Automatic Curriculum Learning for Deep Models Using Active Learning

Ian McWilliam

Master of Science
Artificial Intelligence
School of Informatics
University of Edinburgh
2018

Abstract

This

Acknowledgements

Many thanks

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Ian McWilliam*)

Table of Contents

Chapter 1

Introduction

Supervised learning is the area of machine learning in which algorithms learns the relationship between a set of input features and corresponding 'ground truth' labels, the ultimate goal being to construct a predctive model of the relationships between the inputs and the labels in order to predict the labels of future, unseen input samples. Deep learning models perform this task by building hierarchical representations of the input features throughout a multitude of layers, often using feature maps such as convolutions or recurrent layers to construct complex representations of the inputs. When training a deep model a standard methodology is *gradient descent*, which calculates the gradient of a chosen error function with respect to the free parameters of the model so as to tune the parameters in a way that will minise this error function on the training set of input-label pairs. A popular variant of the gradient descent algorithm is *mini-batch stochastic gradient descent*, which uniformly samples mini-batches of a preset size from the available training data, performing a gradient descent update on each batch until the all training samples have been selected, then repeating until the network converges to a solution. Sampling uniformly from the training data ensures that the mini-batch gradient is an unbiased estimation of the gradient over the whole training set, however the estimation can exhibit high variance. In this paper we analyse approaches for augmenting mini-batch stochastic gradient descent (SGD), using methods inspired by two areas of study; *active learning* and *curriculum learning*.

Active learning is generally used when there is a prohibitive cost to obtaining labels for supervised learning; in such cases it is desirable to know which samples will lead to the greatest best improvement in algorithm performance, selected from a set of unlabeled candidate samples. As such, there is a rich literature in active learning detailing how to choose the most informative samples, in particular using *acquisition functions*

to select which sample(s) to label and use for training. While active learning is usually employed to reduce labeling costs and speed up learning, curriculum learning explores the hypothesis that the overall accuracy of the network can be improved by presenting the training data to the algorithm in a meaningful order. Inspired by the way in which humans and animals learn, REF BENGIO suggest learning can be improved by emphasising easier concepts earlier on in training before introducing difficult samples, or by emphasising more difficult training samples later in training. In their paper REF BENGIO for example, the authors use a the 'Geometric Shapes' dataset, consisting of images of geometric shapes of different complexities, to show that by initially training on 'easy', regular shapes, test classification accuracy is improved.

The substantial challenge with curriculum learning however is that in many domains however it is challenging to identify a clear delineation between 'easy' and 'hard' samples through which to implement curriculum learning; in this paper we propose that the methodologies developed for the active learning approach are well suited to estimating the difficulty of training samples, allowing the automatic construction of learning curricula that will improve the training of deep networks on a wide range of tasks. Specifically, the approach set out in this paper modifies the SGD algorithm by, instead of sampling with uniform probability, sampling training examples proportionally to some measure of 'difficulty', as derived from an active learning style acquisition function metric. We test our methods on three image classification datasets; MNIST, CIFAR 10 and the GeoShapes dataset (a geometric shapes classification dataset with an established curriculum baseline), exploring a range of active learning metrics as well as several curriculum construction methods. Our results show consistent performance against a uniform sampling baseline, with significant reductions in test set error, robust to different network architectures, datasets and curriculum methodologies. The output of this work is a set of flexible methods for improving deep models in a wide range of tasks, as well as an investigation of how using the difficulty and uncertainty of training samples affect learning performance.

In the next section we will introduce in more detail active and curriculum learning, exploring the link between the two approaches and the sometimes contradictory hypotheses they pose. We will then discuss related work where the authors implement similar methods for improving algorithm performance through biasing learning towards certain training samples throughout training. We will then lay out the experimental methodologies and datasets used in the paper before presenting and analysing the results of the tests and concluding with a discussion and suggestions for further

work.

Chapter 2

Background

2.1 Supervised Learning

Briefly introduce concept of supervised learning, input features, labels, training, validation and test set, with some examples. *Supervised Learning* is the sub-discipline of machine learning concerned with modelling the relationships between input-output pairs, usually with the goal of predicting the label of future, unlabeled input samples. A classic example of supervised learning is image classification, where the inputs are often the RGB colour values of pixels in the image, and the output label is a category describing what the image is of. Key to supervised learning is a 'training set' of labeled samples which a learning algorithm can perform inference on in order to understand how the different input values affect the corresponding labels; in order to ensure that the model is not simply memorising the labels in the training set , a held out 'test set' is generally also used to test the performance of the learning algorithm on unseen samples once it has finished learning from the training set. In many setttings a 'validation set' of labeled samples may also be used to monitor the performance of the learning algorithm throughout training. The main aim with supervised learning is to minimize the *generalization error* of the supervised model, generalization error is the model's expected error on future samples; as in most real world settings this cannot be precisely measured it is usually approximated by the error on the test set. While the ultimate goal of a supervised is generally to minimize generalization error, there are several other criteria which can affect the overall performance of the model; in particular, the time taken for the model to converge to an optimal solution is a key metric when testing supervised learning algorithms.

2.2 Deep Learning

Deep learning, as applied in the supervised setting, refers to algorithms which model the relationships in the training set using complex, hierarchical representations of the data, usually doing so using multiple 'deep' layers, as well as feature maps such as convolutions or recurrent layers. In image classification in particular, the state of the art is *convolutional neural networks*, which assigns weights to areas of a predefined size of the input space, as opposed to applying weights to each input node. Doing this allows for the automatic construction of abstract features which has proven extremely effective for analysing images, for example by modeling specific shapes characteristic of certain labels.

2.3 Stochastic Gradient Descent

The standard method for training deep models is *gradient descent (GD)*, an optimization algorithm which varies the parameters in the model depending on the gradient of a chosen error function. To implement GD it is necessary to calculate the gradient of the error function with respect to the parameters of the model, usually this is done layer by layer, starting with the output layer, in a method referred to as *backpropogation*. Calculating this gradient however can be very computationally expensive, particularly when dealing with large training sets; to address this issue a variant of GD, *stochastic gradient descent (SGD)*, is often used. With SGD, instead of calculating the error gradient over the entire training set, only one sample is used to calculate the gradient and update the model parameters. Alternatively a selection or 'mini-batch' of training samples may be used to calculate the gradient, in which case the optimization algorithm is referred to as *mini-batch stochastic gradient descent*. It can be shown that that SGD and mini-batch SGD produce an unbiased estimate of the error gradient, with various convergence proofs showing that SGD will eventually converge to an optimal solution. A disadvantage to SGD however is that the gradient estimate, while unbiased, can exhibit high variance, potentially resulting in extremely slow converge times. There are many adaptations to 'vanilla' SGD, for example momentum based methods and other more sophisticated optimization algortihms which build on SGD to result in better and quicker fitting of the model parameters.

2.4 Active Learning

A key component in any supervised learning effort is labeled data; in many domains it is relatively easy and cheap to obtain large volumes training samples, however in others it can be far more costly, particularly acquiring accurate labels. In medical image analysis for example one may require a domain expert to spend significant time analysing each image before assigning label, or in document tagging it can take time to read a document and assign a topic label. It can therefore be very useful for a designer to understand which samples they should go to the effort of acquiring, labeling and feeding into their chosen learning algorithm, generally measured by how much the chosen samples improve the network performance, compared to if samples were instead selected randomly. We here introduce some of the main methodologies employed for active learning, giving the reader some background to the methods that will be used in this paper.

There are a variety of approaches to the active learning problem, however most involve the use of an *acquisition function*, which selects which sample, from a set of candidate unlabeled examples, should be selected for labeling and training. As the most appropriate training examples varies depending on the learning algorithm, as well as its current state in the training process, the chosen sample is said to be 'queried' by the algorithm. The motivation behind different active learning approaches vary; one of the most common approaches is that of *uncertainty sampling*, wherein the samples that the learning algorithm is most uncertain about labeling are queried. This uncertainty can be captured by analysing the distance to classification threshold of the model outputs; for example one method is to select the sample about which the model is least confident in predicting: (taken from REF SETTLES:)

$$x^*_{LC} = \arg\max_x 1 - P_\theta(\hat{y}|x), \tag{2.1}$$

where

$$\hat{y} = \arg\max_y P_\theta(y|x). \tag{2.2}$$

Where x^*_{LC} is the queried training sample and $P_\theta(y_i|x)$ is the model's predicted probability that sample x is of class y_i, given model parameters θ. Similarly, samples can be queried by their average distance to classification threshold or, similarly, the entropy of the algorithm prediction, again taken from REF SETTLES:

$$x^*_H = \arg\max_x -\sum_i P_\theta(y_i|x) \log P_\theta(y_i|x), \tag{2.3}$$

where the sum runs over the possible classes y_i.

An alternative approach to querying training samples is to estimate the expected change in model parameters, if trained on a given sample. As, in the active learning setting, it is assumed that the label is unavailable, this is calculated as an average across all potential labels. Model change can be estimated by the magnitude of the gradient vector produced by training on the tuple $\langle x, y \rangle$. The acquisition function then selects the sample which maximises the expected gradient size (REF SETTLES AGAIN):

$$x^*_{EGL} = \arg\max_x \sum_i P_\theta(y_i|x) \left\| \nabla \ell_\theta(\mathcal{L} \cup \langle x, y_i \rangle) \right\|, \tag{2.4}$$

where $\|.\|$ is the Euclidean norm, $\mathcal{L}$ is the current set of labeled training samples, ℓ is the objective function used to train the model and $\nabla \ell_\theta(\mathcal{L})$ is the gradient of this objective function with respect to the model parameters θ, when trained on $\mathcal{L}$. This approach therefore finds the sample that leads to the largest expected increase in the gradient when added to the training set $\mathcal{L}$.

Finally, another common approach for active learning is that of *query by committee*; here a population of different models are trained on an initial training set, then the samples about which the models exhibit the most disagreement in their predictions are queried. An example of this is *vote entropy* REF:

$$x^*_{VE} = \arg\max_x -\sum_i \frac{V(y_i)}{C} \log \frac{V(y_i)}{C}, \tag{2.5}$$

where C represent the size of the 'committee' (i.e. the number of models) and $V(y_i)$ is the number of models in the committee that predict predict label y_i. There are obvious parallels here to methods such as ensembling, boosting and bagging, indeed active learning has drawn parallels with several other learning paradigms, such as self-paced learning (REF) and curriculum learning (REF), the latter of which we shall now introduce.

2.5 Curriculum Learning

While active learning uses methods to identify which samples to label and train in order to speed up training in domains with a high labeling cost, *curriculum learning* attempts to present training samples to the learner in a meaningful order that will lead to greater overall generalization performance of the model. The motivation stems from the way in which humans and other animals learn, usually beginning with easy concepts before

moving onto more complex facets of the area of study. The same principle can be applied to training deep models, and the authors of REF suggest that, by initially training only on 'easy' samples, one can reduce overall generalization error. The authors offer several theoretical justifications, for example comparing curriculum learning to *continuation methods* REF; it is proposed that the easier samples represent a smoother, more convex version of the error space of the overall problem, and that, by training on easier samples, the parameters of the model are effectively initialized into an area of parameter space closer to the global optimum. This argument is similar to that of unsupervised pre-training, which again has been shown to lead to better generalized models by initializing the parameters into parts of the error space closer to the global optimum. Comparisons have also been drawn between curriculum learning and *transfer learning*, with the easier samples being seen as a separate task that the model is trained on, before using the weights for a different task (i.e. the harder samples) as in transfer learning.

The example given in REF BENGIO for curriculum learning is the 'GeoShapes' dataset, an image classification where a network attempts to classify whether or a not an image shows a rectangle, ellipse or triangle. In this case there is a natural subset of 'easy' samples; specifically squares (i.e. regular rectangle), circles (regular ellipses) and equilateral triangles. The authors show that, by training initially on only the regular shapes, then transitioning to training on harder shapes, the test set performance is significantly improved compared to training simply on the harder shapes for the entirety of training. One issue with this study is that it can be argued that the curriculum trained model has seen more samples overall than the baseline, as the curriculum model is trained on both an 'easy' training set and a 'hard' training set, whereas the baseline is trained only on the hard training set. A better baseline therefore is a model trained uniformly on the union of the easy and hard training sets. While the authors do comment on this issue, and claim that the curriculum method still outperform uniform sampling from the combined training set, the results we will set out in this paper did not reach the same conclusions.

A key difficulty in implementing curriculum learning is that it is often very difficult to delineate between 'easy' and 'difficult' samples, while it is also hard to ascertain how one should transition from different difficulties. A key issue therefore is that of exploring methods for automating the construction of learning curricula, and it is towards this goal that this paper contributes; specifically investigating how active learning methods can aid such curriculum construction. Having introduced the reader

to active and curriculum learning, the next section will lay out a variety of related work wherein the authors attempt to automate the process of curriculum construction or apply active learning methods with the goal of improving network performance.

Chapter 3

Related Work

3.1 Self Paced Learning

3.2 Transfer Learning

3.3 Reinforcement Learning

3.4 Active Learning

Chapter 4

Methods

4.1 Active Learning Metrics

The purpose of this paper is investigate how different active learning query metrics can be used to automatically construct learning curricula to improve the generalization performance of deep models. As such we select several active learning approaches, each of which can be used in combination with a curriculum construction methodology (REF SECTION) during training. Testing several methods will also allow us to test the robustness of the results and ascertain whether or not performance differences are consistent across different methods.

Each active learning method will be used to score the training samples, with the score then being fed as an input into the curriculum construction method to build the training mini-batches throughout the learning phase. As we are not 'acquiring' samples, rather we are calculating a score for every training sample, the terminology 'acquisition function' would be inappropriate, instead we refer to these score producing functions as *active score functions*. Each function will map training samples to a real number, which is then passed through a softmax function; this has the effect standardizing the various metrics, as well as allowing us to control the score ratio between different functions using the softmax temperature, as well as producing an output that can interpreted as sampling probabilities (see 4.1.4). In order to have consistency across the different methods we invert certain scores so that a *higher* score for a sample always corresponding to the sample being estimated to be *easier* for the sample to classify. The classic curriculum learning approach would therefore bias learning towards samples with high scores, while the classic active learning approach would bias learning towards samples with low scores.

As well as monitoring how the different active score functions affect model performance, we will also investigate how successful the different methods are at identifying which samples are 'hard' or 'easy', by visually inspecting the samples which receive very high or low scores by the different functions throughout training.

4.1.1 Average Absolute Distance to Threshold (AADT)

As laid out in section 2.4, a popular active learning method is to examine the proximity to the classification boundary of the model's outputted probabilities (assuming the model outputs probabilities; we will be using deep models with a softmax output layer). The assumption is that samples that the model is uncertain about classifying will produce probabilities close to the classification boundary; indeed as mentioned in section 2.4 the authors of REF show that prediction variance is inversely proportional to the distance to the boundary. From a curriculum perspective we can estimate a sample's difficulty by the algorithm's uncertainty in predicting the class label, with uncertain samples being seen as hard, and vice versa. We therefore calculate the distance to threshold active score function as follows, note that the score is proportional to the *inverse* of the average distance to threshold, in order to ensure that easier samples have a higher score and vice versa:

$$P_\theta^{AADT}(x_i) = \frac{exp(\frac{S_\theta^{AADT}(x_i)}{\tau})}{\sum_j^N exp(\frac{S_\theta^{DT}(x_j)}{\tau})}, \tag{4.1}$$

where

$$S_\theta^{AADT}(x_i) = \frac{C}{\sum_c^C \left| P_\theta(y_c|x_i) - \frac{1}{C} \right|}. \tag{4.2}$$

Where $|.|$ represents the L1 norm/absolute value function. Here N is the number of training samples, C is the number of output classes and $P_\theta(y_c|x_i)$ is the output softmax probability for class y_c of the model parameterised by θ, given input x_i. We tried a similar approach using the *square* of the distance to threshold, as opposed to the absolute distance to threshold however results were extremely similar.

4.1.2 Classification Entropy (H)

Again, as laid in section 2.4, a popular uncertainty measure is the entropy of the probabilistic model output. Here, samples which produce outputs with higher entropy represent samples that the model is uncertain about classifying, or, from the curriculum

perspective, we see as being 'hard' samples. The classification entropy active score function is calculated as follows:

$$P_\theta^H(x_i) = \frac{exp(\frac{S_\theta^H(x_i)}{\tau})}{\sum_j^N exp(\frac{S_\theta^H(x_j)}{\tau})}, \tag{4.3}$$

where

$$S_\theta^H(x_i) = -\sum_c^C P_\theta(y_c|x) \log P_\theta(y_c|x). \tag{4.4}$$

4.1.3 BALD

Recent advances in Bayesian neural networks and variational inference motivate an alternative approach to measuring uncertainty; while the distance to classification threshold may encapsulate classification uncertainty for samples close to the boundary, it does not consider the uncertainty associated with analysing samples from parts of the feature space that is not represented in the training data. Consider the toy example shown in FIGURE, where a sample may be far from the classification boundary, but so dissimilar from the training samples that we would want to measure the sample as having high classification uncertainty. One approach to this problem is given by REF GAL, where they motivate using the *Monte Carlo dropout* method as a way of approximating variational inference in neural networks. As with the usual dropout procedure, weights are randomly set to zero throughout the training phase, however, unlike the usual approach, dropout is maintained at the test stage, and a number of forward passes are carried out, resulting in a distribution of outputs. The resultant distribution can subsequently be analysed to infer which test samples the model is more or less confident in predicting, for example by comparing the variance of the output distributions. In REF GAL ACTIVE LEARNING, the authors use the MC dropout method to construct an active learning acquisition function *Bayesian Active Learning by Disagreement (BALD)*, which queries points which "maximise the mutual information between predictions and model posterior", identifying samples that have a high probability of being placed into different classes in the different stochastic forward passes. One interpration of the BALD method is that it is similar to the 'Query by Committee" active learning methods, with the different forward passes representing different models' votes.

We calculate the BALD active score function as follows:

$$P_\theta^{BALD}(x_i) = \frac{exp(\frac{S_\theta^{BALD}(x_i)}{\tau})}{\sum_j^N exp(\frac{S_\theta^{BALD}(x_j)}{\tau})}, \tag{4.5}$$

where

$$S_\theta^{BALD}(x_i) = -\sum_c^C \bar{P}_\theta(y_c|x_i)\log(\bar{P}_\theta(y_c|x_i)) + \frac{1}{M}\sum_m^M(\sum_c^C P_\theta^m(y_c|x)\log(P_\theta^m(y_c|x))), \tag{4.6}$$

and

$$\bar{P}_\theta(y_c|x_i) = \frac{\sum_m^M P_\theta^m(y_c|x)}{M}. \tag{4.7}$$

Here M is the number of stochastic forward passes carried out and $P_\theta^m(y_c|x)$ is the softmax probability of class c from the m^{th} forward pass. The score can therefore be interpreted as the difference between the entropy of the average softmax output and the average entropy of the output of each forward pass.

4.1.4 Softmax temperature

In order to homogenize the outputs of the different active score functions, we pass the the scores through a softmax functions, resulting in an output of softmax probabilities summing to 1. Using the softmax function also allows us to use the softmax temperature in order to control the diversity of the sampling probabilities. A common issue with active learning is that the acquisition functions can end up sampling from an unrepresentative subset of the input space, resulting in significant bias in the training of the model (REF!). Indeed, GIVE EXAMPLE OF MAX RATIO FOR DIST2THRESH

We control this effect by using the softmax temperature to target a preset *maximum probability ratio*, defined as follows:

$$MaxRatio = \frac{\max_i P_\theta(x_i)}{\min_i P_\theta(x_i)}. \tag{4.8}$$

To do this we begin with a softmax temperature of 1, then calculate the current maximum probability ratio and increment the temperature until the target ratio is achieved. Pseudocode is given below:

PSEUDOCODE FOR SOFTMAX CONTROL

The downside to this method is it could be skewed by outlier probabilities; if there were one sample with an extremely large sampling probability this method would still achieve a target maximum probability ratio, without achieving sufficient diversity in the sampling probabilities. We investigated using winsorization (REF) as an outlier

removal technique prior to tuning the temperature parameter, however, as the active score functions we use generally did not result in outliers, this did not affect results. We show in section REF how controlling the maximum probability ratio affects training.

4.2 Curriculum Construction

4.2.1 Biased Sampling

4.2.2 Uniform Sampled Tasks

Similar to self-paced learning

4.2.3 Biased Sampled Tasks

4.3 Datasets

Description, example, dataset size, source

4.3.1 MNIST

4.3.2 Geometric Shapes

4.3.3 CIFAR 10

4.4 Architectures

Chapter 5

Results

Levy et al. (2009) Y.Bengio and J.Weston (2009)

Chapter 6

Analysis and Discussion

Chapter 7

Conclusion and Further Work

Bibliography

E.L.Allgower and K.Georg (1980). *Numerical continuation methods. An introduction*. Springer-Verlag.

Levy, A., Patel, N., Wang, J., and Zhang, J. (2009). *Modeling Retail Correlations in Credit Portfolios*. Moody's Analytics.

S.Bubeck, N.-B. (2012). Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Machine Learning*, 5:1–122.

Y.Bengio, J.Louradour, R. and J.Weston (2009). Curriculum learning. *Procedures of the International Conference on Machine Learning*, 26:41–18.