

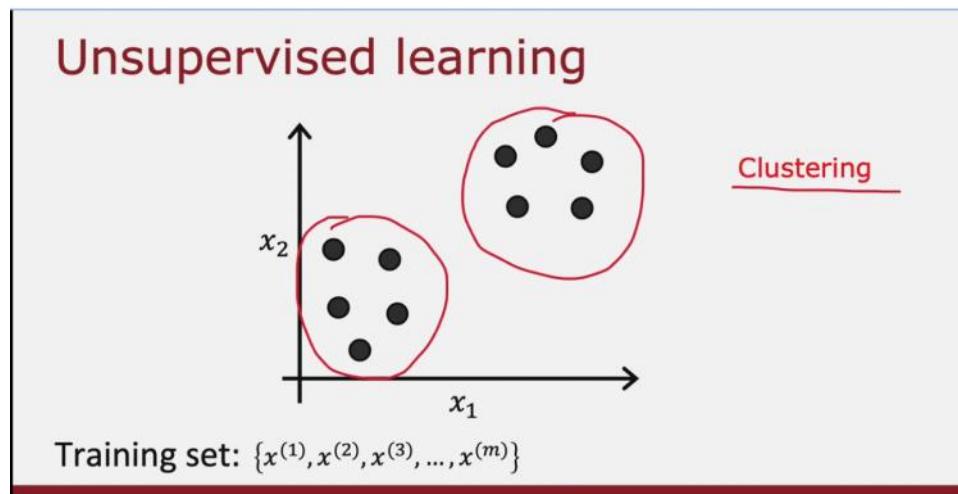
What Is Clustering? (Unsupervised...Applications Of)

Monday, 11 December 2023 05:37

When doing unsupervised learning:

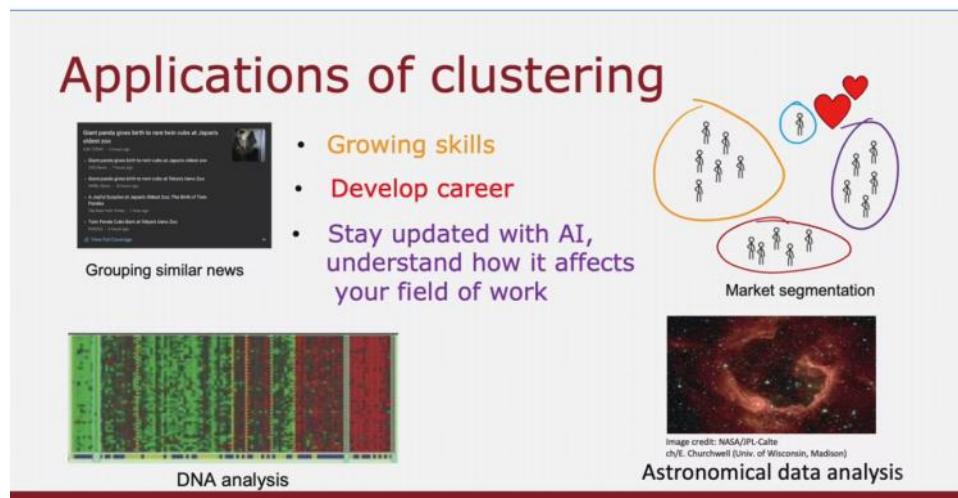
We don't have target labels (y) or Outputs / right answer

Instead we ask the algorithm to find structure in the data



Clustering:

Attempts to find structuring in the data by grouping it



Applications of clustering:

Grouping news together

Discovering market segments

Grouping people by DNA patterns

K-Means Intuition (first decide clusters...)

Monday, 11 December 2023 05:48

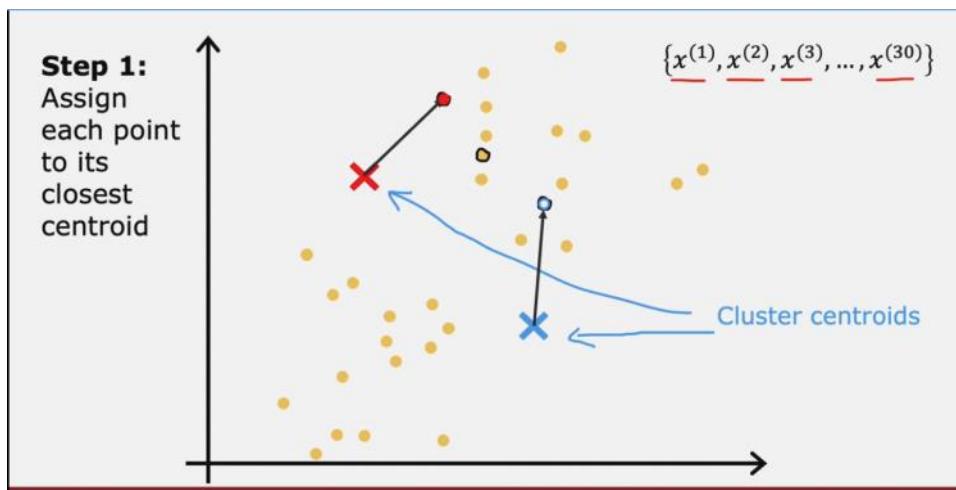
How K-means (clustering) works:

Initial steps:

First you decide how many clusters your looking for

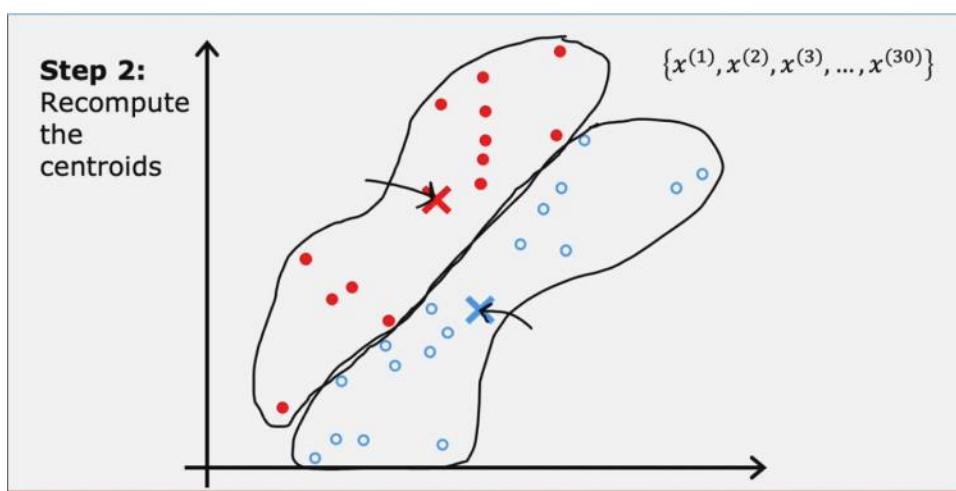
Then the model randomly picks initial points for cluster centroids

Repeatable step one (k-means):



Then it will go through all the data and assigns each point to the closest centroid red or blue (in this case)

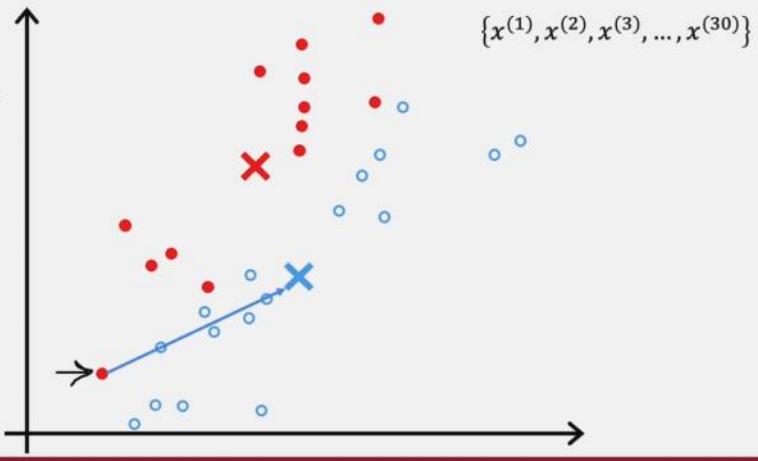
Repeatable step two (k-means):



Move the centroid to the average position of all its assigned datapoints

Keep repeating until no more changes

Step 1:
Assign
each point
to its
closest
centroid



K-Means Algorithm (If no datapoints)

Monday, 11 December 2023 06:12

K-Means:

K-means algorithm

```

Randomly initialize K cluster centroids  $\mu_1, \mu_2, \dots, \mu_K$ 
Repeat {
    # Assign points to cluster centroids
    for  $i = 1$  to  $m$ 
         $c^{(i)} :=$  index (from 1 to  $K$ ) of cluster centroid closest to  $x^{(i)}$ 
    # Move cluster centroids
    for  $k = 1$  to  $K$ 
         $\mu_k :=$  average (mean) of points assigned to cluster  $k$ 
}
 $\mu_1 = \frac{1}{4} [x^{(1)} + x^{(5)} + x^{(6)} + x^{(10)}]$ 

```

First - Randomly initialize cluster centroids (K)

Repeat:

```

# Assign points to cluster centroids
for  $i = 1$  to  $m$ 
     $c^{(i)} :=$  index (from 1 to  $K$ ) of cluster centroid closest to  $x^{(i)}$ 

```

Assign datapoints to closest centroid

```

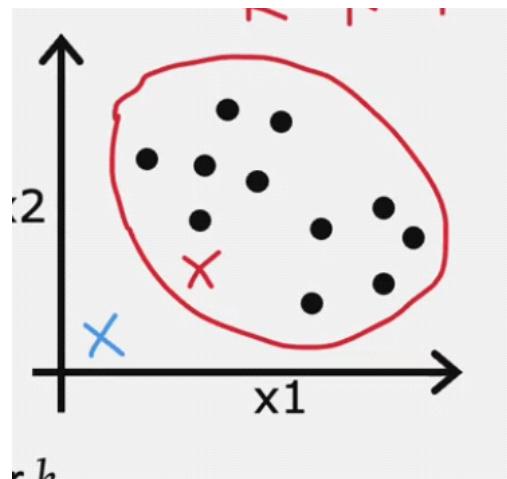
# Move cluster centroids
for  $k = 1$  to  $K$ 
     $\mu_k :=$  average (mean) of points assigned to cluster  $k$ 

```

Move centroid to the average mean position of datapoint vectors assigned

Stop when no more changes

Issue:

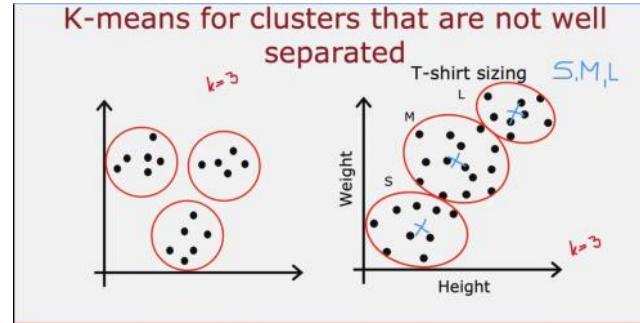


If no datapoints around a cluster either:

Remove it (normally this)

Or Re-initialize it

K-means works on not well defined groups



Optimization Objective (Closest Possible...)

Monday, 11 December 2023 06:46

K-Means Optimization Objective:

K-means optimization objective

$c^{(i)}$ = index of cluster ($1, 2, \dots, K$) 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 function

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

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

$C(i)$ = index of existing clusters

μ_k = existing centroids

$\mu_{c^{(i)}}$: Given an example, what cluster its at

Cost Function (Distortion function) for K-means:

Cost function

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

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

Works by average the squared distance between points and there assigned centroid (j)

Cost function for K-means

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

Repeat {

- # Assign points to cluster centroids
- for $i = 1$ to m
- $c^{(i)}$:= index of cluster centroid closest to $x^{(i)}$
- # Move cluster centroids
- for $k = 1$ to K
- μ_k := average of points in cluster k

}

The goal is to minimize this cost function by assigning datapoints to the closest possible centroid

Moving the centroid

$$\frac{1}{2}(1^2 + 9^2) = \frac{1}{2}(1+8) = 41$$

$$\frac{1}{2}(1+1) = 6$$

$$\frac{1}{2}(5^2 + 5^2) = 25$$

and moving the centroid to the lowest distance mean location for its assigned points

Because the Distortion function works to minimize distance to centroid by:

Assigning datapoints to closest centroid

And moving centroid to the nearest average position of there assigned points

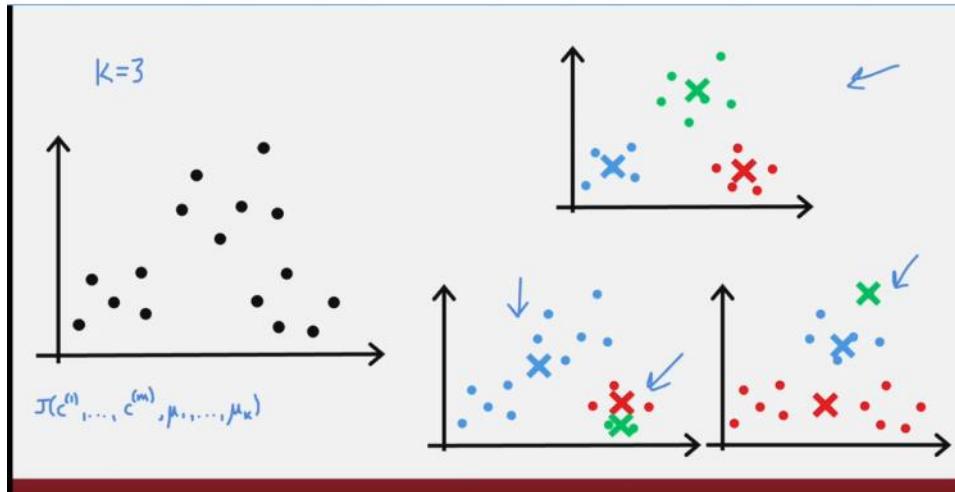
It will always converge,

If it cost ever goes up, bug in code

If it stays the same, its converged

Initializing K-Means (local minima)

Monday, 11 December 2023 07:15



K-Means can get stuck in local-minima causing a high cost and worse model

Random initialization

```
For i = 1 to 100 { 50-1000
    Randomly initialize K-means. k random examples
    Run K-means. Get  $c^{(1)}, \dots, c^{(m)}, \mu_1, \mu_2, \dots, \mu_k \leftarrow$ 
    Computer cost function (distortion)
     $J(c^{(1)}, \dots, c^{(m)}, \mu_1, \mu_2, \dots, \mu_k) \leftarrow$ 
}
```

