ICS 663: Homework 3 - Linear Discriminant Functions

Christopher Mullins

November 11, 2011

Contents

1 Introduction

Suppose that a dataset $D = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ has two classes, and that d_i denotes the class for $\mathbf{x}_i$. For convenience, we can label these classes -1 and 1, so that if $\mathbf{x}_i$ is in the first class, $d_i = -1$.

A dataset with two classes is linearly separable if there exists a vector $\mathbf{a}$ such that $d_i \cdot \mathbf{a}^T \mathbf{x}_i > 0 \; \forall i$. It is not uncommon for datasets to be *approximately* linearly separable, meaning that only a minority of data are misclassified by an appropriately chosen $\mathbf{a}$. For this reason, methods for finding linear separators are of some interest.

The perceptron is a very simple artificial neural network that is capable of finding a linear separator for a dataset by incrementally updating $\mathbf{a}$ based on misclassified results. More explicitly, at each iteration i and for each misclassified sample $\mathbf{x}_j$, we update $\mathbf{a}$ using the following rule:

$$\mathbf{a}_{i+1} = \mathbf{a}_i + \eta(i) \left(d_j - \mathbf{a}_i^T \mathbf{x}_j \right) \mathbf{x}_j .$$

Here, $\eta(i)$ denotes the *learning rate* at iteration i. This is generally a value between 0 and 2, and can change (usually decay) as i gets larger. The perceptron halts when no misclassified samples remain. For data that aren't fully linearly separable, a different halting condition should be used (error threshold, number of iterations, etc.).

The MSE method uses a well-known method for linear regression to find a linear boundary. We first form a matrix with the data:

$$\mathbf{Y} = \begin{pmatrix} d_0 x_{0,0} & d_0 x_{0,1} & \ldots & d_0 x_{0,k} & d_0 \\ d_1 x_{1,0} & d_1 x_{1,1} & \ldots & d_1 x_{1,k} & d_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ d_n x_{n,0} & d_n x_{n,1} & \ldots & d_n x_{n,k} & d_n \end{pmatrix} .$$

We then invent (or calculate) some margin weight vector $\mathbf{b}$ (such that $b_i > 0 \; \forall i$) that we solve for, so that $\mathbf{Ya} = \mathbf{b}$. Since $\mathbf{Y}$ is almost never square in practice ($n \gg k$ in general), the system is *overdetermined*, meaning there is likely not an exact solution. We can, however, find a solution that minimizes the squared error, $\|\mathbf{Ya} - \mathbf{b}\|$.

We can use the pseudoinverse to find such a solution:

$$\mathbf{Y}^\dagger = (\mathbf{Y}^T \mathbf{Y})^{-1} \mathbf{Y}^T .$$

This calculation works if $\mathbf{Y}^T\mathbf{Y}$ is non-singular. If it is singular, one can use the more generalized definition or use a gradient descent method. More generally, the pseudoinverse is:

$$\lim_{\epsilon \to 0} (\mathbf{Y}^T\mathbf{Y} + \epsilon\mathbf{I})^{-1}\mathbf{Y}^T.$$

2 Experiments

For this assignment, I use the perceptron and MSE methods for finding linear discriminant functions. The provided data is shown in figure 1. In the result plots, I omit the legend to leave more room for the separators. The same color and symbols for each class are used in these plots as in figure 1.

