Generative Models

Lecture 4

Energy Based Models

Suppose again we have a data set

$$\mathcal{D} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$$

drawn from an unknown data distribution $p_{data}(\mathbf{x})$.

The goal of generative modeling is to fit a model of the data distribution such that we can synthesize new data points by sampling from the model.

In order to build a model we need a way to represent a probability density function

A common way to represent a density function is to write it as

$$p_\theta(\mathbf{x}) = \frac{\exp(-f_\theta(\mathbf{x}))}{\int \exp(-f_\theta(\mathbf{x}))d\mathbf{x}}$$

where $f_\theta : \mathbb{R}^d \to \mathbb{R}$ is some function such that the integral

$$Z_\theta = \int_{\mathbb{R}^d} \exp(-f_\theta(\mathbf{t}))d\mathbf{t}$$

is finite. The function f_θ is often called the *Energy Function* and Z_θ the *Partition Function*

The numerator is always positive and dividing by the integral assures that

$$\int p_\theta(\mathbf{x}) d\mathbf{x} = 1$$

so the expression satisfies the conditions for being a valid probability density function (pdf).

Any density function p can be represented this way by taking

$$f(\mathbf{x}) = -\log p(\mathbf{x})$$

In this case the partition function Z is obviously $= 1$

Typically the partition function Z_θ is intractable so to estimate parameters we have to figure out a way to get rid of it

We estimate parameters by Maximum Likelihood i.e. by computing

$$\operatorname*{argmax}_\theta p_\theta(\mathcal{D}) = \operatorname*{argmax}_\theta \prod_{\mathbf{x}\in\mathcal{D}} p_\theta(\mathbf{x})$$

or equivalently

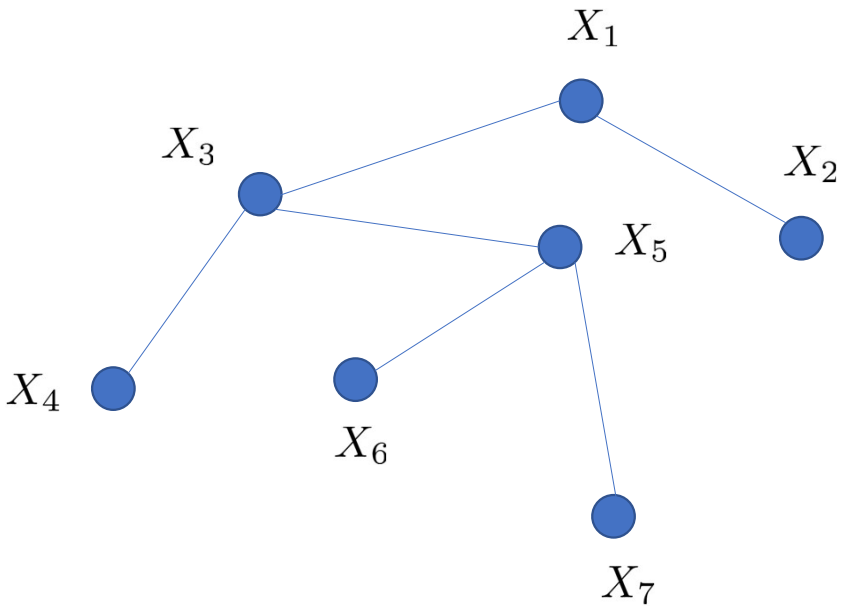
$$\operatorname*{argmax}_\theta \sum_{\mathbf{x}\in\mathcal{D}} \log p_\theta(\mathbf{x})$$

For an energy-based model this would be

$$\underset{\theta}{\operatorname{argmax}} \sum_{\mathbf{x} \in \mathcal{D}} - f_\theta(\mathbf{x}) - \log Z_\theta$$

Commonly energy-based models are used with *Probabilistic Graphical Models (PGM)*

A *Markov Network* (also called a *Markov Random Field*) is an undirected graph where the nodes are random variables



We are interested in the joint distribution

$$p(X_1, X_2, \ldots, X_7)$$

Since there are no particular order and direction in the graph, ordinary conditional distributions are not so useful. Instead we shall use a concept called *factors*

If $\mathbb{D} = \{X_i\}_{\imath \in I}$ is a set of random variables, a factor is a function $\phi(\{X_i\}_{i \in I})$ from the set of values $val(\mathbb{D}) = \prod val(X_i) \to \mathbb{R}$. For instance if each X_i takes values in $\mathbb{R}$, $val(\mathbb{D}) = \mathbb{R}^I$. If ϕ only takes positive values we say it is a positive factor.

A common way to make a factors is from an *Energy* function

$$E : val(\mathbb{D}) \to \mathbb{R}$$

and then define a factor by $\phi = e^{-E}$.

Given factors $\phi_1(X, Y)$ and $\phi_2(Y, Z)$ we define their product $\phi_3(X, Y, Z)$ as the function

$$\phi_3(x, y, z) = \phi_1(x, y)\phi_2(y, z)$$

If the factors are defined by energy functions this corresponds to

$$E_3(x, y, z) = E_1(x, y) + E_2(y, z)$$

A *clique* is a complete subgraph i.e. one where any two vertices are connected.

A maximal clique is a clique which cannot be made larger by adding any vertex.

