AI534 — Written Homework Assignment 1 (30 pts + 6 bonus pts)

This written assignment covers the contents of linear regression and logistic regression. The key concepts covered here include:

- Maximum likelihood estimation (MLE)
- Gradient descent learning
- Decision theory for probabilistic classifiers
- Maximum A Posteriori (MAP) parameter estimation
- Perceptron

1. **MLE for uniform distribution. [3pt]**
Given a set of IID observed samples $x_1, ..., x_n \sim \text{uniform}(0, \theta)$, we wish to estimate the parameter θ.

   (a) (1 pt) Write down the likelihood function of θ.

$$L(\theta) = \prod_{i=1}^{n} p(x_i; \theta) \tag{1}$$

$$= \prod_{i=1}^{n} \frac{1}{\theta} I(x_i \le \theta) = \begin{cases} \frac{1}{\theta^n} & \text{if } \forall x_i \le \theta \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

   (b) (2 pts) Derive the maximum likelihood estimation for θ, which is the value for *theta* that maximizes the function of part (a). (Hint: The likelihood function is a monotonic function. So the maximizing solution is at the extreme— there is no need to take derivative for this case.)

$$\underset{\theta}{\text{argmax}}\, L = \underset{\theta}{\text{argmax}}\, \frac{1}{\theta^n} \quad \text{subject to} \quad \forall x_i \le \theta \tag{3}$$

$$= \max\{x_1, ..., x_n\} = x_{max} \tag{4}$$

   From (3) to (4) is because in order to maximize L, we want θ to be as small as possible, but θ has to be no smaller than any observed values (otherwise L goes to zero). So the smallest value θ is allowed to take is x_{max}. .

2. **Weighted linear regression. [10pt]** In class when discussing linear regression, we assume that the Gaussian noise is iid (identically independently distributed). In practice, we may have some extra information regarding the fidelity of each data point. For example, we may know that some examples have higher noise variance than others. To model this, we can model the noise variable $\epsilon_i, \epsilon_2, \cdots \epsilon_n$ as distinct Gaussian's, i.e., $\epsilon_i \sim N(0, \sigma_i^2)$ with known variance σ_i^2. How will this influence our linear regression model? Let's work it out.

   (a) (3pts) Write down the log likelihood function of $\mathbf{w}$ under this new modeling assumption.
   First we have $p(y_i|\mathbf{x}_i; \mathbf{w}) \sim N(\mathbf{w}^T\mathbf{x}_i, \sigma_i)$
   *Because we assume all data points are **independently** distributed, the joint of all data points can still be factored as the product of individual ones.*

$$l(\mathbf{w}) = \log \prod_{i-1}^{n} p(y_i|\mathbf{x}_i, \mathbf{w}) = \sum_{i=1}^{n} \log P(y_i|\mathbf{x}_i^T\mathbf{w}, \sigma_i^2) \tag{5}$$

$$= \sum_{i=1}^{n} \log \frac{1}{\sqrt{2\pi}\sigma_i} e^{-\frac{1}{2\sigma_i^2}(y_i - \mathbf{x}_i^T\mathbf{w})^2} \tag{6}$$

$$= \sum_{i=1}^{n} \log \frac{1}{\sqrt{2\pi}\sigma_i} - \sum_{i=1}^{N} \frac{1}{2\sigma_i^2}(y_i - \mathbf{x}_i^T\mathbf{w})^2 \tag{7}$$

(b) (1pts) Show that maximizing the log likelihood is equivalent to minimizing a **weighted square loss function** $J(\mathbf{W}) = \sum_{i=1}^{n} a_i(\mathbf{w}^T\mathbf{x}_i - y_i)^2$, and express each a_i in terms of σ_i.
Maximizing the log likelihood is equivalent to minimizing the second term in the above equation, which can be represented as:

$$J(\mathbf{w}) = \sum_{i=1,\ldots n} a_i(y_i - \mathbf{x}_i^T\mathbf{w})^2 \tag{8}$$

where $a_i = \frac{1}{\sigma_i^2}$. Note that you can also use $a_i = \frac{1}{2\sigma_i^2}$, which is equivalent.

