1 Consider an undiscounted MDP having three states, (1, 2, 3), with rewards −1, −2, 0, respectively. State 3 is a terminal state. In states 1 and 2 there are two possible actions: a and b. The transition model is as follows:
- In state 1, action a moves the agent to state 2 with probability 0.6 and makes the agent stay in state 1 with probability 0.4.
- In state 2, action a moves the agent to state 1 with probability 0.6 and makes the agent stay in state 2 with probability 0.4
- In either state 1 or state 2, action b moves the agent to state 3 with probability 0.2 and makes the agent stay put with probability 0.8.
Answer the following questions:

1.1 What can be determined *qualitatively* about the optimal policy in states 1 and 2?

Ans - Intuitively the agent wants to get to State 3 as soon as possible, because it will pay a cost for each time step it spends in states 1 and 2. However, the only action that reaches state 3 (action b) succeeds with low probability, so the agent should minimize the cost it incurs while trying to reach the terminal state. This suggests that the agent should definitely try action b in state 1; in state 2, it might be better to try action a to get to state 1 (which is the better place to wait for admission to state 3), rather than aiming directly for state 3. The decision in state 2 involves a numerical tradeoff.

1.2 Apply policy iteration, showing each step in full, to determine the optimal policy and the values of states 1 and 2. Assume that the initial policy has action b in both states.

(1.2)

- **Initialization :-** $U \leftarrow \langle -1, -2, 0 \rangle$, $P \leftarrow \langle b, b \rangle$
- **Value determination :-**

$$U_1 = -1 + 0.2U_3 + 0.8U_1$$
$$U_2 = -2 + 0.2U_3 + 0.8U_2$$
$$U_3 = 0$$

Solving, we can get, $U_1 = -5$, $U_2 = -10$

- **Policy Update :-**

In state 1, $\sum_j T(1,a,j)U_j = 0.6 \times (-10) + 0.4 \times (-5) = -8$

$\sum_j T(1,b,j)U_j = 0.2 \times (0) + 0.8 \times (-5) = -4$

So action b is preferred for state 1.

In state, 2, $\sum_j T(1,a,j)U_j = 0.6 \times (-5) + 0.4 \times (-10) = -7$

$\sum_j T(1,b,j)U_j = 0.2 \times (0) + 0.8 \times (-10) = -8$

So in state 2, action a is preferred. It changed initial policy. So, proceed.

- **Value determination :-**

$$U_1 = -1 + 0.2U_3 + 0.8U_1$$
$$U_2 = -2 + 0.6U_1 + 0.4U_2$$
$$U_3 = 0$$

Solving,

$U_1 = -5$

$U_2 = -8.33$

- **Policy Update :-**

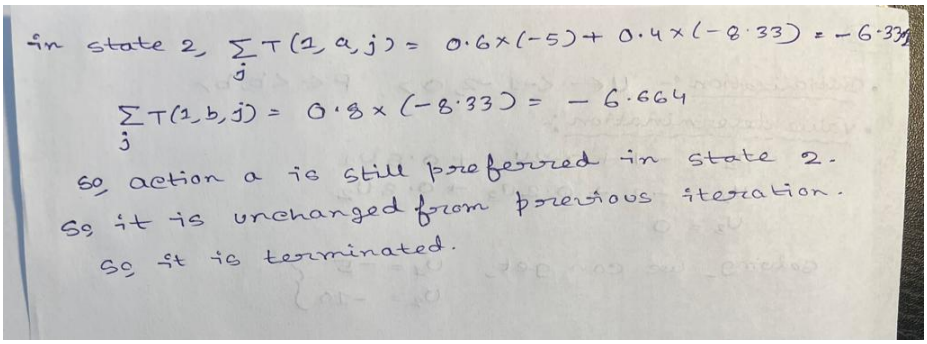
In state 1, $\sum_j T(1,a,j)U_j = 0.6 \times (-8.33) + 0.4 \times (-5)$

$= -7$

$\sum_j T(1,b,j)U_j = 0.8 \times (-5) = -4$

So action b is still preferred in state 1

Continued...



1.3 What happens to policy iteration if the initial policy has action a in both states? Does discounting help? Does the optimal policy depend on the discount factor?

Ans - An initial policy with action a in both states leads to an unsolvable problem. The initial value determination problem has the form:

$u_1 = -1 + 0.4\ u_1 + 0.6\ u_2$

$u_2 = -2 + 0.6\ u_1 + 0.4\ u_2$

and the first two equations are inconsistent. If we were to try to solve them iteratively, we would find the values tending to $-\infty$.

Discounting leads to well-defined solutions by bounding the penalty (expected discounted cost) an agent can incur at either state. However, the choice of discount factor will affect the policy that results. For γ small, the cost incurred in the distant future plays a negligible role in the value computation, because γ n is near 0. As a result, an agent could choose action b in state 2 because the discounted short-term cost of remaining in the non-terminal states (states 1 and 2) outweighs the discounted long-term cost of action b failing repeatedly and leaving the agent in state 2.

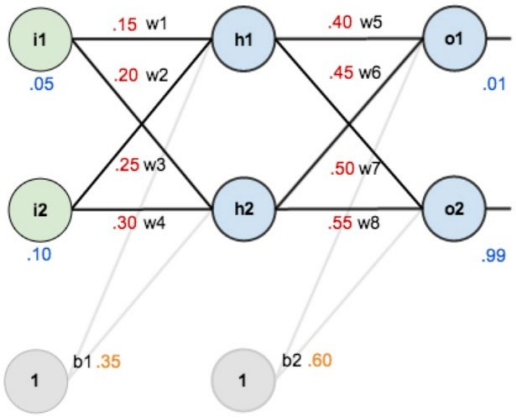
2. Compare and contrast the various CNN-based object detectors using the following metrics: number of parameters, training and test times, performance on ImageNet dataset.

Detector	Performance(mAP)
Faster RCNN	39.8
SSD	43.4 (Ilsvrc det 2014)
RCNN	31.4 (Ilsvrc det 2013)
Overfeat	24.3 (Ilsvrc det 2013)
SPPnet	31.84 (ilsvrc det 2014)

1. Discuss methods for dropout and searching for the best neural architecture.

   Dropout is a regularization method that approximates training a large number of neural networks with different architectures in parallel. During training, some number of layer outputs are randomly ignored or “dropped out”, with a probability of p. This has the effect of making the layer look-like and be treated-like a layer with a different number of nodes and connectivity to the prior layer. In effect, each update to a layer during training is performed with a different “view” of the configured layer. As mentioned in the original Dropout paper, dropping a unit out implies temporarily removing it from the network, along with all its incoming and outgoing connections. Dropout has the effect of making the training process noisy, forcing nodes within a layer to probabilistically take on more or less responsibility for the inputs, and therefore essentially searching for a better architecture. This conceptualization suggests that perhaps dropout breaks-up situations where network layers co-adapt to correct mistakes from prior layers, in turn making the model more robust. Dropout simulates a sparse activation from a given layer, which interestingly, in turn, encourages the network to actually learn a sparse representation as a side-effect. As such, it may be used as an alternative to activity regularization for encouraging sparse representations in autoencoder models.

2. For the neural network shown below with initial weights indicated in the figure, calculate the next values of w7 and w3 assuming a single input i1=0.05, i2=0.10 with corresponding outputso1=0.01 and o2=0.99. Hint: Follow the calculations done in class on 10/06/2021!



Forward pass:

H1 = sigmoid(h1) = sigmoid (w1*i1+w2*i2+b1) = sigmoid(0.15*0.05+0.2*0.1+0.35) = sigmoid(0.3775) = 0.5933

H2 = sigmoid(h2) = sigmoid (w3*i1+w4*i2+b1) = sigmoid(0.25*0.05+0.3*0.1+0.35) = sigmoid(0.3925) = 0.5969

O1 = sigmoid(o1) = sigmoid(w5*h1+w6*h2+b2) = sigmoid(0.4*0.5933 + 0.45*0.5969+0.6) =

sigmoid(1.1059)= 0.7514

O2 = sigmoid(o2) =sigmoid(w7*h1+w8*h2+b2) = sigmoid(0.5*0.5933 + 0.55*0.5969+0.6) = sigmoid(1.2249)= 0.7729

Loss L = 0.5(0.7514-0.01)^2 + 0.5(0.7729-0.99)^2 = 0.5497 +0.0471 = 0.2984

$$\frac{\delta L}{\delta w7} = \frac{\delta L}{\delta O2} * \frac{\delta O2}{\delta o2} * \frac{\delta o2}{\delta w7} = -(0.99 - 0.7729) * \frac{e^{\{-1.2249\}}}{\left(1+e^{\{-1.2249\}}\right)^2} * 0.5933 = -0.2171 * 0.1755 * 0.5933 = -0.0226$$

w7_new = 0.5 - 0.5*(-0.0226) = **0.5113** (considering Learning rate = 0.5)

$$\frac{\delta L}{\delta w3} = \frac{\delta L}{\delta O1} * \frac{\delta O1}{\delta o1} * \frac{\delta o1}{\delta H2} * \frac{\delta H2}{\delta h2} * \frac{\delta h2}{\delta w3} + \frac{\delta L}{\delta O2} * \frac{\delta O2}{\delta o2} * \frac{\delta o2}{\delta H2} * \frac{\delta H2}{\delta h2} * \frac{\delta h2}{\delta w3} =$$
$$-(0.01 - 0.7514) * \frac{e^{\{-1.1059\}}}{(1 + e^{\{-1.1059\}})^2} * 0.45 * \frac{e^{\{-0.3925\}}}{(1 + e^{\{-0.3925\}})^2} * 0.05 + (0.7729 - 0.99)$$
$$* \frac{e^{\{-1.2249\}}}{(1 + e^{\{-1.2249\}})^2} * 0.55 * \frac{e^{\{-0.3925\}}}{(1 + e^{\{-0.3925\}})^2} * 0.05 = 0.0004977$$

w3_new = .25 - 0.5*(0.0004977) = **0.24975**