Київський національний університет імені Тараса Шевченка

МЕТОДИЧНІ ВКАЗІВКИ
до курсу “Алгоритми машинного навчання”

для студентів механіко-математичного факультету

Видавничо-поліграфічний центр
’Київський університет’
2021

Рецензенти:
д-р фіз.-мат.наук, доц. І.В. Розора,
д-р фіз.-мат.наук, доц. О.І. Василик

Рекомендовано до друку вченою радою механіко-математичного факультету
протокол N2 від 16.09.2021 року

МЕТОДИЧНІ ВКАЗІВКИ курсу "Алгоритми машинного навчання " / Упорядник: В.В. Голомозий- К., Видавничо-поліграфічний центр 'Київський університет', 2021 - 28 с.

Навчальне видання

МЕТОДИЧНІ ВКАЗІВКИ
до курсу "Алгоритми машинного навчання"

для студентів механіко-математичного факультету

Упорядник

Голомозий Віталій Вікторович

Зміст

Виконання завдань

Методичні вказівки до курсу "Алгоритми машинного навчання " призначені для студентів III курсу бакалаврату механіко-математичного факультету спеціальностей "Математика", "Статистика".

Зміст та структура матеріалу відповідає програмі курсу та вимогам кредитно - модульної системи організації навчального процесу.

Розділи посібника містять завдання для лабораторних робіт.

Кожен розділ методичних вказівок розбито на дві частини

Теоретичні відомості та **Завдання для лабораторної роботи**

Завдання з розділу **Завдання для лабораторної роботи** виконуються під час лабораторних занять а також самостійно в позаадутирний час за бажанням студента. Завдання виконуються на підставі прослуханих лекцій, отриманих у аудиторії, прикладів продемонстрованих під час занять, та поточних консультацій.

Якість виконання вказаних завдань контролюється викладачами, оцінюється у балах, та враховується згідно кредитно-модульній системі при оцінюванні рівня знань студентів.

Змістовно матеріал посібника розбитий на теми відповідно до модуля.

1. Регресійні задачі

1. Задача регресії та оцінка наймешних квадратів.

Призначення: Навчитися будувати ОНК, та виконувати регресійний аналіз в середовищі R або python.

2. Методи регуляризації та гребенева регресія.

Призначення: Навчитися рахувати оцінку гребеневої регресії.

2. Задачі класифікації

4. Базові алгоритми класифікації

Призначення: Навчитися використовувати алгоритми k-nn, логістичну регресію, наївний баєсівський класифікатор.

5. Опорна машина векторів.

Призначення: Опанувати використання опорної машини векторів, розуміти та вміти застосовувати ядерні перетворення.

6. Моделі засновані на деревах.

Призначення: Використання класифікаційних дерев та алгоритмів ансамблювання заснованих на деревах.

3. Матрична факторизація

7. Ймовірнісна матрична факторизація

Призначення: Навчитися використовувати ймовірнісну матричну факторизацію для побудови рекомендаційних систем.

8. Факторизація невід'ємних матриць.

Призначення: Навчитися використовувати факторизацію невід'ємних матриць у задач зменшення розмірності.

4. Нейронні мережі

9. Щільні нейронні мережі

Призначення: Зрозуміти основні засади побудови нейронних мереж.

10. Згорткові нейронні мережі

Призначення: Навчитися будувати згорткові нейронні мережі за допомогою існуючих бібліотек.

Література

1. *T. Hastie, R. Tibshirani, J. Friedman*, The Elements of statistical learning: Data mining, Inference , and Prediction, Springer Series in Statistics.

2. *G. James, D. Witten, T. Hastie, R. Tibshirani*, An introduction to Statistical Learning: with applications in R, Springer Texts in Statistics.

3. *J. Faraway*, Practical regression and anova using R

4. *Р.Є. Майборода*, Регресія, лінійні моделі, ВПЦ «Київський університет», 2007

1. Регресійні задачі

Теоретичні відомості

Задача регресії та оцінка наймешних квадратів. Нехай маємо вибірку, що складається з $(x_i, y_i), i = 1, N$, де $x_i \in \mathbb{R}^{1\times p}$, $y_i \in \mathbb{R}$.

Число N - називається розміром вибірки, p - кількістю регресорів, x_i - спостереженням, $x^j \in \mathbb{R}^{N\times 1}$ - регресорами, y_i - відгуками.

Матриця дизайну:

$$X = \begin{pmatrix} x_1^1 & x_1^2 & x_1^3 \dots x_1^p \\ x_2^1 & x_2^2 & x_2^3 \dots x_2^p \\ \dots & \dots & \dots \\ x_N^1 & x_N^2 & x_N^3 \dots x_N^p \end{pmatrix} \tag{1.1}$$

Рівняння регресії,

$$Y = X\beta + \varepsilon, \tag{1.2}$$

де

$$Y = (y_i)_{i=1,N}^T \in \mathbb{R}^{N\times 1},\ X \in \mathbb{R}^{N\times p},\ \beta \in \mathbb{R}^{p\times 1},\ \varepsilon \in \mathbb{R}^{N\times 1}.$$

Тут ε_i - це похибка прогнозування i-того відгуку, а ε - це вектор похибок. Ми вважаємо, що ε - це випадковий вектор. Тепер, нашу задачу можна сформулювати так: знайти "найкраще" β яке мінімізує ε.

$\hat{\beta} \in \mathbb{R}^{p\times 1}$ - це оцінка найменших квадратів.

Методи регуляризації та гребенева регресія. Для регуляризації використовують таку задачу оптимізації

$$J(\beta) = ||Y - X\beta||^2 + \lambda||\beta||^2 \to \min, \tag{1.3}$$

Її розв'язком є

$$\hat{\beta}_{RR} = (X^T X + \lambda I_p)^{-1} X^T Y.$$

Це оцінка гребеневої регресії.

