Deep Generative Models

Sebastian Nowozin
Microsoft Research
\( \epsilon \rightarrow f \rightarrow y \)
\[ y = f(x, \varepsilon) \]
Taxonomy

Non-probabilistic

\[ x \xrightarrow{f} y \]

Probabilistic

Generative

\[ \epsilon \xrightarrow{f} y \]

Discriminative

“Conditionally Generative”

\[ x \xrightarrow{\epsilon \cdot f} y \]
NVIDIA’s Progressive GANs [Karras et al., 2018]

Generating non-image data

Representation learning [Zheng et al., 2018]
Learning Probabilistic Models
Learning Probabilistic Models

Assumptions on $P$:

• tractable sampling
• tractable parameter gradient with respect to sample
• tractable likelihood function
Principles of Density Estimation

Integral Probability Metrics
[Müller, 1997]
[Sriperumbudur et al., 2010]

\[ \gamma_F(P, Q) = \sup_{f \in F} \left| \int f \, dP - \int f \, dQ \right| \]

Proper scoring rules
[Gneiting and Raftery, 2007]

\[ S(P, Q) = \int S(P, x) dQ(x) \]

\[ D_f(P \parallel Q) = \int q(x) f \left( \frac{p(x)}{q(x)} \right) dx \]

- Kernel MMD
- Wasserstein GANs

- Variational Autoencoders
- Autoregressive models
- DISCO networks

- Generative adversarial networks
- $f$-GAN, $b$-GAN

- $f$-divergences
[Ali and Silvey, 1966],
[Nguyen et al., 2010]
Variational Autoencoders (VAE)

[Kingma and Welling, 2014], [Rezende et al., 2014]
Variational Autoencoders

```python
def encode(self, x):
    h = F.crelu(self.qlin0(x))
    h = F.crelu(self.qlin1(h))
    h = F.crelu(self.qlin2(h))
    h = F.crelu(self.qlin3(h))

    self.qmu = self.qlin_mu(h)
    self.qln_var = self.qlin_ln_var(h)

def decode(self, z):
    h = F.crelu(self.plin0(z))
    h = F.crelu(self.plin1(h))
    h = F.crelu(self.plin2(h))
    h = F.crelu(self.plin3(h))

    self.pmu = self.plin_mu(h)
    self.pln_var = self.plin_ln_var(h)

def __call__(self, x):
    # Compute q(z|x)
    self.encode(x)

    self.kl = gaussian_kl_divergence(self.qmu, self.qln_var)
    self.logp = 0
    for j in range(self.num_samples):
        # z ~ q(z|x)
        z = F.gaussian(self.qmu, self.qln_var)

        # Compute p(x|z)
        self.decode(z)

        # Compute objective
        self.logp += gaussian_logp(x, self.pmu, self.pln_var)

    self.logp /= self.num_samples
    self.obj = self.kl - self.logp

    return self.obj
```
Maximum Likelihood Estimation (MLE)
[Fisher, 1929]

• MLE extremely successful:
  e.g. least squares and cross-entropy are MLE estimators
\[ p(x|\theta) \]
Maximize the \textit{likelihood} function

\[
L(\theta) = \prod_{i} p(x_i|\theta)
\]

\[
\log L(\theta) = \log \prod_{i} p(x_i|\theta)
\]

\[
\log L(\theta) = \sum_{i} \log p(x_i|\theta)
\]
VAE: Model

\[
p(x|\theta) = \int p(x|z, \theta) p(z) dz
\]

Example
- \( p(z) \) is a multivariate standard Normal
- \( p(x|z, \theta) \) is a neural network outputting a simple distribution (e.g. diagonal Normal)
VAE: Maximum Likelihood Training

- Maximize the data log-likelihood, per-instance variational approximation

\[
\log p(x|\theta) = \log \int p(x|z, \theta)p(z)dz
\]
\[
= \log \int p(x|z, \theta)\frac{q(z)}{q(z)}p(z)dz
\]
\[
= \log \int p(x|z, \theta)\frac{p(z)}{q(z)}q(z)dz
\]
\[
= \log \mathbb{E}_{z \sim q(z)} \left[ p(x|z, \theta)\frac{p(z)}{q(z)} \right]
\]
\[
\geq \mathbb{E}_{z \sim q(z)} \left[ \log p(x|z, \theta)\frac{p(z)}{q(z)} \right]
\]
\[
= \mathbb{E}_{z \sim q(z)} [\log p(x|z, \theta)] - D_{KL}(q(z) \parallel p(z))
\]
Inference networks

\[ x_1, x_2, x_3 \]

\[ q_1(z), q_2(z), q_3(z) \]
Inference networks

• Amortized inference [Stuhlmüller et al., NIPS 2013]
• Inference networks, recognition networks [Kingma and Welling, 2014]
• “Informed sampler” [Jampani et al., 2014]
• “Memory-based approach” [Kulkarni et al., 2015]
VAE: Maximum Likelihood Training

- Maximize the data log-likelihood, inference network variational approximation

\[
\log p(x | \theta) = \log \int p(x | z, \theta) p(z) dz \\
= \log \int p(x | z, \theta) \frac{q(z | x, w)}{q(z | x, w)} p(z) dz \\
= \log \int p(x | z, \theta) \frac{p(z)}{q(z | x, w)} q(z | x, w) dz \\
= \log \mathbb{E}_{z \sim q(z | x, w)} \left[ p(x | z, \theta) \frac{p(z)}{q(z | x, w)} \right] \\
\geq \mathbb{E}_{z \sim q(z | x, w)} \left[ \log p(x | z, \theta) \frac{p(z)}{q(z | x, w)} \right] \\
= \mathbb{E}_{z \sim q(z | x, w)} [\log p(x | z, \theta)] - D_{KL}(q(z | x, w) \parallel p(z))
\]
Autoencoder viewpoint

$$\max_{w,\theta} \mathbb{E}_{z \sim q(z|x,w)} \left[ \log p(x|z, \theta) \right] - D_{KL}(q(z|x, w) \parallel p(z))$$
Reparametrization Trick

- [Rezende et al., 2014]
- [Kingma and Welling, 2014]
Reparametrization Trick

- [Rezende et al., 2014]
  [Kingma and Welling, 2014]
- Stochastic computation graphs
  [Schulman et al., 2015]
(Live Azure Demo)

https://notebooks.azure.com/nowozin/libraries/variational-autoencoder
Problems in VAEs

• Inadequate inference networks
  • Loose ELBO
  • Limits what the generative model can learn

• Parametric conditional likelihood assumptions
  • Limits the expressivity of the generative model
  • “Noise term has to explain too much”

• No control over latent representation that is learned
“Blurry images” in VAE models from [Tulyakov et al., 2017]
Improving Inference Networks

- State of the art in inference network design:
  - NICE [Dinh et al., 2015]
  - Hamiltonian variational inference (HVI) [Salimans et al., 2015]
  - Importance weighted autoencoder (IWAE) [Burda et al., 2016]
  - Normalizing flows [Rezende and Mohamed, 2016]
  - Auxiliary deep generative networks [Maaløe et al., 2016]
  - Inverse autoregressive flow (IAF) [Kingma et al., NIPS 2016]
  - Householder flows [Tomczak and Welling, 2017]
  - Adversarial variational Bayes (AVB) [Mescheder et al., 2017]
  - Deep and Hierarchical Implicit Models [Tran et al., 2017]
  - Variational Inference using Implicit Distributions [Huszár, 2017]
  - Adversarial Message Passing for Graphical Models [Karaletsos, 2016]
  - Jackknife Variational Inference [Nowozin, 2018]
Generative Adversarial Networks

[Goodfellow et al., NIPS 2014], [Nowozin, Cseke, Tomioka, NIPS 2016]
GAN = Implicit Models + Estimation procedure
Classic parametric models

- Density function available
- Limited expressive power
- Mature field in statistics and learning theory
Implicit Model / Neural Sampler / Likelihood-free Model

- Highly expressive model class
- Density function not defined or intractable
- Lack of theory and learning algorithms
- Basis for generative adversarial networks (GANs)
Implicit Models

2. Goodfellow et al. (2014). Generative Adversarial Nets. *NIPS*
Implicit models as building blocks

• For inference (as in AVB), or
• As model component, or
• As regularizer
Implicit Models: Problem 1

\[ p(x) = \int_{z \in G^{-1}(x)} \mu(z) \, dz \]
Implicit Models: Problem 2

\[ p(x) \text{ not defined a.e.} \]
GAN Training Objective [Goodfellow et al., 2014]

- Generator tries to fool discriminator (i.e. generate realistic samples)
- Discriminator tries to distinguish fake from real samples
- Saddle-point problem

\[
\min_{\theta} \max_{\omega} \mathbb{E}_{x \sim P_{\theta}} [\log D_{\omega}(x)] + \mathbb{E}_{x \sim Q} [\log(1 - D_{\omega}(x))]
\]
Progress of GANs, 2013-2018

[Goodfellow et al., 2013] University of Montreal

[Radford et al., 2015] Facebook AI Research

[Roth et al., 2017] Microsoft and ETHZ

[Karras et al., 2018] NVIDIA
Natural Images (Radford et al., 2015, arXiv:1511.06434)
“DCGAN” architecture
Linear interpolation in latent space [Radford et al., 2015]
Estimating $f$-divergences from samples

- Divergence between two distributions

\[ D_f(P \parallel Q) = \int_{x} q(x) f \left( \frac{p(x)}{q(x)} \right) dx \]

- Every convex function $f$ has a Fenchel conjugate $f^*$ so that

\[ f(u) = \sup_{t \in \text{dom } f^*} \{ tu - f^*(t) \} \]

“any convex $f$ can be represented as point-wise max of linear functions”
Estimating $f$-divergences from samples (cont)

$$D_f(P \parallel Q) = \int_X q(x) f \left( \frac{p(x)}{q(x)} \right) \, dx$$

$$= \int_X q(x) \sup_{t \in \text{dom} f^*} \left\{ t \frac{p(x)}{q(x)} - f^*(t) \right\} \, dx$$

$$\geq \sup_{T \in \mathcal{T}} \left( \int_X p(x) T(x) \, dx - \int_X q(x) f^*(T(x)) \, dx \right)$$

$$= \sup_{T \in \mathcal{T}} \left( \mathbb{E}_{x \sim P} [T(x)] - \mathbb{E}_{x \sim Q} [f^*(T(x))] \right)$$

Approximate using: samples from $P$ samples from $Q$

[Nguyen, Wainwright, Jordan, 2010]
$f$-GAN and GAN objectives

- GAN
  $$\min_{\theta} \max_{\omega} \mathbb{E}_{x \sim P_\theta}[\log D_\omega(x)] + \mathbb{E}_{x \sim Q}[\log(1 - D_\omega(x))]$$

- $f$-GAN
  $$\min_{\theta} \max_{\omega} \left(\mathbb{E}_{x \sim P_\theta}[T_\omega(x)] - \mathbb{E}_{x \sim Q}[f^*(T_\omega(x))]\right)$$

- GAN discriminator-variational function correspondence: $\log D_\omega(x) = T_\omega(x)$
- GAN minimizes the Jensen-Shannon divergence (which was also pointed out in Goodfellow et al., 2014)
## $f$-divergences

<table>
<thead>
<tr>
<th>Name</th>
<th>$D_f(P|Q)$</th>
<th>Generator $f(u)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total variation</td>
<td>$\frac{1}{2} \int</td>
<td>p(x) - q(x)</td>
</tr>
<tr>
<td>Kullback-Leibler</td>
<td>$\int p(x) \log \frac{p(x)}{q(x)} , dx$</td>
<td>$u \log u$</td>
</tr>
<tr>
<td>Reverse Kullback-Leibler</td>
<td>$\int q(x) \log \frac{q(x)}{p(x)} , dx$</td>
<td>$- \log u$</td>
</tr>
<tr>
<td>Pearson $\chi^2$</td>
<td>$\int \frac{(q(x) - p(x))^2}{p(x)} , dx$</td>
<td>$(u - 1)^2$</td>
</tr>
<tr>
<td>Neyman $\chi^2$</td>
<td>$\int \frac{(p(x) - q(x))^2}{q(x)} , dx$</td>
<td>$\frac{(1-u)^2}{u}$</td>
</tr>
<tr>
<td>Squared Hellinger</td>
<td>$\int \left( \sqrt{p(x)} - \sqrt{q(x)} \right)^2 , dx$</td>
<td>$(\sqrt{u} - 1)^2$</td>
</tr>
<tr>
<td>Jeffrey</td>
<td>$\int (p(x) - q(x)) \log \left( \frac{p(x)}{q(x)} \right) , dx$</td>
<td>$(u - 1) \log u$</td>
</tr>
<tr>
<td>Jensen-Shannon</td>
<td>$\frac{1}{2} \int p(x) \log \frac{2p(x)}{p(x)+q(x)} + q(x) \log \frac{2q(x)}{p(x)+q(x)} , dx$</td>
<td>$-(u + 1) \log \frac{1+u}{2} + u \log u$</td>
</tr>
<tr>
<td>Jensen-Shannon-weighted</td>
<td>$\int p(x) \pi \log \frac{p(x)}{\pi p(x) + (1-\pi)q(x)} + (1 - \pi)q(x) \log \frac{q(x)}{\pi p(x) + (1-\pi)q(x)} , dx$</td>
<td>$\pi u \log u - (1 - \pi + \pi u) \log(1 - \pi + \pi u)$</td>
</tr>
<tr>
<td>GAN</td>
<td>$\int p(x) \log \frac{2p(x)}{p(x)+q(x)} + q(x) \log \frac{2q(x)}{p(x)+q(x)} , dx - \log(4)$</td>
<td>$u \log u - (u + 1) \log(u + 1)$</td>
</tr>
<tr>
<td>$\alpha$-divergence ($\alpha \notin {0, 1}$)</td>
<td>$\frac{1}{\alpha(\alpha-1)} \int \left( p(x) \left[ \left( \frac{q(x)}{p(x)} \right)^\alpha - 1 \right] - \alpha(q(x) - p(x)) \right) , dx$</td>
<td>$\frac{1}{\alpha(\alpha-1)} (u^\alpha - 1 - \alpha(u - 1))$</td>
</tr>
</tbody>
</table>
Difficulties and Recent Progress

- GANs have been known to be difficult to train
  - "mode collapse": degenerated generators
- 2018: largely a solved issue
  - Stabilized GANs [Roth et al., NIPS 2017]
  - Consensus Optim [Mescheder et al., NIPS 2017]
ImageNet 1000 classes, 128x128, unconditional generation, ResNet 55 [Mescheder et al., ICML 2018], arXiv:1801.04406
ImageNet conditional generation [Mescheder et al., ICML 2018], arXiv:1801.04406
ImageNet conditional generation [Mescheder et al., ICML 2018], arXiv:1801.04406
Playing around with GANs

- [Roth et al., NIPS 2017](https://arxiv.org/abs/1705.09367)
- NIPS 2017 state of the art GAN models
  - Allows 150 layer ResNet GANs
  - TensorFlow implementation
- New state of the art: [Mescheder et al., ICML 2018], code [https://github.com/LMescheder/GAN_stability](https://github.com/LMescheder/GAN_stability)
Thanks!

Sebastian.Nowozin@microsoft.com
Additional Material
Maximum Likelihood

• Fisher, 1928: Estimate model parameters by maximizing the likelihood function
  \[ p(X; \theta) = \prod_i p(x_i; \theta) \]
  
• Equivalently, maximize the log-likelihood (more convenient)
  \[ \log p(X; \theta) = \sum_i \log p(x_i; \theta) \]
Representation Learning

\[ z \]

\[ z \]

\[ z \]

\[ z \]

\[ z \]
VAEs for Representation Learning

Diane Bouchacourt, Ryota Tomioka, Sebastian Nowozin
arXiv:1705.08841, NIPS 2017
Two parts latent code

- Style $s$
- Content $c$
Control over the latent space
Reparametrization Trick

- [Rezende et al., 2014]
- [Kingma and Welling, 2014]
Reparameterization Trick

- [Rezende et al., 2014]
  [Kingma and Welling, 2014]
- Stochastic computation graphs
  [Schulman et al., 2015]