Course Materials for GEN-AI

Northeastern University

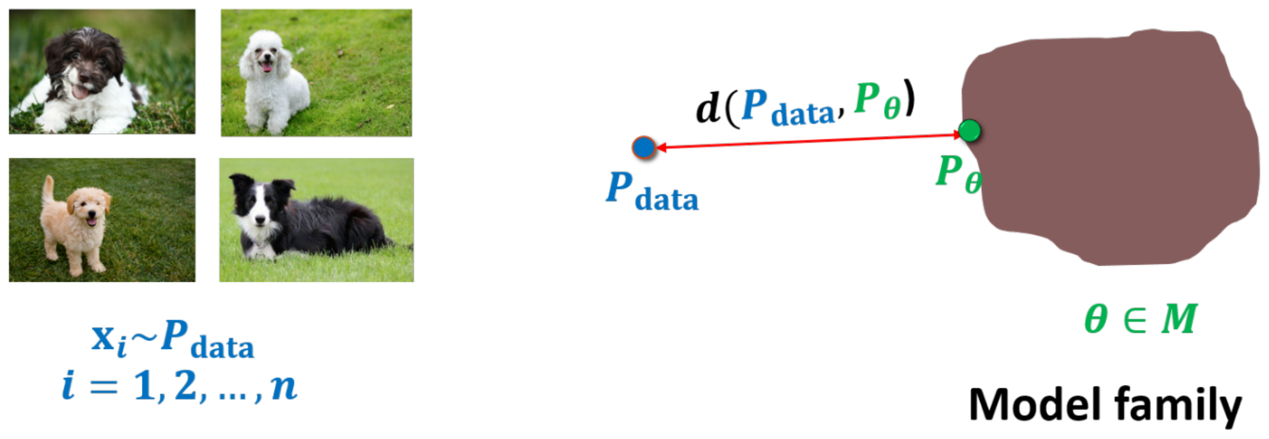
Instructor: Ramin Mohammadi
`r.mohammadi@northeastern.edu`

Thank you for your understanding and collaboration.

Representation

1 Learning a Generative Model

- We are given a training set of examples, e.g., images of dogs.



- We want to learn a probability distribution $p(x)$ over images x such that:
  - **Generation**: If we sample $x_{\text{new}} \sim p(x)$, x_{new} should look like a dog (sampling).
  - **Density Estimation**: $p(x)$ should be high if x looks like a dog, and low otherwise (anomaly detection).
  - **Unsupervised Representation Learning**: We should be able to learn what these images have in common, e.g., ears, tail, etc. (features).
- First question: how to represent $p(x)$?

Basic Discrete Distributions

Bernoulli Distribution (Biased Coin Flip)

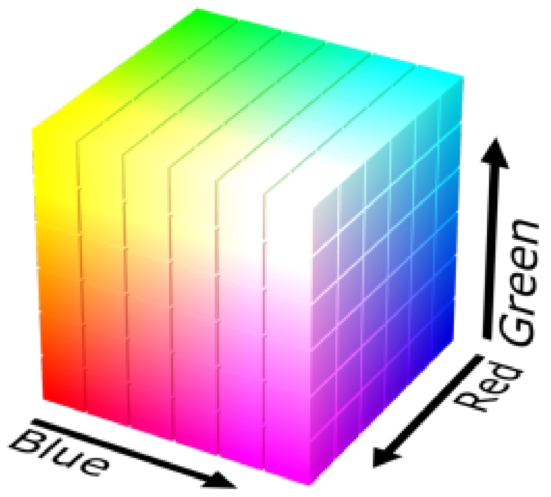
- $D = \{\text{Heads}, \text{Tails}\}$
- Specify $P(X = \text{Heads}) = p$, then $P(X = \text{Tails}) = 1 - p$.
- Write: $X \sim \text{Ber}(p)$.
- Sampling: flip a (biased) coin.

Categorical Distribution (Biased m-Sided Dice)

- $D = \{1, \ldots, m\}$.
- Specify $P(Y = i) = p_i$, such that $\sum p_i = 1$.
- Write: $Y \sim \text{Cat}(p_1, \ldots, p_m)$.
- Sampling: roll a (biased) die.

