Yang_Jinxin_1168646_homework-1

BRIANYANG1106@GMAIL.COM

1 Question 1

1.1 Generate the elements

First we know it's a bivariate normal distribution, we supposed to use the probability density function:

$$P(x_1, x_2) = \frac{1}{2\pi\sigma_1\sigma_2\sqrt{1-\rho^2}} \exp\left\{-\frac{z}{2(1-\rho^2)}\right\}, \tag{1}$$

where

$$z = \frac{(x_1-\mu_1)^2}{\sigma_1^2} - \frac{2\rho(x_1-\mu_1)(x_2-\mu_2)}{\sigma_1\sigma_2} + \frac{(x_2-\mu_2)^2}{\sigma_2^2} \tag{2}$$

But as the homework says, use σI as the covariance matrix, we can find that covariance matrix's:

$$\begin{pmatrix} \sigma_{\mathrm{xx}} & 0 \\ 0 & \sigma_{\mathrm{yy}} \end{pmatrix}$$

So, since $\sigma_{\mathrm{xx}} = \sigma_{\mathrm{yy}} = \sigma = \mathrm{SS}$, we can use univirate normal distribution to represent the bivariate one. That's why I used univirate normal distribution to generate $\boldsymbol{x}$ and $\boldsymbol{y}$ seperately. For my source code, something should be mentioned:

- I used Java and Eclipse to implement the algorithm and generate the elements
- I exported the project files that you can directly import into Eclipse
- just run generator.java (in the "gauss" package, right click and run application)
- it will generate 100 (x, y) and write into "output.data" for plotting and "output.arff" for Weka, simultaniously into "class1/2.data" for class plotting
- then it will plot all the points by using Gnuplot API and generate a png picture
- after close the plotting window, it will calculate 5 parameters' likelihood and display
- if you cannot run the program, please send email to me

1.2 Plot the dataset

Here is the figure by using Gnuplot API:

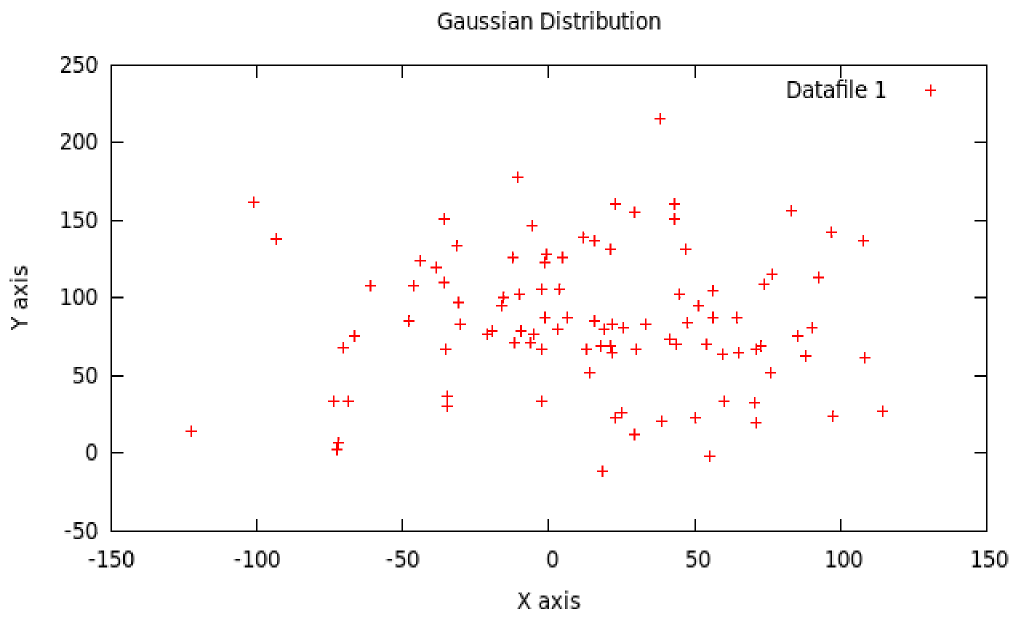

Figure 1. Gaussian dataset

1.3 Evaluate the likelihood

After generating the dataset successfully, we still use Eq. 1 and Eq. 3 to calculate the likelihood which has a $\rho = 0$:

$$P(\mathcal{D}|\theta) = \prod p(x_k|\theta) \tag{3}$$

and the results of 5 parameters are:

$\theta = [\mu_x, \mu_y, \sigma_{x,y}]$	likelihood
$[14, 86, 46]$	1.407106396374 E-459
$[16, 84, 46]$	1.379397361109 E-459
$[18, 86, 46]$	1.117135180463 E-459
$[16, 88, 46]$	1.139575949877 E-459
$[16, 86, 40]$	1.929420458901 E-462

The original $\theta = [16, 86, 46]$, I changed them by different extents to figure out that which one could will be much better. In this case, we want to find the θ which has the biggest likelihood. And we can find that the one has relatively bigger likelihood is exactly closer to the original θ. It just like we use different parameters to test which one is the best one(the red line is origin [1]):

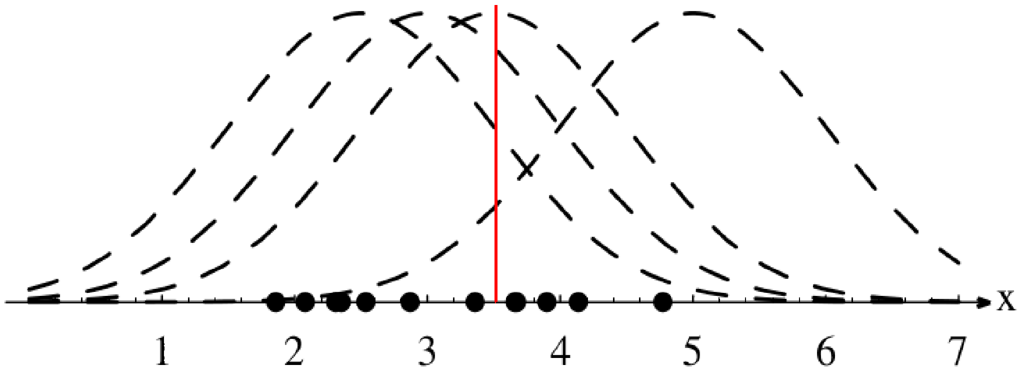

Figure 2. Example for pocessing

2 Question 2

2.1 Create .arff

I create the output.arff in the Java project and it's in "Question2" directory. Here is the class figure:

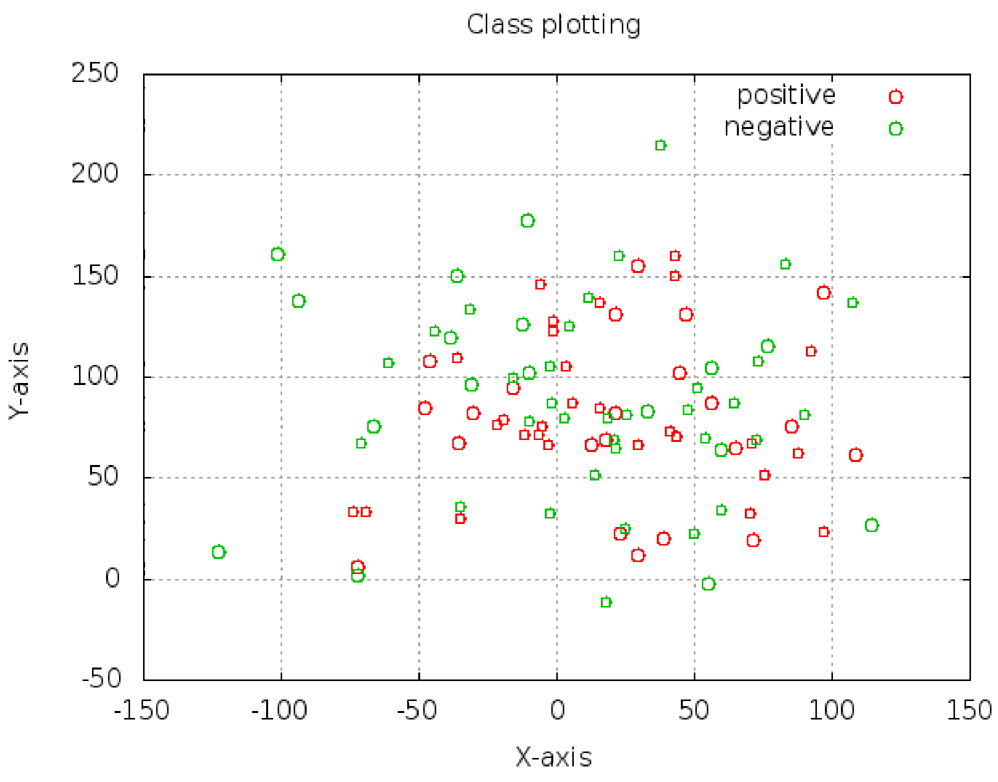

Figure 3. Class plotting

2.2 Screenshots and comparison of three algorithms

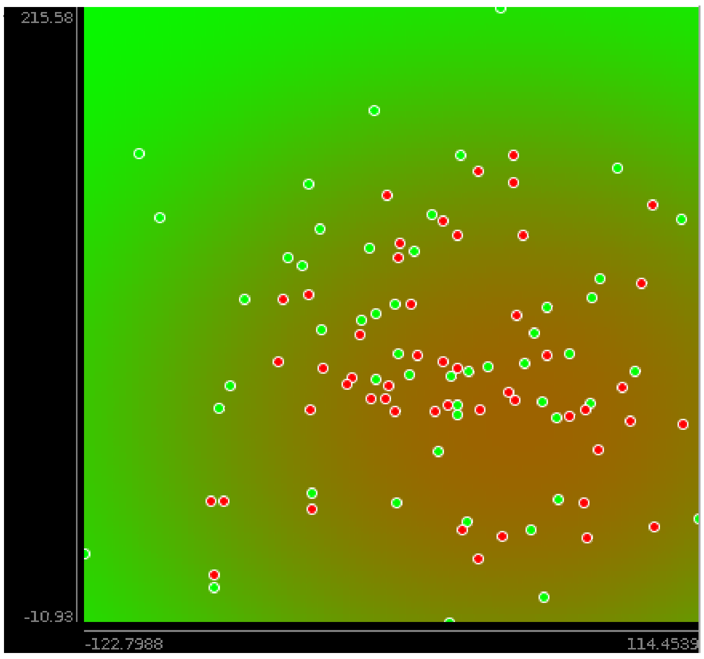

Figure 4. NaiveBayes (NB)

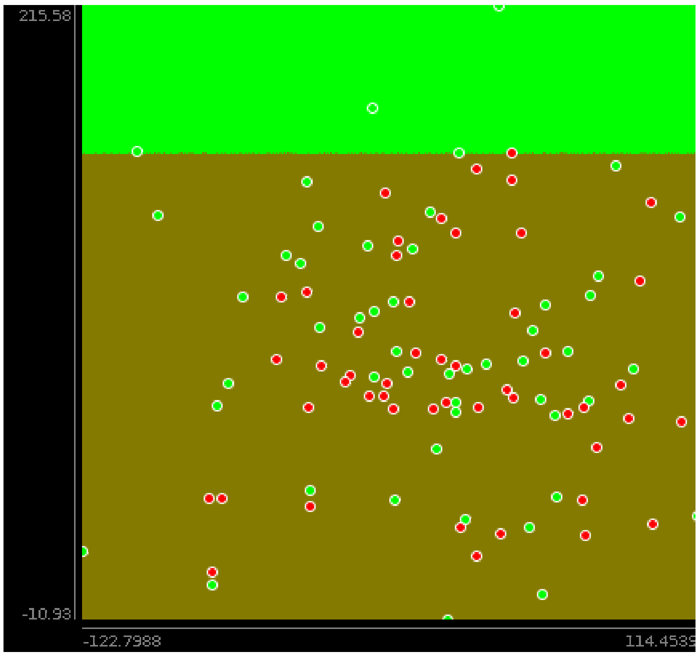

Figure 5. DecisionStump (DS)

