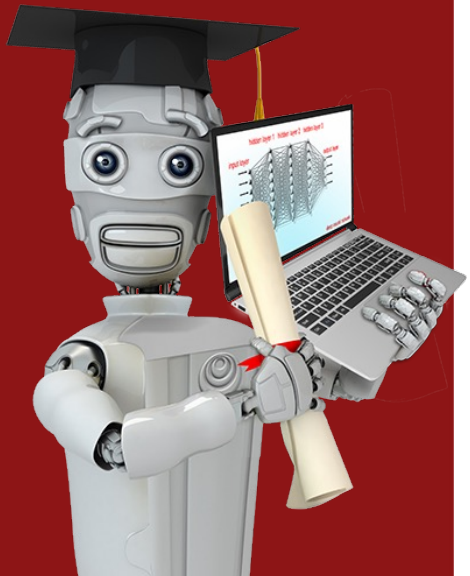
Copyright Notice

DeepLearning.AI

Stanford ONLINE



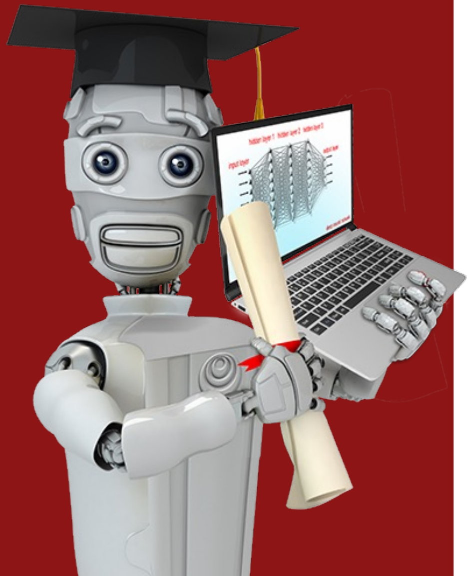
Recommender Systems

DeepLearning.AI

Stanford ONLINE

Recommender System

Making recommendations



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	?

Ratings				
★				
★	★			
★	★	★		
★	★	★	★	
★	★	★	★	★

n_u = no. of users

n_m = no. of movies

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

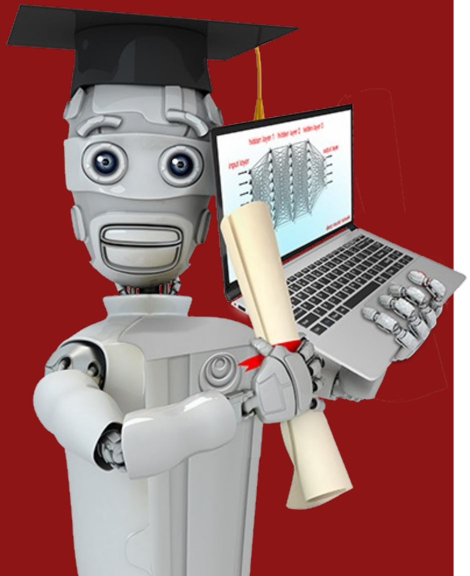
$y^{(i,j)}$ = rating given by user j to movie i (defined only if $r(i,j)=1$)

$$n_u = 4 \qquad r(1,1) = 1$$

$$n_m = 5 \qquad r(3,1) = 0 \qquad y^{(3,2)} = 4$$

DeepLearning.AI

Stanford
ONLINE



Collaborative Filtering

Using per-item features

What if we have features of the movies?

$n_u = 4$
$n_m = 5$
$n = 2$

Movie	Alice(1)	Bob(2)	Carol(3)	Dave(4)	x_1 (romance)	x_2 (action)
Love at last	5	5	0	0	0.9	0
Romance forever	5	?	?	0	1.0	0.01
Cute puppies of love	?	4	0	?	0.99	0
Nonstop car chases	0	0	5	4	0.1	1.0
Swords vs. karate	0	0	5	?	0	0.9

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

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

For user 1: Predict rating for movie *i* as: $w^{(1)} \cdot x^{(i)} + b^{(1)}$ ← just linear regression

$w^{(1)} = \begin{bmatrix} 5 \\ 0 \end{bmatrix}$ $b^{(1)} = 0$ $x^{(3)} = \begin{bmatrix} 0.99 \\ 0 \end{bmatrix}$ $w^{(1)} \cdot x^{(3)} + b^{(1)} = \underline{4.95}$

For user *j*: Predict user *j*'s rating for movie *i* as $w^{(j)} \cdot x^{(i)} + b^{(j)}$

Cost function

Notation:

$r(i,j) = 1$ if user j has rated movie i (0 otherwise)
$y^{(i,j)}$ = rating given by user j on movie i (if defined)
$w^{(j)}, b^{(j)}$ = parameters for user j
$x^{(i)}$ = feature vector for movie i

