BLEU Score [Bilingual Evaluation Understudy]

→ A method for Automatic Evaluation of Machine translation (M/c tran)

→ BLEU is a score for comparing a Candidate translation of text to one or more reference translations (human translation)

→ It can also be used to evaluate text generated for a suite of NLP tasks (like language generation, image caption generation, text summarization, speech recognition)

→ BLEU is a metric for evaluating a generated sentence (after translation) to a reference sentence

→ BLEU Score can be used to evaluate the performance of a Machine translation System particularly when there are multiple equally good answer

→ BLEU metric ranges from 0 to 1

(range) Score = 1.0 – A perfect Match results
Score = 0.0 – A perfect MisMatch results

→ BLEU score was developed for evaluating the predictions made by automatic Machine translations system, but it is not perfect evaluation method.

~~A~~ Benefits of BLEU score

1) Quick & inexpensive to calculate
2) Easy to Understand
3) Language independent
4) It correlates highly with human evaluation
5) Widely adopted method.

→ Machine translation evaluation system require
i) A numerical 'translation closeness' Metric
ii) A good quality human reference translations

→ BLEU score is based on "text string Matches".

OR
⇒ It is a score that quantifies how good machine translation is by computing a

similarity score based on n-gram precision

(n gram) Precision – look at each word in the candidate sentence & assign it a score of 1 if it shows up in any of the reference sentence otherwise 0.

Ex MT output – The cat the cat on the mat
Reference 1: There is a cat on the mat
Reference 2: The cat is on the mat

i) n gram :- Sequence of n consecutive words
Like
i) Unigram – one word Ex → hello
ii) bigram – two word Ex → how how
iii) trigram – three word sequence of words
Ex → hello how are
iv) 4-gram – four word seq. of words
Ex → hello how are you

The main task of BLEU implements is to compare n-gram of the candidate with two n-grams of the reference translation & count the number of matches

→ Here the nos. of matches are position independent

Ex
State / Attribute Behaviour

→ If there are more nos. of matches b/w candidate & reference translation, then better is the translation

→ Higher the BLEU score better is the translation

→ More reference human translations results in better & accurate scores.

Remaining Next Diary

Unit-4

ii) Unigram Precision

Unique Unigram	Count / Score
the	3
cat	2
on	1
mat	1
total counts for the unique unigram in the candidate sentence =	7 (It should be b/w 0 & 1)

Candidate Sentence

the cat the cat on the mat
1 2 3 4 5 6 7

Nos. of unigram in candidate sentence = 7

Unigram precision $= \frac{7}{7} = 1.0$ (normalize)

7 → total nos. of words in CS.

ii) Bigram Precision

Candidate Sentence

the cat the cat on the mat

Bigram - the cat, ~~the~~ cat the, cat on, on the, the mat = 5
1 2 3 4 5

Unique Bigram	Count / Score
the cat	2
cat the	1
cat on	1
on the	1
the mat	1
total cost	6

Bigram precision $= \frac{5}{6}$

$= 0.833$

★ Drawback

Unigram precision is very easy to be over-confident about the quality of machine translation.

★ Example of poor MT output with high precision

Candidate Sentence : the the the the the the the

Unigram precision $= \frac{7}{7} = 1.0$

→ To overcome the unigram precision problem, clipped count & modified precision were proposed

★ Modified N-gram precision

Candidate – the cat the cat on the mat
Reference1 – there is a cat on the mat
Reference2 – the cat is on the mat

i) Unique modified n-gram precision.

