Variational Autoencoders - An Introduction

Devon Graham

University of British Columbia
drgraham@cs.ubc.ca

Oct 31st, 2017

Table of contents

Introduction

- *Auto-Encoding Variational Bayes*, Diederik P. Kingma and Max Welling, ICLR 2014

Introduction

- *Auto-Encoding Variational Bayes*, Diederik P. Kingma and Max Welling, ICLR 2014
- Generative model

Introduction

- *Auto-Encoding Variational Bayes*, Diederik P. Kingma and Max Welling, ICLR 2014
- Generative model
- Running example: Want to generate realistic-looking MNIST digits (or celebrity faces, video game plants, cat pictures, etc)

Introduction

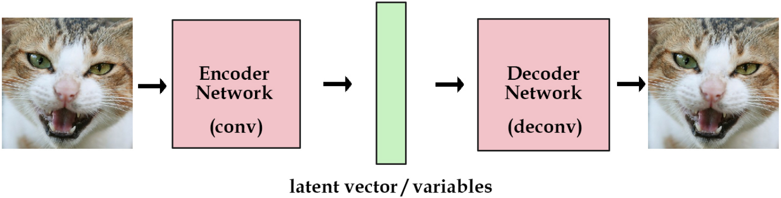
- *Auto-Encoding Variational Bayes*, Diederik P. Kingma and Max Welling, ICLR 2014
- Generative model
- Running example: Want to generate realistic-looking MNIST digits (or celebrity faces, video game plants, cat pictures, etc)
- `https://jaan.io/what-is-variational-autoencoder-vae-tutorial/`

Introduction

- *Auto-Encoding Variational Bayes*, Diederik P. Kingma and Max Welling, ICLR 2014
- Generative model
- Running example: Want to generate realistic-looking MNIST digits (or celebrity faces, video game plants, cat pictures, etc)
- `https://jaan.io/what-is-variational-autoencoder-vae-tutorial/`
- Deep Learning perspective and Probabilistic Model perspective

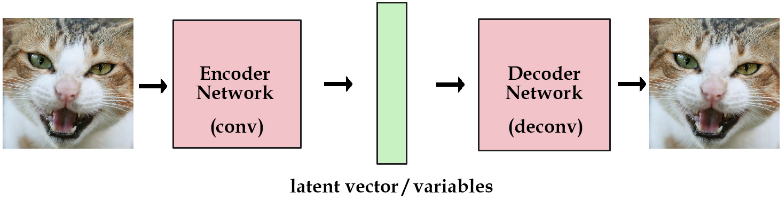
Introduction - Autoencoders

-



Introduction - Autoencoders

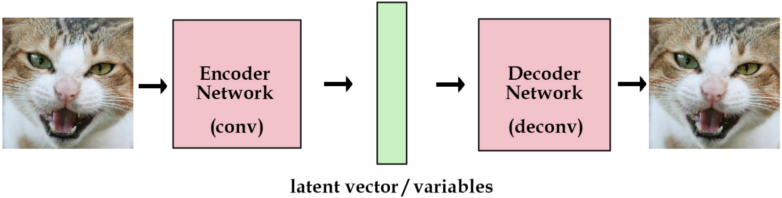
-



- Attempt to learn identity function

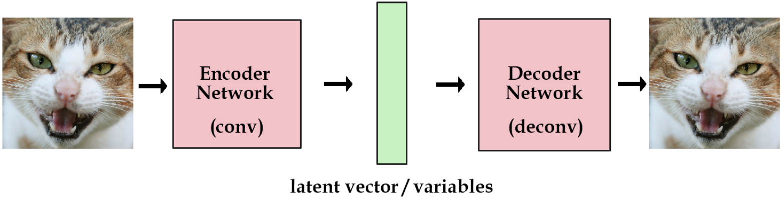
Introduction - Autoencoders

▶



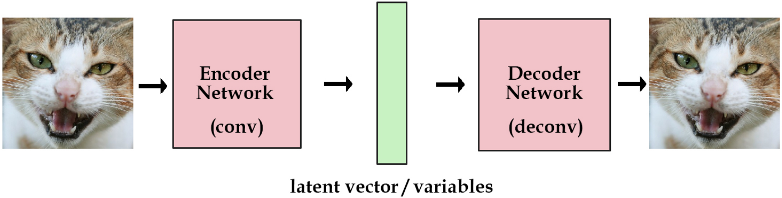
- Attempt to learn identity function
- Constrained in some way (e.g., small latent vector representation)

