

# Discrete Latent Variable Models

Stefano Ermon, Aditya Grover

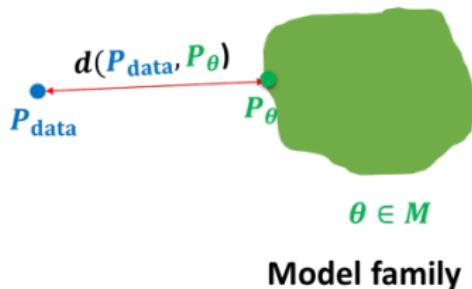
Stanford University

Lecture 15

# Summary



$$\mathbf{x}_i \sim P_{\text{data}} \\ i = 1, 2, \dots, n$$



Major themes in the course

- Representation: Latent variable vs. fully observed
- Objective function and optimization algorithm: Many divergences and distances optimized via likelihood-free (two sample test) or likelihood based methods
- Evaluation of generative models
- Combining different models and variants

Plan for today: Discrete Latent Variable Modeling

# Why should we care about discreteness?

- Discreteness is all around us!
  - Decision Making: Should I attend CS 236 lecture or not?
  - Structure learning

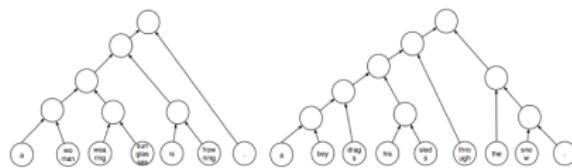


Figure 2: Examples of tree structures learned by our model which show that the model discovers simple concepts such as noun phrases and verb phrases.



Figure 3: Examples of unconventional tree structures.

Source: Yogatama et al., 2017

# Why should we care about discreteness?

- Many data modalities are inherently discrete
  - Graphs



- Text, DNA Sequences, Program Source Code, Molecules, and lots more

# Stochastic Optimization

- Consider the following optimization problem

$$\max_{\phi} E_{q_{\phi}(\mathbf{z})}[f(\mathbf{z})]$$

- Recap example: Think of  $q(\cdot)$  as the inference distribution for a VAE

$$\max_{\theta, \phi} E_{q_{\phi}(\mathbf{z}|\mathbf{x})} \left[ \log \frac{p_{\theta}(\mathbf{x}, \mathbf{z})}{q(\mathbf{z}|\mathbf{x})} \right].$$

- Gradients w.r.t.  $\theta$  can be derived via linearity of expectation

$$\begin{aligned}\nabla_{\theta} E_{q(\mathbf{z}; \phi)} [\log p(\mathbf{z}, \mathbf{x}; \theta) - \log q(\mathbf{z}; \phi)] &= E_{q(\mathbf{z}; \phi)} [\nabla_{\theta} \log p(\mathbf{z}, \mathbf{x}; \theta)] \\ &\approx \frac{1}{k} \sum_k \nabla_{\theta} \log p(\mathbf{z}^k, \mathbf{x}; \theta)\end{aligned}$$

- If  $\mathbf{z}$  is continuous,  $q(\cdot)$  is reparameterizable, and  $f(\cdot)$  is differentiable in  $\phi$ , then we can use reparameterization to compute gradients w.r.t.  $\phi$
- What if any of the above assumptions fail?

# Stochastic Optimization with REINFORCE

- Consider the following optimization problem

$$\max_{\phi} E_{q_{\phi}(z)}[f(z)]$$

- For many class of problem scenarios, reparameterization trick is inapplicable
- Scenario 1:**  $f(\cdot)$  is non-differentiable in  $\phi$  e.g., optimizing a black box reward function in reinforcement learning
- Scenario 2:**  $q_{\phi}(z)$  cannot be reparameterized as a differentiable function of  $\phi$  with respect to a fixed base distribution e.g., discrete distributions
- REINFORCE is a general-purpose solution to both these scenarios
- We will first analyze it in the context of reinforcement learning and then extend it to latent variable models with discrete latent variables

