UNIT 4 U4/0/2024

Machine Learning

COS4852

Year module

Department of Computer Science

School of Computing

CONTENTS

This document contains the material for UNIT 4 for COS4852 for 2024.

1 OUTCOMES

In this Unit you will learn more about the theoretical basis of neural networks, and understand how to apply some of the algorithms used to train a nerual network on a dataset. You will learn to describe and solve a learning problem using artificial neural networks. You will learn about the following structures and algorithms:

1. The threshold logic unit (TLU), often called the Perceptron.
2. Multi-layer networks.
3. The BACKPROPAGATION algorithm used to train multi-layer networks.

After completion of this Unit you will be able to:

1. Understand and recognise appropriate learning problems that can be solved using various artificial neural networks.
2. Understand and describe the Perceptron training rule.
3. Understand and describe the representational power and limitations of Perceptrons.
4. Construct a Perceptron for a given learning task.
5. Train and analyse the Perceptron, including showing a graphical representation of the resulting decision surface.
6. Solve a given learning problem using the gradient descent algorithm.
7. Derive the BACKPROPAGATION algorithm.
8. Solve a given learning problem using the BACKPROPAGATION algorithm.
9. Understand and describe theoretical issues in artificial neural networks, including hidden layer representations, over-fitting of data, generalization and stopping criteria.

2 INTRODUCTION

In this Unit you will study the theory of neural networks and learn to implement some of the algorithms used to train a neural network on a data set.

Neural Networks mimic the structure of biological neurons (those found in animal brains), but means of a mathematical simplification of the how neurons are activated, how signals are transferred between neurons in a network of connected neurons, and their learning process where the strength of connections between neurons are adapted.

These sites give a good overview of neural networks and their history, including the Perceptron:

- IBM site
- EDUCBA site
- BMC blog

3 PREPARATION

3.1 Online textbooks

Chapter 4 in Nilsson's book looks at neural networks, by starting with Threshold Logic Units (TLUs - called Perceptron's elsewhere) and progresses to the BACKPROPAGATION algorithm.

Raúl Rojas' book Neural Networks, provides a detailed exposition of the classic algorithms in neural networks. Chapter 2 discusses the TLU, by working through the idea that individual neurons can be implemented using traditional Boolean logic. No training is required, as the networks are designed to implements specific Boolean functions. Chapter 3 and 4 extends the idea to Perceptrons, which are essentially TLUs with weighted connections, which then allows us to train such a network on data, without explicitly having to design the logic. Chapter 6 introduces the idea of multi-layer networks and the complexity of the functions that can be approximated with such a network. Chapter 7 works through the BACKPROPAGATION algorithm, which is the classic training algorithm for a multi-layer neural network. Chapter 8 addresses the shortcomings of the BACKPROPAGATION algorithm and techniques to address these. The rest of the book gets into more complex and esoteric material, and will not be studied in this unit.

Matt Nielsen has an online book on Deep Learning with some very good material on the BACK-PROPAGATION algorithm.

3.2 Textbooks

Chapter 4 of Mitchell's book provides another detailed discussion on neural networks. Of significance here is Section 4.6.4 that looks at the represenations of the hidden layer, and Section 4.6.5 on generalisation, overfitting and when to stop the training.

3.3 Online material

This YouTube video by the famous Geoffrey Hinton himself gives a very thorough, slow, step-by-step explanation of the BACKPROPAGATION algorithm.

The `brilliant.org` website on BACKPROPAGATION gives a detailed discussion of the BACKPROP-AGATION algorithm used to train feedforward networks using the gradient descent method. This article by David Rumelhart in the CACM gives a brief overview of the BACKPROPAGATION algorithm.

The original articles are in the form of a comprehensive technical report and a more condensed version that appeared in Nature.

Here is a brief summary of the BACKPROPAGATION algorithm.

4 DISCUSSION

4.1 Single Perceptron classification

In Section 4.1 of Nilsson's there is a discussion on hyperplanes and their polarity and the concept of the neural network weight-space. The terms TLU and Perceptron refer to the same construct here (i.e. a neuron with weights and a threshold activation function). Similarly, the terms hyperplane, decision boundary, and decision surface refer to the same concept, which linearly divides the Perceptron input space into two sub-spaces. In a 2-dimensional space the hyperplane is a straight line.

