

Week 2

Collaborative Filtering

Collaborating Filtering Example: Movie Ratings

Example dataset (Assuming you have x_1 and x_2 features)

Modeling Cost Function for Movie Rating

Learning Parameters for Each User's w and b

Learning $x^{(i)}$ (Feature vector for movie i):

Putting them together

How to Minimize this Cost Function (Using Gradient Descent)

Collaborative Filtering Example: Binary Labels

Regression to Binary Classification

Recommender System

Mean Normalization

TensorFlow implementation of Collaborative Filtering

Gradient descent in TensorFlow

Finding related items

Limitations to Collaborative Filtering

Content-based Filtering

Content-based vs Collaborative Filtering

Example of User and Item Features

Notations for CBF

Recommending from a large catalogue

Retrieval

Ranking

Ethics of Recommender Systems

TensorFlow Implementation of Content-based Filtering

Collaborative Filtering

Collaborating Filtering Example: Movie Ratings

Example dataset (Assuming you have x_1 and x_2 features)

What if we have features of the movies?

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	$x^{(1)} = \begin{bmatrix} 0.9 \\ 0 \end{bmatrix}$
Romance forever	5	?	?	0	1.0	0.01	
→ Cute puppies of love	?	4	0	?	0.99	0	$x^{(3)} = \begin{bmatrix} 0.99 \\ 0 \end{bmatrix}$
Nonstop car chases	0	0	5	4	0.1	1.0	
Swords vs. karate	0	0	5	?	0	0.9	

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

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

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

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

Modeling Cost Function for Movie Rating

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 $w^{(j)}, b^{(j)}$

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

number of features

Learning Parameters for Each User's w and b

Cost function

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

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

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

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

Learning $x^{(i)}$ (Feature vector for movie i):

Cost function

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

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

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

→ To learn $x^{(1)}, x^{(2)}, \dots, x^{(n_m)}$:

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

Putting them together

Collaborative filtering

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

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

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

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

Put them together:

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

	j=1	j=2	j=3
	Alice	Bob	Carol
Movie1	5	5	?
Movie2	?	2	3

→ i = 1
→ i = 2

- **NB:** The first summation can be written as follows:

4.1 Collaborative filtering cost function

The collaborative filtering cost function is given by

$$J(x^{(0)}, \dots, x^{(n_m-1)}, w^{(0)}, b^{(0)}, \dots, w^{(n_u-1)}, b^{(n_u-1)}) = \frac{1}{2} \sum_{(i,j):r(i,j)=1} (w^{(j)} \cdot x^{(i)} + b^{(j)} - y^{(i,j)})^2 + \underbrace{\frac{\lambda}{2} \sum_{j=0}^{n_u-1} \sum_{k=0}^{n-1} (w_k^{(j)})^2 + \frac{\lambda}{2} \sum_{i=0}^{n_m-1} \sum_{k=0}^{n-1} (x_k^{(i)})^2}_{\text{regularization}} \quad (1)$$

The first summation in (1) is "for all i, j where $r(i, j)$ equals 1" and could be written:

$$= \frac{1}{2} \sum_{j=0}^{n_u-1} \sum_{i=0}^{n_m-1} r(i, j) * (w^{(j)} \cdot x^{(i)} + b^{(j)} - y^{(i,j)})^2 + \text{regularization}$$

You should now write `cofiCostFunc` (collaborative filtering cost function) to return this cost.

How to Minimize this Cost Function (Using Gradient Descent)

- For w, b and x → note the notation change

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

x is also a parameter

Collaborative Filtering Example: Binary Labels

- How do we apply our algorithm on 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	? ←

1
0
?

Regression to Binary Classification

From regression to binary classification

- Previously:
 - Predict $y^{(i,j)}$ as $\underline{w^{(j)} \cdot x^{(i)} + b^{(j)}}$
 - For binary labels:
Predict that the probability of $y^{(i,j)} = 1$
is given by $\underline{g(w^{(j)} \cdot x^{(i)} + b^{(j)})}$
- where $\underline{g(z) = \frac{1}{1+e^{-z}}}$

Cost function for binary application

Previous cost function:

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

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

$$L(f_{(w,b,x)}(x), y^{(i,j)}) = -y^{(i,j)} \log(f_{(w,b,x)}(x)) - (1 - y^{(i,j)}) \log(1 - f_{(w,b,x)}(x)) \quad \leftarrow \text{Loss for single example}$$

$$J(w, b, x) = \sum_{(i,j): r(i,j)=1} L(\underbrace{f_{(w,b,x)}(x)}_{g(w^{(j)} \cdot x^{(i)} + b^{(j)})}, y^{(i,j)}) \quad \leftarrow \text{cost for all examples}$$

