Lecture 12: Generic neural density estimation

TTIC 31220: Unsupervised Learning and Data Analysis

Instructor: Karen Livescu

TTI-Chicago

May 9, 2017

Important dates

- 5/9 Homework 2 & project proposal (1 page) due
- 5/16 (Brief!) project proposal feedback due
- 5/18 No lecture (Midwest Robotics Workshop)
- 5/23 Homework 3 & project update (1 page) due
- 5/30 Project presentations
- 6/8 Project final report (4 pages) due

Homework 2 reminders/clarifications

- Some leaderboard submissions use SVM classifier by mistake instead of cluster labels – let us know if you did this, and submit again
- Please give a precise mathematical definition for each technique you use (e.g., which variant of spectral clustering?)
- Part 1 (feedback) is required!

(Rough) lecture plan

- Introduction (1)
- Dimensionality reduction/representation learning (4.5)
- Clustering, topic modeling, mixtures, EM (3.5)
- Fixed (human-engineered) feature representations (1)
- **Density estimation, hidden Markov models (3)**
  - Gaussian mixtures, topic models
  - Language models: n-grams, neural LMs
  - HMMs, forward-backward
  - Hidden topic Markov models
  - Today: "generic" neural density estimation
- Semi-supervised learning, distant supervision (2)
- Computer vision applications (1)
- Speech, language, and other sequential data (1)
- Project presentations (1-2)

Summary of course topics

Tasks

- Representation learning
  - Training: Data set $x_i \in \mathcal{A} \longrightarrow y_i \in \mathbf{R}^d$ or $\longrightarrow f(\cdot) : \mathcal{A} \to \mathbf{R}^d$
  - Testing: Data point $x \in \mathcal{A} \longrightarrow f(x) \in \mathbf{R}^d$
- Clustering
  - Training: Data set $x_i \in \mathcal{A} \longrightarrow y_i \in [1 \dots K]$ or $\longrightarrow f(\cdot) : \mathcal{A} \to [1 \dots K]$
  - Testing: Data point $x \in \mathcal{A} \longrightarrow f(x) \in [1 \dots K]$
- Density estimation
  - Training: Data set $x_i \in \mathcal{A} \longrightarrow p(x)$
  - Testing: $p(x) \longrightarrow$ sample, or sample $x \longrightarrow p(x)$
- Fourier representations

Some connections between topics...

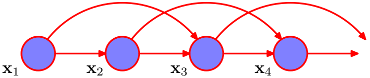
- Clustering as representation learning: vector quantization
- Representation learning for clustering: spectral clustering
- Fourier methods for representation "learning": random Fourier features
- Density estimation as clustering: mixture models

Density estimation thus far

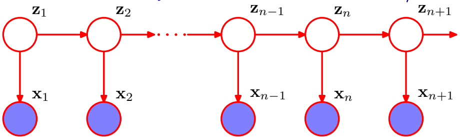
We have covered density estimation for certain narrow classes of densities

- Mixture models: single (1-dim) discrete latent variable, continuous vector observed variable

- Language models: sequence of discrete (1-dim) observed variables (e.g. trigram)



- Hidden Markov models: sequence of discrete (1-dim) hidden variables, sequence of continuous/discrete observed variables



Density estimation: General graphical models

- We would like to generalize to arbitrary combinations of (continuous- or discrete-valued) vector latent variables and vector observations
- Can be viewed as the same graphical model as mixture models, but latent variables are arbitrary:

Density estimation: What for?

- To understand the data
- To give a score (probability/density) to a new example
- To generate new examples
- As a form of representation learning (see today's lecture)

Neural density estimation

Two main types:

- Network outputs $p(\mathbf{x})$ (e.g., neural language models)
- Network outputs parameters of $p(\mathbf{x})$ (e.g., means and variances)

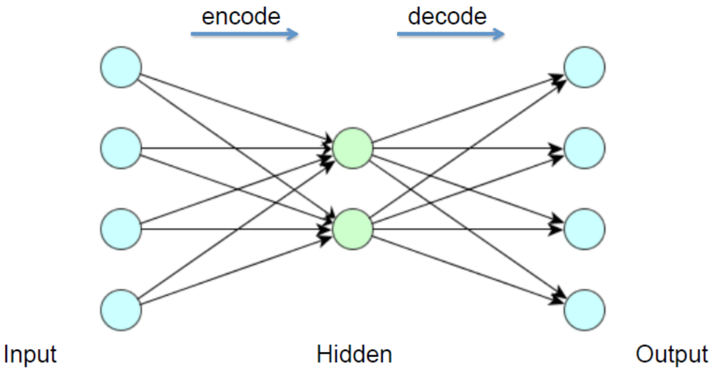
Variational autoencoders for density estimation

