Практична робота №2. Побудова та дослідження моделі логістичної регресії для прогнозування результатів медичних тестів

Предмет: Машинне навчання
Рівень: 4 рік навчання
Датасет: [Healthcare Dataset] Link
Мета: Реалізувати модель логістичної регресії "з нуля" для класифікації результатів медичних тестів на основі даних про пацієнтів. Дослідити вплив різних методів оптимізації та регуляризації на якість моделі.

1. Постановка задачі

Ми маємо дані про пацієнтів. Наша ціль — передбачити результат медичного тесту (Test Results) на основі медично-демографічних факторів. Оскільки цільова змінна може мати 2 або 3 класи, ми розглядаємо два варіанти на вибір: бінарну класифікацію (сигмоїда) та багатокласову класифікацію (softmax). Якщо цільова змінна є бінарною ("Normal" vs. "Abnormal"), ми ігноруємо "Inconclusive" та формулюємо задачу бінарної класифікації та використовуватимемо логістичну регресію (*в цій практичні роботі варто обмежитися бінарною змінною; за реалізацію багатокласової задачі будуть додаткові бали*).

Модель: Імовірність того, що і-й приклад належить до позитивного класу ("Normal"), визначається логістичною (сигмоїдальною) функцією:

$$P(y_i = 1|\mathbf{x}_i, \boldsymbol{\theta}) = \sigma(\mathbf{x}_i^T \boldsymbol{\theta}) = \frac{1}{1 + e^{-\mathbf{x}_i^T \boldsymbol{\theta}}}$$

де:

- $y_i \in 0, 1$ — бінарна цільова змінна.
- $\mathbf{x}_i$— вектор ознак для і-го спостереження (з доданою одиницею для intercept).
- $\boldsymbol{\theta}$ — вектор параметрів моделі, який нам необхідно навчити.

Завдання: Знайти вектор параметрів $\boldsymbol{\theta}$, який мінімізує функцію втрат — негативну логарифмічну правдоподібність (Log Loss / Binary Cross-Entropy).

$$J(\boldsymbol{\beta}) = -\frac{1}{m} \sum_{i=1}^{m} [y_i \log(p_i) + (1 - y_i) \log(1 - p_i)]$$

де $p_i = \sigma(\mathbf{x}_i^T \boldsymbol{\theta})$, а m — кількість спостережень.

Метод: Для мінімізації функції втрат використовуватимемо ітеративні методи оптимізації:
1. **Стохастичний градієнтний спуск (SGD):** Оновлення ваг проводиться для кожного навчального прикладу окремо.
2. **Mini-batch градієнтний спуск:** Оновлення ваг проводиться на невеликих випадкових підмножинах даних (батчах).

Ці підходи є більш ефективними за обчисленнями порівняно з пакетним градієнтним спуском для великих наборів даних і часто ведуть до кращої збіжності.

2. Покрокова інструкція

1) **Підготовка даних та аналіз**
•Завантажте дані за допомогою pandas.
•Створіть цільову змінну Test_Result_Normal (бінарна: 1 якщо Test Results = "Normal", 0 інакше).
•Видаліть колонки, що не несуть прогностичної цінності (наприклад, Name, Date of Admission, Doctor, Hospital, Room Number, Discharge Date).
•Перетворіть категоріальні ознаки (Gender, Blood Type, Medical Condition, Insurance Provider, Admission Type, Medication) за допомогою one-hot encoding (можна використати pd.get_dummies).
•Нормалізуйте числові ознаки (Age, Billing Amount).
•Проведіть аналіз даних:

➔Розподіл цільової змінної.

➔Кореляційна матриця числових ознак.

➔Гістограми для візуалізації розподілів.

2) **Розподіл даних**

•Розділіть дані на тренувальний (60%), валідаційний (20%) та тестовий (20%) набори з використанням train_test_split з random_state=42.

3) **Реалізація логістичної регресії**
Реалізуйте функції:
• Сигмоїда: $\sigma(z^{(i)}) = \dfrac{1}{1+e^{-z^{(i)}}}$
• $z^i = \theta_0 + \theta_1 x_1^{(i)} + \theta_2 x_2^{(i)} + ... + \theta_n x_{(n)}^{(i)}$

• Функція втрат (Log Loss):

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log(h_\theta(x^{(i)})) + (1-y^{(i)})\log(1-h_\theta(x^{(i)}))\right]$$

де $J(\theta)$ — функція втрат, m — кількість прикладів у наборі даних, $y^{(i)}$ — мітка у наборі даних, $h_\theta(x^{(i)}) = \sigma(z^{(i)})$ — прогнозована ймовірність (вихід сигмоїди) для і-того зразка, log() — натуральний логарифм.

• Для багатокласової задачі функція втрат задається як:

$$z_k^{(i)} = \mathbf{x}^{(i)}\boldsymbol{\theta_k}, \quad k = 1, \ldots, K$$

$$h_\theta(x^{(i)})_k = \frac{e^{z_k^{(i)}}}{\sum_{j=1}^{K} e^{z_j^{(i)}}}$$

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)}\ \log\left(h_\theta(x^{(i)})_k\right)$$

• Градієнтний спуск:

➔**Стохастичний (SGD)**: Оновлення ваг для кожного прикладу.
➔**Mini-batch**: Оновлення ваг для батчів розміром batch_size (наприклад, 32).