Introduction - Autoencoders

- 

- Attempt to learn identity function
- Constrained in some way (e.g., small latent vector representation)
- Can generate new images by giving different latent vectors to trained network

Introduction - Autoencoders

- 

- Attempt to learn identity function
- Constrained in some way (e.g., small latent vector representation)
- Can generate new images by giving different latent vectors to trained network
- Variational: use probabilistic latent encoding

Deep Learning Perspective

Deep Learning Perspective

- Goal: Build a neural network that generates MNIST digits from random (Gaussian) noise

Deep Learning Perspective

- Goal: Build a neural network that generates MNIST digits from random (Gaussian) noise
- Define two sub-networks: Encoder and Decoder

Deep Learning Perspective

- Goal: Build a neural network that generates MNIST digits from random (Gaussian) noise
- Define two sub-networks: Encoder and Decoder
- Define a Loss Function

Encoder

- A neural network $q_\theta(z|x)$

Encoder

- A neural network $q_\theta(z|x)$
- Input: datapoint x (e.g. 28×28-pixel MNIST digit)

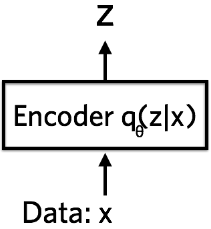
Encoder

- A neural network $q_\theta(z|x)$
- Input: datapoint x (e.g. 28×28-pixel MNIST digit)
- Output: encoding z, drawn from Gaussian density with parameters θ

Encoder

- A neural network $q_\theta(z|x)$
- Input: datapoint x (e.g. 28×28-pixel MNIST digit)
- Output: encoding z, drawn from Gaussian density with parameters θ
- $|z| \ll |x|$

Encoder

- A neural network $q_\theta(z|x)$
- Input: datapoint x (e.g. 28×28-pixel MNIST digit)
- Output: encoding z, drawn from Gaussian density with parameters θ
- $|z| \ll |x|$
- 

Decoder

- A neural network $p_\phi(x|z)$, parameterized by ϕ

Decoder

- A neural network $p_\phi(x|z)$, parameterized by ϕ
- Input: encoding z, output from encoder

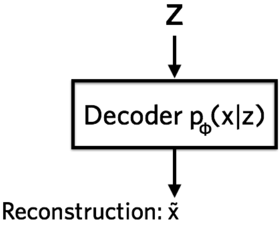
Decoder

- A neural network $p_\phi(x|z)$, parameterized by ϕ
- Input: encoding z, output from encoder
- Output: reconstruction $\tilde{x}$, drawn from distribution of the data

Decoder

- A neural network $p_\phi(x|z)$, parameterized by ϕ
- Input: encoding z, output from encoder
- Output: reconstruction $\tilde{x}$, drawn from distribution of the data
- E.g., output parameters for 28×28 Bernoulli variables

Decoder

- A neural network $p_\phi(x|z)$, parameterized by ϕ
- Input: encoding z, output from encoder
- Output: reconstruction $\tilde{x}$, drawn from distribution of the data
- E.g., output parameters for 28×28 Bernoulli variables
- 

Loss Function

- $\tilde{x}$ is reconstructed from z where $|z| \ll |\tilde{x}|$

Loss Function

- $\tilde{x}$ is reconstructed from z where $|z| \ll |\tilde{x}|$
- How much information is lost when we go from x to z to $\tilde{x}$?

