**Big Data Analysis**

**Anomaly Detection in Network Traffic with K-means**

## Ji-Hyeong Han

### ( [jhhan@seoultech.ac.kr](mailto:jhhan@seoultech.ac.kr) )

Dept. of Computer Science and Engineering



**Contents**

1. Unsupervised Learning
2. K-Means Clustering
3. K-Means Clustering: Optimization Objective
4. K-Means Clustering: Random Initialization
5. K-Means Clustering: Choosing the Number of Clusters
6. Practice

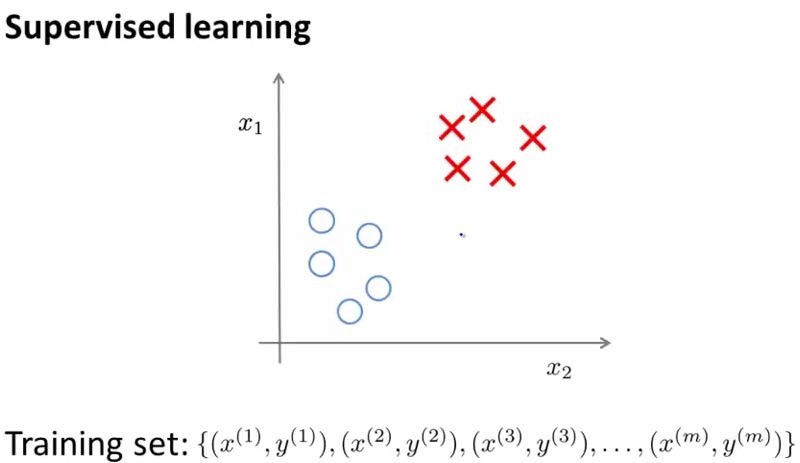
2

# Unsupervised Learning

3

**Supervised Learning**

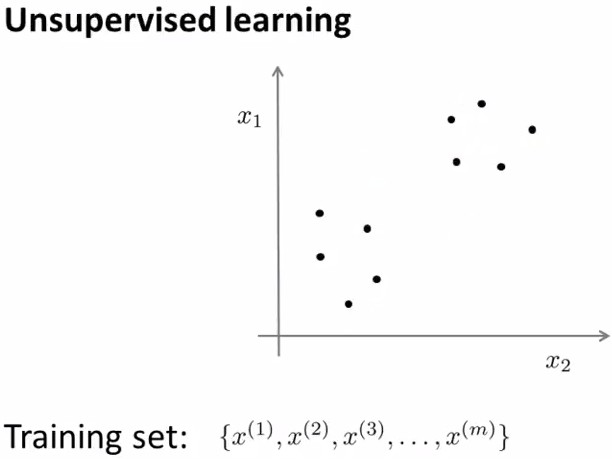
Decision Boundary



### Using labeled data (i.e. training data with right answer 𝑦𝑦)

4

**Unsupervised Learning**

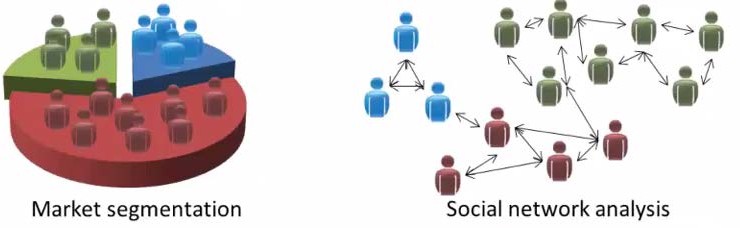


Clustering

### Using unlabeled data (i.e. training data with only 𝑥𝑥, without right answer 𝑦𝑦)

5

**Applications of Clustering**



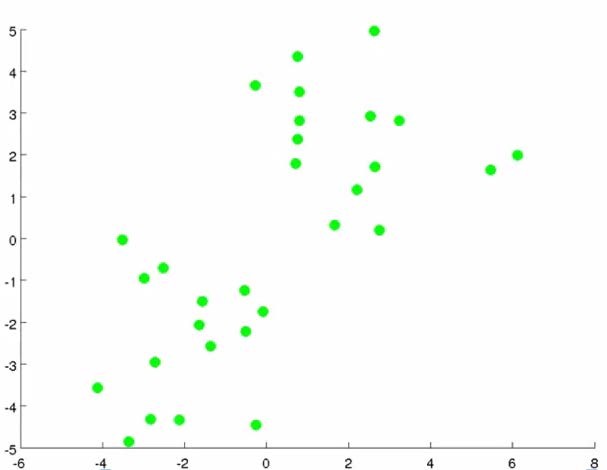
6

# K-Means Clustering

7

**K-Means Clustering Algorithm**

### We want to clustering these unlabeled data into two clusters

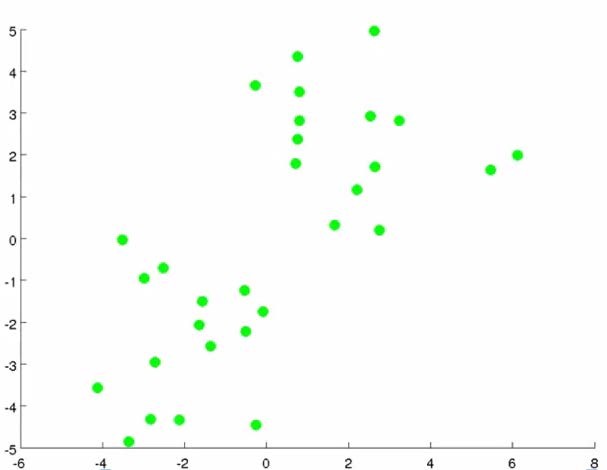


8

