LFD Problem Set 7

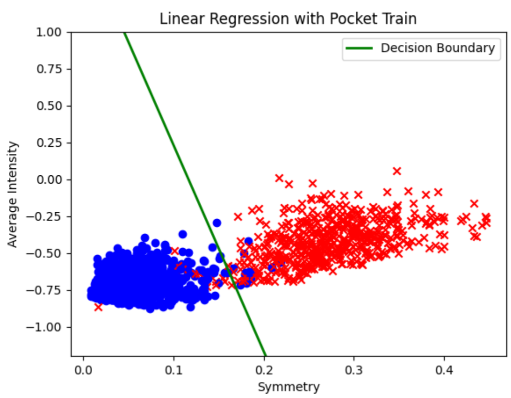
John Cohen

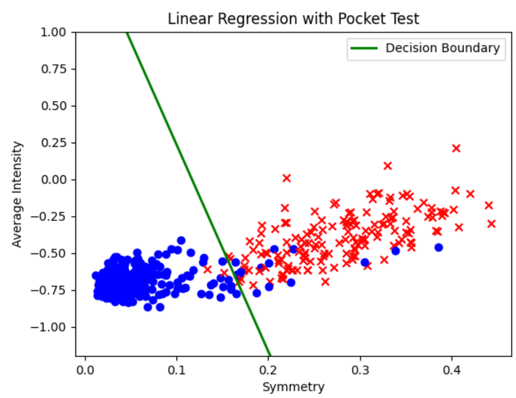
October 28, 2024

Classifying Handwritten Digits: 1 vs. 5

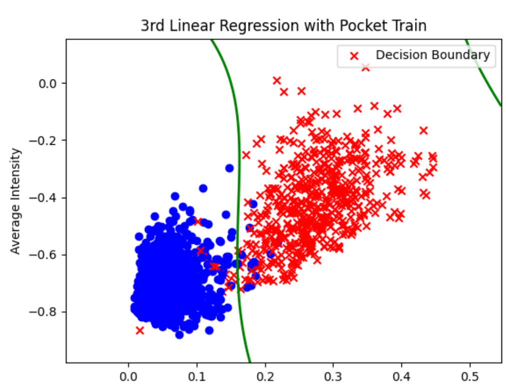
Implement the following classification algorithms for non-separable data:

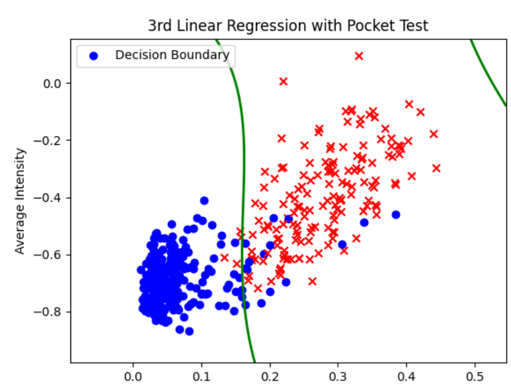
(i) Linear Regression for classification followed by pocket for improvement.





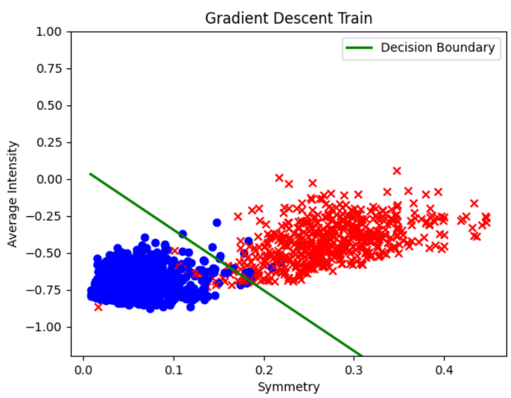
E_{in} : 0.017296604740550 93 E_{test} : 0.03773584905660377

