CS 236: Deep Generative Models

Stefano Ermon

Stanford University

URL: deepgenerativemodels.github.io

Introduction

Challenge: understand complex, unstructured inputs

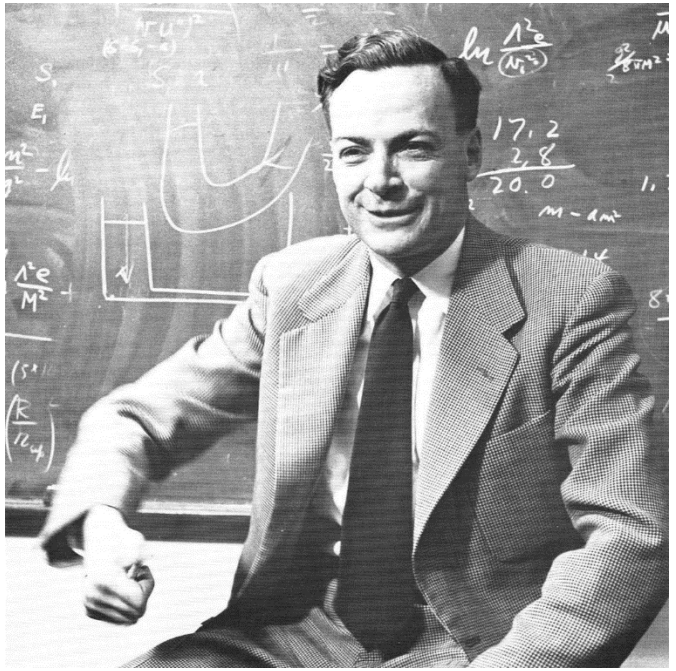
Computer Vision

Computational Speech

Natural Language Processing

Robotics

Introduction



Richard Feynman: "*What I cannot create, I do not understand*"

Generative modeling: "*What I understand, I can* ***create***"

Generative Modeling: Computer Graphics

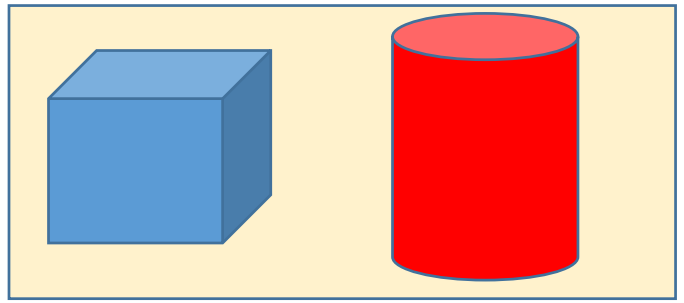
How to generate natural images with a computer?

High level description

Cube(color=blue, position=(x,y,z), size=…)
Cylinder(color=red, position=(x',y',z'), size=..)

Generation (graphics)

Inference (vision as inverse graphics)



Raw sensory outputs

Many of our models will have **similar structure (generation + inference)**

Statistical Generative Models

Statistical generative models are **learned from data**

Data
(e.g., images of bedrooms)

+

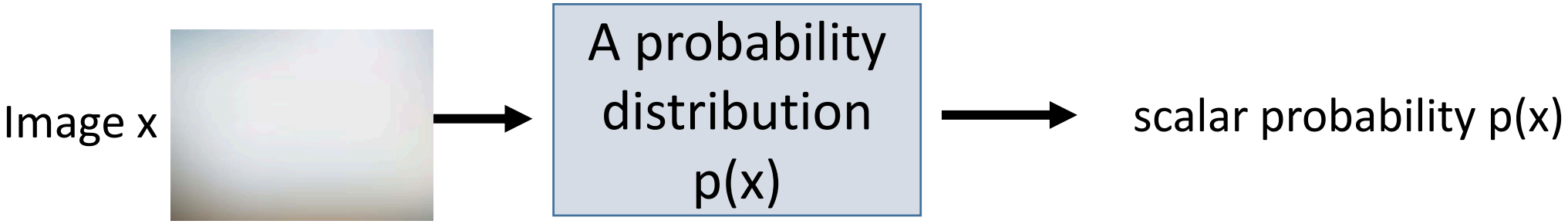
Prior Knowledge
(e.g., physics, materials, ..)

Priors are always necessary, but there is a spectrum

Statistical Generative Models

A statistical generative model is a **probability distribution** p(x)

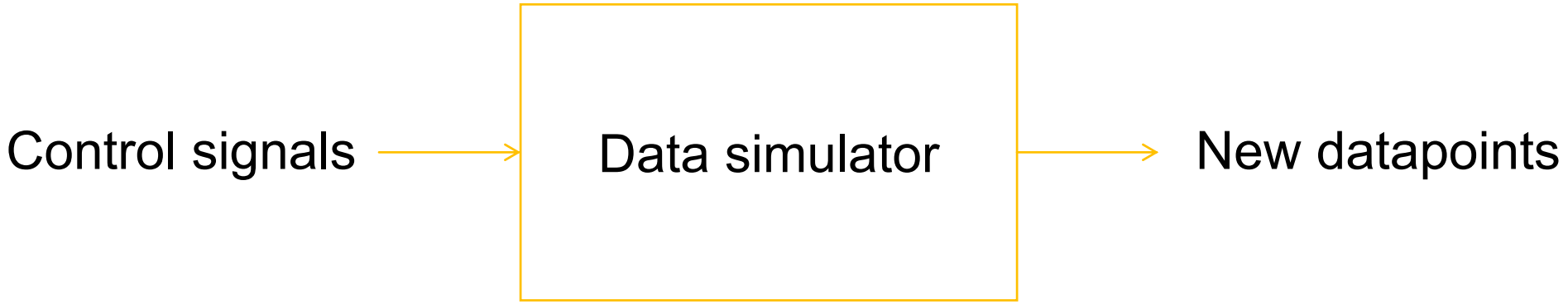
- **Data:** samples (e.g., images of bedrooms)
- **Prior knowledge:** parametric form (e.g., Gaussian?), loss function (e.g., maximum likelihood?), optimization algorithm, etc.



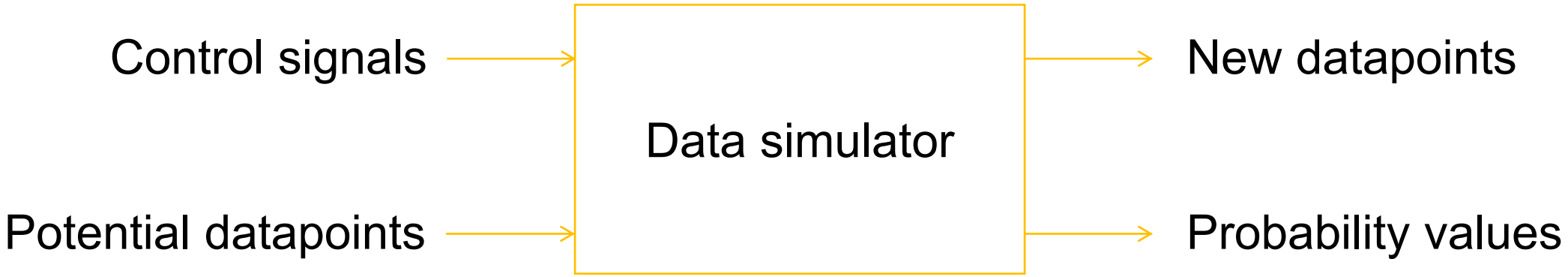
It is generative because **sampling from p(x) generates new images**

Building a simulator for the data generating process

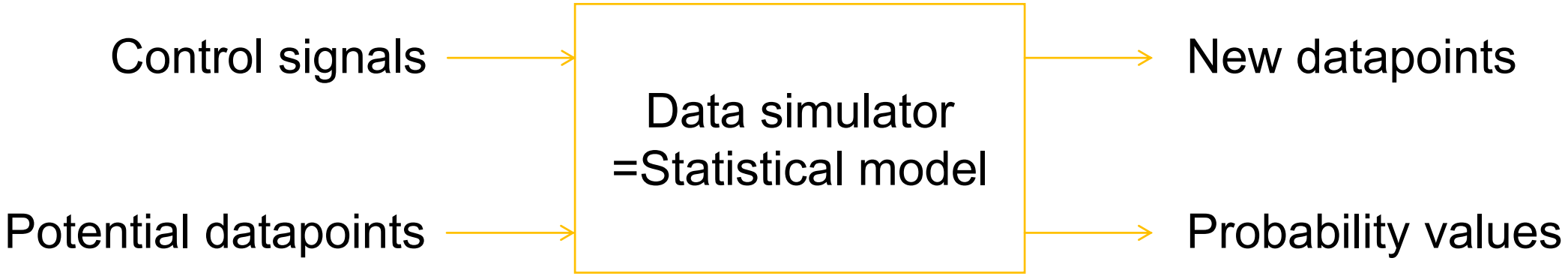
Building a simulator for the data generating process



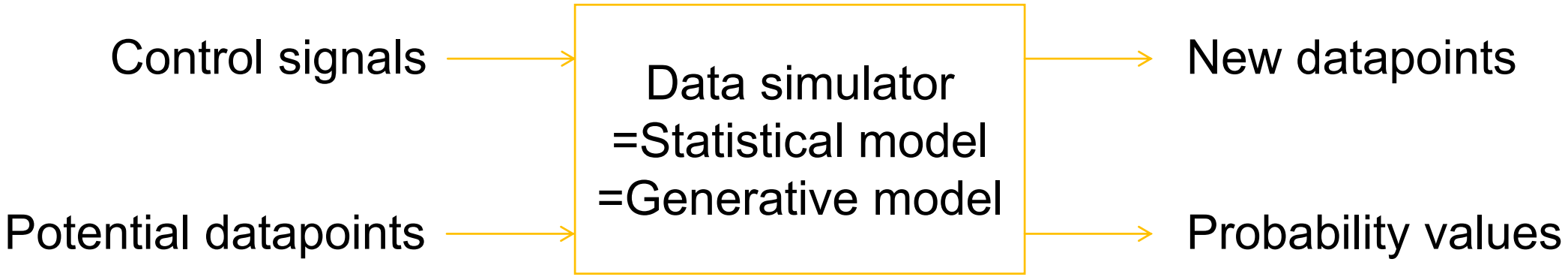
Building a simulator for the data generating process



