Problem 1

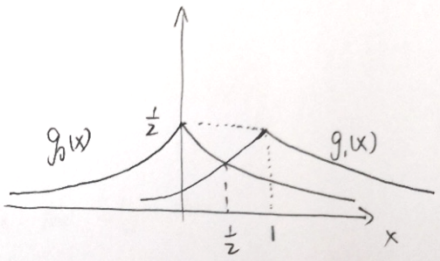
Problem 1:

[Q1] Since $P(Y=0) = P(Y=1) = \frac{1}{2}$

$$h^B(x) = \arg\max_k g_k(x) = \arg\max_k P(x|Y)P(Y)$$
$$= \arg\max_k P(x|Y)$$

Thus, $h^B(x) = \begin{cases} 0 & g_0(x) \geq g_1(x) \\ 1 & g_0(x) < g_1(x) \end{cases}$

Sketch



$$h^B(x) = \begin{cases} 0 & x \leq \frac{1}{2} \\ 1 & x > \frac{1}{2} \end{cases}$$

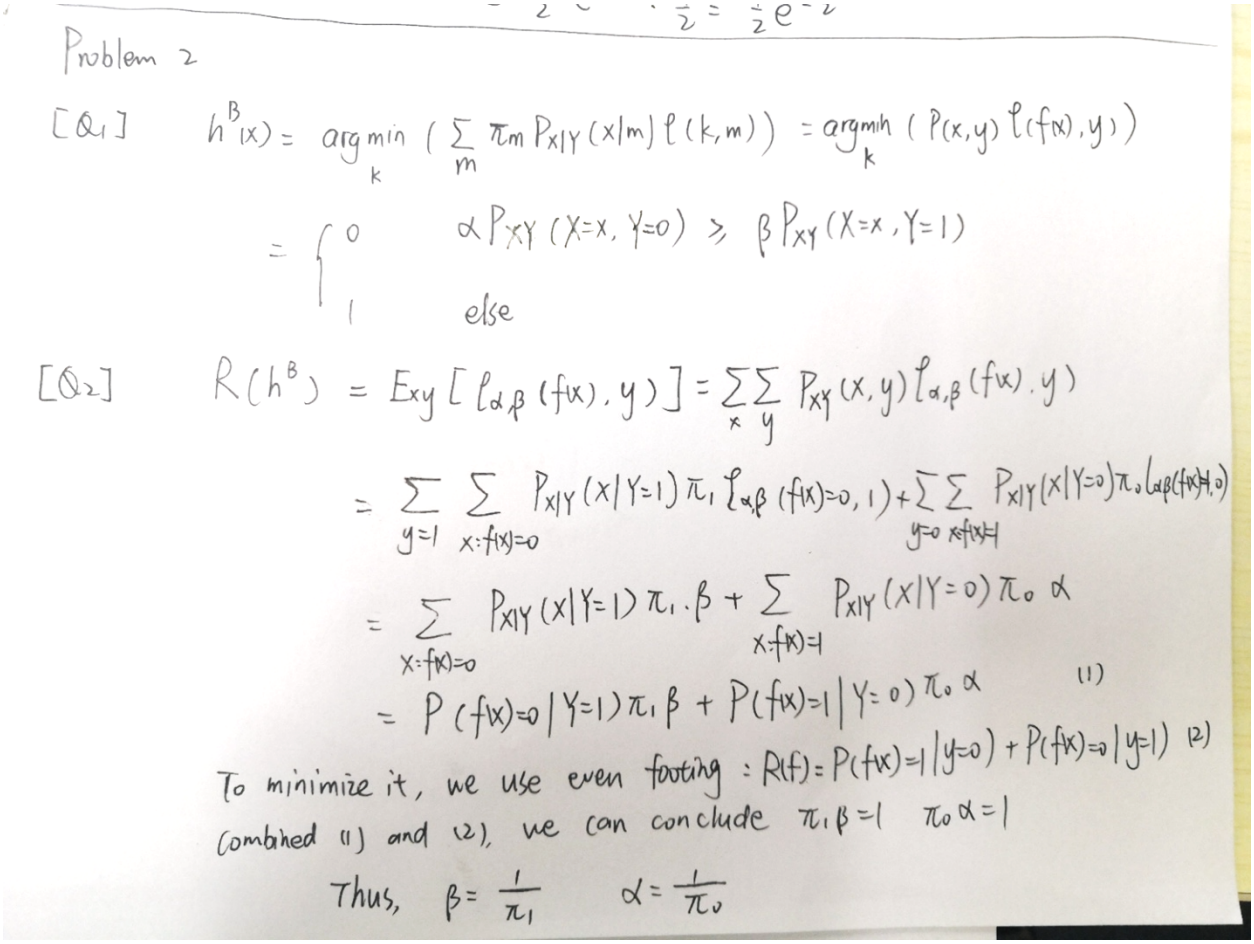
[Q2] $R(h^B) = R(h^B \neq y) = P(h^B(x) \neq 1 | Y=1)P(Y=1) + P(h^B(x) \neq 0 | Y=0)P(Y=0)$

$$= \frac{1}{2}\int_{-\infty}^{\frac{1}{2}} g_1(x)\,dx + \frac{1}{2}\int_{\frac{1}{2}}^{+\infty} g_0(x)\,dx$$

Since $\int_{-\infty}^{\frac{1}{2}} g_1(x) = \int_{\frac{1}{2}}^{+\infty} g_0(x) = \int_{\frac{1}{2}}^{+\infty} \frac{1}{2}e^{-x}\,dx = \frac{1}{2}e^{-x}\Big|_{\frac{1}{2}}^{+\infty} = \frac{1}{2}e^{-\frac{1}{2}}$

Thus, $R(h^B) = 2 \cdot \frac{1}{2}e^{-\frac{1}{2}} \cdot \frac{1}{2} = \frac{1}{2}e^{-\frac{1}{2}}$

Problem 2



Problem 2

[Q1] $h^B(x) = \arg\min_k \left(\sum_m \pi_m P_{X|Y}(x|m)\, \ell(k,m) \right) = \arg\min_k \left(P(x,y)\, \ell(f(x),y) \right)$

$$= \begin{cases} 0 & \alpha P_{XY}(X=x, Y=0) \geq \beta P_{XY}(X=x, Y=1) \\ 1 & \text{else} \end{cases}$$

[Q2] $R(h^B) = E_{XY}[\ell_{\alpha\beta}(f(x), y)] = \sum_x \sum_y P_{XY}(x,y)\, \ell_{\alpha,\beta}(f(x), y)$

$$= \sum_{y=1} \sum_{x:f(x)=0} P_{X|Y}(x|Y=1)\pi_1 \ell_{\alpha,\beta}(f(x)=0, 1) + \sum_{y=0} \sum_{x:f(x)=1} P_{X|Y}(x|Y=0)\pi_0 \ell_{\alpha\beta}(f(x)=1, 0)$$

$$= \sum_{x:f(x)=0} P_{X|Y}(x|Y=1)\pi_1 \cdot \beta + \sum_{x:f(x)=1} P_{X|Y}(x|Y=0)\pi_0 \alpha$$

$$= P(f(x)=0 \mid Y=1)\pi_1 \beta + P(f(x)=1 \mid Y=0)\pi_0 \alpha \quad (1)$$

To minimize it, we use even footing : $R(f) = P(f(x)=1 \mid y=0) + P(f(x)=0 \mid y=1)$ (2)