Loss Function

- $\tilde{x}$ is reconstructed from z where $|z| \ll |\tilde{x}|$
- How much information is lost when we go from x to z to $\tilde{x}$?
- Measure this with reconstruction log-likelihood: $\log p_\phi(x|z)$

Loss Function

- $\tilde{x}$ is reconstructed from z where $|z| \ll |\tilde{x}|$
- How much information is lost when we go from x to z to $\tilde{x}$?
- Measure this with reconstruction log-likelihood: $\log p_\phi(x|z)$
- Measures how effectively the decoder has learned to reconstruct x given the latent representation z

Loss Function

- Loss function is negative reconstruction log-likelihood + regularizer

Loss Function

- Loss function is negative reconstruction log-likelihood + regularizer
- Loss decomposes into term for each datapoint:

$$L(\theta, \phi) = \sum_{i=1}^{N} l_i(\theta, \phi)$$

Loss Function

- Loss function is negative reconstruction log-likelihood + regularizer
- Loss decomposes into term for each datapoint:

$$L(\theta, \phi) = \sum_{i=1}^{N} l_i(\theta, \phi)$$

- Loss for datapoint x_i:

$$l_i(\theta, \phi) = -\mathbb{E}_{z \sim q_\theta(z|x_i)}\left[\log p_\phi(x_i|z)\right] + KL\left(q_\theta(z|x_i)||p(z)\right)$$

Loss Function

- Negative reconstruction log-likelihood:

$$-\mathbb{E}_{z \sim q_\theta(z|x_i)} \left[\log p_\phi(x_i|z) \right]$$

Loss Function

- Negative reconstruction log-likelihood:

$$-\mathbb{E}_{z \sim q_\theta(z|x_i)}\left[\log p_\phi(x_i|z)\right]$$

- Encourages decoder to learn to reconstruct the data

Loss Function

- Negative reconstruction log-likelihood:

$$-\mathbb{E}_{z \sim q_\theta(z|x_i)}\left[\log p_\phi(x_i|z)\right]$$

- Encourages decoder to learn to reconstruct the data
- Expectation taken over distribution of latent representations

Loss Function

- KL Divergence as regularizer:

$$KL\big(q_\theta(z|x_i)||p(z)\big) = \mathbb{E}_{z\sim q_\theta(z|x_i)}\big[\log q_\theta(z|x_i) - \log p(z)\big]$$

Loss Function

- KL Divergence as regularizer:

$$KL\big(q_\theta(z|x_i)||p(z)\big) = \mathbb{E}_{z\sim q_\theta(z|x_i)}\big[\log q_\theta(z|x_i) - \log p(z)\big]$$

- Measures information lost when using q_θ to represent p

Loss Function

- KL Divergence as regularizer:

$$KL\big(q_\theta(z|x_i)||p(z)\big) = \mathbb{E}_{z\sim q_\theta(z|x_i)}\big[\log q_\theta(z|x_i) - \log p(z)\big]$$

- Measures information lost when using q_θ to represent p
- We will use $p(z) = \mathcal{N}(\mathbf{0}, \mathbf{I})$

Loss Function

- KL Divergence as regularizer:

$$KL\big(q_\theta(z|x_i)||p(z)\big) = \mathbb{E}_{z\sim q_\theta(z|x_i)}\big[\log q_\theta(z|x_i) - \log p(z)\big]$$

- Measures information lost when using q_θ to represent p
- We will use $p(z) = \mathcal{N}(\mathbf{0}, \mathbf{I})$
- Encourages encoder to produce z's that are close to standard normal distribution

Loss Function

- KL Divergence as regularizer:

$$KL\big(q_\theta(z|x_i)||p(z)\big) = \mathbb{E}_{z\sim q_\theta(z|x_i)}\big[\log q_\theta(z|x_i) - \log p(z)\big]$$

- Measures information lost when using q_θ to represent p
- We will use $p(z) = \mathcal{N}(\mathbf{0}, \mathbf{I})$
- Encourages encoder to produce z's that are close to standard normal distribution
- Encoder learns a meaningful representation of MNIST digits

