Deep Generative Models

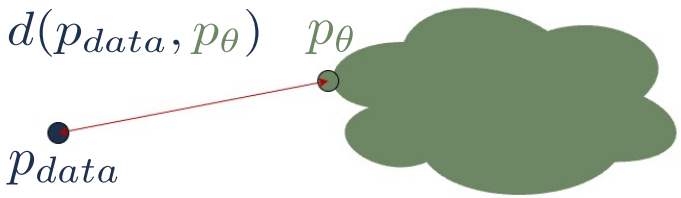
3. Autoregressive Models

- 국가수리과학연구소 산업수학혁신센터 김민중

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 probability distribution?
- 2nd question: How to learn it

Recap: Bayesian networks vs neural models

- Using Chain rule, full general form is

$$p(x_1, x_2, x_3, x_4) = p(x_1)p(x_2|x_1)p(x_3|x_1, x_2)p(x_4|x_1, x_2, x_3)$$

- Bayes Net under conditional independencies

$$p(x_1, x_2, x_3, x_4) \approx p(x_1)p(x_2|x_1)p(x_3|\cancel{x_1}, x_2)p(x_4|\cancel{x_1, x_2}, x_3)$$

- Neural Models

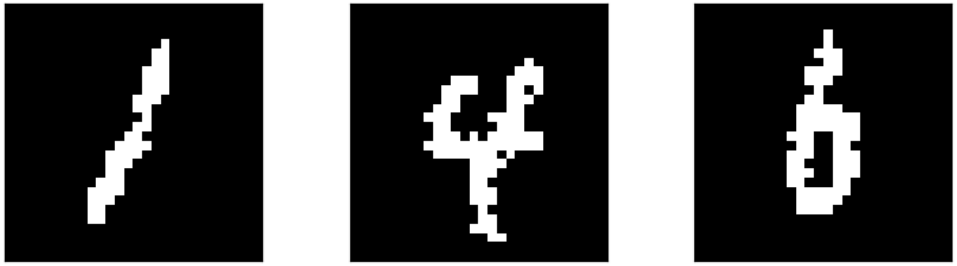
$$p(x_1, x_2, x_3, x_4)$$
$$\approx p(x_1)p_{neural}(x_2|x_1)p_{neural}(x_3|x_1, x_2)p_{neural}(x_4|x_1, x_2, x_3)$$

  - Assume specific functional form for the conditionals
  - A sufficiently deep neural net can approximate any function

Recap: Neural Models for classification

- For classification, we care about $p(Y = 1|\boldsymbol{x})$ and assume that
$$p(Y = 1|\boldsymbol{x}; \boldsymbol{\alpha}) = f(\boldsymbol{x}, \boldsymbol{\alpha})$$
- Linear dependence:
  - Let $z(\boldsymbol{\alpha}, \boldsymbol{x}) \coloneqq \alpha_0 + \sum_{i=1}^{d} \alpha_i x_i$
  - $p(Y = 1|\boldsymbol{x}; \boldsymbol{\alpha}) = \sigma(z(\boldsymbol{\alpha}, \boldsymbol{x}))$ where $\sigma(z) = 1/(1 + e^{-z})$
- Non-linear dependence: let $\boldsymbol{h}(W, \boldsymbol{b}, \boldsymbol{x}) = f(W\boldsymbol{x} + \boldsymbol{b})$ be a non-linear transformation of the inputs (features)
$$p_{neural}(Y = 1|x; \boldsymbol{\alpha}, W, \boldsymbol{b}) = \sigma\left(z\left(\boldsymbol{\alpha}, \boldsymbol{h}(W, \boldsymbol{b}, \boldsymbol{x})\right)\right)$$
$$= \sigma\left(\alpha_0 + \sum_{i=1}^{h} \alpha_i h_i\right)$$
  - Parameters: $\boldsymbol{\alpha}, W, \boldsymbol{b}$
  - Can repeat multiple times to get a neural network

Motivating Example: MNIST



- Given: a dataset D of handwritten digits (binarized MNIST)
- Each image has $d = 28 \times 28 = 784$ pixels. Each pixel can either be black (0) or white (1)
- **Goal**: Learn a probability distribution $p(\boldsymbol{x}) = p(x_1, \cdots, x_{784})$ over $x_i \in \{0,1\}^{784}$ such that when $\boldsymbol{x} \sim p_\theta(\boldsymbol{x})$, $\boldsymbol{x}$ looks like a digit
- Two step process:
  - Parameterize a model family $\{p_\theta(\boldsymbol{x}), \theta \in \mathcal{M}\}$
  - Search for model parameters θ based on training data D

Autoregressive model

- Given a dataset D of d-dimensional datapoint $\boldsymbol{x}$
- For simplicity, let $\boldsymbol{x} \in \{0,1\}^d$
- By the chain rule, (no conditional independence)

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

- where $\boldsymbol{x}_{<i} = (x_1, x_2, \cdots, x_{i-1})^T$ denotes the vector of random variables with index less than i
- Such a Bayesian network is said to obey the autoregressive property

Autoregressive model

- Fix an ordering of variables
- x_i random variable depends on $x_1, x_2, \cdots, x_{i-1}$
- To fully specify $p(x_i|\boldsymbol{x}_{<i})$, we need to specify a probability for 2^{i-1} configurations of the variable $x_1, x_2, \cdots, x_{i-1}$

Autoregressive model

- The conditionals are specified as parameter functions with a fixed number of parameters
- Assume $p(x_i|\boldsymbol{x}_{<i})$~Bernoulli random variable and learn a function mapping $\boldsymbol{x}_{<i}$ to the mean of this Bernoulli distribution
- Hence, 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 $f_i: \{0,1\}^{i-1} \rightarrow (0,1)$
- # of parameters of an autoregressive generative model is $\sum_{i=1}^{d} |\theta_i|$

Fully-visible sigmoid belief network(FVSBN)

- Specify the function as a linear combination of the input elements followed by a sigmoid non-linearity

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

$$f_i(\boldsymbol{x}_{<i}) = \sigma(\boldsymbol{x}_{<i}^T \boldsymbol{w}^{(i)} + b_i)$$

- where σ is a sigmoid, $\boldsymbol{w}^{(i)} \in \mathbb{R}^{i-1}$ and $b_i \in \mathbb{R}$. i.e., $\theta_i = \{\boldsymbol{w}^{(i)}, b_i\}$
- Since f_i requires i parameters, the total number of parameters is $\sum_{i=1}^{d} i = O(d^2)$

Fully-visible sigmoid belief network(FVSBN)

- E.g., $\boldsymbol{x} = (x_1, x_2, x_3)^T$ with $x_i \in \{0,1\}$, then
$$p(\boldsymbol{x}) = p(x_1)p(x_2|x_1)p(x_3|x_1, x_2)$$
- FVSBN model is
$$p_\theta(\boldsymbol{x}) = p_{\theta_1}(x_1)p_{\theta_2}(x_2|x_1)p_{\theta_3}(x_3|x_1, x_2)$$
- where
$$p_{\theta_1}(x_1) = Bern(x_1|\theta_1),\ 0 < \theta_1 < 1,$$
$$p_{\theta_2}(x_2|x_1) = Bern(x_2|f_2(x_1)),\ f_2(x_1) = \sigma(w_1^{(2)}x_1 + b_2)$$
$$p_{\theta_3}(x_3|x_1, x_2) = Bern(x_3|f_3(x_1, x_2)),$$
$$f_3(x_1, x_2) = \sigma(w_1^{(3)}x_1 + w_2^{(3)}x_2 + b_3)$$
- # of trainable parameters is $\sum_{i=1}^{3} |\theta_i| = 1 + 2 + 3$

Fully-visible sigmoid belief network(FVSBN)

- FVSBN model is

$$p_\theta(\boldsymbol{x}) = p_{\theta_1}(x_1)p_{\theta_2}(x_2|x_1)p_{\theta_3}(x_3|x_1,x_2)$$

- where

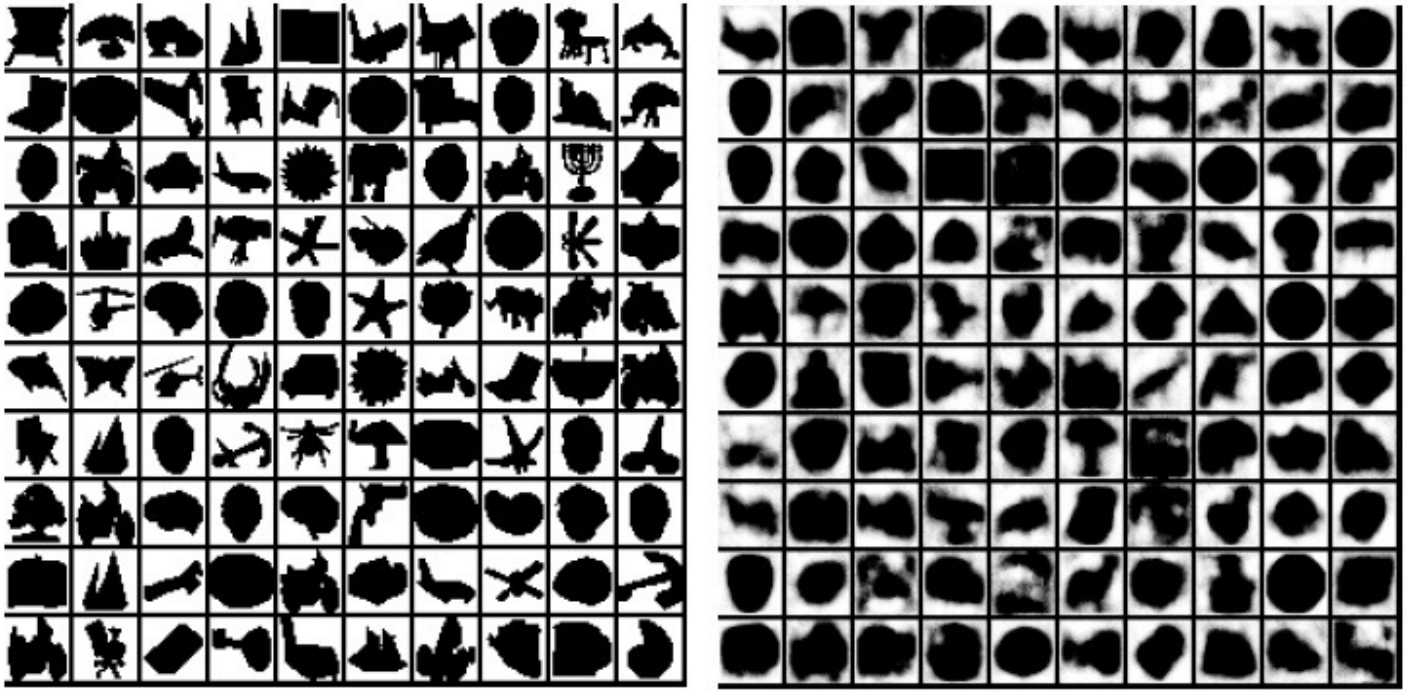
$$p_{\theta_1}(x_1) = Bern(x_1|\theta_1),\ 0 < \theta_1 < 1,$$
$$p_{\theta_2}(x_2|x_1) = Bern(x_2|f_2(x_1)),\ f_2(x_1) = \sigma(w_1^{(2)}x_1 + b_2)$$
$$p_{\theta_3}(x_3|x_1,x_2) = Bern(x_3|f_3(x_1,x_2)),$$
$$f_3(x_1,x_2) = \sigma(w_1^{(3)}x_1 + w_2^{(3)}x_2 + b_3)$$

- # of trainable parameters is $\sum_{i=1}^{3}|\theta_i| = 1 + 2 + 3$
- Since $Bern(x|\mu) = \mu^x(1-\mu)^{(1-x)}$,

$$p_\theta(\boldsymbol{x}) = \theta_1^{x_1}(1-\theta_1)^{(1-x_1)}f_2(x_1)^{x_2}(1-f_2(x_1))^{(1-x_2)}f_3(x_1,x_2)^{x_3}\big(1 - f_3(x_1,x_2)\big)^{(1-x_3)}$$

Fully-visible sigmoid belief network(FVSBN)

- Training data on the left (Caltech 101 Silhouettes)
- Samples from the model on the right.
- Figure from Learning Deep Sigmoid Belief Networks with Data Augmentation, 2015.



Fully-visible sigmoid belief network(FVSBN)

- More flexible parametrizations for the mean function e.g., MLP
- For example, consider 1 hidden layer neural network. i.e.,

$$\boldsymbol{h}_i(\boldsymbol{x}_{<i}) = \sigma(A_i \boldsymbol{x}_{<i} + \boldsymbol{c}_i)$$
$$p_{\theta_i}(x_i|\boldsymbol{x}_{<i}) = Bern\big(x_i|f_i(\boldsymbol{x}_{<i})\big)$$
$$f_i(\boldsymbol{x}_{<i}) = \sigma(\boldsymbol{h}_i^T \boldsymbol{w}^{(i)} + b_i)$$

- where $\boldsymbol{h}_i \in \mathbb{R}^h$ denotes the hidden layer activations for MLP and $\theta_i \in \{A_i \in \mathbb{R}^{h\times(i-1)}, \boldsymbol{c}_i \in \mathbb{R}^h, \boldsymbol{w}^{(i)} \in \mathbb{R}^h, b_i \in \mathbb{R}\}$ are the set of parameters for the mean function
- # of parameters is dominated by $|A_i|$ so given by $O(d^2h)$

$$\boldsymbol{h}_2 = \sigma\left(\underbrace{\begin{pmatrix}\cdot\\ \cdot\\ \cdot\end{pmatrix}}_{A_2} x_1 + \begin{pmatrix}\cdot\\ \cdot\\ \cdot\end{pmatrix}\right), \boldsymbol{h}_3 = \sigma\left(\underbrace{\begin{pmatrix}\cdot & \cdot\\ \cdot & \cdot\\ \cdot & \cdot\end{pmatrix}}_{A_3}\begin{pmatrix}x_1\\ x_2\end{pmatrix} + \begin{pmatrix}\cdot\\ \cdot\\ \cdot\end{pmatrix}\right)$$

Neural Autoregressive Density estimator(NADE)

- An alternation of MLP-based parametrization
- In NADE, parameters are shared across the functions used for evaluating the conditionals
- In particular,

$$\boldsymbol{h}_i(\boldsymbol{x}_{<i}) = \sigma(A_{.,<i}\boldsymbol{x}_{<i} + \boldsymbol{c})$$
$$p_{\theta_i}(x_i|\boldsymbol{x}_{<i}) = Bern\big(x_i|f_i(\boldsymbol{x}_{<i})\big)$$
$$f_i(\boldsymbol{x}_{<i}) = \sigma(\boldsymbol{h}_i^T\boldsymbol{w}^{(i)} + b_i)$$

- where $\theta \in \{A \in \mathbb{R}^{h\times d}, \boldsymbol{c} \in \mathbb{R}^h, \{\boldsymbol{w}^{(i)} \in \mathbb{R}^h\}_{i=1}^d, \{b_i \in \mathbb{R}\}_{i=1}^d\}$ is the full parameters for the mean function
- The weight matrix A and the bias vector $\boldsymbol{c}$ are shared across the conditionals
- # of parameters gets reduced from $O(d^2h)$ to $O(dh)$

Neural Autoregressive Density estimator(NADE)

- Samples from a model trained on MNIST on the left
- Figure from The Neural Autoregressive Distribution Estimator, 2011

Thanks