University of Cambridge

Intro to Machine Learning and Spoken Language Processing

Logistic Classification Lab

Authors:

Joshua Aduol, jna29

June 30, 2016

1 Answers + Notes

a There is no linear boundary to separate the two classes and so a classifier that uses linear basis functions will give a very low accuracy (closer to 50% as each point would be randomly classified)

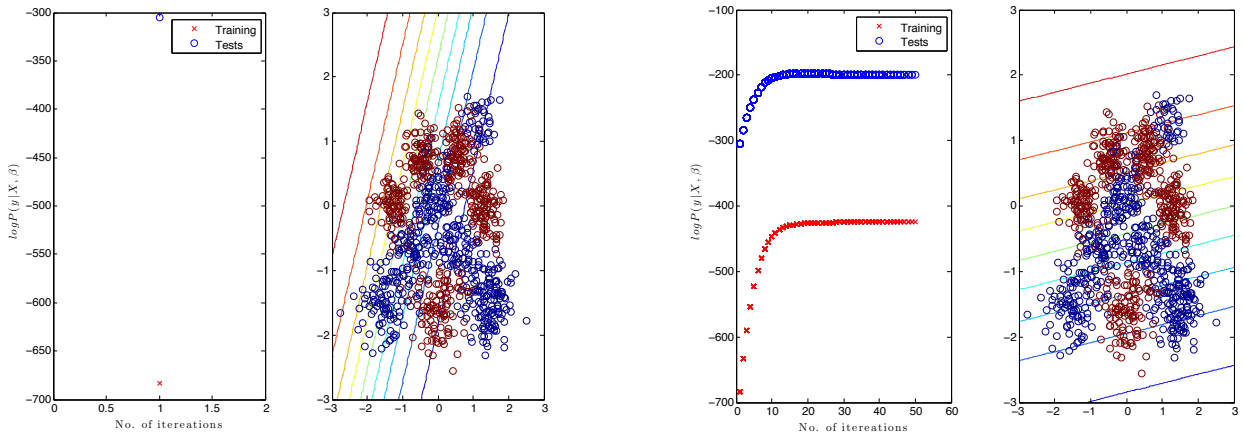
b A 70:30 (training:test) split was used as this is a recommended split in the literature (Validation set?).

c Gradient of the likelihood is given as:

$$\frac{\partial \mathcal{L}(\beta)}{\partial \beta} = \sum_{n=1}^{N} \left(y^{(n)} - \sigma(\beta^T \tilde{\mathbf{x}}^{(n)}) \right) \tilde{\mathbf{x}}^{(n)} \tag{1}$$

d The function `estimateParams.m` takes in a data matrix whereby each row corresponds to a data point. The parameters are then assigned random values initially and gradient ascent is then performed until the norm of the error between the current and previous estimate is below a threshold. The learning rate was chosen large enough such that gradient ascent takes a large number of iterations yet it wasn't too big to cause the gradient ascent to miss the peak of the likelihood function.

e Figure 1 below shows the training and test curves for the linear classifier and the corresponding probability contour at that iteration.



(a) After 1 step **(b)** After 50 steps

Figure 1: Training curves and likelihood contour for training of the linear classifier

f We can see (Figure 1) that the likelihood of the test data is greater than that of the training data because the linear classifier doesn't actually produce a useful boundary and so the points are classified randomly (with equal probabilities for each class) meaning that as the test dataset has fewer points than the training data, it can be fit using a wider range of parameters thus leading to the likelihood being higher. This is evidenced by the ROC area (Figure 2) of 0.5065 which backs up the hypothesis that the learned classifier is acting like a random one.

However, when the log-likelihoods per data point are considered (-0.61843 for the training data and -0.68561 for the test data), we see that the training data has the greater likelihood which is what we expected initially as the parameters are fit to the training data. The small difference could then be due to the reasons explained above. The confusion matrix obtained with $\tau = 0.5$ (Table 1) is clearly far from ideal and thus illustrates the poor performance of the classifier.

