Data Mining

Classification: Alternative Techniques

Slides by Tan, Steinbach, Kumar adapted by Pimprapai Thainiam

International College

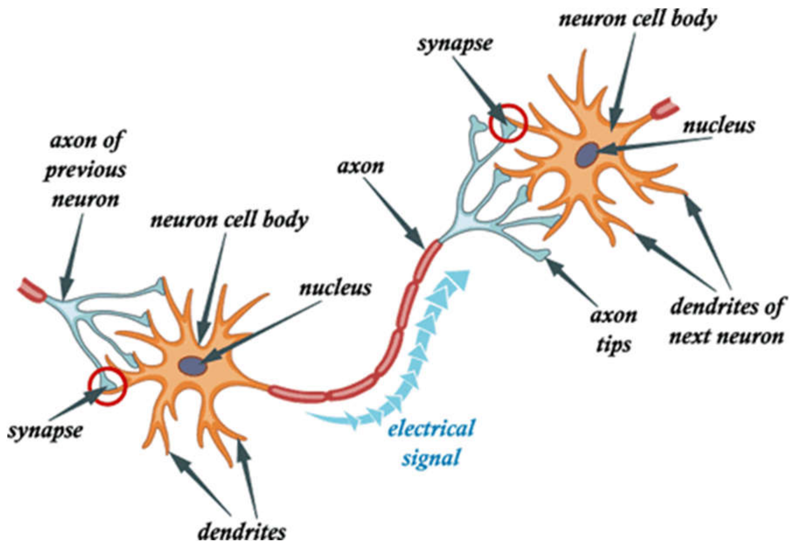
Topics

- Artificial Neural Network
- Support Vector Machine
- Ensemble Methods
- Multiclass Problem

Artificial Neural Network (ANN)

- **Artificial Neural Network (ANN)** was inspired by attempts to simulate biological neural systems.

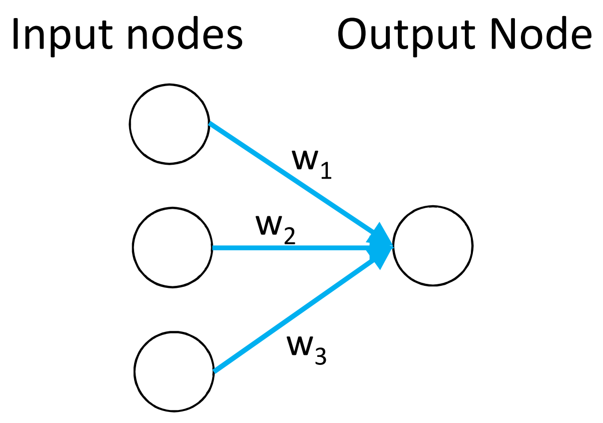
- The human brain consists of
  - **Neurons** are nerve cells.
  - **Axons** are used to link neurons together. They are used to transmit nerve impulses from one neuron to another.
  - **Dendrites** are used to connect neuron to the axons of other neurons.
  - **Synapse** is the contact point between a dendrite and an axon.



Artificial Neural Network (ANN)

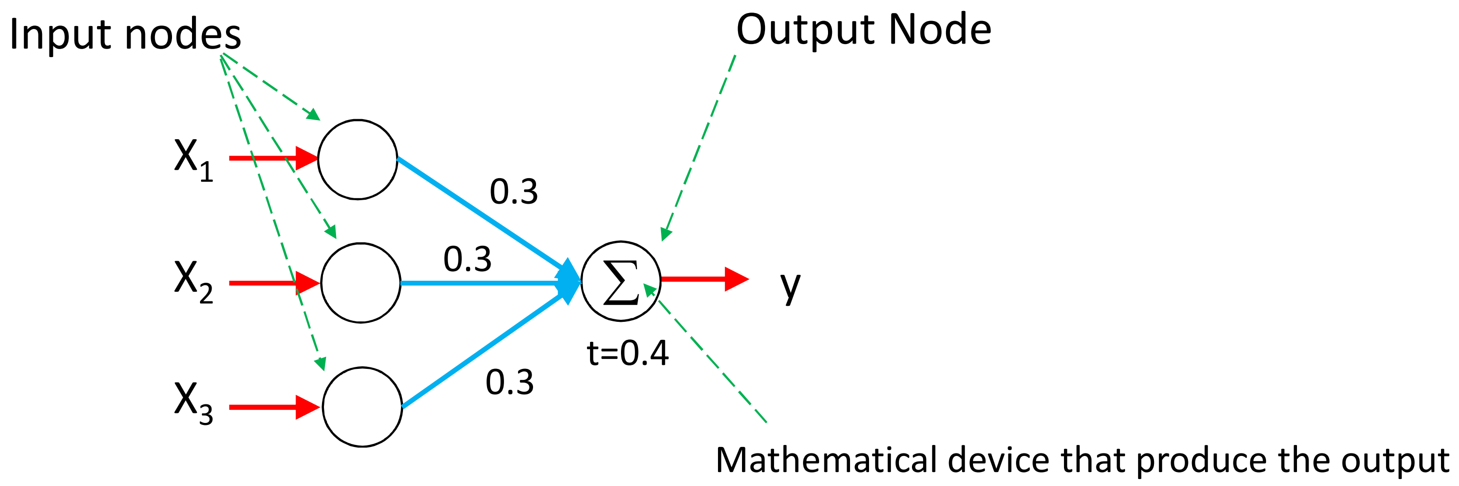
- ANN is composed of an interconnected assembly of nodes and directed links.
- A simplest model of ANN is called **perceptron** consisting of two types of nodes:
  - **Input nodes** are used to represent the input attributes.
  - **Output node** is used to represent the output attribute.
- The nodes in a neural network architecture are known as neurons or units.
- Each input node is connected via a weighted link to the output node.

