CSCI 699 - ProbGen

Probabilistic and Generative Models

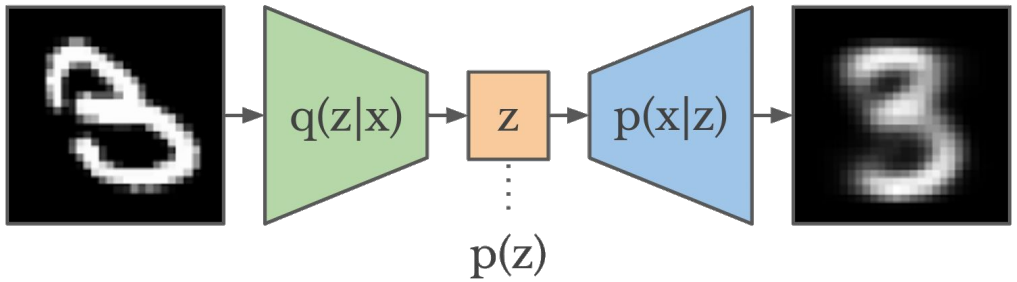
Willie Neiswanger

Lecture 6 - Variational Inference and Variational Autoencoders

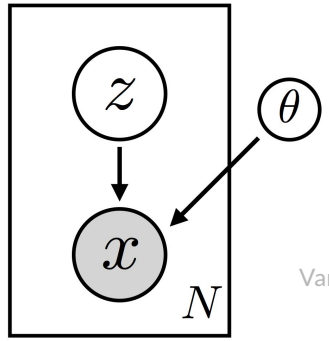
Today

Today

Lecture: The second key type of approximate inference, variational inference (VI) , and the iconic deep generative model, the variational autoencoder (VAE).



"Building Variational Auto-Encoders in TensorFlow", Danijar Hafner

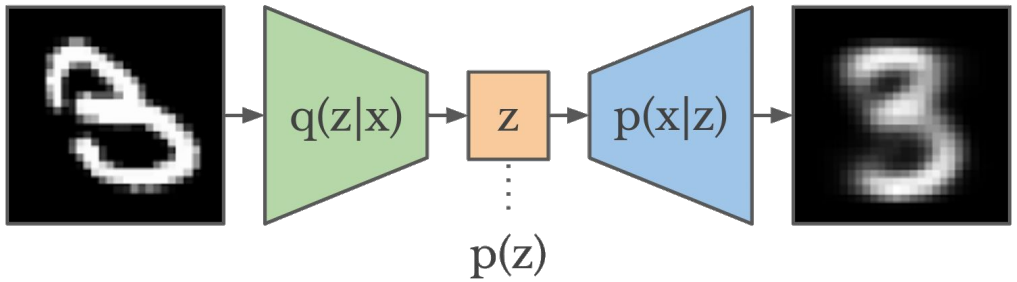


Variational Autoencoder PGM, Pyro documentation

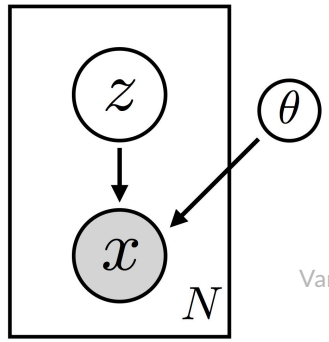
Today

Lecture: The second key type of approximate inference, variational inference (VI) , and the iconic deep generative model, the variational autoencoder (VAE).

- Variational inference (VI) methods, including
  - Evidence lower bound (ELBO), SVI, BBVI.
- Variational autoencoders (VAEs)
  - Reparameterization trick, SGVB.
  - Relation to (classic) autoencoders.



"Building Variational Auto-Encoders in TensorFlow", Danijar Hafner

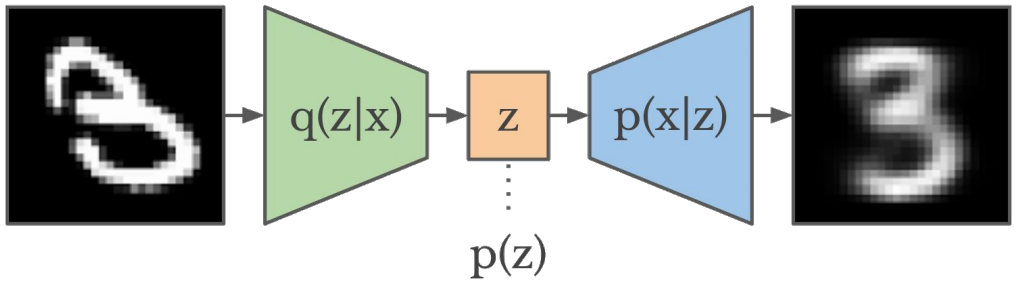


Variational Autoencoder PGM, Pyro documentation

Today

Lecture: The second key type of approximate inference, variational inference (VI) , and the iconic deep generative model, the variational autoencoder (VAE).

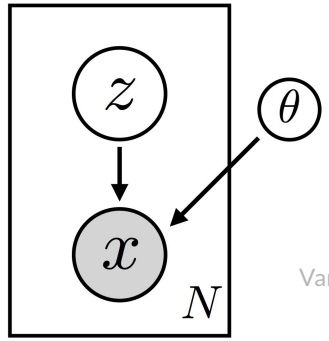
- Variational inference (VI) methods, including
  - Evidence lower bound (ELBO), SVI, BBVI.
- Variational autoencoders (VAEs)
  - Reparameterization trick, SGVB.
  - Relation to (classic) autoencoders.



"Building Variational Auto-Encoders in TensorFlow", Danijar Hafner

After:

- Project feedback/discussion via short in-class meetings with each group.



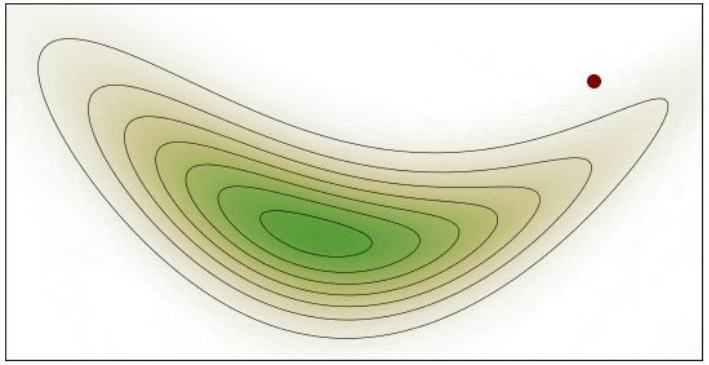
Variational Autoencoder PGM, Pyro documentation

Quick Review: MCMC

Review — Last Class on MCMC

Last week we covered:

- MCMC algorithms, including:
  - Metropolis-Hastings (MH) and Metropolis Algorithm
  - Gibbs Samplings
  - Langevin Monte Carlo (LMC)
  - Hamiltonian Monte Carlo (HMC)



Source: “Creating animations with MCMC”, Krepl 2018, Wikimedia commons,

Review — Last Class on MCMC: Gibbs Sampling

Review — Last Class on MCMC: Gibbs Sampling

A widely-used special case of Metropolis Hastings is known as **Gibbs Sampling**.

Think of it as an *"axially aligned transition function that always yields an acceptance ratio of 1"*.

Review — Last Class on MCMC: Gibbs Sampling

Suppose we have an ordered set of variables in our model: $x_1, \ldots, x_n$

And an initial configuration: $x^0 = (x_1^0, \ldots, x_n^0)$

Then carry out the following procedure (to generate samples from $p(x)$):

Repeat until convergence for $t = 1, 2, 3, \ldots$

- Set $x \leftarrow x^{t-1}$
- For each variable $x_i, \quad i = 1, \ldots, n$
  - Sample $x' \sim p(x_i \mid x_{-i})$
  - Update $x \leftarrow (x_1, \ldots, x_i', \ldots, x_n)$
- Set $x^t \leftarrow x$

Review — Last Class on MCMC: Gibbs Sampling

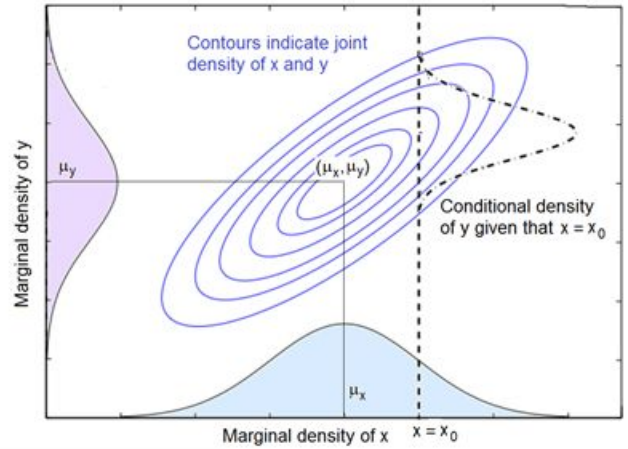
Suppose we have an ordered set of variables in our model: $x_1, \ldots, x_n$

And an initial configuration: $x^0 = (x_1^0, \ldots, x_n^0)$

Then carry out the following procedure (to generate samples from $p(x)$):

Repeat until convergence for $t = 1, 2, 3, \ldots$

- Set $x \leftarrow x^{t-1}$
- For each variable $x_i, \quad i = 1, \ldots, n$
  - Sample $x' \sim p(x_i \mid x_{-i})$
  - Update $x \leftarrow (x_1, \ldots, x_i', \ldots, x_n)$
- Set $x^t \leftarrow x$



Sample from conditional of $p(x)$
(x_{-i} denotes "*all other variables*")

Source: Pietro Vischia - Statistics for HEP

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Review — Last Class on MCMC: Gibbs Sampling

Suppose we have an ordered set of variables in our model: $x_1, \ldots, x_n$

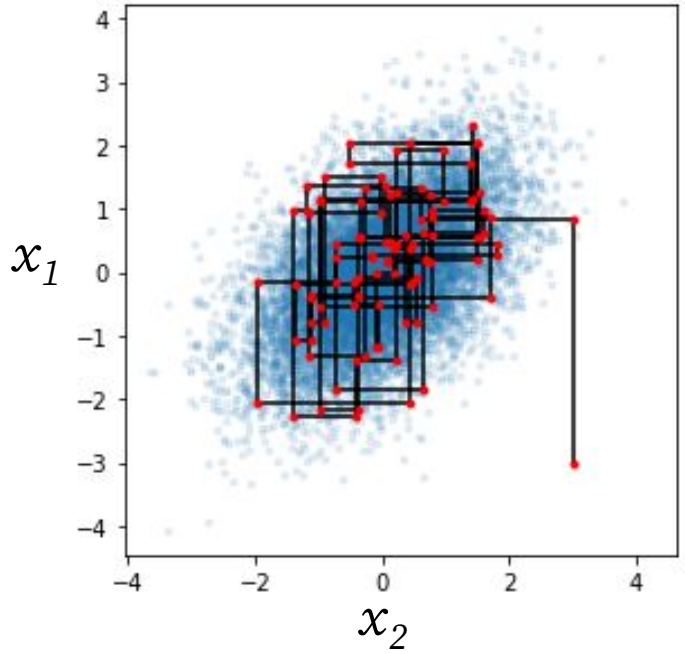
And an initial configuration: $x^0 = (x_1^0, \ldots, x_n^0)$

Then carry out the following procedure (to generate samples from $p(x)$):

Repeat until convergence for $t = 1, 2, 3, \ldots$

- Set $x \leftarrow x^{t-1}$
- For each variable $x_i, \quad i = 1, \ldots, n$
  - Sample $x' \sim p(x_i \mid x_{-i})$
  - Update $x \leftarrow (x_1, \ldots, x_i', \ldots, x_n)$
- Set $x^t \leftarrow x$

Looks like this:



Source: Jessica Stringham, Gibbs Sampling in Python

Review — Last Class on MCMC: Gradient-based MCMC

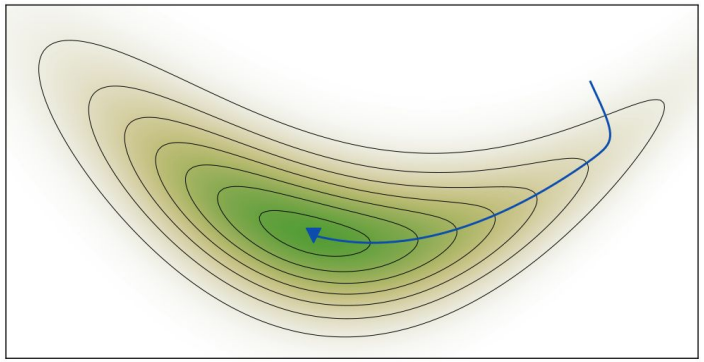
Source: Wikipedia, “Hamiltonian Monte Carlo”

Review — Last Class on MCMC: Gradient-based MCMC

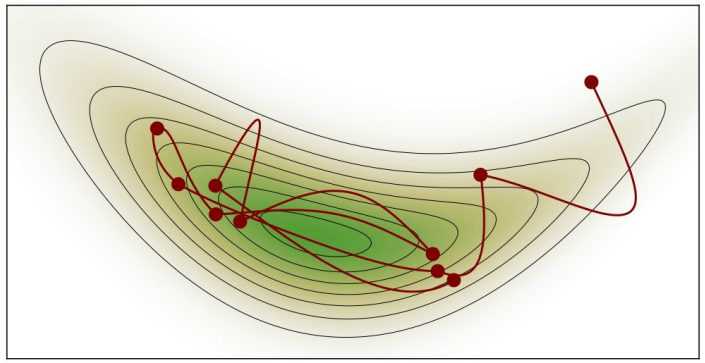
A few popular MCMC methods leverage the *gradient of the PDF* in order to propose a sample in a high mass region from potentially far away in space.

(These methods original came from physics, in particular the field of quantum chromodynamics).

They are called: **Langevin Monte Carlo** (LMC) and **Hamiltonian Monte Carlo** (HMC).



Gradient Descent Path



Hamiltonian Monte Carlo Samples

Source: Wikipedia, "Hamiltonian Monte Carlo"

Review — Last Class on MCMC: Langevin Monte Carlo (LMC)

Each step in LMC can be written:

Next x

$$x_t = x_{t-1} - \frac{\epsilon_t}{2}\nabla_x f(x_{t-1}) + \sqrt{\epsilon_t}\mathcal{N}(0, I_n)$$

Review — Last Class on MCMC: Langevin Monte Carlo (LMC)

Each step in LMC can be written:

Prev. x

$$x_t = x_{t-1} - \frac{\epsilon_t}{2}\nabla_x f(x_{t-1}) + \sqrt{\epsilon_t}\mathcal{N}(0, I_n)$$

Review — Last Class on MCMC: Langevin Monte Carlo (LMC)

Each step in LMC can be written:

Time step (over 2) times gradient of objective (at prev. x)

$$x_t = x_{t-1} - \frac{\epsilon_t}{2}\nabla_x f(x_{t-1}) + \sqrt{\epsilon_t}\mathcal{N}(0, I_n)$$

Review — Last Class on MCMC: Langevin Monte Carlo (LMC)

Each step in LMC can be written:

Time step (square root) times standard normal (multivariate).

$$x_t = x_{t-1} - \frac{\epsilon_t}{2}\nabla_x f(x_{t-1}) + \sqrt{\epsilon_t}\mathcal{N}(0, I_n)$$

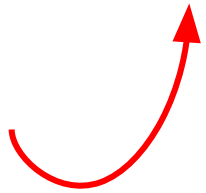
Review — Last Class on MCMC: Langevin Monte Carlo (LMC)

Each step in LMC can be written:

$$x_t = x_{t-1} - \frac{\epsilon_t}{2}\nabla_x f(x_{t-1}) + \sqrt{\epsilon_t}\mathcal{N}(0, I_n)$$

$$x_t = x_{t-1} - \eta_t \nabla_x f(x_{t-1})$$

Compare with gradient descent



Review — Last Class on MCMC: Langevin Monte Carlo (LMC)

Each step in LMC can be written:

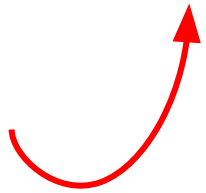
$$x_t = x_{t-1} - \frac{\epsilon_t}{2}\nabla_x f(x_{t-1}) + \sqrt{\epsilon_t}\mathcal{N}(0, I_n)$$

Main difference is an extra "stochastic term"

$$x_t = x_{t-1} - \eta_t \nabla_x f(x_{t-1})$$

Can view this as "*gradient descent plus noise*"

Compare with gradient descent



Review — Last Class on MCMC: Hamiltonian Monte Carlo (HMC)

Review — Last Class on MCMC: Hamiltonian Monte Carlo (HMC)

Similar to LMC (construct a proposal based on gradient of PDF), but…

Take many more steps ⇒ better mixing, and still high acceptance rate.

Review — Last Class on MCMC: Hamiltonian Monte Carlo (HMC)

Similar to LMC (construct a proposal based on gradient of PDF), but…

Take many more steps ⇒ better mixing, and still high acceptance rate.

In particular:

- Take a sequence of gradient steps (in a very specific way), on an *auxiliary variable model*.
- Such that you can travel far away from your previous sample…
- … but stay at a same / similar PDF value.
- (And then still use Metropolis-Hastings to accept/or reject final proposal).

Review — Last Class on MCMC: Hamiltonian Monte Carlo (HMC)

We will use a technique called *auxiliary variables*.

Review — Last Class on MCMC: Hamiltonian Monte Carlo (HMC)

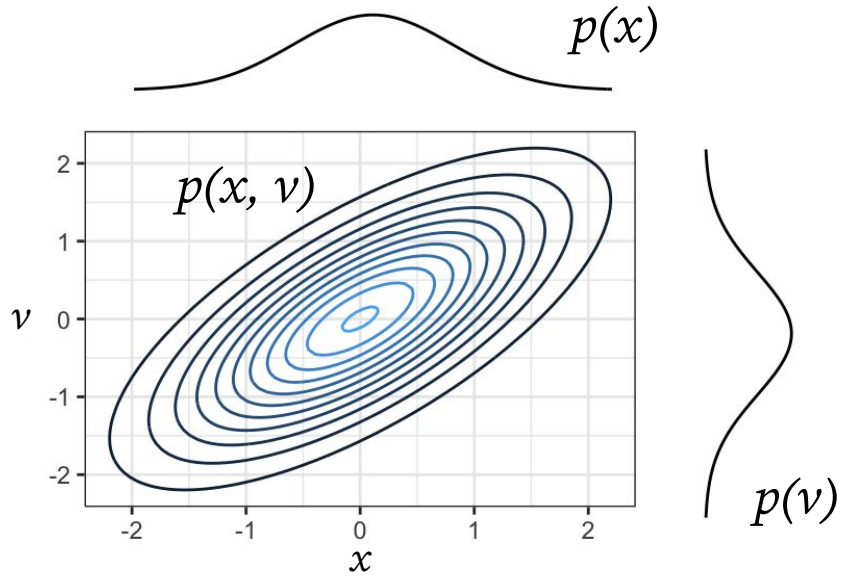
We will use a technique called *auxiliary variables*.

Note that, for a given probability model, you could always introduce *more variables*.

⇒ *i.e.*, you can form a probability distribution over more variables whose marginal is equal to the original probability model).

Suppose we have a probability model $p(x)$

Extend this to the model $p(x, v)$



Variables v are called ***auxiliary variables***

Review — Last Class on MCMC: Hamiltonian Monte Carlo (HMC)

⇒ Then you can simulate
Hamiltonian Dynamics.

Review — Last Class on MCMC: Hamiltonian Monte Carlo (HMC)

⇒ Then you can simulate
Hamiltonian Dynamics.

⇒ Iteratively update x and v
according to update rules:

$$v \leftarrow v - \frac{\epsilon}{2}\nabla_x \log p(x)$$
$$x \leftarrow x + \epsilon v$$
$$v \leftarrow v - \frac{\epsilon}{2}\nabla_x \log p(x)$$

Review — Last Class on MCMC: Hamiltonian Monte Carlo (HMC)

⇒ Then you can simulate *Hamiltonian Dynamics*.

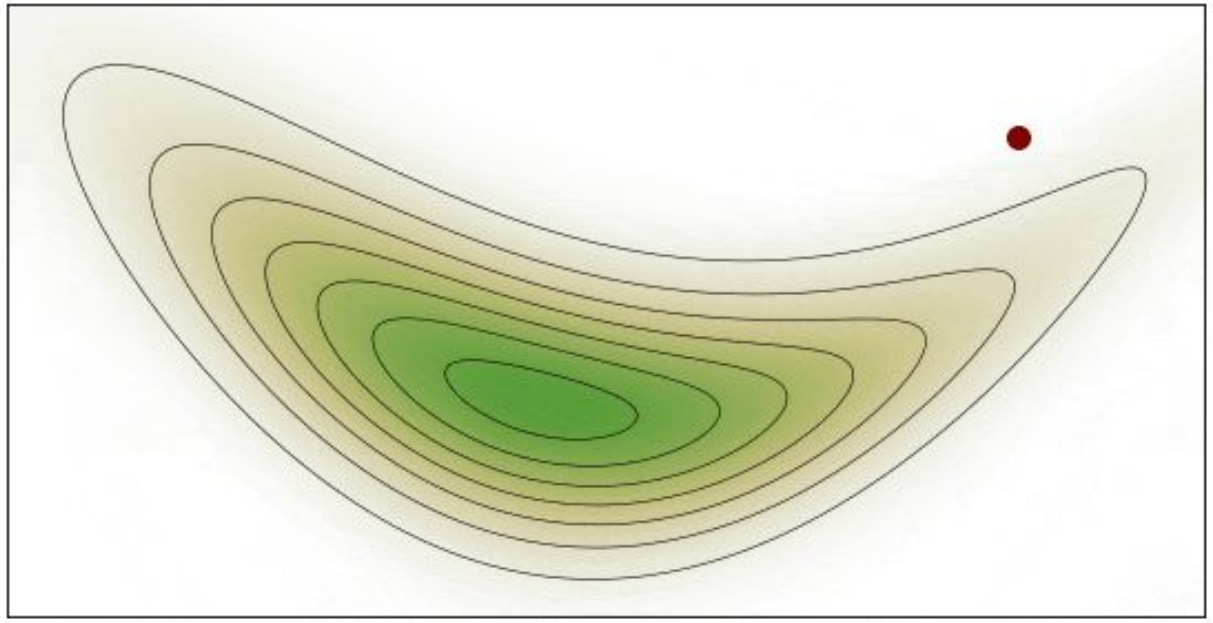
⇒ Iteratively update x and v according to update rules:

$$v \leftarrow v - \frac{\epsilon}{2}\nabla_x \log p(x)$$

$$x \leftarrow x + \epsilon v$$

$$v \leftarrow v - \frac{\epsilon}{2}\nabla_x \log p(x)$$

HMC in practice!



Source: Wikipedia, "Hamiltonian Monte Carlo"

Variational Inference

Variational Inference — Overview

To begin, a few potential shortcomings of sampling-based approximate inference:

Variational Inference — Overview

To begin, a few potential shortcomings of sampling-based approximate inference:

1. Although they are guaranteed to perform *correct inference* given enough time, it is difficult to tell the *approximation quality* after running for a finite amount of time.

Variational Inference — Overview

To begin, a few potential shortcomings of sampling-based approximate inference:

1. Although they are guaranteed to perform *correct inference* given enough time, it is difficult to tell the *approximation quality* after running for a finite amount of time.

2. MCMC methods may require choosing an appropriate proposal distribution (or various hyperparameters). This choice can be difficult.

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Variational Inference — Overview

Instead, we could take the strategy/philosophy of *inference as optimization*.

This method of approximate inference is known as **variational inference (VI)**.

Variational Inference — Overview

Instead, we could take the strategy/philosophy of *inference as optimization*.

This method of approximate inference is known as **variational inference (VI)**.

The main idea:

Variational Inference — Overview

Instead, we could take the strategy/philosophy of *inference as optimization*.

This method of approximate inference is known as **variational inference (VI)**.

The main idea:

- Suppose we are given an intractable (*complex*) probability distribution $p(x)$.

Variational Inference — Overview

Instead, we could take the strategy/philosophy of *inference as optimization*.

This method of approximate inference is known as **variational inference (VI)**.

The main idea:

- Suppose we are given an intractable (*complex*) probability distribution $p(x)$.

E.g., only known up to a normalization constant

Variational Inference — Overview

Instead, we could take the strategy/philosophy of *inference as optimization*.

This method of approximate inference is known as **variational inference (VI)**.

The main idea:

- Suppose we are given an intractable (*complex*) probability distribution $p(x)$.
- VI aims to solve an optimization problem over a set of tractable distributions Q.

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Variational Inference — Overview

Instead, we could take the strategy/philosophy of *inference as optimization*.

This method of approximate inference is known as **variational inference (VI)**.

The main idea:

- Suppose we are given an intractable (*complex*) probability distribution $p(x)$.
- VI aims to solve an optimization problem over a set of tractable distributions Q.
- And, after optimization, return a $q(x) \in Q$ that is *most similar* to $p(x)$.

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Variational Inference — Overview

Instead, we could take the strategy/philosophy of *inference as optimization*.

This method of approximate inference is known as **variational inference (VI)**.

The main idea:

- Suppose we are given an intractable (*complex*) probability distribution $p(x)$.
- VI aims to solve an optimization problem over a set of tractable distributions Q.
- And, after optimization, return a $q(x) \in Q$ that is *most similar* to $p(x)$.

Typically: in terms of KL divergence

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Variational Inference — Overview

Instead, we could take the strategy/philosophy of *inference as optimization*.

This method of approximate inference is known as **variational inference (VI)**.

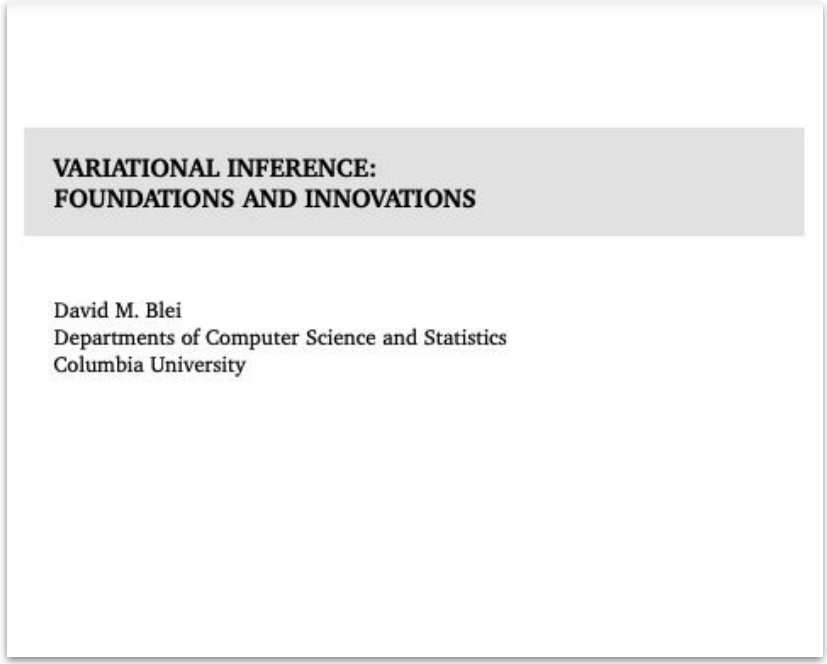
The main idea:

- Suppose we are given an intractable (*complex*) probability distribution $p(x)$.
- VI aims to solve an optimization problem over a set of tractable distributions Q.
- And, after optimization, return a $q(x) \in Q$ that is *most similar* to $p(x)$.
- We can then make an inference query on $q(x)$ instead of $p(x)$.

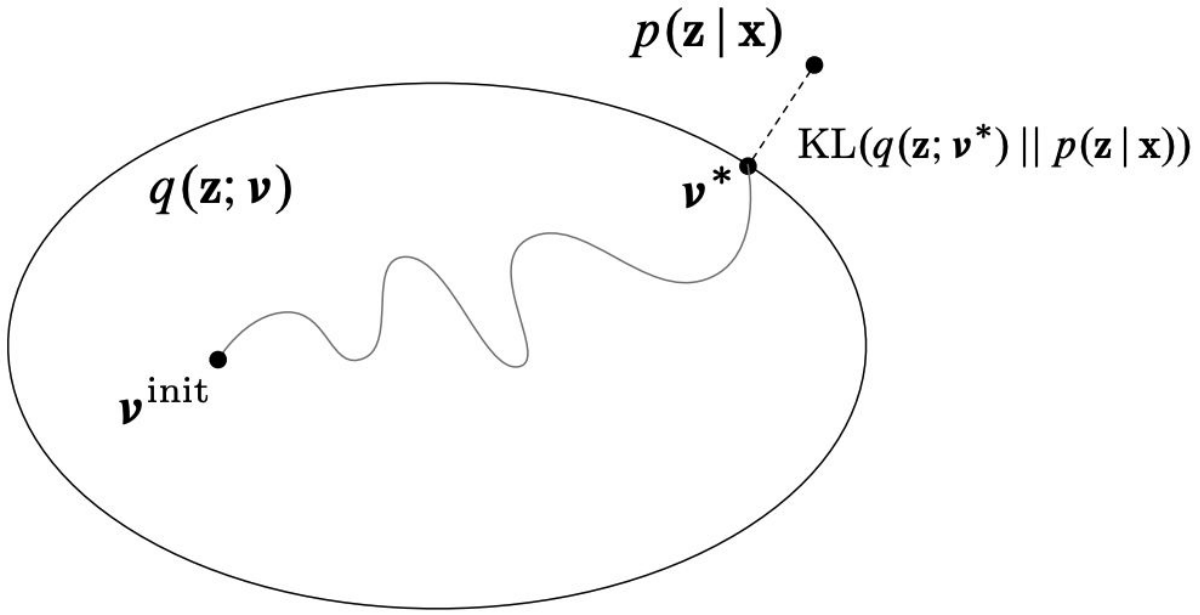
Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Variational Inference — Overview

An illustration from Dave Blei's Variational Inference Tutorial:


VARIATIONAL INFERENCE:
FOUNDATIONS AND INNOVATIONS

David M. Blei
Departments of Computer Science and Statistics
Columbia University

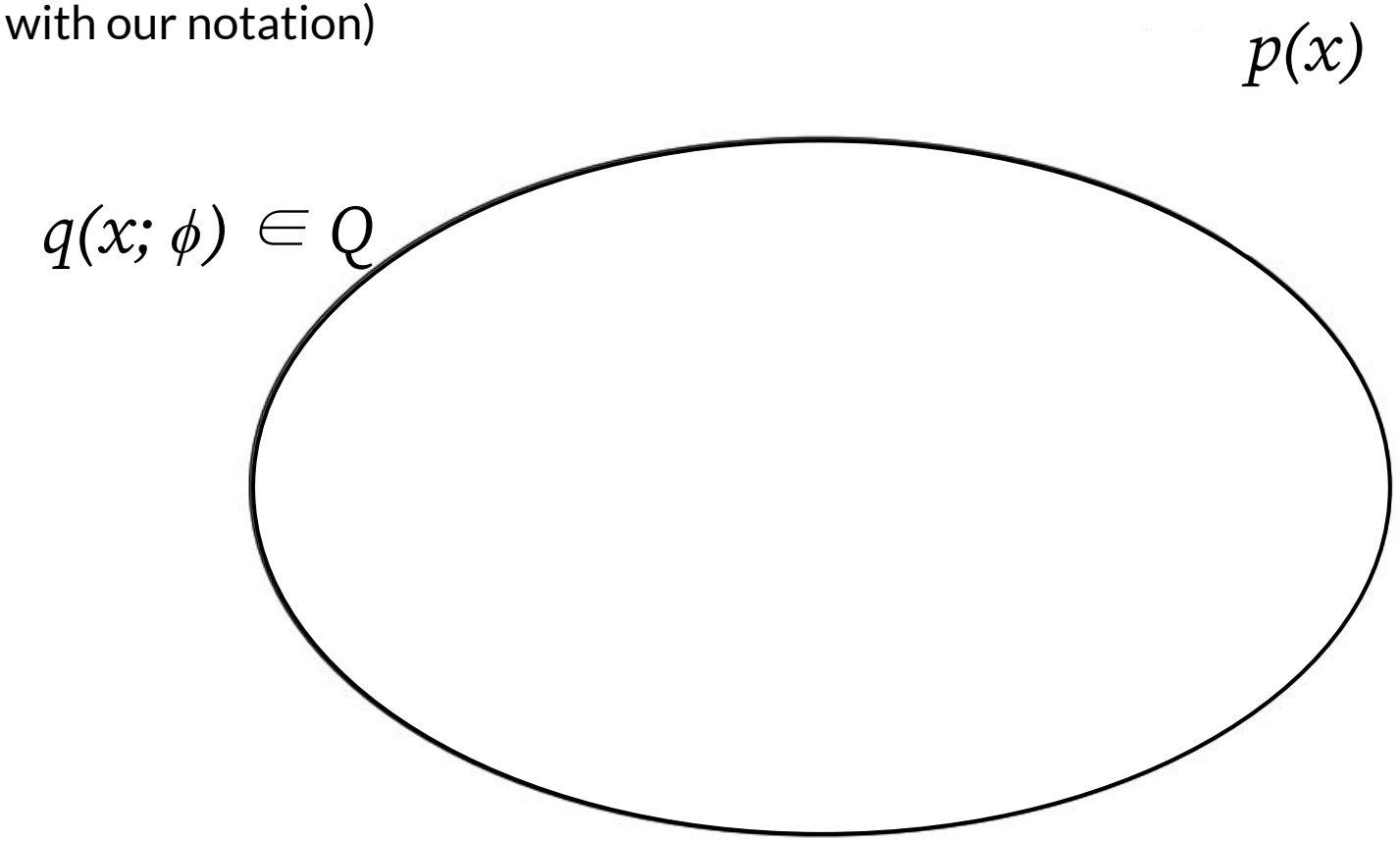


Source: Dave Blei, "Variational Inference"

Variational Inference — Overview

An illustration from Dave Blei's Variational Inference Tutorial:

(replaced with our notation)

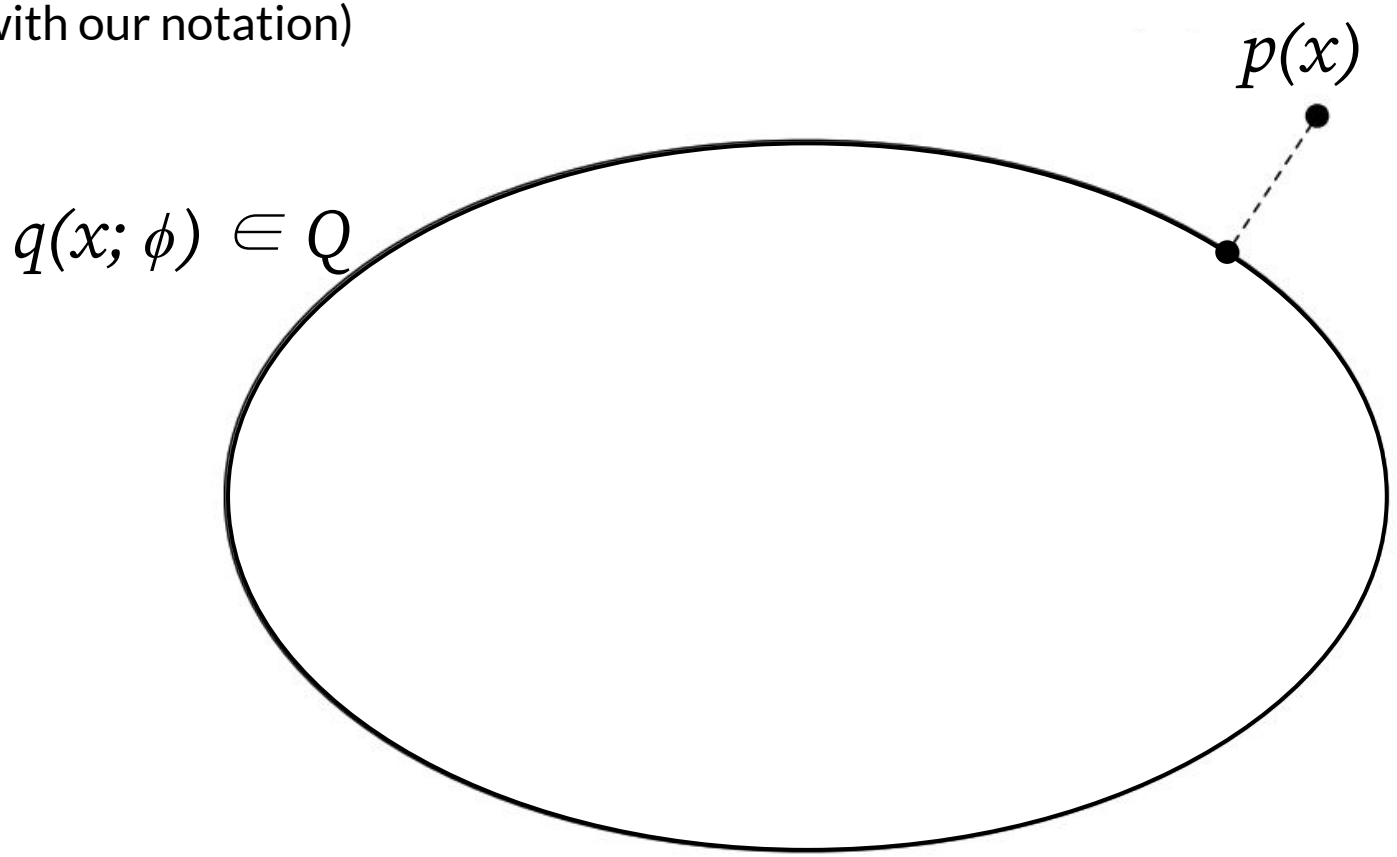


Source: Dave Blei, "Variational Inference"

Variational Inference — Overview

An illustration from Dave Blei's Variational Inference Tutorial:

(replaced with our notation)

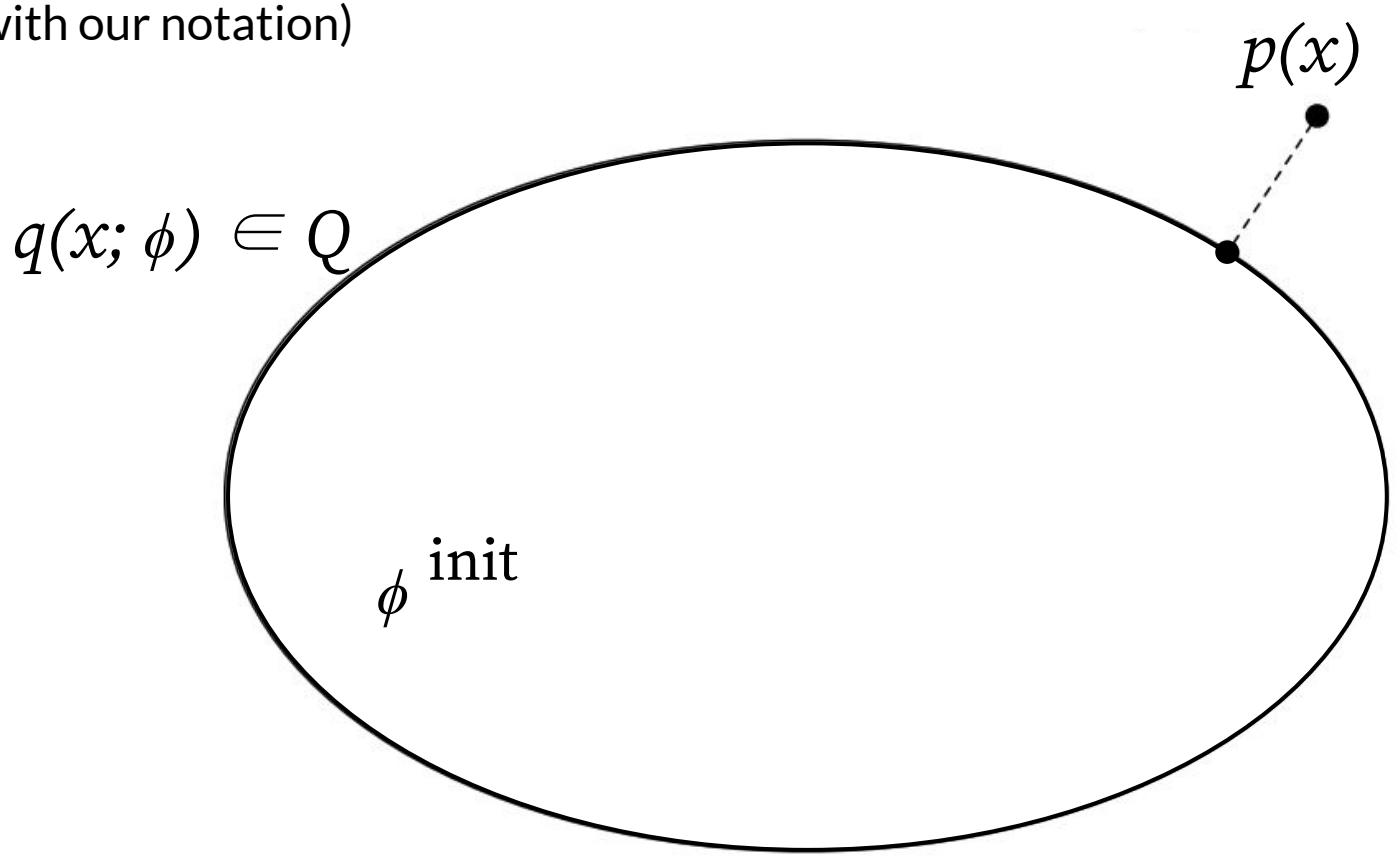


Source: Dave Blei, "Variational Inference"

Variational Inference — Overview

An illustration from Dave Blei's Variational Inference Tutorial:

(replaced with our notation)

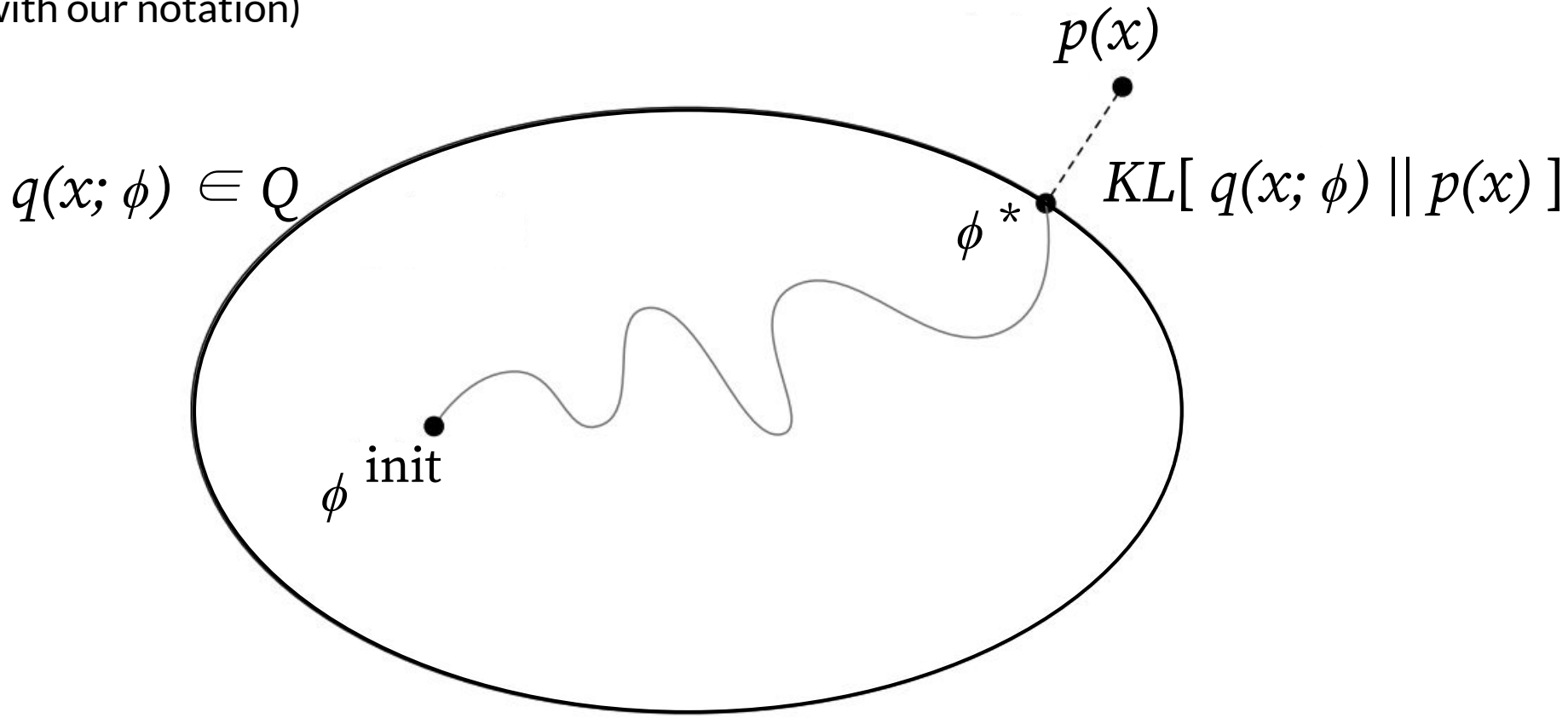


Source: Dave Blei, "Variational Inference"

Variational Inference — Overview

An illustration from Dave Blei's Variational Inference Tutorial:

(replaced with our notation)



Source: Dave Blei, "Variational Inference"

Variational Inference — Overview

Main differences with sampling/MCMC methods:

Variational Inference — Overview

Main differences with sampling/MCMC methods:

- Unlike sampling, variational methods will almost never find the exact/globally optimal solution (even in the limit as $T \to \infty$).

Variational Inference — Overview

Main differences with sampling/MCMC methods:

- Unlike sampling, variational methods will almost never find the exact/globally optimal solution (even in the limit as $T \to \infty$).
- However, it is easy to determine if they have converged (at least to a local optima).

Variational Inference — Overview

Main differences with sampling/MCMC methods:

- Unlike sampling, variational methods will almost never find the exact/globally optimal solution (even in the limit as $T \to \infty$).
- However, it is easy to determine if they have converged (at least to a local optima).
- In practice, variational inference methods often scale better.

Variational Inference — Overview

Main differences with sampling/MCMC methods:

- Unlike sampling, variational methods will almost never find the exact/globally optimal solution (even in the limit as $T \to \infty$).
- However, it is easy to determine if they have converged (at least to a local optima).
- In practice, variational inference methods often scale better.
- ⇒ Amenable to techniques like stochastic gradient optimization, parallelization over multiple processors, and acceleration using GPUs.
    - (While it's easier for VI, this is also a popular field of study for MCMC).

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Kullback-Leibler Divergence

Quick background on Kullback-Leibler (KL) Divergence.

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Kullback-Leibler Divergence

Quick background on Kullback-Leibler (KL) Divergence.

To optimize for an optimal approximation $q(x)$, we must choose an approximating family Q, and an objective to optimize.

The objective should capture the similarity between $q(x)$ and $p(x)$.

The field of information theory provides a tool: **Kullback-Leibler (KL) divergence**.

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Kullback-Leibler Divergence

Intuitively + visually:

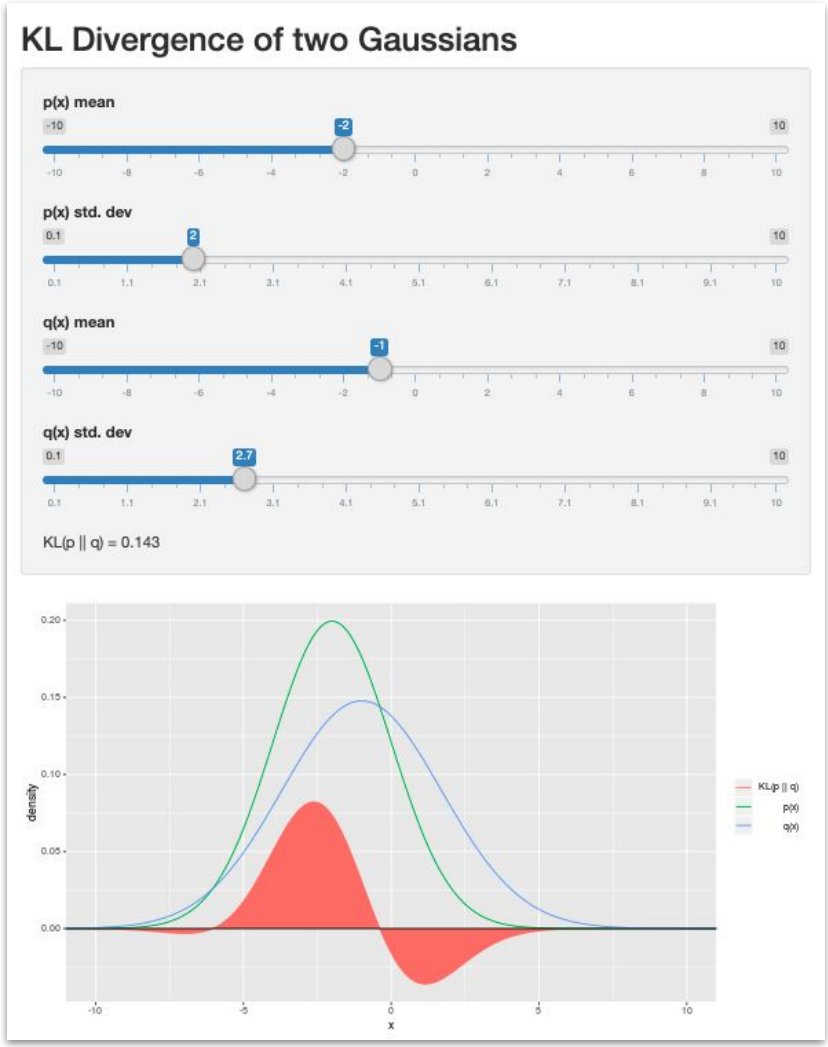
Kullback-Leibler Divergence

Intuitively + visually:

It is a “divergence” (~distance) between distributions $q(x)$ and $p(x)$:

KL divergence online demo:

https://gnarlyware.com/blog/kl-divergence-online-demo/



Kullback-Leibler Divergence

More formally:

Kullback-Leibler Divergence

More formally:

The KL divergence between distributions q and p with *discrete support* is:

$$\mathrm{KL}\left[q \parallel p\right] = \sum_{x} q(x) \log\left(\frac{q(x)}{p(x)}\right) = \mathbb{E}_{q(x)}\left[\log\left(\frac{q(x)}{p(x)}\right)\right]$$

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Kullback-Leibler Divergence

More formally:

The KL divergence between distributions q and p with *discrete support* is:

$$\mathrm{KL}\left[q \parallel p\right] = \sum_{x} q(x) \log\left(\frac{q(x)}{p(x)}\right) = \mathbb{E}_{q(x)}\left[\log\left(\frac{q(x)}{p(x)}\right)\right]$$

And the KL divergence between distributions q and p with *continuous support* is:

$$\mathrm{KL}\left[q \parallel p\right] = \int_{x \in \mathcal{X}} q(x) \log\left(\frac{q(x)}{p(x)}\right) dx = \mathbb{E}_{q(x)}\left[\log\left(\frac{q(x)}{p(x)}\right)\right]$$

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Kullback-Leibler Divergence

The KL divergence has a couple of properties that make it useful in this setting:

Kullback-Leibler Divergence

The KL divergence has a couple of properties that make it useful in this setting:

- $\mathrm{KL}\left[q \parallel p\right] \geq 0$ for all q, p.

Kullback-Leibler Divergence

The KL divergence has a couple of properties that make it useful in this setting:

- $\mathrm{KL}\left[q \parallel p\right] \geq 0$ for all q, p.
- $\mathrm{KL}\left[q \parallel p\right] = 0$ if and only if $q = p$.

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Kullback-Leibler Divergence

The KL divergence has a couple of properties that make it useful in this setting:

- $\mathrm{KL}\left[q \parallel p\right] \geq 0$ for all q, p.
- $\mathrm{KL}\left[q \parallel p\right] = 0$ if and only if $q = p$.

Importantly, note that $\mathrm{KL}\left[q \parallel p\right] \neq \mathrm{KL}\left[p \parallel q\right]$.

⇒ *i.e.*, KL divergence is **not symmetric**.

⇒ (this is why we say that it is a “divergence” and not a “distance”).

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Kullback-Leibler Divergence

Intuitively, when minimizing KL divergence in variational inference:

(in the words of Dave Blei)

Source: Dave Blei, “Variational Inference”

Kullback-Leibler Divergence

$$\mathrm{KL}\left[q \parallel p\right] = \sum_{x} q(x) \log\left(\frac{q(x)}{p(x)}\right)$$

Intuitively, when minimizing KL divergence in variational inference:

(in the words of Dave Blei)

Source: Dave Blei, “Variational Inference”

Kullback-Leibler Divergence

$$\mathrm{KL}\left[q \parallel p\right] = \sum_{x} q(x) \log\left(\frac{q(x)}{p(x)}\right)$$

Intuitively, when minimizing KL divergence in variational inference:

(in the words of Dave Blei)

- If $q(x)$ is high and $p(x)$ is also high (or higher) then we are happy. 😊

Source: Dave Blei, "Variational Inference"

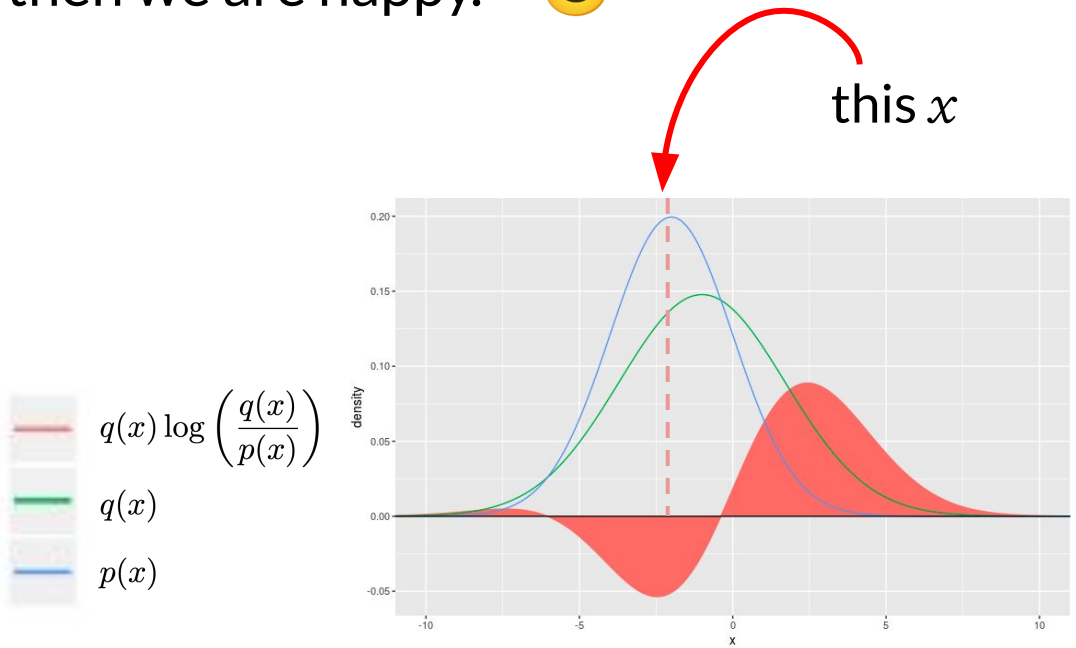
Kullback-Leibler Divergence

$$\mathrm{KL}\left[q \parallel p\right] = \sum_{x} q(x) \log\left(\frac{q(x)}{p(x)}\right)$$

Intuitively, when minimizing KL divergence in variational inference:

(in the words of Dave Blei)

- If $q(x)$ is high and $p(x)$ is also high (or higher) then we are happy. 😊



Source: Dave Blei, "Variational Inference"

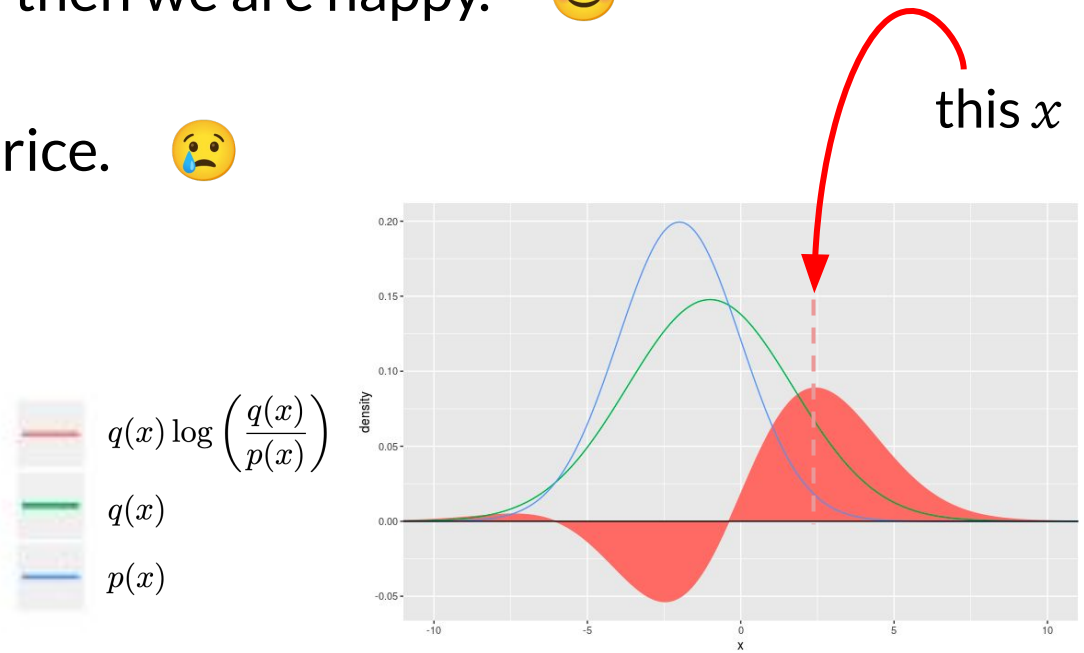
Kullback-Leibler Divergence

$$\mathrm{KL}\left[q \parallel p\right] = \sum_{x} q(x) \log\left(\frac{q(x)}{p(x)}\right)$$

Intuitively, when minimizing KL divergence in variational inference:

(in the words of Dave Blei)

- If $q(x)$ is high and $p(x)$ is also high (or higher) then we are happy. 😊
- If $q(x)$ is high and $p(x)$ is *low*, then we pay a price. 😢



Source: Dave Blei, "Variational Inference"

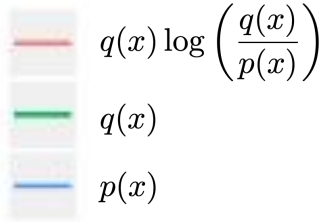
Kullback-Leibler Divergence

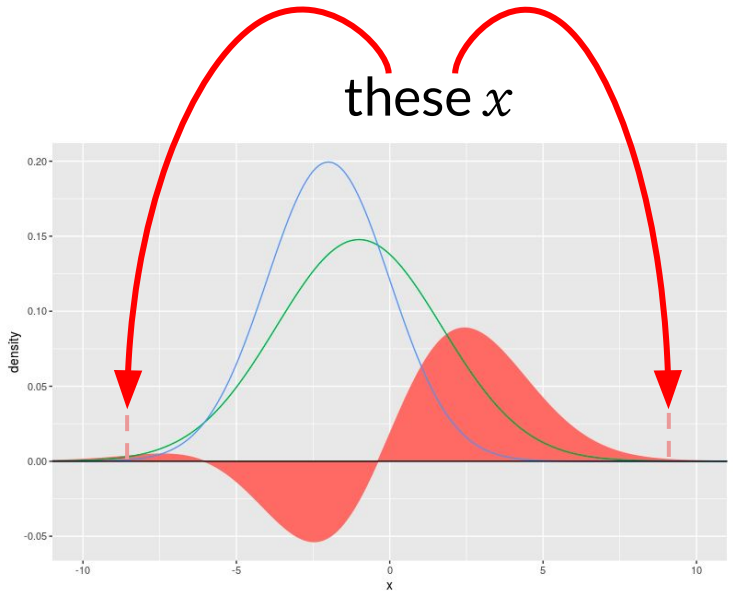
$$\mathrm{KL}\left[q \parallel p\right] = \sum_{x} q(x) \log\left(\frac{q(x)}{p(x)}\right)$$

Intuitively, when minimizing KL divergence in variational inference:

(in the words of Dave Blei)

- If $q(x)$ is high and $p(x)$ is also high (or higher) then we are happy. 😊
- If $q(x)$ is high and $p(x)$ is *low*, then we pay a price. 😢
- If $q(x)$ is low… then we don't care. 🤭





Source: Dave Blei, "Variational Inference"

The Evidence Lower Bound

How do we perform variational inference with the KL divergence?

Next we will go through an overview of the main strategy.

The Evidence Lower Bound

Suppose we have an arbitrary probabilistic model of n variables: $x = (x_1, \ldots, x_n)$

The Evidence Lower Bound

Suppose we have an arbitrary probabilistic model of n variables: $x = (x_1, \ldots, x_n)$

Let's write this in a general way, as a (normalized) product of factors:

$$p_\theta(x_1, \ldots, x_n) = \frac{1}{Z(\theta)} \tilde{p}_\theta(x_1, \ldots, x_n) = \frac{1}{Z(\theta)} \prod_k \phi_k(x_k; \theta)$$

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

The Evidence Lower Bound

Suppose we have an arbitrary probabilistic model of n variables: $x = (x_1, \ldots, x_n)$

Let's write this in a general way, as a (normalized) product of factors:

$$p_\theta(x_1, \ldots, x_n) = \frac{1}{Z(\theta)} \tilde{p}_\theta(x_1, \ldots, x_n) = \frac{1}{Z(\theta)} \prod_k \phi_k(x_k; \theta)$$

Assume $p(x)$ has a parameter θ

The Evidence Lower Bound

Suppose we have an arbitrary probabilistic model of n variables: $x = (x_1, \ldots, x_n)$

Let's write this in a general way, as a (normalized) product of factors:

$$p_\theta(x_1, \ldots, x_n) = \frac{1}{Z(\theta)} \tilde{p}_\theta(x_1, \ldots, x_n) = \frac{1}{Z(\theta)} \prod_k \phi_k(x_k; \theta)$$

Assume $p(x)$ has a parameter θ

Z is a normalization constant

The Evidence Lower Bound

Suppose we have an arbitrary probabilistic model of n variables: $x = (x_1, \ldots, x_n)$

Let's write this in a general way, as a (normalized) product of factors:

$$p_\theta(x_1, \ldots, x_n) = \frac{1}{Z(\theta)} \tilde{p}_\theta(x_1, \ldots, x_n) = \frac{1}{Z(\theta)} \prod_k \phi_k(x_k; \theta)$$

Assume $p(x)$ has a parameter θ

Z is a normalization constant

Recall: can write most directed / undirected PGMs in this way.

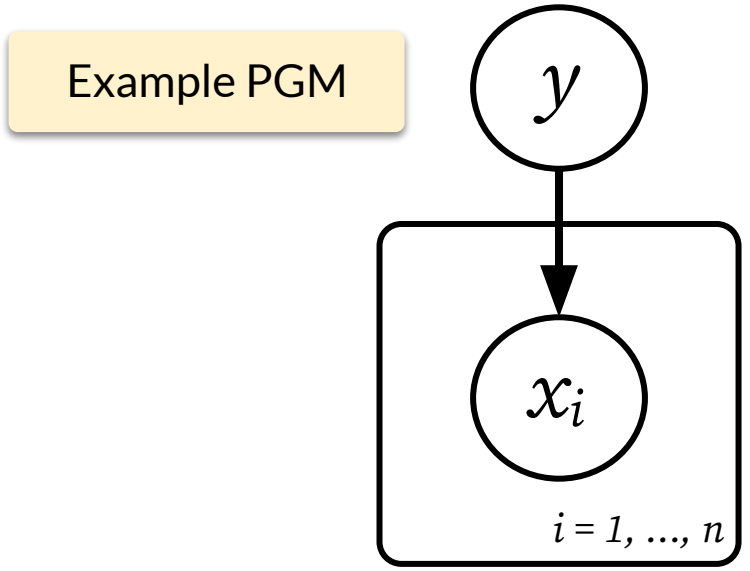
Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

The Evidence Lower Bound

Example for a prototypical PGM (Bayesian network):

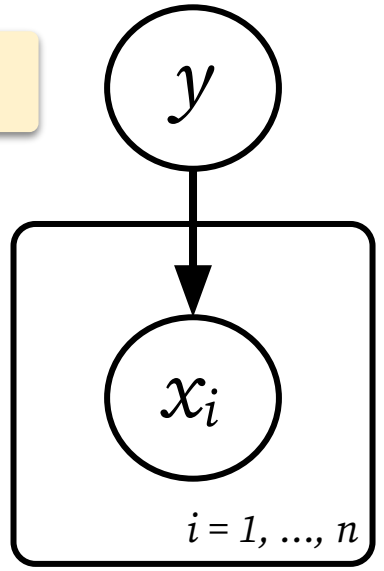
The Evidence Lower Bound

Example for a prototypical PGM (Bayesian network):



The Evidence Lower Bound

Example PGM



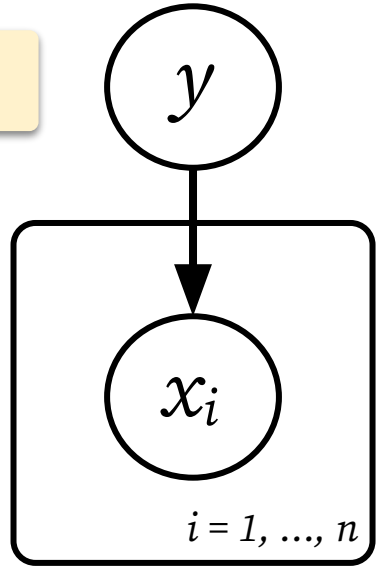
Example for a prototypical PGM (Bayesian network):

Can can write out the joint PDF as:

$$p(y, x_1, \ldots, x_n) = p(y) \prod_i p(x_i \mid y)$$

The Evidence Lower Bound

Example PGM



Example for a prototypical PGM (Bayesian network):

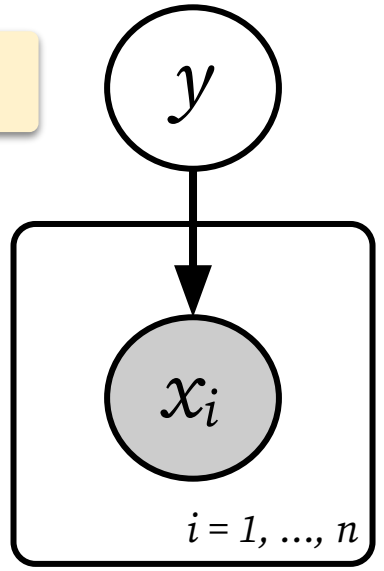
Can can write out the joint PDF as:

$$p(y, x_1, \ldots, x_n) = \underbrace{p(y) \prod_i p(x_i \mid y)}$$

Product of factors

The Evidence Lower Bound

Example PGM



Example for a prototypical PGM (Bayesian network):

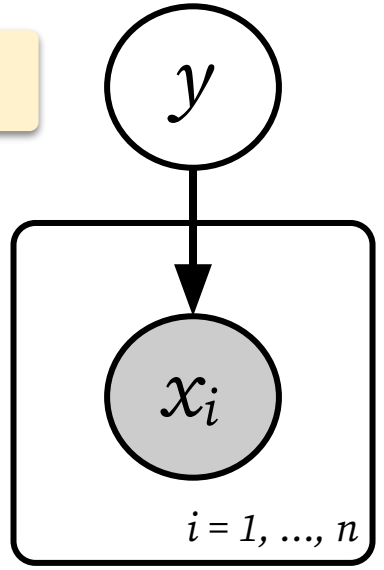
Can can write out the joint PDF as:

$$p(y, x_1, \ldots, x_n) = p(y) \prod_i p(x_i \mid y)$$

Alternatively, we can write out the posterior PDF (prob of latents given observations) as:

The Evidence Lower Bound

Example PGM



Example for a prototypical PGM (Bayesian network):

Can can write out the joint PDF as:

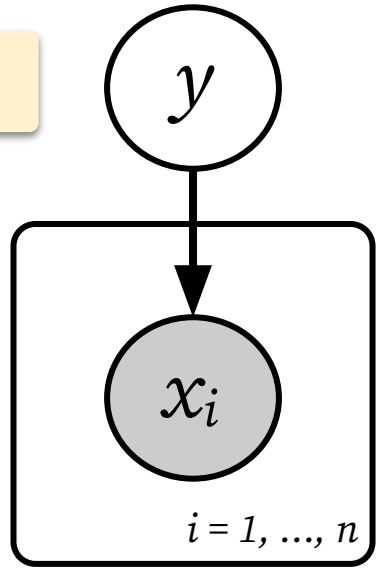
$$p(y, x_1, \ldots, x_n) = p(y) \prod_i p(x_i \mid y)$$

Alternatively, we can write out the posterior PDF (prob of latents given observations) as:

$$p(y \mid x_1, \ldots, x_n) = \frac{p(y, x_1, \ldots, x_n)}{p(x_1, \ldots, x_n)} = \frac{1}{Z} p(y, x_1, \ldots, x_n)$$

The Evidence Lower Bound

Example PGM



Example for a prototypical PGM (Bayesian network):

Can can write out the joint PDF as:

$$p(y, x_1, \ldots, x_n) = p(y) \prod_i p(x_i \mid y)$$

Alternatively, we can write out the posterior PDF (prob of latents given observations) as:

Factorize

$$p(y \mid x_1, \ldots, x_n) = \frac{p(y, x_1, \ldots, x_n)}{p(x_1, \ldots, x_n)} = \frac{1}{Z} p(y) \prod_i p(x_i \mid y)$$

The Evidence Lower Bound

Given a probability model such as this, in the form $p_\theta(x) = \frac{1}{Z(\theta)} \prod_k \phi_k(x_k; \theta)$,

The Evidence Lower Bound

Given a probability model such as this, in the form $p_\theta(x) = \frac{1}{Z(\theta)} \prod_k \phi_k(x_k; \theta)$,

Suppose that we want to perform variational inference (optimize $\mathrm{KL}\left[q \parallel p\right]$).

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

The Evidence Lower Bound

Given a probability model such as this, in the form $p_\theta(x) = \frac{1}{Z(\theta)} \prod_k \phi_k(x_k; \theta)$,

Suppose that we want to perform variational inference (optimize $\mathrm{KL}\left[q \parallel p\right]$).

It is difficult to optimize $\mathrm{KL}\left[q \parallel p\right]$ directly! For a few reasons:

The Evidence Lower Bound

Given a probability model such as this, in the form $p_\theta(x) = \frac{1}{Z(\theta)} \prod_k \phi_k(x_k; \theta)$,

Suppose that we want to perform variational inference (optimize $\mathrm{KL}\left[q \parallel p\right]$).

It is difficult to optimize $\mathrm{KL}\left[q \parallel p\right]$ directly! For a few reasons:

- Potentially intractable normalization constant.
- ⇒ difficult to take gradients due to this.
- In fact, it's difficult even to evaluate $\mathrm{KL}\left[q \parallel p\right]$ (or $p(x)$ for that matter).

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

The Evidence Lower Bound

Therefore, we will not work with the KL divergence directly.

The Evidence Lower Bound

Therefore, we will not work with the KL divergence directly.

Instead, we will use the following objective, which has a similar form as the KL divergence, but only involves the *unnormalized probability* $\tilde{p}(x) = \prod_k \phi_k(x_k; \theta)$.

The Evidence Lower Bound

Therefore, we will not work with the KL divergence directly.

Instead, we will use the following objective, which has a similar form as the KL divergence, but only involves the *unnormalized probability* $\tilde{p}(x) = \prod_k \phi_k(x_k; \theta)$.

For example, if $p(x)$ is the posterior PDF (of latents given observed), then an example of $\tilde{p}(x)$ is the joint PDF.

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

The Evidence Lower Bound

Therefore, we will not work with the KL divergence directly.

Instead, we will use the following objective, which has a similar form as the KL divergence, but only involves the *unnormalized probability* $\tilde{p}(x) = \prod_k \phi_k(x_k; \theta)$.

This objective is:

The Evidence Lower Bound

Therefore, we will not work with the KL divergence directly.

Instead, we will use the following objective, which has a similar form as the KL divergence, but only involves the *unnormalized probability* $\tilde{p}(x) = \prod_k \phi_k(x_k; \theta)$.

This objective is:

$$J(q) = \sum_x q(x) \log \frac{q(x)}{\tilde{p}(x)} = \mathbb{E}_{q(x)} \left[\log \frac{q(x)}{\tilde{p}(x)} \right]$$

For discrete distributions

The Evidence Lower Bound

Therefore, we will not work with the KL divergence directly.

Instead, we will use the following objective, which has a similar form as the KL divergence, but only involves the *unnormalized probability* $\tilde{p}(x) = \prod_k \phi_k(x_k; \theta)$.

This objective is:

$$J(q) = \sum_x q(x) \log \frac{q(x)}{\tilde{p}(x)} = \mathbb{E}_{q(x)} \left[\log \frac{q(x)}{\tilde{p}(x)} \right]$$

For discrete distributions

$$J(q) = \int_{\mathcal{X}} q(x) \log \frac{q(x)}{\tilde{p}(x)} dx = \mathbb{E}_{q(x)} \left[\log \frac{q(x)}{\tilde{p}(x)} \right]$$

For continuous distributions

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

The Evidence Lower Bound

This function is *tractable* (we can evaluate it), and it has the following property:

The Evidence Lower Bound

This function is *tractable* (we can evaluate it), and it has the following property:

Assuming a discrete distribution WLOG

$$
\begin{aligned}
J(q) &= \sum_x q(x) \log \frac{q(x)}{\tilde{p}(x)} \\
&= \sum_x q(x) \log \frac{q(x)}{p(x)} - \log Z(\theta) \\
&= \mathrm{KL}\left[q(x) \parallel p(x)\right] - \log Z(\theta)
\end{aligned}
$$

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

The Evidence Lower Bound

This function is *tractable* (we can evaluate it), and it has the following property:

$$J(q) = \sum_x q(x) \log \frac{q(x)}{\tilde{p}(x)}$$

Definition of $J(q)$

Assuming a discrete distribution WLOG

$$= \sum_x q(x) \log \frac{q(x)}{p(x)} - \log Z(\theta)$$

$$= \mathrm{KL}\left[q(x) \,\|\, p(x)\right] - \log Z(\theta)$$

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

The Evidence Lower Bound

This function is *tractable* (we can evaluate it), and it has the following property:

$$J(q) = \sum_x q(x) \log \frac{q(x)}{\tilde{p}(x)}$$

Definition of $J(q)$

Assuming a discrete distribution WLOG

$$= \sum_x q(x) \log \frac{q(x)}{p(x)} - \log Z(\theta)$$

Definition of $p(x)$

$$= \mathrm{KL}\left[q(x) \,\|\, p(x)\right] - \log Z(\theta)$$

The Evidence Lower Bound

This function is *tractable* (we can evaluate it), and it has the following property:

Assuming a discrete distribution WLOG

$$J(q) = \sum_x q(x) \log \frac{q(x)}{\tilde{p}(x)}$$

Definition of $J(q)$

$$= \sum_x q(x) \log \frac{q(x)}{p(x)} - \log Z(\theta)$$

Definition of $p(x)$

$$= \mathrm{KL}\left[q(x) \parallel p(x)\right] - \log Z(\theta)$$

Definition of KL

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

The Evidence Lower Bound

This function is *tractable* (we can evaluate it), and it has the following property:

Assuming a discrete distribution WLOG

$$J(q) = \sum_x q(x) \log \frac{q(x)}{\tilde{p}(x)}$$

Definition of $J(q)$

$$= \sum_x q(x) \log \frac{q(x)}{p(x)} - \log Z(\theta)$$

Definition of $p(x)$

$$= \mathrm{KL}\left[q(x) \,\|\, p(x)\right] - \log Z(\theta)$$

Definition of KL

$\Rightarrow J(q)$ is equal to the KL divergence minus the *log partition function* (*aka log normalizer*).

(*aka "marginal log-likelihood"*).

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

The Evidence Lower Bound

Also note that KL divergence is non-negative: $\mathrm{KL}\left[q \parallel p\right] \geq 0$

Which implies the following relation:

The Evidence Lower Bound

Also note that KL divergence is non-negative: $\mathrm{KL}\left[q \parallel p\right] \geq 0$

Which implies the following relation:

$$\log Z(\theta) = \mathrm{KL}\left[q(x) \parallel p(x)\right] - J(q)$$

The Evidence Lower Bound

Also note that KL divergence is non-negative: $\mathrm{KL}\left[q \parallel p\right] \geq 0$

Which implies the following relation:

$$\log Z(\theta) = \mathrm{KL}\left[q(x) \parallel p(x)\right] - J(q) \geq -J(q)$$

The Evidence Lower Bound

Also note that KL divergence is non-negative: $\mathrm{KL}\left[q \parallel p\right] \geq 0$

Which implies the following relation:

$$\log Z(\theta) = \mathrm{KL}\left[q(x) \parallel p(x)\right] - J(q) \geq -J(q)$$

$$\implies \log Z(\theta) \geq -J(q)$$

The Evidence Lower Bound

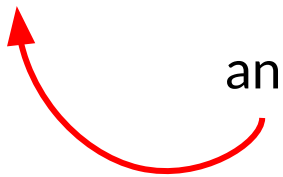
Also note that KL divergence is non-negative: $\mathrm{KL}\left[q \parallel p\right] \geq 0$

Which implies the following relation:

$$\log Z(\theta) = \mathrm{KL}\left[q(x) \parallel p(x)\right] - J(q) \geq -J(q)$$

$$\Rightarrow \log Z(\theta) \geq -J(q)$$

Therefore, $-J(q)$ is a *lower bound* on the log-partition function, $\log Z(\theta)$.



an important quantity!

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

The Evidence Lower Bound

Recall that, for posterior distributions (of latents given observed), *i.e.*,

(Dropped θ for convenience).

$$p(y \mid x_1, \ldots, x_n) = \frac{p(y, x_1, \ldots, x_n)}{p(x_1, \ldots, x_n)} = \frac{1}{Z}\, p(y, x_1, \ldots, x_n)$$

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

The Evidence Lower Bound

Recall that, for posterior distributions (of latents given observed), *i.e.*,

(Dropped θ for convenience).

$$p(y \mid x_1, \ldots, x_n) = \frac{p(y, x_1, \ldots, x_n)}{p(x_1, \ldots, x_n)} = \frac{1}{Z}\, p(y, x_1, \ldots, x_n)$$

the log-partition function is $Z = p(x_1, \ldots, x_n)$, often called the **evidence**.

(*or "marginal log-likelihood"*)

The Evidence Lower Bound

Recall that, for posterior distributions (of latents given observed), *i.e.*,

(Dropped θ for convenience).

$$p(y \mid x_1, \ldots, x_n) = \frac{p(y, x_1, \ldots, x_n)}{p(x_1, \ldots, x_n)} = \frac{1}{Z}\, p(y, x_1, \ldots, x_n)$$

the log-partition function is $Z = p(x_1, \ldots, x_n)$, often called the **evidence**.

(*or "marginal log-likelihood"*)

Thus, $-J(q) \leq \log Z$ is often referred to as the ***evidence lower bound***, or **ELBO**.

The Evidence Lower Bound

To summarize, the ELBO is:

The Evidence Lower Bound

To summarize, the ELBO is:

For discrete distributions

$$-J(q) = -\sum_x q(x) \log \frac{q(x)}{\tilde{p}(x)} = \sum_x q(x) \log \frac{\tilde{p}(x)}{q(x)} = \mathbb{E}_{q(x)} \left[\log \frac{\tilde{p}(x)}{q(x)} \right]$$

The Evidence Lower Bound

To summarize, the ELBO is:

For discrete distributions

$$-J(q) = -\sum_x q(x) \log \frac{q(x)}{\tilde{p}(x)} = \sum_x q(x) \log \frac{\tilde{p}(x)}{q(x)} = \mathbb{E}_{q(x)} \left[\log \frac{\tilde{p}(x)}{q(x)} \right]$$

For continuous distributions

$$-J(q) = -\int_{\mathcal{X}} q(x) \log \frac{q(x)}{\tilde{p}(x)} dx = \int_{\mathcal{X}} q(x) \log \frac{\tilde{p}(x)}{q(x)} dx = \mathbb{E}_{q(x)} \left[\log \frac{\tilde{p}(x)}{q(x)} \right]$$

The Evidence Lower Bound

Also, the ELBO is often written in the form:

$$-J(q) = \mathbb{E}_{q(x)}\left[\log \tilde{p}(x) - \log q(x)\right] = \log Z(\theta) - \mathrm{KL}\left[q(x) \parallel p(x)\right]$$
$$\leq \log Z(\theta)$$

The Evidence Lower Bound

Also, the ELBO is often written in the form:

$$-J(q) = \mathbb{E}_{q(x)}\left[\log \tilde{p}(x) - \log q(x)\right] = \log Z(\theta) - \mathrm{KL}\left[q(x) \parallel p(x)\right]$$
$$\leq \log Z(\theta)$$

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

The Evidence Lower Bound

Also, the ELBO is often written in the form:

$$-J(q) = \mathbb{E}_{q(x)}\left[\log \tilde{p}(x) - \log q(x)\right] = \log Z(\theta) - \mathrm{KL}\left[q(x) \parallel p(x)\right]$$
$$\leq \log Z(\theta)$$

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

The Evidence Lower Bound

Also, the ELBO is often written in the form:

$$
\begin{aligned}
-J(q) = \mathbb{E}_{q(x)}\left[\log \tilde{p}(x) - \log q(x)\right] &= \log Z(\theta) - \mathrm{KL}\left[q(x) \parallel p(x)\right] \\
&\leq \log Z(\theta)
\end{aligned}
$$

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

The Evidence Lower Bound

Also, the ELBO is often written in the form:

$$
\begin{aligned}
-J(q) = \mathbb{E}_{q(x)}\left[\log \tilde{p}(x) - \log q(x)\right] &= \log Z(\theta) - \mathrm{KL}\left[q(x) \parallel p(x)\right] \\
&\leq \log Z(\theta)
\end{aligned}
$$

Importantly, note that:

The Evidence Lower Bound

Also, the ELBO is often written in the form:

$$
\begin{aligned}
-J(q) = \mathbb{E}_{q(x)}\left[\log \tilde{p}(x) - \log q(x)\right] &= \log Z(\theta) - \mathrm{KL}\left[q(x) \parallel p(x)\right] \\
&\leq \log Z(\theta)
\end{aligned}
$$

Importantly, note that:

- The difference between $\log Z(\theta)$ and $-J(q)$ is $\mathrm{KL}\left[q \parallel p\right]$.

The Evidence Lower Bound

Also, the ELBO is often written in the form:

$$
\begin{aligned}
-J(q) = \mathbb{E}_{q(x)}\left[\log \tilde{p}(x) - \log q(x)\right] &= \log Z(\theta) - \mathrm{KL}\left[q(x) \parallel p(x)\right] \\
&\leq \log Z(\theta)
\end{aligned}
$$

Importantly, note that:

- The difference between $\log Z(\theta)$ and $-J(q)$ is $\mathrm{KL}\left[q \parallel p\right]$.
- Therefore, by maximizing the ELBO, we are *minimizing* $\mathrm{KL}\left[q \parallel p\right]$...

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

The Evidence Lower Bound

Also, the ELBO is often written in the form:

$$
\begin{aligned}
-J(q) = \mathbb{E}_{q(x)}\left[\log \tilde{p}(x) - \log q(x)\right] &= \log Z(\theta) - \mathrm{KL}\left[q(x) \parallel p(x)\right] \\
&\leq \log Z(\theta)
\end{aligned}
$$

Importantly, note that:

- The difference between $\log Z(\theta)$ and $-J(q)$ is $\mathrm{KL}\left[q \parallel p\right]$.
- Therefore, by maximizing the ELBO, we are *minimizing* $\mathrm{KL}\left[q \parallel p\right]$...
- ... by "*squeezing*" it between $-J(q)$ and $\log Z(\theta)$.

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

A Few Details on KL Divergence

Recall that $\mathrm{KL}\left[q \parallel p\right] \neq \mathrm{KL}\left[p \parallel q\right]$ (*i.e.*, KL divergence is not symmetric).

Both are 0 when $q = p$, but assign different penalties when $q \neq p$.

So why did we choose one over the other, and how do they differ?

A Few Details on KL Divergence

Perhaps the most important reason we typically use $\mathrm{KL}\left[q \parallel p\right]$ is because:

A Few Details on KL Divergence

Perhaps the most important reason we typically use $\mathrm{KL}\left[q \parallel p\right]$ is because:

$$\mathrm{KL}\left[q(x) \parallel p(x)\right] = \mathbb{E}_{q(x)}\left[\log \frac{q(x)}{p(x)}\right]$$

i.e., an expectation with respect to $q(x)$ … rather than $p(x)$.

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

A Few Details on KL Divergence

Perhaps the most important reason we typically use $\mathrm{KL}\left[q \parallel p\right]$ is because:

$$\mathrm{KL}\left[q(x) \parallel p(x)\right] = \mathbb{E}_{q(x)}\left[\log \frac{q(x)}{p(x)}\right]$$

i.e., an expectation with respect to $q(x)$ … rather than $p(x)$.

Useful because $q(x)$ is a known/tractable distribution, *e.g.*, possible to sample from.

While $p(x)$ is unknown/intractable (*e.g.*, normalization unknown).

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

A Few Details on KL Divergence

However, choosing $\mathrm{KL}\left[q \,\|\, p\right]$ in VI affects the optimal distribution!

When the variational family does not contain the true distribution $p(x)$.

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

A Few Details on KL Divergence

$$\mathrm{KL}\left[q(x) \,\|\, p(x)\right] = \mathbb{E}_{q(x)}\left[\log \frac{q(x)}{p(x)}\right]$$

However, choosing $\mathrm{KL}\left[q \,\|\, p\right]$ in VI affects the optimal distribution!

When the variational family does not contain the true distribution $p(x)$.

Note that $\mathrm{KL}\left[q \,\|\, p\right]$ grows very large when $q(x) > 0$ and $p(x) \to 0$.

Therefore, in VI, if $p(x) = 0$ it means we must have that $q(x) = 0$.

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

A Few Details on KL Divergence

$$\mathrm{KL}\left[q(x) \,\|\, p(x)\right] = \mathbb{E}_{q(x)}\left[\log \frac{q(x)}{p(x)}\right]$$

However, choosing $\mathrm{KL}\left[q \,\|\, p\right]$ in VI affects the optimal distribution!

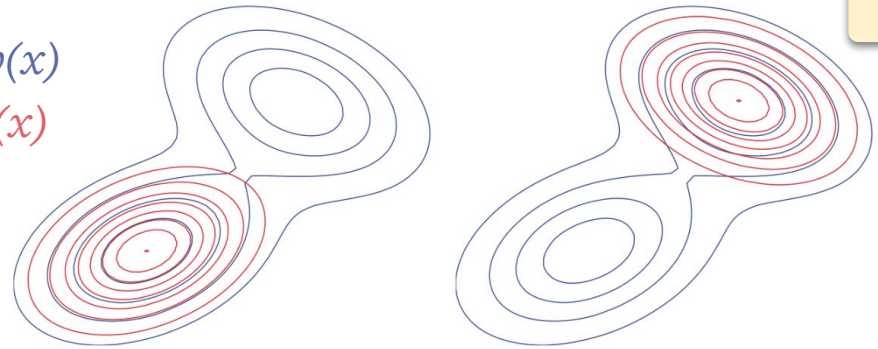
When the variational family does not contain the true distribution $p(x)$.

Note that $\mathrm{KL}\left[q \,\|\, p\right]$ grows very large when $q(x) > 0$ and $p(x) \to 0$.

Therefore, in VI, if $p(x) = 0$ it means we must have that $q(x) = 0$.

We say that $\mathrm{KL}\left[q \,\|\, p\right]$ is "*zero-forcing for $q(x)$*"

Blue - $p(x)$
Red - $q(x)$



$q(x)$ typically underestimates the support of $p(x)$.

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

A Few Details on KL Divergence

$$\mathrm{KL}\left[q(x) \parallel p(x)\right] = \mathbb{E}_{q(x)}\left[\log \frac{q(x)}{p(x)}\right]$$

On the other hand…

A Few Details on KL Divergence

$$\mathrm{KL}\left[p(x) \parallel q(x)\right] = \mathbb{E}_{p(x)}\left[\log \frac{p(x)}{q(x)}\right]$$

On the other hand… consider $\mathrm{KL}\left[p \parallel q\right]$.

Often called "*M-projection*", or "*moment projection*", or just the "*reverse KL*".

A Few Details on KL Divergence

$$\mathrm{KL}\left[p(x) \parallel q(x)\right] = \mathbb{E}_{p(x)}\left[\log \frac{p(x)}{q(x)}\right]$$

On the other hand… consider $\mathrm{KL}\left[p \parallel q\right]$.

Often called "*M-projection*", or "*moment projection*", or just the "*reverse KL*".

Note that $\mathrm{KL}\left[p \parallel q\right]$ grows very large when $q(x) \to 0$ and $p(x) > 0$.

Therefore, in VI, if $p(x) > 0$ it means we must have that $q(x) > 0$.

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

A Few Details on KL Divergence

$$\mathrm{KL}\left[p(x) \parallel q(x)\right] = \mathbb{E}_{p(x)}\left[\log \frac{p(x)}{q(x)}\right]$$

On the other hand… consider $\mathrm{KL}\left[p \parallel q\right]$.

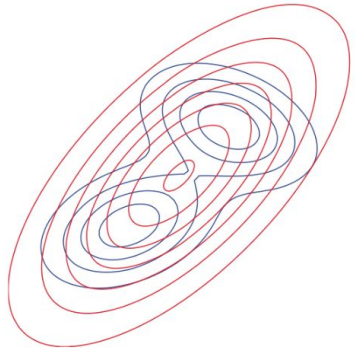
Often called "*M-projection*", or "*moment projection*", or just the "*reverse KL*".

Note that $\mathrm{KL}\left[p \parallel q\right]$ grows very large when $q(x) \to 0$ and $p(x) > 0$.

Therefore, in VI, if $p(x) > 0$ it means we must have that $q(x) > 0$.

We say that $\mathrm{KL}\left[p \parallel q\right]$ is "*zero-avoiding for $q(x)$*"

Blue - $p(x)$
Red - $q(x)$



$q(x)$ typically **over**estimates the support of $p(x)$.

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

A Few Details on KL Divergence

Due to these properties, these are also sometimes called:

Blue - $p(x)$
Red - $q(x)$

$$\mathrm{KL}\left[q \,\|\, p\right]$$

The "*exclusive*" KL divergence.

Blue - $p(x)$
Red - $q(x)$

$$\mathrm{KL}\left[p \,\|\, q\right]$$

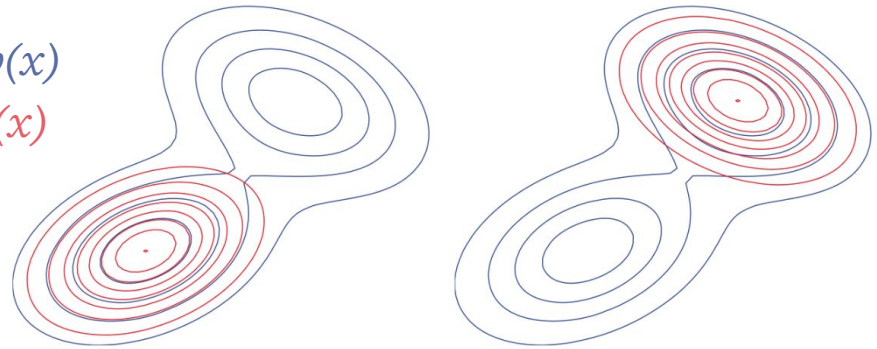
The "*inclusive*" KL divergence.

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

A Few Details on KL Divergence

Due to these properties, these are also sometimes called:

Blue - $p(x)$
Red - $q(x)$



$$\mathrm{KL}\left[q \parallel p\right]$$

The "*exclusive*" KL divergence.

Blue - $p(x)$
Red - $q(x)$

$$\mathrm{KL}\left[p \parallel q\right]$$

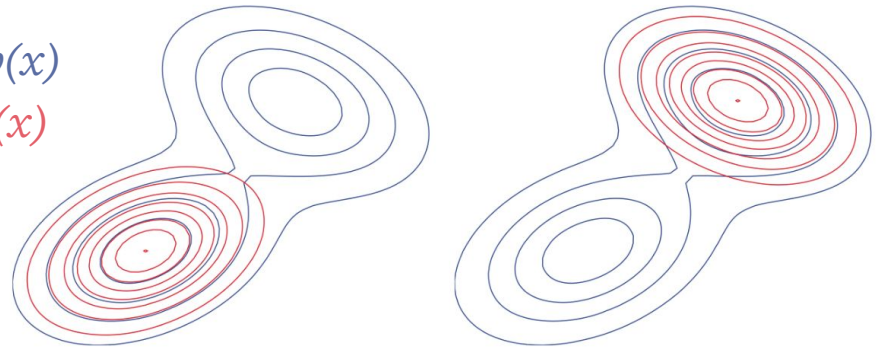
The "*inclusive*" KL divergence.

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

A Few Details on KL Divergence

Due to these properties, these are also sometimes called:

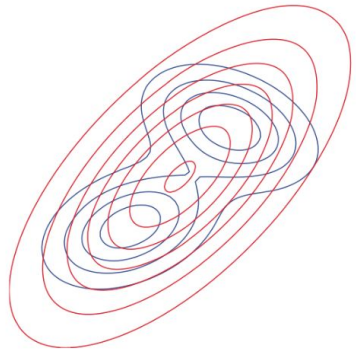
Blue - $p(x)$
Red - $q(x)$



$$\mathrm{KL}\left[q \,\|\, p\right]$$

The "*exclusive*" KL divergence.

Blue - $p(x)$
Red - $q(x)$



$$\mathrm{KL}\left[p \,\|\, q\right]$$

The "*inclusive*" KL divergence.

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

VI with Neural Approximations and the Variational Autoencoder (VAE)

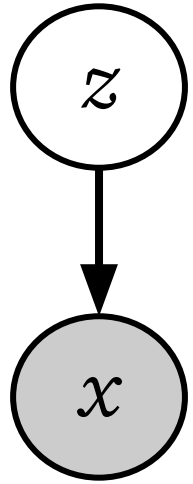
Setup

Setup

For simplicity, consider a latent variable model of the form:

$$p_\theta(x, z) = p_\theta(x \mid z) p_\theta(z)$$

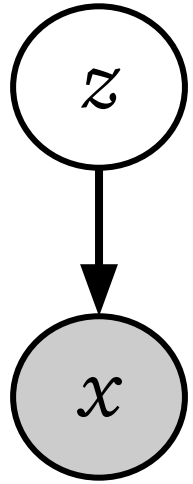
Example PGM



Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Setup

Example PGM



For simplicity, consider a latent variable model of the form:

$$p_\theta(x, z) = p_\theta(x \mid z) p_\theta(z)$$

where:

- $x \in \mathcal{X}$ are observed variables.
- ($\mathcal{X}$ can be either discrete or continuous.)
- $z \in \mathbb{R}^k$ are latent variables.

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

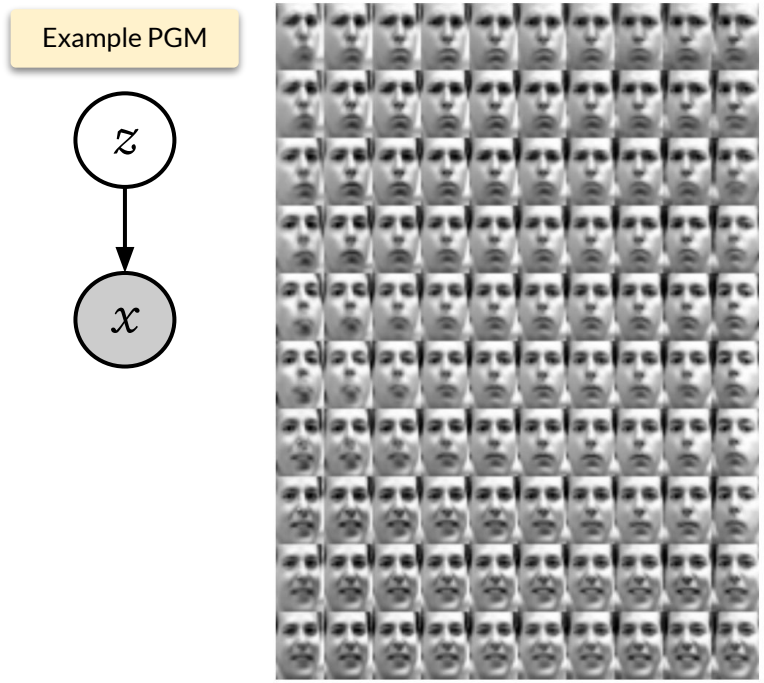
Setup

As an example, for intuition, you can imagine:

Setup

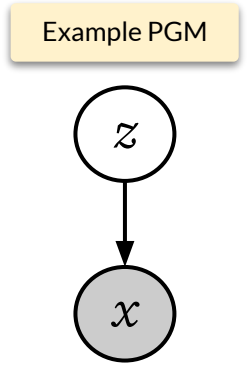
As an example, for intuition, you can imagine:

- x to be an image (*e.g.*, of a human face).



Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Setup



As an example, for intuition, you can imagine:

- x to be an image (*e.g.*, of a human face).
- z to be latent factors, which explain features of the face, *e.g.*:
  - One coordinate could encode whether the face is happy or sad.
  - One coordinate could encode whether the face has short vs long hair.
  - One coordinate could encode whether the face has a beard vs clean shaven.
  - etc.

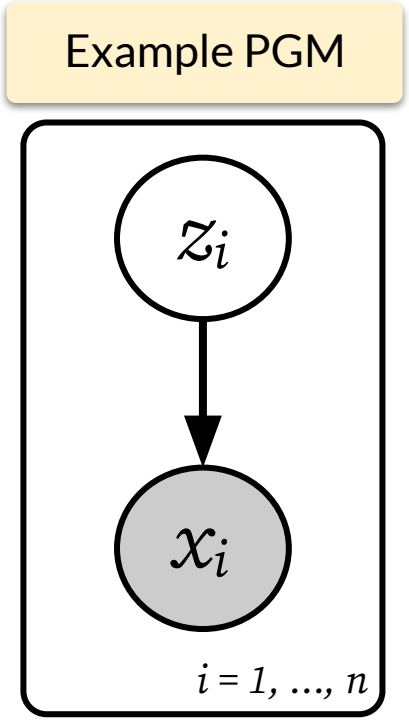
Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Setup

Suppose now that we have a given dataset: $D = \{x^1, x^2, \ldots, x^n\}$

Setup

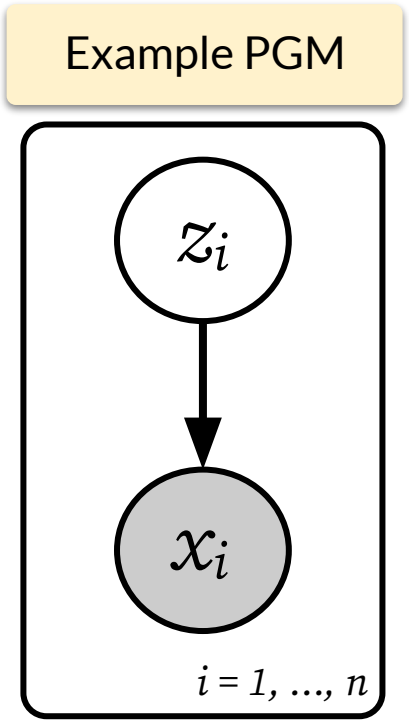
Suppose now that we have a given dataset: $D = \{x^1, x^2, \ldots, x^n\}$



Setup

Suppose now that we have a given dataset: $D = \{x^1, x^2, \ldots, x^n\}$

We are interested in the following inference and learning tasks:



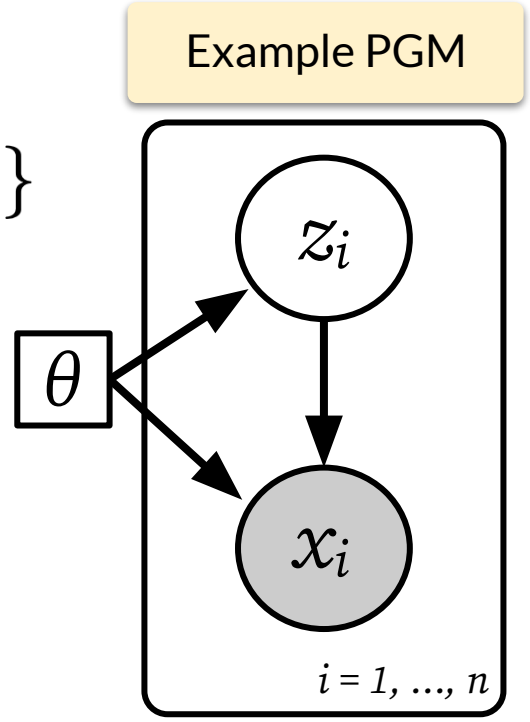
Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Setup

Suppose now that we have a given dataset: $D = \{x^1, x^2, \ldots, x^n\}$

We are interested in the following inference and learning tasks:

(1) Learning the parameters θ of our model p_θ .



Setup

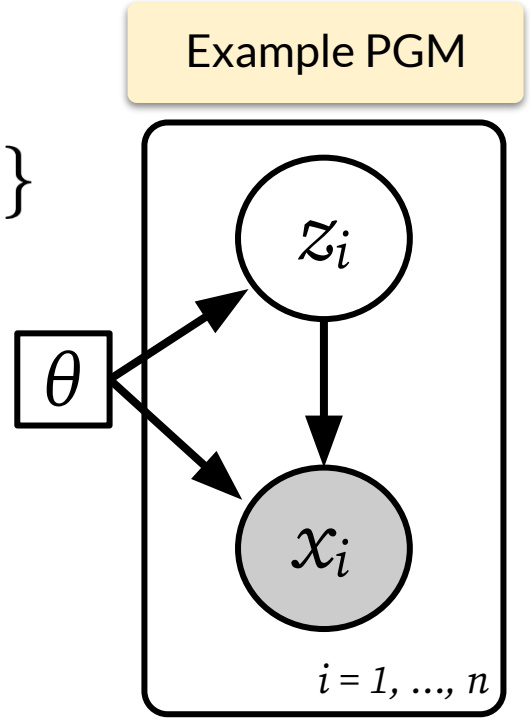
Suppose now that we have a given dataset: $D = \{x^1, x^2, \ldots, x^n\}$

We are interested in the following inference and learning tasks:

(1) Learning the parameters θ of our model p_θ.

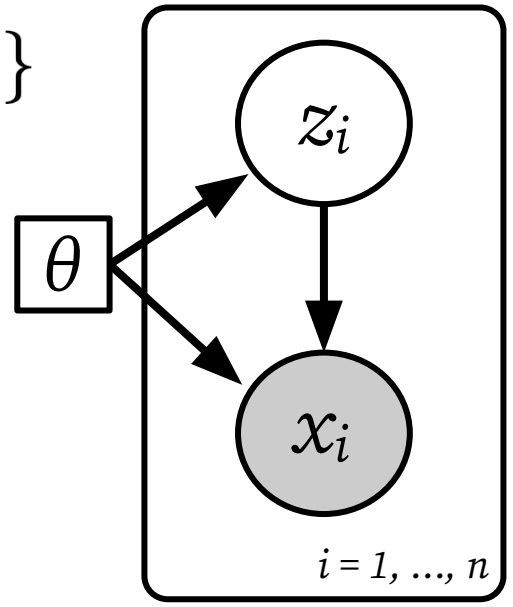
(2) Approximate posterior inference over z:

Given an image x, what are it's latent factors, *i.e.*, what is $p_\theta(z \mid x)$?



Setup

Example PGM



Suppose now that we have a given dataset: $D = \{x^1, x^2, \ldots, x^n\}$

We are interested in the following inference and learning tasks:

(1) Learning the parameters θ of our model p_θ.

(2) Approximate posterior inference over z:

Given an image x, what are it's latent factors, *i.e.*, what is $p_\theta(z \mid x)$?

(3) Approximate marginal inference over x:

Given an image x with missing parts, how do we fill in these parts, *i.e.*, $p_\theta(x_i \mid x_{-i})$?

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Setup

We will make the following two addition assumptions:

Setup

We will make the following two addition assumptions:

- *Intractability*: assume that computing the posterior $p_\theta(z \mid x)$ is intractable (*i.e.*, not able to be computed exactly or analytically)

- *Big dataset*: assume that the dataset D is very large (*e.g.*, too large to fit in memory), and thus we can only work with small, subsampled batches of D.

Setup

We will make the following two addition assumptions:

- *Intractability*: assume that computing the posterior $p_\theta(z \mid x)$ is intractable (*i.e.*, not able to be computed exactly or analytically)

- *Big dataset*: assume that the dataset D is very large (*e.g.*, too large to fit in memory), and thus we can only work with small, subsampled batches of D.

(Many interesting models have these properties; one good example will be the variational autoencoder).

Background: VAE Paper

From ICLR 2014
(>42,000 citations)

Auto-Encoding Variational Bayes

Diederik P. Kingma
Machine Learning Group
Universiteit van Amsterdam
dpkingma@gmail.com

Max Welling
Machine Learning Group
Universiteit van Amsterdam
welling.max@gmail.com

Abstract

How can we perform efficient inference and learning in directed probabilistic models, in the presence of continuous latent variables with intractable posterior distributions, and large datasets? We introduce a stochastic variational inference and learning algorithm that scales to large datasets and, under some mild differentiability conditions, even works in the intractable case. Our contributions are two-fold. First, we show that a reparameterization of the variational lower bound yields a lower bound estimator that can be straightforwardly optimized using standard stochastic gradient methods. Second, we show that for i.i.d. datasets with continuous latent variables per datapoint, posterior inference can be made especially efficient by fitting an approximate inference model (also called a recognition model) to the intractable posterior using the proposed lower bound estimator. Theoretical advantages are reflected in experimental results.

1 Introduction

How can we perform efficient approximate inference and learning with directed probabilistic models whose continuous latent variables and/or parameters have intractable posterior distributions? The variational Bayesian (VB) approach involves the optimization of an approximation to the intractable posterior. Unfortunately, the common mean-field approach requires analytical solutions of expectations w.r.t. the approximate posterior, which are also intractable in the general case. We show how a reparameterization of the variational lower bound yields a simple differentiable unbiased estimator of the lower bound; this SGVB (Stochastic Gradient Variational Bayes) estimator can be used for efficient approximate posterior inference in almost any model with continuous latent variables and/or parameters, and is straightforward to optimize using standard stochastic gradient ascent techniques.

For the case of an i.i.d. dataset and continuous latent variables per datapoint, we propose the Auto-Encoding VB (AEVB) algorithm. In the AEVB algorithm we make inference and learning especially efficient by using the SGVB estimator to optimize a recognition model that allows us to perform very efficient approximate posterior inference using simple ancestral sampling, which in turn allows us to efficiently learn the model parameters, without the need of expensive iterative inference schemes (such as MCMC) per datapoint. The learned approximate posterior inference model can also be used for a host of tasks such as recognition, denoising, representation and visualization purposes. When a neural network is used for the recognition model, we arrive at the *variational auto-encoder*.

2 Method

The strategy in this section can be used to derive a lower bound estimator (a stochastic objective function) for a variety of directed graphical models with continuous latent variables. We will restrict ourselves here to the common case where we have an i.i.d. dataset with latent variables per datapoint, and where we like to perform maximum likelihood (ML) or maximum a posteriori (MAP) inference on the (global) parameters, and variational inference on the latent variables. It is, for example,

Background: VAE Paper

From ICLR 2014
(>42,000 citations)

Introduces two distinct things:

(1) *Auto-encoding variational Bayes* (AEVB) algorithm.
(a stochastic-gradient VI procedure)

(2) A special case (involving neural networks): the *variational autoencoder* (VAE).

Auto-Encoding Variational Bayes

Diederik P. Kingma
Machine Learning Group
Universiteit van Amsterdam
dpkingma@gmail.com

Max Welling
Machine Learning Group
Universiteit van Amsterdam
welling.max@gmail.com

Abstract

How can we perform efficient inference and learning in directed probabilistic models, in the presence of continuous latent variables with intractable posterior distributions, and large datasets? We introduce a stochastic variational inference and learning algorithm that scales to large datasets and, under some mild differentiability conditions, even works in the intractable case. Our contributions are two-fold. First, we show that a reparameterization of the variational lower bound yields a lower bound estimator that can be straightforwardly optimized using standard stochastic gradient methods. Second, we show that for i.i.d. datasets with continuous latent variables per datapoint, posterior inference can be made especially efficient by fitting an approximate inference model (also called a recognition model) to the intractable posterior using the proposed lower bound estimator. Theoretical advantages are reflected in experimental results.

1 Introduction

How can we perform efficient approximate inference and learning with directed probabilistic models whose continuous latent variables and/or parameters have intractable posterior distributions? The variational Bayesian (VB) approach involves the optimization of an approximation to the intractable posterior. Unfortunately, the common mean-field approach requires analytical solutions of expectations w.r.t. the approximate posterior, which are also intractable in the general case. We show how a reparameterization of the variational lower bound yields a simple differentiable unbiased estimator of the lower bound; this SGVB (Stochastic Gradient Variational Bayes) estimator can be used for efficient approximate posterior inference in almost any model with continuous latent variables and/or parameters, and is straightforward to optimize using standard stochastic gradient ascent techniques.

For the case of an i.i.d. dataset and continuous latent variables per datapoint, we propose the Auto-Encoding VB (AEVB) algorithm. In the AEVB algorithm we make inference and learning especially efficient by using the SGVB estimator to optimize a recognition model that allows us to perform very efficient approximate posterior inference using simple ancestral sampling, which in turn allows us to efficiently learn the model parameters, without the need of expensive iterative inference schemes (such as MCMC) per datapoint. The learned approximate posterior inference model can also be used for a host of tasks such as recognition, denoising, representation and visualization purposes. When a neural network is used for the recognition model, we arrive at the *variational auto-encoder*.

2 Method

The strategy in this section can be used to derive a lower bound estimator (a stochastic objective function) for a variety of directed graphical models with continuous latent variables. We will restrict ourselves here to the common case where we have an i.i.d. dataset with latent variables per datapoint, and where we like to perform maximum likelihood (ML) or maximum a posteriori (MAP) inference on the (global) parameters, and variational inference on the latent variables. It is, for example,

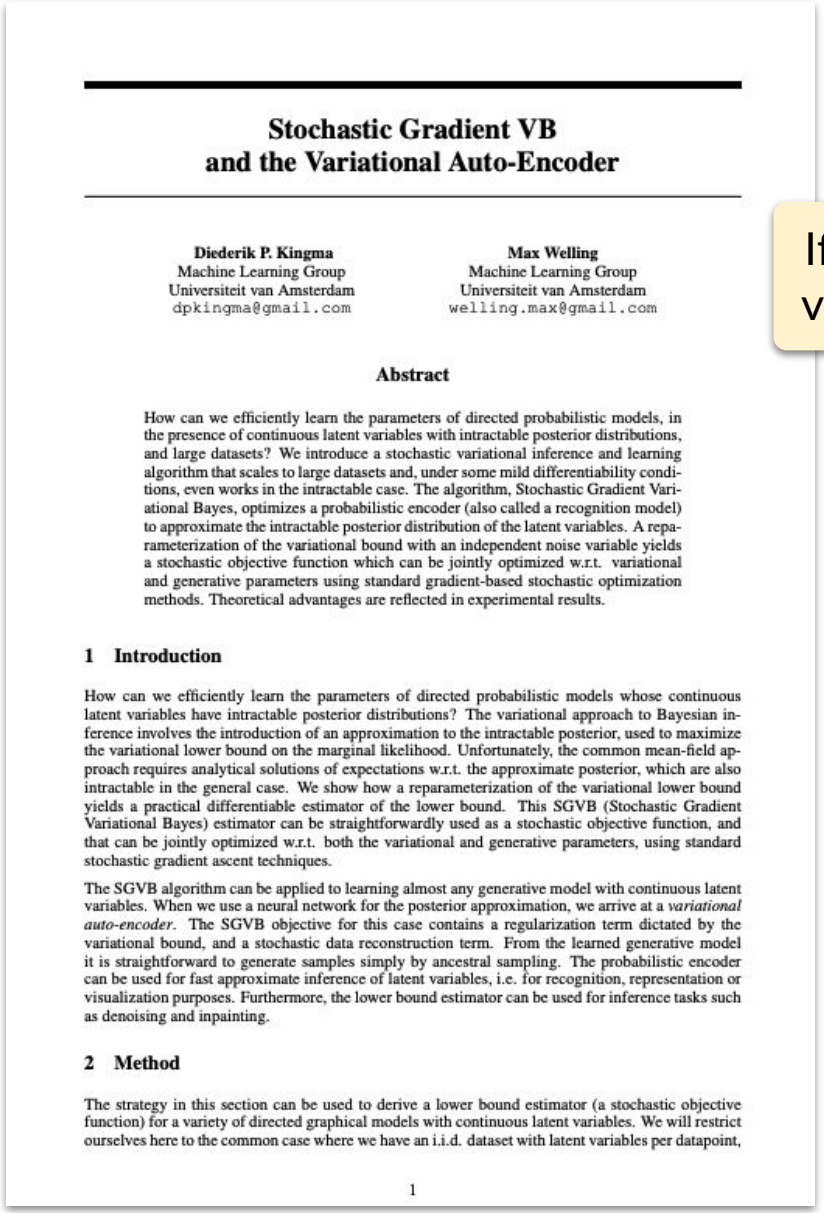
Background: VAE Paper

From ICLR 2014
(>42,000 citations)

Introduces two distinct things:

(1) *Auto-encoding variational Bayes* (AEVB) algorithm.
(a stochastic-gradient VI procedure)

(2) A special case (involving neural networks): the *variational autoencoder* (VAE).



Stochastic Gradient VB and the Variational Auto-Encoder

Diederik P. Kingma
Machine Learning Group
Universiteit van Amsterdam
dpkingma@gmail.com

Max Welling
Machine Learning Group
Universiteit van Amsterdam
welling.max@gmail.com

Abstract

How can we efficiently learn the parameters of directed probabilistic models, in the presence of continuous latent variables with intractable posterior distributions, and large datasets? We introduce a stochastic variational inference and learning algorithm that scales to large datasets and, under some mild differentiability conditions, even works in the intractable case. The algorithm, Stochastic Gradient Variational Bayes, optimizes a probabilistic encoder (also called a recognition model) to approximate the intractable posterior distribution of the latent variables. A reparameterization of the variational bound with an independent noise variable yields a stochastic objective function which can be jointly optimized w.r.t. variational and generative parameters using standard gradient-based stochastic optimization methods. Theoretical advantages are reflected in experimental results.

1 Introduction

How can we efficiently learn the parameters of directed probabilistic models whose continuous latent variables have intractable posterior distributions? The variational approach to Bayesian inference involves the introduction of an approximation to the intractable posterior, used to maximize the variational lower bound on the marginal likelihood. Unfortunately, the common mean-field approach requires analytical solutions of expectations w.r.t. the approximate posterior, which are also intractable in the general case. We show how a reparameterization of the variational lower bound yields a practical differentiable estimator of the lower bound. This SGVB (Stochastic Gradient Variational Bayes) estimator can be straightforwardly used as a stochastic objective function, and that can be jointly optimized w.r.t. both the variational and generative parameters, using standard stochastic gradient ascent techniques.

The SGVB algorithm can be applied to learning almost any generative model with continuous latent variables. When we use a neural network for the posterior approximation, we arrive at a *variational auto-encoder*. The SGVB objective for this case contains a regularization term dictated by the variational bound, and a stochastic data reconstruction term. From the learned generative model it is straightforward to generate samples simply by ancestral sampling. The probabilistic encoder can be used for fast approximate inference of latent variables, i.e. for recognition, representation or visualization purposes. Furthermore, the lower bound estimator can be used for inference tasks such as denoising and inpainting.

2 Method

The strategy in this section can be used to derive a lower bound estimator (a stochastic objective function) for a variety of directed graphical models with continuous latent variables. We will restrict ourselves here to the common case where we have an i.i.d. dataset with latent variables per datapoint,

1

If you look at older versions on arXiv…

Auto-encoding Variational Bayes (AEVB)

Auto-encoding Variational Bayes (AEVB)

The Auto-encoding Variational Bayes (AEVB) algorithm aims to solve the tasks from before, in particular:

Auto-encoding Variational Bayes (AEVB)

The Auto-encoding Variational Bayes (AEVB) algorithm aims to solve the tasks from before, in particular:

(1) Learning the parameters θ of our model p_θ.

(2) Approximate posterior inference over z: $p_\theta(z \mid x)$

We'll start with methods for approximate posterior inference, and then show how this can lead to learning the parameters (and thus to generative modeling!)

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Auto-encoding Variational Bayes (AEVB)

Suppose we have a family of approximations for our model posterior $p_\theta(z \mid x)$, written $q_\phi(z \mid x)$, which is parameterized by some parameter ϕ.

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Auto-encoding Variational Bayes (AEVB)

Suppose we have a family of approximations for our model posterior $p_\theta(z \mid x)$, written $q_\phi(z \mid x)$, which is parameterized by some parameter ϕ.

A note on the family of approximations $q_\phi(z \mid x)$:

- We are assuming that x is some fixed observation (*i.e.*, set before we do any inference).
- So, if we want, we can define a family of approximations as a function of x.
- This effectively yields a different variational approximation $q_\phi(z)$ for each x.

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Auto-encoding Variational Bayes (AEVB)

Suppose we have a family of approximations for our model posterior $p_\theta(z \mid x)$, written $q_\phi(z \mid x)$, which is parameterized by some parameter ϕ.

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Auto-encoding Variational Bayes (AEVB)

Suppose we have a family of approximations for our model posterior $p_\theta(z \mid x)$, written $q_\phi(z \mid x)$, which is parameterized by some parameter ϕ.

Consider an ELBO for this model, written as:

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

Auto-encoding Variational Bayes (AEVB)

Suppose we have a family of approximations for our model posterior $p_\theta(z \mid x)$, written $q_\phi(z \mid x)$, which is parameterized by some parameter ϕ.

Consider an ELBO for this model, written as:

This is $-J(q)$ from before!

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

Auto-encoding Variational Bayes (AEVB)

Suppose we have a family of approximations for our model posterior $p_\theta(z \mid x)$, written $q_\phi(z \mid x)$, which is parameterized by some parameter ϕ.

Consider an ELBO for this model, written as:

This is $-J(q)$ from before!

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

Note that the ELBO satisfies:

$$\log p_\theta(x) = \mathrm{KL}\left[q_\phi(z \mid x) || p_\theta(z \mid x)\right] + \mathcal{L}(p_\theta, q_\phi).$$

Evidence equals KL divergence plus ELBO.

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Auto-encoding Variational Bayes (AEVB)

How can we optimize the ELBO to find the optimal approximation $q_\phi(z \mid x)$?

Auto-encoding Variational Bayes (AEVB)

How can we optimize the ELBO to find the optimal approximation $q_\phi(z \mid x)$?

⇒ We'll use a *black-box variational inference* approach.

This means a VI method that aims to be automatic / applied in a general purpose way.

Auto-encoding Variational Bayes (AEVB)

How can we optimize the ELBO to find the optimal approximation $q_\phi(z \mid x)$?

⇒ We'll use a *black-box variational inference* approach.

This means a VI method that aims to be automatic / applied in a general purpose way.

In particular:

- Maximize the ELBO via gradient descent over ϕ.
- Simply requires q_ϕ to be differentiable with respect to ϕ.

Auto-encoding Variational Bayes (AEVB)

Furthermore

- Instead of just inference, we'll simultaneously do learning of parameters θ.

Auto-encoding Variational Bayes (AEVB)

Furthermore

- Instead of just inference, we'll simultaneously do learning of parameters θ.
- ⇒ Via gradient descent over both ϕ and θ.

Auto-encoding Variational Bayes (AEVB)

Furthermore

- Instead of just inference, we'll simultaneously do learning of parameters θ.
- ⇒ Via gradient descent over both ϕ and θ.
- Optimization of ϕ keeps the ELBO tight around $\log p(x)$ (*i.e.*, low KL divergence).

Auto-encoding Variational Bayes (AEVB)

Furthermore

- Instead of just inference, we'll simultaneously do learning of parameters θ.
- ⇒ Via gradient descent over both ϕ and θ.
- Optimization of ϕ keeps the ELBO tight around $\log p(x)$ (*i.e.*, low KL divergence).
- Optimization of θ keeps pushing the upper bound (*i.e.*, the evidence $\log p(x)$) up.
  - ⇒ Higher likelihood, better learning, and and improved generative model.

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Auto-encoding Variational Bayes (AEVB)

Furthermore

- Instead of just inference, we'll simultaneously do learning of parameters θ.
- ⇒ Via gradient descent over both ϕ and θ.
- Optimization of ϕ keeps the ELBO tight around $\log p(x)$ (*i.e.*, low KL divergence).
- Optimization of θ keeps pushing the upper bound (*i.e.*, the evidence $\log p(x)$) up.
  - ⇒ Higher likelihood, better learning, and and improved generative model.

This is somewhat similar to the expectation-maximization (EM) algorithm…

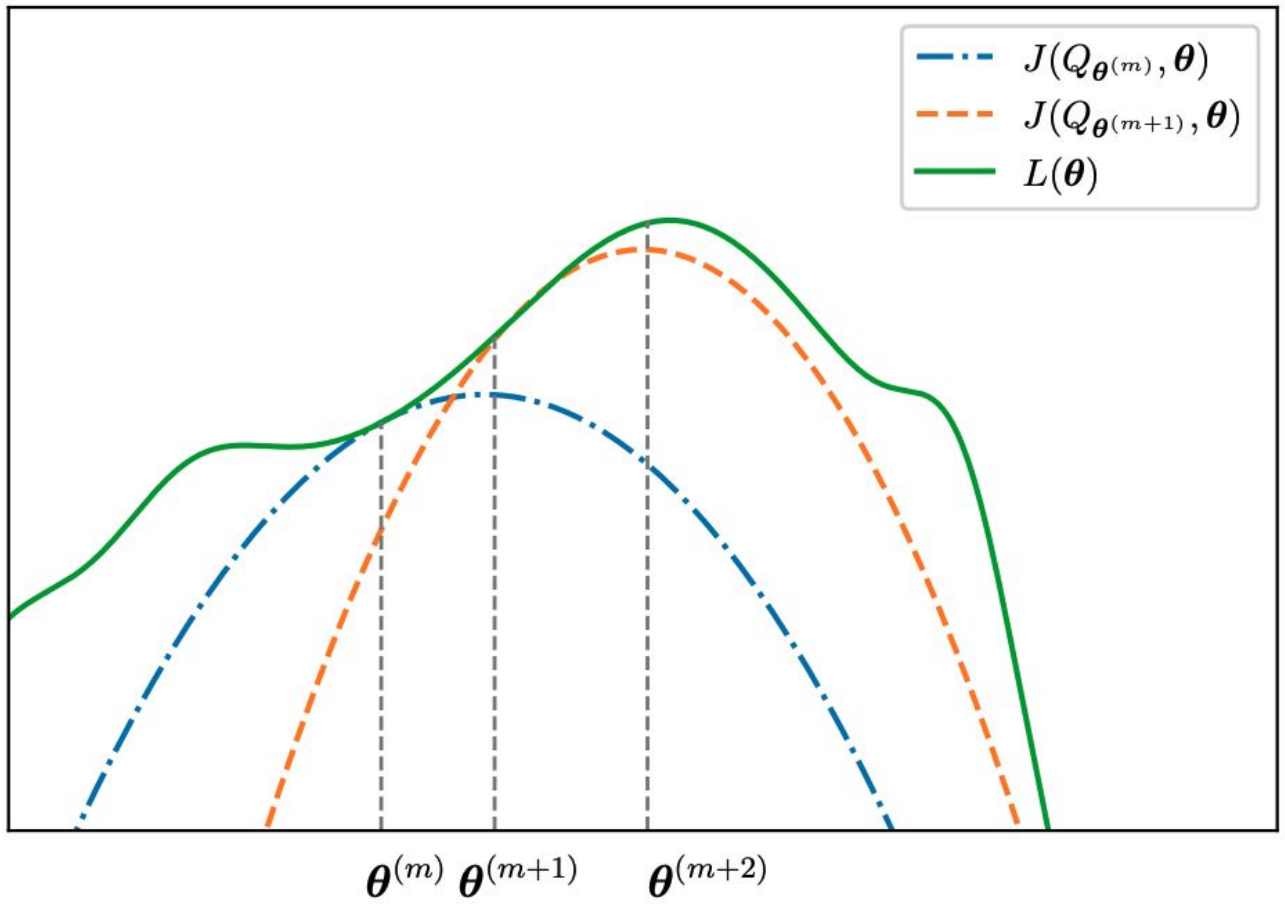
Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Auto-encoding Variational Bayes (AEVB)

Recall the EM algorithm:

Auto-encoding Variational Bayes (AEVB)

Recall the EM algorithm:

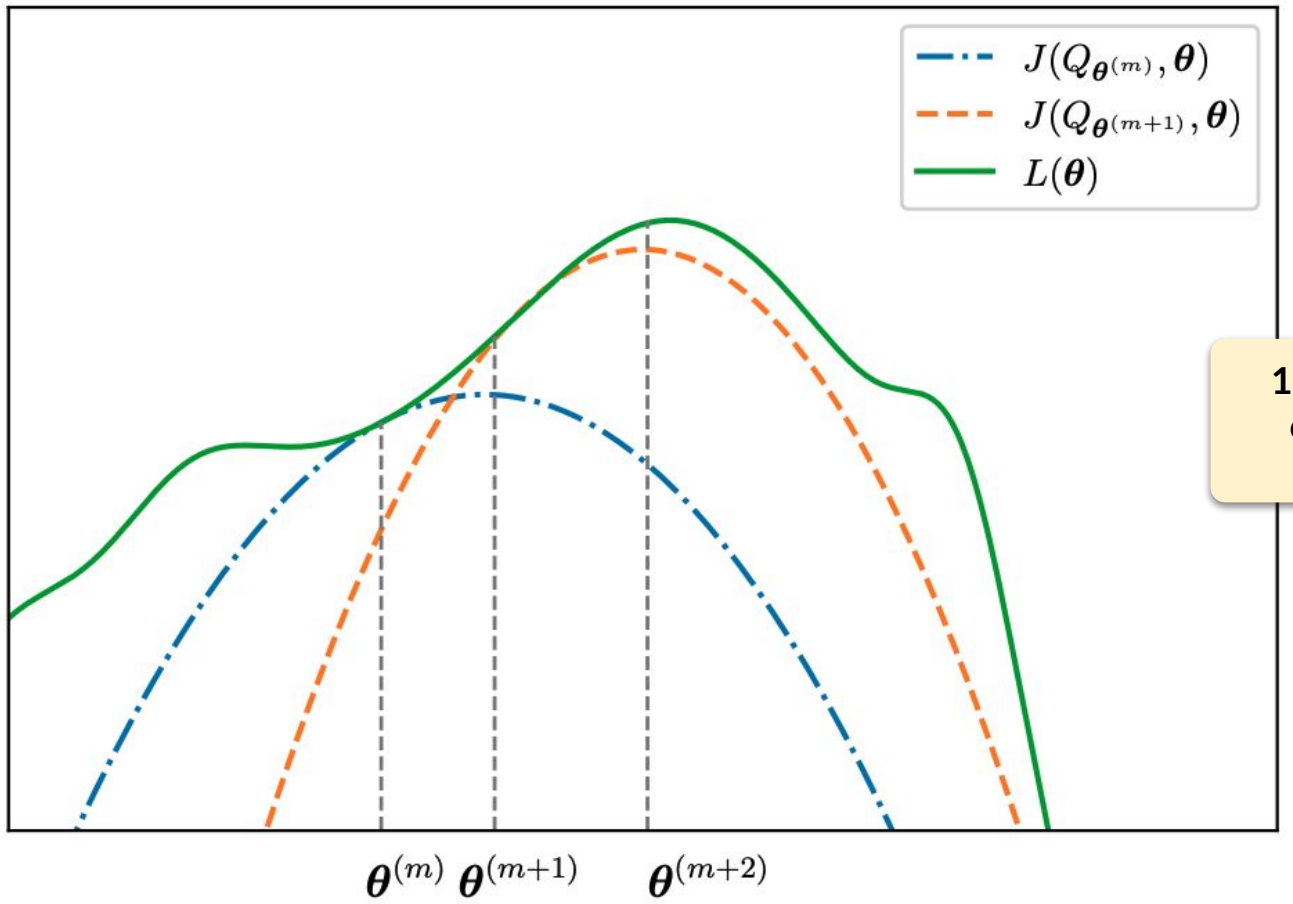


Iteratively forming then optimizing lower bounds to the (*marginal) likelihood* in latent-variable models.

Source: Inspired by Herman Kamper

Auto-encoding Variational Bayes (AEVB)

Recall the EM algorithm:



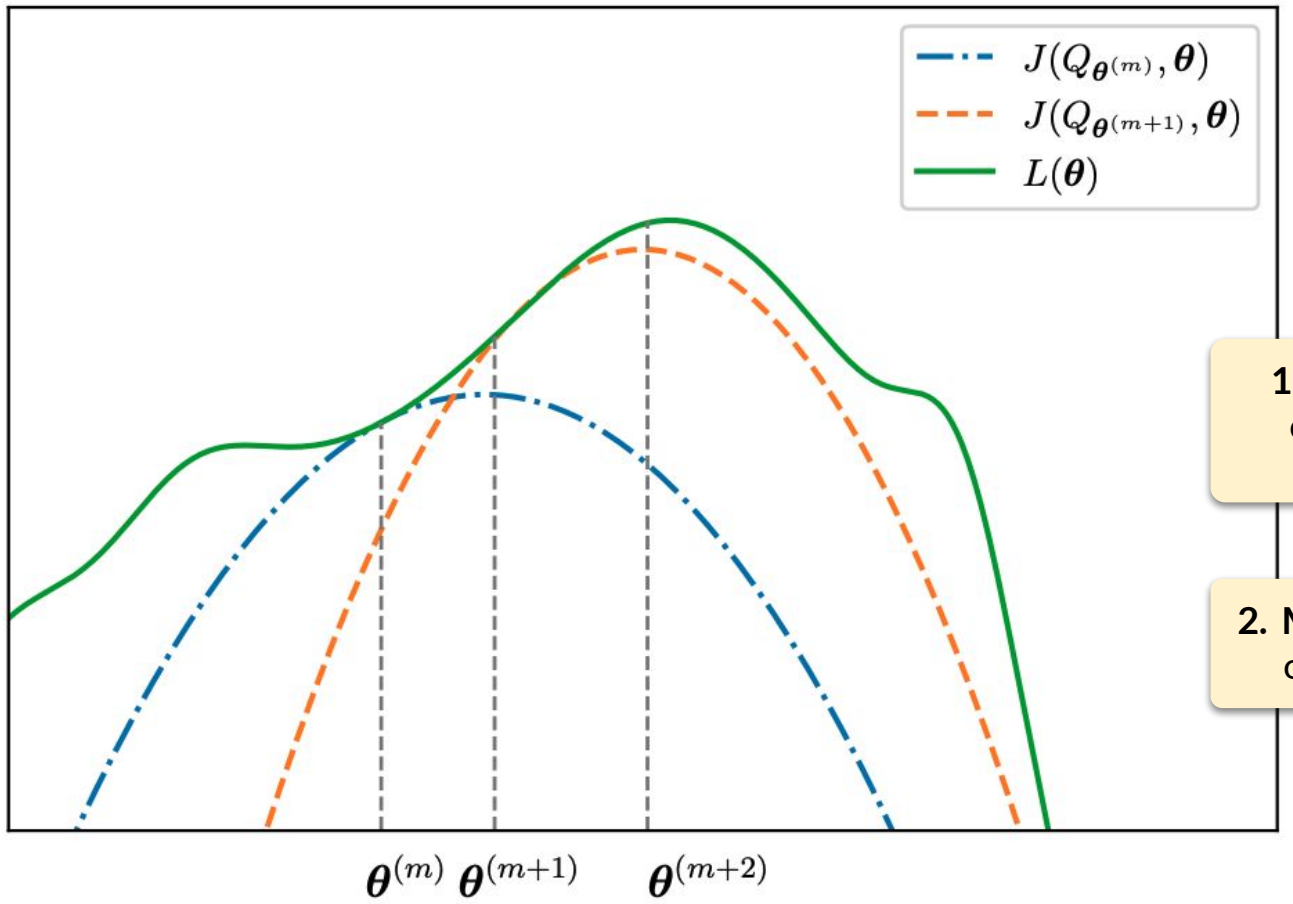
Iteratively forming then optimizing lower bounds to the (*marginal) likelihood* in latent-variable models.

1. **E-Step:** do inference to compute next objective (next lower bound)

Source: Inspired by Herman Kamper

Auto-encoding Variational Bayes (AEVB)

Recall the EM algorithm:



Iteratively forming then optimizing lower bounds to the (*marginal) likelihood* in latent-variable models.

1. **E-Step:** do inference to compute next objective (next lower bound)

2. **M-Step:** maximize this new objective (lower bound).

Source: Inspired by Herman Kamper

Computing the Gradient

Computing the Gradient

So to optimize the ELBO, we need to compute the gradient:

Dropping the conditioning of q on x for now to keep it general.

$$\nabla_{\theta,\phi}\mathbb{E}_{q_\phi(z)}\left[\log p_\theta(x,z) - \log q_\phi(z)\right]$$

Computing the Gradient

So to optimize the ELBO, we need to compute the gradient:

Dropping the conditioning of q on x for now to keep it general.

$$\nabla_{\theta,\phi}\mathbb{E}_{q_\phi(z)}\left[\log p_\theta(x,z) - \log q_\phi(z)\right]$$

Note that an expression for this expectation in closed form is often *not possible*.

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Computing the Gradient

So to optimize the ELBO, we need to compute the gradient:

Dropping the conditioning of q on x for now to keep it general.

$$\nabla_{\theta,\phi}\mathbb{E}_{q_\phi(z)}\left[\log p_\theta(x,z) - \log q_\phi(z)\right]$$

Note that an expression for this expectation in closed form is often *not possible*.

Instead, we could take gradient of a Monte Carlo estimate by sampling z from q_ϕ .

Computing the Gradient

So to optimize the ELBO, we need to compute the gradient:

Dropping the conditioning of q on x for now to keep it general.

$$\nabla_{\theta,\phi}\mathbb{E}_{q_\phi(z)}\left[\log p_\theta(x,z) - \log q_\phi(z)\right]$$

If we do this (gradient of MC estimate), note it is easy to write out the gradient for p_θ:

$$\nabla_\theta\mathbb{E}_{q_\phi(z)}\left[\log p_\theta(x,z) - \log q_\phi(z)\right] = \mathbb{E}_{q_\phi(z)}\left[\nabla_\theta \log p_\theta(x,z)\right]$$

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Computing the Gradient

So to optimize the ELBO, we need to compute the gradient:

Dropping the conditioning of q on x for now to keep it general.

$$\nabla_{\theta,\phi}\mathbb{E}_{q_\phi(z)}\left[\log p_\theta(x,z) - \log q_\phi(z)\right]$$

If we do this (gradient of MC estimate), note it is easy to write out the gradient for p_θ:

$$\nabla_{\theta}\mathbb{E}_{q_\phi(z)}\left[\log p_\theta(x,z) - \log q_\phi(z)\right] = \mathbb{E}_{q_\phi(z)}\left[\nabla_\theta \log p_\theta(x,z)\right]$$

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Computing the Gradient

So to optimize the ELBO, we need to compute the gradient:

Dropping the conditioning of q on x for now to keep it general.

$$\nabla_{\theta,\phi}\mathbb{E}_{q_\phi(z)}\left[\log p_\theta(x,z) - \log q_\phi(z)\right]$$

If we do this (gradient of MC estimate), note it is easy to write out the gradient for p_θ:

$$\nabla_\theta\mathbb{E}_{q_\phi(z)}\left[\log p_\theta(x,z) - \log q_\phi(z)\right] = \mathbb{E}_{q_\phi(z)}\left[\nabla_\theta \log p_\theta(x,z)\right]$$

Just swap the gradient and expectation here.

Computing the Gradient

So to optimize the ELBO, we need to compute the gradient:

Dropping the conditioning of q on x for now to keep it general.

$$\nabla_{\theta,\phi}\mathbb{E}_{q_\phi(z)}\left[\log p_\theta(x,z) - \log q_\phi(z)\right]$$

But it's harder to write out the gradient for q_ϕ :

$$\nabla_{\phi}\mathbb{E}_{q_\phi(z)}\left[\log p_\theta(x,z) - \log q_\phi(z)\right] = \ ??$$

Computing the Gradient

So to optimize the ELBO, we need to compute the gradient:

Dropping the conditioning of q on x for now to keep it general.

$$\nabla_{\theta,\phi}\mathbb{E}_{q_\phi(z)}\left[\log p_\theta(x,z) - \log q_\phi(z)\right]$$

But it's harder to write out the gradient for q_ϕ :

$$\nabla_{\phi}\mathbb{E}_{q_\phi(z)}\left[\log p_\theta(x,z) - \log q_\phi(z)\right] = \ ??$$

Note that we cannot just swap the gradient and expectation, because the expectation is being taken with respect to $q_\phi(z)$.

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

The Score Function Gradient Estimator

One way to estimate this gradient is via the **score function estimator**:

The Score Function Gradient Estimator

One way to estimate this gradient is via the **score function estimator**:

$$\nabla_\phi \mathbb{E}_{q_\phi(z)} \left[\log p_\theta(x, z) - \log q_\phi(z) \right] =$$

$$\mathbb{E}_{q_\phi(z)} \left[\left(\log p_\theta(x, z) - \log q_\phi(z) \right) \nabla_\phi \log q_\phi(z) \right]$$

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

The Score Function Gradient Estimator

One way to estimate this gradient is via the **score function estimator**:

$$\nabla_\phi \mathbb{E}_{q_\phi(z)} \left[\log p_\theta(x, z) - \log q_\phi(z) \right] =$$

$$\mathbb{E}_{q_\phi(z)} \left[\left(\log p_\theta(x, z) - \log q_\phi(z) \right) \nabla_\phi \log q_\phi(z) \right]$$

Expectation with respect to q of…

The Score Function Gradient Estimator

One way to estimate this gradient is via the **score function estimator**:

$$\nabla_\phi \mathbb{E}_{q_\phi(z)} \left[\log p_\theta(x, z) - \log q_\phi(z) \right] =$$

$$\mathbb{E}_{q_\phi(z)} \left[\left(\log p_\theta(x, z) - \log q_\phi(z) \right) \nabla_\phi \log q_\phi(z) \right]$$

Expectation with respect to q of…

Log joint minus log q (same as in the ELBO)

The Score Function Gradient Estimator

One way to estimate this gradient is via the **score function estimator**:

$$\nabla_\phi \mathbb{E}_{q_\phi(z)} \left[\log p_\theta(x, z) - \log q_\phi(z) \right] =$$

$$\mathbb{E}_{q_\phi(z)} \left[\left(\log p_\theta(x, z) - \log q_\phi(z) \right) \nabla_\phi \log q_\phi(z) \right]$$

Expectation with respect to q of…

Log joint minus log q (same as in the ELBO)

Multiplied by the gradient of the log of q.

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

The Score Function Gradient Estimator

One way to estimate this gradient is via the **score function estimator**:

$$\nabla_\phi \mathbb{E}_{q_\phi(z)} \left[\log p_\theta(x, z) - \log q_\phi(z) \right] =$$

$$\mathbb{E}_{q_\phi(z)} \left[\left(\log p_\theta(x, z) - \log q_\phi(z) \right) \nabla_\phi \log q_\phi(z) \right]$$

Expectation with respect to q of…

Log joint minus log q (same as in the ELBO)

Multiplied by the gradient of the log of q.

Follows from some basic algebra/calculus, and takes about half a page to derive :D.

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

The Score Function Gradient Estimator

One way to estimate this gradient is via the **score function estimator**:

$$\nabla_\phi \mathbb{E}_{q_\phi(z)} \left[\log p_\theta(x, z) - \log q_\phi(z) \right] =$$

$$\mathbb{E}_{q_\phi(z)} \left[\left(\log p_\theta(x, z) - \log q_\phi(z) \right) \nabla_\phi \log q_\phi(z) \right]$$

Notably: since the gradient is now inside the expectation, we can evaluate this expectation using Monte Carlo methods!

Expectation with respect to q of…

Log joint minus log q (same as in the ELBO)

Multiplied by the gradient of the log of q.

Follows from some basic algebra/calculus, and takes about half a page to derive :D.

⇒ Now can estimate via Monte Carlo methods.

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

The Score Function Gradient Estimator

However, there is an issue:

The Score Function Gradient Estimator

However, there is an issue:

- The score-function estimator has an important shortcoming: it has **high variance**.

The Score Function Gradient Estimator

However, there is an issue:

- The score-function estimator has an important shortcoming: it has **high variance**.
- What does this mean?
    - Suppose we are using Monte Carlo to estimate a quantity with expected value = 1.

The Score Function Gradient Estimator

However, there is an issue:

- The score-function estimator has an important shortcoming: it has **high variance**.
- What does this mean?
  - Suppose we are using Monte Carlo to estimate a quantity with expected value = 1.
  - If our samples are: $0.9, 1.1, 0.96, 1.05$ (*i.e.*, close to 1), then after a few samples we'll get a good estimate of the true expected value.

The Score Function Gradient Estimator

However, there is an issue:

- The score-function estimator has an important shortcoming: it has **high variance**.
- What does this mean?
  - Suppose we are using Monte Carlo to estimate a quantity with expected value = 1.
  - If our samples are: $0.9, 1.1, 0.96, 1.05$ (*i.e.*, close to 1), then after a few samples we'll get a good estimate of the true expected value.
  - If our samples are: zero 99% of the time, and one-hundred 1% of the time ⇒ expected value is correct, but estimate is worse! (You need many more samples).

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

The Score Function Gradient Estimator

However, there is an issue:

- The score-function estimator has an important shortcoming: it has **high variance**.
- What does this mean?
  - Suppose we are using Monte Carlo to estimate a quantity with expected value = 1.
  - If our samples are: $0.9, 1.1, 0.96, 1.05$ (*i.e.*, close to 1), then after a few samples we'll get a good estimate of the true expected value.
  - If our samples are: zero 99% of the time, and one-hundred 1% of the time ⇒ expected value is correct, but estimate is worse! (You need many more samples).
- This latter issue is an example of a high variance MC estimator.

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

The Score Function Gradient Estimator → SGVB Estimator

One key contribution of the VAE paper is to propose an alternative estimator, which is better behaved (lower variance).

The Score Function Gradient Estimator → SGVB Estimator

One key contribution of the VAE paper is to propose an alternative estimator, which is better behaved (lower variance).

Sometimes referred to as the *stochastic gradient variational Bayes* (SGVB) estimator.

The Score Function Gradient Estimator → SGVB Estimator

One key contribution of the VAE paper is to propose an alternative estimator, which is better behaved (lower variance).

Sometimes referred to as the *stochastic gradient variational Bayes* (SGVB) estimator.

This is done in two steps:

(1) We reformulate the ELBO so that parts of it can be computed in closed form (*i.e.*, without Monte Carlo).

(2) Then we use an alternative gradient estimator (SGVB) based on the so-called **reparameterization trick**.

Reformulating the ELBO

The reformulation of the ELBO is as follows:

Reformulating the ELBO

The reformulation of the ELBO is as follows:

$$\log p(x) \geq \mathbb{E}_{q_\phi(z|x)} \left[\log p_\theta(x \mid z) \right] - \mathrm{KL} \left[q_\phi(z \mid x) || p_\theta(z) \right]$$

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Reformulating the ELBO

The reformulation of the ELBO is as follows:

$$\log p(x) \geq \mathbb{E}_{q_\phi(z|x)} \left[\log p_\theta(x \mid z)\right] - \mathrm{KL}\left[q_\phi(z \mid x) || p_\theta(z)\right]$$

Evidence

Reformulating the ELBO

The reformulation of the ELBO is as follows:

$$\log p(x) \geq \mathbb{E}_{q_\phi(z|x)} \left[\log p_\theta(x \mid z)\right] - \mathrm{KL}\left[q_\phi(z \mid x) || p_\theta(z)\right]$$

Evidence

ELBO (lower bound on the evidence), consisting of two terms:

Reformulating the ELBO

The reformulation of the ELBO is as follows:

$$\log p(x) \geq \mathbb{E}_{q_\phi(z|x)} \left[\log p_\theta(x \mid z)\right] - \mathrm{KL}\left[q_\phi(z \mid x) || p_\theta(z)\right]$$

Evidence

Expectation of the *log-likelihood*

Reformulating the ELBO

The reformulation of the ELBO is as follows:

$$\log p(x) \geq \mathbb{E}_{q_\phi(z|x)} \left[\log p_\theta(x \mid z)\right] - \mathrm{KL}\left[q_\phi(z \mid x) || p_\theta(z)\right]$$

Evidence

Expectation of the *log-likelihood*

KL between q *posterior* and p *prior*

Reformulating the ELBO

The reformulation of the ELBO is as follows:

$$\log p(x) \geq \mathbb{E}_{q_\phi(z|x)} \left[\log p_\theta(x \mid z)\right] - \mathrm{KL}\left[q_\phi(z \mid x) || p_\theta(z)\right]$$

Evidence

Expectation of the *log-likelihood*

KL between q *posterior* and p *prior*

(Can verify this using some algebra on the previous ELBO formulation).

Reformulating the ELBO

The reformulation of the ELBO is as follows:

$$\log p(x) \geq \mathbb{E}_{q_\phi(z|x)} \left[\log p_\theta(x \mid z)\right] - \mathrm{KL}\left[q_\phi(z \mid x) || p_\theta(z)\right]$$

Evidence

Expectation of the *log-likelihood*

KL between q *posterior* and p *prior*

This has an interesting interpretation!

Reformulating the ELBO

$$\log p(x) \geq \mathbb{E}_{q_\phi(z|x)} \left[\log p_\theta(x \mid z)\right] - \mathrm{KL}\left[q_\phi(z \mid x) || p_\theta(z)\right]$$

Evidence

Expectation of the *log-likelihood*

KL between q *posterior* and p *prior*

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Reformulating the ELBO

$$\log p(x) \geq \mathbb{E}_{q_\phi(z|x)} \left[\log p_\theta(x \mid z)\right] - \mathrm{KL}\left[q_\phi(z \mid x) || p_\theta(z)\right]$$

Evidence

Expectation of the *log-likelihood*

KL between q *posterior* and p *prior*

First:

Reformulating the ELBO

$$\log p(x) \geq \mathbb{E}_{q_\phi(z|x)} \left[\log p_\theta(x \mid z)\right] - \mathrm{KL}\left[q_\phi(z \mid x) || p_\theta(z)\right]$$

Evidence

Expectation of the *log-likelihood*

KL between q *posterior* and p *prior*

First:

- Think of x as an observed data point.

Reformulating the ELBO

$$\log p(x) \geq \mathbb{E}_{q_\phi(z|x)} \left[\log p_\theta(x \mid z)\right] - \mathrm{KL}\left[q_\phi(z \mid x) || p_\theta(z)\right]$$

Evidence | Expectation of the *log-likelihood* | KL between q *posterior* and p *prior*

First:

- Think of x as an observed data point.
- Think of $z \sim q_\phi(z \mid x)$ as a *"code"* that describes *x*.
  - (Note that both terms in the right hand side involve an expectation with respect to q.)

Reformulating the ELBO

$$\log p(x) \geq \mathbb{E}_{q_\phi(z|x)} \left[\log p_\theta(x \mid z)\right] - \mathrm{KL}\left[q_\phi(z \mid x) || p_\theta(z)\right]$$

Evidence | Expectation of the *log-likelihood* | KL between q *posterior* and p *prior*

First:

- Think of x as an observed data point.
- Think of $z \sim q_\phi(z \mid x)$ as a *"code"* that describes *x*.
  - (Note that both terms in the right hand side involve an expectation with respect to q.)
- ⇒ We will therefore call q_ϕ the **encoder**.

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Reformulating the ELBO

$$\log p(x) \geq \mathbb{E}_{q_\phi(z|x)} \left[\log p_\theta(x \mid z)\right] - \mathrm{KL}\left[q_\phi(z \mid x) || p_\theta(z)\right]$$

Evidence

Expectation of the *log-likelihood*

KL between q *posterior* and p *prior*

Next:

Reformulating the ELBO

$$\log p(x) \geq \mathbb{E}_{q_\phi(z|x)} \left[\log p_\theta(x \mid z)\right] - \mathrm{KL}\left[q_\phi(z \mid x) || p_\theta(z)\right]$$

Evidence

Expectation of the *log-likelihood*

KL between q *posterior* and p *prior*

Next:

encoder

- The first term is the log-likelihood of the observed x given the code $z \sim q_\phi(z \mid x)$.

Reformulating the ELBO

$$\log p(x) \geq \mathbb{E}_{q_\phi(z|x)} \left[\log p_\theta(x \mid z)\right] - \mathrm{KL}\left[q_\phi(z \mid x) || p_\theta(z)\right]$$

Evidence

Expectation of the *log-likelihood*

KL between q *posterior* and p *prior*

Next:

encoder

- The first term is the log-likelihood of the observed x given the code $z \sim q_\phi(z \mid x)$.
- ⇒ this first term is maximized when $p_\theta(x \mid z)$ assigns high probability to the original x.

Reformulating the ELBO

$$\log p(x) \geq \mathbb{E}_{q_\phi(z|x)} \left[\log p_\theta(x \mid z)\right] - \mathrm{KL}\left[q_\phi(z \mid x) || p_\theta(z)\right]$$

Evidence

Expectation of the *log-likelihood*

KL between q *posterior* and p *prior*

Next:

encoder

- The first term is the log-likelihood of the observed x given the code $z \sim q_\phi(z \mid x)$.
- ⇒ this first term is maximized when $p_\theta(x \mid z)$ assigns high probability to the original x.
- ⇒ *i.e.*, It is trying to *reconstruct x given the code z*.

Reformulating the ELBO

$$\log p(x) \geq \mathbb{E}_{q_\phi(z|x)} \left[\log p_\theta(x \mid z)\right] - \mathrm{KL}\left[q_\phi(z \mid x) || p_\theta(z)\right]$$

Evidence | Expectation of the *log-likelihood* | KL between q *posterior* and p *prior*

Next:

encoder

- The first term is the log-likelihood of the observed x given the code $z \sim q_\phi(z \mid x)$.
- ⇒ this first term is maximized when $p_\theta(x \mid z)$ assigns high probability to the original x.
- ⇒ *i.e.*, It is trying to *reconstruct x given the code z*.
- ⇒ We will therefore call $p_\theta(x \mid z)$ the **decoder**.
  - And the first term we call the *reconstruction error.*

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Reformulating the ELBO

$$\log p(x) \geq \mathbb{E}_{q_\phi(z|x)} \left[\log p_\theta(x \mid z)\right] - \mathrm{KL}\left[q_\phi(z \mid x) || p_\theta(z)\right]$$

Evidence

Expectation of the *log-likelihood*

KL between q *posterior* and p *prior*

Also:

Reformulating the ELBO

$$\log p(x) \geq \mathbb{E}_{q_\phi(z|x)} \left[\log p_\theta(x \mid z)\right] - \mathrm{KL}\left[q_\phi(z \mid x) || p_\theta(z)\right]$$

Evidence

Expectation of the *log-likelihood*

KL between q *posterior* and p *prior*

Also:

- The second term is the KL divergence between $q_\phi(z \mid x)$ and the model's prior $p_\theta(z)$.

Reformulating the ELBO

$$\log p(x) \geq \mathbb{E}_{q_\phi(z|x)} \left[\log p_\theta(x \mid z)\right] - \mathrm{KL}\left[q_\phi(z \mid x) || p_\theta(z)\right]$$

Evidence | Expectation of the *log-likelihood* | KL between q *posterior* and p *prior*

Also:

- The second term is the KL divergence between $q_\phi(z \mid x)$ and the model's prior $p_\theta(z)$.
- (In this model, we will assume the prior on z is a unit Normal distribution). $\mathcal{N}(0, I)$
  - ⇒ this encourages the codes z to look Gaussian distributed.

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Reformulating the ELBO

$$\log p(x) \geq \mathbb{E}_{q_\phi(z|x)} \left[\log p_\theta(x \mid z)\right] - \mathrm{KL}\left[q_\phi(z \mid x) || p_\theta(z)\right]$$

Evidence | Expectation of the *log-likelihood* | KL between q *posterior* and p *prior*

Also:

- The second term is the KL divergence between $q_\phi(z \mid x)$ and the model's prior $p_\theta(z)$.
- (In this model, we will assume the prior on z is a unit Normal distribution). $\mathcal{N}(0, I)$
  - ⇒ this encourages the codes z to look Gaussian distributed.
- We call this second term the *regularization term*.
  - It prevents $q_\phi(z \mid x)$ from, *e.g.*, simply encoding an identity mapping.

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Reformulating the ELBO

$$\log p(x) \geq \mathbb{E}_{q_\phi(z|x)} \left[\log p_\theta(x \mid z)\right] - \mathrm{KL}\left[q_\phi(z \mid x) || p_\theta(z)\right]$$

Evidence

Expectation of the *log-likelihood*

KL between q *posterior* and p *prior*

In summary:

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Reformulating the ELBO

$$\log p(x) \geq \mathbb{E}_{q_\phi(z|x)} \left[\log p_\theta(x \mid z)\right] - \mathrm{KL}\left[q_\phi(z \mid x) || p_\theta(z)\right]$$

Evidence

Expectation of the *log-likelihood*

KL between q *posterior* and p *prior*

In summary:

When we optimize this ELBO, we are trying to fit a $q_\phi(z \mid x)$ that will map an observation x onto a useful latent variable z …

… from which we are able to reconstruct x, via $p_\theta(x \mid z)$.

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Reformulating the ELBO

$$\log p(x) \geq \mathbb{E}_{q_\phi(z|x)} \left[\log p_\theta(x \mid z)\right] - \mathrm{KL}\left[q_\phi(z \mid x) || p_\theta(z)\right]$$

Evidence | Expectation of the *log-likelihood* | KL between q *posterior* and p *prior*

In summary:

When we optimize this ELBO, we are trying to fit a $q_\phi(z \mid x)$ that will map an observation x onto a useful latent variable z …

… from which we are able to reconstruct x, via $p_\theta(x \mid z)$.

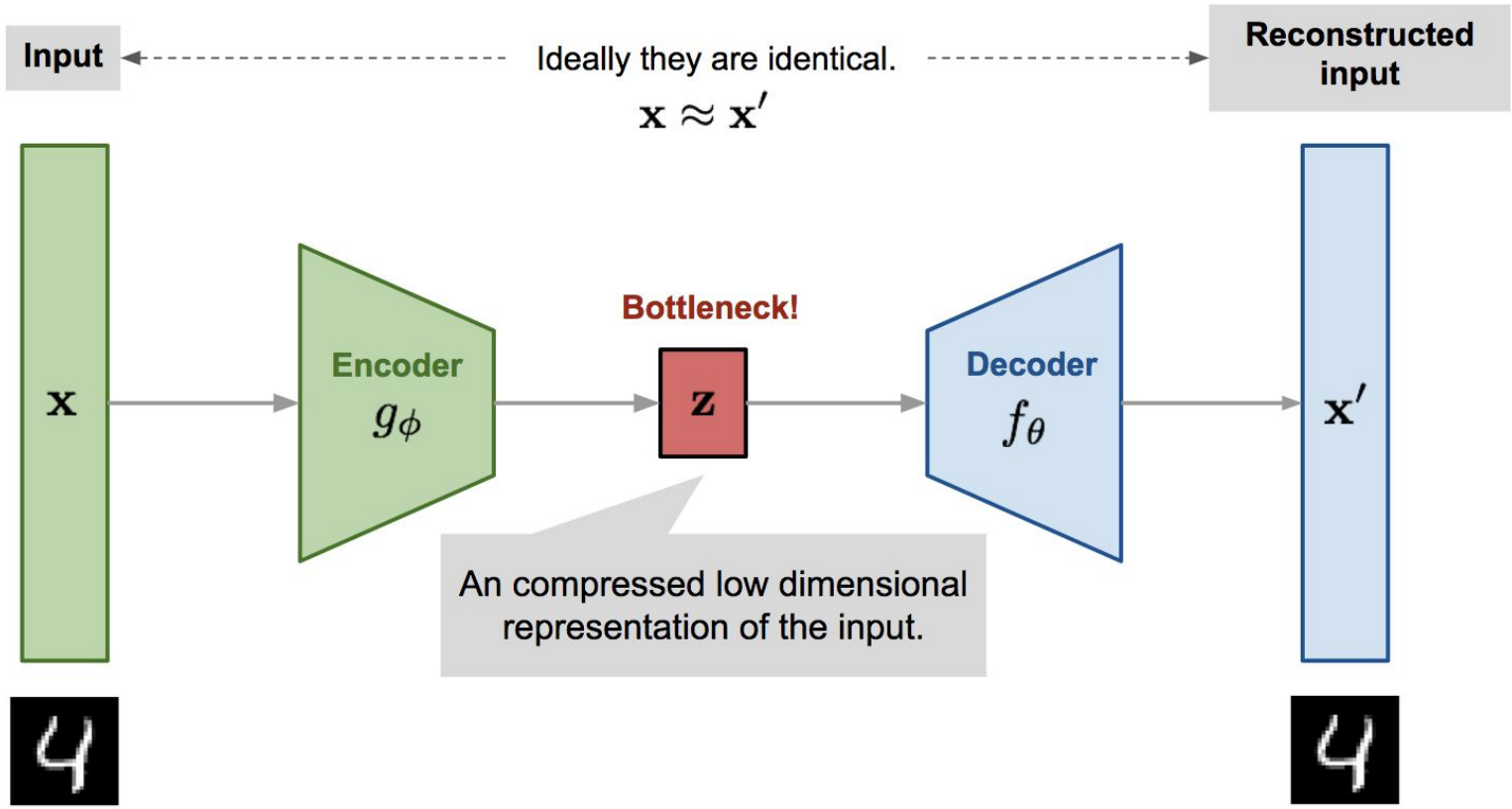
⇒ This is reminiscent of an *autoencoder neural network*.

(*i.e.*, a pair of neural networks, which encode/decode data, aiming to minimize the reconstruction error).

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Reformulating the ELBO

Aside — illustration of an **autoencoder** (first developed in 1990s, or even late 1980s).



Source: Lil'Log, "From Autoencoder to Beta-VAE"

Reparameterization Trick

So how can we use this reformulation to form a *lower-variance gradient estimator*, and then optimize the ELBO?

Reparameterization Trick

So how can we use this reformulation to form a *lower-variance gradient estimator*, and then optimize the ELBO?

We will use something call the **reparameterization trick**!

Reparameterization Trick

So how can we use this reformulation to form a *lower-variance gradient estimator*, and then optimize the ELBO?

We will use something call the **reparameterization trick**!

Recall:

SGVB Estimator

This is done in two steps:

(1) We reformulate the ELBO so that parts of it can be computed in closed form (*i.e.*, without Monte Carlo).

(2) Then we use an alternative gradient estimator (SGVB) based on the so-called **reparameterization trick**.

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Reparameterization Trick

Under certain mild conditions, we can express $q_\phi(z \mid x)$ via the following two-step generative process:

Reparameterization Trick

Under certain mild conditions, we can express $q_\phi(z \mid x)$ via the following two-step generative process:

- First sample a noise variable ϵ from a simple distribution $p(\epsilon)$ (such as a standard Normal $\mathcal{N}(0, I)$):

$$\epsilon \sim p(\epsilon)$$

Reparameterization Trick

Under certain mild conditions, we can express $q_\phi(z \mid x)$ via the following two-step generative process:

- First sample a noise variable ϵ from a simple distribution $p(\epsilon)$ (such as a standard Normal $\mathcal{N}(0, I)$):

$$\epsilon \sim p(\epsilon)$$

- Then we apply a deterministic transformation $g_\phi(\epsilon, x)$ that maps the random noise into a more-complex distribution:

$$z = g_\phi(\epsilon, x)$$

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Reparameterization Trick

Note: for many interesting classes of q_ϕ, it will be possible to define a g_ϕ such that $z = g_\phi(\epsilon, x)$ is distributed according to $q_\phi(z \mid x)$.

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Reparameterization Trick

Note: for many interesting classes of q_ϕ, it will be possible to define a g_ϕ such that $z = g_\phi(\epsilon, x)$ is distributed according to $q_\phi(z \mid x)$.

Graphical model of $q_\phi(z \mid x)$ can be viewed as similar to…

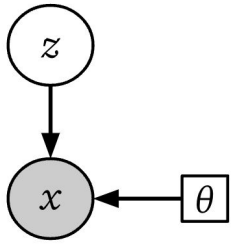
(recall from first PGM lecture)

Famous Bayesian Networks — Neural Transformations in PGMs

Can incorporate more-complex function transformations (or neural networks) into Bayesian networks!

But instead we could do:

$$z \sim p(z) = \mathcal{N}(0, 1)$$
$$x \sim p_\theta(x \mid z) = \mathcal{N}(g_\theta(z), 1)$$



$g_\theta(z)$ might be a complex fixed/known, or have parameters that we learn (*e.g.*, if we use a neural network!).

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Reparameterization Trick

We refer to this as a ***reparameterization trick*** (we are reparameterizing, *i.e.*, writing out $q_\phi(z \mid x)$ in a different way).

Reparameterization Trick

We refer to this as a ***reparameterization trick*** (we are reparameterizing, *i.e.*, writing out $q_\phi(z \mid x)$ in a different way).

Simplest example of the reparameterization trick:

Reparameterization Trick

We refer to this as a ***reparameterization trick*** (we are reparameterizing, *i.e.*, writing out $q_\phi(z \mid x)$ in a different way).

Simplest example of the reparameterization trick:

- Instead of writing: $z \sim q_{\mu,\sigma}(z) = \mathcal{N}(z; \mu, \sigma^2)$

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Reparameterization Trick

We refer to this as a ***reparameterization trick*** (we are reparameterizing, *i.e.*, writing out $q_\phi(z \mid x)$ in a different way).

Simplest example of the reparameterization trick:

- Instead of writing: $z \sim q_{\mu,\sigma}(z) = \mathcal{N}(z; \mu, \sigma^2)$
- We can write: $z = g_{\mu,\sigma}(\epsilon) = \mu + \epsilon \cdot \sigma, \quad \text{where } \epsilon \sim \mathcal{N}(\epsilon\ ;\ 0, 1)$

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Reparameterization Trick

Main advantages of the reparameterization trick:

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Reparameterization Trick

Main advantages of the reparameterization trick:

We can now write the gradient of an expectation (gradient wrt ϕ, expectation of q), for a given test function f, as:

Reparameterization Trick

Main advantages of the reparameterization trick:

We can now write the gradient of an expectation (gradient wrt ϕ, expectation of q), for a given test function f, as:

$$\nabla_\phi \mathbb{E}_{z\sim q_\phi(z|x)}\left[f(x,z)\right] = \nabla_\phi \mathbb{E}_{\epsilon\sim p(\epsilon)}\left[f(x,g_\phi(\epsilon,x))\right]$$

$$= \mathbb{E}_{\epsilon\sim p(\epsilon)}\left[\nabla_\phi f(x,g_\phi(\epsilon,x))\right]$$

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Reparameterization Trick

Main advantages of the reparameterization trick:

We can now write the gradient of an expectation (gradient wrt ϕ, expectation of q), for a given test function f, as:

$$\nabla_\phi \mathbb{E}_{z\sim q_\phi(z|x)}\left[f(x,z)\right] = \nabla_\phi \mathbb{E}_{\epsilon\sim p(\epsilon)}\left[f(x, g_\phi(\epsilon,x))\right]$$

$$= \mathbb{E}_{\epsilon\sim p(\epsilon)}\left[\nabla_\phi f(x, g_\phi(\epsilon,x))\right]$$

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Reparameterization Trick

Main advantages of the reparameterization trick:

We can now write the gradient of an expectation (gradient wrt ϕ, expectation of q), for a given test function f, as:

$$\nabla_\phi \mathbb{E}_{z\sim q_\phi(z|x)}\left[f(x,z)\right] = \nabla_\phi \mathbb{E}_{\epsilon\sim p(\epsilon)}\left[f(x, g_\phi(\epsilon,x))\right]$$

$$= \mathbb{E}_{\epsilon\sim p(\epsilon)}\left[\nabla_\phi f(x, g_\phi(\epsilon,x))\right]$$

Can now swap gradient and expectation!

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Reparameterization Trick

Main advantages of the reparameterization trick:

We can now write the gradient of an expectation (gradient wrt ϕ, expectation of q), for a given test function f, as:

$$\nabla_\phi \mathbb{E}_{z\sim q_\phi(z|x)}\left[f(x,z)\right] = \nabla_\phi \mathbb{E}_{\epsilon\sim p(\epsilon)}\left[f(x, g_\phi(\epsilon,x))\right]$$
$$= \mathbb{E}_{\epsilon\sim p(\epsilon)}\left[\nabla_\phi f(x, g_\phi(\epsilon,x))\right]$$

Can now swap gradient and expectation!

Can now take a Monte Carlo estimate of this expectation.

⇒ yields an estimate of the gradient for ϕ, with much lower variance than the score-function estimator.

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

What to choose for p and q?

What to choose for p and q?

Suppose, after learning θ, we hope to use $p_\theta(x \mid z)$ as a generative model.

(And we hope to use $q_\phi(z \mid x)$ to perform inference in order to learn θ).

What to choose for p and q?

Suppose, after learning θ, we hope to use $p_\theta(x \mid z)$ as a generative model.

(And we hope to use $q_\phi(z \mid x)$ to perform inference in order to learn θ).

Question: how should we specify the form of p_θ and q_ϕ ?

What to choose for p and q?

Suppose, after learning θ, we hope to use $p_\theta(x \mid z)$ as a generative model.

(And we hope to use $q_\phi(z \mid x)$ to perform inference in order to learn θ).

Question: how should we specify the form of p_θ and q_ϕ ?

Note:

What to choose for p and q?

Suppose, after learning θ, we hope to use $p_\theta(x \mid z)$ as a generative model.

(And we hope to use $q_\phi(z \mid x)$ to perform inference in order to learn θ).

Question: how should we specify the form of p_θ and q_ϕ ?

Note:

- We want $p_\theta(x \mid z)$ to be flexible enough to represent the richness of complex/high-dimensional data.

What to choose for p and q?

Suppose, after learning θ, we hope to use $p_\theta(x \mid z)$ as a generative model.

(And we hope to use $q_\phi(z \mid x)$ to perform inference in order to learn θ).

Question: how should we specify the form of p_θ and q_ϕ ?

Note:

- We want $p_\theta(x \mid z)$ to be flexible enough to represent the richness of complex/high-dimensional data.
- We want $q_\phi(z \mid x)$ to be flexible enough to approximate $p_\theta(z \mid x)$ (*i.e.*, the true posterior) well.

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

What to choose for p and q?

Answer: in VAEs, we will parameterize p_θ and q_ϕ using neural networks. 

What to choose for p and q?

Answer: in VAEs, we will parameterize p_θ and q_ϕ using neural networks. 

As an example, we can define $q_\phi(z \mid x)$ to be:

$$q_\phi(z \mid x) = \mathcal{N}\left(z\ ;\ \mu_\phi(x), \mathrm{diag}(\sigma_\phi(x))^2\right)$$

Where $\mu_\phi(x)$ and $\sigma_\phi(x)$ are vector-valued functions of x parametrized by an arbitrary neural network.

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

What to choose for p and q?

Answer: in VAEs, we will parameterize p_θ and q_ϕ using neural networks. 

As an example, we can define $q_\phi(z \mid x)$ to be:

$$q_\phi(z \mid x) = \mathcal{N}\left(z\ ;\ \mu_\phi(x), \mathrm{diag}(\sigma_\phi(x))^2\right)$$

Where $\mu_\phi(x)$ and $\sigma_\phi(x)$ are vector-valued functions of x parametrized by an arbitrary neural network.

⇒ could call them: "*mean network*" and "*standard deviation network*".

What to choose for p and q?

Answer: in VAEs, we will parameterize p_θ and q_ϕ using neural networks.

As an example, we can define $q_\phi(z \mid x)$ to be:

$$q_\phi(z \mid x) = \mathcal{N}\left(z\,;\ \mu_\phi(x), \mathrm{diag}(\sigma_\phi(x))^2\right)$$

More generally, can be applied to any exponential family distribution by parameterizing the sufficient statistics by a function of x.

Where $\mu_\phi(x)$ and $\sigma_\phi(x)$ are vector-valued functions of x parametrized by an arbitrary neural network.

⇒ could call them: "*mean network*" and "*standard deviation network*".

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Variational Autoencoder (VAE)

So what is the variational autoencoder (VAE), and how does it relate to the AEVB algorithm?

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Variational Autoencoder (VAE)

First, to summarize — the AEVB algorithm is simply the combination of:

Variational Autoencoder (VAE)

First, to summarize — the AEVB algorithm is simply the combination of:

(1) The alternative ELBO reformulation.

$$\mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x \mid z)\right] - \mathrm{KL}\left[q_\phi(z \mid x)||p_\theta(z)\right]$$

Expectation of the *log-likelihood*

KL between q *posterior* and p *prior*

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Variational Autoencoder (VAE)

First, to summarize — the AEVB algorithm is simply the combination of:

(1) The alternative ELBO reformulation.

$$\mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x \mid z)\right] - \mathrm{KL}\left[q_\phi(z \mid x)||p_\theta(z)\right]$$

Expectation of the *log-likelihood* — KL between q *posterior* and p *prior*

(2) The black-box (gradient-based) variational inference approach.

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Variational Autoencoder (VAE)

First, to summarize — the AEVB algorithm is simply the combination of:

(1) The alternative ELBO reformulation.

$$\mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x \mid z)\right] - \mathrm{KL}\left[q_\phi(z \mid x) || p_\theta(z)\right]$$

Expectation of the *log-likelihood* — KL between q *posterior* and p *prior*

(2) The black-box (gradient-based) variational inference approach.

(3) The reparameterization-based SGVB (low-variance) estimator.

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Variational Autoencoder (VAE)

First, to summarize — the AEVB algorithm is simply the combination of:

(1) The alternative ELBO reformulation.

$$\mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x \mid z)\right] - \mathrm{KL}\left[q_\phi(z \mid x) || p_\theta(z)\right]$$

Expectation of the *log-likelihood* — KL between *q posterior* and *p prior*

(2) The black-box (gradient-based) variational inference approach.

(3) The reparameterization-based SGVB (low-variance) estimator.

⇒ takes gradient steps on θ and ϕ to optimize the alternative ELBO.

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Variational Autoencoder (VAE)

The **VAE** is a special case — using the AEVB algorithm for a specific choice of p and q.

Variational Autoencoder (VAE)

The **VAE** is a special case — using the AEVB algorithm for a specific choice of p and q.

The model p_θ is parameterized as:

Variational Autoencoder (VAE)

The **VAE** is a special case — using the AEVB algorithm for a specific choice of p and q.

The model p_θ is parameterized as:

$$p_\theta(z) = \mathcal{N}(z;\ 0, I)$$
$$p_\theta(x \mid z) = \mathcal{N}\left(x;\ \mu_\theta(z), \mathrm{diag}(\sigma_\theta(z))^2\right)$$

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Variational Autoencoder (VAE)

The **VAE** is a special case — using the AEVB algorithm for a specific choice of p and q.

The model p_θ is parameterized as:

Prior on latent variable z is a Gaussian.

$$p_\theta(z) = \mathcal{N}(z;\ 0, I)$$

$$p_\theta(x \mid z) = \mathcal{N}\left(x;\ \mu_\theta(z), \mathrm{diag}(\sigma_\theta(z))^2\right)$$

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Variational Autoencoder (VAE)

The **VAE** is a special case — using the AEVB algorithm for a specific choice of p and q.

The model p_θ is parameterized as:

Prior on latent variable z is a Gaussian.

$$p_\theta(z) = \mathcal{N}(z;\ 0, I)$$

$$p_\theta(x \mid z) = \mathcal{N}\left(x;\ \mu_\theta(z), \mathrm{diag}(\sigma_\theta(z))^2\right)$$

Decode latent variable z into data point x.
(Also Gaussian).

Variational Autoencoder (VAE)

The **VAE** is a special case — using the AEVB algorithm for a specific choice of p and q.

The posterior approximation q_ϕ is parameterized as:

Variational Autoencoder (VAE)

The **VAE** is a special case — using the AEVB algorithm for a specific choice of p and q.

The posterior approximation q_ϕ is parameterized as:

$$q_\phi(z \mid x) = \mathcal{N}\left(z;\ \mu_\phi(x), \mathrm{diag}(\sigma_\phi(x))^2\right)$$

Encode data point x as latent variable z.
(Also Gaussian).

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Variational Autoencoder (VAE)

Useful properties of this choice of p_θ and q_ϕ :

Variational Autoencoder (VAE)

Useful properties of this choice of p_θ and q_ϕ :

- Closed-form expression to compute the regularization term (2nd term)

$$\mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x \mid z)\right] - \mathrm{KL}\left[q_\phi(z \mid x) || p_\theta(z)\right]$$

Expectation of the *log-likelihood* | KL between q *posterior* and p *prior*

  - KL divergence of two Gaussians ⇒ closed form expression and gradient!

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Variational Autoencoder (VAE)

Useful properties of this choice of p_θ and q_ϕ :

- Closed-form expression to compute the regularization term (2nd term)

$$\mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x \mid z)\right] - \mathrm{KL}\left[q_\phi(z \mid x) || p_\theta(z)\right]$$

Expectation of the *log-likelihood* | KL between q *posterior* and p *prior*

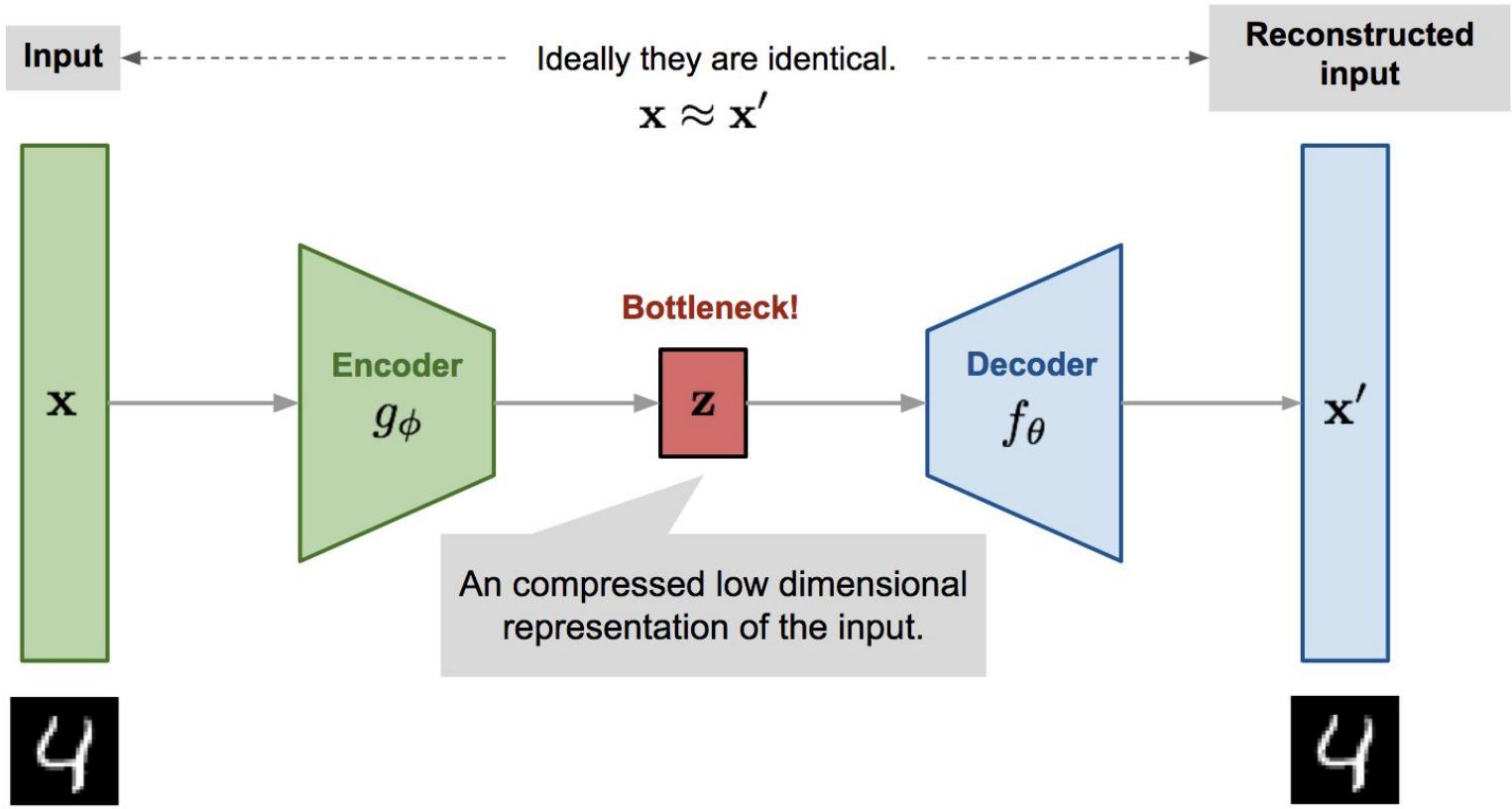
  - KL divergence of two Gaussians ⇒ closed form expression and gradient!

- So we only need to perform Monte Carlo gradient step on the 1st term.
  - And can carry out SGVB estimator using neural network reparameterization trick as described above.

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Variational Autoencoder (VAE)

Visual summary — back to the *autoencoder*:

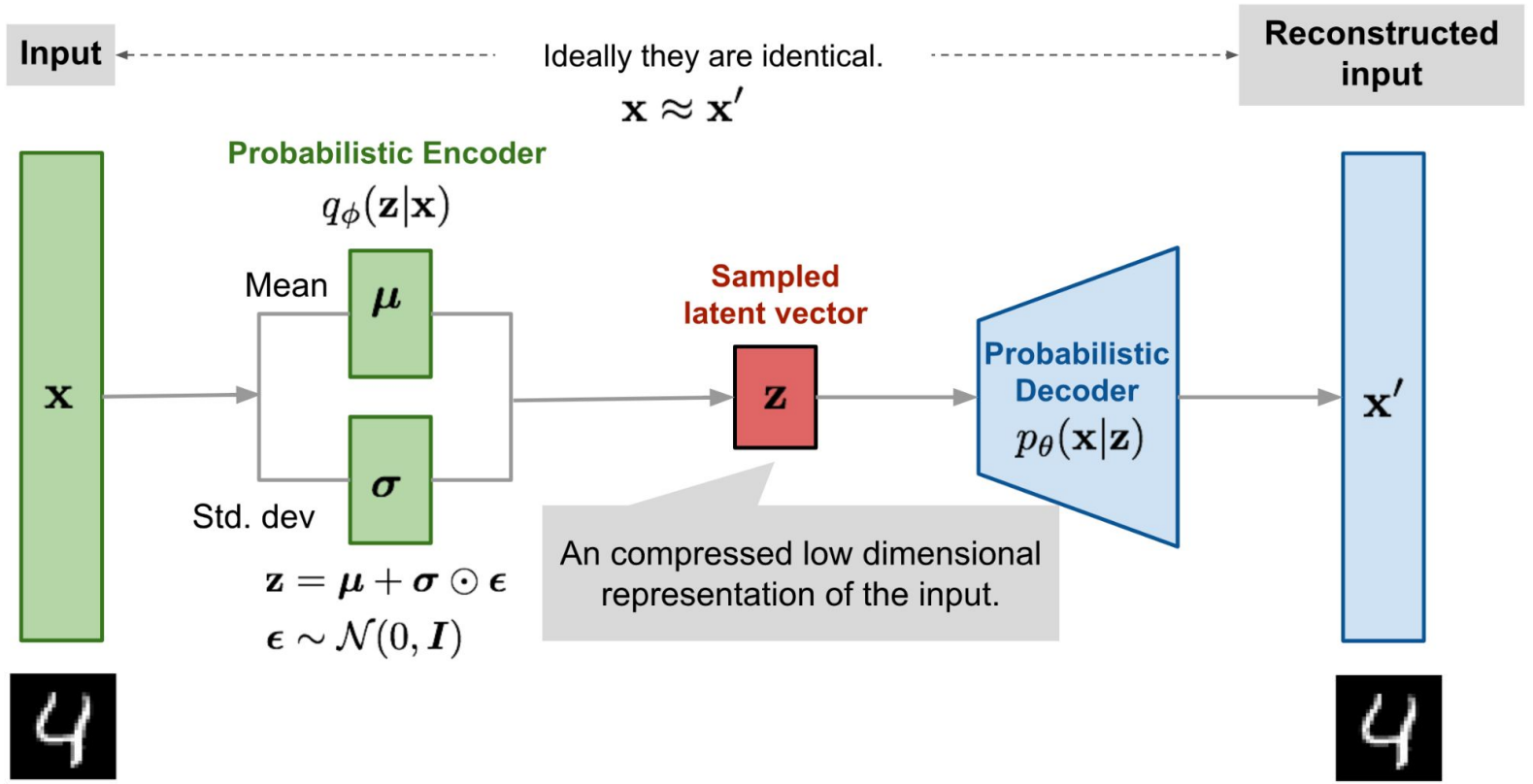


Source: Lil'Log, "From Autoencoder to Beta-VAE"

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Variational Autoencoder (VAE)

Visual summary — instead, a ***variational autoencoder***:



Source: Lil'Log, "From Autoencoder to Beta-VAE"

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Variational Autoencoder (VAE) — Results

So does it work?

Variational Autoencoder (VAE) — Results

So does it work? → Results from the original paper (*ICLR 2014*)

Variational Autoencoder (VAE) — Results

So does it work? → Results from the original paper (*ICLR 2014*)

Auto-Encoding Variational Bayes

Diederik P. Kingma
Machine Learning Group
Universiteit van Amsterdam
dpkingma@gmail.com

Max Welling
Machine Learning Group
Universiteit van Amsterdam
welling.max@gmail.com

Abstract

How can we perform efficient inference and learning in directed probabilistic models, in the presence of continuous latent variables with intractable posterior distributions, and large datasets? We introduce a stochastic variational inference and learning algorithm that scales to large datasets and, under some mild differentiability conditions, even works in the intractable case. Our contributions are two-fold. First, we show that a reparameterization of the variational lower bound yields a lower bound estimator that can be straightforwardly optimized using standard stochastic gradient methods. Second, we show that for i.i.d. datasets with continuous latent variables per datapoint, posterior inference can be made especially efficient by fitting an approximate inference model (also called a recognition model) to the intractable posterior using the proposed lower bound estimator. Theoretical advantages are reflected in experimental results.

1 Introduction

How can we perform efficient approximate inference and learning with directed probabilistic models whose continuous latent variables and/or parameters have intractable posterior distributions? The variational Bayesian (VB) approach involves the optimization of an approximation to the intractable posterior. Unfortunately, the common mean-field approach requires analytical solutions of expectations w.r.t. the approximate posterior, which are also intractable in the general case. We show how a reparameterization of the variational lower bound yields a simple differentiable unbiased estimator of the lower bound; this SGVB (Stochastic Gradient Variational Bayes) estimator can be used for efficient approximate posterior inference in almost any model with continuous latent variables and/or parameters, and is straightforward to optimize using standard stochastic gradient ascent techniques.

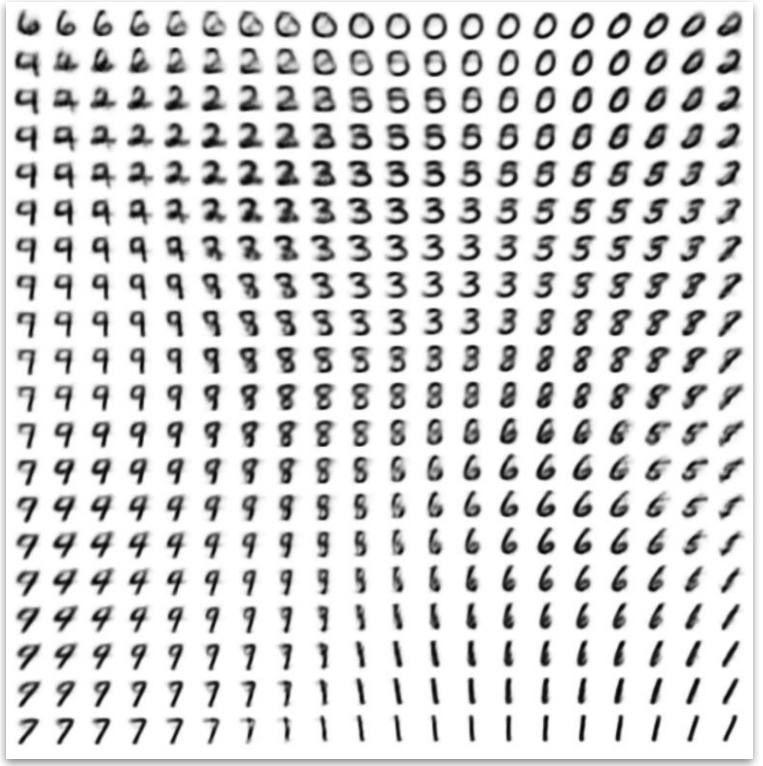
For the case of an i.i.d. dataset and continuous latent variables per datapoint, we propose the Auto-Encoding VB (AEVB) algorithm. In the AEVB algorithm we make inference and learning especially efficient by using the SGVB estimator to optimize a recognition model that allows us to perform very efficient approximate posterior inference using simple ancestral sampling, which in turn allows us to efficiently learn the model parameters, without the need of expensive iterative inference schemes (such as MCMC) per datapoint. The learned approximate posterior inference model can also be used for a host of tasks such as recognition, denoising, representation and visualization purposes. When a neural network is used for the recognition model, we arrive at the *variational auto-encoder*.

2 Method

The strategy in this section can be used to derive a lower bound estimator (a stochastic objective function) for a variety of directed graphical models with continuous latent variables. We will restrict ourselves here to the common case where we have an i.i.d. dataset with latent variables per datapoint, and where we like to perform maximum likelihood (ML) or maximum a posteriori (MAP) inference on the (global) parameters, and variational inference on the latent variables. It is, for example,

1

Frey Faces Dataset



MNIST Dataset

Source: Kingma, Diederik P., and Max Welling. "Auto-encoding variational bayes", 2014

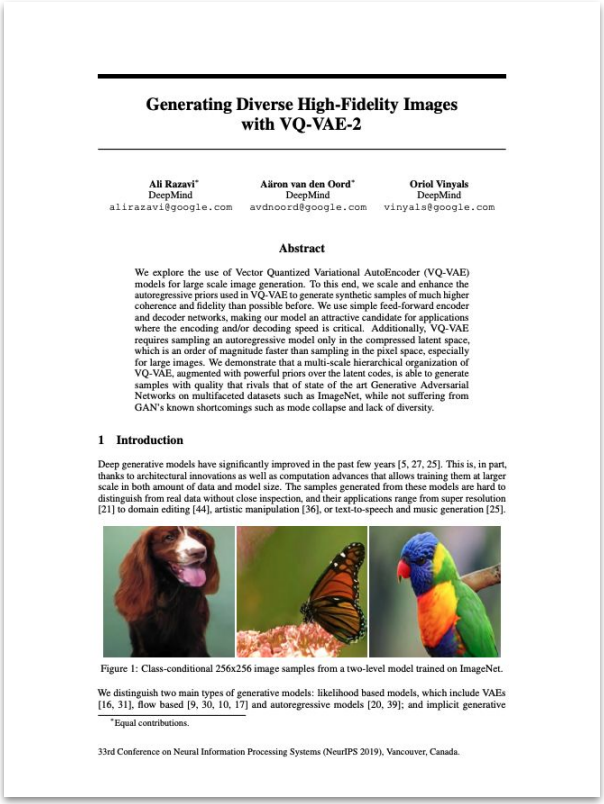
Variational Autoencoder (VAE) — Results

So does it work? → A few years later … (*NeurIPS 2019*)

Source: Razavi, Ali, Aaron Van den Oord, and Oriol Vinyals. "Generating diverse high-fidelity images with vq-vae-2", 2019.

Variational Autoencoder (VAE) — Results

So does it work? → A few years later … (*NeurIPS 2019*)



Generating Diverse High-Fidelity Images with VQ-VAE-2

Ali Razavi* DeepMind alirazavi@google.com
Aäron van den Oord* DeepMind avdnoord@google.com
Oriol Vinyals DeepMind vinyals@google.com

Abstract

We explore the use of Vector Quantized Variational AutoEncoder (VQ-VAE) models for large scale image generation. To this end, we scale and enhance the autoregressive priors used in VQ-VAE to generate synthetic samples of much higher coherence and fidelity than possible before. We use simple feed-forward encoder and decoder networks, making our model an attractive candidate for applications where the encoding and/or decoding speed is critical. Additionally, VQ-VAE requires sampling an autoregressive model only in the compressed latent space, which is an order of magnitude faster than sampling in the pixel space, especially for large images. We demonstrate that a multi-scale hierarchical organization of VQ-VAE, augmented with powerful priors over the latent codes, is able to generate samples with quality that rivals that of state of the art Generative Adversarial Networks on multifaceted datasets such as ImageNet, while not suffering from GAN's known shortcomings such as mode collapse and lack of diversity.

1 Introduction

Deep generative models have significantly improved in the past few years [5, 27, 25]. This is, in part, thanks to architectural innovations as well as computation advances that allows training them at larger scale in both amount of data and model size. The samples generated from these models are hard to distinguish from real data without close inspection, and their applications range from super resolution [21] to domain editing [44], artistic manipulation [36], or text-to-speech and music generation [25].

Figure 1: Class-conditional 256x256 image samples from a two-level model trained on ImageNet.

We distinguish two main types of generative models: likelihood based models, which include VAEs [16, 31], flow based [9, 30, 10, 17] and autoregressive models [20, 39]; and implicit generative

*Equal contributions.

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

FFQH Dataset

Source: Razavi, Ali, Aaron Van den Oord, and Oriol Vinyals. "Generating diverse high-fidelity images with vq-vae-2", 2019.

Next Class

Next Class

Next class, we will cover:

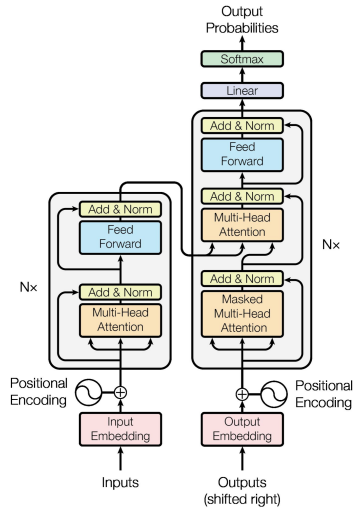
Next Class

Lecture: Autoregressive generative models and their use in large language modeling; other generative models (*e.g.*, GANs).

Next Class

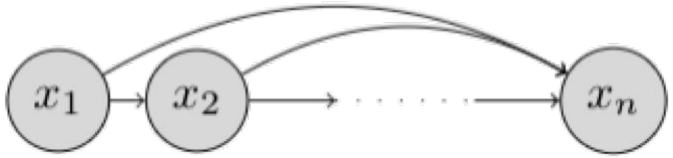
Lecture: Autoregressive generative models and their use in large language modeling; other generative models (*e.g.*, GANs).

- Neural autoregressive density estimation
- Recurrent neural networks
- Attention-based models
- Modern autoregressive transformer LLMs
- Also: Generative adversarial nets (GANs)



"Attention Is All You Need", Vaswani et al., 2017

Once upon a … [EOS]



Source: Stefano Ermon, Deep Graphical Models

After:

- Paper presentations from students.

Paper Presentations

Paper Presentations — Schedule

Starting at the **end of February** we will go through a few (~4) presentations per class (2nd half of class).

Paper Presentations — Schedule

Starting at the **end of February** we will go through a few (~4) presentations per class. Spreadsheet is here:

In the spreadsheet, which I will share, students will sign up for a time slot during the semester.

To make it fair, students who sign up for presentations on the first two dates will get slightly more-lenient grading (a point of extra credit on this assignment).

A	B	C	D	E	F	G	H
Date	Presentation ID	Presenter Name **(sign up!)**	Paper Title	Link/url to paper	Discussion Lead 1 **(sign up!)**	Discussion Lead 2 **(sign up!)**	Scribe **(sign up!)**
February 28							
	1						
	2						
	3						
	4						
March 7							
	5						

In-Class Participation and Discussion

There is a grade for in-class participation/discussion (during student presentations).
Students will sign up for one of three roles (again in the spreadsheet).
Each student has to do each role once during semester.

In-Class Participation and Discussion

There is a grade for in-class participation/discussion (during student presentations).

Students will sign up for one of three roles (again in the spreadsheet).

Each student has to do each role once during semester.

Roles 1 and 2 — Discussion Leads

- Read the paper before the class; Responsible for formulating a short list (~5 questions) for discussion; bring up a couple of these during class; submit all after class.

Role 3 — Scribe

- Read the paper before the class; take notes on paper during presentation; write up a ~1 page summary of the presentation.