Deep Probabilistic Models – Unobserved Data

Deep Learning 2 – 2023

Wilker Aziz
w.aziz@uva.nl

University of Amsterdam
Institute for Logic, Language and Computation

Previously

Probabilistic models prescribe the probability measure of a random experiment

- one of the many ways to achieve that is to specify the pdf (or pmf) of a collection of random variables

Previously

Probabilistic models prescribe the probability measure of a random experiment

- one of the many ways to achieve that is to specify the pdf (or pmf) of a collection of random variables
- a deep probabilistic model uses NNs to parameterise this pdf

Previously

Probabilistic models prescribe the probability measure of a random experiment

- one of the many ways to achieve that is to specify the pdf (or pmf) of a collection of random variables
- a deep probabilistic model uses NNs to parameterise this pdf

For parameter estimation, we decided to employ MLE

Previously

Probabilistic models prescribe the probability measure of a random experiment

- one of the many ways to achieve that is to specify the pdf (or pmf) of a collection of random variables
- a deep probabilistic model uses NNs to parameterise this pdf

For parameter estimation, we decided to employ MLE

- with a tractable and differentiable likelihood function, gradient-based search for NN parameters gives us a general purpose mechanism to approach supervised learning

Previously

Probabilistic models prescribe the probability measure of a random experiment

- one of the many ways to achieve that is to specify the pdf (or pmf) of a collection of random variables
- a deep probabilistic model uses NNs to parameterise this pdf

For parameter estimation, we decided to employ MLE

- with a tractable and differentiable likelihood function, gradient-based search for NN parameters gives us a general purpose mechanism to approach supervised learning

Finally, once the probabilistic model is fully specified (which includes parameter estimation), together with a decision rule, it can power applications (tasks).

Goals for this session

1. Prescribe joint distribution involving discrete and unobserved random variables
2. Estimate parameter via gradient-based MLE
3. Recognise the role of a model's posterior distribution in parameter estimation

Outline

Modelling observed random variables

Our goal is to learn a distribution over a set of **observed** random variables.

Observed random variables are the result of random experiments that have already happened: e.g., sentences in a collection of news articles, number of stars in a product review.

Modelling unobserved random variables

Unobserved random variables are variables that are

- observable in principle, but not available for observation (e.g., the topic of a piece of text, a semantic graph)
- unobservable (e.g., a 100-dimensional sentence embedding)

they help us prescribe and even estimate our models.

Our goal is to learn a distribution over *observed and unobserved rvs*

- make explicit assumptions about statistical dependence
- discover hidden structure
- mimic intuitions/knowledge about the data generating process
- deal with missing data
- estimate uncertainty about predictions

Deterministic predictors may also be available.

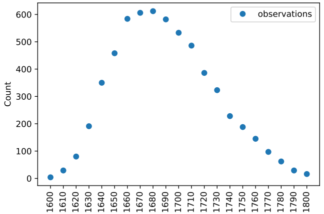
Unobserved random variables are also called latent variables.

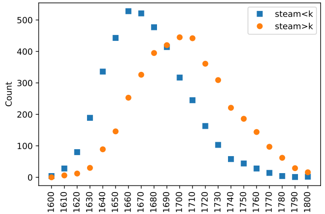
For those interested in Bayesian statistics, note that the presence of unobserved random variables does not imply Bayesian modelling. Bayesian principles are a collection of ideas organised in what is called the Bayesian Theory (or Bayesian Decision Theory) for rational decision making under uncertainty (Bernardo and Smith, 2009). These ideas may cross paths with many aspects of our ML solutions.

Outline

Latent Structure and Over-Dispersion

We found some old manuscripts in an excavation site, historians began labelling them for publication date.





Marginally (left), it looked like our observations could have been drawn from a Poisson distribution.

Left: publication date of documents in the labelled collection.

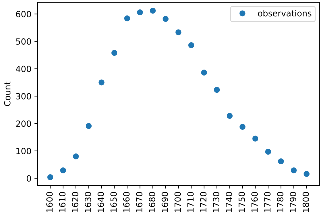
Our historians claim that above some threshold k a document was likely written after 1699 (when Thomas Savery demonstrated his first steam engine to the British Royal society).

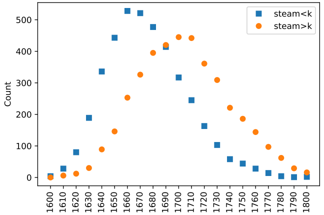
Right: data as a function of the frequency of the word *steam*.

Plotting two streams of data under such criterion reveals what could have been 2 different Poisson distributions.

Latent Structure and Over-Dispersion

We found some old manuscripts in an excavation site, historians began labelling them for publication date.





But they might also have been the result of mixing (right) into one population draws from two different Poisson distributions.

Left: publication date of documents in the labelled collection.

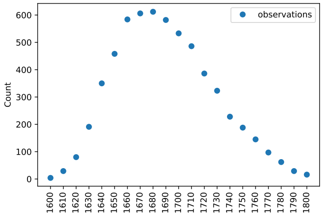
Our historians claim that above some threshold k a document was likely written after 1699 (when Thomas Savery demonstrated his first steam engine to the British Royal society).

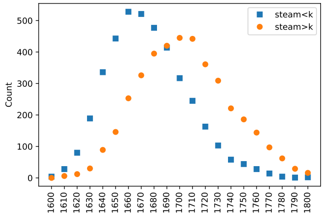
Right: data as a function of the frequency of the word *steam*.

Plotting two streams of data under such criterion reveals what could have been 2 different Poisson distributions.

Latent Structure and Over-Dispersion

We found some old manuscripts in an excavation site, historians began labelling them for publication date.





The way a model *views* the data tells us something about latent factors that account for (cause or correlate with) observed variance.

Left: publication date of documents in the labelled collection.

Our historians claim that above some threshold k a document was likely written after 1699 (when Thomas Savery demonstrated his first steam engine to the British Royal society).

Right: data as a function of the frequency of the word *steam*.

Plotting two streams of data under such criterion reveals what could have been 2 different Poisson distributions.

"Oh, latent variables are like hidden units, right?"

Latent structure here has to do with a partitioning of the probability space in terms of intermediate outcomes that depend on one another.

Hidden layers in an NN output deterministic transformations of their observed inputs. They are not statements about statistical dependence.

Example:

$$Y|h \sim \text{Poisson}\left(\underbrace{\exp\left(\sum_{i=1}^{D} w_i h_i\right)}_{\text{shallow NN}}\right)$$

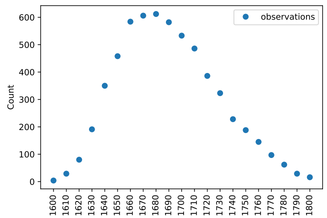
any one draw comes from the exact same Poisson.

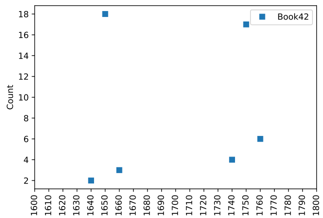
To reveal latent structure that is likely supported by observations, we need to postulate a joint distribution where observations and latent variables interact.

'Interacting' is a matter of **statistical dependence**.

Latent Structure and Multimodality

For a few manuscripts, we obtained labels from multiple experts.





Left: observations for Y across the collection. Right: observations for Y given doc is book-42.

a unimodal conditional $Y|\mathrm{doc}$ seems inappropriate.

When we plot observations for Y (e.g., left) we see the data *marginally*.

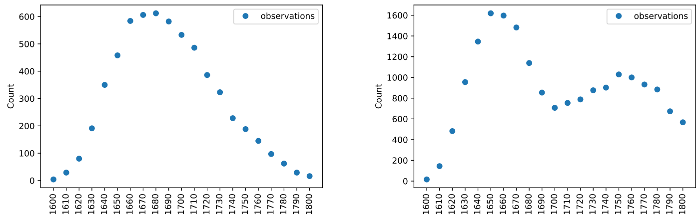
If we intend to model the data conditionally, that plot won't help us pick a family. That is because our choice should be informed by plots of the kind $Y|\mathrm{doc}$. If we group our data into bins, where bin membership depends on matching a specific value of doc, more often than not our bins will each contain a single data point. Should we conclude that *conditionally* our data can be seen as deterministic? By no means!

Be aware of sneaky modelling assumptions. The combination of *'1 bin per unique document'* and *'one plot per bin'* is a modelling choice (for visualisation purposes, but still). One that suffers from data sparsity so tremendously that it makes a random variable look deterministic. Concluding that we can model the data deterministically is in fact an instance of overfitting (by humans).

Note that sometimes we can *construct* more meaningful $Y|\mathrm{doc}$ plots that reveal the stochastic nature of the data. For example, if we have direct access to the mechanism by which observations are generated, we can fix doc and draw Y multiple times (rightmost plot on the slide).

"But NNs can, in principle, learn anything, right?"

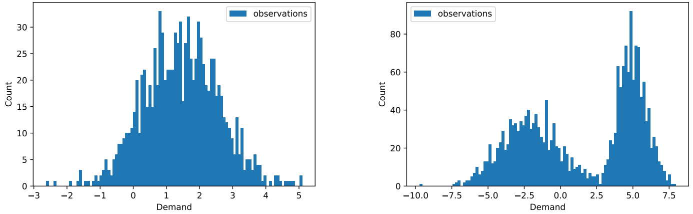
Not quite. We identify a *probability measure* by parameterising a joint pdf (or pmf). Thus our models are limited by the expressiveness of the families we choose.



Left: data look unimodal and could have been drawn from a Poisson.
Right: data look bimodal and a single Poisson seems less likely.

"But NNs can, in principle, learn anything, right?"

Not quite. We identify a *probability measure* by parameterising a joint pdf (or pmf). Thus our models are limited by the expressiveness of the families we choose.



Left: data look unimodal and could have been drawn from a Gaussian.
Right: data look bimodal and a single Gaussian seems less likely.

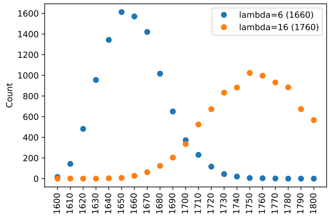
Can we combine simple distributions?

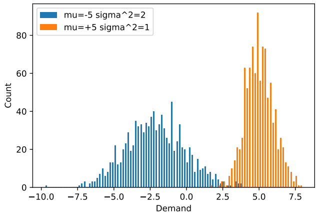
We can however *mix* K members of each family to get a good fit:

For example, with $K = 2$

$$Z|b \sim \text{Bernoulli}(b)$$
$$Y|\lambda, b \sim \text{Poisson}(\lambda_z)$$

$$Z|b \sim \text{Bernoulli}(b)$$
$$Y|\mu, \sigma, b \sim \mathcal{N}(\mu_z, \sigma_z^2)$$





This is known as a **mixture model**. The specific ones on the slide combines two conditional distributions, namely, $Y|Z = 0$ and $Y|Z = 1$. The model mixes its conditional *components* stochastically, a process controlled by a distribution over components, whose probabilities $p(z|\theta)$ are known as *mixing weights*. That is, with probability $p(z|\theta)$ the component $Y|Z = z$ generates a draw in $\mathcal{Y}$. In this example, $p(Z = 1|\theta) = b$ and $p(Z = 0|\theta) = 1 - b$.

For $K > 2$ components, $Z \sim \text{Cat}(\pi_1, \ldots, \pi_K)$, thus $p(z|\pi) = \pi_z$.

Mixture model

A **mixture model** assigns probability density

$$p_{ZY}(z, y|\theta) = p_Z(z|\theta)_{Y|Z}p(y|z, \theta)$$

to joint outcomes in $\mathcal{Z} \times \mathcal{Y}$. That is, it prescribes a *joint distribution* over observed and unobserved random variables.

A mixture model encodes the assumption that data points are each drawn from one of a finite number of independent distributions.

The latent variable Z captures this *unobserved component assignment*. It is governed by a distribution we call the *prior*. Oftentimes this is as simple as a uniform distribution over the sample space $\mathcal{Z}$.

Given an observation y drawn from the mixture, we can *infer* a distribution over component assignments by basic probability calculus, this very famous result is known as Bayes rule:

$$p(z|y, \theta) = \frac{p(z, y|\theta)}{p(y|\theta)} = \frac{p(y|z, \theta)p(z|\theta)}{\sum_{z' \in \mathcal{Z}} p(y|z', \theta)p(z'|\theta)}$$

Note that this *posterior pdf* $p(z|y, \theta)$ involves the *marginal pdf* $p(y|\theta)$, which we discuss next.

Prescribing Flexible Distributions

The **marginal** distribution of the mixture model is potentially multimodal and exhibits richer covariance structure. It assigns probability density

$$p(y|\theta) = \sum_{z=1}^{K} p(z|\theta)p(y|z,\theta)$$

to an outcome $y \in \mathcal{Y}$ by *marginalisation* of assignments $z \in \mathcal{Z}$ of the latent random variable.

Marginal inference for the MM scales linearly in the number of *components*. This is a key type of computation, for example, indispensable for parameter estimation / learning via maximum likelihood.

Say we have a dataset $\mathcal{D}$ of N i.i.d. observations for Y. MLE depends on the log-likelihood function, which in turn depends on assessments of the *marginal probability* of each observation:

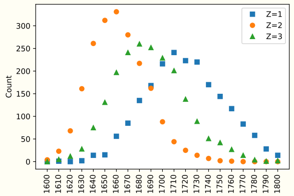
$$\begin{aligned}
\mathcal{L}_{\mathcal{D}}(\theta) &= \sum_{n=1}^{N} \log p(y^{(n)}|\theta) \\
&= \sum_{n=1}^{N} \log \sum_{z^{(n)}=1}^{K} p(y^{(n)}, z^{(n)}|\theta) \\
&= \sum_{n=1}^{N} \log \sum_{z^{(n)}=1}^{K} p(y^{(n)}|z^{(n)}, \theta)p(z^{(n)}|\theta)
\end{aligned}$$

Posterior component assignment

Given an observation y we can infer a distribution over component assignments via Bayes rule

$$p(z|y,\theta) = \frac{p(z,y|\theta)}{p(y|\theta)} = \frac{p(y|z,\theta)p(z|\theta)}{\sum_{z' \in \mathcal{Z}} p(y|z',\theta)p(z'|\theta)}$$

MMs are one of the first options when it comes to organising massive collections of unlabelled data into smaller groups (clustering).



Here is a mixture model (3 Poisson components) of our historical data.

Components in a mixture model are not *labelled* with self-evident information such as *'pre-steam-engine'* and *'post-steam-engine'*, but sometimes by inspecting likely component assignments we can recognise some salient features data bring data points together under a certain component. We can also use it to target annotation efforts, for example, to avoid under-representing certain decades (in our running example).

Labelling components with self-evident information can be done by experts with assistance of posterior queries or even semi-automatically by extending mixture models in interesting ways. See LDA (Blei et al., 2003), for example.

Predictors are welcome

It is also possible to use mixture models in conditional models:

$$p(z, y|x, \theta) = p(z|x, \theta)p(y|x, z, \theta)$$

and we may use x in different ways, e.g.

$$\begin{aligned} p(z, y|x, \theta) =& p(z|\theta)p(y|x, z, \theta) \\ p(z, y|x, \theta) =& p(z|x, \theta)p(y|z, \theta) \end{aligned}$$

Whether to use predictors to parameterise mixing weights, the conditional model, or both will depend on the application.

'Parameterising mixing weights' means specifying a distribution over K mixture components, e.g.

$$Z|x \sim \text{Cat}(g(x; \theta))$$

An alternative to giving control of mixing weights to a neural network, or fixing the weights to something superficially intuitive (like a uniform distribution), is to prescribe a *prior* distribution over the mixing coefficients. This would get you very close to Bayesian realms. Do you know any distribution which has the space of K-dimensional probability vectors as support?

Semi-supervised learning

Suppose some documents are annotated and others are not (as in the example), and say we model generatively.

For labelled documents, we observe (x, y) whose joint probability is

$$p_{XY}(x, y|\theta) = p_Y(y|\theta)_{X|Y}p(x|y, \theta)$$

and the marginal probability of an unlabelled document x is

$$p_X(x|\theta) = \sum_{y \in \mathcal{Y}} p_Y(y|\theta)p_{X|Y}(x|y, \theta)$$

For a countably finite set $\mathcal{Y}$, this is a **mixture model**!

This is a very special mixture model for its components are *labelled* with self-evident information (e.g., decades).

A generative model of this kind can be thought of as a classifier (the *task* point of view), we need only apply Bayes rule to obtain a conditional $p(y|x_*, \theta)$ that can power a decision rule for a novel document x_*.

But, above all, a generative model of this kind is a model of all of our observations (the *random experiment* point of view). Our observations are indeed a collection of documents, where some documents (very few) are labelled for decade. When we model conditionally we call the labelled instances *training data* and ignore all unlabelled instances (the vast majority of our observations).

Besides powering a classification rule, the generative formulation could be used to shed light onto vocabulary shifts over the decades. One way to specify the component $p(x|y, \theta)$ is to assume it generates a document by drawing words independently given a decade-specific parameter θ_y. That is, $X_i|Y = y \sim \text{Cat}(\theta_y)$ for $i = 1, \dots, |x|$.

Competition or cooperation?

In a mixture model the components *compete* to generate a data point. This means they cannot *cooperate* to account for some observed variance.

Sometimes, however, we want to stipulate the presence of a number of latent factors that together contribute to our observations distributing the way they do. Think of it in terms of clustering: sometimes we need overlapping clusters, or rather, *attributes*.

For example, our documents are scientific documents, and the period in consideration covers the European Scientific Revolution, as it came to be named. A number of inventions and new ideas marked this period. Documents were likely influenced by subsets of those ideas, rather than any singe idea in particular.

Like in mixture models we can recognise two roles for the class of models we are about to develop.

They can serve task-driven goals and power models that can predict attributes of an input (e.g., attributes of product, aspects of review, morphological features of a word).

They can serve knowledge-seeking goals and power inferences about latent structure that account (cause or correlate with) observed variance (e.g., in what latent aspects/dimensions are data points related).

Latent factor document model

Let us consider a latent factor model for document modelling:

- a document $x = (x_1, \dots, x_I)$ consists of I i.i.d. categorical draws from that model
- the categorical distribution in turn depends on binary latent factors $z = (z_1, \dots, z_D)$ which are also i.i.d.

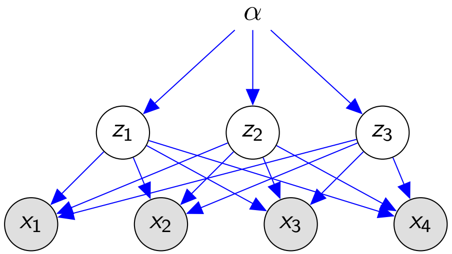
$$\begin{aligned} Z_j &\sim \text{Bernoulli}(\alpha) & (1 \leq d \leq D) \\ X_i|z &\sim \text{Categorical}\left(f(z;\theta)\right) & (1 \leq i \leq I) \end{aligned}$$

Here $f(\cdot;\theta)$ is an NN and θ its parameters.

To keep the model simple we will assume $X_i \perp X_j | Z$ for $i \neq j$. We could, however, relax this conditional independence if we wanted. For example, we could model $X_i|z, x_{<i} \sim \text{Cat}(f(z, x_{<i}; \theta))$.

Graphical model

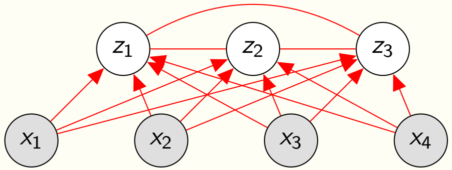
Joint distribution: independent latent variables



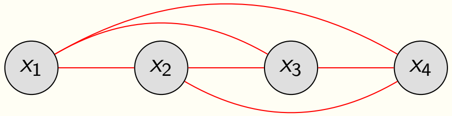
I omit θ from the graphical model, but every $X_i|z$ depends on it.

Suppose, for example, $D = 3$ and $I = 4$.

Posterior



Marginal



I'm omitting θ and α from the graphical models.

Intractable Marginals

In the latent factor model, marginalisation takes time $\mathcal{O}(2^D)$

$$
\begin{aligned}
p(x|\alpha,\theta) &= \sum_{z\in\{0,1\}^D} p(z|\alpha)p(x|z,\theta) \\
&= \sum_{z\in\{0,1\}^D} \prod_{d=1}^{D} \text{Bern}(z_d|\alpha) \prod_{i=1}^{I} \text{Cat}(x_i|f(z;\theta))
\end{aligned}
$$

As a consequence, we cannot assess $\log p(x|\alpha,\theta)$ nor its gradient.

Thinking ahead: if we cannot assess $\log p(x|\alpha,\theta)$ for an observation x, nor its gradient, how are we going to estimate parameters for this model?

Combinatorial Latent Structure

The posterior of the latent factor model reveals attributes that are relevant to an observation.

Sampling from it can help discover discrete factors of variation (e.g. morphological attributes of a word).

Unfortunately, the posterior in this case is a Gibbs distribution whose parameter is intractable to compute

- its natural parameter has length 2^D
- its log-normaliser requires a summation over $z \in \{0, 1\}^D$

Applications

Alignment Learn to match two data structures (e.g., word alignment, phrase alignment, visual question answering).

Data: $\langle x_1, \ldots, x_I \rangle$ and $\langle y_1, \ldots, y_J \rangle$

Generate each part of y using a subset of the parts of x.

- this can be a mixture model
- or a latent factor model
- and there can be constraints on the parts (e.g., disjoint)

(Rios et al., 2018; Deng et al., 2018; Kawakami et al., 2019)

Applications

Latent attribution What parts of the input (or of a computation graph) affect predictions.

Data: $\langle x_1, \ldots, x_I \rangle$ and y

$$
\begin{aligned}
Z_i &\sim \mathrm{Bern}(\alpha) \\
Y|x, z &\sim \mathrm{Cat}(f(x \odot z; \theta))
\end{aligned}
$$

x could also be every hidden state internal to a given NN, and y could be that NN's output $y = g(x; \phi)$

(Lei et al., 2016; Bastings et al., 2019; Cao et al., 2020)

Applications

Compositionality   Learn a computation graph.

Sample a structure z from a prior or conditional distribution $Z|x$ and let this structure determine a composition function to parameterise a distribution $Y|x, z$. This can be used for semi-supervised learning of syntactic/semantic representations, for learning to solve arithmetic expressions, interpretable text classifiers, etc.

(Yogatama et al., 2017; Corro and Titov, 2018; Niculae et al., 2018; Havrylov et al., 2019)

Applications

Controllable generation Learn to affect a conditional generator by controlling a latent prompt.

For example, translation models parameterise a conditional distribution $Y|x, \theta$ over translations of a given input x.

The source may contain a certain word (e.g., doctor), and the target language gender-marks nouns. The source sentence *does not* contain enough information to resolve the ambiguity, wouldn't it be nice to have a mechanism, other than requiring the user to produce a less ambiguous x, to control inflections?

(Hu et al., 2017; Zhou and Neubig, 2017; Ataman et al., 2020)

Applications

(Hu et al., 2017; Zhou and Neubig, 2017; Ataman et al., 2020)

Summary

Mixture model ('learning clusters')

Latent factor model ('learning attributes or overlapping clusters')

Applications:

- unsupervised learning (e.g., word alignments, LDA, IBP)
- semi-supervised learning (e.g., generative classifiers, disentanglement learning)
- transparency (e.g., latent rationales)

Examples:

- word alignments (Brown et al., 1993; Vogel et al., 1996; Rios et al., 2018)
- LDA (Blei et al., 2003)
- IBP (Ghahramani and Griffiths, 2006)
- semi-supervised deep generative models (Kingma et al., 2014; **?**)
- latent rationales (Lei et al., 2016; Bastings et al., 2019)

Outline

Latent Variable Models

When talking about some generic model I will follow this convention

- X is an rv taking on values in $\mathcal{X}$
- $x \in \mathcal{X}$ is an observation
- Z is a *discrete* rv taking on values in $\mathcal{Z}$
- $z \in \mathcal{Z}$ is a latent assignment
- the joint pdf factorises as $p(x, z|\theta) = p(z|\theta)p(x|z, \theta)$
- $p(z|\theta)$ is called the *prior*
- $p(x|z, \theta)$ is called the *observational model*
- $p(x|\theta)$ is the *marginal* (or *evidence*)
- and $p(z|x, \theta) = \frac{p(x,z|\theta)}{p(x|\theta)}$ is the posterior
- anything in the model can be parameterised by NNs

Throughout, we shall assume we have N i.i.d. observations. With deterministic parameters θ, we can make all our arguments in terms of a single observation x. Recall, the likelihood-function $\mathcal{L}_{\mathcal{D}}(\theta)$ is just $\sum_{x \sim \mathcal{D}} \log p(x|\theta)$.

By the way, can you draw a plate diagram for our generic latent variable model?

Many models admit exact marginals

Examples (and the algorithms for marginalisation)

- Mixture models (enumeration)
- HMMs (forward algorithm)
- CFGs (inside algorithm)
- Spanning-tree random fields (matrix-tree theorem)

Tractable marginalisation depends on the conditional independence assumptions of a model (e.g., in an HMM a hidden state is independent of all but its preceding state), not on how that model's probability distributions are parameterised (e.g., a transition distribution in the HMM may be stored in a table, predicted by a log-linear model or by an NN).

Marginalisation algorithms are generally harder to parallelise on GPUs.

Recall that to use NNs in probabilistic models we converged to two constraints on the log-likelihood function:

- differentiability with respect to parameters
- and tractability

If $p(z|\theta)$ and $p(x|z,\theta)$ are differentiable functions of their parameters, there is no impediment to gradient-based parameter estimation. Can you show that to yourself? Hint: expand $\boldsymbol{\nabla}_\theta \log p(x|\theta)$.

Tractability depends on whether $p(x|\theta) = \sum_{z\in\mathcal{Z}} p(x,z|\theta)$, or its logarithm, can be evaluated in feasible time. Though it may seem so, this is not always a matter of cardinality of $\mathcal{Z}$.

For example, there is a Catalan number of trees in a CFG, yet because of the strong independence assumptions in the model, the marginal $p(x|\theta)$ is computable in cubic-time (w.r.t. sequence length) via the inside algorithm. Similarly, there is an exponential number of state sequences in an HMM, but its marginal is computable in linear-time (w.r.t. sequence length) via the forward algorithm.

Neural {MM, HMM, CFG, CRF, ... }

An NN-parameterisation of a classic discrete LVM, for which exact marginals are tractable, still needs to preserve all of that model's statistical assumptions about unobserved random variables.

We won't necessarily achieve a more complex distribution.

Though we may condition on complex data more effectively.

A neural HMM could look like:

$$p(x|\theta) = \sum_{z\in\{1,\ldots,K\}^{|x|}} \prod_{i=1}^{|x|} \underbrace{p(z_i|z_{i-1}, x_{<i}, \theta)}_{\mathrm{Cat}(z_i|g(x_{<i}, z_{i-1};\theta))} \underbrace{p(x_i|z_i, x_{<i})}_{\mathrm{Cat}(x_i|f(x_{<i}, z_i;\theta))}$$

the entire history of already generated words is available for conditioning

NNs allow us to condition on complex observations, like a long history of words $x_{<i}$ in unsupervised part-of-speech tagging.

We cannot, as easily, exploit that power to relax statistical conditional independence assumptions, for those assumptions are crucial in order to maintain *exact and tractable* access to marginal probabilities.

Think of it this way, what makes the HMM the HMM is the first-order (or n-order) Markov assumption $Z_i \perp Z_j|Z_{i-1}$ for j other than i and $i-1$. Relaxing that turns the HMM into something else, for which exact inference is likely impossible. See Wang et al. (2018) for a neural HMM.

Gradient-based MLE

What happens when we autodiff the quantity $\log p(x|\theta)$, which we computed exactly and tractably?

Let's inspect this gradient ourselves $\nabla_\theta \log p(x|\theta)$

Gradient of log-marginal

- It all starts with the derivative of log, followed by chain rule again.

Gradient-based MLE

What happens when we autodiff the quantity $\log p(x|\theta)$, which we computed exactly and tractably?

Let's inspect this gradient ourselves $\nabla_\theta \log p(x|\theta)$

$$= \frac{1}{p(x|\theta)} \nabla_\theta p(x|\theta) =$$

Gradient of log-marginal

- It all starts with the derivative of log, followed by chain rule again.
- The next step requires marginalisation.

Gradient-based MLE

What happens when we autodiff the quantity $\log p(x|\theta)$, which we computed exactly and tractably?

Let's inspect this gradient ourselves $\nabla_\theta \log p(x|\theta)$

$$= \frac{1}{p(x|\theta)} \nabla_\theta p(x|\theta) = \frac{1}{p(x|\theta)} \nabla_\theta \sum_{z \in \mathcal{Z}} p(z, x|\theta)$$

Gradient of log-marginal

- It all starts with the derivative of log, followed by chain rule again.
- The next step requires marginalisation.
- Now we need the gradient of a big sum.

Gradient-based MLE

What happens when we autodiff the quantity $\log p(x|\theta)$, which we computed exactly and tractably?

Let's inspect this gradient ourselves $\nabla_\theta \log p(x|\theta)$

$$
\begin{aligned}
&= \frac{1}{p(x|\theta)} \nabla_\theta p(x|\theta) = \frac{1}{p(x|\theta)} \nabla_\theta \sum_{z \in \mathcal{Z}} p(z, x|\theta) \\
&= \frac{1}{p(x|\theta)} \sum_{z \in \mathcal{Z}} \nabla_\theta p(z, x|\theta) =
\end{aligned}
$$

Gradient of log-marginal

- It all starts with the derivative of log, followed by chain rule again.
- The next step requires marginalisation.
- Now we need the gradient of a big sum.
- Derivatives are linear, so we can sum gradients instead.

Gradient-based MLE

What happens when we autodiff the quantity $\log p(x|\theta)$, which we computed exactly and tractably?

Let's inspect this gradient ourselves $\nabla_\theta \log p(x|\theta)$

$$
\begin{aligned}
&= \frac{1}{p(x|\theta)} \nabla_\theta p(x|\theta) = \frac{1}{p(x|\theta)} \nabla_\theta \sum_{z \in \mathcal{Z}} p(z, x|\theta) \\
&= \frac{1}{p(x|\theta)} \sum_{z \in \mathcal{Z}} \nabla_\theta p(z, x|\theta) = \frac{1}{p(x|\theta)} \sum_{z \in \mathcal{Z}} p(z, x|\theta) \nabla_\theta \log p(z, x|\theta)
\end{aligned}
$$

Gradient of log-marginal

- It all starts with the derivative of log, followed by chain rule again.
- The next step requires marginalisation.
- Now we need the gradient of a big sum.
- Derivatives are linear, so we can sum gradients instead.
- Sums are fine, but let's use the log identity $f' = f(\log f)'$

Gradient-based MLE

What happens when we autodiff the quantity $\log p(x|\theta)$, which we computed exactly and tractably?

Let's inspect this gradient ourselves $\nabla_\theta \log p(x|\theta)$

$$
\begin{aligned}
&= \frac{1}{p(x|\theta)} \nabla_\theta p(x|\theta) = \frac{1}{p(x|\theta)} \nabla_\theta \sum_{z \in \mathcal{Z}} p(z, x|\theta) \\
&= \frac{1}{p(x|\theta)} \sum_{z \in \mathcal{Z}} \nabla_\theta p(z, x|\theta) = \frac{1}{p(x|\theta)} \sum_{z \in \mathcal{Z}} p(z, x|\theta) \nabla_\theta \log p(z, x|\theta) \\
&= \sum_{z \in \mathcal{Z}} \frac{p(z, x|\theta)}{p(x|\theta)} \nabla_\theta \log p(z, x|\theta) =
\end{aligned}
$$

Gradient of log-marginal

- It all starts with the derivative of log, followed by chain rule again.
- The next step requires marginalisation.
- Now we need the gradient of a big sum.
- Derivatives are linear, so we can sum gradients instead.
- Sums are fine, but let's use the log identity $f' = f(\log f)'$
- The marginal is constant for $z \in \mathcal{Z}$, so distribute it over the sum.

Gradient-based MLE

What happens when we autodiff the quantity $\log p(x|\theta)$, which we computed exactly and tractably?

Let's inspect this gradient ourselves $\nabla_\theta \log p(x|\theta)$

$$
\begin{aligned}
&= \frac{1}{p(x|\theta)} \nabla_\theta p(x|\theta) = \frac{1}{p(x|\theta)} \nabla_\theta \sum_{z \in \mathcal{Z}} p(z, x|\theta) \\
&= \frac{1}{p(x|\theta)} \sum_{z \in \mathcal{Z}} \nabla_\theta p(z, x|\theta) = \frac{1}{p(x|\theta)} \sum_{z \in \mathcal{Z}} p(z, x|\theta) \nabla_\theta \log p(z, x|\theta) \\
&= \sum_{z \in \mathcal{Z}} \frac{p(z, x|\theta)}{p(x|\theta)} \nabla_\theta \log p(z, x|\theta) = \sum_{z \in \mathcal{Z}} p(z|x, \theta) \nabla_\theta \log p(z, x|\theta)
\end{aligned}
$$

Gradient of log-marginal

- It all starts with the derivative of log, followed by chain rule again.
- The next step requires marginalisation.
- Now we need the gradient of a big sum.
- Derivatives are linear, so we can sum gradients instead.
- Sums are fine, but let's use the log identity $f' = f(\log f)'$
- The marginal is constant for $z \in \mathcal{Z}$, so distribute it over the sum.
- This gives us a recognisable object! Joint probability, divided by evidence, that's the posterior! And we have a weighted average, coefficients given by a pmf, and we sum over the entire support $\mathcal{Z}$.

Gradient-based MLE

What happens when we autodiff the quantity $\log p(x|\theta)$, which we computed exactly and tractably?

Let's inspect this gradient ourselves $\nabla_\theta \log p(x|\theta)$

$$
\begin{aligned}
&= \frac{1}{p(x|\theta)} \nabla_\theta p(x|\theta) = \frac{1}{p(x|\theta)} \nabla_\theta \sum_{z \in \mathcal{Z}} p(z, x|\theta) \\
&= \frac{1}{p(x|\theta)} \sum_{z \in \mathcal{Z}} \nabla_\theta p(z, x|\theta) = \frac{1}{p(x|\theta)} \sum_{z \in \mathcal{Z}} p(z, x|\theta) \nabla_\theta \log p(z, x|\theta) \\
&= \sum_{z \in \mathcal{Z}} \frac{p(z, x|\theta)}{p(x|\theta)} \nabla_\theta \log p(z, x|\theta) = \sum_{z \in \mathcal{Z}} p(z|x, \theta) \nabla_\theta \log p(z, x|\theta) \\
&= \mathbb{E}_{p(z|x,\theta)} \left[\nabla_\theta \log p(z, x|\theta)\right]
\end{aligned}
$$

Gradient of log-marginal

- It all starts with the derivative of log, followed by chain rule again.
- The next step requires marginalisation.
- Now we need the gradient of a big sum.
- Derivatives are linear, so we can sum gradients instead.
- Sums are fine, but let's use the log identity $f' = f(\log f)'$
- The marginal is constant for $z \in \mathcal{Z}$, so distribute it over the sum.
- This gives us a recognisable object! Joint probability, divided by evidence, that's the posterior! And we have a weighted average, coefficients given by a pmf, and we sum over the entire support $\mathcal{Z}$.
- We have an expectation! The gradient of the log-marginal of x is the expected gradient of the log joint probability of x and z, where x is observed and z is a draw from the posterior distribution $Z|x, \theta$. Dependency on Z makes the gradient of log-joint $G(Z) = \nabla_\theta \log P(Z, X = x)$ a random variable.

Gradient-based MLE

What happens when we autodiff the quantity $\log p(x|\theta)$, which we computed exactly and tractably?

Let's inspect this gradient ourselves $\nabla_\theta \log p(x|\theta)$

$$
\begin{aligned}
&= \frac{1}{p(x|\theta)} \nabla_\theta p(x|\theta) = \frac{1}{p(x|\theta)} \nabla_\theta \sum_{z \in \mathcal{Z}} p(z, x|\theta) \\
&= \frac{1}{p(x|\theta)} \sum_{z \in \mathcal{Z}} \nabla_\theta p(z, x|\theta) = \frac{1}{p(x|\theta)} \sum_{z \in \mathcal{Z}} p(z, x|\theta) \nabla_\theta \log p(z, x|\theta) \\
&= \sum_{z \in \mathcal{Z}} \frac{p(z, x|\theta)}{p(x|\theta)} \nabla_\theta \log p(z, x|\theta) = \sum_{z \in \mathcal{Z}} p(z|x, \theta) \nabla_\theta \log p(z, x|\theta) \\
&= \mathbb{E}_{p(z|x,\theta)} \left[\nabla_\theta \log p(z, x|\theta) \right]
\end{aligned}
$$

Autodiff performs **exact posterior inference** for us!

Gradient of log-marginal

- It all starts with the derivative of log, followed by chain rule again.
- The next step requires marginalisation.
- Now we need the gradient of a big sum.
- Derivatives are linear, so we can sum gradients instead.
- Sums are fine, but let's use the log identity $f' = f(\log f)'$
- The marginal is constant for $z \in \mathcal{Z}$, so distribute it over the sum.
- This gives us a recognisable object! Joint probability, divided by evidence, that's the posterior! And we have a weighted average, coefficients given by a pmf, and we sum over the entire support $\mathcal{Z}$.
- We have an expectation! The gradient of the log-marginal of x is the expected gradient of the log joint probability of x and z, where x is observed and z is a draw from the posterior distribution $Z|x, \theta$. Dependency on Z makes the gradient of log-joint $G(Z) = \nabla_\theta \log P(Z, X = x)$ a random variable.
- The gradient of the log-marginal $\nabla_\theta \log p(x|\theta)$ is deterministic, it is the expected value $\mathbb{E}_{Z|X=x,\theta}[G(Z)]$. You evaluate the marginal, autodiff evaluates the expectation.

Gradient-based MLE

What happens when we autodiff the quantity $\log p(x|\theta)$, which we computed exactly and tractably?

Let's inspect this gradient ourselves $\nabla_\theta \log p(x|\theta)$

$$
\begin{aligned}
&= \frac{1}{p(x|\theta)} \nabla_\theta p(x|\theta) = \frac{1}{p(x|\theta)} \nabla_\theta \sum_{z \in \mathcal{Z}} p(z, x|\theta) \\
&= \frac{1}{p(x|\theta)} \sum_{z \in \mathcal{Z}} \nabla_\theta p(z, x|\theta) = \frac{1}{p(x|\theta)} \sum_{z \in \mathcal{Z}} p(z, x|\theta) \nabla_\theta \log p(z, x|\theta) \\
&= \sum_{z \in \mathcal{Z}} \frac{p(z, x|\theta)}{p(x|\theta)} \nabla_\theta \log p(z, x|\theta) = \sum_{z \in \mathcal{Z}} p(z|x, \theta) \nabla_\theta \log p(z, x|\theta) \\
&= \mathbb{E}_{p(z|x,\theta)} \left[\nabla_\theta \log p(z, x|\theta) \right]
\end{aligned}
$$

Autodiff performs **exact posterior inference** for us!

Gradient of log-marginal

- It all starts with the derivative of log, followed by chain rule again.
- The next step requires marginalisation.
- Now we need the gradient of a big sum.
- Derivatives are linear, so we can sum gradients instead.
- Sums are fine, but let's use the log identity $f' = f(\log f)'$
- The marginal is constant for $z \in \mathcal{Z}$, so distribute it over the sum.
- This gives us a recognisable object! Joint probability, divided by evidence, that's the posterior! And we have a weighted average, coefficients given by a pmf, and we sum over the entire support $\mathcal{Z}$.
- We have an expectation! The gradient of the log-marginal of x is the expected gradient of the log joint probability of x and z, where x is observed and z is a draw from the posterior distribution $Z|x, \theta$. Dependency on Z makes the gradient of log-joint $G(Z) = \nabla_\theta \log P(Z, X = x)$ a random variable.
- The gradient of the log-marginal $\nabla_\theta \log p(x|\theta)$ is deterministic, it is the expected value $\mathbb{E}_{Z|X=x,\theta}[G(Z)]$. You evaluate the marginal, autodiff evaluates the expectation.

Summary

Many discrete LVMs admit tractable marginalisation

Assessing the gradient of the log-marginal probability of an observation corresponds to assessing an expectation under the posterior distribution over latent variables. Think of it this way:

- we need posterior inference to compute the gradient
- and we need the gradient for parameter estimation
- with exact marginals, autodiff assesses the gradient thus abstracting posterior inference away

What happens when we cannot solve $\sum_{z \in \mathcal{Z}} p(x, z|\theta)$?

Many interesting models are such that the exact marginal is intractable. We've seen, for example, the case where $p(x, z|\theta)$ is a latent factor model.

Autodiff cannot differentiate a quantity that cannot be assessed. So if we cannot compute the exact log-marginal probability of an observation, we won't get automatic posterior inference for free. We will have to resort to rather explicit approaches to approximate inference.

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 due posterior inference.
- When posterior inference is tractable, MLE training via stochastic gradient optimisation is available out of the box.

What next?

To model with *any* type of unobserved data we need the power of approximate inference tools, see our Amortised Variational Inference module.

Curious about concrete instances of probabilistic models that are state-of-the-art? Check our Advanced Generative Models module.

References I

Alexander Alemi, Ben Poole, Ian Fischer, Joshua Dillon, Rif A. Saurous, and Kevin Murphy. Fixing a Broken ELBO. 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 159–168, Stockholmsmässan, Stockholm Sweden, July 2018. PMLR. URL http://proceedings.mlr.press/v80/alemi18a.html.

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.

References II

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.

José M Bernardo and Adrian FM Smith. *Bayesian theory*, volume 405. John Wiley & Sons, 2009.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent Dirichlet Allocation. *J. Mach. Learn. Res.*, 3:993–1022, 2003. ISSN 1532-4435. URL http://dl.acm.org/citation.cfm?id=944919.944937. Publisher: JMLR.org.

References III

David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112(518):859–877, 2017. Publisher: Taylor & Francis.

Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. The Mathematics of Statistical Machine Translation: Parameter Estimation. *Computational Linguistics*, 19(2):263–311, 1993. URL https://www.aclweb.org/anthology/J93-2003.

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.

Caio Corro and Ivan Titov. Differentiable Perturb-and-Parse: Semi-Supervised Parsing with a Structured Variational Autoencoder. In *ICLR*, September 2018. URL https://openreview.net/forum?id=BJlgNh0qKQ.

References IV

Yuntian Deng, Yoon Kim, Justin Chiu, Demi Guo, and Alexander Rush. Latent Alignment and Variational Attention. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 9712–9724. Curran Associates, Inc., 2018. URL http://papers.nips.cc/paper/8179-latent-alignment-and-variational-attention.pdf.

Zoubin Ghahramani and Thomas L. Griffiths. Infinite latent feature models and the Indian buffet process. In Y. Weiss, B. Schölkopf, and J. C. Platt, editors, *Advances in Neural Information Processing Systems 18*, pages 475–482. MIT Press, 2006.

References V

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.

Evan Greensmith, Peter L. Bartlett, and Jonathan Baxter. Variance Reduction Techniques for Gradient Estimates in Reinforcement Learning. *Journal of Machine Learning Research*, 5(Nov):1471–1530, 2004. ISSN ISSN 1533-7928. URL https://www.jmlr.org/papers/v5/greensmith04a.html.

Shixiang Gu, Sergey Levine, Ilya Sutskever, and Andriy Mnih. MuProp: Unbiased backpropagation for stochastic neural networks. In *ICLR (poster)*, 2016. URL http://arxiv.org/abs/1511.05176. tex.cdate: 1451606400000 tex.crossref: conf/iclr/2016.

References VI

Serhii Havrylov, Germán Kruszewski, and Armand Joulin. Cooperative learning of disjoint syntax and semantics. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1118–1128, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1115. URL https://www.aclweb.org/anthology/N19-1115.

G. E. Hinton, P. Dayan, B. J. Frey, and R. M. Neal. The Wake-Sleep Algorithm for Unsupervised Neural Networks. *Science*, 268:1158–1161, 1995.

References VII

Matthew D Hoffman and Matthew J Johnson. Elbo surgery: yet another way to carve up the variational evidence lower bound. In *Workshop in Advances in Approximate Bayesian Inference, NIPS*, volume 1, page 2, 2016.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P. Xing. Toward controlled generation of text. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1587–1596. PMLR, 06–11 Aug 2017. URL http://proceedings.mlr.press/v70/hu17e.html.

Michael I. Jordan, Zoubin Ghahramani, Tommi S. Jaakkola, and Lawrence K. Saul. An Introduction to Variational Methods for Graphical Models. *Machine Learning*, 37(2):183–233, November 1999. ISSN 1573-0565. doi: 10.1023/A:1007665907178. URL https://doi.org/10.1023/A:1007665907178.

References VIII

Kazuya Kawakami, Chris Dyer, and Phil Blunsom. Learning to discover, ground and use words with segmental neural language models. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6429–6441, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1645. URL https://www.aclweb.org/anthology/P19-1645.

Durk P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. Semi-supervised Learning with Deep Generative Models. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3581–3589. Curran Associates, Inc., 2014.

References IX

Tao Lei, Regina Barzilay, and Tommi Jaakkola. Rationalizing Neural Predictions. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 107–117, Austin, Texas, November 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1011. URL https://www.aclweb.org/anthology/D16-1011.

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 X

Thomas P. Minka. Expectation propagation for approximate bayesian inference. In *Proceedings of the seventeenth conference on uncertainty in artificial intelligence*, UAI'01, pages 362–369, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc. ISBN 1-55860-800-1. Number of pages: 8 Place: Seattle, Washington.

Andriy Mnih and Karol Gregor. Neural Variational Inference and Learning in Belief Networks. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, pages II–1791–II–1799. JMLR.org, 2014. event-place: Beijing, China.

Andriy Mnih and Danilo Rezende. Variational inference for monte carlo objectives. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *ICML*, volume 48 of *Proceedings of machine learning research*, pages 2188–2196, New York, New York, USA, June 2016. PMLR. URL http://proceedings.mlr.press/v48/mnihb16.html. tex.pdf: http://proceedings.mlr.press/v48/mnihb16.pdf.

References XI

Shakir Mohamed, Mihaela Rosca, Michael Figurnov, and Andriy Mnih. Monte Carlo Gradient Estimation in Machine Learning. *CoRR*, abs/1906.10652, 2019. URL http://arxiv.org/abs/1906.10652.

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 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1108. URL https://www.aclweb.org/anthology/D18-1108.

Ben Poole, Sherjil Ozair, Aaron Van Den Oord, Alex Alemi, and George Tucker. On variational bounds of mutual information. In *International Conference on Machine Learning*, pages 5171–5180. PMLR, 2019.

References XII

Rajesh Ranganath, Sean Gerrish, and David Blei. Black Box Variational Inference. In Samuel Kaski and Jukka Corander, editors, *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics*, volume 33 of *Proceedings of Machine Learning Research*, pages 814–822, Reykjavik, Iceland, April 2014. PMLR. URL http://proceedings.mlr.press/v33/ranganath14.html.

Steven J. Rennie, Etienne Marcheret, Youssef Mroueh, Jerret Ross, and Vaibhava Goel. Self-Critical Sequence Training for Image Captioning. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 1179–1195. IEEE Computer Society, 2017. doi: 10.1109/CVPR.2017.131. URL https://doi.org/10.1109/CVPR.2017.131.

References XIII

Miguel Rios, Wilker Aziz, and Khalil Sima'an. Deep Generative Model for Joint Alignment and Word Representation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1011–1023, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1092. URL https://www.aclweb.org/anthology/N18-1092.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms. *CoRR*, abs/1707.06347, 2017. URL http://arxiv.org/abs/1707.06347.

References XIV

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.

Aki Vehtari, Andrew Gelman, Tuomas Sivula, Pasi Jylänki, Dustin Tran, Swupnil Sahai, Paul Blomstedt, John P Cunningham, David Schiminovich, and Christian P Robert. Expectation propagation as a way of life: A framework for bayesian inference on partitioned data. *Journal of Machine Learning Research*, 21(17):1–53, 2020.

References XV

Stephan Vogel, Hermann Ney, and Christoph Tillmann. HMM-Based word alignment in statistical translation. In *COLING 1996 volume 2: The 16th international conference on computational linguistics*, 1996. URL https://www.aclweb.org/anthology/C96-2141.

Weiyue Wang, Derui Zhu, Tamer Alkhouli, Zixuan Gan, and Hermann Ney. Neural Hidden Markov Model for Machine Translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 377–382, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-2060. URL https://www.aclweb.org/anthology/P18-2060.

References XVI

Ronald J. Williams. Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning. *Machine Learning*, 8(3-4): 229–256, May 1992. ISSN 0885-6125. doi: 10.1007/BF00992696. URL https://doi.org/10.1007/BF00992696.

Dani Yogatama, Phil Blunsom, Chris Dyer, Edward Grefenstette, and Wang Ling. Learning to compose words into sentences with reinforcement learning. In *ICLR*, 2017.

Chunting Zhou and Graham Neubig. Multi-space variational encoder-decoders for semi-supervised labeled sequence transduction. In *Proceedings of the 55th annual meeting of the association for computational linguistics (volume 1: Long papers)*, pages 310–320, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1029. URL https://www.aclweb.org/anthology/P17-1029.