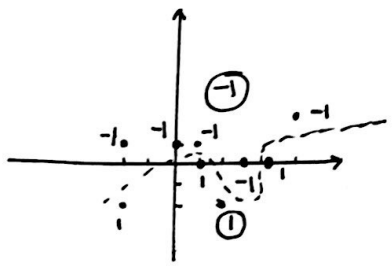
Homework Set #2. – M146

Name: Yangyang Mao UID: 504945234

1.(a)



From figure showned above, we can observe the data is not linearly separable.
Since we can not use one-line to separate those data in to 2 classes.
Thus the algorithm will not converge since the data is not linearly separable.

(b) $w_1 = x_1$, assume b=0. we have $\tilde{w} = \begin{bmatrix} 0 \\ 4 \\ 0 \end{bmatrix}$ $\tilde{x}_1 = \begin{bmatrix} 1 \\ 4 \\ 0 \end{bmatrix}$

$y_1 = \text{sign}(w_1^T x_1) = \text{sign}(16) = +1$

$x = x_2 = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$

$\text{sign}(w^T x) = \text{sign}(4) = +1 \neq y_2$

$w^{new} = w^{old} + y_2 \tilde{x}_2 = \begin{bmatrix} 0 \\ 4 \\ 0 \end{bmatrix} + (-1)\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} -1 \\ +3 \\ -1 \end{bmatrix}$

(c) $\tilde{x} = \tilde{x}_3 = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$ $\text{sign}(w^T x) = -1 = y_3$. we will do nothing.

$\tilde{x} = \tilde{x}_4 = \begin{bmatrix} 1 \\ -2 \\ -2 \end{bmatrix}$ $\text{sign}(w^T x) = -1 \neq y_4$. we will do nothing

$\tilde{x} = \tilde{x}_5 = \begin{bmatrix} 1 \\ -2 \\ 1 \end{bmatrix}$ $\text{sign}(w^T x) = 1 \neq y_5$

$w^{new} = w^{old} + y_4 \tilde{x}_4 = \begin{bmatrix} -1 \\ 3 \\ -1 \end{bmatrix} + (1)\cdot\begin{bmatrix} 1 \\ -2 \\ -2 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ -3 \end{bmatrix}$

$x = \tilde{x}_5 = \begin{bmatrix} 1 \\ -2 \\ 1 \end{bmatrix}$ $\text{sign}(w^T x) = -1 = y_5$. we will do nothing

$x = \tilde{x}_6 = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}$ $\text{sign}(w^T x) = +1 = y_6$. we will do nothing

$x = \tilde{x}_7 = \begin{bmatrix} 1 \\ \frac{1}{2} \\ \frac{1}{2} \end{bmatrix}$ $\text{sign}(w^T x) = -1 = y_7$. we will do nothing.

$x = \tilde{x}_8 = \begin{bmatrix} 1 \\ 3 \\ 0 \end{bmatrix}$ $\text{sign}(w^T x) = +1 \neq y_8$.

$w^{new} = w^{old} + y_8 \tilde{x}_8 = \begin{bmatrix} 0 \\ 1 \\ -3 \end{bmatrix} + (-1)\begin{bmatrix} 1 \\ 3 \\ 0 \end{bmatrix} = \begin{bmatrix} -1 \\ -2 \\ -3 \end{bmatrix}$

(c) Voted perceptron:

$c_1 = 1 \cdot c_2 = 2 \cdot c_3 = 4 \cdot c_4 = 1$

$w_1 = \begin{bmatrix} 0 \\ 4 \\ 0 \end{bmatrix}$ $w_2 = \begin{bmatrix} -1 \\ 3 \\ -1 \end{bmatrix}$ $w_3 = \begin{bmatrix} 0 \\ 1 \\ -3 \end{bmatrix}$ $w_4 = \begin{bmatrix} -1 \\ -2 \\ -3 \end{bmatrix}$

$\hat{y} = \text{sign}\left(\sum_{i=1}^{4} c_i \, \text{sign}(w_i^T x)\right)$

Average perceptron:

$w_{avg} = \frac{1}{8}\sum_{k=1}^{4} c_k w_k = \frac{1}{8}\left(\begin{bmatrix} 0 \\ 4 \\ 0 \end{bmatrix} + \begin{bmatrix} -2 \\ 6 \\ -2 \end{bmatrix} + \begin{bmatrix} 0 \\ 4 \\ -12 \end{bmatrix} + \begin{bmatrix} -1 \\ -2 \\ -3 \end{bmatrix}\right)$

$= \begin{bmatrix} -318 \\ 312 \\ -1718 \end{bmatrix}$

$\hat{y} = \text{sign}(w_{avg}^T x)$

(d) perceptron:

$\hat{y}_1 = -1 \neq y_1$. $\hat{y}_2 = -1 = y_2$. $\hat{y}_3 = -1 = y_3$. $\hat{y}_4 = 1 = y_4$. $\hat{y}_5 = 1 \neq y_5$. $\hat{y}_6 = -1 \neq y_6$. $\hat{y}_7 = -1 = y_7$. $\hat{y}_8 = -1 = y_8$

training error of perceptron: $\frac{3}{8}$

The voted perceptron:

$\hat{y}_1 = \text{Sign}\left(1 \times \text{sign}([0\ 4\ 0]\begin{bmatrix}1\\4\\0\end{bmatrix}) + 2 \times \text{Sign}([-1\ 3\ -1]\begin{bmatrix}1\\4\\0\end{bmatrix}) + 4 \times \text{sign}([0\ 1\ -3]\begin{bmatrix}1\\4\\0\end{bmatrix}) + 1 \times \text{Sign}([-1\ -2\ -3]\begin{bmatrix}1\\4\\0\end{bmatrix}) \right)$

$= \text{Sign}(1\times1 + 2\times1 + 4\times1 + 2\times(-1)) = 1 = y_1$

$\hat{y}_2 = \text{Sign}(1 \times \text{sign}([0\ 4\ 0]\begin{bmatrix}1\\1\\1\end{bmatrix}) + 2 \times \text{sign}([-1\ 3\ -1]\begin{bmatrix}1\\1\\1\end{bmatrix}) + 4 \times \text{Sign}([0\ 1\ -3]\begin{bmatrix}1\\1\\1\end{bmatrix}) + 1 \times \text{Sign}([-1\ -2\ -3]\begin{bmatrix}1\\1\\1\end{bmatrix}))$

$= \text{Sign}(1\times1 + 2\times1 + 4\times(-1) + 1\times(-1)) = -1 = y_2$

$\hat{y}_3 = \text{Sign}(1 \times \text{Sign}([0\ 4\ 0]\begin{bmatrix}0\\1\\1\end{bmatrix}) + 2 \times \text{sign}([-1\ 3\ -1]\begin{bmatrix}0\\1\\1\end{bmatrix}) + 4 \times \text{Sign}([0\ 1\ -3]\begin{bmatrix}0\\1\\1\end{bmatrix}) + 1 \times \text{sign}([-1\ -2\ -3]\begin{bmatrix}0\\1\\1\end{bmatrix}))$

$= \text{Sign}(1\times1 + 2\times(-1) + 4\times(-1) + 1\times(-1)) = -1 = y_3$

$\hat{y}_4 = \text{Sign}(1 \times \text{sign}([0\ 4\ 0]\begin{bmatrix}-1\\-2\\-2\end{bmatrix}) + 2 \times \text{Sign}([-1\ 3\ -1]\begin{bmatrix}-1\\-2\\-2\end{bmatrix}) + 4 \times \text{Sign}([0\ 1\ -3]\begin{bmatrix}-1\\-2\\-2\end{bmatrix}) + \text{Sign}([-1\ -2\ -3]\begin{bmatrix}-1\\-2\\-2\end{bmatrix}))$

$= \text{Sign}(1\times(-1) + 2\times(-1) + 4\times1 + 1\times1) = 1 = y_4$

$\hat{y}_5 = \text{Sign}(1 \times \text{Sign}([0\ 4\ 0]\begin{bmatrix}-1\\-2\\1\end{bmatrix}) + 2 \times \text{Sign}([-1\ 3\ -1]\begin{bmatrix}-1\\-2\\1\end{bmatrix}) + 4 \times \text{sign}([0\ 1\ -3]\begin{bmatrix}-1\\-2\\1\end{bmatrix}) + 1 \times \text{Sign}([-1\ -2\ -3]\begin{bmatrix}-1\\-2\\1\end{bmatrix}))$