- More or less simultaneously proposed by Kingma & Welling (2013) and Rezende *et al.* (2014)
- Popular, fast and relatively easy to train
- Generative model for a vector of random variables $\mathbf{x}$ assumed to be generated from a set of latent variables $\mathbf{z}$: $p(\mathbf{x}) = \int_{\mathbf{z}} p(\mathbf{x}|\mathbf{z})p(\mathbf{z})d\mathbf{z}$
- For today, assume $\mathbf{x}, \mathbf{z}$ are both continuous
- No "meaning" to $\mathbf{z}$ (unlike in some other latent variable models, e.g. topic models)
- Typically used to generate, but has also been used for representation learning

Variational autoencoders for density estimation

- "Autoencoders" because they compute a density of $\mathbf{z}$ given $\mathbf{x}$ via an *encoder* and a density of $\mathbf{x}$ given $\mathbf{z}$ via a *decoder*
- "Variational" because they involve approximating a density via optimization

Reminder: Autoencoders

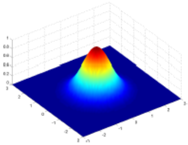


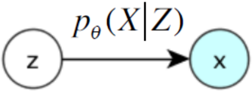
[Dürr 2016]

VAE generation model (decoder)

$z \sim p(z)$ multivariate Gaussian

$x|z \sim p_\theta(x|z)$

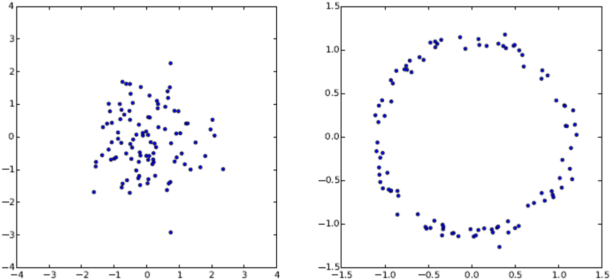




[Dürr 2016]

VAE generation model (decoder)

Key insight: Any d-dimensional distribution can be generated by taking d normally distributed variables and mapping them through some appropriate (possibly very complicated) function

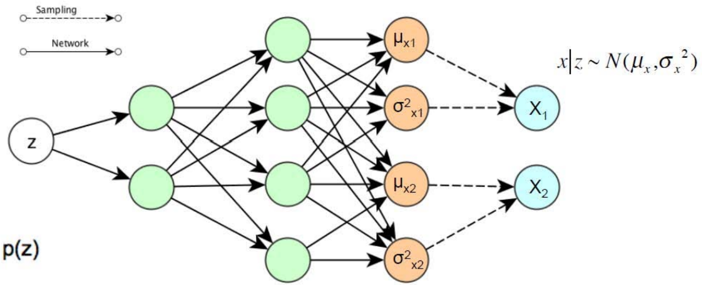


[Doersch 2016]

Here $x = g(z) = z/10 + z/||z||$

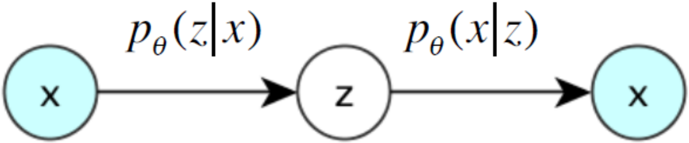
VAE neural decoder

Assume w.l.o.g. that latent variable z is 1-dimensional, x is 2-dimensional



[Dürr 2016]

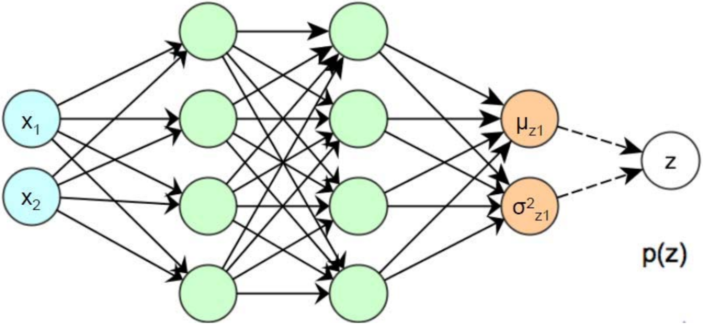
VAE encoder-decoder model



[Dürr 2016]