Consider the case shown in Figure 1. The figure shows the instance space of a Perceptron (a single

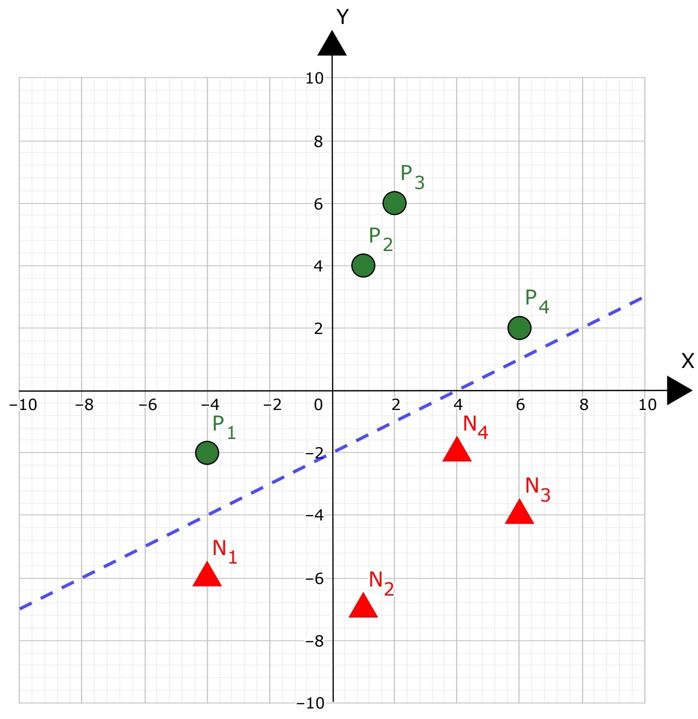


Figure 1: Instances and the hyperplane of a Perceptron.

neuron) with its decision surface/hyperplane. Positive instances are marked as P_i and negative instance as N_i:

$$\begin{aligned} P_1 &= (\text{-}4, \text{-}2) \\ P_2 &= (1, 4) \\ P_3 &= (2, 6) \\ P_4 &= (6, 2) \\ N_1 &= (\text{-}4, \text{-}6) \\ N_2 &= (1, \text{-}7) \\ N_3 &= (6, \text{-}4) \\ N_4 &= (4, \text{-}2) \end{aligned}$$

Let's calculate the values of the weights of the Perceptron, using the position of the decision surface/hyperplane. The cut-off points of the hyperplane on the axes are at $(4, 0)$ and $(0, -2)$.

The decision surface of a Perceptron with two inputs, x and y must satisfy the equation:

$$w_0 + w_1x + w_2y = 0$$

Two known points on the line of the decision surface are the intersection points with the axes, $(4, 0)$ and $(0, -2)$. Inserting these values into the equation for the Perceptron decision surface gives:

$$w_0 + 4w_1 = 0$$

$$w_0 - 2w_2 = 0$$

Which gives:

$$w_0 = \text{-}4w_1$$

$$w_0 = 2w_2$$

Any values of w_0 , w_1 and w_2 that satisfy these ratios will suffice. Choose $w_0 = -4$, to give $w_1 = 1$. Solve $w_0 = 2w_2$ to get $w_2 = \text{-}2$. To summarise:

$$\begin{aligned} w_0 &= \text{-}4 \\ w_1 &= 1 \\ w_2 &= \text{-}2 \end{aligned}$$

The decision surface has a polarity, as shown in Figure 2. In other words it has a positive and negative side, classifying all instances on one side as positive and those on the other side as negative. This is due to the threshold activation function used by the Perceptron.