(c) (3 pts) Take the gradient of the loss function $J(\mathbf{w})$ and provide the batch gradient descent update rule for optimizing $\mathbf{w}$.
Take the gradient of J (using equation 8):

$$\nabla J(\mathbf{w}) = -2\sum_{i=1}^{n} a_i(y_i - \mathbf{x}_i^T\mathbf{w})\mathbf{x}_i \tag{9}$$

The update rule is as follows:

$$\mathbf{w} \leftarrow \mathbf{w} + 2\lambda\sum_{i} a_i(y_i - \mathbf{x}_i^T\mathbf{w})\mathbf{x}_i$$

As can be seen from this solution, this is equivalent to weighting each instance differently according to the noise variance for each instance. Instances with larger noise variance are less reliable (due to larger noise), and thus contributes less (due to smaller weight) in the learning process, which is intuitive.

(d) (3 pts) Derive a closed form solution to this optimization problem. Hint: begin by rewrite the objective into matrix form using a diagonal matrix A with $A(i,i) = a_i$.
Let $\mathbf{y} = \{y_1, y_2, \cdots, y_n\}^T$ be the vector storing all the y values of the training examples, and $\mathbf{X}$ be the data matrix whose rows correspond to training examples, and A be a diagonal matrix with $A(i,i) = a_i$. The objective can written in the following matrix form:

$$J(\mathbf{w}) = (\mathbf{y} - \mathbf{X}\mathbf{w})^T A(\mathbf{y} - \mathbf{X}\mathbf{w})$$

Take the gradient of J in vector form and set it to zero:

$$\nabla J(\mathbf{w}) = \mathbf{X}^T A(\mathbf{X}\mathbf{w} - \mathbf{y}) = 0$$

$$2\mathbf{X}^T A\mathbf{X}\mathbf{w} = 2\mathbf{X}^T A\mathbf{y}$$

$$\mathbf{w}^* = (\mathbf{X}^T A\mathbf{X})^{-1}\mathbf{X}^T A\mathbf{y}$$

3. **Decision theory: working with expectations. [6pt]**
In this problem, you will analyze a scenario where the Maximum A-Posteriori (MAP) decision rule, which you learned in class, is not appropriate. Instead, you'll explore how to make decisions based on minimizing expected costs.

Consider a spam filter that predicts whether an email is spam, using probabilistic predictions. For this filter, there are costs associated with making errors (misclassifying emails), but these costs are not symmetric. Misclassifying a non-spam email as spam (i.e., filtering out an important email) is more costly than misclassifying a spam email as non-spam.

The following table shows the cost of each possible outcome:

- If the filter's prediction is correct, there is no cost.

predicted label $\hat{y}$	true label y	
	non-spam	spam
non-spam	0	1
spam	10	0

Table 1: A mis-classification cost matrix for the spam filter problem.

- If a non-spam email is classified as spam, there is a cost of 10.
- If a spam email is classified as non-spam, there is a cost of 1.

Here we will go through some questions to help you figure out how to use the probability and misclassification costs to make predictions.

(a) (2 pt) You received an email for which the spam filter predicts that it is a spam with $p = 0.8$. We want to make the decision that minimizes *the expected cost*.
Question: Should you classify this particular email as spam or non-spam? [Hint: Compare the expected cost of classifying the email as spam versus non-spam. Choose the classification that results in the lower expected cost.]
For the option of classifying it as a spam, the expected cost would be

$$0.2 * 10 + 0.8 * 0 = 2$$

For the option of classifying it as non-spam, the expected cost would be

$$0.2 \times 0 + 0.8 \times 1 = 0.8$$

Predict nonspam, because the expected cost is lower if we classify it as nonspam.

(b) (2 pts)The MAP decision rule would classify an email as spam if $p > 0.5$, but this rule does not minimize expected cost in this case. We need a new rule that compares p to a different threshold θ. The value of θ should be chosen to minimize the expected cost based on the costs in the table.
Question: What is the value of θ that works for the costs specified in Table 1? [Hint: To find the threshold θ, set up the decision rule by comparing the expected cost of each decision, as you did in (a), then Solve for p in terms of the costs.]
To minimize the expected cost, we would predict spam if its expected cost is lower than that of predicting non-spam. Lets write out the expected cost for each option.
Expected cost for predicting spam:

$$p \times 0 + (1 - p) \times 10 = 10 - 10p$$

Expected cost for predicting non-spam:

$$p \times 1 + (1 - p) \times 0 = p$$

We will predict spam if

$$10 - 10p < p \Rightarrow$$
$$p > \frac{10}{11}$$

So the θ for this given cost table is 10/11. This is consistent with our intuition that is due to the much higher cost of mis-classifying a non-spam email as spam, we have to be very conservative and only filter out an email if it's likelihood of being spam is very high ($> 90.9\%$ in this case).

(c) (2pts) Now, imagine that the optimal decision rule would use $\theta = 1/5$ as the threshold for classifying an email as spam. **Question**: Can you provide a new cost table where this would be the case? [Hint: Use the relationship between the costs and θ that you derived in part (b). Based on this relationship, adjust the misclassification costs in the table to achieve $\theta = 1/5$.]

predicted label $\hat{y}$	*true label y*	
	non-spam	*spam*
non-spam	*0*	*4*
spam	*1*	*0*

You can verify as follows: The expected cost for predicting spam:

$$p \times 0 + (1 - p) \times 1 = 1 - p$$

Expected cost for predicting non-spam:

$$p \times 4 + (1 - p) \times 0 = 4$$

We will predict spam if

$$1 - p < 4p \Rightarrow$$

$$p > \frac{1}{5}$$

Note that you can derive this rule for a general table:

predicted label $\hat{y}$	*true label y*	
	non-spam	*spam*
non-spam	*0*	*b*
spam	a	*0*

And the threshold for this general cost matrix will be $\frac{a}{a+b}$

4. **Maximum A-Posteriori Estimation. [8pt]** Suppose we observe the values of n IID random variables $X_1, \ldots, X_n$ drawn from a single Bernoulli distribution with parameter θ. In other words, for each X_i, we know that $P(X_i = 1) = \theta$ and $P(X_i = 0) = 1 - \theta$. In the Bayesian framework, we treat θ as a random variable, and use a prior probability distribution over θ to express our prior knowledge/preference about θ. In this framework, $X_1, \ldots, X_n$ can be viewed as generated by:

   - First, the value of θ is drawn from a given prior probability distribution
   - Second, $X_1, \ldots, X_n$ are drawn independently from a Bernoulli distribution with this θ value.

   In this setting, Maximum A-Posteriori (MAP) estimation is a way to estimate θ by finding the value that maximizes the posterior probability, given both its prior distribution and the observed data.The MAP estimation of θ is given by:

$$\hat{\theta}_{MAP} = \underset{\hat{\theta}}{\operatorname{argmax}} P(\theta = \hat{\theta} | X_1, \ldots, X_n)$$

   By applying Bayes' theorem, this becomes:

$$\hat{\theta}_{MAP} = \underset{\hat{\theta}}{\operatorname{argmax}} P(X_1, \ldots, X_n | \theta = \hat{\theta}) P(\theta = \hat{\theta}) = \underset{\hat{\theta}}{\operatorname{argmax}} L(\hat{\theta}) p(\hat{\theta})$$

   where $L(\hat{\theta})$ is the likelihood function of the data given θ, and $p(\hat{\theta})$ is the prior distribution over θ. Now consider using a beta distribution as the prior: $\theta \sim Beta(\alpha, \beta)$, whose PDF function is

$$p(\hat{\theta}) = \frac{\hat{\theta}^{(\alpha-1)}(1 - \hat{\theta})^{(\beta-1)}}{B(\alpha, \beta)}$$

   where $B(\alpha, \beta)$ is a normalizing constant.

