ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ІВАНА ФРАНКА
Факультет прикладної математики та інформатики
Кафедра обчислювальної математики

Курсова робота

Використання глибокого навчання для обернених задач

Виконав студент IV курсу групи
ПМп-41 напрямку підготовки
(спеціальності)
113 – "Прикладна математика"
Середович В.В.

Курівник: Музичук Ю.А.

Львів - 2021

Зміст

Вступ

TODO: Вступ про типи обернені задач та глибоке навчання
Оберненими задачами називають такі задачі, коли необхідно відновити параметри які характеризують деяку модель з використанням непрямих спостережень. До них можна віднести багото по відновленню зображень зменшення кількості шуму (deblurring) чи заповнення втрачених даних (inpainting).

1 Постановка задачі

Оберненими задачами будемо вважати такі задачі, в яких невідомим є $n-$ піксельне зображення $\boldsymbol{x} \in \mathbb{R}^n$ яке було отримане з m вимірювань $\boldsymbol{y} \in \mathbb{R}^m$ відповідно до рівняння

$$\boldsymbol{y} = \mathcal{A}(\boldsymbol{x}) + \boldsymbol{\varepsilon} \tag{1.1}$$

де $\mathcal{A}$ - це прямий оператор вимірювання та $\boldsymbol{\varepsilon}$ є певним вектором шуму. Метою задачі є відновлення x з y. Можна розглянути більш загальний випадок моделі неадитивного шуму, який має вигляд

$$\boldsymbol{y} = \mathcal{N}(\mathcal{A}(\boldsymbol{x})) \tag{1.2}$$

де $\mathcal{N}(\cdot)$ є прикладами вибірки з шумом.

Означення 1.1 *Відповідно до поняття, уведеного Жаком Адамаром, задачу 1.2 називають коректно поставленою, якщо вона задовольняє наступні умови:*

1. *Для кожного x розв'язок задачі існує.*
2. *Розв'язок є єдиний для кожного x.*
3. *Розв'язок є стійкий до малих варіацій величини x, тобто достатньо малим зміненням величини x відповідають як завгодно малі зміни величини y.*

Означення 1.2 *Задачу, яка не задовільняє хоча б одну із умов означення 1.1, називають некоректно поставленою.*

Отже, очевидно, що розглянута обернена задача є некоректно (або погано обумовленою), оскільки в ній порушуються умови означення 1.1. Така задача знаходження єдиного розв'язку, яка задовілняє спостереженням є складною або неможливою, за умови відсутності попередніх знаннь про дані.

Оцінку справжнього зображення $\boldsymbol{x}$ з $\boldsymbol{y}$ вимірювання називають задачею реконструкції зображення. Класичні підподи до реконструкції зображеннь припускають наявність деякої попередньої інформації про зображення, яку називають пріором. В якості пріору можуть виступати параметри гладкості, щільності та інші геометричні властивості зображення.

Отже метою даної роботи буде розв'язання таких обернених задач за допомогую алгоритмів глибокого навчання. Зокрема, будемо розглядати проблему зменшення кількості шуму у зображеннях.

2 Структура обернених задач

Якщо розподіл шуму відомий, x можна відновити розвязавши задачу оцінки максимальної ймовірності (Maximum Likelihood):

$$\hat{\boldsymbol{x}}_{\mathrm{ML}} = \arg\max_{\boldsymbol{x}} p(\boldsymbol{y} \mid \boldsymbol{x}) = \arg\min_{\boldsymbol{x}} -\log p(\boldsymbol{y} \mid \boldsymbol{x})$$

де $p(\boldsymbol{y} \mid \boldsymbol{x})$ це ймовірність спостереження $\boldsymbol{y}$ за умови якщо $\boldsymbol{x}$ є справжнім зображенням.

В залежності від умов задачі, можуть бути відомі попередні дані про те яким має бути x. Ці умови можуть бути використанні для формування задачі оцінки максимальної апостеріорної ймовірності (maximum a posteriori), що приводить до задачі

$$\hat{\boldsymbol{x}}_{\mathrm{MAP}} = \arg\max_{\boldsymbol{x}} p(\boldsymbol{x} \mid \boldsymbol{y}) = \arg - \max_{\boldsymbol{x}} p(\boldsymbol{y} \mid \boldsymbol{x}) p(\boldsymbol{x}) = \arg\min_{\boldsymbol{x}} -\ln p(\boldsymbol{y} \mid \boldsymbol{x}) - \ln p(\boldsymbol{x})$$

Для випадку білого гаусівського шуму, формулювання MAP дає:

$$\frac{1}{2}\|\mathcal{A}(\boldsymbol{x}) - \boldsymbol{y}\|_2^2 + r(\boldsymbol{x}) \tag{2.1}$$

де $r(\boldsymbol{x})$ є пропорційним до негативного логарифмічного пріора $\boldsymbol{x}$.. Прикладами такого підходу є регуляризація Тіхонова.

Задача маскимальної апостеріорної оцінки може використовуватись для реконструкції зображень, однак такий такий підхід може бути не таким ефективним, якщо розподіл шуму або прямий оператор є невідомі. Алгоритми основані на використанні машинного навчання дають змогу побороти більшість з цих труднощів, що робить їх ефективною альтернативою класичному підходу.

3 Огляд машинного навчання для розв'язування обернених задач

3.1 Контрольоване і некортнольоване навчання

Перший і найпоширеніший тип розв'язування оберених задач з використанням глибогоко навчання є контрольована інверсія. Ідея полягає у створенні співвідношення між датасетом справжніх зображень x та відповідними вимірюваннями y. Тобто ми можемо натренувати нейронну мережу приймати значення y та реконструювати оберенне значення x. Цей підхід є дуже ефективним, однак є чутливим до змін в опереторі вимірювання A.

Другим типом розв'язування обернених задач є неконтрольованого навчання. Він передбачає, що інформація про пари вхідної та вихідної інформації

x та y невідомі під час тренування. До нього можна віднести ситуації коли відомі тільки справжні зображення x або тільки результати вимірювання y.

Ці два підходи мають фундоментальні відміності і ця робота націлена саме на методи контрольованого навчання, тому що очікується, що вони дадуть кращі результати в порівнянні з класичними методами.

3.2 Класифікація навчання розв'язування обернених задач

TODO

4 Методи глибокого навчання для обернених задач

4.1 Autoencoder

Автоенкодером називають нейрону мережу яка навчається копіювати свої вхідні дані у вихідні. Така мережа має проміжний шар h, також відомий як латентний, який зберігає параметри необхідні для представлення вхідних даних. Таку нейрону мережу можна подати у складі двох частин: функції енкодера $h = f(x)$ та декодера який відтворює $r = g(h)$. Якщо автоенкодеру вдається навчитися просто відтворювати $g(f(x)) = x$ для всіх прикладів, то це не має особливої користі. Тому автоенкодери зазвичай обмежують таким чином щоб вони не могли відтворювати ідеальну копію вхідних даним.

4.2 Denoising autoencoder (DAE)

Класичні автоенкодери мінімізують деяку функцію:

$$L(\boldsymbol{x}, g(f(\boldsymbol{x})))$$

ду L це штрафна функція яка карає $g(f(\boldsymbol{x}))$ за відмінність від $\boldsymbol{x}$, таку як L^2 норму від їх різниці. Це призводить до того, що композиція функцій $g \circ f$ навчається бути функцією тотожнього відобрадення якщо для того є можливість. На відміну від цього, автоенкодер для видалення шуму, або denoising autoencoder (DAE) мінімізує

$$L(\boldsymbol{x}, g(f(\tilde{\boldsymbol{x}})))$$

де $\tilde{\boldsymbol{x}}$ є копією $\boldsymbol{x}$ який був пошкоджений деяким шумом. Отже, такий автоенкодер має відновити пошкодження, натомість простому відтвореню вхідних даних.

TODO? Навчання видалення шуму змушує f та g явно вивчати структуру даних для

4.3 Модель глибокої нейронох мережі для автоенкодера

Нехай маємо набір тренувальних даних:

$$(x^{(1)}, y^{(1)}), \quad (x^{(2)}, y^{(2)}), \quad ... \quad , (x^{(m)}, y^{(m)})$$

$$X = \begin{bmatrix} \vdots & \vdots & & \vdots \\ x^{(1)} & x^{(2)} & \dots & x^{(m)} \\ \vdots & \vdots & & \vdots \end{bmatrix} \quad x \in \begin{bmatrix} x_1 \\ \dots \\ x_n \end{bmatrix}$$

$$Y = \begin{bmatrix} \vdots & \vdots & & \vdots \\ y^{(1)} & y^{(2)} & \dots & y^{(m)} \\ \vdots & \vdots & & \vdots \end{bmatrix} \quad y \in \begin{bmatrix} y_1 \\ \dots \\ y_n \end{bmatrix}$$

де

- X - матриця в якій кожен $i-$ий стовпець є розгорнутим у вектор зображенням з деяким рівнем шум,
  $i = 1, \dots, m$
- Y - матриця в якії кожен $i-$ий стовпець є розгорнутим у вектором справжнього зображення
- m - кількість прикладів
- n - кількість характеристик в кожному прикладі (довжина розгорнутого в вектор зображення)

TODO : задача необхідно знайти... TODO: описати шари Для цього будемо використовувати алгоритм градієнтного спуску (ADAM?) необхідно мінімізувати ...

Алгоритм прямого поширення

$$\begin{aligned} z^{(l)} &= w^{(l)} a^{(l-1)} + b^{(l)} \\ a^{(l)} &= \sigma(z^{(l)}) \end{aligned} \tag{4.1}$$

де

- l - номер шару нейроної мережі, де $l = 1 \dots L$
- $n^{[l]}$ - кількість нейронів в l шарі
- $a^{(l)}$ - вектор стовбець з $n^{[l]}$ елементів (тобто $n^{[l]} \times 1$ матриця), активація нейронів на для шару l
- $b^{(l)}$ - вектор стовбець з $n^{[l]}$ елементів (тобто $n^{[l]} \times 1$ матриця), ваги зміщення

- $w^{(l)}$ - $(n^{[l]} \times n^{[l-1]})$ матриця ваг поміж шарами $l-1$ та l
- σ - це активаційна (стискуюча) функція, яку ми можемо прийняти як логістичну (для діапазону від 0 до 1) або tanh (для діапазону від -1 до 1), або будь-яку іншу диференційовану функцію.

Для кожного прикладу визначеми функцію витрат як середньо квадратичну похибку

$$L(a^{(l,i)}, y^{(i)}) = \frac{1}{2}\|a^{(l,i)} - y^{(i)}\|_{L_2}^2 = \frac{1}{2}\sum_{j=1}^{n}(a_j^{(l,i)} - y_j^{(i)})^2 \tag{4.2}$$

де $\|\cdot\|$ це евклідова відстань також відома як L_2 норма, y - справжне зображення, a - активація, l - номер шару

На всії прикладах обчислюємо штрафну функцію

$$J(\omega, b) = \frac{1}{m}\sum_{i=1}^{m} L(a^{(L,i)}, y^{(i)}) \tag{4.3}$$

Задача полягає в тому щоб знайти параметри $w \in \mathbb{R}^{n_x}, b \in \mathbb{R}$ що мінімізують функцію $J(w,b)$.

Далі обчислюємо похідну від функції середньо квадратичної похибки, останього шару нейронної мережі:

$$da^{(L,i)} = \frac{\partial L(a^{(L,i)}, y^{(i)})}{\partial a^{(L,i)}} = \sum_{j=1}^{n}(a_j^{(L,i)} - y_j^{(i)})$$

$$dz^{(L,i)} = \frac{\partial L(a^{(L,i)}, y^{(i)})}{\partial z^{(L,i)}} = \frac{\partial L(a^{(L,i)}, y^{(i)})}{\partial a^{(L,i)}}\frac{\partial a^{(L,i)}}{\partial z^{(L,i)}} = \sum_{j=1}^{n}(a_j^{(L,i)} - y_j^{(i)})\sigma'(z^{(L,i)});$$

$$dw^{(L,i)} = \frac{\partial L(a^{(L,i)}, y^{(i)})}{\partial w^{(L,i)}} = dz^{(L,i)}\frac{\partial z^{(L,i)}}{\partial w^{(L,i)}} = dz^{(L,i)}a^{(L-1,i)};$$

$$db^{(L-1,i)} = \frac{\partial L(a^{(L,i)}, y^{(i)})}{\partial b^{(L,i)}} = dz^{(L,i)}\frac{\partial z^{(L,i)}}{\partial b^{(L,i)}} = dz^{(L,i)}$$

Перепишемо похідні в більш компактному вигляді:

$$\frac{\partial J(w,b)}{\partial W^{(l)}} = \frac{1}{m}\sum_{i}^{m}\delta^{(l,i)}\left(a^{(l-1,i)}\right)^{\top}$$

$$\frac{\partial J(w,b)}{\partial b^{(l)}} = \frac{1}{m}\sum_{i}^{m}\delta^{(l,i)}$$

$$\delta^{(l,i)} = \begin{cases} \sum_{j=1}^{n} \left(a_j^{(L,i)} - y_j^{(i)}\right)\sigma'(z^{(L,i)}), & \text{якщо } l = L \\ \left((W^{(l)})^\top \delta^{(l+1,i)}\right)\sigma'(z^{(l,i)}), & \text{якщо } l < L \end{cases}$$

В залежності від активаційної функції значення похідної для алгоритму зворотнього поширення буде відрізнятись. Розглянемо наступні варіанти активаційних функцій:

- **Sigmoid** (Logistic function)

$$\begin{aligned} a &= \sigma(z) = \frac{1}{1+e^{-z}} \\ \sigma'(z) &= a(1-a) \end{aligned} \tag{4.4}$$

- **ReLU** (Rectified Linear Units)

$$\begin{aligned} a &= \sigma(z) = \max(0, z) \\ \sigma'(z) &= \begin{cases} 0, z < 0 \\ 1, z \geq 0 \end{cases} \end{aligned} \tag{4.5}$$

- **Tanh** (Hyperbolic tangent)

$$\begin{aligned} a &= \sigma(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} \\ \sigma'(z) &= 1 - a^2 \end{aligned} \tag{4.6}$$

5 Реалізація та аналіз

6 Герерація шуму

TODO

6.1 Реалізація

TODO

6.2 Аналіз

TODO

7 Висновок

TODO

Бібліографія

[1] Gregory Ongie та ін. *Deep Learning Techniques for Inverse Problems in Imaging*. 2020. arXiv: 2005.06001 [eess.IV].

[2] Jonas Adler та Ozan Öktem. "Solving ill-posed inverse problems using iterative deep neural networks". в: *Inverse Problems* 33.12 (листоп. 2017), с. 124007. ISSN: 1361-6420. DOI: 10.1088/1361-6420/aa9581. URL: http://dx.doi.org/10.1088/1361-6420/aa9581.