For user j and movie i, predict rating: $w^{(j)} \cdot x^{(i)} + b^{(j)}$
$m^{(j)}$ = no. of movies rated by user j
To learn $w^{(j)}, b^{(j)}$

number of features

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

Cost function

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

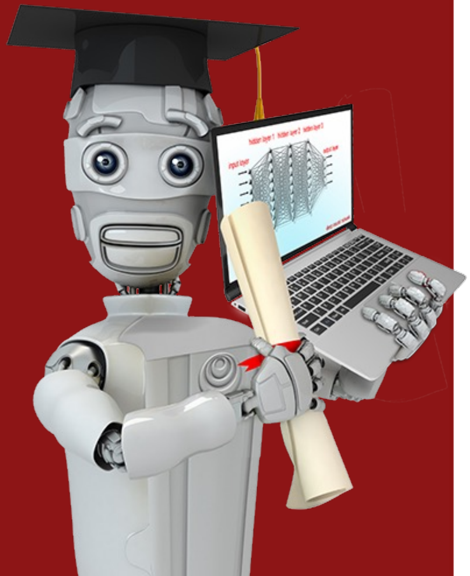
$$J(w^{(j)}, b^{(j)}) = \frac{1}{2}\sum_{i:r(i,j)} \left(w^{(j)} \cdot x^{(i)} + b^{(j)} - y^{(i,j)}\right)^2 + \frac{\lambda}{2}\sum_{k=1}^{n}\left(w_k^{(j)}\right)^2$$

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

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

DeepLearning.AI

Stanford ONLINE



Collaborative Filtering

Collaborative filtering algorithm

Problem motivation

Movie	Alice (1)	Bob (2)	Carol (3)	Dave (4)	x_1 (romance)	x_2 (action)
Love at last	5	5	0	0	0.9	0
Romance forever	5	?	?	0	1.0	0.01
Cute puppies of love	?	4	0	?	0.99	0
Nonstop car chases	0	0	5	4	0.1	1.0
Swords vs. karate	0	0	5	?	0	0.9

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

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

$\rightarrow \quad x^{(1)} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$

Cost function

Given $w^{(1)}, b^{(1)}, \; w^{(2)}, b^{(2)}, \cdots, \; w^{(n_u)}, b^{(n_u)}$

to learn $x^{(i)}$:

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

To learn $x^{(1)}, \; x^{(2)}, \cdots, \; x^{(n_m)}$:

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

Collaborative filtering

	$j=1$	$j=2$	$j=3$
	Alice	Bob	Carol
Movie1 ($i=1$)	5	5	?
Movie2 ($i=2$)	?	2	3

Cost function to learn $w^{(1)}, b^{(1)}, \cdots w^{(n_u)}, b^{(n_u)}$:

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

Cost function to learn $x^{(1)}, \cdots, x^{(n_m)}$:

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

Put them together:

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

Gradient Descent

collaborative filtering

Linear regression (course 1)

repeat {

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

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

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

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

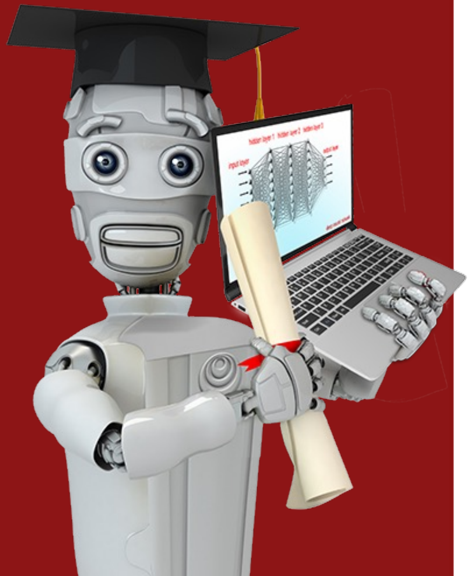
$$x_k^{(i)} = x_k^{(i)} - \alpha \frac{\partial}{\partial x_k^{(i)}} J(w,b,x)$$

}

parameters w, b, x $\quad$ x is also a parameter

DeepLearning.AI

Stanford
ONLINE



Collaborative Filtering

Binary labels:
favs,
likes and clicks

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	?

1
0
?

Example applications

1. Did user j purchase an item after being shown? 1, 0, ?
2. Did user j fav/like an item? 1, 0, ?
3. Did user j spend at least 30sec with an item? 1, 0, ?
4. Did user j click on an item? 1, 0, ?

