Machine Learning (ECE 4850)

Instructor: Dr. Shekaramiz

Class Project 4

Submission Type: Online, Canvas

Introduction

Data-Driven Classification

In this assignment you will explore some data-driven classification models to classify data. You will empirically characterize their performances on a set of data.

The data are produced according to a model described later. Figure 1 shows the training data for $d = 2$ dimensional data. The `x' data is for *class 0*; the `o' data is for *class 1*. This **training data** is in the file **classasgntrain1.dat**. The first two columns of this (ascii) file are the x and y coordinates of $N = 1000$ points of the *class 0* data; the second two columns of the data are the x and y coordinates of the *class 1* data. Instances of data are generated by calling the MATLAB function **gendat2**, as follows:

x = gendat2(*class* , *N*);

where *class* indicates which class of data you want (either 0 or 1), and *N* indicates how many points you want to generate. For example, to generate 20 points of class zero data and store it in the array *samp1* you would call

samp1 = gendat2(0 , 20);

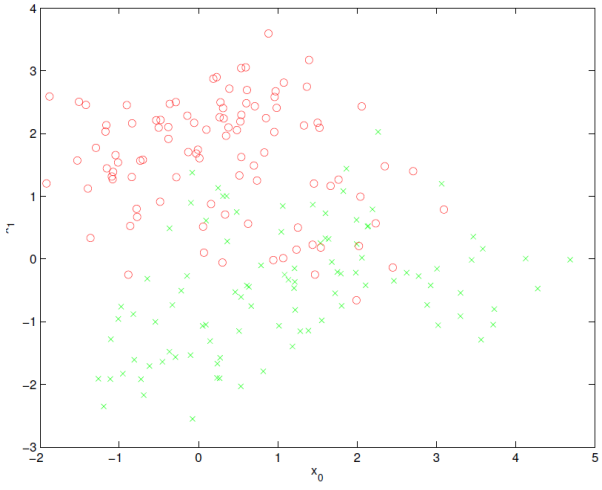

Figure 1: Training data for two-class, two-dimensional problem.

Let N_0 be the number of points of training data from *class 0*, and let N_1 be the number of points of training data from *class 1*. Let $N = N_0 + N_1$ be the total number of training data. Let $\boldsymbol{x}_i$ denote the coordinates of a training vector (thinking of this as a column vector), and let $\boldsymbol{y}_i$ denote the corresponding class value. That is, either $y_i = 0$ or $y_i = 1$, depending on the class the point comes from. Then the set of data $(\boldsymbol{x}_i, \boldsymbol{y}_i), i = 1,2, \dots, N$ is the total set of training data. For some of the discussions below, we assume that the data are ordered so that the first N_0 training points are from *class 0* and the next N_1 training points are from *class 1*.

In the sections below you are introduced to some classification algorithms. Your assignment will be to program each of these in Matlab/Python, then evaluate their performance on the training data provided as well as on new test data that you generate with the *gendat2* function.

The classifiers presented here are described in *The Elements of Statistical Learning Theory by T. Hastie, R. Tib-shirani, and J. Friedman (Springer, 2001)*.

Linear regression

For some classification problems, it suffices to provide a straight line (or plane, in higher dimensions) dividing the two classes. Linear regression is one means of determining such a dividing line (or plane).

Let $\boldsymbol{x} = [x_1, x_2, \dots, x_d]^T$ be a d-dimensional vector. A linear function of $\boldsymbol{x}$ is

$$f(\boldsymbol{x}) = \beta_0 + \sum_{j=1}^{d} x_j \beta_j.$$

The term β_0 is to change the "y-intercept" of the line, to account for non-zero mean data. In the usual regression problem, each point $\boldsymbol{x_i}$ has associated with it some value y_i. We choose the parameters of the line $\boldsymbol{\beta} = [\beta_1, \beta_2, \dots, \beta_d]^T$ in such a way that we get the best match possible over all training points. That is, we desire to minimize

$$RSS(\boldsymbol{\beta}) = \sum_{i=1}^{N} (y_i - \beta_0 - \sum_{j=1}^{d} x_{ij}\beta_j)^2$$

where x_{ij} is the jth coordinate of the training data vector $\boldsymbol{x_i}$, and where RSS stands for "residual sum of squares" (that is, the sum of the squares of the "residuals" or errors). This can be written in more convenient form as follows. First, let

$$\boldsymbol{X} = \begin{bmatrix} 1 \\ \boldsymbol{x} \end{bmatrix}.$$

That is, stack a 1 on top of the column vector $\boldsymbol{x_i}$. Then the linear function can be written

$$f(\boldsymbol{x}) = \boldsymbol{\beta}^T \boldsymbol{X}.$$

Now let y be the stack of data output values for all the data:

$$\boldsymbol{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}.$$

and $\boldsymbol{X}$ be the $N \times (d + 1)$ matrix formed by stacking the data as *rows* (including the 1):

$$X = \begin{bmatrix} 1 & \boldsymbol{x}_1^T \\ 1 & \boldsymbol{x}_2^T \\ \vdots & \vdots \\ 1 & \boldsymbol{x}_N^T \end{bmatrix}.$$

