3F8: Inference
Short Lab Report

Shanzi (Monica) Ran

February 19, 2025

Abstract

This lab report records investigations on performance of the logistic classifier and non-linear feature expansion mechanism (RBF). This is implemented using a simple binary categorised dataset, and the performance of classification models is assessed by the comparison of average log-likelihood between training and test data and error probabilities recorded by the confusion matrix. It has been shown in the exercise that while a simple linear classifier fails to capture complexity of the dataset, RBF-transformed input successfully resolves this issue by providing more flexible decision boundaries.

1 Introduction

In the lab exercise recorded by this report, the properties of a simple logistic classifier is explored by applying it to a binary dataset, followed by investigation of metrics like average log-likelihood and confusion matrices. As classification is a fundamental task of machine learning, this investigation is necessary for the understanding of capabilities of different models and parameters. In this report, this classifier architecture is investigated both theoretically and practically, while non-linear feature expansion techniques are also explored as an alternative method to improve the classification performances.

2 Exercise a)

In this exercise we have to consider the logistic classication model (aka logistic regression) and derive the gradients of the log-likelihood given a vector of binary labels $\mathbf{y}$ and a matrix of input features $\mathbf{X}$. The gradient of the log-likelihood can be writen as

$$\begin{aligned}\frac{\partial \mathcal{L}(\beta)}{\partial \beta} &= \frac{\partial}{\partial \beta}\prod_{n=1}^{N} log\{\sigma(\beta^T\tilde{x}^{(n)})^{y^{(n)}}(1-\sigma(\beta^T\tilde{x}^{(n)}))^{1-y^{(n)}}\} \\ &= \sum_{n=1}^{N}\{y^{(n)}\frac{\partial}{\partial \beta}log\{\sigma(\beta^T\tilde{x}^{(n)})\} + (1-y^{(n)})\frac{\partial}{\partial \beta}log\{1-\sigma(\beta^T\tilde{x}^{(n)})\}\}\end{aligned}$$

Using derivative results (substituting $z = \beta^T\tilde{x}^{(n)}$):

$$\frac{\partial}{\partial z}\log\sigma(z) = \frac{1}{\sigma(z)}\sigma(z)(1-\sigma(z)) = 1-\sigma(z)$$
$$\frac{\partial}{\partial z}\log(1-\sigma(z)) = \frac{1}{1-\sigma(z)}(-\sigma(z)(1-\sigma(z))) = -\sigma(z)$$

We can then write:

$$\begin{aligned}\frac{\partial \mathcal{L}(\beta)}{\partial \beta} &= \sum_{n=1}^{N} \{y^{(n)}(1-\sigma(\beta^T \tilde{x}^{(n)}))\tilde{x}^{(n)} - (1-y^{(n)})\sigma(\beta^T \tilde{x}^{(n)})\tilde{x}^{(n)}\} \\ &= \sum_{n=1}^{N} \{y^{(n)} - \sigma(\beta^T \tilde{x}^{(n)})\}\tilde{x}^{(n)}\end{aligned}$$

3 Exercise b)

In this exercise we are asked to write pseudocode to estimate the parameters β using gradient ascent of the log-likelihood. Our code should be vectorised. The pseudocode to estimate the parameters β is shown below:

```
Function estimate_parameters:

   Input:  feature matrix X, labels y
   Output: vector of coefficients b

    max_iter = 1000 # maximum number of iterations set to avoid infinite training
    eta = 0.01 # learning rate (choice explained below)
    b = zeros(D) # initialise coefficients vector with dimention D
    for iter in range(max_iter):
        grad_old = X.T @ (y - sigmoid(X @ b)) # calculate old gradient
        b_new = b + eta * grad_old # gradient ascent
        if norm(b_new - b) == 0: # check whether has reached local maximum
            break
        b = b_new
   return b
```

The learning rate parameter η is chosen to avoid both divergence due to too large η and slow convergence due to too small η. Therefore, it is here set to the widely accepeted default value of 0.01.

4 Exercise c)

In this exercise we visualise the dataset in the two-dimensional input space displaying each datapoint's class label. The dataset is visualised in Figure 1. By analising Figure 1 we conclude that a linear classifier would not be able to perform the classification task satisfactorily due to the complex structure of the dataset.

