

## **Питання на екзамен із Штучного інтелекту:**

1. Інтелект. Поняття та означення інтелекту
2. Штучний інтелект: поняття, означення, основна задача, характеристика задач
3. Архітектура інтелектуальної системи лінгвістичного аналізу
4. Морфологічний аналіз
5. Моделі представлення синтаксису природньої мови
6. Синтаксичний аналіз тексту (Висхідний аналіз текстів, Нисхідний аналіз текстів)
7. Алгоритм Earley
8. Алгоритм Cocke-Younger-Kasami (CYK)
9. Семантичний аналіз текстів
10. Онтології (Бази знань. Семантичні мережі. Фреймові моделі.)
11. Формати онтологій (модально-рольові відношення - ?)
12. Модально-рольові відношення, граматична відмінність
13. Міри семантичної близкості: ([статья Марченко: http://ekmair.ukma.edu.ua/bitstream/handle/123456789/2167/Metod%20obchyslennia%20semantichnoi%20blyzkosti.pdf?sequence=1&isAllowed=y](http://ekmair.ukma.edu.ua/bitstream/handle/123456789/2167/Metod%20obchyslennia%20semantichnoi%20blyzkosti.pdf?sequence=1&isAllowed=y))
  - косинусна міра ( [https://uk.wikipedia.org/wiki/Косинус\\_подібності](https://uk.wikipedia.org/wiki/Косинус_подібності) , <https://uk.wikipedia.org/wiki/TF-IDF> )
  - Leacock and Chodorov
  - W.N. Palmer
  - Pesnik
15. Case grammar (Падéжная граммáтика, «ролевáя грамматика»)

[https://en.wikipedia.org/wiki/Case\\_grammar](https://en.wikipedia.org/wiki/Case_grammar)

[https://ru.wikipedia.org/wiki/Падежная\\_грамматика](https://ru.wikipedia.org/wiki/Падежная_грамматика)

16. Латентний семантичний аналіз

[https://ru.wikipedia.org/wiki/Латентно-семантический\\_анализ](https://ru.wikipedia.org/wiki/Латентно-семантический_анализ)

[https://uk.wikipedia.org/wiki/Латентно-семантичний\\_аналіз](https://uk.wikipedia.org/wiki/Латентно-семантичний_аналіз)

<https://habr.com/post/110078/>

<https://habr.com/post/240209/>

17. Невід'ємна матрична факторизація ([статья Марченко](#)), алгоритм

Lee&Seung

[https://uk.wikipedia.org/wiki/Розклад\\_невід%27ємних\\_матриць](https://uk.wikipedia.org/wiki/Розклад_невід%27ємних_матриць)

18. Невід'ємна тензорна факторізація ([статья Марченко](#))

19. Латентне розміщення Діріхле

[http://www.machinelearning.ru/wiki/index.php?title=Тематическое\\_моделирование#.D0.9B.D0.B0.D1.82.D0.B5.D0.BD.D1.82.D0.BD.D0.BE.D0.B5\\_.D1.80.D0.B0.D0.B7.D0.BC.D0.B5.D1.89.D0.B5.D0.BD.D0.B8.D0.B5\\_.D0.94.D0.B8.D1.80.D0.B8.D1.85.D0.BB.D0.B5](http://www.machinelearning.ru/wiki/index.php?title=Тематическое_моделирование#.D0.9B.D0.B0.D1.82.D0.B5.D0.BD.D1.82.D0.BD.D0.BE.D0.B5_.D1.80.D0.B0.D0.B7.D0.BC.D0.B5.D1.89.D0.B5.D0.BD.D0.B8.D0.B5_.D0.94.D0.B8.D1.80.D0.B8.D1.85.D0.BB.D0.B5)

[https://ru.wikipedia.org/wiki/Латентное\\_размещение\\_Дирихле](https://ru.wikipedia.org/wiki/Латентное_размещение_Дирихле)

[https://en.wikipedia.org/wiki/Latent\\_Dirichlet\\_allocation](https://en.wikipedia.org/wiki/Latent_Dirichlet_allocation)

20. Кластерізація. Алгоритми кластерізації

[https://uk.wikipedia.org/wiki/Кластерний\\_аналіз](https://uk.wikipedia.org/wiki/Кластерний_аналіз)

21. Алгоритм k-means

[https://ru.wikipedia.org/wiki/Метод\\_k-средних](https://ru.wikipedia.org/wiki/Метод_k-средних)

22. Моделі та методи машинного навчання:

- Naive Bayes

- Linear Regression
- Logistic Regression
- Maximum Entropy

22. Приховані марковські моделі:

[https://uk.wikipedia.org/wiki/Прихована\\_марковська\\_модель](https://uk.wikipedia.org/wiki/Прихована_марковська_модель)

- Viterbi algorithm

[https://neerc.ifmo.ru/wiki/index.php?title=Алгоритм\\_Витерби](https://neerc.ifmo.ru/wiki/index.php?title=Алгоритм_Витерби)

- Forward-backward algorithm
- Baum-Welch algorithm

23. Генетичний алгоритм

[https://uk.wikipedia.org/wiki/Генетичний\\_алгоритм](https://uk.wikipedia.org/wiki/Генетичний_алгоритм)

24. Нейронно-мережевий підхід

- CNN

[https://ru.wikipedia.org/wiki/Свёрточная\\_нейронная\\_сеть](https://ru.wikipedia.org/wiki/Свёрточная_нейронная_сеть)

[https://uk.wikipedia.org/wiki/Згорткова\\_нейронна\\_мережа](https://uk.wikipedia.org/wiki/Згорткова_нейронна_мережа)

- RNN

[https://ru.wikipedia.org/wiki/Рекуррентная\\_нейронная\\_сеть](https://ru.wikipedia.org/wiki/Рекуррентная_нейронная_сеть)

[https://uk.wikipedia.org/wiki/Рекуррентна\\_нейронна\\_мережа](https://uk.wikipedia.org/wiki/Рекуррентна_нейронна_мережа)

### Додаткові питання:

- ➔ Предельная факторизация матриц
- ➔ Неотъемлемая факторизация тензор

## 1. Інтелект. Поняття та означення інтелекту

**Інтелéкт** (від лат. *intellectus* «відчуття», «сприйняття», «розуміння») — це інформаційний потенціал знань конкретної особистості, отриманий в результаті функціонування свідомості, мислення та розуму людини.

Штучний інтелект — це розділ обчислюальної лінгвістики та інформатики який займається формалізацією проблем і завдань інтелекту.

**Інтелéкт** (розум) — сукупність розумових здібностей. За допомогою розуму людина:

- (*розумові здібності*)
  - вчиться, тобто здобуває нові знання та удосконалює свої розумові здібності;
- (*досягнутий рівень розумового розвитку*)
  - розуміє мову та абстрактні ідеї;
  - обдумує бажаний для неї результат (проблеми тощо) та складає план по його досягненню, а потім безпосередньо здійснює заплановане, роблячи висновки (як зі своєї діяльності, так і діяльності інших) та удосконалюючи свій практичний досвід.

## **2. Штучний інтелект: поняття, означення, основна задача, характеристика задач**

Штучний інтелект — розділ [комп'ютерної лінгвістики](#) та [інформатики](#), що опікується формалізацією проблем та завдань, які подібні до дій, які виконує людина

Основні задачі

- розпізнавання, класифікація зображень
- передбачення
- планування(розв'язання через пошук)
- моделювання складної життедіяльності живих істот
- NLP

Штучний інтелект вивчає методи розв'язання задач, для яких не існує способів розв'язання або вони не коректні (через обмеження в часі, [пам'яті](#) тощо). Завдяки такому визначенню інтелектуальні алгоритми часто використовуються для розв'язання [NP-повних задач](#), наприклад, [задачі комівояжера](#).

Моделювання міркувань має на увазі створення [символьних систем](#), на вході яких поставлена деяка задача, а на виході очікується її розв'язок. Як правило, запропонована задача уже [формалізована](#), тобто переведена на математичну форму, але або не має алгоритму розв'язання, або цей алгоритм занадто складний, трудомісткий тощо. В цей напрям входять: [доведення теорем](#), [прийняття рішень і теорія ігор](#), [планування і диспетчеризація](#), [прогнозування](#).

Задачі розпізнавання об'єктів вже частково розв'язуються в рамках інших напрямків. Сюди відносяться [розпізнавання символів](#),

рукописного тексту, мови, аналіз текстів. Особливо слід згадати [комп'ютерне бачення](#), яке пов'язане з машинним навчанням та робототехнікою.

[Робототехніка](#) і штучний інтелект часто поєднуються одне з одним. Об'єднання цих двох наук, створення [інтелектуальних роботів](#), можна вважати ще одним напрямом ШІ.

[Машинне навчання](#) — це розділ штучного інтелекту, що має за основу побудову та дослідження систем, які можуть самостійно навчатись з даних. Наприклад, система машинного навчання може бути натреною на електронних повідомленнях для розрізнення спаму і прийнятних повідомлень. Після навчання вона може бути використана для класифікації нових повідомлень [електронної пошти](#) на [спам](#) та не-спам.

[Машинний зір](#) — це застосування комп'ютерного зору в промисловості та виробництві. В той час як комп'ютерний зір — це загальний набір методів, що дозволяють комп'ютерам бачити, область інтересу машинного зору, як інженерного напрямку, є цифрові [пристрої введення/виведення](#) та [комп'ютерні мережі](#), призначенні для контролю виробничого обладнання, такого як роботи-маніпулятори чи апарати для вилучення бракованої продукції.

### **3. Архітектура інтелектуальної системи лінгвістичного аналізу**

Лінгвістичну обробку, як правило, проходить у три етапи: синтаксико-семантичний аналіз вихідного тексту, застосування правил перетворень з більш-менш абстрактним рівнем представництва та генерування цільового тексту з синтаксичного представлення вхідного тексту.

У моделях, заснованих на ІІ, процедури усунення неоднозначності радикально відрізняються і ґрунтуються перш за все на аналізі ситуації та довідкової інформації (бази знань), в той час як лінгвістичні методи аналізу контексту служать тільки в якості вторинних резервних засобів.

Лінгвістичні евристики є застосовними в конкретних мовах, вони звичайно звужують область пошуку конкретними правилами, зменшують відрізки тексту, на яких потрібно застосовувати повний перебір. Логічні евристики - зв'язані з припущенням, що семантична структура речення найчастіше буває деревом.

## 4. Морфологічний аналіз

*Морфологічний* аналіз можна визначити як аспект дослідження, що полягає у ідентифікації, аналізі та описі структури або форм слів у мові.

Автоматичний морфологічний аналіз тексту (АМА) є одним із етапів роботи систем автоматичного аналізу тексту. У результаті роботи АМА кожному слововживанню припісуються значення граматичних категорій (частина мови, рід, число, відмінок, час, вид, тощо).

Формальним результатом аналізу тексту є орієнтований граф, що узагальнює зв'язки між окремими словами речення. У кожній вершині графу міститься статистично отриманий список спільнокореневих слів, які відповідають одному мовному образу. З метою морфологічного аналізу мовних образів застосуємо контекстно-вільну граматику, що розпізнає слова

$$G = (V_T, V_N, S, P), \quad (1)$$

де  $V_T$  – алфавіт термінальних (основних) символів (морфем);

$V_N$  – алфавіт нетермінальних символів (метазмінних), причому  $V_T \cap V_N = \emptyset$ ;

$S$  – стартовий символ;

$P$  – скінчений набір правил  $\varphi \rightarrow \psi$ , де  $\varphi \in V_N$ , а  $\psi \in F(V)$  – довільні слова вільної напівгрупи слів над алфавітом  $F = V_T \cup V_N$ .

Розглянута граматика (1) породжує мову  $L$  як множину  $L(G)$  всіх правильних слів у граматиці  $G$ . Будемо вважати, що до складу множини  $V_N$  входять  $w$  (*СЛОВО*),  $x$  (*ПРЕФІКС*),  $y$  (*КОРІНЬ*) та  $z$  (*СУФІКС*). Тоді маємо  $V_N = \{w, x, y, z, v\}$ , а  $S = w$ .

## 5. Моделі представлення синтаксису природної мови

**Моделі тензорного представлення даних** про частоту різних типів синтаксичних сполучень слів у реченнях, наприклад 3-мірних сполучень типу subject - verb - object, 4-мірних сполучень типу subject - verb - direct\_object - indirectobject, або інших синтаксичних сполучень довжини, що не перевищує розмір ність тензора N. У тензорі кожен вимір відповідає деякому фіксованому члену речення — підмет, присудок, додаток, означення, обставина та інші. N-мірні тензори містять оцінки частоти вживання сполучень різних наборів слів у реченнях природної мови, де враховуються синтаксичні позиції слів у реченнях. Після обробки великих текстових корпусів та накопичення значного обсягу даних у тензорі, формується N-вимірний масив опису поведінки лексичних одиниць у реченнях даної мови: тобто для множини слів у тензорі дано опис, у які синтаксичні відношення ці слова мають властивість вступа ти, з якими словами вони вступають у ці відношення і з якою частотою.

Після факторизації тензора його дані представлено у вигляді N матриць, що складаються з векторів-стовпчиків розмірності k (де значення k набагато менше, ніж кількість точок-слів у будь-якому з N вимірів тензора). Параметр k - це розмірність латентного семантичного простору, число ознакових вимірів у ньому.

Дана модель дозволяє досить успішно автоматачно виділяти з корпусів текстів такі лінгвістичні структури, як *селективні преференції* в реченнях (selectional preferences) [9] та *субштегоріальні фрейми дієслів* (Verb SubCategorization Frame) [10], які поєднують у собі дані про семантичні та синтаксичні властивості зв'язків-відношень між дієсловами та їх аргументами-іменниками в реченнях природної мови.

**Недоліком такої моделі** можна вважати не гнучкість і обмеженість представлення синтаксису. Розмірність тензора обмежує максимальну довжину речень-словосполучень, що описуються даною моделлю.

**Модель дерев підпорядкування** орієнтована на керуючі зв'язки тільки між словами, а **модель систем складових** враховує ієрархічне відношення вкладеності словосполучень в лінійній структурі тексту. Ці чинники лише наближено описують дійсні комунікативні властивості, що містяться в синтаксичних структурах.

