The Differentiable Cross-Entropy Method

Brandon Amos¹  Denis Yarats¹²  
¹Facebook AI Research   ²New York University

ICML 2020
The cross-entropy method is a powerful optimizer

Iterative sampling-based optimizer that:

1. **Samples** from the domain
2. **Observes** the function’s values
3. **Updates** the sampling distribution

Widely used in **control** and **model-based RL**
Problem: CEM breaks end-to-end learning

A common learning pipeline, e.g. for control, is

1. Fit models with maximum likelihood
2. Run CEM on top of the learned models
3. Hope CEM induces reasonable downstream performance

Objective mismatch issue: models are unaware of downstream performance

Brandon Amos
The Differentiable Cross-Entropy Method
The Differentiable Cross-Entropy Method (DCEM)

Differentiate backwards through the sequence of samples
Using differentiable top-k (LML) and reparameterization

Useful when a fixed point is hard to find, or when unrolling gradient descent hits a local optimum

A differentiable controller in the RL setting
This Talk

Method: The differentiable-cross entropy method

Applications
- Learning deep energy-based models
- Learning embedded optimizers
- Control
Given \( g(x, y) \) and \( f(x) = g(x, y') \), where \( y' \in \{y: g(x, y) = 0\} \)

How can we compute \( D_x f(x) \)?

The Implicit Function Theorem gives

\[
D_x f(x) = -D_y g(x, f(x))^{-1} D_x g(x, f(x))
\]

under mild assumptions.
Foundation: Differentiable top-k operations

[Constrained softmax, constrained sparsemax, Limited Multi-Label Projection]

Optimization perspective of the softmax

$$y^* = \arg\min_y -y^T x - H(y)$$
subject to
$$0 \leq y \leq 1$$
$$1^T y = 1$$

Limited Multi-Label Projection

$$y^* = \arg\min_y -y^T x - H_b(y)$$
subject to
$$0 \leq y \leq 1$$
$$1^T y = k$$
In each iteration, update a distribution $g_\phi$ with:

$$ [X_{t,i}]_{i=1}^N \sim g_{\phi_t} (\cdot) \quad \text{Sample from the domain}$$

$$ v_{t,i} = f_\theta(X_{t,i}) \quad \text{Observe the function values}$$

$$ J_t = \Pi_{\mathcal{L}_k} (v_t / \tau) \quad \text{Compute the differentiable top-k}$$

Update $\phi_{t+1}$ with maximum weighted likelihood

And finally return $\mathbb{E}[g_{\phi_{T+1}} (\cdot)]$

Captures vanilla CEM when the soft top-k is hard
Composed of operations with informative derivatives
This Talk

Method: The differentiable-cross entropy method

Applications
- Learning deep energy-based models
- Learning embedded optimizers
- Control
Deep Structured Energy Models (SPENs/ICNNs)

Key idea: Model $p(x, y) \propto \exp\{-E_{\theta}(x, y)\}$ where $E_{\theta}$ is a deep energy model

Captures non-trivial structures in the output space, while also subsuming feed-forward modes

Feedforward model: $E(x, y) = ||f(x) - y||^2_2$

Predict with the optimization problem:

$$\hat{y} = \arg\min_y E_{\theta}(x, y)$$

Learning can be done by unrolling optimization on $E_{\theta}$ using derivative information $\nabla_y E$
Unrolling gradient descent may learn bad energies

Unrolling optimizers lose the probabilistic interpretation and can overfit to the optimizer.

In this regression setting, GD learns barriers on the energy surface while DCEM fits the data.
This Talk

Method: The differentiable-cross entropy method

Applications
- Learning deep energy-based models
- Learning embedded optimizers
- Control
DCEM can exploit the solution space structure

\[ x^* = \arg\min_{x \in [0,1]^N} f(x) \]

Brandon Amos

The Differentiable Cross-Entropy Method
This Talk

Method: The differentiable-cross entropy method

Applications
- Learning deep energy-based models
- Learning embedded optimizers
- Control
Should RL policies have a system dynamics model or not?

Model-free RL
More general, doesn’t make as many assumptions about the world
Rife with poor data efficiency and learning stability issues

Model-based RL (or control)
A useful prior on the world if it lies within your set of assumptions
Model Predictive Control

Known or learned from data

Cost
System Dynamics
Initial State

Model Predictive Control
Finds an optimal future trajectory

Optimal actions to take next
Differentiable Control via DCEM

A pure **planning problem** given (potentially non-convex) **cost** and **dynamics**:

\[ \tau_{1:T}^* = \arg\min_{\tau_{1:T}} \sum_t C_{\theta}(\tau_t) \text{Cost} \]

subject to

\[ x_1 = x_{\text{init}} \]
\[ x_{t+1} = f_{\theta}(\tau_t) \text{Dynamics} \]
\[ u_\ell \leq u \leq u_u \]

where \( \tau_t = \{x_t, u_t\} \)

**Idea:** Solve this optimization problem with DCEM and differentiate through it.
Differentiable Control via DCEM

A lot of data → Model → Predictions → Loss

... → Layer $z_i$ → DCEM → ...

What can we do with this now?

Augment neural network policies in model-free algorithms with MPC policies
Fight objective mismatch by end-to-end learning dynamics
The cost can also be end-to-end learned! No longer need to hard-code in values

Caveat: Control problems are often intractably high-dimensional, so we use embedded DCEM
DCEM fine-tunes highly non-convex controllers

sites.google.com/view/diff-cross-entropy-method
The Differentiable Cross-Entropy Method

Brandon Amos\textsuperscript{1} Denis Yarats\textsuperscript{2}
\textsuperscript{1}Facebook AI Research \textsuperscript{2}New York University

ICML 2020

\texttt{brandondamos}\texttt{bamos.github.io}\texttt{denisyarats}\texttt{cs.nyu.edu/~dy1042}