WEEK 2

DeepLearning.AI

Stanford
ONLINE

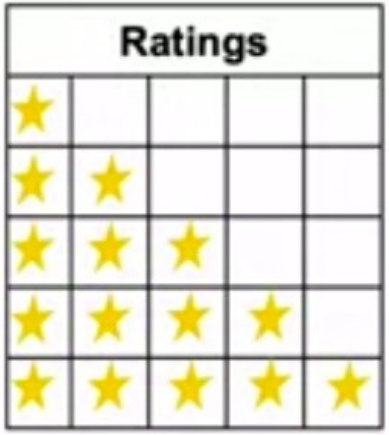
Recommender System

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	?



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 $r(i,j)=1$)

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

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

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

$$x^{(1)} = \begin{bmatrix} 0.9 \\ 0 \end{bmatrix} \qquad x^{(3)} = \begin{bmatrix} 0.99 \\ 0 \end{bmatrix}$$

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

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

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

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

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

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 :

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

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

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

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

Cost function

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

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

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

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

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

Collaborative filtering

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

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

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

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 {

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

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

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

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

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

}

parameters w, b, x x is also a parameter

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

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

Cost function for binary application

Previous cost function:

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

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

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

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

Congratulations! You passed!

Grade received 100% **Latest Submission Grade** 100% **To pass** 80% or higher

Go to next item

1. You have the following table of movie ratings: **1 / 1 point**

Movie	Elissa	Zach	Barry	Terry
Football Forever	5	4	3	?
Pies, Pies, Pies	1	?	5	4
Linear Algebra Live	4	5	?	1

Refer to the table above for question 1 and 2. Assume numbering starts at 1 for this quiz, so the rating for Football Forever by Elissa is at (1,1)

What is the value of n_u

4

Correct
This is the number of users. n_m is the number of movies/items and is 3 in this table.

Linear Algebra Live	4	5	?	1

Refer to the table above for question 1 and 2. Assume numbering starts at 1 for this quiz, so the rating for Football Forever by Elissa is at (1,1)

What is the value of n_u

4

Correct
This is the number of users. n_m is the number of movies/items and is 3 in this table.

2. What is the value of $r(2,2)$ **1 / 1 point**

0

Correct
$r(i,j)$ is a 1 if the movie has a rating and 0 if it does not. In the table above, a question mark indicates there is no rating.

3. In which of the following situations will a collaborative filtering system be the most appropriate learning algorithm (compared to linear or logistic regression)? **1 / 1 point**

- ○ You subscribe to an online video streaming service, and are not satisfied with their movie suggestions. You download all your viewing for the last 10 years and rate each item. You assign each item a genre. Using your ratings and genre assignment, you learn to predict how you will rate new movies based on the genre.
- ◉ You run an online bookstore and collect the ratings of many users. You want to use this to identify what books are "similar" to each other (i.e., if a user likes a certain book, what are other books that they might also like?)
- ○ You're an artist and hand-paint portraits for your clients. Each client gets a different portrait (of themselves) and gives you 1-5 star rating feedback, and each client purchases at most 1 portrait. You'd like to predict what rating your next customer will give you.
- ○ You manage an online bookstore and you have the book ratings from many users. You want to learn to predict the expected sales volume (number of books sold) as a function of the average rating of a book.

Correct
You can find "similar" books by learning feature values using collaborative filtering.

4. For recommender systems with binary labels y, which of these are reasonable ways for defining when y should be 1 for a given user j and item i? (Check all that apply.) **1 / 1 point**

- ☐ y is 1 if user j has not yet been shown item i by the recommendation engine

Correct
You can find "similar" books by learning feature values using collaborative filtering.

4. For recommender systems with binary labels y, which of these are reasonable ways for defining when y should be 1 for a given user j and item i? (Check all that apply.) **1 / 1 point**

- [ ] y is 1 if user j has not yet been shown item i by the recommendation engine
- [x] y is 1 if user j fav/likes/clicks on item i (after being shown the item)

Correct
fav/likes/clicks on an item shows a user's interest in that item. It also shows that an item is interesting to a user.

- [x] y is 1 if user j purchases item i (after being shown the item)

Correct
Purchasing an item shows a user's preference for that item. It also shows that an item is preferred by a user.

- [ ] y is 1 if user j has been shown item i by the recommendation engine

DeepLearning.AI

Stanford
ONLINE

Recommender Systems implementation

Mean normalization

Users who have not rated any movies

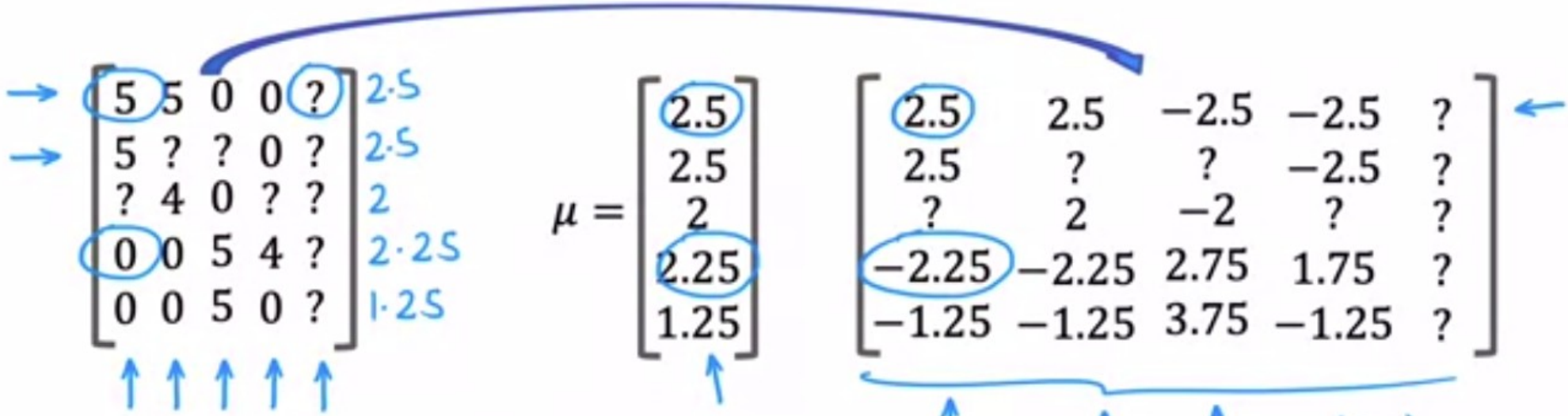
Movie	Alice(1)	Bob (2)	Carol (3)	Dave (4)	Eve (5)	
Love at last	5	5	0	0	?	
Romance forever	5	?	?	0	?	
Cute puppies of love	?	4	0	?	?	
Nonstop car chases	0	0	5	4	?	
Swords vs. karate	0	0	5	?	?	

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

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

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

Mean Normalization



For user j, on movie i predict:

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

User 5 (Eve):

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

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

DeepLearning.AI

Stanford
ONLINE

Recommender Systems implementational detail

TensorFlow implementation

Derivatives in ML

Gradient descent algorithm

Repeat until convergence

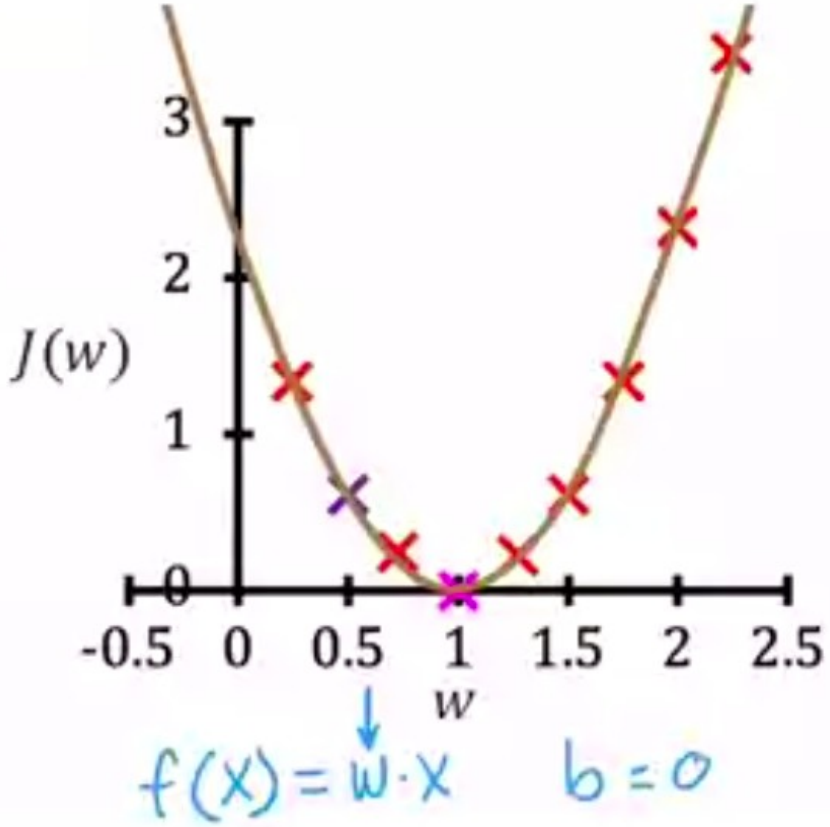
$$w = w - \alpha \frac{d}{dw} J(w,b)$$

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

Learning rate

Derivative

$b = 0$



Custom Training Loop

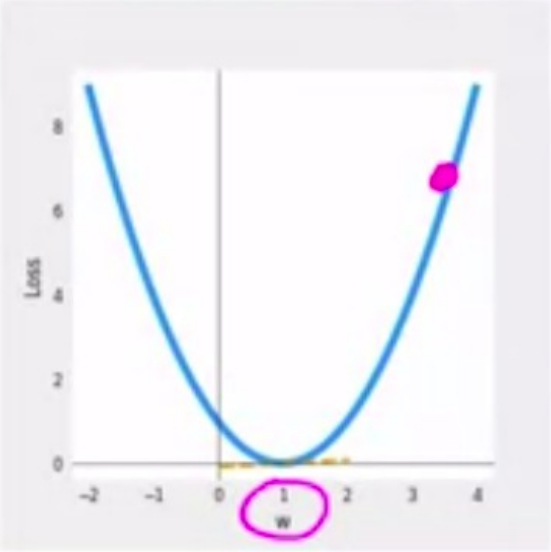
$$J = (\underbrace{wx}_{f(x)} - \underbrace{1}_{y})^2$$

Gradient descent algorithm

Repeat until convergence

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

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)
```

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

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

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

$$X = X - \alpha \frac{d}{dx} J(w,b,x)$$

```
# 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

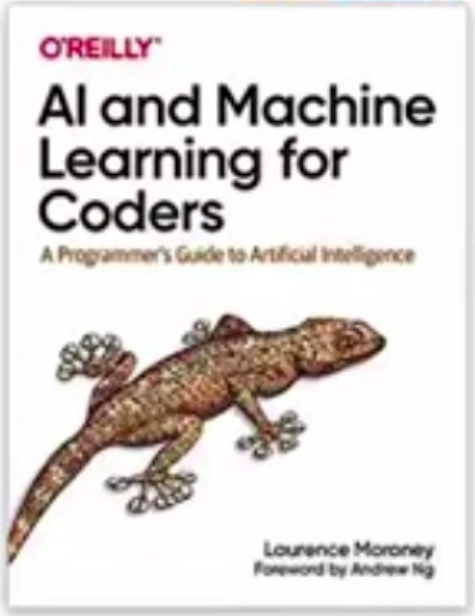
DeepLearning.AI

Stanford
ONLINE

Collaborative Filtering

Finding related items

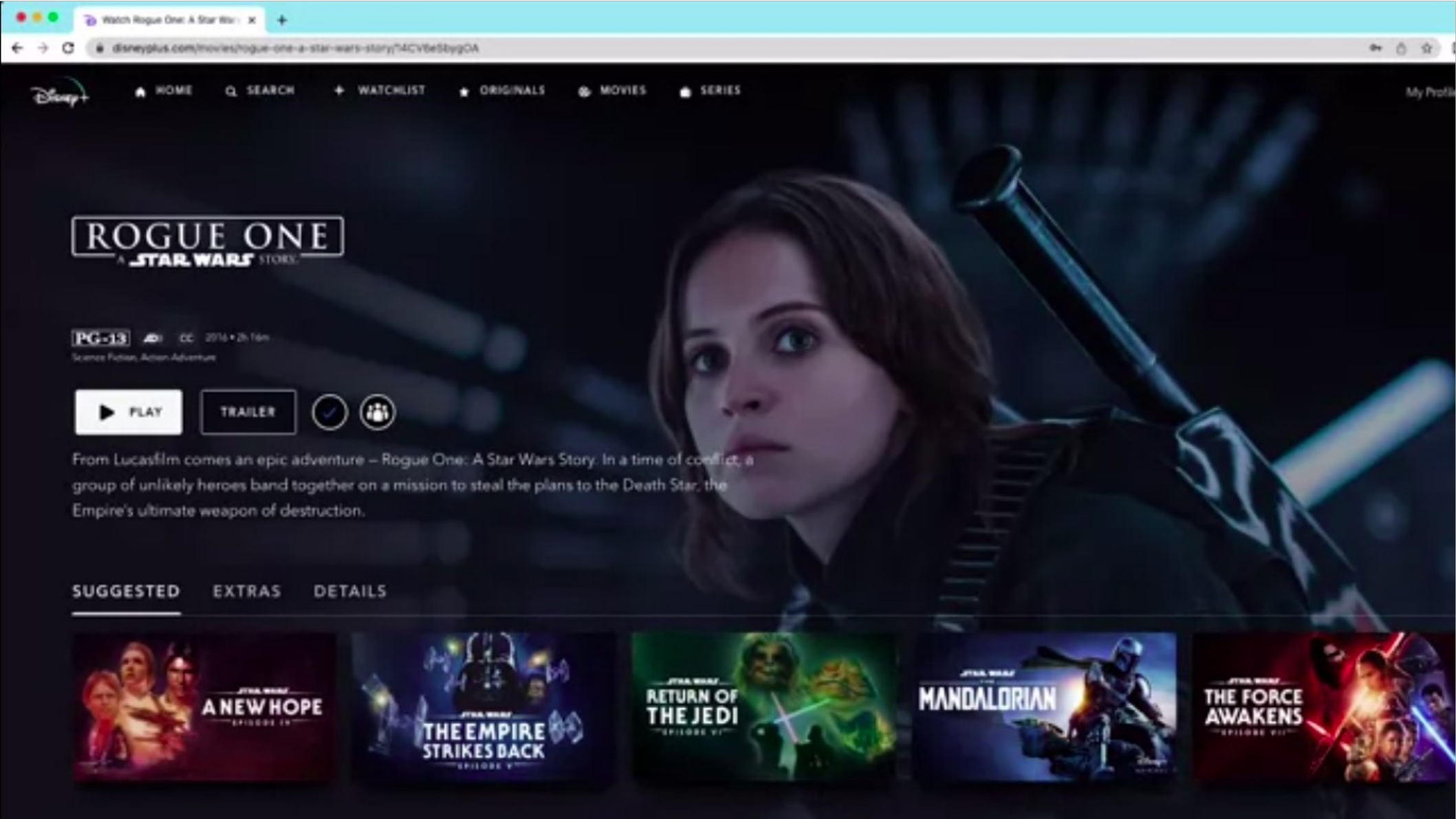
Look inside



AI and Machine Learning for Coders: A Programmer's Guide to Artificial Intelligence 1st Edition

by Laurence Moroney (Author)

4.4 out of 5 stars 113 ratings


HOME
SEARCH
WATCHLIST
ORIGINALS
MOVIES
SERIES
ROGUE ONE
A STAR WARS STORY
PG-13
Science Fiction, Action Adventure
PLAY
TRAILER
From Lucasfilm comes an epic adventure – Rogue One: A Star Wars Story. In a time of conflict, a group of unlikely heroes band together on a mission to steal the plans to the Death Star, the Empire's ultimate weapon of destruction.
SUGGESTED
EXTRAS
DETAILS
A NEW HOPE
THE EMPIRE STRIKES BACK
RETURN OF THE JEDI
MANDALORIAN
THE FORCE AWAKENS

Finding related items

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

romance

action

To find other items related to it,
find item k with $x^{(k)}$ similar to $x^{(i)}$

x_1, x_2, x_3

n

i.e. with smallest distance

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

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

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

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 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

$r(i,j) = 1$ 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

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

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

computed from $\boldsymbol{x}_u^{(j)}$

computed from $\boldsymbol{x}_m^{(i)}$

$x_u^{(j)}$

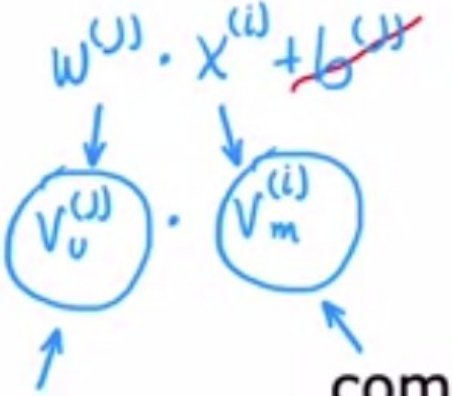
user's preferences $v_u^{(j)} = \begin{bmatrix} 4.9 \\ 0.1 \\ \vdots \\ 3.0 \end{bmatrix}$ likes / likes

