Machine Learning Final Assignment

Name: Punam Das

ID : 21-44946-2

Q1) **Locally Weighted Regression vs Linear Regression:**

- **Linear Regression:** Linear regression aims to fit a linear relationship between the independent variable(s) X and the dependent variable y. The formula for linear regression is:

$y=\beta 0+\beta 1X+\epsilon y=\beta 0+\beta 1X+\epsilon$

Where:

  - y is the dependent variable.
  - X is the independent variable.
  - $\beta 0$ is the intercept.
  - $\beta 1$ is the slope coefficient.
  - ϵ is the error term.

- **Locally Weighted Regression (LWR):** LWR differs from linear regression by giving more weight to data points closer to the point at which a prediction is being made. The formula for LWR is:

- $y = \sum_{i=1}^{n} w_i(y_i - \beta_0 - \beta_1 X_i)$

Where:

  - wi is the weight assigned to each data point.
  - $(yi-\beta 0-\beta 1Xi)$ is the weighted residual.

Advantage of LWR over Linear Regression: The advantage of LWR is that it can capture more complex patterns in the data by considering local information. Linear regression assumes a global linear relationship between variables, which may not always be the case in real-world scenarios.

Q2) **Binary Logistic Regression for Tumor Prediction:**

Binary logistic regression is used to model the probability that a given observation belongs to a particular category (in this case, malignant or benign tumor). The logistic function is used to transform the output of linear regression into a probability between 0 and 1. The model output y is:

$$y = \frac{1}{1+e^{-z}}$$

Where $=\beta 0+\beta 1X1+\beta 2X2+\ldots+\beta nXn$ is the linear combination of features and coefficients.

The learning algorithm for binary logistic regression involves minimizing the logistic loss function using techniques like gradient descent. The logistic loss function is:

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

Where:

- $h\theta(x)$ is the logistic function.
- $y(i)$ is the actual label of the i-th observation.
- $x(i)$ is the feature vector of the i-th observation.

Once trained, the logistic regression model can predict the tumor type (malignant or benign) for a new patient by calculating the probability using the logistic function and applying a threshold (e.g., 0.5) to classify the tumor.

Q3.a) **Softmax Regression:**

- **Cost Function $J(w)$:** For nn training items, the softmax cost function is:

$J(w)=$ $-\frac{1}{n}\sum_{i=1}^{n}\sum_{k=1}^{K} y_k^{(i)}\log(\hat{y}_k^{(i)})$

Where:

- $yk(i)$ is the k-th component of the one-hot encoded label for the i-th item.
- $y\hat{}k(i)$ is the predicted probability of class k for the i-th item.

- **Softmax Output Function *f*(*x*;*w*):** The softmax output function for *K* classes is:

$f(x;w)=\quad \frac{e^{w_j^T x}}{\sum_{k=1}^{K} e^{w_k^T x}}$

Q3.b) **Relationship between Softmax and Binary Logistic Regression:**

Softmax regression is a generalization of binary logistic regression to multiple classes. In binary logistic regression, there are only two possible outcomes (0 or 1), while softmax regression deals with multiple mutually exclusive classes.

Q4) **Ridge/L2 Regularization Penalty Term:**

The penalty term of ridge/L2 regularization is the sum of squares of all the model coefficients (except the intercept term) multiplied by the regularization parameter $\lambda\lambda$. It is added to the cost function to prevent overfitting. The formula for the ridge regularization penalty term is:

Penalty Term= $\lambda \sum_{j=1}^{p} \beta_j^2$

Where:

- λ is the regularization parameter.
- βj are the coefficients of the model.

This penalty term reduces overfitting by shrinking the coefficients towards zero, thereby reducing the model's complexity.

Q5.a) **Policy Iteration for MDP:**

Initialize the value function V(s) arbitrarily

Repeat until convergence {

Policy Evaluation:

for each state s {

$V(s) = \Sigma\_a\ \pi(a|s)\ \Sigma\_s'\ P(s'|s,a)[R(s,a,s') + \gamma V(s')]$

}

Policy Improvement:

for each state s {

$\pi(s) = \text{argmax}\_a\ \Sigma\_s'\ P(s'|s,a)[R(s,a,s') + \gamma V(s')]$

}

}

Q5.b) **Advantage of Exploration-based Policy (ϵ-Greedy):**

The advantage of using an exploration-based policy like ϵ-greedy to solve an MDP is that it allows the agent to explore different actions and learn about the environment. By occasionally choosing random actions (with probability ϵ), the agent can discover potentially better strategies and avoid getting stuck in suboptimal policies.

Q6.a) **Q-Learning as an Off-Policy Algorithm:**

Q-learning is an off-policy algorithm because it learns the value of the optimal policy while following a different (possibly exploratory) policy. It updates the Q-values based on the maximum expected future reward, regardless of the action actually taken.

Q6.b) **Difference between On-Policy and Off-Policy Algorithms:**

- **On-Policy Algorithms:** On-policy algorithms update the policy being followed directly based on the observed experience. They evaluate or improve the same policy that is used to interact with the environment (e.g., SARSA).
- **Off-Policy Algorithms:** Off-policy algorithms learn the value function or optimal policy from data collected using a different policy. They can learn from historical data or explore alternative strategies while still aiming to find the optimal policy (e.g., Q-learning).