Deep generative models for biologists

Clayton W. Seitz

January 16, 2022

Outline

Discriminative and generative models

Say we have a set of variables $\mathrm{x} = (x_1, x_2, ..., x_n)$ which might have some statistical dependence

The variable x might be an amino acid sequence, DNA sequence, microscopy image, etc.

In supervised discriminative learning, we may use observations of x to try and learn distributions such as $p(x_2|x_1)$ (i.e., inference)

In supervised generative learning, we try to explicity learn the joint distribution $p(\mathrm{x}) = p(x_1|x_2, ..., x_n)p(x_2|x_3, ..., x_n), ..., p(x_n)$, which is generally more difficult.

The basic sampling problem

Suppose we are given a joint distribution

$$p(\mathrm{x}) = \frac{1}{Z}\tilde{p}(\mathrm{x})$$

where $p(\mathrm{x})$ is easy to compute but Z is (too) hard to compute.

This very important situation arises in several contexts:

1. In Bayesian models where $p(x_1, x_2) := p(x_1|x_2)p(x_2)$ is easy to compute but $Z = \int p(x_1|x_2)p(x_2)dx_2$ can be very difficult or impossible to compute.

2. In models from statistical physics, e.g. the Ising model, we only know $\tilde{p}(\mathrm{x}) = e^{-H(\mathrm{x})}$ where $H(\mathrm{x})$ is the Hamiltonian - the Ising model is an example of a Markov network or an undirected graphical model.

Approximating the joint distribution

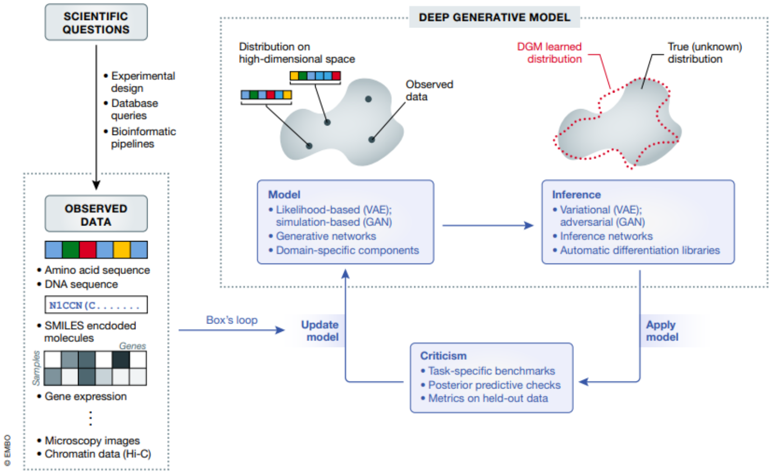
We would like to approximate $p(\mathrm{x})$

Variational methods are generally useful for Bayesian inference like $p(x_1|x_2)$ but can also be used to evaluate $p(\mathrm{x})$ by autoencoding x (called a variational autoencoder)

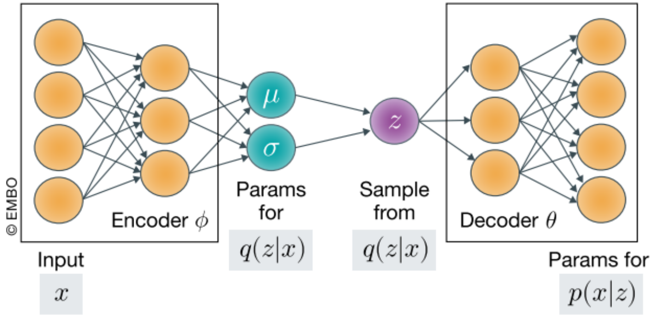
Generative adversarial networks (GANs) model $p(\mathrm{x})$ directly

In special scenarios, we may know $\tilde{p}(\mathrm{x})$ and we can use Monte-Carlo Markov Chain (MCMC) methods

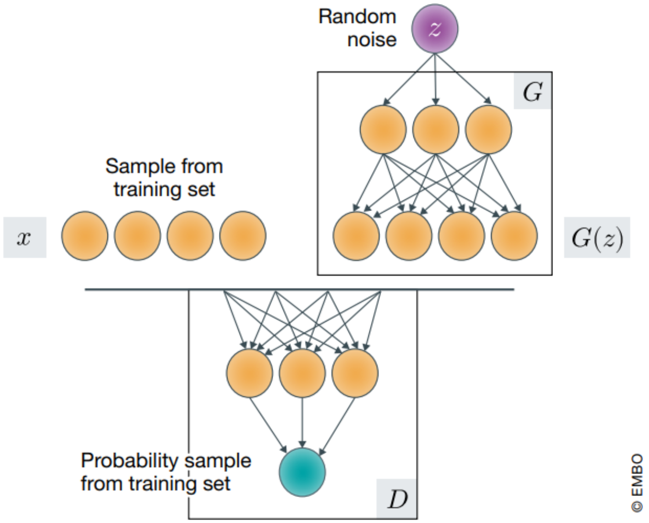
Applying deep generative models to biological data