$x_m^{(i)}$

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

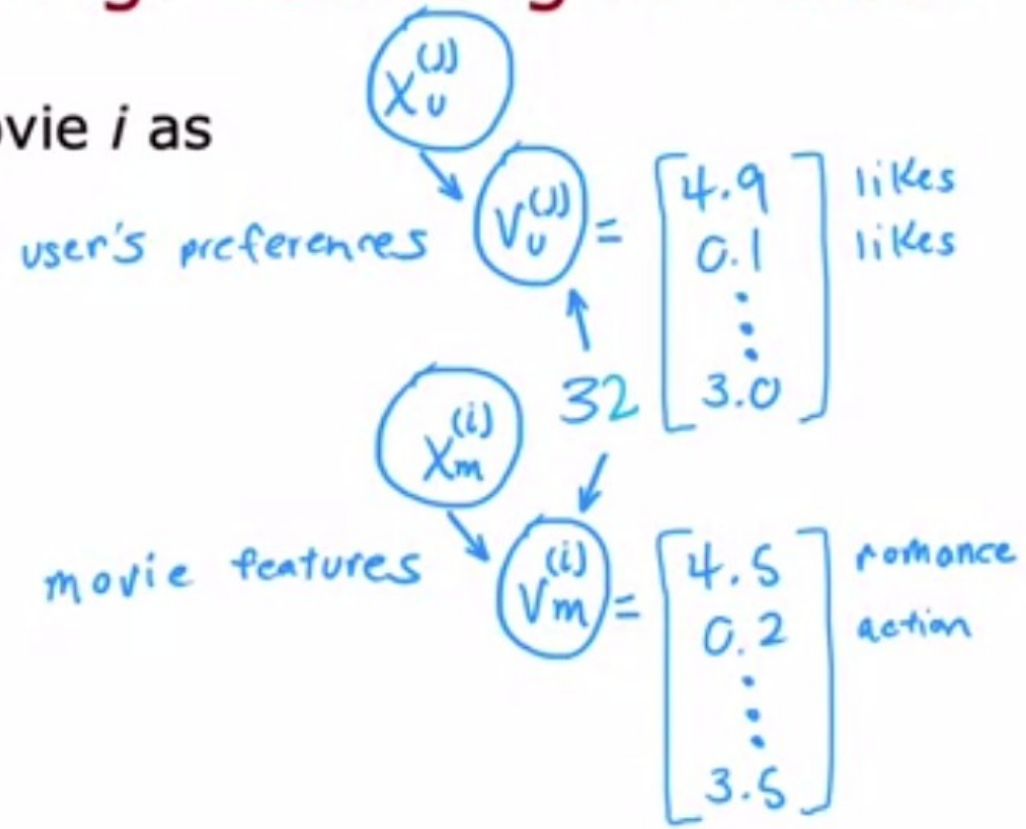
Content-based filtering: Learning to match

Predict rating of user j on movie i as



computed from $\boldsymbol{x}_{\boldsymbol{u}}^{(\boldsymbol{j})}$

computed from $\boldsymbol{x}_{\boldsymbol{m}}^{(\boldsymbol{i})}$



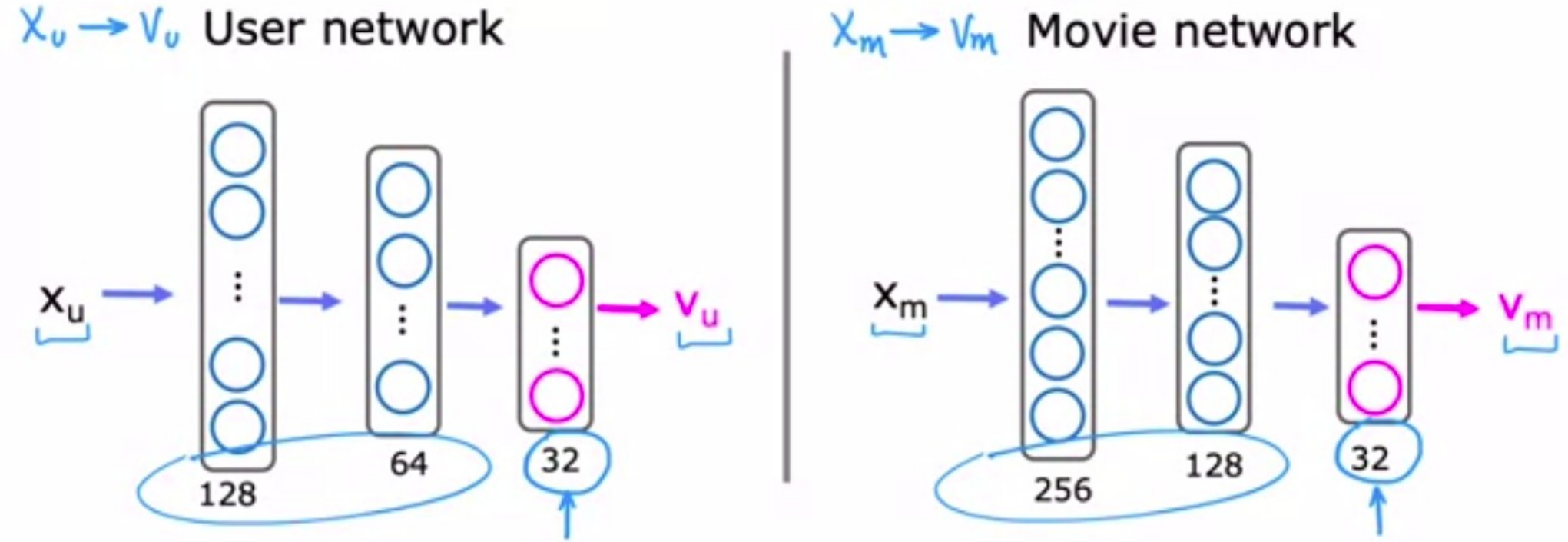
DeepLearning.AI

Stanford
ONLINE

Content-based Filtering

Deep learning for content-based filtering

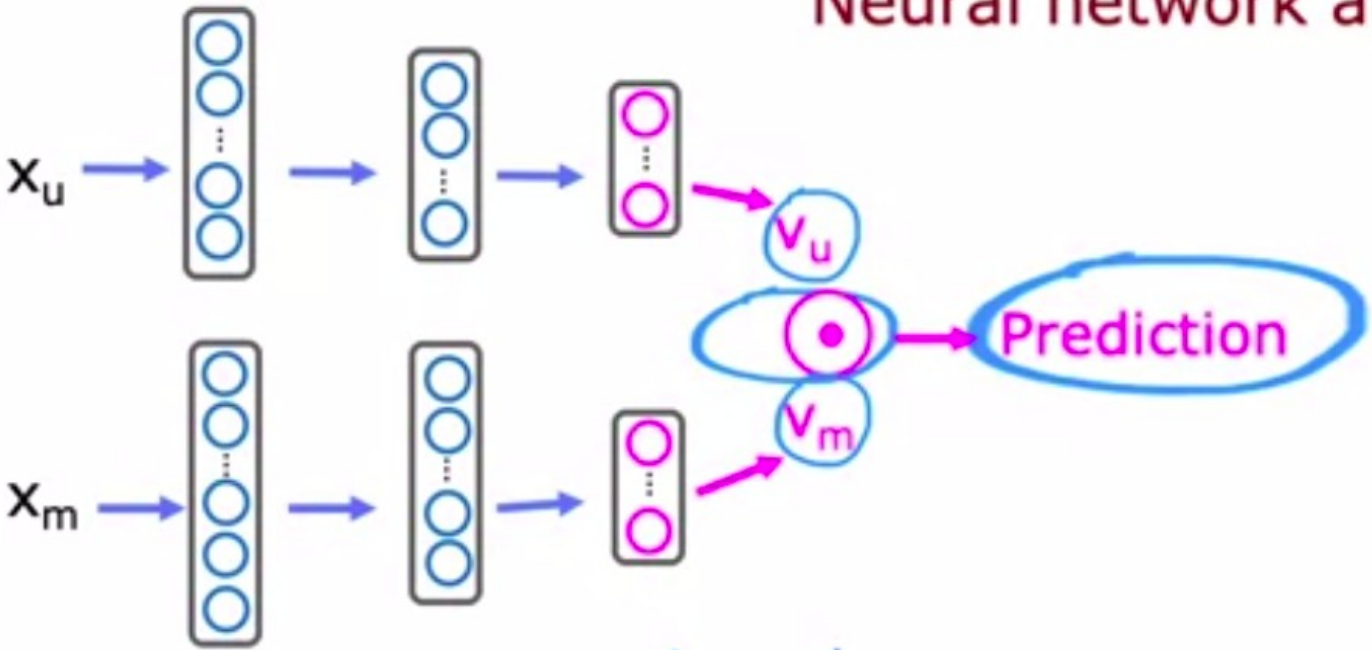
Neural network architecture



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

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

Neural network architecture



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

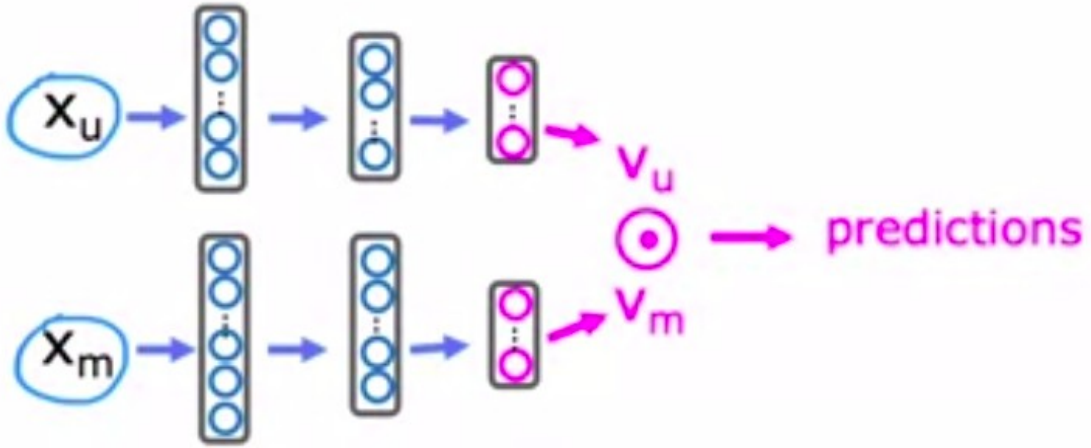
Stanford
ONLINE

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

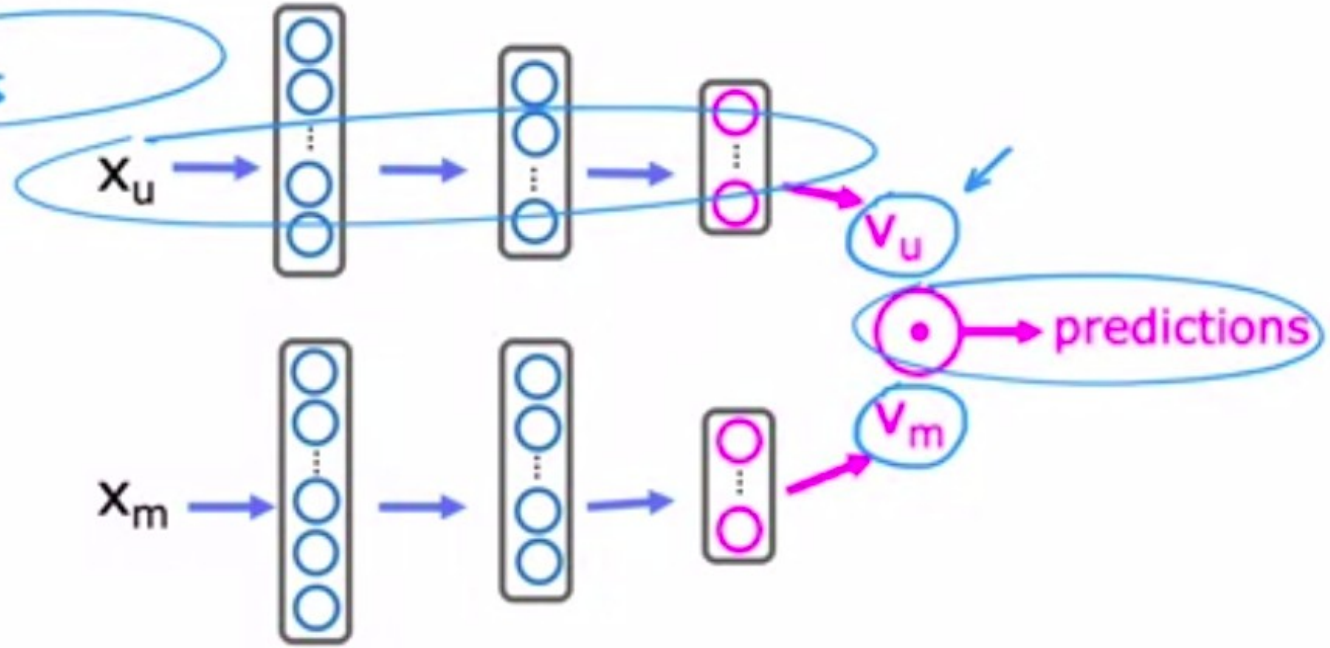
  3) Top 20 movies in the country

- Combine retrieved items into list, removing duplicates and items already watched/purchased

Two steps: Retrieval & ranking

Ranking:

- 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?

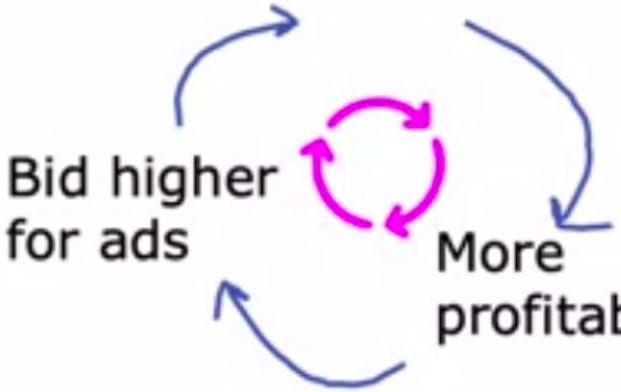
Recommend:

- 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

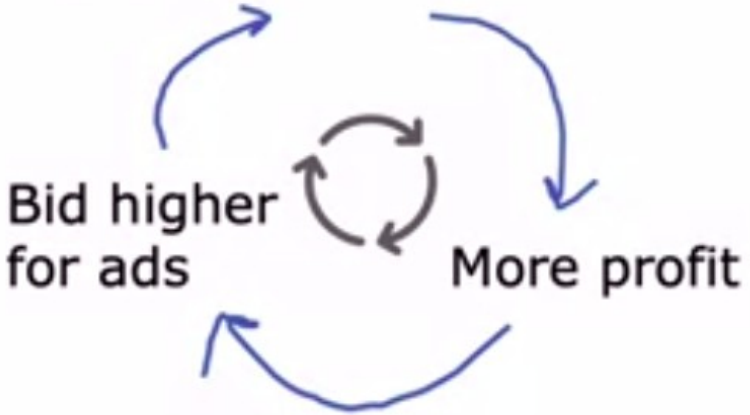
Travel industry

Good travel experience to more users



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?

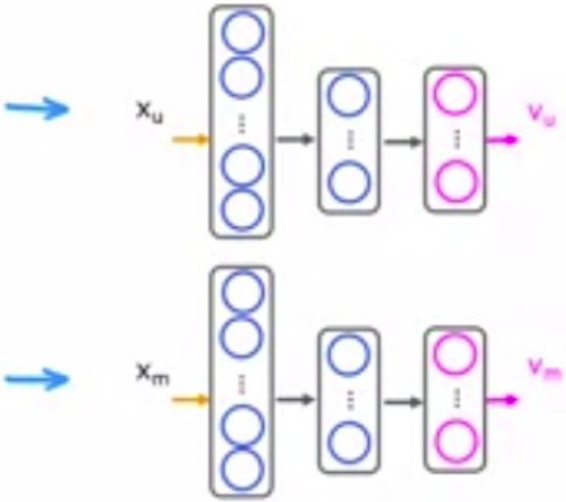
  → Amelioration : Be transparent with users

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 )
])
```

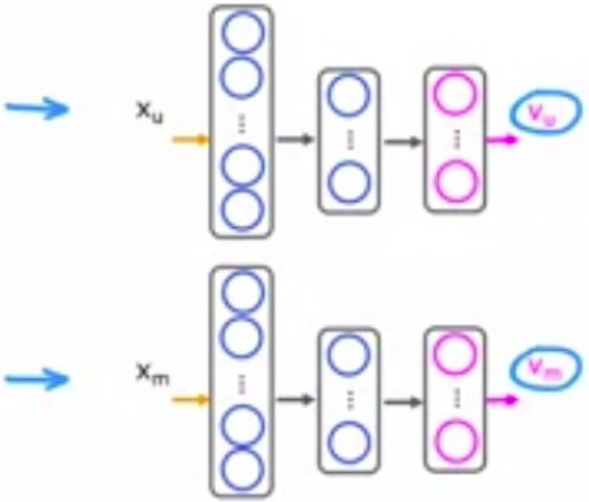
```
# create the user input and point to the base network
input_user = tf.keras.layers.Input(shape=(num_user_features))
vu = user_NN(input_user)
vu = tf.linalg.l2_normalize(vu, axis=1)

# 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()
```




```
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 )
])
```

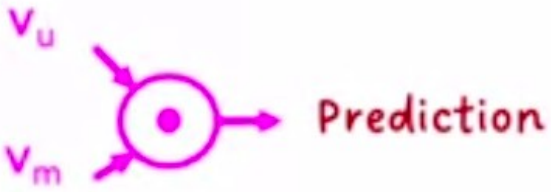
```
# create the user input and point to the base network
input_user = tf.keras.layers.Input(shape=(num_user_features))
vu = user_NN(input_user)
vu = tf.linalg.l2_normalize(vu, axis=1)