(a) (3 pts) Derive the posterior distribution $p(\hat{\theta}|X_1, \ldots, X_n, \alpha, \beta)$. Compare the form of the posterior distribution with that of the beta distribution, you will see the posterior is also a beta distribution. What the updated α and β parameters for the posterior?

$$\begin{aligned} p(\hat{\theta}|X_1, \ldots, X_n) &\propto P(X_1, \ldots, X_n|\theta = \hat{\theta})p(\hat{\theta}) \\ &\propto \prod_{i=1}^{n} \hat{\theta}^{x_i}(1-\hat{\theta})^{(1-x_i)}p(\hat{\theta}) \\ &\propto \hat{\theta}^{\sum_{i=1}^n x_i}(1-\hat{\theta})^{\sum_{i=1}^n (1-x_i)}p(\hat{\theta}) \\ &\propto \hat{\theta}^{\sum_{i=1}^n x_i}(1-\hat{\theta})^{\sum_{i=1}^n (1-x_i)}\hat{\theta}^{(\alpha-1)}(1-\hat{\theta})^{(\beta-1)} \\ &\propto \hat{\theta}^{(\alpha+\sum_{i=1}^n x_i-1)}(1-\hat{\theta})^{(\beta+\sum_{i=1}^n (1-x_i)-1)} \end{aligned}$$

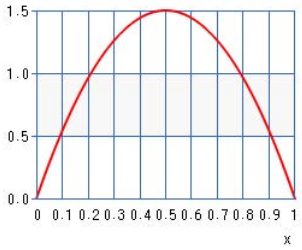
To make this a valid PDF function, we simply need to use $B\left(\alpha + \sum_{i=1}^n x_i, \beta + \sum_{i=1}^n (1-x_i)\right)$ as the denominator. So the posterior $\sim Beta(\alpha + \sum_{i=1}^n x_i, \beta + \sum_{i=1}^n (1-x_i))$.

(b) (2 pts) Suppose we use $Beta(2, 2)$ as the prior, what Beta distribution do we get for the posterior after we observe 5 coin tosses and 2 of them are head? What is the posterior distribution of θ after we observe 50 coin tosses and 20 of them are head? (you don't need to write out the distributions, simply provide the α and β distribution would suffice.

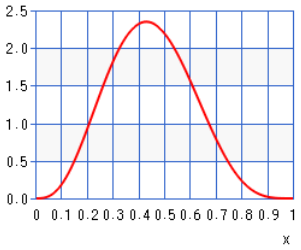
(c) (1pt) Plot the pdf function of the prior $Beta(2, 2)$ and the two posterior distributions. You can use any software (e.g., R, Python, Matlab) for this plot.

(d) (2pt) Assume that $\theta = 0.4$ is the true probability, as we observe more and more coin tosses from this coin, how would the shape of the posterior change as more data is observed? Will the MAP estimate converge toward the true value?

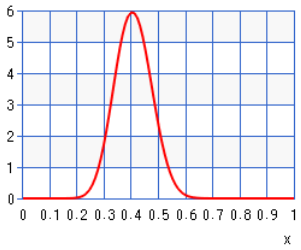
With prior $Beta(2, 2)$ and an observation of n_1 heads and n_0 tails, the posterior for θ is $Beta(2 + n_1, 2 + n_0)$. So after observing 5 coin tosses with 2 heads, the posterior of θ becomes $Beta(4, 5)$. With 50 tosses and 20 heads, the posterior becomes $Beta(22, 32)$. You can see the pdf functions of the prior, and the two posteriors as follows.



(a) $Beta(2, 2)$



(b) $Beta(4, 5)$



(c) $Beta(22, 32)$

As can be seen from the figure, the posterior becomes more and more peaked as we increase the observations. Eventually, all of the probability will concentrate at the true θ value. This is one nice property about the Bayesian approach: with proper prior, when we have very little data, we can fall back onto the prior to avoid catastrophic choices, and as we have more and more data we start to focus primarily on the evidence from the data and the influence of the prior becomes increasingly neglectable.

