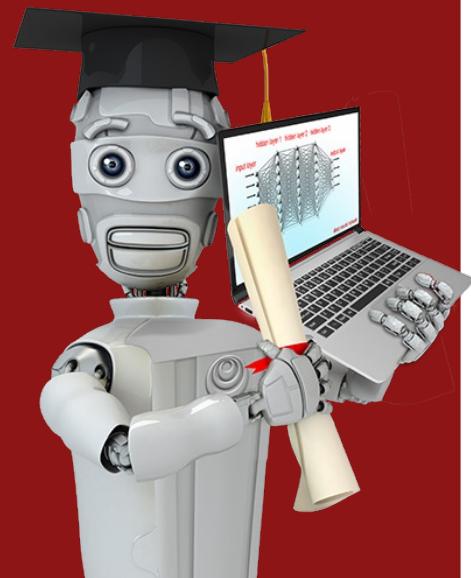


# Copyright Notice

These slides are distributed under the Creative Commons License.

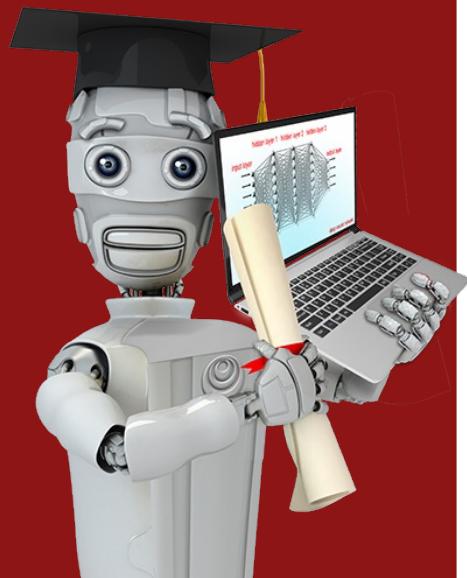
[DeepLearning.AI](#) makes these slides available for educational purposes. You may not use or distribute these slides for commercial purposes. You may make copies of these slides and use or distribute them for educational purposes as long as you cite [DeepLearning.AI](#) as the source of the slides.

For the rest of the details of the license, see <https://creativecommons.org/licenses/by-sa/2.0/legalcode>



# Recommender Systems

---



# Recommender System

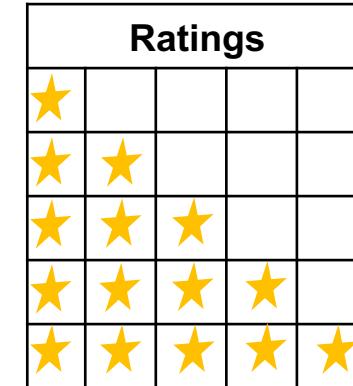
## Making recommendations

# Predicting movie ratings

User rates movies using one to five stars

*zero*

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	?



$$n_u = 4$$

$$r(1,1) = 1$$

$$n_m = 5$$

$$r(3,1) = 0$$

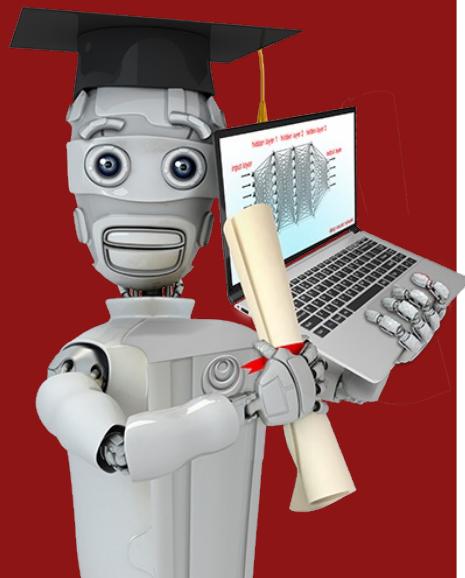
$$y^{(3,2)} = 4$$

$n_u$  = 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  $r(i,j)=1$ )



# 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:  $w^{(1)} \cdot x^{(1)} + b^{(1)}$  ← just linear regression

$$w^{(1)} = \begin{bmatrix} 5 \\ 0 \end{bmatrix} \quad b^{(1)} = 0 \quad x^{(3)} = \begin{bmatrix} 0.9 \\ 0 \end{bmatrix}$$

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

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

# Cost function

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

# Cost function

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

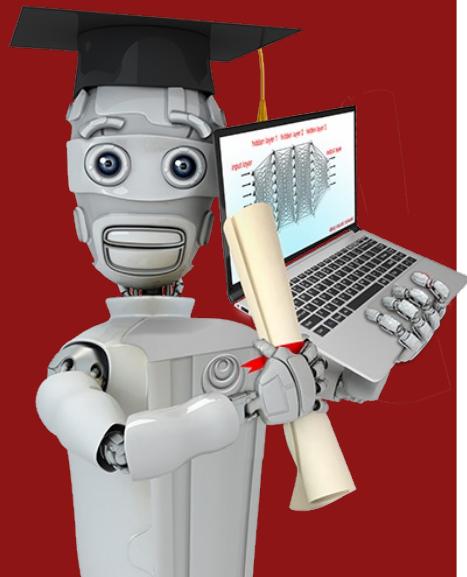
$$J(w^{(j)}, b^{(j)}) = \frac{1}{2} \sum_{i:r(i,j)} (w^{(j)} \cdot x^{(i)} + b^{(j)} - y^{(i,j)})^2 + \frac{\lambda}{2} \sum_{k=1}^n (w_k^{(j)})^2$$

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

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

# 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	?	?
Cute puppies of love	?	4	0	?	?	?
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  $w^{(j)} \cdot x^{(i)} + b^{(j)}$

$$\left. \begin{array}{l} w^{(1)} \cdot x^{(1)} \approx 5 \\ w^{(2)} \cdot x^{(1)} \approx 5 \\ w^{(3)} \cdot x^{(1)} \approx 0 \\ w^{(4)} \cdot x^{(1)} \approx 0 \end{array} \right\} \rightarrow x^{(1)} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

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

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

# Collaborative filtering

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

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

	Alice	Bob	Carol
Movie1	5	5	?
Movie2	?	2	3

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:

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

# Gradient Descent

collaborative filtering

Linear regression (course 1)

repeat {

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

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

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

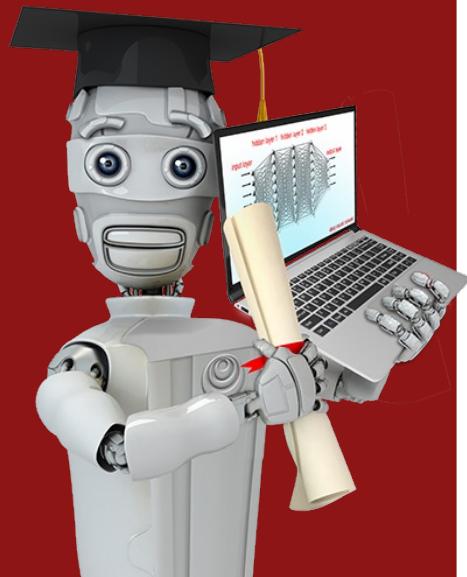
$$b^{(j)} = b^{(j)} - \alpha \frac{\partial}{\partial b^{(j)}} J(w, b, x)$$

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

}

parameters  $w, b, x$

$\times$  is also a parameter



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

# Cost function for binary application

Previous cost function:

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

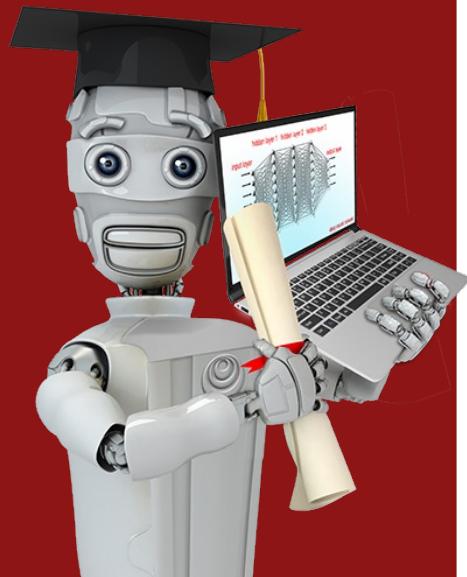
Loss for binary labels  $y^{(i,j)}$ :  $f_{(w,b,x)}(x) = g(w^{(j)} \cdot x^{(i)} + b^{(j)})$

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

Loss for single example

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

cost for all examples  
  $g(w^{(j)} \cdot x^{(i)} + b^{(j)})$



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

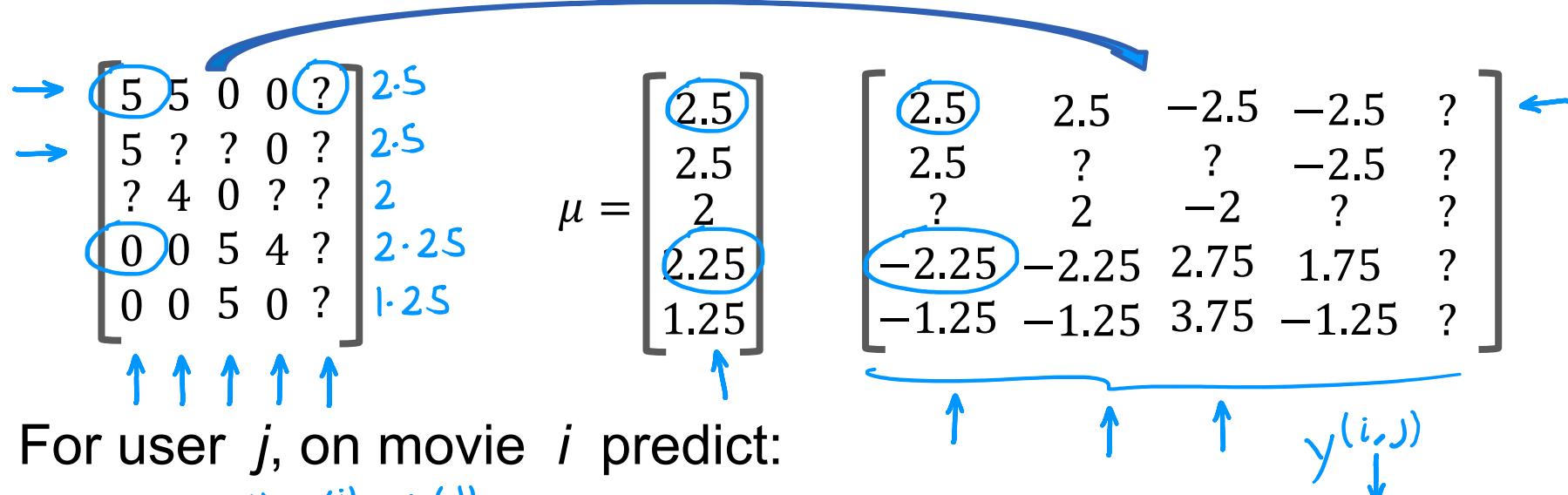
$$\begin{bmatrix} 5 & 5 & 0 & 0 & ? \\ 5 & ? & ? & 0 & ? \\ ? & 4 & 0 & ? & ? \\ 0 & 0 & 5 & 4 & ? \\ 0 & 0 & 5 & 0 & ? \end{bmatrix}$$



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

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

# Mean Normalization



For user  $j$ , on movie  $i$  predict:

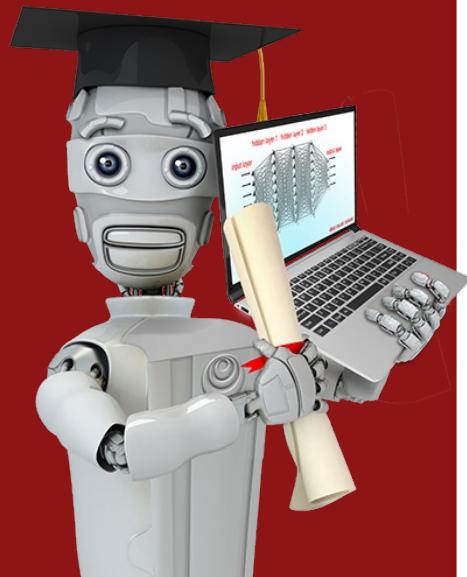
$$w^{(j)} \cdot x^{(i)} + b^{(j)} + \mu_i$$

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

User 5 (Eve):

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

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



# Recommender Systems implementational detail

TensorFlow implementation

# Derivatives in ML

Gradient descent algorithm

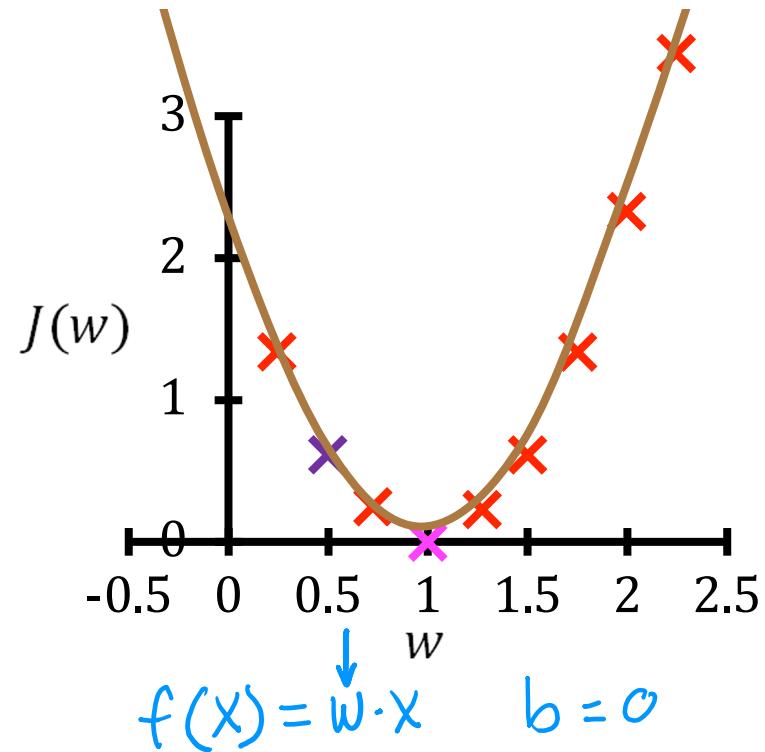
Repeat until convergence

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

Learning rate

Derivative

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

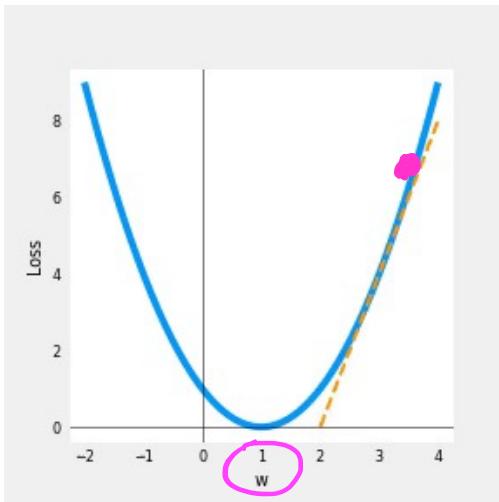


$$J = (\underbrace{wx - 1}_{f(x)})^2 \quad \underbrace{y}_{\text{y}}$$

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
```

```
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 require special function to modify

Auto Diff  
Auto Grad

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

# Implementation in TensorFlow

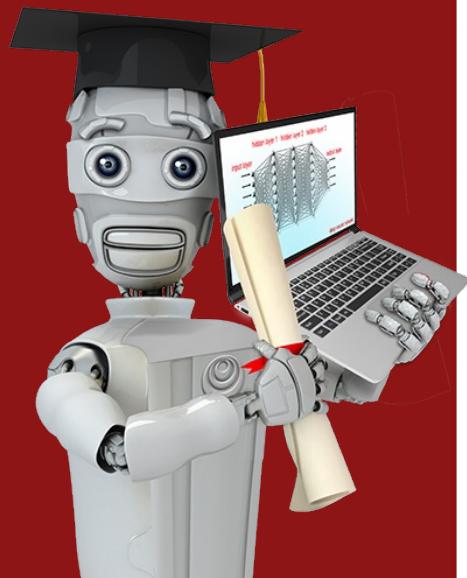
Gradient descent algorithm

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)  
        nu nm  
        # 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



## Collaborative Filtering

# Finding related items

# Finding related items

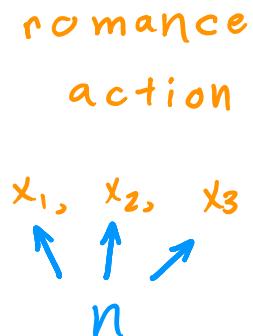
The features  $x^{(i)}$  of item  $i$  are quite hard to interpret.

To find other items related to it,

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

i.e. with smallest  
distance

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



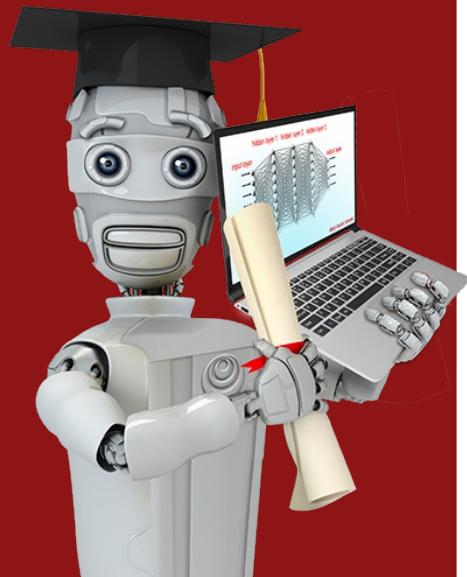
# Limitations of Collaborative Filtering

→ 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

**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)

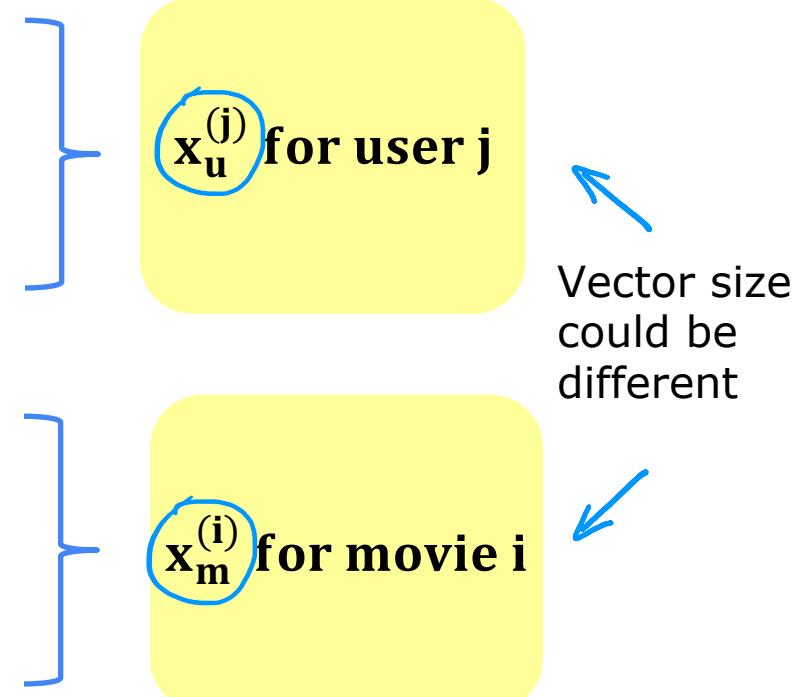
# Examples of user and item features

## User features:

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

## Movie features:

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



# Content-based filtering: Learning to match

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

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

computed from  $x_m^{(i)}$

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

user's preferences

$$v_u^{(j)} = \begin{bmatrix} 4.9 \\ 0.1 \\ \vdots \\ 3.0 \end{bmatrix}$$

likes  
likes

movie features

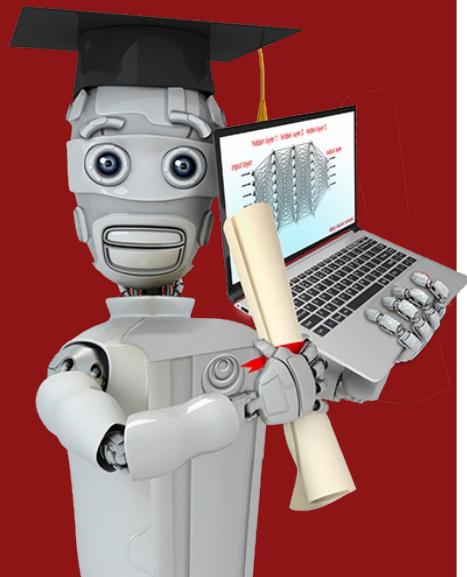
$$v_m^{(i)} = \begin{bmatrix} 4.5 \\ 0.2 \\ \vdots \\ 3.5 \end{bmatrix}$$

romance  
action

$x_u^{(j)}$

$x_m^{(i)}$

32

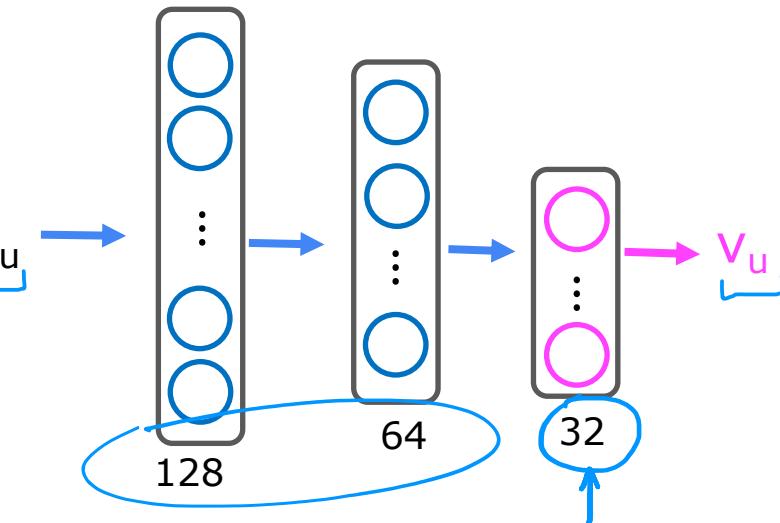


## Content-based Filtering

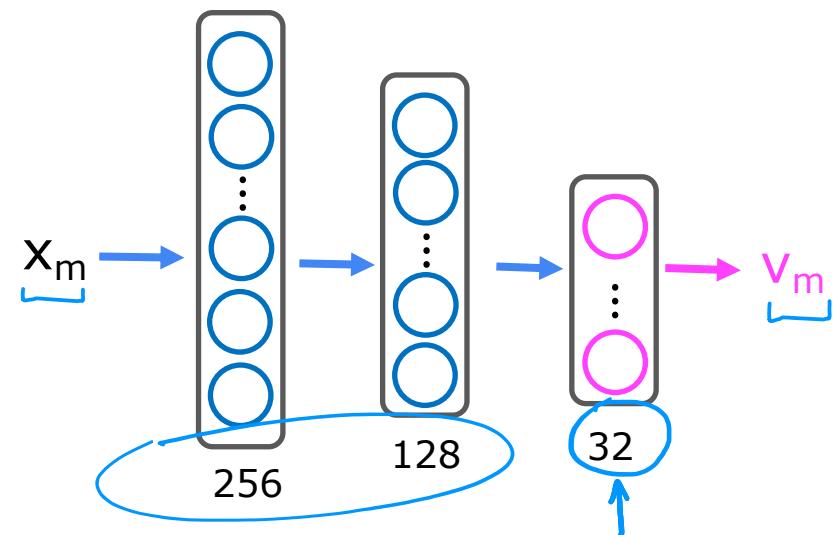
Deep learning for  
content-based filtering

