CSCI 6100 Homework 12

Abdelrahman Ismael (661847519)

December 10, 2020

1. Neural Networks and Back propagation

(a) Using tanh: $\frac{\partial E_{in}}{\partial W^{(1)}} = \frac{1}{4}\begin{bmatrix} -0.0637662 & -0.0637662 \\ -0.0637662 & -0.0637662 \\ -0.127532 & -0.127532 \end{bmatrix}$, $\frac{\partial E_{in}}{\partial W^{(2)}} = \frac{1}{4}\begin{bmatrix} -0.607334 \\ -0.462542 \\ -0.462542 \end{bmatrix}$

Using Identity: $\frac{\partial E_{in}}{\partial W^{(1)}} = \frac{1}{4}\begin{bmatrix} -0.0775279 & -0.0775279 \\ -0.0775279 & -0.0775279 \\ -0.155056 & -0.155056 \end{bmatrix}$, $\frac{\partial E_{in}}{\partial W^{(2)}} = \frac{1}{4}\begin{bmatrix} -0.738406 \\ -0.562366 \\ -0.562366 \end{bmatrix}$

(b) Obtaining the gradient numerically give almost the same results (with gaps of around 10^{-9} to 10^{-13}).

Using tanh: $\frac{\partial E_{in}}{\partial W^{(1)}} = \frac{1}{4}\begin{bmatrix} -0.0637662 & -0.0637662 \\ -0.0637662 & -0.0637662 \\ -0.127532 & -0.127532 \end{bmatrix}$, $\frac{\partial E_{in}}{\partial W^{(2)}} = \frac{1}{4}\begin{bmatrix} -0.607334 \\ -0.462542 \\ -0.462542 \end{bmatrix}$

Using Identity: $\frac{\partial E_{in}}{\partial W^{(1)}} = \frac{1}{4}\begin{bmatrix} -0.0775279 & -0.0775279 \\ -0.0775279 & -0.0775279 \\ -0.155056 & -0.155056 \end{bmatrix}$, $\frac{\partial E_{in}}{\partial W^{(2)}} = \frac{1}{4}\begin{bmatrix} -0.738406 \\ -0.562366 \\ -0.562366 \end{bmatrix}$

2. Neural Network for Digits

(a) Figure 1 (using log scale for the y-axis) shows the $E_{in}(w)$ (on the vertical axis) versus the number of iterations (on the horizontal axis). The algorithm was run for 2 million iterations in a round 20 minutes. (Sorry, I just noticed that I forgot to add the axis titles)

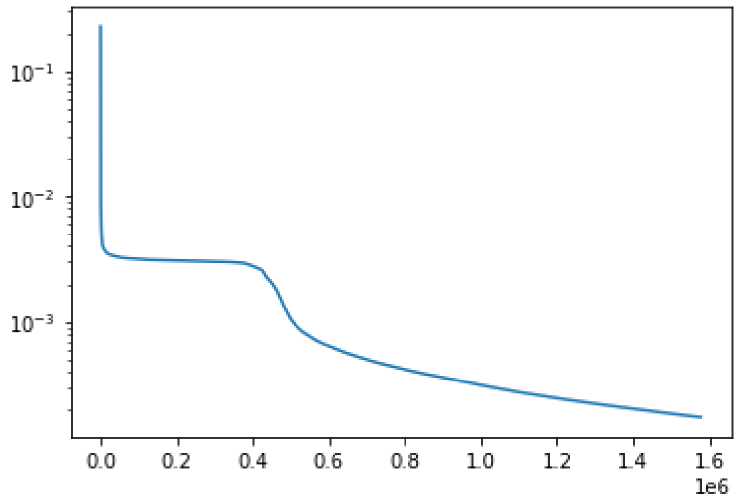

Figure 1: In sample error Vs. number of iterations

Figure 2 shows the obtained decision boundary using the variable learning rate gradient descent after 2 million iterations.

