Deep Generative Models

4. Maximum Likelihood Learning

- 국가수리과학연구소 산업수학혁신센터 김민중

Recap

- **Representation:** how do we model the joint distribution of many random variables?
    - Need compact representation
- **Bayesian network:** A probabilistic graphical model representing variables and their conditional dependencies
- Autoregressive property(no conditional independence)

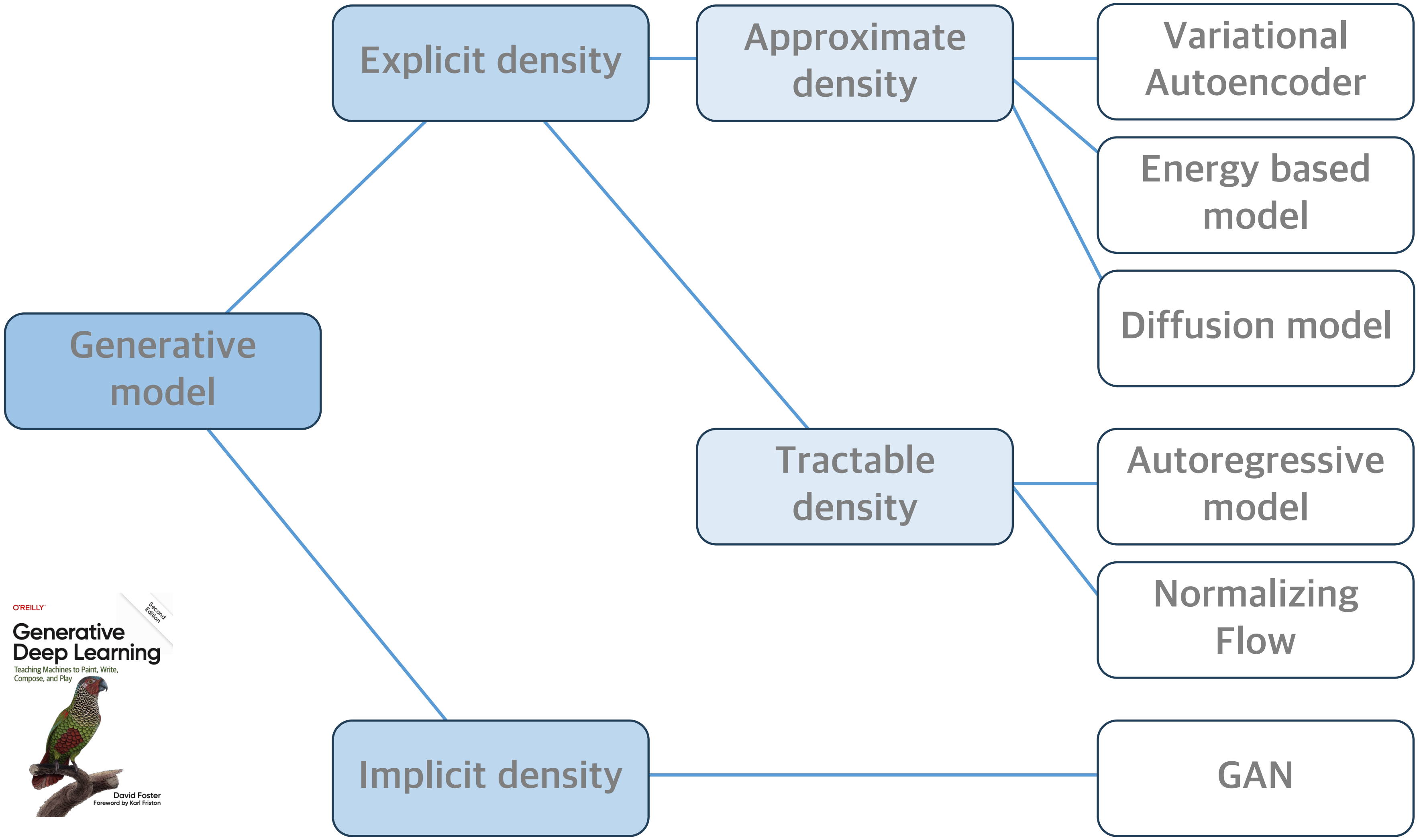
$$p(\boldsymbol{x}) = p(x_1)\prod_{i=2}^{d} p(x_i|\boldsymbol{x}_{<i})$$

