Machine Learning Supervised Learning & Decision Tree

Machine learning:

- **Machine learning** is a subfield of computer science that explores the study and construction of algorithms that can learn from and make predictions on data.

- Such algorithms operate by building a model from example inputs in order to make data-driven predictions or decisions, rather than following strictly static program instructions.

Types of machine learning learning

- **Supervised learning** : is the machine learning task of inferring a function from labeled training data. The training data consist of a set of *training examples*. In supervised learning, each example is a *pair* consisting of an input object and a desired output value. A supervised learning algorithm analyzes the training data and produces an inferred function, which can be used for mapping new examples.

- **Unsupervised learning** :

Learning useful structure *without* labeled classes, optimization criterion, feedback signal, or any other information beyond the raw data

- REINFORCEMENT LEARNING:

Learn how to behave successfully to achieve a goal while interacting with an external environment .(Learn via Experiences!)

An example application

- An emergency room in a hospital measures 17 variables (e.g., blood pressure, age, etc) of newly admitted patients.
- A decision is needed: whether to put a new patient in an intensive-care unit.
- Due to the high cost of ICU, those patients who may survive less than a month are given higher priority.
- Problem: to predict high-risk patients and discriminate them from low-risk patients.

Another application

- A credit card company receives thousands of applications for new cards. Each application contains information about an applicant,
  - age
  - Marital status
  - annual salary
  - outstanding debts
  - credit rating
  - etc.
- Problem: to decide whether an application should approved, or to classify applications into two categories, approved and not approved.

Machine learning and our focus

- Like human learning from past experiences.
- A computer does not have “experiences”.
- A computer system learns from data, which represent some “past experiences” of an application domain.
- Our focus: learn a target function that can be used to predict the values of a discrete class attribute, e.g., approve or not-approved, and high-risk or low risk.
- The task is commonly called: Supervised learning, classification, or inductive learning.

The data and the goal

- Data: A set of data records (also called examples, instances or cases) described by
  - *k* attributes: *A*1, *A*2, … *Ak*.
  - a class: Each example is labelled with a pre-defined class.
- Goal: To learn a classification model from the data that can be used to predict the classes of new (future, or test) cases/instances.

An example: data (loan application)

Approved or not

ID	Age	Has_Job	Own_House	Credit_Rating	Class
1	young	false	false	fair	**No**
2	young	false	false	good	**No**
3	young	true	false	good	**Yes**
4	young	true	true	fair	**Yes**
5	young	false	false	fair	**No**
6	middle	false	false	fair	**No**
7	middle	false	false	good	**No**
8	middle	true	true	good	**Yes**
9	middle	false	true	excellent	**Yes**
10	middle	false	true	excellent	**Yes**
11	old	false	true	excellent	**Yes**
12	old	false	true	good	**Yes**
13	old	true	false	good	**Yes**
14	old	true	false	excellent	**Yes**
15	old	false	false	fair	**No**

An example: the learning task

- Learn a classification model from the data
- Use the model to classify future loan applications into
  - Yes (approved) and
  - No (not approved)
- What is the class for following case/instance?

Age	Has_Job	Own_house	Credit-Rating	Class
young	false	false	good	?

Supervised vs. unsupervised Learning

- Supervised learning: classification is seen as supervised learning from examples.
  - Supervision: The data (observations, measurements, etc.) are labeled with pre-defined classes. It is like that a "teacher" gives the classes (supervision).
  - Test data are classified into these classes too.

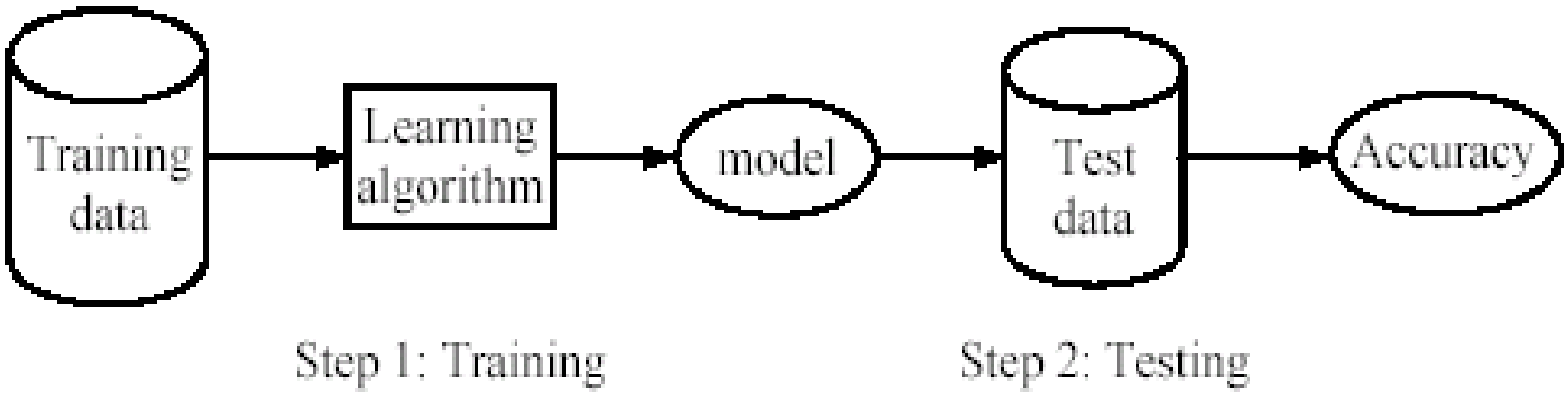
- Unsupervised learning (clustering)
  - Class labels of the data are unknown
  - Given a set of data, the task is to establish the existence of classes or clusters in the data

Supervised learning process: two steps

Learning (training): Learn a model using the training data

Testing: Test the model using unseen test data to assess the model accuracy

$$\textit{Accuracy} = \frac{\text{Number of correct classifications}}{\text{Total number of test cases}},$$



What do we mean by learning?

- Given
  - a data set *D*,
  - a task *T*, and
  - a performance measure *M*,

a computer system is said to **learn** from *D* to perform the task *T* if after learning the system's performance on *T* improves as measured by *M*.

- In other words, the learned model helps the system to perform *T* better as compared to no learning.

An example

- Data: Loan application data
- Task: Predict whether a loan should be approved or not.
- Performance measure: accuracy.

No learning: classify all future applications (test data) to the majority class (i.e., Yes):

Accuracy = 9/15 = 60%.

- We can do better than 60% with learning.

Fundamental assumption of learning

Assumption: The distribution of training examples is identical to the distribution of test examples (including future unseen examples).

- In practice, this assumption is often violated to certain degree.
- Strong violations will clearly result in poor classification accuracy.
- To achieve good accuracy on the test data, training examples must be sufficiently representative of the test data.

Decision trees

- Decision tree learning is one of the most widely used techniques for classification.
  - Its classification accuracy is competitive with other methods, and
  - it is very efficient.

- The classification model is a tree, called decision tree.

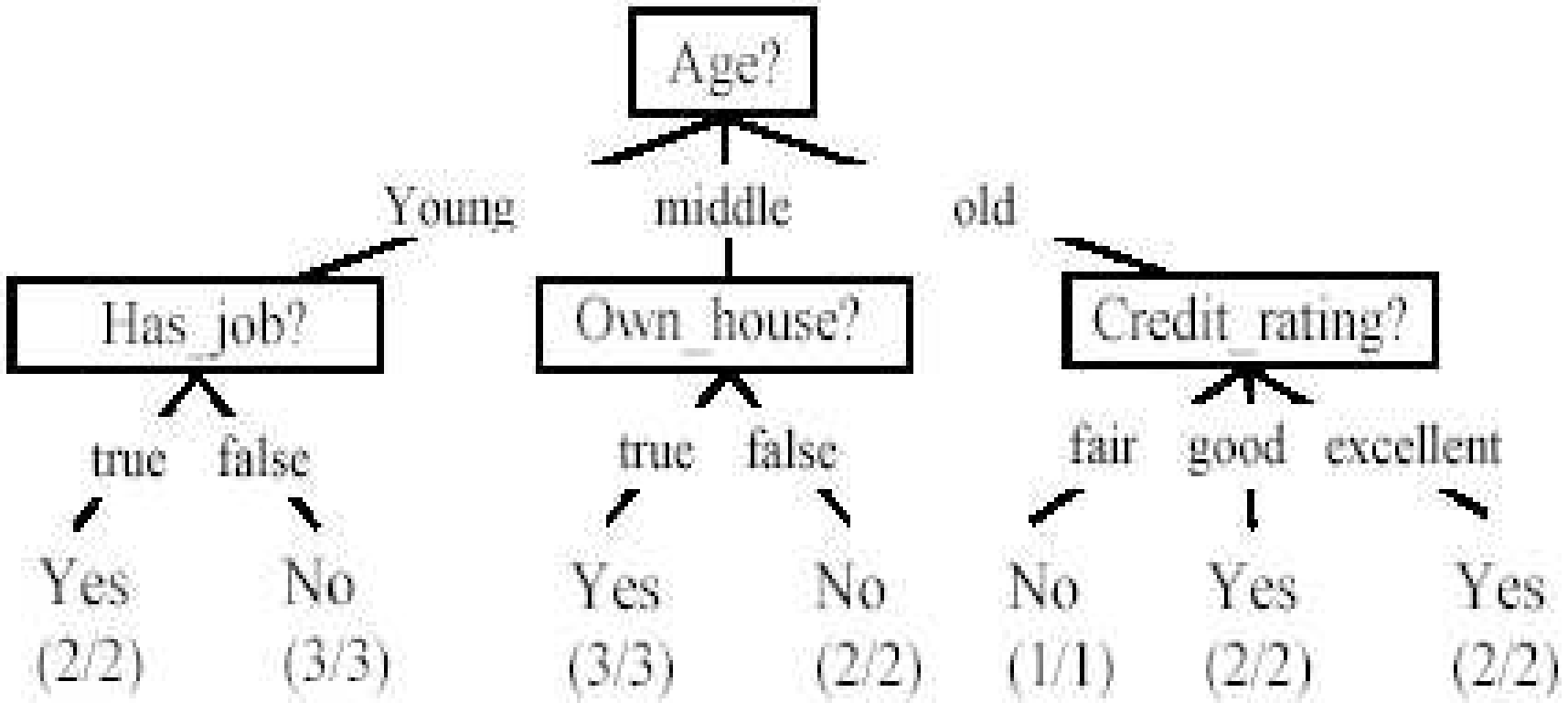
The loan data (reproduced)

Approved or not

ID	Age	Has_Job	Own_House	Credit_Rating	Class
1	young	false	false	fair	**No**
2	young	false	false	good	**No**
3	young	true	false	good	**Yes**
4	young	true	true	fair	**Yes**
5	young	false	false	fair	**No**
6	middle	false	false	fair	**No**
7	middle	false	false	good	**No**
8	middle	true	true	good	**Yes**
9	middle	false	true	excellent	**Yes**
10	middle	false	true	excellent	**Yes**
11	old	false	true	excellent	**Yes**
12	old	false	true	good	**Yes**
13	old	true	false	good	**Yes**
14	old	true	false	excellent	**Yes**
15	old	false	false	fair	**No**

A decision tree from the loan data

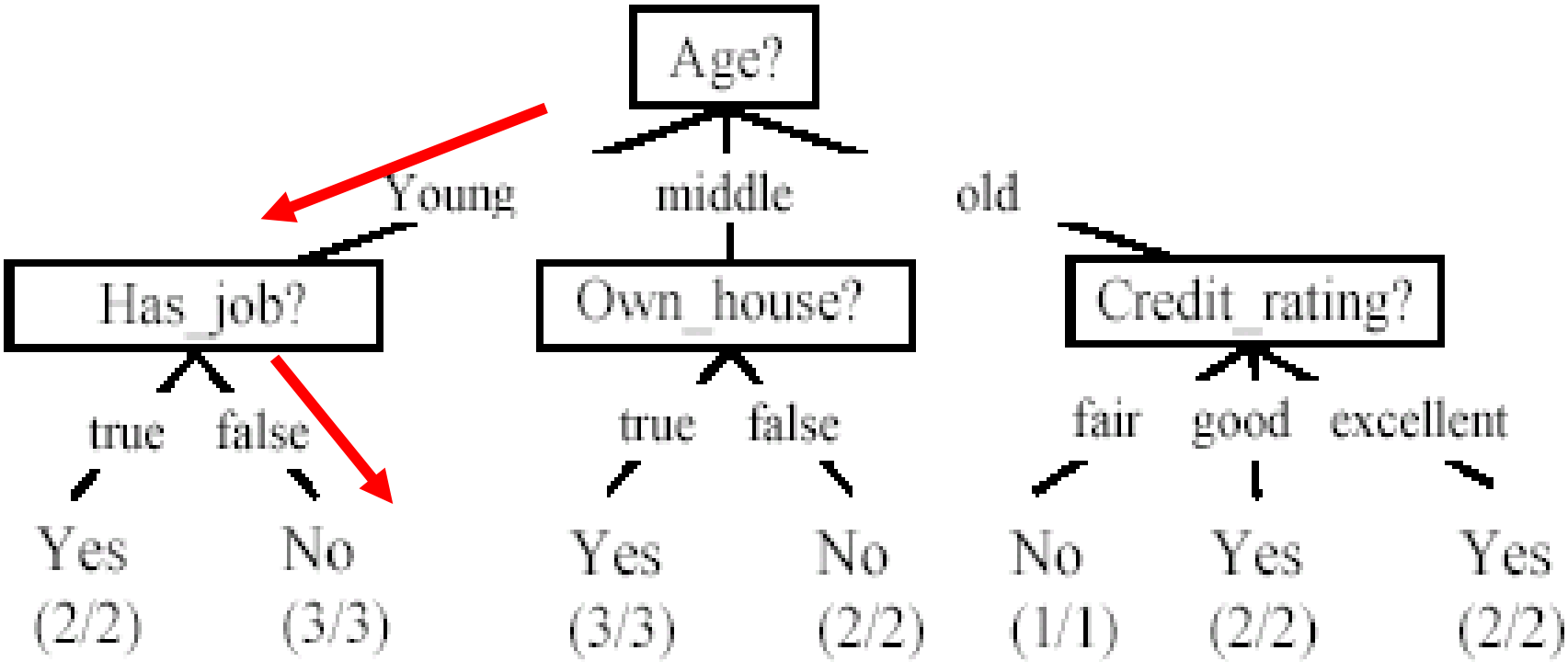
Decision nodes and leaf nodes (classes)



Use the decision tree

Age	Has_Job	Own_house	Credit-Rating	Class
young	false	false	good	?

No



Evaluating classification methods

- **Predictive accuracy**

$$Accuracy = \frac{\text{Number of correct classifications}}{\text{Total number of test cases}}$$

- Efficiency
  - time to construct the model
  - time to use the model
- Robustness: handling noise and missing values
- Scalability: efficiency in disk-resident databases
- Interpretability:
  - understandable and insight provided by the model
- Compactness of the model: size of the tree, or the number of

Evaluation methods

- **Holdout set**: The available data set *D* is divided into two disjoint subsets,
  - the *training set Dtrain* (for learning a model)
  - the *test set Dtest* (for testing the model)
- **Important:** training set should not be used in testing and the test set should not be used in learning.
  - Unseen test set provides a unbiased estimate of accuracy.
- The test set is also called the holdout set. (the examples in the original data set *D* are all labeled with classes.)
- This method is mainly used when the data set *D* is large.

Evaluation methods (cont…)

- **n-fold cross-validation**: The available data is partitioned into *n* equal-size disjoint subsets.
- Use each subset as the test set and combine the rest *n*-1 subsets as the training set to learn a classifier.
- The procedure is run *n* times, which give *n* accuracies.
- The final estimated accuracy of learning is the average of the *n* accuracies.
- 10-fold and 5-fold cross-validations are commonly used.
- This method is used when the available data is not large.

Evaluation methods (cont…)

- **Leave-one-out cross-validation**: This method is used when the data set is very small.
- It is a special case of cross-validation
- Each fold of the cross validation has only a single test example and all the rest of the data is used in training.
- If the original data has *m* examples, this is *m*-fold cross-validation

Evaluation methods (cont…)

- **Validation set**: the available data is divided into three subsets,
  - a training set,
  - a validation set and
  - a test set.
- A validation set is used frequently for estimating parameters in learning algorithms.
- In such cases, the values that give the best accuracy on the validation set are used as the final parameter values.
- Cross-validation can be used for parameter estimating as well.

