Introduction Lecture

Authors: Kan Min-Yen, Ang Shen Ting, Sunil Kumar

Scribe notes: can edit

1b Motivations

Self-Supervised Learning: In short, it's when we get the model to label the data itself and train itself on said labels. Essentially, it's like teaching a dog to take walks on its own.
Compression can be seen as unsupervised learning - see Hutter prize

Composability of neural nets: Unlike traditional machine learning blocks, neural nets lend themselves to a nice property of composability to arrive at the end to end systems to solve complex problems.

Discriminative vs. Generative

1c Likelihood-models

Estimate p_{data} from $x^{(1)}$, $x^{(2)}$, … $x^{(n)} \sim p_{data}(x)$

Want to learn distribution p to:

- Compute $p(x)\ \forall\ x$
- Sample $x \sim p(x)$

First talk about the case of discrete data, but aim to estimate distributions of complex, high dimensional data - e.g. image of 128x128x3 (Width x Height x Colour Channels) lies in a ~ 50,000 dimensional space.

Estimating Frequencies by Counting

Discrete data: samples take on values in a finite set {1,...,k}.

Model: Histogram

- Described by k non-negative numbers: $p_1,...,p_k$
- Training: Count frequencies of occurrence in training data

At runtime,
Inference: Lookup array of $p_1,...,p_k$ for any arbitrary i
Sampling:

1. Compute the cdf of p(x) from $p_1,...,p_k$
2. Draw a random number u from Uniform[0,1]
3. Return min i s.t. $u \leq F_i$

(See Inverse Transform Sampling for more mathematical background/details)

This fails in high dimensions! Most of the images in the possible space would be noise - e.g. MNIST: 28x28 images, binary pixels (each pixel $\in \{0,1\}$) has roughly 10^{236} possibilities.

Function Approximation

Use a parameterized function $p_\theta(x)$ instead of storing each probability, i.e. change the problem to: Learn θ s.t. $p_\theta(x) \approx p_{data}(x)$.

General Procedure:

- Take data $x^{(1)}, x^{(2)}, \ldots x^{(n)} \sim p_{data}(x)$
- Set up model class: set of parameterized distributions $p_\theta($
- Define search problem over: $argmin_\theta\ (\theta, x^{(1)}, \ldots, x^{(n)})$
- Want the loss function and search procedure to:
    - Work in high dimensions
    - Yield θ s.t. $p_\theta(x) \approx p_{data}(x)$
    - Only see the empirical data distribution

Issues with designing Neural Nets

Size of joint distribution table is exponential in the dimensionality of data. To rescue this issue, Bayes Net comes into the picture. Bayes Net is a directed acyclic graph over the variables in the data. This transforms joint distribution to conditional distribution tables, reducing the required space.

Other Resources

1. A more prose-like explanation of PixelCNN and auto-regressive generative models in general: https://towardsdatascience.com/auto-regressive-generative-models-pixelrnn-pixelcnn-32d192911173
2. Inverse Transform Sampling: https://en.wikipedia.org/wiki/Inverse_transform_sampling
3. Stochastic Gradient Descent: http://deeplearning.stanford.edu/tutorial/supervised/OptimizationStochasticGradientDescent/