Statistical Deep Learning Homework 1

工工所碩一 王泓達 R13546017

1. Implementation of the Perceptron Learning Algorithm.

   a) Given a dataset $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ where $x_i \in \mathbb{R}^d$ and $y_i \in \{-1, 1\}$, implement the Perceptron Learning Algorithm: Given $(x_1, y_1), \ldots, (x_n, y_n)$, initialize $w = 0$ and repeat the following steps for M iterations:
   - If $y_i(w \cdot x_i) \leq 0$, update $w \leftarrow w + y_i x_i$.
   - If $y_i(w \cdot x_i) > 0$, do not update.

   b) Use the Iris Dataset to test your algorithm. You can use the first two classes of the Iris Dataset.

```
from sklearn import datasets

iris = datasets.load_iris()
```

   c) Draw a scatterplot for the dataset and plot the decision boundary of the results from (b).

   d) Compare with logistic regression.

```
# x: 100 * 3
y = iris["target"].reshape((100,1)) # y: 100 * 1
w = np.zeros((3,1)) # 3*1
false_flag = 1
iteration_counter = 0

while false_flag:
    i = 0
    while i < len(x):
        if(y[i][0]*(w.reshape((1,3)).dot(x[i]))<=0):
            w += y[i][0]*x[i].reshape((3,1))
            i = len(x)+100
        else:
            if(i==len(x)-1):
                false_flag = 0
            i+=1
    iteration_counter += 1
```

✓ 0.1s

a) I use Python to implement Perceptron Learning Algorithm, my code is presented below.

Code Explaination

x: The matrix of data, the first column stands for 1 to indicate intercept. The second and the third column are the features of the data.

y: The vector of target, we use 1 or -1 to represent the class of iris.

w: The weight of features, he first row is 0 to indicate the weight of intercept. The second and the third row are the weight of the sepal length and sepal width.

false flag: Its initial value is 1, and it will be changed into 0 when $y_i(w \cdot x_i) > 0$ for all i. (i means the row of x)

The code will keep running until false flag is 0. Otherwise, it will update w in every iteration.

b) Here is the code of loading iris data.

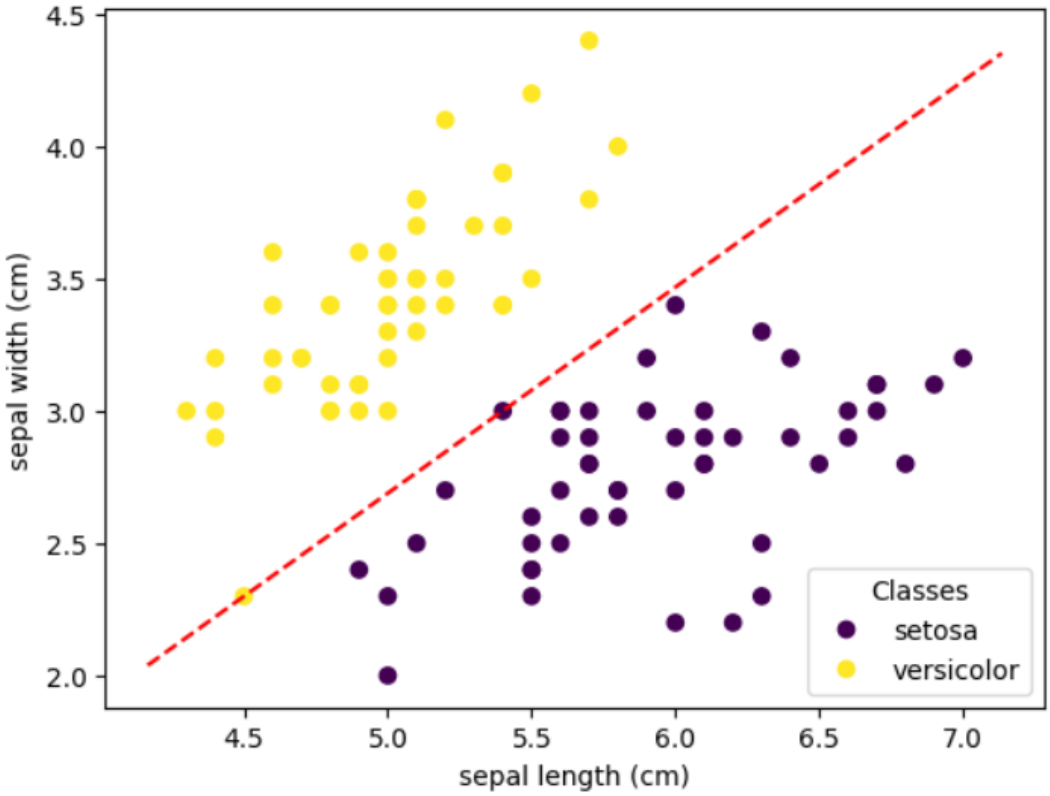
```
from sklearn import datasets
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np

iris = datasets.load_iris()

for i in range(len(iris["data"])):
    if(iris["target"][i]==2):
        iris["data"] = iris["data"][:i, :2]
        iris["target"] = iris["target"][:i]
        break

for i in range(len(iris["target"])):
    if(iris["target"][i]==0):
        iris["target"][i] = 1
    else:
        iris["target"][i] = -1

x = np.zeros((100,3))
for i in range(len(iris["data"])):
    x[i][0] = 1
    for j in range(len(iris["data"][i])):
        x[i][j+1] = iris["data"][i][j]
```