Example of Joint Distribution

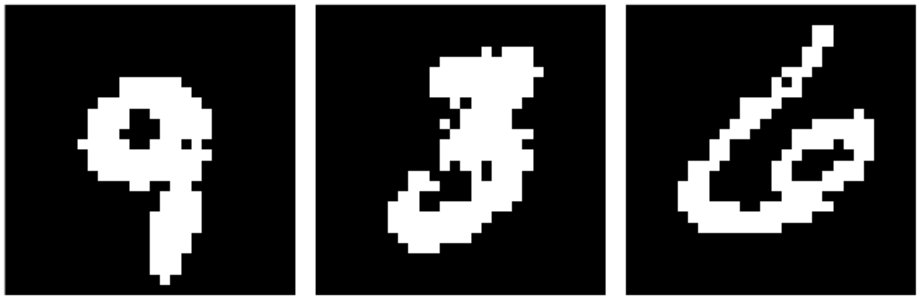
- Modeling a single pixel's color. Three discrete random variables:
  - Red Channel R: $\text{Val}(R) = \{0, \ldots, 255\}$.
  - Green Channel G: $\text{Val}(G) = \{0, \ldots, 255\}$.
  - Blue Channel B: $\text{Val}(B) = \{0, \ldots, 255\}$.



- 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 $p(R = r, G = g, B = b)$?

$$256 \times 256 \times 256 - 1$$

Example of Joint Distribution (Binary Pixels)



- Suppose $X_1, \dots, X_n$ are binary (Bernoulli) random variables, i.e., $\mathrm{Val}(X_i) = \{0, 1\} = \{\text{Black, White}\}$.
- How many possible states? n = number of pixels

$$2 \times 2 \times \cdots \times 2 = 2^n$$

- Sampling from $p(x_1, \dots, x_n)$ generates an image.
- How many parameters to specify the joint distribution $p(x_1, \dots, x_n)$ over n binary pixels?

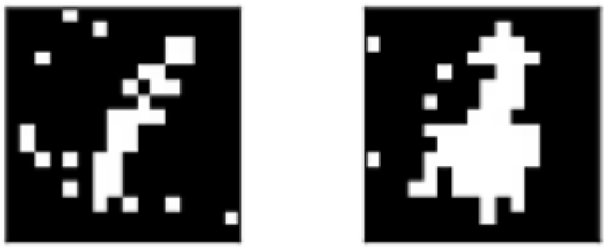
$$2^n - 1$$

Structure Through Independence

- If $X_1, \dots, X_n$ are independent, then:

$$p(x_1, \dots, x_n) = p(x_1)p(x_2)\cdots p(x_n)$$

- How many possible states? 2^n.
- How many parameters to specify the joint distribution $p(x_1, \dots, x_n)$?
- How many to specify the marginal distribution $p(x_1)$? 1.
- 2^n entries can be described by just n numbers (if $|\mathrm{Val}(X_i)| = 2$).
- Independence assumption is too strong. Model not likely to be useful:
  - For example, each pixel is chosen independently when we sample from it.



Key Notion - Conditional Independence

- Two events A, B are conditionally independent given event C if:

$$p(A \cap B \mid C) = p(A \mid C)p(B \mid C)$$

- Random variables X, Y are conditionally independent given Z if for all values $x \in \mathrm{Val}(X), y \in \mathrm{Val}(Y), z \in \mathrm{Val}(Z)$:

$$p(X = x \cap Y = y \mid Z = z) = p(X = x \mid Z = z)p(Y = y \mid Z = z)$$

- We also write:

$$p(X, Y \mid Z) = p(X \mid Z)p(Y \mid Z)$$

- Equivalent definition:

$$p(X \mid Y, Z) = p(X \mid Z)$$

- Notation: $X \perp Y \mid Z$.

Two Important Rules

1. **Chain Rule:** Let $S_1, \dots, S_n$ be events, $p(S_i) > 0$.

$$p(S_1 \cap S_2 \cap \cdots \cap S_n) = p(S_1)p(S_2 \mid S_1)p(S_3 \mid S_1 \cap S_2) \cdots p(S_n \mid S_1 \cap \cdots \cap S_{n-1})$$

2. **Bayes' Rule:** Let S_1, S_2 be events, $p(S_1) > 0$ and $p(S_2) > 0$.

$$p(S_1 \mid S_2) = \frac{p(S_1 \cap S_2)}{p(S_2)} = \frac{p(S_2 \mid S_1)p(S_1)}{p(S_2)}$$

Structure Through Conditional Independence

- Using the Chain Rule:

$$p(x_1, \dots, x_n) = p(x_1)p(x_2 \mid x_1)p(x_3 \mid x_1, x_2) \cdots p(x_n \mid x_1, \dots, x_{n-1})$$

- How many parameters? $1 + 2 + \cdots + 2^{n-1} = 2^n - 1$.
  - $p(x_1)$ requires 1 parameter.
  - $p(x_2 \mid x_1 = 0)$ requires 1 parameter, $p(x_2 \mid x_1 = 1)$ requires 1 parameter.
  - Total: $2^n - 1$.
- Now suppose $X_{i+1} \perp X_1, \dots, X_{i-1} \mid X_i$, then:

$$p(x_1, \dots, x_n) = p(x_1)p(x_2 \mid x_1)p(x_3 \mid x_2) \cdots p(x_n \mid x_{n-1})$$

- How many parameters? Exponential reduction: $2n - 1$.

Example

- Suppose we have 4 random variables $X_1, \dots, X_4$.
- Using the Chain Rule, we can always write:

$$p(x_1, \dots, x_4) = p(x_1)p(x_2 \mid x_1)p(x_3 \mid x_1, x_2)p(x_4 \mid x_1, x_2, x_3)$$

- If $X_4 \perp X_2 \mid \{X_1, X_3\}$, we can simplify as:

$$p(x_1, \dots, x_4) = p(x_1)p(x_2 \mid x_1)p(x_3 \mid x_1, x_2)p(x_4 \mid x_1, x_3)$$

- Bayesian Networks: Assume an ordering and a set of conditional independencies to get compact representation.

Bayesian Networks

Bayes Network - General Idea

- Use conditional parameterization (instead of joint parameterization).
- For each random variable X_i, specify $p(x_i \mid x_{A_i})$ for set X_{A_i} of random variables.
- Then get joint parameterization as:

$$p(x_1, \dots, x_n) = \prod_i p(x_i \mid x_{A_i})$$

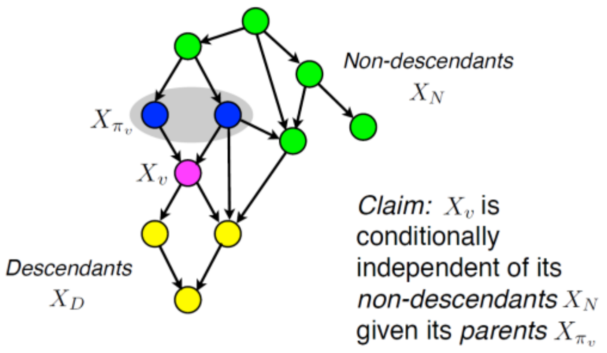
- Need to guarantee it is a legal probability distribution. It has to correspond to a chain rule factorization, with factors simplified due to assumed conditional independencies.

Bayesian Network Conditional Independence

A Bayesian network structure G implies the following conditional independence assumptions:

- For each variable X_v, X_v is independent from its non-descendants given its parents.

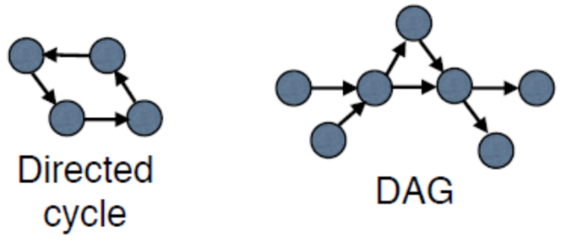
$$X_v \perp \text{NonDescendants}_{X_v} \mid \text{Parents}_{X_v}$$


Non-descendants
X_N
X_{π_v}
X_v
Descendants
X_D
Claim: X_v is conditionally independent of its *non-descendants* X_N given its *parents* X_{π_v}

This configuration is referred to as *local dependencies* in the context of Bayesian networks.

