Звіт з лабораторної роботи №1
Студента 2 курсу магістратури
Групи "статистика"
Варіант №4

Горбунов Даніел Денисович

3 жовтня 2022 р.

Вступ.

У даній роботі реалізовано модель щільної нейронної мережі з оптимізацією на основі міні-пакетів. Також розглянуто та реалізовано окрім класичного градієнтного спуску його навантаження: моментний градієнтний спуск, Root Mean Square Propagation, Adaptive Moment Estimation. Результати роботи мережі продемонстровано візуальними та аналітичними методами.

Постановка задачі.

Маємо таблицю з $n = 1000$ спостережень. Кожне спостереження може відноситися до одного з двох класів. Потрібно побудувати таке рішуче правило, яке б могло визначати більш-менш точно клас, до якого належить відповідне спостереження. Зобразимо дані на площині:

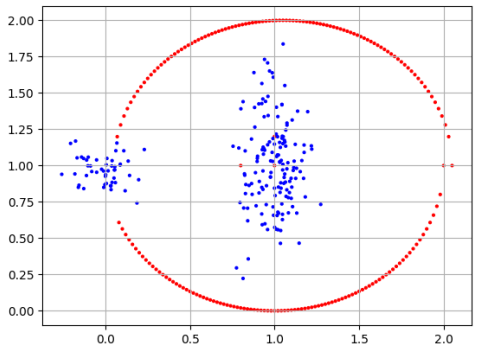

Рис. 1: Діаграма розсіювання даних.

Точки з синього класу (далі – з нульового) за формою нагадують точки, згенеровані з різних двовимірних нормальних розподілів. Точки з червоного класу (далі – з першого) утворюють дві дуги, які в сукупності дуже нагадують коло. Будемо вважати, що класифікатор добре працює, якщо розподіл про класам є точним та геометричні властивості відтворюються відносно непогано.

Хід роботи.

Зауваження з оптимізації функціонала втрат та утворення тренувальної та тестової вибірок.

Надалі будемо працювати з нейронними мережами, де оптимізація відбувається по міні-пакетам. Кількість пакетів буде 20 штук (тобто в кожному міні-пакеті маємо по 17 спостережень, чому так – переконаємося далі). Тренувальна те тестова вибірки утворилися простим випадковим відбором, з обсягами $n \cdot 0.85 = 400 \cdot 0.85 = 340$ та $n \cdot 0.15 = 60$ відповідно. Якість прогнозу нейронної мережі, звісно, не обмежиться лише функціоналом втрат, а й висновками про множини рішень та значення частоти помилок класифікації. Ми не будемо розглядати оптимізацію з динамічним параметром навчання (тобто значення є сталою величиною для всіх епох тренування моделі). Кількість епох, якщо не уточнюється, становить 10000 одиниць.

Оптимізація на основі класичного градієнтого спуску.

Розглянемо нейронну мережу з наступною структурою:

$$X \to \{4 \text{ neurons}\} \to^{\tanh} \{5 \text{ neurons}\} \to^{\tanh} \{4 \text{ neurons}\} \to^{\tanh} \{2 \text{ neurons}\} \to^{\tanh} \{1 \text{ neuron}\} \to^{\text{sigmoid}} \hat{Y}$$

Тобто всього три прихованих шари. Зокрема всі шари, окрім останнього, мають тангенс гіперболічний в якості активаційної функції. Прогноз утворюється обчисленням останньої активації за допомогою сигмоїди. Спробуємо зробити підгонку параметрів для параметра навчання $\alpha = 0.01$. Маємо такі результати підгонки:

```
Усереднені втрати на тренувальних даних:
Epoch #1000 :        Loss = 2.5728066421761384
Epoch #2000 :        Loss = 0.414329740747135
Epoch #3000 :        Loss = 0.2677049142793688
Epoch #4000 :        Loss = 0.22102255639083862
Epoch #5000 :        Loss = 0.19276259409746185
Epoch #6000 :        Loss = 0.1665551107117495
Epoch #7000 :        Loss = 0.14538943190360765
Epoch #8000 :        Loss = 0.13222852509954378
Epoch #9000 :        Loss = 0.12294387946188132
Epoch #10000:        Loss = 0.1813644432734783
Таблиця спряженості на тренувальних даних:
[[171   0]
 [  1 168]]
Таблиця спряженості на тестових даних:
[[29  0]
 [ 2 29]]
```