Recommender System

Mean Normalization

- What if there is a user who has not made any reviews?
- Normalize each row to $\mu = 0$ by subtracting the μ from every row
- Add μ to the equation/model → User 5 (Eve) will now have the ratings adjusted to the mean
- Alternatively, you can normalize the columns, depending on the context (In this context, a movie that has no ratings yet if normalize columns)

Mean Normalization

$$\begin{array}{ccccc}
 \rightarrow & \begin{bmatrix} 5 & 5 & 0 & 0 & ? \\ 5 & ? & ? & 0 & ? \\ ? & 4 & 0 & ? & ? \\ 0 & 0 & 5 & 4 & ? \\ 0 & 0 & 5 & 0 & ? \end{bmatrix} & \begin{matrix} 2.5 \\ 2.5 \\ 2 \\ 2.25 \\ 1.25 \end{matrix} \\
 \rightarrow & & & &
 \end{array}$$

$$\mu = \begin{bmatrix} 2.5 \\ 2.5 \\ 2 \\ 2.25 \\ 1.25 \end{bmatrix}$$

$$\begin{bmatrix} 2.5 & 2.5 & -2.5 & -2.5 & ? \\ 2.5 & ? & ? & -2.5 & ? \\ ? & 2 & -2 & ? & ? \\ -2.25 & -2.25 & 2.75 & 1.75 & ? \\ -1.25 & -1.25 & 3.75 & -1.25 & ? \end{bmatrix}$$

For user j , on movie i predict:

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

$$y^{(i,j)} \downarrow$$

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

User 5 (Eve):

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

$$\underbrace{w^{(5)} \cdot x^{(1)} + b^{(5)}}_0 + \mu_1 = 2.5$$

TensorFlow implementation of Collaborative Filtering

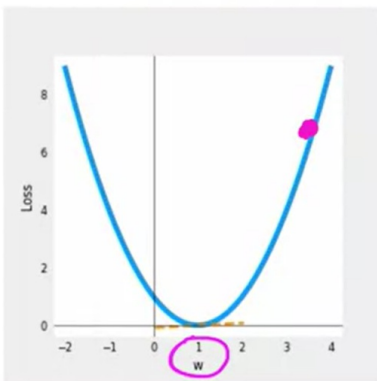
Gradient descent in TensorFlow

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

Gradient descent algorithm
Repeat until convergence

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

Fix $b = 0$ for this example



Custom Training Loop

```

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

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*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)
    
```

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

tf.variables require special function to modify

Implementation in TensorFlow

Gradient descent algorithm

Repeat until convergence

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

```

# 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]))
    
```

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

Finding related items

- Finding a feature x_k that is similar to x_i

Finding related items

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

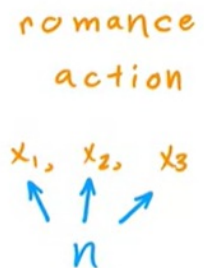
To find other items related to it,

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

i.e. with smallest distance

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

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



Limitations to Collaborative Filtering

- Cold start problem → Unable to give rating predictions on for a new user or product with less ratings

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, ...

Content-based Filtering

Content-based vs Collaborative Filtering

- Match User and product by their ratings
- Match based on features of item and user

Example of User and Item Features

- Note that number of features for user and item can differ

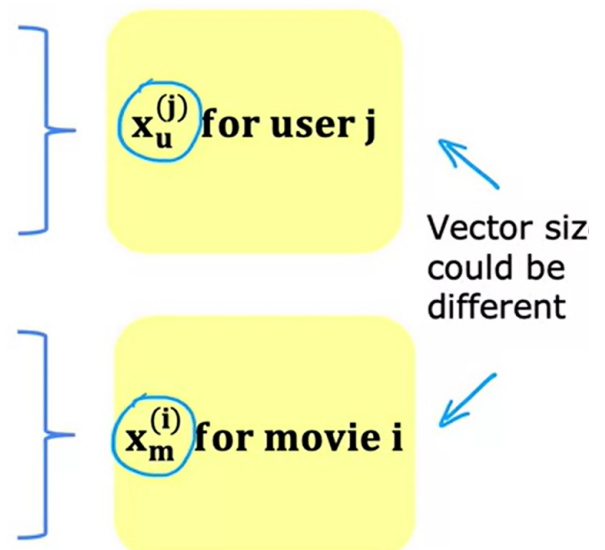
Examples of user and item features

User features:

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

Movie features:

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

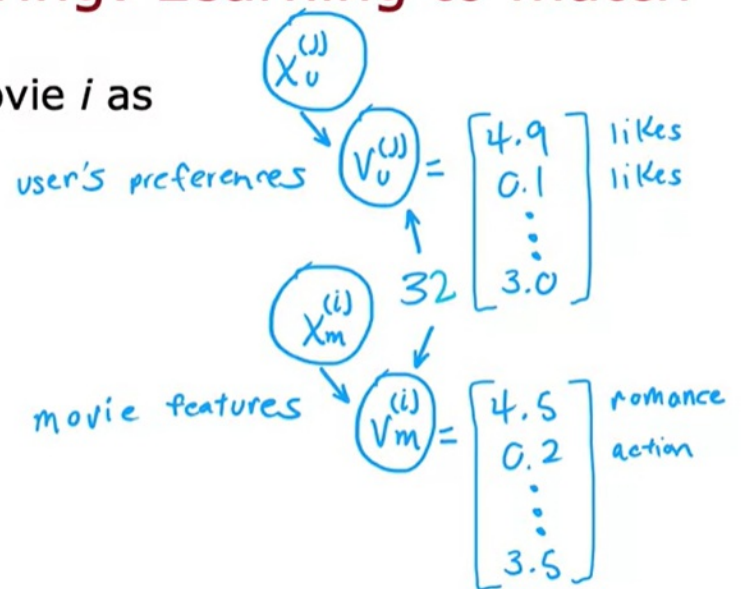
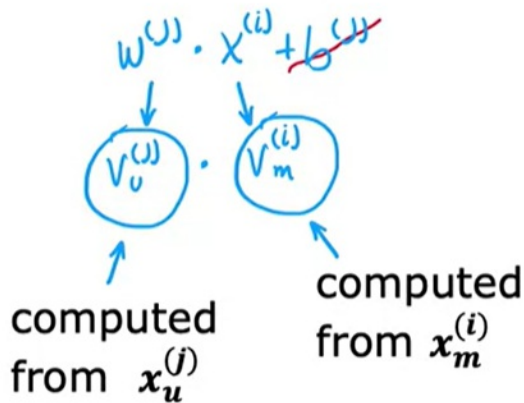


Notations for CBF

- IMPORTANT: $V_u V_u^T$ and $V_m V_m^T$ has the same dimensions!

Content-based filtering: Learning to match

Predict rating of user j on movie i as



Recommending from a large catalogue

- How to efficiently find recommendations from a large number of items?
- Large scale recommender systems do this in 2 steps:
Retrieval and Ranking

Retrieval

- Retrieving more items will result in better performance, but **slower recommendations**.

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

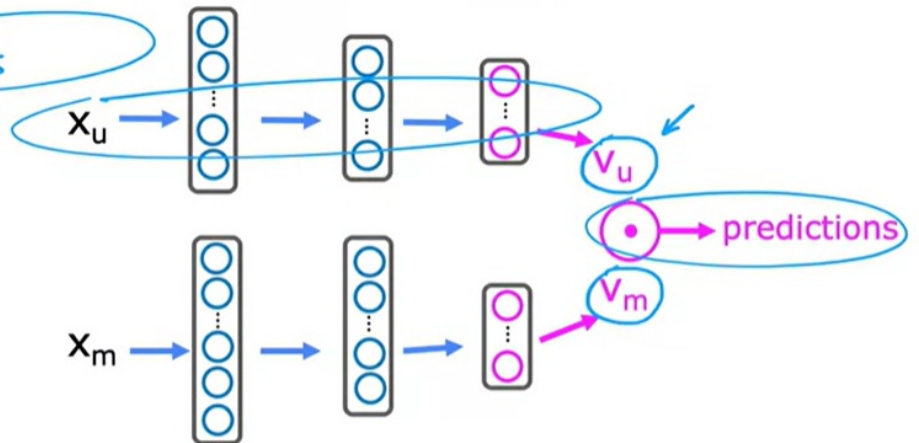
Ranking

- Relatively fast → Retrieval step takes the most time

Two steps: Retrieval & ranking

Ranking:

- Take list retrieved and rank using learned model



- Display ranked items to user

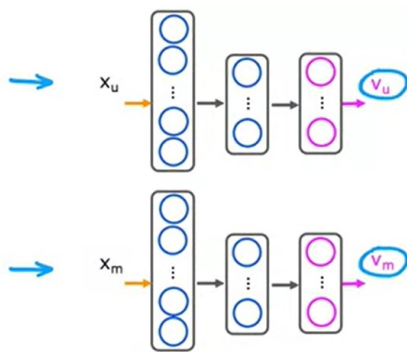
Ethics of Recommender Systems

- Avoid 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

TensorFlow Implementation of Content-based Filtering



```
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()
```

