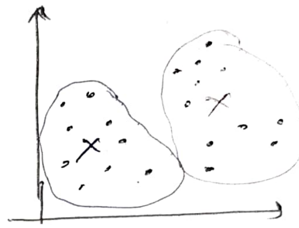
Course

Unsupervised learning

→ Clustering

applicatⁿ → market segmentatⁿ, grping similar news
Astronomical data analysis, DNA analysis

• K-means Algorithm



(x_1, x_2, x_3, x_4 ----)
unlabelled data

cluster centroid

→ calculate distance of each pt from random initially guessed ~~to~~ centroid.

→ ~~to~~ which ever is nearest to the centroid, mark in its ~~dom~~ ~~grop~~ group.

→ calculate average of pt. and then update the centroid location with that average.

→ repeat the process, you will ~~tand~~, land clustering the data in 2 groups.

until no further change in centroid position.

• Algorithm

Randomly initialize K cluster centroids $\mu_1, \mu_2 \cdots \mu_K$

~~Reap~~

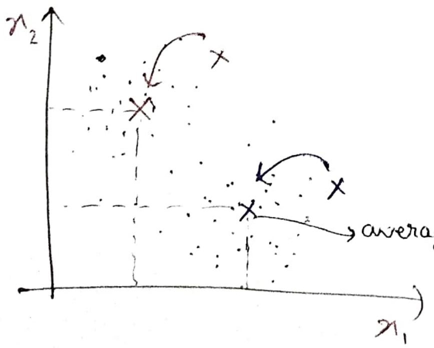
Repeat {

\# Assign points to cluster centroids

for $i = 1$ to m

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

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

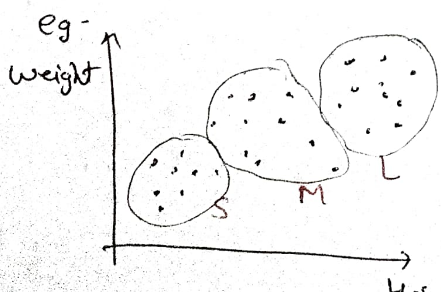


\# move cluster centroids

for $K = 1$ to K

μ_K = average (mean) points assigned to cluster K.

}



→ K-means optimizat^n objective

$c^{(i)}$ = index of cluster $(1, 2, \ldots K)$ to which example $x^{(i)}$ is currently assigned.

μ_K = cluster centroid K

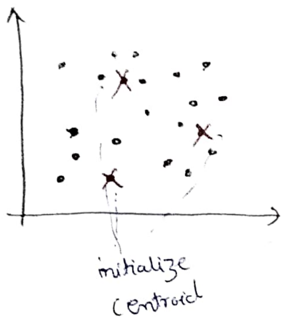
$\mu_{c^{(i)}}$ = cluster centroid of cluster to which example $x^{(i)}$ has been assigned.

Cost funct^n

$$J(c^1, \ldots c^m, \mu_1, \ldots, \mu_K) = \frac{1}{m} \sum_{i=1}^{m} |x^{(i)} - \mu_{c^{(i)}}|^2$$

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

→ K-means initializat^n



Choose $K < m$ — no. of grp to cluster in

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 $c^{(1)}, \ldots, c^{(m)}, \mu_1, \mu_2 \ldots \mu_K$

Computer cost function (distort^n)

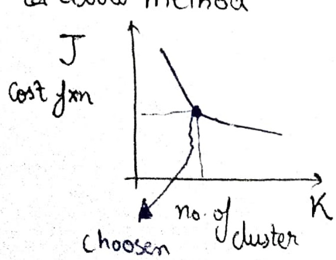
$J(c^{(1)}, \ldots, c^{(m)}, \mu_1, \mu_2 \ldots \mu_K)$

}

Pick set of clusters that gave lowest cost

Choosing value of K (no. of cluster)

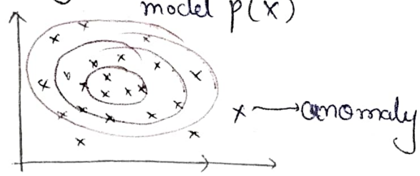
- Elbow method



→ Anomaly Detection Example :

