CS480/680: Introduction to Machine Learning

Homework 1

Due: 11:59 pm, May 29, 2024, submit on LEARN.

Jiaze Xiao
20933691

Submit your writeup in pdf and all source code in a zip file (with proper documentation). Write a script for each programming exercise so that the TA can easily run and verify your results. Make sure your code runs!
[Text in square brackets are hints that can be ignored.]

Exercise 1: Perceptron (8 pts)

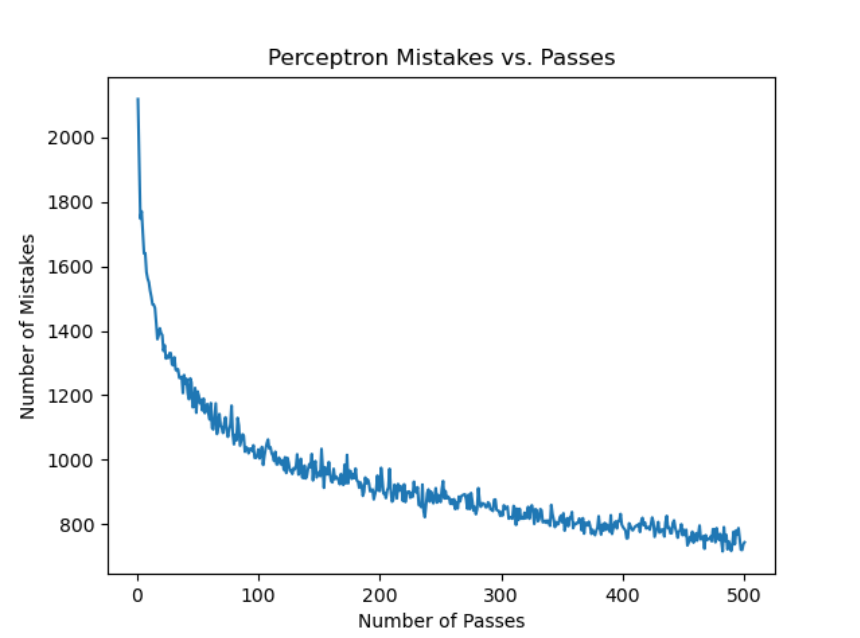
Convention: All algebraic operations, when applied to a vector or matrix, are understood to be element-wise (unless otherwise stated).

Algorithm 1: The perceptron.

Input: $X \in \mathbb{R}^{d \times n}$, $\mathbf{y} \in \{-1, 1\}^n$, $\mathbf{w} = \mathbf{0}_d$, $b = 0$, max_pass $\in \mathbb{N}$
Output: $\mathbf{w}, b, mistake$

```
for t = 1, 2, ..., max_pass do
    mistake(t) ← 0
    for i = 1, 2, ..., n do
        if y_i(⟨x_i, w⟩ + b) ≤ 0 then
            w ← w + y_i x_i                    // x_i is the i-th column of X
            b ← b + y_i
            mistake(t) ← mistake(t) + 1
```

1. (1 pt) Implement the perceptron in Algorithm 1. Your implementation should take input as $X = [\mathbf{x}_1, \ldots, \mathbf{x}_n] \in \mathbb{R}^{d \times n}$, $\mathbf{y} \in \{-1, 1\}^n$, an initialization of the hyperplane parameters $\mathbf{w} \in \mathbb{R}^d$ and $b \in \mathbb{R}$, and the maximum number of passes of the training set [suggested max_pass $= 500$]. Run your perceptron algorithm on the spambase dataset (available on course website), and plot the number of mistakes (y-axis) w.r.t. the number of passes (x-axis).



(run `python3 q1.py` under a1/ex1-perceptron)

2. (1 pt) Using the one-vs-all reduction to implement a multiclass perceptron. You may call your binary implementation. Test your algorithm on the activity dataset (available on course website), and report your final errors on the training and test sets.

   Ans:

   Training Error: 1.89%
   Test Error: 4.92%

   (run `python3 q2.py` under a1/ex1-perceptron)

3. (1 pt) Using the one-vs-one reduction to implement a multiclass perceptron. You may call your binary implementation. Test your algorithm on the activity dataset (available on course website), and report your final errors on the training and test sets.

   Ans:

   Training Error: 1.51%
   Test Error: 5.02%

   (run `python3 q3.py` under a1/ex1-perceptron)

4. (2 pts) Consider the (continuous) piece-wise function

$$f(\mathbf{w}) := \max_k f_k(\mathbf{w}), \tag{1}$$

   where each f_k is continuously differentiable. We define the derivative of f at any $\mathbf{w}$ as follows: first find (any) k such that $f(\mathbf{w}) = f_k(\mathbf{w})$, i.e., $f_k(\mathbf{w})$ achieves the maximum among all pieces; then we let $f'(\mathbf{w}) = f'_k(\mathbf{w})$. [Clearly, the index k that achieves maximum may depend on $\mathbf{w}$, the point we evaluate the derivative at.] Now consider the following problem [padding applied, $y_i \in \{\pm 1\}$]:

$$\min_{\mathbf{w}} \sum_{i=1}^{n} \max\{0, -y_i(\langle \mathbf{x}_i, \mathbf{w}\rangle)\}. \tag{2}$$

   Prove that in each iteration, the (binary) perceptron algorithm essentially picks a term from the above summation, computes the corresponding derivative (say $\mathbf{g}$), and performs a gradient update:

$$\mathbf{w} \leftarrow \mathbf{w} - \mathbf{g}. \tag{3}$$

   [You may ignore the degenerate case when $\langle \mathbf{x}_i, \mathbf{w}\rangle = 0$, and you can use the usual chain rule for our derivative.]

