Homework 3

Helen Ngo

October 11, 2016

Problem 1 Compare two algorithms on a classification task: the Pocket algorithm (designed for classification), and linear regression (not designed for classification). For linear regression, after learning the weights $\mathbf{w}$; we use $h(\mathbf{x}) = \text{sign}(\mathbf{w}^T\mathbf{x})$ to classify $\mathbf{x}$. For the dataset, start by using the given code. Then create another dataset using the same methods as above, which we will use to estimate E_{out}. Try the following three approaches using multiple randomized experiments and explain which works best in terms of both E_{out} and the amount of computation required.

(a) The Pocket algorithm, starting from $\mathbf{w} = 0$.

(b) Linear regression (applied as a classification method).

(c) The Pocket algorithm, starting from the solution given by linear regression.

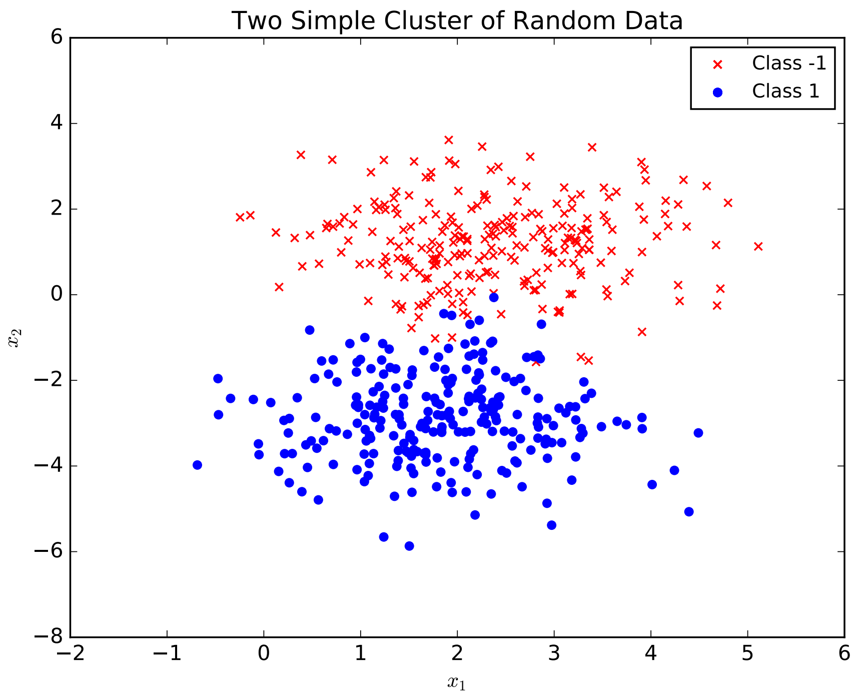
Also, try adding some *significant* outliers to the $y = +1$ class (arbitrarily chosen) of the training dataset and explain how that affects your results.

Solution: Several trials were conducted with samples sizes of 500. In each of the trials, a dataset was created and all three of the approaches above were used. The datasets were created using the generate points functions and the Python code given:

```
def generate_points(self, N=2000):
        # Generates random centers
        ctrs = 3 * np.random.normal(0,1,(2,2))
        # Generates random data following normal distributions
        X,y = make_blobs(n_samples=N, centers=ctrs, n_features=2,
                cluster_std=1.0, shuffle=False, random_state=0)
        y[y==0] = -1 # makes sure we have +1/-1 labels
        # y[[0,1,2,3,4]] = 1
        # Define x1, x2 limits
        self.x1min= np.floor(np.amin(X[:,0]))
        self.x1max = np.ceil(np.amax(X[:,0]))
        self.x2min = np.floor(np.amin(X[:,1]))
        self.x2max = np.ceil(np.amax(X[:,1]))
```