Meaning of ratings:
1 - engaged after being shown item
0 - did not engage after being shown item
? - item not yet shown

From regression to binary classification

→ Previously:

→ Predict $y^{(i,j)}$ as $\mathbf{w}^{(j)} \cdot \mathbf{x}^{(i)} + b^{(j)}$

→ For binary labels:

Predict that the probability of $y^{(i,j)} = 1$ is given by $g(\mathbf{w}^{(j)} \cdot \mathbf{x}^{(i)} + b^{(j)})$

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

Cost function for binary application

Previous cost function:

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

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

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

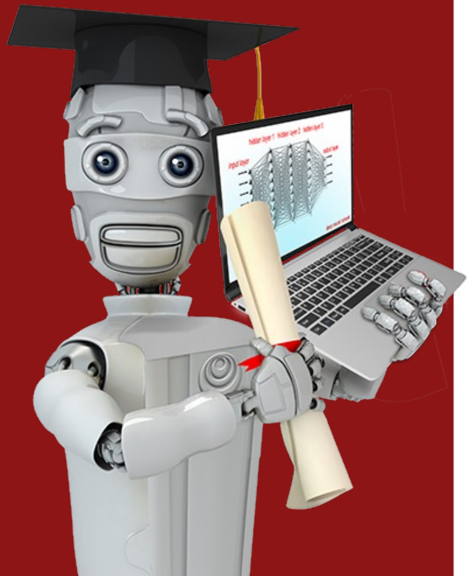
Loss for single example

$$J(w,b,x)=\sum_{(i,j):r(i,j)=1}L(\underbrace{f_{(w,b,x)}(x)}_{g(w^{(j)}\cdot x^{(i)}+b^{(j)})},y^{(i,j)})$$

cost for all examples

DeepLearning.AI

Stanford ONLINE



Recommender Systems implementation

Mean normalization

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	?	0
Romance forever	5	?	?	0	?	0
Cute puppies of love	?	4	0	?	?	0
Nonstop car chases	0	0	5	4	?	0
Swords vs. karate	0	0	5	?	?	0

$$\begin{bmatrix} 5 & 5 & 0 & 0 & ? \\ 5 & ? & ? & 0 & ? \\ ? & 4 & 0 & ? & ? \\ 0 & 0 & 5 & 4 & ? \\ 0 & 0 & 5 & 0 & ? \end{bmatrix}$$

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

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

Mean Normalization

$$\begin{array}{c|c}
\begin{bmatrix} 5 & 5 & 0 & 0 & ? \\ 5 & ? & ? & 0 & ? \\ ? & 4 & 0 & ? & ? \\ 0 & 0 & 5 & 4 & ? \\ 0 & 0 & 5 & 0 & ? \end{bmatrix} & \begin{matrix} 2.5 \\ 2.5 \\ 2 \\ 2.25 \\ 1.25 \end{matrix}
\end{array}
\qquad
\mu = \begin{bmatrix} 2.5 \\ 2.5 \\ 2 \\ 2.25 \\ 1.25 \end{bmatrix}
\qquad
\underbrace{\begin{bmatrix} 2.5 & 2.5 & -2.5 & -2.5 & ? \\ 2.5 & ? & ? & -2.5 & ? \\ ? & 2 & -2 & ? & ? \\ -2.25 & -2.25 & 2.75 & 1.75 & ? \\ -1.25 & -1.25 & 3.75 & -1.25 & ? \end{bmatrix}}_{y^{(i,j)}}$$

$$y^{(i,j)} \rightarrow w^{(j)}, b^{(j)}, x^{(i)}$$

For user j, on movie i predict:

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

User 5 (Eve):

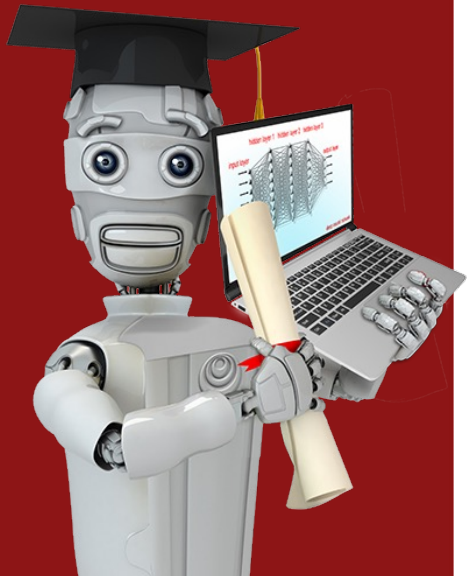
$$w^{(5)} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \qquad b^{(5)} = 0 \qquad \underbrace{w^{(5)} \cdot x^{(1)} + b^{(5)}}_{0} \quad + \mu_1 = 2.5$$

