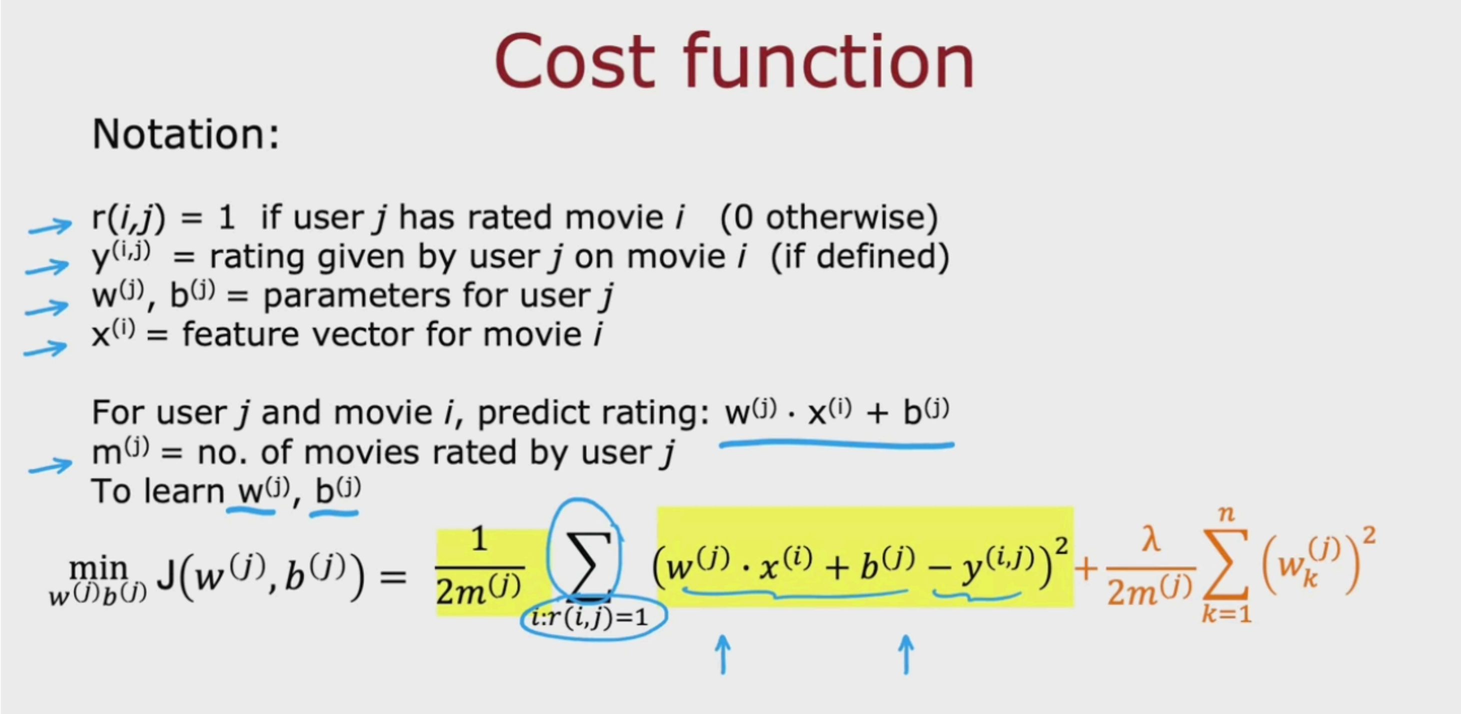
Recommenders

Not only users have features (how much they like each movie) but also movies have their own features (romance, action, ...).



Cost function

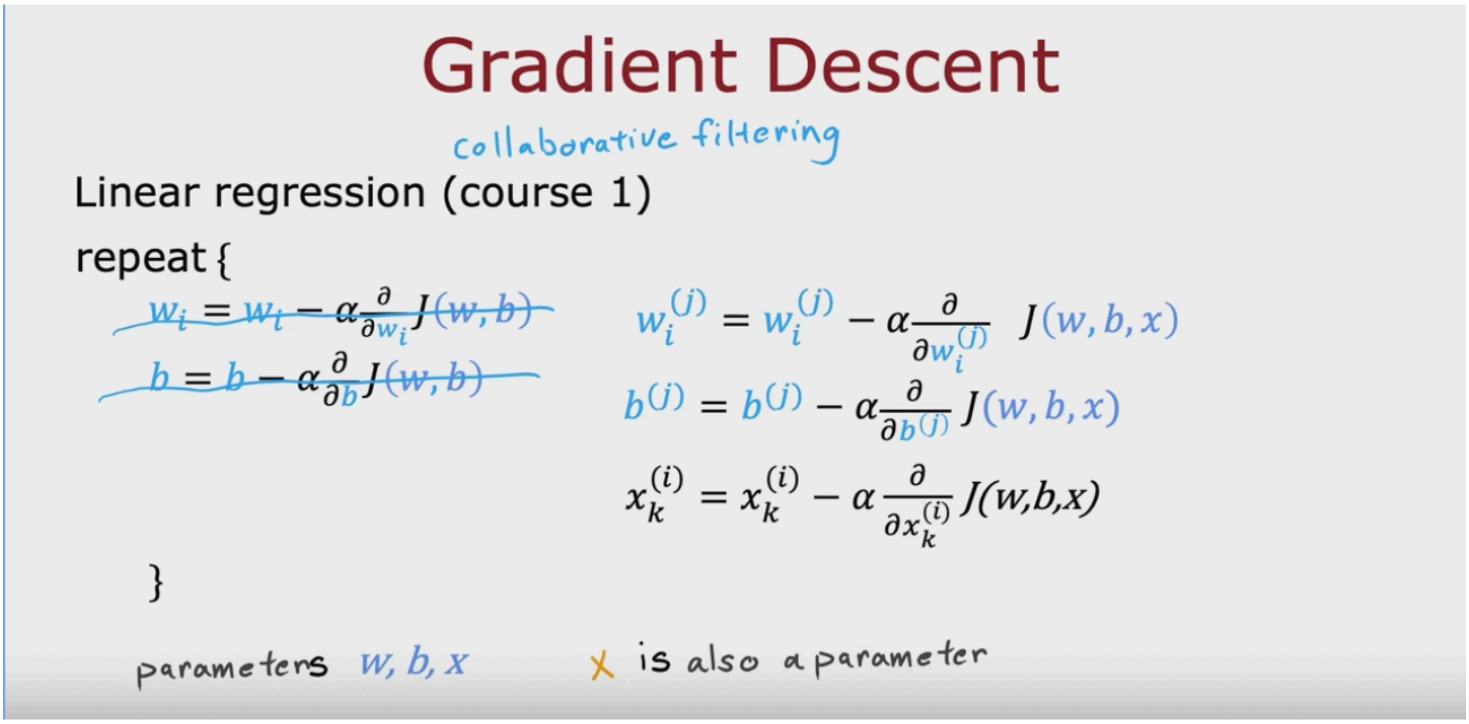
Notation:

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

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

Regularization part shown with red.
n = number of features



Gradient Descent

collaborative filtering

Linear regression (course 1)

repeat {

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

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

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

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

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

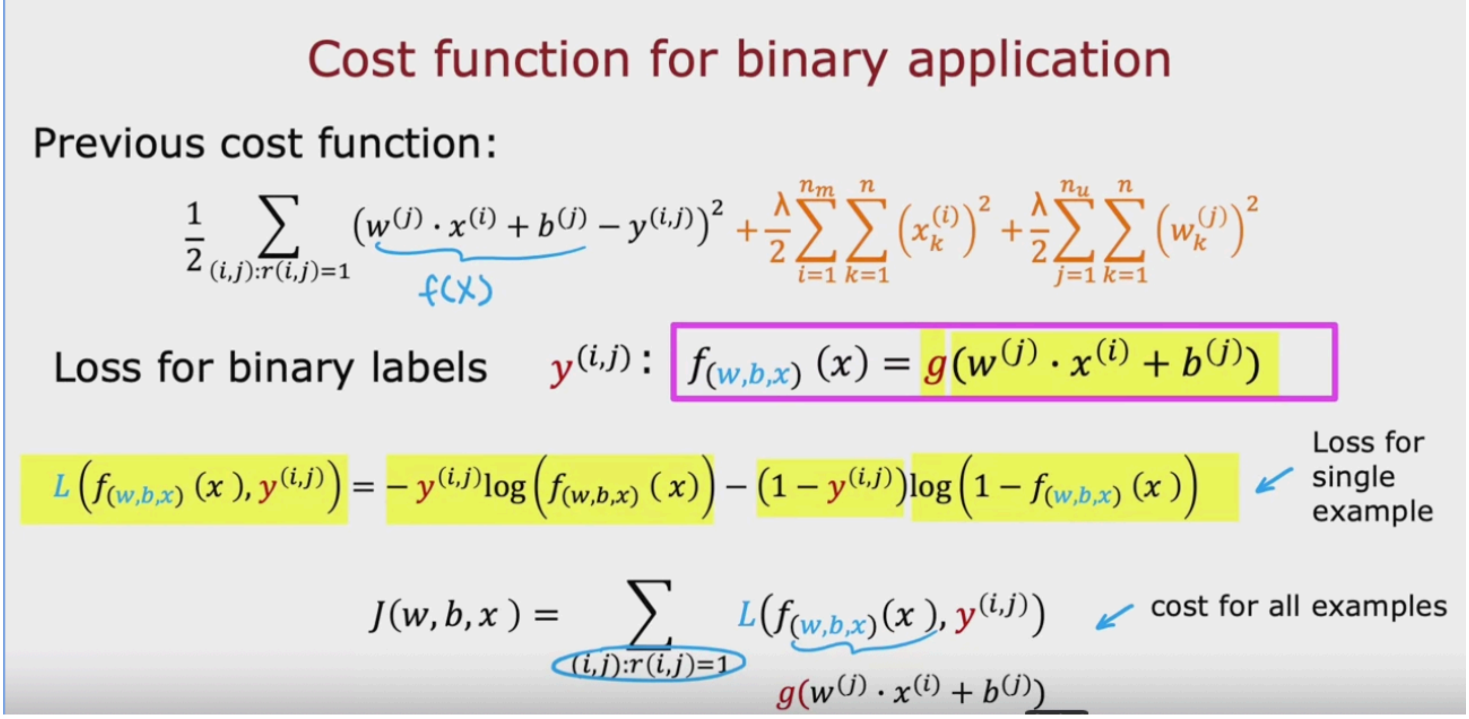
}

parameters w, b, x

x is also a parameter

Binary Labels:

Linear regressin to logistic regression
Regression to Classification



Cost function for binary application

Previous cost function:

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

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

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

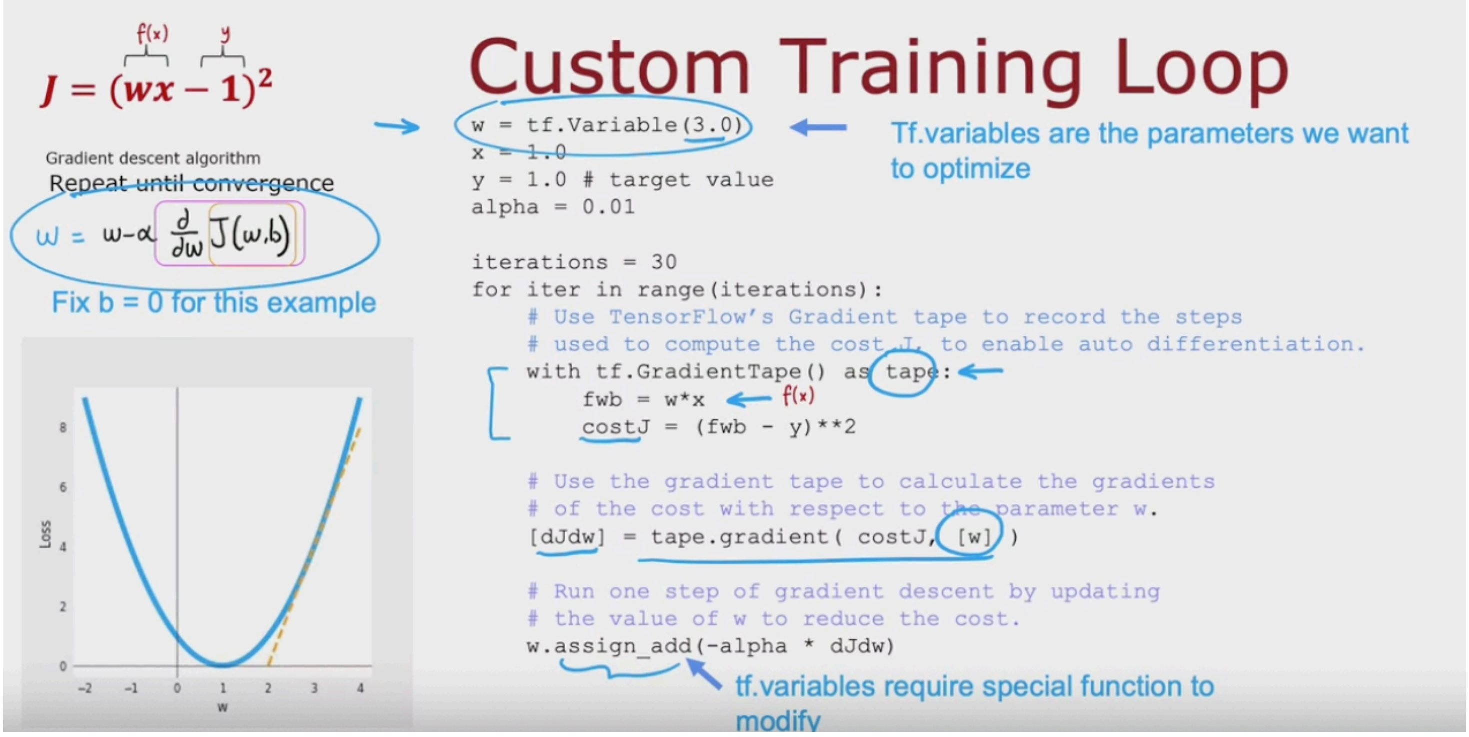
cost for all examples

Mean Normalization

Adding this, help algorithm make better predictions.
For users that does a lot of '?'s. When less information, enables better predictions.
Also makes the algorithm run a little bit faster.

Collaborative Filtering w/ Tensorflow

Automatically calculates derivatives of the cost functions (to optimize the cost function.)



$$J=(\overset{f(x)}{wx}-\overset{y}{1})^2$$

Gradient descent algorithm
Repeat until convergence

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

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

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

tf.variables require special function to modify

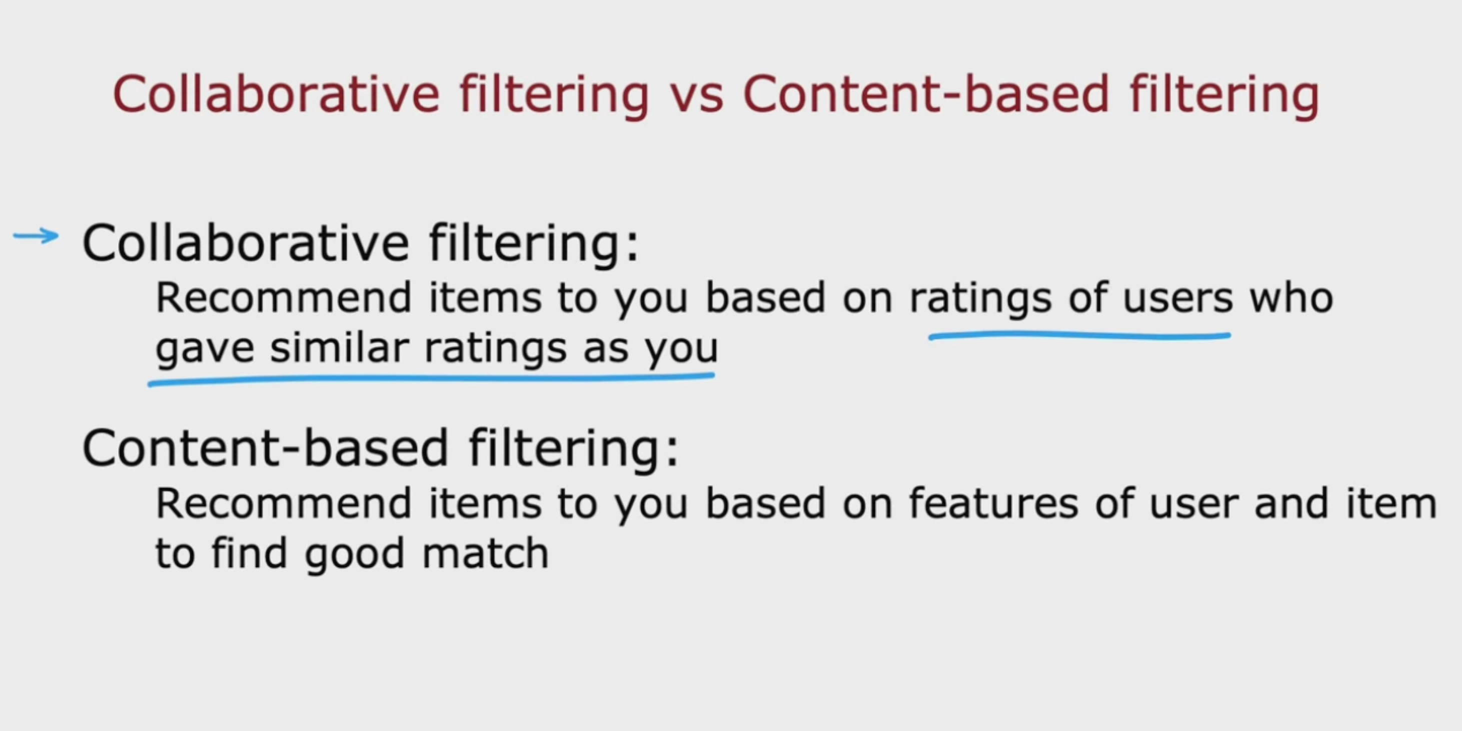
You tell it how to compute the J, it handles the rest.

Finding Related Items

features (x(i)) are hard to interpret.
But they do tell something about the item *collectively*.

find item with similar feature values. (ie. smaller distance)

Content-Based Filtering



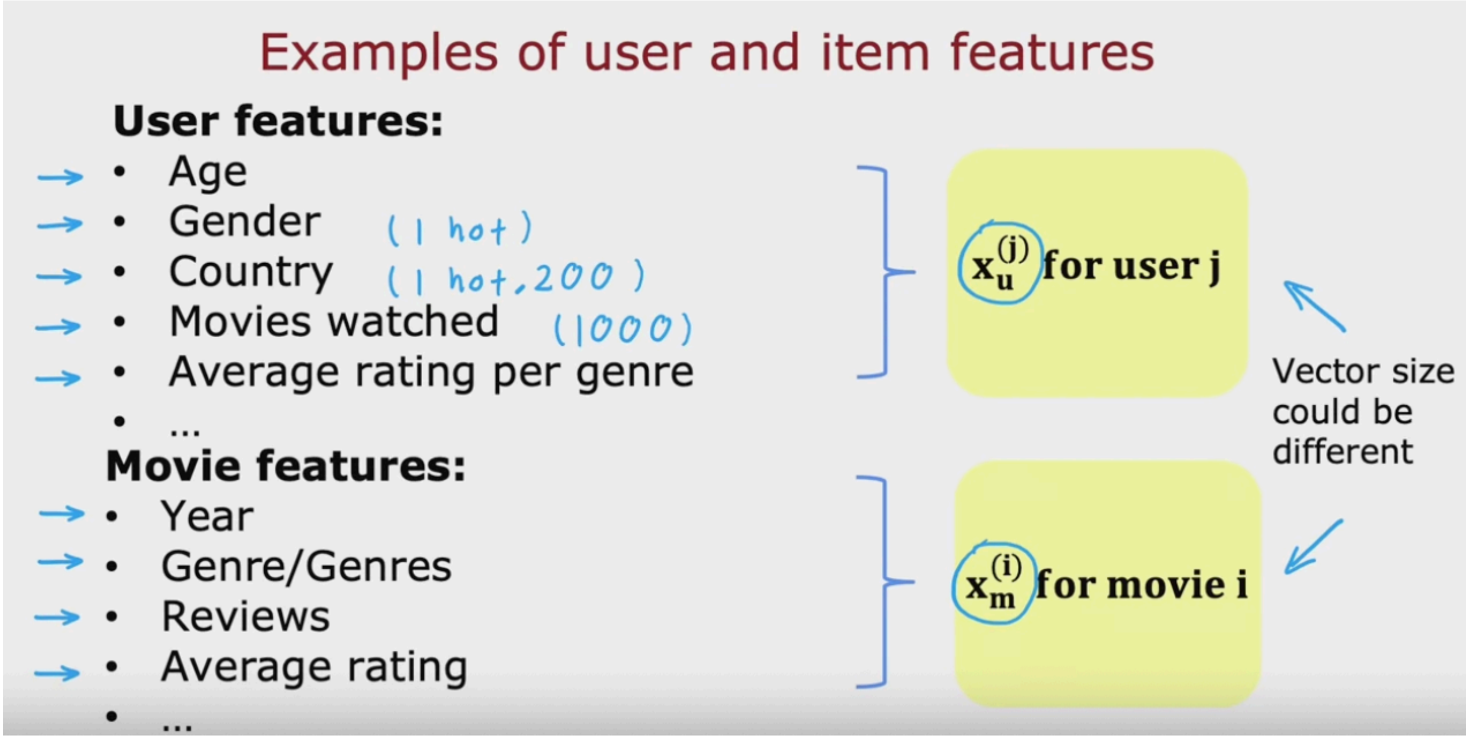
Collaborative filtering vs Content-based filtering

Collaborative filtering:

Recommend items to you based on ratings 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



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

Movie features:

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

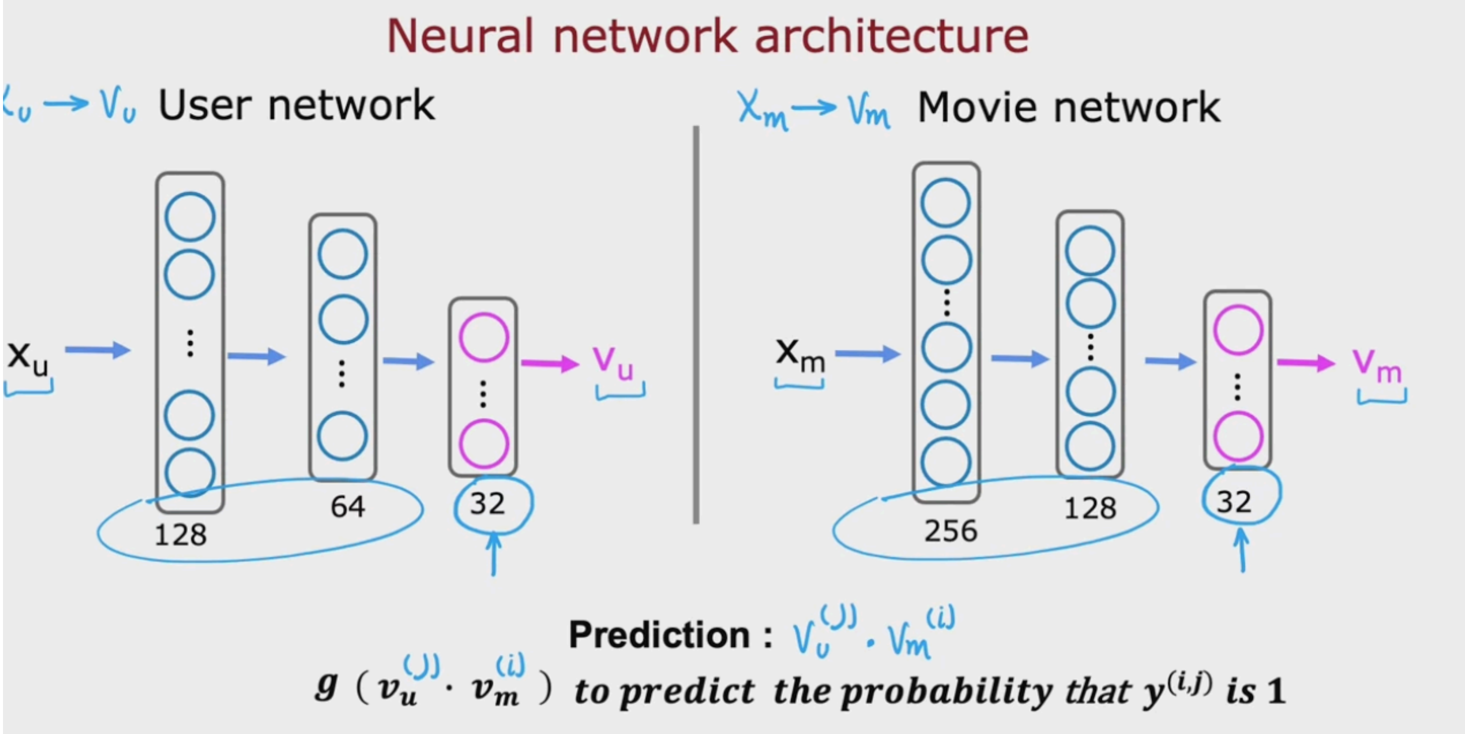
$x_m^{(i)}$ for movie i

Vector size could be different

Content-based Filtering: Learn the match **user & movies**.

Deep Learning for Content Based Filtering

Only the output layer should have the same size.



Neural network architecture

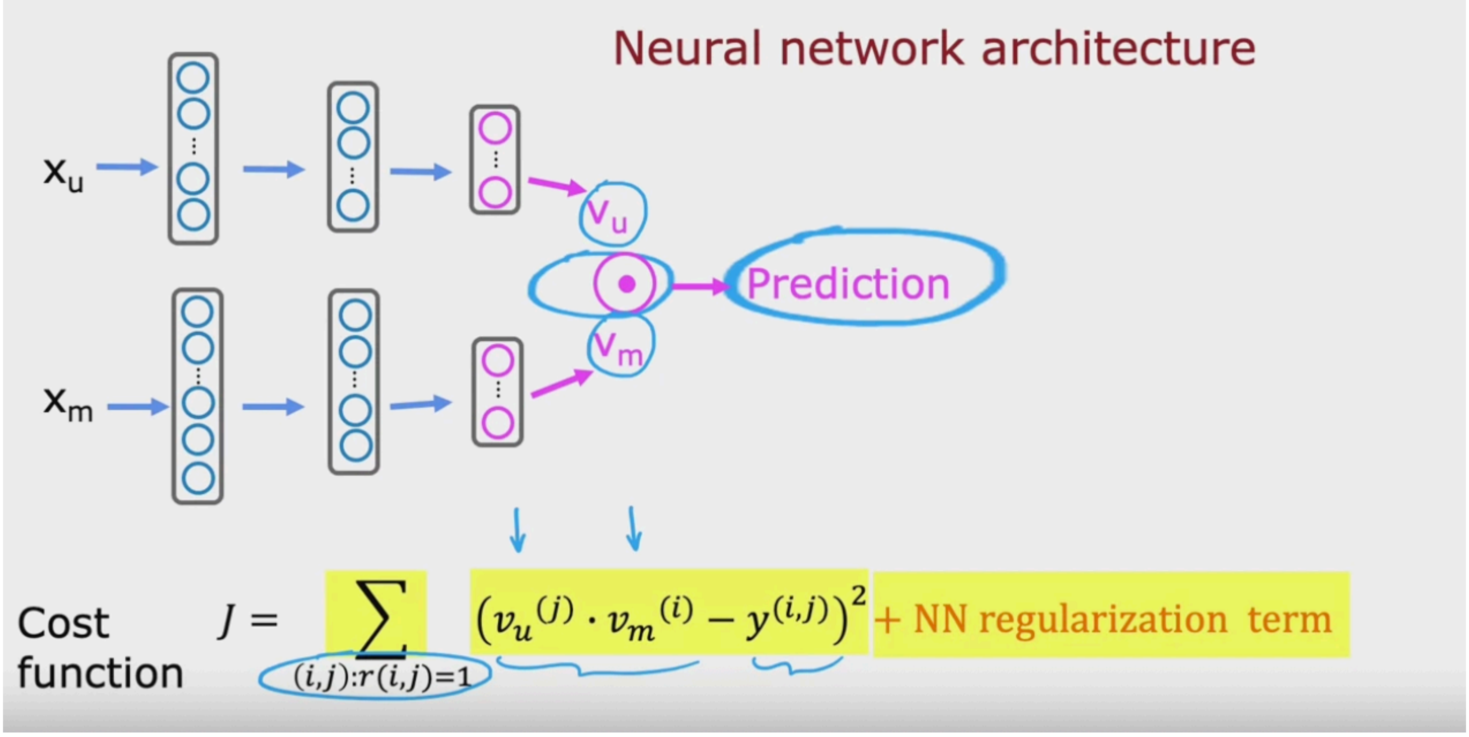
$x_u \rightarrow v_u$ User network

$x_m \rightarrow v_m$ Movie network

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

We can also draw the neural network as a single architecture:



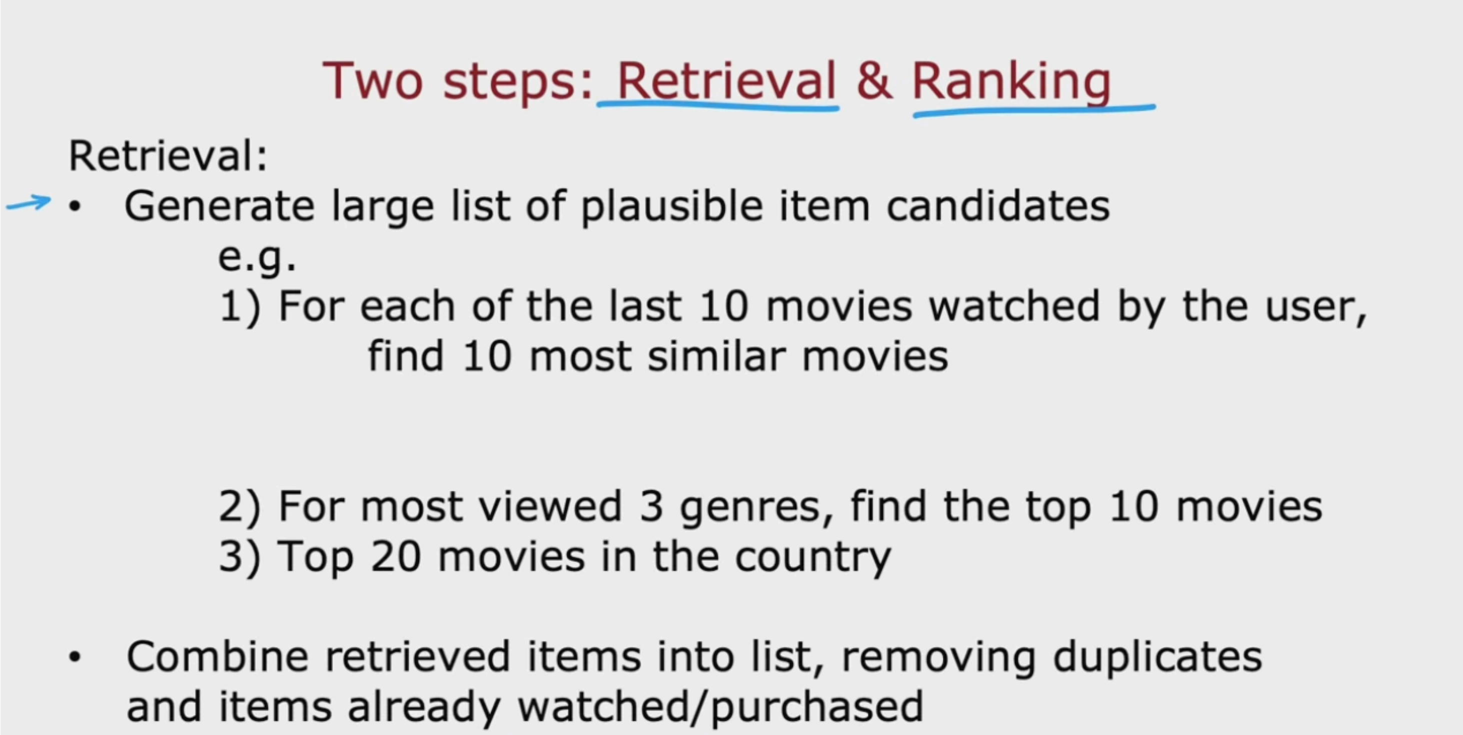
Neural network architecture

Cost function

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

Recommending from a large catalogue

Computationally intensive to run that many items from the neural network every time.



Two steps: Retrieval & Ranking

Retrieval:

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

- 1st Step:
  Eliminate the number of candidates in hand to recommend to user.
  Ensure broad coverage
- 2nd Step:
  Rank whatever left from the elimination using the learned model.
  Display ranked items to user.

How much to retrieve? Test and see.

Ethics:

Mostly, case is what to recommend to the user.
User questions the system? Does the algorithm run for me, or for the company behind/profit maximization?

Payday loans - Squeeze customers - Bid higher for ads
Do not accept ads from exploitative businesses.

TensorFlow for Content Based Filtering