- A Bayesian network is specified by a directed acyclic graph $G = (V, E)$ with:
  1. One node $i \in V$ for each random variable X_i.
  2. One conditional probability distribution (CPD) per node, $p(x_i \mid x_{\mathrm{Pa}(i)})$, specifying the variable's probability conditioned on its parents' values.
- The graph $G = (V, E)$ is called the structure of the Bayesian Network.
- Defines a joint distribution:
$$p(x_1, \dots, x_n) = \prod_{iinV} p(x_i \mid x_{\mathrm{Pa}(i)})$$
- Claim: $p(x_1, \dots, x_n)$ is a valid probability distribution because of ordering implied by DAG.
- Economical representation: Exponential in $|\mathrm{Pa}(i)|$, not $|V|$.

Example - Directed Acyclic Graph



DAG stands for Directed Acyclic Graph

- A Directed Acyclic Graph (DAG) represents the structure of a Bayesian Network.

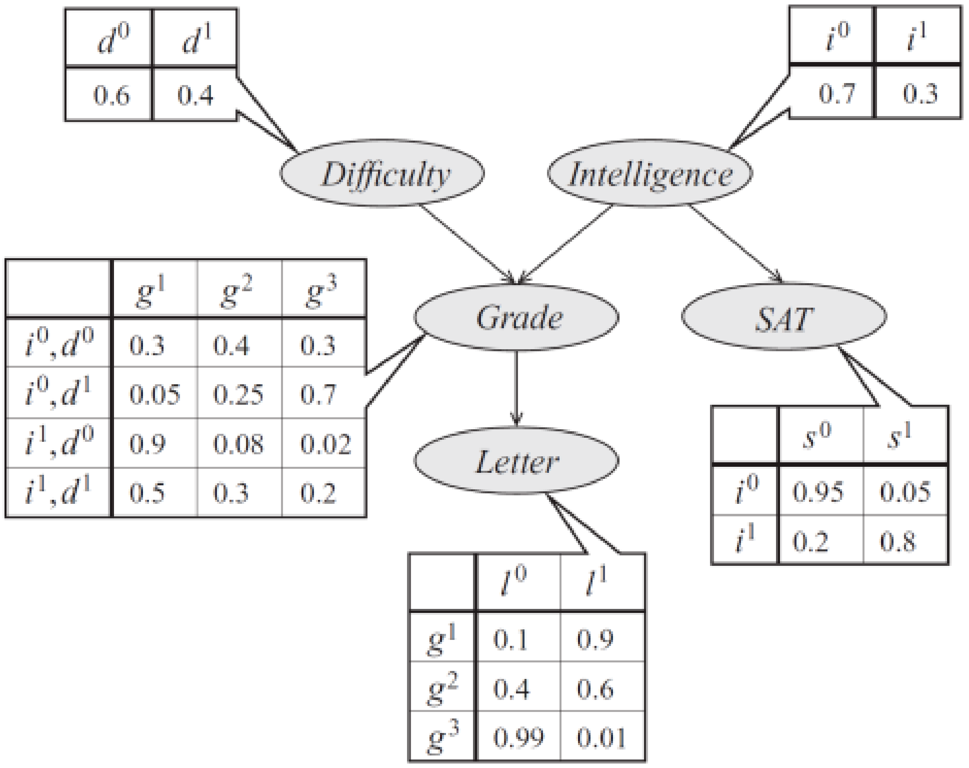
- Cycles are not allowed.

Example Bayesian Network

- Consider the following Bayesian network:

$$p(d, i, g, s, l) = p(d)p(i)p(g \mid i, d)p(s \mid i)p(l \mid g)$$

- What is its joint distribution? The graph defines a factorization.



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

$$p(x_1, \dots, x_n) = \prod_{iinV} p(x_i \mid x_{\text{Pa}(i)})$$

Basically, we represent the join distribution by calculating how these random variables are related to each other locally with respect to this graph (tables). For example, you only need to know how to assign grades given different values for Difficulty and Intelligence.
By making this assumption that global dependencies can be broken down in terms of simpler local dependencies. This way we can get some benefits as conditional probabilities are potentially much smaller and simpler to represent.

And the Idea here is that assuming the above factorization is the same as assuming conditional independencies.

