UNIVERSITY OF CAMBRIDGE

**MPhil in Machine Learning, Speech and Language Technology**
**2015-2016**

# MODULE COURSEWORK FEEDBACK

| | |
|---|---|
| Student Name: | Module Title: |
| CRSiD: | Module Code: |
| College: | Coursework Number: |

***I confirm that this piece of work is my own unaided effort and conforms with the Department of Engineering guidelines on plagiarism***

I declare the word count for this piece is:

Student's Signature:

---

| | |
|---|---|
| Date Marked: | Marker's Name: |

**This piece of work has been completed to a standard which is** *(Please give mark as appropriate)*:

**Marker's Comments:**

# Introduction to Machine Learning and Spoken Language Processing: Machine Learning Coursework

A.

Class 1 data labeled in ‘x’ and Class 0 data labeled in ‘o’ is shown below. The data will not be linear separable by using a classifier with a linear class boundary.

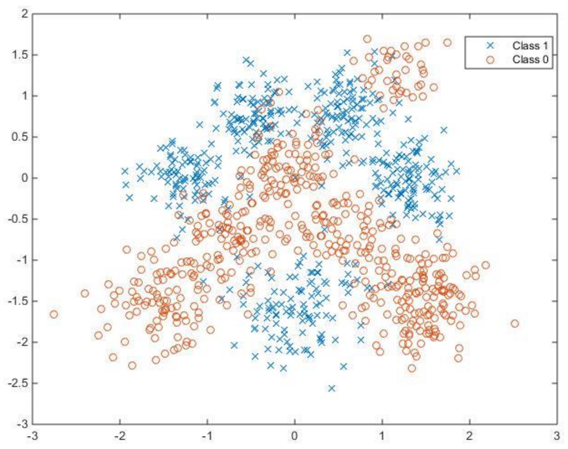
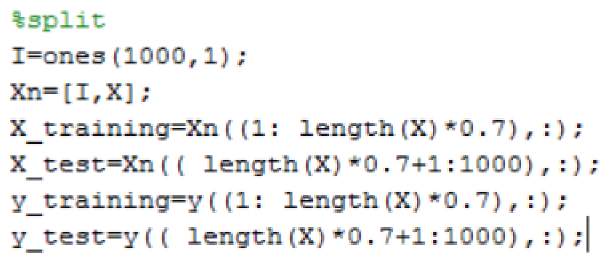


Figure 1, Visualisation of the dataset

B.

The code for split data is show below. Firstly we need to add up a bias, because in linear regession, when all the features are zero, the predicted value would be zero as well, due to outcome passing through the origin. However, the purpose for training is not to get the zero, we need to add a bias parameter to fit data much easier that doesn't pass through the origin because the bias is independent of any input features.



```
%split
I=ones(1000,1);
Xn=[I,X];
X_training=Xn((1: length(X)*0.7),:);
X_test=Xn(( length(X)*0.7+1:1000),:);
y_training=y((1: length(X)*0.7),:);
y_test=y(( length(X)*0.7+1:1000),:);
```

After adding bias, I split the first 700 rows input feature to be training data and the rest of 300 input features to be test data.  I also need to split the first 700 rows binary vector to be training data and the rest of 300 binary vector to be test data because I need to make sure the class label is correctly indicate to the corresponding input features.

C.

The log-Likelihood for $\beta$ is:

$$\mathcal{L}(\beta) = \ln P(\mathbf{y}|\mathbf{X},\beta) = \sum_n \ln P(y^{(n)}|\tilde{\mathbf{x}}^{(n)},\beta)$$

$$= \sum_n [y^{(n)}\ln\sigma(\beta^T\tilde{x}^{(n)}) + (1-y^{(n)})\ln\sigma(-\widetilde{\beta^T\mathbf{x}}^{(n)})]$$