$= \text{Sign}(1\times(-1) + 2\times(-1) + 4\times(-1) + 1\times(-1)) = -1 = y_5$

$\hat{y}_6 = \text{Sign}(1 \times \text{Sign}([0\ 4\ 0]\begin{bmatrix}1\\1\\0\end{bmatrix}) + 2 \times \text{Sign}([-1\ 3\ -1]\begin{bmatrix}1\\1\\0\end{bmatrix}) + 4 \times \text{Sign}([0\ 1\ -3]\begin{bmatrix}1\\1\\0\end{bmatrix}) + 1 \times \text{Sign}([-1\ -2\ -3]\begin{bmatrix}1\\1\\0\end{bmatrix}))$

$= \text{Sign}(1\times1 + 2\times1 + 4\times1 + 1\times(-1)) = 1 = y_6$

$\hat{y}_7 = \text{Sign}(1 \times \text{Sign}([0\ 4\ 0]\begin{bmatrix}1\\5\\2\end{bmatrix}) + 2 \times \text{Sign}([-1\ 3\ -1]\begin{bmatrix}1\\5\\2\end{bmatrix}) + 4 \times \text{Sign}([0\ 1\ -3]\begin{bmatrix}1\\5\\2\end{bmatrix}) + 1 \times \text{Sign}([-1\ -2\ -3]\begin{bmatrix}1\\5\\2\end{bmatrix}))$

$= \text{Sign}(1\times1 + 2\times1 + 4\times(-1) + 1\times(-1)) = -1 = y_7$

$\hat{y}_8 = \text{Sign}(1 \times \text{Sign}([0\ 4\ 0]\begin{bmatrix}1\\3\\0\end{bmatrix}) + 2 \times \text{Sign}([-1\ 3\ -1]\begin{bmatrix}1\\3\\0\end{bmatrix}) + 4 \times \text{Sign}([0\ 1\ -3]\begin{bmatrix}1\\3\\0\end{bmatrix}) + 1 \times \text{Sign}([-1\ -2\ -3]\begin{bmatrix}1\\3\\0\end{bmatrix}))$

$= \text{Sign}(1\times1 + 2\times1 + 4\times1 + 1\times(-1)) = 1 \neq y_8$

training error of voted perceptron: $\frac{1}{8}$

Average perceptron:

$\hat{y}_1 = \text{Sign}\left(\begin{bmatrix}-318\\312\\-1718\end{bmatrix}^T \begin{bmatrix}0\\4\\0\end{bmatrix}\right) = 1 = y_1$ $\quad$ $\hat{y}_2 = \text{Sign}\left(\begin{bmatrix}-318\\312\\-1718\end{bmatrix}^T \begin{bmatrix}1\\1\\1\end{bmatrix}\right) = -1 = y_2$

$\hat{y}_3 = \text{Sign}\left(\begin{bmatrix}-318\\312\\-1718\end{bmatrix}^T \begin{bmatrix}1\\0\\1\end{bmatrix}\right) = -1 = y_3$ $\quad$ $\hat{y}_4 = \text{Sign}\left(\begin{bmatrix}-318\\312\\-1718\end{bmatrix}^T \begin{bmatrix}-1\\-2\\-2\end{bmatrix}\right) = 1 = y_4$