- Factorization:

$$p(d, i, g, s, l) = p(d)p(i)p(g \mid i, d)p(s \mid i)p(l \mid g)$$

- However, by the Chain Rule, any distribution can be written as:

$$p(d, i, g, s, l) = p(d)p(i \mid d)p(g \mid i, d)p(s \mid i, d, g)p(l \mid g, d, i, s)$$

- Thus, we are assuming the following additional independencies:

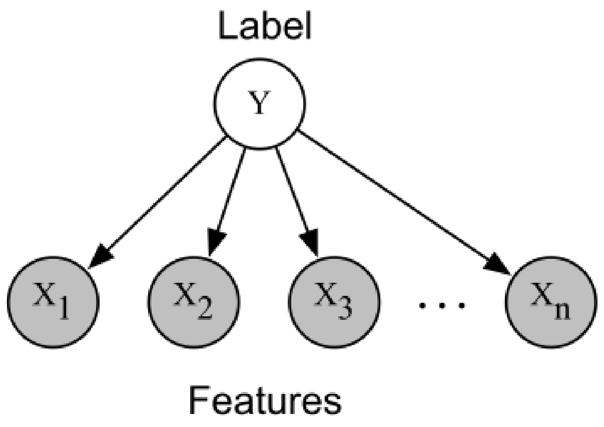
$$D \perp I, \quad S \perp \{D, G\} \mid I, \quad L \perp \{I, D, S\} \mid G$$

1.1 Summary for Bayesian Networks

- 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 identify some conditional independence properties by looking at graph properties

Naive Bayes for Single Label Prediction

- Example: Classify emails as spam ($Y = 1$) or not spam ($Y = 0$).
- Let $1 : n$ index the words in our vocabulary (e.g., English).
- $X_i = 1$ if word i appears in an email, and 0 otherwise.
- Emails are drawn according to some distribution $p(Y, X_1, \ldots, X_n)$.
- Words are conditionally independent given Y:



$$p(y, x_1, \ldots, x_n) = p(y) \prod_{i=1}^{n} p(x_i \mid y)$$

Example - Naive Bayes for Classification

- Example: Classify emails as spam ($Y = 1$) or not spam ($Y = 0$).
- Let $1 : n$ index the words in our vocabulary (e.g., English).
- $X_i = 1$ if word i appears in an email, and 0 otherwise.
- Emails are drawn according to some distribution $p(Y, X_1, \ldots, X_n)$.
- Suppose that the words are conditionally independent given Y. Then:

$$p(y, x_1, \ldots, x_n) = p(y) \prod_{i=1}^{n} p(x_i \mid y)$$

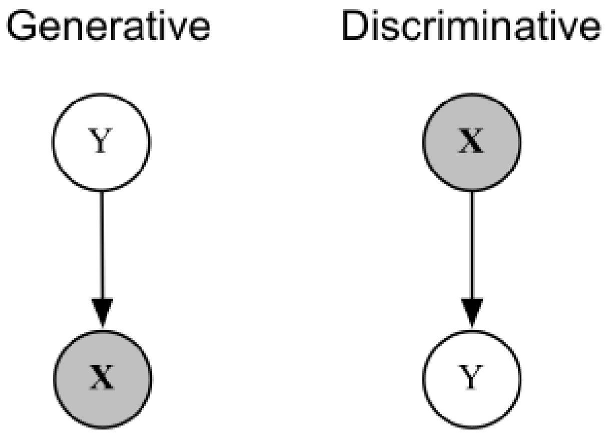
- Estimate parameters from training data. Predict with Bayes' Rule:

$$p(Y = 1 \mid x_1, \ldots, x_n) = \frac{p(Y = 1) \prod_{i=1}^{n} p(x_i \mid Y = 1)}{\sum_{y \in \{0,1\}} p(Y = y) \prod_{i=1}^{n} p(x_i \mid Y = y)}$$

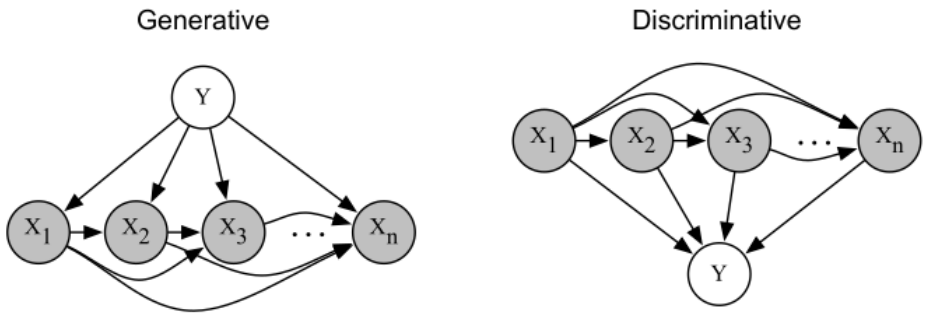
- Are the independence assumptions made here reasonable?
- Philosophy: Nearly all probabilistic models are "wrong," but many are nonetheless useful.

