Київський національний університет імені Тараса Шевченка

МЕТОДИЧНІ ВКАЗІВКИ
до курсу “Статистичні алгоритми 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 N4 від 21.10.2021 року

МЕТОДИЧНІ ВКАЗІВКИ курсу "Статистичні алгоритми навчання " / Упорядник: В.В. Голомозий- К., Видавничо-поліграфічний центр 'Київський університет', 2021 - 20 с.

Навчальне видання

МЕТОДИЧНІ ВКАЗІВКИ до курсу "Статистичні алгоритми навчання"

для студентів механіко-математичного факультету

Упорядник

Голомозий Віталій Вікторович

Зміст

Виконання завдань

Методичні вказівки до курсу "Статистичні алгоритми навчання " призначені для студентів II курсу магістратури механіко-математичного факультету спеціальності "Прикладна та теоретична 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**. Розділ 4 є теоретичним і не передбачає виконання 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. Щільні нейронні мережі

Призначення: Зрозуміти базову архітектуру щільних нейронних мереж. Засвоїти метод градієнтного спуску.

2. Згорткові нейронні мережі

Призначення: Зрозуміти в чому полягає операція згортки та яким чином згорткові нейронні мережі аналізують зображення.

3. Рекурентні нейронні мережі

Призначення: Вивчити основні блоки рекурентної нейронної мережі та навчитися використовувати їх у задачах обробки природньої мови.

4. Теоретичні основи нейронних мереж

Призначення: Засвоїти теоретичні засади нейронних мереж.

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. *I. Goodfellow*, Deep Learning.

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

1. Щільні нейронні мережі

Теоретичні відомості

Нейронна мережа - це багаторівневий алгоритм навчання, основою якого є нейрони та шари. Кожен шар здійснює певне нелійне перетворення попереднього шару, і фінальний шар є прогнозом, або результатом роботи нейронної мережі. Нелінійне перетворення кожного шару здійснюється у два етапи, по-перше здійснюється лінійне перетворення попереднього шару, а потім до нього застосовується деяка активаційна функція (що є нелінійною).

Визначимо нейронну мережу формально. Нехай як і раніше маємо матрицю $X \in \mathbb{R}^{N \times p}$ де N - це кількість спостережень, p - кількість регресорів. Нехай також маємо вектор відгуків $Y \in \mathbb{R}^N$. Як і раніше позначатимом k-тий регресор верхнім індексом (X^k - це вектор-стовпчик довжини N, що представляє всі значення k - того регресора) а i-те спостереження позначатимемо нижнім індексом (тобто X_i - це вектор-рядок з p регресорів, що відповідають i-тому спостереженню).

Архітектурою нейронної мережі назвемо вектор з L чисел, які відповідають кількості нейронів на кожному шарі (тобто маємо L шарів, та m_l - нейронів у кожному з них, $l = 1, L$). Окрім того, на кожному шарі будемо використовувати активаційну функцію $a_l \colon \mathbb{R} \to \mathbb{R}$.

Кожен нейрон буде мати внутрішнє та зовнішнє значення. Будемо позначати шари верхнім індексом в квадратних дужках. Тобто $Z_i^{[l](k)}$ - означатиме, внутрішнє значення k-того нейрона, l-того шару на i-тому спостереженні, відповідно
$A_i^{[l](k)} = a_l(Z_i^{[l](k)})$ - зовнішнє значення значення k-того нейрона, l-того шару на i-тому спостереженні. Ваги будемо позначати $W^{[l]}$ - це матриця вагів для l-того шару, $b^{[l]}$ - вектор зміщень (вільний член) для l-того шару. Тоді, зручно буде припустити, що $A^{[0]} = X$. Тоді матимуть місце такі формули:

$$\begin{gathered} Z_i^{[l]} = A_i^{[l-1]} W^{[l]} + b^{[l]}, \\ A_i^{[l](k)} = a_l(Z_i^{[l](k)}) = a_l(A_i^{[l-1](k)} W_k^{[l]} + b_k^{[l]}), \\ Z^{[l]} \in \mathbb{R}^{N \times m_l}, \\ A^{[l-1]} \in \mathbb{R}^{N \times m_{l-1}}, \\ W^{[l]} \in \mathbb{R}^{m_{l-1} \times m_l} \\ b^{[l]} \in \mathbb{R}^{1, m_l}. \end{gathered} \tag{1.1}$$

В подальшому будемо писати $Z^{[l]} = A^{[l-1]} W^{[l]} + B^{[l]}$, розуміючи $B^{[l]}$ - як матрицю з N однакових рядків $b^{[l]}$. Зауважимо, що $m_0 = p$. Як за цією формулою отримати прогноз нейронної мережі:

1. Значення першого шару: $A^{[1]} = a_1(XW^{[1]} + B^{[1]})$ (тут функція a_1 застосовується покоординатно).