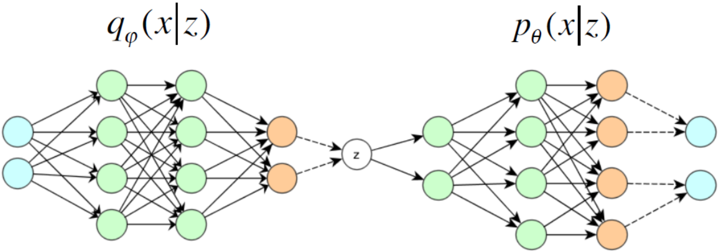
- But $p(z|x)$ can be very costly to estimate
- We will approximate it with another function $p(z|x)$

VAE neural encoder



[Dürr 2016]

Complete VAE



[Dürr 2016]

Learning: Maximizing (a lower bound on the) likelihood

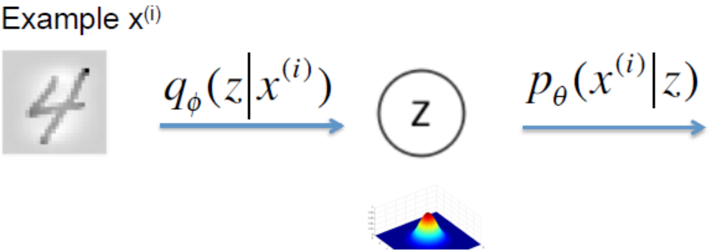
$$
\begin{array}{rcl}
L & = & \log p(x) \\
\text{multiply by 1...} & = & \displaystyle\sum_z q(z|x) \log p(x) \\
 & = & \displaystyle\sum_z q(z|x) \log \left(\frac{p(z,x)}{p(z|x)} \right) \\
\text{multiply by 1...} & = & \displaystyle\sum_z q(z|x) \log \left(\frac{p(z,x)}{q(z|x)} \frac{q(z|x)}{p(z|x)} \right) \\
 & = & \displaystyle\sum_z q(z|x) \log \left(\frac{p(z,x)}{q(z|x)} \right) + \sum_z q(z|x) \log \left(\frac{q(z|x)}{p(z|x)} \right) \\
 & = & L^v + D_{KL}(q(z|x)||p(z|x)) \\
 & \geq & L^v
\end{array}
$$

Rewriting the lower bound...

$$
\begin{aligned}
L^v &= \sum_z q(z|x) \log\left(\frac{p(z,x)}{q(z|x)}\right) \\
&= \sum_z q(z|x) \log\left(\frac{p(x|z)p(z)}{q(z|x)}\right) \\
&= \sum_z q(z|x) \log\left(\frac{p(z)}{q(z|x)}\right) + \sum_z q(z|x) \log p(x|z) \\
&= -D_{KL}\left(q(z|x)||p(z)\right) + E_{q(z|x)}\left(\log p(x|z)\right) \\
\text{for } x_i\ldots &= -D_{KL}\left(q(z|x_i)||p(z)\right) + E_{q(z|x_i)}\left(\log p(x_i|z)\right)
\end{aligned}
$$

- First term: Regularizer; prior $p(z)$ is usually taken to be $\mathcal{N}(0,1)$
- Second term: Reconstruction loss; equals $\log(1)$ if x_i is perfectly reconstructed from z

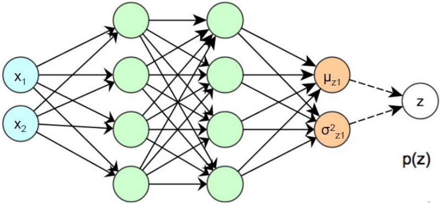
Computing the lower bound for a training example



[Dürr 2016]

Computing the lower bound for a training example

KL divergence term:



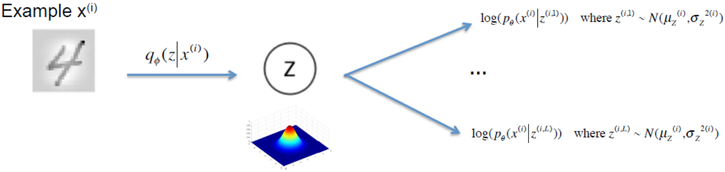
[Dürr 2016]

The KL divergence has a closed form when $p(z) = \mathcal{N}(0, 1)$ and $q(z|x)$ is also Gaussian:

$$-D_{KL}\left(q(z|x_i)||p(z)\right) = \frac{1}{2}\sum_{j=1}^{J} 1 + \log\left(\sigma^2_{z_{i,j}}\right) - \mu^2_{z_{i,j}} - \sigma^2_{z_{i,j}}$$

