CSE445.2 – SP21 – Final Assignment

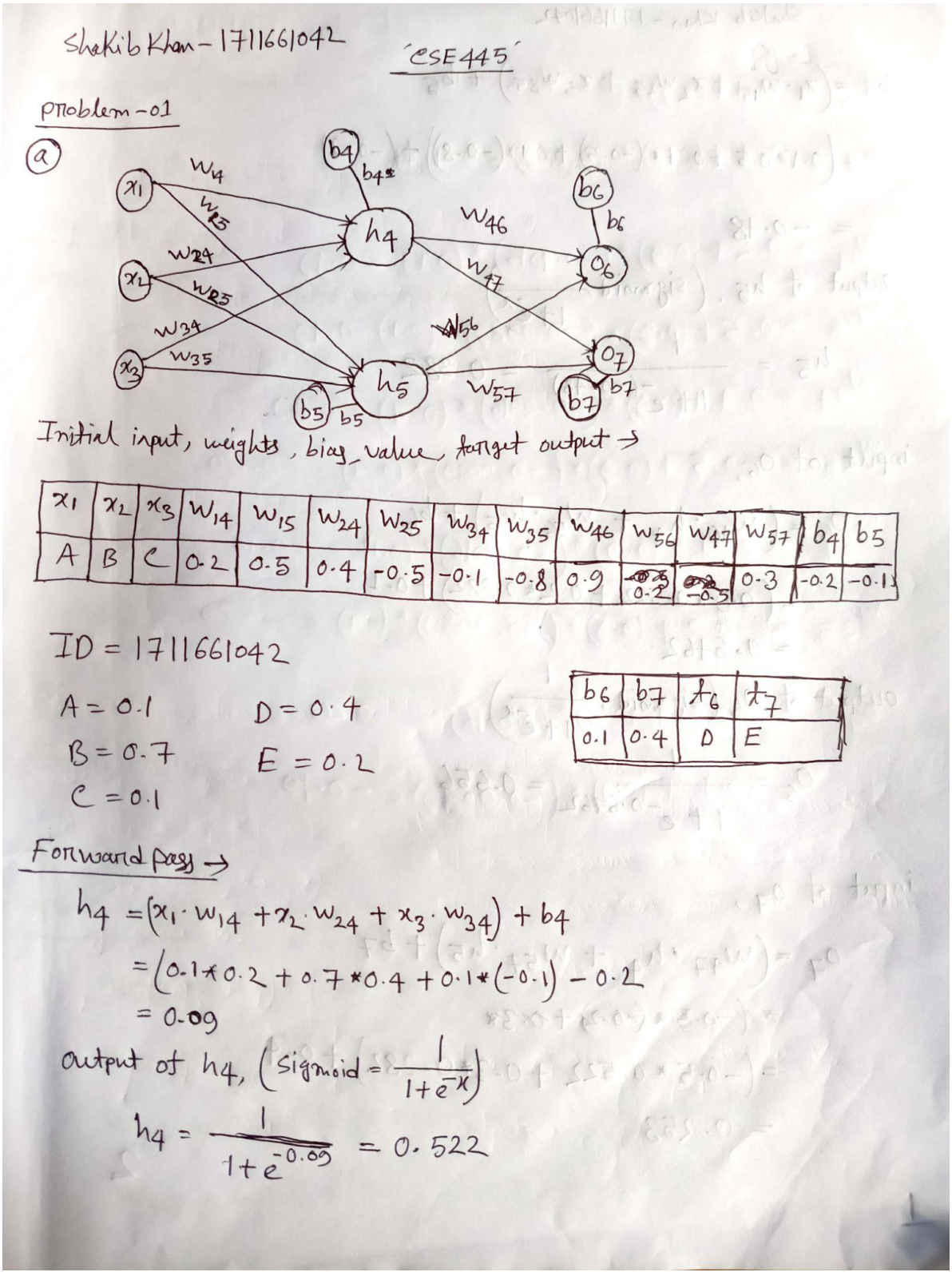
May 23th, 2021 (Sunday)

—

Shakib Khan
1711661042
Submitted to: **SAA3**

Solved Question No.01

For this problem, I used a learning rate of **0.9**



Shakib Khan – 1711661042 "CSE445"

problem – 01

(a)

Initial input, weights, bias_value, target output →

x_1	x_2	x_3	w_{14}	w_{15}	w_{24}	w_{25}	w_{34}	w_{35}	w_{46}	w_{56}	w_{47}	w_{57}	b_4	b_5
A	B	C	0.2	0.5	0.4	-0.5	-0.1	-0.8	0.9	0.2	-0.5	0.3	-0.2	-0.1

ID = 1711661042

A = 0.1 D = 0.4

B = 0.7 E = 0.2

C = 0.1

b_6	b_7	t_6	t_7
0.1	0.4	D	E

Forward pass →

$$h_4 = (x_1 \cdot w_{14} + x_2 \cdot w_{24} + x_3 \cdot w_{34}) + b_4$$

$$= (0.1 * 0.2 + 0.7 * 0.4 + 0.1 * (-0.1) - 0.2$$

$$= 0.09$$

output of h_4, $\left(\text{sigmoid} = \frac{1}{1+e^{-x}}\right)$

$$h_4 = \frac{1}{1+e^{-0.09}} = 0.522$$

Shakib Khan - 1711661042

$$h_5 = (x_1 \cdot w_{15} + x_2 w_{25} + x_3 w_{35}) + b_5$$

$$= (0.1 * 0.5 + 0.7 * (-0.5) + 0.1 * (-0.8)) + (-0.1)$$

$$= -0.48$$

output of h_5, $\left(\text{sigmoid } \frac{1}{1+e^{-x}}\right)$

$$h_5 = \frac{1}{1+e^{-(-0.48)}} = 0.382$$

input of O_6,

$$O_6 = (w_{46} \cdot h_4 + w_{56} \cdot h_5) + b_6$$

$$= (0.9 * 0.522 + 0.2 * 0.382) + 0.1$$

$$= 0.6462$$

output of O_6 $\left(\text{sigmoid } \frac{1}{1+e^{-x}}\right)$

$$O_6 = \frac{1}{1+e^{-0.6462}} = 0.656$$

input of O_7,

$$O_7 = (w_{47} \cdot h_4 + w_{57} \cdot h_5) + b_7$$

$$= (-0.5 * 0.522 + 0.3 * 0.382) + 0.4$$

$$= 0.253$$

Shakib Khan 1711661042

Output O_7 $\left(\text{sigmoid} = \frac{1}{1+e^{-x}}\right)$ $\Rightarrow$ squared loss, $= \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$

$$O_7 = \frac{1}{1+e^{-(0.253)}} = 0.562$$

Backpass,

Error at O_6, ~~$O_6 = (0.4 - 0.656) * (0.656)$~~ $t_6 = 0.4$

$$\frac{\partial E}{\partial w_{j,k}} \Rightarrow O_6 = -(0.4 - 0.656) * (0.656) * (1 - 0.656)$$

$$= 0.057$$

Error at O_7, $O_7 = (0.2 - 0.562) * (0.562) * (1 - 0.562)$ $t_7 = 0.2$

$$= 0.089$$

Error at h_5, $h_5 = -((0.057 * 0.2 + 0.089 * 0.3) * (0.382)(1 - 0.382)$

$$= -0.0089$$

Error at h_4, $h_4 = -((0.057 * 0.9 + 0.089 * (-0.5)) * (0.522) * (1 - 0.522)$

$$= -0.0016$$

update weights $\rightarrow w_{jk} = w_{jk} + \eta \cdot Err_k O_j$ (learning rate $\eta = 0.9$):

$$w_{57} = 0.3 + (0.9) * (0.089)(0.382) = 0.33$$

$$w_{56} = 0.2 + (0.9) * (0.057)(0.382) = 0.21$$

$$w_{46} = 0.9 + (0.9) * (0.057)(0.522) = 0.92$$

$$w_{47} = -0.5 + (0.9) * (0.089)(0.522) = -0.45$$

Shakib Khan 171166142

$b_6 = 0.1 + (0.9) * (0.057) = 0.15$

$b_7 = 0.4 + (0.9) * (0.089) = 0.48$

$w_{14} = 0.2 + (0.9) * (-0.0016) * (0.1) = 0.19$

$w_{15} = 0.5 + (0.9) * (-0.0089) * (0.1) = 0.49$

$w_{24} = 0.4 + (0.9) * (-0.0016) * (0.7) = 0.39$

$w_{25} = -0.5 + (0.9) * (-0.0089) * (0.7) = -0.50$

$w_{34} = -0.1 + (0.9) * (-0.0016) * (0.1) = -0.10$

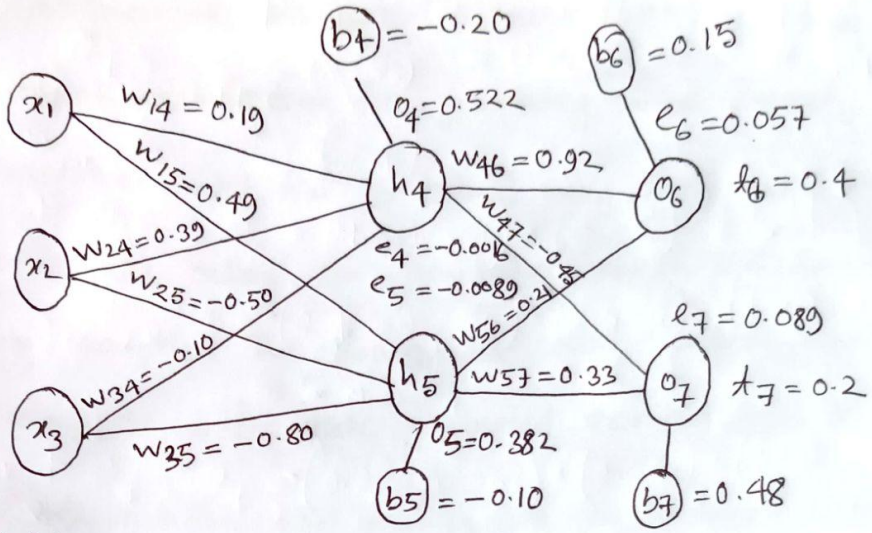
$w_{35} = -0.8 + (0.9) * (-0.0089) * (0.1) = -0.80$