# 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()
```



Congratulations! You passed!

Grade received 100% | Latest Submission Grade 100% | To pass 80% or higher

Go to next item

1. Vector x_u and vector x_m must be of the same dimension, where x_u is the input features vector for a user (age, gender, etc.) x_m is the input features vector for a movie (year, genre, etc.) True or false? **1 / 1 point**

- ○ True
- ◉ False

Correct
These vectors can be different dimensions.

2. If we find that two movies, i and k, have vectors $v_m^{(i)}$ and $v_m^{(k)}$ that are similar to each other (i.e., $||v_m^{(i)} - v_m^{(k)}||$ is small), then which of the following is likely to be true? Pick the best answer. **1 / 1 point**

- ◉ The two movies are similar to each other and will be liked by similar users.
- ○ A user that has watched one of these two movies has probably watched the other as well.
- ○ The two movies are very dissimilar.
- ○ We should recommend to users one of these two movies, but not both.

Correct

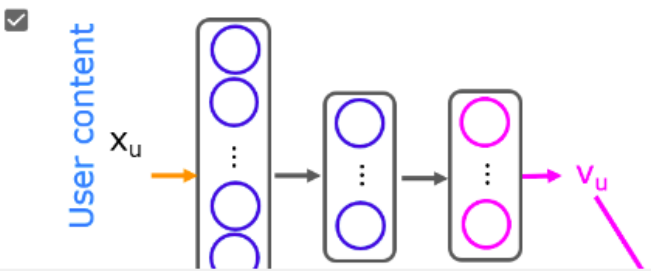
Correct
These vectors can be different dimensions.

2. If we find that two movies, i and k, have vectors $v_m^{(i)}$ and $v_m^{(k)}$ that are similar to each other (i.e., $||v_m^{(i)} - v_m^{(k)}||$ is small), then which of the following is likely to be true? Pick the best answer. **1 / 1 point**

- (●) The two movies are similar to each other and will be liked by similar users.
- ( ) A user that has watched one of these two movies has probably watched the other as well.
- ( ) The two movies are very dissimilar.
- ( ) We should recommend to users one of these two movies, but not both.

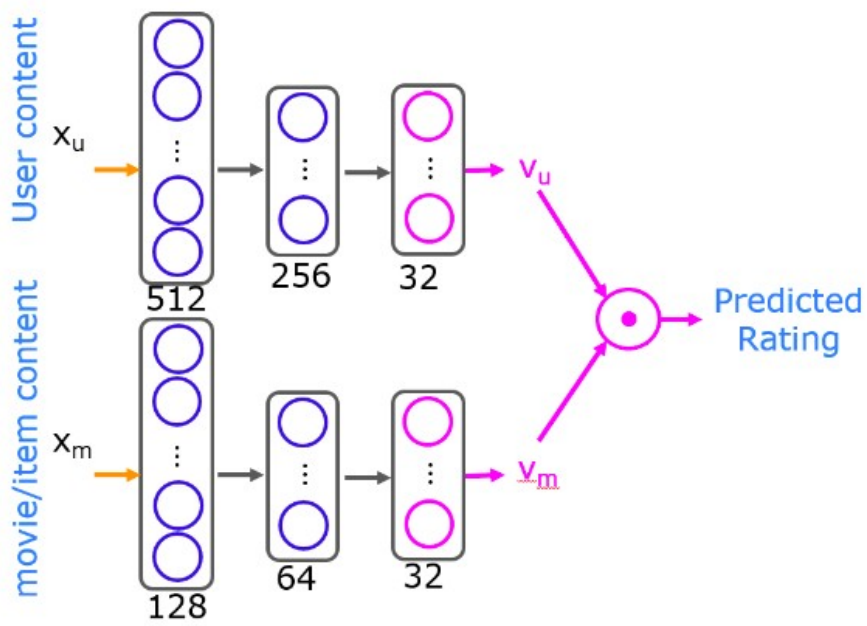
Correct
Similar movies generate similar v_m's.

3. Which of the following neural network configurations are valid for a content based filtering application? Please note carefully the dimensions of the neural network indicated in the diagram. Check all the options that apply: **1 / 1 point**



3. Which of the following neural network configurations are valid for a content based filtering application? Please note carefully the dimensions of the neural network indicated in the diagram. Check all the options that apply:

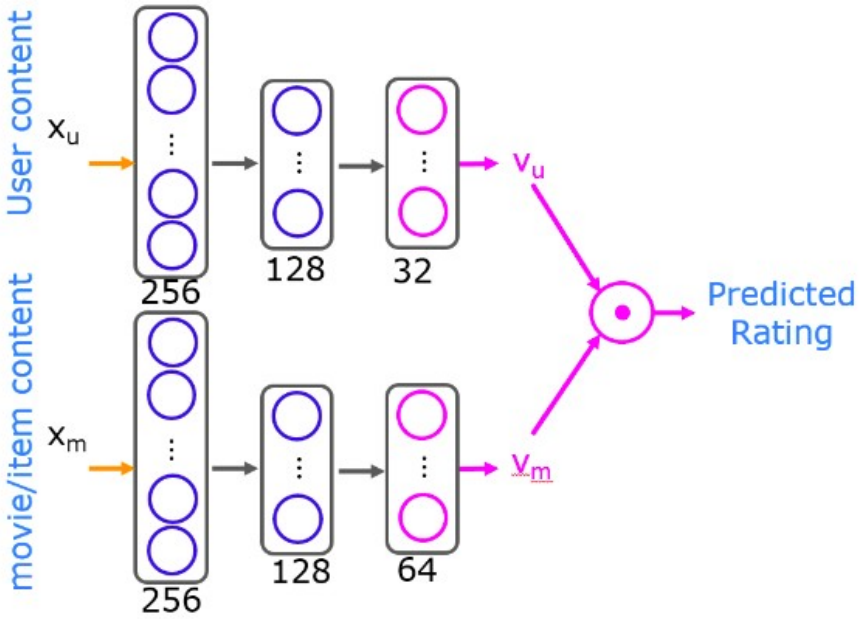
1 / 1 point



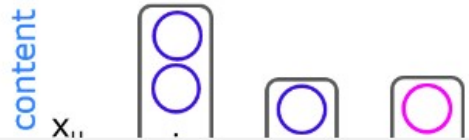
- [x] The user and the item networks have different architectures

Correct
User and item networks can be the same or different sizes.

Correct