•Векторизуйте обчислення за допомогою NumPy.
•Додайте підтримку лінійного члену (intercept).

3.1. Додатково, але бажано (*): валідація та криві навчання

Під час навчання моделі необхідно моніторити процес для запобігання перенавчання (overfitting) та вибору найкращих параметрів.

•Після **кожної епохи** навчання:

1.Розрахуйте значення функції втрат **на тренувальному наборі**.
2.**Використайте валідаційний набір** для розрахунку функції втрат на даних, які не використовуються для оновлення ваг.
3.Запишіть обидва значення для побудови кривих навчання.
•Побудуйте графік (**learning curve**), де:

1.Вісь X: кількість епох
2.Вісь Y: значення функції втрат
3.Дві лінії: loss на тренувальному наборі та loss на валідаційному наборі
•Проаналізуйте криві:

1.**Якщо обидві криві зменшуються і виходять на плато:** навчання пройшло успішно.

2.**Якщо крива валідації починає зростати,** а тренувальна продовжує зменшуватися: це ознака **перенавчання**. Модель "запам'ятовує" тренувальні дані і втрачає здатність до узагальнення.
3.**Якщо обидві криві залишаються високими:** модель **недонавчена** (можливо, потрібно більше епох, змінити швидкість навчання або ускладнити модель).
•На основі аналізу кривих навчання оберіть найкращу версію моделі (наприклад, з епохи, де валідаційний loss був мінімальним) для фінального тестування на тестовому наборі.

```
# Ініціалізація ваг
weights = initialize_weights(n_features)

best_val_loss = float('inf')
best_epoch = 0
best_weights = None
patience = 10
epochs_no_improve = 0

train_losses = []
val_losses = []

for epoch in range(max_epochs):
    # Навчання на тренувальному наборі (SGD або mini-batch)Logistic Regression
    for batch in get_batches(X_train, y_train, batch_size):
        gradients = compute_gradients(batch, weights)
        weights = update_weights(weights, gradients, learning_rate)

    # Розрахунок втрат після епохи
    train_loss = compute_loss(X_train, y_train, weights)
    val_loss = compute_loss(X_val, y_val, weights)  # <-- Використання валідаційного сету

    train_losses.append(train_loss)
    val_losses.append(val_loss)

    # Визначення найкращої моделі
    if val_loss < best_val_loss:
        best_val_loss = val_loss
        best_epoch = epoch
        best_weights = weights.copy()
        epochs_no_improve = 0
    else:
```

```
        epochs_no_improve += 1

    # Рання зупинка
    if epochs_no_improve >= patience:
        print(f"Early stopping at epoch {epoch}")
        break

# Відновлення найкращих ваг
weights = best_weights
```

4. Навчання та оцінка

•Навчіть модель з використанням:

а) Стохастичного градієнтного спуску.
б) Mini-batch градієнтного спуску.

•Побудуйте криві навчання (значення функції втрат для тренувального та валідаційного наборів по епохах).
•Порівняйте швидкість збіжності та стабільність обох методів.

5. Додатково (*): Регуляризація

•Реалізуйте L1 (Lasso) та L2 (Ridge) регуляризацію

•У **логістичній регресії (і взагалі в лінійних моделях)** при регуляризації (L2 або L1) **параметр зсуву** (інтерсепт, bias, тобто θ_0) **не регуляризується**

•Додайте члени регуляризації до функції втрат (формули записані для batch GD, для SGD $m = 1$, mini-batch $m = B$, де B - розмір пакета)

•L2: $J(\theta) + \frac{\lambda}{2m} \sum_{j=1}^{n} \theta_j^2$

•L1: $J(\theta) + \frac{\lambda}{m} \sum_{j=1}^{n} |\theta_j|$

•Адаптуйте градієнт відповідно до обраного типу регуляризації.

6. Оцінка якості моделі

•Обчисліть прогнози на тестовому наборі.
•Побудуйте confusion matrix.
•Розрахуйте метрики: Accuracy, Precision, Recall, F1-score.
•Поясніть, які класи помилково класифікуються найчастіше та чому.

Jupyter Notebook файл має містити:

Коментарі до коду та алгоритму варто робити у комірці типу markdown.

1.Код реалізації моделі та підготовки даних.
2.Графіки: кореляційна матриця, криві навчання для SGD та mini-batch.
3.Порівняльну таблицю метрик для різних методів оптимізації.
4.Висновки щодо ефективності обраних підходів.

Примітка: *Використання готових реалізацій логістичної регресії (наприклад, з scikit-learn) заборонено, крім допоміжних функцій (наприклад, для розділення даних або кодування ознак).*

Критерії оцінювання:
•Коректність реалізації алгоритмів.
•Глибина аналізу даних.
•Якість візуалізацій та інтерпретація результатів.
•Чистота коду та коментарі.

Допоміжні лінки:
1. Logistic Regression
2. YouTube