Classification measures

- Accuracy is only one measure (error = 1-accuracy).
- **Accuracy is not suitable in some applications**.
- In classification involving skewed or highly imbalanced data, e.g., network intrusion and financial fraud detections, we are interested only in the minority class.
  - High accuracy does not mean any intrusion is detected.
  - E.g., 1% intrusion. Achieve 99% accuracy by doing nothing.
- The class of interest is commonly called the **positive class**, and the rest **negative classes.**

Back propagation

- Backpropagation is a systematic method for training multiple-layer ANNs.
- The standard backpropagation algorithm is as follows :

1. Build a network with a chosen number of input, hidden and output units.
2. Initialize all the weights to low random values
3. Choose a single training pair at random
4. Copy the input pattern to the input layer
5. Cycle the network so that the activations from the inputs generate the activations in the hidden and output layers
6. Calculate the error derivative between the output activation and the target output.
7. Backpropagate the summed products of the weights and errors in the output layer in order to calculate the error in the hidden units.
8. Update the weights attached to each unit according to the error in that unit, the output from the unit below it, and the learning parameters, until the error is sufficiently low or the network settles.

Conclusion

- Applications of supervised learning are in almost any field or domain.
- There are numerous classification techniques.
- There are still many other methods, e.g.,
  - Bayesian networks
  - Neural networks
  - Genetic algorithms
  - Fuzzy classification

This large number of methods also show the importance of classification and its wide applicability.

- It remains to be an active research area.

Unsupervised Learning

Supervised learning vs. unsupervised learning

- Supervised learning: discover patterns in the data that relate data attributes with a target (class) attribute.
  - These patterns are then utilized to predict the values of the target attribute in future data instances.
- Unsupervised learning: The data have no target attribute.
  - We want to explore the data to find some intrinsic structures in them.

What is clustering?

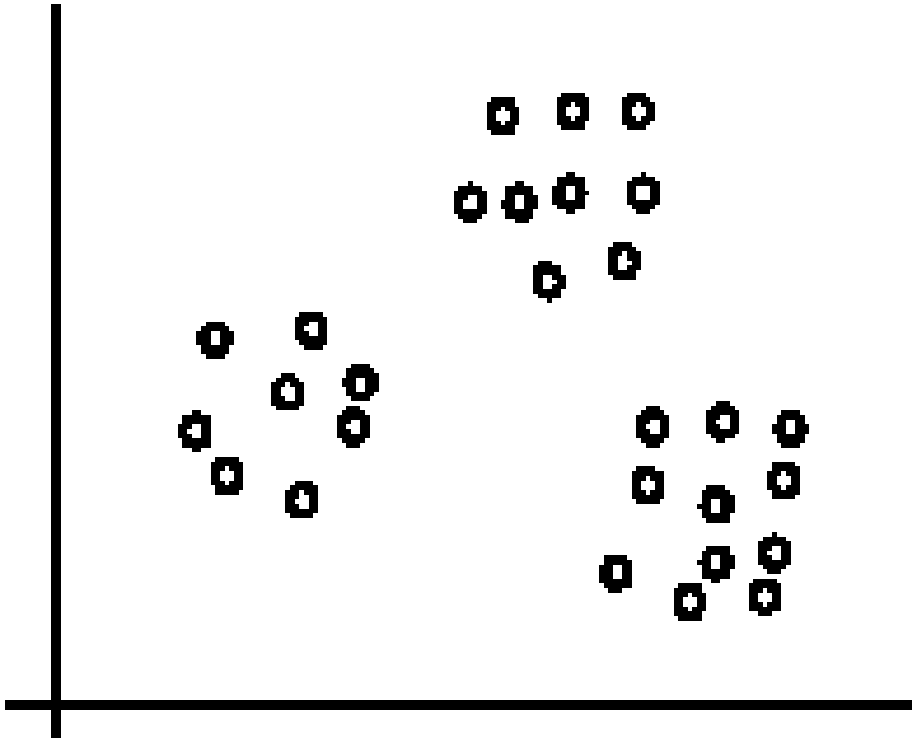
- **Clustering** is the classification of objects into different groups, or more precisely, the partitioning of a data set into subsets (clusters), so that the data in each subset (ideally) share some common trait - often according to some defined distance measure.

Clustering

- Clustering is a technique for finding similarity groups in data, called **clusters**. I.e.,
  - it groups data instances that are similar to (near) each other in one cluster and data instances that are very different (far away) from each other into different clusters.
- Clustering is often called an **unsupervised learning** task as no class values denoting an *a priori* grouping of the data instances are given, which is the case in supervised learning.
- Due to historical reasons, clustering is often considered synonymous with unsupervised learning.

An illustration

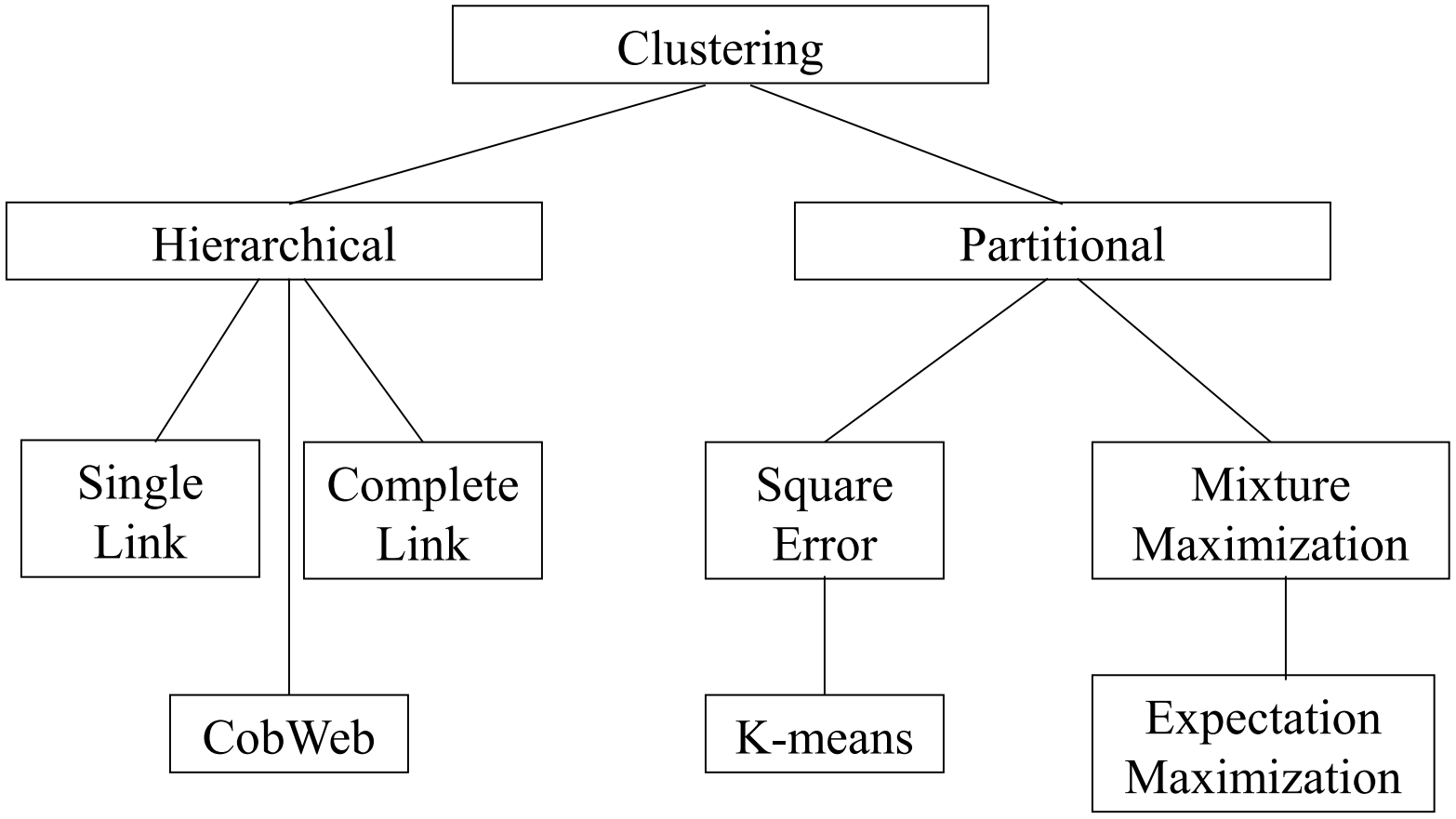
- The data set has three natural groups of data points, i.e., 3 natural clusters.



What is clustering for?

- Let us see some real-life examples
- Example 1: groups people of similar sizes together to make “small”, “medium” and “large” T-Shirts.
  - Tailor-made for each person: too expensive
  - One-size-fits-all: does not fit all.
- Example 2: In marketing, segment customers according to their similarities
  - To do targeted marketing.

Types of Clustering Techniques



Types of Clustering Techniques:

1. **Hierarchical algorithms**: these find successive clusters using previously established clusters.

   1. Agglomerative ("bottom-up"): Agglomerative algorithms begin with each element as a separate cluster and merge them into successively larger clusters.

   2. Divisive ("top-down"): Divisive algorithms begin with the whole set and proceed to divide it into successively smaller clusters.

2. **Partitional clustering:** Partitional algorithms determine all clusters at once. They include:

   - ***K*-means and derivatives**

K-means clustering

- K-means is a partitional clustering algorithm
- Let the set of data points (or instances) *D* be

  {**x**1, **x**2, …, **x**n},

  where **x***i* = (*xi*1, *xi*2, …, *xir*) is a vector in a real-valued space *X* *Rr*, and *r* is the number of attributes (dimensions) in the data.
- The *k*-means algorithm partitions the given data into *k* clusters.
  - Each cluster has a cluster **center**, called **centroid**.
  - *k* is specified by the user

K-means algorithm

- Given *k*, the *k-means* algorithm works as follows:
  1) Randomly choose *k* data points (seeds) to be the initial centroids, cluster centers
  2) Assign each data point to the closest centroid
  3) Re-compute the centroids using the current cluster memberships.
  4) If a convergence criterion is not met, go to 2).

K-means algorithm – (cont …)

Algorithm *k*-means(*k*, *D*)

1 Choose *k* data points as the initial centroids (cluster centers)
2 **repeat**
3 **for** each data point $\mathbf{x} \in D$ do
4 compute the distance from **x** to each centroid;
5 assign **x** to the closest centroid // a centroid represents a cluster
6 **endfor**
7 re-compute the centroids using the current cluster memberships
8 **until** the stopping criterion is met

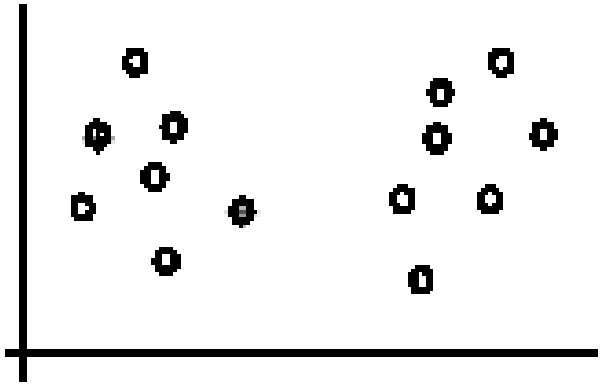
Stopping/convergence criterion

1. no (or minimum) re-assignments of data points to different clusters,
2. no (or minimum) change of centroids, or
3. minimum decrease in the **sum of squared error** (SSE),

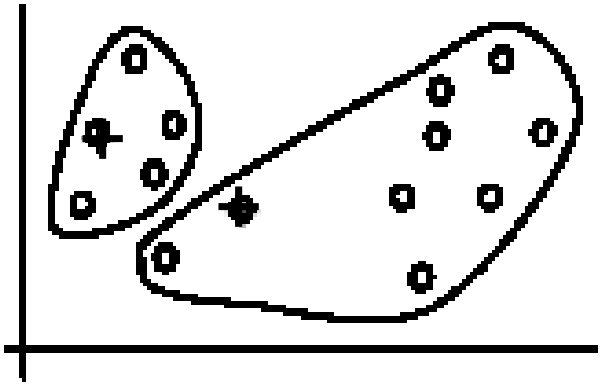
$$SSE = \sum_{j=1}^{k} \sum_{\mathbf{x} \in C_j} dist(\mathbf{x}, \mathbf{m}_j)^2 \quad (1)$$

- *Ci* is the *j*th cluster, **m***j* is the centroid of cluster *Cj* (the mean vector of all the data points in *Cj*), and *dist*(**x**, **m***j*) is the distance between data point **x** and centroid **m***j*.

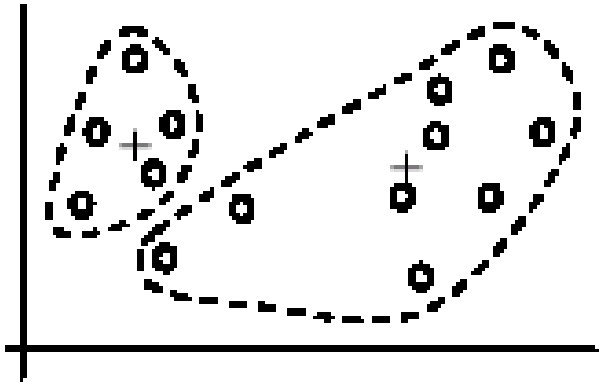
An example



(A). Random selection of k centers

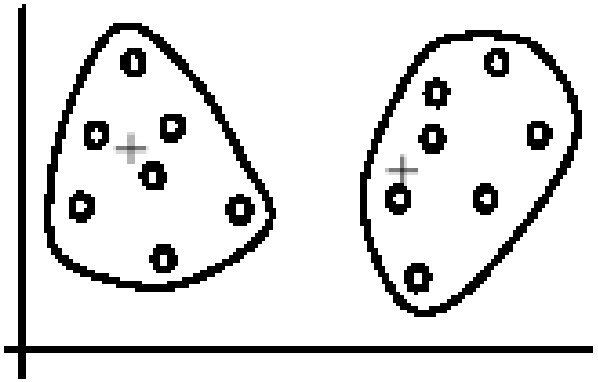


Iteration 1: (B). Cluster assignment

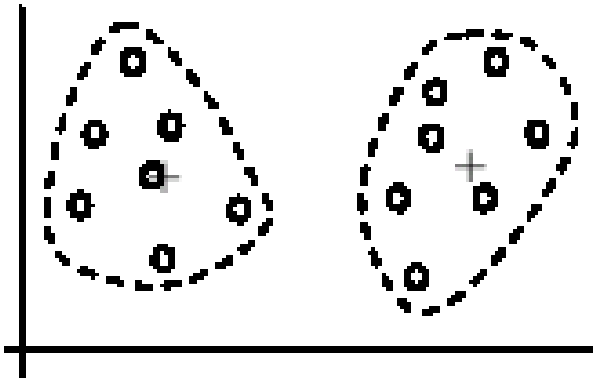


(C). Re-compute centroids

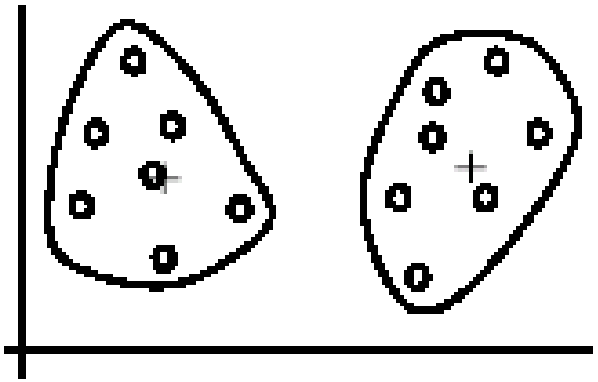
An example (cont …)



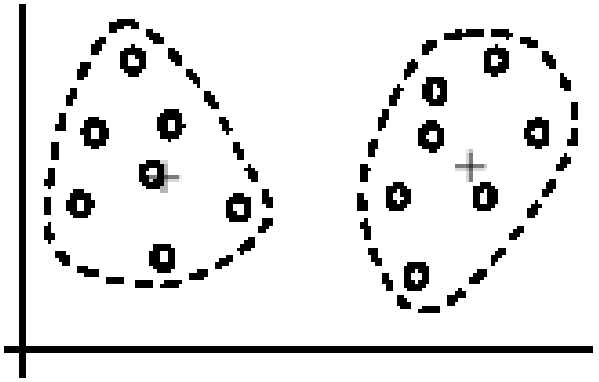
Iteration 2: (D). Cluster assignment



(E). Re-compute centroids



Iteration 3: (F). Cluster assignment



(G). Re-compute centroids

An example distance function

The *k*-means algorithm can be used for any application data set where the **mean** can be defined and computed. In the **Euclidean space**, the mean of a cluster is computed with:

$$\mathbf{m}_j = \frac{1}{|C_j|} \sum_{\mathbf{x}_i \in C_j} \mathbf{x}_i \qquad (2)$$

where $|C_j|$ is the number of data points in cluster C_j. The distance from one data point $\mathbf{x}_i$ to a mean (centroid) $\mathbf{m}_j$ is computed with

$$dist(\mathbf{x}_i, \mathbf{m}_j) = \| \mathbf{x}_i - \mathbf{m}_j \| \qquad (3)$$

$$= \sqrt{(x_{i1} - m_{j1})^2 + (x_{i2} - m_{j2})^2 + \ldots + (x_{ir} - m_{jr})^2}$$

