Math 156: Lab assignment #3

Due: Tuesday October 24th

The goal of this assignment is to get familiar with some basic classification algorithms. Here we will compare the performance of linear classifiers to the k-nearest neighbor classification. We will work with very low dimensional datasets so we can visualize the performance of our algorithms.

1 Theory

Let $\mathcal{D}$ be a dataset consisting of r different classes $\mathcal{D} = \mathcal{D}_1 \cup \mathcal{D}_2 \cup \cdots \cup \mathcal{D}_r$ in $\mathbb{R}^d$. Our goal is to build a classification function $\boldsymbol{f} : \mathbb{R}^d \to \mathbb{R}^r$. Once we have the classification function, we can decide the class of some new point $\boldsymbol{x} \in \mathbb{R}^d$ by choosing the largest entry of the vector $\boldsymbol{f}(\boldsymbol{x}) = (f_1(\boldsymbol{x}), \ldots, f_r(\boldsymbol{x})) \in \mathbb{R}^r$. In other words we classify $\boldsymbol{x}$ by calculating $\arg\max_{i \in \{1,2,\ldots,r\}} f_i(\boldsymbol{x})$

In this assignment we will restrict our attention to linear classifier functions $\boldsymbol{f}(\boldsymbol{x}, W, \boldsymbol{w}_0) = W\boldsymbol{x} + \boldsymbol{w}_0$. Note that if we let $\boldsymbol{x}' = (\boldsymbol{x}, 1) \in \mathbb{R}^{d+1}$ then we can write $\boldsymbol{f}(\boldsymbol{x}, W, \boldsymbol{w}_0) = W'\boldsymbol{x}'$ where W' is a $r \times (d+1)$ matrix whose first d columns are given by W and the last column is $\boldsymbol{w}_0$. We will assume that this has been done and drop the $'$ notation from here on out.

We can identify each class $i \in \{1, 2, \ldots, r\}$ with a target value $\boldsymbol{t}(i) \in \mathbb{R}^r$. Here we will take the target values to be the standard basis vectors $\boldsymbol{t}(i) = \boldsymbol{e}_i$ where $\boldsymbol{e}_1 = (1, 0, \ldots, 0)$ etc. Using the known data $\boldsymbol{x}_n \in \mathcal{D}$, we can compute the coefficients for the linear classifier by minimizing the least squares error

$$J(W) = \sum_{n=1}^{N} ||W\boldsymbol{x}_n - \boldsymbol{t}_n||_2^2$$

where we define $\boldsymbol{t}_n = \boldsymbol{t}(i)$ where i is the class of x_n. In matrix form we may write the least squares error as

$$J(W) = ||WX - T||_F^2$$

where the columns of X and T are the points x_n and t_n respectively.

Task 1: What is the condition for a point x to be on the decision boundary between two classes i and j? What is the equation for that decision boundary?

2 Experiments

2.1 Two Classes

We will begin by generating a data set $\mathcal{D}$ consisting of two classes $\mathcal{D}_1$ and $\mathcal{D}_2$ in $\mathbb{R}^2$. $\mathcal{D}_1$ will consist of 50 randomly sampled points from the normal distribution $\mathcal{N}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$ and $\mathcal{D}_2$ will consist of 50 randomly sampled points from the normal distribution $\mathcal{N}(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)$ where

$$\boldsymbol{\mu}_1 = (1, 2) \quad \boldsymbol{\Sigma}_1 = \begin{pmatrix} 0.1 & 0.05 \\ 0.05 & 0.2 \end{pmatrix}, \quad \boldsymbol{\mu}_2 = (2, 4), \quad \boldsymbol{\Sigma}_2 = \begin{pmatrix} 0.2 & -0.1 \\ -0.1 & 0.3 \end{pmatrix}.$$

The MATLAB function *mvnrnd* will be helpful.

Task 2: Create a scatterplot of the data you just generated. Use different colors and/or markers for each class. Does it make sense to classify this data with a linear classifier?

Task 3: Compute the least squares classification coefficients W. Compute the decision boundary and overlay it onto the scatterplot you created in task 4.

Task 4: Now we will test the performance of the classifier on new points sampled from the distributions. Create a set Y of 100 new points by sampling 50 new points from $\mathcal{N}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$ and 50 new points from $\mathcal{N}(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)$. Classify the points in Y using the classification coefficients from task 5. What is the classification accuracy? Does the performance make sense given the decision boundary plot from task 5?

2.2 Three classes

Now we will create a 3 class dataset. Create a data set $\mathcal{D}$ with 3 classes $\mathcal{D}_1$, $\mathcal{D}_2$ and $\mathcal{D}_3$ by generating 50 points from each of $\mathcal{N}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1), \mathcal{N}(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2), \mathcal{N}(\boldsymbol{\mu}_3, \boldsymbol{\Sigma}_3)$ where

$$\boldsymbol{\mu}_1 = (2, 2)\, \boldsymbol{\Sigma}_1 = \begin{pmatrix} 0.2 & 0.05 \\ 0.05 & 0.3 \end{pmatrix}, \quad \boldsymbol{\mu}_2 = (2, 4), \quad \boldsymbol{\Sigma}_2 = \begin{pmatrix} 0.4 & -0.1 \\ -0.1 & 0.3 \end{pmatrix},$$

and

$$\boldsymbol{\mu}_3 = (3, 3), \quad \boldsymbol{\Sigma}_3 = \begin{pmatrix} 0.5 & -0.3 \\ -0.3 & 0.4 \end{pmatrix}.$$

Task 5: Create a scatter plot where each class is marked by a different color. Compute the classification coefficients W, and overlay the scatter plot with the decision boundaries. Be sure to represent the actual 3-class decision boundaries, and not just the pairwise ones. If you are having trouble plotting the 3-class boundaries you can plot the pairwise boundaries and then highlight the true 3-class boundaries with some highlighter tool (on the computer or on a printed out copy).

Again we will test the performance of our classifier by creating a new set Y of 150 points by sampling 50 new points from each of $\mathcal{N}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$, $\mathcal{N}(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)$, $\mathcal{N}(\boldsymbol{\mu}_3, \boldsymbol{\Sigma}_3)$.

Task 6: Classify the points in Y using the linear classifier you computed in task 5. What percentage of points were classified correctly? Does the success rate make sense given the distribution of the data?

Task 7: Now we will compare the performance of the linear classifier to the k-nearest neighbor classifier. For each point $\boldsymbol{x} \in Y$ compute the k nearest neighbors of $\boldsymbol{x}$ in $\mathcal{D} = \mathcal{D}_1 \cup \mathcal{D}_2 \cup \mathcal{D}_3$. (You may use the built in MATLAB function to do this). Choose the label of $\boldsymbol{x}$ by assigning $\boldsymbol{x}$ to the class containing the largest number of its neighbors. Try this for a few different values of $k \in [1, 15]$. What value of k gives the best accuracy? How does the accuracy compare to the linear classifier? Based on your plot from task 6, comment on why one method might perform better than the other.