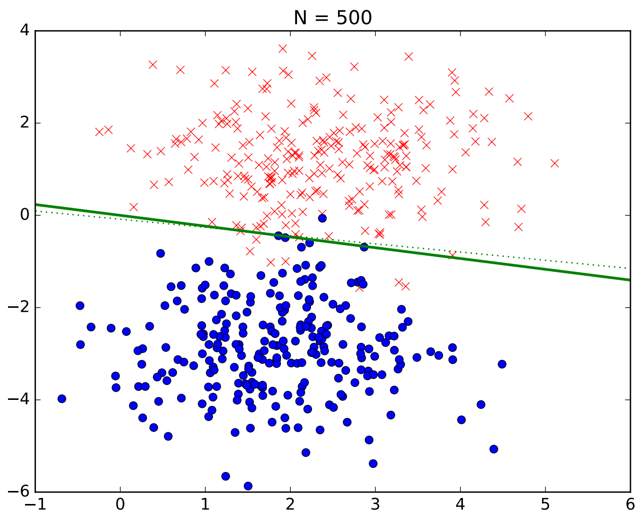
```
# Plots data
c0 = plt.scatter(X[y==-1,0],X[y==-1,1],
          s=20, color='r', marker='x')
c1 = plt.scatter(X[y==1,0],X[y==1,1],
          s=20, color='b', marker='o')
# c2 = plt.scatter(X[y==1,0][[0,1,2,3,4]],
#           X[y==1,1][[0,1,2,3,4]],
#           s=20, color='c', marker='o')
# Display legend
plt.legend((c0,c1), ('Class -1', 'Class 1'),
          loc='upper right',scatterpoints=1, fontsize=11)
# Dispay axis legends and title
plt.xlabel(r'$x_1$')
plt.ylabel(r'$x_2$')
plt.title(r'Two Simple Cluster of Random Data')
# Saves the figure into a .pdf file
plt.savefig('hw3_plot.pdf',bbox_inches='tight')
plt.show()
```

Since Trials 1 and 4 best summarize the difference between random linearly separable and inseparable datasets, they will be further analyzed below, but the final hypothesis for every dataset and algorithm combination can be seen in the Github repository. In Trial 1, a linearly inseparable dataset was produced:

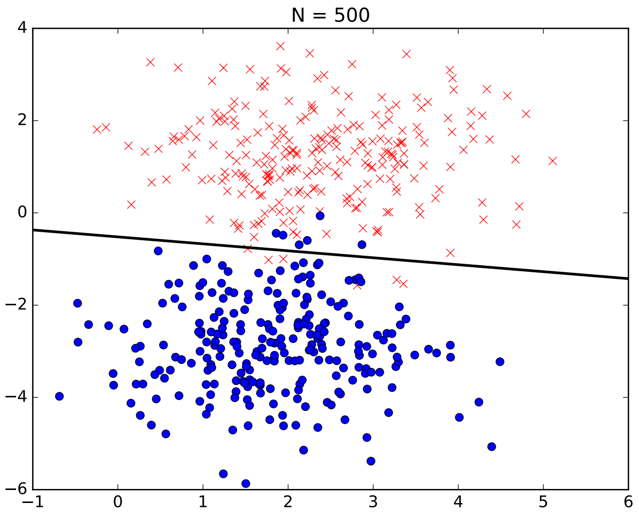


For this dataset, it took

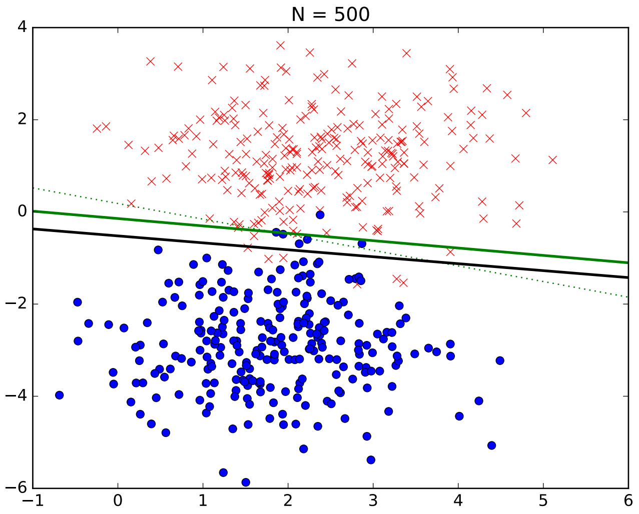
(a) the Pocket algorithm 723 iterations and approximately 9 seconds to converge,



(b) the linear regression 3 second to find a hypothesis, and



(c) the Pocket algorithm using the linear regression solution, 528 iterations and 7 seconds to converge.