Reference	Unique Unigram	Count	Clipped Count
2	the	3	2
1	cat	2	1
1	on	1	1
1	mat	1	1
	total clipped count	7	5

$CR_{max} < Count$
Clipped Count $= CR_{max}$

Unigram modified precision $= \frac{5}{7} = 0.714$

ii)

Reference	Unique Bigram	Count	Clipped Count
1	the cat	2	1
0	cat the	01	0
1	cat on	1	1
1	on the	1	1
1	the mat	1	1
		6	6

Bigram MP $= \frac{4}{6}$

$= 0.667$

iii) Modified n-gram precision

Candidate Sentence: the the the the the the the
Reference Sentence 1: there is a cat on the mat
Reference Sentence 2: the cat is on the mat

~~Unique~~ Unigram modified precision

Unique Unigram	Count	Clipped Count	Ref/max
the	7	2	2

precision $= \frac{2}{7}$

Problem in Modified N-gram precision

-> BLEU scores tend to favor short translations, which can produce very high precision scores, even using modified precision

Ex
Candidate sentence - the cat
Reference sentence - the cat is on the mat

Modified uniform precision $= \frac{1+1}{2} = 1$ → higher precision

Solution is to Brevity Penalty

$$BP = \begin{cases} 1 & \text{if } c > r \\ e^{(1-r/c)} & \text{if } c \le r \end{cases}$$

r → length of reference sentence
c = length of candidate sentence

For above example
$c = 2$, $r = 6$
as $c < r$ $BP = e^{(1-6/2)} = e^{(1-3)} = e^{-2}$

Bleu score
$$= BP \times \exp\left[\frac{1}{N}\sum_{n=1}^{N} P_n\right]$$
exponential
Geometric Mean of all ngram precision

P_n → Modified precision for n-gram
If using bigram, unigram, trigram, N=3
-> Bleu only utilizes a BP for shorter sentence

-> The redudant & ~~layer~~ larger sentences are not penalized properly

SAGE UNIVERSITY | SIRT SAGAR GROUP OF INSTITUTIONS | SAGE SCHOOL | M/s Agrawal Construction Co. | Apollo SAGE HOSPITALS BHOPAL | AGRAWAL POWER PVT. LTD | True Sage Foundation

SAGE UNIVERSITY | SIRT SAGAR GROUP OF INSTITUTIONS | SAGE SCHOOL | M/s Agrawal Construction Co. | Apollo SAGE HOSPITALS BHOPAL | AGRAWAL POWER PVT. LTD | True Sage Foundation

Attention Model

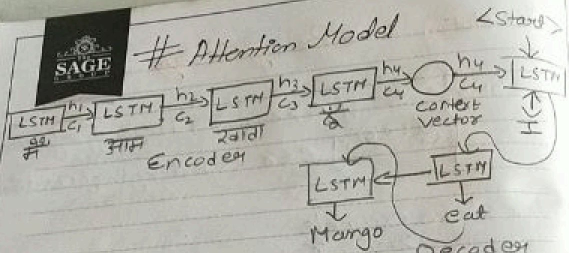


Fig → Translation using Encoder-Decoder

* Context vector (h_4, C_4) → Contains good summary of input sequence { मैं आम खाता हूँ }
(fixed size)

→ For long sentences (input) context vector may not have all necessary information.

→ To solve this problem, attention mechanism is used with Encoder-Decoder, RNN/LSTM

* Attention Model

→ Attention means directing your focus at something & taking greater notice

OR

Concentrating on one or a few things while ignoring others.

→ The attention mechanism is an improvement over the encoder-decoder based on neural based machine translation system in NLP

→ In encoder-decoder RNN/LSTM
⇒ Encoder process the entire input sentence & encode it into a context vector which is the last hidden state of LSTM/RNN

Ex मैं आम खाता हूँ

t1 - मैं - [1 0 0 0] - I
t2 - खाता हूँ [0 0 0.5 0.5] - eat
t3 आम [0 1 0 0] - Mango

α_{ij} : i → importance of word
j → timestamp

	मैं	आम	खाता	है
t1	1	0	0	0
	α_{11}	α_{21}	α_{31}	α_{41}
t2	0	0	0.5	0.5
	α_{12}	α_{22}	α_{32}	α_{42}

α_{12} → 1st word of the sequence at time stamp 2

SAGE UNIVERSITY | SIRT SAGAR GROUP OF INSTITUTIONS | SAGE SCHOOL | M/s Agrawal Construction Co. | Apollo SAGE HOSPITALS BHOPAL | AGRAWAL POWER PVT. LTD. | The Sage Foundation

t_3	मैं	आम	खाता	हूँ
	0	1	0	0
	α_{13}	α_{23}	α_{33}	α_{43}

→ It is expected that context vector contains good summary of input sequence (information of the whole sentence). All the intermediate states of the encoder are ignored, & the final state is supposed to be the initial hidden state of the decoder.

