Assignment 3, COMP4702

Roy Portas, 43560846

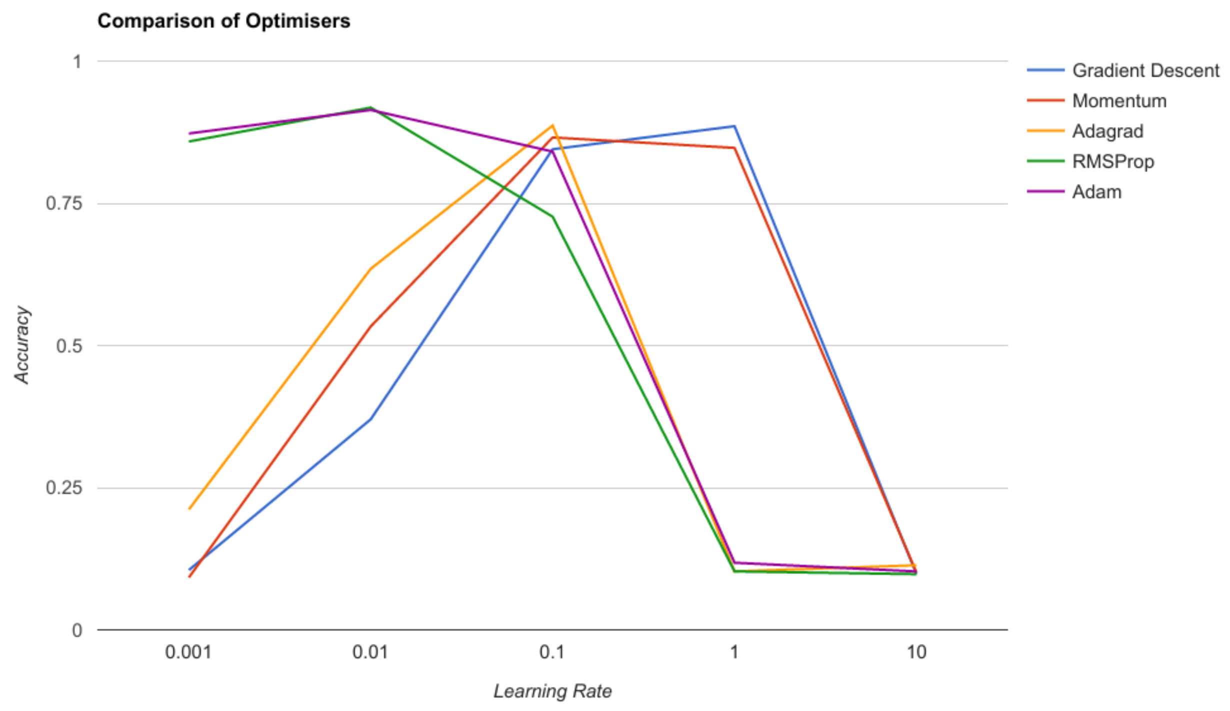
May 20, 2017

Prac 7

Q 7.1

Part A

The network topology that was used is Prac 6 is a 60 unit neural network with one hidden layer. Five different methods were compared, with learning rates ranging from 0.001 to 10. For this test 100 was chosen as the maximum steps, this is relatively small but increasing the maximum steps should only make the model more accurate.



The above figure shows a comparison of different methods with different learning rates. It can be seen that RMSProp and Adam run much better with smaller learning rates, with Adam slightly outperforming RMSProp.

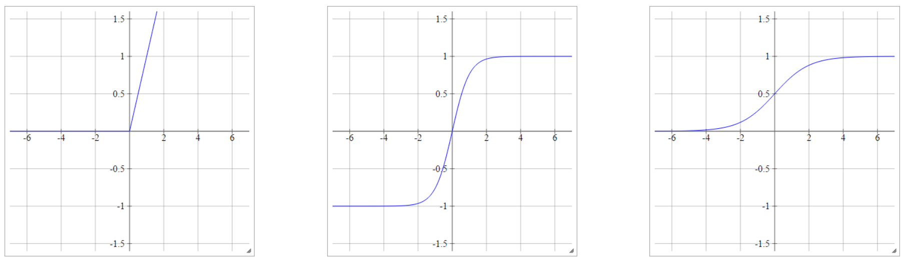
Gradient Descent and Momentum work better with higher learning rates, peaking with a learning rate of 1, with Gradient Descent performing the best of the two.

Adagrad performs the best with a learning rate of 0.1, however does not perform well with higher or lower values, whereas all the other methods work well for multiple learning rates.

All the algorithms start performing poorly when the learning rate increases past 1. The best performing algorithm was RMSProp with a learning rate of 0.01, with an accuracy of 0.9182.

Part B

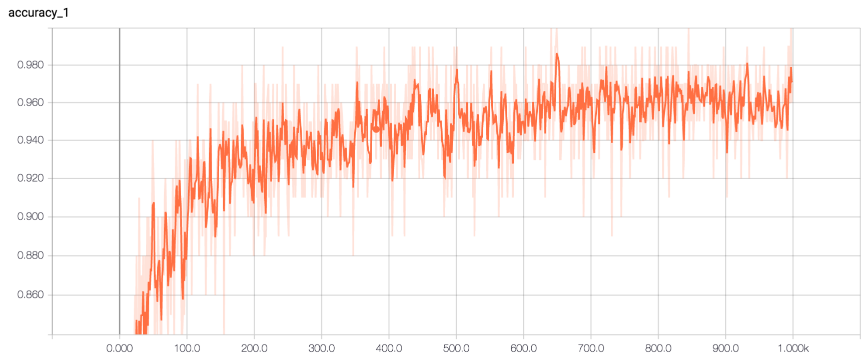
The Adam method was chosen for question with a learning rate of 0.01, beta1 of 0.9 and beta2 of 0.8. The choices for the activation functions are Relu, Tanh and Sigmoid, which are shown in the figure below.



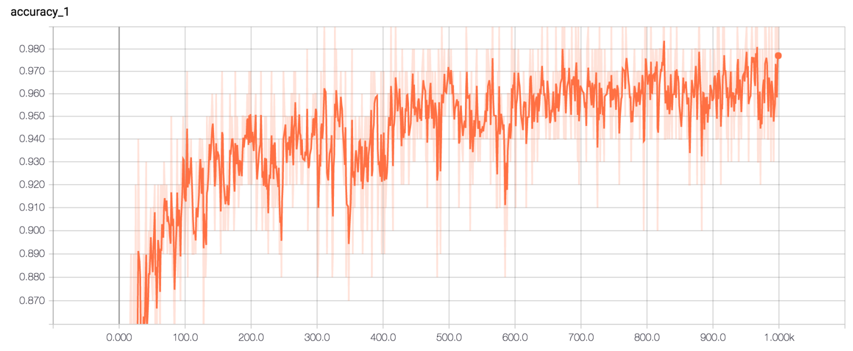
(a) Relu (b) Tanh (c) Sigmoid

Figure 1: Activation Functions

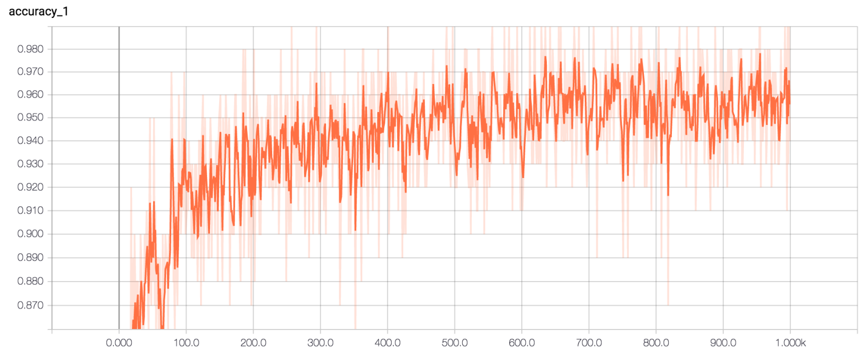
A few things can be attained from the above diagrams. The Relu function has a y range from 0 to infinity, whereas tanh is bounded between -1 and 1, and sigmoid is bounded between 0 and 1. This means that Relu won't be affected by the vanishing gradient problem, this is because the y range for Relu goes to Infinity, whereas the others converge. Having a non infinite range is better for gradient based methods, as it tends to be more stable.



(a) Relu



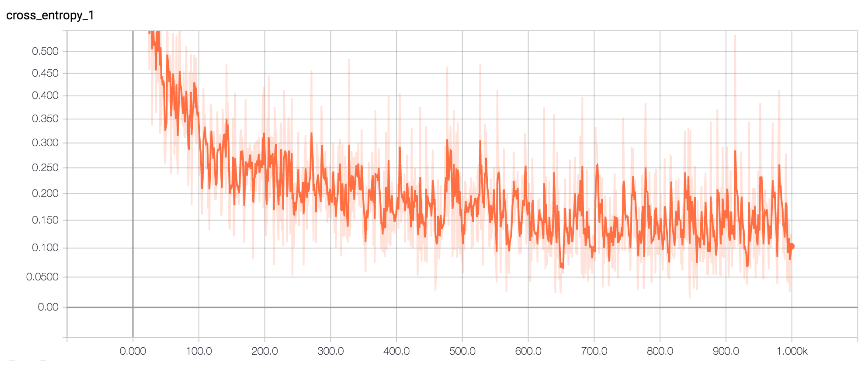
(b) Tanh



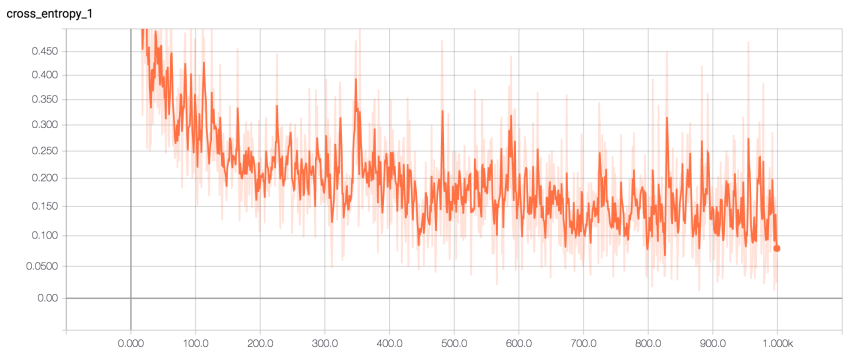
(c) Sigmoid

Figure 2: Activation Function Accuracy

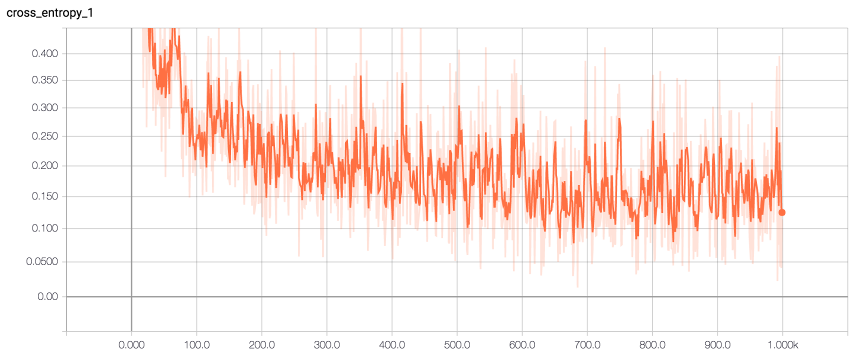
From comparing the three graphs of accuracy, it can be seen that Relu seems to be the most stable, whereas both Tanh and Sigmoid has a lot more variation.



(a) Relu



(b) Tanh

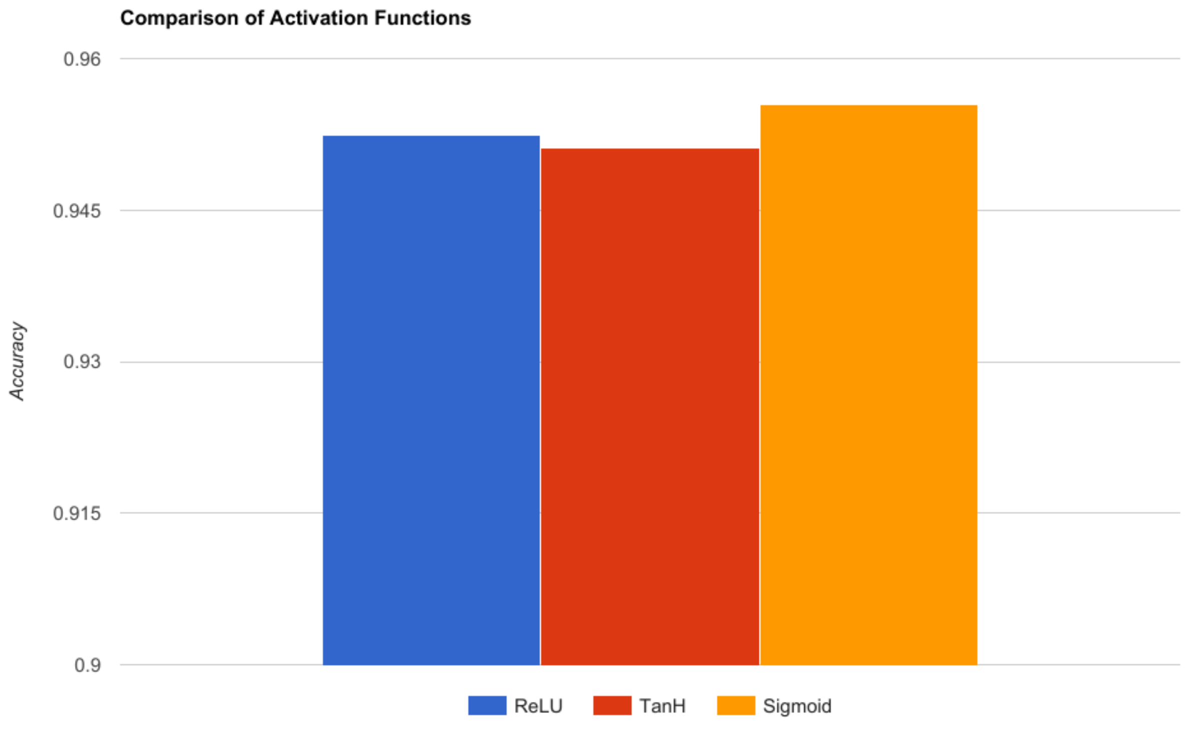


(c) Sigmoid

Figure 3: Activation Function Cross Entropy

The cross entropy graphs show the same relationship, with Relu having the least variation per step.

The Adam method with 1000 steps was tested using each activation function, which produced the following graph.



From the above graph it can be seen that changing the activation function does not make a significant difference to the accuracy, but the sigmoid function had the best accuracy from the three.

Q 7.2

Volume of Weight Matrices

The given convolutional network has two conv-pool layers followed by two fully connected layers. The MNIST dataset is being used, so the input data will be greyscale images of size 28 pixels squared, which can be represented as $\left[28 \times 28 \times 1\right]$

The first conv layer will have 32 filters, so have a size of $\left[28 \times 28 \times 32\right]$. This is followed by a Relu operation but the layer size will remain the same.

The pool layer will decrease the size of the layer as it is using downsampling, this can be calculated using the following formula, where W is the input volume size, F is the spacial extent and S is the stride.

$$\begin{aligned} size &= \frac{W-F}{S}+1 \\ &= \frac{28-2}{1}+1 \\ &= 27 \end{aligned}$$

Thus the size of the first pool layer is $\left[27 \times 27 \times 32\right]$.

The second conv layer will have 64 filters, so a size of $\left[27 \times 27 \times 64\right]$.

Using the equation from before the second pool layer will have a size of $\left[26 \times 26 \times 64\right]$.

The first fully connected layer will have a size of $\left[1 \times 1 \times 1024\right]$.

The second fully connected layer will have a size of $\left[1 \times 1 \times 10\right]$.

Number of Parameters

Equation for input layer:

The input layer has no bias or weights

$$params = 0$$

Equation for calculating the parameters of the first conv layer:

$$\begin{aligned} params &= unique\ weights \times filter\ size \times filter\ size \times input\ depth + bias \\ &= 32 \times 3 \times 3 \times 1 + 32 \\ &= 320 \end{aligned}$$

Equation for first pooling layer:

The pooling layers have no bias or weights

$$params = 0$$

Equation for calculating the parameters of the second conv layer:

$$\begin{aligned}
params &= unique\ weights \times filter\ size \times filter\ size \times input\ depth + bias \\
&= 64 \times 3 \times 3 \times 32 + 64 \\
&= 18,496
\end{aligned}$$

Equation for second pooling layer:

The pooling layers have no bias or weights

$$params = 0$$

Equation for the first fully connected layer:

$$\begin{aligned}
params &= input\ height \times input\ width \times input\ depth \times depth + bias \\
&= 26 \times 26 \times 64 \times 1024 + 1024 \\
&= 44,303,360
\end{aligned}$$

Equation for the second fully connected layer:

$$\begin{aligned}
params &= input\ height \times input\ width \times input\ depth \times depth + bias \\
&= 1 \times 1 \times 1024 \times 10 + 10 \\
&= 10,250
\end{aligned}$$

The total number of parameters is the sum of all the individual parameters

$$\begin{aligned}
total &= \sum params \\
&= 0 + 320 + 0 + 18,496 + 0 + 44,303,360 + 10,250 \\
&= 44,332,426
\end{aligned}$$

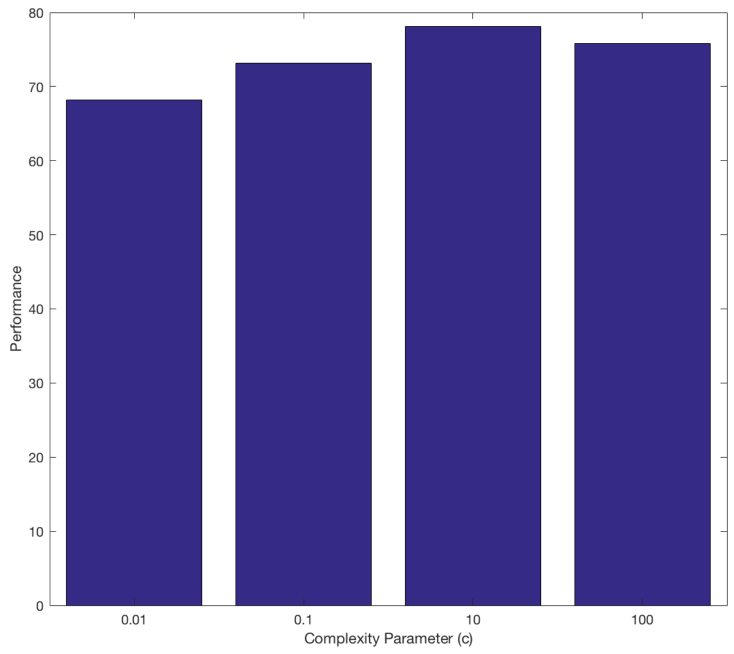
Q 7.4

Prac 8

Q 8.3

The best result from the Q2 was found to be a polynomial kernel function with epsilon parameter of 1. This kernel was used with varying parameters for C, which produced the following graph

Figure 4: Performance vs Perplexity Parameter



It can be seen from the graph that a perplexity of 10 produced the best performance for the given dataset.

Prac 9

Q 9.5

Q 9.6