哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

Big data technology and Practice

李春山

2020年9月22日

内容提要

Chapter 7: ML - Logistic Regression

Chapter 8: ML K-Means Clusters

Chapter 7: ML – Logistic Regression

- The most common supervised learning tasks are regression (predicting values) and classification (predicting classes).
- Now we will turn our attention to classification systems.
- Not all labels are continuous, sometimes you need to predict categories, this is known as classification

Logistic Regression

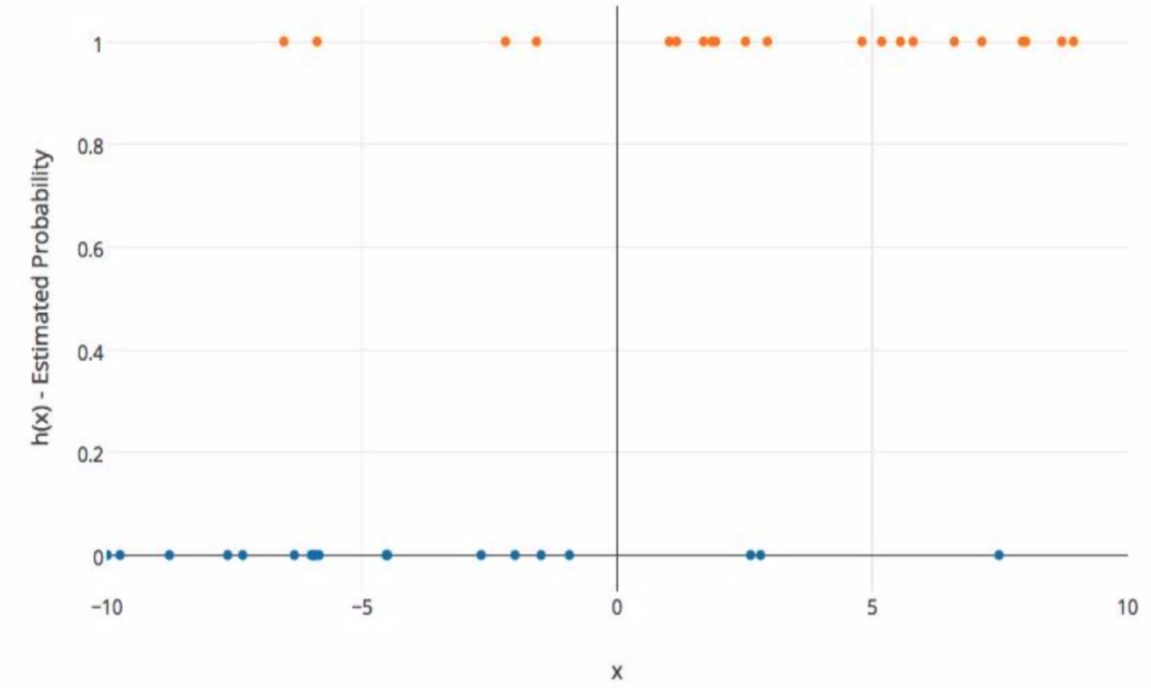
- At first, consider a method for binary classification (problems with two class values). Some examples of classification problems:
  - -Spam versus “Ham” emails
  - -Loan Default (yes/no)
  - -Disease Diagnosis

- Then, discover the logistic regression algorithm for machine learning. It’s one of the basic ways to perform classification .

Logistic Regression

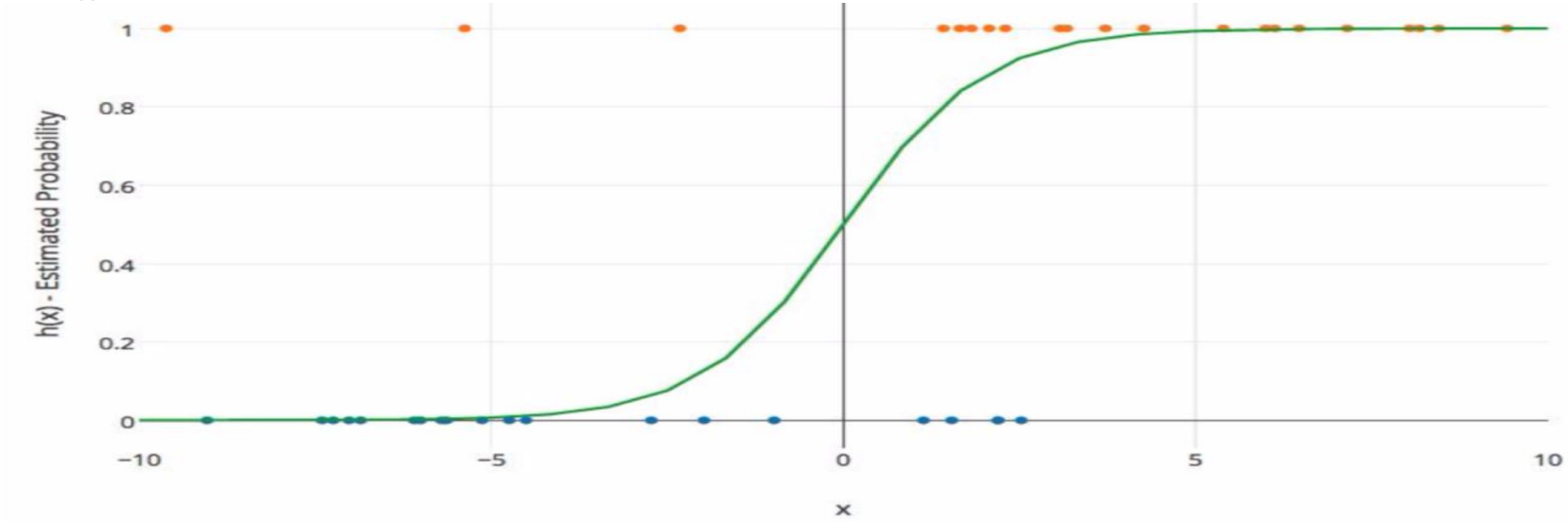
- 1. The basic idea for logistic regression
  - The convention for binary classification is to have two classes 0 and 1.
  - Imagine we plotted out some categorical data against one feature.

