Logistic Classification

3F8 laboratory experiment
Theo A. Brown
Selwyn College, University of Cambridge

February 17, 2022

Abstract

Enter a short summary here.

1 Introduction

Classification is the task of grouping data points according to their shared features, and assigning each group a label. This is often a trivial manual task for small datasets, but with very large collections of data it becomes very time intensive, so the development of good machine classifiers is an important area in statistical learning. Trained classifiers can be used as predictive tools to identify the most probable class that a new input will belong to, which could, for example, be useful in a medical setting, such as identifying the most at-risk patients.

A key tool in classification is the ability to classify non-linear datasets, where the boundary between classes is not a straight line. This report explores the use of a set of non-linear basis functions (in this case, radial functions), which are applied to the inputs before classifying, to allow the discovery of non-linear decision boundaries.

2 Derivation of the gradient

If each datapoint's class label is assumed to be independent and identically generated from a Bernoulli distribution in which the probability of a class label is given by a logistic function applied to weighted inputs, the log-likelihood is as follows:

$$\begin{aligned}\mathcal{L}(w) = \log P(y|X,w) &= \log \prod_{n=1}^{N} \sigma(w^T\tilde{x}^{(n)})^{y^{(n)}} \sigma(-w^T\tilde{x}^{(n)})^{1-y^{(n)}} \\ &= \sum_{n=1}^{N} y^{(n)} \log\sigma(w^T\tilde{x}^{(n)}) + (1-y^{(n)})\log\sigma(-w^T\tilde{x}^{(n)})\end{aligned}$$

The derivative of the log-likelihood is calculated using the identities $\frac{d\sigma(x)}{dx} = \sigma(x)\sigma(-x)$ and $\frac{d\log(x)}{dx} = \frac{1}{x}$. Applying the chain rule:

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

3 Gradient ascent

To find the parameter w that maximises the log-likelihood, the following gradient ascent algorithm is used:

Note that the weights do not necessarily need to be initialised as ones - they could have been selected randomly, or initialised at zero, for example. The Python 3.5+ matrix multiplication operator @ is used to vectorise the operations.

The `learning_rate` parameter is chosen empirically. Starting with a value chosen such the algorithm shows signs of convergence, the learning rate is increased until the algorithm begins to oscillate. The rate is then decreased slightly so that the oscillation is avoided. Using this method, a fast rate of convergence is ensured and divergence avoided.

4 Data visualisation

The data is visualised in the two-dimensional input space in Figure **??**. From the plot it is clear that a linear classifier will perform badly on the data, as the classes are not linearly separable: there is no straight boundary that successfully distinguishes between red and blue data points.

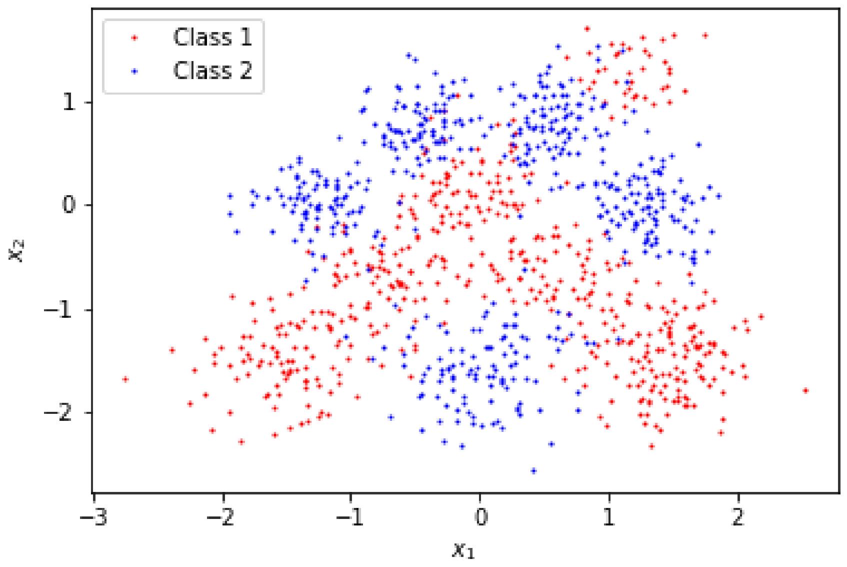

Figure 1: Plot of the two-dimensional input features (1000 datapoints, 2 classes)

5 Model training

The data was split into a training set of 800 points and a test set of 200 points, and the gradient descent algorithm with a learning rate of 0.001 was applied to find the weights for the linear classifier. The mean log-likelihood (where the mean is taken across all of the weights) is plotted in Figure **??** for the training and test datasets at each iteration.

Figure **??** shows that a plateau is reached for the log-likelihood of the parameters in both the training and test sets, after which no further improvements can be made. This suggests an optimum has been found for this model. After training there remains a difference between the performance on the test and training sets, but not at a particularly significant level (see next section).

