Advanced Approximate Inference for Discrete LVMs

Deep Learning 2 – 2022

Wilker Aziz
w.aziz@uva.nl

UNIVERSITY OF AMSTERDAM
Institute for Logic, Language and Computation

Outline

Gradients of the evidence lowerbound

ELBO

$$\mathcal{E}(\lambda, \theta) = \mathbb{E}_{q_{Z|X}(z|x,\lambda)}\left[\log \frac{p_{ZX}(z, x|\theta)}{q_{Z|X}(z|x, \lambda)}\right]$$

Updating generative model

$$\nabla_\theta \mathcal{E}(\lambda, \theta) = \mathbb{E}_{q_{Z|X}(z|x,\lambda)}\left[\nabla_\theta \log \frac{p_{ZX}(z, x|\theta)}{q_{Z|X}(z|x, \lambda)}\right]$$

Updating inference model

$$\nabla_\lambda \mathcal{E}(\lambda, \theta) = \nabla_\lambda \mathbb{E}_{f_Z(z|\lambda)}[\psi(z)]$$

$f_Z(z|\lambda) = q_{Z|X}(z|x, \lambda)$ and $\psi(z) = \log \frac{p_{ZX}(z,x|\theta)}{q_{Z|X}(z|x,\lambda)}$

Score function estimator

$$\frac{\partial}{\partial\lambda}\mathbb{E}_{f_{Z|\lambda}(z)}\left[\psi(z)\right] = \mathbb{E}_{f_{Z|\lambda}(z)}\left[\psi(z)\frac{\partial}{\partial\lambda}\log f_{Z|\lambda}(z)\right]$$

Easy to MC estimate, but noisy.

Reparameterised gradient

$$\begin{aligned}\frac{\partial}{\partial \lambda}\mathbb{E}_{f_{Z|\lambda}(z)}\left[\psi(z)\right] &= \mathbb{E}_{s(\epsilon)}\left[\frac{\partial}{\partial \lambda}\psi(t(\epsilon,\lambda))\right] \\ &= \mathbb{E}_{s(\epsilon)}\left[\frac{\partial}{\partial z}\psi(z)\frac{\partial}{\partial \lambda}t(\epsilon,\lambda)\right]\end{aligned}$$

Easy to MC estimate.

Reparameterisation

- $t(\epsilon, \lambda)$ is invertible and differentiable
- $z = t(\epsilon, \lambda)$ has density $f_Z(z|\lambda)$
- $\epsilon = t^{-1}(z, \lambda)$ has density $s(\epsilon)$

Change of density

$$f_{Z|\lambda}(z) = s(\underbrace{t^{-1}(z,\lambda)}_{\epsilon})|\det J_{t^{-1}}(z,\lambda)|$$

As a result

$$\begin{aligned}\mathbb{E}_{f_Z(z|\lambda)}[\psi(z)] &= \int f_Z(z|\lambda)\psi(z)\mathrm{d}z \\ &= \int s(t^{-1}(z,\lambda))|\det J_{t^{-1}}(z,\lambda)|\psi(z)\mathrm{d}z \\ &= \int s(\epsilon)|\det J_t(\epsilon,\lambda)|^{-1}\psi(t(\epsilon,\lambda))|\det J_t(\epsilon,\lambda)|\mathrm{d}\epsilon \\ &= \int s(\epsilon)\psi(t(\epsilon,\lambda))\mathrm{d}\epsilon\end{aligned}$$

Gradient Estimators

Basic problem: we want to differentiate an expected value wrt λ

$$\frac{\partial}{\partial\lambda}\mathbb{E}_{f_{Z|\lambda}}[\psi(z)] \qquad \text{e.g., } \psi(z) := \log p(x|z,\theta)$$

$$f_{Z|\lambda}(z) := q(z|x,\lambda)$$

but the distribution of Z depends on λ.

We have met the SFE and the reparameterised gradient estimator:

$$\mathbb{E}_{f_\lambda(z)}\left[\underbrace{\color{red}{\psi(z)}\frac{\partial}{\partial\lambda}\log f_{Z|\lambda}(z)}_{\hat{g}_{\text{sfe}}}\right] = \mathbb{E}_{s(\epsilon)}\left[\underbrace{\color{blue}{\frac{\partial}{\partial z}\psi(z)}\frac{\partial}{\partial\lambda}t(\epsilon,\lambda)}_{\hat{g}_{\text{rep}}}\right]$$

- $\hat{g}_{\text{sfe}}$ is typically cursed with variance

From VAEs, you know the *reparameterised gradient estimator*

$$\begin{aligned}\frac{\partial}{\partial\lambda}\mathbb{E}_{f_{Z|\lambda}(z)}[\psi(z)] &= \mathbb{E}_{s(\epsilon)}\left[\frac{\partial}{\partial\lambda}\psi(t(\epsilon,\lambda))\right] \\ &= \mathbb{E}_{s(\epsilon)}\left[\frac{\partial}{\partial z}\psi(z)\frac{\partial}{\partial\lambda}t(\epsilon,\lambda)\right]\end{aligned}$$

it takes an invertible and differentiable transformation t such that

$$t(\epsilon,\lambda) \sim f_{Z|\lambda}$$
$$t^{-1}(z,\lambda) \sim s(\epsilon)$$

Goals Understand why there can't be a $\hat{g}_{\text{rep}}$ for discrete rvs. Meet alternatives to SFE.

A general reparameterisation

For univariate Z, what transformation will always absorb the parameters of the density $f_{Z|\lambda}(z)$?

The argument extends to a vector of independent univariate rvs.

A general reparameterisation

For univariate Z, what transformation will always absorb the parameters of the density $f_{Z|\lambda}(z)$?

$$\underbrace{F_{Z|\lambda}(z)}_{\text{cdf}} \sim \mathcal{U}(\underbrace{0, 1}_{\text{fixed}})$$

The argument extends to a vector of independent univariate rvs.

A general reparameterisation

For univariate Z, what transformation will always absorb the parameters of the density $f_{Z|\lambda}(z)$?

$$\underbrace{F_{Z|\lambda}(z)}_{\text{cdf}} \sim \mathcal{U}(\underbrace{0, 1}_{\text{fixed}})$$

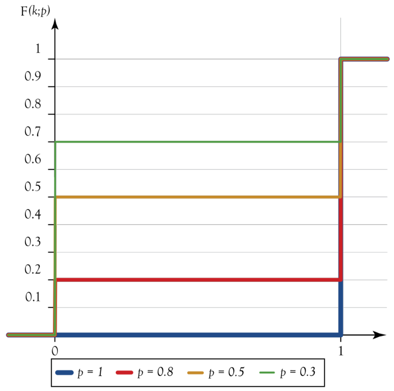
So, if I know the inverse cdf,

$$\epsilon \sim \mathcal{U}(0, 1)$$
$$F^{-1}_{Z|\lambda}(\epsilon) \sim Z|\lambda$$

I have access to $\hat{g}_{\text{rep}}$.

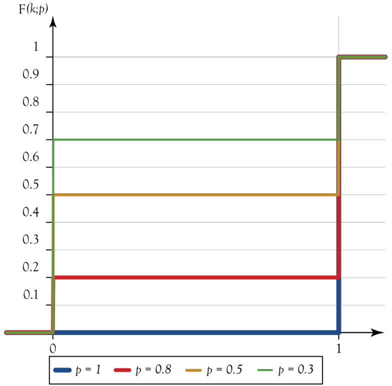
The argument extends to a vector of independent univariate rvs.

Let's reparameterise a Bernoulli



$$Z \sim \text{Bernoulli}(p)$$

Let's reparameterise a Bernoulli

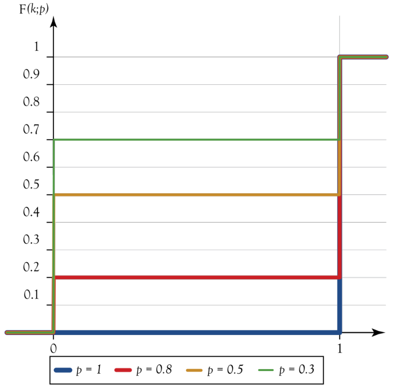


$$Z \sim \text{Bernoulli}(p)$$

$$F^{-1}_{Z|p}(\epsilon) = \begin{cases} 1 & \text{if } \epsilon < p \\ 0 & \text{otherwise} \end{cases}$$

$$= \mathbb{1}_{(0,p)}(\epsilon)$$

Let's reparameterise a Bernoulli

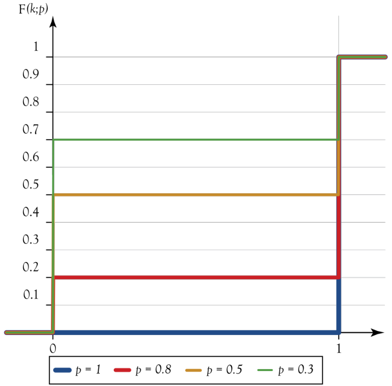


$$Z \sim \text{Bernoulli}(p)$$

$$F^{-1}_{Z|p}(\epsilon) = \begin{cases} 1 & \text{if } \epsilon < p \\ 0 & \text{otherwise} \end{cases}$$
$$= \mathbb{1}_{(0,p)}(\epsilon)$$

How about $\frac{\partial}{\partial p} F^{-1}_{Z|p}(\epsilon)$?

Let's reparameterise a Bernoulli



$$Z \sim \text{Bernoulli}(p)$$

$$F^{-1}_{Z|p}(\epsilon) = \begin{cases} 1 & \text{if } \epsilon < p \\ 0 & \text{otherwise} \end{cases}$$

$$= \mathbb{1}_{(0,p)}(\epsilon)$$

How about $\frac{\partial}{\partial p} F^{-1}_{Z|p}(\epsilon)$? Mostly 0, sometimes undefined!

Discrete case

Discrete variables do not admit a differentiable reparameterisation. The derivatives of the inverse cdf are either 0 or undefined :/

STE's original paper (Bengio et al., 2013).

There are other pseudo-gradients in the literature, for example for relaxed combinatorial random variables (Peng et al., 2018; Mihaylova et al., 2021).

Discrete case

Discrete variables do not admit a differentiable reparameterisation. The derivatives of the inverse cdf are either 0 or undefined :/

The score function estimator is fully general, but very noisy.

STE's original paper (Bengio et al., 2013).

There are other pseudo-gradients in the literature, for example for relaxed combinatorial random variables (Peng et al., 2018; Mihaylova et al., 2021).

Discrete case

Discrete variables do not admit a differentiable reparameterisation. The derivatives of the inverse cdf are either 0 or undefined :/

The score function estimator is fully general, but very noisy.

How about we fake a Jacobian and call it a pseudo-gradient?

$$J_t(\epsilon, \lambda) = \text{diag}(\mathbf{1})$$

This is the ingredient behind the straight-through estimator (STE).

STE's original paper (Bengio et al., 2013).

There are other pseudo-gradients in the literature, for example for relaxed combinatorial random variables (Peng et al., 2018; Mihaylova et al., 2021).

Bernoulli-STE

Consider a VAE where $q(z|x) = \text{Bern}(z|g(x;\lambda))$.

Bernoulli-STE

Consider a VAE where $q(z|x) = \text{Bern}(z|g(x;\lambda))$.
We sample z via a reparameterisation that absorbs λ:

$$\epsilon \sim \mathcal{U}(0,1) \qquad p = g(x;\lambda) \qquad z = \underbrace{\mathbb{1}_{(0,p)}(\epsilon)}_{t(\epsilon,\lambda)}$$

Bernoulli-STE

Consider a VAE where $q(z|x) = \text{Bern}(z|g(x; \lambda))$.
We sample z via a reparameterisation that absorbs λ:

$$\epsilon \sim \mathcal{U}(0, 1) \qquad p = g(x; \lambda) \qquad z = \underbrace{\mathbb{1}_{(0,p)}(\epsilon)}_{t(\epsilon, \lambda)}$$

Optimising the ELBO via reparameterised samples requires

$$\hat{g}_{\text{rep}} = \frac{\partial}{\partial \lambda} \log p(x|z = t(\epsilon, \lambda)) = \frac{\partial}{\partial z} \log p(x|z) \frac{\partial}{\partial \lambda} t(\epsilon, \lambda)$$

Bernoulli-STE

Consider a VAE where $q(z|x) = \text{Bern}(z|g(x;\lambda))$.
We sample z via a reparameterisation that absorbs λ:

$$\epsilon \sim \mathcal{U}(0,1) \qquad p = g(x;\lambda) \qquad z = \underbrace{\mathbb{1}_{(0,p)}(\epsilon)}_{t(\epsilon,\lambda)}$$

Optimising the ELBO via reparameterised samples requires

$$\hat{g}_{\text{rep}} = \frac{\partial}{\partial\lambda} \log p(x|z = t(\epsilon,\lambda)) = \frac{\partial}{\partial z} \log p(x|z) \frac{\partial}{\partial\lambda} t(\epsilon,\lambda)$$

Let's use our *pseudo gradient*

$$\hat{g}_{\text{ste}} := \frac{\partial}{\partial\lambda} t(\epsilon,\lambda) = \frac{\partial}{\partial\lambda} g(x;\lambda) \cancel{\frac{\partial}{\partial p} \mathbb{1}_{(0,p)}(\epsilon)} := 1$$

Concrete Distribution

We can sample from a Categorical distribution via

$$\epsilon_k \sim \text{Gumbel}(0, 1)$$
$$\underbrace{\arg\max_k \; \{\lambda_k + \epsilon_k\}_{k=1}^K}_{z = t(\epsilon, \lambda)} \sim \text{Cat}(\text{softmax}(\lambda))$$

Concrete distribution (Maddison et al., 2017), Gumbel-Softmax distribution (Jang et al., 2017).

Concrete Distribution

We can sample from a Categorical distribution via

$$\epsilon_k \sim \text{Gumbel}(0, 1)$$
$$\underbrace{\arg\max_k \; \{\lambda_k + \epsilon_k\}_{k=1}^K}_{z = t(\epsilon, \lambda)} \sim \text{Cat}(\text{softmax}(\lambda))$$

The problem is that $t(\epsilon, \lambda)$ is not differentiable, but note

$$\text{softmax}\left(\frac{\lambda + \epsilon}{\tau}\right) \to \text{onehot}(z) \qquad \text{as } \tau \to 0$$

and now the transformation is differentiable, but the outcome is dense.

Concrete distribution (Maddison et al., 2017), Gumbel-Softmax distribution (Jang et al., 2017).

Concrete Distribution

We can sample from a Categorical distribution via

$$\epsilon_k \sim \text{Gumbel}(0, 1)$$
$$\underbrace{\arg\max_k \{\lambda_k + \epsilon_k\}_{k=1}^K}_{z=t(\epsilon,\lambda)} \sim \text{Cat}(\text{softmax}(\lambda))$$

The problem is that $t(\epsilon, \lambda)$ is not differentiable, but note

$$\text{softmax}\left(\frac{\lambda + \epsilon}{\tau}\right) \to \text{onehot}(z) \qquad \text{as } \tau \to 0$$

and now the transformation is differentiable, but the outcome is dense.

Concrete distribution (Maddison et al., 2017), Gumbel-Softmax distribution (Jang et al., 2017).

STE: use sparse encoding for z

$$z = \text{onehot}\left(\frac{\lambda + \epsilon}{\tau}\right)$$

but use the Jacobian of dense encoding

$$z = \text{softmax}\left(\frac{\lambda + \epsilon}{\tau}\right)$$

Concrete Distribution

We can sample from a Categorical distribution via

$$\begin{aligned} \epsilon_k &\sim \text{Gumbel}(0,1) \\ \underbrace{\arg\max_k \ \{\lambda_k + \epsilon_k\}_{k=1}^K}_{z=t(\epsilon,\lambda)} &\sim \text{Cat}(\text{softmax}(\lambda)) \end{aligned}$$

The problem is that $t(\epsilon, \lambda)$ is not differentiable, but note

$$\text{softmax}\left(\frac{\lambda + \epsilon}{\tau}\right) \to \text{onehot}(z) \qquad \text{as } \tau \to 0$$

and now the transformation is differentiable, but the outcome is dense.

Concrete distribution (Maddison et al., 2017), Gumbel-Softmax distribution (Jang et al., 2017).

STE: use sparse encoding for z

$$z = \text{onehot}\left(\frac{\lambda + \epsilon}{\tau}\right)$$

but use the Jacobian of dense encoding

$$z = \text{softmax}\left(\frac{\lambda + \epsilon}{\tau}\right)$$

Mixed Binary Variables

Continuous random variables that take on sparse outcomes with non-zero probability mass.
Example I:

- sample $\zeta \sim \mathcal{N}(0, 1)$
- rectify the sample via hardsigmoid $z = \min(1, \max(0, z))$.

What is the probability $\Pr(Z \in \{0\})$?

Mixed Binary Variables

Continuous random variables that take on sparse outcomes with non-zero probability mass.
Example I:

- sample $\zeta \sim \mathcal{N}(0, 1)$
- rectify the sample via hardsigmoid $z = \min(1, \max(0, z))$.

What is the probability $\Pr(Z \in \{0\})$? it is $\Phi(0) = \int_{\mathbb{R}_{<0}} \mathcal{N}(0, 1)\mathrm{d}z$

When $\zeta < 0$ or $\zeta > 1$ the derivative of hardsigmoid is 0, when $0 < \zeta < 1$ the derivative is 1. Hardsigmoid has undefined derivatives for $\zeta = 0$ and $\zeta = 1$, but we will never sample those.

If we had a parameterised Gaussian, we could sample with a reparameterisation and learn the Gaussian parameters.

This has been applied to generate mixed random variables in the support $[0, 1]$.

Spike-and-slab (Rolfe, 2017); HardConcrete (Louizos et al., 2018); Hard-Kumaraswamy (Bastings et al., 2019).

Applications to interpretability (Voita et al., 2019; Cao et al., 2020; Ataman et al., 2020)

Mixed Binary Variables

Continuous random variables that take on sparse outcomes with non-zero probability mass.
Example I:

- sample $\zeta \sim \mathcal{N}(0, 1)$
- rectify the sample via hardsigmoid $z = \min(1, \max(0, z))$.

What is the probability $\Pr(Z \in \{0\})$? it is $\Phi(0) = \int_{\mathbb{R}_{<0}} \mathcal{N}(0, 1)\mathrm{d}z$

When $\zeta < 0$ or $\zeta > 1$ the derivative of hardsigmoid is 0, when $0 < \zeta < 1$ the derivative is 1. Hardsigmoid has undefined derivatives for $\zeta = 0$ and $\zeta = 1$, but we will never sample those.

If we had a parameterised Gaussian, we could sample with a reparameterisation and learn the Gaussian parameters.

This has been applied to generate mixed random variables in the support $[0, 1]$.

Spike-and-slab (Rolfe, 2017); HardConcrete (Louizos et al., 2018); Hard-Kumaraswamy (Bastings et al., 2019).

Applications to interpretability (Voita et al., 2019; Cao et al., 2020; Ataman et al., 2020)

Mixed Binary Variables

Continuous random variables that take on sparse outcomes with non-zero probability mass.

Construction:

- start with a continuous univariate rv ζ with density $s(\zeta)$
- project it to $[0, 1]$ using a function that hits the boundaries of the set
- the projection function is differentiable everywhere except at $\zeta = 0$ and $\zeta = 1$ which have 0 measure under $s(\zeta)$

The path derivative is defined *almost everywhere*, a g-rep can be used.

Can we go beyond univariates?

Yes, if we look into *sparse projections to the probability simplex*.

It turns out the 'hard sigmoid' is the a special case of a more general projection known as *sparsemax*.

Sparsemax

$$\mathrm{sparsemax}(\zeta) = \arg \max_{\zeta \in \Delta_{K-1}} \min ||p - \zeta||^2$$

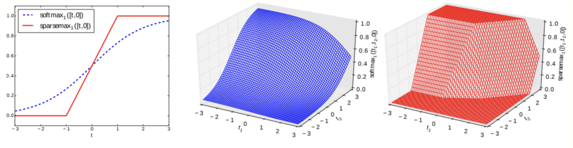


Figure 1. Comparison of softmax and sparsemax in 2D (left) and 3D (two righmost plots).

When $K = 2$, sparsemax is equivalent to hardsigmoid.

Mixed Random Variables

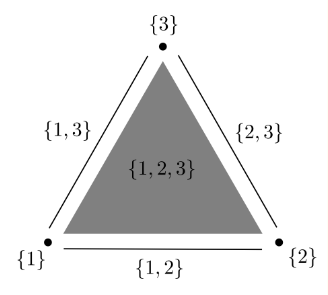
Generalisation of mixed binary random variables to the multivariate case.

Intrinsic view:

- draw ζ in $\mathbb{R}^K$ (e.g., from a multivariate Gaussian)
- project it to Δ_{K-1} (e.g., using sparsemax)

What is the probability that we have a point in one of the faces of the simplex?

The faces of the simplex (e.g., with 3 vertices):



Mixed Random Variables

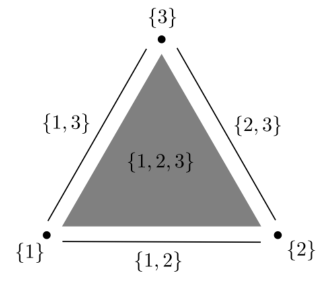
Generalisation of mixed binary random variables to the multivariate case.

Intrinsic view:

- draw ζ in $\mathbb{R}^K$ (e.g., from a multivariate Gaussian)
- project it to Δ_{K-1} (e.g., using sparsemax)

What is the probability that we have a point in one of the faces of the simplex? In this example, we would have to integrate the multivariate Gaussian pdf over the set of points in the inverse of sparsemap.

The faces of the simplex (e.g., with 3 vertices):



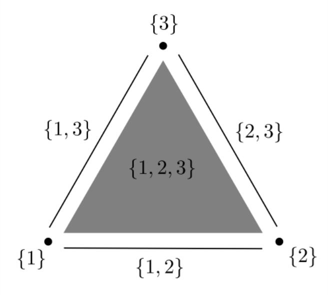
Mixed Random Variables

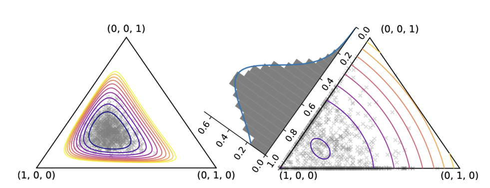
Generalisation of mixed binary random variables to the multivariate case.

Extrinsic view:

- choose a face f of the simplex with probability $P_F(f)$
- draw z from the relative interior of f (e.g., using a LogisticNormal distribution)

The faces of the simplex (e.g., with 3 vertices):





Farinhas et al. (2022)

Outline

Latent Computation Graphs

Estimators built on reparameterisation require

- z to be of some fixed finite dimensionality
- the decoder's computation graph must be independent of z.

Some composition functions are parameterised by their inputs (e.g., a tree-LSTM), they are dynamic computation graphs controlled by the discrete latent.

STE is not an option, so we are back to SFE. Or are we?

$\hat{g}_{\text{rep}}$ differentiates the decoder wrt z

$$\hat{g}_{\text{rep}} = \frac{\partial}{\partial z}\psi(z) \times \frac{\partial}{\partial \lambda} t(\epsilon, \lambda)$$

This cannot work when the computation graph of ψ depends on z (i.e., whenever z cannot be treated as a point in the relative interior of a fixed and finite-dimensional polytope). For example, a tree-LSTM updates its states following a depth-first traversal of an input tree.

Parameterise for Tractability

We can use sparse projections to the probability simplex to parameterise discrete distributions that assign 0 probability mass to most of the outcomes in their supports.

$$
\begin{aligned}
&\nabla_\lambda \mathbb{E}_{q(z|x,\lambda)}[\log p(x,z|\theta) - \log q(z|x,\lambda)] \\
=&\nabla_\lambda q(c_1|x,\lambda)(\log p(x,c_1|\theta) - \log q(c_1|x,\lambda)) \\
&+ \dots \\
&+ \nabla_\lambda q(c_k|x,\lambda)(\log p(x,c_k|\theta) - \log q(c_k|x,\lambda)) \\
&+ \nabla_\lambda \sum_{z=c_{k+1}}^{c_K} \underbrace{q(z|x,\lambda)}_{=0}(\log p(x,z|\theta) - \log q(z|x,\lambda))
\end{aligned}
$$

Effectively, we have an inference network that parameterises a model whose support is small enough for *enumeration*.

Sparse projections: (Martins and Astudillo, 2016; Niculae et al., 2018a)

Latent dynamic computation graphs: (Niculae et al., 2018b)

Sparse marginals: (Correia et al., 2020)

Outline

1. Reparameterised Gradients
2. Sparse Gradients
3. Variance reduction

Control variates

Intuition

To estimate $\mathbb{E}[\psi(z)]$ via Monte Carlo we compute the empirical average of $\hat{\psi}(z)$ where $\hat{\psi}(z)$ is chosen so that $\mathbb{E}[\hat{\psi}(z)] = \mathbb{E}[\psi(z)]$ and $\text{Var}(\psi) > \text{Var}(\hat{\psi})$.

Equivalent expectations

Let $\bar{\psi} = \mathbb{E}[\psi(z)]$ be an expectation of interest

Equivalent expectations

Let $\bar{\psi} = \mathbb{E}[\psi(z)]$ be an expectation of interest

- say we know $\bar{c} = \mathbb{E}[c(z)]$

Equivalent expectations

Let $\bar{\psi} = \mathbb{E}[\psi(z)]$ be an expectation of interest

- say we know $\bar{c} = \mathbb{E}[c(z)]$
- then for $\hat{\psi}(z) \triangleq \psi(z) - b(c(z) - \mathbb{E}[c(z)])$

Equivalent expectations

Let $\bar{\psi} = \mathbb{E}[\psi(z)]$ be an expectation of interest

- say we know $\bar{c} = \mathbb{E}[c(z)]$
- then for $\hat{\psi}(z) \triangleq \psi(z) - b(c(z) - \mathbb{E}[c(z)])$
  it holds that $\mathbb{E}[\hat{\psi}(z)] = \mathbb{E}[\psi(z)]$

Equivalent expectations

Let $\bar{\psi} = \mathbb{E}[\psi(z)]$ be an expectation of interest

- say we know $\bar{c} = \mathbb{E}[c(z)]$
- then for $\hat{\psi}(z) \triangleq \psi(z) - b(c(z) - \mathbb{E}[c(z)])$
  it holds that $\mathbb{E}[\hat{\psi}(z)] = \mathbb{E}[\psi(z)]$
- and $\mathrm{Var}(\hat{\psi}) = \mathrm{Var}(\psi) - 2b\,\mathrm{Cov}(\psi, c) + b^2\,\mathrm{Var}(c)$

Choosing the control variate

1. $\hat{\psi}(z) \triangleq \psi(z) - b(c(z) - \mathbb{E}[c(z)])$
2. $\mathsf{Var}(\hat{\psi}) = \mathsf{Var}(\psi) - 2b\,\mathsf{Cov}(\psi, c) + b^2\,\mathsf{Var}(c)$

How do we choose b and $c(z)$?

Choosing the control variate

1. $\hat{\psi}(z) \triangleq \psi(z) - b(c(z) - \mathbb{E}[c(z)])$
2. $\text{Var}(\hat{\psi}) = \text{Var}(\psi) - 2b\,\text{Cov}(\psi, c) + b^2\,\text{Var}(c)$

How do we choose b and $c(z)$?

- solving $\frac{\partial}{\partial b}\text{Var}(\hat{\psi}) = 0$ yields $b^\star = {}^{\text{Cov}(\psi, c)}/_{\text{Var}(c)}$
  when $\psi(z)$ and $c(z)$ are positively correlated, then we may reduce variance

Choosing the control variate

1. $\hat{\psi}(z) \triangleq \psi(z) - b(c(z) - \mathbb{E}[c(z)])$
2. $\text{Var}(\hat{\psi}) = \text{Var}(\psi) - 2b\,\text{Cov}(\psi, c) + b^2\,\text{Var}(c)$

How do we choose b and $c(z)$?

- solving $\frac{\partial}{\partial b}\text{Var}(\hat{\psi}) = 0$ yields $b^\star = {}^{\text{Cov}(\psi, c)}/_{\text{Var}(c)}$
  when $\psi(z)$ and $c(z)$ are positively correlated, then we may reduce variance

Of course, $\mathbb{E}[c(z)]$ must be known!

MC

We then use the estimate

$$\bar{\psi} \overset{\text{MC}}{\approx} \frac{1}{S}\left(\sum_{s=1}^{S} \psi(z^{(s)}) - bc(z^{(s)})\right) + b\bar{c}$$

MC

We then use the estimate

$$\bar{\psi} \overset{\text{MC}}{\approx} \frac{1}{S}\left(\sum_{s=1}^{S} \psi(z^{(s)}) - bc(z^{(s)})\right) + b\bar{c}$$

And recall that for us

$$\psi(z) = \log \frac{p_{ZX}(z, x|\theta)}{q_{Z|X}(z|x, \lambda)} \frac{\partial}{\partial \lambda} \log q_{Z|X}(z|x, \lambda)$$

and $z^{(s)} \sim q_{Z|X}(z|x, \lambda)$

Expected score

The Expectation of the score function is 0.

$$\mathbb{E}_{q(z|x,\lambda)}\left[\frac{\partial}{\partial\lambda}\log q(z|x,\lambda)\right]$$

Expected score

The Expectation of the score function is 0.

$$\begin{aligned}
&\mathbb{E}_{q(z|x,\lambda)}\left[\frac{\partial}{\partial\lambda}\log q(z|x,\lambda)\right]\\
&=\int q(z|x,\lambda)\frac{\partial}{\partial\lambda}\log q(z|x,\lambda)\mathrm{d}z
\end{aligned}$$

Expected score

The Expectation of the score function is 0.

$$
\begin{aligned}
&\mathbb{E}_{q(z|x,\lambda)}\left[\frac{\partial}{\partial\lambda}\log q(z|x,\lambda)\right] \\
&= \int q(z|x,\lambda)\frac{\partial}{\partial\lambda}\log q(z|x,\lambda)\mathrm{d}z \\
&= \int \frac{\partial}{\partial\lambda} q(z|x,\lambda)\mathrm{d}z
\end{aligned}
$$

Expected score

The Expectation of the score function is 0.

$$
\begin{aligned}
&\mathbb{E}_{q(z|x,\lambda)}\left[\frac{\partial}{\partial\lambda}\log q(z|x,\lambda)\right] \\
&= \int q(z|x,\lambda)\frac{\partial}{\partial\lambda}\log q(z|x,\lambda)\mathrm{d}z \\
&= \int \frac{\partial}{\partial\lambda}q(z|x,\lambda)\mathrm{d}z \\
&= \frac{\partial}{\partial\lambda}\int q(z|x,\lambda)\mathrm{d}z
\end{aligned}
$$

Expected score

The Expectation of the score function is 0.

$$
\begin{aligned}
&\mathbb{E}_{q(z|x,\lambda)}\left[\frac{\partial}{\partial\lambda}\log q(z|x,\lambda)\right] \\
&= \int q(z|x,\lambda)\frac{\partial}{\partial\lambda}\log q(z|x,\lambda)\mathrm{d}z \\
&= \int \frac{\partial}{\partial\lambda}q(z|x,\lambda)\mathrm{d}z \\
&= \frac{\partial}{\partial\lambda}\int q(z|x,\lambda)\mathrm{d}z \\
&= \frac{\partial}{\partial\lambda}1 = {\color{blue}0}
\end{aligned}
$$

Baselines

With

$$\psi(z) = \log \frac{p(z, x|\theta)}{q(z|x, \lambda)} \frac{\partial}{\partial \lambda} \log q(z|x, \lambda)$$

and

$$c(z) = \frac{\partial}{\partial \lambda} \log q(z|x, \lambda)$$

we have

$$\hat{\psi}(z) =$$

Baselines

With

$$\psi(z) = \log \frac{p(z, x|\theta)}{q(z|x, \lambda)} \frac{\partial}{\partial \lambda} \log q(z|x, \lambda)$$

and

$$c(z) = \frac{\partial}{\partial \lambda} \log q(z|x, \lambda)$$

we have

$$\hat{\psi}(z) = \left(\log \frac{p(z, x|\theta)}{q(z|x, \lambda)} - b\right) \frac{\partial}{\partial \lambda} \log q(z|x, \lambda)$$

Baselines

With

$$\psi(z) = \log \frac{p(z, x|\theta)}{q(z|x, \lambda)} \frac{\partial}{\partial \lambda} \log q(z|x, \lambda)$$

and

$$c(z) = \frac{\partial}{\partial \lambda} \log q(z|x, \lambda)$$

we have

$$\hat{\psi}(z) = \left(\log \frac{p(z, x|\theta)}{q(z|x, \lambda)} - b \right) \frac{\partial}{\partial \lambda} \log q(z|x, \lambda)$$

b is known as *baseline* in RL literature.

Examples of baselines

- Moving average of $\log \frac{p(z,x|\theta)}{q(z|x,\lambda)}$ based on previous batches

Examples of baselines

- Moving average of $\log \frac{p(z,x|\theta)}{q(z|x,\lambda)}$ based on previous batches
- A trainable constant b

Examples of baselines

- Moving average of $\log \frac{p(z,x|\theta)}{q(z|x,\lambda)}$ based on previous batches
- A trainable constant b
- A neural network prediction based on x e.g. $b(x;\omega)$

Examples of baselines

- Moving average of $\log \frac{p(z,x|\theta)}{q(z|x,\lambda)}$ based on previous batches
- A trainable constant b
- A neural network prediction based on x e.g. $b(x;\omega)$
- The reward assessed at a deterministic point, e.g. $b(x) = \log \frac{p(z^\star,x|\theta)}{q(z^\star|x,\lambda)}$ where $z^\star = \arg\max_z q(z|x,\lambda)$

Examples of baselines

- Moving average of $\log \frac{p(z,x|\theta)}{q(z|x,\lambda)}$ based on previous batches
- A trainable constant b
- A neural network prediction based on x e.g. $b(x;\omega)$
- The reward assessed at a deterministic point, e.g. $b(x) = \log \frac{p(z^\star,x|\theta)}{q(z^\star|x,\lambda)}$ where $z^\star = \arg\max_z q(z|x,\lambda)$
- The reward assessed at a stochastic point, e.g. $b(x) = \log \frac{p(z',x|\theta)}{q(z'|x,\lambda)}$ where $z' \sim q(z|x,\lambda)$

Trainable baselines

Baselines are predicted by a regression model (e.g. a neural net).

The model is trained using an L_2-loss.

$$\min_{\omega} \left(b(x;\omega) - \log \frac{p(z,x|\theta)}{q(z|x,\lambda)} \right)^2$$

Other techniques

- control variates beyond baselines: Tucker et al. (2017), Grathwohl et al. (2018)
- Rao-Blackwell: Liu et al. (2019)

Summary

Learning discrete LVMs poses challenges for gradient estimation, in particular, gradients of the inference network are challenging.

SFE is the most general, it requires tractable pmf and sampling, nothing else. It is too noisy to be useful without variance reduction techniques.

Summary

Alternatives to SFE are possible in some cases.

STE requires a relaxation of the decoder and introduces biases, violating the requirements for stochastic optimisation.

We can mix a pmf and a pdf to obtain reparameterised and unbiased gradients for a sparse rv. This addresses STE's bias.

Sparse parameterisation of the inference model leads to sparse gradients with many terms evaluating trivially to zero. Enumeration dispenses with relaxations and works for combinatorial variables.

Final Remarks

- Probabilistic models are extremely flexible tools.
- They are interesting precisely because we can make choices about unobserved aspects of the data.
- Discrete latent variables are oftentimes key to revealing interpretable structure, or to imposing some interpretable structure on a joint distribution.
- Learning discrete LVMs is challenging, but recent years have seen amazing progress.
- Join the party! Apply these models, extend them, discover problems with their estimation/evaluation, investigate solutions.
- Avoid approaching LVMs wondering whether they will beat some non-LVM NN. If such NN exists, then you are probably looking at an aspect of the problem that does not require latent variables.

What Next?

- For more material, check
  https://vitutorial.github.io/classes/
- We also have some coding exercises
  https://github.com/vitutorial/exercises
- Check this great tutorial by our friends from DeepSPIN
  https://deep-spin.github.io/tutorial/

See you around!

References I

Duygu Ataman, Wilker Aziz, and Alexandra Birch. A Latent Morphology Model for Open-Vocabulary Neural Machine Translation. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=BJxSI1SKDH.

Jasmijn Bastings, Wilker Aziz, and Ivan Titov. Interpretable neural predictions with differentiable binary variables. In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pages 2963–2977, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1284. URL https://www.aclweb.org/anthology/P19-1284.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

References II

Nicola De Cao, Michael Schlichtkrull, Wilker Aziz, and Ivan Titov. How do Decisions Emerge across Layers in Neural Models? Interpretation with Differentiable Masking. In *EMNLP*, 2020.

Gonçalo M. Correia, Vlad Niculae, Wilker Aziz, and André F. T. Martins. Efficient Marginalization of Discrete and Structured Latent Variables via Sparsity. 2020.

António Farinhas, Wilker Aziz, Vlad Niculae, and Andre Martins. Sparse communication via mixed distributions. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=WAid50QschI.

References III

Will Grathwohl, Dami Choi, Yuhuai Wu, Geoff Roeder, and David Duvenaud. Backpropagation through the Void: Optimizing control variates for black-box gradient estimation. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=SyzKd1bCW.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical Reparameterization with Gumbel-Softmax. In *International Conference on Learning Representations*, 2017.

Runjing Liu, Jeffrey Regier, Nilesh Tripuraneni, Michael Jordan, and Jon Mcauliffe. Rao-blackwellized stochastic gradients for discrete distributions. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *ICML*, volume 97 of *Proceedings of machine learning research*, pages 4023–4031, Long Beach, California, USA, June 2019. PMLR. URL http://proceedings.mlr.press/v97/liu19c.html. tex.pdf: http://proceedings.mlr.press/v97/liu19c/liu19c.pdf.

References IV

Christos Louizos, Max Welling, and Diederik P. Kingma. Learning Sparse Neural Networks through L_0 Regularization. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=H1Y8hhg0b.

Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. The Concrete Distribution: A Continous Relaxation of Discrete Random Variables. In *International Conference on Learning Representations*, 2017.

Andre Martins and Ramon Astudillo. From Softmax to Sparsemax: A Sparse Model of Attention and Multi-Label Classification. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1614–1623, New York, New York, USA, June 2016. PMLR. URL http://proceedings.mlr.press/v48/martins16.html.

References V

Tsvetomila Mihaylova, Vlad Niculae, and André FT Martins. Understanding the mechanics of spigot: Surrogate gradients for latent structure learning. In *EMNLP*, 2021.

Vlad Niculae, Andre Martins, Mathieu Blondel, and Claire Cardie. SparseMAP: Differentiable Sparse Structured Inference. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 3799–3808, Stockholmsmässan, Stockholm Sweden, July 2018a. PMLR. URL http://proceedings.mlr.press/v80/niculae18a.html.

References VI

Vlad Niculae, André F. T. Martins, and Claire Cardie. Towards Dynamic Computation Graphs via Sparse Latent Structure. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 905–911, Brussels, Belgium, October 2018b. Association for Computational Linguistics. doi: 10.18653/v1/D18-1108. URL https://www.aclweb.org/anthology/D18-1108.

Hao Peng, Sam Thomson, and Noah A. Smith. Backpropagating through structured argmax using a SPIGOT. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1863–1873, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1173. URL https://www.aclweb.org/anthology/P18-1173.

Jason Tyler Rolfe. Discrete variational autoencoders. In *ICLR*, 2017.

References VII

George Tucker, Andriy Mnih, Chris J Maddison, John Lawson, and Jascha Sohl-Dickstein. REBAR: Low-variance, unbiased gradient estimates for discrete latent variable models. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 2627–2636. Curran Associates, Inc., 2017.

Elena Voita, Rico Sennrich, and Ivan Titov. The bottom-up evolution of representations in the transformer: A study with machine translation and language modeling objectives. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4396–4406, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1448. URL https://www.aclweb.org/anthology/D19-1448.