Automated Curriculum Learning for Neural Networks

Alex Graves, Marc G. Bellemare, Jacob Menick, Rémi Munos, Koray Kavukcuoglu
{gravesa, bellemare, jmenick, munos, korayk}@google.com
Google DeepMind, London UK

Abstract

We introduce a method for automatically selecting the path, or syllabus, that a neural network follows through a curriculum so as to maximise learning efficiency. A measure of the amount that the network learns from each data sample is provided as a reward signal to a nonstationary multi-armed bandit algorithm, which then determines a stochastic syllabus. We consider a range of signals derived from two distinct indicators of learning progress: rate of increase in prediction accuracy, and rate of increase in network complexity. Experimental results for LSTM networks on three curricula demonstrate that our approach can significantly accelerate learning, in some cases halving the time required to attain a satisfactory performance level.

1. Introduction

Over two decades ago, in *The importance of starting small*, Elman put forward the idea that a curriculum of progressively harder tasks could significantly accelerate a neural network's training (Elman, 1993). However curriculum learning has only recently become prevalent in the field (e.g., Bengio et al., 2009), due in part to the greater complexity of problems now being considered. In particular, recent work on learning programs with neural networks has relied on curricula to scale up to longer or more complicated programs (Sutskever and Zaremba, 2014; Reed and de Freitas, 2015; Graves et al., 2016). We expect this trend to continue as the scope of neural networks widens.

One reason for the slow adoption of curriculum learning is that its effectiveness is highly sensitive to the mode of progression through the tasks. One popular approach is to define a hand-chosen performance threshold for advancement to the next task, along with a fixed probability of returning to earlier tasks, to prevent forgetting (Sutskever and Zaremba, 2014). However, as well as introducing hard-to-tune parameters, this poses problems for curricula where appropriate thresholds may be unknown or variable across tasks. More fundamentally, it presupposes that the tasks can be ordered by difficulty, when in reality they may vary along multiple axes of difficulty, or have no predefined order at all.

We propose to instead treat the decision about which task to study next as a stochastic policy, continuously adapted to optimise some notion of what Oudeyer et al. (2007) termed *learning progress*. Doing so brings us into contact with the intrinsic motivation literature (Barto, 2013), where various indicators of learning progress have been used as reward signals to encourage exploration, including compression progress (Schmidhuber, 1991), information acquisition (Storck et al., 1995), Bayesian surprise (Itti and Baldi, 2009), prediction gain (Bellemare et al., 2016) and variational information maximisation (Houthooft et al., 2016). We focus on variants of prediction gain, and also introduce a novel class of progress signals which we refer to as complexity gain. Derived from minimum description length principles, complexity gain equates acquisition of knowledge with an increase in effective information encoded in the network weights.

Given a progress signal that can be evaluated for each training example, we use a multi-armed bandit algorithm to find a stochastic policy over the tasks that maximises overall progress. The bandit is nonstationary because the behaviour of the network, and hence the optimal policy, evolves during training. We take inspiration from a previous work that modelled an adaptive student with a multi-armed bandit in the context of developmental learning (Lopes and Oudeyer, 2012). Another related area is the field of active learning, where similar gain signals have been used to guide decisions about which data point to label next (Settles, 2010). Lastly, there are parallels with recent work on using Bayesian optimisation to find the best order in which to train a word embedding network on a language corpus (Tsvetkov, 2016); however this differs from our work in that the ordering was entirely determined *be-*

fore each training run, rather than adaptively altered in response to the model's progress.

2. Background

We consider supervised or unsupervised learning problems where target sequences $\mathbf{b}^1, \mathbf{b}^2, \ldots$ are conditionally modelled given their respective input sequences $\mathbf{a}^1, \mathbf{a}^2, \ldots$. For convenience we suppose that the targets are drawn from a finite set $\mathcal{B}$, noting our framework extends to continuous targets, with densities taking the place of probabilities. As is typical for neural networks, sequences may be grouped together in batches $(\mathbf{b}^{1:B}, \mathbf{a}^{1:B})$ to accelerate training. The conditional probability output by the model is

$$p(\mathbf{b}^{1:B} \,|\, \mathbf{a}^{1:B}) = \prod_{i,j} p(\mathbf{b}^i_j \,|\, \mathbf{b}^i_{1:j-1}, \mathbf{a}^i_{1:j-1}).$$

From here onwards, we consider each batch as a single example $\mathbf{x}$ from $\mathcal{X} := (\mathcal{A} \times \mathcal{B})^N$, and write $p(\mathbf{x}) := p(\mathbf{b}^{1:B} \,|\, \mathbf{a}^{1:B})$ for its probability. Under this notation, a *task* is a distribution D over sequences from $\mathcal{X}$. A *curriculum* is an ensemble of tasks $D_1, \ldots, D_N$, and a *sample* is an example drawn from one of the tasks of the curriculum. Finally, a *syllabus* is a time-varying sequence of distributions over tasks.

We consider a neural network to be a parametric probabilistic model p_θ over $\mathcal{X}$, whose parameters are denoted θ. The expected loss of the network on the k^{th} task is

$$\mathcal{L}_k(\theta) := \mathop{\mathbb{E}}_{\mathbf{x} \sim D_k} L(\mathbf{x}, \theta),$$

where $L(\mathbf{x}, \theta) := -\log p_\theta(\mathbf{x})$ is the sample loss on $\mathbf{x}$. Whenever unambiguous, we will simply denote the expected and sample losses by $\mathcal{L}_k$ and $L(\mathbf{x})$ respectively.

2.1. Curriculum Learning

We consider two related settings. In the *multiple tasks* setting, The goal is to perform as well as possible on all tasks in the ensemble $\{D_k\}$; this is captured by the objective function

$$\mathcal{L}_{\text{MT}} := \frac{1}{N} \sum_{k=1}^{N} \mathcal{L}_k.$$

In the *target task* setting, we are only interested in minimizing the loss on the final task D_N. The other tasks then act as a series of stepping stones to the real problem. The objective function in this setting is simply $\mathcal{L}_{\text{TT}} := \mathcal{L}_N$.

2.2. Adversarial Multi-Armed Bandits

We view a curriculum containing N tasks as an N-armed bandit (Bubeck and Cesa-Bianchi, 2012), and a syllabus as an adaptive policy which seeks to maximize payoffs from this bandit. In the bandit setting, an agent selects a sequence of arms (actions) $a_1 \ldots a_T$ over T rounds of play ($a_t \in \{1, \ldots, N\}$). After each round, the selected arm yields a payoff r_t; the payoffs for the other arms are not observed.

The classic algorithm for adversarial bandits is Exp3 (Auer et al., 2002), which uses multiplicative weight updates to guarantee low regret with respect to the best arm. On round t, the agent selects an arm stochastically according to a policy π_t. This policy is defined by a set of weights $w_{t,i}$:

$$\pi_t^{\text{EXP3}}(i) := \frac{e^{w_{t,i}}}{\sum_{j=1}^{N} e^{w_{t,j}}}.$$

The weights are the sum of importance-sampled rewards:

$$w_{t,i} := \eta \sum_{s<t} \tilde{r}_{s,i} \qquad \tilde{r}_{s,i} := \frac{r_s \mathbb{I}_{[a_s=i]}}{\pi_s(i)}.$$

Exp3 acts so as to minimize regret with respect to the single best arm evaluated over the whole history. However, a common occurrence is for an arm to be optimal for a portion of the history, then another arm, and so on; the best strategy is then piecewise stationary. This is generally the case in our setting, as the expected reward for each task changes as the model learns. The Fixed Share method (Herbster and Warmuth, 1998) addresses this issue by using an ϵ-greedy strategy and mixing in the weights additively. In the bandit setting, this is known as the Exp3.S algorithm (also by Auer et al. (2002)):

$$\begin{aligned} \pi_t^{\text{EXP3.P}}(i) &:= (1-\epsilon)\pi_t^{\text{EXP3}}(i) + \frac{\epsilon}{N} \qquad (1) \\ w^{\text{S}}_{t,i} &:= \log\Big[(1-\alpha_t)\exp\Big\{w^{\text{S}}_{t-1,i} + \eta\tilde{r}^{\beta}_{t-1,i}\Big\} \\ &\quad + \frac{\alpha_t}{N-1}\sum_{j\neq i}\exp\Big\{w^{\text{S}}_{t-1,j} + \eta\tilde{r}^{\beta}_{t-1,j}\Big\}\Big] \\ w^{\text{S}}_{1,i} &:= 0 \qquad \alpha_t := t^{-1} \qquad \tilde{r}^{\beta}_{s,i} := \frac{r_s\mathbb{I}_{[a_s=i]} + \beta}{\pi_s(i)}. \end{aligned}$$

2.3. Reward Scaling

The appropriate step size η depends on the magnitudes of the rewards, which may not be known *a priori*. The problem is particularly acute in our setting, where the magnitude depends on how learning progress is measured, and varies over time as the model learns. To address this issue, we adaptively rescale all rewards to lie in the interval $[-1, 1]$ using the following procedure: Let $\mathcal{R}_t$ be the history of unscaled rewards up to time t, i.e. $\mathcal{R}_t = \{\hat{r}_i\}_{i=1}^{t-1}$. Let q_t^{lo} and q_t^{hi} be quantiles of $\mathcal{R}_t$, which we choose here to be the 20$^{\text{th}}$ and 80$^{\text{th}}$ percentiles respectively. The scaled reward r_t is obtained by clipping $\hat{r}_t$ to the interval $[q_t^{\text{lo}}, q^{\text{hi}_t}]$

and then linearly mapping the result to lie in $[-1, 1]$:

$$r_t = \begin{cases} -1 & \text{if } \hat{r}_t < q_t^{\text{lo}} \\ 1 & \text{if } \hat{r}_t > q_t^{\text{hi}} \\ \frac{2(\hat{r}_t - q_t^{\text{lo}})}{q_t^{\text{hi}} - q_t^{\text{lo}}} - 1 & \text{otherwise.} \end{cases} \quad (2)$$

Rather than keeping the entire history of rewards, we use reservoir sampling to maintain a representative sample, and compute approximate quantiles from this sample. These quantiles can be obtained in $\Theta(log|\mathcal{R}_t|)$ time.

3. Learning Progress Signals

Our goal is to use the policy output by Exp3.S as a syllabus for training our models. Ideally we would like the policy to maximize the rate at which we minimize the loss, and the reward should reflect this rate – what Oudeyer et al. (2007) calls *learning progress*. However, it usually is computationally undesirable or even impossible to measure the effect of a training sample on the target objective, and we therefore turn to surrogate measures of progress. Broadly, these measures are either 1) loss-driven, in the sense that they equate reward with a decrease in some loss; or 2) complexity-driven, when they equate reward with an increase in model complexity.

Training proceeds as follows: at each time t, we first sample a task index $k \sim \pi_t$. We then generate a sample from this task, i.e. $\mathbf{x} \sim D_k$. Note that each $\mathbf{x}$ is in general a batch of training sequences, and that in order to reduce noise in the gain signal we draw the whole batch from a single task. We compute the chosen measure of learning progress ν then divide by the time $\tau(\mathbf{x})$ required to process the sample (since it is the *rate* of progress we are concerned with, and processing time may vary from task to task) to get the raw reward $\hat{r} = \nu/\tau(\mathbf{x})$ For the purposes of this work, $\tau(\mathbf{x})$ was simply the length of the longest input sequence in $\mathbf{x}$; for other tasks or architectures a more complex calculation may be required. We then rescale $\hat{r}$ into a reward $r_t \in [-1, 1]$, and provide it to Exp3.S. The procedure is summarized as Algorithm 1.

3.1. Loss-driven Progress

We consider five loss-driven progress signals, all which compare the predictions made by the model before and after training on some sample $\mathbf{x}$. The first two signals we present are instantaneous in the sense that they only depend on $\mathbf{x}$. Such signals are appealing because they are typically cheaper to evaluate, and are agnostic about the overall goal of the curriculum. The remaining three signals more directly measure the effect of training on the desired objective, but require an additional sample $\mathbf{x}'$. In what follows we denote the model parameters before and after training on $\mathbf{x}$ by θ and θ' respectively.

Algorithm 1 Intrinsically Motivated Curriculum Learning

Initially: $w_i = 0$ for $i \in [N]$

for $t = 1 \dots T$ **do**
  $\pi(k) := (1-\epsilon)\frac{e^{w_k}}{\sum_i e^{w_i}} + \frac{\epsilon}{N}$
  Draw task index k from π
  Draw training sample $\mathbf{x}$ from D_k
  Train network p_θ on $\mathbf{x}$
  Compute learning progress ν (Sections 3.1 & 3.2)
  Map $\hat{r} = \nu/\tau(\mathbf{x})$ to $r \in [-1, 1]$ (Section 2.3)
  Update w_i with reward r using Exp3.S (1)
end for

Prediction gain (PG). Prediction gain is defined as the instantaneous change in loss for a sample $\mathbf{x}$, before and after training on $\mathbf{x}$:

$$\nu_{PG} := L(\mathbf{x}, \theta) - L(\mathbf{x}, \theta').$$

When p_θ is a Bayesian mixture model, prediction gain upper bounds the model's information gain (Bellemare et al., 2016), and is therefore closely related to the Bayesian precept that learning is a change in posterior.

Gradient prediction gain (GPG). Computing prediction gain requires an additional forward pass. When p_θ is differentiable, an alternative is to consider the first-order Taylor series approximation to prediction gain:

$$L(\mathbf{x}, \theta') \approx L(\mathbf{x}, \theta) + [\nabla L(\mathbf{x}, \theta)]^\top \Delta_\theta,$$

where Δ_θ is the descent step. Taking this step to be the negative gradient $-\nabla_\theta L(\mathbf{x}, \theta)$ we obtain the gradient prediction gain

$$\nu_{GPG} := \|\nabla L(\mathbf{x}, \theta)\|_2^2.$$

This measures the magnitude of the gradient vector, which has been used an indicator of data salience in the active learning literature (Settles et al., 2008). We will show below that gradient prediction gain is a biased estimate true expected learning progress, and in particular favours tasks whose loss has higher variance.

Self prediction gain (SPG). Prediction gain is a biased estimate of the change in $\mathcal{L}_k(\theta)$, the expected loss on task k. Having trained on $\mathbf{x}$, we naturally expect the sample loss $L(\mathbf{x}, \theta)$ to decrease, even though the loss at other points may increase. Self prediction gain addresses this issue by sampling a second time from the same task and estimating progress on the new sample:

$$\nu_{SPG} := L(\mathbf{x}', \theta) - L(\mathbf{x}', \theta') \qquad \mathbf{x}' \sim D_k.$$

Target prediction gain (TPG). We can take the self-prediction gain idea further and evaluate directly on the loss of interest, which has has also been considered in active learning (Roy and Mccallum, 2001). In the target task setting, this becomes

$$\nu_{TPG} := L(\mathbf{x}', \theta) - L(\mathbf{x}', \theta') \qquad \mathbf{x}' \sim D_N.$$

Although this might seem like the most accurate measure so far, it tends to suffer from high variance, and also runs counter to the premise that, early in training, the model cannot improve on the difficult target task and should instead train on a task that it can master.

Mean prediction gain (MPG). Mean prediction gain is the analogue of target prediction gain in the multiple tasks setting, where it is natural to evaluate our progress across all tasks. We write

$$\nu_{MPG} := L(\mathbf{x}', \theta) - L(\mathbf{x}', \theta') \qquad \mathbf{x}' \sim D_k, k \sim U_N,$$

where U_N denotes the uniform distribution over $\{1, \ldots, N\}$. Mean prediction gain has additional variance from sampling an evaluation task $k \sim U_N$.

3.2. Complexity-driven Progress

So far we have considered gains that gauge the network's learning progress directly, by observing the rate of change in its predictive ability. We now turn to a novel set of gains that instead measure the rate at which the network's complexity increases. These gains are underpinned by the Minimum Description Length (MDL) principle (Rissanen, 1986; Grünwald, 2007): in order to best generalise from a particular dataset, one should minimise the number of bits required to describe the model parameters plus the number of bits required for the model to describe the data.

According to the MDL principle, increasing the model complexity by a certain amount is only worthwhile if it compresses the data by a greater amount. We would therefore expect the complexity to increase most in response to the training examples from which the network is best able to generalise. These examples are exactly what we seek when attempting to maximise learning progress.

MDL training for neural networks (Hinton and Van Camp, 1993) can be practically realised with stochastic variational inference (Graves, 2011; Kingma et al., 2015; Blundell et al., 2015). In this framework a variational posterior $P_\phi(\theta)$ over the network weights is maintained during training, with a single weight sample drawn for each training example. The parameters ϕ of the posterior are optimised, rather than θ itself. The total loss is the expected log-loss of the training dataset[1] (which in our case is the complete curriculum), plus the KL-divergence between the posterior and some fixed (Blundell et al., 2015) or adaptive (Graves, 2011) prior $Q_\psi(\theta)$:

$$L_{VI}(\phi, \psi) = \underbrace{KL(P_\phi \,\|\, Q_\psi)}_{\text{model complexity}} + \underbrace{\sum\nolimits_k \sum_{\mathbf{x} \in D_k} \mathop{\mathbb{E}}_{\theta \sim P_\phi} L(\mathbf{x}, \theta)}_{\text{data cost}}.$$

Since we are using stochastic gradient descent we need to determine the per-sample loss for both the model complexity and the data. Defining $S := \sum_k |D_k|$ as the total number of samples in the curriculum we obtain

$$L_{VI}(\mathbf{x}, \phi, \psi) := \frac{1}{S} KL(P_\phi \,\|\, Q_\psi) + \mathop{\mathbb{E}}_{\theta \sim P_\phi} L(\mathbf{x}, \theta), \quad (3)$$

with $L_{VI}(\phi, \psi) = \sum_k \sum_{\mathbf{x} \sim D_k} L_{VI}(\mathbf{x}, \phi, \psi)$. Some of the curricula we consider are algorithmically generated, meaning that the number of samples in each task is undefined. The treatment suggested by the MDL principle is to divide the complexity cost by the total number of samples generated so far. However we simplified matters by setting S to a large constant that roughly matches the number of samples we expect to see during training.

We used a diagonal Gaussian for both P and Q, allowing us to determine the complexity cost analytically:

$$KL(P_\phi \,\|\, Q_\psi) = \frac{(\mu_\phi - \mu_\psi)^2 + \sigma_\phi^2 - \sigma_\psi^2}{2\sigma_\psi^2} + \ln\left(\frac{\sigma_\psi}{\sigma_\phi}\right),$$

where μ_ϕ, σ_ϕ^2 and μ_ψ, σ_ψ^2 are the mean and variance vectors for P_ϕ and Q_ψ respectively. We adapted ψ with gradient descent along with ϕ, and the gradient of $\mathbb{E}_{\theta \sim P_\phi} L(\mathbf{x}, \theta)$ with respect to ϕ was estimated using the reparameterisation trick[2] (Kingma and Welling, 2013) with a single Monte-Carlo sample. The SoftPlus function $y = \ln(1 + e^x)$ was used to ensure that the variances were positive (Blundell et al., 2015).

Variational complexity gain (VCG). The increase of model complexity induced by a training example can be estimated from the change in complexity following a single parameter update from ϕ to ϕ' and ψ to ψ', yielding

$$\nu_{VCG} := KL(P_{\phi'} \,\|\, Q_{\psi'}) - KL(P_\phi \,\|\, Q_\psi)$$

Gradient variational complexity gain (GVCG). As with prediction gain, we can derive a first order Taylor ap-

[1] MDL deals with *sets* rather than *distributions*; in this context we consider each $\mathcal{D}_k$ in the curriculum to be a dataset sampled from the task distribution, rather than the distribution itself

[2] The reparameterisation trick yields a better gradient estimator for the posterior variance than that used in (Graves, 2011), which requires either calculation of the diagonal of the Hessian, or a biased approximation using the empirical Fisher. The gradient estimator for the posterior mean is the same in both cases.

proximation using the direction of gradient descent:

$$KL(P_{\phi'} \,\|\, Q_{\psi'}) \approx KL(P_\phi \,\|\, Q_\psi) - [\nabla_{\phi,\psi} KL(P_\phi \,\|\, Q_\psi)]^\top \nabla_{\psi,\phi} \mathcal{L}_{MDL}(\mathbf{x}, \phi, \psi)$$
$$\implies \nu_{VCG} \approx C - [\nabla_{\phi,\psi} KL(P_\phi \,\|\, Q_\psi)]^\top \nabla_\phi \mathop{\mathbb{E}}_{\theta \sim P_\phi} L(\mathbf{x}, \theta),$$

where C is a term that does not depend on $\mathbf{x}$ and is therefore irrelevant to the gain signal. We define the gradient variational complexity gain as

$$\nu_{GVCG} := [\nabla_{\phi,\psi} KL(P_\phi \,\|\, Q_\psi)]^\top \nabla_\phi \mathop{\mathbb{E}}_{\theta \sim P_\phi} L(\mathbf{x}, \theta),$$

which is the directional derivative of the KL along the gradient descent direction. We believe that the linear approximation is more reliable here than for prediction gain, as the model complexity has less curvature than the loss surface.

Relationship to VIME. Variational Information Maximizing Exploration (VIME) (Houthooft et al., 2016), uses a reward signal that is closely related to variational complexity gain. The difference is that while VIME measures the KL between the posterior before and after a step in parameter space, we consider the change in KL between the posterior and prior induced by the step. Therefore, while VIME looks for any change to the posterior, we focus only on changes that alter the divergence from the prior. Further research will be needed to assess the relative merits of the two signals.

L2 gain (L2G). Variational inference tends to slow down learning, making it appealing to define a complexity-based progress signal applicable to more conventionally trained networks. Many of the standard neural network regularisation terms, such as Lp-norms, can be viewed as defining an upper bound on model description length (Graves, 2011). We therefore hypothesize that the increase in regularisation cost will be indicative of the increase in model complexity. To test this hypothesis we consider training with a standard L2 regularisation term added to the loss:

$$L_{L2}(\mathbf{x}, \theta) = L(\mathbf{x}, \theta) + \frac{\alpha}{2} \|\theta\|_2^2 \tag{4}$$

where α is an empirically chosen constant. In this case the complexity gain can be defined as

$$\nu_{L2G} := \|\theta'\|_2^2 - \|\theta\|_2^2 \tag{5}$$

where we have dropped the $\alpha/2$ term as the gain will anyway be rescaled to $[-1, 1]$ before use. The corresponding first-order approximation is

$$\nu_{GL2G} := [\theta]^\top \nabla_\theta L(\mathbf{x}, \theta) \tag{6}$$

It is possible to calculate L2 gain for unregularized networks; however we found this an unreliable signal, presumably because the network has no incentive to decrease complexity when faced with uninformative data.

3.3. Prediction Gain Bias

Prediction gain, self prediction gain and gradient prediction gain are all closely related, but incur varying degrees of bias and variance. We now present a formal analysis of the biases present in each, noting that a similar treatment can be applied to our complexity gains.

We assume that the loss L is locally well-approximated by its first-order Taylor expansion:

$$L(\mathbf{x}, \theta') \approx L(\mathbf{x}, \theta) + \nabla L(\mathbf{x}, \theta)^\top \Delta\theta \tag{7}$$

where $\Delta\theta := \theta' - \theta$. For ease of exposition, we also suppose the network is trained with stochastic gradient descent (the same argument leads to similar conclusions when consider higher-order optimization methods):

$$\Delta\theta := -\alpha \nabla L(\mathbf{x}, \theta). \tag{8}$$

We define the true expected learning progress as

$$\nu := \mathop{\mathbb{E}}_{\mathbf{x}' \sim D} [\mathcal{L}(\theta) - \mathcal{L}(\theta')] = \alpha \Big\| \mathop{\mathbb{E}}_{\mathbf{x}' \sim D} \nabla L(\mathbf{x}, \theta) \Big\|^2,$$

with the identity following from (8) (recall that $\mathcal{L}(\theta) = \mathbb{E}_{\mathbf{x}} L(\theta)$). The expected prediction gain is then

$$\nu_{\text{PG}} = \mathop{\mathbb{E}}_{\mathbf{x}' \sim D} [L(\mathbf{x}, \theta) - L(\mathbf{x}, \theta')] = \alpha \mathop{\mathbb{E}}_{\mathbf{x}' \sim D} \big\| \nabla L(\mathbf{x}, \theta) \big\|^2.$$

Defining

$$\mathbb{V}\big(\nabla L(\mathbf{x}, \theta)\big) := \mathbb{E} \big\| \nabla L(\mathbf{x}, \theta) - \mathbb{E} \nabla L(\mathbf{x}', \theta) \|^2,$$

we find that prediction gain is the sum of two terms: true expected learning progress, plus the gradient variance:

$$\nu_{\text{PG}} = \nu + \mathbb{V}\big(\nabla L(\mathbf{x}, \theta)\big).$$

We conclude that *for equal learning progress, a prediction gain-based curriculum maximizes variance*. The problem is made worse when using gradient prediction gain, which actually relies on the Taylor approximation (7). On the other hand, self prediction gain is an unbiased estimate of expected learning progress:

$$\mathop{\mathbb{E}}_{\mathbf{x}} \nu_{\text{SPG}} = \mathop{\mathbb{E}}_{\mathbf{x}, \mathbf{x}' \sim D} [L(\mathbf{x}', \theta) - L(\mathbf{x}', \theta')] = \nu.$$

Naturally, its use of two samples results in higher variance than prediction gain, suggesting a bias-variance trade off between the two estimates.

4. Experiments

To test the efficacy of our approach, we applied all the gains defined in the previous section to three task suites: synthetic language modelling on text generated by n-gram

models, repeat copy (Graves et al., 2014) and the bAbI tasks (Weston et al., 2015)

The network architecture was stacked unidirectional LSTM (Graves, 2013) for all experiments, and the training loss was cross-entropy with either categorical targets and softmax output, or Bernoulli targets and sigmoid outputs, optimised by RMSProp with momentum (Tieleman, 2012; Graves, 2013), using a momentum of 0.9 and a learning rate of 10^{-5} unless specified otherwise. The parameters for the Exp3.S algorithm were $\eta = 10^{-3}$, $\beta = 0$, $\epsilon = 0.05$. For all experiments, one set of networks was trained with variational inference (VI) to test the variational complexity gain signals, and another set was trained with normal maximum likelihood (ML) for the other signals; both sets were repeated 10 times with different random seeds to initialise the network weights. The α regularisation parameter from Eq. (4) for the networks trained with L2 gain signals was 10^{-4} for all experiments. For all plots with a time axis, time is defined as the total number of input steps processed so far. In the absence of hand-designed curricula for these tasks, our performance benchmarks are 1) a fixed uniform policy over all the tasks and 2) directly training on the target task (where applicable). All losses and error rates are measured on independent samples not used for training or reward calculation.

4.1. N-Gram Language Modelling

Our first experiment aims to illustrate and compare the behaviour induced by different gains. We trained character-level Kneser-Ney n-gram models (Kneser and Ney, 1995) on the King James Bible data from the Canterbury corpus (Arnold and Bell, 1997), with the maximum depth parameter n ranging between 0 to 10. We then used each model to generate a separate dataset of 1M characters, which we divided into disjoint sequences of 150 characters. The first 50 characters of each sequence were used as burn-in context for the next 100, which the network was trained to predict. The LSTM network had two layers of 512 cells, and the batch size was 32.

An important characteristic of this dataset is that the amount of linguistic structure increases monotonically with n. Simultaneously, the entropy – and hence, minimum achievable loss – decreases almost monotonically in n. If we believe that learning progress should be higher for interesting data than for data that is difficult to predict, we would expect the gain signals to be drawn to higher n: they should favour structure over noise. We note that in this experiment the curriculum is superfluous: the most efficient strategy for learning the 10-gram source is to directly train on it.

Fig. 1 shows that most of the complexity-based gain signals from Section 3.2 (L2G, GL2G, GVCG) progress rapidly

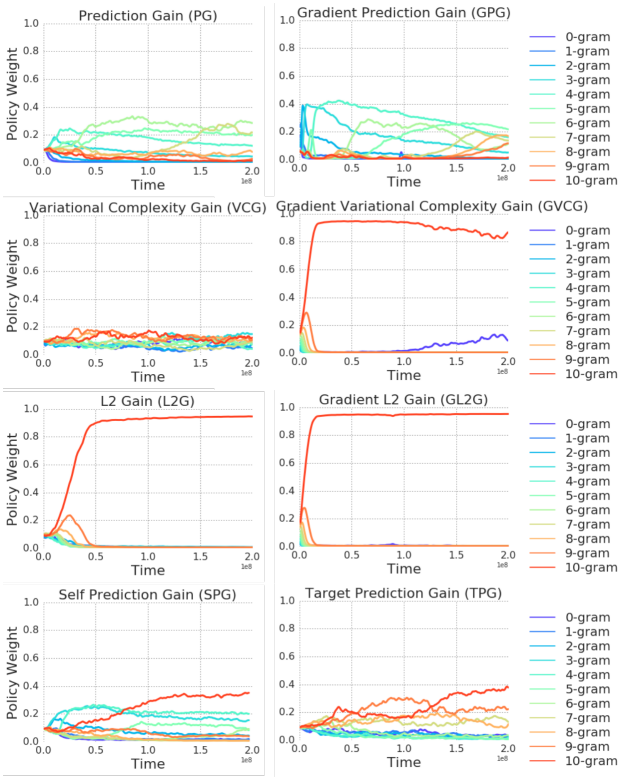

Figure 1. N-gram policies for different gain signals, truncated at 2×10^8 steps. All curves are averages over 10 runs

through the curriculum before focusing strongly on the 10-gram task (though interestingly, GVCG appears to revisit 0-gram later on in training). The clarity of the result is striking, given that sequences generated from models beyond about 6-gram are difficult to distinguish by eye. VCG follows a similar path, but with much less confidence, presumably due to the increased noise. The loss-based measures (PG, GPG, SPG, TG) also tend to move towards higher n, although more slowly and with less certainty. Unlike the complexity gains, they tend to initially favour the lower-n tasks, which may be desirable as we would expect early learning to be more efficient with simpler data.

4.2. Repeat Copy

In the repeat copy task (Graves et al., 2014) the network receives an input sequence of random bit vectors, and is then asked to output that sequence a given number of times. The task has two main dimensions of difficulty: the length of the input sequence and the required number of repeats, both of which increase the demand on the models memory. Neural Turing machines are able to learn a 'for-loop' like algorithm on simple examples that can directly generalise to much harder examples (Graves et al., 2014). For LSTM networks without access to external memory, however, sig-

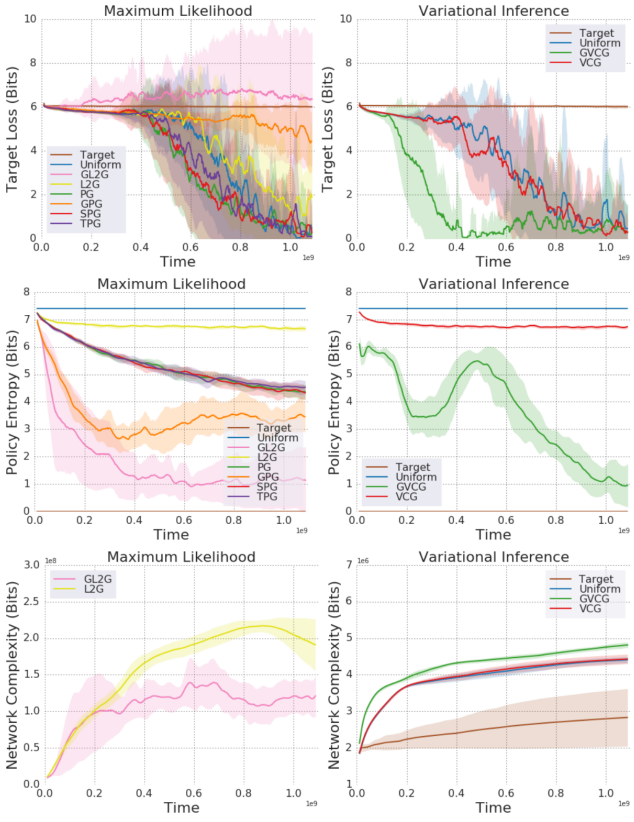

Figure 2. Target task loss (per output), policy entropy and network complexity for the repeat copy task, truncated at 1.1×10^9 steps. Curves are averages over 10 runs, shaded areas show the standard deviation. Network complexity was computed by multiplying the per-sample complexity cost by the total size of the training set.

nificant retraining is required to adapt to harder tasks.

We devised a curriculum with both the sequence length and the number of repeats varying from 1 to 13, giving 169 tasks in all, with length 13, repeats 13 defined as the target task. The LSTM network had a single layer of 512 cells, and the batch size was 32. As the data was generated online, the number of samples S in Eq. (3) (the per-sample VI loss) was undefined; we arbitrarily set it to 169M (1M per task in the curriculum).

Fig. 2 shows that GVCG solves the target task about twice as fast as uniform sampling for VI training, and that the PG, SPG and TPG gains are somewhat faster than uniform for ML training, especially in the early stages. From the entropy plots it is clear that these signals all lead to strongly non-uniform policies. The VI complexity curves also demonstrate that GVCG yields significantly higher network complexity than uniform sampling, supporting our hypothesis that increased complexity correlates with learning progress. Unlike GVCG, the VCG signal did not deviate far from a uniform policy,. L2G and particularly GPG and GL2G were much worse than uniform, suggesting that (1) the bias induced by the gradient approximation has a pernicious effect on learning and (2) that the increase in L2 norm is not a reliable measure of increased network complexity. Training directly on the target task failed to learn at all, underlining the necessity of curriculum learning for this problem.

Fig. 3 reveals a consistent strategy for the GVCG syllabuses, first focusing on short sequences with high repeats, then long sequences with low repeats, thereby decoupling the two dimensions of difficulty. At each stage the loss is substantially reduced across many tasks that the policy does not focus on. Crucially, this means that the network does not have to visit each of the 169 tasks to solve them all, and the syllabus is able to exploit this fact to more efficiently complete the curriculum.

4.3. Babi

The bAbI dataset (Weston et al., 2015) consists of 20 synthetic question-answering problems designed to probe the basic reasoning capabilities of machine learning models. Although bAbI was not specifically designed for curriculum learning, some of the tasks follow a natural ordering of complexity (e.g. 'Two Arg Relations', 'Three Arg Relations') and all are based on a consistent probabilistic grammar, leading us to hope that an efficient syllabus could be found for learning the whole set. The usual performance measure for bAbI is the number of tasks 'completed' by the model, where completion is defined as getting less than 5% of the test set questions wrong.

The data representation followed (Graves et al., 2016), with each word in the observation and target sequences represented as a 1-hot vector, along with an extra binary channel to mark answer prompts. The original datasets were small, with either 1K or 10K questions per task, so as to test generalisation from limited samples. However LSTM is known to perform poorly in this setting (Sukhbaatar et al., 2015; Graves et al., 2016), and we wished to avoid the confounding effect of overfitting on curriculum learning. We therefore used the bAbI code (Weston et al., 2015) to generate 1M stories (each containing one or more questions) for each of the 20 tasks. With so many examples, we found that training and evaluation set performance were indistinguishable, and therefore report training performance only. The LSTM network had two layer of 512 cells, the batch size was 16, and the RMSProp learning rate was 3×10^{-5}.

Fig. 4 shows that prediction gain (PG) clearly improved on uniform sampling in terms of both learning speed and number of tasks completed; for self-prediction gain (SPG) the same benefits were visible, though less pronounced. The other gains were either roughly equal to or worse than uniform. For variational inference training, GVCG was faster

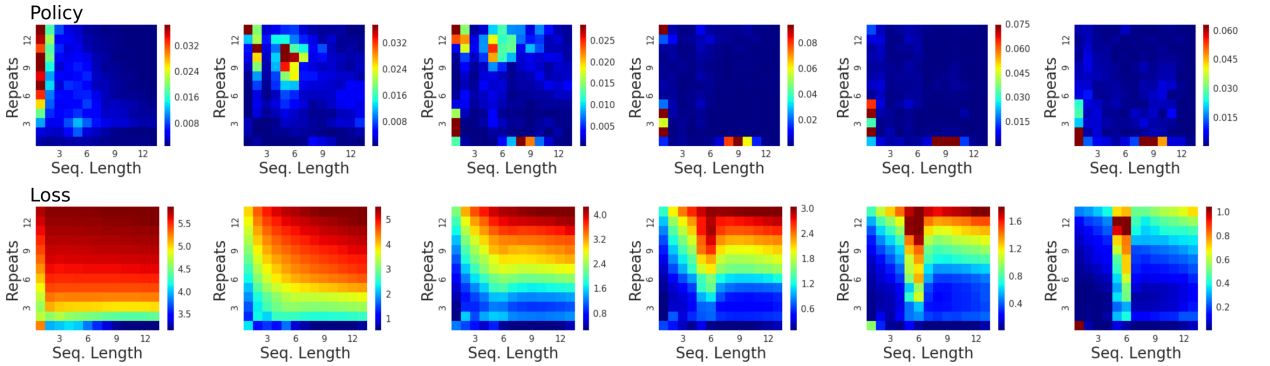

Figure 3. Average policy and loss per output over time for GVCG networks on the repeat copy task. Plots were made by dividing the first 4×10^8 steps into five equal bins, then averaging over the policies of all 10 networks over all times within each bin.

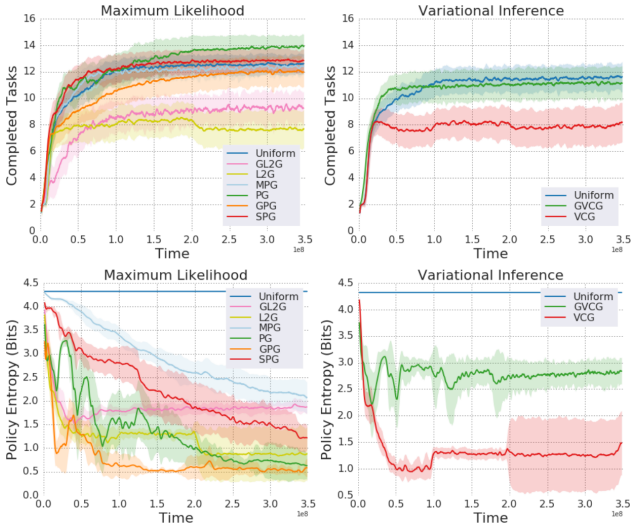

Figure 4. Completion and entropy curves for the bAbI curriculum, truncated at 3.5×10^8 steps. Curves are means over ten runs, shaded areas show standard deviation.

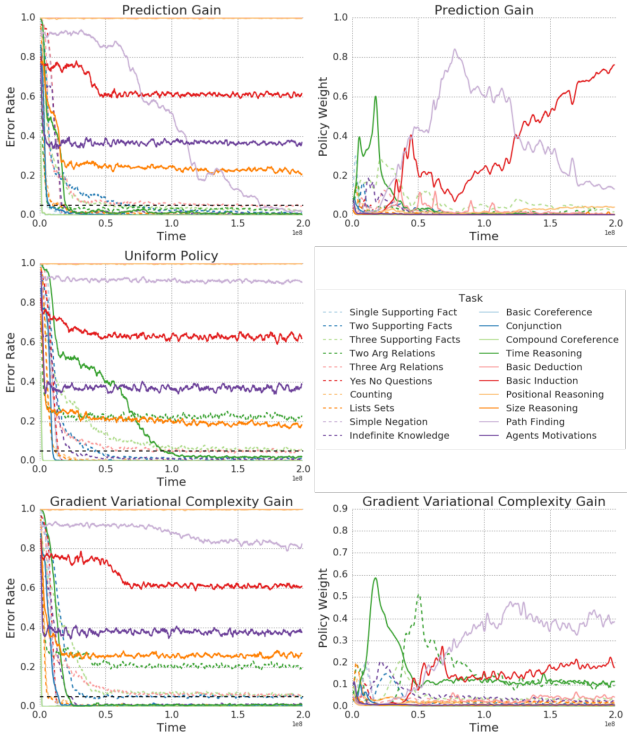

Figure 5. Per-task policy and error curves for bAbI, truncated at 2×10^8 steps. All plots are averaged over 10 runs. Black dashed lines show the 5% error threshold for task completion.

than uniform at first, then slightly worse later on, while VCG performed poorly for reasons that are unclear to us. In general, training with variational inference appeared to hamper progress on the bAbI tasks.

Fig. 5 shows how the PG and GVCG syllabuses accelerate the network's progress by selectively focusing on specific tasks until completion. For example, they both solve 'Time Reasoning' much faster than uniform sampling by concentrating on it early in training; similarly, PG focuses strongly on 'Path Finding' (one of the harder bAbI tasks) until it solves it. Also noteworthy is the way the syllabuses progress from 'Single Supporting Fact' to 'Three Supporting Facts' in order; this shows that our gain signals can discover implicit orderings, and hence opportunities for efficient transfer, in an unsorted curriculum.

5. Conclusion

Our experiments suggest that using a stochastic syllabus to maximise learning progress can lead to significant gains in curriculum learning efficiency, so long as a suitable progress signal is used. We note however that uniformly sampling from all tasks is a surprisingly strong benchmark. We speculate that this is because learning is dominated by gradients from the tasks on which the network is making

fastest progress, inducing a kind of implicit curriculum, albeit with the inefficiency of unnecessary samples. For maximum likelihood training, we found prediction gain to be the most consistent signal, while for variational inference training, gradient variational complexity gain performed best. Importantly, both are instantaneous, in the sense that they can be evaluated using only the samples used for training. As well as being more efficient, this has broader applicability to tasks where external evaluation is difficult, and suggests that learning progress is best assessed on a local, rather than global basis.

References

Arnold, R. and Bell, T. (1997). A corpus for the evaluation of lossless compression algorithms. In *Data Compression Conference, 1997. DCC'97. Proceedings*, pages 201–210. IEEE.

Auer, P., Cesa-Bianchi, N., Freund, Y., and Schapire, R. E. (2002). The nonstochastic multiarmed bandit problem. *SIAM Journal on Computing*, 32(1):48–77.

Barto, A. G. (2013). Intrinsic motivation and reinforcement learning. In *Intrinsically Motivated Learning in Natural and Artificial Systems*, pages 17–47. Springer.

Bellemare, M. G., Srinivasan, S., Ostrovski, G., Schaul, T., Saxton, D., and Munos, R. (2016). Unifying count-based exploration and intrinsic motivation. In *Advances in Neural Information Processing Systems*.

Bengio, Y., Louradour, J., Collobert, R., and Weston, J. (2009). Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 41–48, New York, NY, USA. ACM.

Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. (2015). Weight uncertainty in neural networks. In *Proceedings of The 32nd International Conference on Machine Learning*, pages 1613–1622.

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

Elman, J. L. (1993). Learning and development in neural networks: The importance of starting small. *Cognition*, 48(1):71–99.

Graves, A. (2011). Practical variational inference for neural networks. In Shawe-Taylor, J., Zemel, R. S., Bartlett, P. L., Pereira, F., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 24*, pages 2348–2356. Curran Associates, Inc.

Graves, A. (2013). Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*.

Graves, A., Wayne, G., and Danihelka, I. (2014). Neural turing machines. *arXiv preprint arXiv:1410.5401*.

Graves, A., Wayne, G., Reynolds, M., Harley, T., Danihelka, I., Grabska-Barwińska, A., Colmenarejo, S. G., Grefenstette, E., Ramalho, T., Agapiou, J., et al. (2016). Hybrid computing using a neural network with dynamic external memory. *Nature*, 538(7626):471–476.

Grünwald, P. D. (2007). *The minimum description length principle*. The MIT Press.

Herbster, M. and Warmuth, M. K. (1998). Tracking the best expert. *Machine Learning*, 32(2):151–178.

Hinton, G. E. and Van Camp, D. (1993). Keeping the neural networks simple by minimizing the description length of the weights. In *Proceedings of the sixth annual conference on Computational learning theory*, pages 5–13. ACM.

Houthooft, R., Chen, X., Duan, Y., Schulman, J., De Turck, F., and Abbeel, P. (2016). Vime: Variational information maximizing exploration. In *Advances In Neural Information Processing Systems*, pages 1109–1117.

Itti, L. and Baldi, P. (2009). Bayesian surprise attracts human attention. *Vision research*, 49(10):1295–1306.

Kingma, D. P., Salimans, T., and Welling, M. (2015). Variational dropout and the local reparameterization trick. In *Advances in Neural Information Processing Systems*, pages 2575–2583.

Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Kneser, R. and Ney, H. (1995). Improved backing-off for m-gram language modeling. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 181–184, Detroit, Michigan, USA.

Lopes, M. and Oudeyer, P.-Y. (2012). The strategic student approach for life-long exploration and learning. In *IEEE International Conference on Development and Learning and Epigenetic Robotics (ICDL)*.

Oudeyer, P., Kaplan, F., and Hafner, V. (2007). Intrinsic motivation systems for autonomous mental development. *IEEE Transactions on Evolutionary Computation*, 11(2):265–286.

Reed, S. and de Freitas, N. (2015). Neural programmer-interpreters. *arXiv preprint arXiv:1511.06279*.

Rissanen, J. (1986). Stochastic complexity and modeling. *Ann. Statist.*, 14(3):1080–1100.

Roy, N. and Mccallum, A. (2001). Toward optimal active learning through sampling estimation of error reduction. In *In Proc. 18th International Conf. on Machine Learning*.

Schmidhuber, J. (1991). A possibility for implementing curiosity and boredom in model-building neural controllers. In *From animals to animats: proceedings of the first international conference on simulation of adaptive behavior*.

Settles, B. (2010). Active learning literature survey. *University of Wisconsin, Madison*, 52(55-66):11.

Settles, B., Craven, M., and Ray, S. (2008). Multiple-instance active learning. In *Advances in neural information processing systems*, pages 1289–1296.

Storck, J., Hochreiter, J., and Schmidhuber, J. (1995). Reinforcement driven information acquisition in non-deterministic environments. In *Proceedings of the International Conference on Artificial Neural Networks, vol. 2*.

Sukhbaatar, S., Weston, J., Fergus, R., et al. (2015). End-to-end memory networks. In *Advances in neural information processing systems*, pages 2440–2448.

Sutskever, I. and Zaremba, W. (2014). Learning to execute. *arXiv preprint arXiv:1410.4615*.

Tieleman, T., H. G. (2012). Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Networks for Machine Learning*.

Tsvetkov, Yulia, F. M. L. W. M. B. D. C. (2016). Learning the curriculum with bayesian optimization for task-specific word representation learning. *arXiv preprint arXiv:1605.03852*.

Weston, J., Bordes, A., Chopra, S., and Mikolov, T. (2015). Towards ai-complete question answering: A set of prerequisite toy tasks. *CoRR*, abs/1502.05698.