Introduction

Bryan Eikema and Wilker Aziz

VI Tutorial @ University of Alicante
https://vitutorial.github.io/tour/ua2020

This tutorial is about

probabilistic models *parameterised* by **neural networks**

- in this class we will look into what this means, why this is interesting, and when this is challenging

1. Probabilistic Models
2. Supervised Models Powered by NNs
3. Latent Variable Models Powered by NNs

What is a probabilistic model?

A probabilistic model predicts possible outcomes of an experiment.

Most modern machine learning models are probabilistic.

Two Machine Learning Paradigms

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, . . .

What are the benefits of probabilistic models?

Probabilistic models allows to incorporate assumptions through

- the choice of distribution
- the way that distributions uses side information
- stipulate unobserved data and their properties

They return a distribution over outcomes

Other benefits

- They can generate data
- They allow to model unobserved data
- They can be more compact
- They can provide explanation and can suggest improvements
- They can inform decision makers

Deep Generative Models

Naturally, one would like to combine the advantages of probabilistic models and neural networks. So why not have a neural net with latent variables?

Deep Generative Models

Naturally, one would like to combine the advantages of probabilistic models and neural networks. So why not have a neural net with latent variables?

Short answer: *backpropagation breaks!*

Why are we here today?

Because we want to combine the advantages of probabilistic models and neural networks to potentially

- overcome lack of supervision
- learn from partial supervision
- learn from less data
- shape the way models reason about data

and much more!

What are you getting out of this today?

As we progress we will

- develop a shared vocabulary to talk about probabilistic models powered by NNs
- derive crucial results step by step
- connect concepts and implementation

What are you getting out of this today?

As we progress we will

- develop a shared vocabulary to talk about probabilistic models powered by NNs
- derive crucial results step by step
- connect concepts and implementation

Goal

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

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.

$$X \sim \mathrm{Cat}(\theta_1, \dots, \theta_K) \quad \text{or} \quad X \sim \mathcal{N}(\theta_\mu, \theta_\sigma^2)$$

Supervised problems

We have data $x^{(1)}, \ldots, x^{(N)}$ 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.

$$X \sim \mathrm{Cat}(\theta_1, \ldots, \theta_K) \quad \text{or} \quad X \sim \mathcal{N}(\theta_\mu, \theta_\sigma^2)$$

and estimate parameters θ that assign maximum likelihood $p(x^{(1)}, \ldots, x^{(N)}|\theta)$ to observations

Supervised NN models

Let y be all side information available
e.g. deterministic *inputs/features/predictors*

Have neural networks predict parameters of our probabilistic model

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

and proceed to estimate parameters θ of the NNs

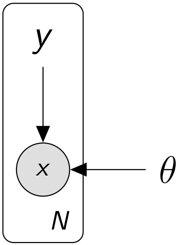
Graphical model

Random variables

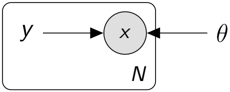
- observed data
  $x^{(1)}, \ldots, x^{(N)}$

Deterministic variables

- inputs or predictors
  $y^{(1)}, \ldots, y^{(N)}$
- model parameters θ



Multiple problems, same language



(Conditional) Density estimation

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

Task-driven feature extraction

Often our side information is itself some high dimensional object

- y is a sentence and x a tree
- y is the source sentence and x is the target
- y 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 a statistical point of view, NNs do not generate data

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

NN as efficient parametrisation

From a statistical point of view, NNs do not generate data

- they parametrise distributions that *by assumption* generate 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. argmax, beam search

Maximum likelihood estimation

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

Maximum likelihood estimation

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

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

Maximum likelihood estimation

Let $p(x|\theta)$ be the probability of an observation x and θ 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

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

Maximum likelihood estimation

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

Given a dataset $x^{(1)}, \ldots, x^{(N)}$ 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 θ refer to all of its parameters

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

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

MLE via gradient-based optimisation

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

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

MLE via gradient-based optimisation

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

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

MLE via gradient-based optimisation

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

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

MLE via gradient-based optimisation

If the log-likelihood is **differentiable** and **tractable** then backpropagation gives 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 θ 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