Combined (1) and (2), we can conclude $\pi_1 \beta = 1 \quad \pi_0 \alpha = 1$

Thus, $\beta = \frac{1}{\pi_1} \quad \alpha = \frac{1}{\pi_0}$

Xueren Ge 903608409

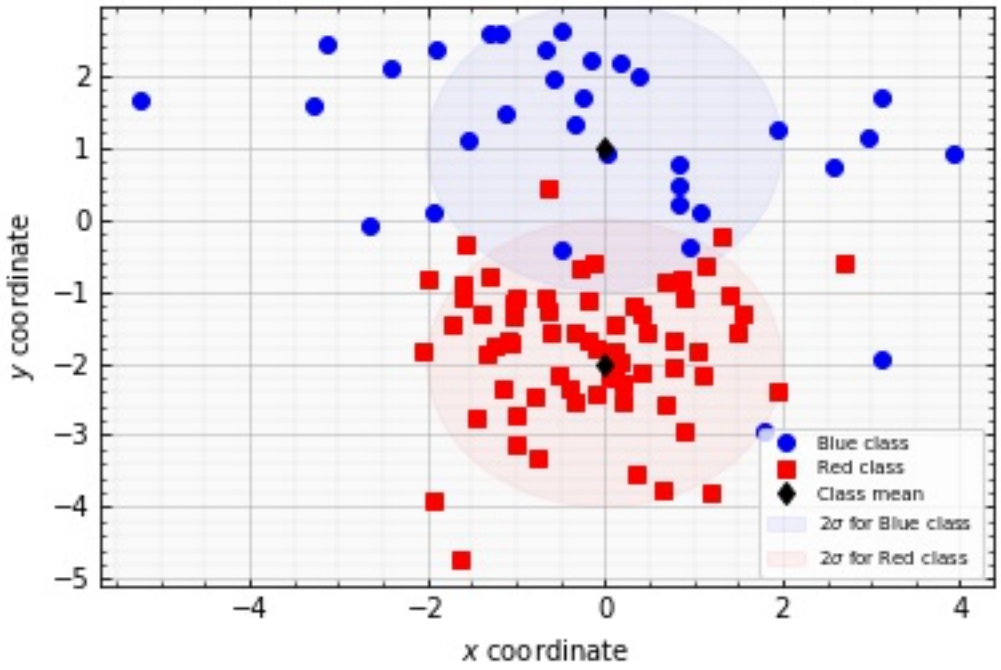
Problem 3

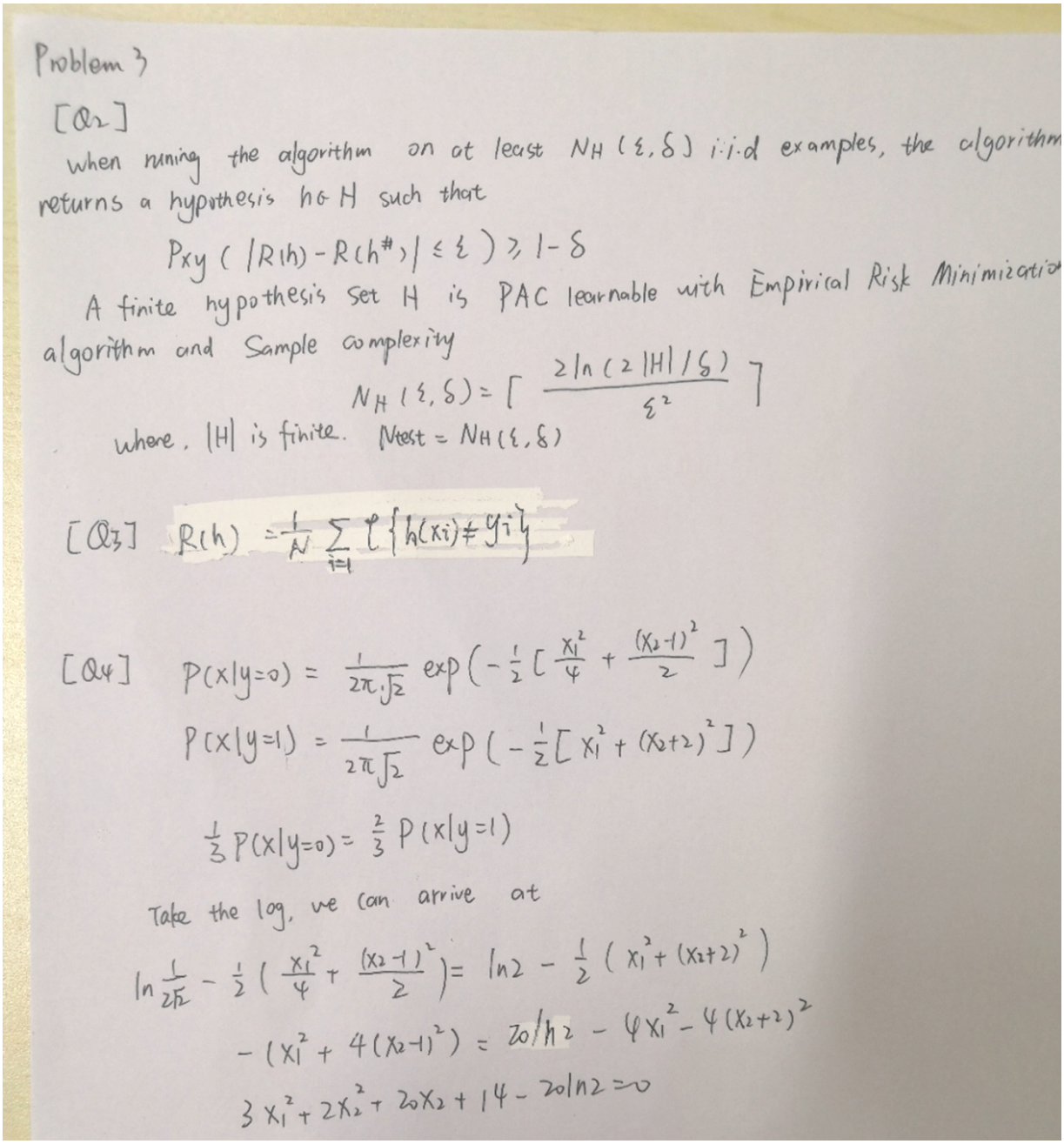
Q1

The code below is what I generate the training dataset

```
BlueDataSet = np.random.multivariate_normal([0,1],[[4,0],[0,2]],33) #label 0
RedDataSet = np.random.multivariate_normal([0,-2],[[1,0],[0,1]],67) #label 1

TrainingData = np.r_[RedDataSet,BlueDataSet]
TrainingLabel = np.r_[np.ones((67,1)),0*np.ones((33,1))]
```

The picture below shows the generating training dataset.





Problem 3

[Q2]

When running the algorithm on at least $N_H(\varepsilon, \delta)$ i.i.d examples, the algorithm returns a hypothesis $h \in H$ such that

$$P_{xy}(|R(h) - R(h^*)| \le \varepsilon) \ge 1 - \delta$$

A finite hypothesis set H is PAC learnable with Empirical Risk Minimization algorithm and Sample complexity

$$N_H(\varepsilon, \delta) = \left\lceil \frac{2\ln(2|H|/\delta)}{\varepsilon^2} \right\rceil$$

where, $|H|$ is finite. $N_{test} = N_H(\varepsilon, \delta)$

[Q3] $R(h) = \frac{1}{N}\sum_{i=1}^{N} \mathbb{1}\{h(x_i) \neq y_i\}$

[Q4] $P(x|y=0) = \frac{1}{2\pi\sqrt{2}} \exp\left(-\frac{1}{2}\left[\frac{x_1^2}{4} + \frac{(x_2-1)^2}{2}\right]\right)$

$P(x|y=1) = \frac{1}{2\pi\sqrt{2}} \exp\left(-\frac{1}{2}\left[x_1^2 + (x_2+2)^2\right]\right)$

$\frac{1}{3}P(x|y=0) = \frac{2}{3}P(x|y=1)$

Take the log, we can arrive at

$$\ln\frac{1}{2\pi} - \frac{1}{2}\left(\frac{x_1^2}{4} + \frac{(x_2-1)^2}{2}\right) = \ln 2 - \frac{1}{2}\left(x_1^2 + (x_2+2)^2\right)$$