   Ans:

   The objective function to minimize is:

$$\sum_{i=1}^{n} \max\{0, -y_i(\langle \mathbf{x}_i, \mathbf{w}\rangle)\}.$$

   This is a piece-wise function, where each term $\max\{0, -y_i(\langle \mathbf{x}_i, \mathbf{w}\rangle)\}$ is differentiable everywhere except at the point where $-y_i(\langle \mathbf{x}_i, \mathbf{w}\rangle) = 0$ which is ignored. Let's define each piece $f_i(\mathbf{w})$ as follows:

$$f_i(\mathbf{w}) = \max\{0, -y_i(\langle \mathbf{x}_i, \mathbf{w}\rangle)\}.$$

   which can be written as:

$$f_i(\mathbf{w}) = \begin{cases} 0 & \text{if } -y_i(\langle \mathbf{x}_i, \mathbf{w}\rangle) \le 0, \\ -y_i(\langle \mathbf{x}_i, \mathbf{w}\rangle) & \text{if } -y_i(\langle \mathbf{x}_i, \mathbf{w}\rangle) > 0. \end{cases}$$

   Derivative of Each Piece

- Case 1: If $-y_i(\langle \mathbf{x}_i, \mathbf{w}\rangle) \le 0$, then $f_i(\mathbf{w}) = 0$, and the derivative is $\nabla f_i(\mathbf{w}) = 0$.
- Case 2: If $-y_i(\langle \mathbf{x}_i, \mathbf{w}\rangle) > 0$, then $f_i(\mathbf{w}) = -y_i(\langle \mathbf{x}_i, \mathbf{w}\rangle)$, and the derivative is:

$$\begin{aligned} f_i(\mathbf{w}+\mathbf{z}) - f_i(\mathbf{w}) &= -y_i(\langle \mathbf{x}_i, \mathbf{z}\rangle) \\ \nabla f_i(\mathbf{w})(\mathbf{z}) &= -y_i(\langle \mathbf{x}_i, \mathbf{z}\rangle) \\ \nabla f_i(\mathbf{w}) &= -y_i \mathbf{x}_i. \end{aligned}$$

In perceptron algorithm, at each iteration, if $\mathbf{x}_i$ is misclassified by $\mathbf{w}$, the perceptron updates $\mathbf{w}$ as follows:

$$\mathbf{w} \leftarrow \mathbf{w} + y_i \mathbf{x}_i.$$

This can be interpreted as follows: 1. Identify a misclassified point $\mathbf{x}_i$. 2. Compute the gradient of the corresponding term $\max\{0, -y_i(\langle \mathbf{x}_i, \mathbf{w}\rangle)\}$, which is $\nabla f_i(\mathbf{w}) = -y_i \mathbf{x}_i$ if $y_i(\langle \mathbf{x}_i, \mathbf{w}\rangle) \le 0$. 3. Perform the gradient update:

$$\mathbf{w} \leftarrow \mathbf{w} - \nabla f_i(\mathbf{w}) = \mathbf{w} - (-y_i \mathbf{x}_i) = \mathbf{w} + y_i \mathbf{x}_i.$$

Therefore, this shows that the perceptron algorithm is performing a gradient update on the objective function given in Equation (2).

5. (1 pt) Consider the following problem, where $y_i \in \{1, 2, \ldots, c\}$:

$$\min_{\mathbf{w}_1, \ldots, \mathbf{w}_c} \sum_{i=1}^{n} \max_{k=1,\ldots,c} \left[\langle \mathbf{x}_i, \mathbf{w}_k \rangle - \langle \mathbf{x}_i, \mathbf{w}_{y_i} \rangle \right]. \tag{4}$$

Show that when $c = 2$, we reduce to the binary perceptron problem in (2). [Try to identify the weights $\mathbf{w}$, using some transformation.]