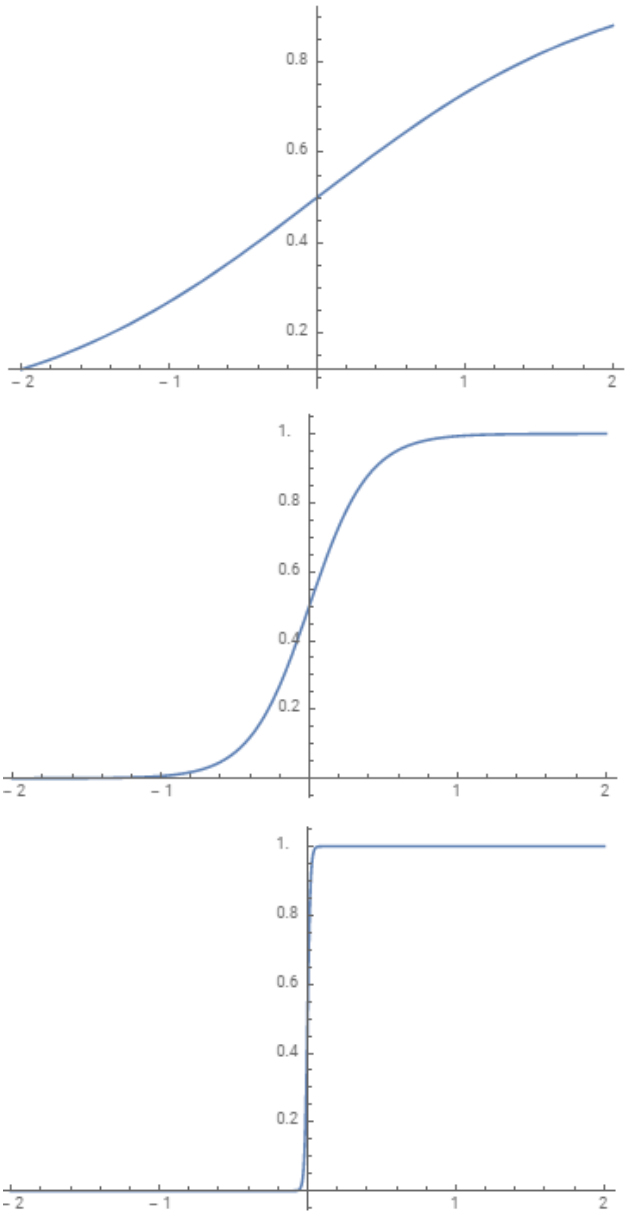
$\hat{y}_5 = \text{Sign}\left(\begin{bmatrix}-318\\312\\-1718\end{bmatrix}^T \begin{bmatrix}-1\\-2\\1\end{bmatrix}\right) = -1 = y_5$ $\quad$ $\hat{y}_6 = \text{Sign}\left(\begin{bmatrix}-318\\312\\-1718\end{bmatrix}^T \begin{bmatrix}1\\1\\0\end{bmatrix}\right) = 1 = y_6$

$\hat{y}_7 = \text{Sign}\left(\begin{bmatrix}-318\\312\\-1718\end{bmatrix}^T \begin{bmatrix}1\\5\\2\end{bmatrix}\right) = 1 \neq y_7$ $\quad$ $\hat{y}_8 = \text{Sign}\left(\begin{bmatrix}-318\\312\\-1718\end{bmatrix}^T \begin{bmatrix}1\\3\\0\end{bmatrix}\right) = 1 \neq y_8$

training error of average perceptron: $\frac{2}{8} = \frac{1}{4}$

Thus, training error: perceptron > Average perceptron > voted perceptron

2(a)



The figures shown above are sigmoid function with different weights, 1, 5, 100, respectively, in same scale. The figures show that with weights increasing, y will increase with x faster. Hence, large weight can lead to a problem of overfitting since it will lead to value of function output too large or too small (more close to 0 or 1), then its slope becomes too small. For example, when weight=1, the slope of the function is similar at all points between [-2,2], when weight=5, the slope of the function is similar at all points between [-0.8,0.8], however, when weight =100, its nearly output neither 0 or 1, the function saturate too fast. Hence, it will lead to weight of unimportant features too large, and then result in overfitting.