**K-Means Clustering Algorithm**

### Randomly initialize two points, called the cluster centroids

* + Why two points? Because, we want to group the data into two clusters



**X**

Cluster Centroid

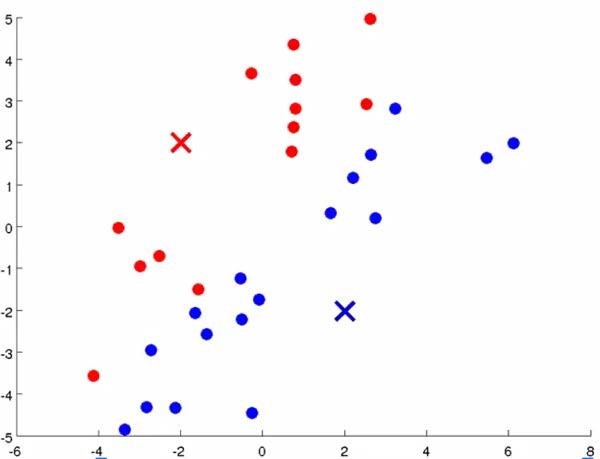
**X**

9

**K-Means Clustering Algorithm**

### K-means clustering algorithm is iterative algorithm of two steps

* + 1st step: assignment step
  + Each of the data points is going to be assigned one of two cluster centroids, depending on whether the data point is closer to the red cluster centroid or the blue cluster centroid

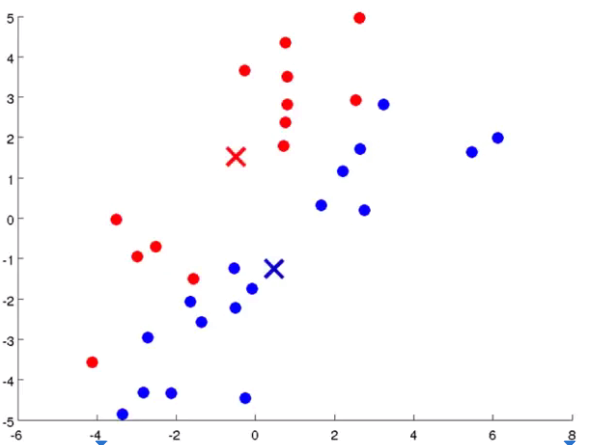
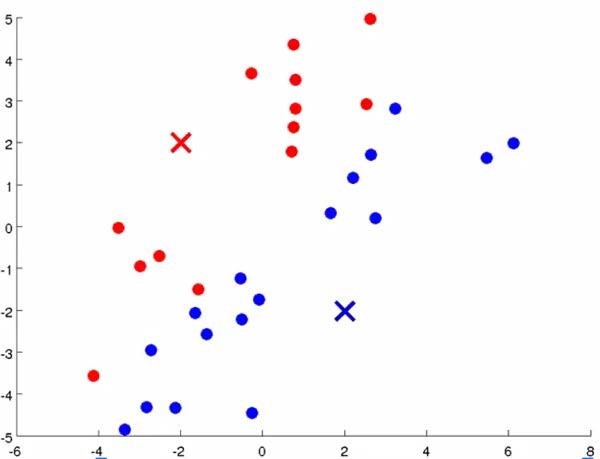
 10

**K-Means Clustering Algorithm**

### K-means clustering algorithm is iterative algorithm of two steps

* + 2nd step: move centroid step
  + Move red and blue cluster centroids to the average of the points colored the same color

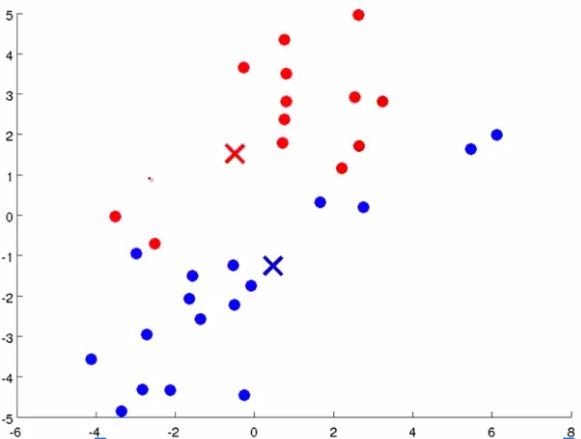
11



**K-Means Clustering Algorithm**

### K-means clustering algorithm is iterative algorithm of two steps

* + 1st step: assignment step
  + Repeat 1st step with the newly moved cluster centroids
  + So the colors of some points are changed