Strengths of k-means

- Strengths:
  - Simple: easy to understand and to implement
  - Efficient: Time complexity: $O(tkn)$,
    where n is the number of data points,

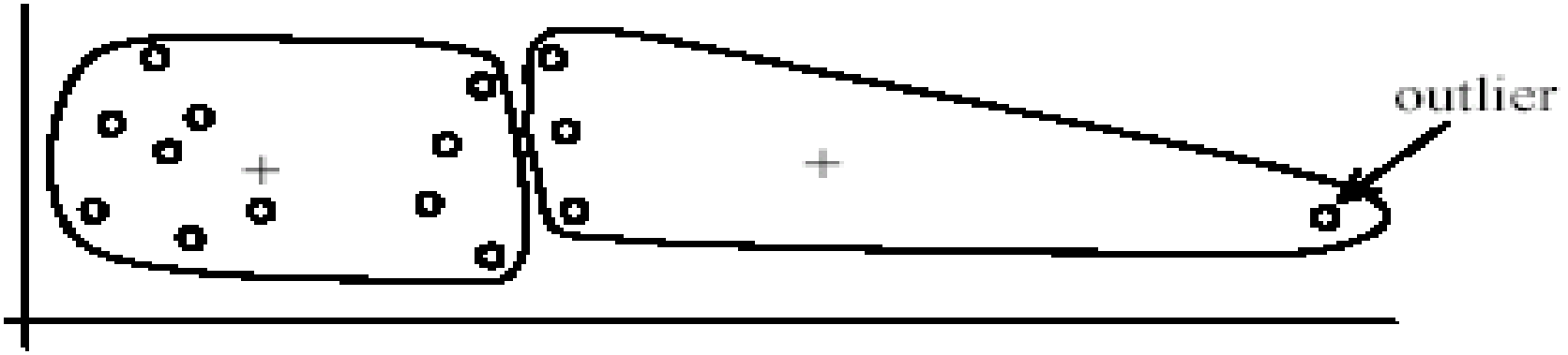
    k is the number of clusters, and

    t is the number of iterations.
  - Since both k and t are small. k-means is considered a linear algorithm.
- K-means is the most popular clustering algorithm.
- Note that: it terminates at a local optimum if SSE is used. The global optimum is hard to find due to complexity.

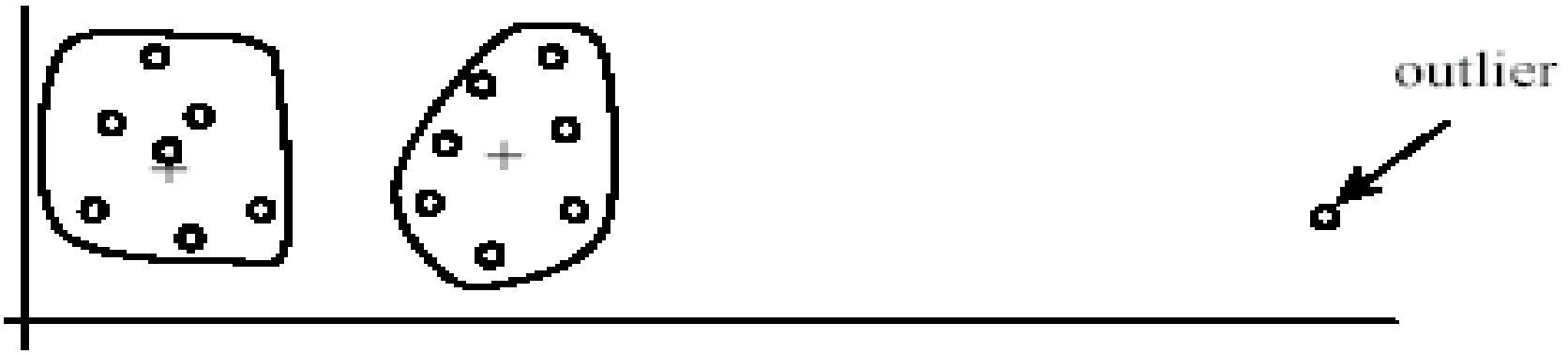
Weaknesses of k-means

- The algorithm is only applicable if the mean is defined.
  - For categorical data, *k*-mode - the centroid is represented by most frequent values.
- The user needs to specify *k*.
- The algorithm is sensitive to **outliers**
  - Outliers are data points that are very far away from other data points.
  - Outliers could be errors in the data recording or some special data points with very different values.

Weaknesses of k-means: Problems with outliers



(A): Undesirable clusters



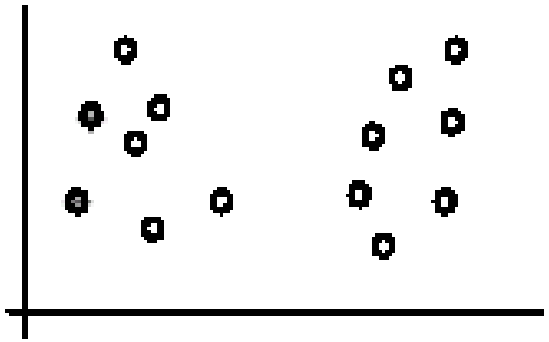
(B): Ideal clusters

Weaknesses of k-means: To deal with outliers

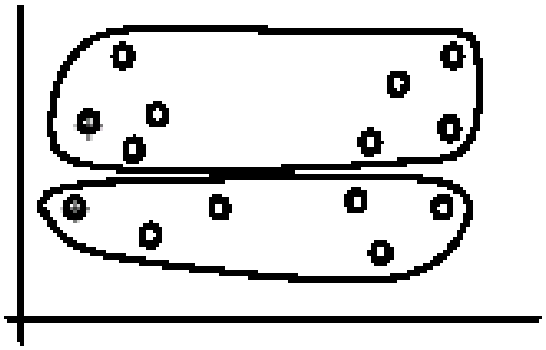
- One method is to remove some data points in the clustering process that are much further away from the centroids than other data points.
  - To be safe, we may want to monitor these possible outliers over a few iterations and then decide to remove them.
- Another method is to perform random sampling. Since in sampling we only choose a small subset of the data points, the chance of selecting an outlier is very small.
  - Assign the rest of the data points to the clusters by distance or similarity comparison, or classification

Weaknesses of k-means (cont …)

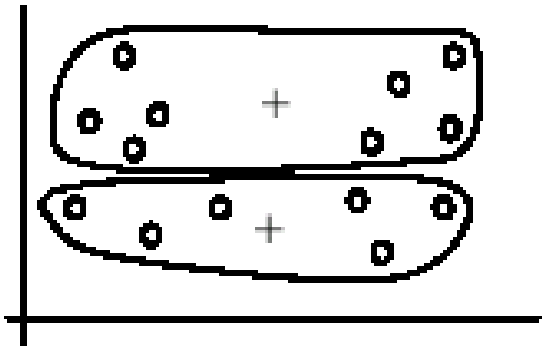
- The algorithm is sensitive to initial seeds.



(A). Random selection of seeds (centroids)



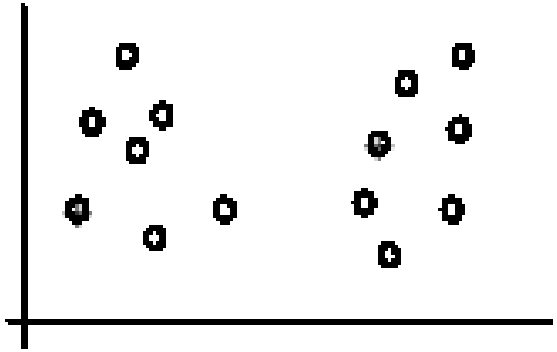
(B). Iteration 1



(C). Iteration 2

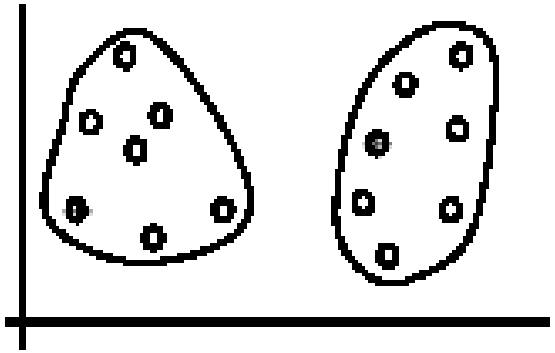
Weaknesses of k-means (cont …)

- If we use different seeds: good results

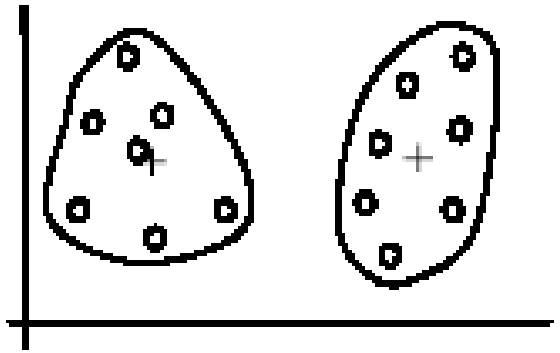


(A). Random selection of k seeds (centroids)

- There are some methods to help choose good seeds



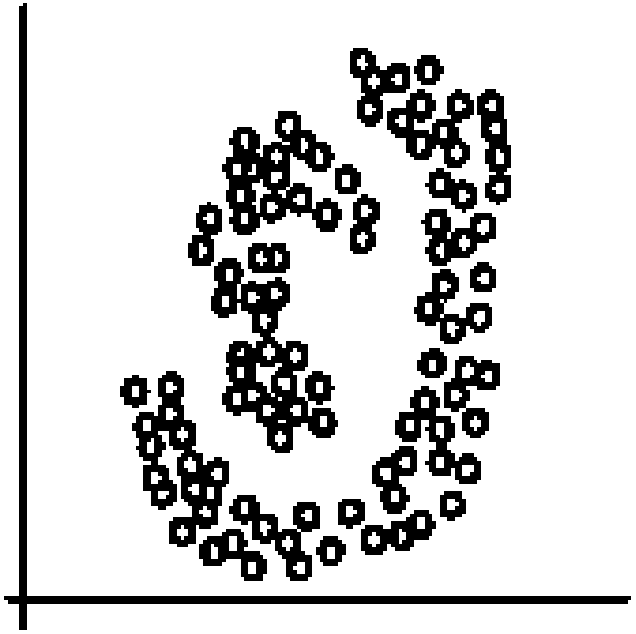
(B). Iteration 1



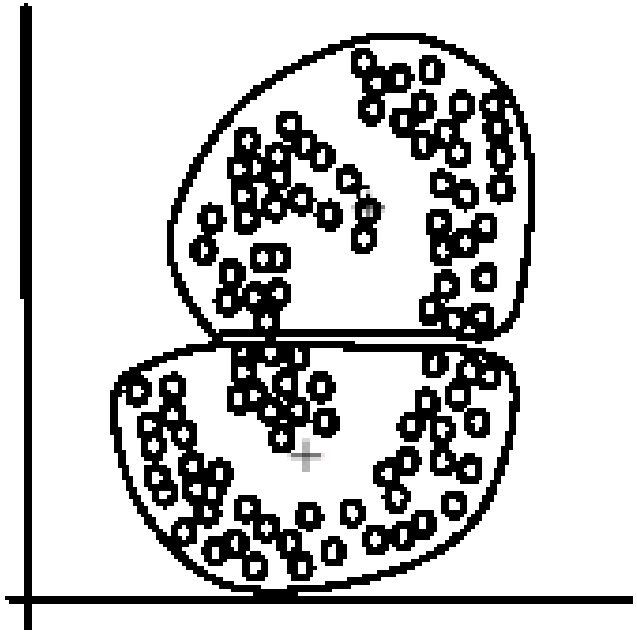
(C). Iteration 2

Weaknesses of k-means (cont …)

- The *k*-means algorithm is not suitable for discovering clusters that are not hyper-ellipsoids (or hyper-spheres).



(A): Two natural clusters



(B): *k*-means clusters

K-means summary

- Despite weaknesses, *k*-means is still the most popular algorithm due to its simplicity, efficiency and
  - other clustering algorithms have their own lists of weaknesses.
- No clear evidence that any other clustering algorithm performs better in general
  - although they may be more suitable for some specific types of data or applications.
- Comparing different clustering algorithms is a difficult task. No one knows the correct clusters!

K-MEANS CLUSTERING

What is clustering?

- **Clustering** is the classification of objects into different groups, or more precisely, the partitioning of a data set into subsets (clusters), so that the data in each subset (ideally) share some common trait - often according to some defined distance measure.

Types of clustering:

1. **Hierarchical algorithms**: these find successive clusters using previously established clusters.

   1. Agglomerative ("bottom-up"): Agglomerative algorithms begin with each element as a separate cluster and merge them into successively larger clusters.

   2. Divisive ("top-down"): Divisive algorithms begin with the whole set and proceed to divide it into successively smaller clusters.

2. **Partitional clustering:** Partitional algorithms determine all clusters at once. They include:

   - ***K*-means and derivatives**
   - Fuzzy *c*-means clustering
   - QT clustering algorithm

Common Distance measures:

- *Distance measure* will determine how the *similarity* of two elements is calculated and it will influence the shape of the clusters.

They include:

1. The Euclidean distance (also called 2-norm distance) is given by:

$$d(x, y) = \sum_{i=1}^{p} |x_i - y_i|$$

2. The Manhattan distance (also called taxicab norm or 1-norm) is given by:

$$d(x, y) = \sqrt[2]{\sum_{i=1}^{p} |x_i - y_i|^2}$$

3.The maximum norm is given by:

$$d(x, y) = \max_{1 \le i \le p} |x_i - y_i|$$

4. The Mahalanobis distance corrects data for different scales and correlations in the variables.

5. Inner product space: The angle between two vectors can be used as a distance measure when clustering high dimensional data

6. Hamming distance (sometimes edit distance) measures the minimum number of substitutions required to change one member into another.

K-MEANS CLUSTERING

- The **k-means algorithm** is an algorithm to cluster *n* objects based on attributes into *k* partitions, where $k < n$.
- It is similar to the expectation-maximization algorithm for mixtures of Gaussians in that they both attempt to find the centers of natural clusters in the data.
- It assumes that the object attributes form a vector space.