3. (a) $\frac{\partial J(w_0,w_1)}{\partial w_0} = \sum_{n=1}^{N} 2\alpha_n (w_0 + w_1 x_{n,1} - y_n)$

$\frac{\partial J(w_0,w_1)}{\partial w_1} = \sum_{n=1}^{N} 2\alpha_n \cdot x_{n,1} (w_0 + w_1 x_{n,1} - y_n)$

(b) $\frac{\partial^2 J(w_0,w_1)}{\partial w_0^2} = \sum_{n=1}^{N} 2\alpha_n (1 + w_1 x_{n,1} - y_n)$ $\frac{\partial^2 J(w_0,w_1)}{\partial w_1^2} = \sum_{n=1}^{N} 2\alpha_n \cdot x_{n,1} (w_0 + x_{n,1} - y_n)$

$\frac{\partial^2 J(w_0,w_1)}{\partial w_0 \partial w_1} = \sum_{n=1}^{N} 2\alpha_n (w_0 + x_{n,1} - y_n)$ $\frac{\partial^2 J(w_0,w_1)}{\partial w_1 \partial w_0} = \sum_{n=1}^{N} 2\alpha_n \cdot x_{n,1} (1 + w_1 x_{n,1} - y_n)$

$$H = \begin{bmatrix} \sum_{n=1}^{N} 2\alpha_n (1 + w_1 x_{n,1} - y_n) & \sum_{n=1}^{N} 2\alpha_n (w_0 + x_{n,1} - y_n) \\ \sum_{n=1}^{N} 2\alpha_n (w_0 + x_{n,1} - y_n) & \sum_{n=1}^{N} 2\alpha_n \cdot x_{n,1} (w_0 + x_{n,1} - y_n) \end{bmatrix}$$

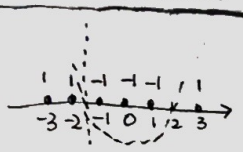
$\forall z \in R^2$. $z^T H z = 2z_1^2 \sum_{n=1}^{N} \alpha_n + 2z_1 z_2 \sum_{n=1}^{N} \alpha_n x_{n,1} + 2z_1 z_2 \sum_{n=1}^{N} \alpha_n x_{n,1} + 2z_2^2 \sum_{n=1}^{N} \alpha_n \cdot x_{n,1}^2$

$= 2\sum_{n=1}^{N} \alpha_n (z_1^2 + 2z_1 z_2 x_{n,1} + z_2^2 x_{n,1}^2) = 2\sum_{n=1}^{N} \alpha_n (z_1 + z_2 x_{n,1})^2 \geq 0$

since $\alpha_n > 0$. $(z_1 + z_2 x_{n,1})^2 \geq 0$. Thus $z^T H z \geq 0$.

Thus. the Hessian is positive semidefinite, thus the function is convex, so Eq.1 has a global optimal solution. since in convex function, any local minimum is a global minimum.

4. (a)



We can take $w_1 = -1$ $w_0 = -1.5$

Then $f(x) = \text{sign}(-x - 1.5)$

The classification accuracy is 5/6.

(b) $K(x,z) = xz(1+xz) = (xz)^2 + xz$

$k_1 = xz$, corresponding $\phi_1(x) = x$.

$k_2 = x^2 z^2$. corresponding $\phi_2(x) = x^2$.

$\therefore \phi(x) = (x, x^2)$

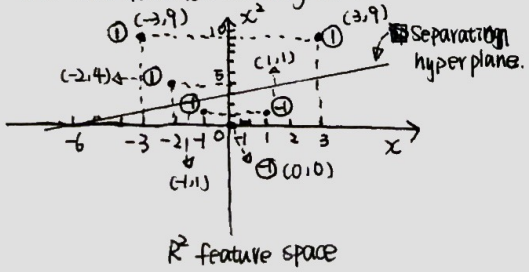
(c) $f(x) = \text{Sign}(w_1 x + w_2 x^2 + w_0)$

We can take $f(x) = \text{sign}((x+1.5)(x-2))$ as shown in the figure above.

$f(x) = \text{Sign}(x^2 - 0.5x - 3)$, $w_0 = -3$. $w_1 = -0.5$. $w_2 = 1$, $w = \begin{bmatrix} -3 \\ -0.5 \\ 1 \end{bmatrix}$

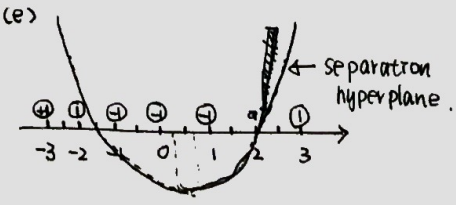
The classification accuray is 1.

(d)



R^2 feature space

(e)



R^1 feature space.

