Unit-3

Knowledge Representation & Reasoning: Propositional logic, Theory of first order logic, Inference in First order logic, Forward & Backward chaining, Resolution, Probabilistic reasoning, Utility theory, Hidden Markov Models (HMM), Bayesian Networks.

3.1 Knowledge Representation & Reasoning

Knowledge-representation is the field of artificial intelligence that focuses on designing computer representations that capture information about the world that can be used to solve complex problems such as diagnosing a medical condition. Knowledge representation incorporates findings from psychology about how humans solve problems and represent knowledge in order to design formalisms that will make complex systems easier to design and build. The justification for knowledge representation is that conventional procedural code is not the best formalism to use to solve complex problems. Knowledge representation makes complex software easier to define and maintain than procedural code and can be used in expert systems.

For example, talking to experts in terms of business rules rather than code lessens the semantic gap between users and developers and makes development of complex systems more practical.

Knowledge representation goes hand in hand with automated reasoning because one of the main purposes of explicitly representing knowledge is to be able to reason about that knowledge, to make inferences, assert new knowledge, etc.

Randall Davis of MIT outlined five distinct roles to analyze a knowledge representation framework:

- A knowledge representation (KR) is most fundamentally a surrogate, a substitute for the thing itself, used to enable an entity to determine consequences by thinking rather than acting, i.e., by reasoning about the world rather than taking action in it.
- It is a set of ontological commitments, i.e., an answer to the question: In what terms should I think about the world?
- It is a fragmentary theory of intelligent reasoning, expressed in terms of three components: (i) the representation's fundamental conception of intelligent reasoning; (ii) the set of inferences the representation sanctions; and (iii) the set of inferences it recommends.
- It is a medium for pragmatically efficient computation, i.e., the computational environment in which thinking is accomplished. One contribution to this pragmatic efficiency is supplied by the guidance a representation provides for organizing information so as to facilitate making the recommended inferences.
- It is a medium of human expression, i.e., a language in which we say things about the world."

Characteristics

- **Primitives.** What is the underlying framework used to represent knowledge? Semantic networks were one of the first knowledge representation primitives.
- **Meta-Representation**. This is also known as the issue of reflection in computer science. It refers to the capability of a formalism to have access to information about its own state.
- **Incompleteness.** Traditional logic requires additional axioms and constraints to deal with the real world as opposed to the world of mathematics.
- **Definitions and Universals vs. facts and defaults**. Universals are general statements about the world such as "All humans are mortal". Facts are specific examples of universals such as "Socrates is a human and therefore mortal". In logical terms definitions and universals are about universal quantification while facts and defaults are about existential quantifications.
- **Non-Monotonic reasoning.** Non-monotonic reasoning allows various kinds of hypothetical reasoning. The system associates facts asserted with the rules and facts used to justify them and as those facts change updates the dependent knowledge as well.
- **Expressive Adequacy**. Researchers should be clear about how expressive (how much of full FOL expressive power) they intend their representation to be.
- **Reasoning Efficiency.** This refers to the run time efficiency of the system. The ability of the knowledge base to be updated and the reasoner to develop new inferences in a reasonable period of time.

3.2 Propositional logic

Propositional logic is the branch of mathematical logic concerned with the study of propositions and formed by other propositions with the use of logical connectives, and how their value depends on the truth value of their components.

Propositional logic, also known as *sentential logic* and *statement logic*, is the branch of logic that studies ways of joining and/or modifying entire propositions, statements or sentences to form more complicated propositions, statements or sentences, as well as the logical relationships and properties that are derived from these methods of combining or altering statements.

A *statement* can be defined as a declarative sentence, or part of a sentence, that is capable of having a truth-value, such as being true or false. So, for example, the following are statements:

- *George W. Bush is the 43rd President of the United States.*
- *Paris is the capital of France.*
- *Everyone born on Monday has purple hair.*

Sometimes, a statement can contain one or more other statements as parts. Consider for example, the following statement:

- *Either Ganymede is a moon of Jupiter or Ganymede is a moon of Saturn.*

While the above compound sentence is itself a statement, because it is true, the two parts, "Ganymede is a moon of Jupiter" and "Ganymede is a moon of Saturn", are themselves statements, because the first is true and the second is false.

Propositional logic, also known as *sentential logic*, is that branch of logic that studies ways of combining or altering statements or propositions to form more complicated statements or propositions. Joining two simpler propositions with the word "and" is one common way of combining statements. When two statements are joined together with "and", the complex statement formed by them is true if and only if *both* the component statements are true. Because of this, an argument of the following form is logically valid:

Paris is the capital of France and Paris has a population of over two million. Therefore, Paris has a population of over two million.

Syntax

A proposition or statement is a sentence which is either true or false. The **syntax** of propositional logic defines the allowable sentences. The **atomic sentences** consist of a single **proposition symbol.** Each such symbol stands for a proposition that can be true or false. We use symbols that start with an uppercase letter and may contain other letters or subscripts; for example: *P*, *Q*, *R* and *North.* There are two proposition symbols with fixed meanings: *True* is the always-true proposition and *False* is the always-false proposition. **Complex sentences** are constructed from simpler sentences, using parentheses and **logical connectives. There are five connectives in common use:**

1. ¬ **Negation (Not):-** A sentence such as ¬P is called the negation of P.
2. ∧ **Conjunction (AND):-** P ∧ Q is the conjunction of P and Q
3. ∨ **Disjunction (OR):-** P ∨ Q is the conjunction of P and Q.
4. → **Implication (Conditional):-** For propositions P and Q, the implication or conditional statement P→Q is false when P is true and Q is false, and is true otherwise. P is called the premise or hypothesis, and Q is called the conclusion.
5. ↔ **Biconditional (if and only if):** For propositions P and Q, the Biconditional statement P↔Q is true whenever both parts have the same truth value. P is called the premise or hypothesis, and Q is called the conclusion.

<u>Truth table of five logical connectives</u>

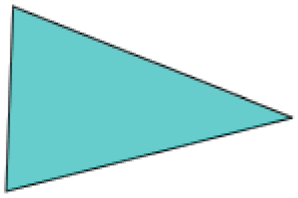
Let:
T=True
F=False

P	**Q**	**¬P**	**P ∧ Q**	**P ∨ Q**	**P → Q**	**P ↔ Q**
F	F	T	F	F	T	T
F	T	T	F	T	T	F
T	F	F	F	T	F	F
T	T	F	T	T	T	T

Example Conditional Statements
If you earn 90% of the possible points in cs 313k, then you will get an A.
Rewritten in logical notation:
$P \rightarrow Q$
Where
P is: You earn 90% of the possible points in cs 313k.
Q is: You will get an A in cs 313k.