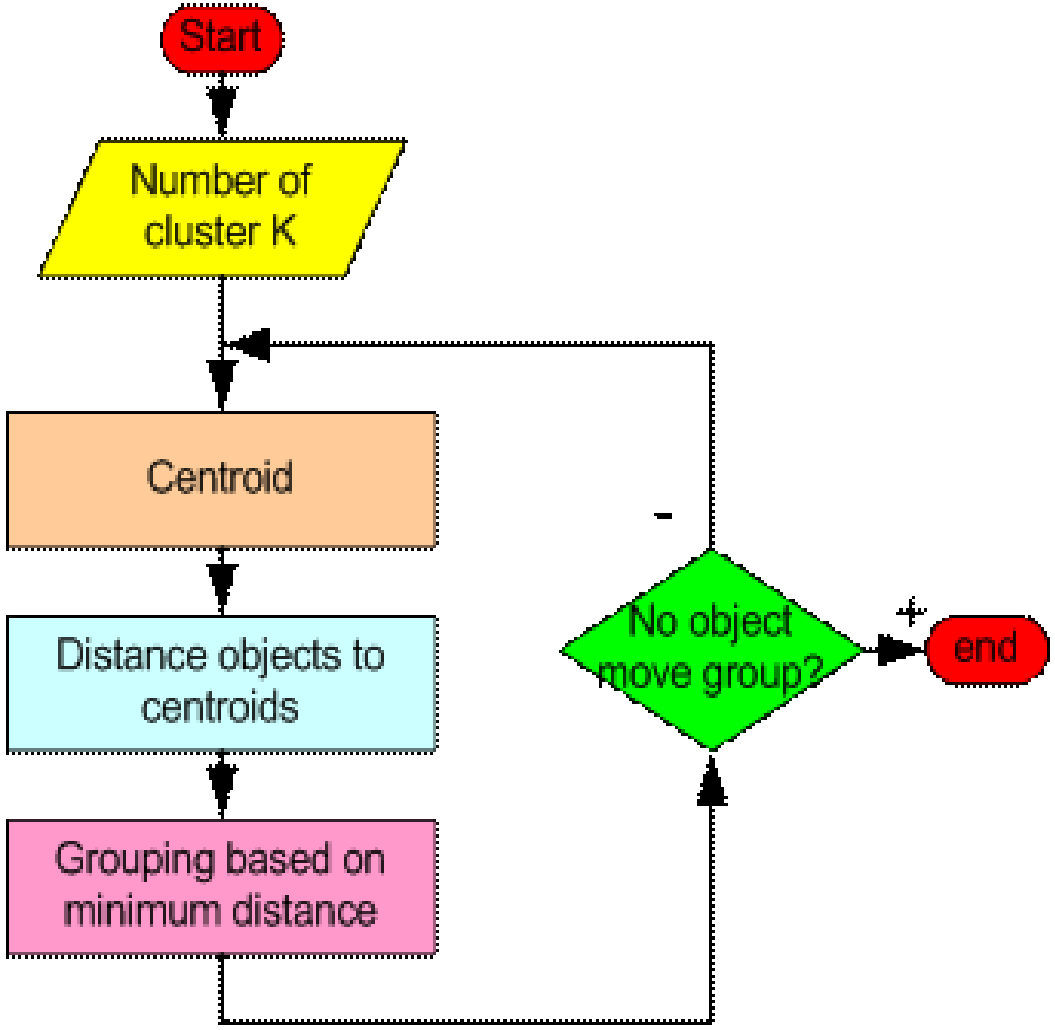
- An algorithm for partitioning (or clustering) N data points into K disjoint subsets Sj containing data points so as to minimize the sum-of-squares criterion

$$J = \sum_{j=1}^{K} \sum_{n \in S_j} \left| x_n - \mu_j \right|^2,$$

where xn is a vector representing the the nth data point and uj is the geometric centroid of the data points in Sj.

- Simply speaking k-means clustering is an algorithm to classify or to group the objects based on attributes/features into K number of group.
- K is positive integer number.
- The grouping is done by minimizing the sum of squares of distances between data and the corresponding cluster centroid.

How the K-Mean Clustering algorithm works?



- **Step 1:** Begin with a decision on the value of k = number of clusters .
- **Step 2**: Put any initial partition that classifies the data into k clusters. You may assign the training samples randomly,or systematically as the following:

1.Take the first k training sample as single-element clusters

2. Assign each of the remaining (N-k) training sample to the cluster with the nearest centroid. After each assignment, recompute the centroid of the gaining cluster.

- **Step 3:** Take each sample in sequence and compute its distance from the centroid of each of the clusters. If a sample is not currently in the cluster with the closest centroid, switch this sample to that cluster and update the centroid of the cluster gaining the new sample and the cluster losing the sample.
- **Step 4 .** Repeat step 3 until convergence is achieved, that is until a pass through the training sample causes no new assignments.

A Simple example showing the implementation of k-means algorithm

Individual	Variable 1	Variable 2
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

Step 1:

Initialization: Randomly we choose following two centroids (k=2) for two clusters.

In this case the 2 centroid are: m1=(1.0,1.0) and m2=(5.0,7.0).

Individual	Variable 1	Variable 2
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

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

Step 2:

- Thus, we obtain two clusters containing:

  {1,2,3} and {4,5,6,7}.

- Their new centroids are:

Individual	Centroid 1	Centroid 2
1	0	7.21
2 (1.5, 2.0)	1.12	6.10
3	3.61	3.61
4	7.21	0
5	4.72	2.5
6	5.31	2.06
7	4.30	2.92

$$m_1 = \left(\frac{1}{3}(1.0+1.5+3.0), \frac{1}{3}(1.0+2.0+4.0)\right) = (1.83, 2.33)$$

$$m_2 = \left(\frac{1}{4}(5.0+3.5+4.5+3.5), \frac{1}{4}(7.0+5.0+5.0+4.5)\right)$$

$$= (4.12, 5.38)$$

$$d(m_1, 2) = \sqrt{|1.0-1.5|^2 + |1.0-2.0|^2} = 1.12$$

$$d(m_2, 2) = \sqrt{|5.0-1.5|^2 + |7.0-2.0|^2} = 6.10$$

Step 3:

- Now using these centroids we compute the Euclidean distance of each object, as shown in table.

- Therefore, the new clusters are:

  {1,2} and {**3**,4,5,6,7}

- Next centroids are: m1=(1.25,1.5) and m2 = (3.9,5.1)

Individual	Centroid 1	Centroid 2
1	1.57	5.38
2	0.47	4.28
3	2.04	1.78
4	5.64	1.84
5	3.15	0.73
6	3.78	0.54
7	2.74	1.08

- Step 4 :

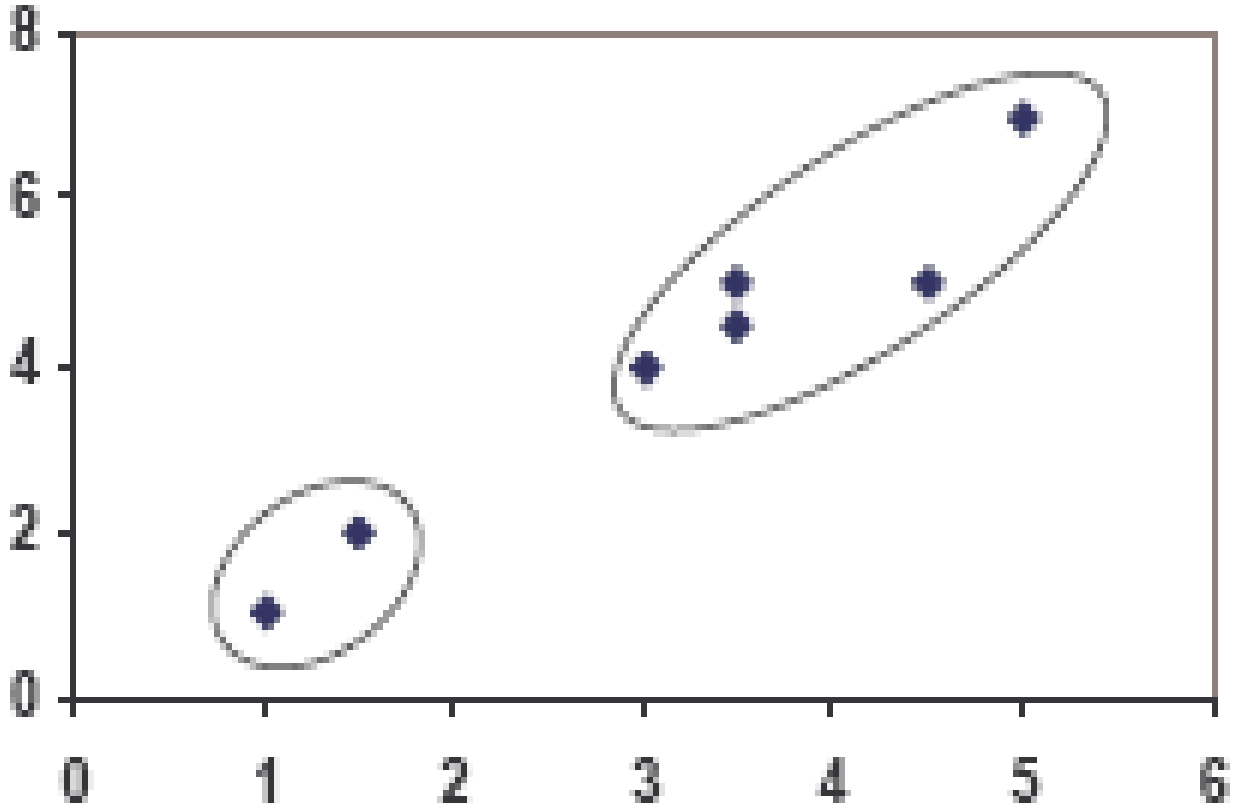
  The clusters obtained are:

  {1,2} and {3,4,5,6,7}

- Therefore, there is no change in the cluster.
- Thus, the algorithm comes to a halt here and final result consist of 2 clusters {1,2} and {3,4,5,6,7}.

Individual	Centroid 1	Centroid 2
1	0.56	5.02
2	0. 56	3.92
3	3.05	1.42
4	6.66	2.20
5	4.16	0.41
6	4.78	0.61
7	3.75	0.72

PLOT



(with K=3)

Individual	$m_1 = 1$	$m_2 = 2$	$m_3 = 3$	cluster
1	0	1.11	3.61	1
2	1.12	0	2.5	2
3	3.61	2.5	0	3
4	7.21	6.10	3.61	3
5	4.72	3.61	1.12	3
6	5.31	4.24	1.80	3
7	4.30	3.20	0.71	3

C_3

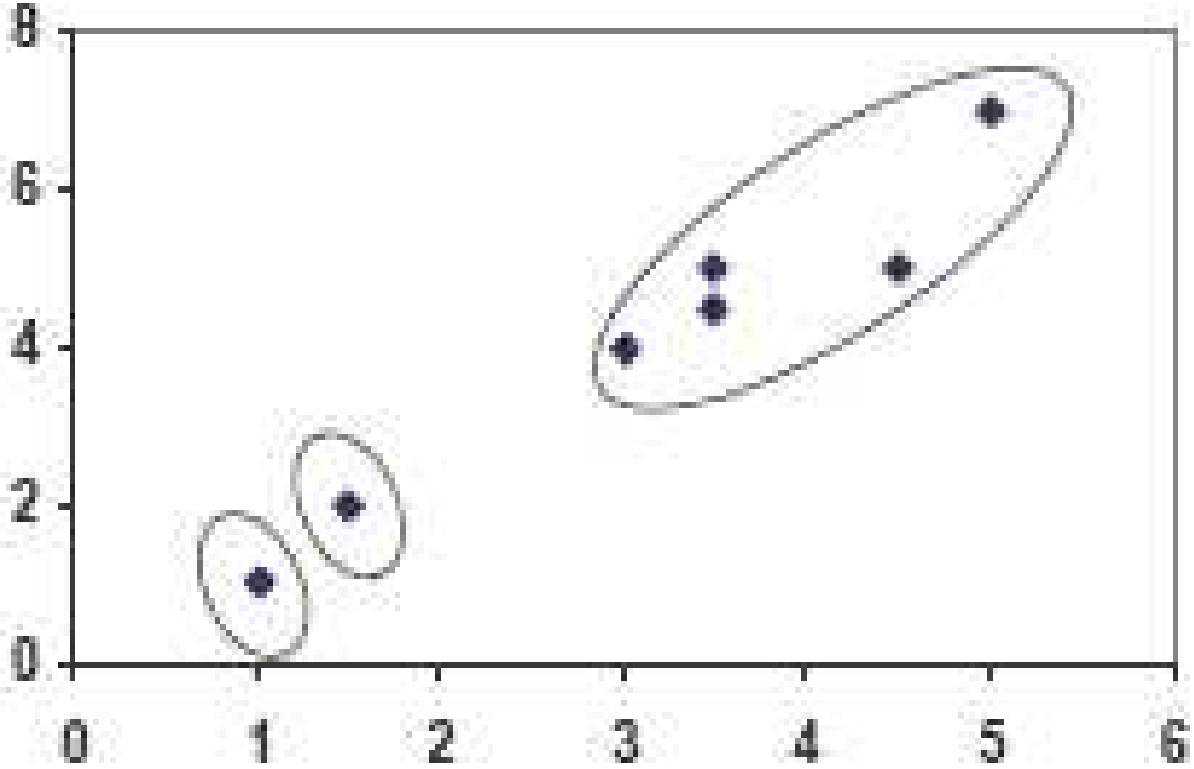
clustering with initial centroids (1, 2, 3)

Step 1

Individual	m_1 (1.0, 1.0)	m_2 (1.5, 2.0)	m_3 (3.9,5.1)	cluster
1	0	1.11	5.02	1
2	1.12	0	3.92	2
3	3.61	2.5	1.42	3
4	7.21	6.10	2.20	3
5	4.72	3.61	0.41	3
6	5.31	4.24	0.61	3
7	4.30	3.20	0.72	3

Step 2

PLOT



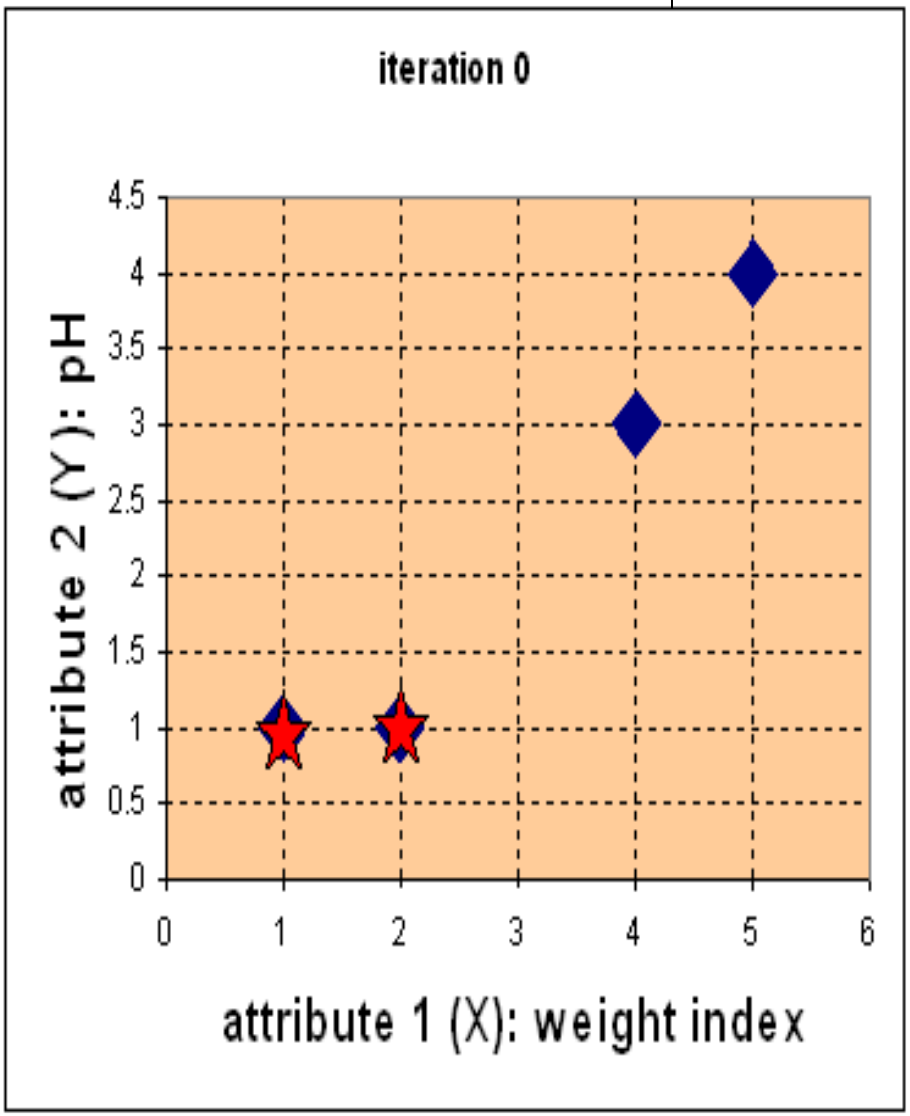
Real-Life Numerical Example of K-Means Clustering

We have 4 medicines as our training data points object and each medicine has 2 attributes. Each attribute represents coordinate of the object. We have to determine which medicines belong to cluster 1 and which medicines belong to the other cluster.