2.(b) $\frac{\partial J}{\partial w_j} = \frac{\partial -\sum_{n=1}^{N}[y_n \log h_w(x_n) + (1-y_n)\log(1-h_w(x_n))] + \partial[\frac{1}{2}\sum_i w_i^2]}{\partial w_j}$

$$= -\sum_{n=1}^{N}\left[\frac{\partial[y_n \log h_w(x_n)]}{\partial w_j} + \frac{\partial[(1-y_n)\log(1-h_w(x_n))]}{\partial w_j}\right] + w_j$$

$$= -\sum_{n=1}^{N}\left[y_n \frac{\partial \log h_w(x_n)}{\partial w_j} + (1-y_n)\frac{\partial \log(1-h_w(x_n))}{\partial w_j}\right] + w_j$$

$g'(z) = g(z)(1-g(z))$

$$\frac{\partial h_w(x)}{\partial w_j} = \frac{\partial g(w^T x)}{\partial w_j} = \frac{\partial g(w^T x)}{\partial (w^T x)}\frac{\partial (w^T x)}{\partial w_j} = g(w^T x)(1-g(w^T x))\frac{\partial \sum_j w_j x_j}{\partial w_j}$$

$$= g(w^T x)(1-g(w^T x))x_j = h_w(x)(1-h_w(x))x_j$$

$$\frac{\partial \log h_w(x_n)}{\partial w_j} = \frac{\partial \log h_w(x_n)}{\partial h_w(x_n)}\frac{\partial h_w(x_n)}{\partial w_j} = \frac{1}{h_w(x_n)} h_w(x_n)(1-h_w(x_n))x_j$$

$$= (1-h_w(x_n))x_j$$

$$\frac{\partial \log(1-h_w(x_n))}{\partial w_j} = -h_w(x_n)x_j$$

$$\frac{\partial J}{\partial w_j} = -\sum_{n=1}^{N}(y_n - h_w(x_n))x_{n,j} + w_j$$

$$= \sum_{n=1}^{N}(h_w(x_n) - y_n)x_{n,j} + w_j$$

update rule: $w^{t+1} \Leftarrow w^t - \eta \frac{\partial J(w)}{\partial w}$

$$= w^t - \eta\left[\sum_{n=1}^{N}(h_w(x_n) - y_n)x_n + \sum_i w_i\right]$$

Above is the gradient descent update rule.

(c) The MCCAP estimate is

$$w^* = \arg\min_w[-L(w)] = \arg\min_w[-\log(\prod_{i=1}^{n} P(y_i|w_0 \cdots w_d) f(w_0 \cdots w_d)]$$

$$= \arg\min_w\left[-\log(P(y_i|w_0 \cdots w_d)) - \log(f(w_0 \cdots w_d))\right]$$

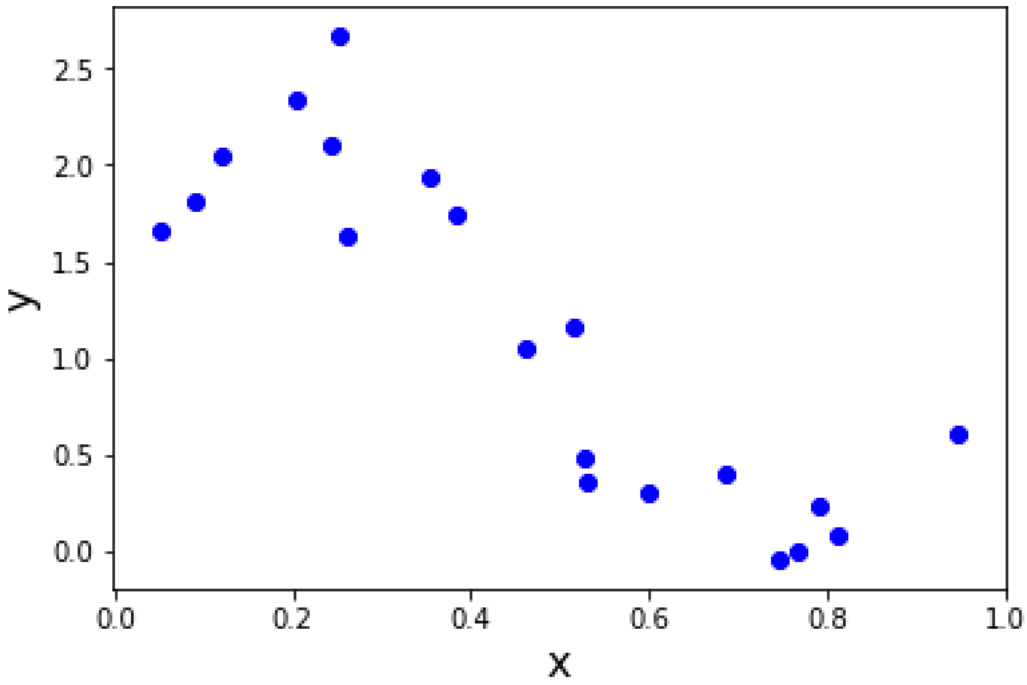
$$f(w_0 \cdots w_d) = \frac{1}{(2\pi)^{\frac{m}{2}}}\exp\left(-\sum_i \frac{w_i^2}{2}\right)$$