Pick set of clusters that gave lowest cost J

Best thing to do is to run K-Means many times with different centroid initializations to convergence

Run this 50-1000

Get the cost for each of your initializations

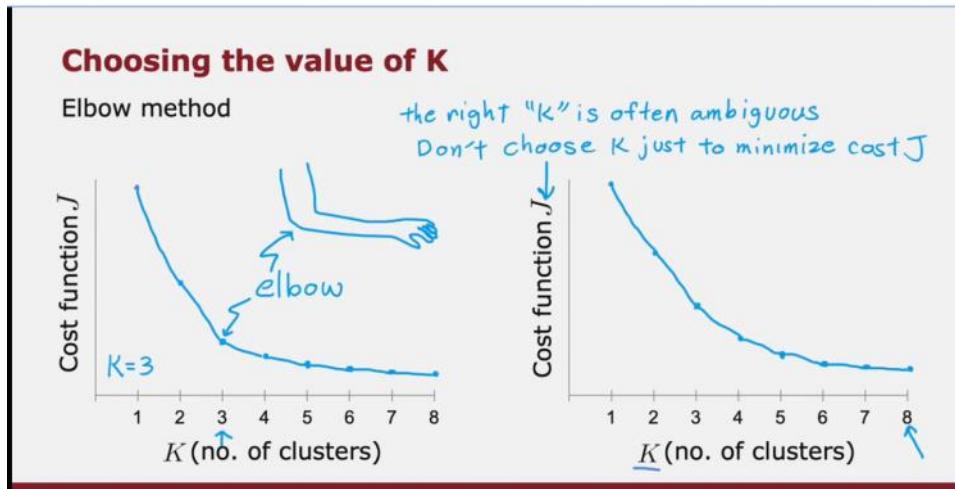
Pick the lowest cost one

Choosing the number of clusters (cost cluster...change)

Monday, 11 December 2023 07:54

Choosing the value of K (Clusters):

This is the first step when doing K-Means

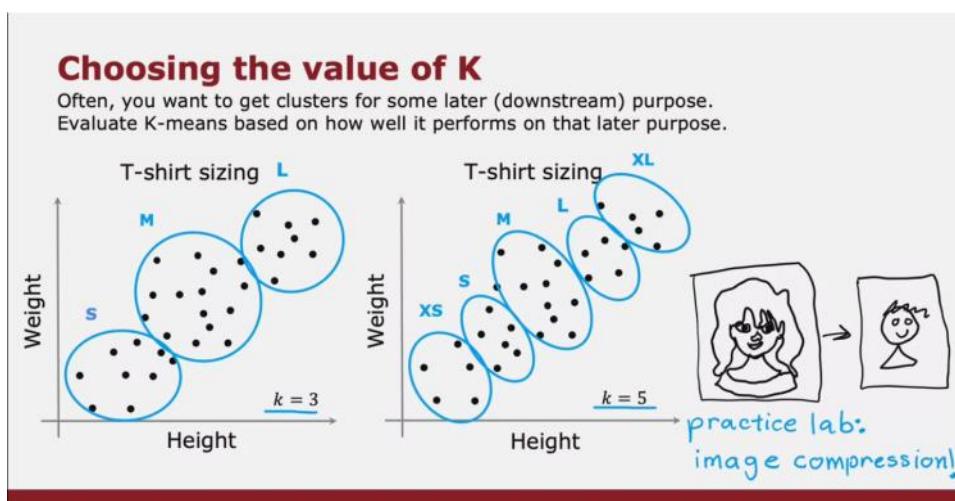


Elbow method measure cost against clusters - look for downtick, 3 in this case

However not recommended because downtick is not always clear

Best way of deciding k (how many clusters):

Look at your downstream purpose



E.g T-Shirt sizing - ExtraSmall,Small,Medium,Large,ExtraLarge - Would mean 5 clusters

K Means Practice Lab

Monday, 11 December 2023 09:19

You

what does these line of python 3 do?

```
for k in range(K):
    points = X[idx == k]
    centroids[k] = np.mean(points, axis = 0)

return centroids
```

ChatGPT

This Python code is part of a process to update the centroids in a k-means clustering algorithm. Let's break it down:

1. `for k in range(K)` : This loop iterates over each cluster index `k` from 0 to `K-1`, where `K` is the total number of clusters.
2. `points = X[idx == k]` : For each cluster `k`, it uses boolean indexing to select the data points from the input data `X` that are assigned to the current cluster (`idx == k` is the condition). These points are stored in the variable `points`.
3. `centroids[k] = np.mean(points, axis=0)` : For each cluster `k`, it calculates the mean (centroid) of the data points assigned to that cluster along the specified axis (axis=0 means along columns). The resulting centroid is then assigned to the `k`-th entry in the array or list `centroids`.
4. This process is repeated for each cluster, updating the centroids based on the mean of the points assigned to each cluster.
5. Finally, the updated centroids are returned from the function.

In summary, this code updates the centroids of K clusters in a k-means algorithm based on the mean of the points assigned to each cluster.



C3_W1_KM
eans_Assi...



C3_W1_KM
eans_Assi...

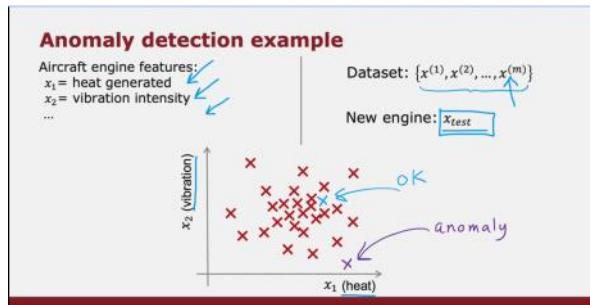
- If you're still stuck, you can check the hints presented below to figure out how to calculate `norm_ij` and `idx[i]`
 - ▼ **Hint to calculate norm_ij**
You can use [np.linalg.norm](#) to calculate the norm
 - ▼ **More hints to calculate norm_ij**
You can compute `norm_ij` as `norm_ij = np.linalg.norm(X[i] - centroids[j])`
 - ▼ **Hint to calculate idx[i]**
You can use [np.argmin](#) to find the index of the minimum value
 - ▼ **More hints to calculate idx[i]**
You can compute `idx[i]` as `idx[i] = np.argmin(distance)`

Need to learn numpy look in discord after-coruse-learning channel

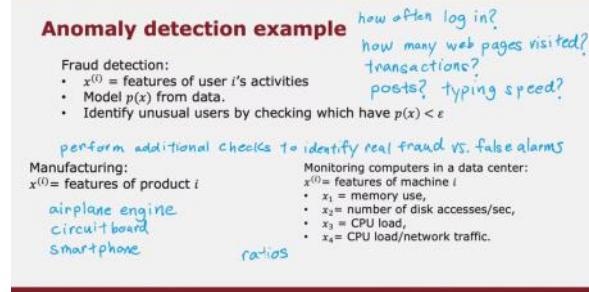
Finding Unusual Event (Works by Calculating...) - Done

Monday, 11 December 2023 09:29

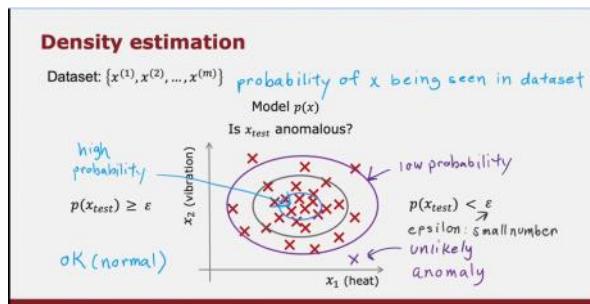
Anomaly Detection Example:



Where is anomaly Detection used:



Density Estimation:



Fraud detection using features like spending patterns etc,

Manufactured device defect checker temperature etc

Data center monitoring, CPU load, network load etc

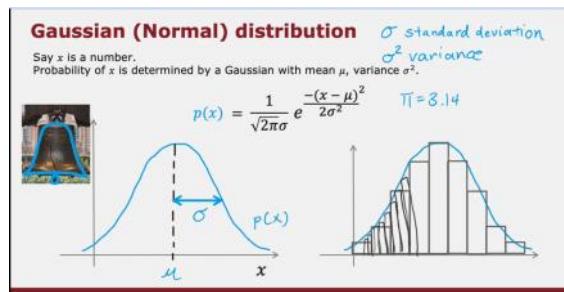
Works by calculating the probability of a features value being equal to its current value given the dataset

If the probability is low less than epsilon (a small number) declare it an anomaly

Gaussian (normal) distributions (parameter Estimation) - Done

Wednesday, 13 December 2023 05:48

Gaussian distribution is same as normal distribution



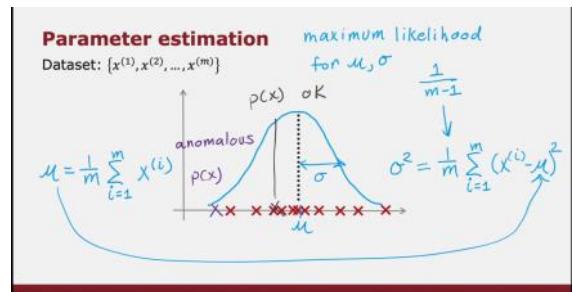
σ = standard deviation

σ^2 = Variance

μ = Mean value of all features

Parameter Estimation:

You can use Gaussian to do anomaly detection by applying a bell curve to your dataset



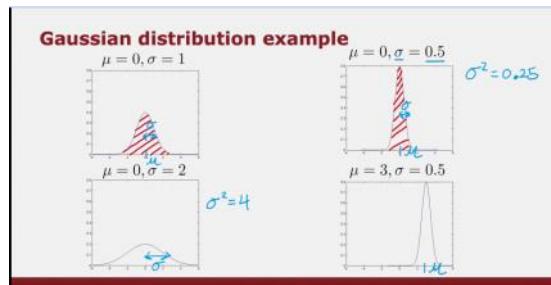
The closer to the edge of the curve, the more anomalous

Take those and put them into the probability:

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

Using a gaussian along with the mean (μ) and variance you can calculate the probability of X

The width of the curve depends on standard deviation



Probabilities must always equal one, so as you decrease standard deviation the curve must get taller and skinnier

The mean value will always be the center of the curve

Anomaly Detection Algorithm (Density Estimation) - Done

Wednesday, 13 December 2023 06:23

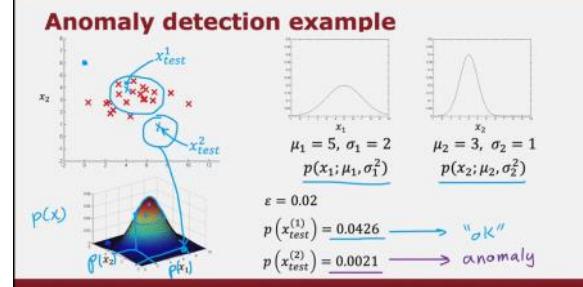
Density Estimation:

Density estimation

Training set: $\{\vec{x}^{(1)}, \vec{x}^{(2)}, \dots, \vec{x}^{(m)}\}$
Each example $\vec{x}^{(i)}$ has n features

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

$$= \prod_{j=1}^n p(x_j; \mu_j, \sigma_j^2) \quad \text{"add"}/\text{"multiply"}$$

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


The closer to the edge of the curve, the more likely its an anomaly (lower probability)

O = standard deviation

O Squared = Variance

μ (index) = Mean value for that features vector

During density estimation we must find the overall probability for the dataset using variance, mean, standard deviation and multination

Anomaly detection algorithm

- Choose n features x_i that you think might be indicative of anomalous examples.
- Fit parameters $\mu_1, \dots, \mu_n, \sigma_1^2, \dots, \sigma_n^2$

$$\mu_j = \frac{1}{m} \sum_{i=1}^m x_j^{(i)}$$

$$\sigma_j^2 = \frac{1}{m} \sum_{i=1}^m (x_j^{(i)} - \mu_j)^2$$
- Given new example x , compute $p(x)$:

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

Anomaly if $p(x) < \epsilon$

Vectorized formula

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

$$\vec{\mu} = \begin{bmatrix} \mu_1 \\ \mu_2 \\ \vdots \\ \mu_n \end{bmatrix}$$

Step 1.

- Choose n features x_i that you think might be indicative of anomalous examples.

Find examples that must be anomalies

Step 2.

Fit your parameters (standard deviation, variance, mean) OR USE ABOVE (2) FORMULA FOR NON-VECTOR

Vectorized formula

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

$$\vec{\mu} = \begin{bmatrix} \mu_1 \\ \mu_2 \\ \vdots \\ \mu_n \end{bmatrix}$$

1. Vector Mean

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

2. Variance

1/m gives you the average value for the dataset

Step 3.

3. Given new example x , compute $p(x)$:

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

Given a new example, compute $P(X)$ using density estimation

Step 4.

Anomaly if $p(x) < \varepsilon$

If small probability (smaller than epsilon): flag as anomaly

We decide the size of epsilon (small number)

Normally happens if its unlike most data in the training set

When given a new example compute the probability using above formula

$$\frac{1}{\sqrt{2\sigma_j^2}} \cdot \exp\left(-\frac{(x_j - \mu_j)^2}{2\sigma_j^2}\right)$$

This is from previous step

This is also from previous step

This is your feature (currently selected)

$$(x_j)$$

Developing and Evaluating An anomaly Detection System (Real Number Evaluation) - Done

Wednesday, 13 December 2023 07:08

Real-Number-evaluation:

The importance of real-number evaluation

When developing a learning algorithm (choosing features, etc.), making decisions is much easier if we have a way of evaluating our learning algorithm.