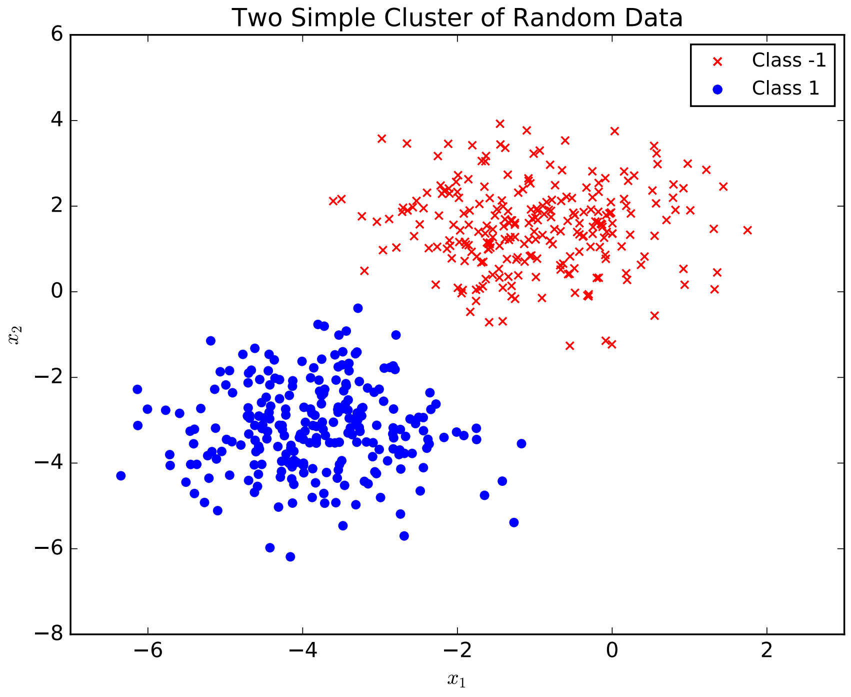
It should be noted that convergence is either no classification error, more than 500 iterations with the same optimal hypothesis, or more than ten thousand total iterations, which can be seen in the pla function.

```
        while self.classification_error(w) != 0:
            it += 1
            # Pick random misclassified point
            x, y = self.choose_miscl_point(w0)
            # Update weights
```

```
                w0 += y*x
                # Update if new weights are better
                if self.classification_error(w)>self.classification_error(w0):
                    w = copy.deepcopy(w0)
                    count = 0
                else:
                    count += 1
                # Converge after 500 iterations with the same wieghts
                if count > 500:
                    break
                # Converge after 10000 iterations overall
                if it > 10000:
                    break
            self.w = w
            # Plot and save the last iteration
            if save:
                if linear_regression:
                    self.plot(vec=w, vec0=w0, lr_vec=lr_vec,
                              file=file, save=True)
                else:
                    self.plot(vec=w, vec0=w0, file=file, save=True)
            # Return number of iterations
            return it
```