$$\log(f(w_0 \cdots w_d)) = -\frac{1}{2}\sum_i w_i^2 + \log\left(\frac{1}{(2\pi)^{\frac{m}{2}}}\right)$$

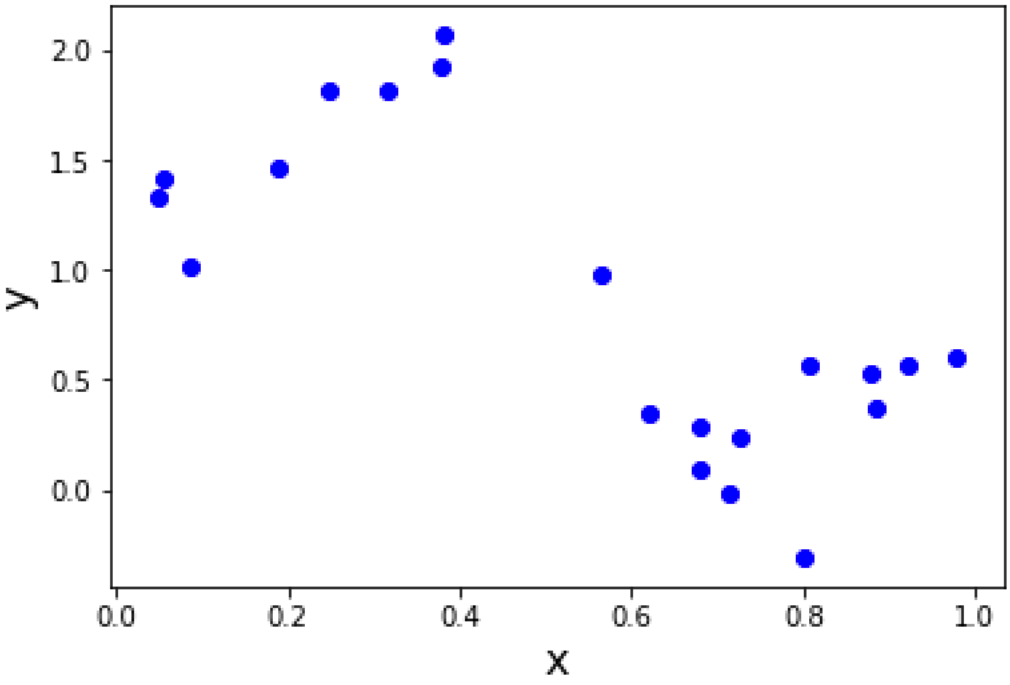
$$\therefore w^* = \arg\min_w -\sum_{n=1}^{N}[y_n \log h_w(x_n) + (1-y_n)\log(1-h_w(x_n))] + \frac{1}{2}\sum w_i^2$$

5.(a)

Visualization for training data:



Visualization for test data:



Neither data set shows a linear relationship. Thus, I think linear regression would not fit the data set well, but polynomial regression might work.