- The X axis represents a feature value and the Y axis represents the probability of belonging to class 1.



It is noted:

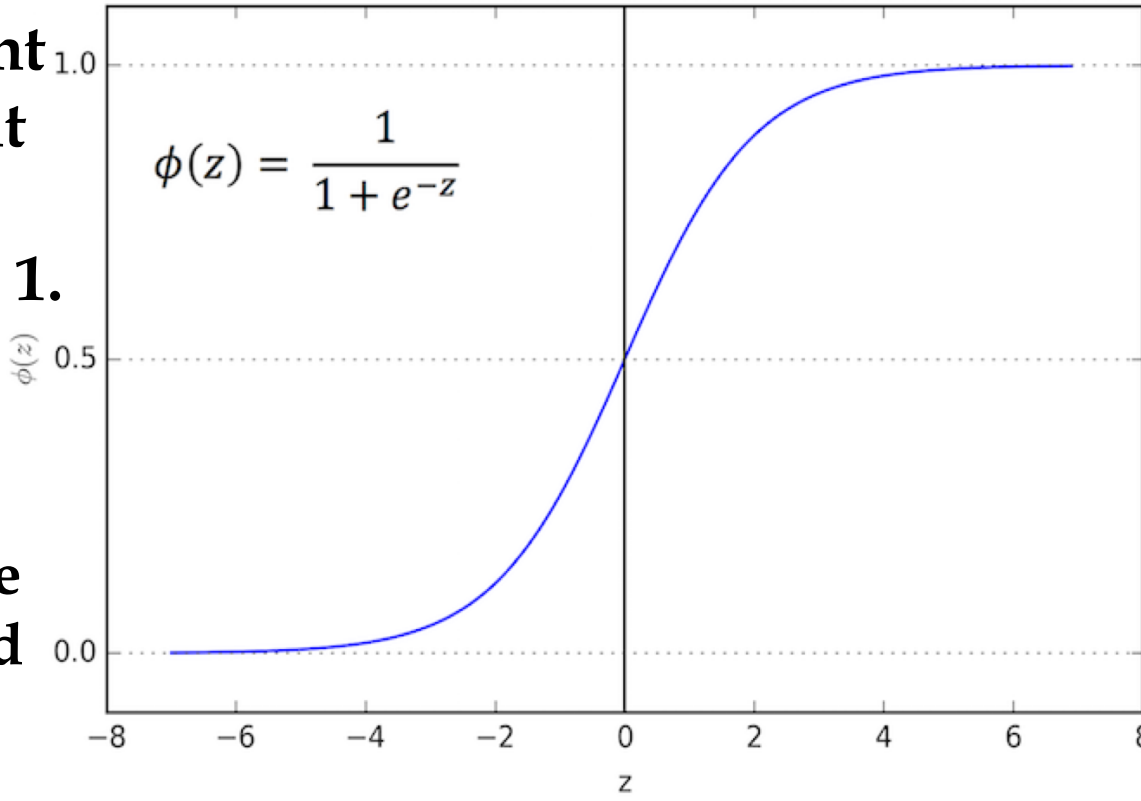
- **We can't use a normal linear regression model on binary groups. It won't lead to a good fit:**
- **A new function is needed to fit binary categorical data!**
- **Sigmoid function:**
  - The Sigmoid (aka Logistic) Function takes in any value and outputs it to be between 0 and 1.



Logistic Regression

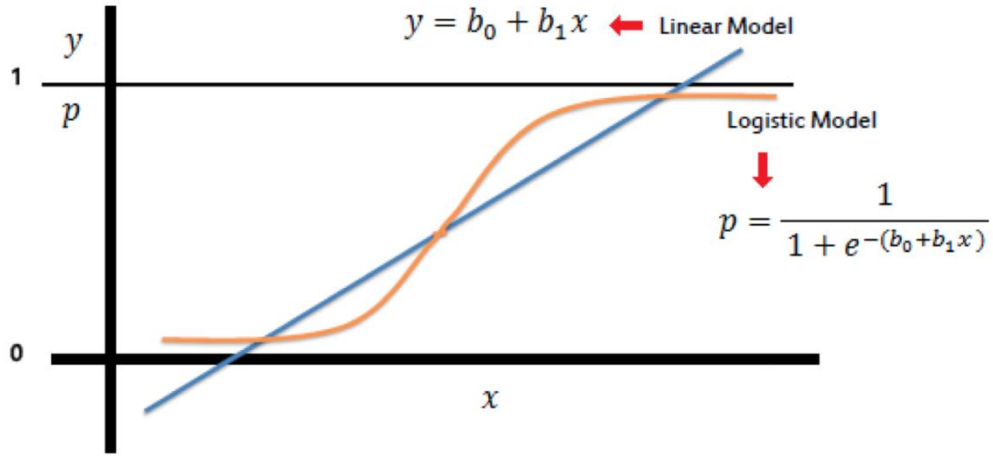
- We can set a cutoff point at 0.5, anything below it results in class 0, anything above is class 1.

- So, we use the logistic function to output a value ranging from 0 to 1. Based on this probability p , we assign a class: p >=0.5 for class 1, otherwise, class 0.



Logistic Regression

- It means we can take our Linear Regression Solution and place it into the Sigmoid Function.



- So, we can use Sigmoid Function as a ML model with parameters b0 and b1 , called Logistic Model.

Next, let us look at some examples for training a logistic regression model on some training data.

Using Spark MLlib

- **https://spark.apache.org/docs/latest/api/python/pyspark.mllib.html#modulepyspark.mllib.classification**

pyspark.mllib.classification module

class `pyspark.mllib.classification.`**`LogisticRegressionModel`**(*weights*, *intercept*, *numFeatures*, *numClasses*) [source]

Classification model trained using Multinomial/Binary Logistic Regression.

