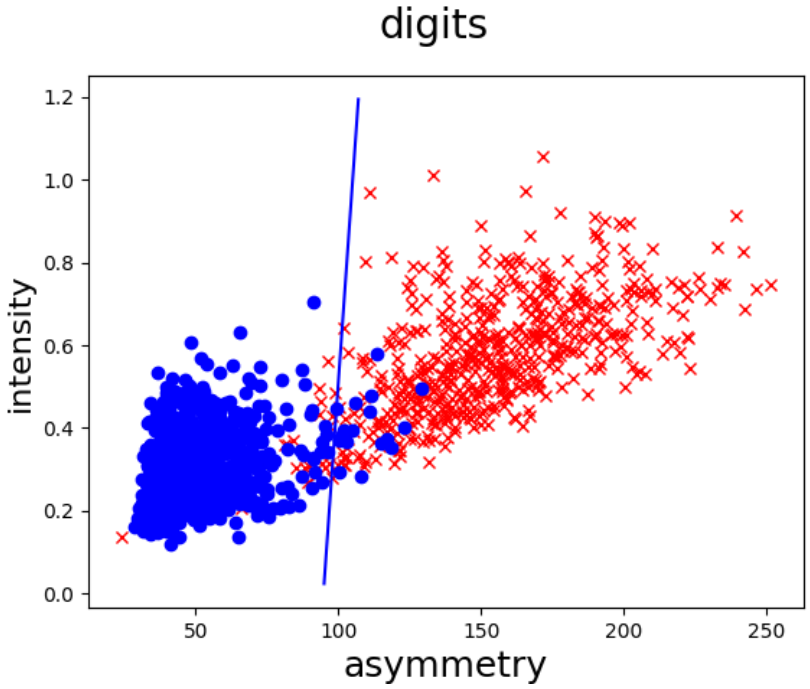
# Machine Learning from Data HW7

Shane O'Brien

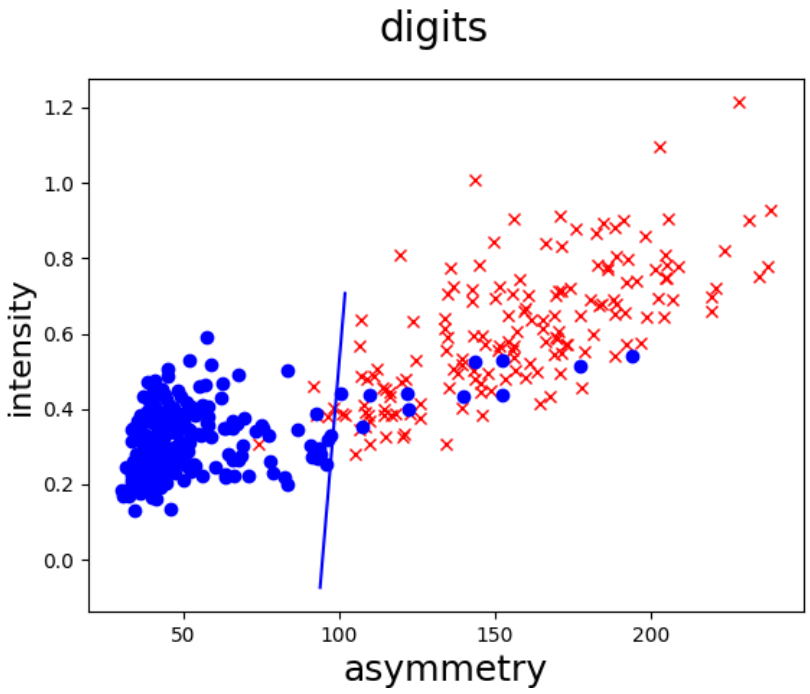
October 2017

## Problem 1

### a




I picked Linear Regression for classification followed by pocket for improvement. I ran 10000 iterations of the PLA. In the image above, you can see the training data with the final $g$ that was achieved.




In the image above, the same line is put on the test data, showing that our learned $g$ is pretty good for classifying our testing data.

**b**

$\mathbf{E}_{in} = 0.0256$ and $\mathbf{E}_{test} = 0.0377$. I calculated this simply by counting how many of the points were incorrectly classified.

