229 Project Final Write-up: Generating Ad Hoc Curricula

Andrew Lampinen (lampinen@stanford.edu)

1 Introduction

Often in machine learning, proofs of convergence of learning algorithms assume that the data distribtuion is static over the course of learning. However, work on Curriculum Learning [1] has shown that in practice, altering the data distribution over the course of learning can improve results, at least in some cases. Specifically, starting with easier examples first and then progressing to harder examples can be beneficial, possibly even making a task learnable when it is not learnable from a reasonable amount of data using the standard strategy. This idea has proven useful in many contemporary networks, e.g. [3].

However, in many cases we lack explicit knowledge about the structure of the problem from which we could construct a curriculum. For example, on the MNIST task, how do we decide which examples are "easy" or "hard?" Perhaps we could have human coders evaluate every image and call an image harder the more they disagree, but that would be prohibitively expensive. Is there a way we can craft curricula for a task while remaining more agnostic to what the task actually is (ideally just from the data vectors and labels)?

2 Proposed ad-hoc curricula

2.1 Data recombination and central tendencies

First, we consider a way of crafting curricula which, while not entirely data agnostic, we think may be beneficial in a wide variety of practical scenarios. The basic idea is very simple. If our data are noisy samples from some distribution, we might think that by combining/averaging data points with the same labels, we will generally yield less noisy/better estimates of the central tendency of the data distribution for this class, which may provide a better place to begin learning. In an ideal case where the data form distinct clusters, the centroids of the clusters will be far from eachother even if individual exemplars may lie close to other clusters. Thus the averaged examples may be "easier" in the sense Bengio et al. described. For example, on MNIST the average of all the 0 examples is probably more 0 like than many of the individual examples. Thus, averaged data may provide "easier" examples. However, this intuition requires making a few important assumptions about the distribution of the data for each class:

1. The data are well represented by their central tendency. (E.g. in an XOR problem, the mean of either class would be a poor representation of that class, because the means of the classes coincide.)
2. The variance within the class is not large relative to the between-class variance. (The central tendency has to convey useful information.)

For example, on a visual recognition task like ImageNet these assumptions would likely be violated – the average of all dog images in ImageNet would likely just be a muddy blur, which would look nothing like a dog. Furthermore, the variance in pictures of dogs is very high, the dog could be facing the camera, jumping after a frisbee, etc. Thus creating curricula by averaging might not be as helpful. However, even here there might be useful color information, for instance, which could be obtained from the average. Similarly, if there is a large contingent of images that center the face of the dog, there might be useful information in the average about dog faces. We think it is an open question whether this strategy would be beneficial there.

2.2 Confusing examples from previous training

We also considered the possibility of deriving an ad-hoc curriculum by training a classifier with no curriculum, and then ranking example easiness by what examples it found most confusing (e.g. easy examples have large positive margin, hard examples have small positive to large negative margin). This can be seen as an approximation to one of the original curricula suggested by Bengio and colleagues, where instead of using the unknown true margin as a measure of easiness we are using the margin from our approximation to the true classification boundary.

This bears some resemblance to active learning strategies that have been suggested previously [5, e.g.]. However, it is distinct because we are deriving a curriculum rather than choosing examples actively online. Furthermore, most active learning methods have favored selecting hard examples at all times, here we are favoring putting them off until after easy examples have been seen.

3 Preliminary test

First, we tested our strategies on the toy example used in [1]. The task is simply to learn a linear classifier with a single layer neural network (i.e. there is no bias, the true target function lies within the space of single layer neural networks). One curriculum they suggested was sorting the data by the margin, with the idea that points farther from the classification boundary are "easier," in the sense that the classifier boundary can be more inaccurate and still classify them correctly. Bengio and colleagues found this resulted in better generalization after training on 200 examples.

We used both this curriculum strategy and a no-curriculum random-order training strategy for comparison. To test our ideas we created:

- a data recombination ad-hoc curriculum consisting of 200 examples ordered as follows:
    1. 10 positive and 10 negative examples (alternated), each created by averaging 10 examples of the expected class
    2. 10 positive and 10 negative examples (alternated), each created by averaging 5 examples of the expected class
    3. The remaining 160 examples were the last 160 examples the non-curriculum network saw.
- a previous classifier-based ad-hoc curriculum consisting of 200 examples sorted by a previous classifier's margin on them after training was complete (from largest to smallest margin).

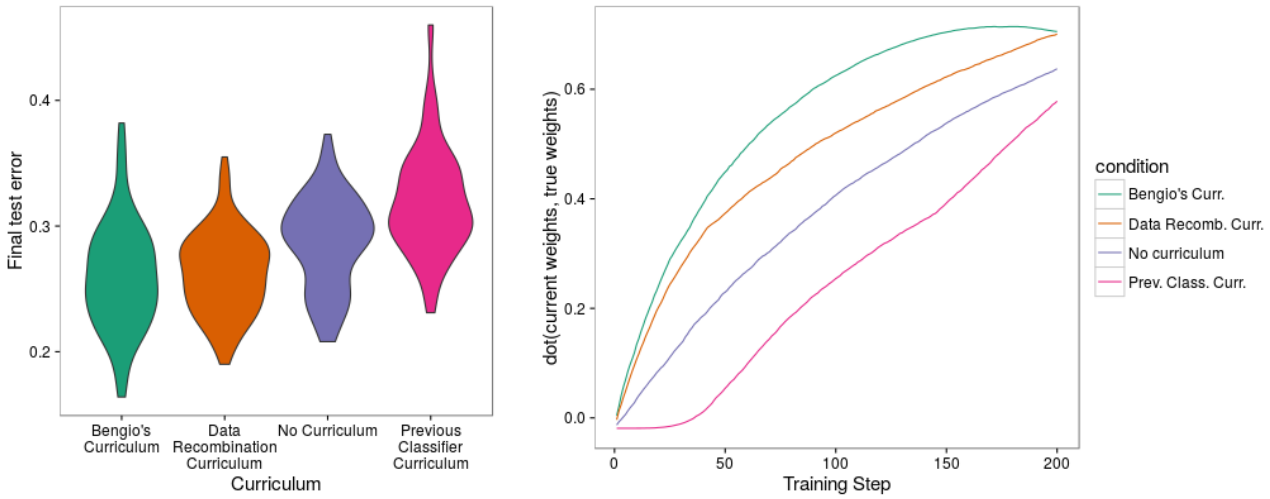
We evaluated these strategies on two metrics:

- Error on 1000 new test examples
- Dot product of learned parameters with true parameters (normalized by the lengths of the vectors)

See Figure 1a for the density plot of the final test error across 100 runs, and Figure 1b for the evolution of the weights over time averaged across 100 runs. The non-curriculum network achieved an average generalization error of 28.7% with a standard deviation of 3.7%, and an average cosine distance from the true weights of 0.363. Bengio and colleagues' margin-based curriculum achieved an average generalization error of 26.0% with a standard deviation of 4.3%, and an average cosine distance from the true weights of 0.295.

3.1 Data recombination results

The data recombination curriculum performed nearly as well on average as the curriculum used by Bengio et al., and performed slightly more consistently. It achieved a mean generalization error of 26.3% with



(a) Final error rates by curriculum (density over 100 runs)

(b) Evolution of dot product between true parameters and current parameters over learning (averaged over 100 runs)

Figure 1: Results on toy model from [1]

a standard deviation of 3.3%, which was significantly better than the non-curriculum strategy (paired t-test, t(99) = 7.9, p = 3.2e-12), and not significantly different than Bengio's curriculum (paired t-test, t(99) = -1.0, p = 0.32). It achieved an average cosine distance from the true weights of 0.299.

This is a very promising result, we were able to craft a curriculum without using knowledge about the data that performed as well as the curriculum used by Bengio and colleagues which relied on explicit information about the structure of the task.

3.2 Previous classifier curriculum results

By contrast, the previous classifier curriculum performed quite poorly, in fact worse than the non-curriculum. It achieved a mean generalization error of 31.9% with a standard deviation of 4.2%, which was significantly worse than the non-curriculum strategy (paired t-test, t(99) = -8.8, p = 5.3e-16), and an average cosine distance from the true weights of 0.422.

It is possible that these results were so poor because the margin estimates are not just wrong, they are wrong in a biased way, since they are all computed from the same classifier. Using many previous classifiers and averaging margins across them might provide more robust estimates that would allow this strategy to work. However, on complex problems like MNIST where training each classifier takes a non-trivial amount of time, this would become prohibitively expensive (at least with the computational resources we have access to). Thus we will not pursue this strategy further in this project.

4 MNIST & Noisy MNIST

Next we decided to test the data recombination strategy on a classic machine learning problem: MNIST. We tested this strategy on both the standard MNIST dataset and a custom version with added noise.

4.1 Implementation

Network Architecture & Training: We used TensorFlow [2] to implement a neural network with two convolutional layers with 5 x 5 filters (32 and 64 filters in the first and second layer respectively) and 2 x 2 max-pooling, followed by a fully connected final layer with the classification outputs. We included dropout between the second convolutional layer and the final convolutional layer. (The code for this was

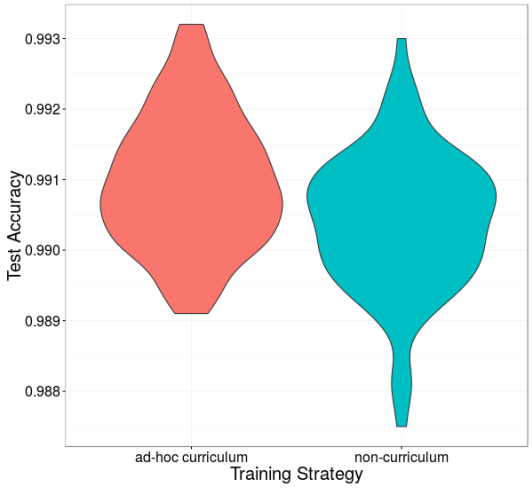
adapted from the TensorFlow MNIST tutorial.) We shared weight initializations between the networks that were trained with and without the curriculum on each trial. We trained the networks by using the cross-entropy error as a loss function, using the Adam optimizer [4] with a learning rate of 0.001 and a batch size of 50. We trained for 20000 batches, and evaluated performance on a validation set every 100 epochs. We used the time step that achieved the max validation performance to evaluate test performance.

Curricula: We compared a non-curriculum training strategy (random order) to a curriculum constructed as follows:

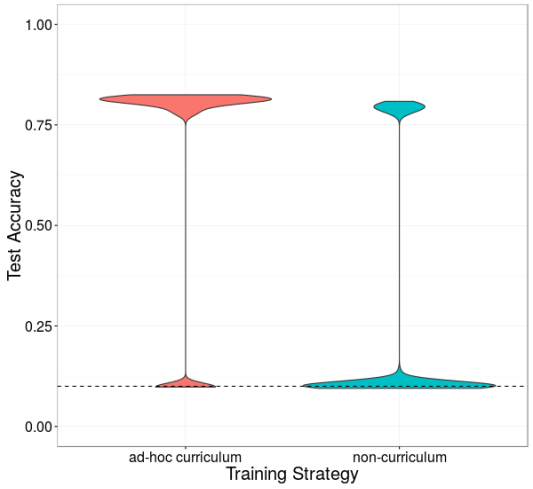
- The first 20 batches consisted of 5 "central exemplars" of each number, each of which was created by averaging together 50 examples of that number.
- The next 40 batches consisted of 5 "central exemplars" of each number, each of which was created by averaging together 20 examples of that number.
- The remaining batches were identical to those seen by the standard network.

4.2 MNIST

First, we tested our curriculum on standard MNIST. The non-curriculum network achieved a mean test accuracy of 99.04% (s.d. 0.10%), while the network trained with a curriculum achieved a mean test accuracy of 99.10% (s.d 0.09%). This difference was significant (paired t-test, t(99) = 4.1, p ¡ 1e-4). See Fig. 2a for a plot of the results. Despite the fact that the curriculum was statistically significantly better, it was not really practically different, so this cannot be considered a strong endorsement of this strategy.



(a) Final test accuracy by curriculum on standard MNIST (density over 100 runs)

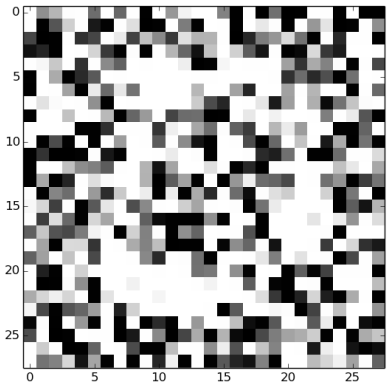


(b) Final test accuracy by curriculum on noisy MNIST (density over 100 runs)

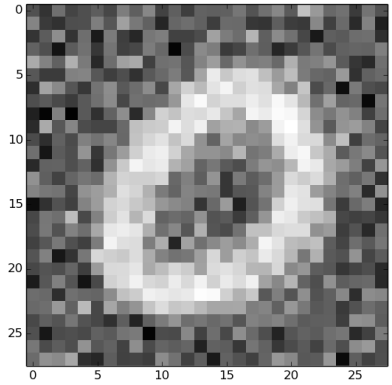
4.3 Noisy MNIST

Next we tested our curriculum on a noisy version of MNIST. This was constructed by adding independent gaussian noise (variance 0.66) and binomial noise (p = 0.66) to each pixel, and then clipping the output to $[0, 1]$. This mimics different possible sources of noise in a sensor system, noise from many random processes that becomes approximately normal and shot noise. This created a very difficult task, with quite noisy images, see Fig. 3a for an example. However, averaging examples is very powerful with this noise structure, see Fig. 3b for an example. We chose this task because we thought it might show the particular benefits of our strategy for noisy data.

The non-curriculum network achieved a mean test accuracy of 26.2% (s.d. 29.3%), while the network trained with a curriculum achieved a mean test accuracy of 66.05% (s.d. 29.2%). This difference was significant (paired t-test, t(99) = 11.0, p ¡ 1e-15, but note that because the data were extremely non-normally distributed, this test may not be appropriate). See Fig. 2b for a plot of the results. The network trained with the curriculum was more consistently achieving high performance than the network trained without, and the maximum performance (over all runs) of the network trained with the curriculum was 82.5%, compared to 80.9% for the networks trained without curricula.



(a) Sample "0" from our Noisy MNIST dataset



(b) Average of 50 examples of a 0 from our Noisy MNIST dataset

5 Conclusions

Our results suggest that using curricula based on data recombination may allow more consistent performance and better generalization than standard training, especially in domains where the data are distributed insuch a way that the mean is a good representation of the distribution (e.g. where the data are all similar up to independent noise). This technique could be extended in many ways, for example clustering a class and finding several central tendencies to use as early training examples.

References

[1] Bengio, Y., Louradour, J., Collobert, R., and Weston, J. Curriculum learning. *Proceedings of the 26th annual international conference on machine learning* (2009), 41–48.

[2] GoogleResearch. TensorFlow: Large-scale machine learning on heterogeneous systems.

[3] Graves, A., Wayne, G., Reynolds, M., Harley, T., Danihelka, I., Grabska-Barwińska, A., Gómez Colmenarejo, S., Grefenstette, E., Ramalho, T., Agapiou, J., Badia, A. P., Moritz Hermann, K., Zwols, Y., Ostrovski, G., Cain, A., King, H., Summerfield, C., Blunsom, P., Kavukcuoglu, K., and Hassabis, D. Hybrid computing using a neural network with dynamic external memory. *Nature Publishing Group 538*, 7626 (2016), 471–476.

[4] Kingma, D., and Ba, J. Adam: A Method for Stochastic Optimization. *International Conference on Learning Representations* (2014), 1–13.

[5] Krogh, A., and Vedelsby, J. Neural Network Ensembles, Cross Validation, and Active Learning. *Advances in Neural Information Processing Systems 7* (1995), 231–238.