It turns out that $\widehat{\boldsymbol{\beta}}$ can be found via

$$\widehat{\boldsymbol{\beta}} = (X^T X)^{-1} X^T \boldsymbol{y}.$$

For a binary classification problem, the $\boldsymbol{y}$ data are modified somewhat. We form the $N \times 2$ matrix Y whose 2 columns indicate respective class membership of the given data point. That is, we form the X matrix $(N \times (d + 1))$ and the corresponding Y matrix $(N \times 2)$ as follows:

$$X = \begin{bmatrix} 1 & \boldsymbol{x}_1^T \\ 1 & \boldsymbol{x}_2^T \\ \vdots & \vdots \\ 1 & \boldsymbol{x}_{N_0}^T \\ 1 & \boldsymbol{x}_{N_{0+1}}^T \\ 1 & \boldsymbol{x}_{N_{0+2}}^T \\ \vdots & \vdots \\ 1 & \boldsymbol{x}_N^T \end{bmatrix}, \quad Y = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ \vdots & \vdots \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ \vdots & \vdots \\ 0 & 1 \end{bmatrix},$$

The matrix Y is called the indicator response matrix. We then find the estimated coefficient matrix $\hat{B}$ via

$$\hat{B} = (X^T X)^{-1} X^T Y.$$

The estimated indicator response matrix is

$$\hat{Y} = X^T \hat{B}$$

To use the classifier after training, suppose that the vector we want to classify is $\boldsymbol{x}$. We form the indicator response vector

$$\hat{\boldsymbol{y}} = [1 \quad \boldsymbol{x}^T]\hat{B}$$

This is a vector with $K = 2$ elements in it, $\hat{\boldsymbol{y}}^T = [\hat{y}_1\ , \hat{y}_2\ ]$. The estimated class $\hat{k}$ corresponds to the column with the largest value:

$$\hat{k} = arg_{k\in\{0,1,\ldots,K-1\}}\ \hat{y}_k \quad .$$

Problem 1: Using the 100 points of training data in *classasgntrain1.dat*, write MATLAB/Python code to train the coefficient matrix $\hat{B}$. Determine how well this linear regression classifier works on the training data and on 10,000 points of test data (5000 from each class) generated by *gendat2.m*. Record your results in a table such as the following:

	Error Rates	
Method	Training	Test
Linear Regression		
Quadratic Regression		
1-Nearest Neighbor		
5-Nearest Neighbor		
15-Nearest Neighbor		

You will be filling in other rows of the table as you progress through this assignment.

Also, make a plot indicating the regions of classification. In this linear classifier it is possible to do this analytically, but for other classifiers this is difficult. Instead, simply closely sample the input space and plot the classification results.

To get you started (and to show you this is not an impossibly difficult assignment), I am providing the solution to this problem. This code provides a complete solution to this problem.

```
% classasgn1.m

% Sample classifier program

% Load the training data and divide into the different classes

load classasgntrain1.dat

x0 = classasgntrain1(:,1:2)'; % data vectors for class 0 (2 x N0)

N0 = size(x0,2);

x1 = classasgntrain1(:,3:4)'; % data vectors for class 1 (2 x N1)

N1 = size(x1,2);

N = N0 + N1;

% plot the data
```

```
clf;
plot(x0(1,:), x0(2,:),'gx');
hold on;
plot(x1(1,:), x1(2,:),'ro');
xlabel('x_0');
ylabel('x_1');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Linear regression classifier
% Build the X matrix
X = [ones(N0,1) x0'; ones(N1,1) x1'];
% Build the indicator response matrix
Y = [ones(N0,1) zeros(N0,1); zeros(N1,1) ones(N1,1)];
% Find the parameter matrix
Bhat = (X'*X) \ X'* Y;
% Find the approximate response
Yhat = X*Bhat;
Yhathard = Yhat > 0.5; % threshold into different classes
nerr = sum(sum(abs(Yhathard - Y)))/2; % count the total number of errors
errrate_linregress_train = nerr / N;
% Now test on new (testing data)
Ntest0 = 5000; % number of class 0 points to generate
Ntest1 = 5000; % number of class 1 points to generate
xtest0 = gendat2(0,Ntest0); % generate the test data for class 0
xtest1 = gendat2(1,Ntest1); % generate the test data for class 1
nerr = 0;
for i=1:Ntest0
   yhat = [1 xtest0(:,i)']*Bhat;
    if(yhat(2) > yhat(1)) % error: chose class 1 over class 0
    nerr = nerr+1;
```

```
    end
end
for i=1:Ntest1
     yhat = [1 xtest1(:,i)']*Bhat;
    if(yhat(1) > yhat(2)) % error: chose class 0 over class 1
        nerr = nerr+1;
    end
end
errrate_linregress_test = nerr / (Ntest0 + Ntest1);
% Plot the performance across the window (that is, plot the classification regions)
xmin = min([x0(1,:) x1(1,:)]); xmax = max([x0(1,:) x1(1,:)]);
ymin = min([x0(2,:) x1(2,:)]); ymax = max([x0(2,:) x1(2,:)]);
xpl = linspace(xmin,xmax,100);
ypl = linspace(ymin,ymax,100);
redpts = []; % class 1 estimates
greenpts = [];% class 0 estimatees
% loop over all points
for x = xpl
      for y = ypl
          yhat = [1 x y]*Bhat;
          if(yhat(1) > yhat(2)) % choose class 0 over class 1
              greenpts = [greenpts [x;y]];
          else
              redpts = [redpts [x;y]];
          end
    end
end
plot(greenpts(1,:), greenpts(2,:),'g.','MarkerSize',0.25); plot(redpts(1,:),
redpts(2,:),'r.','MarkerSize',0.25); axis tight
```