Taking derivatives, using $\frac{\partial\ln\sigma(\beta^T\tilde{x}^{(n)})}{\partial\beta} = \sigma(-\beta^T\tilde{x}^{(n)})$, we get:

$$\frac{\mathcal{L}(\beta)}{\partial\beta} = \sum_n [y^{(n)}\sigma(-\beta^T\tilde{x}^{(n)})\tilde{x}^{(n)} - (1-y^{(n)})\sigma(\beta^T\tilde{\mathbf{x}}^{(n)})\tilde{x}^{(n)}]$$

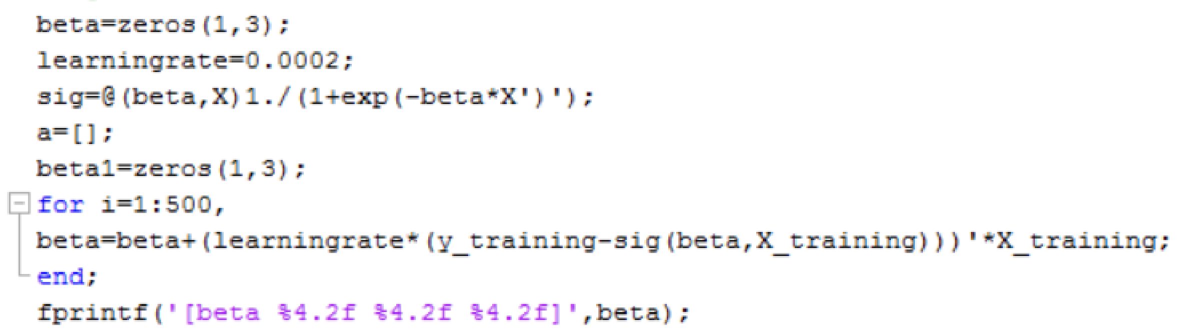
$$= \sum_n [y^{(n)}\sigma(-\beta^T\tilde{x}^{(n)})\tilde{x}^{(n)} + y^{(n)}\sigma(\widetilde{\beta^T\mathbf{x}}^{(n)})\tilde{x}^{(n)} - \sigma(\beta^T\tilde{\mathbf{x}}^{(n)})\tilde{x}^{(n)}]$$

$$= \sum_n [y^{(n)} - \sigma(\beta^T\tilde{\mathbf{x}}^{(n)})\tilde{x}^{(n)}]$$

Therefore, the gradients of the log-likelihood of the parameters, $\frac{\mathcal{L}(\beta)}{\partial\beta} = \sum_n [y^{(n)} - \sigma(\beta^T\tilde{\mathbf{x}}^{(n)})\tilde{x}^{(n)}]$

D.

The following code is the matlab code for estimating the parameter $\beta$. In this section, the linear classification can be expressed as: $P(y^{(n)} = 1|\tilde{\mathbf{x}}^{(n)},\beta) = \sigma(\beta^T\tilde{\mathbf{x}}^{(n)})$ , where $\sigma(\beta^T\tilde{\mathbf{x}}^{(n)})$ is the sigmoid function which is: $\sigma(\beta^T\tilde{\mathbf{x}}^{(n)}) = \frac{1}{1+\exp(-\beta^T\tilde{\mathbf{x}}^{(n)})}$ The method for $\beta$ evaluation is using steepest gradient ascent which is the first order optimization algorithm. The gradient of the log-likelihood gives the optimum direction and the step for current $\beta$ to obtain the next $\beta$. Learning rate is the step size, and it is a constant, if the learning rate is too small, convergence will be very small, and if the learning rate is too big, it will never converge to the optimum. In this case, we want the $\beta$ converge to a certain value which is the optimum value. For the code, I choose learning rate too be 0.0002, and I train it for 500times, but after 400 times training, the $\beta$ is converge to [0.38, 0.13, 0.91]



```
beta=zeros(1,3);
learningrate=0.0002;
sig=@(beta,X)1./(1+exp(-beta*X')');
a=[];
beta1=zeros(1,3);
for i=1:500,
beta=beta+(learningrate*(y_training-sig(beta,X_training)))'*X_training;
end;
fprintf('[beta %4.2f %4.2f %4.2f]',beta);
```

E.