✓ 0.0s

I use sklearn dataset from the handout. If we reach the third class of iris, stop reading data immediately. Thus, our data may only have 2 classes of iris.

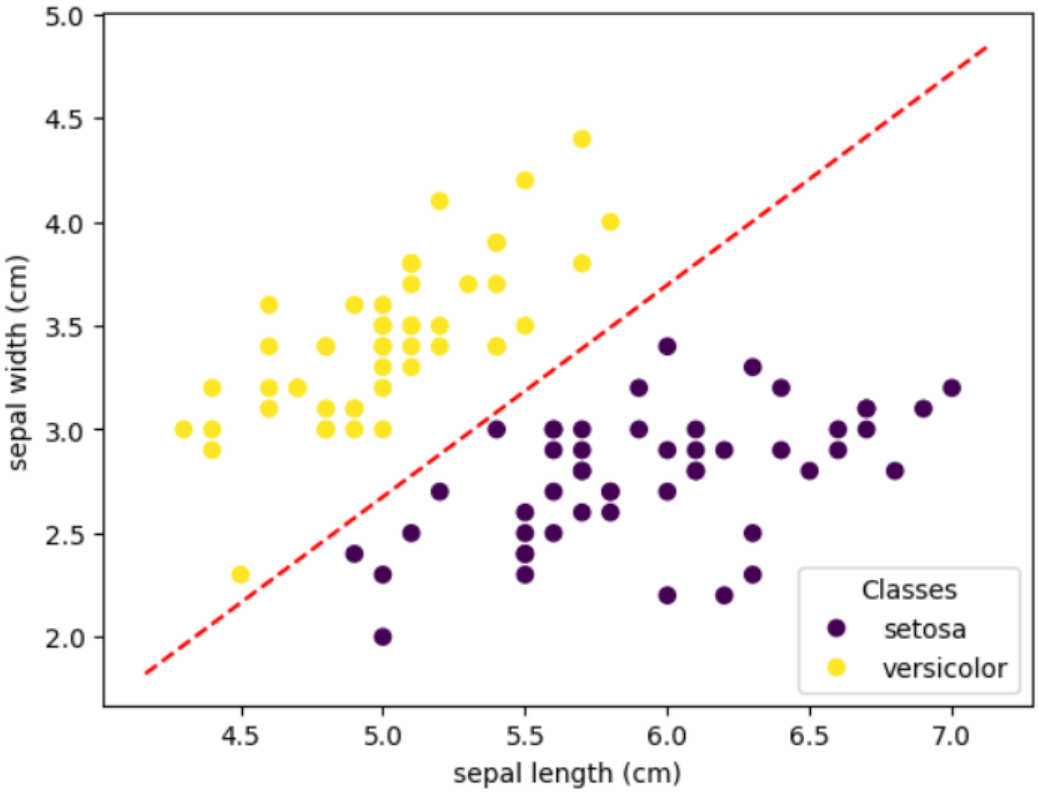
In the lower code, I turn y into 1 and -1, also adding 1 into every row of x to have a intercept.