$$-(x_1^2 + 4(x_2-1)^2) = 20\ln 2 - 4x_1^2 - 4(x_2+2)^2$$

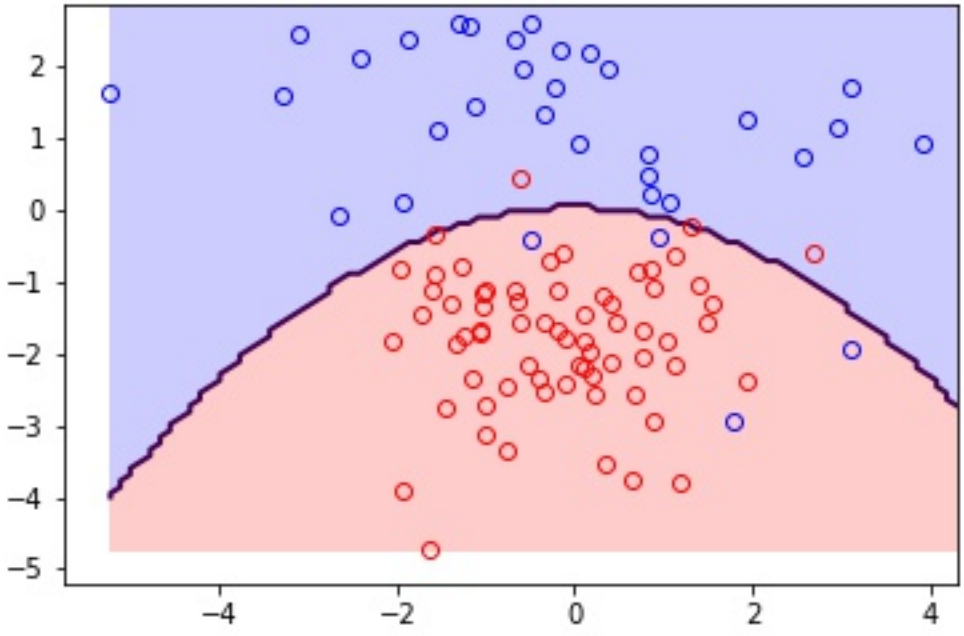
$$3x_1^2 + 2x_2^2 + 20x_2 + 14 - 20\ln 2 = 0$$

As for Q2, I set the testing data as 10,000. 5000 labeled as 1 and 5000 labeled as 0.

Xueren Ge 903608409

Q5

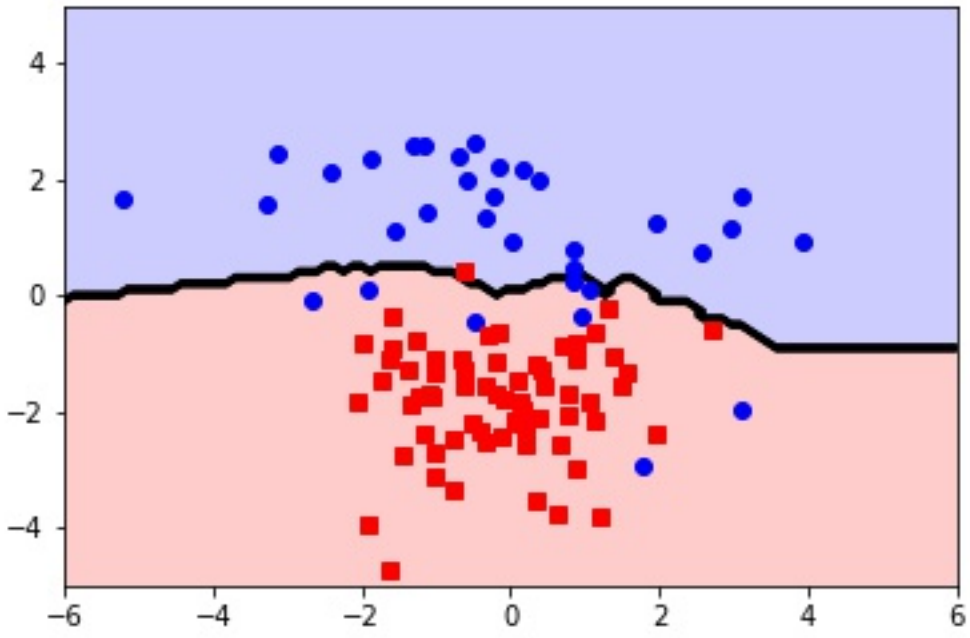
As shown in picture below, I run the Bayes classifier. The black line is the decision boundary, the blue dataset is labeled by 0 and the red dataset is labeled by 1. And the testing risk of Bayes classifier is 0.0973.



Xueren Ge 903608409

Q6

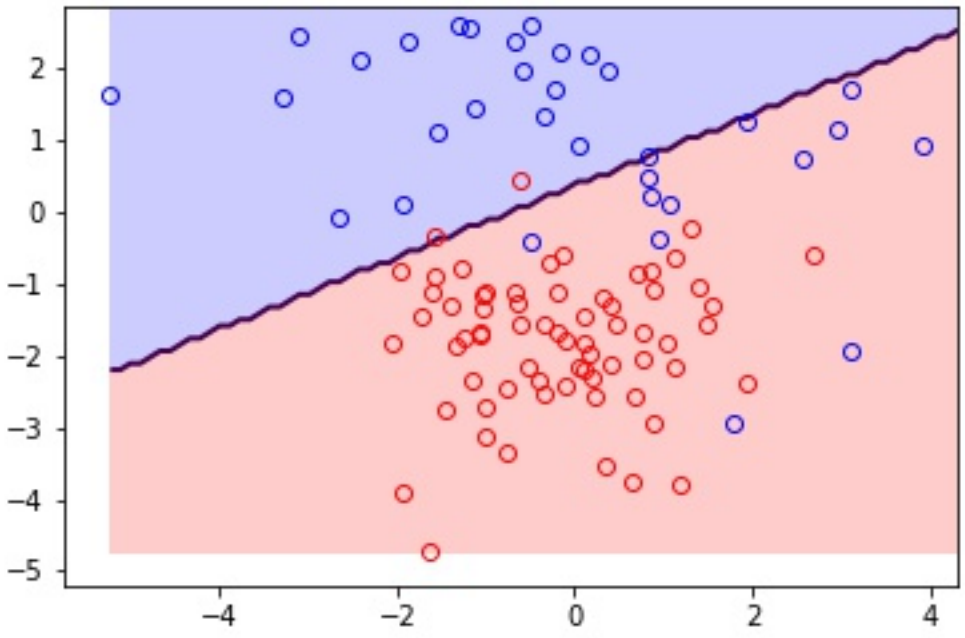
The picture below shows the KNN classifier, the black line is the decision boundary, the blue dataset is labeled by 0 and the red dataset is labeled by 1. And the testing risk of Bayes classifier is 0.1479. I set parameter k as 15. When setting as 10, I found there are still islands so in order not to overfitting, I set it as 15. So the decision boundary is smoother.



Xueren Ge 903608409

Q7

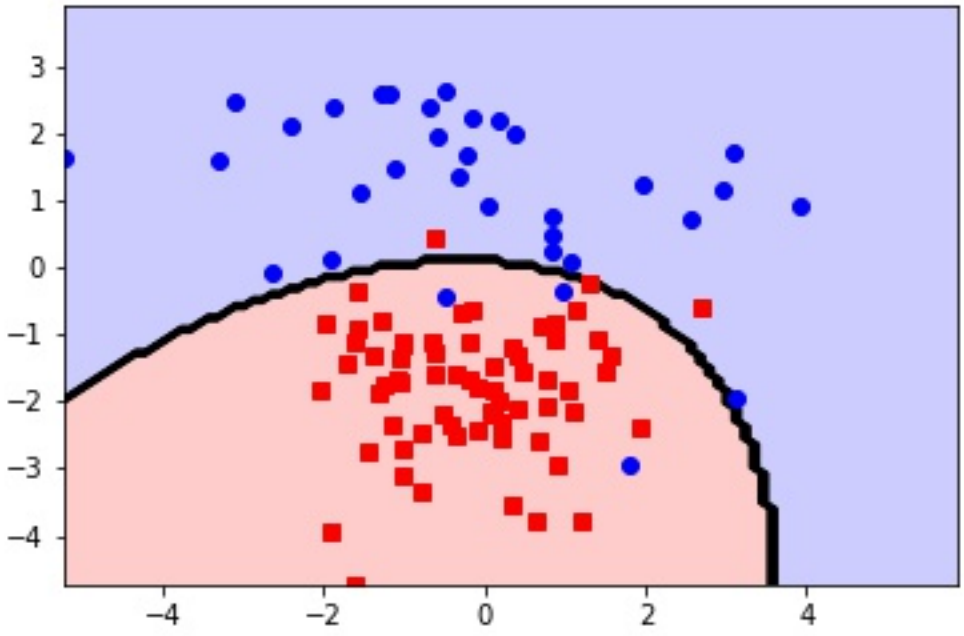
As shown in picture below, I run the LDA classifier. The black line is the decision boundary, the blue dataset is labeled by 0 and the red dataset is labeled by 1. And the testing risk of Bayes classifier is 0.1843. LDA is a linear classifier when k=2. Thus, the decision boundary is linear.