Object	Attribute1 (X): weight index	Attribute 2 (Y): pH
Medicine A	1	1
Medicine B	2	1
Medicine C	4	3
Medicine D	5	4

Step 1:

- **Initial value of centroids** : Suppose we use medicine A and medicine B as the first centroids.
- Let and c1 and c2 denote the coordinate of the centroids, then c1=(1,1) and c2=(2,1)



- **Objects-Centroids distance** : we calculate the distance between cluster centroid to each object. Let us use Euclidean distance, then we have distance matrix at iteration 0 is

$$\mathbf{D}^0 = \begin{bmatrix} 0 & 1 & 3.61 & 5 \\ 1 & 0 & 2.83 & 4.24 \end{bmatrix} \quad \begin{matrix} \mathbf{c}_1 = (1,1) & group-1 \\ \mathbf{c}_2 = (2,1) & group-2 \end{matrix}$$

$$\begin{matrix} A & B & C & D \end{matrix}$$

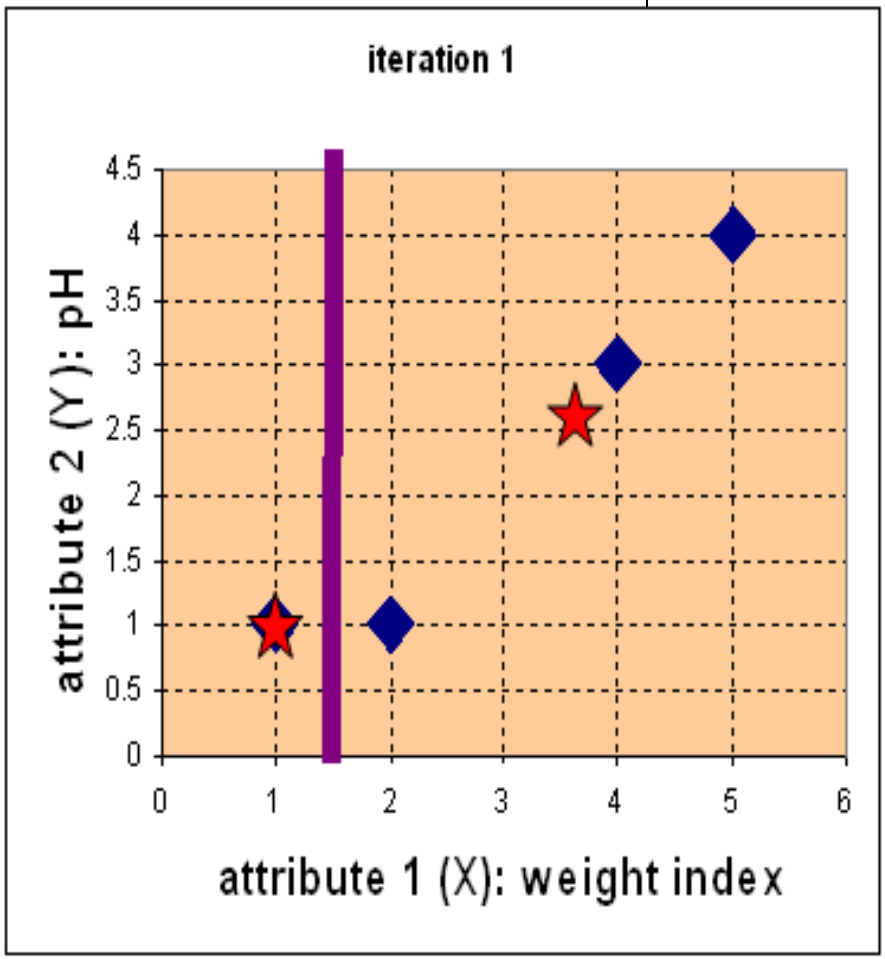
$$\begin{bmatrix} 1 & 2 & 4 & 5 \\ 1 & 1 & 3 & 4 \end{bmatrix} \quad \begin{matrix} X \\ Y \end{matrix}$$

- Each column in the distance matrix symbolizes the object.
- The first row of the distance matrix corresponds to the distance of each object to the first centroid and the second row is the distance of each object to the second centroid.
- For example, distance from medicine C = (4, 3) to the first centroid $\mathbf{c}_1 = (1,1)$ is , $\sqrt{(4-1)^2+(3-1)^2} = 3.61$ and its distance to the second centroid is , $\mathbf{c}_2 = (2,1)$ is $\sqrt{(4-2)^2+(3-1)^2} = 2.83$ etc.

Step 2:

- **Objects clustering** : We assign each object based on the minimum distance.
- Medicine A is assigned to group 1, medicine B to group 2, medicine C to group 2 and medicine D to group 2.
- The elements of Group matrix below is 1 if and only if the object is assigned to that group.

$$\mathbf{G}^0 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \end{bmatrix} \quad \begin{matrix} group-1 \\ group-2 \end{matrix}$$
$$A \quad B \quad C \quad D$$



- **Iteration-1, Objects-Centroids distances** : The next step is to compute the distance of all objects to the new centroids.
- Similar to step 2, we have distance matrix at iteration 1 is

$$D^1 = \begin{bmatrix} 0 & 1 & 3.61 & 5 \\ 3.14 & 2.36 & 0.47 & 1.89 \end{bmatrix} \quad \begin{matrix} c_1 = (1,1) & group-1 \\ c_2 = (\frac{11}{3}, \frac{8}{3}) & group-2 \end{matrix}$$

$$\begin{matrix} A & B & C & D \end{matrix}$$

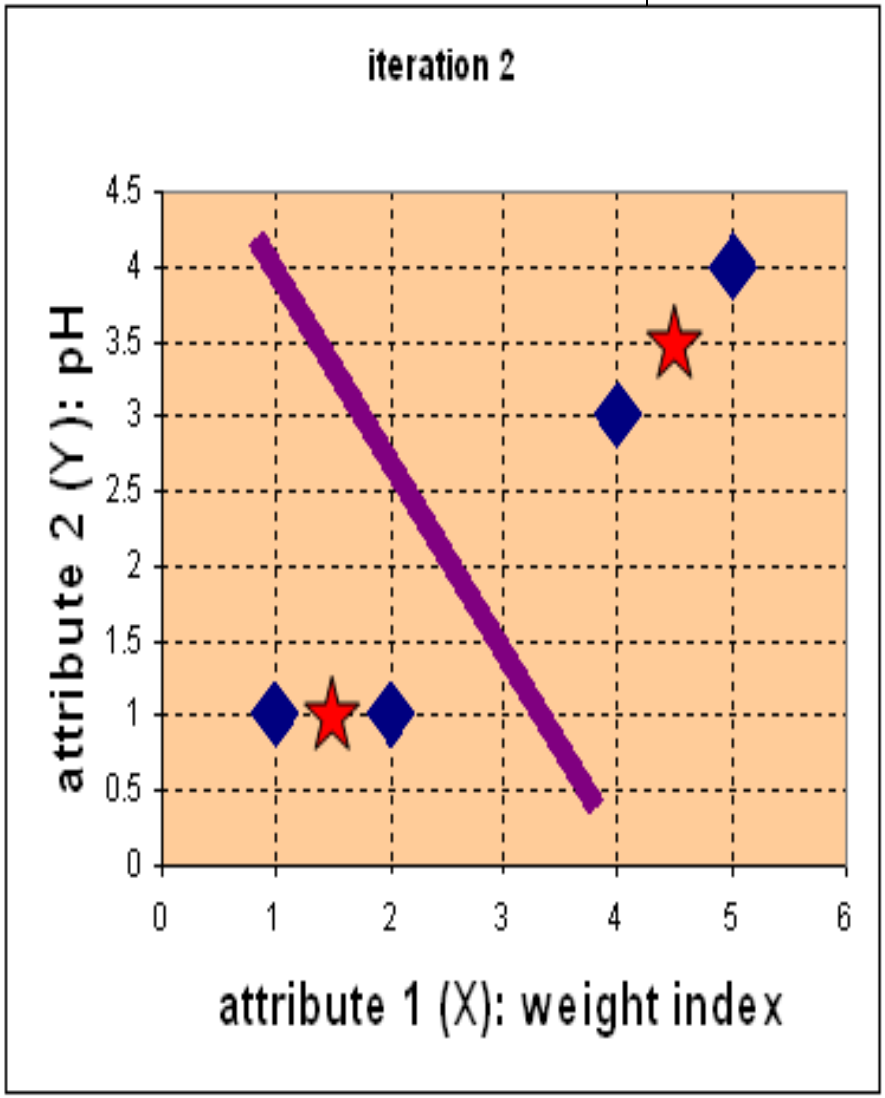
$$\begin{bmatrix} 1 & 2 & 4 & 5 \\ 1 & 1 & 3 & 4 \end{bmatrix} \quad \begin{matrix} X \\ Y \end{matrix}$$

- **Iteration-1, Objects clustering:** Based on the new distance matrix, we move the medicine B to Group 1 while all the other objects remain. The Group matrix is shown below

$$G^1 = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} \quad \begin{matrix} group-1 \\ group-2 \end{matrix}$$

$$A \quad B \quad C \quad D$$

- **Iteration 2, determine centroids:** Now we repeat step 4 to calculate the new centroids coordinate based on the clustering of previous iteration. Group1 and group 2 both has two members, thus the new centroids are $c_1 = (\frac{1+2}{2}, \frac{1+1}{2}) = (1\frac{1}{2}, 1)$ and $c_2 = (\frac{4+5}{2}, \frac{3+4}{2}) = (4\frac{1}{2}, 3\frac{1}{2})$



- **<u>Iteration-2, Objects-Centroids distances</u>** : Repeat step 2 again, we have new distance matrix at iteration 2 as

$$\mathbf{D}^2 = \begin{bmatrix} 0.5 & 0.5 & 3.20 & 4.61 \\ 4.30 & 3.54 & 0.71 & 0.71 \end{bmatrix} \quad \begin{matrix} \mathbf{c}_1 = (1\frac{1}{2}, 1) \quad group-1 \\ \mathbf{c}_2 = (4\frac{1}{2}, 3\frac{1}{2}) \quad group-2 \end{matrix}$$

$$\begin{matrix} A & B & C & D \end{matrix}$$

$$\begin{bmatrix} 1 & 2 & 4 & 5 \\ 1 & 1 & 3 & 4 \end{bmatrix} \quad \begin{matrix} X \\ Y \end{matrix}$$

- **<u>Iteration-2, Objects clustering:</u>** Again, we assign each object based on the minimum distance.

$$\mathbf{G}^2 = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} \begin{matrix} group-1 \\ group-2 \end{matrix}$$

$$A \quad B \quad C \quad D$$

- We obtain result that $\mathbf{G}^2 = \mathbf{G}^1$. Comparing the grouping of last iteration and this iteration reveals that the objects does not move group anymore.
- Thus, the computation of the k-mean clustering has reached its stability and no more iteration is needed..

We get the final grouping as the results as:

Object	Feature1(X): weight index	Feature2 (Y): pH	Group (result)
Medicine A	1	1	1
Medicine B	2	1	1
Medicine C	4	3	2
Medicine D	5	4	2

K-Means Clustering Visual Basic Code

```
Sub kMeanCluster (Data() As Variant, numCluster As Integer)
' main function to cluster data into k number of Clusters
' input:
' + Data matrix (0 to 2, 1 to TotalData);
' Row 0 = cluster, 1 =X, 2= Y; data in columns
' + numCluster: number of cluster user want the data to be clustered
' + private variables: Centroid, TotalData
' ouput:
' o) update centroid
' o) assign cluster number to the Data (= row 0 of Data)

Dim i As Integer
Dim j As Integer
Dim X As Single
Dim Y As Single
Dim min As Single
Dim cluster As Integer
Dim d As Single
Dim sumXY()

Dim isStillMoving As Boolean
isStillMoving = True
if totalData <= numCluster Then
'only the last data is put here because it designed to be interactive
Data(0, totalData) = totalData ' cluster No = total data
Centroid(1, totalData) = Data(1, totalData) ' X
Centroid(2, totalData) = Data(2, totalData) ' Y
Else
'calculate minimum distance to assign the new data
min = 10 ^ 10 'big number
X = Data(1, totalData)
Y = Data(2, totalData)
For i = 1 To numCluster
```

```
Do While isStillMoving
' this loop will surely convergent
'calculate new centroids
' 1 =X, 2=Y, 3=count number of data
ReDim sumXY(1 To 3, 1 To numCluster)
For i = 1 To totalData
sumXY(1, Data(0, i)) = Data(1, i) + sumXY(1, Data(0, i))
sumXY(2, Data(0, i)) = Data(2, i) + sumXY(2, Data(0, i))
Data(0, i))
sumXY(3, Data(0, i)) = 1 + sumXY(3, Data(0, i))
Next i
For i = 1 To numCluster
Centroid(1, i) = sumXY(1, i) / sumXY(3, i)
Centroid(2, i) = sumXY(2, i) / sumXY(3, i)
Next i
'assign all data to the new centroids
isStillMoving = False

For i = 1 To totalData
min = 10 ^ 10 'big number
X = Data(1, i)
Y = Data(2, i)
For j = 1 To numCluster
d = dist(X, Y, Centroid(1, j), Centroid(2, j))
If d < min Then
min = d
cluster = j
End If
Next j
If Data(0, i) <> cluster Then
Data(0, i) = cluster
isStillMoving = True
End If
Next i
Loop
End If
End Sub
```

Weaknesses of K-Mean Clustering

1. When the numbers of data are not so many, initial grouping will determine the cluster significantly.
2. The number of cluster, K, must be determined before hand. Its disadvantage is that it does not yield the same result with each run, since the resulting clusters depend on the initial random assignments.
3. We never know the real cluster, using the same data, because if it is inputted in a different order it may produce different cluster if the number of data is few.
4. It is sensitive to initial condition. Different initial condition may produce different result of cluster. The algorithm may be trapped in the *local optimum*.

Applications of K-Mean Clustering

- It is relatively *efficient and fast.* It computes result at **O(tkn),** where n is number of objects or points, k is number of clusters and t is number of iterations.
- k-means clustering can be applied to *machine learning or data mining*
- *Used on acoustic data in speech understanding to convert waveforms into one of k categories (known as Vector Quantization or Image Segmentation).*
- *Also used for choosing color palettes on old fashioned graphical display devices and Image Quantization.*

CONCLUSION

- *K-means algorithm is* useful for undirected knowledge discovery and is relatively simple. K-means has found wide spread usage in lot of fields, ranging from unsupervised learning of neural network, Pattern recognitions, Classification analysis, Artificial intelligence, image processing, machine vision, and many others.

Statistical learning; Naive Bayes Learning; Classification; Evaluation; Smoothing

Chapter 20, Sections 1–3

Attribution

Modified from Stuart Russell's slides (Berkeley)

Parts of the slides are inspired by Dan Klein's lecture material for CS 188 (Berkeley)

Outline

- ♢ Review: Inductive learning
- ♢ Bayesian learning
- ♢ Maximum *a posterior* and maximum likelihood learning
- ♢ Bayes net learning
  - ML parameter learning with complete data
  - linear regression

Review: Inductive Supervised Learning

♢ **Training Set, Data** of N examples of input-output pairs

$$(x_1, y_1) \ldots (x_N, y_N)$$

such that y_i is generated by unknown function $y = f(x)$

♢ **Learning**: discover a **hypothesis** function h that approximates the true function f

♢ **Test Set** is used to measure accuracy of hypothesis h

♢ Hypothesis h **generalizes** well if it correctly predicts the value of y in novel examples

♢ **Hypothesis space**, Hypothesis being **realizable**

Review: Classes of Learning Problems

$\diamondsuit$ **Classification:** The output y of a true function that we learn is a finite set of values, e.g., *wait* or *leave* in a restaurant; *sunny, cloudy, or rainy.*

$\diamondsuit$ **Regression:** The output y of a true function that we learn is a number, e.g., *tomorrow's temperature.*

$\diamondsuit$ Sometimes the function f is *stochastic* – strictly speaking, it is not a function of x, so what we learn is a conditional probability distribution $\mathbf{P}(Y|x)$.

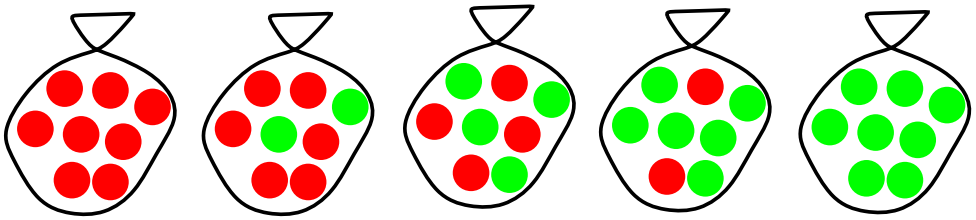
Statistical learning

♢ **Training Set, Data** – evidence – instantiations of all or some of the random variables describing the domain

♢ **Hypotheses** are probabilistic theories of how the domain works

Example: Suppose there are five kinds of bags of candies:

10% are h_1: 100% cherry candies
20% are h_2: 75% cherry candies + 25% lime candies
40% are h_3: 50% cherry candies + 50% lime candies
20% are h_4: 25% cherry candies + 75% lime candies
10% are h_5: 100% lime candies



Then we observe candies drawn from some bag: 
What kind of bag is it? What flavour will the next candy be?

