Machine Learning Coursework 1

Jesus E. Garcia

February 6, 2017

I, Jesus E. Garcia, pledge that this assignment is completely my own work, and that I did not take, borrow or steal work from any other person, and that I did not allow any other person to use, have, borrow or steal portions of my work. I understand that if I violate this honesty pledge, I am subject to disciplinary action pursuant to the appropriate sections of Imperial College London.

1 Problem 1

Given that of the 2,052 people interviewed, 55% supported leaving the EU and 45% supported to stay, we can define:

- X_t as the distribution of the underlying i.i.d. distribution with $X_t \in \{1, 0\}$
- $X_t = 1$ if the decision is to leave
- $X_t = 0$ if the decision is to remain
- Let $a_t = 0$ and $b_t = 1$ then $\Pr(X_t \in [a_t, b_t]) = 1$
- $\mathbb{E}[X_t]$ the expectation of the underlying distribution.
- $\Pr(\mathbb{E}[X_t] < 0.5)$ is the probability that a minority of people vote to leave, or equivalently that a majority vote to stay

With the given data we also have

- $n = 2,052$
- $\sum_{t=1}^{n} X_t = 2,052 * 0.55 = 1128.6$
- $\sum_{t=1}^{n} (b_t - a_t)^2 = n = 2,052$
- $\sum_{t=1}^{n} \mathbb{E}[X_t] = n\,\mathbb{E}[X_t] = 2,052\,\mathbb{E}[X_t]$

We wish to reject the hypothesis that $\mathbb{E}[X_t] \leq 0.5$, or equivalently accept $\mathbb{E}[X_t] > 0.5$ with a confidence interval δ. Hence we wish to find

$$\Pr(\mathbb{E}[X_t] > 0.5) > \delta \tag{1}$$

Or equivalently taking the complement

$$\Pr\left(\mathbb{E}\left[X_t\right] \leq 0.5\right) \leq 1 - \delta \tag{2}$$

Using Hoeffdings inequality:

$$\Pr\left(\sum_{t=1}^{n} X_t - \sum_{t=1}^{n} \mathbb{E}\left[X_t\right] \geq \varepsilon\right) \leq \exp\left(\frac{-2\varepsilon^2}{\sum_{t=1}^{n}(b_t - a_t)^2}\right) \tag{3}$$

Substituting defined values in equation 3 and rearranging:

$$\Pr\left(1128.6 - 2,052\,\mathbb{E}\left[X_t\right] \geq \varepsilon\right) \leq \exp\left(\frac{-2\varepsilon^2}{2,052}\right) \tag{4}$$

$$\Pr\left(\mathbb{E}\left[X_t\right] \leq \frac{1128.6 - \varepsilon}{2,052}\right) \leq \exp\left(\frac{-2\varepsilon^2}{2,052}\right) \tag{5}$$

Finding the value of ε that equates the LHS of equation 2 and equation 5:

$$\frac{1128.6 - \varepsilon}{2,052} = 0.5 \rightarrow \varepsilon = 102.6 \tag{6}$$

We can now equate the RHS of equation 2 and equation 5:

$$1 - \delta = \exp\left(\frac{-2 * 102.6^2}{2,052}\right) \rightarrow \delta = 0.99996 \tag{7}$$

We can therefore reject the probability that a majority of people will vote to remain with a 99.996% confidence interval.

2 Problem 2

2.1 Part a

Given the definition $\rho = \min_{i \in \{1,..,n\}} y^{(i)} w_*^T x^{(i)}$ we can show that $\rho > 0$ by contradiction, proving that $\rho = 0$ and $\rho < 0$ are not possible by contradiction with the given definitions.

