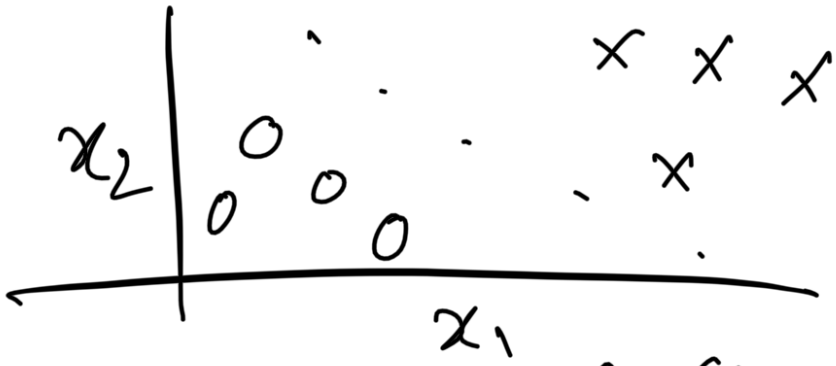
UNSupervised Learning

Clustering

Clustering
Anamoly
Recommender System
Reinforcement Learning

Clustering

Supervised

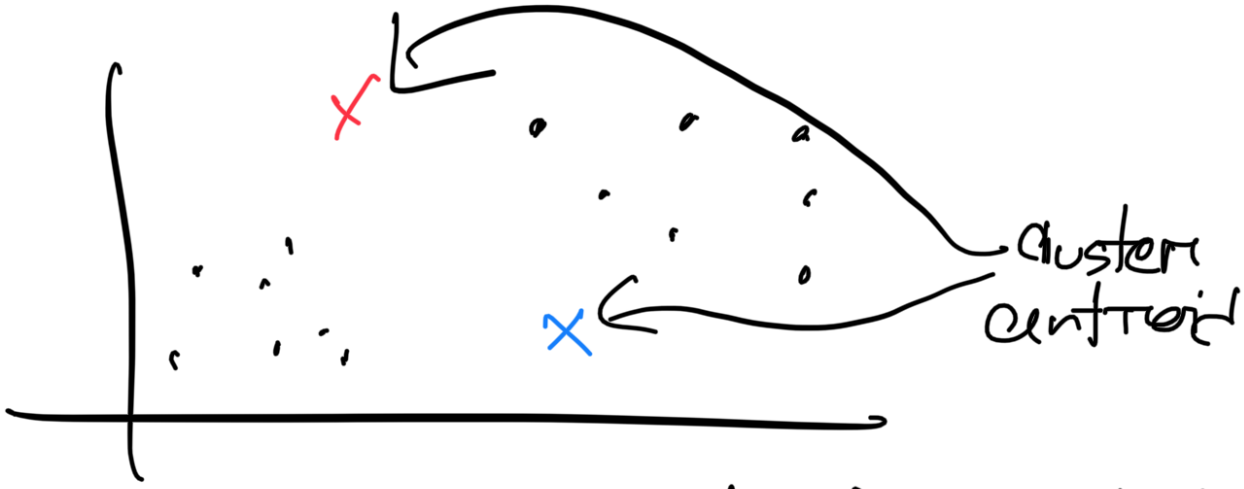


Training set : $\{x^{(1)}, x^{(2)}, x^{(3)} \ldots x^{(m)}\}$

Clustering

Growing Skills
Develop concept

Astronomical data analysis



Step 1: Assign each points to its closet centroids

Step 2: Recompute the centroids

Cluster Centroids

Randomly initialize K cluster $\mu_1, \mu_2, \ldots \mu_K$

Repeat {

Assign points to cluster centroids

for i = 1 to m

$c^{(i)}$ = index (1 to K) of cluster centroid cluster to $x^{(i)}$

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

μ_K

$(K = 2)$

$x^{(1)}, c^{(1)} = 1$

$x^{(12)}, c^{(12)} = 2$

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

Move Cluster centroids

for $k = 1$ to K

μ_k = avg (mean) of points assigned to cluster k

$\mu_1 = \frac{1}{4}[x^{(1)} + x^{(5)} + x^{(6)} + x^{(10)}]$

Optimization Objective

$c^{(i)}$ = index of cluster

μ_k = cluster centroid k

$\mu_{c^{(i)}}$ = " " of which example

Cost function

K-means algo

$c^{(i)}$ = index of clusters $\{1, 2, \ldots K\}$ $x^{(i)}$

example x is assigned

μ_k = cluster centroid k

$\mu_{c^{(i)}}$ = " " of cluster to which example $x^{(i)}$ assigned

Cost func

$$J(c^{(1)}, \ldots c^{(m)}, \mu_1, \ldots \mu_K) =$$

$$\min \frac{1}{m} \sum_{i=1}^{m} \| x^{(i)} - \mu_{c^{(i)}} \|^2$$

distortion

Initializing K-means

Step 0: Randomly initialize K cluster centroids $\mu_1, \mu_2 \ldots \mu_K$

Repeat {

Step 1: Assign points to cluster centroids,

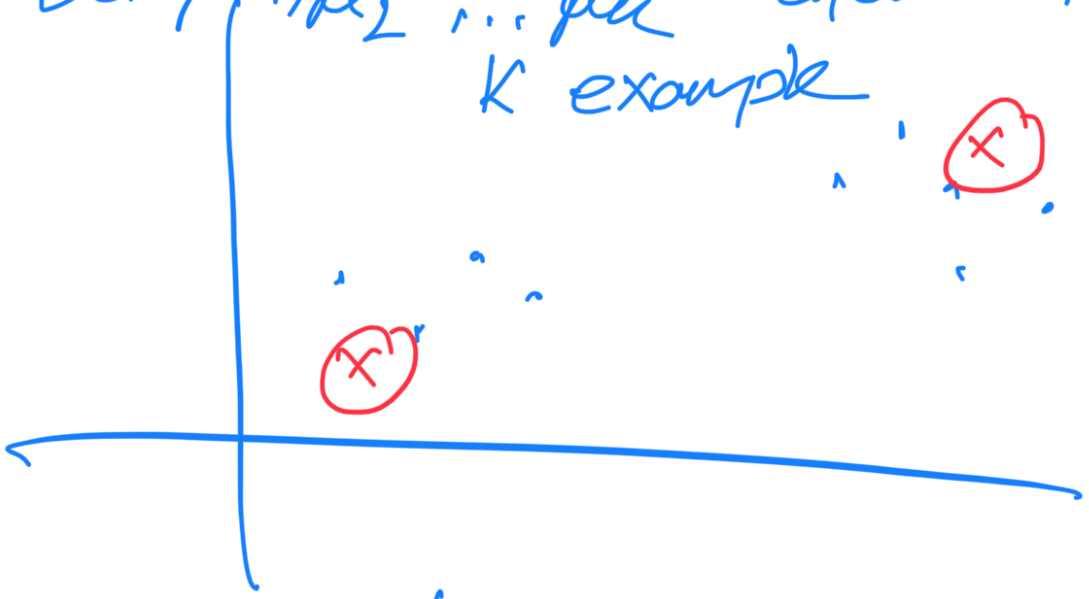
Step 2: Move cluster centroids

{

Choose $k < m$

Randomly pick k training examples

Set $\mu_1, \mu_2 \ldots \mu_k$ equal to these k examples



for $i = 1$ to 100 {

Randomly initialize k-means

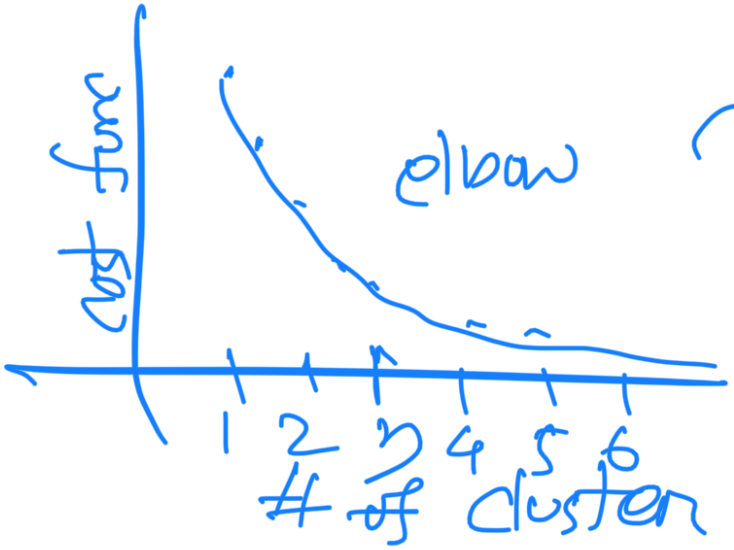
Run k means. Get $c^{(1)} \ldots c^{(m)}$, $\mu_1 \ldots \mu_k$