$b_4 = -0.2 + (0.9) * (0.0016) = -0.20$

$b_5 = -0.1 + (0.9) * (-0.0089) = -0.10$

Shakib Khan 1711661042

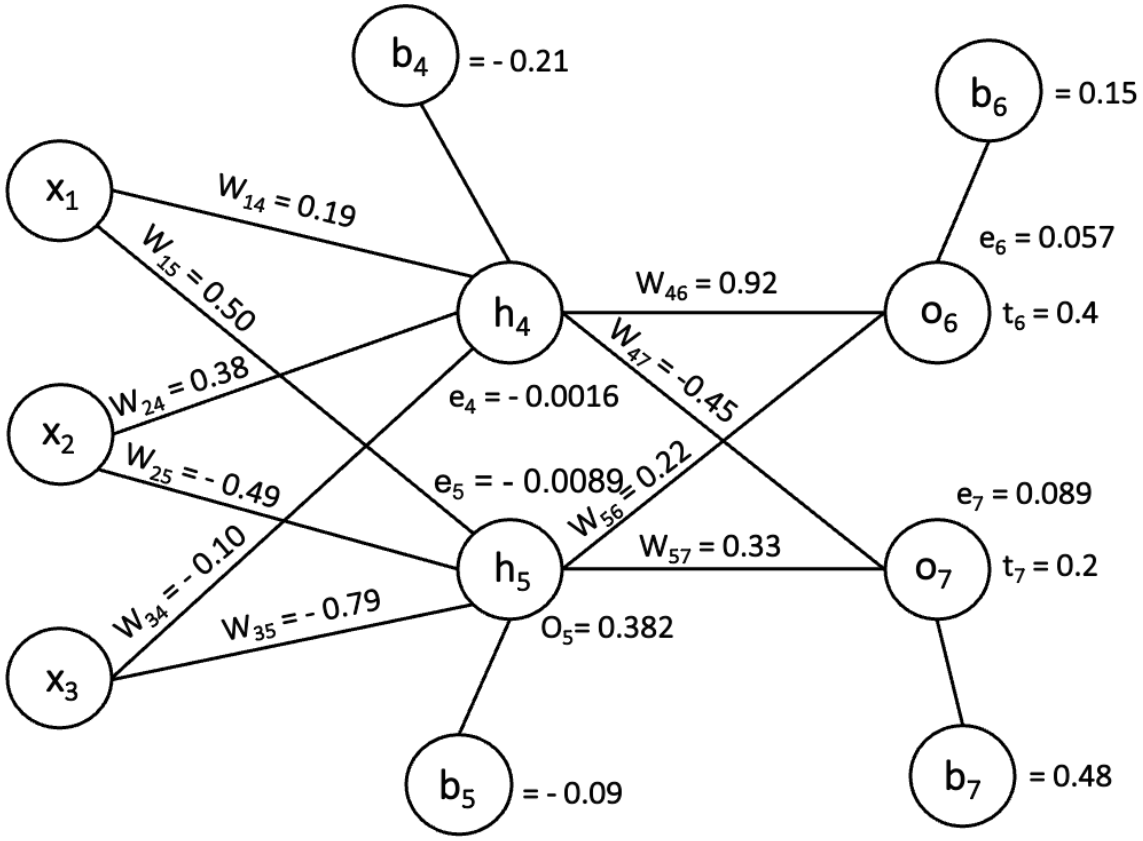
The update network,



update table,

w_{14}	w_{15}	w_{24}	w_{25}	w_{34}	w_{35}	w_{46}	w_{47}	w_{56}	w_{57}
0.19	0.49	0.39	−0.50	−0.10	−080	0.92	−0.45	0.21	0.33

b_4	b_5	b_6	b_7
−0.20	−0.10	0.15	0.48



The updated Neural Network

Screen Shot of my Code

```
'''
My ID is : 1711661042

First three digit ID is : 171
Last two digit ID is : 42

A = x1 = 1/10 = 0.1
B = x2 = 7/10 = 0.7
C = x3 = 1/10 = 0.1

D = 4/10 = 0.4
E = 2/10 = 0.2

Initital input, weight, bias values, and target outputs

x1 x2 x3 w14 w15 w24 w25 w34 w35 w46 w47 w56 w57 b4 b5 b6 b7 t6 t7
A B C 0.2 0.5 0.4 -0.5 -0.1 -0.8 0.9 -0.5 0.2 0.3 -0.2 -0.1 0.1 0.4 D E

Use the backpropagation algorithm to train this network for a single iteration, using a sigmoid
activation function and a squared error loss function.'''

import numpy as np
#weights of hidden layer
x1 = 0.1
x2 = 0.7
x3 = 0.1

w14 = 0.2
w15 = 0.5
w24 = 0.4
w25 = -0.5
w34 = -0.1
w35 = -0.8
b4 = - 0.2
b5 = - 0.1

def sigmoid (x):
    return 1/(1 + np.exp(-x))

#Forward pass
# h4
input_in_h4 = (x1*w14 + x2*w24 +x3*w34 ) + b4
output_of_h4 = sigmoid(input_in_h4)

# h5
input_in_h5 = (x1*w15 + x2*w25 +x3*w35 ) + b5
output_of_h5 = sigmoid(input_in_h5)
```

```
# weights of output layer
w46 = 0.9
w56 = 0.2
w47 = -0.5
w57 = 0.3
b6 = 0.1
b7 = 0.4

# O6
input_in_O6 = (output_of_h4*w46 + output_of_h4*w56 ) + b6
output_of_O6 = sigmoid(input_in_O6)

#O7
input_in_O7 = (output_of_h5*w47 + output_of_h5*w57 ) + b7
output_of_O7 = sigmoid(input_in_O7)

# Error = squared error loss
t6 = 0.4
t7 = 0.2

#Backward Pass
# error O6
error_at_O6 = -(t6 - output_of_O6)*output_of_O6*(1-output_of_O6)

# error O7
error_at_O7 = -(t7 - output_of_O7)*output_of_O7*(1-output_of_O7)

# Error h4
error_at_h4 = -(error_at_O6*w46 + error_at_O7*w56)*output_of_h4*(1-output_of_h4)

# Error h5
error_at_h5 = -(error_at_O6*w47 + error_at_O7*w57)*output_of_h5*(1-output_of_h5)


# Update the weights
learning_rate = 0.9

up_w57 = w57 + learning_rate*error_at_O7*output_of_h5
up_w56 = w56 + learning_rate*error_at_O6*output_of_h5
up_w46 = w46 + learning_rate*error_at_O6*output_of_h4
up_w47 = w47 + learning_rate*error_at_O7*output_of_h4
up_b6 = b6 + learning_rate*error_at_O6
up_b7 = b7 + learning_rate*error_at_O7
up_w14 = w14 + learning_rate*error_at_h4*x1
up_w15 = w15 + learning_rate*error_at_h5*x1
up_w24 = w24 + learning_rate*error_at_h4*x2
up_w25 = w25 + learning_rate*error_at_h5*x2
up_w34 = w34 + learning_rate*error_at_h4*x3
up_w35 = w35 + learning_rate*error_at_h5*x3
up_b4 = b4 + learning_rate*error_at_h4
up_b5 = b5 + learning_rate*error_at_h5
```

```
print('Updated Weights are:\n')
print(f'w14:',up_w14)
print(f'\nw15:',up_w15)
print(f'\nw24:',up_w24)
print(f'\nw25:',up_w25)
print(f'\nw34:',up_w34)
print(f'\nw35:',up_w35)
print(f'\nb4:',up_b4)
print(f'\nb5:',up_b5)
print(f'\nw46:',up_w46)
print(f'\nw47:',up_w47)
print(f'\nw56:',up_w56)
print(f'\nw57:',up_w57)
print(f'\nb6:',up_b6)
print(f'\nb7:',up_b7)
```

```
Updated Weights are:

w14: 0.1983978204714846

w15: 0.5000333705593455

w24: 0.3887847433003922

w25: -0.4997664060845816

w34: -0.10160217952851541

w35: -0.7999666294406546

b4: -0.21602179528515406

b5: -0.09966629440654518

w46: 0.9276037950392275

w47: -0.45645491935172877

w56: 0.22019505412886087

w57: 0.3318577666399764

b6: 0.15283176415741276

b7: 0.483342287817877
```

