Selected Topics in AI: Week 6
Autoencoders

- **Autoencoders** were first introduced with the goal of learning to **reconstruct** the input observations xi with the **lowest error possible.**
- Why would one want to learn to reconstruct the input observations?
  - An autoencoder would be an algorithm that can give as output an image that is as similar as possible to the input one.

    > **Definition 1** *An autoencoder is a type of algorithm with the primary purpose of learning an "informative" representation of the data that can be used for different applications*[a] *by learning to reconstruct a set of input observations well enough.*

  - **Another Definition:** Autoencoders are a specific type of feedforward neural networks where **the input is the same as the output.** They **compress** the input into a **lower-dimensional code** and then **reconstruct the output** from this representation. The code is a compact “summary” or “compression” of the input, also called the latent-space representation.

- **Architecture of the Autoencoders**

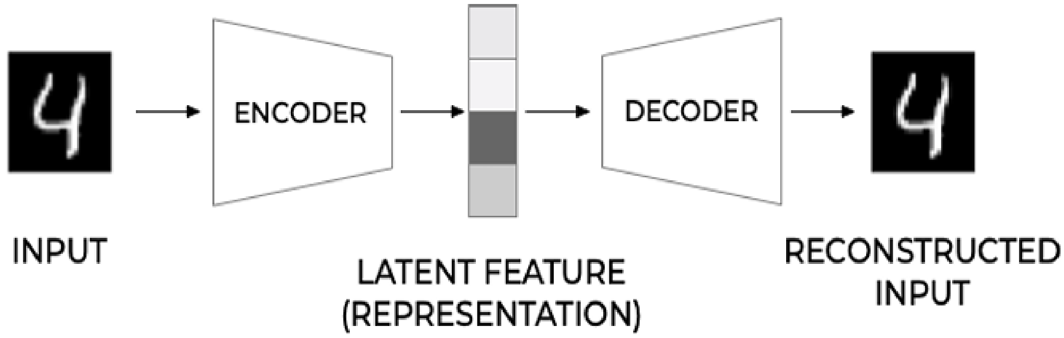

Figure 1: General structure of an autoencoder.

  - The autoencoders' main components are three:
    - an encoder,
    - a latent feature representation,
    - and a decoder.
  - Generally speaking, we want the autoencoder to **reconstruct the input well enough.**
  - Still, at the same time, it should create a latent representation (the output of the encoder part in Figure 1) that is useful and meaningful.
  - For example, latent features on hand-written digits could be the number of lines required to write each number or the angle of each line and how they connect.
  - While learning, **we extract the essential information that will allow us to solve a problem (writing digits, for example)**.
  - In most typical architectures, **the encoder and the decoder are neural networks.**
  - In general, **the encoder can be written as a function g** that will depend on some parameters.

$$\mathbf{h}_i = g(\mathbf{x}_i)$$

- Where hi ∈ Rq (the latent feature representation) is the output of the encoder block in Figure 1 when we evaluate it on the input xi.
- The decoder (and the output of the network that we will indicate with ˜ xi) can be written then as a second generic function f of the latent features.

$$\tilde{\mathbf{x}}_i = f(\mathbf{h}_i) = f(g(\mathbf{x}_i)).$$

- Training an autoencoder simply means finding the functions g(·) and f(·) that satisfy
- $$\arg\min_{f,g} < [\Delta(\mathbf{x}_i, f(g(\mathbf{x}_i))] >$$
- where Δ indicates a measure of how the input and the output of the autoencoder differ (basically our loss function will penalize the difference between input and output) and < · > indicates the average over all observations.

Deep Generative Models and Latent Variables

- **Deep Generative Modeling** is used to **model real-world data distribution.** This type of learning has many applications in synthesizing new data, anomaly detection, learning semantically rich representations of data, and more.
- There are various types of deep generative models, based on how they are trained, and how they learn the data distribution.
- Let's suppose we have a collection of human faces. For example, we have CelebA (CelebFaces Attributes) dataset. Below are some samples from the CelebA-HQ dataset.

- Now suppose looking at the above faces we want to make a new face. What will be the intuitive process? Well, we first draw a rough sketch for the structure of the face, then we decide on the facial attributes, hair color, etc, and finally, we will come up with a new face.
- So intuitively before making an image of a face, there are some **variables** in the image that are important before drawing a face.