# Neural network architecture

$x_u \rightarrow v_u$  User network

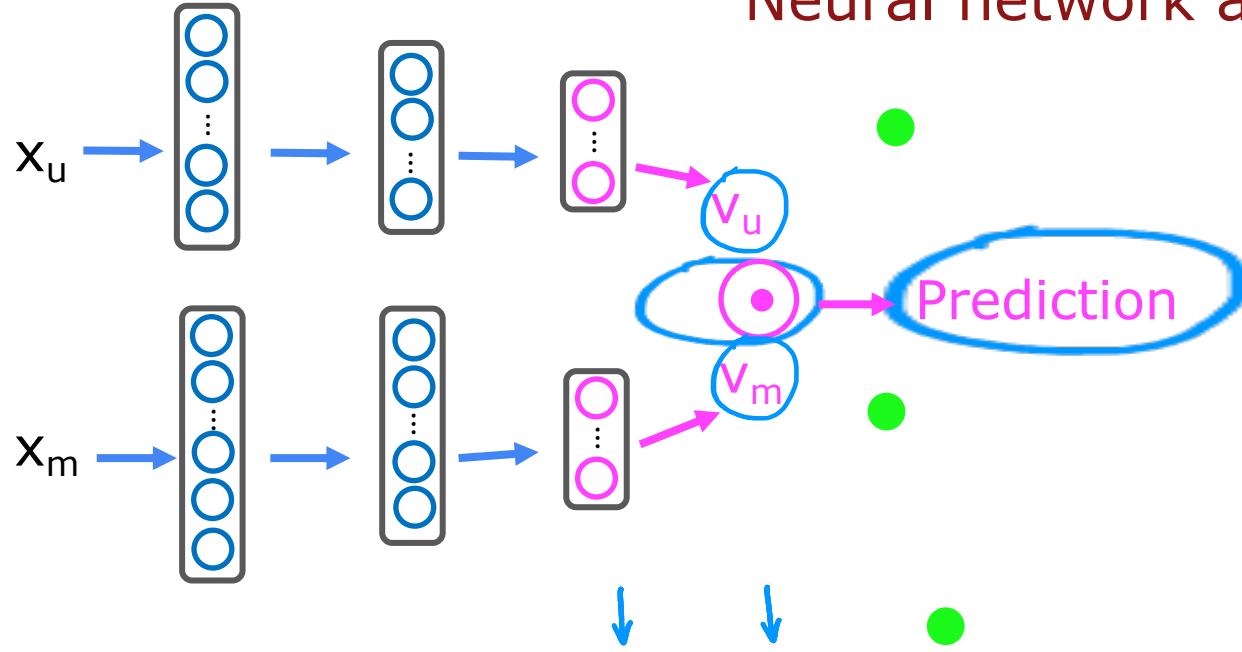


$x_m \rightarrow v_m$  Movie network



**Prediction :**  $v_u^{(ij)} \cdot v_m^{(ij)}$   
 $g(v_u^{(ij)} \cdot v_m^{(ij)})$  to predict the probability that  $y^{(ij)}$  is 1

# Neural network architecture



Cost  
function

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

## Learned user and item vectors:

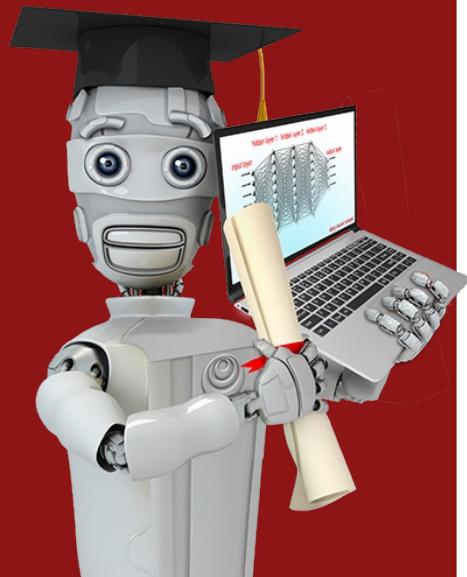
- $v_u^{(j)}$  is a vector of length 32 that describes user j with features  $x_u^{(j)}$
- $v_m^{(i)}$  is a vector of length 32 that describes movie i with features  $x_m^{(i)}$

To find movies similar to movie i:

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

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

- Note: This can be pre-computed ahead of time

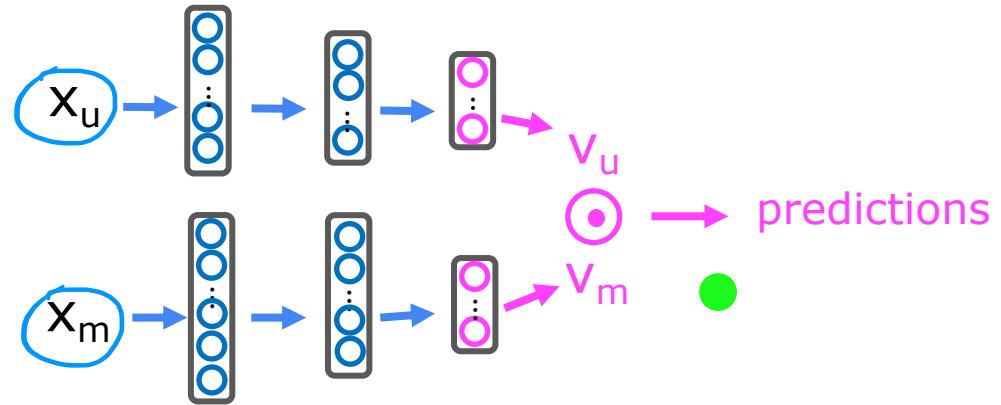


## Advanced implementation

**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

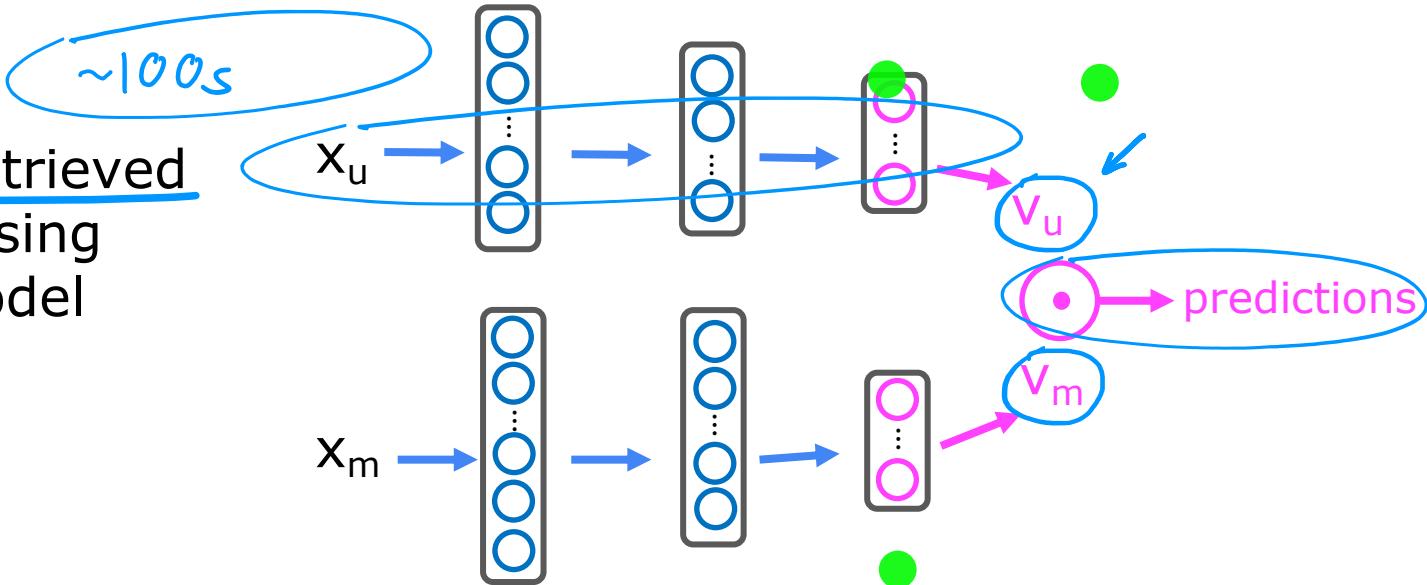
Retrieval:

- • Generate large list of plausible item candidates       $\sim 100s$ 
  - 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:

- Take list retrieved and rank using learned model

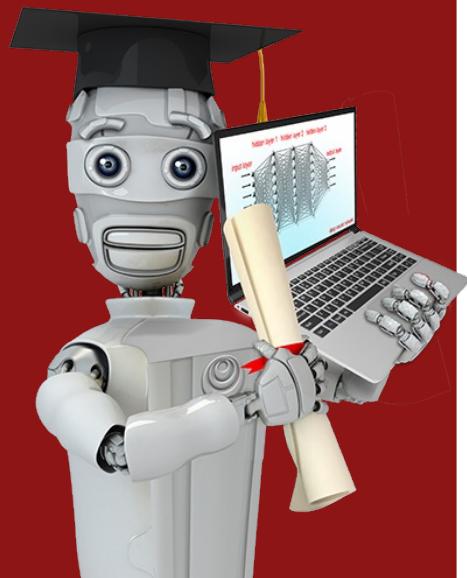


- Display ranked items to user

## 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



## Advanced implementation

Ethical use of  
recommender systems

# What is the goal of the recommender system?

Recommend:

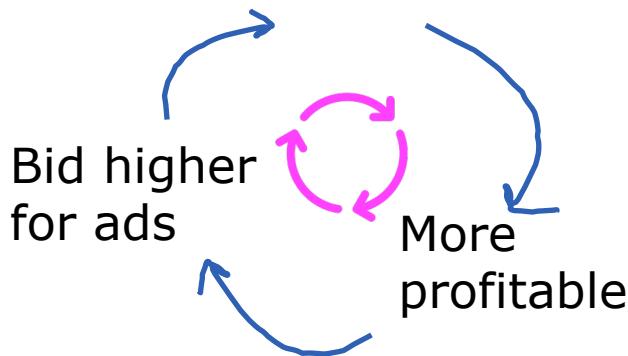
- • Movies most likely to be rated 5 stars by user
- • Products most likely to be purchased
- • Ads most likely to be clicked on *+high bid*
- • Products generating the largest profit
- • Video leading to maximum watch time



