

Chapter-3: Classification

- Classification is a data mining technique used to predict group membership of data instances.
- Classification assigns items on a collection to target categories or classes.
- The goal of classification is to accurately predict the target class for each case in the data.



Stages in classification

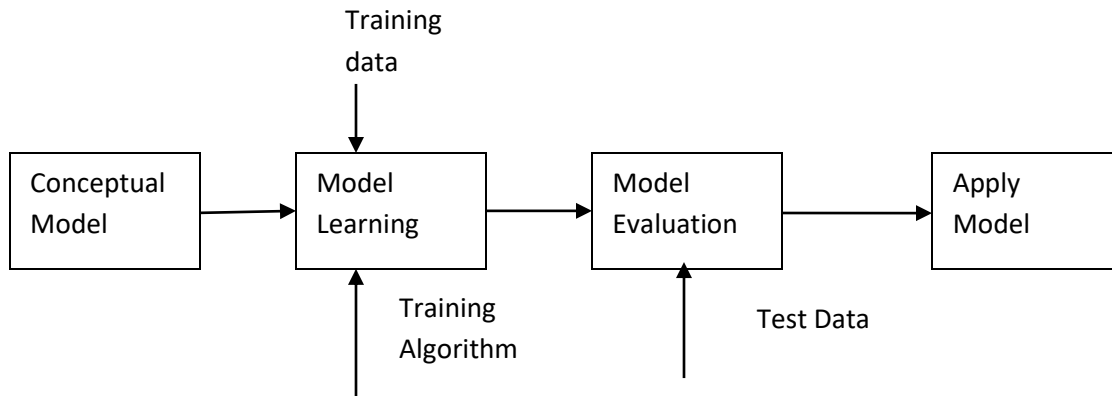


Fig: Stages in classification

Types:

- Decision Tree classifier
- Rule Based Classifier
- Nearest Neighbor Classifier
- Bayesian Classifier
- Artificial Neural Network (ANN) Classifier
- Others

Decision Tree classifier

- A decision tree is tree in which each branch node represents a choice between a number of alternatives and each leaf node represents a classification or decision.

- Decision tree is a classifier in the form of a tree structure where a **leaf node** indicates the class of instances, a **decision node** specifies some test to be carried out on a single attribute value with one branch and sub-tree for each possible outcome of the test.
- A decision tree can be used to classify an instance by starting at root of the tree and moving through it until leaf node. The leaf node provides the corresponding class of instance.



Decision Tree Algorithm

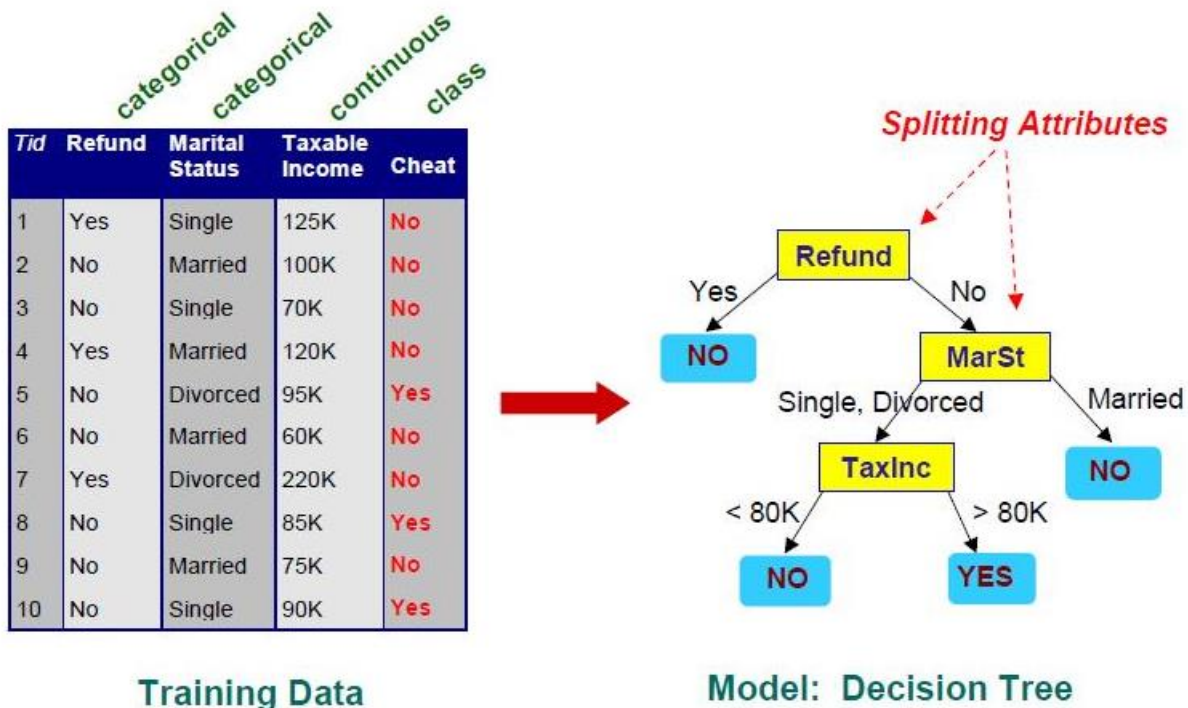
- Hunt's Algorithm
- ID3, J48, C4.5 (Based on Entropy Calculation)
- SLIQ,SPRINT,CART (Based on Gini-Index)