The log-likelihood on training and test datasets  and the prediction by adding probability controus to the plots in part a are shown in figure 2 and figure 3.

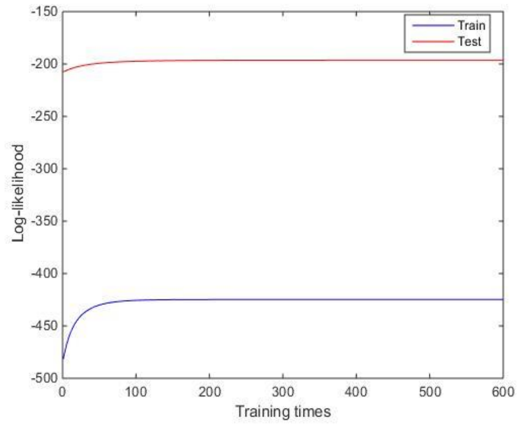


Figure 2, the log-likelihood on training and test dataset

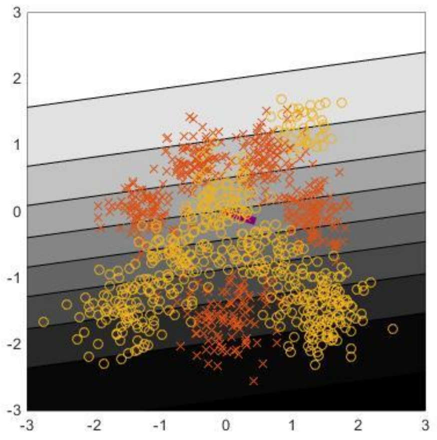
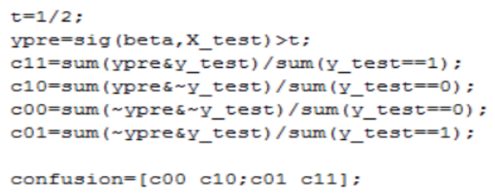


Figure 3, Probability contour

F.

The final training likelihood is 0.5843, and the final test likelihood is 0.5386.

To produce a 2x2 confusion matrix, I am trying to produce a binary vector containing class label for prediction which is shown in the code below.



```
t=1/2;
ypre=sig(beta,X_test)>t;
c11=sum(ypre&y_test)/sum(y_test==1);
c10=sum(ypre&~y_test)/sum(y_test==0);
c00=sum(~ypre&~y_test)/sum(y_test==0);
c01=sum(~ypre&y_test)/sum(y_test==1);

confusion=[c00 c10;c01 c11];
```

According to the code, which provide a 2x2 confusion matrix as shown at table 1

| | | Predicted label, $\hat{y}$ | |
|---|---|---|---|
| | | 0 | 1 |
| Ture label, y | 0 | 0.6752 | 0.3248 |
| | 1 | 0.3287 | 0.6713 |

Table 1, 2x2 confusion matrix

G.

Figure 4 illustrates the prediction's ROC curve. The threshold is varied from 0 to 1.

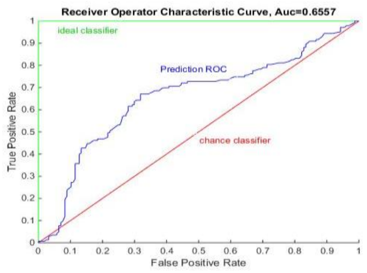
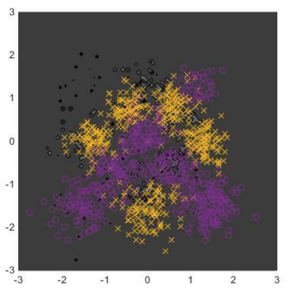


Figure 4, ROC curve of predictions

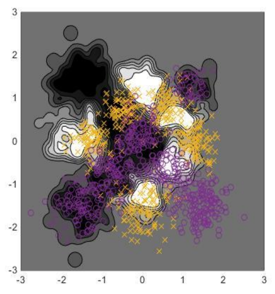
A chance classifier has an area of 0.5, while and ideal one has an area of 1, and the area under the predicting classifier is 0.6557. True positive rate $= \frac{\sum \text{True positive}}{\sum \text{condition positive}}$, and the

False positive rate $= \frac{\sum \text{false positive}}{\sum \text{condition negativ}}$. The performance of this predicting classifier is bad because the area is 0.6557, which shows the evidence that the false positive rate is large resulting in lots of poor predictions. The predicting classifier is closed to chance classifier, which is also a sign that the predicting classifier has error.

