Machine Learning from Data CSCI 4100
Assignment 12
Jae Park (RIN: 661994900)

1. Neural Networks and Back Propagation
(a)

(Identity)
Layer 1:

$$\begin{bmatrix} -0.01938 & -0.01938 \\ -0.01938 & -0.01938 \\ -0.03876 & -0.03876 \end{bmatrix}$$

Layer 2:

$$\begin{bmatrix} -0.1846 \\ -0.1406 \\ -0.1406 \end{bmatrix}$$

(tanh)
Layer 1:

$$\begin{bmatrix} -0.01594 & -0.01594 \\ -0.01594 & -0.01594 \\ -0.03188 & -0.03188 \end{bmatrix}$$

Layer 2:

$$\begin{bmatrix} -0.1518 \\ -0.1156 \\ -0.1156 \end{bmatrix}$$

(b)

(Identity)
Layer 1:

$$\begin{bmatrix} -0.01938 & -0.01938 \\ -0.01938 & -0.01938 \\ -0.03876 & -0.03876 \end{bmatrix}$$

Layer 2:

$$\begin{bmatrix} -0.1845 \\ -0.1406 \\ -0.1406 \end{bmatrix}$$

(tanh)
Layer 1:

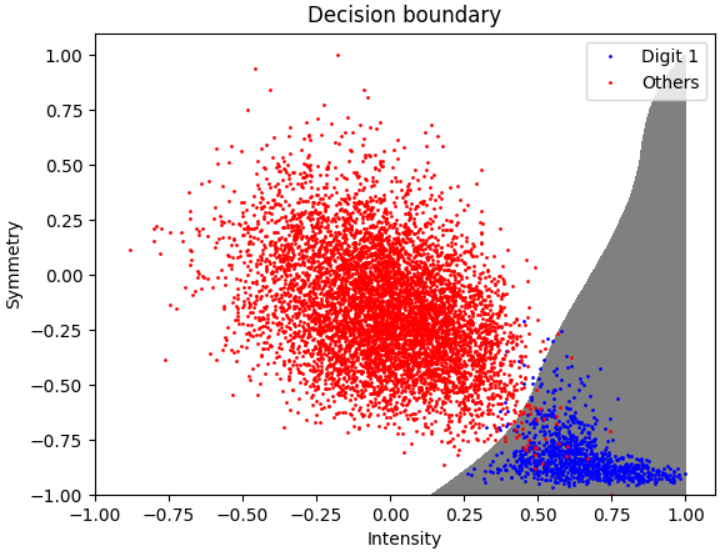
$$\begin{bmatrix} -0.01594 & -0.01594 \\ -0.01594 & -0.01594 \\ -0.03188 & -0.03188 \end{bmatrix}$$

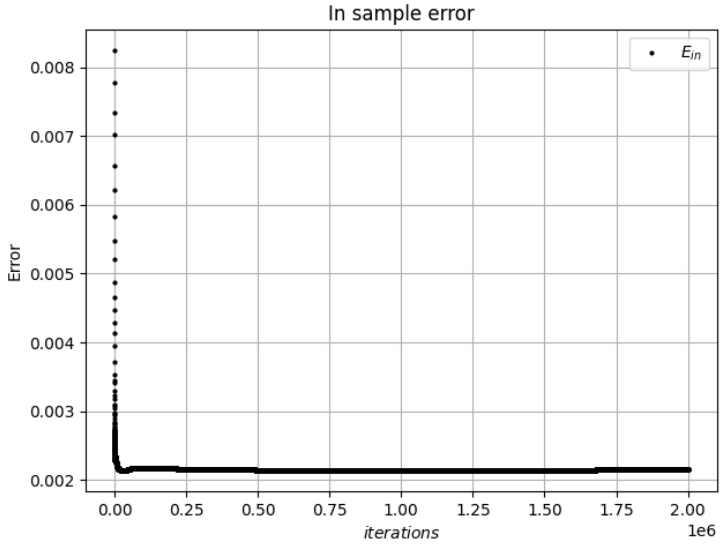
Layer 2:

$$\begin{bmatrix} -0.1518 \\ -0.1156 \\ -0.1156 \end{bmatrix}$$

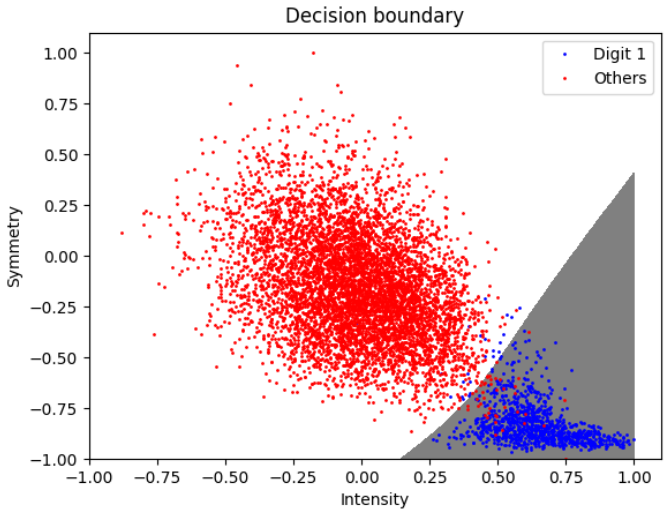
The results from (a) and (b) are essentially identical.