→ Decoder units produce the words in a sentence one after another.

★ Drawback of Encoder-decoder Machine translation system

→ If the encoder makes a bad summary the translation will also be bad.

→ Usually encoder creates a bad summary when it tries to understand a large sentences

→ RNNs cannot remember long sentence & sequences due to vanishing/exploding gradient problem. It can remember the parts which it has just seen.

→ So the performance of the encoder-decoder network degrades rapidly as the length of the input sentence increases.

→ LSTM is supposed to do better than RNN, but it become forgetful in specific cases

→ But in both RNN & LSTM there is no mechanism to give more importance to some of the input words compared to others while translating the sentence

→ So when model generates a sentence (machine translation) it searches for a set of positions in the encoder hidden states where the most relevant information is available. This idea is called Attention

★ Attention is proposed as a solution to the limitation of encoder-decoder model encoding the input sequence to one fixed length vector from which to decode each output time step. This issue is believed to be more of a problem when decoding long sequences.

→ The attention model allows an RNN to pay attention to specific parts of the input that is considered as being important which improves the performance of the resulting model in practice.

→ Instead of sending last hidden state output of decoder, we could take a weighted average of the corresponding word representation & feed it to the decoder.

→ For ex, at timestamp 2, we can take a weighted average of the representation खाता है to get output 'eat' from the decoder
By doing this we are not overloading → the decoder with irrelevant information (like मैं आम)

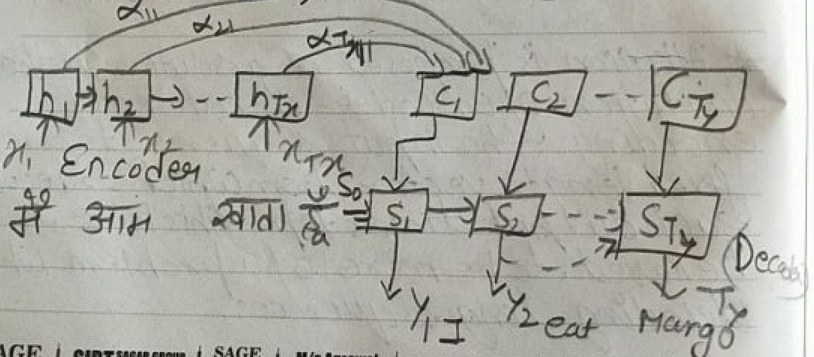


Fig → Encoder-Decoder Model with Attention Mechanism

Prediction of II word

$\boxed{t=2}$

$$C_2 = \alpha_{12} \cdot h_1 + \alpha_{22} \cdot h_2 + \alpha_{32} \cdot h_3 + \alpha_{42} \cdot h_4$$

$$a_{12} = a(s_1, h_1) \quad \alpha_{12} = \frac{e^{a_{12}}}{e^{a_{12}} + e^{a_{22}} + e^{a_{32}} + e^{a_{42}}}$$

$$a_{22} = a(s_1, h_2) \quad \alpha_{22} = \frac{e^{a_{22}}}{e^{a_{12}} + e^{a_{22}} + e^{a_{32}} + e^{a_{42}}}$$

$$a_{32} = a(s_1, h_3) \; \ldots \ldots$$

$$C_t = \sum_{j=1}^{T_x} \alpha_{jt} h_j$$

Context vector

h_j = encoder hidden state at j^{th} timestamp

α_{jt} – Probablity of focusing on the j^{th} word to produce t^{th} word.

