Predicting movie ratings

- users — n_u
- items — n_m
- ratings — $r(i,j) = 1$ if j rates i

Predicting movie ratings

User rates movies using ~~one~~ zero to five stars

Movie	Alice(1)	Bob(2)	Carol(3)	Dave(4)
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	?

n_u = no. of users

n_m = no. of movies

$r(i,j)=1$ if user j has rated movie i

$y^{(i,j)}$ = rating given by user j to movie i (defined only if $r(i,j)=1$)

$n_u = 4$

$n_m = 5$

$r(1,1) = 1$ → rated movie 1

$r(3,1) = 0$ — not rated

+ features

	romance	action
$x^{(1)}$	0.9	0
	1.0	0.01
$x^{(3)}$	0.99	0
	0.1	1.0
	0	0.9

$$x^{(1)} = \begin{bmatrix} 0.9 \\ 0 \end{bmatrix} \qquad x^{(3)} = \begin{bmatrix} 0.99 \\ 0 \end{bmatrix}$$

For User 1 — Alice — movie i

$w^{(1)} x^{(i)} + b^{(1)}$ ← linear regression

3rd movie

$$w^{(1)} = \begin{bmatrix} 5 \\ 0 \end{bmatrix} \quad b^{(1)} = 0 \quad x^{(3)} = \begin{bmatrix} 0.99 \\ 0 \end{bmatrix}$$

$w^{(1)} \cdot x^{(3)} + b^{(1)} = 4.95$

for user j; predict user j's rating for movie i as

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

Cost Function

$$\min \rightarrow \frac{1}{2m^{(j)}} \sum_{i:r(i,j)=1} \left(w^{(j)} \cdot x^{(i)} + b^{(j)} - y^{(i,j)}\right)^2$$

parameters

movies user j has rated

$$+ \frac{\lambda}{2m^{(j)}} \sum_{k=1}^{n} \left(w_k^{(j)}\right)^2$$

for recommender systems

regularization

Cost function

To learn parameters $w^{(j)}, b^{(j)}$ for user j :

$$J(w^{(j)}, b^{(j)}) = \frac{1}{2} \sum_{i:r(i,j)=1} \left(w^{(j)} \cdot x^{(i)} + b^{(j)} - y^{(i,j)}\right)^2 + \frac{\lambda}{2} \sum_{k=1}^{n} \left(w_k^{(j)}\right)^2$$

To learn parameters $w^{(1)}, b^{(1)}, w^{(2)}, b^{(2)}, \ldots w^{(n_u)}, b^{(n_u)}$ for all users :

$$J\begin{pmatrix} w^{(1)}, \ldots, w^{(n_u)} \\ b^{(1)}, \ldots, b^{(n_u)} \end{pmatrix} = \frac{1}{2} \sum_{j=1}^{n_u} \sum_{i:r(i,j)=1} \underbrace{\left(w^{(j)} \cdot x^{(i)} + b^{(j)} - y^{(i,j)}\right)^2}_{f(x)} + \frac{\lambda}{2} \sum_{j=1}^{n_u} \sum_{k=1}^{n} \left(w_k^{(j)}\right)^2$$

Collaborative Filtering :-

Problem motivation

Movie	Alice (1)	Bob (2)	Carol (3)	Dave (4)	x_1 (romance)	x_2 (action)
Love at last	5	5	0	0	?	?
Romance forever	5	?	?	0 ←	?	? $x^{(2)}$
Cute puppies of love	?	4	0	? ←	?	? $x^{(3)}$
Nonstop car chases	0	0	5	4 ←	?	?
Swords vs. karate	0	0	5	? ←	?	?

