COMP6036: Advanced Machine Learning
Feature Selection Challenge

Ashley J. Robinson
ajr2g10@ecs.soton.ac.uk

School of Electronics and Computer Science
University of Southampton

22nd April, 2014

Abstract

1 Introduction

Each dataset has been divided in three section for training, validation and test submission. The final partition labels remain hidden from the development process. The discrete class labels, t, and the classifier outputs, y, are described by equation (1). Keeping to these output values requires using the thresholding function Θ in equation (2). Everything implemented has been developed from scratch for the MATLAB programming language.

$$y, t \in \{-1, +1\} \tag{1}$$

$$\Theta(a) = \begin{cases} +1, & a > 0 \\ -1, & a \leq 0 \end{cases} \tag{2}$$

2 Classifier Architectures

Artificial Neural Networks (ANNs) have been chosen as the classifier architectures. ANNs are efficient learning machines with biological inspiration (Bishop, 2006). They are parametric models which can be adapted during supervised training and are scalable in complexity. The error space can be considered as a function of parameters and the training data, $E(w|D)$, therefore many different optimisation algorithms can be used can used to train the model.

2.1 Perceptron

Figure 1 holds the most simple manifestation of an ANN called a perceptron. Each feature of an input pattern is weighted and then summed, along with a bias term, to produce an output. This output passes through a step activation function for binary classification. Capable of producing a linear separation boundary subject to equation (9) the method serves as starting point for further investigation into the data. The discrete error from from the training patterns is calculated using equation (10). Equation (11) expands the method by considering many different training patterns applied to only one set of weights. The weight vector is trained using the iterative learning rule held in equations (12), (13) and (14) (MacKay, 2005). In practice the offset parameter, b, is added to the weight vector and every input pattern is appended with a constant unit feature to allow complete training.

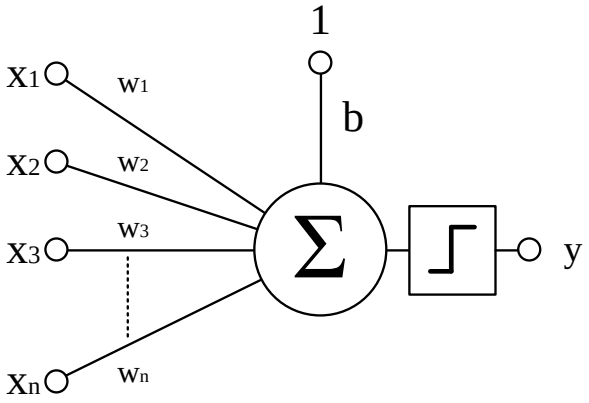

FIGURE 1: Perceptron architecture.

$$y = \Theta\left(b + \sum_{i=1}^{n} w_i x_i\right) \tag{3}$$

$$e = t - y, \;\; e \in \{-2, 0. + 2\} \tag{4}$$

$$\mathbf{Y} = \Theta(\mathbf{X}.\mathbf{W} + \mathbf{b}) \tag{5}$$

$$\mathbf{W}_{t+1} = \mathbf{W}_t + \Delta\mathbf{W} \tag{6}$$

$$\mathbf{E} = (\mathbf{T} - \mathbf{Y})^T \tag{7}$$

$$\Delta\mathbf{W} = \eta * \mathbf{E}\mathbf{X}^T \tag{8}$$

2.2 Multi Layer Perceptron

Section 2.1 can be expanded by concatenating the architecture to produce a multi layer perceptron. They contain hidden layers and when used with non-linear activation functions can produced a non-linear separation boundary (Prügel-Bennett, 2001). Only one hidden layer is implemented for simplicity as shown in Figure 2.

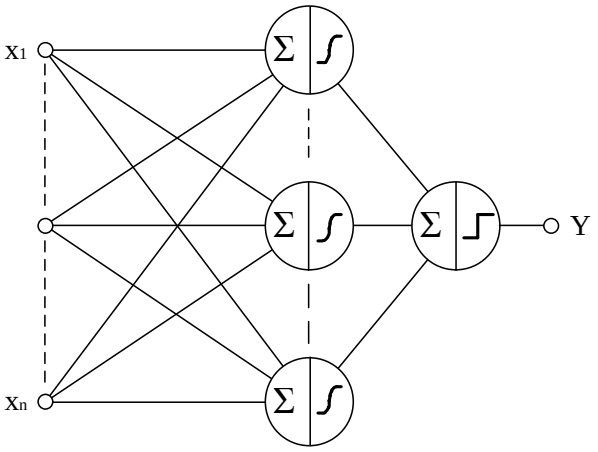

FIGURE 2: Multi layer perceptron architecture.

$$y = \Theta\left(b + \sum_{i=1}^{n} w_i x_i\right) \tag{9}$$

$$e = t - y, \;\; e \in \{-2, 0. + 2\} \tag{10}$$

$$\mathbf{Y} = \Theta(\mathbf{X}.\mathbf{W} + \mathbf{b}) \tag{11}$$

$$\mathbf{W}_{t+1} = \mathbf{W}_t + \Delta\mathbf{W} \tag{12}$$

$$\mathbf{E} = (\mathbf{T} - \mathbf{Y})^T \tag{13}$$

$$\Delta\mathbf{W} = \eta * \mathbf{E}\mathbf{X}^T \tag{14}$$

3 Feature Selection

4 Results

5 Conclusions

References

C.M. Bishop. *Pattern Recognition and Machine Learning*. Information Science and Statistics. Springer, 1 edition, 2006.

David J.C. MacKay. *Information Theory, Inference and Learning Algorithms*. Cambridge University Press, 4 edition, 2005.

Adam Prügel-Bennett. *COMP3008: Machine Learning*. University of Southampton, 2001. Lecture Notes.

A Code Listings