Homework 2

1. (a) $P(y_1, y_2, y_3, ..., y_n | x_1, x_2, x_3, ..., x_n) = \prod_{n=1}^{N} P(y_n|x_n)$
$\Leftrightarrow ln(\prod_{n=1}^{N} P(y_n|x_n)) = \sum_{n=1}^{N} ln(P(y_n|x_n))$
$= \sum_{n=1}^{N} [\![y_n = +1]\!] ln(h(x_n)) + [\![y_n = -1]\!] ln(1 - h(x_n))$
$= \sum_{n=1}^{N} [\![y_n = +1]\!] - ln(h(x_n)^{-1}) + [\![y_n = -1]\!] - ln((1 - h(x_n))^{-1})$
$= \sum_{n=1}^{N} [\![y_n = +1]\!] - ln\frac{1}{h(x_n)} + [\![y_n = -1]\!] - ln\frac{1}{1-h(x_n)}$
$= -\sum_{n=1}^{N} [\![y_n = +1]\!] ln\frac{1}{h(x_n)} + [\![y_n = -1]\!] ln\frac{1}{1-h(x_n)}$
The maximum likelihood function above is maximized if the term inside the summation is minimized. The term inside the function is the error function in question.

   (b) If $h(x) = \theta(w^T x)$ when $y = 1$, then $1 - h(x) = 1 - \theta(w^T x) = \theta(-w^T x)$ when $y = -1$. Therefore, $h(x) = \theta(y w^T x)$.

$E_{in}(w) = \sum_{n=1}^{N} [\![y_n = +1]\!] ln\frac{1}{h(x_n)} + [\![y_n = -1]\!] ln\frac{1}{1-h(x_n)}$
$= \sum_{n=1}^{N} [\![y_n = +1]\!] ln\frac{1}{\theta(w^T x_n)} + [\![y_n = -1]\!] ln\frac{1}{1-\theta(w^T x_n)}$
$= \sum_{n=1}^{N} [\![y_n = +1]\!] ln\frac{1}{\theta(w^T x_n)} + [\![y_n = -1]\!] ln\frac{1}{\theta(-w^T x_n)}$
$= \sum_{n=1}^{N} ln\frac{1}{\theta(y_n w^T x_n)}$
$= \sum_{n=1}^{N} ln(1 + e^{-y_n w^T x_n})$
minimizing the above function is equivalent to minimizing $\frac{1}{N}$ times the above function

2. $\nabla E_{in}(w) = \frac{\partial}{\partial w}(\frac{1}{N}\sum_{n=1}^{N} ln(1 + e^{-y_n w^T x_n}))$

$= \frac{1}{N}\sum_{n=1}^{N} \frac{e^{-y_n w^T x_n}}{1+e^{-y_n w^T x_n}}(-y_n x_n)$

$= -\frac{1}{N}\sum_{n=1}^{N} \frac{(e^{-y_n w^T x_n})(e^{y_n w^T x_n})}{(1+e^{-y_n w^T x_n})(e^{y_n w^T x_n})}(y_n x_n)$

$= -\frac{1}{N}\sum_{n=1}^{N} \frac{y_n x_n}{1+e^{y_n w^T x_n}}$

$= \frac{1}{N}\sum_{n=1}^{N} \frac{1}{1+e^{y_n w^T x_n}}(-y_n x_n)$

$= \frac{1}{N}\sum_{n=1}^{N} \theta(-y_n w^T x_n)(-y_n x_n)$

Gradient descent: $w(t+1) \leftarrow w(t) - \eta \nabla E_{in}(w)$

$= w(t) - \eta\frac{1}{N}\sum_{n=1}^{N} \theta(-y_n w^T x_n)(-y_n x_n)$

$= w(t) + \frac{\eta}{N}\sum_{n=1}^{N} \theta(-y_n w^T x_n)(y_n x_n)$

In the gradient descent formula above, the movement during gradient descent is directly proportional to $\theta(-y_n w^T x_n)$. If x_n was misclassified, it's probability of being positive will be high when y_n is negative and low when y_n is positive. In other words, $w^T x_n$ will be positive when y_n is negative and $w^T x_n$ will be negative when y_n is positive. This means $y_n w^T x_n < 0$ (or equivalently, $-y_n w^T x_n > 0$) for all misclassified examples.

