北京航空航天大學
BEIHANG UNIVERSITY

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 模式识别与智能系统

学生学号 15031204

学生姓名 张家奇

2018年4 月30 日

2 Synthetical Design of Bayesian Classifier

2.1 Introduction

Linear perceptrons allow 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 $p(w_i|x)$ for each class w_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 $p(x|w_i)$. 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.

2.2 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 $p(w_i), p(x|w_i)$, i=1, 2, …, c, and X are known or given, the posterior probability can be derived as follows

$$P(w_i|X) = \frac{P(X|w_i)P(w_i)}{\sum_{i=1}^{c} P(X|w_j)P(w_j)}, i = 1,2, \dots, c$$

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

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

Thus the minimum risk Bayesian decision can be found as

$$a_k^* = Argmin_i(R(a_i|X))$$

2.3 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.

2.4 Contents and Procedures

Stage 1

I conducted this experiment in Matlab. In this stage I designed a Bayesian classifier for the classification of two classes, each of which subject to Gaussian distribution. The data provided in the experiments guide are then used to test my program.

The mean and standard deviation of the two classes' samples are calculated to derive the conditional probability of the two classes, with which multiplied by prior probability

produces the posterior distribution, from which a decision boundary of minimum error can be calculated. The plots of the conditional probability and posterior probability are as follows.

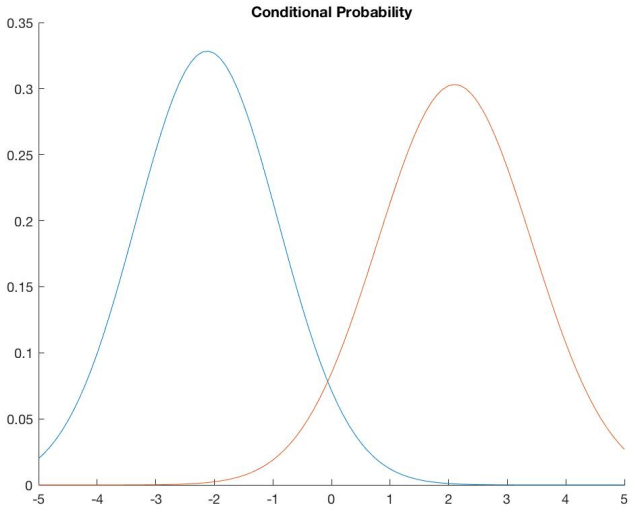

Figure 1 Conditional Probability

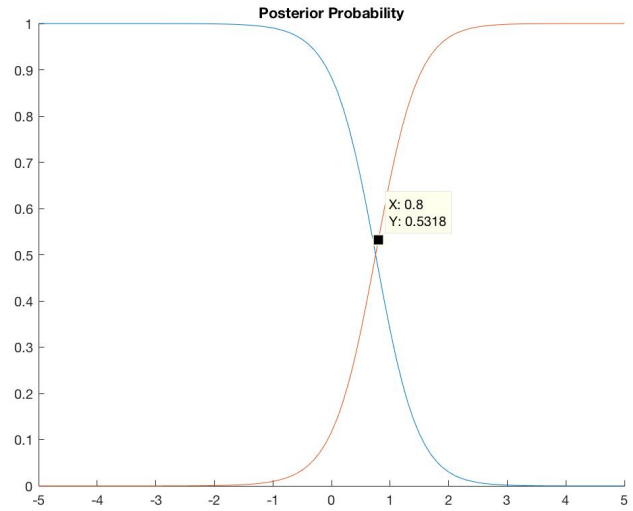

Figure 2 Posterior Probability

As shown in a figure above, a decision boundary of 0.8 guarantees the minimal probability of error.

Having taken the loss parameters for different parameters provided in the experiments guide into consideration, a plot of decision cost can be drawn, from which we can derive a different decision boundary at which the risk is minimal.

