Problem 1: Multi-class Classification

Remember that the first method seen in class for multi-class classification is One vs. All. Assuming you have $K$ classes, the One vs. All does the following:

- For each $k \in [0, K]$, assign label 1 to samples from class $k$ and label 0 to samples from all other classes.
- For each $k \in [0, K]$, build a binary classifier using the dataset created in the previous step. The $k$–th classifier outputs the probability of belonging to class $k$ versus all the other classes. (Hence the name One vs. All).
- Given a sample, predict the probability of belonging to class $k$ versus all the other classes for every $k$.
- Choose the $k$ that has the greatest probability.

Let’s get into botany (No prerequisite in botany required ;)).

![Figure 1: Iris Setosa (Top left), Iris Versicolour (Top right), Iris Virginica (Bottom)](image)

(a) (b) (c)
For this question, you will work with the iris dataset. This dataset is a Machine Learning standard for multi-class classification. Here is a description of the data. The features are:

- Sepal length (in cm)
- Sepal width (in cm)
- Petal length (in cm)
- Petal width (in cm)

The classes are:

- Iris Setosa
- Iris Versicolour
- Iris Virginica

Note: you do not need to import the data or code anything. If you are interested in the dataset, you can find it [here].

In this question, we are only using two features: petal length and sepal length.

1. (10 points) The following plot represents the flowers plotted against the two features. Using the One vs. All method, we should build 3 classifiers. For a logistic regression classifier, what is the shape of the decision boundary? Note: no math required, just a general answer is enough.

![Figure 2: Plot of the iris dataset using two features: petal length, sepal length](image)

The shape of the decision boundary of a binary classifier built using Logistic Regression is a hyperplane in the feature space. In two dimensions, it will simply be a line.
2. (10 points) Let’s assume that we trained three classifiers on the data:
   • the Setosa vs. All classifier
   • the Versicolour vs. All classifier
   • the Virginica vs. All classifier

   Given the plot, rank the classifiers by their accuracy. Hint: the answer should not consist of math. Think about classifiers visually.

   Actual accuracies are given in parenthesis:
   1. Setosa vs All (100%)
   2. Virginica vs All (97%)
   3. Versicolour vs All (66%)

Problem 2: Bias-Variance tradeoff

One of the key results in Machine Learning is the Bias-Variance tradeoff. We will see in class the intuition and the implications of this important result but in this question, you will prove the result. In linear regression, the general assumptions are:
   • $Y = f(X) + \epsilon$ (\(\epsilon\) is a random variable that represents noise, \(f\) is deterministic and is the model we use to map \(X\) to \(Y\))
   • \(\epsilon\) is independent of \(X\) hence of \(\hat{f}(X)\)
   • \(\epsilon\) has mean 0 and variance \(\sigma^2\)

Using data, we build \(\hat{f}\); an estimator of the model. Note: this is a random variable because the data is a random variable. The error of this model is defined as:

\[
\text{Err} = \mathbb{E}[(Y - \hat{f}(X))^2]
\]

1. (5 points) Prove the following (for simplification, we note: \(f = f(X), \hat{f} = \hat{f}(X)\)):

\[
\text{Err} = \mathbb{E}[(f - \hat{f})^2] + \mathbb{E}[\epsilon^2] + 2\mathbb{E}[(f - \hat{f})\epsilon]
\]

\[
\text{Err} = \mathbb{E}[(Y - \hat{f})^2] = \mathbb{E}[(f - \hat{f} + \epsilon)^2] = \mathbb{E}[(f - \hat{f})^2] + \mathbb{E}[\epsilon^2] + 2\mathbb{E}[(f - \hat{f})\epsilon]
\]

2. (5 points) Prove the following:

\[
\mathbb{E}[(f - \hat{f})\epsilon] = 0
\]

\(\epsilon\) is independent of \(X\) therefore of \(\hat{f}(X)\). Furthermore, \(f\) is deterministic. Hence:

\[
\mathbb{E}[(f - \hat{f})\epsilon] = \mathbb{E}[(f - \hat{f})]\mathbb{E}[\epsilon] = 0
\]
3. (5 points) We define the bias of the model as the expected distance between the model and the hypothesis: 
\[ \text{Bias} = \mathbb{E}[f - \hat{f}] \]. Prove the following:
\[
\mathbb{E}[(f - \hat{f})^2] = \text{Var}[-\hat{f}] + \mathbb{E}[f - \hat{f}]^2
\]
\[
\mathbb{E}[(f - \hat{f})^2] = \text{Var}[f - \hat{f}] + \mathbb{E}[f - \hat{f}]^2 = \mathbb{E}[f - \hat{f}]^2 \text{ because } f \text{ is deterministic}
\]

4. (5 points) Derive the expression of the error. \textit{Note}: your result should only depend on \( \text{Var}[\hat{f}] \), \text{Bias}, and \( \sigma \).
\[
\text{Err} = \text{Var}[\hat{f}] + \text{Bias}^2 + \sigma^2
\]
\textit{Note}: think about the implication of this equation. For instance, if your model is perfect, what is the best error you can achieve? 0 right? Not really: \( \sigma^2 \). Moreover, think about the tradeoff in terms of complexity. A simpler model will have lower variance but higher variance and vice-versa for a more complex model.
Problem 3: Softmax Classification

In this question, you will see a new method for multi-class classification: the softmax method. Assume there are $K$ classes. We define a weight matrix $\theta$ such that $\theta_k$ represents a row vector of weights used to classify class $i$. The probability assumption of the softmax model for a given class $k$ and datapoint $x$ is the following:

$$P(Y = k | x, \theta) = \frac{1}{Z} e^{\theta_k x}$$

($Z$ is a constant)

**Note 1**: in this case, $x$ is a column vector. So in this problem, each column of $X$ represents a training example.

**Note 2**: $\theta X$ represents a matrix whose coordinate $(i,j)$ is the score of datapoint $j$ belonging to class $i$.

1. (5 points) Compute $Z$, i.e. find an expression of $P(Y = k | x, \theta)$ that only depends on $\theta_k$ and $x$.

   Using the fact that the sum of probabilities must be equal to 1, we have:

   $$P(Y = k | x, \theta) = \frac{e^{\theta_k x}}{\sum_{i=1}^{K} e^{\theta_i x}}$$

2. (5 points) After computing the probability of belonging to class $k$ for all $k$, how do we assign a class? In other words, given the probabilities, what is the decision rule?

   We choose the class that maximizes the probability, i.e. $k = \arg \max_k \left[ P(Y = k | x, \theta) \right]$

3. (5 points) One of the problems of the softmax method, is that if $\theta_k x$ is large, its exponential will be extremely large. Therefore, we will face overflow errors. One of the methods used to mitigate this effect is to replace $\theta_k x$ by $\theta_k x - \alpha$ where $\alpha = \max_j [\theta_j x]$. Show that this method does not change the probability values and explain why overflow is mitigated.

   $$\tilde{P}(Y = k | x, \theta) = \frac{e^{\theta_k x - \alpha}}{\sum_{i=1}^{K} e^{\theta_i x - \alpha}} = \frac{e^{-\alpha} e^{\theta_k x}}{\sum_{i=1}^{K} e^{-\alpha} e^{\theta_i x}} = P(Y = k | x, \theta)$$

   The overflow is mitigated because $\theta_k x - \alpha < 0 \ \forall k$

4. (5 points) The cost function of the softmax method is:

   $$J(\theta) = -\sum_{i=1}^{m} \log \left[ P(Y = y(i) | X = x(i), \theta) \right]$$

   where $y(i)$ is the true label of sample $i$

   Derive the gradient of the loss with respect to $\theta$. Hint: Compute the gradient of the loss with regards to each vector $\theta_k$.

   Let $\delta_{jy(i)}$ be the indicator function equal to 1 when $j = y(i)$ and 0 otherwise. Then, for one datapoint, we have:

   $$J_i(\theta) = -\theta_{jy(i)} x^{(i)} + \log \left( \sum_{i=1}^{K} e^{\theta_i x^{(i)}} \right)$$
This yields:

\[
\frac{\partial J_i(\theta)}{\partial \theta_j} = -x^{(i)\top} \delta_{y^{(i)}} + \frac{1}{K} \sum_{i=1}^{K} e^{\theta_j x^{(i)}} = x^{(i)\top} \left( \mathbb{P}(Y = j|x^{(i)}, \theta) - \delta_{y^{(i)}} \right)
\]

To get the full loss, we just sum over all datapoints.

**Problem 4: SVM Classification**

This question introduces a new method for multi-class classification: the Support Vector Machine (SVM). You will learn about it later in the quarter, but this question provides you with everything you need to know to solve the problem. Assume there are \( K \) classes. We define a weight matrix \( \theta \) such that \( \theta_i \) represents a row vector of weights used to classify class \( i \). (So far the method is similar to softmax). The SVM cost function is then defined as:

\[
J(\theta) = \sum_{i=1}^{m} \sum_{j \neq y^{(i)}} \max(0, \theta_j x^{(i)} - \theta_{y^{(i)}} x^{(i)} + 1)
\]

For notation purposes, we usually replace \( \theta_j x = s_j \).

1. (5 points) Plot the function. The y axis is the cost and the x axis is \( (s_j - s_{y^{(i)}}) \):

\[
s \mapsto \max(0, s_j - s_{y^{(i)}} + 1)
\]

Explain why it is a “good” cost function. *Hint*: think about the margin i.e. the role of the value 1. The margin ensures that the score of the true class is at least 1 above the score of any other class.

![Figure 3: Hinge Loss function](image)

2. (5 points) In the cost function above, 1 is considered a margin. Show that replacing 1 by \( \lambda > 0 \) will create an equivalent problem. *Hint*: you can assume that rescaling the value of \( \theta \) by a positive number
does not change the problem.

\[ \tilde{J}(\theta) = \sum_{i=1}^{m} \sum_{j \neq y(i)} \max(0, \theta_j x(i) - \theta_{y(i)} x(i) + \lambda) \]

\[ = \sum_{i=1}^{m} \sum_{j \neq y(i)} \lambda \max(0, \tilde{\theta}_j x(i) - \tilde{\theta}_{y(i)} x(i) + 1) \]

\[ = \lambda J(\tilde{\theta}) \text{ with } \tilde{\theta} = \frac{1}{\lambda} \theta \]

Therefore the problem is equivalent since scaling by a positive number does not change the minimal value.

3. (5 points) Derive the gradient of the loss with respect to \( \theta \). Hint: Compute the derivative of the loss for each vector \( \theta_k \). For one datapoint, we have:

\[ J_i(\theta) = \sum_{j \neq y(i)} \max(0, \theta_j x(i) - \theta_{y(i)} x(i) + 1) \]

This yields, for a given \( k \) in \([1, K]\):

Case 1: \( k \neq y(i) \)

\[ \frac{\partial J_i(\theta)}{\partial \theta_k} = x(i)^T \delta_{\theta_k \theta y(i)} \delta_{\theta_j \theta (1)} x(i) + 1 \geq 0 \]

Case 2: \( k = y(i) \)

\[ \frac{\partial J_i(\theta)}{\partial \theta_k} = - \sum_{j \neq y(i)} x(i)^T \delta_{\theta_j \theta (1)} \delta_{\theta_j \theta (1)} x(i) + 1 \geq 0 \]

Both expressions simplify as follows:

\[ \frac{\partial J_i(\theta)}{\partial \theta_k} = x(i)^T \delta_{\theta_k \theta y(i)} \delta_{\theta_j \theta x(i)} x(i) + 1 \geq 0 - \delta_{\theta_k \theta (1)} \sum_{j} x(i)^T \delta_{\theta_j \theta x(i)} \delta_{\theta_j \theta (1)} x(i) + 1 \geq 0 \]

To get the full loss, we just sum over all datapoints.

4. (5 points) Given the gradient, what is the update rule in the gradient descent algorithm? Note: no code required.

We apply the rule seen in class:

\[ \theta := \theta - \alpha \frac{\partial J(\theta)}{\partial \theta} \text{ with } \alpha \text{ the learning rate of the algorithm (hyperparameter)} \]
Problem 5: Weighted Linear Regression

In class, we saw that the cost function for linear regression is:

\[ J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 \]

Here, you can notice that all the samples are weighted equally. However, in certain contexts, some samples may be more relevant than others. For instance, suppose you could detect the outliers in the data - e.g. a sensor reporting an incorrect measurement. Then, common sense would suggest to assign small weights to outliers, because you do not want them to influence your model. To take into account the relative importance of each example, you can use Weighted Linear Regression (WLR). The cost function for WLR is:

\[ J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} w^{(i)}(h_\theta(x^{(i)}) - y^{(i)})^2 \]

Each sample is assigned a weight \( w^{(i)} \).

1. (5 points) Show that you can define a matrix \( W \) such that the cost function can be rewritten as:

\[ J(\theta) = (X\theta - Y)^T W (X\theta - Y) \]

Note: to get credit, you need to explicitly specify \( W \).

Set \( W = \frac{1}{2m} * \text{diag}(w^{(1)}, \ldots, w^{(m)}) \). Then:

\[ (X\theta - Y)^T W (X\theta - Y) = \sum_{i=1}^{m} [X\theta - Y]_i w^{(i)} [X\theta - Y]_i = \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) w^{(i)} (h_\theta(x^{(i)}) - y^{(i)}) = J(\theta) \]

Note: the normalization factor does not matter here as it is equivalent to minimize \( \frac{1}{2m} J \) and \( J \).

2. (5 points) Assume that \( \theta \in \mathbb{R}^d \), \( a \in \mathbb{R}^d \), \( A \in \mathbb{R}^{d \times d} \), and \( A \) is symmetric. \( \nabla_\theta \) is the derivative with respect to \( \theta \). Show the following properties:

\[ \nabla_\theta [a^T \theta] = a \]

\[ \nabla_\theta [\theta^T A \theta] = 2A\theta \]

\[ \frac{\partial a^T \theta}{\partial \theta_i} = \frac{\partial}{\partial \theta_i} \left( \sum_{j=1}^{d} a_j \theta_j \right) = a_i \]

Hence the result.

Using the same indicator notation, we have:

\[ \frac{\partial \theta_i \theta_j}{\partial \theta_k} = \delta_{ik} \theta_j + \delta_{jk} \theta_i \]
Therefore,
\[
\frac{\partial \theta^T A \theta}{\partial \theta_k} = \frac{\partial}{\partial \theta_k} \left( \sum_{j=1}^{d} \sum_{i=1}^{d} \theta_i \theta_j A_{ij} \right)
\]
\[
= \sum_{j=1}^{d} \sum_{i=1}^{d} \left[ \delta_{ik} \theta_j + \delta_{jk} \theta_i \right] A_{ij}
\]
\[
= \sum_{j=1}^{d} \theta_j A_{kj} + \sum_{j=1}^{d} \theta_j A_{ik}
\]
\[
= \sum_{j=1}^{d} \theta_j A_{kj} + \sum_{i=1}^{d} \theta_i A_{ki} \quad (A \text{ is symmetric})
\]
\[
= \lbrack 2A \theta \rbrack_k
\]

3. (5 points) In class, we saw that the normal equation for (unweighted) linear regression is:
\[
X^T X \theta = X^T Y \Rightarrow \theta_{\text{min}} = (X^T X)^{-1} X^T Y
\]
Derive the value of \( \theta \) such that it minimizes the WLR cost function. \textit{Hint:} Compute \( \nabla_\theta J(\theta) \) and set \( \nabla_\theta J(\theta) = 0 \) to find \( \theta_{\text{min}} \).

The loss can be rewritten as:
\[
J(\theta) = (X\theta - Y)^T W (X\theta - Y)
\]
\[
= \theta^T X^T W X \theta - \theta^T X^T W Y - Y^T W X \theta + Y^T W Y
\]
\[
= \theta^T X^T W X \theta - 2 [X^T W Y]^T \theta + Y^T W Y \quad \text{because} \quad \theta^T X^T W Y = Y^T W X \theta
\]
Therefore, applying the formula seen before (all conditions apply), we have:
\[
\nabla_\theta J(\theta) = 2X^T W X - 2X^T W^T Y \Rightarrow \theta_{\text{min}} = (X^T W X)^{-1} X^T W^T Y
\]

4. (5 points) We also saw in section a particular example where we used Locally Weighted Linear Regression. We defined \( w(i) \) as
\[
w(i) = e^{\left(-\frac{(x(i) - x)^2}{2\tau^2}\right)}
\]
How would increasing the value of \( \tau \) affect your cost function?

The smaller the \( \tau \), the faster the exponential decreases around \( x \) therefore the more localised it is around \( x \). The converse applies for a large \( \tau \). The extreme cases are always good to have in mind: if \( \tau = 0 \), the weight is zero except in \( x \). If \( \tau = +\infty \) all weights are equal to 1.