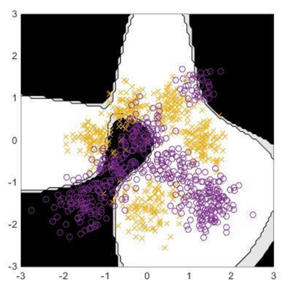
H.



Prediction when l=0.01



prediction when l=0.1

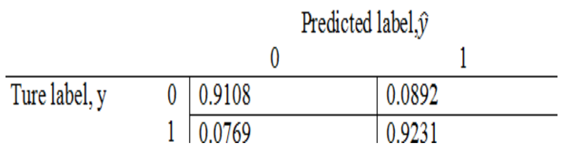


Prediction when l=1

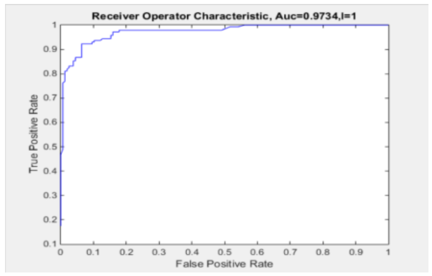
The prediction for RBF width l varied {0.01, 0.1, 1} is shown above. It's a non-linear logistic classification that the decision boundary is vary when l is vary. It's seem when l=0.01, prediction is under-fitting to the output, and when l=0.1, the prediction is over-fitting, and when l=1, the prediction is just in fit.

I)

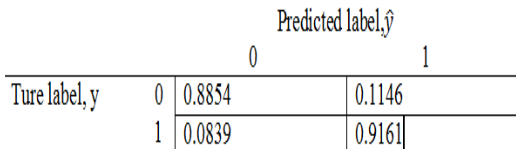
The 2x2 confusion matrixes and the ROC curves with the area under the curves for the model trained by RBF input features with three width l={1, 0.1,0.01} are shown below



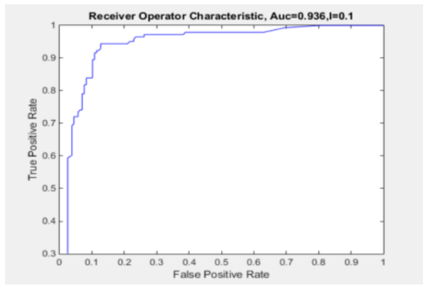
| | | Predicted label,$\hat{y}$ | |
|---|---|---|---|
| | | 0 | 1 |
| Ture label, y | 0 | 0.9108 | 0.0892 |
| | 1 | 0.0769 | 0.9231 |



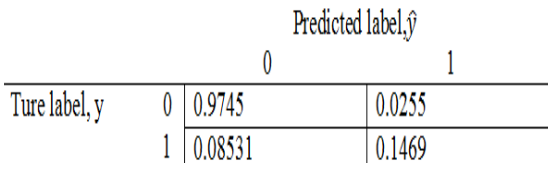
The confusion matrix and ROC curve provided above is when l=1, the final training and test likelihood are0.8737 and 0.8412. The Area under the ROC curve is 0.9734



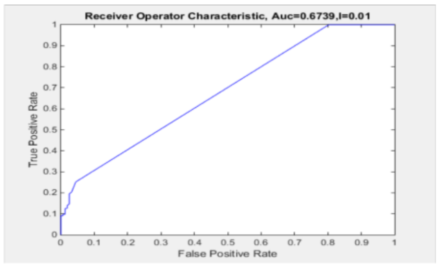
| | | Predicted label,$\hat{y}$ | |
|---|---|---|---|
| | | 0 | 1 |
| Ture label, y | 0 | 0.8854 | 0.1146 |
| | 1 | 0.0839 | 0.9161 |