X_1	X_2	X_3	y
1	0	0	-1
1	0	1	1
1	1	0	1
1	1	1	1
0	0	1	-1
0	1	0	-1
0	1	1	1
0	0	0	-1



Perceptron

- A perceptron computes its output value Y by performing a weighted sum on its input, subtracting a bias factor t from the sum, and then examining the sign of the result.



$$\hat{y} = \begin{cases} 1, & if\ 0.3x_1 + 0.3x_2 + 0.3x_3 - 0.4 > 0; \\ -1, & if\ 0.3x_1 + 0.3x_2 + 0.3x_3 - 0.4 < 0 \end{cases}$$

Perceptron

- The output of a perceptron model can be expressed mathematically as follows:

$$\hat{y} = sign(w_d x_d + w_{d-1} x_{d-1} + \cdots + w_2 x_2 + w_1 x_1 - t)$$

where $w_1, w_2, \ldots, w_d$ are the weights of the input links

$x_1, x_2, \ldots, x_d$ are the input attribute values

- More compact from of perceptron:

dot product

$$\hat{y} = sign[w_d x_d + w_{d-1} x_{d-1} + \cdots + w_1 x_1 + w_0 x_0] = \boxed{sign(\mathbf{w} \cdot \mathbf{x})}$$

where $w_0 = -t$, $x_0 = 1$

$\mathbf{w} \cdot \mathbf{x}$ is the dot product between the weight vector $\mathbf{w}$ and the input attribute vector $\mathbf{x}$

Perceptron

- During the training phase of a perceptron model, the weight parameters $\mathbf{w}$ are repeatedly adjusted until the outputs of the perceptron become consistent with the true outputs of training records.

Perceptron Learning Algorithm

```
1:  Let $D = \{\mathbf{x}_i, y_i \mid i = 1,2,\ldots,N\}$ be the set of training records.
2:  Initialize the weight vector with random values, $\mathbf{w}^{(0)}$ → $\begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix}$
3:  repeat
4:      for each training record $(\mathbf{x}_i, y_i) \in D$ do
5:          Compute the predicted output $\hat{y}_i^{(k)}$
6:          for each weight $w_j$ do
7:              Update the weight, $\boxed{w_j^{(k+1)} = w_j^{(k)} + \lambda\left(y_i - \hat{y}_i^{(k)}\right)x_{ij}}$
8:          end for
9:      end for
10: until stopping criterion is met
```

where $w_j^{(k)}$ is the weight parameter associated with the j^{th} input link after the k^{th} iteration

λ is a parameter known as learning rate

x_{ij} is the value of the j^{th} attribute of the training record $\mathbf{x}_i$

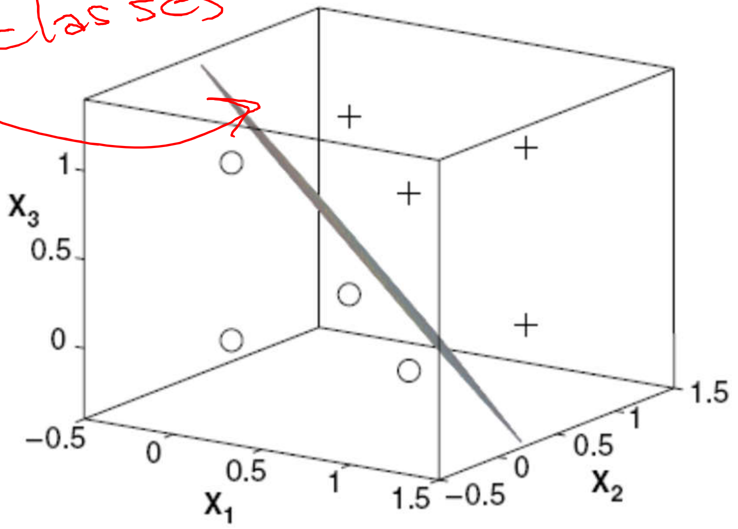
y_i and $\hat{y}_i$ are the actual output and the predicted output after the k^{th} iteration, respectively

Perceptron

- Since the perceptron model $\hat{y} = sign[w_d x_d + w_{d-1} x_{d-1} + \cdots + w_1 x_1 + w_0 x_0]$ is linear in its parameter $\mathbf{w}$ and attributes $\mathbf{x}$, then the **decision boundary** of a perceptron which is obtained by setting $\hat{y} = 0$, is a linear hyperplane that separates the data into two classes.

2 classes

X_1	X_2	X_3	y
1	0	0	-1
1	0	1	1
1	1	0	1
1	1	1	1
0	0	1	-1
0	1	0	-1
0	1	1	1
0	0	0	-1



Note: A hyperplane is a subspace of one dimension less than its ambient space. If a space is 3-dimensional then its hyperplanes are the 2-dimensional planes, while if the space is 2-dimensional, its hyperplanes are the 1-dimensional lines.

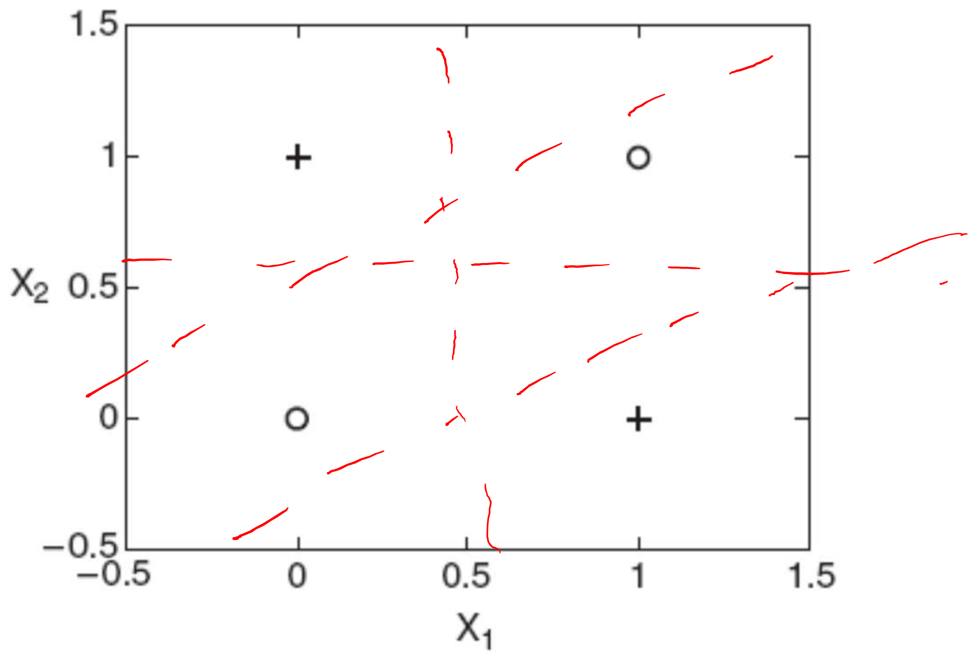
International College

Perceptron

- If the classification problem looks like as follows:

XOR problem

X_1	X_2	y
0	0	−1
1	0	1
0	1	1
1	1	−1



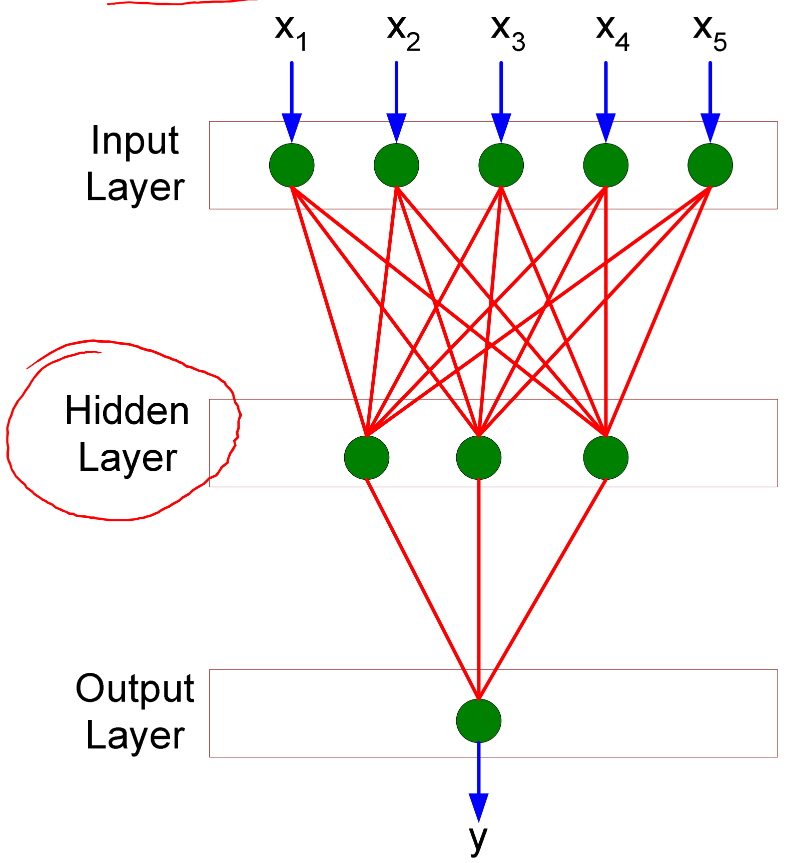
Can we separate this training set using only one hyperplane (line)?

Multilayer Artificial Neural Network

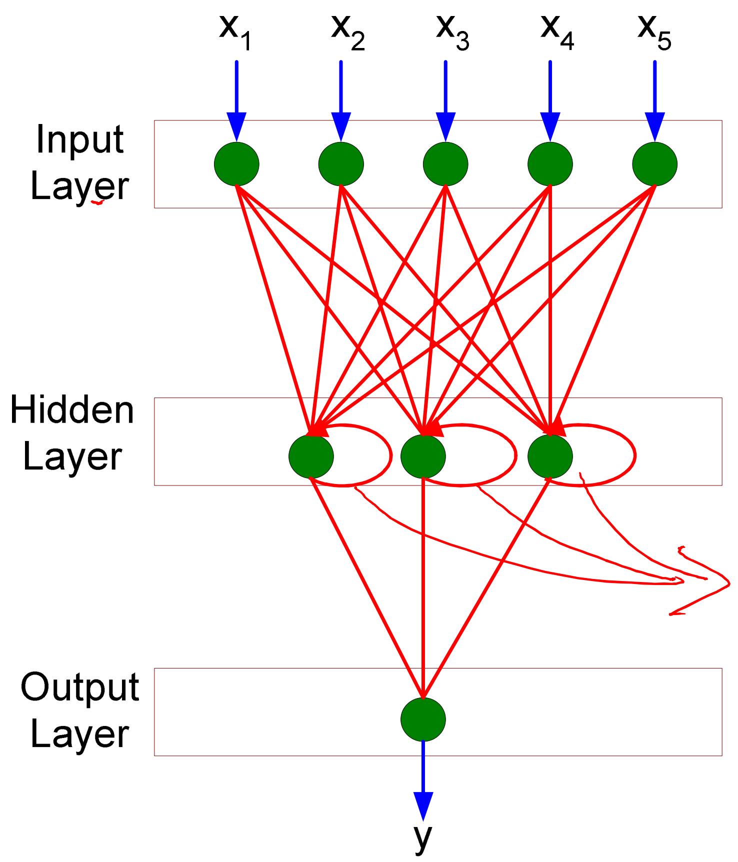
- An artificial neural network has a more complex structure than that of a perceptron model where
  - It may contain several intermediary layers called **hidden layers** and the nodes embedded in these layers are called **hidden nodes** $\Rightarrow$ **a multilayer artificial neural network**.
    - A **feed-forward** neural network: The nodes in one layer are connected only to the nodes in the next layer.
    - A **recurrent** neural network: The links may connect nodes within the same layer or nodes from one layer to the previous layers.
  - The activation function could be other functions such as linear, sigmoid, and hyperbolic tangent function other than the sign function.

Multilayer Artificial Neural Network

A feed-forward artificial neural network (ANN)



A recurrent artificial neural network (ANN)



International College

Characteristics of Artificial Neural Network

- It is important to choose the appropriate network topology (feed-forward, recurrent) for a given problem to avoid model overfitting.
- ANN can handle redundant features.
- ANN is quite sensitive to the presence of noise.
- Training an ANN is a time consuming process, especially when the number of hidden nodes is large.