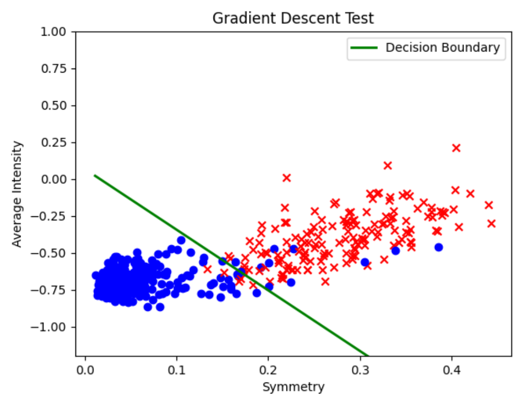




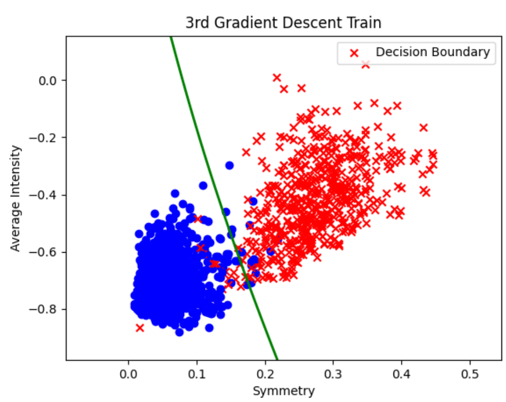
E_{in} : 0.016015374759769378 E_{test} : 0.04481132075471698

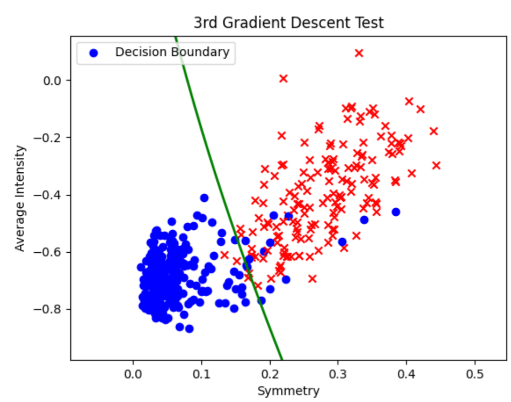
(ii) Logistic regression for classification using gradient descent.





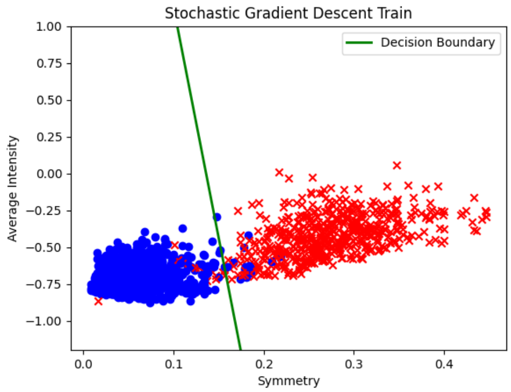
E_{in} : 0.021780909673286355 E_{test} : 0.03773584905660377

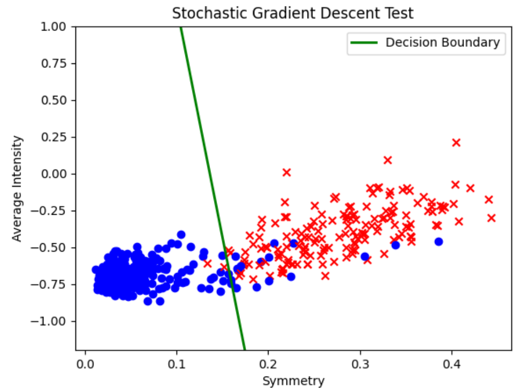




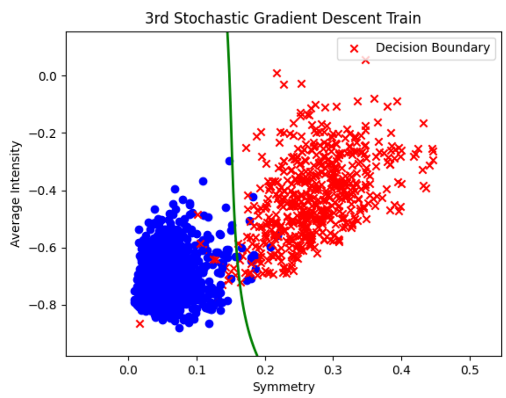
E_{in} : 0.019859064702114 03 E_{test} : 0.04009433962264151

