Perturbed optimizers for learning

Q. Berthet
(Google DeepMind)

“Differentiable Almost Everything” workshop

ICML 2023, Honolulu, Hawaii 🌺
End-to-end differentiable models

ML training: minimizing a loss w.r.t. weights of a model

- Composition of complex, explicit, analytical operations with weights.
- Loss minimization with first-order, gradient-based methods.
- Modern, large-scale models and datasets: automatic differentiation.
End-to-end differentiable models

**ML training:** minimizing a loss w.r.t. weights of a model

- Composition of complex, explicit, analytical operations with weights.
- Loss minimization with first-order, gradient-based methods.
- Modern, large-scale models and datasets: automatic differentiation.
Discrete operations

- Discrete operations / algorithms at the heart of computer science.
- Powerful tool to deal with structured problems / data.
Structured inference and prediction

Discrete operators allow us to leverage structure.

- Use of an implicit and discrete function, challenge for gradients
- Challenge: moving embeddings towards “correct solution” continuously
- In classification, soft “arg” max solution, smooth approximation.
Structured inference and prediction

Discrete operators allow us to leverage structure.

- Use of an implicit and discrete function, challenge for gradients
- Challenge: moving embeddings towards “correct solution” continuously
- In classification, soft “arg” max solution, smooth approximation.
Structured inference and prediction

Discrete operators allow us to leverage structure.

- Use of an implicit and discrete function, challenge for gradients
- Challenge: moving embeddings towards “correct solution” continuously
- In classification, soft “arg” max solution, smooth approximation.
Structured inference and prediction

Discrete operators allow us to leverage **structure**.

- Use of an **implicit** and **discrete** function, challenge for **gradients**
- Challenge: moving embeddings towards “correct solution” continuously
- In **classification**, soft “arg” max solution, smooth approximation.
Structured inference and prediction

Discrete operators allow us to leverage structure.

- Use of an implicit and discrete function, challenge for gradients
- Challenge: moving embeddings towards “correct solution” continuously
- In classification, soft “arg” max solution, smooth approximation.
Softmax

- **Softmax** a smooth, explicit approximation of "hard" argmax over **simplex**.
- Equivalent regularized / perturbed variational definition, with **closed form**

\[ y^*(\theta) = \arg\max_{y \in \mathcal{C}} (y, \theta), \quad [y^*(\theta)]_i = \delta_{i^*}(i). \]
Softmax

- **Softmax** a smooth, explicit approximation of “hard” argmax over **simplex**.
- Equivalent regularized / perturbed variational definition, with **closed form**

\[
y^*(\theta) = \arg\max_{y \in \mathcal{C}} \langle y, \theta \rangle, \quad [y^*(\theta)]_i = \delta_{i^*}(i).
\]
Softmax

- **Softmax** a smooth, explicit approximation of “hard” argmax over **simplex**.
- Equivalent regularized / perturbed variational definition, with **closed form**

\[
y^*(\theta) = \arg\max_{y \in \mathcal{C}} \langle y, \theta \rangle, \quad [y^*(\theta)]_i = \delta_{i^*}(i).
\]
• **Softmax** a smooth, explicit approximation of “hard” argmax over simplex.

• Equivalent **regularized** / perturbed variational definition, with **closed form**

\[
y^*_\epsilon(\theta) = \arg\max_{y \in \mathcal{C}} \langle y, \theta \rangle - \epsilon \Omega(y), \quad [y^*_\epsilon(\theta)]_i = \frac{e^{\theta_i/\epsilon}}{\sum_j e^{\theta_j/\epsilon}}.
\]
Softmax

- **Softmax** a smooth, explicit approximation of “hard” argmax over simplex.
- Equivalent regularized / perturbed variational definition, with closed form

\[ y^*_\varepsilon(\theta) = \mathbb{E}[^{\arg\max}_{y \in C} \langle y, \theta + \varepsilon Z \rangle], \quad [y^*_\varepsilon(\theta)]_i = \frac{e^{\theta_i/\varepsilon}}{\sum_j e^{\theta_j/\varepsilon}}. \]
Softmax

- **Softmax** a smooth, explicit approximation of “hard” argmax over *simplex*.
- Equivalent regularized / *perturbed* variational definition, with *closed form*

\[
y^*_\varepsilon(\theta) = \mathbb{E}[\arg\max_{y \in \mathcal{C}} \langle y, \theta + \varepsilon Z \rangle], \quad [y^*_\varepsilon(\theta)]_i = \frac{e^{\theta_i/\varepsilon}}{\sum_j e^{\theta_j/\varepsilon}}.
\]
• **Softmax** a smooth, explicit approximation of “hard” argmax over **simplex**.