Thus for our graph the maximal cliques are: $\mathbb{D}_1 = \{X_1, X_2\}$, $\mathbb{D}_2 = \{X_1, X_3\}$, $\mathbb{D}_3 = \{X_3, X_4\}$, $\mathbb{D}_4 = \{X_3, X_5, X_6\}$, $\mathbb{D}_5 = \{X_5, X_6, X_7\}$

A clique factorization is the product of factors, one for each maximal clique

A *Gibbs Distribution* P, parametrized by a set of factors $\{\phi_1(\mathbb{D}_1), \phi_2(\mathbb{D}_2), \ldots, \phi_K(\mathbb{D}_K)\}$ is a distribution defined by

$$P(X_1, X_2, \ldots, X_m) = \frac{1}{Z}\phi_1(\mathbb{D}_1) \cdot \phi_2(\mathbb{D}_2) \cdot \cdots \cdot \phi_K(\mathbb{D}_K)$$

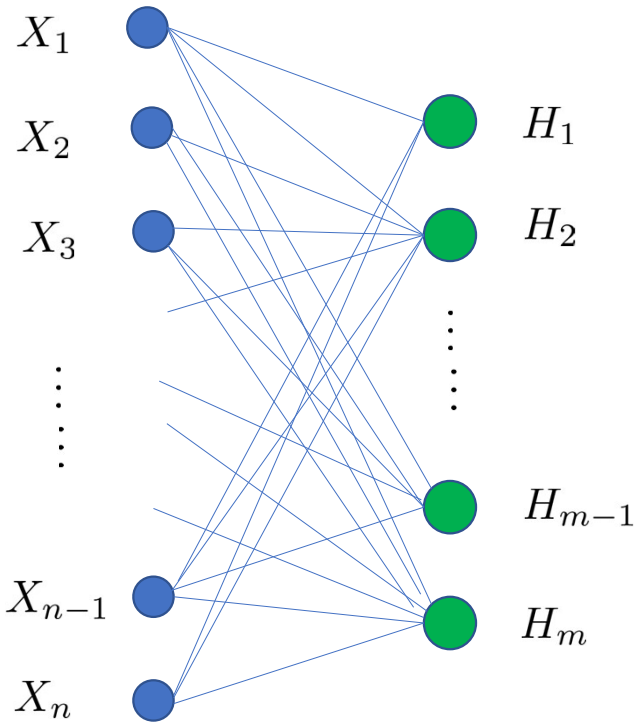
where

$$Z = \int \phi_1(\mathbb{D}_1) \cdot \phi_2(\mathbb{D}_2) \cdot \cdots \cdot \phi_K(\mathbb{D}_K)$$

In our case the Gibbs Distribution of the Markov Network is

$$P(X_1, X_2, \ldots, X_7) = \frac{\phi_1(X_1, X_2)\phi_2(X_1, X_3)\phi_3(X_3, X_4)\phi_4(X_3, X_5, X_6)\phi_5(X_5, X_6, X_7)}{Z}$$

Restricted Boltzmann Machine



This is a *bi-partite* graph. The vertices on the left are called the *visible* nodes and those on the right the *hidden* or *latent* nodes. There are no conncections between two hidden nodes or between two visible nodes. Each visible node is connected to all the hidden nodes and each hidden node is connected to all the visible nodes.

The maximal cliques are $\{X_i, H_j\}_{i\in I, j\in J}$ so the Gibbs factorization is

$$P(\{X_i\}_{i\in I}, \{H_j\}_{j\in J}) = \prod_{i,j} P(X_i, H_j)$$

We require that the X_i and the H_j are all Bernoulli distributions i.e. they only take values 0 and 1. We can replace all integrals by sums

We have

$$P(\mathbf{X}|\mathbf{H}) = \prod_i P(X_i|\mathbf{H})$$

and

$$P(\mathbf{H}|\mathbf{X}) = \prod_j P(H_j|\mathbf{X})$$

We define an energy function E as follows: let $\mathbf{W}$ be an $m \times n$ matrix and let $\mathbf{b}$ and $\mathbf{c}$ be *bias vectors*. We let θ denote the parameter set $\{\mathbf{W}, \mathbf{b}, \mathbf{c}\}$

We define

$$E_\theta(x_i, h_j) = -x_i \, w_{ij} \, h_j - b_i x_i - c_j h_j$$

and the factor

$$\phi(X_i, H_j) = \exp(-E(x_i, h_j)) = \exp(x_i \, w_{ij} \, h_j + b_i x_i + c_j h_j)$$

Given a sample $\mathbf{x}$ of $\mathbf{X} = (X_1, X_2, \ldots, X_n)$ and a sample $\mathbf{h}$ of the latent variable $\mathbf{H} = (H_1, H_2, \ldots, H_m)$ we can write the total energy

$$E(\mathbf{x}, \mathbf{h}) = -\mathbf{x} \cdot \mathbf{W} \cdot \mathbf{h}^T - \mathbf{b} \cdot \mathbf{x} - \mathbf{c} \cdot \mathbf{h}$$

Thus the probability of a sample $(\mathbf{x}', \mathbf{h}')$ is

$$\frac{\exp(-E(\mathbf{x}', \mathbf{h}'))}{\sum_{\mathbf{x},\mathbf{h}} \exp(-E(\mathbf{x}, \mathbf{h}))}$$

$$P_\theta(\mathbf{h}|\mathbf{x}) = \frac{P(\mathbf{h},\mathbf{x})}{P(\mathbf{x})} = \frac{\frac{1}{Z}\exp(-E_\theta(\mathbf{h},\mathbf{x}))}{\frac{1}{Z}\sum_{\mathbf{h}}\exp(-E_\theta(\mathbf{x},\mathbf{h}))} = \frac{\exp(-E_\theta(\mathbf{h},\mathbf{x}))}{\sum_{\mathbf{h}}\exp(-E_\theta(\mathbf{x},\mathbf{h}))}$$