2. Значення другого шару: $A^{[2]} = a_2(A^{[1]} W^{[2]} + B^{[2]}) = a_2(a_1(XW^{[1]} + B^{[1]}) W^{[2]} + B^{[2]})$.

3. Аналогічно обчислення продовжуються далі, і на L-тому кроці отримуємо наш відгук

$$\hat{Y} = A^{[L]} = a_L(A^{[L-1]}W^{[L]}+B^{[L]}) = a_L(a_{L-1}(\dots a_2(a_1(XW^{[1]}+B^{[1]})W^{[2]}+B^{[2]}))W^{[L]}+B^{[L]}).$$

Цей процес обчислення $\hat{Y}$ - називається forward propagation.

Що використовують у якості активаційних функцій? Насправді використовується лише декілька функцій.

1. Сигмоїда: $\sigma(u) = \frac{1}{1+e^{-u}}$.
2. ReLU: $a(u) = u^+ = \max\{0, u\}$.
3. Leaky ReLU: $a(u) = u$, якщо $u \geq 0$ та λu інакше, де λ деяке мале число, як правило 0.01.
4. Гіперболічний тангенс: $tanh(u) = \frac{e^u - e^{-u}}{e^u + e^{-u}}$.
5. Softmax. Функція softmax застосовується до векторів. Нехай $\theta \in \mathbb{R}^r$, тоді

$$\text{softmax}(\theta) = \left(\frac{e^{-\theta_1}}{\sum\limits_{j=1}^{r} e^{-\theta_j}}, \frac{e^{-\theta_2}}{\sum\limits_{j=1}^{r} e^{-\theta_j}}, \dots, \frac{e^{-\theta_r}}{\sum\limits_{j=1}^{r} e^{-\theta_j}} \right).$$

Нехай маємо матрицю $Z^{[L]}$. Тоді $a_L(Z_i^{[L](k)}) = \frac{e^{-Z_i^{[L](k)}}}{\sum\limits_j e^{-Z_i^{[L](j)}}}$. Функцію Softmax використовують на останньому шарі нейронної мережі, що здійснює багатокласову класифікацію. Нехай ми маємо K - класів, тоді покладемо $m_L = K$. І тоді

$$\hat{Y}_i \in \mathbb{R}^{1\times K} = \mathbb{R}^{1\times m_L},$$

а відповідно

$$\hat{Y} = A^{[L]} \in \mathbb{R}^{N\times K},$$

де вектор $\hat{Y}$ - це вектор ймовірностей того, що i-те спостереження потрапляє у певний клас. Виникає невеличка проблема, що полягає у тому, що результат роботи нейронної мережі це матриця $N \times K$ а вектор відгуків Y це вектор $N \times 1$. Тому в таких задачах, вектор Y представляють в вигляді матриці $N \times K$, де в кожному рядочку є лише одна одиниця, що відповідає номеру справжнього класу, а решта нулі.

Завдання для лабораторних робіт

Загальні відомості

Всього передбачено 6 варіантів. Ваш варіант відповідає Вашому номеру у списку з журналу (якщо Ваш номер більше 6 то візьміть залишок від ділення на 6).

Вам буде надано файли з даними, які слід аналізувати, кожному варіанту відповідає свй файл. Файл містить три колонки - перші дві, це матриця дизайну X, остання це відгук Y. Відгук завжди бінарний.

Для проведення **класифікації** потрбіно зробити наступне:

1) Намалювати дані на картинці.

2) Реалізувати алгоритм зворотного розповсюдження похибки за формулами з лекції.

3) Підібрати архітектуру нейронної мережі для найкращої класифікації, натренувати нейронну мережу написаним алгоритмом.

4) Намалювати області класифікації.

6) Напишіть висновок з описом того, що Ви робили, як підбирали оптимальні параметри і як тренувалася нейронна мережа.

Вимоги до виконання, оформлення та здачі

Роботу можна виконувати в R або Python.

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, або надіслано окремим файлом (файлами, але не в архіві).

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. При виявлені плагіату робота анульовується. Якщо виявиться, що робота містить фрагменти іншої роботи, то інша робота теж анульовується, навіть якщо вона була успішно здана раніше.

2. Згорткові нейронні мережі

Теоретичні відомості

Ми вже розглянули щільні нейронні мережі та методи оптимізації. Однак, виявляється, що на практиці щільні нейронні мережі не є дуже корисними самі по собі. Адже навіть порівняно невелика мережа, буте мати дуже багато коефіцієнтів які треба навчити. Інша, досить серйозна, проблема полягає у тому, що щільні нейронні мережі надзвичайно важко інтерпритувати в термінах предметної області задачі, що досліджується. У цьому

розділі ми розглянемо приклад такой предметної області - а саме обробки зображень.