Завдання для лабораторних робіт

Загальні відомості

Всього передбачено 12 варіантів.

Архів з даними слід отримати у викладача, кожному варіанту відповідає один файл. У кожному варіанті буде вказана змінна - відгук, а також змінні - регресори. Пояснення до даних можна знайти у файлі DataDescriptions.pdf. Самі дані взяті з сайту, що відповідає книзі: Edward W.: Regression Modeling with Actuarial and Financial Applications.

Для проведення **регресійного** аналізу потрбіно зробити наступне:

1) Побудувати ОНК, зробити висновки, щодо якості моделі та ОНК.

2) Спробувати покращити оцінку, шляхом використання гребеневої регресії.

3) Спробуйте покращити оцінку додавши у модель нелінійність.

4) З'ясуйте чи можна зменшити кількість регресорів без суттєвої шкоди для моделі.

Також пропонується додаткове завдання 5. Завдання 5 є опціональним і виконується за бажанням студента.

5) Напишіть функцію, що примає число N, а також величини $\beta_i,\ i = 0 : 3$ як параметри. Функція повинна згенерувати три регресора $X_i,\ i = 1,3$ зі стандартним нормальним розподілом, а також похибку з розподілом $\mathcal{N}(0, 0.1)$ розмірності N. Функція повинна повернути дата фрейм з колонкою $Y = \beta_0+\beta_1X_1+\beta_2X_2+\beta_3X_3+\varepsilon$, а також колонками $X_1,\ X_2,\ X_3$. Після цього обчисліть ОНК використовуючи безпосереднью формулу, бібліотечну функцію або за допомогою власної імплементації методу градієнтного спуску (теж оформленого в окрему функцію). Для різних значень параметра N занотуйте час який потрібен для обчислення ОНК кожним із трьох методів. Знайдіть таке N при якому один із методів буде працювати відчутно повільніше. Для чистоти експерименту проведіть його 10 разів для кожного N.

Вимоги до виконання, оформлення та здачі.

Роботу можна виконувати в R або Python. Вибірку слід розбити на тестову та тренувальну у пропорціх 20% та 80%.

Робота має бути оформлена у вигляді .pdf (можливо .doc/.docx) файлу який містить всю необхідну інформацію. Роботи потрібно здавати на парах або надсилати на email.

Робота повинна містити код, та його інтерпритацію. Обов'язково має бути вказано:
1. ОНК, наявність кореляцій між регресорами, якість ОНК (її дисперсія, якість прогнозування, наведена діаграма залишків, коефіцієнт детермінації та результати тесту Фішера).
2. Підбір параметра λ, покращення (чи відсутність покращення) якості прогнозу, зменшення дисперсії.
3. Висновок щодо ефективності нелінійної моделі.
4. Метод що використовувася для оптимального відбору, та досягнуті результати.

Кожна робота буде розглядатися на відповідність критеріям описаним вище, та на обгрунтованість прийнятих рішень. Кожен студент, повинен виконати свою роботу самостійно. Ідентичні, або майже ідентичні роботи прийматися до уваги не будуть.

Варіанти

Варіант 1
Файл з даними: Chicago.csv
Відгук: `theft`
Регресори: Всі окрім `zipcode`

Варіант 2
Файл з даними: CeoCompensation.csv
Відгук: `COMP`
Регресори: `TENURE, EXPER, SALES, VAL, PCNTOWN, PROF`

Варіант 3
Файл з даними: HealthExpend.csv
Відгук: `EXPENDOP`
Регресори: `AGE,famsize,COUNTIP,COUNTOP,EXPENDIP`

Варіант 4
Файл з даними: NAICExpense.csv
Відгук: `EXPENSES`
Регресори: `RBC, STAFFWAGE, AGENTWAGE, LONGLOSS, SHORTLOSS`

Варіант 5
Файл з даними: NAICExpense.csv
Відгук: EXPENSES
Регресори: GPWPERSONAL, GPWCOMM, ASSETS, CASH, LIQUIDRATIO

Варіант 6
Файл з даними: HospitalCosts.csv
Відгук: TOTCHG
Регресори: AGE, LOS, APRDRG

Варіант 7
Файл з даними: RiskSurvey.csv
Відгук: FIRMCOST
Регресори: ASSUME, SIZELOG, INDCOST, CENTRAL, SOPH

Варіант 8
Файл з даними: UNLifeExpectancy.csv
Відгук: LIFEEXP
Регресори: ILLITERATE POP FERTILITY PRIVATEHEALTH HEALTHEXPEND BIRTHATTEND PHYSICIAN GDP
Коментар: В даному файлі деякі дані пропущені. Запропонуйте варіант розв'язання цієї проблеми.

Варіант 9
Файл з даними: WiscHospCosts.csv
Відгук: TOT_CHG
Регресори: NO_DSCHG POPLN NUM_BEDS INCOME CHG_NUM

Варіант 10
Файл з даними: WiscLottery.csv
Відгук: SALES
Регресори: PERPERHH MEDSCHYR MEDHVL PRCRENT PRC55P HHMEDAGE MEDINC POP

Варіант 11
Файл з даними: Medicare.csv
Відгук: COV_CHG
Регресори: TOT_CHG MED_REIB TOT_D NUM_DCHG AVE_T_D

Варіант 12

Файл з даними: MedCPISmooth.csv
Відгук: `value`
Регресори: `PerMEDCPI YEAR MCPISM4 MCPISM8 MCPISMw_2 MCPISMw_-8`

2. Задачі класифікації

Теоретичні відомості

2.1. Базові алгоритми класифікації. Логістична регресія Розглянемо задачу бінароної класифікації. Нехай маємо вибірку (x_i, y_i), $i = 1, N$, $x_i \in \mathbb{R}^{1\times p}$, $y_i \in \{0, 1\}$. Чи можна її розв'язати за допомогою лінійної регресії?