Full Bayesian learning I

♢ **Bayesian learning**: calculates the probability of each hypothesis, given the data, and makes predictions on that basis

♢ The predictions are made by using *all* the hypotheses, weighted by their probabilities, rather than by using a single "best" hypothesis

♢ Thus, learning is reduced to probabilistic inference

Full Bayesian learning II

View learning as Bayesian updating of a probability distribution over the hypothesis space

H is the hypothesis variable, values $h_1, h_2, \ldots$, prior (unconditional) probabilities $\mathbf{P}(H)$

jth observation d_j gives the outcome of random variable D_j
training data $\mathbf{d} = d_1, \ldots, d_N$

Given the data so far, each hypothesis has a posterior (conditional) probability:

$$P(h_i|\mathbf{d}) = \alpha P(\mathbf{d}|h_i)P(h_i)$$

where $P(\mathbf{d}|h_i)$ is called the likelihood of the data under each hypothesis

Full Bayesian learning III

Predictions of unknown quantity X, use a likelihood-weighted average over the hypotheses:

$$\mathbf{P}(X|\mathbf{d}) = \sum_i \mathbf{P}(X|\mathbf{d}, h_i)P(h_i|\mathbf{d}) = \sum_i \mathbf{P}(X|h_i)P(h_i|\mathbf{d})$$

where we assume that *each* hypothesis determines a probability distribution over X.

No need to pick one best-guess hypothesis!

Example I

Suppose there are five kinds of bags of candies:

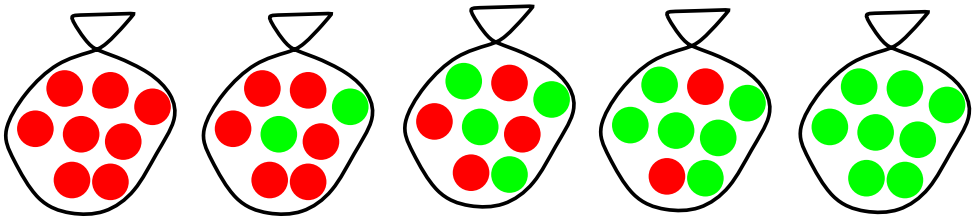
10% are h_1: 100% cherry candies

20% are h_2: 75% cherry candies + 25% lime candies

40% are h_3: 50% cherry candies + 50% lime candies

20% are h_4: 25% cherry candies + 75% lime candies

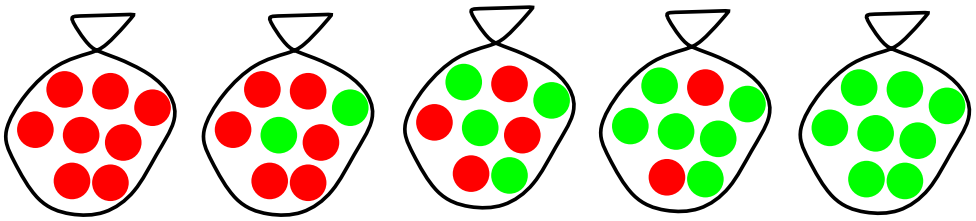
10% are h_5: 100% lime candies



Then we observe candies drawn from some bag: 

What kind of bag is it? What flavour will the next candy be?

Example I



Assume that the prior distribution over $h1, \ldots, h_5$ is given $\langle 0.1, 0.2, 0.4, 0.2, 0.1 \rangle$ (advertised by manufacture)

The likelihood of the data is calculated under the assumption that the observations are independent and identically distributed, so that

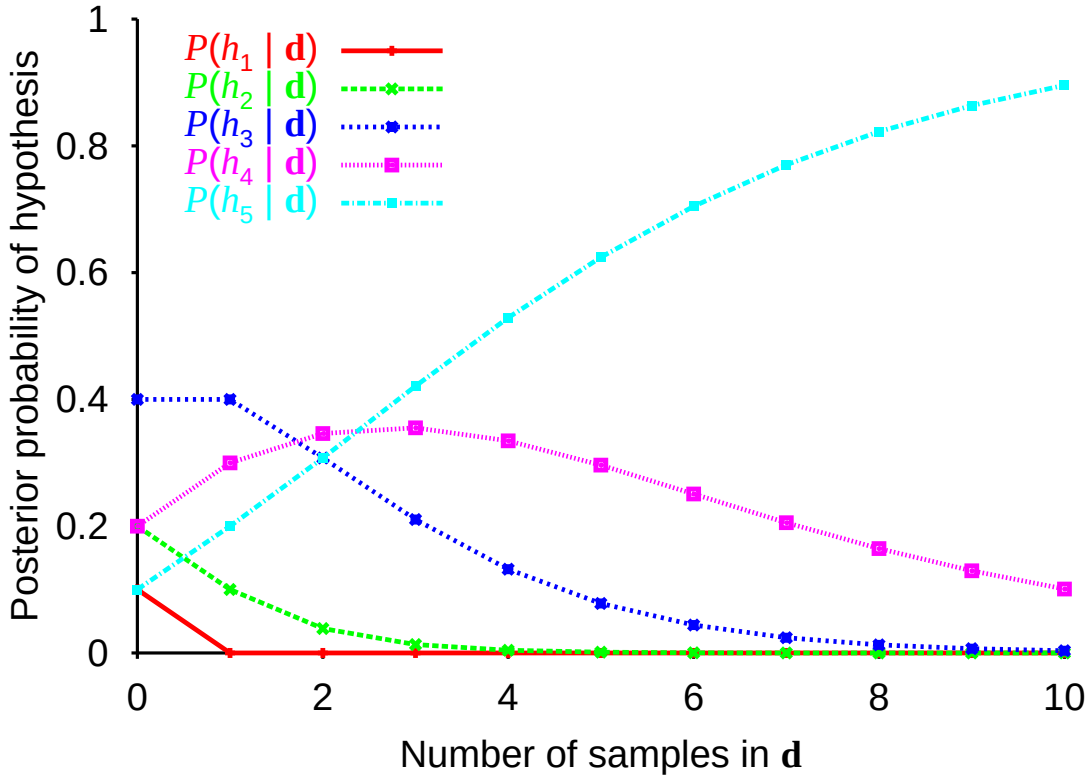
$$P(\mathbf{d}|h_i) = \Pi_j P(d_j|h_i)$$

Then two formulas can be put to work:

♢ probabilities of each hypothesis: $P(h_i|\mathbf{d}) = \alpha P(\mathbf{d}|h_i)P(h_i)$

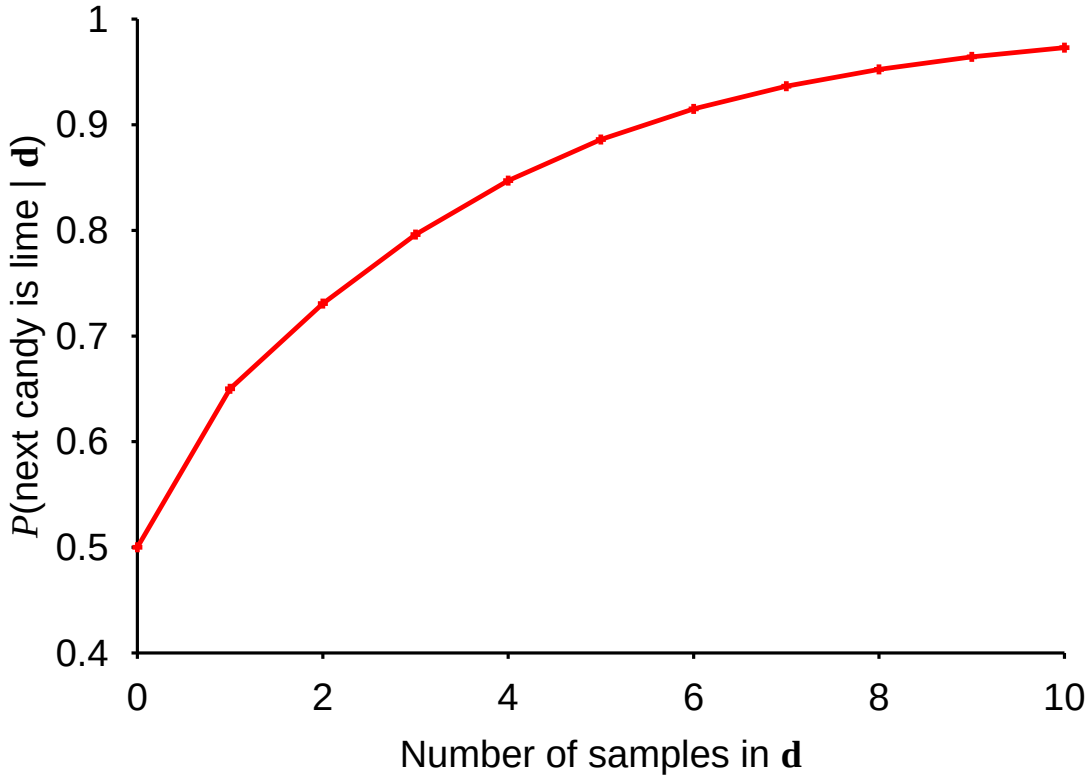
♢ predictions of unknown quantity X: $\mathbf{P}(X|\mathbf{d}) = \Sigma_i \, \mathbf{P}(X|h_i)P(h_i|\mathbf{d})$

Posterior probability of hypotheses



All observations are lime candies

Prediction probability



All observations are lime candies;

$P(\text{next candy is lime}|\mathbf{d}) = \sum_i \mathbf{P}(\text{next candy is lime}|h_i)P(h_i|\mathbf{d})$

MAP approximation I

♢ Bayesian prediction eventually agrees with the true hypothesis (under certain technical conditions)

♢ Summing over the hypothesis space is often intractable
(e.g., 18,446,744,073,709,551,616 Boolean functions of 6 attributes)

♢ Resorting to simplified methods: approximations

Maximum a posteriori (MAP) learning: choose h_{MAP} maximizing $P(h_i|\mathbf{d})$

Make a prediction based on this single *most probable* hypothesis

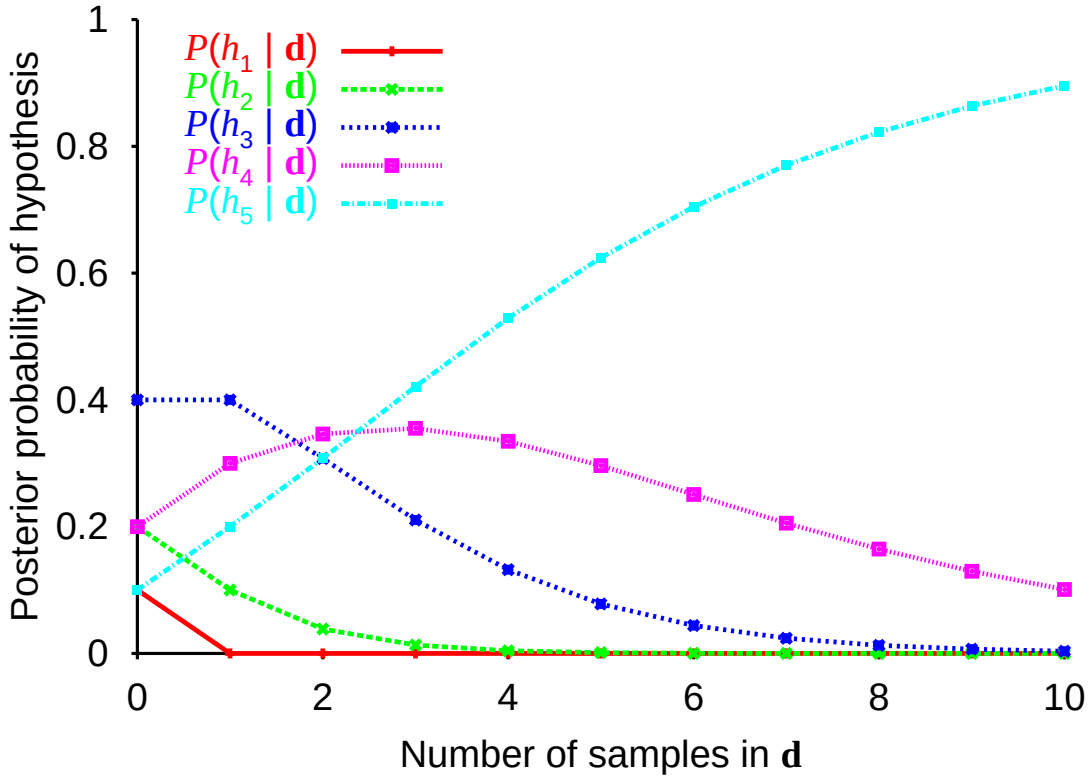
Predictions according to MAP h_{MAP} are *approximately* Bayesian:

$$\mathbf{P}(X|\mathbf{d}) = \Sigma_i \, \mathbf{P}(X|h_i)P(h_i|\mathbf{d}) \approx \mathbf{P}_{\text{MAP}}(X|\mathbf{d})$$

♢ Finding MAP hypothesis is often much easier than Bayesian learning

MAP approximation II

Availability of more data is essential for the MAP method:



All observations are lime candies. h_5 is the winner after 3 candies

Maximum Likelihood approximation

For large data sets, prior becomes irrelevant

Consider a uniform prior, e.g., for $h1, \ldots, h_5$ instead of given prior $\langle 0.1, 0.2, 0.4, 0.2, 0.1$
consider $\langle 0.2, 0.2, 0.2, 0.2, 0.2 \rangle$

Maximum likelihood (ML) learning: choose h_{ML} maximizing $P(\mathbf{d}|h_i)$

I.e., simply get the best fit to the data; identical to MAP for uniform prior
(which is reasonable if all hypotheses are of the same complexity)

(MAP) learning: choose h_{MAP} maximizing $P(h_i|\mathbf{d}) = \alpha P(\mathbf{d}|h_i)P(h_i)$

ML is the "standard" (non-Bayesian) statistical learning method

Learning a Probability Model

♢ **Training Set, Data** of N examples of input-output pairs

$$(x_1, y_1) \ldots (x_N, y_N)$$

such that y_i is generated by unknown function $y = f(x)$

♢ **Learning I**: discover a **hypothesis** function h that approximates the true function f, e.g, Decision Trees

♢ **Learning II**: Given a *fixed structure* of a **probability model** of the domain, discover its parameters from Data: **parameter learning**

As a result given parameters of a problem instance, learned probability model can be used to answer queries about problem instances

Classification: Observed parameters of a given instance and learned probability model of a domain provides probabilistic information on the likelihood of a particular classification

Classification Problems

♢ Classification is the task of predicting labels (class variables) for inputs
♢ Commercially and Scientifically Important

Examples:

- Spam Filtering
- Optical Character Recognition (OCR)
- Medical Diagnoses
- Part of Speech Tagging
- Semantic Role Labeling/Information Extraction
- Automatic essay grading
- Fraud detection

Probabilistic Models

A naive Bayes model:

$$\mathbf{P}(Cause, Effect_1, \ldots, Effect_n) = \mathbf{P}(Cause)\prod_i \mathbf{P}(Effect_i|Cause) \quad (1)$$

where "Cause" is taken to be the "class" variable, which is to be predicted. The "attribute-parameter" variables are the leaves – "Effects".

Model is "naive": assumes parameter variables to be **independent**

Model Training: using Training Set to uncover the conditional probability distribution of parameters $\mathbf{P}(Effect_i|Cause_j)$

Once the model is trained, given values of parameters of a problem instance, we can use (**??**) to classify an instance.

Independence as Abstraction

Model is "naive": assumes parameter variables to be **independent**

May lead to **overconfidence**

Indeed, all CAPS in Spam is not independent of $$ symbols

Yet, it is often a fine abstraction, and a computationally tractable one

Example: Training a Model

Optical Character Recognition

♢ Given a labeled collection M of digits in digital form

♢ nxn grid

♢ Features: $Pixel_{i,j} = on$ or off, Adj

♢ A naive Bayes model:

$$\mathbf{P}(Digit, Pixel_{1,1}, \ldots, Pixel_{n,n}, Adj) = \mathbf{P}(Digit)\prod_{i,j}\mathbf{P}(Pixel_{i,j}|Digit)\mathbf{P}(Adj)$$

Model Training Process: For M

♢ $\mathbf{P}(0) = \frac{count(M,0)}{|M|}, \ldots, \mathbf{P}(9) = \frac{count(M,9)}{|M|}$

♢ $\mathbf{P}(pixel_{1,1} = on|0) = \frac{count(M,0,on,1,1)}{count(M,0)}, \ldots$

♢ $\mathbf{P}(pixel_{1,1} = off|0) = 1 - \mathbf{P}(pixel_{1,1} = on|0), \ldots$

Example: Classification in OCR

Given parameters-attributes-features of an unseen instance and trained model we can compute

$$\mathbf{P}(0, pixel_{1,1} = on, \ldots, Pixel_{n,n} = off, Adj = true) = x_0$$
$$\ldots$$
$$\mathbf{P}(9, pixel_{1,1} = on, \ldots, Pixel_{n,n} = off, Adj) = x_9$$

and then pick the most likely class, i.e., class that corresponds to the maximum value among $x_0, \ldots, x_9$.

