ECE 592 Homework 2

Chekad Sarami (csarami@ncsu.edu)
Kudiyar Orazymbetov (korazym@ncsu.edu)
Nico Casale (ncasale@ncsu.edu)

September 27, 2017

Note that the entries are links.

Contents

List of Figures

Listings

1 Problem 1: Bayesian Classification

1.1 Part A: pdf of a Gaussian Mixture

The pdf of a mixture Gaussian source is given by the following steps. Using the example given in class, where there are two classes, blue and red, we have

$$f(x, Class = red) = \sum_{i=1}^{5} f_i(x, red) \tag{1.1}$$

$$f_i(\boldsymbol{x}) = \sum_{ri=1}^{5} (2\pi\boldsymbol{\Sigma})^{-\frac{1}{2}} \, e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu_{ri}})'\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu_{ri}})} \tag{1.2}$$

$$f(x, Class = blue) = \sum_{i=1}^{5} g_i(x, blue) \tag{1.3}$$

$$g_i(\boldsymbol{x}) = \sum_{bi=1}^{5} (2\pi\boldsymbol{\Sigma})^{-\frac{1}{2}} \, e^{-\frac{1}{2}}(\mathbf{x} - \boldsymbol{\mu_{bi}})'\boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu_{bi}}) \tag{1.4}$$

1.2 Part B: Posterior Probability

Assuming equiprobable classes red and blue, the posterior probability is given as:

$$Pr(class = red|x) = \frac{f(x, Class = red)}{f(x, Class = red) + f(x, Class = blue)} \tag{1.5}$$

1.3 Part C: Code for Optimal Decision Regions

1.3.1 Outline of application of Bayes decision region

From the above equation, given coordinates we can calculate the probability of red or blue class assuming that we have 2 clusters of blue and red color. If they are both normally distributed with equal covariance matrix, then we can exploit Bayesian decision boundary which compares probabilities of both colors at a given coordinate and selects the one with higher probability. Mathematically, for a given $\mathbf{x}$ if

$$Pr(class = red|x) > Pr(class = blue|x) \tag{1.6}$$

then we decide that our data point belongs to red color.

1.3.2 Results with Bayes decision boundary

If we look our decision boundary in 2D, then we should get a linear line separating the two class.

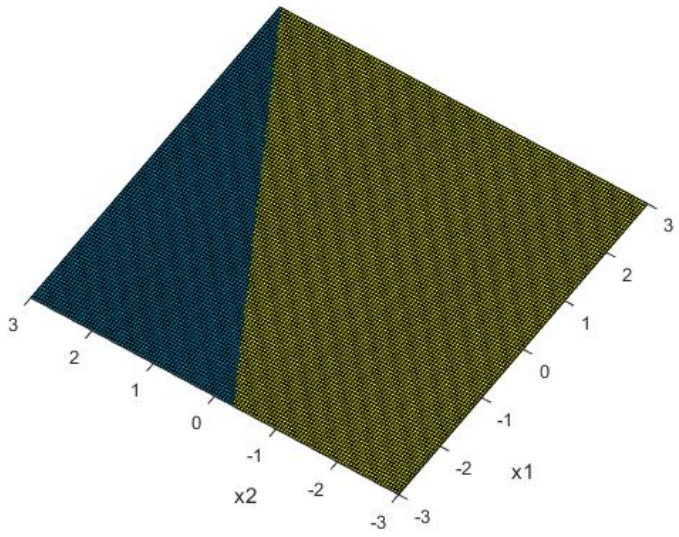

Figure 1.1: Bayes Classifier with same covariance matrix in 2D.

In 3D we should observe overlap of two Gaussian distributions and we should select the most probable one when one decreases in pdf while another increases in pdf.

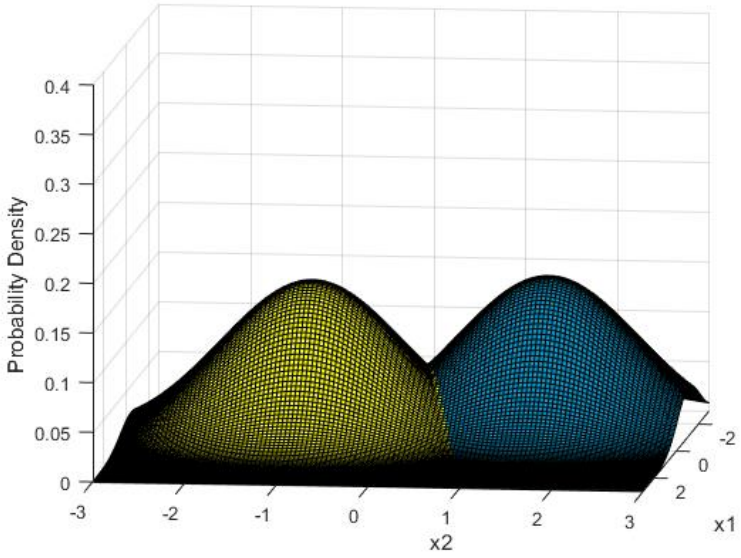

Figure 1.2: Bayes Classifier with same covariance matrix in 3D.

