

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КІЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО”



Системи штучного інтелекту

Логістична регресія

Практична робота #3

Виконав:
Олександр Ковальов

Група:

ІМ-51МН

Курс:

1

1 Логістична регресія

1.1 Реалізація логістичної регресії

1.1.1 Ініціалізація параметрів (parameters_initialization)

Функція ініціалізує ваги W невеликими випадковими значеннями та зсув b нулем.

```
1 def parameters_initialization(m):
2     W = np.random.randn(1, m) * 0.01
3     b = 0.0
4     return W, b
```

1.1.2 Пряме поширення (forwardPropagate)

Функція `forwardPropagate` обчислює лінійну комбінацію входів та ваг (z) і застосовує до неї сигмоїдну функцію активації для отримання прогнозу (y_{hat}).

```
1 def forwardPropagate(X, W, b):
2     z = np.dot(W, X.T) + b
3     y_hat = 1 / (1 + np.exp(-z))
4     return z, y_hat
```

1.1.3 Функція втрат (cost)

Розраховує усереднену бінарну перехресну ентропію, яка вимірює розбіжність між прогнозованими значеннями та істинними мітками.

```
1 def cost(n, y_hat, y_true):
2     ep = 10E-10
3     y_hat_flat = y_hat.reshape(-1)
4     J = - (1.0 / n) * np.sum(y_true * np.log(y_hat_flat + ep) + (1 - y_true) * np.log(1 - y_hat_flat + ep))
5     return J
```

1.1.4 Зворотне поширення (Обчислення градієнтів, backwardPropagate)

Обчислює градієнти (похідні) цільової функції відносно ваг та зсуву, які вказують напрямок для оновлення параметрів.

```
1 def backwardPropagate(n, X, y_hat, y_true):
2     y_hat_row = y_hat.reshape(1, -1)
3     y_true_row = y_true.reshape(1, -1)
4     dz = y_hat_row - y_true_row
5     dW = (1.0 / n) * np.dot(dz, X)
6     db = (1.0 / n) * np.sum(dz)
7     return dW, db
```

1.1.5 Оновлення параметрів (update)

Виконує крок градієнтного спуску, оновлюючи ваги W та зсув b для мінімізації цільової функції.

```
1 def update(lr, dW, db, W, b):
2     W = W - lr * dW
3     b = b - lr * db
4     return W, b
```