c) This is the scatterplot of iris classification. (Perceptron Learning Algorithm)



We can see that it separates two classes of iris successfully, however, there are still two dots next to the decision boundary.

d) This is the scatterplot of iris classification. (Logistic regression)



Effectiveness: Logistic regression is more effective than Perceptron Learning Algorithm we can see that the sum of distance in Logistic regression is smaller than Perceptron Learning Algorithm.

Efficiency: Logistic regression calculate the probability directly, whereas Perceptron Learning Algorithm needs 1444 iterations.

2. Consider the case of the XOR function in which the two points $\{(0,0),(1,1)\}$ belong to one class, and the other two points $\{(1,0),(0,1)\}$ belong to the other class. Show how you can use the ReLU activation function to separate the two classes.

```
def ReLU_1(x):
    model_w1 = np.array([[-1],[1]])
    return max(model_w1.reshape(1,2).dot(x.reshape(2,1)),0)

def ReLU_2(x):
    model_w2 = np.array([[1],[-1]])
    return max(model_w2.reshape(1,2).dot(x.reshape(2,1)),0)

dotx = [(0,0), (0,1), (1,0), (1,1)]
model_x = np.array([[0,0], [0,1], [1,0], [1,1]])

dot_outputs = {}

for i in range(len(dotx)):
    if(ReLU_1(model_x[i]) + ReLU_2(model_x[i])):
        dot_outputs[dotx[i]] = -1
    else:
        dot_outputs[dotx[i]] = 1
```

```
The dots and its classification:  {(0, 0): 1, (0, 1): -1, (1, 0): -1, (1, 1): 1}

Class1: (0, 0), (1, 1)
Class2: (0, 1), (1, 0)
```

Code Explaination

I use the linear combination of two nodes to classify 4 dots.

The first node is $w_1 \cdot x_i$, where x_i is the set of dots and $w_1$is $\begin{bmatrix} 1 \\ -1 \end{bmatrix}$. We can classify (1,0) from other dots in ReLU by this weight.

The second node is $w_2 \cdot x_i$, where x_i is the set of dots and $w_1$is $\begin{bmatrix} -1 \\ 1 \end{bmatrix}$. We can classify (0,1) from other dots in ReLU by this weight.

By the first node adding the second node, we can say that if the summation of ReLU is 0, then two components of dots must be the same. Otherwise, they are different.

3. Show that the derivative of the logistic activation function is at most 0.25, irrespective of the value of its argument. At what value of its argument does the sigmoid activation function take on its maximum value?

Calculating the f.o.c. of logistic activation function

Logistic activation function: $f(x) = \frac{1}{1+e^{-x}}$

$$\begin{aligned}\frac{\partial}{\partial x} f(x) &= \frac{\partial}{\partial x}(1+e^{-x})^{-1} \\ &= -(1+e^{-x})^{-2} * \frac{\partial}{\partial x}(1+e^{-x}) \\ &= -(1+e^{-x})^{-2} * (-e^{-x}) \\ &= (e^{-x}) * (1+e^{-x})^{-2} \\ &= \frac{1}{1+e^{-x}} * \frac{e^{-x}}{1+e^{-x}} \\ &= f(x)(1-f(x))\end{aligned}$$