5 Exercise d)

In this exercise we split the data randomly into training and test sets with 800 and 200 data points, respectively. The pseudocode from exercise a) is transformed into python code as follows:

```
sigmoid_value = predict(X_tilde_train, w)

gradient = X_tilde_train.T @ (y_train - sigmoid_value)

w = w + alpha * gradient
```

We then train the classifier using this code. We fixed the learning rate parameter to be $\eta = 0.001$ to inhibit oscillations. The average log-likelihood on the training and test sets as the optimisation proceeds are shown in Figure 2. By looking at these plots we conclude that for the given ratio of dataset split, the average

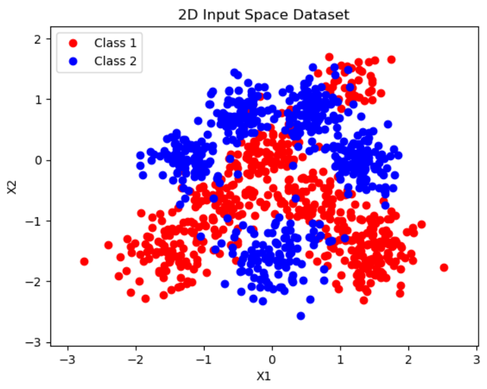

Figure 1: Visualisation of the data.

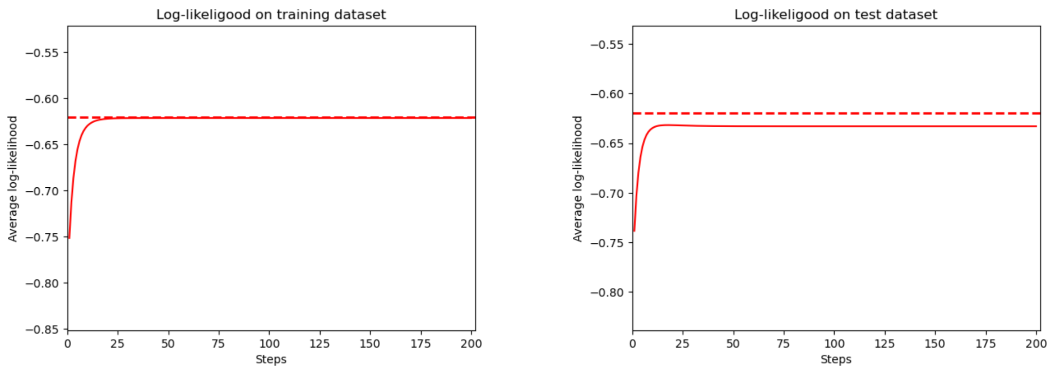

Figure 2: Learning curves showing the average log-likelihood on the training (left) and test (right) datasets.

log-likelihood value on the training dataset converges to a higher value than that of the test dataset. This difference is not large though, which is suspected to be the consequence of too much training data.

Figure 2 displays the visualisation of the contours of the class predictive probabilities on top of the data. This figure shows that the linear classifier model does not separate the categories clearly when applied to the original dataset. Therefore, alternative non-linear feature expansion methods like the RBF is explored in later sections of this report.

6 Exercise e)

The final average training and test log-likelihoods are shown in Table 1. These results indicate that the classifier performs better with the training dataset than with the test dataset, which is reasonble result. The 2x2 confusion matrices on the and test set is shown in Table 2. By analising this table, we conclude that the probability of error for linear classification is approximately equal for the two classes, with an value $P_e \approx 0.28$ (taking average). This error probability agrees with expectation that pure linear classification is not sufficiently complex and accurate for the dataset used.

7 Exercise f)

We now expand the inputs through a set of Gaussian radial basis functions centred on the training datapoints. We consider widths $l = \{0.01, 0.1, 1\}$ for the basis functions. We fix the learning rate parameter to be

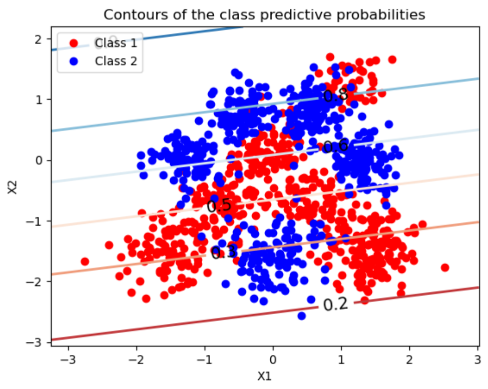

Figure 3: Visualisation of the contours of the class predictive probabilities.

Avg. Train ll	**Avg. Test ll**
-0.627	-0.609

Table 1: Average training and test log-likelihoods.