# REINFORCE for reinforcement learning



- Example: Pulling arms of slot machines Which arm to pull?
- Set  $A$  of possible actions. E.g., pull arm 1, arm 2, . . . , etc.
- Each action  $\mathbf{z} \in A$  has a reward  $f(\mathbf{z})$
- Randomized policy for choosing actions  $q_\phi(\mathbf{z})$  parameterized by  $\phi$ .  
For example,  $\phi$  could be the parameters of a multinomial distribution
- **Goal:** Learn the parameters  $\phi$  that maximize our earnings (in expectation)

$$\max_{\phi} E_{q_\phi(\mathbf{z})}[f(\mathbf{z})]$$

# Policy Gradients

- Want to compute a gradient with respect to  $\phi$  of the expected reward

$$E_{q_\phi(\mathbf{z})}[f(\mathbf{z})] = \sum_{\mathbf{z}} q_\phi(\mathbf{z}) f(\mathbf{z})$$

$$\begin{aligned} \frac{\partial}{\partial \phi_i} E_{q_\phi(\mathbf{z})}[f(\mathbf{z})] &= \sum_{\mathbf{z}} \frac{\partial q_\phi(\mathbf{z})}{\partial \phi_i} f(\mathbf{z}) = \sum_{\mathbf{z}} q_\phi(\mathbf{z}) \frac{1}{q_\phi(\mathbf{z})} \frac{\partial q_\phi(\mathbf{z})}{\partial \phi_i} f(\mathbf{z}) \\ &= \sum_{\mathbf{z}} q_\phi(\mathbf{z}) \frac{\partial \log q_\phi(\mathbf{z})}{\partial \phi_i} f(\mathbf{z}) = E_{q_\phi(\mathbf{z})} \left[ \frac{\partial \log q_\phi(\mathbf{z})}{\partial \phi_i} f(\mathbf{z}) \right] \end{aligned}$$

# REINFORCE Gradient Estimation

- Want to compute a gradient with respect to  $\phi$  of

$$E_{q_\phi(\mathbf{z})}[f(\mathbf{z})] = \sum_{\mathbf{z}} q_\phi(\mathbf{z}) f(\mathbf{z})$$

- The REINFORCE rule is

$$\nabla_\phi E_{q_\phi(\mathbf{z})}[f(\mathbf{z})] = E_{q_\phi(\mathbf{z})} [f(\mathbf{z}) \nabla_\phi \log q_\phi(\mathbf{z})]$$

- We can now construct a Monte Carlo estimate
- Sample  $\mathbf{z}^1, \dots, \mathbf{z}^K$  from  $q_\phi(\mathbf{z})$  and estimate

$$\nabla_\phi E_{q_\phi(\mathbf{z})}[f(\mathbf{z})] \approx \frac{1}{K} \sum_k f(\mathbf{z}^k) \nabla_\phi \log q_\phi(\mathbf{z}^k)$$

- Assumption: The distribution  $q(\cdot)$  is easy to sample from and evaluate probabilities
- Works for both discrete and continuous distributions

# Variational Learning of Latent Variable Models

- To learn the variational approximation we need to compute the gradient with respect to  $\phi$  of

$$\begin{aligned}\mathcal{L}(\mathbf{x}; \theta, \phi) &= \sum_{\mathbf{z}} q_{\phi}(\mathbf{z}|\mathbf{x}) \log p(\mathbf{z}, \mathbf{x}; \theta) + H(q_{\phi}(\mathbf{z}|\mathbf{x})) \\ &= E_{q_{\phi}(\mathbf{z}|\mathbf{x})}[\log p(\mathbf{z}, \mathbf{x}; \theta) - \log q_{\phi}(\mathbf{z}|\mathbf{x})]\end{aligned}$$

- The function inside the brackets also depends on  $\phi$  (and  $\theta, \mathbf{x}$ ). Want to compute a gradient with respect to  $\phi$  of

$$E_{q_{\phi}(\mathbf{z}|\mathbf{x})}[f(\phi, \theta, \mathbf{z}, \mathbf{x})] = \sum_{\mathbf{z}} q_{\phi}(\mathbf{z}|\mathbf{x}) f(\phi, \theta, \mathbf{z}, \mathbf{x})$$

- The REINFORCE rule is

$$\nabla_{\phi} E_{q_{\phi}(\mathbf{z}|\mathbf{x})}[f(\phi, \theta, \mathbf{z}, \mathbf{x})] = E_{q_{\phi}(\mathbf{z}|\mathbf{x})} [f(\phi, \theta, \mathbf{z}, \mathbf{x}) \nabla_{\phi} \log q_{\phi}(\mathbf{z}|\mathbf{x}) + \nabla_{\phi} f(\phi, \theta, \mathbf{z}, \mathbf{x})]$$

