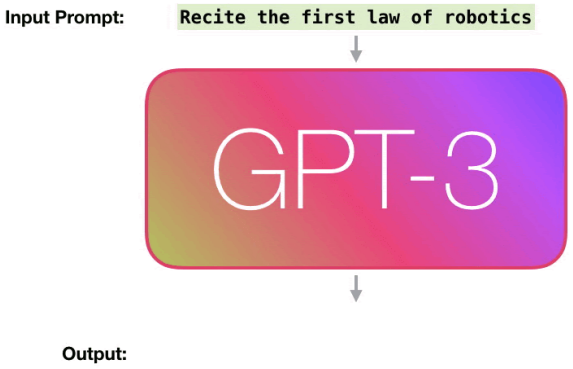
CS M146: Introduction to Machine Learning
Neural Networks – Generative Models

Aditya Grover

https://aditya-grover.github.io/ @adityagrover_

Generative AI

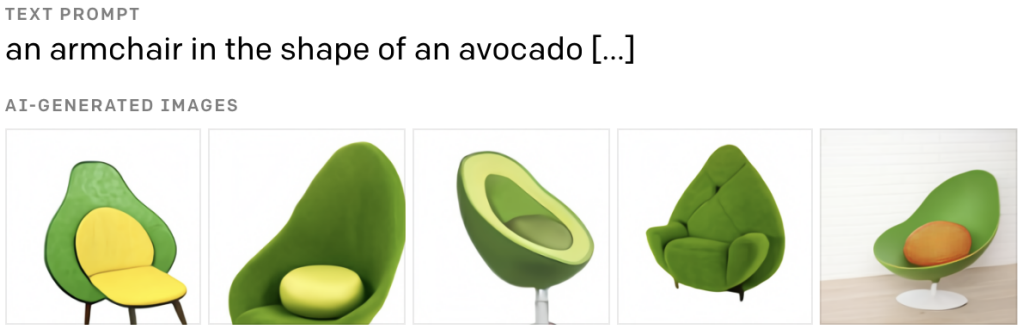


language

[Brown et al., 2020]

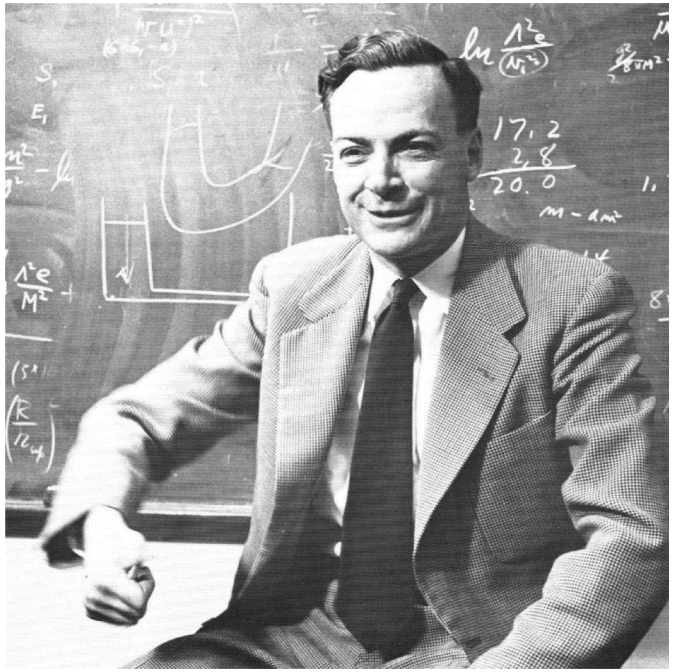
speech

[Radford et al., 2022]



vision

[Ramesh et al., 2021]



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

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

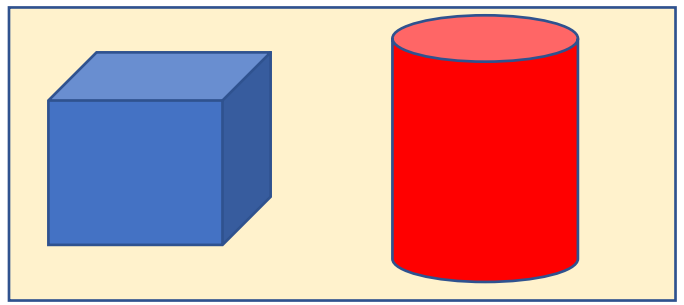
Hand-Engineered Generative Models

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)



Raw sensory outputs

Statistical Generative Models

Statistical generative models are **learned from data**

Data
(e.g., images of dogs)

