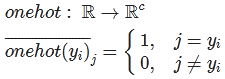
**Лабораторна робота 3 - Логістична регресія.**

У цій роботі Ви побудуєте логістичну регресію для класифікації зображень рукописних символів за датасетом MNIST.

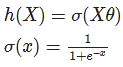
Реалізуйте методи з позначкою #TODO класу LogisticRegression:

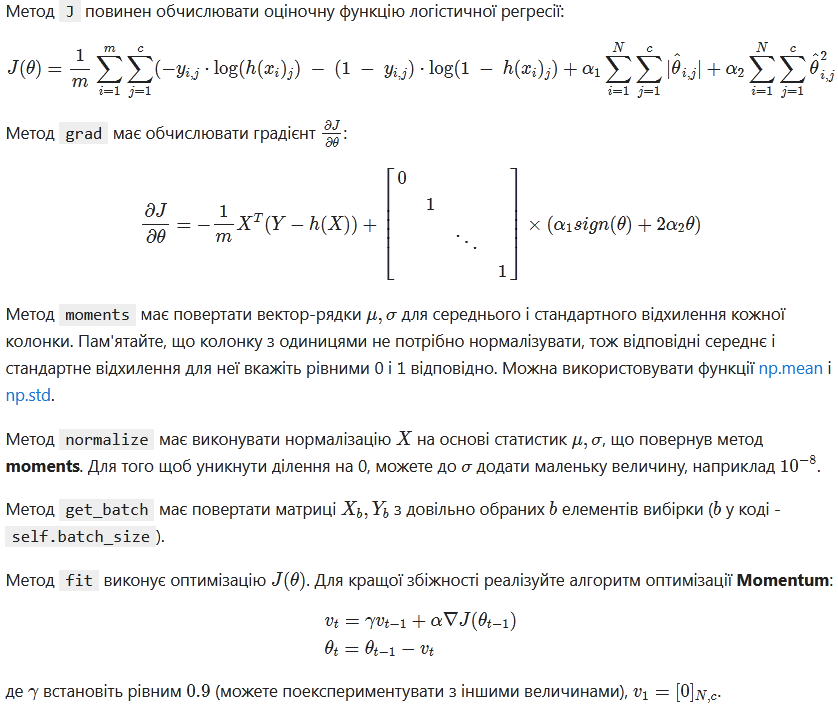
Метод preprocess повинен додавати колонку з одиниць у матрицю X. Опціонально – додайте поліноміальні або будь-які інші нелінійні ознаки.

Метод onehot повинен виконувати onehot-перетворення:

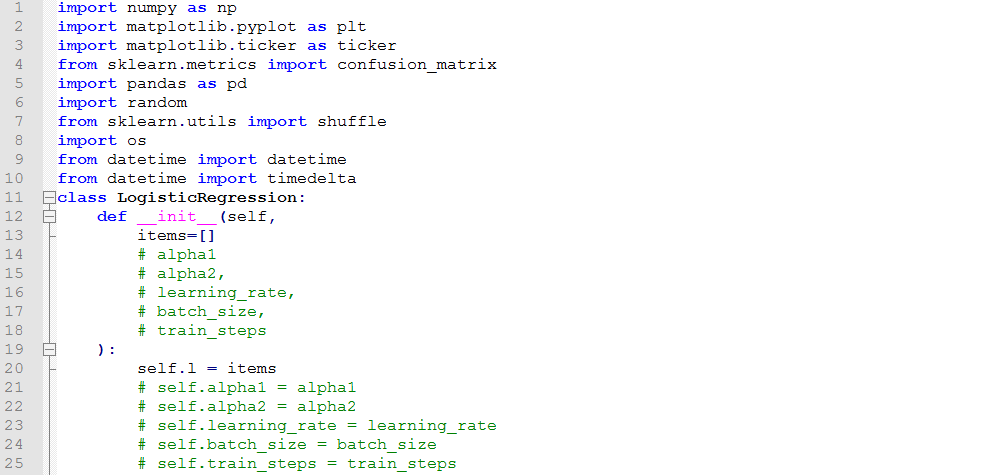


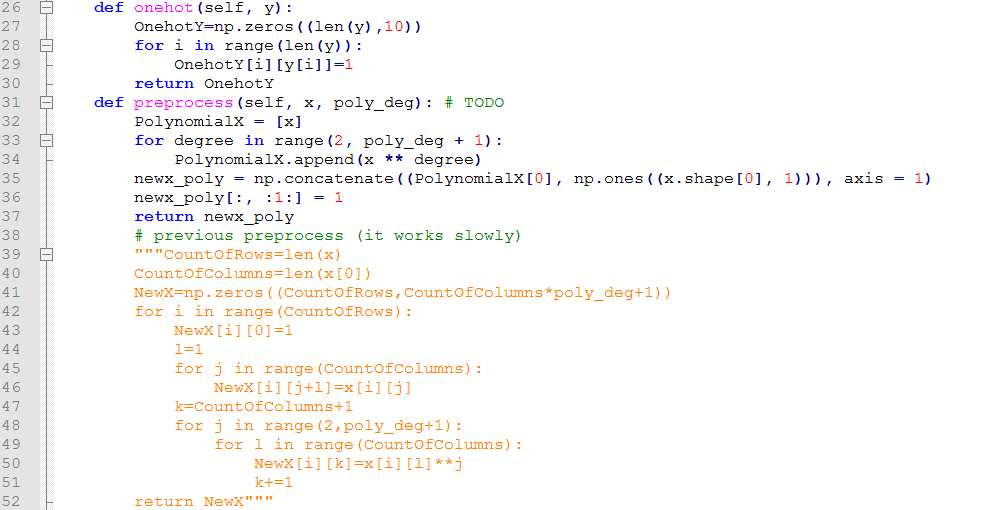
де с – кількість класів. Метод h - гіпотеза:

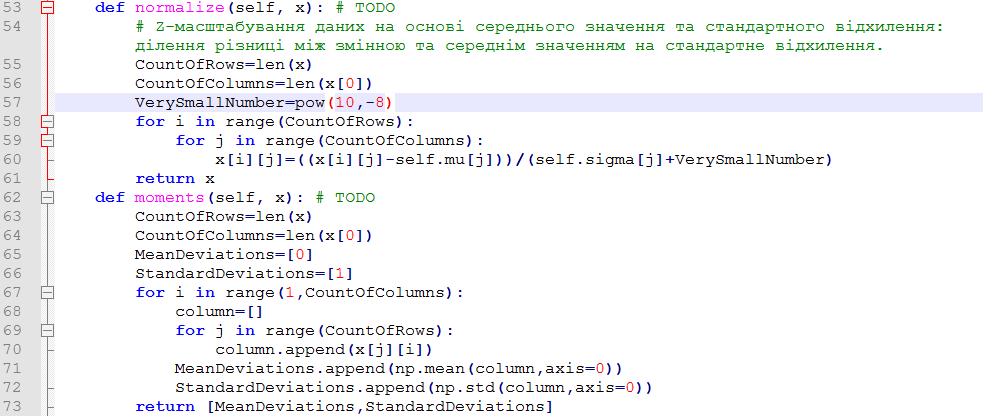


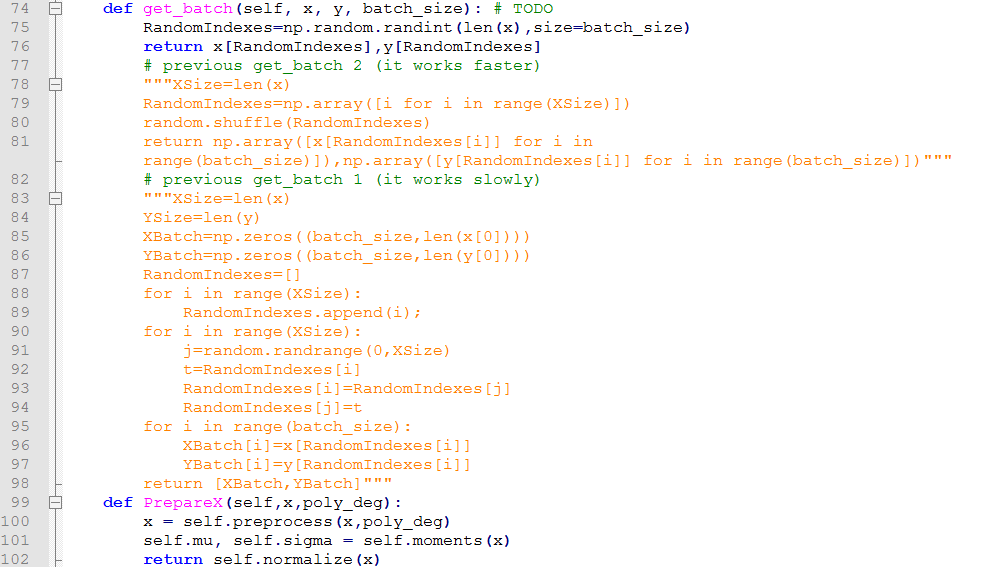


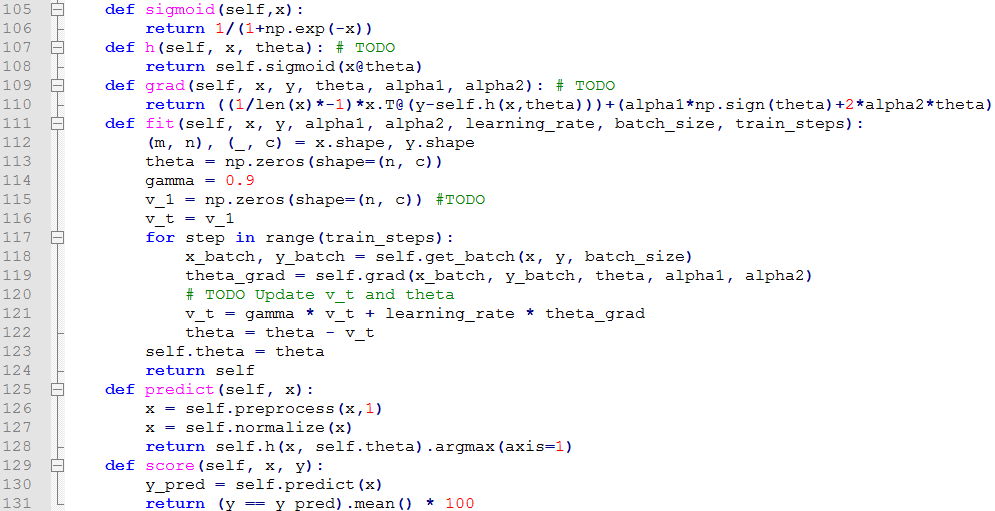
**Код класу LogisticRegression:**











**Опис проведених досліджень**

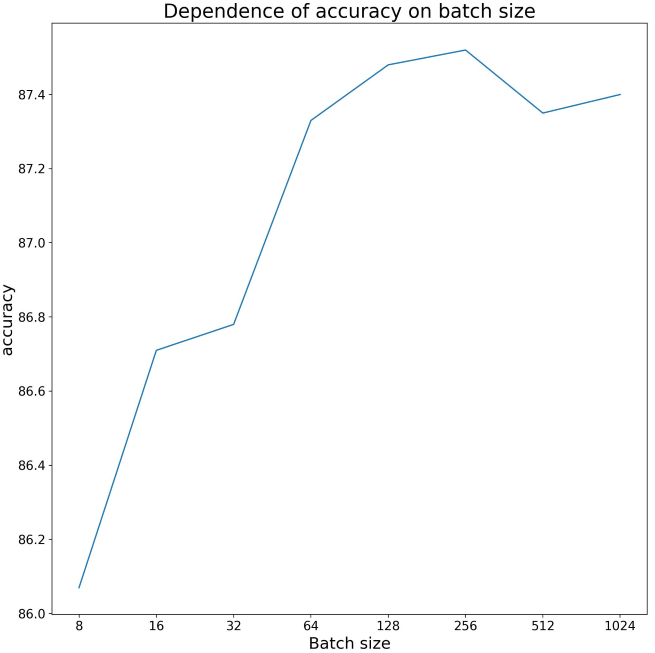
Запустивши код із стандартними параметрами ми отримали наступний вивід:



У рамках даної лабораторної роботи ми також виконали численні експерименти з різними значеннями таких основних параметрів моделі як коефіцієнти alpha1 та alpha2, коефіцієнт швидкості навчання (learning rate), розмір частини вибірки (batch\_size), кількість кроків навчання (train\_steps), будували також поліноміальну логістичну регресію, аналізуючи параметр poly\_deg. Вони підбирались так, щоб побачити межу між недостатнім навчанням та перенавчанням.

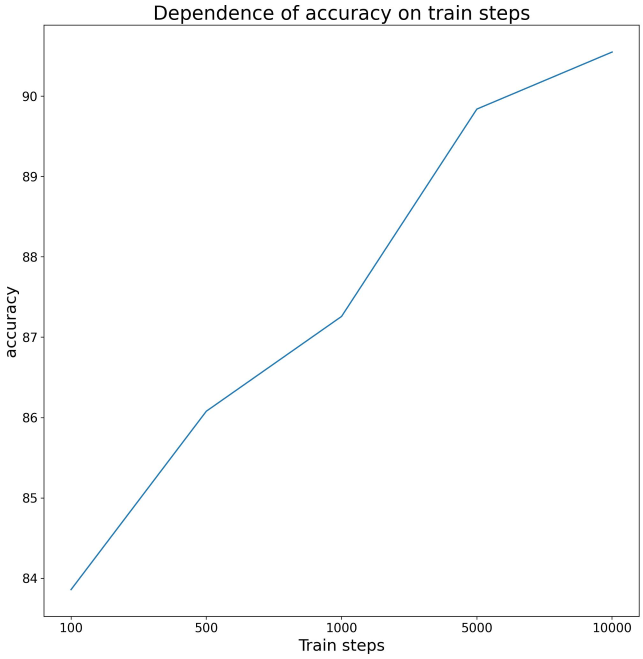
Ми створили декілька версій методів preprocess та get\_batch, задіявши під час реальних випробувань їхні найбільш швидкодіючі версії. Також слід відмітити зміну конструктора та виокремлення деяких підготовчих інструкцій у процедуру PrepareX з метою оптимізації часових витрат на попередню підготовку даних, яка виконується лише один раз перед певним набором експериментів. Окрім тестів з різними параметрами програма також містить код для визначення часових витрат, фіксації результатів у вигляді графіків, діаграм різних типів та текстового csv-файлу, який при необхідності може бути проаналізований табличним процесором.

Тестування створеного рішення проводилось на ПК з процесором  
Intel Core i5-6600 без використання окремого GPU та хмарних сервісів. Тому деякі набори експериментів виконувались у послідовності за спаданням обчислювальної складності, використовуючи в кожній наступній серії експериментів оптимальні параметри з попередніх. Спочатку ми проаналізували залежність score від різних розмірів частини вибірки (batch\_size):