Як на перший погляд, прогнозує класифікатор на основі натренованої мережі непогано. Три помилки маємо, наприклад, для тих точок, які або дуже близько розміщені біля точок з іншого класу, або ж всередині хмарини з точок, як могли побачити з попереднього рисунка. Продемонструємо множини рішень класифікатора далі:

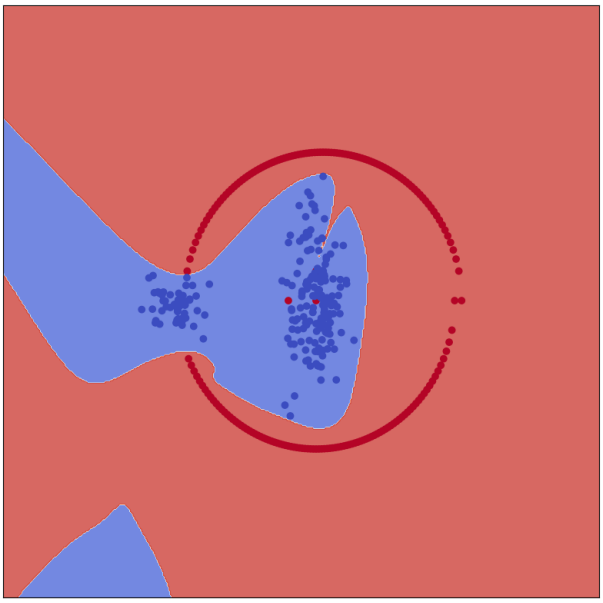

Рис. 2: Множини рішень на основі першого класифікатора. Іще розміщені всі спостереження.

Важко говорити щось про шматок синої області у лівому нижньому куті. Якщо припустити, що поза колом істиним є червоний клас, то це погано, наприклад. Але попередньо ми не знаємо про природу даних поза досліджуваною вибіркою, тому акцентувати увагу на цьому не будемо. Однак зауважимо, що внутрішня область рішень, що відносить до синього класу, здається неприродною – більше пристосованою до розміщених точок. В ідеалі була б якась майже опукла фігура, чого тут і не видно. Тому є підозра, що модель перевчилася.

Отже є ідея зменшення вимірності моделі, тобто зменшити кількість шарів та нейронів:

$$X \to \{4 \text{ neurons}\} \to^{\tanh} \{3 \text{ neurons}\} \to^{\tanh} \{2 \text{ neurons}\} \to^{\tanh} \{1 \text{ neurons}\} \to^{\text{sigmoid}} \hat{Y}$$

Чи справді це дало кращий результат? Переконаємося далі:

```
Epoch #1000 :         Loss = 0.6864385582806146
Epoch #2000 :         Loss = 0.4521220781541563
Epoch #3000 :         Loss = 0.4123584924750722
Epoch #4000 :         Loss = 0.39337485713668285
Epoch #5000 :         Loss = 0.37801555647709717
Epoch #6000 :         Loss = 0.36200926937803846
Epoch #7000 :         Loss = 0.3478067276807413
Epoch #8000 :         Loss = 0.3382448137965123
Epoch #9000 :         Loss = 0.3320201232606076
Epoch #10000:         Loss = 0.32775411671939814
Таблиця спряженості на тренувальних даних:
[[171   0]
 [  1 168]]
Таблиця спряженості на тестових даних:
[[29  0]
 [ 2 29]]
```

Так, втрати трохи вищі за попередні, однак якісь класифікації не змінилася (принаймні на відібраних даних). Ще більше радості доставляє візуалізація рішень:

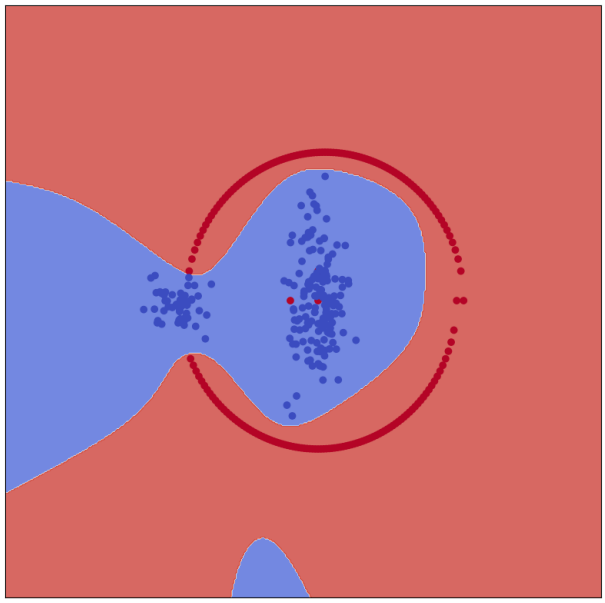

Рис. 3: Множини рішень на основі другого класифікатора. Інопланетна істота вітається.