DeepLearning.AI

Stanford ONLINE

Recommender Systems implementational detail

TensorFlow implementation



Derivatives in ML

Gradient descent algorithm

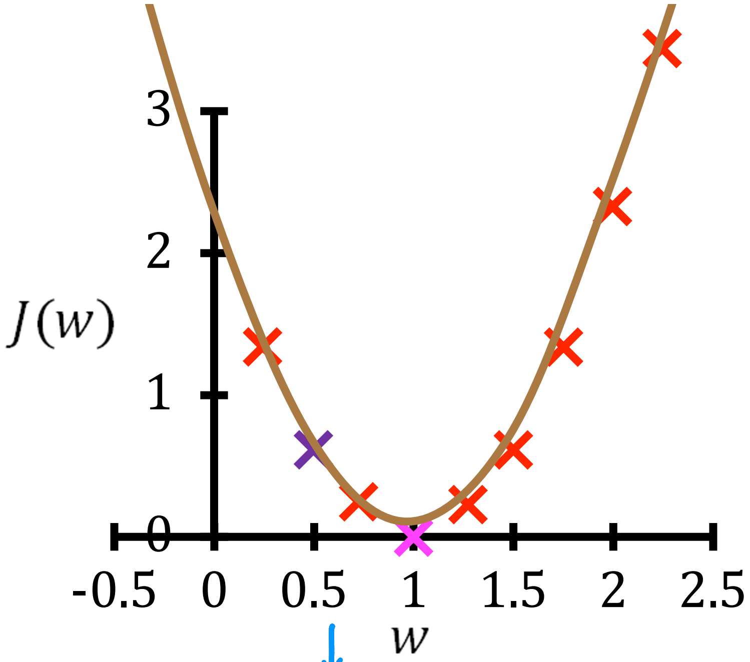
Repeat until convergence

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

$$b = b - \alpha \frac{\partial}{\partial b} J(w,b) \leftarrow \quad b = 0$$

Learning rate

Derivative



$f(x) = w \cdot x \quad b = 0$

Custom Training Loop

$$\underbrace{(wx}_{f(x)} - \underbrace{1)}_{y}$$

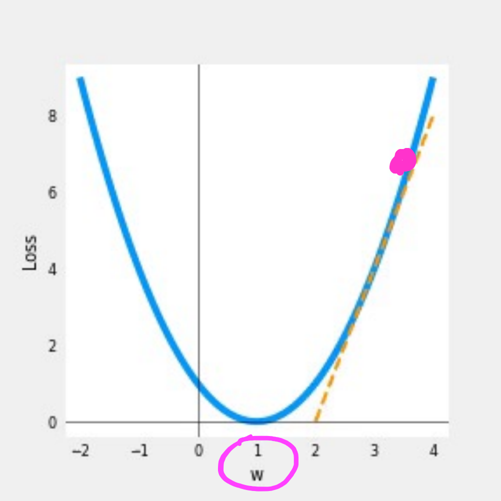
$$J = (wx - 1)^2$$

Gradient descent algorithm

Repeat until convergence

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

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)
```

f(x)

Tf.variables are the parameters we want to optimize

Auto Diff

Auto Grad

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

tf.variables require special function to modify

Implementation in TensorFlow

Gradient descent algorithm

Repeat until convergence

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

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

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

```
# 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 = cofiCostFuncV(X, W, b, Ynorm, R,
            num_users, num_movies, lambda)
```

n_u n_m

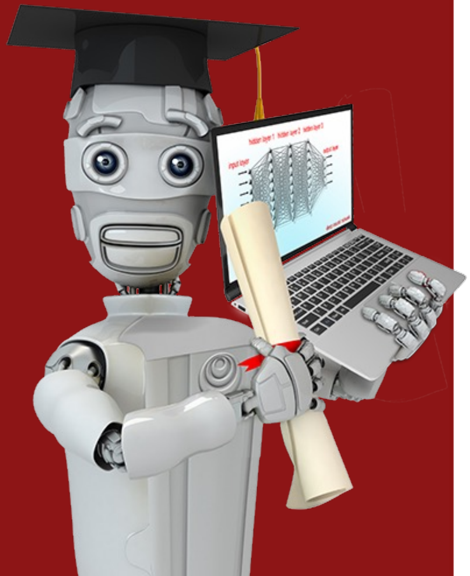
```
    # Use the gradient tape to automatically retrieve
    # the gradients of the trainable variables with respect to
        the loss
    grads = tape.gradient( cost_value, [X,W,b] )