Hunt's Algorithm

- Hunt's algorithm grows a decision tree in a recursive fashion by partitioning the training data into successively into subsets.
- Let D_t be the set of training data that reach a node ' t '. The general recursive procedure is defined as:
 - If D_t contains records that belong the same class y_t , then t is a leaf node labeled as y_t .
 - If D_t is an empty set, then t is a leaf node labeled by the default class, y_d
 - If D_t contains records that belong to more than one class, use an attribute test to split the data into smaller subsets.
- It recursively applies the procedure to each subset until all the records in the subset belong to the same class.
- The Hunt's algorithm assumes that each combination of attribute sets has a unique class label during the procedure.
- If all the records associated with D_t have identical attribute values except for the class label, then it is not possible to split these records any future. In this case, the node is declared a leaf

node with the same class label as the majority class of training records associated with this node.

Eg:



Tree Induction:

Tree induction is based on Greedy Strategy i.e. split the records based on an attribute test that optimize certain criterion.

Issues:

1. **How to split the record?**
2. **How to specify the attribute test condition?**
 - Depends on attribute types and number of ways to split the record i.e. 2-ways split /multi-way split.
 - Depends upon attribute types. (Nominal, Ordinal, Continuous)
3. **When to stop splitting?**
 - When all records are belongs to the same class or all records have similar attributes.
4. **How to determine the best split?**
 - Nodes with homogenous class distribution are preferred.
 - Measure the node impurity.
 - i. Gini-Index
 - ii. Entropy
 - iii. Misclassification Error

Gini-Index

- The Gini Index measures the impurity of data set (D) as:

- $$\text{Gini}(D) = 1 - \sum_1^n p_i^2$$

Where,

n = Number of classes, p_i = Probability of i^{th} class.

- It consider binary split for each attribute.
- When D is partition into D_1 and D_2 then
$$\text{Gini}(D) = D_1/D \text{ Gini}(D_1) + D_2/D \text{ Gini}(D_2)$$
- The attribute that maximize the reduction in impurity is selected as splitting attribute.

Eg:

Refer Class Note

ID3 Algorithm

- The ID3 algorithm begins with the original dataset as the root node.
- On each iteration of the algorithm, it iterates through every unused attribute of the dataset and calculates the entropy (or information gain) of that attribute.
- It then selects the attribute which has the smallest entropy (or largest information gain) value.
- The dataset is then split by the selected attribute to produce subsets of the data.
- The algorithm continues to recur on each subset, considering only attributes never selected before.

Recursion on a subset may stop in one of these cases:

Algorithm

- Every element in the subset belongs to the same class , then the node is turned into a leaf and labeled with the class of the examples
- If the examples do not belong to the same class ,
 - Calculate entropy and hence information gain to select the best node to split data.
 - Partition the data into subset.
- Recursively repeat until all data are correctly classified

Throughout the algorithm, the decision tree is constructed with each non-terminal node representing the selected attribute on which the data was split, and terminal nodes representing the class label of the final subset of this branch.

