成绩

Pattern Recognition and Machine Learning Experiment Report

院（系）名称 自动化科学与电气工程学院

专 业 名 称 自动化

学 生 学 号 15071129

学 生 姓 名 苗子琛

2018 年 4 月

Experiment2 Synthetical Design of Bayesian Classifier

1 Introduction

Linear perceptron allows us to learn a decision boundary that would separate two classes. They are very effective when there are only two classes, and they are well separated. Such classifiers are referred to as discriminative classifiers.

In contrast, generative classifiers consider each sample as a random feature vector, and explicitly model each class by their distribution or density functions. To carry out the classification, the likelihood function should be computed for a given sample which belongs to one of candidate classes so as to assign the sample to the class that is most likely. In other words, we need to compute $p(\omega_i|X)$ for each class ω_i. However, the density functions provide only the likelihood of seeing a particular sample, given that the sample belongs to a specific class. i.e., the density functions can be provided as $p(X|\omega_i)$. The Bayesian rule provides us with an approach to compute the likelihood of the class for a given sample, from the density functions and related information.

2 Principle and Theory

The essence of the Bayesian approach is to provide a mathematical rule explaining how you should change your existing beliefs in the light of new evidence. In other words, it allows us to combine new data with their existing knowledge or expertise. The canonical example is to imagine that a precocious newborn observes his first sunset, and wonders whether the sun will rise again or not. He assigns equal prior probabilities to both possible outcomes and represents this by placing one white and one black marble into a bag. The following day, when the sun rises, the child places another white marble in the bag. The probability that a marble plucked randomly from the bag will be white (i.e., the child's degree of belief in future sunrises) has thus gone from a half to two-thirds. After sunrise the next day, the child adds another white marble, and the probability (and thus the degree of belief) goes from two-thirds to three-quarters. And so on. Gradually, the initial belief that the sun is just as likely as not to rise each morning is modified to become a near-certainty that the sun will always rise.

In terms of classification, the Bayesian theorem allows us to combine prior probabilities, along with observed evidence to arrive at the posterior probability. More or less, conditional probabilities represent the probability of an event occurring given evidence. According to the Bayesian Theorem, if $P(\omega_i)$, $p(X|\omega_i), i = 1,2,\dots,c.$ and X are known or given, the posterior probability can be derived as follows,

$$p(\omega_i|X) = \frac{p(X|\omega_i)P(\omega_i)}{\sum_{j=1}^{c} p(X|\omega_j)P(\omega_j)} \quad i = 1,2,\dots,c \tag{1}$$

Let the series of decision actions as $\{a_1, a_2, \dots, a_c\}$, the conditional risk of decision action a_i can be computed by

$$R(a_i|X) = \sum_{j=1, j\neq i}^{c} \lambda(a_i, \omega_j)\, P(\omega_j|X) \quad i = 1,2,\dots,c \tag{2}$$

Thus, the minimum risk Bayesian decision can be found as

$$a_k^* = Argmin_i\, R(a_i|X) \quad i = 1,2,\dots,c \tag{3}$$

3 Objective

The goals of the experiment are as follows:

(1) To understand the computation of likelihood of a class, given a sample.

(2) To understand the use of density/distribution functions to model a class.

(3) To understand the effect of prior probabilities in Bayesian classification.

(4) To understand how two (or more) density functions interact in the feature space to decide a decision boundary between classes.

(5) To understand how the decision boundary varies based on the nature of density functions.

4 Contents and Procedure

Stage 1:

According to principle and theory above, I devise a Bayesian classifier based on Gaussian normal distribution in MATLAB, **Bayes_multi_classes.m.**

Then I entry data provided, Ω_1 and Ω_2, into program with their prior probability. To compute conditional probability for ω_i, I use data to estimate parameters of Gaussian normal distribution, mean μ and standard deviation σ. The result was show in fig.1.

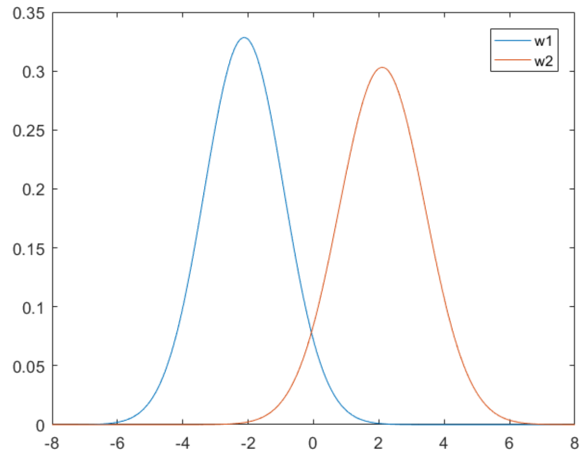

Fig.1. conditional probability of w1 and w2

Next, we can compute the posterior probability of $\mathrm{w_i}$ according to equation (1), the result is shown in fig.2.

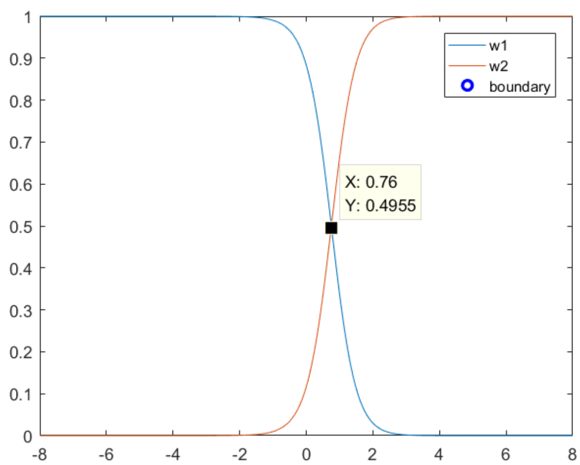

Fig.2. posterior probability of w1 and w2

Here, we get decision boundary without considering decision loss. The boundary, x=0.76, is minimum error Bayesian boundary.

After that, we decide decision boundary according to risk for different decision action, which is computed based on the loss parameters for different decision (table 1). The result is shown in fig.3. The decision boundary, x=1.41, guarantees min risk of classification.

Table 1. the loss parameters for different decision

Real class		ω_1	ω_2	
Decision action	a_1	0	1	Loss parameters
	a_2	6	1	

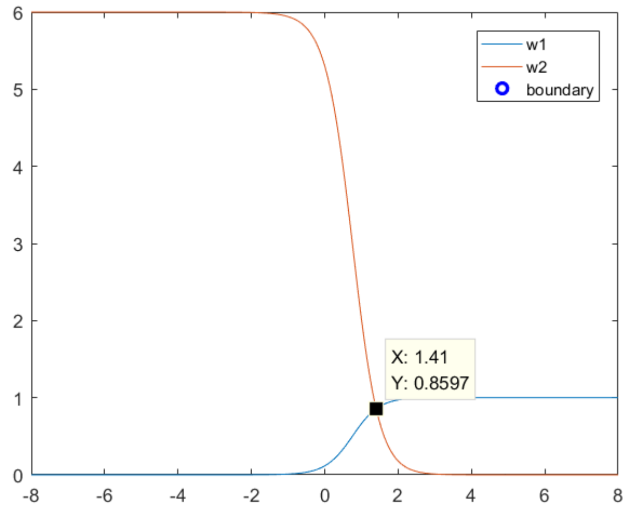

Fig.3. decision boundary for min risk

Stage 2

Firstly, I create dataset with 3 classes and 2 dimensions by Gaussian random numbers. Each class has 100 samples. Datapoints are shown in fig.4.

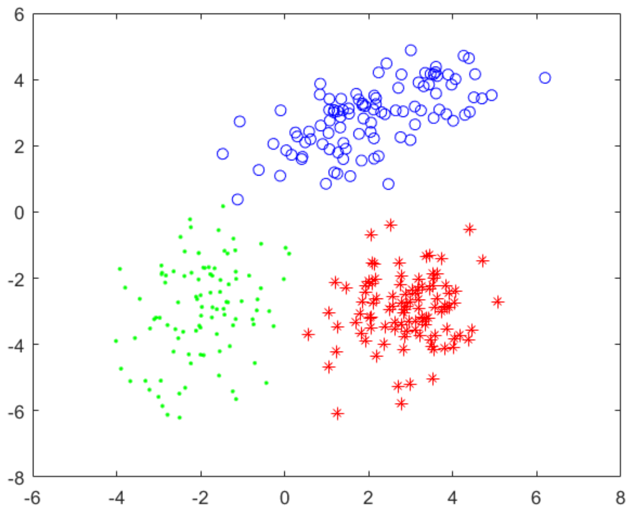

Fig.4. dataset for stage 2

After that I assign prior probability for w_1, w_2, w_3 with 0.7, 0.2 and 0.1. Then just like in stage 1, we can compute conditional probability of w_i by parameter estimation. The result is shown in fig.5.

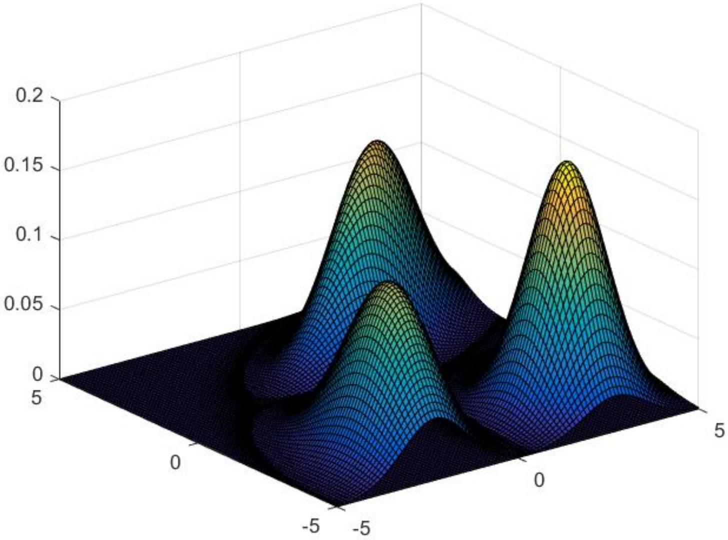

Fig.5. conditional probability for 3 classes

Then, posterior probability is computed according to equation (1). The result is shown in fig.6.

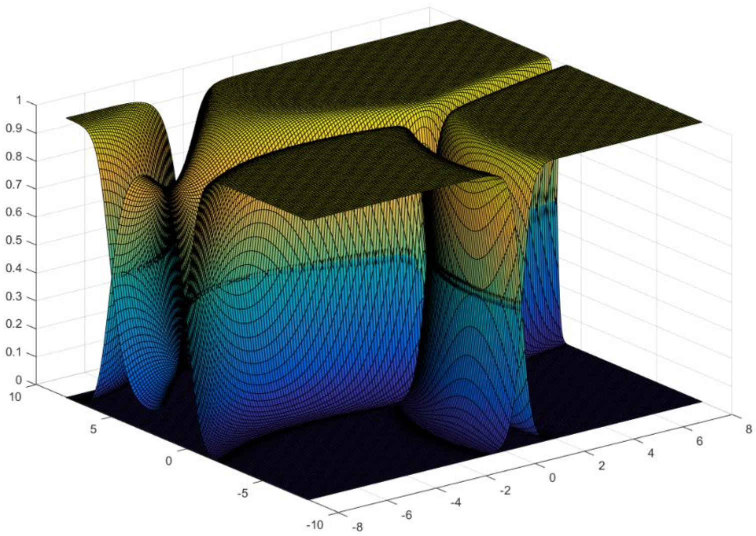

Fig.6. Posterior probability for 3 classes

Then we can determine minimum error Bayesian boundary, which is shown in fig.7.

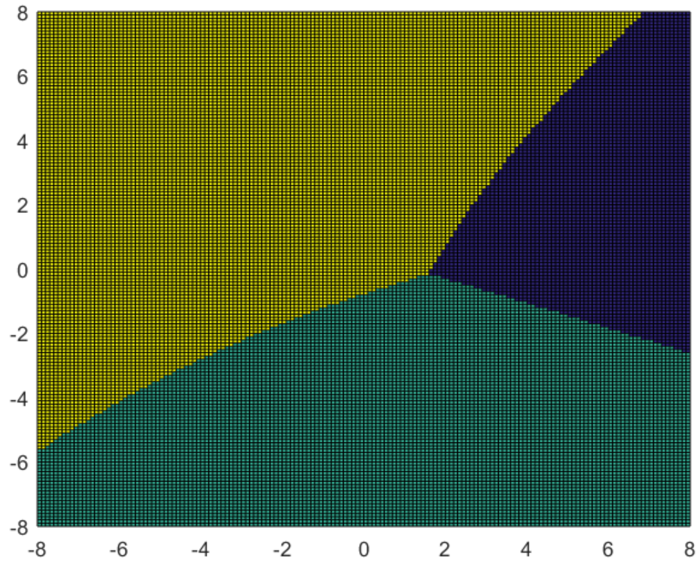

Fig.7. min error decision boundary

After that, I calculate Risk for each decision based on decision loss parameters in table 2. The result is shown in fig.8. The decision boundary was shown in fig.9.

Table 2. the loss parameters for different decision

<table>
<tr><td colspan="2">Real class</td><td>ω_1</td><td>ω_2</td><td>ω_3</td><td rowspan="4">Loss parameters</td></tr>
<tr><td rowspan="3">Decision action</td><td>a_1</td><td>0</td><td>2</td><td>1</td></tr>
<tr><td>a_2</td><td>2</td><td>0</td><td>3</td></tr>
<tr><td>a_3</td><td>3</td><td>5</td><td>0</td></tr>
</table>

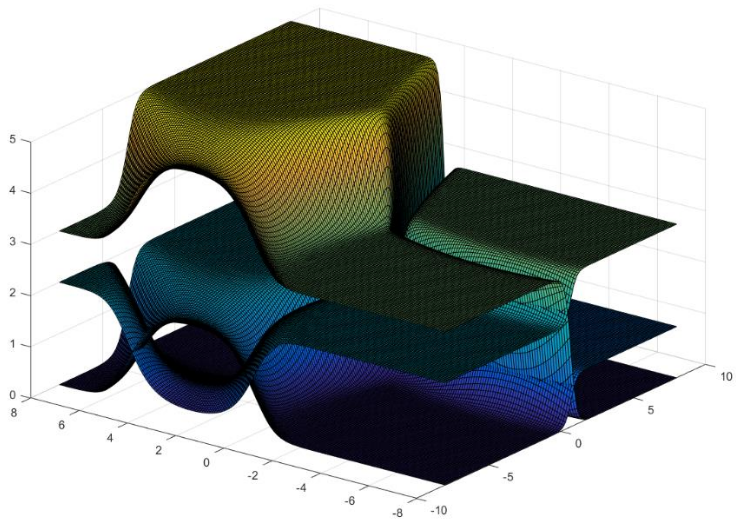

Fig.8. Risk for three classes

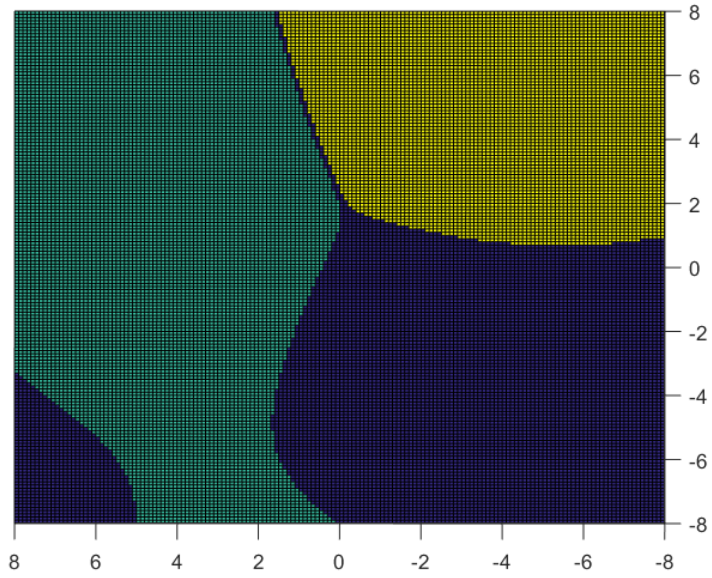

Fig.9. min risk decision boundary

As for intrinsic relationship between classifier for two classes and one for multiple classes, I think the basic ideas of them, i.e. to provide a mathematical rule explaining how you should change your existing beliefs in the light of new evidence, are same. And all estimations and computation processes are same. The only difference is that classifiers for multiple classes have more subregions to determine and they are more difficult to compute due to multivariate probability.

5 Experience

Conducting this experiment enhanced my understanding over the concept of Bayesian classifier. Bayesian classifier is a powerful method to classify samples on the basis of both prior belief and new evidence. It is easy to implement but is effective.

6 Code

Bayes_two_classes.m:

```
%data input
X1 = [-3.9847 -3.5549 -1.2401 -0.9780 -0.7932 -2.8531 -2.7605 -3.7287 -3.5414 -2.2692 -3.4549 -3.0752 -3.9934    -0.9780 -1.5799 -1.4885 -0.7431 -0.4221 -1.1186 -2.3462 -1.0826 -3.4196 -1.3193 -0.8367 -0.6579 -2.9683];
X2 = [2.8792 0.7932 1.1882 3.0682 4.2532 0.3271 0.9846 2.7648 2.6588];
P_w1 = 0.9;
P_w2 = 0.1;
Lambda_a1_w2 = 1;
Lambda_a2_w1 = 6;

x = [-8:0.01:8];
```

```matlab
%compute and plot conditional probabilty density
std1 = std(X1);
mean1 = mean(X1);
std2 = std(X2);
mean2 = mean(X2);
p_X_w1 = (1/(sqrt(2*pi)*std1))*exp(-1*(x-mean1).^2/(2*std1^2));
p_X_w2 = (1/(sqrt(2*pi)*std2))*exp(-1*(x-mean2).^2/(2*std2^2));
figure(1);
plot(x,p_X_w1);hold on;
plot(x,p_X_w2);

% compute and plot posterior probablity
p_X_w1_joint = p_X_w1*P_w1;
p_X_w2_joint = p_X_w2*P_w2;
p_w1_X = p_X_w1_joint./(p_X_w1_joint+p_X_w2_joint);
p_w2_X = p_X_w2_joint./(p_X_w1_joint+p_X_w2_joint);
figure(2);
plot(x,p_w1_X);hold on;
plot(x,p_w2_X);

%Min error Bayes decision boundary
for i = 1:length(x)
    if(p_w1_X(i)<p_w2_X(i))
        disp(['the min error decision boundary is:',num2str(x(i))]);
        plot(x(i),p_w1_X(i),'ob', 'LineWidth', 2);
        break;
    end
end

%Min risk Bayes decision boundary
Risk_a1 = p_w2_X*Lambda_a1_w2;
Risk_a2 = p_w1_X*Lambda_a2_w1;
figure(3);
plot(x,Risk_a1);hold on;
plot(x,Risk_a2);
for i = 1:length(x)
    if(Risk_a1(i)>Risk_a2(i))
        disp(['the min risk decision boundary is:',num2str(x(i))]);
        plot(x(i),Risk_a1(i),'ob', 'LineWidth', 2);
        break;
    end
end
```

Bayes_multi_classes.m

```
%% generate dataset: 2 dimensional 3 classes
mean1_construct = [-2 -3];cov1_construct = [1 0.5; 0.5 2];
mean2_construct = [2 3];    cov2_construct = [2 0.75; 0.75 1];
mean3_construct = [3 -3]; cov3_construct = [1 0.25; 0.25 1];
X1 = mvnrnd(mean1_construct, cov1_construct, 100); %generate the dataset by multivariate normal random numbers
X2 = mvnrnd(mean2_construct, cov2_construct, 100);
X3 = mvnrnd(mean3_construct, cov3_construct, 100);
figure(1);
plot(X1(:, 1), X1(:, 2), '.g');hold on;
plot(X2(:, 1), X2(:, 2), 'ob');hold on;
plot(X3(:, 1), X3(:, 2), '*r');

%% assign the prior probablity
P_w1 = 0.7;
P_w2 = 0.2;
P_w3 = 0.1;

[x, y] = meshgrid(-8 : 0.1 : 8);
%% compute and plot conditional probablity
mean1 = mean(X1); cov1 = cov(X1);
mean2 = mean(X2); cov2 = cov(X2);
mean3 = mean(X3); cov3 = cov(X3);
P_X_w1 = reshape(mvnpdf([x(:), y(:)], mean1, cov1),size(x));
P_X_w2 = reshape(mvnpdf([x(:), y(:)], mean2, cov2),size(x));
P_X_w3 = reshape(mvnpdf([x(:), y(:)], mean3, cov3),size(x));
figure(2);
surf(x,y,P_X_w1);hold on;
surf(x,y,P_X_w2);hold on;
surf(x,y,P_X_w3);

%% compute posterior probablity and make min error decision boundary
p_X_w1_joint = P_X_w1*P_w1;
p_X_w2_joint = P_X_w2*P_w2;
p_X_w3_joint = P_X_w3*P_w3;
p_w1_X = p_X_w1_joint./(p_X_w1_joint+p_X_w2_joint+p_X_w3_joint);
p_w2_X = p_X_w2_joint./(p_X_w1_joint+p_X_w2_joint+p_X_w3_joint);
p_w3_X = p_X_w3_joint./(p_X_w1_joint+p_X_w2_joint+p_X_w3_joint);
figure(3);
surf(x,y,p_w1_X);hold on;
surf(x,y,p_w2_X);hold on;
surf(x,y,p_w3_X);
```

```
Region_result = zeros(size(x));
for i = 1:size(x,1)
    for j = 1:size(x,1)
        [~,Class] = min([p_w1_X(i,j) p_w2_X(i,j) p_w3_X(i,j)]);
        Region_result(i,j) = Class;
    end
end
figure(4);
surf(x,y,Region_result);

%% compute posterior probablity and make min error decision boundary
% define risk matrix
Lambda = [0 2 1; 2 0 3; 3 5 0];
Risk1 = Lambda(1, 1) * p_w1_X + Lambda(1, 2) * p_w2_X + Lambda(1, 3) * p_w3_X;
Risk2 = Lambda(2, 1) * p_w1_X + Lambda(2, 2) * p_w2_X + Lambda(2, 3) * p_w3_X;
Risk3 = Lambda(3, 1) * p_w1_X + Lambda(3, 2) * p_w2_X + Lambda(3, 3) * p_w3_X;
figure(5);
surf(x,y,Risk1);hold on;
surf(x,y,Risk2);hold on;
surf(x,y,Risk3);

Region_result = zeros(size(x));
for i = 1:size(x,1)
    for j = 1:size(x,1)
        [~,Class] = min([Risk1(i,j) Risk2(i,j) Risk3(i,j)]);
        Region_result(i,j) = Class;
    end
end
figure(6);
surf(x,y,Region_result);
```