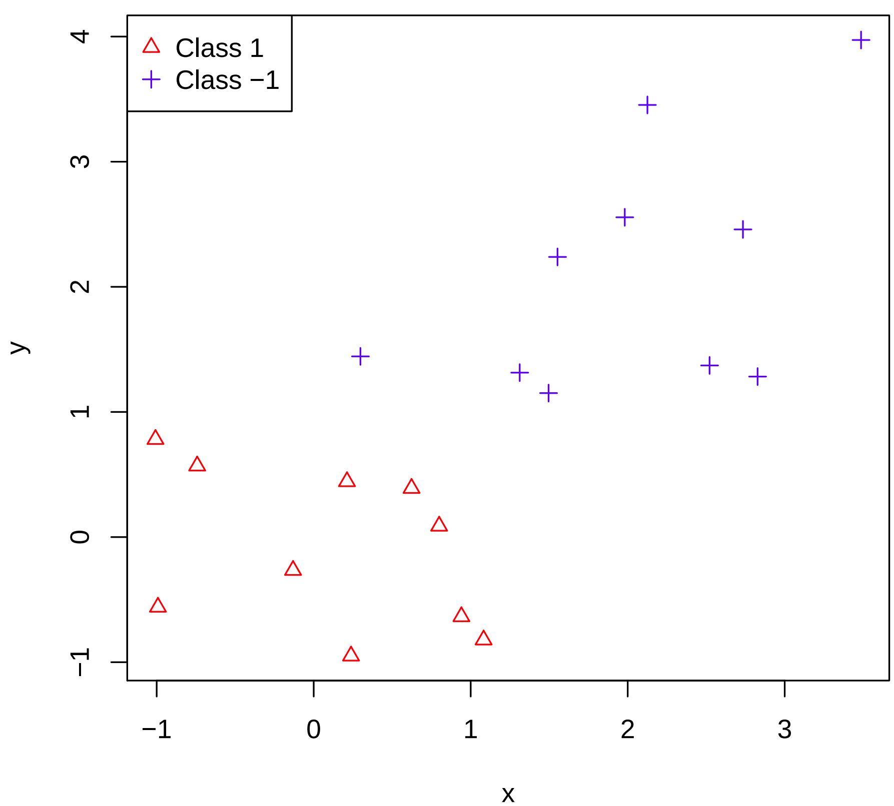

Figure 1: Data provided for homework assignment

2.1 Perceptron

With the data provided, the pseudoinverse $\mathbf{Y}^\dagger$ exists, so using gradient descent is unnecessary.

I ran my perceptron code on the provided data four times and provided the results in a single plot. The black lines represent the decision boundary $\mathbf{a}$ at each epoch. The green line represents the final result produced by the perceptron. It perfectly separates the data into two classes. The results are shown in figure 2 and 3 for $\eta = 0.5$ and $\eta = 0.01$, respectively.

The results are unsurprising. With $\eta = 0.5$, a decision boundary is found within a few iterations. With $\eta = 0.01$, significantly more iterations are needed before the perceptron converges on a solution.