$$\cancel{\alpha_{jt} \propto \frac{e^{a_{jt}}}{\sum_{k=1}^{T} e^{a_{kt}}}}$$

i/p word nos

o/p word nos

$$a_{jt} = a(S_{t-1}, h_j)$$

→ feed forward Neural Network

↑ tanh

→ subscript

$$\alpha_{jt} = \frac{\exp(e_{jt})}{\sum_{k=1}^{T_x} \exp(e_{kt})}$$

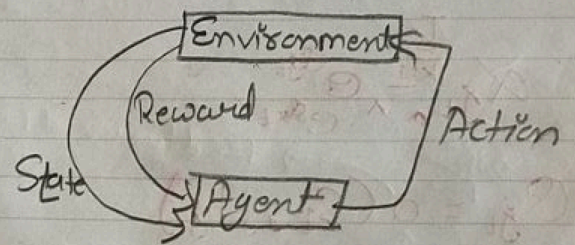
→ The importance of the j^{th} input word for decoding the t^{th} output word can be calculated by

$$e_{jt} = f_{ATT}(S_{t-1}, h_j)$$
or a

a / f_{ATT} = feed forward neural network

The alignment model scores (e) how well each encoded input (h) matches the current output of the decoder (S)

Reinforcement Learning



→ Reinforcement learning an area of machine learning in which computer/software agent learns to perform a task through repeated trial & error interactions with a dynamic environment

→ Agent is awarded or penalized with a point for a correct or a wrong answer

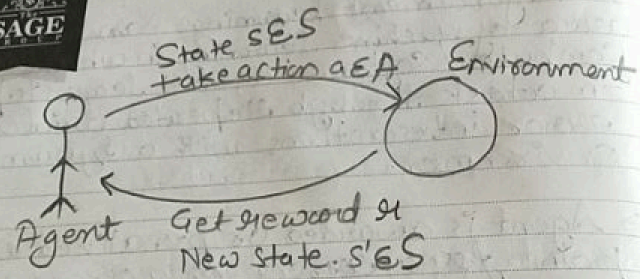
→ On the basis of the positive reward points gained the model trains itself.

→ The agent is acting in an environment. The agent can stay in one of many states ($s \in S$) of the environment & choose to take one of many actions ($a \in A$) to switch from one state to another. Which state the agent will arrive is decided by transition probablities between states (P).

→ Once an action is taken, the environment delivers reward ($r \in R$) as feedback

P.T.O.

SAGE UNIVERSITY | SIRT SAGAR GROUP OF INSTITUTIONS | SAGE SCHOOL | M/s Agrawal Construction Co. | Apollo SAGE HOSPITALS BHOPAL | AGRAWAL POWER PVT. LTD. | The Sage Foundation



→ In RL agent, interacts with the environment trying to take smart actions to maximize cumulative rewards

* Goal of RL

Find a optimal policy that maximize total cumulative rewards of the agent

The interaction b/w the agent & the environment involves a sequence of actions & observed rewards in time $t = 1, 2, \ldots T$.

→ During the process, the agent accumulates the knowledge about the environment learns the optimal policy & makes decisions on which action to take next so as to efficiently learns the best policy

→ RL used in video games, computer games etc.

* Terms used in Reinforcement learning

1) Agent → An entity ~~task~~ that takes action

2) ~~Agent~~ Action (A) → It is the move that an agent makes in a given state in the environment
→ Usually there is a list of discrete possible actions that agent chooses (move left, right, up, down, jump etc).

A → set of all actions
a → a particular actions

3) State (s) → A state is used to represent current situation of the environment from the agent's view (like obstacles, enemies etc).

S → set of all the states
s → a particular state.

4) Reward (r) → A reward is the immediate feedback given by environment to the agent for its action in the given state.

mario touches a coin - he wins points
mario touches enemy - he dies)

$R \rightarrow$ total or cumulative reward
$r \rightarrow$ immediate rewards received after action a..