There are an infinite set of values for w_i that will describe a line that corresponds to the decision surface, but only half of these will classify the data correctly. To test whether the calculated weight values correctly classifies the data substitute the (x_1, x_2) values of any example into the equation of the decision surface, using the calculated weights. Choose the positive instance $P_1 = (\text{-}4, \text{-}2)$:

$$\begin{aligned} & w_0 + w_1x + w_2y \\ = \; & \text{-}4 + (1)(\text{-}4) + (\text{-}2)(\text{-}2) \\ = \; & \text{-}4 \\ < \; & 0 \end{aligned}$$

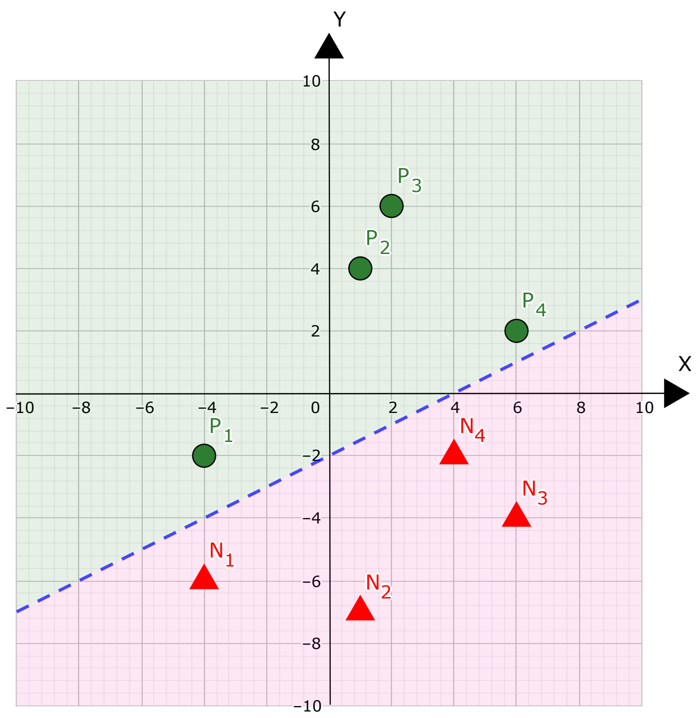


Figure 2: Case (a): polarity of the hyperplane of a Perceptron.

The calculated weights classifies the positive instance at P_1 as negative. This means that the weight values are therefore incorrect.

Change the sign of initial weights, to become:

$$\begin{aligned} w_0 &= 4 \\ w_1 &= \text{-}1 \\ w_2 &= \text{-}2 \end{aligned}$$

which gives another valid equation for the line, but classifies the data point correctly. Test P_1 again:

$$\begin{aligned} & w_0 + w_1 x + w_2 y \\ = & \text{-}4 + (\text{-}1)(\text{-}4) + (\text{-}2)(\text{-}2) \\ = & 2 \\ > & 0 \end{aligned}$$

thereby correctly classifying the positive instance P_1 as positive. These weights are therefore correct. The resulting correct polarity is shown in Figure 2. All the instances should be checked to make sure that there is not some other calculation error, but is left as an exercise to the reader (you).