Xueren Ge 903608409

Q8

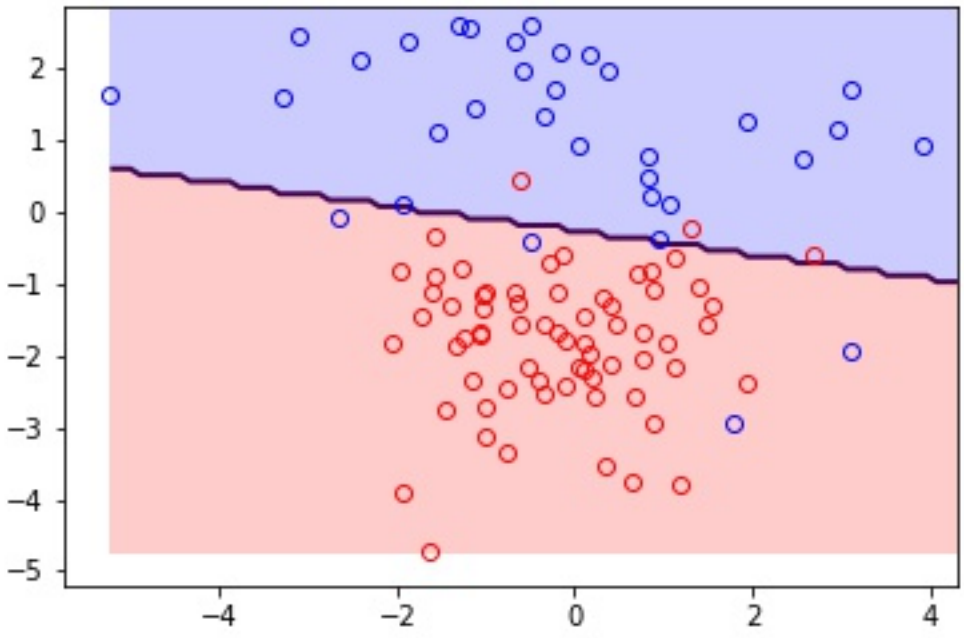
As shown in picture below, I run the QDA classifier. The black line is the decision boundary, the blue dataset is labeled by 0 and the red dataset is labeled by 1. And the testing risk of Bayes classifier is 0.1078. Since the covariance matrices of the generative distributions are now allowed to depend on the class label, it's a un-linear decision boundary.



Xueren Ge 903608409

Q9

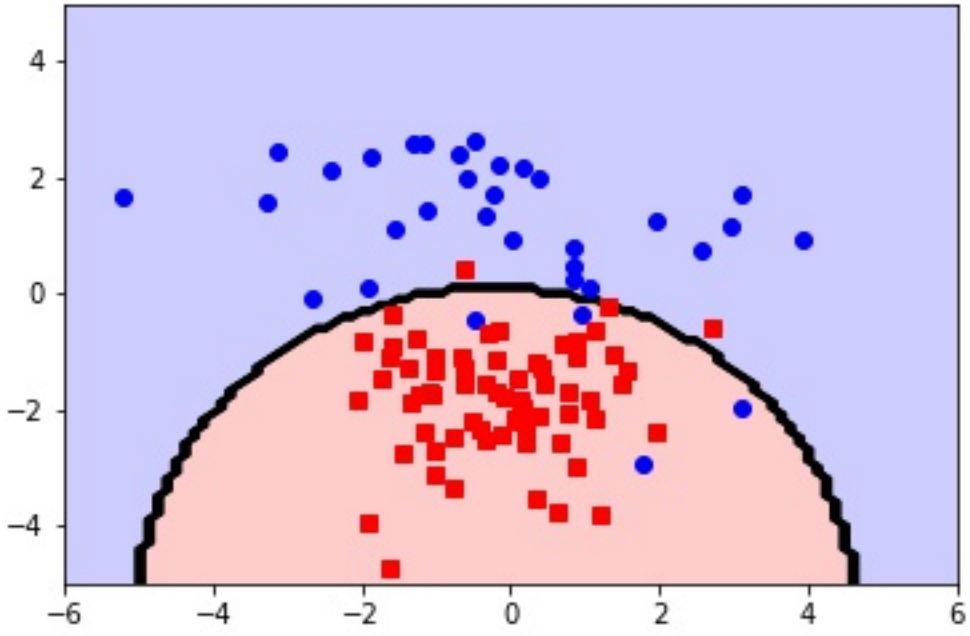
As shown in picture below, I run the Logistic Regression classifier. The black line is the decision boundary, the blue dataset is labeled by 0 and the red dataset is labeled by 1. And the testing risk of Bayes classifier is 0.1182. Logistic Regression classifier is a linear classifier.



Xueren Ge 903608409

Q10

As shown in picture below, I run the Gaussian Naïve Bayes classifier. The black line is the decision boundary, the blue dataset is labeled by 0 and the red dataset is labeled by 1. And the testing risk of Bayes classifier is 0.1052.



Problem 4

Problem 4:

[Q1]: No.

if $X \sim U(0, 0.01)$

$P(x) = \begin{cases} 100, & x \in [0, 0.01] \\ 0 & \text{else} \end{cases}$

Apparently, $P(x) > 1$ when $x \in [0, 0.01]$

[Q2]: Yes

1-NN only remembers itself (point) because the nearest neighbor is the point itself, actually the classifier is overfitting. Thus, the training error must be 0.

[Q3]: No

Naive Bayes require the assumption: Given y, the feature $\{x_i\}_{i=1}^{d}$ of x are independent. $P_{x|y} = \prod_{i=1}^{d} P_{x_i|y}$.

As for bivariate Gussian case, if the covariance matrix Σ is dependent on the label k, the Naive Bayes won't work because even though we know the distribution of data, it violates Naive Bayes's assumption.

However, LDA can handle it. LDA doesn't require the strict constraints. μ_k can be dependent, but covariance matrix is not. In this case, LDA works better

[Q4] No.

We can only numerical calculation to compute the MLE of logistic Regression. Thus, numerical calculation method like GD. SGD. Netwon's Method, they will converge after interations, it only yields one local optima.

[Q5] No.

An hypothesis class is PAC learnable if there exists an algorithm that returns an hypothesis whose true risk is close to the minimum true risk (that's the approximately correct part) with high probability (that's the probably part). A crucial feature is that this has to hold regardless of the exact of the underlying distribution P_{xy}

[Q6] Yes

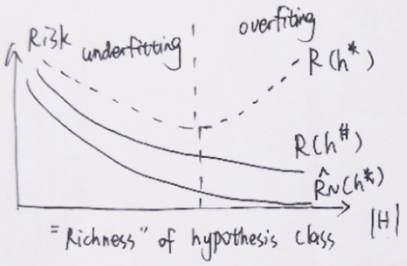
kernel method actually computes the inner product . But kernel avoids computation directly in Hilbert Space, by mapping x to $\Phi(x)$. kernel method gains the same result as computing inner product in Hilbert Space by mapping x to $\Phi($

[Q7] Yes

$$p(x|z)p(y|z) = p(x|y,z)p(y|z) = \frac{P(x,y,z)}{P(y,z)} \cdot \frac{P(y,z)}{P(z)}$$

$$= \frac{P(x,y,z)}{P(z)} = P(x,y|z)$$

[Q8] Yes



As is shown in picture, when Richness of hypothesis grows, it's more likely to reach into zone "overfitting".

[Q9] No

I think it depends on the covariance matrix Σ. The decision boundary could be linear or quadratic. If $P_{X|Y}(X|0)$ and $P_{X|Y}(X|1)$ has different Σ_0, Σ_1, and assume $P_Y(y=0)$, $P_Y(y=1)$ are the same equal to $\frac{1}{2}$

when we plug in in Bayes classifier $h^B(x) = \arg\min_k \eta_k(x) = \arg\min_k P(X|Y)P(Y)$ we will find if $\Sigma_0 = \Sigma_1$, it's linear decision boundary, when $\Sigma_0 \neq \Sigma_1$, it's quadratic decision boundary.