Compute cost function (distortion)

$J(c^{(1)}, \ldots c^{(m)}, \mu_1, \ldots \mu_k)$

}

K means algo

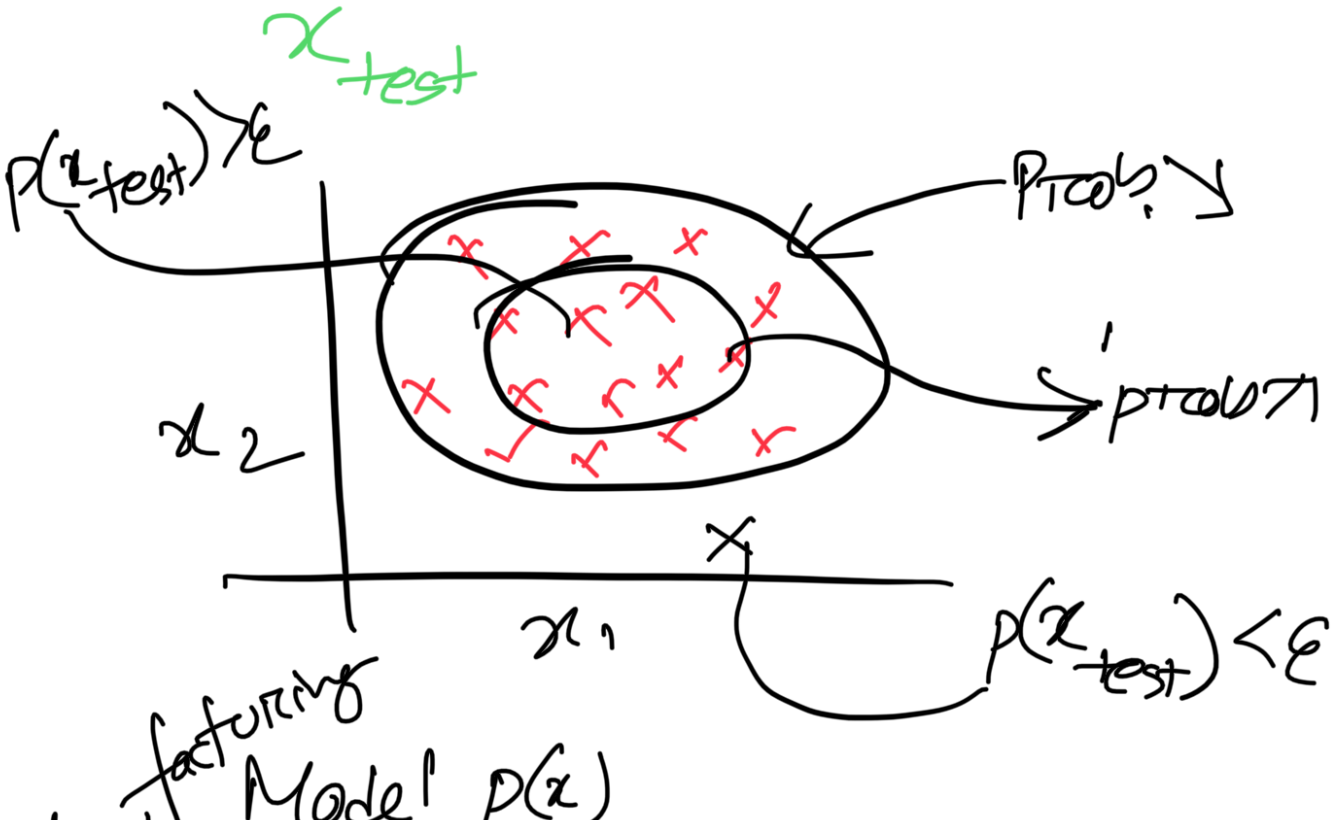


right "K"

ambigious

Anomaly detection

Datasheet : $\{x^{(1)}, x^{(2)}, \ldots, x^{(m)}\}$



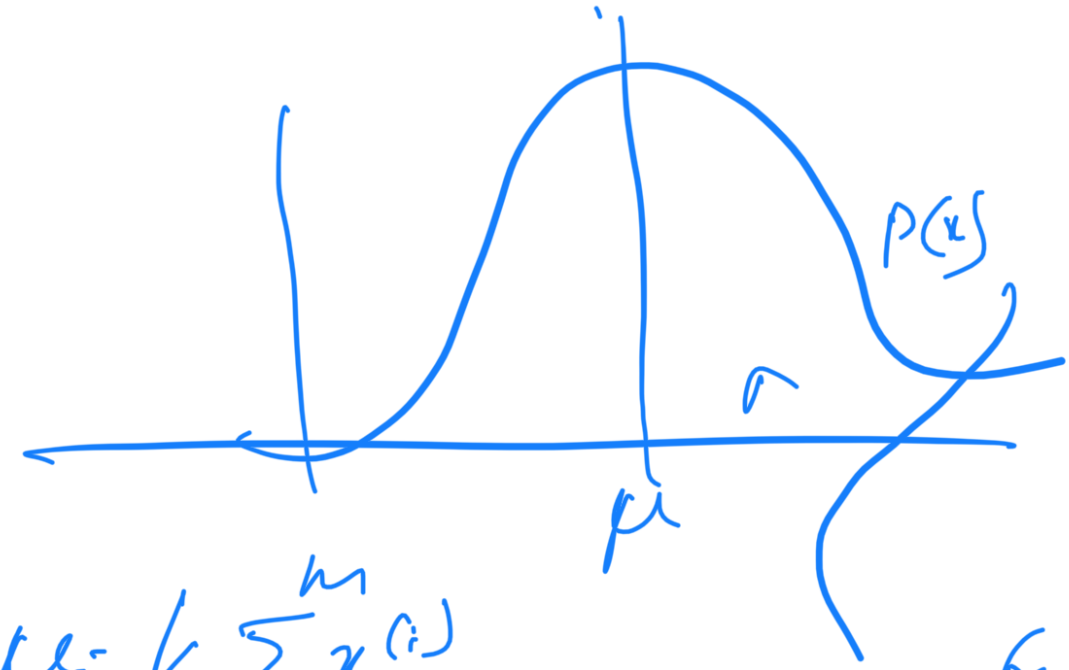
Many

How often log in?

how many web pages visited?