1.4 Part D: Quadratic Discriminant Analysis (QDA)

In QDA, we do not assume that the class covariances are equal. This leads to a quadratic decision boundary, as we will show in the following sections.

1.4.1 pdf for QDA

The pdf of the two classes, where each is composed of a single Gaussian component can be represented as the sum of two multivariate Gaussian distributions:

$$f_{\mathbf{X}}(x_1, x_2) = \sum_{k=1}^{2} \sqrt{|2\pi\mathbf{\Sigma}_k|}^{-1} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^{\mathrm{T}}{\mathbf{\Sigma}_k}^{-1}(\mathbf{x}-\boldsymbol{\mu}_k)\right) \tag{1.7}$$

Note that $\mathbf{\Sigma}_k$ is positive definite and has a non-zero determinant for each class C_k in $k = [1, 2]$ (in the square root.) μ_k is the mean of each class.

1.4.2 Posterior Probability for QDA

The posterior probability for QDA is derived by first using Bayes' Theorem as

$$p(C_k|x) = \frac{p(C_k)p(x|C_k)}{p(x)} \tag{1.8}$$

Where x is the vector of features. In the 2-D case, there are only 2 features, x_1 and x_2. The numerator of Eq. (1.8) is equivalent to the joint probability model, $p(C_k, x_1, x_2)$, which is equivalent to

$$\begin{aligned} p(C_k, x_1, x_2) &= p(x_1, x_2, C_k) \\ &= p(x_1|x_2, C_k)p(x_2, C_k) \\ &= p(x_1|x_2, C_k)p(x_2|C_k)p(C_k) \end{aligned} \tag{1.9}$$

Assuming that each feature (in the x and y dimension) is independent,

$$p(C_k|x_1, x_2) \propto p(C_k, x_1, x_2) \propto p(C_k)p(x_1|C_k)p(x_2|C_k) \tag{1.10}$$

Incorporating the probability of x,

$$p(C_k|x_1, x_2) = \frac{1}{p(x)}p(C_k)p(x_1|C_k)p(x_2|C_k) \tag{1.11}$$

Simplifying,

$$\begin{aligned} p(C_k|x) &= \frac{1}{p(x)}p(C_k)f_k(x) \\ &= \frac{p(C_k)f_k(x)}{\sum_{k'=1}^{2} p(C_{k'})f_{k'}(x)} \end{aligned} \tag{1.12}$$

The Bayesian classifier will choose the C_k that maximizes the conditional probability $p(C_k|x)$. To see that the decision boundary is quadratic, consider the log-likelihood ratio of the two class pdf's:

$$\text{likelihood ratio} = \frac{\sqrt{|2\pi\Sigma_{k=1}|}^{-1} \exp\left(-\frac{1}{2}(x-\mu_{k=1})^T\Sigma_{k=1}^{-1}(x-\mu_{k=1})\right)}{\sqrt{|2\pi\Sigma_{k=0}|}^{-1} \exp\left(-\frac{1}{2}(x-\mu_{k=0})^T\Sigma_{k=0}^{-1}(x-\mu_{k=0})\right)} < t \tag{1.13}$$

Where t is the threshold where the numerator or denominator starts to be larger than the other. Taking the natural log of both sides,

$$(x-\mu_0)^T\Sigma_0^{-1}(x-\mu_0) + \ln|\Sigma_0| - (x-\mu_1)^T\Sigma_1^{-1}(x-\mu_1) - \ln|\Sigma_1| \; > \; T \tag{1.14}$$

Where $T = \ln(t)$. Note that the inequality flips because we divided by $-\frac{1}{2}$. Multiplying out,

$$x^T\Sigma_0^{-1}x - x^T\Sigma_1^{-1}x + x^T\Sigma_0^{-1}\mu_0 - x^T\Sigma_1^{-1}\mu_1 > T - \mu_0^T\Sigma_0^{-1}\mu_0 + \mu_1^T\Sigma_0^{-1}\mu_1 - \ln|\Sigma_0| + \ln|\Sigma_1| \tag{1.15}$$

Note that the quadratic elements of this inequality are the two leftmost terms.

Sources:

- Dr. Baron's supplement on Bayesian Analysis.
- Quadratic Classifiers on Wikipedia
- Linear Discriminant Analysis on Wikipedia
- Naive Bayes Classifiers on Wikipedia

1.4.3 Code for Optimal Decision Regions under QDA

The figure below illustrates the results of a Bayesian classifier without the assumption that the covariance between classes is equal. Under this premise, a quadratic decision boundary is needed to separate the two classes in such a way that $p(C_k|x)$ is maximized for each class.

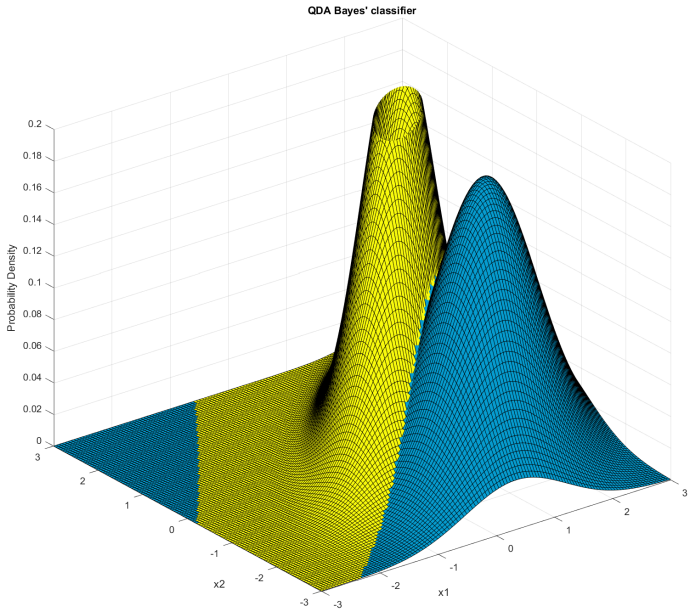

Figure 1.3: Bayes Classifier with QDA.

2. Nearest neighbors using a simple kernel. Recall that the nearest neighbors (NN) classifier takes a "plurality vote" among the K training points nearest to our test point. In this problem, you will modify the implementation of NN such that the nearest neighbor receives weight (K)/[K(K+1)/2], the second nearest weight (K-1)/[K(K+1)/2], down to the K'th nearest neighbor whose weight is (1)/[K(K+1)/2]. (Note that the weights sum to 1, because the values in the numerators of the weights, i.e., 1+2+…+K, sum to K(K+1)/2.) Compare the original NN to the modified version with different weights. You may want to compare them on the class pdf's defined in Problem 1.

Solution: We added a column containing the weights to the top k predicted labels and repredicted the class labels.

```
W = (num_neighbors:-1:1)/(num_neighbors*(num_neighbors+1)/2);

....


A = cat(2,train(3, neighbors)', W');
%A = cat(2,class_samples(neighbors), W');
sum0=0;sum1=0;
for i = 1:num_neighbors
    if A(i,1)==0
        sum0 = sum0+A(i,2);
    else
        sum1 = sum1+A(i,2);
    end
end
class_predicted=(sum1/sum0 > 1); % NN classifier
test_NNK(n1)=class_predicted; % store classification
```

we have modularized the classification.m code into four matlab codes.

```
Main.m : driver code
Knn: regular knn with majority voting
knnWithKernel.m: Knn with kernel
dataSetCreator.m: this creates data set with multivariate Gausisan distribution, given, number of
clusters, N ( size of complete dataset, and cluster_variations
```

```
% main.m
% This is used to compare kNN and the kerneled one
%% creating dataset and spliting it into training and testing sets
clear % often useful to clean up thework space from old variables
close all
num_clusters=5; % number of components (clusters) in mixture model
N=6*800; % total number of samples of training data
cvs = .2:.2:1; % different cluster varations
nns = 3:2:9;
avgErrors = zeros(length(cvs), length(nns),2);
repeat = 20;
for i=1:length(cvs)
    for j=1:length(nns)
        err0tot =0;
        err1tot=0;
        [train, test] = dataSetCreator(num_clusters, N,cvs(i));
        for it =1:repeat
            err0tot= err0tot +           knn(nns(j), train, test);
            err1tot= err1tot+  knnWithKernel(nns(j), train, test)
        end
        avgErrors(i,j,1) = err0tot/repeat;
        avgErrors(i,j,2) = err1tot/repeat;
    end
end
```

The code main.m returns average error for each method. Running above (for 20 times repetition) code we have:

kNN with simple kernel errors

```
avgErrors(:,:,2) =
   0.1425   0.1931   0.1812   0.3056
   0.3863   0.2619   0.4150   0.2956
   0.2919   0.4219   0.2969   0.3606
   0.4600   0.4306   0.3656   0.4494
   0.4262   0.3894   0.4462   0.3963

avgErrors(:,:,1) =

   0.1344   0.1881   0.1781   0.2900
   0.3694   0.2581   0.4025   0.2856
   0.2669   0.4313   0.2875   0.3513
   0.4437   0.4206   0.3619   0.4288
   0.4300   0.3844   0.4375   0.3863

sum(sum(avgErrors))

ans(:,:,1) =

   6.7363


ans(:,:,2) =

   6.9162
```

We see that kNN with kernel has larger average error. Also the difference, exept two instances, kNN outperformed kNN with kernel.

```
avgErrors(:,:,2)-avgErrors(:,:,1)

ans =

   0.0081   0.0050   0.0031   0.0156
   0.0169   0.0037   0.0125   0.0100
   0.0250  -0.0094   0.0094   0.0094
   0.0163   0.0100   0.0037   0.0206
```

-0.0038 0.0050 0.0087 0.0100

As we can see, kNN performs better than kNN with the simple kernel the class pdf's defined in Problem 1.

2 Appendix A: Code Listings

2.1 Code for Problem 1.c

2.1.1 Hw2_Pr1.m

Listing 1: Code to solve problem 1.c.

```
%{
ECE 592 Homework 2
Chekad Sarami

Kudiyar (Cody) Orazymbetov
korazym@ncsu.edu

Nico Casale
ncasale@ncsu.edu
%}
%%
clear;
%setup();
fprintf('ECE 592 hw 2\n');
fprintf(strcat(datestr(now),'\n'));
numClusters=5; % number of components (clusters) in mixture model
N=200; % total number of samples of training data
grid=-3:0.05:3; % test data grid for each dimension
%%
Gmean=randn(2,numClusters); % locations of centers of clusters for green class
Rmean=randn(2,numClusters); % red class
clusterVariance = [1 0; 0 1];
[X1, X2] = meshgrid(grid, grid);
posProb = zeros(size(X1));
%x = [];
%for k =1:5
    for i = 1:length(grid)
        for j = 1:length(grid)
            G(i,j) = mvnpdf([grid(i) grid(j)],Gmean(:,1)',clusterVariance);
            R(i,j) = mvnpdf([grid(i) grid(j)],Rmean(:,2)',clusterVariance);
            posProb(i,j) = max(G(i,j), R(i,j));
         end
    end
mean = 1/2*(Gmean(:,1) + Rmean(:,1));
f = @(i,j) (1/2*(Gmean(:,1) + Rmean(:,1)) ...
                - log(G(i,j)/R(i,j))*(Gmean(:,1) - Rmean(:,1))/sum((Gmean(:,1) - Rmean(:,1)).^2)
                    );
surf(grid,grid,posProb, (G>R)/2);
%w = @(x,y) dot((Gmean(:,1) - Rmean(1,:))',[x; y] - 1/2*(Gmean(:,1) + Rmean(:,1)));
% x = mean(1);
% y = mean(2);
alpha = 0.5;
caxis([min(posProb(:))-.5*range(posProb(:)),max(posProb(:))]);
axis([-3 3 -3 3 0 .4])
xlabel('x1'); ylabel('x2'); zlabel('Probability Density');
```

2.2 Code for Problem 1.d

2.2.1 `hw2_1d.m`

Listing 2: Code to solve problem 1.d.

```
%{
ECE 592 HW 2

Kudiyar (Cody) Orazymbetov
korazym@ncsu.edu

Nico Casale
ncasale@ncsu.edu

Chekad Sarami
csarami@ncsu.edu

%}
%%
clear;

global imagesFolder
imagesFolder = '../images/';
addpath(imagesFolder);

fprintf('ECE 592 Homework 2\n');
fprintf(strcat(datestr(now),'\n'));
numClusters=1; % number of components (clusters) in mixture model
N=200; % total number of samples of training data
grid=-3:0.05:3; % test data grid for each dimension

%%
Gmean=randn(2,numClusters); % locations of centers of clusters for green class
Rmean=randn(2,numClusters); % red class
cov1 = rand(1);
clusterVariance1 = [1 cov1; cov1 1];
cov2 = rand(1);
clusterVariance2 = [1 cov2; cov2 1];
[X1, X2] = meshgrid(grid, grid);
posProb = zeros(size(X1));

for i = 1:length(grid)
   for j = 1:length(grid)
      G(i,j) = mvnpdf([grid(i) grid(j)],Gmean(:,1)',clusterVariance1);
      R(i,j) = mvnpdf([grid(i) grid(j)],Rmean(:,1)',clusterVariance2);
      posProb(i,j) = max(G(i,j), R(i,j));
   end
end

surf(grid,grid,posProb, (G>R)/2);

caxis([min(posProb(:))-.5*range(posProb(:)),max(posProb(:))]);

axis([-3 3 -3 3 0 .3])
title('QDA Bayes'' classifier');
xlabel('x1'); ylabel('x2'); zlabel('Probability Density');
```

```
52
53 %% save images
54 file = sprintf('qda_bayes');
55 file = strcat(imagesFolder, file);
56 print(file, '-dpng');
```