$w^{(1)} = \begin{bmatrix} 5 \\ 0 \end{bmatrix}$, $w^{(2)} = \begin{bmatrix} 5 \\ 0 \end{bmatrix}$, $w^{(3)} = \begin{bmatrix} 0 \\ 5 \end{bmatrix}$, $w^{(4)} = \begin{bmatrix} 0 \\ 5 \end{bmatrix}$

$b^{(1)} = 0$, $b^{(2)} = 0$, $b^{(3)} = 0$, $b^{(4)} = 0$ ←

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

$w^{(1)} \cdot x^{(1)} \approx 5$
$w^{(2)} \cdot x^{(1)} \approx 5$
$w^{(3)} \cdot x^{(1)} \approx 0$
$w^{(4)} \cdot x^{(1)} \approx 0$

→ $x^{(1)} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ ↑ guess

$$J(x^{(i)}) = \frac{1}{2} \sum_{j : r(i,j)=1} \left(w^{(j)} \cdot x^{(i)} + b^{(j)} - y^{(i,j)} \right)^2 + \frac{\lambda}{2} \sum_{k=1}^{n} \left(x_k^{(i)} \right)^2$$

To Learn $x^{(1)}, x^{(2)}, \ldots x^{(nm)}$ features;

$$J = \frac{1}{2} \sum_{i=1}^{n_m} \sum_{j : r(i,j)=1} \left(w^{(j)} \cdot x^{(i)} + b^{(j)} - y^{(i,j)} \right)^2 + \frac{\lambda}{2} \sum_{i=1}^{n_m} \sum_{k=1}^{n} \left(x_k^{(i)} \right)^2$$

Full Cost Function;

(+)

Minimize → Gradient Descent;

Linear regression but

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

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

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

Binary Labels

Binary labels

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

Did user like item? 1 or 0

spend at least 30s? 1 or 0

clicked?

For binary;

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

$$g(z) = \frac{1}{1 + e^{-z}}$$

Cost function for binary application

Loss for binary labels $= y^{(i,j)} \stackrel{?}{=} f_{(w,b,x)}^{(x)} = g(w^{(j)} \cdot x^{(i)} + b^{(j)})$

Cost function for binary application

Previous cost function:

$$\frac{1}{2}\sum_{(i,j):r(i,j)=1}\underbrace{(w^{(j)}\cdot x^{(i)}+b^{(j)}}_{f(x)}-y^{(i,j)})^2+\frac{\lambda}{2}\sum_{i=1}^{n_m}\sum_{k=1}^{n}(x_k^{(i)})^2+\frac{\lambda}{2}\sum_{j=1}^{n_u}\sum_{k=1}^{n}(w_k^{(j)})^2$$

Loss for binary labels $y^{(i,j)}$: $f_{(w,b,x)}(x) = g(w^{(j)} \cdot x^{(i)} + b^{(j)})$

$$L\left(f_{(w,b,x)}(x), y^{(i,j)}\right) = -y^{(i,j)}\log\left(f_{(w,b,x)}(x)\right) - (1-y^{(i,j)})\log\left(1-f_{(w,b,x)}(x)\right)$$

Loss for single example

Cost function: $J(w,b,x) = \sum_{(i,j):r(i,j)=1} L(f_{(w,b,x)}(x), y^{(i,j)})$

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

Mean Normalization

For new user not rated;

$$\begin{bmatrix} 5 & 5 & 0 & 0 & ? \\ 5 & ? & ? & 0 & ? \\ ? & 4 & 0 & ? & ? \\ 0 & 0 & 5 & 4 & ? \\ 0 & 0 & 5 & 0 & ? \end{bmatrix} \quad \mu = \begin{bmatrix} 2.5 \\ 2.5 \\ 2 \\ 2.25 \\ 1.25 \end{bmatrix} \leftarrow \text{avg.}$$

↓ subtract

Normalize Matrix rows (can also do columns)

$$\begin{bmatrix} 2.5 & 2.5 & -2.5 & -2.5 & ? \\ 2.5 & ? & ? & -2.5 & ? \\ -2.25 & -2.25 & 2.75 & 1.75 & ? \\ -1.25 & -1.25 & 3.75 & -1.25 & ? \end{bmatrix}$$

For user j on movie i predict;