$$\begin{aligned}
w_* = w_{t-1} \wedge \rho = 0 \wedge y \in \{1, -1\} &\rightarrow \exists i. w_*^T x^{(i)} = 0 \\
&\rightarrow \exists i. sign(y^{(i)} w_*^T x^{(i)}) = 0 \\
&\rightarrow \exists (t-1) \in \mathcal{D}. sign(w_{t-1}^T x_{t-1}) \neq sign(y_{t-1}) \\
&\rightarrow w_t = w_{t-1} + y_{t-1} x_{t-1} \\
&\rightarrow w_* \neq w_{t-1} \\
&\perp
\end{aligned}$$

$$
\begin{aligned}
w_* = w_{t-1} \wedge \rho > 0 &\rightarrow \exists i. y^{(i)} w_*^T x^{(i)} < 0 \\
&\rightarrow \exists i. sign(w_*^T x^{(i)}) \neq sign(y^{(i)}) \\
&\rightarrow \exists (t-1) \in \mathcal{D}. sign(i.w_t - 1^T x_{t-1}) \neq sign(y_{t-1}) \\
&\rightarrow w_t = w_{t-1} + y_{t-1} x_{t-1} \\
&\rightarrow w_* \neq w_{t-1} \\
&\perp
\end{aligned}
$$

2.2 Part b

We would like to proof that:

$$w_*^T w_t \geq t\rho \tag{8}$$

We will first proof that:

$$w_*^T w_t \geq w_*^T w_{t-1} + \rho \tag{9}$$

The proof follows from the recursive definition of w_{t-1} in the algorithm and the definition of ρ:

$$
\begin{aligned}
w_*^T (w_{t-1} + y_{t-1} x_{t-1}) &\geq w_*^T w_{t-1} + \rho \\
w_*^T y_{t-1} x_{t-1} &\geq \rho \\
w_*^T y_{t-1} x_{t-1} &\geq \min_{i \in \{1,..,n\}} y^{(i)} w_*^T x^{(i)} \\
y_{t-1} x_{t-1} &\geq \min_{i \in \{1,..,n\}} y^{(i)} x^{(i)}
\end{aligned}
$$

We can now proof equation 8 by induction.

Base case:

$$
\begin{aligned}
t = 0 \wedge w_0 = 0 \rightarrow w_*^T 0 &\geq 0\rho \\
&\rightarrow 0 \geq 0
\end{aligned}
$$

Recursion case: assuming $w_*^T w_k \geq k\rho$ holds for a k want to prove that $w_*^T w_{k+1} \geq (k+1)\rho$ holds:

$$
\begin{aligned}
w_*^T w_{k+1} &\geq w_*^T w_k + \rho \\
&\geq k\rho + \rho
\end{aligned} \tag{10}
$$

Equation 10 holds from the assumption on the recursion and the proof of equation 9.

2.3 Part c

We would like to proof that:

$$||w_t||^2 \leq t\mathcal{R}^2 \tag{11}$$

We will first proof that:

$$||w_t||^2 \geq ||w_{t-1}||^2 + ||x_{t-1}||^2 \tag{12}$$

The proof follows from the recursive definition of w_{t-1} in the algorithm and that $\forall t > 0.y_t^2 = 1 \wedge sign(y_t w_t^T x_t) = 1$ by the algorithm's definition:

$$\begin{aligned}
||w_t||^2 &= w_t w_t^T \\
&= (w_{t-1} + y_{t-1}x_{t-1})(w_{t-1} + y_{t-1}x_{t-1})^T \\
&= (w_{t-1} + y_{t-1}x_{t-1})(w_{t-1}^T + y_{t-1}x_{t-1}^T) \\
&= w_{t-1}w_{t-1}^T + y_{t-1}^2 x_{t-1}x_{t-1}^T + y_{t-1}w_{t-1}^T x_{t-1} + y_{t-1}w_{t-1}x_{t-1}^T \\
&= ||w_{t-1}||^2 + 1 * ||x_{t-1}||^2 + 2 * y_{t-1}w_{t-1} \cdot x_{t-1} \\
&\leq ||w_{t-1}||^2 + ||x_{t-1}||^2
\end{aligned}$$

We can now proof equation 11 by induction that where $\mathcal{R} = \max_{i \in \{1,\ldots,n\}} ||x^{(i)}||$.