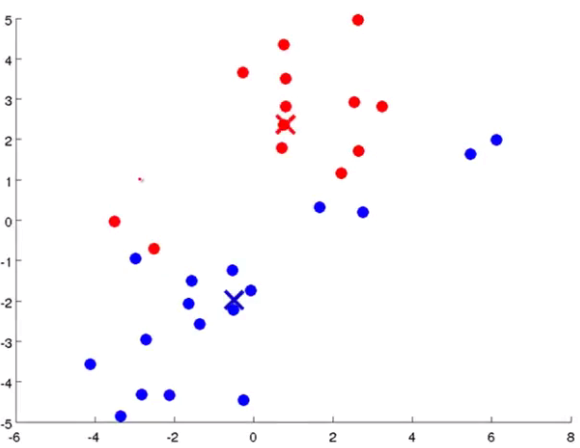
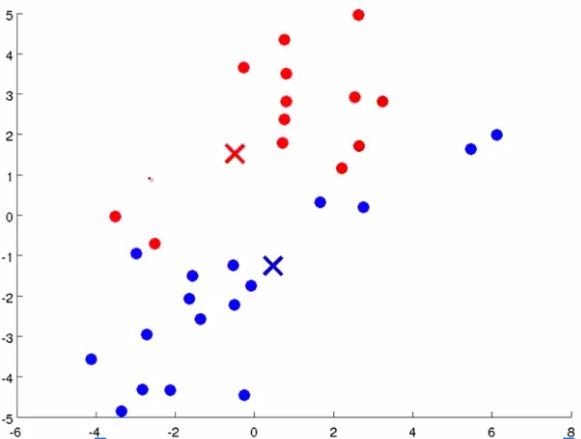


12

**K-Means Clustering Algorithm**

### K-means clustering algorithm is iterative algorithm of two steps

* + 2nd step: move centroid step
  + Repeat 2nd step with the newly colored data points



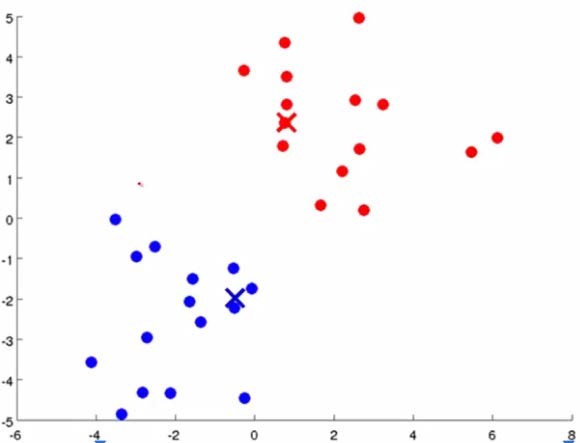
13

**K-Means Clustering Algorithm**

### K-means clustering algorithm is iterative algorithm of two steps

* + 1st step: assignment step
  + Repeat 1st step with the newly moved cluster centroids
  + So the colors of some points are changed

14

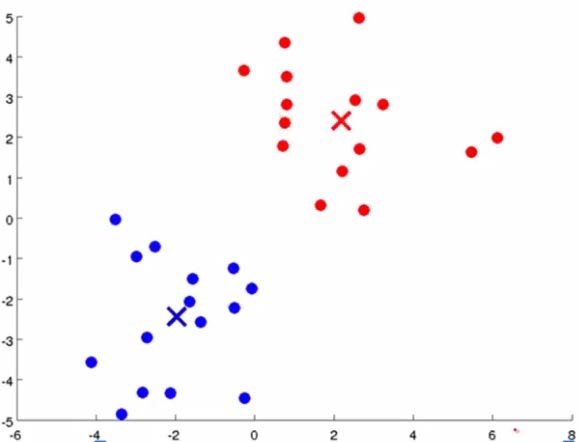


**K-Means Clustering Algorithm**

### K-means clustering algorithm is iterative algorithm of two steps

* + 2nd step: move centroid step
  + Repeat 2nd step with the newly colored data points

15



**K-Means Clustering Algorithm**

### Keep running iterations of K-means clustering algorithm until:

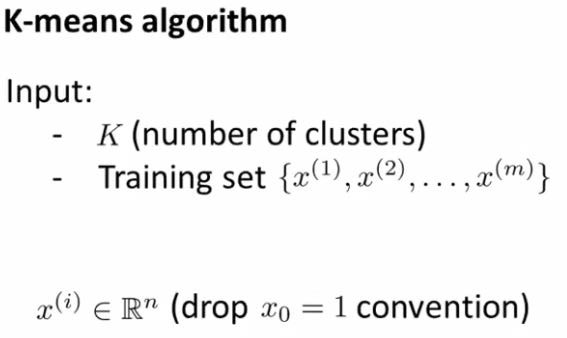
* + The cluster centroids will not change any further
  + The colors of the data points will not change any further

