Machine Learning Lab 2

Junming Zhang 29299527 jz1g17@ecs.soton.ac.uk

Problem 1 and 2

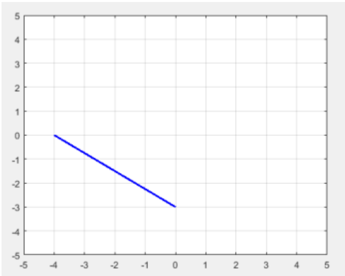

Figure 1.1: line: $3x + 4y + 12 = 0$

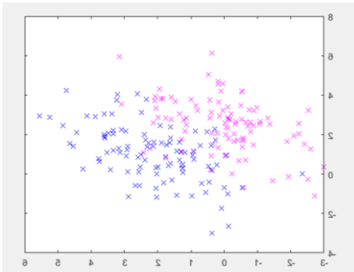

Figure 1.2: Gaussian distribution with different mean

$\mathrm{a1x} + \mathrm{b1y} + \mathrm{c1} = 0$ is a line equation, in this case, $\mathrm{a1} = 3, \mathrm{b1} = 4$ and $\mathrm{c1} = 12$ to two points: $(\frac{-c}{b1} = -3{,}0), (\frac{-c}{a1} = -4{,}0)$ are used to plot the line.

kron command will return the Kronecker product of two matrix. thus, it could be used shifting the mean of one distribution.

Problem 3

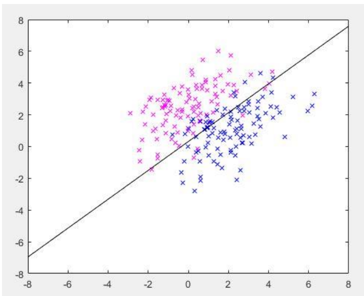

Figure 1.3:Bayes's optimal class boundary

Suppse the densities are isotropic and prios are qual which are $\vec{C} = \sigma^2 * \vec{I}$ *and* $P[w1] = [w2]$

$$\frac{1}{(2\pi)^{\frac{p}{2}}(\det(C))^{\frac{1}{2}}} exp\left\{-\frac{1}{2}(\vec{x}-\vec{m}_1)^t\vec{C}^{-1}(\vec{x}-\vec{m}_1)\right\}P[w1] < | >$$

$$\frac{1}{(2\pi)^{p/2}(\det(C)^{1/2}} exp\left\{-\frac{1}{2}(\vec{x}-\vec{m}_2)^t\vec{C}^{-1}(\vec{x}-\vec{m}_2)\right\}P[w2]$$

Cancel common terms and take log

$$(\vec{x}-\vec{m}_1)^t\vec{C}^{-1}(\vec{x}-\vec{m}_1) < | > (\vec{x}-\vec{m}_2)^t\vec{C}^{-1}(\vec{x}-\vec{m}_2) - log\left\{\frac{P[w1]}{P[w2]}\right\}$$

To rewrite $$\vec{w} = 2\vec{C}^{-1}(\vec{m}_2 - \vec{m}_1) = \begin{bmatrix} 3.3333 \\ -3.6667 \end{bmatrix}$$

And $$b = \left(\vec{m}_1^t \vec{C}^{-1} \vec{m}_1 - \vec{m}_2^t \vec{C}^{-1} \vec{m}_2\right) - log\left\{\frac{P[w1]}{P[w2]}\right\} = 1.6667$$

The boundary is $\vec{w}^t * \vec{x} + b \longrightarrow w_1 x_1 + w_2 x_2 + b \longrightarrow x_2 = -\frac{w_1}{w2} x_1 - \frac{b}{w_2}$

Therefore the line equation is $y = -\frac{3.3333}{3.6667} x_1 + \frac{1.6667}{3.6667}$ which is shown as Figure 1.3.

Problem 4

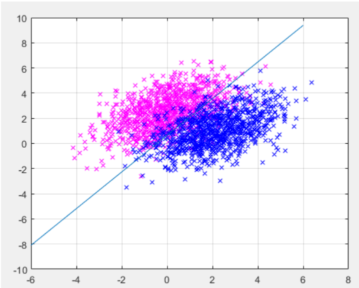

Figure 1.4:Bayes's optimal class boundary

To finish the code in the Appendix, the first step is set up the existed data with label 1 and -1

```
X1 = [Y1 ones(N,1)];% Gaussian with mean = [0 2] is class 1
X2 = [Y2 ones(N,1)];
X2(1:N,3)=-1;% Gaussian with mean = [1.5 0] is class -1
X=[X1;X2];% combine two matrixs into one
%set the class
Y=ones(2*N,1);
Y(N+1:2*N,1)=-1;
% Separate into training and test sets (check: >> doc randperm)
ii = randperm(2*N);%disorganize the sort
Xtr = X(ii(1:N),:);%traning class 1
ytr = Y(ii(1:N),:);%test class 1
Xts = X(ii(N+1:2*N),:);%traning class -1
yts = Y(ii(N+1:2*N),:);%test class -1
```

The second step is set the weight and learning judgement equation to adjust the weight with a correction equation.

The last step is use the learned weight to draw the boundray line for classfication which is shown in Figure 1.4 and observe the Error percentage.

```
% initialize weights
w = randn(3,1);
% Error correcting learning
eta = 0.002;
for iter=1:N
j = ceil(rand*N);% random choose
if ( (ytr(j)*Xtr(j,:))*w < 0 )%learning judgement
w = w + eta*ytr(j)*Xtr(j,:)';%weight adjustment
end
end
% plotting
n1=linspace(-6,6,50);
n2=-(w(1)/w(2))*n1-w(3)/w(2);
plot(n1,n2)
yhts = Xts*w;
PercentageError = (size(find(yts.*yhts < 0),1))/(N);%number of uncorrect weight
```

This part is not finish cause the learning process is unstable due to only trian same input data once, therefore the output weight may wrong in this case. To obtain a high accuracy weight need more train process with same data.