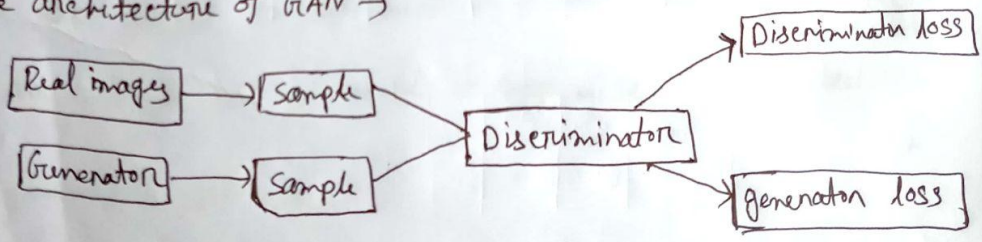
Shakib Khan 1711661042

Problem ➢ 01

(b) A generative adversarial network (GAN) is a type of neural network architecture for generating model. Basically, a gan is generative model that is trained using two neural network models. One model called the 'generator' model that learns to generate new sample. The other model called 'discriminator' and it learns to differentiate generated examples from real examples.

A dataset is the initial training data for the discriminator. Train it with samples from the training dataset, until it achieves a good accuracy. The generator trains based on weather it succeeds in fooling the discriminator.

The architecture of GAN →



GAN's uses in → generate Datasets, realistic photographs, image to image translation, Text to image translation, Fake video's, Speech of famous person, 3D object.

Solved Question No.02

Shakib Khan 1711661042

Problem-02

@ Markov decision process (MDP) is a stochastic control process. It provides a model decision making in situation where outcomes are partly random and partly under the control of a decision making. MDP problems assume a finite number of states and actions. At each time the agent observes a state and executes an action, which costs to be minimized. The cost and the successor state depend only on the current state and the choosen action.
The MDP also define by, a set of states, a set of actions, a transition function, a reward function, a start state, a terminal state.
MDPs are useful to optimization problem. Using MDP we can get an optimal policy where it maximize the expected result.

The equation it state is,

$$P(S_{t+1} = s' \mid S_t = s_t, A_t = a_t)$$

Shakib Khan 1711661042

(b) Let's state from the diagram →

State → The states are (cool, warm, Overheated)

actions → The actions are (slow, fast)

transition probability → ~~((cool, cool) = 1.0~~ ,

T (cool, slow) = 1.0

T (warm, slow) = 0.5

T (cool, fast) = 0.5

T (warm, ~~fast~~ slow) = 0.5

T (warm, fast) = 1.0

T (cool, fast) = 0.5

Reward = (cool → cool) = ~~+0~~ +1 (slow)

(warm → cool) = +1 (slow)

~~(warm → cool)~~ = (warm → warm) = +1 (slow)

(warm → overheated) = −10 (fast)

(cool → warm) = +2 (fast)

(cool → cool) = +2 (fast)

Terminal States = The overheated state is terminated state because there is no actions.

