R22 B.Tech. CSE (AI and ML) Syllabus

JNTU Hyderabad

SUBJECTE CODE:AM502PC MACHINE LEARNING

B.Tech. III Year I Sem.

Unit – 5

Reinforcement Learning – Overview – Getting Lost Example

Markov Chain Monte Carlo Methods – Sampling – Proposal Distribution – Markov Chain Monte Carlo – Graphical Models – Bayesian Networks – Markov Random Fields – Hidden Markov Models – Tracking Methods

Reinforcement Learning – Overview

A classic example of reinforcement learning using the concept of "getting lost" would be a scenario where an autonomous robot is navigating an unfamiliar environment, like a maze, and must learn to reach a specific destination by trial and error, receiving positive rewards for moving closer to the goal and negative rewards for going further away or hitting obstacles, essentially "learning" how to navigate without getting lost through repeated attempts and feedback from the environment

Reinforcement Learning – Overview

Types of Reinforcement learning

There are mainly two types of reinforcement learning, which are:

Positive Reinforcement:

The positive reinforcement learning means adding something to increase the tendency that expected behavior would occur again. It impacts positively on the behavior of the agent and increases the strength of the behavior.

This type of reinforcement can sustain the changes for a long time, but too much positive reinforcement may lead to an overload of states that can reduce the consequences.

Negative Reinforcement:

The negative reinforcement learning is opposite to the positive reinforcement as it increases the tendency that the specific behavior will occur again by avoiding the negative condition.

It can be more effective than the positive reinforcement depending on situation and behavior, but it provides reinforcement only to meet minimum behavior.

Elements of Reinforcement Learning

There are four main elements of Reinforcement Learning, which are given below:

1. Policy
2. Reward Signal
3. Value Function
4. Model of the environment

Elements of Reinforcement Learning

There are four main elements of Reinforcement Learning, which are given below:

1. Policy

A policy can be defined as a way how an agent behaves at a given time. It maps the perceived states of the environment to the actions taken on those states. A policy is the core element of the RL as it alone can define the behavior of the agent. In some cases, it may be a simple function or a lookup table, whereas, for other cases, it may involve general computation as a search process. It could be deterministic or a stochastic policy:

For deterministic policy: a = π(s)

For stochastic policy: π(a | s) = P[At =a | St = s]

Elements of Reinforcement Learning

There are four main elements of Reinforcement Learning, which are given below:

2) Reward Signal: The goal of reinforcement learning is defined by the reward signal. At each state, the environment sends an immediate signal to the learning agent, and this signal is known as a **reward signal**. These rewards are given according to the good and bad actions taken by the agent. The agent's main objective is to maximize the total number of rewards for good actions. The reward signal can change the policy, such as if an action selected by the agent leads to low reward, then the policy may change to select other actions in the future.

3) Value Function: The value function gives information about how good the situation and action are and how much reward an agent can expect. A reward indicates the **immediate signal for each good and bad action**, whereas a value function specifies **the good state and action for the future**. The value function depends on the reward as, without reward, there could be no value. The goal of estimating values is to achieve more rewards.

Elements of Reinforcement Learning

4) Model: The last element of reinforcement learning is the model, which mimics the behavior of the environment. With the help of the model, one can make inferences about how the environment will behave. Such as, if a state and an action are given, then a model can predict the next state and reward.

The model is used for planning, which means it provides a way to take a course of action by considering all future situations before actually experiencing those situations. The approaches for solving the RL problems **with the help of the model** are termed as the **model-based approach**. Comparatively, an approach **without using a model** is called a **model-free approach**.

Key points of this example:

- **State:**
- At any point in time, the robot's "state" is its current location within the maze, including information like proximity to walls and potential pathways.
- **Action:**

- The robot can choose actions like "move forward", "turn left", "turn right" based on its current state.
- **Reward:**
- The robot receives a positive reward when it moves closer to the goal and a negative reward when it moves further away or hits a wall.
- **Learning process:**
- Through repeated trials, the robot learns which actions in each state lead to the highest cumulative reward, effectively building a strategy to navigate the maze without getting lost

How it works:

- **Exploration vs. Exploitation:**

  Initially, the robot might explore different paths randomly to understand the maze layout.

- **Policy Improvement:**

Over time, the robot gradually starts to prioritize actions that have led to positive rewards in the past, leading to a more efficient navigation strategy.

- **Q-Learning:**

A common reinforcement learning algorithm that can be used in this scenario involves creating a "Q-table" where each cell represents a state-action pair, storing the expected future reward for taking a particular action in a given state.

Real-world applications:

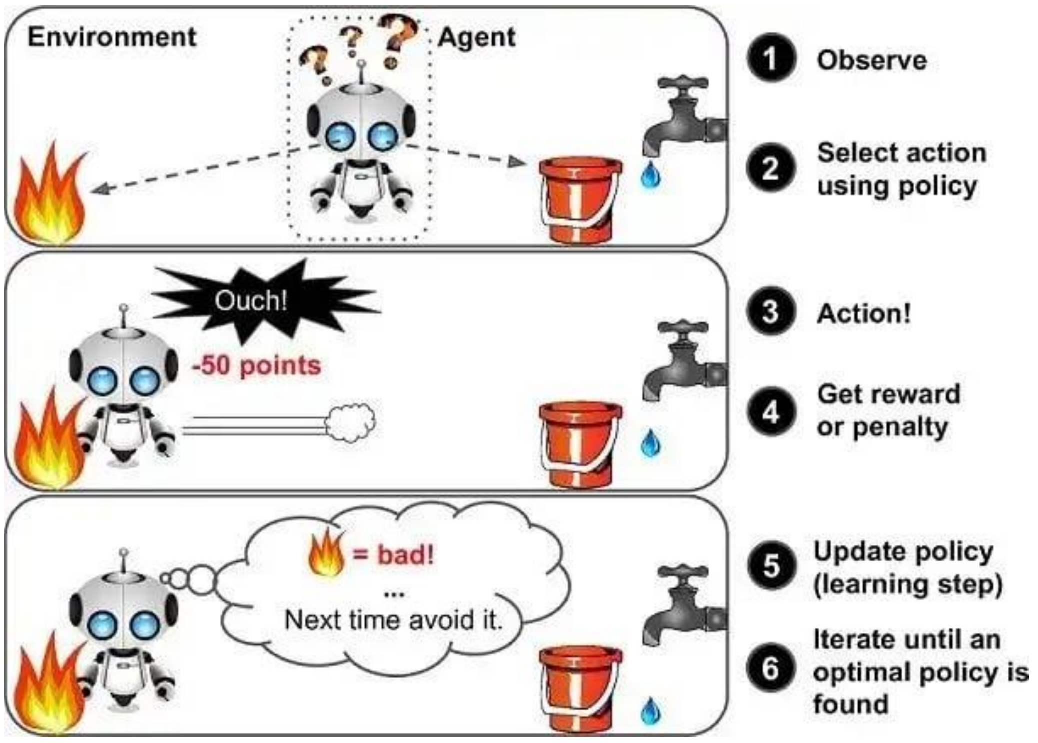
- **Robot navigation:**
- This concept is directly applicable to robots navigating complex environments like warehouses or disaster zones.
- **Autonomous vehicles:**
- Self-driving cars can use reinforcement learning to learn how to navigate roads, avoiding obstacles and making optimal route decisions.
- **Game playing:**
- AI agents in video games can learn to play effectively by receiving rewards for achieving goals within the game

Real-world applications:

- **Robot navigation:**

This concept is directly applicable to robots navigating complex environments like warehouses

or disaster zones.