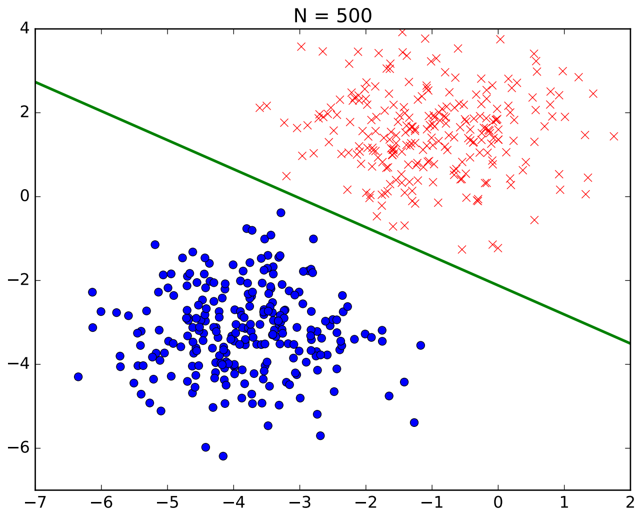
Taking into account how convergence is defined in the code, and what we know about the dataset, we can say that while there is no solution without misclassification, the Pocket algorithms produce hypotheses with lower E_{out}, but at a greater computational cost. For this specific example, the linear regression provides a similar E_{out} result to Pocket algorithms but less than half the computation required. We can see from the plot associated with (c) the Pocket algorithm starting from the solution given by linear regression, that an improvement was made to E_{out} to that of just the linear regression.

In Trial 4, a linearly separable dataset was produced:

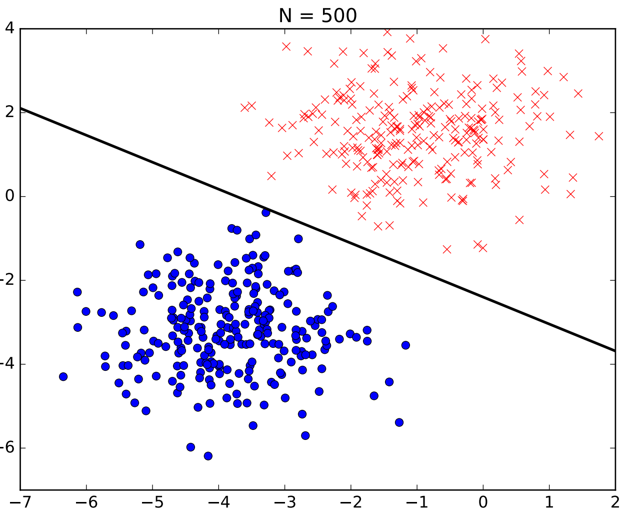


For this dataset, it took

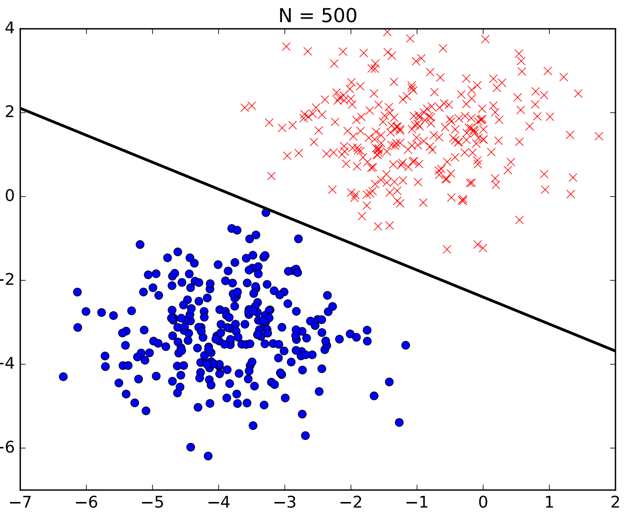
(a) the Pocket algorithm 21 iterations and approximately 2 seconds to converge,