## **6. Синтаксичний аналіз тексту (Висхідний аналіз текстів, Нисхідний аналіз текстів)**

**Синтаксичний аналіз (пárсинг)** (англ. *parsing*) — в інформатиці це процес аналізу вхідної послідовності символів, з метою розбору граматичної структури згідно із заданою [формальною граматикою](#).

Під час синтаксичного аналізу текст оформлюється у [структурну даних](#), зазвичай — в [дерево](#), яке відповідає синтаксичній структурі вхідної послідовності, і добре підходить для подальшої обробки. Зазвичай синтаксичні аналізатори працюють в два етапи: на першому ідентифікуються осмислені [токени](#) (виконується [лексичний аналіз](#)), на другому створюється [дерево розбору](#).

### **Висхідний синтаксичний аналіз.**

#### **Загальні принципи висхідного аналізу**

При такій стратегії дерево синтаксичного аналізу будується, рухаючись від листя (вхідної програми, яка розглядається як рядок символів) до кореня дерева (аксіоми граматики). Аналізатор (розвізнавач) шукає частину рядка, яку можна звести до нетермінального символу. Таку частину рядка називають фразою. У більшості висхідних розвізнавачів відшукується найлівіша фраза, що безпосередньо зводиться до нетермінального символу (така фраза називається основою). Основа заміняється нетермінальним символом. У отриманому рядку знову відшукується основа, заміняється нетермінальним символом і т.д.

Процес продовжується або до отримання початкового символу (аксіоми), або до встановлення неможливості зведення рядка до початкового символу. Послідовність проміжних рядків, яка закінчується початковим символом, утворює розбір. Якщо рядок не зводиться до початкового символу, то розбір не існує, і вхідна програма синтаксично некоректна.

**Низхідний синтаксичний аналіз** — один з методів визначення приналежності вхідного рядка деякій формальній мові, описаній LL(k)-граматикою. Це клас алгоритмів лексичного аналізу, де правила формальної граматики розкриваються, починаючи зі стартового символу до отримання потрібної послідовності токенів.

Для кожного нетермінального символу  $K$  будується функція, яка для будь-якого вхідного слова  $x$  робить дві речі:

- Знаходить найбільший початок  $z$  слова  $x$ , здатний бути початком виводжуваного з  $K$  слова
- Визначає, чи є початок  $z$  виводжуваним з  $K$

Така функція має задовольняти такі критерії:

- зчитувати із ще необробленого вхідного потоку максимальний початок  $A$ , який є початком деякого слова, виводжуваного з  $K$
- визначати чи є  $A$  вивідним з  $K$  або просто невивідним початком виводжуваного з  $K$  слова

У випадку, якщо такий початок зчитати не вдається (і коректність функції для нетермінала  $K$  доведена), тоді вхідні дані не відповідають мові, і потрібно зупинити розбір.

Розбір містить у собі виклики описаних вище функцій. Якщо для зчитаного нетермінала є складене правило, тоді при його розборі

будуть викликані інші функції для розбору терміналів, що входять в нього. Дерево викликів, починаючи із самої «верхньої» функції еквівалентно дереву розбору.

Найбільш простий і «людяний» варіант створення аналізатора, що використовує метод рекурсивного спуску, - безпосереднє програмування за кожним правилом вводу для нетерміналів граматики.

## 7. Алгоритм Earley

**Алгоритм Эрли** (англ. *Earley*) — алгоритм синтаксического анализа предложения по контекстно-свободной грамматике, основанный на методе динамического программирования. В отличие от алгоритма Кока — Янгера — Касами, который требует приведения грамматики к нормальной форме Хомского, алгоритм Эрли привлекателен тем, что не накладывает ограничений на используемую для анализа контекстно-свободную грамматику. Кроме того, **Алгоритм Кока — Янгера — Касами** работает по принципу «снизу-вверх», то есть строит возможные деревья разбора предложения начиная с вершины. В отличие от него Алгоритм Эрли реализует стратегию вывода «слева-направо».

### Основні поняття алгоритму [\[ред.\]](#) [\[ред. код\]](#)

Алгоритм Ерлі за вхідним ланцюжком та граматикою породжує список розбору для даного вхідного ланцюжка в заданій граматиці.

Нехай вхідна граматика задається четвіркою  $G = (T, N, S, P)$ .

Вхідний ланцюжок задається послідовністю терміналів  $w=a_1, a_2, \dots, a_n$

Списком розбору будемо називати послідовність списків ситуацій  $I_0, I_1, \dots, I_n$ .

Ситуацією будемо називати конструкцію вигляду  $[A \rightarrow X_1, \dots, X_k \cdot X_{k+1}, \dots, X_m, i]$  (де  $k, i$  — довільні натуральні числа від 0 до  $m$ ,  $\cdot$  — метасимвол, який не належить ні  $N$  ні  $T$ ), якщо  $A \rightarrow X_1, \dots, X_m$  — правило з  $P$ .

Список ситуацій  $I_j$  для слова  $w$  будемо будувати таким чином:

$[A \rightarrow \alpha \cdot \beta, i]$  ( $i \leq j$ ) належить  $I_j$  тоді і тільки тоді, якщо існують такі  $\gamma$  та  $\delta$ , що  $S \Rightarrow^* \gamma A \delta$ ,  $\gamma \Rightarrow^* a_1 \dots a_i$  та  $\alpha \Rightarrow^* a_{i+1} \dots a_j$ . Тобто певна частина слова  $w$  [ $1..j$ ] може бути виведена використовуючи  $A \rightarrow \alpha \cdot \beta$ .

Зрозуміло, що  $w$  буде належати  $L(G)$  т. і т.т., якщо  $[S \rightarrow \alpha \cdot, 0]$  буде належати  $I_n$ .

Одержаній список розбору може слугувати базою для багатьох алгоритмів, зокрема побудови правого розбору ланцюжка.

### Вхід алгоритму [\[ред.\]](#) [\[ред. код\]](#)

1) Граматика  $G = (T, N, S, P)$

2) Вхідний ланцюжок  $w=a_1, a_2, \dots, a_n$

### Вихід алгоритму [\[ред.\]](#) [\[ред. код\]](#)

Список розбору  $I_0, I_1, \dots, I_n$ .

Спочатку ініціюємо список  $I_0$ .

1. Для всіх правил  $S \rightarrow a$ , включити  $[S \rightarrow \cdot a, 0]$  в  $I_0$ .

Поки в  $I_0$  можна включати, виконуємо кроки 2-3

2. Якщо  $[B \rightarrow \gamma \cdot, 0]$  належить  $I_0$ , то включити в  $I_0$  всі ситуації вигляду  $[A \rightarrow aB \cdot \beta, 0]$  (якщо вони ще не там) для всіх  $[A \rightarrow a \cdot B\beta, 0]$ , що вже належать  $I_0$ .

3. Якщо  $A \rightarrow [a \cdot B\beta, 0]$  належить  $I_0$ , то включити в  $I_0$  ситуації вигляду  $[B \rightarrow \cdot \gamma, 0]$  (якщо вона ще не там) для всіх правил вигляду  $B \rightarrow \gamma$ .

Нехай ми маємо побудовані списки  $I_0, I_1, \dots, I_{j-1}$ .

4. Для кожної ситуації  $[B \rightarrow a \cdot a\beta, i]$  з  $I_{j-1}$  включити до  $I_j$  ситуацію  $[B \rightarrow aa \cdot B, i]$  Поки в  $I_j$  можна включати, виконуємо кроки 5-6.

5. Нехай  $[A \rightarrow a \cdot, i]$  належить до  $I_j$ . Шукати в  $I_i$  ситуації вигляду  $[B \rightarrow a \cdot A\beta, k]$ . Для кожної з них включити до  $I_j$  ситуацію  $[B \rightarrow aA \cdot \beta, k]$ .

6. Нехай  $[A \rightarrow a \cdot B\beta, i]$  належить  $I_j$ . Для кожного правила  $B \rightarrow \gamma$  включити до  $I_j$  ситуацію  $[B \rightarrow \cdot \gamma, j]$

## 8. Алгоритм Cocke-Younger-Kasami (CYK)

**Алгоритм Кока – Янгера – Касами** ([англ. Cocke – Younger – Kasami algorithm](#)), **алгоритм СҮК** либо **СКУ** – алгоритм, позволяющий установить, можно ли в заданной [контекстно-свободной грамматике](#) вывести заданную строку, и если это так, то предоставить её вывод. Другими словами, это алгоритм [синтаксического анализа](#) строки. Алгоритм реализует синтаксический анализ снизу-вверх и основывается на методе [динамического программирования](#).

## 9. Семантичний аналіз текстів

**Семантический анализ** — этап в последовательности действий алгоритма автоматического понимания текстов, заключающийся в выделении семантических отношений, формировании семантического представления текстов. Один из возможных вариантов представления семантического представления — структура, состоящая из «текстовых фактов»<sup>[1]</sup>. Семантический анализ в рамках одного предложения называется *локальным семантическим анализом*.

В общем случае семантическое представление является графом, **семантической сетью**, отражающим бинарные отношения между двумя узлами — смысловыми единицами текста. Глубина семантического анализа может быть разной, а в реальных системах чаще всего строится только лишь синтаксико-семантическое представление текста или отдельных предложений. Так, в работе<sup>[2]</sup> семантический анализ осуществляется одновременно с синтаксическим с помощью механизма **расширенных сетей переходов**.

## **10.Онтології (Бази знань. Семантичні мережі. Фреймові моделі.)**

**Онтологія** в информатике (новолат. *ontologia* от др.-греч. ὄν — род. п. ὄντος — сущее, то, что существует и λόγος — учение, наука) — это попытка всеобъемлющей и подробной **формализации** некоторой области **знаний** с помощью **концептуальной схемы**. Обычно такая схема состоит из **структуре данных**, содержащей все **релевантные** классы объектов, их связи и правила (**теоремы**, ограничения), принятые в этой области.

Современные онтологии строятся по большей части одинаково, независимо от языка написания. Обычно они состоят из **экземпляров**, **понятий**, **атрибутов** и **отношений**.

### **Экземпляры**

**Экземпляры** (англ. *instances*) или индивиды (англ. *individuals*) — это объекты, основные нижеуровневые компоненты онтологии; могут представлять собой как физические объекты (люди, дома, планеты), так и **абстрактные** (числа, слова). Строго говоря, онтология может обойтись и без конкретных объектов, однако, одной из главных целей онтологии является **классификация** таких объектов, поэтому они также включаются.

### **Понятия**

**Понятия** (англ. *concepts*) или **классы** (англ. *classes*) — абстрактные группы, коллекции или наборы объектов. Они могут включать в себя экземпляры, другие классы, либо же сочетания и того, и другого. Пример:

- Понятие «люди», вложенное понятие «человек». Чем является «человек» — вложенным понятием, или экземпляром (индивидуом) — зависит от онтологии.
- Понятие «индивидуы», экземпляр «индивиду».

Классы онтологии составляют **таксономию** — иерархию понятий по отношению вложения<sup>[1]</sup>.

## Атрибуты

Объекты в онтологии могут иметь **атрибуты**. Каждый атрибут имеет по крайней мере имя и значение и используется для хранения информации, которая специфична для объекта и привязана к нему. Например, объект Автомобиль-модели-А имеет такие атрибуты, как:

- *Название: Автомобиль-модели-А*
- *Число-дверей: 4*
- *Двигатель: {4.0Л, 4.6Л}*
- *Коробка-передач: 6-ступенчатая*

Значение атрибута может быть **сложным типом данных**. В данном примере значение атрибута, который называется *Двигатель*, является списком значений простых типов данных.

## Отношения

Важная роль атрибутов заключается в том, чтобы определять **отношения** (зависимости) между объектами онтологии. Обычно отношением является атрибут, значением которого является другой объект.

**Бáза знáнь, БЗ** (англ. *Knowledge base, KB*) — це особливого роду база даних, розроблена для управління знаннями (метаданими), тобто збором, зберіганням, пошуком і видачею знань.

**База знань** — це сукупність відомостей (про реальні об'єкти, процес, події або явища), що відносяться до певної теми або задачі, організована так, щоб забезпечити зручне представлення цієї сукупності як в цілому так і будь-якої її частини. Це означає, що система управління базою знань (саме знань, а не даних) повинна забезпечити представлення й обробку моделі, зіставною за своєю складністю з моделлю, що використовується свідомістю людини.

### **Семантичні мережі.**

[https://uk.wikipedia.org/wiki/Семантична\\_мережа](https://uk.wikipedia.org/wiki/Семантична_мережа)

**Семантична мережа** — інформаційна модель предметної області, що має вигляд орієнтованого графа, вершини якого відповідають об'єктам предметної області, а ребра задають відносини між ними. Об'єктами можуть бути поняття, події, властивості, процеси<sup>[1]</sup>.

### **Класифікація семантичних мереж**

Для всіх семантичних мереж справедливе розділення за арністю і кількістю типів відносин.

За кількістю типів, мережі можуть бути **однорідними** і **неоднорідними**. Однорідні мережі мають тільки один тип відносин (стрілок), наприклад, такою є вищезазначена класифікація біологічних видів (з єдиним відношенням АКО). У неоднорідних мережах кількість типів відносин більше двох. Класичні ілюстрації даної моделі представлення знань представляють саме такі мережі. Неоднорідні мережі представляють більший інтерес для практичних цілей, але і більшу складність для досліджень.

За арністю, типовими є мережі з **бінарними** відносинами (що зв'язують рівно два поняття). Бінарні відносини, дійсно, є простими й

зручно виглядають на графі у вигляді стрілки між двома поняттями. Крім того, вони відіграють виняткову роль у математиці. На практиці, проте, можуть знадобитися відносини, що зв'язують більше двох об'єктів, — **N-арні**. При цьому виникає складність — як відобразити подібний зв'язок на графі, щоб не запутатися. Концептуальні графи (див. [нижче](#)) знімають це ускладнення, представляючи кожне відношення у вигляді окремого вузла.

Крім концептуальних графів існують інші модифікації семантичних мереж, це є ще однією основою для класифікації (**за реалізацією**). Див. детальніше у відповідному розділі [нижче](#).