Get the maximum

Let $g(x) = f(x)(1-f(x))$

We want to find f.o.c. of $f(x)$.

$$\begin{aligned}\frac{\partial}{\partial x} g(x) &= (\frac{\partial}{\partial x} f(x)) * (1-f(x)) + f(x) * \frac{\partial}{\partial x}(1-f(x)) \\ &= f(x)(1-f(x)) * (1-f(x)) + f(x) * (-f(x)) * (1-f(x)) \\ &= (f(x))^3 - 2(f(x))^2 + f(x) + (f(x))^3 - (f(x))^2 \\ &= 2(f(x))^3 - 3(f(x))^2 + f(x) \\ &= f(x)(2f(x)-1)(f(x)-1)\end{aligned}$$

Find x **that can make** $\frac{\partial}{\partial x} g(x) = 0$

$$\begin{aligned}\frac{\partial}{\partial x} g(x) &= f(x)(2f(x)-1)(f(x)-1) \\ f(x) &= 0 \quad or \quad \frac{1}{2} \quad or \quad 1\end{aligned}$$

Because of $0 \le e^{-x} \to 0 \le f(x) \le 1$

$f(x)$ will only be 1 when
$x = -\infty$

$f(x)$ will only be 0 when $x = \infty$

If $\frac{1}{1+e^{-x}} = \frac{1}{2}$:

$$\begin{aligned}\frac{1}{1+e^{-x}} &= \frac{1}{2} \\ 1+e^{-x} &= 2 \\ e^{-x} &= 1 \\ x &= 0\end{aligned}$$

Verify g(x) has maximum

Find if $\frac{\partial}{\partial x^2} g(x) < 0$

$$\begin{aligned}\frac{\partial}{\partial x^2}g(x) &= \frac{\partial}{\partial x}2(f(x))^3 - 3(f(x))^2 + f(x) \\ &= \frac{\partial(2(f(x))^3 - 3(f(x))^2 + f(x))}{\partial f(x)} * \frac{\partial f(x)}{\partial x} \\ &= (6(f(x))^2 - 6f(x)) * f(x)(1 - f(x))\end{aligned}$$

$$\begin{aligned}When x = 0 \rightarrow \frac{\partial}{\partial x^2}g(x) &= (6(f(0))^2 - 6f(0)) * f(0)(1 - f(0)) \\ \frac{\partial}{\partial x^2}g(x) &= (6 * (\frac{1}{2})^2 - 6 * \frac{1}{2}) * \frac{1}{2}(1 - \frac{1}{2}) \\ \frac{\partial}{\partial x^2}g(x) &= (\frac{6}{4} - 3) * \frac{1}{4} = -\frac{3}{8} \\ \frac{\partial}{\partial x^2}g(x) &< 0\end{aligned}$$

When $x = 0 \rightarrow \frac{\partial}{\partial x}g(x) = 0$ and
$\frac{\partial}{\partial x^2}g(x) < 0$

$g(0) = f(0)(1 - f(0)) = \frac{1}{2}(1 - \frac{1}{2}) = \frac{1}{4} = 0.25$

The derivative of the logistic activation function is at most 0.25, irrespective of the value of its argument.

The sigmoid activation function take on its maximum value at $x = 0$.

4. Consider the softmax activation function in the output layer, in which real-valued outputs $v_1 \ldots v_k$ are converted into probabilities as follows:

$$o_i = \frac{\exp(v_i)}{\sum_{j=1}^{k} \exp(v_j)} \quad \forall i \in \{1, \ldots, k\}$$

(a) Show that the value of $\frac{\partial o_i}{\partial v_j}$ is $o_i(1 - o_i)$ when $i = j$. In the case that $i \neq j$, show that this value is $-o_i o_j$.