Topics

- Artificial Neural Network
- **Support Vector Machine**
- Ensemble Methods
- Multiclass Problem

Support Vector Machines

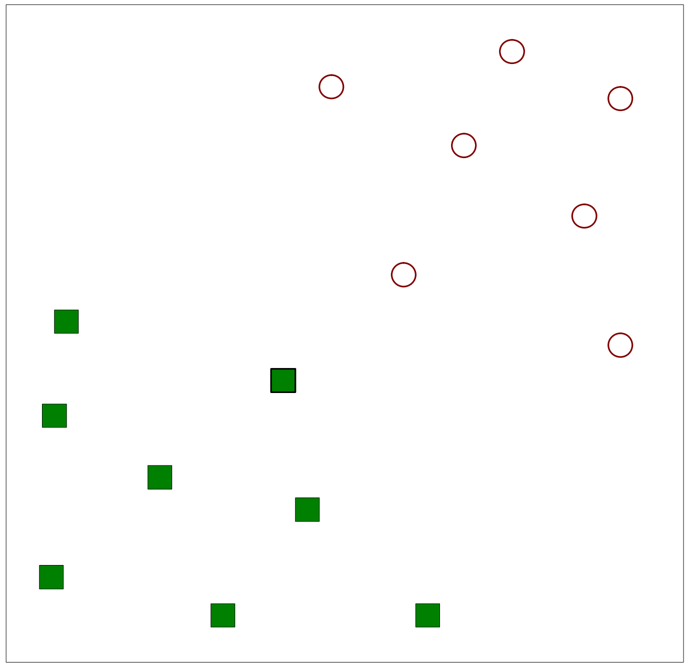
- **Support Vector Machines (SVM)** is a classification technique that associates with finding a linear hyperplane (decision boundary) that will separate the data.

- It works very well with high-dimensional data.

- It represents a **decision boundary** using a subset of the training records known as **support vectors**.