[https://uk.wikipedia.org/wiki/Фрейм\\_\(знання\)](https://uk.wikipedia.org/wiki/Фрейм_(знання))

**Фрейм** (англ. *frame* — «каркас», «рамка») — це структура, що описує деякий складний об'єкт або абстрактний образ або модель для представлення деякої концепції ([метод представлення знань](#)). Модель містить слоти, визначені фасетами. З такої моделі певної концепції нічого не можна забрати, атрибути моделі можна лише заповнити.

Структура фрейма включає три основних типи даних: поняття (назва фрейма), характеристика (назва термінала — вершини нижнього рівня), значення характеристики (заповнювач термінала). У зв'язку з цим можна вважати, що у фреймі реалізовано деякі загальні принципи, що властиві організації [баз даних](#), де як одиниці виділяються об'єкти, характеристики та їхні значення, а також [семантичним сіткам](#), у яких розрізняють абстрактний та конкретний рівень. Фрейм надає засоби організації знань в слотах, що містять характеристики та структури.

## 11.Формати онтологій (модально-рольові відношення - ?)

Сьогодні виділяють три основні класи мов опису онтологій:

- традиційні мови специфікації онтологій: Ontolingua, CycL та мови, засновані на дескриптивних логіках (такі як LOOM), також мови, засновані на фреймах (OKBC, OCML, Flogic);
- більш пізні мови, засновані на Web-стандартах (XOL, SHOE, UPML);
- спеціальні мови для обміну онтологіями через Web: RDF(S), DAML, OIL, OWL

Специализированные (предметно-ориентированные) онтологии — это представление какой-либо области знаний или части реального мира. В такой онтологии содержатся специальные для этой области значения терминов. К примеру, слово «поле» в сельском хозяйстве означает участок земли, в физике — один из видов материи, в математике — класс алгебраических систем.

Общие онтологии используются для представления понятий, общих для большого числа областей. Такие онтологии содержат базовый набор терминов, глоссарий или тезаурус, используемый для описания терминов предметных областей.

Если использующая специализированные онтологии система развивается, то может потребоваться их объединение. Подзадачей объединения онтологий является задача отображения онтологий. И для инженера по онтологиям это серьёзная задача. Онтологии даже близких областей могут быть несовместимы друг с другом. Разница может появляться из-за особенностей местной культуры, идеологии или вследствие использования другого языка описания. Объединение онтологий выполняют как вручную, так и в полуавтоматическом режиме. В целом это — трудоёмкий, медленный и дорогостоящий

процесс. Использование *базисной онтологии* — единого глоссария — несколько упрощает эту работу. Есть научные работы по технологиям объединения, но они по большей части теоретические.

## **12.Модально-рольові відношення, граматична відмінність**

Модально-рольові відношення, за допомогою яких концепти типу “дія” відображаються в онтології, в мові виражаються за допомогою синтаксису (наприклад, відношення ІНСТРУМЕНТ англійською мовою виражається за допомогою прийменникової групи з прийменниками “with” та “by”, українською мовою – за допомогою орудного відмінку іменника (наприклад “вдарити молотком”). Тому необхідним стає введення додаткового типу зв’язку, що прив’язує потрібні синтаксичні шаблони до відповідних рольових семантичних відношень в онтології. Таким чином, отримаємо структурний зв’язок між синтаксисом та семантикою в межах інтегрованої онтологічної бази.

## 13. Міри семантичної близкості:

- **косинусна міра**
- **Leacock and Chodorov**
- **W.N. Palmer**
- **Pesnik**

**Міра семантичної близькості** – це кількісна величина, яка показує на скільки два поняття близькі (тобто пов'язані або схожі) між собою.

Наявні методи визначення семантичної близькості–зв'язності слів природної мови. Існує дві групи методів:

- методи, основою яких є пошуку відстані між концептами у семантичній мережі.
- методи, що базуються на лексичному перетині словарних значень слів.

### **Методи, засновані на відстані між концептами**

Методи цієї групи засновано на відшуканні відстані між двома концептами у семантичній мережі (WordNet, дерево категорій Wikipedia). Між двома поняттями лежить найкоротший шлях і на його основі визначається семантична близькість між словами. Одну з перших таких мір запропонував Резнік (Resnik P):

$$\frac{1}{N_p},$$

де  $N_p$  – кількість вершин у найкоротшому шляху. Очевидний недолік цієї міри у тому, що для деяких концептів вкладеність класів, до яких вони належать, є більшою ніж для інших через їхню природу та

походження. Для розв'язання цієї проблеми Лікок і Ходоров (**Leacock C., Chodorow M.**) запропонували метод, який нормалізує довжину шляху з урахуванням глибини загальної ієархії:

$$-\log\left(\frac{\text{len}(c_1, c_2)}{2D}\right),$$

де D – максимальна глина ієархії.

By і Палмер (Wu Z. and Palmer M.) запропонували інший підхід, що враховує глибину найменшого спільного предка концептів в ієархії:

$$-\log\left(\frac{\text{depth}(\text{LSO}(c_1, c_2))}{\text{depth}(c_1) + \text{depth}(c_2)}\right),$$

де LSO – відстань від найменшого спільного предка концептів до кореня таксономії.

### **Методи, засновані на лексичному перетині**

Перший алгоритм такого типу розробив Леск. Він сконструював алгоритм, який в основі має припущення про те, що пов'язані поняття будуть визначатися (або пояснюватись) однако вими словами. Леск використав цей підхід для розв'язання задачі добору правильного значення слова у деякому контексті. Порядок роботи алгоритму: спочатку шукається лексичний перетин означень заданого слова з означеннями всіх слів тексту. Далі обирається значення слова з найбільшим сумарним перетином, що найбільше відповідає контексту. Лексичний перетин обчислюється як кількість спільних слів двох означень. Недолік такого підходу у тому, що статті у звичайних словниках є досить короткими, а тому можуть погано відображати семантичну

близь кість деяких слів. Бенаржи і Педерсен розширили метод Леска, додавши до перетину словники інших слів, що пов'язані з початковими, тим чи іншим шляхом. Наприклад, в одній з реалізацій використовувався WordNet і додавалися слова, що є безпосередніми предками заданих слів в ієрархії цієї семантичної мережі.

У розробці комп'ютерної системи обчислена семантичної близькості дуже важливо обрати збалансоване, коректне та об'ємне джерело синонімічних словах даних. У таких дослідженнях використовують різноманітні джерела: від звичайних енциклопедій і тезаурусів до таких сучасних баз знань, як: WordNet, ConceptNet, Wikipedia. Найбільш вагомі результати були отримані на основі WordNet та Wikipedia.

---

[https://uk.wikipedia.org/wiki/Косинус\\_подібності](https://uk.wikipedia.org/wiki/Косинус_подібності)

**Косинус подібності** (англ. *cosine similarity*) – коефіцієнт подібності двох не нульових векторів у предгільбертовому просторі, який обчислюється як косинус кута між ними.

Косинус  $0^\circ$  дорівнює 1, а для всіх інших значень кута в інтервалі  $(0, \pi]$  буде менше за 1. Отож, це оцінка напрямку, а не величини: два вектори з одним напрямком мають косинус подібності 1, а два вектори, які утворюють кут  $90^\circ$  один відносно одного, мають подібність 0, а два діаметрально направлені вектори мають подібність -1, незалежно від їх довжини. Косинус подібності часто використовують в позитивному просторі, для якого результат обмежений проміжком  $[0, 1]$ . Назва походить від терміну «направлений косинус»: в цьому випадку

[одиничні вектори](#) максимально «подібні», якщо вони паралельні і максимально «різні», якщо вони [ортогональні](#) (перпендикулярні).

Наприклад, при [інформаційному пошуку](#) та [аналізі тексту](#), кожен термін пов'язаний з окремим виміром, і тому документ характеризується вектором, де значення кожного виміру відповідає кількості разів, що термін з'являється у документі. Тоді косинус подібності дає корисну оцінку того, наскільки подібні два документи у термінах теми.

## 15. Case grammar

[https://en.wikipedia.org/wiki/Case\\_grammar](https://en.wikipedia.org/wiki/Case_grammar)

[https://ru.wikipedia.org/wiki/Падежная\\_грамматика](https://ru.wikipedia.org/wiki/Падежная_грамматика)

**Падежная грамматика**, «ролевая грамматика» — метод описания семантики предложения (за исключением модальных и перформативных элементов) как системы семантических валентностей, в которой значение вершинного глагола диктует роли («падежи»), исполняемые именными составляющими. Иногда рассматривается как одна из разновидностей порождающей семантики

**Валентность** в синтаксисе — способность слова вступать в синтаксические связи с другими элементами

Эта теория анализирует поверхностную синтаксическую структуру предложений, изучая комбинацию глубоких падежей (то есть семантических ролей) — Agent, Object, Benefactor, Location or Instrument — которые требуются конкретным глаголом. Например, для глагола «give» в английском языке требуются агент (A) и объект (O), а также бенефициар (B), например: "Jones (A) gave money (O) to the school (B)."

Падеж описывает ключевые аспекты семантической валентности, глаголов, прилагательных и существительных. Фреймы падежа подчиняются определенным ограничениям, таким как глубокий регистр может встречаться только один раз в предложении. Некоторые из случаев являются обязательными, а другие необязательными. Обязательные дела не могут быть удалены, рискуя привести к неграмотным предложениям.

Фундаментальная гипотеза падежной грамматики заключается в том, что грамматические функции, такие как субъект или объект, определяются глубокой семантической валентностью глагола, который находит его синтаксический коррелят в таких грамматических категориях, как субъект и объект, и в грамматических падежах, таких как именительный падеж. иерархию для универсального правила отбора субъектов:

Agent < Instrumental < Objective

---

**Глубинная структура** (в [генеративной лингвистике](#)) — способ представления предложения. Глубинная структура позволяет отразить смысловую близость ряда предложений, которые содержат одни и те же лексические единицы и отличаются друг от друга только некоторыми грамматическими значениями.

В ранней версии [трансформационно-порождающей грамматики Н. Хомского](#) синтаксические трансформации были разделены на **обязательные**, применение которых для **глубинных структур** данного типа обязательно<sup>[2]:105</sup> и не может быть обойдено (без применения трансформации результирующее предложение было бы **неграмматичным**) и **факультативные**<sup>[3]:451–452</sup>, применение которых необязательно в том смысле, что как подлежащая трансформации структура, так и результат преобразования являются соответствуют грамматичным предложениям

## 16. Латентний семантичний аналіз

[https://ru.wikipedia.org/wiki/Латентно-семантический\\_анализ](https://ru.wikipedia.org/wiki/Латентно-семантический_анализ)

[https://uk.wikipedia.org/wiki/Латентно-семантичний\\_аналіз](https://uk.wikipedia.org/wiki/Латентно-семантичний_аналіз)

**Латентно-семантичний аналіз (ЛСА)** — метод обробки інформації природною мовою, зокрема, дистрибутивної семантики<sup>[en]</sup>, що дозволяє аналізувати взаємозв'язок між набором документів і термінами, які в них зустрічаються, шляхом створення набору понять. ЛСА припускає, що слова, близькі за значенням, зустрічатимуться в подібних фрагментах тексту (дистрибутивна гіпотеза).

З великої частини тексту створюється матриця, що вміщує кількість слів на параграф (рядки містять унікальні слова, а стовпці — текст кожного параграфа). За допомогою математичного методу, що називається **сингулярним розкладом матриці**, кількість рядків матриці зменшують, зберігаючи при цьому структуру подібності у стовпцях. Потім слова порівнюють за допомогою обчислення косинуса кута між двома векторами (скалярний добуток векторів, поділений на добуток їх модулів), що утворено будь-якими двома рядками. Значення, близькі до 1, є дуже схожими словами, тоді як значення, близькі до 0, представляють дуже різномірні слова.

ЛСА можно сравнить с простым видом **нейросети**, состоящей из трех слоев: первый слой содержит множество слов (**термов**), второй – некое множество документов, соответствующих определенным ситуациям, а третий, средний, скрытый слой представляет собой

множество узлов с различными весовыми коэффициентами, связывающих первый и второй слои.

В качестве исходной информации ЛСА использует [матрицу термы-на-документы](#), описывающую набор данных, используемый для обучения системы. Элементы этой матрицы содержат, как правило, веса, учитывающие частоты использования каждого терма в каждом документе и участие терма во всех документах ([TF-IDF](#)). Наиболее распространенный вариант ЛСА основан на использовании разложения матрицы по сингулярным значениям ([SVD – Singular Value Decomposition](#)). С помощью SVD-разложения любая матрица раскладывается во множество ортогональных матриц, линейная комбинация которых является достаточно точным приближением к исходной матрице.

Говоря более формально, согласно теореме о сингулярном разложении<sup>[9]</sup>, любая вещественная прямоугольная матрица может быть разложена на произведение трех матриц:

$$A = U S V^T ,$$

где матрицы  $U$  и  $V$  – ортогональные, а  $S$  – диагональная матрица, значения на диагонали которой называются сингулярными значениями матрицы  $A$ .

Такое разложение обладает замечательной особенностью: если в матрице  $S$  оставить только  $k$  наибольших сингулярных значений, а в матрицах  $U$  и  $V$  – только соответствующие этим значениям столбцы, то произведение получившихся матриц  $\hat{S}$ ,  $U$  и  $V$  будет наилучшим приближением исходной матрицы  $A$  к матрице  $\hat{A}$  ранга  $k$ :

$$\hat{A} \approx A = U S V^T ,$$

Основная идея латентно-семантического анализа состоит в том, что если в качестве матрицы  $\mathbf{A}$  использовалась матрица термы-на-документы, то матрица  $\hat{\mathbf{A}}$ , содержащая только  $k$  первых линейно независимых компонент  $\mathbf{A}$ , отражает основную структуру различных зависимостей, присутствующих в исходной матрице. Структура зависимостей определяется весовыми функциями термов.

Таким образом, каждый терм и документ представляются при помощи векторов в общем пространстве размерности  $k$  (так называемом пространстве гипотез). Близость между любой комбинацией термов и/или документов легко вычисляется при помощи скалярного произведения векторов.

Как правило, выбор  $k$  зависит от поставленной задачи и подбирается эмпирически. Если выбранное значение  $k$  слишком велико, то метод теряет свою мощность и приближается по характеристикам к стандартным векторным методам. Слишком маленькое значение  $k$  не позволяет улавливать различия между похожими термами или документами.

