Automated Curriculum Learning for Neural Networks

Alex Graves, Marc G. Bellemare, Jacob Menick, Remi Munos, Koray Kavukcuoglu

DeepMind

ICML 2017

Presenter: Jack Lanchantin

Outline

Outline

1. Introduction
   - Curriculum Learning
   - Task
   - Multi-Armed Bandits
2. Learning Progress Signals
   - Learning Progress Signals
   - Loss-driven Progress
   - Complexity-driven Progress
3. Experiments
   - 3 tasks
   - N-gram
   - Repeat Copy
   - bAbI

Curriculum Learning (CL)

- "The importance of starting small" (Ellman, 1993)
- CL is highly sensitive to the mode of progression through the tasks
- Previous methods: tasks can be ordered by difficulty
  - in reality they may vary along multiple axes of difficulty, or have no predefined order at all
- This paper: treat the decision about **which task to study next as a stochastic policy**, continuously adapted to optimise some notion of "learning progress"

Curriculum Learning (CL)

- "The importance of starting small" (Ellman, 1993)
- CL is highly sensitive to the mode of progression through the tasks
- Previous methods: tasks can be ordered by difficulty
  - in reality they may vary along multiple axes of difficulty, or have no predefined order at all
- This paper: treat the decision about **which task to study next as a stochastic policy**, continuously adapted to optimise some notion of "learning progress"

Curriculum Learning (CL)

- "The importance of starting small" (Ellman, 1993)
- CL is highly sensitive to the mode of progression through the tasks
- Previous methods: tasks can be ordered by difficulty
  - in reality they may vary along multiple axes of difficulty, or have no predefined order at all
- This paper: treat the decision about **which task to study next as a stochastic policy**, continuously adapted to optimise some notion of "learning progress"

Outline

Curriculum Learning Task

Each example $x \in X$ contains input a and target b:

- **Task:** a distribution D over sequences from X
- **Curriculum:** an ensemble of tasks $D_1, \ldots, D_N$
- **Sample:** an example drawn from one of the tasks of the curriculum
- **Syllabus:** a time-varying sequence of distributions over tasks

The expected loss of the network on the k^{th} task is

$$\mathcal{L}_k(\theta) := \mathbb{E}_{x \sim D_k} L(x, \theta) \tag{1}$$

Where $L(x, \theta) := -\log p_\theta(x)$ is the sample loss on x

Curriculum Learning Task

Each example $x \in X$ contains input a and target b:

- **Task:** a distribution D over sequences from X
- **Curriculum:** an ensemble of tasks $D_1, \dots, D_N$
- **Sample:** an example drawn from one of the tasks of the curriculum
- **Syllabus:** a time-varying sequence of distributions over tasks

The expected loss of the network on the k^{th} task is

$$\mathcal{L}_k(\theta) := \mathbb{E}_{x \sim D_k} L(x, \theta) \tag{1}$$

Where $L(x, \theta) := -\log p_\theta(x)$ is the sample loss on x

Curriculum Learning: Two related settings

1. **Multiple tasks setting:** Perform well on all tasks in $\{D_k\}$:

$$\mathcal{L}_{MT} := \frac{1}{N}\sum_{k=1}^{N}\mathcal{L}_k \tag{2}$$

2. **Target task setting:** Only interested in minimizing the loss on the final task D_N:

$$\mathcal{L}_{TT} := \mathcal{L}_N \tag{3}$$

The other tasks act as a series of stepping stones to the real problem

Outline

Multi-Armed Bandits for CL

- Model a curriculum containing N tasks as an N-armed bandit
- Syllabus: adaptive policy which seeks to maximize payoffs from bandit
- An agent selects a sequence of 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 reward r_t

Exp3 Algorithm for Multi-Armed Bandits

On round t, the agent selects an arm stochastically according to policy π_t. This policy is defined by a set of weights $w_{t,i}$:

$$\pi_t^{EXP3}(i) := \frac{e^{w_{t,i}}}{\sum_{j=1}^{N} e^{w_{t,j}}} \tag{4}$$

The weights are the sum of importance-sampled rewards:

$$w_{t,i} := \eta \sum_{s<t} \tilde{r}_{s,i} \tag{5}$$

$$\tilde{r}_{s,i} := \frac{r_s \mathbb{I}_{[a_s=i]}}{\pi_s(i)} \tag{6}$$

Outline

Learning Progress Signals for CL

- **Goal:** use the **policy output by Exp3 as a syllabus** for training our models
  - Ideally: policy should maximize the rate at which we minimize the loss, and the reward should reflect this rate
  - Hard to measure effect of a training sample on the target objective
- **Method:** Introduce defined measures of progress:
  - Loss-driven: equate reward with a decrease in some loss
  - Complexity-driven: equate reward with an increase in model complexity

Training for Intrinsically Motivated Curriculum Learning

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

T rounds, N number of tasks

Outline

Loss-driven Progress

Loss-driven Progress: Compare the predictions made by the model before and after training on some sample x

1. Prediction Gain (PG)

$$V_{PG} := L(x, \theta) - L(x, \theta') \tag{7}$$

2. Gradient prediction Gain (GPG)

$$L(x, \theta') \approx L(x, \theta) + [\nabla L(x, \theta)]^T \Delta_\theta \tag{8}$$

where Δ_θ is the descent step, $-\nabla_\theta L(x, \theta)$

$$V_{GPG} := ||\nabla L(x, \theta)||_2^2 \tag{9}$$

Loss-driven Progress

Loss-driven Progress: Compare the predictions made by the model before and after training on some sample x

1. Prediction Gain (PG)

$$V_{PG} := L(x, \theta) - L(x, \theta') \tag{7}$$

2. Gradient prediction Gain (GPG)

$$L(x, \theta') \approx L(x, \theta) + [\nabla L(x, \theta)]^T \Delta_\theta \tag{8}$$

where Δ_θ is the descent step, $-\nabla_\theta L(x, \theta)$

$$V_{GPG} := ||\nabla L(x, \theta)||_2^2 \tag{9}$$

Loss-driven Progress

Loss-driven Progress: Compare the predictions made by the model before and after training on some sample x

1. Prediction Gain (PG)

$$V_{PG} := L(x, \theta) - L(x, \theta') \tag{7}$$

2. Gradient prediction Gain (GPG)

$$L(x, \theta') \approx L(x, \theta) + [\nabla L(x, \theta)]^T \Delta_\theta \tag{8}$$

where Δ_θ is the descent step, $-\nabla_\theta L(x, \theta)$

$$V_{GPG} := ||\nabla L(x, \theta)||_2^2 \tag{9}$$

Loss-driven Progress

Loss-driven Progress: Compare the predictions made by the model before and after training on some sample x

3. Self prediction Gain (SPG)

$$V_{SPG} := L(x', \theta) - L(x', \theta') \quad x' \sim D_k \tag{10}$$

4. Target prediction Gain (TPG)

$$V_{TPG} := L(x', \theta) - L(x', \theta') \quad x' \sim D_N \tag{11}$$

5. Mean prediction Gain (MPG)

$$V_{TPG} := L(x', \theta) - L(x', \theta') \quad x' \sim D_k, k \sim U_N \tag{12}$$

Loss-driven Progress

Loss-driven Progress: Compare the predictions made by the model before and after training on some sample x

3. Self prediction Gain (SPG)

$$V_{SPG} := L(x', \theta) - L(x', \theta') \quad x' \sim D_k \tag{10}$$

4. Target prediction Gain (TPG)

$$V_{TPG} := L(x', \theta) - L(x', \theta') \quad x' \sim D_N \tag{11}$$

5. Mean prediction Gain (MPG)

$$V_{TPG} := L(x', \theta) - L(x', \theta') \quad x' \sim D_k, k \sim U_N \tag{12}$$

Loss-driven Progress

Loss-driven Progress: Compare the predictions made by the model before and after training on some sample x

3. Self prediction Gain (SPG)

$$V_{SPG} := L(x', \theta) - L(x', \theta') \quad x' \sim D_k \tag{10}$$

4. Target prediction Gain (TPG)

$$V_{TPG} := L(x', \theta) - L(x', \theta') \quad x' \sim D_N \tag{11}$$

5. Mean prediction Gain (MPG)

$$V_{TPG} := L(x', \theta) - L(x', \theta') \quad x' \sim D_k, k \sim U_N \tag{12}$$

Outline

Complexity-driven Progress

- **So far:** considered gains that gauge the networks learning progress directly, by observing the rate of change in its predictive ability
- **Now:** turn to a set of gains that instead measure the rate at which the networks complexity increases

Minimum Description Length (MDL) principle

- In order to best generalize from a particular dataset, one should minimize: (# of bits required to describe the model parameters) + (# of bits required for the model to describe the data)
- I.e., increasing the model complexity by a certain amount is only worthwhile if it compresses the data by a greater amount
- Therefore, complexity should increase most in response to the training examples from which the network is best able to generalize
  - These examples are exactly what we seek when attempting to maximize learning progress

Minimum Description Length (MDL) principle

- In order to best generalize from a particular dataset, one should minimize: (# of bits required to describe the model parameters) + (# of bits required for the model to describe the data)
- I.e., increasing the model complexity by a certain amount is only worthwhile if it compresses the data by a greater amount
- Therefore, complexity should increase most in response to the training examples from which the network is best able to generalize
  - These examples are exactly what we seek when attempting to maximize learning progress

Background: Variational Inference (from David Blei)

Probabilistic machine learning

- A probabilistic model is a joint distribution of hidden variables $\mathbf{z}$ and observed variables $\mathbf{x}$,

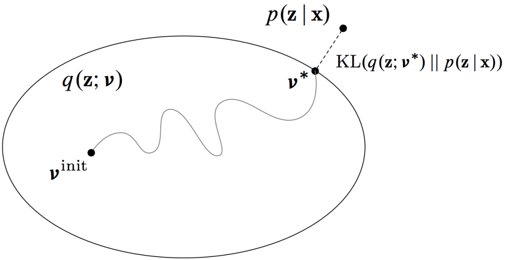
$$p(\mathbf{z}, \mathbf{x}).$$

- Inference about the unknowns is through the **posterior**, the conditional distribution of the hidden variables given the observations

$$p(\mathbf{z} \mid \mathbf{x}) = \frac{p(\mathbf{z}, \mathbf{x})}{p(\mathbf{x})}.$$

- For most interesting models, the denominator is not tractable. We appeal to **approximate posterior inference**.

Variational inference



- VI solves **inference** with **optimization**.
  (Contrast this with MCMC.)
- Posit a **variational family** of distributions over the latent variables,

$$q(\mathbf{z}; \boldsymbol{\nu})$$

- Fit the **variational parameters** $\boldsymbol{\nu}$ to be close (in KL) to the exact posterior.
  (There are alternative divergences, which connect to algorithms like EP, BP, and others.)

Minimum Description Length (MDL) principle

- MDL training in neural nets uses a variational posterior $P_\phi(\theta)$ over the network weights during training with a single weight sample drawn for each training example
- The parameters ϕ of the posterior are optimized rather than θ itself.

Varational Loss in Neural Nets

$$L_{VI}(\phi, \psi) = KL(P_\phi || Q_\psi) + \sum_k \sum_{x \in D_k} \mathbb{E}_{\theta \sim P_\phi} L(x, \theta) \tag{13}$$

$$L_{VI}(x, \phi, \psi) = \frac{1}{S} KL(P_\phi || Q_\psi) + \mathbb{E}_{\theta \sim P_\phi} L(x, \theta) \tag{14}$$

Varational Loss in Neural Nets

$$L_{VI}(\phi, \psi) = KL(P_\phi || Q_\psi) + \sum_k \sum_{x \in D_k} \mathbb{E}_{\theta \sim P_\phi} L(x, \theta) \tag{13}$$

$$L_{VI}(x, \phi, \psi) = \frac{1}{S} KL(P_\phi || Q_\psi) + \mathbb{E}_{\theta \sim P_\phi} L(x, \theta) \tag{14}$$

Complexity-driven Progress for Variational Inference

Variational Complexity Gain (VPG)

$$V_{VPG} := KL(P_{\phi'}||Q_{\psi'}) - KL(P_{\phi}||Q_{\psi}) \quad (15)$$

Gradient Variational Complexity Gain (VPG)

$$V_{GVPG} := [\nabla_{\phi,\psi} KL(P_{\phi}||Q_{\psi})]^T \nabla_{\phi} \mathbb{E}_{\phi \sim P_{\phi}} L(x, \theta) \quad (16)$$

Complexity-driven Progress for Variational Inference

Variational Complexity Gain (VPG)

$$V_{VPG} := KL(P_{\phi'}||Q_{\psi'}) - KL(P_{\phi}||Q_{\psi}) \tag{15}$$

Gradient Variational Complexity Gain (VPG)

$$V_{GVPG} := [\nabla_{\phi,\psi} KL(P_{\phi}||Q_{\psi})]^T \nabla_{\phi} \mathbb{E}_{\phi \sim P_{\phi}} L(x,\theta) \tag{16}$$

Complexity-driven Progress for Maximum Likelihood

L2 Gain (L2G)

$$L_{L2}(x,\theta) := L(x,\theta) + \frac{\alpha}{2}||\theta||_2^2 \tag{17}$$

$$V_{L2G} := ||\theta'||_2^2 - ||\theta||_2^2 \tag{18}$$

$$V_{GL2G} := [\theta]^T \nabla_\theta L(x,\theta) \tag{19}$$

Outline

Experiments

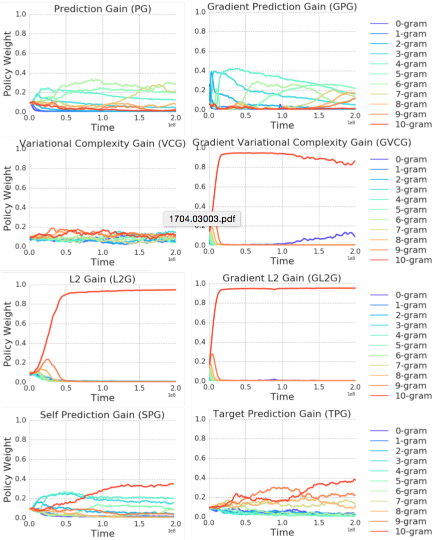
- Applied the previously defined gains in 3 tasks using the same LSTM model
  1. synthetic language modelling on text generated by n-gram models
  2. repeat copy (Graves et al., 2014)
  3. bAbI tasks (Weston et al., 2015)

Outline

N-Gram Language Modeling

- Trained character level Kneser-Ney n-gram models on the King James Bible data from the Canterbury corpus, with the maximum depth parameter n ranging between 0 to 10
- Used each model to generate a separate dataset of 1M characters, which were divided into disjoint sequences of 150 characters
- Since entropy decreases in n, learning progress should be higher for larger n, and thus the gain signals to be drawn to higher n

N-Gram Language Modeling



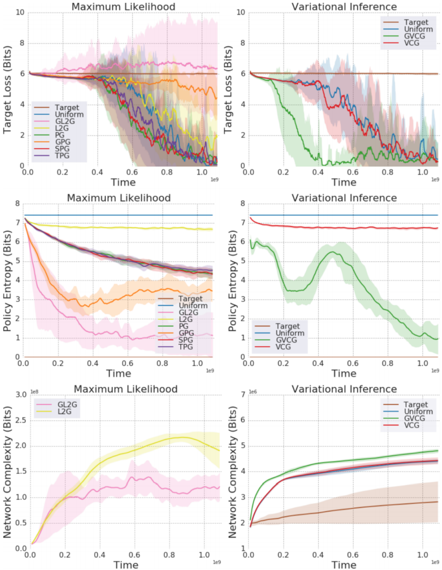
Outline

1. Introduction
   - Curriculum Learning
   - Task
   - Multi-Armed Bandits
2. Learning Progress Signals
   - Learning Progress Signals
   - Loss-driven Progress
   - Complexity-driven Progress
3. Experiments
   - 3 tasks
   - N-gram
   - Repeat Copy
   - bAbI

Repeat Copy

- Network receives an input sequence of random bit vectors, and is then asked to output that sequence a given number of times.
- Sequence length varies from 1-13, and Repeats vary from 1-13 (169 tasks in total)
- Target task is length 13 sequences and 13 repeats
- NTMs are able to learn a for-loop like algorithm on simple examples that can directly generalise to much harder examples. LSTMs require significant retraining for harder tasks

Repeat Copy



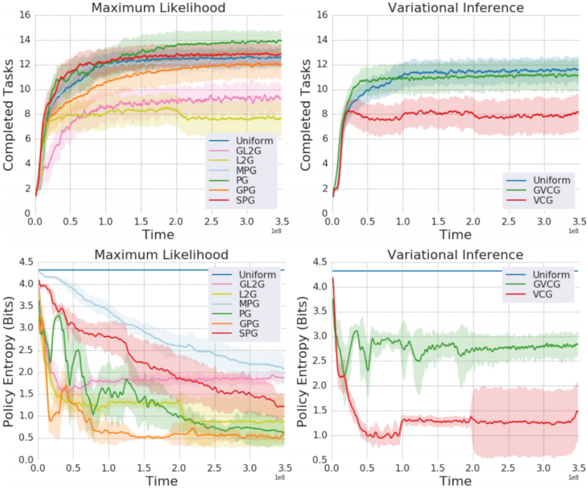
Maximum Likelihood
Variational Inference
Target Loss (Bits)
Policy Entropy (Bits)
Network Complexity (Bits)
Time
Target
Uniform
GL2G
L2G
PG
GPG
SPG
TPG
GVCG
VCG

Outline

bAbI

- 20 synthetic question-answering tasks
- 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

bAbI


Maximum Likelihood
Variational Inference
Completed Tasks
Policy Entropy (Bits)
Time
1e8
Uniform
GL2G
L2G
MPG
PG
GPG
SPG
Uniform
GVCG
VCG

Conclusion

- Using a stochastic syllabus to maximise learning progress *can* lead to significant gains in curriculum learning efficiency, so long as a a suitable progress signal is used
- Uniformly sampling from all tasks is a surprisingly strong benchmark $\rightarrow$ 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