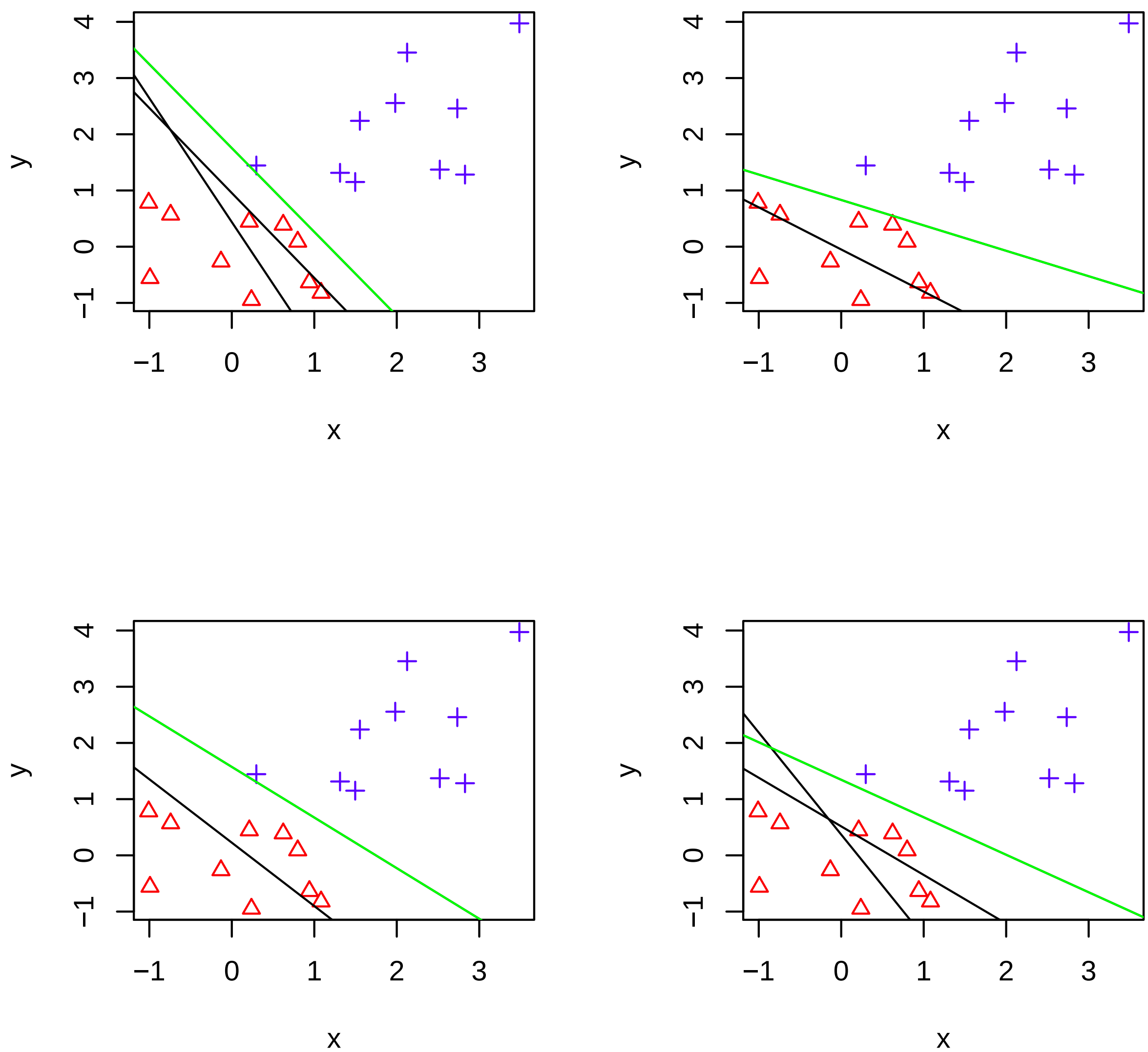

Figure 2: Results for Perceptron with $\eta = 0.5$.

2.2 MSE

One disadvantage of the MSE method for finding a linear decision boundary is that it does not guarantee a perfect separator, even if the data is perfectly separable. In the provided data, the solution provided by MSE does *not* perfectly separate the data. The results are shown in figure 4.

3 Implementation

The implementation I used for the plots included in this report is done in R. The code is split across three files: `perceptron.R`, `mse.R` and `hw3.R`. The former two have code for the perceptron and MSE methods, respectively. The last is a script that seamlessly runs all of the tasks required for HW3 (generates plots and

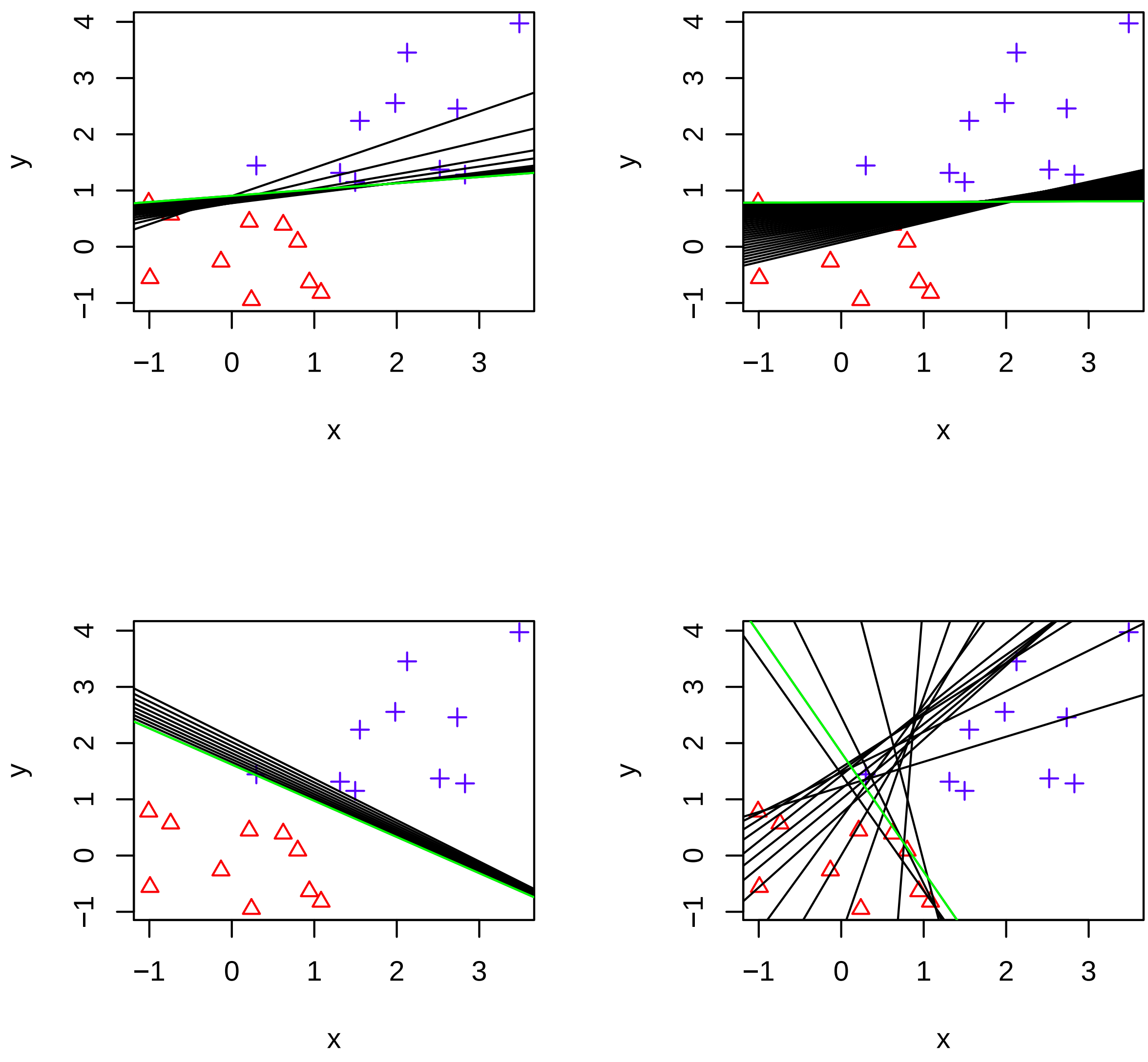

Figure 3: Results for Perceptron with $\eta = 0.01$.

puts them in the `./plots` directory).