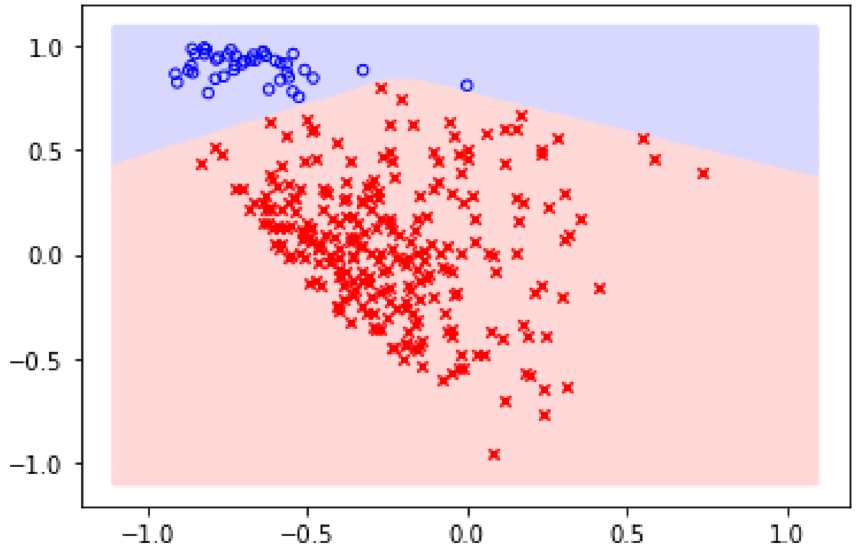

Figure 2: Decision Boundary for the resulting classifier

(b) Figure 3 shows the obtained decision boundary using the variable learning rate gradient descent with weight decay.

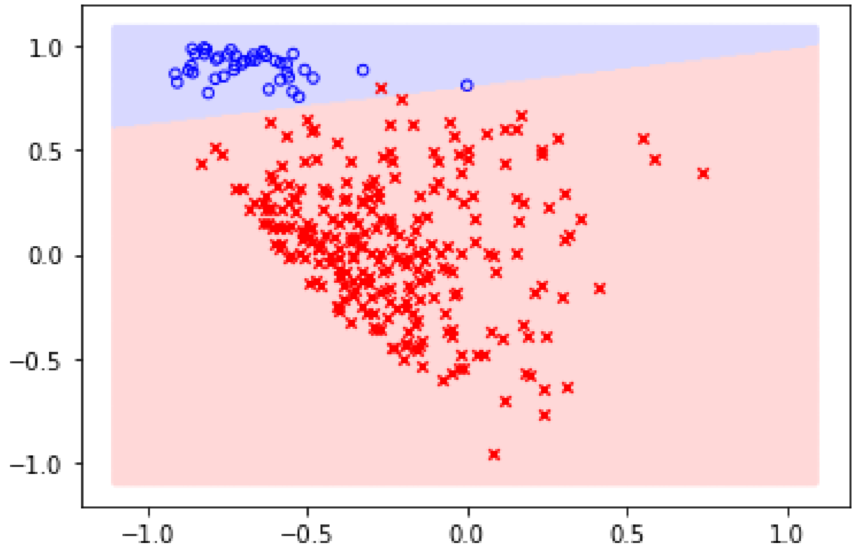

Figure 3: Decision Boundary for the resulting classifier

(c) Figure 4 shows the obtained decision boundary using early stopping with validation set of size 50.

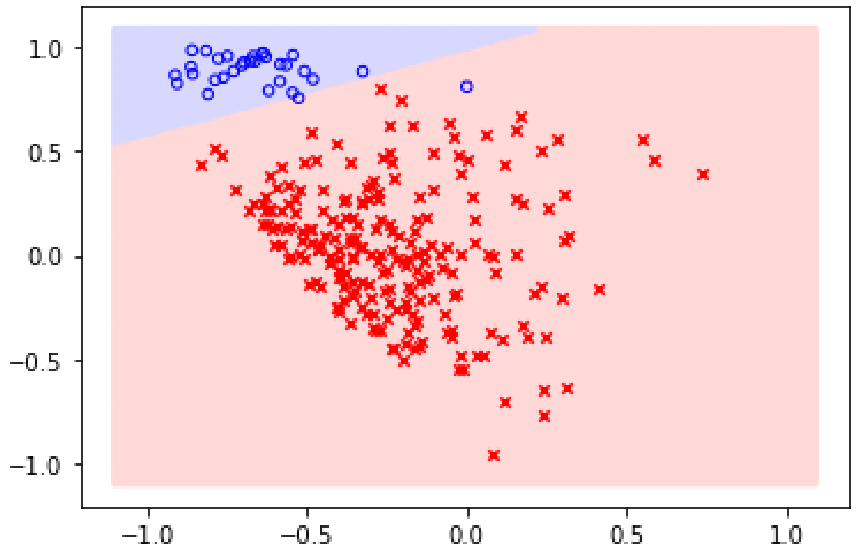

Figure 4: Decision Boundary for the resulting classifier

3. Support Vector Machines

(a) To get the fattest cushion possible, then we need to minimize $\frac{1}{2}w^T w$ subject to $y_n(w^T x_n + b) \geq 1$. So we need $w_1 x_1^{(1)} + w_2 x_2^{(1)} + b \geq 1$ and $-w_1 x_1^{(2)} - w_2 x_2^{(2)} - b \geq 1$, by substituting with pointx_1 and pointx_2 and solving the equations we get $b = w_2 = 0$ and $w_1 = 1$, so $x_1 = 0$ is the equation of the optimal hyperplane.