Существуют три основных разновидности решения задачи методом ЛСА:

- сравнение двух термов между собой;
- сравнение двух документов между собой;
- сравнение терма и документа.

Достоинства метода:

- метод является наилучшим для выявления латентных зависимостей внутри множества документов;
- метод может быть применен как с обучением, так и без обучения (например, для [кластеризации](#));
- используются значения матрицы близости, основанной на частотных характеристиках документов и лексических единиц;
- частично снимается [полисемия и омонимия](#).

Недостатки:

- Существенным недостатком метода является значительное снижение скорости вычисления при увеличении объема входных данных (например, при SVD-преобразовании). Как показано в [\[10\]](#), скорость вычисления соответствует порядку  $N^{2*k}$ , где  $N = N_{doc} + N_{term}$  - сумма количества документов и термов,  $k$  – размерность пространства факторов.
- Вероятностная модель метода не соответствует реальности. Предполагается, что слова и документы имеют [Нормальное распределение](#), хотя ближе к реальности [Распределение Пуассона](#). В связи с этим для практических применений лучше подходит [Вероятностный латентно-семантический анализ](#), основанный на [мультиномиальном распределении](#).

## 17. Невід'ємна матрична факторизація, алгоритм Lee&Seung

[https://www.google.com/url?  
sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&uact=8&ved=2ahUKEwj7iuG376nfAhXRKCwKHT-kD3wQFjAAegQICRAC&url=http%3A%2F%2Firbis-nbuv.gov.ua%2Fcgi-bin%2Firbis\\_nbuv%2Fcgiirbis\\_64.exe%3FC21COM%3D2%26I21DBN%3DUJRN%26P21DBN%3DUJRN%26IMAGE\\_FILE\\_DOWNLOAD%3D1%26Image\\_file\\_name%3DPDF%2FKNU\\_fiz\\_mat\\_2015\\_4\\_27.pdf&usg=AOvVaw27GRRJ2NwfvZNg9u-O5sKa](https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&uact=8&ved=2ahUKEwj7iuG376nfAhXRKCwKHT-kD3wQFjAAegQICRAC&url=http%3A%2F%2Firbis-nbuv.gov.ua%2Fcgi-bin%2Firbis_nbuv%2Fcgiirbis_64.exe%3FC21COM%3D2%26I21DBN%3DUJRN%26P21DBN%3DUJRN%26IMAGE_FILE_DOWNLOAD%3D1%26Image_file_name%3DPDF%2FKNU_fiz_mat_2015_4_27.pdf&usg=AOvVaw27GRRJ2NwfvZNg9u-O5sKa)

**Розклад невід'ємних матриць**<sup>[1]</sup> (NMF(Non-negative matrix factorization)) це група алгоритмів багатовимірного аналізу та лінійної алгебри, де матриця  $V$  розкладається в, зазвичай, дві матриці  $W$ ,  $H$ , враховуючи, що жодна з трьох матриць немає від'ємних елементів.

Нехай матриця  $V$  є добутком матриці  $W$  і  $H$

$$V = WH.$$

При множенні матриць, розміри матриць-множників можуть бути значно меншими від розмірів, ніж в матриці-результаті і саме ця властивість формує основну властивість NMF-алгоритму. Тобто, NMF генерує матриці-множники, які набагато менші в порівнянні з вихідною матрицею. Наприклад, коли  $V$  розміру  $m \times n$ ,  $W$  –  $m \times p$ , а  $H$  –  $p \times n$ , то  $p$  може бути значно меншим від  $n$  і  $m$ .

Дві прості функції вивчені Лі і Сунгом — це квадрат помилки (або Фробеніусова норма). Як цільова функція використовується норма Фробеніуса, що записується так:

$$\min_{W,H} \|V - WH\|_F^2,$$

причому елементи матриць  $W$  та  $H$  повинні бути невід'ємними.

Для такої цільової функції, та для двох початкових матриць  $W_0$  і  $H_0$ , NMF алгоритм складається з ітераційного виконання двох кроків:

$$(H_k)_{i,j} = (H_{k-1})_{i,j} \times \frac{(W_{k-1}^T V)_{i,j}}{(W_{k-1}^T W_{k-1} H_{k-1})_{i,j}}, \quad (3)$$

$$(W_k)_{i,j} = (W_{k-1})_{i,j} \times \frac{(V H_{k-1}^T)_{i,j}}{(W_{k-1} H_{k-1} H_{k-1}^T)_{i,j}}. \quad (4)$$

На практиці, кроки алгоритму повторюються доки не буде досягнута нерухома точка або доки не буде пройдена максимальна кількість ітерацій. Лі і Сун довели дві основні властивості цього алгоритму: по-перше, цільова функція є монотонно спадною під час застосування наведених правил; по-друге, матриці  $W$  і  $H$  стають постійними, тільки у випадку досягнення стаціонарної точки цільової функції. Крім того, важливо відмітити, що цей метод знаходить тільки локальні мініуми, а не глобальні. Також, важливо ще раз зазначити, що такий розклад не є унікальним.

NMF, як засіб обробки природної мови, має кілька переваг над іншими методами виділення ознак. По-перше, матриці  $W$  і  $H$  мають тільки позитивні елементи, що робить процес їх інтерпретації, у термінах обробки природної мови, простішим і логічнішим. По-друге, стовпці  $W$  не повинні бути ортогональними, а отже виділені теми (ознаки) можуть мати спільні риси, що є досить звичайним для

### Приклад на основі текстового аналізу:

- Нехай матриця  $V$  має 10000 рядків і 500 стовпчиків, де слова є в рядках, а документи - стовпчиками. Тобто є 500 проіндексованих документів по 100000 слів.
- Припустимо, нам потрібно знайти алгоритм, що дає 10 можливостей для того, щоб генерувати матриці  $W$  з 10000 рядків і 10 стовпців і коефіцієнти матриці  $H$  з 10 рядків і 500 стовпців.
- Добуток  $W$  на  $H$  має розмірність 10000 на 500 і, якщо розклад спрацював правильно, елементи якого приблизно рівні елементам вихідної матриці  $V$ .
- З цього випливає, що кожен стовпчик з матриці  $WH$  є лінійною комбінацією векторів 10 стовпців  $W$  і відповідних рядків з матриці  $H$ .

Цей останній пункт є основою NMF, тому що ми можемо розглядати кожен вихідний документ у нашому прикладі, який будується з невеликого набору прихованых функцій. NMF генерує ці функції.

Інколи корисно розглядати вектор стовпчик з властивостей матриці  $W$  як архетип документа, що містить набір слів, де значення комірки кожного слова визначає ранг даного слова у функції: чим вище значення слова, тим вище ранг слова в функції. Стовпчик в коефіцієнтах матриці  $H$  представляє оригінальний документ зі значенням, що визначає значення документа у функції. Це виконується, бо кожен рядок матриці  $H$  представляє функцію(властивість). Тепер ми можемо відновити документ (стовпчик вектор) з заданої матриці

лінійною комбінацією властивостей (вектори-стовпчики в  $W$ , де кожне значення рівне значенню клітинки із стовпчика документу у  $H$ ).

У області обробки природномовних текстів NMF зазвичай застосовується до TD-матриці (матриці терм $\times$ документ). Кожен рядок матриці TD відповідає певному документу з корпусу текстів, а кожен стовпець матриці відповідає певному слову. Якщо задано  $m$  документів і загальна кількість слів у них рівна  $n$ , то розмір TD матриці  $V$  рівний  $m \times n$ . NMF використовується для розкладу матриці  $V$  у добуток двох матриць  $W$  і  $H$  з розмірами  $m \times k$  і  $k \times n$  відповідно, де зазвичай  $k < \min(m, n)$ .

Наприклад,  $k$  може бути встановлено в значення очікуваної кількості кластерів. Часто параметр  $k$  називають «кількість ознак».

Інтуїтивно, параметр  $k$  може бути пояснений наступним чином: TD матриця  $V$  («документи»  $\times$ «слова») розкладається у добуток двох матриць –  $V=WH$ . Матриця  $W$  пов'язує «документи» і «ознаки», а матриця  $H$  пов'язує «ознаки» і «слова». Тому, як випливає з властивостей добутку матриць, кожен документ представляється у вигляді лінійної комбінації виділених ознак. Після цього, традиційні методи кластеризації можуть бути застосовані до рядків матриці  $W$ . Рядки матриці  $W$  можна розглядати як вектори семантичних або тематичних ознак  $m$  документів. Для визначення тематичної або семантичної близькості двох документів можна використовувати косинусну міру, що обчислюється через скалярний добуток векторів ознак документів.

## 18. Невід'ємна тензорна факторізація

Модель невід'ємної факторизації тензорів, у якій зберігається дані про частоту використання різних словосполучень у великих текстових корпусах із врахуванням синтаксичної позиції слів. Факторизовані тензорні лінгвістичні моделі дозволяють автоматизовано виділяти з корпусів текстів опис таких лінгвістичних структур, як селективні преференції (selectional preferences) та субкатегоріальні фрейми дієслів [2], що містять дані про семантичні та синтаксичні властивості зв'язків між дієсловами та їх аргументами-іменниками у реченнях.

Тривимірний тензор для зберігання частотних оцінок сполучностей слів – підметів, присудків та прямих додатків, отриманих в результаті аналізу великих текстових корпусів, зображений на рисунку 1.

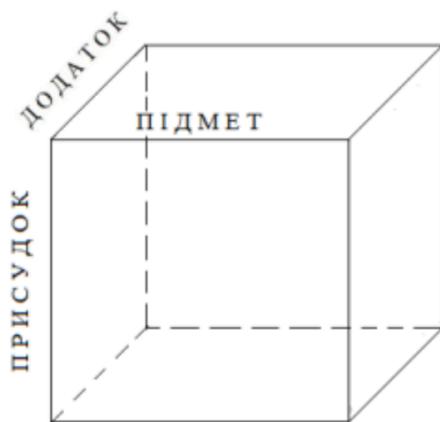


Рисунок 1. Тривимірний тензор для зберігання частотних даних сполучностей слів – підметів, присудків та прямих додатків.

В результаті частотного аналізу текстових корпусів формується розріджений тензор великої розмірності. З метою отримання більш

стислого та зручного представлення до тензору застосовується невід'ємна тензорна факторизація.

Вектори з матриць факторизованого тензору описують комутативні властивості слів.

Основна ідея методу полягає в мінімізації суми квадратів різниць значень між оригіналом тензору та його факторизованою моделлю. Для тривимірного випадку тензору  $T \in R^{D1 \times D2 \times D3}$  це відповідає рівнянню

$$\min_{x_i \in R^{D1}, y_i \in R^{D2}, z_i \in R^{D3}} \| T - \sum_{i=1}^k x_i \circ y_i \circ z_i \|_F^2 , \text{ де } k -$$

число вимірів у факторизованій моделі.

При невід'ємній тензорній факторизації додається обмеження невід'ємності, перетворюючи модель у

$$\min_{x_i \in R_{\geq 0}^{D1}, y_i \in R_{\geq 0}^{D2}, z_i \in R_{\geq 0}^{D3}} \| T - \sum_{i=1}^k x_i \circ y_i \circ z_i \|_F^2$$

В результаті факторизації кожен підмет-іменник, присудок-дієслово та додаток-іменник отримує власний вектор розмірності  $k$  з відповідних матриць.

Оригінальне значення з тензору  $T$  для трійки  $(s, v, o)$   $x_{svo}$  може бути відновлене у факторизованій моделі обчисленням суми

$$x_{svo} = \sum_{i=1}^k s_{si} v_{vi} o_{oi} .$$

## 19.Латентне розміщення Діріхле

Метод латентного размещения Дирихле (latent Dirichlet allocation, LDA) предложен Дэвидом Блеем в 2003 году. В этом методе устранены основные недостатки PLSA.

Метод LDA основан на той же вероятностной модели

$$p(d, w) = \sum_{t \in T} p(d)p(w|t)p(t|d),$$

при дополнительных предположениях:

- векторы документов  $\theta_d = (p(t|d): t \in T)$  порождаются одним и тем же вероятностным распределением на нормированных  $|T|$ -мерных векторах; это распределение удобно взять из параметрического семейства распределений Дирихле  $\text{Dir}(\theta, \alpha)$ ,  $\alpha \in \mathbb{R}^{|T|}$ ;
- векторы тем  $\phi_t = (p(w|t): w \in W)$  порождаются одним и тем же вероятностным распределением на нормированных векторах размерности  $|W|$ ; это распределение удобно взять из параметрического семейства распределений Дирихле  $\text{Dir}(\theta, \beta)$ ,  $\beta \in \mathbb{R}^{|W|}$ .

Для идентификации параметров модели LDA по коллекции документов применяется [самплирование Гиббса, вариационный байесовский вывод](#) или метод [Expectation-Propagation](#).

**Латентное размещение Дирихле** (*LDA*, от англ. *Latent Dirichlet allocation*) — применяемая в [машинаном обучении](#) и [информационном поиске](#) **порождающая модель**, позволяющая объяснять результаты наблюдений с помощью [неявных](#) групп, благодаря чему возможно выявление причин сходства некоторых частей данных. Например, если наблюдениями являются слова, собранные в документы, утверждается, что каждый документ представляет собой смесь небольшого количества тем и что появление каждого слова связано с одной из тем документа

В LDA каждый документ может рассматриваться как набор различных тематик. Подобный подход схож с [вероятностным латентно-семантическим анализом](#) (pLSA) с той разницей, что в LDA предполагается, что распределение тематик имеет в качестве [априори распределения Дирихле](#). На практике в результате получается более корректный набор тематик.

К примеру, модель может иметь тематики классифицируемые как «относящиеся к кошкам» и «относящиеся к собакам», тематика обладает вероятностями генерировать различные слова, такие как «мяу», «молоко» или «котёнок», которые можно было бы

классифицировать как «относящиеся к кошкам», а слова, не обладающие особой значимостью (к примеру, [служебные слова](#)), будут обладать примерно равной вероятностью в различных тематиках.