Base case:

$$\begin{aligned}
t = 0 \wedge ||w_0|| = 0 &\rightarrow 0^2 \geq 0\mathcal{R} \\
&\rightarrow 0 \geq 0
\end{aligned}$$

Recursion case: assuming $||w_k||^2 \leq k\mathcal{R}^2$ holds for a k and having proven equation 12, want to prove $||w_{k+1}||^2 \leq (k+1)\mathcal{R}$

$$\begin{aligned}
||w_{k+1}||^2 &\leq ||w_k||^2 + ||x_k||^2 \\
&\leq ||w_k||^2 + R^2 \\
&\leq kR^2 + R^2 \\
&= (k+1)R^2
\end{aligned} \tag{13}$$

Equation 13 holds from the assumption on the recursion, the proof of equation 12 and the fact that $\forall k.||x_k|| \leq \mathcal{R}$ by definition of $\mathcal{R}$.

2.4 Part d

We would like to proof that:

$$\frac{||w_*||^2\mathcal{R}^2}{\rho^2} \geq t \tag{14}$$

The proof follows by applying the Cauchy-Schwarz inequality to the LHS of the proven equation 8:

$$||w_*||||w_t|| \geq w_*^T w_t \geq t\rho \tag{15}$$

We continue by proving that the relation $||w_*^T||||w_t|| \geq t\rho$ from equation 15 holds by multiplying both sides by $||w_*^T||||w_t||$ and applying the proven relation in equation 11:

$$||w_*||^2 t\mathcal{R}^2 \geq ||w_*||^2||w_t||^2 \geq t\rho||w_*||||w_t|| \tag{16}$$

Applying the Cauchy-Schwarz inequality to the rightmost side of equation 16 and substituting again the proven equation 8:

$$||w_*||^2 t\mathcal{R}^2 \geq t\rho||w_*^T||||w_t|| \geq t\rho w_*^T w_t \geq t^2\rho^2 \tag{17}$$

Simple algebra on the inequality's rightmost and leftmost side finish the proof for equation 14:

$$||w_*||^2 t\mathcal{R}^2 \geq t^2\rho^2 \rightarrow \frac{||w_*||^2\mathcal{R}^2}{\rho^2} \geq t \tag{18}$$

3 Problem 3

Problem 3 will cover implementing the perceptron algorithm in both synthetic and real data sets. In doing so different aspects of the algorithm, such as the error it produces for different data sets, will be explored and explained.

3.1 Part a

The first implementation of the perceptron algorithm gives the weights for a line separating two sets of linearly separable data. Figure 1 shows the outcome of running the algorithm for different sizes of datasets with points uniformly distributed in the unit square.

The algorithm follows the given perceptron definition and will therefore not finish if the data is not linearly separable. Hence it is important to consider the generation of synthetic data. The code provided allows to generate data independently for each class or to generate data of both classes simultaneously. Either way any sampling distribution function can be provided. It is worth noting however that if the two classes are generated separately, the resulting overall distribution might not be as expected. Taking as an example the line used in Figure 1, if instead of generating n samples in total for both classes we were to generate $n/2$ samples for each class, the overall distribution would not be uniformly distributed. This is so because there would be the same number of points in the 2 areas defined by the line, which do not have the same size.

The algorithm does not specify which is the next point to choose with which to update the weights. Therefore this implementation chooses the first point it finds to be wrong, by testing them in the order given. This was chosen so as to not to have to classify all points for every iteration. If choosing a certain point