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

now if $w^{(5)} \begin{bmatrix} 0 \\ 0 \end{bmatrix} + b^{(5)} = 0$

no matter $b^{(5)}$ $\rightarrow 0 + \mu_1$

$= 2.5$

Collaborative Filtering Algorithm in Tensorflow

→ (No dense layer NN etc) → Without fit

Custom Training Loop

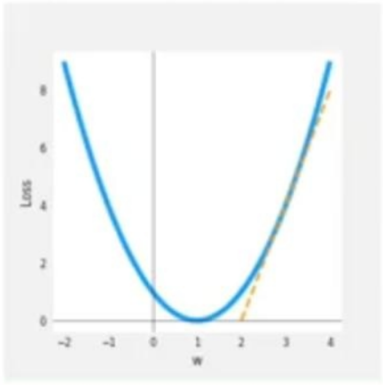
$J = (wx - 1)^2$ ($f(x)$, y)

Gradient descent algorithm

Repeat until convergence

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

Fix b = 0 for this example




```
w = tf.Variable(3.0)
x = 1.0
y = 1.0 # target value
alpha = 0.01

iterations = 30
for iter in range(iterations):
    # Use TensorFlow's Gradient tape to record the steps
    # used to compute the cost J, to enable auto differentiation.
    with tf.GradientTape() as tape:
        fwb = w*x
        costJ = (fwb - y)**2

    # Use the gradient tape to calculate the gradients
    # of the cost with respect to the parameter w.
    [dJdw] = tape.gradient( costJ, [w] )

    # Run one step of gradient descent by updating
    # the value of w to reduce the cost.
    w.assign_add(-alpha * dJdw)
```

Tf.variables are the parameters we want to optimize

Auto Diff software pckg → AutoGrad

tf.variables require special function to modify

Implementation in TensorFlow

Adam optimization (not grd. desc)

```
# 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 = cofiCostFuncV(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]) )
```

$J(w,b,x)$

$w - \alpha \frac{\partial}{\partial w} J$

$b = b - \alpha \frac{\partial}{\partial b} J$

$x = x - \alpha \frac{\partial}{\partial x} J$

normalized rating

derivative of cost func

Adam with gradients

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

Find related items;

Find item k with $x^{(k)}$ → similar to $x^{(i)}$

$$\sum_{l=1}^{n} \left(x_l^{(k)} - x_l^{(i)}\right)^2 \Rightarrow \left\| x^{(k)} - x^{(i)} \right\|^2$$

closest vector dist.

Limitations of Collaborative filtering;

- Cold-Start Problem → new item few users rated
  or new user few rated
- Use side information

two vectors → User vector

w → eg:- [action high, romance low, ...]

all users → [0.9, 0.2, ...]

Collaborate Filtering

all items → Item vector

x → eg:- [0.8, 0.1, ...]

