Discrete Latent Variable Models

Stefano Ermon, Yang Song

Stanford University

Lecture 18
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
- 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
Consider the following optimization problem

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

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

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

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

$$\nabla_\theta E_{q_\phi(z|x)}[\log p_\theta(x, z) - \log q_\phi(z \mid x)] = E_{q_\phi(z|x)}[\nabla_\theta \log p_\theta(x, z)]$$

$$\approx \frac{1}{k} \sum_k \nabla_\theta \log p_\theta(x, z^k)$$

If $z$ is continuous, $q_\phi(\cdot)$ is reparameterizable, and $f(\cdot)$ is differentiable in $\phi$, then we can use reparameterization to compute gradients w.r.t. $\phi$
Consider the following optimization problem

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

Reparameterization trick:

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

\[
\nabla_\phi E_{q_\phi(z)}[f(z)] = \nabla_\phi E_{\epsilon \sim p(\epsilon)}[f(g_\phi(\epsilon))]
= E_{\epsilon \sim p(\epsilon)}[\nabla_\phi f(g_\phi(\epsilon))]
= E_{\epsilon \sim p(\epsilon)}[\nabla_z f(z) \nabla_\phi g_\phi(\epsilon)]
\]

What if any of the above assumptions fails?
Stochastic Optimization with the log derivative trick

Consider the following optimization problem

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

For many class of problem scenarios, reparameterization trick is infeasible

**Scenario 1:** $f(\cdot)$ is non-differentiable in $z$ 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

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 $z \in A$ has a reward $f(z)$
- Randomized policy for choosing actions $q_\phi(z)$ parameterized by $\phi$. For example, $\phi$ could be the parameters of a categorical distribution.
- **Goal:** Learn the parameters $\phi$ that maximize our earnings (in expectation)

$$\max_{\phi} E_{q_\phi(z)}[f(z)]$$
Log derivative trick for gradient estimation

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

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

$$\frac{\partial}{\partial \phi_i} E_{q_\phi(z)}[f(z)] = \sum_z \frac{\partial q_\phi(z)}{\partial \phi_i} f(z) = \sum_z q_\phi(z) \frac{1}{q_\phi(z)} \frac{\partial q_\phi(z)}{\partial \phi_i} f(z)$$

$$= \sum_z q_\phi(z) \frac{\partial \log q_\phi(z)}{\partial \phi_i} f(z) = E_{q_\phi(z)} \left[ \frac{\partial \log q_\phi(z)}{\partial \phi_i} f(z) \right]$$
Log derivative trick for gradient estimation

- Want to compute a gradient with respect to $\phi$ of
  \[ E_{q_\phi(z)}[f(z)] = \sum_z q_\phi(z)f(z) \]

- The log derivative trick gives
  \[ \nabla_\phi E_{q_\phi(z)}[f(z)] = E_{q_\phi(z)}[f(z)\nabla_\phi \log q_\phi(z)] \]

- We can now construct a Monte Carlo estimate
- Sample $z^1, \cdots, z^K$ from $q_\phi(z)$ and estimate
  \[ \nabla_\phi E_{q_\phi(z)}[f(z)] \approx \frac{1}{K} \sum_k f(z^k)\nabla_\phi \log q_\phi(z^k) \]

- 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

$$
\mathcal{L}(x; \theta, \phi) = \sum_z q_\phi(z|x) \log p_\theta(x, z) + H(q_\phi(z|x))
$$

$$
= E_{q_\phi(z|x)}[\log p_\theta(x, z) - \log q_\phi(z|x)]
$$

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

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

The log derivative trick yields

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

We can now construct a Monte Carlo estimate of $\nabla_\phi \mathcal{L}(x; \theta, \phi)$
The log derivative trick has high variance

- Want to compute a gradient with respect to $\phi$ of
  \[ E_{q\phi(z)}[f(z)] = \sum_z q\phi(z)f(z) \]

- The log derivative trick is
  \[ \nabla_\phi E_{q\phi(z)}[f(z)] = E_{q\phi(z)}[f(z)\nabla_\phi \log q\phi(z)] \]