(d)The result I obtained are shown in the table below:

η	0.01	0.0001	0.001	0.0407
number of iterations	821	10000	7481	188
coefficients	[2.44628561 -2.81610856]	[1.91605338 -1.74425507]	[2.44600706 -2.81554509]	[2.44635508 -2.81624955]
Final cost	3.91257648835	5.49161907963	3.91257730388	3.91257642073

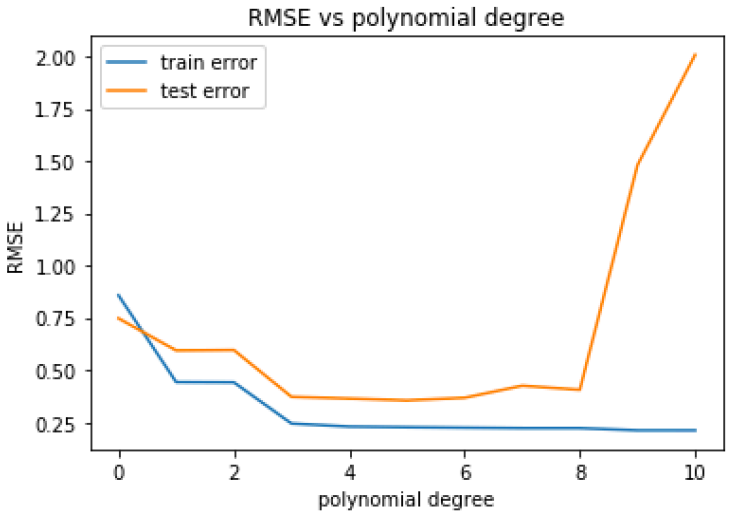
Having a bigger step will lead to a faster converge speed and a smaller number of iterations, and if ends up before maximum iteration, the coefficient and final cost will be similar. Having a small step size will take long time to converge. The result is similar to close-form result.

(e)
coefficient: [2.44640709 -2.81635359]
Final cost: 3.91257640579
It's similar to the result obtained in(d) by using fit_GD. It's faster than GD since it only run one time to get the result. In this case, the closed-form solution is faster than gradient descent.

(f)
number of iterations: 140409
coefficient: [2.44164701 -2.80673099]
final cost: 3.91270361116
Step size decreases when k increases, so it takes too much iterations to run to compute the result since the step becomes rather small when k is large enough. The result is similar to close-form solution.

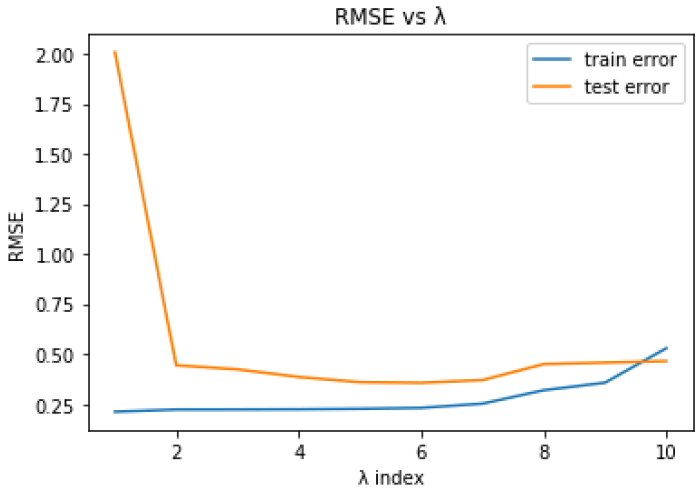
(h) RMSE has following advantages. First, it normalizes by the number of instances, thus it makes data with different set sizes more comparable; Second it's standard deviation of the residuals thus it's on the same scale as the predictions.

(i)



I think polynomial degree=4,5,6 is best fit, since it has a small train error and test error. When degree is larger than 8, the training error increases, which indicates overfitting. When degree is smaller than 3, the error is large for both training error and test error, which indicates underfitting.

(k)



I think λ index=4,5,6 is best fit, which represent $1x10^{-6}$, $1x10^{-5}$, $1x10^{-4}$,respectively. since it has a small train error and test error. When λ is larger than $1x10^{-4}$, the training error increases, which indicates underfitting. When λ is smaller than $1x10^{-6}$, the test error is large while the training error is small, which indicates overfitting.