Shakib Khan 1711661042

problem-2

(c) ① $\gamma = 1.0$, steps $K = 3$

The equation,

$$V_{K+1} \leftarrow \max_a \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma V_K(s') \right]$$

At timestep,

		cool	warm	overheated
at $\Rightarrow K=0$	V_0	0	0	0
at $K=1$	V_1	2	1	0
at $K=2$	V_2	3.5	2.5	0
at $K=3$	V_3	5	4	0

calculate V_1 (cool)

V_1 (cool, slow) $= 1.0(1 + 1*0) = 1$

V_1 (cool, fast) $= 0.5(2+1*0) + 0.5(2+1*0) = 2 \rightarrow$ max

Calculate V_1 (warm)

V_1 (warm, slow) $= 0.5(1+1*0) + 0.5(1+1*0) = 1 \rightarrow$ max

V_1 (warm, fast) $= 1(-10 + 1*0) = -10$

calculate V_2 (cool)

V_2 (cool, slow) $= 1(1 + 1*2) = 3$

V_2 (cool, fast) $= 0.5(2+1*2) + 0.5(2+1*1) = 3.5 \rightarrow$ max

calculate V_1 (overheated)

V_1 (overheated) $= 0$

Shakib Khan 1711661042

calculate V_2(warm)

$V_2(\text{warm, slow}) = 0.5(1+1*2) + 0.5(1+1*1) = 2.5 \rightarrow \max$

$V_2(\text{warm, fast}) = 1(-10+1*1) = -9$

calculate V_2(overheated) $= 0$

calculate V_3(cool)

$V_3(\text{cool, slow}) = 1(1+1*3.5) = 4.5$

$V_3(\text{cool, fast}) = 0.5(2+1*3.5) + 0.5(2+1*2.5) = 5 \rightarrow \max$

calculate V_3(~~fast~~)(warm)

$V_3(\text{warm, slow}) = 0.5(1+1*3.5) + 0.5(1+1*2.5) = 4$

$V_3(\text{warm, fast}) = 1(-10+1*2.5) = -7.5$

calculate V_3(overheated) $= 0$

Shakib Khan 1711661042

problem-2

(c) (ii) given $\gamma = 0.9$

The equation, $V_{K+1} \leftarrow \max_a \sum_{s'} T(s,a,s')\left[R(s,a,s') + \gamma V_K(s')\right]$

Time step,

	cool	warm	overheated
V_0, K=0	0	0	0
V_1, K=1	2	1	0
V_2, K=2	3.35	2.35	0
V_3, K=3	4.6225	3.565	0

calculate, $V_1(cool)$

$V_1(cool, slow) = 1.0(1 + 0.9*0) = 1$

$V_1(cool, fast) = 0.5(2 + 0.9*0) + 0.5(2 + 0.9*0) = 2 \rightarrow max$

calculate $V_2(warm)$

$V_2(warm, slow) = 0.5(1 + 0.9*0) + 0.5(1 + 0.9*0) = 1 \rightarrow max$

$V_2(warm, fast) = 1(-10 + 0.9*0) = -10$

calculate $V_2(overheated) = 0$

Shakib Khan 1711661042

calculate, v_2(cool)

v_2(cool, slow) = 1(1+0.9*2) = 2.8

v_2(cool, fast) = {0.5(2+0.9*2) + 0.5(2+0.9*1) = 3.35 → max

calculated, v_2(warm)

v_2(warm, slow) = 0.5(1+0.9*2) + 0.5(1+0.9*1) = 2.35 → max

v_2(warm, fast) = 1 (-10 + 0.9*1) = -9.1

calculated v_2(overheated) = 0

calculated, v_3(cool)

v_3(cool, slow) = 1 (1+0.9*3.35) = 4.015

v_3(cool, fast) = 0.5(2+0.9*3.35) + 0.5(2+0.9*2.35) = 4.6225 → max

calculated, v_3(warm)

v_3(warm, slow) = 0.5(1+0.9*3.35) + 0.5(1+0.9*2.35) = 3.565 → max

v_3(warm, fast) = 1(-10 + 0.9*2.35) = -7.885

calculated v_3(overheated) = 0

12

problem-02

(d) In value iteration, you can strat with a random value function and then find a new value function is an iterative process, utill reaching the optimal value function. This process based on the bellman equation.