### At this point, K-means clustering has converged

16

**K-Means Clustering Algorithm**

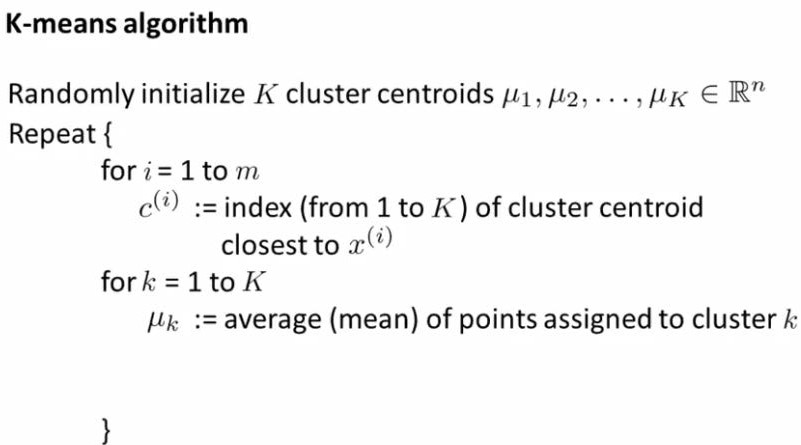
### Formal definition



17

### Formal definition

Assignment Step



Move Centroid Step

𝑐𝑐𝑖𝑖 = min

𝑘𝑘

𝑥𝑥𝑖𝑖 − 𝜇𝜇𝑘𝑘

**K-Means Clustering Algorithm**

2

18

**K-Means Clustering Algorithm**

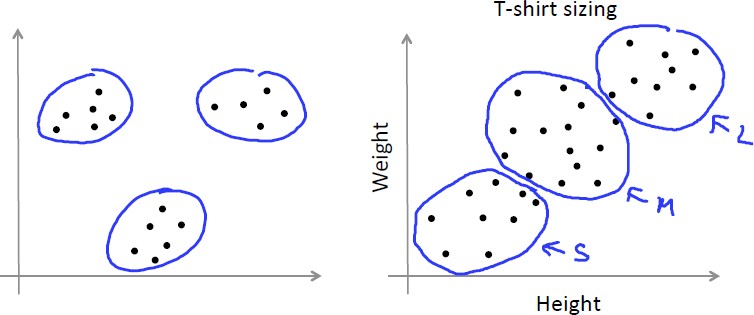
### If there is a cluster centroid with no data points assigned

* + This case is pretty common in real practice
  + Eliminating that cluster centroid
  + As a result, we are going to have (K-1) clusters at the end, instead of K clusters

19

**K-Means Clustering Algorithm**

### K-means clustering for non-separated clusters

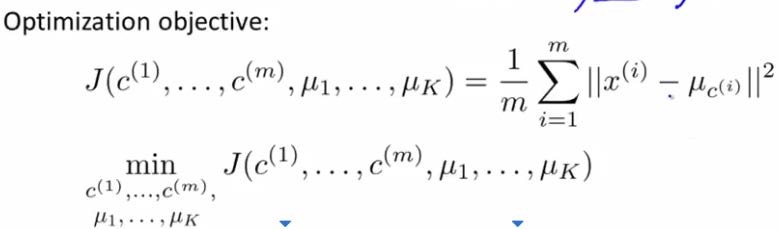
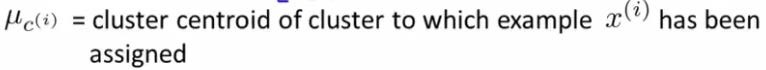
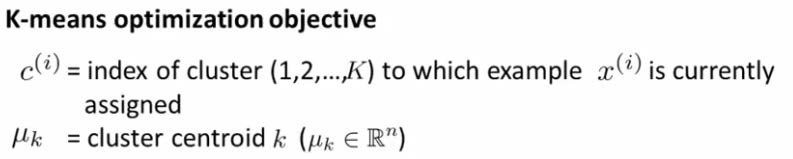


20

# K-Means Clustering: Optimization Objective

21

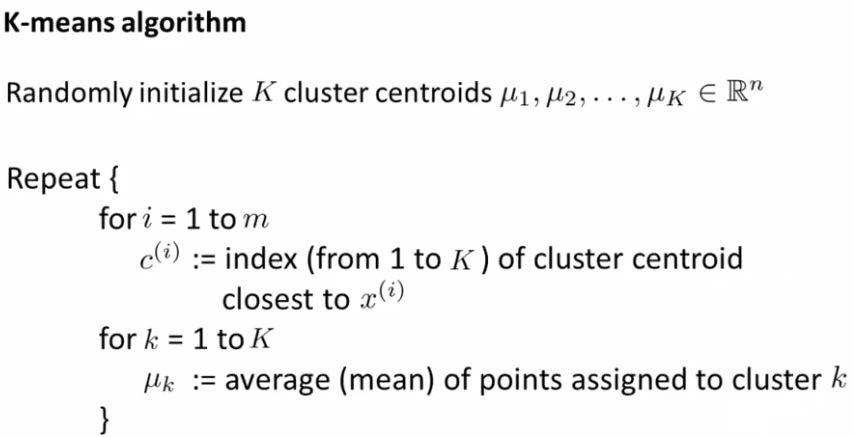
**K-Means Clustering Optimization Objective**



The cost function 𝐽𝐽 of K-means clustering algorithm is often called Distortion Function