Розглянемо "шанси" (odds):

$$\frac{P\{y_i = 1; x_i, \beta\}}{P\{y_i = 0; x_i, \beta\}} = \beta_0 + \sum_{k=1}^{p} \beta_k x_i^k + \varepsilon_i.$$

Відношення шансів приймає значеня від 0 до ∞, тоді як права частина може приймати значення, що сягають $-\infty$. Тому візьмемо від лівої частини логарифм:

$$\ln \frac{P\{y_i = 1; x_i, \beta\}}{1 - P\{y_i = 1; x_i, \beta\}} = \beta_0 + \sum_{k=1}^{p} \beta_k x_i^k + \varepsilon_i.$$

Оскільки самі значення ймовірностей ми не спостерігаємо, отже потреби у ε_i немає (насправді ми наближаємо мат. сподівання, а середнє значення $\varepsilon_i = 0$ тож його можна і потрібно прибрати у всіх формулах). Тоді отримали модель:

$$P\{y_i = 1; x_i, \beta\} = g(x_i \cdot \beta),$$

де ми додали одиничний регресор для вільного члена, і функція g має вигляд:

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

Для обчислення оцінки використовують наближені методи. Має місце наступна рекурентна формула:

$$\beta^{(m+1)} := \beta^{(m)} + \alpha X^T (Y - g(X \cdot \beta^{(m)})).$$

Метод класифікації під назвою **k**-найближчих сусідів ($k-nn$) полягає в тому, що для заданої точки знаходять k точок із навчаючої вибірки (вибірки з точок для яких відома приналежність до певного класу), і класифікують задану точку до тієї групи представників якої найбільше серед обраних k сусідів.

Метод k-найближчих сусідів може бути адаптовано до "невпевного" вибору, шляхом введення другого параметру, такий метод називається (k,l)-правило. Полягає воно в тому що класифікація відбувається якщо за "переможний" клас "проголосувало" не менше ніж l точок з навчаючої вибірки, в іншому разі приймається рішення $\mathcal{D}$. Така модель застосовується у задачах з неоднаковими втратами при неправильній класифікації, і тому надає можливість "меношсті" впливати на вибраний клас.

Двома основними параметрами $k-nn$ класифікатора є власне вибір числа k а також вибір метрики. Ми поговоримо про вплив цих гіперпараметрів пізніше.

Наївний баєсівський класифікатор, є одним із найпростіших генеративних алгоритмів. Його суть полягає у "наївному припущенні":

$$p_k(x_1,\ldots,x_p)=\prod_{j=1}^{p}p_k(x_j).$$

Тоді апостеріорна щільність для класу k матиме вигляд:

$$p(k|x)=\frac{\pi_k p_k(x)}{\sum\limits_{k=1}^{K}\pi_l p_l(x)}\propto \pi_k\prod_{j=1}^{p}p_k(x_j).$$

Тобто, для того, щоб класифікувати об'єкт нам потрібно оцінити величини $p_k(x_i)=P\{X_i|C=K\}$.

2.2. Опорна машина векторів.

Припустимо наш тренувальний набір даних містить N пар (x_i,y_i), де $x_i\in\mathbb{R}^p$, а $y_i\in\{-1,1\}$. Визначимо гіперплощину:

$$\{x:\ f(x)=x\cdot\beta+\beta_0=0\},$$

де β це одиничний вектор, $||\beta||=1$. Класифікаційне правило, створене $f(x)$ виглядає таким чином:

$$G(x)=\text{sign}(x\cdot\beta+\beta_0).$$

За допомогою $f(x)$ можна визначити знакозмінну відстань з точки x до гіперплощини $f(x) = x \cdot \beta + \beta_0 = 0$. Оскільки класи лінійно роздільні то можливо знайти функцію $f(x) = x \cdot \beta + \beta_0$, таку що $y_i f(x_i) > 0,\ \forall i$. Таким чином можливо знайти гіперплощину що створює найбільший відступ між тренувальними точками для класів 1 та -1.

Для початку нагадаємо трохи алгебри. Гіперплощина в $\mathbb{R}^p$ задається формулою $x \cdot \beta + \beta_0 = 0$. Тут β це вектор нормалі, перпендикулярний до гіперплощини. Якщо вздовж вектору β відкласти відстань β_0 отримаємо "новий нуль" - x_0. Для довільної точки $x \in \mathbb{R}^p$ відстань від x до гіперплощини задається формулою:

$$d(L, x) = \beta \cdot (x - x_0) = \frac{1}{||\beta||}(x \cdot \beta + \beta_0),$$

для довільного $x' \in L$: $x' \cdot \beta = -\beta_0$.

Наша задача знайти таку пряму, щоб відстань від усіх тренувальних точок до прямої була максимальною (поки розглядаємо випадок лінійно роздільних даних). Позначимо цю відстань $2M$. Дана проблема може буте перефразована більш зручно:

$$\begin{cases} \min_{\beta,\beta_0} ||\beta|| \\ y_i(x_i^T \beta + \beta_0) \geq 1,\ i = 1, \ldots, N \end{cases} \tag{2.1}$$

Задача розв'язується чисельними методами. Існує варіант опорної машини векторів який допускає потрапляння точок у "неправльну" площину для незначної кількості точок. Розглянуті вище моделі є лінійними. Як і в задачах регресії їх можна зробити нелінійними додавши додаткові ознаки, такі як степені початкових даних, або сплайни. Так, для кожного x_i можна розглянути функцію:

$$h(x_i) = (h_1(x_i), \ldots, h_m(x_i)),$$

і звичайний класифікатор:

$$G(x) = \text{sign}(h(x) \cdot \beta + \beta_0).$$

Зауважимо, що задачу оптимізації яку ми розглядали раніше, можна переформулювати лише у вигляді скалярних добутків,