Mathematically speaking there can be exceptions to this — for example, if the prior puts zero probability around the true parameter value θ^*, no amount of data can actually recover from that because the posterior will be zero at θ^* due to $p(\theta^*) = 0$

5. **Perceptron.** **[3pt]** Assume a data set consists only of a single data point $\{(x, +1)\}$. How many times would the Perceptron algorithm mis-classify this point $\mathbf{x}$ before convergence? What if the initial weight vector $\mathbf{w}_0$ was initialized randomly and not as the all-zero vector? Derive the number of times as a function of $\mathbf{w}_0$ and $\mathbf{x}$.

(a) (1 pts) Case 1: $\mathbf{w}_0 = 0$.

Exactly once. Since $\mathbf{w}_1 = \mathbf{w}_0 + \mathbf{x}$ and $\mathbf{w}_1^T\mathbf{x} = |\mathbf{x}|^2 > 0$. After one update it will be correct.

(b) (2 pts) Case 2: $\mathbf{w}_0! = 0$:
This will depend on the $\mathbf{w}_0$. At the k-th update, we have

$$\mathbf{w}_k = \mathbf{w}_{k-1} + \mathbf{x}$$

, so it follows that

$$\mathbf{w}_i^T\mathbf{x} = \mathbf{w}_{k-1}^T x + |\mathbf{x}|^2 = \mathbf{w}_{k-2}^T\mathbf{x} + 2|\mathbf{x}|^2 = ... = \mathbf{w}_0^T\mathbf{x} + k|\mathbf{x}|^2,$$

Assuming that $\mathbf{w}_0^T\mathbf{x} < 0$, $k = \frac{-\mathbf{w}_0^T\mathbf{x}}{|\mathbf{x}|^2}$

6. **Bonus: MLE for multi-class logistic regression.** **[6 pts]** Consider the maximum likelihood estimation problem for multi-class logistic regression using the soft-max function defined below:

$$p(y = k|\mathbf{x}) = \frac{\exp(\mathbf{w}_k^T\mathbf{x})}{\sum_{j=1}^K \exp(\mathbf{w}_j^T\mathbf{x})}$$

We can write out the likelihood function as:

$$L(\mathbf{w}) = \prod_{i=1}^N \prod_{k=1}^K p(y = k|\mathbf{x}_i)^{y_{ik}}$$

where y_{ik} is an indicator variable taking value 1 if $y_i = k$.

(a) (1 pts) Provide the log-likelihood function. *Take the log of the likelihood function, we have:*

$$l(\mathbf{w}) = \sum_{i=1}^N \sum_{k=1}^K y_{ik} \log p(y_i = k|\mathbf{x}_i) = \sum_{i=1}^N \sum_{k=1}^K y_{ik} \log \frac{\exp(\mathbf{w}_k^T\mathbf{x})}{\sum_{j=1}^K \exp(\mathbf{w}_j^T\mathbf{x})} = \sum_{i=1}^N \sum_{k=1}^K y_{ik}(\mathbf{w}_k^T x - log \sum_{j=1}^K exp(\mathbf{w}_j^T\mathbf{x}))$$

(b) (5 pts) Derive the gradient of the log-likelihood function w.r.t the weight vector $\mathbf{w}_c$ of class c. [Hint: the solution to this problem is provided in the Logistic regression lecture slide. You just need to fill in the missing derivation. Note that for any example $\mathbf{x}_i$, the denominator in the softmax function $\sum_j \exp(\mathbf{w}_j^T\mathbf{x}_i)$ is the same for all k— denoting it as z_i makes it simpler to work through the derivation, but be sure to remember that z_i is a function of all $\mathbf{w}_k$'s.]

