Learning to Learn: Curriculum Selection via Soft Q-Learning for Sample-Efficient Language Models

Scott Viteri Damir Vrabac

1 Overview

We propose an algorithm that enables language models to learn what data to learn from. Rather than training on randomly sampled data, the model takes actions that select training examples, receiving reward based on how well it currently predicts held-out data. A Q-function, parameterized as a small network reading the language model's internal representations, learns to predict the long-run value of each candidate. The policy is derived from the Q-function via a Boltzmann distribution, eliminating the need for a separate policy network. The core hypothesis is that this learned curriculum selection will yield significantly better sample efficiency than random or heuristic curricula.

2 Technical Approach

2.1 Setup

- Language model (GPT-2) with parameters θ, hidden dimension $d = 768$
- Streaming data source, yielding fresh batches of N candidate context windows each step
- Fixed held-out evaluation set D, large; a fresh random subset $\hat{D} \subset D$ of size M is sampled each step for reward computation
- Q-network Q_ϕ with parameters ϕ, implemented as a 2-layer MLP ($d \to 32 \to 1$) reading the base model's residual stream (see Section 2.3)
- Boltzmann temperature $\beta > 0$ controlling exploration
- Learning rate α for the LM gradient step; learning rate η for the Q-network update
- Average reward estimate $\rho \in \mathbb{R}$, updated by exponential moving average with rate τ
- Gradient clipping threshold G (max global norm; applied to both LM and Q-network gradient steps)

2.2 State, Action, and Reward

The state s_k is the current model parameters θ_k, which determine the model's representations and therefore the features available to the Q-network. In practice, s_k is never represented explicitly—it is accessed implicitly through the base model's hidden states when candidates are forwarded through it (Section 2.3).

The action $a_k \in C_k$ is the choice of which training example to train on at step k, where C_k is the batch of N candidates available at that step. Since data is streamed, each step presents a fresh candidate batch, and each candidate is seen at most once. The Q-network generalises across batches via the base model's hidden-state representation.

The reward is the held-out log-probability *after* the training update:

$$r_k = \frac{1}{M} \log p_{\theta_{k+1}}(\hat{D}_k), \tag{1}$$

where $\hat{D}_k$ is a fresh random subset of D of size M, and θ_{k+1} results from training on a_k. This measures the model's absolute predictive fitness as a consequence of the selected action. Resampling $\hat{D}_k$ each step prevents the Q-function from overfitting to a fixed evaluation subset. The cumulative objective $\sum_k r_k$ is the integral of the learning curve: it penalises every step spent with poor predictions, rewarding curricula that reach high performance quickly. Equivalently, maximising $\sum_k r_k$ minimises cumulative regret $\sum_k (L^* - r_k)$ where L^* is the best achievable log-probability.

2.3 Q-Network Architecture

The Q-network reads the base model's internal representations to assess candidate value. For each candidate $x \in C_k$:

1. Forward x through the base model with parameters θ_k, extracting the last-layer hidden state at the final token position: $h_x = \texttt{last\_hidden}(\theta_k, x) \in \mathbb{R}^d$.
2. Detach h_x from the computation graph (no gradients flow into θ).
3. Pass through the Q-network:

$$Q_\phi(s_k, x) = W_2\, \sigma(W_1\, h_x + b_1) + b_2, \tag{2}$$

   where $\phi = \{W_1 \in \mathbb{R}^{32\times d},\ b_1 \in \mathbb{R}^{32},\ W_2 \in \mathbb{R}^{1\times 32},\ b_2 \in \mathbb{R}\}$ and σ is ReLU.

The hidden state h_x implicitly encodes both the model's current state (through θ_k, which determines the representations) and the identity of the candidate (through the input x). By detaching h_x, we ensure that θ is updated only by the language modeling objective, while ϕ is updated only by the TD loss. This prevents the co-adaptation instability that arises when value gradients distort the base model's representations.

2.4 Average Reward Formulation

With absolute log-probability rewards, the cumulative return $\sum_k r_k$ grows without bound. Rather than introducing an artificial discount factor, we subtract a running estimate of the average reward ρ from the TD target. The *differential* Q-function satisfies:

$$Q(s_k, a_k) = r_k - \rho + \beta \log \sum_{a'} \exp\big(Q(s_{k+1}, a')/\beta\big). \tag{3}$$

Subtracting ρ keeps Q-values bounded and centred: they represent how much better or worse an action is relative to recent performance, rather than accumulating unbounded absolute returns.

The policy $\pi(a \mid s) \propto \exp(Q(s,a)/\beta)$ is invariant to the value of ρ, since the softmax cancels any shared additive shift. The role of ρ is therefore purely numerical: it prevents the MLP's outputs from drifting to large absolute values where gradients saturate. In practice, ρ is maintained as an exponential moving average of observed rewards:

$$\rho \ \leftarrow\ (1-\tau)\,\rho + \tau\, r_k. \tag{4}$$

A stale or imprecise ρ does not affect the policy or the relative ordering of Q-values; it only affects the absolute scale of the MLP's output.

This formulation eliminates the discount factor γ entirely. The discount factor has no natural interpretation in the curriculum setting—there is no reason to value early learning more than late learning—and exists in standard RL only to ensure contraction of the Bellman operator.

2.5 Algorithm

The key implementation insight is that each step's forward pass of candidates through the base model serves two purposes: action selection for the *current* step and the bootstrap target for the *previous* step's Q update. This avoids a redundant second forward pass.

Algorithm 1 Curriculum Selection via Differential Soft Q-Learning

Initialise: LM parameters θ; Q-network parameters ϕ; average reward $\rho \leftarrow 0$

// Bootstrap step (no Q update)
$C_0 \leftarrow \textsc{NextBatch}(\text{stream}, N)$
$H_0 \leftarrow \textsc{DetachedHidden}(\theta, C_0)$ ▷ $[N, d]$ — no grad into θ
$q_0 \leftarrow Q_\phi(H_0)$ ▷ $[N, 1]$
$a_0 \leftarrow \textsc{Sample}\big(\textsc{Softmax}(q_0/\beta)\big)$
$h_{\text{prev}} \leftarrow H_0[a_0]$ ▷ Store hidden state of selected action
$\theta \leftarrow \text{SGDStep}(\theta, C_0[a_0], \alpha, G)$ ▷ LM update, clipped
$r_{\text{prev}} \leftarrow \frac{1}{M} \log p_\theta(\hat{D}_0)$ ▷ Reward after update
$\rho \leftarrow r_{\text{prev}}$

for $k = 1, 2, \ldots$ **do** *// Main loop*
  $C_k \leftarrow \textsc{NextBatch}(\text{stream}, N)$
  $H_k \leftarrow \textsc{DetachedHidden}(\theta, C_k)$ ▷ Serves dual purpose below
  $q_k \leftarrow Q_\phi(H_k)$

  // Q update for previous transition (uses current forward pass as bootstrap)
  $V_k \leftarrow \beta \log \sum_{a'} \exp\big(q_k[a']/\beta\big)$ ▷ Soft value of current state
  $y \leftarrow r_{\text{prev}} - \rho + V_k$ ▷ TD target, stop-gradient
  $\hat{q} \leftarrow Q_\phi(h_{\text{prev}})$ ▷ Re-evaluate previous action with current ϕ
  Update ϕ to minimise $\frac{1}{2}(\hat{q} - y)^2$, clipped to $\|\nabla\| \leq G$

  // Action selection for current step
  $a_k \leftarrow \textsc{Sample}\big(\textsc{Softmax}(q_k/\beta)\big)$
  $h_{\text{prev}} \leftarrow H_k[a_k]$

  // LM update and reward
  $\theta \leftarrow \text{SGDStep}(\theta, C_k[a_k], \alpha, G)$
  $r_k \leftarrow \frac{1}{M} \log p_\theta(\hat{D}_k)$ ▷ Fresh $\hat{D}_k \subset D$, reward after update
  $\rho \leftarrow (1 - \tau)\, \rho + \tau\, r_k$
  $r_{\text{prev}} \leftarrow r_k$
end for

Computational cost per step. One forward pass of N candidates through the base model (batched, no gradient); one forward-backward pass of the selected example for LM training; one forward pass of M held-out examples for reward; one MLP forward-backward for the Q update. The candidate forward pass dominates; the MLP and reward computations are comparatively cheap.

Staleness. The Q update at step k uses q_k (computed before the ϕ update) for both the bootstrap target and action selection. This one-step staleness is standard in online Q-learning and negligible in practice given the small MLP learning rate η.

2.6 Interpretation as Free-Energy Minimisation

The Boltzmann policy is the solution to a free-energy minimisation problem at each state:

$$\pi^*(\cdot \mid s) = \arg\min_{\pi} \Big[-\mathbb{E}_{x\sim\pi}\big[Q_\phi(s,x)\big] + \beta\,\mathrm{KL}\big(\pi \,\|\, p_0\big) \Big], \tag{5}$$

where p_0 is the uniform distribution over C_k. Since p_0 is uniform, $\mathrm{KL}(\pi\|p_0) = \log N - H(\pi)$, and the objective reduces (up to constants) to:

$$\pi^*(\cdot \mid s) = \arg\min_{\pi} \Big[-\mathbb{E}_{x\sim\pi}\big[Q_\phi(s,x)\big] - \beta\,H(\pi) \Big]. \tag{6}$$

The first term encourages selecting high-value candidates; the second encourages exploration. The temperature β controls the tradeoff.

ELBO interpretation. Define a latent variable model where the "evidence" is the event that the current trajectory is optimal, with likelihood $p(\text{optimal} \mid x) \propto \exp(Q(s,x)/\beta)$. The policy π acts as a variational posterior over actions, and the free energy is the negative ELBO:

$$\log p(\text{optimal} \mid s) \;\geq\; \mathbb{E}_{x\sim\pi}\left[\frac{Q(s,x)}{\beta}\right] - \mathrm{KL}(\pi \,\|\, p_0). \tag{7}$$

The Boltzmann policy makes this bound tight. The soft value $V(s) = \beta \log \sum_x \exp(Q(s,x)/\beta)$—which appears as the bootstrap term in the TD target—is the log-evidence, measuring the total accessible value in the current candidate batch.

Gradient alignment. To first order in α, the one-step held-out improvement is:

$$r_k - r_{k-1} \;\approx\; \alpha\,\nabla_\theta \log p_\theta(D)^\top \nabla_\theta \log p_\theta(a_{k-1}). \tag{8}$$

The Q-function is therefore learning to predict which training examples have gradients aligned with the held-out set—not only at the current step, but accounting for how the gradient landscape will change over subsequent updates.

3 Evaluation

3.1 Primary Metric: Sample Efficiency

The central question is: how many training examples does the model need to reach a given performance level?

We will measure:

- Perplexity on a fixed evaluation set as a function of training examples seen
- Performance on downstream tasks (e.g., MMLU, HellaSwag) as a function of training examples

3.2 Baselines

- **Random curriculum:** Uniform sampling from candidates
- **Loss-based curriculum:** Prioritise high-loss examples
- **Uncertainty-based curriculum:** Prioritise examples with high model uncertainty
- **Competence-based curriculum:** Examples ordered by difficulty (requires difficulty labels)

3.3 Analysis

Beyond aggregate metrics, we will examine:

- **Curriculum structure:** Does the learned curriculum exhibit interpretable patterns? Developmental stages? Topic clustering?
- **Exploration dynamics:** How does the policy entropy evolve over training? Does the temperature β produce reasonable exploration?
- **Q-function interpretability:** Which candidates receive high Q-values at different stages of training? Does the Q-function learn to identify examples that are valuable given the model's current state?
- **Average reward dynamics:** How does ρ evolve over training? A rising ρ indicates the model is improving; the rate of rise measures curriculum effectiveness. Comparison of ρ trajectories across methods is a direct measure of sample efficiency.

4 Relation to Prior Work

4.1 Curriculum Learning

Graves et al. (2017) use multi-armed bandits to select tasks; Jiang et al. (2018) train a separate "mentor" network to weight examples. Our approach differs by using the language model's own representations as features for the Q-function and applying soft Q-learning with temporal credit assignment, rather than treating selection as a stateless bandit problem.

4.2 Meta-Learning

MAML (Finn et al., 2017) learns initializations for fast adaptation. We share the computational structure—reasoning about the effect of gradient updates—but learn *what* to train on rather than *where* to start.

4.3 RL as Inference

The control-as-inference framework (Levine, 2018; Rawlik et al., 2013) casts optimal control as probabilistic inference, with the Q-function serving as an energy function and the optimal policy as the corresponding Boltzmann distribution. Our algorithm instantiates this framework in the curriculum selection setting: the Q-network provides unnormalized log-probabilities over actions, and the Boltzmann policy is derived analytically without requiring a separate policy network. This is the soft Q-learning approach of Haarnoja et al. (2017), extended to the average reward setting.

4.4 Average Reward RL

The average reward formulation (Mahadevan, 1996; Schwartz, 1993) optimises the long-run reward rate rather than discounted return. It is the natural framework when there is no preferred time

scale and the discount factor would be an arbitrary hyperparameter. Our use of differential Q-values with a soft (entropy-regularised) Bellman equation combines the average reward framework with the maximum entropy RL of Haarnoja et al., avoiding both the unbounded returns of the undiscounted setting and the arbitrary time preference imposed by discounting.

4.5 Intrinsic Motivation

Our objective relates to compression progress (Schmidhuber) and active inference, but avoids the memory requirements of the former and the dark room problem of the latter by using a fixed held-out set as a proxy for predictive success. The absolute log-probability reward (1) provides a direct measure of predictive fitness, connecting to the epistemic homeostasis motivation: the agent is penalised for *being* in a state of poor prediction, not merely for failing to improve.

4.6 Connection to Prior Work

This proposal builds directly on Markovian Transformers for Informative Language Modeling (`https://arxiv.org/abs/2404.18988`), which introduces a framework for training language models with RL to produce causally load-bearing Chain-of-Thought reasoning. Both projects share a common structure: use RL to learn intermediate representations that improve prediction on held-out data. In the Markovian Transformers work, the learned object is a CoT:

$$\text{Question} \rightarrow \text{CoT} \rightarrow \text{Answer}$$

In the current proposal, the learned object is a curriculum:

$$\text{Model State} \rightarrow \text{Selected Data} \rightarrow \text{Improved Predictions}$$

The Markovian Transformers work demonstrates that this general approach is tractable and yields large gains on reasoning benchmarks (e.g., GSM8K: 19.6% $\rightarrow$ 57.1%). The current proposal extends this framework from learning *what to say* to learning *what to study*.

5 Broader Motivation

Language models trained to predict text develop remarkable capabilities from a simple objective. Yet they require extensive post-training to behave agentically and arguably lack a kind of global coherence. One hypothesis: the training process is purely observational—the model never takes actions that affect what it observes.

This project is a stepping stone toward studying whether learned curriculum selection produces qualitatively different agents. The immediate goal is demonstrating sample efficiency gains. The longer-term question is whether controlling one's own learning process contributes to the coherence and agency that current models seem to lack.

5.1 Long-Term Direction: Formalizing Homeostasis

Biological agents are shaped by survival pressures. Hunger, pain, and fatigue are not arbitrary reward signals—they are tied to the organism's continued existence. Current approaches to intrinsic motivation (curiosity, empowerment, compression progress) capture aspects of adaptive behavior but lack this grounding in self-preservation.

A long-term goal of this research program is to formalize homeostasis and survival into a simple, biologically plausible metric that could serve as a foundation for intrinsic motivation in artificial

systems. The current project—learning to select data that improves future prediction—is a minimal step in this direction: the agent takes actions that maintain its predictive capacity, a kind of epistemic homeostasis. The absolute reward formulation makes this connection explicit: the agent is directly penalised for poor predictive fitness at every moment, not merely for failing to improve.