Energy Based Models

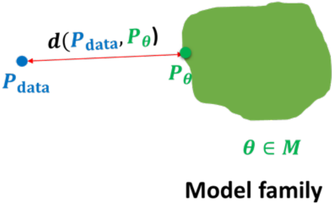
Stefano Ermon, Aditya Grover

Stanford University

Lecture 11

Summary

$\mathbf{x}_i \sim P_{\text{data}}$
$i = 1, 2, \dots, n$



Story so far

- Representation: Latent variable vs. fully observed
- Objective function and optimization algorithm: Many divergences and distances optimized via likelihood-free (two sample test) or likelihood based methods

Plan for today: Energy based models

Likelihood based learning

Probability distributions $p(x)$ are a key building block in generative modeling. Properties:

1. non-negative: $p(x) \geq 0$
2. sum-to-one: $\sum_x p(x) = 1$ (or $\int p(x)dx = 1$ for continuous variables)

Sum-to-one is key:

Total "volume" is fixed: increasing $p(x_{train})$ guarantees that x_{train} becomes relatively more likely (compared to the rest).

Parameterizing probability distributions

Probability distributions $p(\mathbf{x})$ are a key building block in generative modeling. Properties:

1. non-negative: $p(\mathbf{x}) \geq 0$
2. sum-to-one: $\sum_{\mathbf{x}} p(\mathbf{x}) = 1$ (or $\int p(\mathbf{x})d\mathbf{x} = 1$ for continuous variables)

Coming up with a non-negative function $p_\theta(\mathbf{x})$ is not hard. For example:

- $g_\theta(\mathbf{x}) = f_\theta(\mathbf{x})^2$ where f_θ is any neural network
- $g_\theta(\mathbf{x}) = \exp(f_\theta(\mathbf{x}))$ where f_θ is any neural network
- $\cdots$

Problem: $g_\theta(\mathbf{x}) \geq 0$ is easy, but $g_\theta(\mathbf{x})$ might not sum-to-one. $\sum_{\mathbf{x}} g_\theta(\mathbf{x}) = Z(\theta) \neq 1$ in general, so $g_\theta(\mathbf{x})$ is not a valid probability mass function or density

Likelihood based learning

Problem: $g_\theta(\mathbf{x}) \geq 0$ is easy, but $g_\theta(\mathbf{x})$ might not be normalized
Solution:

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

Then by definition, $\int p_\theta(\mathbf{x}) d\mathbf{x} = 1$. Typically, choose $g_\theta(\mathbf{x})$ so that we know the volume *analytically* as a function of θ. For example,

1. $g_{(\mu,\sigma)}(x) = e^{-\frac{(x-\mu)^2}{2\sigma^2}}$. Volume is: $\int e^{-\frac{x-\mu}{2\sigma^2}} dx = \sqrt{2\pi\sigma^2} \rightarrow$ **Gaussian**
2. $g_\lambda(x) = e^{-\lambda x}$. Volume is: $\int_0^{+\infty} e^{-\lambda x} dx = \frac{1}{\lambda}$. $\rightarrow$ **Exponential**
3. Etc.

We can only choose functional forms $g_\theta(\mathbf{x})$ that we can integrate *analytically*. This is very restrictive, but as we have seen, they are very useful as building blocks for more complex models (e.g., conditionals in autoregressive models)

Likelihood based learning

Problem: $g_\theta(\mathbf{x}) \geq 0$ is easy, but $g_\theta(\mathbf{x})$ might not be normalized
Solution:

$$p_\theta(x) = \frac{1}{Volume(g_\theta)} g_\theta(x) = \frac{1}{\int g_\theta(x)dx} g_\theta(x)$$

Typically, choose $g_\theta(x)$ so that we know the volume *analytically*. More complex models can be obtained by combining these building blocks. Two main strategies:

1. **Autoregressive:** Products of normalized objects $p_\theta(x)p_{\theta'(x)}(y)$:
   $$\int_x \int_y p_\theta(x)p_{\theta'(x)}(y)dxdy = \int_x p_\theta(x) \underbrace{\int_y p_{\theta'(x)}(y)dy}_{=1} dx = \int_x p_\theta(x)dx = 1$$
2. **Latent variables:** Mixtures of normalized objects $\alpha p_\theta(x) + (1-\alpha)p_{\theta'}(x)$:
   $\int_x \alpha p_\theta(x) + (1-\alpha)p_{\theta'}(x)dx = \alpha + (1-\alpha) = 1$

How about using models where the "volume"/normalization constant is not easy to compute analytically?