Let $o_i = \frac{e^{v_1}}{e^{v_1}+e^{v_2}+e^{v_3}+\cdots}$

Step 1

$$\begin{bmatrix} \frac{\partial}{\partial v_1}\frac{e^{v_1}}{e^{v_1}+e^{v_2}+e^{v_3}} & \frac{\partial}{\partial v_2}\frac{e^{v_1}}{e^{v_1}+e^{v_2}+e^{v_3}} & \frac{\partial}{\partial v_3}\frac{e^{v_1}}{e^{v_1}+e^{v_2}+e^{v_3}} & \cdots \\ \frac{\partial}{\partial v_1}\frac{e^{v_2}}{e^{v_1}+e^{v_2}+e^{v_3}} & \frac{\partial}{\partial v_2}\frac{e^{v_2}}{e^{v_1}+e^{v_2}+e^{v_3}} & \frac{\partial}{\partial v_3}\frac{e^{v_2}}{e^{v_1}+e^{v_2}+e^{v_3}} & \cdots \\ \frac{\partial}{\partial v_1}\frac{e^{v_3}}{e^{v_1}+e^{v_2}+e^{v_3}} & \frac{\partial}{\partial v_2}\frac{e^{v_3}}{e^{v_1}+e^{v_2}+e^{v_3}} & \frac{\partial}{\partial v_3}\frac{e^{v_3}}{e^{v_1}+e^{v_2}+e^{v_3}} & \cdots \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix}$$

Step2

$$\begin{bmatrix} \frac{\partial}{\partial v_1}\frac{e^{v_1}}{e^{v_1}+e^{v_2}+e^{v_3}} & \frac{\partial}{\partial v_2}\frac{e^{v_1}}{e^{v_1}+e^{v_2}+e^{v_3}} & \frac{\partial}{\partial v_3}\frac{e^{v_1}}{e^{v_1}+e^{v_2}+e^{v_3}} & \cdots \\ \frac{\partial}{\partial v_1}\frac{e^{v_2}}{e^{v_1}+e^{v_2}+e^{v_3}} & \frac{\partial}{\partial v_2}\frac{e^{v_2}}{e^{v_1}+e^{v_2}+e^{v_3}} & \frac{\partial}{\partial v_3}\frac{e^{v_2}}{e^{v_1}+e^{v_2}+e^{v_3}} & \cdots \\ \frac{\partial}{\partial v_1}\frac{e^{v_3}}{e^{v_1}+e^{v_2}+e^{v_3}} & \frac{\partial}{\partial v_2}\frac{e^{v_3}}{e^{v_1}+e^{v_2}+e^{v_3}} & \frac{\partial}{\partial v_3}\frac{e^{v_3}}{e^{v_1}+e^{v_2}+e^{v_3}} & \cdots \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix}$$

Step 3

$$\begin{bmatrix} \frac{\partial}{\partial v_1}e^{v_1}(e^{v_1}+e^{v_2}+e^{v_3})^{-1} & \frac{\partial}{\partial v_2}e^{v_1}(e^{v_1}+e^{v_2}+e^{v_3})^{-1} & \frac{\partial}{\partial v_3}e^{v_1}(e^{v_1}+e^{v_2}+e^{v_3})^{-1} & \cdots \\ \frac{\partial}{\partial v_1}e^{v_2}(e^{v_1}+e^{v_2}+e^{v_3})^{-1} & \frac{\partial}{\partial v_2}e^{v_2}(e^{v_1}+e^{v_2}+e^{v_3})^{-1} & \frac{\partial}{\partial v_3}e^{v_2}(e^{v_1}+e^{v_2}+e^{v_3})^{-1} & \cdots \\ \frac{\partial}{\partial v_1}e^{v_3}(e^{v_1}+e^{v_2}+e^{v_3})^{-1} & \frac{\partial}{\partial v_2}e^{v_3}(e^{v_1}+e^{v_2}+e^{v_3})^{-1} & \frac{\partial}{\partial v_3}e^{v_3}(e^{v_1}+e^{v_2}+e^{v_3})^{-1} & \cdots \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix}$$