We want to take partial gradient with respect to $\mathbf{w}_c$. Before doing that, let's break l into two parts:

$$l(\mathbf{w}) = \sum_{k \neq c}^K \sum_{y_i = k} \log p(y_i = k|\mathbf{x}_i) + \sum_{y_i = c} \log p(y_i = c|\mathbf{x}_i)$$

Note that we have

$$p(y_i = k|\mathbf{x}_i) = \frac{\exp \mathbf{w}_k \cdot \mathbf{x}_i}{\sum_{j=1}^K \exp(\mathbf{w}_j \cdot \mathbf{x}_i)}$$

Take the log:

$$\log p(y_i = k|\mathbf{x}_i) = \mathbf{w}_k \cdot \mathbf{x}_i - \log(\sum_{j=1}^K \exp(\mathbf{w}_j \cdot \mathbf{x}_i))$$

Let $z_i = \sum_{j=1}^K \exp(\mathbf{w}_j \cdot \mathbf{x}_i)$, we have:

$$\log p(y_i = k|\mathbf{x}_i) = \mathbf{w}_k \cdot \mathbf{x}_i - \log z_i$$

Plug this into l, we have:

$$l(\mathbf{w}) = \sum_{k \neq c}^{K} \sum_{y_i = k} (\mathbf{w}_k \cdot \mathbf{x}_i - \log z_i) + \sum_{y_i = c} (\mathbf{w}_c \cdot \mathbf{x}_i - \log z_i)$$

Now take the partial gradient:

$$\begin{aligned} \frac{\partial}{\partial \mathbf{w}_c} l &= -\sum_{k \neq c}^{K} \sum_{y_i = k} \frac{1}{z_i} \frac{\partial z_i}{\partial \mathbf{w}_c} + \sum_{y_i = c} \left(\mathbf{x}_i - \frac{1}{z_i} \frac{\partial z_i}{\partial \mathbf{w}_c} \right) \\ &= \sum_{y_i = c} \mathbf{x}_i - \sum_{k=1}^{K} \sum_{y_i = k} \frac{1}{z_i} \frac{\partial z_i}{\partial \mathbf{w}_c} \end{aligned}$$

Note that the second double summation can be simplified to $\sum_{i=1}^{N}$, where N is the total number of points. We now plug in

$$\frac{\partial z_i}{\partial \mathbf{w}_c} = \mathbf{x}_i \exp(\mathbf{w}_c \cdot \mathbf{x}_i)$$

and $z_i = \sum_{j=1}^{K} \exp(\mathbf{w}_j \cdot \mathbf{x}_i)$, we have:

$$\frac{\partial}{\partial \mathbf{w}_c} l = \sum_{y_i = c} \mathbf{x}_i - \sum_{i=1}^{N} \frac{\exp(\mathbf{w}_c \cdot \mathbf{x}_i)}{\sum_{j=1}^{K} \exp(\mathbf{w}_j \cdot \mathbf{x}_i)} \mathbf{x}_i$$

We will use $\hat{y}_{ic}$ (intuitively meaning the probability of instance i belong to class c) to denote

$$\frac{\exp(\mathbf{w}_c \cdot \mathbf{x}_i)}{\sum_{j=1}^{K} \exp(\mathbf{w}_j \cdot \mathbf{x}_i)}$$

and use y_{ic} to denote a binary indicator variable such that $y_{ic} = 1$ if $y_i = c$ and 0 otherwise. Putting these new notations to use, we arrive at:

$$\begin{aligned} \frac{\partial}{\partial \mathbf{w}_c} l &= \sum_{i=1}^{N} y_{ic} \mathbf{x}_i - \sum_{i=1}^{N} \hat{y}_{ic} \mathbf{x}_i \\ &= \sum_{i=1}^{N} (y_{ic} - \hat{y}_{ic}) \mathbf{x}_i \end{aligned}$$

Therefore, a gradient ascent update rule will be:

$$\forall c \in \{1, \cdots, k\}\text{: } \mathbf{w}_c \leftarrow \mathbf{w}_c + \lambda \sum_{i=1}^{N} (y_{ic} - \hat{y}_{ic}) \mathbf{x}_i$$