**c**

In this case, we have two separate bounds on $\mathbf{E}_{out}$, one from our test set, and one from our training set.

First, let's calculate our bound from the training data. For a linear perceptron in two dimensions, our $d_{vc} = 3$. Then, we have 1621 training points and a $\delta = 0.05$.

$$\mathbf{E}_{out}(g) \leq \mathbf{E}_{in}(g) + \sqrt{\frac{8}{N} ln \frac{4((2N)^{d_{vc}} + 1)}{\delta}}$$

$$\mathbf{E}_{out}(g) \leq \mathbf{E}_{in}(g) + \sqrt{\frac{8}{1621} ln \frac{4((2 * 1621)^3 + 1)}{0.05}}$$

$$\mathbf{E}_{out}(g) \leq 0.0256 + 0.375$$

$$\mathbf{E}_{out}(g) \leq 0.401$$

So let's look at the $\mathbf{E}_{out}$ bound from the test set.

$$\mathbf{E}_{out} \leq \mathbf{E}_{test} + \sqrt{\frac{1}{2N} * ln\frac{2H}{\delta}}$$

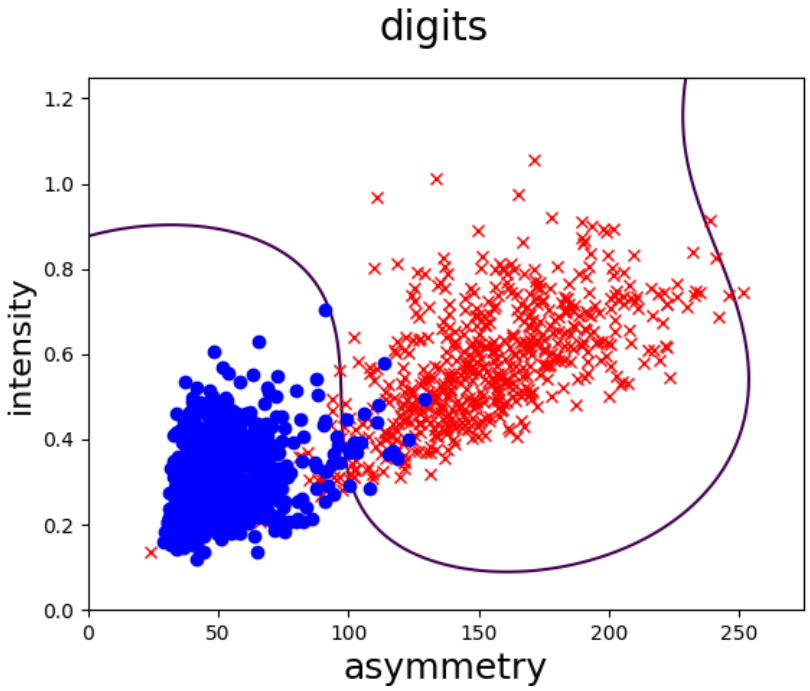
$$\mathbf{E}_{out} \leq \mathbf{E}_{test} + \sqrt{\frac{1}{2 * 424} * ln\frac{2 * 1}{0.05}}$$