User and item networks can be the same or different sizes.

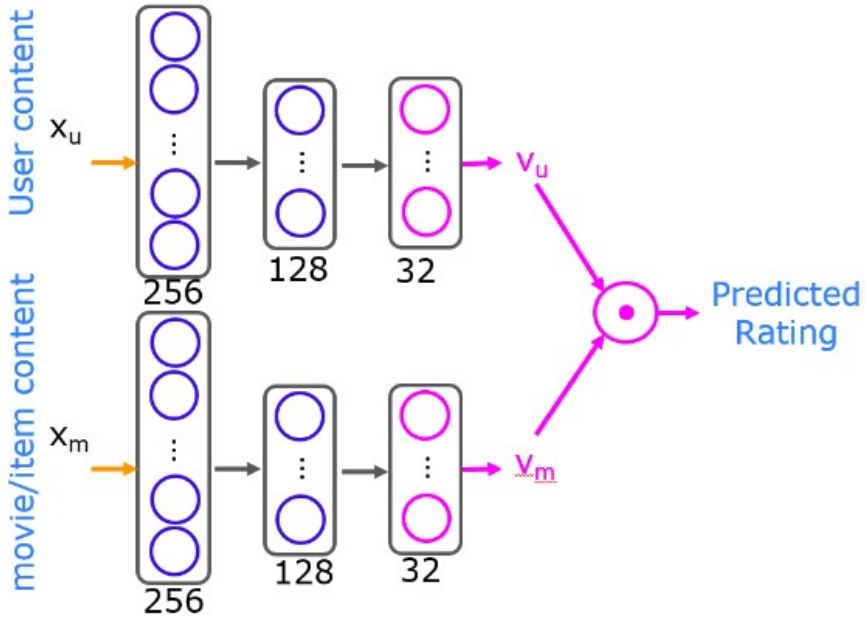


The user vector v_u is 32 dimensional, and the item vector v_m is 64 dimensional



256

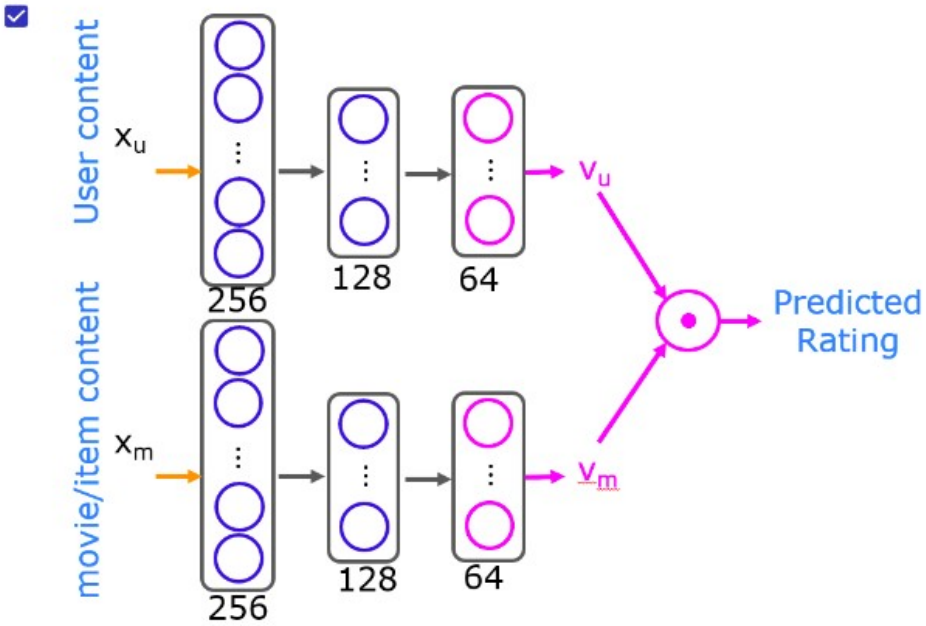
The user vector v_u is 32 dimensional, and the item vector v_m is 64 dimensional



Both the user and the item networks have the same architecture

Correct
User and item networks can be the same or different sizes.

Correct
User and item networks can be the same or different sizes.



The user and item networks have 64 dimensional v_u and v_m vector respectively

Correct
Feature vectors can be any size so long as v_u and v_m are the same size.

4. You have built a recommendation system to retrieve musical pieces from a large database of music, and have an algorithm that uses separate retrieval and ranking steps. If you modify the algorithm to add more musical pieces to the retrieved list (i.e., the retrieval step returns more items), which of these are likely to happen? Check all that apply.

1 / 1 point

- [ ] The quality of recommendations made to users should stay the same or worsen.
- [ ] The system's response time might decrease (i.e., users get recommendations more quickly)
- [x] The quality of recommendations made to users should stay the same or improve.

Correct
A larger retrieval list gives the ranking system more options to choose from which should maintain or improve recommendations.

- [x] The system's response time might increase (i.e., users have to wait longer to get recommendations)

Correct
A larger retrieval list may take longer to process which may *increase* response time.

5.

1 / 1 point

To speed up the response time of your recommendation system, you can pre-compute the vectors v_m for all the items you might recommend. This can be done even before a user logs in to your website and even before you know the x_u or v_u vector. True/False?

- (•) True
- ( ) False

- [ ] The quality of recommendations made to users should stay the same or worsen.
- [ ] The system's response time might decrease (i.e., users get recommendations more quickly)
- [x] The quality of recommendations made to users should stay the same or improve.

Correct
A larger retrieval list gives the ranking system more options to choose from which should maintain or improve recommendations.

- [x] The system's response time might increase (i.e., users have to wait longer to get recommendations)

Correct
A larger retrieval list may take longer to process which may *increase* response time.