→ Density Estimat^n

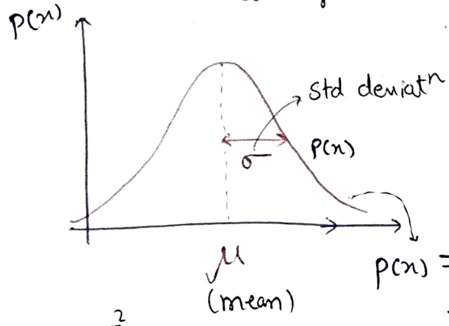
model $p(x)$



$p(x_{test}) < \varepsilon$ ~ Anomaly

$p(x_{test}) \geq \varepsilon$ ~ OK ✓

* → Gaussian Distribution
Normal —"—
Bell-shaped —"—



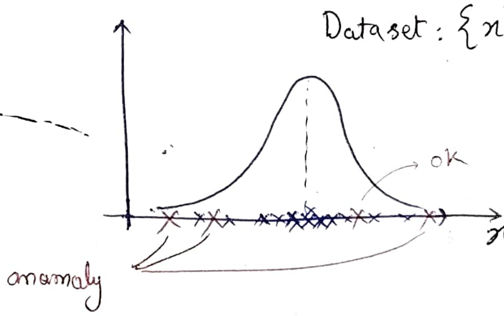
σ^2 → variance

area under curve = 1

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

→ Parameter estimat^n

Dataset : $\{x^1, x^2, \ldots x^m\}$



{→ Algorithm}

$p(x) \neq p(x)$

① • Choose n features x_i that might be anomalous

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

$$p(x) = p(x_1, \mu_1, \sigma_1^2) * p(x_2, \mu_2, \sigma_2^2) * \cdots\cdots p(x_n, \mu_n, \sigma_n^2)$$

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

② • fit parameters, $\mu_1 \ldots \mu_n$, $\sigma_1^2 \ldots \sigma_n^2$

$$\vec{\mu_j} = \frac{1}{m}\sum_{i=1}^{m} x_j^{(i)} \quad \begin{bmatrix} \mu_1 \\ \mu_2 \\ \vdots \\ \mu_n \end{bmatrix}, \quad \sigma_j^2 = \frac{1}{m}\sum_{i=1}^{m}\left(x_j^{(i)} - \mu_j\right)^2$$

$p(x) \Leftarrow$

③

$\varepsilon = 0.02$

$p(x^1_{test}) = 0.0426$ ⟶ OK ✓

$p(x^2_{test}) = 0.002$ ⟶ anomaly

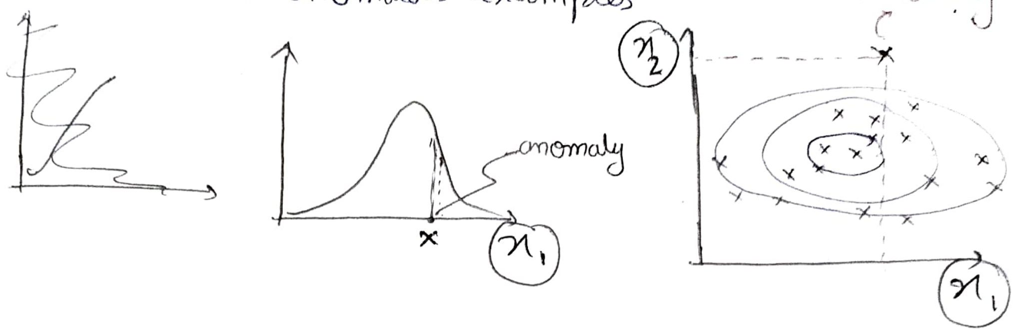
Error analysis in Anomaly detection

↳ $p(x) \geq \varepsilon$ large for normal examples x.

$p(x) < \varepsilon$ small for anomalous examples x

Most Common problem:-

$p(x)$ is comparable. (say, both large) for normal and anomalous examples



Recommender Systems

→ collaborative filtering

eg:- movie ratings

movie (i ↓, j →)	Alice (1)	Bob (2)	___ (3)	___ (4)	x_1 romance	x_2 action
love at last (1)	5	5	0	0	0.9	0
___ (2)	5	?	?	0	1.0	0.01
___ (3)	?	4	0	?	0.99	0
___ (4)	0	0	5	4	0.1	1.0
___ (5)	0	0	5	?	0	0.9

$r(i,j) = 1$ if user j has rated movie 'i'

$y^{(i,j)}$ = rating given by user j on movie i (if defined) i.e if $r^{(i,j)} = 1$

$w^{(j)}, b^{(j)}$ → 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 $w^{(j)}, b^{(j)}$ for user j :-

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

→ To learn parameter for all $w^{(1)}, b^{(1)}, w^{(2)}, b^{(2)}, w^{(3)}, b^{(3)} \ldots\ldots w^{(n_u)}, b^{(n_u)}$

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

→ To learn $x^{1}, \ldots x^{(n_m)}$:

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

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

Gradient Descent

repeat {

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

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

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

}

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.