Example Biconditional Statements
Example 1: Examine the sentences below.

Given:	p: A polygon is a triangle.
	q: A polygon has exactly 3 sides.
Problem:	Determine the truth values of this statement: $(p \rightarrow q) \wedge (q \rightarrow p)$



The compound statement $(p \rightarrow q) \wedge (q \rightarrow p)$ is a conjunction of two conditional statements. In the first conditional, p is the hypothesis and q is the conclusion; in the second conditional, q is the hypothesis and p is the conclusion. Let's look at a truth table for this compound statement.

p	q	$p \rightarrow q$	$q \rightarrow p$	$(p \rightarrow q) \wedge (q \rightarrow p)$
T	T	T	T	T
T	F	F	T	F
F	T	T	F	F
F	F	T	T	T

In the truth table above, when p and q have the same truth values, the compound statement $(p \rightarrow q) \wedge (q \rightarrow p)$ is true. When we combine two conditional statements this way, we have a **biconditional**.

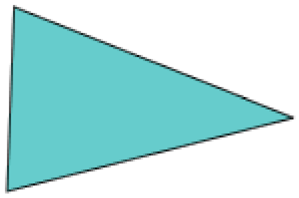
Definition: A biconditional statement is defined to be true whenever both parts have the same truth value. The biconditional operator is denoted by a double-headed arrow $\leftrightarrow$. The biconditional $p \leftrightarrow q$ represents "p if and only if q," where p is a hypothesis and q is a conclusion. The following is a truth table for biconditional $p \leftrightarrow q$.

p	q	$p \leftrightarrow q$
T	T	T
T	F	F
F	T	F
F	F	T

In the truth table above, $p \leftrightarrow q$ is true when p and q have the same truth values, (i.e., when either both are true or both are false.) Now that the biconditional has been defined, we can look at a modified version of Example 1.

Example 1:

Given:	p: A polygon is a triangle.
	q: A polygon has exactly 3 sides.
Problem:	What does the statement $p \leftrightarrow q$ represent?
Solution:	The statement $p \leftrightarrow q$ represents the sentence, "A polygon is a triangle if and only if it has exactly 3



	sides."

Note that in the biconditional above, the hypothesis is: "A polygon is a triangle" and the conclusion is: "It has exactly 3 sides." It is helpful to think of the biconditional as a conditional statement that is true in both directions.

Logical equivalence:- Two statements X and Y are *logically equivalent* if $X \leftrightarrow Y$ is a tautology. Another way to say this is: For each assignment of truth values to the *simple statements* which make up X and Y, the statements X and Y have identical truth values.

$$a \equiv \beta \quad \text{if and only if} \quad a \models \beta \text{ and } \beta \models a \,.$$

Example. Show that $P \rightarrow Q$ and $\sim P \vee Q$ are logically equivalent.

P	Q	$P \rightarrow Q$	$\sim P$	$\sim P \vee Q$
T	T	T	F	T
T	F	F	F	F
F	T	T	T	T
F	F	T	T	T

Since the columns for $P \rightarrow Q$ and $\sim P \vee Q$ are identical, the two statements are logically equivalent.

Validity: - A sentence is valid if it is true in *all* models. For example, the sentence **P ∨ ¬P** is valid. Valid sentences are also known as tautologies.

For any sentences a and β, a $\models \beta$ if and only if the sentence $(a \Rightarrow \beta)$ is valid.

P	**¬P**	**P ∨ ¬P**
F	T	T
T	**F**	**T**

Satisfiability:- A sentence is satisfiable if it is true in *some* model. If a sentence α is true in a model m, then we say that m **satisfies** α , or that m **is a model of** α. Determining the satisfiability of sentences in propositional logic was the first problem proved to be NP-complete. Many problems in computer science are really satisfiability problems. For example, all the constraint satisfaction problems are essentially asking whether the constraints are satisfiable by some assignment.

Unsatifiabality (Contradiction):- A sentence is unsatisfiable if it is false in *all* models. For example, the sentence **$P \land \neg P$** is unsatisfiable.

P	**$\neg P$**	**$P \land \neg P$**
F	T	F
T	**F**	**F**

LOGICAL IDENTITIES

$P \equiv P \land P$	idempotence of $\land$
$P \equiv P \lor P$	idempotence of $\lor$
$P \lor Q \equiv Q \lor P$	commutativity of $\lor$
$P \land Q \equiv Q \land P$	commutativity of $\land$
$(P \lor Q) \lor R \equiv P \lor (Q \lor R)$	associativity of $\lor$
$(P \land Q) \land R \equiv P \land (Q \land R)$	associativity of $\land$
$\neg(P \lor Q) \equiv \neg P \land \neg Q$	DeMorgan's Laws
$\neg(P \land Q) \equiv \neg P \lor \neg Q$	
$P \land (Q \lor R) \equiv (P \land Q) \lor (P \land R)$	distributivity of $\land$ over $\lor$
$P \lor (Q \land R) \equiv (P \lor Q) \land (P \lor R)$	distributivity of $\lor$ over $\land$
$P \lor T \equiv T$	domination laws
$P \land F \equiv F$	
$P \land T \equiv P$	identity laws
$P \lor F \equiv P$	
$P \lor \neg P \equiv T$	negation laws
$P \land \neg P \equiv F$	
$\neg(\neg P) \equiv P$	double negation law
$P \lor (P \land Q) \equiv P$	absorption laws
$P \land (P \lor Q) \equiv P$	
$P \rightarrow Q \equiv \neg P \lor Q$	implication
$P \rightarrow Q \equiv \neg Q \rightarrow \neg P$	contrapositive
$P \leftrightarrow Q \equiv [(P \rightarrow Q) \land (Q \rightarrow P)]$	equivalence
$[(P \land Q) \rightarrow R] \equiv [P \rightarrow (Q \rightarrow R)]$	exportation

Examples with Identities

1. $P \equiv P \wedge P$ - **idempotence of $\wedge$**
"Anna is wretched" is equivalent to "Anna is wretched and Anna is wretched".
2. $P \equiv P \vee P$ - **idempotence of $\vee$**
"Anna is wretched" is equivalent to "Anna is wretched or wretched".
3. $P \vee Q \equiv Q \vee P$ - **commutativity**
"Sam is rich or happy" is equivalent to "Sam is happy or rich".
$3'$. $P \wedge Q \equiv Q \wedge P$
"Sam is rich and Sam is happy" is equivalent to "Sam is happy and Sam is rich".
4. $\neg(P \vee Q) \equiv \neg P \wedge \neg Q$ - **DeMorgan's law**
"It is not the case that Sam is rich or happy" is equivalent to "Sam is not rich and he is not happy".
$4'$. $\neg(P \wedge Q) \equiv \neg P \vee \neg Q$
"It is not true that Abby is quick and strong" is equivalent to "Abby is not quick or Abby is not strong".
5. $P \wedge (Q \vee R) \equiv (P \wedge Q) \vee (P \wedge R)$ - **distributivity**
"Abby is strong, and Abby is happy or nervous" is equivalent to "Abby is strong and happy, or Abby is strong and nervous".
$5'$. $P \vee (Q \wedge R) \equiv (P \vee Q) \wedge (P \vee R)$
"Sam is tired, or Sam is happy and rested" is equivalent to "Sam is tired or happy, and Sam is tired or rested".
6. $P \vee \neg P \equiv T$ - **negation law**
"Ted is healthy or Ted is not healthy" is true.
$6'$. $P \wedge \neg P \equiv F$
"Kate won the lottery and Kate didn't win the lottery" is false.

7. $\neg(\neg P) \equiv P$ - **double negation**
"It is not the case that Tom is not rich" is equivalent to "Tom is rich".
8. $P \vee (P \wedge Q) \equiv P$ - **absorption**
"Kate is happy, or Kate is happy and healthy" is true if and only if "Kate is happy" is true.
8′. $P \wedge (P \vee Q) \equiv P$
"Kate is sick, and Kate is sick or angry" is true if and only if "Kate is sick" is true.
9. $P \rightarrow Q \equiv \neg P \vee Q$ - **implication**
"If I win tne lottery, then I will give you half the money" is true exactly when I either don't win the lottery, or I give you half the money.
10. $P \rightarrow Q \equiv \neg Q \rightarrow \neg P$ - **contrapositive**
"If Anna is healthy, then she is happy" is equivalent to "If Anna is not happy, then she is not healthy".
11. $P \leftrightarrow Q \equiv (P \rightarrow Q) \wedge (Q \rightarrow P)$ **equivalence**
"Anna is healthy if and only if she is happy" is equivalent to "If Anna is healthy, then she is happy, and if Anna is happy, then she is healthy".
12. $(P \wedge Q) \rightarrow R \equiv P \rightarrow (Q \rightarrow R)$ - **exportation**
"Anna is famous implies that if she is rich, then she is happy" is equivalent to "If Anna is famous and rich, then she is happy".

Exercise

1) Determine whether the following argument is valid.

   "If I work whole night on this problem, then I can solve it. If I solve the problem' then I will understand the topic. Therefore, I will work whole night on this problem, and then I will understand the topic"

   Let

   P= I work whole night on this problem

   Q= I can solve it

   R= I will understand the topic

$P \rightarrow Q$= If l work whole night on this problem, then l can solve it

$Q \rightarrow R$= If I solve the problem' then I will understand the topic

$P \rightarrow R$= I will work whole night on this problem, and then I will understand the topic

So result: $((P \rightarrow Q) \land (Q \rightarrow R)) \rightarrow (P \rightarrow R)$

P	Q	R	**$P \rightarrow Q$**	**$Q \rightarrow R$**	**$(P \rightarrow Q) \land (Q \rightarrow R)$**	**$P \rightarrow R$**	**$((P \rightarrow Q) \land (Q \rightarrow R)) \rightarrow (P \rightarrow R)$**
F	F	F	T	T	T	T	T
F	F	T	T	T	T	T	T
F	T	F	T	F	F	T	T
F	T	T	T	T	T	T	T
T	F	F	F	T	F	F	T
T	F	T	F	T	F	T	T
T	T	F	T	F	F	F	T
T	T	T	T	T	T	T	T

So given argument are valid.

2. Prove that the following sentence is valid :

"If prices fall then sell increases. If sell increases then John makes the whole money. But John doesn't make the whole money. Therefore, prices do not fall."

Let

P= prices fall

Q= sell increases

R= John makes the whole money

$P \rightarrow Q$= If prices fall then sell increases

$Q \rightarrow R$= If sell increases then John makes the whole money

$\neg P \rightarrow \neg R$= John doesn't make the whole money. Therefore, prices do not fall topic

So result: $((P \rightarrow Q) \land (Q \rightarrow R)) \rightarrow (\neg R \rightarrow \neg P)$

P	Q	R	$P \rightarrow Q$	$Q \rightarrow R$	$(P \rightarrow Q) \land (Q \rightarrow R)$	$\neg R \rightarrow \neg P$	$((P \rightarrow Q) \land (Q \rightarrow R)) \rightarrow (\neg R \rightarrow \neg P)$
F	F	F	T	T	T	T	T
F	F	T	T	T	T	T	T
F	T	F	T	F	F	T	T
F	T	T	T	T	T	T	T
T	F	F	F	T	F	F	T
T	F	T	F	T	F	T	T
T	T	F	T	F	F	F	T
T	T	T	T	T	T	T	T

So given argument are valid.

3. Prove that the following sentence is valid :

   "If a baby is hungry, then the baby cries. If the baby is not mad, then he does not cry. If a baby is mad, then his face looks abnormal. Therefore, if a baby is hungry, then his face looks abnormal."

   P= baby is hungry

   Q= the baby cries

   R= the baby is mad

   S= his face looks abnormal

   $P \rightarrow Q$= If a baby is hungry, then the baby cries

   $\neg R \rightarrow \neg Q$= If sell increases then John makes the whole money

   $R\rightarrow S$= If a baby is mad, then his face looks abnormal

   $P\rightarrow S$= if a baby is hungry, then his face looks abnormal.

So result: ((P → Q) ∧ (¬R → ¬Q) ∧ (R→S)) →(P→S)

P	Q	R	S	**P → Q**	**¬R**	**¬Q**	**¬R → ¬Q**	R→ **S**	**(P → Q) ∧ (¬R → ¬Q) ∧** (R→S)	P→ **S**	**((P → Q) ∧ (¬R → ¬Q) ∧** (R→S)) →(P→S)
F	F	F	F	**T**	T	T	**T**	**T**	T	T	T
F	F	F	T	**T**	T	T	**T**	**T**	T	T	T
F	F	T	F	**T**	F	T	**T**	**F**	F	T	T
F	F	T	T	**T**	F	T	**T**	**T**	T	T	T
F	T	F	F	**T**	T	F	**F**	**T**	F	T	T
F	T	F	T	**T**	T	F	**F**	**T**	F	T	T
F	T	T	F	**T**	F	F	**T**	**F**	F	T	T
F	T	T	T	**T**	F	F	**T**	**T**	T	T	T
T	F	F	F	**F**	T	T	**T**	**T**	F	F	T
T	F	F	T	**F**	T	T	**T**	**T**	F	T	T
T	F	T	F	**F**	F	T	**T**	**F**	F	F	T
T	F	T	T	**F**	F	T	**T**	**T**	F	T	T
T	T	F	F	**T**	T	F	**F**	**T**	F	F	T
T	T	F	T	**T**	T	F	**F**	**T**	F	T	T
T	T	T	F	**T**	F	F	**T**	**F**	F	F	T
T	T	T	T	**T**	F	F	**T**	**T**	T	T	T

So given argument are valid.

Inference in First order logic

Complex deductive arguments can be judged valid or invalid based on whether or not the steps in that argument follow the nine basic rules of inference. These rules of inference are all relatively simple, although when presented in formal terms they can look overly complex.

Conjunction:

1. P
2. Q
3. Therefore, P and Q.

1. It is raining in New York.
2. It is raining in Boston
3. Therefore, it is raining in both New York and Boston

Simplification

1. P and Q.
2. Therefore, P.

1. It is raining in both New York and Boston.
2. Therefore, it is raining in New York.

Addition

1. P
2. Therefore, P or Q.

1. It is raining
2. Therefore, either either it is raining or the sun is shining.

Absorption

1. If P, then Q.
2. Therfore, If P then P and Q.

1. If it is raining, then I will get wet.
2. Therefore, if it is raining, then it is raining and I will get wet.

Modus Ponens

1. If P then Q.
2. P.
3. Therefore, Q.

1. If it is raining, then I will get wet.
2. It is raining.
3. Therefore, I will get wet.

Modus Tollens

1. If P then Q.
2. Not Q. (~Q).
3. Therefore, not P (~P).

1. If it had rained this morning, I would have gotten wet.
2. I did not get wet.
3. Therefore, it did not rain this morning.

Hypothetical Syllogism

1. If P then Q.
2. If Q then R.
3. Therefore, if P then R.

1. If it rains, then I will get wet.
2. If I get wet, then my shirt will be ruined.
3. If it rains, then my shirt will be ruined.

Disjunctive Syllogism

1. Either P or Q.
2. Not P (~P).
3. Therefore, Q.

1. Either it rained or I took a cab to the movies.
2. It did not rain.
3. Therefore, I took a cab to the movies.

Constructive Dilemma

1. (If P then Q) and (If R then S).
2. P or R.
3. Therefore, Q or S.

1. If it rains, then I will get wet and if it is sunny, then I will be dry.
2. Either it will rain or it will be sunny.
3. Therefore, either I will get wet or I will be dry.

The above rules of inference, when combined with the rules of replacement, mean that propositional calculus is "complete." Propositional calculus is simply another name for formal logic.

Resolution

Resolution is a rule of inference leading to a refutation theorem-proving technique for sentences in propositional logic and first-order logic. In other words, iteratively applying the resolution rule in a suitable way allows for telling whether a propositional formula is satisfiable and for proving that a first-order formula is unsatisfiable; this method may prove the satisfiability of a first-order satisfiable formula, but not always, as it is the case for all methods for first-order logic. Resolution was introduced by John Alan Robinson in 1965.

Resolution in propositional logic

The resolution rule in propositional logic is a single valid inference rule that produces a new clause implied by two clauses containing complementary literals. A literal is a propositional variable or the negation of a propositional variable. Two literals are said to be complements if one is the negation of the other (in the following, a_i is taken to be the complement to b_j). The resulting clause contains all the literals that do not have complements. Formally:

$$\frac{a_1 \vee \ldots \vee a_i \vee \ldots \vee a_n, \quad b_1 \vee \ldots \vee b_j \vee \ldots \vee b_m}{a_1 \vee \ldots \vee a_{i-1} \vee a_{i+1} \vee \ldots \vee a_n \vee b_1 \vee \ldots \vee b_{j-1} \vee b_{j+1} \vee \ldots \vee b_m}$$

where

all as and bs are literals,
a_i is the complement to b_j, and
the dividing line stands for entails

The clause produced by the resolution rule is called the resolvent of the two input clauses.

When the two clauses contain more than one pair of complementary literals, the resolution rule can be applied (independently) for each such pair. However, only the pair of literals that are resolved upon can be removed: all other pair of literals remain in the resolvent clause.

A resolution technique

When coupled with a complete search algorithm, the resolution rule yields a sound and complete algorithm for deciding the satisfiability of a propositional formula, and, by extension, the validity of a sentence under a set of axioms.

This resolution technique uses proof by contradiction and is based on the fact that any sentence in propositional logic can be transformed into an equivalent sentence in conjunctive normal form. The steps are as follows:

1).All sentences in the knowledge base and the negation of the sentence to be proved (the conjecture) are conjunctively connected.

2).The resulting sentence is transformed into a conjunctive normal form with the conjuncts viewed as elements in a set, S, of clauses.

For example

$$(A_1 \lor A_2) \land (B_1 \lor B_2 \lor B_3) \land (C_1)$$

would give rise to a set

$$S = \{A_1 \lor A_2, B_1 \lor B_2 \lor B_3, C_1\}$$

3).The resolution rule is applied to all possible pairs of clauses that contain complementary literals. After each application of the resolution rule, the resulting sentence is simplified by removing repeated literals. If the sentence contains complementary literals, it is discarded (as a tautology). If not, and if it is not yet present in the clause set S, it is added to S, and is considered for further resolution inferences.

4).If after applying a resolution rule the empty clause is derived, the complete formula is unsatisfiable (or contradictory), and hence it can be concluded that the initial conjecture follows from the axioms.

5).If, on the other hand, the empty clause cannot be derived, and the resolution rule cannot be applied to derive any more new clauses, the conjecture is not a theorem of the original knowledge base.

One instance of this algorithm is the original Davis–Putnam algorithm that was later refined into the DPLL algorithm that removed the need for explicit representation of the resolvents.

This description of the resolution technique uses a set S as the underlying data-structure to represent resolution derivations. Lists, Trees and Directed Acyclic Graphs are other possible and common alternatives. Tree representations are more faithful to the fact that the resolution rule is binary. Together with a sequent notation for clauses, a tree representation also makes it clear to see how the resolution rule is related to a special case of the cut-rule, restricted to atomic cut-formulas. However, tree representations are not as compact as set or list representations, because they explicitly show redundant subderivations of clauses that are used more than once in the derivation of the empty clause. Graph representations can be as compact in the number of clauses as list representations and they also store structural information regarding which clauses were resolved to derive each resolvent.

Example

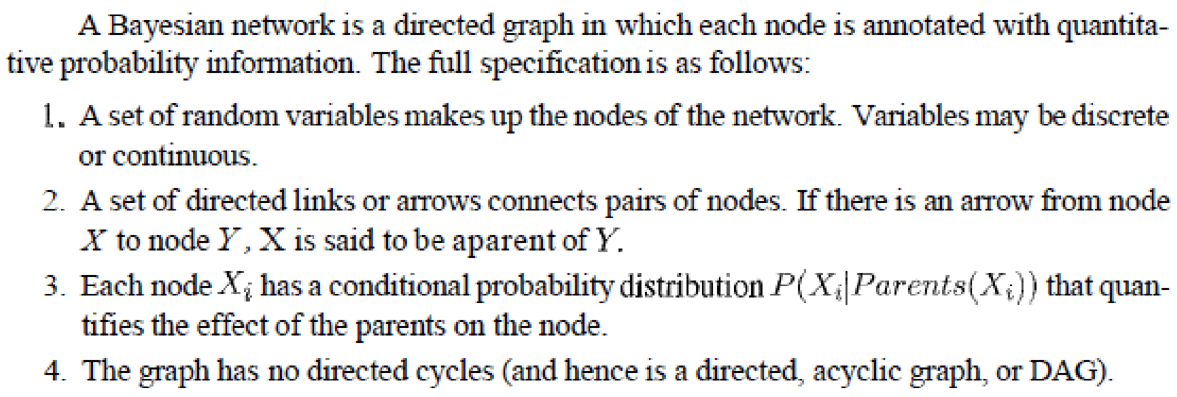
$$\frac{a \lor b, \quad \neg a \lor c}{b \lor c}$$

In English: if a or b is true, and a is false or c is true, then either b or c is true.

If a is true, then for the second premise to hold, c must be true. If a is false, then for the first premise to hold, b must be true.

So regardless of a, if both premises hold, then b or c is true.

Bayesian network



A Bayesian network is a directed graph in which each node is annotated with quantitative probability information. The full specification is as follows:

1. A set of random variables makes up the nodes of the network. Variables may be discrete or continuous.
2. A set of directed links or arrows connects pairs of nodes. If there is an arrow from node X to node Y, X is said to be aparent of Y.
3. Each node X_i has a conditional probability distribution $P(X_i|Parents(X_i))$ that quantifies the effect of the parents on the node.
4. The graph has no directed cycles (and hence is a directed, acyclic graph, or DAG).

Representing Knowledge in uncertain domain

Example

We argued that *Weather* is independent of the other variables; furthermore, we argued that *Toothache* and *Catch* are conditionally independent, given *Cavity.* These relationships are represented by the Bayesian network structure shown in Figure 14.1. Formally, the conditional independence of *Toothache* and *Catch* given *Cavity* is indicated by the absence of a link between *Toothache* and *Catch.* Intuitively, the network represents the fact that *Cavity* is a direct cause of *Toothache* and *Catch,* whereas no direct causal relationship exists between *Toothache* and *Catch.*

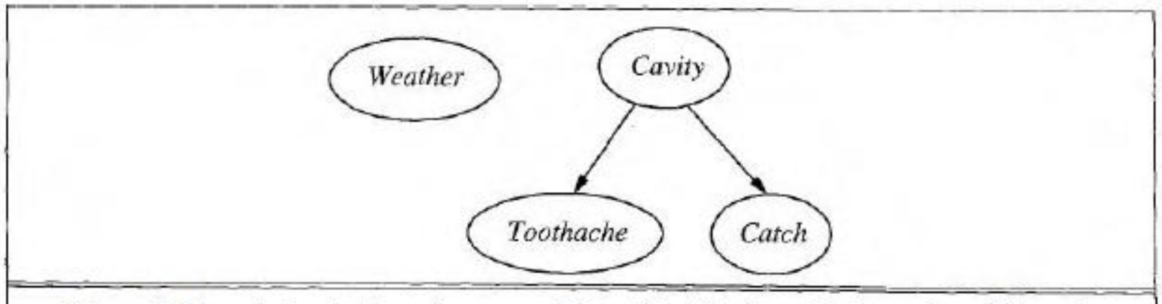
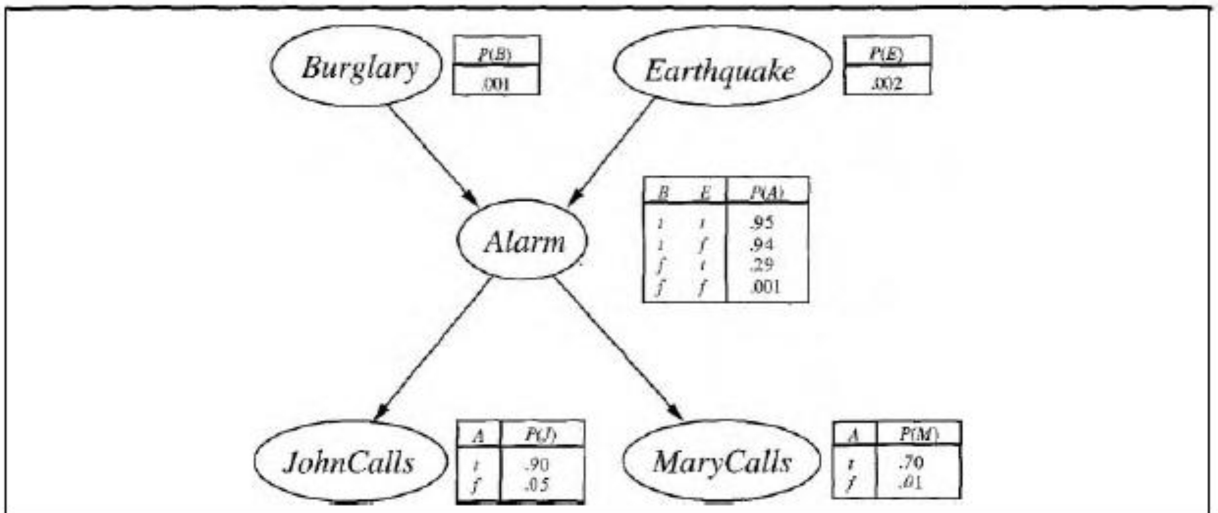


Figure 14.1 A simple Bayesian network in which *Weather* is independent of the other three variables and *Toothache* and *Catch* are conditionally independent, given **Cavity.**

<u>Example</u>

You have a new burglar alarm installed at home. It is fairly reliable at detecting a burglary, but also responds on occasion to minor earthquakes. (This example is due to Judea Pearl, a resident of Los Angeles-hence the acute interest in earthquakes.) You also have two neighbors, John and Mary, who have promised to call you at work when they hear the alarm. John always calls when he hears the alarm, but sometimes confuses the telephone ringing with the alarm and calls then, too. Mary, on the other hand, likes rather loud music and sometimes misses the alarm altogether. Given the evidence of who has or has not called, we would like to estimate the probability of a burglary.

In the figure, each distribution is shown as a **conditional probability table,** or CPT. Each row in a CPT contains the conditional probability of each node value for a **conditioning case.** A conditioning case is just a possible combination of values for the parent nodes-a miniature atomic event, if you like. Each row must sum to 1, because the entries represent an exhaustive set of cases for the variable.



P(B)
.001

P(E)
.002

B	E	P(A)
t	t	.95
t	f	.94
f	t	.29
f	f	.001

A	P(J)
t	.90
f	.05

A	P(M)
t	.70
f	.01

Figure 14.2 A typical Bayesian network, showing both the topology and the conditional probability tables (CPTs). In the CPTs, the letters B, E, A, J, and M stand for *Burglary, Earthquake, Alarm, JohnCalls*, and *MaryCalls*, respectively.

<u>The semantic of the Bayesian network</u>

There are two
ways in which one can understand the semantics of Elayesian networks. The first is to see the network as a representation of the joint probability distribution. The second is to view it as an encoding of a collection of conditional independence statements.

Representing the full joint distribution

A Bayesian network provides a complete description of the domain. Every entry in the full joint probability distribution (hereafter abbreviated as "joint") can be calculated from the information in the network. A generic entry in the joint distribution is the probability of a conjunction of particular assignments to each variable, such as $P(X_1 = x_1 A \ldots \wedge X, = x,)$. We use the notation $P(x_1, \ldots, x,)$ as an abbreviation for this. The value of this entry is given by the formula

$$P(x_1, \ldots, x_n) = \prod_{i=1}^{n} P(x_i|parents(X_i)) \, , \tag{14.1}$$

where *parents(X$_i$)* denotes the specific values of the variables in *Parents(X$_i$)*.

To illustrate this, we can calculate the probability that the alarm has sounded, but neither a burglary nor an earthquake has occurred, and both John and Mary call. We use single-letter names for the variables:

$$\begin{aligned} &P(j \wedge m \wedge a \wedge \neg b \wedge \neg e) \\ &\quad = P(j|a)P(m|a)P(a|\neg b A \neg e)P(\neg b)P(\neg e) \\ &\quad = 0.90 \text{ x } 0.70 \text{ x } 0.001 \text{ x } 0.999 \text{ x } 0.998 = 0.0006'2. \end{aligned}$$

A method for constructing Bayesian networks

First, we rewrite the joint distribution in terms of a conditional probability, using the product rule

$$P(x_1, \ldots, x_n) = P(x_n|x_{n-1}, \ldots, x_1)P(x_{n-1}, \ldots, x_1)$$

Then we repeat the process, reducing each conjunctive probability to a conditional probability and a smaller conjunction. We end up with one big product:

$$\begin{aligned} P(x_1, \ldots, x_n) &= P(x_n|x_{n-1}, \ldots, x_1)P(x_{n-1}|x_{n-2}, \ldots, x_1) \cdots P(x_2|x_1)P(x_1) \\ &= \prod_{i=1}^{n} P(x_i|x_{i-1}, \ldots, x_1) \, . \end{aligned}$$

This identity holds true for any set of random variables and is called the **chain rule.** Comparing it with Equation (14.1), we see that the specification of the joint distribution is equivalent to the general assertion that, for every variable Xi in the network,

$$\mathbf{P}(X_i|X_{i-1}, \ldots, X_1) = \mathbf{P}(X_i|Parents(X_i)) \, , \tag{14.2}$$

provided that $Parents(X_i) \subseteq \{X_{i-1}, \ldots, X_1\}$. This last condition is satisfied by labeling the nodes in any order that is consistent with the partial order implicit in the graph structure.

Hidden Markov model (HMM)

A **hidden Markov model** (**HMM**) is a statistical Markov model in which the system being modeled is assumed to be a Markov process with unobserved (*hidden*) states. An HMM can be presented as the simplest dynamic Bayesian network.

In simpler Markov models (like a Markov chain), the state is directly visible to the observer, and therefore the state transition probabilities are the only parameters. In a *hidden* Markov model, the state is not directly visible, but output, dependent on the state, is visible. Each state has a probability distribution over the possible output tokens. Therefore the sequence of tokens generated by an HMM gives some information about the sequence of states. Note that the adjective 'hidden' refers to the state sequence through which the model passes, not to the parameters of the model; the model is still referred to as a 'hidden' Markov model even if these parameters are known exactly.

Hidden Markov models are especially known for their application in temporal pattern recognition such as speech, handwriting, gesture recognition, part-of-speech tagging, musical score following, partial discharges and bioinformatics.