Loss Function

- KL Divergence as regularizer:

$$KL\big(q_\theta(z|x_i)||p(z)\big) = \mathbb{E}_{z\sim q_\theta(z|x_i)}\big[\log q_\theta(z|x_i) - \log p(z)\big]$$

- Measures information lost when using q_θ to represent p
- We will use $p(z) = \mathcal{N}(\mathbf{0}, \mathbf{I})$
- Encourages encoder to produce z's that are close to standard normal distribution
- Encoder learns a meaningful representation of MNIST digits
- Representation for images of the same digit are close together in latent space

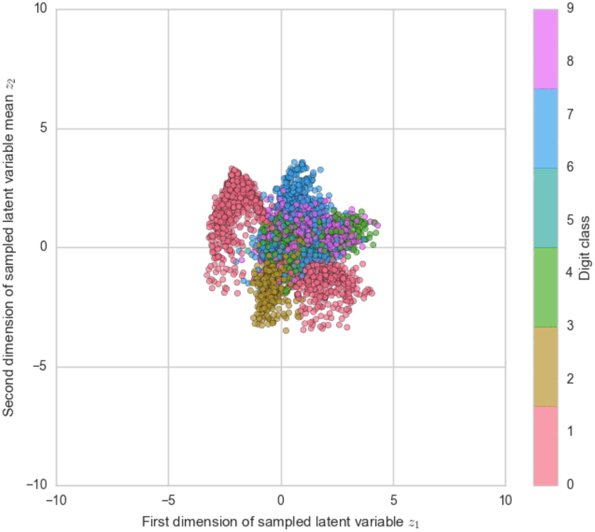
Loss Function

- KL Divergence as regularizer:

$$KL\big(q_\theta(z|x_i)||p(z)\big) = \mathbb{E}_{z\sim q_\theta(z|x_i)}\big[\log q_\theta(z|x_i) - \log p(z)\big]$$

- Measures information lost when using q_θ to represent p
- We will use $p(z) = \mathcal{N}(\mathbf{0}, \mathbf{I})$
- Encourages encoder to produce z's that are close to standard normal distribution
- Encoder learns a meaningful representation of MNIST digits
- Representation for images of the same digit are close together in latent space
- Otherwise could "memorize" the data and map each observed datapoint to a distinct region of space

MNIST latent variable space



Reparameterization trick

- We want to use gradient descent to learn the model's parameters

Reparameterization trick

- We want to use gradient descent to learn the model's parameters
- Given z drawn from $q_\theta(z|x)$, how do we take derivatives of (a function of) z w.r.t. θ?

Reparameterization trick

- We want to use gradient descent to learn the model's parameters
- Given z drawn from $q_\theta(z|x)$, how do we take derivatives of (a function of) z w.r.t. θ?
- We can reparameterize: $z = \mu + \sigma \odot \epsilon$

Reparameterization trick

- We want to use gradient descent to learn the model's parameters
- Given z drawn from $q_\theta(z|x)$, how do we take derivatives of (a function of) z w.r.t. θ?
- We can reparameterize: $z = \mu + \sigma \odot \epsilon$
- $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, and $\odot$ is element-wise product

Reparameterization trick

- We want to use gradient descent to learn the model's parameters
- Given z drawn from $q_\theta(z|x)$, how do we take derivatives of (a function of) z w.r.t. θ?
- We can reparameterize: $z = \mu + \sigma \odot \epsilon$
- $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, and $\odot$ is element-wise product
- Can take derivatives of (functions of) z w.r.t. μ and σ

Reparameterization trick

- We want to use gradient descent to learn the model's parameters
- Given z drawn from $q_\theta(z|x)$, how do we take derivatives of (a function of) z w.r.t. θ?
- We can reparameterize: $z = \mu + \sigma \odot \epsilon$
- $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, and $\odot$ is element-wise product
- Can take derivatives of (functions of) z w.r.t. μ and σ
- Output of $q_\theta(z|x)$ is vector of μ's and vector of σ's

