

Users who have not rated any movies

Movie	Alice(1)	Bob(2)	Carol(3)	Dave(4)	Eve(5)
Love at last	5	5	0	0	?
Romance forever	5	?	?	0	?
Cute puppies of love	?	4	0	?	?
Nonstop car chases	0	0	5	4	?
Swords vs. karate	0	0	5	?	?

For user j , on movie i predict:

$$w^{(j)} \cdot x^{(i)} + b^{(j)} + \mu_i$$

User 5 (Eve):

$$w^{(5)} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad b^{(5)} = 0 \quad \underbrace{w^{(5)} \cdot x^{(i)} + b^{(5)}}_0 + \mu_1 = 2.5$$

Custom Training Loop

Gradient descent algorithm
Repeat until convergence

$J = (w \cdot x - 1)^2$

Fix $b = 0$ for this example

$w = w - \alpha \frac{\partial}{\partial w} J(w, b)$

$\frac{\partial}{\partial w} J(w)$

tf.variables are the parameters we want to optimize

Auto Diff
Auto Grad

iterations = 30

for iter in range(iterations):

Use TensorFlow's GradientTape to record the steps

used to compute the cost J , to enable auto differentiation.

with tf.GradientTape() as tape:

$fwb = w \cdot x$

$cost_j = (fwb - y)^2$

Use the gradient tape to calculate the gradients

of the cost with respect to the parameter w .

$[dJdw] = tape.gradient(cost_j, [w])$

Run one step of gradient descent by updating

the value of w to reduce the cost.

$w.assign_add(-alpha * dJdw)$

tf.variables require special function to modify

DeepLearning.AI Stanford ONLINE Andrew Ng

Implementation in TensorFlow

Gradient descent algorithm

Repeat until convergence

$w = w - \alpha \frac{\partial}{\partial w} J(w, b, x)$

$b = b - \alpha \frac{\partial}{\partial b} J(w, b, x)$

$X = X - \alpha \frac{\partial}{\partial X} J(w, b, x)$

Instantiate an optimizer.

`optimizer = keras.optimizers.Adam(learning_rate=1e-1)`

iterations = 200

for iter in range(iterations):

Use TensorFlow's GradientTape

to record the operations used to compute the cost

with tf.GradientTape() as tape:

Compute the cost (forward pass is included in cost)

`cost_value = cofilCostFunc(X, w, b, Ynorm, R, num_users, num_movies, lambda)`

Use the gradient tape to automatically retrieve

the gradients of the trainable variables with respect to the loss

`grads = tape.gradient(cost_value, [X, w, b])`

Run one step of gradient descent by updating

the value of the variables to minimize the loss.

`optimizer.apply_gradients(zip(grads, [X, w, b]))`

Dataset credit: Harper and Konstan. 2015. The MovieLens Datasets: History and Context

10:05

Finding related items

The features $x^{(i)}$ of item i are quite hard to interpret.

To find other items related to it, find item k with $x^{(k)}$ similar to $x^{(i)}$

i.e. with smallest distance

$$\sum_{l=1}^n (x_l^{(k)} - x_l^{(i)})^2$$

$$\|x^{(k)} - x^{(i)}\|^2$$

romance
action
 x_1, x_2, x_3
 n

Limitations of Collaborative Filtering

→ Cold start problem. How to

- rank new items that few users have rated?
- show something reasonable to new users who have rated few items?

Use side information about items or users:

- Item: Genre, movie stars, studio, ...
- User: Demographics (age, gender, location), expressed preferences, ...