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

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

Big Data

For large N, computing the gradient is inconvenient

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

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

$$= \sum_{s=1}^{N} \mathcal{U}(s|{}^1/_N) N \nabla_\theta \log p(x^{(s)}|\theta)$$

Big Data

For large N, computing the gradient is inconvenient

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

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

$$= \sum_{s=1}^{N} \mathcal{U}(s | {}^{1}/_{N}) N \nabla_\theta \log p(x^{(s)} | \theta)$$

$$= \mathbb{E}_{S \sim \mathcal{U}({}^{1}/_{N})} \left[N \nabla_\theta \log p(x^{(S)} | \theta) \right]$$

S selects data points uniformly at random

Stochastic optimisation

For large N, we can use a gradient estimate

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

Stochastic optimisation

For large N, we can use a gradient estimate

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

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

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

Stochastic optimisation

For large N, we can use a gradient estimate

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

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

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

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

Constraints

Differentiability

- intermediate representations must be continuous
- activations must be differentiable

Tractability

- the likelihood function must be evaluated exactly, thus it's required to be tractable

When do we have intractable likelihood?

Latent variable models contain unobserved random variables

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

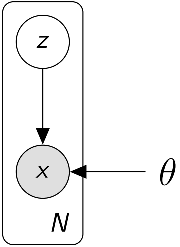
thus assessing the marginal likelihood requires marginalisation of latent variables

$$p(x|\theta) = \int p(x, z|\theta)\,\mathrm{d}z$$

Latent variable model

Latent random variables

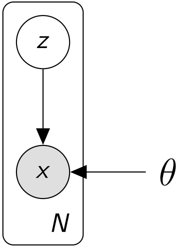
- unobserved
- or unobservable



Latent variable model

Latent random variables

- unobserved
- or unobservable



A joint distribution over data and unknowns

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

Examples of latent variable models

Discrete latent variable, continuous observation

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

Examples of latent variable models

Discrete latent variable, continuous observation

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

Continuous latent variable, discrete observation

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

Intractable gradient

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

Intractable gradient

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

Intractable gradient

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

Intractable gradient

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

Intractable gradient

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

Approximations

Can we approximate the gradient?

- some approximations introduce bias
- others break differentiability
- some approximations suffer from both problems

We prefer unbiased approximations. A large bulk of DGM research goes to efficient unbiased gradient estimation.

Gradient estimates?

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

Gradient estimates?

$$\nabla_\theta \log p(x|\theta) = \int p(z|x,\theta) \nabla_\theta \log p(x,z|\theta)\, \mathrm{d}z$$
$$= \mathbb{E}_{p(z|x,\theta)} \left[\nabla_\theta \log p(x,z|\theta) \right]$$

Gradient estimates?

$$
\begin{aligned}
\nabla_\theta \log p(x|\theta) &= \int p(z|x,\theta) \nabla_\theta \log p(x,z|\theta)\, \mathrm{d}z \\
&= \mathbb{E}_{p(z|x,\theta)} \left[\nabla_\theta \log p(x,z|\theta) \right] \\
&\overset{\text{MC}}{\approx} \frac{1}{K} \sum_{k=1}^{K} \nabla_\theta \log p(x, z_k|\theta) \quad \text{where } z_k \sim p(z|x,\theta)
\end{aligned}
$$

Gradient estimates?

$$
\begin{aligned}
\nabla_\theta \log p(x|\theta) &= \int p(z|x,\theta) \nabla_\theta \log p(x,z|\theta)\,\mathrm{d}z \\
&= \mathbb{E}_{p(z|x,\theta)}\left[\nabla_\theta \log p(x,z|\theta)\right] \\
&\overset{\text{MC}}{\approx} \frac{1}{K}\sum_{k=1}^{K} \nabla_\theta \log p(x,z_k|\theta) \quad \text{where } z_k \sim p(z|x,\theta)
\end{aligned}
$$

But the posterior is not available!

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

Gradient estimates?

$$
\begin{aligned}
\nabla_\theta \log p(x|\theta) &= \int p(z|x,\theta) \nabla_\theta \log p(x,z|\theta)\, \mathrm{d}z \\
&= \mathbb{E}_{p(z|x,\theta)} \left[\nabla_\theta \log p(x,z|\theta)\right] \\
&\overset{\text{MC}}{\approx} \frac{1}{K} \sum_{k=1}^{K} \nabla_\theta \log p(x, z_k|\theta) \quad \text{where } z_k \sim p(z|x,\theta)
\end{aligned}
$$

But the posterior is not available!

$$
p(z|x,\theta) = \frac{p(x,z|\theta)}{p(x|\theta)}
$$

But why latent variable modelling?