Entropy

Entropy $H(S)$ is a measure of the amount of uncertainty in the dataset (S) (i.e. entropy characterizes the dataset (S)).

$$H(S) = - \sum_{x \in X} p(x) \log_2 p(x)$$

Where,

- S The current dataset for which entropy is being calculated (changes every iteration of the ID3 algorithm)
 - X - Set of classes in S
 - $P(x)$ - The probability of each set S
- When $H(S) = 0$, the set S is perfectly classified (i.e. all elements in S are of the same class).
 - In ID3, entropy is calculated for each remaining attribute. The attribute with the smallest entropy is used to split the set S on this iteration.
 - The higher the entropy, the higher the potential to improve the classification here.

Information Gain

Information gain is the measure of the difference in entropy from before to after the set S is split on an attribute A. In other words, how much uncertainty in dataset (S) was reduced after splitting dataset S on attribute A.

$$IG(A, S) = H(S) - \sum_{t \in T} p(t) H(t)$$

Where,

- $H(S)$ - Entropy of dataset S
- T - The subsets created from splitting dataset S by attribute A.
- $P(t)$ - The probability of class t
- $H(t)$ - Entropy of subset t

In ID3, information gain can be calculated (instead of entropy) for each remaining attribute. The attribute with the largest information gain is used to split the set S on this iteration.

Tree Pruning

- Tree Pruning is performed in order to remove anomalies in training data due to noise or outliers.
- The pruned trees are smaller and less complex.

- Pruning is a technique in machine learning that reduces the size of decision trees by removing sections of the tree that provide little power to classify instances.
- The dual goal of pruning is reduced complexity of the final classifier as well as better predictive accuracy by the reduction of overfitting and removal of sections of a classifier that may be based on noisy or erroneous data.
- One of the questions that arise in a decision tree algorithm is the optimal size of the final tree.
- A tree that is too large risks overfitting the training data and poorly generalizing to new samples.
- A small tree might not capture important structural information about the sample space.
- It is hard to tell when a tree algorithm should stop because it is impossible to tell if the addition of a single extra node will dramatically decrease error.
- A common strategy is to grow the tree until each node contains a small number of instances then use pruning to remove nodes that do not provide additional information.
- Pruning should reduce the size of a learning tree without reducing predictive accuracy as measured by a test set or using cross-validation.
- There are many techniques for tree pruning that differ in the measurement that is used to optimize performance.

Tree pruning approaches

- i. Prepruning - The tree is pruned by halting its construction early.
- ii. Postpruning - This approach removes subtree from fully grown tree.

Pre-pruning

- Based on statistical significance test.
- Stop growing the tree when there is no statistically significant association between any attribute and the class at a particular node
- Most popular test: chi-squared test
- ID3 used chi-squared test in addition to information gain.
- Only statistically significant attributes were allowed to be selected by information gain procedure.
- Pre-pruning may stop the growth process prematurely: early stopping
- Pre-pruning faster than post-pruning

Post-pruning

- First, build full tree then, prune it.
- Fully-grown tree shows all attribute interactions
- Problem: some subtrees might be due to chance effects
- Two pruning operations:

- Subtree replacement
- Subtree raising
- Possible strategies:
 - error estimation
 - significance testing
 - MDL principle
- Subtree replacement selects a subtree and replaces it with a single leaf.
- Subtree raising selects a subtree and replaces it with the child one ie, a "sub-subtree" (replaces its parent)

Advantages of Decision Tree Classifier

- Inexpensive to construct
- Extremely fast at classifying unknown records
- Easy to interpret for small-sized trees
- Accuracy is comparable to other classification techniques for many simple data sets

Rule-Based Classifier

- It classifies records by using a collection of “If Then.....” rules. A rule base classifier uses a set of “If Then.....” rules for classification.

eg: If age = youth AND student = yes THEN buys_computer = yes.