$$\mathbf{E}_{out} \leq 0.0377 + 0.0659$$

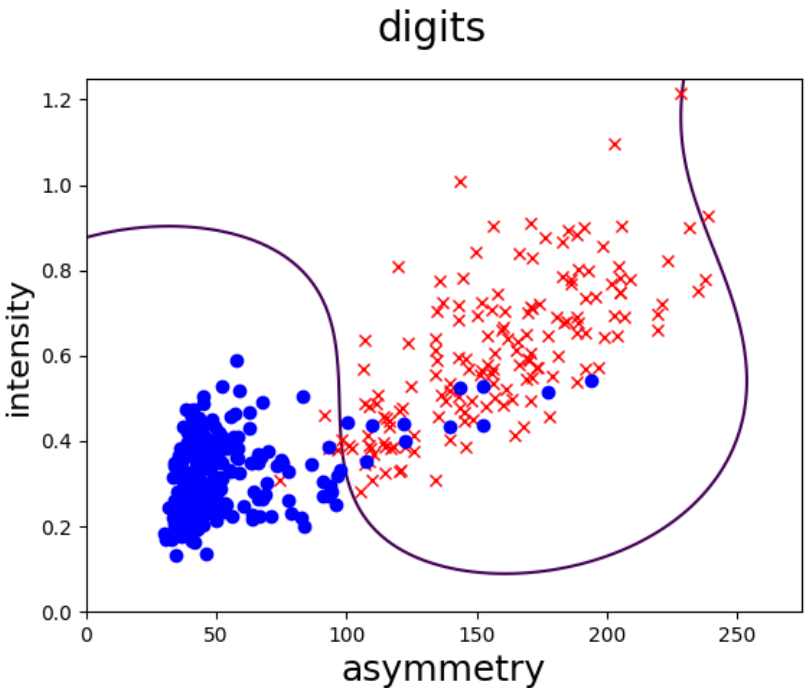
$$\mathbf{E}_{out} \leq 0.103655$$

**d**

I transform the input data from $[1, x, y]$ into $[1, x, y, x^2, xy, y^2, x^3, x^2y, xy^2, y^3]$. Then I used the pseudoinverse method and ran pocket PLA for 10,000 iterations.




Above is the training data run with the algorithm.




Above is the testing data with the result from the training data on top of it. Now, I will calculate the two bounds.

Bound from $E_{in}$ (accounting for $d_{vc} = 10$):

$$\mathbf{E}_{out}(g) \leq \mathbf{E}_{in}(g) + \sqrt{\frac{8}{N} ln \frac{4((2N)^{d_{vc}} + 1)}{\delta}}$$

$$\mathbf{E}_{out}(g) \leq \mathbf{E}_{in}(g) + \sqrt{\frac{8}{1621} ln \frac{4((2 * 1621)^{10} + 1)}{0.05}}$$

$$\mathbf{E}_{out}(g) \leq 0.02306 + 0.6485$$

$$\mathbf{E}_{out}(g) \leq 0.6715$$

Bound from $E_{test}$:

$$\mathbf{E}_{out} \leq \mathbf{E}_{test} + \sqrt{\frac{1}{2N} * ln \frac{2H}{\delta}}$$

$$\mathbf{E}_{out} \leq \mathbf{E}_{test} + \sqrt{\frac{1}{2 * 424} * ln \frac{2 * 1}{0.05}}$$

$$\mathbf{E}_{out} \leq 0.0471 + 0.0659$$

$$\mathbf{E}_{out} \leq 0.113$$

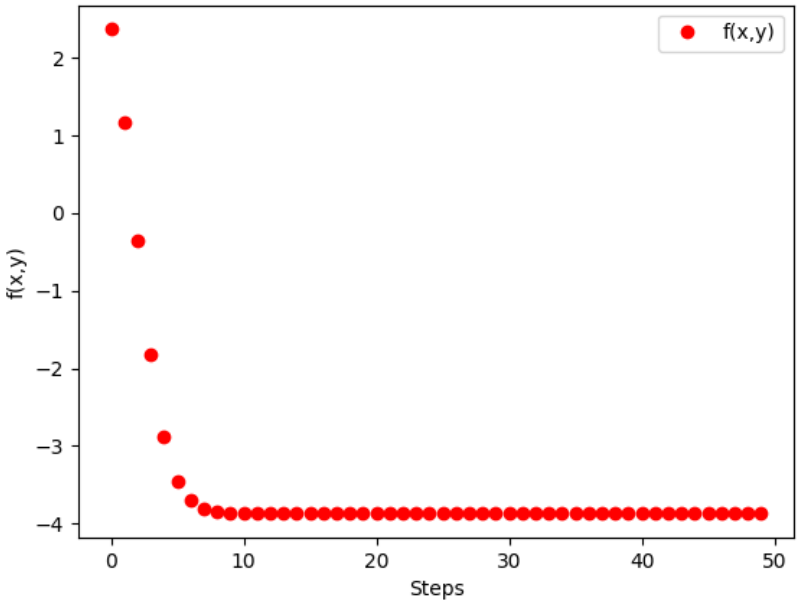
### e

For a client, I would definitely recommend the linear model without the 3rd order transform. Of course, this is assuming we never used the 3rd order transform at all. If you used the 3rd order transform at any point, then you need to account for that.