(b) i. The data points in the Z-space are point $z_1 = [1, 0]$ and point $z_2 = [-1, 0]$.
ii. In the Z-space, the same procedure used in part (a) applies, hence when $z_1 = 0$, that will be the equation of the optimal hyperplane.

(c) Figure 5 shows the obtained decision boundary for the optimal hyperplane in the X-space. While Figure 6 shows the obtained decision boundary for the optimal hyperplane in the X-space after transforming the points and classifying in the Z-space. (Note: Although the hw says in the same plot, I preferred to do them in separate plots to be clear).

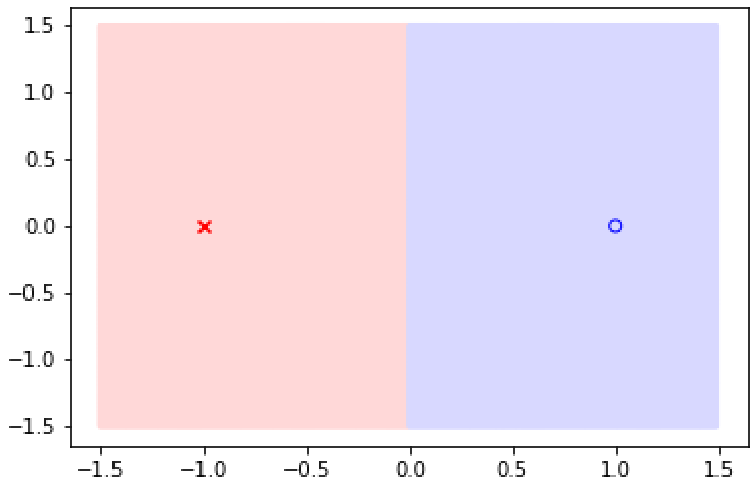

Figure 5: Decision Boundary for the resulting classifier in the X-space

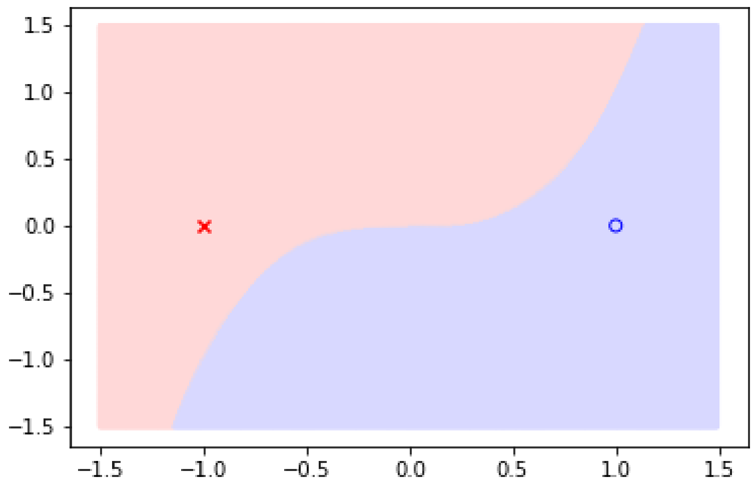

Figure 6: Decision Boundary for the resulting classifier in the X-space after using the Z-space

(d) $K(x, y) = (x_1^3 - x_2)(y_1^3 - y_2) + x_1x_2y_1y_2 = x_1^3y_1^3 - x_1^3y_2 - x_2y_1^3 + x_2y_2 + x_1x_2y_1y_2$.

(e) Classifier functional form is $\text{sign}(x_1^3 - x_2)$.

4. SVM with digits data

(a) Figure 7 shows the obtained decision boundary using a small C of value 0.005.

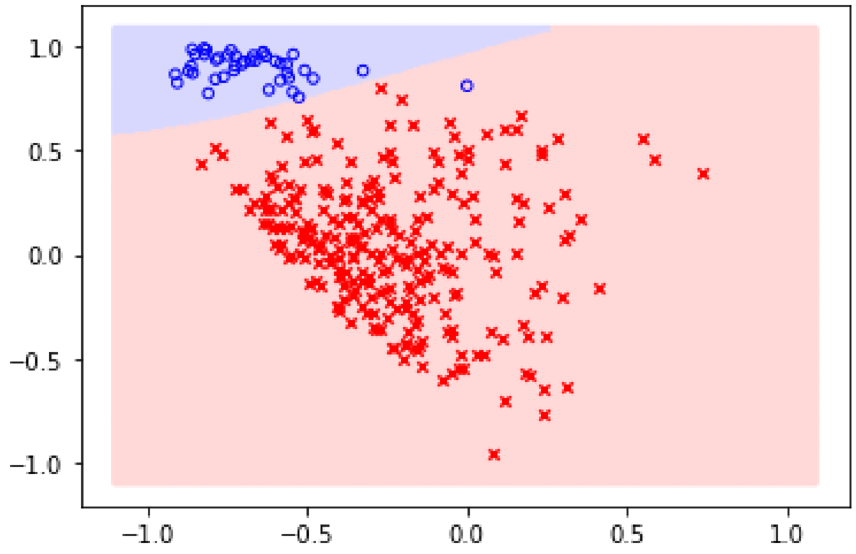

Figure 7: Decision Boundary for the resulting classifier

Figure 8 shows the obtained decision boundary using a large C of value 20.