Ans:

When $c = 2$, the expression becomes:

$$\min_{\mathbf{w}_1, \mathbf{w}_2} \sum_{i=1}^{n} \max\left\{ \langle \mathbf{x}_i, \mathbf{w}_1 \rangle - \langle \mathbf{x}_i, \mathbf{w}_{y_i} \rangle, \langle \mathbf{x}_i, \mathbf{w}_2 \rangle - \langle \mathbf{x}_i, \mathbf{w}_{y_i} \rangle \right\}.$$

Let $f_i(\mathbf{w}) = \max\left\{ \langle \mathbf{x}_i, \mathbf{w}_1 \rangle - \langle \mathbf{x}_i, \mathbf{w}_{y_i} \rangle, \langle \mathbf{x}_i, \mathbf{w}_2 \rangle - \langle \mathbf{x}_i, \mathbf{w}_{y_i} \rangle \right\}$ which can be simplified as:

$$f_i(\mathbf{w}) = \begin{cases} \max\left\{0, \langle \mathbf{x}_i, \mathbf{w}_2 \rangle - \langle \mathbf{x}_i, \mathbf{w}_1 \rangle\right\} & \text{if } y_i = 1, \\ \max\left\{0, \langle \mathbf{x}_i, \mathbf{w}_1 \rangle - \langle \mathbf{x}_i, \mathbf{w}_2 \rangle\right\} & \text{if } y_i = 2. \end{cases}$$

To transform this into the binary perceptron problem, define:

$$\mathbf{w} = \mathbf{w}_1 - \mathbf{w}_2.$$

If $y_i = 1$, we have:

$$\max\left\{0, \langle \mathbf{x}_i, \mathbf{w}_2 \rangle - \langle \mathbf{x}_i, \mathbf{w}_1 \rangle\right\} = \max\left\{0, -\langle \mathbf{x}_i, \mathbf{w} \rangle\right\}.$$

If $y_i = 2$, we have:

$$\max\left\{0, \langle \mathbf{x}_i, \mathbf{w}_1 \rangle - \langle \mathbf{x}_i, \mathbf{w}_2 \rangle\right\} = \max\left\{0, \langle \mathbf{x}_i, \mathbf{w} \rangle\right\}.$$

Combining the above:

$$f_i(\mathbf{w}) = \max\left\{0, -\overline{y_i} \langle \mathbf{x}_i, \mathbf{w} \rangle\right\},$$

where $\overline{y_i} \in \{\pm 1\}$ with the following encoding:

- $\overline{y_i} = 1$ if the original $y_i = 1$.
- $\overline{y_i} = -1$ if the original $y_i = 2$.

Thus, we reduce the multiclass problem for $c = 2$ to the binary perceptron problem:

$$\min_{\mathbf{w}} \sum_{i=1}^{n} \max\{0, -\overline{y}_i \langle \mathbf{x}_i, \mathbf{w} \rangle\}.$$

6. (2 pts) Based on the analogy to the binary case, develop and implement a multiclass perceptron algorithm to solve (4) directly. Run your implementation on the activity dataset (available on course website) and report the final errors on the training and test sets. [Hint: obviously, we want to predict as follows: $\hat{y} = \operatorname{argmax}_{k=1,\dots,c} \langle \mathbf{x}, \mathbf{w}_k \rangle$, i.e., the class k whose corresponding $\mathbf{w}_k$ maximizes the inner product. Explain your algorithm (e.g., through pseudo-code).]

Ans:

Algorithm 2: Multiclass Perceptron

Input: $X \in \mathbb{R}^{n \times d}$, $\mathbf{y} \in \{1, 2, \dots, c\}^n$, $\mathsf{max\_pass} \in \mathbb{N}$
Output: $\{\mathbf{w}_k\}_{k=1}^c$

for $k = 1, 2, \dots, c$ **do**
  $\mathbf{w}_k \leftarrow \mathbf{0}_{d+1}$
for $i = 1, 2, \dots, n$ **do**
  $X_i.append(1)$
for $t = 1, 2, \dots, \mathsf{max\_pass}$ **do**
  for $i = 1, 2, \dots, n$ **do**
    $\hat{y} = \operatorname{argmax}_{k=1,\dots,c} \langle \mathbf{x}, \mathbf{w}_k \rangle$
    if $\hat{y} \neq y_i$ **then**
      $\mathbf{w}_{y_i} \leftarrow \mathbf{w}_{y_i} + \mathbf{x}_i$
      $\mathbf{w}_{\hat{y}} \leftarrow \mathbf{w}_{\hat{y}} - \mathbf{x}_i$

Training Error: 1.71%
Test Error: 4.48%

(run `python3 q6.py` under a1/ex1-perceptron)

Exercise 2: Generalized linear models (6 pts)

Recall that in logistic regression we assumed the *binary* label $\mathsf{Y}_i \in \{0,1\}$ follows the Bernoulli distribution: $\Pr(\mathsf{Y}_i = 1|\mathsf{X}_i) = p_i$, where p_i also happens to be the mean. Under the independence assumption we derived the (conditional) negative log-likelihood function:

$$-\sum_{i=1}^{n}(1-y_i)\log(1-p_i) + y_i \log(p_i). \tag{5}$$

Then, we parameterized the mean parameter p_i through the logit transform:

$$\log\frac{p_i}{1-p_i} = \langle \mathbf{x}_i, \mathbf{w}\rangle + b, \quad \text{or equivalently} \quad p_i = \frac{1}{1+\exp(-\langle \mathbf{x}_i, \mathbf{w}\rangle - b)}. \tag{6}$$

Lastly, we found the weight vector $\mathbf{w}$ and b by minimizing the negative log-likelihood function.

