Collaborative Filtering implementation

Mean Normalization

- Khi một người dùng mới xuất hiện và người này chưa đánh giá item nào, hệ thống sẽ dự đoán tất cả đánh giá của người này là 0 → không thực tế

⇒ Sử dụng Mean Normalization

Bước 1: Tính trung bình các đánh giá của từng item

Bước 2: Chuẩn hóa dữ liệu đầu vào

- Với mỗi giá trị có sẵn i

$$y'_{i,j} = y_{i,j} - \mu_i$$

- Thay vì học trực tiếp trên y, ta huấn luyện trên **giá trị đã chuẩn hóa y'**.

Bước 3: Dự đoán

- Dự đoán chuẩn hóa:

$$\hat{y}'_{i,j} = w^{(j)} \cdot x^{(i)} + b^{(j)}$$

- Dự đoán thực tế:

$$\hat{y}_{i,j} = \hat{y}'_{i,j} + \mu_i$$

Mục đích:

- Mean normalization giúp **hàm mất mát (cost function)** dễ tối ưu hơn.
- **Tăng tốc quá trình hội tụ** khi sử dụng thuật toán gradient descent.

Cài đặt bằng Tensorflow

Với Gradient Descent cơ bản

```
# Bước 1: Khai báo biến
w = tf.Variable(3.0)  # Tham số cần tối ưu
x = 1.0
y = 1.0
alpha = 0.01  # learning rate

# Bước 2: Lặp lại với Gradient Tape
for iter in range(30):
    with tf.GradientTape() as tape:
        f_wx = w * x
        cost = (f_wx - y) ** 2

    # Bước 3: Tính đạo hàm
    dJ_dw = tape.gradient(cost, w)

    # Bước 4: Cập nhật tham số
    w.assign_sub(alpha * dJ_dw)
```

Với Adam

```
optimizer = tf.keras.optimizers.Adam(learning_rate=0.01)

for iter in range(200):
    with tf.GradientTape() as tape:
        J = compute_cost(x, w, b, Ynorm, R, num_users, num_movies, lambda
_)

    grads = tape.gradient(J, [x, w, b])
    optimizer.apply_gradients(zip(grads, [x, w, b]))
```

Tìm kiếm item tương tự

Thuật toán collaborative filtering giúp bạn học được:

- Với mỗi **mặt hàng** i, bạn có một **vector đặc trưng** $x^{(i)}$.
- Mặc dù không thể diễn giải trực tiếp từng thành phần trong vector này, nhưng **tổng thể** vector này phản ánh nội dung của mặt hàng đó.

Nếu bạn muốn tìm các mặt hàng giống mặt hàng i, hãy tìm những mặt hàng k có vector đặc trưng $x^{(k)}$ gần với $x^{(i)}$.

Cụ thể:

Độ giống nhau = $\sum_{l=1}^{n} \left(x_l^{(k)} - x_l^{(i)} \right)^2$

→ Đây là **khoảng cách Euclid bình phương** giữa 2 vector.

Càng gần nhau, thì 2 mặt hàng càng “giống” nhau.