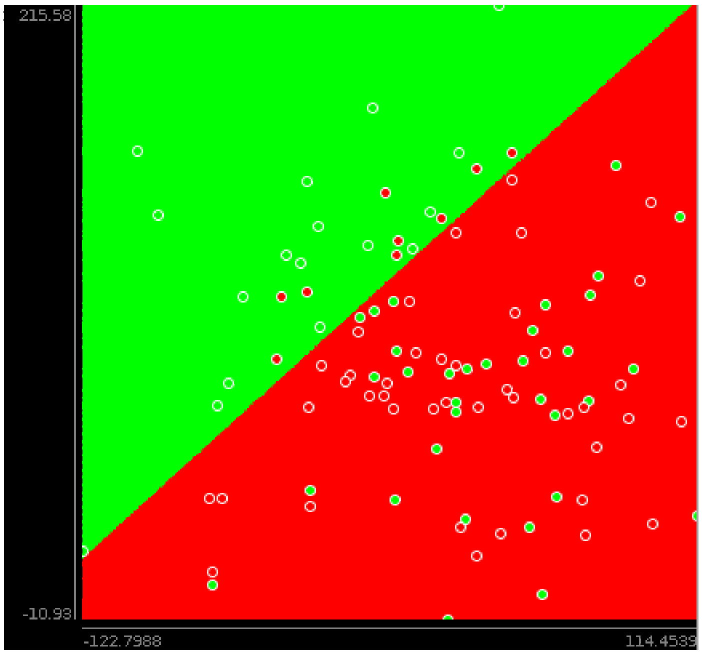

Figure 6. SMO

Without considering the algorithm themselves, through graphics we can find that some area of NB is quite blurry, most part of DS is tan and SMO divides all the points into two part totally. NB is based on Bayes theory and the color of each point of area is decided by the possibilities of red and green points. That's the reason of blurriness. To DS and SMO, they directly draw the boundary but the effects are not that good than NB because there are many misjudgement in the graphics. So here is a simple performance comparison:

$$\text{NB} > \text{SMO} \geqslant \text{DS}$$

2.3 Cross validation

The confusion matrix obtained from these plots (Fig. 1) are shown below:

algorithm	confusion matrix
NB	$\begin{pmatrix} 29 & 21 \\ 29 & 21 \end{pmatrix} \frac{a = \text{positive}}{b = \text{negative}}$
DS	$\begin{pmatrix} 43 & 7 \\ 47 & 3 \end{pmatrix} \frac{a = \text{positive}}{b = \text{negative}}$
SMO	$\begin{pmatrix} 32 & 18 \\ 33 & 17 \end{pmatrix} \frac{a = \text{positive}}{b = \text{negative}}$

Table 1. Confusion matrix

and the weighted avg. parameters:

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area
NB	0.5	0.5	0.5	0.5	0.497	0	0.479	0.512
DS	0.46	0.54	0.389	0.46	0.357	-0.133	0.433	0.465
SMO	0.49	0.51	0.489	0.49	0.478	-0.021	0.49	0.495

Table 2. Weighted avg.

where [2]

		Predicted Label	
		positive	negative
Known Label	positive	True Positive (TP)	False Negative (FN)
	negative	False Positive (FP)	True Negative (TN)

Table 3. Confusion matrix definition

and

$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$	$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$
$F - \text{measure} = \frac{2\,\text{Precision}\,\text{Recall}}{\text{Precision} + \text{Recall}}$	$\text{MCC} = \frac{\text{TP}\,\text{TN} - \text{FP}\,\text{FN}}{\sqrt{(\text{TP}+\text{FP})(\text{TP}+\text{FN})(\text{TN}+\text{FP})(\text{TN}+\text{FN})}}$

Finally, the comparison between 3 algorithms are tabulated:

TP Rate	NB > SMO > DS
FP Rate	NB < SMO < DS
Precision	NB > SMO > DS
Recall	NB > SMO > DS
F-measure	NB > SMO > DS
MCC	NB > SMO > DS

Table 4. Results

Conclusion on performance of the classifiers from the results above:

$$\boldsymbol{NB > SMO > NS}$$

and this is identical with assumption of the Question 2.

Bibliography

[1] Duda. *Pattern Classification.*

[2] Http://webdocs.cs.ualberta.ca/ eisner/measures.html. .