3F8: Inference
Short Lab Report

Author's Name

March 1, 2026

Abstract

This is the abstract.
Try for 1-2 sentences on each of: motive (what it's about), method (what was done), key results and conclusions (the main outcomes).

- Don't exceed 3 sentences on any one.
- Write this last, when you know what it should say!

1 Introduction

1. What is the problem and why is it interesting?
2. What novel follow-up will the rest of your report present?

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

$$\frac{\partial \mathcal{L}(\beta)}{\partial \beta} = \sum_{i=1}^{n} (y_i - \sigma(\mathbf{x}_i^\top \beta)) \tilde{\mathbf{x}}_i \,.$$

where $\sigma(\cdot)$ is the sigmoid function and $\tilde{\mathbf{x}}_i$ is the i-th row of the feature matrix $\mathbf{X}$ with an additional bias term. This allows us to update the parameters β using gradient ascent as :

$$\beta \leftarrow \beta + \eta \sum_{i=1}^{n} (y_i - \sigma(\mathbf{x}_i^\top \beta)) \tilde{\mathbf{x}}_i \,.$$

or in vectorised form as:

$$\beta \leftarrow \beta + \eta \mathbf{X}^\top (\mathbf{y} - \sigma(\mathbf{X}\beta)) \,.$$

where η is the learning rate parameter.

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

   Code:
   for t in 1 to n_steps do:
      b = b + learning_rate * X^T * (y - sigmoid(X*b))
   return b
```

The learning rate parameter η is chosen to be 0.3 as to reduce overshooting, and the number of steps of gradient ascent is set to be 1000.

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 may struggle to separate the two classes, as there is a significant overlap between the two classes in the input space. However, we can also see that there are some regions in the input space where one class is more dominant than the other, which suggests that a linear classifier may still be able to find a decision boundary that separates the two classes to some extent. Overall, while a linear classifier may not be able to perfectly separate the two classes, it may still be able to achieve reasonable performance on this dataset.

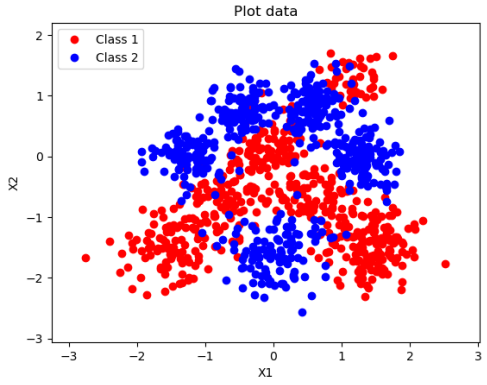

Figure 1: Visualisation of the data.

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:

```
def fit_w(X_tilde_train, y_train, X_tilde_test, y_test, n_steps, alpha):
    w = np.random.randn(X_tilde_train.shape[ 1 ]) # initialise w randomly
    ll_train = np.zeros(n_steps)
    ll_test = np.zeros(n_steps)
    for i in range(n_steps):
        sigmoid_value = predict(X_tilde_train, w)
        log_likelihood_gradient = np.dot(X_tilde_train.T, (y_train - sigmoid_value)) / X_tilde_train.sha
        w = w + alpha * log_likelihood_gradient
```

```
        ll_train[ i ] = compute_average_ll(X_tilde_train, y_train, w)
        ll_test[ i ] = compute_average_ll(X_tilde_test, y_test, w)
        print(ll_train[ i ], ll_test[ i ])

