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

Woonki Kim
kimwoon@oregonstate.edu

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.

   PDF of the uniform distribution uniform$(0, \theta)$:
   $f(x|\theta) = \frac{1}{\theta}$ *for* $0 \leq x \leq \theta$.

   Likelihood estimation for θ
   $L(\theta) = P(x_1, \ldots, x_n|\theta) = \prod_{i=1}^{n} p(x_i|\theta) = \begin{cases} \frac{1}{\theta^n} & \text{if } \forall X_i \leq \theta \\ 0 & \text{otherwise} \end{cases}$

   (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.)
   To derive maximum likelihood estimation for θ*, we need to find* θ *that maximizedthe likelihood function* $L(\theta)$*:*

   $\hat{\theta}_{MLE} = \arg\max_\theta L(\theta) = \arg\max_\theta \frac{1}{\theta^n} \quad , \quad \forall x_i \leq \theta$

   Log likelihood function:$\log L(\theta) = \log\left(\frac{1}{\theta^n}\right) = -n\log(\theta)$

   The function above shows that the log-likelihood decreases as θ *increases. Therefore, to maximize the likelihood, we should choose the smallest possible value of* θ *that satisfies* $\theta \geq \max(x_1, \ldots, x_n)$*. Which means that the MLE for* θ *is:*

   $\hat{\theta} = \max\{x_1, \ldots, x_n\}$

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.

   Likelihood function:

$L(w) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma_i^2}} \exp\left(-\frac{(y_i - w^T x_i)^2}{2\sigma_i^2}\right)$.

Log likelihood function:

$\log L(w) = \sum_{i=1}^{n} \left(-\frac{1}{2}\log(2\pi\sigma_i^2) - \frac{(y_i - w^T x_i)^2}{2\sigma_i^2}\right) = \log \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma_i^2}} \exp\left(-\frac{(y_i - w^T x_i)^2}{2\sigma_i^2}\right)$
$= \sum_{i=1}^{n} \log\left(\frac{1}{\sqrt{2\pi\sigma_i^2}} \exp\left(-\frac{(y_i - w^T x_i)^2}{2\sigma_i^2}\right)\right)$
$= \sum_{i=1}^{n} \left(\log\left((2\pi\sigma_i^2)^{-\frac{1}{2}}\right) + \log\left(\exp\left(-\frac{(y_i - w^T x_i)^2}{2\sigma_i^2}\right)\right)\right)$
$= \sum_{i=1}^{n} \left(-\frac{1}{2}\log(2\pi\sigma_i^2) - \frac{(y_i - w^T x_i)^2}{2\sigma_i^2}\right)$

(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.

We can ignore $-\frac{1}{2}\log(2\pi\sigma_i^2)$ when maximizing the log likelihood, as it is not dependent to w, it is a constant.

So our goal is to maximize $L(w)$: $\sum_{i=1}^{n} -\frac{(y_i - w^T x_i)^2}{2\sigma_i^2}$

Which is equivalent to minimizing $-L(w)$: $\sum_{i=1}^{n} -\frac{(y_i - w^T x_i)^2}{2\sigma_i^2}$

Therefore, maximizing log-likelihood is equivalent to minimizing a weighted square loss function:

$J(W) = \sum_{i=1}^{n} a_i(w^T x_i - y_i)^2$,where $a_i = \frac{1}{\sigma_i^2}$.

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

Weighted square loss function: $J(w) = \sum_{i=1}^{n} a_i(w^T x_i - y_i)^2$.

Gradient of $J(w)$:

$\nabla J(w) = \sum_{i=1}^{n} (\nabla_w \left((w^T x_i - y_i)^2\right))$

$= \sum_{i=1}^{n} (2(w^T x_i - y_i) \cdot \nabla_w (w^T x_i - y_i))$ *(By the chain rule.)*

$= \sum_{i=1}^{n} 2((w^T x_i - y_i)x_i)$

Therefore, the gradient descent update rule for optimizing w:
$w \leftarrow w - \alpha \sum_{i=1}^{n} 2a_i(w^T x_i - y_i)x_i$, α is the learning rate.

(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 $A(i, i) = a_i$.