In the following we generalize the above idea significantly. Let the (conditional) density of Y (given $\mathsf{X} = \mathbf{x}$) be

$$p(y|\mathbf{x}) = \exp\Big[\mu(\mathbf{x})\cdot y - \lambda(\mathbf{x})\Big]\cdot q(y), \tag{7}$$

where $\mu : \mathbb{R}^d \to \mathbb{R}$ is a function of $\mathbf{x}$ and $\lambda(\mathbf{x}) = \log\int_y \exp\big(\mu(\mathbf{x})\cdot y\big)q(y)\mathrm{d}y$ so that $p(y|\mathbf{x})$ is properly normalized wrt y (i.e., integrate to 1). For discrete y (such as in logistic regression), replace the density with the probability mass function and the integral with sum.

As always, you need to supply sufficient derivation details to justify your final answer.

1. (1 pt) Given a dataset $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$, derive the (conditional) negative log-likelihood function of $y_1, \ldots, y_n$, assuming independence and the density form in (7).

   Ans:

$$\begin{aligned}\Pr(\mathsf{Y}_1 = y_1, \ldots, \mathsf{Y}_n = y_n|\mathsf{X}_1 = \mathbf{x}_1, \ldots, \mathsf{X}_n = \mathbf{x}_n) &= \prod_{i=1}^n p(y_i|\mathbf{x}_i)\\ &= \prod_{i=1}^n \exp\Big[\mu(\mathbf{x}_i)\cdot y_i - \lambda(\mathbf{x}_i)\Big]\cdot q(y_i)\end{aligned}$$

   Taking the negative log-likelihood, we obtain:

$$\begin{aligned}&-\log\left(\prod_{i=1}^n \exp\Big[\mu(\mathbf{x}_i)\cdot y_i - \lambda(\mathbf{x}_i)\Big]\cdot q(y_i)\right)\\ =&-\sum_{i=1}^n \log\left(\exp\Big[\mu(\mathbf{x}_i)\cdot y_i - \lambda(\mathbf{x}_i)\Big]\cdot q(y_i)\right)\\ =&-\sum_{i=1}^n \left(\log\left(\exp\Big[\mu(\mathbf{x}_i)\cdot y_i - \lambda(\mathbf{x}_i)\Big]\right) + \log(q(y_i))\right)\\ =&-\sum_{i=1}^n \big(\mu(\mathbf{x}_i)\cdot y_i - \lambda(\mathbf{x}_i) + \log(q(y_i))\big)\end{aligned}$$

   Therefore, the (conditional) negative log-likelihood function of $y_1, \ldots, y_n$ given $\mathbf{x}_1, \ldots, \mathbf{x}_n$ is:

$$\boxed{\sum_{i=1}^n(-\mu(\mathbf{x}_i)\cdot y_i + \lambda(\mathbf{x}_i) - \log(q(y_i)))} \tag{8}$$

2. (1 pt) Plug the usual linear parameterization

$$\mu(\mathbf{x}) = \langle \mathbf{x}, \mathbf{w} \rangle + b = \langle \mathbf{x}, \mathbf{w} \rangle \tag{9}$$

into your (conditional) negative log-likelihood and compute the gradient of the resulting function. [Hint: you may swap differentiation with integral and your gradient may involve implicitly defined terms.]

Ans:

Plug the linear parameterization $\mu(\mathbf{x}) = \langle \mathbf{x}, \mathbf{w} \rangle + b = \langle \mathbf{x}, \mathbf{w} \rangle$ into the negative log-likelihood function:

$$\boxed{\ell_n(\mathbf{w}) = \sum_{i=1}^{n} -\langle \mathbf{x}_i, \mathbf{w} \rangle \cdot y_i + \lambda(\mathbf{x}_i) - \log(q(y_i))} \tag{10}$$

First, consider the term $-\langle \mathbf{x}_i, \mathbf{w} \rangle \cdot y_i$:

$$\nabla_{\mathbf{w}} \left(-\langle \mathbf{x}_i, \mathbf{w} \rangle \cdot y_i \right) = -y_i \mathbf{x}_i.$$

Next, we need to differentiate $\lambda(\mathbf{x}_i)$.

$$\lambda(\mathbf{x}_i) = \log \int_y \exp\big(\langle \mathbf{x}_i, \mathbf{w} \rangle \cdot y\big) q(y) \mathrm{d}y.$$

