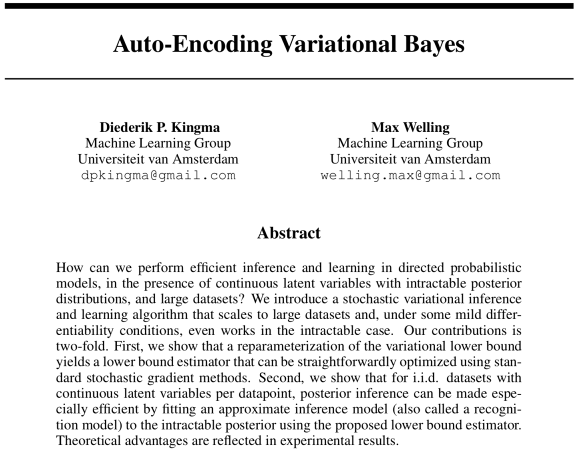
Auto-Encoding What?

Chris Mattioli

September 6, 2018

Auto-Encoding Variational Bayes!



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 is 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.

Notation

- $\boldsymbol{A} \equiv$ matrix
- $\boldsymbol{x} \equiv$ column vector
- $\alpha \equiv$ scalar
- W_{ij} or $w_{ij} \equiv$ element in row i and column j of matrix $\boldsymbol{W}$
- $x_i \equiv$ ith element (component) of vector $\boldsymbol{x}$

The Variational Auto-Encoder

This paper presents an elegant union of two different "machine learning" models:

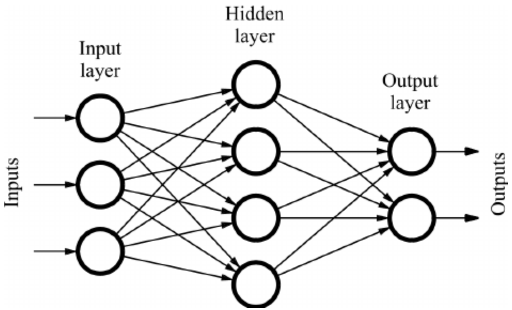
- Neural Networks (e.g., Auto-Encoders)
- (Continuous) Latent Variable Models (i.e., probabilistic graphical models)

this union is known as a *Variational Auto-Encoder*.

Variational Auto-Encoder

Graphical Representation

Neural Network Basics



This network is a *feedforward* network since it contains no cycles. Transitions from layer-to-layer are (typically) nonlinear functions evaluated at a linear transformation of the output of the previous layer.

Mathematical Representation

Neural Network Basics

Let $f_1, f_2, ..., f_{n-1}, f_n$ be a sequence of functions. (Typically these functions are nonlinear and differentiable though they don't have to be). Then a (feedforward) neural network is defined as the following:

$$NN(\boldsymbol{x}) \equiv f_n(\boldsymbol{W}_n f_{n-1}(\boldsymbol{W}_{n-1} f_{n-2}(\dots f_1(\boldsymbol{W}_1 \boldsymbol{x} + \boldsymbol{b}_1)\dots) + \boldsymbol{b}_{n-1}) + \boldsymbol{b}_n) \quad (1)$$

Training the Network

Neural Network Basics

- Neural Networks are supervised learners
- They judge the “closeness” of their attempted answer and the true answer by means of a *loss function* (a.k.a. an objective function)
  - These functions must be differentiable
  - E.g., Mean-Squared-Error: $\ell(\hat{y}, y) = \frac{1}{2}(\hat{y} - y)^2$
- Training a neural network implies minimizing this loss function with respect to the model weights (a.k.a parameters)
- Because this loss function is almost always a nonlinear function of the weights, numerical techniques are used to approximate the optimum.
  - E.g., stochastic gradient descent
- These techniques typically require the calculation of gradients (with repsect the model weights) hence the loss function needs to be differentiable

Backpropagation

Neural Network Basics

The technique used to calculate gradients in feed-forward neural networks is called *Backpropagation*. It starts at the end of the network (the loss function) and calculates derivatives going back at every step.