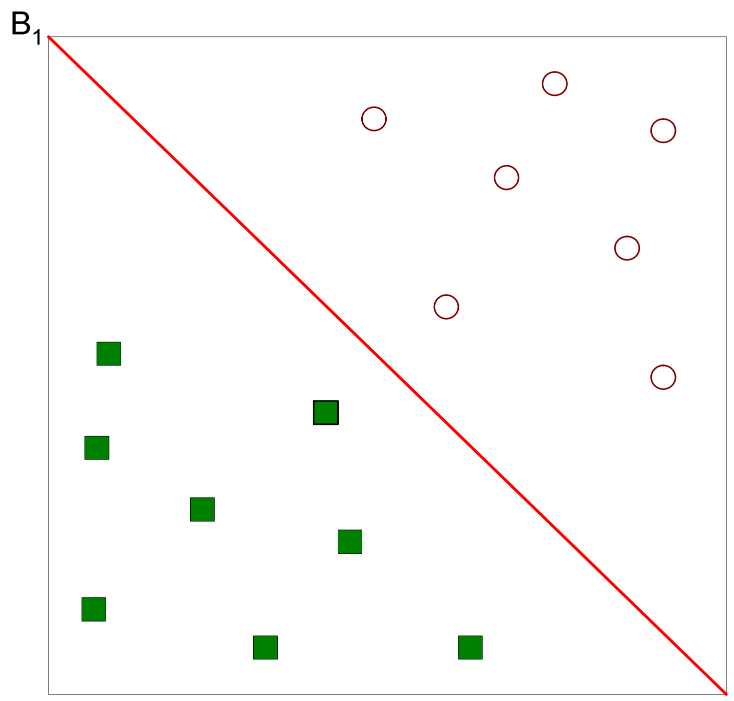
Support Vector Machines

Find a linear hyperplane (decision boundary) that will separate the data



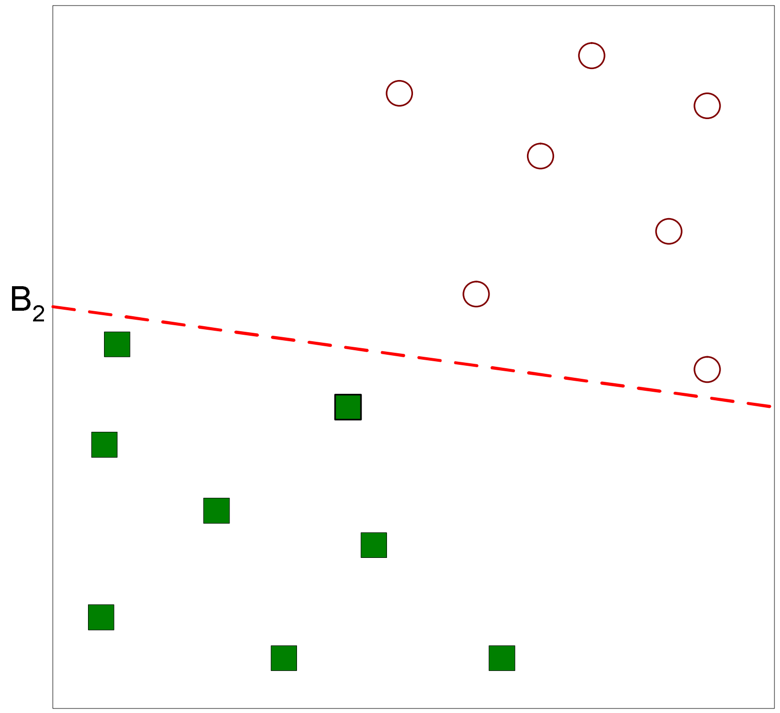
Support Vector Machines

One possible solution



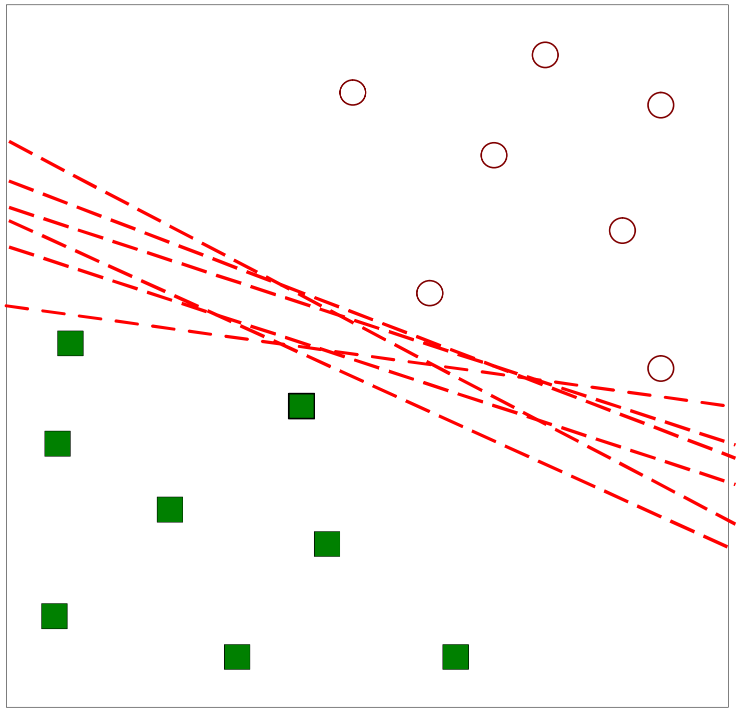
Support Vector Machines

Another possible solution



Support Vector Machines

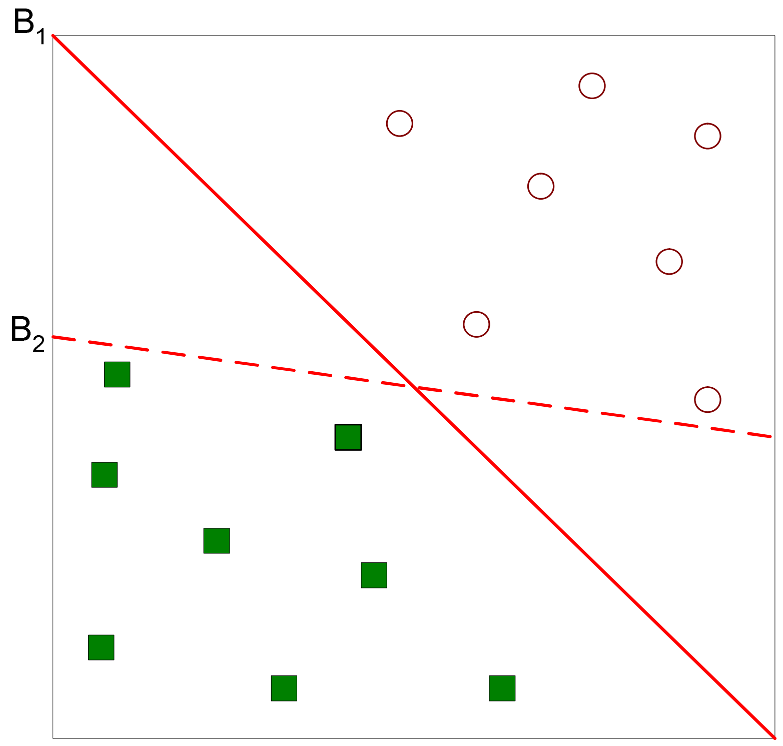
Other possible solutions



Which solution should we select?

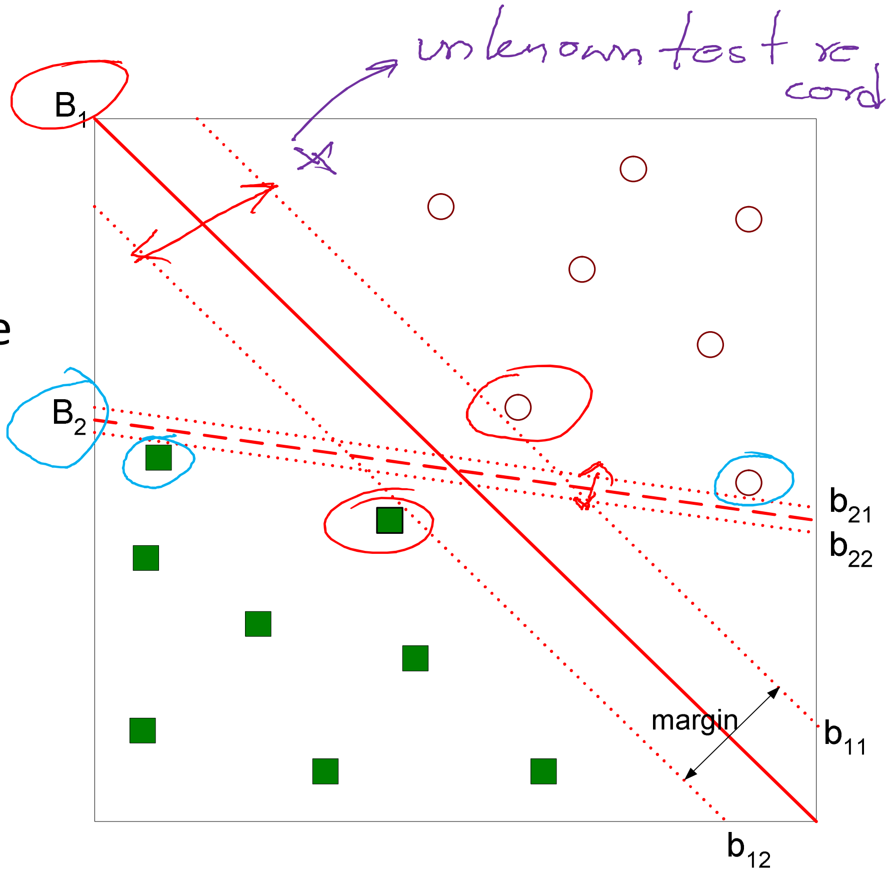
Support Vector Machines

- If we have two hyperplanes (decision boundaries), B_1 or B_2 , which one is better B_1 or B_2?
- How do you define better?