The confusion matrix and ROC curve provided above is when l=0.1, the final training and test likelihood are 0.9382and 0.8604. The Area under the ROC curve is 0.936



| | | Predicted label,$\hat{y}$ | |
|---|---|---|---|
| | | 0 | 1 |
| Ture label, y | 0 | 0.9745 | 0.0255 |
| | 1 | 0.08531 | 0.1469 |



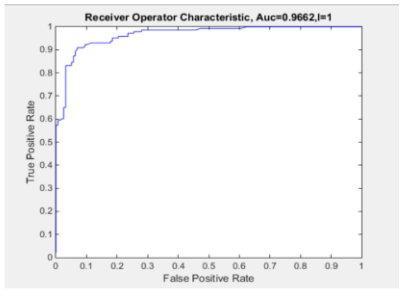
The confusion matrix and ROC curve provided below is when l=0.01, the final training and test likelihood are0.9963 and 0.5112 The Area under the ROC curve is 0.6739.

Comparing the results by using the radial basis functions with the orignal inputs, we can find that by comparing the area under the ROC curve, the radial basis functions provides a better classifier . This is because we have more ture positive ratio using RBF as features input rather than using original input as shown in three confusion matrixs above. We also can find that the final training and test likelihoods provided by using the RBF features as input is much larger than the original lielihoods, this is because β has been trained more accurately to the optimum value. When l=1, the log-likelihood converges slowest, and l=0.01, the log-likelihood converges fastest.

j)

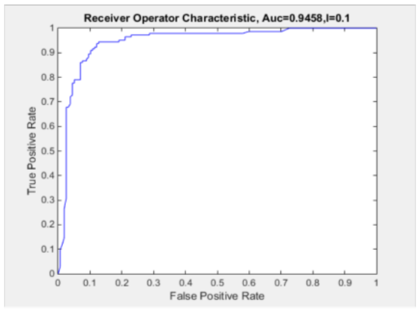
The 2x2 confusion matrixes and the ROC curves with the area under the curves for the models width l={1, 0.1,0.01} are shown below

| | | Predicted label,$\hat{y}$ | |
|---|---|---|---|
| | | 0 | 1 |
| Ture label, y | 0 | 0.8981 | 0.1019 |
| | 1 | 0.0769 | 0.9231 |



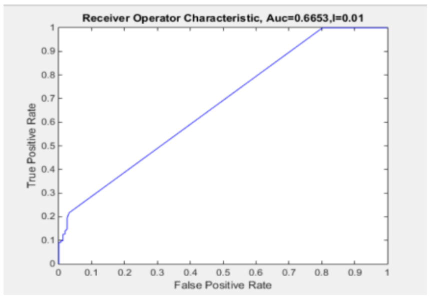
The confusion matrix and ROC curve provided above is when l=1, the maximum training posterior is. 0.85 , and maximum test posterior is 0.68. The Area under the ROC curve is 0.9662

| | | Predicted label,$\hat{y}$ | |
|---|---|---|---|
| | | 0 | 1 |
| Ture label, y | 0 | 0.9045 | 0.0955 |
| | 1 | 0.1049 | 0.8951 |



The confusion matrix and ROC curve provided above is when l=0.1, the maximum training posterior is 0.90, and maximum test posterior is 0.65. The Area under the ROC curve is 0.9458

| | | Predicted label,$\hat{y}$ | |
|---|---|---|---|
| | | 0 | 1 |
| Ture label, y | 0 | 0.9682 | 0.0318 |
| | 1 | 0.7902 | 0.2098 |



The confusion matrix and ROC curve provided below is when l=0.01, the maximum posterior is 0.99, and the maximum test posterior is 0.48. The Area under the ROC curve is 0.6653.

Comparing the results by adding a Gaussian prior over the parameter with the orignal results and the model trainned by using RBF input features. It's show that the classifier performance from maximsing converge faster than the classfiers in i), this is because the gradient was regularized by β.