5. **1 / 1 point**

To speed up the response time of your recommendation system, you can pre-compute the vectors v_m for all the items you might recommend. This can be done even before a user logs in to your website and even before you know the x_u or v_u vector. True/False?

- (•) True
- ( ) False

Correct
The output of the item/movie neural network, v_m is not dependent on the user network when making predictions. Precomputing the results speeds up the prediction process.

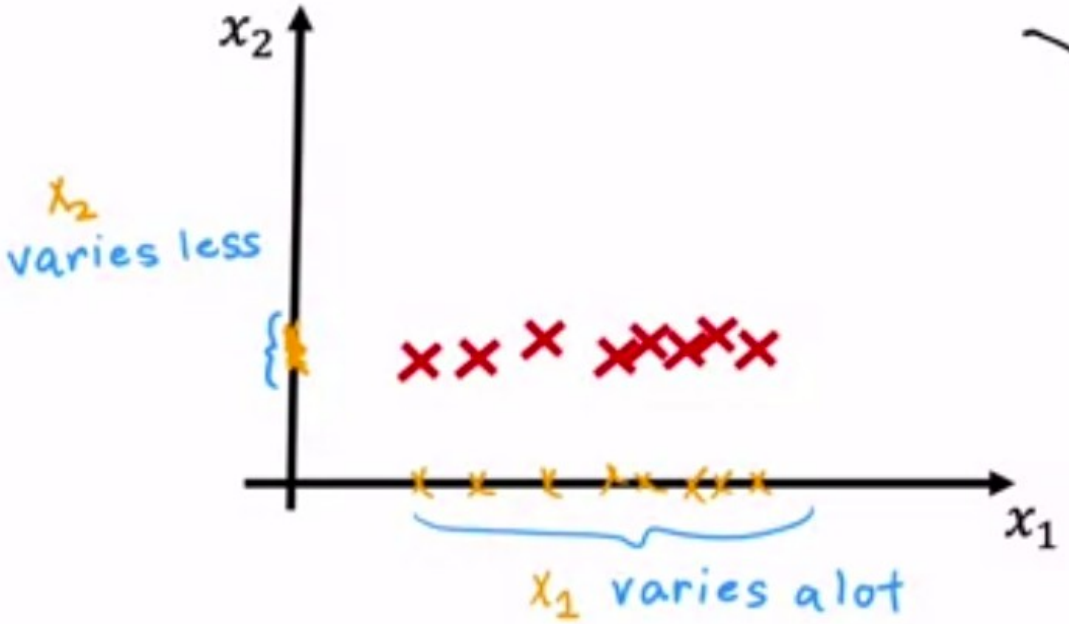
DeepLearning.AI

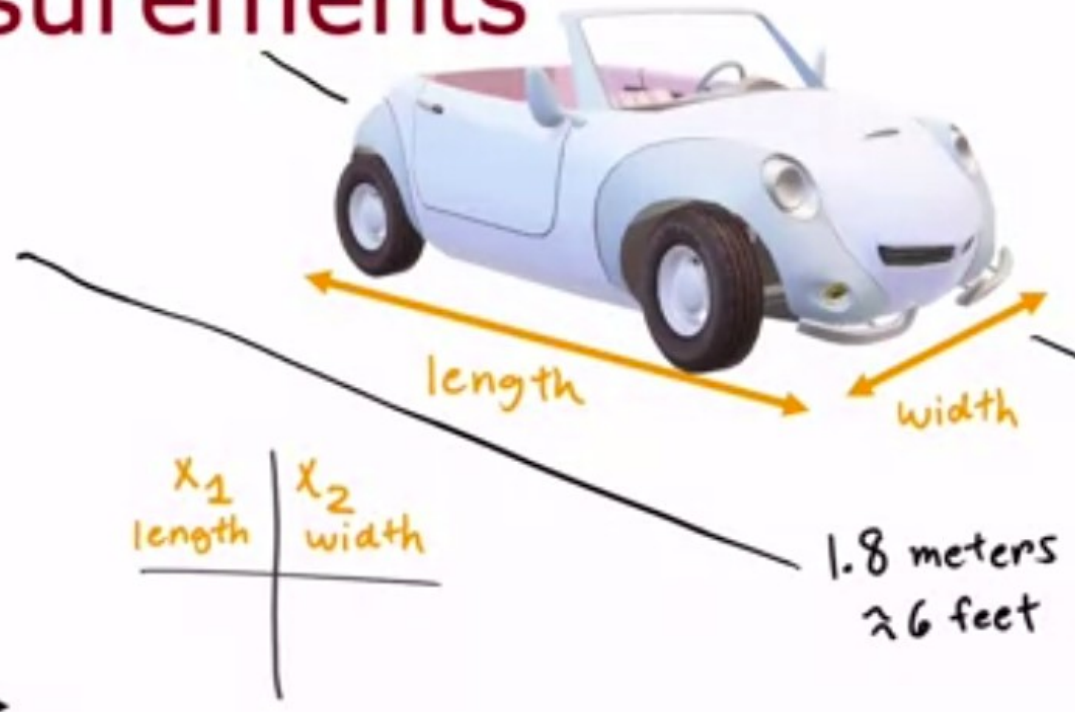
Stanford
ONLINE

Principal Component Analysis (Optional)

Reducing the number of features

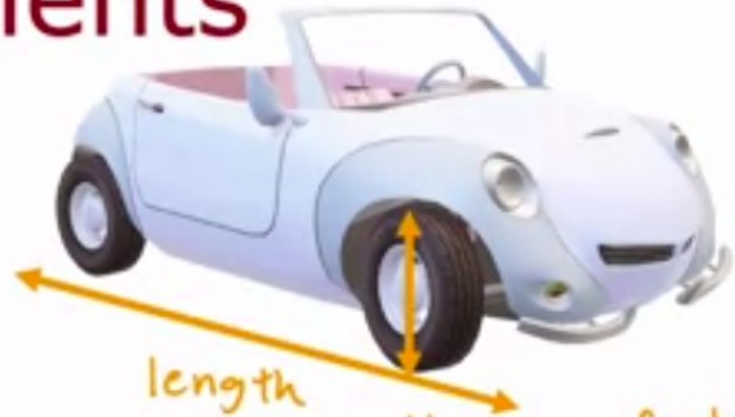
Car measurements

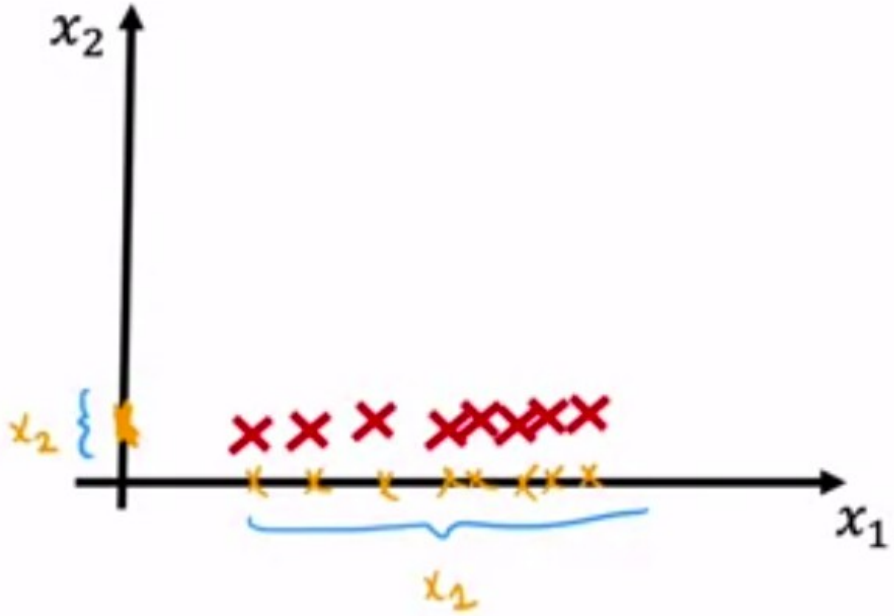




can just take x_1
to reduce number of features

Car measurements



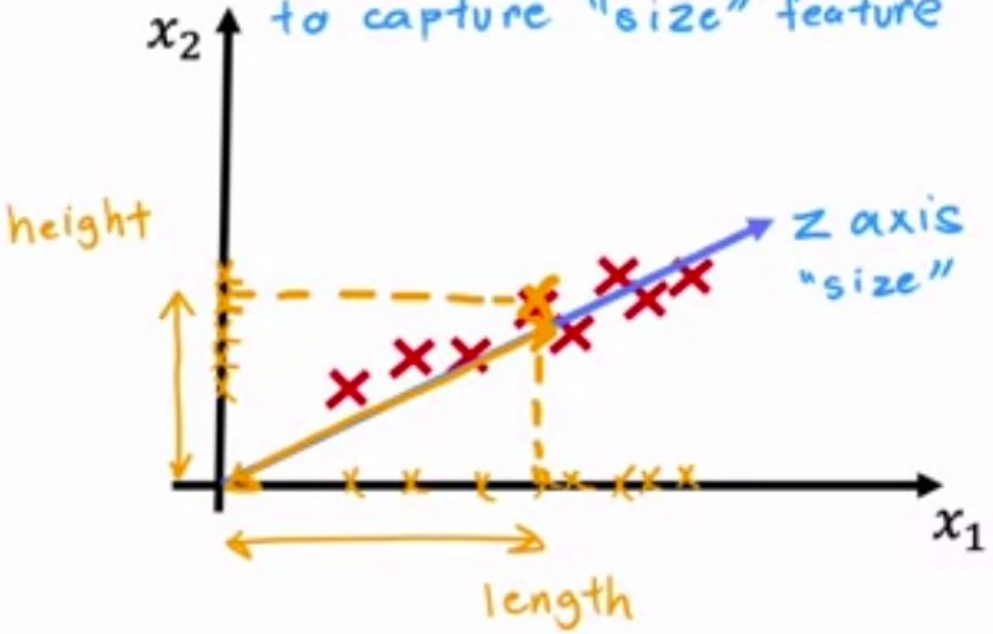


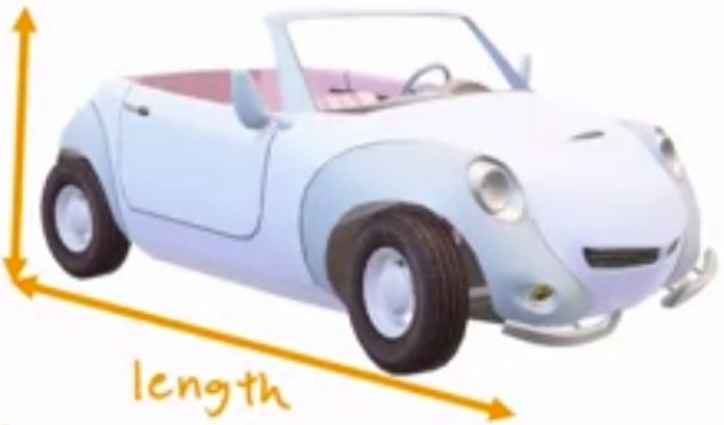
x_1	x_2
length	diameter

can just take x_1
to reduce number of features

Size

PCA: find new axis and coordinates
use fewer numbers
to capture "size" feature



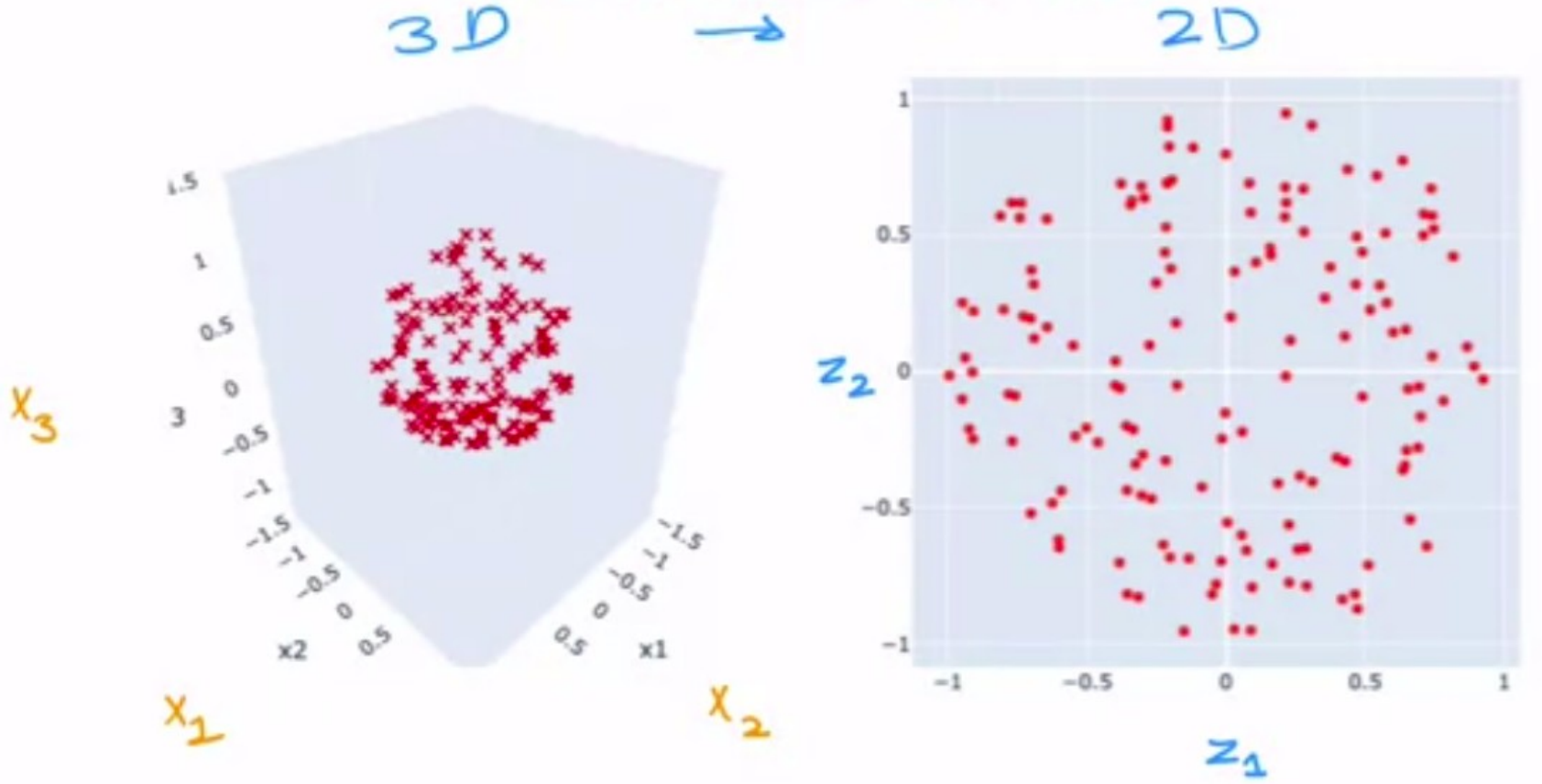


x_1	x_2
length	height

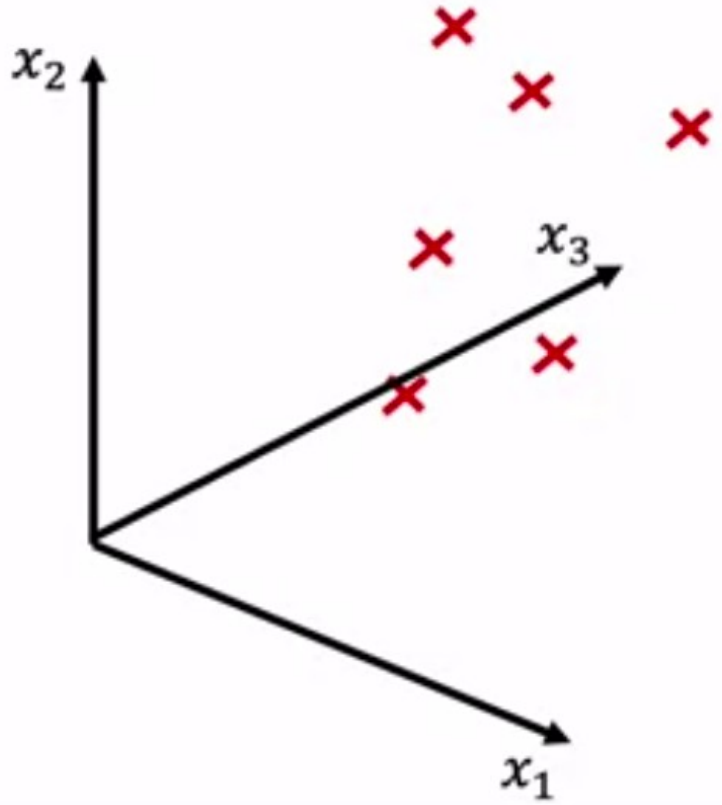
2 → 1 features

many features → 2 or 3 features

From 3D to 2D



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
...	...	...	...



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
...	...	...	...	...

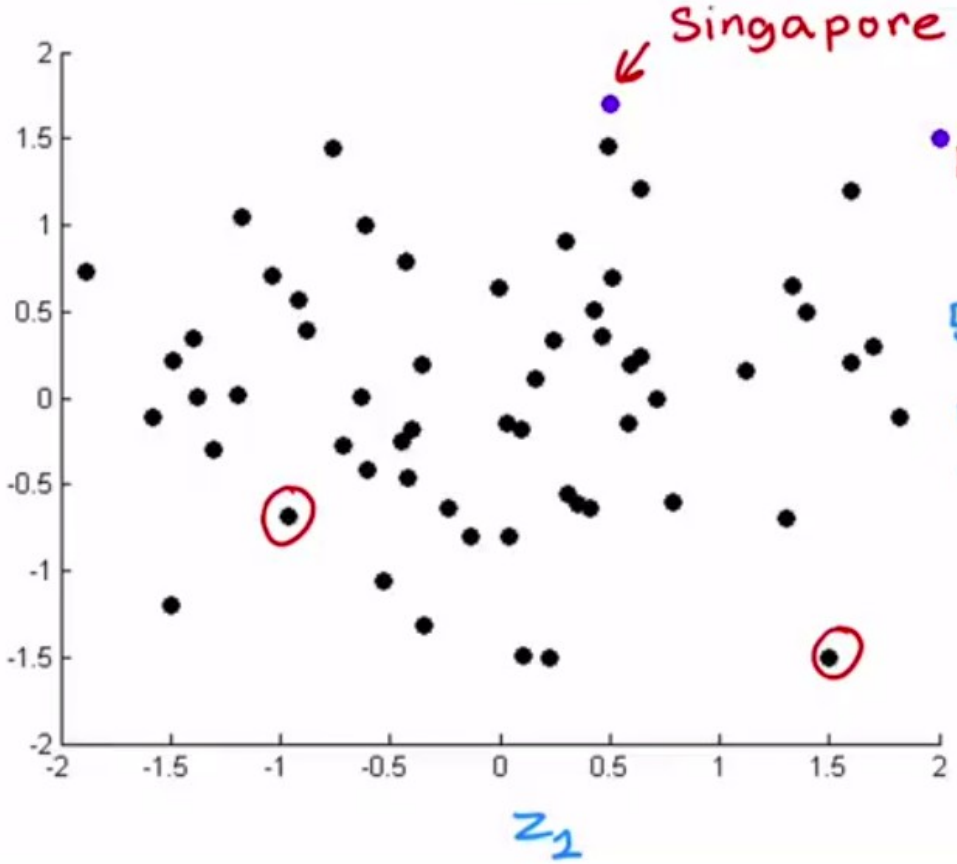
what if 50 features?

50 features

Country	x_1 GDP (trillions of US$)	x_2 Per capita GDP (thousands of intl. $)	x_3 Human Develop-ment Index	x_4 Life expectancy	... Poverty Index (Gini as percentage)	Mean household income (thousands of US$)	...
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



DeepLearning.AI

Stanford
ONLINE

Principal Component Analysis

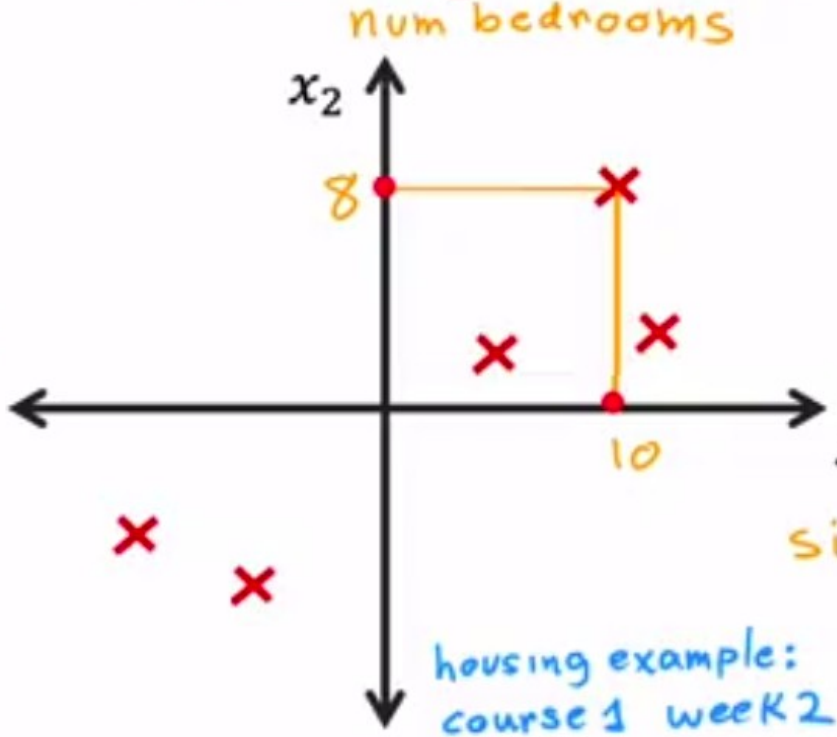
PCA Algorithm

PCA algorithm

coordinates

$x_1 = 10 \quad x_2 = 8$

Can we choose a different axis?

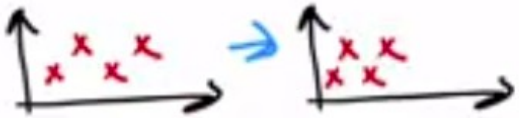


housing example: course 1 week 2

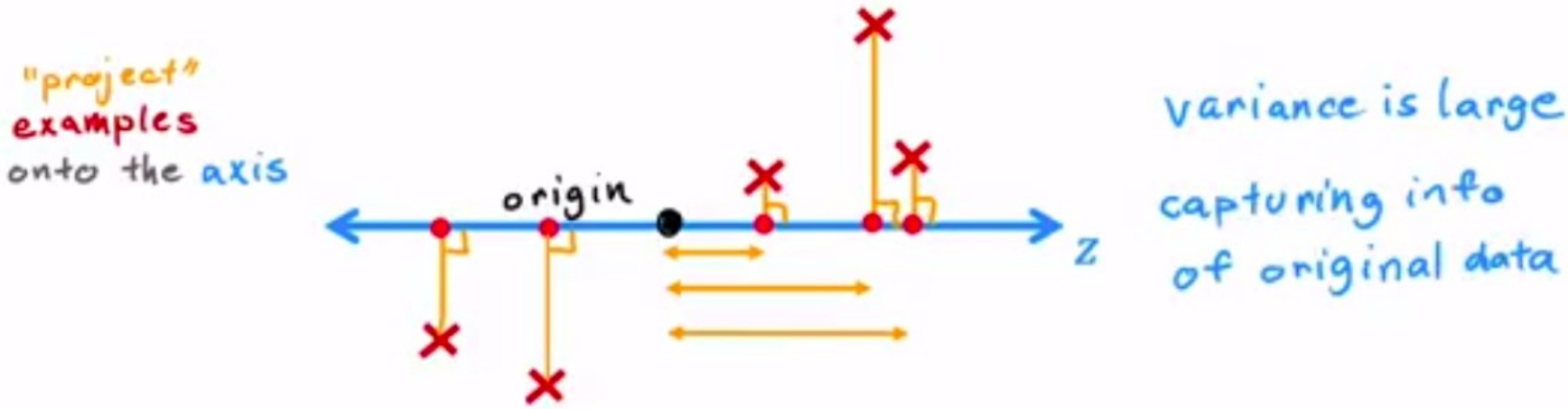
Preprocess features

Normalized to have zero mean

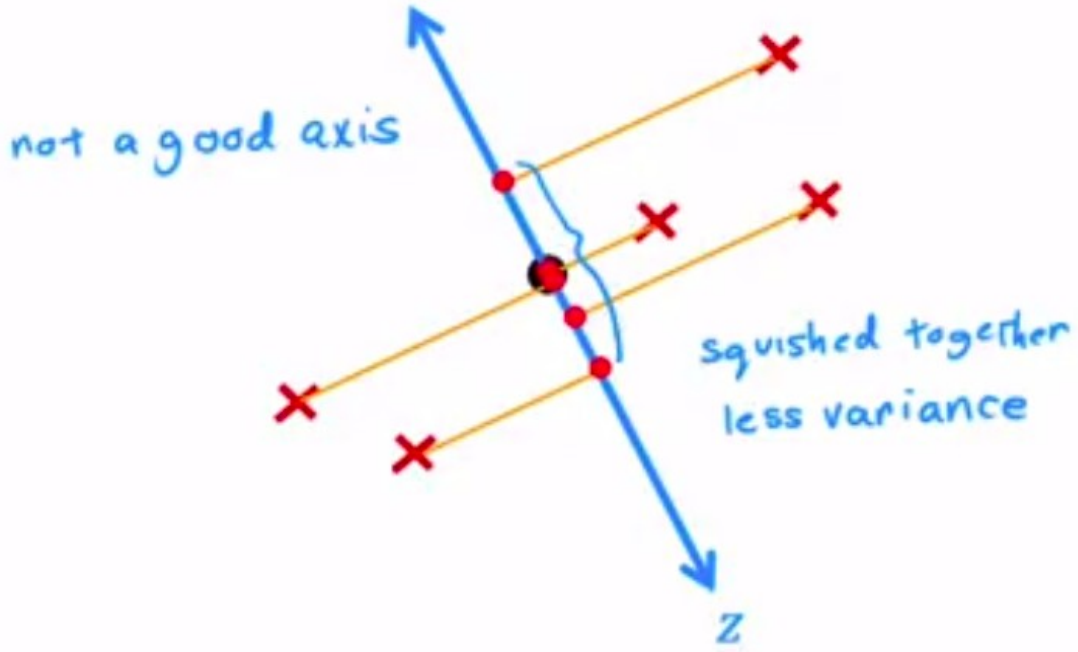
feature scaling



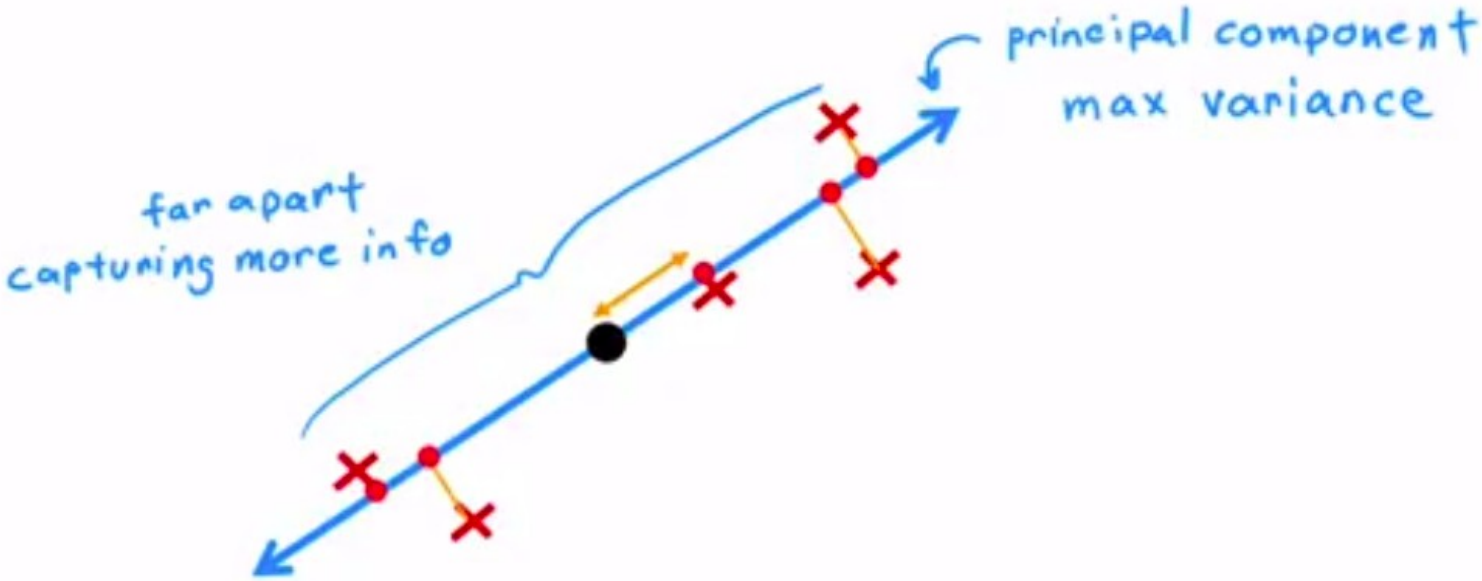
Choose an axis

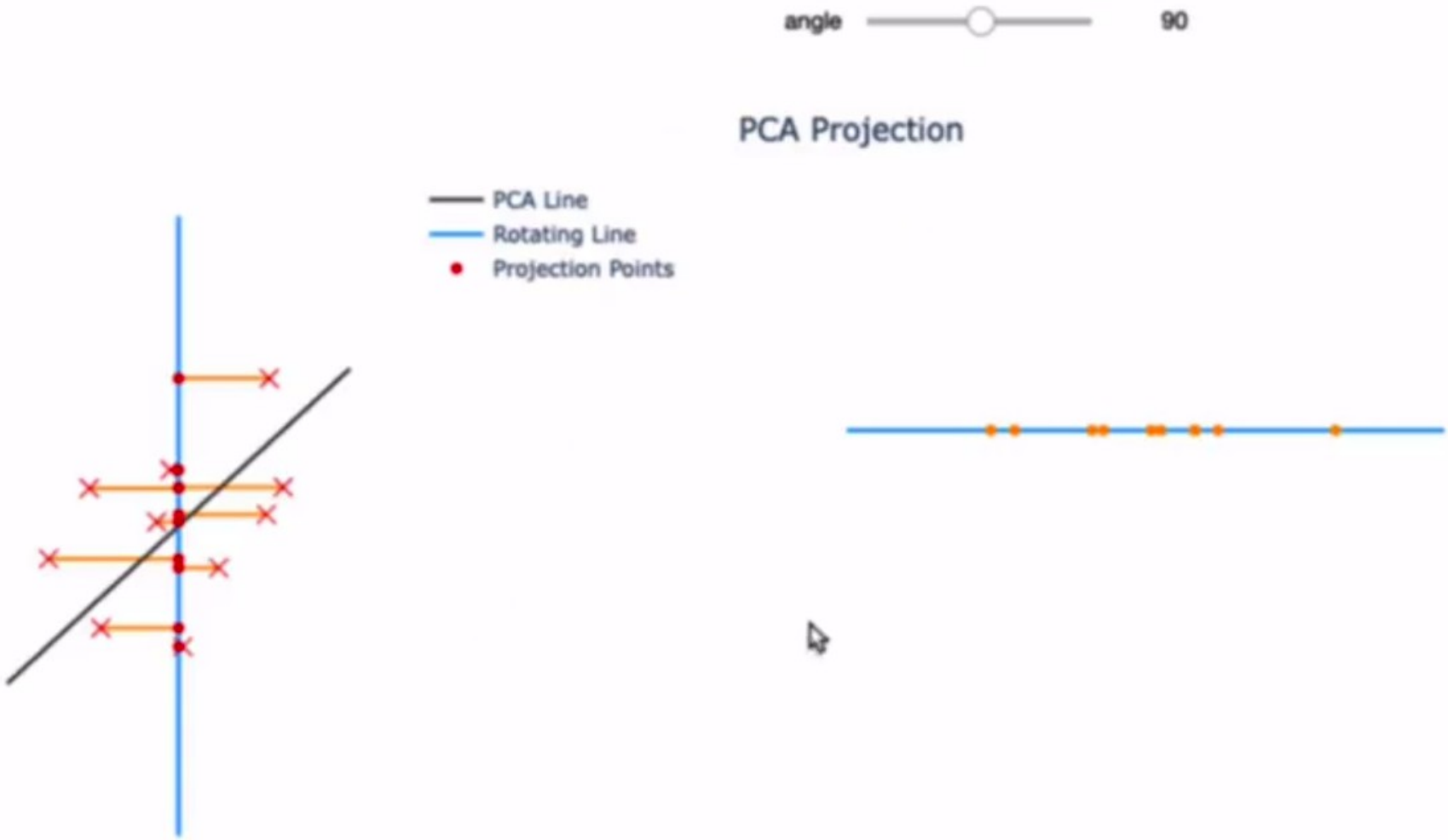


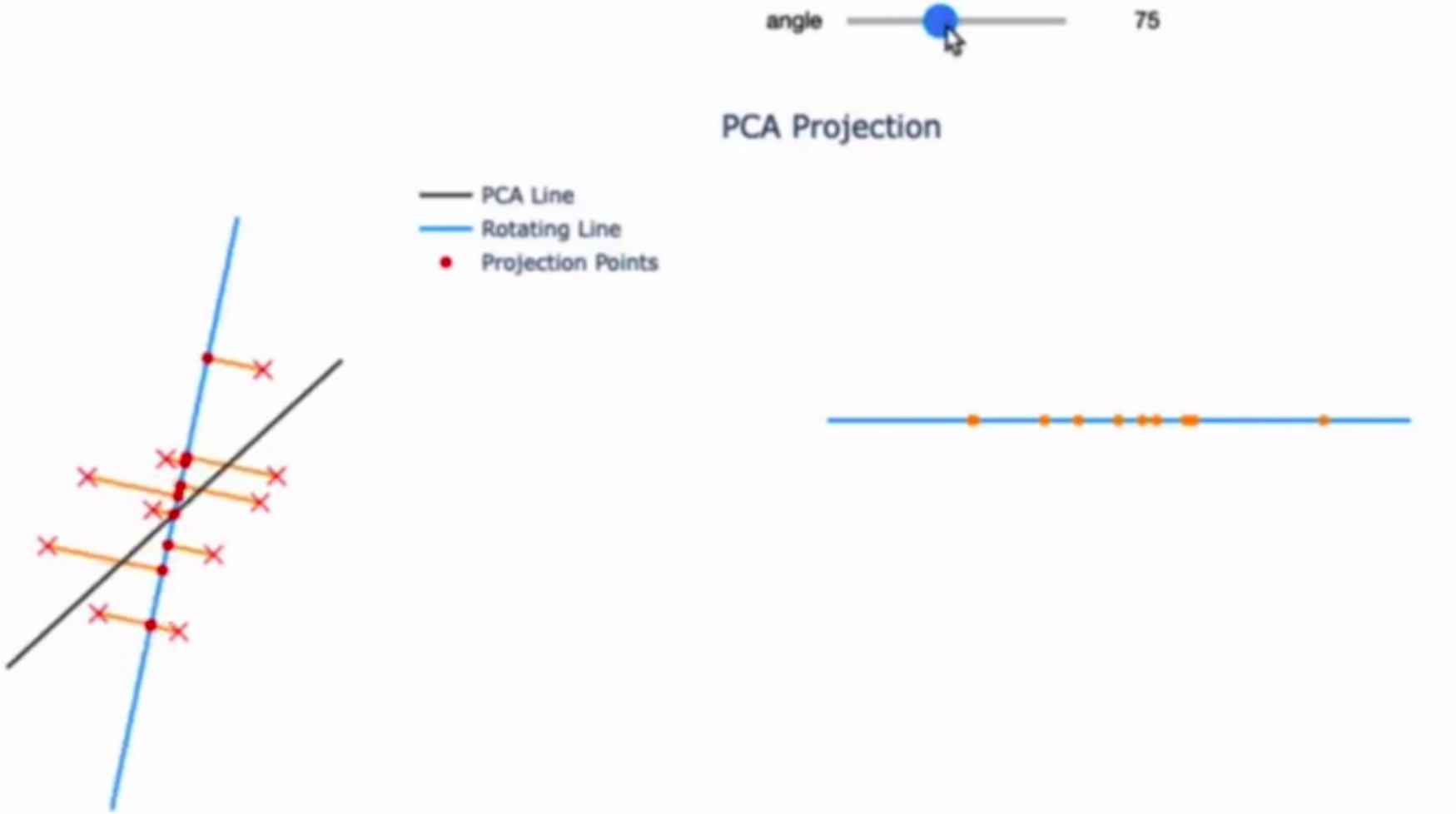
Choose an axis

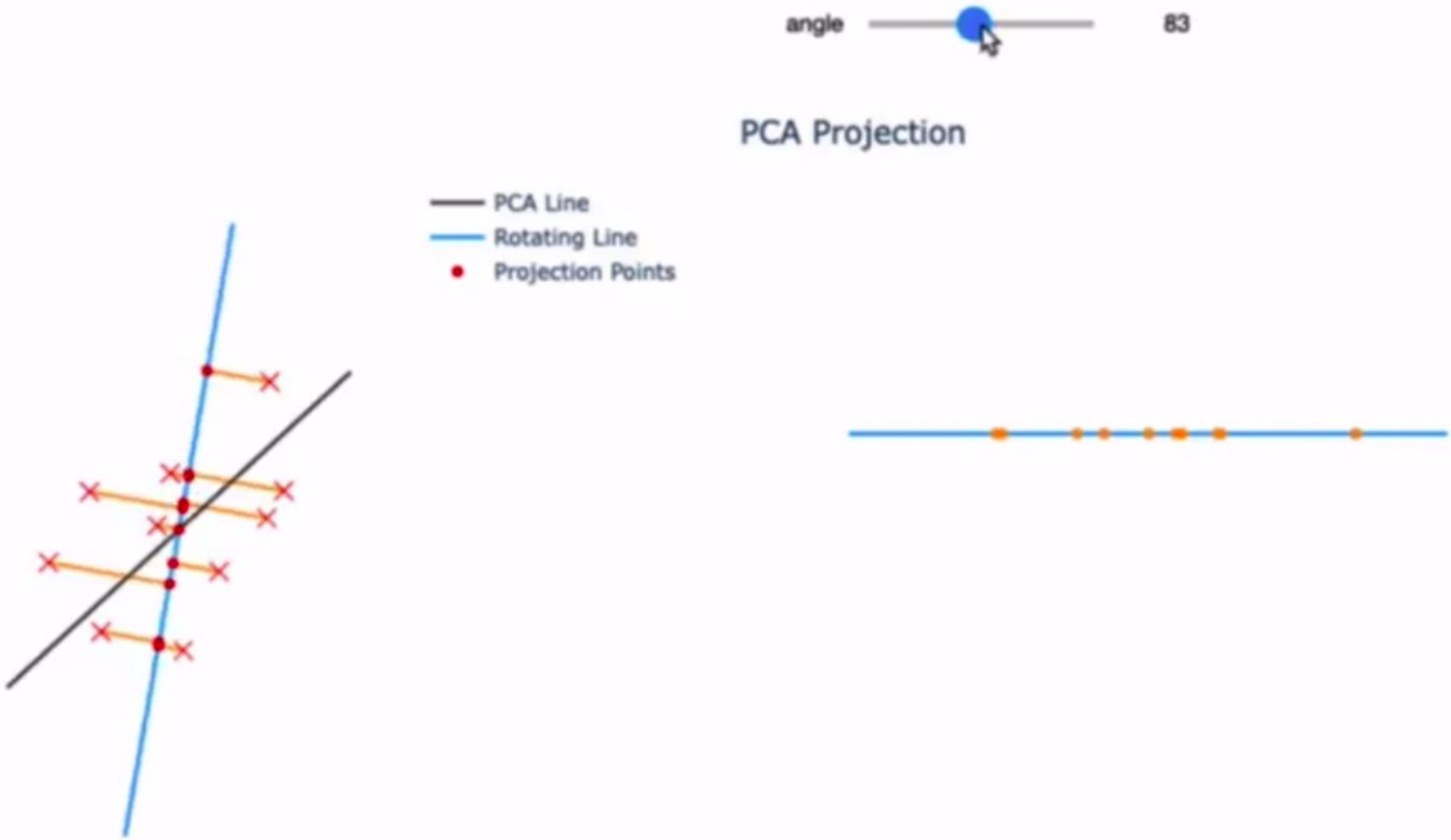


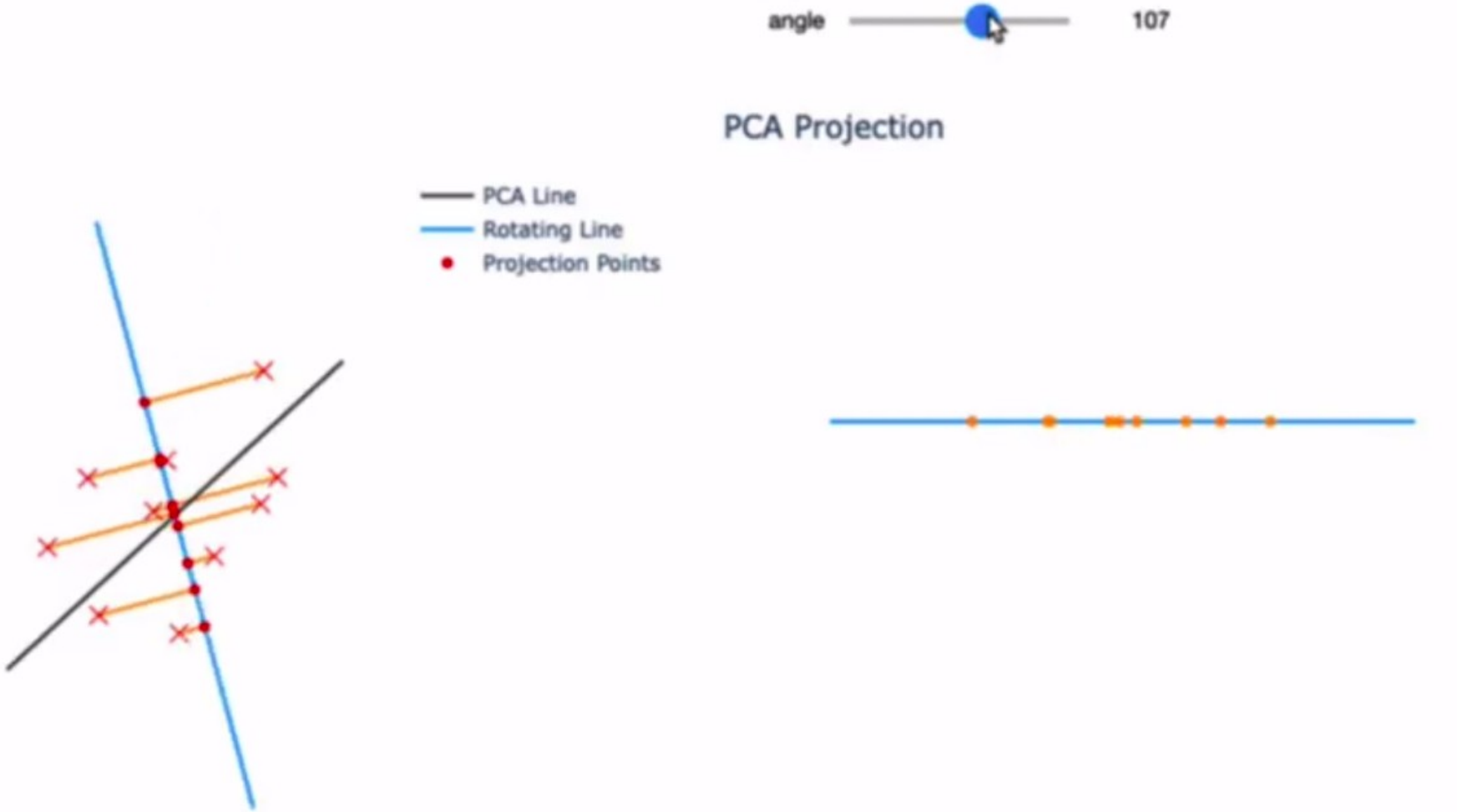
Choose an axis

