Informatics Project Proposal - Automatic Curriculum Learning for Deep Models Using Active Learning

Ian McWilliam s0904776

Abstract

In this paper we propose a project researching how active learning methods can be used to automatically construct training curricula. The aim is to test the hypothesis that training deep models with such 'active curricula' can improve upon other training paradigms such as uniform random sampling, pre-training or traditional curriculum learning. The work will investigate a variety of methodologies for constructing a curriculum using active learning metrics, focusing on how the different training methods affect the performance of convolutional networks on a range of image classification tasks. The output of the project will hopefully be a set of novel ways to improve the training of deep models, as well as a more thorough exploration of the relationship between active and curriculum learning than currently exists in the literature.

1 Introduction

Active Learning

Active learning refers to a learning paradigm wherein machine learning algorithms actively select, or 'query', the samples from which it learns [16]; in the case of deep learning this contrasts with the usual approach of uniformly sampling training instances. The motivation for active learning is that, by allowing it to intelligently select the samples from which it learns, the algorithm can achieve superior generalization performance, from a smaller number of training samples, than if the samples had been chosen randomly [5]. In domains where unlabeled data is abundant but obtaining labels is expensive, active learning can be used to reduce the cost associated with training a deep model, as the active approach allows the designer to obtain labels only for the samples which will be most beneficial to learning.

There are a variety of methods by which the algorithm can query datapoints, however they generally focus on finding the points in the input space that the algorithm is most 'uncertain' about labeling, allowing the algorithm to fill what could be seen as gaps in its knowledge of the domain. We briefly outline a subset of active learning approaches, motivated by [16]:

- Uncertainty Sampling - Directly measuring the algorithm's uncertainty in its prediction; a typical approach would be to estimate the distance to prediction threshold of the algorithm's prediction for each training sample and query the samples closest to the decision boundary. In the case that there are multiple possible labels one can take the average distance to classification threshold across all possible labels, or alternatively choose the samples with the maximum uncertainty across all labels.

- Query by Committee (QBC) - A 'committee' of models are trained on the current training subset, then predict the labels of the remaining training examples. The samples about which there is most disagreement are then queried.

- Expected Model Change - Samples are selected based on their potential to result in the greatest change in the model, if the model was trained with the correct label. In gradient based models this can be implemented by querying the samples that would lead to the largest change in the loss gradient, again this can be averaged or summed across the different possible labels.

We also note the work of Gal et al [7] who use the uncertainty inherent in Bayesian deep models, estimated by approximating variational inference with Monte Carlo dropout methods as in [8].

Curriculum Learning

A related field is that of *curriculum learning*, which explores how the learning process can be improved by presenting training samples to the algorithm in a meaningful order (with the order defining a 'curriculum'), again in contrast to simply sampling uniformly from a training set [2]. Motivated by the way in which humans and animals learn, a curriculum generally begins with 'easy' examples, before transitioning to more challenging ones, with the aim being to improve generalization performance and convergence speed. It is interesting to note that, while similar, active and curriculum learning are somewhat opposed in their learning philosophies, with the former focusing on learning from uncertain/difficult samples and the latter focusing beginning with easy samples before continuing to more difficult ones.

Bengio et al [2] find a theoretical motivation for curriculum learning by suggesting that it can be seen as a *continuation method* (a method of optimization non-convex functions). Curriculum learning has also drawn similarities with *transfer learning*, as the early, 'easier' stages of training can be seen as a separate task, with the network weights then being used as the initial weights for training on more difficult samples. Similarly curriculum learning can be seen as a form of pre-training; Bengio et al [2] also compare the method to unsupervised pre-training methods as in [6], where the authors demonstrate that pre-training allows the supervised learning to begin in a region of parameter space that results in superior generalization performance.

2 Purpose

One of the main difficulties with implementing curriculum learning is that the curriculum must be constructed prior to training, requiring some predefined measure of difficulty with which to order the training samples. In certain domains there may be a natural ordering of difficulty, for example in the geometric shapes example in [2], however in many tasks manually constructing a curriculum is not as straight forward. This motivates the development of methods to automatically construct learning curricula without requiring expert domain knowledge; in this paper we propose using active learning methods to automatically construct such curricula, potentially providing an efficient way to easily improve the training of deep learning algorithms. The purpose of the proposed paper is to test the performance of several such training methods, which we term *active curricula*, comparing their performance to those of several benchmark methods in order to ascertain which methods lead to performance improvements in deep models.

In addition to this, the paper will explore how best to construct a curriculum, using a given active learning method. As discussed in the introduction, active and curriculum learning offer a somewhat dichotomous view on how best to order training samples (for example Kumar et al [14] refer to active learning as being an "anti-curriculum" method). This trade off is one that has not been explored in the literature, and is a dimension on which we will focus in the proposed work.

3 Background

There have been several works that have explored the area of automating the process of constructing learning curricula, as well as similar studies which evaluate methods by which training can be improved by deviating from the usual paradigm of uniform sampling; we introduce a few of these methods here.

Self Paced Learning

In the paper "Self-Paced Learning for Latent Variable Models", M.Pawan Kumar, B. Packar and D.Koller, 2010 [14] introduce the concept of *self-paced learning* wherein, after an initial training period, a learning algorithm is trained beginning with the "easiest" samples. The learning algorithm explored in the paper is the latent SSVM model, as such, easiness in the paper is calculated from how far from the separating hyperplane a training example is, analogous to the classification threshold method discussed previously. The number of samples used for training is decided by difficulty threshold, which is annealed throughout training until all samples are considered, resulting in a learning curriculum that begins with easy samples and progresses to uniformly considering all training examples. The authors test their methods on datasets from a variety of domains, specifically "natural language processing, biology and computer vision", illustrating that their method outperforms a baseline of training on the whole training set. The work is an interesting example of automatically constructing a curriculum but is limited to only considering the latent SSVM model.

Automated Curriculum Learning for Neural Networks

The paper "Automated Curriculum Learning for Neural Networks" by Graves et al, 2017 [10] focusing on automatically constructing curricula for neural networks, specifically deep models. The paper considers the problem wherein learning can be split up into a series of "tasks", with the goal being either to maximise performance on all tasks, or only seeking to maximise performance on a final, "target task". In order to do this the authors use a reinforcement learning approach by modeling an N-task curriculum as an N-arm bandit problem [3], such that the reinforcement learning actions select from which task to sample training examples.

The constructed algorithm is termed "Intrinsically Motivated Curriculum Learning", with the reward signal being calculated from a variety of progress based measures, split into "loss-driven progress" and "complexity-driven progress". Somewhat similar to the previously outlined expected model change, loss-driven progress compares the overall loss of a model before and after training on a sample x (in the paper a sample x refers to a batch of input-target pairs, i.e. it is assumed that all labels are known). The other, more novel, class of reward signals considered are complexity-driven progress signals, which measure changes in the complexity of the network. The rationale behind the complexity methods is derived from the Minimum Description Length (MDL) principle [11], which

suggests that the model complexity should increase most when trained on samples from which it is best able to generalize; training examples that drive the largest increase in model complexity are therefore chosen for training. In order to model the increase the model complexity the authors use several approaches; one is to use stochastic variational inference [13] in order to calculate the change in the KL-Divergence between a posterior over the network weights and a fixed or adaptive prior, defining the Variational complexity gain (VCG):

$$V_{VCG} := KL(P_{\phi'}||Q_{\psi'}) - KL(P_{\phi}||Q_{\psi}) \tag{1}$$

where $P_{\prime\phi}$ and (P_{ϕ} are the variational posterior of the weights before and after training on a sample, and similarly for the prior Q_{ψ}.

The authors test their methods by training LSTM networks on several NLP datasets, with varied results; some methods outperform uniform sampling, however many methods fail to do so across the different datasets. This work represents a similar goal to that of this paper, automatically constructing curricula for deep models, and presents several novel approaches for selecting training samples that could be adapted into active learning acquisition functions. The paper focuses solely on LSTM models however, and is limited to datasets where it is possible to separate different"tasks". The use of reinforcement learning to construct the curriculum is also novel, and it is interesting to note that the final policies generally resulted in an easy to hard curriculum, with the algorithm autonomously discovering implicit orderings in task difficulty and finding that it is best to begin with easier tasks, supporting the curriculum learning rationale in [2].

Active Bias

A recent paper by Chang et al, 2018 [4] conducts similar experiments to the proposed approach, constructing curricula by adapting the sampling probability proportionally to two metrics, specifically classification threshold proximity and prediction variance. They test their methods for training deep models on a variety of datasets, highlighting how their methods can reduce model bias in noisy environments, as illustrated in Figure 2 of [4]. The authors test their training methods on a variety of datasets, including image classification and NLP tasks, showing that consistent out-performance of their methods against a benchmark of uniform random sampling. The authors also explore whether or not selecting 'easier' or 'harder' samples lead to better results, concluding that the answer is task dependent; in easier tasks they find that it is beneficial to train on samples that the model is more uncertain about, and vice versa for more difficult, noisier problems.

The paper is limited in that it considers a relatively limited range of uncertainty metrics and curriculum constriction methods, as well as benchmark comparisons. It would also be interesting to explore how to automatically adjust the preference for harder or easier samples throughout training in order to obtain an optimal solution, as it may be difficult to know beforehand which curriculum method will perform the best.

While these prior works have explored methods for automatically building learning curricula, we believe the work proposed in this paper presents a novel direction for several reasons. The above papers either follow the active learning philosophy of emphasizing 'uncertain'/'difficult' training samples, or the curriculum learning approach of emphasizing 'easier' samples, before moving uniformly sampling; we have not however found a study thoroughly comparing the either approach, nor one that attempts to blend the two approaches for an optimal solution, as we propose to do in this paper. Prior works also generally only compare their methods to relatively simple optimization procedures, for example vanilla SGD or variants thereof. As is explained by Bengio et al in [2], curriculum learning can be compared to pre-training or transfer learning, suggesting that they should also be benchmarks against which the performance of such methods are measured, something which the proposed work will do.

4 Methods

We will test a variety of learning paradigms, each using a training curriculum derived from active learning metrics. Each tested method will therefore require two elements; first, the chosen active learning metric(s), and second, the method in which a curriculum is then constructed using the metrics.

Active Learning Metrics

In active learning, the function used to estimate the model's uncertainty about an input sample is termed the *acquisition* function [16]. We propose to use a variety of acquisition functions to score the training samples, initial experiments will use the following methods:

- Threshold closeness - Average distance to classification threshold across labels.
- Expected model change - Expected change in loss gradient after training on a sample.
- Maximum mean standard deviation - As in [7], using Monte Carlo dropout in order to estimate variational inference, we calculate the standard deviation of the model's outputted label probabilities, averaged across all labels.

In initial experiments we will use these acquisition functions, however, time allowing, we will explore a much broader range of active learning methods.

Curriculum Construction Methods

We will test various approaches for constructing a curriculum from an acquisition function; a simple initial approach will be to construct training batches according to the rank of the uncertainty of each input sample, for example if the batch size is N then the first batch will contain the $1:N$ most uncertain samples, the second batch will consist of the $N+1:2N$ most uncertain samples and so on. In this approach all training samples will be used in every epoch, but in an order defined by the acquisition approach.

Another approach will be to restrict training to only samples with an uncertainty ranking above/-below a certain threshold, with the threshold being annealed throughout training epochs as in [14], until all training samples are considered. Similarly the threshold can be altered in such a way that only the least uncertain samples are considered at the start of training and only the most uncertain towards the end, or vice versa.

The main way in which curricula will be constructed is by adjusting the probability with which training examples are sampled during training. For example, given an active learning acquisition function $\alpha(x_i, f_t)$, which maps the input sample x_i and the the model at time t, f_t to an estimation of the model uncertainty, we can sample training instances proportionally to the model's uncertainty of each sample:

$$\Pr(s_i|f_t) = \frac{\alpha(x_i, f_t)}{\sum_{j=1}^{N} \alpha(x_j, f_t)} \tag{2}$$

Where $\Pr(s_i|f_\tau)$ represents the probability that the training sample x_i is sampled and N is the number of samples in the training set[1].

An alternative method may take the opposite approach and prefer to sample 'easier' examples, about which the model is more certain , in which case we would assign sampling probabilities as:

$$\Pr(s_i|f_t) = \frac{\alpha(x_i, f_t)^{-1}}{\sum_{j=1}^{N} \alpha(x_j, f_t)^{-1}} \tag{3}$$

Again, similar to the annealing approach in [14], we can shift the bias towards more/less uncertain samples throughout training. This can be achieved by setting the sampling probability by

$$\Pr(s_i|f_t) = \frac{\alpha(x_i, f_t)^{\beta_t}}{\sum_{j=1}^{N} \alpha(x_j, f_t)^{\beta_t}} \tag{4}$$

where

$$\beta_t = \frac{2t}{Num\_Epochs} - 1 \tag{5}$$

where Num_Epochs is the number of training epochs. Note that starting with $\beta_1 = -1$ and increasing β_t towards a value of 1 will shift the bias from the least uncertain samples to the most uncertain samples throughout training.

[1] In the case that there is not a pre-constructed training set, and the algorithm is querying synthetic input samples, equation 2 could be estimated with Monte Carlo sampling.

We can also use the acquisition function to alter the weight of the prediction loss of each sample as in [12]. This can serve different purposes; as the training examples are no longer sampled randomly, we introduce bias into the model [4]. By weighting the loss of each sample proportionally to the inverse of the sampling probability, we can correct for this bias; this approach is called *importance sampling*. Concretely, for importance sampling, if we sample from the training data using probabilities as in equation 2, we can performance importance sampling by weighting the loss of sample i by a the weight w_i, defined as follows:

$$w_i \propto Pr(s_i|f_t)^{-1} N_w \tag{6}$$

Where N_w is a normalizing constant which ensures the weights have constant unit average, so as not to alter the global learning rate, as in [4]. Alternatively we can weight the loss of each sample proportionally to the model's uncertainty (as opposed to the inverse of the model's uncertainty, as in importance sampling), in conjunction with altering the sampling probability, i.e.

$$w_i \propto Pr(s_i|f_t) N_w \tag{7}$$

5 Evaluation

Datasets

We will use the different constructed learning methods to train several architectures on a set of image classification tasks, potentially with several different optimization methods (i.e. Stochastic Gradient Descent, AdaGrad, ADAM):

- MNIST - Hand written digit image classification task, taken from `http://yann.lecun.com/exdb/mnist/`.
- CIFAR 10 and 100 - Image classification task, taken from `https://www.cs.toronto.edu/~kriz/cifar.html`.
- Geometric Shapes - Geometric Shapes classification task, taken from `http://www.iro.umontreal.ca/~lisa/twiki/bin/view.cgi/Public/BabyAIShapesDatasets`.

Benchmarks

We compare the constructed training methods to a set of benchmark alternatives -

- Uniform Sampling - Sampling randomly from the training data with a uniform probability for all samples.
- Pre-training - Layer-wise greedy unsupervised pre-training as in [6].
- Pre-constructed curriculum/transfer learning - In the case of the Geometric Shapes database we train with a curriculum as in [2], initially training on the easy 'BasicShapes' dataset, consisting of the same shapes as in 'GeomShapes', but with less variability in the shapes. For the CIFAR 100 dataset we first train a network to classify the images into their 20 super classes, before appending an additional output layer to the network and retraining to classify images into one of the 100 'fine' class labels (i.e. transfer learning).

We will compare the performance of the different training methodologies, measuring the classification error on a held out test set, as well as comparing how quickly the different methods converge.

6 Outputs

The main output of this project will be a comparison of the proposed training methodologies to a range of benchmarks, testing whether learning with a curriculum constructed using active learning methods can reduce generalization error and convergence speeds. Should the methodologies significantly outperform the benchmark training paradigms, it could represent an innovative way to guide the training of deep models without the need to use expert domain knowledge to construct a learning curriculum prior to training. The work will also represent a more thorough experimental exploration

into whether or not it is more beneficial to learn from 'easy' or 'difficult' examples, potentially resulting in an algorithmic approach to identifying which method is best for a given problem so as to automatically construct and optimal learning curriculum.

7 Workplan

References

[1] E.L.Allgower and K.Georg, "Numerical continuation methods. An introduction", *Springer-Verlag*, 1980

[2] Y.Bengio, J.Louradour, R.Collobert and J.Weston, "Curriculum Learning", *Procedures of the International Conference on Machine Learning*, 2009

[3] S.Bubeck, N.Cesa-Bianchi, "Regret Analysis of stochastic and nonstochastic multi-armed bandit problems", *Machine Learning*, Issue 5(1), p1-122, 2012

[4] H-S,Chang, E,Learned-Miller,A.McCallum, "Active Bias: Training More Accuracy Neural Networks by Emphasizing High Variance Samples", *Advances in Neural Information Processing Systems 31*, 2018

[5] D.Cohn, L.Atlas and T.Ladner, "Improving Generalization with Active Learning", *Machine Learning*, Issue 15, p201-221, 1994

[6] D.Erhan, P.A.Manzagol, Y.Bengio, S.Bengio and P.Vincent, "The difficulty of training deep architectures and the effect of unsupervised pre-training", *AI & Statistics*, 2009

[7] Y.Gal, R.Islam, Z.Ghahramani, "Deep Bayesian Active Learning with Image Data", *Advances in Neural Information Processing Systems, Bayesian Deep Learning workshop*, 2016

[8] Y.Gal, R.Islam, Z.Ghahramani, "Dropout as a Bayesian approximation: Representing model uncertainty in deep learning", *Procedures of the International Conference on Machine Learning*, 2016

[9] S.Geman, E.Bienenstock, and R.Doursat, "Neural networks and the bias/variance dilemma", *Neural Computation*, Volume 4, p158, 1992

[10] A.Graves, M.G.Bellemare, J.Menick, R.Munos, and K.Kavukcuoglu. "Automated curriculum learning for neural networks", *Proc. Machine Learning Research*, 70, p1311-1320, 2017

[11] P.D.Grünwald, "The minimum description length principle", *The MIT Press*, 2007

[12] A.Katharopoulos and F.Fleuret, "Not All Samples are Created Equal: Deep Learning with Importance Sampling", arXiv preprint, arXiv:1803.00942, 2018

[13] D.P.Kingma, T.Salimans and M.Welling, "Variational dropout and the local reparameterization trick', *Advances in Neural Information Processing Systems 28*, p2575-2583, 2015

[14] M.Pawan Kumar, B.Packer, D.Koller, "Self-Paced Learning for Latent Variable Models", *Advances in Neural Information Processing Systems 25*, 2010

[15] A.Owen, Y.Zhou, "Safe and effective importance sampling", *Journal of the American Statistical Association*, 95(449), p135-43, 2000

[16] B.Settles, "Active Learning Literature Survey", Computer Sciences Technical Report 1648, 2009

[17] D.Weinshall, G.Cohen, "Curriculum Learning by Transfer Learning: Theory and Experiments with Deep Networks", arXiv preprint, arXiv:1802.03796, 2018