- The ‘If’ part or left hand side of a rule is known as the rule antecedent or precondition where as the ‘Then’ part or right hand side is the rule consequent.
- In the rule antecedent, the condition consists of one or more attribute tests.
- If the condition in a rule antecedent holds true for a given tuple, the rule antecedent is satisfied and that the rule covers the tuple.
- Coverage of a rule is the fraction of records that satisfy the antecedent of a rule.

$$\text{Coverage} = N_{\text{covers}} / D$$

Where,

N_{covers} = number of record that can be classified by the rule.

D = total data set.

- Accuracy of a rule is fraction of records that satisfy both the antecedent and consequent of a rule.

$$\text{Accuracy} = N_{\text{correct}} / N_{\text{covers}}$$

Where,

N_{correct} = Number of records that are correctly classified by the rule

N_{covers} = Number of record that can be classified by the rule

How does Rule-Based Classifier work?

- If a rule is satisfied by a tuple, the rule is said to be triggered. Triggering doesn't always mean firing because there may be more than one rules that can be satisfied.
- Three different cases occur for classification.

Case-I: If only one rule is satisfied

- When any instances is covered by only one rule then the rule fires by returning the class prediction for the tuple defined by the rule.

Case-II: If more than one rules are satisfied

- If more than one rules are triggered, we need a conflict resolution strategy to find which rule is fired.
- Rule ordering or rule ranking or rule priority can be set in case of rules conflict. A rule ordering may be class-based or rule-based.
- Rule-based ordering: Individual rules are ranked based on their quality.
- Class-based ordering: Rules that belong to the same class appear together
- When rule-based ordering is used, the rule set is known as a decision list.

Case-III: If no rule is satisfied

- If any instance not triggered by any rule, use default class for classification. Mostly most frequent class is assigned as default class.

Eg:

| S.No. | Name | Blood Type | Give Birth | Can fly | Live in water | Class |
|-------|--------|------------|------------|---------|---------------|-------|
| 1 | Lemur | Warm | Yes | No | No | ? |
| 2 | Turtle | Cold | No | No | Sometimes | ? |
| 3 | Shark | Cold | Yes | No | Yes | ? |

Rule base

- R1: (Give Birth = No) ^ (Can fly = Yes) => Birds
R2: (Give Birth = No) ^ (Live in Water = Yes) => Fishes
R3: (Give Birth = Yes) ^ (Blood Type = Warm) => Mammals
R4: (Give Birth = No) ^ (Can fly = No) => Reptiles
R5: (Live in Water = Sometimes) => Amphibians

- In above example, R1 and R2 don't have any coverage. R3, R4 & R5 have coverage.
- Instance 1 is triggered by R3, instance 2 is triggered by R4 & R5 and instance 3 is not triggered by any instances.
- Since instance 1 is triggered by only one rule (R3) so it is fired as a class mammal, instance 2 is triggered by more than two rules (R4 & R5) and hence conflict occurs. To resolve the conflict the class can be identified using priority (rule priority or class priority). Instance 3 is not triggered by any rules, to resolve this conflict default class can be used.

Characteristics of Rule-Based Classifier

1. Mutually exclusive Rules

- Classifier contains mutually exclusive rules if all the rules are independent of each other.
- Every record is covered by at most one rule.
- Rules are no longer mutually exclusive if a record may triggered by more than one rule. To make mutually exclusive we apply rule ordering.

2. Exhaustive Rules

- Classifier has exhaustive coverage if it accounts for every possible combination of attribute values (every possible rule).
- Each record is covered by at least one rule.
- Rules are no longer exhaustive if a record may bit trigger any rules. To make rules exhaustive use default class.

Building Classification Rules

- Two approaches are used to build classification rules.

A. Direct Method

- Extract rules directly from data. It is an inductive and sequential approach.