Some reasons

- better handle on statistical assumptions
  e.g. breaking marginal independence

But why latent variable modelling?

Some reasons

- better handle on statistical assumptions
  e.g. breaking marginal independence
- organise a massive collection of data
  e.g. LDA

But why latent variable modelling?

Some reasons

- better handle on statistical assumptions
  e.g. breaking marginal independence
- 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

- better handle on statistical assumptions
  e.g. breaking marginal independence
- 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, alignments

But why latent variable modelling?

Some reasons

- better handle on statistical assumptions
  e.g. breaking marginal independence
- 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, alignments
- uncertainty quantification
  e.g. Bayesian NNs

Examples: Lexical alignment

Generate a word x_i in L1 from a word y_{a_i} in L2

Examples: Lexical alignment

Generate a word x_i in L1 from a word y_{a_i} in L2

$$p(x|y,\theta) \stackrel{\text{ind}}{=} \prod_{i=1}^{|x|} \sum_{a_i=1}^{|y|} \mathcal{U}(a_i|^1/_{|y|}) p(x_i|y_{a_i})$$

Examples: Lexical alignment

Generate a word x_i in L1 from a word y_{a_i} in L2

$$p(x|y,\theta) \stackrel{\text{ind}}{=} \prod_{i=1}^{|x|} \sum_{a_i=1}^{|y|} \mathcal{U}(a_i|{}^{1}/_{|y|}) p(x_i|y_{a_i})$$

a mixture model whose mixture components are labelled by words

marginalisation $O(|x||y|)$

Examples: Rationale extraction

Sentiment analysis based on a subset of the input

Examples: Rationale extraction

Sentiment analysis based on a subset of the input

$$p(x|y,\theta) = \sum_{f_1=0}^{1} \cdots \sum_{f_{|y|}=0}^{1} \left(\prod_{i=1}^{|y|} \text{Bernoulli}(f_i|\theta_{y_i}) \right) p(x|f,y)$$

where $p(x|f,y)$ conditions on y_i iff $f_i = 1$.

Examples: Rationale extraction

Sentiment analysis based on a subset of the input

$$p(x|y,\theta) = \sum_{f_1=0}^{1} \cdots \sum_{f_{|y|}=0}^{1} \left(\prod_{i=1}^{|y|} \text{Bernoulli}(f_i|\theta_{y_i}) \right) p(x|f,y)$$

where $p(x|f,y)$ conditions on y_i iff $f_i = 1$.

A factor model whose factors are labelled by words

marginalisation $O(2^{|y|})$

Examples: Language modelling

A (deterministic) RNNLM aways produces the same conditional $p(x_i|x_{<i}, \theta)$ for a given prefix.

Examples: Language modelling

A (deterministic) RNNLM aways produces the same conditional $p(x_i|x_{<i}, \theta)$ for a given prefix. Isn't it reasonable to expect the conditional to depend on what we are talking about?

Examples: Language modelling

A (deterministic) RNNLM aways produces the same conditional $p(x_i|x_{<i}, \theta)$ for a given prefix. Isn't it reasonable to expect the conditional to depend on what we are talking about? e.g. *Rio de Janeiro ...*

- history: *once was the Brazilian capital*
- tourism: *offers some of Brazil's most iconic landscapes*
- news: *recently hosted the world cup final*

Examples: Language modelling

A (deterministic) RNNLM aways produces the same conditional $p(x_i|x_{<i}, \theta)$ for a given prefix. Isn't it reasonable to expect the conditional to depend on what we are talking about? e.g. *Rio de Janeiro . . .*

- history: *once was the Brazilian capital*
- tourism: *offers some of Brazil's most iconic landscapes*
- news: *recently hosted the world cup final*

$$p(x|\theta) = \int \mathcal{N}(z|0, I) \prod_{i=1}^{|x|} p(x_i|z, x_{<i}, \theta)\mathrm{d}z$$

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