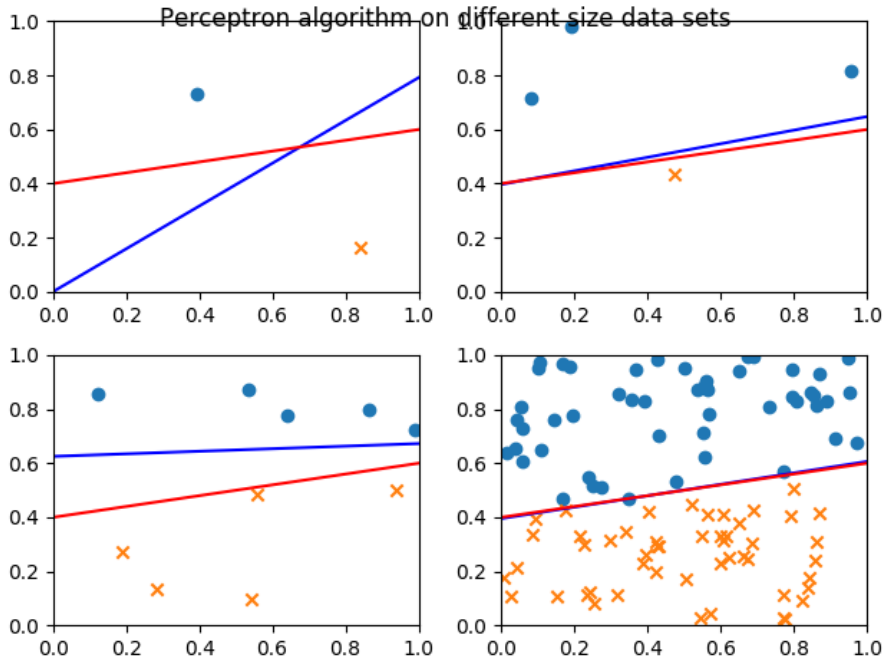


Figure 1: Close-up of a gull

makes the weights converge faster this might not be the fastest solution, but it is still correct.

3.2 Part b

(1) The error for any pair of class separating and perceptron output lines can be calculated as the probability that a given new point falls in the area between the two lines. Since we are dealing with a uniform distribution, this will be the ratio of error area to the total area, which in the unit square is 1. We can therefore define the error through a closed form formula.

Let $a_i x + b_i = y$ be a separator line, where a point (x_p, y_p) is classified as 1 if $a_i x_p + b_i > y_p$ and as -1 otherwise. The maximum values of x and y are also constrained to the interval $[0, 1]$ Then the area between two given lines with parameters a_1, b_1, a_2, b_2 can be found as the area between the bounds defined by y and the limits in x.

To calculate the area the integral must be split into the half before the intersection and the half after the intersection. The intersection can be found from the system of equations:

$$
\begin{aligned}
a_1 x_{int} + b_1 &= y_{int} \\
a_2 x_{int} + b_2 &= y_{int}
\end{aligned} \tag{19}
$$

$$
a_2 b_1 - a_1 b_2 = (a_2 - a_1) y_{int} \rightarrow y_{int} = \frac{a_2 b_1 - a_1 b_2}{a_2 - a_1} \tag{20}
$$

$$
x_{int} = \frac{y_{int} - b_1}{a_1} = \frac{b_1 - b_2}{a_2 - a_1} \tag{21}
$$

We can now take the integral that defines the area for any uniform distribution as:

$$
\begin{aligned}
area = &\int_{x_{min}}^{x_{int}} \int_{y=a_1 x + b_1}^{y=a_2 x + b_2} \frac{1}{(x_{max} - x_{min})} \frac{1}{(y_{max} - y_{min})} dy dx \\
&+ \int_{x_{int}}^{x_{max}} \int_{y=a_2 x + b_2}^{y=a_1 x + b_1} \frac{1}{(x_{max} - x_{min})} \frac{1}{(y_{max} - y_{min})} dy dx
\end{aligned} \tag{22}
$$

Considering that the bounds on x and y are given by the unit square and considering the definition of x_{int} found:

$$
\begin{aligned}
area &= \int_0^{x_{int}} \int_{y=a_2 x + b_2}^{y=a_1 x + b_1} dy dx + \int_{\frac{b_1 - b_2}{a_2 - a_1}}^{x_{max}} \int_{y=a_1 x + b_1}^{y=a_2 x + b_2} dy dx \\
&= \int_0^{\frac{b_1 - b_2}{a_2 - a_1}} (a_2 - a_1) x + (b_2 - b_1) dx + \int_{\frac{b_1 - b_2}{a_2 - a_1}}^{1} (a_1 - a_2) x + (b_1 - b_2) dx \\
&= \left[(a_2 - a_1) \frac{x^2}{2} + (b_2 - b_1) x \right]_0^{x_{int}} + \left[(a_1 - a_2) \frac{x^2}{2} + (b_1 - b_2) x \right]_1^{x_{int}} \\
&= (a_2 - a_1) \frac{2 x_{int}^2 - 1}{2} + (b_2 - b_1)(2 x_{in} - 1)
\end{aligned} \tag{23}
$$

The closed formula for the error will therefore be the absolute value of equation 23, since the lines could be given in any order. The output of the perceptron may be a line above or below the class line. To avoid having to check which it is enoug to take the absolute value.