Environment
Agent
1 Observe
2 Select action using policy
Ouch!
-50 points
3 Action!
4 Get reward or penalty
= bad!
...
Next time avoid it.
5 Update policy (learning step)
6 Iterate until an optimal policy is found

Markov Chain Monte Carlo Methods – Sampling – Proposal Distribution – Markov Chain Monte Carlo – Graphical Models – Bayesian Networks – Markov Random Fields – Hidden Markov Models – Tracking Methods

Monte Carlo methods, or **Monte Carlo experiments**, are a broad class of computational algorithms that rely on repeated random sampling to obtain numerical results. The underlying concept is to use randomness to solve problems that might be deterministic in principle. The name comes from the Monte Carlo Casino in Monaco, where the primary developer of the method.

Monte Carlo methods are mainly used in three distinct problem classes: **optimization, numerical integration,**

and generating draws from a probability distribution. They can also be used to model phenomena with significant uncertainty in inputs, such as calculating the risk of a nuclear power plant failure. Monte Carlo methods are often implemented using computer simulations, and they can provide approximate solutions to problems that are otherwise intractable or too complex to analyze mathematically.

Sampling distribution is essential in various aspects of real life. Sampling distributions are important for inferential statistics. A sampling distribution represents the distribution of a statistic, like the mean or standard deviation, which is calculated from multiple samples of a population. It shows how these statistics vary across different samples drawn from the same population.

In this article, we will discuss the Sampling Distribution in detail and its types along with examples and go through some practice questions too.

Sampling distribution is also known as a **finite-sample distribution**. Sampling distribution is the probability distribution of a statistic based on random samples of a given population. It represents the distribution of frequencies on how spread apart various outcomes will be for a specific population.

Some important terminologies related to sampling distribution are given below:

- **Statistic:** A numerical summary of a sample, such as mean, median, standard deviation, etc.
- **Parameter:** A numerical summary of a population is often estimated using sample statistics.
- **Sample**: A subset of individuals or observations selected from a population.
- **Population:** Entire group of individuals or observations that a study aims to describe or draw conclusions about.
- **Sampling Distribution:** Distribution of a statistic (e.g., mean, standard deviation) across multiple samples taken from the same population.

Central Limit Theorem(CLT): A fundamental theorem in statistics stating that the sampling distribution of the sample mean tends to be approximately normal as the sample size increases, regardless of the shape of the population distribution

Some important terminologies related to sampling distribution are given below:

- **Standard Error:** Standard deviation of a sampling distribution, representing the variability of sample statistics around the population parameter.
- **Bias**: Systematic error in estimation or inference, leading to a deviation of the estimated statistic from the true population parameter.
- **Confidence Interval:** A range of values calculated from sample data that is likely to contain the population parameter with a certain level of confidence.
- **Sampling Method:** Technique used to select a sample from a population, such as simple random sampling, stratified sampling, cluster sampling, etc.
- **Inferential Statistics:** Statistical methods and techniques used to draw conclusions or make inferences about a population based on sample data.
- **Hypothesis Testing:** A statistical method for making decisions or drawing conclusions about a population parameter based on sample data and assumptions about the population.

3 main factors influencing the variability of a sampling distribution are:

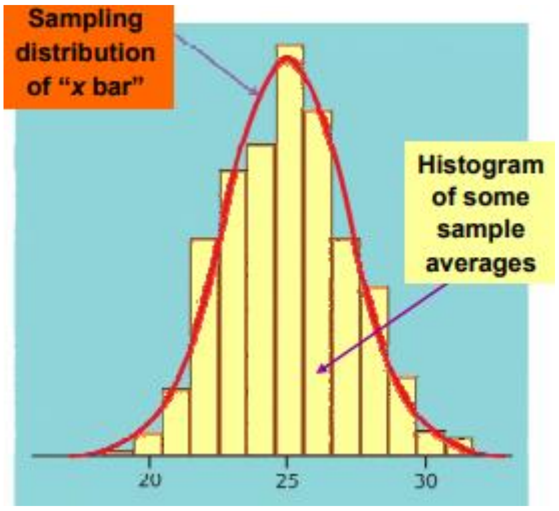
1. **Number Observed in a Population:** The symbol for this variable is "N." It is the measure of observed activity in a given group of data.
2. **Number Observed in Sample:** The symbol for this variable is "n." It is the measure of observed activity in a random sample of data that is part of the larger grouping.
3. **Method of Choosing Sample:** How you chose the samples can account for variability in some cases.

Sampling Distribution of Mean

it focuses on calculating the mean

or rather the average of every sample group

- chosen from the population and plotting the data points. The graph shows a normal distribution where the center is the mean of the sampling distribution, which represents the mean of the entire population.
- Mean, or center of the sampling distribution of x̄, is equal to the population mean, μ.
- μx− = μ



Standard deviation of the sampling distribution is σ/√n,

where n is the sample size.

σxσx = *σ/√n*

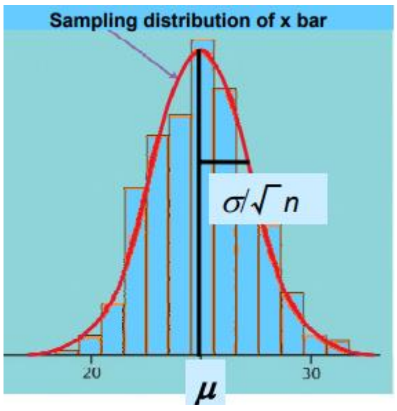
Standard deviation of the sampling distribution measures

how much the sample statistic varies from sample to

sample. It is smaller than the standard deviation

of the population by a factor of √n.

Averages are less variable than individual observations



T-Distribution

Sampling distribution involves a small population or a population about which you don't know much. It is used to estimate the mean of the population and other statistics such as confidence intervals, statistical differences and linear regression. T-distribution uses a t-score to evaluate data that wouldn't be appropriate for a normal distribution.

Formula for the t-score, denoted as t, is:

t = [x - μ] / [s /√(n)]

where:

- **x** is Sample Mean

- **μ** is Population Mean (or an estimate of it)
- **s** is Sample Standard Deviation
- **n** is Sample Size

This formula calculates the difference between the sample mean and the population mean, scaled by the standard error of the sample mean. The t-score helps to assess whether the observed difference between the sample and population means is statistically significant.

Markov chain Monte Carlo methods create samples from a continuous random variable, with probability density proportional to a known function. These samples can be used to evaluate an integral over that variable, as its expected value or variance.

Practically, an ensemble of chains is generally developed, starting from a set of points arbitrarily chosen and sufficiently distant from each other. These chains are stochastic processes of "walkers" which move around randomly according to an algorithm that looks for places with a reasonably high contribution to the integral to move into next, assigning them higher probabilities.

Random walk Monte Carlo methods are a kind of random simulation or Monte Carlo method. However, whereas the random samples of the integrand used in a conventional Monte Carlo integration are statistically independent, those used in MCMC are autocorrelated. Correlations of samples introduces the need to use the Markov chain central limit theorem when estimating the error of mean values.