Real class		w1	w2	
Decision Action	a1	0	1	Loss Parameters
	a2	6	0	

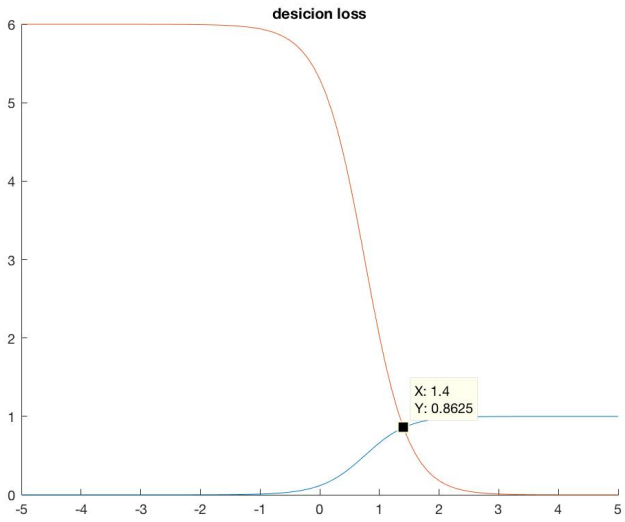

Figure 3 Decision Loss

As shown in the figure, a decision boundary of 1.4 guarantees minimal decision risk, under the loss parameters listed in the table above.

Stage 2

In this stage a multi-class high-dimensional Bayesian classifier was designed. For the convenience of visualization, a dataset of 3 classes in 2-dimensional space is used to test the classifier, with arbitrary prior probabilities and loss parameters.

To start with, a dataset in 2-dimensional space of 3 classes with 50 samples each is generated. Having calculated the mean vector and covariance matrix, we can derive their conditional probability.

Having determined the conditional probability, we multiply it with the prior probability of each class and divide it with the sum of conditional probabilities of all classes, and derive the posterior probability of each class.

The parameters and the plot of the dataset, the prior probability, the conditional probability, the posterior probability, as well as the decision boundary are as follows.

$$p(1) = 0.6, \mu_1 = [-2 \quad -3.5], \Sigma_1 = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 2 \end{bmatrix}$$

$$p(2) = 0.3, \mu_2 = [-1 \quad 4], \Sigma_2 = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}$$

$$p(3) = 0.1, \mu_3 = [1.5 \quad -1], \Sigma_3 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