Quadratic Regression

Obviously the linear regression technique is going to produce linear (or planar) decision boundaries. A richer structure can be obtained by using other basis functions to build the decision function. Simply including cross terms between independent variables is a straightforward way to do this.

To be a quadratic regressor, we augment the linear regressor as follows. The classifier function is of the form

$$f(\boldsymbol{x}) = \beta_0 + \sum_{i=1}^{d} \beta_i x_i + \sum_{ij} \beta_{ij} x_i x_j$$

The second summation introduces the quadratic dependency.

We can stack the information as follows. Let $\boldsymbol{x}_i = [x_1, x_2]^T$training data point. (This also works in higher dimensions, but we will be specific here.) Form a row of the data matrix **X** as

$$[1 \;\; x_1 \;\; x_2 \;\; x_1^2 \;\; x_1 x_2 \;\; x_2^2 \,]$$

then form the **X** matrix as the stack of such rows for all training data. The $\hat{B}$ matrix has K columns, each of the form

$$\begin{matrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \beta_{11} \\ \beta_{12} \\ \beta_{21} \end{matrix}$$

So that there are six unknowns. Given the **Y** matrix as before, the problem has the same structure as before. The least-squares estimate of $\hat{B}$ is

$$\hat{B} = (X^T X)^{-1} X^T Y.$$

Problem 2: For the data described in **Problem 1**, train the regression coefficient matrix $\hat{B}$. Determine the classification error rate on the training data and 10,000 points of test data (as before) and fill the corresponding row of the results table. Plot the classification regions as before.

K-Nearest Neighbor Classifier

The k-nearest neighbor classifier operates on the principle that you start to look like the people that you hang around with. That is, the classification of a vector $\boldsymbol{x}$ should be represented by classifications of neighbors near to it.

Let $N_k(\boldsymbol{x})$ be the set of the k training vectors $\boldsymbol{x_i}$ nearest to the point $\boldsymbol{x}$. Each training input $\boldsymbol{x_i}$ has a corresponding output (classification) value y_i associated with it. The k-nearest neighbor rule forms a classification function

$$f(\boldsymbol{x}) = \frac{1}{k}\sum_{x_i \in N_k(\boldsymbol{x})} y_{\boldsymbol{i}} .$$

That is, it computes the average of the y values for the k nearest training points to $\boldsymbol{x}$. If $f(\boldsymbol{x}) > 0.5$, then the classifier decides that $\boldsymbol{x}$ is in *class 1*. If $f(\boldsymbol{x}) < 0.5$, then the classifier decides that $\boldsymbol{x}$ is in *class 0*.

Problem 3: For the data set described in problem 2, program a k-nearest neighbor function. Make it so that you can change the value of k. Use your k-nearest neighbor function for classification of the training data and 10,000 points of test data for k = 1, k = 5, and k = 15. Comment on the probability of error on the training data when k = 1. Plot the classification regions. Record the probability of classification error for test and training data on the table.

- Turn in your programs, a description of the documents you used to estimate M(x; y), and the final decrypted string.

- Submit your codes along with a technical report that contains an introduction about the project, a section on the results (with figures), and a conclusion section.

- Prepare a set of slides with your teammate (if you have any) and be prepared to present your work in class for 15 minutes.

Good Luck