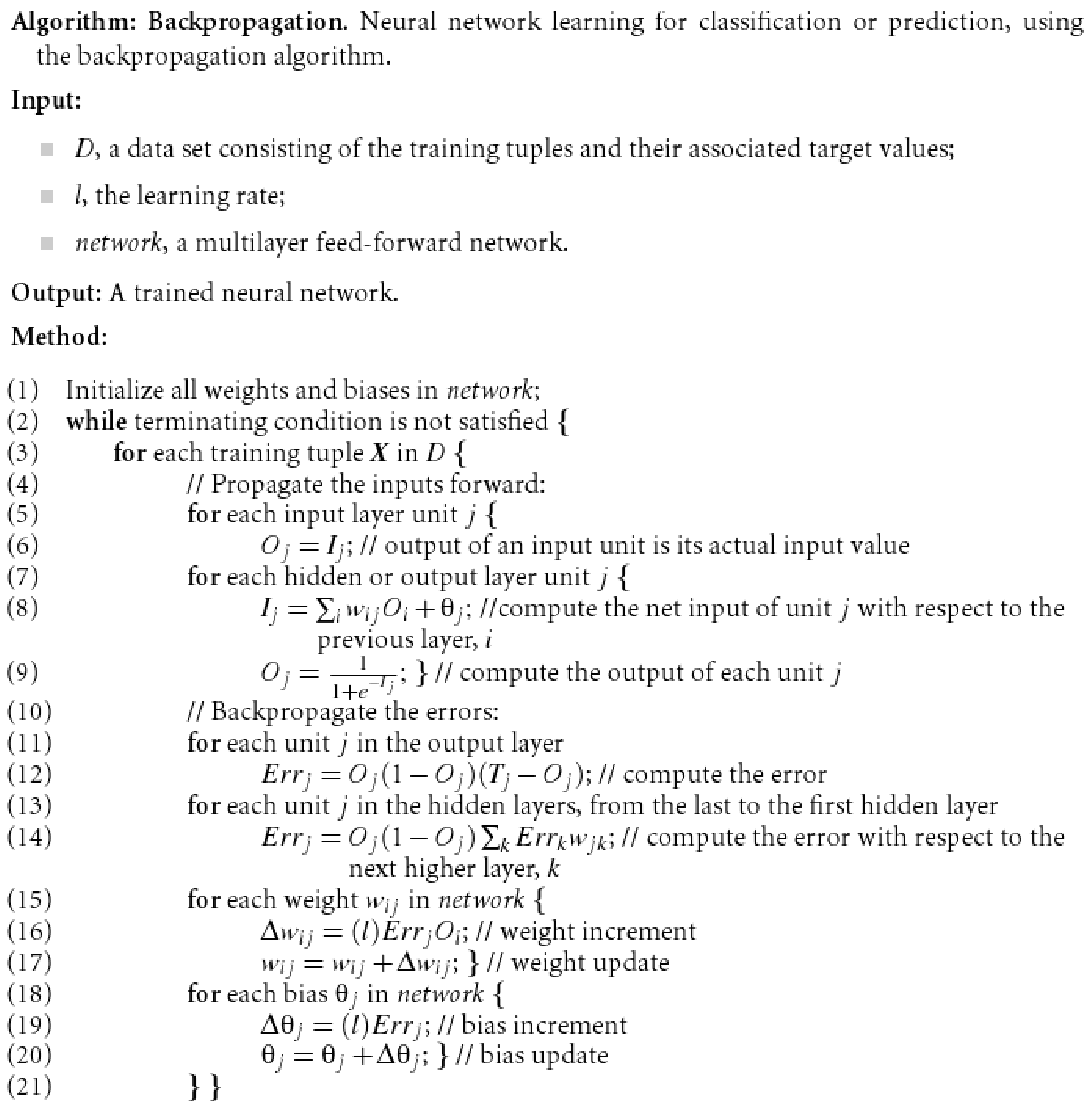
4.2 GradientDescent

The GradientDescent technique is an algorithm that attempts to find a (usually) local minimum (or maximum) of a given function. This is a staple of numerical mathematics and forms the basis of many machine learning algorithms. The premise is relatively easy to understand - think of a water drop running down the surface of an object, for it to end up in a depression in the surface. This animation on Wikipedia shows this in action. The algorithm does this by taking small steps in the direction of the steepest downward gradient up to where it reaches a point where there are no further downward gradients.

This article on the Towards Data Science site gives a thorough discussion of the basic algorithm with two worked examples.

4.3 BACKPROPAGATION algorithm

Figure 3 summarises the BACKPROPAGATION algorithm.



Algorithm: Backpropagation. Neural network learning for classification or prediction, using the backpropagation algorithm.

Input:

- D, a data set consisting of the training tuples and their associated target values;
- l, the learning rate;
- *network*, a multilayer feed-forward network.

Output: A trained neural network.

Method:

```
(1)  Initialize all weights and biases in network;
(2)  while terminating condition is not satisfied {
(3)      for each training tuple $X$ in $D$ {
(4)          // Propagate the inputs forward:
(5)          for each input layer unit $j$ {
(6)              $O_j = I_j$; // output of an input unit is its actual input value
(7)          for each hidden or output layer unit $j$ {
(8)              $I_j = \sum_i w_{ij} O_i + \theta_j$; //compute the net input of unit $j$ with respect to the
                     previous layer, $i$
(9)              $O_j = \frac{1}{1+e^{-I_j}}$; } // compute the output of each unit $j$
(10)         // Backpropagate the errors:
(11)         for each unit $j$ in the output layer
(12)             $Err_j = O_j(1-O_j)(T_j - O_j)$; // compute the error
(13)         for each unit $j$ in the hidden layers, from the last to the first hidden layer
(14)             $Err_j = O_j(1-O_j)\sum_k Err_k w_{jk}$; // compute the error with respect to the
                         next higher layer, $k$
(15)         for each weight $w_{ij}$ in network {
(16)             $\Delta w_{ij} = (l) Err_j O_i$; // weight increment
(17)             $w_{ij} = w_{ij} + \Delta w_{ij}$; } // weight update
(18)         for each bias $\theta_j$ in network {
(19)             $\Delta \theta_j = (l) Err_j$; // bias increment
(20)             $\theta_j = \theta_j + \Delta \theta_j$; } // bias update
(21)         } }
```

