Discrete Latent Variable Models

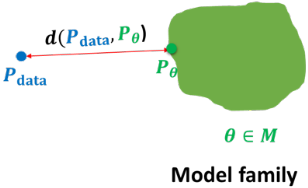
Stefano Ermon

Stanford University

Lecture 17

Summary

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



Major themes in the course

- Representing probability distributions
  - Probability density/mass functions: autoregressive models, flow models, variational autoencoders, energy-based models.
  - Sampling process: Generative adversarial networks.
  - Score function: Score-based generative models
- Distances between distributions: two sample test, maximum likelihood training, score matching, noise contrastive estimation.
- Evaluation of generative models

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

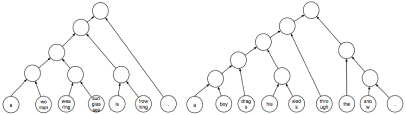


Figure 2: Examples of tree structures learned by our model which show that the model discovers simple concepts such as noun phrases and verb phrases.

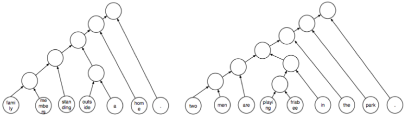
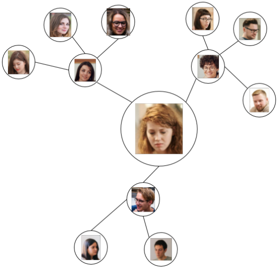


Figure 3: Examples of unconventional tree structures.

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

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

- Gradients w.r.t. θ can be derived via linearity of expectation

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

- If $\mathbf{z}$ is continuous, $q_\phi(\cdot)$ is reparameterizable, and $f(\cdot)$ is differentiable, then we can use reparameterization to compute gradients w.r.t. ϕ

Stochastic Optimization with Reparameterization

Consider the following optimization problem

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

Reparameterization trick:

- $\epsilon \sim p(\epsilon)$
- $\mathbf{z} = g_{\phi}(\epsilon) \sim q_{\phi}(\mathbf{z})$
- $E_{q_{\phi}(\mathbf{z})}[f(\mathbf{z})] = E_{\epsilon \sim p(\epsilon)}[f(g_{\phi}(\epsilon))]$
- Gradient ascent:

$$\begin{aligned} \nabla_{\phi} E_{q_{\phi}}(\mathbf{z})[f(\mathbf{z})] &= \nabla_{\phi} E_{\epsilon \sim p(\epsilon)}[f(g_{\phi}(\epsilon))] \\ &= E_{\epsilon \sim p(\epsilon)}[\nabla_{\phi} f(g_{\phi}(\epsilon))] \\ &= E_{\epsilon \sim p(\epsilon)}[\nabla_{\mathbf{z}} f(\mathbf{z}) \nabla_{\phi} g_{\phi}(\epsilon)] \end{aligned}$$

Assumptions: $f(\mathbf{z})$ is differentiable, and $q_{\phi}(\mathbf{z})$ is reparameterizable.

What if either of the above assumptions fails?

Stochastic Optimization with the log derivative trick

- Consider the following optimization problem

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

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

Multi-armed bandits

- 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 ϕ . For example, ϕ could be the parameters of a categorical distribution
- **Goal:** Learn the parameters ϕ that maximize our earnings (in expectation)

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

Log derivative trick for gradient estimation

- Want to compute a gradient with respect to ϕ of the expected reward

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

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

Log derivative trick for gradient estimation

- Want to compute a gradient with respect to ϕ of

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

- The log derivative trick gives

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

- We can now construct a Monte Carlo estimate
- Sample $\mathbf{z}^1, \cdots, \mathbf{z}^K$ 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 ϕ of

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

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

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

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

The log derivative trick has high variance

- Want to compute a gradient with respect to ϕ of

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

- The log derivative trick is

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

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

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

- Monte Carlo estimates of gradients are unbiased

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

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

Control Variates

- The log derivative trick gives

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

- Given any constant B (a control variate)

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

- To see why,

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

- 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

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

- $h(\mathbf{z})$ 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 $\widehat{f}(\mathbf{z})$ have the same expectation

$$E_{\mathbf{z}^1,\cdots,\mathbf{z}^K \sim q_\phi(\mathbf{z})}[\widehat{f}_{\mathrm{MC}}(\mathbf{z}^1,\cdots,\mathbf{z}^K)] = E_{\mathbf{z}^1,\cdots,\mathbf{z}^K \sim q_\phi(\mathbf{z})}[f_{\mathrm{MC}}(\mathbf{z}^1,\cdots,\mathbf{z}^K)]$$

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

Control variates

- Deriving an alternate Monte Carlo estimate for log derivative gradients based on control variates
- Sample $\mathbf{z}^1, \cdots, \mathbf{z}^K$ from $q_\phi(\mathbf{z})$