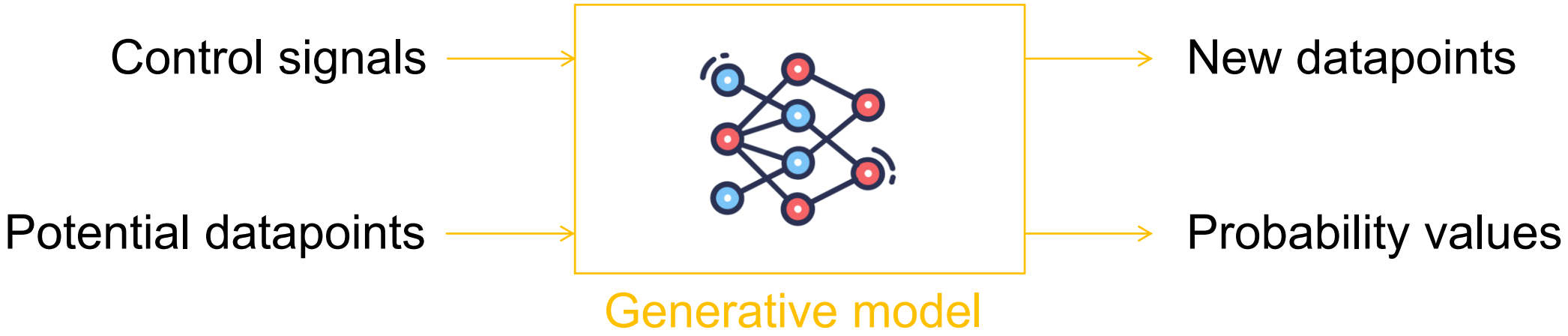
Building a simulator for the data generating process



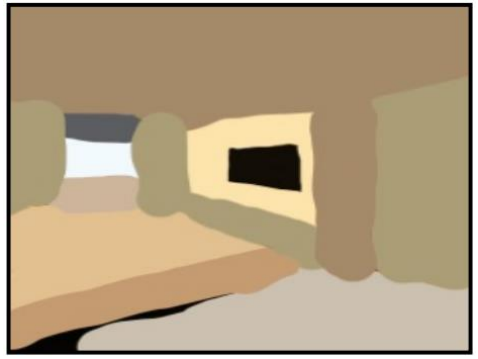
Building a simulator for the data generating process



Building a simulator for the data generating process



Data generation in the real world



Generate

Generative model
of realistic images

Stroke paintings to realistic images

[Meng, He, Song, et al., ICLR 2022]

"Ace of Pentacles"

Generate

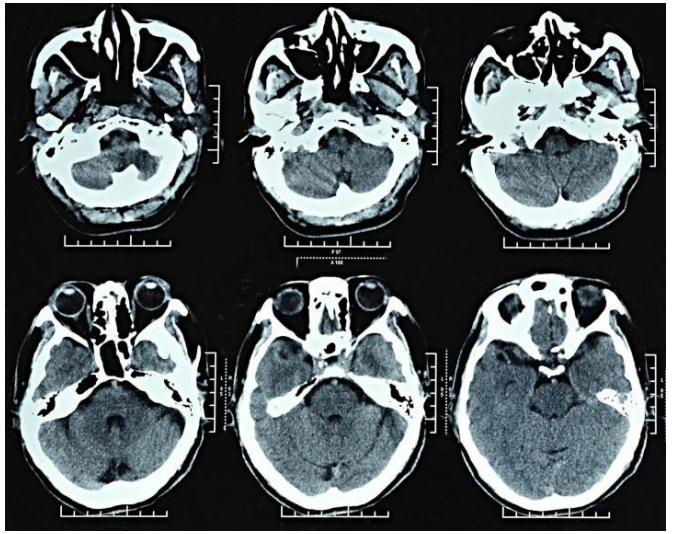
Generative model
of paintings

Language-guided artwork creation

https://chainbreakers.kath.io @RiversHaveWings

Solving inverse problems with generative models

Generate



Generative model
of medical images

Medical image reconstruction

[Song et al., ICLR 2022]

Outlier detection with generative models

High probability

Low probability

Generative model of traffic signs

Outlier detection

[Song et al., ICLR 2018]

Progress in Generative Models of Images -- GANs

Ian Goodfellow, 2019

Progress in Generative Models of Images – Diffusion Models

Song et al., Score-Based Generative Modeling through Stochastic Differential Equations, 2021

Text2Image Diffusion Models

User input:

An astronaut riding a horse

Text2Image Diffusion Models

User input:

A perfect Italian meal

Text2Image Diffusion Models

User input:

泰迪熊穿着戏服，站在太和殿前唱京剧

A teddy bear, wearing a costume, is standing in front of the Hall of Supreme Harmony and singing Beijing opera

Dalle3

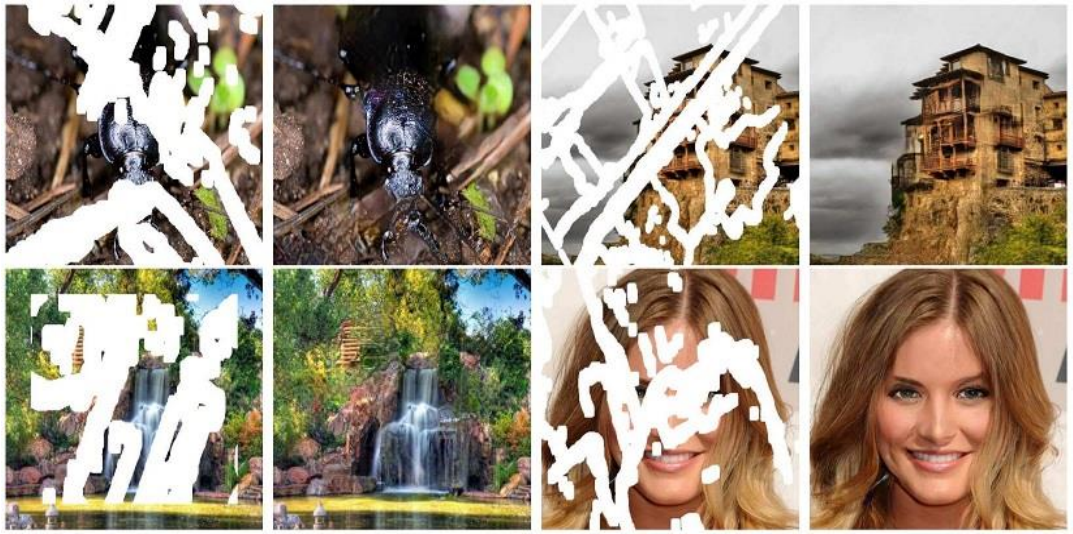
A minimap diorama of a cafe adorned with indoor plants. Wooden beams crisscross above, and a cold brew station stands out with tiny bottles and glasses

Progress in Inverse Problems

P(high resolution | low resolution)

Menon et al, 2020

P(full image| mask)



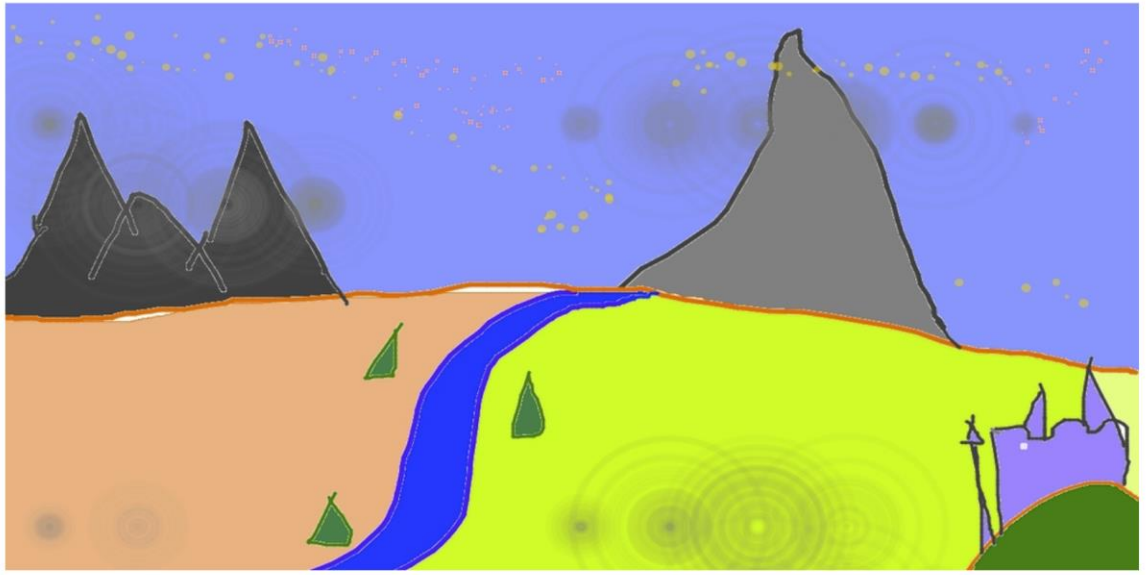
Liu al, 2018

P(color image| greyscale)

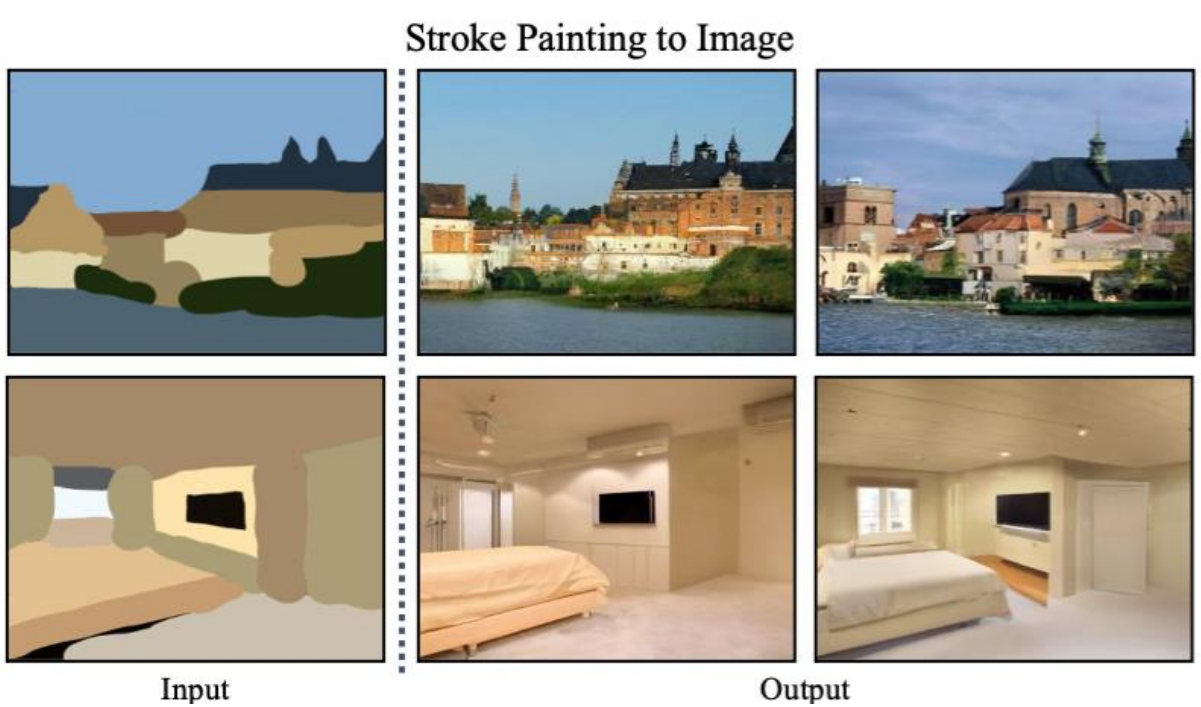
Antic, 2020

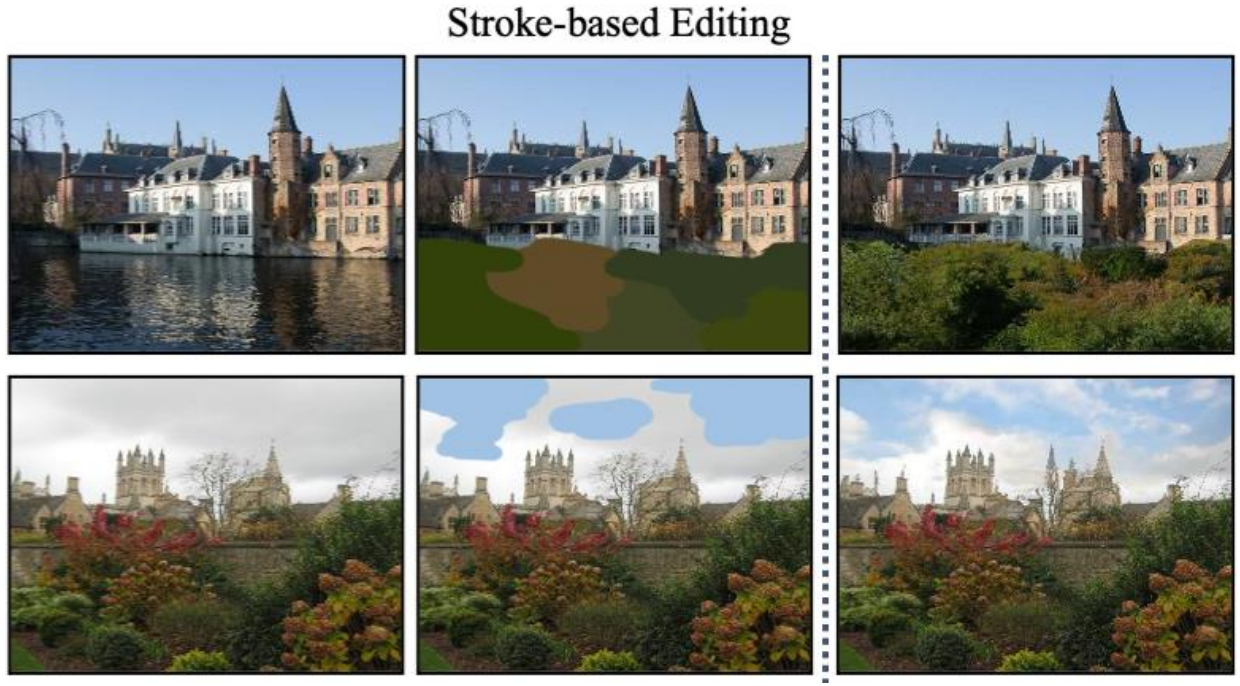
Progress in Inverse Problems

User input:

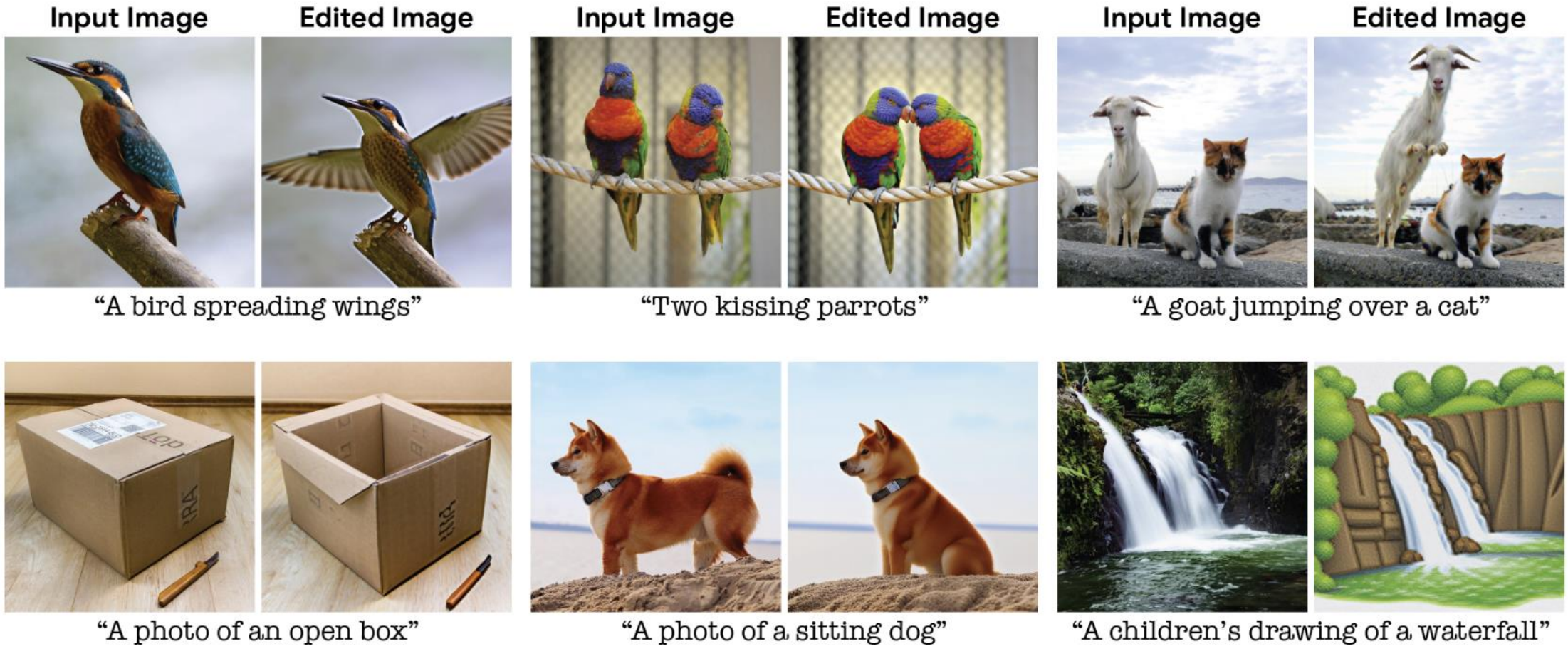


Progress in Inverse Problems



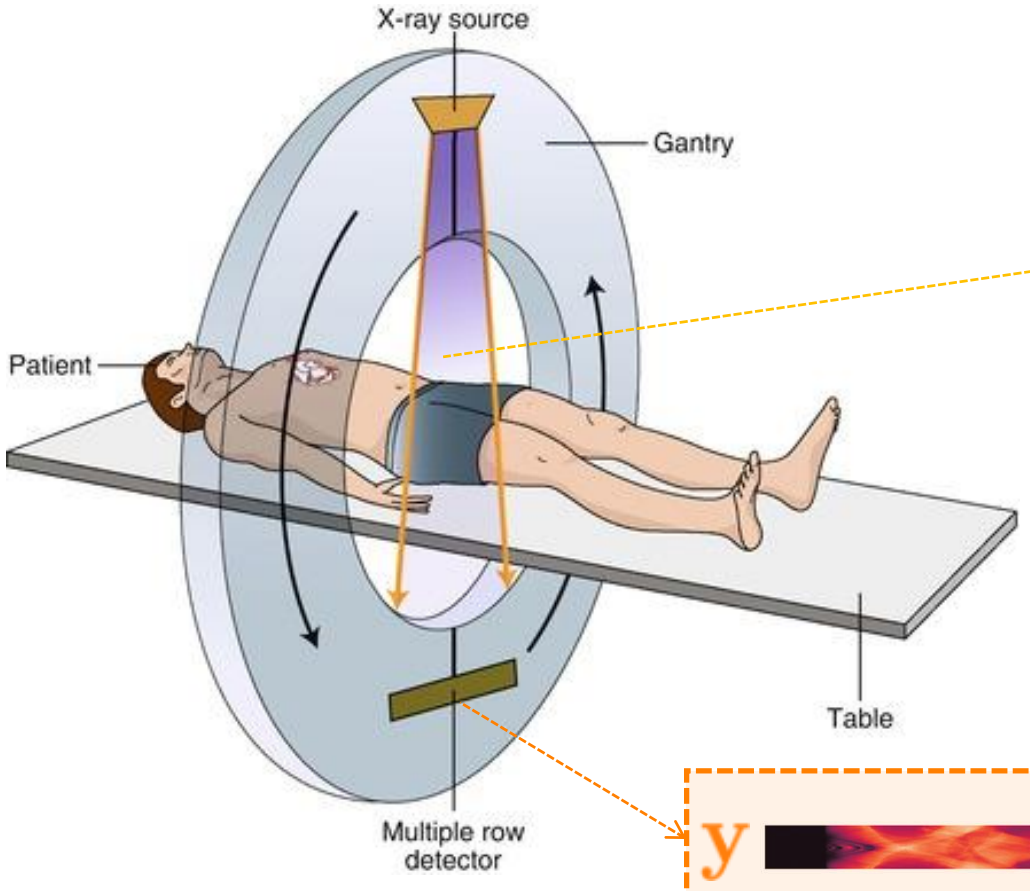


Progress in Inverse Problems



Kawar et al., 2023

Medical image reconstruction



Cross-sectional image

$\mathbf{x}$

$\mathbf{y}$

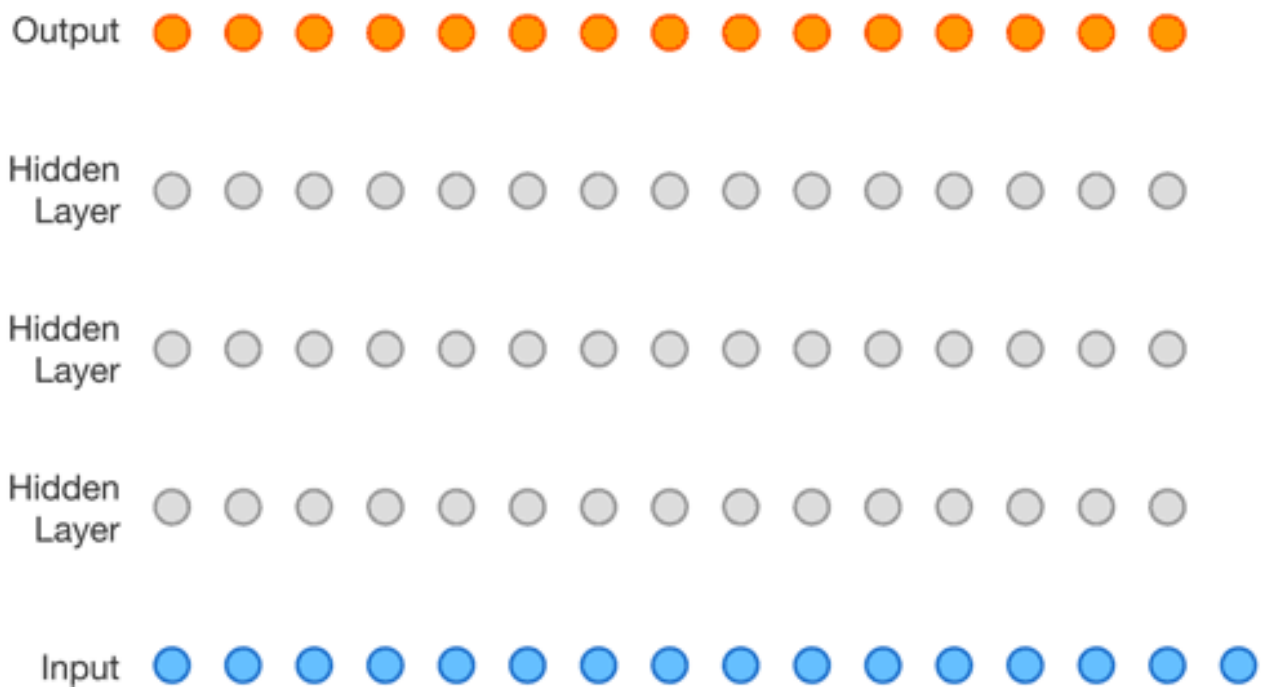
Sparse-view sinogram

Sparse-view computed tomography (CT)

Forward model $p(\mathbf{y} \mid \mathbf{x})$ is given by physical simulation

WaveNet

Generative model of speech signals



Text to Speech

- Parametric
- Concatenative
- WaveNet

- Unconditional
- Music

van den Oord et al, 2016c

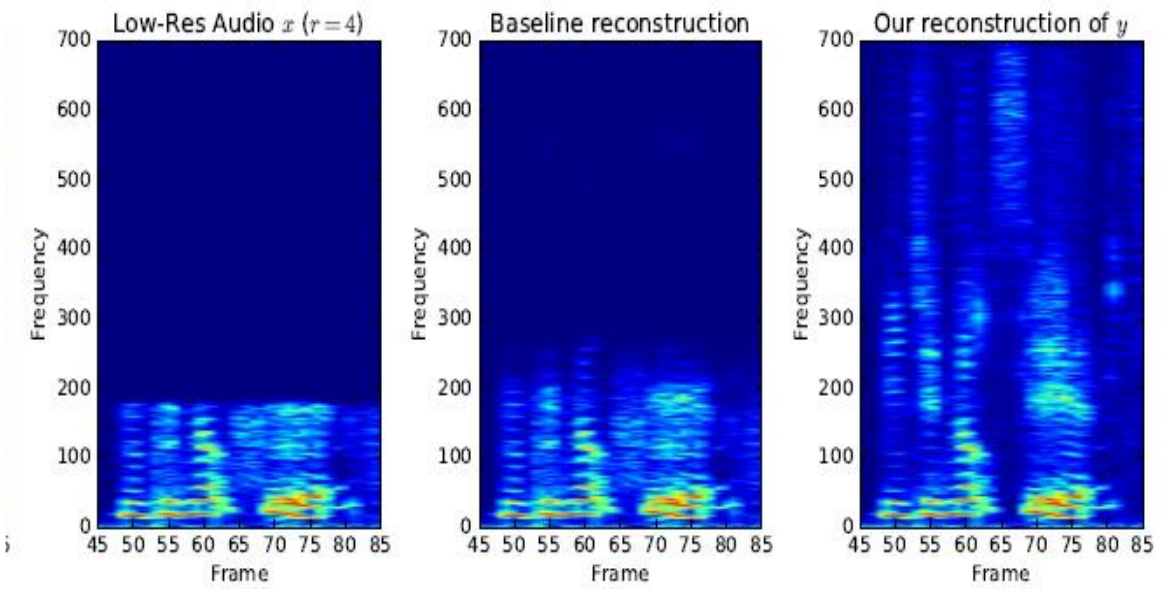
Diffusion Text2Speech

Generative model of speech signals

Betker, Better speech synthesis through scaling 2023

Audio Super Resolution

Conditional generative model P(high-res signal | low-res audio signal)



Low res signal

High res audio signal

Kuleshov et al., 2017

Language Generation

Custom prompt

To get an A+ in deep generative models, students have to

Completion

To get an A+ in deep generative models, students have to be willing to work with problems that are a whole lot more interesting than, say, the ones that most students work on in class. If you're a great student, the question above can be avoided and you'll be able to do great work, but if you're not, you will need to go beyond the basics before getting good.

Now to be clear, this advice is not just for the deep-learning crowd; it is good advice for any student who is taking his or her first course in machine learning.

The key point is that if you have a deep, deep brain of a computer scientist, that's just as important to you.

P(next word | previous words)

Radford et al., 2019
Demo from talktotransformer.com

Language Generation -- ChatGPT

ER

What should I do to get an A+ in CS236 at Stanford?

To get an A+ in CS236 (Deep Generative Models) at Stanford, you will need to excel in both your understanding of the material and your performance in assignments and exams. Here are some general tips to help you achieve this:

1. **Attend Lectures and Engage Actively**: Attend all lectures and actively engage with the material. Take thorough notes, ask questions, and participate in discussions. This will help you understand the concepts better.

2. **Read the Assigned Material**: Make sure to read the assigned textbooks, papers, and supplementary materials. Understanding the theoretical foundations is crucial.

3. **Stay Organized**: Keep a well-organized notebook or digital notes. This will help you quickly review and understand the material.

4. **Seek Help When Needed**: Don't hesitate to ask questions if you're having trouble with a concept. You can ask the professor, teaching assistants, or classmates for clarification.

5. **Complete Assignments Thoroughly**: Take your time to complete assignments, ensuring you fully understand the requirements and concepts involved. Start early and seek help if you're stuck. High-quality assignments are often a significant portion of your grade.

6. **Collaborate, but Don't Plagiarize**: Collaboration is often encouraged, but make sure you understand your institution's policy on collaboration and plagiarism. Always give credit where it's due and submit original work.

…

15. **Meet with the Professor**: If you're aiming for an A+, consider scheduling meetings with the professor to discuss your progress and seek feedback.

Remember that getting an A+ can be highly competitive, and the specific grading criteria may vary from one course to another and one professor to another. It's essential to understand the grading policies and expectations of your instructor. Always aim for excellence, but also keep in mind that learning and understanding the material should be your primary goal. Good luck!

Machine Translation

Conditional generative model P(English text| Chinese text)

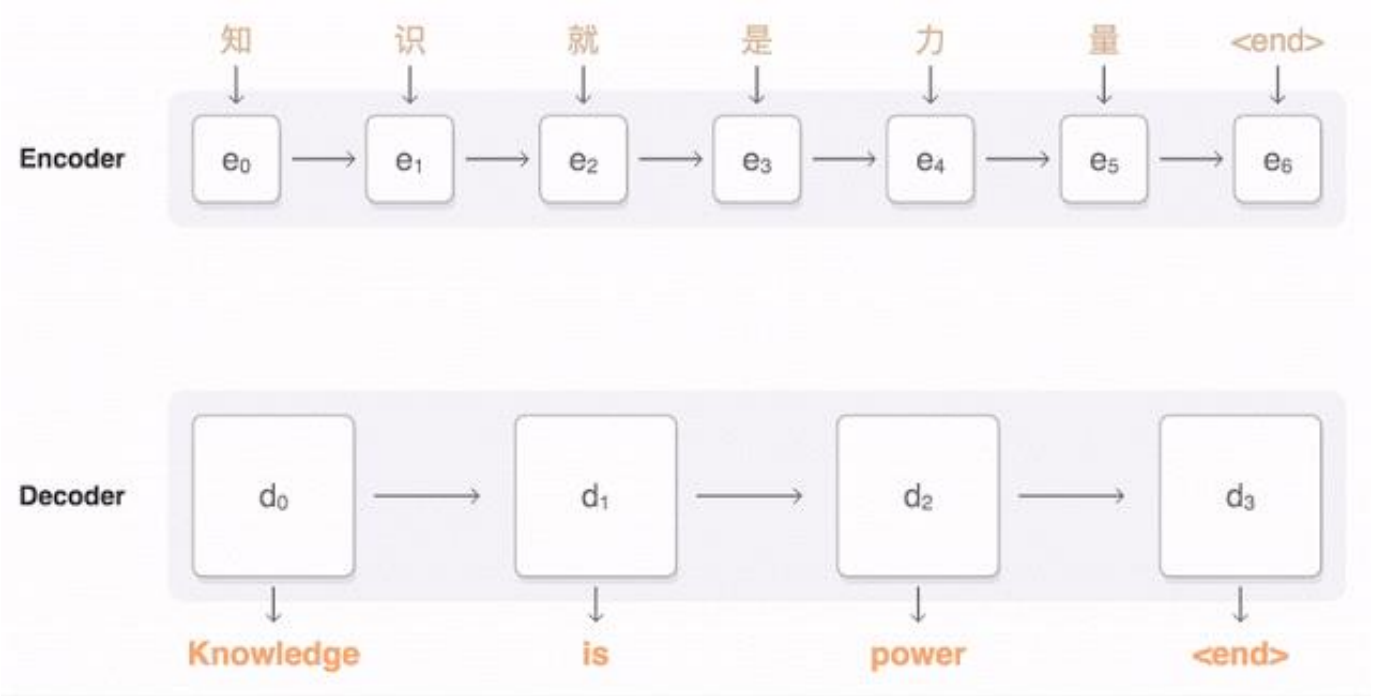
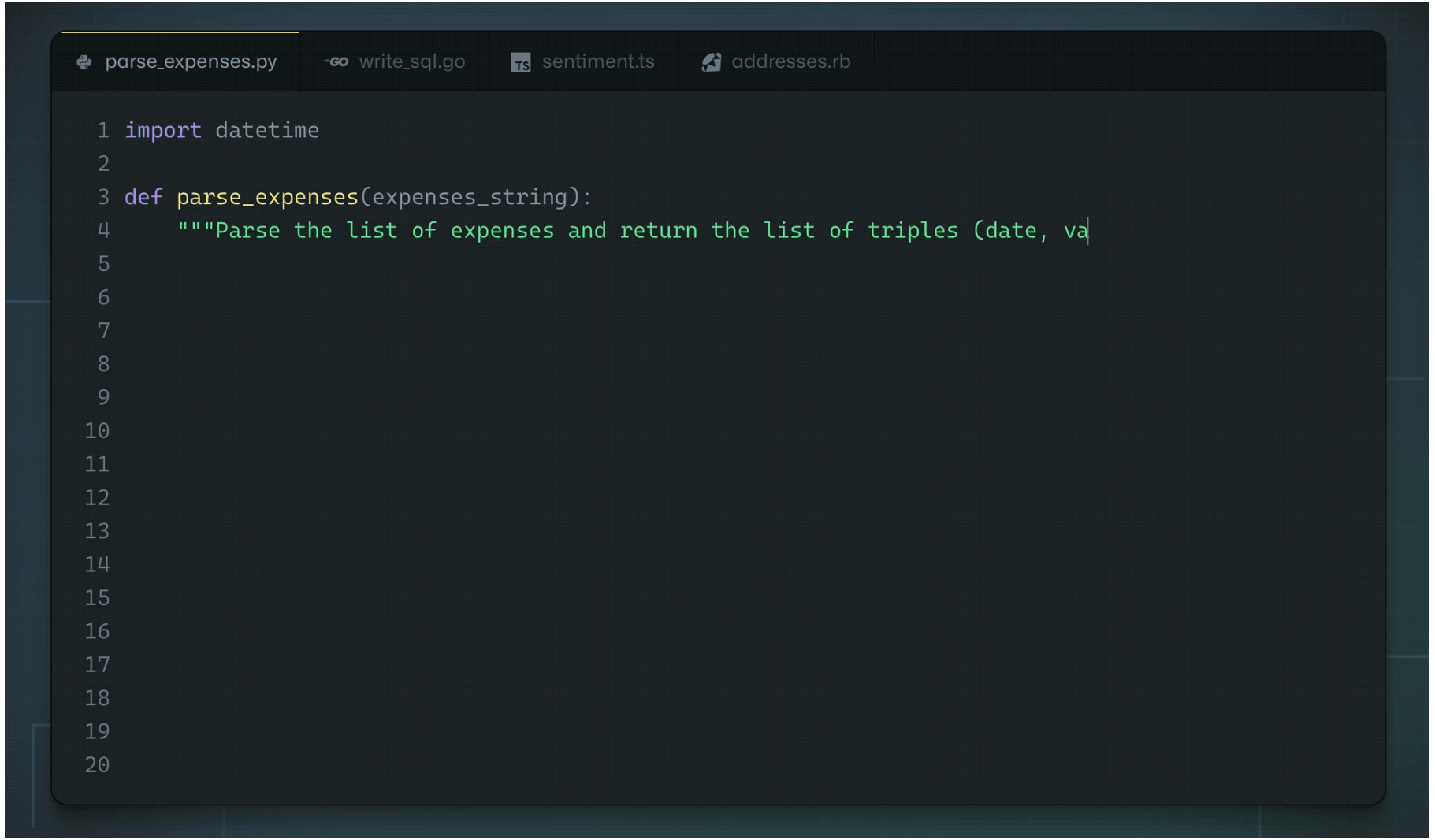


