**Name: Md Sahadul Haque**

**ID: 21-45417-3**

**Q1) Explain how locally weighted regression differs from linear regression, including their formulas. What is an advantage of locally weighted regression over linear regression?**

**Answer:**

Locally Weighted Regression (LWR) differs from linear regression in that it fits a separate linear regression model for each query point by giving more weight to nearby training examples. The formula for LWR is:

$$\hat{y}(x) = \theta^T \cdot x$$

Where:

$\hat{y}(x)$ is the predicted output for a query point $x$.

$\theta$ is the parameter vector.

$x$ is the feature vector.

The locally weighted sum of squared errors is minimized to find the optimal parameter vector $\theta$.

Linear regression, on the other hand, fits a single global linear model to all the training data points, aiming to minimize the sum of squared errors between the observed and predicted values across all data points. The formula for linear regression is:

$$\hat{y}(x) = \theta^T \cdot x$$

Where the symbols have the same meanings as in LWR.

An advantage of LWR over linear regression is that it allows the model to adapt to different patterns in the data based on the proximity of the query point to the training data. This flexibility makes LWR more suitable for situations where the relationship between the input and output variables may vary across different regions of the input space.

**Q2) Given you want to apply a model to predict whether a patient has malignant or benign tumor, where model output y = 1 means malignant and y = 0 means benign. Explain how the binary logistic regression model is used to train patient data and then predict tumor of a new patient. Include formulas and learning algorithm used in your answer.**

**Answer:**

Binary logistic regression is used to predict the probability that a given input belongs to a particular class (e.g., malignant, or benign tumor). It models the probability of the output $y$ being 1 (malignant) as a logistic function of the input features $xx$ and the model parameters $\theta$:

$$h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$$

The learning algorithm for logistic regression involves optimizing the parameters $\theta$ to minimize the logistic loss function over the training data:

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}[y^{(i)}\log(h_\theta(x^{(i)})) + (1-y^{(i)})\log(1-h_\theta(x^{(i)}))]$$

where $m$ is the number of training examples, $x(i)$ is the feature vector of the $i$-th training example, and $y(i)$ is the corresponding label (1 for malignant, 0 for benign).

To predict the tumor type for a new patient, we plug the new patient's feature vector $xx$ into the logistic function $h\theta(x)$ and classify the tumor as malignant ($y$=1) if $h\theta(x)$ is greater than or equal to 0.5, and benign ($y$=0) otherwise.

**Q3.a) Given the output, y(n), of 3 training items of SoftMax regression are represented by the following one-hot vectors where y ϵ {1,2,3}: y1 = [1 0 0], y2 = [0 1 0] and y3 = [0 0 1]. Write the expanded form of the SoftMax cost function J(w) for these 3 items, and the SoftMax output function f(x;w).**

**Answer:**

The SoftMax cost function $J(w)$ for the given three training items can be expanded as follows:

$$J(w) = -\frac{1}{3}\left[y_1 \log(f(x;w)) + y_2 \log(f(x;w)) + y_3 \log(f(x;w))\right]$$

$$J(w) = -\frac{1}{3}\left[\log(f(x;w)) + \log(f(x;w)) + \log(f(x;w))\right]$$

$$J(w) = -\frac{1}{3} \cdot 3 \log(f(x;w))$$

$$J(w) = -\log(f(x;w))$$

The SoftMax output function $f(x;w)$ for a single class $k$ is defined as:

$$f_k(x;w) = \frac{e^{w_k^T x}}{\sum_{j=1}^{K} e^{w_j^T x}}$$

where:

$fk(x;w)$ is the predicted probability of class $k$.

$wk$ is the weight vector for class $k$.

$K$ is the total number of classes.

**b) What is the relationship between SoftMax and binary logistic regression?**

**Answer:**

The relationship between SoftMax and binary logistic regression is that SoftMax regression is a generalization of binary logistic regression to handle multiple classes. In binary logistic regression, there are only two possible classes, and the logistic function is used to model the probability of the positive class. SoftMax regression extends this to more than two classes by using the SoftMax function to model the probability distribution over all classes.

**Q4) What is the penalty term of ridge/L2 regularization and how does it reduce overfitting?**

**Answer:**

The penalty term of ridge/L2 regularization is the sum of squares of all the model parameters, multiplied by a regularization parameter $\lambda$. The regularization term is added to the cost function to penalize large parameter values. The formula for the ridge regularization term is:

Ridge penalty= $\lambda \sum_{j=1}^{p} \theta_j^2$

where:

$\lambda$ is the regularization parameter.

$p$ is the number of parameters.

$\theta j$ are the model parameters.

Ridge regularization reduces overfitting by discouraging the learning algorithm from fitting large parameter values, thus preventing the model from becoming too complex and sensitive to noise in the training data.

**Q5.a) Write the pseudocode/steps of applying Policy iteration to solve an MDP, including the equations.**

**Answer:**

Steps for applying Policy Iteration to solve an MDP:

Initialize the policy arbitrarily.

Repeat until convergence: a. Policy Evaluation:

Given the current policy, compute the value function for each state using the Bellman Expectation Equation. b. Policy Improvement:

Update the policy based on the current value function by selecting the action that maximizes the expected return for each state. Pseudocode is given below:

```
function PolicyIteration(MDP):
    Initialize random policy π
    repeat
        V = PolicyEvaluation(π, MDP)
        π' = PolicyImprovement(V, MDP)
    until π' = π
    return π
function PolicyEvaluation(π, MDP):
    Initialize V arbitrarily
    repeat
        Δ = 0
        for each state s in MDP:
            v = V(s)
            V(s) = Σ_a π(a|s) Σ_{s', r} p(s', r|s, a)[r + γV(s')]
            Δ = max(Δ, |v - V(s)|)
    until Δ < θ
    return V
function PolicyImprovement(V, MDP):
    policy_stable = true
    for each state s in MDP:
        old_action = π(s)
        π(s) = argmax_a Σ_{s', r} p(s', r|s, a)[r + γV(s')]
        if old_action != π(s):
            policy_stable = false
    if policy_stable:
        return π
    else:
        return PolicyImprovement(V, MDP)
```

**b) What is the advantage of using an exploration-based policy like ϵ-greedy, to solve an MDP?**

**Answer:**

The advantage of using an exploration-based policy like ε-greedy to solve an MDP is that it allows the learning agent to balance between exploration and exploitation. By occasionally choosing random actions with a probability ε, the agent can explore new states and actions, which is essential for discovering optimal or near-optimal policies. This exploration prevents the agent from getting stuck in suboptimal solutions and helps it learn better policies over time.

**Q6.a) What makes Q-learning an off-policy algorithm?**

**Answer:**

Q-learning is an off-policy algorithm because it learns the value of the optimal policy independently of the agent's actions. It updates its Q-values using the maximum Q-value for the next state, regardless of the action chosen by the current policy. This decoupling of the target policy from the behavior policy makes Q-learning an off-policy method.

**b) What is the difference between on-policy and off-policy algorithms?**

**Answer:**

The difference between on-policy and off-policy algorithms lies in how they update their value functions. In on-policy methods, such as SARSA, the agent learns the value of the policy it follows (the target policy). In contrast, off-policy methods, such as Q-learning, learn the value of the optimal policy independently of the agent's actions (the target policy can be different from the behavior policy). This allows off-policy methods to learn from historical data collected by any policy, providing more flexibility and potentially faster learning.