Problem Set 2

Basil R. Yap
50.021 Artificial Intelligence - Term 8

May 27, 2018

1 Theory Component

[Q1]. Find *any* separating hyperplane equation for these three sample points: $\boldsymbol{x}_1 = (-2, 2), y_1 = -1$, $\boldsymbol{x}_2 = (4, 0), y_2 = -1$, and $\boldsymbol{x}_3 = (-2, -3), y_3 = +1$. Draw (by hand) or plot (using Python, see matplotlib) the result.

Solution

$$\begin{array}{ll} x_1 = (-2, 2) & y_1 = -1 \\ x_2 = (4, 0) & y_2 = -1 \\ x_3 = (-2, -3) & y_3 = +1 \end{array}$$

$$X = \begin{pmatrix} -2 & 2 & 1 \\ 4 & 0 & 1 \\ -2 & -3 & 1 \end{pmatrix}, \quad Y = \begin{pmatrix} -1 \\ -1 \\ +1 \end{pmatrix}$$

$$\omega = (\omega_1, \omega_2, b) \in \mathbb{R}^{d+1}$$

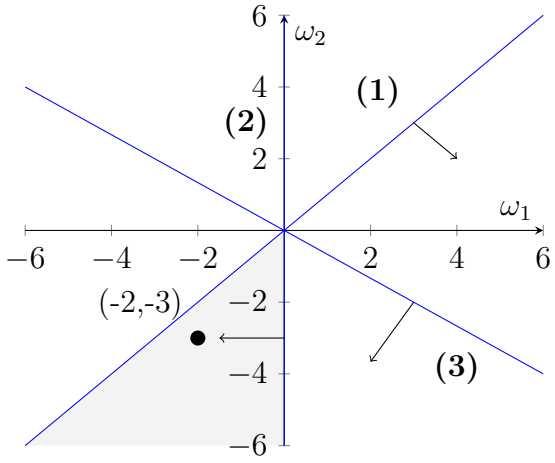
$$z = \omega x \begin{cases} > 0 & \text{If } y = +1 \\ < 0 & \text{If } y = -1 \\ = 0 & \text{If } y = 0 \end{cases}$$

$$\begin{aligned} -2\omega_1 + 2\omega_2 + b &< 0 \\ 4\omega_1 + b &< 0 \\ -2\omega_1 - 3\omega_2 + b &> 0 \end{aligned}$$

Any ω that satisfies this system of equations is a separating hyperplane.

Try solving for $b = 0$

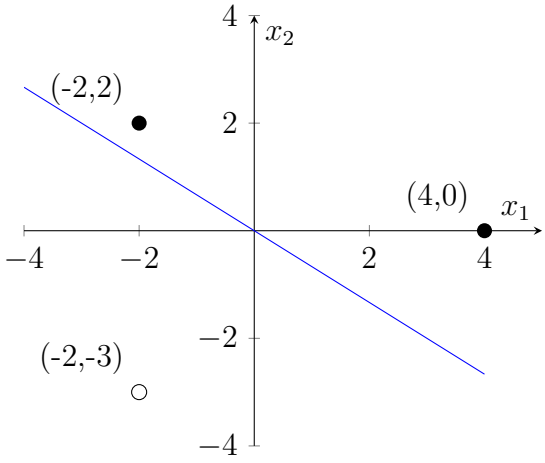
$$
\begin{aligned}
-2\omega_1 + 2\omega_2 &< 0 \\
\omega_2 &< \omega_1 \text{ — (1)} \\
4\omega_1 &< 0 \\
\omega_1 &< 0 \text{ — (2)} \\
-2\omega_1 - 3\omega_2 &> 0 \\
\omega_2 &< -\frac{2}{3}\omega_1 \text{ — (3)}
\end{aligned}
$$



Any value of ω_1 and ω_2 within the grey region is a feasible solution for the system.

$\omega_1 = -2, \omega_2 = -3$ is a feasible solution.