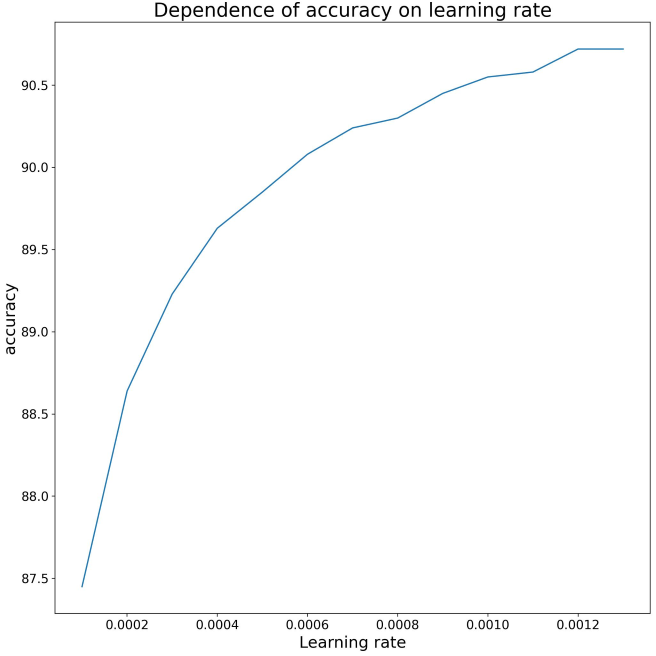


Можна зробити висновок, що найкращим значенням є 256, проте при подальших експериментах, можливо, знайдеться більш кращий розмір.

Після цього, вже з використанням кращого batch\_size, був проаналізований параметр train\_steps (кількість кроків навчання). Звісно, чим він більший, тим точнішою є модель, проте визначення можливої межі початку перенавчання потребує значних обчислювальних потужностей або великих часових витрат:



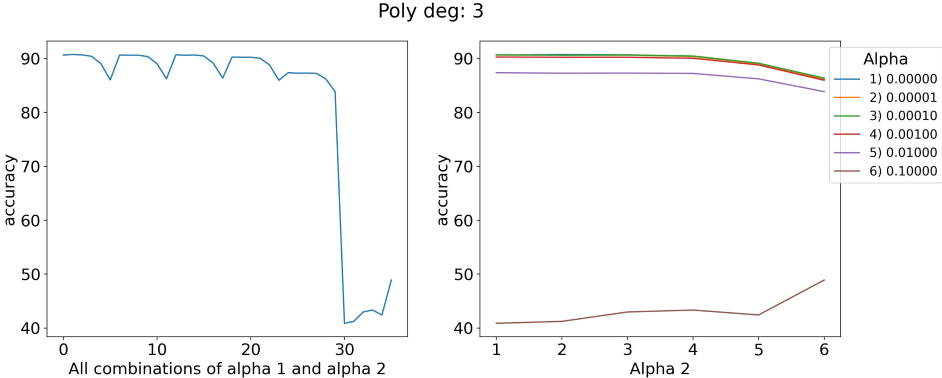
Після цього були досліджені різні значення learning\_rate (коефіцієнт швидкості навчання):

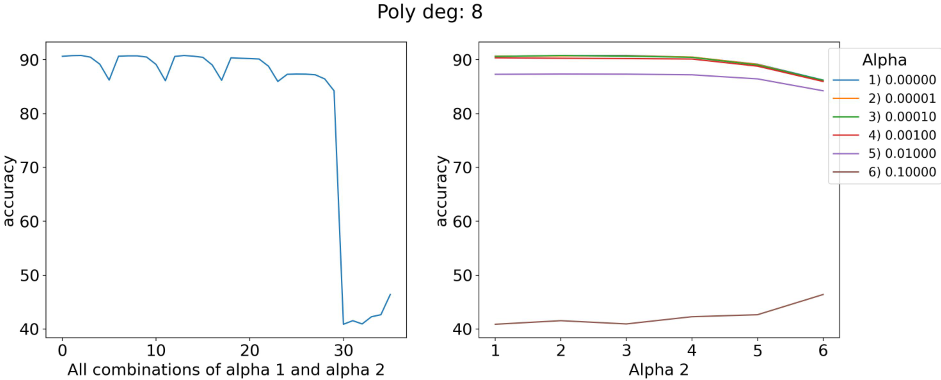


Найкращим значенням із досліджених виявилось 0.0012. Можна стверджувати, що його збільшення призводить до покращення моделі, проте визначення межі, де починається стрімке перенавчання, потребує більшої кількості додаткових експериментів.