$wx + b$

Collaborative Filtering

Reccomend based on ratings of users giving similar ratings

Content Based Filtering

recommend based on features of user & item

eg:- $x_u^{(j)}$ for user j — age/gender/.... Movies watched

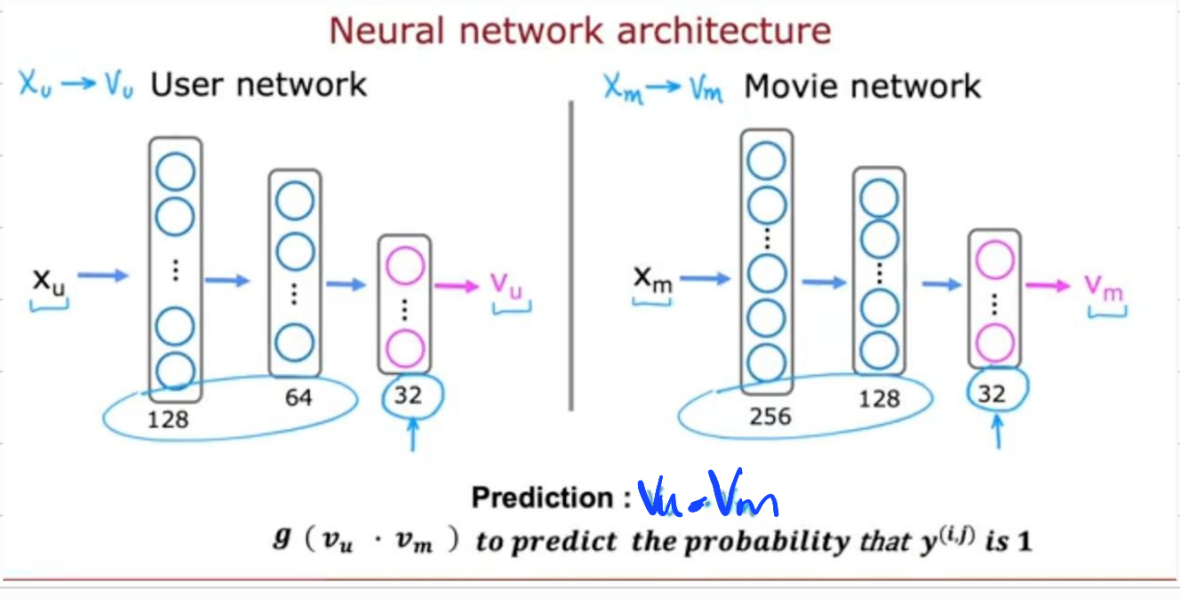
$x_m^{(i)}$ for movie i — year/reviews/genre /avg. rating - ....

feature vectors diff. in size

Content Based; match movie i with user j

$$W^{(j)} \cdot X^{(i)} + b$$

from $x_u^{(j)}$ → $V_u^{(j)}$ $V_m^{(i)}$ ← from $X_m^{(i)}$



Cost Function;

$$J = \sum_{(i,j):r(i,j)=1} \left(Vu^{(i)} \cdot Vm^{(i)} - y^{(i,j)}\right)^2 + NN \text{ regularization}$$

similar movie;

$$\| Vm^{(k)} - Vm^{(i)} \|^2 \longrightarrow \text{small}$$

* pre compute ahead of time

* feature selection · computational req.

For large sets

Retrieval + Ranking

find 10 most similar

3 genres most viewed

top 20 of country

combine/remove duplicate /already purch. ⟶

more results performance ↑

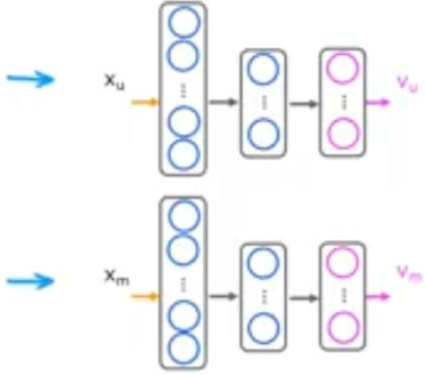
Rank → fine tuned model

Ethical concerns;

⟶ Advertising ? Transparency ?

Maximize profit ?

Amelioration ⟶ Do not accept ads from exploitative businesses




```
user_NN = tf.keras.models.Sequential([
  tf.keras.layers.Dense(256, activation='relu'),
  tf.keras.layers.Dense(128, activation='relu'),
  tf.keras.layers.Dense(32)
])

item_NN = tf.keras.models.Sequential([
  tf.keras.layers.Dense(256, activation='relu'),
  tf.keras.layers.Dense(128, activation='relu'),
  tf.keras.layers.Dense(32 )
])
```

```
# create the user input and point to the base network
input_user = tf.keras.layers.Input(shape=(num_user_features))
vu = user_NN(input_user)
vu = tf.linalg.l2_normalize(vu, axis=1)

# create the item input and point to the base network
input_item = tf.keras.layers.Input(shape=(num_item_features))
vm = item_NN(input_item)
vm = tf.linalg.l2_normalize(vm, axis=1)

# measure the similarity of the two vector outputs
output = tf.keras.layers.Dot(axes=1)([vu, vm])

# specify the inputs and output of the model
model = Model([input_user, input_item], output)

# Specify the cost function
cost_fn = tf.keras.losses.MeanSquaredError()
```

→ Vu → ⊙ →
Vm ↗

cost func

Feature vectors can be any size as long as $V_u = V_m$