Below is the plot of $-2x_1 - 3x_2 = 0$



[Q2]. Find by hand the value of optimum weights $\hat{\boldsymbol{w}}$ and bias $\hat{b}$ using linear regression for the four following sample points: $\boldsymbol{x}_1 = (1, 0), y_1 = +1,\ \boldsymbol{x}_2 = (2, 3), y_2 = +1,\ \boldsymbol{x}_3 = (3, 4), y_3 = -1,\ \boldsymbol{x}_4 = (3, 2), y_4 = -1$. Show your working. If you have to do matrix inversion, then you can use a program to compute the matrix inverse. Answer the following question: How can you obtain a solution via the closed-form method (that is not by using gradients) which does not use an inverse matrix $(X^T X)^{-1}$ to obtain the weight vectoe w?

Solution

$$\begin{aligned} x_1 &= (1, 0) \quad y_1 = +1 \\ x_2 &= (2, 3) \quad y_2 = +1 \\ x_3 &= (3, 4) \quad y_3 = -1 \\ x_4 &= (3, 2) \quad y_4 = -1 \end{aligned}$$

$$X = \begin{pmatrix} 1 & 0 & 1 \\ 2 & 3 & 1 \\ 3 & 4 & 1 \\ 3 & 2 & 1 \end{pmatrix}, \quad Y = \begin{pmatrix} +1 \\ +1 \\ -1 \\ -1 \end{pmatrix}$$

$$\omega = (\omega_1, \omega_2, b) \in \mathbb{R}^{d+1}$$

$$z = \omega x \begin{cases} > 0 & \text{If } y = +1 \\ < 0 & \text{If } y = -1 \\ = 0 & \text{If } y = 0 \end{cases}$$

Using Mean Square Error as the point loss function, $\mathcal{L}_1 = \frac{1}{2}(y - \omega x)^2$

Through a series of operations (proven in the previous problem set and various various resources online), we arrive at the gradient function of the average loss value below[1]:

$$\nabla \mathcal{L}_n(y, z(\omega; x)) = \frac{1}{N}(X^T X \omega - XY)$$

We then find the exact solution at the turning point, $\nabla \mathcal{L}_n(y, z(\omega; x)) = 0$

$$\begin{aligned} \frac{1}{N}(X^T X\ \hat{\omega} - XY) &= 0 \\ X^T X\ \hat{\omega} - XY &= 0 \\ X^T X\ \hat{\omega} &= XY \\ \hat{\omega} &= (X^T X)^{-1} X^T Y \end{aligned}$$

$$\hat{\omega} = \left(\begin{bmatrix} 1 & 2 & 3 & 3 \\ 0 & 3 & 4 & 2 \\ 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 1 \\ 2 & 3 & 1 \\ 3 & 4 & 1 \\ 3 & 2 & 1 \end{bmatrix} \right)^{-1} \begin{bmatrix} 1 & 2 & 3 & 3 \\ 0 & 3 & 4 & 2 \\ 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ -1 \\ -1 \end{bmatrix}$$

Using a calculator,

$$\hat{\omega} = \begin{bmatrix} -1.5 \\ 0.3 \\ 2.7 \end{bmatrix}$$

$$\hat{\omega} = \begin{bmatrix} -1.5 \\ 0.3 \end{bmatrix} *, \quad \hat{b} = 2.7$$

$* \ \hat{\omega}$ *as defined by the question.*

To solve for an optimal solution $\hat{\omega}$ with a closed-form method without use of an inverse matrix, there are three possible cases to consider. The system of equations (i.e. the number of observations with respect to the number of dimensions/variables) can be **under-determined**,**solvable** or **over-determined**.

When the system of equations is **under-determined**, there are not enough observations to properly define $\hat{\omega}$ such that there is a single unique solution to the system of equations. In this case, the system of equations can be solved loosely with respect to $\hat{\omega}$ and a range/set of optimal values can be found.

When the system of equations is **solvable**, there are perfectly sufficient observations to solve for a unique $\hat{\omega}$. By simply solving the system of equations via simultaneous or row echelon method, a single unique solution for $\hat{\omega}$ can be found.

