Computational Intelligence Laboratory

Lecture 8
Generative Models

Thomas Hofmann

ETH Zurich – cil.inf.ethz.ch

24 April 2020

Section 1

Motivation

Faces

What have all these faces in common?

Faces

What have all these faces in common?

They are not faces of real people, but generated.
(taken from StyleGAN (NVIDIA), March 2020)

Voices

What is special about his voice?

News reading

Voices

What is special about his voice?

News reading

It is a computer voice (Amazon Alexa).

Alexa's news-reading voice just got a lot more professional

Alexa now knows which words to emphasize in a sentence

By Jon Porter | @JonPorty | Jan 16, 2019, 11:29am EST

TECH AMAZON ARTIFICIAL INTELLIGENCE

Amazon's Alexa gets a new longform speaking style

With more natural-sounding pauses, the style is intended for longer-form content like podcasts

By Kim Lyons | Apr 16, 2020, 5:45pm EDT

Synthesis vs. Analysis

Using **Synthesis** (or generation)

... rather than **Analysis** (or recognition)

... opens up vastly new possibilities for machine learning

... animation, games, movies, art, mixed reality

Section 2

Variational Autoencoders

Deep Generative Models

Key idea: use power of DNNs to create **complex distributions**

- Sample (simple) random vector, e.g. $\mathbb{R}^m \ni \mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$
  - $\mathcal{N}(0, \mathbf{I})$ = standard normal distribution in m dimensions
- Transform through a (deterministic) deep network $F_\theta : \mathbb{R}^m \to \mathbb{R}^n$
- Induces a (possibly complex) distribution over $\mathbb{R}^n$ w/ parameters θ
  - sample $\mathbf{x}$ by sampling $\mathbf{z}$ and setting $\mathbf{x} = F_\theta(\mathbf{z})$
  - expectations $\mathbf{E}_\mathbf{x}[f(\mathbf{x})] = \mathbf{E}_\mathbf{z}\left[f(F_\theta(\mathbf{z}))\right]$
    $=$ **law of the unconscious statistician**

Can this be made to work?

Deep Generative Models

- If F is invertible: density is given by change of variables formula

$$\mathbf{x} = F_\theta(\mathbf{z}), \qquad \underbrace{p_x(\mathbf{x})}_{\mathbf{x}\text{-density}} = \left| \frac{\partial F_\theta^{-1}(\mathbf{x})}{\partial \mathbf{x}} \right| \underbrace{p_z(F_\theta^{-1}(\mathbf{x}))}_{\mathbf{z}\text{-density}}$$

  - would require network inversion to find pre-image $\mathbf{z} \mapsto F_\theta(\mathbf{z}) \stackrel{!}{=} \mathbf{x}$
  - would require computation of (inverse) Jacobian determinant
  - would also need to compute gradients with respect to θ to learn
  - often impossible (non-invertible), intractable/imprating (dimensionality) $\Longrightarrow$ often not viable to construct density

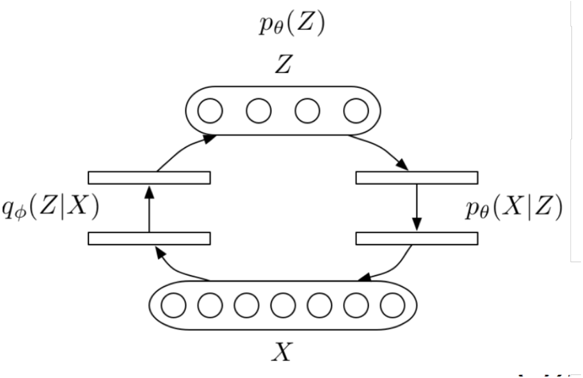
ELBO: Evidence Lower BOund

- Slightly more general: $p_\theta(\mathbf{x}|\mathbf{z})$ (instead of determinsitic F_θ)
- Marginal likelihood $p_\theta(\mathbf{x}) = \int p_\theta(\mathbf{x}|\mathbf{z})p(\mathbf{z})d\mathbf{z}$
- **Variational lower bound**

$$\begin{aligned}\log p_\theta(\mathbf{x}) \geq \mathsf{ELBO}(\phi,\theta) &= \mathbf{E}_{q_\phi}\left[\log p_\theta(\mathbf{x}|\mathbf{z}) + \log \frac{p(\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})}\right] \\ &= \mathbf{E}_{q_\phi}\left[\log p_\theta(\mathbf{x}|\mathbf{z})\right] - \mathsf{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z}))\end{aligned}$$

  - KL $=$ Kullback-Leibler divergence
  - maximize w.r.t. θ (generative model, given q_ϕ)
  - maximize w.r.t. ϕ (inference model, given p_θ)

Variational Autoencoder: Diagram



(courtesy of Teh, NIPS 2017)

ELBO: Generative Model Updates

- Update step for generative model, **stochastic approximation**

$$\nabla_\theta \, \mathbf{E}_{q_\phi} \left[\log p_\theta(\mathbf{x}|\mathbf{z})\right] = \mathbf{E}_{q_\phi} \left[\nabla_\theta \; \log p_\theta(\mathbf{x}|\mathbf{z})\right] \quad \text{(Leibniz integral rule)}$$

$$\approx \frac{1}{L} \sum_{r=1}^{L} \nabla_\theta \; \log p_\theta(\mathbf{x}|\mathbf{z}^{(r)}), \quad \mathbf{z}^{(r)} \overset{iid}{\sim} q_\phi(\cdot|\mathbf{x})$$

  - unbiased gradient estimate (SGD)
  - similar to supervised learning (input $\mathbf{z}$, output $\mathbf{x}$)
  - Gaussian observation model $\equiv$ squared error $\frac{1}{2}(F(\mathbf{z}) - \mathbf{x})^2$
  - inference model performs approximate model inversion

ELBO: Inference Model Updates (1 of 2)

- Update step for inference model:

$$\nabla_\phi \, \mathbf{E}_{q_\phi} \left[\mathcal{L}(\mathbf{x}, \mathbf{z}) \right] = \int \mathcal{L}(\mathbf{x}, \mathbf{z}) \, \nabla_\phi \, q_\phi(\mathbf{z}; \mathbf{x}) d\mathbf{z} = \; \mathbf{E}[\;\; ? \;\;]$$

- **Reinforce trick** (Williams 1992)

$$\nabla_\phi \, \mathbf{E}_{q_\phi} \left[\mathcal{L}(\mathbf{x}, \mathbf{z}) \right] = \mathbf{E}_{q_\phi} \left[\mathcal{L}(\mathbf{x}, \mathbf{z}) \nabla_\phi \log q_\phi(\mathbf{z}; \mathbf{x}) \right]$$

  - as $\nabla q = q \nabla \log q$
  - can be estimated via sampling, but ...
  - variance usually very high $\Longrightarrow$ impractically large number of samples

ELBO: Inference Model Updates (2 of 2)

- **Re-parameterization trick**: use variational distributions such that

$$q_\phi(\mathbf{z};\mathbf{x}) = g_\phi(\zeta;\mathbf{x}), \quad \zeta \sim \text{simple distribution (e.g. } \mathcal{N}(\mathbf{0},\mathbf{I}))$$

- Gradient of expectation can be converted into expectation of gradient (**stochastic backpropagation**)

$$\begin{aligned}\nabla_\phi \, \mathbf{E}_{q_\phi}\left[\mathcal{L}(\mathbf{x},\mathbf{z})\right] &= \mathbf{E}_\zeta\left[\nabla_\phi \, \mathcal{L}(\mathbf{x}, g_\phi(\zeta))\right] \\ &\approx \frac{1}{L}\sum_{r=1}^{L}\left[\nabla_\phi \, \mathcal{L}(\mathbf{x}, g_\phi(\zeta^{(r)}))\right], \quad \zeta^{(r)} \overset{iid}{\sim} \text{simple}\end{aligned}$$

- Example:

$$\zeta \sim \mathcal{N}(\mathbf{0},\mathbf{I}), \; \mathbf{z} = \mu + \mathbf{U}\zeta, \text{then } \; \mathbf{z} \sim \mathcal{N}(\mu, \mathbf{U}\mathbf{U}^\top)$$

- (It is often observed that this leads to lower variance estimates.)

Deep Latent Gaussian Models: Generative Model

- Noise variables

$$\mathbf{z}^l \overset{iid}{\sim} \mathcal{N}(\mathbf{0}, \mathbf{I}), \quad l = 1, \ldots, L$$

- Hidden activities (latent random variables, top-down indexed)

$$\mathbf{x}^L = \mathbf{W}^L \mathbf{z}^L, \quad \mathbf{x}^l = \underbrace{F^l(\mathbf{x}^{l+1})}_{\text{deterministic}} + \underbrace{\mathbf{W}^l \mathbf{z}^l}_{\text{stochastic}}$$

- Hidden layer (conditional) distribution

$$\mathbf{x}^l | \mathbf{x}^{l+1} \sim \mathcal{N}\left(F^l(\mathbf{x}^{l+1}), \mathbf{W}^l {\mathbf{W}^l}^\top\right)$$

- Generated pattern $\mathbf{x} \sim \pi(\mathbf{x}^1)$ (observation/noise model with parameters $\mathbf{x}^1$)

Deep Latent Gaussian Models: Inference Model

- Inference model (amortized inference), $\mathbf{z} = (\mathbf{z}^1, \ldots, \mathbf{z}^L)$

$$q_\phi(\mathbf{z}; \mathbf{x}) = \prod_{l=1}^{L} \mathcal{N}(\mathbf{z}^l | \mu^l(\mathbf{x}), \mathbf{C}(\mathbf{x})), \quad \mathbf{C}(\mathbf{x}) = \mathbf{U}(\mathbf{x})\mathbf{U}(\mathbf{x})^\top$$

where μ and $\mathbf{U}$ are represented by DNNs with input $\mathbf{x}$.

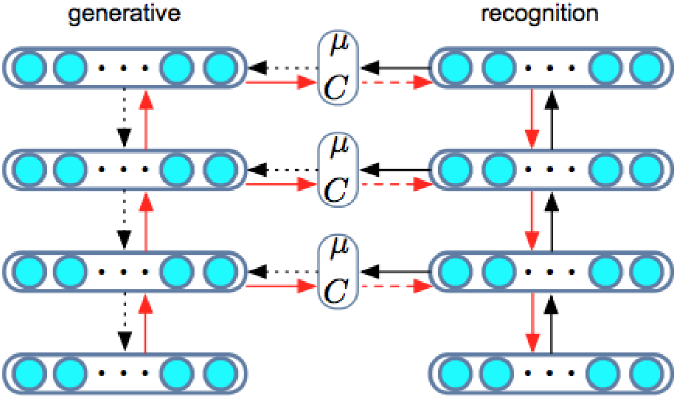
- Update equations can use the Bonnet formula (for $\mathbf{z} \sim \mathcal{N}(\mu, \mathbf{C})$):

$$\nabla_\mu \mathbf{E}[f(\mathbf{z})] = \mathbf{E}[\nabla_\mathbf{z} f(\mathbf{z})]$$

- Similar equation for $\mathbf{U}$ (as an alternative to Price's theorem) Rezende et al. 2014: **Gaussian backpropagation**

$$\nabla_\mathbf{U} \mathbf{E}_\mathbf{z}[f(\mathbf{z})] = \nabla_\mathbf{U} \mathbf{E}_\zeta[f(\mathbf{U}\zeta + \mu)] = \mathbf{E}_\zeta[\zeta^\top \mathbf{g}], \quad \mathbf{g} := \nabla_\xi f(\xi)_{|\xi = \mathbf{U}\zeta + \mu}$$

Variational Autoencoder: Learning Scheme



(from Rezende at al. 2014; Notation: $\mathbf{U} = \mathbf{C}$)

VAE: Examples

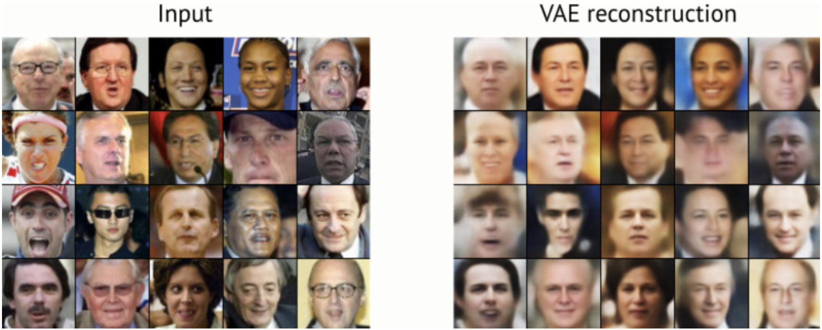
- Face generation

from http://torch.ch/blog/2015/11/13/gan.html

  - not bad, but blurry
  - (often) better training: Generative Adverserial Networks (GANs)

VAE: Examples

- Face reconstruction/denoising



from `http://torch.ch/blog/2015/11/13/gan.html`

Section 3

Generative Adversarial Network

From Optimal Discrimination to Generation

- Proposed by Goodfellow et al., 2014
- Classification problem: **distinguish between data & model**.
  Define joint distribution as mixture

$$\tilde{p}_\theta(\mathbf{x}, y=1) = \tfrac{1}{2}p(\mathbf{x}), \quad \tilde{p}_\theta(\mathbf{x}, y=0) = \tfrac{1}{2}p_\theta(\mathbf{x})\,.$$

- Bayes optimal classifier: posterior

$$q_\theta = p/(p + p_\theta)\,.$$

- Train generator via **minimizing** the logistic likelihood

$$\theta \xrightarrow{\min} \ell^*(\theta) := \mathbf{E}_{\tilde{p}_\theta}\left[y \ln q_\theta(\mathbf{x}) + (1-y)\ln(1 - q_\theta(\mathbf{x}))\right]$$

  - generator's goal: generate samples that are **indistinguishable from real data**, even for the best possible classifier
  - can be shown to be equivalent to **Jensen-Shannon divergence**: $\ell^*(\theta) = \mathsf{JS}(p, p_\theta) - \ln 2$

From Real Discriminaton to Generation

- Optimal classifier is in general inaccessible
- Instead: define a **classification model**

$$q_\phi : \mathbf{x} \mapsto [0; 1], \quad \phi \in \Phi$$

- Define objective via bound

$$\ell^*(\theta) \geq \sup_{\phi \in \Phi} \ell(\theta, \phi)$$

$$\ell(\theta, \phi) := \mathbf{E}_{\tilde{p}_\theta} \left[y \ln q_\phi(\mathbf{x}) + (1 - y) \ln(1 - q_\phi(\mathbf{x})) \right]$$

  - find best classifier within restricted family
  - typically: Φ = weight space of DNN
  - training objective for generator is defined implicitly over $\sup$

Optimizing GANs

- Saddle-point problem

$$\theta^* := \arg\min_{\theta \in \Theta} \{\sup_{\phi \in \Phi} \ell(\theta, \phi)\}$$

  - explicitly performing inner $\sup$ is impractical
  - various methods from optimization / solving games

- SGD as a heuristic (may diverge!)

$$\theta^{t+1} = \theta^t - \eta \nabla_\theta \ell(\theta^t, \phi^t)$$

$$\phi^{t+1} = \phi + \eta \nabla_\phi \ell(\theta^{t+1}, \phi^t)$$

- Ongoing research: find better optimization methods

Section 4

Autoregressive Models

Approaches to Learn a Generative Model

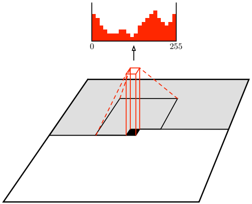
- Variational Autoencoders (VAEs), Generative Adversarial Networks (GANs): complicated learning method; not always successful
- Simpler strategy: Autoregressive models - generate output one variable at a time
  - justified by chain rule: $p(x_1, \ldots, x_m) = \prod_{t=1}^{m} p(x_t | x_{1:t-1})$
- Example: PixelCNN (A. van den Oord et al. 2016)
  - generative model for images
  - network models conditional distribution of every individual pixel given previous pixels (to the left and to the top).
- Similar approaches used for speech (e.g. WaveNet)

Pixel CNN

- Model joint distribution of pixels over image $\mathbf{x}$ as product of conditional distributions, where x_i is a single pixel:

$$p(\mathbf{x}) = \prod_{i=1}^{n^2} p(x_i | x_1, ..., x_{i-1})$$

- Visualization on the left: generate pixel x_i by conditioning on previously generated pixels $x_1, \ldots x_{i-1}$
- Ordering of the pixel dependencies is in raster scan order: row by row and pixel by pixel within every row



Pixel CNN

- Need to make sure the CNN can only use information about pixels above and to the left of the current pixel
- Used to mask the 5x5 filters to make sure the model cannot read pixels below (or strictly to the right) of the current pixel to make its predictions

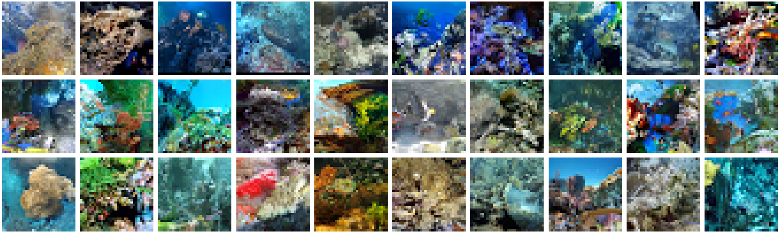
1	1	1	1	1
1	1	1	1	1
1	1	0	0	0
0	0	0	0	0
0	0	0	0	0

Prediction with Pixel CNN

- During sampling the predictions are sequential: every time a pixel is predicted, it is fed back into the network to predict the next pixel
- Drawback: Slow process

Image Generation with Pixel CNN

African elephant



Coral Reef