$\theta(-y_n w^T x_n)$ therefore represents the probability of x_n being misclassified. If x_n is misclassified, $\theta(-y_n w^T x_n)$ will be close to 1. If x_n is not misclassified, $\theta(-y_n w^T x_n)$ will be close to 0. A misclassified example will change the gradient by $\eta \cdot 1 \cdot (y_n x_n)$ while a correctly classified example will change the gradient by $\eta \cdot 0 \cdot (y_n x_n) = 0$.

3. (a) Plots of digit images

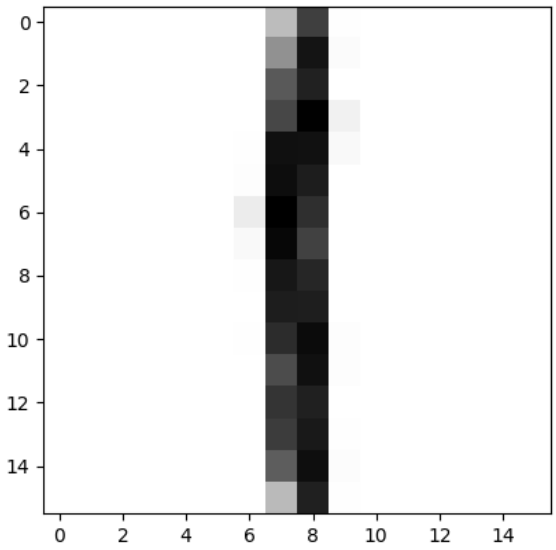

Figure 1: Plot of 1

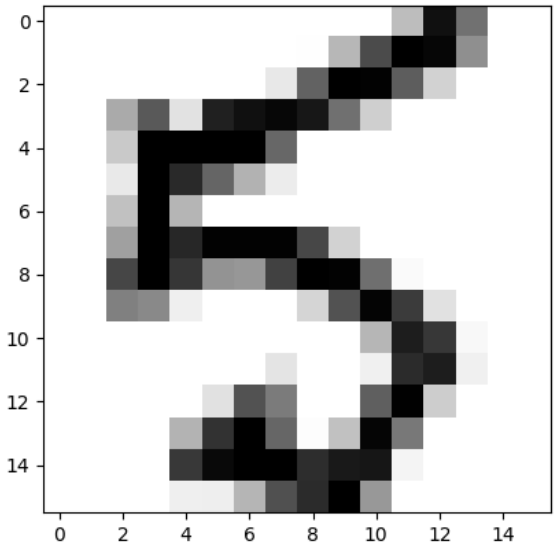

Figure 2: Plot of 5

(b) Feature 1 will be symmetry. I will define a symmetry value of 1 as perfect symmetric over a center vertical line. To calculate symmetry, the image is first split into half down the center vertical line. The right half will then be flipped left-side-right. The two halves will then be subtracted element-wise then squared. The difference in pixel values is squared to keep the sign of the difference positive. This resulting matrix represents the asymmetry of the image. To convert asymmetry to symmetry, each value is subtracted from 4. 4 represents the maximum asymmetry because the maximum difference between the two pixel is 2 from -1 (white) to 1 (black), and the square of 2 is 4. The values of symmetry are averaged then divided by 4 to normalize to the range of 0 to 1.

Feature 2 will be average intensity. Average intensity will be calculated by calculating the mean pixel value.

