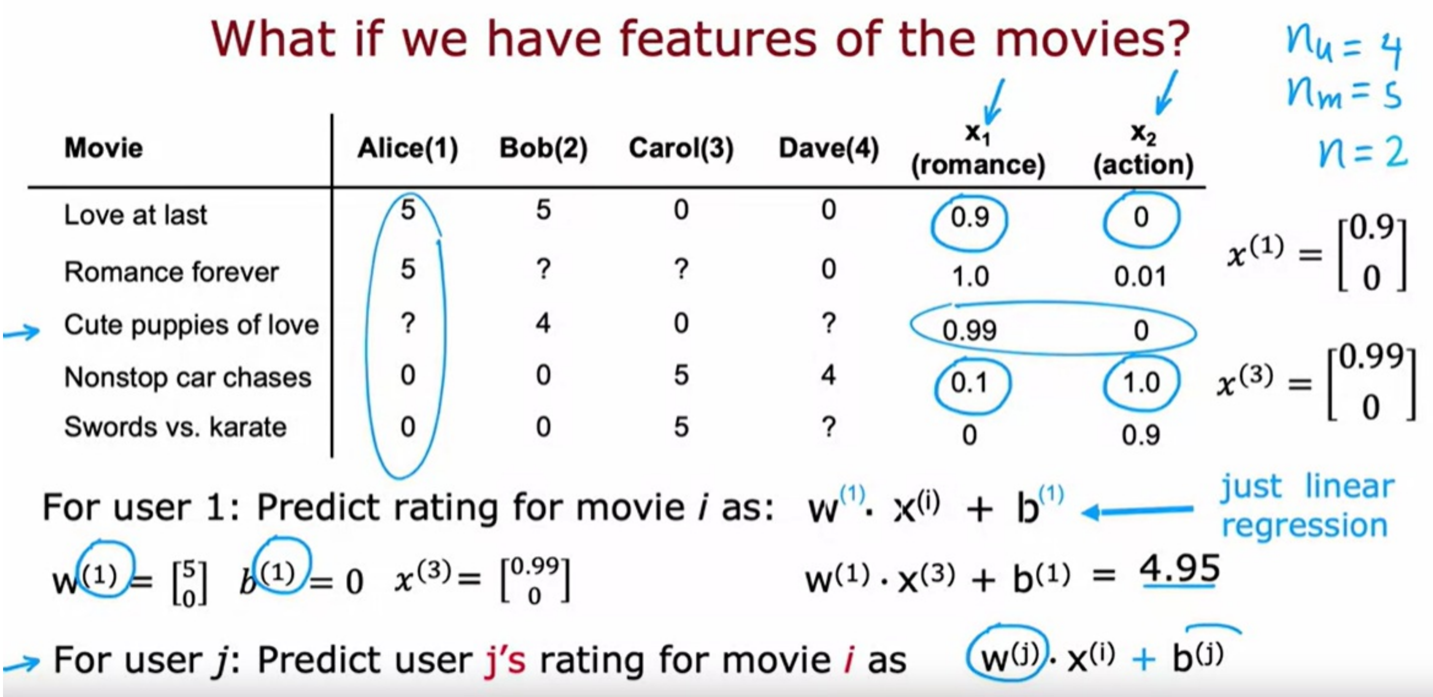
Week 2

Collaborative Filtering

Collaborating Filtering Example: Movie Ratings

Example dataset (Assuming you have x1x_1x1 and x2x_2x2 features)



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

number of features

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

Learning Parameters for Each User's w and bx1x2x_1x_2x1x2

Cost function

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

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

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

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

Learning xix^ixi (Feature vector for movie i):

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

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

Putting them together

Collaborative filtering

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

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

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

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

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

Put them together:

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

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

4.1 Collaborative filtering cost function

The collaborative filtering cost function is given by

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

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

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

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

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

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

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

Loss for single example

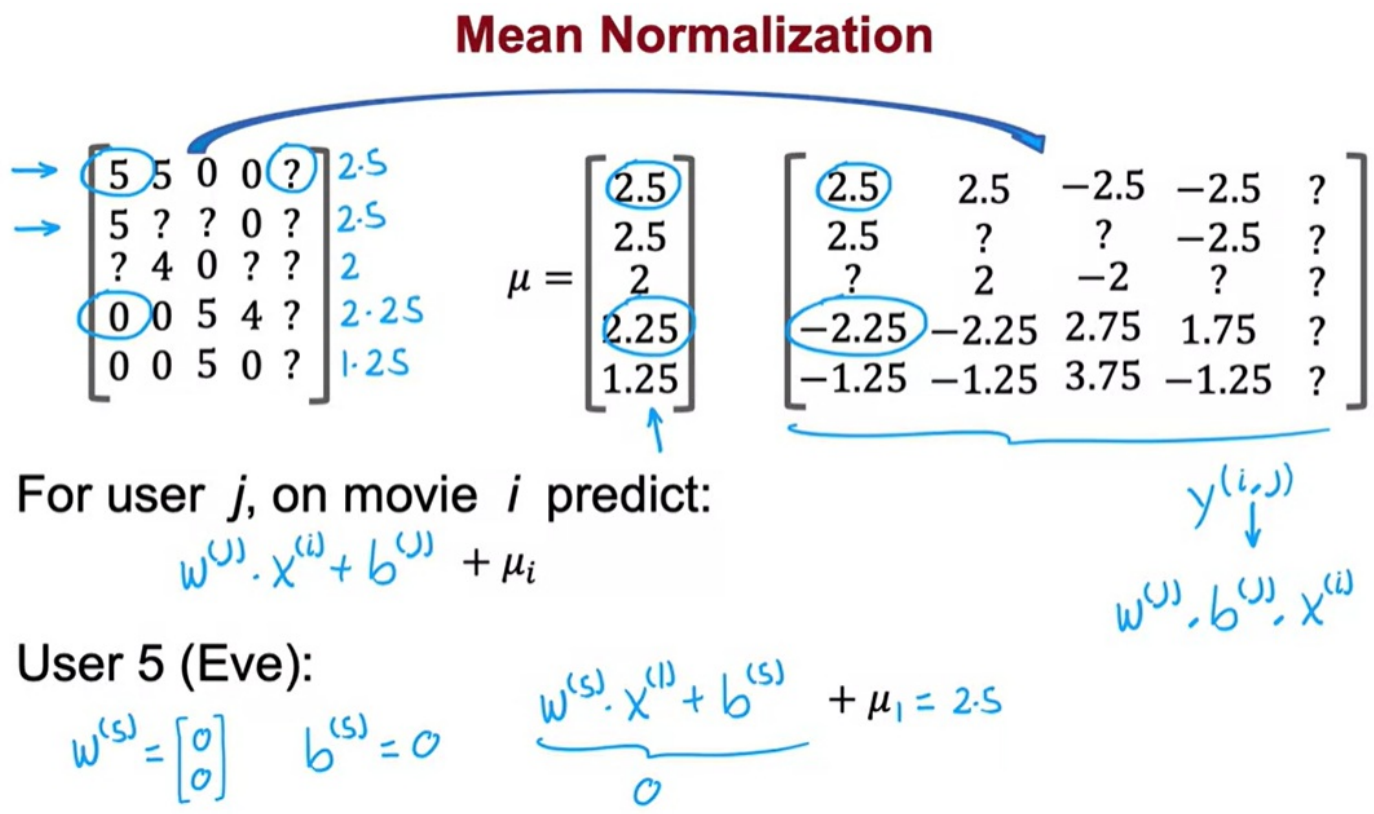
$$J(w,b,x) = \sum_{(i,j):r(i,j)=1} L(\underbrace{f_{(w,b,x)}(x)}_{g(w^{(j)} \cdot x^{(i)} + b^{(j)})}, y^{(i,j)})$$