* Monitor

Gaussian distribution



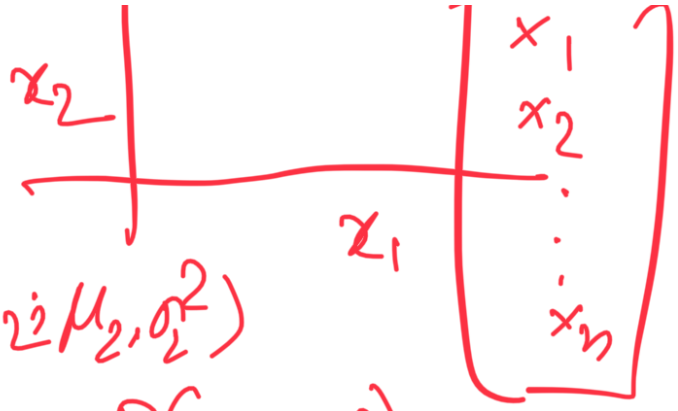
$$\mu = \frac{1}{m}\sum_{i=1}^{m} x^{(i)}$$

$$p(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-(x-\mu)^2}{2\sigma^2}}$$

$$\sigma^2 = \frac{1}{m}\sum_{i=1}^{m} (x^{(i)} - \mu)^2$$

Density estimation

Each example



$$P(\vec{x}) = P(x_1; \mu_1, \sigma_1^2) * P(x_2; \mu_2, \sigma_2^2) * P(x_3; \mu_3, \sigma_3^2) * \dots * P(x_n; \mu_n, \sigma_n^2)$$

$$= \frac{1}{10} \times \frac{1}{20} = \frac{1}{200}$$

$$= \prod_{j=1}^{n} P(x_j; \mu_j, \sigma_j^2)$$

1. choose n features → might have anomalous examples

② Fit

$$\mu_j = \frac{1}{m} \sum_{i=1}^{m} x_j^{(i)} \qquad \sigma_j^2 = \frac{1}{m} \sum_{i=1}^{m} (x_j^{(i)} - \mu_j)^2$$

3. Give new examples

$$P(x) = \prod_{j=1}^{n} P(x_j, \mu_j, \sigma_j^2) = \prod_{j=1}^{n} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x_j - \mu_j)^2}{2\sigma_j^2}\right)$$

if $P(x) < \epsilon$

Evaluating a system

previously observed anamalous
$y=1$
non-anamalous
$y=0$

include a few anamalous data
10k good engine 20 flawed

Train 6000 good
CV 2000 " 10 anamolys
Test " 10 "

fone $\in$ fone $\notin$

Algo evaluation

$$y=\begin{cases} 1 & p(x)<\varepsilon \text{ (anamoly)} \\ 0 & p(x)\geq\varepsilon \text{ (Normal)} \end{cases}$$

metric use for performance

Anamoly detection	Supervised Learning
Very Small ⊕	large number ⊕ &

ex.

very large (-) ex.

diff types of anamolies. Hard of algo to learn future anamolies are different

fraud

Manufacturing

new unseen defects

Monitoring machines in data

(+) examples

Enough (+) examples to get a sense of what positive ex. looks like.

Spam

finding scan defects

Weather prediction

Disease classification

Carefully choosing feat

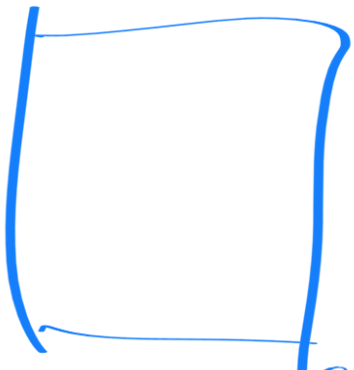
Non-Gaussian features

plt.hist()
np.log(x)

$x_1 \leftarrow \log(x_1)$
$x_2 \leftarrow \log(x_2+1)$
$x_3 \leftarrow \sqrt{x_3} = x_3^{1/2}$
$x_4 \leftarrow x_4^{1/3}$

Recommend System

Predicting movie rating



n_u = # of users
$r(i,j)$ = 1 if user j has rated movie i (i → # movie, j → # user)
$y^{(i,j)}$ → rate

features of movie $n_u = 4$, $n_m = 5$, $n = 2$

$w^{(1)} = \begin{bmatrix} 5 \\ 0 \end{bmatrix}$ $b^{(1)} = 0$ $x^{(3)} = \begin{bmatrix} 0.99 \\ 0 \end{bmatrix}$ $x^{(1)} = \begin{bmatrix} 0.99 \\ 0 \end{bmatrix}$ $x^{(2)} = \begin{bmatrix} 0.99 \\ 0 \end{bmatrix}$

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

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

↳ features

Cost function

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

$$+ \frac{\lambda}{2m^{(j)}} \sum_{k=1}^{n} (w_k^{(j)})^2$$ ← # of featuring

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

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

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

$$w^{(2)} = \begin{bmatrix} 5 \\ 0 \end{bmatrix} \qquad w^{(4)} = \begin{bmatrix} 0 \\ 5 \end{bmatrix}$$

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

$$\left.\begin{array}{l} w^{(1)} \cdot x^{(1)} \simeq 5 \\ w^{(2)} \cdot x^{(2)} \simeq 5 \end{array}\right\} \quad x^{(1)} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

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

$$J\left(x^{(1)}, x^{(2)} \ldots x^{(n_m)}\right) = \frac{1}{2} \sum_{i=1}^{n_m} \sum_{j:r(i,j)=1}$$

$$\cdots \left(w^{(j)} \cdot x^{(i)} + b^{(j)} - y^{(i,j)}\right)^2 + \lambda/2 \sum_{i=1}^{m_m} \sum_{k=1}^{n} \left(x_k^{(i)}\right)^2$$

Put them together

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

Linear regression

items liked by users click on it

Predict $y^{(i,j)} \longrightarrow w^{(j)} \cdot x^{(i)} + b^{(j)}$

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

$$g(z) = \frac{1}{}$$

binary $\qquad 1 + e^{-z}$

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

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

$$\log\left(1 - f_{(w,b,x)}(x)\right)$$

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

Mean Normalization

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

$$\frac{\lambda}{2} \sum_{j=1}^{n_u} \sum_{k=1}^{n} \left(w_k^{(j)}\right)^2 + \frac{\lambda}{2} \sum_{i=1}^{n_m} \sum_{k=1} \left(x_k^{(i)}\right)^2$$

mean normalization

$$\begin{bmatrix} 5 & 5 & 0 & 0 & ? \\ 5 & ? & ? & 0 & ? \\ ? & 4 & 0 & ? & ? \end{bmatrix} \begin{bmatrix} 2.5 \\ 2.5 \\ 2 \end{bmatrix} \qquad \mu = \begin{bmatrix} 2.5 \\ 2.5 \\ 2 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 0 & 5 & 4 & ? \\ 0 & 0 & 5 & 0 & ? \end{bmatrix} \begin{bmatrix} 2.25 \\ 1.25 \end{bmatrix} \quad \begin{bmatrix} 2.25 \\ 1.25 \end{bmatrix}$$

$-\mu$

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

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

for user 5

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

$w^{(5)} \cdot x^{(1)} + b^{(5)} + \mu_1 = 2.25$

Tensorflow implementation

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

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

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

Gradient descent algorithm

Custom Training Loop

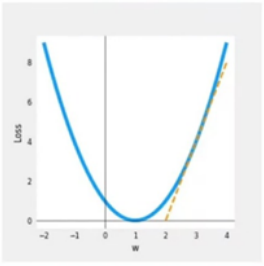
```
w = tf.Variable(3.0)
x = 1.0
```

Tf.variables are the parameters we want to optimize

Repeat until convergence

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

Fix b = 0 for this example




```
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.
```

And if you write this with tf
our gradient tape as tape f.