2. Neural Network from Digits
(a)m=10, 2×10^6 iterations,

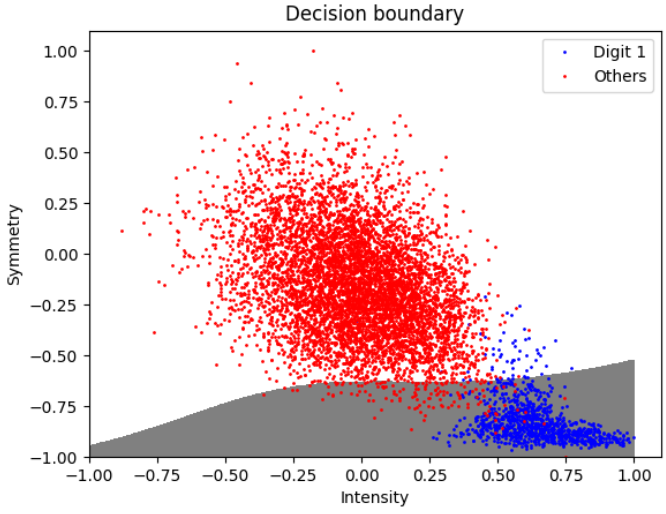




(b) $\lambda = 0.01/N$,



(c) Early stopping,



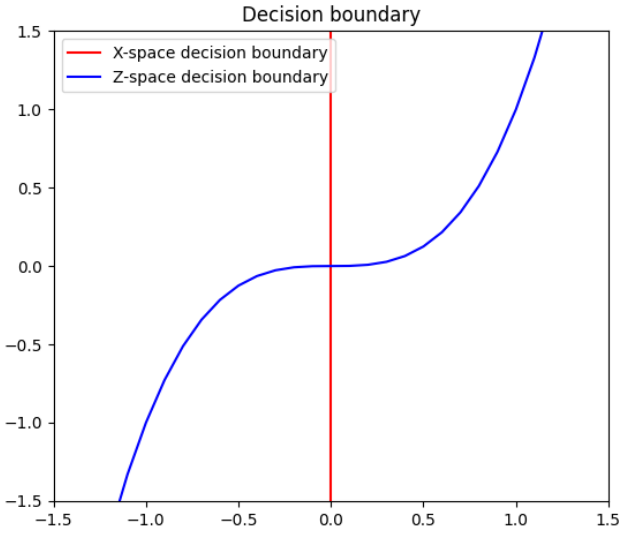
3. Support Vector Machines

(a) With $\mathbf{x_1} = (1, 0), y_1 = +1$ and $\mathbf{x_2} = (-1, 0), y_1 = -1$, the hyperplane that can separate the two with max cushion is the line $x_1 = 0$ that goes through the line segment joining the two points. Therefore, the hyperplane is given as $g(\mathbf{x}) = sign(x_1)$

(b) $\mathbf{z_1} = \begin{bmatrix} 1^3 - 0 \\ 1 \times 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$, $\mathbf{z_2} = \begin{bmatrix} (-1)^3 - 0 \\ -1 \times 0 \end{bmatrix} = \begin{bmatrix} -1 \\ 0 \end{bmatrix}$

Similar to (a), in $\mathbf{z}$ coordinates, the hyperplane is $g(\mathbf{z}) = sign(z_1)$ that cuts across z_1 and z_2. In x coordinates this is equivalent to $g(\mathbf{z}) = sign(x_1^3 - x_2)$.

(c)



(d)

$$K(\mathbf{x}, \mathbf{y}) = \mathbf{z}(\mathbf{x}) \cdot \mathbf{z}(\mathbf{y}) = \begin{bmatrix} x_1^3 - x_2 \\ x_1 x_2 \end{bmatrix} \cdot \begin{bmatrix} y_1^3 - y_2 \\ y_1 y_2 \end{bmatrix} = x_1^3 y_1^3 - x_1^3 y_2 - y_1^3 x_2 + x_2 y_1 + x_1 x_2 y_1 y_2$$