cost for all examples

Recommender System

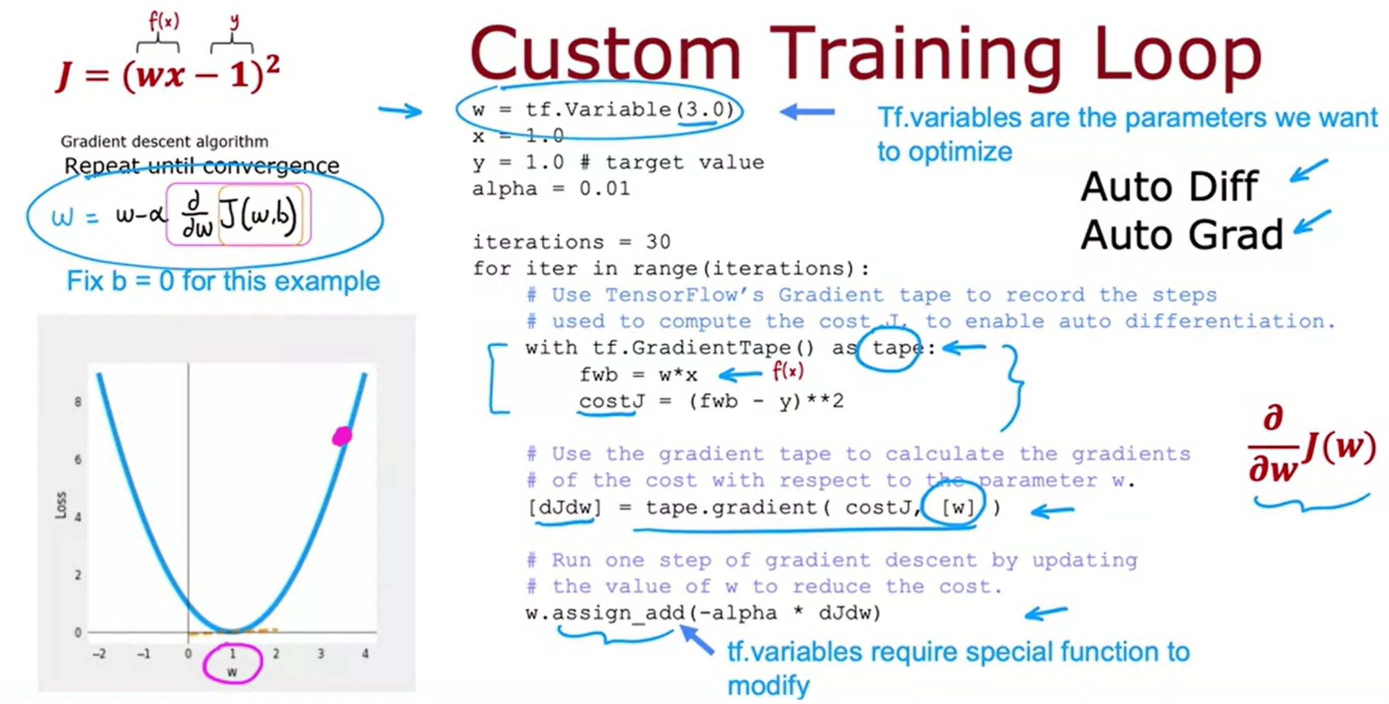
Mean Normalization

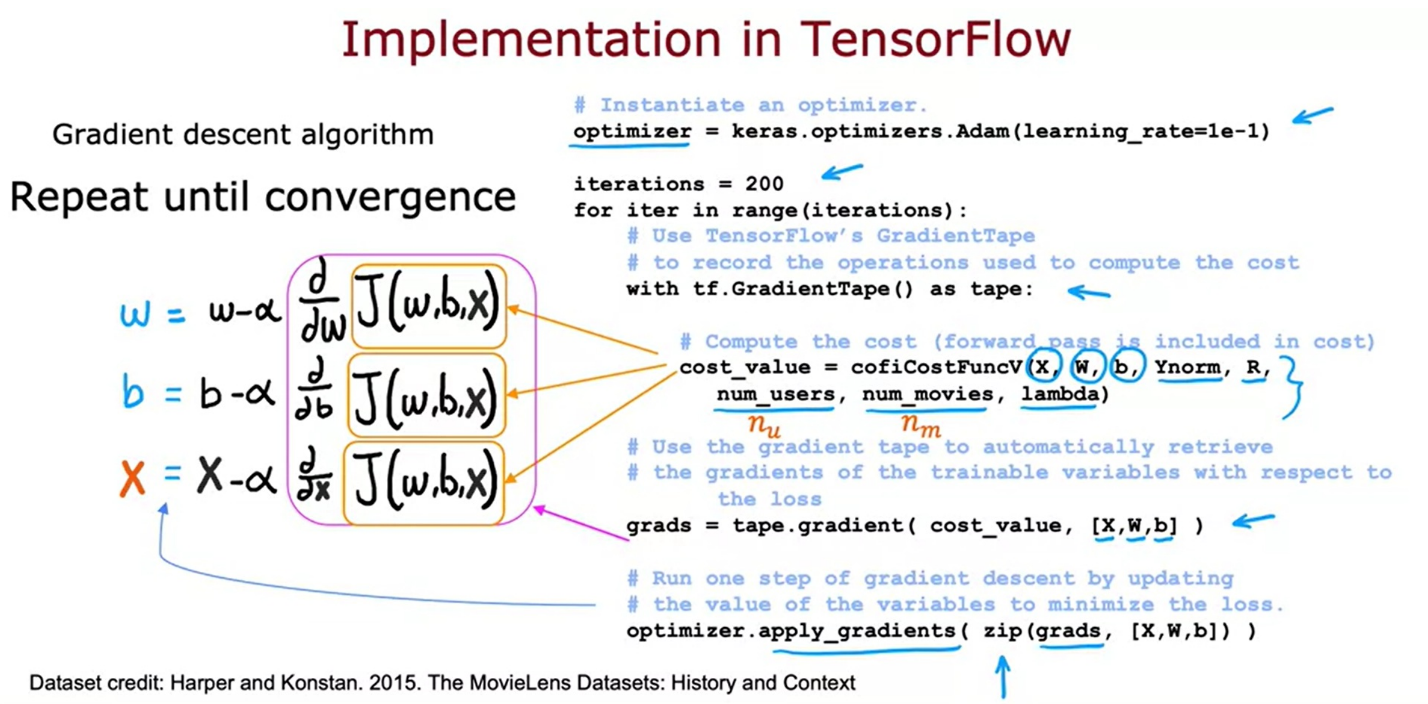
- What if there is a user who has not made any reviews?
- Normalize each row to μ=0μ = 0μ = 0 by subtracting the μμμ from every row
- Add μμμ to the equation/model → User 5 (Eve) will now have the ratings adjusted to the mean
- Alternatively, you can normalize the columns, depending on the context (In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



TensorFlow implementation of Collaborative Filtering

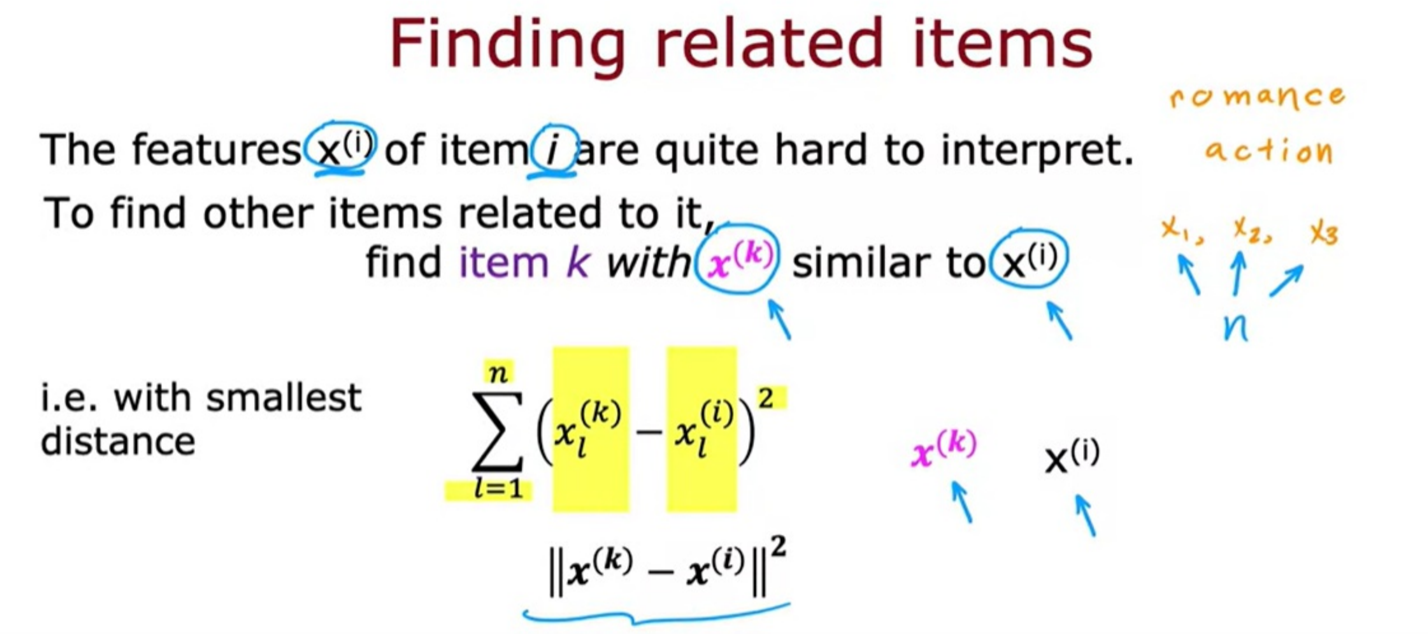
Gradient descent in TensorFlow





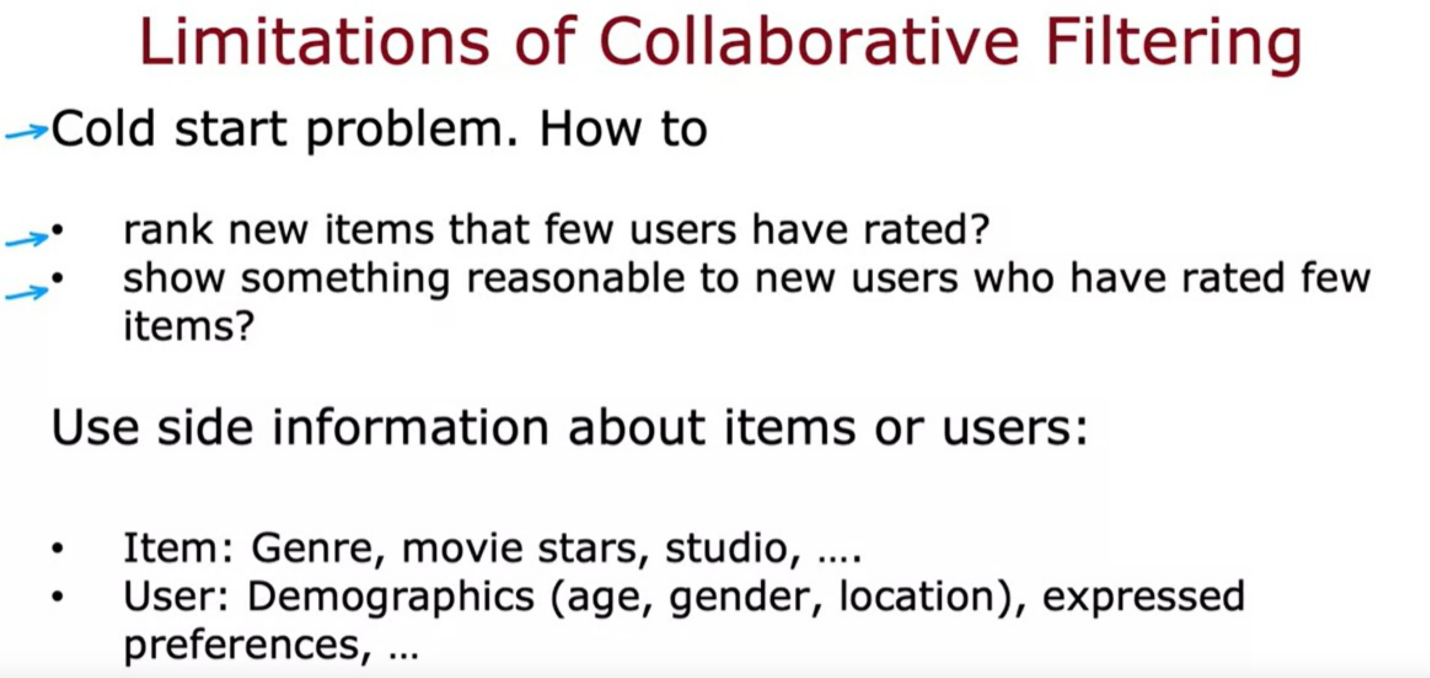
Finding related items

- Finding a feature xkx^kxk that is similar to xix^ixi



Limitations to Collaborative Filtering

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



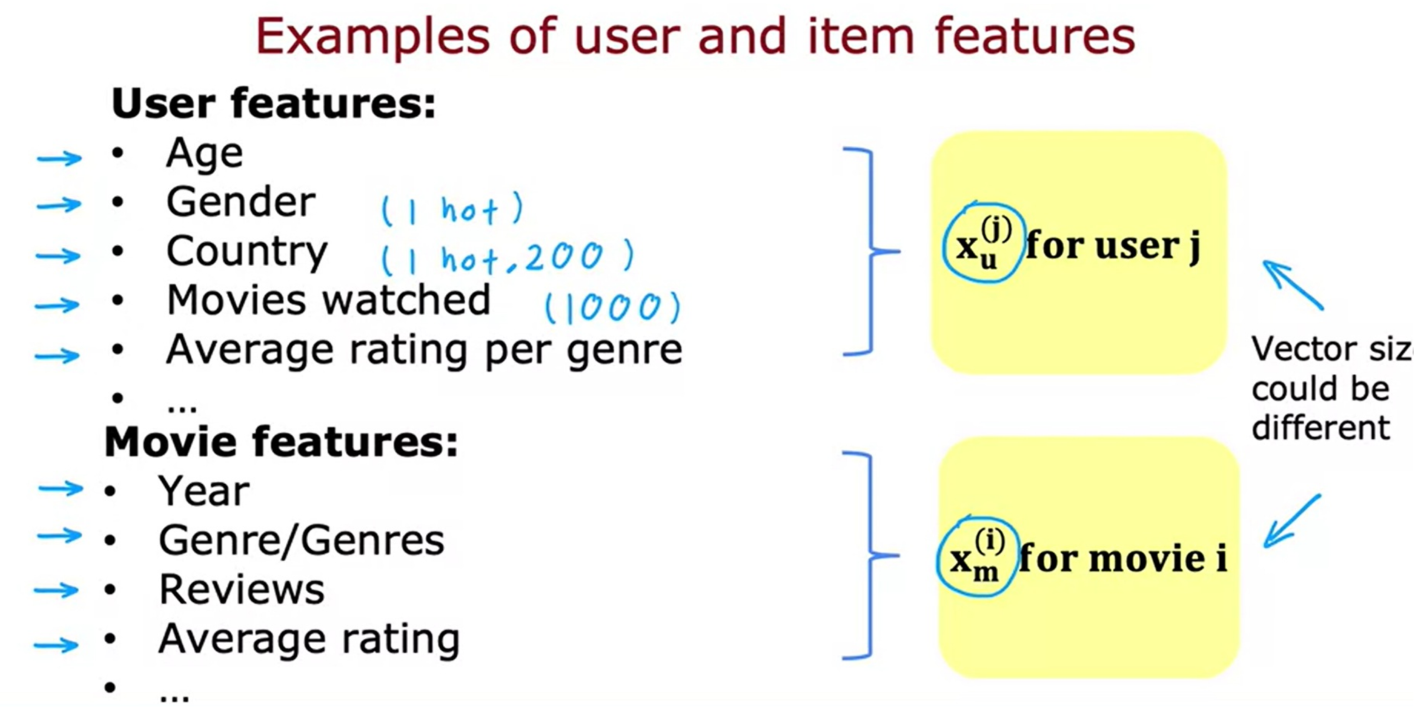
Content-based Filtering

Content-based vs Collaborative Filtering

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

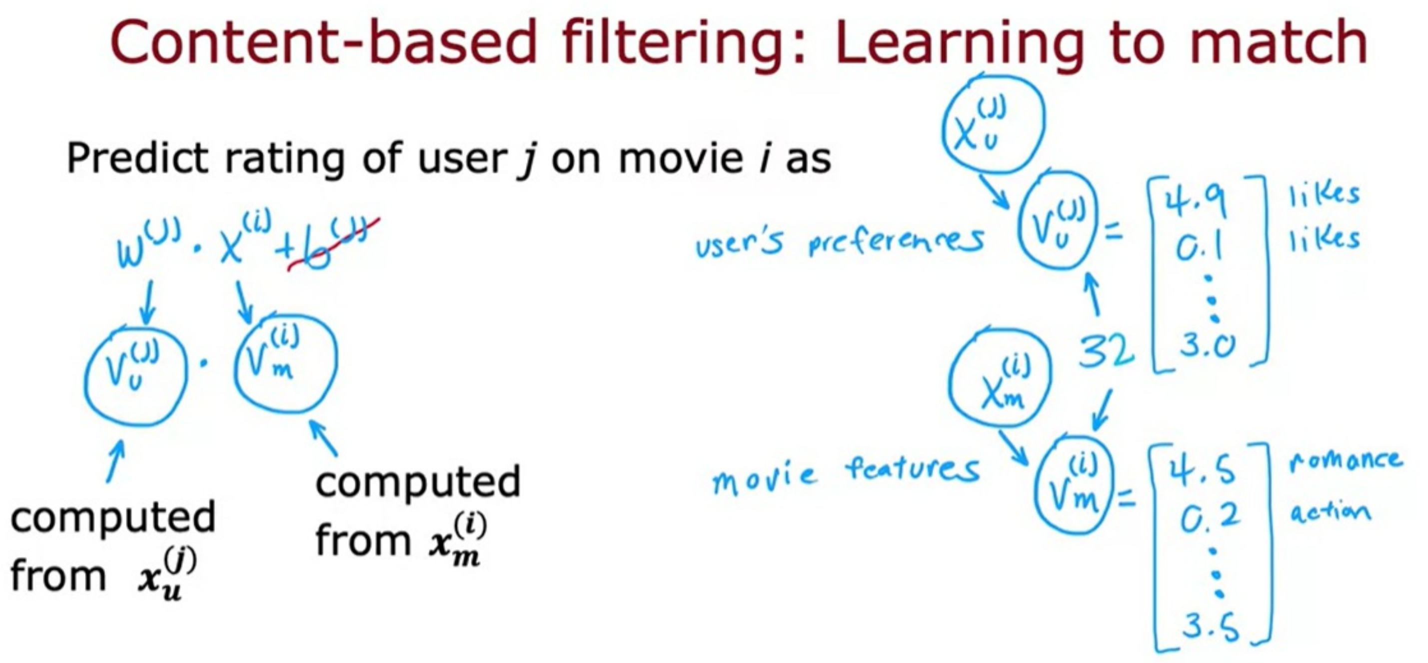
Example of User and Item Features

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



Notations for CBF

- **IMPORTANT: V_u and V_m has the same dimensions!**

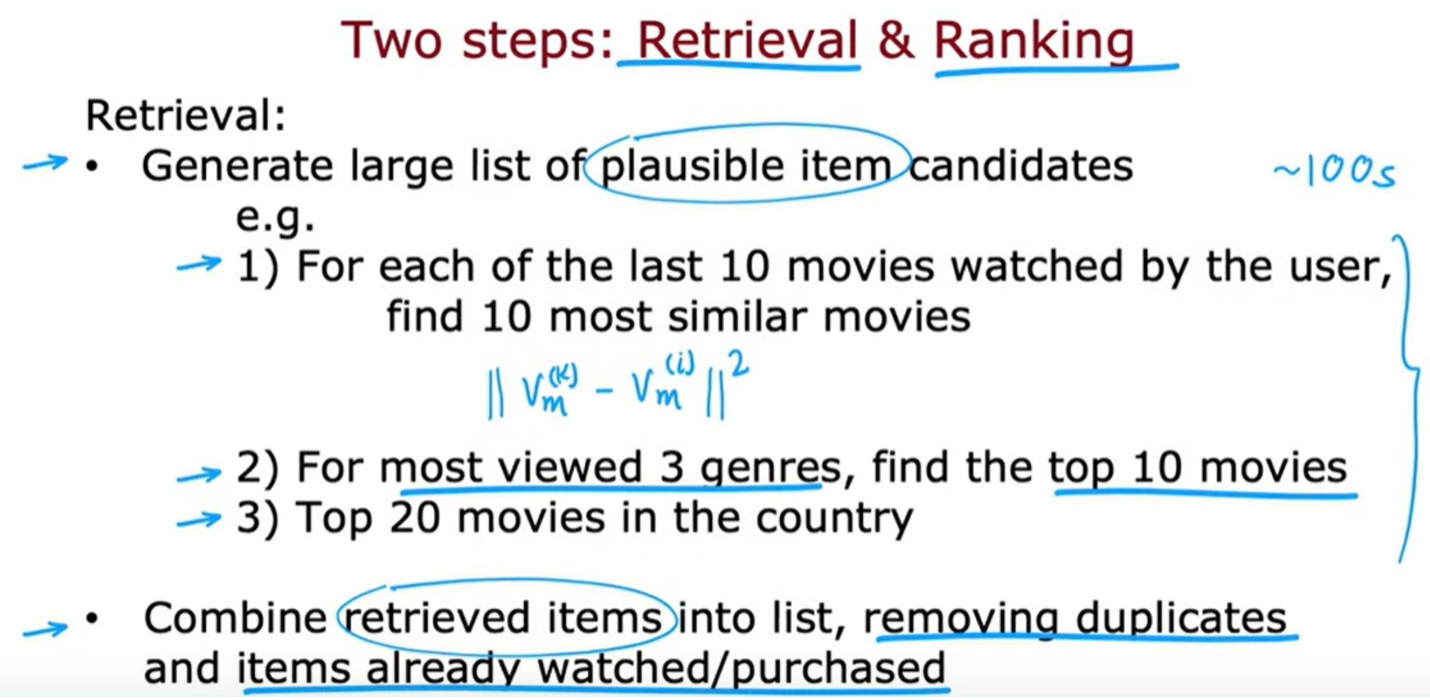


Recommending from a large catalogue

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

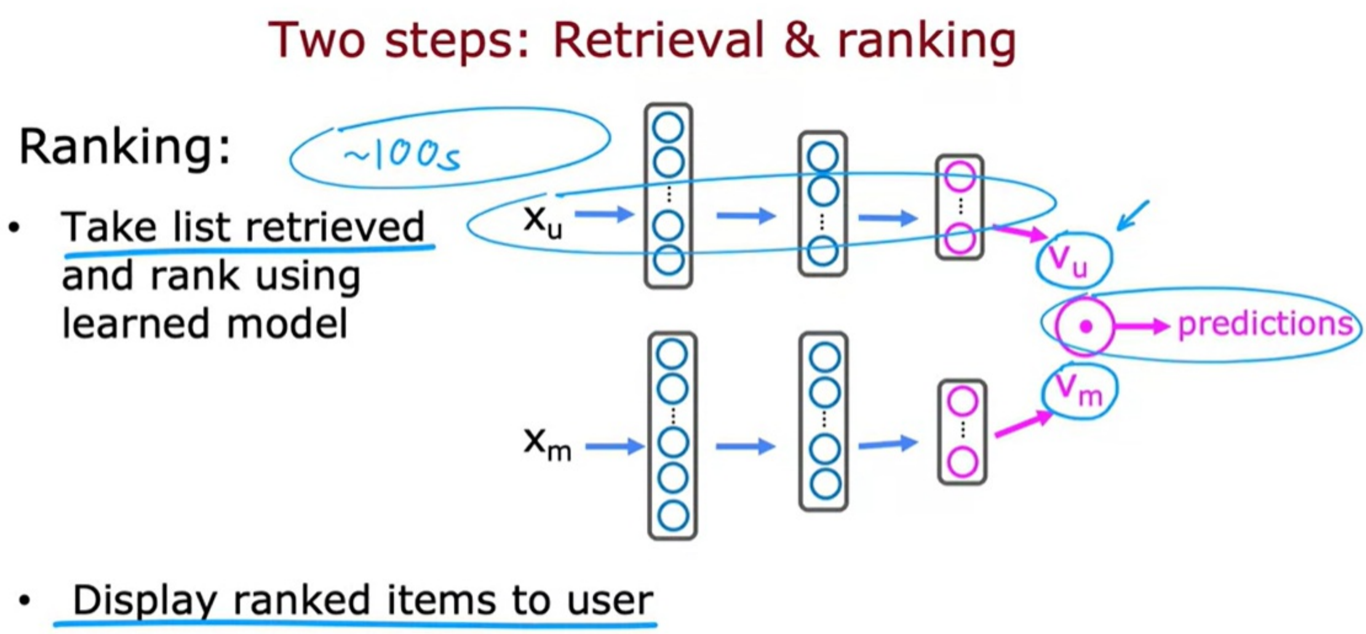
Retrieval

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



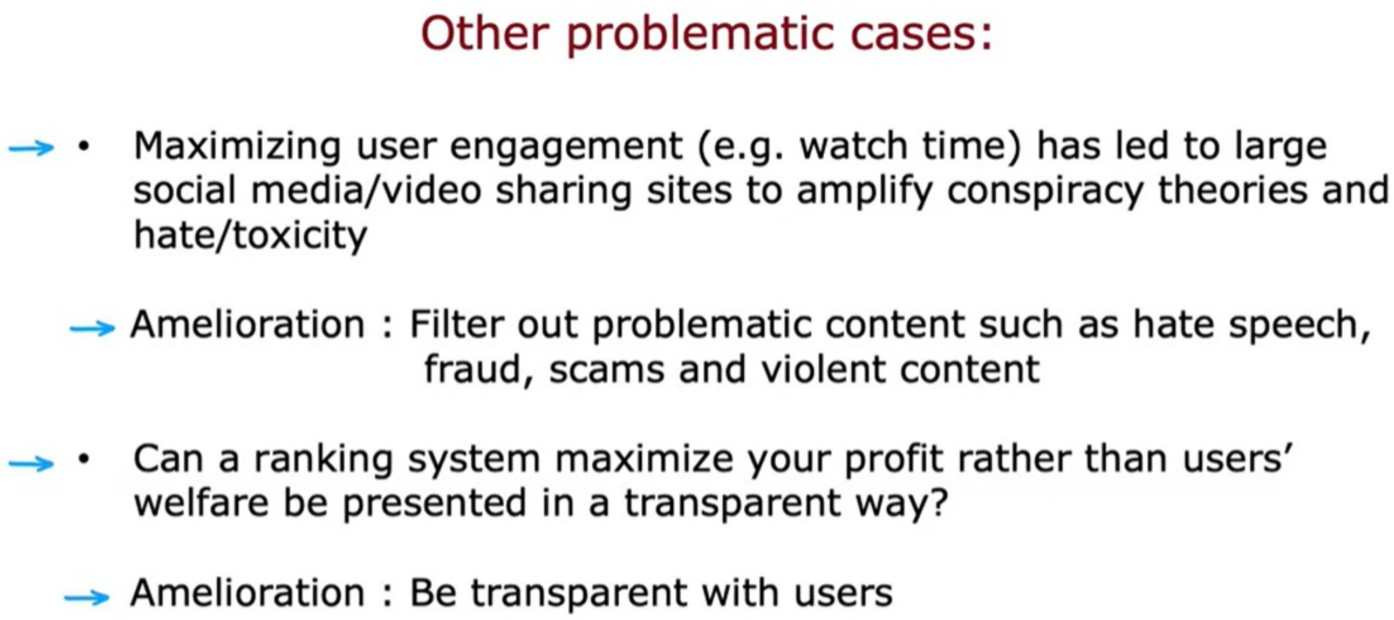
Ranking

- Relatively fast → Retrieval step takes the most time

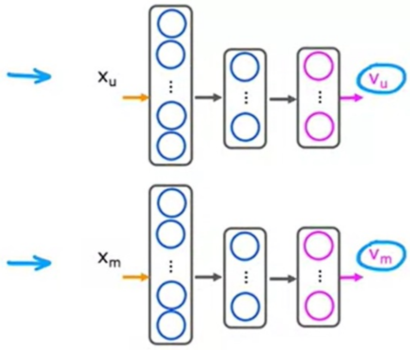


Ethics of Recommender Systems

- Avoid exploitative businesses



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()
```