Assume we have some labeled data, of anomalous and non-anomalous examples.

$$y=1 \quad y=0$$

Training set: $x^{(1)}, x^{(2)}, \dots, x^{(m)}$ (assume normal examples/not anomalous)

$y=0$ for all training examples

Cross validation set: $(x_{cv}^{(1)}, y_{cv}^{(1)}), \dots, (x_{cv}^{(m_{cv})}, y_{cv}^{(m_{cv})})$
Test set: $(x_{test}^{(1)}, y_{test}^{(1)}), \dots, (x_{test}^{(m_{test})}, y_{test}^{(m_{test})})$

} include a few anomalous examples
 $y=1$
mostly normal examples
 $y=0$

When making changes to our model (choosing features, epsilon) we need a way of evaluating our changes

So we need some labeled data to build a cross validation and testing set

Aircraft engines monitoring example

10000 good (normal) engines 2 to 50
20 flawed engines (anomalous) $y=1$
2 $y=0$

Training set: 6000 good engines train algorithm on training set

CV: 2000 good engines ($y = 0$) 10 anomalous ($y = 1$)
use cross validation set tune ϵ tune x_j
Test: 2000 good engines ($y = 0$), 10 anomalous ($y = 1$)

Alternative: No test set use if very few labeled anomalous examples

Training set: 6000 good engines higher risk of overfitting

CV: 4000 good engines ($y = 0$), 20 anomalous ($y = 1$)
tune ϵ tune x_j

By building a cross validation and testing set with labeled normal & anomalous data included we can test our models performance

Using the results from testing with can adjust our features and Epsilon

If you don't have much labeled data just build a cross validation set but this can cause overfitting

Algorithm evaluation

Fit model $p(x)$ on training set $x^{(1)}, x^{(2)}, \dots, x^{(m)}$
On a cross validation/test example x , predict

course 2 week 3
skewed datasets

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

10
2000

Possible evaluation metrics:

- True positive, false positive, false negative, true negative
- Precision/Recall
- F_1 -score

Use cross validation set

Fit model on unlabeled data (because unsupervised)

Then use cross validation / test set to see how many anomalies and false positives / negatives the model has

Anomaly Detection vs. Supervised Learning

Wednesday, 13 December 2023 07:46

Anomaly Detection vs. Supervised Learning

Anomaly detection vs. Supervised learning	
<p>Very small number of positive examples ($y = 1$). (0-20 is common). Large number of negative ($y = 0$) examples.</p> <p>$p(x)$ $y=1$</p> <p>Many different "types" of anomalies. Hard for any algorithm to learn from positive examples what the anomalies look like; <u>future anomalies may look nothing like any of the anomalous examples we've seen so far.</u></p> <p>Fraud</p>	<p>Large number of positive and negative examples.</p> <p>20 positive examples</p> <p>Enough positive examples for algorithm to get a sense of what positive examples <u>are like</u>, <u>future positive examples likely to be similar to ones in training set.</u></p> <p>Spam</p>

Use anomaly Detection when:

You don't have many positive examples (20 is common)

You have a large number of negative examples

The problem is very dynamic and might not be anything like previous examples

Fraud is an example of a use case

Use Supervised Learning:

You have a lot of positive examples

Future positive examples are likely to be similar to the training set

Email spam is an example use case

Anomaly detection vs. Supervised learning	
<ul style="list-style-type: none">→ Fraud detection→ Manufacturing - Finding <u>new previously unseen defects</u> in manufacturing.(e.g. aircraft engines)→ Monitoring machines in a data center⋮	<ul style="list-style-type: none">→ Email spam classification→ Manufacturing - Finding known, previously <u>seen</u> defects $y=1$ <u>scratches</u>→ Weather prediction (sunny/rainy/etc.)• Diseases classification⋮

Choosing What Features To Use (bell shape, error analysis)

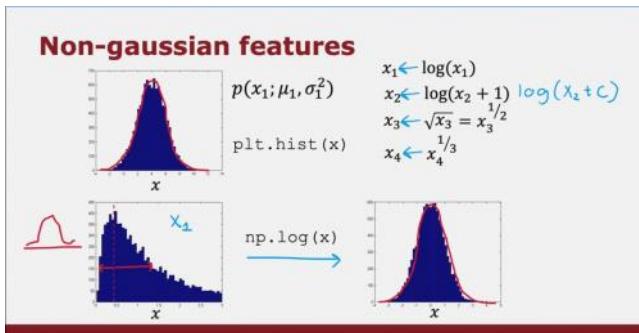
Wednesday, 13 December 2023 08:16

Using a histogram you can see if your feature is Gaussian

Plt.hist(x) will help you do this

Gaussian features will make a 'bell' shape

Non-Gaussian Features:

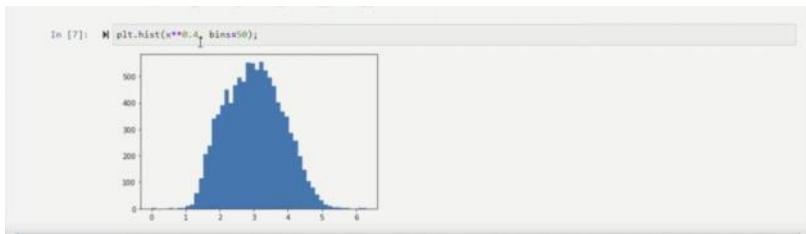


You should convert non-Gaussian features into Gaussian ones

You can use Log etc to do this

The larger the value of C, the lesser the transformation

In python:



Plt.hist(x**0.4) has made this feature more Gaussian

C = 0.4 in this example

Using Np.log:



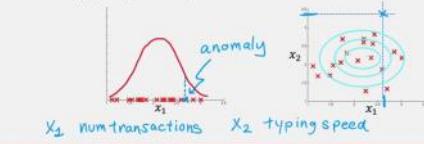
C = 7 in this example

Error Analysis for Anomaly Detection:

Error analysis for anomaly detection

Want $p(x) \geq E$ large for normal examples x .
 $p(x) < E$ small for anomalous examples x .

Most common problem:
 $p(x)$ is comparable for normal and anomalous examples.
($p(x)$ is large for both)



You should try and add features that help show up anomalies better

You can do this by looking at the failed cross validation examples (error analysis)

Feature Engineering:

You can make features that combine others to look for anomalies (high CPU + high network traffic)

Monitoring computers in a data center

Choose features that might take on unusually large or small values
in the event of an anomaly.

x_1 = memory use of computer
 x_2 = number of disk accesses/sec
high x_3 = CPU load
low x_4 = network traffic \leftarrow not unusual
 $x_5 = \frac{\text{CPU load}}{\text{network traffic}}$ $x_6 = \frac{(\text{CPU load})^2}{\text{network traffic}}$

Deciding feature choice based on $p(x)$
Large for normal examples;
Becomes small for anomaly in the cross validation set

Error Metrics For Skewed Dataset (Precision & Recall)

Wednesday, 18 October 2023 08:50

When you have a dataset that isn't balanced very well, you can't use classification errors (train or test)

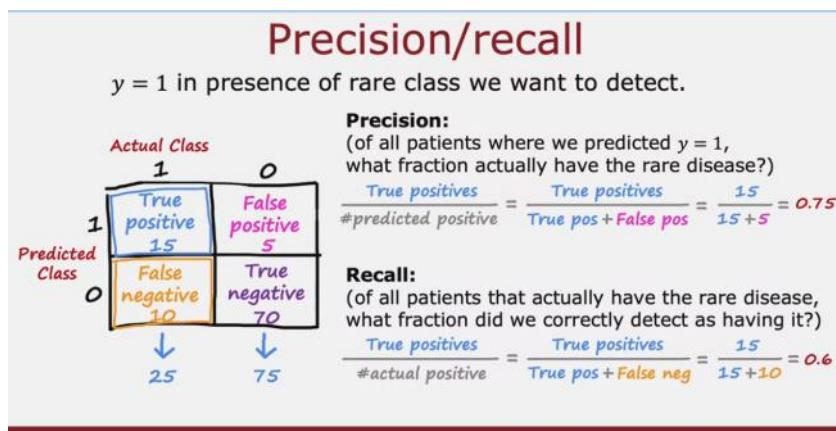
Precision and Recall - Allows you to know how accurate your model is when giving positive readings (Precision) & that the model is detecting most of them (Recall)

Precision - Of all our detected positives, how many were really positive

Recall - How good the model is at detecting the correct classification overall

The higher Precision and Recall the better

Precision / Recall:



Precision:

(of all patients where we predicted $y = 1$, what fraction actually have the rare disease?)

$$\frac{\text{True positives}}{\#\text{predicted positive}} = \frac{\text{True positives}}{\text{True pos} + \text{False pos}} = \frac{15}{15+5} = 0.75$$

Precision - what fraction of the data is positive ($y=1$)

Recall:

(of all patients that actually have the rare disease, what fraction did we correctly detect as having it?)

$$\frac{\text{True positives}}{\#\text{actual positive}} = \frac{\text{True positives}}{\text{True pos} + \text{False neg}} = \frac{15}{15+10} = 0.6$$

Recall - What fraction did our model correctly detect

Trading Off Precision and Recall (Thresholds, Confidence)

Wednesday, 18 October 2023 09:39

Pick the model with the highest Precision and Recall

Trading off precision and recall

Logistic regression: $0 < f_{\vec{w}, b}(\vec{x}) < 1$

- Predict 1 if $f_{\vec{w}, b}(\vec{x}) \geq 0.5$
- Predict 0 if $f_{\vec{w}, b}(\vec{x}) < 0.5$

Suppose we want to predict $y = 1$ (rare disease) only if very confident.

higher Precision, lower recall

Suppose we want to avoid missing too many case of rare disease (when in doubt predict $y = 1$)

lower precision, higher recall

More generally predict 1 if: $f_{\vec{w}, b}(\vec{x}) \geq \text{threshold}$.

precision = $\frac{\text{true positives}}{\text{total predicted positive}}$

recall = $\frac{\text{true positives}}{\text{total actual positive}}$

threshold = 0.99
threshold = 0.01

The higher the threshold, higher the Precision.

Precision is our confidence level when making predictions

The lower the threshold, the higher the Recall, because more examples are detected.

When both Precision and Recall are low, your model isn't very good.

F1 Score:

F1 Score works out how to trade off Precision and Recall

F1 score

How to compare precision/recall numbers?

	Precision (P)	Recall (R)	Average	F ₁ score
Algorithm 1	0.5	0.4	0.45	0.444
Algorithm 2	0.7	0.1	0.4	0.175
Algorithm 3	0.02	1.0	0.501	0.0392

print("y=1")

~~Average = $\frac{P+R}{2}$~~

$$F_1 \text{ score} = \frac{1}{\frac{1}{2}(\frac{1}{P} + \frac{1}{R})} = 2 \frac{PR}{P+R}$$

The higher the F1 Score the better the model (more balanced the Precision and Recall are)

Anomaly Detection

Thursday, 14 December 2023 07:24



C3_W1_An
omaly_De...



C3_W1_An
omaly_De...



LOOPING
METHOD



LOOPING
METHOD

Making Recommendations (...we can make recommendations)

Friday, 15 December 2023 05:43

Predicting Movie Ratings:

Predicting movie ratings				
Movie	User rates movies using one to five stars			
	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	?

$n_u = 4$ $r(1,1) = 1$ $y^{(i,j)} = \text{rating given by user } j \text{ to movie } i$
 $n_m = 5$ $r(3,1) = 0$ $y^{(3,2)} = 4$ (defined only if $r(i,j)=1$)

$\xrightarrow{\hspace{1cm}}$ $n_u = \text{no. of users}$
 $\xrightarrow{\hspace{1cm}}$ $n_m = \text{no. of movies}$
 $\xrightarrow{\hspace{1cm}}$ $r(i,j)=1$ if user j has rated movie i

Ratings

★				
★	★			
★	★	★		
★	★	★	★	
★	★	★	★	★

Using a rating system we can make recommendations

The movies a user hasn't rated we can try and predict there rating

Using per-item features (own model, predictions, regularization)

Friday, 15 December 2023 05:58

Movies with Features:

Movie	What if we have features of the movies?					$N_u = 4$ $N_m = 5$ $n = 2$	
	Alice(1)	Bob(2)	Carol(3)	Dave(4)	x_1 (romance)	x_2 (action)	
Love at last	5	5	0	0	0.9	0	$x^{(1)} = \begin{bmatrix} 0.9 \\ 0 \end{bmatrix}$
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	$x^{(3)} = \begin{bmatrix} 0.99 \\ 0 \end{bmatrix}$
Swords vs. karate	0	0	5	?	0	0.9	

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

If we have features about items in our dataset with can make predictions for the users rating

Above is an example of allices movie rating for 'cute puppies of love' = 4.95

Each user has there own dataset of ratings, there own weights and biases, there own model

In this case we are just using linear regression

Dot product, times in pairs then sums

Cost Function:

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

$R(i,f)$ = has a user rated a movie(0 if no)

$Y(i,j)$ = rating given by a user of movie(i)

$W(j), b(j)$ = parameters for user(j)

$X(i)$ feature vector for movie(i)

Given each user has there own model we must use a cost function to make accurate predictions by finding there correct Weights and Bias

Above is the squared error cost function between our predictions and the real ratings of movies in users dataset

This cost function also does regularization (in orange) to prevent overfitting



Only loops over movies the user has rated

Cost Function for entire dataset:

Cost function	
To learn parameters $w^{(j)}, b^{(j)}$ for user j :	
$J(w^{(j)}, b^{(j)}) = \frac{1}{2} \sum_{l:r(l,j)=1} (w^{(j)} \cdot x^{(l)} + b^{(j)} - y^{(l,j)})^2 + \frac{\lambda}{2} \sum_{k=1}^n (w_k^{(j)})^2$	
To learn parameters $w^{(1)}, b^{(1)}, w^{(2)}, b^{(2)}, \dots, w^{(n_u)}, b^{(n_u)}$ for all users :	
$J(w^{(1)}, b^{(1)}, \dots, w^{(n_u)}, b^{(n_u)}) = \frac{1}{2} \sum_{j=1}^{n_u} \sum_{l:r(l,j)=1} (w^{(j)} \cdot x^{(l)} + b^{(j)} - y^{(l,j)})^2 + \frac{\lambda}{2} \sum_{j=1}^{n_u} \sum_{k=1}^n (w_k^{(j)})^2$	

Above is a cost function for all users given there ratings of movies in there dataset

This can then be trained using gradient descent

N_u = number of users

N_m = number of movies

I = current movie

J = current user

Collaborative Filtering Algorithm (assign new features, multiple users)

Friday, 15 December 2023 06:27

Learning X (about a movie) from users with Collaborative Filtering:

Using collaborative filtering we filter out items that a user might like using reactions by similar users

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	?	2
Cute puppies of love	?	4	0	?	?	x ⁽²⁾
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 $w^{(j)} \cdot x^{(i)} + b^{(j)}$