# Ethical considerations with recommender systems

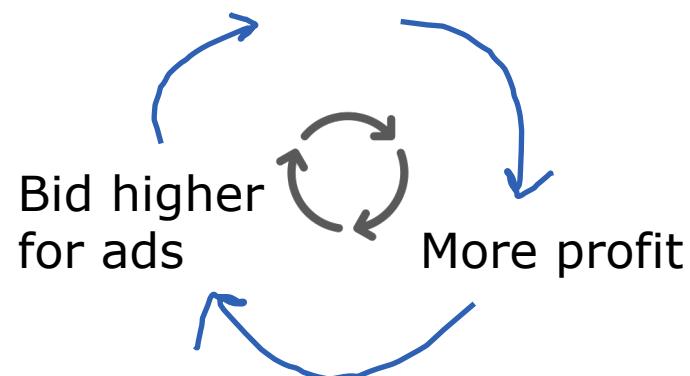
## Travel industry

Good travel experience  
to more users



## Payday loans

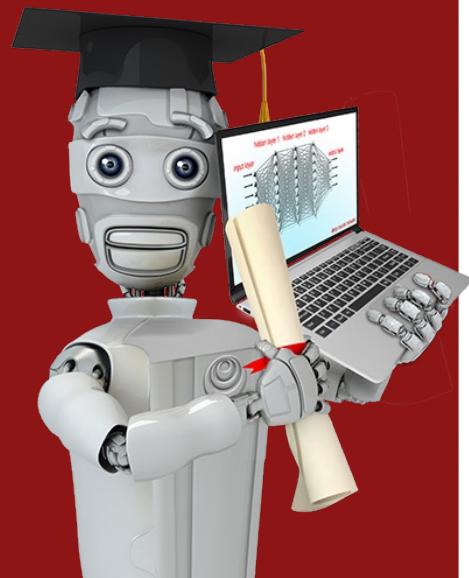
Squeeze customers more



Amelioration: Do not accept ads from exploitative businesses

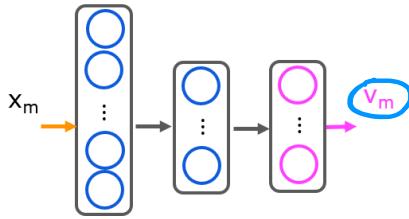
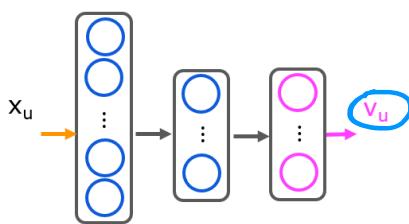
## Other problematic cases:

- • 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



# 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])

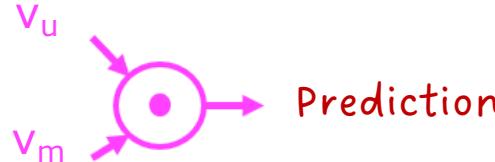
```

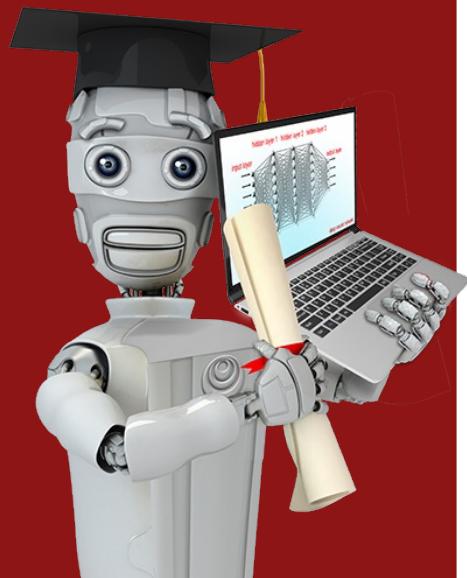
```