Rewriting the objective function in matrix form:

- $X \in \mathbb{R}^{n \times d}$: matrix of input data where each row is x_i^T,

- $w \in \mathbb{R}^d$: weight vector.
- $y \in \mathbb{R}^n$: vector of observed outputs,
- $A \in \mathbb{R}^{n \times n}$: diagonal matrix where $A(i,i) = a_i$.

$J(w) = \sum_{i=1}^{n} a_i (w^T x_i - y_i)^2 = (Xw - y)^T A(Xw - y)$

$= (Xw)^T A(Xw) - y^T A(Xw) + -yA(Xw)^T + y^T Ay$

$= (Xw)^T A(Xw) - 2y^T A(Xw) + y^T Ay$ *(Since* $-y^T A(Xw) = -yA(Xw)^T$*)*

Set $\nabla J(w)$ *to 0:*

$\nabla J(w) = 2X^T AXw - 2X^T Ay = 0$

$\implies 2X^T AXw = 2X^T Ay$

$\implies w = (X^T AX)^{-1} X^T Ay$

Thus, closed-form solution for w is:

$w = (X^T AX)^{-1} X^T Ay$

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:

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

The expected cost of classifying the email as non-spam is:
$Cost_{\hat{y}=\text{non-spam}} = (0.8) \cdot 1 + (0.2) \cdot 0 = 0.8$

When the prediction is correct (i.e., classifying non-spam as non-spam), there is no cost, represented by $(0.2) \cdot 0$. On the other hand, when the email is wrongly classified as non-spam, the expected cost is $(0.8) \cdot 1$, based on the given cost table. Therefore, the expected cost of classifying the email as non-spam is 0.8.

The expected cost of classifying the email as spam is:

$$Cost_{\hat{y}=\text{spam}} = (0.8) \cdot 0 + (0.2) \cdot 10 = 2$$

In this case, there is no cost for correctly classifying the email as spam, represented by $(0.8) \cdot 0$. However, when the prediction is incorrect (i.e., classifying non-spam as spam), the cost is 10, expressed by $(0.2) \cdot 10$. Thus, the expected cost of classifying the email as spam is 2.

As a result, we should classify this email as non-spam since the expected cost is lower.

(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 derive the value of θ that works for the costs specified in Table 1, we need to represent p in terms of associated costs.

For the cost of predicting $\hat{y} =$ non-spam:
$Cost_{\hat{y}=\text{non-spam}} = p \cdot 1 + (1-p) \cdot 0 = p$

For the cost of predicting $\hat{y} =$ spam:
$Cost_{\hat{y}=\text{spam}} = p \cdot 0 + (1-p) \cdot 10 = 10(1-p)$

As shown above, the expected cost of predicting non-spam is lower than that of predicting spam. Therefore, we set the inequality:

$p > 10(1-p)$
$\implies p > 10 - 10p$
$\implies 11p > 10$
$\implies p > \frac{10}{11} \approx 0.909$

Therefore, the threshold θ is approximately 0.909.

(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 $\theta = 1/5$.]

To find the new cost table, we use the relationship between the costs and θ derived earlier. We adjust the misclassification costs to achieve $\theta = \frac{1}{5}$.

The expected cost of each case with the new cost values is expressed as $C_{\text{non-spam}|\text{spam}}$ and $C_{\text{spam}|\text{non-spam}}$.

For $\hat{y} =$ non-spam:
$Cost_{\hat{y}=\text{non-spam}} = (1-p) \cdot 0 + p \cdot C_{\text{non-spam}|\text{spam}} = p \cdot C_{\text{non-spam}|\text{spam}}$

For $\hat{y} = spam$:
$Cost_{\hat{y}=spam} = (1-p) \cdot C_{spam|non\text{-}spam} + p \cdot 0 = (1-p) \cdot C_{spam|non\text{-}spam}$

As before, the expected cost of classifying an email as non-spam is lower. Thus, we have:
$p \cdot C_{non\text{-}spam|spam} > (1-p) \cdot C_{spam|non\text{-}spam}$

Simplifying:
$p \cdot C_{non\text{-}spam|spam} > C_{spam|non\text{-}spam} - p \cdot C_{spam|non\text{-}spam}$
$p \cdot (C_{non\text{-}spam|spam} + C_{spam|non\text{-}spam}) > C_{spam|non\text{-}spam}$
$p > \frac{C_{spam|non\text{-}spam}}{C_{non\text{-}spam|spam}+C_{spam|non\text{-}spam}}$

Given $\theta = \frac{1}{5}$, we get:
$\frac{1}{5} = \frac{C_{spam|non\text{-}spam}}{C_{non\text{-}spam|spam}+C_{spam|non\text{-}spam}}$

Therefore, the new costs are:
$C_{spam|non\text{-}spam} = 1, \quad C_{non\text{-}spam|spam} = 4$

The updated cost table is:

Prediction\True Label	*Spam*	*Non-Spam*
Spam	0	4
Non-Spam	1	0

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?
Using Bayes' Theorem:

$p(\theta|X_1, ..., X_n) = p(X_1, ..., X_n|\theta)p(\theta)$
,where $p(X_1, ..., X_n|\theta)$ is the likelihood function $L(\theta)$. $p(\theta)$ is the prior distribution, which is Beta(α, β).

1.Likelihood Function:
For a Bernoulli distribution with n independent and identically distributed (i.i.d.) samples, the likelihood is the product of the individual probabilities. Suppose we observe k heads (i.e., $X_i = 1$) and $n - k$ tails (i.e., $X_i = 0$), then the likelihood is:

$L(\theta) = p(X_1, ..., X_n|\theta) = \theta^k(1-\theta)^{n-k}$

2.Prior Distribution:
The prior distribution is $\theta \sim$ Beta(α, β), whose probability density function (PDF) is:

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

3. Posterior Distribution:
Using Bayes' theorem, the unnormalized posterior is:

$p(\theta|X_1, ..., X_n) = L(\theta)p(\theta) = \theta^k(1-\theta)^{n-k} \cdot \theta^{\alpha-1}(1-\theta)^{\beta-1} = \theta^{k+\alpha-1}(1-\theta)^{n-k+\beta-1}$

Beta distribution with updated parameters:
$\theta|X_1, ..., X_n \sim Beta(k+\alpha, n-k+\beta) = \frac{\theta^{k+\alpha-1}(1-\theta)^{n-k+\beta-1}}{B(k+\alpha,n-k+\beta)}$

Therefore updated α and β is:
$\alpha_{posterior} = k + \alpha$
$\beta_{posterior} = n - k + \beta$

(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.

- Prior: Beta$(2, 2)$
- Posterior derived in (a): Beta$(k + \alpha, n - k + \beta)$

1) When observed 5 coin tosses with 2 heads:
- $n = 5$
- $k = 2$
- $n - k = 3$

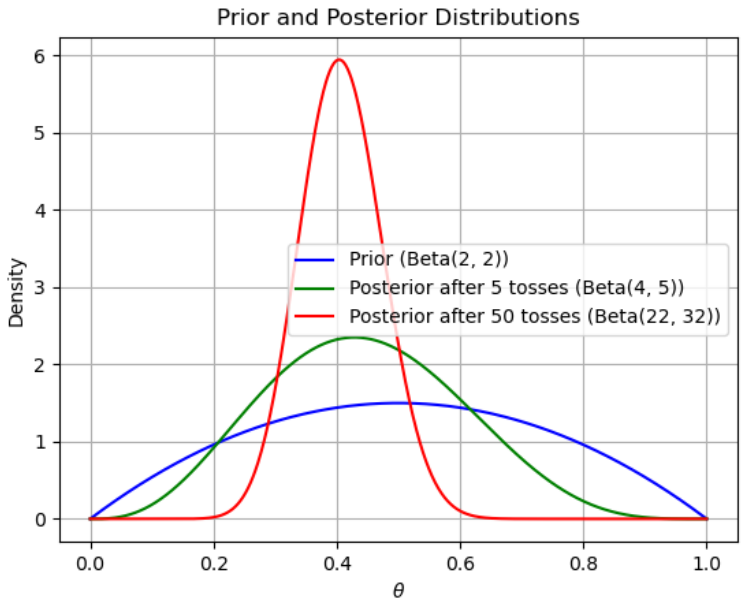
Therefore, the posterior Beta distribution is:
Beta$(k + \alpha, n - k + \beta)$ = Beta$(2 + 2, 3 + 2)$ = Beta$(4, 5)$

2) When observing 50 coin tosses with 20 heads:
- $n = 50$
- $k = 20$
- $n - k = 30$

Therefore, the posterior Beta distribution is:

$Beta(k+\alpha, n-k+\beta) = Beta(20+2, 30+2) = Beta(22, 32)$

(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?