Статистична обробка зображень (а саме класифікація, розпізнавання образів, пошук об'єктів на зображеннях та інші задачі) завжди була надзвичайно важливою областю досліджень з величезним практичним (зокрема комерційним) потенціалом. Однак протягом багатьох років, існуючі методи не давали прйнятних результатів.

Так, навіть найпростіша задача класифікації рукописних цифр довгий час не мала прийнятного розв'язку. Класичні методи, такі як knn або логістична регресія дозволяли здійснювати класифікацію зображень, але точність при цьому була відносно низька. Щодо більш складних задач, типу розпізнавання людини на фотографії, успіхи були ще більш скромними.

Однак сама галузь обробки зображень (мається на увазі графічної обробки фотографами та дизайнерами) активно розвивалася. Було вироблено цілий ряд технік та методів перетворення зображення шляхом накладання так званих фільтрів. Як працює подібний фільтр? Зокрема фільтр Собеля, визначає градієнт - або швидкість зміни інтенсивності кольору у горизонтальному та вертикальному напрямах. Для того, щоб обчислити градієнт виконують наступну операцію. Нехай $X \in \mathbb{R}^{m \times m}$ - це матриця, що містить інтенсивності кожного пікселя на зображенні (вважаємо, що зображення має розмір $m \times m$ і лише один канал, наприклад grayscale). Тоді для обчислення горизонтального градієнта використовують матрицю:

$$F_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \tag{2.1}$$

А для вертикального

$$F_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 2 \\ -1 & -2 & -1 \end{bmatrix} \tag{2.2}$$

Такі 3×3 матриці послідовно накладаються на зображення, починаючи з верхнього лівого кута. Накладання полягає у тому, що обчислюється поелементна сума всіх членів фільтру помножених на відповідний елемент зображення.

Нахай для прикладу маємо 5×5 зображення:

$$X = \begin{pmatrix} 150 & 150 & 0 & 92 & 96 \\ 145 & 145 & 2 & 94 & 100 \\ 130 & 100 & 10 & 12 & 200 \\ 120 & 113 & 19 & 99 & 96 \\ 183 & 115 & 22 & 103 & 88 \end{pmatrix} \tag{2.3}$$

Тоді накладемо фільтр (скажімо F_x) у лівий верхній кут і обчислимо значення: $150*1+150*0-1*0+145*2+145*0-2*2+1*130+0*100-1*10 = 556$. Отже значення фільтру у точці $(1,1)$ рівне 556. Потім зсуваємо фільтр

на одиницю вправо і отримаємо значення: $150*1+0*0-92*1+145*2+2*0-94*2+100*1+10*0-12*1=248$. Отже значення фільтру у точці $(2,1)$ рівне 248. Потім зсуваємо ще на одинцю вправо і обчислюємо значення в точці $(3,1)$. Далі вправо зсувати вже не можна, тому ми розміщуємо фільтр на один рядок нижче, починаючи з лівого боку і обчислюємо значення у точках $(1,2)$, $(2,2)$, $(3,2)$. Далі зсуваємо, ще на одинцю вниз і обчислюємо значення в точках $(1,3)$, $(2,3)$, $(3,3)$. На цьому обчислення закінчуються. В результаті накладання фільтру F_x на матрицю X отримали деяку 3×3 матрицю, назвемо її G_x.

Якщо "намалювати" цю матрицю, тобто масштабувати, так щоб числа були в інтервалі від 0 до 255 то побачимо, що на ній виділені вертикальні лінії. Аналогічно зробимо з фільтром F_y і отримаємо 3×3 матрицю G_y. Далі поелементно обчислимо $\sqrt{G_x^2+G_y^2}$ - це і будуть градієнти в кожній точці отримані за допомогою фільтру Собеля. Напрямок градієнту задається $\theta = \text{atan}\left(\frac{G_y}{G_x}\right)$.

В результаті отримаємо таке зображення як показано на малюнку. Можна помітити, що фільтр "з'їв" по одному пікселю з кожного боку, тобто "відфільтроване" зображення стало $m-2\times m-2$.

Зауважимо, що описане накладання фільтру називається "згорткою", хоча це не є згорткою у математичному сенсі!

Повертаючись до задач машинного навчання, основною проблемою при аналізі зображень була їх багатовимірність та відсутність сенсу у застосуванні алгоритмів попіксельно. Тому з'явилася ідея, використати знання предметної області (накладання фільтрів) для навчання. Зрозуміло, що розмірність вона зменшує не суттєво, але дані стають більш розрідженими і їх стає легше аналізувати.