These algorithms create Markov chains such that they have an equilibrium distribution which is proportional to the function given.

In statistics, **Markov chain Monte Carlo** (**MCMC**) is a class of algorithms used to draw samples from a probability distribution. Given a probability distribution, one can construct a Markov chain whose elements' distribution approximates it – that is, the Markov chain's equilibrium distribution matches the target distribution. The more steps that are included, the more closely the distribution of the sample matches the actual desired distribution.

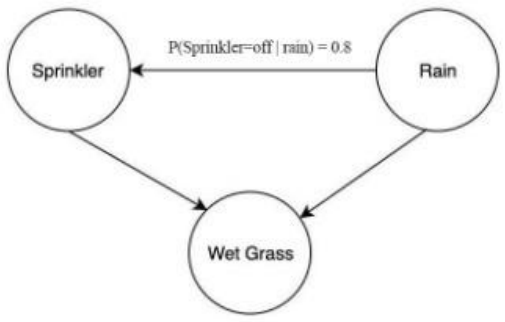
Markov chain Monte Carlo methods are used to study probability distributions that are too complex or too highly dimensional to study with analytic techniques alone. Various algorithms exist for constructing such Markov chains, including the Metropolis–Hastings algorithm.

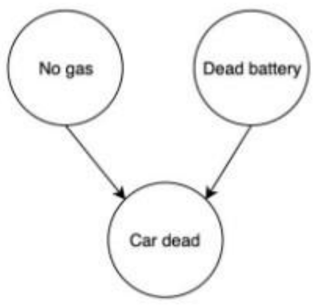
Graphical Model

One major difference between Machine Learning (ML) and Deep Learning (DL) is the
amount of domain knowledge sought to solve a problem. ML algorithms regularly exploit domain knowledge. However, the solution can be biased if the knowledge is incomplete. However, if it is done correctly, we can solve problems more efficiently. The Graphical model (GM) is a branch of ML which uses a graph to represent a domain problem. Many ML & DL algorithms, including Naive Bayes' algorithm, the Hidden Markov Model, Restricted Boltzmann machine and Neural Networks, belong to the GM. Studying it allows us a bird's eye view on many ML algorithms. In this article, we focus on the basic in representing a problem using a graphical model. Later
in this series, we will discuss how inference is made and how the model is trained.

Probabilistic graphical modeling combines both probability and graph theory. The probabilistic part reason under uncertainty. So we can use probability theory to model and argue the real-world problems better. The graph part models the dependency or correlation.

In GM, we model a domain problem with a collection of random variables ($X_1, \dots,$ Xn) as a joint distribution $p(X_1, \dots, Xn)$. The model is represented by a graph. Each node in the graph represents a variable with each edge representing the dependency or correlation between two variables.





For example, the sprinkler problem above is modeled by three variables:
1. whether the sprinkler is automatically on or off,
2. whether it is raining, and
3. whether the grass is wet.
And we model the problem with the joint probability $p(X_1, \dots, X_n)$

The likelihood of the observations $P(E)$ — the chance of the lab results (Evidence E).

The marginal probability $P(X_1)$, P(X_1, X_3), etc … — the chance of having Alzheimer when you are 70.
· Conditional probability (or a posterior belief based on evidence), P($Y|E$) — the chance of having Alzheimer given your parent has it.
· Maximum a Posterior (MAP), arg max $P(X, E)$ — the most likely disease given the lab results.
For example, the marginal probability of having the grass wet is computed by summing over other variables. For the conditional probability, we can first apply Bayes' Theorem followed by the corresponding marginal probabilities.

$$p(X_1) = \sum_{X_2} \sum_{X_3} \dots \sum_{X_n} p(X_1, \dots, X_n) \qquad p(X_1 \mid X_2) = \frac{\sum_{X_3} \dots \sum_{X_n} p(X_1, \dots, X_n)}{\sum_{X_1} \sum_{X_3} \dots \sum_{X_n} p(X_1, \dots, X_n)}$$

Joint probability

Which probability distributions below is more complex? Let's give some serious thoughts here because it helps us to understand machine learning (ML) better.

$$p(d, i, g, s, l)\,, \qquad p(d)\,p(i)\,p(g \mid d, i)\,p(s \mid i)\,p(l \mid g)$$

For the L.H.S. distribution, we can expand it using the chain rule.

$$p(d, i, g, s, l) = \underbrace{p(d)\,p(i \mid d)}_{p(d,\,i)}\,p(g \mid d, i)\,p(s \mid d, i, g)\,p(l \mid d, i, g, s)$$

Let's assume that after further analysis of the domain problem, we discover the independence claims below. For example, the first claim is variable *i* is independent of *d*.

$$i \perp d,\ s \perp \{d, g\},\ l \perp \{d, i, s\} \qquad (a \perp b \text{ means } a \text{ is independent of } b)$$

Therefore, we can simplify the R.H.S. expression as:

Simplified to

$$p(d, i, g, s, l) = p(d)\,p(i)\,p(g \mid d, i)\,p(s \mid i)\,p(l \mid g)$$

Original: $p(d, i, g, s, l) = p(d)\,p(i \mid \cancel{d})\,p(g \mid d, i)\,p(s \mid \cancel{d}, i, \cancel{g})\,p(l \mid \cancel{d}, \cancel{i}, g, \cancel{s})$

s is independent of *d* or *g*

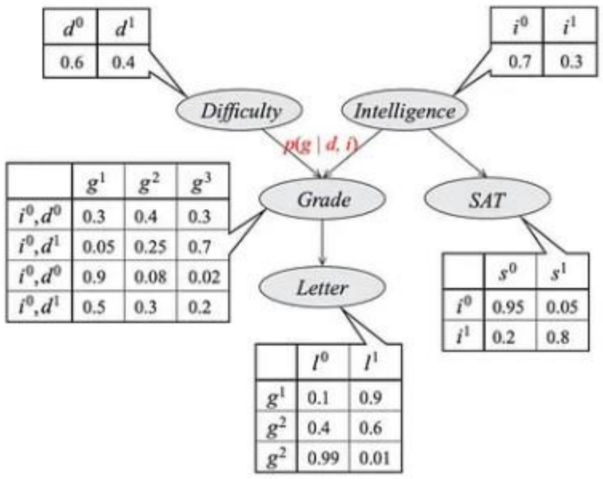
We can draw this dependency using a directed Graphical model. Each node represents a variable and to quantify the dependency, we associate a conditional probability for each node.

$$P(x_i \mid x_{A_i})$$

the parents of x_i

These model parameters are presented as the table entries below.

These model parameters are presented as the table entries below.



d^0	d^1
0.6	0.4

i^0	i^1
0.7	0.3

	g^1	g^2	g^3
i^0,d^0	0.3	0.4	0.3
i^0,d^1	0.05	0.25	0.7
i^0,d^0	0.9	0.08	0.02
i^0,d^1	0.5	0.3	0.2

	s^0	s^1
i^0	0.95	0.05
i^1	0.2	0.8

	l^0	l^1
g^1	0.1	0.9
g^2	0.4	0.6
g^2	0.99	0.01

So what is the complexity for these models? $p(d, i, g, s, l)$ have 5 variables with the possible combinations of 32 (2). To model ^{5} p, we sample data for 32 possibilities. On
the contrary, the graph above has 4 tables with only 26 parameters.
Don't underestimate the curse of exponential. 64 variables will have a combination of

18,446,744,073,709,551,616 (20 digits)

i.e. a simple 8 × 8 grayscale image will contain enough variables (pixels) that make an
ML problem too complex to solve. In ML, it is important to discover independence to break the curse of this exponential complexity, for example, Naive Bayes classifier, Hidden Markov Model, Maximum likelihood estimation with i.i.d., etc… The distribution of a variable can be calculated given all its neighbors are know. In ML, the original grand design is missing. We spot global insight through localized findings.