# 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()

```

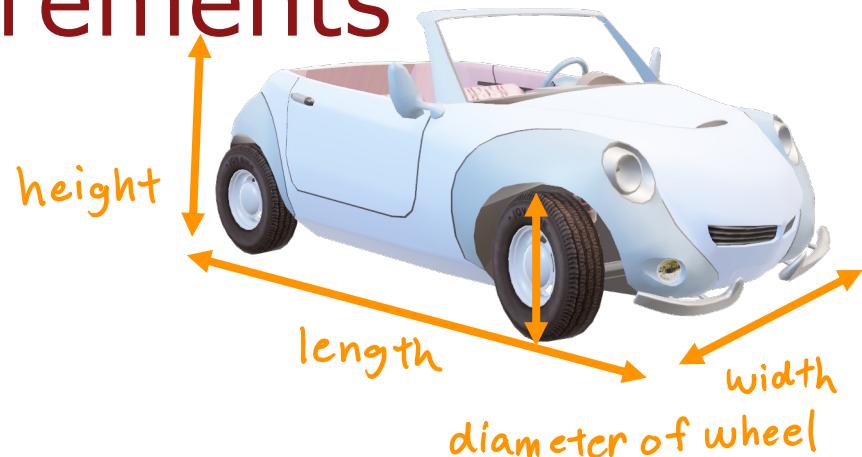




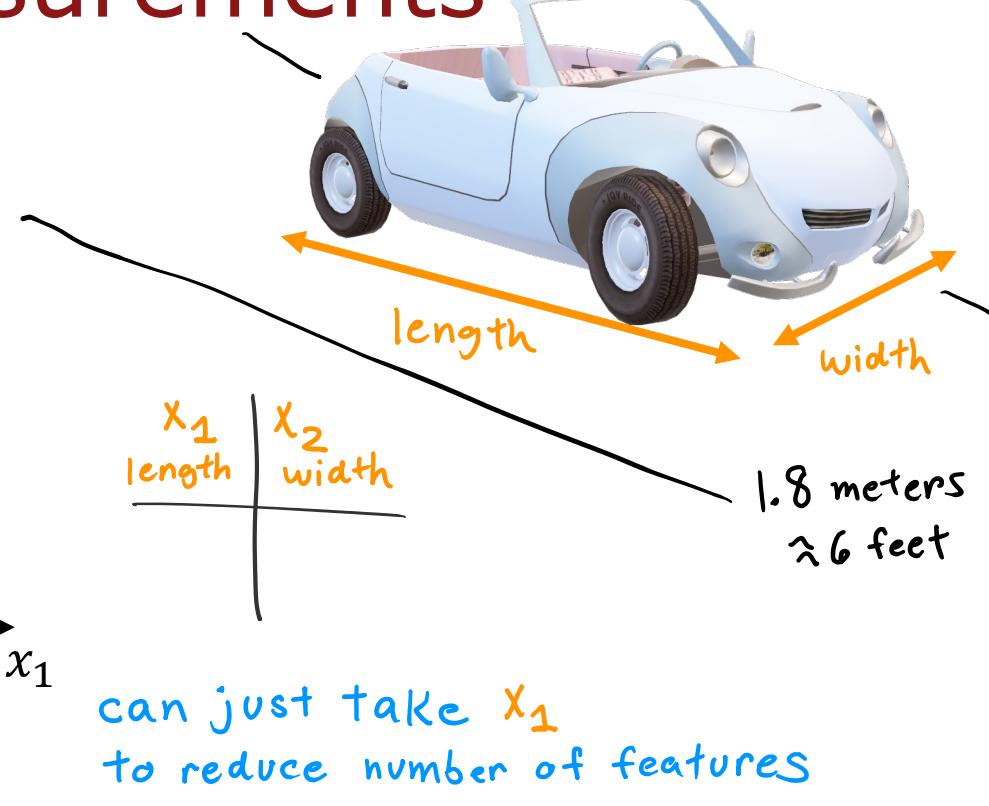
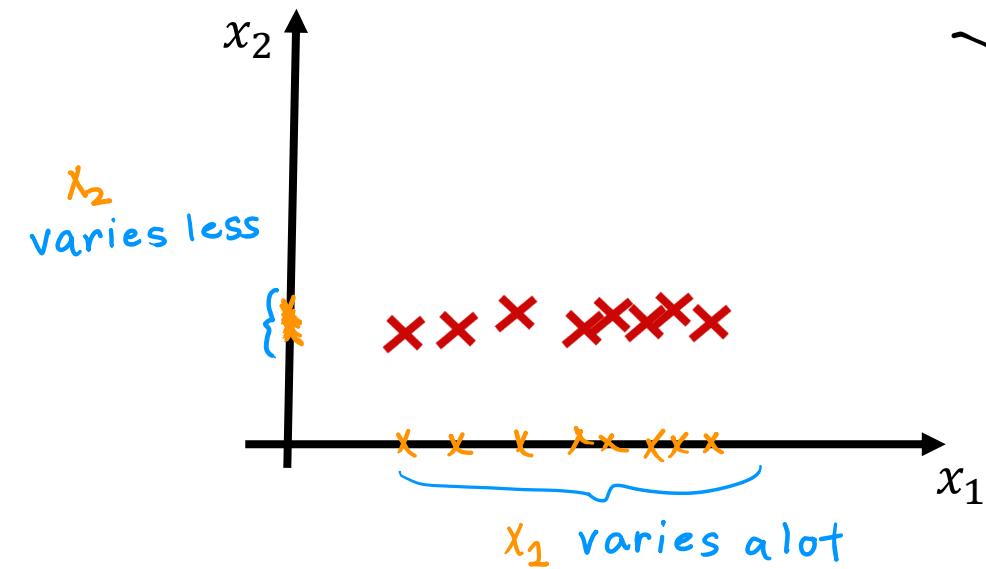
# Principal Component Analysis (Optional)

Reducing the number of features

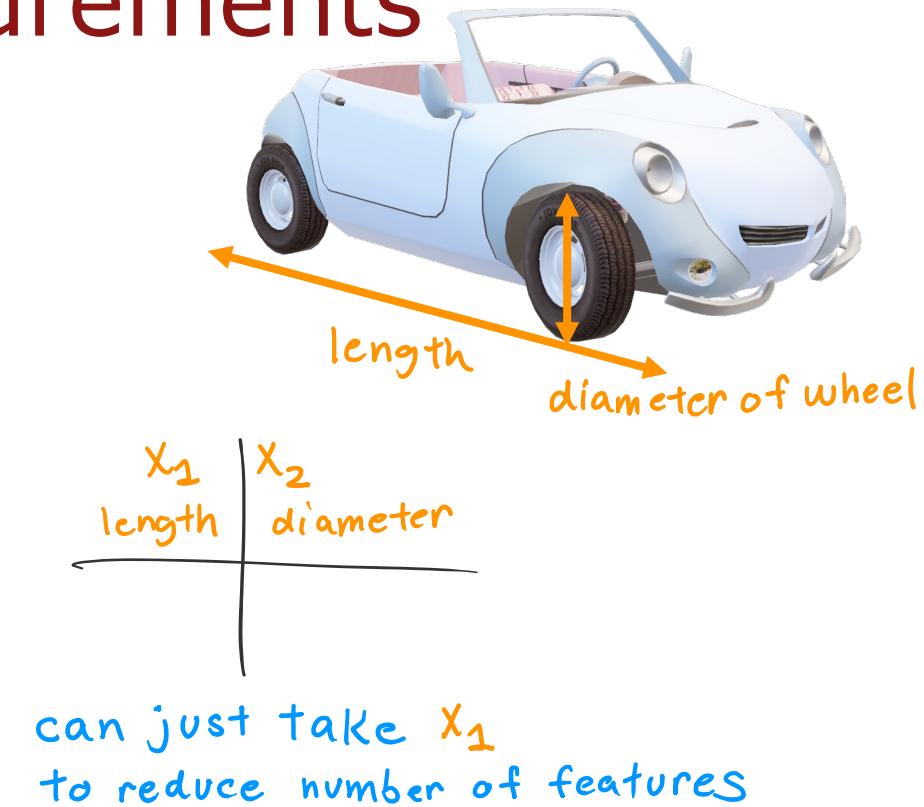
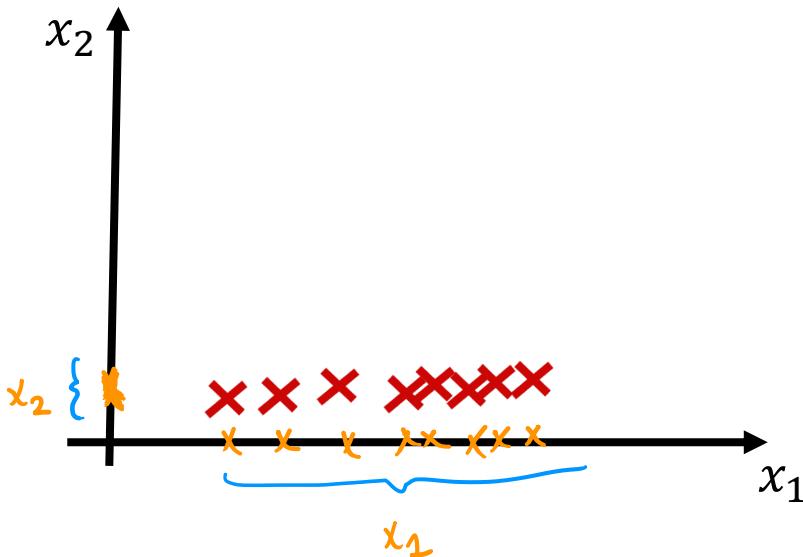
# Car measurements



# Car measurements



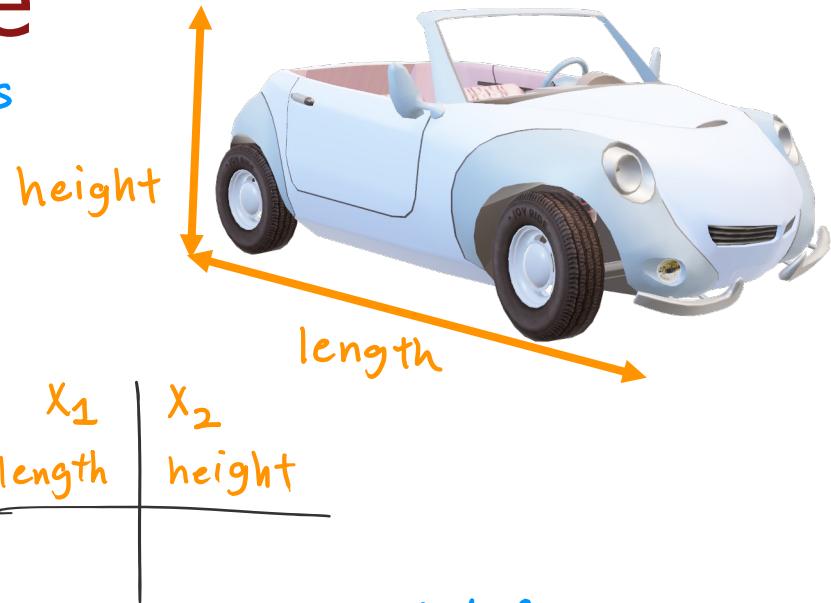
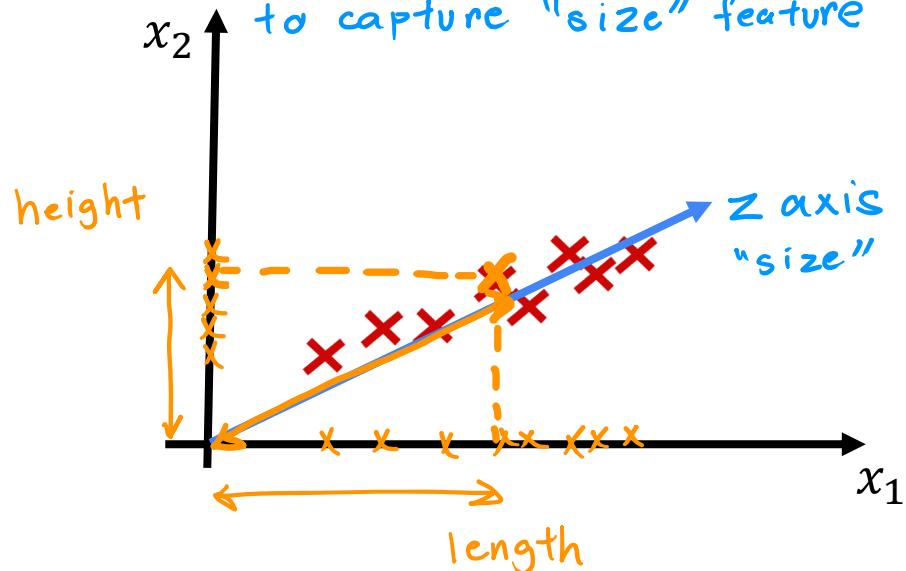
# Car measurements



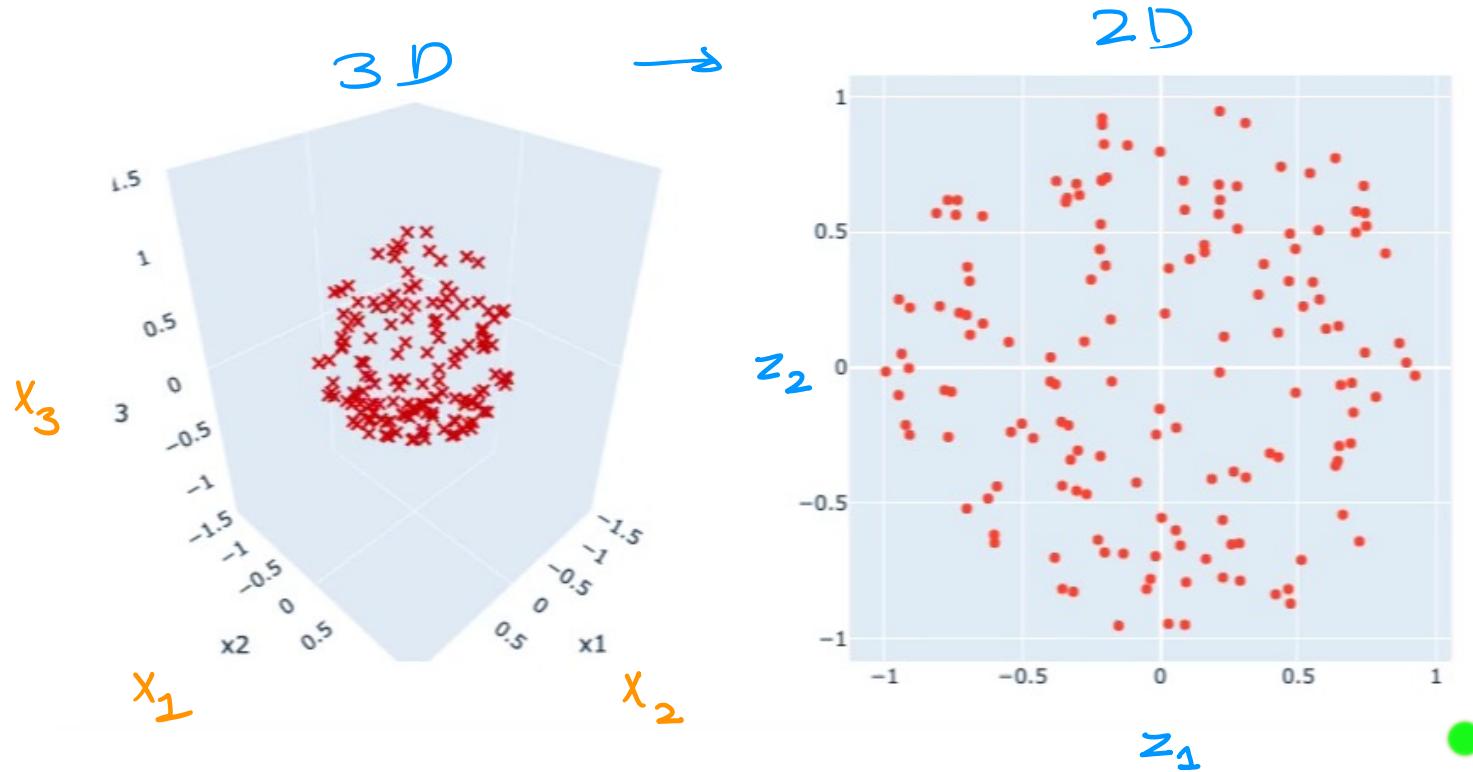
# Size

PCA: find new axis and coordinates

use fewer numbers  
to capture "size" feature



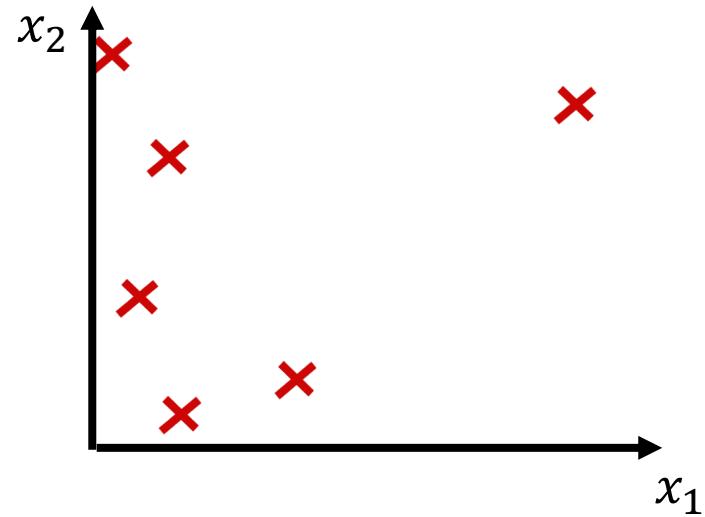
# From 3D to 2D



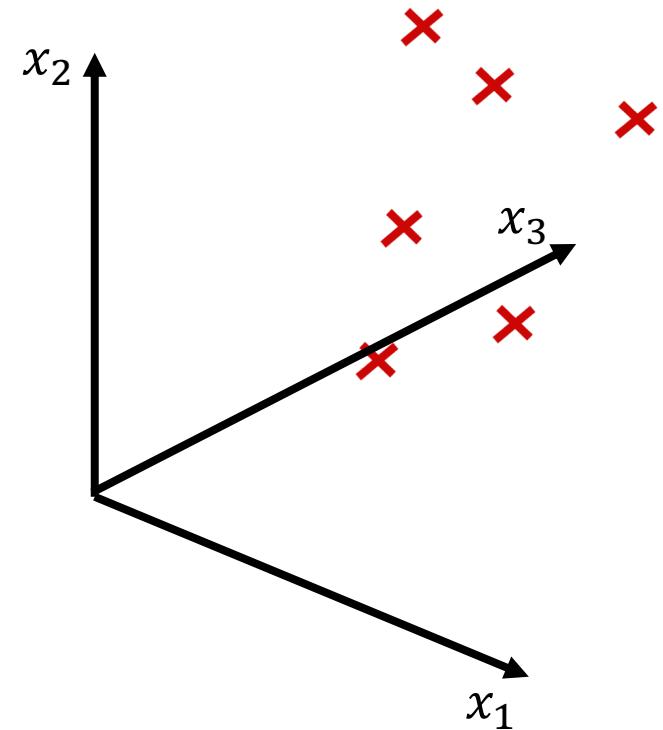
Country	GDP (trillions of US\$)
Canada	1.577
China	5.878
India	1.632
Russia	1.48
Singapore	0.223
USA	14.527
...	...



Country	GDP (trillions of US\$)	Per capita GDP (thousands of intl. \$)
Canada	1.577	39.17
China	5.878	7.54
India	1.632	3.41
Russia	1.48	19.84
Singapore	0.223	56.69
USA	14.527	46.86



Country	GDP (trillions of US\$)	Per capita GDP (thousands of intl. \$)	Human Development Index
Canada	1.577	39.17	0.908
China	5.878	7.54	0.687
India	1.632	3.41	0.547
Russia	1.48	19.84	0.755
Singapore	0.223	56.69	0.866
USA	14.527	46.86	0.91
...	...	...	...



what if 50 features?