where, in policy iteration, you start with a random policy, then find the value function of that policy, then find a new improved policy based on the previous value function. in this process the policy is guaranteed to be strict improvment over the privious one

we have problems with value iteration like

- it is slow per iteration
- The max state rarely changes

To tackle this problem, we used the policy iteration,

- Fixed ~~acro~~ a policy until it convarges.
- it is pretty fast to compute
- update policy using one-step look-ahead

So, that's why policy iteration convarge soonen than value iteration.

Solved Question No.03

Shakib Khan 1711661042

Problem-03

(a) Regularization is a technique which is used to solve the overfitting problem of the machine learning models.

The 'regularization' term refers to normalizes and moderates weights attached to a feature so that algorithm do not rely on just a few ~~weights~~ features to predict the result. This technique helps to avoid the problem of overfitting.

There are two types of regularization

→ L1 Regularization or Lasso Regularization

→ L2 Regularization or Ridge Regularization.

The Lasso regularization adds a penalty to the error function. The penalty is the sum of the absolute value of weights

$$\sum_{i=1}^{n} \left(y_i - \hat{y}_i\right)^2 + \rho \sum_{i=1}^{n} |w_i|$$ when ρ is tuning parameter

The Ridge regularization adds a penalty to the error function. The penalty is the sum of the squared value of weights

$$\sum_{i=1}^{n} \left(y_i - \hat{y}_i\right)^2 + \rho \sum_{i=1}^{n} (w_i)^2$$ when, ρ is tuning parameter.

Shakib Khan 1711661042

Problem - 3

(b) In machine learning, Bias is the difference between the average prediction of our model and the correct value which we are trying to predict. So, high biased model cause underfitting because it pays very little attention to the training data. On the other hand, variance is the variability of model prediction for a given data point or a value which tell us the spread of our data. So, a high variance model cause overfitting.

Bias and variance tradeoff -

if our model is too simple and has very few parameters then it may have high bias and low variance. Besides, if our model has a large number of parameters then it's going to have high variance and low bias. So, we need to find the right balance without overfitting and underfitting data.

Shakib Khan 1711661042

Problem-03

(e) Support vectors are data points that are closer to the hyperplane and impact the position and orientation of the hyperplane. Using these support vectors, we can maximize the margin of our classifier. By removing the support vectors will change the position of the hyperplane. These are the points that help us to build our SVM.

The SVM's job is to find a hyperplane is an N-dimensional space where N is the number of features that distinctly classifies the data points.

Shakib Khan 1711661042

problem-03

(d) The difference between Stochastic Gradient Descent and Batch gradient descent.

Stochastic Gradient Descent (SGD)	Batch Gradient Descent (BGD)
→ it picks up random instance of training data at each step and the computes.	→ it involves calculation over the full training set.
→ It is much faster.	→ It is very slow on very large dataset.
→ It is less computationally expensive	→ It is become very computationally expensive
→ This gives a quite good result on traing data	→ The result some time biased
→ Reaches the convergance much faster	→ Convergence is slow.

Shakib Khan 1711661042

Problem-03

(e) Ensemble methods are techniques that create multiple models and then combine them to produce improved results. This method usually produces more accurate solutions than a single model would. There are lots of techniques to use this algorithm.

- Max voting method is generally used for classification problems. in max voting, multiple models are used to make predictions. The predictions by each model are considered as vote. The prediction which we get from the majority of the models are used as the final prediction.
- Stacking technique that use prediction from multiple models like decision tree, Knn, svm to build a new model.
- The idea of Bagging is combining the results of multiple models to get a generalized result.
- Random Forest is an extension of the bagging method. it search for the best feature among a random subset
- AdaBoost or Adaptive Boost is the simplest boosting algorithms. AdaBoost assigns weights to the observation which are incorrectly predicted and the subsequent model to predict value correctly.

18

END
Thank You :)