Застосування різних фільтрів разом з такими методами як SVM дозволило досягнути суттєвого прогресу і перейти від розпізнавання рукописних чисел, наприклад до задачі класифікації зображення на предмет наявності чи відсутності на ньому чогось (людини, обличчя і т.д.). Зокрема, є досить відомий алгоритм із застосуванням іншого фільру (HoG) та ядерного SVM у задачі класифікації фотографії на наявність на ній людини. Цей алгоритм планувалося використовувати для автоматизації роботи світлофорів.

Однак проблеми все ще залишалися. Фотографії потрібно було обробляти вручну накладаючи різні фільтри. Зображення великого розміру все ще дуже погано піддавалися аналізу. Багатокласова класифікація була фактично неможливою. Задачі типу пошуку об'єкту на зображенні та визначення його координат залишалися досить складними.

Варто зазначити, що спроби використати нейронні мережі для аналізу зображень майже не припинялися. Особливим ентузіастом у цій галузі був Ян ЛеКун. Тому досить природно з'явилася ідея використовувати згортки у нейронних мережах. Спочатку використовувалися фіксовані (детерміновані згортки) типу оператора Собеля або HoG. Але згодом, стало очевидно, що нейронна мережа може сама "вивчити" потрібні фільтри!

Отже розглянемо архітектуру подібної нейронної мережі (їх стали нази-

вати згортковими). 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" утвроюючи $(m-k+1) \times (m-k+1) \times K$ 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 12 варіантів. Ваш варіант відповідає залишку від ділення Вашого номера у журналі на 12 (якщо залишок рівний 0 то у Вас варіант 12).

Для виконання роботи слід використати один із існуючих фреймворків для роботи з нейронними мережами (рекомендується keras або pytorch). 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.

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
Опис даних: http://www.cs.utoronto.ca/ kriz/cifar.html
Самі дані: http://www.cs.utoronto.ca/ kriz/cifar-10-python.tar.gz

Варіант 2
Опис даних: http://www.vision.caltech.edu/Image_Datasets/Caltech101/
Самі дані: http://www.vision.caltech.edu/Image_Datasets/Caltech101/101_-ObjectCategories.tar.gz
Додаткова інформація: Скачайте архів (130МБ) та розархівуйте його. Отримаєте папку 101_ObjectCategories (150МБ). В цій папці буде багато інших папок. Візьміть для класифікації папки: Faces, Leopards, Motorbikes, airplanes, anchor, ant, barrel, bass. Кожна папка містить файли певної категорії (тобто назва папки це правильний відгук). Розділіть зображення в кожній папці на тренувальну та тестову частину та побудуйте CNN яка класифікуватиме зображення із тестової вибірки до одного з Ваших 8 класів.

Варіант 3
Опис даних: http://www.vision.caltech.edu/Image_Datasets/Caltech101/
Самі дані: http://www.vision.caltech.edu/Image_Datasets/Caltech101/101_-ObjectCategories.tar.gz
Додаткова інформація: Скачайте архів (130МБ) та розархівуйте його. Отримаєте папку 101_ObjectCategories (150МБ). В цій папці буде багато інших папок. Візьміть для класифікації папки: beaver cellphone dollar_bill garfield kangaroo minaret rhino stegosaurus windsor_chair. Кожна папка містить файли певної категорії (тобто назва папки це правильний відгук). Розділіть зображення в кожній папці на тренувальну та тестову частину та побудуйте CNN яка класифікуватиме зображення із тестової вибірки до одного з Ваших 9 класів.

Варіант 4
Опис даних: http://www.vision.caltech.edu/Image_Datasets/Caltech101/
Самі дані: http://www.vision.caltech.edu/Image_Datasets/Caltech101/101_-ObjectCategories.tar.gz
Додаткова інформація: Скачайте архів (130МБ) та розархівуйте його. Отримаєте папку 101_ObjectCategories (150МБ). В цій папці буде багато інших папок. Візьміть для класифікації папки: binocular chair dolphin

gerenuk ketch nautilus rooster stop_sign wrench Кожна папка містить файли певної категорії (тобто назва папки це правильний відгук). Розділіть зображення в кожній папці на тренувальну та тестову частину та побудуйте CNN яка класифікуватиме зображення із тестової вибірки до одного з Ваших 9 класів.

Варіант 5
Опис даних: http://www.vision.caltech.edu/Image_Datasets/Caltech101/
Самі дані: http://www.vision.caltech.edu/Image_Datasets/Caltech101/101_-ObjectCategories.tar.gz
Додаткова інформація: Скачайте архів (130МБ) та розархівуйте його. Отримаєте папку 101_ObjectCategories (150МБ). В цій папці буде багато інших папок. Візьміть для класифікації папки: bonsai chandelier dragonfly gramophone lamp octopus saxophone strawberry yin_yang. Кожна папка містить файли певної категорії (тобто назва папки це правильний відгук). Розділіть зображення в кожній папці на тренувальну та тестову частину та побудуйте CNN яка класифікуватиме зображення із тестової вибірки до одного з Ваших 9 класів.