Now

$$\begin{aligned} -E_\theta(h_s = 1, \mathbf{x}) &= \sum_r x_r \, w_{rs} \, h_s + \mathbf{x}\cdot\mathbf{b} + c_s \\ &= \mathbf{x}\cdot\mathbf{W}_{|s} + \mathbf{x}\cdot\mathbf{b} + c_s \end{aligned}$$

In particular

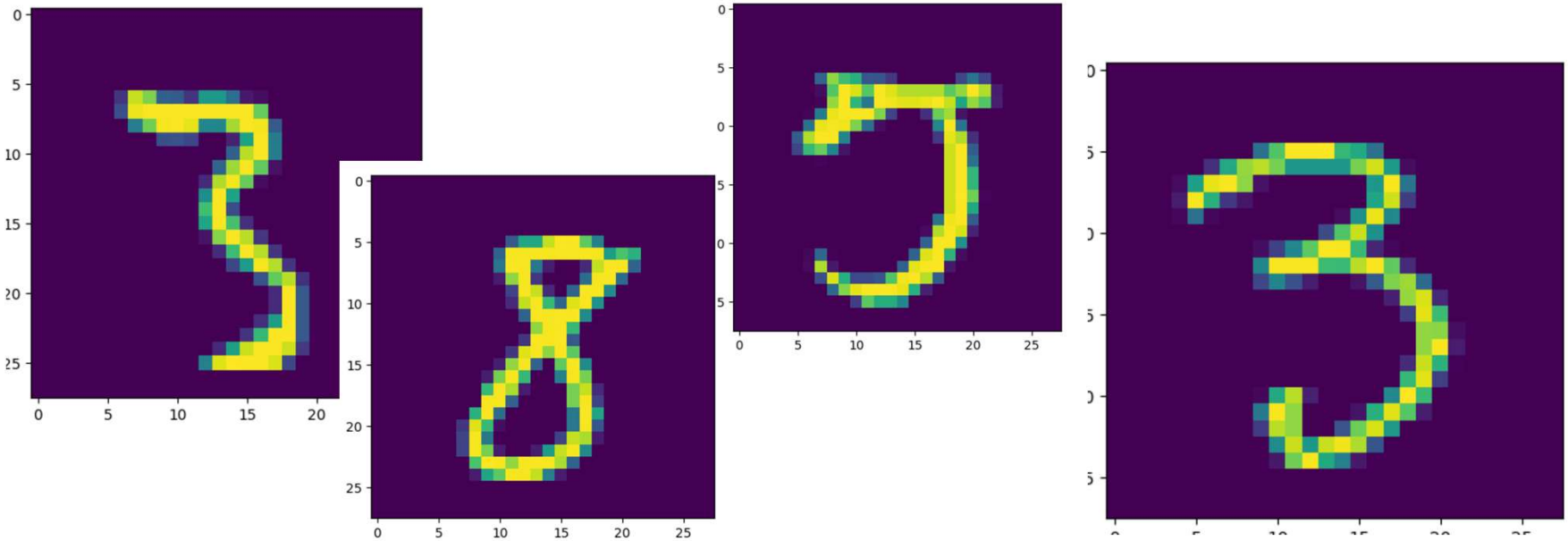
$$\begin{aligned} P(h_s = 1|\mathbf{x}) &= \frac{\exp(\mathbf{x}W_{|s} + \mathbf{x}\cdot\mathbf{b} + c_s)}{\exp(\mathbf{x}\cdot\mathbf{b}) + \exp(\mathbf{x}W_{|s} + \mathbf{x}\cdot\mathbf{b} + c_s)} \\ &= \frac{1}{1 + \exp(-\mathbf{x}W_{|s} - c_s)} \\ &= \sigma(\mathbf{x}W|s + c_s) \end{aligned}$$

Similarly

$$P_\theta(x_r = 1|\mathbf{h}) = \sigma(W_{-r} \cdot \mathbf{h} + b_r)$$

It follows that $P_\theta(h_s|\mathbf{x})$ is Bernoulli with parameter $\sigma(\mathbf{x} \cdot W_{|s} + c_s)$ and $P_\theta(x_r|\mathbf{h})$ is Bernoulli with parameter $\sigma(W_{-r} \cdot \mathbf{h} + b_r)$

For our dataset we shall use a set of images of handwritten digits $0, 1, 2, \ldots, 9$ known as the *MNIST* dataset



There are $60,000$ of these images which are 28×28 pixels black-and-white.

We can view them as $784 = 28 \times 28$ arrays of 0's and 1's (this is not quite corrects, the actual pixel values are between 0 and 1 so we will view the pixel value as the parameter of a Bernoulli distribution).

Thus we can view an image as a sample of a 784 dimensional random variable

$$\mathbf{X} = (X_1, X_2, \ldots, X_{784})$$

where each X_i is a Bernoulli random variable.

These Bernoulli variable obviously cannot be independent, otherwise the samples from $\mathbf{X}$ would just be random 28×28 arrays of 0's and 1's.

Instead we shall view them as the visible nodes in a Restricted Bolzmann machine where the hidden nodes $H_1, H_2, \ldots, H_m$ are also Bernoulli random variables.