By the Leibniz integral rule, the gradient of $\lambda(\mathbf{x}_i)$ with respect to $\mathbf{w}$ is:

$$\begin{aligned}
\nabla_{\mathbf{w}} \lambda(\mathbf{x}_i) =& \frac{\int_y \frac{\partial}{\partial \mathbf{w}} \exp\big(\langle \mathbf{x}_i, \mathbf{w} \rangle \cdot y\big) q(y) \mathrm{d}y}{\int_y \exp\big(\langle \mathbf{x}_i, \mathbf{w} \rangle \cdot y\big) q(y) \mathrm{d}y} \\
=& \frac{\int_y \exp\big(\langle \mathbf{x}_i, \mathbf{w} \rangle \cdot y\big) y \mathbf{x}_i q(y) \mathrm{d}y}{\int_y \exp\big(\langle \mathbf{x}_i, \mathbf{w} \rangle \cdot y\big) q(y) \mathrm{d}y} \\
=& \frac{\int_y \exp\big(\langle \mathbf{x}_i, \mathbf{w} \rangle \cdot y\big) y \mathbf{x}_i q(y) \mathrm{d}y}{\exp(\log \int_y \exp\big(\langle \mathbf{x}_i, \mathbf{w} \rangle \cdot y\big) q(y) \mathrm{d}y)} \\
=& \int_y \frac{\exp\big(\langle \mathbf{x}_i, \mathbf{w} \rangle \cdot y\big) y \mathbf{x}_i q(y)}{\exp(\lambda(\mathbf{x}_i))} \mathrm{d}y \\
=& \mathbf{x}_i \int_y y p(y|\mathbf{x}_i) \mathrm{d}y \\
=& \mathbf{x}_i \mathbb{E}[y|\mathbf{x}_i]
\end{aligned}$$

Putting it all together, the gradient of $\ell_n(\mathbf{w})$ is:

$$\boxed{\nabla \ell_n(\mathbf{w}) = \sum_{i=1}^{n} (\mathbb{E}[y|\mathbf{x}_i] - y_i) \mathbf{x}_i} \tag{11}$$

3. (1 pt) Let us revisit linear regression, where

$$p(y|\mathbf{x}) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(y-\nu(\mathbf{x}))^2}{2}\right) \tag{12}$$

Identify the functions $\mu(\mathbf{x})$, $\lambda(\mathbf{x})$ and $q(y)$ for the above specialization. Based on the linear parameterization in Ex 2.2, derive the negative log-likelihood and gradient. [Hint: you may simply plug into the more general result in Ex 2.2. Compare with what you already learned about linear regression to make sure both Ex 2.2 and Ex 2.3 are correct.]

Ans:

The given density can be rewritten in exponential form:

$$\begin{aligned} p(y|\mathbf{x}) &= \exp\left(-\frac{(y-\nu(\mathbf{x}))^2}{2} - \log\sqrt{2\pi}\right) \\ &= \exp\left(y\nu(\mathbf{x}) - \frac{\nu(\mathbf{x})^2}{2} - \frac{y^2}{2} - \log\sqrt{2\pi}\right) \end{aligned}$$

According to (7), we identify:

$$\boxed{\mu(\mathbf{x}) = \nu(\mathbf{x})} \tag{13}$$

The term $\lambda(\mathbf{x})$ depends on $\mathbf{x}$:

$$\boxed{\lambda(\mathbf{x}) = \frac{\nu(\mathbf{x})^2}{2}} \tag{14}$$

The remaining terms that are independent of $\mathbf{x}$ fall into $q(y)$:

$$\boxed{q(y) = \exp\left(-\frac{y^2}{2} - \log\sqrt{2\pi}\right) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{y^2}{2}\right)} \tag{15}$$

Negative Log-likelihood:

$$\begin{aligned} \ell_n(\mathbf{w}) &= -\sum_{i=1}^n \log p(y_i|\mathbf{x}_i) \\ &= -\sum_{i=1}^n \log\left(\frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(y_i-\nu(\mathbf{x}_i))^2}{2}\right)\right) \\ &= -\sum_{i=1}^n \left(-\frac{(y_i-\nu(\mathbf{x}_i))^2}{2} - \log\sqrt{2\pi}\right) \end{aligned}$$

For the linear parameterization $\nu(\mathbf{x}_i) = \langle \mathbf{x}_i, \mathbf{w}\rangle$, the negative log-likelihood becomes:

$$\boxed{\ell_n(\mathbf{w}) = \sum_{i=1}^n \left(\frac{(y_i - \langle \mathbf{x}_i, \mathbf{w}\rangle)^2}{2} + \log\sqrt{2\pi}\right)} \tag{16}$$

Thus,

$$\begin{aligned} \nabla \ell_n(\mathbf{w}) &= \frac{\partial}{\partial \mathbf{w}} \sum_{i=1}^n \left(\frac{(y_i - \langle \mathbf{x}_i, \mathbf{w}\rangle)^2}{2} + \log\sqrt{2\pi}\right) \\ &= \sum_{i=1}^n (y_i - \langle \mathbf{x}_i, \mathbf{w}\rangle) \cdot (-\mathbf{x}_i) \end{aligned}$$

Therefore:

$$\boxed{\nabla \ell_n(\mathbf{w}) = \sum_{i=1}^{n} (\langle \mathbf{x}_i, \mathbf{w} \rangle - y_i)\mathbf{x}_i} \tag{17}$$

4. (1 pt) Let us revisit logistic regression, where

$$\Pr(\mathsf{Y} = y|\mathbf{x}) = [\nu(\mathbf{x})]^y [1 - \nu(\mathbf{x})]^{1-y}, \quad \text{where} \quad y \in \{0, 1\}. \tag{18}$$

Identify the functions $\mu(\mathbf{x})$, $\lambda(\mathbf{x})$ and $q(y)$ for the above specialization. Based on the linear parameterization in Ex 2.2, derive the negative log-likelihood and gradient. [Hint: Compare with what you already learned about logistic regression.]