[Q10] Yes

$$\eta_1(x) = \frac{1}{1+\exp(-(\hat{w}^T x + \hat{b}))}$$

Since the function $x \mapsto \frac{1}{1+e^{-x}}$ is called the logistic map, the corresponding classifier inherited the name and is defined as

$$h^{LR}(x) = \mathbb{1}\{\eta_1(x) \geq \tfrac{1}{2}\} = \mathbb{1}\{\hat{w}^T x + \hat{b} \geq 0\}$$

Thus it's linear classifier

Xueren Ge 903608409

Appendix Code

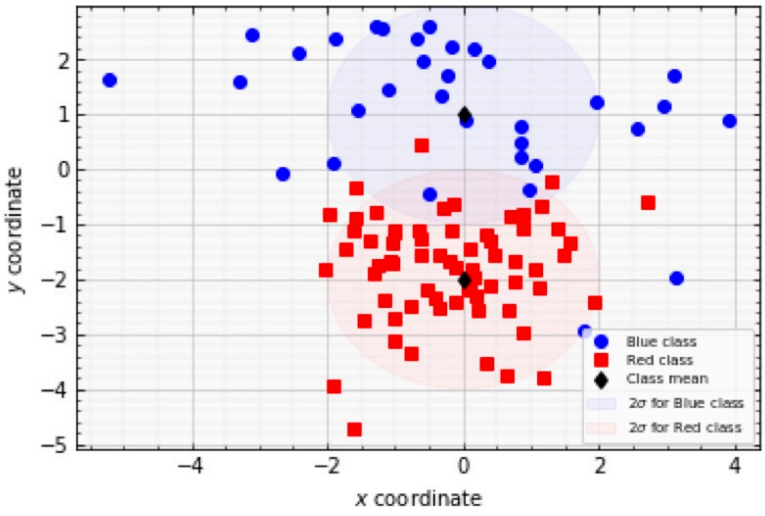
Mid

October 10, 2020

[30]:
```
import numpy as np
import matplotlib
from matplotlib import pyplot as plt
from matplotlib.colors import LinearSegmentedColormap
from sklearn import *
import math
from sklearn import neighbors
from sklearn.naive_bayes import GaussianNB
```

[31]:
```
BlueDataSet = np.random.multivariate_normal([0,1],[[4,0],[0,2]],33) #label 0
RedDataSet = np.random.multivariate_normal([0,-2],[[1,0],[0,1]],67) #label 1

TrainingData = np.r_[RedDataSet,BlueDataSet]
TrainingLabel = np.r_[np.ones((67,1)),0*np.ones((33,1))]

TestData_b = np.random.multivariate_normal([0,1],[[4,0],[0,2]],5000)
TestData_r = np.random.multivariate_normal([0,-2],[[1,0],[0,1]],5000)
TestData = np.r_[TestData_r,TestData_b]
TestLabel = np.r_[np.ones((5000,1)),0*np.ones((5000,1))]

fid = plt.figure()

Axes = plt.subplot(1,1,1)
Axes.axes.tick_params(which='both',direction='in',top=True, right=True)
plt.minorticks_on()
Axes.set_facecolor((0,0,0,0.02))
plt.plot(BlueDataSet[:,0],BlueDataSet[:,1],'o',color='b',label='Blue class')
plt.plot(RedDataSet[:,0],RedDataSet[:,1],'s',color='r',label='Red class')
plt.plot(0,1,'kd',label='Class mean')
plt.plot(0,-2,'kd')
BlueClassSdev = matplotlib.patches.Circle((0,1),radius=2,color='blue',alpha=0.
 ↪05,label='2$\sigma$ for Blue class')
Axes.add_patch(BlueClassSdev)
RedClassSdev = matplotlib.patches.Circle((0,-2),radius=2,color='red',alpha=0.
 ↪05,label='2$\sigma$ for Red class')
Axes.add_patch(RedClassSdev)
plt.grid(True,which='major',linewidth=0.5)
```

```
plt.grid(True,which='minor',linewidth=0.1)
# plt.xlim([-5,5])
# plt.ylim([-4,4])
plt.xlabel("$x$ coordinate")
plt.ylabel("$y$ coordinate")
plt.legend(loc='lower right',fontsize='x-small')
plt.savefig('Q1.jpg')
```



[32]:
```
my_colors = [(0,0,1),(1,0,0)]#RGB
my_cm = LinearSegmentedColormap.from_list('my_cm', my_colors, N=2)
```

1 Bayes

[33]:
```
#Need to create a mesh to evaluate classifier
x_grid = np.linspace(1.0*np.min(TrainingData[:,0]), 1.5*np.max(TrainingData[:
 ↪,0]), 100)
y_grid = np.linspace(1.0*np.min(TrainingData[:,1]), 1.5*np.max(TrainingData[:
 ↪,1]), 100)
x_mesh, y_mesh = np.meshgrid(x_grid, y_grid)
```

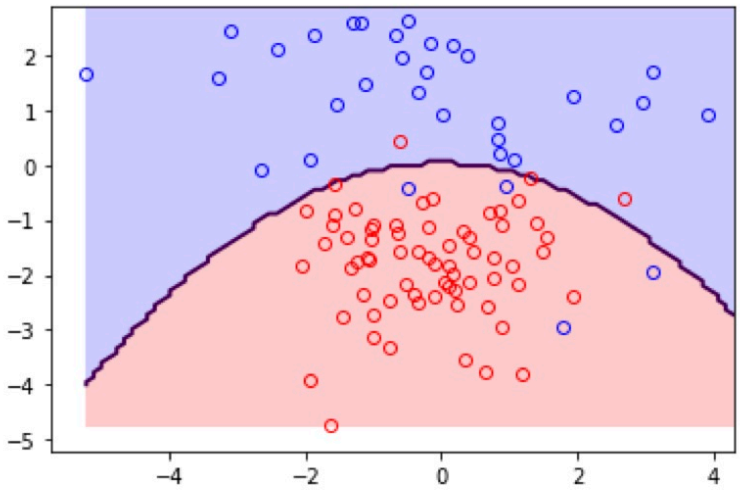
[34]:
```
def BayesPredict(xcoord, ycoord):
    pred = np.zeros(xcoord.shape)
    for i in np.arange(0, len(xcoord)):
```

```
        lin_class = 3*xcoord[i]**2+20*ycoord[i]+14-20*np.log(2)
        if lin_class >= 0:
            pred[i] = 0
        else:
            pred[i] = 1
    return (pred)

Test_prediction = BayesPredict(TestData[:,0], TestData[:,1])
count=0
for i in range(len(Test_prediction)):
    if Test_prediction[i] != TestLabel[i]:
        count+=1
print("test risk=", count/10000)

prediction = BayesPredict(x_mesh.ravel(), y_mesh.ravel())
prediction = prediction.reshape(x_mesh.shape)
```

```
test risk= 0.0973
```

[35]:
```
plt.plot(BlueDataSet[:,0], BlueDataSet[:,1], 'o', markerfacecolor="None",␣
 ↪markeredgecolor='b', label='Blue')
plt.plot(RedDataSet[:,0], RedDataSet[:,1], 'o', markerfacecolor="None",␣
 ↪markeredgecolor='r', label='Red')
plt.xlim([1.1*np.min(TrainingData[:,0]), 1.1*np.max(TrainingData[:,0])])
plt.ylim([1.1*np.min(TrainingData[:,1]), 1.1*np.max(TrainingData[:,1])])
Bayes = plt.contourf(x_mesh, y_mesh, prediction, levels = [-1,0,1], cmap=my_cm,␣
 ↪alpha=0.2)
Bayes = plt.contour(x_mesh, y_mesh, prediction, [-1,0,1,2], linewidths=2,␣
 ↪color='k')
plt.savefig('Q2.jpg')
```

```
/Users/gexueren/opt/anaconda3/lib/python3.7/site-
packages/ipykernel_launcher.py:6: UserWarning: No contour levels were found
within the data range.

/Users/gexueren/opt/anaconda3/lib/python3.7/site-
packages/ipykernel_launcher.py:6: UserWarning: The following kwargs were not
used by contour: 'color'
```