    # 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

DeepLearning.AI

Stanford
ONLINE

Collaborative Filtering



Finding related items

Finding related items

romance

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

action

To find other items related to it,

$x_1, \quad x_2, \quad x_3$

find item k *with* $x^{(k)}$ similar to $x^{(i)}$

n

i.e. with smallest distance

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

$x^{(k)}$ $\quad$ $x^{(i)}$

$$\underbrace{\left\|\boldsymbol{x}^{(k)} - \boldsymbol{x}^{(i)}\right\|^2}$$

Limitations of Collaborative Filtering

Cold start problem. How to

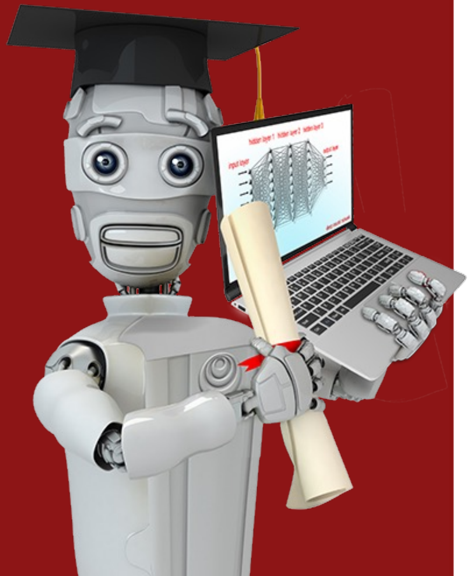
- rank new items that few users have rated?
- show something reasonable to new users who have rated few items?

Use side information about items or users:

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

DeepLearning.AI

Stanford ONLINE



Content-based Filtering

Collaborative filtering vs Content-based filtering

Collaborative filtering vs Content-based filtering

→ **Collaborative filtering:**

Recommend items to you based on rating of users who gave similar ratings as you

→ **Content-based filtering:**

Recommend items to you based on features of user and item to find good match

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

$y^{(i,j)}$ rating given by user j on item i (if defined)

Examples of user and item features

User features:

- Age
- Gender (1 hot)
- Country (1 hot, 200)
- Movies watched (1000)
- Average rating per genre
- ...

$x_u^{(j)}$ **for user j**

Vector size could be different

Movie features:

- Year
- Genre/Genres
- Reviews
- Average rating
- ...

$x_m^{(i)}$ **for movie i**

Content-based filtering: Learning to match

Predict rating of user *j* on movie *i* as

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

$v_u^{(j)} \cdot v_m^{(i)}$

computed from $x_u^{(j)}$

computed from $x_m^{(i)}$

$x_u^{(j)}$

user's preferences $v_u^{(j)} = \begin{bmatrix} 4.9 \\ 0.1 \\ \vdots \\ 3.0 \end{bmatrix}$ likes, likes

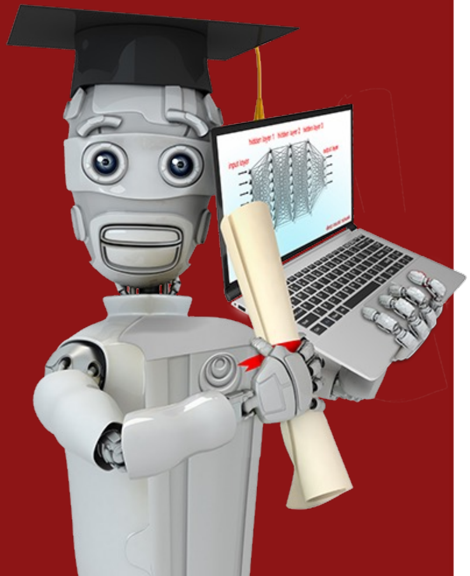
32

$x_m^{(i)}$

movie features $v_m^{(i)} = \begin{bmatrix} 4.5 \\ 0.2 \\ \vdots \\ 3.5 \end{bmatrix}$ romance, action