Evaluation

♢ Split Labeled Data into Three Categories (80/10/10; 60/20/20):

1. Training set
2. Held-out set
3. Test set

♢ Decide on Features (Parameters, Attributes): attribute-value pairs that characterize each instance

♢ Experimentation-Evaluation Cycle:

1. Learn parameters, (e.g., model probabilities) on training set
2. Tune set of features on held-out set
3. Compute accuracy on test set: **accuracy** – fraction of instances predicted correctly

Feature Engineering

Feature Engineering is crucial!

- ♢ Features translate into hypotheses space
- ♢ Too few features: cannot fit the data
- ♢ Too many features: overfitting

Generalization and Overfitting

♢ Relative frequency parameters will **overfit** the training data

- Since training set did not contain 3 with pixel i, j on during training does not mean it does not exist (but note how we will assign probability 0 to such event!)
- Unlikely that every occurrence if "minute" is 100% spam
- Unlikely that every occurrence if "seriously" is 100% ham
- Similarly, what happens to the words that never occur in training set?
- Unseen events should not be assigned 0 probability

♢ To generalize better: **smoothing** is essential

Estimation: Smoothing

♢ Intuitions Behind Smoothing

- We have some prior expectation about parameters
- Given little evidence, we should prefer prior
- Given a lot of evidence the data should rule

♢ Maximum likelihood estimate

$$P_{ML}(x) = \frac{count(x)}{\text{total samples}}$$

does not account for above intuitions

Consider three coin flips: Head, Head, Tail; what is $P_{ML}(x)$

Estimation: Laplace Smoothing

♢ Laplace's estimate

$$P_{LAP}(x) = \frac{count(x) + 1}{\text{total samples} + |X|}$$

- Pretend that every outcome appeared once more than it did
- Note how it elegantly deals with earlier unseen events

♢ Laplace's estimate – extended with strength factor:

$$P_{LAP,k}(x) = \frac{count(x) + k}{\text{total samples} + k|X|}$$

Consider three coin flips: Head, Head, Tail; what are $P_{ML}(x)$, $P_{LAP}(x)$, $P_{LAP,k}(x)$?

♢ There are many ways to introduce smoothing as well as methods to account for unknown events

Summary

Full Bayesian learning gives best possible predictions but is intractable

MAP learning balances complexity with accuracy on training data

Maximum likelihood assumes uniform prior, OK for large data sets

Learning Models, Naive Bayses Nets

Classification Problem by Means of Naive Bayses Nets

Evaluation Concepts

Smoothing

Naïve Bayes Model

Outline

- Background
- Probability Basics
- Probabilistic Classification
- Naïve Bayes
- Example: Play Tennis
- Relevant Issues
- Conclusions

Background

- There are three methods to establish a classifier

  a) Model a classification rule directly

  Examples: k-NN, decision trees, perceptron, SVM

  b) Model the probability of class memberships given input data

  Example: multi-layered perceptron with the cross-entropy cost

  c) Make a probabilistic model of data within each class

  Examples: naive Bayes, model based classifiers

- *a*) and *b*) are examples of discriminative classification
- *c*) is an example of generative classification
- *b*) and *c*) are both examples of probabilistic classification

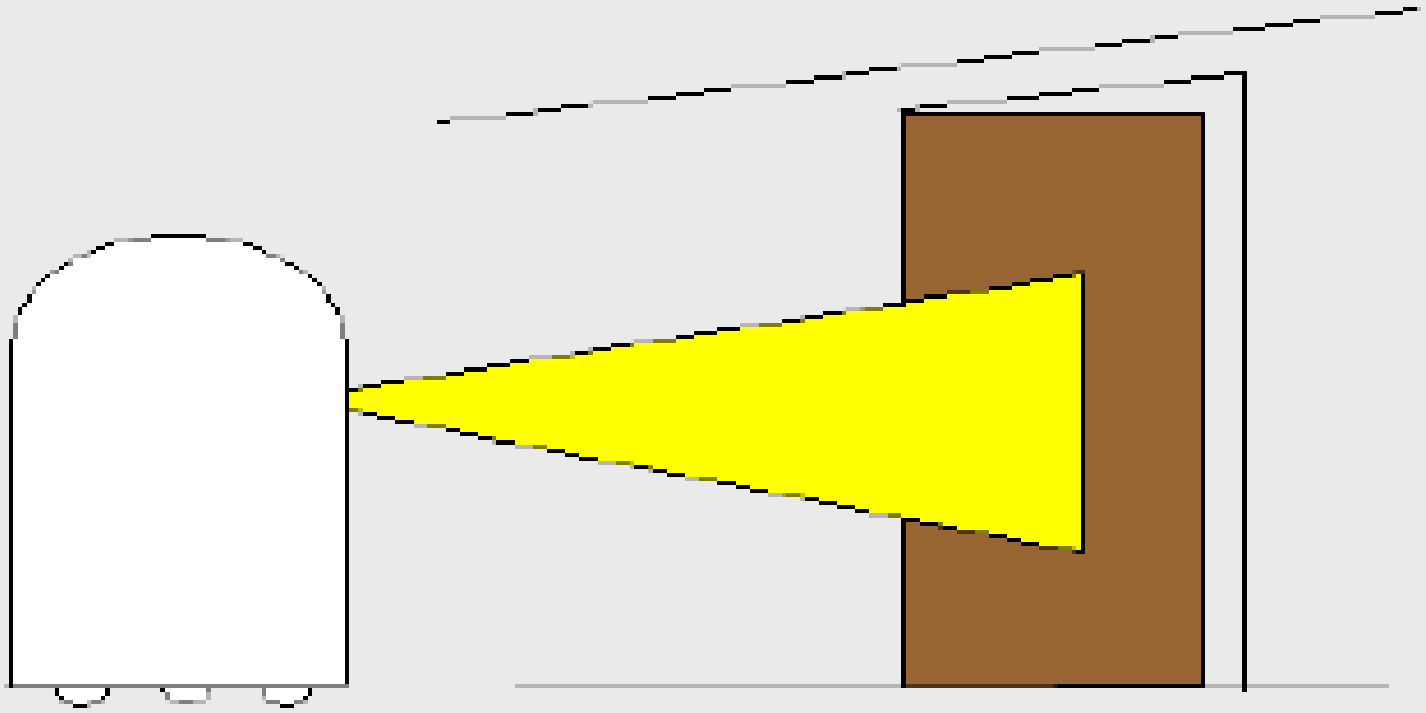
Probability Basics

- Prior, conditional and joint probability
  - Prior probability: $P(X)$
  - Conditional probability: $P(X_1 \mid X_2), P(X_2 \mid X_1)$
  - Joint probability: $\mathbf{X}$ $(X_1, X_2), P(\mathbf{X})$ $P(X_1, X_2)$
  - Relationship: $P(X_1, X_2)$ $P(X_2 \mid X_1)P(X_1)$ $P(X_1 \mid X_2)P(X_2)$
  - Independence: $P(X_2 \mid X_1)$ $P(X_2), P(X_1 \mid X_2)$ $P(X_1), P(X_1, X_2)$ $P(X_1)P(X_2)$
- Bayesian Rule

$$P(C \mid \mathbf{X}) \quad \frac{P(\mathbf{X} \mid C)P(C)}{P(\mathbf{X})} \quad Posterior \quad \frac{Likelihood \quad Prior}{Evidence}$$

Simple Example of State Estimation

- Suppose a robot obtains measurement z
- What is *P(doorOpen|z)*?



Causal vs. Diagnostic Reasoning

- $P(open|z)$ is diagnostic.
- $P(z|open)$ is causal.
- Often causal knowledge is easier to obtain.
- Bayes rule allows us to use causal knowledge:

count frequencies!

$$P(open \mid z) = \frac{P(z \mid open)P(open)}{P(z)}$$

Example

- $P(z|open) = 0.6 \quad P(z|\neg open) = 0.3$
- $P(open) = P(\neg open) = 0.5$

$$P(open \mid z) = \frac{P(z \mid open)P(open)}{P(z \mid open)p(open) + P(z \mid \neg open)p(\neg open)}$$

$$P(open \mid z) = \frac{0.6 \cdot 0.5}{0.6 \cdot 0.5 + 0.3 \cdot 0.5} = \frac{2}{3} = 0.67$$

- z raises the probability that the door is open.

Probabilistic Classification

- Establishing a probabilistic model for classification
  - Discriminative model

$$P(C \mid \mathbf{X}) \quad C \quad c_1, \quad , c_L, \mathbf{X} \quad (X_1, \quad , X_n)$$

  - Generative model

$$P(\mathbf{X} \mid C) \quad C \quad c_1, \quad , c_L, \mathbf{X} \quad (X_1, \quad , X_n)$$

- MAP classification rule
  - MAP: Maximum A Posterior
  - Assign x to c^* if $P(C \quad c^* \mid \mathbf{X} \quad \mathbf{x}) \quad P(C \quad c \mid \mathbf{X} \quad \mathbf{x}) \quad c \quad c^*, c \quad c_1, \quad , c_L$
- Generative classification with the MAP rule
  - Apply Bayesian rule to convert: $P(C \mid \mathbf{X}) \quad \frac{P(\mathbf{X} \mid C)P(C)}{P(\mathbf{X})} \quad P(\mathbf{X} \mid C)P(C)$

Naïve Bayes

- Bayes classification

$$P(C \mid \mathbf{X}) \quad P(\mathbf{X} \mid C)P(C) \quad P(X_1, \quad, X_n \mid C)P(C)$$

Difficulty: learning the joint probability $P(X_1, \quad, X_n \mid C)$

- Naïve Bayes classification
  - Making the assumption that all input attributes are independent

$$P(X_1, X_2, \quad, X_n \mid C) \quad P(X_1 \mid X_2, \quad, X_n; C)P(X_2, \quad, X_n \mid C)$$
$$P(X_1 \mid C)P(X_2, \quad, X_n \mid C)$$
$$P(X_1 \mid C)P(X_2 \mid C) \quad P(X_n \mid C)$$

  - MAP classification rule

$$[P(x_1 \mid c^*) \quad P(x_n \mid c^*)]P(c^*) \quad [P(x_1 \mid c) \quad P(x_n \mid c)]P(c), \quad c \quad c^*, c \quad c_1, \quad, c_L$$

Naïve Bayes

- Naïve Bayes Algorithm (for discrete input attributes)
  - Learning Phase: Given a training set **S**,

    For each target value of c_i $(c_i \quad c_1, \quad , c_L)$

    $\hat{P}(C \quad c_i)$ estimate $P(C \quad c_i)$ with examples in **S**;

    For every attribute value a_{jk} of each attribute x_j $(j \quad 1, \quad , n; k \quad 1, \quad , N_j)$

    $\hat{P}(X_j \quad a_{jk} \mid C \quad c_i)$ estimate $P(X_j \quad a_{jk} \mid C \quad c_i)$ with examples in **S**;

    Output: conditional probability tables; for $x_j, \; N_j \quad L$ elements
  - Test Phase: Given an unknown instance $\mathbf{X} \quad (a_1, \quad , a_n)$

    Look up tables to assign the label c^* to **X'** if

    $[\hat{P}(a_1 \mid c^*) \quad \hat{P}(a_n \mid c^*)]\hat{P}(c^*) \quad [\hat{P}(a_1 \mid c) \quad \hat{P}(a_n \mid c)]\hat{P}(c), \quad c \quad c^*, c \quad c_1, \quad , c_L$

Example

- Example: Play Tennis

PlayTennis: training examples

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Example

- Learning Phase

Outlook	Play=*Yes*	Play=*No*
Sunny	2/9	3/5
Overcast	4/9	0/5
Rain	3/9	2/5

Temperature	Play=*Yes*	Play=*No*
Hot	2/9	2/5
Mild	4/9	2/5
Cool	3/9	1/5

Humidity	Play=*Yes*	Play=No
High	3/9	4/5
Normal	6/9	1/5

Wind	Play=*Yes*	Play=*No*
Strong	3/9	3/5
Weak	6/9	2/5

$P(\text{Play}=Yes) = 9/14$ $P(\text{Play}=No) = 5/14$

Example

- Test Phase
  - Given a new instance,

    $\mathbf{x}'$=(Outlook=*Sunny*, Temperature=*Cool*, Humidity=*High*, Wind=*Strong*)
  - Look up tables

P(Outlook=*Sunny*|Play=*Yes*) = 2/9

P(Temperature=*Cool*|Play=*Yes*) = 3/9

P(Huminity=*High*|Play=*Yes*) = 3/9

P(Wind=*Strong*|Play=*Yes*) = 3/9

P(Play=*Yes*) = 9/14

P(Outlook=S*unny*|Play=*No*) = 3/5

P(Temperature=*Cool*|Play==*No*) = 1/5

P(Huminity=*High*|Play=*No*) = 4/5

P(Wind=*Strong*|Play=*No*) = 3/5

P(Play=*No*) = 5/14

  - MAP rule

P(*Yes*|$\mathbf{x}'$): [P(*Sunny*|Yes)P(*Cool*|*Yes*)P(*High*|Yes)P(*Strong*|*Yes*)]P(Play=*Yes*) = 0.0053

P(*No*|$\mathbf{x}'$): [P(*Sunny*|No) P(*Cool*|No)P(*High*|*No*)P(*Strong*|*No*)]P(Play=*No*) = 0.0206

Given the fact P(*Yes*|$\mathbf{x}'$) < P(*No*|$\mathbf{x}'$), we label $\mathbf{x}'$ to be "*No*".

Relevant Issues

- Violation of Independence Assumption
  - For many real world tasks, $P(X_1, \ldots, X_n \mid C) \quad P(X_1 \mid C) \quad P(X_n \mid C)$
  - Nevertheless, naïve Bayes works surprisingly well anyway!
- Zero conditional probability Problem
  - If no example contains the attribute value $X_j \quad a_{jk}$, $\hat{P}(X_j \quad a_{jk} \mid C \quad c_i) \quad 0$
  - In this circumstance, $\hat{P}(x_1 \mid c_i) \quad \hat{P}(a_{jk} \mid c_i) \quad \hat{P}(x_n \mid c_i) \quad 0$ during test
  - For a remedy, conditional probabilities estimated with

$$\hat{P}(X_j \quad a_{jk} \mid C \quad c_i) \quad \frac{n_c \quad mp}{n \quad m}$$

n_c : number of training examples for which $X_j \quad a_{jk}$ and $C \quad c_i$

n : number of training examples for which $C \quad c_i$

p : prior estimate (usually, $p \quad 1/t$ for t possible values of X_j)

m : weight to prior (number of "virtual" examples, $m \quad 1$)

Relevant Issues

- Continuous-valued Input Attributes
  - Numberless values for an attribute
  - Conditional probability modeled with the normal distribution

$$\hat{P}(X_j \mid C = c_i) = \frac{1}{\sqrt{2\pi}\,\sigma_{ji}} \exp\left(-\frac{(X_j - \mu_{ji})^2}{2\sigma_{ji}^2}\right)$$

μ_{ji} : mean (avearage) of attribute values X_j of examples for which $C = c_i$

σ_{ji} : standard deviation of attribute values X_j of examples for which $C = c_i$

  - Learning Phase: for $\mathbf{X} = (X_1, \ldots, X_n)$, $C = c_1, \ldots, c_L$
    Output: normal distributions and $P(C = c_i)$ $i = 1, \ldots, L$
  - Test Phase: for $\mathbf{X} = (X_1, \ldots, X_n)$
    - Calculate conditional probabilities with all the normal distributions
    - Apply the MAP rule to make a decision

Conclusions

- Naïve Bayes based on the independence assumption
  - Training is very easy and fast; just requiring considering each attribute in each class separately
  - Test is straightforward; just looking up tables or calculating conditional probabilities with normal distributions
- A popular generative model
  - Performance competitive to most of state-of-the-art classifiers even in presence of violating independence assumption
  - Many successful applications, e.g., spam mail filtering
  - Apart from classification, naïve Bayes can do more…

EM Algorithm

Introduction

- The EM algorithm was explained and given its name in a classi c 1977 paper by Arthur Dempster, Nan Laird, and Donald Rubi n.
- They pointed out that the method had been "proposed many ti mes in special circumstances" by earlier authors.
- EM is typically used to compute maximum likelihood estimates given incomplete samples.
- The EM algorithm estimates the parameters of a model iterative ly.
  - Starting from some initial guess, each iteration consists of
    - an E step (Expectation step)
    - an M step (Maximization step)

Applications

- Filling in missing data in samples
- Discovering the value of latent variables
- Estimating the parameters of HMMs
- Estimating parameters of finite mixtures
- Unsupervised learning of clusters
- …

EM Algorithm

Simple Example

Silly Example

Let events be "grades in a class"

w_1 = Gets an A P(A) = ½

w_2 = Gets a B P(B) = μ

w_3 = Gets a C P(C) = 2μ

w_4 = Gets a D P(D) = ½-3μ

(Note $0 \leq \mu \leq 1/6$)

Assume we want to estimate μ from data. In a given class there were

a A's
b B's
c C's
d D's

What's the maximum likelihood estimate of μ given a,b,c,d ?