Discriminative versus Generative Models

- Using the chain rule:
$$p(Y, X) = p(X \mid Y)p(Y) = p(Y \mid X)p(X)$$
- Corresponding Bayesian networks:



  - Generative model: Learn $p(X \mid Y)$ and $p(Y)$.
  - Discriminative model: Learn $p(Y \mid X)$.
- If all we need for prediction is $p(Y \mid X)$:
  - In the generative model, we need to specify/learn both $p(Y)$ and $p(X \mid Y)$, then compute $p(Y \mid X)$ via Bayes' rule.
  - In the discriminative model, it suffices to estimate just $p(Y \mid X)$.
- A discriminative model is only useful for discriminating Y's label when given X.
- Since X is a random vector, chain rules give:
$$p(Y, X) = p(Y)p(X_1 \mid Y)p(X_2 \mid Y, X_1) \cdots p(X_n \mid Y, X_1, \ldots, X_{n-1})$$
$$p(Y, X) = p(X_1)p(X_2 \mid X_1)p(X_3 \mid X_1, X_2) \cdots p(Y \mid X_1, \ldots, X_{n-1}, X_n)$$

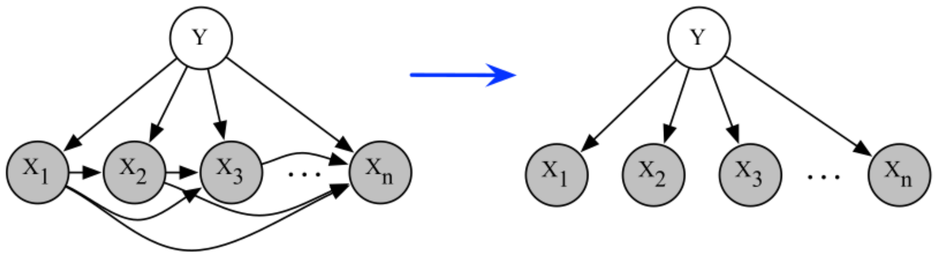


- Key choices:
  1. In the generative model: $p(Y)$ is simple, but how do we parameterize $p(X_i \mid X_{\text{pa}(i)}, Y)$?
  2. In the discriminative model: How do we parameterize $p(Y \mid X)$?

Naive Bayes

- In the generative model, assume that $X_i \perp X_{-i} \mid Y$ (Naive Bayes assumption).

$$p(Y, X_1, \ldots, X_n) = p(Y) \prod_{i=1}^{n} p(X_i \mid Y)$$



Generative Models Are Still Useful

- Using chain rule:

$$p(Y, X) = p(X \mid Y)p(Y) = p(Y \mid X)p(X)$$

- Generative models allow marginalizing over unobserved variables.
- If some X_i variables are unobserved, we can still compute $p(Y \mid X_{\text{evidence}})$ by marginalizing over unseen variables.

Neural Models

In discriminative models (ex: logistic regression), we assume:

- $$p(Y = 1 \mid x; \theta) = f(x, \theta)$$
- Linear dependence:

$$z(\theta, x) = \theta_0 + \sum_{i=1}^{n} \theta_i x_i$$

$$p(Y = 1 \mid x; \theta) = \sigma(z(\theta, x)), \quad \sigma(z) = \frac{1}{1 + e^{-z}}$$

- Dependence might be too simple.

we can do things more flexible by assuming some nonlinear dependence and that is were we use NNs.

- we can apply linear / nonlinear transformation or stack them however we wants

$$h(A, b, x) = f(Ax + b)$$

$$p_{\text{Neural}}(Y = 1 \mid x; \theta, A, b) = \sigma\left(\theta_0 + \sum_{i=1}^{h} \theta_i h_i\right)$$

- More flexible, but with more parameters: A, b, θ.
- We will learn a richer set of dependencies this way.

once we learned how to predict one thing we can repeat to predict other components and that is the idea of autoregressive models, for example we can generate a high dimensional output (ex: image) as a number of individual component and chain rule gives you a way of predicting individual component given the previous one and that is how you build a neural autoregressive model

Bayesian Networks vs Neural Models

- Chain rule general idea (Fully general):

$$p(x_1, x_2, x_3, x_4) = p(x_1)p(x_2 \mid x_1)p(x_3 \mid x_1, x_2)p(x_4 \mid x_1, x_2, x_3)$$

- Bayesian Network assumes conditional independencies:

$$p(x_1, x_2, x_3, x_4) \approx p(x_1)p(x_2 \mid x_1)p(x_3 \mid x_2)p(x_4 \mid x_1)$$

  we do this by assuming conditional independence. example above we assumed x_4 only depends on x_1 which is a strong assumption and does not work in higher dimensions like images.

- Neural models assume specific functional forms for the conditionals:

$$p(x_1, x_2, x_3, x_4) \approx p(x_1)p(x_2 \mid x_1)p_{\text{Neural}}(x_3 \mid x_1, x_2)p_{\text{Neural}}(x_4 \mid x_1, x_2, x_3)$$

  This way we let the model to learn all these conditional dependencies instead of making the strong assumptions. (This assumption that this conditional relationship can be captured by NNs itself might or might not be the case in practice).

- Using NNs is one way to get tractability to the extend that this NNs are not to big and somehow you are able to tie them together.
- A sufficiently deep neural net can approximate any function.

Continuous 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: $X \sim \mathcal{N}(\mu, \sigma)$ if $p_X(x) = \frac{1}{\sigma\sqrt{2\pi}}e^{-(x-\mu)^2/2\sigma^2}$
  - Uniform: $X \sim \mathcal{U}(a, b)$ if $p_X(x) = \frac{1}{b-a}\mathbf{1}_{[a \le x \le b]}$
  - Etc.
- If X is a continuous random vector, we can usually represent it using its **joint probability density function**:
  - Gaussian: $X \sim \mathcal{N}(\mu, \Sigma)$ if

$$p_X(x) = \frac{1}{\sqrt{(2\pi)^k|\Sigma|}} \exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right)$$

- Chain rule, Bayes rule, etc all still apply. For example,

$$p_{X,Y,Z}(x, y, z) = p_X(x)p_{Y|X}(y \mid x)p_{Z|X,Y}(z \mid x, y)$$

Continuous variables

- This means we can still use Bayesian networks with continuous (and discrete) variables or combination of NNs and Bayesian models. Examples:
  - **Mixture of 2 Gaussian**: Using Bayesian Network for two random variables $Z \to X$ (Z is parent to X) with factorization
  $$p_{Z,X}(z,x) = p_Z(z)p_{X|Z}(x \mid z)$$
    - $Z \sim \text{Bernoulli}(p)$
    - $X \mid (Z=0) \sim \mathcal{N}(\mu_0, \sigma_0), X \mid (Z=1) \sim \mathcal{N}(\mu_1, \sigma_1)$
    - The parameters are $\mu_0, \sigma_0, \mu_1, \sigma_1$
  - Or we could have a Network $Z \to X$ with factorization $p_{Z,X}(z,x) = p_Z(z)p_{X|Z}(x \mid z)$
    - $Z \sim \mathcal{U}(a,b)$
    - $X \mid (Z=z) \sim \mathcal{N}(z, \sigma)$
    - The parameters are a, b, σ
  - **Variational autoencoder**: Network $Z \to X$ with factorization
  $$p_{Z,X}(z,x) = p_Z(z)p_{X|Z}(x \mid z)$$
    - $Z \sim \mathcal{N}(0,1)$
    - $X \mid (Z=z) \sim \mathcal{N}(\mu_\theta(z), e^{\sigma_\phi(z)})$ where $\mu_\theta : \mathbb{R} \to \mathbb{R}$ and σ_ϕ are neural networks with parameters (weights) θ, ϕ respectively
    - Even if μ_θ, σ_ϕ are very deep (flexible), functional form is still Gaussian

References

[1] Stefano Ermon. *CS236*