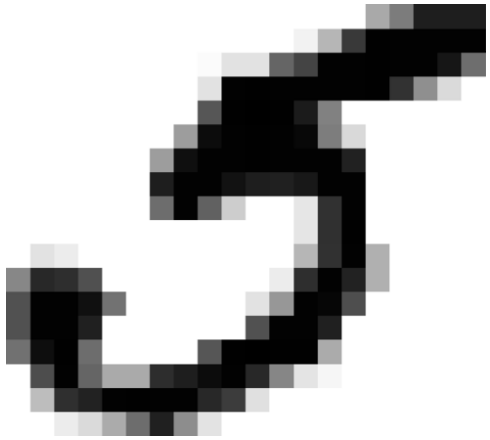
Parameters:
- **weights** – Weights computed for every feature.
- **intercept** – Intercept computed for this model. (Only used in Binary Logistic Regression. In Multinomial Logistic Regression, the intercepts will not bea single value, so the intercepts will be part of the weights.)
- **numFeatures** – The dimension of the features.
- **numClasses** – The number of possible outcomes for k classes classification problem in Multinomial Logistic Regression. By default, it is binary logistic regression so numClasses will be set to 2.

Using Spark MLlib

- **Let's see an example of how to run a logistic regression with Python and Spark!**
- **An example:**
- **2. Using Scikit-Learn**
- **So, we can use Sigmoid Function as a ML model with parameters b0 and b1 , called Logistic Model.**
- **Each image is labeled with the digit it represents.**

Using Spark MLlib

- **Let's simplify the problem for now and only try to identify one digit, 5.**
- **This "5-detector" will be an example of abinary classifier, distinguishing between just two classes, 5 and not-5.**



- **Understanding the dataset:**
  - There are 70,000 images, and each image has 784 features. This is because each image is 28~28 pixels, and each feature simply represents one pixel's intensity, from 0 (white) to 255 (black).
  - Let's see an example of how to run a binary classifier (SGDClassifier) with Python with Scikit-Learn.
  - The example, Binary_Classifier.ipynb is given in …/Labs/Labs

Using Spark MLlib

- **4. Performance Measures**
  - Evaluating a classifier is often significantly trickier. There are many performance measures available. We discuss two well known ones.
  - 1). Confusion Matrix
    - It helps us to find the accuracy of the model and avoid overfitting.
    - This is how "confusion matrix" looks like:

		Actual class	
		Cat	**Non-cat**
Predicted class	**Cat**	5 True Positives	2 False Positives
	Non-cat	3 False Negatives	17 True Negatives

Two classes:
- **Cat**
- **Non-cat**

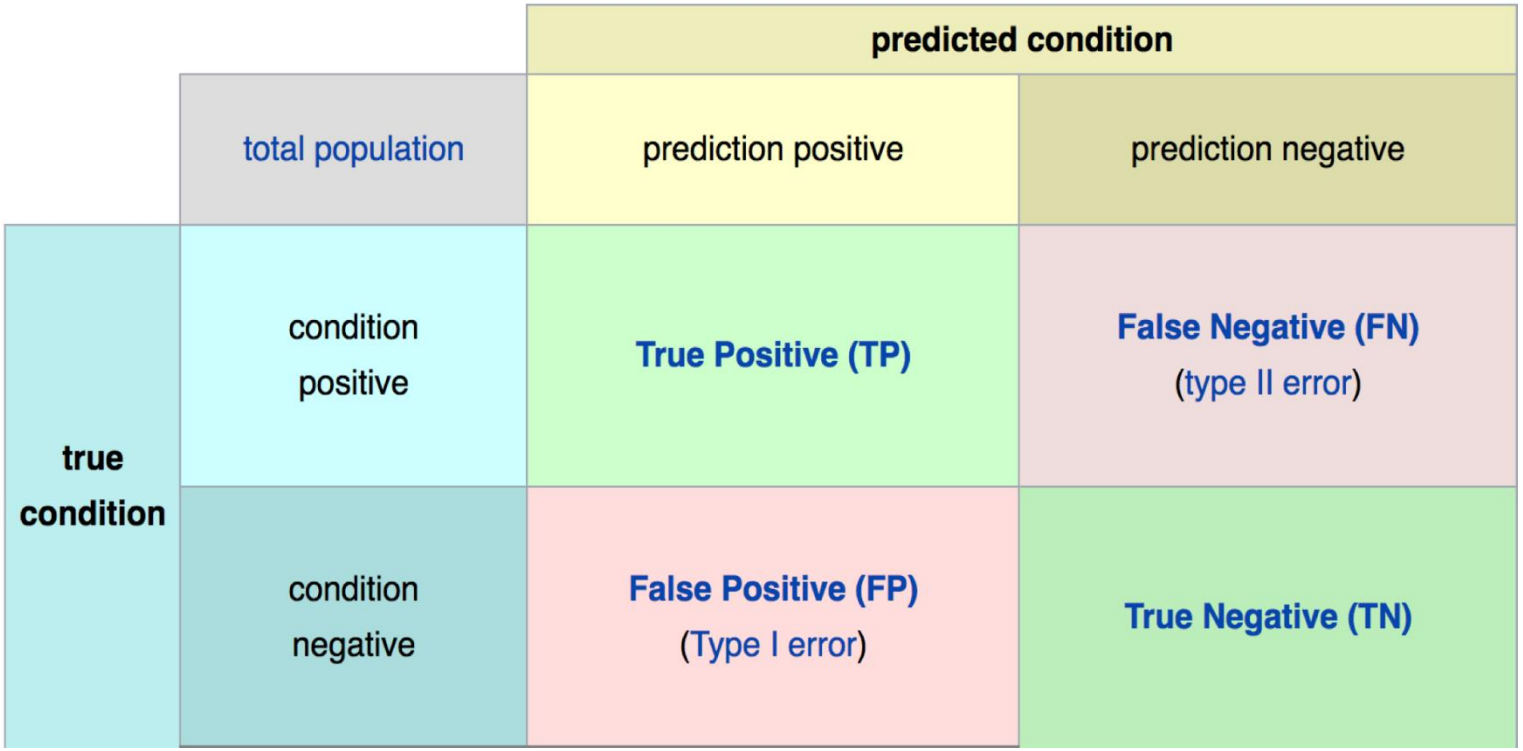
Totally, 27 instances for training

Using Spark MLlib

- You can calculate the accuracy of your model with:

$$\frac{\overset{5}{\text{True Positive}} + \overset{17}{\text{True Negatives}}}{\text{True Positive} + \text{True Negatives} + \text{False Positives} + \text{False Negatives}} = 22/27 = 0.8148$$

