WELCOME TO THE

UCL x DeepMind lecture series

In this lecture series, leading research scientists from leading AI research lab, DeepMind, will give 12 lectures on an exciting selection of topics in Deep Learning, ranging from the fundamentals of training neural networks via advanced ideas around memory, attention, and generative modelling to the important topic of responsible innovation.

Please join us for a deep dive lecture series into Deep Learning!

#UCLxDeepMind

TODAY'S SPEAKER

Andriy Mnih

Andriy Mnih is a research scientist at DeepMind. He works on generative modelling, representation learning, variational inference, and gradient estimation for stochastic computation graphs. He did his PhD on learning representations of discrete data at the University of Toronto, where he was advised by Geoff Hinton. Prior to joining DeepMind, Andriy was a post-doctoral researcher at the Gatsby Unit, University College London, working with Yee Whye Teh.

TODAY'S LECTURE

Modern Latent Variable Models and Variational Inference

Latent variable models provide a powerful and flexible framework for generative modelling. After introducing this framework along with the concept of inference, which is central to it, this lecture will focus on two types of modern latent variable models: invertible models and intractable models. Special emphasis will be placed on understanding variational inference as a key to training intractable latent variable models.

Modern Latent Variable Models and Variational Inference

Andriy Mnih

UCL x DeepMind Lectures

Lecture Outline

DeepMind

1 Generative Modelling

What are generative models?

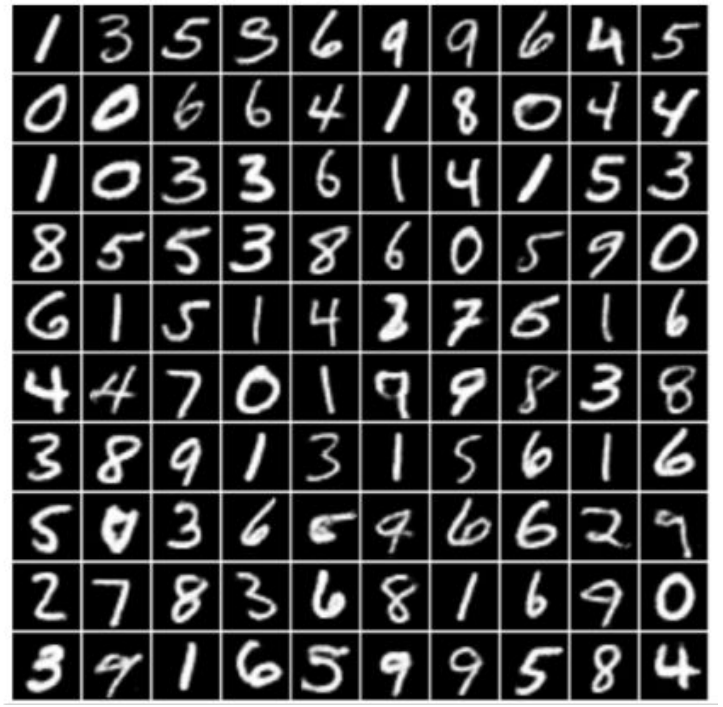
- Generative models are probabilistic models of high-dimensional data
    - Describe the probabilistic process of generating an observation
    - The emphasis is on capturing the dependence between the dimensions
    - Provide a way of generating new datapoints
- Historically, generative modelling was considered to be a subfield of unsupervised learning (i.e. learning with no labels).
    - Conditional generative models make the boundary between supervised and unsupervised learning blurry.
- Generative models can be used for essentially any kind of data:
    - Text, images, audio, biological sequences, etc.

Uses of generative models

- Density estimation and outlier detection
- Data compression
- Mapping from one domain to another
  - Language translation, text-to-speech
- Planning in model-based reinforcement learning
- Representation learning
- Understanding the data

Progress in generative models

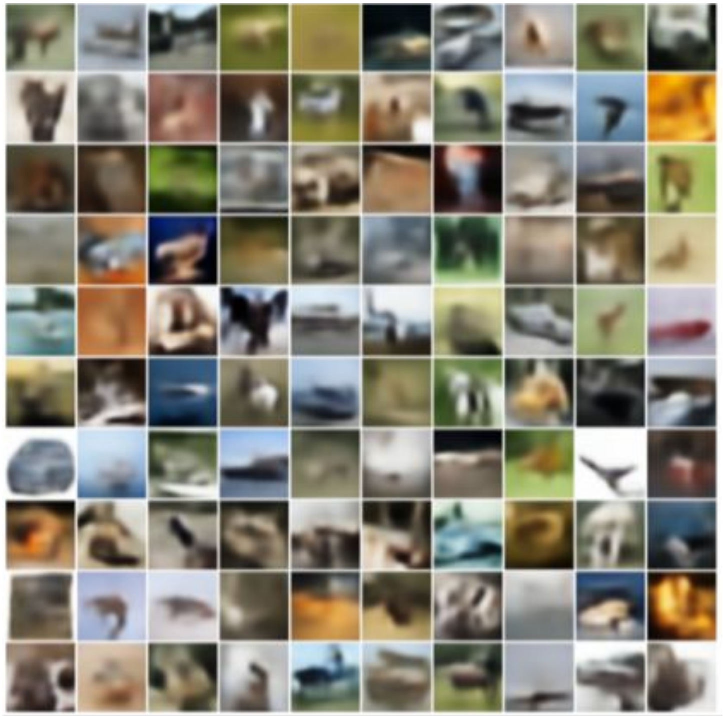
Samples from 2014:



Deep AutoRegressive Networks, Gregor et al.