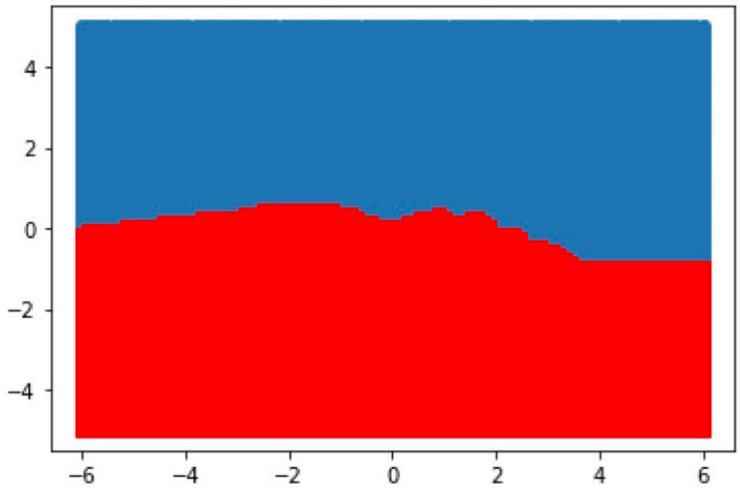
2 KNN

[36]:
```
NearestNeighbors = 15
KNNClassifier = neighbors.KNeighborsClassifier(n_neighbors=NearestNeighbors)
# TrainingData = np.r_[RedDataSet,BlueDataSet]
# TrainingLabel = np.r_[np.ones((67,1)),0*np.ones((33,1))]
KNNClassifier.fit(TrainingData,TrainingLabel.ravel())
Test_prediction = KNNClassifier.predict(TestData)
count=0
for i in range(len(Test_prediction)):
    if Test_prediction[i] != TestLabel[i]:
        count+=1
print("test risk=", count/10000)

x_grid = np.linspace(-6,6,100)
y_grid = np.linspace(-5,5,100)
x_mesh, y_mesh = np.meshgrid(x_grid,y_grid)
PredictedLabel = KNNClassifier.predict(np.c_[x_mesh.ravel(),y_mesh.ravel()])
Xcoord = x_mesh.ravel()
Ycoord = y_mesh.ravel()
BlueRegionX = Xcoord[PredictedLabel==0]
BlueRegionY = Ycoord[PredictedLabel==0]
RedRegionX = Xcoord[PredictedLabel==1]
RedRegionY = Ycoord[PredictedLabel==1]
plt.plot(BlueRegionX,BlueRegionY.ravel(),'o')
```

```
plt.plot(RedRegionX,RedRegionY.ravel(),'rs')
```

```
test risk= 0.1479
```

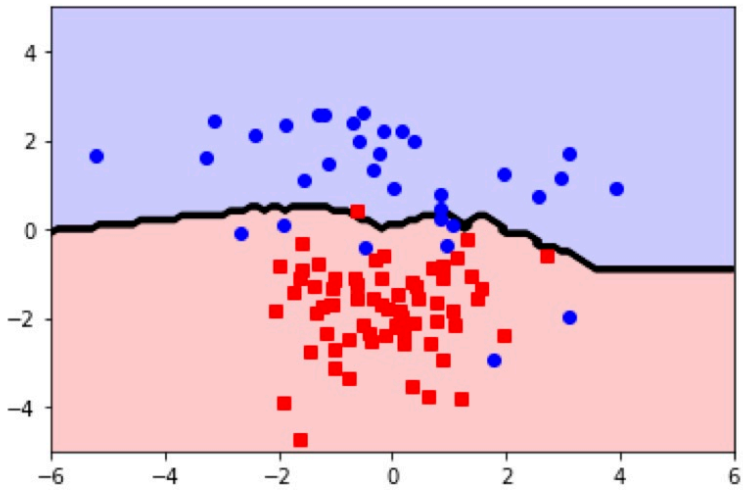
[36]:
```
[<matplotlib.lines.Line2D at 0x1a1ce27e90>]
```



[37]:
```
my_colors = [(0,0,1),(1,0,0)]#RGB
my_cm = LinearSegmentedColormap.from_list('my_cm', my_colors, N=2)
```

[38]:
```
plt.contourf(x_mesh,y_mesh,PredictedLabel.reshape(100,100),levels=[-1,0,1],␣
 ↪cmap = my_cm, alpha=0.2)
# plt.contourf(x_mesh,y_mesh,PredictedLabel.reshape(100,100),cmap = my_cm,␣
 ↪alpha=0.2)
plt.plot(BlueDataSet[:,0],BlueDataSet[:,1],'o',color='b',label='Blue class')
plt.plot(RedDataSet[:,0],RedDataSet[:,1],'s',color='r',label='Red class')
plt.contour(x_mesh,y_mesh,PredictedLabel.
 ↪reshape(100,100),linewidth=2,colors='k')
plt.savefig('Q3.jpg')
```

```
/Users/gexueren/opt/anaconda3/lib/python3.7/site-
packages/ipykernel_launcher.py:5: UserWarning: The following kwargs were not
used by contour: 'linewidth'
  """
```



3 LDA classifier

[39]:
```
#Need to create a mesh to evaluate classifier
x_grid = np.linspace(1.0*np.min(TrainingData[:,0]), 1.5*np.max(TrainingData[:
 ↪,0]), 100)
y_grid = np.linspace(1.0*np.min(TrainingData[:,1]), 1.5*np.max(TrainingData[:
 ↪,1]), 100)
x_mesh, y_mesh = np.meshgrid(x_grid, y_grid)
```

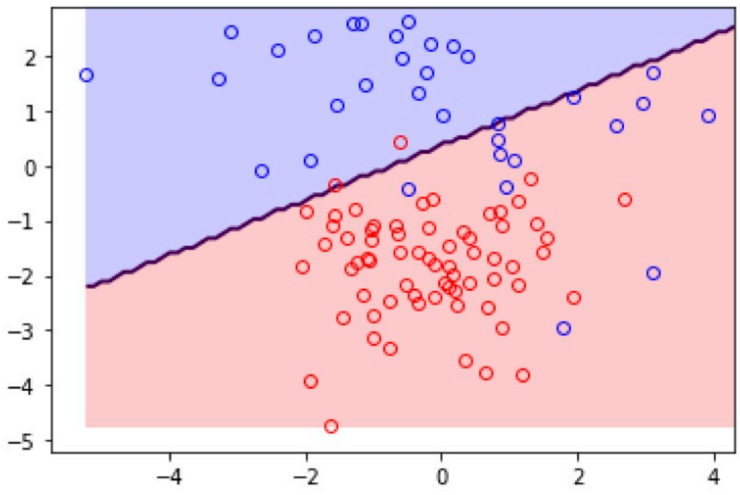
[40]:
```
mu_b = np.transpose([[-2,1]])
mu_r = np.transpose([[2,-1]])

pi_b = 1/3
pi_r = 2/3

sigma_1 =[[0,0],[0,0]]
for i in range(len(BlueDataSet)):
    sigma_1 += (np.transpose([[BlueDataSet[i][0],BlueDataSet[i][1]]]) - mu_b).
 ↪dot(np.transpose(np.transpose([[BlueDataSet[i][0],BlueDataSet[i][1]]]) -␣
 ↪mu_b))
sigma_2 =[[0,0],[0,0]]
for i in range(len(BlueDataSet)):
    sigma_2 += (np.transpose([[RedDataSet[i][0],RedDataSet[i][1]]]) - mu_r).
 ↪dot(np.transpose(np.transpose([[RedDataSet[i][0],RedDataSet[i][1]]]) - mu_r))
```

```
Sigma = (sigma_1 + sigma_2) / 100
invSigma = np.linalg.inv(Sigma)

def LDAPredict(xcoord, ycoord):
    pred = np.zeros(xcoord.shape)
    for i in np.arange(0, len(xcoord)):
        lin_class_b = 1/2*((np.array([[xcoord[i]],[ycoord[i]]]) - mu_b).T).
 ↪dot(invSigma).dot(np.array([[xcoord[i]],[ycoord[i]]]) - mu_b) - np.log(pi_b)
        lin_class_r = 1/2*((np.array([[xcoord[i]],[ycoord[i]]]) - mu_r).T).
 ↪dot(invSigma).dot(np.array([[xcoord[i]],[ycoord[i]]]) - mu_r) - np.log(pi_r)
        if lin_class_b > lin_class_r:
            pred[i] = 1
        else:
            pred[i] = 0
    return (pred)

Test_prediction = LDAPredict(TestData[:,0], TestData[:,1])
count=0
for i in range(len(Test_prediction)):
    if Test_prediction[i] != TestLabel[i]:
        count+=1
print("test risk=", count/10000)

prediction = LDAPredict(x_mesh.ravel(), y_mesh.ravel())
prediction = prediction.reshape(x_mesh.shape)
```

```
test risk= 0.1843
```

