M2177.003100
Deep Learning

[13: Introduction to Generative Models]

Electrical and Computer Engineering
Seoul National University

(last compiled at 13:35:00 on 2020/11/15)

Outline

References

- *Deep Learning* by Goodfellow, Bengio and Courville Link
  - Chapter 19: Approximate Inference
  - Chapter 20: Deep Generative Models
- *Pattern Recognition and Machine Learning* by Bishop
  - Chapter 10: Approximate Inference
- online resources:
  - *Stanford CS231n: CNN for Visual Recognition* Link
  - *CVPR 2018 GAN Tutorial* Link
  - *NIPS 2016 Variational Inference Tutorial* Link
  - *NIPS 2016 GAN Tutorial* Link

Outline

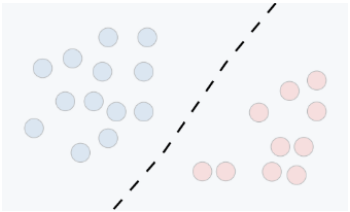
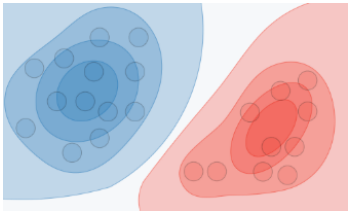
Supervised vs unsupervised learning

	supervised	**unsupervised**
data	(x, y) x: data, y: label	x just data, no labels
goal	learn a *function* to map $x \mapsto y$	learn inherent *structure* of the data
examples	classification regression object detection semantic segmentation	clustering dimensionality reduction representation learning density estimation

source: Fei-Fei Li, J. Johnson, S. Yeung, CS231n: Convolutional Neural Networks for Visual Recognition (2017), http://cs231n.stanford.edu/2017/index.html

Discriminative vs generative models

- assume supervised learning
  - goal: learn a *function* f to map $\boldsymbol{x} \mapsto y$

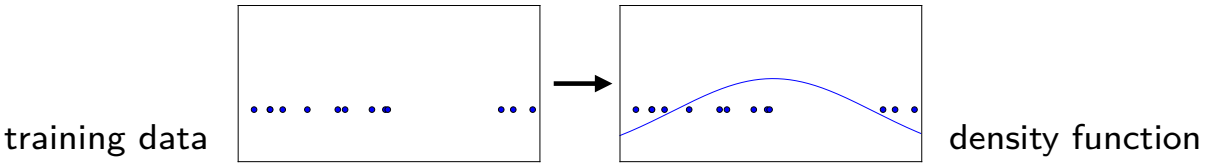
	discriminative	**generative**
goal	directly estimate $p(y \mid \boldsymbol{x})$	estimate $p(\boldsymbol{x} \mid y)$; then deduce[1] $p(y \mid \boldsymbol{x})$
should evaluate	$f(\boldsymbol{x}) = \operatorname{argmax}_y p(y \mid \boldsymbol{x})$	$f(\boldsymbol{x}) = \operatorname{argmax}_y p(\boldsymbol{x} \mid y)p(y)$
what's learned	decision boundary	probability _____ of data
		
examples	support vector machine	Gaussian mixture, Bayes nets

sources: Ng and Jordan (2002). On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In *Advances in Neural Information Processing Systems*, pages 841–848;
https://stanford.edu/~shervine/teaching/cs-229/cheatsheet-supervised-learning

[1] *e.g.* using Bayes rule: $p(y \mid \boldsymbol{x}) = \frac{p(\boldsymbol{x} \mid y)p(y)}{p(\boldsymbol{x})}$

Generative modeling in unsupervised learning

- density estimation
  - goal: learn $p(\boldsymbol{x})$



- sample generation
  - given training data, generate new samples from the same distribution

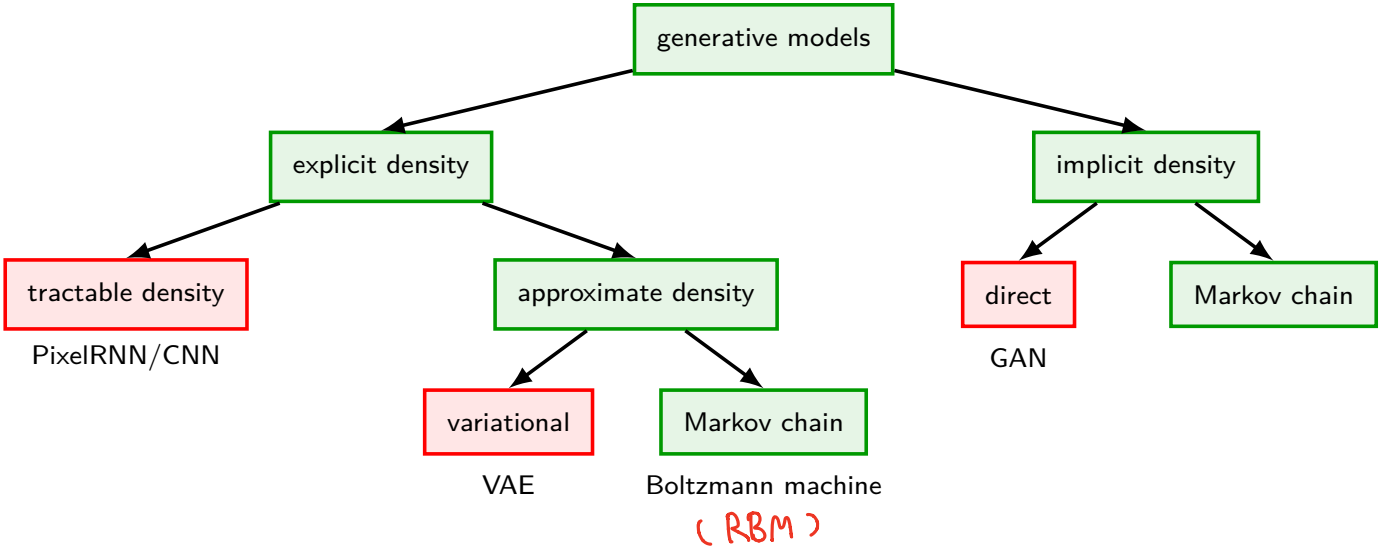
training data $\sim p_{\text{data}}(\boldsymbol{x})$

generated samples $\sim p_{\text{model}}(\boldsymbol{x})$

  - want to learn $p_{\text{model}}(\boldsymbol{x})$ similar to $p_{\text{data}}(\boldsymbol{x})$

image sources: Goodfellow (2016) https://media.nips.cc/Conferences/2016/Slides/6202-Slides.pdf; Karras et al. "Progressive growing of GANs for improved quality, stability, and variation." *arXiv preprint arXiv:1710.10196* (2017)

Taxonomy of generative models



1. explicit density estimation
   - explicitly define and solve for $p_{\text{model}}(\boldsymbol{x})$
2. implicit density estimation
   - learn a model that can sample from $p_{\text{model}}(\boldsymbol{x})$ w/o explicitly defining it

image modified from: Goodfellow (2016) https://media.nips.cc/Conferences/2016/Slides/6202-Slides.pdf

Trends in generative models

- conventional GM approaches have one of three drawbacks[2]:
  1. require strong assumptions about structure in data
  2. make approximations $\Rightarrow$ suboptimal results
  3. rely on computationally expensive inference procedures (*e.g.* MCMC)

- recent advances
  - train neural nets as powerful function approximators through backprop

  $\Rightarrow$ gives framework for backprop-based function approximators to build GMs

e.g. variational autoencoder: one of the most popular such frameworks
  - assumptions of this model: weak
  - training: fast via backprop
  - make an approximation but error is small given high-capacity models

[2] Doersch, C. (2016). Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908*

- generative models based on neural nets:

1. autoregressive models (or fully visible belief nets)
   - model $p(\boldsymbol{x})$ as $\prod_{i=1}^{n} p(x_i \mid x_1, \ldots, x_{i-1})$

   e.g. PixelRNN, PixelCNN, WaveNet

2. Helmholtz machines
   - model $p(\boldsymbol{x})$ as $\int p(\boldsymbol{z})p(\boldsymbol{x} \mid \boldsymbol{z})d\boldsymbol{z}$
   - two components: recognition net (encoder) + generative net (decoder)
   - use variational inference to maximize ELBO

   e.g. variational autoencoder (VAE)

3. generative adversarial network (GAN)
   - no explicit density modeling
   - two components: generator + discriminator
   - train models by solving minimax problem

Why study generative models? (Goodfellow, 2016)

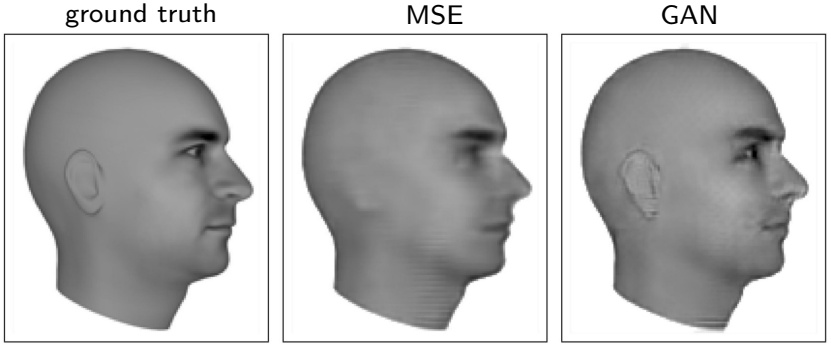
- excellent test of our ability
  - to represent/manipulate high-dim/complicated probability distributions
- can be incorporated into reinforcement learning (RL)
  - simulate possible futures for planning
- can be trained with missing data
  - can provide predictions on inputs that are missing data

  $\Rightarrow$ facilitate semi-supervised learning

image source: Chen et al. "High resolution face completion with multiple controllable attributes via fully end-to-end progressive generative adversarial networks." *arXiv preprint arXiv:1801.07632* (2018)

- enable machine learning to work with multi-modal outputs

  e.g. a single input $\rightarrow$ many different correction answers



$\leftarrow$ video frame prediction

- MSE: averaging

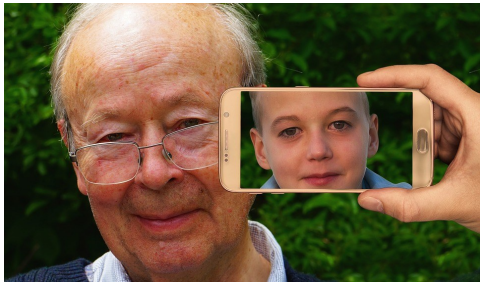
$\Rightarrow$ blurry

- GAN: sharp

- enable inference of latent representations

  $\Rightarrow$ can be useful as general features

  e.g. identity-preserving latent representations $\rightarrow$



age conditional GAN

image sources: Lotter et al. "Unsupervised learning of visual structure using predictive generative networks." *arXiv preprint arXiv:1511.06380* (2015); https://images.app.goo.gl/KymUzyWb9LSHdHtf9

- many tasks require realistic generation of samples

  e.g. single image super-resolution, art creation, image-to-image translation

first AI art sold at Christie ($432,000)

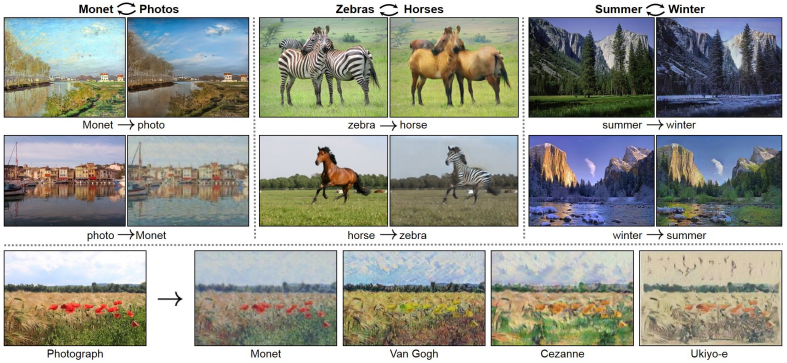
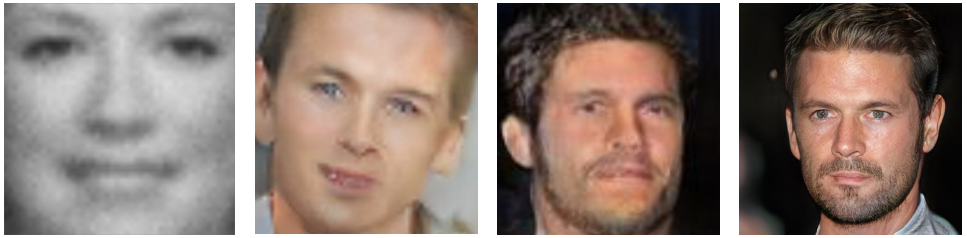


image-to-image translation

CycleGAN



2014 2015 2016 2017

3.5 years of progress on faces

PGGAN

image sources: "Edmond de Belamy" by Obvious (Christie's); Zhu et al. "Unpaired image-to-image translation using cycle-consistent adversarial networks." *Proceedings of the IEEE International Conference on Computer Vision* (2017); Brundage et al. (2017)

Outline

Fully visible belief network (FVBN)

- explicit density model
- how it works:
  1. decompose likelihood of each input $\boldsymbol{x}$ into a product of 1D distributions

$$\underbrace{p(\boldsymbol{x})}_{\substack{\text{likelihood} \\ \text{of input } \boldsymbol{x}}} = \underbrace{\prod_{i=1}^{n} p(x_i \mid x_1, \ldots, x_{i-1})}_{\substack{\text{probability of } i\text{-th feature} \\ \text{given all previous features}}}$$

  2. maximize likelihood of training data
- complex distribution over feature values
  - we express it using a neural net
- main issue:
  - need to define ordering of "previous features"

PixelRNN

- context: image (ICML 2016 best paper[3])
- idea: the same as language modeling applied to image
  - generate image pixels starting from corner
  - model dependency on previous pixels using 2D LSTM
- limitation
  - sequential generation $\Rightarrow$ slow

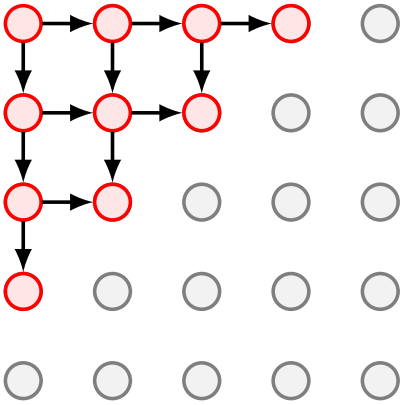


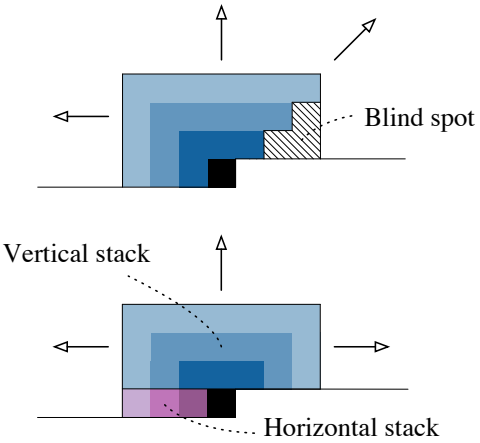
image source: van den Oord et al. (2016) Pixel recurrent neural networks. *arXiv preprint arXiv:1601.06759*

[3] this paper also proposed a simple PixelCNN, which was revised later in their NeurIPS paper (next page)

PixelCNN

- still generate image pixels starting from corner
- what's new
  - use CNN (not LSTM) to model dependency on previous pixels
  - two conv stacks to remove blind spots
- training
  - maximize likelihood of training images
  - loss: softmax loss at each pixel
  - faster[4] than PixelRNN
- image generation
  - must still proceed sequentially

  $\Rightarrow$ still slow



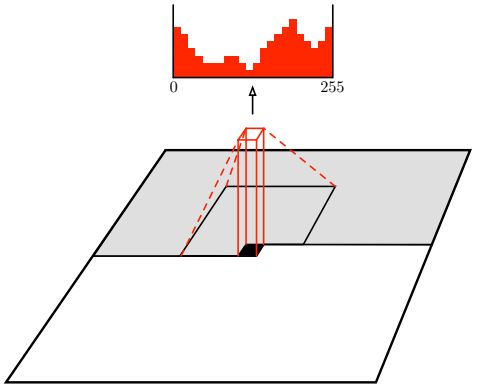
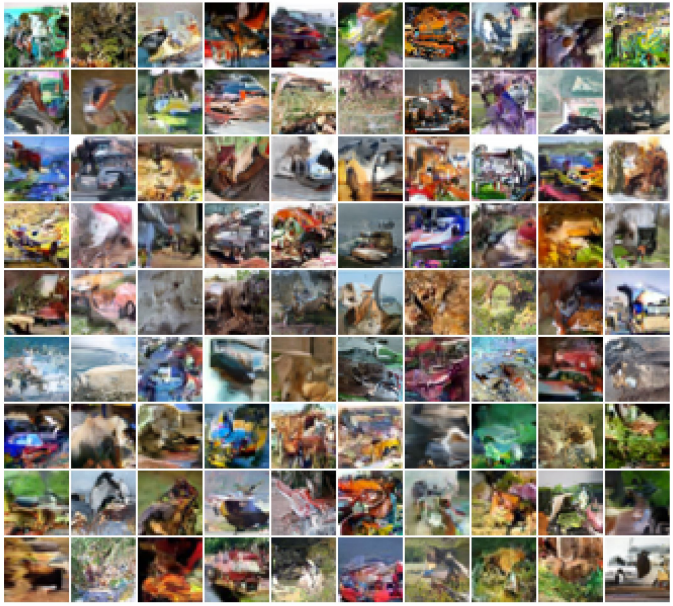


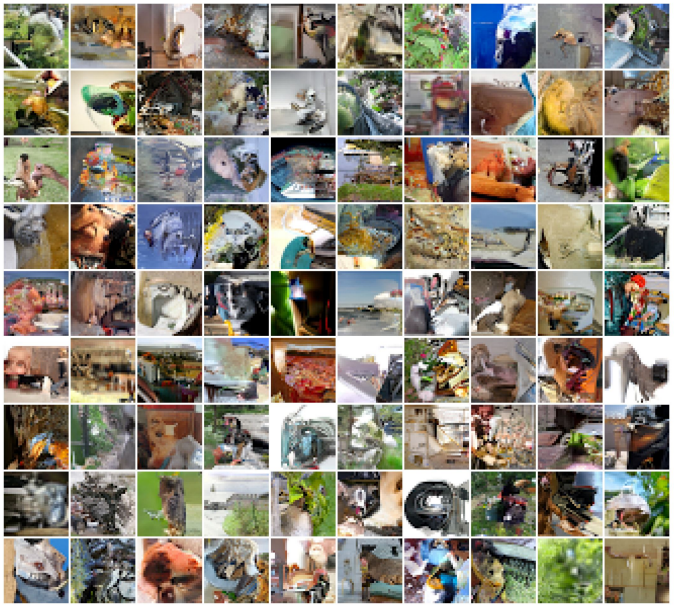
image source: van den Oord et al. (2016) Conditional image generation with PixelCNN decoders. In *Advances in Neural Information Processing Systems*, pages 4790–4798

[4] we can parallelize convolutions since context region values are known from training images

Examples of generated samples



32×32 CIFAR-10



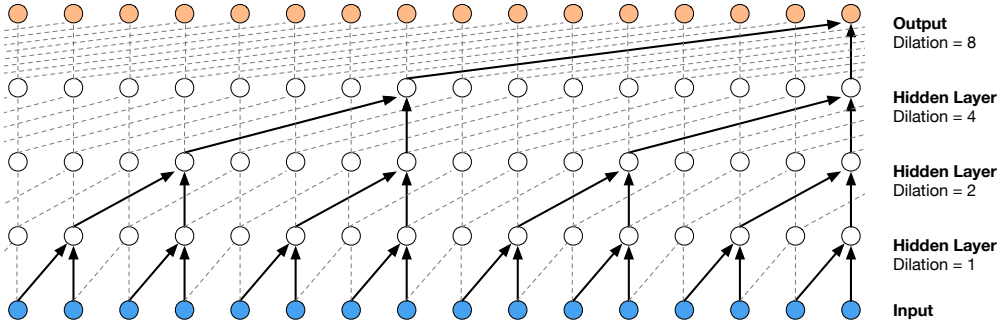
32×32 ImageNet

image source: van den Oord et al. (2016) Pixel recurrent neural networks. *arXiv preprint arXiv:1601.06759*

WaveNet

- can generate raw audios (trained on raw audio waveforms)
  - similar to PixelCNN in structure, but much more successful
  - amazing quality but slow generation (2 min to synthesize 1 sec audio)

- architecture
  - use dilated convolution to vastly increase receptive field



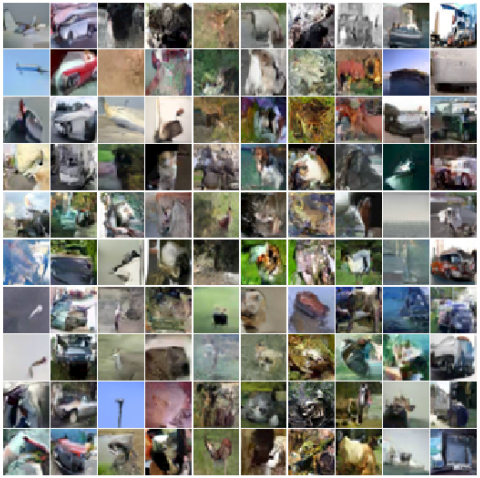
  - stack many layers like the above together
  - use classification (not regression) to generate next sample point
  - convert continuous audio $\rightarrow$ 256-level quantized classes

image source: van den Oord et al. (2016) WaveNet: A generative model for raw audio. In *SSW*, page 125

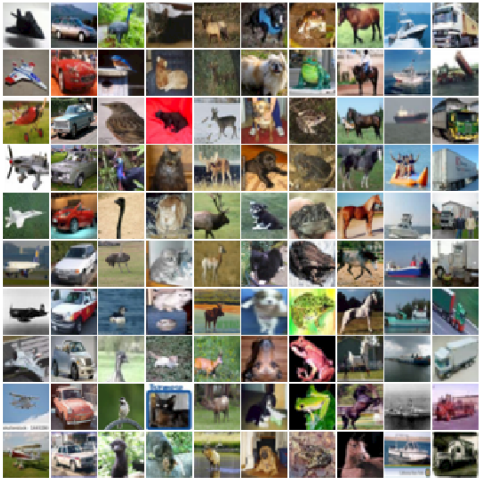
Remarks

- autoregressive models: pros and cons
  - can explicitly compute $p(\boldsymbol{x})$
  - good samples
  - sequential generation: slow
- to improve performance
  - gated convolution layers
  - short-cut connections
  - discretized logistic loss
  - multi-scale
  - parallelization
  - and many more!
- more information: ▸ Link

PixelCNN++ samples from CIFAR-10



real CIFAR-10 images



sources: Fei-Fei Li, J. Johnson, S. Yeung, CS231n: Convolutional Neural Networks for Visual Recognition (2017), `http://cs231n.stanford.edu/2017/index.html`; Salimans et al. (2017) PixelCNN++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications. *arXiv preprint arXiv:1701.05517*

Outline

Probabilistic machine learning

- a probabilistic model
  - a joint distribution of hidden variables $\mathbf{z}$ and observed variables $\mathbf{x}$

$$p(\boldsymbol{z}, \boldsymbol{x})$$

  - describes how (a portion of) the world works

- inference about the unknowns:
  - through **posterior** $p(z|x)$

  i.e. conditional distribution of the hidden variables given the observations

$$p(\boldsymbol{z} \mid \boldsymbol{x}) = \frac{p(\boldsymbol{z}, \boldsymbol{x})}{p(\boldsymbol{x})} \begin{array}{l} \leftarrow \text{model} \\ \leftarrow \text{data} \end{array}$$

  - the posterior links data and model $\Rightarrow$ used in all downstream analyses

- posterior inference:
  - therefore a central task in probabilistic models

Posterior inference

- refers to
  - (1) computing posterior distribution $p(\boldsymbol{z} \mid \boldsymbol{x})$ or
  - (2) taking expectations computed wrt this distribution

e.g. expectation-maximization (EM) algorithm

- evaluates expectation of complete-data log likelihood wrt posterior distribution of latent variables:

$$\mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{old}}) \triangleq \sum_{\mathbf{z}} p(\boldsymbol{z} \mid \boldsymbol{x}, \boldsymbol{\theta}^{\text{old}}) \log p(\boldsymbol{x}, \boldsymbol{z} \mid \boldsymbol{\theta})$$

- for many models of practical interest
  - it is intractable to do (1)/(2) $\Rightarrow$ called "challenge of inference"

- challenge of inference
  - makes it difficult to train probabilistic models

Challenge of inference

- general reasons
  - dimensionality of latent space: too high to work with directly
  - posterior: highly complex $\Rightarrow$ expectations are not analytically tractable
- reasons in deep learning
  - interactions between hidden variables (*e.g.* connections between layers)
  - most neural nets with multiple layers of hidden variables
    - have intractable posterior distributions
- a solution: approximate posterior inference

Bayesian view

- we setup the general problem
  - consider a joint density of latent variables $\mathbf{z}$ and observations $\mathbf{x}$

$$p(\boldsymbol{z}, \boldsymbol{x}) = p(\boldsymbol{z})p(\boldsymbol{x} \mid \boldsymbol{z})$$

  - latent variables help govern distribution of data in Bayesian models

- a Bayesian model
  - draws latent variables from a prior $p(\boldsymbol{z})$, and then
  - relates them to observations through likelihood $p(\boldsymbol{x} \mid \boldsymbol{z})$

- inference in a Bayesian model
  - conditions on data and computes posterior $p(\boldsymbol{z} \mid \boldsymbol{x})$

this often requires approximate inference

Approximate posterior inference

1. stochastic (*e.g.* Markov chain Monte Carlo: MCMC)
   - given infinite computational resource, can generate exact results
   - approximation arises from using a finite amount of processor time
   - ☹ sampling methods can be computationally demanding
   - ⇒ their use is often limited to small-scale problems
   - ☹ difficult to know whether independent samples are being generated

2. deterministic (*e.g.* variational inference: VI)
   - some of which scale well to large applications
   - based on analytical approximations to posterior $p(\boldsymbol{z} \mid \boldsymbol{x})$ $\simeq q$
   - *e.g.* assume it factorizes or has a parametric form (like Gaussian)
   - ⇒ never generate exact results

Markov chain Monte Carlo (MCMC)

- dominant paradigm for decades
  1. construct an ergodic[5] Markov chain on $\mathbf{z}$

     ↑ its stationary distribution = posterior $p(\boldsymbol{z} \mid \boldsymbol{x})$
  2. sample from the chain to collect samples from the stationary distribution
  3. approximate the posterior with an empirical estimate

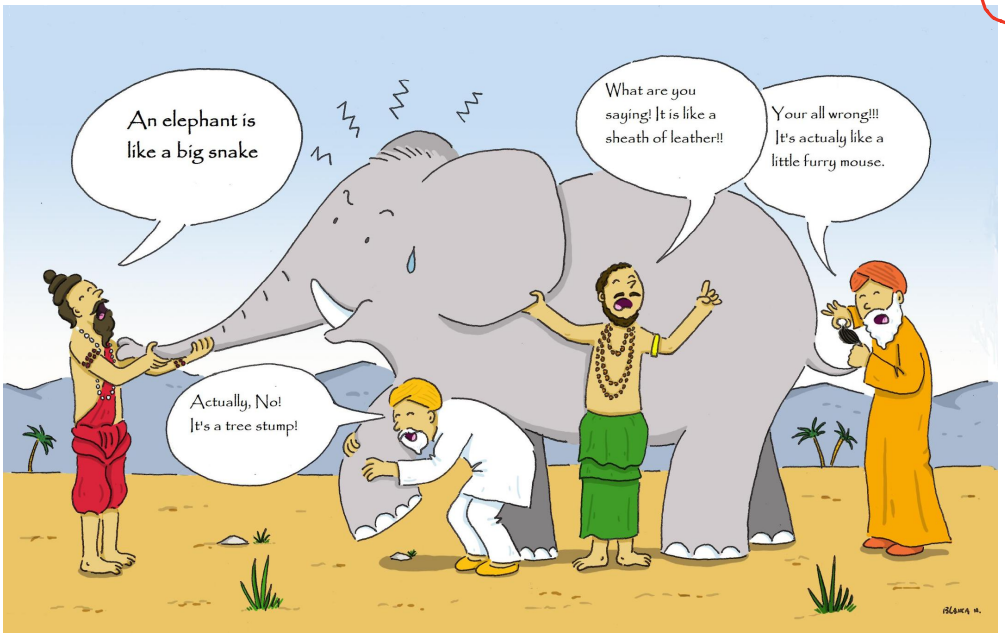
     ↑ constructed from (a subset of) the collected samples

- widely studied, extended, and applied

  e.g. Metropolis-Hastings algorithm, Gibbs sampling

- too slow for large data sets/complex models
  - a good alternative in these settings: variational inference

[5] an ergodic system is a dynamical system with the following property: no matter where you start, you run the system and keep running it, then any other points in the space you will eventually reach



- goal: sample from a small subset of high-dim region where sampling is hard
  - often p has some (connected) structure
  - start somewhere, and start moving around (red dots)
  - try to move toward a region of high probability, and then get there
  - try to stay in the regions of high probability (green regions), doing random walks
  - ⁂ MCMC: do this by forming a Markov chain

image source: Wild Equus. Elephant & blind sages by Blanca Marti for Equilibre

Variational inference (VI)

- main idea: turn inference into optimization (rather than sampling)

1. posit $\mathcal{Q}$, a *family* of approximate densities over $\mathbf{z}$
   - each $q(\boldsymbol{z}) \in \mathcal{Q}$: a candidate approximation to the exact posterior

$p(z|x) \simeq q(z) \in \mathcal{Q}$

2. find the member of $\mathcal{Q}$ that minimizes D_{KL} to the exact posterior

$$q^*(\boldsymbol{z}) = \underset{q(\boldsymbol{z}) \in \mathcal{Q}}{\operatorname{argmin}} D_{\mathrm{KL}}[q(\boldsymbol{z}) \,||\, p(\boldsymbol{z} \,|\, \boldsymbol{x})]$$

3. approximate the posterior with q^*
   - use $q^*(\boldsymbol{z})$ in place of $p(\boldsymbol{z} \,|\, \boldsymbol{x})$ in downstream modeling

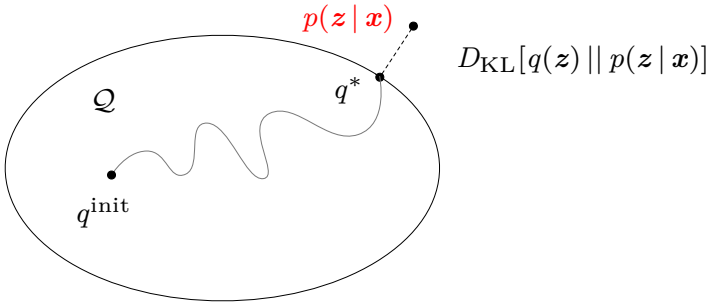


image source: Blei et al. (2017) Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877

Comparison

- different approaches to solving the same problem

	Markov chain Monte Carlo	variational inference
what it does	samples a Markov chain	solves an optimization problem
approximates posterior with	samples from the chain	optimization results
a tool for	simulating (sampling) from densities	approximating densities

when to use MCMC and when to use VI?

- MCMC
  - more computationally intensive than VI, but
  - guarantees producing (asymptotically) exact samples from target density

  $\Rightarrow$ suited to smaller data sets/scenarios
    - where we happily pay a heavier cost for more precise samples

- VI
  - has no such guarantees (only finds a density close to the target)
  - but faster than MCMC (also can use stochastic/distributed optimization)

  $\Rightarrow$ suited to large data sets/scenarios
    - where we want to quickly explore many models

Outline

The phrase "variational"[6]

- an umbrella term that refers to mathematical tools for
  - optimization-based formulations of problems
  - associated techniques for their solution
- origin: *calculus of variations* (optimization over a space of functions)
- general idea:
  - express a quantity of interest as the solution of an optimization problem
- the optimization problem can then be "relaxed"
  - either by approximating
    - the function to be optimized or
    - the set over which the optimization takes place
  - such relaxations allow us to
    - approximate the original quantity of interest

[6] Wainwright, M. J., Jordan, M. I., et al. (2008). Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1–2):1–305

Variational inference (variational Bayes)

- inference = solving the following optimization:

$$q^*(\boldsymbol{z}) = \underset{q(\boldsymbol{z})\in\mathcal{Q}}{\operatorname{argmin}} D_{\mathrm{KL}}[q(\boldsymbol{z}) \,||\, p(\boldsymbol{z}\,|\,\boldsymbol{x})] \tag{1}$$

  - complexity of $\mathcal{Q}$ determines complexity of this optimization

- choosing $\mathcal{Q}$ needs a good balance
  1. restrict $\mathcal{Q}$ sufficiently [for efficiency]
     $\Rightarrow$ it comprises only tractable distributions
  2. allow $\mathcal{Q}$ to be sufficiently rich/flexible [for accuracy]
     $\Rightarrow$ it can provide a good approximation to the true posterior

Optimization challenge

- the objective (1): not computable

$$q^*(\boldsymbol{z}) = \underset{q(\boldsymbol{z}) \in \mathcal{Q}}{\operatorname{argmin}} D_{\mathrm{KL}}[q(\boldsymbol{z}) \,||\, p(\boldsymbol{z} \,|\, \boldsymbol{x})] \tag{1}$$

- reason:

$$\begin{aligned} D_{\mathrm{KL}}[q(\boldsymbol{z}) \,||\, p(\boldsymbol{z} \,|\, \boldsymbol{x})] &= \mathbb{E}_{\mathbf{z} \sim q(\boldsymbol{z})}[\log q(\boldsymbol{z})] - \mathbb{E}_{\mathbf{z} \sim q(\boldsymbol{z})}[\log p(\boldsymbol{z} \,|\, \boldsymbol{x})] \\ &= \mathbb{E}_{\mathbf{z}}[\log q(\boldsymbol{z})] - \mathbb{E}_{\mathbf{z}}[\log p(\boldsymbol{z}, \boldsymbol{x})] + \underbrace{\log p(\boldsymbol{x})}_{\text{intractable}} \end{aligned} \tag{2}$$

- solution:
  - optimize an alternative objective called evidence lower bound (ELBO)

also known as negative variational free energy

Evidence lower bound (ELBO)

- rearranging (2) gives

$$\underbrace{\log p(\boldsymbol{x})}_{\substack{\text{a constant} \\ \text{wrt } q(\boldsymbol{z})}} - D_{\mathrm{KL}}[q(\boldsymbol{z}) \,||\, p(\boldsymbol{z} \,|\, \boldsymbol{x})] = \underbrace{\mathbb{E}_{\mathbf{z}}[\log p(\boldsymbol{z}, \boldsymbol{x})] - \mathbb{E}_{\mathbf{z}}[\log q(\boldsymbol{z})]}_{\substack{\triangleq \mathcal{L}(q) \quad \text{“ELBO”}}} \tag{3}$$

  - maximizing ELBO $=$ minimizing the D_{KL}

- examining ELBO gives intuitions about optimal variational density

$$\begin{aligned}
\mathcal{L}(q) &= \mathbb{E}_{\mathbf{z}}[\log p(\boldsymbol{z}, \boldsymbol{x})] - \mathbb{E}_{\mathbf{z}}[\log q(\boldsymbol{z})] \\
&= \mathbb{E}_{\mathbf{z}}[\log p(\boldsymbol{z})] + \mathbb{E}_{\mathbf{z}}[\log p(\boldsymbol{x} \,|\, \boldsymbol{z})] - \mathbb{E}_{\mathbf{z}}[\log q(\boldsymbol{z})] \\
&= \underbrace{\mathbb{E}_{\mathbf{z}}[\log p(\boldsymbol{x} \,|\, \boldsymbol{z})]}_{\substack{\text{expected log likelihood} \\ \text{of the data}}} - \underbrace{D_{\mathrm{KL}}[q(\boldsymbol{z}) \,||\, p(\boldsymbol{z})]}_{\substack{D_{\mathrm{KL}} \text{ between} \\ \text{prior } p(\boldsymbol{z}) \text{ and } q(\boldsymbol{z})}}
\end{aligned}$$

ELBO ("variational objective"): $\mathcal{L}(q) = \mathbb{E}_{\mathbf{z}\sim q(\boldsymbol{z})}[\log p(\boldsymbol{x} \mid \boldsymbol{z})] - D_{\mathrm{KL}}[q(\boldsymbol{z}) \,||\, p(\boldsymbol{z})]$

which values of $\boldsymbol{z}$ will this objective encourage $q(\boldsymbol{z})$ to place its mass on?

- first term: $\mathbb{E}_{\mathbf{z}}[\log p(\boldsymbol{x} \mid \boldsymbol{z})]$
  - *i.e.* an expected likelihood
    - encourages densities that explain observed data[7]
- second term: $-D_{\mathrm{KL}}[q(\boldsymbol{z}) \,||\, p(\boldsymbol{z})]$
  - *i.e.* negative divergence between variational density and prior
  - $\Rightarrow$ encourages densities close to prior
- thus the variational objective mirrors
  - usual balance between likelihood and prior

[7] *i.e.* densities that place their mass on configurations of the latent variables that explain the observed data

Why called ELBO?

- ELBO $\mathcal{L}(q)$ lower-bounds the (log) evidence $p(\boldsymbol{x})$

  i.e. for any $q(\boldsymbol{z})$

$$\log p(\boldsymbol{x}) \geq \mathcal{L}(q)$$

- proof:
  - notice that (3) gives

$$\log p(\boldsymbol{x}) = \mathcal{L}(q) + \underbrace{D_{\mathrm{KL}}[q(\boldsymbol{z}) \,||\, p(\boldsymbol{z} \,|\, \boldsymbol{x})]}_{\geq 0}$$

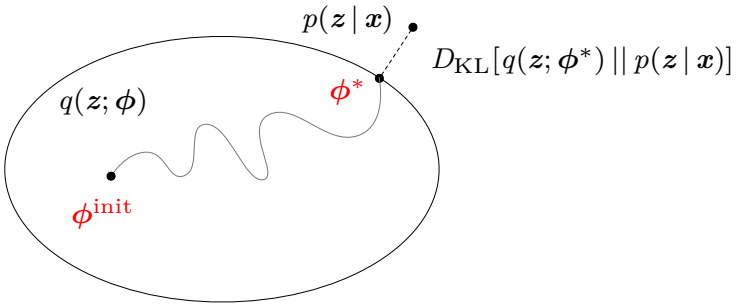
  - can also be derived through Jensen's inequality

Common restrictions on $\mathcal{Q}$

1. parameterization
   - use a parametric distribution governed by parameters $\boldsymbol{\phi}$

$$q(\boldsymbol{z}; \boldsymbol{\phi}) = q_{\boldsymbol{\phi}}(\boldsymbol{z})$$

   - $\boldsymbol{\phi}$: variational parameters



   - ELBO $\mathcal{L}$ now becomes a function of $\boldsymbol{\phi}$: $\mathcal{L}(q, \boldsymbol{\phi})$
   - inference $=$ finding $\boldsymbol{\phi}^*$
   - spoiler: VAE uses a neural net to model $q_{\boldsymbol{\phi}}(\boldsymbol{z})$ and sgd to train it

image source: Blei et al. (2017) Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877

2. factorization

- assume q factorizes (called "mean-field approach")

$$q(\boldsymbol{z}) = \prod_{j=1}^{m} q_j(z_j)$$

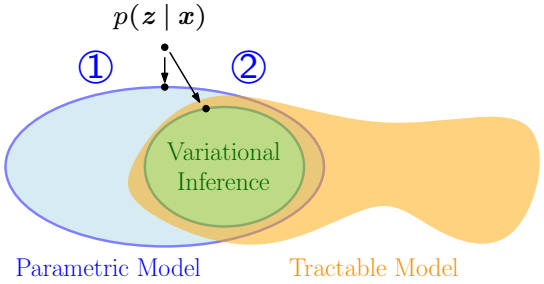
$\boldsymbol{z} = (z_1, z_2, \cdots, z_m)$

- each latent variable z_j :
  - ▷ governed by its own variational factor, the density $q_j(z_j)$
  - ▷ can take on any form appropriate to the corresponding random var

e.g. a continuous variable: a Gaussian factor

a categorical variable: a categorical factor

core idea behind VI: maximize $\mathcal{L}$ over a restricted family of distributions q



① $q(\boldsymbol{z}; \boldsymbol{\phi}_{\text{MLE}})$ Maximum Likelihood Expec

② $q(\boldsymbol{z}; \boldsymbol{\phi}_{\text{ELBO}})$

image source: Chen et al. (2018) On the use of bootstrap with variational inference: Theory, interpretation, and a two-sample test example. *The Annals of Applied Statistics*, 12(2):846–876

Variational inference recipe

1. start with a model: $p(\boldsymbol{x}, \boldsymbol{z})$
2. choose a variational approximation: $q(\boldsymbol{z}; \boldsymbol{\phi}) = q_{\boldsymbol{\phi}}(\boldsymbol{z})$
3. write down ELBO: $\mathcal{L}(\boldsymbol{\phi}) = \mathbb{E}_{\mathbf{z} \sim q(\mathbf{z}; \boldsymbol{\phi})}[\log p(\boldsymbol{x}, \boldsymbol{z}) - \log q(\boldsymbol{z}; \boldsymbol{\phi})]$
4. compute the expectation/integral: *e.g.* $\mathcal{L}(\boldsymbol{\phi}) = \boldsymbol{x}\boldsymbol{\phi}^2 + \log \boldsymbol{\phi}$
5. take derivatives: *e.g.* $\nabla_{\boldsymbol{\phi}} \mathcal{L}(\boldsymbol{\phi}) = 2\boldsymbol{x}\boldsymbol{\phi} + \frac{1}{\boldsymbol{\phi}}$
6. optimize: $\boldsymbol{\phi}_{t+1} = \boldsymbol{\phi}_t + \varepsilon \nabla_{\boldsymbol{\phi}} \mathcal{L}(\boldsymbol{\phi})$

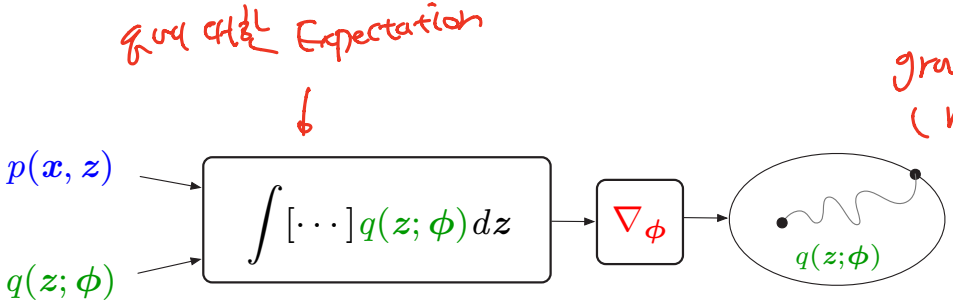


image source: Blei et al. (2017) Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877

Outline

Summary

- generative models: density estimation and sample generation
  - explicit density: PixelRNN/CNN, variational autoencoder (VAE)
  - implicit density: generative adversarial network (GAN)
- generative models are versatile and useful for many tasks
  - representation/manipulation of high-dim distributions
  - reinforcement learning, semi-supervised learning
  - multi-modal outputs, inference of latent representations
- approximate inference schemes fall into two classes: stochastic or deterministic
  - their strengths and weaknesses are complementary to each other
- variational inference: inference $\rightarrow$ optimization
  - goal: approximate posterior $p(\boldsymbol{z} \,|\, \boldsymbol{x}) \approx q(\boldsymbol{z}) \in \mathcal{Q}$ [$\mathbf{z}$: latent, $\mathbf{x}$: observed]
  - $\mathcal{Q}$: a family of restricted (parameterized/factorized) distributions over $\mathbf{z}$
  - $q^*(\boldsymbol{z}) = \operatorname{argmin}_{q(\boldsymbol{z}) \in \mathcal{Q}} D_{\mathrm{KL}}[q(\boldsymbol{z}) \,||\, p(\boldsymbol{z} \,|\, \boldsymbol{x})]$
  - VAE: we parameterize q using a neural net and optimize by backprop