Country	GDP (trillions of US\$)	Per capita GDP (thousands of intl. \$)	Human Development Index	Life expectancy
Canada	1.577	39.17	0.908	80.7
China	5.878	7.54	0.687	73
India	1.632	3.41	0.547	64.7
Russia	1.48	19.84	0.755	65.5
Singapore	0.223	56.69	0.866	80
USA	14.527	46.86	0.91	78.3
...	...	...	...	...

50 features

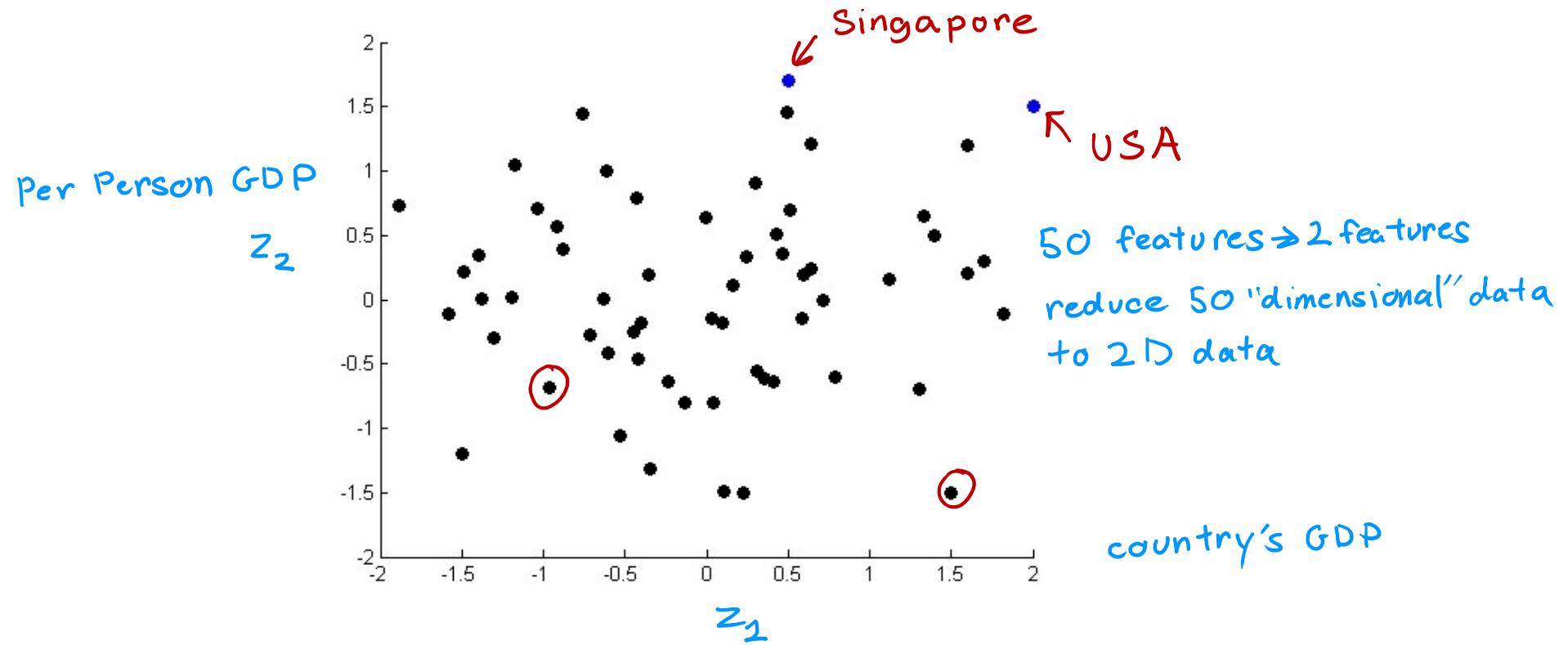
Country	$x_1$ GDP (trillions of US\$)	$x_2$ Per capita GDP (thousands of intl. \$)	$x_3$ Human Development Index	$x_4$ Life expectancy	...	Poverty Index (Gini as percentage)	Mean household income (thousands of US\$)	...
Canada	1.577	39.17	0.908	80.7		32.6	67.293	...
China	5.878	7.54	0.687	73		46.9	10.22	...
India	1.632	3.41	0.547	64.7		36.8	0.735	...
Russia	1.48	19.84	0.755	65.5		39.9	0.72	...

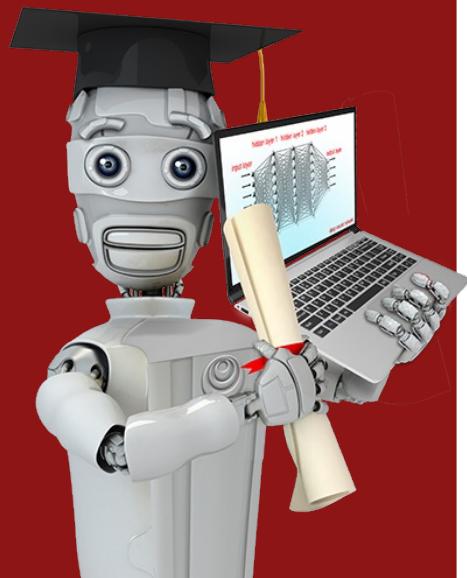
  

Country	$z_1$	$z_2$
Canada	1.6	1.2
China	1.7	0.3
India	1.6	0.2
Russia	1.4	0.5



# Data visualization





# Principal Component Analysis

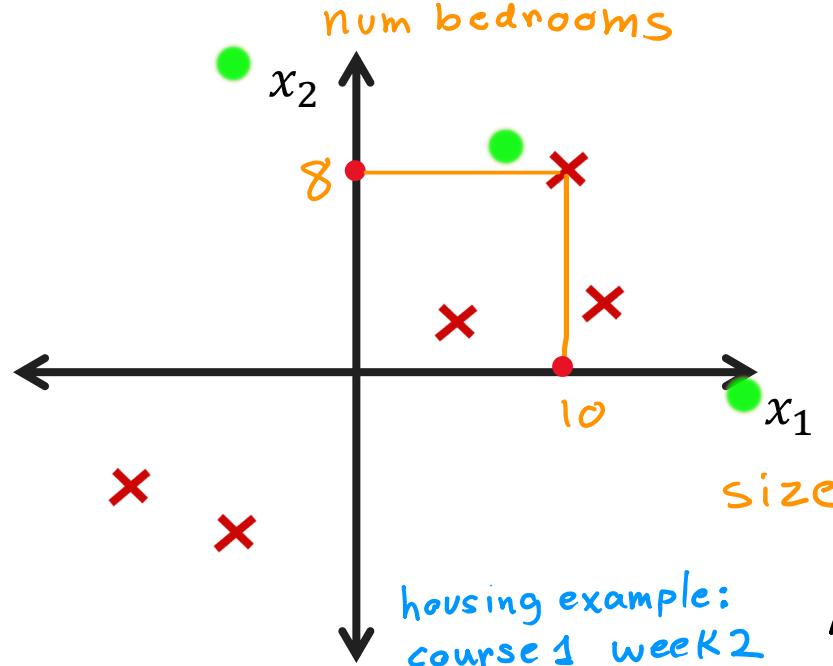
## PCA Algorithm

# PCA algorithm

coordinates

$$x_1 = 10 \quad x_2 = 8$$

Can we choose  
a different axis?



Preprocess features

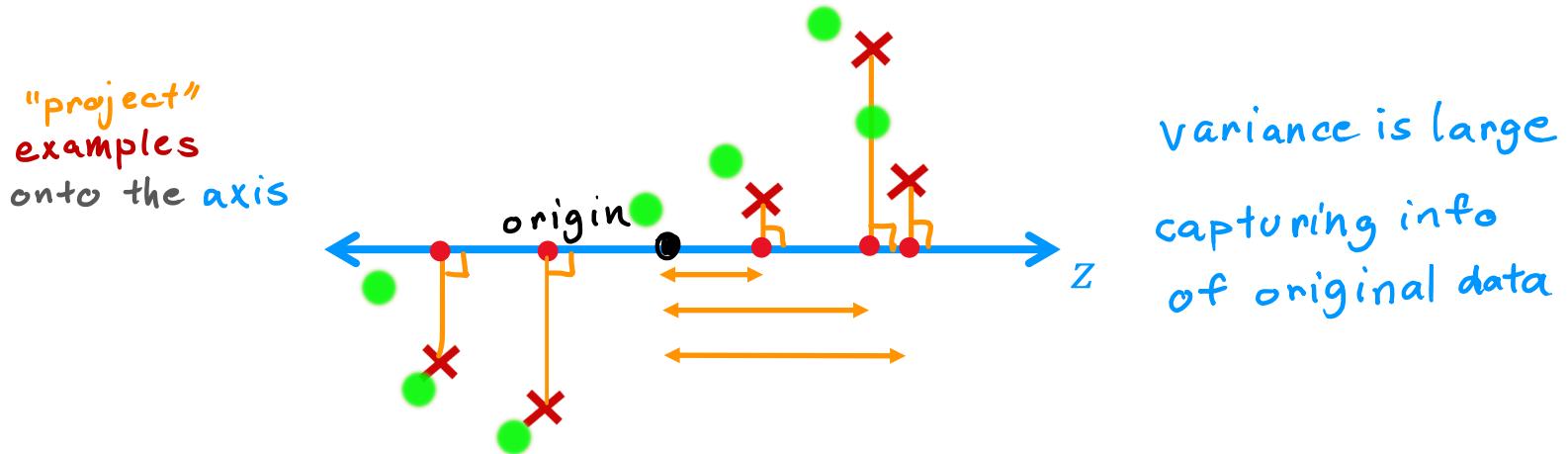
Normalized to  
have zero mean



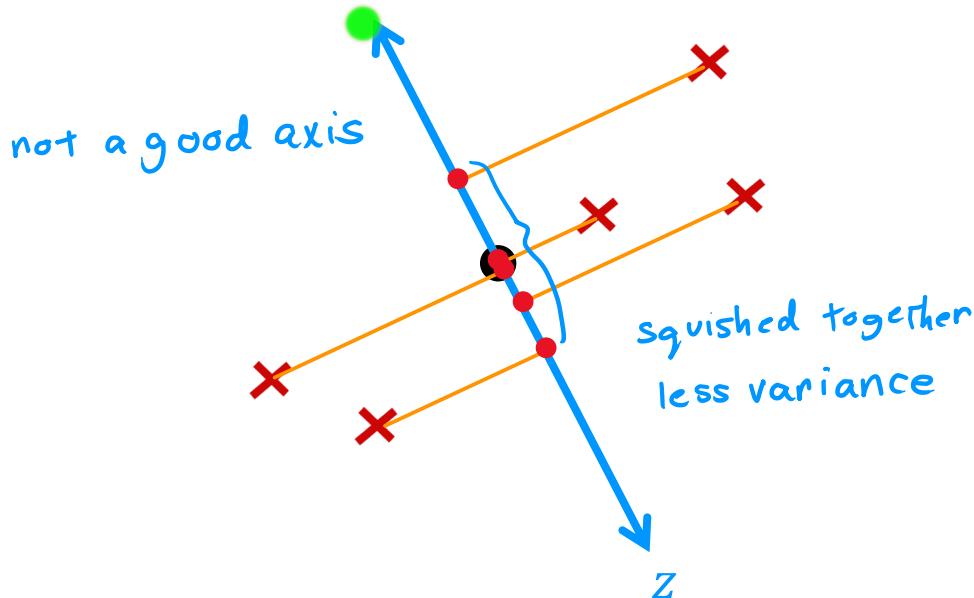
feature scaling



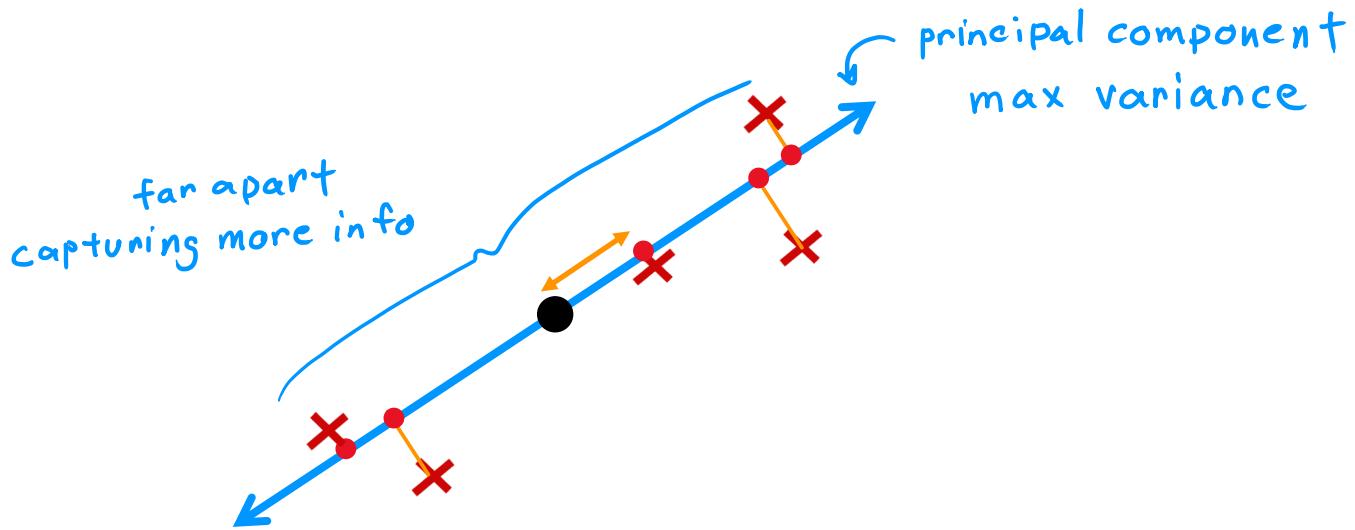
# Choose an axis



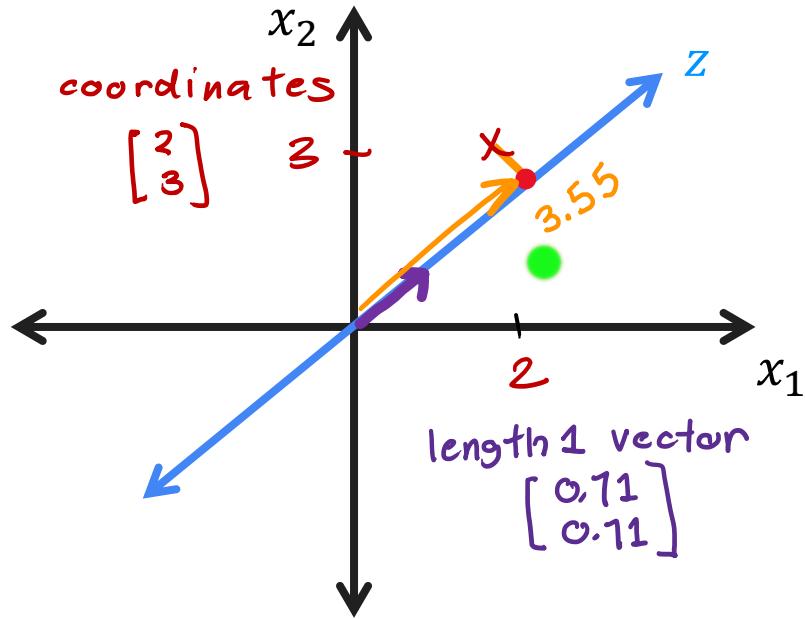
# Choose an axis



# Choose an axis



# Coordinate on the new axis

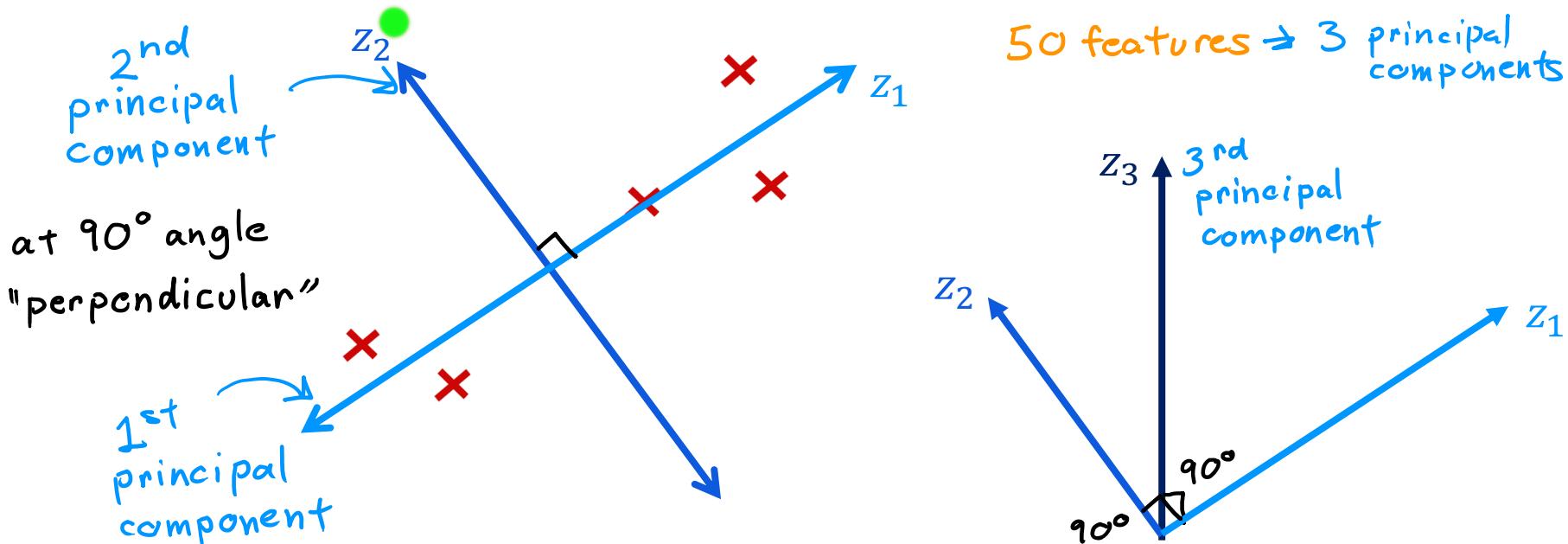


dot product

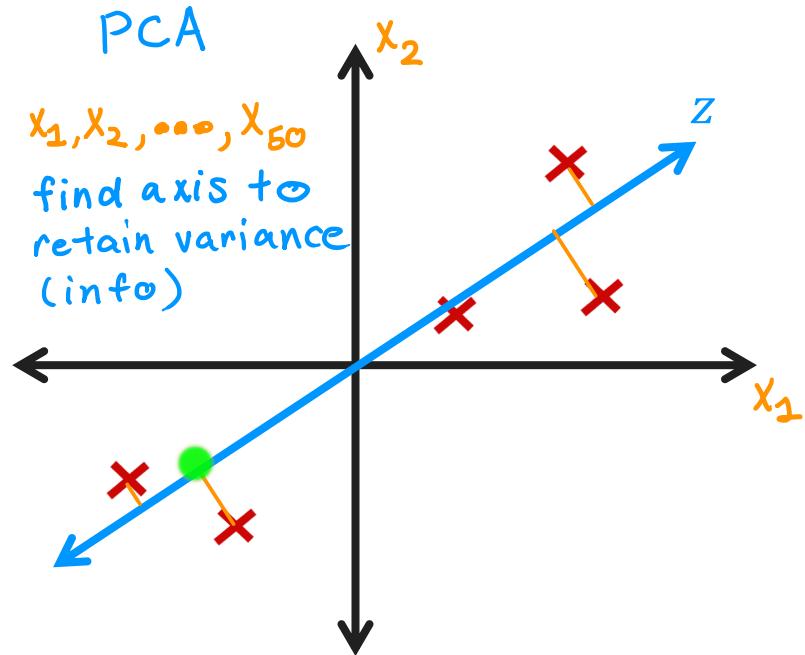
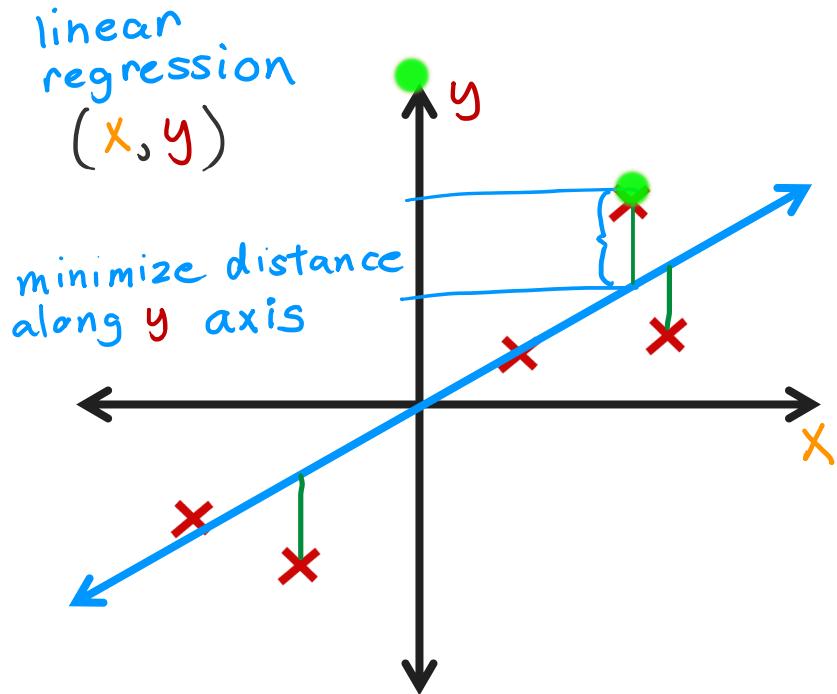
$$\begin{bmatrix} 2 \\ 3 \end{bmatrix} \cdot \begin{bmatrix} 0.71 \\ 0.71 \end{bmatrix}$$

$$2 \times 0.71 + 3 \times 0.71 = 3.55$$

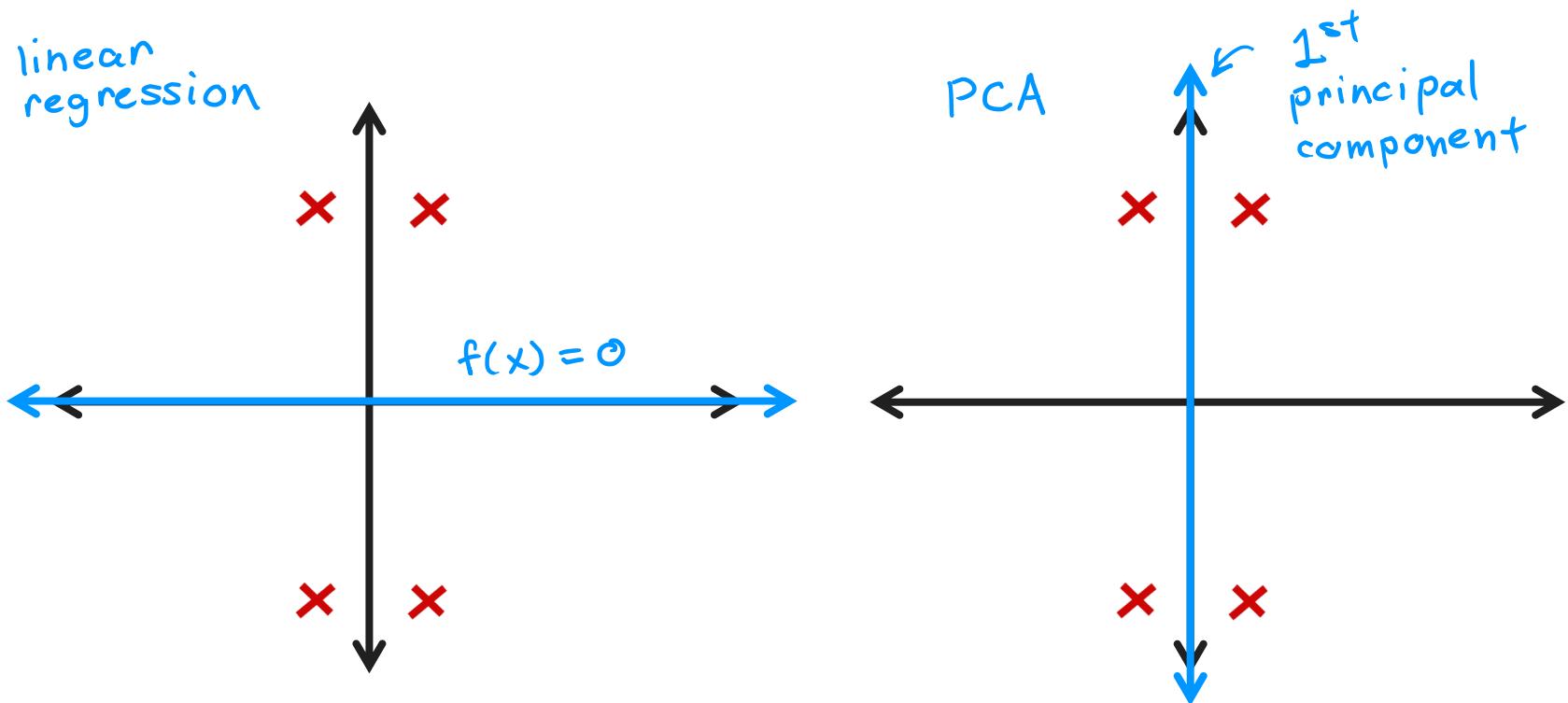
# More principal components



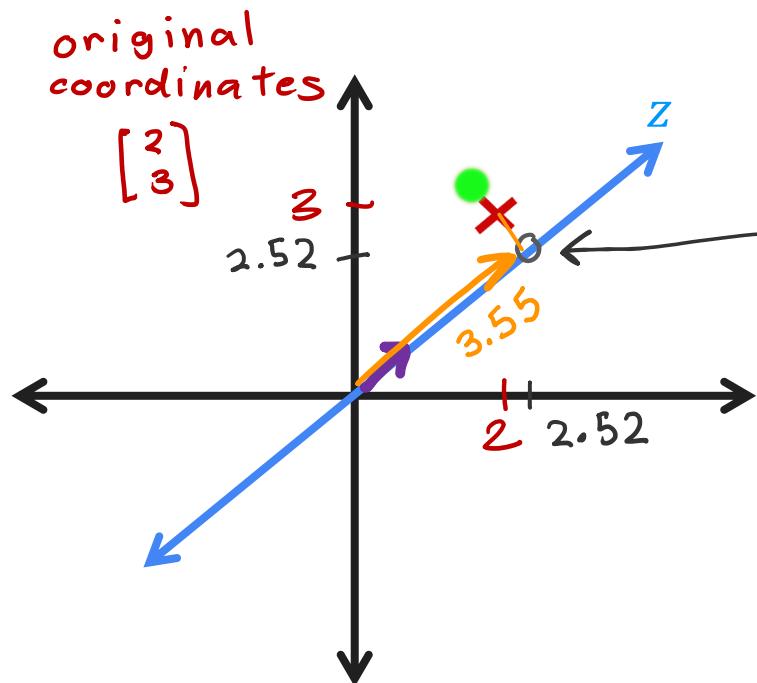
# PCA is not linear regression



# PCA is not linear regression



# Approximation to the original data

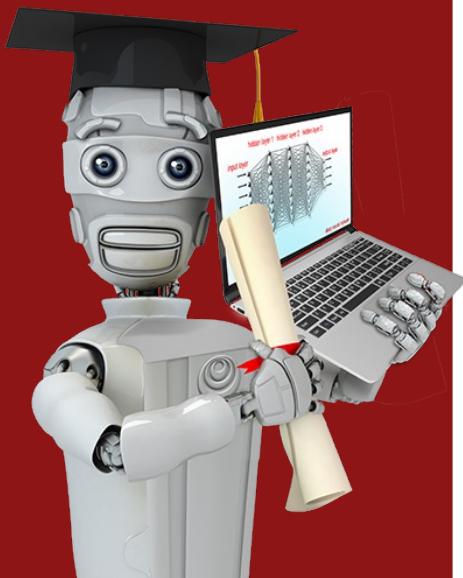


given  $z = 3.55$ ,  
find original  $(x_1, x_2)$  (approximately)  
"reconstruction"

$$3.55 \times \begin{bmatrix} 0.71 \\ 0.71 \end{bmatrix} = \begin{bmatrix} 2.52 \\ 2.52 \end{bmatrix}$$



Stanford  
ONLINE



PCA

PCA in Code

# PCA in scikit-learn

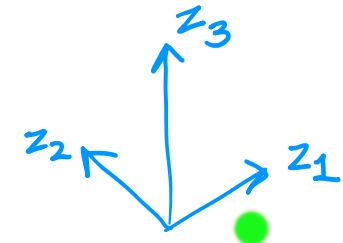


Optional pre-processing: Perform feature scaling

*for visualization*

1. "fit" the data to obtain 2 (or 3) new axes (principal components)

*fit includes mean normalization*



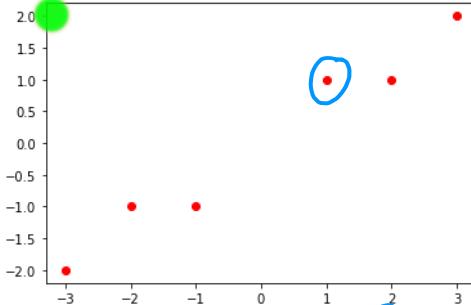
2. Optionally examine how much variance is explained by each principal component.  
*info*

`explained_variance_ratio`

3. Transform (project) the data onto the new axes

`transform`



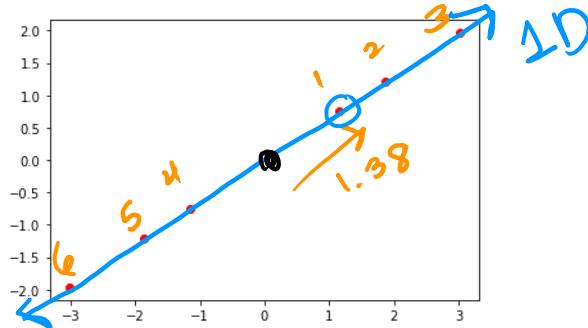


# Example

```
X = np.array([[1, 1], [2, 1], [3, 2],
[-1, -1], [-2, -1], [-3, -2]])
```

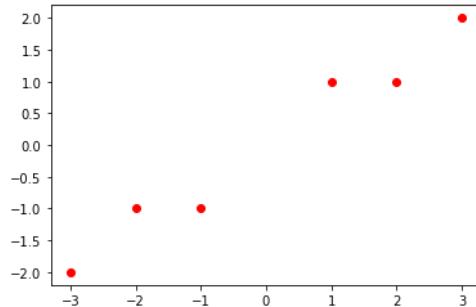
2D

```
pca_1 = PCA(n_components=1)
pca_1.fit(X)
pca_1.explained_variance_ratio_ 0.992
X_trans_1 = pca_1.transform(X)
X_reduced_1 = pca_1.inverse_transform(X_trans_1)
```



0.992

array([
1 [ 1.38340578],  
2 [ 2.22189802],  
3 [ 3.6053038 ],  
4 [-1.38340578],  
5 [-2.22189802],  
6 [-3.6053038 ]])



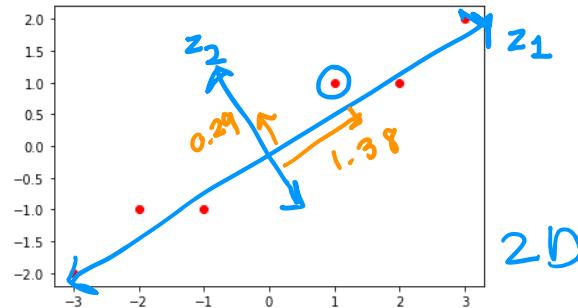
# Example

```
X = np.array([[1, 1], [2, 1], [3, 2],
[-1, -1], [-2, -1], [-3, -2]])
```

2D

```
pca_2 = PCA(n_components=2)
pca_2.fit(X)
pca_2.explained_variance_ratio_
X_trans_2 = pca.transform(X)
X_reduced_2 = pca.inverse_transform(X_trans_2)
```

$$\begin{matrix} z_1 & z_2 \\ 0.992 & 0.008 \end{matrix}$$



2D

$z_1$

$z_2$

```
array([
→ [ 1.38340578,  0.2935787 ],
[ 2.22189802, -0.25133484],
[ 3.6053038 ,  0.04224385],
[-1.38340578, -0.2935787 ],
[-2.22189802,  0.25133484],
[-3.6053038 , -0.04224385]])
```

# Applications of PCA

❖ Visualization *reduce to 2 or 3 features*

Less frequently used for:

- Data compression  
(to reduce storage or transmission costs)  $50 \rightarrow 10$
- Speeding up training of a supervised learning model

$$n = 1000 \rightarrow 100$$