(e) Similar to (a), in the $\mathbf{Z}$-space, the hyperplane is given as

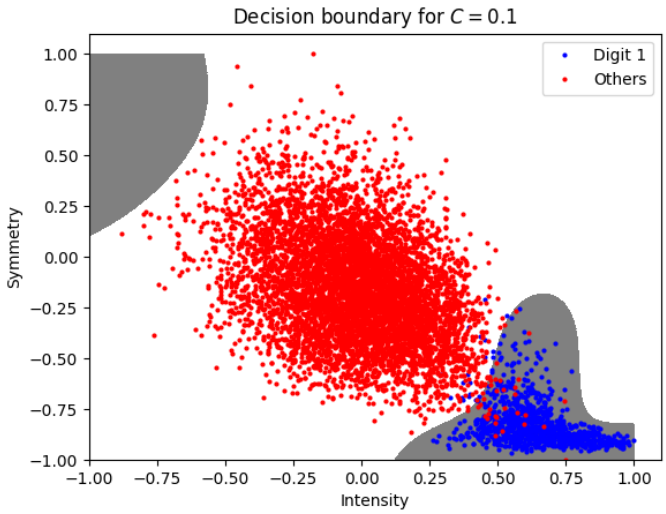
$sign(z_1) = sign(\begin{bmatrix} 1 \\ 0 \end{bmatrix}^T \mathbf{z} + b^*)$ with $b^* = 0$. ... LFD (8.25)

Now, the explicit form in $\mathbf{X}$-space is

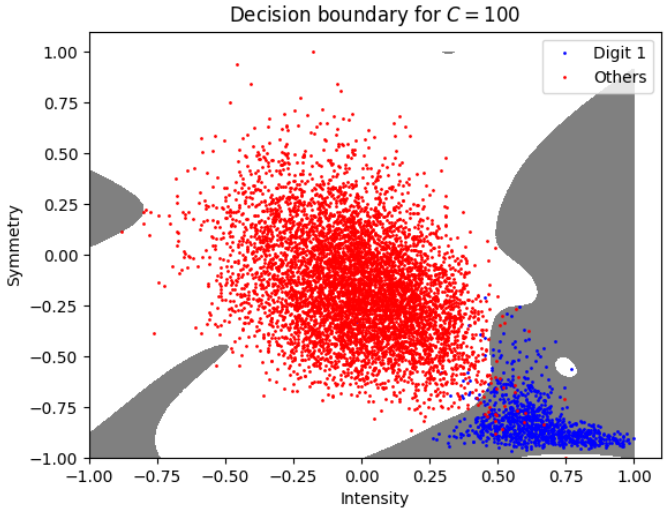
$sign(\begin{bmatrix} 1 \\ 0 \end{bmatrix}^T \mathbf{z}) = sign(\begin{bmatrix} 1 \\ 0 \end{bmatrix}^T \begin{bmatrix} x_1^3 - x^2 \\ x_1 x_2 \end{bmatrix}) = sign(x_1^3 - x^2)$

4. SVM with Digits Data

(a)
$C = 0.1$, $E_{in} = 0.00740$. With small C, E_{in} is relatively large.

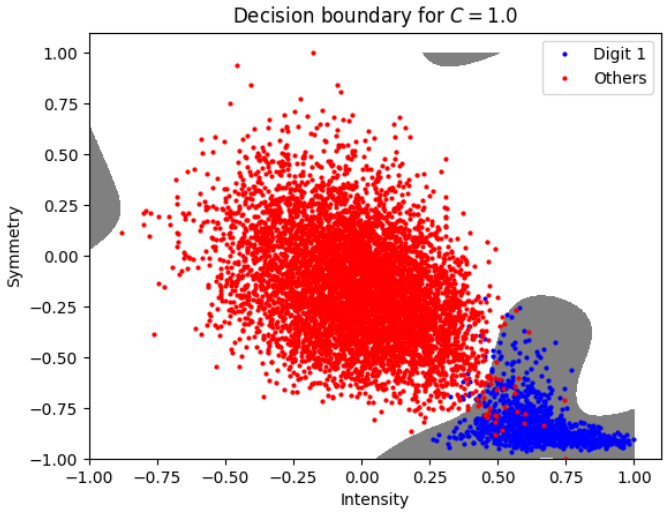


$C = 100$, $E_{in} = 0.00194$. With larger C, E_{in} is relatively small.



(b)
As C gets larger, we expect that the complexity of the decision boundaries increase and the decision boundarys get more fitted to the data. However, as C increases, at some point in C, the decision boundary will be fitted enough such that E_{in} converges to a certain value.

(c)
The minimum $E_{test} = 0.0114598903$ is given when $C = 1$.



5. Compare methods
(i) Regularized linear model: 0.01619
(iv) Neural network: 0.00771
(v) SVM with 8th order polynomial kernel and C selected by CV: 0.01146

$(iv) < (v) < (i)$

Neural Network performed the best, followed by SVM and linear model. It's probably due to its ability to fit the model well to data, when instead SVM and the linear model try to find a hyperplane that separates the data. Cross validation helps with improving the overall performance as it prevents overfitting to a certain data set. Empirically, Neural Net felt a bit faster in terms of running time than SVM. However, NN probably takes up more memory, so that is the trade-off.