If you'd like to run the code, you can use the command `R --no-save --slave < hw3.R`.

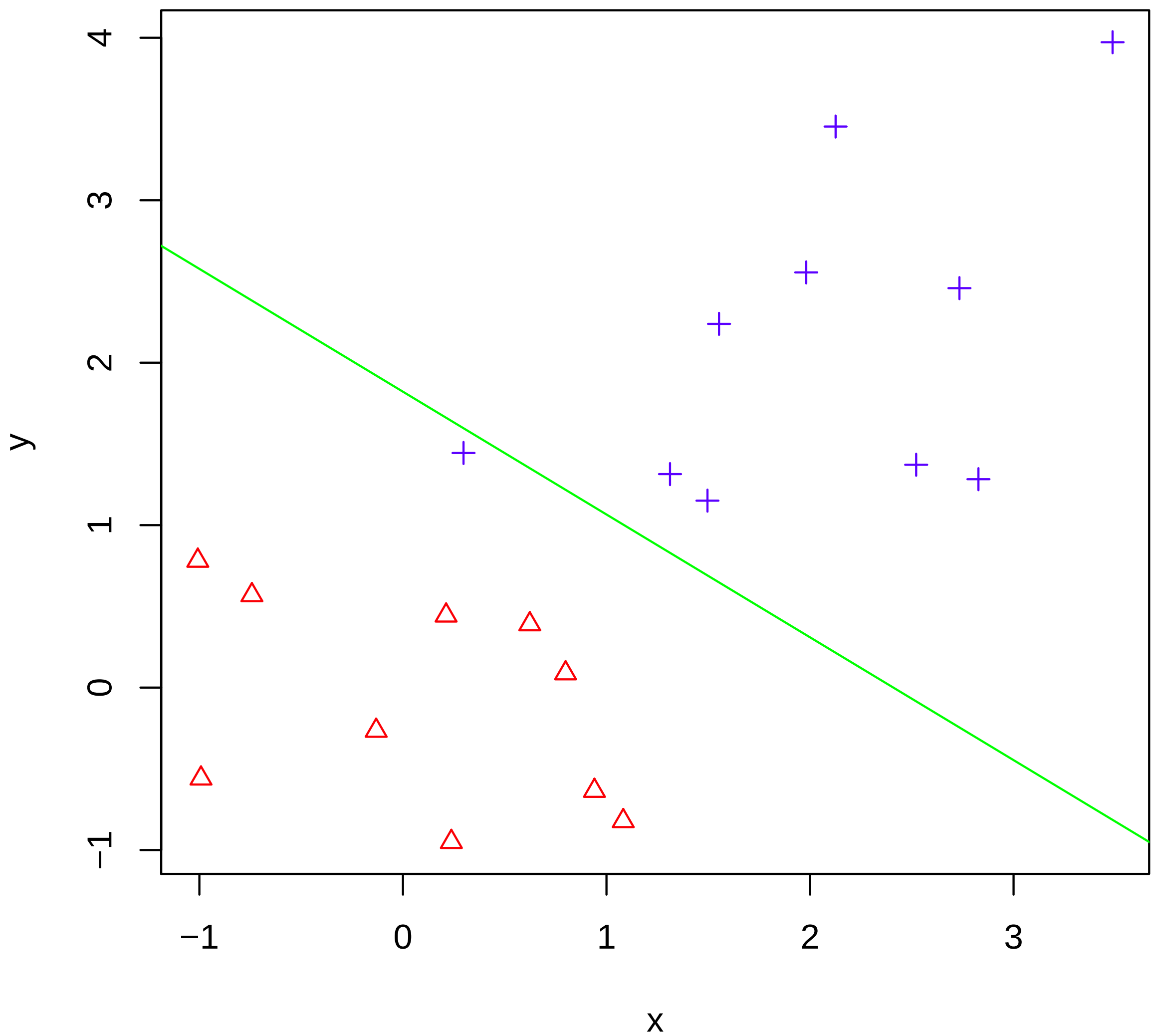

Figure 4: Results for MSE