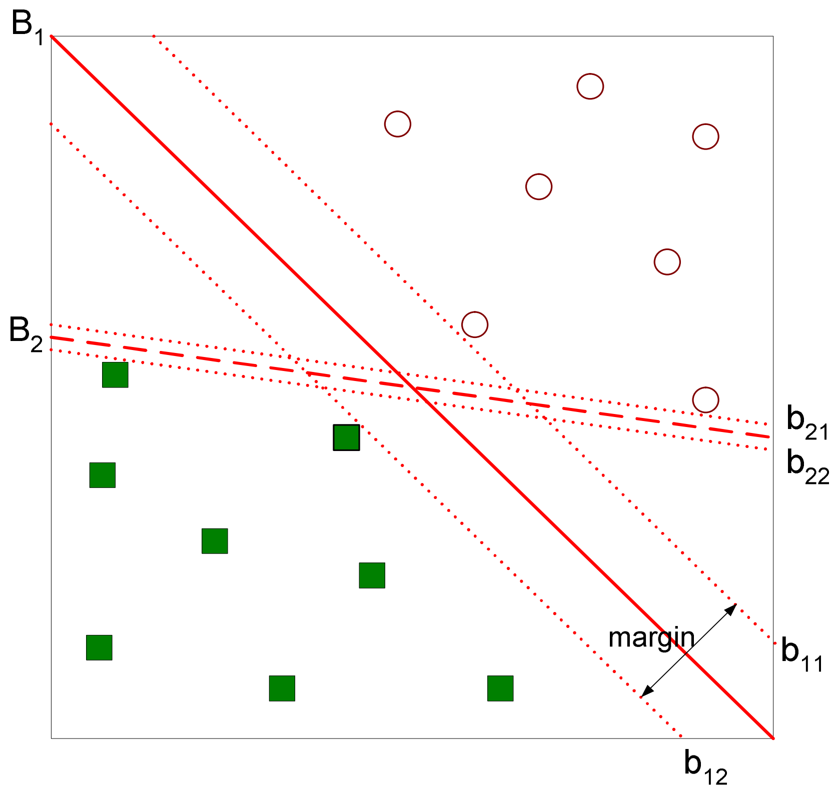
Support Vector Machines

- The classifier must choose one hyperplane to represent its decision boundary, based on how well they are expected to perform on test records.
- Both B_1 and B_2 are hyperplanes that can separate the training records into their class correctly.
- The other hyperplanes, b_{i1} and b_{i2}, are obtained by moving a parallel hyperplane away from the decision boundary until it touches the closet square and the closet circle, respectively.
- The distance between b_{i1} and b_{i2} is known as **the margin of the classifier**.



Support Vector Machines

- The best decision boundary should be the one with highest margin; in this case, B_1 is better than B_2 because it provides maximum margin.
- For any training set, we should find **the maximum margin hyperplane** to be a decision boundary.
- This is because decision boundaries with large margins tend to have better generalization errors than those with small margins.
- Classifiers that produce decision boundaries with small margins are more susceptible to model overfitting and tend to generalize poorly on previously unseen records.



Topics

- Artificial Neural Network
- Support Vector Machine
- **Ensemble Methods**
- Multiclass Problem

Ensemble Methods

- The classification techniques except the nearest-neighbor method, predict the class labels of unknown records using a single classifier induced from training data.
- Techniques for improving classification stability and accuracy by aggregating the prediction of multiple classifiers are known as **ensemble** or **classifier combination** methods.
- An ensemble method constructs a set of base classifiers from training data and performs classification by taking a vote on the prediction made by each base classifiers.
- This kind of method can be used to reduce variance in the prediction and reduce overfitting.

Ensemble Methods

The ensemble of classifiers can be constructed in four ways:

1. **By manipulating the training set:** The multiple training sets are created by resampling the original data according to some sampling distribution. (e.g. **Bagging** and **Boosting**)
2. **By manipulating the input attributes:** A subset of input attributes is chosen to form each training set. This approach works very well with data sets that contain highly redundant attributes. (e.g. **Random Forests**)
3. **By manipulating the class labels:** This method can be used when the number of classes is sufficiently large. An ensemble of base classifiers is created by repeatedly train a binary training data obtained from relabeling the original training data. (e.g. **Error-correcting output coding method**)

Ensemble Methods

4. **By manipulating the learn algorithm:** Many learning algorithms can be manipulated in such a way that applying the algorithm several times on the same training data may result in different models. For example, an ensemble of decision trees, instead of choosing the best splitting attribute at each node, we can randomly choose one of the top k attributes for splitting.

Note: The first three approaches are generic methods that are applicable to any classifiers, whereas the fourth approach depends on the type of classifier used.

General Procedure for Ensemble Methods

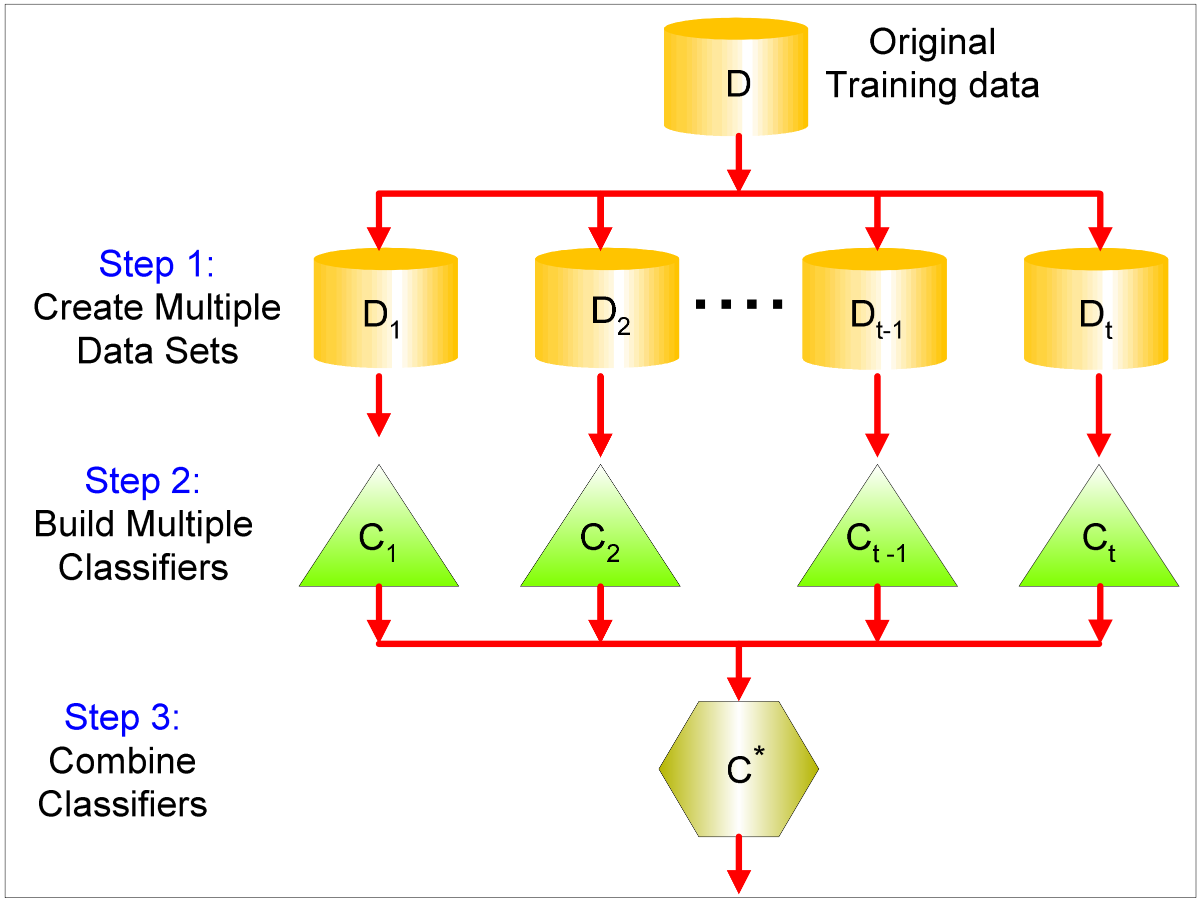
Steps to build an ensemble classifier:

1. Create a training set D_i from the original training set D according to the type of ensemble method used.
2. Build a base classifier C_i from each training set D_i
3. A test record $\mathbf{x}$ is classified by combining the predictions made by the base classifier $C_i(\mathbf{x})$:

$$C^*(\mathbf{x}) = Vote(C_1(\mathbf{x}), C_2(\mathbf{x}), \ldots, C_k(\mathbf{x}))$$

The class can be obtained by taking a majority vote on the individual predictions or by weighting each prediction with the accuracy of the basic classifier.

Ensemble Methods



D
Original Training data
Step 1: Create Multiple Data Sets
D_1
D_2
....
D_{t-1}
D_t
Step 2: Build Multiple Classifiers
C_1
C_2
C_{t-1}
C_t
Step 3: Combine Classifiers
C^*

International College

Bagging (Bootstrap Aggregation) Method

- **Bagging** repeatedly samples (with replacement) from a data set according to uniform probability distribution (all data records have the same selecting probability).
- Each bootstrap sample has the same size as the original data.
- Example:

Original Data	1	2	3	4	5	6	7	8	9	10
Bagging (Round 1)	7	8	10	8	2	5	10	10	5	9
Bagging (Round 2)	1	4	9	1	2	3	2	7	3	2
Bagging (Round 3)	1	8	5	10	5	5	9	6	3	7

Boosting

- **Boosting** is an iterative procedure used to adaptively change the distribution of training data so that the base classifiers will focus on records that are hard to classify (previously misclassified records).
- The selecting weight of each training record will be recalculated for each round of creating a training set where $\sum_{i=1}^{N} w_i = 1$
  - For the first round, all N records are assigned equal weights, $\frac{1}{N}$, so that they are equally likely to be chosen for training.
  - For the subsequent round, the weights of the training records are updated according to the test results on the original training data obtained from the previous round.
    - Records that are wrongly classified will have their weights increased
    - Records that are classified correctly will have their weights decreased

Boosting

- Example: Suppose the training record number 4 is hard to classify, then the weight for this record will be increased in order to increase its opportunity to be chosen in subsequent rounds .

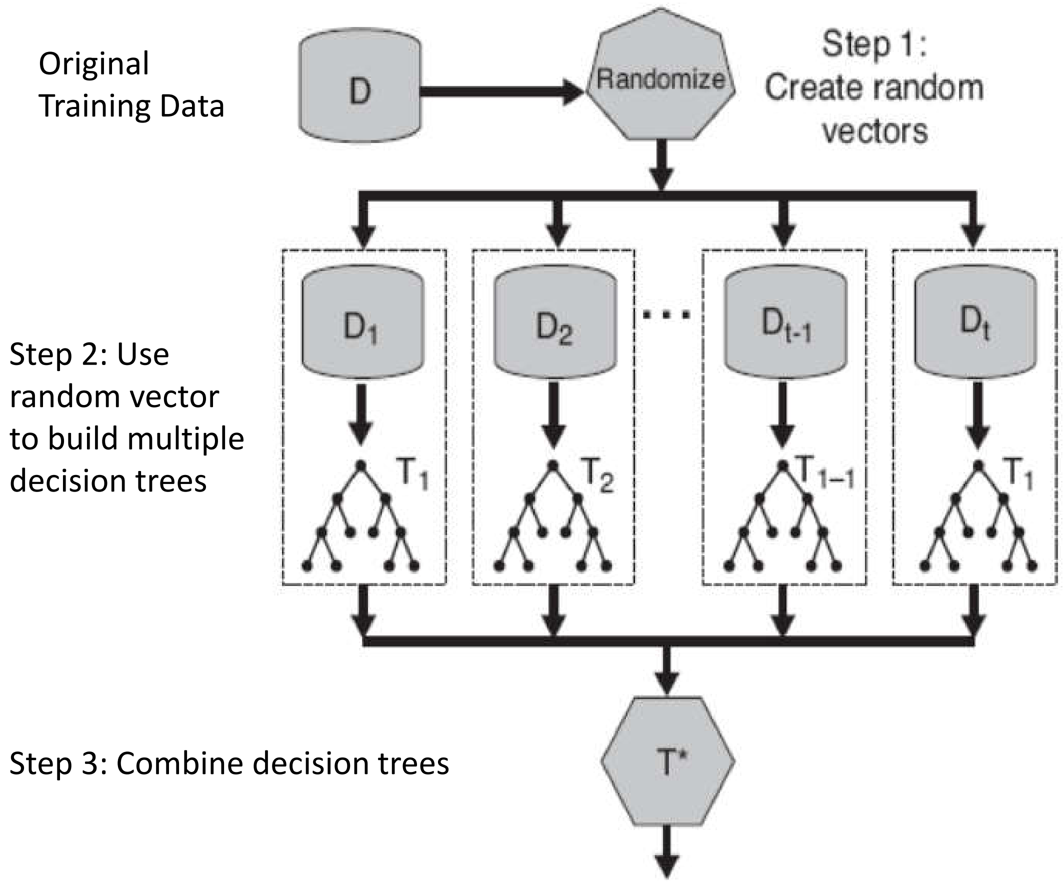
Original Data	1	2	3	4	5	6	7	8	9	10
Boosting (Round 1)	7	3	2	8	7	9	4	10	6	3
Boosting (Round 2)	5	4	9	4	2	5	1	7	4	2
Boosting (Round 3)	4	4	8	10	4	5	4	6	3	4

- Popular algorithm: **AdaBoost** (Adaptive Boosting)

Random Forests

- **Random forests** combines the prediction made by multiple decision trees where each tree is generated based on the values of an independent set of random vectors (attributes).