Thus a sample from $H_1, H_2, \ldots, H_m$ would be a sequence of 0's and 1's and any sample from the hidden layer would produce

$$P(\mathbf{X}|H_1 = h_1, H_2 = h_2, \ldots, H_m = h_m) = \prod P(X_r|\mathbf{H} = \mathbf{h})$$

Let our dataset be $\mathcal{D} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$. We assume that these points are samples from some unknown distribution

To approximate the unknown distribution with a distribution P_θ we want the log likelihood $\log P_\theta(\mathbf{x})$ to be high if $\mathbf{x}$ is in $\mathcal{D}$ and low if $\mathbf{x}$ is a random point

$$
\begin{aligned}
\mathcal{L}(\widetilde{\mathbf{x}}) &= \log P_\theta(\widetilde{\mathbf{x}}) \\
&= \log \sum_{\mathbf{h}} P_\theta(\widetilde{\mathbf{x}}, \mathbf{h}) \text{ marginalizing} \\
&= \log \sum_{\mathbf{h}} \frac{\exp(-E_\theta(\widetilde{\mathbf{x}}, \mathbf{h}))}{Z} \\
&= \log \sum_{\mathbf{h}} \exp(-E_\theta(\widetilde{\mathbf{x}}, \mathbf{h})) - \log Z \\
&= \log \sum_{\mathbf{h}} \exp(-E_\theta(\widetilde{\mathbf{x}}, \mathbf{h})) - \log \sum_{\mathbf{x}, \mathbf{h}} \exp(-E(\mathbf{x}, \mathbf{h}))
\end{aligned}
$$

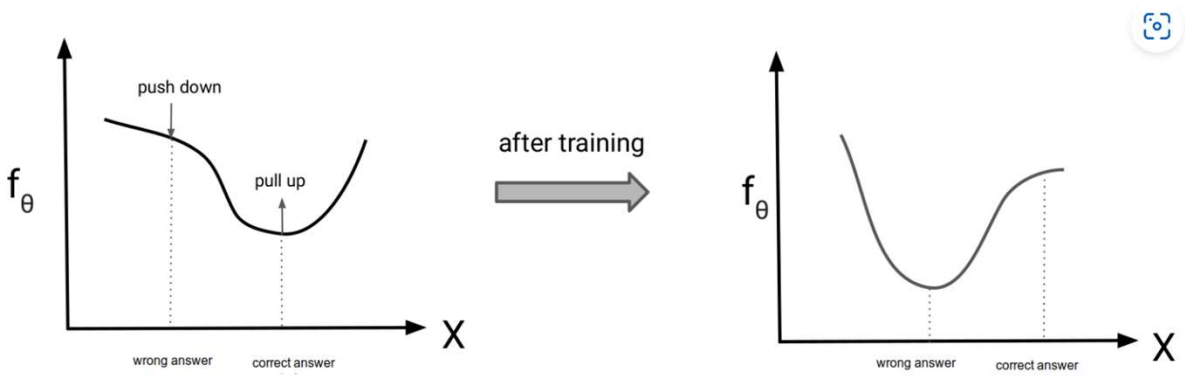
So we want

$$\log \sum_{\mathbf{h}} \exp(-E_\theta(\widetilde{\mathbf{x}}, \mathbf{h})) \text{ to be large for } \widetilde{\mathbf{x}} \in \mathcal{D}$$

and

$$\log \sum_{\mathbf{x},\mathbf{h}} \exp(-E(\mathbf{x}, \mathbf{h})) \text{ to be small for random } \mathbf{x}, \mathbf{h}$$

Contrastive Divergence



Gibbs Sampling

We can sample from the joint distribution $P_\theta(\mathbf{X}, \mathbf{H})$ as follows

Start with a data point $\widetilde{\mathbf{x}} = (\widetilde{x}_1, \widetilde{x}_2, \ldots, \widetilde{x}_{784})$, the $\widetilde{x}_j$ are 0 or 1

The distribution

$$\begin{aligned} &P_\theta(H_1, H_2, \ldots, H_n | X_1 = \widetilde{x}_1, X_2 = \widetilde{x}_2, \ldots) \\ &= \prod_j P_\theta(H_j | X_1 = \widetilde{x}_1, X_2 = \widetilde{x}_2, \ldots) \end{aligned}$$

and each of the distributions

$$P_\theta(H_j | X_1 = \widetilde{x}_1, X_2 = \widetilde{x}_2, \ldots)$$

are Bernoulli with parameter

$$\sigma(\widetilde{\mathbf{x}} W | j + c_j)$$

Now we can sample $\mathbf{h} = (h_1, h_2, \ldots, h_m)$ with each

$$h_j \sim Bernoulli(\sigma(\widetilde{\mathbf{x}}\mathbf{W}|j + c_j))$$

The distribution

$$\begin{aligned} &P_\theta(X_1, X_2, \ldots, X_{784}|H_1 = h_1, H_2 = h_2, \ldots, H_m = h_m) \\ &= \prod_i P_\theta(X_i|H_1 = h_1, H_2 = h_2, \ldots, H_m = h_m) \end{aligned}$$

and each of the distributions in the product are Bernoulli with parameter

$$\sigma(\mathbf{W}_{-i} \cdot \mathbf{h} + b_i)$$

Now sample $\mathbf{x} = (x_1, x_2, \ldots, x_{784})$ with each

$$x_i \sim Bernoulli(\sigma(W_{-i} \cdot \mathbf{h} + b_i))$$

Continuing this way $(\mathbf{x}_m, \mathbf{h}_{m-1})$ will be a sample from $P_\theta(\mathbf{X}, \mathbf{H})$ for m sufficiently large