Step 4

$$\begin{bmatrix} \frac{\partial}{\partial v_1}e^{v_1}(e^{v_1}+e^{v_2}+e^{v_3})^{-1} & \frac{\partial}{\partial v_2}e^{v_1}(e^{v_1}+e^{v_2}+e^{v_3})^{-1} & \frac{\partial}{\partial v_3}e^{v_1}(e^{v_1}+e^{v_2}+e^{v_3})^{-1} & \cdots \\ \frac{\partial}{\partial v_1}e^{v_2}(e^{v_1}+e^{v_2}+e^{v_3})^{-1} & \frac{\partial}{\partial v_2}e^{v_2}(e^{v_1}+e^{v_2}+e^{v_3})^{-1} & \frac{\partial}{\partial v_3}e^{v_2}(e^{v_1}+e^{v_2}+e^{v_3})^{-1} & \cdots \\ \frac{\partial}{\partial v_1}e^{v_3}(e^{v_1}+e^{v_2}+e^{v_3})^{-1} & \frac{\partial}{\partial v_2}e^{v_3}(e^{v_1}+e^{v_2}+e^{v_3})^{-1} & \frac{\partial}{\partial v_3}e^{v_3}(e^{v_1}+e^{v_2}+e^{v_3})^{-1} & \cdots \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix}$$

Step 5

$$
\begin{cases}
\frac{\partial}{\partial v_1} e^{v_1}(e^{v_1}+e^{v_2}+e^{v_3}+\cdots)^{-1} = (\frac{\partial}{\partial v_1}e^{v_1}) * (e^{v_1}+e^{v_2}+e^{v_3}+\cdots)^{-1} + (\frac{\partial}{\partial v_1}(e^{v_1}+e^{v_2}+e^{v_3}+\cdots)^{-1}) * (e^{v_1}) \\
\qquad = (e^{v_1}) * (e^{v_1}+e^{v_2}+e^{v_3}+\cdots)^{-1} + (-(e^{v_1}+e^{v_2}+e^{v_3})^{-2})(\frac{\partial}{\partial v_1}(e^{v_1}+e^{v_2}+e^{v_3}+\cdots)) * (e^{v_1} \\
\qquad = (e^{v_1}) * (e^{v_1}+e^{v_2}+e^{v_3}+\cdots)^{-1} - (e^{v_1}+e^{v_2}+e^{v_3}+\cdots)^{-2} * (e^{v_1}) * (e^{v_1}) \\
\qquad = (e^{v_1}) * (e^{v_1}+e^{v_2}+e^{v_3}+\cdots)^{-1} * (1 - (e^{v_1}) * (e^{v_1}+e^{v_2}+e^{v_3}+\cdots)^{-1}) \\
\qquad = o_1(1-o_1) \\
\frac{\partial}{\partial v_2} e^{v_1}(e^{v_1}+e^{v_2}+e^{v_3}+\cdots)^{-1} = (\frac{\partial}{\partial v_2}e^{v_1}) * (e^{v_1}+e^{v_2}+e^{v_3}+\cdots)^{-1} + (\frac{\partial}{\partial v_2}(e^{v_1}+e^{v_2}+e^{v_3}+\cdots)^{-1}) * (e^{v_1}) \\
\qquad = 0 * (e^{v_1}+e^{v_2}+e^{v_3}+\cdots)^{-1} + (-(e^{v_1}+e^{v_2}+e^{v_3})^{-2})(\frac{\partial}{\partial v_2}(e^{v_1}+e^{v_2}+e^{v_3}+\cdots)) * (e^{v_1}) \\
\qquad = -(e^{v_1}+e^{v_2}+e^{v_3}+\cdots)^{-2} * (e^{v_2}) * (e^{v_1}) \\
\qquad = -(e^{v_1}) * ((e^{v_1}+e^{v_2}+e^{v_3}+\cdots)^{-1}) * (e^{v_2}) * ((e^{v_1}+e^{v_2}+e^{v_3}+\cdots)^{-1}) \\
\qquad = -o_1 o_2
\end{cases}
$$