		$\hat{y}$	
		0	1
y	0	0.7075	0.2925
	1	0.2753	0.7247

Table 2: Confusion matrix on the test set.

$\eta = \{0.01, 0.01, 0.0001\}$ for each $l = \{0.01, 0.1, 1\}$, respectively. Figure 4 displays the visualisation of the contours of the resulting class predictive probabilities on top of the data for each choice of $l = \{0.01, 0.1, 1\}$.

8 Exercise g)

The final final training and test log-likelihoods per datapoint obtained for each setting of $l = \{0.01, 0.1, 1\}$ are shown in tables 3, 4 and 5. These results indicate that as l value increases, the classifier performance on training dataset gets worse, but it also exhibits better adaptation to unobserved test dataset, which indicate that although the model is less sensitive and accurate when identifying details, it has successfully prevented overfitting as depicted in Fig 4. The 2×2 confusion matrices for the three models trained with $l = \{0.01, 0.1, 1\}$ are show in tables 6, 7 and 8. After analysing these matrices, we can say that although $l = 0.01$ and $l = 1$ settings produce good classification results for one of the categories, it is $l = 0.1$ that gives the most balanced confusion matrix, where the error probability of each category is approximately the same

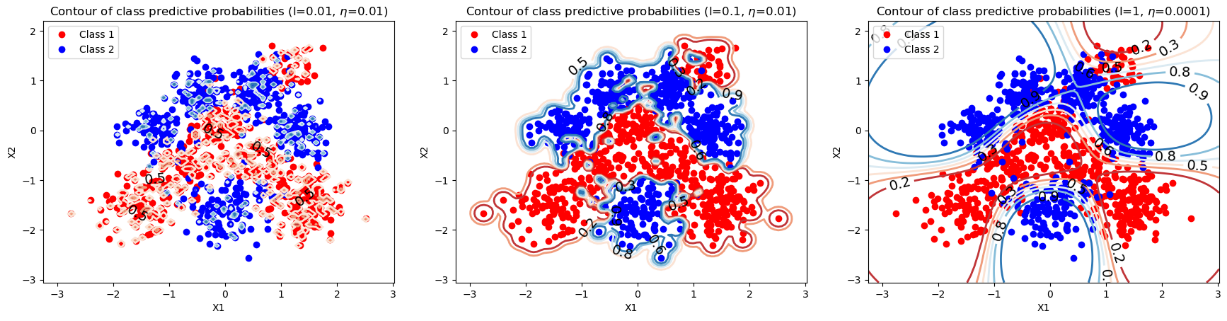

Figure 4: Visualisation of the contours of the class predictive probabilities for $l = 0.01$ (left), $l = 0.1$ (middle), $l = 1$ (right).

Avg. Train ll	**Avg. Test ll**
-0.013	-0.664

Table 3: Results for $l = 0.01$

Avg. Train ll	**Avg. Test ll**
-0.036	-0.463

Table 4: Results for $l = 0.1$

Avg. Train ll	**Avg. Test ll**
-0.244	-0.233

Table 5: Results for $l = 1$

		$\hat{y}$ 0	1
y	0	0.996	0.004
	1	0.174	0.826

Table 6: Conf. matrix $l = 0.01$.

		$\hat{y}$ 0	1
y	0	0.945	0.055
	1	0.049	0.951

Table 7: Conf. matrix $l = 0.1$.

		$\hat{y}$ 0	1
y	0	0.887	0.113
	1	0.079	0.921

Table 8: Conf. matrix $l = 1$.

value 0.05. When we compare these results to those obtained using the original inputs we conclude that all three models produce better classification results than that of the initial model. Hence, non-linear expansion of features like RBF could significantly boost the performance of linear classifier regardless of RBF settings. Combining with decision boundary contours displayed in Fig 4, it is then concluded that non-linear feature expansion is a useful technique when classifying datasets with complex features.

9 Conclusions

In this lab exercise, the logistic regression model is explored as a method for binary data classification. It has been observed that although simple implementation of the linear model fails to accurately capture the complexity of the dataset used, pre-processing the dataset by RBF feature expansion is generally helpful in improving the probability of error. It should also be noted how a larger RBF length scale l will positively affect the model by preventing overfitting at the cost of classification precision. However, this conclusion is yet to be complete as more non-linear filtering methods should be investigated and compared to the optimal performance. The dataset used in this exercise is also relatively small, so verbatim implementation for larger datasets could also be a concern subject to future explorations.