AI534 — Written Homework Assignment 1 (32 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.) (3pts)
Given a set of i.i.d. observed samples $x_1, x_2, ..., x_n \sim \text{uniform}(0, \theta)$, we wish to estimate the parameter θ.

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

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

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

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

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

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

   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) (10 pts) 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 Gassians, 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.*

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

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

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

(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:

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

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) Derive a batch gradient descent update rule for optimizing this objective.
Take the gradient of J (using equation 8):

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

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)
For this problem, we will work through an scenario where using the Maximum A-Posteriori decision rule as described in class is not appropriate. Consider a spam filter that gives a probabilistic prediction for each email being a spam. Critically, there is a cost associated with filtering out non-spam emails as well as letting spam email through. But the cost is not symmetric. The following table specifies the costs.

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.

As you can see, if the prediction is correct, there is no cost. But if you mis-classify a non-spam email as a spam, there is a significantly higher cost (10) than the other way around (1).

Here we will go through some questions to help you figure out how to use the probability to make filtering decisions.

(a) (1 pt) For a given email $\mathbf{x}$, the spam filter predicts it being a spam with $p = 0.8$, what is the expected cost of classifying it as *spam*?
If the email is non-spam, classifying it as spam would incur a cost of 10, this outcome has 0.2 probability.
If the email is indeed spam, the cost is 0. This outcome has 0.8 probability.
So the expected cost for classifying it as spam would be

$$0.2 \times 10 + 0.8 \times 0 = 2$$

(b) (1 pt) We want to minimize the expected cost, how should we classify this particular email?
For the option of classifying it as non-spam, the expected cost would be

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

, which is lower than the expected cost of classifying it as spam. Hence classifying it as non-spam will minimize the expected cost.

(c) (2pts) As you can see from parts (a) and (b), using the MAP decision rule (aka comparing p to 0.5) to make prediction for this email is not appropriate and leads to higher expected mis-classification cost. Please devise a decision rule that compares p, the predicted probability of being spam, to a new threshold θ such that we can minimize the expected misclassification cost based on the costs specified in Table 1.
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).

(d) (2pts) Can you provide a cost table for which the decision rule minimizing the expected misclassification cost would use a $\theta = 1/5$?

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. (8 pts) (Maximum A-Posteriori Estimation.) 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 natural way to estimate the value of θ by choosing the most probable value given both its prior distribution and the observed data $X_1, \ldots, X_n$. Specifically, the MAP estimation of θ is given by

$$\begin{aligned}\hat{\theta}_{MAP} &= \operatorname*{argmax}_{\hat{\theta}} P(\theta = \hat{\theta} | X_1, \ldots, X_n) \\ &= \operatorname*{argmax}_{\hat{\theta}} P(X_1, \ldots, X_n | \theta = \hat{\theta}) P(\theta = \hat{\theta}) \\ &= \operatorname*{argmax}_{\hat{\theta}} L(\hat{\theta}) p(\hat{\theta})\end{aligned}$$

   where $L(\hat{\theta})$ is the data likelihood function and $p(\hat{\theta})$ is the density function of the prior. Now consider using a beta distribution for 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 to make it a proper probability density function.

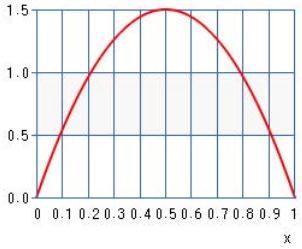
(a) (4 pts) Derive the posterior distribution $p(\hat{\theta}|X_1, \ldots, X_n, \alpha, \beta)$ and show that it is also a Beta distribution.

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

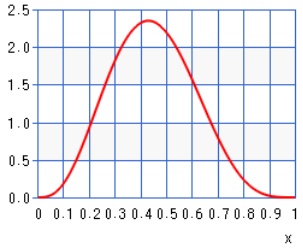
To make this a valid PDF function, we simply need to use $B\left(\alpha + \sum_{i=1}^n x_i, \beta + \sum_{i=1}^n (1-x_i)\right)$ as the denominator. So the posterior $\sim Beta(\alpha + \sum_{i=1}^n x_i, \beta + \sum_{i=1}^n (1-x_i))$.

(b) (4 pts) Suppose we use $Beta(2,2)$ as the prior, what is the posterior distribution of θ 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? Plot the pdf function of the prior as well as the two posterior distributions (you can use any software for this). Assume that $\theta = 0.4$ is the true probability, as we observe more and more coin tosses from this coin, what do you expect to happen to the posterior?

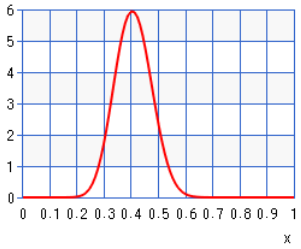
With prior $Beta(2,2)$ and an observation of n_1 heads and n_0 tails, the posterior for θ is $Beta(2+n_1, 2+n_0)$. So after observing 5 coin tosses with 2 heads, the posterior of θ becomes $Beta(4,5)$. With 50 tosses and 20 heads, the posterior becomes $Beta(22,32)$. You can see the pdf functions of 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) (3 pts) 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: 10 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) (2 pts) Compute 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) (8 pts) Compute the gradient of the log-likelihood function w.r.t the weight vector $\mathbf{w}_c$ of class c.
[1]

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:

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

[1] Here are a few tips to help you work through the math. First, the Logistic regression lecture slide actually provides the solution to this problem. But you will need to fill in the missing derivation. Second, for a particular 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. So denoting it as z_i can make it simpler to work through the derivation. But be sure to remember that z_i is a function of all $\mathbf{w}_k$'s)

Now take the partial gradient:

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

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:

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

Therefore, a gradient ascent update rule will be:

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