Ans:

The given probability can be rewritten as:

$$\begin{aligned} p(y|\mathbf{x}) &= \exp\left[y \log \nu(\mathbf{x}) + (1-y)\log(1-\nu(\mathbf{x}))\right] \\ &= \exp\left[y \frac{\nu(\mathbf{x})}{1-\nu(\mathbf{x})} + \log(1-\nu(\mathbf{x}))\right] \end{aligned}$$

According to (7), we identify:

$$\boxed{\mu(\mathbf{x}) = \log\left(\frac{\nu(\mathbf{x})}{1-\nu(\mathbf{x})}\right)} \tag{19}$$

$$\begin{aligned} \Rightarrow \nu(\mathbf{x}) &= \frac{1}{1+\exp(-\mu(\mathbf{x}))} \\ &= \frac{1}{1+\exp(-\langle \mathbf{x}, \mathbf{w} \rangle)} \end{aligned}$$

The term $\lambda(\mathbf{x})$ depends on $\mathbf{x}$:

$$\boxed{\lambda(\mathbf{x}) = -\log(1-\nu(\mathbf{x}))} \tag{20}$$

The remaining terms that are independent of $\mathbf{x}$ fall into $q(y)$:

$$\boxed{q(y) = 1} \tag{21}$$

Negative Log-Likelihood:

$$\begin{aligned} \ell_n(\mathbf{w}) &= -\sum_{i=1}^{n} \log p(y_i|\mathbf{x}_i) \\ &= -\sum_{i=1}^{n} \left[y_i \log \nu(\mathbf{x}_i) + (1-y_i)\log(1-\nu(\mathbf{x}_i))\right] \\ &= -\sum_{i=1}^{n} \left[y_i \log \frac{1}{1+\exp(-\langle \mathbf{x}_i, \mathbf{w} \rangle)} + (1-y_i)\log(1 - \frac{1}{1+\exp(-\langle \mathbf{x}_i, \mathbf{w} \rangle)})\right] \end{aligned}$$

Therefore,

$$\boxed{\ell_n(\mathbf{w}) = \sum_{i=1}^{n} \log(1+\exp(-\langle \mathbf{x}_i, \mathbf{w} \rangle)) + (1-y_i)\langle \mathbf{x}_i, \mathbf{w} \rangle} \tag{22}$$

Gradient:

$$\begin{aligned}\nabla \ell_n(\mathbf{w}) =& \frac{\partial}{\partial \mathbf{w}} \sum_{i=1}^n \log(1+\exp(-\langle \mathbf{x}_i, \mathbf{w}\rangle)) + (1-y_i)\langle \mathbf{x}_i, \mathbf{w}\rangle \\ =& \sum_{i=1}^n \frac{-\mathbf{x}_i \exp(-\langle \mathbf{x}_i, \mathbf{w}\rangle)}{1+\exp(-\langle \mathbf{x}_i, \mathbf{w}\rangle)} + \mathbf{x}_i(1-y_i) \\ =& \sum_{i=1}^n -\mathbf{x}_i \frac{1+\exp(-\langle \mathbf{x}_i, \mathbf{w}\rangle) - 1}{1+\exp(-\langle \mathbf{x}_i, \mathbf{w}\rangle)} + \mathbf{x}_i(1-y_i) \\ =& \sum_{i=1}^n -\mathbf{x}_i \left(1 - \frac{1}{1+\exp(-\langle \mathbf{x}_i, \mathbf{w}\rangle)}\right) + \mathbf{x}_i(1-y_i) \\ =& \sum_{i=1}^n \mathbf{x}_i \left(\frac{1}{1+\exp(-\langle \mathbf{x}_i, \mathbf{w}\rangle)} - y_i\right)\end{aligned}$$

Thus,

$$\boxed{\nabla \ell_n(\mathbf{w}) = \sum_{i=1}^n \left(\frac{1}{1+\exp(-\langle \mathbf{x}_i, \mathbf{w}\rangle)} - y_i\right)\mathbf{x}_i} \tag{23}$$

5. (2 pts) Now let us tackle something new. Let

$$\Pr(\mathsf{Y}=y|\mathbf{x}) = \frac{[\nu(\mathbf{x})]^y}{y!}\exp(-\nu(\mathbf{x})), \quad \text{where} \quad y=0,1,2,\ldots. \tag{24}$$

Identify the functions $\mu(\mathbf{x})$, $\lambda(\mathbf{x})$ and $q(y)$ for the above specialization. Based on the linear parameterization in Ex 2.2, derive the negative log-likelihood and gradient. [Hint: Y here follows the Poisson distribution, which is useful for modeling integer-valued events, e.g., the number of customers at a given time.]