Summary

- Deep Learning objective is to minimize the loss function:

$$L(\theta, \phi) = \sum_{i=1}^{N} \Big(-\mathbb{E}_{z \sim q_\theta(z|x_i)} \big[\log p_\phi(x_i|z) \big] + KL\big(q_\theta(z|x_i) || p(z)\big) \Big)$$

Probabilistic Model Perspective

Probabilistic Model Perspective

- Data x and latent variables z

Probabilistic Model Perspective

- Data x and latent variables z
- Joint pdf of the model: $p(x, z) = p(x|z)p(z)$

Probabilistic Model Perspective

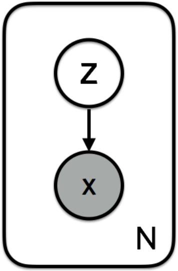
- Data x and latent variables z
- Joint pdf of the model: $p(x, z) = p(x|z)p(z)$
- Decomposes into likelihood: $p(x|z)$, and prior: $p(z)$

Probabilistic Model Perspective

- Data x and latent variables z
- Joint pdf of the model: $p(x, z) = p(x|z)p(z)$
- Decomposes into likelihood: $p(x|z)$, and prior: $p(z)$
- Generative process:
  Draw latent variables $z_i \sim p(z)$
  Draw datapoint $x_i \sim p(x|z)$

Probabilistic Model Perspective

- Data x and latent variables z
- Joint pdf of the model: $p(x, z) = p(x|z)p(z)$
- Decomposes into likelihood: $p(x|z)$, and prior: $p(z)$
- Generative process:
  Draw latent variables $z_i \sim p(z)$
  Draw datapoint $x_i \sim p(x|z)$
- Graphical model:



Probabilistic Model Perspective

- Suppose we want to do inference in this model

Probabilistic Model Perspective

- Suppose we want to do inference in this model
- We would like to infer good values of z, given observed data

Probabilistic Model Perspective

- Suppose we want to do inference in this model
- We would like to infer good values of z, given observed data
- Then we could use them to generate real-looking MNIST digits

Probabilistic Model Perspective

- Suppose we want to do inference in this model
- We would like to infer good values of z, given observed data
- Then we could use them to generate real-looking MNIST digits
- We want to calculate the posterior:

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

Probabilistic Model Perspective

- Suppose we want to do inference in this model
- We would like to infer good values of z, given observed data
- Then we could use them to generate real-looking MNIST digits
- We want to calculate the posterior:

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

- Need to calculate evidence: $p(x) = \int p(x|z)p(z)dz$

Probabilistic Model Perspective

- Suppose we want to do inference in this model
- We would like to infer good values of z, given observed data
- Then we could use them to generate real-looking MNIST digits
- We want to calculate the posterior:

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

- Need to calculate evidence: $p(x) = \int p(x|z)p(z)dz$
- Integral over all configurations of latent variables ☹

Probabilistic Model Perspective

- Suppose we want to do inference in this model
- We would like to infer good values of z, given observed data
- Then we could use them to generate real-looking MNIST digits
- We want to calculate the posterior:

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

- Need to calculate evidence: $p(x) = \int p(x|z)p(z)dz$
- Integral over all configurations of latent variables ☹
- Intractable

Probabilistic Model Perspective

- Variational inference to the rescue!

Probabilistic Model Perspective

- Variational inference to the rescue!
- Let's approximate the true posterior $p(z|x)$ with the 'best' distribution from some family $q_\lambda(z|x)$

Probabilistic Model Perspective

- Variational inference to the rescue!
- Let's approximate the true posterior $p(z|x)$ with the 'best' distribution from some family $q_\lambda(z|x)$
- Which choice of λ gives the 'best' $q_\lambda(z|x)$?