Така картина виглядає більш природно. Аномальний шмат множини рішення до синього класу, звісно, залишився, але хай так, бо базовий ”тренд” модель вхопила.

Оптимізація на основі моментного градієнтого спуску.

Спробуємо покращити попередню модель, взявши до уваги трюк зі експоненційно зваженими середніми. Оберемо параметр затухання β_1 рівним 0.9. Архітектура моделі та її гіперпараметри залишаємо такими ж, як і раніше. В результаті:

```
Epoch #1000 :         Loss = 0.5822054866907165
Epoch #2000 :         Loss = 0.4155432571915944
Epoch #3000 :         Loss = 0.37986641025536333
Epoch #4000 :         Loss = 0.3608647152680183
Epoch #5000 :         Loss = 0.34887665951777785
Epoch #6000 :         Loss = 0.34079131957884196
Epoch #7000 :         Loss = 0.3350436042470838
Epoch #8000 :         Loss = 0.3307779908700198
Epoch #9000 :         Loss = 0.32750411197977863
Epoch #10000:         Loss = 0.3249261139749029
Таблиця спряженості на тренувальних даних:
[[171   0]
 [  1 168]]
Таблиця спряженості на тестових даних:
[[29  0]
 [ 2 29]]
```

Більш цікавим виявиться наступний рисунок. Аномальної множини, яка була помітна у попередніх моделях, відсутня на ньому (принаймні в околі даних). Внутрішня фігура вже не так сильно нагадувала коло, як нещодавно, але загальні форми дотримані.

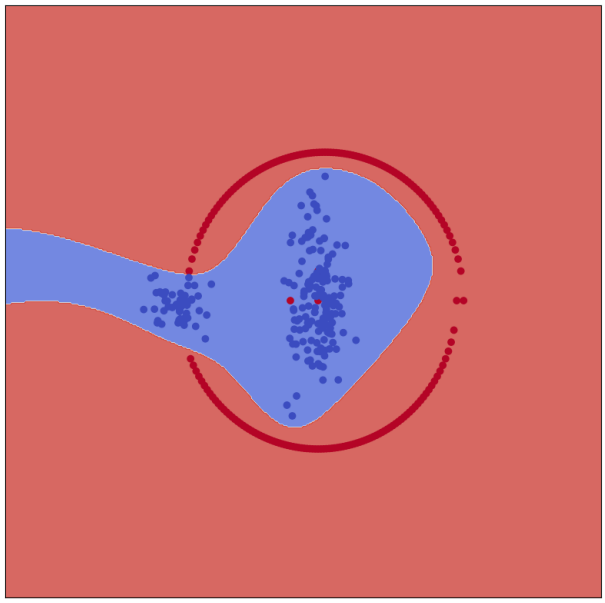

Рис. 4: Множини рішень на основі першого класифікатора. Шия та голова.

Оптимізація на основі RMS Propagation.

А тепер замість техніки моментів застосуємо техніку нормування за допомогою тих самх середніх. Параметр затухання β_2 поклали рівним 0.99.

```
Epoch #1000 :         Loss = 3.624139702827847
Epoch #2000 :         Loss = 0.5357623623604777
Epoch #3000 :         Loss = 0.3923479266225075
Epoch #4000 :         Loss = 0.3418406260360364
Epoch #5000 :         Loss = 0.32074268257535893
Epoch #6000 :         Loss = 0.3123043662949637
Epoch #7000 :         Loss = 0.30924739924852174
Epoch #8000 :         Loss = 0.30816648733951324
Epoch #9000 :         Loss = 0.3077700700404102
Epoch #10000:         Loss = 0.30762239523691354
Таблиця спряженості на тренувальних даних:
[[171   0]
 [  1 168]]
Таблиця спряженості на тестових даних:
[[29  0]
 [ 2 29]]
```

Одразу переходимо до візуалізації:

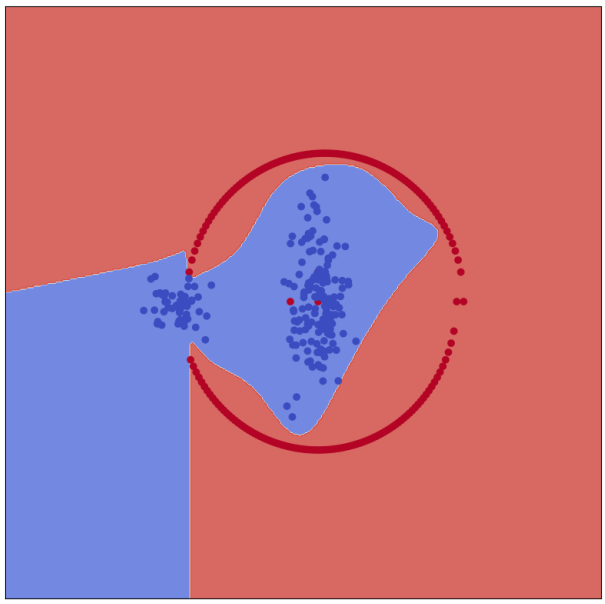

Рис. 5: Множини рішень на основі першого класифікатора. Sandy Cheeks.

Назовні маємо тепер щось схоже на конус. Всередині кола ніякої опуклості немає, хоча умовний еліпс можна розмістити в отриману фігуру. Хоча є шанс, що "догенеровані" точки попадуть у потрібну множину рішень.

Оптимізація на основі ADAM.

Застосуємо оптимізацію ADAM для згаданої раніше моделі. Параметри затухання β_{moment} та β_{rmsprop} поклали рівним 0.9 та 0.99 відповідно.

```
Epoch #1000 :         Loss = 0.7583770694632755
Epoch #2000 :         Loss = 0.4359684626684036
Epoch #3000 :         Loss = 0.36664635215413266
Epoch #4000 :         Loss = 0.3396572793640159
Epoch #5000 :         Loss = 0.32625463610437294
Epoch #6000 :         Loss = 0.318505802361172
Epoch #7000 :         Loss = 0.31373738057963757
Epoch #8000 :         Loss = 0.31077970464196353
Epoch #9000 :         Loss = 0.3090248859659832
Epoch #10000:         Loss = 0.30808277278902596
Таблиця спряженості на тренувальних даних:
[[171   0]
 [  1 168]]
Таблиця спряженості на тестових даних:
[[29  0]
 [ 2 29]]
```

Одразу переходимо до візуалізації:

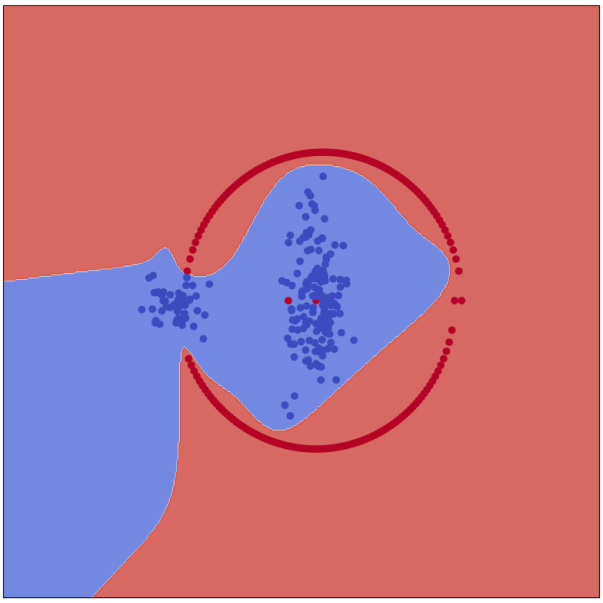

Рис. 6: Множини рішень на основі першого класифікатора. Sandy Cheeks.

Ситуація майже як і в попередньому випадку, хоча внутрішня множина рішень виглядає порівняно краще, ніж мали за RMSProp.

Висновки.

Найкращі результати вийшли для нейронної мережі з меншою розмірністю, на основі класичного градієнтного спуску або техніки ADAM. Насправді варто було б зробити наступні речі:

1. Розглянути моделі із затуханням параметра навчання.
2. Вибрати наближенно вдалі параметри в оптимізаторах за допомогою, наприклад, пошуку на гратці.
3. Розглянути підгонки з іншими значеннями зерна в генераторі псевдовипадкових чисел.

Незважаючи на це, результати вийшли задовільними.

Програмний код.

Програмну реалізацію нейронної мережі, що використовувалася в рамках цієї роботи, можна знайти за посиланням:
`https://github.com/MilkyCousin/Python-Probability-and-Statistics/tree/master/neural_networks_2mag`