6 Model performance

The predictive distribution of the model is shown in Figure **??**. It is clear that this model fails to accurately classify the two classes. It identifies that the general principle that a greater proportion of the points with negative x_2 values are class 1 and more points with positive x_2 values are class 2, but it is apparent to the human eye that this is overly simplified, failing to identify the islands of class 1 in class 2 and vice versa.

The mean log-likelihood of the trained model evaluated on the training and test sets is shown in Figure **??**. It shows a difference of 3-4% between the training and test data, which is sufficiently small to suggest that overfitting has been avoided. As expected, the performance is marginally better for the training set, as this was the data that the model was fit to.

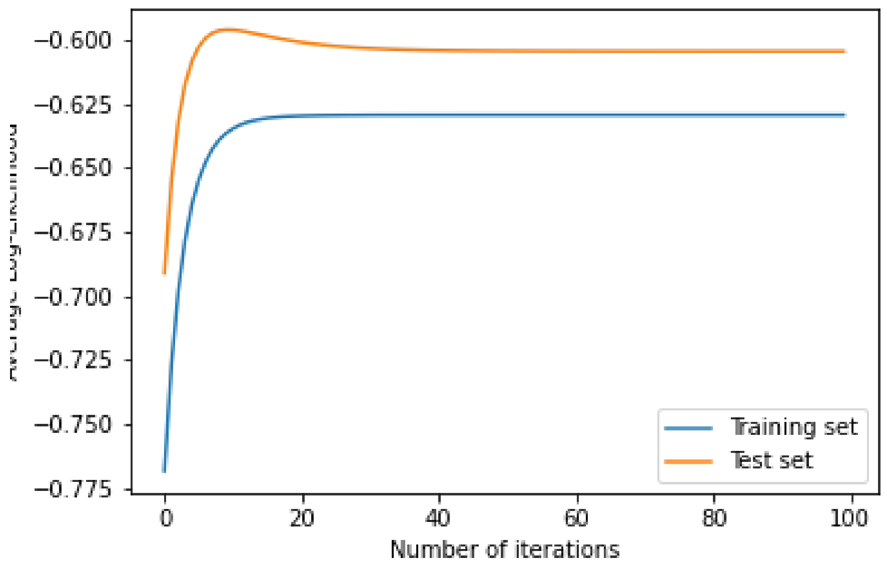

Figure 2: Plot of the mean log-likelihood of the weights for the test and training data, over the course of model training

A hard threshold is applied to the data The confusion matrix

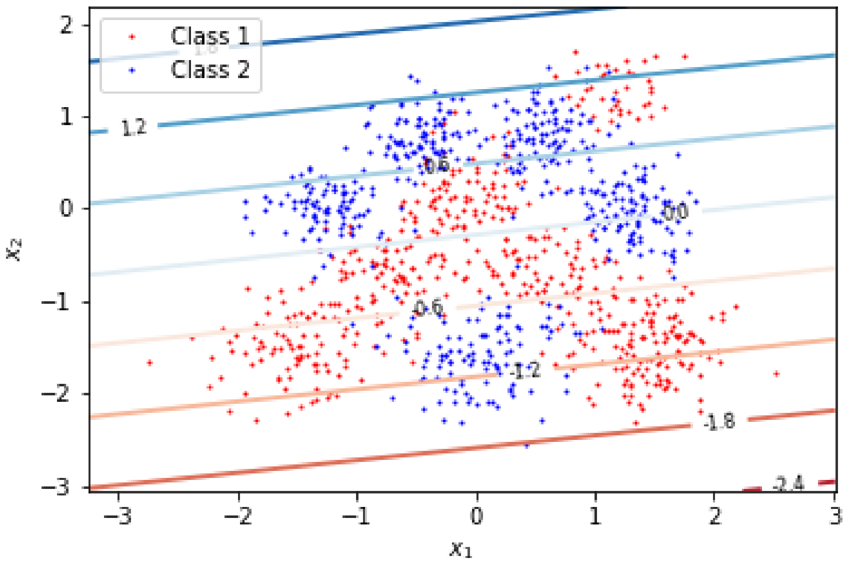

Figure 3: Model predictive distribution overlaid on input datapoints, coloured by true classification

Performance metric	**Value**
Mean log-likelihood, training data	-0.62927
Mean log-likelihood, test data	-0.60443
$\begin{pmatrix} P(\hat{y}=0\|y=0) & P(\hat{y}=1\|y=0) \\ P(\hat{y}=0\|y=1) & P(\hat{y}=1\|y=1) \end{pmatrix}$	$\begin{pmatrix} 0.723 & 0.277 \\ 0.303 & 0.697 \end{pmatrix}$