    return w, ll_train, ll_test
```

We then train the classifier using this code. We fixed the learning rate parameter to be $\eta = 0.3$ as to prevent over-shooting. 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 the likelihood plateaus very early, likely a consequence of the limitations of a linear boundary.

Figure 3 displays the visualisation of the contours of the class predictive probabilities on top of the data. We can see that the limitations of the linear classifier is evident, as we find that the boundary is clearly unable to separate the clusters of the two classes with one single boundary, resulting in a boundary with high uncertainty as seen by the spread of the contours.

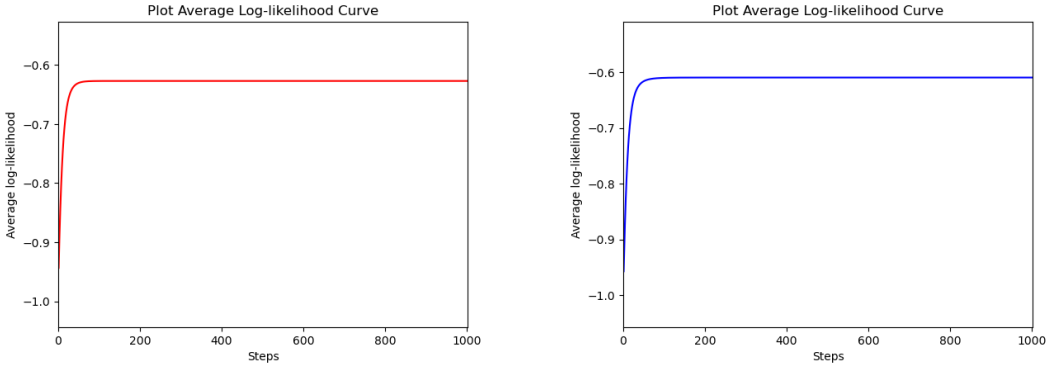

Figure 2: Learning curves showing the average log-likelihood on the training (left) and test (right) datasets.

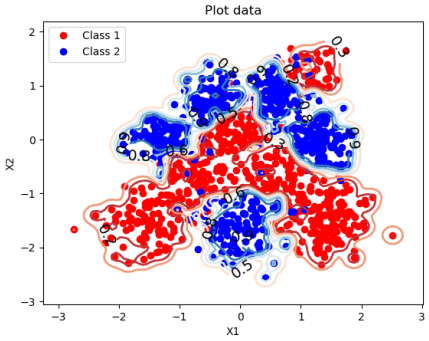

Figure 3: Visualisation of the contours of the class predictive probabilities.

6 Exercise e)

The final average training and test log-likelihoods are shown in Table 1. The 2x2 confusion matrices on the test set is shown in Table 2. By analising this table, we conclude that the model is able to correctly classify 74% of the positive class and 71% of the negative class, which indicates that the model has a decent performance on the test set. We also found that through our experiments, there are often fluctuations of

whether the performance favours class 1 or class 2, which may be due to the permutation function chosen at the beginning of the code to split the data into training and test sets. This suggests that the performance of the model may be sensitive to the specific data points included in the training and test sets, which highlights the importance of using a robust method for splitting the data and evaluating the model's performance. For the results obtained here, we found that the training set of 800 samples contained 391 samples of class 2 and 409 samples of class 1, while the test set of 200 samples contained 109 samples of class 1 and 91 samples of class 0. This balanced distribution of classes in both the training and test sets may have contributed to the balanced performance on both classes, as the model had sufficient examples of each class to learn from during training and to be evaluated on during testing.

Avg. Train ll	**Avg. Test ll**
-0.6271	-0.6093

Table 1: Average training and test log-likelihoods to 4 s.f.

		$\hat{y}$	
		0	1
y	0	0.71	0.29
	1	0.26	0.74

Table 2: Confusion matrix on the test set.

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 again to be $\eta = \{0.3, 0.3, 0.03\}$ for each $l = \{0.01, 0.1, 1\}$, with number of training steps $n = \{3000, 8000, 30000\}$, respectively. The learning rate and number of training steps are chosen ad-hoc such that the log-likelihood plateaus for each choice of l, while reducing over-shooting for larger values of l. Figure 4 displays the visualisation of the contours of the resulting class predictive probabilities on top of the data for each choice of $l = \{0.01, 0.1, 1\}$. The choice of the width l has a significant impact on the resulting decision boundary. When l is too small (e.g. $l = 0.01$), the decision boundary becomes very complex and overfits the training data, resulting in a highly non-linear boundary that may not generalise well to new data points.

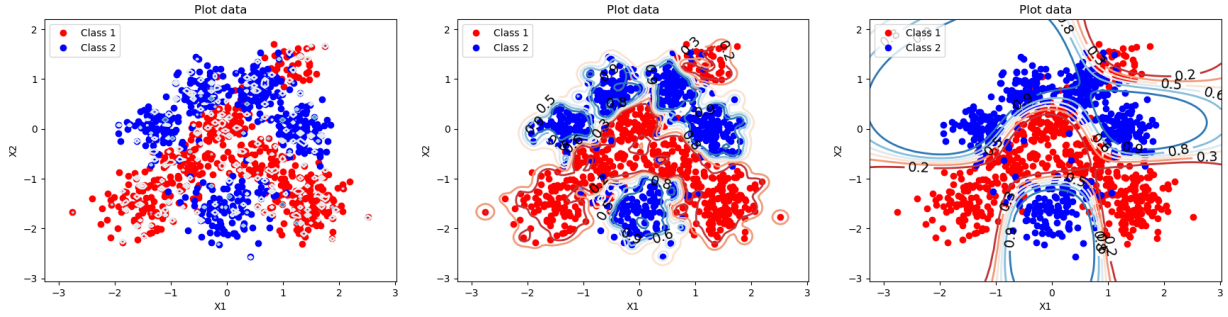

Figure 4: Visualisation of the contours of the class predictive probabilities for $l = 0.01$ (left), $l = 0.1$ (middle), $l = 1$ (right).

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. 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... When we compare these results to those obtained using the original inputs we conclude that...

		$\hat{y}$	
		0	1
y	0	0.97	0.03
	1	0.93	0.07

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

		$\hat{y}$	
		0	1
y	0	0.96	0.04
	1	0.08	0.92

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

		$\hat{y}$	
		0	1
y	0	0.94	0.06
	1	0.06	0.94

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

Avg. Train ll	**Avg. Test ll**
-0.4814	-0.6802

Table 3: Results for $l = 0.01$

Avg. Train ll	**Avg. Test ll**
-0.1419	-0.2223

Table 4: Results for $l = 0.1$

Avg. Train ll	**Avg. Test ll**
-0.2073	-0.2007

Table 5: Results for $l = 1$

9 Conclusions

1. Draw together the most important results and their consequences.
2. List any reservations or limitations.