DeepLearning.AI

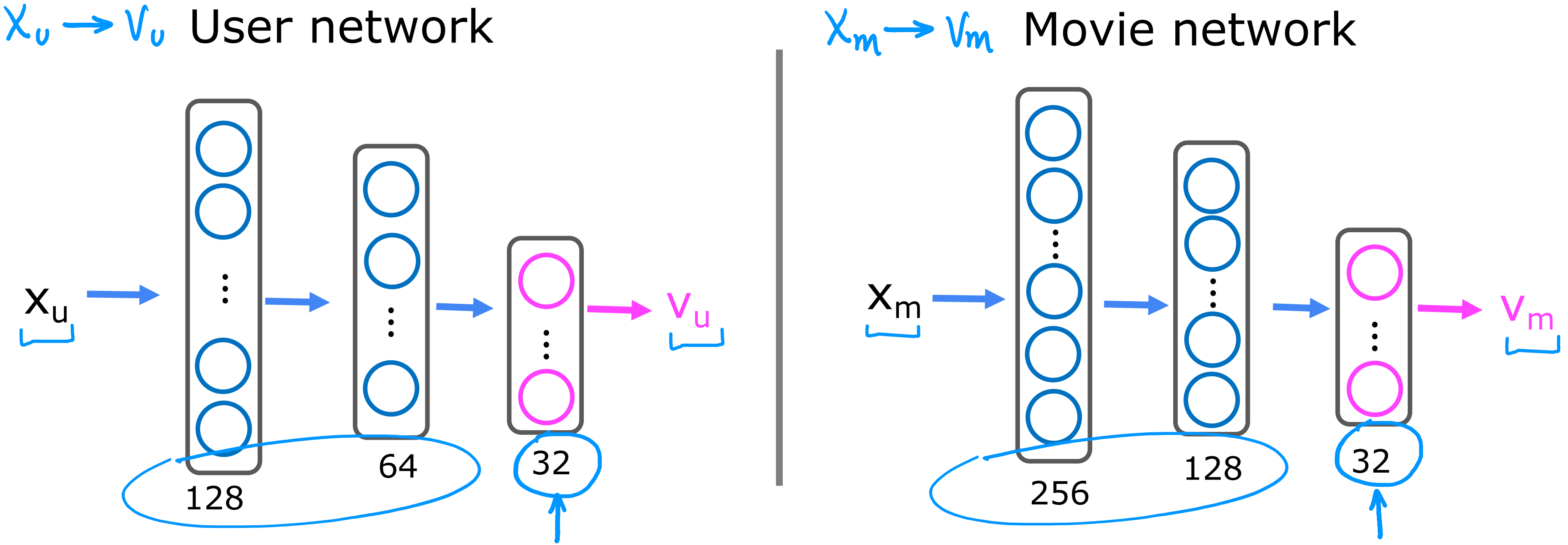
Stanford ONLINE



Content-based Filtering

Deep learning for content-based filtering

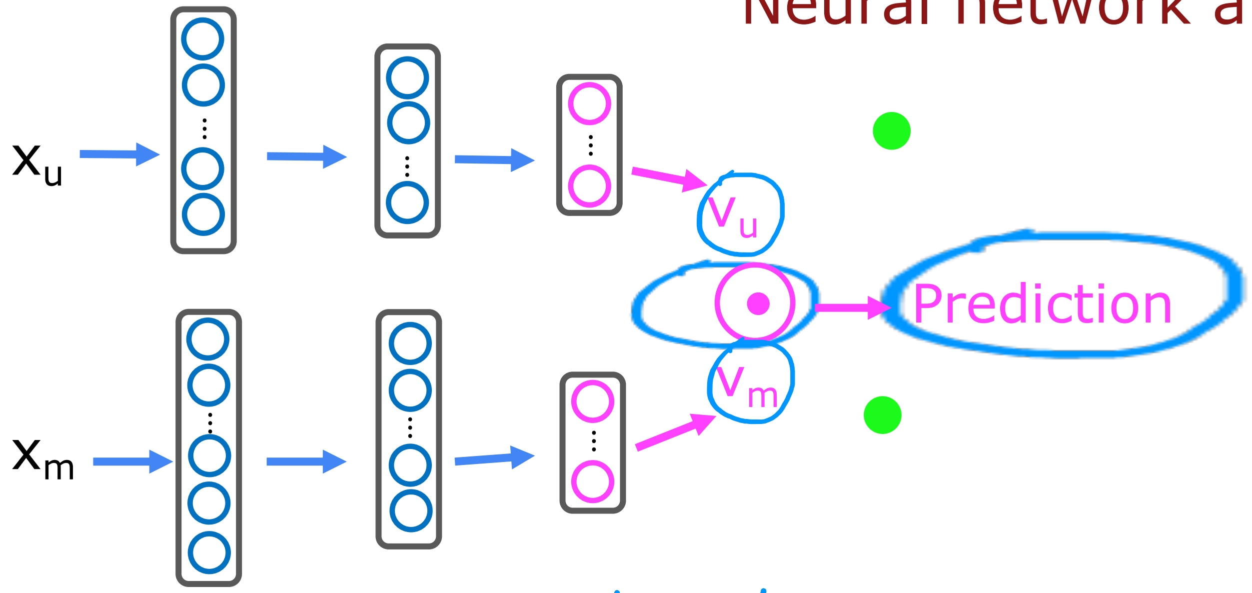
Neural network architecture



Prediction : $V_u^{(j)} \cdot V_m^{(i)}$

$g\,(v_u^{(j)} \cdot v_m^{(i)})$ *to predict the probability that* $y^{(i,j)}$ *is* 1

Neural network architecture



Cost function

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

Learned user and item vectors:

$v_u^{(j)}$ is a vector of length 32 that describes user j with features $x_u^{(j)}$

$v_m^{(i)}$ is a vector of length 32 that describes movie i with features $x_m^{(i)}$