Ans:

The given probability can be rewritten as:

$$\Pr(\mathsf{Y}=y|\mathbf{x}) = \exp\left[y\log\nu(\mathbf{x}) - \nu(\mathbf{x}) - \log y!\right].$$

According to (7), we identify:

$$\boxed{\mu(\mathbf{x}) = \log\nu(\mathbf{x})} \tag{25}$$

The term $\lambda(\mathbf{x})$ is:

$$\boxed{\lambda(\mathbf{x}) = \nu(\mathbf{x})} \tag{26}$$

The remaining terms that are independent of $\mathbf{x}$ fall into $q(y)$:

$$\boxed{q(y) = \frac{1}{y!}} \tag{27}$$

Negative Log-Likelihood:

$$\begin{aligned}\ell_n(\mathbf{w}) =& -\sum_{i=1}^n \log p(y_i|\mathbf{x}_i) \\ =& -\sum_{i=1}^n \log\left(\frac{[\nu(\mathbf{x}_i)]^y}{y!}\exp(-\nu(\mathbf{x}_i))\right) \\ =& -\sum_{i=1}^n \left[y_i\log\nu(\mathbf{x}_i) - \nu(\mathbf{x}_i) - \log y_i!\right]\end{aligned}$$

For the linear parameterization $\nu(\mathbf{x}) = \exp(\mu(\mathbf{x})) = \exp(\langle \mathbf{x}, \mathbf{w} \rangle)$, this becomes:

$$\boxed{\ell_n(\mathbf{w}) = -\sum_{i=1}^{n} [y_i \langle \mathbf{x}_i, \mathbf{w} \rangle - \exp(\langle \mathbf{x}_i, \mathbf{w} \rangle) - \log y_i!]} \tag{28}$$

Gradient:

$$\boxed{\nabla \ell_n(\mathbf{w}) = \sum_{i=1}^{n} (\exp(\langle \mathbf{x}_i, \mathbf{w} \rangle) - y_i) \mathbf{x}_i} \tag{29}$$

Exercise 3: Ordinal regression (4 pts)

In many applications, the "labels" have an inherent order. For example, the letter grade A is preferred to B, which is preferred to C, etc. More generally, consider c ordinal labels $1, 2, \ldots, c$, where we prefer label k than $k+1$, for each $k = 1, \ldots, c-1$. [The preference is transitive, i.e., any "smaller" label is preferred over a "larger" label.]

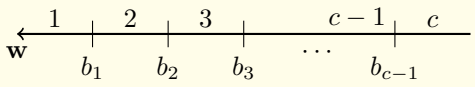
1. (2 pts) Let us consider $c-1$ *parallel* hyperplanes $H_k := \{\mathbf{x} : \langle \mathbf{x}, \mathbf{w} \rangle + b_k = 0\}$, which partition our space into c rectangular regions. We define our prediction as

$$\hat{y} \leq k \iff \langle \mathbf{x}, \mathbf{w} \rangle + b_k > 0, \tag{30}$$

or more explicitly,

$$\hat{y} = k \iff [\langle \mathbf{x}, \mathbf{w} \rangle + b_k > 0 \text{ and } \langle \mathbf{x}, \mathbf{w} \rangle + b_{k-1} \leq 0], \tag{31}$$

where $b_0 := -\infty$ and $b_c := \infty$.



The ordering in the labels is now respected, if we constrain $b_1 \leq b_2 \leq \cdots \leq b_{c-1}$:

$$\hat{y} \leq k \implies \hat{y} \leq l, \quad \forall\, l \geq k. \tag{32}$$

We learn the weights $\mathbf{w}$ and $b_1, \ldots, b_{c-1}$ by reducing to a sequence of (coupled) binary classifications:

$$\min_{\mathbf{w}, b_1 \leq b_2 \leq \cdots \leq b_{c-1}} \frac{\lambda}{2}\|\mathbf{w}\|_2^2 + \sum_{k=1}^{c-1}\sum_{i=1}^{n} \max\{0, 1 - ([\![y_i = k]\!] - [\![y_i = k+1]\!])(\langle \mathbf{x}_i, \mathbf{w} \rangle + b_k)\}, \tag{33}$$

where recall that $[\![A]\!]$ is 1 if A is true and 0 otherwise. It is clear that when $c = 2$, the above reduces to the familiar soft-margin SVM. Derive the Lagrangian dual of (33). [If it helps, you may ignore the constraint $b_1 \leq \ldots \leq b_{c-1}$.]