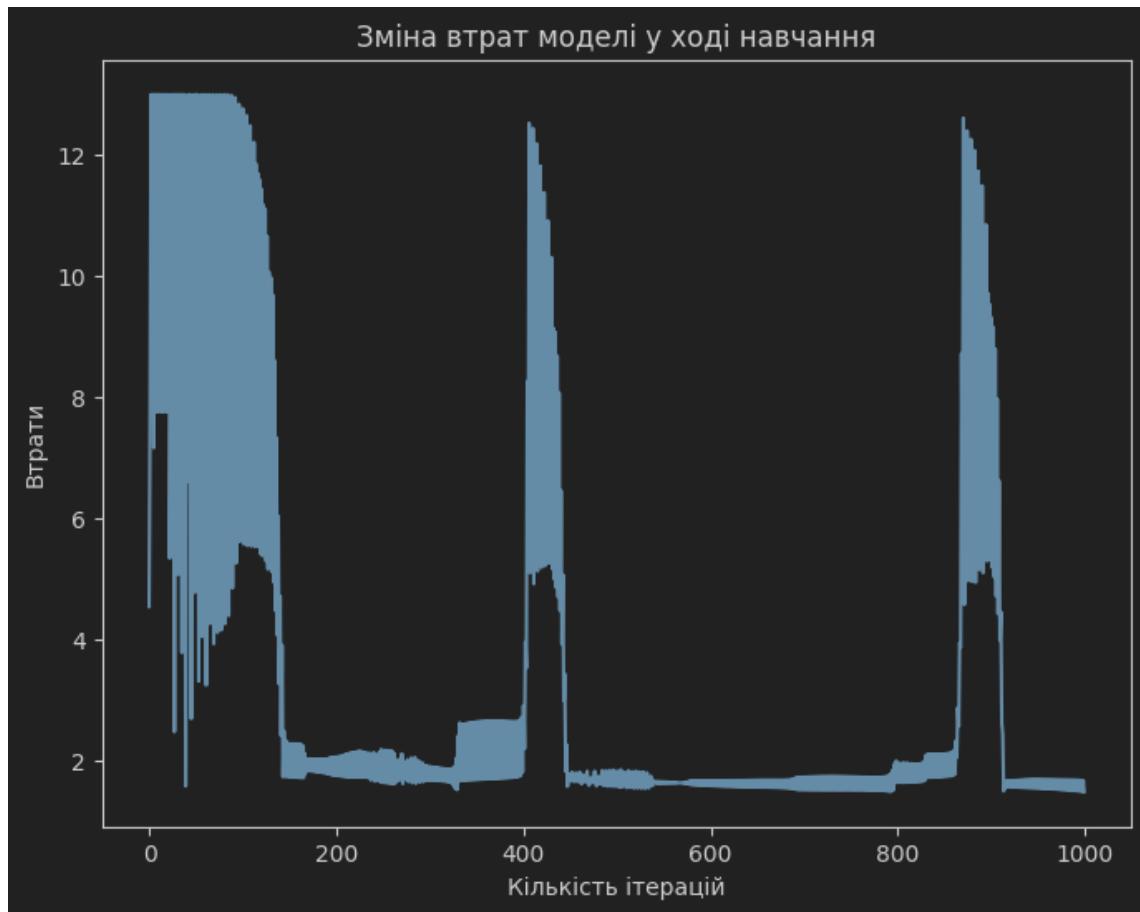
1.2 Результати експериментів

Модель була навчена з такими гіперпараметрами:

1. Швидкість навчання (α): 0.001
2. Кількість ітерацій (`n_iters`): 1000

Показник	Значення
Точність на тестовій вибірці	0.8859 (88.59%)
Фінальна втрата на тестовій вибірці	1.6788
Фінальна втрата на навчальній вибірці	1.6735 (на 980-й ітерації)

Графік зміни втрат під час навчання показує, що цільова функція зменшується, але дуже шумно та нестабільно.



Це, ймовірно, пов'язано з тим, що ознаки в наборі даних не були масштабовані і мають дуже різні діапазони значень.

Дослідження впливу гіперпараметрів показує, що швидкість навчання (`lr`) є критично важливим параметром. При низькому значенні, наприклад $lr = 0.00001$, модель навчається надто повільно, і за

1000 ітерацій цільова функція не встигає досягти мінімуму. Це призводить до недонавчання (underfitting) та низької точності. З іншого боку, занадто висока швидкість навчання, як $lr = 0.01$, змушує алгоритм "перестрибувати" мінімум, через що втрати хаотично коливаються або зростають, роблячи навчання неможливим.

Аналогічно, кількість ітерацій (`n_iters`) має бути достатньою для збіжності моделі. Якщо зупинити навчання завчасно (напр., при 200 ітераціях), модель залишиться недонавченою, оскільки не встигне оптимально налаштувати свої ваги. І навпаки, значне збільшення кількості ітерацій, скажімо до 10000, може дещо покращити точність, але після досягнення плато це лише збільшує час навчання без суттєвого результату.

1.3 Допомога

Робота була виконана самостійно.

1.4 Висновки

Проведена робота демонструє повний цикл реалізації логістичної регресії з нуля. Аналіз гіперпараметрів дозволяє зробити висновок, що їх вибір – це завжди компроміс. Крім того, висока нестабільність цільової функції під час навчання вказує на необхідність попередньої обробки даних, зокрема масштабування ознак. Стандартизація або нормалізація даних, найімовірніше, зробила б процес збіжності більш плавним і стабільним, потенційно підвищивши підсумкову точність. Таким чином, успішне навчання моделі вимагає як ретельного підбору гіперпараметрів, так і якісної підготовки даних.