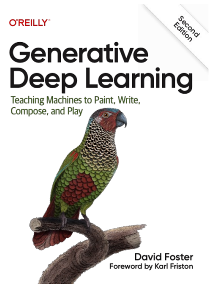
- For $i > 1$,

$$p_{\theta_i}(x_i|\boldsymbol{x}_{<i}) = Bern\big(x_i|f_i(\boldsymbol{x}_{<i})\big)$$

- where θ_i denotes the set of parameters used to specify the mean function $f_i\colon \{0,1\}^{i-1} \to (0,1)$

Taxonomy of Generative model approaches

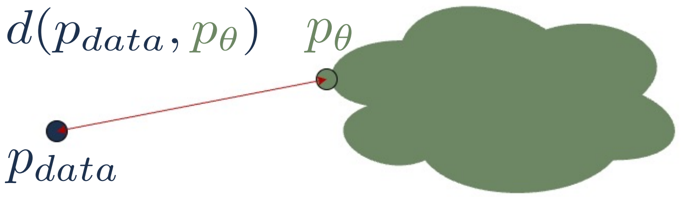




Learning a generative model

- We are given a training dataset of examples.

$$\mathbf{x}_i \sim p_{data}$$
$$i = 1, 2, ..., N$$



$\theta \in \mathcal{M}$

Model family

- Generation: sample $\boldsymbol{x}_{new}$ should look like training set(sampling)
- Density estimation
- Unsupervised representation learning: learn what these images have in common features
- 1st question: How to represent p_θ
- **2nd question: how to learn it**

Setting

- Let us assume that the domain is governed by some underlying distribution p_{data}
- We are given a dataset D of N samples from p_{data}
- The standard assumption is that the data instances are **independent and identically distributed (IID)**
- We are also given a family of models $\mathcal{M}$, and our task is to learn some "good" distribution in this set:
    - For example, $\mathcal{M}$ could be all Bayes nets with a given graph structure, for all possible choices of the CPD tables
    - For example, a FVSBN for all possible choices of the logistic regression parameters , $\theta =$ concatenation of all logistic regression coefficients

Goal of learning

- The goal of learning is to return a model p_θ that precisely captures the distribution p_{data} from which our data was sampled
- This is in general not achievable because of
  - limited data only provides a rough approximation of the true underlying distribution
  - computational reasons
- We want to select p_θ to construct the ”best” approximation to the underlying distribution p_{data}
- What is **”best”?**

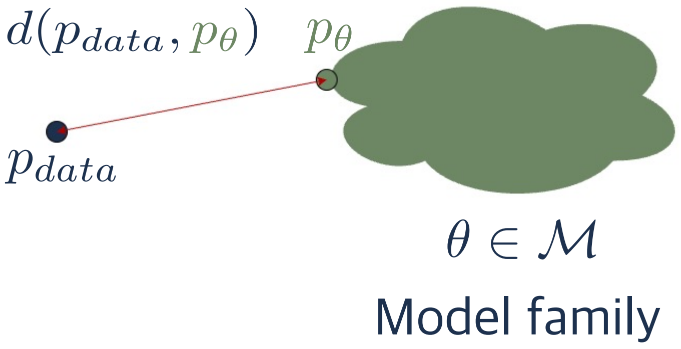
What is “best”?

- This depends on what we want to do
  - Density estimation: we are interested in the full distribution (so later we can compute whatever conditional probabilities we want)
  - Specific prediction tasks: we are using the distribution to make a prediction
    - Is this email spam or not?
    - Structured prediction: Predict next frame in a video, or caption given an image

Learning as density estimation

- We want to learn the full distribution so that later we can answer any probabilistic inference query
- In this setting, we can view the learning problem as density estimation
- We want to construct p_θ as "close" as possible to p_{data} (recall we assume we are given a dataset D of samples from p_{data})

$$\mathbf{x}_i \sim p_{data}$$
$$i = 1, 2, ..., N$$



- How do we evaluate "closeness "?

KL-divergence

- How should we measure distance between distributions?
- **The Kullback-Leibler divergence (KL-divergence)** between two distributions p and q is defined as

$$D(p \parallel q) := -\sum_{\boldsymbol{x}} p(\boldsymbol{x}) \log \frac{q(\boldsymbol{x})}{p(\boldsymbol{x})}$$

- $D(p \parallel q) \geq 0$ for all p and q, with equality if and only if $p = q$.
  - Prove it(exercise)

$$E_{\boldsymbol{x} \sim p}\left[-\log \frac{q(\boldsymbol{x})}{p(\boldsymbol{x})}\right] \geq -\log\left(E_{\boldsymbol{x} \sim p}\left[\frac{q(\boldsymbol{x})}{p(\boldsymbol{x})}\right]\right)$$

$$= -\log\left(\sum_{\boldsymbol{x}} p(\boldsymbol{x}) \frac{p(\boldsymbol{x})}{q(\boldsymbol{x})}\right) = 0$$

- KL-divergence is asymmetric, i.e., $D(p \parallel q) \neq D(q \parallel p)$

Learning as density estimation

- We want to learn the full distribution so that later we can answer any probabilistic inference query
- In this setting, we can view the learning problem as density estimation
- We want to construct p_θ as ”close” as possible to p_{data} (recall we assume we are given a dataset D of samples from p_{data})
- How do we evaluate “closeness ”?
- KL-divergence is one possibility:

$$D(p_{data} \parallel p_\theta) = E_{\boldsymbol{x} \sim p_{data}}\left[\log \frac{p_{data}(\boldsymbol{x})}{p_\theta(\boldsymbol{x})}\right]$$

- $D(p_{data} \parallel p_\theta) = 0$ iff two distributions are equal

Expected log-likelihood

- We can simplify this somewhat:

$$D(p_{data} \parallel p_\theta) = E_{\boldsymbol{x} \sim p_{data}}\left[\log \frac{p_{data}(\boldsymbol{x})}{p_\theta(\boldsymbol{x})}\right]$$
$$= E_{\boldsymbol{x} \sim p_{data}}[\log p_{data}(\boldsymbol{x})] - E_{\boldsymbol{x} \sim p_{data}}[\log p_\theta(\boldsymbol{x})]$$

- The first term does not depend on p_θ
- Then, minimizing KL divergence is equivalent to maximizing the expected log-likelihood

$$\arg\min_{p_\theta} D(p_{data} \parallel p_\theta) = \arg\min_{p_\theta} -E_{\boldsymbol{x} \sim p_{data}}[\log p_\theta(\boldsymbol{x})]$$
$$= \arg\max_{p_\theta} E_{\boldsymbol{x} \sim p_{data}}[\log p_\theta(\boldsymbol{x})]$$

  - Asks that p_θ assign high probability to instances sampled from p_{data}, to reflect the true distribution
  - Because of $\log$, samples $\boldsymbol{x}$ where $p_\theta(\boldsymbol{x}) \approx 0$ weigh heavily in objective

Maximum likelihood

- Approximate the expected log-likelihood

$$E_{\boldsymbol{x}\sim p_{data}}[\log p_\theta(\boldsymbol{x})]$$

- with the empirical log-likelihood:

$$E_D[\log p_\theta(\boldsymbol{x})] = \frac{1}{|D|}\sum_{\boldsymbol{x}\in D} \log p_\theta(\boldsymbol{x})$$

- Maximum likelihood learning is then:

$$\arg\max_{p_\theta} \frac{1}{|D|}\sum_{\boldsymbol{x}\in D} \log p_\theta(\boldsymbol{x})$$

Main idea in Monte Carlo Estimation

- Express the quantity of interest as the expected value of a random variable

$$E_{\boldsymbol{x}\sim p}[g(\boldsymbol{x})] = \int g(\boldsymbol{x})p(\boldsymbol{x})d\boldsymbol{x} = \sum_{\boldsymbol{x}} g(\boldsymbol{x})p(\boldsymbol{x})$$

- Generate N samples $\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \cdots, \boldsymbol{x}^{(N)}$ from the distribution p with respect to which the expectation was taken

$$\hat{g}\left(\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \cdots, \boldsymbol{x}^{(N)}\right) := \frac{1}{N}\sum_{n=1}^{N} g(\boldsymbol{x}^{(n)})$$

- where $\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \cdots, \boldsymbol{x}^{(N)}$ are independent samples from p
- Note that $\hat{g}$ is a random variable

Properties of the Monte Carlo Estimate

- Unbiased

$$E[\hat{g}] = E_{\boldsymbol{x}\sim p}[g(\boldsymbol{x})]$$

- Convergence: By law of large numbers

$$\hat{g} = \frac{1}{N}\sum_{n=1}^{N} g(\boldsymbol{x}^{(n)}) \to E_{\boldsymbol{x}\sim p}[g(\boldsymbol{x})] \text{ for } N \to \infty$$

- Variance

$$V[\hat{g}] = V\left[\frac{1}{N}\sum_{n=1}^{N} g(\boldsymbol{x}^{(n)})\right] = \frac{V_{\boldsymbol{x}\sim p}[g(\boldsymbol{x})]}{N}$$

- Thus, variance of the estimator can be reduced by increasing the number of samples.

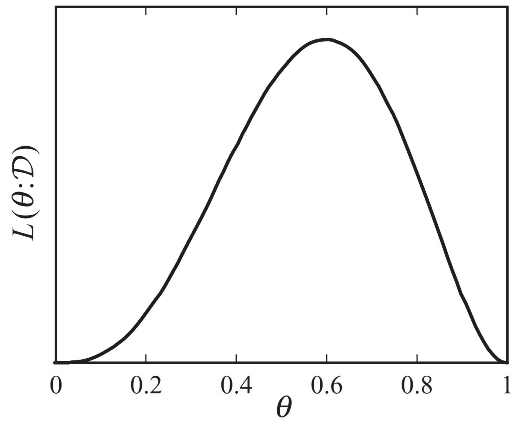
Single variable example: A biased coin

- Two outcomes: heads (H) and tails (T)
- Data set: Tosses of the biased coin, e.g., $D = \{H, H, T, H, T\}$
- Assumption: the process is controlled by a probability distribution $p_{data}(x)$ where $x \in \{H, T\}$
- Class of models $\mathcal{M}$: all probability distributions over $x \in \{H, T\}$
- Example learning task: How should we choose $p_\theta(x)$ from $\mathcal{M}$ if 3 out of 5 tosses are heads in D?

MLE scoring for the coin example

- We represent our model: $p_\theta(x = H) = \theta$ and $p_\theta(x = T) = 1 - \theta$
- Observed data: $D = \{H, H, T, H, T\}$
- Likelihood of data

$$\prod_i p_\theta\left(x^{(i)}\right) = \theta \cdot \theta \cdot (1 - \theta) \cdot \theta \cdot (1 - \theta)$$



- Optimize for θ which makes D most likely
- What is the solution in this case? $\theta = 0.6$, optimization problem can be solved in closed-form

Extending the MLE principle to autoregressive models

- Given an autoregressive model with d variables and factorization

$$p_\theta(\boldsymbol{x}) = p_{\theta_1}(x_1) \prod_{i=2}^{d} p_{\theta_i}(x_i | \boldsymbol{x}_{<i})$$

- where $\theta = (\theta_1, \cdots, \theta_d)$ are the parameters of all the conditionals
- Training data $D = \{\boldsymbol{x}^{(1)}, \cdots, \boldsymbol{x}^{(N)}\}$
- Maximum likelihood estimate of the parameters θ?
  - Decomposition of Likelihood function

$$L(\theta, D) = \prod_{n=1}^{N} p_\theta\left(\boldsymbol{x}^{(n)}\right) = \prod_{n=1}^{N} \prod_{i=1}^{d} p_{\theta_i}\left(x_i^{(n)} \middle| \boldsymbol{x}_{<i}^{(n)}\right)$$

  - Goal: $\arg\max_\theta L(\theta, D) = \arg\max_\theta \log L(\theta, D)$

MLE Learning: Gradient Descent

$$L(\theta, D) = \prod_{n=1}^{N} p_\theta\left(\boldsymbol{x}^{(n)}\right) = \prod_{n=1}^{N} \prod_{i=1}^{d} p_{\theta_i}\left(x^{(n)}{}_i \middle| \boldsymbol{x}_{<i}^{(n)}\right)$$

- Goal: $\arg\max_\theta L(\theta, D) = \arg\max_\theta \log L(\theta, D)$
- Let $\ell(\theta) \coloneqq L(\theta, D)$
  - Initialize $\theta^0 = \left(\theta_1^0, \cdots, \theta_d^0\right)$ at random
  - Compute $\nabla_\theta \ell(\theta)$(by back propagation)
  - $\theta^{t+1} = \theta^t - \alpha_t \nabla_\theta \ell(\theta)$
- Non-convex optimization problem, but often works well in practice

MLE Learning: Stochastic Gradient Descent

$$\ell(\theta) = \log L(\theta, D) = \prod_{n=1}^{N} \prod_{i=1}^{d} \log p_{\theta_i}\left(x_i^{(n)} \middle| \boldsymbol{x}_{<i}^{(n)}\right)$$

- $\ell(\theta) = \log L(\theta, D)$
    - Initialize $\theta^0 = \left(\theta_1^0, \cdots, \theta_d^0\right)$ at random
    - Compute $\nabla_\theta \ell(\theta)$(by back propagation)
    - $\theta^{t+1} = \theta^t - \alpha_t \nabla_\theta \ell(\theta)$
- What is the gradient with respect to θ_k? (no parameter sharing)

$$\nabla_{\theta_k} \ell(\theta) = \sum_{n=1}^{N} \nabla_{\theta_k} \sum_{i=1}^{d} \log p_{\theta_i}\left(x_i^{(n)} \middle| \boldsymbol{x}_{<i}^{(n)}\right)$$

$$= \sum_{n=1}^{N} \nabla_{\theta_k} \log p_{\theta_k}\left(x_k^{(n)} \middle| \boldsymbol{x}_{<k}^{(n)}\right)$$

MLE Learning: Stochastic Gradient Descent

- Initialize $\theta^0 = (\theta_1^0, \cdots, \theta_d^0)$ at random
- Compute $\nabla_\theta \ell(\theta)$(by back propagation)
- $\theta^{t+1} = \theta^t - \alpha_t \nabla_\theta \ell(\theta)$

$$\nabla_\theta \ell(\theta) = \sum_{n=1}^{N} \sum_{i=1}^{d} \nabla_\theta \log p_{\theta_i}\left(x_i^{(n)} \middle| \boldsymbol{x}_{<i}^{(n)}\right)$$

- What if $N = |D|$ is huge?

$$\nabla_\theta \ell(\theta) = N \sum_{n=1}^{N} \frac{1}{N} \sum_{i=1}^{d} \nabla_\theta \log p_{\theta_i}\left(x_i^{(n)} \middle| \boldsymbol{x}_{<i}^{(n)}\right)$$

$$= E_{x^{(n)} \sim D}\left[N \sum_{i=1}^{d} \nabla_\theta \log p_{\theta_i}\left(x_i^{(n)} \middle| \boldsymbol{x}_{<i}^{(n)}\right)\right]$$

- Monte Carlo: $x^{(n)} \sim D$; $\nabla_\theta \ell(\theta) \approx N \sum_{i=1}^{d} \nabla_\theta \log p_{\theta_i}\left(x_i^{(n)} \middle| \boldsymbol{x}_{<i}^{(n)}\right)$

Empirical Risk and Overfitting

- Empirical risk minimization can easily overfit the data
  - Extreme example: The data is the model (remember all training data)
- Generalization: the data is a sample, usually there is vast number of samples that you have never seen
- Your model should generalize well to these "never-seen" samples
- Thus, we typically restrict the hypothesis space of distributions that we search over

How to avoid Overfitting?

- Hard constraints, e.g., by selecting a less expressive model family
  - Smaller neural networks with less parameters
  - Weight sharing
- Soft preference for “simpler” models: Occam Razor
- Augment the objective function with regularization

$$objective(\boldsymbol{x}, M) = loss(\boldsymbol{x}, M) + R(M)$$

- Evaluate generalization performance on a held-out validation set

Recap

- For autoregressive models, it is easy to compute $p_\theta(\boldsymbol{x})$
  - When parameters are not shared, evaluate in parallel each conditional $\log p_{\theta_i}\left(x_i^{(n)} \middle| \boldsymbol{x}_{<i}^{(n)}\right)$
- Natural to train them via maximum likelihood
- Higher log-likelihood doesn't necessarily mean better looking samples

Thanks