The same result holds wherever i is because of the property of differential.

$$
\begin{cases}
\frac{\partial}{\partial v_i}\frac{e^{v_i}}{e^{v_1}+e^{v_2}+e^{v_3}+\cdots} = \frac{\partial o_i}{\partial v_j} = o_i(1-o_i)\,, & i=j \\
\frac{\partial}{\partial v_j}\frac{e^{v_i}}{e^{v_1}+e^{v_2}+e^{v_3}+\cdots} = \frac{\partial o_i}{\partial v_j} = -o_i o_j\,, & i\neq j
\end{cases}
$$

(b) Use the above result to show that :

$$
\frac{\partial L}{\partial v_i} = o_i - y_i
$$

Assume that we are using the cross-entropy loss $L = -\sum_{i=1}^{k} y_i \log(o_i)$, where $y_i \in \{0,1\}$ is the one-hot encoded class label over different values of $i \in \{1,\ldots,k\}$.

f.o.c. of o_i is:

$$
\begin{cases}
\frac{\partial}{\partial v_i}\frac{e^{v_i}}{e^{v_1}+e^{v_2}+e^{v_3}+\cdots} = \frac{\partial o_i}{\partial v_j} = o_i(1-o_i)\,, & i=j \\
\frac{\partial}{\partial v_j}\frac{e^{v_i}}{e^{v_1}+e^{v_2}+e^{v_3}+\cdots} = \frac{\partial o_i}{\partial v_j} = -o_i o_j\,, & i\neq j
\end{cases}
$$

Step 1

$L = -y_1\log(o_1) - y_2\log(o_2) - y_3\log(o_3) - \cdots - y_k\log(o_k)$

Step 2

$\frac{\partial L}{\partial v_i} = -\frac{\partial}{\partial v_i}y_1\log(o_1) - \frac{\partial}{\partial v_i}y_2\log(o_2) - \frac{\partial}{\partial v_i}y_3\log(o_3) - \cdots - \frac{\partial}{\partial v_i}y_k\log(o_k)$

Step 3

In the case of $i \neq j$:

$$
\begin{aligned}
\frac{\partial}{\partial v_i}y_k\log(o_k) &= -((\frac{\partial}{\partial v_i}y_k) * \log(o_k) + (\frac{\partial}{\partial v_i}\log(o_k)) * y_k) \\
&= -(0 * \log(o_k) + (\frac{\partial}{\partial v_i}\log(o_k)) * y_k) \\
&= -(\frac{\partial}{\partial v_i}o_k * \frac{1}{o_k} * y_k) \\
&= -(-o_i o_k * \frac{1}{o_k} * y_k) \\
&= o_i * y_k
\end{aligned}
$$

In the case of $i = j$:

$$
\begin{aligned}
\frac{\partial}{\partial v_i} y_i \log(o_i) &= -((\frac{\partial}{\partial v_i} y_i) * \log(o_i) + (\frac{\partial}{\partial v_i} \log(o_k)) * y_i) \\
&= -(0 * \log(o_i) + (\frac{\partial}{\partial v_i} \log(o_i)) * y_i) \\
&= -(\frac{\partial}{\partial v_i} o_i * \frac{1}{o_i} * y_i) \\
&= -(o_i(1-o_i) * \frac{1}{o_i} * y_i) \\
&= -(1-o_i) * y_i
\end{aligned}
$$

Conclusion of Step 3:

$$
\begin{cases}
\frac{\partial}{\partial v_i} y_k \log(o_k) = o_i * y_k \ , & i \neq k \\
\frac{\partial}{\partial v_i} y_i \log(o_i) = (1-o_i) * y_i \ , & i = k
\end{cases}
$$