- The **random vectors** are generated from a fixed probability distribution.



Original Training Data
D
Randomize
Step 1: Create random vectors
Step 2: Use random vector to build multiple decision trees
D_1
D_2
...
D_{t-1}
D_t
T_1
T_2
T_{1-1}
T_1
Step 3: Combine decision trees
T^*

Random Forests

Three approaches for random vectors:

1. **Forest-RI:** It randomly selects F input attributes to split each node of the decision tree, then the decision to split a node is determined from these selected F attributes. We can increase randomness by implementing bagging scheme.

   - If F is small, then the trees tend to become less correlated and less strength.

   - If F is large, then the trees tend to become more correlated and more strength.

   The number of attributes should be $F = \log_2 d + 1$ where d is the total number of attributes.

2. **Forest-RC:** It creates linear combinations of the input attributes. At each node, F new attributes are generated, and the best of them is subsequently selected to split the node. A new attribute is generated by randomly selecting L of the input attributes, then they are linearly combines using coefficients generated from a uniform distribution in the range of [-1,1].

Random Forests

3. This approach randomly select one of the F best splits at each node of the decision tree. It still needs to examine all the splitting attributes at each node of the decision tree, thus it does not save runtime.

Note: The classification accuracies of random forests are quite comparable to the AdaBoost algorithm. It is also more robust to noise and runs much faster than the AdaBoost algorithm.

Error-Correcting Output Coding

- At each round of creating a training set and building a base classifier:
  - Transform the original training data into a binary class problem by randomly partitioning the class labels into two disjoint subsets, A_0 and A_1.
  - Relabel all records in A_0 and A_1 to class 0 and class 1, respectively.
  - Use this relabeled training set to train a base classifier
  - Predict class of a test record by taking the majority votes obtained from applying each base classifier C_i.
    - If the test record is predicted as class 0, then all the classes that belong to A_0 will receive a vote.
    - If the test record is predicted as class 1, then all the classes that belong to A_1 will receive a vote.

Other Popular Classification Approaches

- **Logistic Regression:** It is a regression model where the dependent variable (DV) is categorical variable, so, it can be used as a classification method.

- **Linear Discriminant Analysis:** It is used to find a linear combination of attributes that separates two or more classes of objects. The resulting combination may be used as a linear classifier, or, more commonly, for dimensionality reduction before later classification.

Topics

- Artificial Neural Network
- Support Vector Machine
- Ensemble Methods
- **Multiclass Problem**

International College

Multiclass Problem

- Some of the classification techniques, such as support vector machines and AdaBoost, are originally designed for binary classification problems.
- Some of the classification problems are multiclass problems.
- Let $Y = \{y_1, y_2, \dots, y_K\}$ is the set of classes of the data.
- **Three approaches** for extending the binary classifiers to handle multiclass problems:
  1. **One-against-rest (1-r) approach:** It decomposes the multiclass problem into K binary problems.
     - Create a binary problem for each class $y_i \in Y$ by relabeling all records that belong to y_i as positive examples, and relabeling the remaining records as negative examples.
     - Construct a binary classifier to separate positive examples (class y_i) from negative examples.
     - Apply all K classifiers to a test record, and use a voting scheme to combine the predictions. If a test record is classified as negative, then all classes except for the positive class receive a vote.

Multiclass Problem

- **Example:** $Y = \{y_1, y_2, y_3, y_4\}$ Suppose a test record is classified as {+,-,-,-} from 4 classifiers.

Result	$\mathbf{y_1}$	$\mathbf{y_2}$	$\mathbf{y_3}$	$\mathbf{y_4}$
+	1	0	0	0
-	1	0	1	1
-	1	1	0	1
-	1	1	1	0
Total	4	2	2	2

This test record is classified as class y_1.

Multiclass Problem

2. One-against-one (1-1) approach: It constructs $\frac{K(K-1)}{2}$ binary classifiers, where each classifier is used to distinguish between a pair of classes, (y_i, y_j).

- Create a binary problem for each pair of classes (y_i, y_j) by ignoring records that do not belong to either y_i or y_j.
- Construct a binary classifier to separate y_i and y_j.
- Apply all $\frac{K(K-1)}{2}$ classifiers to a test record, then use a voting scheme to combine the predictions.

Multiclass Problem

- **Example:** $Y = \{y_1, y_2, y_3, y_4\}$ Suppose a test record is classified as follows:

Binary pair of classes	$+: y_1$ $-: y_2$	$+: y_1$ $-: y_3$	$+: y_1$ $-: y_4$	$+: y_2$ $-: y_3$	$+: y_2$ $-: y_4$	$+: y_3$ $-: y_4$
Classification Results	+	+	-	-	-	-

Result	y_1	y_2	y_3	y_4
+	1	0	-	-
+	1	-	0	-
-	0	-	-	1
-	-	0	1	-
-	-	0	-	1
-	-	-	0	1
Total	**2**	**0**	**1**	**3**

This test record is classified as class y_4.

International College

Multiclass Problem

3. Error-correcting output coding (ECOC): The first two approaches are sensitive to the binary classification errors, but this method provides a more robust way for handling multiclass problems.

- Each class y_i is represented by a unique bit string of length n known as its codeword. Codeword for each class could be obtained by using coding theory.
- Train n binary classifiers to predict each bit of the codeword string.
- Produce the codeword for a test record by applying n binary classifiers to classify each bit string of the codeword.
- Predict class of a test record by assigning a test record to the class that has the lowest Hamming distance.

Note: The Hamming distance between two strings of equal length is the number of positions at which the corresponding symbols are different.

Multiclass Problem

- **Example:** $Y = \{y_1, y_2, y_3, y_4\}$ Suppose we encode the classes using the following 7-bits codewords:

Class	Codeword							Hamming Distance
$\mathbf{y_1}$	1	1	1	1	1	1	1	1
$\mathbf{y_2}$	0	0	0	0	1	1	1	3
$\mathbf{y_3}$	0	0	1	1	0	0	1	3
$\mathbf{y_4}$	0	1	0	1	0	1	0	3
Result	0	1	1	1	1	1	1	

This test record is classified as class y_1.