(b) the linear regression 3 second to find a hypothesis, and

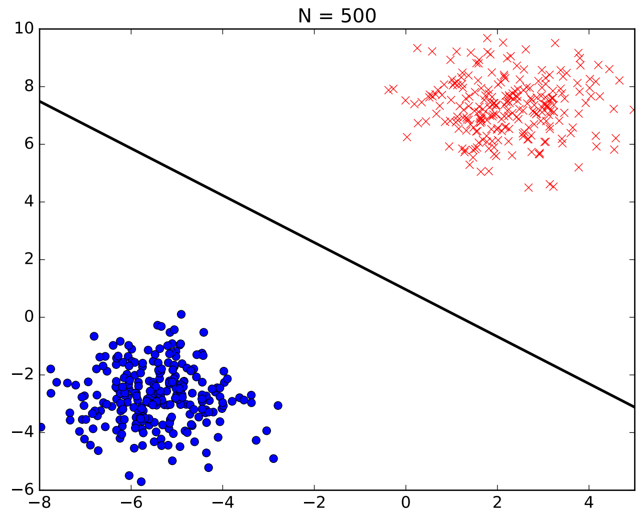


(c) the Pocket algorithm using the linear regression solution, 0 iterations and 3 seconds to converge.

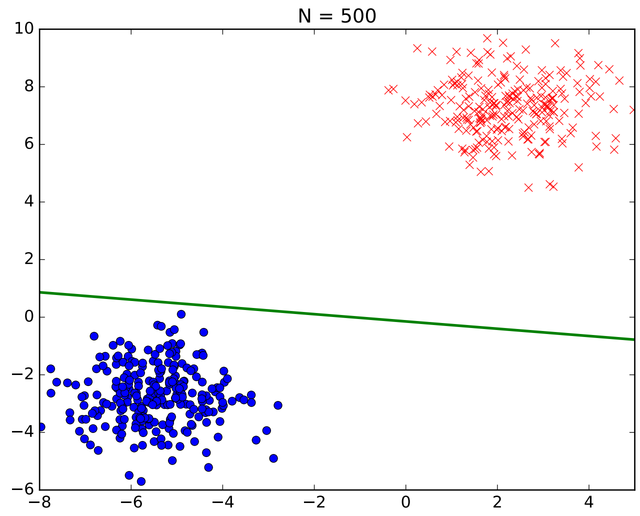


The $E_{out} = 0$ for all of the approaches, so the the amount of computation required is the most important. The pocket algorithm took the least amount of physical time to find a solution; it was faster by one second. Based on this dataset, it can be concluded that for linearly separable datasets that a pocket algorithm may be advantageous to an approach using linear regression. It should also be noted that the Pocket algorithm using the linear regression solution took zero iterations. This is because the starting weights given by the regression was already produced a classification with no error.

Next, a linearly separable data set was examined, both as its original dataset and modified with significant outliers. Without modifications the dataset is obviously linearly separable such that linear regression, used for classification, gives a hypothesis with the least square error and has no misclassification; the Pocket algorithms with a linear regression solution start will give the same hypothesis.



It only took the Pocket algorithm two iterations to produce a hypothesis with no misclassification.