5 17 3 2



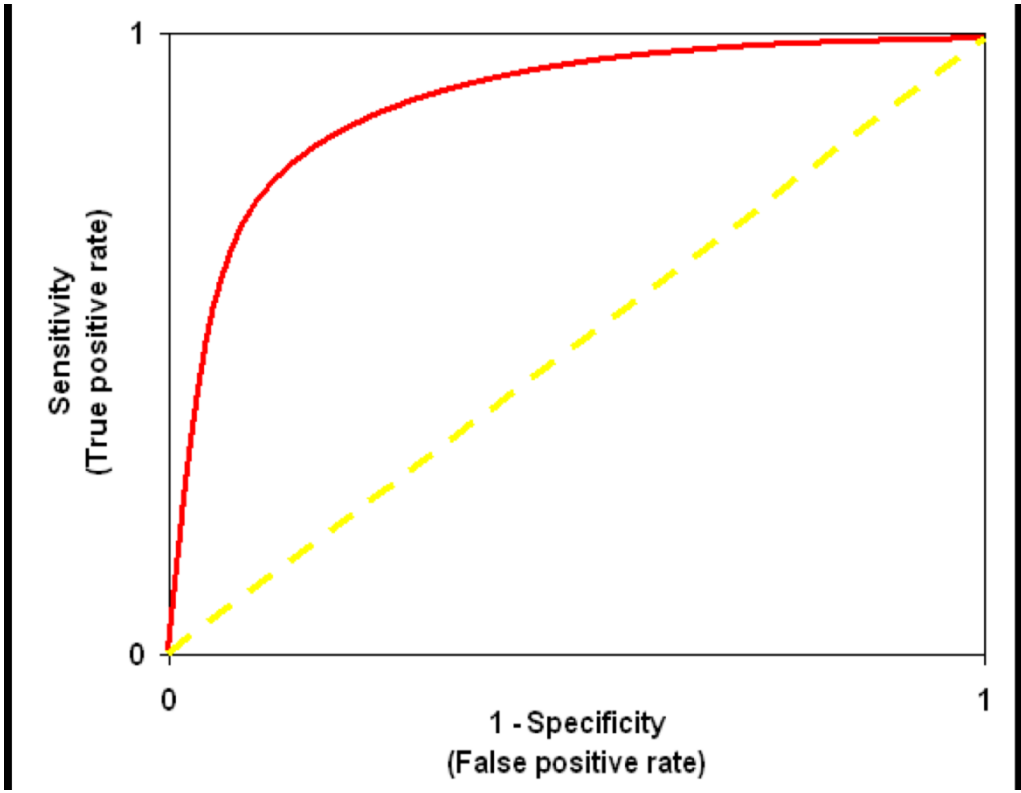
Using Spark MLlib

- **From confusion matrix, other performance measures can be derived also:**

		predicted condition		
	total population	prediction positive	prediction negative	Prevalence $= \frac{\Sigma \text{ condition positive}}{\Sigma \text{ total population}}$
true condition	condition positive	**True Positive (TP)**	**False Negative (FN)** (type II error)	True Positive Rate (TPR), Sensitivity, Recall, Probability of Detection $= \frac{\Sigma \text{ TP}}{\Sigma \text{ condition positive}}$
	condition negative	**False Positive (FP)** (Type I error)	**True Negative (TN)**	False Positive Rate (FPR), Fall-out, Probability of False Alarm $= \frac{\Sigma \text{ FP}}{\Sigma \text{ condition negative}}$
	Accuracy $= \frac{\Sigma \text{ TP} + \Sigma \text{ TN}}{\Sigma \text{ total population}}$	Positive Predictive Value (PPV), Precision $= \frac{\Sigma \text{ TP}}{\Sigma \text{ prediction positive}}$	False Omission Rate (FOR) $= \frac{\Sigma \text{ FN}}{\Sigma \text{ prediction negative}}$	Positive Likelihood Ratio (LR+) $= \frac{\text{TPR}}{\text{FPR}}$
		False Discovery Rate (FDR) $= \frac{\Sigma \text{ FP}}{\Sigma \text{ prediction positive}}$	Negative Predictive Value (NPV) $= \frac{\Sigma \text{ TN}}{\Sigma \text{ prediction negative}}$	Negative Likelihood Ratio (LR−) $= \frac{\text{FNR}}{\text{TNR}}$

2). *The Receiver Operator Curve (ROC)

- The Receiver Operator Curve (ROC) curve was developed during World War II to help analyze radar data.
- It illustrates the diagnostic ability of a binary classifier system.
- The ROC curve is created by plotting the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings.