- We can now construct a Monte Carlo estimate of  $\nabla_{\phi} \mathcal{L}(\mathbf{x}; \theta, \phi)$

# REINFORCE Gradient Estimates have High Variance

- Want to compute a gradient with respect to  $\phi$  of

$$E_{q_\phi(\mathbf{z})}[f(\mathbf{z})] = \sum_{\mathbf{z}} q_\phi(\mathbf{z}) f(\mathbf{z})$$

- The REINFORCE rule is

$$\nabla_\phi E_{q_\phi(\mathbf{z})}[f(\mathbf{z})] = E_{q_\phi(\mathbf{z})} [f(\mathbf{z}) \nabla_\phi \log q_\phi(\mathbf{z})]$$

- Monte Carlo estimate: Sample  $\mathbf{z}^1, \dots, \mathbf{z}^K$  from  $q_\phi(\mathbf{z})$

$$\nabla_\phi E_{q_\phi(\mathbf{z})}[f(\mathbf{z})] \approx \frac{1}{K} \sum_k f(\mathbf{z}^k) \nabla_\phi \log q_\phi(\mathbf{z}^k) := f_{\text{MC}}(\mathbf{z}^1, \dots, \mathbf{z}^K)$$

- Monte Carlo estimates of gradients are unbiased

$$E_{\mathbf{z}^1, \dots, \mathbf{z}^K \sim q_\phi(\mathbf{z})} [f_{\text{MC}}(\mathbf{z}^1, \dots, \mathbf{z}^K)] = \nabla_\phi E_{q_\phi(\mathbf{z})}[f(\mathbf{z})]$$

- Almost never used in practice because of high variance
- Variance can be reduced via carefully designed control variates

# Control Variates

- The REINFORCE rule is

$$\nabla_{\phi} E_{q_{\phi}(\mathbf{z})}[f(\mathbf{z})] = E_{q_{\phi}(\mathbf{z})} [f(\mathbf{z}) \nabla_{\phi} \log q_{\phi}(\mathbf{z})]$$

- Given any constant  $B$  (a control variate)

$$\nabla_{\phi} E_{q_{\phi}(\mathbf{z})}[f(\mathbf{z})] = E_{q_{\phi}(\mathbf{z})} [(f(\mathbf{z}) - B) \nabla_{\phi} \log q_{\phi}(\mathbf{z})]$$

- To see why,

$$\begin{aligned} E_{q_{\phi}(\mathbf{z})} [B \nabla_{\phi} \log q_{\phi}(\mathbf{z})] &= B \sum_{\mathbf{z}} q_{\phi}(\mathbf{z}) \nabla_{\phi} \log q_{\phi}(\mathbf{z}) = B \sum_{\mathbf{z}} \nabla_{\phi} q_{\phi}(\mathbf{z}) \\ &= B \nabla_{\phi} \sum_{\mathbf{z}} q_{\phi}(\mathbf{z}) = B \nabla_{\phi} 1 = 0 \end{aligned}$$

- Monte Carlo gradient estimates of both  $f(\mathbf{z})$  and  $f(\mathbf{z}) - B$  have same expectation
- These estimates can however have different variances

# Control variates

- Suppose we want to compute

$$E_{q_\phi(\mathbf{z})}[f(\mathbf{z})] = \sum_{\mathbf{z}} q_\phi(\mathbf{z}) f(\mathbf{z})$$

- Define

$$\hat{f}(\mathbf{z}) = f(\mathbf{z}) + a (h(\mathbf{z}) - E_{q_\phi(\mathbf{z})}[h(\mathbf{z})])$$

