Data Mining:
Classification: Advanced Methods

Bayesian Belief Networks, Backpropagation

Outline

- Probability theory 
- Bayesian approach to Probability theory
- Variables and probability distributions
- Joint events and marginalization
- Conditional Probability
- Bayes Theorem
- Chain Rule
- Independence and Conditional Independence
- Bayesian Belief Networks

Probability Theory: basics

- Frequentist approach to probability:
- the probability P of an uncertain event A, written P(A), is defined by the frequency of that event based on previous observations.
- For example, if A is the statement :"90% of Indians like Cricket" then, for a random sample of population the probability of A occurring is P(A)=0.90

Outline

- Probability theory
- Bayesian approach to Probability theory 
- Variables and probability distributions
- Joint events and marginalization
- Conditional Probability
- Bayes Theorem
- Chain Rule
- Independence and Conditional Independence
- Bayesian Belief Networks

Bayesian approach to Probability

- Frequentist approach to probability is defined by the frequency of that event based on previous observations.
- What if no such history is available?
- For example, "testing a new flight control system for possible faults".

Bayesian approach to Probability

- Bayesian probability is a formalism that allows us to reason about beliefs under conditions of uncertainty.
- If we have observed that a particular event has happened, such as India winning the ICC World Cup in 1983, then there is no uncertainty about it. However, suppose *a* is the statement
  - "India win the ICC World Cup in the year 2019"
- Since this is a statement about a future event, nobody can state with any certainty whether or not it is true. Different people may have different beliefs in the statement depending on their specific knowledge of factors that might effect its likelihood.

Bayesian approach to Probability

- Let Norman and Martin be two persons with their respective ***beliefs*** in ***a***

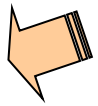
  "India win the ICC World Cup in the year 2019"

- Norman has a **strong belief** about the team based on his knowledge about the team and past achievements.
- Martin has a **weaker belief** on the statement **a** based on some inside knowledge about the internal politics within the team.
- Thus, in general, a persons subjective belief in a statement **a** will depend on some body of knowledge **K** represented as

$$P(a|K)$$

- Norman's belief in **a** is different from Martin's because they are using different K's.

Outline

- Probability theory
- Bayesian approach to Probability theory
- Variables and probability distributions 
- Joint events and marginalization
- Conditional Probability
- Bayes Theorem
- Chain Rule
- Independence and Conditional Independence
- Bayesian Belief Networks

Variables and probability distributions

- We have seen the example of the uncertain event **a**="India win the ICC World Cup in the year 2019".We can think of this as just one state of the variable A which represents "ICC World Cup winners in 2019 ".In this case A has many states, one for each team entering the world cup. We write this as

$$A = \{a_1, a_2, \dots, a_n\}$$

where a_1="India", a_2="Australia", a_3="England"

- A is a finite discrete variable.
- In general, if A is a variable with states $a_1, a_2, \dots, a_n$:

$$\sum_{i=1}^{n} P(a_i) = 1$$

- The probability distribution of A written $P(A)$, is simply the set of values $\{P(a_1), P(a_2), P(a_3)\}$

Outline

- Probability theory
- Bayesian approach to Probability theory
- Variables and probability distributions
- Joint events and marginalization 
- Conditional Probability
- Bayes Theorem
- Chain Rule
- Independence and Conditional Independence
- Bayesian Belief Networks

Joint event and Marginalization

- For our test control system X, let us suppose it is made up of two subsystems. Let A be the number of critical faults in the first subsystem and B be the number of critical faults in the second subsystem.
- Suppose that

$$A = \{a_1, a_2, a_3\} \text{ where } a_1=0,\ a_2=1 \text{ and } a_3=\text{"A>1"}$$

$$B = \{b_1, b_2, b_3\} \text{ where } b_1=0,\ b_2=1 \text{ and } b_3=\text{"A>1"}$$

- The overall number of critical faults in the system (Joint events A and B) also called the ***Joint Probability Distribution*** can be expressed as:

$$P(A,B)$$

- which is simply the set of values

$$\{P(a_1,b_1), P(a_1,b_2), P(a_1,b_3), P(a_2,b_1), P(a_2,b_2), P(a_2,b_3),$$
$$P(a_3,b_1), P(a_3,b_2), P(a_3,b_3)\}$$

Joint events and marginalization

- If we know the joint probability distribution $P(A, B)$ then we can calculate $P(A)$ by a formula (called *marginalisation*)

$$P(a) = \sum_i P(a, b_i)$$

- This is because the events (a, b_1), (a, b_2), ..., (a, b_m) are mutually exclusive. When we calculate $P(A)$ in this way from the joint probability distribution we say that the variable B is marginalised out of $P(A, B)$.

Outline

- Probability theory
- Bayesian approach to Probability theory
- Variables and probability distributions
- Joint events and marginalization
- Conditional Probability 
- Bayes Theorem
- Chain Rule
- Independence and Conditional Independence
- Bayesian Belief Networks

Conditional probability

- In general we write $P(A|B)$ to represent a ***belief*** in A under the assumption that B is known.
- Conditional probability is expressed as:

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

- In those cases where $P(A|B) = P(A)$ we say that A and B are *independent*.
- If $P(A|B,C) = P(A|C)$ we say that A and B are *conditionally independent* given C.

Outline

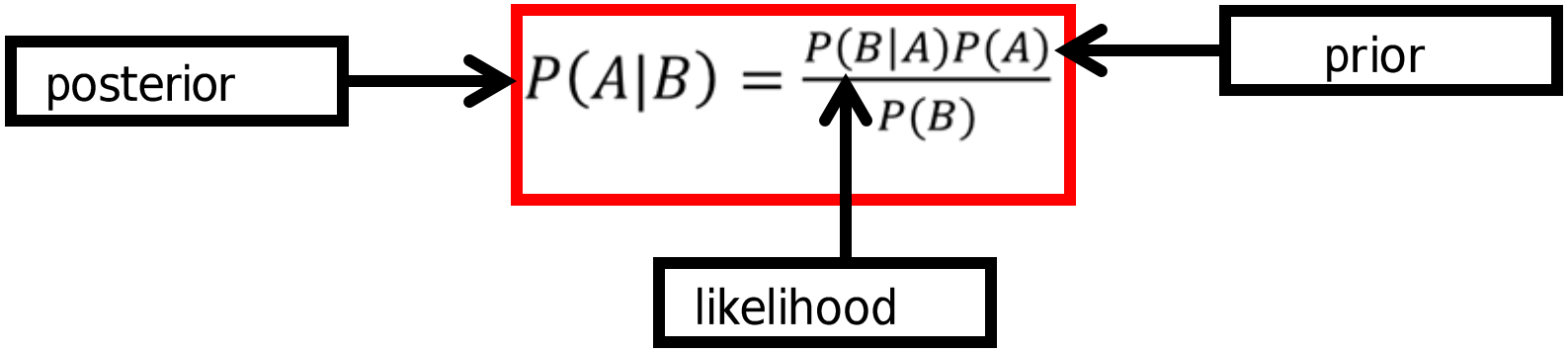
- Probability theory
- Bayesian approach to Probability theory
- Variables and probability distributions
- Joint events and marginalization
- Conditional Probability
- Bayes Theorem 
- Chain Rule
- Independence and Conditional Independence
- Bayesian Belief Networks

Bayes' Theorem

- True Bayesians actually consider conditional probabilities as more basic than joint probabilities .
- Conditional probability is expressed as:

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

- Rearranging the equation we get $P(A|B)P(B) = P(A,B)$
- By symmetry, we get $P(B|A)P(A) = P(A,B)$
- So, $P(A|B)P(B) = P(B|A)P(A)$ which gives



Bayes' Theorem

- A: 'Person has cancer, $P(A)$=0.1 (*prior*)
- B : 'Person is smoker', $P(B) = 0.5$
- $P(A|B)$=0.8 (*likelihood*)
- What is $P(A|B)$? (*posterior*)

Posterior probability likelihood prior

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

- $P(A|B)=\frac{0.8\times0.1}{0.5}$
- So $P(A|B)$=0.16

Bayes Rule Example

Black +White balls

Has 3 times of black balls

Has 3 times of white balls

Black + white balls

- Select any one bag and pick 5 balls at random, replacing each ball after it has been selected.
- The result is that we find 4 white balls and one black.
- What is the probability that we were using the bag with mainly white balls?

Bayes Rule Example

Black +White balls

Has 3 times of black balls

Has 3 times of white balls

Black + white balls

- Let A be the random variable "bag chosen" then $A = \{a_1, a_2\}$ where a_1 represents "bag with mostly white balls" and a_2 represents "bag with mostly black balls" .
- We know that $P(a_1) = P(a_2) = {}^1/_2$ since we choose the bag at random.
- Let B be the event "4 white balls and one black ball chosen from 5 selections".

Bayes Rule Example

Black +White balls

Has 3 times of black balls

Has 3 times of white balls

Black + white balls

- Then we have to calculate $P(a_1|B)$

$$P(a_1|B) = \frac{P(B|a_1).P(a_1)}{P(B|a_1).P(a_1) + P(B|a_2).P(a_2)}$$

- Now, for the bag with mostly white balls the probability of a ball being white is ¾ and the probability of a ball being black is ¼. Thus, we can use the Binomial Theorem, to compute $P(B|a_1)$ as:

$$P(B|a_1) = \binom{5}{1}\left(\frac{3}{4}\right)^4\left(\frac{1}{4}\right)^1 = \frac{405}{1024}$$

Bayes Rule Example

Black +White balls

Black + white balls

Has 3 times of black balls

Has 3 times of white balls

- Similarly

$$P(B|a_2)=\binom{5}{1}\left(\frac{1}{4}\right)^4\left(\frac{3}{4}\right)^1=\frac{15}{1024}$$

- Hence,

$$P(a_1|B)=\frac{405/1024}{405/1024+15/1024}=\frac{405}{420}=0.964$$

Outline

- Probability theory
- Bayesian approach to Probability theory
- Variables and probability distributions
- Joint events and marginalization
- Conditional Probability
- Bayes Theorem
- Chain Rule 
- Independence and Conditional Independence
- Bayesian Belief Networks

Likelihood Ratio

- We have seen that Bayes' rule computes $P(A|B)$ in terms of $P(B|A)$.
- The expression $P(B|A)$ is called the likelihood of A.
- In the example above A had two values a_1 and a_2.
- The ratio $\frac{P(B|a_1)}{P(B|a_2)}$ is called the likelihood ratio.
- In the example the likelihood ratio computes to 405/15 = 27. This tells us that the 'odds' on the bag being the one mainly with white balls is 27 to 1.

Chain rule

- Rearranging conditional probability we get the so-called *product rule*:

$$P(A,B) = P(A|B)P(B)$$

- We can extend this for three variables:

$$P(A,B,C) = P(A|B,C)P(B,C) = P(A|B,C)P(B|C)P(C)$$

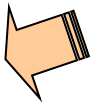
- and in general to n variables:

$$\boldsymbol{P(A_1, A_2, \ldots A_n) = P(A_1|A_2, \ldots A_n)P(A_2|A_3, \ldots A_n)P(A_{n-1}|A_n)P(A_n)}$$

- In general we refer to this as the *chain rule*.
- This formula is especially significant for Bayesian Belief Nets . It provides a means of calculating the full joint probability distribution ; in BBNs many of the variables $\boldsymbol{A_i}$ will be conditionally independent which means that the formula can be simplified as shown here

$$P(A_1, \ldots, A_n) = \prod_{i=1}^{n} P(A_i \mid Parents(A_i))$$

Outline

- Probability theory
- Bayesian approach to Probability theory
- Variables and probability distributions
- Joint events and marginalization
- Conditional Probability
- Bayes Theorem
- Chain Rule
- Independence and Conditional Independence 
- Bayesian Belief Networks

Independence and conditional independence

- The conditional probability of A given B is represented by $P(A|B)$. The variables A and B are said to be *independent* if $P(A) = P(A|B)$ (or alternatively if $P(A,B) = P(A)P(B)$)
- **Example1:** Suppose Norman and Martin each toss separate coins. Let A represent the variable "Norman's toss outcome", and B represent the variable "Martin's toss outcome".
- Both A and B have two possible values (Heads and Tails). It would be uncontroversial to assume that A and B are independent. Evidence about B will not change our belief in A.