DeepLearning.AI Stanford ONLINE Andrew Ng

3:50 / 11:37

Custom Training loop

↳ Auto diff / Auto grad

Implementation in TensorFlow

Gradient descent algorithm

Repeat until convergence

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

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

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

```
# 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 Andrew Ng

Collaborative filtering

To find other items related to

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

$$\sum_{i}^{n} \left(x_j^{(k)} - x_j^{(i)}\right)^2 \quad x^{(k)} \; x^{(i)}$$

Cold start

↳ rank new items

↳ show something reasonable

User side info...

Item: Genre, movie stars, studio

User: Demographics (age, gender

Collaborative filtering

Recommend item → based on user rating

Similar to you ↵

Content based filtering

Recommend items → features of users & item → Good Match

New tools to return the best points if you go back & re watch something

User features:
- Age
- Gender
- Country
- Movie watched
- Average rating

$x_v^{(j)}$ for user j

Movie features
- Year
- Genre
- Reviews
- Average rating

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

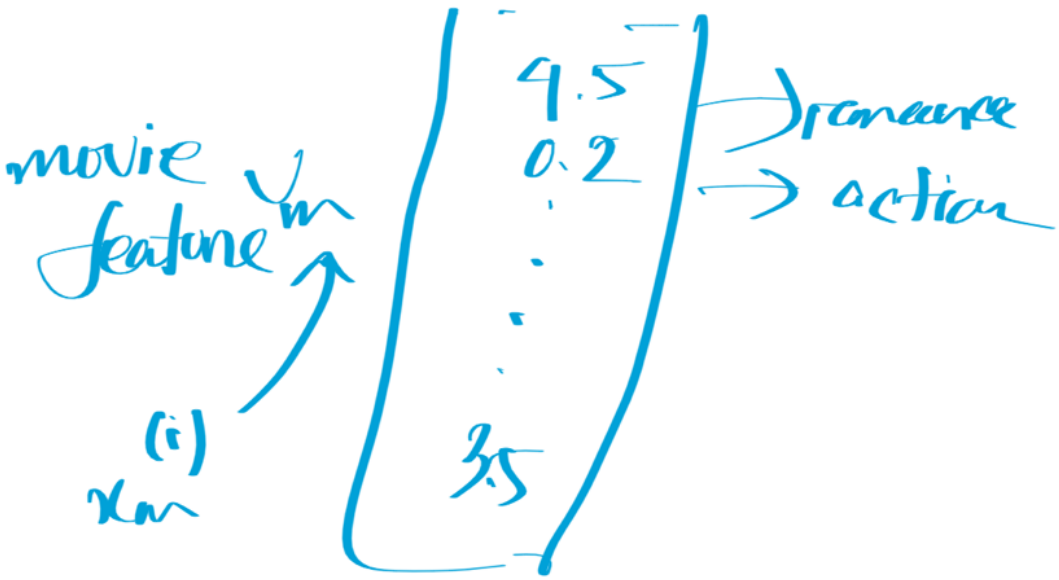
$w^{(j)T} x^{(i)} + b^{(j)}$

$v_v^{(j)}$ — user's preference

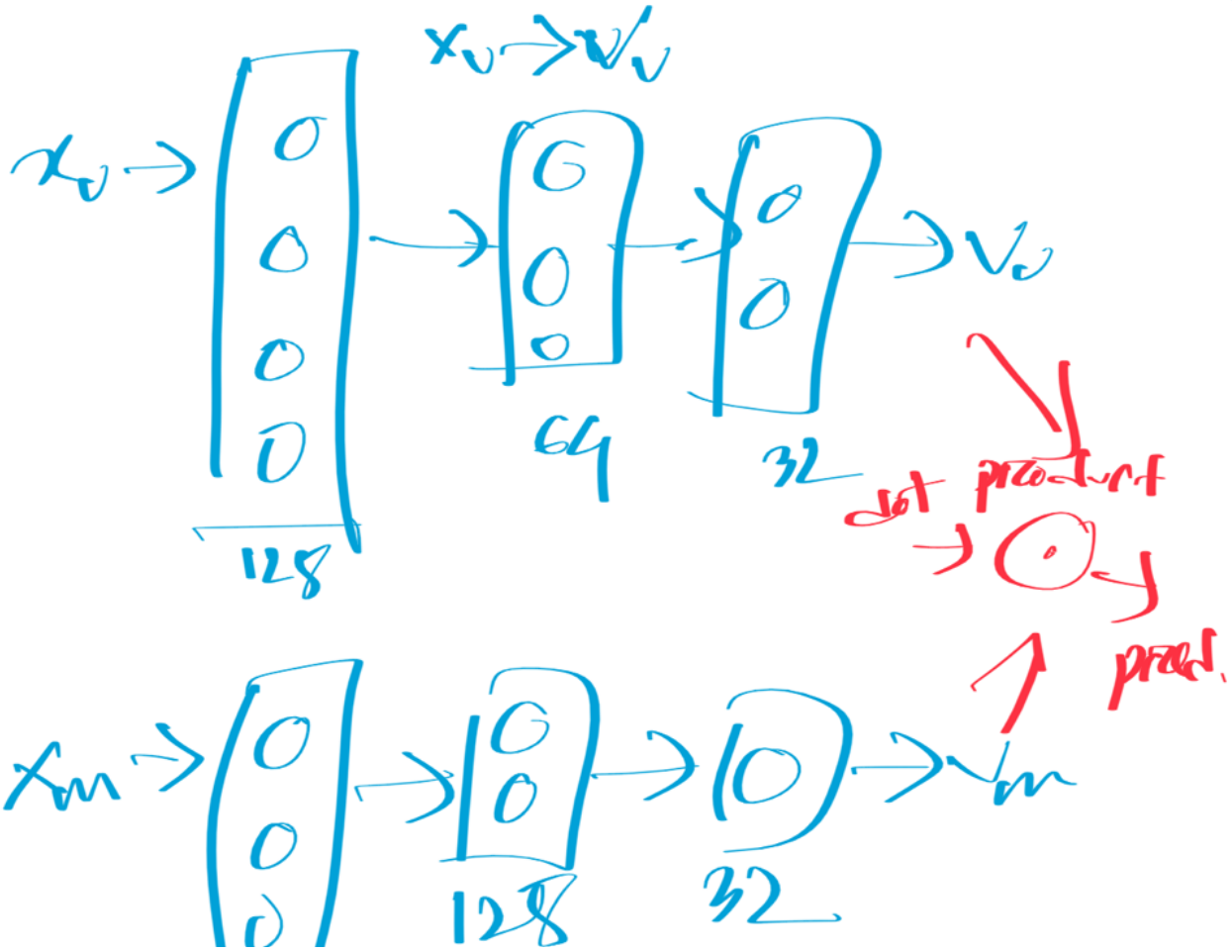
$v_m^{(i)}$

$x_v^{(j)} \Rightarrow v_v^{(j)} = \begin{bmatrix} 4.9 \\ 0.1 \\ \vdots \end{bmatrix}$

Computed from $x_u^{(j)}$) Computed from $x_m^{(i)}$



Neural Network architecture



256

$g(v_u^{(j)}, v_m^{(i)}) \rightarrow$ predict the probability $y^{(i,j)} = 1$

Cost

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

regularization

$v_0^{(j)} \rightarrow 32$ length $\rightarrow$ user j features $x_0^{(j)}$

$v_m^{(i)} \rightarrow 32$ " $\rightarrow$ movie i features $x_m^{(i)}$

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

efficiently find recommendation from a large set

2 Steps
Retrieval
large list of items

Ranking
rank the list suggest very similar five

→ 10 movies → 10 most similar

$||v_m^{(k)} - v_m^{(i)}||_2$

→ 3 genre → Suggest 10
→ top 20 → in the country

Display the ranks

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

$\frac{1}{2} \sum_{k=1}$

Collaborative filtering