Sequential Covering

1. Start from an empty rule
2. Grow a rule using the Learn-One-Rule function
3. Remove training records covered by the rule
4. Repeat Step (2) and (3) until stopping criterion is met

Aspects of Sequential Covering

- Rule Growing
- Instance Elimination
- Rule Evaluation
- Stopping Criterion
- Rule Pruning

i. Rule Growing

CN2 Algorithm:

- Start from an empty conjunct: $\{\}$
- Add conjuncts that minimizes the entropy measure: $\{A\}, \{A,B\}, \dots$
- Determine the rule consequent by taking majority class of instances covered by the rule

RIPPER Algorithm:

- Start from an empty rule: $\{\} \Rightarrow \text{class}$
- Add conjuncts that maximize FOIL's information gain measure:
 $R_0: \{\} \Rightarrow \text{class}$ (initial rule)
 $R_1: \{A\} \Rightarrow \text{class}$ (rule after adding conjunct)
 $\text{Gain}(R_0, R_1) = t [\log(p_1/(p_1+n_1)) - \log(p_0/(p_0+n_0))]$
 Where, t : number of positive instances covered by both R_0 and R_1
 p_0 : number of positive instances covered by R_0
 n_0 : number of negative instances covered by R_0
 p_1 : number of positive instances covered by R_1
 n_1 : number of negative instances covered by R_1

ii. Instance Elimination

- We need to eliminate instances otherwise, the next rule is identical to previous rule.
- We remove positive instances to ensure that the next rule is different.
- We remove negative instances to prevent underestimating accuracy of rule

iii. Rule Evaluation

Metrics:

$$\text{Accuracy} = \frac{n_c}{n}$$

$$\text{Laplace} = \frac{n_c + 1}{n + k}$$

$$\text{M-estimate} = \frac{n_c + kp}{n + k}$$

n : Number of instances

n_c : Number of instances covered by rule

k : Number of classes

p : Prior probability

iv. Stopping Criterion and Rule Pruning

Stopping criterion

- Compute the gain
- If gain is not significant, discard the new rule.

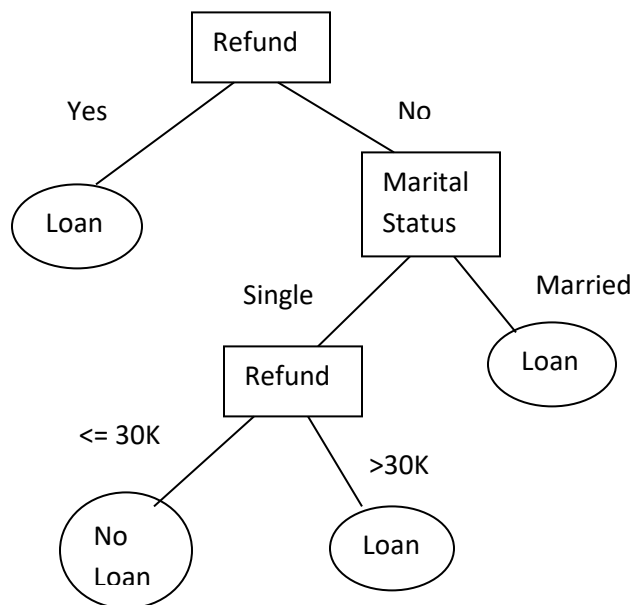
Rule Pruning

- Similar to post-pruning of decision trees.
- Reduced Error Pruning:
 - Remove one of the conjuncts in the rule
 - Compare error rate on validation set before and after pruning
 - If error improves, prune the conjunct

B. Indirect Method:

← Extract rules from other classification models (e.g. decision trees, neural networks, etc).

Eg; Rule Extraction from Decision Tree



Rules:

R1: (Refund = Yes) => Loan

R2: (Refund = No) ^ (Marital Status = Married) => Loan

Rule simplification

Complex rules can be simplified. In above example R2 can be simplified as:

r2: (Marital Status = Married) => Loan

Advantages of Rule-Based Classifiers

- As highly expressive as decision trees
- Easy to interpret
- Easy to generate
- Can classify new instances rapidly
- Performance comparable to decision trees

Instance Based Classifier

- It Store the training records and • Use training records to predict the class label of unseen cases.

Examples:

i. Rote-learner

- Memorizes entire training data and performs classification only if attributes of record match one of the training examples exactly

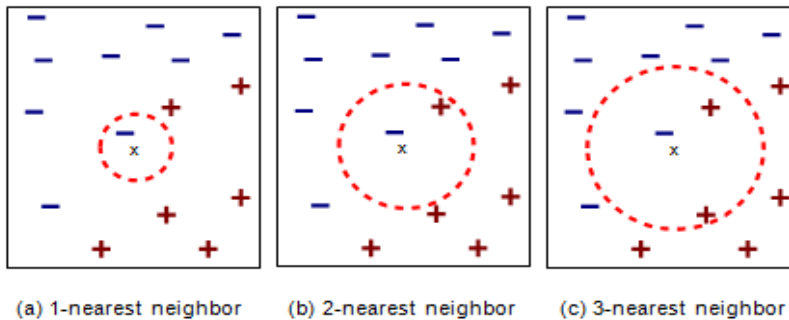
ii. Nearest neighbor

- Uses k “closest” points (nearest neighbors) for performing classification. K-closest neighbor of a record ‘X’ are data points that have the K-smallest distance of ‘X’.
- Classification based on learning by analogy i.e. by comparing a given test tuple with training tuple that are similar to it.
- Training tuples are described by n-attributes.
- When given an unknown tuple, a k-nearest- neighbor classifier searches the pattern space for the k-training tuples that are closest to the unknown tuple.
- Nearest neighbor classifier requires:
 - Set of stored records
 - Distance matrix to compute distance between records. For distance calculation any standard approach can be used such as Euclidean distance.
 - The value of ‘K’, the number of nearest neighbor to retrieve.
- To classify the unknown records
 - Compute distance to other training records.
 - Identify the k-nearest neighbor.
 - Use class label nearest neighbors to determine the class label of unknown record. In case of conflict, use majority vote for classification.

Issues of classification using k-nearest neighbor classification

i. Choosing the value of K

- One of challenge in classification is to choose the appropriate value of K. If K is too small, it is sensitive to noise points. If K is too large, neighbor may include points from other classes.
- With the change of value of K, the classification result may vary.



ii. Scaling Issue

- Attribute may have to be scaled to prevent distance measure from being dominated by one of attributes. Eg. Height, Temperature etc.

iii. Distance computing for non-numeric data.

- Use Distance as 0 for the same data and maximum possible distance for different data.

iv. Missing values

- Use maximum possible distance

Disadvantages:

- Poor accuracy when data have noise and irrelevant attributes.
- Slow when classifying test tuples.
- Classifying unknown records are relatively expensive.

Bayesian Classifier

- Bayesian classification is based on Baye's Theorem.
 - It is a statistical classifier that predicts class membership probabilities such as the probability that a given tuple belongs to a particular class.
- Baye's Law

$$P(A/B) = P(B/A) P(A) / P(B)$$

- Has high accuracy and speed for large databases.
- Has minimum error rate in comparison to all other classifier

Types.

1. Bayesian Belief Networks (Graphical Method)

- Bayesian Belief Network specifies joint conditional probability distributions.
- Bayesian Networks and Probabilistic Network are known as belief network.
- It allows class conditional independencies to be defined between subsets of variables.
- It provides a graphical model of causal relationship on which learning can be performed.
- It represents a set of random variables and their conditional dependencies via a directed acyclic graph

2. Naïve Bayesian Classifier

- The Naive Bayes Classifier technique is based on the so-called Bayesian theorem and is particularly suited when the dimensionality of the inputs is high.
- It simplifies the computational complexity.
- Naïve Bayesian Classifier assumes that the effect of an attribute value on a given class is independent of the value of other attributes i.e. class conditional independence.
- Let D be a training set of tuples and C1, C2,, Cm are their associated classes.
- Given a tuple X, the classifier will predict that X belongs to the class having highest posterior probability conditioned on X i.e. the Naïve Bayesian classifier predicts that tuple x belong to the class Ci if and only if

$$P(C_i/X) > P(C_j/X) \text{ for } 1 \leq j \leq m, j \neq i$$

$$\text{i.e. } P(C_i/X) = P(X/C_i)P(C_i) / P(X) \text{ maximum}$$

$$P(X) = \text{Constant}$$

$$P(C_i) \Rightarrow P(C_1) = P(C_2) = \dots = P(C_m)$$

So we need to maximize $P(X/C_i)$

Naïve assumption is class condition independence,

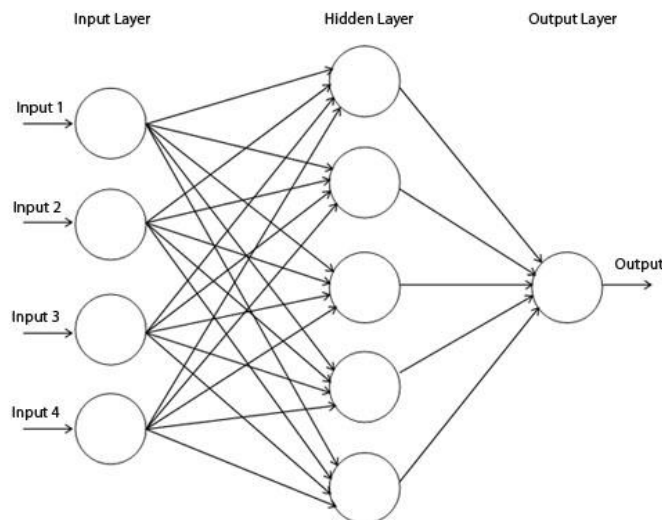
$$\begin{aligned} P(X/C_i) &= \prod_{k=1}^n P\left(\frac{x_k}{C_i}\right) \\ &= P(x_1/C_i) * P(x_2/C_i) * \dots * P(x_n/C_i) \end{aligned}$$

These probabilities can be calculated from training tuples.

Eg: Refer class notes

Artificial Neural Network (ANN) Classifier

- It is set of connected i/o units in which each connection has a weight associated with it.
- During the learning phase the network learns by adjusting the weights so as to be able to predict the correct class label of i/p labels.
- It also referred as connectionist learning due to connection between units.
- It has long training time and poor interpretability but has tolerance to noisy data.
- It can classify pattern on which they have not been trained.
- Well suited for continuous valued i/ps.
- It has parallel topology and processing.
- Before training the network topology must be designed by:
 - i. **Specifying number of i/p nodes/units:** Depends upon number of independent variable in data set.
 - ii. **Number of hidden layers:** Generally only layer is considered in most of the problem. Two layers can be designed for complex problem. Number of nodes in the hidden layer can be adjusted iteratively.
 - iii. **Number of output nodes/units:** Depends upon number of class labels of the data set.
 - iv. **Learning rate:** Can be adjusted iteratively.
 - v. **Learning algorithm:** Any appropriate learning algorithm can be selected during training phase.
 - vi. **Bias value:** Can be adjusted iteratively.
- During training the connection weights must be adjusted to fit i/p values with the o/p values.



Back propagation algorithm

Step 1: Initialization: Set all the weights and thresholds levels of the network to random numbers uniformly distributed inside a small range.

Step 2: Activation: Activate the back propagation neural network by applying i/ps and desired o/ps.

- i. Calculate the actual o/ps of the neurons in the hidden layers.
- ii. Calculate the actual o/ps of the neurons in the o/p layers.

Step 3: Weight training:

- i. Updates weights in the back propagation network by propagating backwards the errors associated with the o/p neurons.
- ii. Calculate error gradient of o/p layer and hence of neurons in the hidden layer.

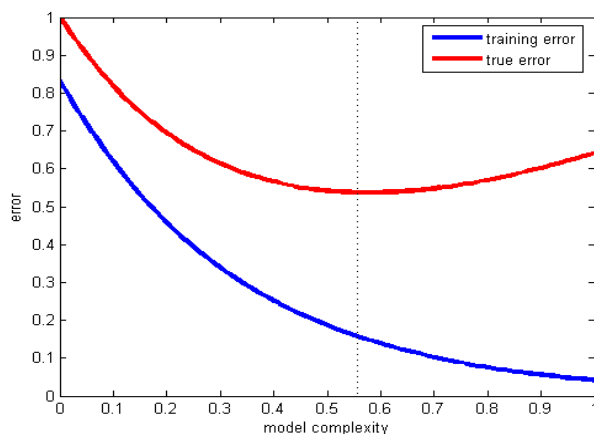
Step 4: Iteration: Increase iteration by repeating steps 2 and 3 until selected error criteria is satisfied.