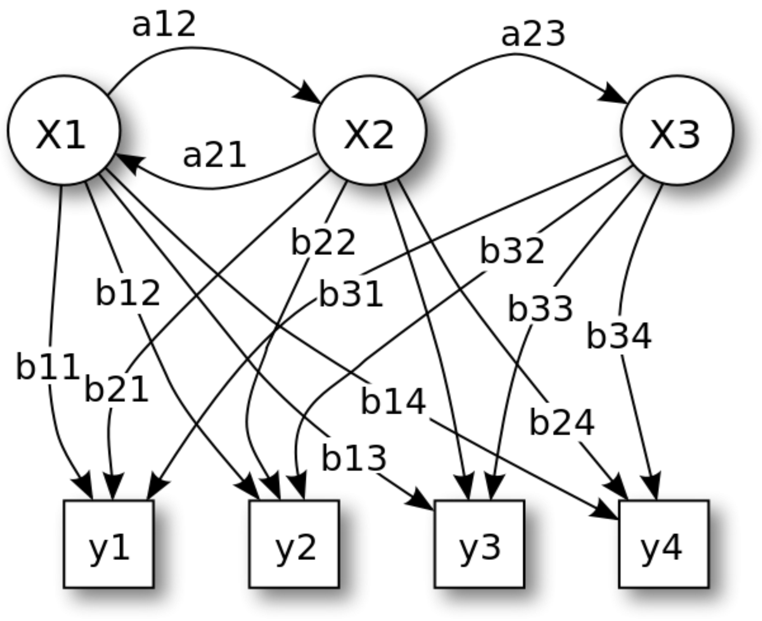


Figure 1. Probabilistic parameters of a hidden Markov model (example)
x — states
y — possible observations
a — state transition probabilities
b — output probabilities
Each oval shape represents a random variable that can adopt any of a number of values. The random variable $x(t)$ is the hidden state at time t (with the model from the above diagram, $x(t) \in \{ x_1, x_2, x_3 \}$). The random variable $y(t)$ is the observation at time t (with $y(t) \in \{ y_1, y_2, y_3, y_4 \}$). The arrows in the diagram (often called a trellis diagram) denote conditional dependencies.
It is clear that the conditional probability distribution of the hidden variable $x(t)$ at time t, given the values of the hidden variable x at all times, depends *only* on the value of the hidden variable $x(t-1)$: the values at time $t-2$ and before have no influence. This is called the Markov property. Similarly, the value of the observed variable $y(t)$ only depends on the value of the hidden variable $x(t)$ (both at time t).

2.3 Elements Of Hidden Markov Model

We now define elements Of HMM, HMM is characterized by following

1. Number of state N

2. Number of distinct observation symbol per state M, $V = V_1, V_2, \cdots, V_M$

3. State transition probability, $a_{ij} = P[q_{t+1} = S_i | q_t = S_j]$, $1 \leq i, j \leq N$

4. Observation symbol probability distribution in state j,$B_j(K) = P[V_k \; at \; t | q_t = S_j]$

5. The initial state distribution $\pi = \pi_i$ where $\pi_i = P[q_1 = S_i] \qquad 1 \leq i \leq N$[2]

Given appropriate value of N,M,A,B and π, HMM can be used as **generator** to give an observation sequence

$$O = O_1 \; O_2 \; O_3 \cdots O_T$$

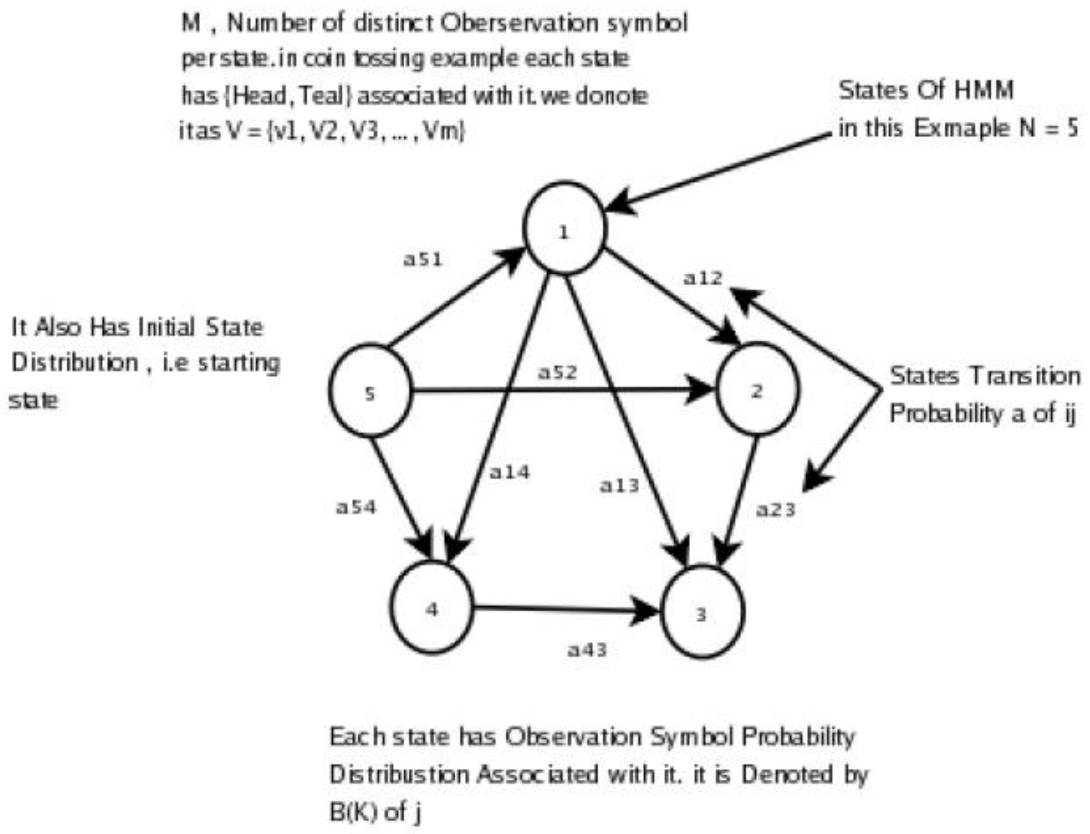

M , Number of distinct Oberservation symbol per state.in coin tossing example each state has {Head, Teal} associated with it we donote it as V = {v1, V2, V3, ... , Vm}
States Of HMM in this Exmaple N = 5
It Also Has Initial State Distribution , i.e starting state
States Transition Probability a of ij
1
2
3
4
5
a51
a12
a52
a14
a13
a54
a23
a43
Each state has Observation Symbol Probability Distribustion Associated with it. it is Denoted by B(K) of j