When the system of equations is **over-determined**, the number of observations is more than sufficient solve for $\hat{\omega}$, to the point where the value of $\hat{\omega}$ calculated may not hold true for all observations (this occurs when the data is not linearly separable). Other than an analytical solution using an inverse matrix or gradient descent, perhaps the only other way to get a reasonable estimate of $\hat{\omega}$ would be to solve the system of equation using the row echelon method and then subsequently relaxing the problem by dropping rows/observations from the matrix. This would provide a reasonable solution for most observations, but may not result in the lowest possible loss value.

[Q3] In logistic regression, we model the probability of a label $y \in \{-1, 1\}$ given a sample point $\boldsymbol{x} \in \mathbb{R}^2$ as,

$$P(Y = y|\boldsymbol{x}) = \frac{1}{1 + e^{-y(\boldsymbol{w}\cdot\boldsymbol{x}+c)}},$$

where c is the bias term. Answer the following questions,

1. Obviously the function above alone is not linear. However, given that after we know what the values $\boldsymbol{w}$ are (after training), *prove* that the *decision boundary* obtained from logistic regression is linear, i.e.: the decision boundary is a hyperplane with equation $\boldsymbol{w} \cdot \boldsymbol{x} = 0$.
   Hint: Since $\boldsymbol{x}$ is two-dimensional, the decision boundary is the set of $\boldsymbol{x}$ such that $P(Y = -1|\boldsymbol{x}) = P(Y = 1|\boldsymbol{x})$.

2. Suppose we have obtained the optimum weights, $\hat{w} = (w_1, w_2)$ and bias $\hat{c}$ in the case of 2-dimensional data points. The decision boundary is supposedly a line that separate points with positive and negative labels, which has an equation in the form of $x_2 = mx_1 + k$. Obtain an expression for m and k in terms of w_1, w_2 and $\hat{c}$. When this expression of the form $x_2 = mx_1 + k$ is not defined?

Solution

1) For any given x, the boundary condition would be when there is an equal likelihood of $Y = -1$ and $Y = +1$.

$$\begin{aligned}
\therefore \text{Evaluate } \Pr(Y = -1|x) &= \Pr(Y = +1|x) \\
\frac{1}{1 + e^{\omega\cdot x+c}} &= \frac{1}{1 + e^{-(\omega\cdot x+c)}} \\
1 + e^{\omega\cdot x+c} &= 1 + e^{-(\omega\cdot x+c)} \\
e^{\omega\cdot x+c} &= e^{-(\omega\cdot x+c)} \\
\omega \cdot x + c &= -(\omega \cdot x + c) \\
\omega \cdot x + c &= 0
\end{aligned}$$

$\therefore$ The decision boundary is linear.

2) From **Part 1**, the decision boundary line can be defined by $\omega \cdot x + c = 0$

Now consider for the case when weights and bias are optimal, $\hat{\omega} \cdot x + \hat{c} = 0$

$$\begin{bmatrix}\omega_1 & \omega_2\end{bmatrix} \cdot \begin{bmatrix}x_1 \\ x_2\end{bmatrix} + \hat{c} = 0$$

$$\omega_1 x_1 + \omega_2 x_2 + \hat{c} = 0$$
$$\omega_2 x_2 = -\omega_1 x_1 - \hat{c}$$
$$x_2 = -\frac{\omega_1}{\omega_2} x_1 - \frac{\hat{c}}{\omega_2}$$
$$m = -\frac{\omega_1}{\omega_2} \ , \ k = -\frac{\hat{c}}{\omega_2}$$
$$x_2 = mx_1 + k \text{ is not defined when } \omega_2 = 0$$

[Q4]. The following is the regularized logistic regression loss function.

$$L(w) = -\frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)} \log(h_w(x^{(i)})) + (1 - y^{(i)}) \log(1 - h_w(x^{(i)}))\right] + \frac{\lambda}{2}||w||^2$$

where $h_w(x) = \frac{1}{1+e^{-w \cdot x}}$. Compute the gradient of the loss with respect to w and write down the gradient descent update equation. Show all steps clearly.

Solution

$$\mathcal{L}(\omega) = -\frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)} \log(h_\omega(x^{(i)})) + (1 - y^{(i)}) \log(1 - h_\omega(x^{(i)}))\right] + \frac{\lambda}{2}||\omega||^2$$

$$\nabla\mathcal{L}(\omega) = \frac{d}{d\omega}\left(-\frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)} \log(h_\omega(x^{(i)})) + (1 - y^{(i)}) \log(1 - h_\omega(x^{(i)}))\right] + \frac{\lambda}{2}||\omega||^2\right)$$
$$= -\frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)} \frac{h'_\omega(x^{(i)})}{h_\omega(x^{(i)})} + (1 - y^{(i)}) \frac{-h'_\omega(x^{(i)})}{1 - h_\omega(x^{(i)})}\right] + \lambda\omega$$

$$h_\omega(x) = \frac{1}{1 + e^{-\omega \cdot x}}$$

$$h'_\omega(x) = \frac{d}{d\omega}\left(\frac{1}{1 + e^{-\omega \cdot x}}\right)$$
$$= -\left(\frac{1}{(1 + e^{-\omega \cdot x})^2}\right)(-xe^{-\omega \cdot x})$$
$$= xh_\omega(x)^2 e^{-\omega \cdot x}$$

$$\nabla\mathcal{L}(\omega) = -\frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)} \frac{x^{(i)} h_\omega(x^{(i)})^2 e^{-\omega \cdot x^{(i)}}}{h_\omega(x^{(i)})} + (1 - y^{(i)}) \frac{-x^{(i)} h_\omega(x^{(i)})^2 e^{-\omega \cdot x^{(i)}}}{1 - h_\omega(x^{(i)})}\right] + \lambda\omega$$
$$= -\frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)} x^{(i)} e^{-\omega \cdot x^{(i)}} h_\omega(x^{(i)}) + (y^{(i)} - 1)(x^{(i)} e^{-\omega \cdot x^{(i)}}) \frac{h_\omega(x^{(i)})^2}{1 - h_\omega(x^{(i)})}\right] + \lambda\omega$$

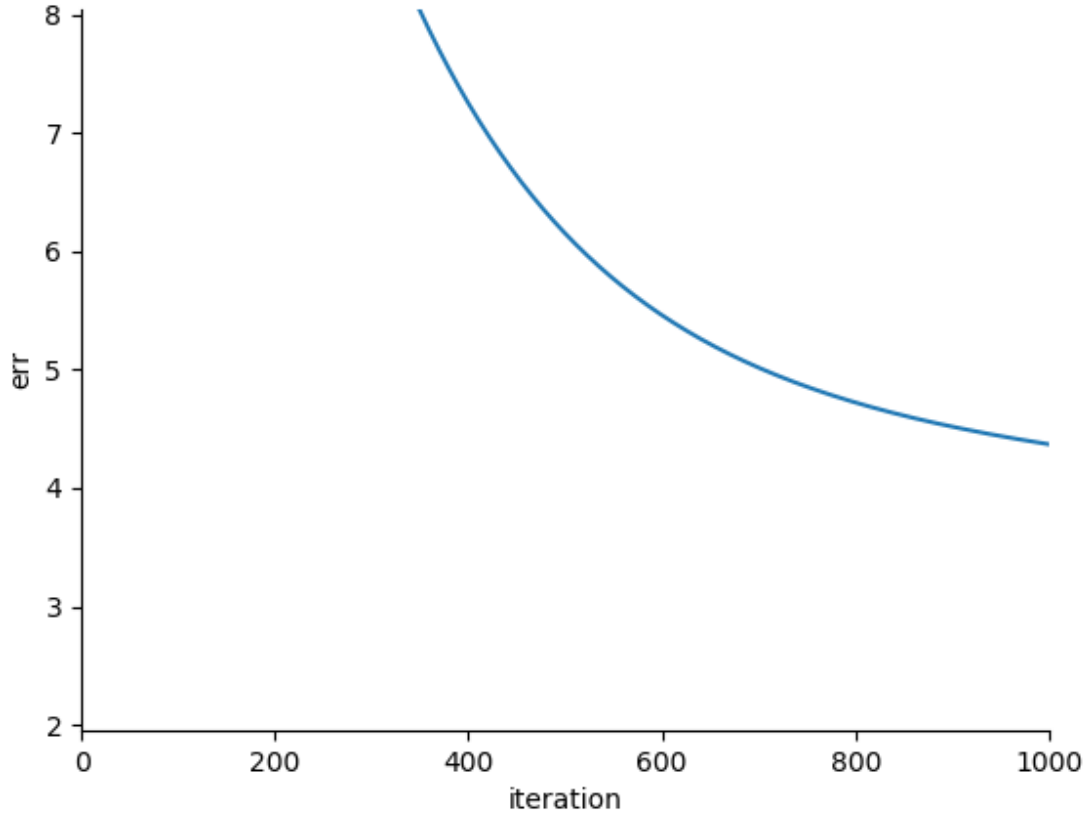
For each k step size, the new loss function value is given by

