Графічне представлення ієрархічного дерева кластерів (рис. 4.5) виконується за допомогою функції dendrogram.

```
>> X=normrnd(0,1,15,5)+weibrnd(1,2,15,5);
>> Y = pdist(X);
>> Z = linkage(Y);
>> t1=0.1*(max(Z(:,3)));
>> t2=0.3*(max(Z(:,3)));
>> t3=0.4*(max(Z(:,3)));
```

```
>> dendrogram(Z);
```

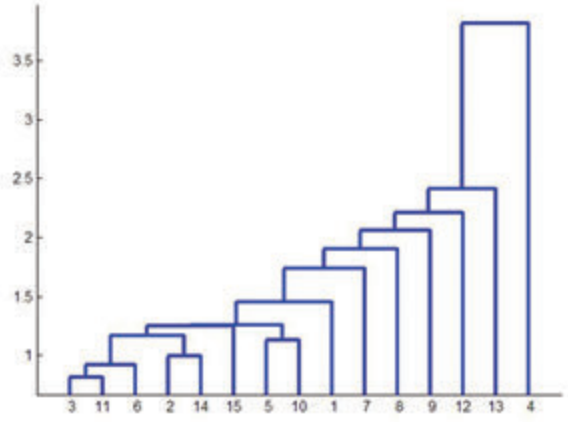


Рис. 4.5 – Ієрархічне дерево кластерів

```
>> T1 = cluster(Z,'cutoff', t1);
>> T2 = cluster(Z,'cutoff', t2);
>> T3 = cluster(Z,'cutoff', t3);
>> cat(2,T1,T2,T3)
ans =
    4    1    1
   12    1    1
   11    1    1
   10    1    1
    3    1    1
    1    1    1
    5    1    1
    6    1    1
    7    1    1
    3    1    1
   11    1    1
    8    1    1
    9    1    1
   12    1    1
    2    1    1
```

Приклад 2. В якості початкових даних використовується 10-ти вимірна випадкова величина X. Об'єм вибірки дорівнює 25 елементам. X

розподілена за нормальним законом. Графічне представлення ієрархічного дерева кластерів (рис. 4.6) виконується за допомогою функції dendrogram. Максимальне число кластерів прийняте рівним 5.

```
>> X=normrnd(0,1,25,10);
>> Y = pdist(X);
>> Z = linkage(Y);
>> dendrogram(Z);
```

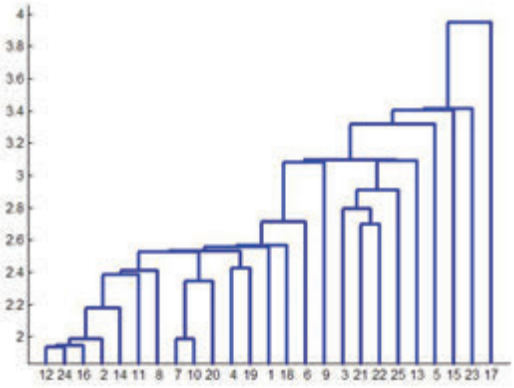


Рис. 4.6 – Ієрархічне дерево кластерів

```
>> T = cluster(Z,'maxclust', 5);
>> T'
ans =
    2 2 2 2 1 2 2 2 2 2 2 2 2 2 3 2 5 2 2 2 2 2 4 2 2
```

4.6. Метод суфіксних дерев

Метод суфіксних дерев (Suffix Tree Clustering) був розроблений для швидкого пошуку підрядків в рядках.

Суфікс W рядку S – це такий рядок, в якому конкатенація (зчеплення рядків) VW збігається з S для деякого (можливо, порожнього) рядка. Суфікс називається власним, якщо $|V| \neq 0$. Наприклад, для рядка «substring» підрядок «sub» є власним префіксом, «ring» – власним суфіксом. Рядок V називається при цьому префіксом.

Суфіксне дерево – це дерево, що містить всі суфікси рядки. Воно складається з вершин, гілок і суфіксних покажчиків (міток). Мітка вузла в дереві визначається як конкатенація підрядків, що маркують ребра шляху від кореня дерева до цього вузла.

Існують алгоритми, наприклад, алгоритм Уконена (E. Ukkonen), що реалізують побудову суфіксних дерев за $O(n)$ кроків, де n – довжина рядка. Гілки дерева позначаються окремими буквами або частинами суфіксів рядка. Суфікс, відповідний певній вершині, можна отримати шляхом об'єднання букв, які знаходяться на гілках, починаючи від кореневої вершини і закінчуючи поточною.

Розглянемо найпростіший варіант побудови суфіксного дерева, сформованого «нальоту», по мірі надходження нових символів в «хвіст» рядку. Нехай рядок S в остаточному вигляді складається з послідовності символів $t_1,..., t_n$. Вводиться поняття префікса, тобто послідовності символів $t_1, ..., t_i$ – початок рядка. Суфіксне дерево будується не тільки для всього рядка, але і для всіх його префіксів. Розглянемо алгоритм детальніше по кроках:

0. Будується суфіксне дерево для t_1.

1. Суфіксне дерево розширюється шляхом додавання гілок t_1t_2.

n – 1. Суфіксне дерево для $t_1, ..., t_{n-1}$ розширюється до дерева для $t_1,..., t_n$.

n. Суфіксне дерево для $t_1, ..., t_n$ розширюється до дерева для $t_1,..., t_n\$$ ($ – кінець тексту).

Послідовність кроків наведеного алгоритму для рядка abca$ наведена на рис. 4.7.

В даний час ідеологія суфіксних дерев застосовується для кластеризації результатів роботи інформаційно-пошукових систем в інтерактивному режимі.

Зупинимося докладніше на принципах побудови і застосування суффіксних дерев в разі, коли в якості текстового рядка використовується весь текст документа, а в якості символів - слова.

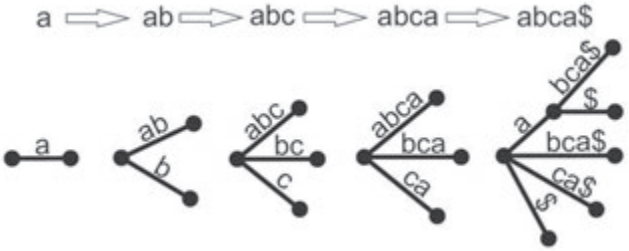


Рис. 4.7 – Процес побудови суфіксного дерева

Спочатку при побудові дерева документи, що одержані від пошукової системи, підлягають очищенню від пунктуації, слова з документів приводяться до канонічної форми. Після цього для знайдених документів будується суфіксне дерево, але в цьому випадку гілкам приписуються терми (слова або словосполучення), а не букви як при традиційному підході. В результаті вершинам дерева відповідають фрази, які можна отримати, об'єднавши всі терми, що знаходяться на гілках, які ведуть від кореня до вибраної вершині дерева. У вершинах дерева, що мають нащадків, розташовані посилання на документи, в яких зустрічається фраза, відповідна вершині. Її можна отримати, об'єднавши всі слова, що знаходяться на ребрах на шляху від кореня дерева до даної вершині. Множина документів, на які вказують ці посилання, утворюють базові кластери. Потім відбувається укрупнення базових кластерів і отримання остаточного набору кластерів. На рис. 4.8 наведено суфіксне дерево для пропозиції «I know you know I know». 6 вузлів в цьому прикладі марковані як прямокутники з цифрами від 1 до 6.

Суфіксне дерево можна генерувати також з декількох речень. На рис. 4.9 наведено приклад загального суфіксного дерева для трьох речень «cat ate cheese», «mouse ate cheese too» і «cat ate mouse too». Внутрішні вузли на

даному графі представлені як кола, які позначені буквами від a до f. Одинадцять листків в цьому прикладі марковані прямокутниками, перша цифра в яких – номер речення, з якого взято суфікс, а друга цифра – позиція в реченні.

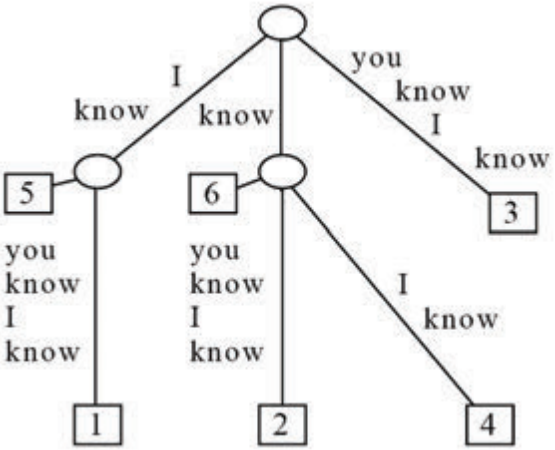


Рис. 4.8 – Приклад суфіксного дерева для одного речення

Кластери визначаються на основі близькості між множинами документів, що відповідають вузлам суфіксного дерева, за наступним правилом. Нехай B_m і B_n – базові кластери $|B_m|$, $|B_n|$ – їх розміри. $|B_m \cap B_n|$ – кількість спільних документів для цих кластерів. Близькість $sim(B_m, B_n)$ між B_m і B_n задається умовою:

$$sim(B_m, B_n) = \begin{cases} 1, & \textit{если } \dfrac{|B_m \cap B_n|}{|B_m|} > \alpha, \ \dfrac{|B_m \cap B_n|}{|B_n|} > \alpha; \\ 0, & \textit{иначе}, \end{cases}$$

де α – деякий поріг, що приймає значення від 0 до 1, наприклад 0,6.

На рис. 4.10 показані результати формування кластерів для випадку, наведеного на рис. 4.9 з 6-ю вузлами (від a до f). У випадку *a* поріг

α = 0.7, в випадку *b* α = 0.6, у випадку *c* поріг також дорівнює 0.6, проте слово ate виключено з допомогою «стоп-словника».

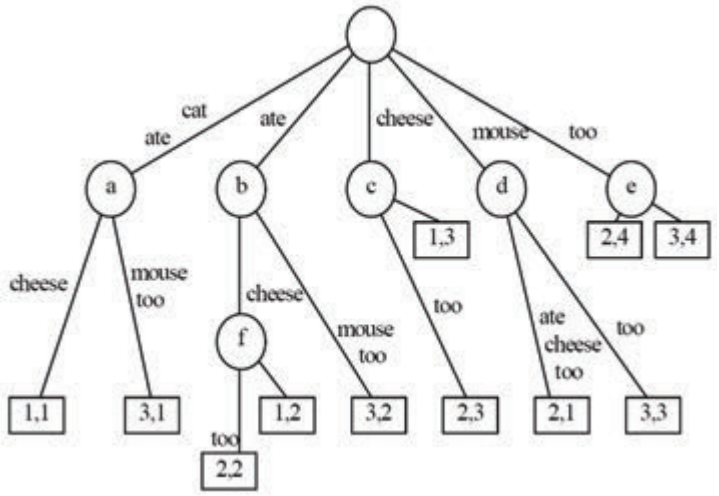


Рис. 4.9 – Приклад загального суфіксного дерева для декількох речень

На відміну від більшості моделей кластеризації, що розглядають текст як "Bag of Words", в методі суфіксних дерев враховується порядок слів, тобто лінгвістичних особливостей текстів, що будуть згруповані. На відміну від багатьох інших методів, кластери, сформовані за допомогою методу суфіксних дерев можуть перетинатися, що цілком відповідає реальності. До переваг цього методу можна віднести також наочність представлення його результатів і високу швидкість роботи.

4.7. Самоорганізаційні карти

Самоорганізаційні карти (Self-Organizing Maps – SOM) – це один із різновидів нейромережевих алгоритмів. Основною відмінністю даного підходу від інших нейромереж, є те, що при навчанні використовується метод навчання без вчителя, тобто результат навчання залежить тільки від структури вхідних даних. Нейронні мережі цього типу часто застосовуються для вирішення найрізноманітніших завдань, від

відновлення пропусків у даних до аналізу даних і пошуку закономірностей, наприклад, у фінансовій сфері.

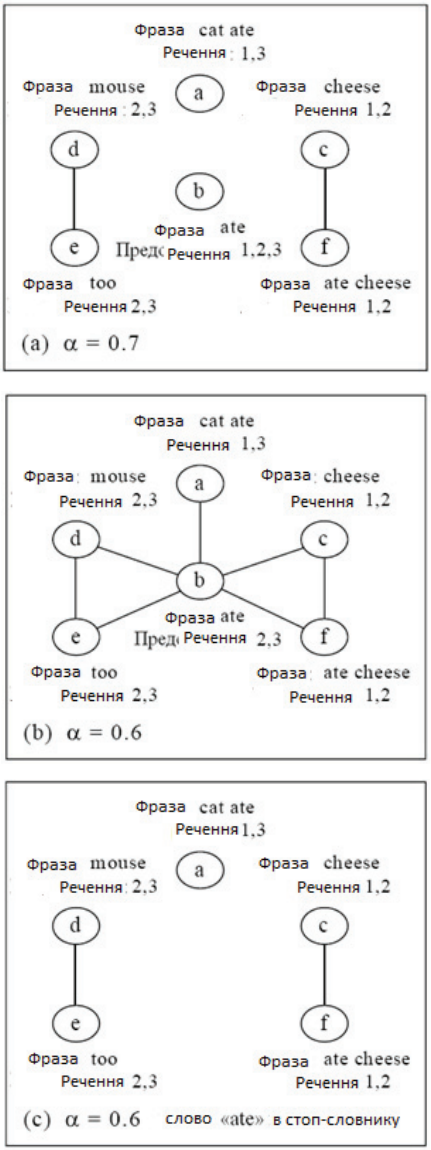


Рис. 4.10 – Графи кластерів суфіксного дерева

Алгоритм функціонування самоорганізаційних карт, що самі навчаються, є одним з варіантів кластеризації багатовимірних векторів. Прикладом таких алгоритмів може служити алгоритм найближчих середніх (c-means). Важливою відмінністю алгоритму SOM є те, що в ньому всі нейрони (вузли, центри класів) впорядковані в деяку структуру (зазвичай двовимірну сітку). При цьому в ході навчання модифікується не тільки нейрон-переможець, а також і його сусіди, але в меншому ступені. За рахунок цього SOM можна вважати одним з методів проектування багатовимірного простору на простір з низькою розмірністю. При використанні цього алгоритму вектори, подібні у вихідному просторі, виявляються поруч і на отриманій карті.

SOM має на увазі використання впорядкованої структури нейронів. Зазвичай використовуються одно- і двовимірні сітки. При цьому кожен нейрон являє собою *n*-мірний вектор-стовпець $\bar{w} = \begin{bmatrix} w_1 \\ w_2 \\ \dots \\ w_n \end{bmatrix}$, де *n* визначається розмірністю вихідного простору (розмірністю вхідних векторів). Застосування одно- і двовимірних сіток пов'язано з тим, що виникають проблеми при відображенні просторових структур більшої розмірності.

Зазвичай нейрони розташовуються у вузлах двовимірної сітки з прямокутними або шестикутними комірками. При цьому нейрони також взаємодіють один з одним. Величина цього взаємодії визначається відстанню між нейронами на карті.

При реалізації алгоритму SOM заздалегідь задається конфігурація сітки (прямокутна або шестикутна), а також кількість нейронів у мережі. При цьому початковий радіус навчання (neighborhood) в значній мірі впливає на здатність узагальнення за допомогою, отриманою карти. У разі, коли кількість вузлів карти перевищує кількість прикладів в навчальній вибірці, то успіх використання алгоритму у значній мірі залежить від

відповідного вибору початкового радіуса навчання. Однак у разі, коли розмір карти становить десятки тисяч нейронів, час, необхідний на навчання карти, зазвичай буває занадто великим для вирішення практичних завдань, таким чином, необхідно досягати допустимого компромісу при виборі кількості вузлів.

Перед початком навчання карти необхідно ініціалізувати вагові коефіцієнти нейронів. Вдало обраний спосіб ініціалізації може істотно прискорити навчання і привести до отримання якісних результатів. Існують три способи ініціювання початкової ваги:

1. Ініціалізація випадковими значеннями, коли всім значенням ваги даються малі випадкові величини.

2. Ініціалізація прикладами, коли в якості початкових значень ваги задаються значення випадково вибраних прикладів з навчальної вибірки.

3. Лінійна ініціалізація. У цьому випадку значення ваги ініціюються значеннями векторів, лінійно впорядкованих вздовж лінійного підпростору, що проходить між двома головними власними векторами вихідного набору даних. Власні вектори можуть бути знайдені, наприклад, за допомогою процедури Грама-Шмідта.

Навчання самоорганізаційних карт складається з послідовності корекцій векторів, що представляють собою нейрони. На кожному кроці навчання з вихідного набору даних випадково вибирається один з векторів, а потім проводиться пошук найбільш схожого на нього вектора коефіцієнтів нейронів. При цьому вибирається нейрон-переможець, який найбільш подібний до вхідного вектора. Під подібністю у цій задачі розуміється відстань між векторами, яка зазвичай обчислюється в евклідовому просторі. Після того, як знайдений нейрон-переможець, здійснюється коригування значень ваги нейромережі. При цьому вектор, що описує нейрон-переможець, і вектори, що описують його сусідів у сітці, переміщаються в напрямку вхідного вектора.

Навчання карти Кохонена

Традиційно карта Кохонена (КК) представляється двовимірним шаром нейронів, кожен з яких пов'язаний з вхідним шаром. Кожен нейрон в карті Кохонена має два параметри:

1. Вектор w ваг зв'язків. Розмірність w дорівнює кількості входів нейромережі.

2. Двовимірні координати z, що визначають положення нейрона щодо інших нейронів.

Сам шар нейронів працює за принципом WTA (Winner Takes All). Відповідно до цього принципу, в шарі нейронів активованим («переможцем») вважається тільки один нейрон. Для різних вхідних векторів переможцями можуть бути різні нейрони. При навчанні карти Кохонена коригуються ваги зв'язків нейрона-переможця і тих нейронів, які до нього ближче.

У картах Кохонена нейрон переможець визначається по тому, наскільки вектор його ваг близький до поточного вхідного вектора. Це можна визначити, просто обчисливши вираз:

$$winner_i = \arg\min_{j=1..K} r(\mathbf{x}_i, \mathbf{w}_j),$$

де r – метрика, $\mathbf{x}_i$ – поточний вхідний вектор, $\mathbf{w}_j$ – вектор ваг j-го нейрона, K – кількість нейронів в мережі Кохонена. Можна використовувати і функцію активації, наприклад, такого вигляду:

$$f(\mathbf{x}_i, \mathbf{w}_j) = \exp\left(-\left\|\mathbf{x}_i - \mathbf{w}_j\right\|\right),$$

де $\|\cdot\|$ – евклідова міра. Ця функція приймає максимальне значення, якщо $\mathbf{x}_i = \mathbf{w}_j$, або $r(\mathbf{x}_i, \mathbf{w}_j) = 0$.

Ініціалізація нейронів мережі проводиться таким чином:

1. Випадкова ініціалізація векторів ваг нейронів.

2. Початкові координати нейронів збігаються з вузлами прямокутної решітки. У деяких випадках координати задають з малим шумом.

Відзначимо, що ініціалізація ваг нейронів робить істотний вплив на результат, проте найкращий спосіб ініціалізації, що забезпечує оптимальний варіант, невідомий. Проте, будемо використовувати випадкову ініціалізацію. Приклад мережі 3x3 показаний на Рис. 4.11.

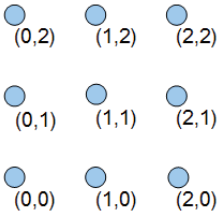


Рис. 4.11 – Початкове розташування нейронів в карті Кохонена розміром 3x3. Показані двовимірні координати нейронів. Ваги нейронів не показані

Серед алгоритмів навчання найбільш часто використовуються два:

1. Послідовний (Iterative). Оновлення ваг проводиться після кожного навчального прикладу.
2. Пакетний (Batch Map). Спочатку пред'являються все приклади, а потім проводиться оновлення ваг.

Пакетний алгоритм вимагає великих витрат по пам'яті, однак може сходитися в кілька разів швидше послідовного.

Алгоритм послідовного навчання

Формули для оновлення ваги зв‘язків для j-го нейрона по послідовному алгоритму мають вигляд:

$$\mathbf{w}_j(t+1) = \mathbf{w}_j(t) + \eta(t) h_{cj}(t)[\mathbf{x}_i(t) - \mathbf{w}_j(t)],$$

де η – швидкість навчання, h_{cj} – функція відстані від j-го нейрона до нейрона переможця, $\mathbf{x}_i(t)$ – i-й вектор, що навчається.

Вибір вигляду h_{cj} відіграє важливу роль в навчанні карти Кохонена. Типовим прикладом є Гаусіан:

$$h_{cj} = \exp\left(-\frac{d_{cj}^2}{2\sigma^2}\right),$$

де $d_{cj}^2 = \left\|\mathbf{z}_j - \mathbf{z}_c\right\|$ – відстань на площині між j-м нейроном і поточним нейроном-переможцем; σ – числовий параметр, що має назву *ефективної ширини*, визначає розмір околиці навколо нейрона переможця, в якому проводиться корекція ваг. Чим менше σ, тим менше нейронів з околиці переможця, на яких впливає пропонований вхідний вектор $\mathbf{x}_i$.

Особливістю навчання КК є поступове зменшення розміру околу. Часто застосовують:

$$\sigma(t) = \sigma_0 \exp\left(-\frac{t}{\tau_1}\right), \quad t = 1,2,\ldots,$$

де τ_1 – параметр, який відповідає за швидкість убування $\sigma(t)$. Рекомендоване значення для τ_1:

$$\tau_1 = \frac{T}{\sigma_0},$$

де T – передбачуваний час навчання, наприклад $T = 1000$.

Рекомендоване значення для σ_0:

$$\sigma_0 = r,$$

де r – радіус «решітки» нейронів, визначається по двовимірним координатам нейронів. Для випадку прямокутної решітки з цілочисельними координатами вузлів, можна обчислювати,

$$r = \left\lfloor \frac{\min\{H, W\}}{2} \right\rfloor,$$

де W і H – відповідно ширина і висота конфігурації нейронів, $\lfloor\cdot\rfloor$ – взяття найближчого цілого знизу (відкидання дробової частини). Так, якщо нейрони розташовані у вигляді сітки 7×7, то $r = 3$, $\sigma_0 = 3$.

Роботу алгоритму послідовного навчання можна представити таким чином: для кожного вхідного вектора з навчального набору шукається

нейрон, значення ваги зв'язків якого найближче до цього вектору. В результаті навчання значення ваги нейрона-переможця стають ще сильніше «схожі» на вхідний вектор, крім цього змінюються і значення ваги нейронів, які близькі до нейрона-переможця.

Для збіжності алгоритму навчання необхідне виконання наступної умови: $\eta(t) \to 0$ при $t \to +\infty$.

Часто використовується співвідношення:

$$\eta(t) = \eta_0 \exp\left(-\frac{t}{\tau_2}\right), \quad t = 1,2,...,$$

де η_0 – початкова швидкість навчання, τ_2 – параметр, який регулює спадання швидкості навчання. Рекомендовані значення для цих параметрів: $\eta_0 = 0.1$, $\tau_2 = 1000$, таким чином, досягається досить повільне убування $\eta(t)$, при якому більшу частину часу $\eta(t) > 0.01$, щоб уникнути ранньої збіжності процесу навчання.

В результаті навчання значення ваги нейронів сходяться до координат центрів скупчень точок даних. При цьому алгоритм навчання гарантує, що ні в яких двох нейронів не буде однакових векторів ваги.

Алгоритм пакетного навчання

Ідея пакетного навчання збігається з такою для послідовного, однак формули для поновлення ваг зв'язків для j-го нейрона щодо пакетного алгоритму істотно відрізняються:

$$\mathbf{w}_j(t+1) = \frac{\sum_{i \in O_j} n_i h_{ij} \bar{\mathbf{x}}_i}{\sum_{i \in O_j} n_i h_{ij}},$$

де O_j – окіл *j*-го нейрона, що визначається параметром σ; n_i - кількість вхідних векторів, для яких *i*-й нейрон виявився переможцем;

$\bar{\mathbf{x}}_i = \frac{1}{n_i}\sum_k [winner_k = i]\mathbf{x}_i$ – середнє значення вхідних векторів, для яких переможцем виявився i-й нейрон, тут $[winner_k = i]$ – індикаторна функція:

$$[winner_k = i] = \begin{cases} 1, & winner_k = i \\ 0, & winner_k \neq i \end{cases}$$

Вибір виду h_{ij} та інших параметрів збігається з рекомендаціями для алгоритму послідовного навчання.

Алгоритм пакетного навчання сходиться набагато швидше алгоритму послідовного навчання. Відзначимо, що оновлення ваги за наведеними формулами може бути ефективно розгалужено, тому що значення ваги нейронів оновлюються незалежно.

В результаті навчання карти Кохонена можуть з'явитися «мертві» нейрони, тобто нейрони, які не є переможцями ні для одного вхідного вектора. Такі нейрони не описують жодного скупчення вхідних векторів і часто їх можна відкинути. Виявлення мертвих нейронів можна проводити після навчання карти Кохонена. Для цього для кожного нейрона підраховується, скільки разів він виявився переможцем після пред'явлення всіх навчальних векторів.

Оскільки навчена карта Кохонена представляє опис вихідних даних, то є помилка представлення цих даних і помилку можна обчислити, наприклад, так:

$$err = \sum_i \left\| \mathbf{w}_{winner_i} - \mathbf{x}_i \right\|^2 .$$

Алгоритм навчання карти Кохонена

Таким чином, загальний алгоритм навчання карти Кохонена КК можна описати таким чином:

1. Вибір структури шару нейронів (конфігурація нейронів).

2. Ініціалізація ваг і координат нейронів.

3. Поки не виконано критерій зупинки: пред'явлення прикладів з навчальної множини і корекція ваг і координат нейронів.

4. Визначення та відкидання «мертвих» нейронів.

4.8. Побудова карт Кохонена у середовищі Matlab

Завдання: використовуючи вбудовані функції пакета нейронних мереж математичної середовища Matlab, побудувати нейронну мережу з шаром Кохонена, яка розділить на кластери множину вхідних даних і виявить їх центри. На навчену мережу подати новий вхідний вектор і визначити, до якого кластеру він відноситься.

Для створення нейронної мережі з шаром Кохонена скористаємося вбудованою у середовище Matlab функцією newc:

```
X=[0 1; 0 1];
clusters=5;
points=5; %Завдання кількості точок в кластері
std_dev=0.01;
p=nngenc(X,clusters,points,std_dev); % Моделювання вхідних даних
h=newc([0 1;0 1],5,.1);  % Створення шару Кохонена
h.trainParam.epochs=50;  % Завдання кількості циклів навчання
h=init(h);
h=train(h,p);
w=h.IW{1}; % Виведення даних та виявлення кластерів
plot(p(1,:),p(2,:),'^r'),grid;
hold on;
plot(w(:,1),w(:,2),'ob');
xlabel('p(1)');
ylabel('p(2)');
A=0.6
B=0.5
```

```
p=[A;B];
plot(A,B,'^k')
y=sim(h,p)  % Опитування мережі
```

Результат роботи програми можна побачити в командному вікні:

У= (2,1)

Пред'явлений вектор віднесений до другого кластеру.

Застосування алгоритму до задачі кластеризації. На вхід нейронної мережі подаються дані щодо ваги и росту людей. Необхідно виявити три кластери за вагою і ростом:

- нормальний;
- надлишок ваги;
- недолік ваги.

Нижче наведено програму побудови КК мовою системи Matlab.

```
% Вхідні данні (перший рядок матриці – ріст; другий – вага)
p=[175 180 182 175 183 176 183 176 183 176 175 180 178 180 178 182 178 182 179 174 172 179; 70 75 100 99 42 48 76 72 40 45 92 96 70 69 95 90 79 82 80 50 96 91]
% Створюємо мережу Кохонена з 3 кластерами
h=newc([0 200;0 100],3,.1)
h.trainParam.epochs=500; % Завдання кількості циклів навчання
h=train(h,p)
w=h.IW{1};
plot(p(1,:),p(2,:),'^r');
hold on;
plot(w(:,1),w(:,2),'ob');
xlabel('Rost');
ylabel('Ves');
% Завдання нового вхідного вектора
% Тестування-опитування мережі
A=181
B=65
p=[A;B];
plot(A,B,'+r')
y=sim(h,p)
A =181
B = 65
```

Результат роботи програми можна побачити в командному вікні:

y= (2,1) 1

Вектор віднесено до другого кластеру.

Питання для самоперевірки

1. Яке призначення кластерного аналізу?
2. Чим кластерний аналіз відрізняється від процесу класифікації?
3. У чому полягає особливість методу k-means?
4. Для чого використовуються SOM-карти?
5. Яким чином можна навчати карти Кохонена?

Література до розділу

1. *Ландэ Д.В., Снарский А.А., Безсуднов И.В.* Интернетика: Навигация в сложных сетях: модели и алгоритмы. – М.: Либроком (Editorial URSS), 2009. – 264 с. 2. *Додонов А.Г., Ландэ Д.В., Путятин В.Г.* Компьютерные сети и аналитические исследования. – К.: ИПРИ НАН Украины, 2014. – 486 с. 3. *Мандель И. Д.* Кластерный анализ. – М.: Финансы и Статистика, 1988. 4. *Уиллиамс У. Т., Ланс Д. Н.* Методы иерархической классификации // Статистические методы для ЭВМ / Под ред. М. Б. Малютов. – М.: Наука, 1986. – С. 269–301. 5. *Руденко О.Г., Бодянский Е.В.* Искусственные нейронные сети – Харьков, 2005. – 407 с. 6. *Дебок Г., Кохонен Т.* Анализ финансовых данных с помощью самоорганизующихся карт. Пер. с англ. – М.: Альпина, 2001. – 317 с. 7. *Медведев В.С., Потемкин В.Г.* Нейронные сети. MATLAB 6 – М.: ДИАЛОГ-МИФИ, 2002. – 496 с. 4. *Чубукова И.А.* Data Mining. — 2000. — 326 с.

5. ОСНОВИ М‘ЯКИХ ОБЧИСЛЕНЬ

5.1. Вступ до м'яких обчислень

М'які обчислення – поняття, введене Лотфі Заде у 1994 році, що об'єднує в загальний клас неточні, наближені методи розв'язання задач, що часто не мають рішення за поліноміальний час. Завдання, які вирішуються такого класу методами, виникають в області біології, медицини, гуманітарних наук, менеджменту.

Технології м'яких обчислень орієнтовані на рішення задач управління слабкоструктурованими об'єктами управління; інструментарій м'яких обчислень використовує техніку нечітких систем (нечіткі множини, нечітку логіку, нечіткі регулятори), нечіткі нейронні мережі, генетичні алгоритми та еволюційне моделювання (в тому числі імунні алгоритми, алгоритми ройового інтелекту – на основі поведінкових реакцій груп тварин, мурах, бджіл). Різні методи м'яких обчислень можуть доповнювати один одного і часто використовуються спільно.

5.2. Класифікація м'яких обчислень

М'які обчислення – це спеціальна комп'ютерна методологія, заснована на нечіткій логіці, генетичних обчисленнях, нейрокомпьютингу і імовірнісних обчисленнях. Ведучий принцип м'яких обчислень – це урахування неточності, невизначеності, часткової істини і апроксимації для досягнення робастності, низьку ціну рішення, більшої відповідності з реальністю.

Складові частини м'яких обчислень охоплюють:

- нечітку логіку;
- нейрокомп‘ютерні обчислення;
- генетичні алгоритми;
- мурашині алгоритми тощо.

Нечітка логіка

Вперше термін нечітка логіка (fuzzy logic) був введений американським професором Лотфі Заде у 1965 році в роботі "Нечіткі множини" в журналі "Інформатика та управління".

Областю впровадження алгоритмів нечіткої логіки є експертні системи, у тому числі: системи нелінійного контролю за процесами (виробництво);· самонавчальні системи (або класифікатори), дослідження ризикових і критичних ситуацій; розпізнавання образів; фінансовий аналіз (ринки цінних паперів); дослідження даних (корпоративні сховища); вдосконалення стратегій управління та координації дій, наприклад, складне промислове виробництво.

Міць і інтуїтивна простота нечіткої логіки як методології вирішення проблем забезпечує її ефективне використання у вбудованих системах контролю і аналізу інформації. При цьому відбувається підключення людської інтуїції і досвіду оператора.

Недоліками нечітких систем є:

- відсутність стандартної методики конструювання нечітких систем;
- неможливість математичного аналізу нечітких систем існуючими методами;
- застосування нечіткого підходу в порівнянні з імовірнісним не приводить до підвищення точності обчислень.

Нейрокомп‘ютерні обчислення

Нейрокомп‘ютерні обчислення – нейрокомп‘ютинг – це технологія створення систем обробки інформації (наприклад, нейронних мереж), які здатні автономно генерувати методи, правила і алгоритми обробки у вигляді адаптивної відповіді в умовах функціонування у конкретному інформаційному середовищі. Нейрокомп‘ютінг є фундаментально новим підходом, в рамках цього підходу розглядаються системи обробки

інформації істотно відрізняються від інших систем і методів. Ця технологія охоплює паралельні, розподілені, адаптивні системи обробки інформації, здатні «вчитися» обробляти інформацію, що функціонують в інформаційному середовищі. Таким чином, нейрокомп‘ютінг можна розглядати як перспективну альтернативу програмованим обчисленням, принаймні, у тих областях, де його вдається застосовувати.

Новий підхід не вимагає готових алгоритмів і правил обробки – система повинна «вміти» виробляти правила і модифікувати їх в процесі вирішення конкретних завдань обробки інформації. Для багатьох завдань, де такі алгоритми невідомі, або ж відомі, але вимагають значних витрат на розробку програмного забезпечення (наприклад, при обробці зорової та слухової інформації, розпізнаванні образів, аналізі даних, управлінні) нейрокомп‘ютінг дає ефективні паралельні методи розв'язання, що легко і швидко реалізуються. Цікавим є також і зворотна задача: аналізуючи навчену систему, визначити розроблений нею алгоритм вирішення задачі.

Недоліки нейрокомп‘ютінгу в тому, що кожен алгоритм створюється спеціально для вирішення конкретних завдань, пов'язаних з нелінійної логікою і теорією самоорганізації.

Генетичні алгоритми

Генетичний алгоритм – це евристичний алгоритм пошуку, який використовується для вирішення завдань оптимізації та моделювання шляхом випадкового підбору, комбінування і варіації параметрів, які визначаються, з використанням механізмів, що нагадують біологічну еволюцію. Цей алгоритм є різновидом еволюційних обчислень, за допомогою яких вирішуються оптимізаційні задачі з використанням методів природної еволюції, таких як успадкування, мутації, відбір і кросинговер. Відмінною особливістю генетичного алгоритму є акцент на використання оператора «схрещування», який виробляє операцію

рекомбінації рішень-кандидатів, роль якої аналогічна ролі схрещування в живій природі.

Завдання генетичних алгоритмів формалізується таким чином, щоб її рішення могло бути закодовано у вигляді вектора генів («генотипу»), де кожен ген може бути бітом, числом або якимось іншим об'єктом. У класичних реалізаціях генетичних алгоритмів передбачається, що генотип має фіксовану довжину. Однак існують варіації цих алгоритмів, що вільні від цього обмеження.

5.3. Клітинні автомати

Концепція клітинних автоматів була вперше запропонована Дж. фон Нейманом і розвинута С.Вольфрамом у фундаментальній монографії “Новий вид науки”.

Клітинні автомати є корисними дискретними моделями для дослідження динамічних систем. Дискретність моделі, точніше, можливість представити модель у дискретній формі, на цей час відноситься до істотних переваг, що відкриває широкі можливості використання комп'ютерних технологій. Клітинні автомати в цьому сенсі займають особливе місце, оскільки в них дискретність поєднується з іншими перевагами.

Головним достоїнством клітинних автоматів є їх абсолютна сумісність із алгоритмічними методами рішення завдань. Скінчений набір формальних правил, заданий на скінченій множині елементів (кліток), допускає точну реалізацію у вигляді алгоритмів. Однак звідси випливає й головний недолік клітинних автоматів: обчислювальні труднощі, що виникають при відповідних розрахунках. Адже на кожній ітерації необхідно сканувати весь набір кліток і для кожної з них виконувати необхідні операції. Коли й кліток, і ітерацій дійсно багато, це вимагає значних ресурсів, у тому числі часових. Тому довгий час клітинні автомати сприймалися в основному як забавна, хоча й повчальна гра, що не має

практичної цінності. Але в останні роки, у зв'язку з розвитком комп'ютерних технологій, вони починають швидко входити до арсеналу інструментальних засобів, що використовуються на практиці в найрізноманітніших галузях науки й техніки.

Клітинні автомати, по своїй суті, є просторово-немобільними дискретними індивідуум-орієнтованими моделями. У традиційній системі клітинних автоматів всі клітини рівноправні (простір однорідний), у той час як в індивідуум-орієнтований – крім опису комірок існує поняття індивідуума, що може займати різні комірки (і кілька різних індивідуумів можуть займати одну комірку). Відомо, що у системі клітинних автоматів клітини міняють свій стан синхронно, і цикл моделювання являє собою перебір клітин. В індивідуум-орієнтованих моделях цикл може складатися з перебору індивідуумів. Тобто, у клітинному автоматі моделювання засноване на розбивці простору на однорідні ділянки, а в індивідуум-орієнтованих моделях описуються сутності, які міняють положення в просторі. Звичайно, клітини в клітинному автоматі можуть перебувати в різних станах, і за допомогою визначення складних станів можна моделювати наявність особливостей у клітинах та переміщення станів між клітинами. Але це можливо лише при істотних обмеженнях.

Клітинний автомат являє собою дискретну динамічну систему, сукупність однакових клітин, певним чином з'єднаних між собою. Всі клітини утворять мережу (решітку) клітинних автоматів. Стан кожної клітини визначаються станом клітин, що входять у її локальний окіл та називаються найближчими сусідами. Тобто околом клітинного автомата з номером j називається множина його найближчих сусідів. Стан j-го клітинного автомата в момент часу $t + 1$, таким чином, визначається в такий спосіб:

$$y_j(t + 1) = F(y_j(t), O(j), t),$$

де F – деяке правило, яке можна виразити, наприклад, мовою булевої алгебри. У багатьох завданнях, вважається, що сам елемент відноситься

до своїх найближчих сусідів, тобто $y_j \in O(j)$, у цьому випадку формула спрощується: $y_j(t + 1) = F(O(j), t)$. Клітинні автомати у традиційному розумінні задовольняють таким правилам:

- зміна значень всіх клітин відбувається одночасно (одиниця виміру – такт);
- мережа клітинних автоматів однорідна, тобто правила зміни станів для всіх клітин однакові;
- на клітину можуть вплинути лише клітини з її локального околу;
- множина станів клітин скінченна.

Теоретично клітинні автомати можуть мати будь-яку розмірність, однак найчастіше розглядають одномірні та двовимірні системи клітинних автоматів.

Модель, яку будемо розглядати далі, є двовимірною, тому подальший формалізм стосується цього випадку. У двовимірному клітинному автоматі решітка реалізуються двовимірним масивом. Тому в цьому випадку зручно перейти до двох індексів, що цілком коректно для скінченних решіток.

У випадку двовимірних решіток, елементами яких є квадрати, найближчими сусідами, що входять в окіл елемента $y_{i,j}$ можна вважати або тільки елементи, розташовані внизу та зверху і зліва та справа від нього (так званий окіл фон Неймана: $y_{i-1,j}$, $y_{i,j-1}$, $y_{i,j}$, $y_{i,j+1}$, $y_{i+1,j}$), або ще додані до них діагональні елементи (окіл Мура: $y_{i-1,j-1}$, $y_{i-1,j}$, $y_{i-1,j+1}$, $y_{i,j-1}$, $y_{i,j}$, $y_{i,j+1}$, $y_{i+1,j-1}$, $y_{i+1,j}$, $y_{i+1,j+1}$). У моделі Мура кожна клітина має вісім сусідів.

Для усунення крайових ефектів скінченні решітки топологічно «згортаються у тор» (Рис. 5.1), тобто перший рядок вважається продовженням останнього, а останній – попередником першого. Те ж саме відноситься й до стовпців.

Це дозволяє визначати загальне співвідношення значення клітки на кроці $t + 1$ у порівнянні із кроком t :

$y_{i,j}(t) =$

$F(y_{i-1,j-1}(t), y_{i-1,j}(t), y_{i-1,j+1}(t), y_{i,j-1}(t), y_{i,j}(t), y_{i,j+1}(t), y_{i+1,j-1}(t), y_{i+1,j}(t), y_{i+1,j+1}(t))$.

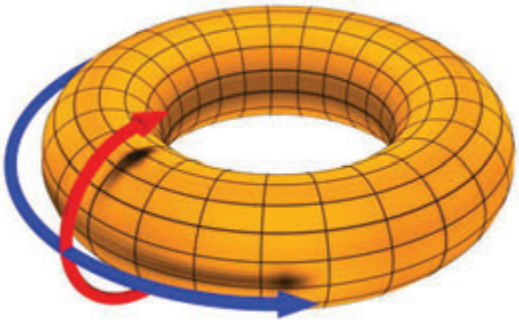


Рис. 5.1. Згортання площини в тор

Моделювання еволюційних процесів

С. Вольфрам, класифікуючи різні клітинні автомати, виділив ті, динаміка яких істотно залежить від початкового стану. Підбираючи різні початкові стани, можна одержувати найрізноманітніші конфігурації та типи поводження. Саме до таких систем ставиться класичний приклад – гра "Життя", винайдена Дж. Конвеєм.

Клітинні автомати з успіхом застосовуються при моделюванні процесів поширення інновацій. У статті Т. Брауна розглядається модель електорального процесу, де передбачається, що виборчі переваги людини визначаються установками його найближчого оточення.

У одній з моделей припускається, що індивід приймає рішення голосувати в момент $t + 1$ за республіканців або демократів у відповідності з правилом простої більшості. Враховуються погляди індивіду та чотирьох його найближчих сусідів у момент t (окіл фон Неймана). Модель досліджувалася на великому часовому відрізку – до 20000 тактів. Виявилося, що партійна боротьба приводить до дуже складних конфігурацій, які істотно залежать від вихідного розподілу.

Клітинні автомати є корисними дискретними моделями для дослідження динамічних систем. Головною перевагою клітинних автоматів є їх абсолютна сумісність із алгоритмічними методами рішення завдань. Скінчений набір формальних правил, заданий на скінченій множині елементів (кліток), допускає точну реалізацію у вигляді алгоритмів.

З метою отримання навичок моделювання еволюційних процесів розглянемо реалізацію гри «Життя» у середовищі, розгляд якого було розпочато у попередньому розділі. Правило варіанту гри «Життя», моделюванню якого присвячено цій розділ, такі. Клітина знаходиться у одному з двох станів – живому та неживому (чорному та білому). Якщо сусідами клітини є менше двох або більше трьох чорних клітин, то на наступному кроці вона зафарбовується у білий колір (вмирає). Якщо сусідами клітини є рівно три чорних клітини, то на наступному кроці вона зафарбовується у чорний колір (народжується).

Нижче наведено фрагмент програми мовою системи Matlab, що реалізує гру «Життя» на полі розміром 40×40 клітин для заданої початкової конфігурації (у вигляді хреста). У програмі забезпечено її рекурентний виклик. Результати роботи програми наведено на Рис. 5.2.

```
clear
colormap(gray);
N=40
hold on
for i=1:N
        for j=1:N
                c(i,j)=0;
        end
end
% Початкова конфігурація
c(N/2-1,N/2)=1;
c(N/2+1,N/2)=1;
c(N/2,N/2-1)=1;
c(N/2,N/2+1)=1;
c(N/2,N/2)=1;
% Відображення початкової конфігурації
d=1-c;
```

```
pcolor(d);
% Головний цикл еволюції
for u=1:15
        hold on
        c1=c;
        for i=2:N-1
                for j=2:N-1
                m=c(i-1,j-1)+c(i,j-1)+c(i+1,j-1)+c(i-1,j)+c(i+1,j)+c(i-
        1,j+1)+c(i,j+1)+c(i+1,j+1);
                if m<2
                        c1(i,j)=0;
                end
                if m==3 & c(i,j)==0
                        c1(i,j)=1;
                end
                if m>2 & c(i,j)==1
                        c1(i,j)=1;
                end
                if m>3
                        c1(i,j)=0;
                end
           end
        end
        c=c1;
        for j=1:N
                c(1,j)=c1(N-1,j);
                c(N,j)=c1(2,j);
        end
        for j=1:N
                 c(j,1)=c1(j,N-1);
                  c(j,1)=c1(j,2);
        end
        d=1-c;
        pcolor(d)
        pause (1)
end
```

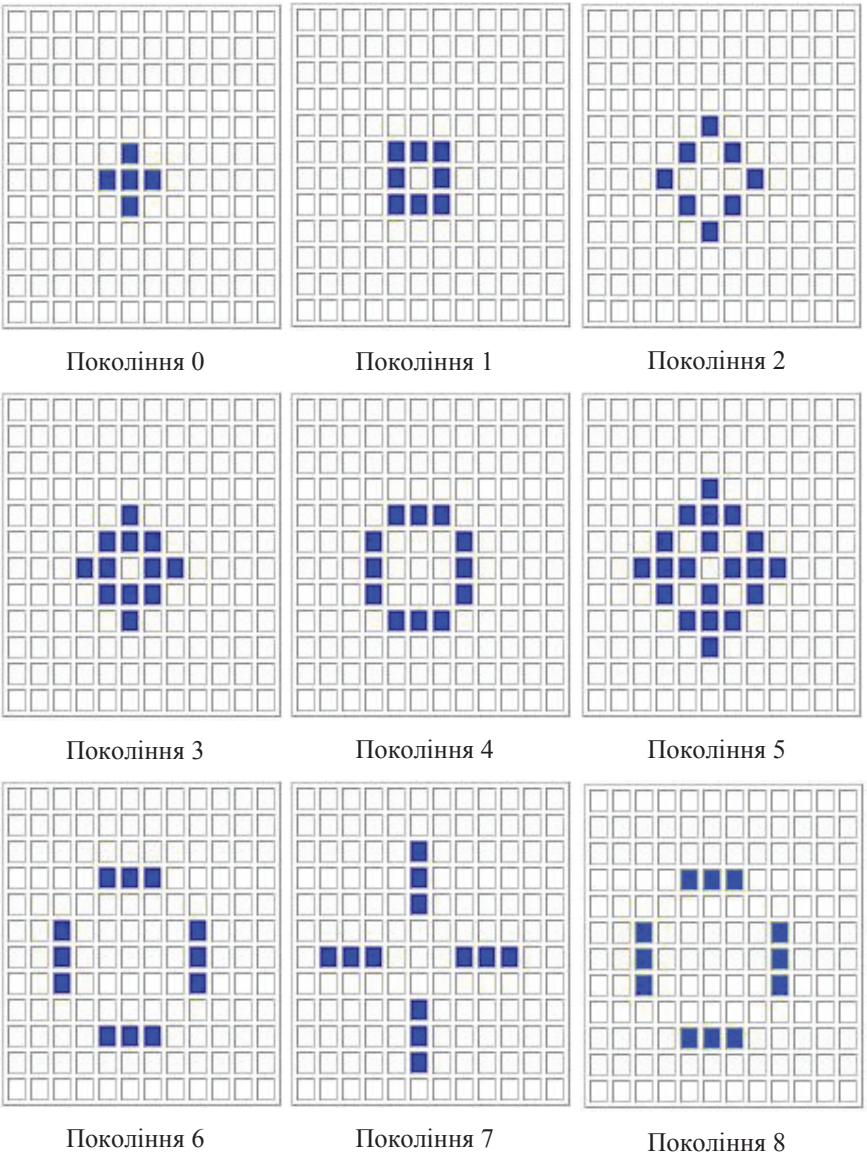


Рис. 5.2. Покоління гри «Життя» з конфігурацією «хрест»

5.4. Мурашиний алгоритм

Мурашиний алгоритм (алгоритм оптимізації мурашиної колонії, англ. Ant Colony Optimization, ACO) — один з ефективних поліноміальних алгоритмів для знаходження наближених розв'язків задачі комівояжера, а також аналогічних завдань пошуку маршрутів на графах. Підхід запропонований бельгійським дослідником Марко Доріго (Marco Dorigo). Суть підходу полягає в аналізі та використанні моделі поведінки мурах, що шукають дороги від колонії до їжі.

У основі алгоритму лежить поведінка мурашиної колонії – маркування вдалих доріг великою кількістю феромону. Робота починається з розміщення мурах у вершинах графа (містах), потім починається рух мурах – напрям визначається імовірнісним методом, на підставі формули:

$$P_i = \frac{l_i^q f_i^p}{\sum_{k=0}^{N} l_k^q f_k^p}$$

де:

P_i — ймовірність переходу шляхом *i*;

l_i — довжина *i*-го переходу;

f_i — кількість феромонів на *i*-ому переході;

q – величина, яка визначає «жадібність» алгоритму;

p – величина, яка визначає «стадність» алгоритму ($p + q = 1$).

Результат не є точним і навіть може бути одним з гірших, проте, в силу імовірності рішення, повторення алгоритму може видавати досить точний результат.

В природі мурахи спочатку блукають довільним чином, і по знаходженні їжі повертаються до колонії, залишаючи по собі феромоновий слід. Якщо інші мурахи знаходять такий шлях, вони схильні припинити свої блукання, натомість слідувати позначеним шляхом, посилюючи його під час повернення у разі знаходження їжі (Рис. 5.3).

Однак, з часом, феромонові шляхи випаровуються, тоді привабливість шляхів зменшується. Чим більше часу потрібно мурасі, щоб подолати дорогу, тим більше часу мають феромони, щоб випаруватись. Натомість, короткий шлях проходиться частіше, отже щільність феромонів стає більшою на короткому шляху. Випаровування феромонів також надає перевагу уникнення локально найкращих шляхів. Якби випаровування не відбувалось взагалі, шляхи обрані першим мурахою тяжіли б стати вкрай привабливими для наступних. В цьому разі, розвідка можливих шляхів була б обмеженою.

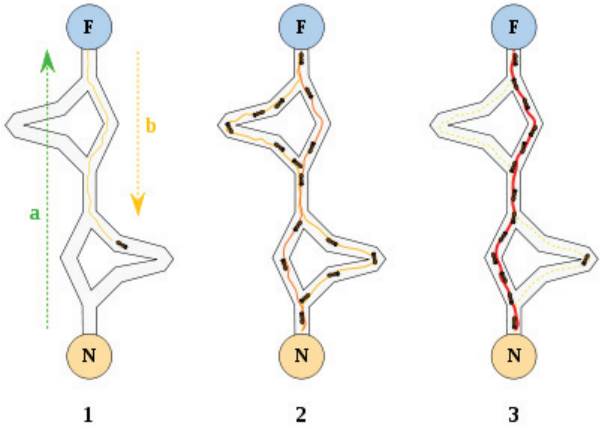


Рис. 5.3 – Переміщення мурах

Таким чином, коли мураха знаходить вдалий (тобто короткий) шлях з колонії до джерела їжі, інші мурахи швидше слідуватимуть йому, і позитивний зворотний зв'язок зрештою призведе до обрання цього шляху всіма мурахами. Ідея мурашиного алгоритму полягає в наслідуванні поведінки з «симулятором мурахи», що прогулюється графом, який представляє проблему, що треба розв'язати. Первісна ідея прийшла зі спостереження за використанням харчових ресурсів серед мурах, де мурахи, окремо обмежені в своїх пізнавальних можливостях, колективно здатні знайти найкоротший шлях між джерелом їжі і гніздом.

Перша мураха знаходить джерело їжі (*F*-Food), через якийсь шлях (а), тоді повертається до гнізда (*N*-Nest), залишивши позаду слід з феромонів (б).

Мурахи без розбору обирають всі чотири шляхи, але підсилення основної стежки робить її привабливішою як найкоротший шлях.

Мурахи обирають коротшій шлях, довгі відтинки втрачають щільність феромонового сліду.

В серії дослідів на колонії мурах з вибором між двома шляхами різної довжини до джерела їжі, біологи спостерігали, що мурахи тяжіють до використання найкоротшого шляху. Модель, що пояснює таку поведінку така:

- Мураха (що зветься «бліц») рухається більш-менш випадково по колонії.
- Якщо вона знаходить джерело їжі, вона повертається прямо до гнізда, залишаючи по собі феромоновий слід.
- Ці феромони заманливі, ближні мурахи схилятимуться до слідування, більш чи менш точно, цим шляхом.
- Повертаючись до колонії, мурахи підсилюватимуть маршрут.
- Якщо наявні два шляхи до одного й того самого джерела їжі тоді, за певний час, коротшій шлях пройде більше мурах ніж довшій.
- Короткий шлях все більш посилюватиметься, і таким чином ставатиме привабливішим.
- Довгий шлях з часом зникне, бо феромони вивітряться.
- Зрештою, всі мурахи визначатимуться і через це обиратимуть найкоротшій шлях.

Мурахи використовують навколишнє середовище як посередник для зв'язку. Вони обмінюються інформацією непрямо, а через відкладання феромонів, які уточнюють статус їхньої роботи. Інформація, що розповсюджується через феромони, має місцеву дію, лише мурахи

розташовані поруч із відкладеними феромонами помічають їх. Таку систему називають «стігмерґі» (*stigmergy*) і вона трапляється в багатьох спільнотах соціальних тварин (її вивчали на прикладі розбудови стовпів у гніздах термітів). Спільне розв'язання проблем, занадто складних для одного мурахи, є хорошим прикладом самоорганізованої системи. Система покладається на позитивний зворотний зв'язок (відкладення феромонів, які приваблюють інших мурах, що підсилять їх у свою чергу) і негативний (зникнення маршруту через випаровування забезпечує оптимальність роботи системи). Теоретично, якщо щільність феромонів залишатиметься постійною на всіх напрямках, жоден із шляхів не стане головним. Однак, через зворотний зв'язок, малі відмінності на підмаршрутах підсилюватимуться і дозволятимуть зробити вибір. Алгоритм зсуватиметься з нестабільного стану, де жоден з підмаршрутів не привабливіший за інші, у бік стабільного стану, де маршрут утворений з найсильніших підмаршрутів.

Покроковий опис загальної схеми

Припустимо, що навколишнє середовище для мурах представляє повнозв’язний неорієнтований граф. Кожне ребро має вагу, яка позначається як відстань між двома вершинами, що ним з’єднується. Граф є двохскерованим, тому мураха може подорожувати по грані в будь-якому напрямку.

Ймовірність включення ребра в маршрут окремої мурахи пропорційна до кількості феромонів на цьому ребрі, а кількість відкладеного феромону пропорційне до довжини маршруту. Чим коротший маршрут, тим більше феромону буде відкладено на його ребрах, отже, більша кількість мурах буде включати його до синтезу власних маршрутів. Моделювання такого підходу, що використовує тільки додатній зворотний зв'язок, призводить до передчасної збіжності – більшість мурашок рухається по локально-оптимальному маршруту.

Уникнути цього можна моделюючи від'ємний зворотний зв'язок у вигляді випаровування феромону. Причому, якщо феромон випаровується швидко, то це призводить до втрати пам'яті колонії і забування хороших рішень, з іншого боку, збільшення часу випару може призвести до отримання стійкого локального оптимального рішення.

Подорож мурахи

Цикли у шляху мурахи заборонено, оскільки в алгоритм включено список табу. Після завершення блукання мурахи довжина шляху може бути підрахована – вона дорівнює сумі довжин всіх ребер, якими подорожувала мураха. Наступне рівняння показує кількість феромону, який був залишений на кожному ребрі (i, j) у час t для мурашки k. Змінна Q є константою, довжина шляху – $L^k(t)$:

$$\Delta\tau_{ij}^k(t) = \frac{Q}{L^k(t)}.$$

Кількість феромону вздовж кожного ребра пройденого мурахою шляху визначається формулою:

$$\tau_{ij}(t) = \Delta\tau_{ij}(t) + \left(\tau_{ij}^k(t) \times \rho\right).$$

Важливо, що дане рівняння застосовується до всього шляху, при цьому кожне ребро позначається феромоном пропорційно до довжини шляху. Тому слід дочекатися, поки мураха закінчить подорож і лише потім оновити рівні феромону, в іншому випадку справжня довжина шляху залишиться невідомою. Константа ρ (коефіцієнт оновлення шляху) приймає значення між 0 і 1.

Випаровування феромонів

На початку шляху у кожного ребра є шанс бути обраним. Щоб поступово видалити ребра, які входять в гірші шляхи графа, до всіх ребер застосовується процедура випаровування феромону. Використовуючи константу ρ, отримуємо вираз для випаровування феромону:

$$\tau_{ij}(t) = \Delta\tau_{ij}(t) \times (1 - \rho).$$

Області застосування

Алгоритм оптимізації мурашиної колонії може бути успішно застосований для вирішення складних комплексних завдань оптимізації, мета яких – пошук і визначення найбільш відповідного рішення для оптимізації цільової функції з дискретної множини можливих рішень.

Типовими прикладами вирішення такого завдання є задачі календарного планування, завдання маршрутизації транспорту, різних мереж (GPRS, телефонні, комп'ютерні тощо), розподіл ресурсів та робіт. Дослідження показали, що метод мурашиних колоній може давати результати, навіть кращі, ніж при використанні генетичних алгоритмів і нейронних мереж.

Етапи вирішення задачі за допомогою мурашиних алгоритмів

Для того щоб побудувати відповідний мурашиний алгоритм для вирішення будь-якої задачі, потрібно:

1. Уявити завдання у вигляді набору компонент і переходів або набором неорієнтованих зважених графів, на яких мурахи можуть будувати рішення.

2. Визначити значення сліду феромону.

3. Визначити евристику поведінки мурахи, коли будуємо рішення.

4. Якщо можливо, то реалізувати ефективний локальний пошук.

5. Вибрати специфічний ACO (Ant Colony Optimization) алгоритм і застосувати для розв'язання задачі.

6. Налаштувати параметр ACO алгоритму.

Також визначальними є:

• кількість мурах;

• баланс між вивченням і використанням;

• поєднання з жадібними евристиками або локальним пошуком;

• момент, коли оновлюється феромон.

Застосування мурашиних алгоритмів для задачі комівояжера

Завдання формулюється як задача пошуку мінімального за вартістю замкнутого маршруту по всіх вершинах без повторень на повному зваженому графі з n вершинами. Змістовно вершини графа є містами, які повинен відвідати комівояжер, а ваги ребер відображають відстані (довжини) або вартості проїзду. Ця задача є NP-складною, і точний переборний алгоритм її вирішення має факторіальну складність.

Моделювання поведінки мурах пов'язано з розподілом феромону на стежці – ребрі графа у задачі комівояжера. При цьому ймовірність включення ребра в маршрут окремого мурахи пропорційна кількості феромону на цьому ребрі, а кількість відкладеного феромону пропорційна довжині маршруту. Чим коротше маршрут, тим більше феромону буде відкладено на його ребрах, отже, більшу кількість мурах буде включати його в синтез власних маршрутів. Моделювання такого підходу, що використовує тільки позитивний зворотний зв'язок, призводить до передчасної збіжності – більшість мурашок рухається за локально оптимальним маршрутом. Уникнути цього можна, моделюючи негативний зворотний зв'язок у вигляді випаровування феромону. При цьому якщо феромон випаровується швидко, то це призводить до втрати пам'яті колонії і забування хороших рішень, з іншого боку, великий час випаровування може привести до отримання стійкого локального оптимального рішення.

Тепер з урахуванням особливостей задачі комівояжера, можемо описати локальні правила поведінки мурах при виборі шляху:

1. Мурахи мають власну «пам'ять». Оскільки кожне місто може бути відвідане тільки один раз, то у кожного мурашки є список вже відвіданих міст – список заборон. Позначимо через J_{ik} список міст, які необхідно відвідати мурасі k, що знаходиться в місті i.

2. Мурахи мають «зір» – видимість є евристичне бажання відвідати місто *j*, якщо мураха знаходиться в місті *i*. Будемо вважати, що видимість обернено пропорційна відстані між містами $\eta_{ij} = 1/D_{ij}$.

3. Мурахи мають «нюх» – вони можуть вловлювати слід феромона, що підтверджує бажання відвідати місто *j* з міста *i* на підставі досвіду інших мурах. Кількість феромону на ребрі (*i*, j) в момент часу *t* позначимо через $\tau_{ij}(t)$.

4. На цій підставі ми можемо сформулювати ймовірностно-пропорційне правило, яке визначає ймовірність переходу *k*-го мурахи з міста *i* в місто *j*:

$$\begin{cases} P_{ij,k} = \dfrac{[\tau_{ij(t)}]^{\alpha}[\eta_{ij}]^{\beta}}{\sum_{l \in J_{ik}}[\tau_{il}]^{\alpha}[\eta_{il}]^{\beta}} & j \in J_{ik} \\ P_{ij,k} = 0, \quad j \notin J_{ik} \end{cases}$$

де α, β – параметри, що задають ваги сліду феромону.

При $\alpha = 0$ алгоритм вироджується до жадібного алгоритму (буде обране найближче місто). Зауважимо, що вибір міста є ймовірнісним, правило лише визначає ширину зони міста *j*. У загальну зону всіх міст J_{ik} кидається випадкове число, яке і визначає вибір мурашки. Правило не змінюється в ході алгоритму, але у двох різних мурах значення ймовірності переходу будуть відрізнятися, тому що вони мають різний список дозволених міст.

5. Пройшовши ребро (*i*, *j*), мураха відкладає на ньому деяку кількість феромону, яке повинно бути пов'язано з оптимальністю зробленого вибору. Нехай $T_{ik}(t)$ є маршрут, пройдений мурахою *k* до моменту часу *t*, $L_k(t)$ – довжина цього маршруту, *a* Q – параметр, що має значення порядку довжини оптимального шляху. Тоді кількість феромону, що відкладається, може бути задано у вигляді:

$$\Delta\tau_{ij}^{k} = \begin{cases} \dfrac{Q}{L_k(t)}, & (i,j) \in T_k(t); \\ 0, & (i,j) \notin T_k(t). \end{cases}$$

Правила зовнішнього середовища визначають, в першу чергу, випаровування феромону.

На початку алгоритму кількість феромону на ребрах приймається рівним невеликому позитивному числу. Загальна кількість мурах залишається постійною і дорівнює кількості міст; кожна мураха починає маршрут зі свого міста.

Додаткова модифікація алгоритму може складатися у веденні так званих «елітних» мурах, які підсилюють ребра найкращого маршруту, знайденого з початку роботи алгоритму. Позначимо через *T** найкращий поточний маршрут, через *L** – його довжину. Тоді, якщо в колонії є e елітних мурах, то ребра маршруту отримають додаткову кількість феромону:

$$\Delta\tau_e = e \times Q / L^*.$$

Алгоритм використання мурашиного алгоритму в задачі комівояжера виглядає наступним чином.

1. Ініціалізація матриці відстаней D.
2. Ініціалізація параметрів алгоритму $\alpha, \beta, e, Q.$
3. Ініціалізація ребер – присвоєння видимості η_{ij} і початкової концентрації феромона.
4. Розміщення мурах без збігів.
5. Вибір початкового маршруту *L**.
6. Цикл по часу життя колонії $t = 1 \ldots t_{max}$
7. Цикл по усім мурахам $k = 1 \ldots m$
8. Побудувати маршруту $L_k(t)$
9. Кінець циклу по мурахам
10. Перевірка усіх $L_k(t)$

11.Якщо $L_k(t)$ краще, то перепризначуються L^* і T^*.

12.Цикл по усіх ребрах графа

13.Оновити сліди феромону

14.Кінець циклу по ребрах

15.Кінець циклу за часом

16.Вивести найкоротший шлях T^* та його довжину L^*

Складність даного алгоритму, як можна помітити, залежить від часу життя колонії, кількості міст і кількості мурах в колонії.

Якість одержуваних рішень багато в чому залежить від параметрів у ймовірностно-пропорційному правилі вибору шляху на основі поточної кількості феромону і від параметрів правил відкладання і випаровування феромону. Можливо, що динамічне адаптаційне налаштування цих параметрів може сприяти отриманню кращих рішень. Важливу роль відіграє і початковий розподіл феромона, а також вибір умовно оптимального рішення на кроці ініціалізації.

Перспективними шляхами поліпшення мурашиних алгоритмів є різні адаптації параметрів з використанням бази нечітких правил і їх гібридизація, наприклад, застосування генетичних алгоритмів.

Питання для самоперевірки

1. Що таке м’які обчислення?
2. Назвіть класифікацію видів м'яких обчислень.
3. В чому полягає ідея мурашиного алгоритму?
4. Як застосовується мурашиний алгоритм для вирішення задачі комівояжера.

Література до розділу

1. *Заде Л.А.* Роль мягких вычислений и нечеткой логики в понимании, конструировании и развитии информационных интеллектуальных систем // Новости искусственного интеллекта. 2001. № 2-3. С. 7-11. 2. *Кричевский М.Л.* Методы исследований в менеджменте: учебное пособие. – М. :

КНОРУС, 2016. – 296 с. 3. *Ситюк В.Є.* Прогнозування. Моделі. Методи. Алгоритми: Навчальний посібник. – К.: «Маклаут», 2008. – 364 с. 4. *Ландэ Д.В., Снарский А.А., Безсуднов И.В.* Интернетика: Навигация в сложных сетях: модели и алгоритмы. – М.: Либроком (Editorial URSS), 2009. – 264 с. 5. *Штовба С.Д.* Муравьиные алгоритмы // Exponenta Pro. Математика в приложениях, 2003. – №4, – С. 70-75. 6. *Штовба С.Д.* Проектирование нечетких систем средствами MATLAB . – М.: Горячая линия – Телеком, 2007. – 288 с. 7. *Герасимов Б.М., Дивизинюк М.М., Субач И.Ю.* Системы поддержки принятия решений: проектирование, разработка, оценка эффективности: Монография. – Севастополь: издательский центр СНИЯЭиП, 2004. – 320 с. 8. *Герасимов Б.М., Субач І.Ю.* Функціональна структура підсистеми нечіткого виводу інтелектуальної системи підтримки прийняття рішень // Автоматизація виробничих процесів, 2006. – № 2(23). – С. 7–43. 9. *Субач І.Ю.* Знаходження нечітких асоціативних правил у реляційних базах даних телекомунікаційного підприємства // Зв'язок, 2005. – № 3. – С. 54–57.

6. МОДЕЛІ ІНФОРМАЦІЙНИХ ПОТОКІВ

Аналіз динаміки інформаційних потоків, що генеруються у веб-просторі стає сьогодні одним з найбільш інформативних методів дослідження актуальності тих або інших тематичних напрямків. Ця динаміка обумовлена факторами, багато з яких не піддаються точному аналізу. Однак загальний характер часової залежності кількості тематичних публікацій в Інтернеті все ж таки допускає побудову математичних моделей.

У поведінці інформаційних потоків спостерігаються дві характерні риси: по-перше, виразна тенденція до постійного зростання їх обсягів, а по-друге, ускладнення динамічної структури. Спостереження часових залежностей числа повідомлень в мережних інформаційних потоках переконливо свідчать про те, що механізми їхньої генерації та поширення, очевидно, зв'язані із складними нелінійними процесами загальної мережної динаміки.

У літературі традиційними вважаються два класи моделей інформаційних потоків: лінійні й експонентні. Серед останніх виділяється модель Бартона-Кеблера, запропонована у свій час для опису процесу старіння інформаційних ресурсів:

$$m(t) = I - ae^{-t}(-t) - be^{-2t},$$

де $m(t)$ – частка корисної інформації в загальному потоці I; перша експонента відповідає статичним ресурсам, а друга – динамічним (новинним).

Обидва класи мають істотну обмеженість – монотонний характер часової залежності. Тобто вони мало придатні для вивчення реальної динаміки мережних інформаційних потоків.

6.1. Поняття інформаційних потоків

Під тематичним інформаційним потоком будемо розуміти послідовність повідомлень, що відповідають певному тематичному запиту.

Отже, під тематичним інформаційним потоком будемо розуміти кількість документів, що у деякому змісті відносяться до заданої теми. Розглянемо загальну картину динаміки тематичних інформаційних потоків, обмежившись механізмами, типовими для динамічного сегмента Інтернет.

Численні факти свідчать про те, що в дійсності динаміка тематичних інформаційних потоків визначається комплексом внутрішніх нелінійних механізмів, які лише частково корелюють з об'єктивним оточенням. Очевидно, що ця динаміка в принципі не може бути пояснена деяким одним фактором, який повністю відповідає за всю розмаїтість ефектів, що спостерігаються. Саме ця обставина й надає особливу актуальність проблемі моделювання динаміки мережних тематичних потоків.

Загальний інформаційний потік, який вимірюється в кількості повідомлень, є величиною відносно стабільною. Змінюються в часі лише обсяги повідомлень, які відповідають тій або іншій тематиці. Іншими словами, зростання кількості публікацій по одній темі супроводжується зменшенням публікацій на інші теми, так що для кожного проміжку часу T маємо:

$$\int_0^T \sum_{i=1}^{M} y_i(t)dt = NT,$$

де $y_i(t)$ – кількість публікацій в одиницю часу, а M – загальна кількість всіх можливих тим. Звичайно, передбачається, що частина $n_i(t)$ завжди дорівнює нулю. Тобто для локальних часових проміжків можна спостерігати так званий «тематичний баланс».

Основний інтерес в такому формулюванні представляє вивчення динаміки окремого тематичного потоку, який описується щільністю $n_i(t)$.

Теоретично можна припустити, що множини публікацій, асоційованих з певним набором тематик, перетинаються, тобто існують публікації, які можуть бути віднесені одночасно до декількох різних тем. Загалом кажучи, така політематичність дійсно спостерігається, вона є ефектом, який необхідно враховувати, але в першому наближенні будемо вважати, що його внесок не спотворює загальну картину.

Подібне розуміння мережевих тематичних інформаційних потоків, мабуть, дозволяє більш менш адекватно описувати загальні закономірності їх динаміці.

У практичному плані часто виявляється цілком задовільним спрощене розуміння інформаційного потоку як деякої залежної від часу величини $X(t)$, яка описується рівнянням:

$$\frac{dX(t)}{dt} = F\left(X(t), t\right).$$

Далі, кожна тематика також має ряд характерних властивостей, які допускають деяку класифікацію, наприклад, на основі особливостей її утворення та відтворення в часі:

- публікації на «разову» тему, часова залежність числа яких різко зростає, виходить на насичення, а потім убуває та асимптотично спрямовується до нуля;
- публікації за темами, що періодично з'являються у загальному інформаційному потоці, які після закінчення обмеженого проміжку часу практично зникають з нього;
- публікації за темою, часова залежність кількості яких коливається біля деякого значення та ніколи не зникає повністю.

Відповідно до цього повідомлення можуть підрозділятися на аналогічні категорії, причому кожна з них має власну специфіку розвитку в часі.

Ще складніше виглядає синхронна зміна кількості повідомлень з декількох тематичних інформаційних потоків. Їх поведінка чітко нагадує

процеси взаємодії популяцій у біоценозах. Так, наприклад, у ряді випадків збільшення числа публікацій за однією темою супроводжується скороченням числа публікацій за іншою. Загальна динаміка у цьому випадку може описуватися системою рівнянь, кожне з яких відноситься до окремого монотематичного потоку. Підкреслимо, що загальні політематичні потоки є стаціонарними по кількості публікацій, динаміка ж в основному визначається «конкурентною боротьбою» окремих тематик.

У літературі описано багато різновидів систем «конкурентної боротьби» для різних модифікацій моделі в залежності від цілого ряду припущень щодо реальних умов протікання процесів. У найпростішому вигляді такі рівняння можуть мати такий вигляд:

$$\frac{dm_i(t)}{dt} = p_i \cdot m_i(t) - \sum_{j=1}^{N_m} r_{ij} \cdot m_i(t) \cdot m_j(t),$$

де N_m – кількість тематик.

Приведена система рівнянь описує перерозподіл публікацій між тематиками, які утворюють фіксований набір. Але в реальному житті тематики (сюжети) з'являються і з часом зникають, тому необхідно ввести в ці рівняння відповідні корективи. Це можна зробити по-різному, наприклад, визначивши коефіцієнти p_i і r_{ij} залежними від часу так, щоб кожен сюжет мав власний максимум активності на певному проміжку часу.

6.2. Моделі інформаційних потоків

Лінійна модель

У деяких випадках динаміка тематичних інформаційних потоків, що може бути виражена кількістю публікацій за певний період, її інтенсивністю, обумовленою, наприклад, зміною активності тематики (її підвищенням або старінням), відбувається лінійно, тобто кількість повідомлень у момент часу *t* можна, відповідно, представити формулою:

$$y(t) = y(t_0) + v(t - t_0),$$

де t_0 – деякий стартовий час відліку, $y(t)$ – кількість повідомлень на час t, v – середня швидкість збільшення (зменшення) інтенсивності тематичного інформаційного потоку.

Важливі характеристики інформаційного потоку можуть бути кількісно оцінені флуктуацією цього потоку – зміною середньоквадратичного відхилення $\sigma(t)$, що обчислюється за формулою:

$$\sigma(t_n) = \sqrt{\frac{1}{n}\sum_{i=0}^{n}\left[y(t_i) - \left(y(t_0) + v(t_i - t_0)\right)\right]^2}.$$

Якщо ця величина змінюється пропорційно квадратному кореню від часу, то процес зміни кількості публікацій по обраній темі можна вважати процесом з незалежними приростами. При цьому зв'язками з попередніми тематичними публікаціями можна зневажити.

У випадку, коли середньоквадратичне відхилення пропорційно деякому ступеню часу: $\sigma(t) \propto t^{\mu}$ $(\frac{1}{2} \le \mu \le 1)$, чим більше значення μ, тим вище кореляція між поточними та попередніми повідомленнями в інформаційному потоці.

Експоненціальна модель

У деяких випадках процес зміни актуальності тематики (збільшення або зменшення кількості тематичних повідомлень в інформаційному потоці в одиницю часу) апроксимується експонентною залежністю, яку можна виразити формулою:

$$y(t) = y(t_0)e^{\lambda\cdot(t-t_0)},$$

де λ – середня відносна зміна інтенсивності тематичного інформаційного потоку.

У реальності актуальність тематики є дискретною величиною, вимірюваною в моменти часу t_0, ..., t_n, яка лише апроксимується наведеною вище залежністю. У рамках даної моделі справедливо:

$$y(t_i) = y(t_0)e^{\lambda\cdot(t_i-t_0)} = y(t_0)\,\mathrm{e}^{\lambda\cdot(t_i-t_{i-1}+t_{i-1}-t_0)} = y(t_{i-1})\,\mathrm{e}^{\lambda\cdot(t_i-t_{i-1})}.$$

Звідки:

$$\frac{y(t_i)}{y(t_{i-1})} = \mathrm{e}^{\lambda\cdot(t_i-t_{i-1})}.$$

Введемо позначення: $\lambda(t_i)$ – відносна зміна інтенсивності тематичного інформаційного потоку в момент часу t_i:

$$\lambda(t_i) = \lambda\cdot(t_i - t_{i-1})$$

і прологарифмуємо наведене вище рівняння:

$$\lambda(t_i) = \ln\frac{y(t_i)}{y(t_{i-1})}.$$

Відносна зміна інтенсивності в момент часу t_i на практиці також часто обчислюється як співвідношення:

$$\lambda(t_i) = \ln\frac{y(t_i)}{y(t_{i-1})} \approx \frac{y(t_i) - y(t_{i-1})}{y(t_{i-1})}.$$

Зміна флуктуацій величини $\lambda(t_i)$ щодо середнього значення може оцінюватися за стандартним відхиленням:

$$\sigma(t_n) = \sqrt{\frac{1}{n}\sum_{i=0}^{n}\left(\lambda(t_i) - \lambda\right)^2}.$$

У цьому випадку також, якщо $\sigma(t)$ змінюється пропорційно кореню квадратному від часу, то можна говорити про процес із незалежними збільшеннями – кореляції між окремими повідомленнями несуттєві. У випадку наявності значної залежності повідомлень спостерігається співвідношення: $\sigma(t) \propto t^{\mu}$, причому μ перевищує ½, але обмежено 1.

Значення μ, що перевищує ½, свідчить щодо наявності довгострокової пам'яті в інформаційному потоці. Такий клас процесів одержав назву автомодельних, для яких передбачається кореляція між кількістю повідомлень, що публікуються у різні моменти часу.

Логістична модель

На відмінність від моделі Бартона-Кеблера у реальній динаміці інформаційних потоків мають місце як процеси росту, так і спаду кількості документів. Тому для побудови реалістичної картини, безумовно, потрібно застосовувати більш гнучку модель.

Насамперед, варто сказати, що документи в інформаційному потоці в багатьох відносинах нагадують популяції живих організмів. Вони в певному сенсі «народжуються», «вмирають» і дають «потомство» (документи, що містять інформацію, що раніше з'явилася в декількох інших документах). У сучасній науковій літературі поняття популяції часто використовується в широкому тлумаченні, і тому цілком обгрунтовано введення його і при моделюванні інформаційних потоків.

Логістичну модель можна розглядати як узагальнення експонентної моделі Мальтуса, яка передбачає пропорційність швидкості росту функції $y(t)$ в кожен момент часу її значенню:

$$\frac{dy(t)}{dt} = ky(t),$$

де k – деякий коефіцієнт.

У реальному житті, як правило, динамічні системи мають досить ефективні зворотні зв'язки, які дозволяють коригувати характер процесів, що відбуваються в них, і тим самим утримувати їх у певних рамках. Інформаційні операції, коригуючи ці зворотні зв'язки в певні періоди еволюційного процесу, можуть досить ефективно вплинути на характер поведінки всієї системи.

Найбільш простим узагальненням закону Мальтуса, яке дозволяє вирішити (принаймні, принципово) проблему необмеженого зростання розв’язку, є заміна постійного коефіцієнта k деякою функцією часу $k(t)$. Природно, ця функція повинна бути обрана таким чином, щоб дотримувалися такі умови:

- розв’язок рівняння мав би прийнятну поведінку;

- структура функції мала б певний зміст з погляду на явище, яке досліджується.

Головна ідея логістичної моделі складається в тому, що для обмеження швидкості росту на функцію $y(t)$ накладається додаткова умова, відповідно до якого її значення не повинне перевищувати деякої величини. Найпростіший спосіб обмежити зростання експонентної залежності розв'язку наведеного вище рівняння полягає у введенні для неї граничного значення. Для цього виберемо $k(t)$ такого вигляду:

$$k(t) = k \cdot [N - ry(t)],$$

де N – граничне значення, яке функція $y(t)$ не може перевищити, r – коефіцієнт, що описує негативні для даної тенденції процеси, k – коефіцієнт пропорційності. Причому передбачається, що завжди $n_0 \le N$. Тоді замість першого рівняння маємо:

$$\begin{cases} \dfrac{dy(t)}{dt} = ky(t)(N - ry(t)), \\ y(t_0) = y_0. \end{cases}$$

Модель, що заснована на наведеному вище рівнянні, називається логістичною. Незважаючи на уявну простоту, подібне узагальнення закону Мальтуса аж ніяк не є примітивним. Навпроти, воно дозволяє явно включити в опис динаміки популяцій винятково важливий зворотний зв'язок. Логістичне рівняння, власне кажучи, варто вважати феноменологічним: ми не знаємо, як діють конкретні механізми, що знижують по мірі зростання $y(t)$ швидкість її зміни. І це, у даному випадку, серйозна перевага.

Існує два класи розв'язків логістичного рівняння, які, залежно від значень коефіцієнтів k_0 і n_0, описують зростання та убування залежності $y(t)$. Їхня типова поведінка зображене на рис. 6.1. Як видно, логістична модель, на відміну від закону Мальтуса, описує досягнення системою деякого рівноважного стану.

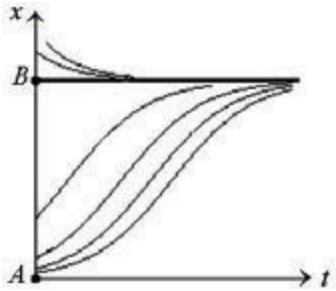


Рис. 6.1 – Узагальнена логістична модель

Наведене вище логістичне рівняння має два рівноважних розв'язки: $y(t) = 0$ і $y(t) = N$. З формальної точки зору перший з них хитким, тому що при малих значеннях $y(t)$ його відхилення від нуля приводить до зростання. Однак у практичному плані це не зовсім так. Справа в тому, що реальні обсяги інформаційних потоків є дискретними множинами, і якщо в якийсь момент $y(t)$ приймає значення, менше за одиницю, то зрости воно вже не зможе. Тому у випадку опису того, що відбувається в реальності, розв'язок $y(t) = 0$ також варто вважати рівноважним.

Друге ж рішення $y(t) = N$ є рівноважним у будь-якому сенсі. Дійсно, при $y(t) > N$ включаються механізми спаду залежності, а при $y(t) < N$ – відповідно зростання.

Розглянемо поведінку динаміки тематичного інформаційного потоку, обсяг якого визначається логістичним рівнянням. Необхідно підкреслити, що висновки, які будуть зроблені нижче, залишаються (з точністю до числових значень констант) справедливими також при будь-яких значеннях коефіцієнтів і навіть для широкого класу моделей з різними функціями обмеження експонентного зростання.

На Рис. 6.1 зображена результуюча залежність обсягів інформаційного потоку від часу при різних початкових умовах. У точках A і B швидкість зміни кількості повідомлень спрямовується до нуля: це стаціонарні стани.

Між A і B швидкість позитивна (кількість повідомлень зростає), а вище точки B – негативна (кількість повідомлень убуває).

Модель передбачає, що згодом встановлюється стаціонарний режим B, що виглядає цілком природно: більший інформаційний потік зменшується, менший – збільшується.

Логістична модель задовільно описує численні явища насичення. Поблизу A, коли обсяг інформаційного потоку малий, вона дуже близька до мальтузіанської моделі. Але при досить великих x спостерігається різка відмінність від мальтузіанського зростання: замість спрямування x до нескінченності кількості публікацій наближається до стаціонарного значення B.

Розглянемо, як логістична модель може застосовуватися під час аналізу інформаційних потоків, а саме визначення мінімальної початкової кількості c повідомлень (яку можна, наприклад, виділити для початку деякої рекламної кампанії). Нехай x – обсяг тематичного інформаційного потоку. На динаміку цієї величини здійснюється вплив інших тематик, які зменшують його розповсюдження, що описується таким чином:

$$\dot{x} = x - x^2 - c.$$

Обчислення показують, що поведінка системи різко змінюється при деякому критичному значенні c (Рис. 6.2).

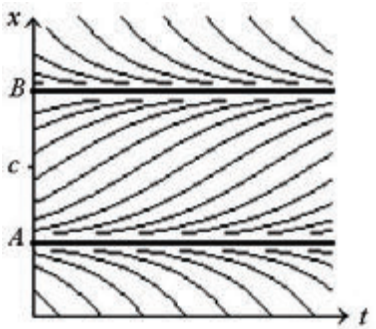


Рис. 6.2 – Дві постійні точки логістичної моделі

Якщо величина c мала, то зміни (у порівнянні з відсутністю конкуренції, коли $c=0$) полягають у наступному. Система має два рівноважних стани, A і B. Стан B стійкий: обсяг інформаційного потоку в цьому випадку трохи менша, ніж при безконкурентній ситуації. Цей обсяг відновлюється при малих відхиленнях від рівноважного значення B.

Стан A нестійкий: якщо внаслідок будь-яких причин обсяг інформаційного потоку впаде нижче рівня A, то надалі кількість тематичних повідомлень буде зведена нанівець за цілком скінченний час.

Очевидно, що при наявності сприятливих зовнішніх умов (при деякій щільності ресурсу) обсяг інформаційного потоку зростає вільно, що сприяє логістичному росту. У цьому випадку навіть більш складні моделі повинні давати результати, подібні наведеним.

Отже, логістична модель успішно описує досягнення тематичним інформаційним потоком деякого рівноважного стану.

Інформаційну динаміку в загальному випадку можемо представити як процес, обумовлений виникненням і зникненням окремих тематик, що відбуваються на тлі загальних тенденцій інформаційного простору.

Зафіксуємо певну тематику й припустимо, що в момент часу $t=0$ існує n_0 фонових публікацій. Внаслідок того, що (у рамках прийнятої моделі) актуальність тематики зберігається протягом проміжку часу λ, можна розглядати окремо дві часові області: $0<t\le\lambda$ с $D>0$ и $t>\lambda$ с $D=0$ (у рамка даної моделі $D=const$ для кожної області – рівень актуальності теми) і, відповідно, функції $u(t)$ і $v(t)$, які є рішеннями для цих областей й “зшиваються” у точці λ:

$$y(t)=\begin{cases} u(t), & 0<t<\lambda, \\ v(t), & t>\lambda, \\ u(t)=v(t), & t=\lambda. \end{cases}$$

Першій області відповідає процес зростання кількості публікацій в умовах ненульової актуальності теми ($D > 0$) і, можливо, перехід до стану насичення.

Реакція медійних засобів ніколи не буває миттєвою: завжди існує певна затримка в часі. Цей аспект ураховується в моделі шляхом введення фактору запізнювання τ.

Відповідна динаміка описується рівнянням, що після перевизначення коефіцієнтів та їхнього нормування до N, для функції $u(t)$ можна представити у вигляді:

$$\frac{du(t-\tau)}{dt} = pu(t-\tau)(1-qu(t-\tau)) + Du(t-\tau),$$

$$u(0) = n_0.$$

Підкреслимо, що змістовно величина p визначає нормовану ймовірність появи публікації в одиницю часу незалежно від актуальності теми. Цей фактор відображає фонові механізми генерації інформації (типовим прикладом може бути механічний передрук матеріалів із престижних інформаційних джерел). Величина ж D характеризує безпосередній вплив актуальності даної теми. Параметр q характеризує зменшення швидкості росту кількості публікацій і є величиною, зворотною до асимптотичного значення залежності $u(t)$ при $D = 0$.

Для другої області, описуваною функцією $v(t)$, відповідно, маємо:

$$\frac{dv(t-\lambda)}{dt} = pv(t-\lambda)(1-qv(t-\lambda)).$$

При цьому повинне враховуватися умова рівності функцій $u(t)$ й $v(t)$ у момент $t = \lambda$:

$$v(\lambda) = u(\lambda).$$

Наведені вище нелінійні диференціальні рівняння є варіантами запису рівняння Бернуллі:

$$y' = ay^2 + by,$$

яке лінеарізується стандартною заміною $z = 1/y$:

$$z' + bz + a = 0.$$

Загальне рішення цього рівняння має вигляд:

$$z = \frac{1}{\mu(x)}[C - a\int \mu(x)dx]$$

з інтегруючим множником:

$$\mu(x) = e^{bx}.$$

Змінні C для першої області визначаються виходячи із початкових умов, а для другий – з умови «зшивання». Шляхом нескладних перетворень знаходимо рішення для першої області:

$$u(t) = \frac{u_s}{1 + (\frac{u_s}{n_0} - 1)\exp[-(p + D)(t - \tau)]},$$

де u_s – асимптотичне значення u, величина якого визначає область насичення:

$$u_s = \frac{p + D}{pq}$$

Таким чином, ми бачимо, що модель правильно описує залежність, що має *S*-подібну (логістичну) форму, представлену на Рис. 6.3.

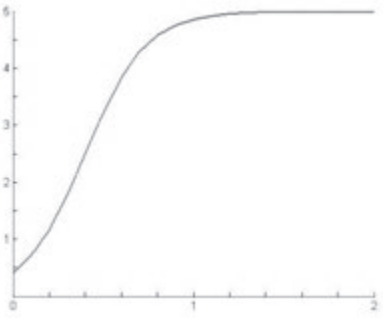


Рис. 6.3 – Функція зростання $u(t)$

Рішення не залежить від значення n_0, що свідчить про неістотність початкових умов для інформаційної динаміки. Яким би не була початкова кількість публікацій, насичення буде визначатися винятково параметрами,

які характеризують фонову швидкість зростання кількості публікацій, кількісну міру актуальності й негативні для процесу фактори.

Крива, представлена на Рис. 6.3, має точку перегину:

$$t_{\inf} = \frac{1}{p+D}\ln(\frac{u_s}{n_0} - 1) + \tau .$$

Таким чином, для першої області маємо так звану *S*-подібну залежність, а при $t \sim t_{\inf}$ поведінка $u(t)$ наближається до лінійної й відповідає лінійній моделі.

Представимо тепер для зручності вираження для $u(t)$ трохи в іншому вигляді:

$$u(t) = \frac{u_s \exp[(p+D)t]}{\exp[(p+D)t] + (\frac{u_s}{n_0} - 1)\exp[(p+D)\tau]},$$

звідки видно, що за умови

$$t < \frac{1}{p+D}\ln(\frac{u_s}{n_0} - 1) + \tau = t_{\inf}$$

залежність $u(t)$ має експонентний характер, тобто для значень *t*, значно менших $t_{\inf}$, модель збігається з експонентною моделлю.

Для другої області, відповідно, маємо (Рис. 6.4):

$$v(t) = \frac{v(\lambda)}{qv(\lambda) + (1 - qv(\lambda))\exp[-p(t-\lambda)]},$$

з огляду на умову «зшивки»:

$$v(\lambda) = u(\lambda).$$

Якщо залежність $u(t)$ встигає досягти насичення за проміжок часу $t < \lambda$, то наведене вище рівняння можна спростити, представивши його у такий спосіб:

$$v(t) = \frac{v_s(p+D)}{p + D(1 - \exp[-p(t-\lambda)])},$$

де $v_s = 1/q$ – асимптотичне значення залежності $v(t)$.

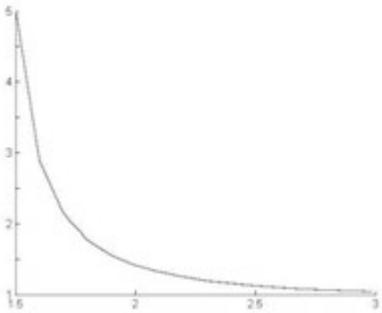


Рис. 6.4 – Функція спаду $v(t)$

Як й очікувалося, величина v_s також не залежить ні від початкової умови, ні від умови "зшивання" з функцією $u(t)$ на границі областей. Як видно, отримана залежність має область насичення u_s (при $t \le \lambda$ і асимптотику v_s, що описує поступове зменшення числа публікацій до фонового рівня. А це означає, що вона, принаймні, на якісному рівні, узгоджується із загальною уявою про характер інформаційної динаміки, що отримується на основі дослідних даних. Крім того, на локальних ділянках вона непогано апроксимується лінійною й експонентною моделями. Типова повна залежність $y(t)$ наведена на Рис. 6.5.

У випадку інформаційних потоків, які асоціюються з конкретними темами, необхідно описувати динаміку кожного з таких потоків окремо, беручи до уваги те, що ріст одного з них автоматично приводить до зменшення інших і навпаки. Тому обмеження на кількість повідомлень по всіх тематиках поширюється й на сукупність всіх монотематичних потоків.

У випадку вивчення загального інформаційного потоку спостерігається явище "перетікання" публікацій з одних тематик, до інших. Дійсно, кожен інформаційний ресурс, веб-сайт, має певні потужності, які залежать як від технічних аспектів, так і від кон'юнктури предметної області, тобто кожен ресурс публікує більш-менш стандартну кількість повідомлень в одиницю часу.

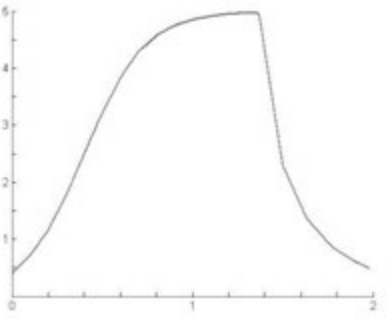


Рис. 6.5 – Узагальнений графік динаміки тематичного інформаційного потоку

Загальна динаміка повинна описуватися системою рівнянь, кожне з яких відноситься до окремого монотематичного потоку. Підкреслимо, що загальні політематичні потоки є стаціонарними по кількості публікацій, динаміка ж в основному визначається «конкурентною боротьбою» окремих тематик.

Наведену вище систему рівнянь «конкурентної боротьби» в рамках узагальненої логістичної моделі можна представити в такому вигляді:

$$\frac{dy_i(t)}{dt} = \left(p_i + D_i(t,\lambda_i)\right)\cdot\left(y_i(t) - \sum_j r_{ij}\cdot y_i(t)y_j(t)\right).$$

У цих співвідношеннях коефіцієнти p_i та D_i мають той же зміст, що й раніше, а λ_i є точками, у яких відповідні D_i досягають максимальних значень.

Безсумнівною перевагою логістичної моделі є те, що вона поєднує в собі простоту вихідних формулювань із гнучкістю в постановці завдань.

6.3. Кореляція інформаційних потоків

Канонічне логістичне рівняння описує динаміку одиничної популяції, яка взаємодіє винятково з «навколишнім середовищем». У дійсності ж подібні ситуації виникають украй рідко, оскільки популяції (інформаційні

потоки) активно взаємодіють між собою. У теорії популяційної динаміки розроблений класифікація різних форм такої взаємодії:

- нейтралізм (відсутність прямого впливу популяцій один на одного);
- конкуренція (взаємне придушення популяцій);
- аменсалізм (однобічне придушення однієї популяції);
- хижацтво (знищення особинами однієї популяції особин іншої);
- симбіоз (продуктивне співіснування популяцій).

Кожна з цих форм має варіанти, так що загальна картина взаємодії популяцій виглядає досить складною та різноманітною. При цьому варто також ураховувати, що взаємодія популяцій може бути не тільки прямою (наприклад, поїдання одним видом іншого), але й опосередкованим (наприклад, спільне споживання обмежених ресурсів).

У динаміку взаємодіючих популяцій виділяються дві категорії впливів, що відрізняються часовим характером:

- фазові (однократні);
- параметричні (постійні).

У рамках логістистичної моделі опис n взаємодіючих популяцій у загальному випадку здійснюється за допомогою системи рівнянь, записаної у такому вигляді:

$$\begin{cases} \dfrac{dm_i(t)}{dt} = m_i(t)[p_i - \sum_{j=1}^{n} q_{ij} m_j(t)] \\ m_i(0) = m_{0i} \end{cases}$$

Тут тип описуваного процесу визначається величиною та знаком коефіцієнтів p_i та q_{ij}, причому варто мати на увазі, що в кожнім рівнянні діагональні члени m_i m_i відповідають внутрішньовидовому, а перехресні m_i m_j – міжвидовій взаємодії.

Важливим моментом є також та поведінка, яку мала б популяція при відсутності взаємодії. Наведена вище система рівнянь може описувати

широкий спектр залежностей, однак її рішення (що відносяться до реальних процесів), відповідають одному з наступних режимів:

- стаціонарний;
- автоколивальний;
- квазістохастичний.

Як правило, ці режими повною мірою проявляють себе на досить великих проміжках часу.

Наведений вище опис динаміки популяцій у рамках логістичної моделі спочатку було сформульовано для біологічних систем, однак на даний час поширено також на інші області досліджень, у тому числі й тематичні інформаційні потоки.

У зв'язку з великою можливою кількістю тематик, головним моментом при моделюванні вважають характер взаємодії тематичного потоку із зовнішньої стосовно нього середовищем.

Нижче наведені результати моделювання типових ситуацій з наступним порівнянням отриманих залежностей з наборами дослідних даних. Таким чином, у першому наближенні можна виявити загальні найважливіші закономірності.

Монотематична динаміка

Найбільш простий випадок часової залежності числа повідомлень, що надходять до інформаційного потоку у зв'язку з деякою подією. У цьому випадку крива динаміки інформаційного потоку виглядає просто: спочатку вона різко зростає, досягає насичення, а потім убуває, спрямовуючись при цьому до деякого значення, близького до нуля.

Реалізація моделі є дуже спрощеною, тому необхідно побудувати її в більш адекватному варіанті. Замість прямокутної сходинки для визначення змінної інтенсивності тематики використається гладка функція, а саме, така, що відповідає розподілу Гауса:

$$R(t) = ae^{-b(t-\tau)^2}$$

Параметр τ фіксує положення максимуму часової залежності реакції мережі на подію, що пройшла.

При цьому виникає принаймні два способи інтерпретувати поведінки потоку:

- подія відбувається в початковий момент, але реакція на нього зростає поступово (свого роду інерція сприйняття);
- подія відбувається в момент t, і реакція на нього досить швидка, але вона очікувана, і тому обговорення її починається заздалегідь.

Обидва ці варіанта реалістичні та можуть зіставлятися з дослідними даними. Типова якісна залежність монотематичного інформаційного потоку від часу наведена на Рис. 6.6.

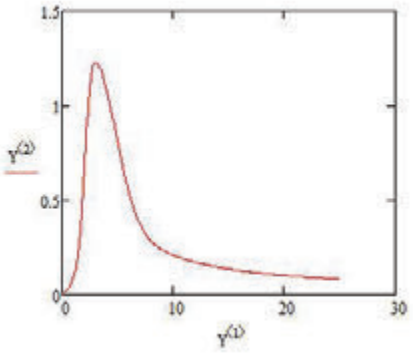


Рис. 6.6 – Якісна залежність монотематичного інформаційного потоку (вісь Y) від часу (вісь X)

Зображена на рис. 6.6 крива істотно відрізняється від стандартної гаусіани. Отже, отриманий результат не є тривіальним: вигляд функції $R(t)$ лише частково визначає результуючу форму залежності.

При виборі функції $q(t)$, що описує ефект зміни числа публікацій, викликаний зміною ефективного обсягу доступних ресурсів, необхідно виходити з того, що він може проявлятися у двох варіантах, які відрізняються перевагою того або іншого типу зворотних зв'язків: самозбудження та самозгасання. У першому випадку тема вже після

завершення подій, що її породили, стає усе більш і більше актуальною, у другому ж її актуальність постійно зменшується.

У загальному вигляді вираз для $q(t)$ виберемо у такий спосіб:

$$q(t) = (c_0 + ct)^h + d[1 + \sin(\omega t + \varphi)],$$

$$h = \pm 1.$$

Перший член у цьому виразі описує зменшення (h = 1) або збільшення (h = –1) доступної області ресурсів, а другий – періодичні її зміни (одиниця у квадратних дужках забезпечує те, що функція $q(t)$ є завжди позитивною). Типові залежності, з урахуванням наведеного виразу, представлені на Рис. 6.7 і Рис. 6.8.

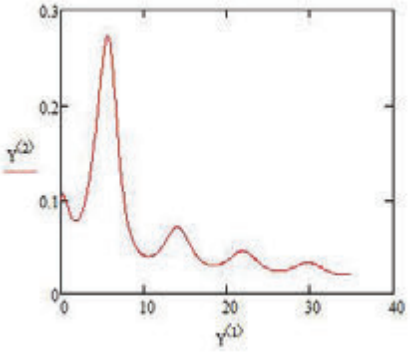


Рис. 6.7 – Випадок h = 1

Наведені співвідношення виявляються досить гнучкими, щоб описати загальну поведінку часових залежностей обсягів тематичних інформаційних потоків. Наведені формули містять досить багато параметрів, однак у практичному плані, як правило, їхній набір виявляється надлишковим.

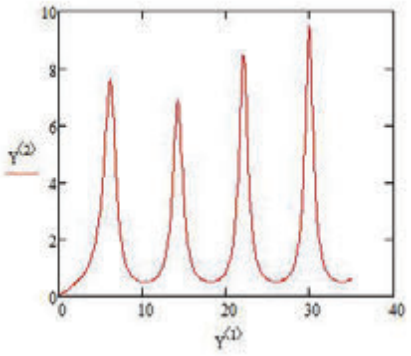


Рис. 6.8 – Випадок h = –1

Наведений модельний опис експериментальних даних на перший погляд може здатися занадто спрощеним, оскільки не становить великої складності змоделювати відповідну криву у рамках інших парадигм. Сильна ж сторона даної моделі полягає у тому, що при її простоті вона дозволяє без подальших істотних ускладнень описувати такі нетривіальні процеси, як динаміку взаємодіючих тематик.

Динаміка взаємодіючих тематик

Вивчення динаміки у випадку взаємодії тематик ускладнюється тим, що реальні інформаційні потоки містять безліч залежностей, у відношенні яких, важко сказати, які з них взаємодіють переважно між собою. Більше того, одержавши той або інший набір дослідних даних, зазвичай буває важко з повною визначеністю віднести його до взаємодіючих або невзаємодіючих тематик.

Основні типи взаємодій, що описуються системами логістичних рівнянь добре відомі й включають декілька характерних форм. Як приклад опишемо дві найцікавіші з них: конкуренцію і симбіоз.

Конкуренція

Випадку конкуренції відповідають позитивні значення обох перехресних коефіцієнтів q_{ij}. Це означає, що взаємодія тематик

відбувається таким чином, що зростання числа публікацій за однією з них супроводжується скороченням числа публікацій за іншою.

На рис. 6.9 і 6.10 наведено два характерних випадки конкуренції тематик.

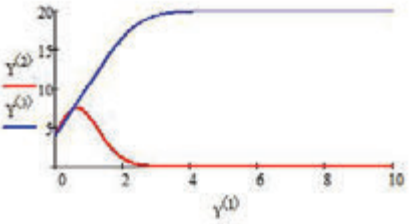


Рис. 6.9 – Конкуренція, випадок 1

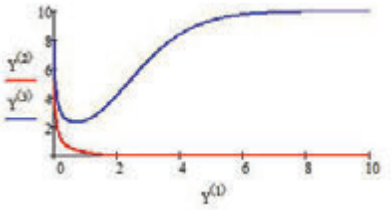


Рис. 6.10 – Конкуренція, випадок 2

Цікавим випадком конкуренції є автоколивальний режим, у якому кількості публікацій здійснюють незатухаючі коливання. Він може виникати при нульових значеннях діагональних коефіцієнтів q_{ii}.

Приклад подібної залежності зображено на рис.6.11.

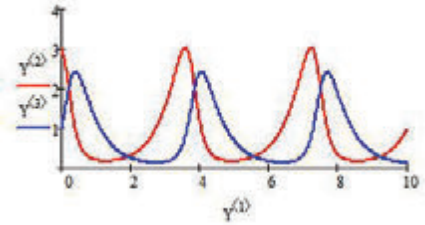


Рис. 6.11 – Конкуренція, випадок 3 – автоколивальний режим

Симбіоз

Симбіоз виникає при негативних значеннях коефіцієнтів p_2 та q_{21}, тобто при умовах, коли тематичні потоки не тільки споживають певні ресурси, але й «підживлюють» один одного. Приклад наведений на рис. 6.12.

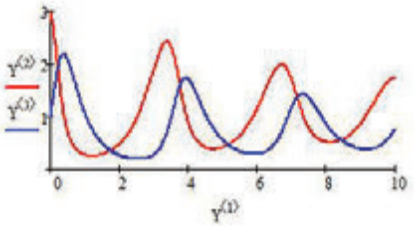


Рис. 6.12 – Симбіоз

Щоб краще зрозуміти особливості узагальненої логістичної моделі, подивимося на те, як тематики впливають одна на одну. Для цього спочатку побудуємо залежності для двох окремих тематик, які еволюціонують кожна за законом, обумовленим функціями $p(t)$ і $q(t)$, а потім дослідимо їхню спільну динаміку, за умови, що відповідні закони залишилися незмінними. Результати наведені відповідно на рис. 6.13 та 6.14.

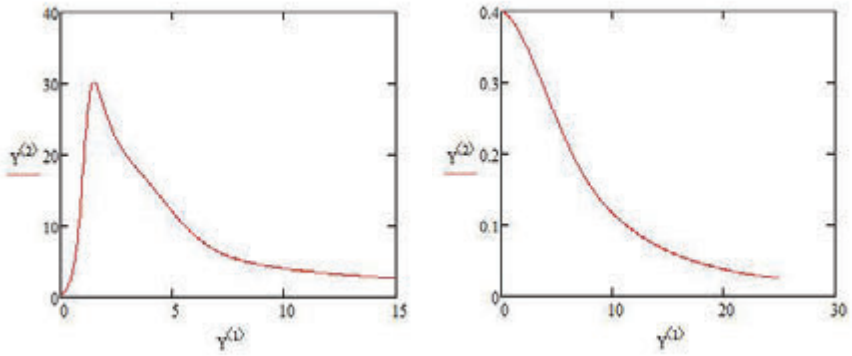


Рис. 6.13 – Роздільна еволюція тематик

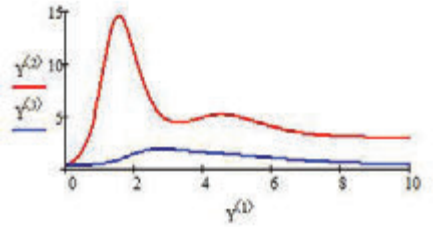


Рис. 6.14 – Сумісна динаміка тематик

З рис. 6.14 видно, що взаємний вплив тематик носить не тільки кількісний, але і якісний характер. Поведінка кожної кривої, дійсно, визначається не тільки їхніми власними функціями $p(t)$ і $q(t)$, але й характером впливу одна на одну.

6.4. Автокореляції інформаційних потоків

Важливими властивостями автокореляційної функції є можливість виявлення гармонічних складових, а також самоподібності вихідного процесу. Зупинимося більш детально на формалізмі кореляційного аналізу.

Якщо позначити через X_t член ряду кількості публікацій (наприклад, кількості електронних повідомлень, що надійшли у день t, $t = 1, \ ..., \ N$), то функція автокореляції для цього ряду X визначається як:

$$F(k) = \frac{1}{N-k} \sum_{t=1}^{N-k} (X_{t+k} - m)(X_t - m),$$

де m – середнє значення ряду X, яке надалі, не обмежуючи спільності, можна вважати рівним нулю. Коефіцієнти кореляції для рядів вимірів X довжиною N розраховуються за формулою:

$$R(k) = \frac{F(k)}{\sigma^2},$$

де $F(k)$ – функція автокореляції; σ^2 – дисперсія.

Відомо, що функція автокореляції має ту властивість, що якщо прихована періодична складова існує, то її значення асимптотично наближається до квадрату середнього значення вихідного ряду. Крім того,

функція автокореляції періодичного ряду також є періодичною, містить основну частоту та гармоніки.

Якщо вихідний ряд періодичний, тобто може бути представленій як:

$$X_t = \frac{a_0}{2} + \sum_{n=1}^{\infty} a_n \cos(n\omega t + \theta_n),$$

то його функція автокореляції буде дорівнювати:

$$F(k) = \frac{a_0^2}{4} + \frac{1}{2}\sum_{n=1}^{\infty} a_n^2 \cos n\omega k.$$

Тобто ряд вимірів X є сумішшю деякої змістовної складової N та синусоїдального сигналу S, отже функція автокореляції ряду X містить явно виражену періодичну складову, частоту і гармоніки, проте без фазових кутів θ_n. Розглянемо такий ряд:

$$X_t = N_t + S_t.$$

Знайдемо функцію автокореляції для цього ряду (значення приведені до середнього $m = 0$):

$$F(k) = \tfrac{1}{N-k}\sum_{t=1}^{N-k} X_{k+t} X_t =$$
$$= \tfrac{1}{N-k}\sum_{t=1}^{N-k} (N_{k+t} + S_{k+t})(N_t + S_t) =$$
$$= \tfrac{1}{N-k}\sum_{t=1}^{N-k} N_{k+t} N_t + \tfrac{1}{N-k}\sum_{t=1}^{N-k} S_{k+t} S_t + \tfrac{1}{N-k}\sum_{t=1}^{N-k} N_{k+t} S_t + \tfrac{1}{N-k}\sum_{t=1}^{N-k} S_{k+t} N_t.$$

Очевидно, перший доданок – це функція неперіодична, яка асимптотично спрямовується до нуля. Оскільки взаємна кореляція між N та S відсутня, то третій і четвертий доданок також прямують до нуля. Таким чином, найзначніший ненульовий внесок робить другий доданок – автокореляція сигналу S. Звідси функція автокореляції ряду X залишається періодичною.

Для експериментального підтвердження розглянутої гіпотези була згенерована послідовність, яка за своєю природою нагадує реальний інформаційний потік. Передбачалося, що щоденна кількість повідомлень в

мережі зростає за експонентним законом (з дуже невеликим значенням експонентного ступеня), і на цю кількість накладаються коливання, пов'язані з тижневою циклічністю інформаційних джерел. Також приймається до уваги елемент випадковості, виражений відповідними відхиленнями.

Для отримання відповідного часового ряду були розглянуті значення функції:

$$y = A\exp(0.001x) + \sin(\pi x / 7) + R(x),$$

яка реалізує просту модель інформаційного потоку – експонента відповідає за зростання кількості публікацій в часі (загальна тенденція), синус – за тижневу періодичність, $R(x)$ – випадкові відхилення. Кількість публікацій не може бути від'ємним числом.

На рис. 6.15 зображений графік моделі (вісь абсцис – змінна x – день, вісь ординат – змінна y – кількість публікацій).

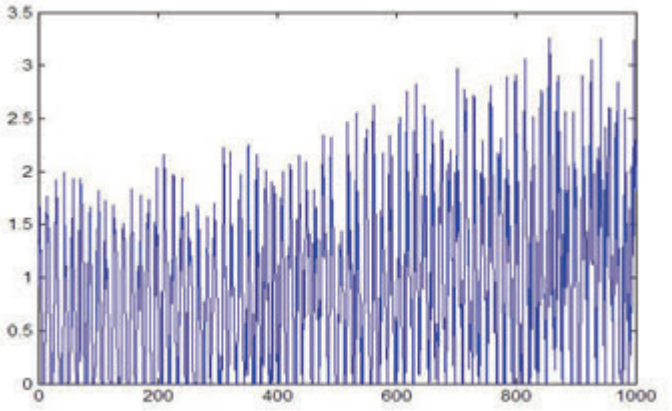


Рис. 6.15 – Модель потоку з експонентним зростанням

На Рис. 6.16 наведено графік значень коефіцієнтів кореляції (вісь абсцис – змінна k, вісь ординат – коефіцієнт кореляції $R(k)$.

Графічне представлення коефіцієнта автокореляції для ряду спостережень, що відповідає динаміці розглянутого віще тематичного інформаційного потоку свідчить про незмінність кореляційних властивостей за днями тижня (Рис. 6.17), а тренд – про можливу самоподібність вихідного часового ряду.

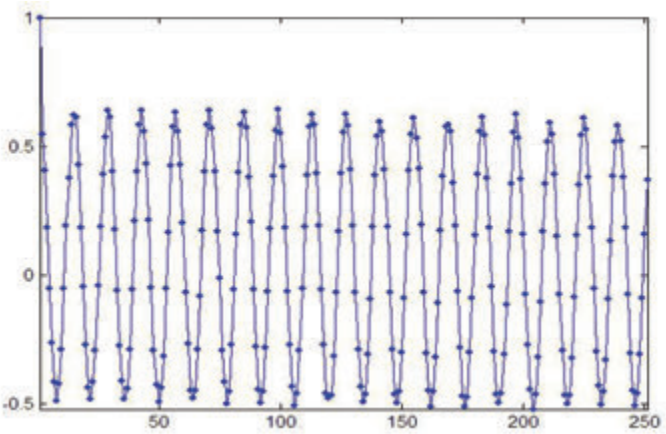


Рис. 6.16 – Значення коефіцієнтів кореляції моделі

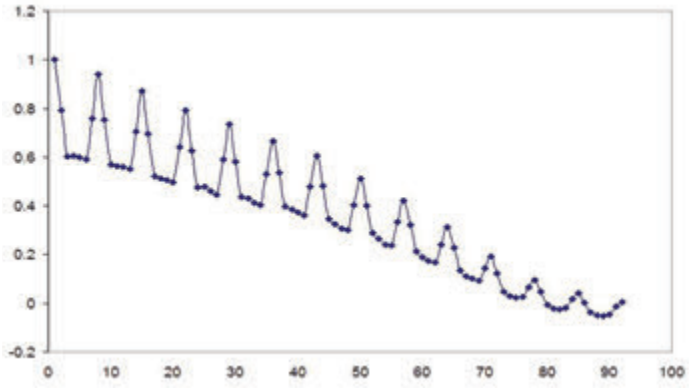


Рис. 6.17 – Коефіцієнти кореляції ряду спостережень $R(k)$ (вісь ординат) залежно від k (вісь абсцис)

Кореляційна функція «згладженого» ряду не містить явно виражених гармонік і підтверджує припущення того, що основна періодична складова даного ряду відповідає 7-денному (тижневому) циклу (Рис. 6.18). Разом з

тим, коефіцієнти кореляції ряду спостережень, усередненого за тижнями (Рис. 6.19), апроксимуються гіперболічною функцією, що свідчить про довгострокову залежність початкового ряду. Взаємна залежність членів згладженого ряду без урахування циклічної складової також підтверджується порівнянням з «перемішаним» рядом.

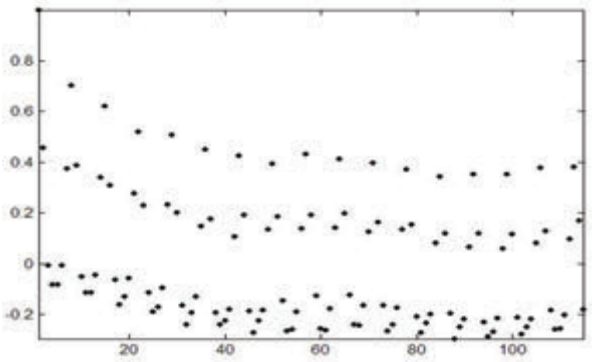


Рис. 6.18 – Коефіцієнти кореляції ряду спостережень $R(k)$ (вісь ординат) у залежності від k (вісь абсцис)

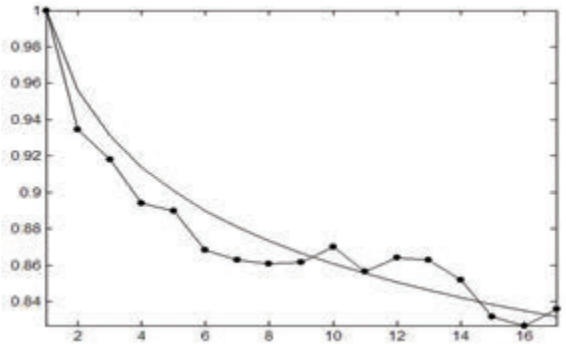


Рис. 6.19 – Коефіцієнти кореляції ряду спостережень $R(k)$, усередненого за тижнями

6.5. Розрахунок автокореляційної функції

Нижче наведена процедура обчислення автокореляційної функції ряду, що генерується. Після обчислення значень кореляційної функції забезпечується побудова відповідного графіку (Рис. 6.20).

```
clear
N=400
x=rand(1,N)
for j=3:N
   x(j)=exp(0.001*j)+sin(j/2);
end
mean(x)
for k=1:N/5
      f(k)=0;
      for t=1:N-k
         f(k)= f(k)+(x(t)-mean(x))*(x(t+k)-mean(x));
      end
      f(k)=f(k)/(N-k);
end
plot(f)
```

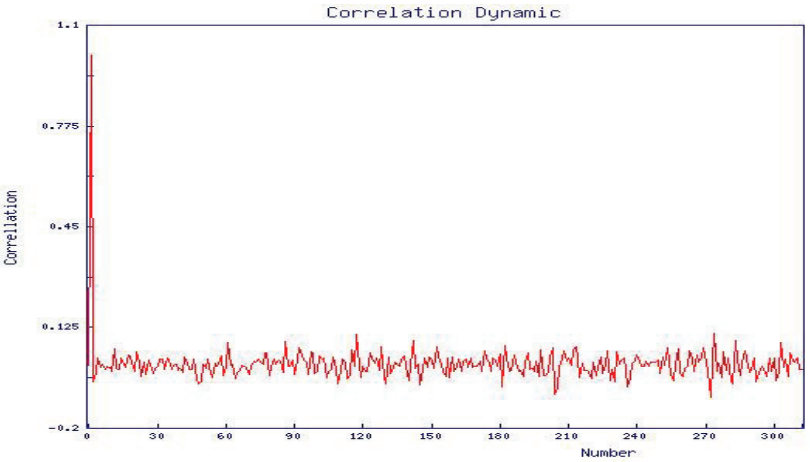


Рис. 6.20 – Кореляційна функція

6.6. Моделювання інформаційних потоків із застосуванням клітинних автоматів

Звернемося ще до одного напрямку у вивченні процесів, пов'язаних з інформаційними потоками – до дифузії інформації. Під дифузією розуміють взаємне проникнення друг у друга дотичних речовин, викликане, наприклад, тепловим рухом їхніх часток. Модель базується на тому припущенні, що інформація також у певному сенсі складається з «часток» – документів (повідомлень). Множину процесів, близьких до динаміки інформаційних потоків, можна моделювати досить точно, якщо чітко параметризувати та встановити їхні граничні параметри.

Процеси дифузії інформації, як і процеси дифузії у фізиці, досить точно моделюються за допомогою методів клітинних автоматів. Клітинні автомати є корисними дискретними моделями для дослідження динамічних систем. Дискретність моделі, а точніше, можливість представити модель у дискретній формі, може вважатися важливою перевагою, оскільки відкриває широкі можливості використання комп'ютерних технологій. Клітинні автомати в цьому сенсі займають особливе місце, оскільки їхня дискретність поєднується з іншими перевагами.

Як спрощену модель дифузії інформації спочатку розглянемо визнану модель поширення інновацій. Подібна модель функціонує за такими правилами: кожен індивід, що здатний прийняти інновацію, відповідає одній квадратній клітці на двовимірній площині. Кожна клітка може перебувати у двох станах: 1 – новинка прийнята; 0 – новинка не прийнята. Передбачається, що автомат, сприйнявши інновацію один раз, запам'ятовує її назавжди (стан 1, що не може бути зміненим). Автомат схвалює рішення відносно прийнятті новинки, орієнтуючись на думку восьми найближчих сусідів, тобто якщо в околі даної клітки (використається окіл Мура) є m прихильників новинки, p – імовірність її прийняття (генерується в ході

роботи моделі) і якщо $pm > R$, (R – фіксоване граничне значення), то клітка приймає інновацію (значення 1). На думку авторів цієї моделі, клітинне моделювання дозволяє будувати значно більше реалістичні моделі ринку інновацій, чим традиційні підходи.

Разом з тим динаміці поширення інформації властиві деякі додаткові властивості, які були враховані в представленій нижче моделі. У моделі дифузії інформації, поряд з тими ж умовами, які відносяться до клітинного простору, околу Мура та імовірнісне правило прийняття новини, додатково до в умовам дифузії інновацій передбачалося, що клітка може бути в одному із трьох станів: 1 – «свіжа новина» (клітка офарблюється в чорні кольори); 2 – новина, що застаріла, але збережена у вигляді відомостей (сіра клітка); 3 – клітка не має інформації, переданої новинним повідомленням (клітка біла, інформація не дійшла або вже забута). У моделі прийняті такі правила поширення повідомлень:

- спочатку все поле складається з білих кліток за винятком однієї, чорної, яка першою «прийняла» новину (рис. 2.26а);
- біла клітка може перефарбовуватися тільки в чорний колір або залишатися білою (вона може одержувати новину або залишатися «у невіданні»);
- біла клітка перефарбовується, якщо виконується умова, аналогічна моделі дифузії інновацій: $pm > 1$ (ця умова трохи модифікується для $m \le 2$: $1.5 \cdot pm > 1$);
- якщо клітка чорна, а навколо її винятково чорні й сірі, то вона перефарбовується в сірий колір (новина застаріває, але зберігається як відомості);
- якщо клітка сіра, а навколо її винятково сірі й чорні, то вона перефарбовується в білий колір (відбувається старіння новини при її загальновідомості).

Описана система клітинних автоматів цілком реалістично відбиває процес поширення повідомлень серед окремих інформаційних джерел. Було реалізовано наведений вище алгоритм на полі розміром 40 х 40 (розміри були обрані винятково з метою наочності). З'ясувалося, що стан системи клітинних автоматів повністю стабілізується за обмежену кількість тактів, тобто процес еволюції виявився збіжним. Приклад роботи моделі наведений на рис. 6.21.

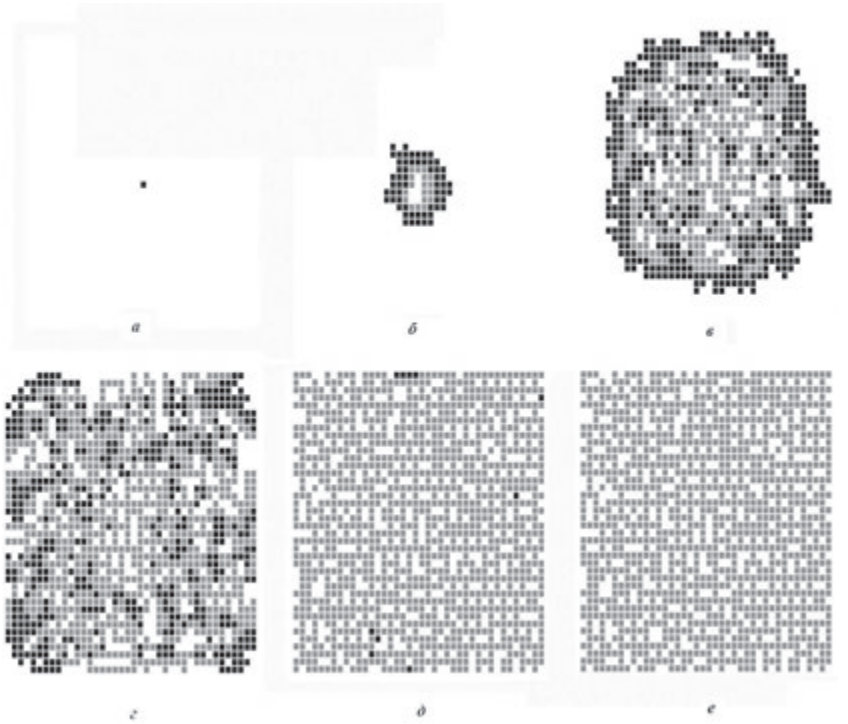


Рис. 6.21 – Процес еволюції системи клітинних автоматів «дифузії новин»: а – вихідний стан; б-д – проміжні стани; е – кінцевий стан

Численні експерименти з даним клітинним автоматом показують, що період його збіжності становить від 80 до 150 кроків.

Типові залежності кількості кліток, які перебувають у різних станах залежно від кроку ітерації наведені на рис. 6.22. При аналізі наведених графіків варто звернути увагу на такі особливості: 1 – сумарна кількість

кліток, які перебувають у всіх трьох станах на кожному кроці ітерації постійна й дорівнює кількості кліток, 2 – при стабілізації клітинних автоматів співвідношення сірих, білих і чорних кліток приблизно становить: 3:1:0; існує точка перетину всіх трьох кривих.

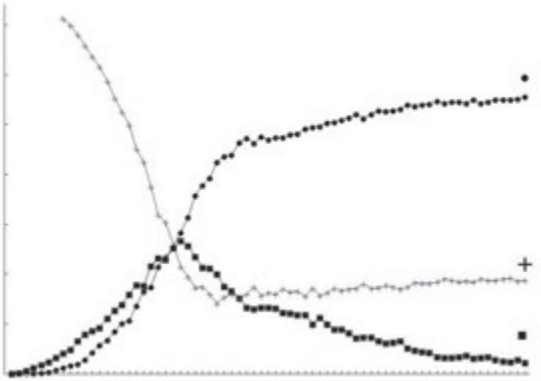


Рис. 6.22 – Кількість клітин кожного із кольорів залежно від кроку еволюції: білі клітки – (+); сірі клітки – (•); чорні клітки – (■)

Фрагмент програми мовою системи Matlab, що реалізує наведену модель дифузії інформації наведено нижче:

```
clear
colormap(gray)
for i=1:20
  for j=1:20
        c(i,j)=0;
  end
end
c(10,10)=1;
d=1-c;
pcolor(d)
% evolution
for u=1:20 % cycle
  c1=c;
```

```
    for i=2:19
        for j=2:19
          p=rand(1);
          m=c(i-1,j-1)+c(i,j-1)+c(i+1,j-1)+c(i-1,j)+c(i,j)+c(i+1,j)+
          c(i-1,j+1)+c(i,j+1)+c(i+1,j+1)
              a=1.5*p*m
              if m>2 & p*m>1
                  c1(i,j)=1;
              end
              if m<3 & a>1
                  c1(i,j)=1;
              end
              if m==8
                  c1(i,j)=0;
              end
        end
    end
    for i=2:19
      for j=2:19
          c(i,j)=c1(i,j);
      end
    end
    d=1-c;
    pcolor(d)
    pause(1)
end
```

Детальний аналіз отриманих залежностей дозволив провести аналогії даної моделі «дифузії інформації» з деякими аналітичними моделями, які дають підстави припустити, що еволюція сірих кліток описується деякою безперервною функцією:

$$x_g = f\left(t, \tau_g, \gamma_g\right),$$

де t – час (крок еволюції), τ_g – зсув за часом, що забезпечує одержання необхідного фрагмента аналітичної функції, γ_g – параметр поведінки (нахилу) даної функції.

Відповідно, динаміка білих кліток x_w (кількість кліток у момент t) може моделюватися «перевернеою» функцією x_g з аналогічними параметрами:

$$x_w = 1 - f\left(t, \tau_w, \gamma_w\right).$$

Оскільки, як було сказано вище, завжди виконується умова балансу, тобто загальна кількість кліток у будь-який момент часу завжди постійно, та умову нормування можна записати в такий спосіб:

$$x_g + x_w + x_b = 1,$$

де x_w – кількість чорних кліток у момент часу t.

Таким чином, одержуємо:

$$x_b = 1 - x_g - x_w = f\left(t, \tau_w, \gamma_w\right) - f\left(t, \tau_g, \gamma_g\right).$$

Вигляд представленої залежності дозволяє припустити, що як функція $f\left(t, \tau, \gamma\right)$ може бути обрана логістична функція:

$$f\left(t, \tau, \gamma\right) = \frac{C}{1 + e^{\gamma(t-\tau)}},$$

де C – деяка нормуюча константа.

На рис. 6.23 наведені графіки залежності x_g, x_w, x_b від кроку еволюції системи клітинних автоматів, отримані в результаті аналітичного моделювання.

Слід зазначити, що залежність дифузії новин, отримана в результаті моделювання, добре погоджується з «життєвою» поведінкою тематичних інформаційних потоків на інтернет-джерелах (веб-сайтах), а на локальних часових проміжках – із традиційними моделями.

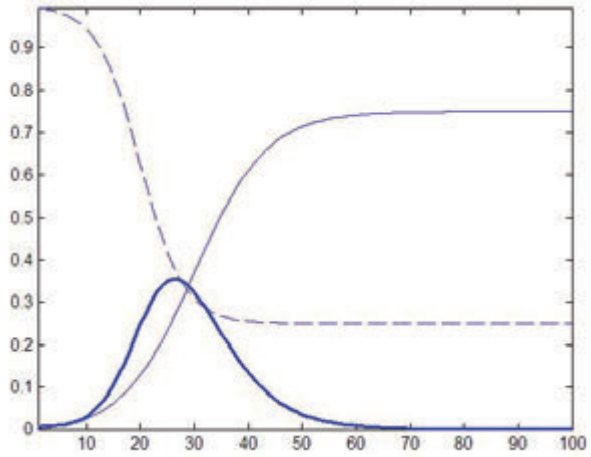


Рис. 6.23 – Безперервні залежності, отримані в результаті аналітичного моделювання, залежно від кроку еволюції: суцільна лінія – сірі (x_g); пунктирна лінія – білі (x_w); суцільна жирна лінія – чорні (x_b)

Питання для самоперевірки

1. Що таке інформаційний потік?
2. Чим характеризуються лінійна, експоненціальна та логістична моделі?
3. Як використовуються клітинні автомати для дослідження інформаційних потоків?
4. Довідатись, як за допомогою кореляційної функція може бути виявлена періодична складова, що впливає на ряд вимірів.
5. Вкажіть властивості автокореляційної функції.

Література до розділу

1. *Ландэ Д.В., Снарский А.А., Безсуднов И.В.* Интернетика: Навигация в сложных сетях: модели и алгоритмы. – М.: Либроком (Editorial URSS), 2009. – 264 с. 2. *Ландэ Д.В.* Основы интеграции информационных потоков:

Монография – К.: Инжиниринг, 2006. – 240 с. 3. *Додонов А.Г., Ландэ Д.В., Путятин В.Г.* Інформаційні потоки в глобальних комп'ютерних мережах. – К: Наукова думка, 2009. – 295 с. *4. Ситюк В.Є.* Прогнозування. Моделі. Методи. Алгоритми: Навчальний посібник. – К.: «Маклаут», 2008. – 364 с. 2. *Дуброва Т.А., Архипова М.Ю.* Статистические методы прогнозирования в экономике. Учебно-методический комплекс. – М.: Изд. Центр «ЕАОИ», 2008. – 136 с. 5. *Дубров А.М.* Многомерные статистические методы и основы эконометрики: Учебное пособие / А.М. Дубров. – М.: МЭСИ, 2008. – 79 с.

7. МЕТОДИ ФРАКТАЛЬНОГО АНАЛІЗУ

7.1. Поняття «фрактал»

Термін фрактал, був запропонований Б. Мандельбротом у 1975 році для позначення нерегулярних самоподібних математичних структур. Основне визначення фракталу, дане Мандельбротом, звучало так: "Фракталом називається структура, що складається із частин, які в якомусь змісті подібні до цілого".

Головна особливість фракталів полягає у тому, що їх розмірність не укладається у звичні геометричні уявлення. Фракталам характерна геометрична «порізанність». Тому використовується спеціальне поняття фрактальної розмірності, введене Ф. Хаусдорфом та А. Безиковичем. Розмірність фракталів не є цілим числом, характерним для звичних геометричних об'єктів.

Приклади абстрактних фракталів

Алгоритм побудови фрактальної множини Мандельброта (Рис. 7.1) заснований на ітеративному обчисленні за формулою:

$$Z[i+1] = Z[i] \times Z[i] + C,$$

де Z й C – комплексні змінні.

Ітерації виконуються для кожної стартової точки С прямокутної або квадратної області – підмножині комплексної площини. Ітераційний процес триває доти, поки $Z[i]$ не вийде за межі окружності заданого радіуса, центр якої лежить у точці (0,0), або після досить великої кількості ітерацій. Залежно від кількості ітерацій, протягом яких $Z[i]$ залишається усередині окружності, встановлюються кольори точок.

Завдяки тому, що кількість ітерацій відповідає номеру кольору, то точки, що перебувають ближче до множини Мандельброта, мають більш яскраве забарвлення.

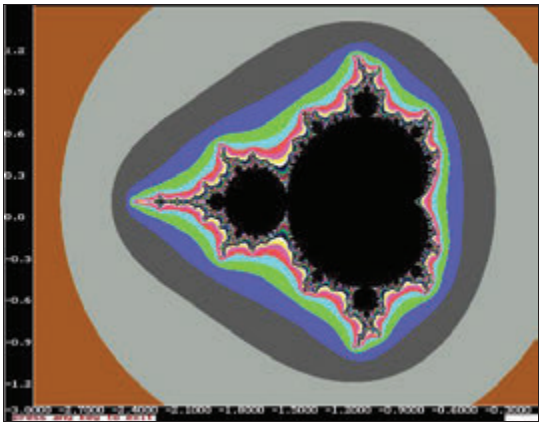


Рис. 7.1 – Множина Мандельброта

Побудова іншої фрактальної множини, сніжинки Коха (Рис. 7.2), починається із правильного трикутника, довжина сторони якого дорівнює 1. Сторона трикутника вважається базовою ланкою – вихідним положенням. Далі, на будь-якому кроці ітерації кожна ланка замінюється на утворюючий елемент – ламану, що складається по краях з відрізків довжиною 1/3 від довжини ланки, між якими розміщаються дві сторони правильного трикутника із стороною в 1/3 довжини ланки. Всі відрізки – сторони отриманої кривої вважаються базовими ланками для наступної ітерації. Крива, одержувана в результаті *n*-ї ітерації при будь-якому кінцевому *n*, називається предфракталом, і лише при *n*, що прямує до нескінченності, крива Коха стає фракталом. Одержана в результаті ітераційного процесу фрактальна множина являє собою лінію нескінченної довжини, що обмежує кінцеву площу. Дійсно, при кожному кроці число сторін результуючого багатокутника збільшується у 4 рази, а довжина кожної сторони зменшується тільки у 3 рази, тобто довжина багатокутника на *n*-й ітерації дорівнює $3*(4/3)^n$ і прямує до нескінченності з ростом *n*.

Площа під кривою, якщо прийняти площу утворюючого трикутника за 1, дорівнює:

$$S = 1 + 1/3 \sum_{k=0}^{\infty} (4/9)^k = 1{,}6.$$

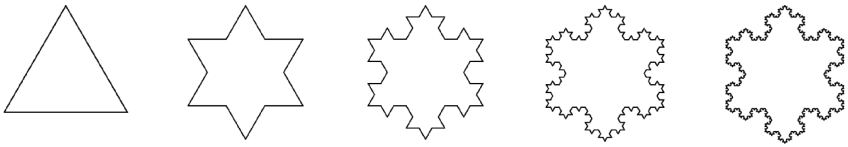


Рис. 7.2. Перші 5 поколінь сніжинки Коха

У 80-х роках XX століття як простий метод одержання фрактальних структур з'явився метод "Систем Ітераційних Функцій" (Iterated Functions System – IFS). IFS являє собою систему функцій, що відображають одну багатомірну множина на іншу. Найбільш простою реалізацією IFS є афінні перетворення площини:

$X' = A \times X + B \times Y + C;$

$Y' = D \times X + E \times Y + F.$

У 80-х роках американські вчені М. Барнслі та А. Слоан запропонували ідею стиску та зберігання графічної інформації, засновану на міркуваннях теорії фракталів і динамічних систем. На підставі цієї ідеї був створений алгоритм фрактального стиску інформації, що дозволяє стискати деякі зразки графічної інформації у 500-1000 разів. При цьому кожне зображення кодується декількома простими афінними перетвореннями.

За алгоритмом Барнслі відбувається виділення в зображенні пар областей, менша з яких подібна більшій, і збереження декількох коефіцієнтів, які кодують перетворення, що переводить більшу область у меншу. Потрібно, щоб множина таких областей покривало все зображення.

Як приклад використання IFS для побудови фрактальних структур, можна навести криву "дракона" Хартера-Хейтуея (Рис. 7.3). IFS застосовується для стиску зображень, наприклад, фотографій, що засновано на виявленні

локальної самоподібності (на відміну від фракталів, де спостерігається глобальна самоподібність).

Фрактали у природі

Один із кращих прикладів прояву фракталів у природі – структура берегових ліній. Дійсно, інколи на кілометровому відрізку узбережжя виглядає настільки ж порізаним, як і на стокілометровому.

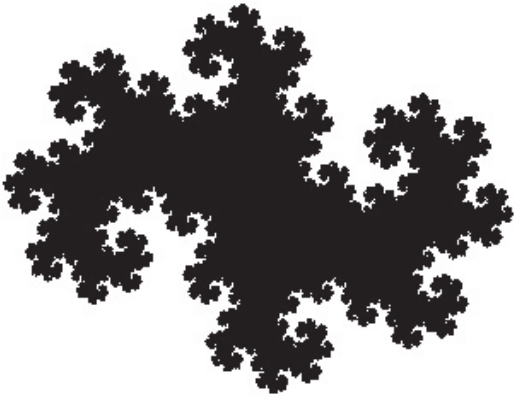


Рис. 7.3 – "Дракон" Хартера-Хейтуея

Досвід показує, що довжина берегової лінії L залежить від масштабу l, яким проводяться виміри, і збільшується із зменшенням останнього за степеневим законом $L = \Lambda l^{1-\alpha}, \Lambda = const$. Так, наприклад, для узбережжя Великобританії $\alpha \approx 1.24$, тобто фрактальна розмірність берегової лінії Великобританії дорівнює 1.24.

7.2. Інформаційний простір і фрактали

На цей час інформаційний простір прийнято розглядати як стохастичний. У багатьох моделях інформаційного простору вивчаються структурні зв'язки між тематичними множинами, що входять у цей простір. Самоподібність інформаційного простору виражається, насамперед у тому, що при його лавиноподібному зростанні, частотні та

рангові розподіли, одержувані в таких розрізах, як джерела, автори, тематика практично не міняють своєї форми. Застосування теорії фракталів при аналізі інформаційного простору дозволяє із загальної позиції глянути на закономірності, що становлять основи інформатики. Наприклад, тематичні інформаційні масиви сьогодні представляють самоподібні структури, що розвиваються і за своєю суттю є стохастичними фракталами, тому що їхня самоподібність справедлива лише на рівні математичних очікувань, як наприклад, розподілу кластерів за розмірами.

В інформаційному просторі виникають, формуються, ростуть і розмножуються кластери – групи взаємозалежних документів. Системи, засновані на кластерному аналізі, самостійно виявляють нові ознаки об'єктів і розподіляють об'єкти за новими групами.

Фрактальні властивості характерні для кластерів інформаційних веб-сайтів, на яких публікуються документи, що відповідають певним тематикам. Ці кластери, як набори тематичних документів, являють собою фрактальні структури, що мають низку унікальних властивостей.

Топологія та характеристики моделей веб-простору виявляються приблизно однаковими для різних підмножин, підтверджуючи тим самим спостереження про те, що "веб – це фрактал", тобто властивості структури всього веб-простору Bow Tie вірні і для його окремих підмножин.

З іншого боку, теорія фракталів розглядається як підхід до статистичного дослідження, що дозволяє одержувати важливі характеристики інформаційних потоків, не вдаючись у детальний аналіз їхньої внутрішньої структури та зв'язків. Для послідовності повідомлень тематичних інформаційних потоків у відповідності зі скейлинговим принципом, кількість повідомлень, резонансів на події реального миру, пропорційна деякому ступеню кількості джерел інформації (кластерів). Відомо, що всі основні закони наукової комунікації, такі як закони Парето, Лотки, Бредфорда, Ципфа, можуть бути узагальнені саме в рамках теорії

стохастичних фракталів. Точно так само, як й у традиційних наукових комунікаціях, множина повідомлень в Інтернеті за однією тематикою в часі являє собою динамічну кластерну систему, що виникає в результаті ітераційних процесів. Цей процес обумовлюється републікаціями, однобічним або взаємним цитуванням, різними публікаціями – відбиттями тих самих подій реального миру, прямими посиланнями тощо.

Фрактальна розмірність у кластерній системі, що відповідає тематичним інформаційним потокам, показує ступінь заповнення інформаційного простору повідомлень протягом певного часу:

$$N_{publ}(\varepsilon t) = \varepsilon^{\rho} N_k(t)^{\rho},$$

де N_{publ} – розмір кластерної системи (загальне число документів в інформаційному потоці); N_k – розмір — число кластерів (тематик або джерел); ρ – фрактальна розмірність інформаційного масиву; ε – коефіцієнт масштабування. У наведеному співвідношенні між кількістю документів і кластерів проявляється властивість збереження внутрішньої структури множини при зміні масштабів його зовнішнього розгляду.

7.3. Метод DFA

Для дослідження часових рядів сьогодні усе ширше використовується теорія фракталів. Часові ряди, породжувані тематичними інформаційними потоками, зокрема, мають фрактальні властивості та можуть розглядатися як стохастичні фрактали. Цей підхід розширює область застосування теорії фракталів на інформаційні потоки, динаміка яких описується засобами теорії випадкових процесів.

Метод DFA являє собою варіант дисперсійного аналізу, що дозволяє досліджувати ефекти тривалих кореляцій у нестаціонарних рядах. При цьому аналізується середньоквадратична помилка лінійної апроксимації в залежності від розміру відрізку апроксимації. У рамках цього методу спочатку здійснюється приведення даних до нульового середнього

(вирахування середнього значення $\langle F \rangle$ з вихідного часового ряду F_n, $n = 1,...,N$) і будується випадкове блукання $y(k)$:

$$y(k) = \sum_{n=1}^{N} [F(n) - < F >_N].$$

Потім ряд значень $y(k)$, $k = 1,...N$ розбивається на відрізки, що не перекриваються довжини n, у межах кожного з яких методом найменших квадратів визначається рівняння прямої, що найкраще за критерієм χ^2 апроксимує послідовність $y(k)$.

Знайдена апроксимація $y_n(k)$ ($y_n(k) = ak + b$) розглядається як локальний тренд. При цьому коефіцієнти *a* та *b* обчислюються таким чином:

$$a = \frac{n\sum ky(k) - (\sum k)(\sum y(k))}{n\sum k^2 - (\sum k)^2};$$

$$b = \frac{(\sum y(k))(\sum k^2) - (\sum k)(\sum ky(k))}{n\sum k^2 - (\sum k)^2}.$$

Далі обчислюється середньоквадратична помилка лінійної апроксимації у широкому діапазоні значень *n*. Вважається, що залежність $D(n)$ часто має ступеневий характер $D(n) \sim n^a$, тобто наявність лінійної ділянки в подвійному логарифмічному масштабі $\lg D(\lg n)$ дозволяє говорити про існування скейлингу.

Як видно на Рис. 7.4. $D(n)$ для вибраного інформаційного потоку ступеневим чином залежить від n, тобто в подвійному логарифмічному масштабі ця залежність близька до лінійної.

Нижче наведено фрагмент програмного коду мовою системи Matlab, що розраховує значення $D(n)$ для спеціально згенерованої послідовності, що моделює інформаційний потік.

```
clear
N=400; % кількість точок вихідного ряду
% Генерація вихідного ряду x(j)
x=rand(1,N);
for j=2:N
x(j)=2*x(j-1)*sin(x(j))+x(j);
end
% Розрахунок середнього – функція mean
sred=mean(x);
% Розрахунок блукання y(i)
for i=1:N
xx(i)=x(i)-sred;
end
y(1)=xx(1);
for i=2:N
y(i)=y(i-1)+xx(i);
end
% Виведення графіку y(i)
plot(y);
% Відрізки довжиною n
for n=3:N/10
for k=1:N
    f(k)=y(k);
    x=0;
    sx=0;
    sx2=0;
    sy=0;
    sxy=0;
    for j=-round(n/2):round(n/2)
       if (k+j>0 && k+j<N+1)
```

```
            sx=sx+(k+j);
            sx2=sx2+(k+j)^2;
            sy=sy+y(j+k);
            sxy=sxy+(j+k)*y(j+k);
            x=x+1;
        end
    end
    a=(x*sxy-sx*sy)/(x*sx2-sx^2);
    b=(sy*sx2-sx*sxy)/(x*sx2-sx^2);
    f(k)=a*k+b;
    d(k)=0
    for j=1:n
        d(k)=d(k)+(f(k)-y(n*(k-1)+j))^2;
    end
   D(n)=0;
   for k=1:round(N/n)-1
      D(n)=D(n)+d(k)
   end
   D(n)=sqrt(D(n)/N);
end
D'
plot(D);
```

Як видно по Рис. 7.4, $D(n)$ для побудованого модельного ряду ступеневим чином залежить від n, тобто в подвійному логарифмічному масштабі ця залежність близька до лінійної.

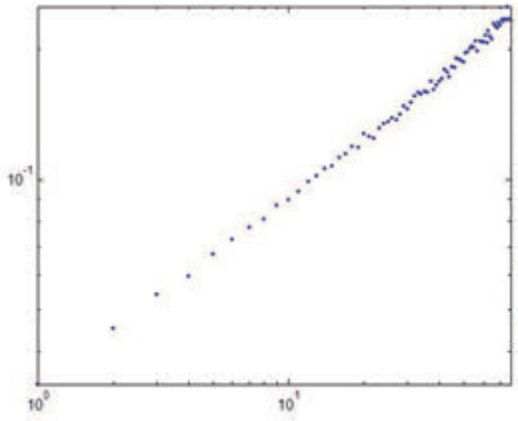


Рис. 7.4 – Залежність $D(n)$ модельного ряду (вісь Y) від довжини відрізку апроксимації n (вісь X) у подвійній логарифмічній шкалі

7.4. Фактор Фано

Для вивчення поводження процесів і підтвердження їх самоподібності прийнято використати ще один показник – індекс розкиду дисперсії (IDC), так званий фактор Фано. Ця величина визначається як відношення дисперсії числа подій (у нашому випадку – точок ряду) на вікні спостережень заданої ширини k до відповідного математичного очікування:

$$F(k) = \sigma^2(k) / m(k).$$

Для самоподібних процесів виконується співвідношення:

$$F(k) = 1 + Ck^{2H-1},$$

де C й H – константи. На Рис. 7.5. наведено графік значень $F(k)$ у подвійному логарифмічному масштабі.

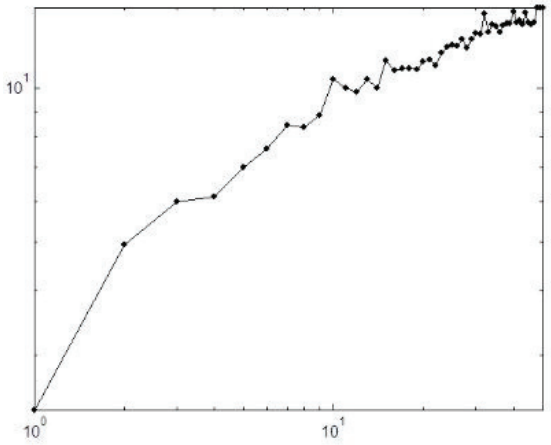


Рис. 7.5 – Залежність фактора Фано від ширини вікна спостережень

7.5. Показник Херста

Показник Херста (*H*) зв'язують із коефіцієнтом нормованого розмаху (*R/S*), де *R* – «розмах» часового ряду, а *S* – стандартне відхилення.

Херст експериментально виявив, що для багатьох часових рядів справедливо: $R/S = (N/2)^H$. Показник Херста пов'язується з традиційною фрактальною розмірністю (*D*) простим співвідношенням:

$$D = 2 - H.$$

Відомо, що показник Херста являє собою міру персистентності – схильності поведінки процесу до трендів (на відміну від звичайного броунівського руху). Значення $H > 1/2$ означає, що спрямована в певну сторону динаміка процесу в минулому, найімовірніше, спричинить продовження руху у тому ж напрямку. Якщо $H < 1/2$, то прогнозується, що процес змінить спрямованість. $H = 1/2$ означає невизначеність – броунівський рух.

Для вивчення фрактальних характеристик часових рядів $F(n)$, $n = 1, \ldots, N$, досліджуються значення показника Херста, який визначається із співвідношення:

$$R(N)/S_N \cong \left(N/2\right)^H, \quad N >> 1.$$

Тут S_N – стандартне відхилення:

$$S_N = \sqrt{\frac{1}{N}\sum_{n=1}^{N}\left(F(n) - \left\langle F\right\rangle_N\right)^2},$$

$$\left\langle F\right\rangle_N = \frac{1}{N}\sum_{n=1}^{N} F(n),$$

R – так званий розмах:

$$R(N) = \max_{1\le n\le N} X(n,N) - \min_{1\le n\le N} X(n,N),$$

де:

$$X(n,N) = \sum_{i=1}^{n}\left(F(i) - \left\langle F\right\rangle_N\right).$$

Якщо розмір ряду вимірів N досить великий, то розрахунок середнього значення і стандартного відхилення при зростанні N за наведеною формулою не є раціональним. Більш економними з обчислювальної точки зору є ітераційні методи. Обґрунтуємо формулу розрахунку середнього $\left\langle F\right\rangle_{N+1}$, якщо відомо N, $F(N+1)$, $\left\langle F\right\rangle_N$:

$$\left\langle F\right\rangle_{N+1} = \frac{1}{N+1}\sum_{n=1}^{N+1} F(n) = \frac{1}{N+1}\left(\sum_{n=1}^{N} F(n) + F(N+1)\right) = \frac{1}{N+1}\left(N\left\langle F\right\rangle_N + F(N+1)\right).$$

Формула для обчислення стандартного відхилення S_{N+1} на основі значень N, $F(N)$, $F(N+1)$, $\left\langle F\right\rangle_{N+1}$, S_N обґрунтовується дещо складніше:

$$S_{N+1}^2 = \frac{1}{N+1}\sum_{n=1}^{N+1}\left(F(n) - \left\langle F\right\rangle_{N+1}\right)^2,$$

$$(N+1)S_{N+1}^2 = \sum_{n=1}^{N+1}\left(F(n)^2 - 2F(n)\left\langle F\right\rangle_{N+1} + \left\langle F\right\rangle_{N+1}^2\right) =$$

$$= \sum_{n=1}^{N} F(n)^2 + F(N+1)^2 - 2\left\langle F\right\rangle_{N+1}\sum_{n=1}^{N} F(n) - 2\left\langle F\right\rangle_{N+1} F(N+1) + (N+1)\left\langle F\right\rangle_{N+1}^2.$$

Те ж саме для попереднього значення:

$$NS_N^2 = \sum_{n=1}^{N}\left(F(n)^2 - 2F(n)\langle F\rangle_N + \langle F\rangle_N^2\right) = \sum_{n=1}^{N}F(n)^2 - N\langle F\rangle_N^2,$$

Звідки:

$$\sum_{n=1}^{N}F(n)^2 = NS_N^2 + N\langle F\rangle_N^2.$$

Підставивши це значення у рівняння для S_{N+1}^2, отримуємо:

$$S_{N+1}^2 = \frac{1}{N+1}\left(NS_N^2 + N\langle F\rangle_N^2 + F(N+1)^2 - (N+1)\langle F\rangle_{N+1}^2\right).$$

Наведені формули ітеративного розрахунку будуть використовуватися в програмі мовою Matlab, фрагмент якої (для фіксованого значення *N*) наведено нижче.

```
clear
N=400
% Генерація вихідного ряду f(j)
f=rand(1,N);
for j=2:N
        f(j)=2*f(j-1)*sin(f(j))+f(j);
end
F=mean(f);
S=std(f);

for t=1:N
   x(t)=0;
   for j=1:t
     x(t)=x(t)+f(j)-F;
   end
end
mi=min(x)
ma=max(x)
R=ma-mi;
```

H=log (R/S)/log(N/2)

Після виконання відповідної програми (для різних значень *N*) в інтерфейсі користувача буде відображено список значень R/S. Далі цей список через буфер обміну завантажується до інтерфейсу програми Microsoft Excel. Після цього засобами цієї програми будується графік, у якому вказується логарифмічна шкала для обох осей (Рис. 7.6). Використовуючи засоби апроксимації можна переконатися у справедливості степеневого співвідношення для значень R/S.

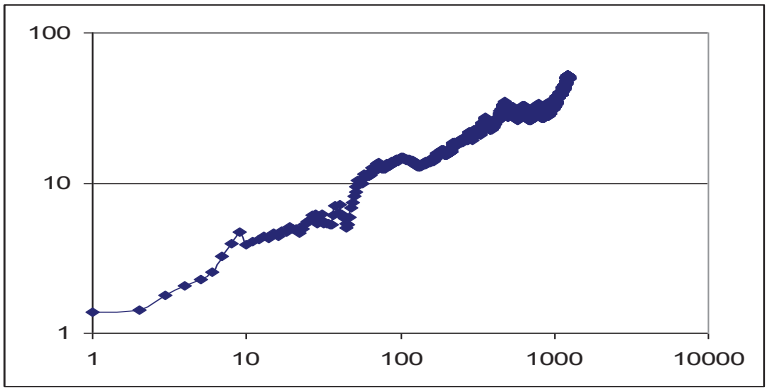


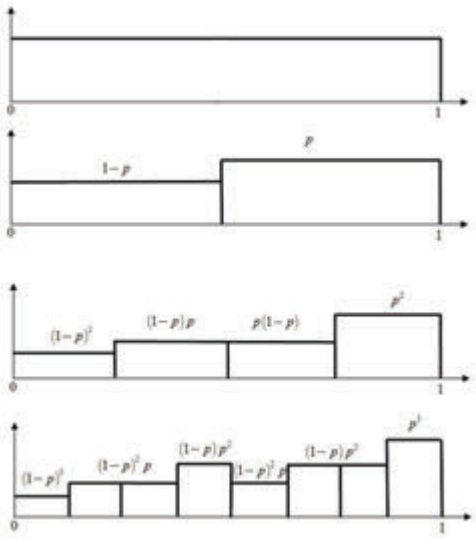
Рис. 7.6 – Динаміка зміни показника значень R/S для модельного ряду у подвійній логарифмічній шкалі

7.6. Множина Кантора

Найбільш загальний опис природи самоподібних об'єктів дає теорія мультифракталів, які характеризуються нескінченною ієрархією розмірностей, і дозволяють відрізнити однорідні об'єкти від неоднорідних. Концепція мультифрактального формалізму дає ефективний інструмент для вивчення і кількісного опису широкого різноманіття складних систем.

Поняття мультифракталу можна пояснити за допомогою спеціальної моделі, яка описує процедуру розподілу спадщини (наприклад, золота) між поколіннями спадкоємців. Дана модель реалізує так зване неоднорідне множина Кантора (Y. Kantor), що будується в такий спосіб. На початку

передбачається, що все «золото» приписується відрізку – цьому відрізку відповідає «предок» – персона, 100% спадщини якої буде розподілятися (рис. 7.7 а).



а)

б)

в)

г)

Рис. 7.7 – Питомі частини спадщини (вісь ординат): а – вихідний стан; б – перший крок моделі; в – другий крок; г – третій крок

На першій ітерації спадщина ділиться між двома спадкоємцями (в рамках всієї цієї моделі у кожного «предка» є рівно по два спадкоємця) на дві нерівні частини – старший спадкоємець отримує p-у частину, а молодший $(1\text{-}p)$-у частину (ці пропорції фіксуються і для наступних поколінь спадкоємців), при цьому для визначеності дотримуються умови: $1 > p > (1- p) > 0$. Відповідно, відрізок $[0, 1]$ ділиться на дві частини $(0, 0.5)$ – перший відрізок відповідає молодшому спадкоємцю, а другий $[0.5, 1]$ – старшому (рис. 7.7 б).

Наступний, другий крок, аналогічний першому. Кожен відрізок, що відповідає першому поколінню спадкоємців, знову ділиться на два рівні (з точністю до однієї середньої точки) частини, а спадкоємці отримують свої

частки, старші (праві відрізки) p-у, молодші (ліві відрізки) (1- p)-у. Відповідно, найбагатший спадкоємець на цьому кроці отримує p^2-у частину (p > 0.5) початкового «золота», а найбідніший -$(1- p)^2$-у (рис. 7.7 в).

Вже на третьому кроці (рис. 7.7 г) з'ясовується, що спадщина розподіляється досить складним чином. Для аналізу цього розподілу зручно розглянути деревоподібну структуру, представлену на рис. 7.8.

Нескладно помітити існуючий у цій моделі зв'язок між двійковим записом числа і величиною спадщини. Справді, виберемо на даному відрізку осі абсцис деяку точку x^*, наприклад $x^* = 1/5$, двійкове розкладання якої наступне: 1/5® 0.00110011... Кожен нуль у двійковому запису означає перехід вліво (L) по дереву, представленому на Рис. 7.8, а одиниця – перехід вправо (R). З одного боку, виконуючи ці переходи *LLRRLLRR*... ми наближаємося до $x^* = 1/5$ відповідно до процедури, заданої у розглянутій моделі. З іншого боку, кожен крок вправо дає множення величини спадщини попереднього покоління на p, а вліво на 1- p.

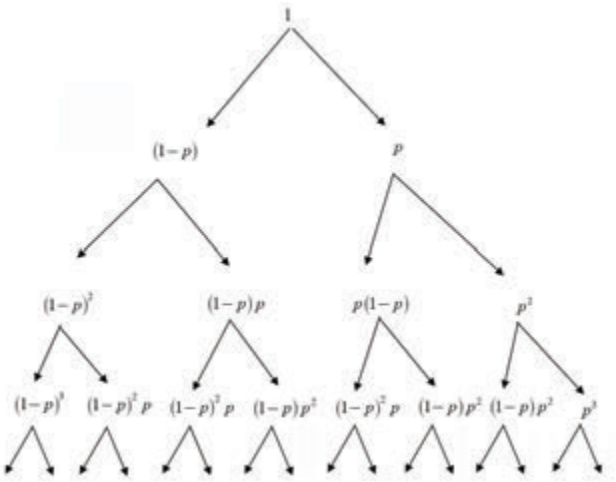


Рис. 7.8 – Дерево покрокового розподілення спадку

Наприклад, для шостого кроку, для відрізка, всередині якого лежить $x^* = 1/5$, частка початкової спадщини становить $p^2(1-p)^4$, а для n-го кроку:

$$x^* \to p^k(1-p)^{n-k},$$

де на n-ому кроці маємо k нулів і $n-k$ – одиниць.

На цьому же кроці є n відрізків розміром $1/2^n$, з них $p^k(1-p)^{n-k}$ «золота» мають $C_n^k = n!/(n-k)!k!$ відрізків (нащадків). Останнє визначає кількість способів, при яких при проходженні по дереву (рис.7.8) можна прийти до величини, що дорівнює $p^k(1-p)^{n-k}$ (кількість способів розміщення k нулів і $n-k$ одиниць).

Таким чином, повна ймовірність зустріти відрізок зі значенням $p^k(1-p)^{n-k}$ (або нащадка з кількістю «золота» $p^k(1-p)^{n-k}$) дорівнює $C_n^k p^k(1-p)^{n-k}$.

При $n>>1$, використовуючи формулу Стірлінга, можна дати наступну оцінку величини C_n^k: $C_n^k \approx \left(\frac{k}{n}\right)^k \left(1-\frac{k}{n}\right)^{k-n} = 2^{nH\left(\frac{k}{n}\right)} = \left(\frac{1}{2^n}\right)^{-H\left(\frac{k}{n}\right)}$,

де $H(x) = -x\log_2 x - (1-x)\log_2(1-x)$, $x = k/n$.

Графік функції $H(x)$ наведено на Рис. 7.9.

Множина усіх відрізків із заздалегідь заданим значенням k/n є фрактальним (при $n \to 1$) , для якого достатньо просто обмислюється фрактальна вимірність D.

Виходячи з того, що на кожному кроці n розмір кожного відрізку дорівнює $1/2^n$, а всього таких відрізків – C_n^k, отримаємо:

$$D = \lim_{n\to\infty} \frac{\log_2 C_n^k}{\log_2 1/2^n} = H\left(\frac{k}{n}\right).$$

Таким чином, завдяки властивостям функції $H(x)$ (см. Рис. 7.9), існує багато спектрів фрактальних розмірностей.

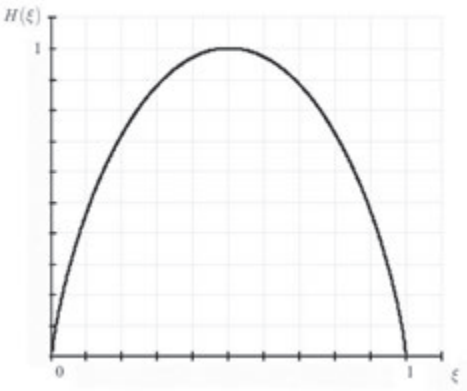


Рис. 7.9 – Графік функції $H(x)$

В даній моделі розподілення спадщини величина k/n визначає усіх нащадків, які мають однакову кількість «золота» (визначену частину початкової спадщини). Кількість таких нащадків зростає при рості n та збільшується по фрактальному закону. При крайніх значеннях $k = 0$ і $k = n$ фрактальна розмірність дорівнює нулю. Множина багатих нащадків, що мають p^n «золота» складається з однієї людини, те ж саме справедливо і для самого бідного, що має $(1- p)^n$ початкової спадщини.

7.7. Мультифрактали

Носієм мультифрактальної міри є множина L – об'єднання фрактальних підмножин L_α. Тобто мультифрактал можна визначити як деяке об'єднання різноманітних однорідних фрактальних підмножин L_α початкової множини L, кожне з яких має своє власне значення фрактальної розмірності.

Для характеристики мультифрактальної множини використовують, так звану функцію мультифрактального спектру $f(\alpha)$ (спектр сингулярностей мультифракталу), до якого підходив би термін «фрактальна розмірність». Величина $f(\alpha)$ дорівнює хаусдорфовій розмірності однорідної фрактальної підмножини L_α з початкової множини

L, яка дає домінуючий вклад в деяку статистичну суму (під час розподілення при заданих значеннях деякого порядку моментів q).

Крім того, для визначення мультифракталу використовуються узагальнені фрактальні розмірності D_q. Відповідно до мультифрактального формалізму, загальні фрактальні розмірності D_q визначаються співвідношенням:

$$D_q = \lim_{r \to 1} \frac{1}{q-1} \frac{\ln \sum_{i=1}^{N} p_i^q}{\ln r},$$

де p_i - ймовірність того, випадкова величина (нормований по загальній сумі елемент числового ряду) потрапляє в деякий діапазон r.

Далі вводиться показник мультифрактального скейлинга τ, який визначається на основі значень D_q і q:

$$\tau(q) = (1-q)D_q.$$

Функції $f(\alpha)$ і $\tau(q)$ зв‘язані друг з другом співвідношенням:

$$\tau(q) = f(\alpha) - q\alpha,$$

де α як функція від q визначається з рішення рівняння:

$$\frac{d}{d\alpha}\left(q\alpha - f(\alpha)\right) = 0.$$

Якщо відома фрактальна розмірність D_q (або показник мультифрактального скейлинга $\tau(q)$), то мультифрактальний спектр може бути знайдений по формулі:

$$f(\alpha(q)) = \tau(q) + q\alpha(q),$$

де

$$\alpha(q) = -\frac{d\tau(q)}{dq}.$$

Ці співвідношення задають криву $f(\alpha)$ параметрично (як функцію від параметра q) і представляють собою, так зване перетворення Лежандра від змінних q і τ до змінних α і f.

При аналізі ряду динаміки подій використовувався наступний метод розрахунку мультифрактальних характеристик. Значення досліджуваного ряду нормуються $Z_i = \frac{\xi_i}{\sum_k \xi_k}$ і асоціюються з вірогідністю p_i в рамках наведеної вище формули для розрахунку узагальнених фрактальних розмірностей D_q.

Після нормування весь діапазон значень ряду $\left[0,\ N\right]$ розділяється на $n = N/m$ комірок (ділянок) довжиною m. Після цього визначається наступна сума:

$$S_m^Z(q) = \sum_{k=1}^{n} \left(\bar{Z}_k^{(m)}\right)^q,$$

де

$$\bar{Z}_k^{(m)} = \sum_{l=1}^{m} Z_{(k-1)m+l}.$$

Як виявилося для рядів, що задаються динамікою публікацій, $\log S_m^Z(q)$ добре апроксимується лінійною залежністю від $\log m$, в результаті чого з'явилася можливість говорити, що числовий ряд – мультифрактал. Нахил апроксимуючої лінії, отриманий методом найменших квадратів – $\tau(q)$ визначався формулою:

$$\log S_m^Z(q) \cong \tau^Z(q) \log m + \text{const}.$$

Наведені нижче розрахунки відносяться до аналізу числового ряду, що відображує проблематику використання антивірусного програмного забезпечення (щодобова динаміка публікацій в інтернет-новинах повідомлень по даній темі, що отримується за допомогою системи контент-моніторингу), а також ряду, отриманого за уточненою тематикою (початковий запит був розширений словом «троянській»). Відповідні добові діаграми приведені на Рис. 7.10.

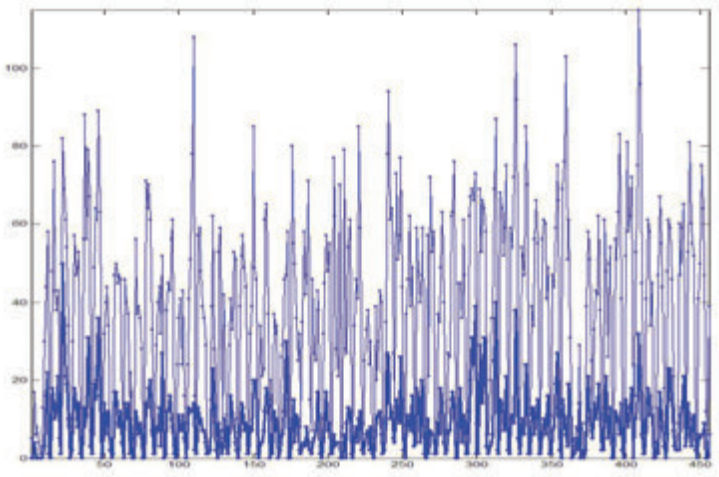


Рис. 7.10 – Діаграми інтенсивності публікацій по основній (тонка сполучна лінія) і уточненої (жирна лінія) тематики: вісь абсцис – порядкові номери днів, вісь ординат – кількість публікацій

На Рис. 7.11 показана поверхня – залежність $\tau(m, q)$ від q і m для динаміки появлення документів. Відповідно до формули:

$$f(\alpha(q)) = \tau(q) - q\tau'(q),$$

було визначено мультифрактальний спектр досліджуваного ряду.

У багатьох мультифрактальних дослідженнях основним об'єктом аналізу є залежність мультифрактального спектра f від індексу сингулярності (показника Ліпшиця-Гельдера) α. Дана залежність для рядів, відповідних основної і уточненої тематики представлена на Рис. 7.12.

Отже, ряди, відповідної динаміки появи публікацій, в розглянутих випадках мають мультифрактальну природу. Разом з тим відповідні досліджуваним рядам залежності (Рис. 7.13), мають різні параметри кривизни. Цей факт свідчить, з одного боку, про те, що ряд, відповідний підтематиці менш стабільний, ніж ряд, відповідний всій тематиці, а з

іншого боку, про те, що розглянута підтематика не є репрезентативною для аналізу потоку публікацій по загальній тематиці.

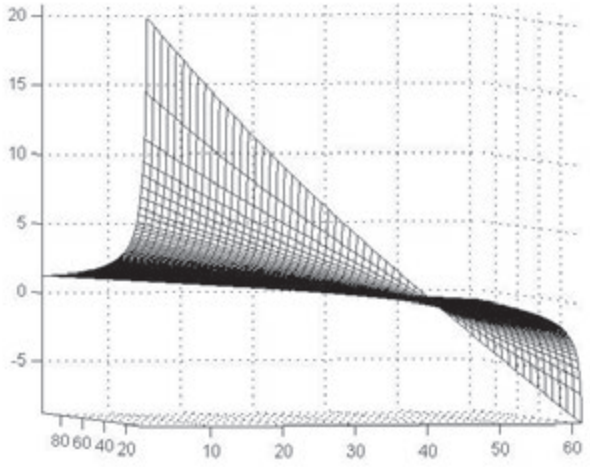


Рис. 7.11 – Значення $\tau(q,m)$ для досліджуваного ряду (запит «банк»)

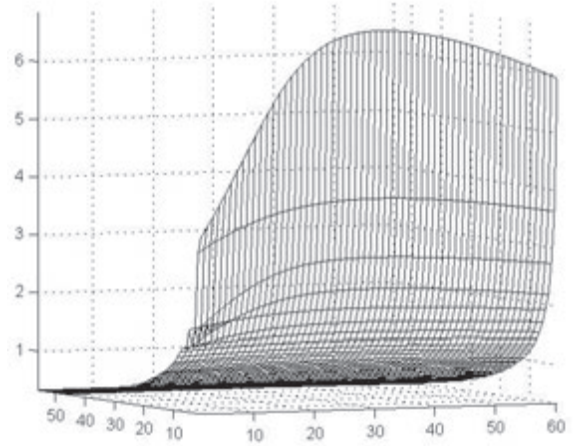


Рис. 7.12 – Значення мультифрактального спектру $f(q,m)$ для досліджуваного ряду

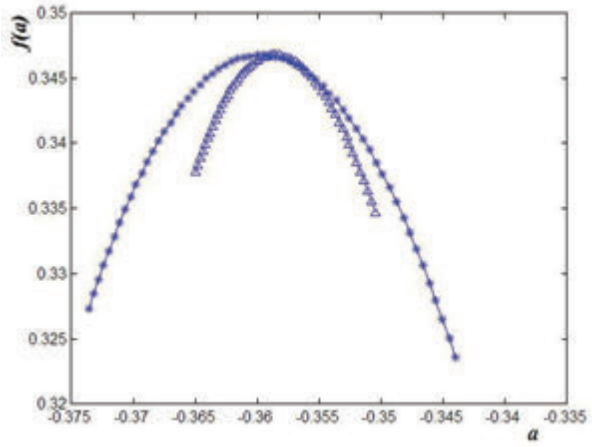


Рис. 7.13 – Порівняння мультфрактальних спектрів досліджуваних рядів – за основною тематикою (#) і уточненою (*) від індексу сингулярності

У свою чергу, для формування репрезентативних вибірок з масивів документів, може бути застосований підхід, заснований на подібності мультифрактальних спектрів, який доповнює традиційні методи, що базуються на виявленні змістовної подібності документів. Практична цінність завдання виявлення репрезентативних вибірок, заснована на даному підході, може бути виражена в таких додатках, як пред'явлення користувачеві доступних для огляду результатів пошуку, що відображають весь спектр документального масиву (з урахуванням коливань інтенсивності публікацій по днях) або виділення підмножин документів для подальших детальних досліджень.

7.8. Розрахунок мультифрактального спектру

Для дослідження властивостей мультфракталів для модельних рядів розробляється програма мовою Matlab, фрагмент якої наведено нижче:

```
clear;
N=400;
x=rand(1,N);
su=x(1);
```

```
for j=2:N
   x(j)=2*x(j-1)*sin(x(j))+x(j);
   su=su+x(j);
end
for j=1:N
   x(j)=x(j)/su;
end
plot(x);
z=round(N/4);
for m=2:z  % Ширина n-полоси
   n=round(N/m);  % n – Кількість полос
   tau_old=0;
   for q=-1:0.1:1
      S=0;
      for k=1:n-1
         t=0;
         for j=1:m
            t=t+x((k-1)*m+j);
         end
         S=S+t^q;
      end
      tau1=(log(S))/log(m);
      a(round(q*10)+11,m-1)=(tau1-tau_old)/0.1;
      fa(round(q*10)+11,m-1)=-q*a(round(q*10)+11,m-1)+tau1;
      tau_old=tau1;
   end
   fa(1,m-1)=fa(2,m-1);
   a(1,m-1)=a(2,m-1);
   if m==5
      plot(a(:,m-1),fa(:,m-1));
   end
end
```

Питання для самоперевірки

1. Назвіть основне визначення фракталу?
2. В чому полягає метод DFA?

3. Як показник Херста пов'язаний з фрактальними властивостями часового ряду?
4. Дайте визначення мультифракталу.
5. Яким чином можна візуалізувати показники мультифракталу?

Література до розділу

1. *Ландэ Д.В., Снарский А.А., Безсуднов И.В.* Интернетика: Навигация в сложных сетях: модели и алгоритмы – М.: Либроком (Editorial URSS), 2009. – 264 с. 2. *Додонов А.Г., Ландэ Д.В., Путятин В.Г.* Компьютерные сети и аналитические исследования. – К.: ИПРИ НАН Украины, 2014. – 486 с. 3. *Мандельброт Б.* Фрактальная геометрия природы. – М.: Институт компьютерных исследований, 2002. – 656 с. 4. *Федер Е.* Фракталы. – М.: Мир, 1991. – 254 с. 5. *Гринченко В.Т., Мацыпура В.Т., Снарский А.А.* Введение в нелинейную динамику. Хаос и фракталы. Изд. 2. – М: УРСС, 2007. – 263 с. 6. *Божокин С.В., Паршин Д.А.* Фракталы и мультифракталы. – Ижевск: НИЦ «Регулярная и хаотическая динамика», 2001. – 128 с.

8. ВЕЙВЛЕТ-АНАЛІЗ ДАНИХ

На цей час успішно розвивається новий і важливий напрямок в теорії і технології обробки сигналів, зображень та часових рядів, що отримав назву вейвлет-перетворення, яке добре пристосоване для вивчення структури неоднорідних процесів.

Вейвлети представляють собою особливі функції у вигляді коротких хвиль (сплесків) з нульовим інтегральним значенням і з локалізацією по осі незалежної змінної (t або x), здатних до зсуву по цій осі і до масштабування (розтягування / стиснення). Будь-який з найбільш часто використовуваних типів вейвлетів породжує повну ортогональну систему функцій.

У разі вейвлет-аналізу (декомпозиції) процесу (сигналу) у зв'язку зі зміною масштабу вейвлети здатні виявити відмінність в характеристиках процесу на різних шкалах, а за допомогою зсуву надають можливість аналізувати властивості процесу в різних точках на всьому досліджуваному інтервалі.

8.1. Узагальнений ряд Фур'є

Відомо, що довільна функція $f(t)$, для якої виконується умова

$$\int_{t_2}^{t_2}\left[f(t)\right]^2 dt < \infty,$$

може бути представлена ортогональною системою функцій $\{\varphi_n(t)\}$, тобто

$$f(t) = C_0\varphi_0(t) + \ldots + C_n\varphi_n(t) + \ldots = \sum_{n=0}^{\infty} C_n\varphi_n(t),$$

де коефіцієнти визначаються із співвідношення:

$$C_n = \frac{1}{\|\varphi_n\|^2}\int_{t_1}^{t_2} f(t)\varphi_n(t)dt,$$

де

$$\|\varphi_n\|^2 = \int_{t_1}^{t_2} \varphi_n^2(t)\,dt \quad -$$

квадрат норми, або енергія базисної функції $\varphi_n(t)$. При цьому передбачається, що ніяка з базисних функцій не дорівнює тотожно нулю і на інтервалі ортогональности $[t_1, t_2]$ виконується умова:

$$\int_{t_1}^{t_2} \varphi_n(t)\varphi_k(t)\,dt = \begin{cases} \|\varphi_n(t)\|^2, & k = n, \\ 0, & k \neq n. \end{cases}$$

Базисна функція $\varphi_n(t)$, для якої квадрат норми дорівнює одиниці $\left(\|\varphi_n(t)\|^2 = 1\right)$, називається нормованою (нормальною), а вся система функцій $\{\varphi_n(t)\}$ – ортонормованою або ортонормальною. У цьому випадку говорять, що задано ортонормований базис.

Ряд розкладення, у якому коефіцієнти C_n визначаються за наведеною вище формулою, називається узагальненим рядом Фур'є.

Добуток вигляду $C_n\varphi_n(t)$, що входить до цього ряду, являє собою спектральну складову сигналу $f(t)$, а сукупність коефіцієнтів $\{C_0, ..., C_n, ...\}$ називається спектром сигналу.

Суть спектрального аналізу сигналу $f(t)$ полягає у визначенні коефіцієнтів C_n (експериментально або аналітично). На основі ряду можливий синтез (апроксимація) сигналів при фіксованому числі N ряду

$$\hat{f}(t) = C_0\varphi_0(t) + ... + C_n\varphi_N(t) = \sum_{n=0}^{N} C_n\varphi_n(t),$$

При цьому узагальнений ряд Фур'є має наступну важливу властивість: при заданій системі базисних функцій $\{\varphi_n(t)\}$ і числі доданків N він забезпечує найкращий синтез (апроксимацію), даючи мінімум середньоквадратичної помилки ε, під якою розуміється величина:

$$\varepsilon = \int_{t_1}^{t_2} \left[f(t) - \hat{f}(t) \right]^2 dt = \int_{t_1}^{t_2} \left[f(t) - \sum_{n=0}^{N} C_n \varphi_n(t) \right]^2 dt.$$

Ортогональна система називається повною, якщо збільшенням N можна зробити ε як завгодно малим. Ряд розкладу у цьому випадку називається збіжним у середньоквадратичному.

Відносна помилка μ синтезу визначається за формулою:

$$\mu = \varepsilon / \mathrm{E},$$

де E – енергія сигналу, що чисельно рівна квадрату норми сигналу $f(t)$, тобто

$$E = \|F\|^2 = \int_{t_1}^{t_2} \left[f(t) \right]^2 dt .$$

Ця формула із урахуванням розкладу в ряд може бути записана як:

$$E = \|F\|^2 = \sum_{n=0}^{\infty} C_n^2 \|\varphi_n\|^2 ,$$

а при використанні ортонормованій системи функцій $\{\varphi_n(t)\}$ як:

$$E = \sum_{n=0}^{\infty} C_n^2 .$$

Щодо вибору раціональної системи ортогональних функцій

Вирішення цього питання залежить від поставленого завдання. Так при аналізі і синтезі сигналів, що впливають на лінійні ланцюги, найбільшого поширення набула система гармонійних функцій. По-перше, гармонійні коливання на відміну від інших зберігають свою форму при проходженні через ці ланцюги; змінюються лише амплітуда і початкова фаза. По-друге, широко використовується добре розроблений в теорії ланцюгів символічний метод. Із множини інших завдань найбільш важливою є задача наближеного розкладання складних сигналів, при якій необхідна точність забезпечується при мінімумі членів ряду.

Для представлення безперервних сигналів застосовуються поліноми і функції Лагерра, Лежандра, Чебишева, Ерміта і ін. Для подання сигналів з точками розриву використовуються кусочно-постійні функції Уолша, Хаара, Радемахера. Для дискретизації безперервних сигналів у часі використовується ортогональний ряд Котельникова. В останні роки широко використовуються базисні функції типу вейвлетів.

8.3. Вейвлети

До кола сучасних інструментальних засобів оцінки рядів спостережень відноситься також вейвлет-аналіз. Він особливо ефективний у тих випадках, коли крім загальних спектральних характеристик потрібно виявляти локальні в часі особливості поведінки процесу, що досліджується. Основою вейвлет-аналізу є вейвлет-перетворення, яке є особливим типом лінійного перетворення, базисні функції якого (вейвлети) мають специфічні властивості. Аналіз даних з використанням вейвлет-перетворень є зручним, надійним і потужним інструментом дослідження часових рядів і дозволяє представити результати у наочному вигляді, зручному для інтерпретації.

Вейвлетом (малою хвилею) називається деяка функція, зосереджена в невеликій околиці деякої точки та різко убутна до нуля по мірі видалення від неї як у часовий, так і в частотній області. Існують найрізноманітніші вейвлети, що мають різні властивості. Разом з тим, усі вейвлети мають вигляд коротких хвильових пакетів з нульовим інтегральним значенням, локалізованих на часовій осі, які є інваріантними до зсуву і до масштабування.

До будь-якому вейвлету можна застосувати дві операції:

- зрушення, тобто переміщення області його локалізації в часі;
- масштабування (розтягання або стиск).

Головна ідея вейвлет-претворення полягає в тому, що нестаціонарний часовий ряд розподіляється на окремі проміжки (так званні

«вікна спостереження»), і на кожному з них виконується обчислення скалярного добутку (величини, що показує ступінь близькості двох закономірностей) досліджуваних даних із різними зрушеннями деякого вейвлету на різних масштабах. Вейвлет-перетворення генерує набір коефіцієнтів, за допомогою яких представляється початковий ряд. Вони є функціями двох змінних: часу і частоти, і тому утворюють поверхню у трьохвимірному просторі. Ці коефіцієнти, що показують, наскільки поведінка процесу в даній точці аналогічно вейвлету на даному масштабі. Чим ближче вигляд аналізованої залежності в околі даної точки до вигляду вейвлету, тим більшу абсолютну величину має відповідний коефіцієнт. Негативні коефіцієнти показують, що залежність схожа на "дзеркальне відбиття" вейвлету.

Технологія використання вейвлетів дозволяє виявляти одиничні та нерегулярні «сплески», різкі зміни значень кількісних показників у різні періоди часу, зокрема, обсягів тематичних публікацій в Інтернет. При цьому можуть виявлятися моменти виникнення циклів, а також моментів, коли за періодами регулярної динаміки настають хаотичні коливання.

Часовий ряд може апроксимуватися кривою, яка, у свою чергу, може бути представлена у вигляді суми гармонійних коливань різної частоти й амплітуди. При цьому коливання, що мають низьку частоту, відповідають за повільні, плавні, великомасштабні зміни значень часового ряду, а високочастотні – за короткі, дрібномасштабні зміни. Чим сильніше змінюється описувана даною закономірністю величина на даному масштабі, тим більшу амплітуду мають складові на відповідній частоті. Таким чином, досліджуваний часовий ряд можна розглядати в частотно-часовій області, у залежності як від часу, так і від частоти.

Вейвлет-перетворення одновимірного сигналу / функції - це його представлення у вигляді узагальненого ряду або інтеграла Фур'є по системі базисних функцій

$$\psi_{ab}(t) = \frac{1}{\sqrt{a}}\psi\left(\frac{t-b}{a}\right),$$

сконструйованих з материнського (вихідного) вейвлету $\psi(t)$, що володіє певними властивостями за рахунок операцій зсуву в часі (b) і зміни часового масштабу (a). Множник $1/\sqrt{a}$ забезпечує незалежність норми цих функцій від масштабуючого числа a.

Отже, для заданих значень параметрів a і b функція $\psi_{ab}(t)$ і є тим вейвлетом, що породжується материнським вейвлетом $\psi(t)$. У якості базисних функцій, що утворюють ортогональний базис, можна використовувати широкий набір вейвлетів.

Для практичного застосування важливо знати ознаки, якими неодмінно повинна володіти початкова функція, щоб стати вейвлетом. Наведемо тут основні з них.

Обмеженість. Квадрат норми функції повинен бути скінченним:

$$\|\psi\|^2 = \int_{-\infty}^{\infty} |\psi(t)|^2 dt < \infty.$$

Локалізація. Вейвлет-перетворення на відміну від перетворення Фур'є використовує локалізовану вихідну функцію і у часі, і у частоті. Для цього достатньо, щоб виконувалися умови:

$$|\psi(t)| \le C(1+|t|)^{-1-\varepsilon} \quad i \quad |f_{\psi}(\omega)| \le C(1+\omega)^{-1-\varepsilon}, \text{ при } \varepsilon > 0.$$

Наприклад, дельта-функція $\delta(t)$ і гармонійна функція не задовольняють необхідній умові одночасної локалізації у часовій і частотній областях.

Нульове середнє. Графік вихідної функції повинен осцілювати (бути знакозмінним) навколо нуля на осі часу і мати нульову площу:

$$\int_{-\infty}^{\infty} \psi(t)\,dt = 0.$$

З цієї умови стає зрозумілим вибір назви «вейвлет» – маленька хвиля.

Рівність нулю площі функції $\psi(t)$, тобто нульового моменту, призводить до того, що Фур'є-перетворення $f_{\psi}(\omega)$ цієї функції дорівнює нулю при ω = 0 і має вигляд смугового фільтра. При різних значеннях параметра *a* це буде набір смугових фільтрів.

Часто для додатків буває необхідно, щоб не тільки нульовий, а й всі перші *n* моментів дорівнювали нулю:

$$\int_{-\infty}^{\infty} t^n \psi(t)\, dt = 0.$$

Вейвлети *n*-го порядку дозволяють аналізувати більш тонку (високочастотну) структуру сигналу, пригнічуючи його складові, що змінюються повільно.

Автомодельність. Характерною ознакою вейвлет-перетворення є його самоподібність. Усі вейвлети конкретного сімейства $\psi_{ab}(t)$ мають те саме число осциляцій, що і материнський вейвлет $\psi(t)$, оскільки отримані з нього за допомогою масштабних перетворень (*a*) і зсуву (*b*).

Найбільш поширені дійсні базиси конструюються на основі похідних функції Гауса ($g_0(t) = \exp\left(-t^2/2\right)$). Це обумовлено тією обставиною, що функція Гауса має найкращі показники локалізації як у часовій, так і у частотній областях. При *n* = 1 отримуємо вейвлет першого порядку, так званий WAVE-вейвлет з рівним нулю нульовим моментом. При n = 2 отримуємо МНАТ-вейвлет, «мексиканський капелюх» (Mexican Hat). У нього нульовий і перший моменти дорівнюють нулю. Він має кращу розподільну здатність, ніж WAVE-вейвлет. Приклади графіків базових вейвлетів наведено на Рис. 8.1.

Неперервне вейвлет-перетворення

Неперервне пряме і зворотне вейвлет-перетворення для функції $f(t)$ будується за допомогою неперервних масштабних перетворень і переносів

вибраного вейвлету $\psi(t)$ з довільними значеннями масштабного коефіцієнта a та параметра зсуву b:

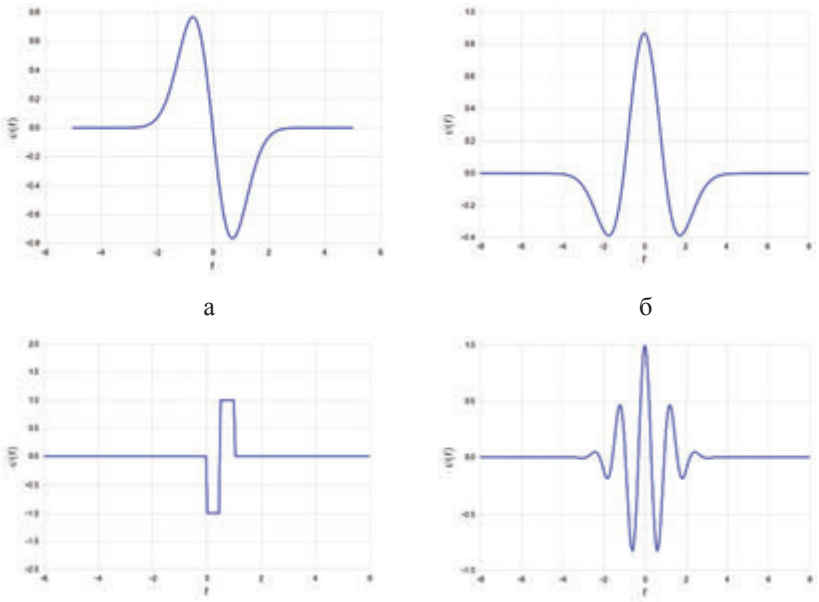
$$W_f(a,b) = \left(f(t), \psi_{ab}(t)\right) = \frac{1}{\sqrt{a}} \int_{-\infty}^{\infty} f(t) \psi\left(\frac{t-b}{a}\right) dt,$$

$$f(t) = \frac{1}{C_\psi} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} W_f(a,b) \psi_{ab}(t) \frac{dadb}{a^2},$$

де C_ψ – нормуючий коефіцієнт

$$C_\psi = \int_{-\infty}^{\infty} |\Psi(\omega)|^2 |\omega|^{-1} d\omega < \infty,$$

(., .) - скалярний добуток відповідних співмножників, $\Psi(\omega)$ – Фур'є-перетворення вейвлету $\psi(t)$. Для ортонормованих вейвлетів C_ψ.



а б в г

Рис. 8.1 – Приклади вейвлетів, що часто використовуються у додатках: (а) гаусова хвиля (перша похідна гаусом функції), (б) мексиканський капелюх, (в) вейвлет Хаара, (г) вейвлет Морлі (дійсна частина).

Таким чином, вейвлет-спектр $W_f(a,b)$ (wavelet spectrum, або time-scale-spectrum – масштабно-часовий спектр) на відміну від Фур'є-спектра (single spectrum) є функцією двох аргументів: перший аргумент a (часової масштаб) аналогічний періоду осциляцій, тобто обернений частоті, а другий b – аналогічний зміщенню сигналу по осі часу.

Слід зазначити, що $W_f(a_0,b)$ характеризує часову залежність (при $a=a_0$), тоді як залежності $W_f(a,b_0)$ можна поставити у відповідність частотну залежність (при $b=b_0$).

Якщо досліджуваний сигнал $f(t)$ являє собою одиночний імпульс тривалістю τ_u, зосереджений в околі $t=t_0$, то його вейвлет-спектр матиме найбільше значення в околі точки з координатами $a=\tau_u$, $b=t_0$.

Отримані коефіцієнти представляються у графічному вигляді картою коефіцієнтів перетворення, або скейлограмою. На скейлограмі по одній осі відкладаються зрушення вейвлету (вісь часу), а по іншій – масштаби (вісь масштабів), після чого точки схеми, що отримується, офарбовуються залежно від величини відповідних коефіцієнтів (чим більше коефіцієнт, тим яскравіше кольори зображення). На скейлограмі видні всі характерні риси вихідного ряду: масштаб та інтенсивність періодичних змін, напрямок і величина трендів, наявність, розташування та тривалість локальних особливостей.

Наприклад, відомо, що комбінація декількох різних коливань може мати настільки складну форму, що виявити їх візуально не представляється можливим. Періодичні зміни, що відбуваються для значень коефіцієнтів вейвлет-перетворення на деякій неперервній множині частот виглядають як ланцюжок "пагорбів", що мають вершини, розташовані в точках (по осі часу), у яких ці зміни досягають найбільших значень.

Іншим важливим показником є виражена тенденція динаміки часового ряду (тренд) поза залежністю від періодичних коливань.

Наявність тренда може бути неочевидною при простому розгляді часового ряду, наприклад, якщо тренд поєднується з періодичними коливаннями. Тренд відбивається на скейлограмі як плавна зміна яскравості уздовж осі часу одночасно на всіх масштабах. Якщо тренд наростаючий, то яскравість буде збільшуватися, якщо убутний – зменшуватися. Ще одним важливим фактором, якому необхідно враховувати при аналізі часових рядів, є локальні особливості, тобто можливі різкі, стрибкоподібні зміни характеристик вихідного ряду. Локальні особливості представлені на скейлограмі вейвлет-перетворення як лінії різкого перепаду яскравості, що виходять із точки, що відповідає часу виникнення стрибка. Аналіз локальних особливостей дозволяє відновити інформацію щодо динаміки вихідного процесу та у деяких випадках прогнозувати подібні ситуації.

Кожний з основних факторів динаміки часового ряду має свій, характерний відбиток на скейлограмі, при цьому вся аналітична інформація представляється в наочному та зручному для вивчення вигляді. Завдяки наочності подання результатів у вигляді скейлограми, іноді досить одного погляду, щоб побачити на ній найбільш суттєві фактори. На Рис. 8.2 наведена скейлограма – результат неперервного вейвлет-аналізу (вейвлет Гауса) часового ряду, що відповідає процесу, який розглядався вище.

Наведений приклад показує, що вейвлет-аналіз дозволяє виявляти не тільки очевидні аномалії в досліджуваному ряді, але й критичні значення, які приховані за відносно невеликими абсолютними значеннями елементів ряду. Наприклад, на скелетоні на більшості частот відмічено не тільки 250-й день, але й неявні екстремуми (25-й та 75-й дні).

На рис. 8.3 наведено динаміку зміни готівкового курсу продажу долара США в банках України протягом 2008 року.

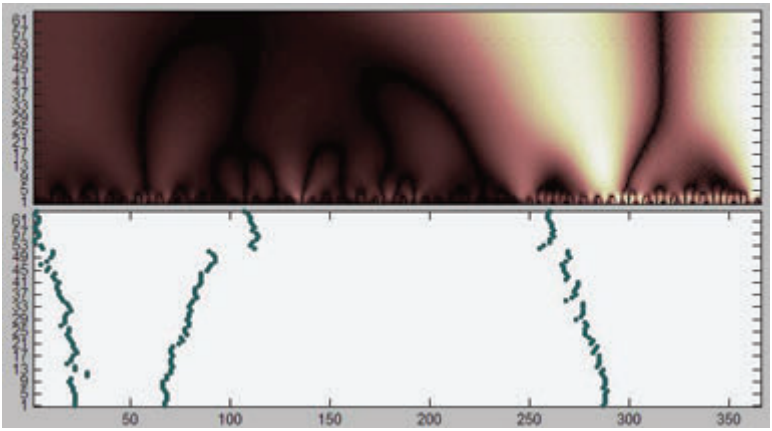


Рис. 8.2 – Результат вейвлет-аналізу (неперервне вейвлет-перетворення): зверху – вейвлет-скейлограма; знизу – лінії локальних максимумів (скелетон)

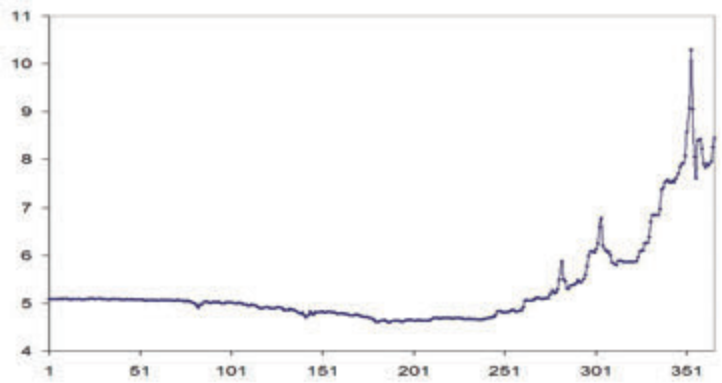


Рис. 8.3 – Динаміка зміни курсу готівкового долара у гривнях впродовж 2008 р.

На рис. 8.4 наведено застосування вейвлет-аналізу до цього ряду. Зсув останньої лінії локальних трендів на цій скейлограмі у порівнянні з попередній свідчить про те, що готівковий курс є похідною від загальнокризових явищ.

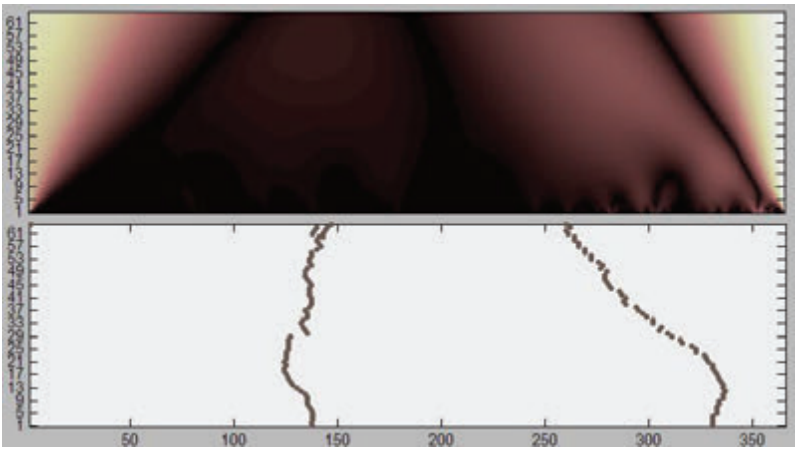


Рис. 8.4 – Результат вейвлет-аналізу ряду курсу готівкового долара: зверху – вейвлет-скейлограма; знизу – лінії локальних максимумів (скелетон)

8.4. Побудова вейвлет-скейлограм

Розглянемо приклади побудови вейвлет-скейлограм у середовищі Matlab.

Приклад 1. Мексиканська шляпа

```
clear
N=100
t=1:1:100
x=rand(1,N)
for j=2:N
        x(j)=2*x(j-1)*sin(x(j))+x(j)
end
figure(1); plot(t,x)
subplot(211); plot(t,x)
subplot(212); c=cwt(x,1:1:20,'mexh','absvil',[0,1])
```

Приклад 2. Вейвлет Хаара

```
clear
N=100
t=1:1:100
x=rand(1,N)
for j=2:N
        x(j)=2*x(j-1)*sin(x(j))+x(j)
end
% Багатооконність
figure(1); plot(t,x)
subplot(211); plot(t,x)
subplot(212); c=cwt(x,1:1:20,'haar','absvil',[0,1])
% Set number of iterations and wavelet name.
iter = 10;
wav = 'haar';
% Апроксімація вейвлет-функції
% за допомогою каскадного алгоритму
for i = 1:iter
   [phi,psi,xval] = wavefun(wav,10);
   plot(xval,psi);
   hold on
end
title(['Approximations of the wavelet ',wav, ...
      ' for 1 to ',num2str(iter),' iterations']);
hold off
```

Питання для самоперевірки

1. Визначення і властивості узагальненого перетворення Фур‘є.
2. Визначення і властивості вейвлет-перетворення

Література до розділу

1. *Астафьева Н.М.* Вейвлет-анализ: основы теории и примеры применения // Успехи физических наук, 1996. – 166. – No 11. – P. 1145-1170. 2. *Buckheit J., Donoho D.* Wavelab and reproducible research // Stanford University Technical Report 474: Wavelets and Statistics Lecture Notes, 1995. – 27 p. 3. *Додонов А.Г., Ландэ Д.В., Цыганок В.В., Андрейчук О.В., Каденко С.В., Грайворонская А.Н.* Распознавание информационных операций – К.: Инжиниринг, 2017. – 282 с.

9. ОСНОВИ КОНЦЕПЦІЇ СКЛАДНИХ МЕРЕЖ

Останнім часом все більшої популярності отримує область дискретної математики, що має назву теорії складних мереж (Complex Networks), що вивчає характеристики мереж, враховуючи не тільки їх топологію, а й статистичні феномени, розподіл ваг окремих вузлів і ребер, ефекти протікання і провідності в таких мережах, як мережі струму, рідини, інформації тощо. Виявилося, що властивості багатьох реальних мереж істотно відрізняються від властивостей класичних випадкових графів.

Незважаючи на те, що в розгляд теорії складних мереж потрапляють різні мережі, найбільший внесок в розвиток цієї теорії внесли дослідження соціальних мереж. Термін «соціальна мережа» означає зосередження соціальних об'єктів, які можна розглядати як мережу (або граф), вузли якої – представники соціуму, а зв'язки – соціальні відносини. Цей термін був введений у 1954 році соціологом з «Манчестерской школи» Дж. Барнсом (J. Barnes) в роботі «Класи і збори в норвезькому острівному приході». У другій половині XX століття поняття «соціальна мережа» стало популярним у західних дослідників, при цьому як вузли соціальних мереж стали розглядати не тільки люди, а й інші об'єкти, яким властивий соціальні зв'язки. У теорії соціальних мереж отримало розвиток такий напрямок, як аналіз соціальних мереж (Social Network Analysis, SNA). Сьогодні термін «соціальна мережа» позначає поняття, яке виявилося ширше свого соціального аспекту, воно включає, наприклад, багато інформаційних мереж, у тому числі і веб-простір.

В рамках теорії складних мереж розглядають не тільки статистичні, але динамічні мережі, для розуміння структури яких необхідно враховувати принципи їх еволюції.

9.1. Параметри складних мереж

У теорії складних мереж виділяють три основних напрямки: дослідження статистичних властивостей, які характеризують поведінку мереж; створення моделі мереж; передбачення поведінки мереж при зміні структурних властивостей.

У прикладних дослідженнях зазвичай застосовують такі типові для мережевого аналізу характеристики, як розмір мережі, мережева щільність, ступінь центральності тощо.

При аналізі складних мереж як і в теорії графів досліджуються параметри окремих вузлів; параметри мережі в цілому; мережеві підструктури.

Параметри вузлів мережі

Для окремих вузлів виділяють наступні параметри:

- вхідна ступінь вузла – кількість ребер графа, які входять в вузол;
- вихідна ступінь вузла – кількість ребер графа, які виходять з вузла;
- відстань від даного вузла до кожного з інших;
- середня відстань від даного вузла до інших;
- ексцентричність (eccentricity) – найбільша з геодезичних відстаней (мінімальних відстань між вузлами) від даного вузла до інших;
- посередництво (betwetnness), що показує, скільки найкоротших шляхів проходить через даний вузол;
- центральність – загальна кількість зв'язків даного вузла по відношенню до інших.

Загальні параметри мережі

Для розрахунку індексів мережі в цілому використовують такі параметри, як: кількість вузлів, кількість ребер, геодезична відстань між

вузлами, середня відстань від одного вузла до інших, щільність (відношення кількості ребер в мережі до максимально можливої кількості ребер при заданій кількості вузлів), кількість симетричних, транзитивних і циклічних тріад, діаметр мережі (найбільша геодезична відстань в мережі) тощо.

Існує декілька актуальних завдань дослідження складних мереж, серед яких можна виділити наступні:

- визначення клік в мережі. Кліки – це підгрупи або кластери, в яких вузли пов'язані між собою сильніше, ніж з членами інших клік;
- відокремлення компонент (частин мережі), які пов'язані всередині і не пов'язані між собою;
- знаходження блоків і перемичок. Вузол називається перемичкою, якщо при його вилученні мережа розпадається на незв'язані частини;
- виділення угруповань – груп еквівалентних вузлів (які мають максимально схожі профілі зв'язків).

Розподіл ступенів вузлів

Важливою характеристикою мережі є функція розподілу ступенів вузлів $P(k)$, яка визначається як ймовірність того, що довільний вузол мережі i має ступінь $k_i = k$. Мережі, які характеризуються різними $P(k)$, демонструють досить різну поведінку. $P(k)$ в деяких випадках може бути розподілами Пуассона ($P(k) = e^{-m} m^k / k!$, де m – математичне очікування), експонентним ($P(k) = e^{-k/m}$) або ступеневим ($P(k) \sim 1/k^{\gamma}$, $k \neq 0$, $\gamma > 0$).

Мережі із ступеневим розподілом ступенів вузлів називаються безмасштабними (scale-free). Саме безмасштабний розподіл часто спостерігається в реально існуючих складних мережах. При ступеневому

розподілі можливе існування вузлів з дуже високим ступенем, що практично не спостерігається в мережах з пуасонівським розподілом.

Шлях між вузлами

Шлях між вузлами – це послідовність вершин і ребер, що з'єднують дві вершини. Відстань між вузлами – кількість кроків, які потрібно зробити, щоб добратися від однієї вершини до іншої. Вершини в графі можуть бути зв'язані прямо або опосередковано. Для всієї мережі існує поняття середньої відстані між вершинами l:

$$l = \frac{1}{\frac{1}{2}n(n+1)} \sum_{i \geq j} d_{ij},$$

де d_{ij} – найкоротша відстань між вузлами i та j.

Коефіцієнт кластеризації

Д. Уаттс і С. Строгатц визначили такий параметр складних мереж, як коефіцієнт кластеризації, що відповідає рівню зв'язності вузлів у мережі.

Коефіцієнт кластеризації для окремого вузла мережі визначається в такий спосіб. Нехай з вузла виходить k ребер, які з'єднують його з k іншими вузлами, найближчими сусідами. Якщо припустити, що всі найближчі сусіди з'єднані безпосередньо один з одним, то кількість ребер між ними складало би $\frac{1}{2}k(k-1)$. Тобто це число, що відповідає максимально можливій кількості ребер, якими могли б з'єднуватися найближчі сусіди обраного вузла. Відношення реальної кількості ребер, що з'єднують найближчих сусідів вузла i до максимально можливого й називається коефіцієнтом кластеризації вузла $C(i)$. Природно, ця величина не перевищує одиниці.

Коефіцієнт кластеризації може визначатися як для кожного вузла, так і для всієї мережі. Відповідно, рівень кластеризації мережі визначається як нормована по кількості вузлів сума відповідних коефіцієнтів окремих

вузлів. Розглянутий нижче феномен «малих світів» безпосередньо пов'язаний з рівнем кластеризації.

Еластичність мережі

Властивість еластичності мереж відноситься до розподілу відстаней між вузлами при вилученні окремих вузлів. Більшість мереж базується на їх зв'язності, тобто існуванні шляхів між парами вузлів. Якщо вузол буде вилучений з мережі, типова довжина цих шляхів збільшується, і в остаточному підсумку пари вузлів стануть роз'єднаними.

Р. Альберт при дослідженні атак на інтернет-сервери досліджувала ефект вилучення вузла мережі з 326000 сторінковою підмножиною веб-простору. Середня відстань між двома вузлами, як функція числа вилучених вузлів, майже не змінилося при випадковому видаленні вузлів (висока еластичність). Разом з тим цілеспрямоване видалення вузлів з найбільшою кількістю зв'язків приводить до руйнування мережі. Таким чином, Інтернет є високоеластичною мережею по відношенню до випадкової відмови вузла в мережі, але високочутливою до навмисної атаки на вузли з високими ступенями зв'язків з іншими вузлами.

Структура співтовариства

Про "структуру співтовариства" можна говорити тоді, коли існують групи вузлів, що мають високу щільність ребер між собою, при тому, що щільність ребер між окремими групами – низька. Традиційний метод для виявлення структури співтовариств – кластерний аналіз. Існують десятки прийнятних для цього методів, що базуються на різних мірах відстаней між вузлами, зважених шляхових індексах між вузлами тощо. Для великих соціальних мереж наявність структури співтовариств виявилося невід'ємною властивістю.

9.2. Модель слабких зв'язків

Деякі властивості соціальних мереж не укладаються в рамки властивостей мереж із традиційним ієрархічним зв'язком. До таких властивостей відносяться й так званні “слабкі” зв'язки. Аналогом слабких зв'язків є, наприклад, відносини з далекими знайомими та колегами. У деяких випадках ці зв'язки виявляються більш ефективними, чим зв’язки «сильні». Так в області мобільного зв'язку групою вчених з Великобританії, США та Угорщини у 2005 році, був отриманий концептуальний висновок, що «слабкі» соціальні зв'язки між індивідуумами виявляються найважливішими для існування соціальної мережі.

Для дослідження були проаналізовані дзвінки 4,6 млн. абонентів мобільного зв'язку, що становить близько 20% населення однієї європейській країні. Зважаючи на все, це перший випадок у світовій практиці, коли вченим вдалося одержати та проаналізувати таку велику вибірку даних щодо міжособистісної комунікації. У соціальній мережі було виявлено 7 млн. соціальних зв'язків, якщо зворотні дзвінки були зроблені протягом 18 тижнів. Частота та тривалість розмов використовувалися для того, щоб визначити силу кожного соціального зв'язку. Виявлено, що саме слабкі соціальні зв'язки (один-два зворотних дзвінка протягом 18 тижнів) зв'язують воєдино більшу соціальну мережу. Якщо ці зв'язки забрати, то мережа розпадеться на окремі фрагменти. Якщо ж забрати сильні зв'язки, то нічого страшного з мережею не відбудеться – вона залишиться єдиною (Рис. 9.1).

На підставі проведених досліджень вчені зробили висновок, що саме слабкі зв'язки є тим феноменом, що зв'язує велике суспільство в єдине ціле. Треба думати, що даний висновок справедливий і для веб, хоча академічних досліджень у цій області дотепер не проводилося.

1 2 3

Рис. 9.1 – Структура мережі*:*

1) повна карта мережі соціальних комунікацій; 2) соціальна мережа, з якої вилучені слабкі зв'язки; 3) карта, з якого вилучені сильні зв'язки: структура зберігає наскрізну провідність

9.3. Концепція малих світів

Незважаючи на величезні розміри деяких соціальних мереж, у більшості з них (у веб, зокрема) існує порівняно короткий шлях між двома будь-якими вузлами – геодезична відстань. Ще в 1967 р. психолог С. Мілгран прийшов до висновку, що існує ланцюжок знайомств, у середньому довжиною, рівною шість, практично між двома будь-якими громадянами США. Угорськими математиками П. Ердошем та А. Реньі показано, що середня відстань між двома вершинами у випадковому графі росте як логарифм від числа вершин.

Ефект малих світів наочно демонструється процедурою, представленою Д. Уаттсом і С. Строгатцом. На Рис. 9.2 представлено 3 стану мережі: регулярна мережа – кожен вузол якої з'єднаний із чотирма сусідніми, та ж мережа, у якої деякі «ближні» зв'язки випадковим чином замінені «далекими» (саме в цьому випадку виникає феномен «малих світів») і випадкова мережа, коли кількість подібних замін перевищив деякий поріг.

Як виявилися саме ті мережі, вузли яких мають одночасно деяку кількість локальних і випадкових «далеких» зв'язків, демонструють й ефект малого миру, і високий рівень кластеризації.

На рис. 9.3 наведено графіки зміни середньої довжини шляху та коефіцієнта кластеризації від ймовірності встановлення «далеких зв'язків» (у напівлогарифмічній шкалі) для штучної мережі Д. Уаттса і С. Строгатца.

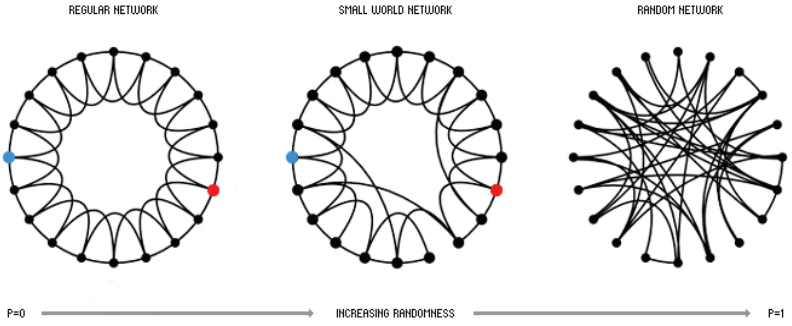


Рис. 9.2 – Модель Д. Уаттса і С. Строгатца

Саме до таких мереж відноситься веб-простір, для якого підтверджений феномен малих світів. Проведений аналіз топології Мережі показав, що великі вузли мають більше зв'язків між собою, ніж із малими вузлами, тоді як малі вузли мають більше зв'язків з великими вузлами, ніж між собою. Вчені назвали цей феномен "клубом багатих" (rich-club phenomenon). Дослідження показало, що 27% всіх з'єднань мають місце між усього 5% найбільших вузлів, 60% доводиться на з'єднання інших 95% вузлів з 5% найбільших і тільки 13% – це з'єднання між вузлами, які не входять до лідируючих.

Ці дослідження дають підстави думати, що залежність веб-простір від великих вузлів значно більше, ніж передбачалося раніше, тобто він ще більш вразливий відносно зловмисних атак.

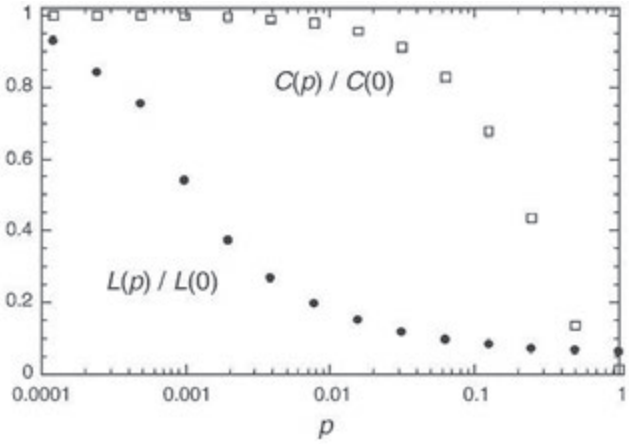


Рис. 9.3 – Динаміка зміни довжини шляху та коефіцієнта кластеризації

З концепцією «малих світів» зв'язаний практичний підхід, іменований «мережною мобілізацією», що реалізується над структурою «малих світів». Зокрема, швидкість поширення інформації завдяки цьому ефекту зростає на порядки, у порівнянні з теоретично можливою, адже більшість пар вузлів мережі з'єднані короткими шляхами.

9.4. Перколяція

При вивченні «малих світів» виявився цікавий феномен, логічно пов'язаний з поняттям перколяції (протікання), популярному в сучасній фізиці. Виявляється, що багато питань, що виникають при аналізі мережної безпеки безпосередньо відносяться до теорії перколяції.

Перед теорією перколяції стоїть багато питань, які виходять за рамки дискретної математики та теорії ймовірностей. Найпростіше, очищене від всіх фізичних і математичних нашарувань, формулювання завдання теорії перколяції має такий вигляд: «Дана решітка із зв'язками, випадкова частина яких проводить сигнал (повітря, струм, інформацію ...), а інша частина – не проводить. Основне питання – чому дорівнює мінімальна концентрація провідних зв'язків, при якій ще існує шлях через всю решітку?».

На цей час відомо багато важливих узагальнень перколяційного завдання, наприклад розглядаються випадки, коли «непровідні» зв'язки проводять, але багато гірше провідних; можна говорити про різні значення провідності для різних зв'язків; можна розглядати односпрямовані «діодні зв'язки» тощо.

До завдань, розв'язуваних у рамках теорії перколяції та аналізу мереж відносяться такі, як визначення граничного рівня провідності (пропускної здатності), зміни довжини шляху, та його траєкторії (звивистості, паралельності) при наближенні до граничного рівня провідності, кількості вузлів, яких необхідно вивести з ладу, щоб порушити зв'язність мережі.

9.5. Топологія веб-простору

Останнім часом одержала велику популярність теорія фракталів і детермінованого хаосу, що знаходить свої застосування в різних областях, у тому числі й при аналізі веб-простору.

У листопаді 1999 року Андрій Брёдер (Andrei Broder) і його співавтори з компаній AltaVista, IBM і Compaq математично описали "карту" ресурсів і гіперзв'язків веб-простору. Простеживши за допомогою пошукового механізму AltaVista понад 200 млн. Web-сторінок і декілька мільярдів посилань, розміщених на цих сторінках, вчені прийшли до висновку про структуру веб-простору як орієнтованого графа, в якому вершини відповідають веб-сторінкам, а ребра – гіперпосиланнями, що з'єднують ці сторінки. В рамках цієї моделі завдання аналізу структури зв'язків між окремими веб-сторінками було виявлено:

- центральне ядро (28% Web-сторінок) – компоненти сильної зв'язності (SCC).
- 22% Web-сторінок – це "відправні веб-сторінки" (IN). Вони містять гіперпосилання, які в кінцевому рахунку ведуть до ядра, але з ядра до них потрапити не можна.

- стільки ж – 22% "кінцевих веб-сторінок" (OUT), до яких можна прийти по посиланнях з ядра, але не можна повернутися назад.
- 22% Web-сторінок – відростки – повністю ізольовані від центрального ядра: це або "миси", пов'язані гіперпосиланнями зі сторінками будь-який інший категорії, або "перешийки", що з'єднують дві Web-сторінки, що не входять в ядро.

У моделі враховані і "острови", які взагалі не перетинаються з іншими ресурсами у веб-просторі. Єдиний спосіб виявити ресурси цієї групи – знати адресу.

Топологія та характеристики цієї моделі, що отримала назву Bow Tie (краватка-метелик – див. Рис. 9.4), виявилися приблизно однаковими для різних підмножин веб-простору, побічно підтверджуючи тим самим його фрактальну природу у тому плані, що властивості структури всього веб-простору виявилися також вірні і для його окремих підмножин.

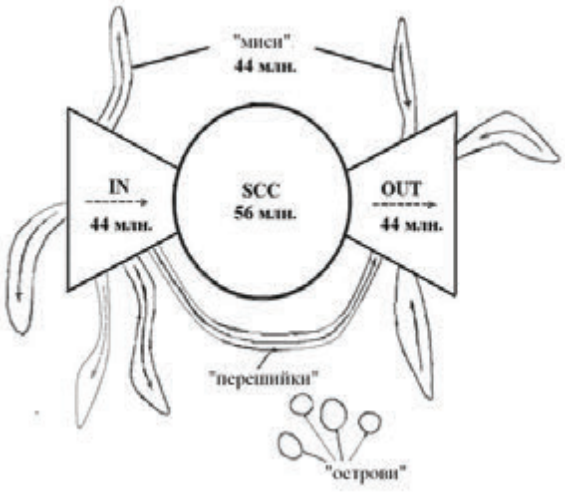


Рис. 9.4 – Модель веб-простору Bow Tie

Таким чином, алгоритми, що використають інформацію щодо структури веб-простору, повинні працювати також на окремих його підмножинах. Ця властивість структури веб-простору сьогодні вже досить

широко використовується при рішенні багатьох завдань, наприклад, для оптимізації ефективності механізмів сканування, при побудові нових веб-сервісів, для рішення завдань аналізу та прогнозу.

На цей час наука все ще далека від повного розуміння структури складних мереж і процесів, що відбуваються у них.

Мабуть, краще відбиття існуючого положення з вивченням топології веб-простору, це модель «зім'ятого вебу», запропонована датчанином Л. Бйорнеборном. «Зім'ятий веб» у цій моделі асоціюється із зім'ятим папером. При цьому шлях між обраними точками на зім'ятому папері найчастіше коротше, тому що протилежні частини аркушів паперу з'єднані разом. Відповідно до цієї моделі кожне нове гіперпосилання змінює всі існуючі зв'язки, створюючи нові деформації «зім'ятої» мережі. Тобто кожен новий гіперзв'язок – «гачок», що розтягує або деформує форму існуючої мережі веб-простору.

9.6. Візуалізація складних мереж

Одним з напрямків аналізу складних мереж є візуалізація. Візуалізація має важливе значення, оскільки найчастіше дозволяє робити важливі висновки щодо характеру взаємодії вузлів, не прибігаючи до точних методів аналізу. При відображенні моделі складної мережі доцільним може бути:

- розміщення вузлів мережі у двох вимірах;
- просторове впорядкування об'єктів в одному вимірі відповідно до деякої кількісної властивості;
- використання загальних для всіх мережних діаграм методів для відображення кількісних та якісних властивостей об'єктів і відносин.

Як приклади візуалізації мереж можна розглянути деякі розробки компанії TouchGraph. Так, наприклад, TouchGraph Amazon відображає мережу, породжену книгами та зв'язками між ними (за тематиками,

авторами, видавництвами). Одним із самих динамічних новинних ресурсів Інтернет сьогодні можна вважати також соціальні мережі. Компанія TouchGraph, зокрема, реалізувала інтерфейс для відображення мережі Facebook – TouchGraph Facebook Browser (http://www.touchgraph.com/facebook).

У випадку візуалізації веб-простору засобами TouchGraph SEO Browser (http://www.touchgraph.com/seo) ребрами виступають не гіперпосилання, а відносини подібності. TouchGraph SEO Browser представляє собою Java-аплет, що дозволяє візуалізувати зв'язки між веб-сайтами, що розраховуються в пошуковій системі Google. У цьому інтерфейсі можна побачити всі сайти, зв'язані з вихідним заданим, при цьому користувач може задавати глибину зв'язків і відображати взаємозв'язки різних сайтів. TouchGraph SEO Browser досить корисний інструмент також при пошуку сайтів, пов'язаних з вихідним загальною тематикою.

У якості ще одного інструмента для аналізу та візуалізації соціальних мереж можна привести програму NetVis (http://www.netvis.org), що використовує online-дані та імпортовані csv файли.

Ще один широко розповсюджений програмний інструмент з відкритим кодом візуалізації і дослідження складних мереж – це Gephi (http://gephi.org), написаний мовою Java на платформі NetBeans. Gephi використовується в ряді дослідницьких проектів в академічній, журналістській та інших сферах, наприклад, у візуалізації глобального підключення вмісту New York Times та вивчення мережевого трафіку Twitter під час соціальних хвилювань, широко використовується в гуманітарних науках.

Також широко відомі програми візуалізації та аналізу соціальних/організаційних мереж InFlow (поточна версія доступна за адресою http://www.orgnet.com/inflow3.html) і система аналізу соціальних мереж UCINET (https://sites.google.com/site/ucinetsoftware/home) з

інтегрованою до неї вільно розповсюджуваною програмою візуалізації NetDraw.

Питання для самоперевірки

1. Основна концепція складних мереж?
2. Назвіть основні показники складних мереж.
3. Як будується структура веб-простору?
4. У чому полягає сутність концепції «малих світів».

Література до розділу

1. *Newman M.E.J.* The structure and function of complex networks // SIAM Review. – 2003. – V. 45. – pp. 167-256. 2. *Снарский А.А., Ландэ Д.В.* Моделирование сложных сетей: учебное пособие. – К.: Инжиниринг, 2015. – 212 с. 3. Ландэ Д.В., Снарский А.А., Безсуднов И.В. Интернетика: Навигация в сложных сетях: модели и алгоритмы. – М.: Либроком (Editorial URSS), 2009. – 264 с. 4. Додонов А.Г., Ландэ Д.В., Путятин В.Г. Компьютерные сети и аналитические исследования. – К.: ИПРИ НАН Украины, 2014. – 486 с.

10. МОДЕЛІ ІНФОРМАЦІЙНОГО ПОШУКУ

10.1. Булева модель пошуку

Булева модель базується на теорії множин і математичній логіці. Популярність цієї моделі пов'язана з простотою її реалізації, яка дозволяє індексувати та виконувати пошук у великих документальних масивах. У рамках булевої моделі документи та запити представляються у вигляді множин термів. Кожен терм представляється як булева змінна: 0 (терм із запиту не присутній у документі) або 1 (терм із запиту присутній у документі). При цьому вага терму в документі приймає лише два значення: $w_i^{(j)} \in \{0,1\}$.

Класична булева модель

У булевої моделі запит користувача являє собою логічний вираз, у якому терми зв'язуються логічними операторами кон'юнкції (AND, $\wedge$), диз'юнкції (OR, $\vee$) і заперечення (NOT, $\neg$). Відомо, що будь-який логічний вираз можна представити диз'юнкцією деяких виразів, з'єднаних між собою операцією кон'юнкції (диз'юнктивною нормальною формою, ДНФ).

Покажемо, як на практиці запит, складений з логічних операторів і дужок, які визначають пріоритет операторів, приводиться до ДНФ. Розглянемо приклад (Рис. 10.1) – запит: $q = a \wedge (b \vee \neg c)$. Якщо множині документів, що містять терми із запиту a, b та c, покласти у відповідність діаграми Ейлера, то легко можна бачити, що вихідний запит еквівалентний запиту $(a \wedge b \wedge c) \vee (a \wedge b \wedge \neg c) \vee (a \wedge \neg b \wedge \neg c)$, тобто диз'юнктивна нормальна форма запиту прийме вигляд: $q_{dnf} = (1,1,1) \vee (1,1,0) \vee (1,0,0)$, де кожна із двійкових компонентів асоціюється з a, b або c (або їх запереченнями), а кома між двійковими компонентами – з операціями кон'юнкції.

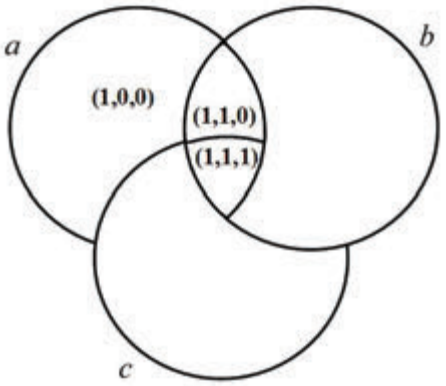


Рис. 10.1 – Три кон'юктивних компоненти запиту $q = a \wedge (b \vee \neg c)$

У булевої моделі запит – це булевий вираз. Завдяки тому, що він приводиться до диз'юнктивної нормальної форми, можна записати:

$$q \equiv q_{dnf} = \bigvee_{i=1,..,N} q_{cc}^{(i)},$$

де $q_{cc}^{(i)}$ – i-а кон'юнктивна компонента форми запиту q_{dnf}. Тоді міра близькості документа $d^{(j)}$ та запиту q – $sim(d^{(j)}, q)$ (від англ. – *similarity*, близькість) у булевої моделі визначається виразом:

$$sim(d^{(j)}, q) = \begin{cases} 1, & якщо \quad \exists q_{cc}^{(i)} \mid (q_{cc}^{(i)} \in q_{dnf}) \wedge (\forall k, \quad g_k(q_{cc}^{(i)}) = g_k(d^{(j)})), \\ 0, & інакше. \end{cases}$$

Тобто $sim(d^{(j)}, q)$ приймає значення 1, якщо існує така кон'юнктивна компонент $q_{cc}^{(i)}$, яка входить у диз'юнктивну нормальну форму q_{dnf}, що інверсна функція кожного терму k даної кон'юнктивної компоненти збігається із цією же інверсною функцією для документа $d^{(j)}$. У противному випадку $sim(d^{(j)}, q)$ дорівнює нулю.

Таким чином, якщо $sim(d^{(j)}, q) = 1$, то відповідно до булевої моделі документ $d^{(j)}$ вважається релевантним (відповідним) запиту q. У противному випадку документ не є релевантним. Вагових розходжень,

необхідних для ранжирування документів за рівнем відповідності запиту, в булевої моделі не передбачено, що є істотним недоліком. Нижче буде розглянуто розширену булеву модель, у якій цей недолік усувається.

Розширена булева модель

Недолік класичної булевої моделі пов'язаний з нівелюванням значимості окремих термів. Це приводить до неможливості ранжирування результатів пошуку за рівнем їхньої відповідності запитам.

Для того щоб усунути цей недолік і разом з тим використати обчислювальні переваги булевої моделі була запропонована розширена булева модель. Відповідно до цієї моделі, кожному терму приписується вага – значення з інтервалу [0, 1]. Нехай D – масив документів, у документ $\vec{d} \in D$ входять терми x та y. Тоді парам $\left[x,\vec{d}\right]$ і $\left[y,\vec{d}\right]$ ставляться у відповідність вагові значення $\hat{x}$ й $\hat{y}$. Ці значення можуть бути визначені, наприклад, за формулою:

$$\hat{x} = f_x \frac{idf_x}{\max\ idf},\quad \hat{y} = f_y \frac{idf_y}{\max\ idf},$$

де f_x – нормалізована частота терму x в документі $\vec{d}$, а idf_x – величина, зворотна нормованій кількості тих документів у всьому масиві (інверсна частота), які містять терм x.

Замість документа в рамках цієї моделі розглядається вектор з термів (у найпростішому випадку – двох) $\vec{d} = (\hat{x}, \hat{y})$, що також можна розглядати як точку у квадраті [0, 1] ×[0, 1]. Близькість між документами можна інтерпретувати як деяку нормовану відстань між відповідними точками, при цьому вводяться такі міри близькості між документом і запитом для двох типів запитів:

$$sim(q_{or}, d) = \sqrt{\frac{\hat{x}^2 + \hat{y}^2}{2}};$$

$$sim(q_{and}, d) = 1 - \sqrt{\frac{(1-\hat{x})^2 + (1-\hat{y})^2}{2}}.$$

Якщо ще більше розширити модель і припустити, що відповідні терми можуть мати вагу в запиті, відповідно, *a* та *b*, то визначається:

$$sim(q_{or}, d) = \sqrt{\frac{a^2\hat{x}^2 + b^2\hat{y}^2}{a^2 + b^2}};$$

$$sim(q_{and}, d) = 1 - \sqrt{\frac{a^2(1-\hat{x})^2 + b^2(1-\hat{y})^2}{a^2 + b^2}}.$$

У розглянутих вище випадках використалася модель ступеня 2, що легко узагальнюється до моделі ступеня p, де $1 \le p < \infty$. У даному випадку передбачається, що запити складаються з m термів, з'єднаних узагальненими операторами диз'юнкції ($q_{or} = k_1 \vee^p k_2 \vee^p ... \vee^p k_m$) і кон'юнкції ($q_{and} = k_1 \wedge^p k_2 \wedge^p ... \wedge^p k_m$). При цьому вводиться таке визначення близькості запиту і документа:

$$sim(q_{or}, d) = \left(\frac{\hat{x}_1^p + \hat{x}_2^p + ... + \hat{x}_m^p}{m} \right)^{\frac{1}{p}};$$

$$sim(q_{and}, d) = 1 - \left(\frac{(1-\hat{x}_1)^p + (1-\hat{x}_2)^p + ... + (1-\hat{x}_m)^p}{m} \right)^{\frac{1}{p}}.$$

Очевидно, що для $p = 1$ справедливо:

$$sim(q_{or}, d) = sim(q_{and}, d) = \frac{\hat{x}_1 + \hat{x}_2 + ... + \hat{x}_m}{m}.$$

При $p = 1$ буде виконуватися:

$sim(q_{or}, d) \to \max(\hat{x}_i);$

$sim(q_{and}, d) \to \min(\hat{x}_i).$

Якщо запит є більше складним, то близькість документа і запиту може бути обчислена за допомогою двох наведених вище правил (для диз'юнкції й

кон'юнкції). Наприклад, у випадку, якщо запит являє собою вираз $q = (k_1 \vee^p k_2) \wedge^p k_3$, то виконується:

$$sim(q,d) = 1 - \left(\frac{1}{2} \left(\left(1 - \left(\frac{1}{2} \left(\hat{x}_1^p + \hat{x}_2^p \right) \right)^{\frac{1}{p}} \right)^p + (1 - \hat{x}_3)^p \right) \right)^{\frac{1}{p}}.$$

10.2. Модель нечіткого пошуку

Теорія нечітких множин (Fuzzy set theory), запропонована Л. А. Заде, що розширює класичну теорію множин, ґрунтується на ідеї, що функція приналежності елемента множини може приймати довільні значення в інтервалі [0, 1], а не тільки 0 або 1.

За визначенням, нечіткою множиною A на універсальній множині U (будь-якої природи) є сукупність пар $(\mu_A(U), U)$, де $\mu_A(U)$ – ступінь приналежності елемента $u \in U$ нечіткій множині A. Ступінь приналежності – це число з діапазону $[0, 1]$. Чим вище ступінь приналежності, тим у більшій мірі елемент u відповідає властивостям нечіткої множини. Функцією приналежності називається функція, що дозволяє обчислити ступінь приналежності довільного елемента універсальної множини до нечіткої множини.

Нехай A і B – дві нечітких множини над універсальною множиною U, а $\overline{A}$ – доповнення A до U та $u \in U$. Функцію приналежності μ можна визначити багатьма способами, наприклад, таким чином:

$\mu_{\overline{A}}(u) = 1 - \mu_A(u)$;

$\mu_{A \cup B}(u) = \max(\mu_A(u), \mu_B(u))$;

$\mu_{A \cap B}(u) = \min(\mu_A(u), \mu_B(u))$.

Л. Заде ввів також поняття носія, висоти та точки переходу нечіткої множини. Носієм нечіткої множини A називається множина таких точок $u \in U$, для яких величина $\mu_A(u)$ позитивна. Висотою нечіткої множини A

називається величина $\sup_{u \in U} \mu_A(u)$. Точкою переходу нечіткої множини A називається такий елемент множини U, ступінь приналежності якого до множини A дорівнює 1/2.

Як приклад розглядається універсальна множина U, що являє собою інтервал $[0, 100]$ (який інтерпретується як вік людини), і змінна u, яка приймає значення із цього інтервалу. При цьому нечітка підмножина універсальної множини U, яка позначується терміном «старий», може бути задана функцією приналежності вигляду (рис. 10.2):

$$\mu_A(u) = \begin{cases} 0, & (0 \le u \le 50), \\ \left(1 + \left(\dfrac{u-50}{5}\right)^{-2}\right)^{-1}, & (50 \le u \le 100). \end{cases}$$

У цьому прикладі носієм нечіткої множини «старий» є інтервал $[50, 100]$, висота множини «старий» близька до 1, а точкою переходу є значення $u = 55$ (Рис. 10.2). Таким чином, якщо вік людини становить 55 років, то його відповідність поняттю «старий» становить 0.5, а якщо 70, то близько до 0.9. Чоловік молодше 50 років ($\mu_A(u) = 0$) взагалі не може назватися старим. Помітимо, що функція приналежності вибиралася із суб'єктивних уявлень щодо того, з якого віку починається старість.

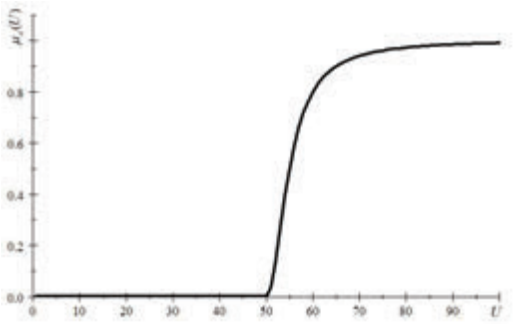


Рис. 10.2 – Функція приналежності нечіткої множини, обумовленої поняттям «старий»

У моделі нечіткого пошуку кожен терм запиту розглядається як нечітка множина, а кожен документ розглядається по ступені приналежності до цієї множини. При цьому розглядається матриця взаємозв'язків термів $\hat{c}$, елемент якої c_{il} визначає взаємозв'язок між термами з індексами i та l:

$$c_{il} = \frac{n_{il}}{n_i + n_l - n_{il}},$$

де n_i – кількість документів, що містять терм t_i, n_l – кількість документів, які містять терм t_l, а n_{il} – кількість документів, які містять обидва терми.

У рамках моделі нечітких множин, документ $d^{(j)}$ має ступінь приналежності μ_{ij}, яка розраховується за формулою:

$$\mu_{ij} = 1 - \prod_{x_l \in d_j} (1 - c_{il}).$$

Документ $d^{(j)}$ асоціюється з термом t_i, якщо його власні терми асоціюються з t_i. Якщо хоча б одних терм t_l зі $d^{(j)}$ строго асоціюється з термом з індексом t_i (тобто $c_{il} \sim 1$), то $\mu_{ij} \sim 1$ і терм t_i є «добрим» нечітким індексом документа $d^{(j)}$. Крім того, замість функції μ_{ij} при обчисленні функції приналежності у випадку реалізації оператора диз'юнкції між різними термами також застосовується підсумовування, що також є деяким огрубінням моделі, яка розглядається. Для обчислення кон'юнкції, відповідно, розглядається добуток елементів c_{il}.

Для пошуку в моделі нечітких множин використається запит, подібний звичайному булевському виразу. Так само, як і у випадку булевої логіки, запит може бути представлений у диз'юнктивній нормальній формі, а саме:

$$q_{dnf} = cc_1 \vee cc_2 \vee ... \vee cc_p,$$

де cc_i – i-а кон'юнктивна компонента, p – кількість таких компонент q_{dnf}.

Відповідно, функція приналежності μ_{qj} документа $d^{(j)}$ до нечіткої множини D_q, відповідна запиту q (надалі застосовується як параметр для ранжирування релевантних документів), обчислюється таким чином:

$$\mu_{qj} = 1 - \prod_{i=1}^{p} \left(1 - \mu_{cc_i j}\right).$$

10.3. Векторна-просторова модель пошуку

Багато відомих пошукових систем базуються на векторно-просторовій моделі даних (Vector Space Model), яку запропонував Г. Солтон у 1975 р. і вперше застосував в системі SMART. У рамках цієї алгебраїчної моделі пошуку документ описується вектором у евклідовому просторі, у якому кожному терму з документа, ставиться у відповідність його вагове значення, яке визначається на основі статистичної інформації щодо його появи як в окремому документі, так і у всьому документальному масиві. Опис запиту, що відповідає необхідній користувачеві тематиці, також являє собою вектор у тому ж евклідовому просторі термів. Для оцінки близькості запиту та документа використається скалярний добуток відповідних їм векторів.

Кожному терму t_i у документі $d^{(j)}$ відповідає деяка ненегативна вага $w_i^{(j)}$. Кожному запиту q, який поєднує терми, не з'єднані між собою ніякими логічними операторами, також відповідає вектор вагових значень w_i^q. Таким чином, кожен документ і запит можуть бути представлені у вигляді $n-$мірного вектора, де n – загальна кількість термів у словнику. Відповідно до розглянутої моделі, близькість документа $d^{(j)}$ до запиту q, які розглядаються як інформаційні вектори $\vec{d}_j = (w_1^{(j)}, w_2^{(j)}, ..., w_n^{(j)})$ та $\vec{q} = (w_1^q, w_2^q, ..., w_n^q)$, оцінюється як їхній скалярний добуток. При цьому вагу окремих термів можна обчислювати різними способами. Один з

можливих найпростіших підходів – використати як вагу терму $w_i^{(j)}$ в документі нормалізовану частоту $freq_i^{(j)}$ його появи в документі, тобто:

$$w_i^{(j)} = freq_i^{(j)} / \max_{1 \le k \le n} freq_k^{(j)}.$$

Обчислену таким чином вагу терму в документі прийнято позначати абревіатурою $tf_i^{(j)}$ або просто *TF* (англ. – *Term Frequency* – частота терму).

Однак цей підхід не враховує, наскільки часто розглянутий терм зустрічається у всьому масиві документів, так звану, дискримінаційну силу терму. Тому у випадку, коли доступна статистика використання термів у всьому документальному масиві, виявляється більш ефективною така формула обчислення ваги:

$$w_i^{(j)} = tf_i^{(j)} \cdot \log \frac{N}{n_i},$$

де n_i – кількість документів, у яких використається терм t_i, а N – загальна кількість документів у масиві. Наприклад, якщо деяке слово зустрічається в кожному документі масиву, то його використання в запиті, мабуть, недоречне. Відповідно, у цьому випадку $n_i = N$, отже, $w_i^{(j)} = tf_i^{(j)} \cdot \log \frac{N}{N} = 0.$

Слід зазначити, що наведена вище формула багаторазово уточнювалася з метою покращення відповідності документів, які видаються пошуковою системою, запитам користувачів. У 1988 році Дж. Солтоном був запропонований такий варіант для обчислення ваги терму:

$$w_i^q = \left(0.5 + \frac{freq_i^q}{\max_{1 \le l \le n} freq_l^q} \right) \cdot \log \frac{N}{n_i},$$

де $freq_i^q$ – частота терму t_i із запиту в тексті документа.

Звичайно вагові значення $w_i^{(j)}$ нормуються шляхом поділу на їхню загальну суму. Такий метод зважування термів має стандартне позначення

– $TF \cdot IDF$, де TF указує на частоту появи терму в документі, а IDF – на величину, зворотну кількості документів у масиві, які містять даний терм (*Inverse Document Frequency*).

Коли виникає завдання визначення близькості двох документів або документа і запиту, у цій моделі використається простий скалярний добуток $sim(d^{(1)}, d^{(2)})$ двох векторів вагових значень $(w_1^{(1)}, w_2^{(1)}, ..., w_n^{(1)})$ та $(w_1^{(2)}, w_2^{(2)}, ..., w_n^{(2)})$, що відповідає косинусу кута між цими векторами – образами документів $d^{(1)}$ та $d^{(2)}$. Очевидно, $sim(d^{(1)}, d^{(2)})$ належить до діапазону [0, 1]. Чим більше величина $sim(d^{(1)}, d^{(2)})$ – тим більш близькі документи $d^{(1)}$ та $d^{(2)}$. Для будь-якого документа d маємо $sim(d, d) = 1$. Аналогічно мірою близькості документа $d^{(j)}$ та запиту q є величина:

$$sim(d_j, q) = \frac{\vec{d}^{(j)} \cdot \vec{q}}{|\vec{d}^{(j)}| \cdot |\vec{q}|} = \frac{\sum_{i=1}^{n} w_i^{(j)} w_i^q}{\sqrt{\sum_{i=1}^{n} (w_i^{(j)})^2} \sqrt{\sum_{i=1}^{n} (w_i^q)^2}}.$$

Векторно-просторова модель представлення даних забезпечує такі можливості, як:

- обробку запитів без обмежень їхньої довжини;
- простоту реалізації режиму пошуку подібних документів (кожен документ може розглядатися як запит);
- збереження результатів пошуку з можливістю виконання уточнюючого пошуку.

Разом з тим у векторно-просторової моделі не передбачено використання логічних операцій у запитах, що істотно обмежує її застосування.

10.4. Ймовірнісна модель пошуку

Ймовірнісна модель пошуку ґрунтується на припущенні, що терми запиту по-різному розподілені серед релевантних і нерелевантних

документів. При цьому використаються формули розрахунку ймовірності на основі теореми Байєса.

Основне питання, що вирішується за допомогою цієї моделі: як велика ймовірність того, що документ релевантний запиту. Функціонування моделі базується як на експертних оцінках, які визнають документи з навчальної колекції релевантними/нерелевантними, так і на наступних оцінках ймовірності того, що документ є релевантним запиту виходячи зі складу його термів.

Якщо для запиту відомі ці оцінки ймовірностей для всіх документів, то документи можна сортувати за ними та виводити користувачам. Тобто ймовірнісна модель пошуку передбачає визначення ймовірностей відповідності документів запиту, подальше сортування та надання документів користувачеві.

В ймовірнісній моделі використовується спрощення, яке допускає незалежність входження в документ будь-якої пари термів (тому такий підхід називається «наївним» байєсівським). При цьому пошуку передбачається наявність навчальних наборів релевантних і нерелевантних документів, обраних користувачем або отриманих автоматично при деякому початковому припущенні. Ймовірність того, що документ, який надійшов, є релевантним, розраховується на підставі співвідношення появи термів у релевантному та нерелевантному масиві документів.

У випадку застосування експертних оцінок процес пошуку є ітераційним. Завдяки режиму зворотного зв'язку, на кожному кроці ітерації визначається множина документів, які позначаються користувачем як такі, що задовольняють його інформаційним потребам.

Розглянемо основу модель, а саме байєсівський підхід, більш детально. Нехай X, Y – дві незалежних події, X, $Y \subset G$, $G-$ базовий ймовірнісний простір.

Ймовірність X за умови Y визначається таким чином:

$$P(X \mid Y) = \frac{P(X \bigcap Y)}{P(Y)}.$$

Відомо, що із цього співвідношення випливає формула Байєса:

$$P(X \mid Y) = \frac{P(X \bigcap Y)}{P(Y)}; \quad P(Y \mid X) = \frac{P(Y \bigcap X)}{P(X)};$$

$$P(Y \bigcap X) = P(X \bigcap Y) \Rightarrow P(X \mid Y) = \frac{P(Y \mid X) \cdot P(X)}{P(Y)}.$$

Розглянемо умовні ймовірності двох подій, а саме того, що документ релевантний (R) запиту – $P(R \mid q,d)$, де $q-$ запит, $d-$документ, а також того, що документ нерелевантний ($\overline{R}$) запиту – $P(\overline{R} \mid q,d)$.

У рамках імовірнісної моделі вводиться поняття квоти релевалентності як міри близькості документа запиту – $O(R)$:

$$O(R) = \frac{P(R)}{P(\overline{R})} = \frac{P(R)}{1 - P(R)}.$$

Очевидно, що квота менша ніж 1 для ймовірності $P(R) < 0.5$ і більша за 1 для ймовірності $P(R) > 0.5$.

Визначимо квоту тієї події, що документ релевантний запиту:

$$O\left(R \mid q,d\right) = \frac{P(R \mid q,d)}{P(\overline{R} \mid q,d)}.$$

Для чисельника цієї формули справедливо:

$$P(R \mid d,q) = \frac{P(R \bigcap q \bigcap d)}{P(q \bigcap d)}.$$

Величина $P(R \bigcap q \bigcap d)$ у наведеному виразі інтерпретується як ймовірність події, що полягає у тому, що документ d релевантний запиту q, а величина $P(q \bigcap d)$ – ймовірність того, що за запитом q буде виданий документ d. Використовуючи формулу Байєса для чисельника і знаменника одержуємо:

$$P(R \mid d,q) = \frac{P(d \mid R \bigcap q) \cdot P(R \bigcap q)}{P(d \mid q) \cdot P(q)} = \frac{P(d \mid R \bigcap q) \cdot P(R \mid q)}{P(d \mid q)}.$$

Підставляючи вирази $P(R \mid d,q)$ та $P(\bar{R} \mid d,q)$ у чисельник і знаменник формули для квоти релевалентності, одержуємо:

$$O(R \mid d,q) = \frac{\dfrac{p(d \mid R \bigcap q) \cdot p(R \mid q)}{p(d \mid q)}}{\dfrac{p(d \mid \bar{R} \bigcap q) \cdot p(\bar{R} \mid q)}{p(d \mid q)}} = \frac{p(R \mid q)}{p(\bar{R} \mid q)} \times \frac{p(d \mid R \bigcap q)}{p(d \mid \bar{R} \bigcap q)}.$$

Перейдемо до розгляду документа як вектора термів. Нехай $T = \{t_1, \ ..., \ t_n\}$ – множина термів, які втримуються в масиві документів D. Документ розглядається як вектор з бінарних значень ваги термів $\vec{d} = (w_1, ..., w_n)$, що входять до нього, де:

$$w_i = \begin{cases} 1, & t_i \in d; \\ 0, & t_i \notin d. \end{cases}$$

Тоді припускаючи незалежність термів у рамках розглянутої «наївної» байєсовської моделі, одержуємо:

$$p(d \mid R,q) = p(\vec{d} \mid R,q) = \prod_{i=1}^{n} p(t_i \mid R,q).$$

У результаті квота релевалентності приймає вигляд:

$$O(R \mid q,d) = \frac{p(R \mid q)}{p(\bar{R} \mid q)} \cdot \frac{p(d \mid R \bigcap q)}{p(d \mid \bar{R} \bigcap q)} = O(R \mid q) \cdot \prod_{i=1}^{n} \frac{p(t_i \mid R \bigcap q)}{p(t_i \mid \bar{R} \bigcap q)}.$$

Тут $O(R \mid q)$ – квота релевалентності для запиту без урахування документів. Модель передбачає ще одне спрощення, а саме те, що для термів, які не входять до запиту (для $t_i \in T \setminus q$), передбачається однакова ймовірність їхньої появи в релевантних і нерелевантних документах, тобто:

$$t_i \in T \setminus q : p(t_i \mid R,q) = p(t_i \mid \bar{R},q).$$

Розкладемо добуток у формулі квоти релевалентності таким чином:

$$O(R \mid q,d) = O(R \mid q) \cdot \prod_{t_i \in q \bigcap d} \frac{p(t_i \mid R \bigcap q)}{p(t_i \mid \bar{R} \bigcap q)} \cdot \prod_{t_i \in q \setminus d} \frac{p(t_i \mid R \bigcap q)}{p(t_i \mid \bar{R} \bigcap q)} \cdot \prod_{t_i \notin q} \frac{p(t_i \mid R \bigcap q)}{p(t_i \mid \bar{R} \bigcap q)}.$$

У наведених позначеннях під знаком добутку вираз $q \cap d$ означає множину загальних термів у запиті та документі, $q \setminus d$ - множина слів, що входять у запит, але відсутні в документі, а q – множина слів, що входять у запит.

Останній співмножник дорівнює одиниці через вищенаведене припущення. Введемо позначення для ймовірностей того, що слово присутнє в документі, за умови того, що документ релевантний або нерелевантний запиту:

$r_i = p(w_i = 1 \mid R, q);$

$n_i = p(w_i = 1 \mid \bar{R}, q).$

У цих позначеннях виконується:

$$O(R \mid q,d) = O(R \mid q) \cdot \prod_{t_i \in q \cap d} \frac{r_i}{n_i} \cdot \prod_{t_i \in q \setminus d} \frac{1-r_i}{1-n_i}.$$

Зважаючи на те, що:

$$\prod_{t_i \in q \cap d} \frac{(1-r_i)(1-n_i)}{(1-n_i)(1-r_i)} = 1,$$

одержуємо:

$$O(R \mid q,d) = O(R \mid q) \cdot \prod_{t_i \in q \cap d} \frac{r_i(1-n_i)}{n_i(1-r_i)} \cdot \prod_{t_i \in q} \frac{1-r_i}{1-n_i}.$$

Для дослідження релевантної послідовності елементів досить ураховувати тільки другий співмножник, тому що тільки в ньому присутні ознаки, пов'язані з документом. Крім того, значення цього співмножника можна про логарифмувати (логарифм – монотонна функція, що не міняє рангів документів). Тобто можна аналізувати суму:

$$\sum_{t_i \in q \cap d} \log \frac{r_i(1-n_i)}{n_i(1-r_i)} = \sum_{t_i \in q \cap d} \left[\log \frac{r_i}{n_i} + \log \frac{1-n_i}{1-r_i} \right].$$

Розглянемо наближені значення, отримані на основі аналізу деякої попередньо отриманої навчальної вибірки:

$$\tilde{r}_i = \frac{rel_i}{rel};\ \ \tilde{n}_i = \frac{nrel_i}{nrel},$$

де rel_i – кількість релевантних документів, які містять терм з індексом i; $nrel_i$ – відповідно, кількість нерелевалентних документів.

Тобто можна аналізувати суму, яку називають пошуковим статусом:

$$SV = \sum_{t_i \in q \cap d} \log \frac{\tilde{r}_i (1 - \tilde{n}_i)}{\tilde{n}_i (1 - \tilde{r}_i)} = \sum_{t_i \in q \cap d} \log \frac{\frac{rel_i}{rel} \cdot \left(1 - \frac{nrel_i}{nrel}\right)}{\frac{nrel_i}{nrel} \cdot \left(1 - \frac{rel_i}{rel}\right)}.$$

Після елементарних перетворень, одержуємо:

$$SV = \sum_{t_i \in q \cap d} SV_i = \sum_{t_i \in q \cap d} \log \frac{rel_i (nrel - nrel_i)}{nrel_i (rel - rel_i)}.$$

Як приклад розглянемо масив документів, що складається з двох частин: навчальної вибірки – документів $d^{(1)}$,..., $d^{(6)}$ (Табл. 10.1) і нових документів – $d^{(7)}$,..., $d^{(9)}$ (Табл. 10.2), для яких необхідно оцінити рівень релевалентності.

Таблиця 10.1

	t_1	t_2	t_3	t_4	R
$d^{(1)}$	1	0	0	1	1
$d^{(2)}$	1	1	0	1	1
$d^{(3)}$	0	1	1	0	1
$d^{(4)}$	0	0	1	1	0
$d^{(5)}$	0	0	1	1	0
$d^{(6)}$	1	1	0	0	0
rel_i	2	2	1	2	$rel = 3$
$nrel_i$	1	1	2	2	$nrel = 3$
$\exp(SV_i)$	4	4	1/4	1	

Припустимо, що запит складається із чотирьох термів – t_1, t_2, t_3, t_4 (відповідно, це та частина словника, яка істотна для аналізу). У таблиці окремим стовпцем наведена деяка експертна оцінка R релевалентності для документів з навчальної вибірки.

За цим даними розраховуються значення rel_i та $nrel_i$, а також експоненти від відповідної складової статусу релевалентності $\exp(SV_i)$, які наведені в останніх трьох рядках Таблиці 10.1.

Припустимо, що необхідно проаналізувати нові документи $d^{(7)}$, $d^{(8)}$, $d^{(9)}$, зустрічальність термінів t_1, t_2, t_3, t_4 для яких наведена у відповідних стовпцях Таблиці 10.2. Для нових документів статус релевалентності розраховується відповідно до вищенаведеної формули. Результати, наведені в Таблиці 10.2, свідчать, зокрема, про значимий рівень релевалентності документа $d^{(8)}$, розрахованого відповідно до ймовірнісної моделі.

Таблиця 10.2

	t_1	t_2	t_3	t_4	Статус (SV)
$d^{(7)}$	0	1	0	1	$2 = \log 4 + \log 1$
$d^{(8)}$	1	1	0	0	$4 = \log 4 + \log 4$
$d^{(9)}$	1	0	1	1	$0 = \log 4 + \log 1/4 + \log 1$

10.5. Ранжирування результатів пошуку

Ранжирування – процес, при якому пошукова система вибудовує результати пошуку в певному порядку за принципом найбільшої відповідності конкретним запитам. Представлення результатів пошуку залежить від алгоритму ранжирування. Ранжирування результатів пошуку за рівнем релевантності можливо не для всіх моделей пошуку (наприклад, неможливо для булевої моделі). Перспективний підхід до ранжирування –

використання багатопрофільних шкал, сформованих на основі метаданих, мережевих властивостей, даних про користувачів.

Ранжирування текстових і гіпертекстових документів істотно різниться. Текстові документи можуть ранжируватися за рівнями релевантності та іншими параметрами, що екстрагуються з текстів. Ранжирування гіпертекстових документів можливо також за властивостями, які обумовлюються мережевою структурою, гіперпосиланнями. Наприклад, для визначення авторитетності веб-сторінки як джерела інформації або посередника використовується аналіз графа, створеного веб-документами і відповідними гіперпосиланнями. Два найвідоміших алгоритму ранжирування веб-сторінок, заснованих на зв'язках, HITS (Hyperlink Induced Topic Search) і PageRank, були розроблені в 1996 році в IBM Дж. Клейнберг (J. Kleinberg) і в Стенфордському Університеті С. Бріном (S. Brin) і Л. Пейджем (L. Page).

Обидва алгоритми призначені для вирішення "проблеми надмірності", властивої широким запитам, збільшення точності результатів пошуку на основі методів аналізу складних мереж.

Алгоритм HITS

Алгоритм HITS (Hyperlink Induced Topic Search), запропонований Дж. Клейнбергом, є реалізацією латентно-семантичного індексування (див. п. 4.2) до ранжирування видачі інформаційно-пошукових систем.

Алгоритм HITS забезпечує вибір з інформаційного масиву кращих «авторів» (першоджерел, на які введуть посилання) і «посередників» (документів, від яких йдуть посилання цитування). Зрозуміло, що сторінка є хорошим посередником, якщо вона містить посилання на цінні першоджерела, і навпаки, сторінка є хорошим автором, якщо вона згадується хорошими посередниками.

Для кожного документу $d_j \in D$ рекурсивно обчислюється його значимість як автора $a(d_j)$ і посередника $h(d_j)$ за формулами:

$$a(d_j) = \sum_{i=1,\ i \neq j}^{|D|} h(d_i),\ \ h(d_j) = \sum_{i=1,\ i \neq j}^{|D|} a(d_i).$$

Покажемо, що алгоритм HITS подібний LSA. Введемо поняття матриці інціденцій A, елемент a_{ij} якої дорівнює одиниці, коли документ d_i містить посилання на документ d_j, і нулю в іншому випадку. Скористаємося сингулярним розкладанням: $A = USV^T$, де S – квадратна діагональна матриця з невід'ємними діагональними елементами $s_{i,i}$. Розглянемо матрицю $A^T A$, для якої справедливо: $A^T A = VSU^T USV^T = VS^2V^T$, де S^2 – діагональна матриця з елементами $s_{i,i}^2$. Відповідно, для матриці AA^T буде справедливо $AA^T = US^2U^T$. Як і при LSA, власні вектори, які відповідають найбільшим сингулярним значенням AA^T (або $A^T A$), будуть відповідати статистично найбільш важливим авторам (або посередникам).

Алгоритм обчислення рангів HITS призводить до зростання рангів документів при збільшенні кількості і ступеня пов'язаності документів відповідного співтовариства. В цьому випадку в результати пошуку системи, що використовує алгоритм HITS, можуть потрапити у великій кількості документи за темами, відмінним від інформаційної потреби користувача, але тісно пов'язаних між собою, тобто частина отриманих результатів може відхилятися від домінуючої тематики. У цьому випадку відбувається, так зване, зрушення тематики (topic drift) за рахунок наявності «тісно зв'язаних спільнот» документів (Tightly-Knit Community, TKC).

В матричній нотації наведені вище інтерактивні формули можна записати наступним чином:

$$a^{(k)} = G^T h^{(k-1)};\ \ h^{(k)} = Ga^{(k)},$$

де $a^{(k)}$ і $h^{(k)}$ – вектори розміром $1 \times n$, що відповідають авторству і посередництву кожного із n вузлів.

Наведений вище ітеративний алгоритм можна записати таким чином:

1. Ініціалізація: $h^{(0)} = e$.
2. Визначити у циклі:

$a^{(k)} = G^T h^{(k-1)}$;

$h^{(k)} = G a^{(k)}$;

$k = k + 1$.

Нормалізація $a^{(k)}$, $h^{(k)}$.

Наведений вираз можна спростити:

$a^{(k)} = G^T G a^{(k-1)}$;

$h^{(k)} = G G^T y^{(k-1)}$.

Ці два вирази визначають алгоритм знаходження авторства і посередництва через розрахунок власних векторів матриць $G^T G$ і $G G^T$.

Практична реалізація алгоритму HITS мовою системи Matlab

Приклад 1. Реалізація за ітеративним алгоритмом:

```
clear
N=10;
g=[0 1 1 0 1 0 1 0 1 1;
  0 0 1 1 0 0 1 0 0 0;
  1 0 0 0 1 1 0 1 1 0;
  0 1 0 0 0 1 0 0 0 1;
  0 0 0 1 0 0 1 1 0 1;
  0 0 1 0 0 0 1 0 1 1;
  0 0 0 1 0 0 0 1 0 1;
  1 1 0 0 0 1 0 1 1 0;
  1 1 0 1 0 1 1 0 0 1;
  1 0 0 1 0 0 0 0 1 0]
for i=1:N
        a(i)=1/N;
        h(i)=1/N;
```

```
end;
 for u=1:10
  for i=1:N

    for j=1:N
        if g(i,j)==1
            h(i)=h(i)+a(j);
        end;
    end;
  end;
  x=sum(h);
  for i=1:N
    h(i)=h(i)/x;
  end
h
  for i=1:N
     for j=1:N
        if g(j,i)==1
            a(i)=a(i)+h(j);
        end;
    end;
  end;
  x=sum(a);
  for i=1:N
    a(i)=a(i)/x;
  end
a
end
```

Приклад 1. Матрична реалізація:

```
clear
N=10;
      g=[0 1 1 0 1 0 1 0 1 1;
        0 0 1 1 0 0 1 0 0 0;
        1 0 0 0 1 1 0 1 1 0;
        0 1 0 0 0 1 0 0 0 1;
        0 0 0 1 0 0 1 1 0 1;
        0 0 1 0 0 0 1 0 1 1;
        0 0 0 1 0 0 0 1 0 1;
        1 1 0 0 0 1 0 1 1 0;
        1 1 0 1 0 1 1 0 0 1;
        1 0 0 1 0 0 0 0 1 0]
for i=1:N
       a(i)=1/N;
       h(i)=1/N;
end;
for u=1:10
  for i=1:N
    for j=1:N
      if g(i,j)==1
         h(i)=h(i)+a(j);
      end;
    end;
  end;
  x=sum(h);
  for i=1:N
    h(i)=h(i)/x;
  end
h
  for i=1:N
```

```
    for j=1:N
        if g(j,i)==1
            a(i)=a(i)+h(j);
        end;
    end;
  end;
  x=sum(a);
  for i=1:N
    a(i)=a(i)/x;
  end
a
end
k=0;
res=1;
eps=0.001;
for i=1:N
a(i)=1/N;
h(i)=1/N;
end;
while res>eps
    prev=a;
    k=k+1;
    a=a*g';
    a=a*g;
    a=a/sum(a)
    res=norm(a-prev,1);
end
k
a
h=a*g';
```

h=h/sum(h)

Алгоритм PHITS

Для вирішення цієї проблеми як деяке розширення стандартного алгоритму HITS був запропонований алгоритм PHITS. В рамках цього алгоритму передбачається: $D-$ множина документів, що цитують, $C-$ множина посилань, $Z-$множина класів (факторів). Передбачається також, що подія $d \in D$ відбувається з ймовірністю $P(d)$.

Умовні ймовірності $P(c|z)$ і $P(z|d)$ використовуються для опису залежностей між наявністю посилання $c \in C$, латентним фактором $z \in Z$ і документом $d \in D$.

Оцінюється функція правдоподібності:

$$L(D,C) = \prod_{c \in C, d \in D} P(d,c) = \prod_{c \in C, d \in D} P(d)P(c|d),$$

де

$$P(c|d) = \sum_{z \in Z} P(c|z)P(z|d).$$

Мета алгоритму PHITS полягає в тому, щоб підібрати $P(z)$, $P(c|z)$, $P(d|z)$, для того, щоб максимізувати $L(D,C)$.

Після цього:

$P(c|z)$ – ранги авторів;

$P(d|z)$ – ранги посередників.

Для обчислення рангів необхідно задати кількість факторів в множині Z, і тоді $P(c|z)$ буде характеризувати якість сторінки як автора в контексті тематики z. До недоліків методу треба віднести те, що ітеративний процес найчастіше зупиняється не на абсолютному, а на локальному максимумі функції правдоподібності L. Разом з тим в

ситуаціях, коли в множині знайдених веб-сторінок немає явного домінування тематики запиту, алгоритм PHITS кращий за HITS.

Алгоритм PageRank

Алгоритм PageRank близький по ідеології до літературного індексу цитування та розраховується для довільного документа з урахуванням кількості посилань з інших документів на даний документ. При цьому PageRank як і HITS, на відміну від літературного індексу цитування, не вважає все посилання рівнозначними.

Принцип розрахунку рангу веб-сторінок PageRank ґрунтується на моделі «випадкового блукання» користувача за наступним алгоритмом: він відкриває випадкову веб-сторінку, з якої переходить по випадково обраному гіперпосиланню. Потім користувач переміщається на іншу веб-сторінку і знову активізує випадкове гіперпосилання і т.д., постійно переходячи від сторінки до сторінки, ніколи не повертаючись. Іноді користувачу таке блукання набридає, і він знову переходить на випадкову веб-сторінку – не по посиланню, а набравши вручну деяку випадкову адресу. У цьому випадку ймовірність того, що блукаючи у веб-просторі користувач перейде на деяку певну веб-сторінку – це її ранг. PageRank веб-сторінки тим вище, чим більше інших сторінок посилається на неї, і чим ці сторінки є популярнішими.

Нехай є n сторінок $\{d_1, ..., d_n\}$, які посилаються на даний документ (веб-сторінку A), а $C(A)$ – загальне число посилань з веб-сторінки A на інші документи. Визначається деяке фіксоване значення δ як ймовірність того, що користувач, переглядаючи яку-небудь веб-сторінку з множини D, перейде на сторінку A за посиланням, а не набираючи її веб-адреси у явному вигляді. В рамках моделі ймовірність продовження цим користувачем веб-серфінгу по мережі з N веб-сторінок без використання гіперпосилань, шляхом ручного введення адреси (URL) з випадкової

сторінки $1-\delta$ (альтернатива переходу за гіперпосиланнями). Індекс PageRank $PR(A)$ для сторінки A розглядається як ймовірність того, що користувач опиниться в деякий випадковий момент часу на цій сторінці:

$$PR(A) = (1-\delta)/N + \delta\sum_{i=1}^{n}\frac{PR(d_i)}{C(d_i)}.$$

За цією формулою індекс сторінки легко підраховується простим ітераційним алгоритмом.

Випадкові подорожі

Для обчислення PageRank мережа представляється у вигляді орієнтованого графу, вершини якого відповідають сторінкам, а ребра — гіперпосиланням між ними. Нехай до пошукового індексу внесено *n* сторінок. Тоді для моделювання випадкової подорожі створюється матриця переходів M розміру $n \times n$. Елемент цієї матриці m_{ij}, що знаходиться в рядку i та стовпчику j має значення $1/k$, якщо сторінка з номером j має k вихідних посилань, серед яких є одне на сторінку з номером i. Якщо такого посилання нема, то елемент має значення 0.

Розподіл ймовірностей знаходження випадкового мандрівника може бути описано вектор-стовпчиком, рядок j якого дорівнюватиме ймовірності перебування на сторінці j. Цей вектор відповідатиме найпростішому, ідеалізованому варіанту PageRank.

Припустімо, що мандрівник може розпочати блукання з будь-якої із n індексованих сторінок з однаковою ймовірністю. Тоді значення елементів початкового вектора дорівнюватимуть $1/n$. Ймовірність знаходження на будь-якій зі сторінок на наступному кроці його блукання можна визначити обчисленням Mv_0. Іще через крок – $M\left(Mv_0\right) = M^2 v_0$, і так далі. Таким чином, процес випадкового блукання є Марковським, а вектор розподілів v — власним для матриці M.

За умови, якщо:

1. Граф гіперпосилань зв'язний — тобто, з кожної вершини існує шлях до будь-якої іншої;
2. В графі відсутні тупикові вершини — тобто, з кожної вершини є хоча б одне вихідне гіперпосилання, то з часом розподіл ймовірностей знаходження мандрівника сягне границі: $v = Mv$.

Проте, через те, що граф гіперпосилань реальної мережі Веб не завжди зв'язний та має тупикові вершини, у формулу випадкового блукання доводиться запровадити можливість випадкового «стрибка» на іншу вершину. Таким чином, формула обчислення розподілів знаходження v' матиме вигляд:

$$v' = \beta Mv + (1-\beta)e/n,$$

де:

β — константа, що зазвичай має значення в проміжку 0.8…0.9;

e — одиничний вектор (розміром n, всі елементи якого дорівнюють 1);

n — кількість індексованих сторінок, відповідно — кількість вершин графу;

βMv — моделює випадок, коли з ймовірністю β мандрівник вирішує обрати вихідне посилання з поточної сторінки;

$(1-\beta)e/n$ — вектор, кожен елемент якого дорівнює (1 − β)/n і який моделює появу мандрівника на будь-якій сторінці з ймовірністю (1 − β).

Якщо масштабувати PageRank таким чином, що де N — число всіх сторінок, для яких проводиться розрахунок PageRank, то PR (p) можна розглядати як розподіл ймовірності по всіх сторінках. Для обчислення PageRank складається матриця M розміром $N \times N$, де кожному елементу m_{ij} матриці присвоюється значення $PR_0(p) = 1/C(p)$ у тому випадку, якщо з i-ї сторінки є посилання на j-у, що все залишилися елементи матриці заповнюються нулями. Таким чином, обчислення PageRank зводиться до відшукання власного вектора матриці M що досягається

множенням матриці M на вектор PR_j на кожному кроці ітерації. Введення коефіцієнта загасання гарантує, що процес сходиться.

Практична реалізація методу PageRank мовою системи Matlab

Приклад 1. Реалізація з опису алгоритмом веб-серфінгу:

```
clear
N=10;
C=10000;
g=[0 1 1 0 1 0 1 0 1 1;
  0 0 1 1 0 0 1 0 0 0;
  1 0 0 0 1 1 0 1 1 0;
  0 1 0 0 0 1 0 0 0 1;
  0 0 0 1 0 0 1 1 0 1;
  0 0 1 0 0 0 1 0 1 1;
  0 0 0 1 0 0 0 1 0 1;
  1 1 0 0 0 1 0 0 1 0;
  1 1 0 1 0 1 1 0 0 1;
  1 0 0 1 0 0 0 0 1 0];
for i=1:10
  pr(i)=0;
end;
eps=0.01;
k=round((N-1)*rand(1))+1;
pr(k)=pr(k)+1;
res=1;
for w=1:100
   prev=pr;
for u=1:C
     i=round((N-1)*rand(1))+1;
  while g(k,i)==0
     i=round((N-1)*rand(1))+1;
```

```
  end
  pr(i)=pr(i)+1;
  gamma=rand(1);
  if gamma>0.2
     k=i;
  else
     k=round((N-1)*rand(1))+1;
     pr(k)=pr(k)+1;
  end
end
res
x=sum(pr);
pr=pr/x
res=norm(pr-prev,1)
end
```

Приклад 2. Матрична реалізація

```
clear
N=10;
C=50000;
g=[0 1/6 1/6 0 1/6 0 1/6 0 1/6 1/6;
  0 0 1/3 1/3 0 0 1/3 0 0 0;
  1/5 0 0 0 1/5 1/5 0 1/5 1/5 0;
  0 1/3 0 0 0 1/3 0 0 0 1/3;
  0 0 0 1/4 0 0 1/4 1/4 0 1/4;
  0 0 1/4 0 0 0 1/4 0 1/4 1/4;
  0 0 0 1/3 0 0 0 1/3 0 1/3;
  1/4 1/4 0 0 0 1/4 0 0 1/4 0;
  1/6 1/6 0 1/6 0 1/6 1/6 0 0 1/6;
  1/3 0 0 1/3 0 0 0 0 1/3 0];
for i=1:N
  pr(i)=1/N;
```

```
end;
alpha=0.15
eps=0.00001
rws=ones(1,N)*g'
nonz=find(rws)
zer=setdiff(1:N,nonz)
l=length(zer)
a=sparse(zer,ones(l,1),ones(l,1),N,1)
k=0;
res=1;
while res>eps
   prev=pr;
   k=k+1;
   pr=alpha*pr*g+(alpha*pr*a+1-alpha)*((1/N)*ones(1,N));
   res=norm(pr-prev,1);
end
k
pr
```

Незважаючи на відмінності HITS і PageRank, в цих алгоритмах загальне те, що авторитетність (вага) вузла залежить від ваги інших вузлів, а рівень "посередника" залежить від того, наскільки авторитетні вузли, на які він посилається.

Розрахунок авторитетності окремих документів сьогодні широко використовується в таких додатках, як визначення порядку сканування документів в мережі роботом інформаційно-пошукових систем, ранжирування результатів пошуку, формування тематичних оглядів тощо

Алгоритм Salsa

Алгоритм ранжирування Salsa (Stochastic Approach for Link-Structure Analysis – Стохастичний Алгоритм Аналізу Структури Зв'язків) був

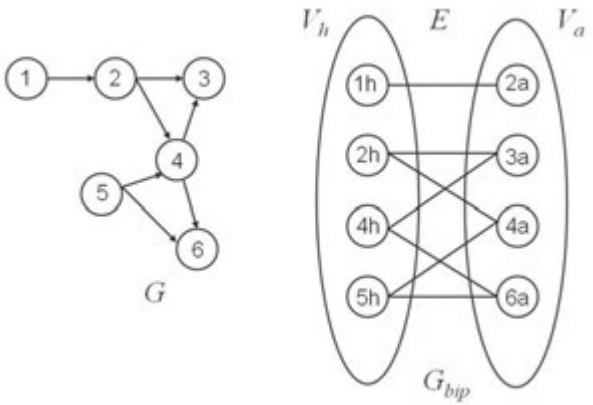
запропонований Ш. Мораном (Sh. Moran) і Р. Лемпелем (R. Lempel) як певний симбіоз алгоритмів PageRank і HITS, що дозволяє скоротити наслідки створення ТКС – «тісно пов'язаних спільнот» документів.

Як і в методі PageRank в Salsa передбачається модель випадкового блукання користувача по веб-графу, однак передбачається наявність двостороннього «серфінгу». Відповідно до алгоритму Salsa:

1. З довільного вузла v, користувач випадковим чином повертається до вузла u, який посилається на v. Вибір v робиться випадково, за умови, що вузли v і u належать веб-графу.

2. З u користувач навмання переходить до вузла w, якщо існує зв'язок $(u,\ w)$.

Веб-граф G (Рис.10.3 а) може бути перетворений на дводольний ненаправлений граф G_{bip} (Рис. 10.3 б), і визначений як сукупність $G_{bip} = (V_h,\ V_a,\ E)$, де h позначає посередників, V_h – сукупність вузлів-посередників (тих, з яких виходять посилання), a – авторів, V_a – сукупність вузлів-авторів (тих, на які ведуть посилання). Необхідно відзначити, що ті ж самі вузли можуть бути одночасно і авторами і посередниками.



а) *б)*

Рис. 10.3 – Алгоритм Salsa: конструювання дводольного графу

Кожна неізольована сторінка $s \in G$ представлена в G_{bip} одним або двома вузлами s_h і s_a.. У цьому дводольному графі Salsa реалізує два різних випадкових переходи. При кожному переході можливо «відвідування» вузлів тільки з однією з двох частин графа G_{bip}.

Кожен шлях довжини два в G_{bip} являє собою обхід одного гіперзв'язку (при проходженні від частки посередників до частки авторів в G_{bip}), і відхід уздовж гіперзв'язку (при проходженні в зворотному напрямку). Це рух у зворотному напрямку нагадує танець сальса, який асоціюється з назвою даного алгоритму.

Так як посередники і автори, які стосуються теми t повинні бути явно виражені в G_{bip} (доступні з багатьох вузлів завдяки прямим посиланнями або коротким шляхам), передбачається що автори з V_a і посередники з V_h, які стосуються теми t, будуть найбільш часто відвідуваними при випадкових «блукання» користувачів.

В алгоритмі Salsa досліджуються дві різних ланцюга Маркова, які асоціюються з цими випадковими блуканнями: ланцюг на стороні авторів G_{bip} (ланцюг авторів), і ланцюг на стороні посередників G_{bip}.

Такий підхід дозволяє ввести дві стохастичні матриці переходу ланцюгів Маркова, які визначаються наступним чином: будується матриця інціденцій W орієнтованого графа G. Позначимо W_r як матрицю, отриману діленням кожного ненульового елемента W на суму значень відповідного рядка, а через W_c - матрицю, отриману діленням кожного ненульового елемента W на суму елементів у відповідному стовпці. Тоді, матриця H, відповідна посередникам буде складатися з ненульових рядків і стовпців $W_r W_c^T$, а матриця авторів A, відповідно, буде складатися з ненульових рядків і стовпців $W_c^T W_r$. В рамках алгоритму Salsa ігноруються рядки і стовпці матриць A і H, які складаються повністю з нулів, так як

за визначенням G_{bip}, всі вузли мають не менше одного зв'язку. В результаті матриці A і H використовуються для обчислення рангів тим же шляхом, що і в алгоритмі HITS.

Показано що, ймовірність переходу до вузла v як до автора, що сходяться в процесі ітераційного процесу, має дуже просту форму:

$$\pi_v = c_1 \cdot InDegree(v),$$

Ймовірність повернення до вузла u як до посередника:

$$\pi_u = c_2 \cdot OutDegree(u),$$

де c_1 і c_2 – деякі константи, а $InDegree$ и $OutDegree$ – це кількість вихідних і вхідних посилань, відповідно.

Р. Лемпель і С. Моран продемонстрували, що алгоритм Salsa менш чутливий до ефекту тісно пов'язаних спільнот, ніж HITS, але за умов, що вручну в документах видаляються посилання, що не відносяться до досліджуваної теми. Ця вимога на практиці веде до великих витрат, в результаті чого поки не відомо випадків використання цього алгоритму ранжирування в реально працюючих системах.

10.6. Оцінка якості пошуку

Існує багато характеристик інформаційного пошуку, з яких дві визнані основними – це повнота (англ. – *recall*) і точність (англ. – *precision*). Багато уваги на даний час приділяється також такій змістовній характеристиці, як пертинентність. Ця характеристика інформаційно-пошукових систем означає відповідність отриманих у результаті пошуку документів інформаційним потребам користувача, а не формальній відповідності документа запиту.

Якщо під релевалентністю розуміється формальна відповідність інформації, видаваною системою, запиту, то на практиці використовується інше, неформальне поняття – пертинентність. Для користувача пертинентність, співвідношення обсягу корисної для нього інформації до загального обсягу отриманої інформації, має вирішальне значення. При

цьому варто враховувати, що формальний запит до системи є предметом творчого осмислення інформаційної потреби та не завжди точно відбиває останню. Невміння більшістю користувачів правильно формулювати запити та одержувати прийнятні результати породило наприкінці XX сторіччя думку відносно Інтернет, як про величезний інформаційний смітник. Досягнення високої пертинентності – основне поле конкурентної боротьби сучасних пошукових систем. Саме для максимального задоволення інформаційних потреб користувачів мережні ІПС сьогодні максимально інтелектуалізуються – набутили широкого застосування технології та методи семантичних і нейронних мереж, Text Mining.

Для обчислення показників якості пошуку прийнято розглядати таблицю, яку заповнюють за результатами пошуку в навчальній колекції документів. Цей підхід був запропонований у рамках створеної Американським Інститутом Стандартів (NIST) конференції з оцінки систем текстового пошуку – Text REtrieval Conference (TREC, http://trec.nist.gov/). Таблиця результатів пошуку має такий вигляд:

Документи	Видані	Не видані
Релевантні	*a*	*c*
Не релевантні	*b*	*d*

За допомогою цієї таблиці показники інформаційного пошуку розраховуються таким чином:

Коефіцієнт повноти (*recall*):

$$r = \frac{a}{a+c}.$$

Коефіцієнт точності (*precision*):

$$p = \frac{a}{a+b}.$$

Коефіцієнт акуратності (*accuracy*):

$$acc = \frac{a+d}{a+b+c+d}.$$

Помилка (*error*):

$$err = \frac{b+c}{a+b+c+d}.$$

F-міра (F-measure):

$$F = \frac{2}{\frac{1}{p}+\frac{1}{r}}$$

Середня точність (*average precision*):

$$ArgPrec = \frac{1}{k}\sum_{i=1}^{k} prec\_rel(i),$$

де k – кількість документів, релевантних деякому запиту, i – номер релевантного запиту документа, $prec\_rel(i)$ – точність $i-$го релевантного документа (документи ранжируються за релевантністю). Якщо $i-$й релевантний документ не знайдений, то $prec\_rel(i)$=0.

Як одна з визнаних метричних характеристик інформаційного пошуку розглядається 11-точковий графік повноти/точності TREC (РОМІП), яка відбиває зміни точності залежно від повноти та дає більш повну інформацію, ніж метрична характеристика у вигляді однієї цифри. По осі абсцис на графіку відкладаються значення повноти, по осі ординат – значення точності. Якщо для запиту відомо n релевантних документів, то повнота може приймати дискретні значення 0, 1/n, 2/n, ... , 1. Для того, щоб одержати загальний графік повноти/точності для множини запитів:

1. Розглядаються фіксовані значення повноти 0.0, 0.1, 0.2, ... , 1.0 (усього 11 значень).
2. Використається спеціальна процедура інтерполяції точності для даних фіксованих значень повноти.
3. Для множини запитів виробляється усереднення точності для заданих рівнів повноти.

Розглянемо приклад (Рис. 10.4). Нехай колекція документів містить 20 документів, 4 з яких релевантні запиту. Нехай система видає як

результати запиту всі ці документи, ранжирувані так, що релевантними є перший, другий, четвертий і п'ятнадцятий. Для різних значень точності в цьому випадку повнота приймає значення 0.25, 0.5, 0.75 та 1.0. Відповідно до правил інтерполяції, для значень повноти від 0 до 0.5 точність дорівнює 1.0 (тому що перші два документи задають рівень точності 1.0), для значень повноти 0.6 і 0.7 точність дорівнює 0.75, для значень повноти 0.8, 0.9 і 1.0 точність дорівнює 0.27 (4/15).

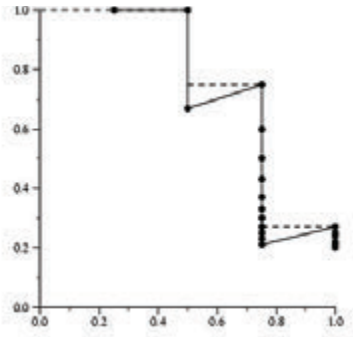


Рис. 10.4 – Залежність точності (вісь 0Y) від повноти (вісь 0X) для розглянутого прикладу. Пунктирною лінією позначені інтерпольовані значення

При оцінці різних інформаційно-пошукових систем за допомогою 11-точкового графіка кращою вважається та система, у якій висока точність досягається при малій повноті, що свідчить про гарне ранжирування результатів пошуку. Крім того, кращою визнається та система, для якої площа під відповідною інтерполяційною кривою є найбільшою.

Повнота пошуку (recall) тісно пов'язана з оперативністю охоплення інформації системою. Наприклад, створена одноразово база даних Інтернет-ресурсів є "зліпком" стану Інтернет у конкретний момент часу. Якщо ця база не буде поновлюватися, наявні в ній посилання на документи стануть «мертвими». Другий аспект, пов'язаний з повнотою інформації, що видається користувачеві за його запитом.

Додатково до розглянутих пошукових характеристик пошукових систем велике значення мають такі технологічні характеристики, як:

- швидкість обробки запитів;
- повнота охоплення ресурсів;
- доступність, тобто ймовірність одержання відповіді від системи;
- знаходження документів, подібних знайденим;
- можливість уточнення запитів;
- можливість підключення автоматичних перекладачів тощо.

На семінарі TREC доповіді традиційно зв'язуються з тематикою «доріжок» (*tracks*). Доріжка TREC – це фактично окреме конкретне завдання інформаційного пошуку зі строго визначеними правилами оцінки систем-учасників. Від учасників не вимагається участь у всіх доріжках відразу, тому в них є можливість зосередитися на вирішенні тільки однієї із запропонованих задач. Відповідно до одного з принципів TREC, завдання та правила проведення доріжок визначаються учасниками – оргкомітет лише координує проведення секцій. Приведемо приклади лише деяких доріжок, обговорюваних у рамках TREC:

- Пошук за запитом у веб-колекції.
- Пошук у нормативній колекції.
- Пошук по документу-зразку.
- Класифікація веб-сайтів.
- Класифікація веб-сторінок.
- Класифікація нормативно-правових документів.
- Кластеризація новинного потоку.
- Контекстно-залежне анотування.

Питання для самоперевірки

1. Перелічить моделі пошуку
2. Основні властивості булевої моделі пошуку.
3. Основні властивості розширеної булевої моделі пошуку.

4. Основні властивості ймовірнісної моделі пошуку.
5. Основні властивості векторно-просторової моделі пошуку.
6. Основні властивості нечіткого пошуку.
7. Основні принципи ранжирування результатів пошуку.

Література до розділу

1. *Маннинг Д. , Рагхаван П., Шютце Х.* Введение в информационный поиск. – «ВИЛЬЯМС», 2014. – 528 с. 2. *Ландэ Д.В., Снарский А.А., Безсуднов И.В.* Интернетика: Навигация в сложных сетях: модели и алгоритмы. – М.: Либроком (Editorial URSS), 2009. – 264 с. 3. *Додонов А.Г., Ландэ Д.В., Путятин В.Г.* Компьютерные сети и аналитические исследования. – К.: ИПРИ НАН Украины, 2014. – 486 с. 4. *Герасимов Б.М., Сергеев О.Ю., Субач И.Ю.* Извлечение информационных фраз из первичных электронных документов в информационно-поисковых системах // Управляющие системы и машины, 2006. – №1. – С.26–29. 5. *Сергеєв О.Ю., Сомов С.В., Субач І.Ю.* Метод модифікації вихідних запитів користувачів автоматизованими інформаційно-пошуковими системами // Радіоелектронні і комп'ютерні системи, 2006. – №5. – С.30–35.

11. ЕЛЕМЕНТИ КОМП‘ЮТЕРНОЇ ЛІНГВІСТИКИ

11.1. Закони Ципфа

Дж. Ципфа (G. Zipf) вивчав використання статистичних властивостей мови в текстових документах і виявив кілька емпіричних законів, які представив як емпіричне підтвердження свого «принципу найменшої кількості зусиль». Він експериментально показав, що розподіл слів природної мови підпорядковується закону, який часто називають першим законом Ципфа, який відноситься до розподілу частоти слів у тексті. Цей закон можна сформулювати таким чином. Якщо для якогось досить великого тексту скласти список всіх слів, що зустрілися в ньому, а потім ранжирувати ці слова у порядку убування частоти їх появи в тексті, то для будь-якого слова добуток його рангу і частоти появи буде величиною постійною: $f \cdot r = c$, де f – частота появи слова в тексті; r – ранг слова в списку; c – емпірична постійна величина (коефіцієнт Ципфа). Для слов'янських мов, зокрема, коефіцієнт Ципфа становить приблизно 0,06-0,07 (Рис. 11.1).

Наведена залежність відображає той факт, що існує невеликий словник, який становить більшу частину слів тексту. Це головним чином службові слова. Наприклад, аналіз роману «Том Сойєр», дозволив виділити 11 000 англійських слів. При цьому було виявлено дванадцять слів (the, and, і ін.), кожне з яких охоплює понад 1% лексем в романі. Закон Ципфа був багаторазово перевірений на багатьох масивах. Ципф пояснював наведений вище гіперболічний розподіл «принципом найменшої кількості зусиль», припускаючи що при створенні тексту менше зусиль йде на повторення деяких слів, ніж на використання нових, тобто на звернення до «оперативної пам'яті, а не до довготривалої».

Ципф сформулював ще одну закономірність, так званий другий закон Ципфа, що полягає у тому, що частота і кількість слів, які входять в текст з даної частотою, також пов'язані подібним співвідношенням, а саме:

$$N(f) = \frac{B}{f^b},$$

де $N(f)$ – кількість різних слів, кожне з яких використовується в тексті f раз, B - константа нормування.

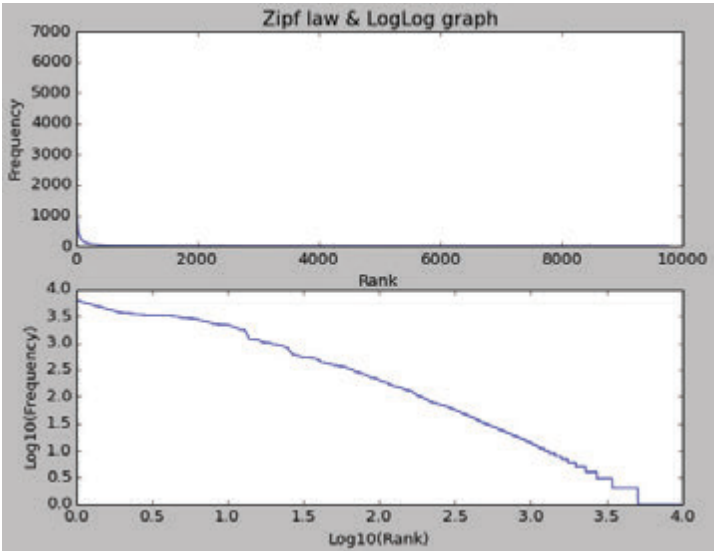


Рис. 14.1 – Типовий графік, що підтверджує закон Ципфа

Існує проста кількісна модель визначення залежності частоти від рангу. Припустимо, що генерується випадковий текст мавпою на друкарській машинці. З ймовірністю p генерується пробіл, а з ймовірністю $(1-p)$ – інші символи, кожен з яких має рівну ймовірність. Показано, що отриманий таким чином текст буде давати результати, близькі за формою до розподілу Ципфа.

Легко бачити, що ймовірність генерації слова зменшується з довжиною, ймовірність слова з n непробільних символів дорівнює:

$$(1-p)^n \cdot p,$$

де p – ймовірність генерації пробілу.

Ця модель була вдосконалена відповідно до фактичних емпіричними даними, коли ймовірності генерації окремих символів були задані на основі аналізу великого текстового масиву. Отримане цілком пояснює закон Ципфа.

Більш складну модель генерації випадкового тексту, що задовольняє другому закону Ципфа, запропонував Г.А. Саймон (H.A. Simon). Умови цієї моделі досить прості: якщо текст досяг розміру в n слів, тоді те, яким буде $(n+1)$-е слово тексту визначається двома припущеннями:

1. Нехай $N(f,n)$- кількість різних слів, кожне з яких використовувалося f раз серед перших n слів тексту. Тоді ймовірність того, що $(n+1)$-м виявиться слово, яке до того використовувалося f раз пропорційно fЧN(f,n) – загальній кількості появи всіх слів, кожне з яких до цього використовувалося f раз.
2. З імовірністюd $(n+1)$-м словом буде нове слово.

Розподіл Ципфа часто спотворюється на практиці у зв'язку з нестачею обсягів текстових корпусів, що призводить до проблеми оцінки параметрів статистичних моделей. Разом з тим співвідношення між рангом і частотою була взята Солтоном в 1975 році як відправна точка для вибору термінів для індексування. Далі їм розглядалася ідея сортування слів відповідно до їх частоти в текстовому масиві. Як другий крок високочастотні слова можуть бути усунені, тому що вони не є хорошими відмітними ознаками для окремих документів з текстового масиву. На третьому кроці терми з низькою частотою, яка визначається деяким порогом (наприклад слова, які зустрічаються тільки один раз або двічі) видаляються, тому що вони рідко використовуються в запитах користувачів. Використовуючи цей підхід, можна значно зменшити розмір індексу пошукової системи. Більш принциповий підхід до підбору індексних термів – урахування їх вагових значень. У вагових моделях середньочастотні терми виявляються

найвагомішими, так як вони є найбільш суттєвими при відборі того чи іншого документа (найбільш частотні слова зустрічаються одночасно у великій кількості документів, а низькочастотні можуть не входити в документи, що цікавлять користувача).

Ще один емпіричний закон, сформульований Ципфом полягає в тому, що кількість значень слова корелює з квадратним коренем його частоти. Малося на увазі, що нечасто використовувані слова більш однозначні, а це підтверджує те, що високочастотні слова не підходять для внесення в індекси інформаційно-пошукових систем.

Ципф також визначив, що довжина слова обернено пропорційна його частоті, що може бути легко перевірено шляхом простого аналізу списку службових слів. Останній закон дійсно є прикладом принципу економії зусиль: коротші слова вимагають менше зусиль при відтворенні, і таким чином, використовуються більш часто.

Хоча закон Ципфа дає цікаві характеристики слів в текстових масивах, у загальному випадку помічені деякі обмеження його застосовності при отриманні статистичних характеристик документальних масивів, що складаються з множини незалежних документів різних авторів. Законами Ципфа задовольняють не тільки слова з одного тексту, але багато об'єктів сучасного інформаційного простору.

Нижче наведено текст програми мовою системи Matlab, що реалізує перевірку закономірності Ципфа.

```
clear
f=fopen('strug.txt','rt');
s=' ';
while feof(f)==0
        buf=fgetl(f);
        s=[s ' ' buf];
end
upper(s)
```

```
s=razdel(s);  % Вилучення службових символів (окремий модуль)
 n=size(p);
 N=n(1,2);
 st='';
 for i=1:N-1
         w=s(p(i)+1:p(i+1));
         st=char(st,w);
 end
 st1=sortrows(st);
 old='';
 ind=1;
 j=0;
 n=size(st1);
 M=n(1,2)
 N=n(1,1)
 old=st1(1,1:M);
 ind=1;
 for i=2:N
    if strcmp(old,st1(i,1:M))==1
       j=j+1;
    else
       nw(ind)=j;
       ind=ind+1;
       old=st1(i,1:M);
       j=1;
   end
 end
 nw(ind)=j;
 ind;
 x=sort(nw,'descend');
```

```
x
plot(x)
```

11.2. Закономірність Бредфорда

Закономірність С. Бредфорда (S. Bredford), відомого документаліста, одного з авторів універсальної десяткової класифікації – УДК, полягає в наступному: якщо наукові журнали розташувати в порядку убування числа розміщених у них статей з конкретного предмета, то отриманий список можна розбити на три зони таким чином, щоб кількість статей в кожній зоні по заданому предмету була однаковою. Ці три зони представляють: ядро – профільні журнали, безпосередньо присвячені розглянутій тематиці, журнали, частково присвячені заданій області та журнали, тематика яких досить далека від розглянутого предмета. С. Бредфорд в 1934 р встановив наступне співвідношення для кількості журналів в різних зонах:

$$\frac{N_3}{N_2} = \frac{N_2}{N_1} = const,$$

де кількість журналів в першій зоні – N_1, в другій – N_2, в третій – N_3.

Бредфорд спочатку розглядав знайдену закономірність тільки як специфічний випадок розподілу Ципфа для системи періодичних видань з науки і техніки. Однак в подальшому виявилося, що ця ж закономірність справедлива і для періодичних видань з багатьох інших предметних областей, а також для наборів веб-сайтів, що належать до певної тематики.

11.2. Закон Хіпса

У комп'ютерній лінгвістиці емпіричний закон Г.С. Хіпса (H.S. Heaps) пов'язує обсяг документа з об'ємом словника унікальних слів, які входять в цей документ. Здавалося б, словник унікальних слів повинен насичуватися, а його обсяг стабілізуватися при збільшенні обсягів тексту. Виявляється, це не так! Для всіх відомих сьогодні текстів відповідно до закону Хіпса, ці значення пов'язані співвідношенням (рис. 11.2):

$$v(n) = \alpha n^{\beta},$$

де v – це обсяг словника унікальних слів, складений з тексту, який складається з n унікальних слів, α і β – емпірично параметри. Для європейських мов α приймає значення від 10 до 100, а β – від 0.4 до 0.6.

Закон Хіпса справедливий не тільки для унікальних слів, але і для багатьох інших інформаційних об'єктів, що цілком природно, тому що доведено, що він є наслідком закону Ципфа.

Нижче наведено текст програми мовою системи Matlab, що реалізує перевірку закономірності Хіпса.

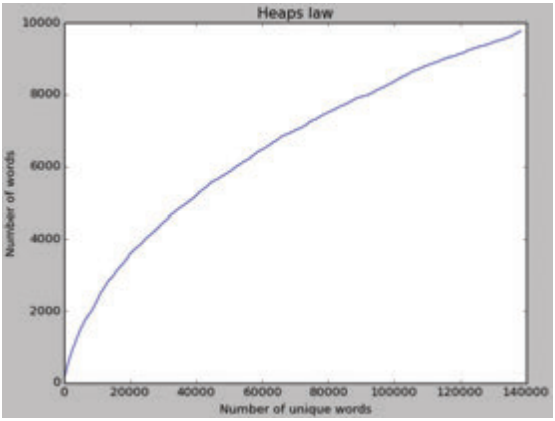


Рис. 11.2 – Типовий графік, що підтверджує закон Хіпса

Нижче наведено текст програми мовою системи Matlab, що реалізує перевірку закономірності Хіпса.

```
clear
f=fopen('strug.txt','rt');
s1='';
while feof(f)==0
  buf=fgetl(f);
  s1=[s1 ' ' buf];
  end
s=upper(s1);
```

```
s=razdel(s);
p=findstr(s,' ');
n=size(p);
N=n(1,2);
st='';
for i=1:N-1
if p(i+1)>p(i)+1
   w=s(p(i)+1:p(i+1));
   st=char(st,w);
end
end
n=size(st);
M=n(1,2);
N=n(1,1);
uw(1,1:M)=st(1,1:M);
nuw=1;
for i=1:N
   pr=0;
   for j=1:nuw
      if strcmp(st(i,1:M),uw(j,1:M))==1
         pr=1;
         break;
      end
   end
   if pr==0
      nuw=nuw+1;
      uw(nuw,1:M)=st(i,1:M);
   end
   nw(i)=nuw;
end
```

nw'

plot(nw)

11.4. Визначення ваги слів у документі

TFIDF

TFIDF (TF – Term Frequency, IDF – Inverse Document Frequency) – статистична міра, яка використовується для оцінки важливості слова в контексті документа, що є частиною колекції документів або корпусу. Вага деякого слова пропорційний кількості вживання цього слова в документі, і обернено пропорційна частоті вживання слова в інших документах колекції.

TF (Term Frequency – частота слова) – відношення числа входження деякого слова до загальної кількості слів документа. Таким чином, оцінюється доля слова t_i в межах окремого документа.

$$tf(t_i, d) = \frac{n_i}{\sum_k n_k},$$

де n_i є число входжень слова в документ, а в знаменнику – загальне число слів в даному документі.

IDF (Inverse Document Frequency – зворотна частота документа) – інверсія частоти, з якою деяке слово зустрічається в документах колекції. Урахування IDF зменшує вагу широковживаних слів. Для кожного унікального слова в межах конкретної колекції документів існує тільки одне значення IDF.

$$idf(t_i, D) = \log \frac{|D|}{|d_j \supset t_i|},$$

де $|D|$— кількість документів в корпусі; $|d_j \supset t_i|$ – кількість документів, в яких зустрічається t_i (коли t_i не дорівнює 0).

Вибір основи логарифма в формулі декілька знижує роль інверсної частоти. Таким чином, міра TFIDF є добутком двох співмножників:

$$TFIDF\left(t_i, D\right) = \frac{n_i}{\sum\limits_k n_k} \cdot \log \frac{\left|D\right|}{\left|d_j \supset t_i\right|}.$$

Алгоритм TFIDF найчастіше використовують при аналізі текстів в інформаційному пошуку, наприклад, як один із критеріїв релевантності документа пошуковому запиту, при розрахунку міри близькості документів при кластеризації.

Дисперсійна оцінка дискримінантної сили слова

Дискримінантна оцінка сили слова – це оцінка важливості даного слова в тексті. Існують декілька методів визначення дискримінантної сили слова. Наприклад в векторно-просторовій моделі пошуку застосовують метод TFIDF, в якому використовується частота появи терміна в документі та частота появи документів у загальному масиві з цим терміном.

Основна ідея застосування оцінки дискримінантної сили слова полягає в тому, що слова, які рівномірно розподілені по всьому тексту, не будуть ключовим.

Зафіксуємо деяке слово A з вихідного тексту. Позначимо $\langle \Delta A \rangle$ – середнє значення мінімальної кількості інших слів між двома повтореннями слова A. Аналогічно визначимо середнє квадратичне значення – $\langle \Delta A^2 \rangle$. Тоді дисперсійна оцінка слова A визначається за формулою:

$$\sigma_A = \frac{\sqrt{\langle \Delta A^2 \rangle - \langle \Delta A \rangle^2}}{\langle \Delta A \rangle}.$$

11.5. Основні елементи та технології Text Minig

Інформаційно-пошукові системи та системи інтеграції інформаційних потоків вирішують проблему знаходження необхідної інформації, але залишають без уваги такі завдання, як узагальнення даних – їхню обробку та аналіз, зокрема, автоматичне реферування, виявлення особливостей та

аномалій, витяг понять тощо. Одним із самих перспективних напрямків узагальнення інформаційних потоків на даний час є метод "глибинного аналізу текстів" (Text Mining).

Ідеологію Text Mining у застосуванні до інформаційних потоків можна сформулювати як контент-моніторинг – постійне відтворене в часі виконання їхнього змістовного аналізу (контент-аналізу). Саме безперервна аналітична обробка повідомлень є характерною рисою цього методу, що дозволяє формувати інформаційні портрети, автоматичні дайджести, виявляти сюжетні ланцюжки, нові поняття і їхні взаємозв'язки, розраховувати різноманітні рейтинги. Передбачається, що системи такого типу зможуть позбавити користувачів від інформаційного шуму, дозволять виявляти головні тенденції, знаходити події, які взаємозв’язані тощо.

Технологія глибинного аналізу тексту – Text Mining – це той самий інструментарій, який дозволяє аналізувати великі обсяги інформації в пошуках тенденцій, понять і взаємозв'язків, здатних допомогти в прийнятті стратегічних рішень. Крім того, Text Mining – це новий вид пошуку, що на відміну від традиційних підходів не тільки знаходить списки документів, формально релевантних запитам, але й допомагає зрозуміти зміст документів, розібратися із тією або іншою проблематикою.

Клод Фогель, один із засновників і головний технолог компанії Semio, свого часу пояснював: "Використовуючи аналогію з бібліотекою, технологія Text Mining подібна до відкриття книги перед читачем з підкресленою необхідною інформацією. Порівняйте це з видачею читачеві купи документів і книг, у яких десь міститься інформація, потрібна читачеві, однак знайти її непросто".

Процес осмисленого пошуку є далеко не тривіальним, часто в колекції документів присутній тільки натяк на необхідну інформацію. Необхідні потужні інтелектуальні можливості щоб знайти те, що потрібно. У назві технології слово "mining" (видобуток руди) виступає як метафора знаходження глибоко "заритої" інформації.

Для обробки текстів у технології Text Mining цілком справедливе визначення, дане для загального випадку видобутку даних (Data Mining) Пятецьким-Шапиро з GTE Labs: "Процес виявлення в сирих даних раніше невідомих нетривіальних практично корисних і доступних для інтерпретації знань, необхідних для прийняття рішень у різних сферах людської діяльності." Як і більшість когнітивних технологій – Text Mining – це алгоритмічне виявлення раніше не відомих зв'язків і кореляцій у наявних текстових даних.

Сформована у середині 90-х років XX століття як напрямок аналізу неструктурованих текстів, технологія Text Mining відразу ж взяла на озброєння методи класичного аналізу даних, таких як класифікація або кластеризація. В Text Mining з'явилися й додаткові можливості, такі як автоматичне реферування текстів і витяг понять, фактів.

Можливості сучасних систем Text Mining можуть застосовуватися при управлінні знаннями для виявлення шаблонів у текстах, для автоматичного "виштовхування" або розміщення інформації із профайлів, що цікавлять користувачів, створення документальних оглядів. Технології Text Mining, крім того, властива об'єктивність – відсутній суб'єктивізм, притаманний людині-аналітику.

Важлива компонента технології Text Mining пов'язана з витягом із тексту документа його характерних елементів або властивостей, які можуть використовуватися як метадані, ключові слова, анотації. Інше важливе завдання полягає у віднесенні документа до деяких категорій із заданої схеми систематизації. Text Mining також забезпечує новий рівень семантичного пошуку документі. Відповідно до вже сформованої методології до основних елементів Text Mining відносяться: сумаризація (summarization), тобто автоматичне реферування, витяг з текстів феноменів, понять (*feature extraction*), кластеризація (*clustering*), класифікація (*classification*), відповідь на запити (*question answering*), тематичне індексування (*thematic indexing*) і пошук за ключовими словами

(*keyword searching*). Також у деяких випадках наведений набір доповнюють засоби підтримки та створення таксономії (*oftaxonomies*) і тезаурусів (*thesauri*).

О. Лінден, директор компанії Gartner Research, виділив чотири основних види застосувань технологій Text Mining:

- класифікація тексту, у якій зазвичай використовуються статистичні кореляції термів для побудови правил розміщення документів у визначені категорії;
- кластеризація, що базується на ознаках документів і використовує лінгвістичні та математичні методи без залучення заздалегідь визначених категорій. Результат кластеризації – таксономія або візуальна карта, що забезпечує ефективне охоплення великих обсягів даних;
- семантичні мережі або аналіз зв'язків, які визначають появу термів у документі для забезпечення навігації;
- витяг фактів, призначений для одержання деяких фактів з тексту з метою поліпшення класифікації, пошуку та кластеризації.

Так склалося, що найчастіше в Text Mining зустрічається задача класифікації – віднесення об'єктів бази даних до заздалегідь визначених категорій. Фактично задача класифікації – це класична задача розпізнавання, де по навчальній вибірці система відносить новий об'єкт до тієї або іншої категорії. Особливість же системи Text Mining полягає в тому, що кількість об'єктів та їх атрибутів може бути дуже великою, тому повинні бути передбачені інтелектуальні механізми оптимізації процесу класифікації.

Друге завдання – кластеризація – виділення компактних підгруп об'єктів із близькими властивостями. Система повинна самостійно знайти ознаки та розділити об'єкти по підгрупах. Кластеризація, як правило, передує класифікації, оскільки дозволяє визначити групи об'єктів. Розрізняють два основних типи кластеризації – ієрархічну та бінарну.

Ієрархічна кластеризація полягає в побудові дерева кластерів, у кожному з яких розміщається невелика група документів. Бінарна кластеризація, наприклад, забезпечує групування документальних кластерів за властивостями подібності, тобто в один кластер містяться найближчі за своїми властивостями документи.

Можна назвати ще кілька завдань технології Text Mining, наприклад, прогнозування, яке полягає в тому, щоб передбачати за значеннями одних ознак об'єкта значення інших.

Ще одне завдання – знаходження виключень, аномалій, тобто пошук об'єктів, які своїми характеристиками сильно виділяються в загальній колекції. Для цього спочатку з'ясовуються середні параметри об'єктів, а потім досліджуються ті об'єкти, параметри яких найбільше відрізняються від середніх значень. Як відомо, пошук виключень широко застосовується, наприклад, у роботі спецслужб. Подібний аналіз часто проводиться після класифікації, для того, щоб з'ясувати, наскільки остання була точною.

Окремо від задачі кластеризації стоїть задача пошуку зв'язаних ознак (полів, понять) окремих документів. Від прогнозування це завдання відрізняється тим, що заздалегідь не відомо, за якими саме ознаками реалізується взаємозв'язок, у цьому випадку ціль саме в тому і складається, щоб знайти зв'язки ознак. Це завдання близьке до кластеризації, але не за масивом документів, а за множиною властивих їм ознак.

І нарешті, для обробки та інтерпретації результатів Text Mining велике значення має візуалізація, яка є ключовою ланкою при представленні схем неструктурованих текстових документів. Зокрема, сучасні системи класу Text Mining можуть здійснювати аналіз великих масивів документів і формувати предметні покажчики понять і тематик, притаманних цим документам. Візуалізація може застосовуватися при дослідженні документів та їх класів, також використовується як спосіб представлення контенту всього масиву документів, для реалізації навігаційного механізму.

11.6. Мережі мови

Першим кроком у застосуванні теорії складних мереж до текстових документів є формування мережевої моделі цих документів у вигляді сукупності вузлів і зв'язків, тобто побудова мереж мови (Language Network), в яких виявляються найвагоміші вузли, іноді це так звані опірні слова або відповідні словосполучення.

Поряд з послідовним аналізом текстових документів, побудова мереж, вузлами яких є такі елементи, як слова або словосполучення, тобто фрагменти природної мови, дозволяє виявляти структурні елементи текстів, без яких тексти втрачають свою зв'язність. Відомо декілька підходів до побудови мереж з текстів і різні способи інтерпретації вузлів і зв'язків, що приводить, відповідно, до різних видів представлення таких мереж. Вузли можуть бути сполучені між собою, якщо відповідні їм слова стоять поряд у тексті, належать до одного речення або абзацу, сполучені синтаксично, або семантично .

Збереження синтаксичних зв'язків між словами призводить до представлення тексту у вигляді спрямованої мережі (Directed Network), де напрям зв'язку відповідає підпорядкуванню слова.

Якщо поставити у відповідність кожному слову вузол мережі і з'єднати кожні два вузли зв'язком тоді, коли відповідні ним слова стоять у реченні поруч, то таке представлення називають *L*-простором. У *L*-просторі, так само як і в інших наведених нижче мережевих моделях, при виникненні кратних зв'язків прийнято зберігати лише один з них.

Традиційно розрізняють чотири різновиди мереж мови (просторів):

1. *L*-простір. Зв'язуються сусідні слова, які належать до одного речення. Кількість сусідів для кожного слова (вікно слова) визначається радіусом взаємодії R, найчастіше розглядається випадок R = 1.

2. *B*-простір. Розглядаються вузли двох видів, відповідні реченням і словам, що належать до них.

3. *P*-простір. Усі слова, які належать до одного речення, зв'язуються між собою.

4. *C*-простір. Речення зв'язуються між собою, якщо у них застосовуються однакові слова.

У випадку *L*-простору зв'язки можуть враховувати не лише «найближчих сусідів», але і групи з декількох слів, які знаходяться на певній відстані один від одного. Для цього вводиться поняття «радіусу дії» R: при $R = 1$ зв'язок існує лише між найближчими сусідами, при $R = 2$ – між найближчими і наступними близькими сусідами і т. д. Змінна R може приймати значення від $R = 1$ до $R_{\max}$, де $R_{\max} + 1$ – загальна кількість слів у реченні. Зростання «радіусу взаємодії» R у цьому випадку призводить до зростання кількості зв'язків, досягаючи насичення при $R = R_{\max}$.

Ще один спосіб представити текст у вигляді мережі полягає у використанні дводольних (bipartite) графів. У такому представленні (*B*-простір) розглядаються вузли двох видів. Один вид відповідає реченням, другий – словам. Зв'язок між різними вузлами означає, що слово належить реченню.

У *P*-просторі усі слова, що належать одному реченню, вважаються зв'язаними між собою.

У *C*-просторі вузли відповідають реченню, а зв'язок між вузлами-реченнями встановлюється у тому випадку, якщо у відповідних реченнях є загальні слова.

Для мережі, побудованої на підставі Британського національного корпусу (*L*-простір мови, $R = 1$) виявилось, що ця мережа англійської мови безмасштабна, а поведінка степеню $P(k)$ характеризується двома режимами степеневого розподілу із значеннями відповідних степеневих показників $\gamma = 1{,}5$ для $k < 2000$ і $\gamma = 2{,}7$ для $k > 2000$.

Згідно з визначенням, якщо середня довжина найкоротшого шляху зростає з розміром (кількістю вузлів) мережі повільніше за будь-яку степеневу функцію, то мережа є «малим світом». Мережі малого світу надзвичайно компактні. Для згаданої вище мережі англійської мови довжина найкоротшого шляху складає лише $< l >= 2,63$. Оскільки зростання R призводить лише до додавання нових зв'язків, то $< l >$ зменшуються із зростанням R.

Специфічною формою кореляції в мережах є утворення кластерів. Коефіцієнт кластеризації C характеризує схильність мережі до утворення сполучених трійок вузлів. Відомо, що для повного графа $C = 1$, а для мережі у формі дерева $C = 0$.

Відношення середнього коефіцієнта кластеризації досліджуваних мереж до коефіцієнта кластеризації класичного випадкового графа свідчить про те, що мережі мови є добре корельованими структурами. Такі кореляції збільшуються із зростанням «радіусу взаємодії» R.

Для Британського національного корпусу на підставі аналізу текстів, які містили $\approx 10^7$ слів, набуте значення коефіцієнта кластеризації $< C >= 0,687$.

У випадку розгляду P-простору кожне слово-вузол пов'язано з усіма іншими словами, які належать спільному реченню. Таким чином, кожне речення тексту входить у мережу як повний граф – кліка взаємозв'язаних вузлів. Різні речення-кліки об'єднуються в мережу завдяки загальним словам. У L-просторі слова зв'язуються в межах вікна, розмір якого характеризуються величиною R. Коли розмір вікна R стає рівним розміру речення, то представлення цього речення в L- і в P-просторах співпадають. Відповідно, коли розмір вікна стає рівним розміру найбільшого речення тексту ($R = R_{\max}$), то представлення усього тексту в L- і в P-просторах співпадають.

На практиці доведено, що мережа мови є сильно корельованим безмасштабним малим світом (Scale-Free Small World). Існує ряд праць, в яких зроблена спроба пояснити властивості мереж мови за допомогою сценарію переважного приєднання (Preferential Attachment), розглядаючи їх як результат процесу зростання, коли нові вузли-слова з більшою ймовірністю приєднуються до вузлів-хабів, що вже мають багато зв'язків.

Нижче наведено текст програми мовою системи Matlab, що реалізує побудову і відображення мережі мови за деяким літературним твором, текст якого наведено у файлі str1.txt, а як поняття – ключові слова, вузли мережі мови використовуються заздалегідь вибрані слова, які розміщено у файлі sll.txt. Для відображення мережі застосовувалися функції biograph і view (останні рядки програми).

```
clear
g=fopen('sl1.txt','rt');
N=0;
while feof(g)==0
   N=N+1;
   buf=fgetl(g);
   if N==1
      names=buf;
   else
      names=char(names,buf);
   end
end
n=size(names);
M=n(1,2)
N=n(1,1)
for i=1:N
   for j=1:N
      g(i,j)=0;
```

```
    end
end
f=fopen('str1.txt','rt');
i=0;
while feof(f)==0
    i=i+1;
    s=fgetl(f);
    s=upper(s);
    s=razdel(s);
    if length(s)>0
        for ii=1:N
            c=M-1;
            x=findstr(names(ii,1:M),' ');
            if length(x)>0
                c=x(1)-1;
            end;
            for jj=1:N
                d=M-1;
                x=findstr(names(jj,1:M),' ');
                if length(x)>0
                    d=x(1)-1;
                end;
                if ii~=jj
          b=length(regexpi(s,names(ii,1:c)))*length(regexpi(s,names(jj,1:d)));
                    if b>0
                        g(ii,jj)=1;
                    end
                end
            end
        end
```

```
    end
end
fclose(f)
obj=biograph(g,names);
view(obj)
```

11.7. Мережі горизонтальної видимості

В рамках концепції складних мереж запропоновано декілька методів побудови мереж на основі часових рядів, серед яких можна назвати декілька методів побудови графів видимості, зокрема, так званий граф горизонтальній видимості (Horizontal Visibility Graph – HVG). Ці підходи також дозволяють будувати мережеві структури на підставі текстів, в яких окремим словам або словосполученням деяким спеціальним чином поставлені у відповідність числові вагові значення. Як функцію, яка ставить у відповідність слову число, можна розглядати, наприклад, порядковий номер унікального слова в тексті, довжину слова, загальноприйняту оцінку TFIDF (в канонічному вигляді, що дорівнює добутку частоти слова в фрагменті тексту – Term Frequency – на двійковий логарифм від величини , зворотної кількості фрагментів тексту, в яких це слово зустрілося – Inverse Document Frequency) або її варіанти, а також інші вагові оцінки.

Ряди з цифрових значень відповідних слів можна перетворити в графи горизонтальній видимості, в яких вузлам відповідають не тільки цифрові значення, але самі слова, які мають певне змістовне значення. Мережа мови з використанням алгоритму горизонтальній видимості будується в три етапи. На першому – на горизонтальній осі відзначається ряд вузлів, кожен з яких відповідає словам в порядку появи в тексті, а по вертикальній осі відкладаються вагові чисельні оцінки (візуально – набір вертикальних ліній, рис. 11.1).

На другому етапі будується традиційний граф горизонтальній видимості. Для цього між вузлами встановлюється зв'язок, якщо вони знаходяться в "прямої видимості", тобто якщо їх можна з'єднати горизонтальною лінією, яка не перетинає ніякої вертикальної лінії, розміщеної між цими вузлами.

Алгоритм побудови графа горизонтальній видимості можна навести зручним для обчислення способом. Так наприклад, на Рис. 11.3 для вузла-слова A_1^{n+2} (верхній індекс – номер слова в тексті, нижній – номер появи конкретного слова, у даному випадку – слова A) суміжними в мережі вважаються слова B_3^n і C_7^{n+5} та встановлюються ребра-зв'язки, такі що B_3^n – найближчим зліва від A_1^{n+2} слово з ваговою оцінкою $\sigma_n = \sigma_B$, яка перевищує вагову оцінку слова A ($\sigma_{n+2} = \sigma_A$), а $C_7^m\left(m = n+5\right)$ – найближче праворуч від A_1^{n+2} слово, для якого $\sigma_{105} > \sigma_{102}$.

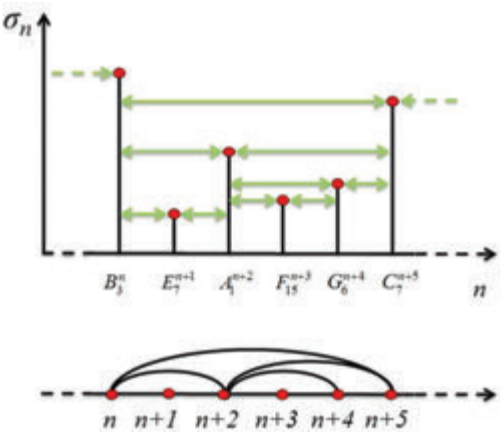


Рис. 11.3 – Приклад побудови графа горизонтальній видимості

На третьому, завершальному етапі, отримана на попередньому етапі мережа компактифікується. Всі вузли з вибраним словом, наприклад словом A, об'єднуються у один вузол. Всі зв'язки таких вузлів також об'єднуються. Важливо відзначити, що між будь-якими двома вузлами при

цьому залишається не більше ніж один зв'язок – кратні зв'язки вилучаються. Зокрема це означає, що міра (число зв'язків) вузла не перевищує суму ступенів $\sum_k A_k^n$.

У підсумку виходить нова мережа слів – компактифікованого графу горизонтальній видимості (КГГВ) – Рис. 11.4.

Для КГГВ-мереж слів було визначено розподіл ступенів вузлів, який виявився близьким до степеневого ($P(k) = Ck^{\alpha}$), тобто ці мережі є безмасштабнимі. Були проведені розрахунки параметрів мереж для багатьох текстових документів. В результаті виявилося, що для всіх з них коефіцієнт α змінювався в діапазоні від 0,95 до 1,05.

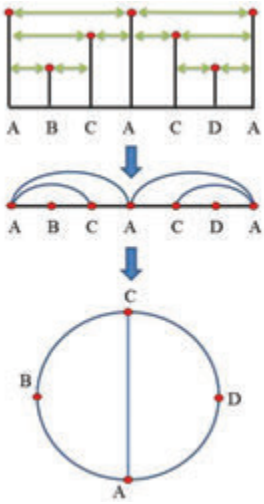


Рис. 11.4 – Етапи побудови компактифікованого графу горизонтальній видимості

До складу вузлів з найбільшими ступенями в КГГВ-мережі, разом з особистими займенниками та іншими службовими словами, потрапили слова, які визначають інформаційну структуру тексту.

Для порівняння було додатково досліджено поведінку простих мереж мови, коли на першому етапі побудови мережі зв'язуються сусідні слова,

які входять в текст (L-простір, R = 1), а на другому відбувається компактифікацією мережі. Очевидно, вага вузлів в цій мережі відповідає частоті появи слів, а їх розподіл – закону Ципфа. При цьому найбільші ступені мають вузли, які відповідають словами з найбільшою частотою, що мають велике значення для зв'язності тексту, але є малоцікавими з точки зору дослідження інформаційної структури.

Якщо позначити Ψ – множину з N різних слів (наприклад, N = 100), які відповідають найвагомішим вузлам наведеної простої мережі слів, а Λ – множина слів, які відповідають найвагомішим вузлів КГГВ, то множина $\Omega = \Lambda \setminus \Psi$ відповідає інформаційно значущим словами, які мають, крім того , важливе значення і для зв'язності тексту.

Як приклад зіставлялися 100 найбільш вагомих вузлів для трьох даних типів мереж слів за текстами Законів України "Про телекомунікації" та "Про захист персональних даних".

У КГГВ-мережі по тексту Закону України "Про телекомунікації" з урахуванням значень TF-IDF до складу множини потрапили такі слова, як «Державне», «Регулювання», «Ринок», «Інтернет», «Провайдер», «Трафік».

У КГГВ-мережі для цього ж тексту за ваговими значеннями слів, відповідних дисперсійним оцінками, додатково в множину потрапили такі слова, як «Суб'єкт», «Ресурс», «Переоформлення», «Рішення», «Споживачів» і ін.

При аналізі тексту Закону України "Про захист персональних даних" у множину (для КГГВ-мережі з урахуванням вагових значень слів за алгоритмом TFIDF) потрапили такі слова, як «Інформація», «Відстрочення», «Орган», «База», « Віключено».

У КГГВ-мережі для тексту цього законодавчого акту за ваговими значеннями слів, які відповідають дисперсійним оцінками, в множину потрапили додатково такі слова, як «Використання», «Права», «Уповноважений», «Особа».

11.8. Мережа природних ієрархій термінів

Для вирішення завдання побудови термінологічної онтології предметної області потрібне проведення комплексних досліджень, певним етапом яких є побудова так званих словникових номенклатур, предметних словників, тезаурусів. Ефективний автоматичний відбір окремих термінів для таких конструкцій – не вирішене остаточно завдання, а проблема встановлення зв'язків, автоматичної побудови мереж з таких термінів – до сих пір не розв’язана.

Як термінологічну основу для формування онтології пропонується використовувати мережу природної ієрархії термінів, яка базується на інформаційно-значущих елементах тексту. Опорні терміни, як правило, вибираються з урахуванням такої властивості, як дискримінантна сила. Однак однієї цієї властивості часто недостатньо для якісного відображення змісту предметної області. Іноді слова з низькою дискримінантної силою, наприклад, найбільш частотні слова з обраної предметної області (слова «Android», «IOS», «додаток» в корпусі текстів за тематикою сучасних гаджетів) виявляються найважливішими для даної задачі.

Як підхід до вирішення актуального завдання побудови термінологічної онтології розглядаються принципи і методика формування мережі природних ієрархій термінів (МПІТ), що базується на контенті текстів обраної спрямованості. "Природність" ієрархій термінів в цьому випадку розуміється як відмова при формуванні мережі від методів змістовного аналізу текстів, обмежуючись фактично статистичним аналізом. Зв'язки в такій мережі визначаються природним взаємним розміщенням слів і словосполучень з текстів. Така мережа, створювана повністю автоматично, може розглядатися в якості основи для подальшого автоматизованого формування термінологічної онтології (моделі предметної області) за участю експертів.

Методика формування мережі природних ієрархій термінів, передбачає реалізацію послідовності етапів, які розглянемо докладно:

1. На першому етапі формується вихідний текстовий корпус. Як приклад такого корпусу можуть розглядатися тексти науково-популярних статей, опублікованих на веб-сайті «Компьютерра онлайн» (http://www.computerra.ru), присвячених проблематиці мобільних пристроїв. До складу цього текстового корпусу було включено близько 230 статей загальним обсягом понад 800 тис. символів. Попередня обробка такого текстового корпусу передбачала виокремлення фрагментів текстів (окремих статей, абзаців, речень, слів), виключення нетекстових символів, вилучення флективних закінчень.

2. На другому етапі кожному окремому терміну з тексту (слову, біграмі або триграмі) ставиться у відповідність оцінка їх "дискримінантної сили", а саме TFIDF, яка в канонічному вигляді дорівнює добутку частоти відповідного терміна (Term Frequency) в фрагменті тексту на двійковий логарифм від величини, зворотної до кількості фрагментів тексту, в яких цей термін зустрівся (Inverse Document Frequency). Для послідовностей термінів і їх вагових значень будуються компактифіцировані графи горизонтальної видимості (КГГВ) і виконується перевизначення вагових значень слів уже за цим алгоритмом. Дана процедура дозволяє враховувати в подальшому крім термінів з великою дискримінантної силою також високочастотні терміни, які мають велике значення для загальної тематики. Мережа слів будується також в три етапи. На першому етапі на горизонтальній осі відзначається ряд вузлів, кожен з яких відповідає словами в порядку появи в тексті, а по вертикальній осі відкладаються вагові чисельні оцінки (TFIDF). На другому етапі будується традиційний граф горизонтальній видимості. Вважається, що між вузлами існує зв'язок, якщо вони знаходяться в «прямої видимості», тобто якщо їх можна з'єднати горизонтальною лінією, що не перетинає жодну іншу вертикальну лінію. На третьому, заключному етапі, мережа компактифікується. Всі

вузли з однаковим словом об'єднуються в один вузол, зв'язки таких вузлів також об'єднуються. У якості вагових оцінок окремих слів в подальшому використовуються ступені відповідних їм вузлів в КГГВ. Після цього всі терміни тексту упорядковуються відповідно до зменшення розрахованих вагових значень вузлів КГГВ. Подальшому аналізу не підлягають терміни з так званого стоп-словника, що є важливими для зв'язності тексту, але не несуть інформаційного навантаження. Це, як правило, фіксований набір службових слів. Експертним шляхом визначається необхідний розмір МПІТ (число N), після чого вибирається відповідна кількість одиничних слів, біграм і триграм (всього $N + N + N$ елементів) з найбільшими ваговими значеннями по КГГВ.

3. На третьому етапі з відібраних термінів будуються мережі природних ієрархій термінів, в яких як вузли розглядаються самі терміни, а зв'язку відповідають входженням одних термінів в інші. На Рис. 11.5 проілюстрований принцип побудови зв'язків МПІТ. Різні геометричні фігури на цій ілюстрації відповідають різним словами. Першому рядку відповідає множина одиничних слів, другий – множина біграм, а третій – множина триграм. Якщо одиничне слово входить в біграму або триграму, або біграма входить у триграму, утворюється зв'язок, який позначається стрілкою. Множина вузлів, яким відповідають терміни, і зв'язків утворюють трирівневу мережу природної ієрархії термінів.

Після формування МПІТ (побудови матриці інцідентності) здійснюється її відображення програмними засобами аналізу і візуалізації графів. Для завантаження мереж природних ієрархій термінів в бази даних формується матриця інцідентності загальноприйнятого формату csv розмірністю $(N + N + N) \times (N + N + N)$ елементів.

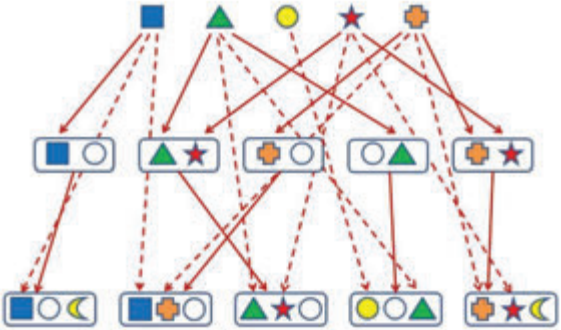


Рис. 11.5 – Трирівнева мережа природних ієрархій термінів

На Рис. 11.6 представлена невелика мережа природної ієрархії термінів розміром 30 + 30 + 30, яка візуалізувати засобами системи Gephi (https://gephi.org/).

Рис. 11.6 – Візуалізація зв‘язаного фрагменту МПІТ розміром 30+30+30

На рис. 11.7 наведені окремі фрагменти більшої мережі природної ієрархії термінів розміром 200 + 200 + 200.

11.9. Онтології

Термін «онтологія» вживається в кількох областях знань і має два різних значення:

- філософська дисципліна, яка вивчає найбільш загальні характеристики буття і сутностей;
- в інженерії знань: артефакт, структура, яка описує значення елементів деякої системи.

Таким чином, чіткої взаємної обумовленості між значеннями терміна «онтологія» в філософії і в інженерії знань не простежується. Зв'язок між ними носить скоріше асоціативний характер.

Рис. 11.7 – Фрагмент МПІТ розміром 200+200+200

Під онтологією тут ми будемо розуміти систему понять предметної області (модель предметної області), яка представляється як набір сутностей, які об'єднані різними відносинами. Онтології набули широкого поширення в задачах представлення знань, семантичної інтеграції інформаційних ресурсів, інформаційного пошуку і т.п. У науці про штучний інтелект онтологія – це "специфікація концептуалізації предметної області», або спрощено, документ або файл, який формально задає зв'язки між поняттями. Це свого роду словник понять предметної

області і сукупність явно певних припущень щодо змісту цих понять. Найчастіше онтологія представляється як ієрархія понять, пов'язаних відносинами певних видів. Розвинена онтологія формалізується засобами логіки і допускає можливості формування логічних тверджень.

Онтології використовуються для формальної специфікації понять і відносин, які характеризують певну область знань. Перевагою онтології як способу представлення знань є їх формальна структура, яка спрощує комп'ютерну обробку.

Терміну "онтологія" задовольняє широкий спектр структур, які представляють знання про ту чи іншу предметну область. Так до онтології можна віднести ряд структур, які відрізняються різним рівнем формалізації:

- глосарій;
- проста таксономія;
- тезаурус (таксономія з термінами);
- понятійна структура з довільним набором відносин;
- повністю аксіоматізуєма теорія.

У загальному вигляді структура онтології являє собою набір елементів чотирьох категорій:

- поняття;
- відношення;
- аксіоми;
- окремі екземпляри.

Дослідники виділяють прикладну онтологію, онтологію області знання, загальну і репрезентаційну онтологію (йдеться про онтологію метарівня, яка включає в себе репрезентаційні першоелементи).

Онтології можуть бути також розділені на одномовні і багатомовні. Вже існує ряд онтологій, орієнтованих на представлення знань декількома мовами, наприклад, EuroWordNet, MikroKosmos і деякі інші. Також виділяється особливий тип онтології – лексична онтологія (або

лінгвістична). Відмітною властивістю такої онтології є фіксація в одному ресурсі понять (слів) разом з їх мовними властивостями. Така онтологія тісно взаємопов'язана з семантикою граматичних елементів (слів, іменних груп і ін.) Основними джерелами понять в онтології даного типу є значення мовних одиниць. Їх також відрізняє своєрідний набір відносин, зазвичай властивий для мовних елементів: синонімія, гіпонімія, меронімія, а також ряд інших. Коло завдань, що вирішуються такою онтологією, тісно взаємопов'язано з обробкою природної мови.

Онтологію, зокрема, можна ефективно використовувати для підвищення точності інформаційного пошуку – пошукова система буде видавати тільки такі документи, де потрібне поняття відтворюється точно, а не ті, в текстах яких зустрілося задане ключове слово.

Мова опису онтологій — формальна мова, використовувана для кодування онтології. Існує декілька подібних мов, наприклад:

- OWL — Ontology Web Language, стандарт W3C, мова для семантичних тверджень, розроблена як розширення RDF і RDFS;
- KIF (Knowledge Interchange Format або формат обміну знаннями) — заснований на S-виразах синтаксис для логіки;
- Common Logic — спадкоємець KIF (стандартизований — ISO/IEC 24707:2007).
- CycL — онтологічна мова, що використовується в проекті Cyc, заснована на численні предикатів із деякими розширеннями вищого порядку.
- DAML+OIL (FIPA)

Для роботи з мовами онтологій існує декілька видів технологій: редактори онтологій (для створення онтологій), DBMS онтологій (для зберігання й звертання до онтології) і сховища онтологій (для роботи з декількома онтологіями).

Мова опису онтологій для семантичного веб

OWL (Web Ontology Language) – це мова опису онтологій для семантичної павутини. Мова OWL дозволяє описувати класи і відносини між ними, властиві для веб-документів і додатків. OWL заснований на більш ранніх мовах OIL і DAML OIL і є рекомендованим консорціумом W3C. В основі мови — уявлення дійсності в моделі даних «об'єкт — властивість». OWL придатна для опису не тільки веб-сторінок, але і будь-яких об'єктів дійсності. Кожному елементу опису в цій мові (в тому числі властивостями, що зв'язує об'єкти) ставиться у відповідність URI (Uniform Resource Identifier), що представляє собою уніфікований ідентифікатор ресурсів в Інтернеті — компактний рядок літер, який однозначно ідентифікує окремий абстрактний чи фізичний ресурс. Основне призначення таких ідентифікаторів — зробити можливою взаємодію з представленнями ресурсів через мережу, використовуючи спеціальні протоколи. URI визначається схемами, які визначають синтаксис та відповідні протоколи. Найбільш поширеною формою URI є уніфікований локатор ресурсів (URL), який ще називають веб-адресою..

OWL має три діалекти (у порядку зростання виразності):

- OWL Lite призначена для користувачів, які потребують передусім класифікаційної ієрархії і простих обмежень. Наприклад, при тому, що вона підтримує обмеження кардинальності (кількості елементів), допускаються значення кардинальності тільки 0 або 1. Для розробників повинно бути простіше в своїх продуктах забезпечити підтримку OWL Lite, чим виразніших варіантів OWL. Зокрема, OWL Lite дозволяє швидко перенести існуючі тезауруси і інші таксономії. OWL Lite має нижчу формальну складність, ніж OWL DL.

- OWL DL призначена для користувачів, яким потрібна максимальна виразність при збереженні повноти обчислень (всі логічні висновки, що припускаються тією чи іншою онтологією, будуть гарантовано обчислюваними) і розв'язуваності (всі обчислення завершаться за певний час). OWL DL включає всі мовні конструкції OWL, але вони можуть використовуватися лише згідно з певним обмеженням (наприклад, клас може бути підкласом багатьох класів, але не може сам бути представником іншого класу). OWL DL так названий через його відповідності дескрипційній логіці — дисципліні, в якій розроблені логіки, що складають формальну основу OWL.
- OWL Full призначена для користувачів, яким потрібна максимальна виразність і синтаксична свобода RDF без гарантій обчислення. Наприклад, у OWL Full клас може розглядатися одночасно як сукупність індивідів і як один індивід у своєму власному значенні. OWL Full дозволяє будувати такі онтології, які розширюють склад зумовленого (RDF або OWL) словника.

У Стенфордському університеті (США) розроблена програмна платформа – редактор онтології Protégé (http://protege.stanford.edu), а також організовано співдружність ентузіастів, яке налічує кілька тисяч учасників, які поповнюють базу онтології для самих різних предметних областей. Заслуговує також уваги проект Estrella (www.estrellaproject.org), в рамках якого розроблена онтологія LKIF (Legal Knowledge Interchange Format) – мова для подання юридичних знань і обміну між правовими інформаційними базами.

Питання для самоперевірки

1. В чому полягає закон Ципфа?
2. Яким чином можна обчислити ваги слів у тексті?
3. Які основні елементи Text Mining?

4. Що таке мережі мови?
5. Для чого застосовуються графи горизонтальної видимості?

Література до розділу

1. *Ланде Д.В.* Елементи комп'ютерної лінгвістики в правовій інформатиці. – К.: НДІІП НАПрН України, 2014. – 168 с. **2.** Автоматическая обработка текстов на естественном языке и компьютерная лингвистика: учеб. пособие / *Большакова Е.И., Клышинский Э.С., Ландэ Д.В., Носков А.А., Пескова О.В., Ягунова Е.В.* – М.: МИЭМ, 2011. – 272 с. **3.** *Головач Ю., Пальчиков В.* Лис Микита і мережі мови, Журн. Фіз. Досл., 2006. – № 10. – С. 247-291. **4.** *Ландэ Д.В.* Моделирование контентных сетей // Проблеми інформатизаціх та управління: Збірник наукових праць: Випуск 1(37). – К.: НАУ, 2012. – С.78–84. **5.** *Герасимов Б.М., Сергеєв О.Ю., Субач І.Ю.* Оцінка релевантності знайдених документів у розподілених інформаційно-пошукових системах // Автоматизація виробничих процесів, 2005. - №1(20). – С.49-57. **6.** Герасимов Б.М., Нікіфоров Є.В., Субач І.Ю. Моделі надання знань для використання в системах підтримки прийняття рішень // Науково-технічна інформація, 2005. – 1. – С.7–11.

ДОДАТОК. КОРОТКІ ВІДОМОСТІ ЩОДО СИСТЕМИ MATLAB

Технологічним середовищем проведення практичних занять є система Matlab (MATrix LABoratoty), що призначена для виконання наукових та інженерних розрахунків, з її допомогою вирішуються завдання обчислювальної математики, математичної статистики, методів оптимізації та ін. До складу Matlab входить кілька десятків спеціалізованих пакетів – тулбоксів (Toolbox), призначених для обробки даних у різних галузях науки і техніки (методи оптимізації, статистика, нейронні мережі, цифрова обробка сигналів і т.п.).

Система MATLAB орієнтована на роботу з масивами даних, яка застосовується в таких областях, як теорія управління, цифрова обробка сигналів і зображень, дослідження операцій тощо. Наведена нижче інформація є необхідною для початку роботи в системі.

Оператори

Оператор – це спеціальне позначення для певної операції над даними – операндами. Найпростішими арифметичними операторами є знаки суми +, віднімання –, множення * і ділення /. Оператори використовуються разом з операндами. Наприклад, у виразі 2 + 3 знак + є оператором складання, а числа 2 і 3 – операндами. Більшість операторів у Matlab відноситься до матричних операцій. Наприклад, оператори множення * і ділення / обчислюють добуток і частку від ділення двох багатовимірних масивів, векторів або матриць. Існує ряд спеціальних операторів, наприклад, оператор \, який означає ділення справа наліво, а оператори .* і ./ означають, відповідно, поелементне множення і поелементне ділення масивів.

Оператори системи Matlab діляться на три категорії:

- арифметичні оператори, що дозволяють конструювати арифметичні вирази і виконувати числові обчислення.

- оператори відносин, що дозволяють порівнювати числові операнди.
- логічні оператори, що дозволяють будувати логічні вирази.

Арифметичні оператори:

+	– додавання
-	– віднімання
*	– множення матриць
.*	– поелементне множення масивів
^	– зведення матриці до степеня
.^	– поелементне зведення до степеня
\	– ліве ділення матриць
/	– праве ділення матриць
.\	– ліве ділення масивів
./	– праве ділення масивів

При роботі з масивом чисел встановлені такі приорітети операцій:

рівень 1:	(.^), (^)
рівень 2:	(+), (-)
рівень 3:	(.*), (./), (.\), (*), (/), (\)
рівень 4:	(+), (-)
рівень 5:	(:)

Усередині кожного рівня оператори мають рівний приорітет і обчислюються в порядку дотримання зліва направо. Заданий за умовчанням порядок приорітетів може бути змінений за допомогою круглих дужок.

Арифметичні оператори системи Matlab працюють, як правило, з масивами однакових розмірів. Для векторів і прямокутних масивів обидва операнди мають бути однакового розміру, за винятком єдиного випадку, коли один з них – скаляр. Якщо один з операндів скалярний, то в системі

Matlab він розширюється до розмірів другого операнда і задана операція застосовується до кожного елементу. Така операція називається розширенням скаляра.

Оператори відношення

У системі Matlab визначені оператори відношення :

- < менше;
- <= менше або рівно;
- > більше;
- > = більше або рівно;
- == рівно тотожно;
- ~ = не рівно.

Оператори відношення виконують поелементне порівняння двох масивів рівної розмірності. Для векторів і прямокутних масивів, обидва операнди мають бути однакового розміру, за винятком випадку коли один з них скаляр. В цьому випадку Matlab порівнює скаляр з кожним елементом іншого операнда. Позиції, де це співвідношення істинне, набувають значення 1, де помилкове – 0.

Логічні оператори

До складу логічних операторів системи Matlab входять наступні оператори:

&	І
|	Або
~	Ні

Логічні оператори реалізують поелементне порівняння масивів однакової розмірності. Для векторів і прямокутних масивів обидва операнди мають бути однакового розміру, за винятком випадку, коли один з них скаляр. У останньому випадку Matlab порівнює скаляр з кожним елементом іншого операнда. Позиції, де це співвідношення істинне, набувають значення 1, де помилкове – 0.

Функції

Функції – це унікальні імена об'єктів, які виконують певні перетворення своїх аргументів і повертають результати цих перетворень. При цьому результат обчислення функції з одним вихідним параметром підставляється на місце її виклику, що дозволяє використовувати функції в математичних виразах, наприклад функцію sin в 2*sin(pi/2). Функції в загальному випадку мають список аргументів (параметрів), що поміщаються в круглі дужки.

Матриці

- ZEROS – формування масиву нулів
- ONES – формування масиву одиниць
- EYE – формування одиничної матриці
- RAND – формування масиву елементів, розподілених за рівномірним законом
- MESHGRID – формування вузлів двовимірної і тривимірної сіток
- ":" – формування векторів і підматриць

ZEROS – формування масива нулів

Y = zeros(n)

Y = zeros(m, n)

Y = zeros(size(A))

Функція Y = zeros(n) формує масив нулів розміру n × n

Функція Y = zeros(m, n) формує масив нулів розміру m × n

Функція Y = zeros(size(A)) формує масив нулів співвимірний з масивом A

ONES – формування масива одиниць

Y = ones(n)

Y = ones(m, n)

Y = ones(size(A))

Функція Y = ones(n) формує масив одиниць розміру n × n

Функція Y = ones(m, n) формує масив одиниць розміру m × n

Функція Y = ones(size(A)) формує масив одиниць співвимірний з масивом A

EYE – формування одиничної матриці

Y = eye(n)

Y = eye(m, n)

Y = eye(size(A))

Функція Y = ones(n) формує одиничну матрицю розміру n × n

Функція Y = ones(m, n) формує одиничну матрицю розміру m × n

Функція Y = ones(size(A)) формує одиничну матрицю співвимірну з матрицею A.

RAND – формування масива елементів, розподілених за рівномірним законом

X = rand(n)	rand
X = rand(m, n)	rand('seed')
X = rand(size(A))	rand('seed', x0)

Функція X = rand(n) формує масив розміру n × n, елементами якого є випадкові величини, розподілені за рівномірним законом в інтервалі (0, 1).

Функція X = rand(m, n) формує масив розміру m × n, елементами якого є випадкові величини, розподілені за рівномірним законом в інтервалі (0, 1).

Функція X = rand(size(A)) формує масив співвимірний з матрицею A, елементами якого є випадкові величини, розподілені за рівномірним законом в інтервалі (0, 1).

Функція rand без аргументів формує одне випадкове число, що підкоряється рівномірному закону розподілу в інтервалі (0, 1), який змінюється при кожному наступному виклику.

Функція rand('seed') повертає поточне значення бази (початкового значення) генератора випадкових чисел.

Функція rand('seed', x0) привласнює базі (початковому значенню) генератора випадкових чисел значення x0.

Алгоритм генерації рівномірно розподілених випадкових чисел заснований на лінійному конгруентном методі. Обчислення наступного випадкового числа реалізоване згідно із співвідношенням:

$$seed = (7^7 \text{ x } seed) \ (\text{mod}(2^{31} - 1).$$

MESHGRID – формування вузлів двовимірної і тривимірної сіток

[X, Y] = meshgrid(x, y)

[X, Y] = meshgrid(x)

[X, Y, Z] = meshgrid(x, y, z)

Функція [X, Y] = meshgrid(x, y) формує масиви X і Y, які визначають координати вузлів прямокутника, що задається векторами x і y. Цей прямокутник задає область визначення функції від двох змінних, яку можна побудувати у вигляді 3D-поверхности.

Функція [X, Y] = meshgrid(x) є скороченою формою запису функції [X, Y] = meshgrid(x, x).

Функція [X, Y, Z] = meshgrid(x, y, z) формує масиви X, Y і Z, які визначають координати вузлів паралелепіпеда, що задається векторами x, y і z.

«:» – формування векторів і підматриць

j : k	A(i1 : i2, j1 : j2)
j : i : k	A(n1 : n2)

Оператор ":" застосовується для формування векторів і матриць або для виділення з них підвекторів, підматриць, підблоків масиву.

Виделення підблоків

A(i1: i2, j1: j2) - виділення підблоку масиву A з рядками i1 : i2 і стовпцями j1 : j2.

A(i,:) - i -й рядок масиву A;

A(:, j) - j -й стовпець масиву A.

Оскільки в мові Matlab елементи масиву впорядковані по стовпцях, то допустимі оператори вигляду $A(n_1: n_2)$, які виділяють пронумеровані елементи з номера n_1 до номера n_2. Оператор A(:) записує усі елементи масиву A у вигляді стовпця.

Основні операції над масивами

- SUM, CUMSUM – підсумовування елементів масиву
- PROD, CUMPROD – добуток елементів масиву
- SORT – сортування елементів масиву за збільшенням
- MAX – визначення максимальних елементів масиву
- MIN – визначення мінімальних елементів масиву
- MEAN – визначення середніх значень елементів масиву
- STD – визначення стандартних відхилень елементів масиву

SUM, CUMSUM – сумування елементів масиву

sx = sum(X)

csx = cumsum(X)

Функція sx = sum(X) у разі одновимірного масиву повертає суму елементів масиву; у разі двовимірного масиву – це вектор-рядок, що містить суми елементів кожного стовпця.

Функція csx = cumsum(X), крім того, повертає усі проміжні результати підсумовування.

PROD, CUMPROD – добуток елементів масиву

px = prod(X)

cpx = cumprod(X)

Функція px = prod(X) у разі одновимірного масиву повертає добуток елементів масиву; у разі двовимірного масиву - це вектор-рядок, що містить добутки елементів кожного стовпця.

Функція cpx = cumprod(X), крім того, повертає усі проміжні результати.

SORT – сортування елементів масиву за зростанням

Y = sort(X)

[Y, I] = sort(X)

Функція Y = sort(X) у разі одновимірного масиву упорядковує елементи масиву за збільшенням; у разі двовимірного масиву відбувається впорядкування елементів кожного стовпця.

MAX – визначення максимальних елементів масиву

Y = max(X)

[Y, I] = max(X)

C = max(A, B)

Функція Y= max(X) у разі одновимірного масиву повертає найбільший елемент; у разі двовимірного масиву – вектор-рядок, що містить максимальні елементи кожного стовпця. Таким чином, max(max(X)) – це найбільший елемент масиву.

Функція [Y, I] = max(X) окрім самих максимальних елементів повертає вектор-рядок індексів цих елементів в цьому стовпці.

Функція C = max(A, B) повертає масив C тих же розмірів, які мають масиви A і B, кожен елемент якого максимальний з відповідних елементів цих масивів.

MIN – визначення мінімальних елементів масиву

Y = min(X)

[Y, I] = min(X)

C = min(A, B)

Функція Y = min(X) у разі одновимірного масиву повертає найменший елемент; у разі двовимірного масиву – вектор-рядок, що містить мінімальні елементи кожного стовпця. Таким чином, min(min(X)) – це найменший елемент масиву.

Функція [Y, I] = min(X) окрім самих мінімальних елементів повертає вектор-рядок індексів цих елементів в цьому стовпці.

Функція C = min(A, B) повертає масив C тих же розмірів, які мають масиви A і B, кожен елемент якого мінімальний з відповідних елементів цих масивів.

MEAN – визначення середніх значень елементів масиву

mx = mean(X)

Функція mx = mean(X) у разі одновимірного масиву повертає середнє арифметичне елементів масиву; у разі двовимірного масиву – вектор-рядок, що містить середнє арифметичне елементів кожного стовпця. Таким чином, mean(mean(X)) – це арифметичне середнє елементів масиву, що співпадає зі значенням mean(X(:)).

STD – визначення стандартних відхилень елементів масиву

sx = std(X)

Функція sx = std(X) у разі одновимірного масиву повертає стандартне відхилення елементів масиву; у разі двовимірного масиву – вектор-рядок, що містить стандартне відхилення елементів кожного стовпця.

Математичні функції

У системі Matlab існує велика бібліотека математичних функцій. Аргументи функцій завжди вказуються в круглих дужках після імені функції і, якщо їх більше одного, розділяються комами. Розглянемо вбудовані математичні функції системи Matlab, які застосовуються до чисел, скалярних змінних і до масивів.

- ABS – абсолютне значення
- SIGN – обчислення знаку числа
- CEIL, FIX, FLOOR, ROUND – функції округлення
- SQRT – квадратний корінь
- EXP – експоненціальна функція
- LOG – функція натурального логарифма
- SIN – функція синуса
- COS – функція косинуса
- TAN – функція тангенса

ABS – абсолютне значення

Y = abs(X)

Для масиву дійсних чисел X функція Y = abs(X) повертає масив Y абсолютних значень елементів X.

SIGN – обчислення знаку числа

S = sign(Z)

Для масивів дійсних чисел X функція S = sign(X) повертає масив S тих самих розмірів, в якому на місці позитивного числа стоїть 1, на місці нульового – 0, на місці негативного – (–1).

CEIL, FIX, FLOOR, ROUND – функції округлення

Y = ceil(X)

Y = fix(X)

Y = floor(X)

Y = round(X)

Для масивів дійсних чисел X :

- функція Y = ceil(X) повертає значення, округлені до найближчого цілого >=X;
- функція Y = fix(X) повертає значення з усіканням дробової частини числа;
- функція Y = floor(X) повертає значення, округлені до найближчого цілого<= X;
- функція Y = round(X) повертає значення, округлені до найближчого цілого.

SQRT – квадратний корінь

V = sqrt(Z)

Функція V = sqrt(Z) обчислює квадратне коріння елементів масиву Z. Для негативних і комплексних значень результат є комплексним числом.

EXP – експоненціальна функція

V = exp(Z)

Функція V = exp(Z) обчислює експоненти значень елементів масиву Z.

LOG – функція натурального логарифма

V = log(Z)

Функція V = log(Z) обчислює натуральний логарифм значень елементів масиву Z.

SIN – функція синусу

V= sin(Z)

Функція V = sin(Z) обчислює синус від значень елементів масиву Z.

COS – функція косинусу

V = cos(Z)

Функція V = cos(Z) обчислює косинус від значень елементів масиву Z.

TAN – функция тангенсу

V = tan(Z)

Функція V = tan(Z) обчислює тангенс від значень елементів масиву Z.

Графічні команди і функції

До складу системи Matlab входить потужна графічна підсистема, яка підтримує як засоби візуалізації двовимірної і тривимірної графіки на екрані терміналу, так і засоби презентаційної графіки.

Основні графічні функції:

- PLOT – графік в лінійному масштабі
- LOGLOG – графік в логарифмічному масштабі
- PLOT3 – побудова ліній і точок в тривимірному просторі
- MESH, MESHC, MESHZ – тривимірна сітчаста поверхня
- SURF, SURFC – затінена сітчаста поверхня
- BAR – стовпчикові діаграми
- HIST – побудова гістограми

PLOT – графік у лінійному масштабі

plot(Y)

plot(X, Y)

plot(X, Y, S)

plot(X_1, Y_1, S_1, X_2, Y_2, S_2, ...)

Команда plot(Y) будує графік елементів одновимірного масиву Y залежно від номера елемента; якщо елементи масиву у комплексні, то

будується графік plot(real(Y), imag(Y)). Якщо Y – двовимірний дійсний масив, то будуються графіки для стовпців; у разі комплексних елементів їх уявні частини ігноруються.

Команда plot(X, Y) відповідає побудові звичайної функції, коли одновимірний масив x відповідає значенням аргументу, а одновимірний масив Y – значенням функції. Якщо один з масивів X або Y, або обидва двовимірні, реалізуються наступні побудови:

- якщо масив Y двовимірний, а масив X одновимірний, то будуються графіки для стовпців масиву Y залежно від елементів вектора X;
- якщо двовимірним є масив X, а масив Y одновимірний, то будуються графіки стовпців масиву X залежно від елементів вектора Y;
- якщо обидва масиви X і Y двовимірні, то будуються залежності стовпців масиву Y від стовпців масиву X.

Команда plot(x, y, s) дозволяє виділити графік функції, вказавши спосіб відображення лінії, точок, колір ліній і точок за допомогою строкової змінної s, яка може включати до трьох символів з наступної таблиці:

Тип лінії		**Тип точки**		**Колір**	
Неперервна	-	Точка	.	Жовтий	y
Штрихова	--	Плюс	+	Фіолетовий	m
Подвійний пунктир	:	Зірочка	*	Блакитний	c
Штрих-пунктир	-.	Коло	o	Червоний	r
		Хрестик	x	Зелений	g
				Синій	b
				Білий	w
				Чорний	k

Якщо колір лінії не вказаний, він вибирається за умовчанням з шести перших кольорів, з жовтого до синього, повторюючись циклічно.

Команда plot(x_1, y_1, s_1, x_2, y_2, s_2, ...) дозволяє об'єднати на одному графіку декілька функцій $y_1(x_1)$, $y_2(x_2)$, …, визначивши для кожної з них свій спосіб відображення.

Звернення до команд plot вигляду plot(x, y, s1, x, y, s2) дозволяє для графіка y(x) визначити додаткові властивості, для визначення яких застосування однієї строкової змінної s1 недостатньо, наприклад, при завданні різних кольорів для лінії і для точок на ній.

LOGLOG – графік у логарифмічному масштабі

loglog(x, y)

loglog(x, y, s)

loglog(x1, y1, s1, x2, y2, s2, ...)

Команди loglog(...) рівносильні функціям plot, за винятком того, що вони використовують по обох осях логарифмічний масштаб замість лінійного.

PLOT3 – побудова ліній і точок у тривимірному просторі

plot3(x, y, z)

plot3(X, Y, Z)

plot3(x, y, z, s)

plot3(x1, y1, z1, s1, x2, y2, z2, s2, ...)

Команди plot3(...) є тривимірними аналогами функції plot(…).

Команда plot3(x, y, z), де x, y, z – одновимірні масиви однакового розміру, будує точки з координатами x(i), y(i), z(i) і сполучає їх прямими лініями.

Команда plot3(X, Y, Z), де X, Y, Z – двовимірні масиви однакового розміру, будує точки з координатами x(i,:), y(i,:), z(i,:) для кожного стовпця і сполучає їх прямими лініями.

Команда plot3(x, y, z, s) дозволяє виділити графік функції z(x, y), вказавши спосіб відображення лінії, спосіб відображення точок, колір ліній і точок за допомогою строкової змінної s, яка може включати до трьох символів з тієї ж таблиці, що і для функції plot.

Команда plot3(x1, y1, z1, s1, x2, y2, z2, s2, ..) дозволяє об'єднати на одному графіку декілька функцій z1(x1, y1), z2(x2, y2), ..., визначивши для кожної з них свій спосіб відображення.

MESH, MESHC, MESHZ – сітчата поверхня у тривимірному просторі

mesh(X, Y, Z)	meshc(X, Y, Z)	meshz(X, Y, Z)
mesh(Z)	meshc(Z)	meshz(Z)

Команда mesh(X, Y, Z) дозволяє виводити на екран сітчасту поверхню для значень масиву Z, визначених на множині значень масивів X і Y. Колір вузлів пропорційний висоті поверхні.

Група команд meshc(...) додатково до тривимірних поверхонь будує проекцію ліній постійного рівня.

Група команд meshz(..) додатково до тривимірних поверхонь будує площину відліку на нульовому рівні, закриваючи поверхню, що лежить нижче за цей рівень.

SURF, SURFC – затінена сітчаста поверхня

surf(Z,C)	surfc(Z,C)
surf(X, Y, Z)	surfc(X, Y, Z)
surf(Z)	surfc(Z)

Команда surf(Z, C) використовує сітку, яка визначається одновимірними масивами x = 1, : n і y = 1 : m.

Команди surf(X, Y, Z), surf(x, y, Z), surf(Z) використовують як масив кольори C = Z, тобто колір в цьому випадку пропорційний висоті поверхні.

Група команд surfc(...) додатково до тривимірних затінених поверхонь будує проекцію ліній постійного рівня.

BAR – стовпчикові діаграми

bar(Y); bar(Y, '<тип лінії>')

bar(X,Y); bar(X,Y, '<тип лінії>')

Команда bar(Y) виводить графік елементів одновимірного масиву Y у вигляді столбцовой діаграми.

Команда bar(X, Y) виводить графік елементів масиву Y у вигляді стовпців в позиціях, визначуваних масивом X, елементи якого мають бути впорядковані в порядку зростання.

Якщо X і Y – двовимірні масиви однакових розмірів, то кожна діаграма визначається відповідною парою стовпців і вони надбудовуються одна над іншою.

Команди bar(Y, '<тип лінії>'), bar(X, Y, '<тип лінії>') дозволяють задати тип ліній, використовуваних для побудови стовпчикових діаграм, по аналогії з командою plot.

HIST – побудова гістограми

hist(y)

hist(y, n)

hist(y, x)

Команди hist(...) підраховують і відображують на графіці кількість елементів масиву y, значення яких потрапляють в заданий інтервал; для цього увесь діапазон значень y поділяється на n інтервалів (за умовчанням 10) і підраховується кількість елементів в кожному інтервалі.

Команда hist(y) виводить гістограму для 10 інтервалів.

Команда hist(y, n) виводить гістограму для n інтервалів.

Команда hist(y, x) виводить гістограму з урахуванням діапазону зміни змінної x.

Завантаження системи Matlab і робота у головному вікні

Завантажити Matlab: кнопка **Пуск\Програми\MATLAB**.

Обчислення можна проводити прямо в командному вікні. Для цього у головному вікні після символа >> набираються відповідні команди. Результат буде збережено у змінній *ans*, що створюється автоматично. При виконанні великої кількості команд (наприклад, при реалізації деякогоо

алгоритму) можна оформити обчислення у вигляді програм (скриптів) – М-файлів.

Створення М-файлів

М-файли містять послідовності команд. Фактично у М-файлі міститься код програми користувача, який зручно редагувати і доповнювати. Для створення М-файла слід зайти в меню File\New\M-file. В результаті відкривається текстовий редактор Matlab, призначений для створення і редагування цих файлів. Ім'я файлу не повинне починатися з цифри, містити пропуски.

Використання довідникової системи

Matlab містить велику кількість вбудованих функцій. Для того, щоб користуватися вбудованою функцією необхідно знати, які параметри і в якій послідовності слід їй передавати. Довідку щодо вбудованої функції можна отримати набравши в командному вікні команду help і далі ім'я функції, що використовується, наприклад, help sin. Виклик розширеної довідки здійснюється з головного меню Matlab: Help\Product help (або натиснути клавішу F1). Пошук потрібної функції здійснюється з рядку пошуку по деякому ключовому слову.

Робота з векторами і матрицями

Оголошення векторів і матриць може здійснюватися безпосередньо в середовищі Matlab. Нижче наведено приклади.

Створення вектора:

```
>> V=[1 2 3 4 5];
```

Створення матриці:

```
>> M=[1 2 3 4 5; 10 20 30 40 50; 100 200 300 400 500]
```

Якщо після оголошення вектора (або матриці) поставити крапку з комою (;), то результат V (або M) буде обчислений, але значення змінних виводиться в командне вікно не будуть.

Доступ до окремого елементу вектора здійснюється шляхом вказівки індексу в круглих дужках, наприклад: V(3) – третій елемент вектора V.

Доступ до окремого елементу матриці також здійснюється шляхом вказівки індексів, розділених комою, наприклад: M(2,3) – елемент матриці з другого рядка і третього стовпця матриці M.

Доступ до стовпця матриці здійснюється за допомогою використання символу ":" поряд з індексом, наприклад, M(:, 3) – доступ до третього стовпця матриці M, M(2,:) – доступ до усього другого рядка матриці M.

Додавання стовпців до матриці здійснюється за допомогою спеціального символу ";". Наприклад, додавання рядка S=[1 2 3 4 5] до матриці M=[1 2 3 4 5; 1 2 3 4 5; 1 2 3 4 5] виконується за допомогою команди:

```
>> M=[M;S]
```

Для додавання стовпця до матриці використовується кома замість крапки з комою в останньому операторові, наприклад:

```
>> M=[1 2 3 4 5; 1 2 3 4 5; 1 2 3 4 5];
>> S=[1; 2; 3];
>> M=[M,S]
```

Для видалення стовпців використовується послідовність символів "[ ]", наприклад для видалення усього третього стовпця матриці M вводиться команда:

```
>> M(:,3)=[]
```

Видалення усього другого рядка матриці M:

```
>> M(2,:)=[]
```

Транспонування вектора і матриці здійснюється допомогою операції, визначуваної символом "’", наприклад:

```
>> b=[1 2 3 4];
>> s=b’
>> s
1
2
3
4
```

Арифметичні операції з векторами і матрицями

За умовчанням усі дії: +, –, *, /, ^ (піднесення до ступеня) виконуються над матрицями за правилами лінійної алгебри.

Для поелементного перемножування двох векторів необхідно використовувати оператор точка «.»:

```
>> d=a.*b;
```

У результаті перемножування вектора на вектор за правилами лінійної алгебри отримуємо число (другий вектор при цьому потрібно транспонувати):

```
>>a=[2 3 4 5];
>>b=[1 2 3 4];
>>c=a*(b')
```

Створення строкової змінної здійснюється з використанням одинарних лапок. Створення масиву строкових змінних здійснюється за допомогою масиву елементів, наприклад:

```
>> D = {'ein','zwei','drei'}
```

Робота з графікою

Основна команда для роботи двомірною графікою – plot.

Наведемо приклад програми, що реалізовує побудову графіка $f(x) = x\sin(x)$ (рис. Д1).

Для цього оголосимо вектор аргумент, наприклад:

```
>> x=0:0.1:20
```

Тут елементи вектора x набуватимуть значень від 0 до 28 з кроком 0.1.

Після цього обчислюються значення цільової функції:

```
>> y=x.*sin(x)
```

Безпосередня побудова графіка:

```
>> plot(x,y)
```

Відображення сітки:

```
>> grid on
```

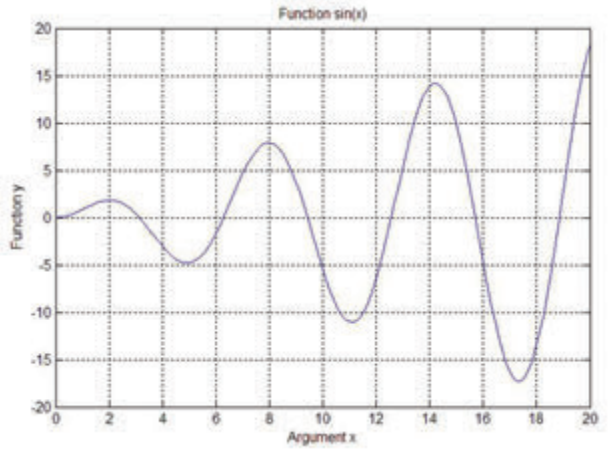


Рис. Д.1 – Графік функції $f(x) = x\sin(x)$

Заголовок графіка:

```
>> title ('Function sin(x)')
```

Підпис по осі x:

```
>> xlabel('Argument x')
```

Підпис по осі y:

```
>> ylabel('Function y')
```

Розглянемо побудову тривимірних графіків функцій на прикладі функції $f(x, y) = x^2 - y^2 + xy + 2$.

Визначення лівої межі для x:

```
>> Lx=-5;
```

Визначення правої межі для x:

```
>> Rx=5;
```

Визначення кроку по осі x:

```
>> stepx=0.05;
```

Визначення лівої межі для y:

```
>> Ly=-5;
```

Визначення правої межі для y:

```
>> Ry=5;
```

Визначення кроку по осі y:

```
>> stepy=0.05;
```

Дані для тривимірного графіка:

```
>> xs=Lx:stepx:Rx;
>> ys=Ly:stepy:Ry;
```

Створення сітки координат :

```
>>[X,Y] = meshgrid(xs,ys);
```

Обчислення функції:

```
>> Z=X.^2-Y.^2+X.*Y+2;
```

Побудова тривимірного графіка (рис. Д.2) :

```
>> surf(X,Y,Z)
```

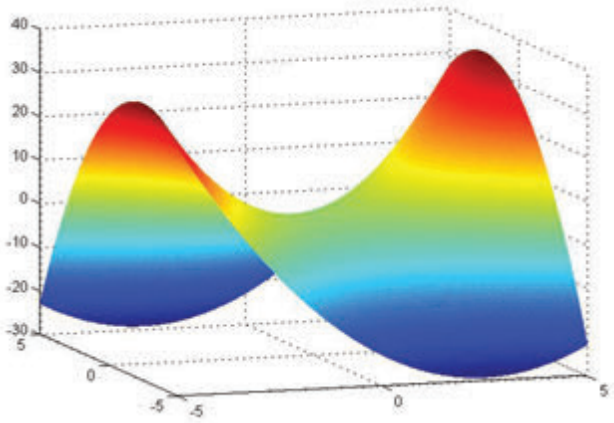


Рис. Д.2 – Графік функції $f(x) = x^2 - y^2 + xy + 2$

Для збереження вікна з графіком у графічному форматі необхідно набрати в командному вікні код:

```
>> print -dtiff -r300 'c:/tmp/Graphic'
```

Тут – dtiff – формат файлу (tiff), – r300 – розподільна здатність (300 dpi), Graphic – ім'я файлу з рисунком, включаючи ім'я каталогу.

Для нотаток

Для нотаток

Для нотаток

Навчальне видання

Д.В. Ланде
І.Ю. Субач
Ю.Є. Бояринова

ОСНОВИ ТЕОРІЇ І ПРАКТИКИ ІНТЕЛЕКТУАЛЬНОГО АНАЛІЗУ ДАНИХ У СФЕРІ КІБЕРБЕПЕКИ

Навчальний посібник

Підписано до друку 27.07.2018
Формат 60 х 84/8. Офсетний друк.
Умов. др. арк. 12,5. Наклад 200 прим.
Віддруковано з оригінал-макета ФОП Крячун Ю.В.

ISBN 978-966-2577-12-9