To maximize the log-likelihood with respect to the parameters $\theta = (\mathbf{W}, \mathbf{b}, \mathbf{c})$ we need to compute the gradients)

$$
\begin{aligned}
\nabla_\theta \mathcal{L}(\widetilde{\mathbf{x}}) &= \nabla_\theta \log \sum_{\mathbf{h}} \exp(-E_\theta(\widetilde{\mathbf{x}}, \mathbf{h})) - \nabla_\theta \log \sum_{\mathbf{x},\mathbf{h}} \exp(-E_\theta(\mathbf{x}, \mathbf{h})) \\
&= \frac{\nabla_\theta \sum_{\mathbf{h}} \exp(-E_\theta(\widetilde{\mathbf{x}}, \mathbf{h}))}{\sum_{\mathbf{h}} \exp(-E_\theta(\mathbf{x}_i, \mathbf{h}))} - \frac{\nabla_\theta \sum_{\mathbf{x},\mathbf{h}} \exp(-E_\theta(\mathbf{x}, \mathbf{h}))}{\sum_{\mathbf{x},\mathbf{h}} \exp(-E_\theta(\mathbf{x}, \mathbf{h}))} \\
&= \frac{\sum_{\mathbf{h}} - \exp(-E_\theta(\widetilde{\mathbf{x}}, \mathbf{h})) \nabla_\theta E_\theta(\widetilde{\mathbf{x}}, \mathbf{h})}{\sum_{\mathbf{h}} \exp(-E_\theta(\widetilde{\mathbf{x}}, \mathbf{h}))} \\
&- \frac{\sum_{\mathbf{x},\mathbf{h}} - \exp(-E_\theta(\mathbf{x}, \mathbf{h})) \nabla_\theta E_\theta(\mathbf{x}, \mathbf{h})}{\sum_{\mathbf{x},\mathbf{h}} \exp(-E_\theta(\mathbf{x}, \mathbf{h}))}
\end{aligned}
$$

Now

$$\sum_{\mathbf{h}} \frac{-\exp(-E_\theta(\widetilde{\mathbf{x}}, \mathbf{h}))\nabla_\theta E_\theta(\widetilde{\mathbf{x}}, \mathbf{h})}{\sum_{\mathbf{h}} \exp(-E_\theta(\widetilde{\mathbf{x}}, \mathbf{h}))}$$

the sums are over all possible values of $\mathbf{H}$(in all 2^m terms)

$$= \sum_{\mathbf{h}} \frac{-\exp(-E_\theta(\widetilde{\mathbf{x}}, \mathbf{h}))}{\sum_{\mathbf{h}} \exp(-E_\theta(\widetilde{\mathbf{x}}, \mathbf{h}))} \nabla_\theta E_\theta(\widetilde{\mathbf{x}}, \mathbf{h})$$

$$= -\sum_{\mathbf{h}} P(\mathbf{h}|\widetilde{\mathbf{x}}) \nabla_\theta E_\theta(\mathbf{h}, \widetilde{\mathbf{x}})$$

$$= -\mathbb{E}_{P(\mathbf{H}|\mathbf{X}=\widetilde{\mathbf{x}})}(\nabla_\theta E_\theta(\widetilde{\mathbf{x}}, \mathbf{h}))$$

and

$$-\sum_{\mathbf{x},\mathbf{h}} \frac{-\exp(-E_\theta(\mathbf{x}, \mathbf{h}))\nabla_\theta E_\theta(\mathbf{x}, \mathbf{h})}{\sum_{\mathbf{x},\mathbf{h}} \exp(-E_\theta(\mathbf{x}, \mathbf{h}))} = \mathbb{E}_{P(\mathbf{X},\mathbf{H})}(\nabla_\theta E_\theta(\mathbf{x}, \mathbf{h})))$$

So the gradient of the log-likelihood

$$\nabla_\theta \mathcal{L}(\widetilde{\mathbf{x}}) = -\mathbb{E}_{P(\mathbf{H}|\mathbf{X}=\widetilde{\mathbf{x}})}(\nabla_\theta E_\theta(\widetilde{\mathbf{x}}, \mathbf{h})) + \mathbb{E}_{P(\mathbf{X},\mathbf{H})}(\nabla_\theta E_\theta(\mathbf{x}, \mathbf{h}))$$

and

$$\nabla_\theta \mathcal{L}(\mathcal{D}) = -\sum_{\widetilde{\mathbf{x}} \in \mathcal{D}} \mathbb{E}_{P(\mathbf{H}|\mathbf{X}=\widetilde{\mathbf{x}})}(\nabla_\theta E_\theta(\widetilde{\mathbf{x}}, \mathbf{h})) + n\mathbb{E}_{P(\mathbf{X},\mathbf{H})}(\nabla_\theta E_\theta(\mathbf{x}, \mathbf{h}))$$

Now the gradients are expressed as expectations so we can use Monte Carlo to estimate them and then do gradient descent

Let P_{data} denote the true distribution of the data then

$$\mathbb{E}_{P_{data}(\mathbf{x})}(\mathbb{E}_{P_\theta(\mathbf{h}|\mathbf{x})}(\nabla_\theta E(\mathbf{x},\mathbf{h}))) \equiv \frac{1}{N}\sum_{\mathbf{x}\in\mathcal{D}}\mathbb{E}_{P(\mathbf{h}|\mathbf{x})}(\nabla_\theta E(\mathbf{x},\mathbf{h}))$$

Starting with the $\mathbf{x} \in \mathcal{D}$ and applying the Gibbs sampler sufficiently many times to get a pair

$$(\mathbf{x}', \mathbf{h}') \sim P_\theta$$

Thus

$$\mathbb{E}_{P(\mathbf{x},\mathbf{h})}(\nabla_\theta E(\mathbf{x}, \mathbf{h})) \equiv \frac{1}{N} \sum_{\mathbf{x} \in \mathcal{D}} \nabla_\theta E(\mathbf{x}', \mathbf{h}') \equiv \mathbb{E}_{P_{data}}(\nabla_\theta E(\mathbf{x}', \mathbf{h}'))$$

It follows that

$$\frac{1}{N} \nabla_\theta \mathcal{L}(\mathcal{D}) \equiv \mathbb{E}_{P_{data}(\mathbf{x})}(\mathbb{E}_{P_\theta(\mathbf{h}|\mathbf{x})}(\nabla_\theta E(\mathbf{x}, \mathbf{h}))) - \mathbb{E}_{P_{data}}(\nabla_\theta E(\mathbf{x}', \mathbf{h}'))$$

In practice it turns out that it is enough to take a few or even just one, Gibbs sampler step (this is *Contrastive Divergence*), so approximately

$$-\nabla_\theta \mathcal{L}(\mathcal{D}) \equiv \sum_{\mathbf{x}\in\mathcal{D}} \sum_{\mathbf{h}} P_\theta(\mathbf{h}|\mathbf{x}) \nabla_\theta E(\mathbf{x},\mathbf{h}) - \sum_{\mathbf{x}\in\mathcal{D}} \nabla_\theta E(\mathbf{x}',\mathbf{h}'))$$

We have

$$\begin{aligned}
\log \sum_{\mathbf{h}} E(\mathbf{x},\mathbf{h}) &= \log \sum_{\mathbf{h}} \exp(\mathbf{b}\cdot\mathbf{x} + \mathbf{c}\cdot\mathbf{h} + \mathbf{x}^T\cdot\mathbf{W}\cdot\mathbf{h}) \\
&= \mathbf{b}\cdot\mathbf{x} + \log \sum_{\mathbf{h}} \exp(\sum_j (c_j + \mathbf{x}^T\cdot\mathbf{W}_j) h_j) \\
&= \mathbf{b}\cdot\mathbf{x} + \log \sum_{\mathbf{h}} \prod_j \exp((c_j + \mathbf{x}^T\cdot\mathbf{W}_j) h_j) \\
&= \mathbf{b}\cdot\mathbf{x} + \log \prod_j \sum_{h_j} \exp((c_j + \mathbf{x}^T\cdot\mathbf{W}_j) h_j)
\end{aligned}$$

But $h_j = 0$ or 1 so

$$\sum \exp((c_j + \mathbf{x}^T\cdot\mathbf{W}_j) h_j) = 1 + \exp(c_j + \mathbf{x}^T\cdot\mathbf{W}_j)$$

It follows that

$$\log \sum_{\mathbf{h}} E(\mathbf{x}, \mathbf{h}) = \mathbf{b} \cdot \mathbf{x} + \sum_{j} \log(1 + \exp(c_j + \mathbf{x} \cdot \mathbf{W}_j)$$

so we can write

$$\begin{aligned}
\log P(\mathbf{x}) &= \log \sum_{\mathbf{h}} P(\mathbf{x}, \mathbf{h}) \\
&= \log \sum_{\mathbf{h}} \frac{E(\mathbf{x}, \mathbf{h})}{Z} \\
&= \log \sum_{\mathbf{h}} \frac{E(\mathbf{x}, \mathbf{h})}{-} \log Z \\
&= \mathbf{b} \cdot \mathbf{x} + \sum_{j} \log(1 + \exp(c_j + \mathbf{x} \cdot \mathbf{W}_j)) - \log Z
\end{aligned}$$

The expression

$$\mathcal{F}(\mathbf{x}) = -\log \sum_{\mathbf{h}} E(\mathbf{x}, \mathbf{h})$$

is known as the free energy

Let $\mathbf{x} \in \mathcal{D}$ be a data point and let $\mathbf{x}'$ be the first step in the Gibbs Sampler applied to $\mathbf{x}$.

Contrastive Divergence uses

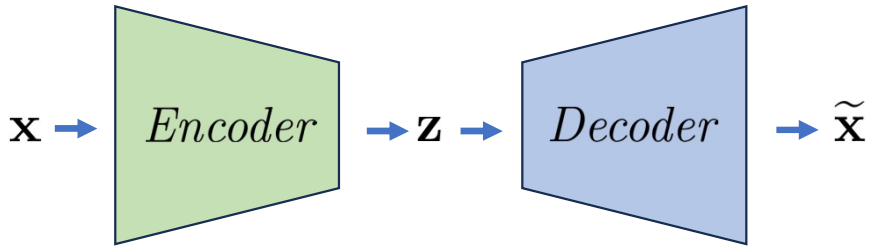
$$\mathcal{F}(\mathbf{x}) - \mathcal{F}(\mathbf{x}')$$

as an approximation to $\mathcal{L}(\mathbf{x})$ and so we can use this difference as a loss function to maximize the Maximum Likelihood

Variational Autoencoders

An Autoencoder has two parts, an *Encoder* and a *Decoder*. In the Encoder part the model outputs from a data point $\mathbf{x}$, a latent vector $\mathbf{z} = Enc(\mathbf{z})$. Typically $\dim \mathbf{z} < \dim \mathbf{x}$ so we can view $\mathbf{z}$ as a compressed version of $\mathbf{x}$.

The Decoder part inputs the latent $\mathbf{z}$ and outputs a vector $Dec(\mathbf{z}) = \widetilde{\mathbf{x}}$ with $\dim \widetilde{\mathbf{x}} = \dim \mathbf{x}$ and the model is trained so that $\widetilde{\mathbf{x}}$ is close to the original data point $\mathbf{x}$.



Thus an Autoencoder can be thought of as a way to compress the original data to lower dimension such that the original data can be approximately reconstructed from the compressed version.