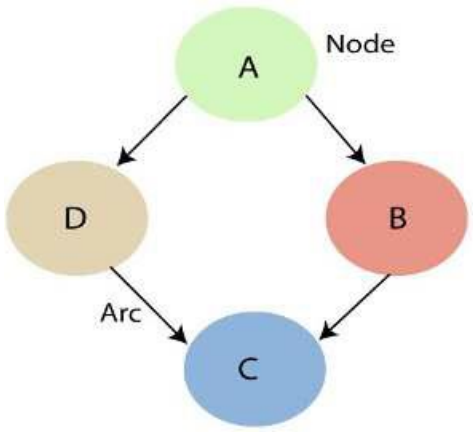
Bayesian Belief Network

Bayesian belief network is key computer technology for dealing with probabilistic events and
to solve a problem which has uncertainty. We can define a Bayesian network as:
"A Bayesian network is a probabilistic graphical model which represents a set of variables
and their conditional dependencies using a directed acyclic graph."
It is also called a **Bayes network, belief network, decision network**, or **Bayesian model**.

Bayesian networks are probabilistic, because these networks are built from a **probability**
distribution, and also use probability theory for prediction and anomaly detection.
Real world applications are probabilistic in nature, and to represent the relationship between
multiple events, we need a Bayesian network. It can also be used in various tasks including **prediction, anomaly detection, diagnostics, automated insight, reasoning, time**
series prediction, and **decision making under uncertainty**.
Bayesian Network can be used for building models from data and experts opinions, and it
consists of two parts:

- **Directed Acyclic Graph**
- **Table of conditional probabilities.**

The generalized form of Bayesian network that represents and solve decision problems under
uncertain knowledge is known as an **Influence diagram**.
A Bayesian network graph is made up of nodes and Arcs (directed links), where:



Each **node** corresponds to the random variables, and a variable can
be **continuous** or **discrete**.

- **Arc or directed arrows** represent the causal relationship or conditional probabilities
between random variables. These directed links or arrows connect the pair of nodes in
the graph.

These links represent that one node directly influence the other node, and if there is no directed link that means that nodes are independent with each other

- **In the above diagram, A, B, C, and D are random variables represented by the nodes of the network graph.**
- **If we are considering node B, which is connected with node A by a directed arrow, then node A is called the parent of Node B.**
- **Node C is independent of node A.**

The Bayesian network has mainly two components:

- **Causal Component**
- **Actual numbers**

Each node in the Bayesian network has condition probability distribution **$P(X_i |Parent(X_i))$**,
which determines the effect of the parent on that node.

Bayesian network is based on Joint probability distribution and conditional probability. So
let's first understand the joint probability distribution:

Joint probability distribution:

If we have variables x1, x2, x3,....., xn, then the probabilities of a different combination of x1,
x2, x3.. xn, are known as Joint probability distribution.

$P[x_1, x_2, x_3,....., x_n]$, it can be written as the following way in terms of the joint probability
distribution.

$= P[x_1| x_2, x_3,....., x_n]P[x_2, x_3,....., x_n]$

$= P[x_1| x_2, x_3,....., x_n]P[x_2|x_3,....., x_n]....P[x_{n-1}|x_n]P[x_n]$.

In general for each variable Xi, we can write the equation as:

$P(X_i|X_{i-1},........., X_1) = P(X_i |Parents(X_i))$

Explanation of Bayesian network:

Let's understand the Bayesian network through an example by creating a directed acyclic
graph:

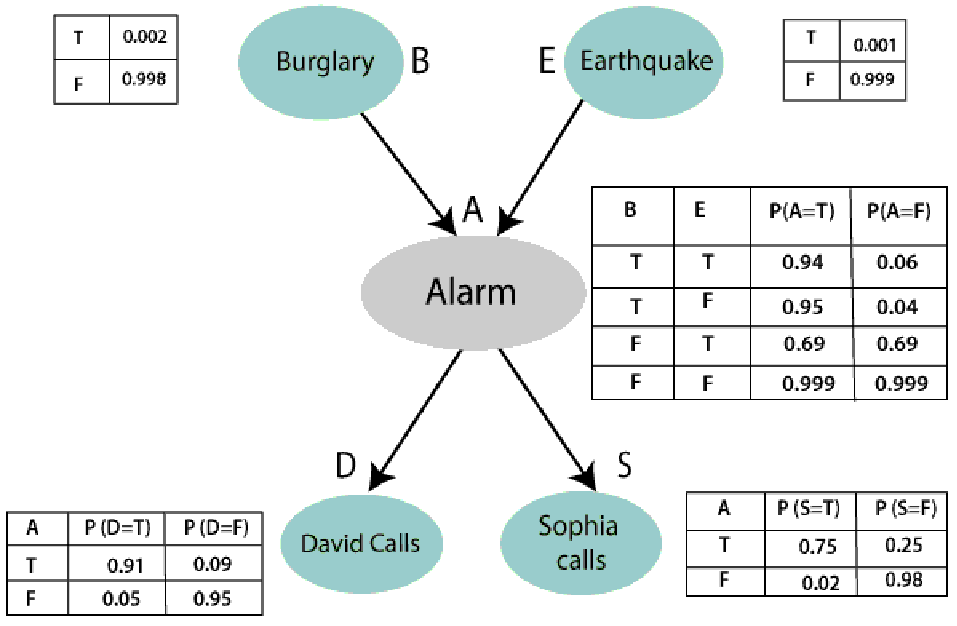
Problem:
Calculate the probability that alarm has sounded, but there is neither a burglary, nor an earthquake occurred, and David and Sophia both called the Harry.
Solution:

- The Bayesian network for the above problem is given below. The network structure is
showing that burglary and earthquake is the parent node of the alarm and directly

affecting the probability of alarm's going off, but David and Sophia's calls depend
on
alarm probability.

- The network is representing that our assumptions do not directly perceive the
burglary
and also do not notice the minor earthquake, and they also not confer before
calling.
- The conditional distributions for each node are given as conditional
probabilities table
or CPT.
- Each row in the CPT must be sum to 1 because all the entries in the table
represent an
exhaustive set of cases for the variable.

can write the events of problem statement in the form of probability: **P[D, S, A, B, E]**,
can rewrite the above probability statement using joint probability distribution:
P[D, S, A, B, E]= P[D | S, A, B, E]. P[S, A, B, E]
=P[D | S, A, B, E]. P[S | A, B, E]. P[A, B, E]
= P [D| A]. P [S| A, B, E]. P[A, B, E]
= P[D | A]. P[S | A]. P[A| B, E]. P[B, E]
= P[D | A]. P[S | A]. P[A| B, E]. P[B |E]. P[E]



T	0.002
F	0.998

T	0.001
F	0.999

B	E	P(A=T)	P(A=F)
T	T	0.94	0.06
T	F	0.95	0.04
F	T	0.69	0.69
F	F	0.999	0.999

A	P (D=T)	P (D=F)
T	0.91	0.09
F	0.05	0.95

A	P (S=T)	P (S=F)
T	0.75	0.25
F	0.02	0.98

Let's take
the observed probability for the Burglary and earthquake component:

P(B= True) = 0.002, which is the probability of burglary.