$$\mathcal{L}(\omega') \leftarrow \mathcal{L}(\omega) + k\nabla\mathcal{L}(\omega)$$

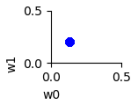
$$\mathcal{L}(\omega) + k\nabla\mathcal{L}(\omega) = -\frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)}\log(h_\omega(x^{(i)})) + (1-y^{(i)})\log(1-h_\omega(x^{(i)}))\right] + \frac{\lambda}{2}||\omega||^2$$
$$-\frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)}x^{(i)}e^{-\omega\cdot x^{(i)}}h_\omega(x^{(i)}) + (y^{(i)}-1)(x^{(i)}e^{-\omega\cdot x^{(i)}})\frac{h_\omega(x^{(i)})^2}{1-h_\omega(x^{(i)})}\right] + \lambda\omega$$

2 Coding Component

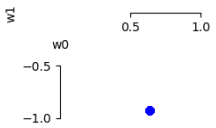
```
import learnThu8
import numpy as np

def gd(fxn, dfxn, x0, step_size=.01, max_iter=1000, eps=0.00001):
        x = x0
        ylist = []
        xlist = []

        for i in range(max_iter):
                ylist.append(fxn(x))
                xlist.append(x)
                diff = step_size * dfxn(x)
                x -= diff

                if (np.absolute(diff) < eps).all():
                        break

        return x, ylist, xlist

learnThu8.gd = gd

if __name__ == "__main__":
        learnThu8.t6()
```


8
7
6
5
4
3
2
err
0
200
400
600
800
1000
iteration

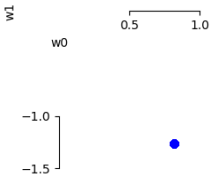
step size = 0.0001



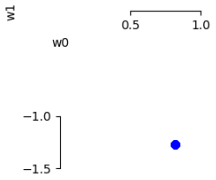
step size = 0.001



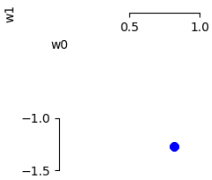
step size = 0.01



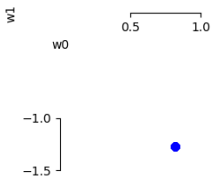
step size = 0.05

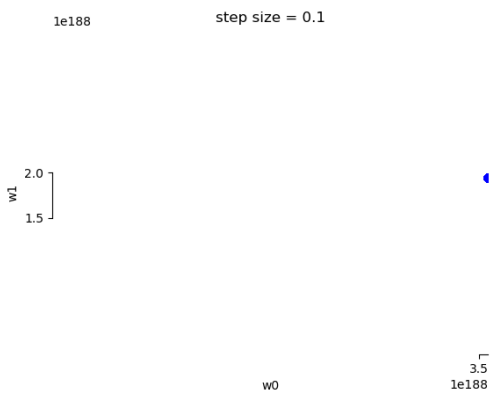


step size = 0.07



step size = 0.075





References

[1] Ng, A. (2000). CS229 Lecture notes. *CS229 Lecture notes*, *1*(1), 11.