- $h(\mathbf{z})$  is referred to as a control variate
- Assumption:  $E_{q_\phi(\mathbf{z})}[h(\mathbf{z})]$  is known
- Monte Carlo gradient estimates of  $f(\mathbf{z})$  and  $\hat{f}(\mathbf{z})$  have the same expectation

$$E_{\mathbf{z}^1, \dots, \mathbf{z}^K \sim q_\phi(\mathbf{z})}[\hat{f}_{\text{MC}}(\mathbf{z}^1, \dots, \mathbf{z}^K)] = E_{\mathbf{z}^1, \dots, \mathbf{z}^K \sim q_\phi(\mathbf{z})}[f_{\text{MC}}(\mathbf{z}^1, \dots, \mathbf{z}^K)]$$

but different variances

- Can try to learn and update the control variate during training

# Control variates

- Can derive an alternate Monte Carlo estimate for REINFORCE gradients based on control variates
- Sample  $\mathbf{z}^1, \dots, \mathbf{z}^K$  from  $q_\phi(\mathbf{z})$

$$\begin{aligned}& \nabla_\phi E_{q_\phi(\mathbf{z})}[f(\mathbf{z})] \\&= \nabla_\phi E_{q_\phi(\mathbf{z})}[f(\mathbf{z}) + a(h(\mathbf{z}) - E_{q_\phi(\mathbf{z})}[h(\mathbf{z})])] \\&\approx \frac{1}{K} \sum_k f(\mathbf{z}^k) \nabla_\phi \log q_\phi(\mathbf{z}^k) + a \left( \frac{1}{K} \sum_{k=1}^K h(\mathbf{z}^k) - E_{q_\phi(\mathbf{z})}[h(\mathbf{z})] \right) \\&:= f_{\text{MC}}(\mathbf{z}^1, \dots, \mathbf{z}^K) + a \left( h_{\text{MC}}(\mathbf{z}^1, \dots, \mathbf{z}^K) - E_{q_\phi(\mathbf{z})}[h(\mathbf{z})] \right) \\&:= \hat{f}_{\text{MC}}(\mathbf{z}^1, \dots, \mathbf{z}^K)\end{aligned}$$

- What is  $\text{Var}(\hat{f}_{\text{MC}})$  vs.  $\text{Var}(f_{\text{MC}})$ ?

# Control variates

- Comparing  $\text{Var}(\hat{f}_{\text{MC}})$  vs.  $\text{Var}(f_{\text{MC}})$

$$\begin{aligned}\text{Var}(\hat{f}_{\text{MC}}) &= \text{Var}(f_{\text{MC}} + a(h_{\text{MC}} - E_{q_\phi(z)}[h(z)])) \\ &= \text{Var}(f_{\text{MC}} + ah_{\text{MC}}) \\ &= \text{Var}(f_{\text{MC}}) + a^2 \text{Var}(h_{\text{MC}}) + 2a \text{Cov}(f_{\text{MC}}, h_{\text{MC}})\end{aligned}$$

- To get the optimal coefficient  $a^*$  that minimizes the variance, take derivatives w.r.t.  $a$  and set them to 0

$$a^* = -\frac{\text{Cov}(f_{\text{MC}}, h_{\text{MC}})}{\text{Var}(h_{\text{MC}})}$$

## Control variates

- Comparing  $\text{Var}(\hat{f}_{\text{MC}})$  vs.  $\text{Var}(f_{\text{MC}})$

$$\text{Var}(\hat{f}_{\text{MC}}) = \text{Var}(f_{\text{MC}}) + a^2 \text{Var}(h_{\text{MC}}) + 2a \text{Cov}(f_{\text{MC}}, h_{\text{MC}})$$

- Setting the coefficient  $a = a^* = -\frac{\text{Cov}(f_{\text{MC}}, h_{\text{MC}})}{\text{Var}(h_{\text{MC}})}$

$$\begin{aligned}\text{Var}(\hat{f}_{\text{MC}}) &= \text{Var}(f_{\text{MC}}) - \frac{\text{Cov}(f_{\text{MC}}, h_{\text{MC}})^2}{\text{Var}(h_{\text{MC}})} \\ &= \text{Var}(f_{\text{MC}}) - \frac{\text{Cov}(f_{\text{MC}}, h_{\text{MC}})^2}{\text{Var}(h_{\text{MC}})\text{Var}(f_{\text{MC}})} \text{Var}(f_{\text{MC}}) \\ &= (1 - \rho(f_{\text{MC}}, h_{\text{MC}})^2) \text{Var}(f_{\text{MC}})\end{aligned}$$

- Correlation coefficient  $\rho(f_{\text{MC}}, h_{\text{MC}})$  is between -1 and 1. For maximum variance reduction, we want  $f_{\text{MC}}$  and  $h_{\text{MC}}$  to be highly correlated

