Unsupervised & Generative DL I – Explicit Density Learning

INTELLIGENT SYSTEMS FOR PATTERN RECOGNITION (ISPR)

DAVIDE BACCIU – DIPARTIMENTO DI INFORMATICA - UNIVERSITA' DI PISA

DAVIDE.BACCIU@UNIPI.IT
Lecture Outline

○ Motivation
  ● Why unsupervised?
  ● Why generative?

○ The DL way to generative learning
  ● Learning distributions with fully visible information (**RNN**)
  ● Learning distributions with latent information (**VAE**)

○ Applications
  ● Generating faces and bedrooms
  ● Latent space arithmetic
The Problem

○ Characterize the data
  ● Data distribution
  ● Data variances

○ To allow
  ● Understanding data
  ● Generating new observations
  ● ..and ultimately reasoning

Autoencoders and Manifold Learning
The Problem

Labelled data is **costly and difficult** to obtain

A **sustainable future** for deep learning

- Learning the **latent structure** of data
- Discover important features
- Learn **task independent** representations
- Introduce (if any) supervision only on few samples
Why Generative?

- Focusing too much on discrimination rather than on characterizing data can cause issues
  - Reduced interpretability
  - Adversarial examples

- Generative models (try to) characterize data distribution
  - Understand the data ⇒ Understand the world
  - Understand data variances ⇒ Learn to steer them
  - Understand normality ⇒ Detect anomalies
Approaching the Problem from a DL Perspective

*Given training data, learn a (deep) neural network that can generate new samples from (an approximation of) the data distribution*
Approaching the Problem from a DL Perspective

Given training data, learn a (deep) neural network that can generate new samples from (an approximation of) the data distribution

Two approaches

- Explicit $\Rightarrow$ Learn a model density $P_\theta(x)$
- Implicit $\Rightarrow$ Learn a process that samples data from $P_\theta(x) \approx P(x)$
A Taxonomy

Generative DL

- Explicit
  - Visible
    - Sampling RNN
    - Flow-based
  - Latent
    - Variational
      - Variational AEs
      - Diffusion Model
    - Stochastic
      - Boltzmann
      - Machines
  - Intractable densities
- Implicit
  - Direct
    - Generative Adversarial Networks
  - Stochastic
    - Generative Stochastic Networks
  - Tractable densities

Adapted from I. Goodfellow, Tutorial on Generative Adversarial Networks, 2017
Learning with Fully Visible Information

If all information is fully visible the joint distribution can be computed from the \textit{chain rule factorization}

\[
P(x) = \prod_{i=1}^{N} P(x_i | x_1, \ldots, x_{i-1})
\]

- Probability of a pixel having a certain intensity value, given the known intensity of its predecessor
- Need to be able to define a sensible ordering for the chain rule
- Conditional distribution difficult to compute

Bayesian Networks
Approximating the Conditional Probability

If all information is fully visible the joint distribution can be computed from the chain rule factorization

Scan the image according to a schedule and encode the dependency from previous pixels in the states of an RNN
Approximating the Conditional Probability

If all information is fully visible the joint distribution can be computed from the chain rule factorization.

Scan the image according to a schedule and encode the dependency from previous pixels in the states of an RNN.
Approximating the Conditional Probability

If all information is fully visible the joint distribution can be computed from the chain rule factorization.

Scan the image according to a schedule and encode the dependency from previous pixels in the states of an RNN.
Approximating the Conditional Probability

If all information is fully visible the joint distribution can be computed from the chain rule factorization.

Scan the image according to a schedule and encode the dependency from previous pixels in the states of an RNN.
Approximating the Conditional Probability

If all information is fully visible the joint distribution can be computed from the chain rule factorization.

Scan the image according to a schedule and encode the dependency from previous pixels in the states of an RNN.
Generating Images Pixel by Pixel

PixelCNN
Row LSTM
Diagonal BiLSTM

A. van der Oord et al., Pixel Recurrent Neural Networks, 2016
Generating Images Pixel by Pixel - Results

A. van der Oord et al., Pixel Recurrent Neural Networks, 2016
From Visible to Latent Information

With **only visible information**, we try to learn the $\theta$ parameterized model distribution

$$P_\theta(x) = \prod_{i}^{N} P_\theta(x_i|x_1, \ldots, x_{i-1})$$

Now we introduce a latent process regulated by unobservable variables $z$

$$P_\theta(x) = \int P_\theta(x|z)P_\theta(z)dz$$

Typically **intractable** for nontrivial models (cannot be computed for all $z$ assignments)
A Neural Network with Latent Variables?

We have already introduced a probabilistic twist on AE.

Autoencoder (AE) neural networks
As an additional push in the probabilistic interpretation, we assume to be able to generate the reconstruction from a sampled latent representation.

Of course we don’t have access to the true distributions, so how do we approximate them?
Variational Autoencoders (VAE) – The Catch

\[ \tilde{x} \]

Decoder \( g \)

\[ z \]

Represent the \( P(\tilde{x}|z) \) distribution through a neural network \( g \) (remember the denoising autoencoder)

Sample \( z \) from a simple distribution such as a Gaussian

\[ z \sim \mathcal{N}(\mu(x), \sigma(x)) \]

At training time sample \( z \) conditioned on data \( x \) and train the decoder \( g \) to reconstruct \( x \) itself from \( z \)
VAE Training – Is it all this easy?

Unfortunately for you: no!

Ideally, one would like to train maximizing

\[
L(D) = \prod_{i=1}^{N} P(x_i) = \prod_{i=1}^{N} \int P(x_i|z)P(z)dz
\]

Intractable \hspace{1cm} Non differentiable

Variational approximation \hspace{1cm} Reparameterization
Reparameterization Trick

\[ \tilde{x} \]
\[ z \sim \mathcal{N}(\mu(x), \sigma(x)) \]
\[ \mu(x) \quad \sigma(x) \]

Non-differentiable operation

Sampling is limited to non differentiable variable \( \epsilon \) \( \Rightarrow \) Can backpropagate

\[ \epsilon \sim \mathcal{N}(0,1) \]
Variational Approximation

The revenge of the ELBO (Evidence Lower BOund)

$$\log P(x|\theta) \geq \mathbb{E}_Q[\log P(x,z)] - \mathbb{E}_Q[\log Q(z)] = \mathcal{L}(x, \theta, \phi)$$

Maximizing the ELBO allows approximating from below the intractable log-likelihood \(\log P(x)\)

$$\mathcal{L}(x, \theta, \phi) = \mathbb{E}_Q[\log P(x|z)] + \mathbb{E}_Q[\log P(z)] - \mathbb{E}_Q[\log Q(z)]$$

Decoder estimate of the conditional, made possible and differentiable through the reparameterization trick

Need a \(Q(z)\) function to approximate \(P(z)\)
Variational Autoencoder – The Full Picture

Encoder network $Q(z|x, \phi)$ with parameters $\phi$

Decoder network $P(\tilde{x}|z, \theta)$ with parameters $\theta$

Encoder Q

$x$

$\tilde{x}$

$z$

$\mu(x)$ $\sigma(x)$

Sample $\epsilon \sim \mathcal{N}(0,1)$

Training time architecture

$\mathcal{N}(0,1)$
VAE Training

Training is performed by backpropagation on $\theta, \phi$ to optimize the ELBO

$$
\mathcal{L}(x, \theta, \phi) = \mathbb{E}_Q \left[ \log P(x|z = \mu(x) + \sigma^{1/2}(x) * \epsilon, \theta) \right] - KL(Q(z|x, \phi) || P(z|\theta))
$$

Can be computed in closed form when both $Q(z)$ and $P(z)$ are Gaussians

$$
KL(\mathcal{N}(\mu(x), \sigma(x)) || \mathcal{N}(0,1))
$$

Train the encoder to behave like a Gaussian prior with zero-mean and unit-variance
VAE Final Loss

In principle we would like to optimize the following loss by SGD

$$\mathbb{E}_{X \sim D} \left[ \mathbb{E}_{z \sim Q} \left[ \log P(x|z) \right] - KL(Q(z|x, \phi)||P(z)) \right]$$

which can be rearranged following the reparametrization trick

$$\mathbb{E}_{X \sim D} \left[ \mathbb{E}_{\epsilon \sim \mathcal{N}(0,1)} \left[ \log P(x|z = \mu(x) + \sigma^{1/2}(x) \ast \epsilon, \theta) \right] - KL(Q(z|x, \phi)||P(z)) \right]$$

No expectation is w.r.t distributions that depend on model parameters

⇒ We can move gradients into them
Information Theoretic Interpretation

$$\mathbb{E}_{X \sim D}[\mathbb{E}_{Z \sim Q}[\log P(x|z)] - KL(Q(z|x, \phi)||P(z))]$$

Number of bits required to reconstruct $x$ from $z$ under the ideal encoding (i.e. $Q(z|x)$ is generally suboptimal)

Number of bits required to convert an uninformative sample from $P(z)$ into a sample from $Q(z|x)$

Information gain - Amount of extra information that we get about $X$ when $z$ comes from $Q(z|x)$ instead of from $P(z)$
Sampling the VAE (a.k.a. testing)

At test time detach the encoder, sample a random encoding and generate the sample as the corresponding reconstruction.

\[
\tilde{x} \\
\text{Decoder } g \\
\text{Sample } z \sim \mathcal{N}(0,1)
\]
VAE vs Denoising/Contractive AE

Contractive AE

Variational AE

x noisy samples

x original sample
VAE Examples - Digits

Organization of data in the latent space

Reconstruction of points sampled from latent space

Image credits @ fastforwardlabs.com
VAE Examples - Faces

Hou et al, Deep Feature Consistent Variational Autoencoder, 2017

Latent space interpolation
Conditional Generation (CVAE)

\[ \tilde{x} \]

\[ \text{Decoder } g \]

\[ \text{Encoder } Q \]

\[ x \]

Training

\[ z \]

Sample \( \epsilon \sim \mathcal{N}(0,1) \)

\[ y \]

Inference

\[ \tilde{x} \]

\[ \text{Decoder } g \]

Sample \( z \sim \mathcal{N}(0,1) \)

Learns the conditional distribution \( P(x|y) \) (this is the simplest possible form of CVAE)
Take Home Messages

○ PixelRNN/ PixelCNN – Learn explicit distributions by optimizing exact likelihood
  ● Yields good samples and excellent likelihood estimates
  ● Inefficient sequential generation

○ VAE – Learn complex distributions over latent variables through a variational approximation using neural networks
  ● Learns a latent representation useful for inference
  ● Can lead to poor generated sample quality
Next Lecture

- Learning a sampling process
- Generative adversarial networks
- Hybrid Variational-Adversarial approaches