Після цього експериментальним шляхом було встановлено, що коефіцієнт поліноміальності (poly\_deg) більший ніж 5 призводить до перенавчання, тобто до збільшення score. Ми намагались підібрати його у різних комбінаціях разом із параметрами alpha1 та alpha2, отримавши схожі один на одного графіки та діаграми:

|  |  |
| --- | --- |
|  |  |





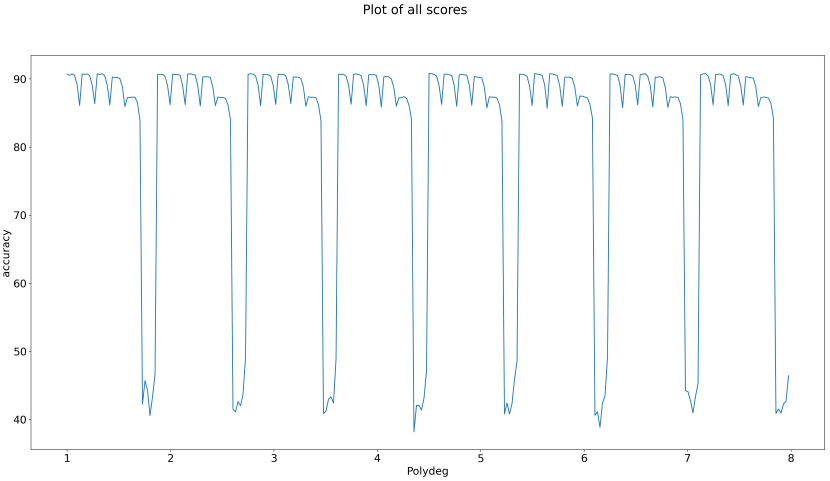
|  |  |
| --- | --- |
|  |  |

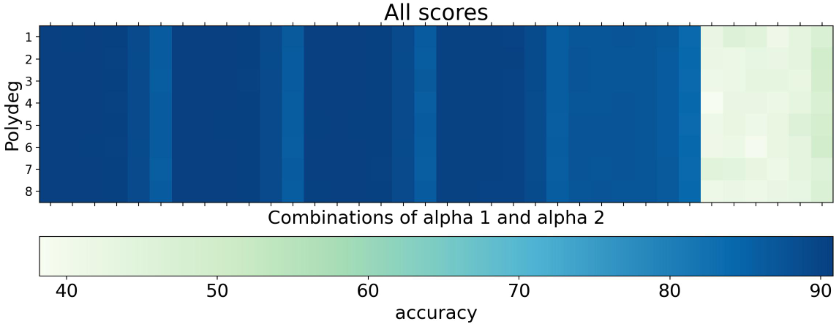
Можна зробити висновок, що оптимальні значення alpha1 та alpha1 знаходяться в межах до 0.01, вище – стрімке зменшення якості моделі, проте найкращим значенням alpha1 та alpha2 виявився 0, poly\_deg – 5. Це свідчить про негативний вплив коефіцієнтів реґуляризації на якість моделі, а поліноміальність покращує точність лише в певній мірі.

Найгірше значення accuracy 38.21000% було отримано при таких параметрах:

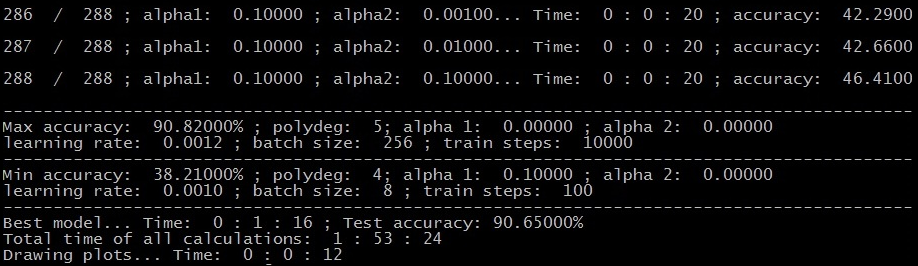
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| poly\_deg | alpha1 | alpha2 | learning\_rate | batch\_size | train\_steps |
| 4 | 0.10000 | 0.00000 | 0.0010 | 8 | 100 |

Наведемо глобальний графік та діаграму типу matshow для всіх комбінацій коефіцієнтів (результат пошуку параметрів alpha1 та alpha2 разом з poly\_deg):





Після завершення роботи програми ми отримуємо наступний вивід на екран:



Візуалізація матриці помилок для кращої моделі:

