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)
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](#)

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.
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 \( f(X) \)
- \( \epsilon \) has mean 0 and variance \( \sigma^2 \)

Using data, we build \( \hat{f} \); an estimator of the model. Note: \( \hat{f} \) is a random variable. However, you can consider that the data \( X \) is given: it’s deterministic, constant, not a random variable; so \( f(X) \) is also deterministic.

The error of the model is defined as:

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

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

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

2. (5 points) Prove the following:

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

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:

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

4. (5 points) Derive the expression of the error. Note: your result should only depend on \( \text{Var}[\hat{f}] \), \( \text{Bias} \), and \( \sigma \).
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 $\theta_i$ 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:

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

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 $x$.

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?

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.
Problem 4: Machine Translation

You are about to build a simple Machine Translation system. Specifically, you will make use of word vectors to create your Machine Translation system. You will create an $X$ matrix where each row corresponds to a word vector trained on the English corpus. You will also create a $Y$ matrix where each row corresponds to a word vector trained on the French corpus. Concretely, $X_i$, the $i$-th row of $X$, is a vector representing word $i$ in English. Similarly, $Y_i$ is a vector representing the French equivalent of word $i$. You will now learn a mapping from $X$ to $Y$ using gradient descent. Specifically, you will minimize the following:

$$ F = \frac{1}{m} \|XR - Y\|^2_F $$

Once you have that mapping, given an English word vector, you can multiply it by $R$ and use the euclidean distance to find the closest French vector. The word for that vector will then be your translation. In this problem we will ask you to compute $\frac{\partial F}{\partial R}$.

Note: $X$ is of dimension $(m, n)$, $R$ is of dimension $(n, n)$, and $Y$ is of dimension $(m, n)$. $m$ corresponds to the number of training examples.

$$ \|A\|_F = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} |a_{ij}|^2} $$ denotes the Frobenius norm.

1. (5 points) Compute the gradient of $F$ with respect to $R$. In other words compute $\frac{\partial F}{\partial R}$. No justification needed, other than making sure your dimensions match.

   (Hint: For some matrix $X$, it holds that: $\frac{\partial}{\partial X} \|X\|^2_F = \frac{\partial}{\partial X} \text{tr}(X^T X)$, where $\text{tr}(X^T X)$ is the trace of the matrix $X^T X$).

2. (5 points) Given the gradient you just computed, how do you update your $R$ variable?
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 \( W \).

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:

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

3. (5 points) In class, we saw that the normal equation for (unweighted) linear regression is:

\[ X^T X \theta = X^T Y \implies \theta_{\text{min}} = (X^T X)^{-1} X^T Y \]

Derive the value of \( \theta \) such that it minimizes the WLR cost function. Hint: Compute \( \nabla_\theta J(\theta) \) and set \( \nabla_\theta J(\theta) = 0 \) to find \( \theta_{\text{min}} \).

4. (5 points) We also saw in section a particular example where we used Locally Weighted Linear Regression. We defined \( w^{(i)} \) as

\[ w^{(i)} = e^{-\frac{(x^{(i)} - x)^2}{2\tau^2}} \]

How would increasing the value of \( \tau \) affect your cost function?