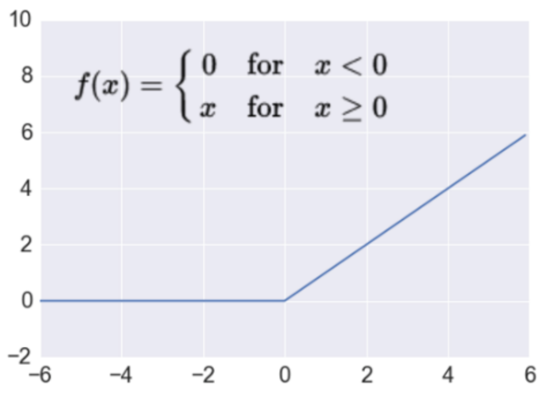
EBU4203 Introduction to AI – Week 2 Tutorial 2023

Q1: Draw a graphical representation of the Rectified Linear Unit (ReLU) activation function and provide a concise explanation of its key advantages in neural network activation functions.



ReLU is computationally efficient, making it a fast activation function
It can accelerate the convergence of gradient descent during training
The linear, non-saturating nature of ReLU helps mitigate the vanishing gradient problem.

(Slides Page 25)

Q2: Name two methods to prevent overfitting in training Deep Neural Networks (DNNs).

Any two of: Early Stopping, Dropout, Regularization.

(Slides Page 58 - 64)

Overfitting problem is that a model with high capacity fits the noise in the data instead of the underlying relationship, like shown in the right figure.

Methods to prevent overfitting:

1. Refer to the test results instead of consistent training. Once the test results become bad, the training should be stopped.
2. Regularization. A regularization term that penalizes large weights is added to the loss function, which avoid overfitting.
3. Dropout. The term “dropout” refers to dropping out the nodes (input and hidden layer) in a neural network (as seen in Figure). All the forward and backwards connections with a dropped node are temporarily removed, thus creating a new network architecture out of the parent network. The nodes are dropped by a dropout probability of p.
4. Early stopping. Stop when the validation accuracy (or loss) has not improved after *n* epochs.

Q3: In the context of reinforcement learning, define the concept of a State Value Function and elaborate on its role in aiding an agent's decision-making process

The state-value function is a function used to estimate the expected cumulative reward that an agent can obtain in the current environment state.

It represents the expected value of the future cumulative rewards that can be obtained by taking actions according to a specific policy (such as a particular decision-making rule) starting from the current state.

State-value functions help the agent evaluate the goodness of states to make better decisions. In mathematical notation, the state-value function is typically denoted as V(s), where s represents the state.

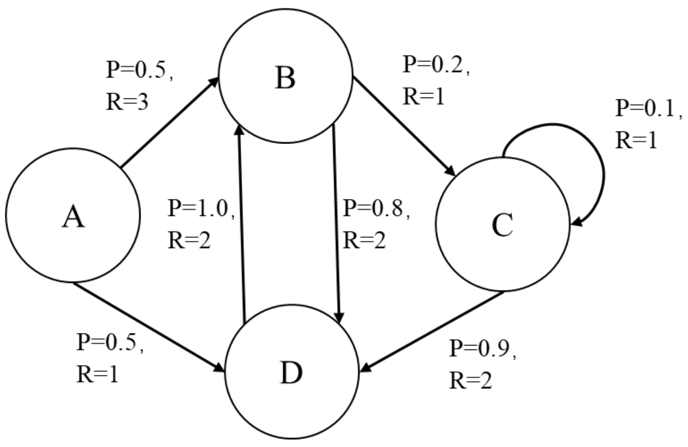
(Slides page 114 - 118)

The state-value function can be estimated by a variety of methods, such as using tables, function approximation (e.g., linear function approximation or neural networks). By constantly interacting with the environment, an agent can optimize its policy by updating the estimation of the state-value function.

The state value function plays an important role in reinforcement learning. By estimating the state-value function, an agent can evaluate the goodness of the current state and choose an action based on the estimation of the state-value function. Based on the state-value function, different policy selection methods, such as the ε-greedy, can be used to determine agent's choice of action.

Both value function methods (e.g., Q-learning and DQN) and policy gradient methods can use the state-value function as an auxiliary function to optimize the policy. The state-value function can provide information about the state to help the agent make decisions and learn.

Q4: Consider the following deterministic Markov Decision Process (MDP). The discount factor γ is 0.9. States are represented as A, B, C, and D. Arrows indicate state transition with corresponding actions. The action probability and immediate rewards are P and R next to the arrows, respectively. The MDP starts with an initial value function of $V_0(A)=V_0(B)=V_0(C)=V_0(D)=4$.



a) Complete the Markov Chain transition matrix for the given problem.

$$\mathbb{p} = \begin{bmatrix} 0 & 0.5 & 0 & 0.5 \\ 0 & 0 & 0.2 & 0.8 \\ 0 & 0 & 0.1 & 0.9 \\ 0 & 1.0 & 0 & 0 \end{bmatrix}$$

As shown in the figure, *P* next to the arrow indicates the probability that the agent chooses the action. Since this is a deterministic MDP, after selecting an action, it will deterministically transition to the corresponding state. Therefore, the state transition matrix can be written as a 4X4 matrix with a total of 4 states. For example, the element on the first row and first column represents the probability of transitioning from state A to state A; and we can know from the question, it is 0.

b) For **one iteration**, calculate the value function V$_1$(D).

Bellman equation: $V(s) = E\,[r_{t+1} + \gamma\, V(s_{t+1}) \mid S_t = s]$

$$V_1(D) = 1.0 * [2 + 0.9 * V_0(B)] = 5.6$$

c) For **one iteration**, calculate the value function V$_1$ of states A, B, and C.

Bellman equation: $V(s) = E\,[r_{t+1} + \gamma\, V(s_{t+1}) \mid S_t = s]$

$$V_1(C) = 0.1 * [1 + 0.9V_0(C)] + 0.9 * [2 + 0.9V_0(D)] = 5.5$$

$$V_1(B) = 0.2 * [1 + 0.9V_0(C)] + 0.8 * [2 + 0.9V_0(D)] = 5.4$$

$$V_1(A) = 0.5 * [3 + 0.9V_0(B)] + 0.5 * [1 + 0.9V_0(D)] = 5.6$$

According to Bellman equation, the state-value function can be calculated by the following recurrence relation:

$$V(s) = E\,[r_{t+1} + \gamma\, V(s_{t+1}) \mid S_t = s]$$

where *V(s)* denotes the value of state s, *r* denotes the immediate reward obtained after taking an action in state s, *γ* is the discount factor (used to measure the importance of future rewards).

The implication of Bellman equation is that the state-value function V(s) is equal to the expectation of the discounted value of the next state after the immediate reward R. By applying Bellman equation recursively, the value function can be computed for all states.

(Slides page 98 – 101, 106, 114 - 118)

Q5: Given the provided data pairs where (y_1, y_2, y_3) represent the true data points, and $(\hat{y}_1, \hat{y}_2, \hat{y}_3)$ represent the predicted data points:

- $y_1 = [0, 0, 1], \hat{y}_1 = [0.1, 0.2, 0.7]$
- $y_2 = [0, 1, 0], \hat{y}_2 = [0.1, 0.7, 0.2]$
- $y_3 = [1, 0, 0], \hat{y}_3 = [0.3, 0.4, 0.3]$

Answer the following questions:

a) Calculate the Mean Squared Error of this dataset

Solutions:

Sample 1 MSE loss = $\frac{[(0-0.1)^2 + (0-0.2)^2 + (0-0.7)^2]}{3} = 0.0467$

Sample 2 MSE loss = $\frac{[(0-0.1)^2 + (1-0.7)^2 + (0-0.2)^2]}{3} = 0.0467$

Sample 3 MSE loss = $\frac{[(1-0.3)^2 + (0-0.4)^2 + (0-0.3)^2]}{3} = 0.2467$

The overall MSE of this dataset is

$$\frac{0.0467 + 0.0467 + 0.2467}{3} = 0.1133$$

b) Calculate the Cross Entropy Loss

Solutions:

$$\begin{aligned}
\text{sample 1 loss} &= -(0 \times log0.1 + 0 \times log0.2 + 1 \times log0.7) = 0.35 \\
\text{sample 2 loss} &= -(0 \times log0.1 + 1 \times log0.7 + 0 \times log0.2) = 0.35 \\
\text{sample 3 loss} &= -(1 \times log0.3 + 0 \times log0.4 + 0 \times log0.4) = 1.20
\end{aligned}$$

$$L = \frac{0.35 + 0.35 + 1.2}{3} = 0.63$$

Cross-entropy loss, or log loss, measures the performance of a classification model whose output is a probability value between 0 and 1. Cross-entropy loss increases as the predicted probability diverges from the actual label.

For each sample, the Cross Entropy can be calculated by

$$\mathcal{L}(\theta) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K}\left[y_k^{(i)} \, log \, \hat{y}_k^{(i)} + \left(1 - y_k^{(i)}\right) \, log\left(1 - \hat{y}_k^{(i)}\right)\right]$$

(Slides page 34)

The Mean Squared Error measures how close a regression line is to a set of data points. It is a risk function corresponding to the expected value of the squared error loss. Mean square error is calculated by taking the average, specifically the mean, of errors squared from data as it relates to a function.

