

Notes on Simple Nets

1 Learning a quadratic function

1.1 Introduction

We trained a simple neural net to learn the function $f(x) = x^2$ and explore aspects of the landscape of the related loss function. The network we trained, depicted in Fig.1, has a single input neuron, a single hidden layer with two neurons and a single output neuron. The network we trained uses the sigmoid function as its activation function and so the output of the network is a number in the interval $[0, 1]$. For this reason, we focus on learning the function $f(x) = x^2$ over the domain $[-1, 1]$ so that the range of f matches the range of our network.

To train the network, we generated an array of random 10000 points uniformly distributed in the interval $[0, 1]$ and then computed the value of $f(x)$ for each random point. This gave us a data set to train our network on. A second data set was generated, in the same way, to be used for testing how well the network generalises. The network was trained using the stochastic gradient descent method for 50 epochs with a mini-batch size of 50. The evolution of the network's output as a function of x is shown in Fig.2

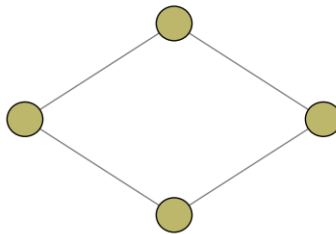


Figure 1

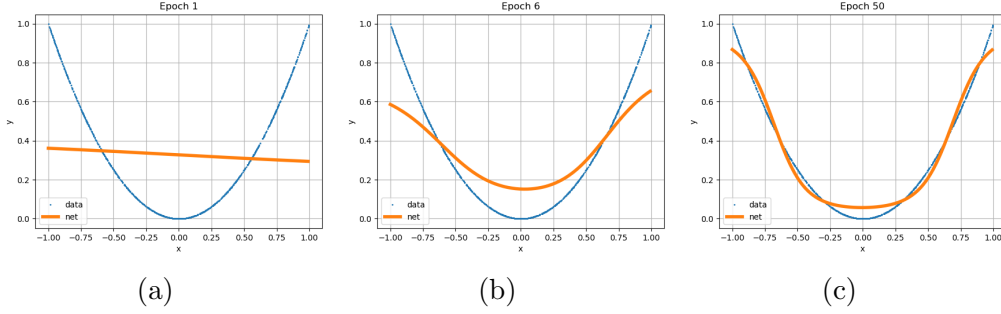


Figure 2

Once the network had been trained, we then looked at the performance of the network. The loss function used to evaluate the performance of the net was the squared Euclidean distance between the output of the net and the target output averaged over the test data.

The change in loss due to a variation in each of the networks parameters was calculated. We number the parameters of the network 0 to 6. The weights connecting the first two layers are numbers 0 and 1 while the weights connecting the second and third layers are numbered 2 and 3. The biases of the hidden layers are numbered 4 and 5 while the bias for the output neuron is given the number 6. Given the trained weights and biases $\{\omega_i\}$, we vary a particular parameter by adding a constant $\delta\omega_i$ ranging from -10 to 10 while holding all other parameters fixed. The results are shown in Fig.3. We see each curve having a local minimum at $\delta\omega_i = 0$ indicating the network found a local minimum of the loss function while being trained.

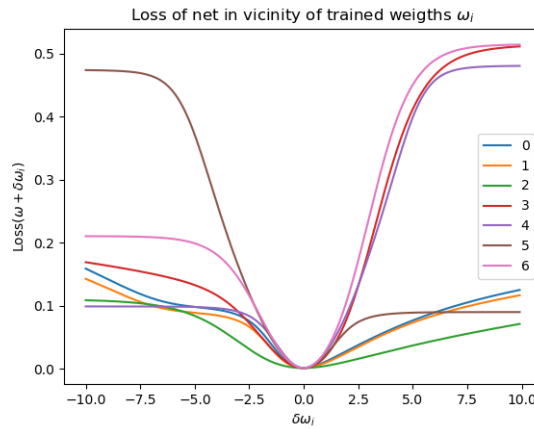


Figure 3

$$\begin{array}{ccccccccccccccccccc}
& & \supset & \searrow & & \downarrow & & & \supset & \downarrow & & & \supset & \downarrow & & & & \\
A_0 & \xrightarrow{\omega} & A_1 & \xrightarrow{\sigma} & A_1 & \xrightarrow{\omega} & A_2 & \xrightarrow{\sigma} & A_2 & \cdots & \xrightarrow{\omega} & A_n & \xrightarrow{\sigma} & A_n & & & & \\
x & \rightarrow & z_1 & \rightarrow & a_1 & \rightarrow & z_2 & \rightarrow & a_2 & \cdots & \rightarrow & z_n & \rightarrow & y & & & &
\end{array} \quad (1)$$