5) Environment :-
The physical world in which the agent interacts.

→ Environment takes the agents current state & action as input & return agents reward & its next state as output

6) Policy (π) :-
Method to map agent's state to actions or the strategy that the agent employs to determine its action based on the current state.

→ The agent's policy $\pi(s)$ provide the guideline on what is the optimal action to take in a certain state with the goal to maximize the total rewards

7) Value - Future reward that an agent would receive by taking an action in a particular state

{Total amount of reward the agent expect
cumulative reward = wining the game}

Ex PacMan game
Agent → Pacman
Environment - grid world
States - location of pacman in the grid world
Reward - Eating food
Punishment (Killed by ghost) - loses the game.

Markov Decision Process (MDP)

→ MDP is a mathematical framework that can solve most Reinforcement learning problems with discrete actions.

→ With MDP, an agent can arrive at an optimal policy for maximum rewards over time.

→ Any reinforcement learning task composed of a set of states, actions & rewards that follows the markov property would be considered as Markov Decision process

SAGE UNIVERSITY | SIRT SAGAR GROUP OF INSTITUTIONS | SAGE SCHOOL | M/s Agrawal Construction Co. | Apollo SAGE HOSPITALS BHOPAL | AGRAWAL POWER PVT. LTD. | The Sage Foundation

→ To make a decision about what action to take at time $t+1$, we only have to consider the environment at time t & no more (Markov property)

★ GOAL:-

The aim of MDP is to train an agent to find a policy that will return the maximum cumulative rewards from taking a series of actions in one or more states

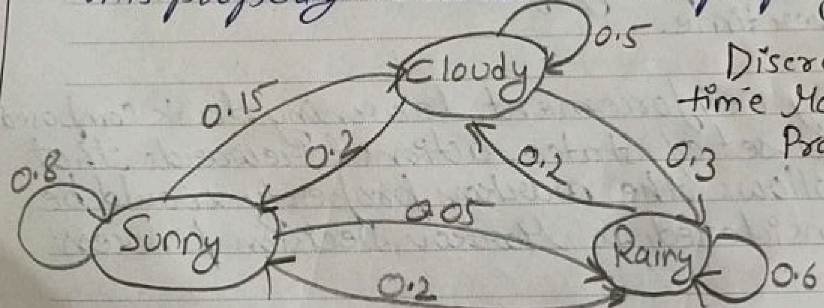
$$\pi(s) : S \rightarrow A$$

★ Markov Process:-

If given entire history of states $(S_0, S_1, \ldots S_t)$ the future state (S_{t+1}) depends only on present state (S_t)

$$P[S_{t+1} / S_0, S_1, \ldots S_t] = P[S_{t+1} / S_t]$$

This property is called Markov property



Discrete time Markov Process

Let state is weather on any given day

Tomorrow's weather depends on today's weather

One step transition probablity matrix M

$$\begin{array}{c|ccc|c} S_t & & S_{t+1} & & \\ & \text{Sunny} & \text{Cloudy} & \text{Rainy} & \\ \hline \text{Sunny} & 0.8 & 0.15 & 0.05 & \rightarrow 1 \\ \text{Cloudy} & 0.2 & 0.5 & 0.3 & \rightarrow 1 \\ \text{Raing} & 0.2 & 0.2 & 0.6 & \rightarrow 1 \end{array}$$

→ At each time step, the process is in some state S_t

→ The agent may choose any action a_t available in state S_t

→ The process responds at the next time step by moving into state S_{t+1} & giving the agent a corresponding reward $r_{a_t}(S_t, S_{t+1})$

→ The state S_t captures all relevant information from the history $(S_0, S_1, \ldots S_{t-1})$. Once the state S_t is known the history may be thrown away.

→ As MDP satisfies Markov property the probablity of transitioning from state S_t to S_{t+1} depends only on S_t & a_t

$$P[S_{t+1} / S_0, \ldots S_t, a_0, a_1 \ldots a_t]$$
$$= P[S_{t+1} / S_t, a_t]$$
$$= P[S_{t+1} / S_t]$$

★ State transition Matrix

For a Markov state s & successor state s', the state transition probability is defined by

$$P_{ss'} = P[S_{t+1} = s' / S_t = s]$$

State transition Matrix P defines transition probabilities from all states s to all successor states s', where each row of the matrix sums to 1

$$P = \begin{bmatrix} P_{11} & P_{12} & \ldots & P_{1n} \\ P_{21} & P_{22} & \ldots & P_{2n} \\ \vdots & & & \\ P_{n1} & P_{n2} & \ldots & P_{nn} \end{bmatrix}$$

★ Markov Decision Process -

→ That means an agent is supposed to decide the best action to select based on his current state & when this step is repeated, the problem is known as MDP

→ The MDP process is used to model an environment for an agent to learn

→ It is mathematical representation of the environment

★ Parts of MDP model - $\langle S, A, T, R, \gamma \rangle$

i) States (S) :- A state is used to represent the current situation of the environment from the agent's view.

S - set of possible states

ii) Action A(s) :- A → set of possible Actions
An action is how an agent gets from one state s to another state s'

iii) Transition function T(S, a, s') or P(s'/s, a)
The transition function is the probability that an agent gets to the next

SAGE UNIVERSITY | SIRT SAGAR GROUP OF INSTITUTIONS | SAGE SCHOOL | M/s Agrawal Construction Co. | Apollo SAGE HOSPITALS BHOPAL | AGRAWAL POWER PVT. LTD. | Sage Foundation

state (s') given that he takes an action (a) from his current State (S)

OR

This is the probability of getting to state s' given that we take action α in State s

iv) Rewards:-

This function rewards an agent for taking the right actions & punishes (with negative rewards) for wrong actions.

R(S, S') - the reward for moving from State S to s'
or R(S, a)

The purpose MDP is to find a policy $\pi(s)$
This policy specifies the action to be taken by the agent when in a state s

~~α-discount factor~~

v) γ - discount factor

Help us to evaluate the expected reward relative to the advantage or disadvantage of each state. For ex→ Given the state I am in assuming I take the best possible action now & at each subsequent step what long term reward can I expect

Bellman Equations:-

Firstly we know that goal of RL is given the current state we are in, choose the optimal action which will maximize the
⇒ long-term expected reward provided by the environment

⇒ Bellman equation helps to solve MDP or we can say it helps in finding optimal policy & Value function.

i) Value function

→ Each state is associated with a value function V(s) predicting the expected amount of future rewards we are able to receive in this state by acting the corresponding policy.

→ It is equal to expected total reward for an agent starting from state s

$$V(S) = E[R / S_t = s]$$

SAGE UNIVERSITY | SIRT SAGAR GROUP OF INSTITUTIONS | SAGE SCHOOL | M/s Agrawal Construction Co. | Apollo SAGE HOSPITALS BHOPAL | AGRAWAL POWER PVT. LTD. | The Sage Foundation

or $V(s) = E[R_{t+1} + \gamma V(S_{t+1}) / S_t = s]$

or $V_\pi(s) = E[R / S_t = s]$

π - Policy , γ - discount factor

OR $V_\pi(s) = E\left[\sum_{t=0} \gamma^t r_{t+1} / S_0 = s\right]$

→ The value function estimates value of a state

→ It depends on the policy by which the agent picks action to perform

→ The state with high V is more valueable than the state with low V

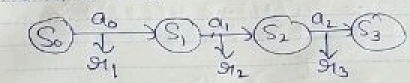
γ → discount factor – a factor that is multiplied to future reward to get its present value

Usually $0 \le \gamma \le 1$

present reward = γ^k * future reward after k steps

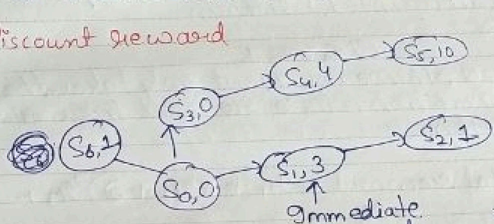
Ex → 10 points after 2 timesteps ⇒
Present reward $= (0.5)^2 * 10$

