Curriculum Learning

Yoshua Bengio[1] YOSHUA.BENGIO@UMONTREAL.CA
Jérôme Louradour[1,2] JEROMELOURADOUR@GMAIL.COM
Ronan Collobert[3] RONAN@COLLOBERT.COM
Jason Weston[3] JASONW@NEC-LABS.COM
(1) U. MONTREAL, P.O. BOX 6128, MONTREAL, CANADA (2) A2IA SA, 40BIS FABERT, PARIS, FRANCE
(3) NEC LABORATORIES AMERICA, 4 INDEPENDENCE WAY, PRINCETON, NJ, USA

Abstract

Humans and animals learn much better when the examples are not randomly presented but organized in a meaningful order which illustrates gradually more concepts, and gradually more complex ones. Here, we formalize such training strategies in the context of machine learning, and call them "curriculum learning". In the context of recent research studying the difficulty of training in the presence of non-convex training criteria (for deep deterministic and stochastic neural networks), we explore curriculum learning in various set-ups. The experiments show that significant improvements in generalization can be achieved. We hypothesize that curriculum learning has both an effect on the speed of convergence of the training process to a minimum and, in the case of non-convex criteria, on the quality of the local minima obtained: curriculum learning can be seen as a particular form of continuation method (a general strategy for global optimization of non-convex functions).

1. Introduction

Humans need about two decades to be trained as fully functional adults of our society. That training is highly organized, based on an education system and a curriculum which introduces different concepts at different times, exploiting previously learned concepts to ease the learning of new abstractions. By choosing which examples to present and in which order to present them to the learning system, one can *guide* training and remarkably increase the speed at which learning can occur. This idea is routinely exploited in *animal training* where it is called **shaping** (Skinner, 1958; Peterson, 2004; Krueger & Dayan, 2009).

Previous research (Elman, 1993; Rohde & Plaut, 1999; Krueger & Dayan, 2009) at the intersection of cognitive science and machine learning has raised the following question: can machine learning algorithms benefit from a similar training strategy? The idea of training a learning machine with a curriculum can be traced back at least to Elman (1993). The basic idea is to *start small*, learn easier aspects of the task or easier subtasks, and then gradually increase the difficulty level. The experimental results, based on learning a simple grammar with a recurrent network (Elman, 1993), suggested that successful learning of grammatical structure depends, not on innate knowledge of grammar, but on starting with a limited architecture that is at first quite restricted in complexity, but then expands its resources gradually as it learns. Such conclusions are important for developmental psychology, because they illustrate the adaptive value of starting, as human infants do, with a simpler initial state, and then building on that to develop more and more sophisticated representations of structure. Elman (1993) makes the statement that this strategy could make it possible for humans to learn what might otherwise prove to be unlearnable. However, these conclusions have been seriously questioned in Rohde and Plaut (1999). The question of guiding learning of a recurrent neural network for learning a simple language and increasing its capacity along the way was recently revisited from the cognitive perspective (Krueger & Dayan, 2009), providing evidence for faster convergence using a shaping procedure. Similar ideas were also explored in robotics (Sanger, 1994), by gradually making the learning task more difficult.

We want to clarify when and why a curriculum or

Appearing in *Proceedings of the 26th International Conference on Machine Learning*, Montreal, Canada, 2009.

"starting small" strategy can benefit machine learning algorithms. We contribute to this question by showing several cases - involving vision and language tasks - in which very simple multi-stage curriculum strategies give rise to improved generalization and faster convergence. We also contribute to this question with the introduction of a hypothesis which may help to explain some of the advantages of a curriculum strategy. This hypothesis is essentially that a well chosen curriculum strategy can act as a continuation method (Allgower & Georg, 1980), i.e., can help to find better local minima of a non-convex training criterion. In addition, the experiments reported here suggest that (like other strategies recently proposed to train deep deterministic or stochastic neural networks) the curriculum strategies appear on the surface to operate like a regularizer, i.e., their beneficial effect is most pronounced on the test set. Furthermore, experiments on convex criteria also show that a curriculum strategy can speed the convergence of training towards the global minimum.

2. On the difficult optimization problem of training deep neural networks

To test the hypothesis that a curriculum strategy could help to find better local minima of a highly non-convex criterion, we turn our attention to training of deep architectures, which have been shown to involve good solutions in local minima that are almost impossible to find by random initialization (Erhan et al., 2009). Deep learning methods attempt to learn feature hierarchies. Features at higher levels are formed by the composition of lower level features. Automatically learning multiple levels of abstraction may allow a system to induce complex functions mapping the input to the output directly from data, without depending heavily on human-crafted features. A theoretical motivation for deep architectures comes from complexity theory: some functions can be represented compactly with an architecture of depth k, but require an exponential size architecture when the depth is restricted to be less than k (Håstad & Goldmann, 1991; Bengio, 2009). However, training deep architectures involves a potentially intractable non-convex optimization problem (Bengio, 2009), which complicates their analysis. There were no good algorithms for training fully-connected deep architectures before 2006, when Hinton et al. (2006) introduced a learning algorithm that greedily trains one layer at a time. It exploits an unsupervised generative learning algorithm for each layer: a Restricted Boltzmann Machine (RBM) (Freund & Haussler, 1994). It is conceivable that by training each layer one after the other, one first learns the simpler concepts (represented in the first layer), then slightly more abstract concepts (represented in the second layer), etc. Shortly after, strategies for building deep architectures from related variants were proposed (Ranzato et al., 2007; Bengio et al., 2007). These works showed the advantage of deep architectures over shallow ones and of the unsupervised pre-training strategy in a variety of settings. Deep architectures have been applied with success not only in classification tasks (Ranzato et al., 2007; Bengio et al., 2007; Larochelle et al., 2007; Ranzato et al., 2008; Vincent et al., 2008), but also in regression (Salakhutdinov & Hinton, 2008), dimensionality reduction (Hinton & Salakhutdinov, 2006; Salakhutdinov & Hinton, 2007), natural language processing (Collobert & Weston, 2008; Weston et al., 2008), and collaborative filtering (Salakhutdinov et al., 2007).

Nonetheless, training deep architectures is a difficult problem. Erhan et al. (2009) and Larochelle et al. (2007) studied this question experimentally to clarify why deeper networks can sometimes generalize much better and why some strategies such as unsupervised pre-training can make this possible. Erhan et al. (2009) found that unsupervised pre-training makes it possible to start the supervised optimization in a region of parameter space corresponding to solutions that were not much better in terms of final training error but substantially better in terms of test error. This suggested a dual effect of unsupervised pre-training, both in terms of helping optimization (starting in better basins of attraction of the descent procedure in parameter space) and as a kind of regularizer.

The experiments presented here suggest that pre-training with a curriculum strategy might act similarly to unsupervised pre-training, acting both as a way to find better local minima and as a regularizer. They also suggest that they help to reach faster convergence to a minimum of the training criterion.

3. A curriculum as a continuation method

Continuation methods (Allgower & Georg, 1980) are optimization strategies for dealing with minimizing non-convex criteria. Although these global optimization methods provide no guarantee that the global optimum will be obtained, they have been particularly useful in computational chemistry to find approximate solutions of difficult optimization problems involving the configurations of molecules (Coleman & Wu, 1994; Wu, 1997). The basic idea is to first optimize a smoothed objective and gradually consider less smoothing, with the intuition that a smooth version

of the problem reveals the global picture. One defines a single-parameter family of cost functions $C_\lambda(\theta)$ such that C_0 can be optimized easily (maybe convex in θ), while C_1 is the criterion that we actually wish to minimize. One first minimizes $C_0(\theta)$ and then gradually increases λ while keeping θ at a local minimum of $C_\lambda(\theta)$. Typically C_0 is a highly smoothed version of C_1, so that θ gradually moves into the basin of attraction of a dominant (if not global) minimum of C_1. Applying a continuation method to the problem of minimizing a training criterion involves a sequence of training criteria, starting from one that is easier to optimize, and ending with the training criterion of interest.

At an abstract level, a curriculum can also be seen as a sequence of training criteria. Each training criterion in the sequence is associated with a different set of weights on the training examples, or more generally, on a reweighting of the training distribution. Initially, the weights favor "easier" examples, or examples illustrating the simplest concepts, that can be learned most easily. The next training criterion involves a slight change in the weighting of examples that increases the probability of sampling slightly more difficult examples. At the end of the sequence, the reweighting of the examples is uniform and we train on the target training set or the target training distribution.

One way to formalize this idea is the following. Let z be a random variable representing an example for the learner (possibly an (x, y) pair for supervised learning). Let $P(z)$ be the target training distribution from which the learner should ultimately learn a function of interest. Let $0 \leq W_\lambda(z) \leq 1$ be the weight applied to example z at step λ in the curriculum sequence, with $0 \leq \lambda \leq 1$, and $W_1(z) = 1$. The corresponding training distribution at step λ is

$$Q_\lambda(z) \propto W_\lambda(z)P(z) \quad \forall z \tag{1}$$

such that $\int Q_\lambda(z)dz = 1$. Then we have

$$Q_1(z) = P(z) \quad \forall z. \tag{2}$$

Consider a monotonically increasing sequence of λ values, starting from $\lambda = 0$ and ending at $\lambda = 1$.

Definition We call the corresponding sequence of distributions Q_λ (following eqns 1 and 2) a **curriculum** if the *entropy of these distributions increases*

$$H(Q_\lambda) < H(Q_{\lambda+\epsilon}) \quad \forall \epsilon > 0 \tag{3}$$

and $W_\lambda(z)$ is *monotonically increasing in* λ, i.e.,

$$W_{\lambda+\epsilon}(z) \geq W_\lambda(z) \quad \forall z, \forall \epsilon > 0. \tag{4}$$

To illustrate this definition, consider the simple setting where Q_λ is concentrated on a finite set of examples, and increasing λ means adding new examples to that set: the support of Q_λ increases with λ, and the sequence of training distributions corresponds to a sequence of embedded training sets, starting with a small set of easy examples and ending with the target training set. We want the entropy to increase so as to increase the diversity of training examples, and we want the weights of particular examples to increase as they get "added" into the training set.

In the experiments below the sequence of training sets is always discrete. In fact the curriculum strategy worked in some of our experiments with a sequence of just two steps: first a set of easy examples, and then the target training set. At the other extreme, if training proceeds in a stochastic manner by sampling training examples from a distribution, then one could imagine a continuous sequence of sampling distributions which gradually gives more weight $W_\lambda(z)$ to the more difficult examples, until all examples have an equal weight of 1.

Up to now we have not defined what "easy examples" meant, or equivalently, how to sort examples into a sequence that illustrates the simpler concepts first. In the following experiments we explore a few simple ways to define a curriculum, but clearly a lot more work is needed to explore different curriculum strategies, some of which may be very specific to particular tasks.

4. Toy Experiments with a Convex Criterion

4.1. Cleaner Examples May Yield Better Generalization Faster

One simple way in which easy examples could help is by being less "noisy", as shown theoretically (Derényi et al., 1994) in the case of a Teacher-Learner pair of Perceptrons. In the supervised classification setting, an example is considered noisy if it falls on the incorrect side of the decision surface of the Bayes classifier. Noisy examples can slow down convergence, as illustrated with the following toy experiment. Two-dimensional inputs are generated from a different Gaussian for each one of the two classes. We define class targets $y = 1$ and $y = -1$ respectively. The Gaussian mean for class y is at $(y/\sqrt{2}, y/\sqrt{2})$ and both Gaussians have standard deviation 1. Starting from random initial parameters (50 times), we train a linear SVM with 50 training examples. Let w be the weight vector of the Bayes classifier. We find that training only with "easy" examples (for which $yw'x > 0$) gives rise to lower generalization error: 16.3% error vs 17.1% error (average over 50 runs), and the difference is statistically significant.

In principle one could argue that difficult examples can be more informative than easy examples. Here the difficult examples are probably not useful because they confuse the learner rather than help it establish the right location of the decision surface. This experiment does not involve a curriculum strategy yet, but it may help to understand why easier examples could be useful, by avoiding to confuse the learner.

4.2. Introducing Gradually More Difficult Examples Speeds-up Online Training

We train a Perceptron from artificially generated data where the target is $y = \text{sign}(w' x_{\text{relevant}})$ and w is sampled from a Normal(0,1). The training pairs are (x, y) with $x = (x_{\text{relevant}}, x_{\text{irrelevant}})$, i.e., some of the inputs are irrelevant, not predictive of the target class. Relevant inputs are sampled from a Uniform(0,1) distribution. Irrelevant inputs can either be set to 0 or to a Uniform(0,1). The number of irrelevant inputs that is set to 0 varies randomly (uniformly) from example to example, and can be used to sort examples from the easiest (with all irrelevant inputs zeroed out) to the most difficult (with none of the irrelevant inputs zeroed out). Another way to sort examples is by the margin $yw'x$, with easiest examples corresponding to larger values. The learning rate is 1 (it does not matter since there is no margin and the classifier output does not depend on the magnitude of $w'x$ but only on its sign). Initial weights are sampled from a Normal(0,1). We train the Perceptron with 200 examples (i.e., 200 Perceptron updates) and measure generalization error at the end. Figure 1 shows average estimated generalization error measured at the end of training and averaged across 500 repetitions from different initial conditions and different random sampling of training examples. We compare a **no curriculum** setting (random ordering), with a **curriculum** setting in which examples are ordered by easiness, starting with the easiest examples, and two easiness criteria (number of noisy irrelevant inputs, margin $yw'x$). All error rate differences between the curriculum strategy and the no-curriculum are statistically significant (differences of more than .01 were all statistically significant at 5% under a t-test).

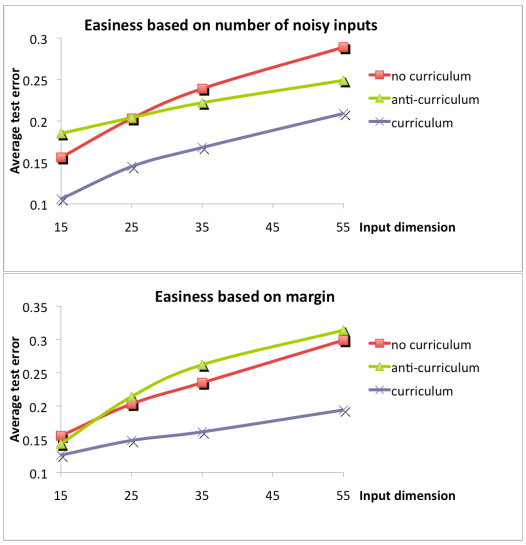

Figure 1. Average error rate of Perceptron, with or without the curriculum. Top: the number of nonzero irrelevant inputs determines easiness. Bottom: the margin $yw'x$ determines easiness.

5. Experiments on shape recognition

The task of interest here is to classify geometrical shapes into 3 classes (rectangle, ellipse, triangle), where the input is a 32×32 grey-scale image. As shown in Figure 2, two different datasets were generated: whereas `GeomShapes` data consist in images of rectangles, ellipses and triangles, `BasicShapes` data only include special cases of the above: squares, circles and equilateral triangles. The difference between `BasicShapes` data and `GeomShapes` data is that `BasicShapes` images exhibit less variability in shape. Other degrees of variability which are present in both sets are the following: object position, size, orientation, and also the grey levels of the foreground and background. Besides, some geometrical constraints are also added so as to ensure that any shape object fits entirely within the image, and a minimum size and minimum contrast (difference in grey levels) between foreground and background is imposed.

Note that the above "easy distribution" occupying a very small volume in input space compared to the target distribution does not contradict condition 4. Indeed, the non-zero weights (on easy examples) can initially be very small, so that their final weight in the target distribution can be very small.

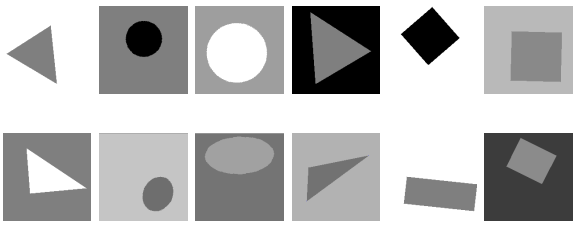

Figure 2. Sample inputs from `BasicShapes` (top) and `GeomShapes` (bottom). Images are shown here with a higher resolution than the actual dataset (32x32 pixels).

The experiments were carried out on a multi-layer neural network with 3 hidden layers, trained by stochas-

tic gradient descent on the negative conditional log-likelihood, i.e., a task which is known to involve a difficult non-convex optimization problem (Erhan et al., 2009). An epoch is a stochastic gradient descent pass through a training set of 10 000 examples. The curriculum consists in a 2-step schedule:

1. Perform gradient descent on the `BasicShapes` training set, until "switch epoch" is reached.
2. Then perform gradient descent on the `GeomShapes` training set.

Generalization error is always evaluted on the `GeomShapes` test set. The baseline corresponds to training the network only on the `GeomShapes` training set (for the same number of training epochs), and corresponds to "switch epoch"=0. In our experiments, there is a total of 10 000 examples in both training sets, and 5 000 examples for validation, 5 000 for testing. All datasets are available at www.iro.umontreal.ca/~lisa/ptwiki/BabyAIShapesDatasets

The hyper-parameters are the following: learning rate of stochastic gradient descent and number of hidden units. The selection of hyper-parameters is simplified using the following heuristic: all hyper-parameters were chosen so as to have the best baseline performance on the `GeomShapes` validation set without curriculum. These hyper-parameter values are then used for the curriculum experiments.

Figure (3) shows the distribution of test errors over 20 different random seeds, for different values of the "switch epoch": 0 (the baseline with no curriculum) and the powers of 2 until 128. After switching to the target training distribution, training continues either until 256 epochs or until validation set error reaches a minimum (early stopping). The figure shows the distribution of test error (after early stopping) as a function of the "switch epoch". Clearly, the best generalization is obtained by doing a 2-stage curriculum where the first half of the total allowed training time (of 256 epochs) is spent on the easier examples rather than on the target examples.

One potential issue with this experiment is that the curriculum-trained model overall saw more examples than the no-curriculum examples, although in the second part of training (with the target distribution) both types of models converge (in the sense of early stopping) to a local minimum with respect to the error on the target training distribution, suggesting that different local minima are obtained. Note also that the easy examples have less variability than the hard examples (only a subset of the shape variations are shown, e.g. only squares instead of all kinds of rectangles). To eliminate the explanation that better results are obtained with the curriculum because of seeing more examples, we trained a no-curriculum model with the union of the `BasicShapes` and `GeomShapes` training sets, with a final test error still significantly worse than with the curriculum (with errors similar to "switch epoch"=16). We also verified that training only with `BasicShapes` yielded poor results.

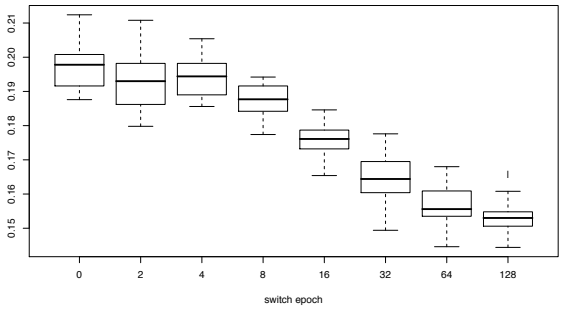

Figure 3. Box plot of test classification error distribution as a function of the "switch epoch", with a *3-hidden-layers neural network* trained by stochastic gradient descent. Each box corresponds to 20 seeds for initializing the parameters. The horizontal line inside the box represents the median (50th percentile), the borders of the box the 25th and the 75th percentile and the ends of the bars the 5th and 95th percentiles.

6. Experiments on language modeling

We are interested here in training a *language model*, predicting the best word which can follow a given context of words in a correct English sentence. Following Collobert and Weston (2008) we only try to compute a score for the next word that will have a large rank compared to the scores of other words, and we compute the score with the architecture of Figure 4. Whereas other language models prior to Collobert and Weston (2008) optimized the log-likelihood of the next word, the ranking approach does not require computing the score over all the vocabulary words during training, as shown below. Instead it is enough to sample a negative example. In Collobert and Weston (2008), the main objective is to learn an embedding for the words as a side-effect of learning to compute this score. The authors showed how to use these embeddings in several language modeling tasks, in a form of multi-task learning, yielding improved results.

Given any fixed size window of text $\boldsymbol{s}$, we consider a language model $f(\boldsymbol{s})$ which produces a score for these windows of text. We want the score of a correct window of text $\boldsymbol{s}$ to be larger, with a margin of 1, than any other word sequence $\boldsymbol{s^w}$ where the last word has been replaced by another word w of the vocabulary. This corresponds to minimizing the expected value of the

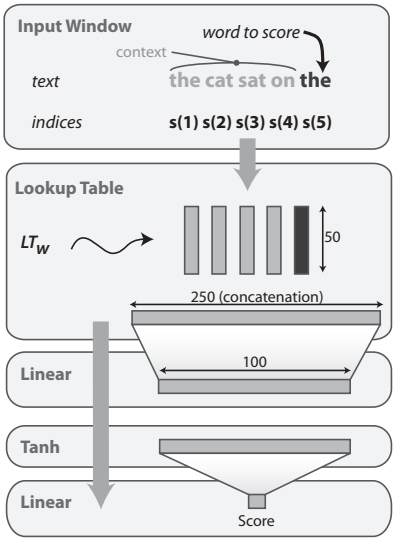

Figure 4. Architecture of the deep neural network computing the score of the next word given the previous ones.

following ranking loss over sequences s sampled from a dataset $\mathcal{S}$ of valid English text windows:

$$C_s = \sum_{w \in \mathcal{D}} \frac{1}{|\mathcal{D}|} C_{s,w} = \sum_{w \in \mathcal{D}} \frac{1}{|\mathcal{D}|} \max(0,\, 1 - f(s) + f(s^w)) \tag{5}$$

where $\mathcal{D}$ is the considered word vocabulary and $\mathcal{S}$ is the set of training word sequences. Note that a stochastic sample of the gradient with respect to C_s can be obtained by sampling a counter-example word w uniformly from $\mathcal{D}$. For each word sequence s we then compute $f(s)$ and $f(s^w)$ and the gradient of $\max(0,\, 1 - f(s) + f(s^w))$ with respect to parameters.

6.1. Architecture

The architecture of our language model (Figure 4) follows the work introduced by Bengio et al. (2001) and Schwenk and Gauvain (2002), and closely resembles the one used in Collobert and Weston (2008). Each word $i \in \mathcal{D}$ is *embedded* into a d-dimensional space using a look-up table $LT_W(\cdot)$: $LT_W(i) = W_i$, where $W \in \mathbb{R}^{d \times |\mathcal{D}|}$ is a matrix of parameters to be learnt, $W_i \in \mathbb{R}^d$ is the i^{th} column of W and d is the embedding dimension hyper-parameter. In the first layer an input window $\{s_1, s_2, \ldots s_n\}$ of n words in $\mathcal{D}$ is thus transformed into a series of vectors $\{W_{s_1}, W_{s_2}, \ldots W_{s_n}\}$ by applying the look-up table to each of its words.

The feature vectors obtained by the look-up table layer are then concatenated and fed to a classical linear layer. A non-linearity (like $\tanh(\cdot)$) follows and the score of the language model is finally obtained after applying another linear layer with one output.

The cost (5) is minimized using stochastic gradient descent, by iteratively sampling pairs $(\boldsymbol{s}, w)$ composed of a window of text $\boldsymbol{s}$ from the training set $\mathcal{S}$ and a random word w, and performing a step in the direction of the gradient of $C_{s,w}$ with respect to the parameters, including the matrix of embeddings W.

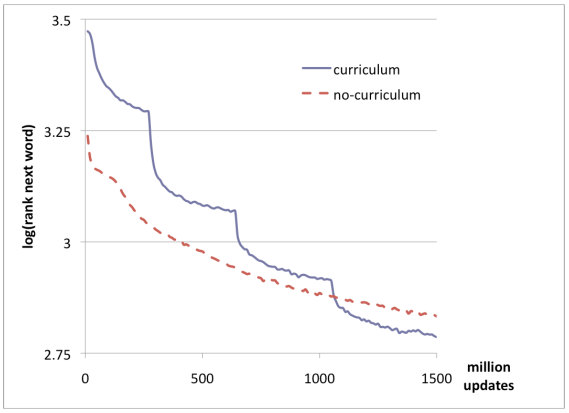

Figure 5. Ranking language model trained with vs without curriculum on Wikipedia. "Error" is log of the rank of the next word (within 20k-word vocabulary). In its first pass through Wikipedia, the curriculum-trained model skips examples with words outside of 5k most frequent words (down to 270 million from 631 million), then skips examples outside 10k most frequent words (doing 370 million updates), etc. The drop in rank occurs when the vocabulary size is increased, as the curriculum-trained model quickly gets better on the new words.

6.2. Experiments

We chose the training set $\mathcal{S}$ as all possible windows of text of size $n = 5$ from Wikipedia (`http://en.wikipedia.org`), obtaining 631 million windows processed as in Collobert and Weston (2008). We chose as a curriculum strategy to grow the vocabulary size: the first pass over Wikipedia was performed using the $5,000$ most frequent words in the vocabulary, which was then increased by $5,000$ words at each subsequent pass through Wikipedia. At each pass, any window of text containing a word not in the considered vocabulary was discarded. The training set is thus increased after each pass through Wikipedia. We compare against no curriculum, where the network is trained using the final desired vocabulary size of $20,000$. The evaluation criterion was the average of the log of the rank of the last word in each test window, taken in a test set of $10,000$ windows of text not seen during the training, with words from the most $20,000$ frequent ones (i.e. from the target distribution). We chose the word embedding dimension to be $d = 50$, and the number of hidden units as 100.

In Figure 5, we observe that the log rank on the target

distribution with the curriculum strategy crosses the error of the no-curriculum strategy after about 1 billion updates, shortly after switching to the target vocabulary size of 20,000 words, and the difference keeps increasing afterwards. The final test set average log-ranks are 2.78 and 2.83 respectively, and the difference is statistically significant.

7. Discussion and Future Work

We started with the following question left from previous cognitive science research (Elman, 1993; Rohde & Plaut, 1999): can machine learning algorithms benefit from a curriculum strategy? Our experimental results in many different settings bring evidence towards a positive answer to that question. It is plausible that some curriculum strategies work better than others, that some are actually useless for some tasks (as in Rohde and Plaut (1999)), and that better results could be obtained on our data sets with more appropriate curriculum strategies. After all, the art of teaching is difficult and humans do not agree among themselves about the order in which concepts should be introduced to pupils.

From the machine learning point of view, once the success of some curriculum strategies has been established, the important questions are: why? and how? This is important to help us devise better curriculum strategies and automate that process to some extent. We proposed a number of hypotheses to explain the potential advantages of a curriculum strategy:

- faster training in the online setting (i.e. faster both from an optimization and statistical point of view) because the learner wastes less time with noisy or harder to predict examples (when it is not ready to incorporate them),
- guiding training towards better regions in parameter space, i.e. into basins of attraction (local minima) of the descent procedure associated with better generalization: a curriculum can be seen as a particular continuation method.

Faster convergence with a curriculum was already observed in (Krueger & Dayan, 2009). However, unlike in our experiments where capacity is fixed throughout the curriculum, they found that compared to using no curriculum, worse results were obtained with fixed neural resources. The reasons for these differences remain to be clarified. In both cases, though, an appropriate curriculum strategy acts to help the training process (faster convergence to better solutions), and we even find that it regularizes, giving rise to lower generalization error for the same training error. This is like in the case of unsupervised pre-training (Erhan et al., 2009), and again it remains to be clarified why one would expect improved generalization, for both curriculum and unsupervised pre-training procedures.

The way we have defined curriculum strategies leaves a lot to be defined by the teacher. It would be nice to understand general principles that make some curriculum strategies work better than others, and this clearly should be the subject of future work on curriculum learning. In particular, to reap the advantages of a curriculum strategy while minimizing the amount of human (teacher) effort involved, it is natural to consider a form of *active selection* of examples similar to what humans (and in particular children) do. At any point during the "education" of a learner, some examples can be considered "too easy" (not helping much to improve the current model), while some examples can be considered "too difficult" (no small change in the model would allow to capture these examples). It would be advantageous for a learner to focus on "interesting" examples, which would be standing near the frontier of the learner's knowledge and abilities, neither too easy nor too hard. Such an approach could be used to at least automate the pace at which a learner would move along a predefined curriculum. In the experiments we performed, that pace was fixed arbitrarily. This kind of strategy is clearly connected to active learning (Cohn et al., 1995), but with a view that is different from the standard one: instead of focusing on the examples near the decision surface to quickly infer its location, we think of the set of examples that the learner succeeds to capture and gradually expand that set by preferentially adding examples near its border.

Curriculum learning is related to boosting algorithms, in that difficult examples are gradually emphasized. However, a curriculum starts with a focus on the easier examples, rather than a uniform distribution over the training set. Furthermore, from the point of view of the boosted weighted sum of weak learners, there is no change in the training criterion: the change is only from the point of view of the next weak learner. As far as the boosted sum is concerned, we are following a functional gradient on the same training criterion (the sum of exponentiated margins). Curriculum strategies are also connected to transfer (or multi-task) learning and lifelong learning (Thrun, 1996). Curriculum learning strategies can be seen as a special form of transfer learning where the initial tasks are used to *guide* the learner so that it will perform better on the final task. Whereas the traditional motivation for multi-task learning is to improve generalization by *sharing across tasks*, curriculum learning adds the notion of *guiding the optimization process*, either to converge faster, or more importantly, to *guide the learner towards better local minima*.

Acknowledgements: The authors thank NSERC, CIFAR, and MITACS for support.

References

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

Bengio, Y. (2009). Learning deep architectures for AI. *Foundations & Trends in Mach. Learn.*, *to appear.*

Bengio, Y., Ducharme, R., & Vincent, P. (2001). A neural probabilistic language model. *Adv. Neural Inf. Proc. Sys. 13* (pp. 932–938).

Bengio, Y., Lamblin, P., Popovici, D., & Larochelle, H. (2007). Greedy layer-wise training of deep networks. *Adv. Neural Inf. Proc. Sys. 19* (pp. 153–160).

Cohn, D., Ghahramani, Z., & Jordan, M. (1995). Active learning with statistical models. *Adv. Neural Inf. Proc. Sys. 7* (pp. 705–712).

Coleman, T., & Wu, Z. (1994). *Parallel continuation-based global optimization for molecular conformation and protein folding* (Technical Report). Cornell University, Dept. of Computer Science.

Collobert, R., & Weston, J. (2008). A unified architecture for natural language processing: Deep neural networks with multitask learning. *Int. Conf. Mach. Learn. 2008* (pp. 160–167).

Derényi, I., Geszti, T., & Györgyi, G. (1994). Generalization in the programed teaching of a perceptron. *Physical Review E*, *50*, 3192–3200.

Elman, J. L. (1993). Learning and development in neural networks: The importance of starting small. *Cognition*, *48*, 781–799.

Erhan, D., Manzagol, P.-A., Bengio, Y., Bengio, S., & Vincent, P. (2009). The difficulty of training deep architectures and the effect of unsupervised pre-training. *AI & Stat.'2009.*

Freund, Y., & Haussler, D. (1994). *Unsupervised learning of distributions on binary vectors using two layer networks* (Technical Report UCSC-CRL-94-25). University of California, Santa Cruz.

Håstad, J., & Goldmann, M. (1991). On the power of small-depth threshold circuits. *Computational Complexity*, *1*, 113–129.

Hinton, G. E., Osindero, S., & Teh, Y.-W. (2006). A fast learning algorithm for deep belief nets. *Neural Computation*, *18*, 1527–1554.

Hinton, G. E., & Salakhutdinov, R. (2006). Reducing the dimensionality of data with neural networks. *Science*, *313*, 504–507.

Krueger, K. A., & Dayan, P. (2009). Flexible shaping: how learning in small steps helps. *Cognition*, *110*, 380–394.

Larochelle, H., Erhan, D., Courville, A., Bergstra, J., & Bengio, Y. (2007). An empirical evaluation of deep architectures on problems with many factors of variation. *Int. Conf. Mach. Learn.* (pp. 473–480).

Peterson, G. B. (2004). A day of great illumination: B. F. Skinner's discovery of shaping. *Journal of the Experimental Analysis of Behavior*, *82*, 317–328.

Ranzato, M., Boureau, Y., & LeCun, Y. (2008). Sparse feature learning for deep belief networks. *Adv. Neural Inf. Proc. Sys. 20* (pp. 1185–1192).

Ranzato, M., Poultney, C., Chopra, S., & LeCun, Y. (2007). Efficient learning of sparse representations with an energy-based model. *Adv. Neural Inf. Proc. Sys. 19* (pp. 1137–1144).

Rohde, D., & Plaut, D. (1999). Language acquisition in the absence of explicit negative evidence: How important is starting small? *Cognition*, *72*, 67–109.

Salakhutdinov, R., & Hinton, G. (2007). Learning a nonlinear embedding by preserving class neighbourhood structure. *AI & Stat.'2007.*

Salakhutdinov, R., & Hinton, G. (2008). Using Deep Belief Nets to learn covariance kernels for Gaussian processes. *Adv. Neural Inf. Proc. Sys. 20* (pp. 1249–1256).

Salakhutdinov, R., Mnih, A., & Hinton, G. (2007). Restricted Boltzmann machines for collaborative filtering. *Int. Conf. Mach. Learn. 2007* (pp. 791–798).

Sanger, T. D. (1994). Neural network learning control of robot manipulators using gradually increasing task difficulty. *IEEE Trans. on Robotics and Automation*, *10.*

Schwenk, H., & Gauvain, J.-L. (2002). Connectionist language modeling for large vocabulary continuous speech recognition. *International Conference on Acoustics, Speech and Signal Processing* (pp. 765–768). Orlando, Florida.

Skinner, B. F. (1958). Reinforcement today. *American Psychologist*, *13*, 94–99.

Thrun, S. (1996). *Explanation-based neural network learning: A lifelong learning approach.* Boston, MA: Kluwer Academic Publishers.

Vincent, P., Larochelle, H., Bengio, Y., & Manzagol, P.-A. (2008). Extracting and composing robust features with denoising autoencoders. *Int. Conf. Mach. Learn.* (pp. 1096–1103).

Weston, J., Ratle, F., & Collobert, R. (2008). Deep learning via semi-supervised embedding. *Int. Conf. Mach. Learn. 2008* (pp. 1168–1175).

Wu, Z. (1997). Global continuation for distance geometry problems. *SIAM Journal of Optimization*, *7*, 814–836.