Table 1: Confusion matrix for $\tau = 0.5$

		$\hat{y}$	
		0	1
y	0	0.52229	0.47771
	1	0.27273	0.72727

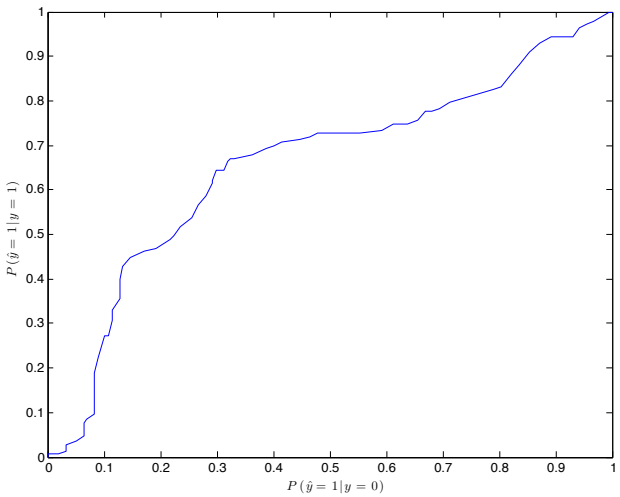


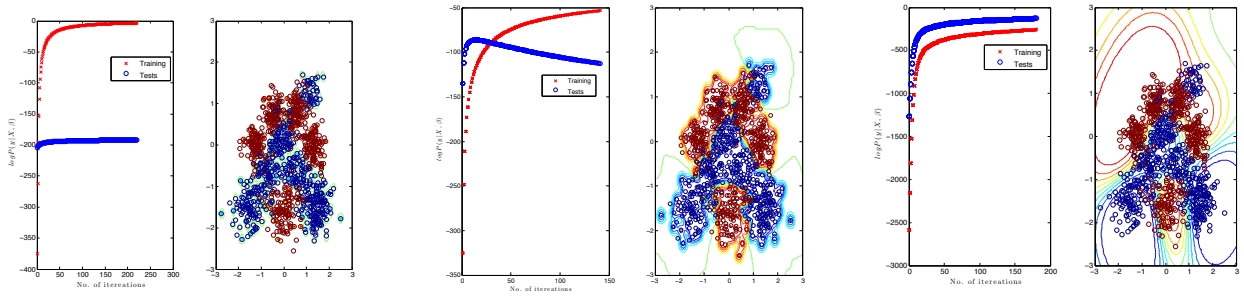
Figure 2: ROC for $\tau = 0.5$ with the linear classifier

g With non-linear RBFs, we are able to obtain better decision boundaries but the sharpness of this boundary depends on the l parameter of the new feature expanded outputs. Figure 3 shows how the choice of l affects the decision boundaries obtained and how the corresponding log-likelihoods for the training and test data vary with training.

With $l = 0.01$, the decision boundary (green contours) is fit around the $y = 0$ data points sharply thus suggesting that the training data may have been overfit. This is backed up by the training likelihood being much greater than the test likelihood indicating that the learned parameters don't generalise well to unseen data (and the corresponding training and test log-likelihoods per data point of -0.0053966 and -0.64008, respectively).

When $l = 1$, as the test likelihood is slightly greater than the training likelihood, this indicates that the learned model may be too general and the optimal parameters haven't been found yet. This can also be inferred from the contour plot whereby the contours are very wide indicating that the learned parameters are too liberal in classifying the points.

With $l = 0.1$, a balance between the two extremes is achieved whereby the probability contour obtained clusters the data well without generalising too much to unseen data points or overfitting the training data. This is backed up by the likelihood plots in which the log-likelihood for the training data isn't as high as it was for $l = 0.01$ and the test log-likelihood isn't as low as before. This setting can also be seen to be a better classifier than the other two settings because its ROC area (0.8884) was the greatest ($l = 0.01$, area $= 0.5428$ and $l = 1$, area $=0.7422$).



(a) $l = 0.01$ **(b)** $l = 0.1$ **(c)** $l = 1$

Figure 3: Likelihood + contour plots for $l = 0.01, 0.1\&1$

Confusion matrices

Table 2: $l = 0.01$

		$\hat{y}$	
		0	1
y	0	0.89172	0.10828
	1	0.35664	0.64376

Table 3: $l = 0.1$

		$\hat{y}$	
		0	1
y	0	0.88535	0.11465
	1	0.062937	0.93706

Table 4: $l = 1$

		$\hat{y}$	
		0	1
y	0	0.87261	0.12739
	1	0.06993	0.93007

h Adding a Gaussian prior over the parameters that acts like an L2 norm regularizer for the parameters corresponds to subtracting the previous parameter estimate from the likelihood gradient i.e.

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

The main takeaway from training the classifier with this is that the final likelihoods for the test and training datasets are more similar than earlier because the regularization stops the parameters from overfitting the training data. The drawback is that the accuracy decreases as indicated by the somewhat lower ROC areas (0.052, 0.7785 and 0.7476 for $l = 0.01, 0.1$ & 1, respectively)