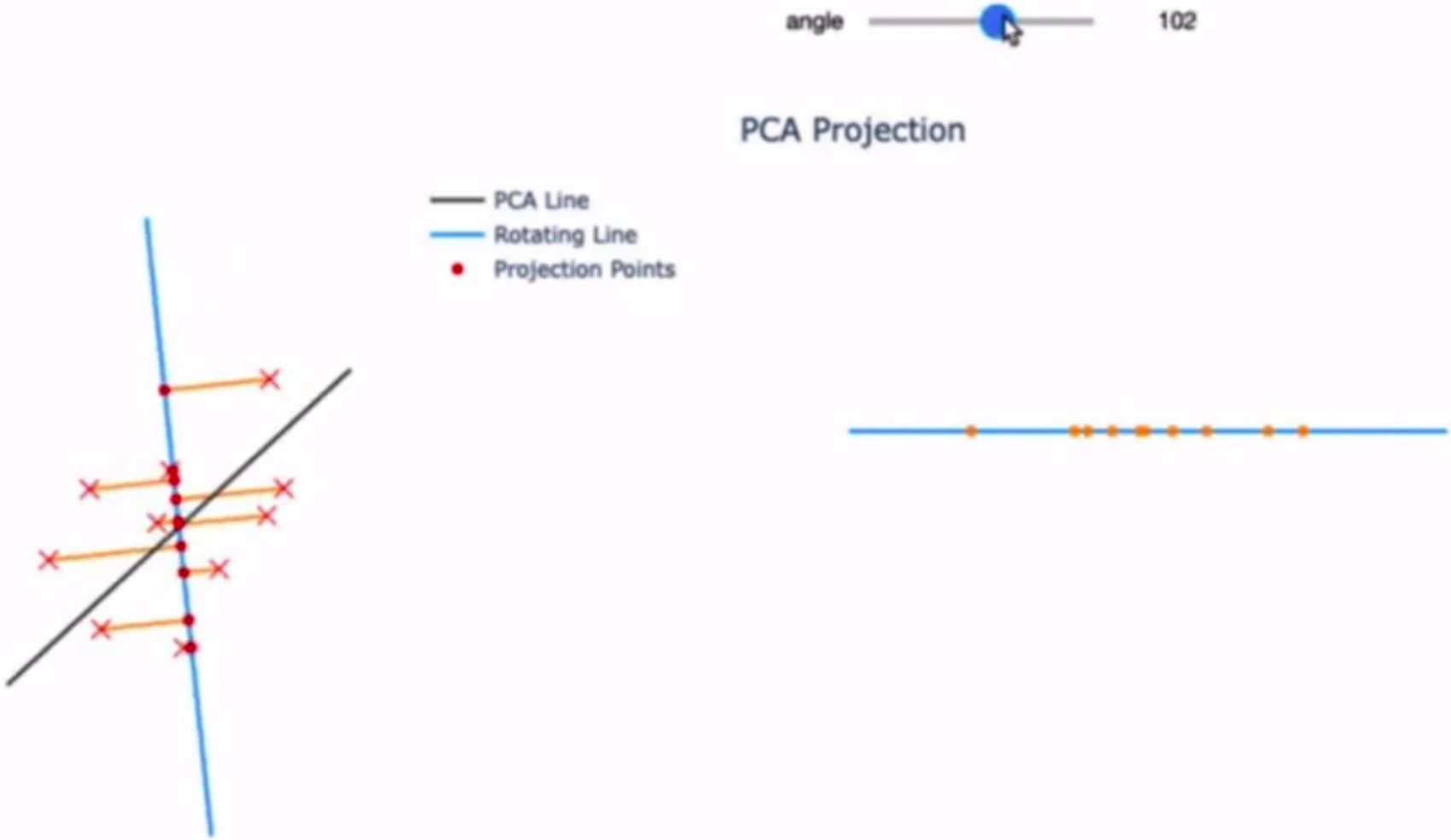


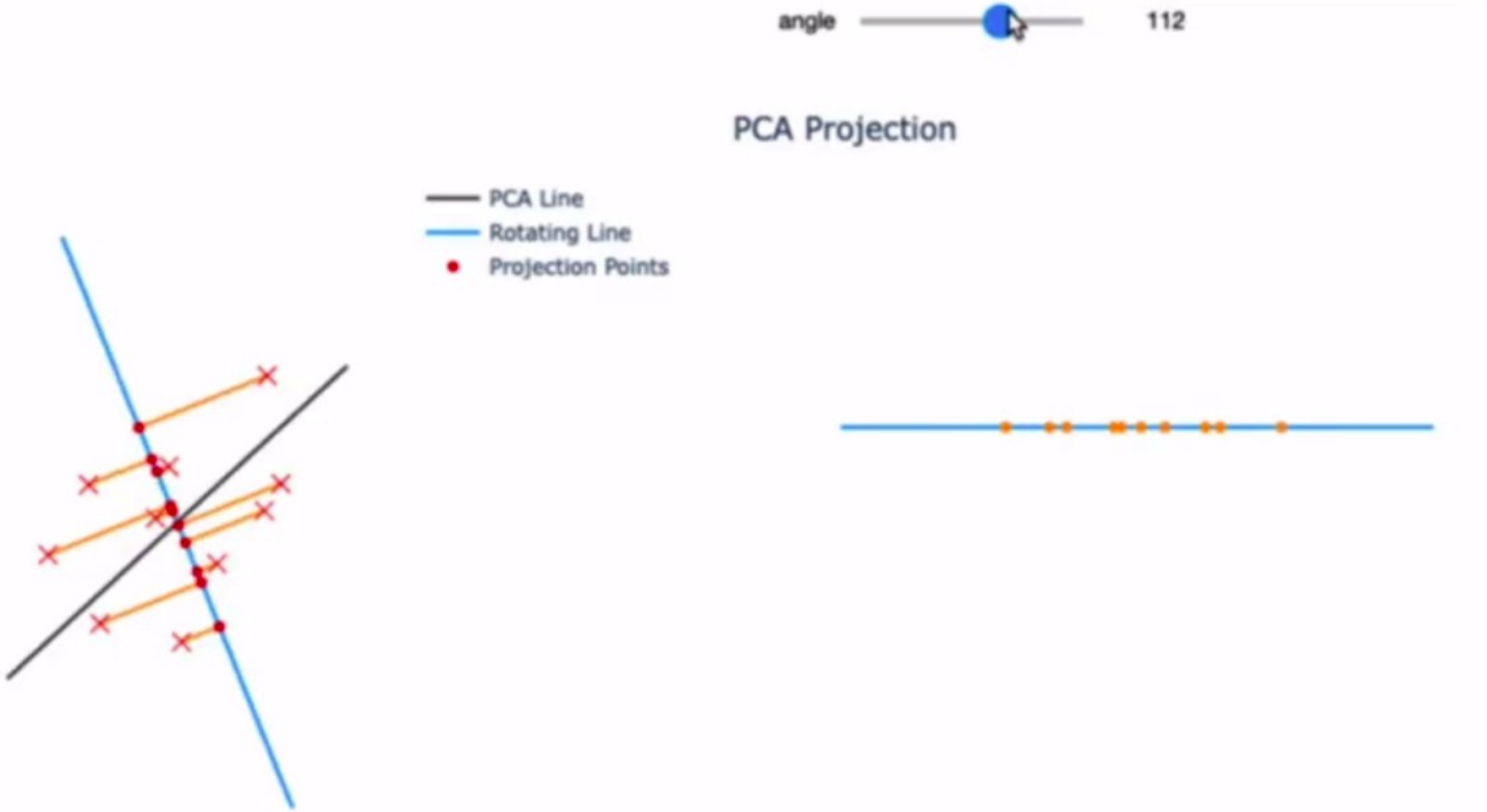

angle
90
PCA Projection
PCA Line
Rotating Line
Projection Points

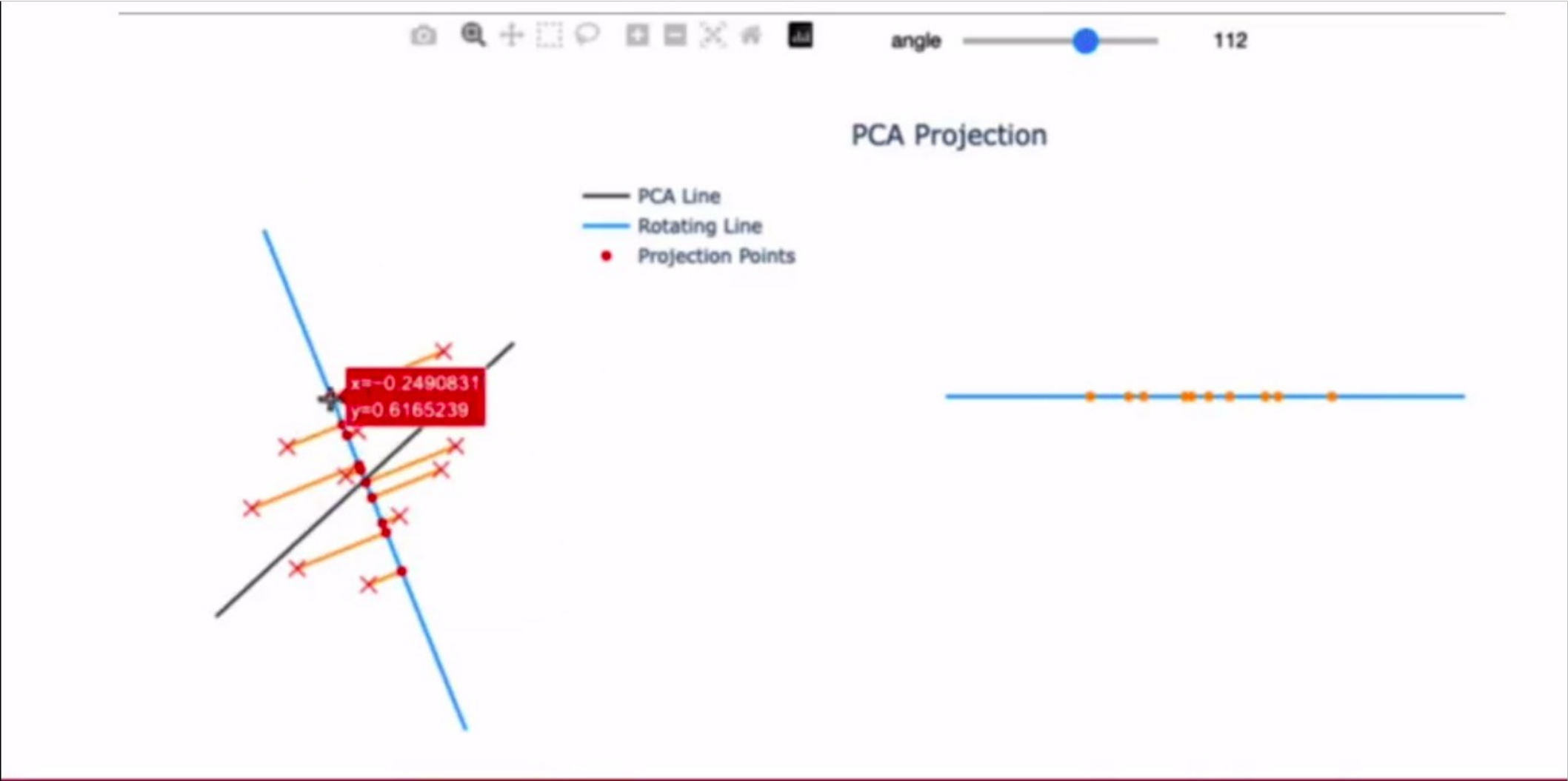

angle
75
PCA Projection
PCA Line
Rotating Line
Projection Points

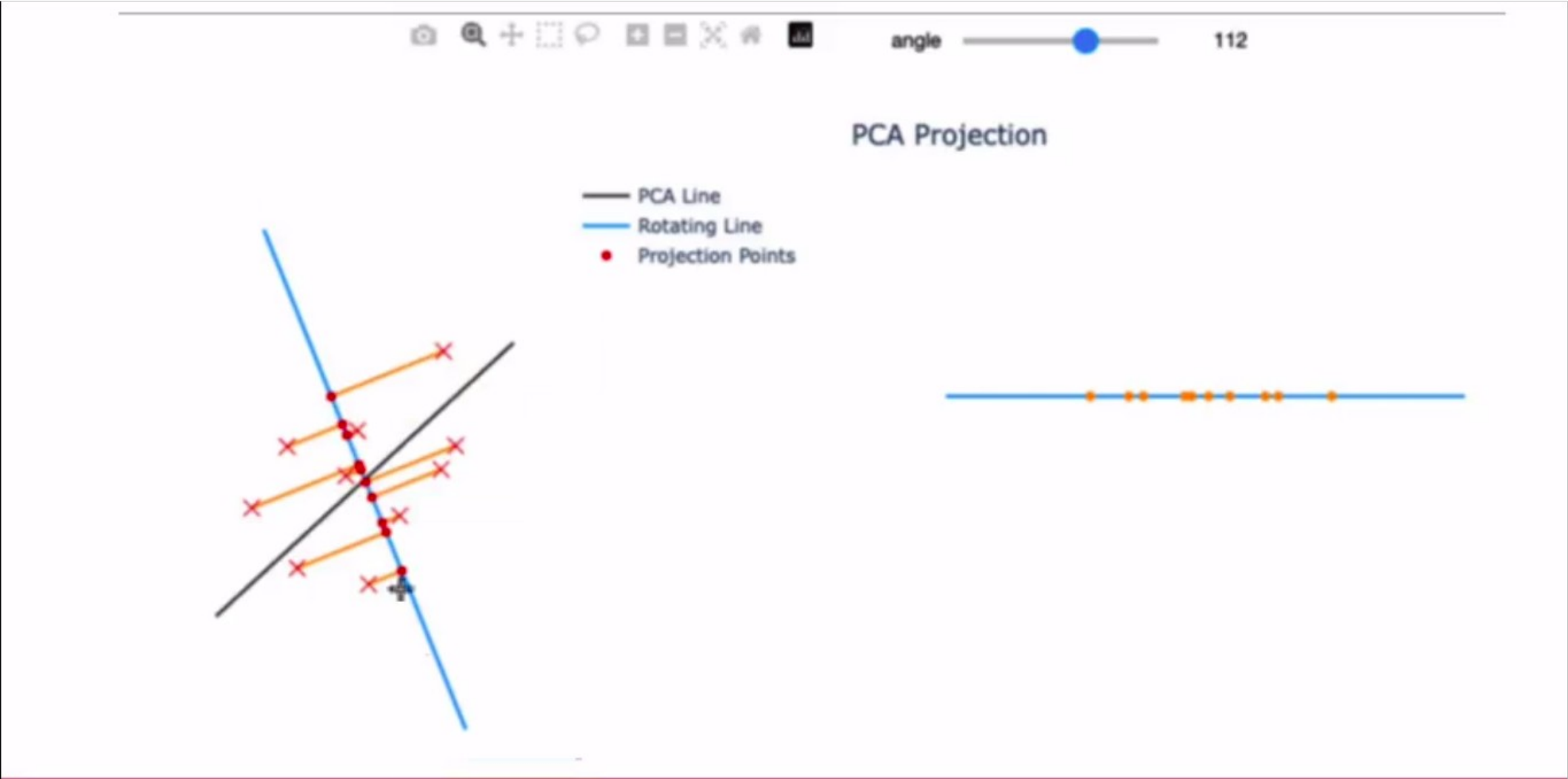

angle
83
PCA Projection
PCA Line
Rotating Line
Projection Points

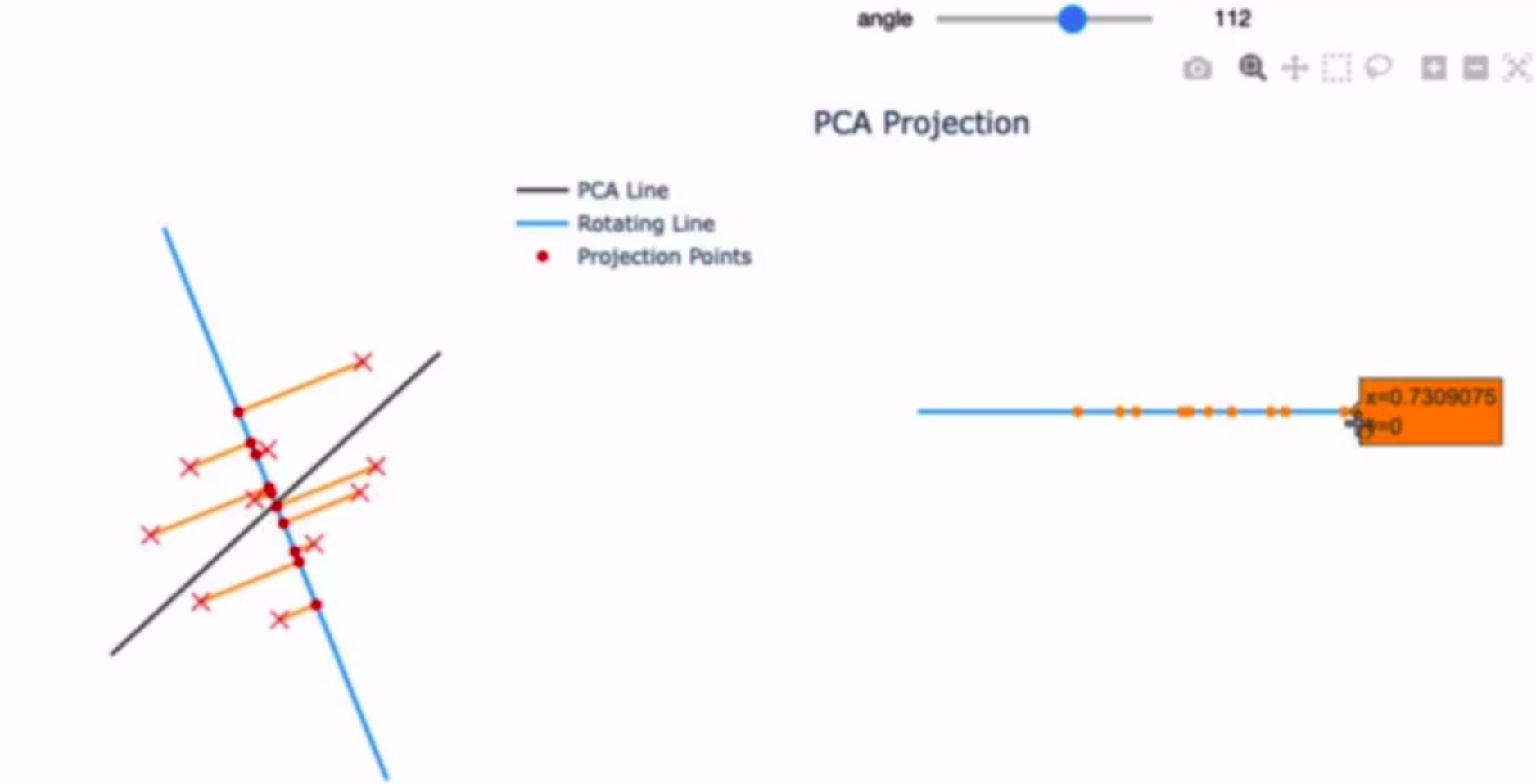

angle
107
PCA Projection
PCA Line
Rotating Line
Projection Points

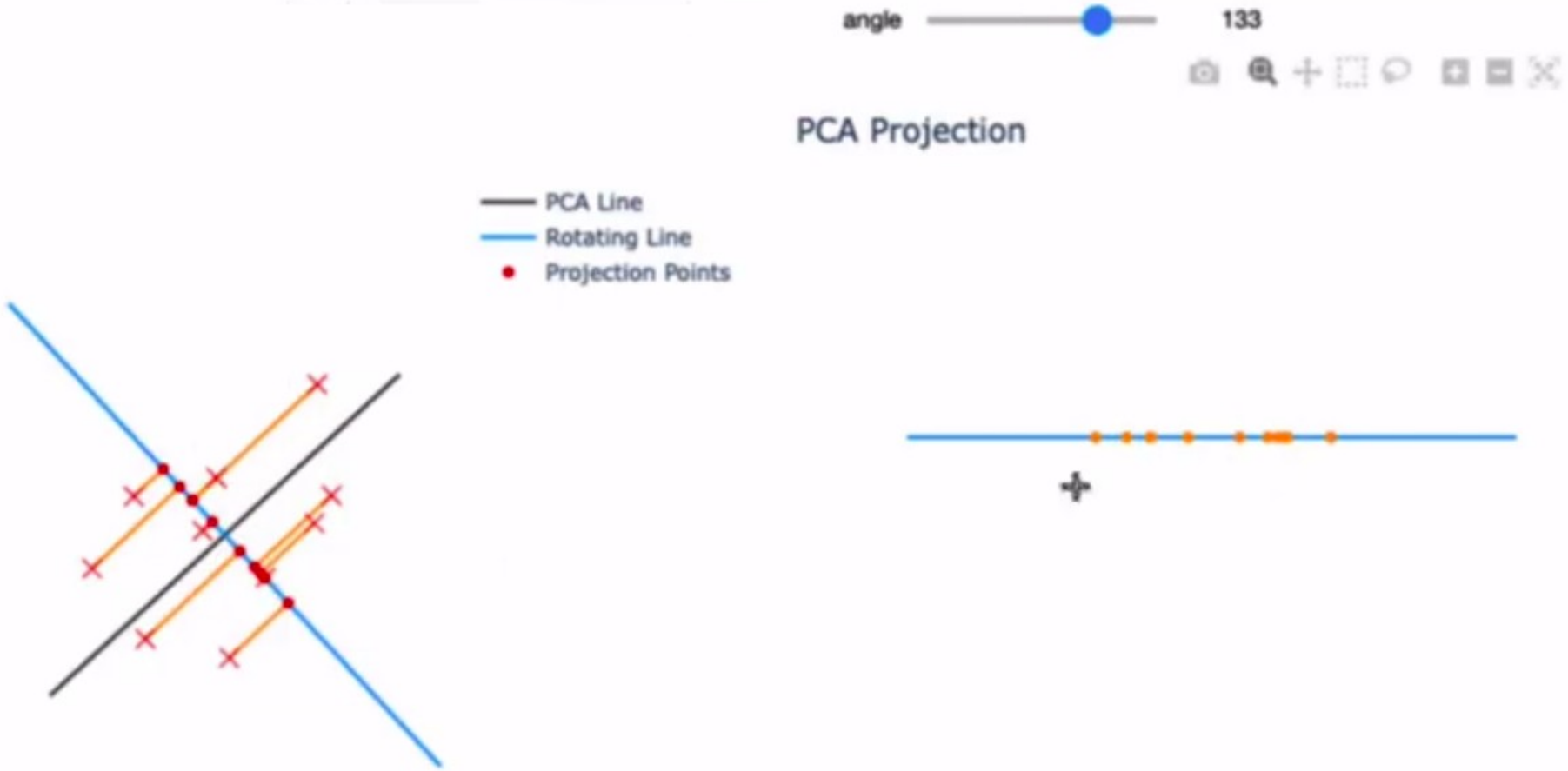

angle
102
PCA Projection
PCA Line
Rotating Line
Projection Points

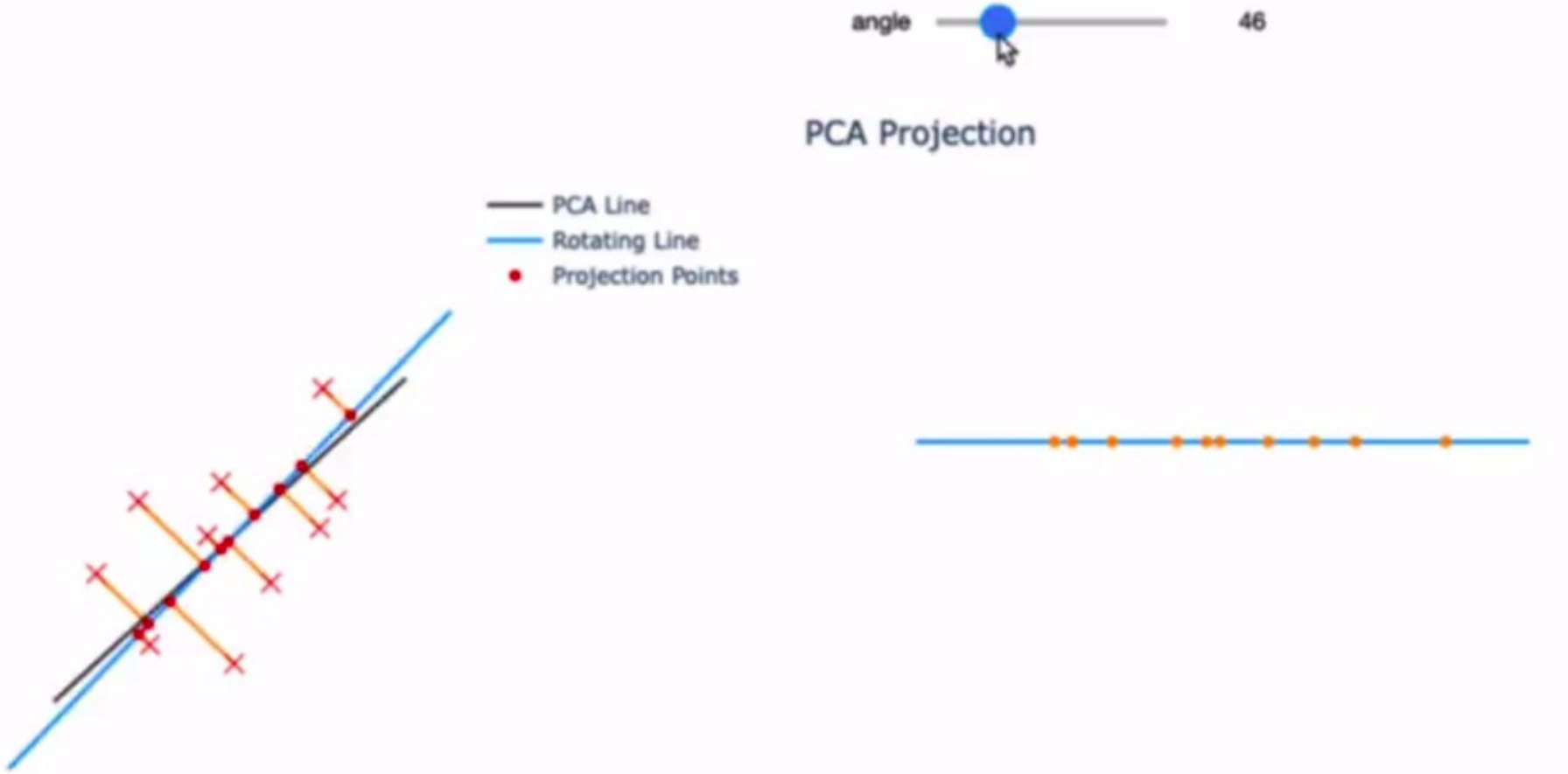

angle
112
PCA Projection
PCA Line
Rotating Line
Projection Points

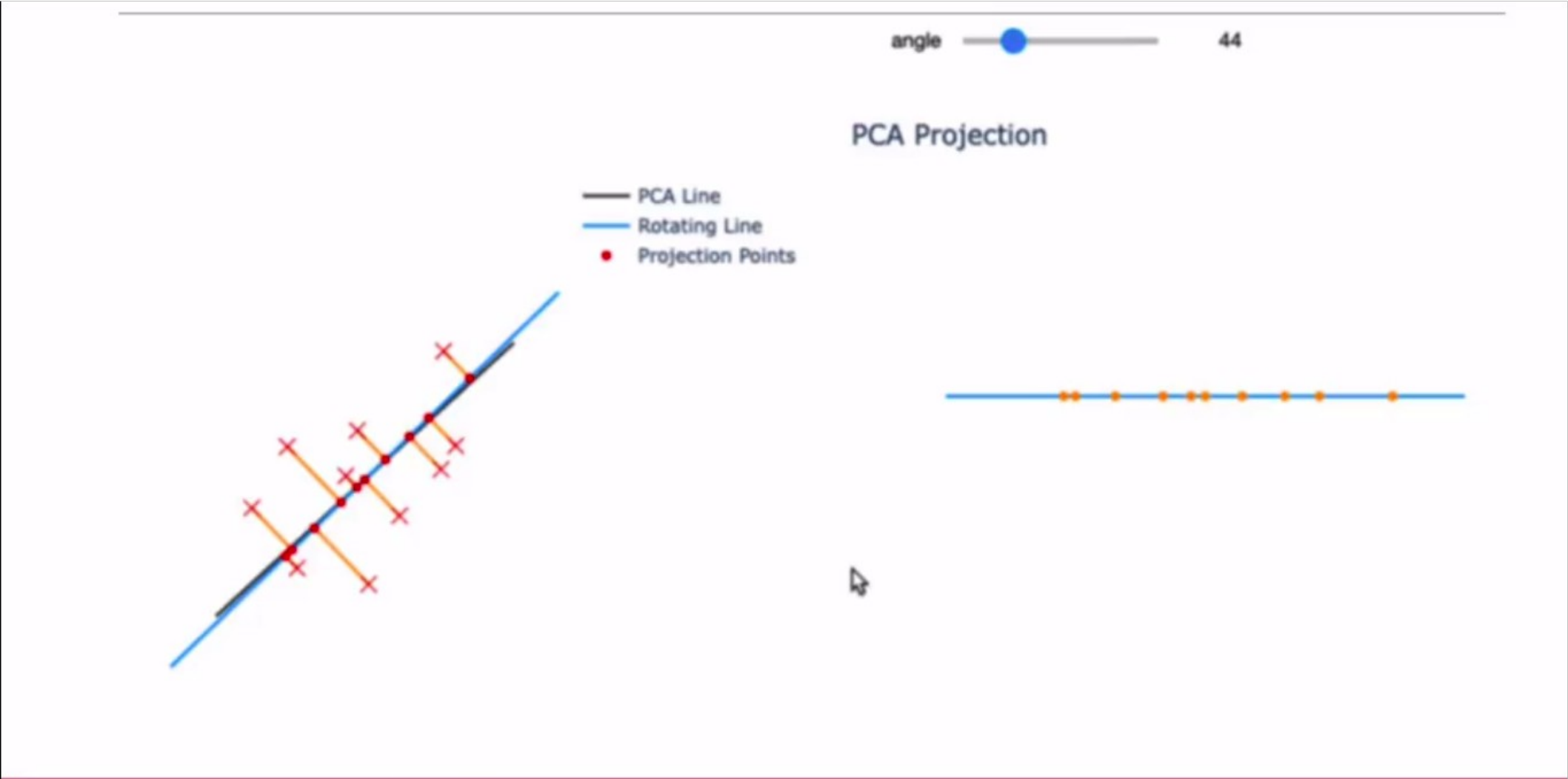

angle
112
PCA Projection
PCA Line
Rotating Line
Projection Points
x=-0.2490831
y=0.6165239


angle
112
PCA Projection
PCA Line
Rotating Line
Projection Points

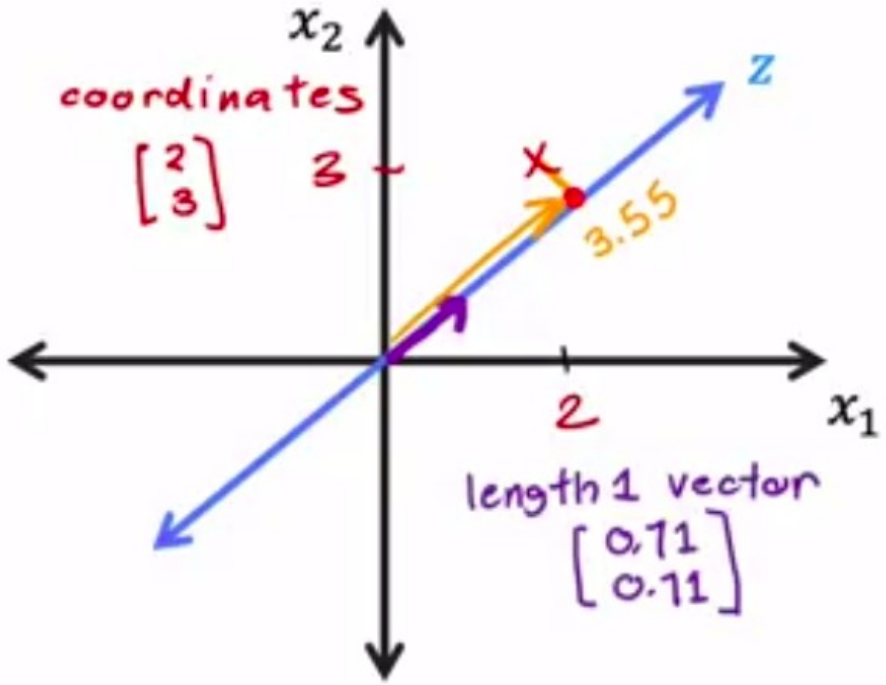

angle
112
PCA Projection
PCA Line
Rotating Line
Projection Points
x=0.7309075
y=0


angle
133
PCA Projection
PCA Line
Rotating Line
Projection Points


angle
46
PCA Projection
PCA Line
Rotating Line
Projection Points


angle
44
PCA Projection
PCA Line
Rotating Line
Projection Points

Coordinate on the new axis

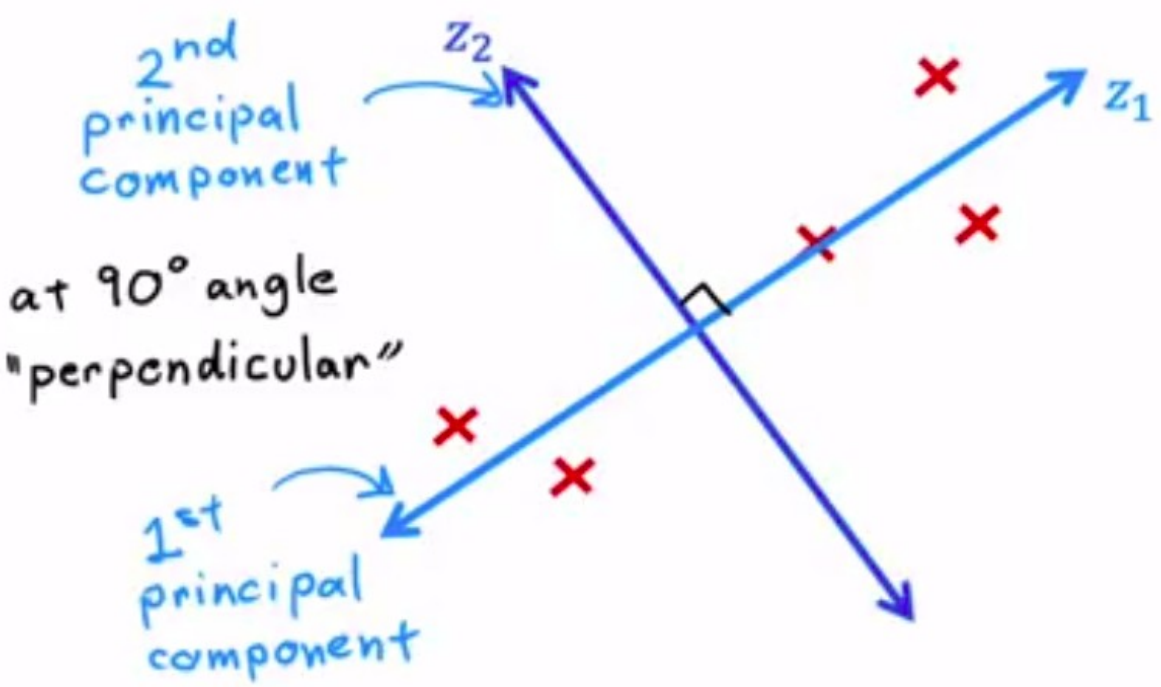


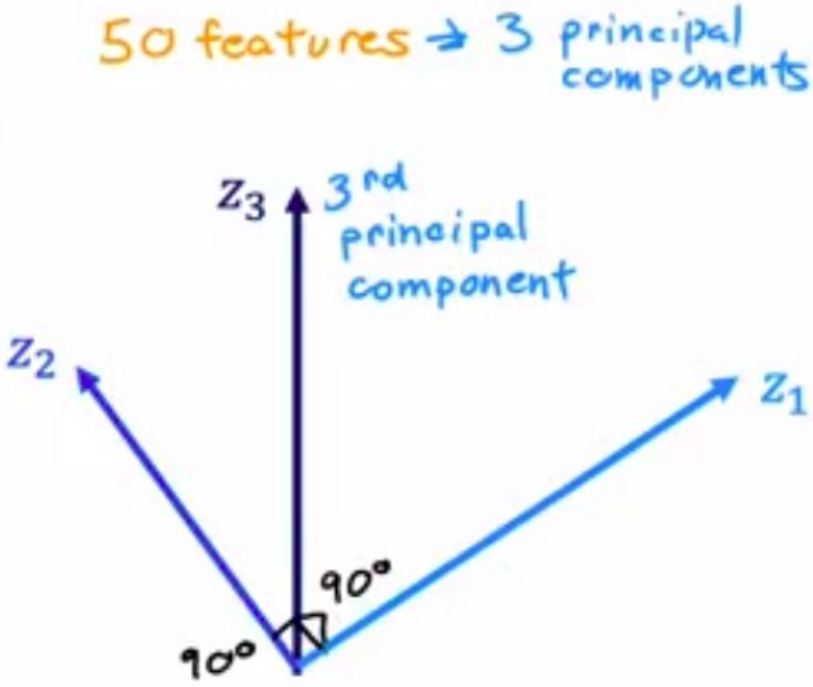
dot product

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

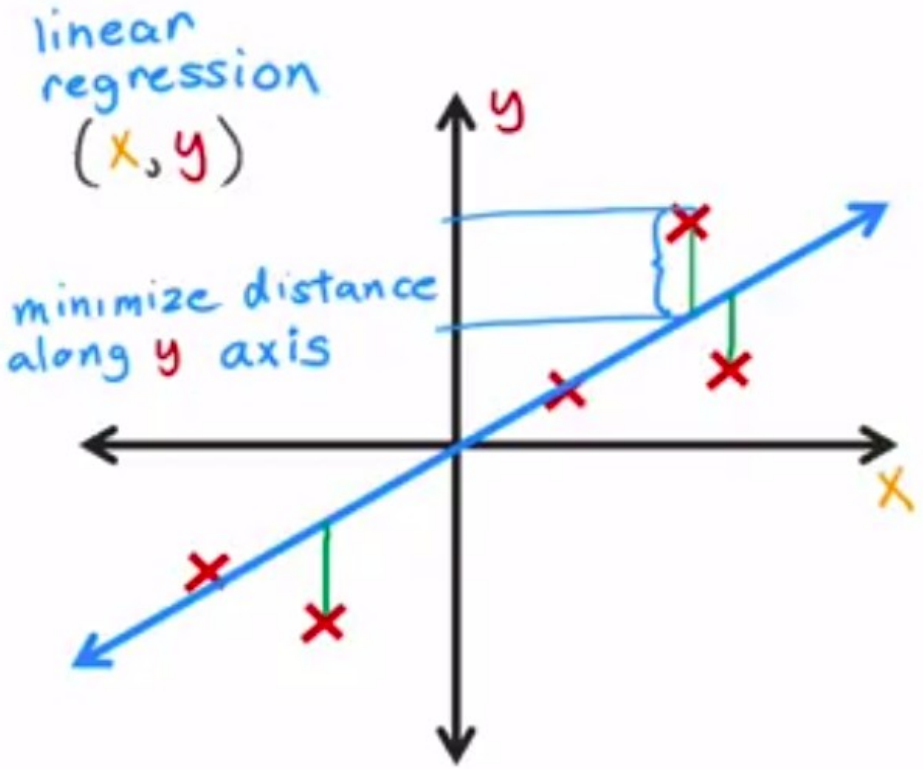
$2 \times 0.71 + 3 \times 0.71 = 3.55$

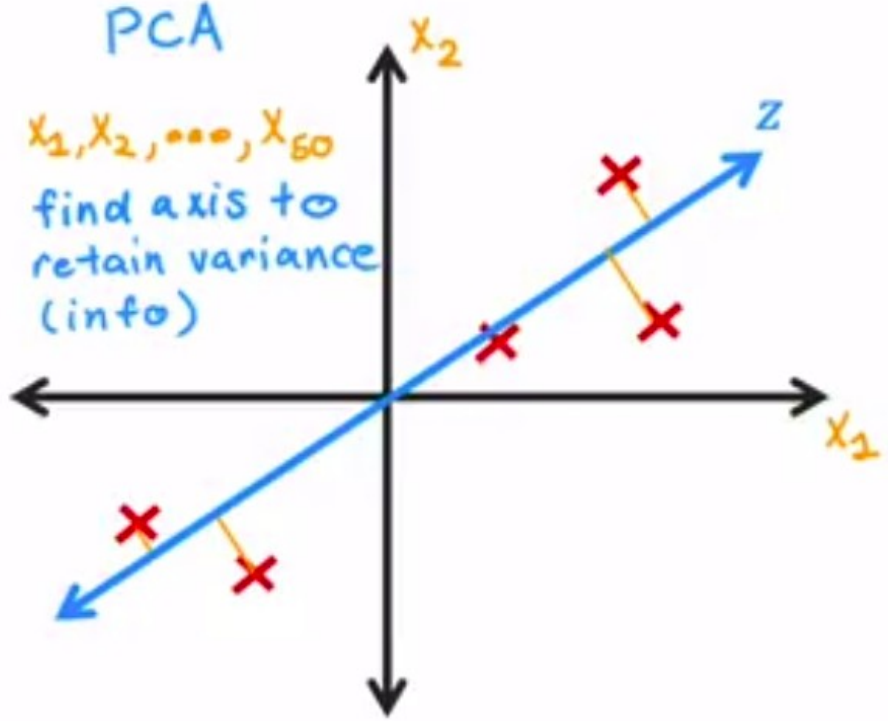
More principal components



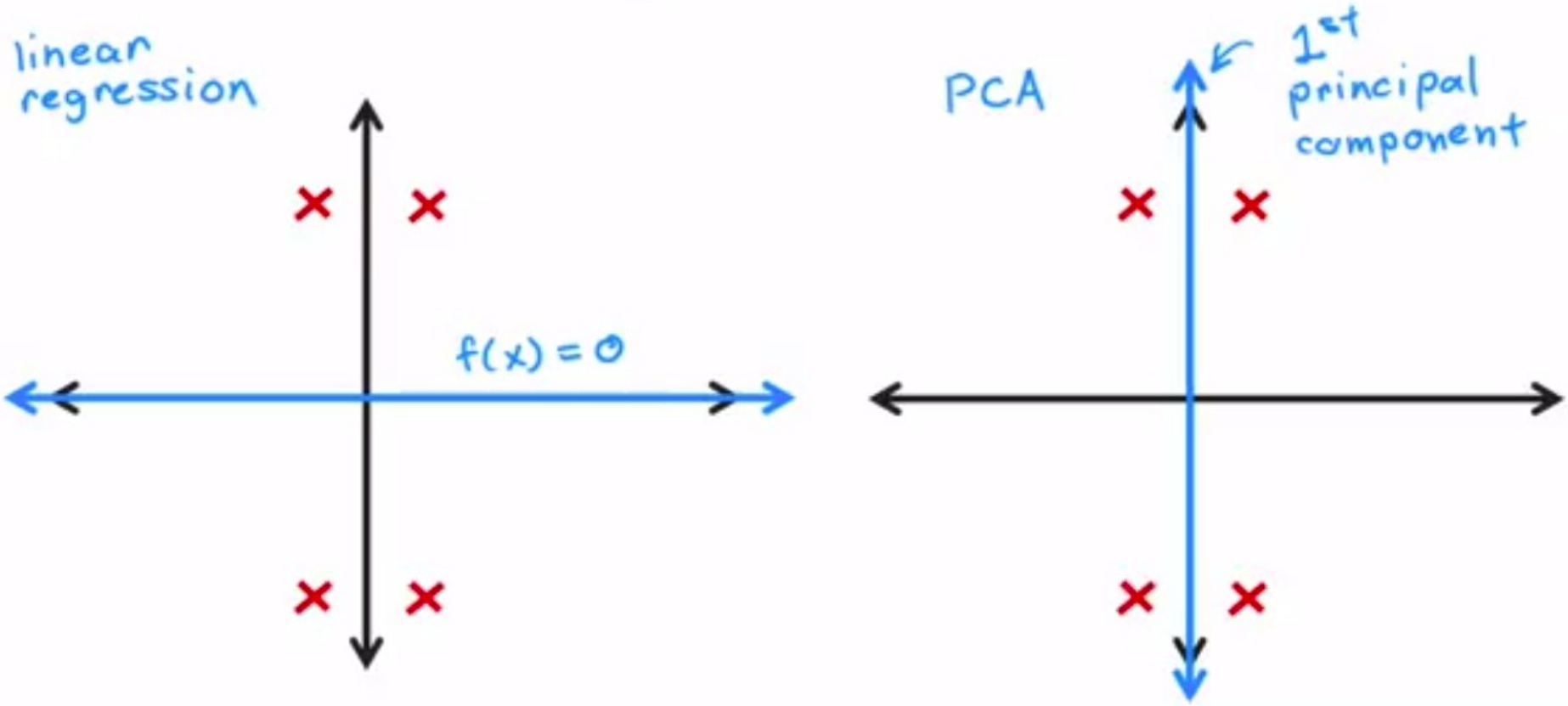


PCA is not linear regression

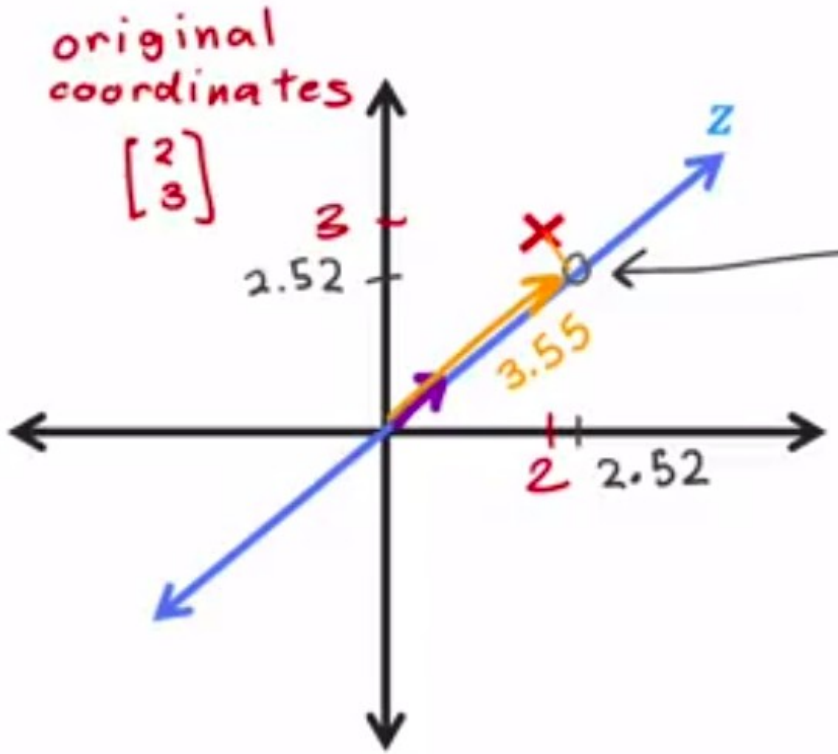




PCA is not linear regression



Approximation to the original data



given $z = 3.55$,

find original (x_1, x_2) (approximately)

"reconstruction"

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

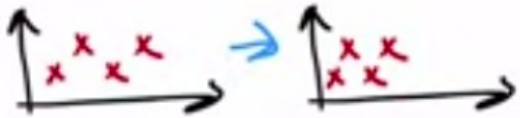
DeepLearning.AI

Stanford
ONLINE

PCA

PCA in Code

PCA in scikit-learn

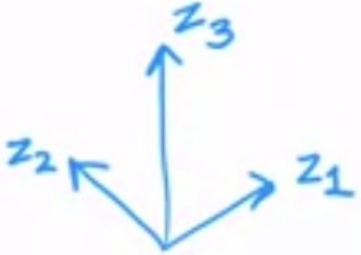


Optional pre-processing: Perform feature scaling

for visualization

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

   `fit` includes mean normalization



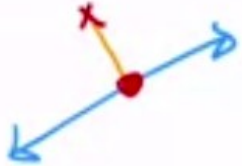
info

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

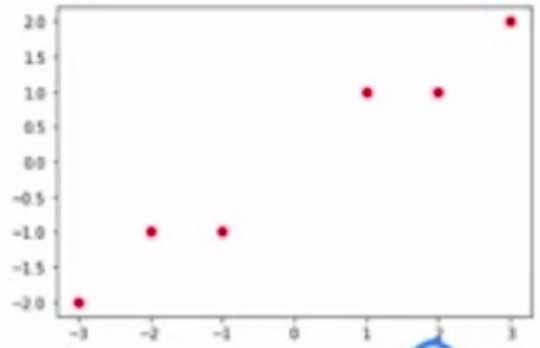
   `explained_variance_ratio`

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

   `transform`



Example

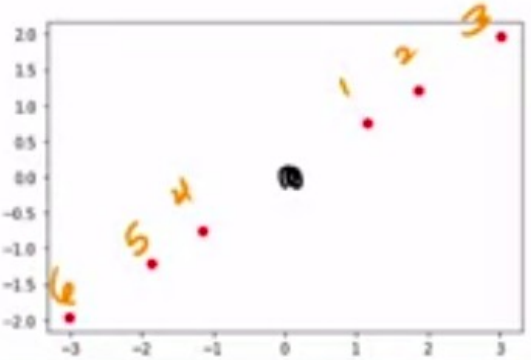