22

**K-Means Clustering Optimization Objective**

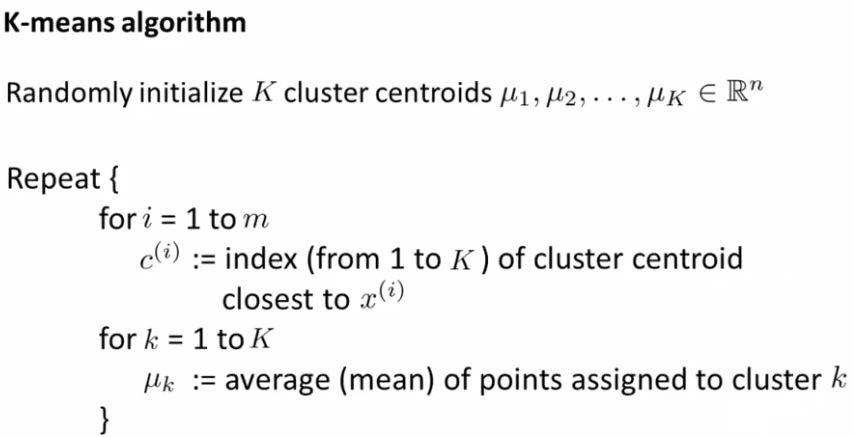


:: Cluster Assignment Step

Minimize cost function 𝐽𝐽 with respect to 𝑐𝑐1, 𝑐𝑐2, … , 𝑐𝑐𝑚𝑚 (while holding 𝜇𝜇1, … , 𝜇𝜇𝑘𝑘 fixed)

23

**K-Means Clustering Optimization Objective**



:: Move Centroid Step

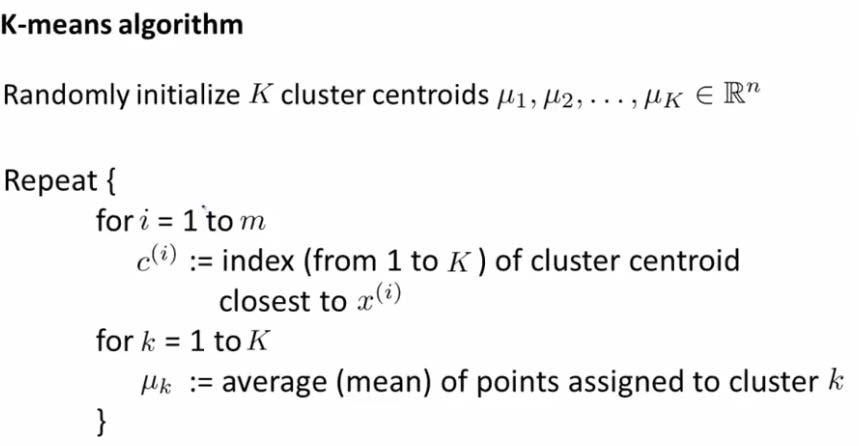
Minimize cost function 𝐽𝐽 with respect to 𝜇𝜇1, … , 𝜇𝜇𝑘𝑘

24

# K-Means Clustering: Random Initialization

25

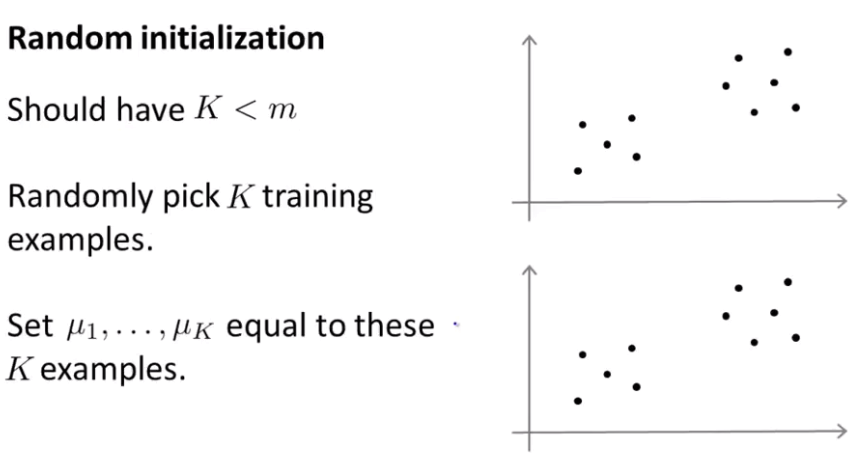
**K-Means Clustering Random Initialization**



26

**K-Means Clustering Random Initialization**

### Assume K=2

**X**

**X**

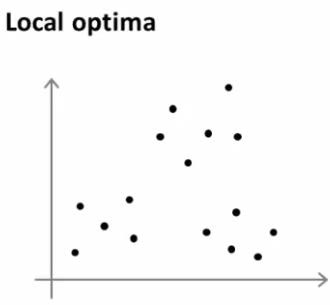
**X X**

27

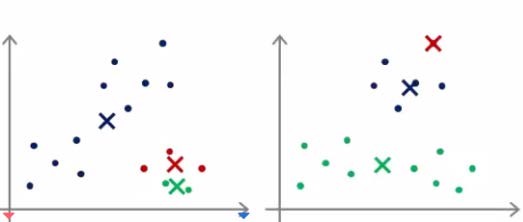
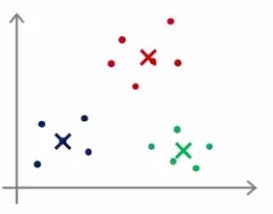
**K-Means Clustering Random Initialization**

### K-means clustering can end up converging to different solutions depending on exactly how the cluster centroids were initialized (i.e. depending on the random initialization)