Probabilistic Model Perspective

- Variational inference to the rescue!
- Let's approximate the true posterior $p(z|x)$ with the 'best' distribution from some family $q_\lambda(z|x)$
- Which choice of λ gives the 'best' $q_\lambda(z|x)$?
- KL divergence measures information lost when using q_λ to approximate p

Probabilistic Model Perspective

- Variational inference to the rescue!
- Let's approximate the true posterior $p(z|x)$ with the 'best' distribution from some family $q_\lambda(z|x)$
- Which choice of λ gives the 'best' $q_\lambda(z|x)$?
- KL divergence measures information lost when using q_λ to approximate p
- Choose λ to minimize $KL\big(q_\lambda(z|x)||p(z|x)\big) = KL\big(q_\lambda||p\big)$

Probabilistic Model Perspective

- $$\begin{aligned} KL\big(q_\lambda || p\big) &:= \mathbb{E}_{z\sim q_\lambda}\big[\log q_\lambda(z|x) - \log p(z|x)\big] \\ &= \mathbb{E}_{z\sim q_\lambda}\big[\log q_\lambda(z|x)\big] - \mathbb{E}_{z\sim q_\lambda}\big[\log p(x,z)\big] \\ &\qquad + \log p(x) \end{aligned}$$

Probabilistic Model Perspective

- $$\begin{aligned} KL\big(q_\lambda \| p\big) &:= \mathbb{E}_{z\sim q_\lambda}\big[\log q_\lambda(z|x) - \log p(z|x)\big] \\ &= \mathbb{E}_{z\sim q_\lambda}\big[\log q_\lambda(z|x)\big] - \mathbb{E}_{z\sim q_\lambda}\big[\log p(x,z)\big] \\ &\quad + \log p(x) \end{aligned}$$

- Still contains $p(x)$ term! So cannot compute directly

Probabilistic Model Perspective

- $$\begin{aligned} KL(q_\lambda || p) &:= \mathbb{E}_{z \sim q_\lambda}\big[\log q_\lambda(z|x) - \log p(z|x)\big] \\ &= \mathbb{E}_{z \sim q_\lambda}\big[\log q_\lambda(z|x)\big] - \mathbb{E}_{z \sim q_\lambda}\big[\log p(x,z)\big] \\ &\quad + \log p(x) \end{aligned}$$

- Still contains $p(x)$ term! So cannot compute directly
- But $p(x)$ does not depend on λ, so still hope

Probabilistic Model Perspective

- Define **E**vidence **L**ower **BO**und:

$$ELBO(\lambda) := \mathbb{E}_{z \sim q_\lambda}\big[\log p(x, z)\big] - \mathbb{E}_{z \sim q_\lambda}\big[\log q_\lambda(z|x)\big]$$

Probabilistic Model Perspective

- Define **E**vidence **L**ower **BO**und:

$$ELBO(\lambda) := \mathbb{E}_{z\sim q_\lambda}\left[\log p(x,z)\right] - \mathbb{E}_{z\sim q_\lambda}\left[\log q_\lambda(z|x)\right]$$

- Then

$$\begin{aligned} KL(q_\lambda || p) &= \mathbb{E}_{z\sim q_\lambda}\left[\log q_\lambda(z|x)\right] - \mathbb{E}_{z\sim q_\lambda}\left[\log p(x,z)\right] + \log p(x) \\ &= -ELBO(\lambda) + \log p(x) \end{aligned}$$

Probabilistic Model Perspective

- Define **E**vidence **L**ower **BO**und:

$$ELBO(\lambda) := \mathbb{E}_{z\sim q_\lambda}\big[\log p(x,z)\big] - \mathbb{E}_{z\sim q_\lambda}\big[\log q_\lambda(z|x)\big]$$

- Then

$$\begin{aligned} KL\big(q_\lambda || p\big) &= \mathbb{E}_{z\sim q_\lambda}\big[\log q_\lambda(z|x)\big] - \mathbb{E}_{z\sim q_\lambda}\big[\log p(x,z)\big] + \log p(x) \\ &= -ELBO(\lambda) + \log p(x) \end{aligned}$$

- So minimizing $KL\big(q_\lambda || p\big)$ w.r.t. λ is equivalent to maximizing $ELBO(\lambda)$