Autoencoders are quite simple to code

The Encoder has 3 linear layers

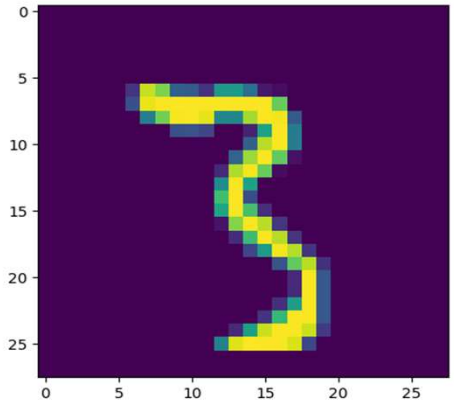
The Decoder also has 3 linear layers and at the end a sigmoid which squeezes the the output between 0 and 1

```
Encoder = nn.Sequential(
            nn.Linear(784,256),
            nn.ReLU(),
            nn.Linear(256,64),
            nn.Tanh(),
            nn.Linear(64,comp_dim))

Decoder = nn.Sequential(
            nn.Linear(comp_dim,64),
            nn.Tanh(),
            nn.Linear(64,256),
            nn.Tanh(),
            nn.Linear(256,784),
            nn.Sigmoid()
)

```

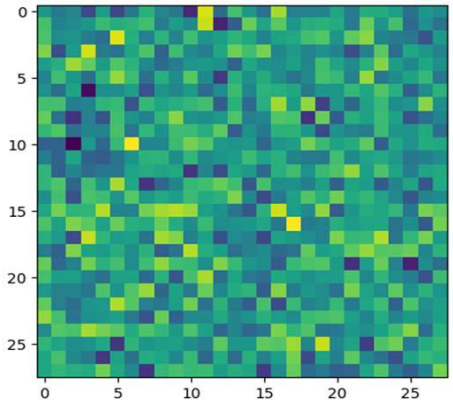
```
AutoEncoder = nn.Sequential(
                    Encoder,
                    Decoder )
```

Before Training

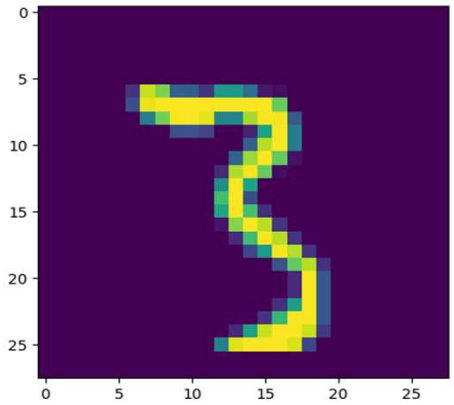
Input



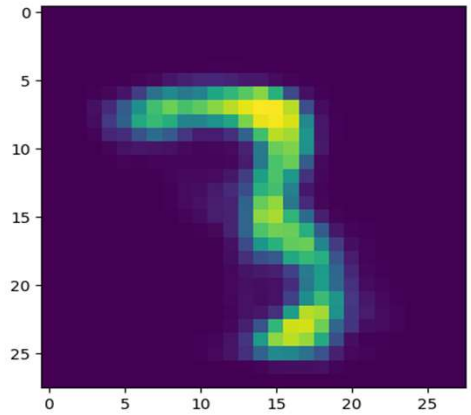
Output



After Training



8-dimensional latent tensor



```
z = Encoder(x)
z
```

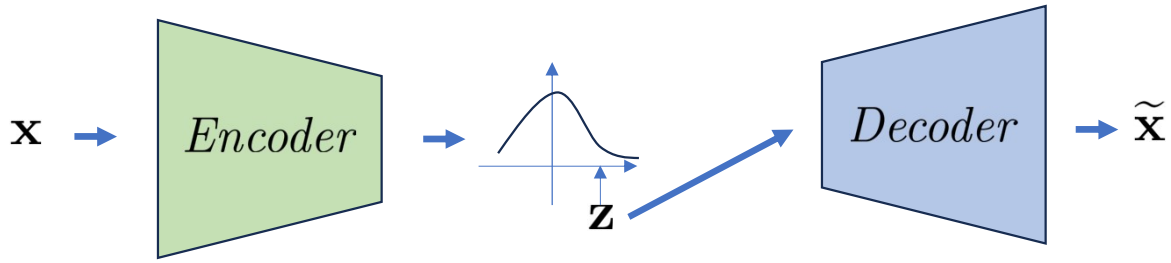
```
tensor([[[ 0.4082,  1.0688,  1.3389,  1.7421,  2.7509, -1.2342, -0.2108,
           1.2860]]], grad_fn=<ViewBackward0>)
```

The encoder compressed the input from a 728-dimensional vector to an 8-dimensional latent vector and the decoder was able to take the latent vector and reproduce a recognizable version of the input

The *Variational Autoencoder* also consist of an *Encoder* and a *Decoder*

But instead of the Encoder producing a latent vector, it now outputs a *distribution*

The Decoder takes a *sample* from this distribution and tries to reproduce the input



The Encoder, instead of mapping an input $\mathbf{x}$ to a latent vector $\mathbf{z}$, maps the input to a distribution $q_\phi(\mathbf{z}|\mathbf{x})$ (here ϕ represents the parameters of the Encoder network)

