1. Basics and Algorithms
2. Decision Tree Classifier [human oriented]
3. Rule Based Classifier
4. Nearest Neighbor Classifier
5. Bayesian Classifier
6. Artificial Neural Network Classifier
7. Issues: Overfitting, Validation, Model Comparison

Classification, which is the task of assigning objects to one of the several predefined categories, is a universal problem that encompasses many diverse applications.

Classification is a data mining technique that assigns categories to a collection of data in order to aide in more accurate predictions and analysis.

Data mining techniques classification is the most commonly used data mining technique which contains a set of pre-classified samples to create a model which can classify the large set of data. This technique helps in deriving important information about data and metadata (data about data). This technique is closely related to cluster analysis technique and it uses decision tree or neural network system. There are two main processes involved in this technique

- **Learning** – In this process the data are analyzed by classification algorithm
- **Classification** – In this process the data is used to measure the precision of the classification rules

Examples:

- Detecting spam email messages based upon the message header and content
- Categorizing cells as bad or good based upon the results of MRI scans
- Classifying galaxies based upon their shapes
- Classification high-risk or low-risk patients based on conditions.

WHY CLASSIFICATION?

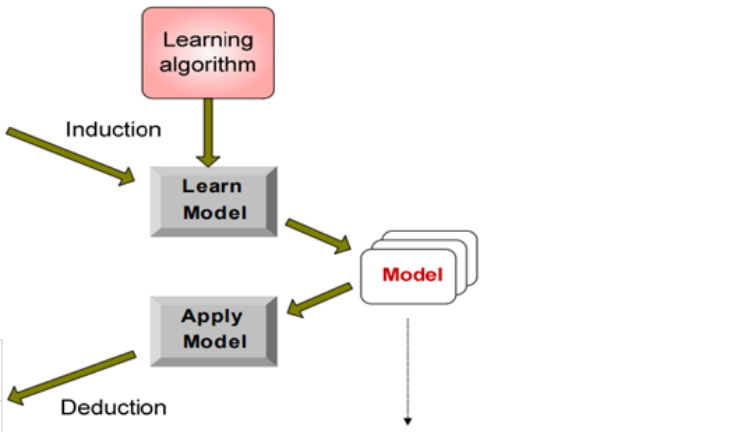
- Very large databases are becoming the norm in today's world of "big data." Imagine a database with multiple terabytes of data — a terabyte is one *trillion* bytes of data.
- Facebook alone crunches *600 terabytes of new data every single day* (as of 2014, the last time it reported these specs). The primary challenge of big data is how to make sense of it.
- And sheer volume is not the only problem: big data also tends to be diverse, unstructured and fast-changing. Consider audio and video data, social media posts, 3D data or geospatial data. This kind of data is not easily categorized or organized.
- To meet this challenge, a range of automatic methods for extracting useful information has been developed, among them *classification*.

HOW CLASSIFICATION WORKS

- The goal is to create a set of classification rules that will answer a question, make a decision, or predict behavior. To start, a set of training data is developed that contains a certain set of attributes as well as the likely outcome.
- The job of the classification algorithm is to discover how that set of attributes reaches its conclusion.
- **Scenario**: Perhaps a credit card company is trying to determine *which prospects should receive a credit card offer.*

Training Data / Training Set

Name	Age	Gender	Annual Income	Credit Card Offer
John Doe	25	M	$39,500	No
Jane Doe	56	F	$125,000	Yes



Predictor Data / Test Set

Name	Age	Gender	Annual Income	Credit Card Offer
Jack Frost	42	M	$88,000	
Mary Murray	16	F	$0	

Fig. Credit Card Offer

- 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.
- Predicts categorical class labels (discrete or nominal)
- Classifies data (constructs a model) based on the training set and the values (class labels) in a classifying attribute and uses it in classifying new data.
- The **goal** of classification is to accurately predict the target class for each case in the data
  – Apply the model i.e. classifier to previously unseen records to predict their class (class should be predicted as accurately as possible)
  – Carry out deployment based on the model (e.g. implement more profitable marketing strategies)

- Classification is the task of learning a target function **f** that maps each attribute set **x** to one of the predefined class labels **y**. target function is also known as a classification model.
- The data set can be divided into
  – **Training set** used to build the model
  – **Test set** used to determine the accuracy of the model

1. Basics and Algorithms

- Classification
  - Given a collection of records (*training set*). Each record contains a set of attributes, one of the attributes is the class.
  - Find a model for class attribute as a function of the values of other attributes.
  - Goal: previously unseen records should be assigned a class as accurately as possible.
    - A test set is used to determine the accuracy of the model. Usually, the given data set is divided into training and test sets, with training set used to build the model and test set used to validate it.
  - predicts categorical class labels (discrete or nominal)
  - classifies data (constructs a model) based on the training set and the values (class labels) in a classifying attribute and uses it in classifying new data
- Prediction
  - models continuous-valued functions, i.e., predicts unknown or missing values
- Typical applications
  - **Credit approval**: Verified credit card transactions as real or fraudulent
  - **Target marketing**:
  - **Medical diagnosis**: Predicting tumor cells as good or bad, Classifying secondary structures of protein as alpha-helix, beta-sheet, or random coil
  - **Fraud detection**: Classifying credit card transactions as valid or fraudulent
  - **Categorizing news** stories as finance, weather, entertainment, sports, etc.

- **Classification according to the kinds of databases mined**: Database systems can be classified according to different criteria (such as data models, other types of data or applications involved), each of which may require its own data mining technique. *For instance, if classifying according to **data models**, we may have a relational, transactional, object-relational, or data warehouse mining system. If classifying according to the **special types of data** handled, we may have a spatial, time-series, text, stream data, multimedia data mining system, or a World Wide Web mining system.*
- **Classification according to the kinds of knowledge mined**: Data mining systems can be categorized according to the kinds of knowledge they mine, that is, based on data mining functionalities, such as characterization, discrimination, association and correlation analysis, classification, prediction, clustering, outlier analysis, and evolution analysis.
- **Classification according to the kinds of techniques utilized**: Data mining techniques can be described according to the degree of user interaction involved (e.g., autonomous systems, interactive exploratory systems, query-driven systems) or the methods of data analysis employed (e.g., database-oriented or data warehouse– oriented techniques, machine learning, statistics, visualization, pattern recognition, neural networks, and so on).
- **Classification according to the applications adapted**: Data mining systems may be tailored specifically for finance, telecommunications, DNA, stock markets, e-mail, and so on. Different applications often require the integration of application-specific methods. Therefore, a generic, all-purpose data mining system may not fit domain-specific mining tasks.

Supervised vs. Unsupervised Learning

- Supervised learning (classification)
  – **Supervision**: The training data (observations, measurements, etc.) are accompanied by labels indicating the class of the observations
  – New data is classified based on the training set

- Unsupervised learning (clustering)
  - The class labels of training data are unknown
  - Given a set of measurements, observations, etc. with the aim of establishing the existence of classes or clusters in the data

Classification—A Two-Step Process

- Model construction: describing a set of predetermined classes
  - Each tuple/sample is assumed to belong to a predefined class, as determined by the class label attribute
  - The set of tuples used for model construction is training set
  - The model is represented as classification rules, decision trees, or mathematical formulae
- Model usage: for classifying future or unknown objects
  - Estimate accuracy of the model
    - The known label of test sample is compared with the classified result from the model
    - Accuracy rate is the percentage of test set samples that are correctly classified by the model
    - Test set is independent of training set, otherwise over-fitting will occur
  - If the accuracy is acceptable, use the model to classify data tuples whose class labels are not known

NAME	RANK	YEARS	TENURED
Mike	Assistant Prof	3	no
Mary	Assistant Prof	7	yes
Bill	Professor	2	yes
Jim	Associate Prof	7	yes
Dave	Assistant Prof	6	no
Anne	Associate Prof	3	no

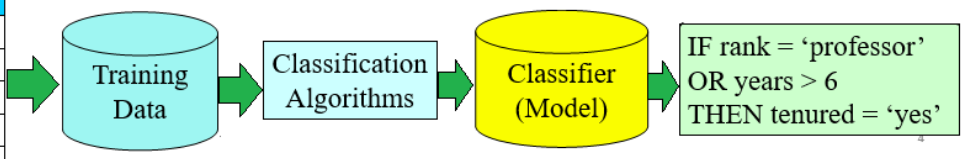


Fig. Process (1): Model Construction

NAME	RANK	YEARS	TENURED
Mike	Assistant Prof	3	no
Mary	Assistant Prof	7	yes
Bill	Professor	2	yes
Jim	Associate Prof	7	yes
Dave	Assistant Prof	6	no
Anne	Associate Prof	3	no

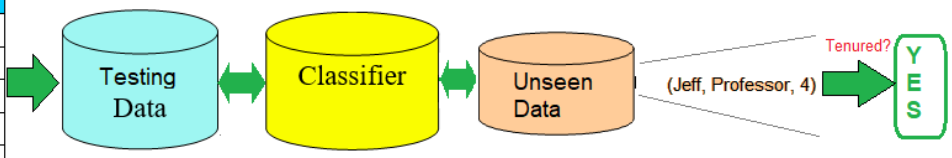


Fig. Process (2): Using the Model in Prediction

Uses of Classification Model

Classification techniques are most suited for predicting or describing data sets with binary or nominal categories. They are less effective for ordinal categories (e.g. to classify a person as a member of high-, medium-, or low-income group) because they do not consider the implicit order among the categories. Other forms of relationships, such as the subclass, superclass relationships among categories (e.g. human and apes are primates, which in turn, is a subclass of mammals) are also ignored.

- Descriptive Modeling
  - It serves as an explanatory tool to distinguish between objects of different classes.
  - Example: it would be useful – for both biologists and others – to have a descriptive model that summarizes the data and explain what features define a vertebrate as a mammal, reptile, bird, fist, or amphibian.

Name	Body Temperature	Skin Cover	Gives Birth	Aquatic Creature	Aerial Creature	Has Legs	Class Label
Human	Warm-blooded	Hair	Yes	No	No	Yes	Mammal
Python	Cold-blooded	Scales	No	No	No	No	Reptile
Whale	Warm-blooded	Scales	Yes	Yes	No	No	Mammal

- Predictive Modeling
  - used to predict the class label of unknown records.
  - a classification can be treated as a black box that automatically assigns a class label when presented with the attribute set of an unknown record.
  - suppose given the following characteristics of a creature known as a gila monster:

Name	Body Temperature	Skin Cover	Gives Birth	Aquatic Creature	Aerial Creature	Has Legs	Class Label
Frog	Cold-blooded	None	No	Semi	No	Yes	? *(amphibian)*

***** Types of Classifier:

i. Decision Tree classifier
ii. Rule Based Classifier
iii. Nearest Neighbor Classifier
iv. Bayesian Classifier
v. Artificial Neural Network (ANN) Classifier

*** Stages in Classification**

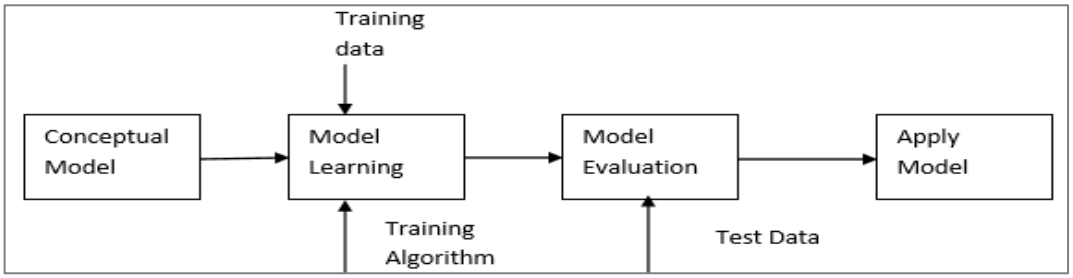
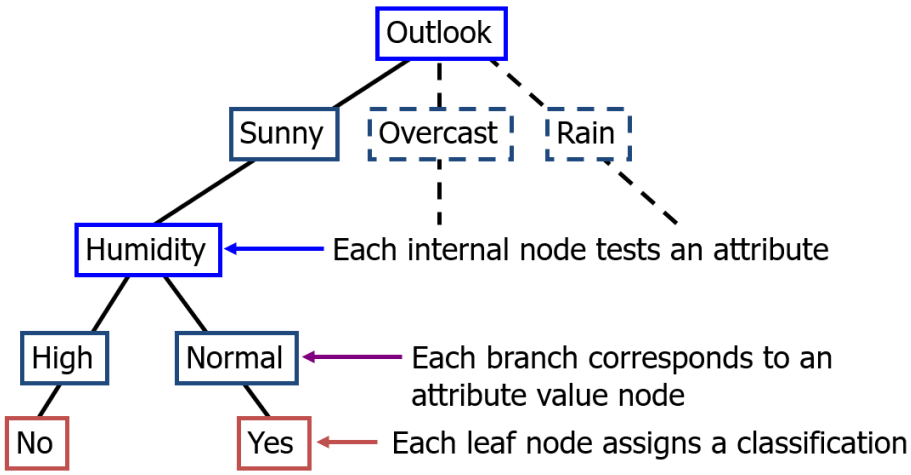


Fig: Stages in classification

2. Decision Tree Classifier [human oriented]

- 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.
- **Example**:
  - Imagine we've got a set of data containing several types, or *classes*.
    - E.g. information about customers, and class=whether or not they buy anything.
  - Can we predict, i.e *classify*, whether a previously unseen customer will buy something?
  - The idea is to ***ask a series of questions***, starting at the root, that will lead to a leaf node.
  - The ***leaf node provides the classification***.



Algorithm for Decision Tree Induction

- **Basic algorithm**
  - Tree is constructed in a top-down recursive divide-and-conquer manner
  - At start, all the training examples are at the root
  - Attributes are categorical (if continuous-valued, they are discredited in advance)
  - Examples are partitioned recursively based on selected attributes
  - Test attributes are selected on the basis of a heuristic or statistical measure (e.g., information gain)
- **Conditions for stopping partitioning**
  - All samples for a given node belong to the same class
  - There are no remaining attributes for further partitioning – majority voting is employed for classifying the leaf
  - There are no samples left

Classification by Decision Tree Induction

- Decision Tree
  - A flowchart like tree structure a flow
  - Branch represents an outcome of the test
  - Leaf node represent class labels or class distribution
- Two Phases of Tree Generation
  - Tree Construction
    - At start all the training examples are at the root
    - Partition examples recursively based on selected attributes
  - Tree Pruning
    - Identify and remove branches that reflect noise or outliers
    - Once the tree is build
      - Use of decision tree: Classifying an unknown sample

Building a decision tree

1. Select an attribute
2. Create the subsets of the example data for each value of the attribute
3. For each subset
   - *if not all the elements of the subset belong to same class repeat the steps 1-3 for the subset*

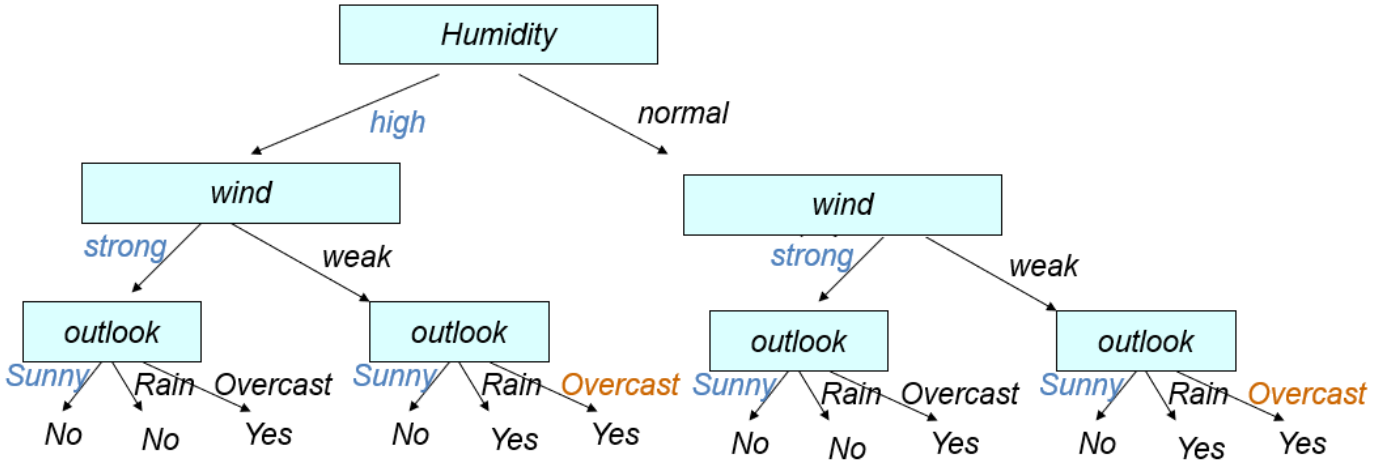
Decision Trees and Logic - for Play Tennis

Training Data

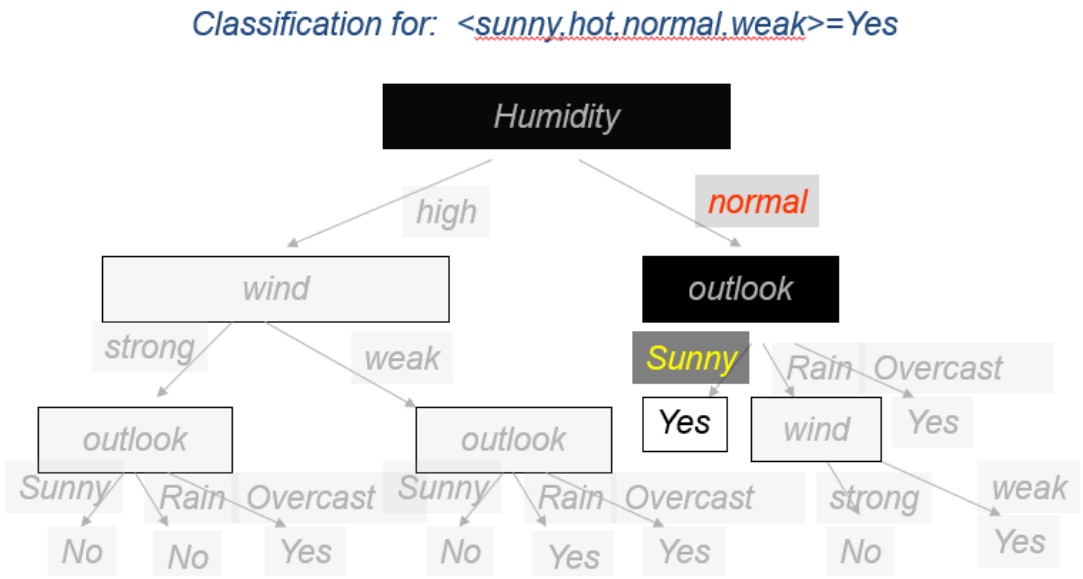
Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal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

The decision tree can be expressed as an expression or if-then-else sentences:

- *(humidity=high ∧ wind=strong ∧ outlook=overcast) ∨*
- *(humidity=high ∧ wind=weak ∧ outlook=overcast) ∨*
- *(humidity=normal ∧ outlook=sunny) ∨*
- *(humidity=normal ∧ outlook=overcast) ∨*
- *(humidity=normal ∧ outlook=rain ∧ wind=weak) ⇒ 'Yes'*

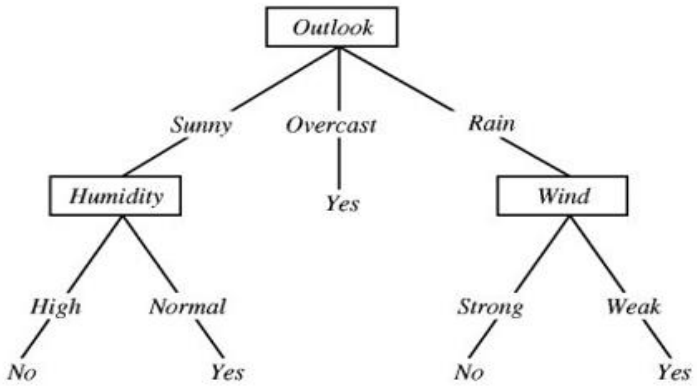


- **Classifying:** < sunny, hot, normal, weak > =?
- **Classification for:** < sunny, hot, normal, weak > **= Yes**



A Big Problem…

Here's another tree from the same training data that has a different attribute order:



Which attribute should we choose for each branch?

Decision Tree Algorithm

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

i. Hunt's Algorithm

- The decision tree is constructed in a recursive fashion by partitioning the training data into successively into subsets until each path ends in a pure subset.
- There are three steps that are done until the tree is fully grown.
  - Examine the record data and find the best attribute for the first node.
  - Split the record data based on this attribute
  - Recurse on each corresponding child node choosing other attributes
- 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 single class y_t, then t is a leaf node labelled as y_t.
  - If D_t is an empty set, then t is a leaf node labelled 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 Dt 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.

… Example of Hunt's Algorithm in Class Notes

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

Design Issues of Decision Tree Induction:

- How to split the record?
- How to specify the attribute test condition?
  - Depends on attribute types and number of ways to split the record i.e. 2-ways (Binary) split or multiway split.
  - Depends upon attribute types. (Nominal, Ordinal, Continuous)
- When to stop splitting?
  - When all records are belonging to the same class or all records have similar attributes.
  - At what point do you decide to stop the tree-growing process? This is another crucial area that must not be overlooked. One might assume that all we need to is to allow the recursive steps play out all the way through to the end.
- How to determine the best split?
  - Nodes with homogenous class distribution are preferred.
  - Measure the node impurity.
    - Gini-Index
    - Entropy
    - 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, pi = Probability of i^{th} class.

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

… Example of GINI Index Algorithm in Class Notes

Entropy

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

$$H(S) = \text{-p1 log p1 - p2 log p2 ... - -pn log pn} = -\sum_{x \in X} p(x) \log_2 p(x)$$

Where,

S The current dataset for which entropy is being calculated (changes every iteration of the ID3 algorithm)
X - Set of classes in
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.

… Example of Entropy in ID3 Algorithm Example Class Notes

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 on this iteration.

… Example of Information Gain in ID3 Algorithm Example Class Notes

ii. 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.

Algorithm

- Every element in the subset belongs to the same class, then the node is turned into a leaf and labelled 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 smallest entropy (or largest information gain) value to split data.
  - Partition the data into subset.
- Recursively repeat until all data are correctly classified

… Example of ID3 Algorithm in Class Notes

Advantages of using ID3

- Understandable prediction rules are created from the training data.
- Builds the fastest tree and builds a short tree.
- Only need to test enough attributes until all data is classified.
- Finding leaf nodes enables test data to be pruned, reducing number of tests.
- Whole dataset is searched to create tree.

Disadvantages of using ID3

- Data may be over-fitted or over-classified, if a small sample is tested.
- Only one attribute at a time is tested for making a decision.
- Classifying continuous data may be computationally expensive.

Tree Pruning *Trimming*

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. Pruning reduces the complexity of the final classifier, and hence improves predictive accuracy by the reduction of overfitting.

One of the methods used to address over-fitting in decision tree is called **pruning** which is done after the initial training is complete.

- 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.
- 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.

Tree pruning approaches: to address overfitting (due to noise, lack of samples)

I. **Pre-pruning** - The tree is pruned by halting its construction at some node after checking some measures e.g. Information gain, Gini index, etc
II. **Post pruning** - This approach removes subtree form fully grown tree.

I. 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 e.g. ID3 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

II. 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: selects a subtree and replaces it with a single leaf.
  - Subtree raising: selects a subtree and replaces it with the child one i.e. a *"sub-subtree" replaces its parent*)

- Possible **strategies**:
  - error estimation
  - significance testing
  - MDL principle

  … Examples of Pre and Post Pruning in Class Notes

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

3. Rule Based Classifier

Rule-based classifier makes use of a set of IF-THEN rules for classification. We can express a rule in the following from –

IF *condition* THEN *conclusion*

Let us consider a rule R1,

R1: IF age = youth AND student = yes THEN buy_computer = yes

We can also write rule R1 as follows –

R1: (age = youth) ^ (student = yes))(buys computer = yes)

Points to remember –

- The IF part of the rule is called **rule antecedent** or **precondition**.
- The THEN part of the rule is called **rule consequent**.
- The **antecedent** part the condition consist of one or more attribute tests and these tests are logically ANDed.
- The **consequent** part consists of class prediction.

A rule **R** covers an instance x if the attributes of the instance satisfy the condition (LHS) of the rule

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

Name	Blood Type	Give Birth	Can Fly	Live in Water	Class
hawk	warm	no	yes	no	?
grizzly bear	warm	yes	no	no	?

The rule R1 covers a hawk ⇒ Class = Bird
The rule R3 covers the grizzly bear ⇒ Class = Mammal

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.

Quality of a classification

Quality of a classification rule can be evaluated by

- **Coverage**: fraction of records that satisfy the antecedent of a rule
  Coverage = N_{covers} / D
  Where, N_{covers} = *number of record that can be classified by the rule*. D = *total data set*
- **Accuracy**: fraction of records covered by the rule that belong to the class on the RHS
  Accuracy = $N_{correct}$ / N_{covers}
  Where, Ncorrect = *Number of records that are correctly classified by the rule* N_{covers} = *Number of record that can be classified by the rule*

Tid	Refund	Marital Status	Taxable Income	Class
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

(Status = Single) → No
Matching LHS = 4, AND LHS = RHS = 2
Coverage = 40%, Accuracy = 50%

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, the rule set is known as a decision list.
- Class-based ordering: Rules that belong to the same class appear together

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 are:

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 : *Turning left and turning right, Tossing a coin are Mutually Exclusive (you can't do both at the same time)*

- 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 trigger by more than one rule. To make mutually exclusive we apply rule ordering (ranked or ordered according to their priority or quality not by same class appearing together).

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 not 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 training 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

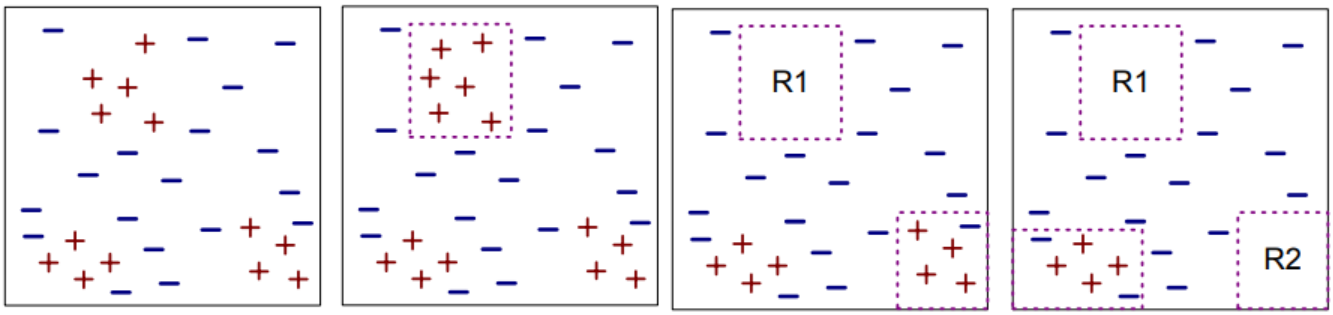


Fig. Sequential Covering

Aspects of Sequential Covering

i. Rule Growing
ii. Instance Elimination
iii. Rule Evaluation
iv. Stopping Criterion
v. Rule Pruning

i. Rule Growing CN2 Algorithm:

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

RIPPER Algorithm:

- Start from an empty rule: {} => class
- Add conjuncts that maximize FOIL's information gain measure:

R0: {} => class (initial rule)
R1: {A} => class (rule after adding conjunct)
Gain (R0, R1) = t [log (p1/(p1+n1)) – log (p0/(p0 + n0))]

Where, t: number of positive instances covered by both R0 and R1
p0: number of positive instances covered by R0
n0: number of negative instances covered by R0
p1: number of positive instances covered by R1
n1: number of negative instances covered by R1

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

Compare rules R2 and R3 in the diagram

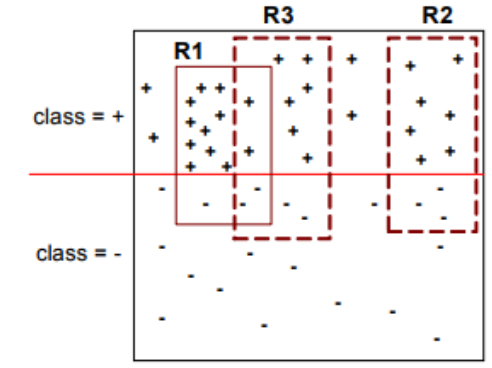


Fig. Instance Elimination

iii. Rule Evaluation

- Metrics:
  - Accuracy = $\frac{n_c}{n}$
  - Laplace = $\frac{n_c + 1}{n + k}$
  - 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

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

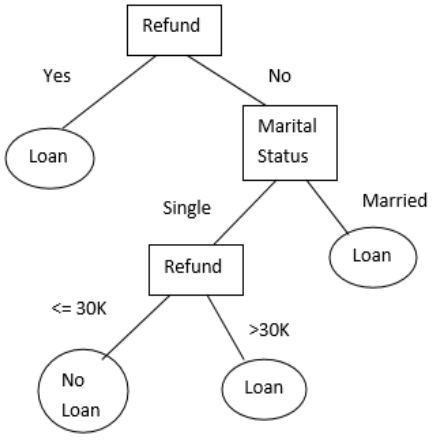
v. Rule Pruning

- 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. learn decision trees then convert to rules, learn neural networks then extract the rules, etc.).*

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



Eg; Rule Extraction from Decision Tree

Advantages of Rule-Based Classifiers

- As highly expressive as decision trees
- Easy to interpret and Easy to generate

- Can classify new instances rapidly
- Performance comparable to decision trees

1. Nearest Neighbor Classifier: *uses conditional probability to categorize spam emails.*

Learning by analogy: "Tell me who your friends are then I'll tell you who you are."

One of the simplest decision procedures that can be used for classification technique is the nearest neighbor (NN) rule. It classifies a sample based on the category of its nearest neighbor. When large samples are involved, it can be shown that this rule has a probability of error. The nearest neighbor based classifiers use some or all the patterns available in the training set to classify a test pattern. These classifiers essentially involve finding the similarity between the test pattern and every pattern in the training set.

- an instance is assigned to the most common class among the instances similar to it
  - ✓ how to measure similarity between instances
  - ✓ how to choose the most common class
- Learning Algorithm:
  - Store training examples
- Prediction Algorithm:
  - To classify a new example x by finding the training example (x^i ,y^i) that is nearest to x'
  - Guess the class $y=y^i$
- Uses k "closest" points (nearest neighbors) for performing classification. K-closet 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 matric to compute distance between records. For distance calculation any standard approach can be used sch 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.

Classification steps

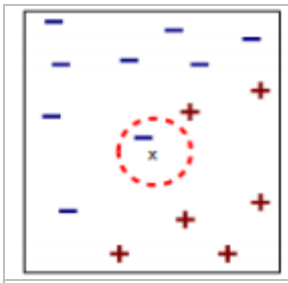
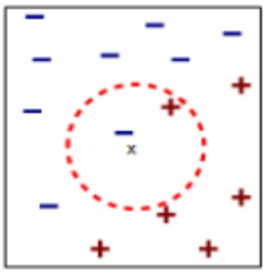
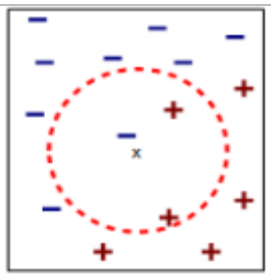
1. **Training** phase: a model is constructed from the training instances.
   - Classification algorithm finds relationships between predictors and targets
   - Relationships are summarized in a model
2) **Testing** phase: test the model on a test sample whose class labels are known but not used for training the model
3) **Usage** phase: use the model for classification on new data whose class labels are unknown

How does it works ?

1) Initialization, define k
2) Compute distance (test instance, each training instance)
3) Sort the distance
4) Take k nearest neighbors
5) Apply simple majority
6) Class

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.

		
1 – Nearest Neighbor	2 – Nearest Neighbor	3 – Nearest Neighbor

ii. **Scaling Issue**
- Attribute may have to be scaled to prevent distance measure from being dominated by one of attributes. E.g. 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

Advantages
- It is conceptually simple.
- It does not require learning (term: memory-base).
- It can be used even with few examples.
- Even for moderate k: wonderful performer.
- It works very well in low dimensions for complex decision surfaces.
- For R∗ = 0: consistent.

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

2. 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

$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)}$$

where **A** and **B** are events and **P(B) != 0**

P(A|B) is a conditional probability: the likelihood of event **A** occurring given that **B** is true.
P(B|A) is also a conditional probability: the likelihood of event **B** occurring given that A is true.
P(A) and **P(B)** are the probabilities of observing A and **B** independently of each other; this is known as the marginal probability.

- 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

$$P(Ci/X) > P(Cj/X) \text{ for } 1 \leq j \geq m, j \neq i$$

i.e. P(Ci/X) = P(X/Ci)P(Ci)/ P(X) maximum
P(X) = Constant
P(Ci) => P(C1) = P(C2) = ……. = P(Cm)

So we need to maximize P(X/Ci)
Naïve assumption is class condition independence,

$$P(X/Ci) = \prod_{k=1}^{n} P(\frac{Xk}{Ci})$$

$$= P(x1/Ci) * P(x2/Ci) * \ldots\ldots\ldots\ldots * P(xn/Ci)$$

These probabilities can be calculated from training tuples.
… Example of Naïve Bayesian Classifier is shown in Class Notes

Applications of Naive Bayes Algorithms

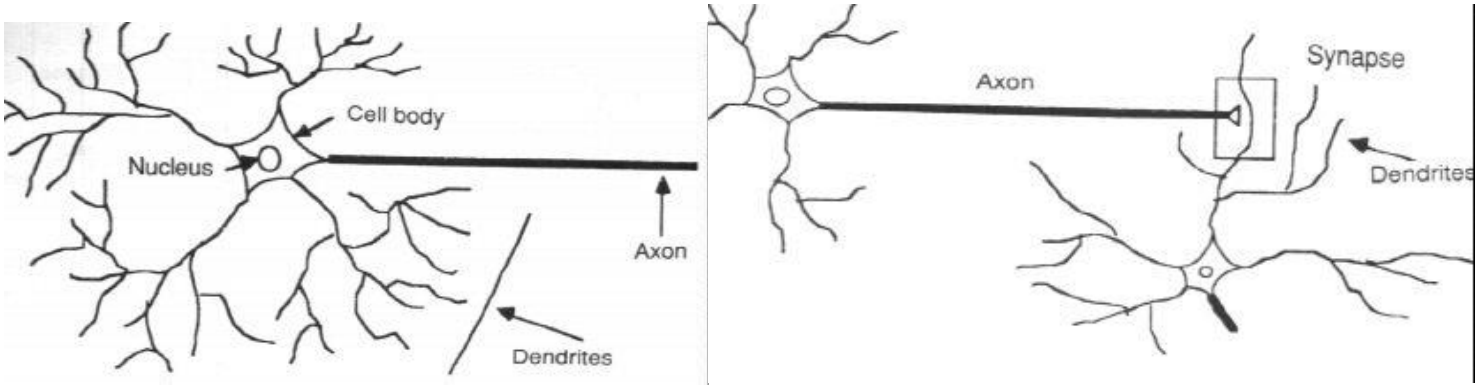
- **Real time Prediction:** Naive Bayes is an eager learning classifier and it is sure fast. Thus, it could be used for making predictions in real time.
- **Multi class Prediction:** This algorithm is also well known for multi class prediction feature. Here we can predict the probability of multiple classes of target variable.
- **Text classification/ Spam Filtering/ Sentiment Analysis:** Naive Bayes classifiers mostly used in text classification (due to better result in multi class problems and independence rule) have higher success rate as compared to other algorithms. As a result, it is widely used in Spam filtering (identify spam e-mail) and Sentiment Analysis (in social media analysis, to identify positive and negative customer sentiments)
- **Recommendation System:** Naive Bayes Classifier and Collaborative Filtering together builds a Recommendation System that uses machine learning and data mining techniques to filter unseen information and predict whether a user would like a given resource or not.

3. Artificial Neural Network Classifier

What are Neural Networks?

- Neural networks are a new method of programming computers.
- Programs that employ neural nets are also capable of learning on their own and adapting to changing conditions.
- An Artificial Neural Network (ANN) is an information processing paradigm that is inspired by the biological nervous systems, such as the human brain's information processing mechanism.
- The key element of this paradigm is the novel structure of the information processing system. It is composed of a large number of highly interconnected processing elements (neurons) working in unison to solve specific problems. NNs, like people, learn by example.
- An NN is configured for a specific application, such as pattern recognition or data classification, through a learning process. Learning in biological systems involves adjustments to the synaptic (between nerve cells) connections that exist between the neurons. This is true of NNs as well.

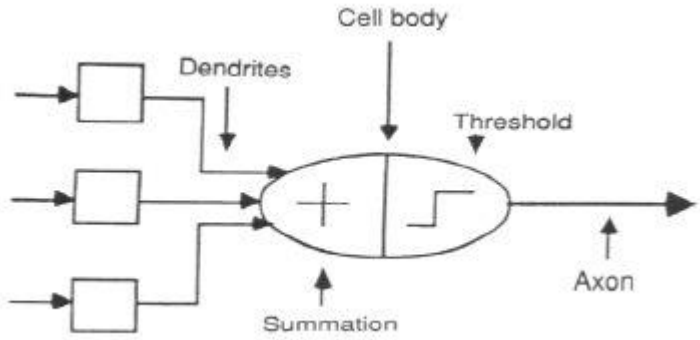
How the Human Brain learns?



- In the human brain, a typical neuron collects signals from others through a host of fine structures called *dendrites*.
- The neuron sends out spikes of electrical activity through a long, thin stand known as an *axon*, which splits into thousands of branches.
- At the end of each branch, a structure called a *synapse* converts the activity from the axon into electrical effects that inhibit (prevent) or excite activity in the connected neurons.

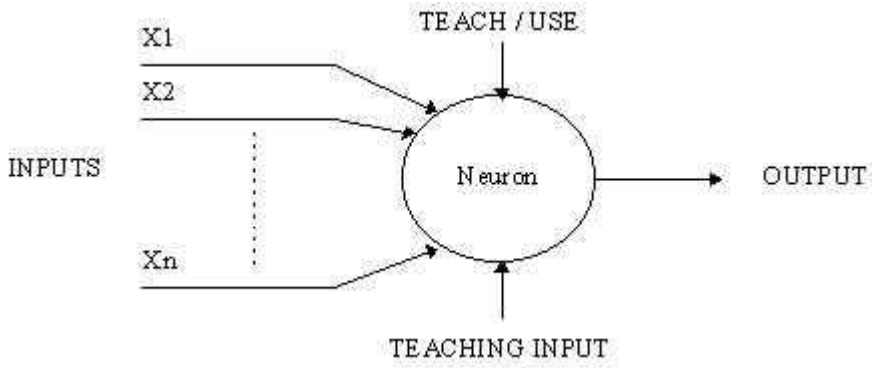
A Neuron Model

- When a neuron receives excitatory input that is sufficiently large compared with its inhibitory input, it sends a spike of electrical activity down its axon. Learning occurs by changing the effectiveness of the synapses so that the influence of one neuron on another change.



- We conduct these neural networks by first trying to deduce the essential features of neurons and their interconnections.
- We then typically program a computer to simulate these features.

A Simple Neuron



- An artificial neuron is a device with many inputs and one output.
- The neuron has two modes of operation;
  a. **Training mode**, the neuron can be trained to fire (or not), for particular input patterns.
  b. **Using mode**, when a taught input pattern is detected at the input, its associated output becomes the current output. If the input pattern does not belong in the taught list of input patterns, the firing rule is used to determine whether to fire or not.
- A firing rule determines how one calculates whether a neuron should fire for any input pattern. It relates to all the input patterns, not only the ones on which the node was trained on previously.

- 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:
  - **Specifying number of i/p nodes/units**: Depends upon number of independent variable in data set.
  - **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.
  - **Number of output nodes/units**: Depends upon number of class labels of the data set.
  - **Learning rate**: Can be adjusted iteratively. v. Learning algorithm: Any appropriate learning algorithm can be selected during training phase.
  - **Bias value**: Can be adjusted iteratively.
- During training the connection weights must be adjusted to fit i/p values with the o/p values.

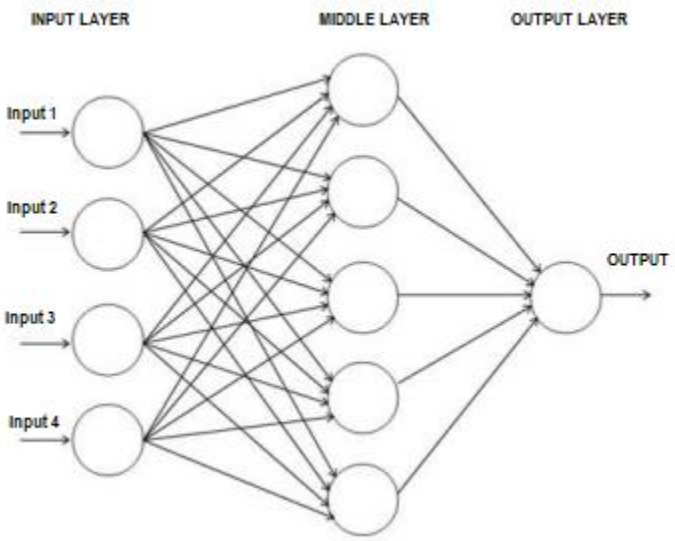


Fig. Diagram of an Neural Networks

Network Layers

- **Input Layer** - The activity of the input units represents the raw information that is fed into the network.
- **Hidden Layer** - The activity of each hidden unit is determined by the activities of the input units and the weights on the connections between the input and the hidden units.
- **Output Layer** - The behavior of the output units depends on the activity of the hidden units and the weights between the hidden and output units.

Back propagation algorithm

Backpropagation is a common method for training a neural network.

- **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.
  - Calculate the actual o/ps of the neurons in the hidden layers.
  - Calculate the actual o/ps of the neurons in the o/p layers.
- **Step 3: Weight training**:
  - Updates weights in the back-propagation network by propagating backwards the errors associated with the o/p neurons.
  - 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.

Neural Network in Use

Since neural networks are best at identifying patterns or trends in data, they are well suited for prediction or forecasting needs including:

- sales forecasting
- industrial process control
- customer research
- data validation
- risk management

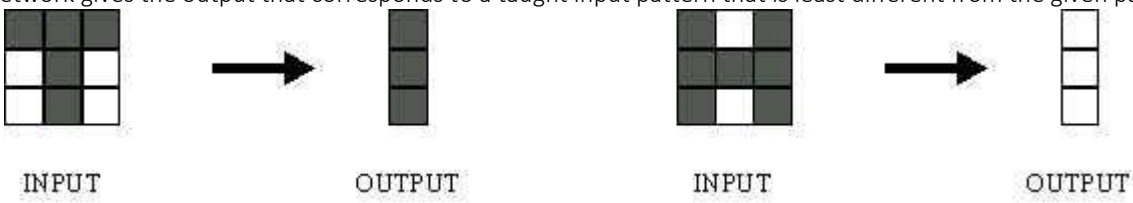
ANN are also used in the following specific examples: Recognition of speakers in communications; Diagnosis of hepatitis; Undersea mine detection; Texture analysis; Three-dimensional object recognition; Hand-written word recognition; and Facial recognition.

Example: Neural networks in Medicine

- Artificial Neural Networks (ANN) are currently a 'hot' research area in medicine and it is believed that they will receive extensive application to biomedical systems in the next few years.
- At the moment, the research is mostly on modeling parts of the human body and recognizing diseases from various scans (e.g. cardiograms, CAT scans, ultrasonic scans, etc.).
- Neural networks are ideal in recognizing diseases using scans since there is no need to provide a specific algorithm on how to identify the disease.
- Neural networks learn by example so the details of how to recognize the disease are not needed.
- What is needed is a set of examples that are representative of all the variations of the disease.
- The quantity of examples is not as important as the **'quality'**. The examples need to be selected very carefully if the system is to perform reliably and efficiently.

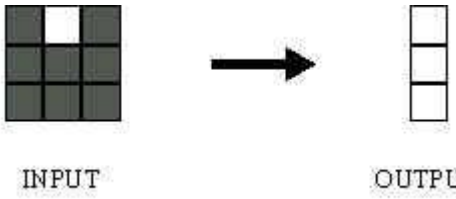
Example: Neural networks in Pattern Recognition

- An important application of neural networks is pattern recognition. Pattern recognition can be implemented by using a feed-forward neural network that has been trained accordingly.
- During training, the network is trained to associate outputs with input patterns.
- When the network is used, it identifies the input pattern and tries to output the associated output pattern.
- The power of neural networks comes to life when a pattern that has no output associated with it, is given as an input.
- In this case, the network gives the output that corresponds to a taught input pattern that is least different from the given pattern.

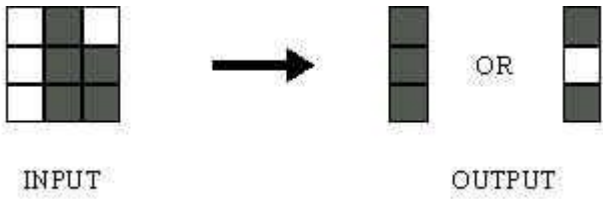


- Suppose a network is trained to recognize the patterns T and H. The associated patterns are all black and all white respectively as shown above.

- Since the input pattern looks more like a 'T', when the network classifies it, it sees the input closely resembling 'T' and outputs the pattern that represents a 'T'.



- The input pattern here closely resembles 'H' with a slight difference. The network in this case classifies it as an 'H' and outputs the pattern representing an 'H'.



- Here the top row is 2 errors away from a 'T' and 3 errors away from an H. So the top output is a black.
- The middle row is 1 error away from both T and H, so the output is random.
- The bottom row is 1 error away from T and 2 away from H. Therefore the output is black.
- Since the input resembles a 'T' more than an 'H' the output of the network is in favor of a 'T'.

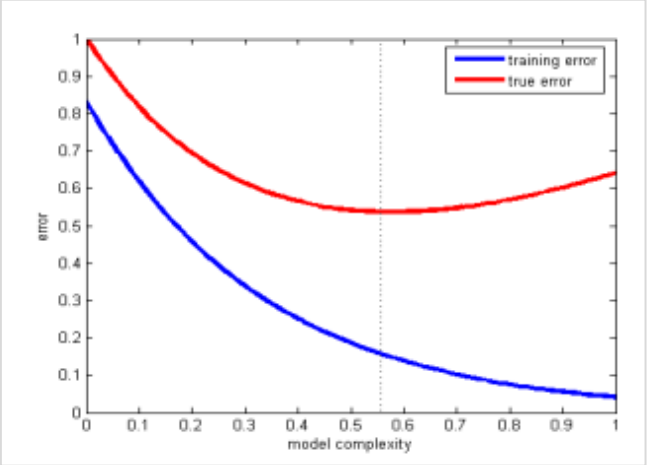
Disadvantage of Neural Network

- The individual relations between the input variables and the output variables are not developed by engineering judgment so that the model tends to be a black box or input/output table without analytical basis.
- The sample size has to be large.
- Requires lot of trial and error so training can be time consuming.

4. Issues : Overfitting, Validation, Model Comparison

A. 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 of Overfitting.

- Noise in training data: e.g. Fig(a) decision boundary is distorted by noise point
- Incomplete training data: e.g. Fig(b) difficult to predict correctly the class labels of the region
- Error in assumed theory.

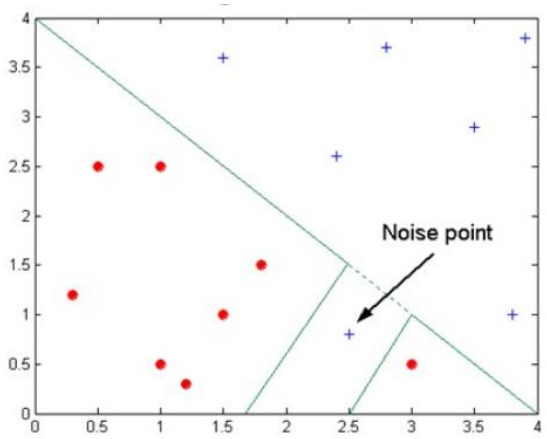


Fig. Overfitting due to the Noise

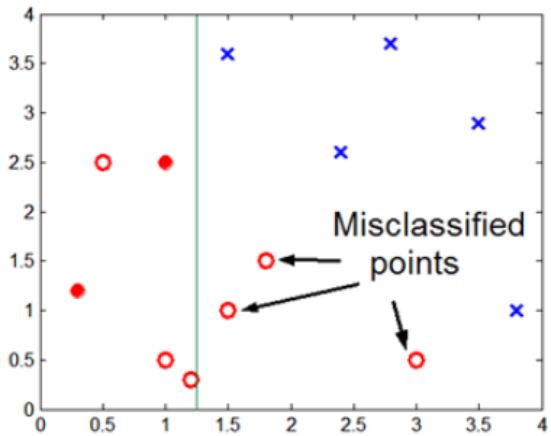


Fig.(b) Overfitting due to the incomplete data

B. Validation

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

Validation Approaches:

i. 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.

ii. 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.

C. 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

i. 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	**Negative**
Actual Class	**Positive**	True Positive (TP)	False Negative (FN)
	Negative	False Positive (FP)	True Negative (TN)

Where true positive(TP) is an actual positive that has been predicted to be positive.
Where true negative(TN) is an actual negative that has been predicted to be negative.
Where false positive(FP) is an actual negative that has been predicted to be positive.
Where false negative(FN) is an actual positive that has been predicted to be negative.

- Accuracy (total no of predictions that were correct) = (TP + TN) / Total data count (TP+TN+FP+FN)
- Precision or Positive Predictive Value (proportion of positive cases that were correctly identified) = TP / (TP + FP) or TN/ (TN + FN)
- Sensitivity or Recall (the proportion of actual positive cases which are correctly identified) = TP/(FN+FP)
- 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:

	a = yes	**B = no**
a = yes	9	2
b = no	3	5

TP Rate = TP / (TP +TN) = 9 / (9+5) = 64.28%
FP Rate = FP / (FP +FN) = 3/(3+2) = 60%
Recall Value = TP Rate

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 -∞ to +∞ 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.

… Reference : Er. Pratap Sapkota