```
X = np.array([[1, 1], [2, 1], [3, 2],
              [-1, -1], [-2, -1], [-3, -2]])
```

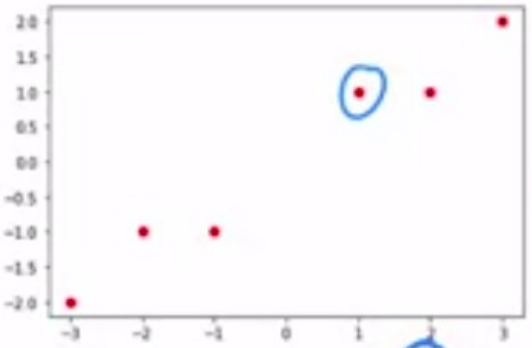
```
pca_1 = PCA(n_components=1)
pca_1.fit(X)
pca_1.explained_variance_ratio_
X_trans_1 = pca_1.transform(X)
X_reduced_1 = pca.inverse_transform(X_trans_1)
```

0.992

```
array([
 [ 1.38340578],
 [ 2.22189802],
 [ 3.6053038 ],
 [-1.38340578],
 [-2.22189802],
 [-3.6053038 ]])
```



Example



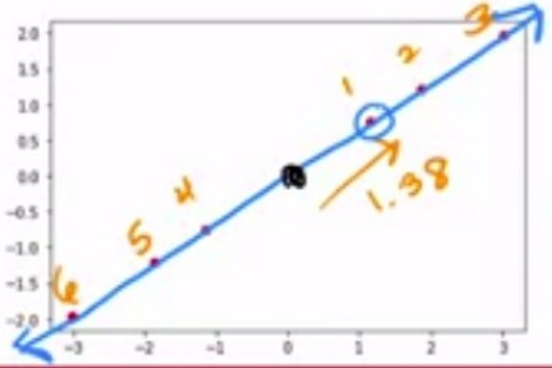
```
X = np.array([[1, 1], [2, 1], [3, 2],
              [-1, -1], [-2, -1], [-3, -2]])
```

2D

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

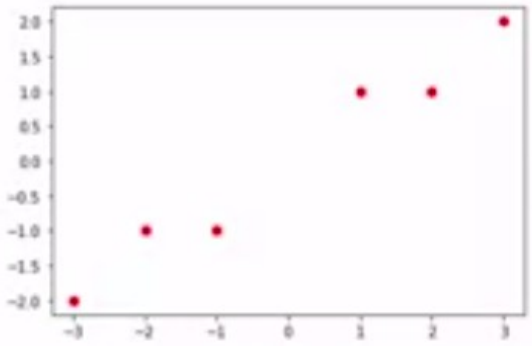
0.992

