Deep Generative Models

2. Representation

• 국가수리과학연구소 산업수학혁신센터 김민중

Overview

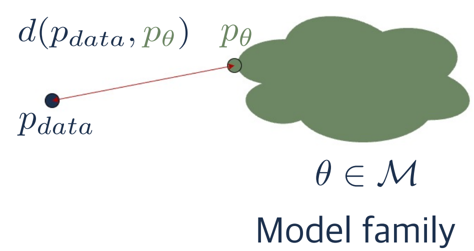
- Representing probability distributions
  - Curse of dimensionality
  - Crash course on graphical model(Bayesian networks)
  - Neural models

Road map and Challenges

- **Representation**: how do we model the joint distribution of many random variables?
  - Need compact representation
- **Learning**: what is the right way to compare probability distributions?

$\mathbf{x}_i \sim p_{data}$

$i = 1, 2, ..., N$

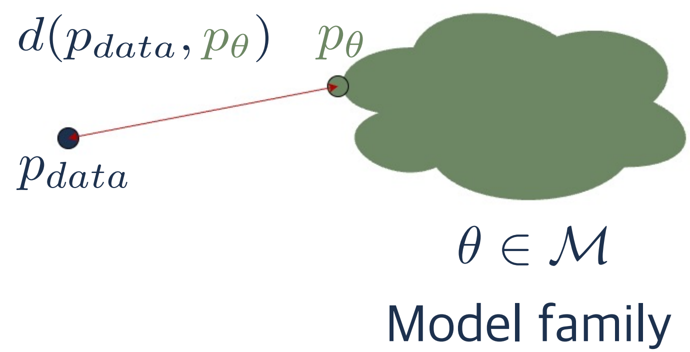


- **Inference**: how do we invert the generation process (e.g., vision as inverse graphics)?
  - Unsupervised learning: recover high-level descriptions (features) from raw data

Learning a generative model

- We are given a training dataset of examples.

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



- Our goal is to learn the parameters of a generate model θ within a model family $\mathcal{M}$ s.t. the model distribution p_θ is close to the distribution p_{data}

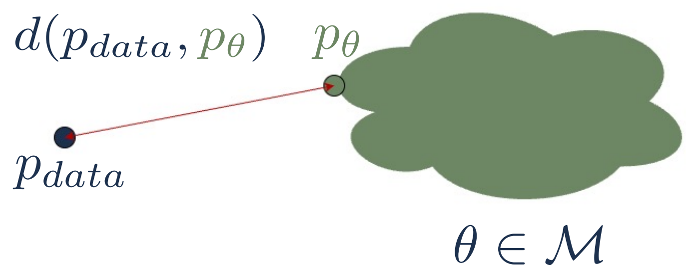
Learning a generative model

- Mathematically, we can specify our goal as the following optimization problem:

$$\min_{\theta \in \mathcal{M}} d(p_{data}, p_\theta)$$

- where p_{data} is accessed via the dataset D and $d(\cdot,\cdot)$ is a notion of distance between probability distributions

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



Model family

The purpose of generative model

- Generation: sample $\boldsymbol{x}_{new}$ should look like training set(sampling)
- Density estimation
- Unsupervised representation learning: learn what these images have in common features

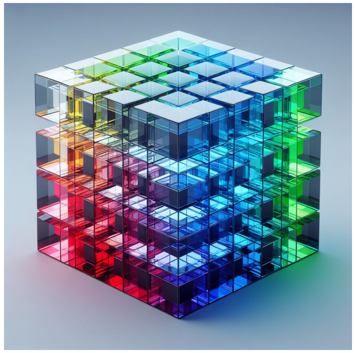
- How to represent probability distribution?