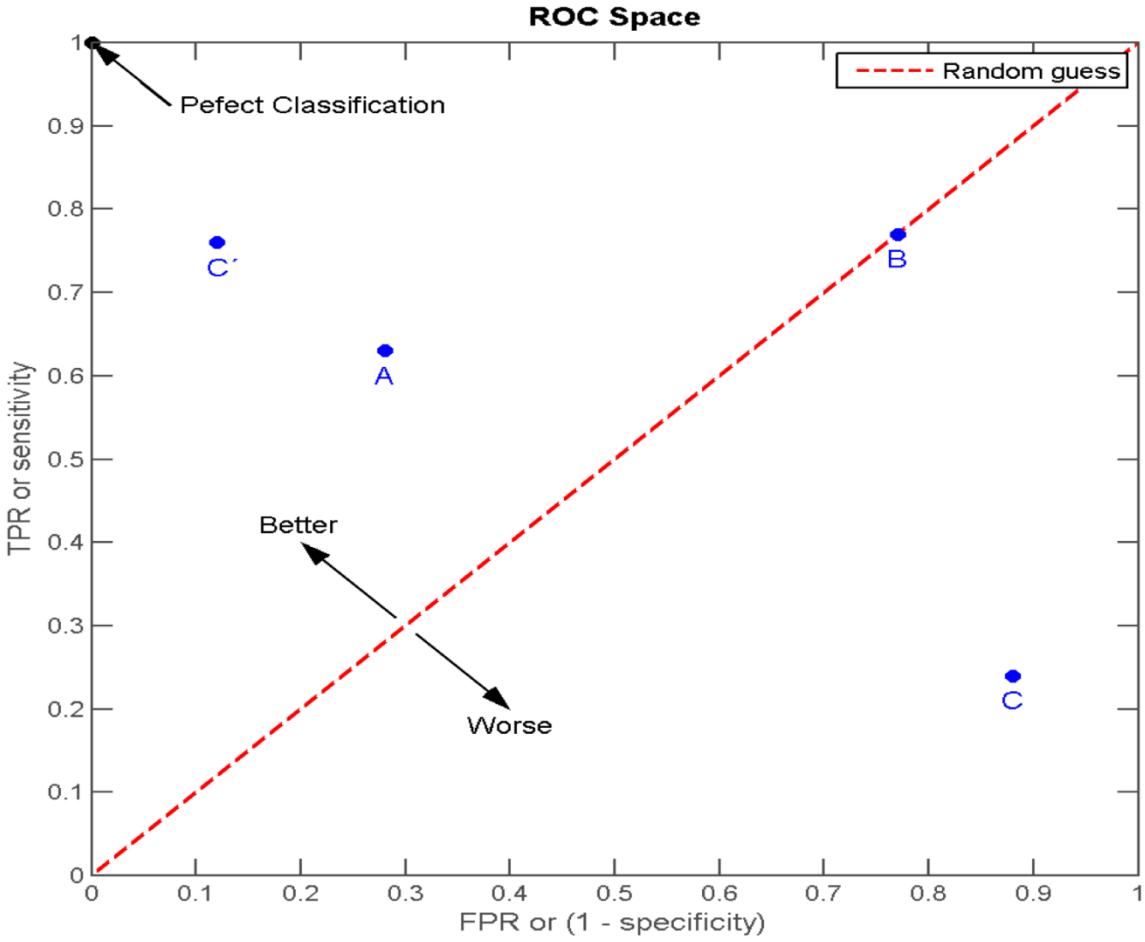


2). *The Receiver Operator Curve (ROC)

- Using a simple random guess model to predict a class

A full discussion of the ROC curve is beyond the scope of this course.

For now, you just need to know that the area under the curve is a metric for how well a model fit the data.



5. Samples for Logistic Regression with Scitik_learn and Spark

- Two examples:
  - Binary_Classifier.ipynb (Demo)
  - Titanic_Logistic_Regression.ipynb

- **We will have a practice with them in the lab class.**

内容提要

Chapter 7: ML - Logistic Regression

Chapter 8: ML K-Means Clusters

ML K-Means Clusters

- **1. About Clustering**
  - Frequently, you will find many works on data analytics dealing with creating groups from data, instead of trying to predict classes or values.

- A typical clustering problem look like?
  - -Cluster Similar Documents
  - -Cluster Customers based on Features
  - -Market Segmentation
  - Identify similar physical groups

About Clustering

- **The key distinction: the previous supervised learning deals with historical labeled data. In clustering, you have unlabeled data that only contains features and you want to see if there are patterns in the data that would allow you to create groups (or clusters).**
- **As data are unlabeled, you attempt to "discover" possible labels, through clustering. Thus, you can think of "clustering" as an attempt to create labels.**
- **In "clustering", you input some unlabeled data and the unsupervised learning algorithm, which generates possible clusters of the data.**

Correctness in Clustering

- It is not easy to evaluate the groups or clusters for "correctness".
- For example, you could cluster tumors into two groups, hoping to separate between benign and malignant. But there is no guarantee that the clusters will fall along those lines, it will just split into the two most separable groups.
- Also depending on the clustering algorithm, it may be up to you to decide beforehand how many clusters you expect to create!

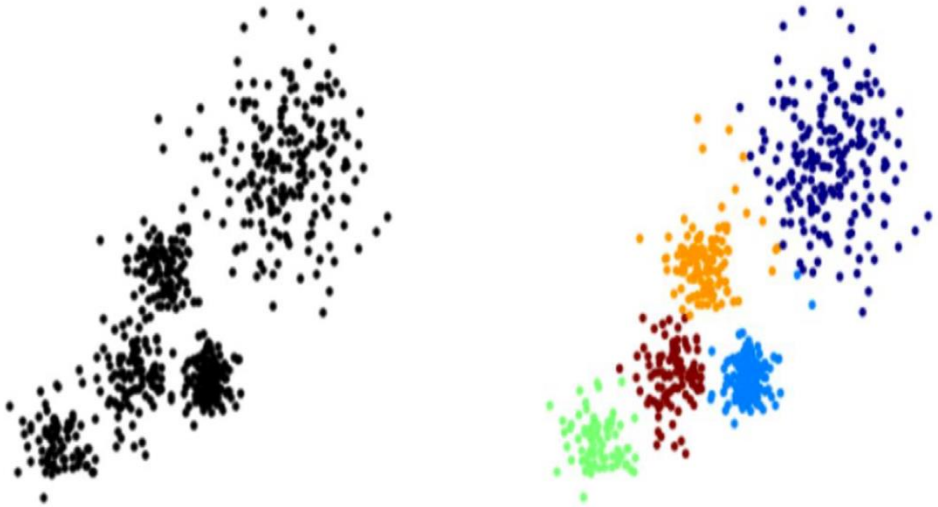
Correctness in Clustering

- **It will be up to you to decide what the groups actually represent.**
- **Sometimes this is easy, sometimes it's really hard!**
- **A lot of clustering problems have no 100% correct approach or answer, that is the nature of unsupervised learning!**

2. K-Means Clustering

- **K-Means Clustering is an unsupervised learning algorithm that will attempt to group similar clusters together in your data.**

