北京航空航天大學
BEIHANG UNIVERSITY

Pattern Recognition and Machine Learning

Experiment Report

School of Automation Science and Electrical Engineering

Kun Xiao

14051166

May,2018

Experiment 2 Synthetical Design of Bayesian Classifier

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 $p(\omega_i|X)$ for each class ωi. 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 $P(\omega_i)$, $P(X|\omega_i)$, $\mathrm{i} = 1,2, \cdots, \mathrm{c},$ and X are known or given, the posterior probability can be derived as follows

$$P(\omega_i|X) = \frac{P(X|\omega_i)P(\omega_i)}{\sum_{j=1}^{c} P(X|\omega_j)P(\omega_j)} \qquad \text{i=1,…, c} \qquad (1)$$

Let the series of decision actions as $\{\mathrm{a}_1, \mathrm{a}_2, \cdots, \mathrm{a}_\mathrm{c}\}$, the conditional risk of decision action $\mathrm{a_i}$ can be computed by

$$R(a_i|X) = \sum_{j=1, j\neq i}^{c} \lambda(a_i, \omega_j)P(\omega_j|X), \qquad \text{i=1,…, c} \qquad (2)$$

Thus the minimum risk Bayesian decision can be found as

$$a^*_{\mathrm{k}} = \mathrm{Arg\ min_i}\ R(a_i|X), \qquad \text{i=1,…, c} \qquad (3)$$

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:

(1) According to the above principle and theory in section 2.2, design a Bayesian classifier for the classification of two classes of patterns which are subjected to Gaussian normal distribution and compile the corresponding programme codes.

(2) In view of the normal cell class ω_1 , the corresponding data of sample features are extracted as

Ω_1 ={ -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},

and the sample features of abnormal cell class ω_2 are listed as

Ω_2 = { 2.8792， 0.7932，1.1882，3.0682，4.2532，0.3271,0.9846,2.7648,2.6588}

The prior probabilities of both ω_1 and ω_2 are known as

$p(\omega_1) = 0.9, \qquad p(\omega_2) = 0.1$

The loss parameters for different decision action are given as table 1

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	0	

Suppose the conditional probability distributions are Gaussian, find the conditional probability density functions $p(X|\omega_1)$ and $p(X|\omega_2)$ and complete the design of Bayessian classifier with minimum risk, and then give a comparative analysis with the situation without considering decision loss.

Draw the curves of prior and posterior probability density functions, $p(X|\omega_1)$, $p(X|\omega_2)$, $p(\omega_1|X)$, and $p(X|\omega_2)$, give the classifying decision boundary function and illustration of classification result.

Stage 2 (towards an in depth study):

(1) Create a pattern dataset of multiple classes and high dimension with more than 50 samples for each class. Then design a Bayesian classifier and complete the corresponding experiments and comparative analysis by using self-supposed prior probabilities and loss parameters for the same terms as stage 1.

(2) Think and analyse the intrinsic relationship between the classifier of two classes and the one of multiple classes, give your comments.

Stage 3：

Explore the questions in the previous section and design experiments to answer

5. Initial Condition

For Stage 1, the initial condition is listed in Content and Procedure.
For Stage 2, the initial condition is listed as follows(risk is omitted) :

$$\mu_1 = [2,2], \mu_2 = [0,0], \mu_3 = [-2,-2]$$

$$\Sigma_1 = \begin{bmatrix} 1,0 \\ 0,9 \end{bmatrix}, \Sigma_2 = \begin{bmatrix} 4,0 \\ 0,16 \end{bmatrix}, \Sigma_3 = \begin{bmatrix} 4,0 \\ 0,9 \end{bmatrix}$$