#### 2.4.2. Латентное размещение Дирихле

Латентное размещение Дирихле (LDA, Latent Dirichlet Allocation) — это порождающая модель, объясняющая результаты наблюдений с помощью неявных групп, что позволяет получить объяснение, почему некоторые части данных схожи. Например, если наблюдениями являются слова, собранные в тексты, утверждается, что каждый текст представляет собой смесь небольшого количества тем и что появление каждого слова связано с одной из тем документа. LDA впервые был представлен в качестве графической модели для обнаружения тем Дэвидом Блеем, Эндрю Нг и Майклом Джорданом в 2002 году [12].

В модели LDA каждый текст генерируется независимо, по следующей схеме [21]:

1. Случайно выбрать для текста его распределение по темам  $\theta_d$
2. Для каждого слова в текста:
  - (a) случайно выбрать тему из распределения  $\theta_d$ , полученного на 1-м шаге.
  - (b) случайно выбрать слово из распределения слов в выбранной теме  $\phi_t$ .

В рассматриваемом наборе текстов  $D$  каждый текст состоит из  $n_d$  слов. Наблюдаемыми переменными являются слова в тексте -  $w_{dn}$ . Все остальные переменные — скрытые. Для каждого текста  $d$  переменная  $\theta_d$  представляет собой распределение тем в данном тексте. В классической модели LDA количество тем фиксировано и изначально задается параметром  $T$ .

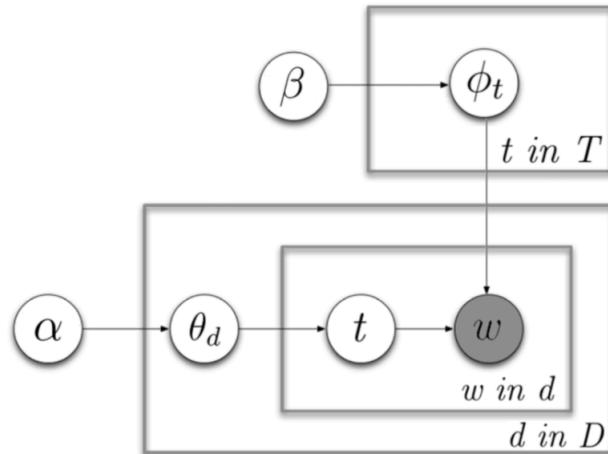


Рис. 2. Графическое представление модели LDA.

В модели LDA предполагается, что параметры  $\theta_d$  и  $\phi_w$  распределены следующим образом:  $\theta \sim Dir(\alpha)$ ,  $\phi \sim Dir(\beta)$ , где  $\alpha$  и  $\beta$  — задаваемые вектора-параметры (т.н. гиперпараметры) распределения Дирихле [21].

Основным недостатком распределения Дирихле является отсутствие убедительных лингвистических обоснований. С точки зрения текстов предположение о распределении Дирихле не является обоснованным.

Также существует множество других тематических моделей, основанных на вышеупомянутых. Выбор модели определяется в зависимости от условий поставленной задачи. Подробно обзор представлен в работе [24].

## 20. Кластерізація. Алгоритми кластерізації

[https://uk.wikipedia.org/wiki/Кластерний\\_аналіз](https://uk.wikipedia.org/wiki/Кластерний_аналіз)

Кластеризація це процес розбиття набору об'єктів на підмножини (кластери), в такий спосіб, що об'єкти всередині кластера більш схожі (в сенсі деяких властивості або характеристик) між собою, ніж до об'єктів з інших кластерах. Існує багато методів кластеризації, більшість з них потрапляють в одну з наступних категорій: ієрархічна, центроїдна (*k-means*, *k-medoids*), статистична (ЕМ-алгоритм), графова (спектральні) кластеризація.

В загальному випадку алгоритми кластеризації отримують на вхід множину точок деякого  $n$ -мірного простору, яке потрібно розділити на кластери так, щоб всередині одного кластера лежали подібні точки. Існує декілька способів знайти міру подібності точок. Точка в досліджуваному просторі представляється як вектор характеристик  $x = (x_1, x_2, \dots, x_n)$ .

**Нечітка кластеризація** — є однією з форм кластеризації, в якій кожна точка може належати більш ніж одному кластеру.

Нечіткий кластер задається функцією приналежності  $z_i = (z_{iI}), i \in I$ , так що величини приналежності  $z_i (0 \leq z_i \leq 1)$  інтерпретуються як ступені приналежності об'єктів  $i$  кластеру (для чітких кластерів значення  $z_i$  можуть бути тільки 1 та 0).

Нечітке розбиття з системою центроїдів — це  $K$  центроїдів  $c_k = (c_{k_1}, c_{k_2}, \dots, c_{k_V})$  в просторі ознак і векторів приналежності  $z_k = (z_{1k}, \dots, z_{ik}, \dots, z_{Nk})$ ,  $0 \leq z_{ik} \leq 1$ , таких що  $\sum_k z_{ik} = 1$  для всіх  $i \in I$ , тобто повна приналежність як би розподілена між кластерами

**Типовими кластерними моделями є:**

- **Моделі зв'язності.** Наприклад, ієрархічна кластеризація або таксономія будуються на основі відстані між вузлами.
- **Центроїдні моделі.** Наприклад, метод **K-середніх** (K-means) представляє кожен кластер єдиним усередненим вектором.
- **Статистичні моделі.** Кластери будуються ґрунтуючись на статистичних розподілах. Таких як **багатовимірний нормальний розподіл** з допомогою **ЕМ-алгоритму**.
- **Моделі засновані на щільності.** Наприклад, в **DBSCAN** і в **OPTICS** кластери визначаються як зв'язані області відповідної щільності у просторі даних.
- **Групові моделі.** Деякі алгоритми не забезпечують вдосконалену модель для своїх результатів, а просто описують групування об'єктів.
- **Графові моделі.** Поняття **кліки** (така підмножина вершин, в якій кожна пара вершин з'єднана ребром) у **графі** слугує прототипом кластеру. Пом'якшення вимоги до повної зв'язності (тобто, частина ребер може бути відсутня) призводить до поняття відомого як квазі-кліка. Вони будуються алгоритмом **HCS<sup>[en]</sup>**.
- **Нейронні моделі.** Найбільш відомою моделлю нейронної мережі з навчанням без учителя є **нейронна мережа Кохонена**. Ці моделі, як правило, можна охарактеризувати як схожі на одну або

подібні якісь з наведених вище моделей, включаючи моделі у підпросторах, коли нейронні мережі реалізують [метод головних компонент](#) або [аналіз незалежних компонент](#)<sup>[en]</sup>.

«Кластеризацією» зазвичай вважають такий набір кластерів, які містять усі об'єкти набору даних. Додатково, можна розглянути [відношення](#) між кластерами. Наприклад, ієрархію вкладеності кластерів один у одного. Грубо можна виділити такі кластеризації:

- **Жорстка кластеризація.** Кожен об'єкт або належить кластеру або ні.
- **М'яка кластеризація** (також [нечітка кластеризація](#)). Кожен об'єкт належить кожному кластеру до певної міри. Наприклад, це ймовірність належності кластеру.

---

Для заданого набору з  $m$  текстових документів, метою кластеризації за темами є знаходження такого поділу документів на групи, що документи з одного кластера мають спільну тему. Важливо відзначити, що список тем, не є відомим до початку процесу кластеризації.

Традиційні методи кластеризації документів часто засновані на векторно-частотній моделі (Vector Space Model – VSM). У цій моделі документ представляється як вектор частот слів. Нехай  $T = \{t_1, \dots, t_n\}$  – набір слів з усіх документів. Тоді частотний вектор  $X_i$  документа  $d_i$  формується як  $X_i = \{x_{1i}, \dots, x_{ni}\}$ , причому

$$x_{ij} = f_{ij} \log \frac{m}{c_j},$$

де  $f_{ij}$  позначає частоту терміну  $t_j$  у документі  $d_i$ . Кількість документів у яких присутній терм  $t_j$ , позначається як  $c_j$ .

Таким чином, VSM використовує слова, як міру схожості документів. Після введення такої моделі, очевидним чином визначаються метрики на частотних векторах, наприклад, косинусна відстань або Евклідова метрика. Завдяки представленню документів у термінах лінійної алгебри, до них можуть бути застосовані традиційні методи кластеризації.

Одним з основних недоліків VSM, є припущення, про те що слова є статистично незалежними між собою. Однак, це часто не так для реальних текстів. Теми, концепції та семантика є ключовими ознаками документа. Для отримання таких особливостей з текстів були розроблені спеціальні методи, відомі під загальною назвою "виділення ознак" (feature extraction). Метою таких методів є виділення основних концепцій (або тем) текстів, та подання документів у вигляді їх комбінації. Такі методи успішно застосовуються у задачах обробки природно мовних текстів, таких як: кластеризація документів, знаходження семантичної відстані тощо.

## 21. Алгоритм k-means

[https://ru.wikipedia.org/wiki/Метод\\_k-средних](https://ru.wikipedia.org/wiki/Метод_k-средних)

Мета методу — розділити  $n$  спостережень на  $k$  кластерів, так щоб кожне спостереження належало до кластера з найближчим до нього **середнім значенням**. Метод базується на мінімізації суми квадратів відстаней між кожним спостереженням та центром його кластера, тобто функції

$$\sum_{i=1}^N d(x_i, m_j(x_i))^2,$$

де  $d$  — метрика,  $x_i$  —  $i$ -ий об'єкт даних, а  $m_j(x_i)$  — центр кластера, якому на  $j$ -ій ітерації приписаний елемент  $x_i$ .

---

Принцип алгоритму полягає в пошуку таких центрів кластерів та наборів елементів кожного кластера при наявності деякої функції  $\Phi(\cdot)$ , що виражає якість поточного розбиття множини на  $k$  кластерів, коли сумарне квадратичне відхилення елементів кластерів від центрів цих кластерів буде найменшим:

$$V = \sum_{i=1}^k \sum_{x_j \in S_i} (x_j - \mu_i)^2$$

де  $k$  — число кластерів,  $S_i$  — отримані кластери,  $i = 1, 2, \dots, k$ ,  $\mu_i$  — центри мас векторів  $x_j \in S_i$ .

---

У методі k-середніх процес породження розбиття з центроїдами починається з деяких початкових точок — центроїд (експериментально показано, що найкраще, якщо це об'єкти множини, яку ми аналізуємо, а не абстрактні розташування), а потім здійснюється послідовність ітерацій. Кожна ітерація складається з двох кроків:

1. *Оновлення кластерів:* При заданих  $K$  центроїдах  $c_k$  ( $k = 1, 2, \dots, K$ ), кожний об'єкт  $i \in I$  приписується до одного з центроїдів згідно з так званим правилом мінімальної відстані: обчислюються відстані від  $i$  до  $c_k$ , після чого  $i$  віходить до найближчого з  $c_k$  (рисунок 2(b)). Об'єкти, приписані до  $c_k$ , утворюють оновлений кластер  $S_k$  ( $k = 1, 2, \dots, K$ ) (рисунок 2(c)).

2. *Оновлення центроїдів:* Для кожного кластера  $S_k$  обчислюється його центр ваги (внутрішньокластерне середнє за кожною ознакою), який і оголошується новим центроїдом  $c_k'$  ( $k=1,2,\dots,K$ ) (рисунок 2(d))

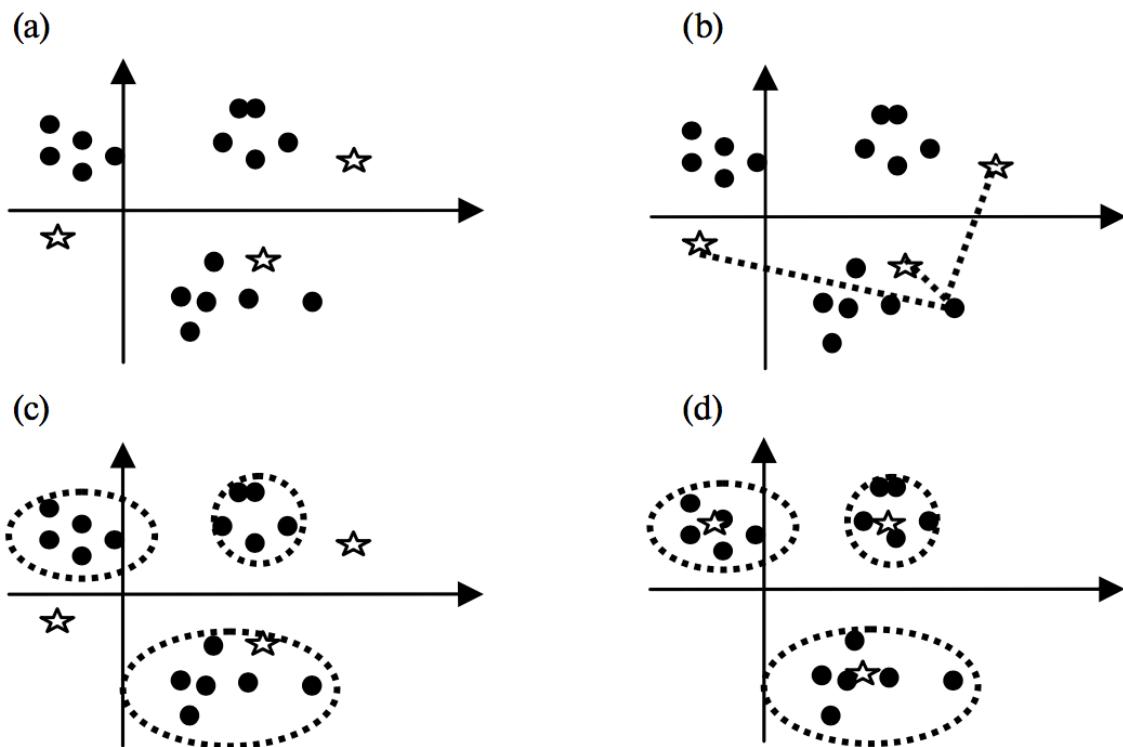


Рисунок 2. Основні кроки паралельного методу k-середніх: (а) ініціалізація центроїдів, (б) оновлення кластерів (пунктиром показані відстані від об'єкта до центру ваги), (с) закінчення процесу оновлення кластерів, (д) закінчення процесу оновлення центроїдів

Процес методу k-середніх оновлює центроїди так, щоб вони потрапили в місця відносно високої щільності об'єктів. З іншого боку, його можна інтерпретувати як процес формування типологій, де центроїди служать моделями «типових представників». Цей метод працює швидко і може бути організований з економією пам'яті, а також в розподіленому режимі.