- **The overall goal is to divide data into distinct groups such that observations within each group are similar.**



1). The K-Means Example:

- **To illustrate a k-means algorithm, consider the following data set consisting of the scores of two variables A and B on each of seven individuals:**

Subject	A	B
1	1.0	1.0
2	1.5	2.0
3	3.0	4.0
4	5.0	7.0
5	3.5	5.0
6	4.5	5.0
7	3.5	4.5

- **Each subject (or instance) represents a point, say, subject 3, corresponding to (A3, B3) = (3.0, 4.0) and so, any two points has a distance.**

The K-Means Example:

- **This dataset will be grouped into two clusters.**
- **Initially, let the A and B values of the two individuals furthest apart (using the Euclidean distance measure), define the initial cluster means, called "Mean Vector (centroid)":**

	Individual	Mean Vector (centroid)
Group 1	1	(1.0, 1.0)
Group 2	4	(5.0, 7.0)

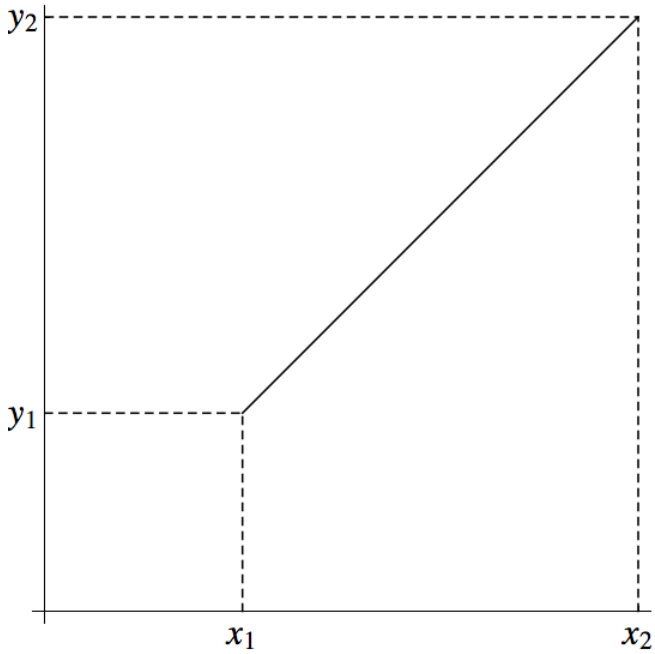
"The longest distance" between two points

The K-Means Example:

- The distance between any two points on the real line (one dimension) is the absolute value of the numerical difference of their coordinates.

$$\sqrt{(x-y)^2} = |x-y|.$$

Euclidean distance measure in two dimensions:



More general expression:

The **Euclidean distance** between points **p** and **q** is the length of the line segment connecting them ($\overline{\mathbf{pq}}$).

In Cartesian coordinates, if $\mathbf{p} = (p_1, p_2, \ldots, p_n)$ and $\mathbf{q} = (q_1, q_2, \ldots, q_n)$ are two points in Euclidean *n*-space, then the distance (d) from **p** to **q**, or from **q** to **p** is given by the Pythagorean formula:

$$\begin{aligned} \mathrm{d}(\mathbf{p}, \mathbf{q}) = \mathrm{d}(\mathbf{q}, \mathbf{p}) &= \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2} \\ &= \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}. \end{aligned} \qquad (1)$$

The position of a point in a Euclidean *n*-space is a Euclidean vector. So, **p** and **q** are Euclidean vectors, starting from the origin of the space, and their tips indicate two points. The **Euclidean norm**, or **Euclidean length**, or **magnitude** of a vector measures the length of the vector:

$$\|\mathbf{p}\| = \sqrt{p_1^2 + p_2^2 + \cdots + p_n^2} = \sqrt{\mathbf{p} \cdot \mathbf{p}},$$

where the last equation involves the dot product.

More general expression:

- **The remaining individuals are now examined in sequence and allocated to the cluster to which they are closest, in terms of Euclidean distance to the cluster mean.**
- **The mean vector is recalculated each time a new member is added. This leads to the following series of steps:**

At step 2, subject 2 joins group 1, as it is closest to subject 1 (not subject 4).

	Cluster 1		Cluster 2	
Step	Individual	Mean Vector (centroid)	Individual	Mean Vector (centroid)
1	1	(1.0, 1.0)	4	(5.0, 7.0)
2	1, 2	(1.2, 1.5)	4	(5.0, 7.0)
3	1, 2, 3	(1.8, 2.3)	4	(5.0, 7.0)
4	1, 2, 3	(1.8, 2.3)	4, 5	(4.2, 6.0)
5	1, 2, 3	(1.8, 2.3)	4, 5, 6	(4.3, 5.7)
6	1, 2, 3	(1.8, 2.3)	4, 5, 6, 7	(4.1, 5.4)

More general expression:

- **Now the initial partition has changed, and the two clusters at this stage having the following characteristics:**

	Individual	Mean Vector (centroid)
Cluster 1	1, 2, 3	(1.8, 2.3)
Cluster 2	4, 5, 6, 7	(4.1, 5.4)

More general expression:

- But we cannot yet be sure that each individual has been assigned to the right cluster.