P(B= False)= 0.998, which is the probability of no burglary.

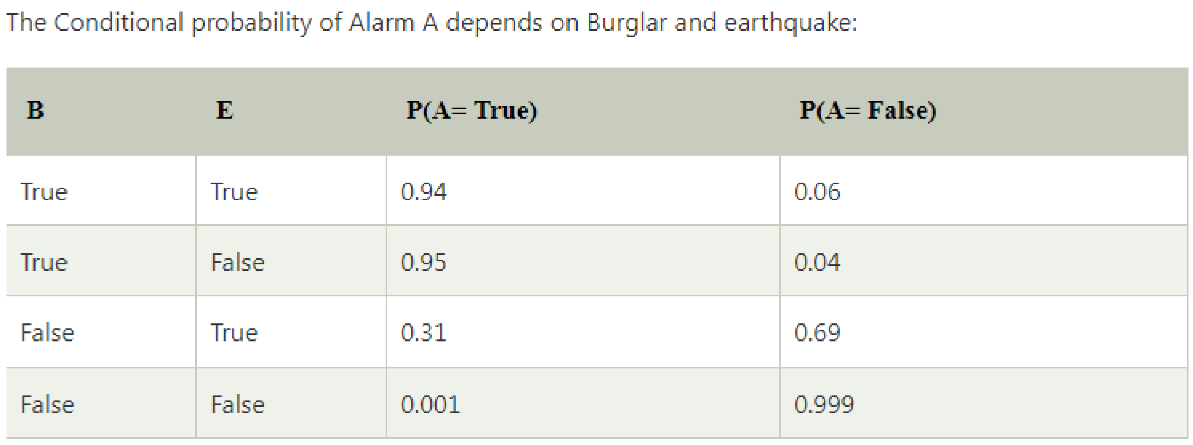
P(E= True)= 0.001, which is the probability of a minor earthquake

P(E= False)= 0.999, Which is the probability that an earthquake not occurred.

We can provide the conditional probabilities as per the below tables:

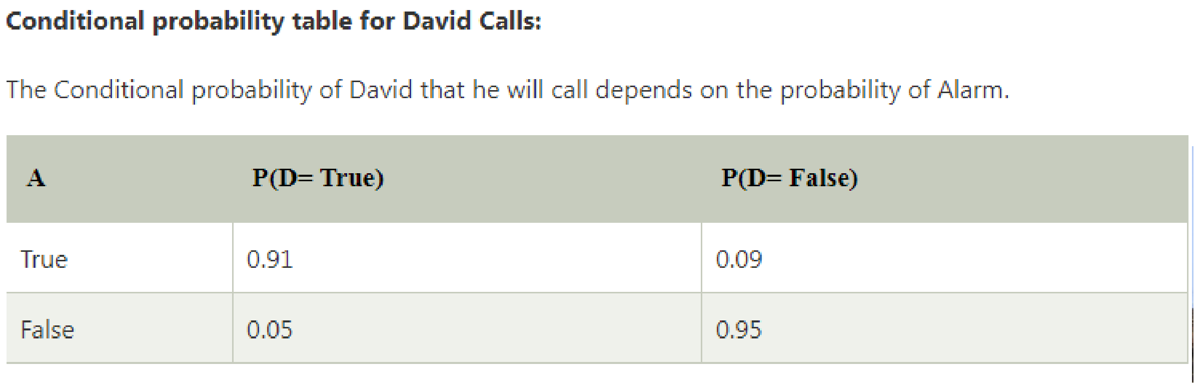
Conditional probability table for Alarm A:

The Conditional probability of Alarm A depends on Burglar and earthquake



The Conditional probability of Alarm A depends on Burglar and earthquake:

B	E	P(A= True)	P(A= False)
True	True	0.94	0.06
True	False	0.95	0.04
False	True	0.31	0.69
False	False	0.001	0.999



Conditional probability table for David Calls:

The Conditional probability of David that he will call depends on the probability of Alarm.

A	P(D= True)	P(D= False)
True	0.91	0.09
False	0.05	0.95

Conditional probability table for Sophia Calls:

The Conditional probability of Sophia that she calls is depending on its Parent Node "Alarm."

A	P(S= True)	P(S= False)
True	0.75	0.25
False	0.02	0.98

From the formula of joint distribution, we can write the problem statement in the form of probability distribution:

P(S, D, A, ¬B, ¬E) = P (S|A) *P (D|A)*P (A|¬B ^ ¬E) *P (¬B) *P (¬E).

= 0.75* 0.91* 0.001* 0.998*0.999

= 0.00068045.

Hence, a Bayesian network can answer any query about the domain by using Joint distribution.

Gibbs Sampling

- Fix the values of observed variables
- Set the values of all non observed variables randomly
- Perform a random walk through the space of complete variable assignments.   On each move
- 1. pick a variable X
- 2. calculate Pr (X=true   all other variables)
- 3. set X to true with that probability

Repeat many times. Frequency with which any variable X is true is its posterior probability

Converges to true posterior when frequencies stop changing significantly

- Stationary distribution, mixing

calculate Pr (X=true   all other variables)

Recall: a variable is independent of all others given its Markov Blanket

- parents   - children - other parents of children

So problem becomes calculating Pr(X=true   MB(x))

- we solve this sub-problem exactly

- fortunately, it is easy to solve

P(X)= α P(X| Parents(X)) ╥ P(Y| Parents(Y))

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

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

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

= αP(X|A)P(B|X,C)

P(X|E) = $\frac{number\ of\ samples\ with\ X=x}{total\ number\ of\ samples}$

Advantages

- No samples are discarded

- No problem with samples of low weight

Can be implemented very efficiently

- **Disadvantages**
- **Can get stuck if relationship between two variables**
- **Many variations have been devised to make MCMCM**

Markov Random fields

A Markov random Field is an undirected graph

where each node captures the (discrete or Gaussian) probability distribution of a variable and

the edges represent dependencies between those variables and are weighted to represent the relative strengths of the dependencies.

The defining difference between a Bayesian network and a Markov random field is that a Bayesian network is directed while a Markov random field is undirected.

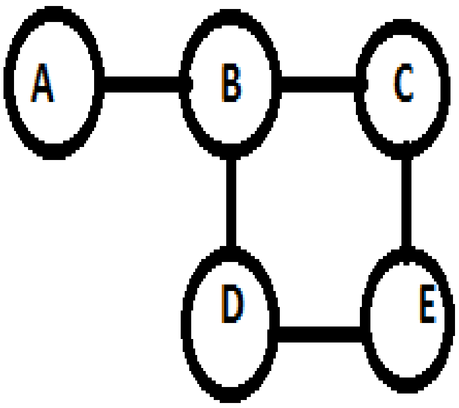
a hidden Markov model is directed and thus a sub-type of Bayesian network rather than a sub-type of Markov random field.

A **conditional random field** is a Markov random field that has specifically been set up to allow the values of a specific variable or variables to be predicted based on the values of the other variables

Undirected graphical models edge represents the potential between two variables, syntactically, Factorization distribution probabilities between variable.

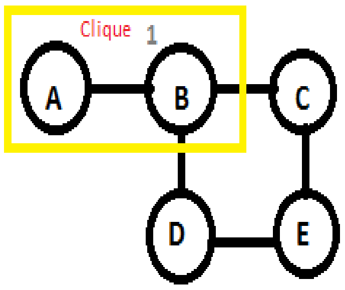
CPDs for Bayesian networks, we have tables to incorporate relations between nodes in **Markov networks**.

However, there are two crucial differences between these tables and CPDs.



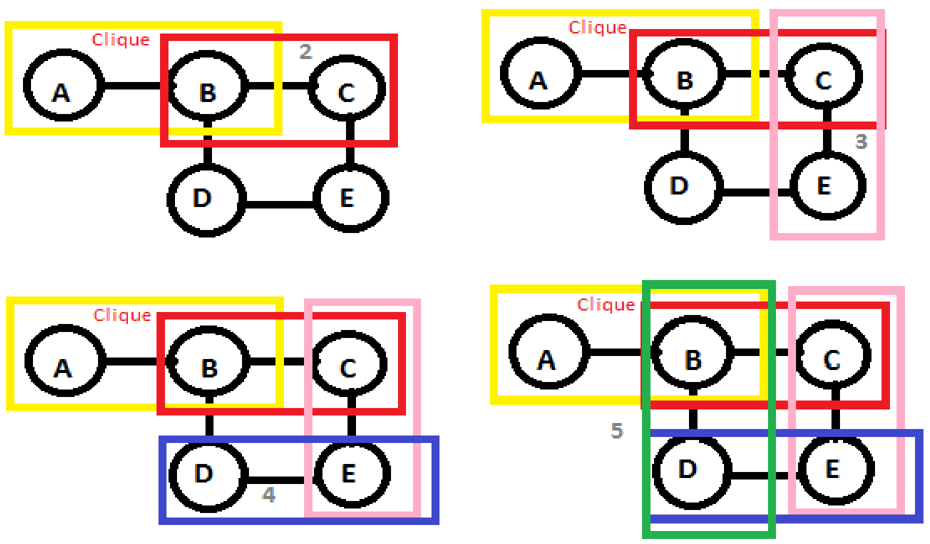
Clique in graph theory.it is a subset of vertices of an undirected graph.

P(A, B, C, D, E) α Φ(A,B) Φ (B,C) Φ (B,D) Φ (C,E) Φ (D,E)

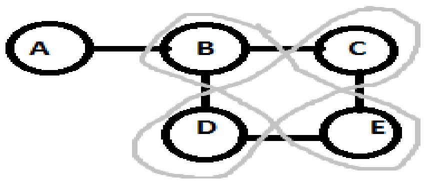


P(X) = $\frac{1}{z}\Pi_{C eclique}\phi_C(x_C)$ **(Potential functions)**

Such that: It induces sub graph is complete in every vertices in a clique is adjacent. So, clique in this graph adjust adjacently one by one.

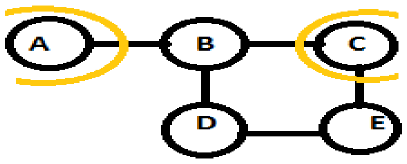


It is some different if we join **D,C** and **B,E** clique over here then it is also change its probability.



P(A, B, C, D, E) α Φ(A,B) Φ (B,C,D) Φ (C,D,E)

Some undirected graphic model has Markov Random Field. In MRF certain paths between **A** and **C**.



```
A  -> B  -> C
A  -> B  -> D  -> E  -> C
```

Independence properties such as Markov properties

Any 2 subsets if variables are conditionally independent given a separating subset.

- ✓ If we take **'A'** as a subset and **'C'** as one subset then there is wore between them. So, there is no way to go

  between **'A'** and **'C'** without getting threw the subset. So, we are using **(A, B)** than **B, C, D, E**.

Therefore, A and C are separating subsets

- ✓ Any 2 subset of variable are conditionally independent given a separating subset.
- ✓ {B,D}, {B,E} and {B,D,E} are separating subsets.

Hidden Markov Model

HMM is specified by a setoff states Q, a set of transition probabilities A, a set of observation likelihoods B, A defined start state and end state(s), and a set of observation symbols O, is not Constructed from the state set Q which means observations may be disjoint from states.

AccordingtoSajja [2012]aMarkovchainisaspecialcaseofaweightedautomatonin which the input sequence uniquely determinesstates the automaton will go through for that input sequence. Since they can’t represent inherently ambiguous problems. Markov chain is only useful for assigning probabilities to unambiguous sequences.

AMarkov chain is specifiedbythefollowing components:

$Q= q_1 q_2 .. .q_N$ aset ofstates

$A = [a_{ij}]_{N\times N}$ a transition probability matrix A, each a_{ij}representing the probability of moving from state i to state j.

q_0,q_{end} a special start state and end state which are not associated with observations.

A Markov chain embodies an important assumption about these probabilities In a first-order Markov chain, the probabilityof a particular state is dependent onlyon the

immediateprevious state, Symbolically

Markov Assump玲椀on: $P(q_i | q_1 ... q_{i-1}) = P(q_i | q_{i-1})$

Sinceeacha_{ij}expressestheprobability$p(q_j|q_i)$,thelawsofprobabilityrequirethatthe values
of the probabilities for some state must sum to 1.
·= ·1, ·2, ..., ·$_N$ an initial probability distribution over states. ·$_i$is the probabilitythattheMarkovchainwillstartinstate$i$.Some states$j$mayhave ·$_j$=0,meaningthattheycannotbeinitial states.
Each ·$_i$expressestheprobability$p(q_i|START)$,allthe ·probabilitiesmustsumto1:

Markov chain is useful when we need to compute the probability for a sequence of events that we can observe in the world. It is useful even in events in which one is interested, but may not be directly observable in the world. For example forNER named entities tagsare not observed in real world as we observe sequence of various weather cold,hot,rain.ButsequenceofwordsobservedinrealworldandNERhasto infer the correct named entities tags from these word sequence.The named entity tags are said to be hidden because they are not observed.

HMM Assumptions

MarkovAssumption:Afirst-orderHiddenMarkovModelinstantiatestwosimplifying assumptions. One: as incase of first-order Markov chain, the probability of a particular state is dependent only on the immediate previous state:
$P(q_i|q_1 \ldots q_{i-1}) = P(q_i|q_{i-1})$
Independent Assumption: The probability of an output observation $o_i$is dependent only on the state that produced the observation q_i, and not on any other state or any
other observations:
$P(o_i|q_1 \ldots q_i, \ldots, q_n, o_1, \ldots, o_i, \ldots, o_n) = P(o_i|q_i)$
*TheStationaryAssumption*Hereitisassumedthatstatetransitionprobabilitiesare independentofthe actualtimeatwhichthetransitionstakeplace. Mathematically,
$P(q_{t1+1} = j \mid q_{t1} = i) = P(q_{t2+1} = j | q_{t2} = i)$fortime $t_1$and t_2
Having described the structure of an HMM, it will be logical to introduce the algorithms for computing things with them. An influential tutorial byRabiner [1989],

introduced the idea that Hidden Markov Models should be characterized by three fundamental problems: *Evaluation problem* can be used for isolated (word) recognition. *Decoding problem* is related to the continuous recognition as well as to the segmentation. *Learning problem* must be solved, if we want to train an HMM for
the subsequent use of recognition tasks.
*Evaluation:*Byevaluationitismeantwhatistheprobabilitythataparticular sequenceofsymbolsisproducedbyaparticularmodel?ThatisgivenanHMM ·=

(A,B) and an observation sequence O, to determine the likelihood $P(O| \cdot)$. For evaluation followingtwo algorithms are specified : the *forward algorithm* or the *backwards algorithm,* it may be notedthis does not mean the forward-backward algorithm.

Learning Generally, the learning problem is the adjustment of the HMM parameters,
so that the given set of observations (called the *training set*) is represented by the model in the best way for the intended application. More explicitly given an observation sequence O and the set of states in the HMM, learn the HMM parameters
A and B. Thus it would be clear that the ``quantity'' which is to be optimized during
the learning process can be different from application to application. In other words
there may be several *optimization criteria* for learning, out of them a suitable one is selected depending on the application.
Inessenceitistofindthemodelthatbestfitsthedata.Forobtainingthedesired model the
following 3 algorithms are applied:
· MLE(maximumlikelihood estimation)
· Viterbitraining(differentfromViterbidecoding)
· BaumWelch=forward-backwardalgorithm

AdvantagesandDisadvantagesof HMM

・Theunderlyingtheoreticalbasisismuchmoresound,elegantandeasyto understand.

・Itiseasiertoimplementandanalyze.

・HMM taggers are very simple to train (just need to compile counts from thetraining corpus).

・Performsrelativelywell(over 90%performanceonnamedentities).

・StatisticiansarecomfortablewiththetheoreticalbaseofHMM.

・Libertytomanipulatethe trainingandverification processes.

・Mathematical/theoreticalanalysisoftheresultsandprocesses.

・Incorporatespriorknowledgeintothearchitecturewithgooddesign.

・Initializethemodelclosetosomething believedtobe correct.

・Iteliminateslabelbiasproblem.

It has also been proved effective for a number of other tasks, such as speechrecognition, handwriting recognitionand sign language recognition.

・Because each HMM uses onlypositive data, they scale well; since new words can be added without affecting learnt HMMs.

Disadvantages:

・In order to define joint probabilityover observation and label sequence HMM needs to enumerate all possible observation sequences.

・Maindifficultyis modeling probabilityofassigning atagto wordcan be very difficult if "words" are complex.

・Itisnotpracticaltorepresentmultipleoverlappingfeaturesandlongterm dependencies.

・Number of parameters to be evaluated is huge. So it needs a large data set for training.

・Itrequires hugeamountoftraininginordertoobtainbetter results.

・HMMs onlyuse positive data to train. In other words, HMM training involves maximizing the observed probabilities for examples belonging to a class. But it does not minimize the probability of observation of instances from other classes.

・It adopts the Markovian assumption: that the emission and the transition probabilities depend only on the current state, which does not map well to many real-world domains;

Tracking Methods

Object tracking aims at estimating bounding boxes and the identities
of objects in videos. It takes in a set of initial object detection, develops a visual model for the
objects, and tracks the objects as they move around in a video. Furthermore, object tracking
enables us to assign a unique ID to each tracked object, making it possible for us to count
unique objects in a video. Often, there's an indication around the object being tracked, for
example, a surrounding square that follows the object, showing the user where the object is

Object tracking has a wide range of applications in computer vision, such as surveillance,
human-computer interaction, traffic flow monitoring, human activity recognition.

Simple Online And Realtime Tracking (SORT)

Simple Online And Realtime Tracking (SORT) is a lean implementation of a tracking-by
detection framework. Keeping in line with Occam's Razor, it ignores appearance features
beyond the detection component. SORT uses the position and size of the bounding boxes for
both motion estimation and data association through frames. Faster RCNN is used as the
object detector. The displacement of objects in the consecutive frames is estimated by a linear
constant velocity model which is independent of other objects and camera motion. The state of
each target is defined as x = [u, v, s, r, u,' v,' s'] where (u,v) represents the center of the
bounding box r and u indicate scale and aspect ratio. The other variables are the respective
velocities.

For the ID assignment, i.e., data association task the new target states are used to predict the
bounding boxes that are later on compared with the detected boxes in the current timeframe.
The IOU metric and the Hungarian algorithm are utilized for choosing the optimum box to
pass on the identity. SORT achieves **74.6 MOTA** and 76.9 IDF1 on the MOT17 dataset with
291 ID switches and **30 FPS**.

DeepSORT

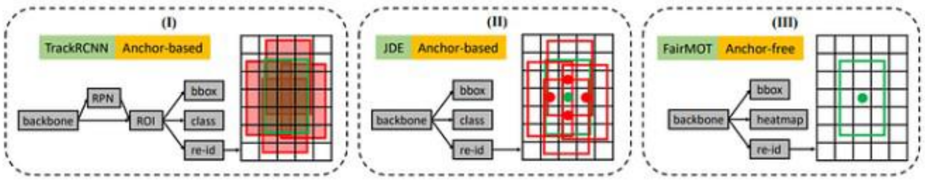
Despite achieving overall good performance in terms of tracking precision and accuracy,
SORT has a high number of identity switches. It fails in many of the challenging scenarios
like occlusions, different camera angles, etc. To overcome these
limitations DeepSORT replaces the association metric with a more informed metric that
combines motion and appearance information. In particular, a “deep appearance” distance
metric is added. The core idea is to obtain a vector that can be used to represent a given image.

To do this DeepSORT creates a classifier and strips the final classification layer, this leaves us
with a dense layer that produces a single feature vector.
In addition to that, DeepSORT adds extra dimensions to its motion tracking model. The state
of each target is denoted on the eight-dimensional state space (u, v, γ, h, x,˙ y,˙ γ, ˙ h˙) that
contains the bounding box center position (u, v), aspect ratio γ, height h, and their respective
velocities in image coordinates. These additions enable DeepSORT to effectively handle
challenging scenarios and reduce the number of identity switches by 45%.
DeepSORT

achieves **75.4 MOTA** and 77.2 IDF1 on the MOT17 dataset with 239 ID switches but a
lower **FPS of 13**.

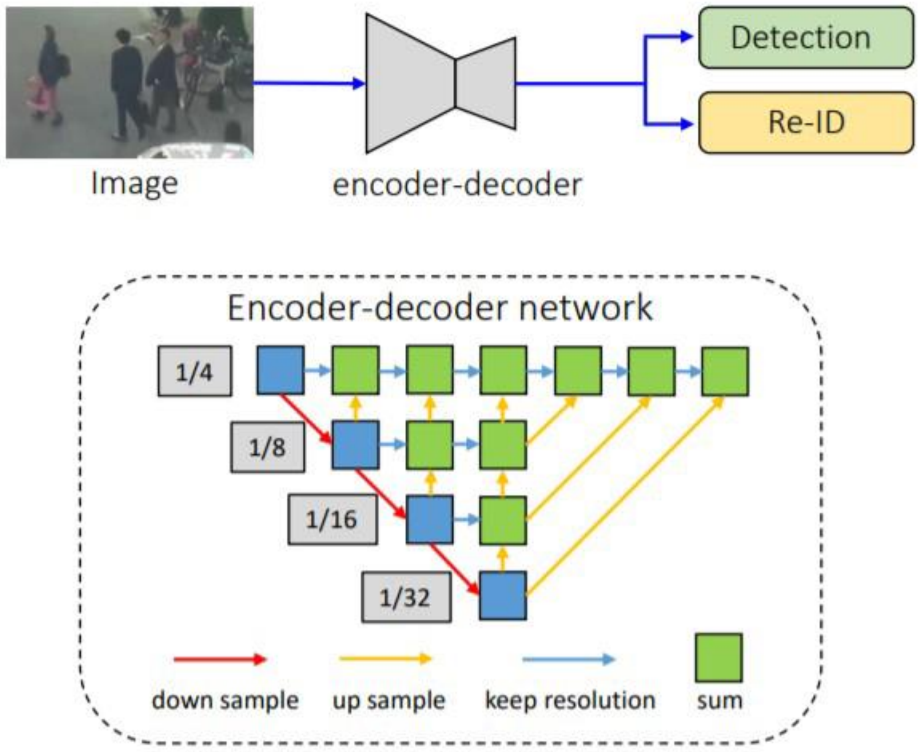
FairMOT

Approaching object tracking from a multi-task learning perspective of object detection and re-
ID in a single network is appealing since it allows shared optimization of the two tasks.
However, the two tasks tend to compete with each other. In particular, previous works usually
treat re-ID(re-Identification) as a secondary task whose accuracy is heavily affected by the
primary detection task. As a result, the network is biased to the primary detection task which is
not fair to the re-ID task.



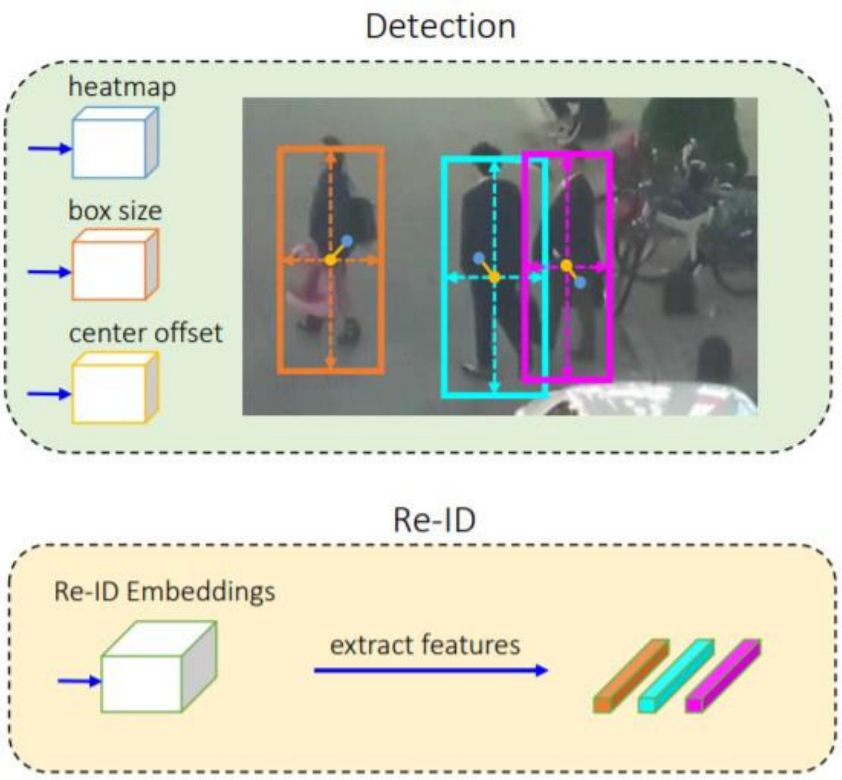
Comparison of the existing one-shot trackers and FairMOT. The three methods extract re-ID features differently. TrackR-CNN extracts re-ID features for all positive anchors using ROI-Align. JDE extracts re-ID features at the centers of all positive anchors. FairMOT extracts re-ID features at the object center

FairMOT is a new tracking approach built on top of the anchor-free object detection
architecture CenterNet. The detection and re-ID tasks are treated equally in FairMOT which
essentially differs from the previous "detection first, re-ID secondary"
frameworks. It has a simple network structure that consists of two homogeneous branches for detecting objects and
extracting re-ID features.



Overview of FairMOT and its encoder-decoder network

FairMOT adopts ResNet-34 as the backbone as it offers a good balance between accuracy and speed. An enhanced version of Deep Layer Aggregation (DLA) is applied to the backbone to fuse multi-layer. In addition, convolution layers in all up-sampling modules are replaced by deformable convolutions so that they can dynamically adjust the receptive field according to object scales and poses. These modifications also help to solve the alignment issue.



Detection and Re-ID branches

The detection branch is built on top of CenterNet, three parallel heads are appended to DLA-
34 to estimate heatmaps, object center offsets, and bounding box sizes, respectively. The Re-
ID branch aims to generate features that can distinguish objects. Ideally, affinity among
different objects should be smaller than that between the same objects. To achieve this goal,
FairMOT applies a convolution layer with 128 kernels on top of backbone features to extract
re-ID features for each location. The re-ID features are learned through a classification task.
All object instances of the same identity in the training set are treated as the same class. All
these optimizations help FairMOT achieve **77.2 MOTA** and 79.8 IDF1 on the MOT17 dataset
with **25.9 FPS** running speed.

TransMOT

Advances in deep learning have inspired us to learn spatial-temporal relationships using deep
learning. Especially, the success of Transformer suggests a new paradigm of modeling
temporal dependencies through the powerful self-attention mechanism. But transformer-based
trackers have not been able to achieve state-of-the-art performance for several reasons

Videos can contain a large number of objects. Modeling the spatial-temporal relationships
of these objects with a general Transformer is inefficient because it does not take the
spatial-temporal structure of the objects into consideration.
· It requires a lot of computation resources and data for a transformer to model long-term
temporal dependencies.
· The DETR-based object detector used in these works is still not state-of-the-art.
· TransMOT is a new spatial-temporal graph Transformer that solves all these issues. It
arranges the trajectories of all the tracked objects as a series of sparse weighted graphs
that are constructed using the spatial relationships of the targets. TransMOT then uses
these graphs to create a spatial graph transformer encoder layer, a temporal transformer
encoder layer, and a spatial transformer decoder layer to model the spatial-temporal
relationships of the objects. The sparsity of the weighted graph representations makes
it more computationally efficient during training and inference. It is also a more effective model than a regular transformer because it exploits the structure of the objects.

· To further improve the tracking speed and accuracy, TransMOT introduces a cascade
association framework to handle low-score detections and long-term occlusions.
TransMOT can also be combined with different object detectors or visual feature
extraction sub-networks to form a unified end-to-end solution. This enables developers
to leverage state-of-the-art object detectors for object tracking. TransMOT
achieves **76.7 MOTA** and 75.1 IDF1 on the MOT17 dataset with **9.6 FPS**.
· **ByteTrack**
· Most tracking methods obtain identities by associating detection boxes with scores
higher than a threshold. The objects with low detection scores are simply ignored,

which brings non-negligible true object missing and fragmented trajectories. BYTE is
an effective association method that utilizes all detection boxes from high scores to
low ones in the matching process.