Ans:

$$\max\{0, 1 - ([\![y_i = k]\!] - [\![y_i = k+1]\!])(\langle \mathbf{x}_i, \mathbf{w} \rangle + b_k)\} = \max_{0 \leq \alpha \leq 1} \alpha(1 - ([\![y_i = k]\!] - [\![y_i = k+1]\!])(\langle \mathbf{x}_i, \mathbf{w} \rangle + b_k))$$

$$b_1 \leq b_2 \leq \cdots \leq b_{c-1} = b_i \leq b_{i+1} \quad \forall i \in \{1, \ldots, c-2\}$$

The Lagrangian Dual of the primal problem is:

$$\max_{0 \leq \alpha \leq 1,\ \beta \geq 0} \min_{\mathbf{w}, b} \frac{\lambda}{2}\|\mathbf{w}\|_2^2 + \sum_{k=1}^{c-1}\sum_{i=1}^{n} \alpha_{ik}\left[1 - ([\![y_i = k]\!] - [\![y_i = k+1]\!])(\langle \mathbf{x}_i, \mathbf{w} \rangle + b_k)\right] + \sum_{k=1}^{c-2} \beta_k (b_k - b_{k+1})$$

Solving inner unconstrained problem by setting derivative to 0:

For $\mathbf{w}$:

$$\frac{\partial}{\partial \mathbf{w}} = \lambda \mathbf{w} - \sum_{k=1}^{c-1}\sum_{i=1}^{n} \alpha_{ik}([\![y_i = k]\!] - [\![y_i = k+1]\!])\mathbf{x}_i = 0$$

$$\mathbf{w} = \frac{1}{\lambda}\sum_{k=1}^{c-1}\sum_{i=1}^{n} \alpha_{ik}([\![y_i = k]\!] - [\![y_i = k+1]\!])\mathbf{x}_i$$

For each b_k:

$$\frac{\partial}{\partial b_k} = -\sum_{i=1}^{n} \alpha_{ik}([\![y_i = k]\!] - [\![y_i = k+1]\!]) + \beta_k - \beta_{k-1} = 0, \quad \beta_0 = \beta_{c-1} = 0$$

$$\beta_k - \beta_{k-1} = \sum_{i=1}^{n} \alpha_{ik}([\![y_i = k]\!] - [\![y_i = k+1]\!])$$

Since

$$\begin{aligned}\sum_{k=1}^{c-2} \beta_k(b_k - b_{k+1}) &= \sum_{k=1}^{c-2} \beta_k b_k - \sum_{k=1}^{c-2} \beta_k b_{k+1} \\ &= \sum_{k=1}^{c-2} \beta_k b_k - \sum_{k=2}^{c-1} \beta_{k-1} b_k \\ &= \sum_{k=1}^{c-1} \beta_k b_k - \sum_{k=1}^{c-1} \beta_{k-1} b_k \\ &= \sum_{k=1}^{c-1} b_k(\beta_k - \beta_{k-1}),\end{aligned}$$

we can plug in $\mathbf{w}$ and $\beta_k - \beta_{k-1}$ back to the Lagrangian Dual and simplify:

$$\max_{0 \le \alpha \le 1} \; -\frac{1}{2\lambda} \left\| \sum_{k=1}^{c-1} \sum_{i=1}^{n} \alpha_{ik}([\![y_i = k]\!] - [\![y_i = k+1]\!]) \mathbf{x}_i \right\|_2^2 + \sum_{k=1}^{c-1} \sum_{i=1}^{n} \alpha_{ik}.$$

Change to minimization:

$$\boxed{\min_{0 \le \alpha \le 1} \; \frac{1}{2\lambda} \left\| \sum_{k=1}^{c-1} \sum_{i=1}^{n} \alpha_{ik} \mathbf{x}_i([\![y_i = k]\!] - [\![y_i = k+1]\!]) \right\|_2^2 - \sum_{k=1}^{c-1} \sum_{i=1}^{n} \alpha_{ik}}$$

which is the Lagrangian dual of (33) as required.

2. (2 pts) In the previous formulation, to learn b_k, essentially we take class k as positive and class $k+1$ as negative. Can you find a "better" alternative? Write down the formulation. [Hint: it would be similar to (33).]

   Ans:

$$\boxed{\min_{\mathbf{w}, b_1 \le b_2 \le \cdots \le b_{c-1}} \; \frac{\lambda}{2}\|\mathbf{w}\|^2 + \sum_{k=1}^{c-1} \sum_{i=1}^{n} \max\left\{0, 1 - \operatorname{sign}(k - y_i)\left(\langle \mathbf{x}_i, \mathbf{w} \rangle + b_k\right)\right\}}$$

   Here, $\operatorname{sign}(k - y_i) = 1$ if $y_i \le k$, 0 otherwise.

   By using $\operatorname{sign}(k - y_i)$, each classifier $\langle \mathbf{x}_i, \mathbf{w} \rangle + b_k$ is not just distinguishing between two consecutive classes but is effectively assessing whether the instance i falls below or above the ordinal threshold k. This adjustment means that each boundary b_k isn't just learning to separate k from $k+1$ but is learning to separate $1, 2, \ldots, k$ from $k+1, \ldots, c$.