Probabilistic Model Perspective

- Since no two datapoints share latent variables, we can write:

$$ELBO(\lambda) = \sum_{i=1}^{N} ELBO_i(\lambda)$$

Probabilistic Model Perspective

- Since no two datapoints share latent variables, we can write:

$$ELBO(\lambda) = \sum_{i=1}^{N} ELBO_i(\lambda)$$

- Where

$$ELBO_i(\lambda) = \mathbb{E}_{z \sim q_\lambda(z|x_i)}\big[\log p(x_i, z)\big] - \mathbb{E}_{z \sim q_\lambda(z|x_i)}\big[\log q_\lambda(z|x_i)\big]$$

Probabilistic Model Perspective

- We can rewrite the term $ELBO_i(\lambda)$:

$$
\begin{aligned}
ELBO_i(\lambda) &= \mathbb{E}_{z\sim q_\lambda(z|x_i)}\Big[\log p(x_i, z)\Big] - \mathbb{E}_{z\sim q_\lambda(z|x_i)}\Big[\log q_\lambda(z|x_i)\Big] \\
&= \mathbb{E}_{z\sim q_\lambda(z|x_i)}\Big[\log p(x_i|z) + \log p(z)\Big] \\
&\qquad - \mathbb{E}_{z\sim q_\lambda(z|x_i)}\Big[\log q_\lambda(z|x_i)\Big] \\
&= \mathbb{E}_{z\sim q_\lambda(z|x_i)}\Big[\log p(x_i|z)\Big] \\
&\qquad - \mathbb{E}_{z\sim q_\lambda(z|x_i)}\Big[\log q_\lambda(z|x_i) - \log p(z)\Big] \\
&= \mathbb{E}_{z\sim q_\lambda(z|x_i)}\Big[\log p(x_i|z)\Big] - KL\Big(q_\lambda(z|x_i)||p(z)\Big)
\end{aligned}
$$

Probabilistic Model Perspective

- How do we relate λ to ϕ and θ seen earlier?

Probabilistic Model Perspective

- How do we relate λ to ϕ and θ seen earlier?
- We can parameterize approximate posterior $q_\theta(z|x,\lambda)$ by a network that takes data x and outputs parameters λ

Probabilistic Model Perspective

- How do we relate λ to ϕ and θ seen earlier?
- We can parameterize approximate posterior $q_\theta(z|x,\lambda)$ by a network that takes data x and outputs parameters λ
- Parameterize the likelihood $p(x|z)$ with a network that takes latent variables and outputs parameters to the data distribution $p_\phi(x|z)$

Probabilistic Model Perspective

- How do we relate λ to ϕ and θ seen earlier?
- We can parameterize approximate posterior $q_\theta(z|x, \lambda)$ by a network that takes data x and outputs parameters λ
- Parameterize the likelihood $p(x|z)$ with a network that takes latent variables and outputs parameters to the data distribution $p_\phi(x|z)$
- So we can re-write

$$ELBO_i(\theta, \phi) = \mathbb{E}_{z \sim q_\theta(z|x_i)}\big[\log p_\phi(x_i|z)\big] - KL\big(q_\theta(z|x_i)||p(z)\big)$$

Probabilistic Model Objective

- Recall the Deep Learning objective derived earlier. We want to minimize:

$$L(\theta, \phi) = \sum_{i=1}^{N} \Big(- E_{z \sim q_\theta(z|x_i)} \big[\log p_\phi(x_i|z) \big] + KL\big(q_\theta(z|x_i) || p(z)\big) \Big)$$

Probabilistic Model Objective

- Recall the Deep Learning objective derived earlier. We want to minimize:

$$L(\theta, \phi) = \sum_{i=1}^{N} \Big(- E_{z \sim q_\theta(z|x_i)} \big[\log p_\phi(x_i|z) \big] + KL\big(q_\theta(z|x_i) || p(z)\big) \Big)$$

- The objective just derived for the Probabilistic Model was to maximize:

$$ELBO(\theta, \phi) = \sum_{i=1}^{N} \Big(\mathbb{E}_{z \sim q_\theta(z|x_i)} \big[\log p_\phi(x_i|z) \big] - KL\big(q_\theta(z|x_i) || p(z)\big) \Big)$$

- They are equivalent!