Independence and conditional independence

- **Example2:** Now suppose both Martin and Norman toss the same coin. Again let A represent the variable "Norman's toss outcome", and B represent the variable "Martin's toss outcome". Assume also that there is a possibility that the coin is biased towards heads but we do not know this for certain.
- In this case A and B are not independent. For example, observing that B is Heads causes us to increase our belief in A being Heads (in other words $P(a|b) > P(b)$ in the case when a =Heads and b =Heads).

Independence and conditional independence

- In Example 2 the variables A and B are both dependent on a separate variable C, "the coin is biased towards Heads" (which has the values *True* or *False*).
- Although A and B are not independent, it turns out that once we *know* for certain the value of C then any evidence about B cannot change our belief about A. Specifically:

$$P(A|C) = P(A|B,C)$$

- In such a case we say that ***A and B are conditionally independent given C***.

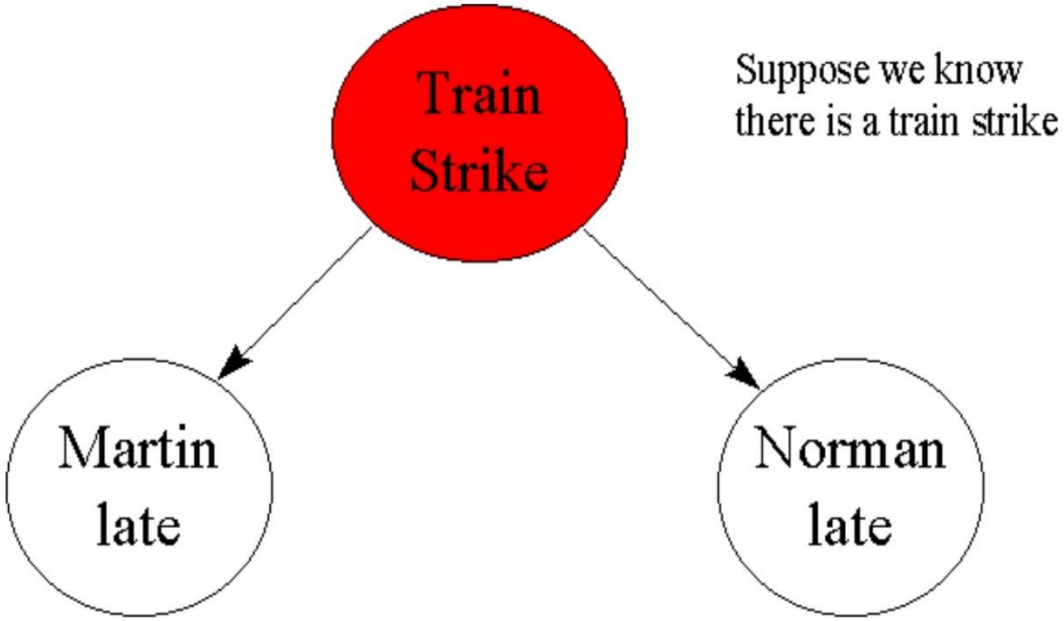
Independence and conditional independence

- In many real life situations variables which are believed to be independent are actually only independent conditional on some other variable.
- **Example 3** Suppose that Norman and Martin live on opposite sides of the City. Norman comes to work by train while Martin drives. Let A represent the variable "Norman late" (which has values *true* or *false*) and similarly let B represent the variable "Martin late".
- Are A and B independent?

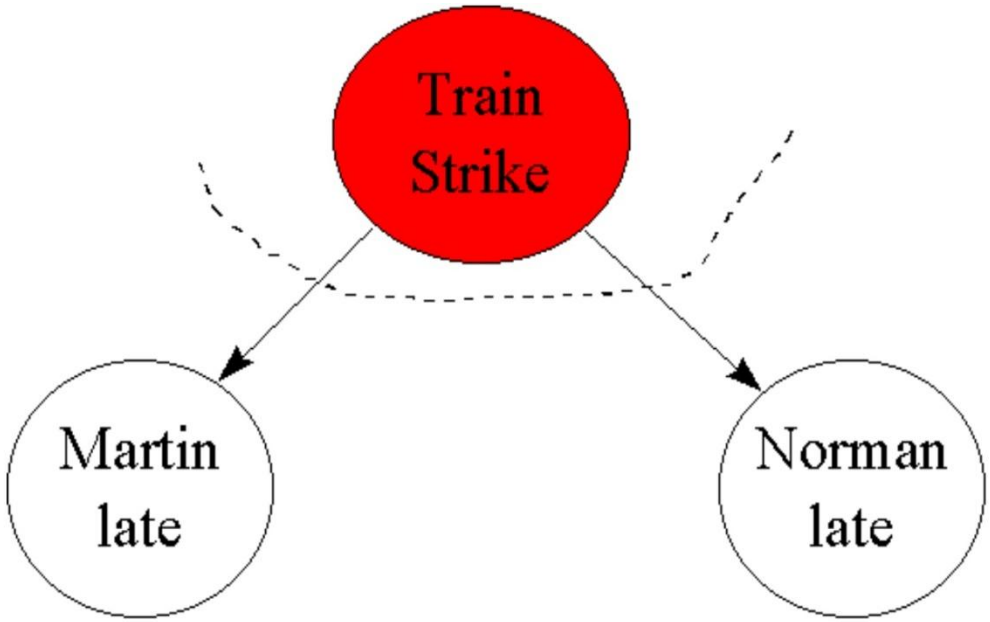
Independence and conditional independence

- However, even if Norman and Martin lived and worked in different countries there may be factors
- such as an international fuel shortage which could mean that A and B are not independent.
- In practice any model of uncertainty should take account of all *reasonable* factors. Thus while, say, a meteorite hitting the Earth might be reasonably excluded it does not seem reasonable to exclude the fact that both A and B may be affected by a Train strike (C).
- Clearly $P(A)$ will increase if C is true; but $P(B)$ will also increase because of extra traffic on the roads. Thus, the situation is represented in the following animation

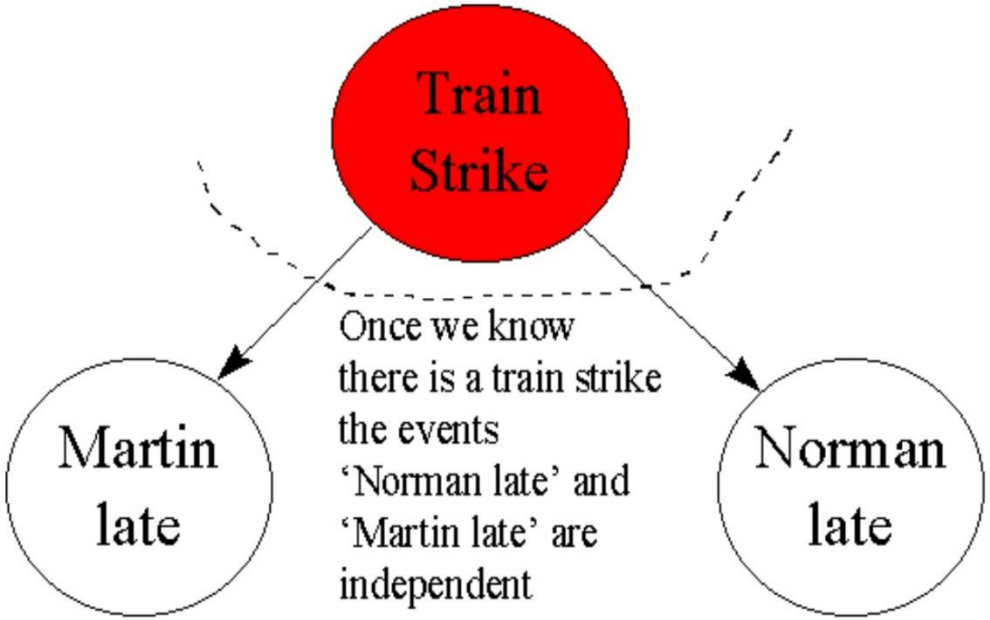
Independence and conditional independence



Independence and conditional independence



Independence and conditional independence



- Now, "Martin late" and "Norman late" are conditionally independent given "Train strike"

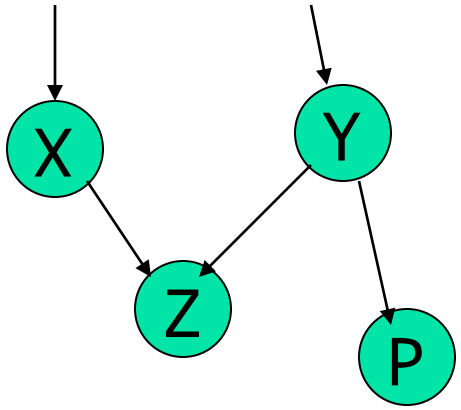
Outline

- Probability theory
- Bayesian approach to Probability theory
- Variables and probability distributions
- Joint events and marginalization
- Conditional Probability
- Bayes Theorem
- Chain Rule
- Independence and Conditional Independence
- Bayesian Belief Networks 

Bayesian Belief Networks (BBN)

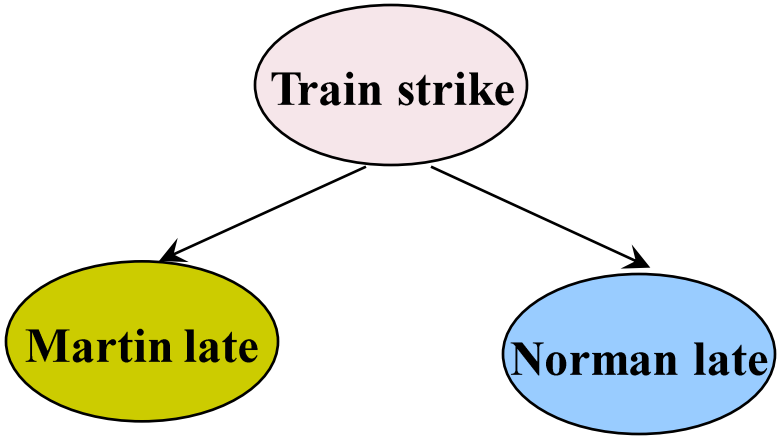
Bayesian Belief Networks

- Bayesian belief networks (also known as Bayesian networks, probabilistic networks): allow *class conditional independencies* between *subsets* of variables
- A (*directed acyclic*) graphical model of causal relationships
  - Represents dependency among the variables
  - Gives a specification of joint probability distribution



- ❑ Nodes: random variables
- ❑ Links: dependency
- ❑ X and Y are the parents of Z, and Y is the parent of P
- ❑ No dependency between Z and P
- ❑ Has no loops/cycles

Bayesian Belief Network: An Example



CPT: Conditional Probability Table

	Train strike	
Martin late	True	False
True	0.6	0.5
False	0.4	0.5

	Train strike	
Normal late	True	False
True	0.8	0.1
False	0.2	0.9

	Train strike
True	0.1
False	0.9

Martin is often late but a train strike only increases the likelihood of his lateness by a small amount. In the event of a train strike Martin is less likely to be late than Norman.