```
array([
 [ 1.38340578],
 [ 2.22189802],
 [ 3.6053038 ],
 [-1.38340578],
 [-2.22189802],
 [-3.6053038 ]])
```



1D

Example



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

2D

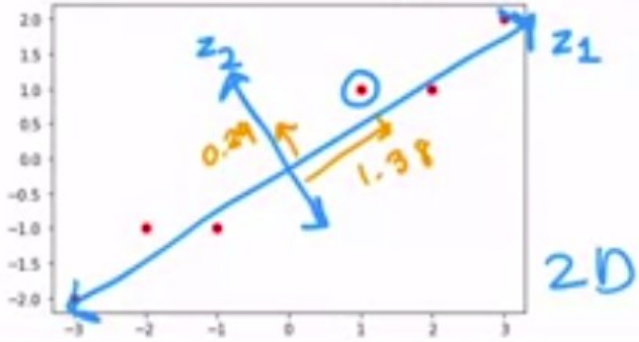
```
pca_2 = PCA(n_components=2)
pca_2.fit(X)
pca_2.explained_variance_ratio_
X_trans_2 = pca.transform(X)
X_reduced_2 = pca.inverse_transform(X_trans_2)
```

z_1 z_2

0.992 0.008

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]])
```



2D

Applications of PCA

- Visualization reduce to 2 or 3 features

Less frequently used for:

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

$n = 1000 \rightarrow 100$