так функція L_D може бути зображена у вигляді:

$$L_D = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j < h(x_i), h(x_j) >,$$

розв'язок цієї задачі може бути представлений у вигляді:

$$f(x) = h(x) \cdot \beta + \beta_0 = \sum_{i=1}^{N} \alpha_i y_i < h(x), h(x_i) > + \beta_0.$$

Таким чином бачимо, що знати всю функцію $h(x)$ непотрібно, а потрібно знати лише скалярні добутки, або ядерну функцію:

$$K(x, x') =< h(x), h(x') > .$$

Щоб функція $K(x, y)$ могла зображатись у вигляді скалярного добутку вона повинна бути симетричною, додатньо напівизначеною.

Популярними ядерними функціями є:

Поліноміальне ядро, степені d $K(x, x') = (1+ < x, x' >)^d$,
Радіальне (Гаусове) ядро $K(x, x') = e^{-\gamma ||x-x'||^2}$,
Тангенс гіперболічний $K(x, x') = tanh(k_1 < x, x' > +k_2)$. (2.2)

2.3. Алгоритми засновані на деревах. Ідея методів заснованих на деревах полягає у поділі простору значень на набір прямокутників, і потім натренувати просту просту модель у кожному прямокутнику. Подібні моделі дуже прості, але тим не менше демонструють гарні результати на певних типах задач. Побудова дерева є складною алгоритмічною задачею, тому для цього слід використовувати бібліотечні функції.

Ансеблеві (ensemble) алгоритми полягають у тому, щоб натренувати декілька класифікаторів для вирішення однієї і тієї ж проблеми. Такі алгоритми називають також - алгоритмами заснованими на голосуванні (committee-based algorithm), або навчанням системи з багатьма класифікаторами.

Типовим прикладом такого підходу є Boosting, та алгоритм AdaBoost, який дозволяє натренувати декілька дуже простих алгоритмів (як правило вирішуючих "кущів" або коренів - це дерево з одним вузлом) які працюють лише трохи краще за випадковий вибір та однак є досить "дешевими" в тренуванні. В

результаті комбінації великої кількості прострих алгоритмів вдається побудувати класифікатор який демонструє значні показники ефективності і навіть перевершує гарно натреновані складні алгоритми.

Зауважимо, що в групових алгоримах можуть використовуватись будь-які алгортими, включаючи досить складні, такі як нейронні мережі.

Існує два фундаментально різних підходи до ансамблювання - **послідовний**, коли наступний алгоритм вибирається з урахуванням результатів попереднього, та **паралельний** коли декілька алагоритмів тренуються одночасно.

Завдання для лабораторних робіт

Загальні відомості

Всього передбачено 12 варіантів.

Кожному варіанту відповідає два файли - один з розширенням .data - містить дані, другий з розширенням .names, містить інформацію про колонки. У кожному варіанті буде вказана змінна - відгук, в якості регресорів слід узяти всі числові змінні. Самі дані взяті з сайту https://archive.ics.uci.edu/ml/index.php (це досить відомий сайт, що містить багато інформації та датасетів для машинного навчання).

Для проведення **класифікації** потрбіно зробити наступне:

1) Опрацювати дані на предмет пропущених значень. Якщо можливо то видалити рядки із пропущеними значеннями, якщо ж їх занадто багато то замінити пропущені значення середніми по регресору.

2) Побудувати наївний байєсівський класифікатор.

3) Провести класифікацію методом k-nn, спробуйте різні значення n.

4) Провести класифікацію за допомогою логістичної регресії.

5) Провести класифікацію за допомогою svm, підберіть різні ядра та гіперпараметри.

6) Провести класифікацію за допомогою дерев, boosting та random forest. Спробуйте різні конфігурації.

7) Додайте до моделі факторні дані. Для цього можна скористатися різними техніками: присвоїти кожному фактору деяке число і проводити звичайний аналіз (для яких методів цей підхід може спрацювати краще?), інший підхід полягає у тому щоб розбити вибірку на підвибірки по кожному значенню фактора і аналізувати підвибірки окремо. Деякі алгоритми класифікації можуть працювати з факторними даними, можете використати такі алгоритми. Для виконання цього пункту можна обрати один алгоритм, який на Вашу думку, підійде найкраще.

8) Напишіть висновок у якому Ви порівняєте різні алгоритми класифікації та опишите отримані результати

Також пропонується додаткове завдання. Завдання є опціональним і виконується за бажанням студента.

9) Скачайте датасет
https://www.kaggle.com/team-ai/spam-text-message-classification
що містить набір повідомлень (спамових або ні) та побудуйте спам-фільтр на основі наївного байєсівського класифікатора.
https://en.wikipedia.org/wiki/Naive_Bayes_spam_filtering.

Вимоги до виконання, оформлення та здачі

Роботу можна виконувати в R або Python. Вибірку слід розбити на тестову та тренувальну у пропорціх 20% та 80%.

Робота має бути оформлена у вигляді .pdf (можливо .docx, як показала практика .doc-файли не завжди відкриваються) файлу який містить всю необхідну інформацію. Роботи потрібно здавати на парах або надсилати на email. У файлі має бути вказано: призвіще ім'я, номер варіанту, назва групи. Якщо pdf/docx файл не містить коду, то файл з кодом слід прикріпити окремо. Якщо робота надсилається на email, то в темі листа має бути вказано: "[ML-LAB-3], <ПРИЗВІЩЕ ІМ'Я>, <Група>" (символи [] потрібні, символи <> - ні).