Progress in generative models

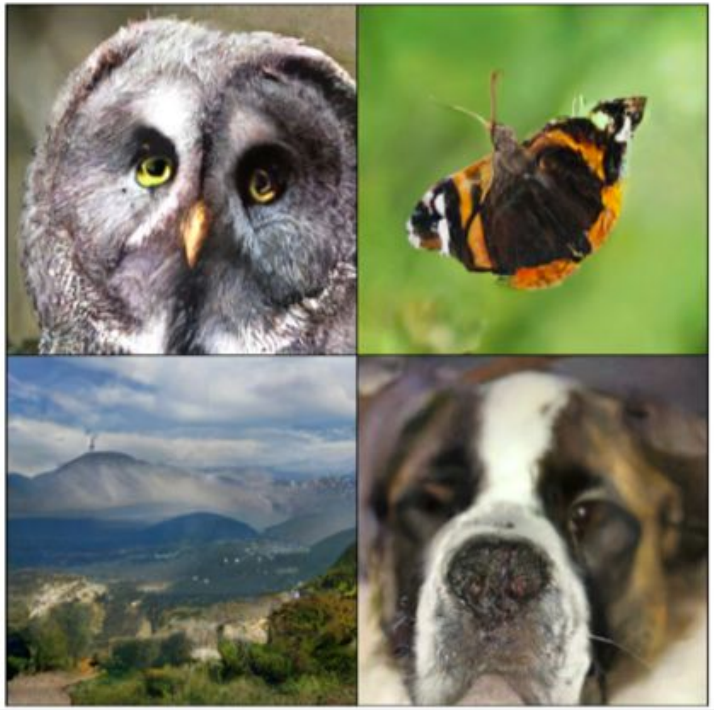
Samples from 2015:



DRAW: A Recurrent Neural Network For Image Generation, Gregor et al.

Progress in generative models

Samples from 2019:



Hierarchical Autoregressive Image Models with Auxiliary Decoders, De Fauw et al.

Types of generative models

Generative models used in deep learning:

- **Autoregressive models**
  - RNN & Transformer language models, NADE, PixelCNN, WaveNet
- **Latent variable models**
  - Tractable: e.g. Invertible / flow-based models (RealNVP, Glow, etc.)
  - Intractable: e.g. Variational Autoencoders
- **Implicit models**
  - Generative Adversarial Networks (GANs) and variants

Autoregressive models

- Model the 1-dimensional conditional distributions instead of modelling the joint distribution directly:
    - Based on the chain rule: $p(\mathbf{x}) = \prod_{d=1}^{D} p(x_d | x_1, .., x_{d-1})$
    - Trained with maximum likelihood
- Pros
    - 1-dimensional distributions are easy to model
    - Simple and efficient training
    - No sampling at training time
- Cons
    - Slow, sequential generation — one dimension at a time
    - Usually much better at modelling local structure than global structure

Latent variable models

- Specify the generative process in terms of unobserved/latent variables and the transformation that maps them to the observation.
    - Trained with maximum likelihood (usually with some approximations)
- Pros
    - Powerful and well-understood framework
    - Easy to incorporate prior knowledge / structure into models
        - Well-suited for representation learning / interpretability
    - Fast generation
- Cons
    - Conceptually more complex than fully observed models
    - Need to use approximate inference or restricted models

Generative Adversarial Networks

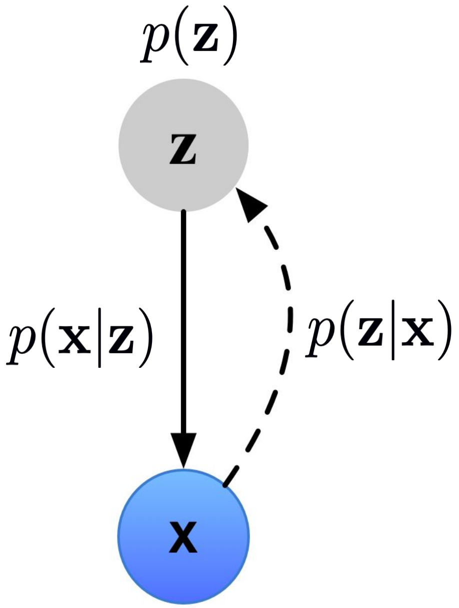
- Model: A neural net that maps noise vectors to observations
- Training: use the learning signal from a classifier trained to discriminate between samples from the model and the training data
- Pros
  - Can generate very realistic images
  - Conceptually simple implementation
  - Fast generation
- Cons
  - Cannot be used to compute probability of observations
  - "Mode collapse": Models ignore regions of the data distribution
  - Training can be unstable and requires many tricks to work well

DeepMind

2 Latent Variable Models & Inference

Latent variable models

- A latent variable model (LVM) defines a distribution over observations $\mathbf{x}$ by using a (vector) latent variable $\mathbf{z}$ and specifying:
    - The **prior** distribution $p(\mathbf{z})$ for the latent variable
    - The **likelihood** $p(\mathbf{x}|\mathbf{z})$ that connects the latent variable to the observation
- The prior and the likelihood define the **joint** distribution $p(\mathbf{x}, \mathbf{z}) = p(\mathbf{x}|\mathbf{z})p(\mathbf{z})$
- We will be interested in computing the **marginal likelihood** $p(\mathbf{x})$ and the **posterior** distribution $p(\mathbf{z}|\mathbf{x})$.