* In particular, K-means clustering can end up at local optima

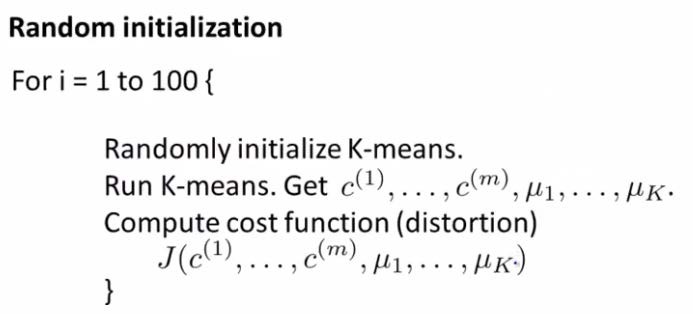


28



**K-Means Clustering Random Initialization**

### To avoid getting stuck in local optima, try multiple random initializations



* Multiple random initialization shows good performance when K is small (ex: K=2~10)
* If K is very large (ex: K>10), multiple random initialization is less likely to make a huge difference.
  + There is a higher chance that your first random initialization will give you a pretty descent solution already 29

# K-Means Clustering:

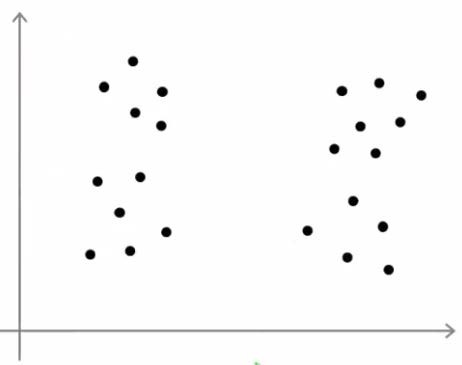
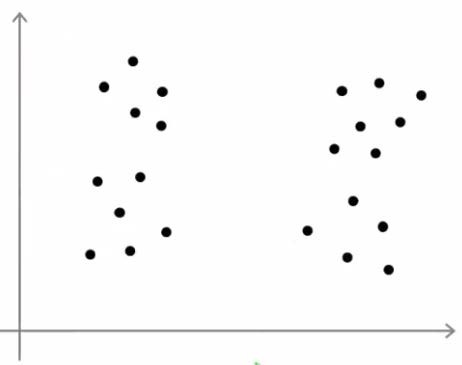
**Choosing the Number of Clusters**

30

**How to Choose the Number of Clusters, K**

### Choose the K manually

* + By looking at visualization

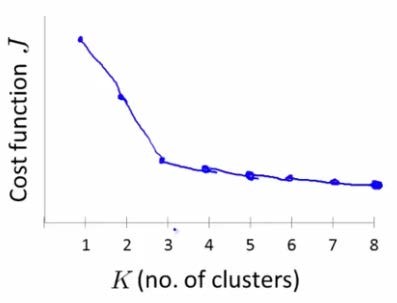


### There isn’t one right answer

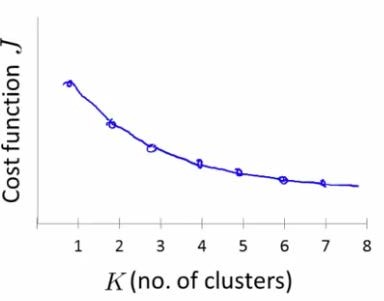
* + Because it is unsupervised learning
  + Because we don’t have labels
  + So, there isn’t always a clear cut answer 31

**How to Choose the Number of Clusters, K**

### Elbow method



Elbow



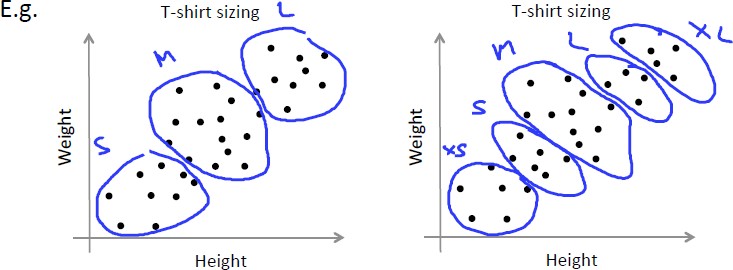
Elbow ??

* + Elbow method is worth to try, but not necessarily

32

**How to Choose the Number of Clusters, K**

### Sometimes, we are running K-means clustering to get cluster to use for some later/downstream purpose

* In that case, we can evaluate K-means clustering based on a metric for how well it performs for that later purpose
* T-shirt sizing example
  + K=3 or K=5
  + Evaluate how well performing for selling purpose, from the perspective of T-shirt business
    - Do I want to have more T-shirt sizes so that my T-shirts fit customers better?
    - Do I want to have fewer T-shirt sizes so that I can sell them to customers more cheaply?