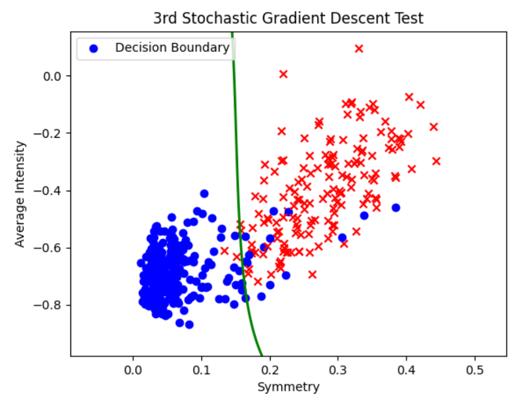
(iii) Logistic regression for classification using stochastic gradient descent.





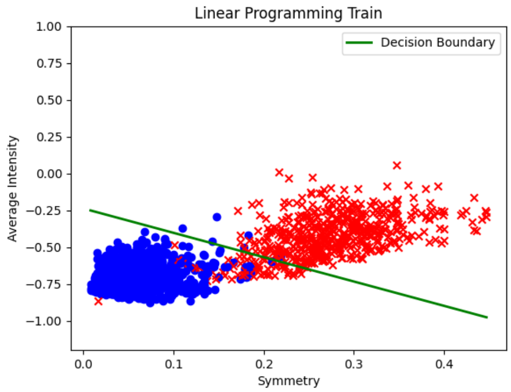
E_{in} : 0.017937219730941704 E_{test} : 0.04245283018867924

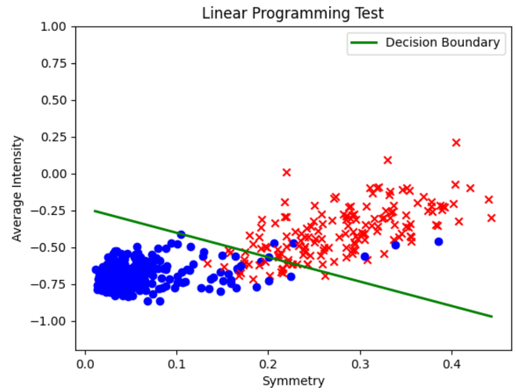




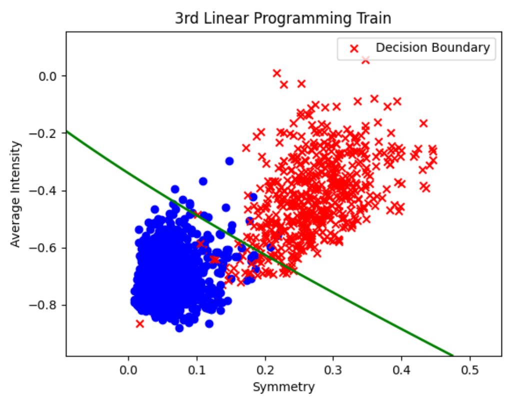
E_{in} : 0.016015374759769378 E_{test} : 0.04009433962264151

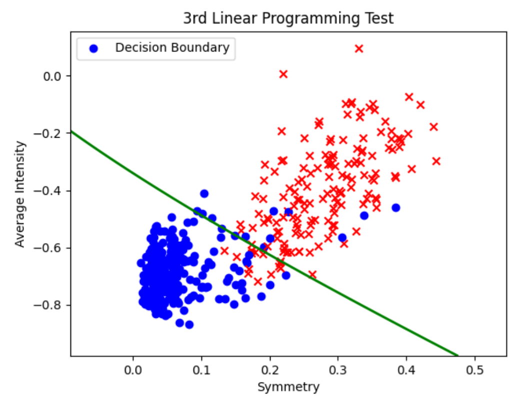
(iv) Linear Programming for classification (Graduate, 6xxx-level, only).