Робота повинна містити коментарі, кожен фрагмент коду повинен мати пояснення - навіщо це робиться і які результати. Кожен метод повинен бути проаналізованим на предмет якості. Для вимірювання якості можна використовувати точність, похибку, precision, F1-score, confusion table.

Наприкінці роботи має бути висновок який повинен містити короткий огляд методів та отриманих результатів. У висновку має бути вказано який метод(и) виявились найкращими.

Кожна робота буде розглядатися на відповідність критеріям описаним вище, та на обгрунтованість прийнятих рішень. Кожен студент, повинен виконати свою роботу самостійно. Ідентичні, або майже ідентичні роботи прийматися до уваги не будуть. Прошу не брати чужі роботи і робити в них косметичні зміни, типу заміни на свій датасет.

Варіанти

Варіант 1
URL: https://archive.ics.uci.edu/ml/datasets/Abalone

Відгук: `Rings`, розбийте на три групи < 8, $[8, 12]$, > 12
Факторна змінна: `Sex`

Варіант 2
URL: https://archive.ics.uci.edu/ml/datasets/Adult
Відгук: `Остання колонка`
Факторна змінна: `Education`

Варіант 3
URL: https://archive.ics.uci.edu/ml/datasets/Arrhythmia
Відгук: `Остання колонка`
Регресори: `Колонки 1, 4, 5-15`
Факторна змінна: `Sex`

Варіант 4
URL: https://archive.ics.uci.edu/ml/datasets/Credit+Approval
Дані: `crx.data, crx.names`
Відгук: `Остання колонка`
Факторна змінна: `A13`

Варіант 5
URL: https://archive.ics.uci.edu/ml/datasets/Contraceptive+Method+Choice
Дані: `cmc.data, cmc.names`
Відгук: `Остання колонка`
Факторна змінна: `Колонка 8`

Варіант 6
URL: https://archive.ics.uci.edu/ml/datasets/Cylinder+Bands
Відгук: `Остання колонка`
Факторна змінна: `Колонка 2`

Варіант 7
URL: https://archive.ics.uci.edu/ml/datasets/Echocardiogram
Відгук: `Остання колонка`
Факторна змінна: `Колонка 4`
Коментар: `Зверніть увагу на опис даних, деякі рядки мають бути видалені!`

Варіант 8
URL: https://archive.ics.uci.edu/ml/datasets/Flags

Відгук: `religion, колонка 7`
Факторна змінна: `language`

Варіант 9
URL: https://archive.ics.uci.edu/ml/datasets/Glass+Identification
Відгук: `Остання колонка`
Факторна змінна: `Fe, колонка 10, перетворіть її на фактор з двома значеннями - менше середнього та більше середнього`

Варіант 10
URL: https://archive.ics.uci.edu/ml/datasets/Haberman%27s+Survival
Відгук: `Остання колонка`
Факторна змінна: `Перетворіть на фактор колонку Age - до 30 років, 30-60, більше 60`

Варіант 11
URL: https://archive.ics.uci.edu/ml/datasets/Heart+Disease
Дані: `processed.clivelend.data, heart-disease.names`
Відгук: `Остання колонка`
Факторна змінна: `Колонка 2`

Варіант 12
URL: https://archive.ics.uci.edu/ml/datasets/Hepatitis
Відгук: `Перша колонка`
Факторна змінна: `Комбінація 9 та 10, 4 можливих значення`

3. Матрична факторизація

Теоретичні відомості

Ймвірнісна матрична факторизація. Нехай ми маємо матрицю M, що складється з N_1 рядків (кожен рядок представляє користувача), та N_2 колонок (кожна колонка представляє об'єкт, фільми, пісню, книгу). m_{ij} - це рейтинг, який i-тий користувач поставив j-тому об'єкту. Така матриця буде мати багато пропущених значень, бо користувачі як правило не оцінюють всі доступні фільми а лише деякі.

Ми хочемо розкласти матрицю M у добуток матриць UV, $U \in \mathbb{R}^{N_1 \times d}$ та $V \in \mathbb{R}d \times N_2$, де $d << \min\{N_1, N_2\}$. Таким чи-

ном ми отримаємо деяких d параметрів які справді важливі для кожного користувача, в той час як всього у нас N_2 об'єктів, і це число може бути дуже великим, але нам не потрібно його знати всі N_2 рейтингів, а лише вивчити d параметрів. Зауважимо, що ці параметри - це латентні параметри і у них може не бути якоїсь зрозумілої інтерпритації.

Таким чином i-тий користувач буде представлений d-вимірним вектором, i-тим рядком у матрці U, а j-тий об'єкт, теж d-вимірним вектором, j-тою колонкою у матриці V.

Алгоритм.

Вхід: Неповна матриця M, з множиною Ω. Ранг d (вибирається як параметр для алгоритму).

Вихід: N_1 записів для користувачів $u_i \in \mathbb{R}^d$, N_2 для об'єктів, $v_j \in \mathbb{R}^d$.

Ініціалізація: згенерувати $v_j \in \mathcal{N}(0, \lambda^{-1} I)$.

Для кожної ітерації:

for $i = 1, \ldots, N_1$ обчислити вектор u_i:

$$u_i = \left(\lambda \sigma^2 I + \sum_{j \in \Omega_{u_i}} v_j v_j^T \right)^{-1} \left(\sum_{j \in \Omega_{u_i}} M_{ij} v_j \right).$$

for $j = 1, \ldots, N_2$ обсчислити вектор v_j:

$$v_j = \left(\lambda \sigma^2 I + \sum_{i \in \Omega_{v_j}} u_i u_i^T \right)^{-1} \left(\sum_{i \in \Omega_{v_j}} M_{ij} u_i \right).$$

Факторизація невід'ємних матриць. Припустимо, ми маємо матрицю $X \in \mathbb{R}^{N_1 \times N_2}$, такую що всі X_{ij} відомі (немає пропущених значень), а також всі $X_{ij} \geq 0$. Наша задача розкласти матрицю X у добуток матриць $W \in \mathbb{R}^{N_1 \times k}$ та $H \in \mathbb{R}^{k \times N_2}$, причому $W_{ik} \geq 0$ та $H_{kj} \geq 0$.

