Curriculum Learning by Transfer Learning: Theory and Experiments with Deep Networks

Daphna Weinshall
School of Computer Science and Engineering
Hebrew University of Jerusalem, Israel
DAPHNA@MAIL.HUJI.AC.IL

Gad Cohen
School of Computer Science and Engineering
Hebrew University of Jerusalem, Israel
GAD.A.COHEN@GMAIL.COM

Abstract

Our first contribution in this paper is a theoretical investigation of curriculum learning in the context of stochastic gradient descent when optimizing the least squares loss function. We prove that the rate of convergence of an ideal curriculum learning method in monotonically increasing with the difficulty of the examples, and that this increase in convergence rate is monotonically decreasing as training proceeds. In our second contribution we analyze curriculum learning in the context of training a CNN for image classification. Here one crucial problem is the means to achieve a curriculum. We describe a method which infers the curriculum by way of transfer learning from another network, pre-trained on a different task. While this approach can only approximate the ideal curriculum, we observe empirically similar behavior to the one predicted by the theory, namely, a significant boost in convergence speed at the beginning of training. When the task is made more difficult, improvement in generalization performance is observed. Finally, curriculum learning exhibits robustness against unfavorable conditions such as strong regularization.

1. Introduction

Biological organisms can learn to perform tasks (and often do) by observing a sequence of labeled events, just like supervised machine learning. But unlike machine learning, in human learning supervision is often accompanied by a curriculum. Thus the order of presented examples is rarely random when a human teacher teaches another human. Likewise, the task may be divided by the teacher into smaller sub-tasks, a process sometimes called shaping (Krueger & Dayan, 2009) and typically studied in the context of reinforcement learning (e.g. Graves et al., 2017). Although remaining in the fringes of the main stream in machine learning, curriculum learning has been identified as a key challenge for machine learning throughout (e.g., Mitchell, 1980; 2006; Wang & Cottrell, 2015).

We focus here on curriculum learning based on ranking (or weighting as in (Bengio et al., 2009)) of the training examples, which is used to guide the order of presentation of examples to the learner. Risking over simplification, the idea is to first present the learner primarily with examples of higher weight or rank, later to be followed by examples with lower weight or rank. Ranking may be based on the difficulty of each training example as evaluated by the teacher, from easiest to the most difficult. We do not delve any deeper into the question of machine teaching, but see (Goldman & Kearns, 1995; Khan et al., 2011).

In Section 2 we investigate this strict definition of curriculum learning theoretically, in the context of the least squares loss function (i.e. regression) and optimization based on stochastic gradient descent. This is often the optimization method of choice for very large regression problems. We first define the difficulty of a training point as its loss with respect to the ideal classifier (the optimal linear regression hypothesis). We then prove that curriculum learning, when given the ranking of training points by their difficulty thus defined, is expected (probabilistically) to significantly speed up learning especially at the beginning of training. This theoretical result is supported by empirical evidence obtained in the deep learning scenario of curriculum learning described in Section 3.2, where similar behavior is observed.

But such ideal ranking is rarely available. In fact, the need for such supervision has rendered curriculum learning less useful in machine learning, since ranking by difficulty is hard to obtain. Moreover, even when it is provided by a human teacher, it may not reflect the true difficulty as it affects the machine learner. For example, in visual object recognition it has been demonstrated that what makes an image difficult to a neural network classifier may not correspond with whatever makes it difficult to a human observer, an observation that has been taken advantage of in the recent

work on adversarial examples (Szegedy et al., 2013). Possibly, this is one of the reasons why curriculum learning is rarely used in practice (but see, e.g., Zaremba & Sutskever, 2014; Amodei et al., 2016; Jesson et al., 2017).

In the second part of this paper we focus on this question - how to rank (or weight) the training examples without the aid of a human teacher. This is paramount when a human teacher cannot provide a reliable difficulty score for the task at hand, or when obtaining such a score by human teachers is too costly. This question is also closely related to transfer learning, as we investigate the use of another classifier to provide the ranking of the training examples by their presumed difficulty. This form of transfer should not be confused with the notion of transfer discussed in (Bengio et al., 2009) in the context of multi-task and life-long learning (Thrun & Pratt, 2012), where knowledge is transferred from earlier tasks (e.g. the discrimination of easy examples) to later tasks (e.g. the discrimination of difficult examples). Rather, we investigate the transfer of knowledge from one classifier to another, as in *teacher classifier* to *student classifier*. In this form curriculum learning has not been studied in the context of deep learning, and hardly ever in the context of other classification paradigms.

There are a number of issues to be addressed in order to design a method for transfer curriculum learning. First, what determines the difficulty of an example? Second, which classifier should be used to compute the curriculum? Should it be a more powerful network pre-trained to do a different task (e.g. Szegedy et al., 2015), or a smaller network that can be quickly trained on the final task? Should we use the same network to continuously rank the training set in order to achieve an adaptive curriculum (as in *self-paced learning:* Kumar et al., 2010; Jiang et al., 2015)? In Section 3 we investigate the first two possibilities, leaving the incorporation of self-paced learning into the scheme for later work. We focus primarily on transfer from a big network pre-trained on a different task. Differently from previous work, it is not the instance representation which is being transferred but rather the ranking of training examples.

Why is this a good idea? This kind of transfer assumes that a powerful pre-trained network is only available at train time, and cannot be used at test time even for the computation of a test point's representation. This may be the case, for example, when the powerful network is too big and therefore inaccessible on simple devices. One can no longer expect to have access to the transferred representation at test time, while ranking can be used at train time in order to improve the learning of the target smaller network (see related discussion of network compression in (Chen et al., 2015; Kim et al., 2015), for example).

Once the training examples are sorted, the next issue to address is how this can be used to aid learning, or in other words, design a scheduling algorithm which benefits from the ranking. In Section 3 we describe our method, an algorithm which uses the ranking to construct a schedule for the order of presentation of training examples. In subsequent empirical evaluations we compare the performance of the method when using a curriculum which is based on different scheduling options, including the control conditions where difficult examples are presented first or arbitrary scheduling.

The main results of the empirical study can be summarized as follows: (i) Learning rate is always faster with curriculum learning, especially at the beginning of training. (ii) Final generalization is sometimes improved with curriculum learning, especially when the conditions for learning are hard: the task is difficult, the network is small, or - in a somewhat equivalent manner - when strong regularization is enforced. This pattern of results is consistent with the prior art mentioned above (see e.g. Bengio et al., 2009).

2. Theoretical analysis: regression

We start with some notations in Section 2.1, followed in Section 2.2 by the rigorous analysis of curriculum learning when used to optimize the regression loss. In Section 2.3 we show supporting empirical evidence for the main theoretical results, obtained using the deep learning setup described later in Section 3.2.

2.1. Notations and definitions

Let $\mathbb{X} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ denote the training data, where $\mathbf{x}_i \in \mathbb{R}^d$ denotes the i-th data point and y_i its corresponding label. In regression problems $y_i \in \mathbb{R}$. The goal is to find a hypothesis $h(\mathbf{x}) \in \mathcal{H}$ that best approximates the unknown generating function $f(\mathbf{x}_i) = y_i$ under the loss $\mathcal{L}(\mathbb{X}, h(\mathbf{x})) = \mathbb{E}_{\mathcal{D}(\mathbf{x})} L(\mathbf{x}, \theta)$, where θ denotes the parameters of $h(\mathbf{x})$.

Our definition of curriculum learning is based on ordering (or ranking) the training examples: it is an iterative learning method which relies on gradual exposure, based on this ranking, of the learner to the training examples. The ranking of examples is based on their relative difficulty, which is defined as follows:

Definition 1 (Difficulty Score)**.** *The difficulty of point* $\mathbf{x}$ *is measured by its loss with respect to an optimal hypothesis* $\bar{h}(\mathbf{x})$*, namely,* $L(\mathbf{x}, \bar{\theta})$.

2.2. Rate of convergence: theory

We investigate stochastic gradfient descent when being used to optimize the regression loss function, and when sampling is guided by the points' *Difficulty Score* as defined above. Specifically, we investigate the differential effect of a point's *Difficulty Score* (henceforth denoted DS) on the

convergence rate of SGD with DS-biased sampling.

In regression problems, the goal is to minimize

$$\begin{aligned}\mathcal{L}(\mathbb{X},\mathbf{w}) &= \frac{1}{n}\sum_{i=1}^{n} L(\mathbf{x}_i,\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n}(h(\mathbf{x}_i)-y_i)^2 \\ &= \frac{1}{n}\sum_{i=1}^{n}(\mathbf{a}^t\mathbf{x}_i+b-y_i)^2 = \frac{1}{n}\|A\mathbf{w}-\mathbf{c}\|^2\end{aligned} \tag{1}$$

In (1), A denotes the $n\times(d+1)$ matrix whose i^{th} row is the vector $[\mathbf{x}_i^t,1]$, $\mathbf{w}=\left[\begin{smallmatrix}\mathbf{a}\\ b\end{smallmatrix}\right]\in\mathbb{R}^{d+1}$, and $\mathbf{c}$ denotes the vector whose i^{th} element is y_i. With some abuse of notation, henceforth $\mathbf{x}_i$ will denote the vector $\left[\begin{smallmatrix}\mathbf{x}_i\\ 1\end{smallmatrix}\right]$.

In general the output hypothesis $h(\mathbf{x})$ is determined by minimizing $\mathcal{L}(\mathbb{X},\mathbf{w})$ with respect to $\mathbf{w}$. Here $\mathcal{L}(\mathbb{X},\mathbf{w})$ is convex, and its global minimum can be computed in closed form from the data as shown in (2). However, gradient descent can be used to find the minimum of this function, which is efficient when n is very large. The convexity of $\mathcal{L}(\mathbb{X},\mathbf{w})$ guarantees that gradient descent will converge to $\bar{\mathbf{w}}$:

$$\bar{\mathbf{w}} = \arg\min_{\mathbf{w}} \mathcal{L}(\mathbb{X},\mathbf{w}) = (A^tA)^{-1}A^t\mathbf{c} \tag{2}$$

The iterative curriculum learning algorithm computes a sequence of estimators $\{\mathbf{w}_t\}_{t=1}^T$ for the parameters of the optimal hypothesis $\bar{\mathbf{w}}$. This is based on a sequence of training points $\{\mathbf{x}_t\}_{t=1}^T$, sampled non uniformly from the training data while favoring easy points at the beginning of training. Other than sampling, the update step at time t follows SGD:

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta\frac{\partial L(\mathbf{x}_t,\mathbf{w})}{\partial\mathbf{w}}|_{\mathbf{w}=\mathbf{w}_t} \tag{3}$$

Since $\mathcal{L}(\mathbb{X},\mathbf{w})$ is convex and achieves its global minimum at $\bar{\mathbf{w}}$, it follows from Def. 1 that the *Difficulty Score* of training point $\mathbf{x}_i$ is determined by $L(\mathbf{x}_i,\bar{\mathbf{w}})$. We analyze the gradient step taken by the algorithm at time t. The main theorem below states that the expected rate of convergence is monotonically decreasing with the *Difficulty Score* of the training points sampled at time t.

Before stating the theorem more precisely, we need a few more definitions and some lemmas. We first derive the gradient step at time t as determined by point $\mathbf{x}_i$:

$$\frac{\partial L(\mathbf{x}_i,\mathbf{w})}{\partial\mathbf{w}} = 2\mathbf{x}_i(\mathbf{x}_i^t\mathbf{w}-y_i) \propto \mathbf{x}_i \tag{4}$$

Let Ω denote the hyperplane on which this gradient vanishes $\frac{\partial L(\mathbf{x}_i,\mathbf{w})}{\partial\mathbf{w}}=0$. This hyperplane is defined by $\mathbf{x}_i^t\mathbf{w}=y_i$, namely $\mathbf{x}_i$ defines its normal direction. Thus (4) implies that the gradient step at time t is perpendicular to Ω as illustrated in Fig. 1.

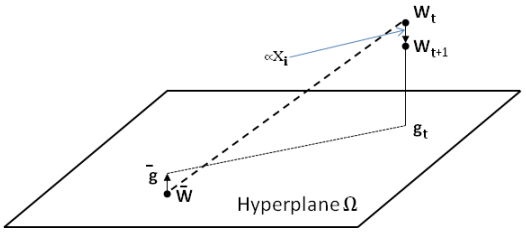

Figure 1. The geometry of the gradient step at time t.

Let Ψ denote the *Difficulty Score* of point $\mathbf{x}_i$. We first show that the distance between $\bar{\mathbf{w}}$ and Ω determines the *Difficulty Score* Ψ of point $\mathbf{x}_i$.

Lemma 1. *Fix the training point $\mathbf{x}_i$, and let $\bar{g}$ denote the projection of $\bar{\mathbf{w}}$ on the corresponding hyperplane Ω. The* Difficulty Score *of $\mathbf{x}_i$ is $\Psi = \|\mathbf{x}_i\|^2\|\bar{\mathbf{w}}-\bar{g}\|^2$.*

Proof.

$$\begin{aligned}\Psi &= L(\mathbf{x}_i,\bar{\mathbf{w}}) = L(\mathbf{x}_i,\bar{\mathbf{g}}+(\bar{\mathbf{w}}-\bar{\mathbf{g}})) \\ &= \|\mathbf{x}_i^t\bar{\mathbf{g}}+\mathbf{x}_i^t(\bar{\mathbf{w}}-\bar{\mathbf{g}})-y_i\|^2 \\ &= \|\mathbf{x}_i^t(\bar{\mathbf{w}}-\bar{\mathbf{g}})\|^2 = \|\mathbf{x}_i\|^2\|\bar{\mathbf{w}}-\bar{\mathbf{g}}\|^2\end{aligned} \tag{5}$$

The first transition in the last line follows from $\bar{\mathbf{g}}\in\Omega \Longrightarrow \mathbf{x}_i^t\bar{\mathbf{g}}-y_i=0$. The second transition follows from the fact that both $\mathbf{x}_i$ and $(\bar{\mathbf{w}}-\bar{\mathbf{g}})$ are perpendicular to Ω, and therefore parallel to each other. □

Recall that $\mathbf{x}_i,\mathbf{w}\in\mathbb{R}^{d+1}$. Next, we shift to a hyperspherical coordinate system in order to describe these vectors, with pole (origin) fixed at $\bar{\mathbf{w}}$ and polar axis (zenith direction) $\vec{\mathcal{O}}=\bar{\mathbf{w}}-\mathbf{w}_t$ (see Fig. 2). Let $0\le\vartheta\le\pi$ denote the polar angle (or elevation) between a vector $\mathbf{x}_i$ and the polar axis $\vec{\mathcal{O}}$ in this coordinate system. Let $\Phi=[\varphi_1\ldots,\varphi_{d-1}]$ denote the remaining angles, and let $r=\|\mathbf{x}_i\|$. The hyperspherical vector representation of $\mathbf{x}_i$ is $[r,\vartheta,\Phi]$.

From (4) the gradient step induced by $\mathbf{x}_i$ at point $\mathbf{w}_t$, whose length is fixed at η, is

$$\mathbf{s} = -sign(\mathbf{x}_i^t\mathbf{w}_t-y_i)\eta\frac{\mathbf{x}_i}{\|\mathbf{x}_i\|} \tag{6}$$

Let $-\pi\le\alpha\le\pi$ denote the angle between $\mathbf{s}$ and the ideal step direction at time t $\vec{\mathcal{O}}$. Let $\lambda=\|\bar{\mathbf{w}}-\mathbf{w}_t\|$. Let the angle $\beta\in[0,\frac{\pi}{2})$ be the following (see Fig. 2)

$$\beta = \beta(r,\Psi,\lambda) = \arccos(\min(\frac{\sqrt{\Psi}}{\lambda r},1)) \tag{7}$$

The second lemma establishes the relationship between ϑ and α. We note in passing that when $|\alpha|>\frac{\pi}{2}$, the gradient step effectively increases the gap between $\mathbf{w}_t$ - the current estimate of $\mathbf{w}$ - and the target $\bar{\mathbf{w}}$ instead of decreasing it.

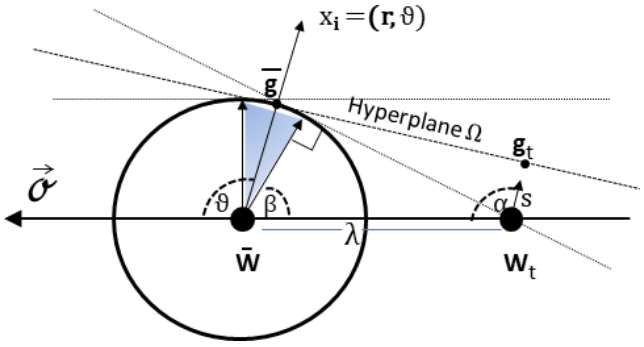

Figure 2. The $2D$ plane defined by the vectors $\vec{\mathcal{O}} = \bar{\mathbf{w}} - \mathbf{w}_t$ and $\bar{\mathbf{g}} - \bar{\mathbf{w}}$. The circle centered on $\bar{\mathbf{w}}$ has radius $\|\bar{\mathbf{w}} - \bar{\mathbf{g}}\| = \frac{\sqrt{\Psi}}{\|\mathbf{x}_i\|}$ from Lemma 1. The highlighted region corresponds to the range of angles ϑ for which $|\alpha| > \frac{\pi}{2}$.

Lemma 2. *With the definitions given above*

- $0 \leq \vartheta \leq \pi - \beta \;\; \Rightarrow \;\; \alpha = \vartheta$
- $\pi - \beta < \vartheta \leq \pi \;\; \Rightarrow \;\; \alpha = \vartheta - \pi$

In particular, $|\alpha| > \frac{\pi}{2}$ *iff* $\frac{\pi}{2} < \vartheta < \pi - \beta$.

Proof. Fig. 2 shows a planar section of the parameter space, the $2D$ plane formed by the two intersecting lines $\vec{\mathcal{O}}$ and $\bar{\mathbf{g}} - \bar{\mathbf{w}}$. Ω is perpendicular to $\mathbf{x}_i$ which is parallel to $\bar{\mathbf{g}} - \bar{\mathbf{w}}$, and is therefore projected onto a line in this plane. Note that by the definition of the hyperspherical coordinate system, we only need to consider the half plane where $0 \leq \vartheta \leq \pi$.

The lemma follows from (6) because when $0 \leq \vartheta \leq \pi - \beta$: $sign(\mathbf{x}_i^t \mathbf{w}_t - y_i) < 0$, while when $\pi - \beta < \vartheta \leq \pi$: $sign(\mathbf{x}_i^t \mathbf{w}_t - y_i) > 0$.

□

In order to proceed in analyzing the effect of the gradient step $\mathbf{s}$, we divide the vector $\mathbf{s}$ to 2 components: $\mathbf{s}_{\mathcal{O}}$ the projection of $\mathbf{s}$ on the polar axis $\vec{\mathcal{O}}$, and $\mathbf{s}_{\perp}$ - the perpendicular component:

$$\mathbf{s} = [\mathbf{s}_{\mathcal{O}}, \mathbf{s}_{\perp}] = [\eta \cos\alpha, \eta \sin\alpha] \tag{8}$$

The last equality follows from (6) and the definition of α.

Let $f(r)f(\vartheta, \Phi)$ denote the probability distribution from which the training points $\{\mathbf{x}_i\}_{i=1}^n$ are sampled. This choice assumes the statistical independence of r and $[\vartheta, \Phi]$. Assume further that this distribution does not depend on Ψ - the DS of $\mathbf{x}_i$ $\forall i$.

Lemma 3. *Let* $\mathcal{S}(\Psi) = \mathbb{E}[\mathbf{s}_{\mathcal{O}}]$ *denote the expected length of the gradient step projected on the polar axis* $\vec{\mathcal{O}}$ *for all training points* $\mathbf{x}_i$ *with* Difficulty Score Ψ. *Then*

$$\mathcal{S}(\Psi) = \int_0^R r^d f(r) \int_0^{\pi-\beta} \eta \cos\vartheta \sin^{d-1}\vartheta \; g(\vartheta) d\vartheta dr - \int_0^R r^d f(r) \int_{\pi-\beta}^{\pi} \eta \cos\vartheta \sin^{d-1}\vartheta \; g(\vartheta) d\vartheta dr$$

where

$$g(\vartheta) = \int_0^{\pi} \dots \int_0^{2\pi} f(\vartheta, \Phi) \tilde{J}(\Phi) d\varphi_1 \dots d\varphi_{d-1} \tag{9}$$

$$\tilde{J}(\Phi) = \sin^{d-2}\varphi_1 \dots \sin\varphi_{d-2} \tag{10}$$

Proof. We first recall that the Jacobian of the transformation from the Cartesian coordinate system in $\mathbb{R}^{d+1}$ to the hyperspherical coordinate system is

$$J(r, \vartheta, \Phi) = r^d \sin^{d-1}\vartheta \sin^{d-2}\varphi_1 \dots \sin^1 \varphi_{d-2} \sin^0 \varphi_{d-1}$$

$$= r^d \sin^{d-1}\vartheta \tilde{J}(\Phi) \tag{11}$$

By definition

$$\mathbb{E}[\eta \cos\alpha] = \int \dots \int \eta \cos\alpha f(r) f(\vartheta, \Phi) J(r, \vartheta, \Phi) dr \; d\vartheta \; d\varphi_1 \dots d\varphi_{d-1}$$

From (11) and using Lemma 2 to substitute ϑ for α in two different ranges, with $\mathbf{s}_{\mathcal{O}} = \eta \cos\alpha$, the lemma follows. □

Finally, let δ denote the convergence rate at time t:

$$\delta = \|\mathbf{w}_t - \bar{\mathbf{w}}\|^2 - \|\mathbf{w}_{t+1} - \bar{\mathbf{w}}\|^2$$

Lemma 4. *At time* t *the expected convergence rate for training points* $\mathbf{x}$ *with* difficulty score *fixed at* Ψ *is:*

$$\mathbb{E}[\delta] = 2\lambda \mathcal{S}(\Psi) - \eta^2 \tag{12}$$

Proof. From (8)

$$\mathbb{E}[\mathbf{s}_{\mathcal{O}}^2] = \eta^2 \mathbb{E}[\cos^2\alpha] = \eta^2 (1 - \mathbb{E}[\sin^2\alpha]) = \eta^2 - \mathbb{E}[\mathbf{s}_{\perp}^2]$$

It follows that

$$\begin{aligned} \mathbb{E}[\delta] &= \lambda^2 - \mathbb{E}[(\lambda - \mathbf{s}_{\mathcal{O}})^2 + \mathbf{s}_{\perp}^2] \\ &= \lambda^2 - (\lambda^2 - 2\lambda \mathbb{E}[\mathbf{s}_{\mathcal{O}}] + \mathbb{E}[\mathbf{s}_{\mathcal{O}}^2]) - \mathbb{E}[\mathbf{s}_{\perp}^2] \\ &= 2\lambda \mathcal{S}(\Psi) - \eta^2 \end{aligned}$$

□

We can now state the main theorem.

Theorem 1. *The expected convergence rate, taken over all training points $\mathbf{x}_i$ with* difficulty score Ψ*, is monotonically decreasing with their Difficulty Score Ψ. It is monotonically increasing with the distance λ between the current estimate of the hypothesis $\mathbf{w}_t$ and the optimal hypothesis $\bar{\mathbf{w}}$ if $\mathcal{S}(\Psi) \geq 0$.*

Proof. From Lemma 4 $\frac{\partial \mathbb{E}[\delta]}{\partial \Psi} = 2\lambda \frac{\partial \mathcal{S}(\Psi)}{\partial \Psi}$, thus we need to show that $\frac{\partial \mathcal{S}(\Psi)}{\partial \Psi} \leq 0$. Using Lemma 3

$$\begin{aligned}\frac{\partial \mathcal{S}(\Psi)}{\partial \Psi} &= \int_0^R \frac{\partial}{\partial \Psi}\left[\int_0^{\pi-\beta} \eta \cos\vartheta \sin^{d-1}\vartheta g(\vartheta) d\vartheta \right. \\ &\left. - \int_{\pi-\beta}^{\pi} \eta \cos\vartheta \sin^{d-1}\vartheta g(\vartheta) d\vartheta\right] r^d f(r) dr \\ &= 2\int_0^R \left[\eta \cos\beta \sin^{d-1}\beta \; g(\pi-\beta)\frac{\partial \beta}{\partial \Psi}\right] r^d f(r) dr\end{aligned}$$

where[1] from (7)

$$\frac{\partial \beta}{\partial \Psi} = \begin{cases} -\frac{1}{\sqrt{1-\frac{\Psi}{(\lambda r)^2}}} \cdot \frac{1}{2\lambda r\sqrt{\Psi}} & r > \frac{\sqrt{\Psi}}{\lambda} \\ 0 & r < \frac{\sqrt{\Psi}}{\lambda} \end{cases} \tag{13}$$

Since $\beta \in [0, \frac{\pi}{2}) \Rightarrow \cos\beta \geq 0$, while from (9) $g(\pi-\beta) \geq 0$ (integration over a positive integrand) and from (13) $\frac{\partial \beta}{\partial \Psi} \leq 0$ with probability 1, it follows that

$$\frac{\partial \mathcal{S}(\Psi)}{\partial \Psi} \leq 0$$

Similarly, from Lemma 4 $\frac{\partial \mathbb{E}[\delta]}{\partial \lambda} = 2\lambda \frac{\partial \mathcal{S}(\Psi)}{\partial \lambda} + 2\mathcal{S}(\Psi)$. If $\mathcal{S}(\Psi) \geq 0$, we only need to show that $\frac{\partial \mathcal{S}(\Psi)}{\partial \lambda} \geq 0$.

$$\frac{\partial \mathcal{S}(\Psi)}{\partial \lambda} = 2\int_0^R \left[\eta \cos\beta \sin^{d-1}\beta g(\pi-\beta)\frac{\partial \beta}{\partial \lambda}\right] r^d f(r) dr$$

where from (7)

$$\frac{\partial \beta}{\partial \lambda} = \begin{cases} \frac{1}{\sqrt{1-\frac{\Psi}{(\lambda r)^2}}} \cdot \frac{\sqrt{\Psi}}{\lambda^2 r} & r > \frac{\sqrt{\Psi}}{\lambda} \\ 0 & r < \frac{\sqrt{\Psi}}{\lambda} \end{cases} \tag{14}$$

Now $\frac{\partial \beta}{\partial \lambda} \geq 0$ with probability 1, and it follows that

$$\frac{\partial \mathcal{S}(\Psi)}{\partial \lambda} \geq 0$$

□

Corollary 1. *Although $\mathbb{E}[\delta]$ may be negative, $\mathbf{w}_t$ converges faster to $\bar{\mathbf{w}}$ when the training points are sampled from easier examples with smaller Ψ.*

[1] We ignore the point $r = \frac{\sqrt{\Psi}}{\lambda}$ where $\beta(r)$ is not differentiable, since its measure is 0.

Assume that the probability distribution of the data points is symmetric with respect to $\vartheta = \frac{\pi}{2}$ so that $f(\vartheta, \Phi) = f(\pi - \vartheta, \Phi)$. It follows from (9) that $g(\vartheta) = g(\pi - \vartheta)$. It can then be shown that

$$\mathcal{S}(\Psi) = 2\int_0^R r^d f(r) \int_0^{\beta} \eta \cos\vartheta \sin^{d-1}\vartheta \; g(\vartheta) d\vartheta dr$$

and therefore $\mathcal{S}(\Psi) \geq 0$ (positive integrand for $\beta \in [0, \frac{\pi}{2})$).

Corollary 2. *If $f(\vartheta, \Phi) = f(\pi - \vartheta, \Phi)$, then the rate of convergence to $\bar{\mathbf{w}}$ is monotonically increasing with the distance λ. This implies that we should expect fsster convergence at the beginning of the optimization.*

2.3. Rate of convergence: deep networks

While the corollaries above apply to a rather simple situation, when using the *Difficulty Score* to guide SGD while minimizing the convex regression loss, their predictions can be empirically tested with the deep learning architecture and loss which are described in Section 3.2. There an additional challenge is posed by the fact that the empirical ranking is not based on the ideal definition in Def. 1, but rather on an estimate derived from another classifier.

Still, the empirical results as shown in Fig. 3 demonstrate agreement with the theoretical analysis of the regression loss. Specifically, in epoch 0 there is a big difference between the average errors in estimating the gradient direction, which is smallest for the easiest examples and highest for the most difficult examples as predicted by Corollary 1. This difference in significantly reduced after 10 epochs, and becomes insignificant after 20 epochs, in accordance with Corollary 2.

Discussion. Since the loss function used to train neural networks is hardly ever convex, there exists the additional question of how to escape unfavorable local minima, a question which has no parallel in the analysis of the convex regression loss. Still, one may speculate regarding the implication of the above results as they concern this question. Specifically, Fig. 3 shows that the variance in the direction of the gradient step defined by easier points is very significantly smaller than that defined by hard points, especially at the beginning of training. If the initial point $\mathbf{w}_0$ does not lie in the basin of attraction of the desired global minimum $\bar{\mathbf{w}}$, and if, in agreement with Lemma 1, the shared component of the easy gradient steps points in the direction of the global minimum, or a more favorable local minimum, then the likelihood of escaping the local minimum decreases with a point's *Difficulty Score*. This suggests another possible advantage for curriculum learning at the initial stages of training.

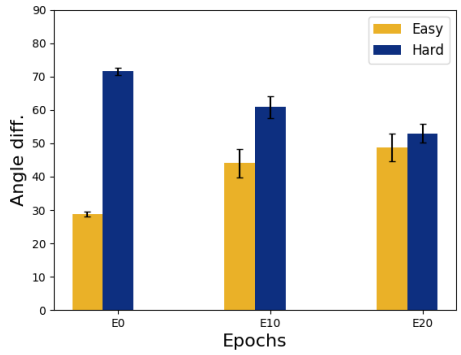

Figure 3. Results using the empirical setup described in Section 3.2. The average angular difference (in degrees) between the gradient computed based on a batch of 100 examples, and the true gradient based on all the training examples, is shown for 2 cases: the easiest examples (yellow) and the most difficult training examples (blue). Standard error bars are plotted based on 100 repetitions. Three Condition are shown: beginning of training (E0), 10 epochs into training (E10), and 20-epochs into training (E20).

3. Curriculum learning in deep networks

As discussed in the introduction, a practical curriculum learning method should address two main challenges: ranking and scheduling of training examples. These challenges are discussed in Section 3.1. In Section 3.2 we discuss the empirical evaluation of our method.

3.1. Method

Ranking examples by knowledge transfer

The main novelty of our proposed method lies in this step, where we rank the training examples by estimated difficulty in the absence of human supervision. Difficulty is estimated based on knowledge transfer from another classifier. We investigated 2 such mechanisms.

Transfer from a more powerful learner. It is a common practice now to treat one of the upstream layers of a pre-trained network as a representation (or embedding) layer. This layer activation is then used for representing similar objects and train a simpler classifier (such as SVM, or shallower NNs) to perform a different task, related but not identical to the original task the network had been trained on. In computer vision such embeddings are commonly obtained by training a deep network on the recognition of a very large database such as ImageNet (Deng et al., 2009). These embeddings have been shown to provide better semantic representations of images (as compared to more traditional image features) in a number of related tasks, including the classification of small datasets (Sharif Razavian et al., 2014), image annotation (Donahue et al., 2015) and structured predictions (Hu et al., 2016).

In this practice, the activation in the penultimate layer of a large and powerful pre-trained network is the loci of knowledge transfer from one network to another. Repeatedly, as in (Sharif Razavian et al., 2014), it has been shown that competitive performance can be obtained by training a shallow classifier on this representation in a new related task. Here we propose to use the confidence of such a classifier, e.g. the margin of an SVM classifier, as the estimator for the difficulty of each training example. This measure is then used to sort the training data. We note that unlike the traditional practice of reusing a pre-trained network, here we only transfer information from one learner to another. The goal is to achieve a smaller classifier that can conceivably be used with simpler hardware, without depending on access to the powerful learner at test time.

Transfer from a weak learner. Alternatively, rough ranking can be obtained from a small weak learner, which can be trained quickly and at a low computational cost in the pre-learning stage. The difficulty of each training datapoint is estimated based on the confidence of the trained weak network in its prediction, which can be evaluated using MC-dropout (Gal & Ghahramani, 2015) for example. The goal is not to achieve competitive results with the weak network, but rather to speed up the learning process of the target network at the cost of a small computational overhead beforehand.

Scheduling the training examples

In agreement with prior art, e.g. the definition of curriculum in (Bengio et al., 2009), we investigated curriculum learning where the scheduling of examples changes with time, giving priority to easier examples at the beginning of training. We explored two variants of the basic scheduling idea:

Fixed. The distribution used to sample examples from the training data is gradually changed in fixed steps. Initially all the weight is put on the easiest examples. In subsequent steps the weight of more difficult example is gradually increased, until the final step in which the training data is sampled uniformly (or based on some prior distribution on the training set).

Adaptive. Similar to the previous mode, but where the length of each step is not fixed, but is being determined adaptively based on the current loss of the training data.

3.2. Empirical evaluation

Experimental setup

Datasets. For evaluation we used 2 data sets: CIFAR-100 (Krizhevsky & Hinton, 2009) and STL-10 (Coates et al., 2010). In all cases, as is commonly done, the data was

pre-processed using global contrast normalization; cropping and flipping were used for STL-10.

network architecture. We used Convolution Neural Networks (CNN) which excel at image classification tasks. More specifically, we used two architectures - *Large* and *Small*, in accordance with the number of parameters. The *Large* network is comprised of four blocks, each with two convolution layers, ELU activation, and max-pooling. This is followed by a fully connected layer, for a total of 1,208,101 parameters. The *Small* network consists of only three hidden layers, for a total of 4,557 parameters. During training, we applied dropout and $l2$ regularization on the weights, and used either SGD or ADAM to optimize the cross-entropy loss.

Scheduling mechanisms: control. As described above, our method is based on a scheduling mechanism which favors the presentation of easier examples at the beginning of training. In order to isolate the contribution of this idea as against other spurious consequences of data scheduling, we compared performance with the following control conditions: `control-curriculum`, identical scheduling mechanism where the underlying ranking of the training examples is random and unrelated to estimated difficulty; and `anti-curriculum`, identical scheduling mechanism but favoring the more difficult examples at the beginning of training.

Controlling for Task difficulty

Evidence from prior art is conflicting regarding where the benefits of curriculum learning lie, which is to be expected given the variability in the unknown sources of the curriculum supervision information and its quality. We observed in our empirical study that the benefits depended to a large extent on the difficulty of the task. We always saw faster learning in the beginning of the training process, while lower generalization error was seen only when the task is relatively difficult. We therefore added controls for the following 3 sources of task difficulty:

Inherent task difficulty. To investigate this factor, we took advantage of the fact that CIFAR-100 is a hierarchical dataset with 100 classes and 20 super-classes, each including 5 member classes. We therefore trained a network to discriminate the 5 member classes of 2 super-classes as 2 separate tasks: "small mammals" (task 1) and "aquatic mammals" (task 2). These are expected to be relatively hard learning tasks. We also trained a network to discriminate 5 random well separated classes: camel, clock, bus, dolphin, orchid. This task (task 3) is expected to be easier.

Size of classification network. For a given task, classification performance is significantly affected by the size of the network and its architecture. We assume, of course, that we operate in the domain where the number of model parameters is smaller than can be justified by the training data (i.e., there is no overfit). We therefore used networks of different sizes in order to evaluate how curriculum learning is affected by *task difficulty* as determined by the network's strength (see Fig. 4a-b). In this domain, the smaller the network is, the more difficult the task is likely to be (clearly, many other factors participate in the determination of task difficulty).

Regularization and optimization. Regularization is used to constrain the family of hypotheses, or models, so that they possess such desirable properties as smoothness. Regularization effectively decreases the number of degrees of freedom in the model. In fact, most optimization methods, other then vanilla stochastic gradient descent, incorporate some form of regularization and smoothing, among other inherent properties. Therefore the selection of optimization method also plays a role in determining the effective size of the final network. Below we evaluate the effectiveness of our method as it depends on the amount of regularization and the optimization method (see Fig. 5).

Results

Fig. 4a shows typical results when training the *Large* CNN (see network's details above) to classify a subset of 5 CIFAR100 images (task 1 defined in Section 3.2), using a slow learning rate and Adam optimization. In this setup we see that curriculum learning speeds up the learning rate at the beginning of the training, but converges to the same performance as regular training. When we make learning more difficulty by using the *Small* network, performance naturally decreases, but now we see that curriculum learning also improves the final generalization performance (Fig. 4b). Similar results are shown for the STL-10 dataset (Fig. 4c).

Fig. 6 shows comparative results when controlling for inherent task difficulty in 3 tasks as described in Section 3.2, using a faster learning rate and SGD optimization. Task difficulty can be evaluated in retrospect from the final performance seen in each plot. As can be clearly seen in the figure, the improvement in final accuracy with curriculum learning is larger when the task is more difficult.

Fig. 5 shows comparative results when manipulating the level of regularization. Too much regularization harms performance (Fig. 5b-c), but curriculum learning shows robustness and seems least affected by this degradation.

4. Summary and Discussion

We investigated curriculum learning where the presentation of easy examples is given preference at the beginning of training. We started with the theoretical investigation of this strict definition in the context of the least squares loss

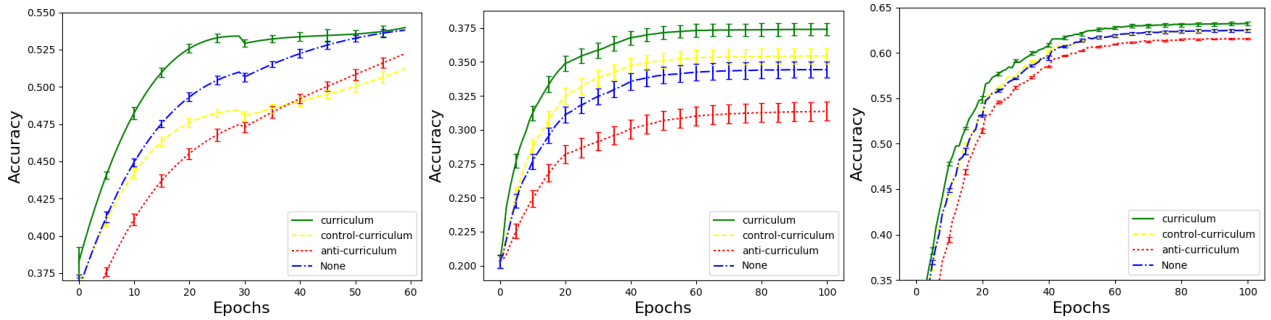

Figure 4. Accuracy in test classification as a function of training time. We show results with 4 scheduling methods: our method denoted `curriculum` (solid green), `control-curriculum` with random ordering (dashed yellow), `anti-curriculum` where more difficult examples are preferred at the beginning of training (dotted red), and `none` with no curriculum learning (dashdotted blue). a-b) Learning CIFAR100 task 1, where the *Large* network is used in a) and the *Small* network in b). c) Learning to classify STL-10 images.

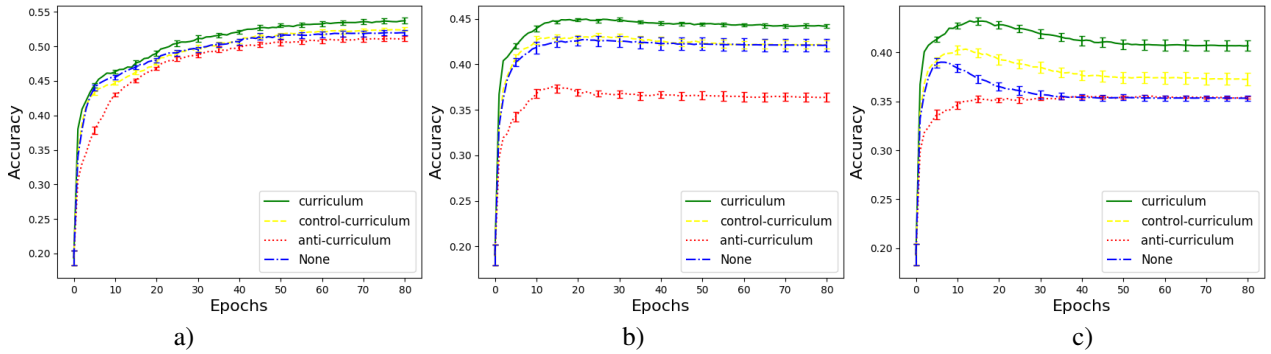

Figure 5. Accuracy in test classification as a function of training time, as seen in the learning of task 1. Panels are organized by level of regularization, from low (a) to high and most detrimental (c). 4 scheduling methods are shown as described in the caption of Fig. 4.

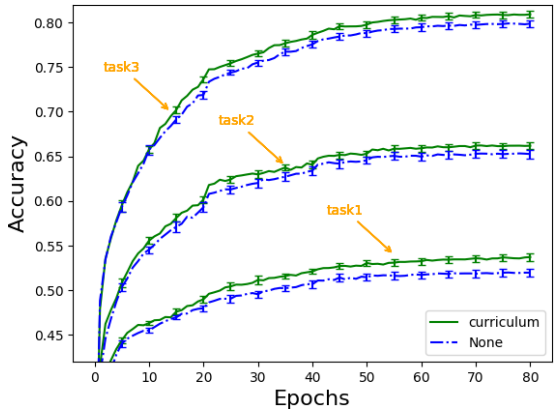

Figure 6. Accuracy in test classification in 3 tasks each involving 5 classes from the CIFAR100 datasets (see Section 3.2). In each task, the performance of learning with a curriculum (solid green) and without (dashdotted blue) is shown.

function, showing that curriculum learning accelerates the learning rate in agreement with prior empirical evidence. While not shedding light on its affect on the classifier's final performance, our analysis suggests that the direction of a gradient step based on "easy" examples may be more effective in traversing the input space towards the ideal minimum of the loss function. Under some circumstances this may increase the likelihood to escape the basin of attraction of a low quality local minimum in favor of higher quality local minima even in the general case with a non-convex loss function.

In the second part of this paper we described a curriculum learning method for deep networks based on stochastic gradient descent. The method relies on knowledge transfer from other (pre-trained) networks in order to rank the training examples by difficulty. We described extensive experiments where we evaluated our proposed method under different task difficulty conditions and against a variety of control conditions. In all cases curriculum learning has been shown to increase the rate of convergence at the beginning of training, in agreement with the theoretical results. With more difficult tasks, curriculum learning has been shown to improve generalization performance (see Fig. 6).

Acknowledgements

This work was supported in part by a grant from the Israel Science Foundation (ISF) and by the Gatsby Charitable Foundations.

References

Amodei, Dario, Ananthanarayanan, Sundaram, Anubhai, Rishita, Bai, Jingliang, Battenberg, Eric, Case, Carl, Casper, Jared, Catanzaro, Bryan, Cheng, Qiang, Chen, Guoliang, et al. Deep speech 2: End-to-end speech recognition in english and mandarin. In *International Conference on Machine Learning*, pp. 173–182, 2016.

Bengio, Yoshua, Louradour, Jérôme, Collobert, Ronan, and Weston, Jason. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pp. 41–48. ACM, 2009.

Chen, Wenlin, Wilson, James, Tyree, Stephen, Weinberger, Kilian, and Chen, Yixin. Compressing neural networks with the hashing trick. In *International Conference on Machine Learning*, pp. 2285–2294, 2015.

Coates, Adam, Lee, Honglak, and Ng, Andrew Y. An analysis of single-layer networks in unsupervised feature learning. *Ann Arbor*, 1001(48109):2, 2010.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

Donahue, Jeffrey, Anne Hendricks, Lisa, Guadarrama, Sergio, Rohrbach, Marcus, Venugopalan, Subhashini, Saenko, Kate, and Darrell, Trevor. Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2625–2634, 2015.

Gal, Yarin and Ghahramani, Zoubin. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. *arXiv preprint arXiv:1506.02142*, 2, 2015.

Goldman, Sally A and Kearns, Michael J. On the complexity of teaching. *Journal of Computer and System Sciences*, 50(1):20–31, 1995.

Graves, Alex, Bellemare, Marc G, Menick, Jacob, Munos, Remi, and Kavukcuoglu, Koray. Automated curriculum learning for neural networks. *arXiv preprint arXiv:1704.03003*, 2017.

Hu, Hexiang, Zhou, Guang-Tong, Deng, Zhiwei, Liao, Zicheng, and Mori, Greg. Learning structured inference neural networks with label relations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2960–2968, 2016.

Jesson, Andrew, Guizard, Nicolas, Ghalehjegh, Sina Hamidi, Goblot, Damien, Soudan, Florian, and Chapados, Nicolas. Cased: Curriculum adaptive sampling for extreme data imbalance. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pp. 639–646. Springer, 2017.

Jiang, Lu, Meng, Deyu, Zhao, Qian, Shan, Shiguang, and Hauptmann, Alexander G. Self-paced curriculum learning. In *AAAI*, volume 2, pp. 6, 2015.

Khan, Faisal, Mutlu, Bilge, and Zhu, Xiaojin. How do humans teach: On curriculum learning and teaching dimension. In *Advances in Neural Information Processing Systems*, pp. 1449–1457, 2011.

Kim, Yong-Deok, Park, Eunhyeok, Yoo, Sungjoo, Choi, Taelim, Yang, Lu, and Shin, Dongjun. Compression of deep convolutional neural networks for fast and low power mobile applications. *arXiv preprint arXiv:1511.06530*, 2015.

Krizhevsky, Alex and Hinton, Geoffrey. Learning multiple layers of features from tiny images. 2009.

Krueger, Kai A and Dayan, Peter. Flexible shaping: How learning in small steps helps. *Cognition*, 110(3):380–394, 2009.

Kumar, M Pawan, Packer, Benjamin, and Koller, Daphne. Self-paced learning for latent variable models. In *Advances in Neural Information Processing Systems*, pp. 1189–1197, 2010.

Mitchell, Tom M. *The need for biases in learning generalizations*. Department of Computer Science, Laboratory for Computer Science Research, Rutgers Univ. New Jersey, 1980.

Mitchell, Tom Michael. *The discipline of machine learning*, volume 9. Carnegie Mellon University, School of Computer Science, Machine Learning Department, 2006.

Sharif Razavian, Ali, Azizpour, Hossein, Sullivan, Josephine, and Carlsson, Stefan. Cnn features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 806–813, 2014.

Szegedy, Christian, Zaremba, Wojciech, Sutskever, Ilya, Bruna, Joan, Erhan, Dumitru, Goodfellow, Ian, and Fergus, Rob. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

Szegedy, Christian, Liu, Wei, Jia, Yangqing, Sermanet, Pierre, Reed, Scott, Anguelov, Dragomir, Erhan, Dumitru, Vanhoucke, Vincent, Rabinovich, Andrew, Rick Chang,

Jen-Hao, et al. Going deeper with convolutions. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

Thrun, Sebastian and Pratt, Lorien. *Learning to learn*. Springer Science & Business Media, 2012.

Wang, Panqu and Cottrell, Garrison W. Basic level categorization facilitates visual object recognition. *arXiv preprint arXiv:1511.04103*, 2015.

Zaremba, Wojciech and Sutskever, Ilya. Learning to execute. *arXiv preprint arXiv:1410.4615*, 2014.