(Refer textbook for mathematical equations)

Issues:

Overfitting

- Overfitting occurs when a statistical model describes random error or noise instead of the underlying relationship.
- Overfitting generally occurs when a model is excessively complex, such as having too many parameters relative to the number of observations.
- A model which has been overfit will generally have poor predictive performance.
- Overfitting depends not only on the number of parameters and data but also the conformability of the model structure.
- In order to avoid overfitting, it is necessary to use additional techniques (e.g. cross-validation, pruning (Pre or Post), model comparison,



Reason

- . Noise in training data.
- . Incomplete training data.
- . Flaw in assumed theory.

Validation

Validation techniques are motivated by two fundamental problems in pattern recognition: model selection and performance estimation

Validation Approaches:

- One approach is to use the entire training data to select our classifier and estimate the error rate, but the final model will normally overfit the training data.
- A much better approach is to split the training data into disjoint subsets cross validation (The Holdout Method)

Cross Validation (The holdout method)

Data set divided into two groups. Training set: used to train the classifier and Test set: used to estimate the error rate of the trained classifier

Total number of examples = Training Set + Test Set

Approach:

Random Sub sampling

- Random Sub sampling performs K data splits of the dataset
- Each split randomly selects (fixed) no. examples without replacement
- For each data split we retrain the classifier from scratch with the training examples and estimate error with the test examples

K-Fold Cross-Validation

- K-Fold Cross validation is similar to Random Sub sampling.
- Create a K-fold partition of the dataset, For each of K experiments, use K-1 folds for training and the remaining one for testing.
- The advantage of K-Fold Cross validation is that all the examples in the dataset are eventually used for both training and testing.
- The true error is estimated as the average error rate

Leave-one-out Cross-Validation

- Leave-one-out is the degenerate case of K-Fold Cross Validation, where K is chosen as the total number of examples where one sample is left out at each experiment.

Model Comparison:

- Models can be evaluated based on the output using different method :
 - i. Confusion Matrix
 - ii. ROC Analysis
 - iii. Others such as: Gain and Lift Charts, K-S Charts

Confusion Matrix (Contingency Table):

- A confusion matrix contains information about actual and predicted classifications done by classifier.
- Performance of such system is commonly evaluated using data in the matrix.
- It is also known as a contingency table or an error matrix, is a specific table layout that allows visualization of the performance of an algorithm.
- Each column of the matrix represents the instances in a predicted class, while each row represents the instances in an actual class.

| | Predicted Positive | Predicted Negative |
|--------------------------|---------------------------|---------------------------|
| Positive Examples | True Positive (TP) | False Negative (FN) |
| Negative Examples | False Positive (FP) | True Negative (TN) |

Accuracy: $(TP + TN) / \text{Total data count}$

Precision: $TP / (TP + FP)$ or $TN / (TN + FN)$

True Positive Rate (TPR): $TP / (TP + TN)$

True Negative Rate (TNR): $TN / (TP + TN)$

False Positive Rate (FPR): $FP / (FP + FN)$

False Negative Rate (FNR): $FN / (FP + FN)$

Example: Refer class note

ROC Analysis

- Receiver Operating Characteristic (ROC), or ROC curve, is a graphical plot that illustrates the performance of a binary classifier system as its discrimination threshold is varied.
- The curve is created by plotting the true positive rate against the false positive rate at various threshold settings.
- The ROC curve is thus the sensitivity as a function of fall-out.
- In general, if the probability distributions for both detection and false alarm are known, the ROC curve can be generated by plotting the cumulative distribution function (area under the probability distribution from $-\infty$ to $+\infty$) of the detection probability in the y-axis versus the cumulative distribution function of the false-alarm probability in x-axis.
- ROC analysis provides tools to select possibly optimal models and to discard suboptimal ones independently from (and prior to specifying) the cost context or the class distribution.
- ROC analysis is related in a direct and natural way to cost/benefit analysis of diagnostic decision making.