Recommend items ⟶ rating of users gave similar rating as you

Conten-based ⟶ feature of users & item to find good fit

$r(i,j)=1$ ⟶ User j has rated item i

$Y(i,j)$ rating given by j on item i

User features

↓

Age/gender/Country/Movie watched/ Avg. rating per genre

$x_j^{(j)}$ for user

Movie features

Year/genre/genres/Review/Avg rating

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

$x_m \neq x_u$ (Size not be same)

Learn to match

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

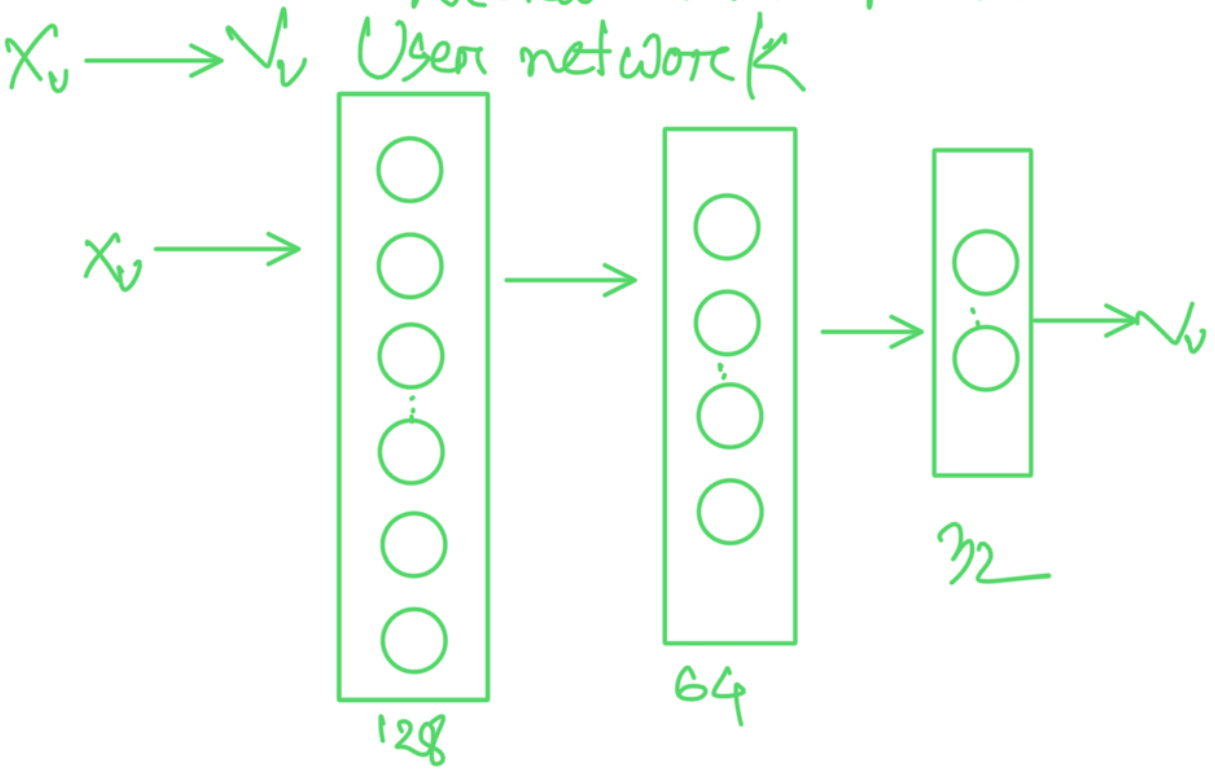
$v_u^{(j)}$ $\quad$ $v_m^{(i)}$

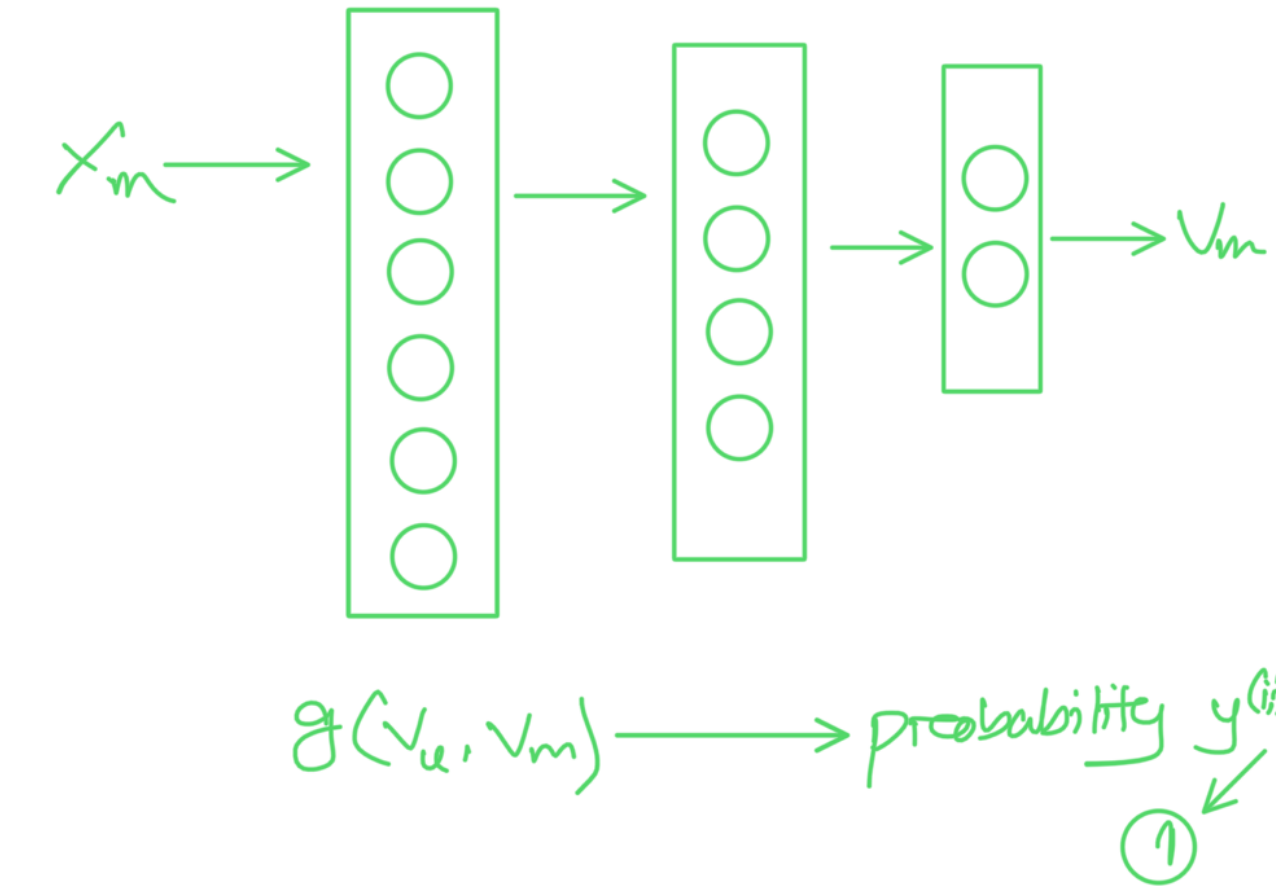
$x_u^{(j)}$ $\quad$ movie $x_m^{(i)}$

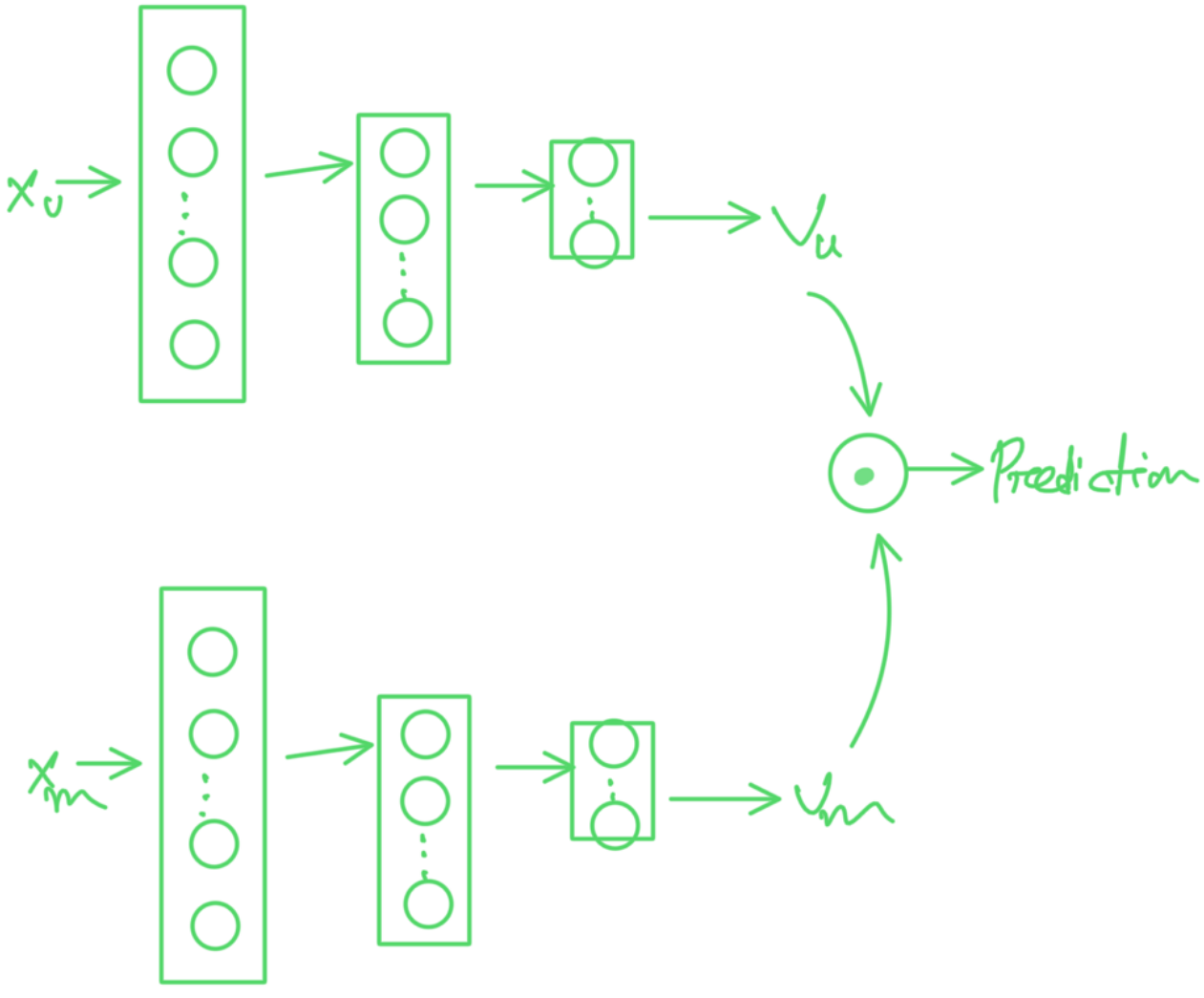
Content Based filtering

Neural Network architecture

$x_u \rightarrow v_u$ User network




x_m
v_m
$g(v_u, v_m)$
probability $y^{(ij)}$
1


x_u
v_u
Prediction
x_m
v_m

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

To find movies similar to movie i:

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

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

Recommending from large Dataset

Retrieval & Ranking

① for each 10 watched movie find 10 $||v_m^{(k)} - v_m^{(i)}||^2$