Applications - Image generation

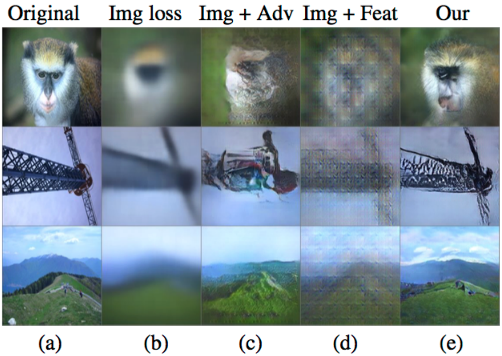


Figure 1: Reconstructions from AlexNet FC6 with different components of the loss.

A. Dosovitskiy and T. Brox. Generating images with perceptual similarity metrics based on deep networks. *arXiv preprint arXiv*:1602.02644, 2016.

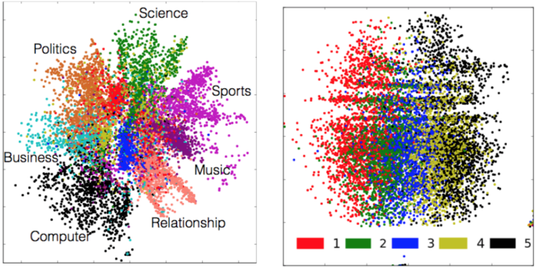
Applications - Caption generation

a man with a snowboard next to a man with glasses | a big black dog standing on the grass | a player is holding a hockey stick | a desk with a keyboard | a man is standing next to a brown horse | a box full of apples and oranges

Figure 2: Examples of generated caption from unseen images on the validation dataset of ImageNet.

Y. Pu, Z. Gan, R. Henao, X. Yuan, C. Li, A. Stevens, and L. Carin. Variational autoencoder for deep learning of images, labels and captions. In NIPS, 2016.

Applications - Semi-/Un-supervised document classification



(a) Yahoo (b) Yelp

Figure 3: Visualizations of learned latent representations.

Z. Yang, Z. Hu, R. Salakhutdinov, and T. Berg-Kirkpatrick. Improved variational autoencoders for text modeling using dilated convolutions. In *Proceedings of The 34rd International Conference on Machine Learning*, 2017.

Applications - Pixel art videogame characters

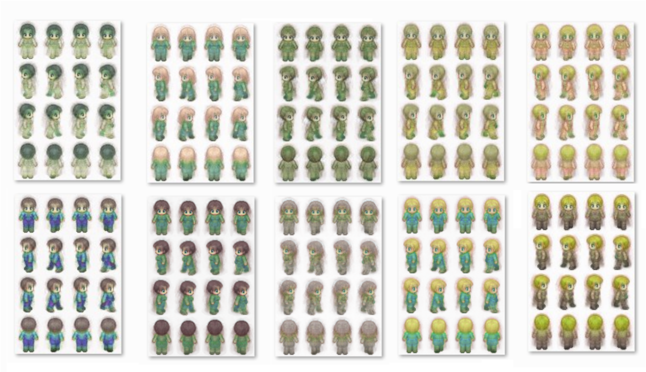


Figure 6: Samples of the generated characters

https://mlexplained.wordpress.com/category/generative-models/vae/.

Conclusion

- We derived the same objective from

Conclusion

- We derived the same objective from
- 1) A deep learning point of view, and

Conclusion

- We derived the same objective from
- 1) A deep learning point of view, and
- 2) A probabilistic models point of view

Conclusion

- We derived the same objective from
- 1) A deep learning point of view, and
- 2) A probabilistic models point of view
- Showed they are equivalent

Conclusion

- We derived the same objective from
- 1) A deep learning point of view, and
- 2) A probabilistic models point of view
- Showed they are equivalent
- Saw some applications

Conclusion

- We derived the same objective from
- 1) A deep learning point of view, and
- 2) A probabilistic models point of view
- Showed they are equivalent
- Saw some applications
- Thank you. Questions?