E_{in} : 0.04228058936579116 E_{test} : 0.06839622641509434





E_{in} : 0.035233824471492634 E_{test} : 0.0589622641509434

Use each method to find a separator using the training data and your 2 features from a previous assignment. The target is +1 if the example is a 1 and -1 for a 5.

(a) For each method, plot the separator with the training and test data (separate plots).

See above

(b) For each method, compute E_{in} on the training data and E_{test} on the test data.

See above

(c) Pick the method with the minimum E_{in} and bound the true out-of-sample error using E_{in} and a tolerance $\delta = 0.05$.

1st Order: Linear Regression with Pocket: $E_{out}(g) = 0.01729660474055093 + \sqrt{\frac{8}{1561} ln(\frac{4((2*1561)^3+1)}{0.05})}$

3rd Order: Logistic regression with SGD: $E_{out}(g) = 0.016015374759769378 + \sqrt{\frac{8}{1561} ln(\frac{4((2*1561)^{10}+1)}{0.05})}$

(d) Pick the method with the minimum E_{test} and bound the true out-of-sample error using E_{test} and a tolerance$\delta = 0.05$.

1st Order: Logistic regression Gradient Descent:$E_{out}(g) = 0.03773584905660377 + \sqrt{\frac{8}{424} ln(\frac{4((2*424)^3+1)}{0.05})}$

3rd Order: Logistic regression with SGD: $E_{out}(g) = 0.04009433962264151 + \sqrt{\frac{8}{424} ln(\frac{4((2*424)^{10}+1)}{0.05})}$

(e) Repeat (a)-(d) using a 3rd order polynomial transform.

See above

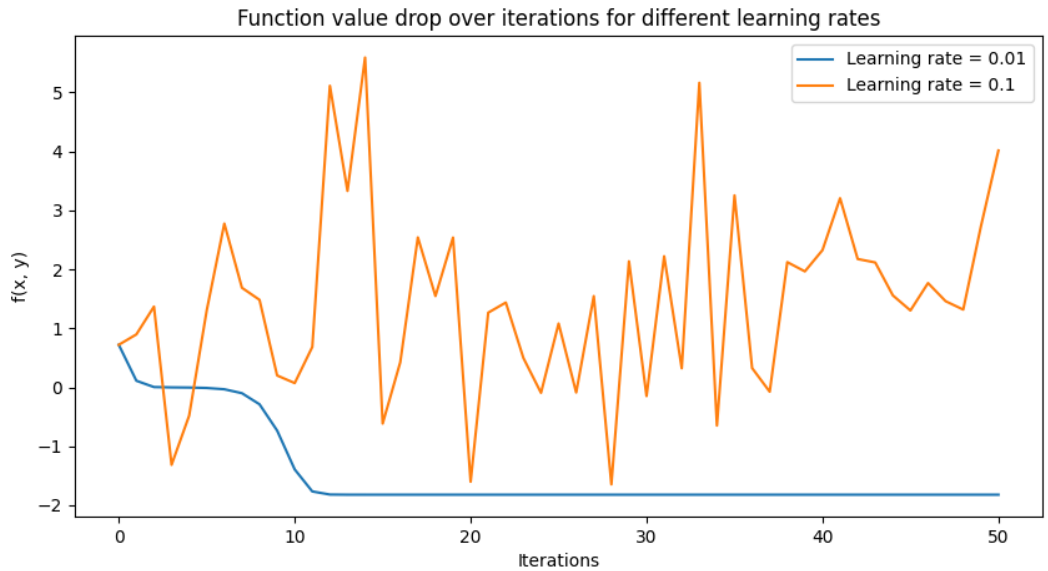
(f) As your final deliverable to a customer, would you use the linear model with or without the 3rd order polynomial transform? Explain.

I would use the non-third order linear model. It had some of the best results for minimizing E_{out} and E_{test} while maintaining a smaller error bar compared to that of the third order polynomial transform. The error bar is largely dependent on the d_{VC} as you can see above. We avoid the problem of over-fitting the data no using it and for the data, a third order transform was not only unnecessary, it was excessive.

Gradient Descent on a "Simple" Function

Consider the function $f(x, y) = x^2 + 2y^2 + 2\sin(2\pi x)\sin(2\pi y)$.