При цьому можливо, що матриця X має багато нулів. Факторизація яку ми хочемо отримати буде приблизною: $X_{ij} \approx \sum\limits_k W_{ik} H_{kj}$.

Цей підхід має багато різних застосувань, наприклад:

1. Аналіз текстових даних - X_{ij} - це кількість входжень i-того

слова в j-тий документ.
2. Аналіз зображень (зокрема ідентифікація облич) - $N \times M$ векторизоване зображення може бути представлене колонкою X.

Існує дві цільові функції (що веде до двох різних алгоритмів): Перший варіант це мінімізація норми Фробеніуса:

$$||X - WH||_F^2 = \sum_i \sum_j (X_{ij} - (WH)_{ij})^2.$$

Другий це дивергенція:

$$D(X||WH) = -\sum_i \sum_j [X_{ij} \ln(WH)_{ij} - (WH)_{ij}].$$

Важливо, що обидва алгоритма досить швидкі з обчислювальної точки зору.

Задача полягає у мінімізації цільової функції за обмежень невід'ємності для матриць H та W.

Теорема 1. *Послідовність $||X - WH||_F$ не зростає при заміні:*

$$H_{ij} = H_{ij} \frac{(W^T X)_{ij}}{(W^T WH)_{ij}},$$

$$W_{ij} = W_{ij} \frac{(XH^T)_{ij}}{(WHH^T)_{ij}}.$$

Норма буде інваріантною для такої послідовності тоді і лише тоді коли W та H стаціонарні точки цієї відстані.

Теорема 2. *Функція $D(X||WH)$ не зростає при заміні:*

$$H_{kl} = H_{kl} \frac{\sum_i W_{ik} X_{il} / (WH)_{il}}{\sum_j W_{jk}},$$

$$W_{ij} = W_{ij} \frac{\sum_k H_{jk} X_{ik} / (WH)_{ik}}{\sum_l H_{jl}}.$$

Дівергенція буде інваріантною для цієї послідовності тоді і лише тоді коли W та H є стаціонарними точкими дівергенції.

Завдання для лабораторних робіт

Всього передбачено 12 варіантів. В кожному варіанті буде вказано посилання по якому можна скачати датасет для аналізу а також додаткову інформацію.

Завдання полягає у тому, щоб побудувати рекомендаційну систему.

1. Зчитайте дані, та перетворіть, за потреби у форму матриці (можливо з пропущеними даними).
2. Імплементуйте алгоритм ймовірнісної матричної факторизації в якому d, λ, σ^2-будуть параметрами.
3. Підберіть найкращі d, λ, σ^2. Найголовніше тут d. λ та σ^2 можете покласти рівними одиниці і підбирати якщо залишається час.

Якщо у Вашому датасеті більше 100 000 значень, або матриця виходить суттєво більшою ніж 1.5 млн елементів (включаючи пропущені значення), то можете урізати свій датасет. Для валідації викиньте кілька спостережень і покажіть як алгоритм їх спрогнозує.

Вимоги до виконання, оформлення та здачі

Роботу можна виконувати в R або Python. Вибірку слід розбити на тестову та тренувальну у пропорціх 20% та 80%.

Робота має бути оформлена у вигляді .pdf (можливо .docx, як показала практика .doc-файли не завжди відкриваються) файлу який містить всю необхідну інформацію. Роботи потрібно здавати на парах або надсилати на email. У файлі має бути вказано: призвіще ім'я, номер варіанту, назва групи. Якщо pdf/docx файл не містить коду, то файл з кодом слід прикріпити окремо. Якщо робота надсилається на email, то в темі листа має бути вказано: "[ML-LAB-3], <ПРИЗВІЩЕ ІМ'Я>, <Група>" (символи [] потрібні, символи <> - ні).

Робота повинна містити коментарі, кожен фрагмент коду повинен мати пояснення - навіщо це робиться і які результати. Опишіть швидкість роботи алгоритму, кількість ітерацій, досягнуті значення цільової функції та підібрані гіперпараметри. Алгорим повинен працювати притомний час, не більше години для одного набору гіперпараметрів та 10-20 ітерацій (взагалі кажучи, це повинно займати кілька хвилин).

Наприкінці роботи має бути висновок який повинен містити короткий огляд методів та отриманих результатів.

Кожна робота буде розглядатися на відповідність критеріям описаним вище, та на обгрунтованість прийнятих рішень. Кожен студент, повинен виконати свою роботу самостійно. Ідентичні, або майже ідентичні роботи прийматися до уваги не будуть. Прошу не брати чужі роботи і робити в них косметичні зміни, типу заміни на свій датасет. Також не треба присилати мені подібні задачі з інтернету, по-перше я їх вже бачив, по-друге там часто не зовсім ця задача. В разі виявлення плагіату робота буде анульована. Якщо буде виявлено, що робота містить матеріал з роботи іншого студента, анульовано буде обидві роботи.