② most viewed 3 genre → top 10

③ Top 20 movies in the country

Combined retrieved item → remove duplicates or already watched

Ranking → Based on predicting → Display ranks

Another procedure → do Vm then take Vc & do dot product

real time → Retrieval steps

Retrieve more → performance → recommend

to optimize → carry offline exp.

retrieve additional items more relevant recommend

$P(y^{(i,j)}) = 1$

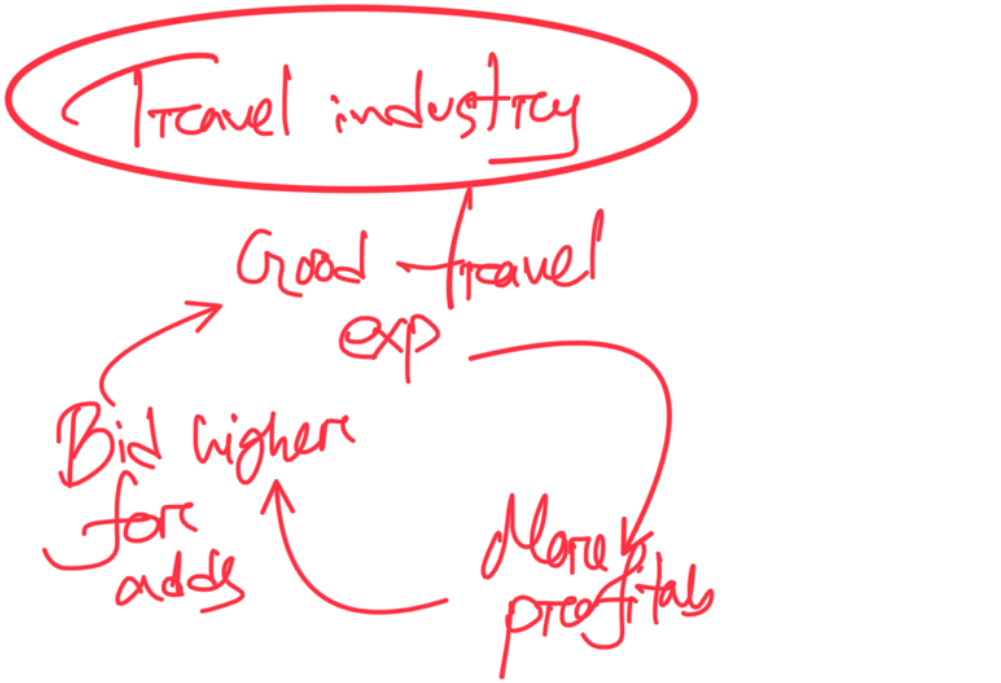
Movies → 5☆

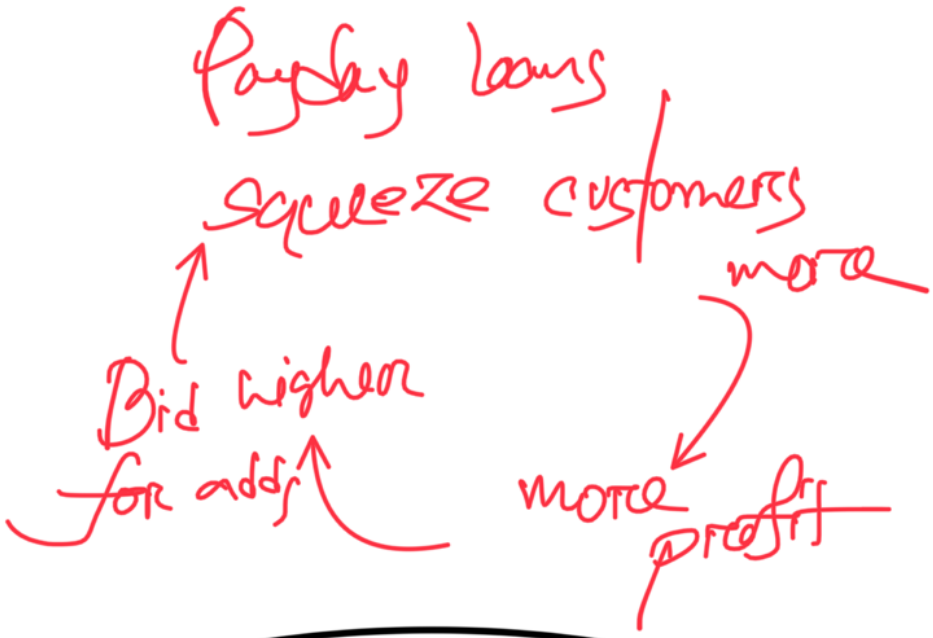
products most likely purchased

Ads → clicked ⊕ high bid

Products → largest profit

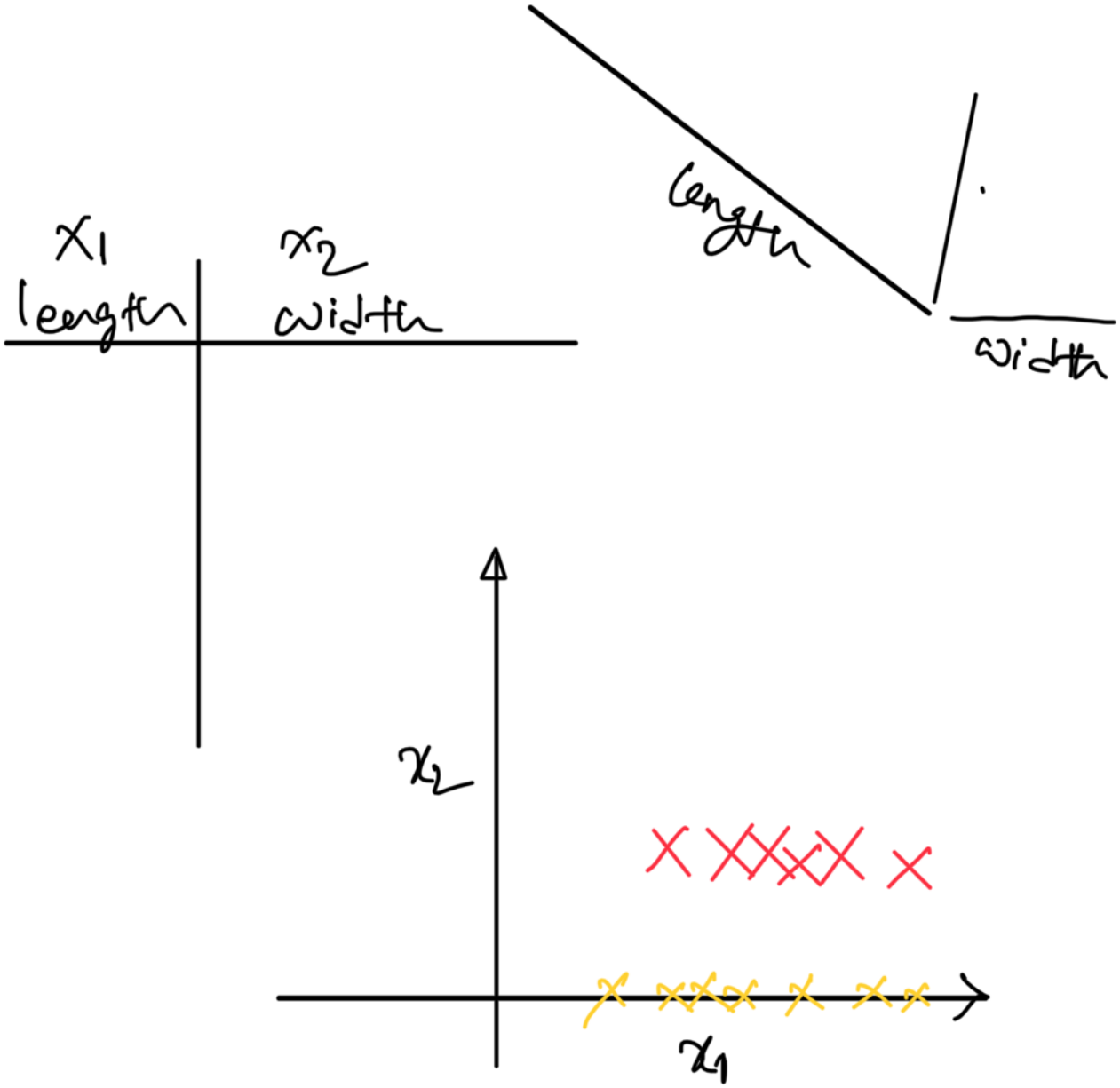
Video → max watched time





Principle Component Analysis

Car Measurement

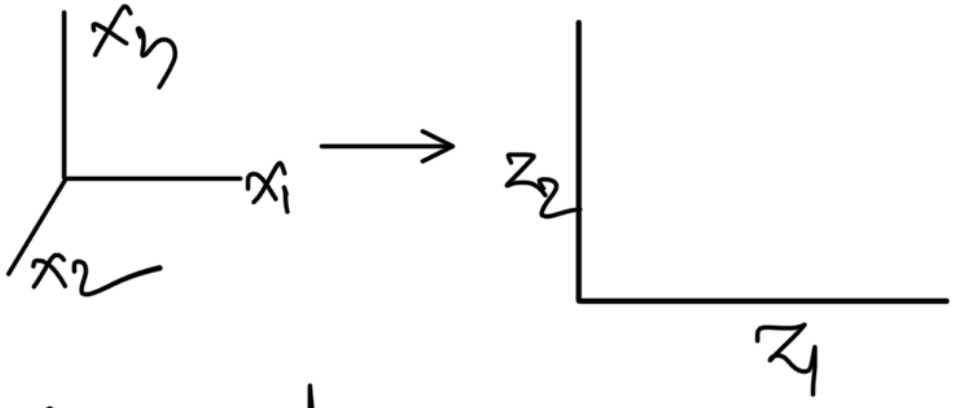

x_1
length
x_2
width
length
width
x_2
x_1
x_1 (length)
x_2 (height)
x_2
height
Z axis

find/find new axis & coordinates
use fewer numbers
to capture "size" feature

$2 \rightarrow 1$

$1000 \rightarrow 1$

$3D \rightarrow 2D$



life expectancy

50 features $\rightarrow$ 2 features

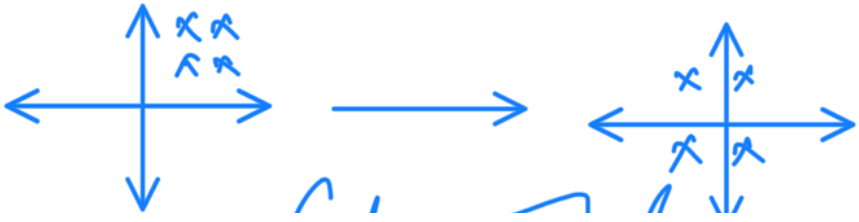
50 D $\rightarrow$ 2 D

---

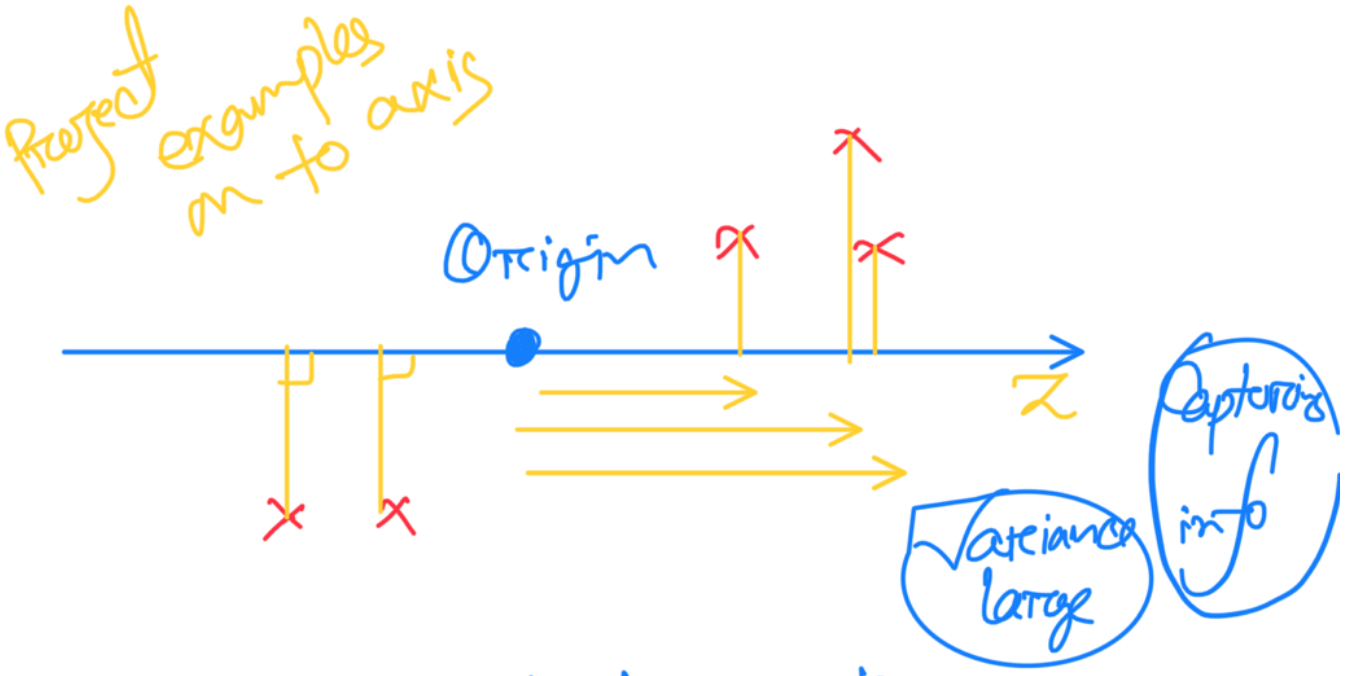