Варіант 6
Опис даних: https://github.com/zalandoresearch/fashion-mnist
Самі дані: Перейдіть до розділу Get the Data. Там будуть посилання на дані та класи. Окрім того вибірка вже розбита на тестову та тренувальну.

Варіант 7
Опис даних: http://www.cs.columbia.edu/CAVE/software/softlib/coil-20.php
Самі дані: http://www.cs.columbia.edu/CAVE/databases/SLAM_coil-20_-coil-100/coil-20/coil-20-proc.zip

Варіант 8
Опис даних: http://www1.cs.columbia.edu/CAVE/software/softlib/coil-100.php
Самі дані: http://www.cs.columbia.edu/CAVE/databases/SLAM_coil-20_-coil-100/coil-100/coil-100.zip
Додаткова інформація: Датасет містить зображення для 100 різних категорій. Візьмітьь перші двадцять.

Варіант 9
Опис даних: http://www1.cs.columbia.edu/CAVE/software/softlib/coil-100.php
Самі дані: http://www.cs.columbia.edu/CAVE/databases/SLAM_coil-20_-coil-100/coil-100/coil-100.zip
Додаткова інформація: Датасет містить зображення для 100 різних категорій. Візьмітьь другі двадцять (21-40).

Варіант 10
Опис даних: http://www1.cs.columbia.edu/CAVE/software/softlib/coil-100.php
Самі дані: http://www.cs.columbia.edu/CAVE/databases/SLAM_coil-20_-coil-100/coil-100/coil-100.zip

Додаткова інформація: Датасет містить зображення для 100 різних категорій. Візьмітьь треті двадцять (41-60).

Варіант 11
Опис даних: http://www.vision.caltech.edu/Image_Datasets/Caltech101/
Самі дані: http://www.vision.caltech.edu/Image_Datasets/Caltech101/101_-ObjectCategories.tar.gz
Додаткова інформація: Скачайте архів (130МБ) та розархівуйте його. Отримаєте папку 101_ObjectCategories (150МБ). В цій папці буде багато інших папок. Візьміть для класифікації папки: brontosaurus cougar_face elephant hawksbill llama pagoda scissors tick Кожна папка містить файли певної категорії (тобто назва папки це правильний відгук). Розділіть зображення в кожній папці на тренувальну та тестову частину та побудуйте CNN яка класифікуватиме зображення із тестової вибірки до одного з Ваших 8 класів.

Варіант 12
Опис даних: http://www.vision.caltech.edu/Image_Datasets/Caltech101/
Самі дані: http://www.vision.caltech.edu/Image_Datasets/Caltech101/101_-ObjectCategories.tar.gz
Додаткова інформація: Скачайте архів (130МБ) та розархівуйте його. Отримаєте папку 101_ObjectCategories (150МБ). В цій папці буде багато інших папок. Візьміть для класифікації папки: brain cougar_body electric_-guitar grand_piano laptop okapi schooner sunflower Кожна папка містить файли певної категорії (тобто назва папки це правильний відгук). Розділіть зображення в кожній папці на тренувальну та тестову частину та побудуйте CNN яка класифікуватиме зображення із тестової вибірки до одного з Ваших 8 класів.

3. Рекурентні нейронні мережі

Теоретичні відомості

У попередньому розділі ми побачили як можна побудувати нейронну мережу яка використовує просторову структуру вхідних даних. В цьому розділі, ми покажемо яким чином можна спроектувати нейронну мережу для аналізу послідовностей.

Аналіз послідовностей має суттєві відмінності від аналізу зображень. Тут під послідовностями, ми майже завжди матимемо на увазі текст. (Хоча рекурентні нейронні мережі також застосовуються до аналізу ДНК послідовностей, до аналізу голосової інформації та ін.).

Отже перша суттєва відмінність полягає у тому, що при аналізі послідовностей ми маємо чітку "часову" пряму, або напрям - зліва-направо.
По-друге, зв'язок між окремими елементами послідовності може зберігатися протягом довгого "часу" (наприклад, щоб зрозуміти речення потрібно дочитати його до кінця).

По-третє, довжина послідовності варіюється (на відміну від фіксованих розмірів зображення).
По-четверте, кожен елемент послідовності як правило має значення (на відміну від пікселів у зображеннях, які по окремості не несуть ніякої інформації). Як наслідок, не дуже зрозуміло, яким чином порівнювати послідовності, які послідовності є "близькими" та як оцінити "якість" послідовності.

Розглянемо типові задачі, які розв'язують рекурентні нейронні мережі.

1. Sentiment analysis. Це типова задача класифікації, коли на вхід подається повідомлення і потрібно визначити його емоційне забарвлення.
2. Named entity recognition. Це задача пошуку певних об'єктів у послідовності (наприклад імен чи географічних об'єктів). На виході з алгоритму ми матимемо інформацію про те де знаходяться шукані об'єкти. Або більш точно - чи є кожен елемент послідовності шуканим об'єктом.
3. Machine translation. Машинний переклад, це переклад з однієї мови на іншу.

Існує також багато інших задач, наприклад: text summarization, question answering, text generation і так далі. Але перелічені три задачі будуть для нас особливо важливими, оскільки вони задають різні типові архітектури для рекурентних мереж.

Проблеми які виникаються при аналізі текстових даних.

1. Як представити "дані"? Що є "спостереженням"? Слово, символ, речення?
2. Як обробляти спостереження різної довжини?
3. Як врахувати зв'язок між словами, які знаходяться далеко одне від одного?
4. Як забезпечети "зворотній" аналіз, коли зміст слова може змінитися в залежності від того, що буде далі?
4. Як представити текстові дані для машинної обробки?

Для розв'язання цих задач існує багато різних моделей. Ми коротко оглянемо основні, в той час як повний огляд виходить далекі за межі даного курсу.

Існує дві принципових моделі для роботи з текстовими даними - що базується на символах та на словах. Більш поширеною є модель, що базується на словах, тому на ній ми і зосередимося.

РНН працює наступним чином:
1. У якості вхідних даних розглядається певний текст/набір текстів (часто його називають корпусом).
2. Кожне "спостереження" x_i - це речення, а ознакою або регресором буде слово у реченні x_i^2 - це друге слово у i-тому реченні. Основна відмінність РНН полягає у тому, що кількість ознак p не є фіксованою.
3. Кожне слово переводиться у певний числовий вектор. Найпростіший ва-

ріант - це one-hot вектор. Варто зазначити, що в сучасних застосуваннях такий підхід уже майже не використовують.
4. Отже, прихований шар нейронної мереж h отримує на вхід числовий вектор (фіксованої довжини!), що представляє наступне слово, та "стан" мережі у попередній момент часу. На основі цих двох об'єктів, прихований шар обчислює свій новий стан, та, можливо, відгук y^j. Новий стан, разом із наступним словом передається до наступної копії прихованого шару.
5. Ключовою особливістю РНН є те, що прихований шар є одним і тим же для кожного слова! Тобто, ваги прихованого шару використовуються і навчаються одночасно.

Більш формально:

$$h_{n+1} = g(W_h h_n + W_x x_i^{n+1} + b),$$

де g - активаційна функція, h_n - попередній стан нейронної мережі (як правило вектор), x_i^{n+1} - вектор (фіксованої довжини!), що представляє $n+1$-ше слово у реченні i, W_x - матриця вагів, b- вільний член.

Яким же чином, таку мережу можна натренувати? Таким же чином, як і звичайну нейронну мережу - нам потрібні будуть тренувальні, розмічені дані з "правильними" відповідями, цільова функція, по якій рахується градієнт, який потім "розповсюджується назад у часі" по нейронній мережі.

Однак такий підхід має суттєві недоліки, зокрема Vanishing gradient. Щоб якимось чином, цю проблему вирішити, а також забезпечити довгострокову "пам'ять" нейронній мережі використовують більш складні будівельні блоки ніж ми розглянули до цих пір.

Першим таким блоком яким ми розглянемо буде Gated Recurrent Unit (GRU).

Отже GRU блок, має наступну структуру. Ключовим елементом, є так звана "комірка пам'яті" (memory cell), на малюнку позначена як h_t. На вхід до GRU блоку подається поточне значення пам'яті h_{t-1} та слово x^t.

Спочатку обчислюється значення так званих гейтів (gate):

$$\Gamma_u = \sigma(W_{uh} h_{t-1} + W_{ux} x^t + b_u)$$

$$\Gamma_r = \sigma(W_{rh} h_{t-1} + W_{rx} x^t + b_r)$$

Основна задача гейту - визначити, коли слід "записати" певну інформацію у пам'ять. Наприклад, мережа може записати 1, щоб позначити, що підмет жіночого роду, для того, щоб використати цю інформацію пізніше. Зауважимо, що Γ як правило 0 або 1.

Після цього обчислюється нове значення за формулою:

$$\tilde{h}_t = \tanh(W_h(\Gamma_r * h_{t-1}) + W_x x^t + b).$$

Далі ми обчислюємо нове значення пам'яті за формулою:

$$h_t = \Gamma_u * \tilde{h}_t + (1 - \Gamma_u) * h_{t-1},$$

де $*$ - операція поелементного множення.

Завдання для лабораторних робіт

Всього передбачено 1 варіант.