Trivial Statistics

P(A) = ½ P(B) = μ P(C) = 2μ P(D) = ½-3μ

P(*a,b,c,d* | μ) = $(½)^a(μ)^b(2μ)^c(½-3μ)^d$

log P(*a,b,c,d* | μ) = *a*log ½ + *b*log μ + *c*log 2μ + *d*log (½-3μ)

FOR MAX LIKE μ, SET $\frac{\partial \text{LogP}}{\partial \mu} = 0$

$$\frac{\partial \text{LogP}}{\partial \mu} = \frac{b}{\mu} + \frac{2c}{2\mu} - \frac{3d}{1/2 - 3\mu} = 0$$

Gives max like $\mu = \frac{b+c}{6(b+c+d)}$

So if class got

A	B	C	D
14	6	9	10

Max like $\mu = \frac{1}{10}$

Boring, but true!

EM Algorithm

Problem

with Hidden Info

Same Problem with Hidden Information

Someone tells us that

Number of High grades (A's + B's) = h

Number of C's = c

Number of D's = d

What is the max. like estimate of μ now?

REMEMBER

P(A) = ½

P(B) = μ

P(C) = 2μ

P(D) = ½-3μ

Same Problem with Hidden Information

REMEMBER
P(A) = ½
P(B) = μ
P(C) = 2μ
P(D) = ½-3μ

Someone tells us that

Number of High grades (A's + B's) = h

Number of C's = c

Number of D's = d

What is the max. like estimate of μ now?

We can answer this question circularly:

EXPECTATION

If we know the value of μ we could compute the expected value of a and b

Since the ratio a:b should be the same as the ratio ½ : μ

$$a = \frac{1/2}{1/2 + \mu} h \qquad b = \frac{\mu}{1/2 + \mu} h$$

MAXIMIZATION

If we know the expected values of a and b we could compute the maximum likelihood value of μ

$$\mu = \frac{b + c}{6(b + c + d)}$$

E.M. for our Trivial Problem

REMEMBER

P(A) = ½

P(B) = μ

P(C) = 2μ

P(D) = ½-3μ

We begin with a guess for μ

We iterate between EXPECTATION and MAXIMALIZATION to improve our estimates of μ and *a* and *b*.

Define μ(t) the estimate of μ on the t'th iteration

b(t) the estimate of *b* on t'th iteration

$$\mu(0) = \text{initial guess}$$

$$b(t) = \frac{\mu(t)h}{1/2 + \mu(t)} = \mathrm{E}[b \mid \mu(t)]$$

E-step

$$\mu(t+1) = \frac{b(t)+c}{6(b(t)+c+d)}$$

$$= \max \text{ like est of } \mu \text{ given } b(t)$$

M-step

Continue iterating until converged.

Good news: Converging to local optimum is assured.

Bad news: I said "local" optimum.

EM Algorithm

EM Clustering Algorithm

Given a training data set: X={x(1),x(2),...,x(n)}

Z={z(1),z(2),...,z(n)}

z(i) is the calss/group label of sample x(i).

As we are in Clustering setting,

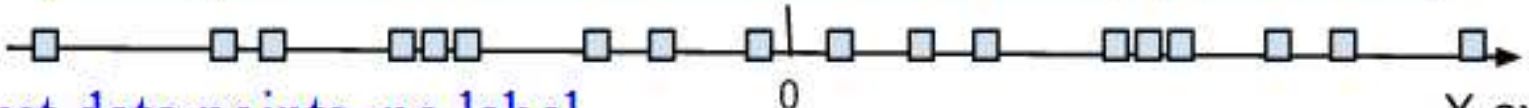
X is Given and Z is unknown

Now, we model the data by specifying a joint distribution p(x(i), z(i))=p(x(i)|z(i))p(z(i))

$$z^{(i)} \sim \text{Multinomial}(\phi)$$
$$\phi_j \geq 0, \sum_{i=1}^{k} \phi_j = 1$$
$$k = \#\ \text{of}\ z^{(i)}\text{'s values}$$
$$\phi_j = p(z^{(i)} = j)$$
$$x^{(i)} | z^{(i)} = j \sim \mathcal{N}(\mu_j, \Sigma_j)$$

⇨ each $x^{(i)}$ was generated by randomly choosing $z^{(i)}$ from $\{1, \ldots, k\}$, and then $x^{(i)}$ was drawn from one of k Gaussians.

The parameters of our model are thus ϕ, μ and Σ.



Just data points, no label

E-M

Incomplete Data

X={x(1),x(2),...,x(n)} Given
Z={z(1),z(2),...,z(n)} unknown

unknown

The parameters of our model ϕ, μ, Σ

What is the value of z(i)?

We can answer this question circularly:

EXPECTATION

If we know the expected values of Z we could compute the maximum likelihood value of ϕ, μ, Σ

MAXIMIZATION

If we know the values of ϕ, μ, Σ we could compute the expected values of Z

We begin with a guess for ϕ, μ, Σ, and then iterate between EXPECTATION and MAXIMALIZATION to improve our estimates of ϕ, μ, Σ and Z

Continue iterating until converged.

$$\ell(\phi, \mu, \Sigma) = \sum_{i=1}^{m} \log p(x^{(i)}|z^{(i)}; \mu, \Sigma) + \log p(z^{(i)}; \phi)$$

Maximizing this with respect to ϕ, μ and Σ gives the parameters:

$$\begin{aligned}
\phi_j &= \frac{1}{m} \sum_{i=1}^{m} 1\{z^{(i)} = j\}, \\
\mu_j &= \frac{\sum_{i=1}^{m} 1\{z^{(i)} = j\} x^{(i)}}{\sum_{i=1}^{m} 1\{z^{(i)} = j\}}, \\
\Sigma_j &= \frac{\sum_{i=1}^{m} 1\{z^{(i)} = j\} (x^{(i)} - \mu_j)(x^{(i)} - \mu_j)^T}{\sum_{i=1}^{m} 1\{z^{(i)} = j\}}.
\end{aligned}$$

Repeat until convergence: {

(E-step) For each i, j, set

$$w_j^{(i)} := p(z^{(i)} = j | x^{(i)}; \phi, \mu, \Sigma)$$

(M-step) Update the parameters:

$$\begin{aligned}
\phi_j &:= \frac{1}{m} \sum_{i=1}^{m} w_j^{(i)}, \\
\mu_j &:= \frac{\sum_{i=1}^{m} w_j^{(i)} x^{(i)}}{\sum_{i=1}^{m} w_j^{(i)}}, \\
\Sigma_j &:= \frac{\sum_{i=1}^{m} w_j^{(i)} (x^{(i)} - \mu_j)(x^{(i)} - \mu_j)^T}{\sum_{i=1}^{m} w_j^{(i)}}
\end{aligned}$$

}

Naive Bayes Classifiers

This article discusses the theory behind the Naive Bayes classifiers and their implementation.

Naive Bayes classifiers are a collection of classification algorithms based on **Bayes' Theorem**. It is not a single algorithm but a family of algorithms where all of them share a common principle, i.e. every pair of features being classified is independent of each other.
To start with, let us consider a dataset.

Consider a fictional dataset that describes the weather conditions for playing a game of golf. Given the weather conditions, each tuple classifies the conditions as fit("Yes") or unfit("No") for plaing golf.

Here is a tabular representation of our dataset.

	OUTLOOK	TEMPERATURE	HUMIDITY	WINDY	PLAY GOLF
0	Rainy	Hot	High	False	No
1	Rainy	Hot	High	True	No
2	Overcast	Hot	High	False	Yes
3	Sunny	Mild	High	False	Yes
4	Sunny	Cool	Normal	False	Yes
5	Sunny	Cool	Normal	True	No

	OUTLOOK	TEMPERATURE	HUMIDITY	WINDY	PLAY GOLF
6	Overcast	Cool	Normal	True	Yes
7	Rainy	Mild	High	False	No
8	Rainy	Cool	Normal	False	Yes
9	Sunny	Mild	Normal	False	Yes
10	Rainy	Mild	Normal	True	Yes
11	Overcast	Mild	High	True	Yes
12	Overcast	Hot	Normal	False	Yes
13	Sunny	Mild	High	True	No

The dataset is divided into two parts, namely, **feature matrix** and the **response vector**.

- Feature matrix contains all the vectors(rows) of dataset in which each vector consists of the value of **dependent features**. In above dataset, features are 'Outlook', 'Temperature', 'Humidity' and 'Windy'.

- Response vector contains the value of **class variable**(prediction or output) for each row of feature matrix. In above dataset, the class variable name is 'Play golf'.

Assumption:
The fundamental Naive Bayes assumption is that each feature makes an:

- independent
- equal

contribution to the outcome.

With relation to our dataset, this concept can be understood as:

- We assume that no pair of features are dependent. For example, the temperature being 'Hot' has nothing to do with the humidity or the outlook being 'Rainy' has no effect on the winds. Hence, the features are assumed to be **independent**.

- Secondly, each feature is given the same weight(or importance). For example, knowing only temperature and humidity alone can't predict the outcome accuratey. None of the attributes is irrelevant and assumed to be contributing **equally** to the outcome.

Note: The assumptions made by Naive Bayes are not generally correct in real-world situations. In-fact, the independence assumption is never correct but often works well in practice.
Now, before moving to the formula for Naive Bayes, it is important to know about Bayes' theorem.

Bayes' Theorem

Bayes' Theorem finds the probability of an event occurring given the probability of another event that has already occurred. Bayes' theorem is stated mathematically as the following equation:

$$P(A|B) = \frac{P(B|A)\,P(A)}{P(B)}$$

where A and B are events

- Basically, we are trying to find probability of event A, given the event B is true. Event B is also termed as **evidence**.
- P(A) is the **priori** of A (the prior probability, i.e. Probability of event before evidence is seen). The evidence is an attribute value of an unknown instance(here, it is event B).
- P(A|B) is a posteriori probability of B, i.e. probability of event after evidence is seen.

Now, with regards to our dataset, we can apply Bayes' theorem in following way:

$$P(y|x) = \frac{P(x|y)\,P(y)}{P(x)}$$

where, y is class variable and X is a dependent feature vector (of size *n*) where:

$$X = (x_1, x_2, x_3 \cdots x_n)$$

Just to clear, an example of a feature vector and corresponding class variable can be: (refer 1st row of dataset)

```
X = (Rainy, Hot, High, False)

y = No
```

So basically, P(X|y) here means, the probability of "Not playing golf" given that the weather conditions are "Rainy outlook", "Temperature is hot", "high humidity" and "no wind".

Naive assumption

Now, its time to put a naive assumption to the Bayes' theorem, which is, **independence**among the features. So now, we split **evidence** into the independent parts.

Now, if any two events A and B are independent, then,

```
P(A,B) = P(A)P(B)
```

Hence, we reach to the result:

$$P(y|x_1,\dots,x_n) = \frac{P(x_1|y)P(x_2|y)\cdots P(x_n|y)P(y)}{P(x_1)P(x_2)\cdots P(x_n)}$$

which can be expressed as:

$$P(y|x_1,\dots,x_n) = \frac{P(y)\prod_{i=1}^{n} P(x_i|y)}{P(x_1)P(x_2)\cdots P(x_n)}.$$

Now, as the denominator remains constant for a given input, we can remove that term:

$$P(y \mid x_1, \ldots, x_n) \propto P(y) \prod_{i=1}^{n} P(x_i \mid y)$$

Now, we need to create a classifier model. For this, we find the probability of given set of inputs for all possible values of the class variable *y* and pick up the output with maximum probability. This can be expressed mathematically as:

$$y = \operatorname{argmax}_y P(y) \prod_{i=1}^{n} P(x_i \mid y)$$

So, finally, we are left with the task of calculating P(y) and $P(x_i \mid y)$.

Please note that P(y) is also called **class probability** and $P(x_i \mid y)$ is called **conditional probability**.

The different naive Bayes classifiers differ mainly by the assumptions they make regarding the distribution of $P(x_i \mid y)$.
Let us try to apply the above formula manually on our weather dataset. For this, we need to do some precomputations on our dataset.

We need to find $P(x_i \mid y_j)$ for each x_i in X and y_j in y. All these calculations have been demonstrated in the tables below:

Outlook

	Yes	No	P(yes)	P(no)
Sunny	2	3	2/9	3/5
Overcast	4	0	4/9	0/5
Rainy	3	2	3/9	2/5
Total	9	5	100%	100%

Temperature

	Yes	No	P(yes)	P(no)
Hot	2	2	2/9	2/5
Mild	4	2	4/9	2/5
Cool	3	1	3/9	1/5
Total	9	5	100%	100%

Humidity

	Yes	No	P(yes)	P(no)
High	3	4	3/9	4/5
Normal	6	1	6/9	1/5
Total	9	5	100%	100%

Wind

	Yes	No	P(yes)	P(no)
False	6	2	6/9	2/5
True	3	3	3/9	3/5
Total	9	5	100%	100%

Play		P(Yes)/P(No)
Yes	9	9/14
No	5	5/14
Total	14	100%

So, in the figure above, we have calculated $P(x_i \mid y_j)$ for each x_i in X and y_j in y manually in the tables 1-4. For example, probability of playing golf given that the temperature is cool, i.e P(temp. = cool | play golf = Yes) = 3/9.

Also, we need to find class probabilities (P(y)) which has been calculated in the table 5. For example, P(play golf = Yes) = 9/14.

So now, we are done with our pre-computations and the classifier is ready!

Let us test it on a new set of features (let us call it today):

```
today = (Sunny, Hot, Normal, False)
```

So, probability of playing golf is given by:

$$P(\text{yes}|\text{today}) = \frac{P(\text{Sunny outlook}|\text{yes})\,P(\text{Hot temp}|\text{yes})\cdot P(\text{Normal Humidity}|\text{yes})\,P(\text{No wind}|\text{yes})\,P(\text{yes})}{P(\text{today})}$$

and probability to not play golf is given by:

$$P(\text{No}|\text{today}) = \frac{P(\text{Sunny outlook}|\text{No})\,P(\text{Hot temp}|\text{No})\,P(\text{Normal Humidity}|\text{No})\,P(\text{No wind}|\text{No})\,P(\text{No})}{P(\text{today})}$$

Since, P(today) is common in both probabilities, we can ignore P(today) and find proportional probabilities as:

$$P(\text{yes}|\text{today}) \propto \frac{2}{9}\cdot\frac{2}{9}\cdot\frac{6}{9}\cdot\frac{6}{9}\cdot\frac{9}{14} \approx 0.0141$$

and

$$P(\text{No}|\text{today}) \propto \frac{3}{5}\cdot\frac{2}{5}\cdot\frac{1}{5}\cdot\frac{2}{5}\cdot\frac{5}{14} \approx 0.0068$$

Now, since

$$P(\text{yes}|\text{today}) + P(\text{No}|\text{today}) = 1$$

These numbers can be converted into a probability by making the sum equal to 1 (normalization):

$$P(\text{yes} \mid \text{today}) = \frac{0.0141}{0.0141 + 0.0068} = 0.67.$$

and

$$P(\text{No} \mid \text{today}) = \frac{0.0068}{0.0141 + 0.0068} = 0.33$$

Since

$$P(\text{yes} \mid \text{today}) > P(\text{No} \mid \text{today}).$$

So, prediction that golf would be played is 'Yes'.

The method that we discussed above is applicable for discrete data. In case of continuous data, we need to make some assumptions regarding the distribution of values of each feature. The different naive Bayes classifiers differ mainly by the assumptions they make regarding the distribution of $P(x_i \mid y)$.