$$p(\omega_1) = 0.7, p(\omega_2) = 0.25, p(\omega_3) = 0.05$$

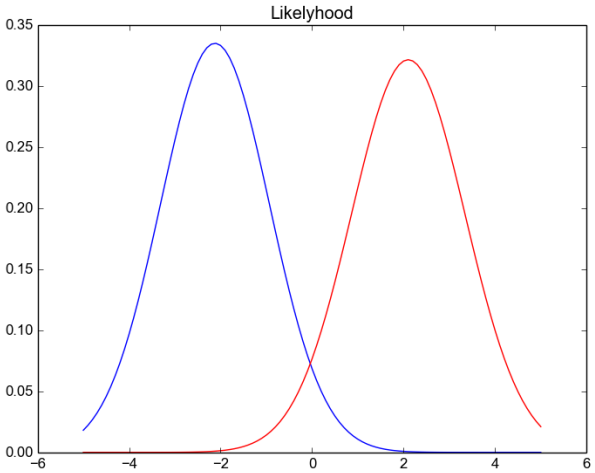
6. Source Code

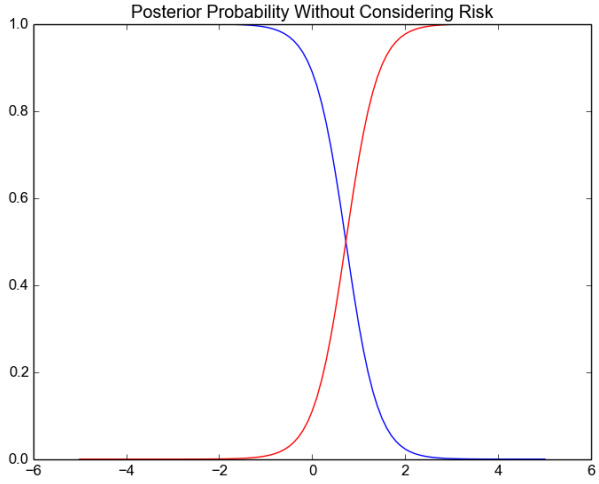
See https://github.com/robin-shaun/BUAA-Pattern-Recognition-and-Machine-Learning/Experient2 or Appendix.

7. Results and Analysis

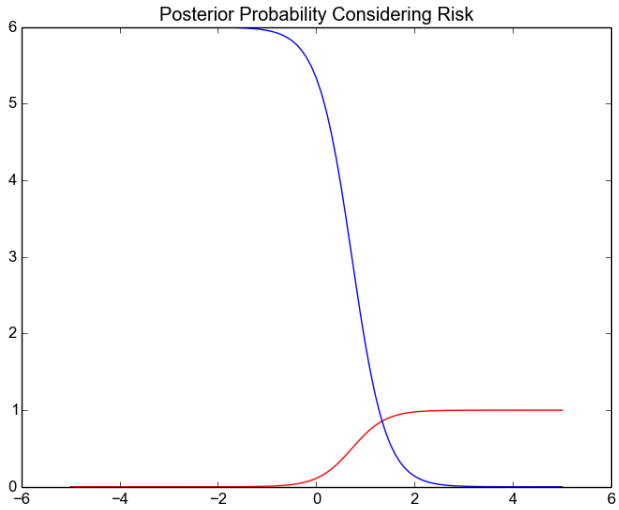
The results are listed as follows.

Stage 1:



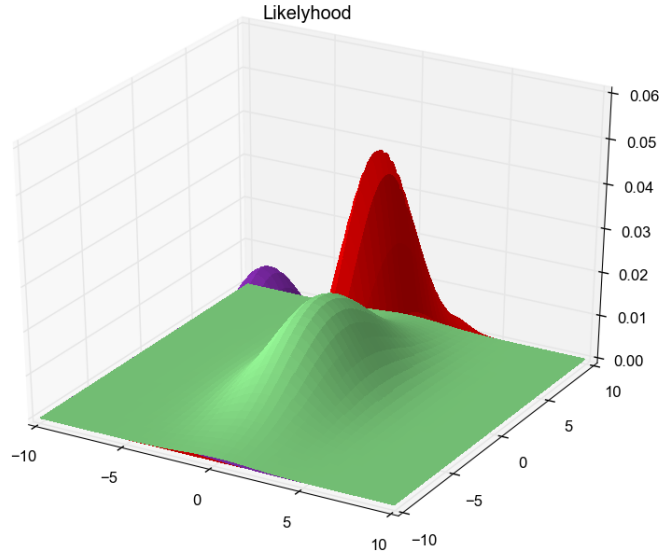


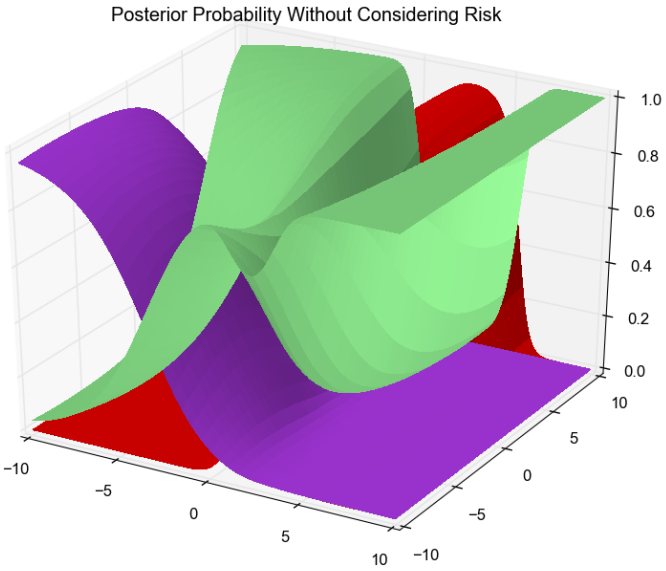
The decision boundary without considering the risk is 0.8

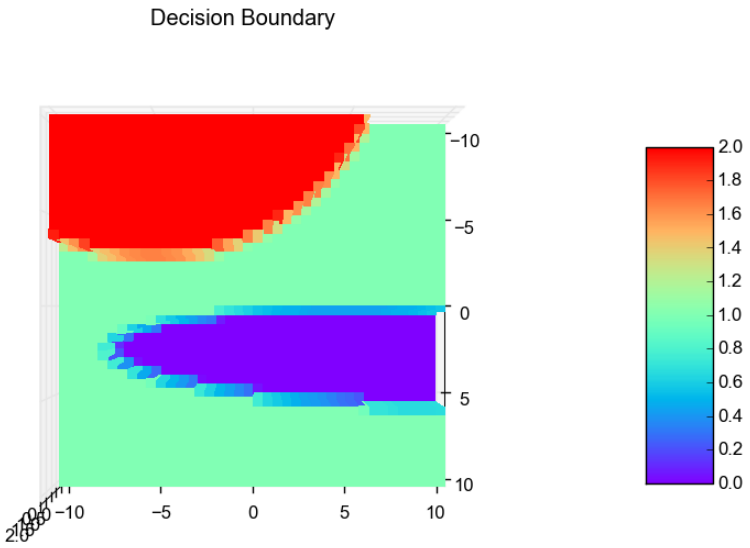


The decision boundary considering the risk is 1.4

Stage 2:







Analysis: 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.