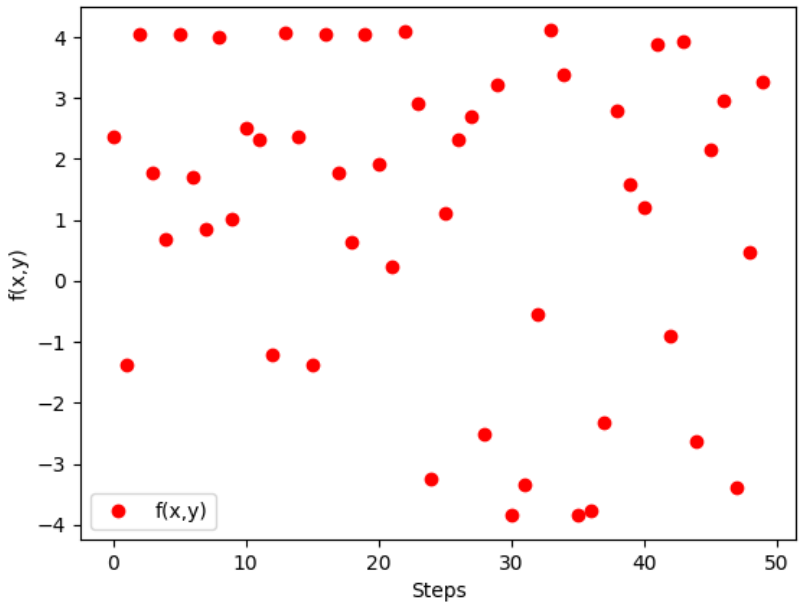
The linear model without the 3rd order transform is the better option here. This is because your $E_{in}$ is very low, and you have a very tight bound. The data (because you picked good features), is very good to separate linearly. If you were to use a 3rd order transform, your $E_{out}$ bound becomes unusable.

## Problem 2

### a




The above graph is when $\eta$ is set to 0.01 and the starting location is (0.1,0.1). This makes sense because the function quickly finds its minimum in 50 iterations.




The above graph is when $\eta$ is set to 0.1 and the starting location is (0.1,0.1). This makes sense because the step is too large. The minimization can't occur because gradient descent keeps overstepping the minimum

### b

Below is a chart of all the $\eta$s and starting positions requested, with the $f(x, y)$ value in the box.

| Starting Position | $\eta = 0.1$ | $\eta = 0.01$ |
|---|---|---|
| (0.1,0.1) | 3.275 | -3.875 |
| (1,1) | -0.485 | -2.875 |
| (0.5,0.5) | -3.340 | -3.875 |
| (-1,-1) | 3.878 | -0.878 |

This illustrates how hard it is to find a true minimum. Our output from gradient descent could change depending on the step distance and the starting position.

## Problem 3.16

### a

The cost of something is the summation of the possible result penalties times the probability of it occurring. In this case, we have defined $g(x)$ as being the proba-

bility of a person being a correct person. So, cost(accept) = $g(x)*0+(1-g(x))c_a$. This simplifies to cost(accept) = $(1-g(x))c_a$. Then, for cost(reject), this is cost(reject) = $g(x)*c_r+(1-g(x))*0$. This simplifies to cost(reject) = $g(x)*c_r$

**b**

We just set $g(x)$ equal to $\kappa$ and cost(accept) = cost(reject). This is because if we want to figure out our threshold, we need to look at the problem when $g(x)$ gives us a value exactly on our threshold. If $g(x)$ gives us a value above or below the threshold, we just go with that option.

$$cost(reject) = cost(accept)$$

$$(1-g(x))c_a = g(x)c_r$$

$$c_a - c_a g(x) = g(x)c_r$$

$$c_a = g(x)c_r + g(x)c_a$$

$$c_a = g(x)(c_r + c_a)$$

$$g(x) = \frac{c_a}{c_r + c_a}$$

**c**

For the supermarket, $c_a = 1$ and $c_r = 10$. This means that

$$\kappa = \frac{c_a}{c_a + c_r}$$

$$= \frac{1}{1+10}$$

$$= \frac{1}{11}$$

For the CIA, $c_a = 1000$ and $c_r = 1$. This means that

$$\kappa = \frac{c_a}{c_a + c_r}$$

$$= \frac{1000}{1000+1}$$

$$= \frac{1000}{1001}$$