Generative models: variational autoencoder



Generative models: adversarial networks



Cool biological applications of VAEs and GANs

Sequencing, Imaging, Other stuff

Monte-Carlo Markov Chain (MCMC)

- MCMC algorithms were originally developed in the 1940's by physicists at Los Alamos
- They were interested in modeling the probabilistic behavior of collections of atomic particles
- Simulation was difficult – the normalization constant Z was not known
- The term "Monte-Carlo" was coined at Los Alamos.
- Ulam and Metropolis overcame this problem by constructing a Markov chain for which the desired distribution was the stationary distribution
- Introduced to statistics and generalized with the Metropolis-Hastings algorithm (1970) and the Gibbs sampler of Geman and Geman (1984).

Markov Chains

For a state space Ω s.t. $\mathrm{x}_t \in \Omega$. x_t is a Markov process if:

$$P(\mathrm{x}_t|\mathrm{x}_{t-1}, \mathrm{x}_{t-2}, ..., \mathrm{x}_{t-N}) = P(\mathrm{x}_t|\mathrm{x}_{t-1})$$

which is commonly called the *memoryless property*.

- x_t can be generally be N-dimensional
- The chain is called *homogeneous* if $T(\mathrm{x}_t|\mathrm{x}_{t-1})$ is time-invariant.
- For discrete Ω, T is a matrix of probabilities with $T_{ij} = \Pr(i \rightarrow j)$
- For continuous Ω, T is the joint probability density $T(x_t, x_{t-1})$

Markov Chains

The Chapman-Kolmogorov equation marginalizes $T(x_t, x_{t-1})$:

$$\begin{aligned} P(x_t) &= \int T(x_t, x_{t-1}) dx_{t-1} \\ &= \int T(x_t | x_{t-1}) P(x_{t-1}) dx_{t-1} \end{aligned}$$

The chain satisfies *detailed balance* if

$$T(x_t, x_{t-1}) P(x_t) = T(x_{t-1}, x_t) P(x_{t-1})$$

which guarantees there is a unique stationary distribution $P_0(x_t)$

Monte-Carlo Markov Chain (MCMC)

A stationary distribution satisfies

$$P_0(x_t) = \int T(x_t|x_{t-1}) P_0(x_{t-1}) dx_{t-1}$$

- If a process is Markov e.g., Brownian motion, Ornstein-Uhlenbeck, $P_0(x_t)$ is a solution to the SDE
- We can also design $T(x_t, x_{t-1})$ s.t. $P_0(x_t)$ is a distribution we cannot sample from easily such as the Ising model
- The notion of “time” in the second case is artificial
- There are several MCMC algorithms, we will focus on Gibbs MCMC

Gibbs sampling

- Suppose $p(\mathrm{x})$ is a p.d.f. or p.m.f. that is difficult to sample from directly.
- Suppose, though, that we *can* easily sample from the conditional distributions e.g., $p(x_1|x_2, ..., x_n)$.
- The Gibbs sampler proceeds as follows:
  1. set x to some initial starting values
  2. then sample $x_1|x_2, ..., x_n$, then sample $x_2|x_1, ..., x_n$, and so on.

Gibbs sampling

0. Set (x_0, y_0) to some starting value.
1. Sample $x_1 \sim p(x|y_0)$, that is, from the conditional distribution $X \mid Y = y_0$.
   Current state: (x_1, y_0)
   Sample $y_1 \sim p(y|x_1)$, that is, from the conditional distribution $Y \mid X = x_1$.
   Current state: (x_1, y_1)
2. Sample $x_2 \sim p(x|y_1)$, that is, from the conditional distribution $X \mid Y = y_1$.
   Current state: (x_2, y_1)
   Sample $y_2 \sim p(y|x_2)$, that is, from the conditional distribution $Y \mid X = x_2$.
   Current state: (x_2, y_2)

   $\vdots$

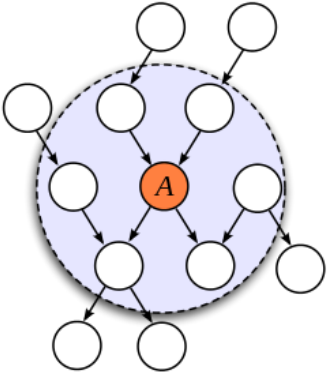
Repeat iterations 1 and 2, M times.

Bayesian inference using Gibbs sampling

Joint distributions factor according to

$$P(\mathrm{x}) = P(x_1|x_2, ..., x_n)P(x_2|x_3, ..., x_n), ..., P(x_n)$$

$P(x_1|x_2, ..., x_n)$ may not include all $n-1$ variables



The useful information is called a Markov blanket

Learning graph structure

Learning the graph structure $\mathcal{G} = (V, E)$ is a common task in machine learning.

References I