Variational Autoencoders

November 13, 2020

Table of contents

Overview

Deep Generative Models

- Can generate new samples (click here)
- Can interpolate across given samples
- If likelihood based, can be used to assess anomaly.

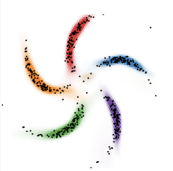
Composing PGM's with NN's

We can compose probabilistic graphical models with neural networks to exploit their complementary strengths.

Data

GMM

GMM-SVAE



The resulting model is expressive, but also interpretable/decomposable.

Johnson, M., Duvenaud, D. K., Wiltschko, A., Adams, R. P., & Datta, S. R. (2016). Composing graphical models with neural networks for structured representations and fast inference. In Advances in neural information processing systems (pp. 2946-2954).

Parameterizing Conditional Distributions with Neural Networks

- Have neural networks output *parameters* of probability distributions, rather than predictions directly.

Parameterizing Conditional Distributions with Neural Networks

- Have neural networks output *parameters* of probability distributions, rather than predictions directly.
- The cost function for optimization is now probabilistic (e.g., maximum likelihood, minimum KL-divergence) rather than minimizing a distance to the target.

Deep Latent Variable Models (DLVMs)

Let us introduce DLVMs by comparing them to the latent variable models (MM, HMM, LDA) we've considered so far.

Deep Latent Variable Models (DLVMs)

Let us introduce DLVMs by comparing them to the latent variable models (MM, HMM, LDA) we've considered so far.

How does the latent state z_i impact the data distribution for a particular sample, $x^{(i)} \sim F_{\theta_{z_i}}$?

Previously:	$\theta^{(i)} = \theta_{z_i}$	select one of K fixed parameters
Here:	$\theta^{(i)} = \texttt{NeuralNetwork}_{\eta}(z_i)$	flexibly create a sample-specific parameter

Deep Latent Variable Models (DLVMs)

Let us introduce DLVMs by comparing them to the latent variable models (MM, HMM, LDA) we've considered so far.

How does the latent state z_i impact the data distribution for a particular sample, $x^{(i)} \sim F_{\theta_{z_i}}$?

Previously:	$\theta^{(i)} = \theta_{z_i}$	select one of K fixed parameters
Here:	$\theta^{(i)} = \texttt{NeuralNetwork}_\eta(z_i)$	flexibly create a sample-specific parameter

Notes:

1. The latent variables now have continuous support (although this is an artifact of our choices for earlier examples)

$$\text{Previously:} \quad z_i \in \{1, ..., K\}$$
$$\text{Here:} \quad z_i \in \mathbb{R}^k$$

Deep Latent Variable Models (DLVMs)

Let us introduce DLVMs by comparing them to the latent variable models (MM, HMM, LDA) we've considered so far.

How does the latent state z_i impact the data distribution for a particular sample, $x^{(i)} \sim F_{\theta_{z_i}}$?

Previously:	$\theta^{(i)} = \theta_{z_i}$	select one of K fixed parameters
Here:	$\theta^{(i)} = \texttt{NeuralNetwork}_\eta(z_i)$	flexibly create a sample-specific parameter

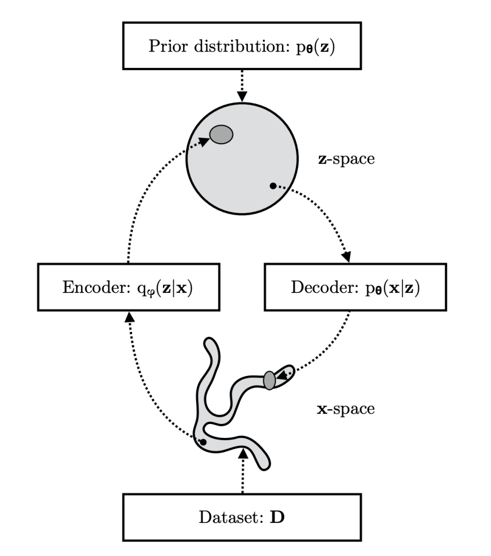
Notes:

1. The latent variables now have continuous support (although this is an artifact of our choices for earlier examples)

$$\text{Previously:} \quad z_i \in \{1, ..., K\}$$
$$\text{Here:} \quad z_i \in \mathbb{R}^k$$

2. The parameters we need to learn, η, is fixed in dimension.

Variational Autoencoders



Probabilistic model

Simplification

For ease of illustration, we restrict our attention to a variational autoencoder that applies i.i.d assumptions and Gaussian distributions (and therefore real-valued observations) throughout. Note that neither assumption is necessary.

Probabilistic decoder

Consider a parametric frequentist latent variable model, with

- observations $x = (x^{(i)})_{i=1}^N, \quad x^{(i)} \in \mathbb{R}^d$
- latent variables $z = (z^{(i)})_{i=1}^N, \quad z^{(i)} \in \mathbb{R}^k$
- parameter θ (fixed but to be learned)

Let us model our observations x via the factorization

$$p_\theta(x|z) = \prod_i p_\theta(x^{(i)}|z^{(i)})$$

Probabilistic decoder

Consider a parametric frequentist latent variable model, with

- observations $x = (x^{(i)})_{i=1}^N, \quad x^{(i)} \in \mathbb{R}^d$
- latent variables $z = (z^{(i)})_{i=1}^N, \quad z^{(i)} \in \mathbb{R}^k$
- parameter θ (fixed but to be learned)

Let us model our observations x via the factorization

$$p_\theta(x|z) = \prod_i p_\theta(x^{(i)}|z^{(i)})$$

Let the likelihood of each observation $x^{(i)}$ be obtained by using a Multi-Layer Perceptron (MLP), parameterized by weights θ, to map latent variable $z^{(i)}$ to parameters governing a Gaussian distribution of observation $x^{(i)}$.

Probabilistic decoder

Consider a parametric frequentist latent variable model, with

- observations $x = (x^{(i)})_{i=1}^{N}, \quad x^{(i)} \in \mathbb{R}^d$
- latent variables $z = (z^{(i)})_{i=1}^{N}, \quad z^{(i)} \in \mathbb{R}^k$
- parameter θ (fixed but to be learned)

Let us model our observations x via the factorization

$$p_\theta(x|z) = \prod_i p_\theta(x^{(i)}|z^{(i)})$$

Let the likelihood of each observation $x^{(i)}$ be obtained by using a Multi-Layer Perceptron (MLP), parameterized by weights θ, to map latent variable $z^{(i)}$ to parameters governing a Gaussian distribution of observation $x^{(i)}$.

$$x^{(i)} \mid z^{(i)}, \theta \sim \mathcal{N}\Big(\mu_{x^{(i)}}(z^{(i)}, \theta),\ \Sigma_{x^{(i)}}(z^{(i)}, \theta)\Big) \tag{2.1}$$

Probabilistic decoder

Consider a parametric frequentist latent variable model, with

- observations $x = (x^{(i)})_{i=1}^{N}, \quad x^{(i)} \in \mathbb{R}^d$
- latent variables $z = (z^{(i)})_{i=1}^{N}, \quad z^{(i)} \in \mathbb{R}^k$
- parameter θ (fixed but to be learned)

Let us model our observations x via the factorization

$$p_\theta(x|z) = \prod_i p_\theta(x^{(i)}|z^{(i)})$$

Let the likelihood of each observation $x^{(i)}$ be obtained by using a Multi-Layer Perceptron (MLP), parameterized by weights θ, to map latent variable $z^{(i)}$ to parameters governing a Gaussian distribution of observation $x^{(i)}$.

$$x^{(i)} \mid z^{(i)}, \theta \sim \mathcal{N}\Big(\mu_{x^{(i)}}(z^{(i)}, \theta),\ \Sigma_{x^{(i)}}(z^{(i)}, \theta)\Big) \tag{2.1}$$

Since the MLP maps latent variables, z, to the parameters of a probability distribution over observed data, x, we refer to it as a **probabilistic decoder.**

Notes on notation

1. $\mathcal{N}(M, V)$ refers to the Gaussian density with mean M and covariance V.
2. $\mu_{x^{(i)}}(z^{(i)}, \theta)$ is meant to denote the mean parameter for a distribution over observed datum $x^{(i)}$; that parameter is a function of latent variable z and learnable parameter θ. Notation should be similarly interpreted throughout this section.

Probabilistic encoder

Probabilistic encoder

Let us additionally put a prior distribution on the latent variables:

$$p_\theta(z) = \prod_i p_\theta(z^{(i)}) = \prod_i \mathcal{N}(\mathbf{0}, \mathbb{I})$$

Probabilistic encoder

Let us additionally put a prior distribution on the latent variables:

$$p_\theta(z) = \prod_i p_\theta(z^{(i)}) = \prod_i \mathcal{N}(\mathbf{0}, \mathbb{I})$$

In this case, the posterior distribution, $p_\theta(z|x)$, is intractable.

Probabilistic encoder

Let us additionally put a prior distribution on the latent variables:

$$p_\theta(z) = \prod_i p_\theta(z^{(i)}) = \prod_i \mathcal{N}(\mathbf{0}, \mathbb{I})$$

In this case, the posterior distribution, $p_\theta(z|x)$, is intractable.

However, we consider an approximation by using a Multi-Layer Perceptron (MLP), parameterized by weights ϕ, to map observation x to parameters governing a Gaussian distribution of latent variable z:

$$q_\phi(z|x) = \prod_i q_\phi(z^{(i)}|x^{(i)})$$

$$z^{(i)}|x^{(i)}, \phi \sim \mathcal{N}\left(\mu_{z^{(i)}}(x^{(i)}, \phi),\ \Sigma_{z^{(i)}}(x^{(i)}, \phi)\right) \tag{2.2}$$

Probabilistic encoder

Let us additionally put a prior distribution on the latent variables:

$$p_\theta(z) = \prod_i p_\theta(z^{(i)}) = \prod_i \mathcal{N}(\mathbf{0}, \mathbb{I})$$

In this case, the posterior distribution, $p_\theta(z|x)$, is intractable.

However, we consider an approximation by using a Multi-Layer Perceptron (MLP), parameterized by weights ϕ, to map observation x to parameters governing a Gaussian distribution of latent variable z:

$$q_\phi(z|x) = \prod_i q_\phi(z^{(i)}|x^{(i)})$$

$$z^{(i)}|x^{(i)}, \phi \sim \mathcal{N}\left(\mu_{z^{(i)}}(x^{(i)}, \phi),\ \Sigma_{z^{(i)}}(x^{(i)}, \phi)\right) \tag{2.2}$$

Since the MLP maps observations, x, to to the parameters of a probability distribution over latent variables, z, we refer to it as a **probabilistic encoder.**

Probabilistic encoder

- We may regard the probabilistic encoder as an approximation to the posterior distribution over latent variables which results from using the probabilistic decoder as a likelihood.
- The probabilistic encoder is sometimes also referred to as a **recognition model**.

Sample Implementation

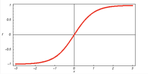
Sample Implementation

Following Appendix C.2 of the VAE paper, we provide a sample implementation for the probabilistic encoder and decoder.

Probabilistic decoding

We may, for example, specifically assume that a latent variable $z^{(i)}$ can be probabilistically decoded into observation $x^{(i)}$ via the following process

$$
\begin{aligned}
h^{(i)} &= \mathtt{tanh}(W_1 \, z^{(i)} + b_1) \\
\mu_{x^{(i)}} &= W_{21} h^{(i)} + b_{41}, \quad \log \sigma^2_{x^{(i)}} = W_{22} h^{(i)} + b_{22} \\
x | z &\sim \mathcal{N}(\mu_{x^{(i)}}, \Sigma_{x^{(i)}}), \quad \text{where } \mathrm{diag}(\Sigma_{x^{(i)}}) = \sigma^2_{x^{(i)}}
\end{aligned}
$$



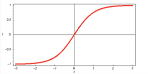
The hyperbolic tangent (`tanh`) function

where (W_1, W_{21}, W_{22}) are the weights and (b_1, b_{21}, b_{22}) are the biases of a Multi-Layer Perceptron (MLP).

Probabilistic decoding

We may, for example, specifically assume that a latent variable $z^{(i)}$ can be probabilistically decoded into observation $x^{(i)}$ via the following process

$$
\begin{aligned}
h^{(i)} &= \mathtt{tanh}(W_1\, z^{(i)} + b_1) \\
\mu_{x^{(i)}} &= W_{21} h^{(i)} + b_{41}, \quad \log \sigma^2_{x^{(i)}} = W_{22} h^{(i)} + b_{22} \\
x|z &\sim \mathcal{N}(\mu_{x^{(i)}}, \Sigma_{x^{(i)}}), \quad \text{where } \mathrm{diag}(\Sigma_{x^{(i)}}) = \sigma^2_{x^{(i)}}
\end{aligned}
$$



The hyperbolic tangent ($\mathtt{tanh}$) function

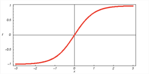
where (W_1, W_{21}, W_{22}) are the weights and (b_1, b_{21}, b_{22}) are the biases of a Multi-Layer Perceptron (MLP).

Letting $\theta := (W_1, W_{21}, W_{22}, b_1, b_{21}, b_{22})$, we may use the trained decoder to define the likelihood, $p_\theta(x|z)$, as defined in (2.1).

Probabilistic encoding

We may, for example, specifically assume that an observation $x^{(i)}$ can be probabilistically encoded into latent variable $z^{(i)}$ via the following process

$$h^{(i)} = \texttt{tanh}(W_3 x^{(i)} + b_1)$$
$$\mu_{z^{(i)}} = W_{41} h^{(i)} + b_{41}, \quad \log \sigma^2_{z^{(i)}} = W_{42} h^{(i)} + b_{42}$$
$$z^{(i)} \sim \mathcal{N}(\mu_{z^{(i)}}, \Sigma_{z^{(i)}}), \quad \text{where } \operatorname{diag}(\Sigma_{z^{(i)}}) = \sigma^2_{z^{(i)}}$$



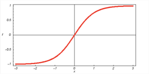
The hyperbolic tangent (tanh) function

where (W_3, W_{41}, W_{42}) are the weights and (b_3, b_{41}, b_{42}) are the biases of a Multi-Layer Perceptron (MLP).

Probabilistic encoding

We may, for example, specifically assume that an observation $x^{(i)}$ can be probabilistically encoded into latent variable $z^{(i)}$ via the following process

$$h^{(i)} = \texttt{tanh}(W_3 x^{(i)} + b_1)$$
$$\mu_{z^{(i)}} = W_{41} h^{(i)} + b_{41}, \quad \log \sigma^2_{z^{(i)}} = W_{42} h^{(i)} + b_{42}$$
$$z^{(i)} \sim \mathcal{N}(\mu_{z^{(i)}}, \Sigma_{z^{(i)}}), \quad \text{where } \mathrm{diag}(\Sigma_{z^{(i)}}) = \sigma^2_{z^{(i)}}$$



The hyperbolic tangent (`tanh`) function

where (W_3, W_{41}, W_{42}) are the weights and (b_3, b_{41}, b_{42}) are the biases of a Multi-Layer Perceptron (MLP).

Letting $\phi := (W_3, W_{41}, W_{42}, b_3, b_{41}, b_{42})$, we may use the trained encoder to define the approximate posterior, $q_\phi(z|x)$, as defined in (2.2).

Inference

We use variational inference to jointly optimize (θ, ϕ). For example, in our sample implementation, we have

$$\theta = (W_1, W_{21}, W_{22}, b_1, b_{21}, b_{22}) \qquad \text{generative parameters}$$
$$\phi = (W_3, W_{41}, W_{42}, b_3, b_{41}, b_{42}) \qquad \text{variational parameters}$$

In particular, we construct $\mathcal{F}(\theta, \phi; x)$, a lower-bound on the marginal likelihood, $p_\theta(x)$, via the entropy/energy decomposition which is standard in variational inference:

$$\mathcal{F}(\theta, \phi; x) = \mathbb{E}_{q_\phi(z|x)}[-\log q_\phi(z|x)) + \log p_\theta(x, z)] \tag{4.1}$$

We train the model by performing stochastic gradient descent on the variational lower bound $\mathcal{F}$. (How is this different than what we've seen?)

We train the model by performing stochastic gradient descent on the variational lower bound $\mathcal{F}$. (How is this different than what we've seen?)

During training, the objective function (4.1) is approximated by performing a Monte Carlo approximation of the expectation.

Given minibatch $x^{(i)}$, we would like to take L samples from $q_\phi(z|x^{(i)})$

$$z^{(i.l)} \sim q_\phi(z^{(i,l)}|x^{(i)})$$

and obtain the following estimator:

$$\mathcal{F}(\theta, \phi; x^{(i)}) \approx \frac{1}{L}\sum_{l=1}^{L} -\log q_\phi(z^{(i,l)}|x^{(i)}) + \log p_\theta(x^{(i)}, z^{(i,l)}) \qquad (4.2)$$

Reparametrization trick

However, naively backpropagating gradients in this case would have a problem:

Reparametrization trick

However, naively backpropagating gradients in this case would have a problem: It would ignore the role of the parameter in the sampling step.

Reparametrization trick

However, naively backpropagating gradients in this case would have a problem: It would ignore the role of the parameter in the sampling step.

Thus, we use the **reparameterization trick** – basically, *we remove the parameters from the sampling mechanism*.

Reparametrization trick

However, naively backpropagating gradients in this case would have a problem: It would ignore the role of the parameter in the sampling step.

Thus, we use the **reparameterization trick** – basically, *we remove the parameters from the sampling mechanism*.

Example	$q_\varphi(z)$	$p(\varepsilon)$	$g(\varphi, \varepsilon)$	Also...
Normal dist.	$z \sim N(\mu,\sigma)$	$\varepsilon \sim N(0,1)$	$z = \mu + \sigma * \varepsilon$	Location-scale familie: Laplace, Elliptical, Student's t, Logistic, Uniform, Triangular, ...
Exponential	$z \sim \exp(\lambda)$	$\varepsilon \sim U(0,1)$	$z = -\log(1 - \varepsilon)/\lambda$	Invertible CDF: Cauchy, Logistic, Rayleigh, Pareto,Weibull, Reciprocal, Gompertz, Gumbel and Erlan, ...
Other	$z \sim \log N(\mu,\sigma)$	$\varepsilon \sim N(0,1)$	$z = \exp(\mu + \sigma * \varepsilon)$	Gamma, Dirichlet, Beta, Chi-Squared, and F distributions

Image Credit: DP. Kingma

Reparametrization trick

However, naively backpropagating gradients in this case would have a problem: It would ignore the role of the parameter in the sampling step.

Thus, we use the **reparameterization trick** – basically, *we remove the parameters from the sampling mechanism*.

Example	$q_\varphi(z)$	$p(\varepsilon)$	$g(\varphi, \varepsilon)$	Also...
Normal dist.	$z \sim N(\mu,\sigma)$	$\varepsilon \sim N(0,1)$	$z = \mu + \sigma * \varepsilon$	Location-scale familie: Laplace, Elliptical, Student's t, Logistic, Uniform, Triangular, ...
Exponential	$z \sim \exp(\lambda)$	$\varepsilon \sim U(0,1)$	$z = -\log(1 - \varepsilon)/\lambda$	Invertible CDF: Cauchy, Logistic, Rayleigh, Pareto,Weibull, Reciprocal, Gompertz, Gumbel and Erlan, ...
Other	$z \sim \log N(\mu,\sigma)$	$\varepsilon \sim N(0,1)$	$z = \exp(\mu + \sigma * \varepsilon)$	Gamma, Dirichlet, Beta, Chi-Squared, and F distributions

Image Credit: DP. Kingma

Rk: In our case, the variational distribution is a multivariate normal, so $p(\epsilon)$ is simply a Gaussian with zero mean and identity covariance.

Reparametrization trick

Using the *reparameterization trick*, we construct a differentiable transformation g_ϕ of parameterless distribution $p(\epsilon)$ such that $g_\phi(\epsilon, x^{(i)})$ has the same distribution as $q_\phi(z^{(i)}|x^{(i)})$.

Using this trick, we take L samples $\{\epsilon_1, ..., \epsilon_L\}$ from $p(\epsilon)$ and obtain the estimator:

$$\mathcal{F}(\theta, \phi; x^{(i)}) \approx \frac{1}{L} \sum_{l=1}^{L} -\log q_\phi\big(g_\phi(\epsilon^{(l)}, x^{(i)})|x^{(i)}\big) + \log p_\theta\big(x^{(i)}, g_\phi(\epsilon^{(l)}, x^{(i)})\big) \tag{4.3}$$

Anomaly Scoring

Anomaly Scoring

How can we use this model to assess the anomalouness of some new sample $x^{(i)}$?

Anomaly Scoring

How can we use this model to assess the anomalouness of some new sample $x^{(i)}$?

1. Take L samples, $\{z^{(i,1)}, ..., z^{(i,L)}\}$ from the fitted variational distribution (i.e, the encoder), $q_\phi(z^{(i)}|x^{(i)})$.

Anomaly Scoring

How can we use this model to assess the anomalouness of some new sample $x^{(i)}$?

1. Take L samples, $\{z^{(i,1)}, ..., z^{(i,L)}\}$ from the fitted variational distribution (i.e, the encoder), $q_\phi(z^{(i)}|x^{(i)})$.
2. Each such sample, $z^{(i,l)}$, determines a specific form of the fitted likelihood (i.e. the decoder) by specifying its parameters,

$$p_\theta(x^{(i)} \mid z^{(i,l)}) = p_\theta\left(x^{(i)} \mid \mu_{x^{(i)}}(z^{(i,l)}) \quad , \quad \Sigma_{x^{(i)}}(z^{(i,l)})\right)$$

.

Anomaly Scoring

How can we use this model to assess the anomalouness of some new sample $x^{(i)}$?

1. Take L samples, $\{z^{(i,1)}, ..., z^{(i,L)}\}$ from the fitted variational distribution (i.e, the encoder), $q_\phi(z^{(i)}|x^{(i)})$.
2. Each such sample, $z^{(i,l)}$, determines a specific form of the fitted likelihood (i.e. the decoder) by specifying its parameters,
$$p_\theta(x^{(i)} \mid z^{(i,l)}) = p_\theta\left(x^{(i)} \mid \mu_{x^{(i)}}(z^{(i,l)}) \quad , \quad \Sigma_{x^{(i)}}(z^{(i,l)})\right)$$
.
3. Compute the *reconstruction probability* of the sample as the mean of these likelihoods:
$$\texttt{reconstruction probability}(x^{(i)}) := \frac{1}{L}\sum_{l=1}^{L} p_\theta\left(x^{(i)} \mid \mu_{x^{(i)}}(z^{(i,l)}), \Sigma_{x^{(i)}}(z^{(i,l)})\right)$$

An, J., & Cho, S. (2015). Variational autoencoder based anomaly detection using reconstruction probability. Special Lecture on IE, 2(1).

Evaluation

What do you think of this approach to anomaly detection?

Evaluation

What do you think of this approach to anomaly detection?

My thoughts on VAE vs. NF for anomaly detection:

- VAE's use of latent variables, and hence the need to jointly learn the generative and variational parameters, is awkward.
- Normalizing flows learn a single invertible map (from z-space to x-space) , rather than separately learning a function and its inverse.
- Thus:
  - The learned inverse is exact, rather than approximate
  - Inference is simpler – no latent variables (or variational inference) necessary!
  - Anomaly scores are exact, not approximate.
- Perhaps it's not surprising, then, that I have obtained higher quality results in practice for anomaly detection with NF's than VAE's.