## Переваги

Головні переваги методу k-середніх — його простота та швидкість виконання. Метод k-середніх більш зручний для кластеризації великої кількості спостережень, ніж метод ієрархічного кластерного аналізу (у якому дендограмами стають перевантаженими і втрачають наочність).

## Недоліки

Одним із недоліків простого методу є порушення умови зв'язності елементів одного кластера, тому розвиваються різні модифікації методу, а також його нечіткі аналоги ([англ.](#) *fuzzy k-means methods*), у яких на першій стадії алгоритму допускається приналежність одного елемента множини до декількох кластерів (із різним ступенем приналежності).

Незважаючи на очевидні переваги методу, він має суттєві недоліки:

1. Результат класифікації сильно залежить від випадкових початкових позицій кластерних центрів
2. Алгоритм чутливий до викидів, які можуть викривлювати середнє
3. Кількість кластерів повинна бути заздалегідь визначена дослідником

## 22. Моделі та методи машинного навчання:

- **Naive Bayes**

<https://ru.wikipedia.org/wiki/>

### Наївний байесовський класифікатор

- **Linear Regression**
- **Logistic Regression**
- **Maximum Entropy**

- **Naive Bayes**

**Наївний баєсів класифікатор** — ймовірнісний класифікатор, що використовує теорему Баєса для визначення ймовірності приналежності спостереження (елемента вибірки) до одного з класів при припущені (наївному) незалежності змінних.

**Теорема Баєса.** Теорема Баєса задається математично наступним рівнянням

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)},$$

де  $A$  та  $B$  є подіями.

$P(A)$  — априорна ймовірність гіпотези  $A$ .

$P(A|B)$  — ймовірність гіпотези  $A$ , якщо настала подія  $B$  (апостеріорна ймовірність).

$P(B|A)$  — ймовірність настання події  $B$ , якщо гіпотеза  $A$  істина.

$P(B)$  — повна ймовірність настання події  $B$ .

Вероятностная модель для классификатора — это условная модель

$$\begin{aligned}
p(C, F_1, \dots, F_n) &= \\
&= p(C) p(F_1, \dots, F_n | C) = \\
&= p(C) p(F_1 | C) p(F_2, \dots, F_n | C, F_1) = \\
&= p(C) p(F_1 | C) p(F_2 | C, F_1) p(F_3, \dots, F_n | C, F_1, F_2) = \\
&= p(C) p(F_1 | C) p(F_2 | C, F_1) \cdot \dots \cdot p(F_n | C, F_1, F_2, F_3, \dots, F_{n-1})
\end{aligned}$$

используя повторные приложения определений [условной вероятности](#):

$$p(C | F_1, \dots, F_n) = \frac{p(C) p(F_1, \dots, F_n | C)}{p(F_1, \dots, F_n)}.$$

и т. д. Теперь можно использовать «наивные» предположения [условной независимости](#): предположим, что каждое свойство  $F_i$  условно независимо от любого другого свойства  $F_j$  при  $j \neq i$ . Это означает:

$$p(F_i | C, F_j) = p(F_i | C)$$

таким образом, совместная модель может быть выражена как:

$$\begin{aligned}
p(C, F_1, \dots, F_n) &= p(C) p(F_1 | C) p(F_2 | C) p(F_3 | C) \cdot \dots \cdot p(F_n | C) = \\
&= p(C) \prod_{i=1}^n p(F_i | C).
\end{aligned}$$

Это означает, что из предположения о независимости, условное распределение по классовой переменной  $C$  может быть выражено так:

$$p(C | F_1, \dots, F_n) = \frac{1}{Z} p(C) \prod_{i=1}^n p(F_i | C)$$

где  $Z$  — это масштабный множитель, зависящий только от  $F_1, \dots, F_n$ , то есть константа, если значения переменных известны.

- **Linear Regression**

[https://ru.wikipedia.org/wiki/Линейная\\_регрессия](https://ru.wikipedia.org/wiki/Линейная_регрессия)

**Линейная регрессия** — используемая в [статистике](#) регрессионная модель зависимости одной (объясняемой, зависимой) переменной  $y$  от другой или нескольких других переменных (факторов, регрессоров, независимых переменных)  $x$  с [линейной функцией](#) зависимости.

## Регрессионная модель

$$y = f(x, b) + \varepsilon, E(\varepsilon) = 0,$$

где  $b$  — параметры модели,  $\varepsilon$  — случайная ошибка модели; называется линейной регрессией, если функция регрессии  $f(x, b)$  имеет вид

$$f(x, b) = b_0 + b_1 x_1 + b_2 x_2 + \dots + b_k x_k,$$

где  $b_j$  — параметры (коэффициенты) регрессии,  $x_j$  — регрессоры (факторы модели),  $k$  — количество факторов модели<sup>[1]</sup>.

Коэффициенты линейной регрессии показывают скорость изменения зависимой переменной по данному фактору, при фиксированных остальных факторах (в линейной модели эта скорость постоянна):

$$\forall j \quad b_j = \frac{\partial f}{\partial x_j} = \text{const}$$

Параметр  $b_0$ , при котором нет факторов, называют часто константой. Формально — это значение функции при нулевом значении всех факторов. Для аналитических целей удобно считать, что константа — это параметр при «факторе», равном 1 (или другой произвольной постоянной, поэтому константой называют также и этот «фактор»). В таком случае, если перенумеровать факторы и параметры исходной модели с учетом этого (оставив обозначение общего количества факторов —  $k$ ), то линейную функцию регрессии можно записать в следующем виде, формально не содержащем константу:

$$f(x, b) = b_1 x_1 + b_2 x_2 + \dots + b_k x_k = \sum_{j=1}^k b_j x_j = x^T b,$$

где  $x^T = (x_1, x_2, \dots, x_k)$  — вектор регрессоров,  $b = (b_1, b_2, \dots, b_k)^T$  — вектор-столбец параметров (коэффициентов).

Линейная модель может быть как с константой, так и без константы. Тогда в этом представлении первый фактор либо равен единице, либо является обычным фактором соответственно.

## Матричное представление [ [править](#) | [править код](#) ]

Пусть дана [выборка](#) объёмом  $n$  наблюдений переменных  $y$  и  $x$ . Обозначим  $t$  — номер наблюдения в выборке. Тогда  $y_t$  — значение переменной  $y$  в  $t$ -м наблюдении,  $x_{tj}$  — значение  $j$ -го фактора в  $t$ -м наблюдении. Соответственно,  $x_t^T = (x_{t1}, x_{t2}, \dots, x_{tk})$  — вектор регрессоров в  $t$ -м наблюдении. Тогда линейная регрессионная зависимость имеет место в каждом наблюдении:

$$y_t = b_1 x_{t1} + b_2 x_{t2} + \dots + b_k x_{tk} = \sum_{j=1}^k b_j x_{tj} = x_t^T b + \varepsilon_t, \quad E(\varepsilon_t) = 0, \quad t = 1..n$$

Введём обозначения:

$$y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} \quad \text{— вектор наблюдений зависимой переменной } y$$

$$X = \begin{pmatrix} x_{11} & x_{12} & \dots & x_{1k} \\ x_{21} & x_{22} & \dots & x_{2k} \\ \vdots & & & \\ x_{n1} & x_{n2} & \dots & x_{nk} \end{pmatrix} \quad \text{— матрица факторов.}$$

$$\varepsilon = \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{pmatrix} \quad \text{— вектор случайных ошибок.}$$

Тогда модель линейной регрессии можно представить в матричной форме:

$$y = Xb + \varepsilon$$

## • Logistic Regression

[https://ru.wikipedia.org/wiki/Логистическая\\_регрессия](https://ru.wikipedia.org/wiki/Логистическая_регрессия)

[https://uk.wikipedia.org/wiki/Логістична\\_регресія](https://uk.wikipedia.org/wiki/Логістична_регресія)

**Логістична регресія** (англ. *logistic regression*) або **логіт-регресія** (англ. *logit model*<sup>[1]</sup>) — статистичний регресійний метод, що застосовують у випадку, коли залежна змінна є категорійною, тобто може набувати тільки двох значень (чи, загальніше, скінченної множини значень).

Логистическая **регистрация** применяется для предсказания вероятности возникновения некоторого события по значениям множества признаков. Для этого вводится так называемая **зависимая переменная**  $y$ , принимающая лишь одно из двух значений — как правило, это числа 0 (событие не произошло) и 1 (событие произошло), и множество **независимых переменных** (также называемых признаками, предикторами или регрессорами) — **вещественных**  $x_1, x_2, \dots, x_n$ , на основе значений которых требуется вычислить вероятность принятия того или иного значения зависимой переменной. Как и в случае **линейной регрессии**, для простоты записи вводится фиктивный признак  $x_0 = 1$ .

Делается предположение о том, что вероятность наступления события  $y = 1$  равна:

$$\mathbb{P}\{y = 1 | x\} = f(z),$$

где  $z = \theta^T x = \theta_0 + \theta_1 x_1 + \dots + \theta_n x_n$ ,  $x$  и  $\theta$  — **векторы-столбцы** значений независимых переменных  $1, x_1, \dots, x_n$  и параметров (коэффициентов регрессии) — вещественных чисел  $\theta_0, \dots, \theta_n$ , соответственно, а  $f(z)$  — так называемая **логистическая функция** (иногда также называемая **сигмоидом** или **логит-функцией**):

$$f(z) = \frac{1}{1 + e^{-z}}.$$

Так как  $y$  принимает лишь значения 0 и 1, то вероятность принять значение 0 равна:

$$\mathbb{P}\{y = 0 | x\} = 1 - f(z) = 1 - f(\theta^T x).$$

Для краткости **функцию распределения**  $y$  при заданном  $x$  можно записать в таком виде:

$$\mathbb{P}\{y | x\} = f(\theta^T x)^y (1 - f(\theta^T x))^{1-y}, \quad y \in \{0, 1\}.$$

Фактически, это есть **распределение Бернулли** с параметром, равным  $f(\theta^T x)$ .

- **Maximum Entropy**

[https://ru.wikipedia.org/wiki/Принцип\\_максимума\\_энтропии](https://ru.wikipedia.org/wiki/Принцип_максимума_энтропии)

[https://studopedia.ru/9\\_152147\\_printsip-maksimuma-entropii.html](https://studopedia.ru/9_152147_printsip-maksimuma-entropii.html)

**Принцип максимума энтропии** утверждает, что наиболее характерными **распределениями вероятностей** состояний неопределенной среды являются такие распределения, которые максимизируют выбранную **меру неопределенности** при заданной информации о «поведении» среды.

**Принцип максимума энтропии** утверждает, что если плотность распределения некоторой случайной величины неизвестна, то из логических соображений следует выбрать такую плотность распределения, которая обеспечивает **максимизацию энтропии случайной величины при учете всех известных ограничений**. Применение этого критерия приводит к решению, отличающемуся минимальным смещением, так как плотность распределения любого другого вида будет обладать большим смещением «в сторону» информации, содержащейся в известном наборе данных. Плотность распределения, обеспечивающую максимум энтропии, особенно легко определять в тех случаях, когда все известные ограничения представлены в форме средних оценок, таких, например, как математические ожидания и дисперсии плотности распределения.

Энтропия совокупности образов с плотностью распределения  $p(x)$  определяется как

$$H = - \int p(x) \ln p(x) dx \quad (1)$$

где  $p(x)$  - это плотность распределения  $p(x|w_i)$ .

Пусть априорная информация о случайной величине  $\mathbf{x}$  задается в виде:

$$\int_x p(x) dx = 1 \quad (*)$$

и

$$\int_x b_k(x) p(x) dx = a_k, k = 1, 2, \dots, Q \quad (**)$$

**Постановка задачи:** необходимо так задать плотности распределения  $p(\mathbf{x})$ , чтобы величина энтропии при выполнении ограничений (\*) и (\*\*), которые являются априорной информацией, была максимальной.

## **23. Приховані марковські моделі:**

[https://uk.wikipedia.org/wiki/Прихована\\_марковська\\_модель](https://uk.wikipedia.org/wiki/Прихована_марковська_модель)

- **Viterbi algorithm**
- **Forward-backward algorithm**
- **Baum-Welch algorithm**

**Прихована марковська модель, ПММ** (англ. *hidden Markov model, HMM*) — це статистична марковська модель, у якій система, що моделюється, розглядається як марковський процес із неспостережуваними (прихованими) станами.

У простіших марковських моделях (таких як ланцюги Маркова) стан є безпосередньо видимим спостерігачеві, і тому ймовірності переходу станів є єдиними параметрами. У прихованій марковській моделі стан не є видимим безпосередньо, але вихід, залежний від стану, видимим є. Кожен стан має ймовірнісний розподіл усіх можливих вихідних значень. Отже, послідовність символів, згенерована ПММ, дає якусь інформацію про послідовність станів.

- **Viterbi algorithm**

[https://neerc.ifmo.ru/wiki/index.php?title=Алгоритм\\_Витерби](https://neerc.ifmo.ru/wiki/index.php?title=Алгоритм_Витерби)

[https://ru.wikipedia.org/wiki/Алгоритм\\_Витерби](https://ru.wikipedia.org/wiki/Алгоритм_Витерби)

**Алгоритм Витерби** — алгоритм поиска наиболее подходящего списка состояний (называемого *путём Витерби*), который в контексте цепей Маркова получает наиболее вероятную последовательность произошедших событий.

Алгоритм Витерби позволяет сделать наиболее вероятное предположение о последовательности состояний [скрытой Марковской модели](#) на основе последовательности наблюдений.