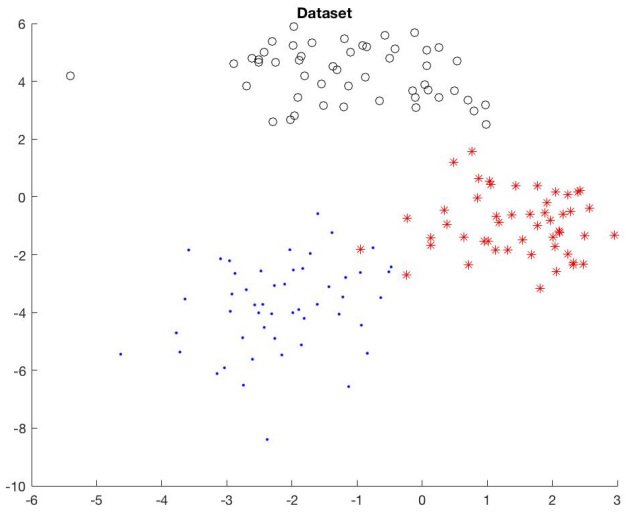

Figure 4 Dataset of 3 Classes

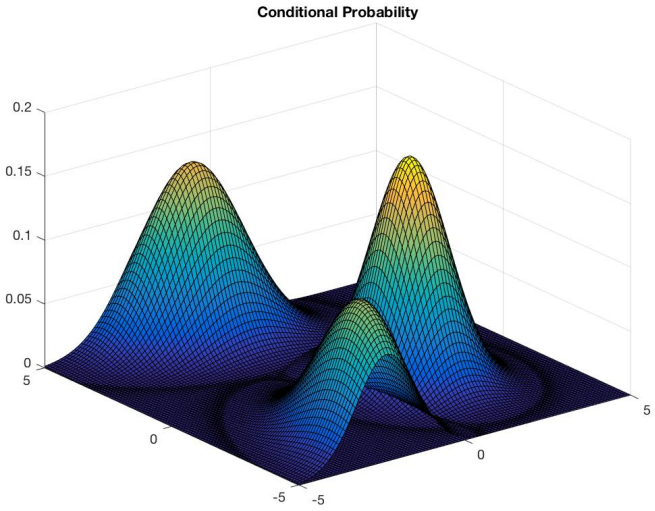

Figure 5 Conditional Probability of 3 classes

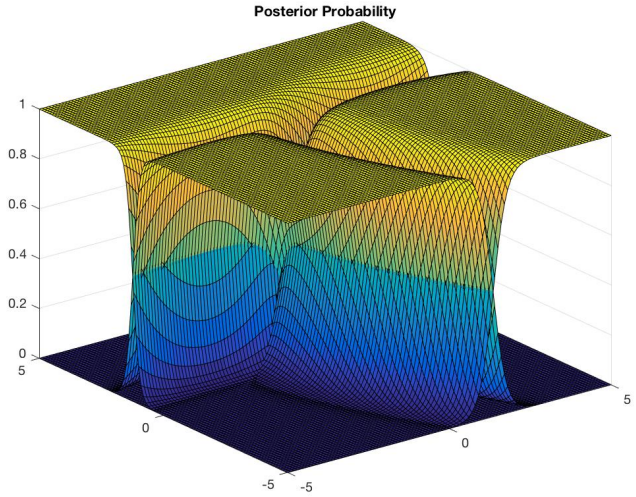

Figure 6 Posterior Probability

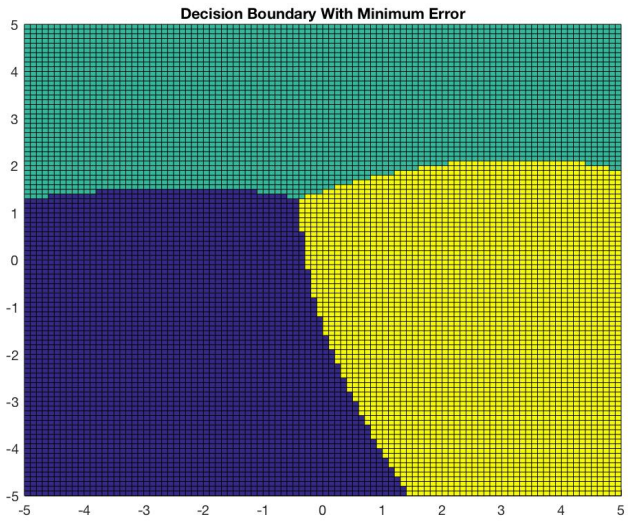

Figure 7 Decision Boundary with Minimum Error

Having considered the following decision loss, a plot illustrating the decision loss is derived, from which we can determine a new decision boundary which tends to minimize the decision risk, listed in the table below.

Real Class		w1	w2	w3	
Decision Action	a1	0	3	6	Loss Parameters
	a2	1	0	1	
	a3	8	3	0	

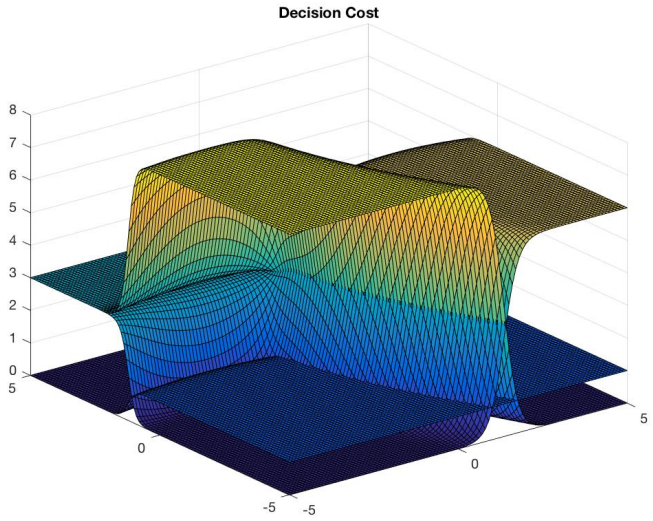

Figure 8 Decision Cost for Each Class

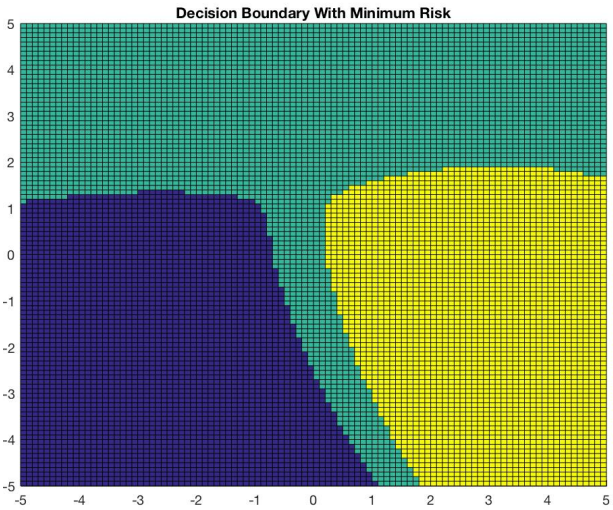

Figure 9 Decision Boundary with Minimal Risk

As seen in the plot above, the decision boundary shifted slightly after having considered the decision loss.

As for the relationships between the classifier between two class with those of multiple classes, I think they although share considerable similarities, some problems will emerge only after a certain level of dimension is reached. For datasets with higher dimensions, the samples in high-dimensional space become sparse, impeding the estimation of probability distributions.

2.5 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, low-computation and thus very effective.

However, in most circumstances where the dimension of feature space is very high, it will become increasingly difficult to estimate the conditional probability of each class due to lack of evidence. Detailed experiment results and specific code are on my github homepage: https://github.com/MagicOwO/BUAA-Pattern_Recognition_and_Machine_Learning

2.6 Codes

2.6.1 One-dimensional classifier

```
w1 = [-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];
w2 = [2.8792 0.7932 1.1882 3.0682 4.2532 0.3271 0.9846 2.7648 2.6588];
```

```
% Range of domain
x = [-5 : 0.1 : 5];

% Calculate the conditional probability
mean1 = mean(w1);
std1 = std(w1);
mean2 = mean(w2);
std2 = std(w2);

% Plot the conditional probability
conditional1 = (1 / (sqrt(2 * pi) * std1)) * exp(-1 * (x - mean1) .^ 2 / (2 * std1
^ 2));
conditional2 = (1 / (sqrt(2 * pi) * std2)) * exp(-1 * (x - mean2) .^ 2 / (2 * std2
^ 2));
figure(1)
hold on;
plot(x, conditional1)
plot(x, conditional2)
title('Conditional Probability')

% Calculate the prior probability
prior1 = 0.9;
prior2 = 0.1;

% Calculate the posterior probability
posterior1 = prior1 * conditional1;
posterior2 = prior2 * conditional2;
posterior_sum = posterior1 + posterior2;
posterior1 = posterior1 ./ posterior_sum;
posterior2 = posterior2 ./ posterior_sum;
figure(2);
hold on
title('Posterior Probability')
plot(x, posterior1)
plot(x, posterior2)
for i = 1 : length(x)
    if(posterior1(i) < 0.5)
        disp(['Decision Boundary with maximum accuracy: ', num2str(x(i))])
        break
    end
end

% Consider the decision risk
risk21 = 6;
risk12 = 1;
cost1 = posterior2 * risk12;
cost2 = posterior1 * risk21;
figure(3)
hold on
plot(x, cost1)
plot(x, cost2)
title('desicion loss')
for i = 1 : length(x)
    if(cost1(i) > cost2(i))
        disp(['Decision Boundary with minimal cost: ', num2str(x(i))])
        break
    end
```

```
end
```

2.6.2 Two-dimensional classifier

```
% Parameters
prior_prob1 = 0.6;
prior_prob2 = 0.3;
prior_prob3 = 0.1;
decision_cost = [0 1 8; 3 0 3; 6 1 0];

% Generate dataset and plot
mean1_init = [-2 -3.5];
cov1_init = [1 0.5; 0.5 2];
mean2_init = [-1 4];
cov2_init = [2 0; 0 1];
mean3_init = [1.5, -1];
cov3_init = [1 0; 0 1];
w1 = mvnrnd(mean1_init, cov1_init, 50);
w2 = mvnrnd(mean2_init, cov2_init, 50);
w3 = mvnrnd(mean3_init, cov3_init, 50);
figure(1)
hold on
plot(w1(:, 1), w1(:, 2), '.b')
plot(w2(:, 1), w2(:, 2), 'ok')
plot(w3(:, 1), w3(:, 2), '*r')
title('Dataset')

% Init grid
[x, y] = meshgrid(-5 : 0.1 : 5);
figure(1)
hold on

% Conditional probability parameters and plot
mean1 = mean(w1);
cov1 = cov(w1(:, 1), w1(:, 2));
mean2 = mean(w2);
cov2 = cov(w2(:, 1), w2(:, 2));
mean3 = mean(w3);
cov3 = cov(w3(:, 1), w3(:, 2));
figure(2)
cond_prob1 = reshape(mvnpdf([x(:), y(:)], mean1, cov1), size(x));
cond_prob2 = reshape(mvnpdf([x(:), y(:)], mean2, cov2), size(x));
cond_prob3 = reshape(mvnpdf([x(:), y(:)], mean3, cov3), size(x));
surf(x, y, cond_prob1)
hold on
surf(x, y, cond_prob2)
surf(x, y, cond_prob3)
title('Conditional Probability')

% Calculate posterior probability and plot
post_prob1 = prior_prob1 * cond_prob1;
post_prob2 = prior_prob2 * cond_prob2;
post_prob3 = prior_prob3 * cond_prob3;
post_prob_all = post_prob1 + post_prob2 + post_prob3;
post_prob1 = post_prob1 ./ post_prob_all;
post_prob2 = post_prob2 ./ post_prob_all;
post_prob3 = post_prob3 ./ post_prob_all;
```

```
figure(3)
surf(x, y, post_prob1)
hold on
surf(x, y, post_prob2)
surf(x, y, post_prob3)
title('Posterior Probability')

% Plot decision region with minimum error
region = calculate_decision_region(x, post_prob1, post_prob2, post_prob3, 'max');
figure(4)
hold on
surf(x, y, region)
title('Decision Boundary With Minimum Error')

% Calculate posterior probability with minimal risk and plot
cost1 = decision_cost(1, 1) * post_prob1 + decision_cost(2, 1) * post_prob2 +
decision_cost(3, 1) * post_prob3;
cost2 = decision_cost(1, 2) * post_prob1 + decision_cost(2, 2) * post_prob2 +
decision_cost(3, 2) * post_prob3;
cost3 = decision_cost(1, 3) * post_prob1 + decision_cost(2, 3) * post_prob2 +
decision_cost(3, 3) * post_prob3;
figure(5)
surf(x, y, cost1)
hold on
surf(x, y, cost2)
surf(x, y, cost3)
title('Decision Cost')

% Plot decision region with minimal risk
region = calculate_decision_region(x, cost1, cost2, cost3, 'min');
figure(6)
hold on
surf(x, y, region)
title('Decision Boundary With Minimum Risk')


function region = calculate_decision_region(x, prob1, prob2, prob3, option)
assert(nargin == 5);
region = zeros(size(prob1));
for i = 1 : size(x, 1)
    for j = 1 : size(x, 2)
        if(strcmp(option, 'max'))
            [~, loc] = max([prob1(i, j), prob2(i, j), prob3(i, j)]);
        elseif(strcmp(option, 'min'))
            [~, loc] = min([prob1(i, j), prob2(i, j), prob3(i, j)]);
        end
        region(i, j) = loc - 1;
    end
end
end
```