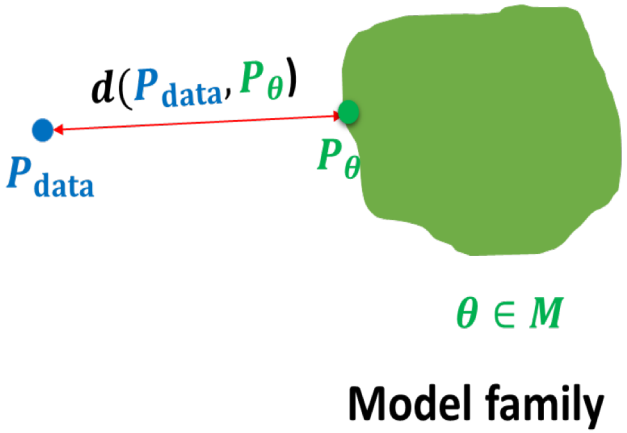
+

Prior Knowledge
(e.g., biology, background,
loss function, hypothesis class, ..)

Priors are always necessary, but there is a spectrum

Generative Modeling as Learning Probability Distributions

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

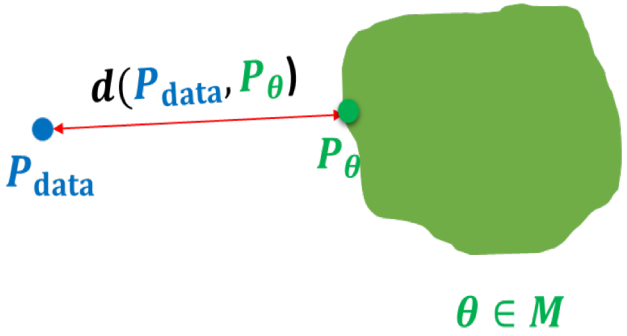


- **Given:** Training set of n datapoints $X = \{\boldsymbol{x}^{(i)}\}$
  - E.g., n dog images

Assumption: There exists a true probability distribution $p_{\text{data}}(\boldsymbol{x})$ and our training dataset denotes independent samples from this distribution

Generative Modeling as Learning Probability Distributions

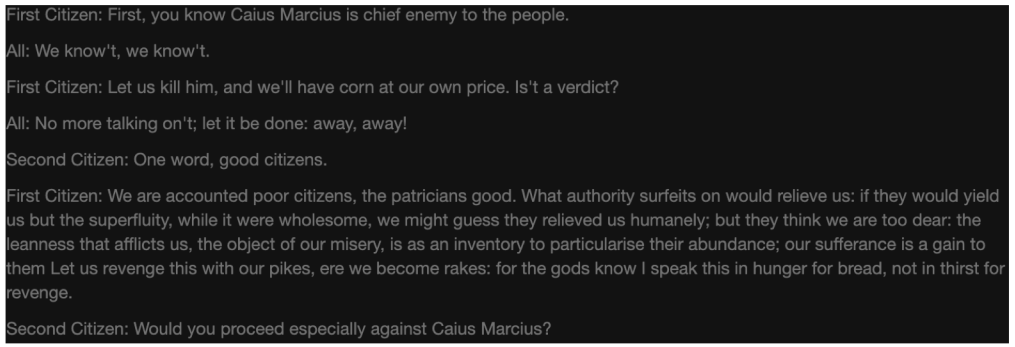
$x_i \sim P_{\text{data}}$
$i = 1, 2, \ldots, n$



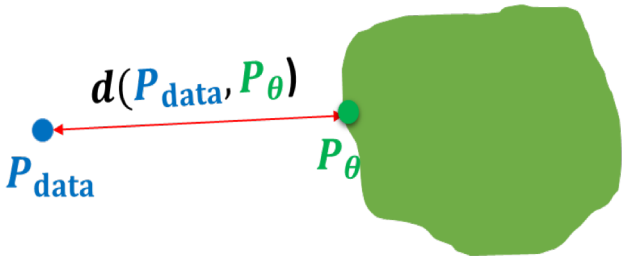
- **Given:** Training set of n images of dogs $X = \{\boldsymbol{x}^{(i)}\}$
- **Goal:** Learn a probability distribution $p_{\boldsymbol{\theta}}(\boldsymbol{x})$ that is "close to" $p_{\text{data}}(\boldsymbol{x})$
- Ideally, $p_{\boldsymbol{\theta}}(\boldsymbol{x})$ will be:
  - High when $\boldsymbol{x}$ looks like a dog
  - Low otherwise, e.g., when $\boldsymbol{x}$ is a cat, car, any non-dog object or even noise!

Why generative? After learning $\boldsymbol{p_\theta}$, we can sample from it to generate more $\boldsymbol{x}$'s similar to our training set

Another Example: Language Model



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



$\theta \in M$

Model family

- **Given:** Training set of n English sentences from Shakespeare $X = \{\boldsymbol{x}^{(i)}\}$
- **Goal:** Learn a probability distribution $p_{\boldsymbol{\theta}}(\boldsymbol{x})$ that is "close to" $p_{\text{data}}(\boldsymbol{x})$
- Ideally, $p_{\boldsymbol{\theta}}(\boldsymbol{x})$ will be:
  - High when $\boldsymbol{x}$ looks like a grammatically correct sentence in English written in the style of Shakespeare
  - Low otherwise, e.g., when $\boldsymbol{x}$ is a gibberish sentence, or written in a different style, or langauge

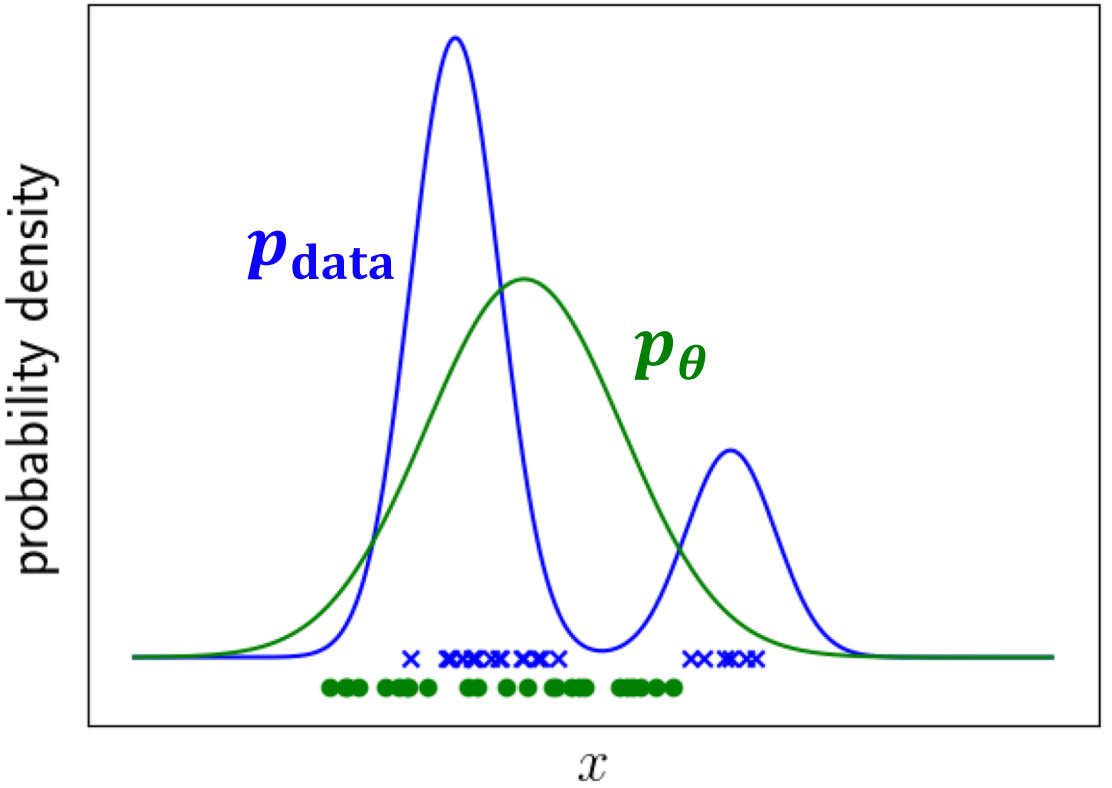
Warmup – 1 dimensional x

$\boldsymbol{p_{\text{data}}}$ data distribution (unknown)

$\times$ data samples (known)

$\boldsymbol{p_\theta}$ generative model

● generated samples



$\boldsymbol{p_\theta}$ not a great model

Parameterizing Probability Distributions

- Real data is high dimensional, E.g.,
  - An image can contain 1000's of pixels. Each pixel can take 256 values.
  - A sentence can contain many words. Each word can belong to a huge vocabulary