(a) Implement gradient descent to minimize this function. Let the initial values be $x_0 = 0.1$; $y_0 = 0.1$, let the learning rate be $\eta = 0.01$ and let the number of iterations be 50; Give a plot of the how the function value drops with the number of iterations performed. Repeat this problem for a learning rate of $\eta = 0.01$. What happened?



The larger learning rate scattered the points as the step kept overshooting the mark. A smaller η value reduces this possibility as there will be less possible functions it will "over-shoot" on however it will require more iterations. Make the η too large and you will get a series of random points with no sense or indication of convergence as shown above for $\eta = 0.1$ (i.e. the learning rate).

(b) Obtain the "minimum" value and the location of the minimum you get for gradient descent using the same η and number of iterations as in part (a), starting from the following initial points: $(0.1, 0.1), (1, 1), (-0.5, -0.5), (-1, -1)$. A table with the location of the minimum and the minimum values will suffice. You should now appreciate why finding the "true" global minimum of an arbitrary function is a hard problem.

Starting point	Final (x, y)	Minimum value
(0.1, 0.1)	(0.2438, -0.2379)	-1.8201
(1, 1)	(1.2181, 0.7128)	0.5933
(-0.5, -0.5)	(-0.7314, -0.2379)	-1.3325
(-1, -1)	(-1.2181, -0.7128)	0.5933

Problem 3.16

In Example 3.4, it is mentioned that the output of the final hypothesis $g(x)$ learned using logistic regression can be thresholded to get a 'hard' (+-1) classification. This problem shows how to use the risk matrix introduced in Example 1.1 to obtain such a threshold.

Consider finerprint verification, as in Example 1.1. After learning from the data using logistic regression, you produce the final hypothesis

$$g(x) = P[y = +-1|x],$$

which is your estimate of the probability that $y = +-1$. Suppose that the cost matrix is given by

$$TABLE$$

For a new person with fingerprint x, you compute $g(x)$ and you now need to decide whether to accept or reject the person (i.e., you need a hard classification). So, you will accept if $g(x) \geq k$, where k is the threshold.

(a) Define the cost(accept) as your expect cost if you accept the person. Similarly define cost(reject). Show that

$$\text{cost(accept)} = (1 - g(x))c_a,$$
$$\text{cost(reject)} = g(x)c_r.$$

The cost of something is the expected value of that event happening. Thus we have cost(accept) $= E[$accept and cost(reject) $= E[$reject. Remember $g(x) = P[y = +1]$ and $1 - g(x) = P[y = -1]$. $E[\text{accept}] = 0 * P[y = +1] + P[y = -1] * c_a = (1 - g(x))c_a$. $E[\text{reject}] = P[y = +1] * c_r + P[y = -1] * 0 = g(x)c_r$.

(b) Use part (a) to derive a condition on $g(x)$ for accepting the person and hence show that

$$k = \frac{c_a}{c_a + c_r}.$$

If cost of reject is higher than that of accept, accept that person. From part (a); cost(reject) $\geq$ cost(reject) $\implies g(x) * c_r \geq (1 - g(x)) * c_a \implies g(x) * (c_r + c_a) \geq c_a \implies g(x) \geq \frac{c_a}{c_r + c_a}$. With $g(x) \geq k$, we get $k = \frac{c_a}{c_r + c_a}$

(c) Use the cost-matrices for the Supermarket and CIA applications in Example 1.1 to compute the threshold k for each of these two cases. Give some intuition for the thresholds you get.

Supermarket: $c_a = 1$,$c_r = 10$, thus $k = \frac{1}{10+1} = \frac{1}{11}$
CIA: $c_a = 1000$,$c_r = 1$, thus $k = \frac{1000}{1+1000} = \frac{1000}{1001}$
This makes a lot of time. The Supermarket has a lot less risk allowing a false positive however with the CIA the information is classified and thus a false positive (allowing a terrorist to see details threatening national security) is much worse. This comes with much more risk. Therefore, the threshold for $g(x) \geq k$ has to be much higher to prevent false positives / false acceptances.