Step 4

$$
\begin{aligned}
\frac{\partial L}{\partial v_i} &= o_i * y_1 + o_i * y_2 + o_i * y_3 + \cdots + (-(1-o_i)) * y_i \\
&= o_i * y_1 + o_i * y_2 + o_i * y_3 + \cdots - y_i + o_i * y_i \\
&= (\sum_{i=1}^{k} o_i * y_i) - y_i \\
&= o_i * (\sum_{i=1}^{k} y_i) - y_i \\
&= o_i - y_i
\end{aligned}
$$

The same result holds wherever i is because of the property of differential.

$$
\begin{aligned}
L &= -\sum_{i=1}^{k} y_i \log(o_i) \\
\frac{\partial L}{\partial v_i} &= o_i - y_i
\end{aligned}
$$

5. Consider the elastic-net optimization problem:

$$\min_{\beta} \|\mathbf{y} - \mathbf{X}\beta\|^2 + \lambda \left[\alpha\|\beta\|_2^2 + (1-\alpha)\|\beta\|_1\right]$$

Show how one can turn this into a lasso problem, using an augmented version of $\mathbf{X}$ and $\mathbf{y}$.

We have the problem

$\min_{\beta} ||y - X\beta||_2^2 + \lambda[\ \alpha||\beta||_2^2 + (1-\alpha)||\beta||_1\ ]$

We want to turn this into a lasso problem

$\arg\min_{\beta} ||\widehat{y^{LASSO}} - \widehat{X^{LASSO}}\beta||_2^2 + \lambda[\ ||\beta||_1\ ]$

Let the size of $X_{m\times n}$, $\beta_{n\times 1}$ and $y_{m\times 1}$

$$\begin{aligned}&\min_{\beta} ||y - X\beta||_2^2 + \lambda[\ \alpha||\beta||_2^2 + (1-\alpha)||\beta||_1\ ] \\ =&\min_{\beta} ||y - X\beta||_2^2 + \lambda\alpha||\beta||_2^2 + \lambda(1-\alpha)||\beta||_1\end{aligned}$$

Find the addition entries of y and X.

$(y_{m+1} - X_{(m+1)\times n}\ \beta_{n\times 1})^2 = \lambda\alpha||\beta||_2^2 + \nu_1||\beta||_1 + \nu_2\ ,\quad \nu_1,\ \nu_2 \in \mathbb{R}^{1\times 1}$

Since r.h.s. do not have y, we can set $y_{m+1} = 0$ and let every entry in $X_{m+1} = \sqrt{\lambda\alpha}$.

we cat finally get $\sum_{j=1}^{n}(0 - \sqrt{\lambda\alpha}\beta_j)^2 = \lambda\alpha(\beta_j)^2 = \lambda\alpha||\beta||_2^2$

After getting rid of $||\beta||_2^2$, we can make the new formula be LASSO by change the coefficient of $||\beta||_1$.

Answer

$\widehat{y^{LASSO}} =$

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_m \\ 0 \end{bmatrix}$$

$\widehat{X^{LASSO}} =$

$$\begin{bmatrix} x_{11} & x_{12} & x_{13} & \cdots & x_{1n} \\ x_{21} & x_{22} & x_{23} & \cdots & x_{2n} \\ x_{31} & x_{32} & x_{33} & \cdots & x_{3n} \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ x_{m1} & x_{m2} & x_{m3} & \cdots & x_{mn} \\ \sqrt{\lambda\alpha} & \sqrt{\lambda\alpha} & \sqrt{\lambda\alpha} & \cdots & \sqrt{\lambda\alpha} \end{bmatrix}$$

LASSO form

$$\arg\min_{\beta} ||\widehat{y^{LASSO}} - \widehat{X^{LASSO}}\beta||_2^2 + \lambda^*[\ ||\beta||_1\ ]$$