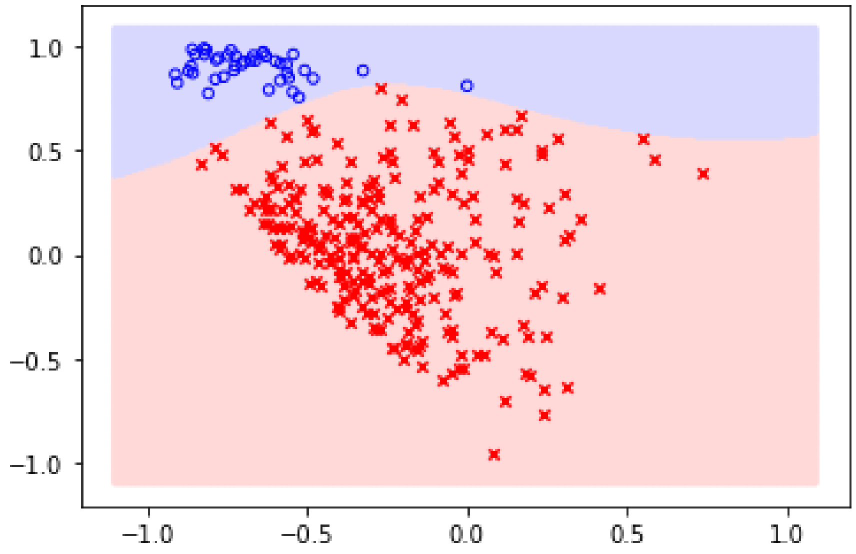

Figure 8: Decision Boundary for the resulting classifier

(b) Looking at the decision boundary for the values of C, that it gets more complex as C goes up.

(c) Using cross validation, Figure 9 shows the obtained decision boundary using a C of value 0.01, the test error was 0.01167.

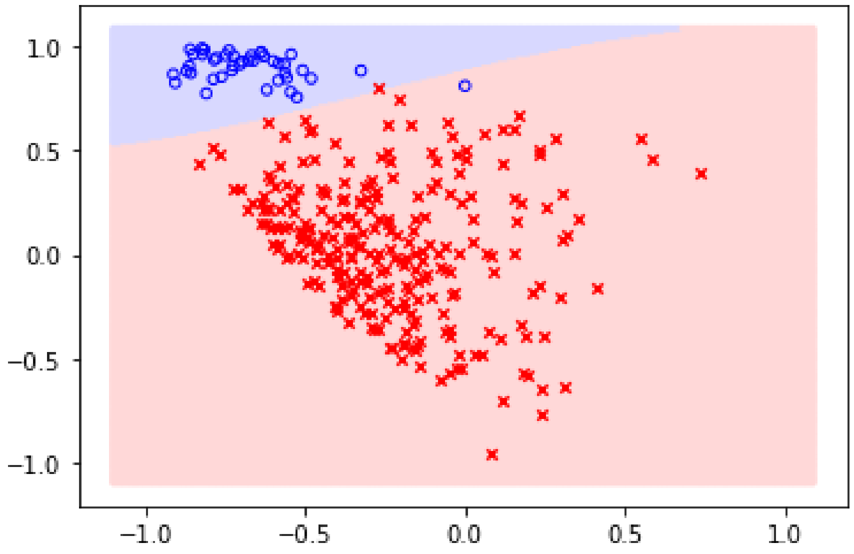

Figure 9: Decision Boundary for the resulting classifier

5. Compare Methods: Linear, k-NN, RBF-network, Neural Network, SVM

i- For Linear model with 8th order polynomial transform and λ selected by CV, $E_{test} = 0.0401411$.
ii- For k-NN rule with k selected by CV, $E_{test} = 0.0096688$.
iii- For RBF-network with number of centers selected by CV, $E_{test} = 0.0115581$.
iv- For Neural network with early stopping, $E_{test} = 0.014114$.
v- For SVM with 8th order polynomial kernel and C selected by CV, $E_{test} = 0.01167$.

Comparing all models, the test error obtained from the K-NN was the least, while the test errors coming from RBF-network, Neural networks and SVM were almost the same. However, I think that Neural Networks or SVM would be my to go models, since they were more flexible and not computationally demanding, especially the SVM which was almost instantaneous with the quadratic programming solver.