Computing the lower bound for a training example

Reconstruction term:



[Dürr 2016]

Approximate the expectation by sampling B samples from $q(z|x_i)$:

$$\begin{aligned} L^v &= -D_{KL}\left(q(z|x)||p(z)\right) + E_{q(z|x)}\left(\log p(x|z)\right) \\ &\approx -D_{KL}\left(q(z|x)||p(z)\right) + \frac{1}{B}\sum_{l=1}^{B}\left(\log p(x_i|z_{i,l})\right) \end{aligned}$$

Often just use $B = 1$

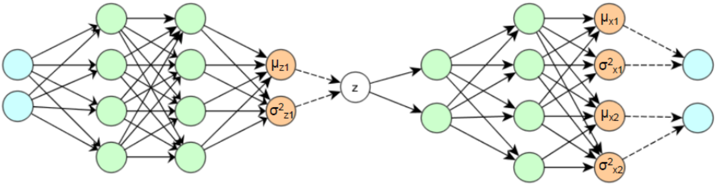
Computing the lower bound for a training example

Reconstruction term when $B = 1$ and $p(x|z)$ is Gaussian:

$$(\log p(x_i|z_i)) = \sum_{j=1}^{D} \frac{1}{2} \log \sigma_{x_j}^2 + \frac{\left(x_{i,j} - \mu_{x_j}\right)^2}{2\sigma_{x_j}^2}$$

This is just a least squares loss!

Putting it all together...



Cost: Regularisation

$$-D_{KL}\left(q(z|x^{(i)})||p(z)\right) = \frac{1}{2}\sum_{j=1}^{J}\left(1 + \log(\sigma_{z_j}^{(i)^2}) - \mu_{z_j}^{(i)^2} - \sigma_{z_j}^{(i)^2}\right)$$

We use mini batch gradient decent to optimize the cost function over all $x^{(i)}$ in the mini batch

Cost: Reproduction

$$-\log\left(p(x^{(i)}|z^{(i)})\right) = \sum_{j=1}^{D}\frac{1}{2}\log(\sigma_{x_j}^2) + \frac{(x_j^{(i)} - \mu_{x_j})^2}{2\sigma_{x_j}^2}$$

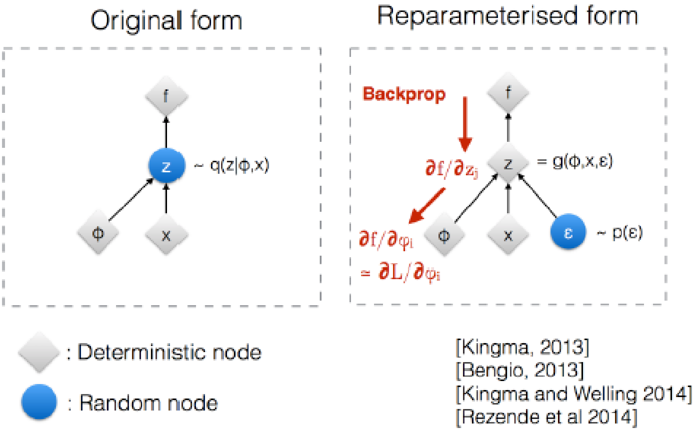
Least Square for constant variance

[Dürr 2016]

Can learn all parameters via backpropagation... or can we?

Reparameterization trick

- Not so fast... can't backpropagate through the random sampling
- Instead we will use the "reparameterization trick"

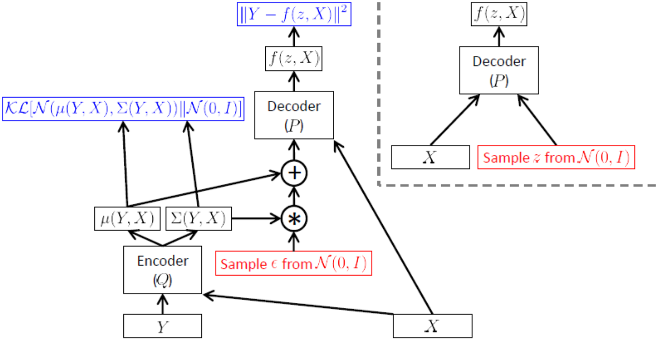


[Kingma 2015]

$$z = \mu_x + \sigma^{1/2} x \times \epsilon$$

Extension: Conditional VAE

Learns a different distribution for each (typically discrete) value of a conditioning variable (note change of notation... sorry!)



[Doersch 2016]