- Monte Carlo estimate: Sample $z^1, \ldots, z^K$ from $q\phi(z)$
  \[ \nabla_\phi E_{q\phi(z)}[f(z)] \approx \frac{1}{K} \sum_k f(z^k)\nabla_\phi \log q\phi(z^k) := f_{MC}(z^1, \ldots, z^K) \]

- Monte Carlo estimates of gradients are unbiased
  \[ E_{z^1,\ldots,z^K\sim q\phi(z)}[f_{MC}(z^1, \ldots, z^K)] = \nabla_\phi E_{q\phi(z)}[f(z)] \]

- Almost never used in practice because of high variance
- Variance can be reduced via carefully designed control variates
The log derivative trick gives
\[
\nabla_\phi E_{q_\phi(z)}[f(z)] = E_{q_\phi(z)}[f(z)\nabla_\phi \log q_\phi(z)]
\]

Given any constant $B$ (a control variate)
\[
\nabla_\phi E_{q_\phi(z)}[f(z)] = E_{q_\phi(z)}[(f(z) - B)\nabla_\phi \log q_\phi(z)]
\]

To see why,
\[
E_{q_\phi(z)}[B\nabla_\phi \log q_\phi(z)] = B \sum_z q_\phi(z)\nabla_\phi \log q_\phi(z) = B \sum_z \nabla_\phi q_\phi(z)
\]
\[
= B\nabla_\phi \sum_z q_\phi(z) = B\nabla_\phi 1 = 0
\]

Monte Carlo gradient estimates of both $f(z)$ and $f(z) - B$ have same expectation

These estimates can however have different variances
Control variates

- Suppose we want to compute
  \[ E_{q_\phi(z)}[f(z)] = \sum_z q_\phi(z)f(z) \]

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

- \( h(z) \) is referred to as a control variate
- Assumption: \( E_{q_\phi(z)}[h(z)] \) is known
- Monte Carlo gradient estimates of \( f(z) \) and \( \hat{f}(z) \) have the same expectation
  \[ E_{z^1, \ldots, z^K \sim q_\phi(z)}[\hat{f}_{MC}(z^1, \ldots, z^K)] = E_{z^1, \ldots, z^K \sim q_\phi(z)}[f_{MC}(z^1, \ldots, z^K)] \]
  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 $z^1, \ldots, z^K$ from $q_\phi(z)$

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

\[
\approx \frac{1}{K} \sum_k f(z^k) \nabla_\phi \log q_\phi(z^k) + a \left( \frac{1}{K} \sum_{k=1}^K h(z^k) - E_{q_\phi(z)}[h(z)] \right)
\]

\[
:= f_{MC}(z^1, \ldots, z^K) + a \left( h_{MC}(z^1, \ldots, z^K) - E_{q_\phi(z)}[h(z)] \right)
\]

\[
:= \hat{f}_{MC}(z^1, \ldots, z^K)
\]

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

\[
\text{Var}(\hat{f}_{MC}) = \text{Var}(f_{MC} + a \left( h_{MC} - E_{q\phi(z)}[h(z)] \right)) \\
= \text{Var}(f_{MC} + ah_{MC}) \\
= \text{Var}(f_{MC}) + a^2 \text{Var}(h_{MC}) + 2a \text{Cov}(f_{MC}, h_{MC})
\]

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

\[
a^* = -\frac{\text{Cov}(f_{MC}, h_{MC})}{\text{Var}(h_{MC})}
\]
Control variates

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

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

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

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

- Correlation coefficient $\rho(f_{MC}, h_{MC})$ is between -1 and 1. For maximum variance reduction, we want $f_{MC}$ and $h_{MC}$ to be highly correlated.
Latent variable models with discrete latent variables are often referred to as belief networks.

Variational learning objective is same as ELBO

\[
\mathcal{L}(\mathbf{x}; \theta, \phi) = \sum_z q_\phi(\mathbf{z}|\mathbf{x}) \log p_\theta(\mathbf{x}, \mathbf{z}) + H(q_\phi(\mathbf{z}|\mathbf{x}))
\]

\[
= E_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}, \mathbf{z}) - \log q_\phi(\mathbf{z}|\mathbf{x})]
\]

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

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
- Learning objective

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

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

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

- Gradient ascent w.r.t. \( \theta, \psi, B \).
Towards reparameterized, continuous relaxations

Consider the following optimization problem

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

Reparameterization trick is not directly applicable for discrete \( z \)

The log derivative trick is a general-purpose solution, but needs careful design of control variates

**Next:** Relax \( z \) to a continuous random variable with a reparameterizable distribution
**Gumbel Distribution**

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

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

- 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 $\beta$.

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

- Let \( z \) denote a \( k \)-dimensional categorical random variable with distribution \( q \) parameterized by class probabilities \( \pi = \{\pi_1, \pi_2, \ldots, \pi_k\} \). We will represent \( z \) as a one-hot vector.

- **Gumbel-Max reparameterization trick** for sampling from categorical random variables

  \[
  z = \text{one-hot} \left( \arg \max_i (g_i + \log \pi_i) \right)
  \]

  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(\( \pi \)) 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. \( \pi \)
Relaxing Categorical Distributions to Gumbel-Softmax

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

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

- **Key idea:** Replace arg max with soft max to get a Gumbel-Softmax random variable $\hat{z}$

- Output of softmax is differentiable w.r.t. $\pi$

- **Gumbel-Softmax Sampler (differentiable w.r.t. $\pi$):**

  $$\hat{z} = \text{softmax} \left( i \left( \frac{g_i + \log \pi}{\tau} \right) \right)$$

  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 $\tau > 0$

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

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

- As $\tau \to 0$, 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 $\tau \to \infty$, samples from $\text{Gumbel-Softmax}(\pi, \tau)$ are similar to samples from $\text{Categorical} \left( \left[ \frac{1}{k}, \frac{1}{k}, \ldots, \frac{1}{k} \right] \right)$ (i.e., uniform over $k$ categories)

Source: Jang et al., 2017
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)

Source: Maddison et al., 2018
Gumbel-Softmax in action

- Original optimization problem

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

where \(q_\phi(z)\) is a categorical distribution and \(\phi = \pi\)

- Relaxed optimization problem

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

where \(q_\phi(\hat{z})\) 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 \(\hat{z}\) is not a discrete category. If the function \(f(\cdot)\) explicitly requires a discrete \(z\), then we estimate **straight-through gradients**:
  - Use hard \(z \sim \text{Categorical}(z)\) for evaluating objective in forward pass
  - Use soft \(\hat{z} \sim \text{GumbelSoftmax}(\hat{z}, \tau)\) for evaluating gradients in backward pass
For discovering rankings and matchings in an unsupervised manner, \( z \) is represented as a permutation

A \( k \)-dimensional permutation \( z \) is a ranked list of \( k \) indices \( \{1, 2, \ldots, k\} \)

Stochastic optimization problem

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

where \( q_\phi(z) \) 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 \( S_k \) and parameterized by \( k \) positive scores:
  \[
  s := (s_1, s_2, \ldots, s_k)
  \]

- **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 \( z_2, z_3, \ldots, z_k \)

- **PDF** for PL distribution:
  \[
  q_s(z) = \frac{s_{z_1}}{Z} \cdot \frac{s_{z_2}}{Z - s_{z_1}} \cdot \frac{s_{z_3}}{Z - \sum_{i=1}^{2} s_{z_i}} \cdots \frac{s_{z_k}}{Z - \sum_{i=1}^{k-1} s_{z_i}}
  \]

  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 $g_1, g_2, \ldots, g_k$ to the log scores $\log s_1, \log s_2, \ldots, \log s_k$
  - $\tilde{s}_i = g_i + \log s_i$

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

\[ s \xrightarrow{z} f \]
\[ \log s \xrightarrow{g_{\text{sort}}} z \xrightarrow{f} \]

(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" (Grovel et al. 2019) for more details.
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