This formula was used to predict the error on a given perceptron output. However, further thought revealed that this is an overestimation of the error for many cases. This is so because it assumes that the intersection falls within the unit square and that both lines intersect the the left and right borders of the unit square. While the given line does intersect the left and right corners, the output of the perceptron might not, and the intersection of both may fall beyond bounds as they could be almost parallel. Simulation confirmed this,

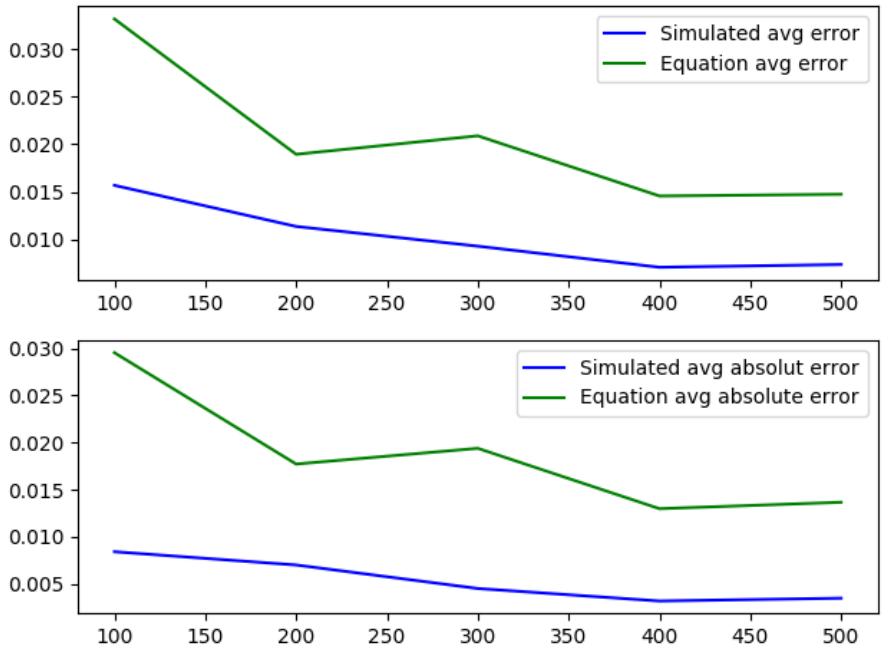


Figure 2: Top: average difference between algorithm an predicted (green) and simulated (blue) errors. Bottom:Average of the absolute value of same differences

since generating new test points yielded an error below the predicted in many cases, far beyond the accuracy of the simulated error.

A finite finishing algorithm was therefore developed to predict the error precisely. The algorithm developed takes into account where the intersection occurs and what borders the lines cross to determine the area as the area of one or two polygons. This polygons might have to have the corners of the unit square included, which was the hardest part of the algorithm. The area can be quickly computed through the "shoelace formula" as was obtained from: http://stackoverflow.com/questions/24467972/calculate-area-of-polygon-given-x-y-coordinates.

Running simulations to determine which method was better provided convincing results. Figure 2 shows the average of the difference and of the absolute value of the difference of the values produced by the algorithm, compared to the formula and the simulation. As we can observe the formula significantly overestimates and while the simulation incurs in a 2% error, this errors are randomly distributed and therefore the average of the absolute value of the difference is smaller.