- So, we compare each individual's distance to its own cluster mean and to that of the opposite cluster. And we find:

Individual	Distance to mean (centroid) of Cluster 1	Distance to mean (centroid) of Cluster 2
1	1.5	5.4
2	0.4	4.3
3	2.1	1.8
4	5.7	1.8
5	3.2	0.7
6	3.8	0.6
7	2.8	1.1

More general expression:

- Only individual 3 is nearer to the mean of the opposite cluster (Cluster 2) than its own (Cluster 1).
- In other words, each individual's distance to its own cluster mean should be smaller that the distance to the other cluster's mean (which is not the case with individual 3).
- Thus, individual 3 is relocated to Cluster 2 resulting in the new partition:

	Individual	Mean Vector (centroid)
Cluster 1	1, 2	(1.3, 1.5)
Cluster 2	3, 4, 5, 6, 7	(3.9, 5.1)

More general expression:

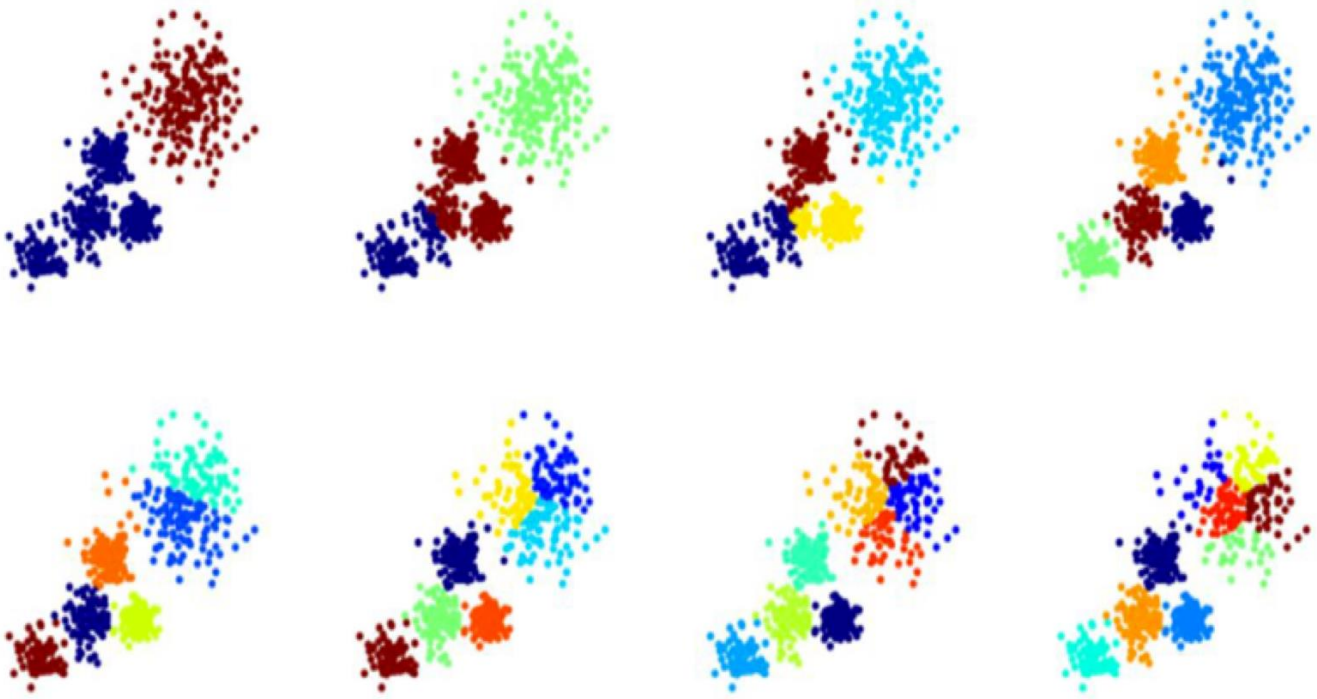
- The iterative relocation would now continue from this new partition until no more relocations occur.
- However, in this example each individual is now nearer its own cluster mean than that of the other cluster and the iteration stops, choosing the latest partitioning as the final cluster solution.
- Also, it is possible that the k-means algorithm won't find a final solution.
- In this case it would be a good idea to consider stopping the algorithm after a pre-chosen maximum of iterations.

2). The K-Means Clustering Algorithm

- Choose a number of Clusters "K"
- Randomly assign each point to a cluster
- Until clusters stop changing, repeat the following:
  - For each cluster compute the cluster centroid by taking the mean vector of points in the cluster
  - Assign each data point to the cluster for which the centroid is the closest

3. Choosing a K value

- There is no easy answer for choosing a “best” K value. One way is the elbow method:

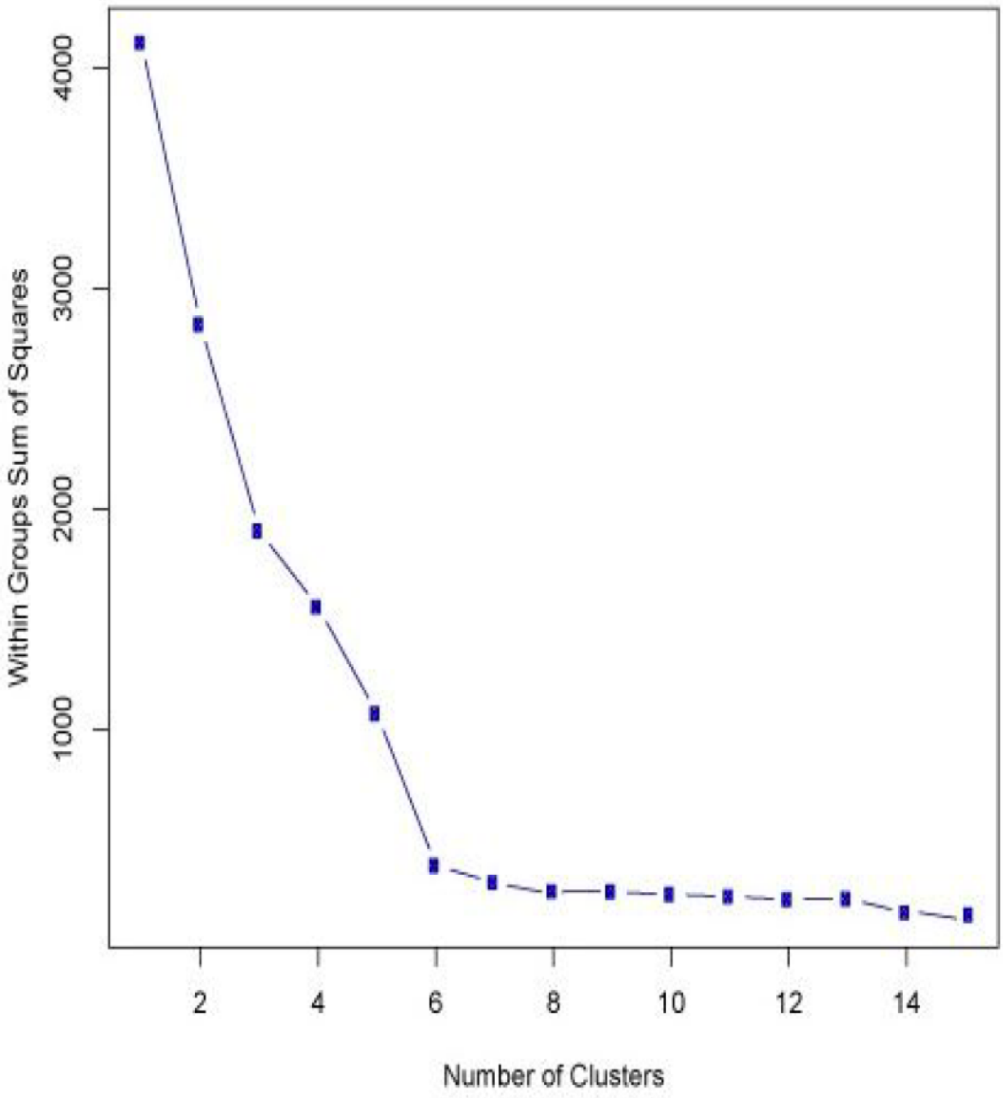


3. Choosing a K value

- At first, compute the sum of squared error (SSE) for some values of k (for example 2, 4, 6, 8, etc.).
- The SSE = “the sum of the squared distance between each member of the cluster and its centroid”.
- If you plot k against the SSE, you will see that the error decreases as k gets larger; this is because when the number of clusters increases, they should be smaller, so distortion is also smaller.
- The idea of the elbow method is to choose the k at which the SSE decreases abruptly. This produces an "elbow effect" in the graph, as you can see in the following picture:

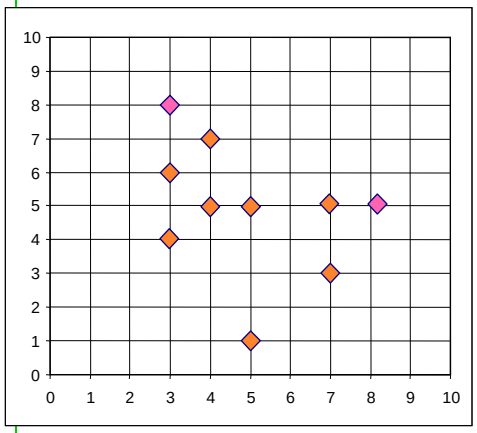
3. Choosing a K value

- PySpark by itself doesn't support a plotting mechanism, but you could use collect() and then plot th results with matplotlib or other visualization libraries.
- But don't take this as a strict rule when choosing a K value!
- A lot of depends more on th context of the exact situation (domain knowledge)



基于划分的方法-K-means算法实例

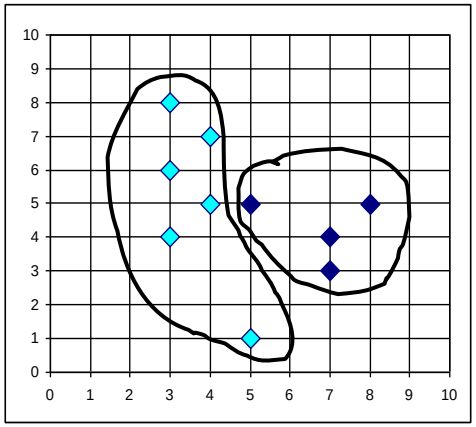
K-means 算法运行过程



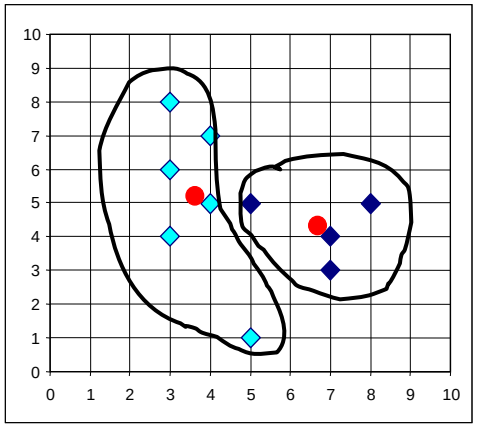
K=2

Arbitrarily choose K object as initial cluster center

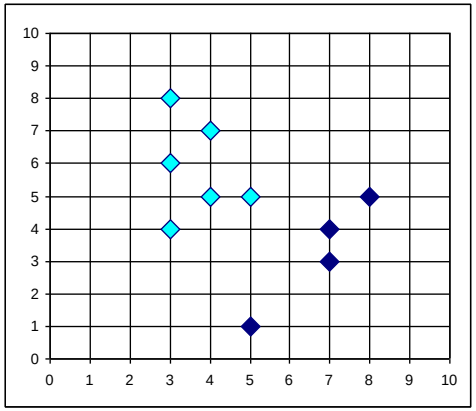
Assign each objects to most similar center



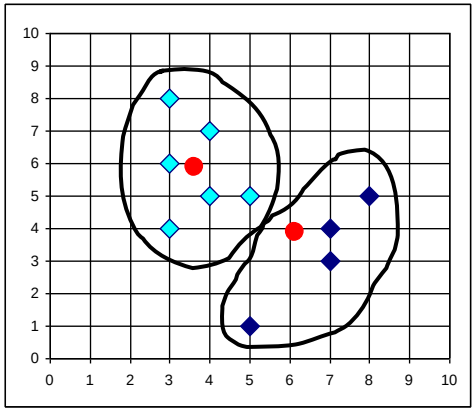
Update the cluster means



reassign



Update the cluster means



reassign

4. Samples for Clustering with Spark

- **Two examples are given in the directory …/Labs/Lab3**
  - Classification_with_Clustering.ipynb and
  - Clustering_Exercises.ipynb with its SOLUTION.

- **We will have a practice with them in the lab class.**

哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

结束

2020年9月22日