Figure 3: BACKPROPAGATION algorithm

This `brilliant.org` site gives a thorough discussion on the derivation of the gradients for the gradient descent training process used by BACKPROPAGATION. It also has a very good example of an implementation of the BACKPROPAGATION algorithm in pure `Python` (no ML libraries that hide the detail).

Michael Nielsen has an online book on Deep Learning. Chapter 2 of his book goes into detail about the BACKPROPAGATION algorithm (the backbone behind many Deep Learning algorithms).

Matt Mazur does a very thorough step-by-step explanation of the BACKPROPAGATION algorithm, although he does not train the bias weights. The bias weights are just as important as the other weights, and the training process is exactly the same as for the other weights.

5 ACTIVITIES

5.1 TASK 1 - COMPLETE PERCEPTRON CALCULATIONS

Complete the calculations for the other instances of the example Perceptron decision boundary shown earlier.

Re-do the Perceptron calculations for sets of 2D data that you create yourself (below are examples of such data sets that you can use). Make careful observations about the linear separability of the data and the effect of that on the decision boundary. You will have to find a suitable decision boundary yourself. Think how to solve the problem using more than one Perceptron.

Data set 1

$$\begin{aligned}
P_1 &= (6, -1) \\
P_2 &= (1, 4) \\
P_3 &= (2, 6) \\
P_4 &= (6, 2) \\
N_1 &= (-2, -1) \\
N_2 &= (-1, -5) \\
N_3 &= (3, -3) \\
N_4 &= (-2, 3)
\end{aligned}$$

Data set 2

$$\begin{aligned}
P_1 &= (6, \text{-}1) \\
P_2 &= (1, 3) \\
P_3 &= (2, 6) \\
P_4 &= (6, 2) \\
N_1 &= (\text{-}2, \text{-}1) \\
N_2 &= (\text{-}1, \text{-}5) \\
N_3 &= (3, \text{-}3) \\
N_4 &= (\text{-}1, 5)
\end{aligned}$$

Data set 3

$$\begin{aligned}
P_1 &= (6, \text{-}1) \\
P_2 &= (1, 1) \\
P_3 &= (2, 6) \\
P_4 &= (6, 2) \\
N_1 &= (\text{-}2, \text{-}1) \\
N_2 &= (\text{-}1, \text{-}5) \\
N_3 &= (3, \text{-}1) \\
N_4 &= (0, \text{-}3)
\end{aligned}$$

5.2 TASK 2 - STUDY THE MATERIAL

Find and read all the online material shown earlier in this document. Study the relevant concepts carefully and thoroughly.

As a first step watch the video by Geoffrey Hinton. This should give you a very good understanding of the BACKPROPAGATION algorithm. Once you've done that the other material should not be a challenge, and you will have documents that you study that contains the details.

5.3 TASK 3 - ADVANCED NEURAL NETWORKS

Read this *Nature* review paper on Deep Learning. This paper is from 2015, and much has happened in this field since, but it still gives a good overview of the basic kinds of Deep Learning models that have been developed.

If you want to learn more about Neural Networks you can watch the entire series of videos by Geoffrey Hinton, which takes this much further than this Unit.

5.4 TASK 4 - THEORETICAL ISSUES IN NEURAL NETWORKS

Study the material provided above (Hinton's videos are particularly useful - the slides used in the videos are also available), with specific reference to:

- Stopping criteria for BACKPROPAGATION training.
- The effects of *momentum* and *gain* parameters.
- The problem of *overfitting*, how to deal with it by splitting the data-set into a training and a test set vs. a training, test, and validation sets.
- Find the well-known 8×3×8 binary encoding problem and study what happens in the hidden layer. Mitchell's book is one source.
- How autoencoders work. Here is one paper on the topic.