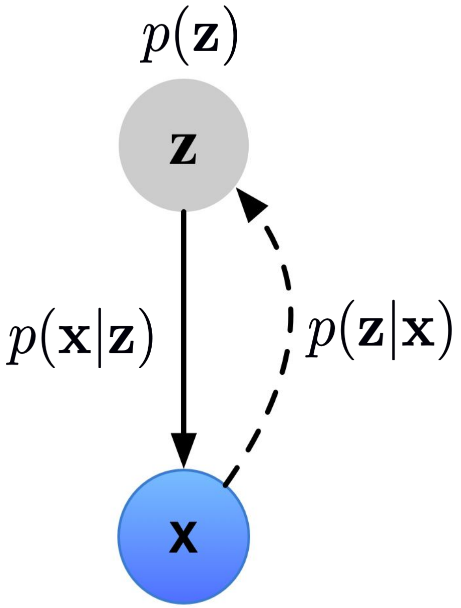


Latent variable models

- We can think of the latent variable value as an **explanation** for the observation.
- To **generate** an observation from the model, we sample as follows:

$$\mathbf{z} \sim p(\mathbf{z})$$
$$\mathbf{x} \sim p(\mathbf{x}|\mathbf{z})$$

- Much of this lecture will be concerned with the inverse process, going from observations to latent values, which is called **inference**.



Inference

- **Inference** refers to computing the posterior distribution for the given observation:

$$p(\mathbf{z}|\mathbf{x}) = \frac{p(\mathbf{x}, \mathbf{z})}{p(\mathbf{x})} = \frac{p(\mathbf{x}, \mathbf{z})}{\int p(\mathbf{x}, \mathbf{z}) d\mathbf{z}}$$

- This requires solving the important sub-problem of computing the **marginal likelihood** of the observation:

$$p(\mathbf{x}) = \int p(\mathbf{x}, \mathbf{z}) d\mathbf{z}$$

Inference is the inverse of generation

- We can generate $(\mathbf{x}, \mathbf{z})$ pairs from the model in two ways:

$$\mathbf{z} \sim p(\mathbf{z}) \qquad\qquad \mathbf{x} \sim p(\mathbf{x})$$

$$\mathbf{x} \sim p(\mathbf{x}|\mathbf{z}) \qquad\qquad \mathbf{z} \sim p(\mathbf{z}|\mathbf{x})$$

- Since $p(\mathbf{x}|\mathbf{z})p(\mathbf{z}) = p(\mathbf{x}, \mathbf{z}) = p(\mathbf{z}|\mathbf{x})p(\mathbf{x})$, the joint distribution of these pairs is **exactly the same**, no matter how they were generated.
- We can think of inference / computing $p(\mathbf{z}|\mathbf{x})$ as inverting $p(\mathbf{x}|\mathbf{z})$ probabilistically.

Why is inference important?

- **Explaining** the observation: Inferring the posterior distribution for a datapoint allows us to determine which latent configurations could have plausibly generated it.

- **Learning**: training latent variable models requires performing inference in the inner loop, as we will see later.

Inference for a mixture of Gaussians

- Model specification:

$$P(z) = \text{Categorical}(\pi_1, \ldots, \pi_K)$$
$$p(x|z) = \mathcal{N}(\mu_z, \sigma_z^2)$$

- Marginal likelihood:

$$p(x) = \sum_{z=1}^{K} P(z)p(x|z) = \sum_{z=1}^{K} \pi_z \mathcal{N}(\mu_z, \sigma_z^2)$$

- Posterior distribution:

$$P(z|x) = \frac{p(x|z)P(z)}{p(x)} = \frac{\pi_z \mathcal{N}(\mu_z, \sigma_z^2)}{\sum_{i=1}^{K} \pi_i \mathcal{N}(\mu_i, \sigma_i^2)}$$

Maximum likelihood learning

- **Maximum Likelihood** is the dominant estimation principle for probabilistic models.
- We look for the parameter values that maximize the probability the training set under the model:

$$\theta^{ML} = \arg\max_{\theta} \sum_{i=1}^{N} \log p_\theta(\mathbf{x}_i)$$

- For latent variable models, this optimization problem does not have a closed-form solution, so iterative algorithms are used.

The gradient of the marginal log-likelihood

Let's compute the gradient of the marginal log-likelihood for a single datapoint:

$$\nabla_\theta \log p_\theta(\mathbf{x}) = \frac{\nabla_\theta p_\theta(\mathbf{x})}{p_\theta(\mathbf{x})} = \frac{\int \nabla_\theta p_\theta(\mathbf{x}, \mathbf{z}) d\mathbf{z}}{p_\theta(\mathbf{x})}$$

$$= \frac{\int p_\theta(\mathbf{x}, \mathbf{z}) \nabla_\theta \log p_\theta(\mathbf{x}, \mathbf{z}) d\mathbf{z}}{p_\theta(\mathbf{x})}$$

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

Using the identity

$$\nabla_\theta \log p_\theta(\mathbf{x}, \mathbf{z}) = \frac{\nabla_\theta p_\theta(\mathbf{x}, \mathbf{z})}{p_\theta(\mathbf{x}, \mathbf{z})}$$

- We need to compute the posterior distribution to compute the gradient!
- **Inference performs credit assignment over latent configurations.**

Exact inference is hard

- As a rule, exact inference is hard.
  - Inference for continuous latent variables involves computing high-dimensional integrals of complicated functions.
  - Inference for discrete latent variables involves summing over exponentially many latent configurations.

- Exceptions: Some models with tractable inference
  - Mixture models
  - Linear-Gaussian models
  - Invertible models

Avoiding intractable inference

Strategies for avoiding intractable inference:

1. Designing models for which inference is tractable
   - Simpler training but less expressive models
2. Using approximate inference
   - More involved, but also more flexible — can use more expressive models

DeepMind

3 Invertible Models & Exact Inference

Invertible models

- **Invertible models** (also known as **normalizing flows**) are a class of tractable yet powerful generative models.
- Key idea: approximate the data distribution by transforming the prior distribution using an invertible function.
    - This can be done jointly for all dimensions or autoregressively, one dimension at a time.
- Inference and maximum likelihood learning are tractable in these models.

Invertible models

- Invertible models generate observations by applying an invertible and differentiable transformation $f_\theta(\mathbf{z})$ to samples from the prior:

$$\mathbf{z} \sim p(\mathbf{z})$$
$$\mathbf{x} = f_\theta(\mathbf{z})$$

- Since $f_\theta(\mathbf{z})$ is invertible, there is a one-to-one correspondence between observations and latent configurations, and inference is easy:

$$\mathbf{z} = f_\theta^{-1}(\mathbf{x})$$

- How can we compute the marginal likelihood?

Invertible models

- We can relate the densities of $\mathbf{x}$ and $\mathbf{z}$ as follows:

$$p_\theta(\mathbf{x}) = p_\theta(\mathbf{z}) \left| \det \frac{\partial \mathbf{z}}{\partial \mathbf{x}} \right|$$

Change of variables

$$= p(f_\theta^{-1}(\mathbf{x})) \left| \det \frac{\partial f_\theta^{-1}(\mathbf{x})}{\partial \mathbf{x}} \right|$$

Since $\mathbf{z} = f_\theta^{-1}(\mathbf{x})$

- The determinant of the Jacobian takes into account the local change in volume due to applying $f_\theta(\mathbf{z})$.

- For maximum likelihood learning we need the mapping $f_\theta^{-1}(\mathbf{x})$, the determinant of its Jacobian, as well as their gradients to be efficiently computable.

Independent Component Analysis

- ICA is probably the simplest and oldest invertible model.
- It uses a factorial prior and a linear mapping (a square matrix):

$$\mathbf{z} \sim \prod_i p(\mathbf{z}_i)$$
$$\mathbf{x} = \mathbf{A}\mathbf{z}$$

- Inference is just matrix inversion: $\mathbf{z} = \mathbf{A}^{-1}\mathbf{x}$
- Inference in a trained ICA model allows us to recover independent sources that explain the data linearly.
- To recover independent sources, the prior has to be non-Gaussian.
  - Usually, heavy-tailed distributions such as Cauchy or Logistic are used

Constructing invertible models

- General strategy: construct complex models by chaining simple invertible functions.
    - Composition of invertible functions is invertible
    - Can parameterize either the forward mapping $f_\theta(\mathbf{z})$ or the backward mapping $f_\theta^{-1}(\mathbf{x})$
- Functions that can be inverted only numerically with iterative algorithms can also be an option.
- A growing list of invertible building blocks:
    - Linear layers, autoregressive / coupling layers, Sylvester flows, residual layers

Limitations of invertible models

- Invertible models are appealing because they provide a rare combination of tractability and high expressive power.
- They do have a number of limitations:
  - The latent space and data space dimensionality must be the same
  - The latent variables have to be continuous
  - Observations have to be continuous or quantized
  - Expressive models require a lot of layers, parameters, and thus memory
  - Hard to incorporate structure: lack of flexibility in model design
- Due to their tractability, invertible model are also useful as components for larger latent variable models.

DeepMind

4 Variational Inference

The appeal of intractable models

- **Do you want the wrong answer to the right question or the right answer to the wrong question**? --David Blei
- In many cases, a model is more than just a black box that generates samples or makes predictions.
  - We might want latent variables to be interpretable or related to each other in a specific way.
  - We might want to structure of the model in a particular way, e.g. based on our knowledge of the process being modelled.
- Almost all such models will be intractable and thus will require **approximate inference**.

Example: ICA variations

- We saw that the ICA model with the same number of latent dimensions as data dimensions is tractable.
- What happens if we change the model slightly?
  - Introduce an observation noise model in the likelihood
  - Use more latent dimensions than data dimensions
  - Use fewer latent dimensions than data dimensions
- Any one of these changes makes the model **intractable**.

Approximate inference

Two classes of approaches to approximate inference:

- **Markov Chain Monte Carlo**: generate samples from the exact posterior using a Markov Chain
  - Very general; exact in the limit of infinite time / computation
  - Computationally expensive; convergence is hard to diagnose
- **Variational inference**: approximate the posterior with a tractable distribution, e.g. fully factorized or autoregressive
  - Fairly efficient, as inference is reduced to optimization w.r.t. the distribution parameters
  - Cannot trade computation for accuracy easily

Variational inference

- Variational inference (VI) turns the task of finding the posterior distribution into an optimization problem.
- We approximate the exact posterior $p_\theta(\mathbf{z}|\mathbf{x})$ with a **variational posterior** $q_\phi(\mathbf{z}|\mathbf{x})$.
- ϕ are the **variational parameters** which we will optimize over to fit the variational posterior to the exact posterior.
- We can use any distribution for $q_\phi(\mathbf{z}|\mathbf{x})$ as long as
  1. We can sample from it
  2. We can compute $\log q_\phi(\mathbf{z}|\mathbf{x})$ and its gradient w.r.t ϕ.
- A fully factorized distribution $q_\phi(\mathbf{z}|\mathbf{x}) = \prod_i q_\phi(\mathbf{z}_i|\mathbf{x})$ is often used.

Training with variational inference

- What do we use as the training objective if the marginal log-likelihood is intractable?
- The variational posterior induces a **variational lower bound** $\mathcal{L}_{\theta,\phi}(\mathbf{x})$ on the marginal log-likelihood.
- We train a model with VI by maximizing $\mathcal{L}_{\theta,\phi}(\mathbf{x})$ with respect to both the model parameters θ and the variational parameters ϕ.

Bounding the marginal log-likelihood

- For any density $q_\phi(\mathbf{z})$ (as long as $q_\phi(\mathbf{z}) > 0$ whenever $p(\mathbf{z}) > 0$) we have:

$$\begin{aligned}\log p_\theta(\mathbf{x}) &= \log \int p_\theta(\mathbf{x}, \mathbf{z}) d\mathbf{z} \\ &= \log \int q_\phi(\mathbf{z}) \frac{p_\theta(\mathbf{x}, \mathbf{z})}{q_\phi(\mathbf{z})} d\mathbf{z} \\ &\geq \int q_\phi(\mathbf{z}) \log \frac{p_\theta(\mathbf{x}, \mathbf{z})}{q_\phi(\mathbf{z})} d\mathbf{z} \\ &= \mathbb{E}_{q_\phi(\mathbf{z})} \left[\log \frac{p_\theta(\mathbf{x}, \mathbf{z})}{q_\phi(\mathbf{z})} \right]\end{aligned}$$

Using the Jensen's inequality:
$\log \mathbb{E}_{q_\phi(\mathbf{z})} [f(\mathbf{z})] \geq \mathbb{E}_{q_\phi(\mathbf{z})} [\log f(\mathbf{z})]$

- Therefore $\mathcal{L}_{\theta,\phi}(\mathbf{x}) = \mathbb{E}_{q_\phi(\mathbf{z})} \left[\log \frac{p_\theta(\mathbf{x}, \mathbf{z})}{q_\phi(\mathbf{z})} \right] \leq \log p_\theta(\mathbf{x})$ for any such $q_\phi(\mathbf{z})$

Variational lower bounds

- In this lecture, we will focus on the **Evidence Lower Bound (ELBO)**, which is obtained by using the variational posterior $q_\phi(\mathbf{z}|\mathbf{x})$ as $q_\phi(\mathbf{z})$:

$$\mathcal{L}_{\theta,\phi}(\mathbf{x}) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log \frac{p_\theta(\mathbf{x},\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})}\right]$$

- ELBO is the simplest and by far the most widely variational lower bound.
- Another option: The Importance-Weighted Bound (a.k.a. IWAE)
  - A multi-sample generalization of ELBO
  - Allows trading off additional computation for better approximation of the marginal likelihood

Review: Kullback–Leibler divergence

- Kullback–Leibler divergence provides a way of quantifying the difference between two distributions.
- KL divergence is defined as $\mathbb{KL}(q(\mathbf{z})||p(\mathbf{z})) = \mathbb{E}_{q(\mathbf{z})}\left[\log \frac{q(\mathbf{z})}{p(\mathbf{z})}\right]$
- KL divergence is
  - Non-negative: $\mathbb{KL}(q(\mathbf{z})||p(\mathbf{z})) \geq 0$ for any $q(\mathbf{z})$ and $p(\mathbf{z})$;
  - $\mathbb{KL}(q(\mathbf{z})||p(\mathbf{z})) = 0$ if and only if $q(\mathbf{z}) = p(\mathbf{z})$ almost everywhere;
  - Not symmetric: in general, $\mathbb{KL}(q(\mathbf{z})||p(\mathbf{z})) \neq \mathbb{KL}(p(\mathbf{z})||q(\mathbf{z}))$.

Fitting the variational posterior

- What happens when we maximize the ELBO w.r.t. the variational parameters?
- Rewriting the ELBO: $\mathcal{L}_{\theta,\phi}(\mathbf{x}) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log \frac{p_\theta(\mathbf{x})p_\theta(\mathbf{z}|\mathbf{x})}{q_\phi(\mathbf{z}|\mathbf{x})}\right]$

$$= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(\mathbf{x})\right] + \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log \frac{p_\theta(\mathbf{z}|\mathbf{x})}{q_\phi(\mathbf{z}|\mathbf{x})}\right]$$

$$= \log p_\theta(\mathbf{x}) - \mathbb{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z}|\mathbf{x}))$$

- This means maximizing $\mathcal{L}_{\theta,\phi}(\mathbf{x})$ w.r.t. ϕ is equivalent to minimizing $\mathbb{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z}|\mathbf{x}))$, as $\log p_\theta(\mathbf{x})$ does not depend on ϕ.
- $\mathbb{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z}|\mathbf{x}))$ is known as the **variational gap**, because it is the difference between $\log p_\theta(\mathbf{x})$ and the variational bound $\mathcal{L}_{\theta,\phi}(\mathbf{x})$.

Fitting the variational posterior

- Maximizing the ELBO w.r.t. ϕ fits the variational posterior to the exact posterior by minimizing $\mathbb{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z}|\mathbf{x}))$.
- This is remarkable because we cannot compute $\mathbb{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z}|\mathbf{x}))$ or even $p_\theta(\mathbf{z}|\mathbf{x})$!
- Since $\mathcal{L}_{\theta,\phi}(\mathbf{x}) = \log p_\theta(\mathbf{x}) - \mathbb{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z}|\mathbf{x}))$, the ELBO is the difference between two intractable quantities, which happens to be tractable.
- If the variational distribution family is expressive enough, the ELBO is maximized w.r.t. ϕ when the variational posterior is equal to the exact posterior (and the variational gap is zero).

Training the model

- What happens when we update the model parameters to increase the ELBO?
- The decomposition $\mathcal{L}_{\theta,\phi}(\mathbf{x}) = \log p_\theta(\mathbf{x}) - \mathbb{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z}|\mathbf{x}))$ tells us that:
  1. $\log p_\theta(\mathbf{x})$ will increase and/or
  2. The variational gap $\mathbb{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z}|\mathbf{x}))$ will decrease.
- Increasing $\log p_\theta(\mathbf{x})$ is good — this is exactly what maximum likelihood learning does.
- What about decreasing the variational gap?

Training the model

- Decreasing the variational gap by updating the **variational** parameters, makes the variational distribution closer the true posterior.
  - Improves the variational approximation without affecting the model
- Decreasing the variational gap by updating the **model** parameters, makes the exact posterior closer to the variational posterior.
  - Makes variational inference in the model more accurate
  - But it often does so by making the model fit the data less well.
- To mitigate this effect and make learning with VI as close as possible to maximizing likelihood, **we should use the most expressive variational posterior we can**.

Variational pruning

- The pressure from VI to make the exact posterior be closer to the variational posterior often results in some of the latent variables not being used by the model.
  - For these variables, the posterior will be identical to the prior.
- This effect is called **variational pruning**.
  - In the context of variational autoencoders, this is also known as **posterior collapse**.
- Variational pruning can be a good thing or a bad thing:
  - + It automatically chooses the dimensionality for the latent space.
  - – It prevents us from fitting the data arbitrarily well even if we use an arbitrarily high dimensional latent space.

Choosing the form of the variational posterior

- The default choice is a fully factorized distribution $q_\phi(\mathbf{z}|\mathbf{x}) = \prod_i q_\phi(\mathbf{z}_i|\mathbf{x})$
  - In classic VI, this is known as the **mean field** approximation.
- Several options for more expressive posteriors:
  - Mixture distributions
  - Gaussian with a non-diagonal covariance matrix
  - Autoregressive
  - Flow-based
- Trade-off: speed of training vs. quality of the variational approximation

Amortized variational inference

- The posterior distribution is **different** for each observation $\mathbf{x}$.
- In classic variational inference, we learn a different set of variational parameters ϕ for each datapoint using iterative optimization.
- **Amortized inference**: Instead performing such per-datapoint optimization, train a neural net, called the **inference network,** to predict the variational parameters from the observation.
    - ϕ in $q_\phi(\mathbf{z}|\mathbf{x})$ now refers to the inference network parameters.
    - Introduced in the context of Helmholtz Machines and popularised by Variational Autoencoders.
- The inference network in trained jointly with the model by maximizing the ELBO.

Variational vs. exact inference

- What did we gain by using variational inference?
    - Can now train and perform inference in models with intractable posteriors in an efficient and principled way.
        - Freedom in model design
    - Inference is fast compared to MCMC methods.
- What did we lose?
    - VI can make the model effectively less expressive and thus lead to suboptimal performance.

DeepMind

5 Gradient Estimation in Variational Inference

Maximizing the ELBO

- To train a model with VI, we need to maximize the ELBO

$$\mathcal{L}_{\theta,\phi}(\mathbf{x}) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log \frac{p_\theta(\mathbf{x},\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})}\right]$$

  w.r.t. the model parameters θ and variational parameters ϕ.

- How do we compute the required gradients?
- In modern variational inference, we estimate gradients using Monte Carlo sampling.
  - We trade deterministic gradient estimation of classic variational inference for much broader applicability.

Gradient w.r.t. the model parameters

- Estimating the gradient for the model parameters is easy using Monte Carlo sampling from the variational posterior:

$$\begin{aligned}\nabla_\theta \mathcal{L}_{\theta,\phi}(\mathbf{x}) &= \nabla_\theta \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log \frac{p_\theta(\mathbf{x},\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})}\right] \\ &= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\nabla_\theta \log p_\theta(\mathbf{x},\mathbf{z})\right] \\ &\approx \frac{1}{K}\sum_{k=1}^{K} \nabla_\theta \log p_\theta(\mathbf{x},\mathbf{z}^k) \text{ with } \mathbf{z}^k \sim q_\phi(\mathbf{z}|\mathbf{x})\end{aligned}$$

Since $q_\phi(\mathbf{z}|\mathbf{x})$ does not depend on θ

- We simply generate one or more samples from the variational posterior and average the resulting gradients of the log-joint.

Gradient w.r.t. the variational parameters

- Estimating the gradient w.r.t. the variational parameters is more involved.
  - Want to estimate it using samples from the variational posterior which depends on ϕ.
  - Need to take this dependence into account somehow:

$$\nabla_\phi \mathcal{L}_{\theta,\phi}(\mathbf{x}) = \nabla_\phi \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log \frac{p_\theta(\mathbf{x},\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})}\right]$$
$$\approx ???$$

- Thankfully, estimating gradients of expectations of this form is a well-studied problem.

Gradients of expectations

Two major types of gradient estimators for $\nabla_\phi \mathbb{E}_{q_\phi(\mathbf{z})}[f(\mathbf{z})]$:

- REINFORCE / likelihood-ratio estimator
  - Very general
    - Applicable to both discrete and continuous latent variables
    - Can handle non-differentiable functions $f(\mathbf{z})$
  - Tends to have relatively high variance: needs variance reduction.
- Reparameterization / pathwise estimator
  - Less general
    - Applicable only to continuous latent variables
    - Requires $f(\mathbf{z})$ to be differentiable
  - Tends to have relatively low variance

Reparameterization trick

- The **reparameterization trick** provides an effective way of computing gradients of the form $\nabla_\phi \mathbb{E}_{q_\phi(\mathbf{z})}\left[f(\mathbf{z})\right]$ for many common densities $q_\phi(\mathbf{z})$.
- We **reparameterize** a sample from $q_\phi(\mathbf{z})$ by expressing it as a function of a sample $\boldsymbol{\epsilon}$ from some fixed distribution $p(\boldsymbol{\epsilon})$:

$$\mathbf{z} = g(\boldsymbol{\epsilon}, \phi)$$

- As long as $g(\boldsymbol{\epsilon}, \phi)$ is differentiable w.r.t. ϕ,

$$\begin{aligned}\nabla_\phi \mathbb{E}_{q_\phi(\mathbf{z})}\left[f(\mathbf{z})\right] &= \nabla_\phi \mathbb{E}_{p(\boldsymbol{\epsilon})}\left[f(g(\boldsymbol{\epsilon}, \phi))\right] \\ &= \mathbb{E}_{p(\boldsymbol{\epsilon})}\left[\nabla_\phi f(g(\boldsymbol{\epsilon}, \phi))\right] \\ &= \mathbb{E}_{p(\boldsymbol{\epsilon})}\left[\nabla_\mathbf{z} f(\mathbf{z})|_{\mathbf{z}=g(\boldsymbol{\epsilon},\phi)} \nabla_\phi g(\boldsymbol{\epsilon}, \phi)\right]\end{aligned}$$

Since $p(\boldsymbol{\epsilon})$ does not depend on ϕ

Using the Chain Rule

Reparameterization trick

- The reparameterization trick essentially moves the dependence on the distribution parameters inside the expectation.
    - This can be seen as propagating gradients through $\mathbf{z}$.
    - To get the correct gradients, the function mapping $\boldsymbol{\epsilon}$ to $\mathbf{z}$ has to be differentiable w.r.t. the distribution parameters ϕ.
- Reparameterizing a Gaussian variable $z \sim \mathcal{N}(\mu, \sigma^2)$:

$$z = \mu + \sigma\epsilon \text{ with } \epsilon \sim \mathcal{N}(0, 1^2)$$

    - Note that this mapping from $\boldsymbol{\epsilon}$ to $\mathbf{z}$ is differentiable w.r.t. both parameters of the distribution, as required.

Reparameterizing distributions

- Any distribution in the scale-location family (Laplace, Cauchy, Student's *t*, etc.) can be reparameterized in the same way.
- Distributions like Gamma and Dirichlet can be parameterized using *implicit* reparameterization.
- Not every distribution can be reparameterized in a differentiable way.
    - For example, discrete variables are not reparameterizable in this way.
- Deep learning frameworks such as TensorFlow and PyTorch support reparameterization for many continuous distributions, making it easy to propagate gradients through samples.

DeepMind

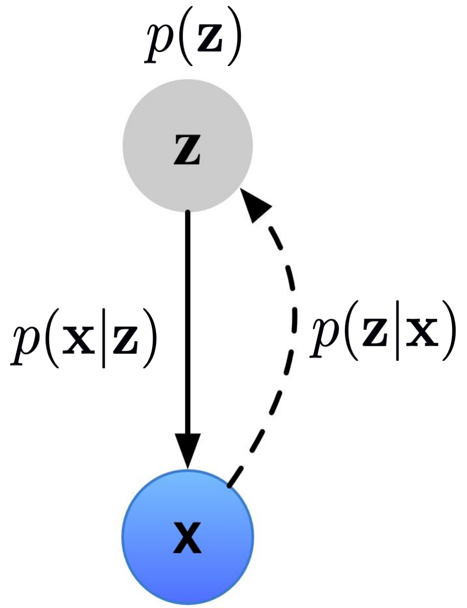
6 Variational Autoencoders

Variational autoencoders

- A VAE is a generative model with continuous latent variables:
  - The likelihood $p_\theta(\mathbf{x}|\mathbf{z})$ and the variational posterior $q_\phi(\mathbf{z}|\mathbf{x})$ are parameterized with neural networks.
  - Both the prior and the variational posterior are usually fully factorized Gaussians.
  - VAEs are trained using amortized variational inference, taking advantage of the reparameterization trick.
- The introduction of VAEs in 2014 was a breakthrough in generative modelling due to their power and scalability.

Variational autoencoders

- Prior: $p(\mathbf{z}) = \mathcal{N}(0, I)$
- Likelihood / decoder:
  - For binary data:
    $$p_\theta(\mathbf{x}|\mathbf{z}) = \text{Bernoulli}(\text{NN}_\theta(\mathbf{z}))$$
  - For real-valued data:
    $$p_\theta(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\text{NN}_\theta(\mathbf{z}), \text{diag}(\text{NN}_\theta(\mathbf{z})))$$
- Variational posterior / encoder:
  $$q_\phi(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\text{NN}_\phi(\mathbf{x}), \text{diag}(\text{NN}_\phi(\mathbf{x})))$$
- Can also use other types of neural nets (e.g. ConvNets) instead of fully-connected neural networks.



ELBO decomposition for VAEs

- VAEs are trained by maximizing the ELBO, which usually written as
$$\mathcal{L}_{\theta,\phi}(\mathbf{x}) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(\mathbf{x}|\mathbf{z})\right] - \mathbb{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z}))$$
- The first term measures how well the model predicts / reconstructs an observation from a sample from the variational posterior.
    - Known as the negative reconstruction error.
- The second term acts as a regularizer, pushing the variational posterior towards the prior.
    - It measures the amount of information about the observation in the latents.
    - Often computed analytically.

The VAE framework

- VAE is now more of a framework than a specific model
  - Can refer to any continuous latent variable model trained using amortized variational inference and the reparameterization trick.
- VAEs have been improved and extended in many ways:
  - Multiple latent layers
  - Non-Gaussian latent variables
  - More expressive priors and posteriors
    - Mixtures, autoregressive, flow-based, implicit, etc.
  - More expressive decoders
    - ResNet, autoregressive (e.g. RNN, PixelCNN)
  - Improved amortized inference (e.g. iterative, variance reduction)

Conclusion

- Invertible models and intractable latent variable models provide two powerful approaches to likelihood-based generative modelling.
- They make different tradeoffs between ease of inference and modelling flexibility / power.
- Models of different types can be combined to take advantage of their complementary strengths.
- This research area is developing rapidly and there are still many contributions to be made.

Thank you

References

Latent variable models and variational inference

- Blei, D. M., Kucukelbir, A., & McAuliffe, J. D. (2017). Variational inference: A review for statisticians. Journal of the American statistical Association, 112(518), 859-877.
- Murphy, Kevin P. Machine learning: a probabilistic perspective. MIT Press, 2012.
- Bishop, Christopher M. Pattern recognition and machine learning. Springer, 2006.
- MacKay, David JC. Information theory, inference and learning algorithms. Cambridge University Press, 2003.

Variational autoencoders and the reparameterization trick

- Kingma, D. P., & Welling, M. (2014). Auto-encoding variational Bayes. International Conference on Learning Representations.
- Rezende, D. J., Mohamed, S., & Wierstra, D. (2014). Stochastic backpropagation and approximate inference in deep generative models. International Conference on Machine Learning.
- Figurnov, M., Mohamed, S., & Mnih, A. (2018). Implicit reparameterization gradients. Neural Information Processing Systems.

References

Invertible Models

- Dinh, L., Sohl-Dickstein, J., & Bengio, S. (2017). Density estimation using Real NVP. International Conference on Learning Representations
- Rezende, D., & Mohamed, S. (2015). Variational Inference with Normalizing Flows. In International Conference on Machine Learning.
- Papamakarios, G., Nalisnick, E., Rezende, D. J., Mohamed, S., & Lakshminarayanan, B. (2019). Normalizing Flows for Probabilistic Modeling and Inference. arXiv preprint arXiv:1912.02762
- Hyvärinen, A., & Oja, E. (2000). Independent component analysis: algorithms and applications. Neural networks, 13(4-5), 411-430.