Figure from Google AI research blog.

Code Generation



OpenAI Codex

Video Generation

Suddenly, the walls of the embankment broke and there was a huge flood

Video Generation

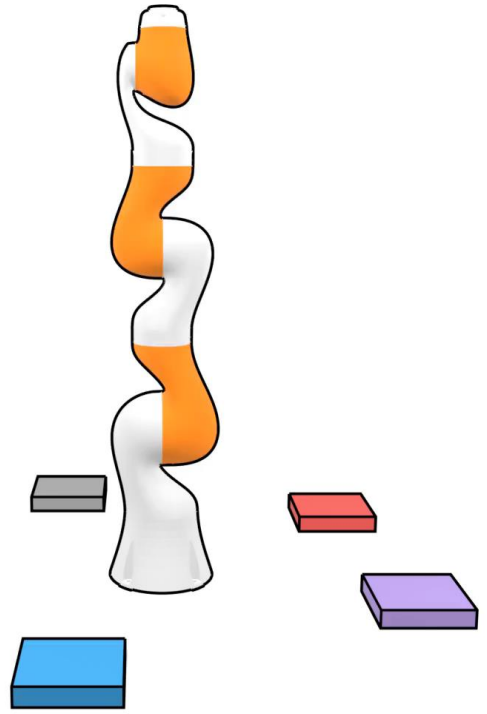
a couple sledding down a snowy hill on a tire roman chariot style

Video Generation

Imitation Learning

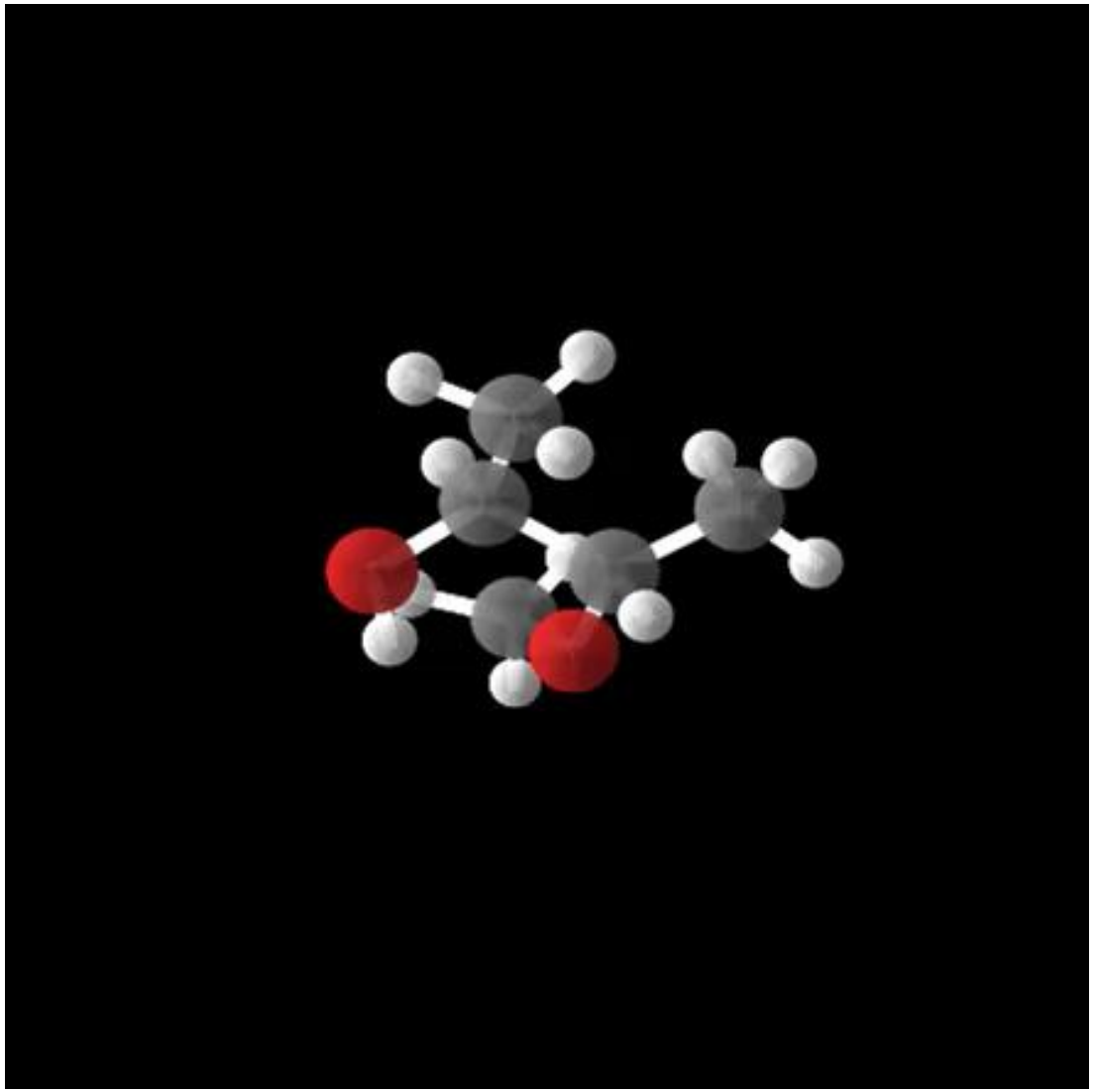
Conditional generative model P(actions | past observations)

Li et al., 2017



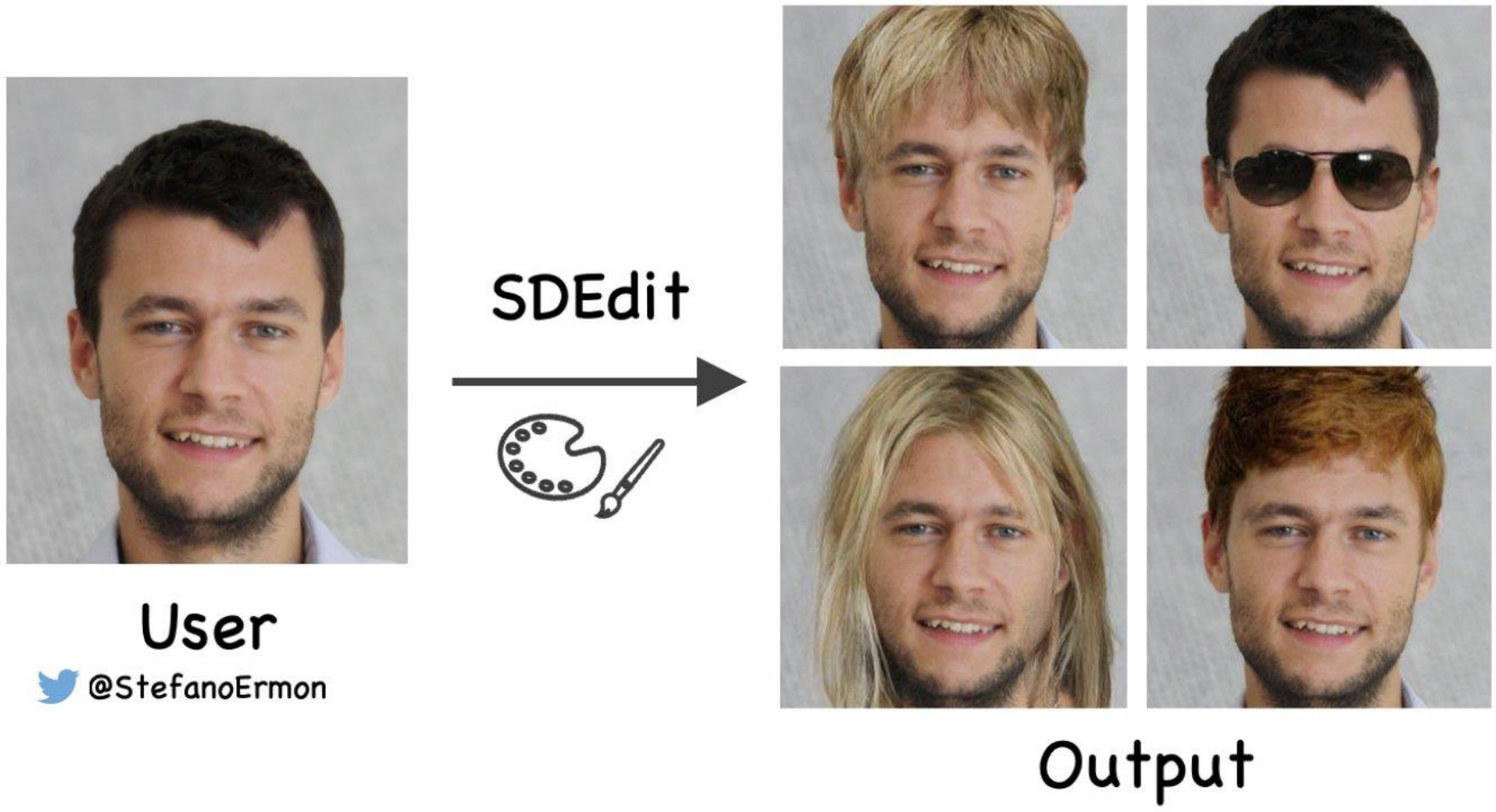
Janner et al., 2022

Molecule generation



DeepFakes

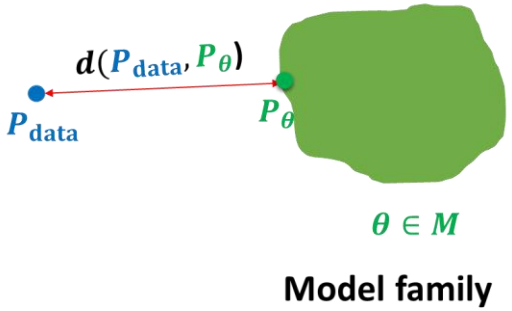
Which image is real?