Варіанти

Варіант 1
Опис даних: http://www2.informatik.uni-freiburg.de/ cziegler/BX/
Самі дані: http://www2.informatik.uni-freiburg.de/ cziegler/BX/BX-CSV-Dump.zip

Варіант 2
Опис даних: https://www.kaggle.com/tamber/steam-video-games/
Самі дані: https://www.kaggle.com/tamber/steam-video-games/download

Варіант 3
Опис даних: http://files.grouplens.org/datasets/hetrec2011/hetrec2011-movielens-readme.txt
Самі дані: http://files.grouplens.org/datasets/hetrec2011/hetrec2011-movielens-2k-v2.zip

Варіант 4
Опис даних: http://files.grouplens.org/datasets/hetrec2011/hetrec2011-lastfm-readme.txt
Самі дані: http://files.grouplens.org/datasets/hetrec2011/hetrec2011-lastfm-2k.zip
Додаткова інформація: Використовуйте кількість прослуховувань в якості метрики (рейтингу)

Варіант 5
Опис даних: https://guoguibing.github.io/librec/datasets.html, CiaoDVD

Самі дані: https://guoguibing.github.io/librec/datasets/CiaoDVD.zip

Варіант 6
Опис даних: https://www.kaggle.com/tamber/steam-video-games/
Самі дані: https://www.kaggle.com/tamber/steam-video-games/download

Варіант 7
Опис даних: http://www2.informatik.uni-freiburg.de/ cziegler/BX/
Самі дані: http://www2.informatik.uni-freiburg.de/ cziegler/BX/BX-CSV-Dump.zip

Варіант 8
Опис даних: https://www.kaggle.com/tamber/steam-video-games/
Самі дані: https://www.kaggle.com/tamber/steam-video-games/download

Варіант 9
Опис даних: http://files.grouplens.org/datasets/hetrec2011/hetrec2011-movielens-readme.txt
Самі дані: http://files.grouplens.org/datasets/hetrec2011/hetrec2011-movielens-2k-v2.zip

Варіант 10
Опис даних: http://files.grouplens.org/datasets/hetrec2011/hetrec2011-lastfm-readme.txt
Самі дані: http://files.grouplens.org/datasets/hetrec2011/hetrec2011-lastfm-2k.zip
Додаткова інформація: Використовуйте кількість прослуховувань в якості метрики (рейтингу)

Варіант 11
Опис даних: https://guoguibing.github.io/librec/datasets.html, CiaoDVD
Самі дані: https://guoguibing.github.io/librec/datasets/CiaoDVD.zip

Варіант 12
Опис даних: https://www.kaggle.com/tamber/steam-video-games/
Самі дані: https://www.kaggle.com/tamber/steam-video-games/download

Теоретичні відомості

Щільні нейронні мережі. Почнемо з розгляду нейронних мереж з одним прихованим шаром:

Зауважимо, що як правило, ми додаємо один вхідний нейрон що рівний 1.

Для задачі класифікації на K класів, отримаємо K вихідних нейронів кожен з яких моделює ймовірність приналежності до відповідного класу.

Прихований шар складатиметься з M нових змінних Z_m:

$$Z_m = \sigma(\alpha_{0m} + \alpha_m^T X),\ m = 1, \dots, M,$$

$$T_k = \beta_{0k} + \beta_k^T Z,\ k = 1, \dots, K,$$
$$f_k(X) = g_k(T),\ k = 1, \dots, K,$$

де $Z = (Z_1, \dots, Z_M)$, та $T = (T_1, \dots, T_k)$.

Активаційна функція σ як правило вибирається як сігмоїда $\sigma(v) = \frac{1}{1+e^{-v}}$ або RECtified Linear Unit (reclu) функція $reclu(z) = z^+ = \max 0, z$.

Вихідна функція $g_k(T)$ це фінальна трансформація вектору T. Для регресії як правило використовується функція $g_k(T) = T_k$ а для класифікації софтмакс функція:

$$g_k(T) = \frac{e^{T_k}}{\sum\limits_{l=1}^{K} e^{T_l}}.$$

Нейрони всередині мережі називаються прихованими нейронами, оскільки величини Z_m не спостерігаються безпосередньо. Взагалі кажучи, нейронна мережа може мати більш ніж один прихований рівень, але як ми побачимо пізніше, з теоретичної точки зору одного рівня достатньо.

Зауважимо, що якщо σ це лінійна функція то вся нейронна мережа перетвороюється на лінійну модель. Таким чином можна уявити собі нейронну мережу як нелінійне узагальнення лінійної регресії.

Розглянемо процедуру підгонки такої нейронної мережі.

Невідомі параметри в нейронних мережах часто називають вагами. Позначимо множину всіх вагів через θ, що включає в

сееб:

$$\{\alpha_{0m}, \alpha_m;\ m = 1, \dots, M\},\ M(p+1)\text{вагів},$$
$$\{\beta_{0k}, \beta_k;\ k = 1, \dots, K\},\ K(M+1)\text{вагів}.$$

Для регресії використаємо суму квадратів похибок:

$$R(\theta) = \sum_{k=1}^{K} \sum_{i=1}^{N} (y_{ik} - f_k(x_i))^2.$$

Для класифікації розглянемо або середньоквадратичну похибку, або крос-ентропію:

$$R(\theta) = -\sum_{i=1}^{N} \sum_{k=1}^{K} y_{ik} \log f_k(x_i),$$

із відповідним класифікатором $G(x) = \text{argmax}_k f_k(x)$.

Із softmax активаційною функцією та крос-ентропією як цільовою функцією нейронна мережа виходить в точності еквівалентною логістичній регресії.

Для нейронних мереж із кількома прихованими шарами конструкція аналогічна.

Згорткові нейронні мережі. Розглянемо архітектуру згорткової нейронної мережі. По-перше, на вхід мережі подаються вже не вектори а багатовимірні масиви, двовимірні для одноканальних зображень або тривимірні для RGB-зображень. Тоді "матриця" X стає у даному представленні 4-вимірною! А саме $X \in \mathbb{R}^{m \times m \times 3 \times N}$, де $m \times m$ це розмір зображення, 3 - це кількість каналів, N - кількість зображень.