Norman is very unlikely to be late normally (0.1) but if there is a train strike he is very likely to be late (0.8)

Analysing a BBN: entering evidence and propagation

Evidence:

- Having entered the probabilities we can now use Bayesian probability to do various types of analysis. For example, we might want to calculate the (unconditional) probability that Norman is late:

$$P(Norman\ late)$$
$$= P(Norman\ late|train\ strike) * P(train\ strike)$$
$$+ P(Norman\ late)|no\ train\ strike) * P(no\ train\ stike)$$
$$= (0.8 * 0.1) + (0.1 * 0.9) = \mathbf{0.17}$$

- This is called the marginal probability .
- Similarly, we can calculate the marginal probability that Martin is late to be **0.51**.

Analysing a BBN: entering evidence and propagation

Propagation:

- However, the most important use of BBNs is in ***revising* probabilities** in the light of actual observations of events. Suppose, for example, that we *know* there is a train strike. In this case we can **enter the evidence** that 'train strike' = true. The conditional probability tables already tell us the revised probabilities for Norman being late (0.8) and Martin being late (0.6). Suppose, however, that we do not know if there is a train strike but do know that Norman is late. Then we can enter the evidence that 'Norman late' = true and we can use this observation to determine:
  - a) the (revised) probability that there is a train strike; and
  - b) the (revised) probability that Martin will be late.
- To calculate a) we use Bayes theorem :

$$P(train\ strike|Norman\ late) = \frac{P(Norman\ late|train\ strike)*P(train\ strike)}{P(Norman\ late)} = \frac{0.8*0.1}{0.17} = 0.47$$

Analysing a BBN: entering evidence and propagation

- Thus, the observation that Norman is late significantly increases the probability that there is a train strike (up from 0.1 to 0.47).
- Moreover, we can use this revised probability to calculate b):

$$P(Martin\ late)$$
$$= P(Martin\ late|train\ strike) * P(train\ strike)$$
$$+ P(Martin\ late|no\ train\ strike) * P(no\ train\ strike)$$
$$= (0.6 * 0.47) + (0.5 * 0.53) = \mathbf{0.55}$$

- Thus, the observation that Norman is late has slightly increased the probability that Martin is late.
- When we enter evidence and use it to update the probabilities in this way we call it **propagation**.

How evidence is transmitted

- Hard evidence (instantiation):
  - Hard evidence for a node X is evidence that the state of X is definitely a particular value.
  - suppose X represents the result of a particular match for a football team {win, lose, draw}.
  - Then an example of hard evidence would be knowledge that the match is definitely won. In this case we also say X is *instantiated* as the value 'win'.
- Soft evidence:
  - Soft evidence for a node X is any evidence that enables us to update the prior probability values for the states of X. For example, if we know that the team is winning the match 3-0 at half-time, then the probability of *win* would be quite high, while the probabilities of both *lose* and *draw* would be quite low (compared with ignorance prior values).

How evidence is transmitted

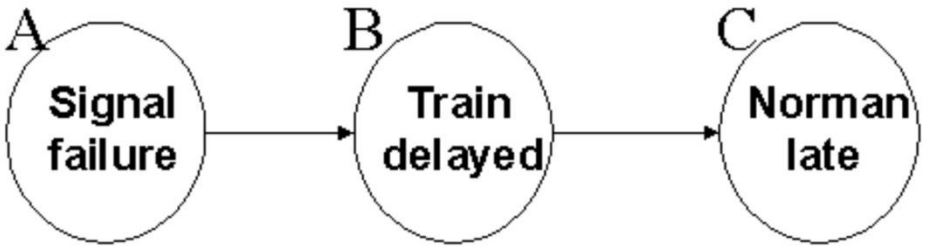
Types of connections to transmit evidence

- Serial connection
- Diverging connection
- Converging connection

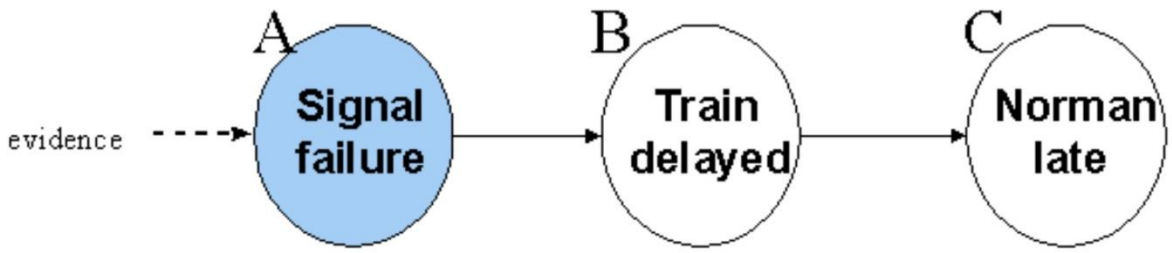
How evidence is transmitted

Serial connection

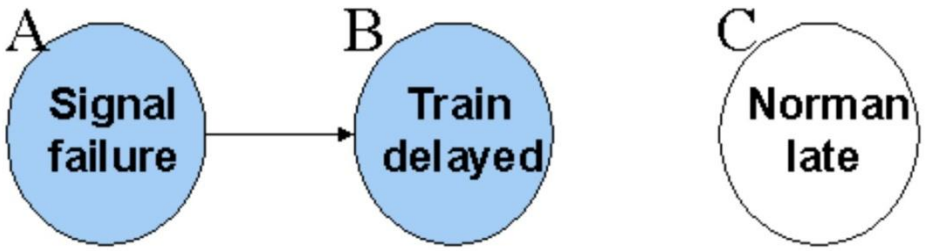
- Suppose we have some evidence that a signal failure has occurred (A). Then clearly this knowledge increases our belief that the train is delayed (B), which in turn increases our belief that Norman is late (C). Thus evidence about A is transmitted through B to C. In general any evidence about A will be transmitted through B to C as shown in the following animation:



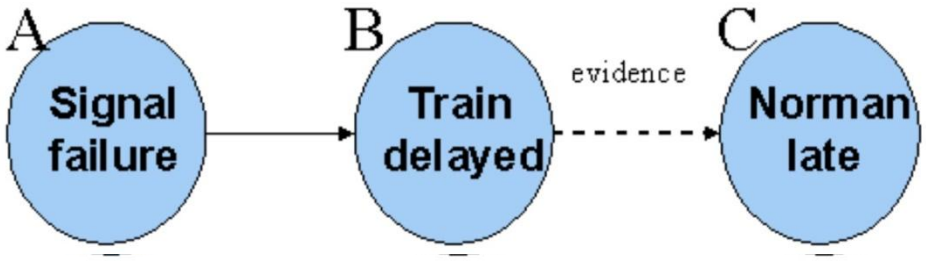
How evidence is transmitted



How evidence is transmitted



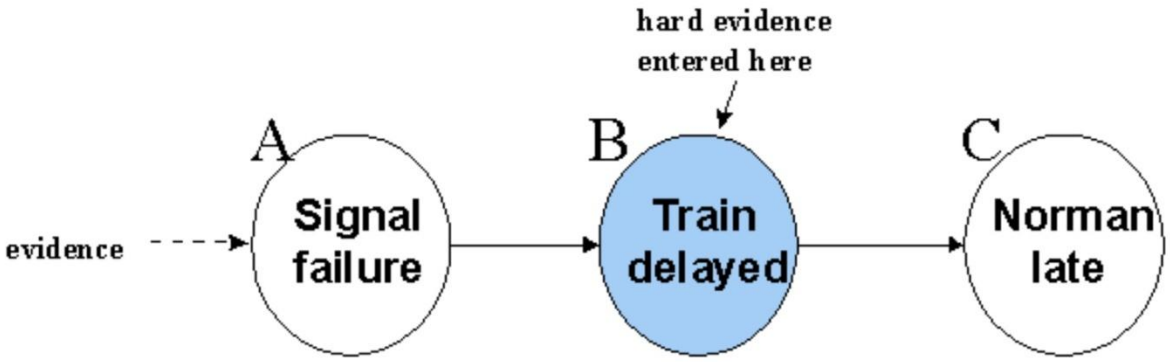
How evidence is transmitted



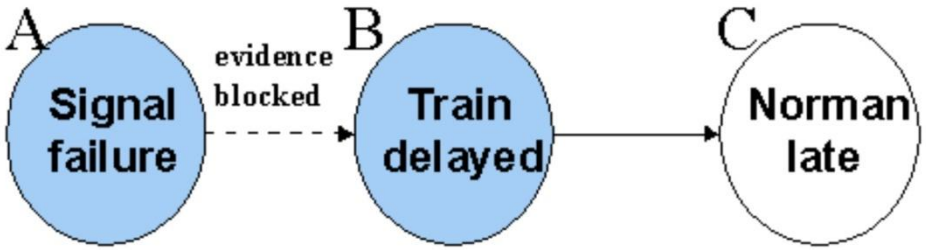
How evidence is transmitted

- However, now suppose that we know the true status of B; for example, suppose we know that the train is delayed (that is, we have hard evidence for B).
- In this case any knowledge about A is irrelevant to C because our knowledge of B essentially 'overwrites it'; any new information about the likelihood of a signal failure (A) is not going to change our belief about Norman being late once we know that the train is delayed.
- In other words the evidence from A cannot be transmitted to C because B blocks the channel as shown in the following animation

How evidence is transmitted



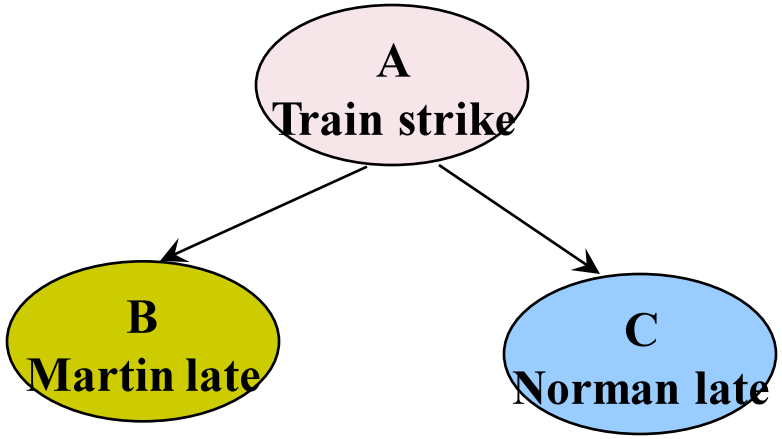
How evidence is transmitted



- In summary, in a serial connection evidence can be transmitted from A to C unless B is instantiated.
- Formally we say that A and C are d-separated given B.

How evidence is transmitted

- Diverging connection:



- Any evidence about A is transmitted to both B and C. For example, if we have evidence which increases our belief in a train strike (A) then this in turn will increase our belief in both Martin being late (B) and Norman being late (C).
- Whether information about B can be transmitted to C (and vice versa)?

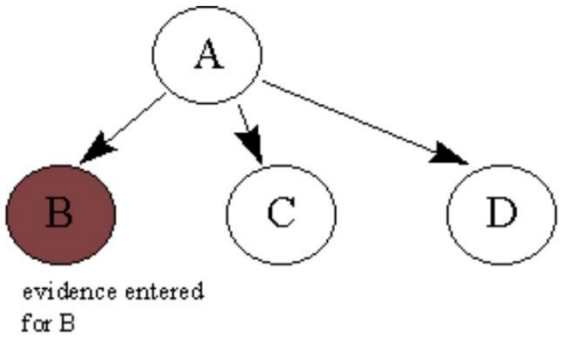
How evidence is transmitted

Diverging connection:

- If there is no hard evidence about A but if we have some evidence about B then this increases our belief about A. This in turn increases our belief about C. In other words, evidence about B (Martin late) is transmitted through to C (Norman late).
- If there is hard evidence about A then in this case, evidence about B does not change in anyway our belief about C. This is because the certainty of A blocks the evidence from being transmitted (it becomes irrelevant once we know A for certain). The value of C is only influenced by the certainty of A. Thus, when A is known for certain, B and C become independent.
- Because the independence of B and C is conditional on the certainty of A, we say formally that B and C are *conditionally independent* (given A).
- In summary: evidence can be transmitted from B to C through a diverging connection A unless A is instantiated. We say that B and C are d-separated given A.

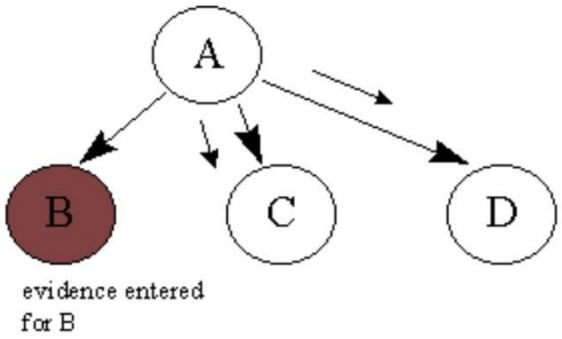
How evidence is transmitted

Diverging connection:



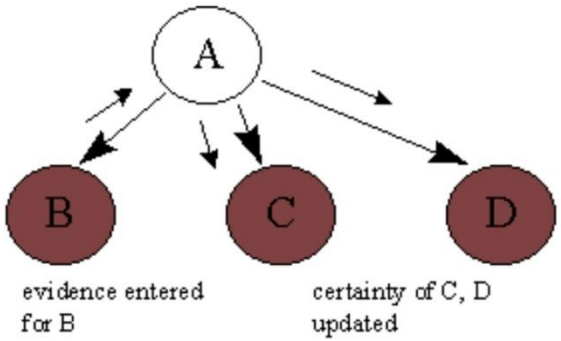
How evidence is transmitted

Diverging connection:



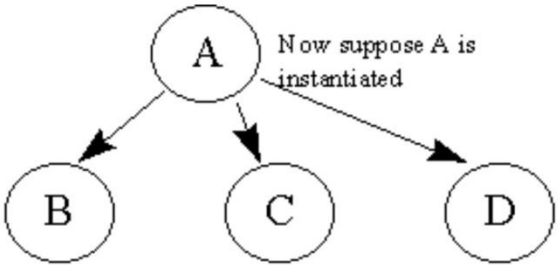
How evidence is transmitted

Diverging connection:



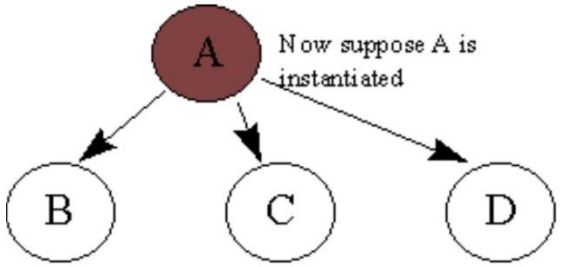
How evidence is transmitted

Diverging connection:



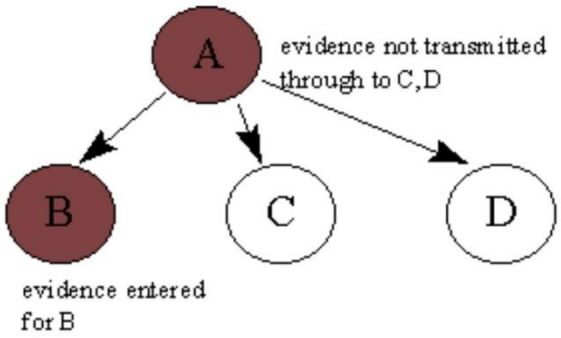
How evidence is transmitted

Diverging connection:



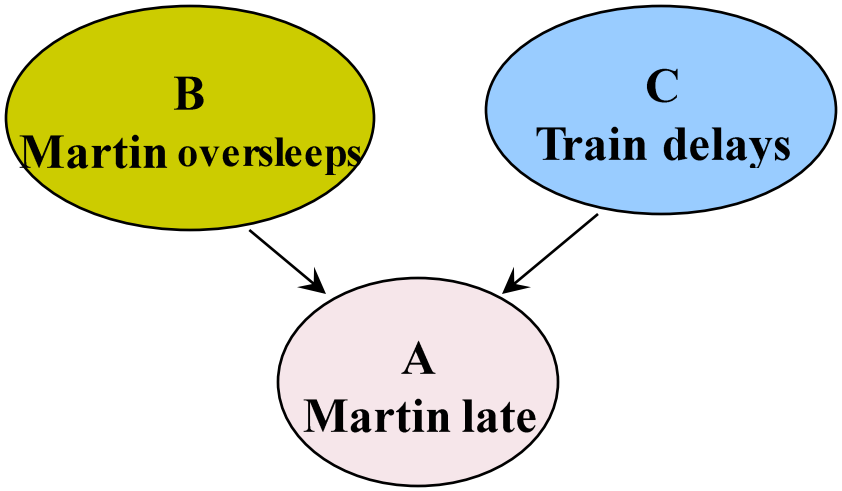
How evidence is transmitted

Diverging connection:



How evidence is transmitted

- Converging connection:



- Whether evidence can be transmitted between B and C?

How evidence is transmitted

Converging connection:

- If A is not known, then parents B and C are independent. So, no evidence is transmitted between them.
- If anything is known about A then parents B and C become dependent.
- Suppose that Martin hangs his coat everyday when he arrives. If coat is not hung by 9:00AM then we can assume that he is late. However, he may not wear coat everyday(soft-evidence).
- Even this 'soft' evidence about Martin being late increases our belief in both B*:Martin oversleeping* and C*:Train delays*. We say that B and C are conditionally dependent (on A).
- It follows that in a converging connection evidence can only be transmitted between the parents B and C when the converging node A has received some evidence (which can be soft or hard).

d-separation

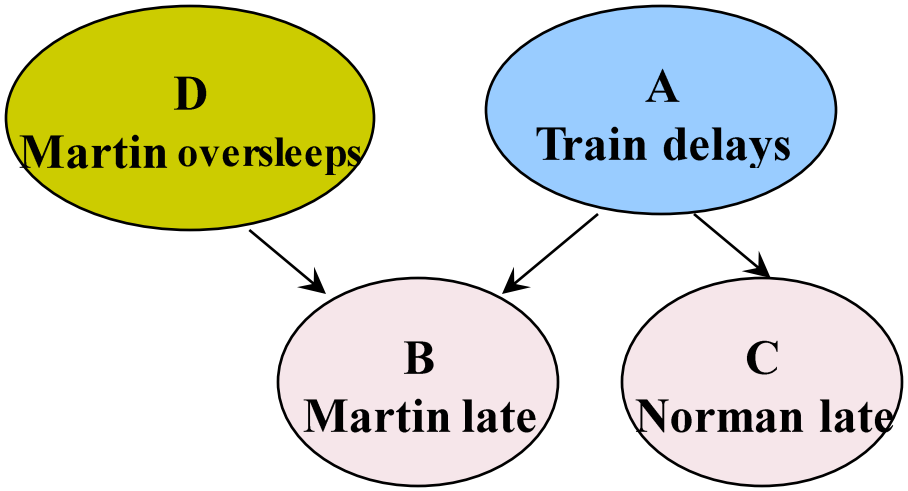
- 1. In a serial connection from B to C via A, evidence from B to C is blocked only when we have hard evidence about A.
- 2. In a diverging connection where B and C have the common parent A, evidence from B to C is blocked only when we have hard evidence about A.
- 3. In a converging connection where A has parents B and C any evidence about A results in evidence transmitted between B and C.
- In cases 1 and 2 we say that the nodes B and C are d-separated when there is hard evidence of A. In case 3 B and C are only d-separated when there is *no* evidence about A.
- In general two nodes which are not d-separated are said to be d-connected.
- These three cases enable us to determine in general whether any two nodes in a given BBN are dependent (d-connected) given the evidence entered in the BBN.

d-separation

Definition of d-separation: Two nodes X and Y in a BBN are *d-separated* if, for all paths between X and Y, there is an intermediate node A for which either:

- 1. the connection is serial or diverging and the state of A is known for certain; or
- 2. the connection is converging and neither A (nor any of its descendants) have received any evidence at all.

Why do we need a BBN for the probability computations?



- For the above case we have to construct a new conditional probability table for node B to reflect the fact that it is conditional on parents A and D.

Why do we need a BBN for the probability computations?

		Martin over sleeps	True		False	
		Train strike	True	False	True	False
Martin late	True		0.8	0.5	0.6	0.5
	False		0.2	0.5	0.4	0.5

CPT for node B (Martin late)

Martin over sleeps	
True	0.4
False	0.6

Table for node D (Martin over sleeps)

Why do we need a BBN for the probability computations?

- We have already seen that in this initialised state the probability that Martin is late is 0.51 and the probability that Norman is late is 0.17.
- Suppose we find out that Martin is late. This evidence increases our belief in both of the possible causes (namely a train strike A and Martin oversleeping B). Specifically, applying Bayes theorem yields a revised probability of A of 0.13 (up from the prior probability of 0.1) and a revised probability of D of 0.41 (up from the prior probability of 0.4). However, if we had to bet on it, our money would be firmly on Martin oversleeping as the more likely cause. Now suppose we also discover that Norman is late. Entering this evidence and applying Bayes yields a revised probability of 0.54 for a train strike and 0.44 for Martin oversleeping. Thus the odds are that the train strike, rather than oversleeping, have caused Martin to be late. We say that Martin's lateness has been 'explained away'.
-

Why do we need a BBN for the probability computations?

- BBNs make explicit the dependencies between different variables. In general there may be relatively few direct dependencies (modelled by arcs between nodes of the network) and this means that many of the variables are conditionally independent .
- In the simple example (previous slide) the nodes 'Norman late' and 'Martin late' are conditionally independent (there is no arc linking them); once the value of 'train strike' is known knowledge of 'Norman late' does not effect the probability of 'Martin late' and vice versa.
- The existence of unlinked (conditionally independent) nodes in a network drastically reduces the computations necessary to work out all the probabilities we require. In general, all the probabilities can be computed from the joint probability distribution . Crucially, this joint probability distribution is far simpler to compute when there are conditionally independent nodes.

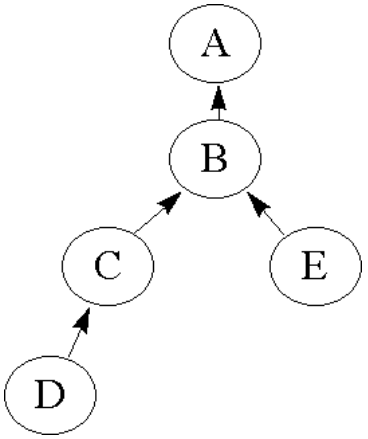
Why do we need a BBN for the probability computations?

- Suppose, for example, that we have a network consisting of five variables (nodes) A,B,C,D,E.
- If we do not specify the dependencies explicitly then we are essentially assuming that all the variables are dependent on each other.
- The chain rule enables us to calculate the joint probability distribution $P(A, B, C, D, E)$ as

$$P(A, B, C, D, E) = P(A|B, C, D, E) * P(B|C, D, E) * P(C|D, E) * P(D|E) * P(E)$$

Why do we need a BBN for the probability computations?

- However, suppose that the dependencies are explicitly modelled in a BBN as:



- Then the joint probability distribution p(A,B,C,D,E) is much simplified:

$$P(A,B,C,D,E) = P(A|B) * P(B|C,E) * P(C|D) * P(D) * P(E)$$

- The generalized form is

$$P(A_1,\ldots,A_n) = \prod_{i=1}^{n} P(A_i \mid Parents(A_i))$$

Why should we use BBNs?

- BBNs on their own enable us to model uncertain events and arguments about them. The intuitive visual representation can be very useful in clarifying previously opaque assumptions or reasonings hidden in the head of an expert. With BBNs, it is possible to articulate expert beliefs about the dependencies between different variables and BBNs allow an injection of scientific rigour when the probability distributions associated with individual nodes are simply 'expert opinions'.

Training Bayesian Networks: Several Scenarios

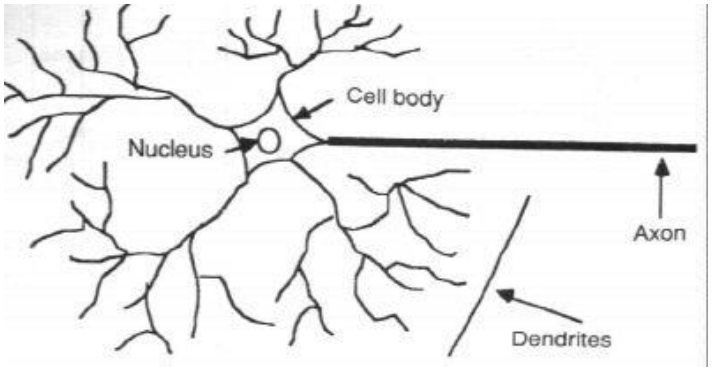
- Scenario 1: Given both the network structure and all variables observable: *compute only the CPT entries*
- Scenario 2: Network structure known, some variables hidden: *gradient descent* (greedy hill-climbing) method, i.e., search for a solution along the steepest descent of a criterion function
  - Weights are initialized to random probability values
  - At each iteration, it moves towards what appears to be the best solution at the moment, w.o. backtracking
  - Weights are updated at each iteration & converge to local optimum
- Scenario 3: Network structure unknown, all variables observable: search through the model space to *reconstruct network topology*
- Scenario 4: Unknown structure, all hidden variables: No good algorithms known for this purpose
- D. Heckerman. A Tutorial on Learning with Bayesian Networks. In *Learning in Graphical Models,* M. Jordan, ed.. MIT Press, 1999.

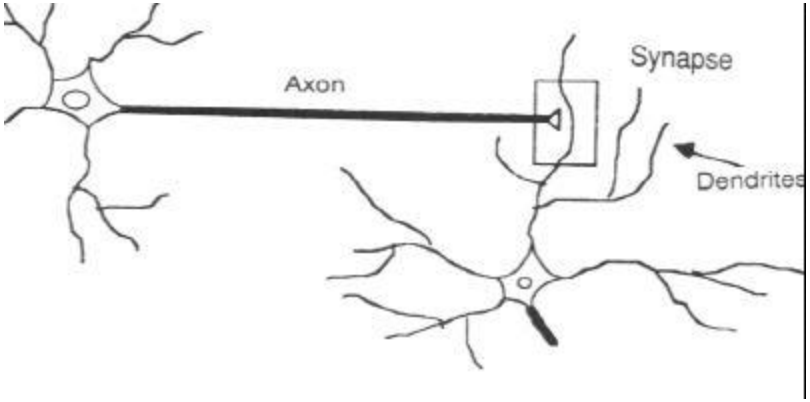
Artificial Neural Networks

Background

- 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 connections that exist between the neurons. This is true of NNs as well.

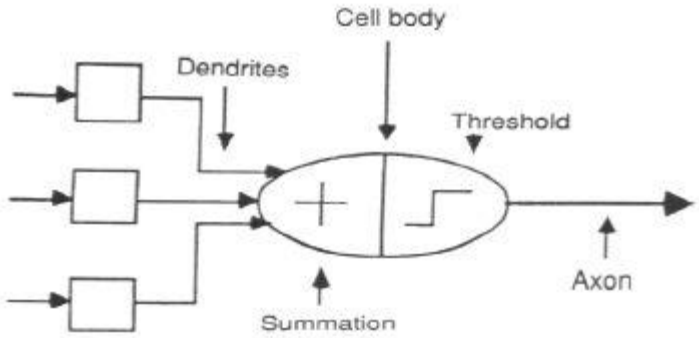
How the Human Brain learns





- 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 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 changes.
- 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

Artificial neurons

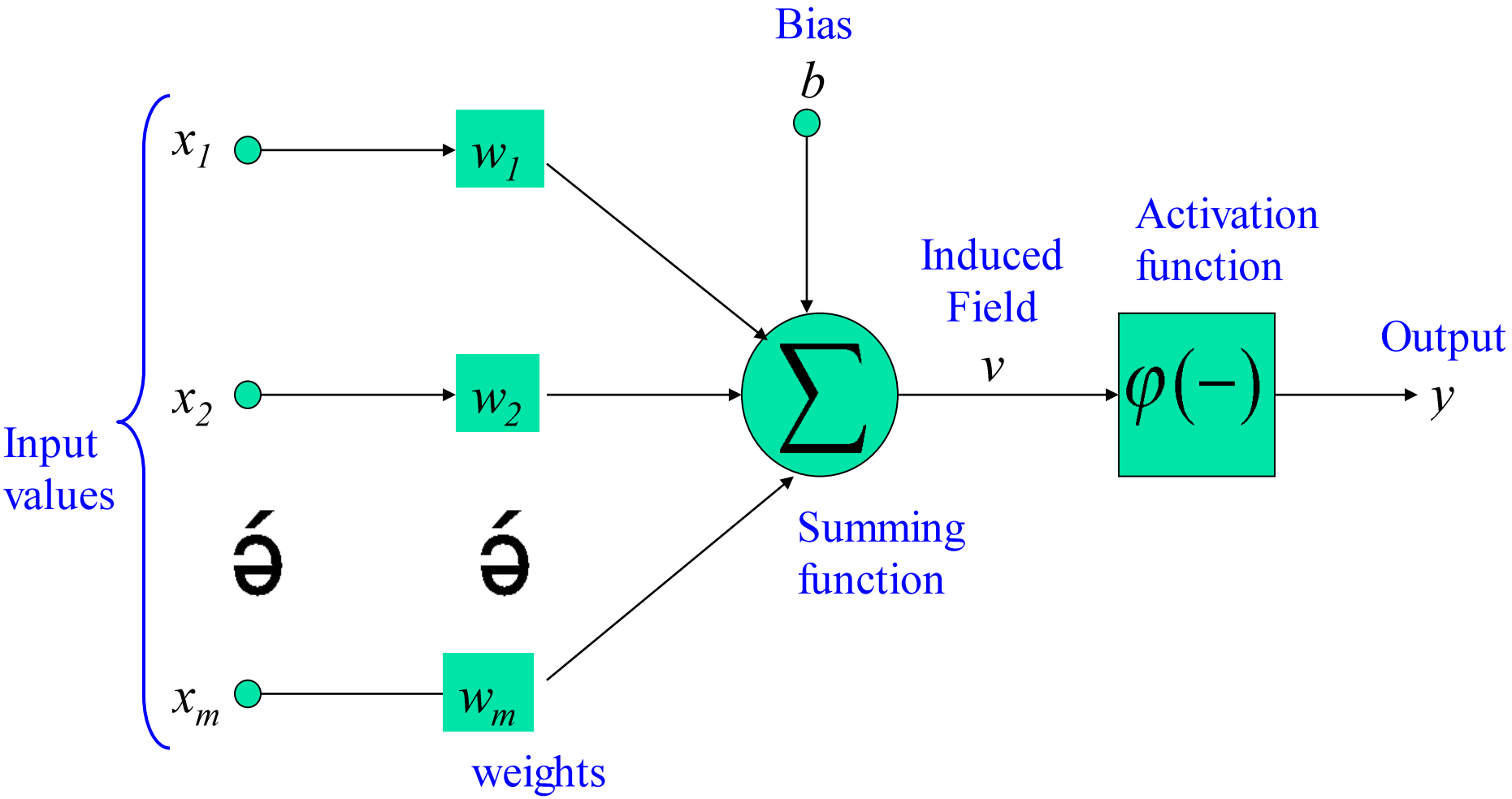
- The neuron is the basic information processing unit of a NN. It consists of:
  1. A set of links, describing the neuron inputs, with weights $W_1, W_2, \ldots, W_m$
  2. An adder function (linear combiner) for computing the weighted sum of the inputs:
     (real numbers)

$$u = \sum_{j=1}^{m} w_j x_j$$

  3. Activation function φ for limiting the amplitude of the neuron output. Here 'b' denotes bias.

$$y = \varphi(u + b)$$

Artificial neurons



Bias of a Neuron

- The bias *b* has the effect of applying a transformation to the weighted sum *u*

$$v = u + b$$

- The bias is an external parameter of the neuron. It can be modeled by adding an extra input.
- *v* is called induced field of the neuron

$$v = \sum_{j=0}^{m} w_j x_j$$

$$w_0 = b$$

Neuron Models

- The choice of activation function φ determines the neuron model.

Examples:

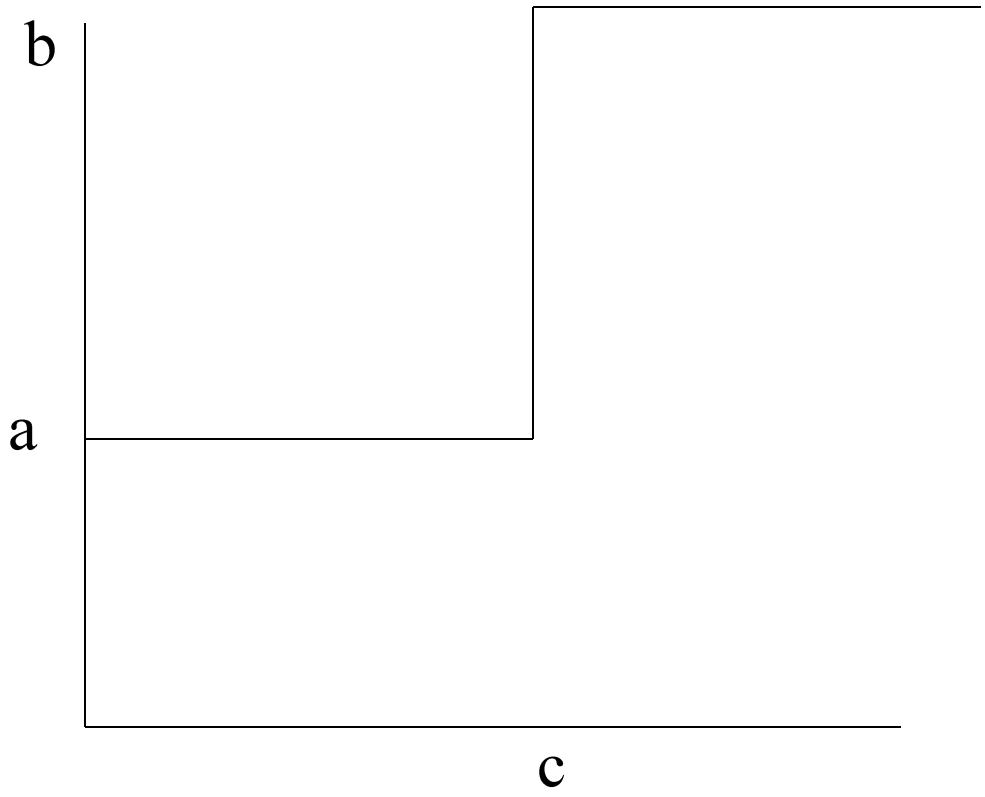
- step function: $$\varphi(v) = \begin{cases} a & \text{if } v < c \\ b & \text{if } v > c \end{cases}$$

- ramp function: $$\varphi(v) = \begin{cases} a & \text{if } v < c \\ b & \text{if } v > d \\ a + ((v-c)(b-a)/(d-c)) & \text{otherwise} \end{cases}$$

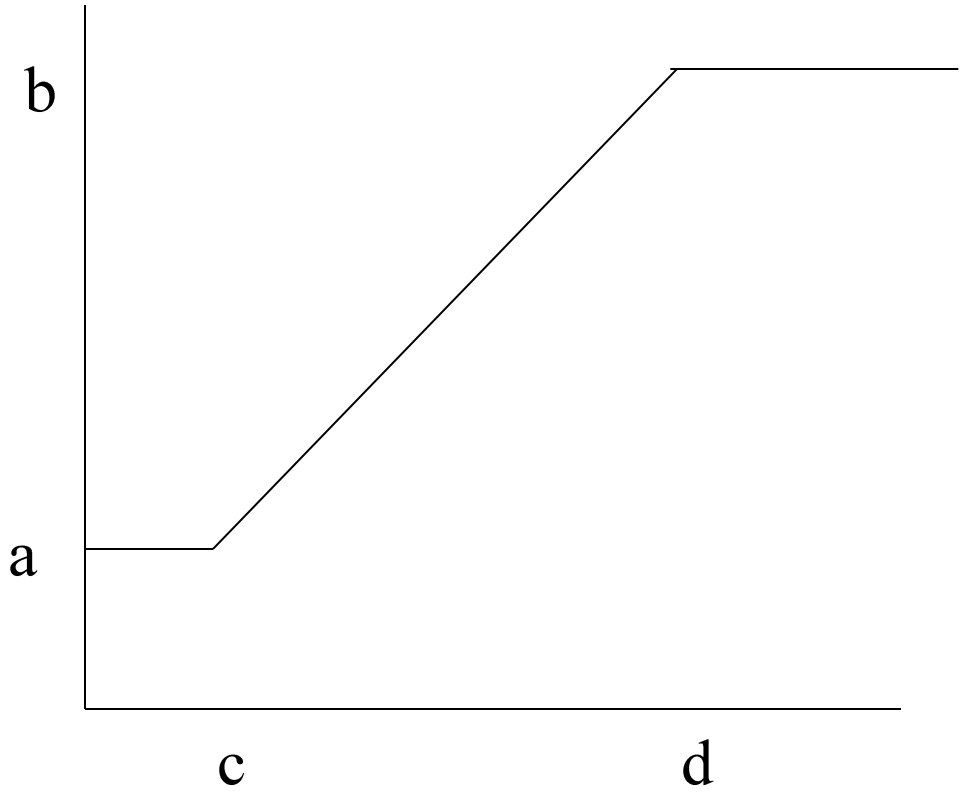
- sigmoid function with z,x,y parameters $$\varphi(v) = z + \frac{1}{1 + \exp(-xv + y)}$$

- Gaussian function: $$\varphi(v) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2}\left(\frac{v-\mu}{\sigma} \right)^2 \right)$$

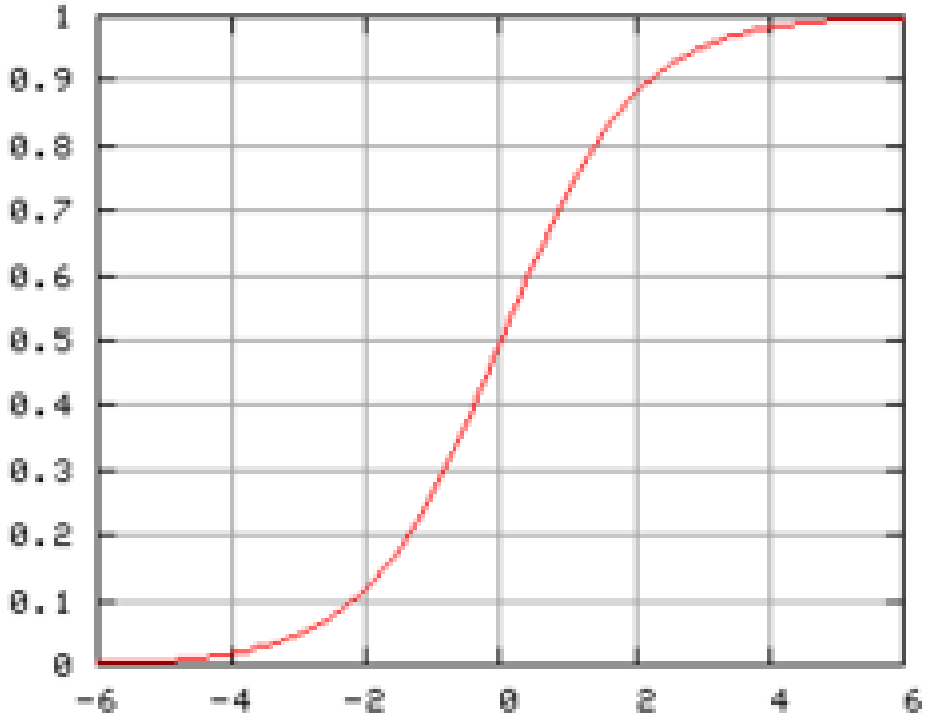
Step Function



Ramp Function



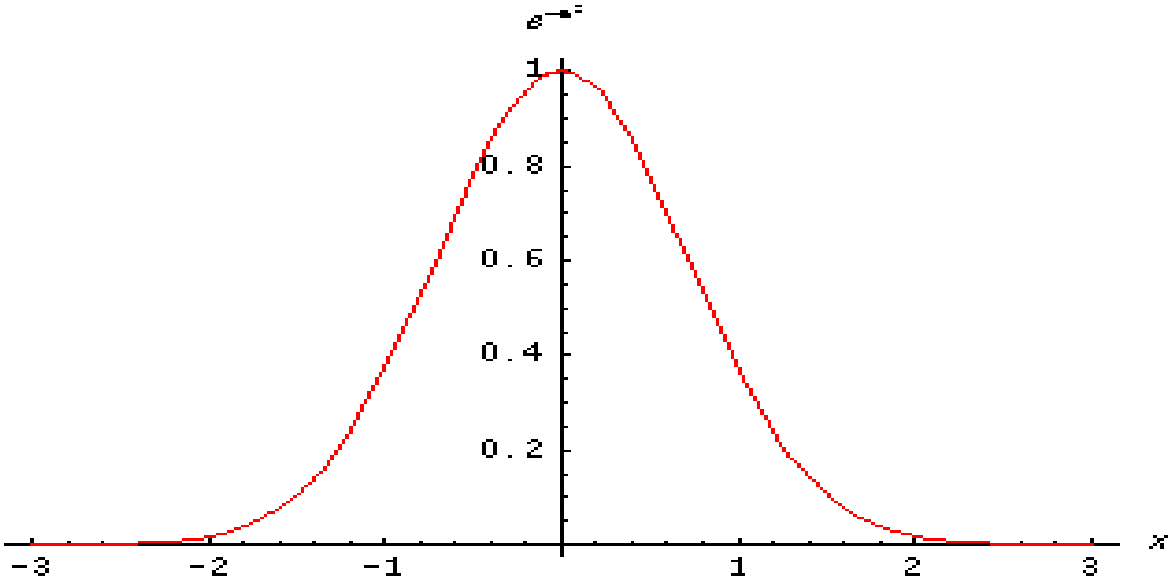
Sigmoid function



Gaussian function

- The **Gaussian function** is the probability function of the normal distribution. Sometimes also called the frequency curve.

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/2\sigma^2},$$

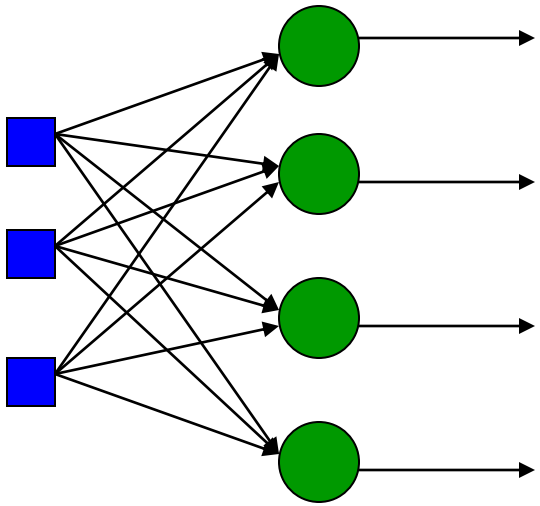


Network Architectures

- Three different classes of network architectures
  - single-layer feed-forward
  - multi-layer feed-forward
  - recurrent
- The architecture of a neural network is linked with the learning algorithm used to train

Single Layer Feed-forward

Input layer of source nodes

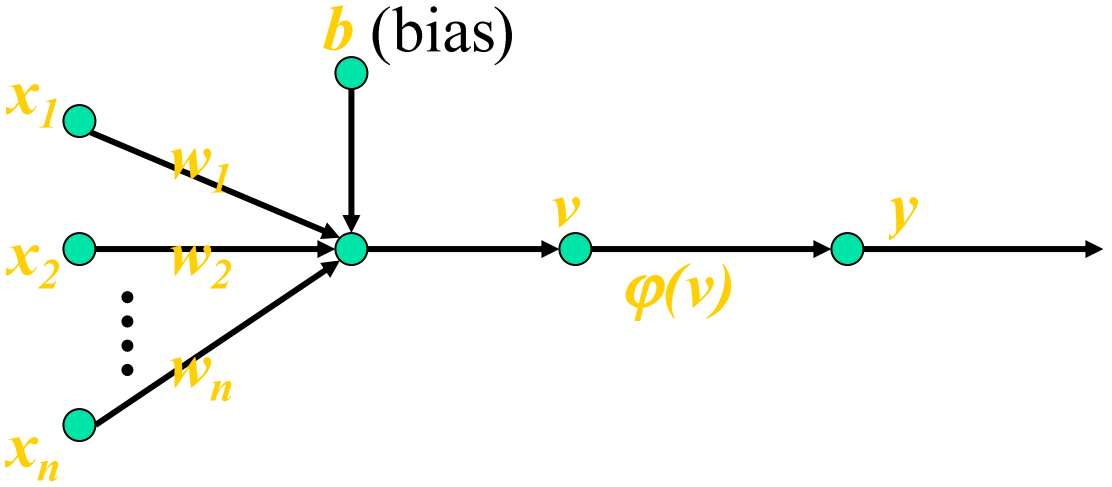


Output layer of neurons

Perceptron: Neuron Model
(Special form of single layer feed forward)

- The perceptron was first proposed by Rosenblatt (1958) is a simple neuron that is used to classify its input into one of two categories.
- A perceptron uses a step function that returns +1 if weighted sum of its input ≥ 0 and -1 otherwise

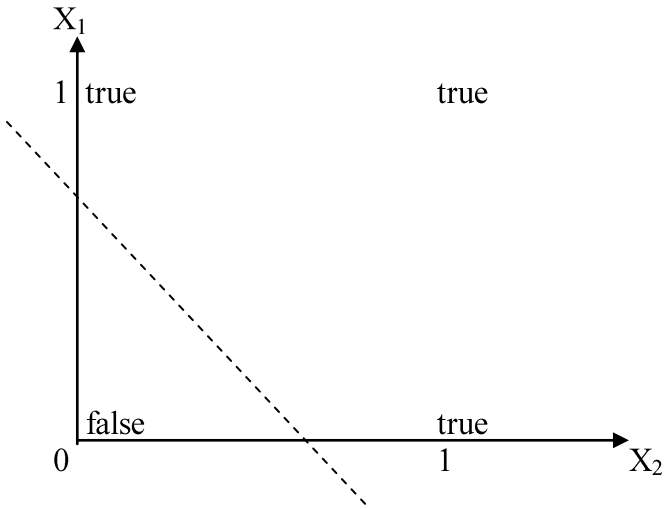
$$\varphi(v) = \begin{cases} +1 \text{ if } v \geq 0 \\ -1 \text{ if } v < 0 \end{cases}$$



Perceptron for Classification

- The perceptron is used for binary classification.
- First train a perceptron for a classification task.
  - Find suitable weights in such a way that the training examples are correctly classified.
  - Geometrically try to find a hyper-plane that separates the examples of the two classes.
- The perceptron can only model linearly separable classes.
- When the two classes are not linearly separable, it may be desirable to obtain a linear separator that minimizes the mean squared error.
- Given training examples of classes C_1, C_2 train the perceptron in such a way that :
  - *If the output of the perceptron is +1 then the input is assigned to class C_1*
  - *If the output is -1 then the input is assigned to C_2*

Boolean function OR – Linearly separable



Learning Process for Perceptron

- Initially assign random weights to inputs between -0.5 and +0.5
- Training data is presented to perceptron and its output is observed.
- If output is incorrect, the weights are adjusted accordingly using following formula.

$$wi \leftarrow wi + (a * xi * e)$$, where 'e' is error produced and 'a' ($-1 < a < 1$) is learning rate

  - 'a' is defined as 0 if output is correct, it is +ve, if output is too low and –ve, if output is too high.
  - Once the modification to weights has taken place, the next piece of training data is used in the same way.
  - Once all the training data have been applied, the process starts again until all the weights are correct and all errors are zero.
  - Each iteration of this process is known as an epoch.

Example: Perceptron to learn OR function

- Initially consider w1 = -0.2 and w2 = 0.4
- Training data say, x1 = 0 and x2 = 0, output is 0.
- Compute y = Step(w1*x1 + w2*x2) = 0. Output is correct so weights are not changed.
- For training data x1=0 and x2 = 1, output is 1
- Compute y = Step(w1*x1 + w2*x2) = 0.4 = 1. Output is correct so weights are not changed.
- Next training data x1=1 and x2 = 0 and output is 1
- Compute y = Step(w1*x1 + w2*x2) = - 0.2 = 0. Output is incorrect, hence weights are to be changed.
- Assume a = 0.2 and error e=1

  wi = wi + (a * xi * e) gives w1 = 0 and w2 =0.4
- With these weights, test the remaining test data.
- Repeat the process till we get stable result.

Perceptron: Limitations

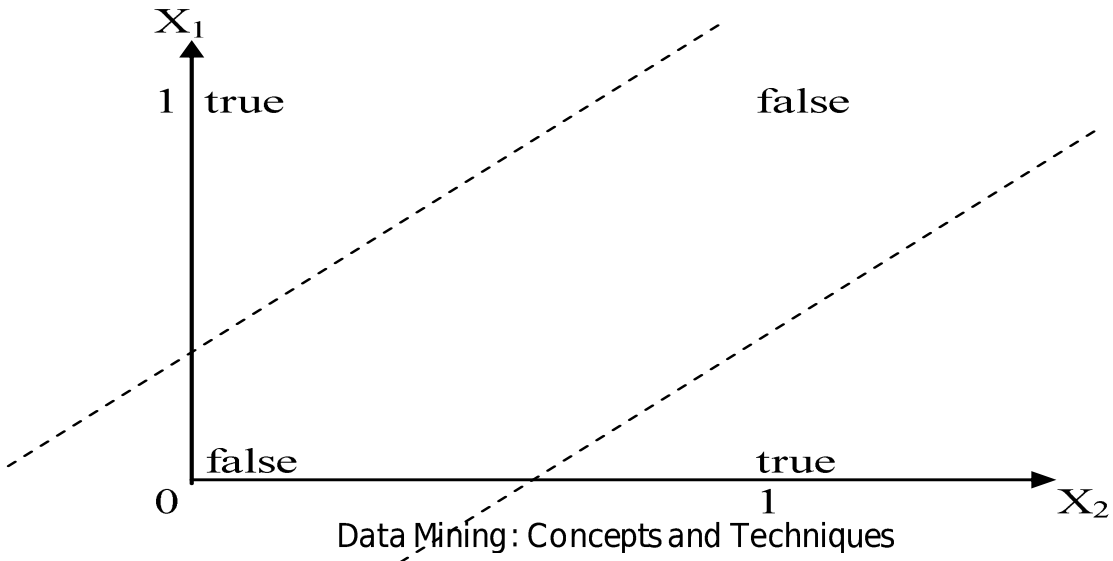
- The perceptron can only model linearly separable functions,
  - those functions which can be drawn in 2-dim graph and single straight line separates values in two part.
- Boolean functions given below are linearly separable:
  - AND
  - OR
  - COMPLEMENT
- It cannot model XOR function as it is non linearly separable.
  - When the two classes are not linearly separable, it may be desirable to obtain a linear separator that minimizes the mean squared error.

XOR – Non linearly separable function

- A typical example of non-linearly separable function is the XOR that computes the logical exclusive or..
- This function takes two input arguments with values in {0,1} and returns one output in {0,1},
- Here 0 and 1 are encoding of the truth values false and true,
- The output is true if and only if the two inputs have different truth values.
- XOR is non linearly separable function which can not be modeled by perceptron.
- For such functions we have to use multi layer feed-forward network.

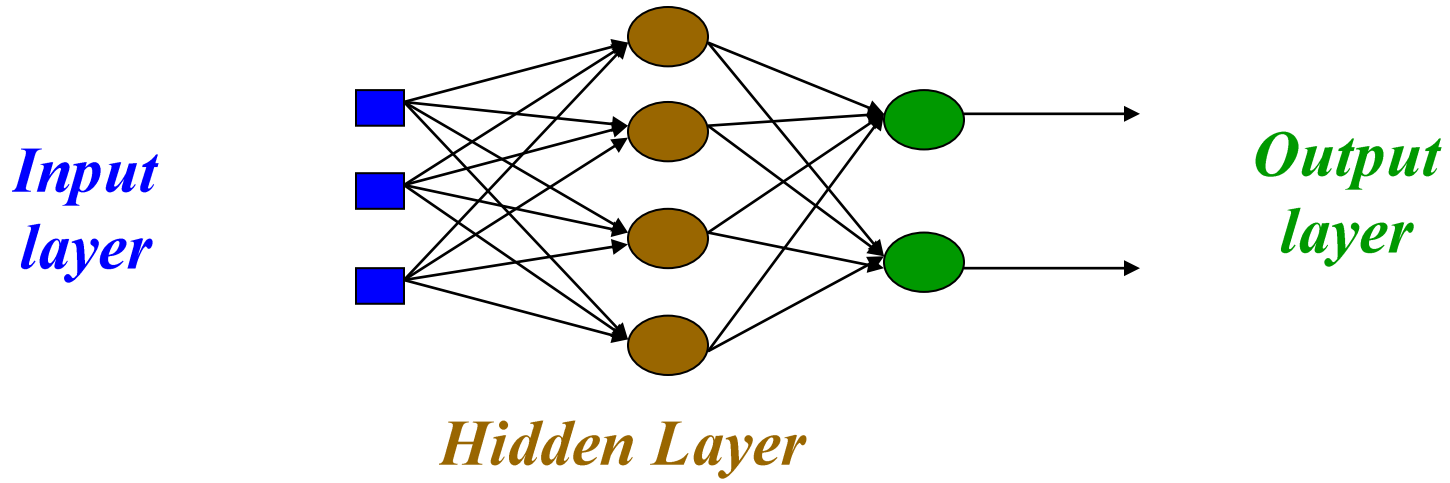
Input		Output
X_1	X_2	X_1 XOR X_2
0	0	0
0	1	1
1	0	1
1	1	0

- These two classes (true and false) cannot be separated using a line. Hence XOR is non linearly separable.



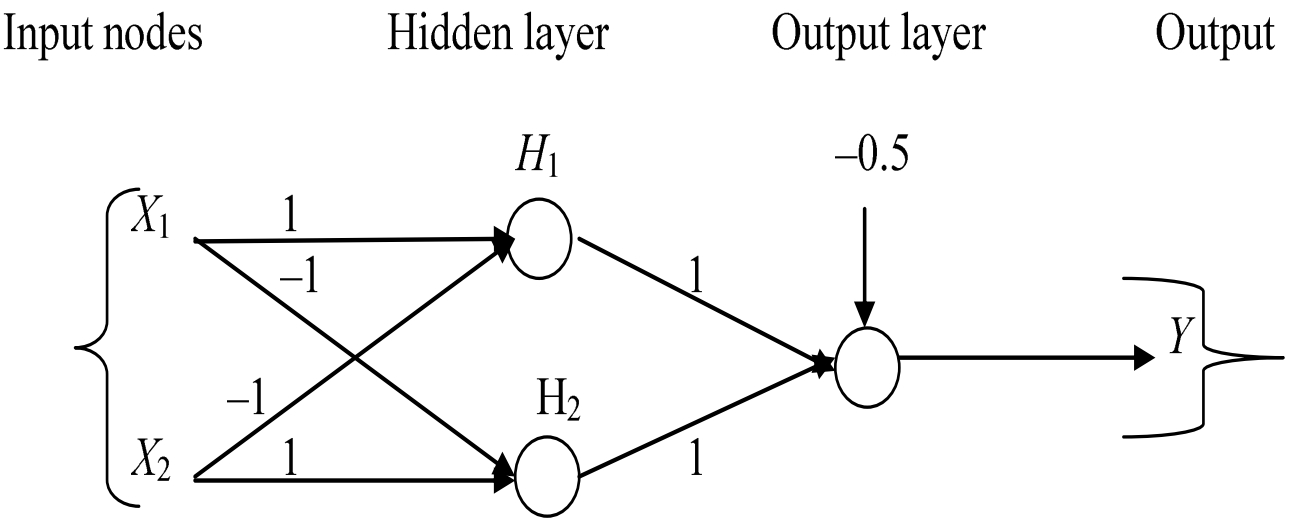
Multi layer feed-forward NN (FFNN)

- FFNN is a more general network architecture, where there are hidden layers between input and output layers.
- Hidden nodes do not directly receive inputs nor send outputs to the external environment.
- FFNNs overcome the limitation of single-layer NN.
- They can handle non-linearly separable learning tasks.



FFNN for XOR

- The ANN for XOR has two hidden nodes that realizes this non-linear separation and uses the sign (step) activation function.
- Arrows from input nodes to two hidden nodes indicate the directions of the weight vectors (1,-1) and (-1,1).
- The output node is used to combine the outputs of the two hidden nodes.



Inputs		Output of Hidden Nodes		Output Node	X_1 XOR X_2
X_1	X_2	H_1	H_2		
0	0	0	0	–0.5 → 0	0
0	1	–1 → 0	1	0.5 → 1	1
1	0	1	–1 → 0	0.5 → 1	1
1	1	0	0	–0.5 → 0	0

Since we are representing two states by 0 (false) and 1 (true), we will map negative outputs (–1, –0.5) of hidden and output layers to 0 and positive output (0.5) to 1.

FFNN NEURON MODEL

- The classical learning algorithm of FFNN is based on the gradient descent method.
- For this reason the activation function used in FFNN are continuous functions of the weights, differentiable everywhere.
- The activation function for node i may be defined as a simple form of the sigmoid function in the following manner:

$$\varphi(Vi) = \frac{1}{1+e^{(-A*Vi)}}$$

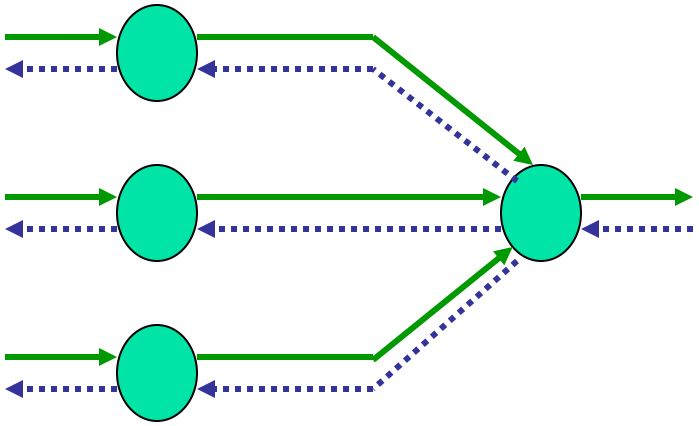
where $A > 0$, $V_i = \sum W_{ij} * Y_j$, such that W_{ij} is a weight of the link from node i to node j and Y_j is the output of node j.

Training Algorithm: Backpropagation

- The Backpropagation algorithm learns in the same way as single perceptron.
- It searches for weight values that minimize the total error of the network over the set of training examples (training set).
- Backpropagation consists of the repeated application of the following two passes:
  - Forward pass: In this step, the network is activated on one example and the error of (each neuron of) the output layer is computed.
  - Backward pass: in this step the network error is used for updating the weights. The error is propagated backwards from the output layer through the network layer by layer. This is done by recursively computing the local gradient of each neuron.

Backpropagation

- Back-propagation training algorithm



Network activation
Forward Step

Error propagation
Backward Step

- Backpropagation adjusts the weights of the NN in order to minimize the network total mean squared error.

Contd..

- Consider a network of three layers.
- Let us use i to represent nodes in input layer, j to represent nodes in hidden layer and k represent nodes in output layer.
- w_{ij} refers to weight of connection between a node in input layer and node in hidden layer.
- The following equation is used to derive the output value Yj of node j

$$\mathbf{Yj} = \frac{1}{1+e^{-X_j}}$$

where, $X_j = \sum x_i \,.\, w_{ij} - \theta_j$, $1 \leq i \leq n$; n is the number of inputs to node j, and θ_j is threshold for node j

Total Mean Squared Error

- The error of output neuron *k* after the activation of the network on the *n-th* training example (x(n), d(n)) is:

$$e_k(n) = d_k(n) - y_k(n)$$

- The network error is the sum of the squared errors of the output neurons:

$$E(n) = \sum e_k^2(n)$$

- The total mean squared error is the average of the network errors of the training examples.

$$E_{AV} = \frac{1}{N} \sum_{n=1}^{N} E(n)$$

Weight Update Rule

- The Backprop weight update rule is based on the gradient descent method:
  - It takes a step in the direction yielding the maximum decrease of the network error E.
  - This direction is the opposite of the gradient of E.
- Iteration of the Backprop algorithm is usually terminated when the sum of squares of errors of the output values for all training data in an epoch is less than some threshold such as 0.01

$$w_{ij} = w_{ij} + \Delta w_{ij} \qquad \Delta \mathrm{w}_{\mathrm{ij}} = -\eta \frac{\partial E}{\partial \mathrm{w}_{\mathrm{ij}}}$$

Backprop learning algorithm (incremental-mode)

n=1;
initialize weights randomly;
while (stopping criterion not satisfied or n <max_iterations)
    for each example (x,*d*)
    - run the network with input x and compute the output y
    - update the weights in backward order starting from those of the output layer:

$$w_{ji} = w_{ji} + \Delta w_{ji}$$

    with Δw_{ji} computed using the (generalized) Delta rule
    end-for
    n = n+1;
end-while;

Stopping criterions

- Total mean squared error change:
  - Back-prop is considered to have converged when the absolute rate of change in the average squared error per epoch is sufficiently small (in the range [0.1, 0.01]).
- Generalization based criterion:
  - After each epoch, the NN is tested for generalization.
  - If the generalization performance is adequate then stop.
  - If this stopping criterion is used then the part of the training set used for testing the network generalization will not used for updating the weights.

NN DESIGN ISSUES

- Data representation
- Network Topology
- Network Parameters
- Training
- Validation

Data Representation

- Data representation depends on the problem.
- In general ANNs work on continuous (real valued) attributes. Therefore symbolic attributes are encoded into continuous ones.
- Attributes of different types may have different ranges of values which affect the training process.
- Normalization may be used, like the following one which scales each attribute to assume values between 0 and 1.

$$x_i = \frac{x_i - \min_i}{\max_i - \min_i}$$

for each value x_i of i^{th} attribute, $\min_i$ and $\max_i$ are the minimum and maximum value of that attribute over the training set.

Network Topology

- The number of layers and neurons depend on the specific task.
- In practice this issue is solved by trial and error.
- Two types of adaptive algorithms can be used:
  - start from a large network and successively remove some neurons and links until network performance degrades.
  - begin with a small network and introduce new neurons until performance is satisfactory.

Network parameters

- How are the weights initialized?
- How is the learning rate chosen?
- How many hidden layers and how many neurons?
- How many examples in the training set?

Initialization of weights

- In general, initial weights are randomly chosen, with typical values between -1.0 and 1.0 or -0.5 and 0.5.
- If some inputs are much larger than others, random initialization may bias the network to give much more importance to larger inputs.
- In such a case, weights can be initialized as follows:

$$w_{ij} = \pm \frac{1}{2N} \sum_{i=1,\ldots,N} \frac{1}{|x_i|}$$

For weights from the input to the first layer

$$w_{jk} = \pm \frac{1}{2N} \sum_{i=1,\ldots,N} \frac{1}{\varphi(\sum w_{ij} x_i)}$$

For weights from the first to the second layer

Choice of learning rate

- The right value of η depends on the application.
- Values between 0.1 and 0.9 have been used in many applications.
- Other heuristics is that adapt η during the training as described in previous slides.

Training

- Rule of thumb:
  - the number of training examples should be at least five to ten times the number of weights of the network.
- Other rule:

$$N > \frac{|W|}{(1-a)}$$

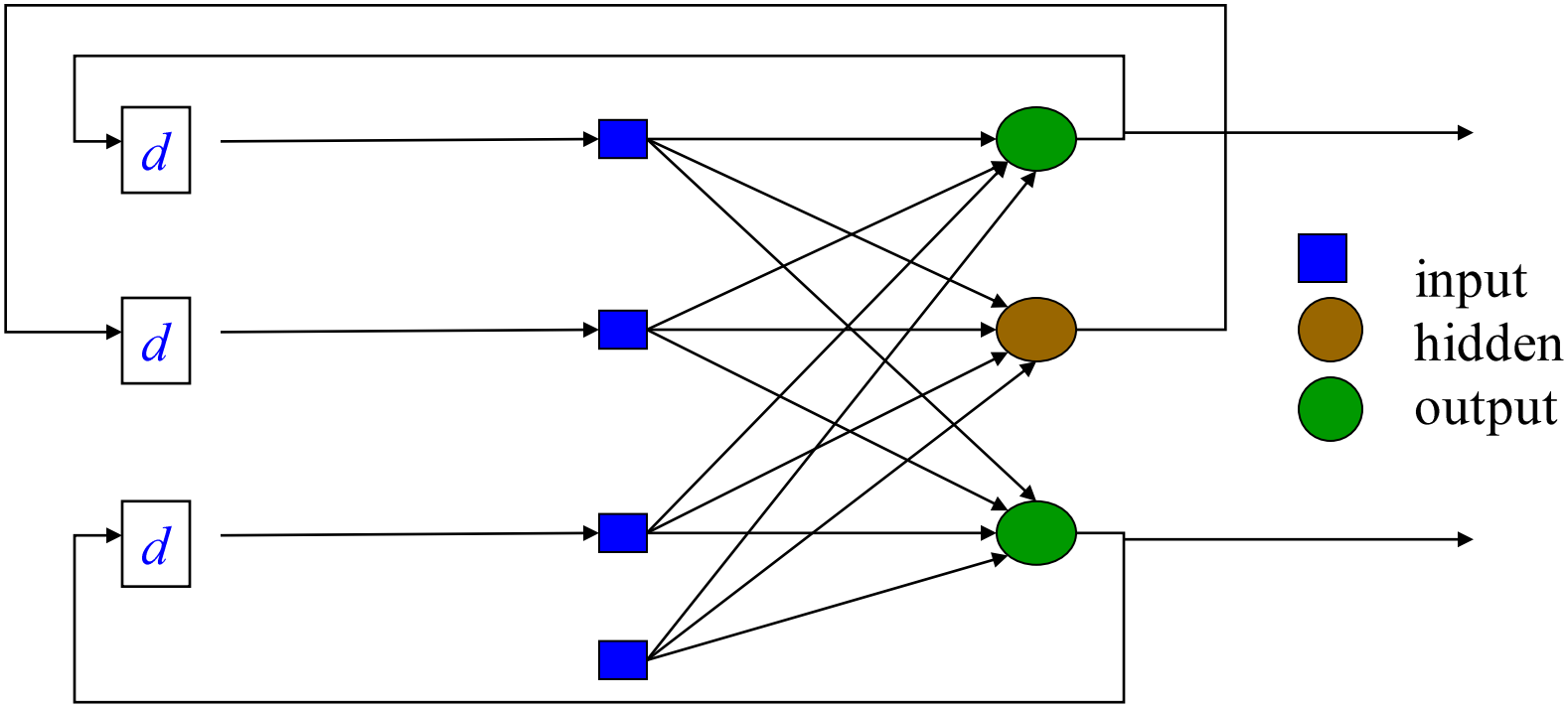
|W|= number of weights
a=expected accuracy on test set

Recurrent Network

- FFNN is acyclic where data passes from input to the output nodes and not vice versa.
  - Once the FFNN is trained, its state is fixed and does not alter as new data is presented to it. It does not have memory.
- Recurrent network can have connections that go backward from output to input nodes and models dynamic systems.
  - In this way, a recurrent network's internal state can be altered as sets of input data are presented. It can be said to have memory.
  - It is useful in solving problems where the solution depends not just on the current inputs but on all previous inputs.
- Applications
  - predict stock market price,
  - weather forecast

Recurrent Network Architecture

- Recurrent Network with *hidden neuron*: unit delay operator *d* is used to model a dynamic system



Learning and Training

- During learning phase,
  - a recurrent network feeds its inputs through the network, including feeding data back from outputs to inputs
  - process is repeated until the values of the outputs do not change.
- This state is called equilibrium or stability
- Recurrent networks can be trained by using back-propagation algorithm.
- In this method, at each step, the activation of the output is compared with the desired activation and errors are propagated backward through the network.
- Once this training process is completed, the network becomes capable of performing a sequence of actions.

Thank you