# Adaptive Perturbation-Based Gradient Estimation for Discrete Latent Variable Models

Pasquale Minervini, Luca Franceschi, Mathias Niepert

UCL · The University of Edinburgh · University of Stuttgart Germany

## Combinatorial Solvers

We consider the problem of back-propagating through a combinatorial solver, i.e., a (black-box) component that works as follows:

E.g., shortest path

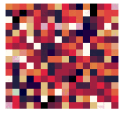
$\theta \mapsto \text{solve}(\theta)$ such that $\text{solve}(\theta) = \arg\max_{z \in \mathscr{C}} \text{score}(\theta, z)$

Input weights

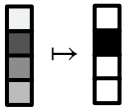
with $\text{score}(\theta, z) = \theta^\top z$

Linear score function

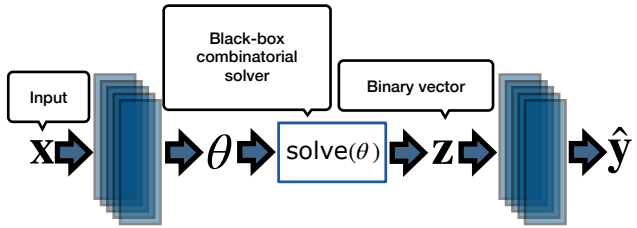
E.g., the set of all paths

$\theta =$  $\mathbf{z} = \begin{bmatrix} 1 & 0 & \dots & 0 \\ 1 & 0 & \dots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \dots & 1 \end{bmatrix}$

**Examples of combinatorial solvers:**
- Top-$k$ functions
- Shortest path algorithms
- Maximum spanning three algorithms
- Scheduling algorithms
- Supply chain optimisation algorithms
- .. and many more

Top-1: 

## Back-Propagating through Combinatorial Solvers




**Problem:** How do we estimate the gradient of the loss wrt. $\theta$, $\nabla_\theta \mathscr{L}(\hat{\mathbf{y}}, \mathbf{y})$?

## Combinatorial Solvers and Exponential Families

Invoking a **combinatorial solver** $\mathbf{z} \leftarrow \text{solve}(\theta)$ on $\theta \in \mathbb{R}^n$ to obtain a **discrete solution** $\mathbf{z} \in \{0,1\}^n$ is equivalent to computing the MAP state of an **exponential family distribution** $p(\mathbf{z}; \theta)$:

e.g., top-$k$, shortest path algorithm, maximum spanning tree algorithm..

Maximum a Posteriori (MAP) estimation of $\mathbf{z}$ wrt. $p(\cdot; \theta)$

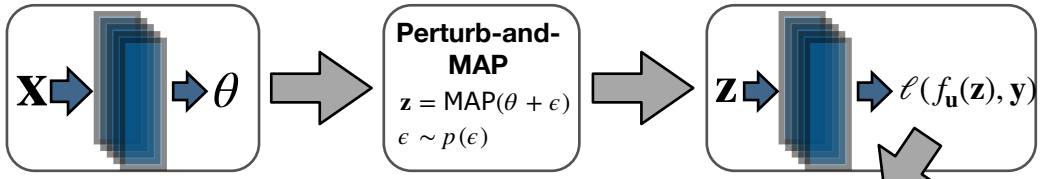
$\mathbf{z} \leftarrow \text{solve}(\theta) \equiv \mathbf{z} \leftarrow \arg\max_{\overline{\mathbf{z}}} p(\overline{\mathbf{z}}; \theta)$

Space of solutions

with $p(\mathbf{z}; \theta) = \begin{cases} \exp(\langle \mathbf{z}, \theta \rangle) - Z(\theta) & \text{if } \mathbf{z} \in \mathscr{C} \\ 0 & \text{otherwise} \end{cases}$

## Implicit MLE In a Nutshell

Allows to efficiently sample from $p$: $\mathbf{z} \sim p(\mathbf{z}; \theta)$




**Approximate MLE Gradients**

$\nabla_\theta L \approx \frac{1}{\lambda} [\text{MAP}(\theta + \epsilon) - \text{MAP}(\theta' + \epsilon)]$

**Construct Target Distribution**

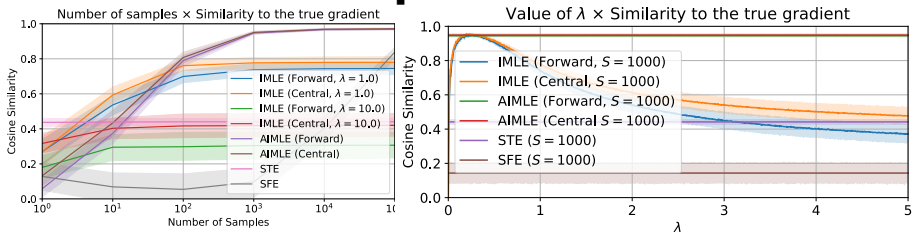
$q(\mathbf{z}; \theta') = p(\mathbf{z}; \theta - \lambda \nabla_{\mathbf{z}} L(f_{\mathbf{u}}(\overline{\mathbf{z}}), y))$

**Problem:** How do we select $\lambda$?

## Adaptive Implicit MLE

Very simple idea:
- Initialise $\lambda \leftarrow 0$
- During training, **adaptively change** $\lambda$ until the gradient estimates satisfy some **desired sparsity criteria**

## Experiments




Learning to Explain - Aroma, K = 10