33

# Practice

34

**Data Set**

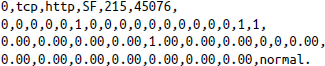
### KDD Cup 1999 data set

* + To detect anomalous network traffic

### Download the data

* + <http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html>
  + Download “kddcup.data\_10\_percent.gz”
  + See “kddcup.names” : A list of features

### Data example



* + TCP connection to an HTTP service – 215 bytes were sent and 45,706 bytes were received

### Build K-means cluster models to find potentially new and unknown attacks

35

**A First Take on Clustering**

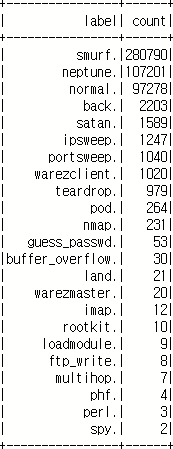
### Read the data sets

36

**A First Take on Clustering**

### Explore the data set



- There are 23 distinct labels,

and the most frequent are smurf. and neptune. attacks

37

**A First Take on Clustering**

### Note that the data contains nonnumeric features

* + For example, the second column may be tcp, udp, or icmp, but K-means clustering requires numeric features
  + The final label column is also nonnumeric and K-means clustering is unsupervised learning, so it doesn’t need the label
  + To begin, these nonnumeric columns will simply be ignored

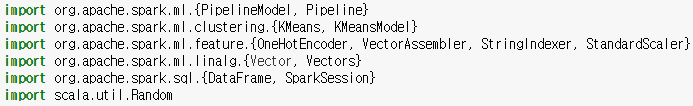


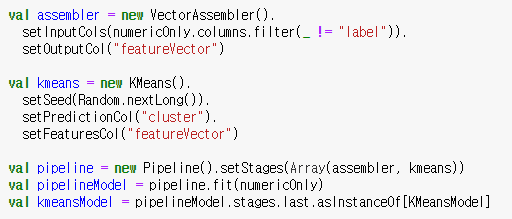
38

**A First Take on Clustering**

### Creating K-means clustering of the data follows the same pattern as Decision Tree model, i.e. *VectorAssembler*

* A *VectorAssembler* creates a feature vector, a *KMeans* implementation creates a model from the feature vectors, and a *Pipeline* stitches it all together



39

**A First Take on Clustering**

### From the resulting model, we can extract and examine the cluster centers



* It’s not easy to interpret the numbers intuitively
* But each of these represents the centroid of each cluster that the model produced
  + The values are the coordination of the centroid in terms of each of the numeric input features

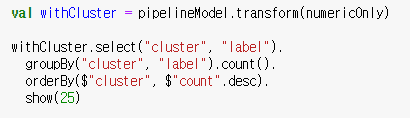
### Two vectors are printed, meaning K-means was fitting k=2 clusters to the data

* + But it is not enough to accurately model the distinct grouping within the data

40

**A First Take on Clustering**

### It is a good way to use the given labels to get an intuitive sense of what went into these two clusters by counting the labels within each cluster



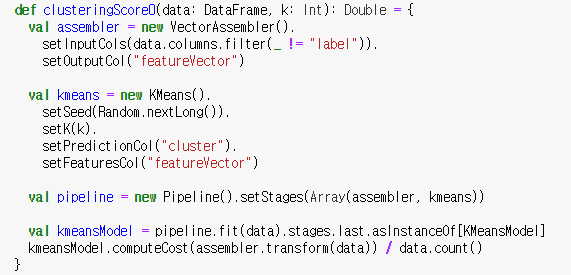
* + The result shows that the clustering was not at all helpful
  + Only one data point ended up in cluster 1

41

**Choosing K**

### We need to calculate the cost using *computeCost* method offered by

*KMeansModel* in Spark to evaluate the quality of a clustering

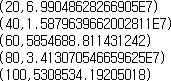
* + There is no simple *Evaluator* implementation for K-means clustering in Spark (not like those available to compute multiclass classification metrics)
  + It’s simple enough to manually evaluate the clustering cost for several values of *k*

42

**Choosing K**

* Now using the defined *clusteringScore0* method, we can evaluate the different *k* values





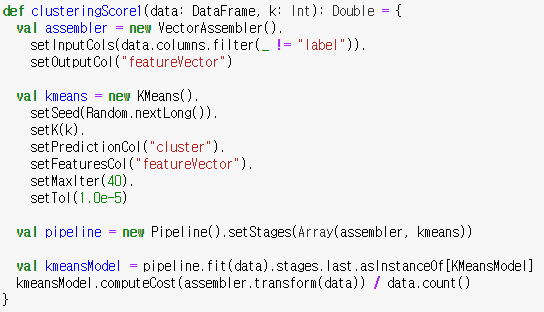
* + The printed result shows that the cost decreases as *k* increases
  + But, in the preceding results, the cost for k=80 is higher than k=60
  + This shouldn’t happen because higher *k* always permits at least as good a clustering as a lower *k*
  + So, we can think that in this clustering when k=80, the algorithm is stuck in the suboptimal clustering, or it may have stopped early before it reached its local optimum

43

**Choosing K**

### We can improve it by running the iteration longer

* + The K-means clustering algorithm in Spark has a threshold via *setTol*() that controls the minimum amount of cluster centroid movement considered significant
  + Increasing the maximum number of iterations with *setMaxIter*() also prevents it from potentially stopping too early



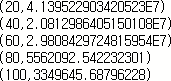
Increase from default 20

Decrease from default 1.0e-4

44

**Choosing K**

### This time, at least the costs decrease consistently



* Still we want to find a point, “elbow” in a graph of *k* versus cost

45

**Feature Normalization**

### We can normalize each feature by converting it to a standard score

* + Subtracting the mean of the feature’s values from each value, and dividing by the standard deviation



