Deep Generative Models

Roman Isachenko

Moscow Institute of Physics and Technology

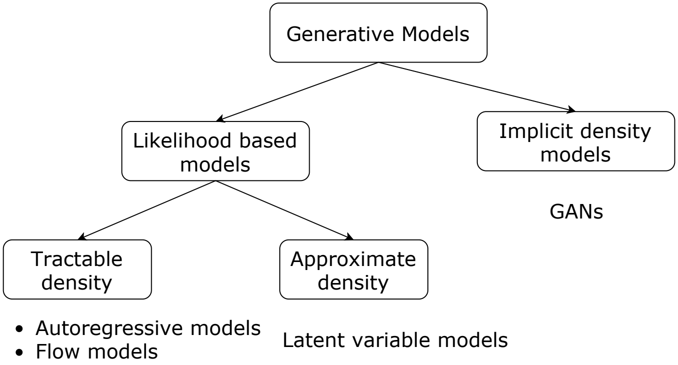
2020

Logistics

- homeworks: 30 points
  - hw1: autoregressive models
  - hw2: latent variable models
  - hw3: flow models
  - hw4: adversarial models
- exam: 30 points
- final project: 40 points

Last year course page: link
Admission: link

Generative models zoo



Motivation

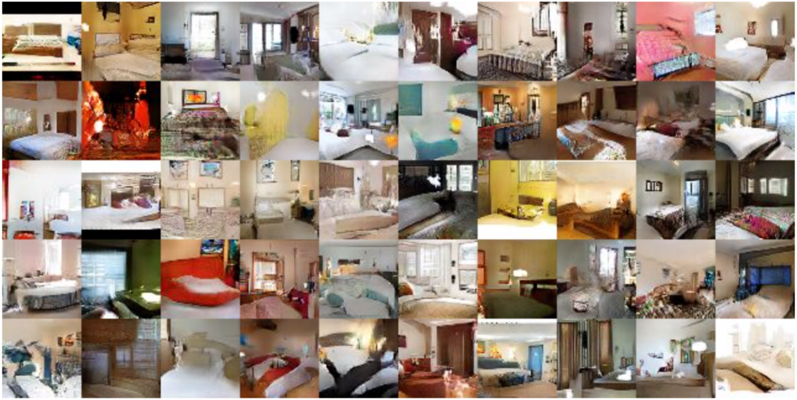
- **"Pure" Reinforcement Learning** (cherry)
  - The machine predicts a scalar reward given once in a while.
  - **A few bits for some samples**
- **Supervised Learning** (icing)
  - The machine predicts a category or a few numbers for each input
  - Predicting human-supplied data
  - **10→10,000 bits per sample**
- **Unsupervised/Predictive Learning** (cake)
  - The machine predicts any part of its input for any observed part.
  - Predicts future frames in videos
  - **Millions of bits per sample**

LeCun, NIPS 2016 Keynote

Applications: Image generation (VAE)

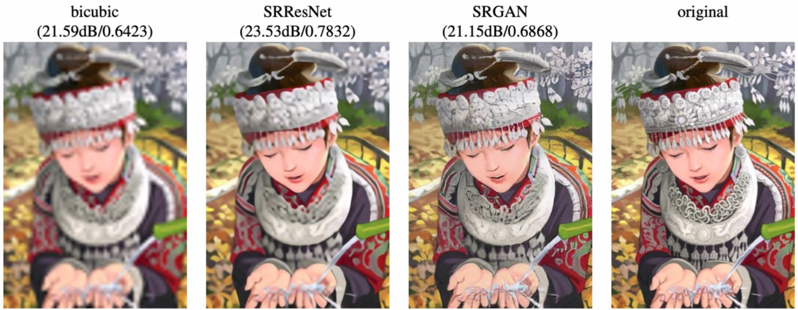
Kingma D. P., Welling M. Auto-encoding variational bayes
https://arxiv.org/pdf/1312.6114.pdf

Applications: Image generation (DCGAN)



Radford A., Metz L., Chintala S. Unsupervised representation learning with deep convolutional generative adversarial networks https://arxiv.org/abs/1511.06434

Applications: SuperResolution (SRGAN)



Ledig C. et al. Photo-realistic single image super-resolution using a generative adversarial network https://arxiv.org/abs/1609.04802

Applications: Domain translation (CycleGAN)

Zhu J. Y. et al. Unpaired image-to-image translation using cycle-consistent adversarial networks https://arxiv.org/abs/1703.10593

Applications: Face generation (StyleGAN)

Karras T., Laine S., Aila T. A style-based generator architecture for generative adversarial networks https://arxiv.org/abs/1812.04948

Applications: Face generation (VQ-VAE-2)

Razavi A., Oord A., Vinyals O. Generating Diverse High-Fidelity Images with VQ-VAE-2 https://arxiv.org/abs/1906.00446

Applications

- Audio Generation (WaveNet, ...)
- Video Generation (DVD-GAN)
- NLP (Transformer, BERT, GPT-3, ...)
- Compression

Problem Statement

Given samples $\{\mathbf{x}_i\}_{i=1}^n \in X$ from unknown distribution $p(\mathbf{x})$.

Goal

learn a distribution $p(\mathbf{x})$ for

- evaluating $p(\mathbf{x})$ for new samples;
- sampling from $p(\mathbf{x})$.

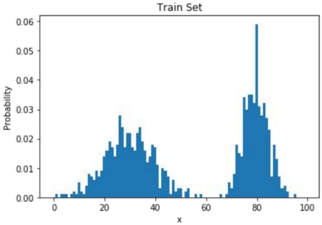
Challenge

Data is complex and high-dimensional (curse of dimensionality).

Histogram as a generative model

The histogram is totally defined by

$$p_k = p(x = k) = \frac{\sum_{i=1}^{k} [x_i = k]}{n}.$$



Problem: curse of dimensionality.

MNIST: 28x28 gray-scaled images
$2^{28\times 28} - 1$ parameters to specify $p(\mathbf{x})$

Question: How many parameters do we need in the case of independent features?

$$p(\mathbf{x}) = p(x_1) \cdot \dots \cdot p(x_m).$$

---

https://drive.google.com/file/d/1sHTVdppBqStzL1G1AHdWQrzHiqNFkzGH/view

Maximum likelihood

Fix probabilistic model $p(\mathbf{x}|\boldsymbol{\theta})$ – the set of parameterized distributions .
Instead of searching true $p(\mathbf{x})$ over all probability distributions, learn function approximation $p(\mathbf{x}|\boldsymbol{\theta}) \approx p(\mathbf{x})$.

MLE problem

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} p(\mathbf{X}|\boldsymbol{\theta}) = \arg\max_{\boldsymbol{\theta}} \prod_{i=1}^{n} p(\mathbf{x}_i|\boldsymbol{\theta}) = \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{n} \log p(\mathbf{x}_i|\boldsymbol{\theta}).$$

The problem is solved with SGD.

Requirements

- efficiently compute $\log p(\mathbf{x}|\boldsymbol{\theta})$;
- efficiently compute gradient of $\log p(\mathbf{x}|\boldsymbol{\theta})$.

Autoregressive model

MLE problem

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} p(\mathbf{X}|\boldsymbol{\theta}) = \arg\max_{\boldsymbol{\theta}} \prod_{i=1}^{n} p(\mathbf{x}_i|\boldsymbol{\theta}) = \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{n} \log p(\mathbf{x}_i|\boldsymbol{\theta}).$$

Challenge

$p(\mathbf{x}|\boldsymbol{\theta})$ could be intractable.

Likelihood as product of conditionals

Let $\mathbf{x} = (x_1, \dots, x_m)$, $\mathbf{x}_{1:i} = (x_1, \dots, x_i)$. Then

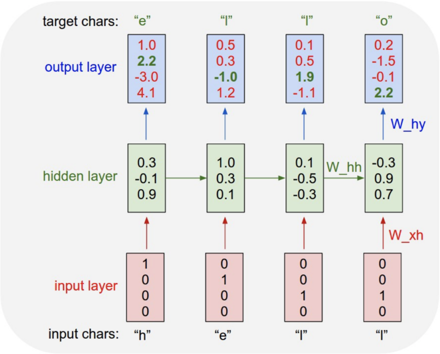
$$p(\mathbf{x}|\boldsymbol{\theta}) = \prod_{i=1}^{m} p(x_i|\mathbf{x}_{1:i-1}, \boldsymbol{\theta}); \quad \log p(\mathbf{x}|\boldsymbol{\theta}) = \sum_{i=1}^{m} \log p(x_i|\mathbf{x}_{1:i-1}, \boldsymbol{\theta}).$$

Autoregressive models

$$\log p(\mathbf{x}|\boldsymbol{\theta}) = \sum_{i=1}^{m} \log p(x_i|\mathbf{x}_{1:i-1}, \boldsymbol{\theta})$$

- Each conditional could be modelled by neural network.
- To extend to high dimensions share parameters across conditionals.
- Sampling is sequential.

Char RNN (2015)




```
PANDARUS:
Alas, I think he shall be come approached and the day
When little srain would be attain'd into being never fed,
And who is but a chain and subjects of his death,
I should not sleep.

Second Senator:
They are away this miseries, produced upon my soul,
Breaking and strongly should be buried, when I perish
The earth and thoughts of many states.

DUKE VINCENTIO:
Well, your wit is in the care of side and that.

Second Lord:
They would be ruled after this chamber, and
my fair nues begun out of the fact, to be conveyed,
Whose noble souls I'll have the heart of the wars.

Clown:
Come, sir, I will make did behold your worship.

VIOLA:
I'll drink it.
```

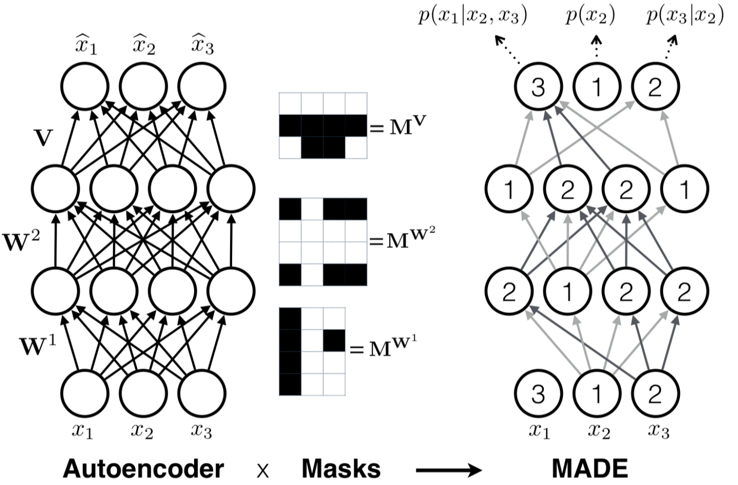
Drawback

Sequential computation of all conditionals $p(x_i|\mathbf{x}_{1:i-1}, \boldsymbol{\theta})$.

---

http://karpathy.github.io/2015/05/21/rnn-effectiveness/

MADE (2015)



Germain M. et al. Made: Masked autoencoder for distribution estimation

https://arxiv.org/pdf/1502.03509.pdf

WaveNet (2016)

Goal

Efficient generation of raw audio waveforms with natural sounds.

Solution

Autoregressive model

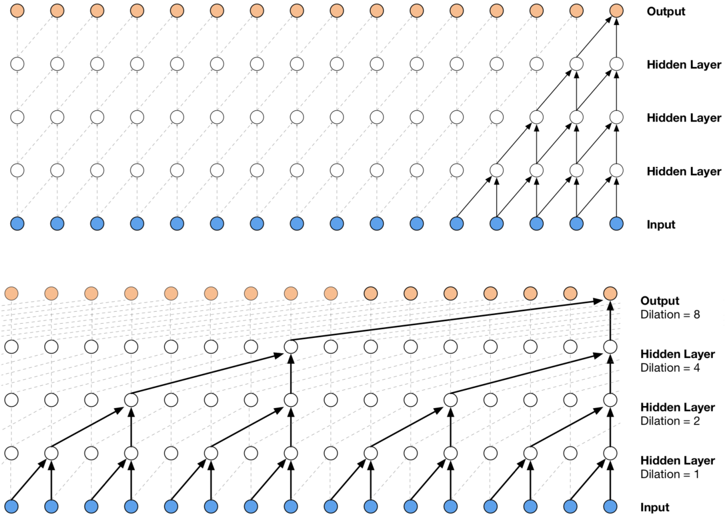
$$p(\mathbf{x}|\boldsymbol{\theta}) = \prod_{t=1}^{T} p(x_t|\mathbf{x}_{1:t-1}, \boldsymbol{\theta}).$$

The model uses causal dilated convolutions.

Oord A. et al. Wavenet: A generative model for raw audio

https://arxiv.org/pdf/1609.03499.pdf

WaveNet (2016)



Oord A. et al. Wavenet: A generative model for raw audio

https://arxiv.org/pdf/1609.03499.pdf

PixelCNN (2016)

Goal

Modeling the distribution of natural images.

Solution

Autoregressive model

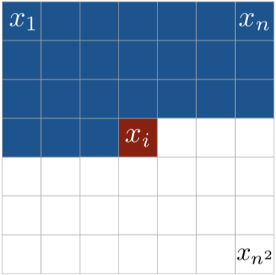
$$p(\mathbf{x}|\boldsymbol{\theta}) = \prod_{i=1}^{n^2} p(x_i|\mathbf{x}_{1:i-1}, \boldsymbol{\theta}).$$

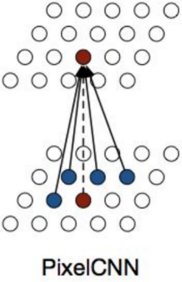
- masked convolutions;
- dependencies over RGB channels.

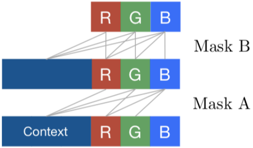
Oord A., Kalchbrenner N., Kavukcuoglu K. Pixel recurrent neural networks
https://arxiv.org/pdf/1601.06759.pdf

PixelCNN (2016)

1	1	1
1	0	0
0	0	0



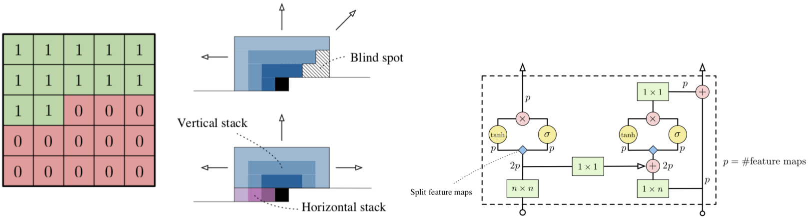




Oord A., Kalchbrenner N., Kavukcuoglu K. Pixel recurrent neural networks

https://arxiv.org/pdf/1601.06759.pdf

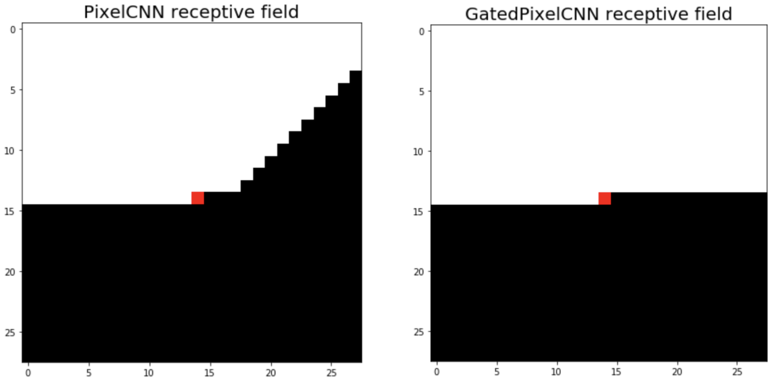
GatedPixelCNN (2016)



Van den Oord A. et al. Conditional image generation with pixelcnn decoders

https://arxiv.org/pdf/1606.05328.pdf

GatedPixelCNN (2016)



Van den Oord A. et al. Conditional image generation with pixelcnn decoders

https://arxiv.org/pdf/1606.05328.pdf

Extensions

- **PixelCNN++**: *Improving the PixelCNN with Discretized Logistic Mixture Likelihood and Other Modifications* https://arxiv.org/pdf/1712.09763.pdf (mixture of logistics instead of softmax);
- **PixelSNAIL**: *An Improved Autoregressive Generative Model* https://arxiv.org/pdf/1712.09763.pdf (self-attention to learn optimal autoregression ordering).

Summary

- Sampling from autoregressive models are trivial, but sequential
  - sample $x_0 \sim p(x_0)$;
  - sample $x_1 \sim p(x_1|x_0)$;
  - ....
- Estimating probability:

$$p(\mathbf{x}) = \prod_{i=1}^{m} p(x_i|\mathbf{x}_{1:i-1}).$$

- Work on both continuous and discrete data.
- There is no natural way to do unsupervised learning.

References

- **MADE**: *Masked Autoencoder for Distribution Estimation*
  https://arxiv.org/pdf/1502.03509.pdf
  Summary: Create masked autoencoder that models autoregression (autoregression allows to make the distribution properly normalized). Sampling is performed iteratively (to generate MNIST image 784 forward passes are needed). Discrete data.
- **PixelRNN + PixelCNN**: *Pixel recurrent neural networks*
  https://arxiv.org/abs/1601.06759
  Summary: 2 models are proposed: PixelRNN, PixelCNN. The models are autoregression and sampling is sequential. For RNN two types of LSTM blocks are used: Row LSTM and DiagonalBiLSTM. CNN uses Masked convolutions. RNN outperforms, but is slower.
- **GatedPixelCNN**: *Conditional Image Generation with PixelCNN Decoders*
  https://arxiv.org/pdf/1606.05328.pdf
  Summary: Improvements for PixelCNN: gated units (like in lstm), horizontal+vertical stacks (remove blind spots). The result is now similar to PixelRNN.
- **WaveNet**: *a Generative Model for Raw Audio*
  https://arxiv.org/pdf/1609.03499.pdf
  Summary: Model for autoregressive audio generation, inspired by PixelCNN. Use causal convolutions for the right conditioning, and dilated atrous convolution to extend receptive field.
- **PixelCNN++**: *Improving the PixelCNN with Discretized Logistic Mixture Likelihood and Other Modifications*
  https://arxiv.org/pdf/1701.05517.pdf
  Summary: Improved version of PixelCNN. Models mixture of logistic mixture distribution instead of softmax. Architectural modifications: skip connections, up/down sampling, dropout. Experiment with dequantization: discretization works better.
- **PixelSNAIL**: *An Improved Autoregressive Generative Model*
  https://arxiv.org/pdf/1712.09763.pdf
  Summary: Autoregressive model. Uses masked causal convolutions. Adjust self-attention to PixelCNN.