Roadmap and Key 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?

$\mathbf{x}_i \sim P_{\text{data}}$
$i = 1, 2, \ldots, n$



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

Syllabus

- Fully observed likelihood-based models
  - Autoregressive
  - Flow-based models
- Latent variable models
  - Variational learning
  - Inference amortization
  - Variational autoencoder
- Implicit generative models
  - Two sample tests, embeddings, F-divergences
  - Generative Adversarial Networks
- Energy Based Models
- Score-based Diffusion Generative Models
- Learn about algorithms, theory & applications

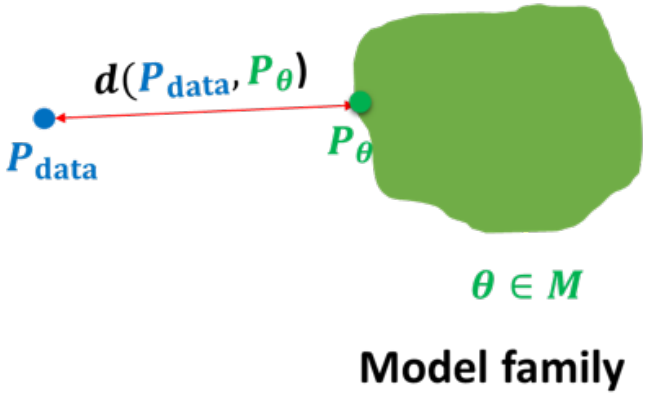
Overview

- What is a generative model
- Representing probability distributions
  - Curse of dimensionality
  - Crash course on graphical models (Bayesian networks)
  - Generative vs discriminative models
  - Neural models

Learning a generative model

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

$$x_i \sim P_{\text{data}}$$
$$i = 1, 2, \ldots, n$$



- We want to learn a probability distribution $p(x)$ over images x such that
  - **Generation:** If we sample $x_{new} \sim p(x)$, 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 $p(x)$

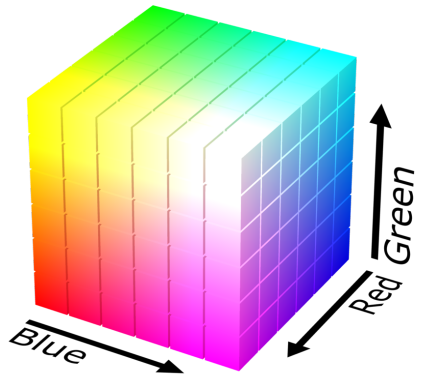
Basic discrete distributions

- Bernoulli distribution: (biased) coin flip
  - $D = \{Heads, Tails\}$
  - Specify $P(X = Heads) = p$. Then $P(X = Tails) = 1 - p$.
  - Write: $X \sim Ber(p)$
  - Sampling: flip a (biased) coin
- Categorical distribution: (biased) m-sided dice
  - $D = \{1, \cdots, m\}$
  - Specify $P(Y = i) = p_i$, such that $\sum p_i = 1$
  - Write: $Y \sim Cat(p_1, \cdots, p_m)$
  - Sampling: roll a (biased) die

Example of joint distribution

Modeling a single pixel's color. Three discrete random variables:

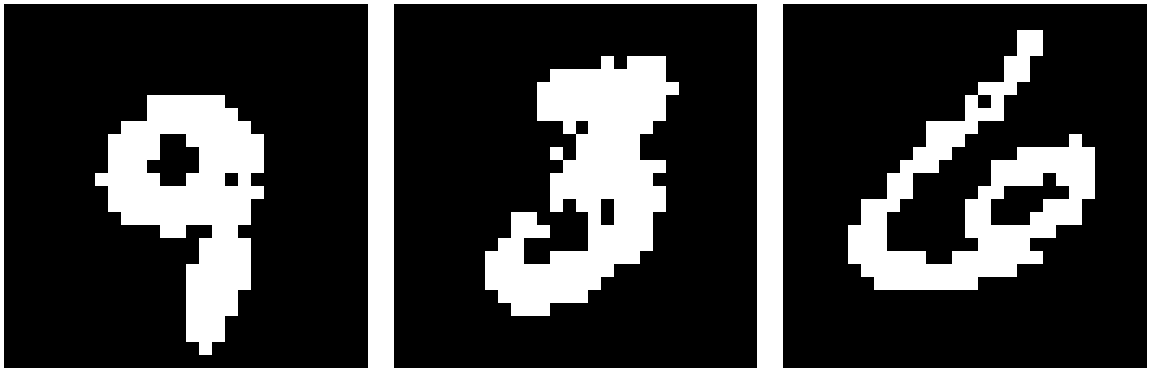
- Red Channel R. $\text{Val}(R) = \{0, \cdots, 255\}$
- Green Channel G. $\text{Val}(G) = \{0, \cdots, 255\}$
- Blue Channel B. $\text{Val}(B) = \{0, \cdots, 255\}$



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

$$256 * 256 * 256 - 1$$

Example of joint distribution



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

$$\underbrace{2 \times 2 \times \cdots \times 2}_{n \text{ times}} = 2^n$$

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

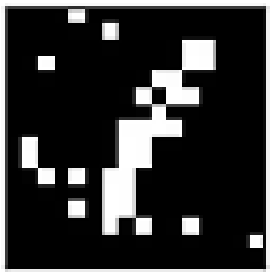
$$2^n - 1$$

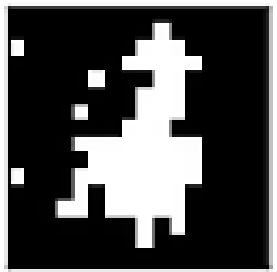
Structure through independence

- If $X_1, \ldots, X_n$ are independent, then

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

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





Two important rules

1. **Chain rule** Let $S_1, \ldots S_n$ be events, $p(S_i) > 0$.

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

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 Chain Rule

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

- 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 parameters.
  - $\cdots$
- $2^n - 1$ is still exponential, chain rule does not buy us anything.
- Now suppose $X_{i+1} \perp X_1, \ldots, X_{i-1} \mid X_i$, then

$$\begin{aligned} p(x_1, \ldots, x_n) &= p(x_1)p(x_2 \mid x_1)p(x_3 \mid \cancel{x_1}, x_2) \cdots p(x_n \mid \cancel{x_1, \cdots, x_{n-1}}) \\ &= p(x_1)p(x_2 \mid x_1)p(x_3 \mid x_2) \cdots p(x_n \mid x_{n-1}) \end{aligned}$$

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

Bayes Network: General Idea

- Use conditional parameterization (instead of joint parameterization)
- For each random variable X_i, specify $p(x_i|\mathbf{x}_{\mathbf{A_i}})$ for set $\mathbf{X}_{\mathbf{A_i}}$ of random variables
- Then get joint parametrization as

$$p(x_1, \ldots, x_n) = \prod_i p(x_i|\mathbf{x}_{\mathbf{A_i}})$$

- 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

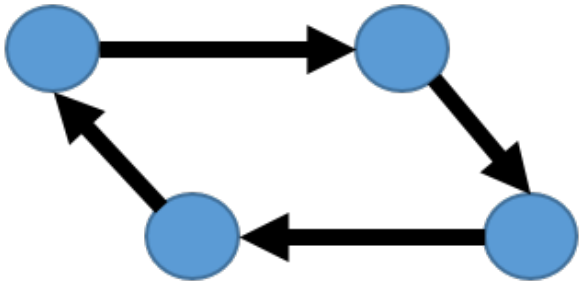
Bayesian networks

- A **Bayesian network** is specified by a *directed* **acyclic** graph (DAG) $G = (V, E)$ with:
  1. One node $i \in V$ for each random variable X_i
  2. One conditional probability distribution (CPD) per node, $p(x_i \mid \mathbf{x}_{\mathrm{Pa}(i)})$, 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:

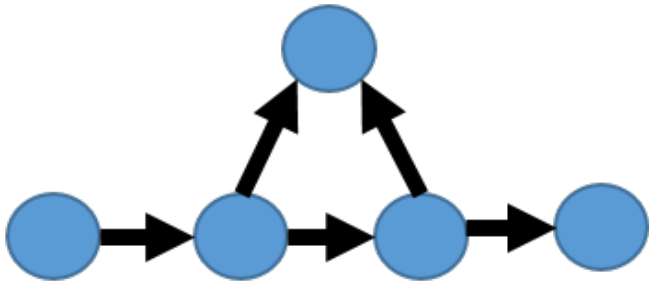
$$p(x_1, \ldots x_n) = \prod_{i \in V} p(x_i \mid \mathbf{x}_{\mathrm{Pa}(i)})$$

- Claim: $p(x_1, \ldots x_n)$ is a valid probability distribution because of ordering implied by DAG
- **Economical representation**: exponential in $|\mathrm{Pa}(i)|$, not $|V|$

Example



Directed cycle

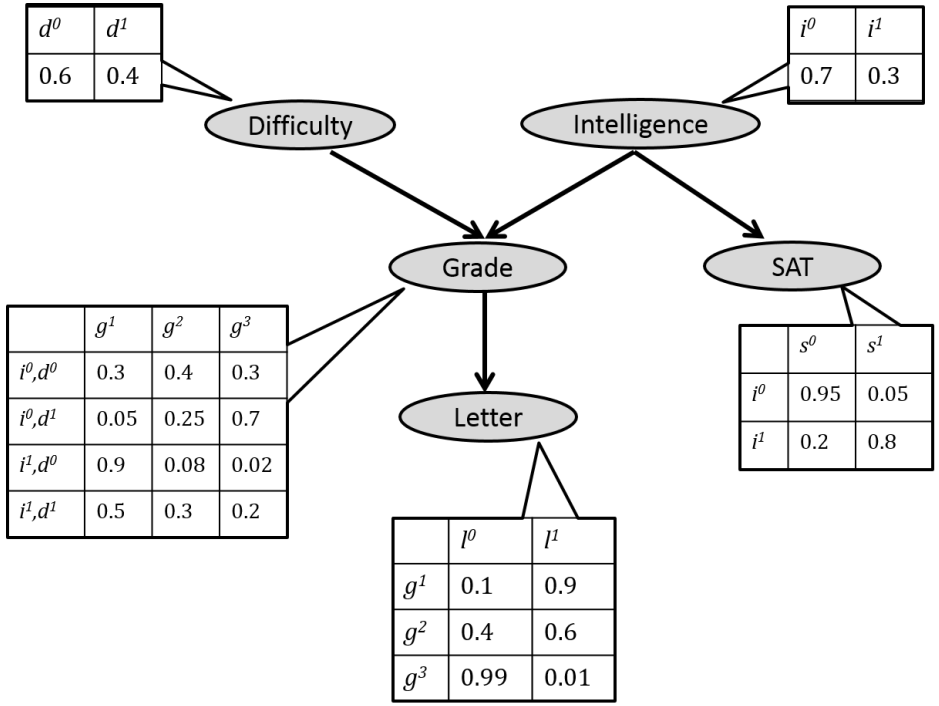


DAG

DAG stands for Directed Acyclic Graph

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

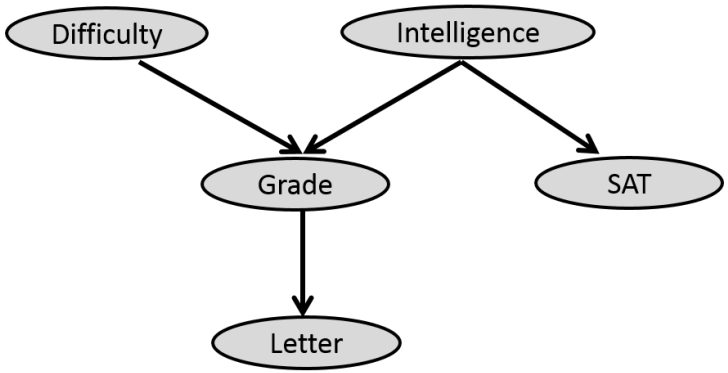
	l^0	l^1
g^1	0.1	0.9
g^2	0.4	0.6
g^3	0.99	0.01

- What is its joint distribution?

$$p(x_1, \ldots x_n) = \prod_{i \in V} p(x_i \mid \mathbf{x}_{\mathrm{Pa}(i)})$$

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

Bayesian network structure implies conditional independencies!



- The joint distribution corresponding to the above BN factors as

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

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

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

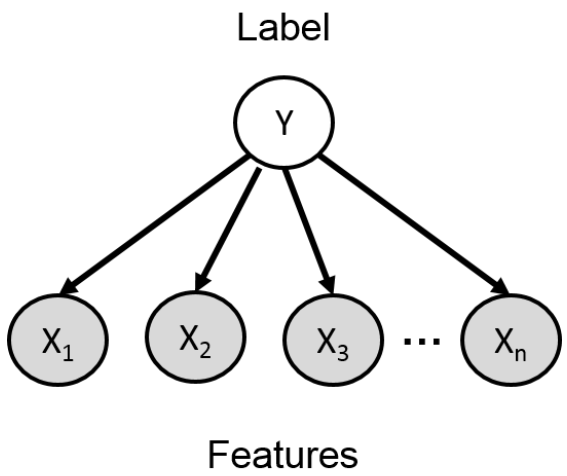
- Thus, we are assuming the following additional independencies:
$D \perp I, \qquad S \perp \{D, G\} \mid I, \qquad L \perp \{I, D, S\} \mid G.$

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)
- Next: generative vs. discriminative; functional parameterizations

Naive Bayes for single label prediction

- Classify e-mails 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 e-mail, and 0 otherwise
  - E-mails are drawn according to some distribution $p(Y, X_1, \ldots, X_n)$
- Words are conditionally independent given Y:



- Then

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

Example: naive Bayes for classification

- Classify e-mails 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 e-mail, and 0 otherwise
  - E-mails are drawn according to some distribution $p(Y, X_1, \ldots, X_n)$
- Suppose that the words are conditionally independent given Y. Then,

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

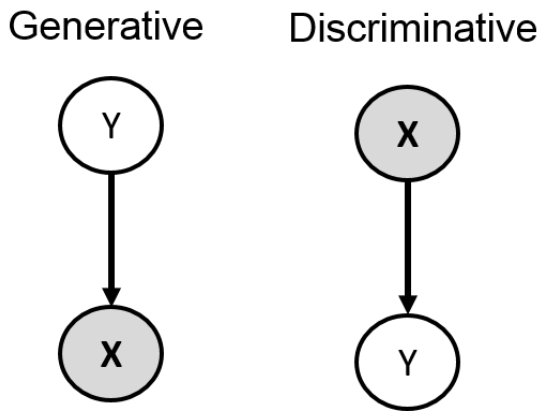
Estimate parameters from training data. **Predict** with Bayes rule:

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

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

Discriminative versus generative models

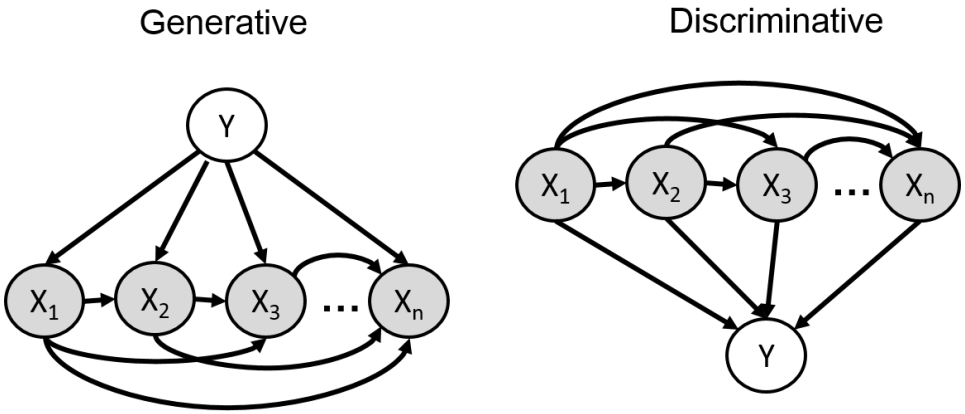
- Using chain rule $p(Y, \mathbf{X}) = p(\mathbf{X} \mid Y)p(Y) = p(Y \mid \mathbf{X})p(\mathbf{X})$. Corresponding Bayesian networks:



- However, suppose all we need for prediction is $p(Y \mid \mathbf{X})$
- In the left model, we need to specify/learn *both* $p(Y)$ and $p(\mathbf{X} \mid Y)$, then compute $p(Y \mid \mathbf{X})$ via Bayes rule
- In the right model, it suffices to estimate just the **conditional distribution** $p(Y \mid \mathbf{X})$
  - We never need to model/learn/use $p(\mathbf{X})$!
  - Called a **discriminative** model because it is only useful for discriminating Y's label when given $\mathbf{X}$

Discriminative versus generative models

- Since $\mathbf{X}$ is a random vector, chain rules will give
  - $p(Y, \mathbf{X}) = p(Y)p(X_1 \mid Y)p(X_2 \mid Y, X_1) \cdots p(X_n \mid Y, X_1, \cdots, X_{n-1})$
  - $p(Y, \mathbf{X}) = p(X_1)p(X_2 \mid X_1)p(X_3 \mid X_1, X_2) \cdots p(Y \mid X_1, \cdots, X_{n-1}, X_n)$

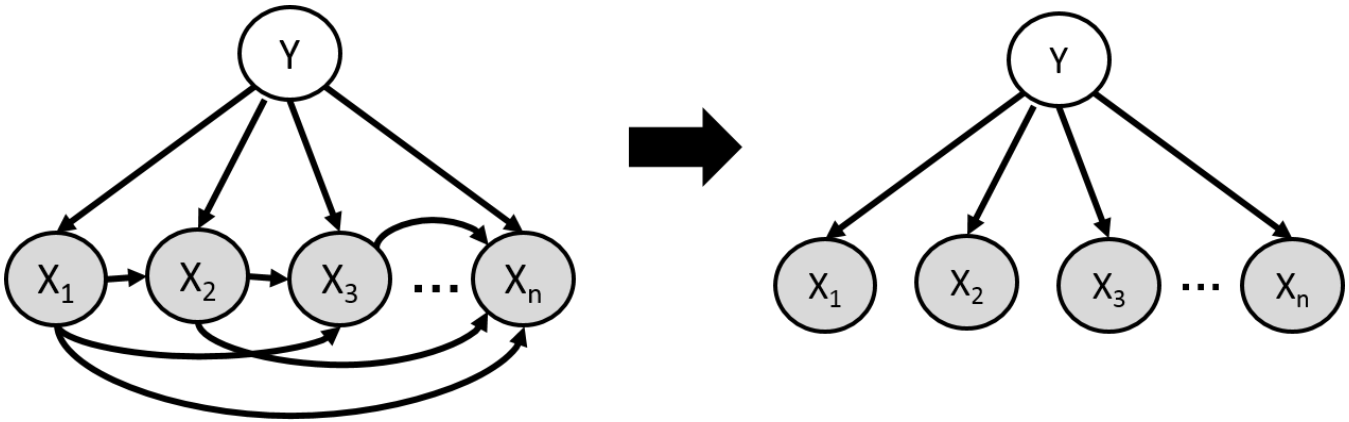


We must make the following choices:

1. In the generative model, $p(Y)$ is simple, but how do we parameterize $p(X_i \mid \mathbf{X}_{pa(i)}, Y)$?
2. In the discriminative model, how do we parameterize $p(Y \mid \mathbf{X})$? Here we assume we don't care about modeling $p(\mathbf{X})$ because $\mathbf{X}$ is always given to us in a classification problem

Naive Bayes

1. For the generative model, assume that $X_i \perp \mathbf{X}_{\_i} \mid Y$ (**naive Bayes**)



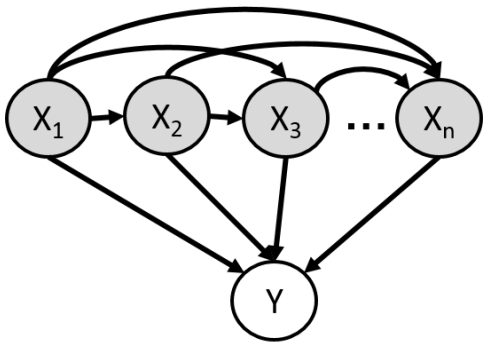
Logistic regression

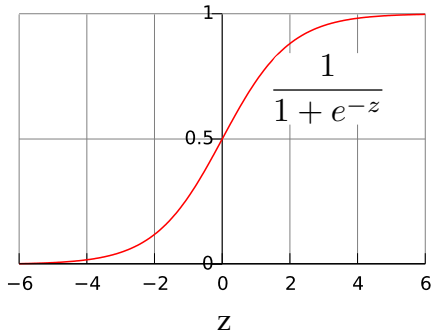
1. For the discriminative model, assume that

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

2. Not represented as a table anymore. It is a parameterized function of $\mathbf{x}$ (regression)
   - Has to be between 0 and 1
   - Depend in some *simple* but reasonable way on $x_1, \cdots, x_n$
   - Completely specified by a vector $\boldsymbol{\alpha}$ of $n+1$ parameters (**compact representation**)