Appendix

Source Code

```
import numpy
import matplotlib
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from scipy.stats import multivariate_normal
from matplotlib import cm

class Bayesian_Classifier_MultiDim:
    def __init__(self,domainx,domainy,prior_probability1,prior_probability2,prior_probability3,risk):
        self.domainX,self.domainY=numpy.meshgrid(domainx,domainy)
        self.prior_probability1=prior_probability1
        self.prior_probability2=prior_probability2
        self.prior_probability3=prior_probability3
        self.risk=risk

    def calConProbability(self,sample1,sample2,sample3):
        self.mean1=numpy.mean(sample1,axis=1)
        self.cov1=numpy.cov(sample1)
        self.mean2=numpy.mean(sample2,axis=1)
        self.cov2=numpy.cov(sample2)
        self.mean3=numpy.mean(sample3,axis=1)
        self.cov3=numpy.cov(sample3)

    def calLikelyhood(self):
        #Calculate Likelyhood
        pos =numpy.empty(self.domainX.shape+(2,))
        pos[:,:,0]=self.domainX;
        pos[:,:,1]=self.domainY;
        self.likelyhood1=multivariate_normal.pdf(pos,mean=self.mean1,cov=self.cov1)
        self.likelyhood2=multivariate_normal.pdf(pos,mean=self.mean2,cov=self.cov2)
        self.likelyhood3=multivariate_normal.pdf(pos,mean=self.mean3,cov=self.cov3)

        fig1=plt.figure(1)
        ax1=Axes3D(fig1)
        plt.title('Likelyhood')
        ax1.plot_surface(self.domainX, self.domainY, self.likelyhood1, color='r',linewidth=0, antialiased=False)
        ax1.plot_surface(self.domainX, self.domainY,
self.likelyhood2,color=(151/255,251/255,152/255),linewidth=0, antialiased=False)
        ax1.plot_surface(self.domainX, self.domainY, self.likelyhood3,
color=(153/255,50/255,204/255),linewidth=0, antialiased=False)
```

```
    def calPostProbability(self):
        #Calculate Posterior Probability
        self.posterior_probability1=self.prior_probability1*self.likelyhood1
        self.posterior_probability2=self.prior_probability2*self.likelyhood2
        self.posterior_probability3=self.prior_probability3*self.likelyhood3

posterior_probability_sum=self.posterior_probability1+self.posterior_probability2+self.posterior_probability3
        self.posterior_probability1=self.posterior_probability1/posterior_probability_sum
        self.posterior_probability2=self.posterior_probability2/posterior_probability_sum
        self.posterior_probability3=self.posterior_probability3/posterior_probability_sum

        fig2=plt.figure(2)
        ax2=Axes3D(fig2)
        plt.title('Posterior Probability Without Considering Risk')
        ax2.plot_surface(self.domainX, self.domainY, self.posterior_probability1, color='r',linewidth=0,
antialiased=False)
        ax2.plot_surface(self.domainX, self.domainY, self.posterior_probability2,
color=(151/255,251/255,152/255),linewidth=0, antialiased=False)
        ax2.plot_surface(self.domainX, self.domainY, self.posterior_probability3,
color=(153/255,50/255,204/255),linewidth=0, antialiased=False)

    def calDecisionBoundary(self):
        ##Calculate Decision Boundary
        region=numpy.zeros([numpy.size(self.domainX,axis=0),numpy.size(self.domainX,axis=1)])
        for i in range(numpy.size(self.domainX,axis=0)):
            for j in range(numpy.size(self.domainX,axis=1)):

region[i,j]=numpy.argmax([self.posterior_probability1[i,j],self.posterior_probability2[i,j],self.posterior_probability
3[i,j]])
        fig3=plt.figure(3)
        ax3=Axes3D(fig3)
        plt.title('Decision Boundary')
        ax3.view_init(elev=90,azim=0)
        surf3=ax3.plot_surface(self.domainX, self.domainY, region, cmap='rainbow',linewidth=0,
antialiased=False)
        fig3.colorbar(surf3, shrink=0.5, aspect=5)


if __name__=='__main__':
    #build samples
    mean1=numpy.array([2,2])
    mean2=numpy.array([0,0])
    mean3=numpy.array([-2,-2])
```

```
    cov1=numpy.array([[1,0],[0,9]])
    cov2=numpy.array([[4,0],[0,16]])
    cov3=numpy.array([[4,0],[0,9]])
    sample1=numpy.random.multivariate_normal(mean1, cov1, 100).T
    sample2=numpy.random.multivariate_normal(mean2, cov2, 100).T
    sample3=numpy.random.multivariate_normal(mean3, cov3, 100).T

    domainx=numpy.arange(-10.0,10.0,0.05)
    domainy=numpy.arange(-10.0,10.0,0.05)

    prior_probability1=0.7
    prior_probability2=0.25
    prior_probability3=0.05

    risk=numpy.array([[0,3,7],[8,0,8],[0,2,6]])


bayesian_Classifier_MultiDim=Bayesian_Classifier_MultiDim(domainx,domainy,prior_probability1,prior_probab
ility2,prior_probability3,0.1)
    bayesian_Classifier_MultiDim.calConProbability(sample1,sample2,sample3)
    bayesian_Classifier_MultiDim.calLikelyhood()
    bayesian_Classifier_MultiDim.calPostProbability()
    bayesian_Classifier_MultiDim.calDecisionBoundary()
    plt.show()
```