(2 and 3) The error of the algorithm was then considered by evaluating

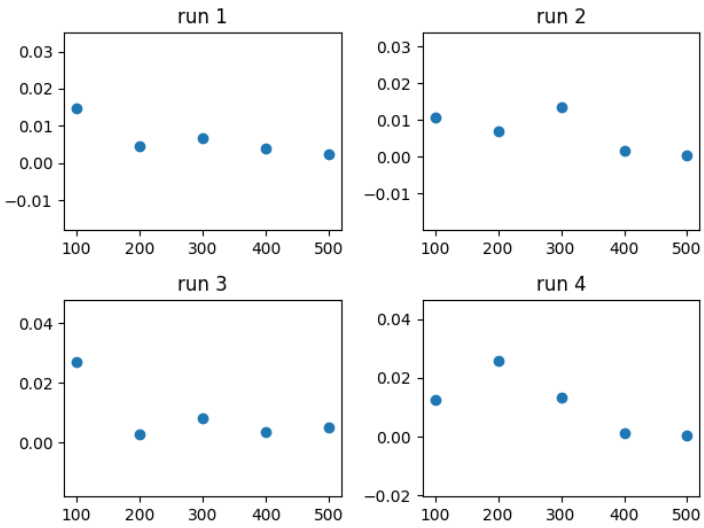


Figure 3: Errors for 4 runs of different size training sets

the error over different number of training points. Figure 3 shows the error produced for different amount of training data over 4 different runs. While the test error appears to be partly random there also seems to be a slight correlation towards a decrease in error with sample size.

This was confirmed by plotting the average and 90% confidence interval over 100 runs. As shown in Figure 4, the both the error and the confidence interval decrease with the number of training points. This is caused by the fact that the perceptron algorithm produces a line that can separate the given input. If the given training points are random, then the more points there are, the more probability there is of a point falling closer to the separating line. To separate the point, the output of the perceptron will have to be closer to that of the real line.

(4) The described process was then repeated for the case where the feature vetors are uniformly distributed over the region $K_\gamma = (x_1, x_2) \in [0,1]^2 : |x_2 - x_1 - 0.1| > \gamma$. Given the discussion of the methods of calculating the error, the approach taken was to modify the area finding algorithm, considering the limitations in the equation and in the simulation. The algorithm was easily modified by running it on the two lines defining the new feature space region, and taking only one of the sides of the area under the intersecting lines. Figure 5 shows the average and 90% confidence intervals for the error under the different margins $\gamma = 0.3, 0.1, 0.01, 0.001$. It can be observed how as the margin decreases so do the average error and its confidence intervals.

The bound on the number of iterations $\frac{||w_*||^2\mathcal{R}^2}{\rho^2} \geq t$ as proved in equation 18 was also checked. The key to understanding the bound is that all its components are random and therefore the bound will vary from run to run of the algorithm

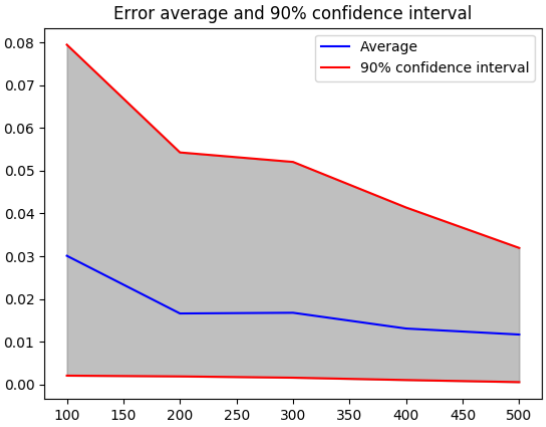


Figure 4: Average and 90%conifidence interval of the perceptron error

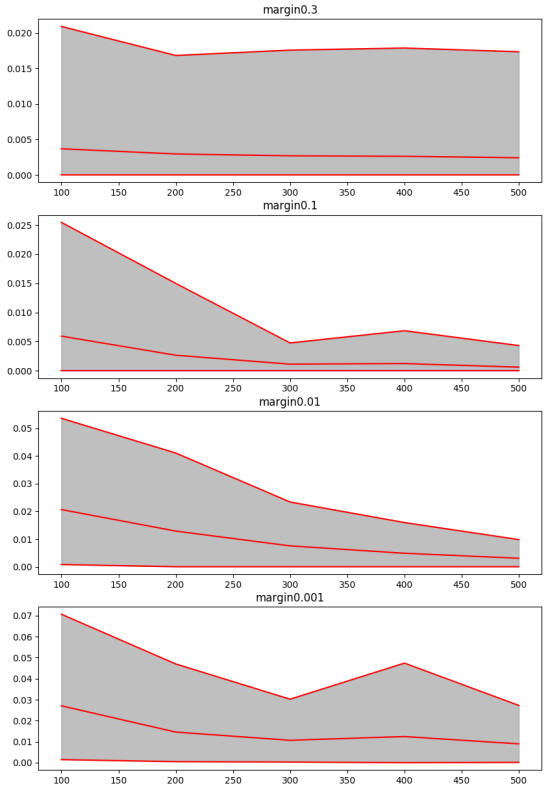


Figure 5: Average and 90%conifidence interval of the perceptron error for different margins

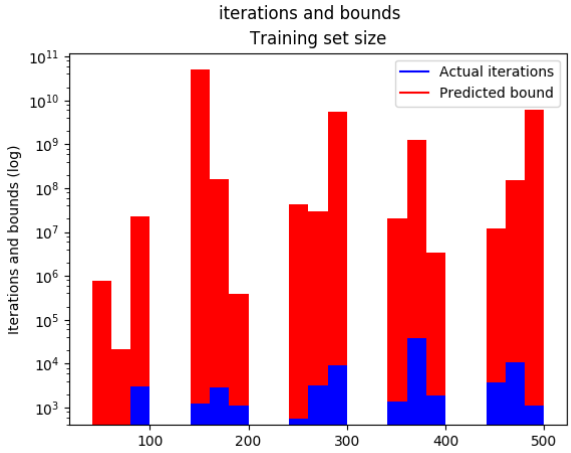


Figure 6: Predicted bound on number of iterations (red) and actual iterations (blue)

with different data sets. By the definitions of each component we know that R and ρ are both related to the max and min points given as data respectively. Also $||w_*||^2$ can take very different values since there are infinite definitions of a line in the normal form $ax+by=c$ for any line in the intercept form $mx+c=y$. This is so because for any normal definition a, b and c we have that $2a$, $2b$ and $2c$ define the same line. The result is that as observed in Figure 6 the bounds an iterations for the different training sizes do not show any relation with th training size. Note also that the variability is really high, since the figure has a logarithmic y-axis. It can also be observed that all of the actual iterations do fall under the predicted bound.

3.3 Part c

(1) The previously used perceptron algorithm assumed that the input training data was linearly separable or iterated indefinitely. To use the algorithm on data which might not be linearly separable two changes where implemented. First was to limit the number iterations so as to not end in an infinite loop. Second, the w that best fits the training data is saved. Since we might iterate indefinitely and in doing so the error might cycle, we would like to end with the best fit line.

The result of the modified perceptron with the real data was that th raw dataset was linearly separable and the 2-D feature set was not. Figure 7 shows the resulting output line, along with both the training and the test data used. As it can be observed, some of the training data is also misclassified.

(2) The two data sets result in different train and test error. While the 2D set converges in less than 25 iterations to a %24.27 training error and a %44.50

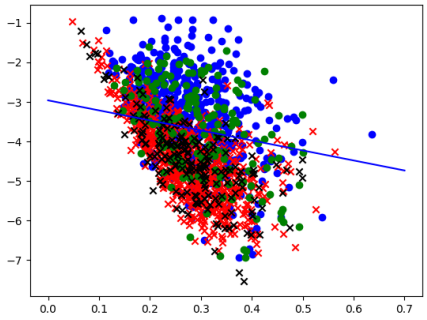
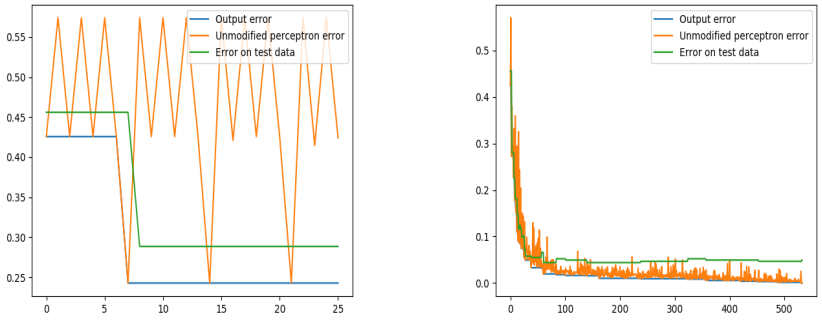


Figure 7: Perceptron output (blue line) to separate 2D feature training set (red crosses and blue dots) overlayed with the test set (black crosses and green dots)

test error, the raw data set converges in 532 iterations to a %0 training error and a %0.49 test error. The raw data set performs substantially better, which is expected since it has more data available to which to learn from.



(a) 2D feature set

(b) Raw data

Figure 8: Training error (blue) as a function of iterations compared to the old version (orange) and the test error (Green)

Figures 8b and 8a show how the error evolves as the algorithm steps through iterations, along with how the test error changes. The orange series represents the error the old version of the algorithm would have at an iteration, given that it does not save the best performing w. As we can see since the new algorithm does save this w it always performs better. Moreover, after the minimum training error achievable has been found, the old perceptron iterates in cycles.

(3) The optimal linear regression weights can be quickly computed using a library to compute the Moore-Penrose pseudoinverse. Using this as the initial

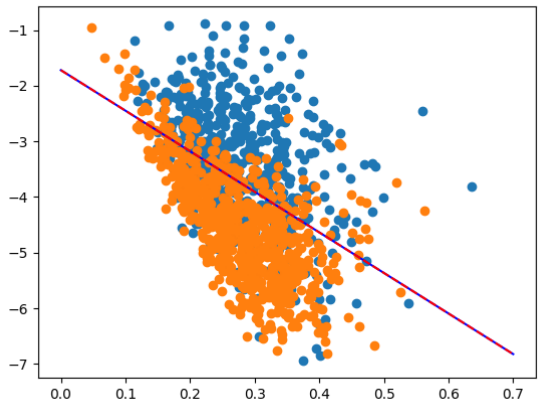


Figure 9: Output of the perceptron (blue line) running with weights initialized to the optimal linear regression line (red dotted)

weights for the algorithm improved its performance. As can be observed in Figure 9, the optimal regression line in dotted red matches really closely the output line in blue. Running the algorithm like this resulted in a %22.93 training error, but the same test error. Since the raw data version already converged to a linearly separable set of weights there was no improvement in the training error nor was there in the test. Hoever the algorithm did converge faster. It converged in 514 iterations instead of the original 532.

(3) Assuming that all data points are generated in an i.i.d fashion, we can use Hoeffdings inequality to predict a high probability upper bound of the difference between the empirical test error and the ideal test error for a given hypothesis chosen h, defined as:

$$empiricalerror = R_{emp}(h) = \frac{1}{n}\sum_{i=1}^{n}\mathbb{I}(h(x_i \neq f(x_i)) \tag{24}$$

$$idealerror = R_{ideal}(h) = \mathbb{P}(h(x_i \neq f(x_i)) \tag{25}$$

In this case we will need the two sided version of the inequality:

$$\Pr\left(|R_{emp}(h)X_t - R_{ideal}(h)| > \varepsilon\right) \leq 2\exp\left(-2\varepsilon^2 n\right) \tag{26}$$

We can therefore define ϵ as the maximum difference between the empirical test error defines and ideal test error with probability $1-\delta$ to be:

$$\epsilon \leq \sqrt{\frac{2\log\frac{2}{\delta}}{n}} \tag{27}$$

Since the data used has $n = 1273$, we can let $\delta = 0.1$ and say with 90% confidence that the bound on ϵ will be:

$$\epsilon \leq \sqrt{\frac{2 \log \frac{2}{0.1}}{1273}} = 0.0686 \tag{28}$$