Пусть у нас есть скрытая марковская модель с пространством состояний  $S$ , начальными вероятностями  $\pi_i$  нахождения в состоянии  $i$  и вероятностями  $a_{i,j}$  перехода из состояния  $i$  в состояние  $j$ . Пусть мы наблюдаем на выходе  $y_1, \dots, y_T$ . Тогда наиболее вероятная последовательность состояний  $x_1, \dots, x_T$  задается рекуррентными соотношениями:<sup>[2]</sup>

$$\begin{aligned} V_{1,k} &= P(y_1 \mid k) \cdot \pi_k \\ V_{t,k} &= P(y_t \mid k) \cdot \max_{x \in S} (a_{x,k} \cdot V_{t-1,x}) \end{aligned}$$

Здесь  $V_{t,k}$  — это вероятность наиболее вероятной последовательности состояний, ответственная за появление первых  $t$  наблюдаемых символов, завершающаяся в состоянии  $k$ . Путь Витерби может быть найден при помощи указателей, запоминающих, какое состояние  $x$  появлялось во втором уравнении. Пусть  $\text{Ptr}(k, t)$  — функция, которая возвращает значение  $x$ , использованное для подсчета  $V_{t,k}$ , если  $t > 1$ , или  $k$  если  $t = 1$ . Тогда

$$\begin{aligned} x_T &= \arg \max_{x \in S} (V_{T,x}) \\ x_{t-1} &= \text{Ptr}(x_t, t) \end{aligned}$$

## Входные данные:

1. Пространство наблюдений  $O = \{o_1, o_2 \dots o_N\}$
  2. Пространство состояний  $S = \{s_1, s_2 \dots s_K\}$
  3. Последовательность наблюдений  $Y = \{y_1, y_2 \dots y_T\}$
  4. Матрица  $A$  переходов из  $i$ -того состояния в  $j$ -ое, размером  $K \times K$
  5. Матрица эмиссии  $B$  размера  $K \times N$ , которая определяет вероятность наблюдения  $o_j$  из состояния  $s_i$
  6. Массив начальных вероятностей  $\pi$  размером  $K$ , показывающий вероятность того, что начальное состояние  $s_i$

## **Выходные данные:**

$X = \{x_1, x_2 \dots x_T\}$  – последовательность состояний, которые привели к последовательности наблюдений  $Y$ .

- Forward-backward algorithm

[https://ru.wikipedia.org/wiki/Алгоритм\\_прямого- обратного\\_хода](https://ru.wikipedia.org/wiki/Алгоритм_прямого-обратного_хода)

<https://neerc.ifmo.ru/wiki/index.php?title=Алгоритм %22Вперед-Назад%22>

**Алгоритм «прямого-обратного» хода** — алгоритм для вычисления **апостериорных вероятностей** последовательности состояний при наличии последовательности наблюдений. Иначе говоря, алгоритм, вычисляющий вероятность специфической последовательности наблюдений.

**Алгоритм включает три шага:**

1. вычисление прямых вероятностей
2. вычисление обратных вероятностей
3. вычисление сглаженных значений

Далее будем рассматривать в качестве базовой матрицы эмпирическую матрицу вероятностных значений, а не распределения вероятности. Мы преобразовываем распределения вероятности, связанные с данной скрытой Марковской моделью в матричный вид следующим образом. Матрица переходных вероятностей  $P(X_t|X_{t-1})$  (для данной случайной переменной  $X_t$ , представляющая все возможные состояния в скрытой марковской модели, будет представлена матрицей  $T$ . В этой матрице индекс строки  $i$  обозначает начальное состояние, а индекс столбца  $j$  - конечное состояние. Например, ниже представлена система, для которой вероятность оставаться в том же состоянии после каждого шага равна 70 %, а вероятность перейти к другому состоянию равна 30 %. Тогда матрица вероятностей переходов выглядит следующим образом:  $T = \begin{pmatrix} 0.7 & 0.3 \\ 0.3 & 0.7 \end{pmatrix}$

Точно так же мы представим вероятности новых состояний для данных наблюдаемых состояний, заданных как свидетельство, в матрице наблюдений  $O_t$ , где каждый диагональный элемент содержит вероятность нового состояния, учитывая наблюдаемое состояния в момент  $t$ . Отметим, что  $t$  указывает специфическое наблюдение в последовательности наблюдений. Все другие элементы в матрице будут нулями. В примере, описанном ниже, первое наблюдаемое доказательство ( $t = 1$ ) — «зонтик». Поэтому  $O_1$  был бы определен как:  $O_1 = \begin{pmatrix} 0.9 & 0.0 \\ 0.0 & 0.2 \end{pmatrix}$

Исходя из этого описания мы можем вычислить следующую прямую вероятность. Пусть набор прямых вероятностей будет сохранён в ёщё одной матрице  $f_{1:t+1}$ . Здесь  $1 : t + 1$  указывает на то, что вычисленные вероятности зависят от всех прямых вероятностей от 1 до  $t + 1$ , включая текущую матричную вероятность, которую мы опишем как  $f_{1:t}$ . Следовательно,  $f_{1:t+1}$  равно произведению транспонированной матрицы с текущими прямыми вероятностями и матрицей наблюдения для следующего свидетельства в потоке наблюдения. После этого получается матрица, которая требует нормализации, то есть полученные значения должны быть разделены на сумму всех значений в матрице. Коэффициент нормализации задан  $\alpha$ . Вычисление прямых вероятностей описано формулой:  $f_{1:t+1} = \alpha O_{t+1} T^T f_{1:t}$

Можем представить вычисление обратной вероятности  $b_{k+1:t}$ , которое начинается с конца последовательности аналогичным способом. Пусть конец последовательности будет описан индексом  $k$ , начинающийся с 0. Поэтому выполнение от  $k$  к  $t = 0$  и вычисляя каждую обратную вероятность может быть описано следующей формулой:  $b_{k+1:t} = T O_{k+1} b_{k+2:t}$

Отметьте, что мы используем не транспонированную матрицу  $T$  и что значение элементов изменилось. Также отметим, что в качестве окончательного результата мы не используем обычное матричное произведение, а поточечное. Эта операция умножает каждую переменную в одной матрице с соответствующей переменной в другой. Третий и конечный шаг — это вычисление сглаженных вероятностей  $sv_k$ . Сглаженные вероятности полученные поточечным произведением Формула определена как  $sv_k = \alpha b_{k+1:t} f_{1:k}$ . Ниже показан следующий пример.

- **Baum-Welch algorithm**

[https://ru.wikipedia.org/wiki/Алгоритм\\_Баума\\_—\\_Велша](https://ru.wikipedia.org/wiki/Алгоритм_Баума_—_Велша)

[https://neerc.ifmo.ru/wiki/index.php?title=Алгоритм\\_Баума-Велша](https://neerc.ifmo.ru/wiki/index.php?title=Алгоритм_Баума-Велша)

**Алгоритм Баума — Велша** используется в **информатике** и **статистике** для нахождения неизвестных параметров **скрытой марковской модели** (HMM). Он использует **алгоритм прямого-обратного хода** и является частным случаем обобщённого ЕМ-алгоритма.

## Алгоритм Баума — Велша оценки скрытой модели Маркова [ править | править код ]

Скрытая модель Маркова — это **вероятностная модель** множества случайных переменных  $\{O_1, \dots, O_t, Q_1, \dots, Q_t\}$ . Переменные  $O_t$  — известные дискретные наблюдения, а  $Q_t$  — «скрытые» дискретные величины. В рамках скрытой модели Маркова есть два независимых утверждения, обеспечивающих сходимость данного алгоритма:

1.  $t$ -я скрытая переменная при известной  $(t-1)$ -ой переменной независима от всех предыдущих  $(t-1)$  переменных, то есть  
$$P(Q_t | Q_{t-1}, O_{t-1}, \dots, Q_1, O_1) = P(Q_t | Q_{t-1}) ;$$
2.  $t$ -е известное наблюдение зависит только от  $t$ -го состояния, то есть не зависит от времени,  $P(O_t | Q_t, Q_{t-1}, O_{t-1}, \dots, Q_1, O_1) = P(O_t | Q_t) .$

Далее будет предложен алгоритм «предположений и максимизаций» для поиска максимальной вероятностной оценки параметров скрытой модели Маркова при заданном наборе наблюдений. Этот алгоритм также известен как алгоритм Баума — Велша.

$Q_t$  — это дискретная случайная переменная, принимающая одно из  $N$  значений  $(1 \dots N)$ . Будем полагать, что данная модель Маркова, определённая как  $P(Q_t | Q_{t-1})$ , однородна по времени, то есть независима от  $t$ . Тогда можно задать  $P(Q_t | Q_{t-1})$  как независящую от времени **стохастическую матрицу** перемещений  $A = \{a_{ij}\} = p(Q_t = j | Q_{t-1} = i)$ . Особый случай для времени  $t = 1$  определяется начальным распределением  $\pi_i = P(Q_1 = i)$ .

Будем считать, что мы в состоянии  $j$  в момент времени  $t$ , если  $Q_t = j$ . Последовательность заданных состояний определяется как  $q = (q_1, \dots, q_T)$ , где  $q_t \in \{1 \dots N\}$  является состоянием в момент  $t$ .

Наблюдение может иметь одно из  $L$  возможных значений,  $O_t \in \{o_1, \dots, o_L\}$ . Вероятность заданного вектора наблюдений в момент времени  $t$  для состояния  $j$  определяется как  $b_j(o_t) = P(O_t = o_t | Q_t = j)$  ( $B = \{b_{ij}\}$  — это матрица  $L$  на  $N$ ). Заданная последовательность наблюдений  $O$  выражается как  $O = (O_1 = o_1, \dots, O_T = o_T)$ .

Следовательно, мы можем описать скрытую модель Маркова с помощью  $\lambda = (A, B, \pi)$ . При заданном векторе наблюдений  $O$  алгоритм Баума — Велша находит  $\lambda^* = \max_{\lambda} P(O | \lambda)$ .  $\lambda$  максимизирует вероятность наблюдений  $O$ .

**Исходные данные:**  $\lambda = (A, B, \pi)$  со случайными начальными условиями.

Алгоритм итеративно обновляет параметр  $\lambda$  до сходжения в одной точке.

### Прямая процедура [править | править код]

Определим  $\alpha_i(t) = p(O_1 = o_1, \dots, O_t = o_t, Q_t = i \mid \lambda)$ , что является вероятностью получения заданной последовательности  $o_1, \dots, o_t$  для состояния  $i$  в момент времени  $t$ .

$\alpha_i(t)$  можно вычислить рекурсивно:

1.  $\alpha_i(1) = \pi_i \cdot b_i(O_1);$
2.  $\alpha_j(t+1) = b_j(O_{t+1}) \sum_{i=1}^N \alpha_i(t) \cdot a_{ij}.$

### Обратная процедура [править | править код]

Данная процедура позволяет вычислить  $\beta_i(t) = p(O_{t+1} = o_{t+1}, \dots, O_T = o_T \mid Q_t = i, \lambda)$  вероятность конечной заданной последовательности  $o_{t+1}, \dots, o_T$  при условии, что мы начали из исходного состояния  $i$ , в момент времени  $t$ .

Можно вычислить  $\beta_i(t)$ :

1.  $\beta_i(T) = p(Q_t = i, \lambda) = 1;$
2.  $\beta_i(t) = \sum_{j=1}^N \beta_j(t+1) a_{ij} b_j(O_{t+1}).$

Используя  $\alpha$  и  $\beta$  можно вычислить следующие значения:

- $\gamma_i(t) \equiv p(Q_t = i \mid O, \lambda) = \frac{\alpha_i(t) \beta_i(t)}{\sum_{j=1}^N \alpha_j(t) \beta_j(t)},$
- $\xi_{ij}(t) \equiv p(Q_t = i, Q_{t+1} = j \mid O, \lambda) = \frac{\alpha_i(t) a_{ij} \beta_j(t+1) b_j(o_{t+1})}{\sum_{i=1}^N \sum_{j=1}^N \alpha_i(t) a_{ij} \beta_j(t+1) b_j(O_{t+1})}.$

Имея  $\gamma$  и  $\xi$ , можно определить:

- $\bar{\pi}_i = \gamma_i(1),$
- $\bar{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_{ij}(t)}{\sum_{t=1}^{T-1} \gamma_i(t)},$
- $\bar{b}_i(k) = \frac{\sum_{t=1}^T \delta_{O_t, o_k} \gamma_i(t)}{\sum_{t=1}^T \gamma_i(t)}.$

Используя новые значения  $A$ ,  $B$  и  $\pi$ , итерации продолжаются до сходжения.

## 24. Генетичний алгоритм

[https://uk.wikipedia.org/wiki/Генетичний\\_алгоритм](https://uk.wikipedia.org/wiki/Генетичний_алгоритм)

Генетичний алгоритм — це еволюційний алгоритм пошуку, що використовується для вирішення задач оптимізації і моделювання шляхом послідовного підбору, комбінування і варіації шуканих параметрів з використанням механізмів, що нагадують біологічну еволюцію.

Особливістю генетичного алгоритму є акцент на використання оператора «схрещення», який виконує операцію рекомбінацію рішень-кандидатів, роль якої аналогічна ролі схрещення в живій природі.

Задача кодується таким чином, щоб її вирішення могло бути представлено в вигляді масиву подібного до інформації складу хромосоми. Цей масив часто називають саме так «хромосома». Випадковим чином в масиві створюється деяка кількість початкових елементів «осіб», або початкова популяція. Особи оцінюються з використанням функції пристосування, в результаті якої кожній особі присвоюється певне значення пристосованості, яке визначає можливість виживання особи. Після цього з використанням отриманих значень пристосованості вибираються особи, допущені до схрещення (селекція). До осіб застосовується «генетичні оператори» (в більшості випадків це оператор схрещення (crossover) і оператор мутації (mutation)), створюючи таким чином наступне покоління осіб. Особи наступного покоління також оцінюються застосуванням генетичних операторів і виконується селекція і мутація. Так моделюється еволюційний процес, що продовжується декілька життєвих циклів

(поколінь), поки не буде виконано критерій зупинки алгоритму. Таким критерієм може бути<sup>[\[джерело 2\]](#)</sup>:

- знаходження глобального, або надоптимального вирішення;
- вичерпання числа поколінь, що відпущені на еволюцію;
- вичерпання часу, відпущеного на еволюцію.