• Equivalent regularized / **perturbed** variational definition, with **closed form**

$$ y^*_\epsilon(\theta) = \mathbb{E}_{y \in \mathcal{C}}[\argmax (y, \theta + \epsilon Z)] , \quad [y^*_\epsilon(\theta)]_i = \frac{e^{\theta_i/\epsilon}}{\sum_j e^{\theta_j/\epsilon}} . $$
• **Softmax** a smooth, explicit approximation of “hard” argmax over simplex.

• Equivalent regularized / **perturbed** variational definition, with **closed form**

\[
y^*_\varepsilon(\theta) = \mathbb{E}[\arg\max_{y \in \mathcal{C}} \langle y, \theta + \varepsilon Z \rangle], \quad [y^*_\varepsilon(\theta)]_i = \frac{e^{\theta_i/\varepsilon}}{\sum_j e^{\theta_j/\varepsilon}}.
\]
Generalized softmax (?)

- Most discrete optimizers can be naturally written as

$$y^*(\theta) = \arg\max_{y \in C} \langle y, \theta \rangle$$

- How to generalize these methods, to have **differentiable** proxies?
Generalized softmax (?)

- Most discrete optimizers can be naturally written as

\[ y^*(\theta) = \arg\max_{y \in C} \langle y, \theta \rangle \]

- How to generalize these methods, to have **differentiable** proxies?
Generalized softmax (?)

- Most discrete optimizers can be naturally written as

\[ y^*(\theta) = \arg\max_{y \in C} \langle y, \theta \rangle \]

- How to generalize these methods, to have \textbf{differentiable} proxies?
Generalized softmax (?)

- Most discrete optimizers can be naturally written as

\[ y^*(\theta) = \arg\max_{y \in C} \langle y, \theta \rangle \]

- How to generalize these methods, to have **differentiable** proxies?
Learning with Differentiable Perturbed Optimizers

NeurIPS 2020
**Perturbed maximizer**

**Discrete decisions:** optimizers of linear program over $C$, convex hull of $Y \subseteq \mathbb{R}^k$

\[
F(\theta) = \max_{y \in C} \langle y, \theta \rangle, \quad \text{and} \quad y^*(\theta) = \arg\max_{y \in C} \langle y, \theta \rangle = \nabla_{\theta} F(\theta).
\]

**Perturbed maximizer:** average of solutions for inputs with noise $\varepsilon Z$

\[
F_{\varepsilon}(\theta) = \mathbb{E}[\max_{y \in C} \langle y, \theta + \varepsilon Z \rangle], \quad y_{\varepsilon}^*(\theta) = \mathbb{E}[y^*(\theta + \varepsilon Z)] = \mathbb{E}[\arg\max_{y \in C} \langle y, \theta + \varepsilon Z \rangle] = \nabla_{\theta} F_{\varepsilon}(\theta).
\]
**Perturbed maximizer**

**Discrete decisions:** optimizers of linear program over $\mathcal{C}$, convex hull of $\mathcal{Y} \subseteq \mathbb{R}^k$

---

**Perturbed maximizer:** average of solutions for inputs with noise $\varepsilon Z$

$$ F_\varepsilon(\theta) = \mathbb{E}[\max_{y \in \mathcal{C}} \langle y, \theta + \varepsilon Z \rangle], \quad y_\varepsilon^*(\theta) = \mathbb{E}[y^*(\theta + \varepsilon Z)] = \mathbb{E}[\arg\max_{y \in \mathcal{C}} \langle y, \theta + \varepsilon Z \rangle] = \nabla_\theta F_\varepsilon(\theta). $$
Perturbed maximizer

**Discrete decisions:** optimizers of linear program over $\mathcal{C}$, convex hull of $\mathcal{Y} \subseteq \mathbb{R}^k$

\[
\begin{align*}
F_\varepsilon(\theta) &= \mathbb{E}_{y \in \mathcal{C}}[\max \langle y, \theta + \varepsilon Z \rangle ], \quad y_\varepsilon^*(\theta) = \mathbb{E}[y^*(\theta + \varepsilon Z)] = \mathbb{E}[\arg\max_{y \in \mathcal{C}} \langle y, \theta + \varepsilon Z \rangle ] = \nabla_\theta F_\varepsilon(\theta).
\end{align*}
\]
Perturbed maximizer

**Discrete decisions:** optimizers of linear program over \( C \), convex hull of \( Y \subseteq \mathbb{R}^k \)

\[
\begin{align*}
\mathbb{R}^k & \quad \bigcirc \\
& \quad \downarrow \\
& \quad \downarrow \\
\theta & \quad \xrightarrow{y^*_\varepsilon(\cdot)} \\
& \quad \bigcirc \\
C & \quad \downarrow \\
& \quad \bigcirc \\
\end{align*}
\]

**Perturbed maximizer:** average of solutions for inputs with noise \( \varepsilon Z \)

\[
F_\varepsilon(\theta) = \mathbb{E}[\max_{y \in C} \langle y, \theta + \varepsilon Z \rangle] , \quad y^*_\varepsilon(\theta) = \mathbb{E}[y^*(\theta + \varepsilon Z)] = \mathbb{E}[\arg\max_{y \in C} \langle y, \theta + \varepsilon Z \rangle] = \nabla_\theta F_\varepsilon(\theta).
\]
Perturbed maximizer

Discrete decisions: optimizers of linear program over $\mathcal{C}$, convex hull of $\mathcal{Y} \subseteq \mathbb{R}^k$

Perturbed maximizer: average of solutions for inputs with noise $\varepsilon Z$

$$F_\varepsilon(\theta) = \mathbb{E}[\max_{y \in \mathcal{C}} \langle y, \theta + \varepsilon Z \rangle], \quad y_\varepsilon^*(\theta) = \mathbb{E}[y^*(\theta + \varepsilon Z)] = \mathbb{E}[\arg\max_{y \in \mathcal{C}} \langle y, \theta + \varepsilon Z \rangle] = \nabla_{\theta} F_\varepsilon(\theta).$$
Perturbed model

Model of optimal decision under uncertainty Luce (1959), McFadden et al. (1973)

\[ Y = \arg\max_{y \in \mathcal{C}} \langle y, \theta + \varepsilon Z \rangle \]

Follows a perturbed model with \( Y \sim p_\theta(y) \), expectation \( y^*_\varepsilon(\theta) = E_{p_\theta}[Y] \).


![Diagram of perturbed models and shortest paths]

**Example.** Over the unit simplex \( \mathcal{C} = \Delta^d \) with Gumbel noise \( Z \), \( F(\theta) = \max_i \theta_i \).

\[ F_\varepsilon(\theta) = \varepsilon \log \sum_{i \in [d]} e^{\theta_i/\varepsilon}, \quad p_\theta(e_i) \propto \exp(\langle \theta, e_i \rangle/\varepsilon), \quad [y^*_\varepsilon(\theta)]_i = \frac{e^{\theta_i/\varepsilon}}{\sum_j e^{\theta_j/\varepsilon}} \]
Why? and How?

Learning problems:

Features $X_i$, model output $\theta_w = g_w(X_i)$, prediction $y_{\text{pred}} = y^*_\varepsilon(\theta_w)$, loss $L$

$$F(w) = L(y^*_\varepsilon(\theta_w), y_i), \quad \text{gradients require } \partial_\theta y^*_\varepsilon(\theta_w).$$

Monte Carlo estimates. Perturbed maximizer and derivatives as expectations.

For $\theta \in \mathbb{R}^d$, $Z^{(1)}, \ldots, Z^{(M)}$ i.i.d. copies

$$y^{(\ell)} = y^*(\theta + \varepsilon Z^{(\ell)})$$

Unbiased estimate of $y^*_\varepsilon(\theta)$ given by

$$\bar{y}_{\varepsilon,M}(\theta) = \frac{1}{M} \sum_{\ell=1}^M y^{(\ell)}.$$
Properties

**Mirror maps:** For $C$ with full interior, $Z$ with smooth density $\mu$, full support $F_\varepsilon$ strictly convex, gradient Lipschitz. $\Omega$ strongly convex, Legendre type.

**Differentiability.** Functions are smooth in the inputs. For $\mu(z) \propto \exp(-\nu(z))$

$$y_\varepsilon^*(\theta) = \nabla_\theta F_\varepsilon(\theta) = E[y^*(\theta + \varepsilon Z)] = E[F(\theta + \varepsilon Z)\nabla_z \nu(Z) / \varepsilon],$$
$$\partial_\theta y_\varepsilon^*(\theta) = \nabla^2 F_\varepsilon(\theta) = E[y^*(\theta + \varepsilon Z)\nabla_z \nu(Z)^\top / \varepsilon].$$

Perturbed maximizer $y_\varepsilon^*$ never locally constant in $\theta$. Abernethy et al. (2014)
Properties

Mirror maps: For $C$ with full interior, $Z$ with smooth density $\mu$, full support $F_\varepsilon$ strictly convex, gradient Lipschitz. $\Omega$ strongly convex, Legendre type.

Differentiability. Functions are smooth in the inputs. For $\mu(z) \propto \exp(-\nu(z))$

$$y_\varepsilon^*(\theta) = \nabla_\theta F_\varepsilon(\theta) = \mathbb{E}[y^*(\theta + \varepsilon Z)] = \mathbb{E}[F(\theta + \varepsilon Z)\nabla_z \nu(Z)/\varepsilon],$$

$$\partial_\theta y_\varepsilon^*(\theta) = \nabla^2 F_\varepsilon(\theta) = \mathbb{E}[y^*(\theta + \varepsilon Z)\nabla_z \nu(Z)^\top/\varepsilon].$$

Perturbed maximizer $y_\varepsilon^*$ never locally constant in $\theta$. Abernethy et al. (2014)
Learning with perturbed optimizers

Machine learning pipeline: variable $X$, discrete label $y$, model outputs $\theta = g_w(X)$

Labels are solutions of optimization problems (one-hots, ranks, shortest paths)

Small modification of the model: end-to-end differentiable
Learning with perturbed optimizers

Machine learning pipeline: variable $X$, discrete label $y$, model outputs $\theta = g_w(X)$

Labels are solutions of optimization problems (one-hots, ranks, shortest paths)

Small modification of the model: end-to-end differentiable
### Learning with perturbed optimizers

Machine learning pipeline: variable $X$, discrete label $y$, model outputs $\theta = g_w(X)$

Labels are solutions of optimization problems (one-hots, ranks, shortest paths)

Small modification of the model: end-to-end differentiable
Learning with perturbations and Fenchel-Young losses

Within the same framework, possible to virtually bypass the optimization block

Fenchel-Young losses Easier to implement, no Jacobian of $y^*_\epsilon$. Blondel et al (20)

Population loss minimized at ground truth for perturbed generative model.
Motivated by model where \( y_i = \arg\max_{y \in C} \langle g_w(X_i) + \varepsilon Z_i, y \rangle \)

Stochastic gradients for empirical loss only require

\[
\nabla_{\theta} L(\theta = g_w(X_i); y_i) = y^*_\varepsilon(\theta) - y_i = y^*_\varepsilon(g_w(X_i)) - y_i.
\]

Simulated by a doubly stochastic scheme.
**Computations**

**Monte Carlo estimates.** Perturbed maximizer and derivatives as expectations.

For $\theta \in \mathbb{R}^d$, $Z^{(1)}, \ldots, Z^{(M)}$ i.i.d. copies

$$y^{(\ell)} = y^*(\theta + \varepsilon Z^{(\ell)})$$

Unbiased estimate of $y^*_\varepsilon(\theta)$ given by

$$\bar{y}_{\varepsilon,M}(\theta) = \frac{1}{M} \sum_{\ell=1}^{M} y^{(\ell)}.$$

**Supervised learning:**

Features $X_i$, model output $\theta_w = g_w(X_i)$, prediction $y_{\text{pred}} = y^*_\varepsilon(\theta_w)$.

Stochastic gradient in $w$:

$$\nabla_w F_i(w) = \partial_w g_w(X_i) \cdot (y^*_\varepsilon(\theta) - Y_i)$$
Computations

Monte Carlo estimates. Perturbed maximizer and derivatives as expectations.

For $\theta \in \mathbb{R}^d$, $Z^{(1)}, \ldots, Z^{(M)}$ i.i.d. copies

$$y^{(\ell)} = y^*(\theta + \varepsilon Z^{(\ell)})$$

Unbiased estimate of $y^*_\varepsilon(\theta)$ given by

$$\bar{y}_{\varepsilon,M}(\theta) = \frac{1}{M} \sum_{\ell=1}^{M} y^{(\ell)}.$$

Supervised learning:

Features $X_i$, model output $\theta_w = g_w(X_i)$, prediction $y_{\text{pred}} = y^*_\varepsilon(\theta_w)$.

Stochastic gradient in $w$ (doubly stochastic scheme)

$$\nabla_w F_i(w) = \partial_w g_w(X_i) \cdot \left( \frac{1}{M} \sum_{\ell=1}^{M} y^*(\theta + \varepsilon Z^{(\ell)}) - Y_i \right).$$
Experiments

**Learning to rank:** Experiments on 4k instances of 100 vectors to rank.

Robustness to noise observed for some tolerated variance

Fenchel-Young loss is convex in \( w \): linear model, possible theoretical analysis.
Experiments

Learning from shortest paths: From 10k examples of Warcraft $96 \times 96$ RGB images, representing $12 \times 12$ costs, and matrix of shortest paths. (Vlastelica et al. 19)

Train a CNN for 50 epochs, to learn costs recovery of optimal paths.
● Deep embedding and alignment of protein sequences

Nature methods, 2023
Deep embedding and alignment of protein sequences

Nature methods, 2023
Protein alignment

- Learning character-wise embeddings of protein sequences.
- Using it to compute costs of an alignment problem (dynamic programming).
Protein alignment

- Learning character-wise embeddings of protein sequences.
- Using it to compute costs of an alignment problem (dynamic programming).
Differentiable alignment

- Given fixed substitution/insertion costs, local alignment problem.
- Smith-Waterman problem solved by DP, to align proteins.
- Non-differentiable solution, introducing perturbations

Alignment as a biologically-plausible sequence similarity measure

Source: Dr. Vered Caspi
Differentiable alignment

- Given fixed substitution/insertion costs, local alignment problem.
- Smith-Waterman problem solved by DP, to align proteins.
- Non-differentiable solution, introducing perturbations

Differentiable Smith-Waterman alignments

Technique based on team’s methodological work
Yields a differentiable version of the alignment algorithm
Differentiable alignment

- Given fixed substitution/insertion costs, local alignment problem.
- Smith-Waterman problem solved by DP, to align proteins.
- Non-differentiable solution, introducing perturbations.

Differentiable Smith-Waterman alignments

Technique based on team’s methodological work
Yields a differentiable version of the alignment algorithm
DEDAL: End-to-end learning to align

- From sequences to embeddings, costs, to perturbed alignments.
- Transformer architecture trained on databases of aligned proteins.
Self-supervised learning of audio representations from permutations with differentiable ranking

A. Carr, Q. Berthet, M. Blondel, O. Teboul, N. Zeghidour
IEEE Signal Processing Letters, 2021
• **Applications:**

  Digital pathology Thandiackal et al. (22), Patch selection Cordonnier et al. (21), Video Token Selection Wang et al. (22), . . .

• **Algorithmic improvements:** parallelized optimization Dubois-Taine et al. (22)
Differentiable Clustering with Perturbed Spanning Forests

Preprint, 2023
Differentiable Clustering

- Transformer architecture trained on databases of aligned proteins.

- Semi-supervised clustering.

- Discovery of held-out classes.

- **Presentation** and **poster** today.
Mahalo!