The posterior distribution we derived before:
$\Longrightarrow Beta(s+2, n-s+2)$

Therefore, the MAP estimate
$\Longrightarrow \hat{\theta}_{MAP} = \frac{s+1}{n+2}$

As the number of tosses n increases, the observed number of heads s will converge toward the true probability $\theta = 0.4$.
$\frac{s}{n} \to \theta$ *as* $n \to \infty$ *(By the law of large numbers)*

Thus, we can express this in terms of our MAP estimate:
$\Longrightarrow \hat{\theta}_{MAP} = \frac{s+1}{n+2} \approx \frac{\theta n+1}{n+2}$ *as* $n \to \infty$
$\Longrightarrow \hat{\theta}_{MAP} \approx \frac{0.4n+1}{n+2}$ *(Since s/n approaches 0.4 as $n \to \infty$)*
$\Longrightarrow \lim_{n\to\infty} \hat{\theta}_{MAP} = \lim_{n\to\infty} \frac{0.4n+1}{n+2} = 0.4$

Therefore, as we observe more and more data (i.e., as $n \to \infty$), the shape of the posterior distribution shrinks towards $\theta = 0.4$, and the MAP estimate $\hat{\theta}_{MAP}$ will converge to the true value $\theta = 0.4$. This convergence aligns with the behavior of the Maximum Likelihood Estimate (MLE), which also approaches $\frac{s}{n}$ as n increases, thereby reinforcing our understanding of Bayesian estimation in this context.

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: $\mathbf{w}_0 = 0$.

if $x = 0$: perceptron will misclassify infinitely.(does not converge)

if $(x! = 0)$:

The prediction of perceptron is based on the sign of $w^T x$.
$\implies \hat{y}_0 = sign(w^T x) = sign(0x) = 0$
Since the prediction is 0 and the true label is +1 , it is considered incorrect.

After misclassification perceptron updates weight vector.
$w_1 = w_0 + yx = 0 + (1)x = x$

The prediction of next iteration is now
$\hat{y}_1 = sign(w_1^T x) = sign(x^T x)$
Since $x^T x = ||x||_2{}^2$ is always positive(since $x! = 0$),

$sign(x^T x) > 0$
The prediction is correct ($\hat{y}_1 = +1$), iteration ends.

Therefore, when the initial weight vector is $w_0 = 0$, the Perceptron misclassifies the point once before converging.

(b) (2 pts) Case 2: $\mathbf{w}_0! = 0$:

There are two possibilities:
1)If $w_0^T x > 0$, the prediction is correct ($\hat{y}_0 = +1$) and the algorithm does not update the weights. In this case, the algorithm converges immediately.

2)If $w_0^T x \leq 0$, the prediction is incorrect, and the algorithm updates the weights. If the initial prediction is incorrect, the weight vector is updated as: $w_1 = w_0 + x$

After the first update, the prediction becomes:
$\hat{y}_1 = sign(w_1^T x) = sign((w_0 + x)^T x) = sign(w_0^T x + x^T x)$

if $w_0 + x^T x > 0$, the sign of $w_0^T x + x^T x$ will now be positive and converge.
if $w_0 + x^T x < 0$ is negative value, we can solve at most how many iterations will there be by using convergence theorem.

Covergence Theorem:

Norm of the input vector R: Since we only have one input vector x, $R = \|x\|$

Margin γ: The margin γ is defined as the minimum value of $y\frac{w^* \cdot x}{\|w^*\|}$, where w^* is the optimal weight vector. In this case, since $y = +1$,$\gamma = \frac{w^* \cdot x}{\|w^*\|}$

For a single data point, the optimal weight vector w^* is any vector that satisfies $w^* \cdot x > 0$. The simplest choice is to set $w^* = x$, which gives us:
$\gamma = \frac{x \cdot x}{\|x\|} = \|x\|$

3. Plugging into the convergence bound: Now we can apply the Perceptron Convergence Theorem. For the single point $(x, +1)$, the number of updates M is bounded by: Since $R = \|x\|$ and

$\gamma = \|x\|$, *we have:*

$M \leq \frac{\|x\|^2}{\|x\|^2} = 1$

Therefore, perceptron will mis-classify none or at most once before converging.

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.

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

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

$= \sum_{i=1}^{N} \sum_{k=1}^{K} \log p(y = k|x_i)^{y_{ik}}$

$= \sum_{i=1}^{N} \sum_{k=1}^{K} y_{ik} \log p(y = k|x_i)$

$= \sum_{i=1}^{N} \sum_{k=1}^{K} y_{ik} \log(\frac{\exp(w_k^T x_i)}{\sum_{j=1}^{K} \exp(w_j^T x_i)})$ *(Since softmax function of* $p(y = k|x_i)$ *is* $\frac{\exp(\mathbf{w}_k^T \mathbf{x})}{\sum_{j=1}^{K} \exp(\mathbf{w}_j^T \mathbf{x})}$*)*

$= \sum_{i=1}^{N} \sum_{k=1}^{K} y_{ik} (\log \exp(w_j^T x_i) - \log \sum_{j=1}^{K} \exp(\mathbf{w}_j^T \mathbf{x}))$

$= \sum_{i=1}^{N} \sum_{k=1}^{K} y_{ik} (w_k^T x_i - \log(\sum_{j=1}^{K} \exp(w_j^T x_i)))$

(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.]

$\log L(w) \sum_{i=1}^{N} \sum_{k=1}^{K} y_{ik} (w_k^T x_i - \log(\sum_{j=1}^{K} \exp(w_j^T x_i)))$

$= \sum_{i=1}^{N} \sum_{k=1}^{K} y_{ik} w_k^T x_i - \sum_{i=1}^{N} \sum_{k=1}^{K} y_{ik} \log(\sum_{j=1}^{K} \exp(w_j^T x_i))$

1. Differentiate $\sum_{i=1}^{N} \sum_{k=1}^{K} y_{ik} w_k^T x_i$

$\frac{\partial}{\partial w_c} \sum_{i=1}^{N} \sum_{k=1}^{K} y_{ik} w_k^T x_i = \sum_{i=1}^{N} y_{ic} x_i$
(Since we differentiate this with respect to w_c*, only the terms with* $k = c$ *survive)*

2. Differentiate $\sum_{i=1}^{N}\sum_{k=1}^{K} y_{ik}\log(\sum_{j=1}^{K}\exp(w_j^T x_i))$

Let, $z_i = \sum_{j=1}^{K}\exp(w_j^T x_i)$

$\frac{\partial}{\partial w_c}\sum_{i=1}^{N}\sum_{k=1}^{K} y_{ik}\log(\sum_{j=1}^{K}\exp(w_j^T x_i))$

$= \frac{\partial}{\partial w_c}\sum_{i=1}^{N}\sum_{k=1}^{K} y_{ik}\log z_i$

$= \sum_{i=1}^{N}\frac{\partial}{\partial w_c}\log z_i$ *(Since,* $\sum_{k=1}^{K} y_{ik}$ *is not dependent on w, it is a constant)*

$= \sum_{i=1}^{N}\frac{1}{z_i}\frac{\partial z_i}{\partial w_c}$

$= \sum_{i=1}^{N}\frac{1}{z_i}\exp(w_c^T x_i)x_i$
$(\frac{\partial z_i}{\partial w_c} = \frac{1}{\partial w_c}\sum_{j=1}^{K}\exp(w_j^T x_i) = \exp(w_c^T x_i)x_i$ *by the chain rule and equation only holds when j=c)*

By replacing z_i *to* $\sum_{j=1}^{K}\exp(w_j^T x_i)$*:*

$\sum_{i=1}^{N}\frac{1}{z_i}\exp(w_c^T x_i)x_i = \sum_{i=1}^{N}\frac{\exp(w_c^T x_i)}{\sum_{j=1}^{K}\exp(w_j^T x_i)}x_i = \sum_{i=1}^{N} p(y=c|x_i)x_i$

3. Putting together

$\sum_{i=1}^{N} y_{ic}x_i + \sum_{i=1}^{N}\frac{\exp(w_c^T x_i)}{\sum_{j=1}^{K}\exp(w_j^T x_i)}x_i$

$= \sum_{i=1}^{N}\left(y_{ic} - p(y=c|x_i)\right)x_i.$