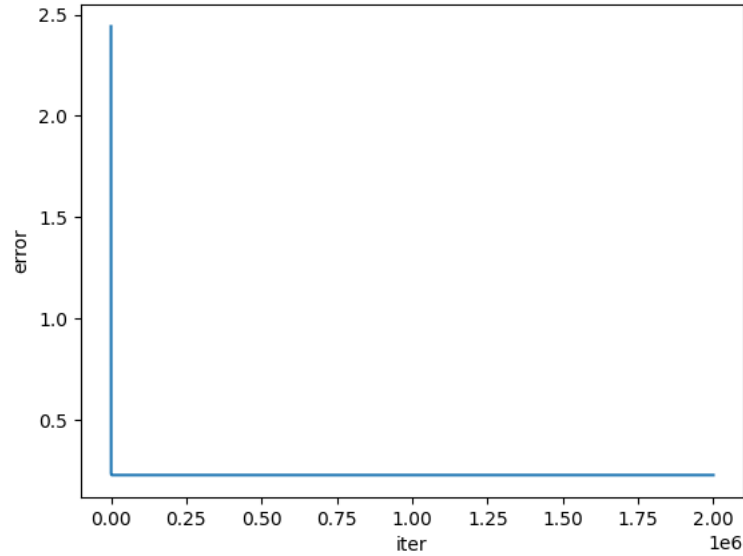


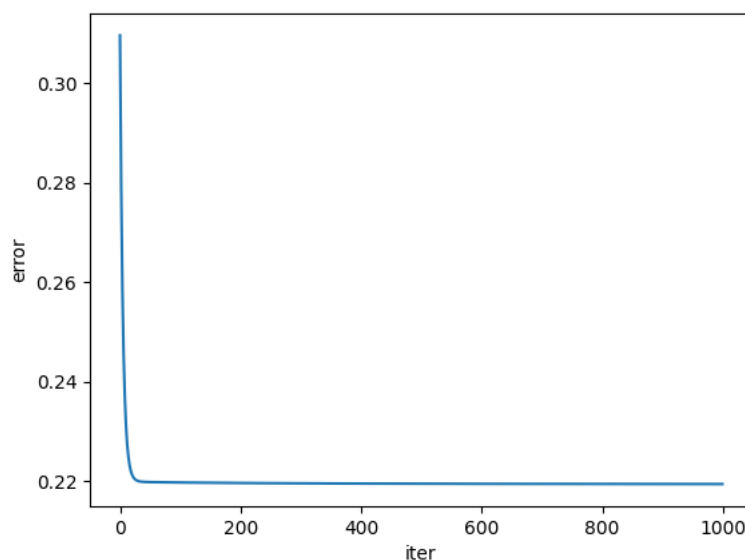
1. (a) For the input point $[2,1]$ and $y = -1$, with weights 0.15 applied. For identity: The first layer returns with results $\begin{bmatrix} 0.2778125 & 0.2778125 \\ 0.13890625 & 0.13890625 \end{bmatrix}$ and layer two returns with results $\begin{bmatrix} 0.47529864 & 0.47529864 \end{bmatrix}$. For tanh: first layer w1: $\begin{bmatrix} 0.27569254 & 0.27569254 \\ 0.13784627 & 0.13784627 \end{bmatrix}$, second layer $\begin{bmatrix} 0.47167168 & 0.47167168 \end{bmatrix}$. The $E_{in}(w)$ obtained was 0.3172898231266975
- (b) After peturbing the weights, I achieved the following: identity w1: $\begin{bmatrix} 0.27796641 & 0.27796641 \\ 0.1389832 & 0.1389832 \end{bmatrix}$ w2: $\begin{bmatrix} 0.47564328 & 0.47564328 \end{bmatrix}$ and tanh: w1: $\begin{bmatrix} 0.2758401 & 0.2758401 \\ 0.13792005 & 0.13792005 \end{bmatrix}$ w2: $\begin{bmatrix} 0.47200485 & 0.47200485 \end{bmatrix}$. $E_{in}(w)$ was 0.31737905364130986

As stated by the hint, after peturbing the weights, the results were relatively similar, only different in the final few digits, therefore we can conclude that our algorithm is correct.

2. (a) After running for 2×10^6 iterations, the following E_{in} versus iterations graph was created:

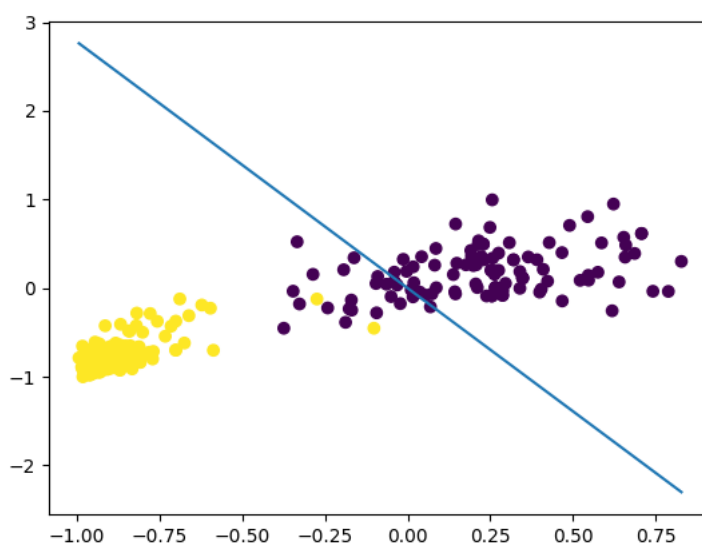


Since it seems difficult to infer the values, I decided to rerun the algorithm with a lower iterations for $iter = 1000$, this was the graph that it plotted:



It seems to converge pretty quickly at *iterations* ≈ 10

I believe that I did something wrong for this neural network to classify the digits because it seems like the decision region is incorrect... However, I added it regardless to show work:

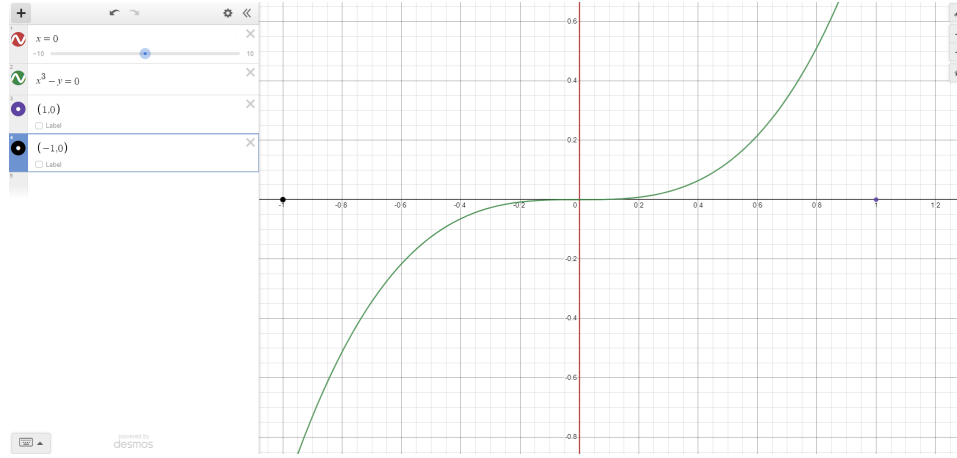



This achieved a $E_{in} = 0.266$, pretty high for the neural network, I was expecting lower.

- (b) Since my algorithm was wrong :(, I decided to focus on the other parts of the homework, it's unfortunately that I couldn't get the algorithm working. However, for the **Why divide iteration # by N**, I believe it is to adjust the learning rate over time.

- (c) omit
- (d) omit
3. (a) We're given two two-dimensional inputs, $x_1 = (1, 0)$ and $x_2 = (-1, 0)$. The optimal separating hyperplane is simply a line. For SVM, we want to find the separator that maximizes the margin. Both of these lines depict a horizontal line with slope 0, the perpendicular bisector is the negative reciprocal of the line, which is a vertical line. A vertical line in $x = 0$ is our solution.

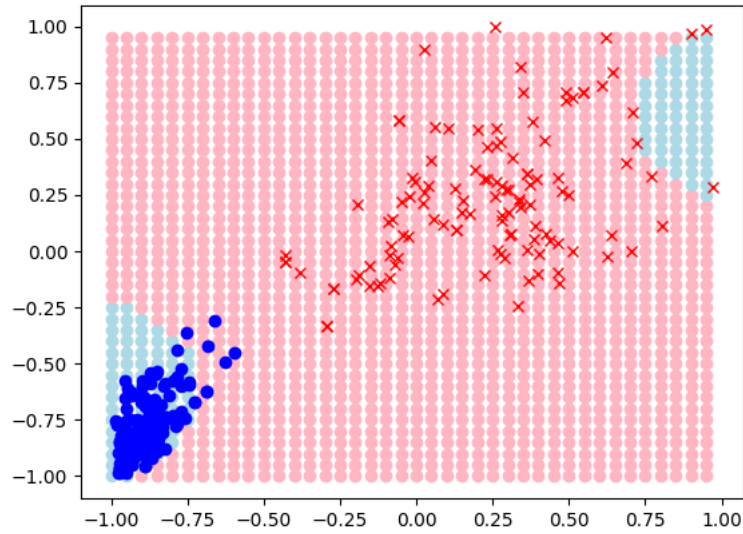
- (b) (i) They are the same point $z_1 = (1, 0)$ and $z_2 = (-1, 0)$
(ii) The same, $z = 0$ and it's a vertical line.



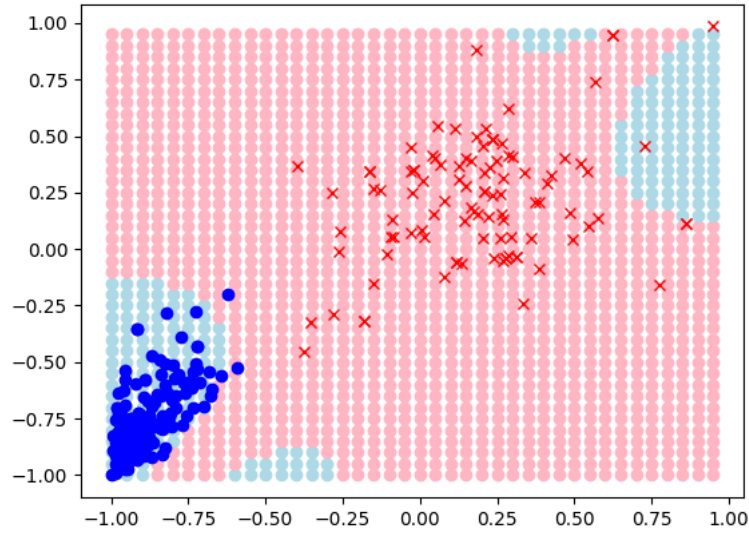
- (c)  desmos
- (d) For $z(x) = \begin{bmatrix} x_1^3 - x_2 \\ x_1 x_2 \end{bmatrix}$, and $z(y) = \begin{bmatrix} y_1^3 - y_2 \\ y_1 y_2 \end{bmatrix}$, we can derive the following:

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

- (e) $h(x) = \text{sign}(x_1^3 - x_2 + x_1 x_2)$
4. (a) For small C , the value $C = 0.01$ was chosen, the following plot was generated with the plotted decision regions:



For large C , the value $C = 10$ was chosen, the decision boundary was:



- (b) For a small C , the complexity was low, we see more of less balanced regions of blues and reds for the decision boundary. When increasing the complexity, we see that the reds in the center have a smaller margin, meaning that the decision boundaries are more complex as also seen by the small bulbs of blue appearing on the sides.
- (c) After running C from 0.01 to 10, the lowest C achieved was at $C = 1.77$, it had a $E_{test} = 0.031738$, the following decision boundary is plotted with the validation set data:

