Generative models: An introduction

Miloš Jordanski
Generative models

- Training data $D = \{x_1, \ldots, x_N\} \sim p_{data}(x)$

- Result: distribution $p_{model}(x) \approx p_{data}(x)$
Generative models

• $p_{model}(x)$ directly

• Generate samples from $p_{model}(x)$
Applications

• Additional training data
• Missing values
• Semi-supervised learning
• Reinforcement learning
• Multiple correct answers
• Text-to-Image Synthesis
• Learn useful embeddings
• ...
Maximum likelihood

• Maximum likelihood:

\[ \theta^* = \arg \max_{\theta} \prod_{i=1}^{m} p_{\text{model}}(x^{(i)}; \theta) \]

\[ = \arg \max_{\theta} \log \prod_{i=1}^{m} p_{\text{model}}(x^{(i)}; \theta) \]

\[ = \arg \max_{\theta} \sum_{i=1}^{m} \log p_{\text{model}}(x^{(i)}; \theta). \]

• Minimize Kullback-Leibler divergence:

\[ \theta^* = \arg \min_{\theta} D_{KL}(p_{\text{data}}(x)\|p_{\text{model}}(x; \theta)) \]
Explicit density

- Probability density: $p_{model}(x; \theta)$

- How to design $p_{model}(x; \theta)$?
  - Tractable density
  - Approximate density
Tractable density

• Fully visible belief networks:

\[ p_{\text{model}}(x) = \prod_{i=1}^{n} p_{\text{model}}(x_i|x_1, \ldots, x_{i-1}) \]

• WaveNet, NADE, RNN, PixelRNN, PixelCNN

• Disadvantages:
  • O(n) sample generation
  • Non parallelizable sample generation
  • Generation not controlled by a latent code
Tractable density

• Nonlinear independent component analysis
• Continuous, differentiable, invertible transformation $g$ between $X$ i $Z$, $x = g(z)$:

$$p_x(x) = p_z(g^{-1}(x)) \left| \det\left(\frac{\partial g^{-1}(x)}{\partial x}\right) \right|$$

• Disadvantages:
  • Choosing transformation $g$
  • $\dim(X) = \dim(Z)$
Approximate density

• Deterministic approximation (variational methods)

• Lower bound:

\[ \mathcal{L}(x; \theta) \leq \log p_{model}(x; \theta) \]

• Computationally tractable for carefully designed lower bound

• Disadvantages:

  • gap between lower bound and true likelihood
Approximate density

• Stochastic approximation (Deep Boltzmann Machines)

• Markov chain Monte Carlo:

\[ x' \sim q(x'|x) \]

• Disadvantages:

  • Slow convergence
  • Curse of dimensionality
  • Slow sample generation
Implicit density

• Sample generation from $p_{model}(x; \theta)$
  • Markov chain
    • Generative Stochastic Network
    • Disadvantages: slow sample generation for high dimensional data
  • Sample generation in a single step
    • GAN
Models with latent variables

- $x$ – observed variables
- $z$ – latent variables

\[ p_{\text{model}}(x) = \int p_{\text{model}}(x, z) \, dz \]
Models with latent variables

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

\[ p(z) \text{ - simple distribution} \]

\[ p(x|z; \theta) = g(z) \text{ – neural network} \]
Expectation Maximization

\[ lnp(x; \theta) = \mathcal{L}(q; \theta) + KL(q||p) \]

\[ \mathcal{L}(q; \theta) = \int_z q(z) \ln \left( \frac{p(x, z|\theta)}{q(z)} \right) \]

\[ KL(q||p) = -\int_z q(z) \ln \left( \frac{p(z|x, \theta)}{q(z)} \right) \]

\[ KL(q||p) \geq 0, KL(q||p) = 0 \iff q(z) = p(z|x, \theta) \]

\[ \mathcal{L}(q; \theta) = lnp(x|\theta) - KL(q||p) \leq lnp(x|\theta) \text{ lower bound} \]
Expectation Maximization

• E step: for fixed parameters $\theta^{old}$, maximize $\mathcal{L}(q; \theta)$ with respect to $q$: $q(z) = p(z|x; \theta^{old})$
• M step: for fixed $q(z)$ maximize $\mathcal{L}(q; \theta)$ with respect to $\theta$
Variational Inference

Problems:

• E step: evaluation of $p(z|x; \theta^{old})$

• M step: compute expectation with respect to $p(z|x; \theta^{old})$

Solutions:

• Stochastic approximation: Markov chain Monte Carlo

• Deterministic approximation: parametrize $q(z; \omega)$ i minimize $KL(q||p)$
Variational Autoencoders

- \( p(x) = \int_z p(x, z; \theta) = \int_z p(x|z; \theta)p(z) \)

- How to define the latent variables \( z \)?
  - \( z_i \sim N(0, 1) \) independent

- How to calculate integral?
  - Sample \( \{z_1, z_2, ..., z_N\} \) and calculate \( p(x) \approx \frac{1}{N} \sum_{i=1}^{N} p(x|z_i) \)
  - \( q(z|x; \omega) \sim p(z|x; \theta) \)
Variational Autoencoder

\[ \ln p(x; \theta) - KL(q(z|x; \omega)||p(z|x; \theta)) = \int_z q(z|x; \omega) \ln \left( \frac{p(x,z; \theta)}{q(z|x; \omega)} \right) \]

\[ \ln p(x; \theta) - KL(q(z|x; \omega)||p(z|x; \theta)) = \int_z q(z|x; \omega) \ln \left( \frac{p(x|z; \theta)p(z)}{q(z|x; \omega)} \right) \]

\[ \ln p(x; \theta) - KL(q(z|x; \omega)||p(z|x; \theta)) = \int_z q(z|x; \omega) \ln p(x|z; \theta) + \int_z q(z|x; \omega) \ln \left( \frac{p(z)}{q(z|x; \omega)} \right) \]

\[ \ln p(x; \theta) - KL(q(z|x; \omega)||p(z|x; \theta)) = E_{z \sim q(z|x; \omega)}[\ln p(x|z; \theta)] - KL(q(z|x; \omega)||p(z)) \]

\[ \mathcal{L}(x; \omega, \theta) = E_{z \sim q(z|x; \omega)}[\ln p(x|z; \theta)] - KL(q(z|x; \omega)||p(z)) \]
Stochastic gradient descent

- $q(z|x; \omega) = N(\mu(x; \omega), \Sigma(x; \omega))$, $\mu(x; \omega), \Sigma(x; \omega)$ neural network
- $KL(N(\mu(x), \Sigma(x)) || N(0, 1)) = \frac{1}{2} (\text{Tr}(\Sigma(x)) + \mu^T(x)\mu(x) - k - \log\det(\Sigma(x)))$
- $E_{z \sim q(z|x; \omega)}[\ln p(x|z; \theta)] \approx \ln p(x|z_i; \theta), z_i \sim q(z|x; \omega)$
- Maximize using stochastic gradient descent:

$$E_{x \sim D}[E_{z \sim q(z|x; \omega)}[\ln p(x|z; \theta)] - KL(q(z|x; \omega)||p(z))]$$
Reparameterization trick

- $q(z|x; \omega) = N(\mu_z(x), \Sigma_z(x))$
- $z = \mu_z(x) + \Sigma_z(x)^{1/2} \epsilon_z, \epsilon_z \sim N(0, I)$
- $E_{x \sim D}[E_{z \sim N(0,1)} \left[ ln p \left( x \mid z = \mu(x; \omega) + \Sigma_z^{1/2}(x; \omega) * \epsilon \right); \theta \right)] - KL(q(z|x; \omega)\mid\mid p(z))$
Variational autoencoders - results
Variational autoencoders - results
Variational autoencoders - results
Generative Adversarial Networks

• Advantages:
  • Sample generation in a single step (does not depend on dimensionality of X)
  • Does not use Markov chain
  • Does not use lower bound
  • Generation function has very few restrictions

• Disadvantages:
  • Hard to train
Generative Adversarial Networks

![Cumulative number of GAN papers by year](chart.png)
GAN - architecture

$D(x)$ tries to be near 1

Differentiable function $D$

$x$ sampled from data

$D$ tries to make $D(G(z))$ near 0, $G$ tries to make $D(G(z))$ near 1

Differentiable function $G$

Input noise $z$
GAN - discriminator

\[ J^{(D)}(\theta^{(D)}, \theta^{(G)}) = -\frac{1}{2}E_{x \sim p_{data}} \log D(x) - \frac{1}{2}E_{z \sim p_{gauss}} \log (1 - D(G(z))) \]

- Optimal discriminator:

\[ D^*(x) = \frac{p_{data}(x)}{p_{model}(x) + p_{data}(x)} \]
Zero-sum game

- N players
- Payoff matrix

\[\begin{array}{ccc}
\text{Red} & \text{A} & \text{B} & \text{C} \\
\hline
1 & 30 & -10 & 20 \\
2 & -10 & 20 & -20 \\
\end{array}\]

- Nash equilibrium of a game
GAN - generator

\[ J^{(G)}(\theta^{(D)}, \theta^{(G)}) = -J^{(D)}(\theta^{(D)}, \theta^{(G)}) \]

- Minimax game:
  \[ V(\theta^{(D)}, \theta^{(G)}) = -J^{(D)}(\theta^{(D)}, \theta^{(G)}) \]

- Solution:
  \[ \theta^{(D)*} = \underset{\theta^{(G)} \theta^{(D)}}{\text{argminmax}} V(\theta^{(D)}, \theta^{(G)}) \]

In practice:
\[ J^{(G)}(\theta^{(D)}, \theta^{(G)}) = -\frac{1}{2} E_z \log(D(G(z))) \]
GAN - Tricks

• $x \in [-1, 1]$

• Using labels

• One-side label smoothing

• Batch normalization

• Running more steps of one player
GAN - rezultati

Training Data

Samples
Plug and Play Generative Networks
Summary

• Explicit density
  • Tractable density
  • Approximate density
    • Deterministic approximations
    • Stochastic approximations

• Implicit density
  • Markov chain
  • GAN
THANKS!