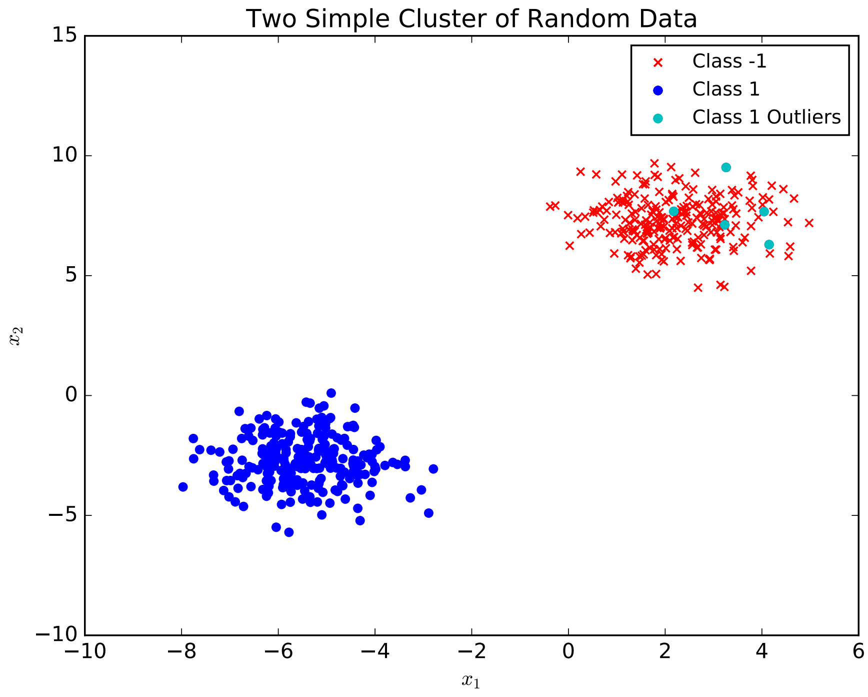


While there is no misclassification with this hypothesis, the line produced is significantly closer to class 1 than class -1.

In order to arbitrarily choose significant outlier to the $y = +1$ class, five of the data points for the $y = -1$ class was modified into the $y = +1$ class. This can be seen in the initial python code shown above, commented out. The only modification that cannot be seen is the the addition to the legend:

```
plt.legend((c0,c1,c2), ('Class -1', 'Class 1', 'Class 1 Outliers'),
                        loc='upper right',scatterpoints=1, fontsize=11)
```

In the plot below, of the dataset with the modification, the purposefully chosen outliers are colored cyan so that they can be easily identified as chosen outliers instead of random outliers.



With the outliers, the Pocket algorithm and Pocket algorithm with a linear regression start, must wait until convergence at 502 and 501 iterations, respectively. It can be seen for all the algorithms, that the final hypothesis does not change from their respective final hypothesis of the dataset without outliers.

In conclusion there are advantages to each of the algorithms, but when the separability of the dataset is unknown it is worth starting the Pocket algorithms with weights from linear regression. It only takes about three second to find the least square error line. The modified Pocket algorithm has all the benefits of linear regression, including ignoring outliers, and the ability to reduce classification error like the perceptron.

Figure 1: Final hypothesis using Pocket algorithm.

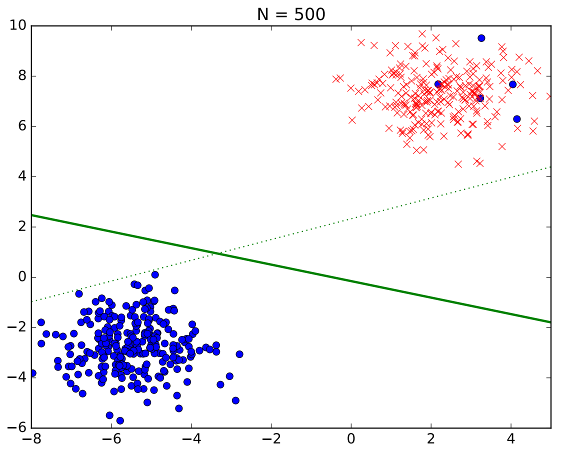


Figure 2: Final hypothesis using linear regression.

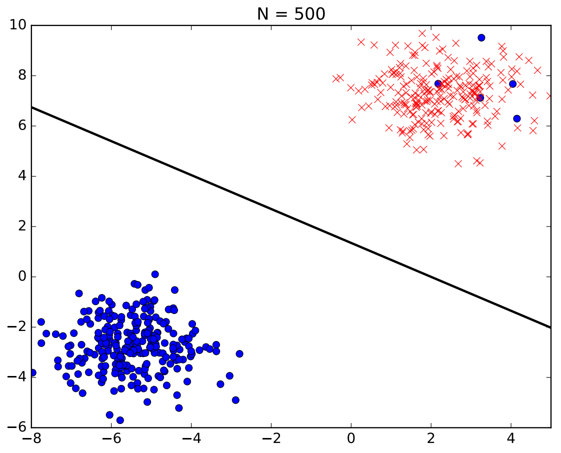


Figure 3: Final hypothesis using Pocket algorithm with a linear regression start.