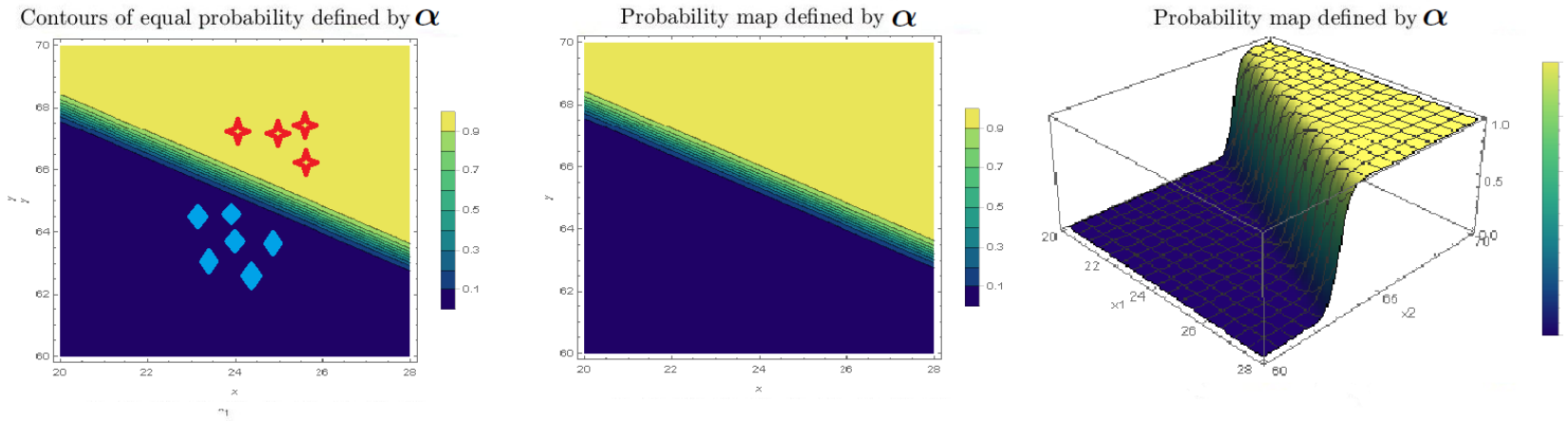
Linear dependence: let $z(\boldsymbol{\alpha}, \mathbf{x}) = \alpha_0 + \sum_{i=1}^{n} \alpha_i x_i$. Then, $p(Y = 1 \mid \mathbf{x}; \boldsymbol{\alpha}) = \sigma(z(\boldsymbol{\alpha}, \mathbf{x}))$, where $\sigma(z) = 1/(1 + e^{-z})$ is called the **logistic function**:





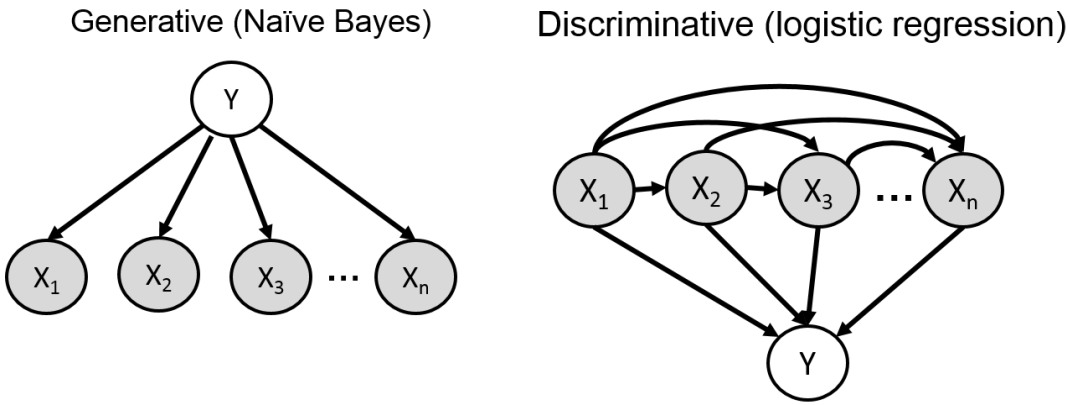
Logistic regression

Linear dependence: let $z(\boldsymbol{\alpha}, \mathbf{x}) = \alpha_0 + \sum_{i=1}^{n} \alpha_i x_i$. Then, $p(Y = 1 \mid \mathbf{x}; \boldsymbol{\alpha}) = \sigma(z(\boldsymbol{\alpha}, \mathbf{x}))$, where $\sigma(z) = 1/(1 + e^{-z})$ is called the **logistic function**



1. Decision boundary $p(Y = 1 \mid \mathbf{x}; \boldsymbol{\alpha}) > 0.5$ is linear in $\mathbf{x}$
2. Equal probability contours are straight lines
3. Probability rate of change has very specific form (third plot)

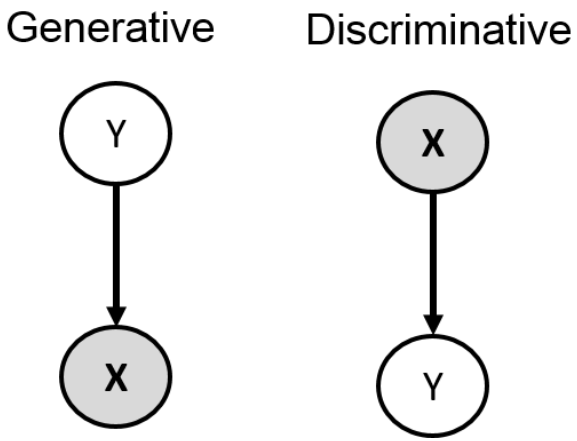
Discriminative models are powerful



- Logistic model does *not* assume $X_i \perp \mathbf{X}_{-i} \mid Y$, unlike naive Bayes
- This can make a big difference in many applications
- For example, in spam classification, let $X_1 = 1[\text{“bank” in e-mail}]$ and $X_2 = 1[\text{“account” in e-mail}]$
- Regardless of whether spam, these always appear together, i.e. $X_1 = X_2$
- Learning in naive Bayes results in $p(X_1 \mid Y) = p(X_2 \mid Y)$. Thus, naive Bayes **double counts the evidence**
- Learning with logistic regression sets $\alpha_1 = 0$ or $\alpha_2 = 0$, in effect ignoring it

Generative models are still very useful

Using chain rule $p(Y, \mathbf{X}) = p(\mathbf{X} \mid Y)p(Y) = p(Y \mid \mathbf{X})p(\mathbf{X})$. Corresponding Bayesian networks:



1. Using a conditional model is only possible when $\mathbf{X}$ is always observed
   - When some X_i variables are unobserved, the generative model allows us to compute $p(Y \mid \mathbf{X}_{evidence})$ by marginalizing over the unseen variables

Neural Models

1. In discriminative models, we assume that

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

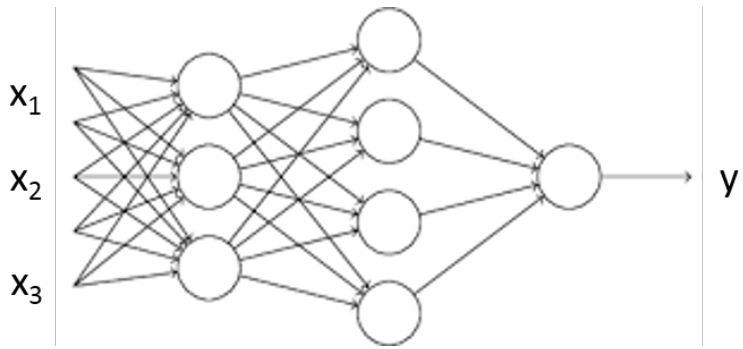
2. **Linear** dependence:
   - let $z(\boldsymbol{\alpha}, \mathbf{x}) = \alpha_0 + \sum_{i=1}^{n} \alpha_i x_i$.
   - $p(Y = 1 \mid \mathbf{x}; \boldsymbol{\alpha}) = \sigma(z(\boldsymbol{\alpha}, \mathbf{x}))$, where $\sigma(z) = 1/(1 + e^{-z})$ is the **logistic function**
   - Dependence might be too simple
3. **Non-linear** dependence: let $\mathbf{h}(A, \mathbf{b}, \mathbf{x}) = f(A\mathbf{x} + \mathbf{b})$ be a non-linear transformation of the inputs (*features*).
   $p_{\text{Neural}}(Y = 1 \mid \mathbf{x}; \boldsymbol{\alpha}, A, \mathbf{b}) = \sigma(\alpha_0 + \sum_{i=1}^{h} \alpha_i h_i)$
   - More flexible
   - More parameters: $A, \mathbf{b}, \boldsymbol{\alpha}$

Neural Models

1. In discriminative models, we assume that

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

2. **Linear** dependence: let $z(\boldsymbol{\alpha}, \mathbf{x}) = \alpha_0 + \sum_{i=1}^{n} \alpha_i x_i$.
   $p(Y = 1 \mid \mathbf{x}; \boldsymbol{\alpha}) = f(z(\boldsymbol{\alpha}, \mathbf{x}))$, where $f(z) = 1/(1 + e^{-z})$ is the **logistic function**
   - Dependence might be too simple
3. **Non-linear** dependence: let $\mathbf{h}(A, \mathbf{b}, \mathbf{x}) = f(A\mathbf{x} + \mathbf{b})$ be a non-linear transformation of the inputs (*features*).
   $p_{\text{Neural}}(Y = 1 \mid \mathbf{x}; \boldsymbol{\alpha}, A, \mathbf{b}) = f(\alpha_0 + \sum_{i=1}^{h} \alpha_i h_i)$
   - More flexible
   - More parameters: $A, \mathbf{b}, \boldsymbol{\alpha}$
   - Can repeat multiple times to get a neural network



Bayesian networks vs neural models

- Using Chain Rule

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

  Fully General

- Bayes Net

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

  Assumes conditional independencies

- Neural Models

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

  Assume specific functional form for the conditionals. 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 $p_X(x) = \frac{1}{b-a} 1[a \leq x \leq b]$
  - Etc.
- If $\boldsymbol{X}$ is a continuous random vector, we can usually represent it using its **joint probability density function**:
  - Gaussian: if $p_X(x) = \frac{1}{\sqrt{(2\pi)^n |\boldsymbol{\Sigma}|}} \exp\left(-\frac{1}{2}(x-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(x-\boldsymbol{\mu})\right)$
- 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. Examples:
- **Mixture of 2 Gaussians**: Bayes net $Z \to X$ with factorization $p_{Z,X}(z,x) = p_Z(z)p_{X|Z}(x \mid z)$ and
  - $Z \sim Bernoulli(p)$
  - $X \mid (Z=0) \sim \mathcal{N}(\mu_0, \sigma_0)$, $X \mid (Z=1) \sim \mathcal{N}(\mu_1, \sigma_1)$
  - The parameters are $p, \mu_0, \sigma_0, \mu_1, \sigma_1$
- Bayes net $Z \to X$ 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**: Bayes net $Z \to X$ with factorization $p_{Z,X}(z,x) = p_Z(z)p_{X|Z}(x \mid z)$ and
  - $Z \sim \mathcal{N}(0,1)$
  - $X \mid (Z=z) \sim \mathcal{N}(\mu_\theta(z), e^{\sigma_\phi(z)})$ where $\mu_\theta : \mathbb{R} \to \mathbb{R}$ and σ_ϕ are neural networks with parameters (weights) θ, ϕ respectively
  - **Note**: Even if μ_θ, σ_ϕ are very deep (flexible), functional form is still Gaussian