Figure 3: Elements of HMM

Example

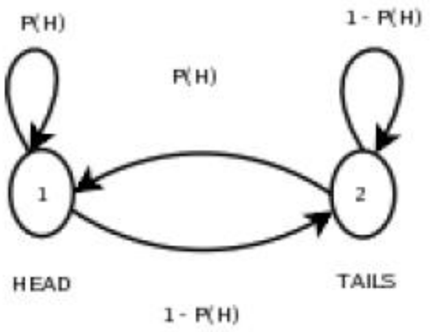
Coin Toss Models: Assume the following scenario. You are in a room with a barrier through which you cannot see what is happening. On the other side of the barrier is another person who is performing a coin (or multiple coin) tossing experiment. The other person will not tell you anything about what he is doing exactly, he will only tell you the result of each coin flip. Thus a sequence of hidden coin tossing experiments is performed, with the observation sequence consisting of a series of heads and tails, for e.g., a typical observation sequence would be

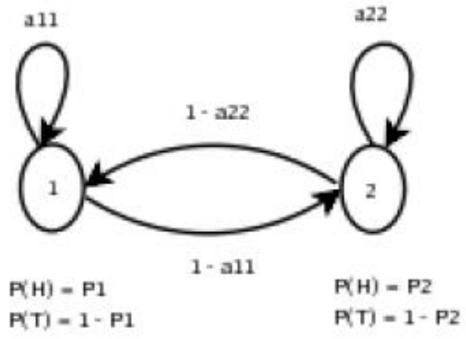
O = O1 O2 O3 · · ·OT (Head Tail Head · · · Tail)

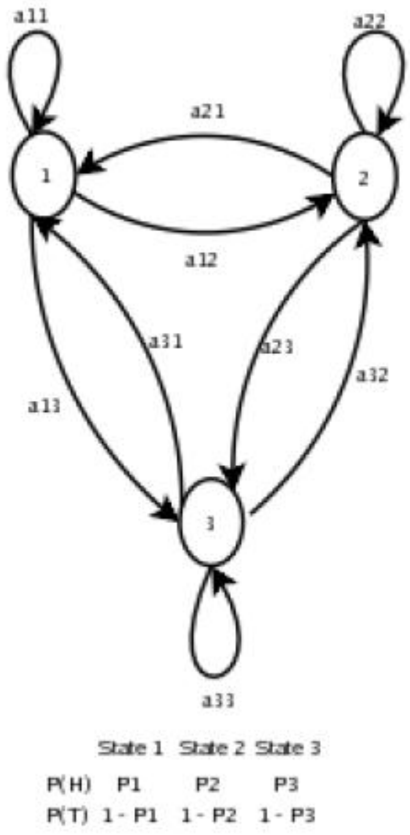
to model this coin tossing event with HMM, the fundamental question is

• What the state in model correspond to

• How many state in the model should there be







Speech Recognition using HMM

3.1 Block Diagram Of Speech Recognition

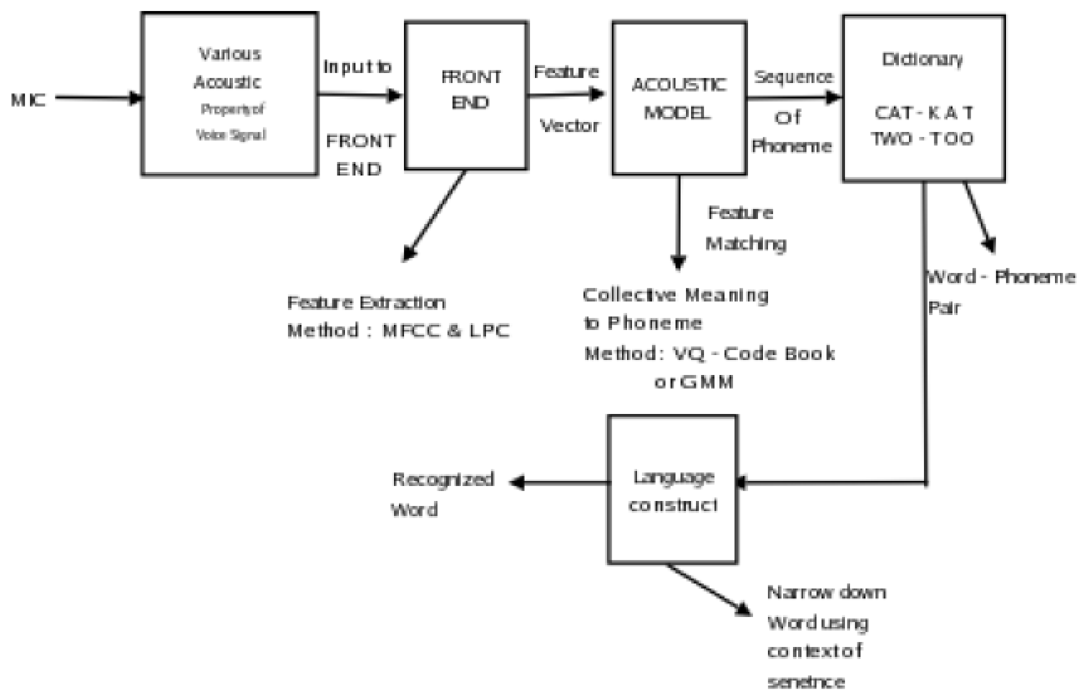

MIC
Various Acoustic Property of Voice Signal
Input to FRONT END
FRONT END
Feature Vector
ACOUSTIC MODEL
Sequence Of Phoneme
Dictionary
CAT - K A T
TWO - TOO
Feature Extraction Method : MFCC & LPC
Feature Matching
Collective Meaning to Phoneme
Method: VQ - Code Book or GMM
Word - Phoneme Pair
Recognized Word
Language construct
Narrow down Word using context of senetnce

Figure 6: Block Diagram of Speech Recognition

Speech recognition consists of two main modules, feature extraction and feature matching.

1. The purpose of feature extraction module is to convert speech waveform to some type of representation for further analysis and processing, this extracted information is known as feature vector. The process of converting voice signal to feature vector is done by signal-processing front end module. As shown in above block diagram input to front-end is noise free voice sample and output of it is feature vector.
2. In feature matching, the extracted feature vector from unknown voice sample is scored against acoustic model, the model with max score wins ,and it's output is considered as recognized word. Following are the few method for implementing front-end(for extracting feature factor)
   - MFCC (Mel-Frequency Cepstrum Coefficient)
   - LPC (Linear Predictive Coding)

Once the feature vector are obtained we build the acoustic model. The acoustic model is used to score the unknown voice sample. As shown in block diagram, Output of front-end is given as input to acoustic model. Different types of acoustic model are

- VQ-Code Book [1]
- GMM-Gaussian Mixture Model