$w^{(1)} \cdot x^{(1)} \approx 5 \rightarrow x^{(1)} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$

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

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

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

Using each of our users trained models, we can assign new features for our items

E.g. Alice normally likes romance, so if she rates it highly, it's likely romantic

Learning features of an item is only possible if multiple users have ratings for this item

This feature information will allow us to recommend items to similar users

Cost Function For X (features):

Collaborative filtering		
Cost function to learn $w^{(1)}, b^{(1)}, \dots, w^{(n_u)}, b^{(n_u)}$:		
$w^{(1)}, b^{(1)}, \dots, w^{(n_u)}, b^{(n_u)}$	$\min_{w^{(1)}, b^{(1)}, \dots, w^{(n_u)}, b^{(n_u)}} \frac{1}{2} \sum_{j=1}^{n_u} \sum_{i:r(l,j)=1} (w^{(j)} \cdot x^{(i)} + b^{(j)} - y^{(i,j)})^2 + \frac{\lambda}{2} \sum_{k=1}^{n_u} \sum_{j=1}^{n_u} (w_k^{(j)})^2$	$j=1 \quad j=2 \quad j=3$ Alice Bob Carol
$x^{(1)}, \dots, x^{(n_m)}$	$\min_{x^{(1)}, \dots, x^{(n_m)}} \frac{1}{2} \sum_{i=1}^{n_m} \sum_{j:r(i,j)=1} (w^{(j)} \cdot x^{(i)} + b^{(j)} - y^{(i,j)})^2 + \frac{\lambda}{2} \sum_{i=1}^{n_m} \sum_{k=1}^{n_u} (x_k^{(i)})^2$	
Put them together:	$J(w, b, x) = \frac{1}{2} \sum_{(i,j):r(i,j)=1} (w^{(j)} \cdot x^{(i)} + b^{(j)} - y^{(i,j)})^2 + \frac{\lambda}{2} \sum_{j=1}^{n_u} \sum_{k=1}^{n_u} (w_k^{(j)})^2 + \frac{\lambda}{2} \sum_{i=1}^{n_m} \sum_{k=1}^{n_u} (x_k^{(i)})^2$	

As long as we have the correct weights and bias's for our users / trained models we can predict the features for our movies

Above cost function goes over all the ratings given by all users who have watched the film

Make sure you transfer regularization from training the user model for finding features model

X is the movie we want to learn about

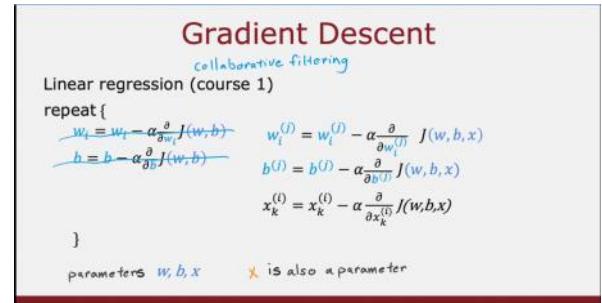
$\sum_{i=1}^{n_m} \sum_{j:r(i,j)=1}$

Each user that has rated movie

$\sum_{i=1}^{n_m} \sum_{j:r(i,j)=1}$

Each movie that has been rated by user - same thing

Gradient Descent for X (movies):



X is now a parameter in our gradient descent

X(i) is the movie

X(i)k is the exact feature for the current movie (i)

A is learning rate, d = derivative

Binary Labels: Favs, Likes, Clicks (data to our... using binary...)

Friday, 15 December 2023 07:16

Binary Labels:

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

Using binary labels we can provide data to our recommender system

Using binary events like this we can tune our recommender model because we will know how the user interacted with a piece of information

From regression to binary classification:

Because we are now using binary values to suggest if a user engaged with something, we must now use Logistic Regression

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 $g(w^{(j)} \cdot x^{(i)} + b^{(j)})$
where $g(z) = \frac{1}{1+e^{-z}}$

G = activation function

Cost Function:

Cost function for binary application

Previous cost function:

$$\frac{1}{2} \sum_{(i,j) \in \tau(i,j)=1} (w^{(j)} \cdot x^{(i)} + b^{(j)} - y^{(i,j)})^2 + \frac{\lambda}{2} \sum_{i=1}^m \sum_{k=1}^n (x_k^{(i)})^2 + \frac{\lambda}{2} \sum_{j=1}^n \sum_{k=1}^m (w_k^{(j)})^2$$

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

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

$J(w, b, x) = \sum_{(i,j) \in \tau(i,j)=1} L(f_{(w,b,x)}(x), y^{(i,j)})$ cost for all examples
 $g(w^{(j)} \cdot x^{(i)} + b^{(j)})$

Because we are now doing binary classification, we use logistic loss for cost function

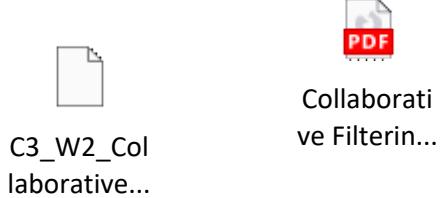
Looping over all items each user has a binary classification for

X are the features of each items / move

$R(i,j) = 1$ means a user has rated an item

Lab - Collaborative Filtering Recommender Systems

07 January 2024 10:35



Mean Normalization (new users..., Rating by other)

Friday, 15 December 2023 08:11

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	?	?

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

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

Mean normalization helps make better predictions for new users without a large dataset to train a model on

Mean Normalization

$\mu = \begin{bmatrix} 2.5 \\ 2.5 \\ 2 \\ 2.25 \\ 1.25 \end{bmatrix}$

For user j , on movie i predict: $w^{(j)} \cdot x^{(i)} + b^{(j)} + \mu_i$

User 5 (Eve): $w^{(s)} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad b^{(s)} = 0 \quad \underbrace{w^{(s)} \cdot x^{(i)} + b^{(s)}}_0 + \mu_i = 2.5$

By adding the Mean ratings by other users in current dataset we can assume values for new users using Mean Normalization

For user j , on movie i predict: $w^{(j)} \cdot x^{(i)} + b^{(j)} + \mu_i$

User 5 (Eve): $w^{(s)} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad b^{(s)} = 0 \quad \underbrace{w^{(s)} \cdot x^{(i)} + b^{(s)}}_0 + \mu_i = 2.5$

Eve, a new user has no weights or bias because she has no dataset or a few small one, by adding

mean normalization we can recommend based on average scores

This also means the algorithm will run a bit faster in practice

$$\mu = \begin{bmatrix} 2.5 \\ 2.5 \\ 2 \\ 2.25 \\ 1.25 \end{bmatrix}$$
$$\text{Normalized Matrix: } \begin{bmatrix} 2.5 & 2.5 & -2.5 & -2.5 & ? \\ 2.5 & ? & ? & -2.5 & ? \\ ? & 2 & -2 & ? & ? \\ -2.25 & -2.25 & 2.75 & 1.75 & ? \\ -1.25 & -1.25 & 3.75 & -1.25 & ? \end{bmatrix}$$

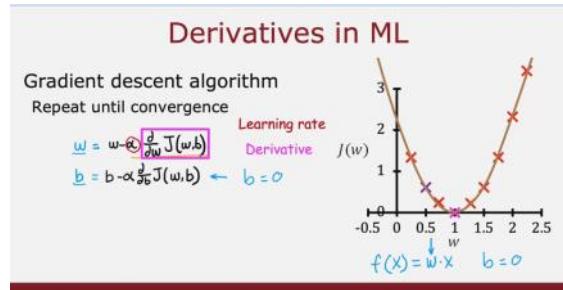
In this case, we normalized the rows to find the mean

We could have normalized the columns, this would have helped us predict a new movies score (although this wouldn't really work for this purpose)

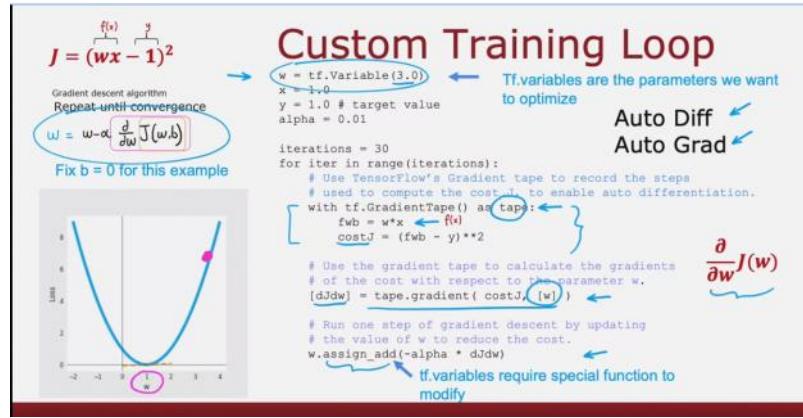
TensorFlow Implementation Of Collaborative Filtering (Tensorflow can be used...)

Friday, 15 December 2023 08:37

TensorFlow Implementation Of Collaborative Filtering:



TensorFlow can be used to help calculate Derivatives for Gradient Descent



Tf.variable(3.0) means set variable to 3.0 and I want to optimize this variable with gradient descent

```
# used to compute the cost J to end
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])
```

Tf.GradientTape() contains the set of steps needed to compute the cost once you show it the cost function

This is AutoGrad / AutoDiff & it allows you to do gradient descent without needing to calculate the derivative yourself

```
# Use the gradient tape to calculate the gradients
# of the cost with respect to the parameter w.
[dJdw] = tape.gradient(costJ, [w])
```

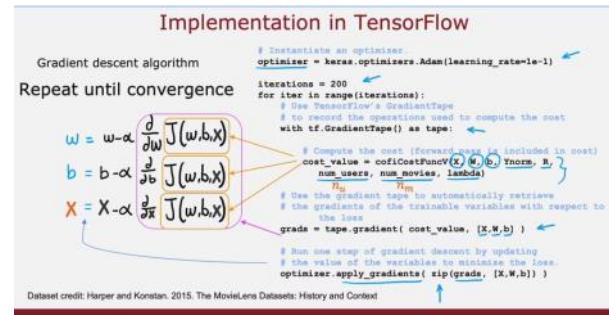
Tf.gradient() calculates the gradient

```
# Run one step of gradient descent by updating
# the value of w to reduce the cost.
w.assign_add(-alpha * dJdw)
```

Then by updating the value you can run gradient descent

This allows you to install gradient descent without yourself, working out the Derivative, this is called AutoDiff

Collaborative Filtering Cost Function:



This is using adams optimizer with the collaborative filtering cost function

```
# Compute the cost (forward pass is included in cost)
cost_value = cofiCostFuncV(X, W, b, Ynorm, R,
                            num_users, num_movies, lambda)
```

All from cost function

Ynorm = Mean Normalization

Lambda = Regularization

```
# 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])
```

Tape.gradient with array of variable will give you gradients / derivatives for all of them

```
# 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]))
```

This will use adams optimizer to do gradient descent

Zip puts numbers in right order for applied gradience

Why not model.fit:

Because collaborative filtering doesn't fit into dense layers / standard neural network structures

So we have to build gradient descent + cost function ourselves using tensorflow tools AutoDiff + Adams optimizer

Finding Related Items (features, distance between, limitations)

Friday, 15 December 2023 09:23

Find 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

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

romance
action

x_1, x_2, \dots, x_n

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

Features are hard to understand on their own, but together they tell us about the item (movies etc)

We can try and find similar items by checking the difference between features

Less distance the better

Limits of Collaborative Filtering:

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 }

Without a lot of user rating data Collaborative filtering doesn't work well

Collaborative filtering struggles showing new users engaging content - mean normalization doesn't fix this entirely

We can't load new information like Genre, Movie stars etc into the model

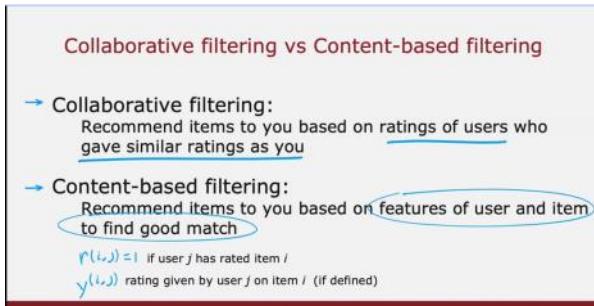
We can't load user information like Age, gender, location etc into the model

Content based filtering can fix these issues

Collaborative Filtering Vs Content-Based Filtering (User Rating, Common features)

Thursday, 21 December 2023 06:11

Collaborative Filtering vs Content-Based Filtering:



With Collaborative filtering we use the experiences of other users to see if similar users might like a piece of content

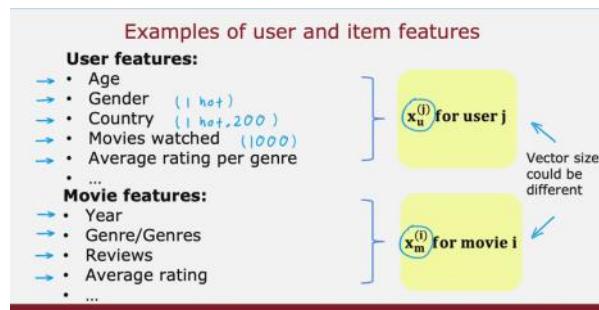
A Content-Based filtering model uses both the features of the current user and the features of items to find matches

You still have a user rating system with content-based filtering

$R(i,j)$ = if a user has rated an item

$Y(i,j)$ = what rating a user gave a piece of content

Examples of user and item features:



Example users features:

Age

Gender

Country

Average rating per genre

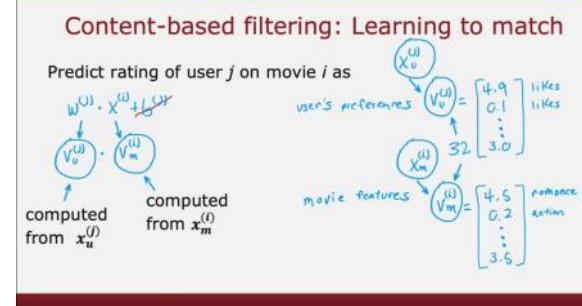
Example Movie Features:

Year

Reviews

Average rating

Content-based Filtering: Learning To Match



$v_u^{(j)}$ is the vector of our user's feature

$v_m^{(i)}$ is the vector of our movie feature

We times them together with dot product

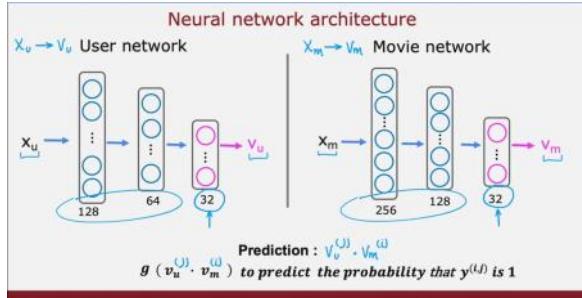
In order to do matrix multiplication these vectors (v) must be the same length

Deeping Learning For Content-Based Filtering (general: neural nets, ...difference)

Thursday, 21 December 2023 06:42

Neural Network Architecture:

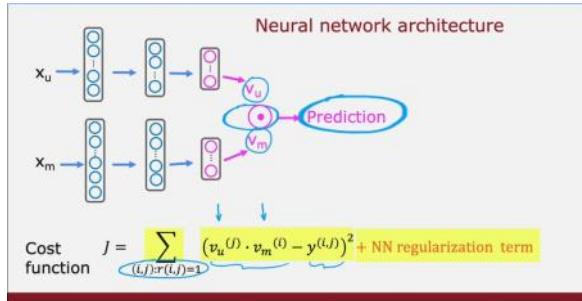
For content based filtering we have two neural nets, one for user, one for item



To compute our user vector we use a neural network with 32 output values

To compute our movie vector we also use a neural network with 32 output values

Our networks must have the same amount of output neurons



To make our prediction we then use a dot product or if binary, a sigmoid activation function on output layer

Our cost function goes over all movies and all the movies the user has rated

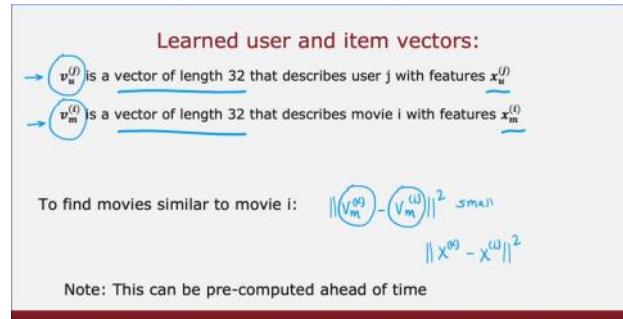
This cost function trains all the parameters for both the user and movie models

We judge them based on there squared error cost

You can also do regularization if high variance

You need to spend a lot of time feature engineering to get this to work correctly

Learned User and item vector:



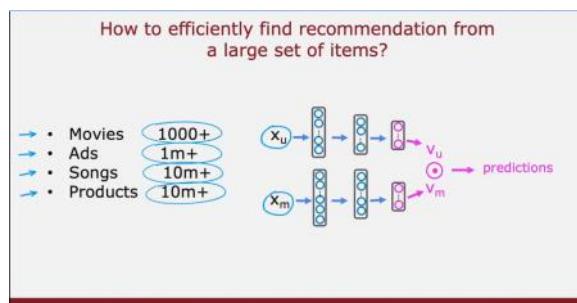
We can use our item vectors to find similar items by measuring the distance between features

We can do this ahead of time, meaning it doesn't have to be done in real-time

Recommending From A Large Catalogue

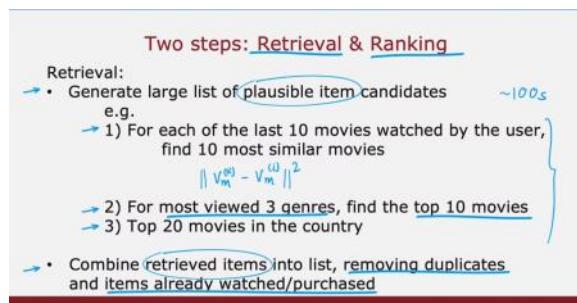
Thursday, 21 December 2023 07:23

Recommending from a large catalogue:



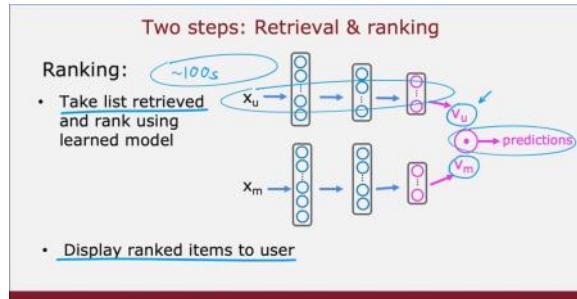
If you have a large catalogue having to run neural network inference is infeasible

Two Steps:



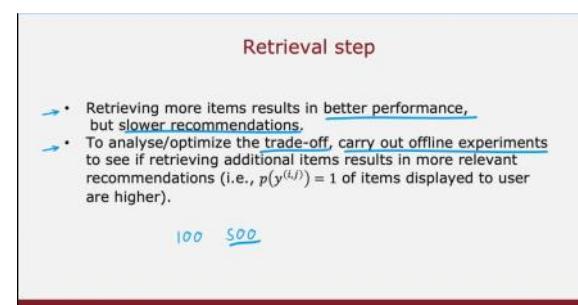
1. First step is Retrieval, we find a list of plausible items to recommend

Look for similar movies to the last ones they watched - measuring the distance between feature



Then do inference to rank our list using our content filtering model, if we already have computed the movie features (vm) because we measure distance, this can be done much quicker

Retrieval Step:



The more items we retrieve the less performance

To work out the trade off between performance and items. Do offline experiments. See if the user reacts better to your recommendations awaiting diminishing returns

Ethical Use Of Recommender Systems

Thursday, 21 December 2023 07:58

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

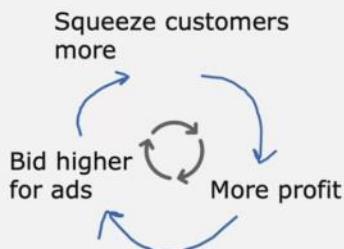


Ethical considerations with recommender systems

Travel industry



Payday loans



Amelioration: Do not accept ads from exploitative businesses

Other problematic cases:

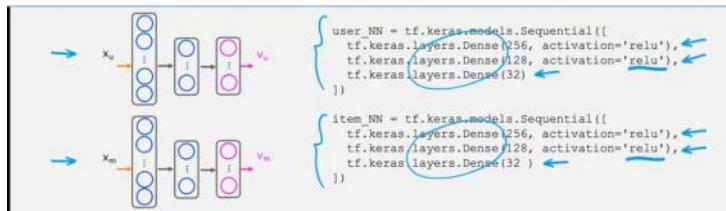
- • Maximizing user engagement (e.g. watch time) has led to large social media/video sharing sites to amplify conspiracy theories and hate/toxicity
- Amelioration : Filter out problematic content such as hate speech, fraud, scams and violent content
- • Can a ranking system maximize your profit rather than users' welfare be presented in a transparent way?

Amelioration : Be transparent with users

TensorFlow Implementation Of Content-Based Filtering

Thursday, 21 December 2023 08:28

TensorFlow Implementation Of Content-Based Filtering:



Both our networks use Forward prop (Sequential) and have the same amount of output neurons

```
# 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)
```

Prepares our features input_user / input_item for the neural networks by turning them into vector

Then feeds it to the networks (vu / vm)

Then applies normalization to length of the vectors

```
{ # measure the similarity of the two vector output
    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()
```

Output is the final prediction

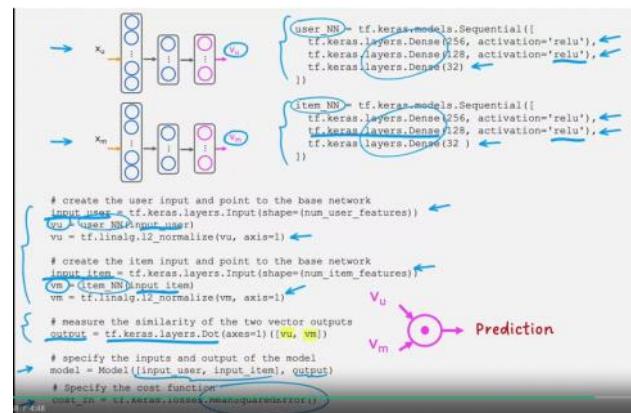
```
# specify the inputs and output of the model
model = Model([input_user, input_item], output)
```

Tells Keras what our inputs & Outputs are

```
# Specify the cost function
cost_fn = tf.keras.losses.MeanSquaredError()
```

Calculates the cost of our model

Code complete:



Deep Learning for Content-Based Filtering

08 January 2024 06:45



C3_W2_Rec
SysNN_As...

Deep
Learning f...

When doing squared difference sum after subtraction & squaring

```
|: # GRADED_FUNCTION: sq_dist
# UNQ_C2
def sq_dist(a,b):
    """
    Returns the squared distance between two vectors
    Args:
        a (ndarray (n,)): vector with n features
        b (ndarray (n,)): vector with n features
    Returns:
        d (float) : distance
    """
    ### START CODE HERE ###

    #Squared difference - sum after squaring
    squared = np.square(a-b)

    d = np.sum(squared)

    ### END CODE HERE ###
    return d
```

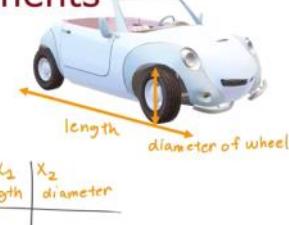
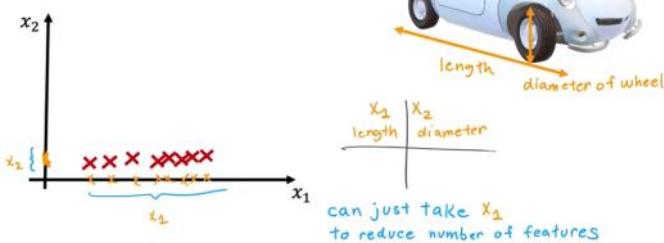
Reducing the number of features

08 January 2024 06:51

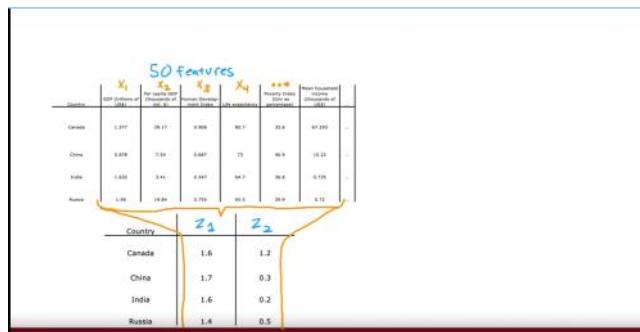
If you have lots of features, you can use PCA to visualize those features

PCA is used to visualize data

Car measurements



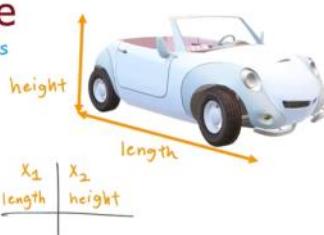
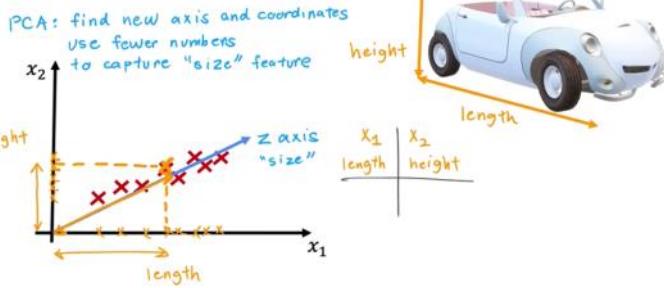
can just take x_1 to reduce number of features



PCA will prioritize features that change the most

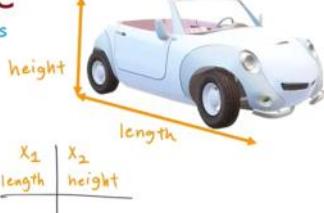
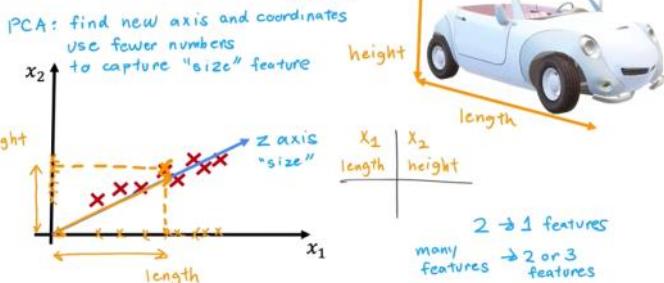
This is done with the goal of reducing features

Size



PCA works to find new axis in the data to reduce the amount of features needed while getting the same amount of data

Size



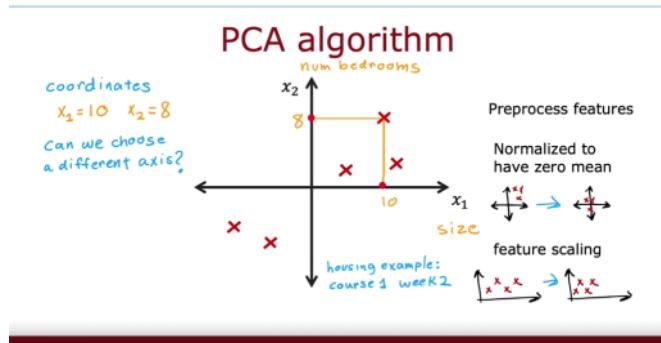
2 → 1 features
many features → 2 or 3 features

PCA can get you from thousands of features to 2 or 3

PCA algorithm

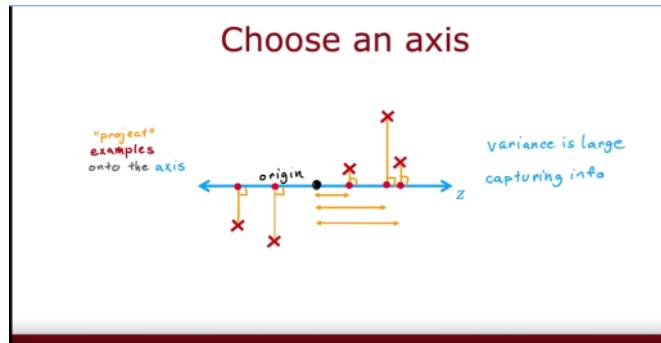
08 January 2024 07:22

PCA algorithm:



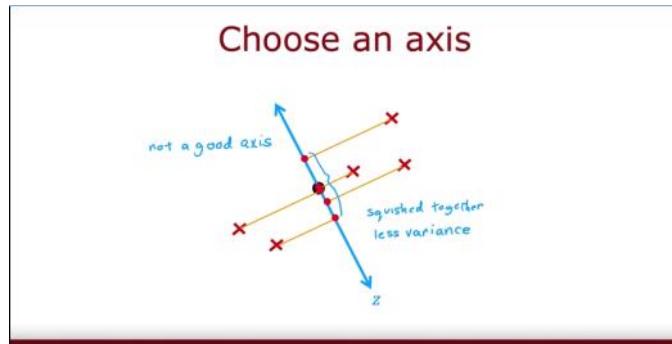
To use PCA you must first do normalization and feature scaling

Origin = 0 on axis

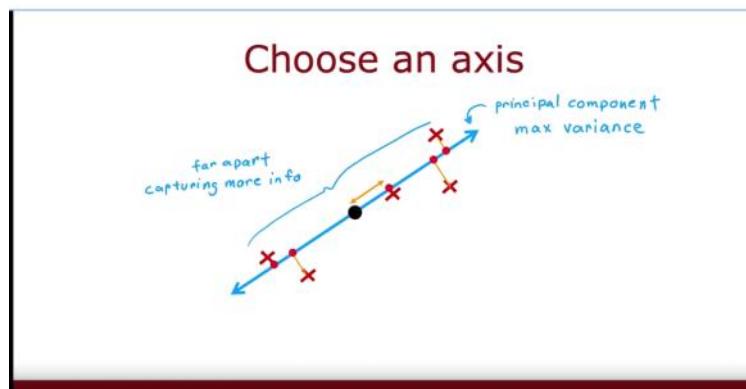


We are going to project it to the axis (find it points on the axis for our data)

The higher the variance in the projections the more information we are getting from the features

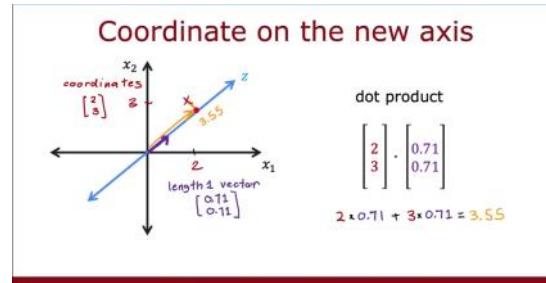


The more squished together the data is, the less variance, the worse the selection.



PCA will work to find the best axis with the max variance

How PCA works:



You can use the dot product of your coordinate axis vector + your feature values vector to calculate projections

The projection is the one number we would use to capture this data in PCA

If you have multiple PCA axis, they will be at 90 degrees from each other

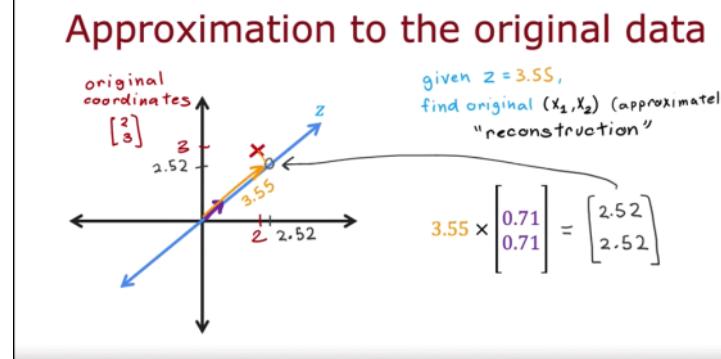
PCA Is not Linear Regression

PCA is unsupervised learning algorithm

PCA tries to find axis while retaining data, not Y

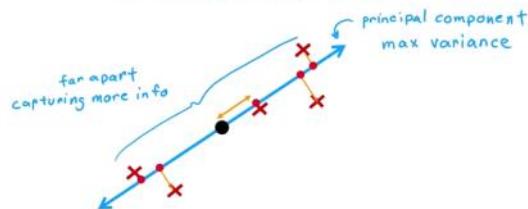
PCA tries to minimize how many features / axis required for modelling

Linear Regression tries to predict Y



You can try and find the original features from PCA using 'reconstruction' approximately

Choose an axis



Principal Component Analysis Page 36

PCA in Code

08 January 2024 08:09

1. You may need to normalize and feature scale before PCA

2. Choose how many axis you want

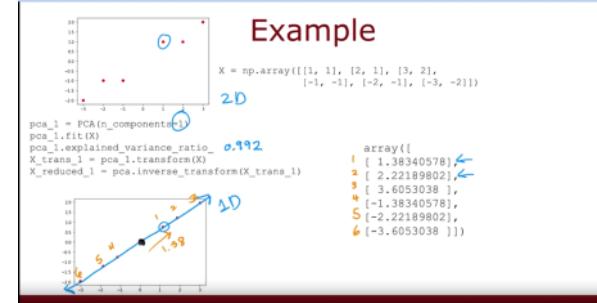
3. Run the 'fit' function

Remember fit function does mean normalization

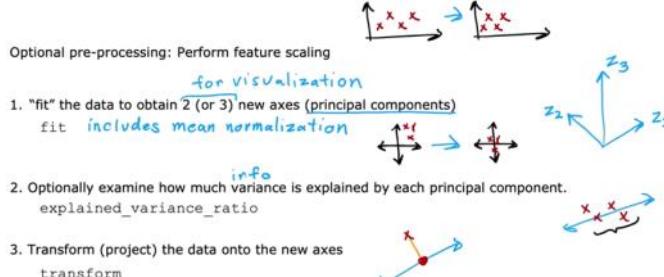
4. Look at your variance of each axis to see how much info you have kept

This can be done using the 'explained_variance_ratio'

You can then transform onto axis



PCA in scikit-learn



Higher the 'explained_variance_ratio' the better

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

X is our features

```
pca_1 = PCA(n_components=1)
pca_1.fit(X)
```

Components='s is how many axis we want

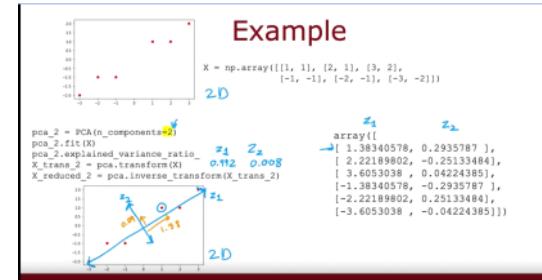
pca_1.explained_variance_ratio_ 0.992

The higher the better, more variance, if you have two or more axis, add them together.

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

For each of our training examples, here is the distance from origin (zero)

Two axis:



Example

PCA is normally used to visualize data

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 → 100

PCA & Data visualization Lab

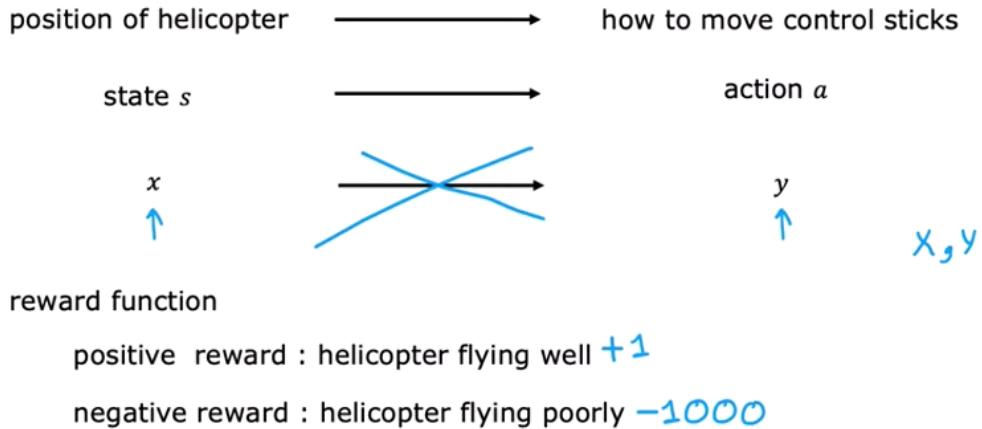
08 January 2024 08:39

What is Reinforcement Learning? (Controlling robots, Reward function, Don't need...)

09 January 2024 05:48

Controlling robots via supervised learning doesn't work well as its hard to get a good dataset

Reinforcement Learning



With a reward function, you reward good behaviour (not crashing) with +1 and punish bad behaviour (crashing) with -10.

You adjust your reward based on how good or bad the action was

You don't need a 'Y' correct output, you just need a reward

Action is things like 'moving to the left'

Applications:

Controlling robots

Factory Optimization

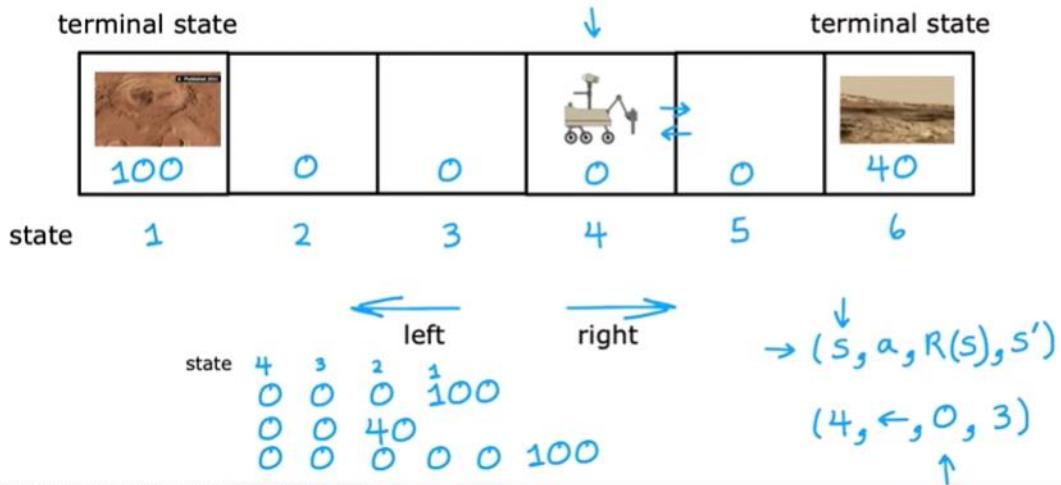
Financial (stock) trading

Playing games (including video games)

Mars Rover Example (Time step, Terminal)

09 January 2024 06:10

Mars Rover Example



At every time step the 'robot' is in a state, and it chooses an action to get a new state with the goal of reaching a reward state.

Terminal State is the final state, nothing more happens

The Return In Reinforcement Learning (Return, Discount Factor)

09 January 2024 06:23

Return

	100	0	0	0		40
state	1	2	3	4	5	6

$$\text{Return} = 0 + (0.9)0 + (0.9)^2 0 + (0.9)^3 100 = 0.729 \times 100 = 72.9$$

$$\text{Return} = R_1 + \gamma R_2 + \gamma^2 R_3 + \dots \text{ (until terminal state)}$$

$$\begin{aligned} \text{Discount Factor } \gamma &= 0.9 & 0.99 & 0.999 \\ &\gamma = 0.5 \end{aligned}$$

$$\text{Return} = 0 + (0.5)0 + (0.5)^2 0 + (0.5)^3 100$$

Negative Rewards:

The discount factor will push negative rewards as far into the future as possible

The return is the sum of all rewards that could be received by a series of actions weighted by the discount factor.

Discount Factor means the longer the model takes, the less rewards. Makes it impatient.

The lower we set the discount factor the more impatient it becomes because the rewards decrease more with time.

Your first reward is not discounted

With each state change the discount factor gets another indices (squared, cubed)

Example of Return

100	50	25	12.5	6.25	40
100	0	0	0	0	40
1	2	3	4	5	6

← return
← reward

$$\gamma = 0.5$$

The return depends on the actions you take.

100	2.5	5	10	20	40
100	0	0	0	0	40
1	2	3	4	5	6

$$0 + (0.5)0 + (0.5)^2 40 = 10$$

100	50	25	12.5	20	40
100	0	0	0	0	40
1	2	3	4	5	6

$$0 + (0.5)40 = 20$$

Making Decisions: Policies In Reinforcement Learning

09 January 2024 06:49

Policy

state s policy π action a

100	←	←	→	→	40
100	←	←	←	←	40
100	→	→	→	→	40
100	←	←	←	→	40

$$\begin{aligned}\pi(1) &= \text{left} \\ \pi(2) &= \text{left} \\ \pi(3) &= \text{left} \\ \pi(4) &= \text{left} \\ \pi(5) &= \text{right}\end{aligned}$$

A policy is a function $\pi(s) = a$ mapping from states to actions, that tells you what action a to take in a given state s .

Pie = Policy

Pie(S) = Action to take given state due to policy

A Policy describes what action to take in each state

The goal of reinforcement learning

100					40
-----	--	--	--	--	----

Find a policy π that tells you what action ($a = \pi(s)$) to take in every state (s) so as to maximize the return.

The goal of reinforcement learning is to find a policy that maximises returns

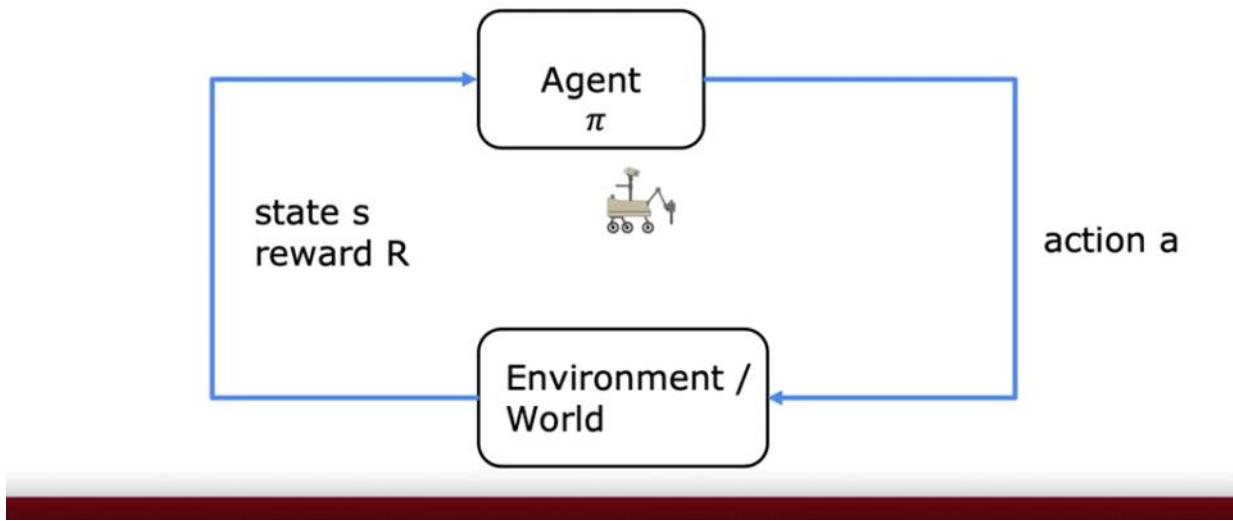
Review of key concepts (Markov Decision Process)

09 January 2024 07:02

	Mars rover	Helicopter	Chess
↙ states	6 states	position of helicopter	pieces on board
↙ actions	↔	how to move control stick	possible move
↙ rewards	100, 0, 40	+1, -1000	+1, 0, -1
↙ discount factor γ	0.5	0.99	0.995
↙ return	$R_1 + \gamma R_2 + \gamma^2 R_3 + \dots$	$R_1 + \gamma R_2 + \gamma^2 R_3 + \dots$	$R_1 + \gamma R_2 + \gamma^2 R_3 + \dots$
↙ policy π	100 ← → 40	Find $\pi(s) = a$	Find $\pi(s) = a$

Markov Decision Process (MDP):

Markov Decision Process (MDP)



Controlling an agent with a policy via actions to change the environment, changing our state and possibly reward

State Action Value Function Lab

10 January 2024 06:31



State-action
value fu...



State-action
value fu...

State-action value function definition (QSA, create policies)

09 January 2024 07:49

State action value function can be called Q function

S = state

A = action

State action value function (Q-function)

$Q(s, a) =$ Return if you
 • start in state s .
 • take action a (once).
 • then behave optimally after that.

$Q(s, a)$

← return
← action
← reward

100	50	25	12.5	20	40
100	0	0	0	0	40

$Q(2, \leftarrow) Q(2, \rightarrow)$

100	50	25	12.5	20	40
100	0	0	0	0	40

1 2 3 4 5 6

$$Q(2, \rightarrow) = 12.5 \\ 0 + (0.5)0 + (0.5)^2 0 + (0.5)^3 100 \\ Q(2, \leftarrow) = 50 \\ 0 + (0.5)100 \\ Q(4, \leftarrow) = 12.5 \\ 0 + (0.5)0 + (0.5)^2 0 + (0.5)^3 100$$

Q(S,A) is the state action value function

Q(S,A) = The return of starting in state S, taking an action then behaving optimally afterwards

Picking actions

→

100	50	25	12.5	20	40
100	0	0	0	0	40

← return
← action
← reward

→

100	50	25	12.5	12.5	10	6.25	20	40	40
100	0	0	0	0	0	0	0	40	40

1 2 3 4 5 6

$Q(4, \leftarrow) = 12.5$
 $Q(4, \rightarrow) = 10$

$\max_a Q(s, a)$
 $\pi(s) = a$

$Q(s, a) =$ Return if you
 • start in state s .
 • take action a (once).
 • then behave optimally after that.

The best possible return from state s is $\max_a Q(s, a)$.

The best possible action in state s is the action a that gives $\max_a Q(s, a)$.

Q^*

Optimal Q function

Q(S,A) can be used to create policies because it shows the best course of actions given a starting state, larger Q(SA) the better

The best possible return from state S is the highest Q(SA)

Q* and Optimal Q mean the same as Q

State-action value function example

09 January 2024 08:07



State-actio
n value fu...



State-actio
n value fu...

Bellman Equation (S,A,'Gamma, ...MaxQ)

09 January 2024 08:27

If you can compute the Q of SA, then you can make good policies

Remember Q(S,A) is the sum of all your rewards by taking an action and then following the best course of action.

You use the bellman Equation to calculate the Q(SA)

S = Current State

A = current action

' (prime) = the next action

Gamma = Discount Factor

a' = action that you take in state

s' = state you get after taking an action

Bellman Equation

$$Q(s, a) = R(s) + \gamma \max_{a'} Q(s', a')$$

100	100	50	12.5	25	6.25	12.5	10	6.25	20	40	40
100	0	0	0	0	0	0	0	0	0	40	40

$$\begin{aligned} s &= 2 \\ a &\Rightarrow \\ s' &= 3 \end{aligned}$$

$$\begin{aligned} Q(2, \rightarrow) &= R(2) + 0.5 \max_{a'} Q(3, a') \\ &= 0 + (0.5)25 = 12.5 \\ Q(4, \leftarrow) &= R(4) + 0.5 \max_{a'} Q(3, a') \\ &= 0 + (0.5)25 = 12.5 \end{aligned}$$

MaxQ is 'behaving optimally'

a' : Max of all possible actions (a') (the best of actions) calculate the Q

$$\max_{a'} Q(s', a')$$

Reminder:

Bellman Equation

$Q(s, a) =$ Return if you

- start in state s .
- take action a (once).
- then behave optimally after that.

s : current state
 a : current action

1	2	3	4	5	6
R(1)=100	R(2)=0	...	R(6)=40		

$R(s) =$ reward of current state

Bellman explained:

Explanation of Bellman Equation

$$\begin{aligned} Q(s, a) &= \text{Return if you} \\ &\quad \cdot \text{start in state } s. \\ &\quad \cdot \text{take action } a \text{ (once).} \\ &\quad \cdot \text{then behave optimally after that.} \\ \hookrightarrow \text{The best possible return from state } s' \text{ is } \max_{a'} Q(s', a') \\ Q(s, a) &= R(s) + \gamma \max_{a'} Q(s', a') \end{aligned}$$

Reward you get right away Return from behaving optimally starting from state s' .

$$R_1 + \gamma R_2 + \gamma^2 R_3 + \gamma^3 R_4 + \dots$$

$$Q(s, a) = R_1 + \gamma [R_2 + \gamma R_3 + \gamma^2 R_4 + \dots]$$

r(s) the reward you get right away

Y = discount factor

$\max Q(s, a')$ = Return from behaving optimally starting from state s'

Explanation of Bellman Equation

$$Q(s, a) = R(s) + \gamma \max_{a'} Q(s', a')$$

100	100	50	12.5	25	6.25	12.5	10	6.25	20	40	40
100	0	0	0	0	0	0	0	0	0	40	40

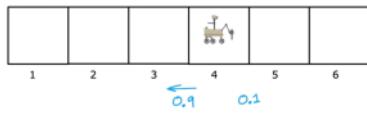
$$\begin{aligned} Q(4, \leftarrow) &= 0 + (0.5)0 + (0.5)^1 0 + (0.5)^2 100 \\ &= 0 + (0.5)(0 + (0.5)0 + (0.5)^1 100) \\ &= 0 + (0.5) m \end{aligned}$$

Random (stochastic) Environment (what expected, Return)

10 January 2024 06:14

Thing don't always manage to do as expected.

Stochastic Environment



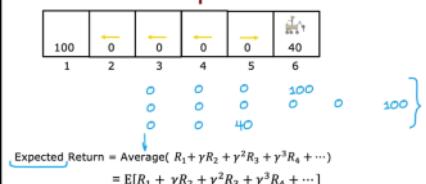
Take a robot who is going state 3 but slips into state 5

This is called a stochastic environment

In a stochastic environment the return is the average of your rewards. This is called your 'expected' return

E = Expected return

Expected Return



This is because the 'robot' might slip etc

Change of bellman:

Expected Return

Goal of Reinforcement Learning:

Choose a policy $\pi(s) = a$ that will tell us what action a to take in state s so as to maximize the expected return.

Bellman
Equation:

$$Q(s, a) = R(s) + \gamma E[\max_{a'} Q(s', a')]$$

↑ ↓
3 2 or 4

Bellman now uses the average expected path because we can't predict state with certainty

The Q values will drop because you can't control the object (robot) as well anymore due to environment

Example Of Continuous State Space Applications (dynamic, updated)

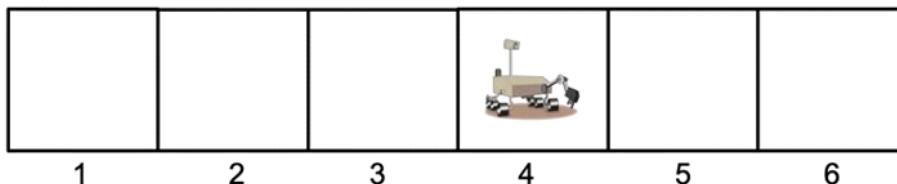
10 January 2024

06:45

A Continuous state means a dynamic state like driving, often shown by a numbers

Discrete vs Continuous State

Discrete State:



Continuous State:



$$s = \begin{bmatrix} x \\ y \\ \theta \\ \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix}$$

Continuous states are states that contain number like coordinates that are continuously calculated

Autonomous Helicopter



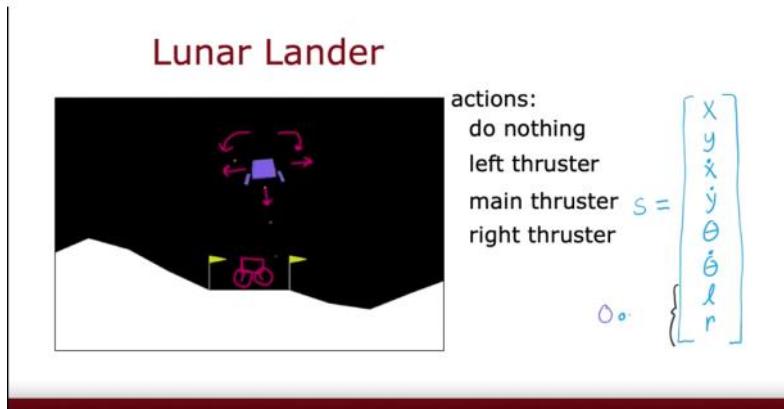
$$s = \begin{bmatrix} x \\ y \\ z \\ \phi \\ \theta \\ \omega \\ \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{\phi} \\ \dot{\theta} \\ \dot{\omega} \end{bmatrix}$$

You then take these Continuous state vectors and build a policy with them

Lunar Lander

10 January 2024 07:01

Lunar Lander Example:



Policy:

Goal is to learn a policy given our vector to maximize return

Set a high discount factor of 0.9 to not rush the activity

Continuous state vector:

X,Y

Velocity X & Y

Tilt

LegLandedLeft (Binary)

LegLandedRight (Binary)

Reward Function



- Getting to landing pad: 100 – 140
- Additional reward for moving toward/away from pad.
- Crash: -100
- Soft landing: +100
- Leg grounded: +10
- Fire main engine: -0.3
- Fire side thruster: -0.03

Reward Function:

Getting to landing pad 100-140 Depending on how good the flight was

Additional Reward for getting closer to the pad

Crash - 100

Soft Landing + 100

Leg Grounded + 10

Fire main engine -0.3

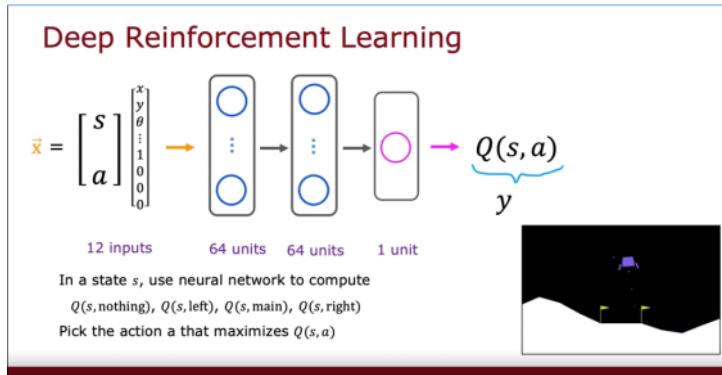
Fire side thruster -0.03

Put thought into what behaviours you want to reward for your task

Learning The State-Value Function

10 January 2024 08:05

Deep Reinforcement Learning:



Q(S,A) = The return of starting in state S, taking an action, then behaving optimally afterwards.

In Deep reinforcement learning networks, we take in the state vector as well as the completed action

The objective of the network is to produce an accurate Q(S,A) - State action value function

Because this is now a neural network, X is our state vector, Y is our Q(S,A)

This will work well because we can use the neural net to compute the Q(S,A) for all available actions and pick the highest one

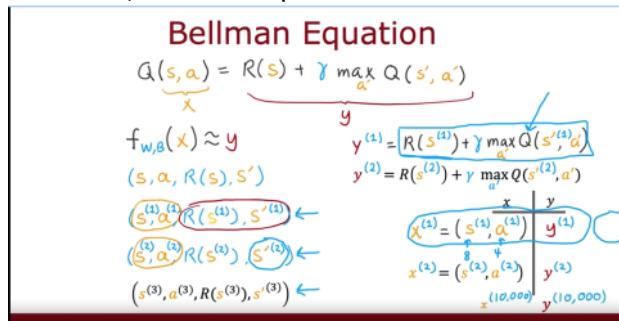
Reinforcement Learning Algorithm:

Initialize neural net with random guess of Q(S,A)

Repeat:

Take actions to get $(s, a, R(s), s')$ as replay buffer tuples

Store these 10,00 most recent tuples



s' is the series of states after to 'behave optimally', this element is trainable

Train:

Create a training set of tuples

$$X = (s, a)$$

$$Y =$$

$$y = R(s) + \gamma \max_{a'} Q(s', a')$$

Means: Max of all possible actions (a') (the best of actions afterwards) calculate the Q

This means 'behaving optimally' from current state, however Q is a trainable guess we are trying to improve

Learning Algorithm

Initialize neural network randomly as guess of $Q(s, a)$.

Repeat {

 Take actions in the lunar lander. Get $(s, a, R(s), s')$.

 Store 10,000 most recent $(s, a, R(s), s')$ tuples.



Replay Buffer

Train neural network:

 Create training set of 10,000 examples using

$$x = (s, a) \text{ and } y = R(s) + \gamma \max_{a'} Q(s', a')$$

 Train Q_{new} such that $Q_{new}(s, a) \approx y$.

 Set $Q = Q_{new}$.

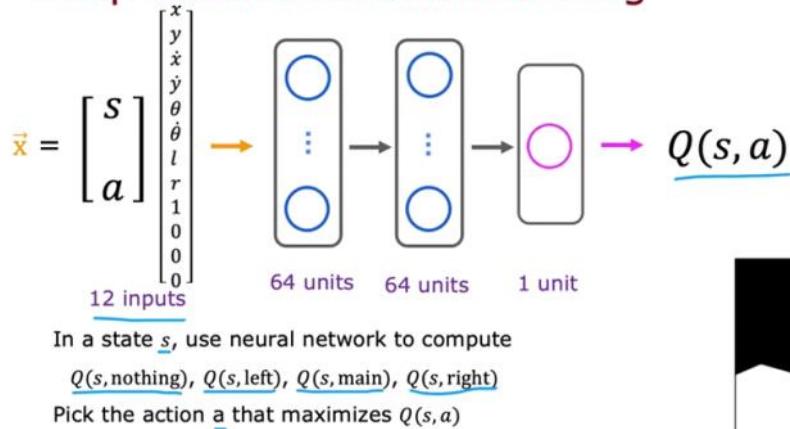
$$\begin{array}{ll} x, y & x^{(1)}, y^{(1)} \\ & \vdots \\ & x^{(10000)}, y^{(10000)} \end{array}$$

This is sometimes called DQN network.

Algorithm Refinement: Improved Neural Network Architecture

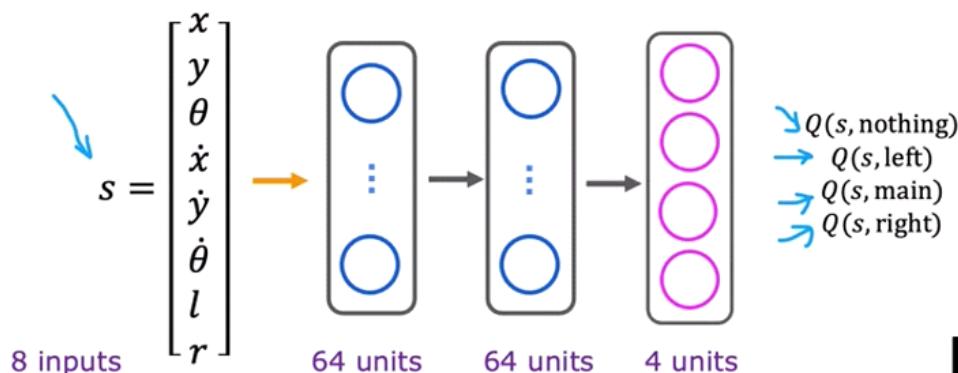
11 January 2024 05:39

Deep Reinforcement Learning



Running inference on all the different actions your model could take is inefficient

Deep Reinforcement Learning



In a state s , input s to neural network.

Pick the action a that maximizes $\underline{Q(s, a)}$. $R(s) + \gamma \max_{a'} Q(s', a')$



An Optimization is to only take in the state vector and have the model output the Q for all possible actions, then pick the highest one

Make it much more efficient

Algorithm Refinement: Epsilon-Greedy Policy

11 January 2024 06:01

Learning Algorithm

Initialize neural network randomly as guess of $Q(s, a)$.

Repeat {

 Take actions in the lunar lander. Get $(s, a, R(s), s')$.

 Store 10,000 most recent $(s, a, R(s), s')$ tuples.



Train model:

 Create training set of 10,000 examples using

$$x = (s, a) \text{ and } y = R(s) + \gamma \max_a Q(s', a').$$

 Train Q_{new} such that $Q_{new}(s, a) \approx y$. $f_{w.b}(x) \approx y$

 Set $Q = Q_{new}$.

Epsilon-greedy policy is used to take actions to build our reply buffer / dataset

How to choose actions while still learning?

In some state s

Option 1:

Pick the action a that maximizes $Q(s, a)$.

$Q(s, \text{main})$ is low



Option 2:

→ With probability 0.95, pick the action a that maximizes $Q(s, a)$. Greedy, "Exploitation"

→ With probability 0.05, pick an action a randomly. Exploration

ε -greedy policy ($\varepsilon = 0.05$)

0.95

Start ε high

1.0 → 0.01

Gradually decrease

You should most of the time pick the action with the highest $Q(S, A)$ even though $Q(S, A)$ is a learned estimate. This is called exploitation / greedy.

However, you should pick an action randomly (5% of the time) because this allows it to try new things if $Q(S, A)$'s parameters are wrong. which to start with they will be. This is called an exploration step

Overall this is called Epsilon greedy policy. Epsilon being how much you 'explore'.

You should start with a high Epsilon, explore a lot

Then decrease as you learn a better value for Q

Algorithm Refinement: Mini-Batch and Soft Updates (optional)

11 January 2024 06:37

How to choose actions while still learning?

x	y
2104	400
1416	232
1534	315
852	178
...	...
3210	870

$J(w, b) = \frac{1}{2m} \sum_{i=1}^m (f_{w,b}(x^{(i)}) - y^{(i)})^2$

$m = 100,000,000$

$m' = 1,000$

repeat {

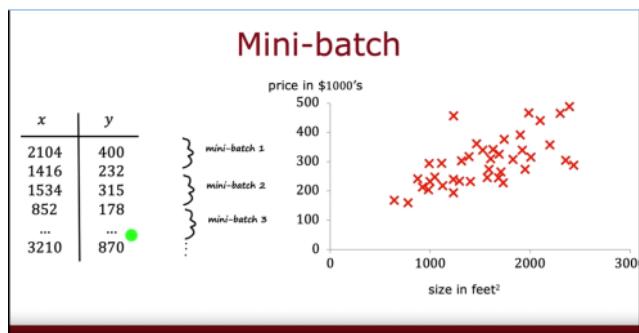
$w = w - \alpha \frac{\partial}{\partial w} \frac{1}{2m} \sum_{i=1}^m (f_{w,b}(x^{(i)}) - y^{(i)})^2$

$b = b - \alpha \frac{\partial}{\partial b} \frac{1}{2m} \sum_{i=1}^m (f_{w,b}(x^{(i)}) - y^{(i)})^2$

}

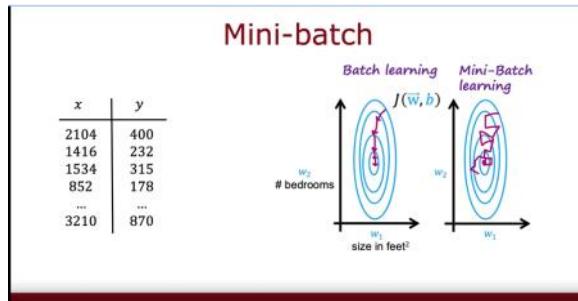
If you have a large dataset then using a squared error cost function will take ages

Because with each gradient descent step you must compute an average of your dataset for your derivative



We can make mini-batch which are a small subset of the data (about 1000 examples) to run gradient descent with

Mini-batch subsets are changed with each step of gradient descent



Mini batch gradient descent will converge into global minimum during gradient descent but it will take longer

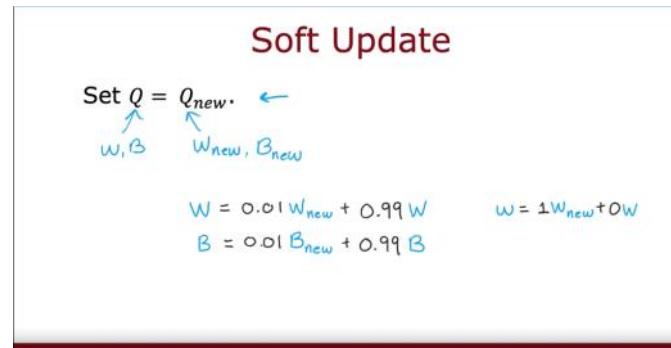
Mini-batch is much faster with a large dataset

When it comes to RNN's:

This means we can take 1000 tuples from our replay buffer each time to train the network

Soft updates:

The network will have some parameters Weights + bias



Soft updates means we only allow W and B to be changed by a small amount each training cycle

This stops us from going 'backwards' after a bad training cycle increasing odds of convergence.

The State Of Reinforcement Learning

11 January 2024 07:22

Reinforcement learning often works well in simulation but not great in the real world

Reinforcement learning is used less than supervised and unsupervised learning

Lots of great research going currently

Deep Q-Learning - Lunar Lander

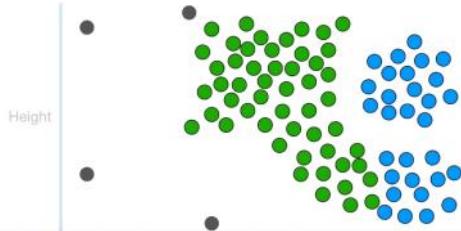
15 January 2024 06:19

By eye clustering can be done

If a cluster is nested (a cluster wraps around another) k-means will struggle to correctly cluster the data



Instead, because of the nesting, a simple clustering method might get something weird like this...



Dbscan is good for finding nested clusters within a high feature count dataset

By eye we cluster data by the density of points around them

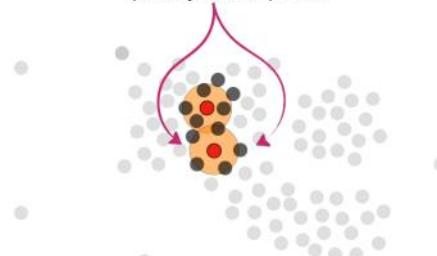
How DBSCAN works:

DBSCAN works by measuring the density around points by drawing a 'radius' around a random point

The size of the 'radius' is user defined - EPS



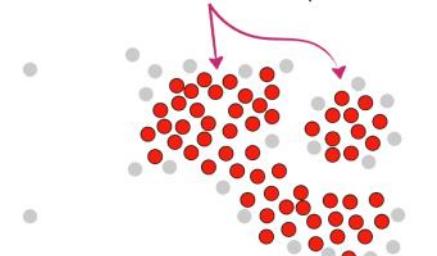
...is close to 5 other points because the orange circle overlaps, at least partially, 5 other points.



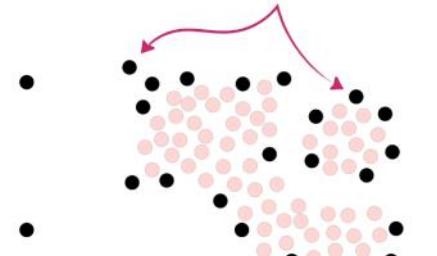
The center of these 'radius' are called Core points and the amount of points it takes for one to be declared is user defined. 4 is used in example below - MIN_SAMPLE



Ultimately, we can call all of these red points Core Points because they are all close to 4 or more other points...



...and the remaining points are Non-Core.



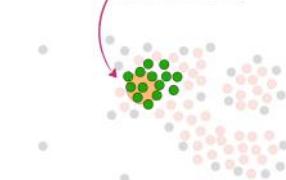
How DBSCAN clusters (part 2):

Now we have our core points we draw our radius around a random point.

All the core points within this radius join it. Then all there core points also join, creating a spread and formation of a cluster.



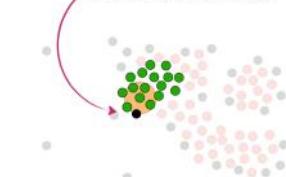
...and extend it to other Core Points that are close by.



We only ever add core points to the cluster



...and at this point, we only add the Core Points to the first cluster.

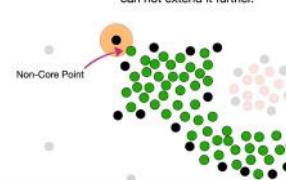


All the core points are added extending it further.

Then the non-core points that are within the radius of a core point are added to the cluster but cannot spread the cluster any further



So, unlike Core Points, Non-Core Points can only join a cluster. They can not extend it further.



Then we are done forming the first cluster

Then this process continues, starting by randomly picking a non-clustered core point to form a 2nd cluster

Any non-core points remaining are declared outliers and not added to cluster

Ties:

If a point could be in both clusters, whatever cluster 'claims' it first, keeps it.

Summary:

We picked a random first point to start the first cluster

Then neighbouring core points joined and extended the first cluster

Then non-core points only joined the first cluster and did not extend it any further

When to increase / decrease EPS:

Increase EPS when you have too many small clusters i.e labelled noise

Increase EPS on low density / spread out datasets

Decrease EPS when clusters are too large

Decrease EPS when there are close but unique clusters in your data

When to increase / decrease Min Sample:

Increase min_samples if you have too many clusters i.e labelled noise

Increase min_samples on large datasets with low density (values far away from each other)

Decrease min samples if important clusters are labelled as noise, decreasing it will allow more data points into the clusters

Decrease it if clusters are too sparse

Min samples rule: features + 1

HDBSCAN Avoids you needing to set EPS

To set min samples, use K-distance (Y AXIS):

<https://colab.research.google.com/drive/1zKJWgkz209WVu1Gp0ZGSSYf7ViQseek>

Elbow point can be used for EPS

Flat most point for min_sample_size

HDBSCAN

11 November 2024 21:31

This doesn't require you to set an EPS (Elision) , it calculates that for you:

<https://github.com/fkatada/sckit-hdbscan>

https://scikit-learn.org/stable/modules/generated/sklearn.cluster.HDBSCAN.html#sklearn.cluster.HDBSCAN.get_params

But you should still plot a k-distance graph to help you decide min_cluster_size, find the flattest point

[K Means Variance.ipynb - Colab](#)

Adjust k range (1,50) until it goes flat then elbow point is the best for min_cluster_size, in this case K = 13 is good, so min_sample_size = 13 looks good

Use an optics graph to help further

OPTICS (Ordering Points To Identify the Clustering Structure) is an alternative to DBSCAN that can handle varying densities more effectively, and its **reachability plot** is useful for visualizing density changes across the dataset.

- **How to Use:** Use `sklearn.cluster.OPTICS` to generate a reachability plot, which shows how reachable each point is from its nearest neighbors.
- **Interpretation:** The reachability plot displays "valleys" that indicate dense clusters and "peaks" that represent sparser regions. Setting **Min_Samples** based on the number of points in these valleys (i.e., dense regions) can give a good balance. If clusters are deep and well-defined, a lower **Min_Samples** may be appropriate. If the valleys are shallow, a higher **Min_Samples** can reduce noise.