- Such factors or variables are called **Latent Variables.**
- These variables **don't appear explicitly** but are **important in the data generation process.**
- So, when drawing a new face image, we go from a low-level representation of data **(Latent Variables)** to a high-level representation.
- Now, define the above scenario mathematically.
    - In that case, we have **high dimensional data x** ϵ X^D (face images, D-dimensional vector), and for each image, we have some **low dimensional latent variables z** ϵ Z^M (pose, face color, hairs, etc M-dimensional vector).
    - The generative process of a face image can be described as

$$z \sim P(z) \quad x \sim P(x|z) \tag{1}$$

    - The above notation denotes z and x are being sampled from respective probability distributions.
    - What it means is that we first obtain some latent vector z (for example, deciding facial attributes and all), and then we generate **a face image x based on those latent vectors.**
    - One thing to note is that both image and latent variables are sampled from **a probability distribution.**
    - All these variables have a **certain valid range,** and there are certain values for each variable that are **more probable** than others. Hence, real-world variables can be thought of as **random variables following some probability distribution.**
    - Similarly, in order to understand the **conditional distribution** above, let's say I pick a random height, for example, 174 cm, the probability of the individual being a male for that height would be higher than being a female as

generally males are taller. Hence, the occurrence of some variables affects the likelihood of other variables.

Model and the Objective

- The idea of the latent variable model is that, if we have some distribution of latent variables z and we know the conditional distribution P(x|z), we can get P(x) from the probability theory as follows:

$$P(x) = \int_z P(X, Z)dz = \int_z P(z)P(x|z)dz \qquad (2)$$

- Now P(z) is prior. P(x|z) is a distribution whose parameters can be learned to **maximize the likelihood of the data under that distribution.**
- let's take prior (P(z)) as **normal distribution N(0,1)**, and P(x|z) a **Gaussian whose parameters (mean and sigma) are learned by neural networks** as a function of z.

$$P(x|z) = \mathcal{N}(\mu_\theta(z), \Sigma_\theta(z)) \qquad (3)$$

- Hence, the marginal likelihood is now given as:

$$P(x) = \int_z P(z)P(x|z)dz = \int_z \mathcal{N}(0,1)\mathcal{N}(\mu_\theta(z), \Sigma_\theta(z))dz$$

- The objective is to maximize **P(x)** given the data x ϵ X^D (x is D dimensional). We have n data points (training points) that we assume are **independent** of each other, and the total likelihood of the dataset is given as the **product of the likelihood of each data point x^(i).**

- Normally, we maximize **log-likelihood,** which transforms the **product into summation** and the total log-likelihood is given as the **sum of the log-likelihood** of each data point x^(i):

$$\max_{\theta} \ \log P_\theta(X) = \sum_{i=1}^{n} log \int_z p_z(z) p_\theta(x^i|z) dz \qquad (4)$$

- The θ in the above equation comes from the fact, that we define P(x|z) as some distribution parameterized by θ, which can be optimized to fit the distribution to the dataset.
- Let’s first look at how to solve the above integral, once we have a solution for it, **we can derive the equation for maximizing P(x).**
- It can be inferred that the integral inside the log is not tractable (no analytical solution). Hence, we have to **numerically** integrate it. But normally, we work in high-dimensional spaces, so there is a **curse of dimensionality** and we cannot integrate it **numerically.**
- What else can we do? We can rewrite the above equation as **an expectation concerning P(z).**

$$\max_{\theta} \ \log P_\theta(X) = \sum_{i=1}^{n} log \ \mathbb{E}_{z\sim P(z)}[p_\theta(x^i|z)] \qquad (5)$$

-

Latent Variable Models

- Generative models share characteristics. They are often unsupervised and don’t make use of observed inputs.
- Generative models have multiple uses. They can estimate density or map between domains.
- There are at least three classes of generative models:
  - Autoregressive models
  - Generative adversarial models
  - **Latent variable models.**
- Latent variable models understand **what causes data.** They introduce an **unobserved latent variable z** and a **prior p(z).**
- They specify a likelihood $p(x|z)$ from which observations can be generated — $z \sim p(z)$, $x \sim p(x|z)$. This defines a joint distribution $p(x,z) = p(x|z) * p(z)$.