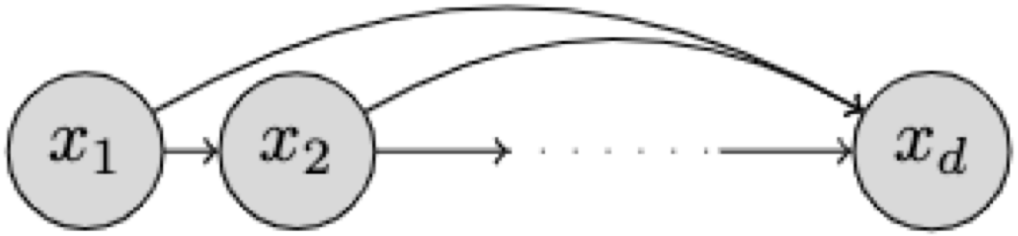
- **Key challenge:** How do we represent and learn probability distributions when $\boldsymbol{x}$ is high-dimensional?

Autoregressive Generative Models

- **Key idea:** Decompose the joint distribution as a product of 1d probability conditionals using chain rule of probability

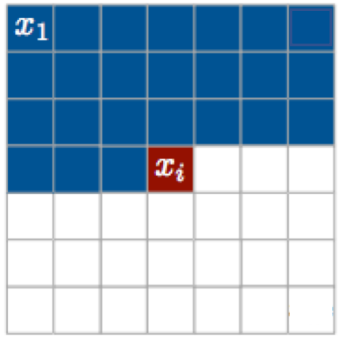
$$p_{\boldsymbol{\theta}}(\boldsymbol{x}) = \prod_{i=1}^{d} p_{\boldsymbol{\theta}}(x_i \,|x_1, x_2, \cdots, x_{i-1}) = \prod_{i=1}^{d} p_{\boldsymbol{\theta}}(x_i \,|x_{<i})$$

- Intuitively, given an ordered input $\boldsymbol{x}$, each x_i (pixel/word) depends on all other inputs that came before it



Example - Images

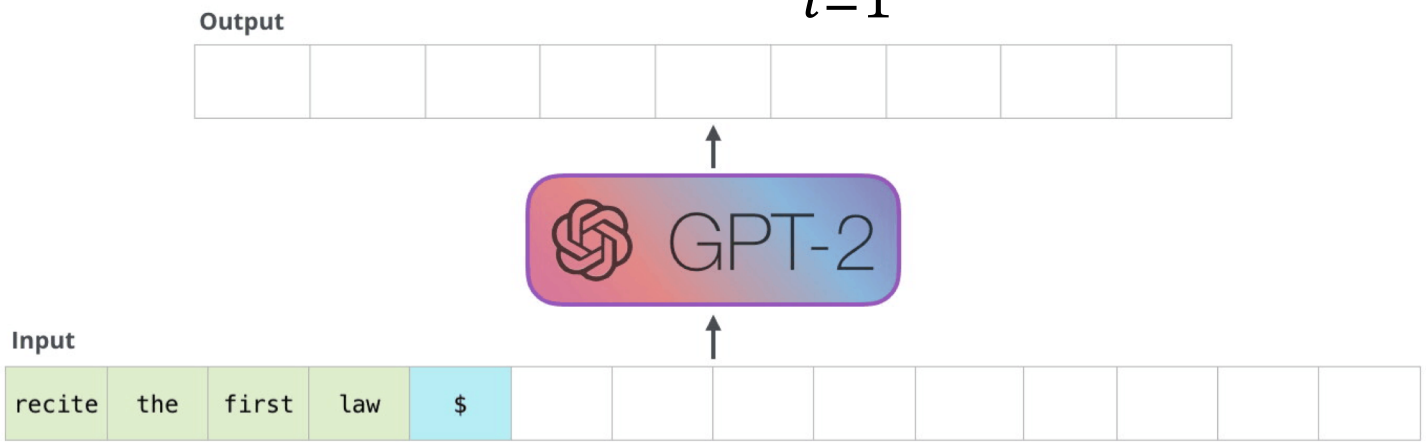
$$p(x_{1:d}) = \prod_{i=1}^{d} p(x_i | x_{<i})$$



- Think of image flattened out from left to right, top to bottom
- Each x_i is a pixel value:
  - 0 -255 for colored images, 0-1 for black and white images

Example - Text

$$p(x_{1:d}) = \prod_{i=1}^{d} p(x_i | x_{<i})$$



- Each x_i is a word that depends on previous words
- How many classes for each x_i?
  - Depends on length of vocabulary. Typically in the order of 10^5 words.