Для виконання роботи слід використати один із існуючих фреймворків для роботи з нейронними мережами (рекомендується keras або pytorch). Дана робота передбачає аналіз стандартного датасету imdb.

Завдання полягає у наступному.

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

2) Натренуйте власне вкладення (embedding), проаналізуйте як якість алгоритму змінюється зі зміною розмірності латентного простору?

3) Побудуйте рекурентну нейронну мережу. Спробуйте простий рекурентний шар, або кілька таких шарів. Проаналізуйте як змінюється результат при зміні параметрів.

4) Спробуйте покращити результати додавши LSTM (https://www.rdocumentation.org/packages/keras/versions/2.3.0.0/topics/layer_lstm) або GRU блок(и).

5) Спробуйте покращити результати використавши різні налаштування (різні оптимізатори, learning rate, batch size).

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

7) Напишіть висновок у якому Ви опишете отримані результати.

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

8) Завантажте існуючий embedding (word2vec або GloVe) та використайте його у Вашій роботі замість того, щоб тренувати свій власний. Чи призведе це до покращення результатів?

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.

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.

4. Теоретичні основи нейронних мереж

Теоретичні відомості

Позначимо, через $I_n = [0,1]^n$ - n-вимірний гіперкуб, а через $C(I_n)$ - простір усіх неперервних функцій на I_n. Через $||\cdot||$ - позначимо рівномірну (супремумну) норму.

Позначимо через $M(I_n)$ простір скінчених, знакозмінних регулярних Борелівських мір на I_n.

Означення. Будемо казати, що функція $\sigma\colon \mathbb{R} \to \mathbb{R}$ є дискримінаційною, якщо з того, що для деякої міри $\mu \in M(I_n)$

$$\int_{I_n} \sigma(y^T x + \theta) d\mu(x) = 0$$

для всіх $y \in \mathbb{R}^n$ та $\theta \in \mathbb{R}$ випливає $\mu = 0$.

Означення. Будемо казати, що функція $\sigma\colon \mathbb{R} \to \mathbb{R}$ є сігмоїдальною, якщо $\sigma(t) \to 1,\ t \to +\infty$, $\sigma(t) \to 0,\ t \to -\infty$.

Теорема 1. *нехай σ довільна дискрімінаційна функція. Тоді скінченні суми у вигляді:*

$$G(x) = \sum_{j=1}^{N} \alpha_j \sigma(y_j^T x + \theta_j)$$

є щільними в $C(I_n)$. Іншими словами, для довільної $f \in C(I_n)$ та $\varepsilon > 0$, існує сума $G(x)$ така, що:

$$|G(x) - f(x)| < \varepsilon,$$

для всіх $x \in I_n$.

Доведення.

Нехай $S \subset C(I_n)$ це множина функцій вигляду $G(X)$ як визначено у формулюванні теореми. Очевидно, що S це лінійний підпростір $C(I_n)$. Доведемо, що замикання S рівне $C(I_n)$.

Нехай це не так, і замикання S (позначимо його R) формує деякий підпростір в $C(I_n)$. Тоді за наслідком до теореми Хана-Банаха, існує обмежений (в сенсі норми, або неперервний) лінійний функціонал на $C(I_n)$ позначимо його через L, такий що $L \neq 0$ але $L(R) = L(S) = 0$.

За теоремою Рісса:

$$L(h) = \int_{I_n} h(x) d\mu(x)$$

для деякої міри $\mu \in M(I_n)$ та для всіх $h \in C(I_n)$. Зокрема, оскільки $\sigma(y^T x + \theta) \in R$ для всіх y та θ, ми отримуємо, що

$$\int_{I_n} \sigma(y^T x + \theta) d\mu(x) = 0$$

для всіх y, θ.

Однак ми припустили, що σ є дискримінаційною функцією, отже з цієї умови випливає, що $\mu = 0$ що протирічить припущенню. Отже S має бути щільною в $C(I_n)$.

□

Покажемо тепер, що кожна сігмоїдальна функція є дискримінаційною. Зауважимо, що в нейронних мережах, як правило функції зростаючі, але нам таке обмеження не потрібне.

Лема 1. *Будь-яка обмежена, вимірна, неперервна сігмоїдальна функція σ є дискримінаційною.*

Доведення.

Щоб показати це помітимо, що для довільних x, y, θ, ϕ:

$$\sigma(\lambda(y^T x + \theta) + \phi) \begin{cases} \to 1, \text{ для } y^T x + \theta > 0, \ \lambda \to +\infty, \\ \to 0, \text{ для } y^T x + \theta < 0, \ \lambda \to +\infty, \\ = \sigma(\phi), \text{ для } y^T x + \theta = 0, \ \forall \lambda \end{cases} \tag{4.1}$$

Отже функція $\sigma_\lambda(x) = \sigma(\lambda(y^T x + \theta) + \phi)$ збігається поточково та обмежено до функції (при $\lambda \to +\infty$):

$$\gamma(x) = \begin{cases} 1, \text{ при } y^T x + \theta > 0, \\ 0, \text{ при } y^T x + \theta < 0, \\ \sigma(\phi), \text{ при } y^T x + \theta = 0 \end{cases} \tag{4.2}$$

Позначимо через $\Pi_{y,\theta}$ гіперпростір визначений як $\{x | y^T x + \theta = 0\}$ та покладемо $H_{y,\theta}$ як відкритий півпростір $\{x | y^T x + \theta > 0\}$. Тоді за теоремою Лебега про мажоровану збіжність:

$$0 = \int\limits_{I_n} \sigma_\lambda(x) d\mu(x) = \int\limits_{I_n} \gamma(x) d\mu(x) = \sigma(\phi)\mu(\Pi_{y,\theta}) + \mu(H_{y,\theta})$$

для всіх ϕ, θ, y.

Тепер ми покажемо, що якщо міра всіх півплощин рівна 0 то і міра μ сама рівна 0. Це було б очевидно для невід'ємних мір, але ми маємо справу зі знакозмінними.

Зафіксуємо y. Для обмеженої вимірної функції h, визначимо лінійний функціонал F, наступним чином:

$$F(h) = \int\limits_{I_n} h(y^T x) d\mu(x)$$

та помітимо, що F обмежений функціонал на $L^\infty(\mathbb{R})$ оскільки μ скінчена знакозмінна міра. Нехай h - індикаторна функція на інтервалі $[\theta, \infty)$

($h(u) = 1$ якщо $u \geq \theta$ та 0 інакше), отже:

$$F(h) = \int\limits_{I_n} h(y^T x) d\mu(x) = \mu(\Pi_{y,-\theta}) + \mu(H_{y,-\theta}) = 0.$$

Аналогічно, $F(h) = 0$ якщо h це інидикаторна функція відкритого інтервалу (θ, ∞). За лінійністю $F(h) = 0$ для індикаторних функцій для довільного інтервалу, а отже і для довільної простої функції. Оскільки прості функції щільні в $L^\infty(\mathbb{R})$, то $F = 0$.

Зокрема для обмежених вимірних функцій $s(u) = \sin(m \cdot u)$ та $c(u) = \cos(m \cdot u)$:

$$F(s + ic) = \int_{I_n} \cos(m^T x) + i \sin(m^T x) d\mu(x) = \int_{I_n} \exp(im^T x) d\mu(x) = 0$$

для всіх m. Отже, перетворення Фур'є міри μ рівне нулю, а отже і $\mu = 0$.

Таким чином σ - дискримінаційна.

□

Вправа: показати, що rectified linear unit є дискримінаційною функцією.

Покажемо тепер як використовувати отримані результати для задач класифікації. Нехай m це міра Лебега на I_n. Позначимо $P_1, P_2, \dots P_k$ розбиття I_n на k неперетинних вимірних підмножин I_n. Визначимо вирушуючу функцію f:

$$f(x) = j, \ x \in P_j.$$

Маємо наступну теорему (Цибенко):

Теорема 2. *Нехай σ неперервна сігмоїдальна функція. Нехай f - фирішуюча функція для довільного скінченого вимірного розбиття I_n. Тоді для довільного $\varepsilon > 0$ існує скінчена сума у вигляді:*

$$G(x) = \sum_{j=1}^{N} \alpha_j \sigma(y_j^T x + \theta_j)$$

та множина $D \subset I_n$ така що $m(D) \geq 1 - \varepsilon$ та

$$|G(x) - f(x)| < \varepsilon, \ x \in D.$$

Доведення.

За теоремою Лузіна (довільна вимірна функція є неперервною майже на всій своїй області визначення, або нехай $f\colon [a, b] \to \mathbb{R}$ є вимірною, тоді для довільного $\varepsilon > 0$ існує компактна множина $E \subset [a, b]$ така, що функція f є неперервною на E та $\mu(E^c) < \varepsilon$, де E^c це доповнення). існує неперервна функція h та множина D, такі що $m(D) \geq 1 - \varepsilon$, що $h(x) = f(x)$ для всіх $x \in D$. Оскільки h неперервна, і за теоремою про апроксимацію ми можемо знайти функцію G у такому вигляді як вказано в умові теореми, що $|G(x) - h(x)| < \varepsilon$ для всіх $x \in I_n$. Тоді для всіх $x \in D$:

$$|G(x) - f(x)| = |G(x) - h(x)| < \varepsilon.$$

□

Таким чином бачимо, що похибку класифікації можна зробити як завгодно малою.