Далі, на зображення накладається ряд фільтрів розмірності $k \times k \times 3$ (тут підсумовування буде вестися одночасно по всім каналам, тобто у сумі буде $3k^2$ доданків). Кожен фільтр перетворює $m \times m \times 3$ зображення на $(m-k+1) \times (m-k+1) \times 1$ масив. Потім декілька таких фільтрів "з'єднуються" утворюючи $(m-k+1) \times (m-k+1) \times K$ масив, де K - це кількість фільтрів.

(Візуалізацію цього процесу для одноканальних зображень можна подивитися тут:
https://cs231n.github.io/convolutional-networks/).

Така конструкція утворює один шар згорткової нейронної мережі (зауважимо, що до згортки ще додають вільний член, як у щільних нейронних мережах, один вільний член на фільтр).

Ми вже бачили, що використання фільтру $k \times k \times 3$ перетворює зображення $m \times m \times 3$ у масив $(m-k+1) \times (m-k+1) \times 1$. Іноді така зміна розмірності небажана (при чому "бажаною" може бути як більша так і менша розмірність!). Окрім того, для великих зображень, рух по одному пікселю може бути занадто повільним. Тому для операції згортки визначають ще два параметри: padding (p) та stride (s). Padding полягає у тому, що края зображення доповнюються нулями, щоби після згортки отримати зображення більшого розміру (ніж $m-k+1$), а stride означає "крок" або зсув згортки. У прикладах вище ми мали stride=1, що означає, що ми зсуваємо фільтр на один піксель. Тоді розмір об'єкту після застосування такої згортки буде $\left[\frac{m+2p-k}{s}+1\right] \times \left[\frac{m+2p-k}{s}+1\right] \times 1$, де [] - це ціла частина.

Більше про "арифметику" згорткових нейронних мереж можна почитати тут: https://arxiv.org/pdf/1603.07285.pdf.

Зауважимо, що згортка від згортки - знову буде деякою згорткою (як і з множенням матриць), тому для того щоб нейронна мережа працювала необхідно додати в систему нелінійність. Для цього після згортки застосовують активаційну функцію (покоординатно).

І останнє, про що залишилося сказати, це про операцію pooling. Виявляється, що у зображення далеко не всі пікселі несуть змістовне навантаження і частину пікселів можна просто викинути, суттєво зменшивши розмірність. У згорткових нейронних мережах так і роблять. Після застосування згортки, отриманий (двовимірний) масив розбивається на блоки (часто 2×2) з яких вибирається максимальне значення. В результаті отримується масив значно менших розмірів.

Активаційну функцію при цьому, як правило, застосовують перед pooling-ом.

Таким чиному утворюється "стандартний блок" згорткової нейронної мережі: Згортка -> Активація -> Pooling. Сюди іноді додаються BatchNormalization та/або Dropout про який ми говорили раніше.

Завдання для лабораторних робіт

Загальні відомості

Всього передбачено 3 варіанти.

Для виконання роботи слід використати один із існуючих фреймворків для роботи з нейронними мережами (рекомендується keras або pytorch). В обох рекомендованих фреймворках є вбудовані датасети які потрібні для даної роботи. Роботу можна виконувати будь-якою мовою програмування (рекомендується R або Python).

Завдання полягає у наступному.

1) Завантажте датасет, підготуйте його для входу в нейронну мережу.

2) Побудуйте щільну нейронну мережу для класифікації. Проведіть аналіз отриманих результатів. Чи має місце перенавчання?

3) Побудуйте згорткову нейронну мережу. Проаналізуйте чи змінився результат. З'ясуйте чи мало місце перенавчання.

4) Спробуйте покращити результати використавши різні налаштування (різні оптимізатори, learning rate, batch size) а також додавши шари BatchNormalization та Dropout.

5) Проаналізуйте процес навчання. Як змінювалася похибка та точність?

6) Напишіть висновок у якому Ви опишете отримані результати.

Також пропонується додаткове завдання. Завдання є опціональним і виконується за бажанням студента.

7) Скачайте датасет
https://www.kaggle.com/c/dogs-vs-cats/data
що містить набір зображень котів та собак (розмір датасету більше 800MB). Побудуйте згорткову нейронну мережу для класифікації. Використайте генератор для тренування даних (функція fit_generator, https://tensorflow.rstudio.com/reference/keras/fit_generator/). Спробуйте досягнути точності класифікації на тестовій вибірці у 83% або вище. За потреби використайте аугментацію.

Дана задача потребуватиме сучасної відеокарти, або Google Colab.

Додаткова інформація

Dropout в r-keras: https://tensorflow.rstudio.com/reference/keras/layer_dropout/
BatchNormalization в r-keras: https://tensorflow.rstudio.com/reference/keras/layer_batch_normalization/
Дане завдання не вимагає особливих обчислювальних потужностей. Однак, якщо виникає потреба (або бажання) можна використовувати Google Colab (як для python так і для r).

Вимоги до виконання, оформлення та здачі

Робота має бути оформлена у вигляді .pdf (можливо .doc/.docx) файлу який містить висновки та коментарі до роботи. Код може бути включено у файл зі звітом. Рекомендується викласти звіт та код на github та надіслати посилання.

Кожна робота буде розглядатися на відповідність критеріям описаним вище, та на обгрунтованість прийнятих рішень. Кожен студент, повинен виконати свою роботу самостійно. Ідентичні, або майже ідентичні роботи прийматися до уваги не будуть.

Варіанти

Варіант 1 fashion_mnist

Варіант 2 cifar10

Варіант 3 mnist