Aleksander Skraastad

Assignment 3

TDT4137

Neural Networks

Autumn 2015

Norwegian University of Science and Technology

Table of contents

Chapter 1

Tasks

1.1 Task A

See attachment python file.

1.2 Task B

```
[71] --- Training AND ----------------------
Weight vector changed: array([-0.09610704,  0.08977465])
Weight vector changed: array([ 0.10389296,  0.28977465])
Weight vector changed: array([ 0.30389296,  0.48977465])
Weight vector changed: array([ 0.30389296,  0.28977465])
Training complete in 5 epochs
[71] --- Training OR ----------------------
Weight vector changed: array([-0.45409499,  0.02831457])
Weight vector changed: array([-0.25409499,  0.02831457])
Weight vector changed: array([-0.05409499,  0.22831457])
Weight vector changed: array([ 0.14590501,  0.22831457])
Training complete in 3 epochs
[71] --- Testing AND ----------------------
Success! on array([0, 0]) with output 0
Success! on array([0, 1]) with output 0
```

```
Success! on array([1, 0]) with output 0
Success! on array([1, 1]) with output 1
[71] --- Testing OR ----------------------
Success! on array([0, 0]) with output 0
Success! on array([0, 1]) with output 1
Success! on array([1, 0]) with output 1
Success! on array([1, 1]) with output 1
```

For these two runs, the weights move in the same direction (towards positive inf) even thought they have different random values at initialization.

Reducing the threshold below zero on the OR training will shift the decision boundary outside our expected range, thus making the perceptron fail to converge.

Without an additional bias vector, we need to shift the threshold upwards of 0.5 for the AND train to guarantee that it will converge. This can be altered if we introduce a bias vector and set it to 1. Then we can use the threshold as a confidence level.

1.3 Task C

It is kind of hard to state an exact amount that gives good results, as they vary somewhat between runs. I have managed to get good results even on 1 hidden layer, but the fluctuations are much greater than when using 8.

I would say that I get somewhat consistenly good results by using more than 2 hidden layers.

The hidden layer has managed at this level to create an internal representation of a transformation function that cumulatively sums up to the same value as the input. By the look of it the network is a scaling function that scales all input down to the range from negative min to positive max of the trained input integers.

The network does not cope well with odd input types, as they are all scaled to either min or max of the trained input values give or take. Either positive or negative.