[41]:
```
plt.plot(BlueDataSet[:,0], BlueDataSet[:,1], 'o', markerfacecolor="None",␣
 ↪markeredgecolor='b', label='Blue')
plt.plot(RedDataSet[:,0], RedDataSet[:,1], 'o', markerfacecolor="None",␣
 ↪markeredgecolor='r', label='Red')
plt.xlim([1.1*np.min(TrainingData[:,0]), 1.1*np.max(TrainingData[:,0])])
plt.ylim([1.1*np.min(TrainingData[:,1]), 1.1*np.max(TrainingData[:,1])])
Bayes = plt.contourf(x_mesh, y_mesh, prediction, levels = [-1,0,1], cmap=my_cm,␣
 ↪alpha=0.2)
Bayes = plt.contour(x_mesh, y_mesh, prediction, [-1,0,1,2], linewidths=2,␣
 ↪color='k')
plt.savefig('Q4.jpg')
```

```
/Users/gexueren/opt/anaconda3/lib/python3.7/site-
packages/ipykernel_launcher.py:6: UserWarning: No contour levels were found
within the data range.

/Users/gexueren/opt/anaconda3/lib/python3.7/site-
packages/ipykernel_launcher.py:6: UserWarning: The following kwargs were not
used by contour: 'color'
```



4 QDA

[42]:
```
#Need to create a mesh to evaluate classifier
x_grid = np.linspace(1.0*np.min(TrainingData[:,0]), 1.5*np.max(TrainingData[:
 ↪,0]), 100)
y_grid = np.linspace(1.0*np.min(TrainingData[:,1]), 1.5*np.max(TrainingData[:
 ↪,1]), 100)
x_mesh, y_mesh = np.meshgrid(x_grid, y_grid)
```

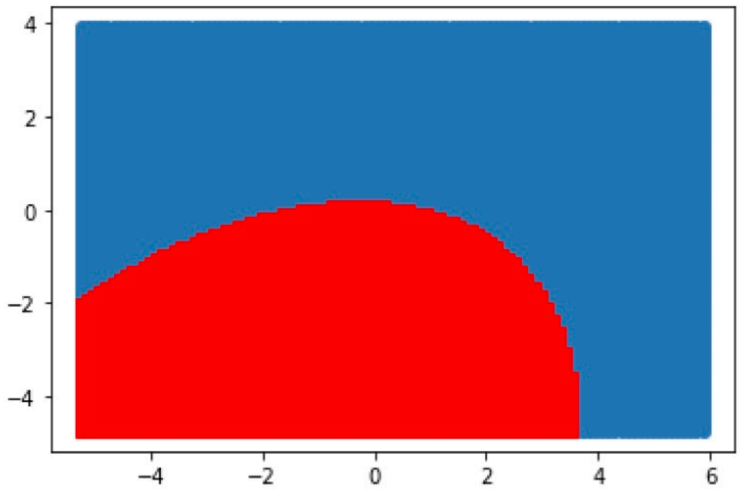
[43]:
```
QDAmodel = discriminant_analysis.QuadraticDiscriminantAnalysis()
QDAmodel.fit(TrainingData, TrainingLabel)

Test_prediction = QDAmodel.predict(TestData)
count=0
for i in range(len(Test_prediction)):
    if Test_prediction[i] != TestLabel[i]:
        count+=1
print("test risk=", count/10000)

PredictedLabel = QDAmodel.predict(np.c_[x_mesh.ravel(),y_mesh.ravel()])
Xcoord = x_mesh.ravel()
Ycoord = y_mesh.ravel()
BlueRegionX = Xcoord[PredictedLabel==0]
```

```
BlueRegionY = Ycoord[PredictedLabel==0]
RedRegionX = Xcoord[PredictedLabel==1]
RedRegionY = Ycoord[PredictedLabel==1]
plt.plot(BlueRegionX,BlueRegionY.ravel(),'o')
plt.plot(RedRegionX,RedRegionY.ravel(),'rs')
```

```
test risk= 0.1078
```

```
/Users/gexueren/opt/anaconda3/lib/python3.7/site-
packages/sklearn/utils/validation.py:760: DataConversionWarning: A column-vector
y was passed when a 1d array was expected. Please change the shape of y to
(n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

[43]:
```
[<matplotlib.lines.Line2D at 0x1a1c3eb090>]
```

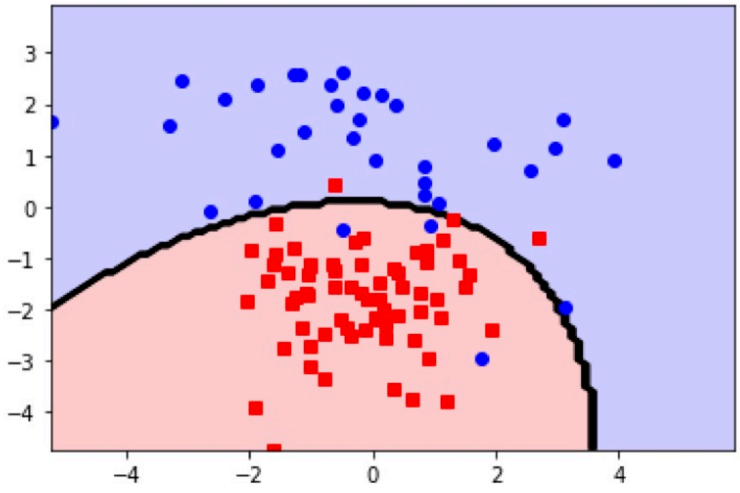


[44]:
```
my_colors = [(0,0,1),(1,0,0)]#RGB
my_cm = LinearSegmentedColormap.from_list('my_cm', my_colors, N=2)
```

[45]:
```
plt.contourf(x_mesh,y_mesh,PredictedLabel.reshape(100,100),levels=[-1,0,1],␣
 ↪cmap = my_cm, alpha=0.2)
# plt.contourf(x_mesh,y_mesh,PredictedLabel.reshape(100,100),cmap = my_cm,␣
 ↪alpha=0.2)
plt.plot(BlueDataSet[:,0],BlueDataSet[:,1],'o',color='b',label='Blue class')
plt.plot(RedDataSet[:,0],RedDataSet[:,1],'s',color='r',label='Red class')
```

```
plt.contour(x_mesh,y_mesh,PredictedLabel.
 ↪reshape(100,100),linewidth=2,colors='k')
plt.savefig('Q5.jpg')
```

```
/Users/gexueren/opt/anaconda3/lib/python3.7/site-
packages/ipykernel_launcher.py:5: UserWarning: The following kwargs were not
used by contour: 'linewidth'
  """
```



5 Logistic Regression

[46]:
```
#Need to create a mesh to evaluate classifier
x_grid = np.linspace(1.0*np.min(TrainingData[:,0]), 1.5*np.max(TrainingData[:
 ↪,0]), 100)
y_grid = np.linspace(1.0*np.min(TrainingData[:,1]), 1.5*np.max(TrainingData[:
 ↪,1]), 100)
x_mesh, y_mesh = np.meshgrid(x_grid, y_grid)
```

[47]:
```
mu_b = np.transpose([[0,1]])
mu_r = np.transpose([[0,-2]])

pi0 = 1/3
pi1 = 2/3
```

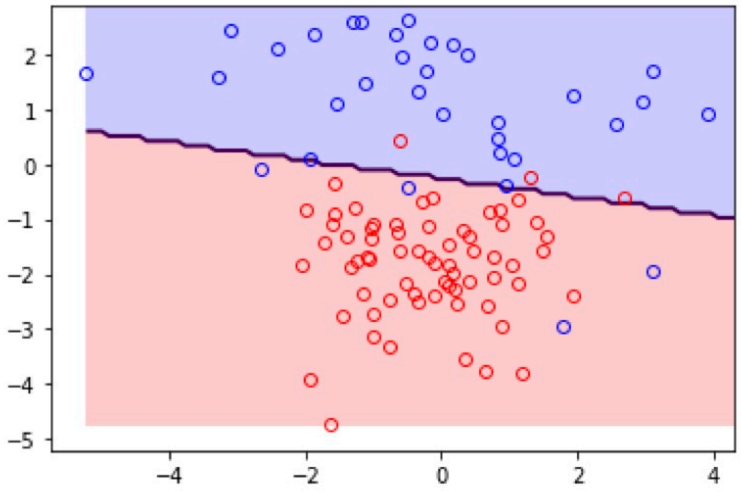
```
sigma_1 =[[0,0],[0,0]]
for i in range(len(BlueDataSet)):
    sigma_1 += (np.transpose([[BlueDataSet[i][0],BlueDataSet[i][1]]]) - mu_b).
 ↪dot(np.transpose(np.transpose([[BlueDataSet[i][0],BlueDataSet[i][1]]]) -␣
 ↪mu_b))
sigma_2 =[[0,0],[0,0]]
for i in range(len(BlueDataSet)):
    sigma_2 += (np.transpose([[RedDataSet[i][0],RedDataSet[i][1]]]) - mu_r).
 ↪dot(np.transpose(np.transpose([[RedDataSet[i][0],RedDataSet[i][1]]]) - mu_r))
Sigma = (sigma_1 + sigma_2)/100