The Decoder maps a sample

$$\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})$$

to a distribution $p_\theta(\mathbf{x}|\mathbf{z})$

$$
\begin{aligned}
\log p_\theta(\mathbf{x}) &= \log p_\theta(\mathbf{x}) \int q_\phi(\mathbf{z}|\mathbf{x}) d\mathbf{z} = \int \log p_\theta(\mathbf{x}) q_\phi(\mathbf{z}|\mathbf{x}) d\mathbf{z} \\
&= \int \log \frac{p_\theta(\mathbf{x}) p_\theta(\mathbf{z}|\mathbf{x})}{p_\theta(\mathbf{z}|\mathbf{x})} q_\phi(\mathbf{z}|\mathbf{x}) d\mathbf{z} = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \left(\log \frac{p_\theta(\mathbf{x}, \mathbf{z})}{p_\theta(\mathbf{z}|\mathbf{x}))} \right) \\
&= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \left(\log \frac{p_\theta(\mathbf{x}, \mathbf{z}) q_\phi(\mathbf{z}|\mathbf{x})}{p_\theta(\mathbf{z}|\mathbf{x}) q_\phi(\mathbf{z}|\mathbf{x})} \right) \\
&= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \left(\log \frac{p_\theta(\mathbf{x}, \mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})} \right) + \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \left(\log \frac{q_\phi(\mathbf{z}|\mathbf{x})}{p_\theta(\mathbf{z}|\mathbf{x})} \right)
\end{aligned}
$$

The second term we can write

$$\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left(\log \frac{q_\phi(\mathbf{z}|\mathbf{x})}{p_\theta(\mathbf{z}|\mathbf{x})}\right) = -\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left(\log \frac{p_\theta(\mathbf{z}|\mathbf{x})}{q_\phi(\mathbf{z}|\mathbf{x})}\right) = D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z}|\mathbf{x}))$$

It follows that we can express the log-likelihood as

$$\log p_\theta(\mathbf{x}) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left(\log \frac{p_\theta(\mathbf{x},\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})}\right) + D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z}|\mathbf{x}))$$

The first term $\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left(\log \frac{p_\theta(\mathbf{x},\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})}\right)$ is precisely the $ELBO$

Since the KL-divergence is always ≥ 0 we have

$$\log p_\theta(\mathbf{x}) \geq ELBO$$

so we will want to maximize the $ELBO$ i.e. to find

$$\underset{\theta,\phi}{\operatorname{argmax}}\, \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left(\log \frac{p_\theta(\mathbf{x},\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})}\right)$$

Rewriting

$$p_\theta(\mathbf{x}, \mathbf{z}) = p_\theta(\mathbf{x}|\mathbf{z})p(\mathbf{z})$$

where $p(\mathbf{z})$ is some prior distribution of $\mathbf{z}$ which we typically choose as $\mathcal{N}(0, I)$, we get

$$\begin{aligned}
ELBO &= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}(\log p_\theta(\mathbf{x}|\mathbf{z})) - \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}(q_\phi(\mathbf{z}|\mathbf{x})) + \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}(\log p(\mathbf{z})) \\
&= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}(\log p_\theta(\mathbf{x}|\mathbf{z})) + (\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}(\log p(\mathbf{z})) - \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}(\log q_\phi(\mathbf{x}|\mathbf{z}))) \\
&= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}(\log p_\theta(\mathbf{x}|\mathbf{z})) + \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left(\log \frac{\log p(\mathbf{z})}{\log q_\phi(\mathbf{z}|\mathbf{x})}\right) \\
&= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}(\log p_\theta(\mathbf{x}|\mathbf{z})) - D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z}))
\end{aligned}$$

The $\log p_\theta(\mathbf{x}|\mathbf{z})$ can be viewed as a reconstruction log likelihood dependent on a specific value of the latent variable $\mathbf{z}$ and so we want to maximize the expectation over all the values of $\mathbf{z}$

$$\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}(\log p_\theta(\mathbf{x}|\mathbf{z}))$$

The KL-divergence term

$$D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z}))$$

which we want to minimize can be viewed as a regularizer which tries to keep the latent distribution $q_\phi(\mathbf{z}|\mathbf{x})$ close to the prior $p(\mathbf{z})$

This is exactly the same computation as our earlier analysis of approximating the posterior distribution by a variational distribution.

Here $q_\phi(\mathbf{z}|\mathbf{x})$ is the variational distribution and $p_\theta(\mathbf{x}|\mathbf{z})$ the likelihood

We saw in lecture 2 that

$$\log p_\theta(\mathbf{x}) \leq ELBO = \mathbb{E}_{q_\phi(\mathbf{x}|\mathbf{z})}(\log p_\theta(\mathbf{x}|\mathbf{z})p(\mathbf{z}) - \log q_\phi(\mathbf{z}|\mathbf{x}))$$

Home Work 2

Problem 1.

Write an Encoder network with 3 linear layers that inputs MNIST images and outputs an 8-dimensional vector of means and an 8-dimensional positive vector of variances

Problem 2.

Write code that inputs an 8-dimensional vector μ and an 8-dimensional positive vector σ and a sample ε from an 8-dimensional $\mathcal{N}(0, I)$ to generate a sample from $\mathcal{N}(\mu, \sigma I)$

Problem 3 Use your code from Problem 2. to write a Decoder network that takes a sample from $\mathcal{N}(\mu, \sigma I)$ (an 8-dimensional vector). Use 3 linear layers and ReLU activation functions and sigmoid function to output a 784-dimensional tensor, resphape it to 28×28 and plot the resulting image.

Problem 4. Put the Encoder and Decoder together to write a Variational Autoencoder and train it on the MNIST dataset.

Fix an input image and use the output of the Encoder to make a μ and a σ. Make 10 samples from $\mathcal{N}(\mu, \sigma I)$ and plot the output of the Decoder on these samples.