At time $t=0$



Present value of total reward at $t=0$ for
$S_0 = r_1 + \gamma r_2 + \gamma^2 r_3 + \gamma^3 r_4 + \ldots$

Discount reward



$\gamma = 0.5$

$S_0 \to S_1 \to S_2$ total reward $= 3 + 1 = 4$

$S_0 \to S_3 \to S_4 \to S_5$ ⇒ total reward $= 0 + 4 + 10 = 14$

(Discount reward) Cumulative reward of state

$S_0 = 0 + 0.5 \times 4 + 0.5^2 \times 10$

So, the discounted return at timestep t is given by

SAGE UNIVERSITY | SIRT SAGAR GROUP OF INSTITUTIONS | SAGE SCHOOL | M/s Agrawal Construction Co. | Apollo SAGE HOSPITALS BHOPAL | AGRAWAL POWER PVT. LTD. | Sage Foundation

total discount reward

$= r(t+1) + \gamma r(t+2) + \gamma^2 r(t+3) + \dots$

$= \sum_{i=0}^{\infty} \gamma^i r_{i+1}$

Goal of agent is to maximize expected (discounted) return

$0 \leq \gamma \leq 1$

$\gamma = 0$, it means agent only takes the most immediate reward into consideration

Total reward $= r_1 + \gamma r_2 + \gamma^2 r_3 + \dots$
$= r_1$

If $\gamma = 1$, the return is without discount

Total reward $= r_1 + \gamma r_2 + \gamma^2 r_3$
$= r_1 + r_2 + r_3$

For large value of γ, the agent take care of more distant future to take more rewards.

Value function $V_\pi(s) = E\left[\sum_{i=0}^{\infty} \gamma^i r_{i+1}\right]$
$\forall s \in S$

Markov decision process objective is to find optimal policy where all states have max value function

$$\boxed{V^*(s) = \max V_\pi(s)}$$

Bellman Equation

→ It helps to solve Markov decision process or we can say it helps in finding optimal policy & value function

For Deterministic Environment

Bellman equation is given by

π-policy $V_\pi(s) = \max_a \{R(s,a) + \gamma V(s')\}$

Bellman equation answer the following question

Given the state I am in, assuming I take the best possible action now & at each ~~stat~~ subsequent step, what long term reward can I expect?

OR

What is the value of the state?

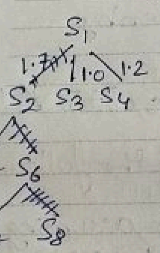
Where

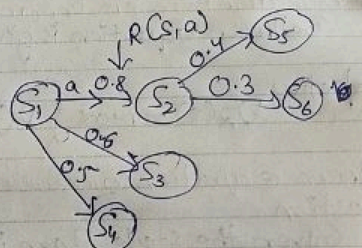
$V(s)$ = Value for being in a certain state s

$V(s')$ - It is the value of being in the next state after taking an action a

$\gamma \rightarrow$ discount factor

$\max_a \rightarrow$ Here we want maximum of action (which maximizes the value) a so that agent is in the optimal state





$$V(s_1) = \max \begin{cases} 0.8 + \gamma V(s_2) \\ 0.6 + \gamma V(s_3) \\ 0.5 + \gamma V(s_4) \end{cases}$$

$$V(s_2) = \max \begin{cases} 0.4 + \gamma V(s_5) \\ 0.3 + \gamma V(s_6) \end{cases}$$

Bellman equation refers to a set of equations that decompose the value function into immediate reward plus the discounted future values

For Non deterministic / Stochastic environment

Bellman equations

$$V(s) = \max_a \left\{ R(s,a) + \gamma \sum_{s'} P(s,a,s') V(s') \right\}$$