CS 485 : Project Report

Curriculum Learning

Mihir Kulkarni (12D020007)*

Supervised by: Prof Shivaram Kalyanakrishnan*

May 3, 2016

1 Abstract

I describe work done on experiments in curriculum learning, introduced in [1]. I found trends in convergence behaviour for certain training regimes on neural nets, among others, described below. This should serve as a start point to further investigate how these neural nets behave in terms of error as one changes training data.

2 Introduction

2.1 Inspiration

Humans learn best when they are initially shown easier examples illustrating a concept, and are then shown increasingly complex ones.

For several machine learning problems, it turns out that strategies inspired from this methodology, which involve "starting small", seem to work better than presenting all examples at once, in terms of generalization and convergence.

2.2 Curricula

Formally, for a supervised learning problem, we can view a curriculum as a sequence of training criterion, each identified by a particular weighing on the training data points. These weighings are designed so as to assign heavier weights to easier examples for the initial training criterion, and become progressively more "complex" (formally, undergo an increase in entropy) as one moves to later criterion in the sequence.

*Department of Computer Science and Engineering, IIT Bombay

2.3 Optimizing nonconvex training criterion

Problems such as training deep neural networks consist of optimizing an objective which is highly non-convex. Therefore, depending on the start point, one reaches local minima that may or may not be close to optimal. Deep architectures seem to give fairly good such minima, that are nearly impossible to achieve with random initialization. However, they can be computationally intractable.

Bengio et al hypothesize in [1] that the curriculum learning method acts essentially as a continuation strategy. This means that one optimizes a smoothed version of the objective, before iterative using the solution to optimize increasingly non-convex versions, before finally optimizing the final cost. This gets the parameter to be optimized to the attraction basin of a dominant, if not global, optimum of the training criterion.

3 Experiments on the MNIST dataset

The MNIST dataset (`http://yann.lecun.com/exdb/mnist/`) consists of 28x28 labelled images of handwritten digits, which includes 50,000 training, 10,000 validation and 10,000 test samples.

I ran the following experiments on this dataset:

3.1 Naive Bayes

I trained a Naive Bayes classifier using the 50,000 training points, and classified the "easy" points as the one classified as correct by this classifier.

3.2 Logistic regression

Using the code from `http://deeplearning.net/tutorial/logreg.html`, I trained a logistic regression classifier on the MNIST dataset, and compared results for a no-curriculum strategy with various curricula.

3.3 Neural net

3.3.1 Structure

This used the code from `http://deeplearning.net/tutorial/mlp.html`. I used a neural net with:

1. 28 x 28 input neurons.
2. 300 hidden neurons.
3. 10 output neurons.
4. a training algorithm which uses minibatch stochastic gradient descent with certain early stopping parameters.

The non-linearity used to transform the input neurons to the hidden ones is *tanh*, while the output consists of a softmax layer with ten neurons.

I compared results for a no-curriculum strategy with various curricula, standardizing results using forty training epochs each.

3.3.2 Types of curricula and results

1. **No curriculum**
   This converged to an error of around 2.85% after forty epochs.

2. **Choosing fewer classes in the initial criterion**
   For doing this, I computed the confusion matrix for the test data to figure

   a) Frequency of misclassifying each class.

   b) Which other classes is each class confused for.

   I then selected a subset of classes that were easily distinguishable (not confused with each other and low on error frequency) and proceeded to run the training with two training datasets:

   a) All data points from the subset of classes chosen, which was used for a fixed initial number of epochs, until a "switch" epoch.

   b) The full training data, used for the remaining epochs.

   Results
   As is clear from Figure 1, the neural net trained on partial-class curriculum I'd designed had a huge error during the start of training, and began behaving similar to the one trained on the full data after the switch epoch. This was perhaps because there was no training done for the classes outside the initial subset during before the switch epoch.

3. **Picking equalized samples from each class**
   This involved creating a random permutation of the training data, and picking an equal number of indices from the data for each class, to form one minibatch (except the last several ones, which may have less than ten elements from certain classes) for each. This gives us an **equal number of samples from each class** for most minibatches. Note that there is no multiple-stage curriculum here, it is just an ordering of the data in minibatches.

   Results
   From Figure 2, this performs consistently worse than the original data. This might be because several batches have data from less than ten classes.

Figure 1: Error versus epochs for the fewer-classes criterion

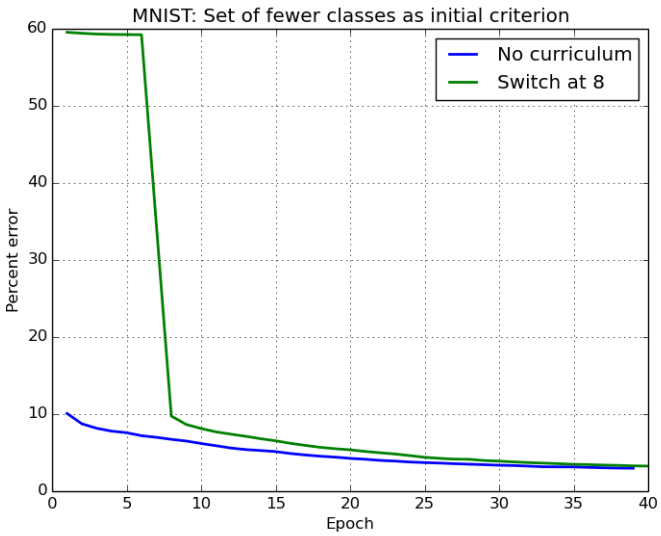
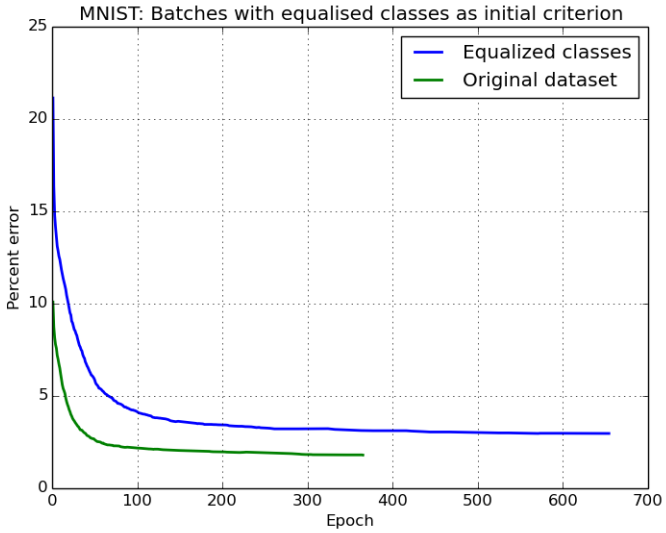


Figure 2: Error versus epochs for the equalized-classes criterion



4 Experiments on the BabyShapes dataset

The BabyShapes dataset, available at `http://www.iro.umontreal.ca/~lisa/twiki/bin/view.cgi/Public/BabyAIShapesDatasets`, consists of 32x32 images of geometric shapes of three types: rectangles, ellipses and triangles. This consists of various sizes of training sets. The pixels take values (intensity) within a small set of values (usually less

than three to four per image). Following [1], I used two datasets to form the training curriculum:

1. The BasicShapes dataset, which consists of 10,000 samples of "basic" shapes, i.e. squares, circles and near-equilateral triangles. These form the "easy" examples for the initial training set of our curriculum.

2. The GeomShapes dataset, which has 10,000 samples of arbitrary rectangles, ellipses and triangles.

4.1 One hidden layer neural network

This used the network identical to the one described at `http://deeplearning.net/tutorial/mlp.html`, but with 500 hidden neurons, and three output neurons at the softmax layer.

4.1.1 Training

I trained the network on the BasicShapes dataset for 16 and 32 epochs, before switching to GeomShapes for the rest. As seen from the figure, the curriculum regime performed markedly better than the usual.

4.2 Results

Note that in Figure 3, the regime with the switch at 16 epochs:

1. performed identically to the one with switch epoch 32, till epoch 16.

2. did not see improvement in validation error for several epochs beyond 16, which is why it performed an early exit after those epochs.

Also, note the spike in error after (switch) epoch 32 for the curriculum learner. I'm not sure why exactly this happens, but it might be because suddenly switching datasets might induce a transition phase where the parameter matrix is moving from a minima for the first dataset to one for another.

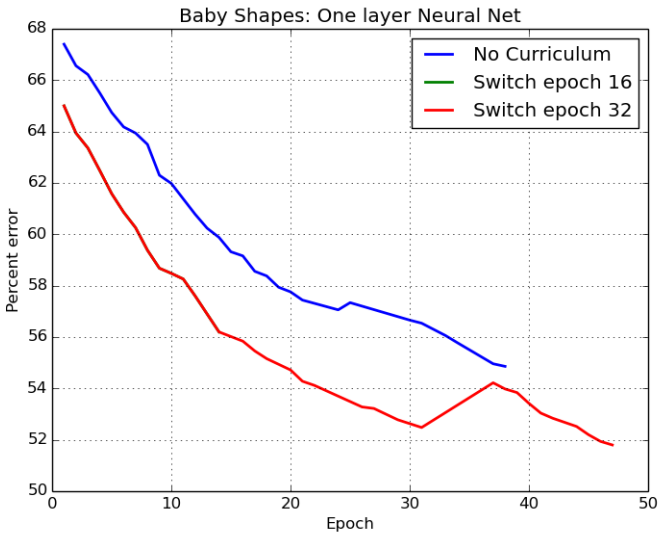
4.3 Three hidden layer neural network

Since [1] uses a three hidden layer neural network, I changed mine to one with 200, 200 and 100 neurons in its hidden layers.

4.4 Training

The training regime was identical to the one-layer neural net, with switch epochs 0 (no curriculum), 32, 48 and 64 and no switch (using only BasicShapes) being compared, for around 100 epochs.

Figure 3: Error versus epochs for BabyShapes (one layer net)



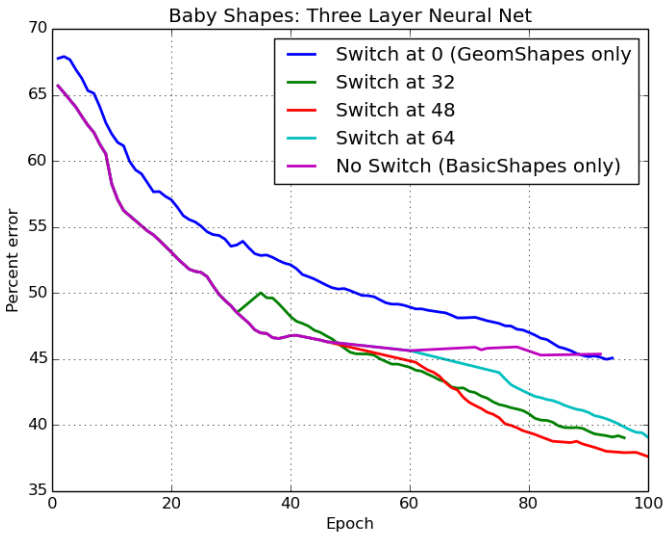
4.5 Results

The results are seen in Figure 4. Note that

1. The curve for the pure BasicShapes dataset plateaus at around epoch 40.
2. The curve for switch epoch 32 spikes just after the switch epoch. It dips quickly after that however, achieving the second-best overall performance at the end.
3. There is no spike in error in the curves for switch epochs 48 and 64. These curves dip after their switch epochs (after some flat behaviour before them). The curve for switch epoch 48 achieves the best overall performance.
4. Training on only the GeomShapes dataset begins with the highest error, ends up beating only the pure BasicShapes dataset curve, around 50 epochs after the latter plateaus.

This shows us that while the network trained on BasicShapes begins better than GeomShapes, it plateaus after a certain point, while the one trained on GeomShapes keeps on improving. While the characteristic error spike is present for the curve with switch epoch 32, it isn't for those with switch epochs that occur beyond the beginning of the plateau for the BasicShapes curve. The best performance (till epoch 100) is achieved by the curve that switches closest to, but after, the beginning of the BasicShapes plateau (epoch 48).

Figure 4: Error versus epochs for BabyShapes (three layer net)



5 Further work

1. Verifying whether switching closest to the plateau is the optimal switch epoch, in terms of final error and convergence speed, by checking till convergence for various switch epochs.

2. Figuring out why there is a spike in the error just after switching the dataset, and whether that ties into where the switch should be made.

3. Identifying a method to generate curricula for various types of problems, based on developing a broader theory.

4. Testing with separately with examples from the "easier" and the "harder" datasets, to see how each switch epoch performs on each.

6 Acknowledgement

I'd like to thank my guide, Prof. Shivaram Kalyanakrishnan, for introducing me to the field and for his timely guidance.

References

[1] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, Jason Weston: *Curriculum Learning*, ICML'09.