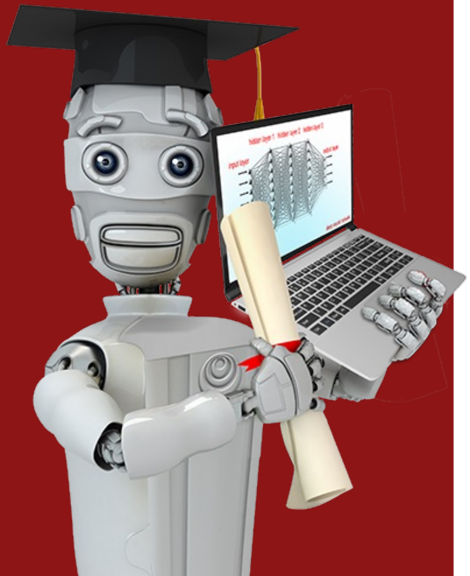
To find movies similar to movie i: $\|v_m^{(k)} - v_m^{(i)}\|^2$ small

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

Note: This can be pre-computed ahead of time

DeepLearning.AI

Stanford ONLINE

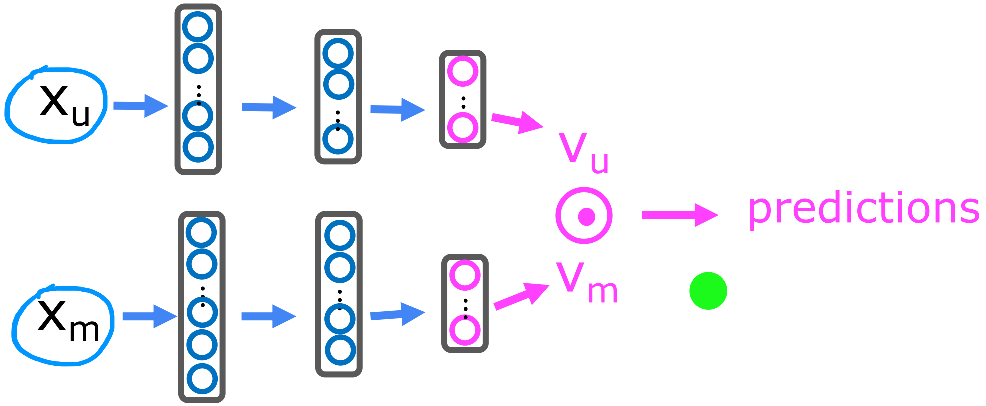


Advanced implementation

Recommending from a large catalogue

How to efficiently find recommendation from a large set of items?

- Movies 1000+
- Ads 1m+
- Songs 10m+
- Products 10m+



Two steps: Retrieval & Ranking

Retrieval:

- Generate large list of plausible item candidates ~100s
  - e.g.
  - 1) For each of the last 10 movies watched by the user, find 10 most similar movies

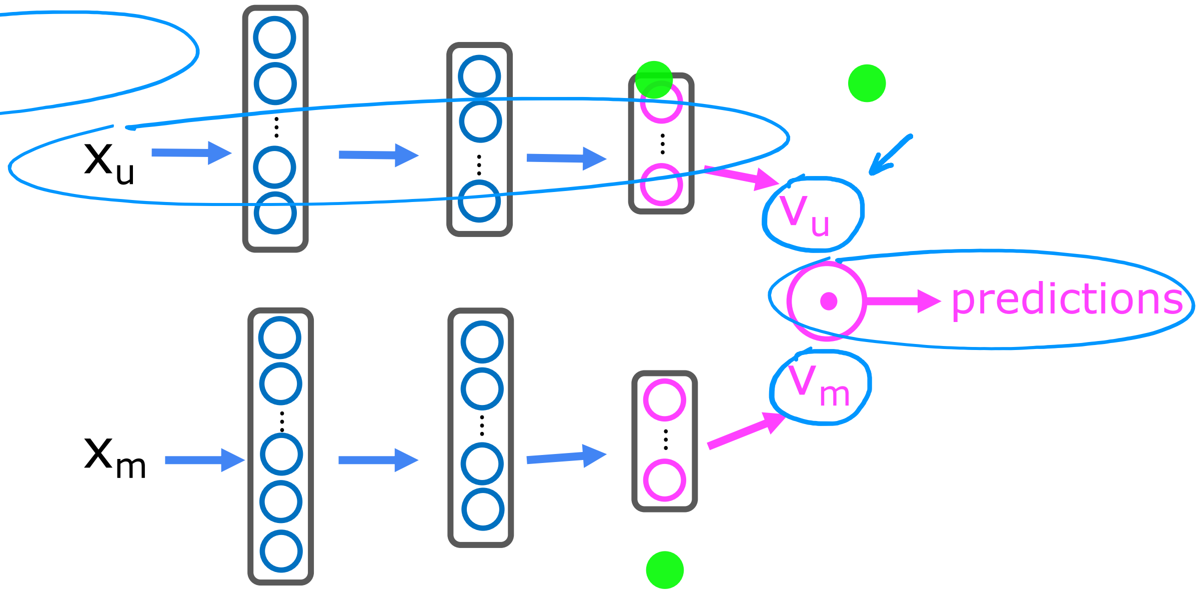
$$\| v_m^{(k)} - v_m^{(i)} \|^2$$

  - 2) For most viewed 3 genres, find the top 10 movies
  - 3) Top 20 movies in the country
- Combine retrieved items into list, removing duplicates and items already watched/purchased

Two steps: Retrieval & ranking

Ranking: ~100s

- Take list retrieved and rank using learned model



- Display ranked items to user

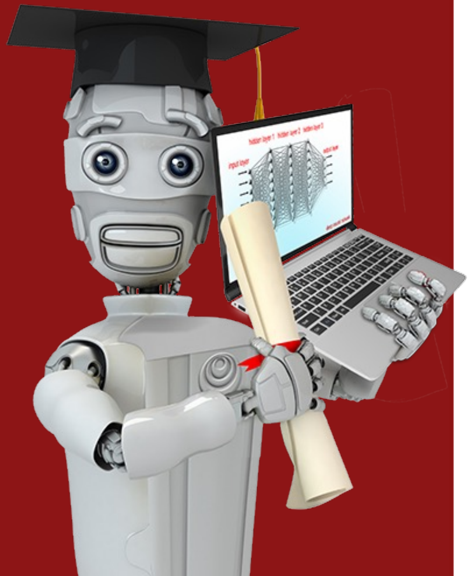
Retrieval step

- Retrieving more items results in better performance, but slower recommendations.
- To analyse/optimize the trade-off, carry out offline experiments to see if retrieving additional items results in more relevant recommendations (i.e., $p(y^{(i,j)}) = 1$ of items displayed to user are higher).

100 500

DeepLearning.AI

Stanford ONLINE



Advanced implementation

Ethical use of recommender systems

What is the goal of the recommender system?

Recommend:

- Movies most likely to be rated 5 stars by user
- Products most likely to be purchased
- Ads most likely to be clicked on + high bid
- Products generating the largest profit
- Video leading to maximum watch time

Ethical considerations with recommender systems

Travel industry

Good travel experience to more users

Bid higher for ads

More profitable

Payday loans

Squeeze customers more

Bid higher for ads

More profit

Amelioration: Do not accept ads from exploitative businesses

Other problematic cases:

- Maximizing user engagement (e.g. watch time) has led to large social media/video sharing sites to amplify conspiracy theories and hate/toxicity

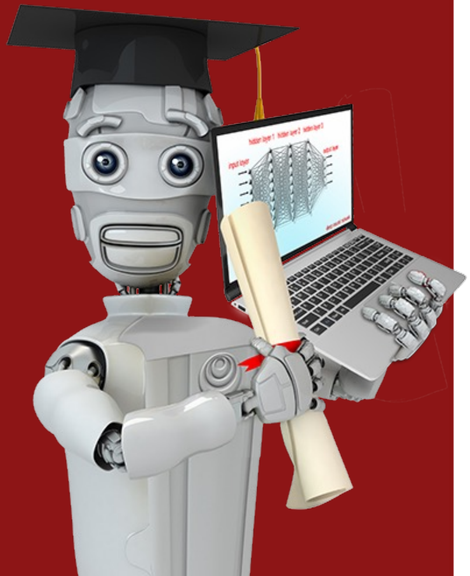
Amelioration : Filter out problematic content such as hate speech, fraud, scams and violent content

- Can a ranking system maximize your profit rather than users' welfare be presented in a transparent way?

Amelioration : Be transparent with users

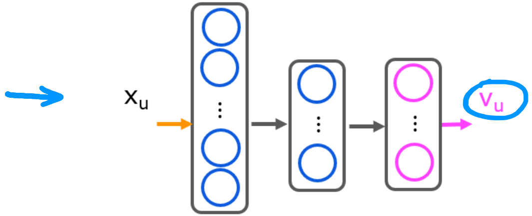
DeepLearning.AI

Stanford ONLINE

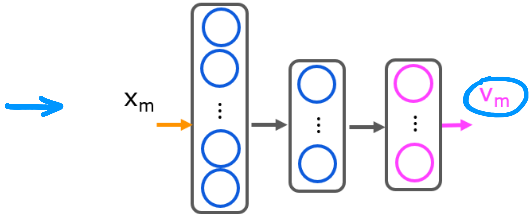


Content-based Filtering

TensorFlow Implementation




```
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()
```