Basic discrete distributions

- Bernoulli distribution: (biased) coin flip
  - $D = \{H, T\}$
  - Specify $p(X = H) = \mu$. Then $p(X = T) = 1 - \mu$.
  - Write: $X \sim Ber(\mu)$ or $X = Ber(X|\mu)$
  - Sampling: flip a (biased) coin
- Categorical distribution: (biased) K-sided dice
  - $D = \{1,2,\cdots,K\}$
  - Specify $p(Y = i) = \mu_i$ such that $\sum_{i=1}^{K} \mu_i = 1$
  - Write: $Y \sim Cat(\mu_1, \mu_2, \cdots, \mu_K)$ or $Y = Cat(Y|\mu_1, \mu_2, \cdots, \mu_K)$
  - Sampling: roll a (biased) dice

Example of joint distribution

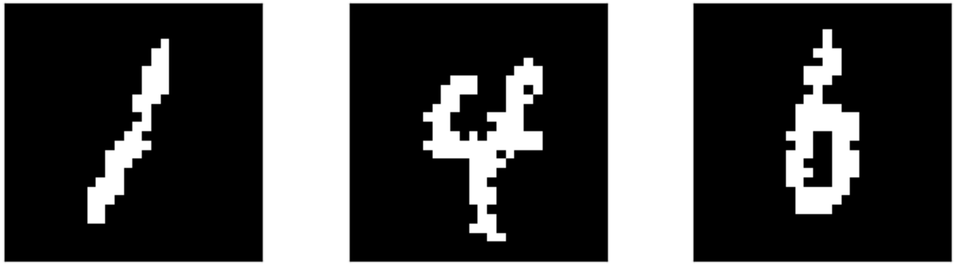
- Modeling a single pixel's color. Three discrete random variables:
  - Red channel $R \in \{0, \cdots, 255\}$
  - Green channel $G \in \{0, \cdots, 255\}$
  - Blue channel $B \in \{0, \cdots, 255\}$



- Sampling from the joint distribution $(r, g, b) \sim p(R, G, B)$ randomly generates a color for the pixel
  - How many parameters do we need to specify the joint distribution $p(R = r, G = g, B = b)$?

Answer: $256 \cdot 256 \cdot 256 - 1$

Example of joint distribution



- Suppose $X_1, X_2, \cdots, X_d$ are binary (Bernoulli) random variables with d-dim binary image, i.e., $x_i \in \{0,1\} = \{Black, White\}$
- How many possible images (states)?

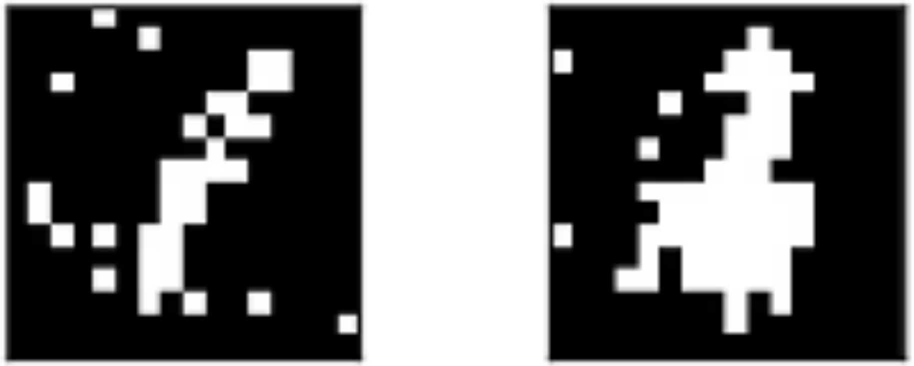
Answer: 2^d

- Sampling from $p(x_1, x_2 \cdots, x_d)$ generates an image
- How many parameters to specify the joint distribution over n binary pixels? (why? show it later)

Answer: $2^d - 1$

Structure through independence

- If $X_1, X_2, \cdots, X_d$ are independent, then
$$p(x_1, x_2 \cdots, x_d) = p(x_1)p(x_2)\cdots p(x_d)$$
- How many parameters specify the joint distribution $p(x_1, x_2 \cdots, x_d)$?
  - 2^d entries can be described by just d numbers
- However, independence assumption is too strong. Model not likely to be useful
  - E.g., each pixel is chosen independently



Structure through conditional independence

- Back to general case, using Chain Rule

$$p(x_1, x_2 \cdots, x_d)$$
$$= p(x_1)p(x_2|x_1)p(x_3|x_1, x_2) \cdots p(x_d|x_1, x_2, \cdots, x_{d-1})$$

- How many parameters? $1 + 2 + \cdots 2^{d-1} = 2^d - 1$
  - $p(x_1)$ requires 1 parameter
  - $p(x_2|x_1 = 0)$ requires 1 parameter
  - $p(x_2|x_1 = 1)$ requires 1 parameter
  - …
- $2^d - 1$ is still exponential. I.e., the chain rule does not give us anything
- Now suppose $X_{i+1} \perp X_1, \cdots, X_{i-1}|X_i$ (conditional independence)

$$p(x_1, x_2 \cdots, x_d)$$
$$= p(x_1)p(x_2|x_1)p(x_3|\cancel{x_1}, x_2) \cdots p(x_d|\cancel{x_1, x_2, \cdots}, x_{d-1})$$

- We only need $2d - 1$ parameters

Bayes Network

- Use conditional parameterization (instead of joint parametrization)
- For each random variable x_i , specify $p(x_i|x_{A_i})$ for set x_{A_i} of random variables $(A_i \subset \{1,2,\cdots,d\} \setminus \{i\})$
- Then get joint parametrization as

$$p(x_1, x_2, \cdots, x_d) = \prod_i p(x_i|x_{A_i})$$

- Need to guarantee it is a legal probability distribution
- It must correspond to a chain rule factorization, with factors simplified due to assumed conditional independencies

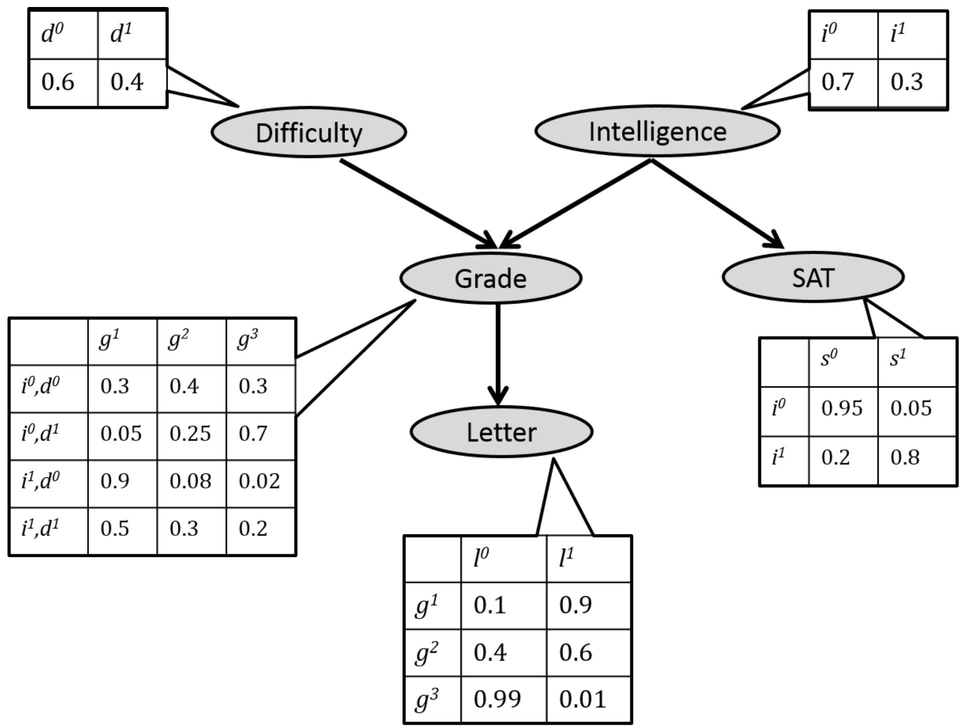
Bayes Network

- A Bayesian network is specified by a directed acyclic graph (DAG) $G = (V, E)$ with:
  - One node $i \in V$ for each random variable X_i
  - One conditional probability distribution (CPD) per node, $p(x_i | \boldsymbol{x}_{Pa(i)})$, specifying the variable's probability conditioned on its parents' values
- Graph $G = (V, E)$ is called the structure of the Bayesian Network
- Defines a joint distribution:

$$p(x_1, x_2, \cdots, x_d) = \prod_{i \in V} p\left(x_i \middle| \boldsymbol{x}_{Pa(i)}\right)$$

Example

- Consider the following Bayesian network:



d^0	d^1
0.6	0.4

i^0	i^1
0.7	0.3

	g^1	g^2	g^3
i^0,d^0	0.3	0.4	0.3
i^0,d^1	0.05	0.25	0.7
i^1,d^0	0.9	0.08	0.02
i^1,d^1	0.5	0.3	0.2