For each sample, the MSE can be calculated by

$$\mathcal{L}(\theta) = \frac{1}{n}\sum_{i=1}^{n}\left(y^{(i)} - \hat{y}^{(i)}\right)^2$$

(Slides page 35)

Q6. In Figure 1, given a 4x4 Gridworld with the agent starts at the initial place s_{11} and tries to find the route to reach the goal place s_{44} with the possible movement as North, South, East and West. In the given Gridworld, s_{xy} defines different positions that the agent will pass through, where:

- $s_{11}$is the starting state (S)
- $s_{44}$is the goal state (G), when the agent arrives here, it will receive a reward +10
- s_{23} and s_{32} is a penalty state (P), when the agent arrives there, it will get a negative reward -5
- When the agent arrives all the other states, it will get a negative reward -1
- assuming a discount factor $\gamma = 0.8$

  Assume a deterministic policy is used.

s_{11}, S	s_{12}	s_{13}	s_{14}
s_{21}	s_{22}	s_{23}, P	s_{24}
s_{31}	s_{32}, P	s_{33}	s_{34}
s_{41}	s_{42}	s_{43}	s_{44}, G

Figure 1. the Map of Gridworld.

a) Formulate the Gridworld example as a Markov Decision Process with defining each component.

Formulate the MDP:

State space: $S = \{S_{11},\ S_{12}, \dots, S_{44}\}$

Action space: $A = \{North, South, East, West\}$

Transition probability: given a deterministic policy, $P(s'|s, a) = 1$ for intended direction, and 0 otherwise.

Reward function: $R(S_{44}) = +10, R(S_{23}) = R(S_{32}) = -5$

and $R(other\ states) = -1$

A Markov Decision Process (MDP) is a mathematical framework used for modelling RL problems. In an MDP, an agent interacts with an environment to make decisions and receive feedback.

The basic elements of an MDP:

- State: represent specific states of the system or environment. At each time step, the agent is in a certain state.
- Action: represent the operations or decisions that the agent can take in a given state. The agent selects an action based on the current state to influence the environment.
- Transition probability: represent the probability distribution of transitioning to the next state given the current state and chosen action. It describes the dynamic nature of the environment.
- Reward: represent the immediate reward obtained by the agent after taking a certain action. Immediate rewards can be positive, negative, or zero.
- Discount factor: represents the importance given to future rewards. The discount factor determines the balance between immediate and future rewards.

(Slides page 106 - 107, 111 - 112)

b) Q-learning algorithm, as one of the classical reinforcement learning (RL) algorithms, is commonly used to solve this Markov Decision Process problem. Answer the following few questions about Q-learning algorithm:

  i) There are two types of RL algorithms, model-based RL and model-free RL, which one is Q-learning belong to? Give the reasons.
  Q-learning is model-free RL. It does not require knowledge or **a model of the environment**. Instead, it learns by **interacting with the environment** and receiving feedback in the form of **rewards or penalties**.

Model-Free RL:

- learns strategies directly without the need for an explicit model of the environment.
- Agent interacts with the real environment and relies on real environment feedback and reward for learning, and as a result, it may take irreversible and disruptive actions.

Q-learning belongs to model-free methods.

(Slides page 120, 121)

  ii) Given the update rule of Q-learning as
$$Q^{update}(s_t, a_t) = Q(s_t, a_t) + \alpha[r_{t+1} + \gamma max_a Q(s_{t+1}, a) - Q(s_t, a_t)]$$

  Assume the agent starts from the place s_{11} at $t = 0$, it knows the action is 'East', and thus reaches the place, s_{12} at $t = 1$. The initial Q-value of place s_{12} is set as $Q(s_{12}, a) = 0.2$ and assuming $Q(s_{11}, a) = 0$ and learning rate $\alpha = 0.1$, what is the Q-value update would be?

$$Q^{update}(s_{11}, East) = 0 + 0.1[-1 + 0.8 max_a Q(s_{12}, a) - 0)$$

$$= 0 + 0.1[-1 + 0.8 \cdot 0.2 - 0] = -0.084$$

The equation for updating the Q-value is given by:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t))$$

where α is learning rate, γ is discount factor.

From the question, we know the current Q(s_{11}) is 0, $\alpha = 0.1$, $\gamma = 0.8$, the reward for s_{12} is -1, and Q(s_{12}) is 0.2. Bringing these values into the formula yields the answer.

(Slides page 124 - 130)