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

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

Control variates

- Comparing $\mathrm{Var}(\widehat{f_{\mathrm{MC}}})$ vs. $\mathrm{Var}(f_{\mathrm{MC}})$

$$
\begin{aligned}
\mathrm{Var}(\widehat{f_{\mathrm{MC}}}) &= \mathrm{Var}(f_{\mathrm{MC}} + a\left(h_{\mathrm{MC}} - E_{q_\phi(\mathbf{z})}[h(\mathbf{z})]\right)) \\
&= \mathrm{Var}(f_{\mathrm{MC}} + a h_{\mathrm{MC}}) \\
&= \mathrm{Var}(f_{\mathrm{MC}}) + a^2 \mathrm{Var}(h_{\mathrm{MC}}) + 2a\mathrm{Cov}(f_{\mathrm{MC}}, h_{\mathrm{MC}})
\end{aligned}
$$

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

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

Control variates

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

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

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

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

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

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

- 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 the log derivative trick + control variates
- **Control Variate 1:** Constant baseline B
- **Control Variate 2:** Input dependent baseline $h_\psi(\mathbf{x})$
- Gradient ascent w.r.t. ϕ with the log derivative trick + control variates

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

- Gradient ascent w.r.t. θ
- Optimize ψ, B to minimize $E_{q_\phi(\mathbf{z}|\mathbf{x})}[(f(\phi,\theta,\mathbf{z},\mathbf{x}) - h_\psi(\mathbf{x}) - B)^2]$

Towards reparameterized, continuous relaxations

- Consider the following optimization problem

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

- Reparameterization trick is not directly applicable for discrete $\mathbf{z}$
- The log derivative trick is a general-purpose solution, but needs careful design of control variates
- **Next:** Relax $\mathbf{z}$ to a continuous random variable with a reparameterizable distribution

Gumbel Distribution

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

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

- E.g., predicting maximum water level in a 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 μ and a scale parameter β

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

Categorical Distributions

- Let $\mathbf{z}$ denote a k-dimensional categorical random variable with distribution q parameterized by class probabilities $\boldsymbol{\pi} = \{\pi_1, \pi_2, \ldots, \pi_k\}$. 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 $g_1, g_2, \ldots, g_k$ are i.i.d. samples drawn from Gumbel$(0, 1)$
- In words, we can sample from Categorical$(\boldsymbol{\pi})$ by taking the $\arg\max$ over k Gumbel perturbed log-class probabilities $g_i + \log \pi_i$
- Reparametrizable since randomness is transferred to a fixed Gumbel$(0, 1)$ distribution!
- Problem: $\arg\max$ is non-differentiable w.r.t. $\boldsymbol{\pi}$

Relaxing Categorical Distributions to Gumbel-Softmax

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

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

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

$$\hat{\mathbf{z}} = \operatorname{soft}\max_i \left(\frac{g_i + \log \boldsymbol{\pi}}{\tau}\right)$$

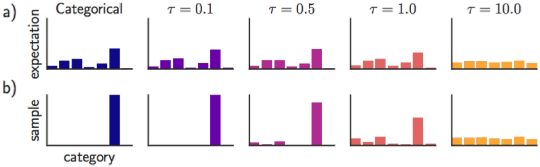
where $\tau > 0$ is a tunable parameter referred to as the temperature

Bias-variance tradeoff via temperature control

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

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

- Temperature τ controls the degree of the relaxation via a bias-variance tradeoff

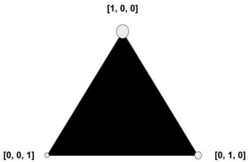


Source: Jang et al., 2017

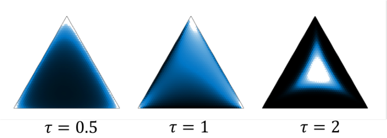
- As $\tau \to 0$, samples from Gumbel-Softmax$(\boldsymbol{\pi}, \tau)$ are similar to samples from Categorical$(\boldsymbol{\pi})$
  Pro: low bias in approximation **Con:** High variance in gradients
- As $\tau \to \infty$, samples from Gumbel-Softmax$(\boldsymbol{\pi}, \tau)$ are similar to samples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

Geometric Interpretation

- Consider a categorical distibution with class probability vector $\boldsymbol{\pi} = [0.60, 0.25, 0.15]$
- 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)



Source: Maddison et al., 2018

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 $\phi = \boldsymbol{\pi}$
- 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 $\phi = \{\boldsymbol{\pi}, \tau\}$
- Usually, temperature τ 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

Summary

- Discovering discrete latent structure e.g., categories, rankings, matchings etc. has several applications
- Stochastic Optimization w.r.t. parameterized discrete distributions is challenging
- The log derivative trick 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