invSigma = np.linalg.inv(Sigma)

w = invSigma.dot(mu_r - mu_b)
b = 0.5*(np.transpose(mu_b).dot(invSigma.dot(mu_b)) - np.transpose(mu_r).
 ↪dot(invSigma.dot(mu_r)))+np.log(pi1/pi0)

def LogisticPredict(xcoord, ycoord):
    pred = np.zeros(xcoord.shape)
    for i in np.arange(0, len(xcoord)):
        lin_class = np.transpose(w).dot([[xcoord[i]], [ycoord[i]]]) + b
        if lin_class >= 0:
            pred[i] = 1
        else:
            pred[i] = 0
    return (pred)

Test_prediction = LogisticPredict(TestData[:,0], TestData[:,1])
count=0
for i in range(len(Test_prediction)):
    if Test_prediction[i] != TestLabel[i]:
        count+=1
print("test risk=", count/10000)

prediction = LogisticPredict(x_mesh.ravel(), y_mesh.ravel())
prediction = prediction.reshape(x_mesh.shape)
```

```
test risk= 0.1182
```

[48]:
```
plt.plot(BlueDataSet[:,0], BlueDataSet[:,1], 'o', markerfacecolor="None",␣
 ↪markeredgecolor='b', label='Blue')
plt.plot(RedDataSet[:,0], RedDataSet[:,1], 'o', markerfacecolor="None",␣
 ↪markeredgecolor='r', label='Red')
plt.xlim([1.1*np.min(TrainingData[:,0]), 1.1*np.max(TrainingData[:,0])])
plt.ylim([1.1*np.min(TrainingData[:,1]), 1.1*np.max(TrainingData[:,1])])
Bayes = plt.contourf(x_mesh, y_mesh, prediction, levels = [-1,0,1], cmap=my_cm,␣
 ↪alpha=0.2)
```

```
Bayes = plt.contour(x_mesh, y_mesh, prediction, [-1,0,1,2], linewidths=2,␣
 ↪color='k')
plt.savefig('Q6.jpg')
```

```
/Users/gexueren/opt/anaconda3/lib/python3.7/site-
packages/ipykernel_launcher.py:6: UserWarning: No contour levels were found
within the data range.

/Users/gexueren/opt/anaconda3/lib/python3.7/site-
packages/ipykernel_launcher.py:6: UserWarning: The following kwargs were not
used by contour: 'color'
```



6 Gaussian Naive Bayes classifier

[49]:
```
GNBmodel = GaussianNB()
GNBmodel.fit(TrainingData,TrainingLabel)
```

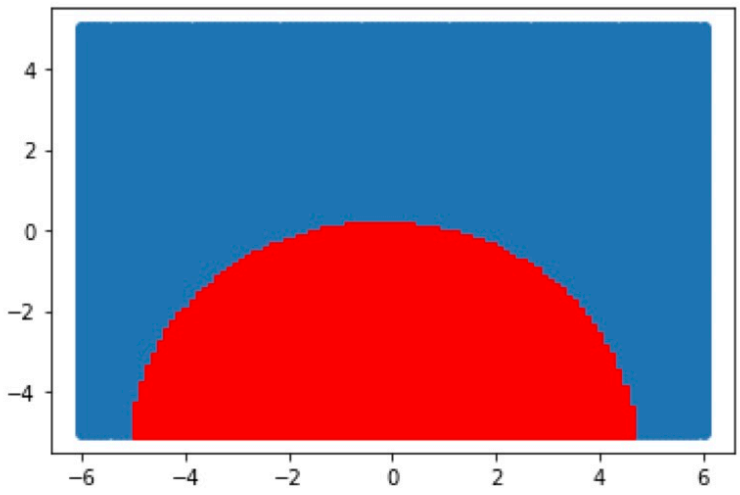
```
/Users/gexueren/opt/anaconda3/lib/python3.7/site-
packages/sklearn/naive_bayes.py:206: DataConversionWarning: A column-vector y
was passed when a 1d array was expected. Please change the shape of y to
(n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

[49]:
```
GaussianNB(priors=None, var_smoothing=1e-09)
```

[50]:
```
x_grid = np.linspace(-6,6,100)
y_grid = np.linspace(-5,5,100)
x_mesh, y_mesh = np.meshgrid(x_grid,y_grid)
PredictedLabel = GNBmodel.predict(np.c_[x_mesh.ravel(),y_mesh.ravel()])

Test_prediction = GNBmodel.predict(TestData)
count=0
for i in range(len(Test_prediction)):
    if Test_prediction[i] != TestLabel[i]:
        count+=1
print("test risk=", count/10000)

Xcoord = x_mesh.ravel()
Ycoord = y_mesh.ravel()
BlueRegionX = Xcoord[PredictedLabel==0]
BlueRegionY = Ycoord[PredictedLabel==0]
RedRegionX = Xcoord[PredictedLabel==1]
RedRegionY = Ycoord[PredictedLabel==1]
plt.plot(BlueRegionX,BlueRegionY.ravel(),'o')
plt.plot(RedRegionX,RedRegionY.ravel(),'rs')
```

```
test risk= 0.1052
```

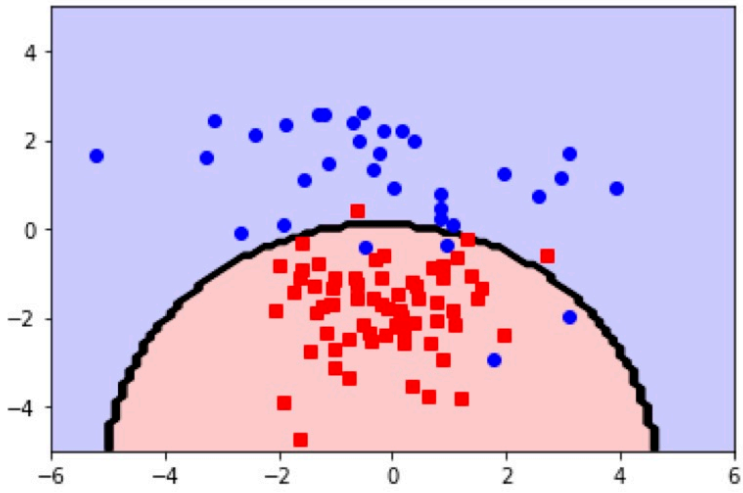
[50]:
```
[<matplotlib.lines.Line2D at 0x1a1c1ce350>]
```



[51]:
```
my_colors = [(0,0,1),(1,0,0)]#RGB
my_cm = LinearSegmentedColormap.from_list('my_cm', my_colors, N=2)
```

[52]:
```
plt.contourf(x_mesh,y_mesh,PredictedLabel.reshape(100,100),levels=[-1,0,1],␣
 ↪cmap = my_cm, alpha=0.2)
# plt.contourf(x_mesh,y_mesh,PredictedLabel.reshape(100,100),cmap = my_cm,␣
 ↪alpha=0.2)
plt.plot(BlueDataSet[:,0],BlueDataSet[:,1],'o',color='b',label='Blue class')
plt.plot(RedDataSet[:,0],RedDataSet[:,1],'s',color='r',label='Red class')
plt.contour(x_mesh,y_mesh,PredictedLabel.
 ↪reshape(100,100),linewidth=2,colors='k')
plt.savefig('Q7.jpg')
```

```
/Users/gexueren/opt/anaconda3/lib/python3.7/site-
packages/ipykernel_launcher.py:5: UserWarning: The following kwargs were not
used by contour: 'linewidth'
  """
```



[ ]: