

# Welcome and Introduction

Philip Schulz and Wilker Aziz

<https://github.com/philschulz/VITutorial>

# About us . . .

## Wilker Aziz

- ▶ Research associate at UvA
- ▶ VI, Sampling methods, Machine Translation

## Philip Schulz

- ▶ Applied Scientist at Amazon
- ▶ VI, Machine Translation, Bayesian Models

# Why are we here today?

Because NN models work

- ▶ at least one drawback is low data efficiency

# Why are we here today?

Because NN models work

- ▶ at least one drawback is low data efficiency

We usually turn to probabilistic models to deal with

- ▶ lack of supervision
- ▶ partial supervision
- ▶ lack of inductive bias

# Why are we here today?

Because NN models work

- ▶ at least one drawback is low data efficiency

We usually turn to probabilistic models to deal with

- ▶ lack of supervision
- ▶ partial supervision
- ▶ lack of inductive bias

But it turns out NNs and probabilistic modelling do not go along very well

- ▶ well...

# Why are we here today?

Because NN models work

- ▶ at least one drawback is low data efficiency

We usually turn to probabilistic models to deal with

- ▶ lack of supervision
- ▶ partial supervision
- ▶ lack of inductive bias

But it turns out NNs and probabilistic modelling do not go along very well

- ▶ well... that's no longer true!

# What are you getting out of this today?

As we progress we will

- ▶ develop a shared vocabulary to talk about generative models powered by NNs
- ▶ deriving crucial results step by step

# What are you getting out of this today?

As we progress we will

- ▶ develop a shared vocabulary to talk about generative models powered by NNs
- ▶ deriving crucial results step by step

Goal

- ▶ you should be able to navigate through fresh literature
- ▶ and start combining probabilistic models and NNs



# Problems

Supervised problems: “learn a distribution over observed data”

- ▶ sentences in natural language, images, videos,  
...

Unsupervised problems: “learn a distribution over observed and unobserved data”

- ▶ sentences in natural language + parse trees,  
images + bounding boxes ...

# Supervised problems

We have data  $x^{(1)}, \dots, x^{(N)}$  e.g.

- ▶ sentences, images, ...

generated by some **unknown** procedure

# Supervised problems

We have data  $x^{(1)}, \dots, x^{(N)}$  e.g.

- ▶ sentences, images, ...

generated by some **unknown** procedure  
which we assume can be captured by a probabilistic model

- ▶ with **known** probability (mass/density) function e.g.

$$X \sim \text{Cat}(\pi_1, \dots, \pi_K) \quad \text{or} \quad X \sim \mathcal{N}(\mu, \sigma^2)$$

# Supervised problems

We have data  $x^{(1)}, \dots, x^{(N)}$  e.g.

- ▶ sentences, images, ...

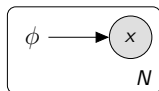
generated by some **unknown** procedure  
which we assume can be captured by a probabilistic model

- ▶ with **known** probability (mass/density) function e.g.

$$X \sim \text{Cat}(\pi_1, \dots, \pi_K) \quad \text{or} \quad X \sim \mathcal{N}(\mu, \sigma^2)$$

and proceed to **estimate parameters** that assign maximum likelihood to observations

# Multiple problems, same language



(Conditional) Density estimation

Parsing	Side information ( $\phi$ ) a sentence	Observation ( $x$ ) its syntactic/semantic parse tree/graph
Translation	a sentence	its translation
Captioning	an image	caption in English
Entailment	a text and hypothesis	entailment relation

# Where does deep learning kick in?

Let  $\phi$  be all side information available  
e.g. deterministic *inputs/features*

Have neural networks predict parameters of our probabilistic model

$$X|\phi \sim \text{Cat}(\pi_{\theta}(\phi)) \quad \text{or} \quad X|\phi \sim \mathcal{N}(\mu_{\theta}(\phi), \sigma_{\theta}(\phi)^2)$$

and proceed to **estimate parameters**  $\theta$  of the NNs

# Task-driven feature extraction

Often our side information  $\phi$  is itself some high dimensional data

- ▶  $\phi$  is a sentence and  $x$  a tree
- ▶  $\phi$  is the source sentence and  $x$  is the target
- ▶  $\phi$  is an image and  $x$  is a caption

and part of the job of the NNs that parametrise our models is to also **deterministically** encode that input in a low-dimensional space

# NN as efficient parametrisation

From the statistical point of view NNs do not generate data

- ▶ they parametrise distributions that *by assumption* govern data
- ▶ compact and efficient way to map from complex side information to parameter space



# NN as efficient parametrisation

From the statistical point of view NNs do not generate data

- ▶ they parametrise distributions that *by assumption* govern data
- ▶ compact and efficient way to map from complex side information to parameter space

Prediction is done by a decision rule outside the statistical model

- ▶ e.g. beam search

# Maximum likelihood estimation

Let  $p(x|\theta)$  be the probability of an observation  $x$   
and  $\theta$  refer to all of its parameters

# Maximum likelihood estimation

Let  $p(x|\theta)$  be the probability of an observation  $x$   
and  $\theta$  refer to all of its parameters

Given a dataset  $x^{(1)}, \dots, x^{(N)}$  of i.i.d. observations,

# Maximum likelihood estimation

Let  $p(x|\theta)$  be the probability of an observation  $x$  and  $\theta$  refer to all of its parameters

Given a dataset  $x^{(1)}, \dots, x^{(N)}$  of i.i.d. observations, the log-likelihood function gives us a criterion for parameter estimation

$$\mathcal{L}(\theta|x^{(1:N)}) = \log \prod_{s=1}^N p(x^{(s)}|\theta)$$

# Maximum likelihood estimation

Let  $p(x|\theta)$  be the probability of an observation  $x$  and  $\theta$  refer to all of its parameters

Given a dataset  $x^{(1)}, \dots, x^{(N)}$  of i.i.d. observations, the log-likelihood function gives us a criterion for parameter estimation

$$\begin{aligned}\mathcal{L}(\theta|x^{(1:N)}) &= \log \prod_{s=1}^N p(x^{(s)}|\theta) \\ &= \sum_{s=1}^N \log p(x^{(s)}|\theta)\end{aligned}$$

# MLE via gradient-based optimisation

If the log-likelihood is **differentiable** and **tractable** then backpropagation can give us the gradient

$$\nabla_{\theta} \mathcal{L}(\theta | x^{(1:N)}) = \nabla_{\theta} \sum_{s=1}^N \log p(x^{(s)} | \theta)$$

# MLE via gradient-based optimisation

If the log-likelihood is **differentiable** and **tractable** then backpropagation can give us the gradient

$$\nabla_{\theta} \mathcal{L}(\theta | x^{(1:N)}) = \nabla_{\theta} \sum_{s=1}^N \log p(x^{(s)} | \theta) = \sum_{s=1}^N \nabla_{\theta} \log p(x^{(s)} | \theta)$$

# MLE via gradient-based optimisation

If the log-likelihood is **differentiable** and **tractable** then backpropagation can give us the gradient

$$\nabla_{\theta} \mathcal{L}(\theta | x^{(1:N)}) = \nabla_{\theta} \sum_{s=1}^N \log p(x^{(s)} | \theta) = \sum_{s=1}^N \nabla_{\theta} \log p(x^{(s)} | \theta)$$

and we can update  $\theta$  in the direction

$$\gamma \nabla_{\theta} \mathcal{L}(\theta | x^{(1:N)})$$

to attain a local maximum of the likelihood function



# Big Data

For large  $N$ , computing the gradient is inconvenient

$$\nabla_{\theta} \mathcal{L}(\theta | x^{(1:N)}) = \underbrace{\sum_{s=1}^N \nabla_{\theta} \log p(x^{(s)} | \theta)}_{\text{too many terms}}$$

# Big Data

For large  $N$ , computing the gradient is inconvenient

$$\begin{aligned}\nabla_{\theta} \mathcal{L}(\theta | x^{(1:N)}) &= \underbrace{\sum_{s=1}^N \nabla_{\theta} \log p(x^{(s)} | \theta)}_{\text{too many terms}} \\ &= \sum_{s=1}^N \frac{1}{N} N \nabla_{\theta} \log p(x^{(s)} | \theta)\end{aligned}$$

# Big Data

For large  $N$ , computing the gradient is inconvenient

$$\begin{aligned}
 \nabla_{\theta} \mathcal{L}(\theta | x^{(1:N)}) &= \underbrace{\sum_{s=1}^N \nabla_{\theta} \log p(x^{(s)} | \theta)}_{\text{too many terms}} \\
 &= \sum_{s=1}^N \frac{1}{N} N \nabla_{\theta} \log p(x^{(s)} | \theta) \\
 &= \sum_{s=1}^N \mathcal{U}(s|1/N) N \nabla_{\theta} \log p(x^{(s)} | \theta)
 \end{aligned}$$

# Big Data

For large  $N$ , computing the gradient is inconvenient

$$\begin{aligned}
 \nabla_{\theta} \mathcal{L}(\theta | x^{(1:N)}) &= \underbrace{\sum_{s=1}^N \nabla_{\theta} \log p(x^{(s)} | \theta)}_{\text{too many terms}} \\
 &= \sum_{s=1}^N \frac{1}{N} N \nabla_{\theta} \log p(x^{(s)} | \theta) \\
 &= \sum_{s=1}^N \mathcal{U}(s|1/N) N \nabla_{\theta} \log p(x^{(s)} | \theta) \\
 &= \mathbb{E}_{S \sim \mathcal{U}(1/N)} [N \nabla_{\theta} \log p(x^{(S)} | \theta)]
 \end{aligned}$$

$S$  selects data points uniformly at random

# Stochastic optimisation

For large  $N$ , we can use a gradient estimate

$$\nabla_{\theta} \mathcal{L}(\theta | x^{(1:N)}) = \underbrace{\mathbb{E}_{S \sim \mathcal{U}(1/N)} [N \nabla_{\theta} \log p(x^{(S)} | \theta)]}_{\text{expected gradient : )}}$$

# Stochastic optimisation

For large  $N$ , we can use a gradient estimate

$$\nabla_{\theta} \mathcal{L}(\theta | x^{(1:N)}) = \underbrace{\mathbb{E}_{S \sim \mathcal{U}(1/N)} [N \nabla_{\theta} \log p(x^{(S)} | \theta)]}_{\text{expected gradient : )}}$$

$$\stackrel{\text{MC}}{\approx} \frac{1}{M} \sum_{m=1}^M N \nabla_{\theta} \log p(x^{(s_i)} | \theta)$$

$$S_i \sim \mathcal{U}(1/N)$$

# Stochastic optimisation

For large  $N$ , we can use a gradient estimate

$$\nabla_{\theta} \mathcal{L}(\theta | x^{(1:N)}) = \underbrace{\mathbb{E}_{S \sim \mathcal{U}(1/N)} [N \nabla_{\theta} \log p(x^{(S)} | \theta)]}_{\text{expected gradient : )}}$$

$$\stackrel{\text{MC}}{\approx} \frac{1}{M} \sum_{m=1}^M N \nabla_{\theta} \log p(x^{(s_i)} | \theta)$$

$$S_i \sim \mathcal{U}(1/N)$$

and take a step in the direction

$$\gamma \frac{N}{M} \underbrace{\nabla_{\theta} \mathcal{L}(\theta | x^{(s_1:s_M)})}_{\text{stochastic gradient}}$$

where  $x^{(s_1:s_M)}$  is a random mini-batch of size  $M$

# DL in NLP recipe

Maximum likelihood estimation

- ▶ tells you which **loss** to optimise (i.e. negative log-likelihood)

Automatic differentiation (*backprop*)

- ▶ “give me a tractable forward pass and I will give you **gradients**”

Stochastic optimisation powered by backprop

- ▶ general purpose gradient-based optimisers



# Tractability is central

Likelihood gives us a differentiable objective to optimise for

- ▶ but we need to stick with **tractable** likelihood functions

# When do we have intractable likelihood?

**Unsupervised problems** contain unobserved random variables

$$p(x, z|\theta) = \overbrace{p(z)}^{\text{prior}} \underbrace{p(x|z, \theta)}_{\text{observation model}}$$

# When do we have intractable likelihood?

**Unsupervised problems** contain unobserved random variables

$$p(x, z|\theta) = \overbrace{p(z)}^{\text{prior}} \underbrace{p(x|z, \theta)}_{\text{observation model}}$$

thus assessing the marginal likelihood requires  
**marginalisation of latent variables**

$$p(x|\theta) = \int p(x, z|\theta) \, dz = \int p(z)p(x|z, \theta) \, dz$$

# Examples of latent variable models

Discrete latent variable, continuous observation

$$p(x|\theta) = \sum_{c=1}^K \underbrace{\text{Cat}(c|\pi_1, \dots, \pi_K)}_{\text{too many forward passes}} \underbrace{\mathcal{N}(x|\mu_\theta(c), \sigma_\theta(c)^2)}_{\text{forward pass}}$$

# Examples of latent variable models

Discrete latent variable, continuous observation

$$p(x|\theta) = \underbrace{\sum_{c=1}^K \text{Cat}(c|\pi_1, \dots, \pi_K)}_{\text{too many forward passes}} \underbrace{\mathcal{N}(x|\mu_\theta(c), \sigma_\theta(c)^2)}_{\text{forward pass}}$$

Continuous latent variable, discrete observation

$$p(x|\theta) = \underbrace{\int \mathcal{N}(z|0, I) \text{Cat}(x|\pi_\theta(z)) \, dz}_{\text{infinitely many forward passes}}$$

forward pass

# Gradient

Exact gradient is intractable

$$\nabla_{\theta} \log p(x|\theta)$$

# Gradient

Exact gradient is intractable

$$\nabla_{\theta} \log p(x|\theta) = \nabla_{\theta} \log \underbrace{\int p(x, z|\theta) \mathrm{d}z}_{\text{marginal}}$$

# Gradient

Exact gradient is intractable

$$\begin{aligned}\nabla_{\theta} \log p(x|\theta) &= \nabla_{\theta} \log \underbrace{\int p(x, z|\theta) \mathrm{d}z}_{\text{marginal}} \\ &= \underbrace{\frac{1}{\int p(x, z|\theta) \mathrm{d}z} \int \nabla_{\theta} p(x, z|\theta) \mathrm{d}z}_{\text{chain rule}}\end{aligned}$$



# Gradient

Exact gradient is intractable

$$\begin{aligned}
 \nabla_{\theta} \log p(x|\theta) &= \nabla_{\theta} \log \underbrace{\int p(x, z|\theta) \, dz}_{\text{marginal}} \\
 &= \underbrace{\frac{1}{\int p(x, z|\theta) \, dz} \int \nabla_{\theta} p(x, z|\theta) \, dz}_{\text{chain rule}} \\
 &= \frac{1}{p(x|\theta)} \int \underbrace{p(x, z|\theta) \nabla_{\theta} \log p(x, z|\theta)}_{\text{log-identity for derivatives}} \, dz
 \end{aligned}$$

# Gradient

Exact gradient is intractable

$$\begin{aligned}
 \nabla_{\theta} \log p(x|\theta) &= \nabla_{\theta} \log \underbrace{\int p(x, z|\theta) \, dz}_{\text{marginal}} \\
 &= \underbrace{\frac{1}{\int p(x, z|\theta) \, dz} \int \nabla_{\theta} p(x, z|\theta) \, dz}_{\text{chain rule}} \\
 &= \frac{1}{p(x|\theta)} \int \underbrace{p(x, z|\theta) \nabla_{\theta} \log p(x, z|\theta)}_{\text{log-identity for derivatives}} \, dz \\
 &= \int p(z|x, \theta) \nabla_{\theta} \log p(x, z|\theta) \, dz
 \end{aligned}$$

# Gradient

Exact gradient is intractable

$$\begin{aligned}
 \nabla_{\theta} \log p(x|\theta) &= \nabla_{\theta} \log \underbrace{\int p(x, z|\theta) \, dz}_{\text{marginal}} \\
 &= \underbrace{\frac{1}{\int p(x, z|\theta) \, dz} \int \nabla_{\theta} p(x, z|\theta) \, dz}_{\text{chain rule}} \\
 &= \frac{1}{p(x|\theta)} \int \underbrace{p(x, z|\theta) \nabla_{\theta} \log p(x, z|\theta)}_{\text{log-identity for derivatives}} \, dz \\
 &= \int p(z|x, \theta) \nabla_{\theta} \log p(x, z|\theta) \, dz \\
 &= \mathbb{E}_{p(z|x, \theta)} [\nabla_{\theta} \log p(x, Z|\theta)]
 \end{aligned}$$

# But why latent variable modelling?

Some reasons

- ▶ organise a massive collection of data  
e.g. LDA

# But why latent variable modelling?

Some reasons

- ▶ organise a massive collection of data  
e.g. LDA
- ▶ learn from unlabelled data  
e.g. semi-supervised learning

# But why latent variable modelling?

Some reasons

- ▶ organise a massive collection of data  
e.g. LDA
- ▶ learn from unlabelled data  
e.g. semi-supervised learning
- ▶ induce discrete representations  
e.g. parse trees, dependency graphs,  
permutations, alignments

# But why latent variable modelling?

Some reasons

- ▶ organise a massive collection of data  
e.g. LDA
- ▶ learn from unlabelled data  
e.g. semi-supervised learning
- ▶ induce discrete representations  
e.g. parse trees, dependency graphs,  
permutations, alignments
- ▶ uncertainty quantification  
e.g. Bayesian NNs

# Deep Generative Models

Probabilistic models parametrised by neural networks



# Deep Generative Models

Probabilistic models parametrised by neural networks

- ▶ explicit modelling assumptions  
one of the reasons why there's so much interest

# Deep Generative Models

Probabilistic models parametrised by neural networks

- ▶ explicit modelling assumptions  
one of the reasons why there's so much interest
- ▶ but requires efficient inference

# Deep Generative Models

Probabilistic models parametrised by neural networks

- ▶ explicit modelling assumptions  
one of the reasons why there's so much interest
- ▶ but requires efficient inference  
which is the reason why we are here today