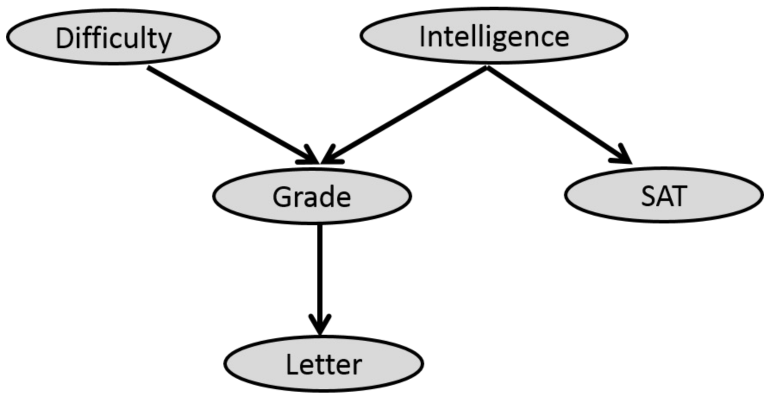
	s^0	s^1
i^0	0.95	0.05
i^1	0.2	0.8

	l^0	l^1
g^1	0.1	0.9
g^2	0.4	0.6
g^3	0.99	0.01

- What is its joint distribution?

$$p(d,i,g,s,l) = p(d)p(i)p(g|i,d)p(s|i)p(l|g)$$

Bayesian network structure implies conditional independencies



- The joint distribution corresponding to the above BN factors as

$$p(d, i, g, s, l) = p(d)p(i)p(g|i, d)p(s|i)p(l|g)$$

- However, by the chain rule, any distribution can be written as

$$p(d, i, g, s, l) = p(d)p(i|d)p(g|i, d)p(s|i, d, g)p(l|g, d, i, s)$$

- Thus, we are assuming the following additional independencies:

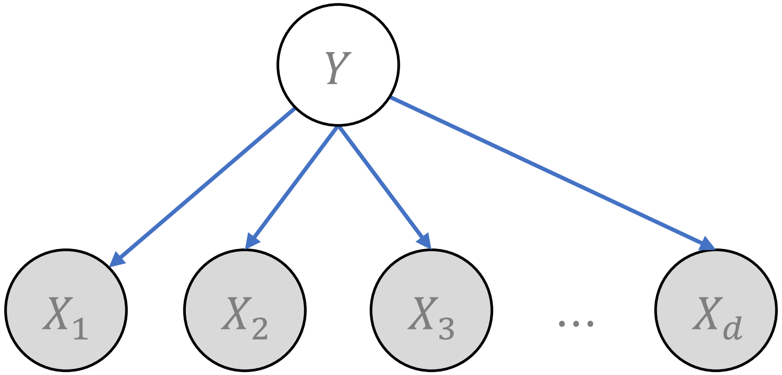
$$D \perp I, \qquad S \perp \{D, G\}|I, \qquad L \perp \{I, D, S\}|G$$

Summary

- Bayesian networks given by (G, p) where p is specified as a set of local conditional probability distributions associated with G's nodes
- Efficient representation using a graph-based data structure
- Computing the probability of any assignment is obtained by multiplying CPDs
- Can sample from the joint by sampling from the CPDs according to the DAG ordering
- Can identify some conditional independence properties by looking at graph properties
- In this class, graphical models will be simple (e.g., only 2 or 3 random vectors)

Example: Naïve Bayes for single label prediction

- Classify e-mails as spam ($Y = 1$) or not spam ($Y = 0$)
  - Let $1{:}d$ index be the words in our vocabulary
  - $X_i = 1$ if word i appears in an e-mail, and 0 otherwise
  - E-mails are drawn according to some distribution $p(Y, X_1, X_2, \cdots, X_d)$
- Words are conditionally independent given Y:



$$p(y, x_1, x_2, \cdots, x_d) = p(y) \prod_{i=1}^{d} p(x_i|y)$$

Example: Naïve Bayes for classification

- Classify e-mails as spam ($Y = 1$) or not spam ($Y = 0$)
  - Let $1{:}d$ index be the words in our vocabulary
  - $X_i = 1$ if word i appears in an e-mail, and 0 otherwise
  - E-mails are drawn according to some distribution $p(Y, X_1, X_2, \cdots, X_d)$
- Words are conditionally independent given Y. Then,

$$p(y, x_1, x_2, \cdots, x_d) = p(y) \prod_{i=1}^{d} p(x_i|y)$$

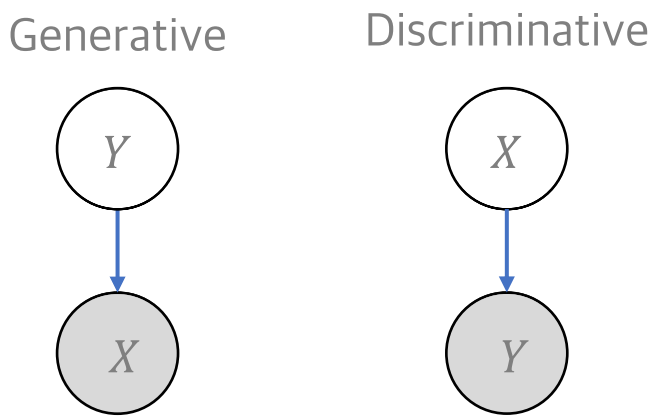
- Estimate parameters from training data. Predict with Bayes rule:

$$p(Y = 1|x_1, x_2, \cdots, x_d) = \frac{p(Y = 1) \prod_{i=1}^{d} p(x_i|Y = 1)}{\sum_{y=\{0,1\}} p(Y = y) \prod_{i=1}^{d} p(x_i|Y = y)}$$

- Are the independence assumptions made here reasonable?
- Philosophy: Nearly all probabilistic models are "wrong", but many are nonetheless useful

Discriminative vs generative models

- Using chain rule $p(Y, \boldsymbol{X}) = p(\boldsymbol{X}|Y)p(Y) = p(Y|\boldsymbol{X})p(\boldsymbol{X})$
- Corresponding Bayesian networks:

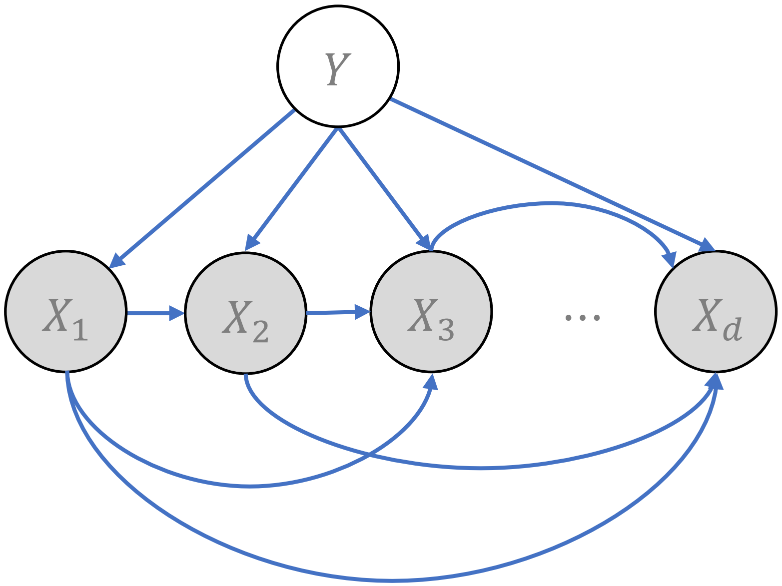


- All we need for prediction is $p(Y|\boldsymbol{X})$
- In the left model, we need to specify/learn both $p(Y)$ and $p(\boldsymbol{X}|Y)$, then compute $p(Y|\boldsymbol{X})$ via Bayes rule
- In the right model, it suffices to estimate just the conditional distribution $p(Y|\boldsymbol{X})$
  - We are not interested in $p(\boldsymbol{X})$!

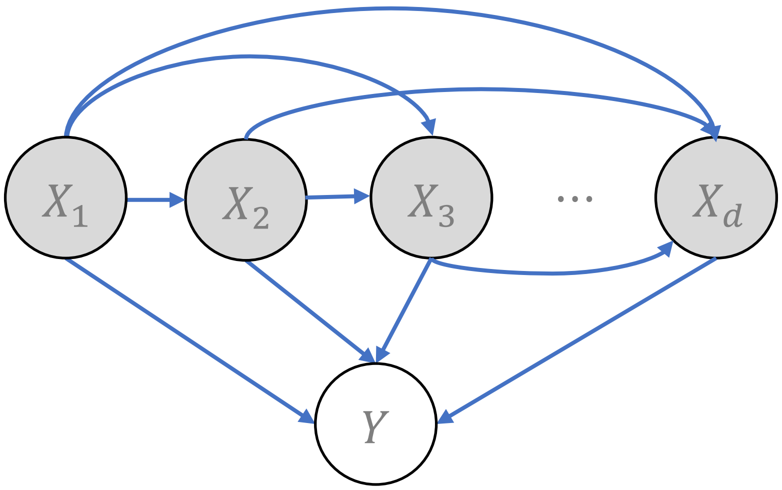
Discriminative vs generative models

- Since $\boldsymbol{X}$ is a random vector, chain rules will give
  - $p(Y,\boldsymbol{X}) = p(Y)p(X_1|Y)p(X_2|Y,X_1)\cdots p(X_d|Y,X_1,\cdots,X_{d-1})$
  - $p(Y,\boldsymbol{X}) = p(X_1)\,p(X_2|X_1)p(X_3|X_1,X_2)\cdots p(Y|X_1,\cdots,X_d)$

Generative



Discriminative



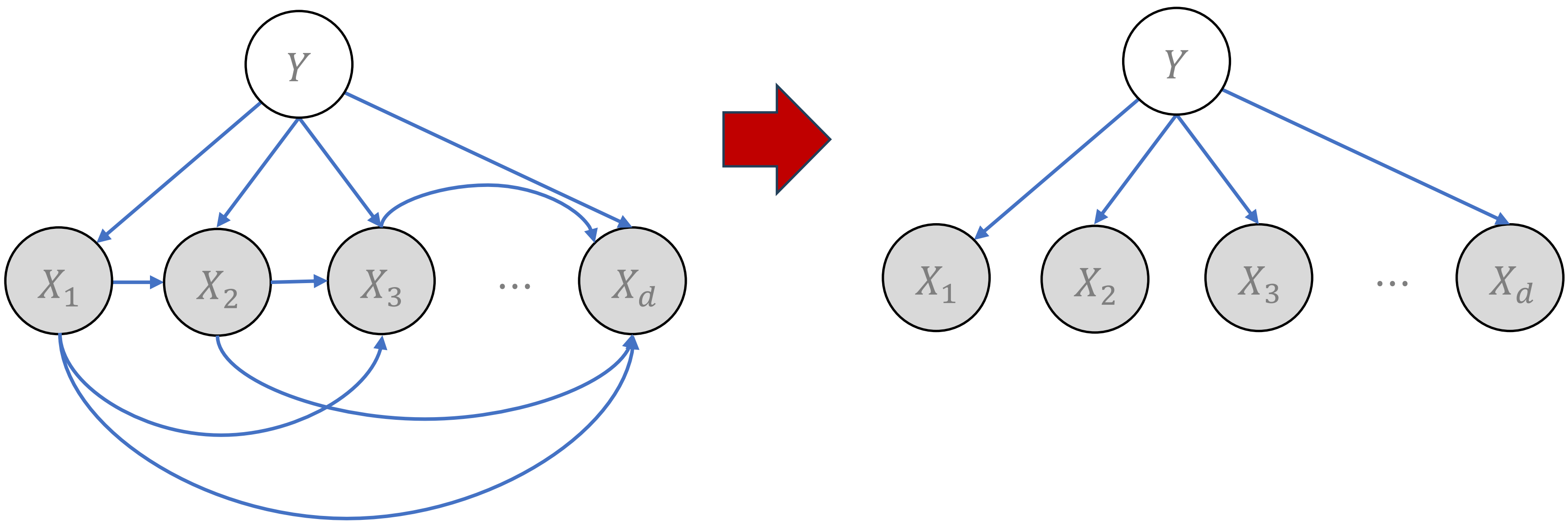
Discriminative vs generative models

- We must make the following choices:
  - **Generative model**: $p(Y)$ is simple, but how do we parametrize $p\left(X_i|\boldsymbol{X}_{pa(i)}, Y\right)$?
  - **Discriminative model**: how do we parametrize $p(Y|\boldsymbol{X})$? Here we assume we do not care about modeling $p(\boldsymbol{X})$ because $\boldsymbol{X}$ is always given to us in a classification problem

Naïve Bayes

- Assume that

$$X_i \perp \boldsymbol{X}_{-i} | Y$$



Logistic regression

- For the discriminative model, assume that

$$p(Y=1|\boldsymbol{x};\boldsymbol{\alpha}) = f(\boldsymbol{x},\boldsymbol{\alpha})$$

- where $\boldsymbol{x}^T = (x_1, \cdots, x_d)$, $\boldsymbol{\alpha}^T = (\alpha_0, \alpha_1, \cdots, \alpha_d)$
- It is a parametrized function of $\boldsymbol{x}$ (regression)
  - Has to be between 0 and 1
  - Depend in some simple but reasonable way on $x_1, \cdots, x_d$
  - Completely specified by a vector $\boldsymbol{\alpha}$ of $d+1$ parameters
- Linear dependence: let $z(\boldsymbol{\alpha},\boldsymbol{x}) \coloneqq \alpha_0 + \sum_{i=1}^{d} \alpha_i x_i$
- Then

$$p(Y=1|\boldsymbol{x};\boldsymbol{\alpha}) = \sigma(z(\boldsymbol{\alpha},\boldsymbol{x}))$$

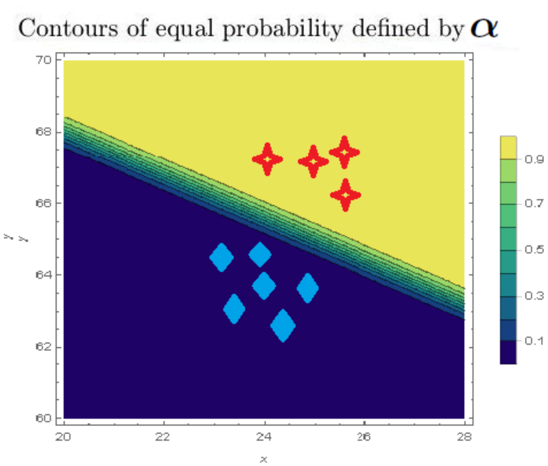
- where $\sigma(z) = 1/(1+e^{-z})$ is called the logistic(sigmoid) function

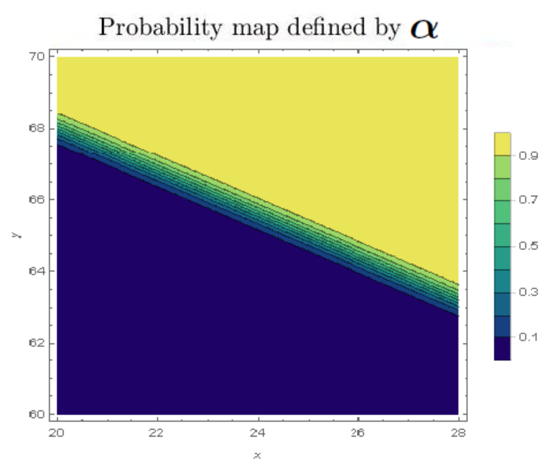
Logistic regression

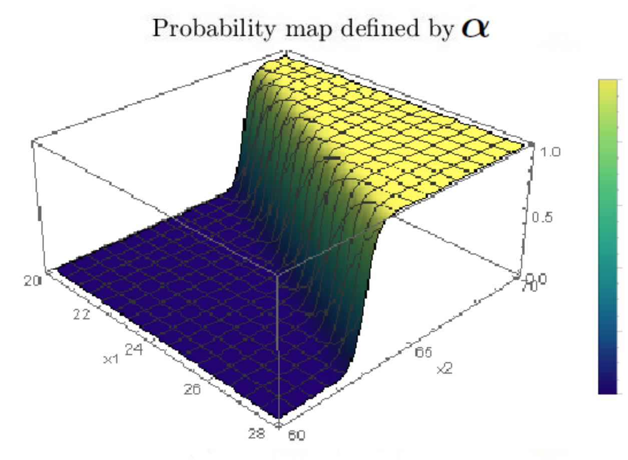
- Linear dependence: let $z(\boldsymbol{\alpha}, \boldsymbol{x}) \coloneqq \alpha_0 + \sum_{i=1}^{d} \alpha_i x_i$
- Then

$$p(Y = 1|\boldsymbol{x}; \boldsymbol{\alpha}) = \sigma(z(\boldsymbol{\alpha}, \boldsymbol{x}))$$

- where $\sigma(z) = 1/(1 + e^{-z})$ is called the logistic function







- Decision boundary $p(Y = 1|\boldsymbol{x}; \boldsymbol{\alpha})$ is linear in $\boldsymbol{x}$
- Equal probability contours are hyperplanes
- Probability rate of change has very specific form (third plot)

Remark: Logistic regression

- Losigtic model does not assume $X_i \perp \boldsymbol{X}_{-i} | Y$, unlike Naïve Bayes
- This can make a big difference in many applications
  - E.g., in spam classification, let $X_1 = 1$ ["bank" in e-mail] and $X_2 = 1$ ["account" in e-mail]
  - Regardless of whether spam, these always appear together, i.e., $X_1 = X_2$
  - Learning in Naïve Bayes results in $p(X_1|Y) = p(X_2|Y)$. Thus, Naïve Bayes double counts the evidence

Neural Models for discriminate models

- We 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 $\sigma(z) = 1/(1+e^{-z})$
  - Dependence might be too simple
- Non-linear dependence: let $\boldsymbol{h}(W,\boldsymbol{b},\boldsymbol{x}) = g(W\boldsymbol{x}+\boldsymbol{b})$ be a non-linear transformation of the inputs (features)

$$p_{neural}(Y=1|x;\boldsymbol{\alpha},W,\boldsymbol{b}) = \sigma\left(z\left(\boldsymbol{\alpha},\boldsymbol{h}(W,\boldsymbol{b},\boldsymbol{x})\right)\right)$$
$$= \sigma\left(\alpha_0 + \sum_{i=1}^{h} \alpha_i h_i\right)$$

  - More flexible and parameters: $\boldsymbol{\alpha}, W, \boldsymbol{b}$
  - Can repeat multiple times to get a neural network

Continuous random variables

- If X is a continuous random variable, we can usually represent it using its probability density function $p_X: \mathbb{R} \to \mathbb{R}^+$
- However, we cannot represent this function as a table anymore
- Typically consider parameterized densities:
  - Gaussian: $X = N(X|\mu, \sigma)$ if $p_X(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/2\sigma^2}$
  - Uniform: $X = U(X|a, b)$ if $p_X(x) = \frac{1}{b-a} 1_{[a \le x \le b]}$
- If X is a continuous random vector, we can usually represent it using its joint probability density function:
  - Gaussian: $p_X(\boldsymbol{x}) = \frac{1}{\sqrt{(2\pi)^d |\Sigma|}} \exp\left(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu})^T \Sigma^{-1} (\boldsymbol{x} - \boldsymbol{\mu})\right)$

Continuous random variables

- We can still use Bayesian networks with continuous (and discrete) variables
- Mixture of 2 Gaussian: Bayes net $Z \rightarrow X$ with factorization $p_{Z,X}(z,x) = p_Z(z)p_{X|Z}(x|z)$ and
  - $Z = Ber(Z|\mu)$
  - $p(X|Z=0) = N(X|\mu_0, \sigma_0)$, $X|(Z=1) = N(X|\mu_1, \sigma_1)$
  - The parameters are μ, μ_0, σ_0, μ_1, σ_1
- Variational autoencoder: Bayes net $Z \rightarrow X$ with factorization $p_{Z,X}(z,x) = p_Z(z)p_{X|Z}(x|z)$ and
  - $Z = N(Z|0, I)$
  - $p(X\big|Z=z) = N\left(X|\mu_\theta(z), e^{\sigma_\phi(z)}I\right)$ where μ_θ and σ_ϕ are neural networks with parameters (weights) θ, ϕ respectively

Thanks