(c) Plot of features

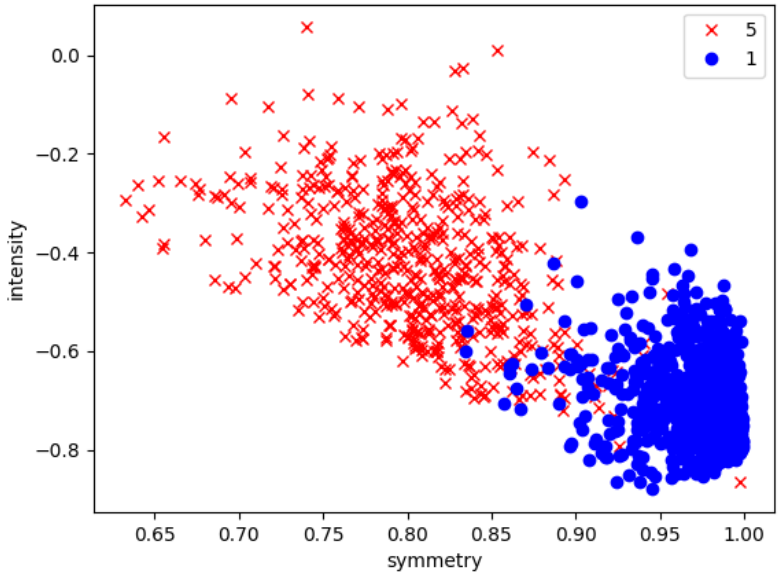

Figure 3: Plot of symmetry and intensity

4. (a) Plots of training and test performance

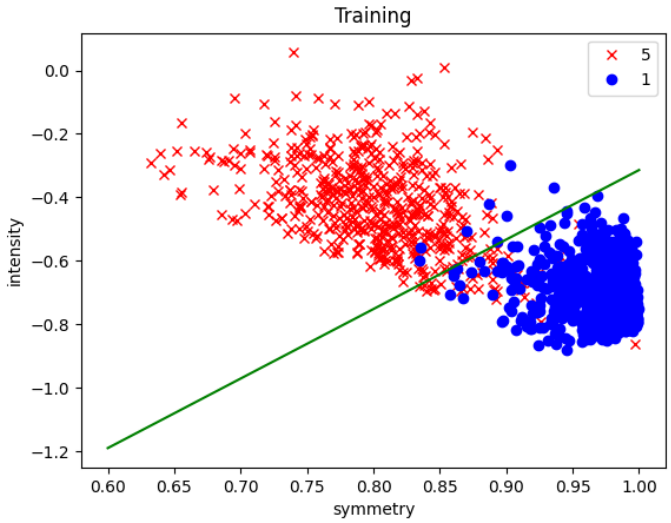

Figure 4: Training data

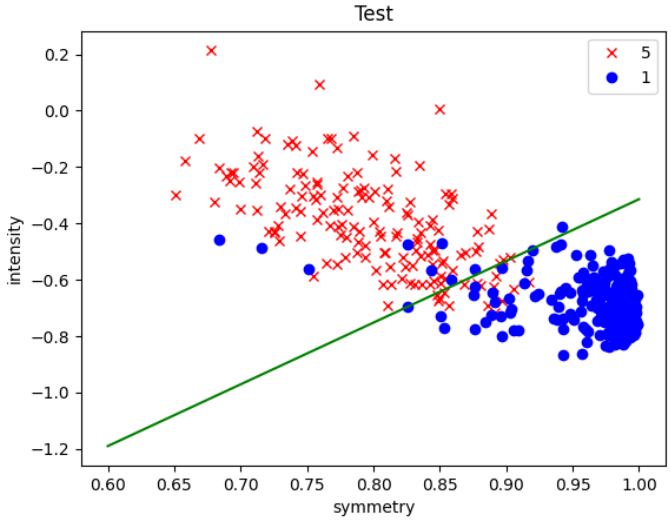

Figure 5: Test data

(b) Performance:
$E_{training} = 0.1134$
$E_{test} = 0.1648$

(c) The optimal value of $\lambda = e^{-44}$ was found by partitioning the training data into 3 groups and cross validating for $\lambda = e^{-100}$ to $\lambda = e^{0}$. The final result was $w = [-42.297, 42.900, -7.063]$ with $E_{training} = 0.0745$ and $E_{test} = 0.161$

(d) $E_{training} = 0.0710$ and $E_{test} = 0.153$ with
$w = [-14.19, -2.99, 3.68, 5.93, -2.01, -0.28, 13.02, -6.57, 2.80, -0.74]$

(e) The training error is better by a factor of 1.6 using 3rd order polynomial transform. However, the testing error does not improve much. I would use linear model without input transformation to save on computation time and resources.