z $\rightarrow$ axis

PCA Algorithm

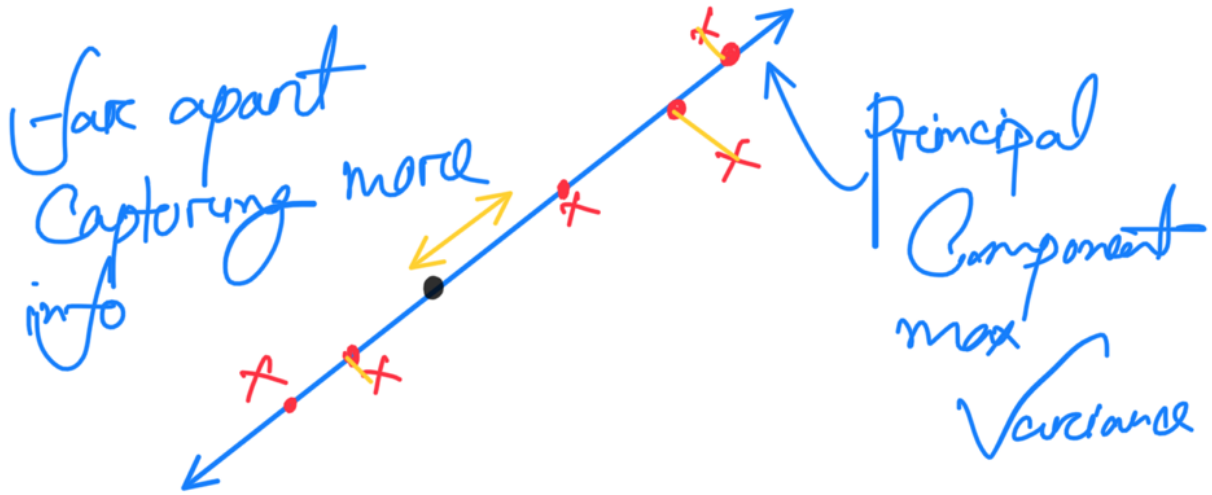
Normalized to have zero mean



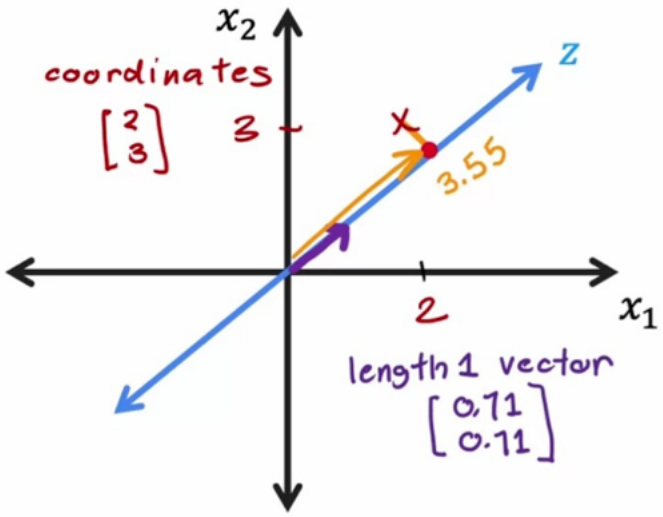
Feature Scaling



Squished x axis



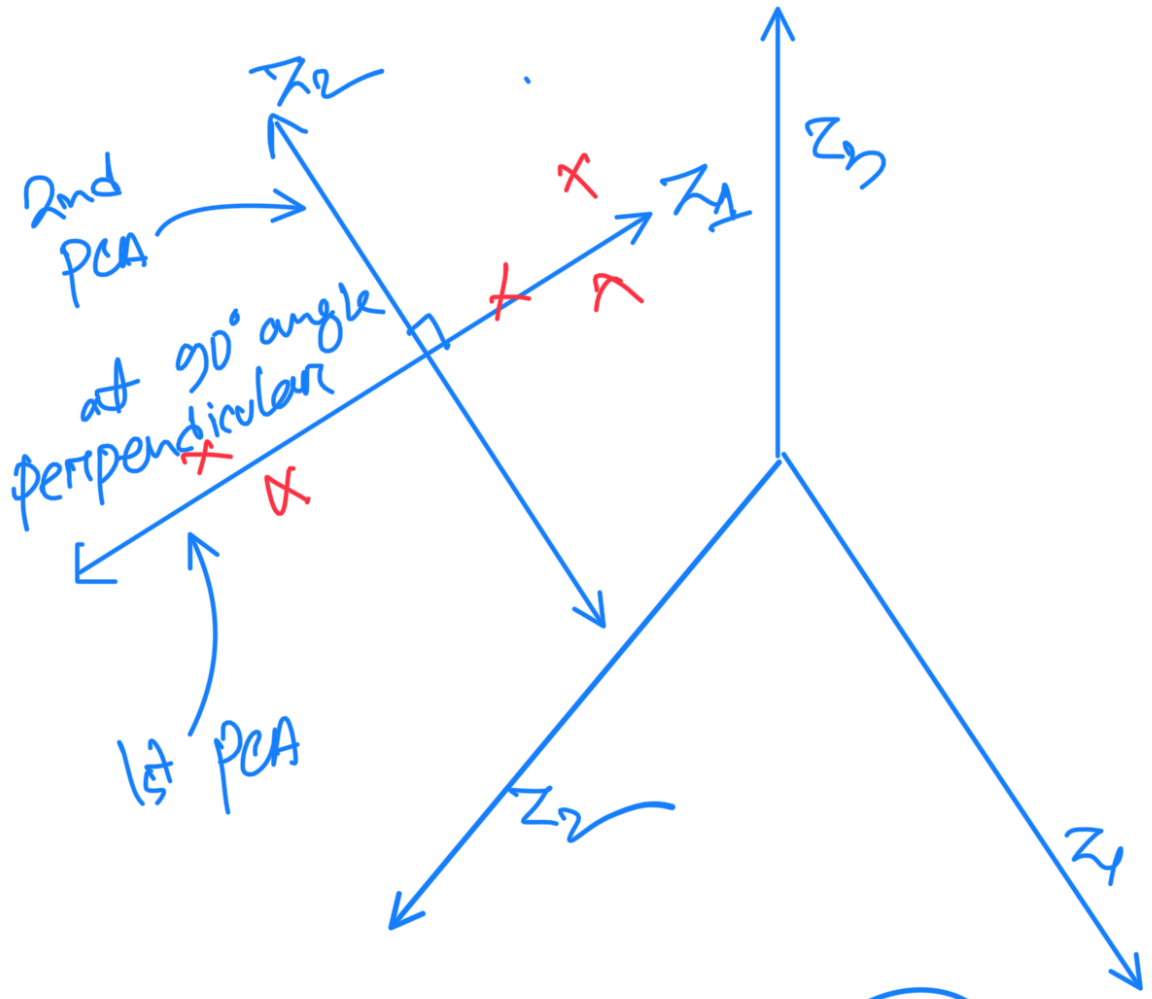
Coordinate on the new axis



dot product

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

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



linear regression
minimize distance along y axis

PCA

$x_1, x_2, \ldots, x_{50}$

find axis to retain variance (info)

reconstruction

$z = 3.55$

find original (x_1, x_2) approximately

$$z \times \begin{bmatrix} 0.71 \\ \end{bmatrix} \quad \begin{bmatrix} 0.72 \\ \end{bmatrix}$$

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

---

preprocessing : perform feature scaling

① fit the data to obtain 2(or 3) new axis
(principle components)
include mean normalization

② Originally examine how much variance is explained by each (PCA)

explained_variance_ratio

③ Transform (project) the data onto new axis

transform

Application
visualization
Data compression
(to reduce storage or transmission costs)
Speed of training data (Supervised learning)