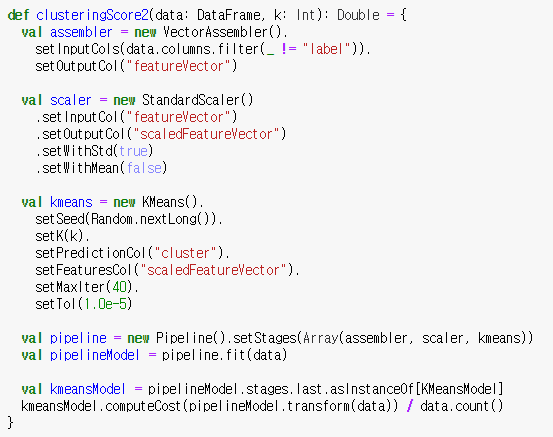
### In Spark, MLlib provides *StandardScaler* and it can be easily added to the clustering pipeline

* In fact, subtracting has no effect on the clustering because the subtraction effectively shifts all the data points by the same amount in the same direction
  + This does not affect interpoint Euclidean distances

46

**Feature Normalization**

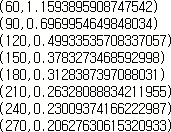
### We can define the new evaluation method with normalized data



47

**Feature Normalization**

### We can see the result on a higher range of k



* + This has helped put dimensions on more equal footing, and the absolute distances between points is much smaller in absolute terms
  + But, there isn’t yet an obvious value of k

48

**Categorical Variables**

### We left out several features because they aren’t numeric

* + Adding them back in some form should produce a better-informed clustering

### The categorical features can translate into several binary indicator features using one-hot-encoding, which can be viewed as numeric dimensions

* + For example, the second column contains the protocol type: tcp, udp, or icmp
  + TCP might be [1, 0, 0]; UDP might be [0, 1, 0]; and so on.

### In Spark, MLlib provides components that implement this one-hot- encoding

* + When we deal with string type value to one-hot-encoding, we need two steps

1. String values are converted to integer indices like 0,1,2 using

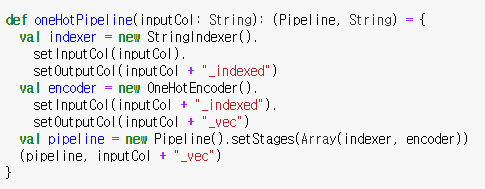
*StringIndexer*

1. These integer indices are encoded into a vector using *OneHotEncoder*

49

**Categorical Variables**

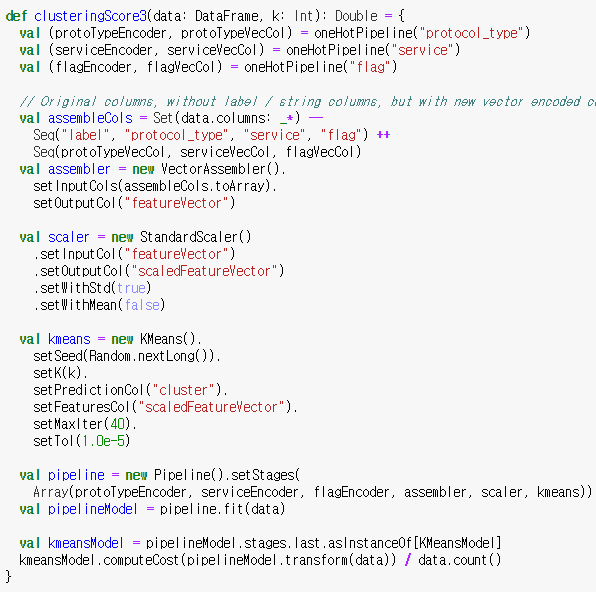
### Implement a Pipeline for one-hot-encoding



50

**Categorical Variables**

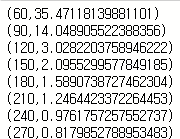
* Define new cost method with all of the data including categorical variables

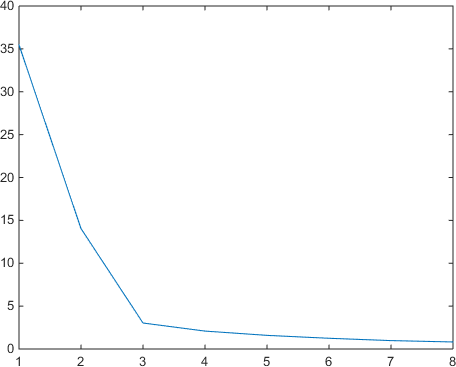
51

**Categorical Variables**

### Run and see the result







Elbow point: k=120

52

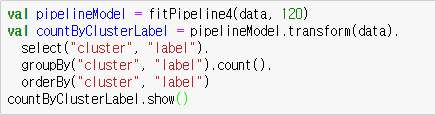
**Clustering in Action**

### We can cluster the full normalized data set with k=120

53

**Clustering in Action**

### cont’d



54

**Clustering in Action**

### Now we can make an actual anomaly detector

* Anomaly detection amounts to measuring a new data point’s distance to its nearest centroid
  + If this distance exceeds some threshold, it is anomalous
  + This threshold might be chosen to be the distance of, say, the 100th- farthest data point from among known data

55

**Clustering in Action**

### Implement anomaly detector



* + Network security expert would be more able to interpret why this is or is not actually a strange connection. It appears unusual at least because it is labeled

normal, but involves connections to 51 different hosts 56