Representing Probability Conditionals

- Conditional $p_{\boldsymbol{\theta}}(x_i \mid x_{<i})$ is a probability distribution over a 1D x_i
  - During training, we have to pick a distribution type and parameterization scheme for $p_{\boldsymbol{\theta}}$
- **Distribution Type:** Depends on whether x_i is discrete or continuous
  - Continuous x_i: $p_{\boldsymbol{\theta}}$ can be a Gaussian distribution
  - Discrete x_i: $p_{\boldsymbol{\theta}}$ can be a categorical (multinomial) distribution

Representing Probability Conditionals

- **Distribution Type:** Depends on whether x_i is discrete or continuous
  - Continuous x_i: $p_{\boldsymbol{\theta}}$ can be a Gaussian distribution
  - Discrete x_i: $p_{\boldsymbol{\theta}}$ can be a categorical (multinomial) distribution
- The parameters $\boldsymbol{\theta}$ signify a mapping to the sufficient statistics of the chosen distribution for $p_{\boldsymbol{\theta}}$
  - Gaussian: $\boldsymbol{\theta}$ are parameters of any function that map to mean, variance of $p_{\boldsymbol{\theta}}$
  - Categorical: $\boldsymbol{\theta}$ are parameters of any function that map to probabilities of each category
- Parameters $\boldsymbol{\theta}$ can be shared or separate for each $p_{\boldsymbol{\theta}}(x_i \mid x_{<i})$
  - Shared: one function with d*k outputs where k is number of sufficient stats for a distribution type (e.g., k=2 for Gaussian, C for categorical)
  - Separate: d functions, each with k outputs

Parameterizing Conditionals via Neural Networks

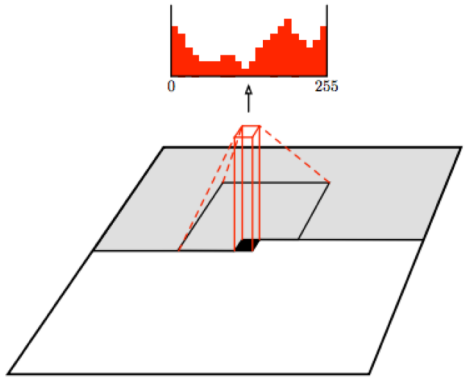
Linear/multi-layer NN based parameterizations

- Fully-visible sigmoid belief networks (FVSBN): O hidden layers
- Neural autoregressive density estimator (NADE): 1 hidden layer NN
- Masked autoencoder distribution estimation (MADE): multilayer NN

Neal, 1992, Larochelle & Murray, 2011, Germain et al., 2015

Parameterizing Conditionals via Modern Neural Networks

- Convolutional Neural Networks (CNN) – used for image data



1	1	1	1	1
1	1	1	1	1
1	1	0	0	0
0	0	0	0	0
0	0	0	0	0

- Recurrent Neural Networks (RNN) – used for sequential data
- Transformers – improvement over RNNs. Can process much larger sequences
  - Used in ChatGPT, Bard, etc.

van den Oord, 2016, Sutskevar et al., 2011, Radford et al., 2018

Learning Probability Conditionals

- **Learning** maximizes the model log-likelihood over the dataset $\mathcal{D}$
  - Each $p_{\boldsymbol{\theta}}(x_i \,|x_{<i})$ is essentially predicting x_i given $x_{<i}$ - Supervised learning!

- When x_i is continuous, the objective is mean squared error with labels x_i [regression]

- When x_i is discrete, the objective is softmax cross-entropy with labels x_i [multi-class classification]

Sampling

- Directed model permits **sequential sampling**, one variable at a time starting with x_1

$$\widehat{x_1} \sim p_\theta(x_1)$$
$$\widehat{x_2} \sim p_\theta(x_2|\widehat{x_1})$$
$$\widehat{x_3} \sim p_\theta(x_3|\widehat{x_1}, \widehat{x_2})$$
$$\ldots$$
$$\widehat{x_d} \sim p_\theta(x_d|\widehat{x_{<d}})$$

- Why sequential?
  - Each conditional x_i depends on the **sampled values** $\widehat{x_{<i}}$
- Explains why there is a time lag in producing each word when you use ChatGPT

Rome Was Not Built In A Day

Summary

- Generative Models learn parameterized probability distributions for any given dataset
  - Once learned, can sample from the model to generate more data

- Autoregressive Generative Model (e.g., ChatGPT)
  - Derived from the chain rule of probability as a product of conditionals
  - Each conditional is represented via a neural network
  - Learning each conditional is equivalent to doing supervised learning
  - Sampling from the model is sequential – one variable at a time conditioned on other variables.

Thank You!

- Thank You for a Great Quarter!