# Neural Variational Inference and Learning (NVIL)

- Latent variable models with discrete latent variables are often referred to as belief networks
- Variational learning objective is same as ELBO

$$\begin{aligned}\mathcal{L}(\mathbf{x}; \theta, \phi) &= \sum_{\mathbf{z}} q_{\phi}(\mathbf{z}|\mathbf{x}) \log p(\mathbf{z}, \mathbf{x}; \theta) + H(q_{\phi}(\mathbf{z}|\mathbf{x})) \\ &= E_{q_{\phi}(\mathbf{z}|\mathbf{x})}[\log p(\mathbf{z}, \mathbf{x}; \theta) - \log q_{\phi}(\mathbf{z}|\mathbf{x})] \\ &:= E_{q_{\phi}(\mathbf{z}|\mathbf{x})}[f(\phi, \theta, \mathbf{z}, \mathbf{x})]\end{aligned}$$

- Here,  $\mathbf{z}$  is discrete and hence we cannot use reparameterization

# Neural Variational Inference and Learning (NVIL)

- NVIL (Mnih&Gregor, 2014) learns belief networks via REINFORCE + control variates
- Learning objective

$$\mathcal{L}(\mathbf{x}; \theta, \phi, \psi, B) = E_{q_\phi(\mathbf{z}|\mathbf{x})}[f(\phi, \theta, \mathbf{z}, \mathbf{x}) - h_\psi(\mathbf{x}) - B]$$

- **Control Variate 1:** Constant baseline  $B$
- **Control Variate 2:** Input dependent baseline  $h_\psi(\mathbf{x})$
- Both  $B$  and  $\psi$  are learned via gradient descent
- Gradient estimates w.r.t.  $\phi$

$$\begin{aligned} & \nabla_\phi \mathcal{L}(\mathbf{x}; \theta, \phi, \psi, B) \\ &= E_{q_\phi(\mathbf{z}|\mathbf{x})} [(f(\phi, \theta, \mathbf{z}, \mathbf{x}) - h_\psi(\mathbf{x}) - B) \nabla_\phi \log q_\phi(\mathbf{z}|\mathbf{x}) + \nabla_\phi f(\phi, \theta, \mathbf{z}, \mathbf{x})] \end{aligned}$$

# Towards reparameterized, continuous relaxations

- Consider the following optimization problem

$$\max_{\phi} E_{q_{\phi}(z)}[f(z)]$$

- What if  $z$  is a discrete random variable?
  - Categories
  - Permutations
- Reparameterization trick is not directly applicable
- REINFORCE is a general-purpose solution, but needs careful design of control variates
- Today:** Relax  $z$  to a continuous random variable with a reparameterizable distribution

# Gumbel Distribution

- Setting: We are given i.i.d. samples  $y_1, y_2, \dots, y_n$  from some underlying distribution. How can we model the distribution of

$$g = \max\{y_1, y_2, \dots, y_n\}$$

- E.g., predicting maximum water level in a river for a particular river based on historical data to detect flooding
- The **Gumbel distribution** is very useful for modeling extreme, rare events, e.g., natural disasters, finance
- CDF for a Gumbel random variable  $g$  is parameterized by a location parameter  $\mu$  and a scale parameter  $\beta$

$$F(g; \mu, \beta) = \exp \left( -\exp \left( -\frac{g - \mu}{\beta} \right) \right)$$

- Note: If  $g$  is a  $\text{Gumbel}(\mu, \beta)$  r.v.,  $-\log g$  is an  $\text{Exponential}(\mu, \beta)$  r.v. Often, Gumbel r.v. are referred to as doubly exponential r.v.