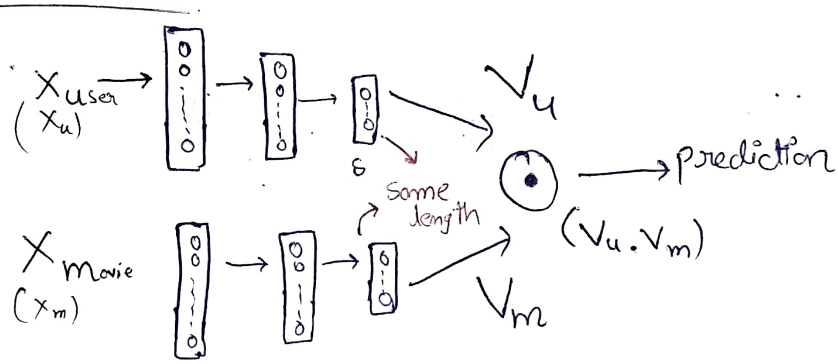
predict rating of user 'j' on movie 'i' as

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

$$V_u^{(j)} \cdot V_m^{(i)}$$

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

N-N architecture



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

$v_u^{(j)}$ - is a vector of length 32 that describes user j with feature $x_u^{(j)}$

$v_m^{(i)}$ - is a vector of ~~long~~ length 32 that describes movie i with features $x_m^{(i)}$

To find movies similar to movie i :

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

($v_m^{(k)}$ → other movie k)

for large set of items

Two steps : Retrieval & Ranking

* Retrieval :
    - Generate large list of plausible item candidates.

8.43 × 49 + 22 × x = 9 × (22 + 49)

eg:-

1) for each of the last 10 watched movies by ~~user~~ user find 10 most similar movies

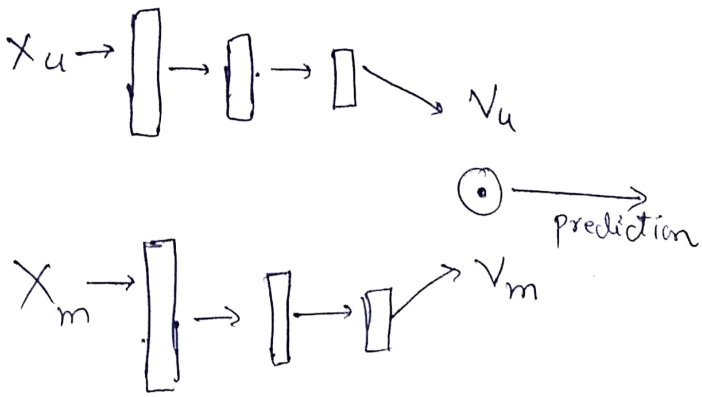
$$\cancel{\|v_k^{(i)}} \quad \|v_m^{(k)} - v_m^{(i)}\|^2$$

2) For most viewed 3 genres, find the top 10 movies

3) Top 20 movies in Country

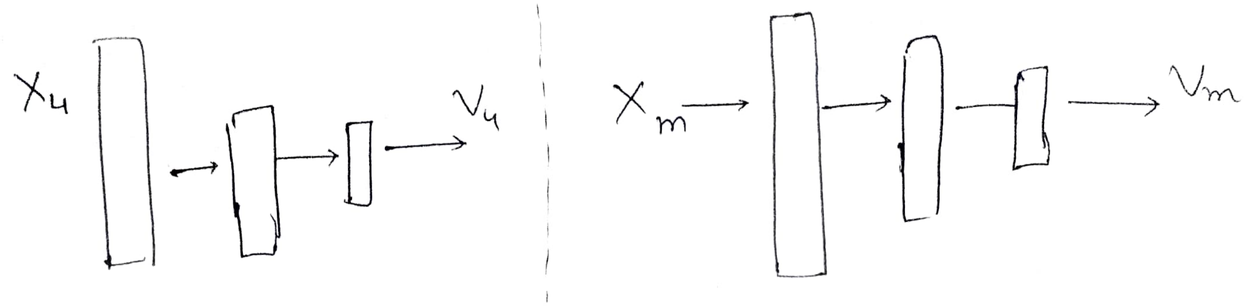
- Combine retrieved items into list, removing ~~dy~~ duplicates and items ~~already~~ already watched/purchased.

* Ranking:
    - Take list retrieved and rank using learned model



- Display ranked items to user

Tensorflow Implementation




```
user_NN = tf.Keras.models.Sequential([
            tf.Keras.layers.Dense(256, activation='relu'),
            ————''————————( 128 ——————————)
            ————''————————( 32 ——————————)
          ])

item_NN = tf. ...  ————''————
          ————''————  (256 ————————
          ————''————  (128 ————''————
          ————''————  (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)

# ——''———————''———————''———————

input_item = ———''————(shape=(num_item_features))
vm = ———''———
vm = ——''—— (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()
```