**Можна виділити такі етапи генетичного алгоритму:**

1. Створення початкової популяції:
2. Обчислення функції пристосованості для осіб популяції (оцінювання)
3. Повторювання до виконання критерію зупинки алгоритму:
  1. Вибір індивідів із поточної популяції (селекція)
  2. Схрещення або/та мутація
  3. Обчислення функції пристосуваності для всіх осіб
  4. Формування нового покоління

## 25. Нейронно-мережевий підхід

- CNN
  - RNN
- 
- **CNN (convolutional neural network)**

[https://ru.wikipedia.org/wiki/Свёрточная\\_нейронная\\_сеть](https://ru.wikipedia.org/wiki/Свёрточная_нейронная_сеть)

[https://uk.wikipedia.org/wiki/Згорткова\\_нейронна\\_мережа](https://uk.wikipedia.org/wiki/Згорткова_нейронна_мережа)

Згорткові нейронні мережі (ЗНМ, англ. *convolutional neural network*, *CNN*, *ConvNet*) в машинному навчанні — це клас глибинних штучних нейронних мереж прямого поширення.

Ідея сверточных нейронных сетей заключается в чередовании сверточных слоёв (англ. *convolution layers*) и субдискретизирующих слоёв (англ. *subsampling layers* или англ. *pooling layers*, слоёв подвыборки).

ЗНМ використовують різновид багатошарових перцептронів, розроблений так, щоби вимагати використання мінімального обсягу **попередньої обробки**.

ЗНМ складається з шарів входу та виходу, а також із декількох **прихованіх шарів**. Приховані шари ЗНМ зазвичай складаються зі згорткових шарів, агрегувальних шарів, повноз'єднаних шарів та шарів нормалізації.

В сверточной нейронной сети в операции свёртки используется лишь ограниченная матрица весов небольшого размера, которую 40 из 50

«двигают» по всему обрабатываемому слою (в самом начале — непосредственно по входному изображению), формируя после каждого сдвига сигнал активации для нейрона следующего слоя с аналогичной позицией. То есть для различных нейронов выходного слоя используются одна и та же матрица весов, которую также называют **ядром свёртки**.

Операция субдискретизации (англ. *subsampling*, англ. *pooling*, также переводимая как «операция подвыборки» или операция объединения), выполняет уменьшение размерности сформированных карт признаков. За счёт данной операции, помимо ускорения дальнейших вычислений, сеть становится более инвариантной к масштабу входного изображения.

- **RNN (Recurrent neural network)**

[https://ru.wikipedia.org/wiki/Рекуррентная\\_нейронная\\_сеть](https://ru.wikipedia.org/wiki/Рекуррентная_нейронная_сеть)

[https://uk.wikipedia.org/wiki/Рекурентна\\_нейронна\\_мережа](https://uk.wikipedia.org/wiki/Рекурентна_нейронна_мережа)

**Рекурентні нейронні мережі (РНМ, англ. *recurrent neural networks*, *RNN*)** — це клас **штучних нейронних мереж**, у якому з'єднання між вузлами утворюють **орієнтований цикл**. Це створює внутрішній стан мережі, що дозволяє їй проявляти динамічну поведінку в часі. На відміну від **нейронних мереж прямого поширення**, РНМ можуть використовувати свою внутрішню пам'ять для обробки довільних послідовностей входів.

Щоби мінімізувати загальну **похибку**, може застосовуватися **градієнтний спуск** для зміни кожної ваги пропорційно **похідній** похибки

по відношенню до цієї ваги, за умови, що нелінійні функції активації є диференційовними.

Рекурсивные нейронные сети представляют собой более общий случай рекуррентных сетей, когда сигнал в сети проходит через структуру в виде дерева (обычно бинарные деревья).<sup>[16]</sup> Те же самые матрицы весов используются рекурсивно по всему графу в соответствии с его топологией.<sup>[17][18]</sup> Рекурсивные нейронные сети находят применение в задачах обработки естественного языка.<sup>[19]</sup> Существуют также тензорные рекурсивные нейронные сети (RNTN, Recursive Neural Tensor Network), которые используют тензорные функции для всех узлов в дереве.

Сеть с долговременной и кратковременной памятью (англ. *Long short term memory*, LSTM) представляет собой систему глубинного обучения, при реализации которой удалось обойти проблему исчезновения или зашкаливания градиентов в процессе обучения методом обратного распространения ошибки. Сеть LSTM обычно модерируется с помощью рекуррентных вентилей, которые называются вентили (gates) «забывания».<sup>[30]</sup> Ошибки распространяются назад по времени через потенциально неограниченное количество виртуальных слоёв. Таким образом происходит обучение в LSTM<sup>[31]</sup>, при этом сохраняя память о тысячах и даже миллионах временных интервалов в прошлом.

## Методи глобальної оптимізації

Тренування ваг у нейронній мережі можливо моделювати як нелінійну задачу [глобальної оптимізації](#)<sup>[en]</sup>. Цільову функцію для оцінки пристосованості або похибки певного вагового вектора може бути сформовано таким чином: Спершу ваги в мережі встановлюються відповідно до цього вагового вектора. Далі, мережа оцінюється за тренувальною послідовністю. Як правило, для представлення похибки поточного вагового вектора використовують суму квадратів різниць між передбаченнями та цільовими значеннями, вказаними в тренувальній послідовності. Потім для мінімізації цієї цільової функції може бути застосовано довільні методики глобальної оптимізації.

Найуживанішим методом глобальної оптимізації для тренування РНМ є [генетичні алгоритми](#), особливо в неструктуртованих мережах.<sup>[67][68]</sup>  
<sup>[69]</sup>

Спочатку генетичний алгоритм кодується вагами нейронної мережі в наперед визначеному порядку, коли один ген у хромосомі представляє одне зважене з'єднання, і так далі; вся мережа представляється єдиною хромосомою. Функція пристосованості обчислюється наступним чином: 1) кожна вага, закодована в хромосомі, призначається відповідному зваженню мережі; 2) потім тренувальний набір зразків представляється мережі, яка поширює вхідні сигнали далі; 3) до функції пристосованості повертається середньоквадратична похибка; 4) ця функція потім веде процес генетичного відбору.

Популяцію складають багато хромосом; таким чином, багато різних нейронних мереж еволюціють, поки не буде досягнуто критерію зупинки. Пошириеною схемою зупинки є: 1) коли нейронна мережа

засвоїла певний відсоток тренувальних даних, або 2) коли досягнуто мінімального значення середньоквадратичної похибки, або 3) коли було досягнуто максимального числа тренувальних поколінь. Критерій зупинки оцінюється функцією пристосованості при отриманні нею оберненого значення середньоквадратичної похибки з кожної з нейронних мереж під час тренування. Отже, метою генетичного алгоритму є максимізувати функцію пристосованості, знизивши таким чином середньоквадратичну похибку.

Для пошуку доброго набору ваг можуть застосовуватися й інші методики глобальної (та/або еволюційної) оптимізації, такі як [імітація відпалу](#) та [метод рою часток](#).

**Додаткові питання:**

- Предельная факторизация матриц
- Неотъемлемая факторизация тензор

## Naive Bayes classifier (Наївний бассів класифікатор)

**Наївний бассів класифікатор** — ймовірнісний класифікатор, що використовує теорему Баєса для визначення ймовірності приналежності спостереження (елемента вибірки) до одного з класів при припущені (наївному) незалежності змінних.

У теорії ймовірностей дві випадкові події називаються **незалежними**, якщо настання однієї з них не змінює вірогідність настання іншої. Аналогічно, дві випадкові величини називають **незалежними** якщо значення однієї з них не впливає на розподіл значень іншої.

Вважатимемо, що дано фіксований ймовірнісний простір  $(\Omega, \mathcal{F}, P)$ .

**Означення.** Дві події  $A, B \in \mathcal{F}$  називають **незалежними**, якщо

$$P(A \cap B) = P(A) \cdot P(B).$$

**Зауваження.** В тому випадку, якщо ймовірність однієї події, скажемо  $B$  ненульова, тобто  $P(B) > 0$ , визначення незалежності еквівалентне:

$$P(A|B) = P(A),$$

тобто умовна ймовірність події  $A$  за умови  $B$  дорівнює безумовній вірогідності події  $A$ .

**Теорема Баєса.** Теорема Баєса задається математично наступним рівнянням

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)},$$

де  $A$  та  $B$  є подіями.

$P(A)$  — априорна ймовірність гіпотези  $A$ .

$P(A|B)$  — ймовірність гіпотези  $A$ , якщо настала подія  $B$  (апостеріорна ймовірність).

$P(B|A)$  — ймовірність настання події  $B$ , якщо гіпотеза  $A$  істина.

$P(B)$  — повна ймовірність настання події  $B$ .

Тобто, якщо на основі значень змінних можна однозначно визначити, до якого класу належить спостереження, байєсів класифікатор повідомить ймовірність приналежності до цього класу.

У проміжних же випадках, коли спостереження може з різною ймовірністю належати до різних класів, результатом роботи класифікатора буде вектор, компоненти якого є ймовірностями приналежності до того чи іншого класу.

## Баєсова фільтрація спаму

**Баєсова фільтрація спаму** (англ. *Naive Bayes spam filtering*) — метод для фільтрації спаму, заснований на застосуванні наївного баєсова класифікатора, що спирається на пряме використання теореми Баєса.

Під час навчання фільтру для кожного слова в тексті вираховують та зберігають його «**вагу**» — оцінку ймовірності того, що текст із цим словом — спам. У найпростішому випадку як оцінку використовують частоту: «появ в спамі / появ всього». У складніших випадках можлива попередня обробка тексту: приведення слів до початкової форми, видалення службових слів, обчислення «ваги» для цілих фраз, транслітерація тощо.

Під час перевірки нового тексту ймовірність «спаму» обчислюють за вказаною вище формулою для множини гіпотез. В даному випадку «гіпотези» — це слова, і для кожного слова «достовірність гіпотези»

$$P(A_i) = \frac{N_{word_i}}{N_{words\ total}} \text{ — частка цього слова в тексті,}$$

а «залежність події від гіпотези»  $P(B|A_i)$  — обчислена раніше «вага» слова.

Тобто «вага» тексту в даному випадку — усереднена «вага» всіх його слів.

Віднесення тексту до «спаму» чи «не-спаму» проводиться в залежності від того, чи перевищує його «вага» якусь планку, задану користувачем (зазвичай беруть 60-80%). Після ухвалення рішення стосовно тексту в базі даних оновлюються «ваги» для слів, що входять до його складу.

Поштові байесовські фільтри ґрунтуються на теоремі Байеса. Теорема Байєса використовується кілька разів в контексті спаму:

- в перший раз, щоб обчислити ймовірність, що повідомлення — спам, знаючи, що дане слово з'являється в цьому повідомленні;
- вдруге, щоб обчислити ймовірність, що повідомлення — спам, враховуючи всі його слова (або відповідні їх підмножини);
- іноді в третій раз, коли зустрічаються повідомлення з рідкісними словами.

## Обчислення ймовірності того, що повідомлення, що містить дане слово, є спамом

Припустимо, що підозрюване повідомлення містить слово «Replica». Більшість людей, які звикли отримувати електронного листа, знає, що це повідомлення, швидше за все, буде спамом, а точніше — пропозицією продати підроблені копії годинників відомих брендів. Програма виявлення спаму, однак, не «знає» такі факти; все, що вона може зробити — обчислити ймовірності.

Формула, яка використовується програмним забезпеченням, щоб визначити це, отримана з теореми Байеса і формули повної ймовірності:

$$\Pr(S|W) = \frac{\Pr(W|S) \cdot \Pr(S)}{\Pr(W)} = \frac{\Pr(W|S) \cdot \Pr(S)}{\Pr(W|S) \cdot \Pr(S) + \Pr(W|H) \cdot \Pr(H)},$$

де

- $\Pr(S|W)$  — умовна ймовірність того, що **повідомлення-спам**, за умови, що слово «Replica» знаходитьться в ньому;
- $\Pr(S)$  — повна ймовірність того, що довільне **повідомлення-спам**;
- $\Pr(W|S)$  — умовна ймовірність того, що слово «replica» з'являється в повідомленнях, якщо вони є **спамом**;
- $\Pr(H)$  — повна ймовірність того, що довільне повідомлення **не спам** (тобто «ham»);
- $\Pr(W|H)$  — умовна ймовірність того, що слово «replica» з'являється в повідомленнях, якщо вони є **не спам** («ham»).

## Спамовість слова

Недавні статистичні дослідження показали, що на сьогоднішній день ймовірність будь-якого повідомлення бути спамом становить щонайменше 80%:  
 $\Pr(S) = 0.8$ ;  $\Pr(H) = 0.2$ .

Однак більшість байесовських програм виявлення спаму роблять припущення про відсутність априорних переваг у повідомлення бути «spam», а не «ham», і вважає, що у обох випадків є рівні ймовірності 50%:  
 $\Pr(S) = 0.5$ ;  $\Pr(H) = 0.5$ .

Про фільтри, які використовують цю гіпотезу, кажуть як про фільтри «без упереджень». Це означає, що у них немає ніякого упередження щодо вхідних

повідомлень електронної пошти. Дане припущення дозволяє спрощувати загальну формулу до:

$$\Pr(S|W) = \frac{\Pr(W|S)}{\Pr(W|S) + \Pr(W|H)}.$$

Значення  $\Pr(S|W)$  — називають «спамовістю» слова  $W$ ; при цьому число  $\Pr(W|S)$ , що використовується в наведеній вище формулі, наближено дорівнює відносній частоті повідомлень, які містять слово  $W$  в повідомленнях, ідентифікованих як спам під час фази навчання, тобто:

$$\Pr(W_i|S) = \frac{\text{count}(M : W_i \in M, M \in S)}{\sum_j \text{count}(M : W_j \in M, M \in S)}.$$

Так само  $\Pr(W|H)$  наближено дорівнює відносній частоті повідомлень, що містять слово  $W$  в повідомленнях, ідентифікованих як «ham» під час фази навчання.

$$\Pr(W_i|H) = \frac{\text{count}(M : W_i \in M, M \in H)}{\sum_j \text{count}(M : W_j \in M, M \in H)}.$$

Для того, щоб ці наближення мали сенс, набір навчальних повідомлень повинен бути великим. Також бажано, щоб набір навчальних повідомлень відповідав 50% гіпотезі про перерозподіл між спамом і «ham», тобто що набори повідомлень «spam» і «ham» мали один і той же розмір.

Звичайно, визначення, чи є повідомлення «spam» або «ham», що базується тільки на присутності лише одного певного слова, схильне до помилок. Саме тому байесовські фільтри спаму намагаються розглянути кілька слів і комбінувати їх спамовість, щоб визначити повну ймовірність того, що повідомлення є спамом.