# Categorical Distributions

- Let  $\mathbf{z}$  denote a  $k$ -dimensional categorical random variable with distribution  $q$  parameterized by class probabilities  $\pi = \{\pi_1, \pi_2, \dots, \pi_k\}$ . We will represent  $\mathbf{z}$  as a one-hot vector
- Gumbel-Max reparameterization trick** for sampling from categorical random variables

$$\mathbf{z} = \text{one\_hot} \left( \arg \max_i (g_i + \log \pi_i) \right)$$

where  $g_1, g_2, \dots, g_k$  are i.i.d. samples drawn from  $\text{Gumbel}(0, 1)$

- Reparametrizable since randomness is transferred to a fixed  $\text{Gumbel}(0, 1)$  distribution!
- Problem:  $\arg \max$  is non-differentiable w.r.t.  $\pi$

# Relaxing Categorical Distributions to Gumbel-Softmax

- Gumbel-Max Sampler (non-differentiable w.r.t.  $\pi$ ):

$$\mathbf{z} = \text{one\_hot} \left( \arg \max_i (g_i + \log \pi) \right)$$

- **Key idea:** Replace  $\arg \max$  with  $\text{soft max}$  to get a Gumbel-Softmax random variable  $\hat{\mathbf{z}}$
- Output of softmax is differentiable w.r.t.  $\pi$
- Gumbel-Softmax Sampler (differentiable w.r.t.  $\pi$ ):

$$\hat{\mathbf{z}} = \text{soft max}_i \left( \frac{g_i + \log \pi}{\tau} \right)$$

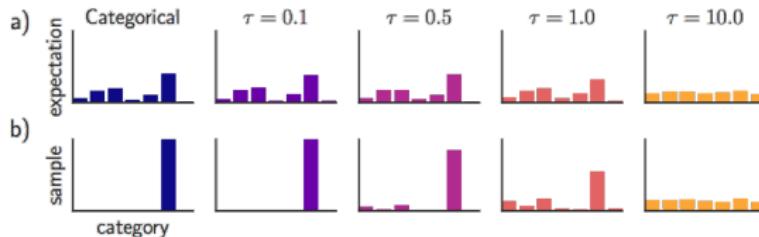
where  $\tau > 0$  is a tunable parameter referred to as the temperature

# Bias-variance tradeoff via temperature control

- Gumbel-Softmax distribution is parameterized by both class probabilities  $\pi$  and the temperature  $\tau > 0$

$$\hat{z} = \operatorname{soft\ max}_i \left( \frac{g_i + \log \pi}{\tau} \right)$$

- Temperature  $\tau$  controls the degree of the relaxation via a bias-variance tradeoff

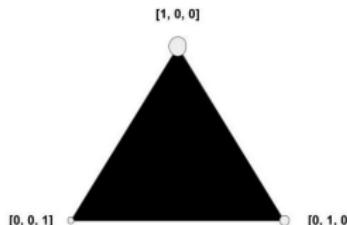


Source: Jang et al., 2017

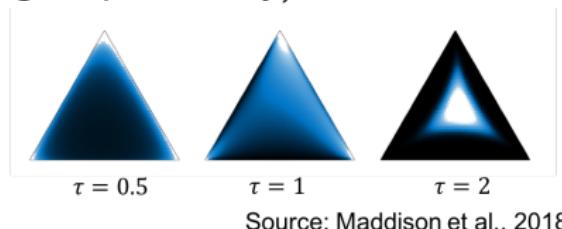
- As  $\tau \rightarrow 0$ , samples from  $\text{Gumbel-Softmax}(\pi, \tau)$  are similar to samples from  $\text{Categorical}(\pi)$   
**Pro:** low bias in approximation **Con:** High variance in gradients
- As  $\tau \rightarrow \infty$ , samples from  $\text{Gumbel-Softmax}(\pi, \tau)$  are similar to samples from  $\text{Categorical}\left(\left[\frac{1}{k}, \frac{1}{k}, \dots, \frac{1}{k}\right]\right)$  (i.e., uniform over  $k$  categories)

# Geometric Interpretation

- Consider a categorical distribution with class probability vector  $\pi = [0.60, 0.25, 0.15]$
- Define a probability simplex with the one-hot vectors as vertices



- For a categorical distribution, all probability mass is concentrated at the vertices of the probability simplex
- Gumbel-Softmax samples points within the simplex (lighter color intensity implies higher probability)



# Gumbel-Softmax in action

- Original optimization problem

$$\max_{\phi} E_{q_{\phi}(\mathbf{z})}[f(\mathbf{z})]$$

where  $q_{\phi}(\mathbf{z})$  is a categorical distribution and  $\phi = \pi$

- Relaxed optimization problem

$$\max_{\phi} E_{q_{\phi}(\hat{\mathbf{z}})}[f(\hat{\mathbf{z}})]$$

where  $q_{\phi}(\hat{\mathbf{z}})$  is a Gumbel-Softmax distribution and  $\phi = \{\pi, \tau\}$

- Usually, temperature  $\tau$  is explicitly annealed. Start high for low variance gradients and gradually reduce to tighten approximation
- Note that  $\hat{\mathbf{z}}$  is not a discrete category. If the function  $f(\cdot)$  explicitly requires a discrete  $\mathbf{z}$ , then we estimate **straight-through gradients**:
  - Use hard  $\mathbf{z} \sim \text{Categorical}(\mathbf{z})$  for evaluating objective in forward pass
  - Use soft  $\hat{\mathbf{z}} \sim \text{GumbelSoftmax}(\hat{\mathbf{z}}, \tau)$  for evaluating gradients in backward pass

## Combinatorial, Discrete Objects: Permutations

- For discovering rankings and matchings in an unsupervised manner,  $\mathbf{z}$  is represented as a permutation
- A  $k$ -dimensional permutation  $\mathbf{z}$  is a ranked list of  $k$  indices  $\{1, 2, \dots, k\}$
- Stochastic optimization problem

$$\max_{\phi} E_{q_{\phi}(\mathbf{z})}[f(\mathbf{z})]$$

where  $q_{\phi}(\mathbf{z})$  is a distribution over  $k$ -dimensional permutations

- First attempt: Each permutation can be viewed as a distinct category.  
Relax categorical distribution to Gumbel-Softmax
- Infeasible because number of possible  $k$ -dimensional permutations is  $k!$ . Gumbel-softmax does not scale for combinatorially large number of categories

# Plackett-Luce (PL) Distribution

- In many fields such as information retrieval and social choice theory, we often want to rank our preferences over  $k$  items. The **Plackett-Luce (PL) distribution** is a common modeling assumption for such rankings
- A  $k$ -dimensional PL distribution is defined over the set of permutations  $\mathcal{S}_k$  and parameterized by  $k$  positive scores  $\mathbf{s}$
- **Sequential sampler** for PL distribution
  - Sample  $z_1$  without replacement with probability proportional to the scores of all  $k$  items

$$p(z_1 = i) \propto s_i$$

- Repeat for  $z_2, z_3, \dots, z_k$

- **PDF** for PL distribution

$$q_{\mathbf{s}}(\mathbf{z}) = \frac{s_{z_1}}{Z} \frac{s_{z_2}}{Z - s_{z_1}} \frac{s_{z_3}}{Z - \sum_{i=1}^2 s_{z_i}} \cdots \frac{s_{z_k}}{Z - \sum_{i=1}^{k-1} s_{z_i}}$$

where  $Z = \sum_{i=1}^k s_i$  is the normalizing constant

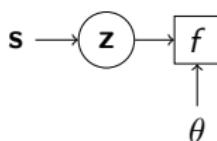
# Relaxing PL Distribution to Gumbel-PL

- **Gumbel-PL reparameterized sampler**

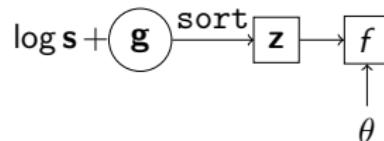
- Add i.i.d. standard Gumbel noise  $g_1, g_2, \dots, g_k$  to the log scores  $\log s_1, \log s_2, \dots, \log s_k$

$$\tilde{s}_i = g_i + \log s_i$$

- Set  $\mathbf{z}$  to be the permutation that sorts the Gumbel perturbed log-scores,  $\tilde{s}_1, \tilde{s}_2, \dots, \tilde{s}_k$



(a) Sequential Sampler



(b) Reparameterized Sampler

Figure: Squares and circles denote deterministic and stochastic nodes.

- **Challenge:** the sorting operation is non-differentiable in the inputs
- **Solution:** Use a differentiable relaxation. See the paper "Stochastic Optimization for Sorting Networks via Continuous Relaxations" for more details

# Summary

- Discovering discrete latent structure e.g., categories, rankings, matchings etc. has several applications
- Stochastic Optimization w.r.t. parameterized discrete distributions is challenging
- REINFORCE is the general purpose technique for gradient estimation, but suffers from high variance
- Control variates can help in controlling the variance
- Continuous relaxations to discrete distributions offer a biased, reparameterizable alternative with the trade-off in significantly lower variance