COL341 : Machine Learning
Assignment 2.1

Gaurav Jain (2019CS10349)

October 5, 2021

1 Part (c) : SGD Variant Selection

There are many variations of the vanilla stochastic gradient descent algorithm. In this part, we try to select the best version from the set of the given variants of SGD.

1.1 Variant Selection using Architecture 1

Architecture 1 contains four hidden layers with 512, 256, 128 and 64 neurons respectively. The output layer is a softmax layer with 46 neurons. ([512,256,128,64,46])
We compare the variants by plotting Cross-entropy loss v/s number of epochs. We use architecture 1 and run the model for 10 epochs with an adaptive learning rate method with initial learning rate 0.1
It is generally observed that adaptive learning rate methods with large initial rate seems to converge loss faster compared to fixed one. Thus, for our problem where we want to converge loss as fast as possible, we start with this method. We want to find the best activation function also for the network. Thus, three plot are present below with activation functions log-sigmoid, tanh and relu.

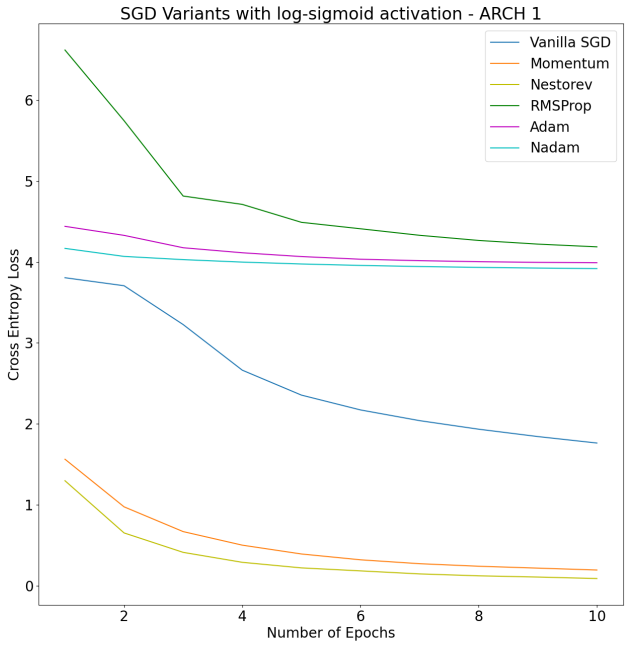

Figure 1: SGD Variants with log-sigmoid (ARCH 1)

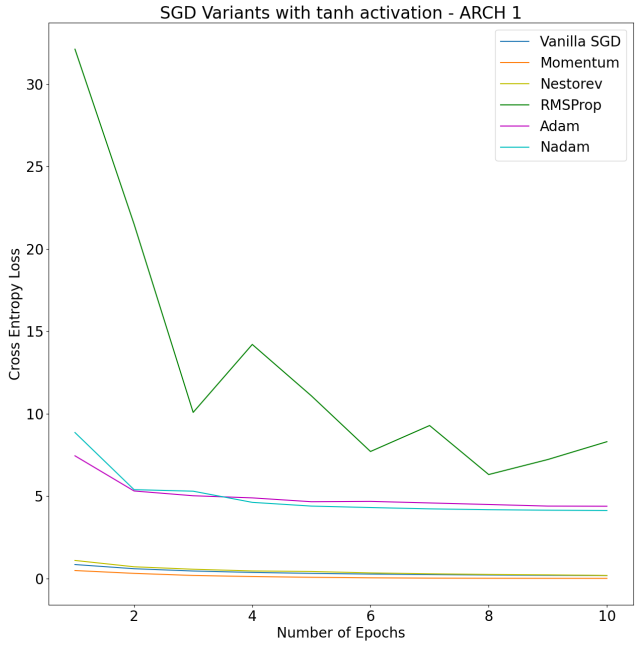

Figure 2: SGD Variants with tanh (ARCH 1)

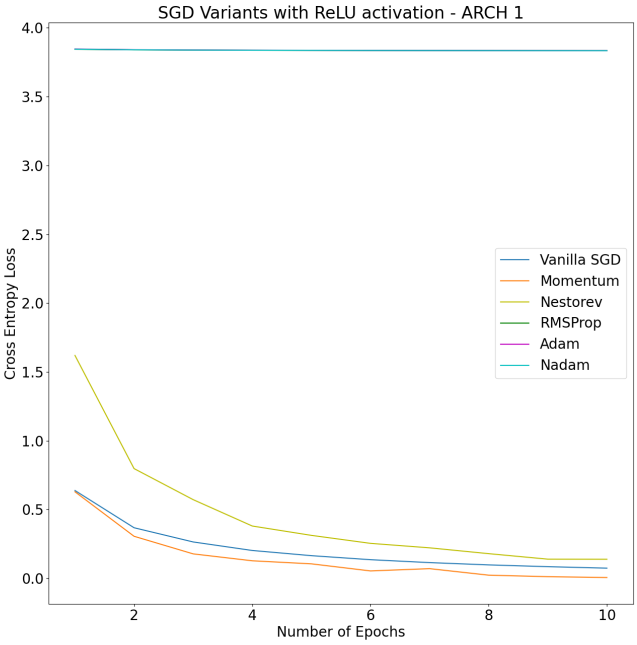

Figure 3: SGD Variants with relu (ARCH 1)

Observations:

1. It is visible in all three plots that Adam, Nadam and RMSProp don't perform very well. They tend to have large loss and the change in loss is also not significant in Adam and Nadam.
2. Vanilla SGD seems to perform well using adaptive learning rate method with large initial learning rate. Simple variants of Vanilla SGD like SGD with momentum and Nesterov SGD also seems to perform well for all three activation functions.

3. Loss is highest using log-sigmoid activation function and Momentum SGD achieves minimum loss(almost zero) in case of relu activation function. tanh also gives small loss values for Vanilla, Momentum and Nesterov variants.

Since relu tends to reduce the loss at a larger rate, we select relu as our activation function for the model. The best performing variant using relu is Momentum and hence it is used in our model using architecture 1.

1.2 Variant Selection using Architecture 2

Architecture 2 contains one hidden layer with 256 neurons. The output layer is a softmax layer with 46 neurons. ([256,46])
We compare the variants by plotting Cross-entropy loss v/s number of epochs. We use architecture 2 and run the model for 10 epochs with an adaptive learning rate method with initial learning rate 0.1
We also want to find the best activation function for the network. Thus, three plot are present below with activation functions log-sigmoid, tanh and relu.

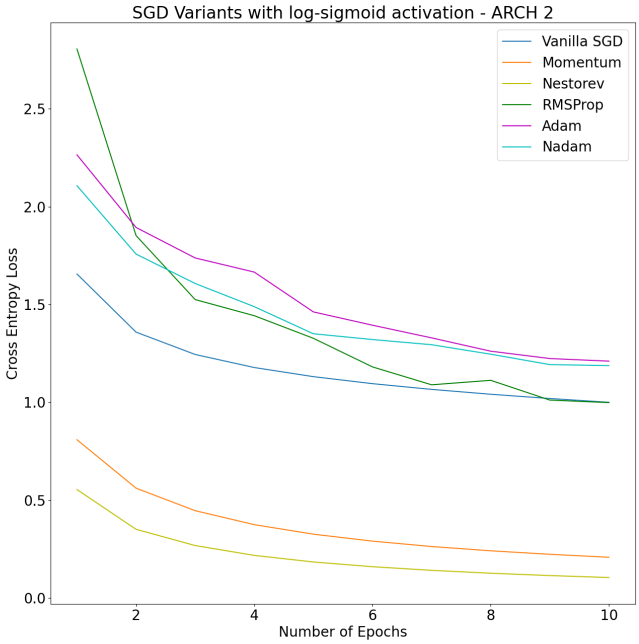

Figure 4: SGD Variants with log-sigmoid (ARCH 2)

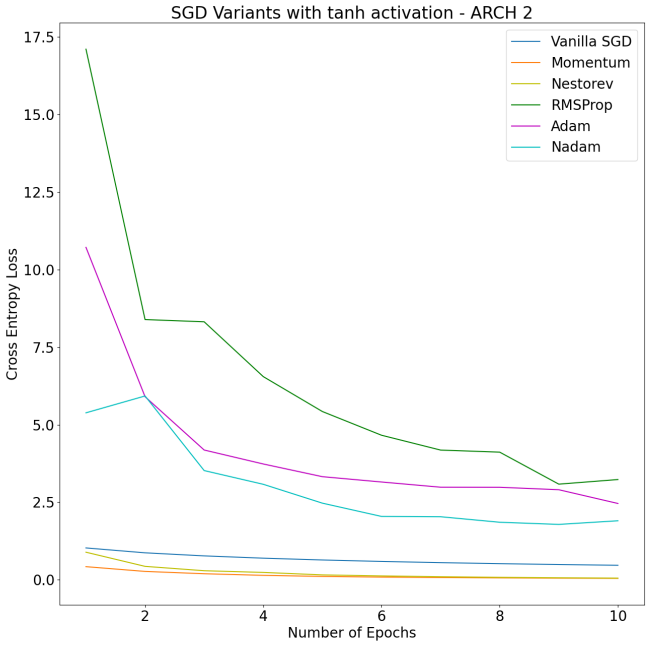

Figure 5: SGD Variants with tanh (ARCH 2)

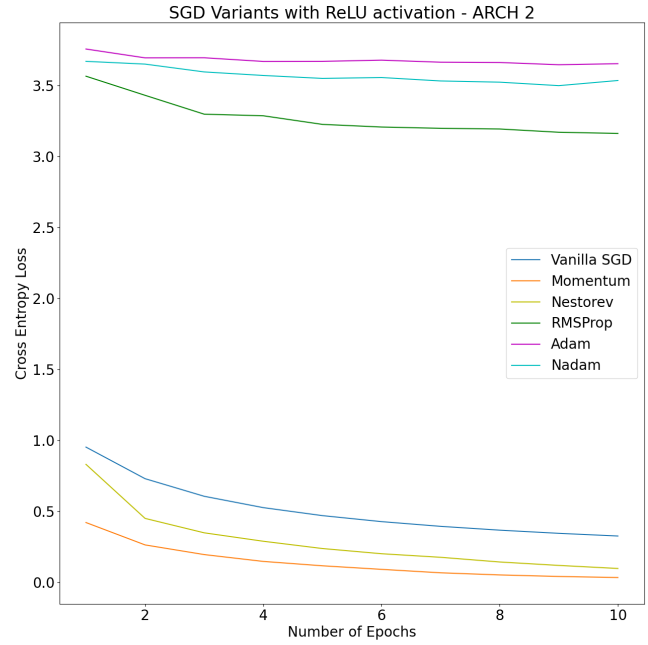

Figure 6: SGD Variants with relu (ARCH 2)

Observations:

1. Similar to architecture 1, Momentum and Nesterov SGD variants seems to perform well for all three activation functions.
2. Adam, Nadam and RMSProp also seems to reduce the loss at a large rate but they will require large number of iterations.

3. Loss is still high using log-sigmoid. using momentum SGD variant and relu activation, the loss seems to tend to zero.

Thus using above observations, we can select Momentum SGD with relu activation function for models using architecture 2. Momentum SGD runs quickly and thus, for the given problem we give a large number of epochs (70) although the change in loss is very small for higher epochs.

2 Part (d) : Architecture Selection

We are done with the task of selecting the best variant of SGD for our problem. We have also selected the best activation function. Using the combination of these two entities, we are able to converge the cross-entropy loss faster.
Thus, to select the best architecture, we vary two parameters: (1) Number of hidden layers and (2) Number of neurons in each hidden layer.
Before that, we should argue for softmax v/s any other activation function in last layer. From the theory of multiclass logistic regression, softmax function is an excellent method of predicting the classes of unseen data.So using this theory we can argue that softmax is the best choice for output layer.
We plot accuracy of validation set v/s number of epochs for different architectures used in the model. The first plot is using architectures of different number of hidden layers. The plot is given below:

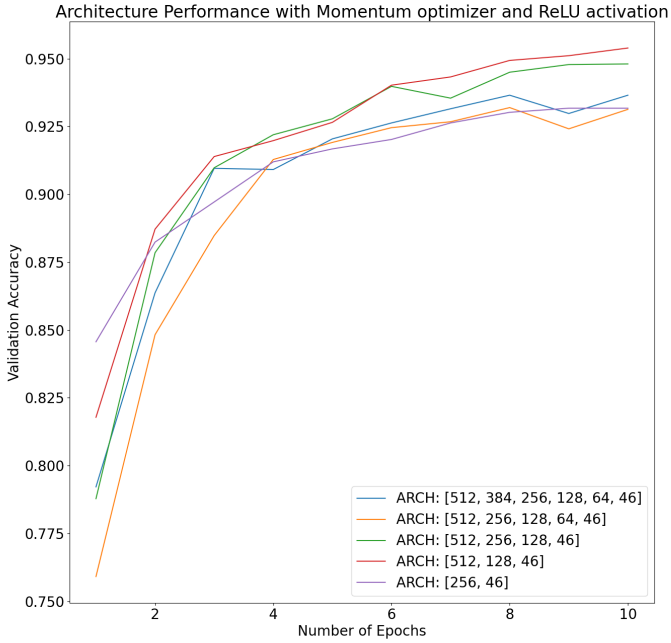

Figure 7: Architecture Performance with Momentum Optimizer and ReLU activation

All architectures start with large improvement in validation set accuracy and then slows down after a certain number of epochs. Architectures with 4 and 5 hidden layers are converging to a lower score of validation set accuracy compared to architectures with 2 and 3 hidden layers.
Over-training is possible for these architectures. Whereas, the architecture with only one hidden layer shows over-fitting of training set. Thus, we should select architectures having 2 or 3 hidden layers.

After selecting number of hidden layers, we should select number of neurons present in each hidden layer. We generate a sample of possible neurons by varying neurons present in each layer by doubling them or reducing to half their values. Again, validation set accuracy is used as a metric for selecting the best architecture. The plot of validation set accuracy v/s number of epochs is present below:

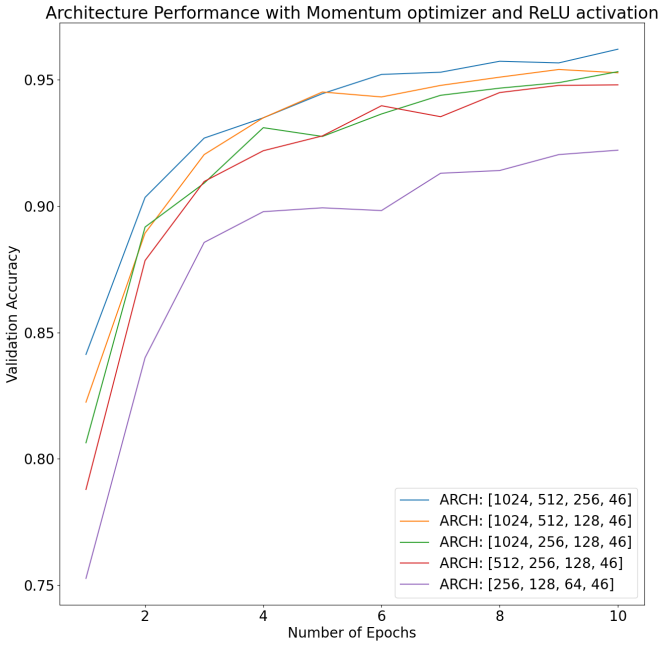

Figure 8: Architecture Performance with Momentum Optimizer and ReLU activation

It is clearly observable that the smaller number of neurons reduces the validation set accuracy by a large amount. Thus, the only samples that we should consider should have larger number of neurons. [1024,512,256,46] is the architecture with best validation set accuracy after 10 epochs. This architecture is selected for the model. This architecture shows high accuracy for validation set and runs quickly too as we are using momentum SGD and relu activation.

Best architecture details:

- Layers = [1024,512,256,46]
- Activation = ReLU
- SGD Variant = SGD with Momentum ($\beta = 0.9$)
- Adaptive Learning Rate with initial learning rate 0.1
- Batch Size = 100
- Epochs = 20
- Loss = Cross-Entropy loss