$$\ell(NN, y) = \frac{1}{2}(NN - y)^2, NN(\boldsymbol{x}) = f_3(\boldsymbol{W}_3 f_2(\boldsymbol{W}_2 f_1(\boldsymbol{W}_1 \boldsymbol{x} + \boldsymbol{b}_1) + \boldsymbol{b}_2) + \boldsymbol{b}_3)$$

$$\frac{\partial \ell}{\partial \boldsymbol{W}_3} = (NN - y)\frac{\partial f_3}{\partial f_3(\cdot)} f_2,$$

$$\frac{\partial \ell}{\partial \boldsymbol{W}_2} = (NN - y)\frac{\partial f_3}{\partial f_3(\cdot)} \boldsymbol{W}_3 \frac{\partial f_2}{\partial f_2(\cdot)} f_1,$$

$$\frac{\partial \ell}{\partial \boldsymbol{W}_1} = (NN - y)\frac{\partial f_3}{\partial f_3(\cdot)} \boldsymbol{W}_3 \frac{\partial f_2}{\partial f_2(\cdot)} \boldsymbol{W}_2 \frac{\partial f_1}{\partial f_1(\cdot)} \boldsymbol{x}$$

Stochastic Gradient Descent

Neural Network Basics

There are many different optimization techniques that can be used in neural network training. Stochastic gradient descent is common and has the following update rule:

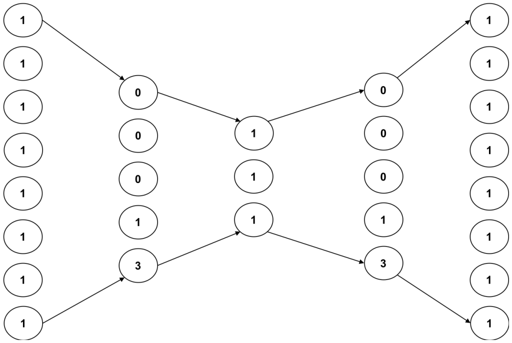
$$\boldsymbol{W}_i \leftarrow \boldsymbol{W}_i - \eta \frac{\partial \ell}{\partial \boldsymbol{W}_i}$$

where η is called the *learning rate*.

The Auto-Encoder

Neural Network Basics

The auto-encoder is a special type of neural network that learns a lower dimensional representation of the training data. This representation is known as an encoding. Instead of training against a label, it is trained on a reconstruction.



Variational Auto-Encoder

Probabilistic Inference

Variational Bayesian Inference

- Probabilistic inference is the act of asking a probability model a question.
- Given a probability distribution(s), probabilistic inference is the process of computing functions of the distribution(s).

Bayes Rule

The following equation is known as *Bayes Rule*:

$$P(\boldsymbol{z}|\boldsymbol{x}) = \frac{P(\boldsymbol{x}|\boldsymbol{z})P(\boldsymbol{z})}{P(\boldsymbol{x})} \tag{2}$$

$\boldsymbol{x} \equiv$ the observed/visible data
$\boldsymbol{z} \equiv$
latent variables, unobserved/hidden (these can also be model parameters)
$P(\boldsymbol{z}|\boldsymbol{x}) \equiv$ Posterior
$P(\boldsymbol{z}) \equiv$ Prior
$P(\boldsymbol{x}|\boldsymbol{z}) \equiv$ Likelihood
$P(\boldsymbol{x}) \equiv$ Evidence

Bayesian Inference

Variational Bayesian Inference

Bayesian Inference is centrally concerned with the *posterior distribution* which can be used in at least two important ways:

- Update prior beliefs as new data is observed
- Used to generate the Posterior Predictive Density which can be understood as the probability of seeing the next observation given the current observations.

When the posterior distribution is of the form of an established family of distributions (e.g., normal distribution), inference is easy. However, there commonly arise scenarios where this is not true (e.g., non-conjugate prior)

Latent Variable Model

Variational Bayesian Inference

A latent variable model makes the following assumption: data that I observe can be described by/is controlled by/is influenced by some hidden piece of information that I cannot directly know or observe. This hidden information is known as a latent variable

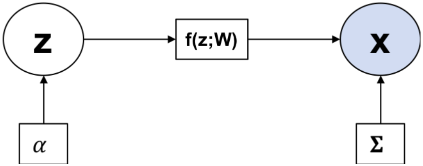
$$\boldsymbol{x} \equiv N\text{-dimensional observed data}, \boldsymbol{z} \equiv D\text{-dimensional latent variable}$$

Latent Variable Model, Example 1

$$\boldsymbol{x}|\boldsymbol{z} \sim \mathcal{N}(f(\boldsymbol{z}), \boldsymbol{\Sigma}), \;\; f(\boldsymbol{z}) = \boldsymbol{W}\boldsymbol{z},$$

$$\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{0}, \alpha^{-1}\boldsymbol{I}),$$

$$\boldsymbol{W}\ (N \times D), \boldsymbol{\Sigma}\ (N \times N), \alpha \text{ are not random}$$



Latent Variable Model, Example 1

Variational Bayesian Inference

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

In this example, the product of two normal distributions is normal:

$$p(\boldsymbol{z}|\boldsymbol{x}) \propto \mathcal{N}(f(\boldsymbol{z}), \boldsymbol{\Sigma})\mathcal{N}(\boldsymbol{0}, \alpha^{-1}\boldsymbol{I})$$

$$p(\boldsymbol{z}|\boldsymbol{x}) = \mathcal{N}\left(\left(\alpha + \boldsymbol{W}^\top\boldsymbol{\Sigma}^{-1}\boldsymbol{W}\right)^{-1}\boldsymbol{W}^\top\boldsymbol{\Sigma}^{-1}\boldsymbol{x}, \left(\alpha + \boldsymbol{W}^\top\boldsymbol{\Sigma}^{-1}\boldsymbol{W}\right)^{-1}\right)$$

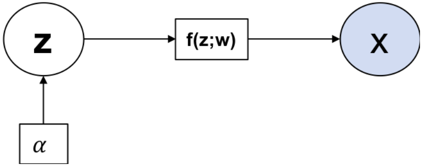
Note that this is the "LVM version" of Bayesian Linear Regression

Latent Variable Model, Example 2

Variational Bayesian Inference

$$x|\boldsymbol{z} \sim \text{Ber}(f(\boldsymbol{z})),\; f(\boldsymbol{z}) = \sigma(\boldsymbol{w}^\top \boldsymbol{z}),$$
$$\boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, \alpha^{-1}\boldsymbol{I}),$$

$\boldsymbol{w}$ $(D \times 1), \alpha$ are not random, $\sigma(\cdot)$ is the logistic sigmoid



Latent Variable Model, Example 2

Variational Bayesian Inference

$$p(\boldsymbol{z}|x) \propto (2\pi\alpha)^{-\frac{D}{2}} f(\boldsymbol{z})^{x}(1 - f(\boldsymbol{z}))^{1-x} e^{-\frac{\alpha}{2}\boldsymbol{z}^\top \boldsymbol{z}}$$

$$p(\boldsymbol{z}|x) = ?$$

The form of this posterior is unknown to us which makes our inference task intractable (can't perform integration, can't find normalization constant).

Is there a way to find a close approximation?

$$q(\boldsymbol{z}) \approx p(\boldsymbol{z}|\boldsymbol{x})$$

KL Divergence: Similarity Between Two Distributions

Variational Bayesian Inference

The Kullback-Leibler Divergence is a “measure” of similarity between two distributions:

$$KL[q(\boldsymbol{z})||p(\boldsymbol{z}|\boldsymbol{x})] \equiv \int_{\boldsymbol{z}} q(\boldsymbol{z}) \log \frac{q(\boldsymbol{z})}{p(\boldsymbol{z}|\boldsymbol{x})} d\boldsymbol{z}$$

It has the important properties:

$$KL[q||p] \geq 0, \quad \forall q, p$$

$$KL[p||p] = 0, \quad \forall p$$

$$KL(tq_1 + (1-t)q_2||tp_1 + (1-t)p_2) \leq tKL(q_1||p_1) + (1-t)KL(q_2||p_2)$$

for $0 \leq t \leq 1$ (KL Divergence is convex).

KL Divergence Rewritten

Variational Bayesian Inference

An equivalent definition of KL divergence is as follows:

$$\log p(\boldsymbol{x}) = \mathcal{L}(q) + KL[q(\boldsymbol{z})||p(\boldsymbol{z}|\boldsymbol{x})]$$

where

$$\mathcal{L}(q) = \mathbb{E}_{q(\boldsymbol{z})}[\log p(\boldsymbol{x}|\boldsymbol{z})] - KL[q(\boldsymbol{z})||p(\boldsymbol{z})]$$

is called the *variational lower bound*

Objective: maximize $\mathcal{L}(q)$

Variational Bayesian Inference

Our objective is the following:

$$q^*(\boldsymbol{z}) = \arg\max_q \mathcal{L}(q)$$

Note that:

$$\begin{aligned} \mathcal{L}(q) &= \log p(\boldsymbol{x}) \quad \text{when } q = p \\ \mathcal{L}(q) &< \log p(\boldsymbol{x}) \quad \text{when } q \neq p \end{aligned}$$

So by maximizing $\mathcal{L}(q)$ over the possible choices of q, we effectively minimize the divergence between q and p. It's prudent to constrain the choice of q to ensure computational tractability and flexibility.

Variational Auto-Encoder

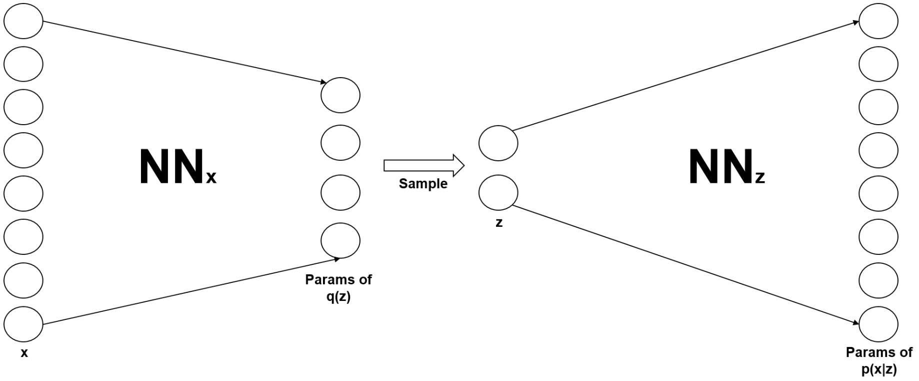
Set Up
Variational Auto-Encoder

- We claim that $\boldsymbol{x}$ is generated by some distribution parameterized by $NN_z(\boldsymbol{z})$, e.g., $\boldsymbol{x}|\boldsymbol{z} \sim \text{Ber}(NN_z(\boldsymbol{z}))$
- We establish a prior distribution for $\boldsymbol{z}$, e.g., $\boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, \alpha^{-1}\boldsymbol{I})$
- We *choose* the family of the approximating posterior distribution, e.g., $\boldsymbol{z}|\boldsymbol{x} \sim \mathcal{N}(\cdot, \cdot)$
- We claim that this distribution is parameterized by $NN_x(\boldsymbol{x})$, e.g., $\boldsymbol{z}|\boldsymbol{x} \sim \mathcal{N}(NN_x(\boldsymbol{x}), \beta^{-1}\boldsymbol{I})$

Goal: train NN_x and NN_z using the objective function $\mathcal{L}(\cdot)$. (Note that this learning is unsupervised!)

Topology

Variational Auto-Encoder



$\mathcal{L}(q)$ revisited
Variational Auto-Encoder

Recall:

$$\mathcal{L}(q) = \mathbb{E}_{q(\boldsymbol{z})}[\log p(\boldsymbol{x}|\boldsymbol{z})] - KL[q(\boldsymbol{z})||p(\boldsymbol{z})]$$

This expectation is problematic in the same way that inspired variational inference in the first place, that is, it is intractable to compute analytically in most cases.

The solution to this, presented in the paper, is the *stochastic gradient variational estimator* defined as:

$$\tilde{\mathcal{L}}(q) = \frac{1}{M}\sum_{m} \log p(\boldsymbol{x}|\boldsymbol{z}_m) - KL[q(\boldsymbol{z})||p(\boldsymbol{z})]$$

The Sampling Step

Variational Auto-Encoder

The “middle” portion of a Variational Auto-Encoder features a sampling step; however generally speaking, the act of sampling a random variable is *not* differentiable.

The solution to this, presented in the paper, is the *reparameterization trick*, which is a deterministic expression of $\boldsymbol{z}$ by means of an independent random variable ϵ.

$$\boldsymbol{z} = g(\epsilon; NN_x(\boldsymbol{x}))$$

The Form of $g(\cdot)$

Variational Auto-Encoder

The function $g(\epsilon; NN_x(\boldsymbol{x}))$ depends on $q(\boldsymbol{z})$. The following is a few examples of reparameterizations for different continuous distributions:

- If $\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2)$ then $g(\boldsymbol{\epsilon}) = \boldsymbol{\mu} + \boldsymbol{\sigma}^2 \odot \boldsymbol{\epsilon}$ where $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$
- If $z \sim \text{Exp}(\lambda)$ then $g(\epsilon) = -\frac{1}{\lambda}\log(-\epsilon + 1)$ where $\epsilon \sim \text{Uniform}(0, 1)$
- If $z \sim \text{Gamma}(r, \theta)$ and $r \geq 1,\ r \in \mathbb{Z}$ then $g(\boldsymbol{\epsilon}) = -\theta \sum_{i=1}^{r} \log(-\epsilon_i + 1)$ where $\epsilon_i \sim \text{Uniform}(0, 1)$

Comparison to Standard Auto-Encoders

- A SAE's encodings are uninterpretable
    - Though a VAE suffers from the same, you not only know the distribution of the encoding but you also get to select that distribution!
- SAE's are *not* generative; their encodings are direct
    - Because VAE's encode parameters of a distribution, that distribution can be leveraged to generate new samples