Energy based model

$$p_\theta(x) = \frac{1}{\int \exp(f_\theta(\mathbf{x}))d\mathbf{x}} \exp(f_\theta(\mathbf{x})) = \frac{1}{Z(\theta)} \exp(f_\theta(\mathbf{x}))$$

The volume/normalization constant

$$Z(\theta) = \int \exp(f_\theta(\mathbf{x}))d\mathbf{x}$$

is also called the partition function. Why exponential (and not e.g. $f_\theta(\mathbf{x})^2$)?

1. Want to capture very large variations in probability. log-probability is the natural scale we want to work with. Otherwise need highly non-smooth f_θ.
2. Exponential families. Many common distributions can be written in this form.
3. These distributions arise under fairly general assumptions in statistical physics (maximum entropy, second law of thermodynamics). $-f_\theta(\mathbf{x})$ is called the **energy**, hence the name. Intuitively, configurations $\mathbf{x}$ with low energy (high $f_\theta(\mathbf{x})$) are more likely.

Energy based model

$$p_\theta(\mathbf{x}) = \frac{1}{\int \exp(f_\theta(\mathbf{x}))d\mathbf{x}} \exp(f_\theta(\mathbf{x})) = \frac{1}{Z(\theta)} \exp(f_\theta(\mathbf{x}))$$

Pros:

1. extreme flexibility: can use pretty much any function $f_\theta(\mathbf{x})$ you want

Cons (lots of them):

1. Sampling from $p_\theta(\mathbf{x})$ is hard
2. Evaluating and optimizing likelihood $p_\theta(\mathbf{x})$ is hard (learning is hard)
3. No feature learning (but can add latent variables)

Curse of dimensionality: The fundamental issue is that computing $Z(\theta)$ numerically (when no analytic solution is available) scales exponentially in the number of dimensions of $\mathbf{x}$. Nevertheless, some tasks do not require knowing $Z(\theta)$

Applications of Energy based models

$$p_\theta(\mathbf{x}) = \frac{1}{\int \exp(f_\theta(\mathbf{x}))d\mathbf{x}} \exp(f_\theta(\mathbf{x})) = \frac{1}{Z(\theta)} \exp(f_\theta(\mathbf{x}))$$

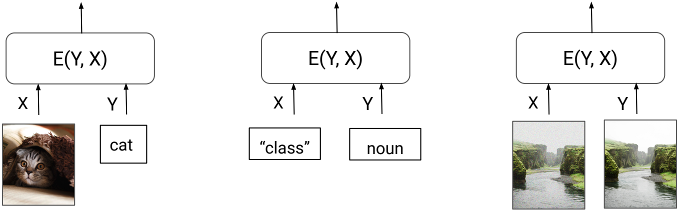
Given $\mathbf{x}$, $\mathbf{x}'$ evaluating $p_\theta(\mathbf{x})$ or $p_\theta(\mathbf{x}')$ requires $Z(\theta)$. However, their ratio

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

does not involve $Z(\theta)$. This means we can easily check which one is more likely. Applications:

1. anomaly detection
2. denoising

Applications of Energy based models

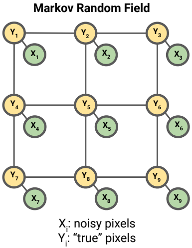


object recognition *sequence labeling* *image restoration*

Given a trained model, many applications require relative comparisons. Hence $Z(\theta)$ is not needed.

Example: Ising Model

- There is a true image $\mathbf{y} \in \{0,1\}^{3\times3}$, and a corrupted image $\mathbf{x} \in \{0,1\}^{3\times3}$. We know $\mathbf{x}$, and want to somehow recover $\mathbf{y}$.



- We model the joint probability distribution $p(\mathbf{y}, \mathbf{x})$ as

$$p(\mathbf{y}, \mathbf{x}) = \frac{1}{Z} \exp\left(\sum_i \psi_i(x_i, y_i) + \sum_{(i,j)\in E} \psi_{ij}(y_i, y_j) \right)$$

  - $\psi_i(x_i, y_i)$: the i-th corrupted pixel depends on the i-th original pixel
  - $\psi_{ij}(y_i, y_j)$: neighboring pixels tend to have the same value

- How did the original image $\mathbf{y}$ look like? Solution: maximize $p(\mathbf{y}|\mathbf{x})$

Example: Product of Experts

- Suppose you have trained several models $q_{\theta_1}(\mathbf{x})$, $r_{\theta_2}(\mathbf{x})$, $t_{\theta_3}(\mathbf{x})$. They can be different models (PixelCNN, Flow, etc.)
- Each one is like an *expert* that can be used to score how likely an input $\mathbf{x}$ is.
- Assuming the experts make their judgments indpendently, it is tempting to ensemble them as

$$p_{\theta_1}(\mathbf{x})q_{\theta_2}(\mathbf{x})r_{\theta_3}(\mathbf{x})$$

- To get a valid probability distribution, we need to normalize

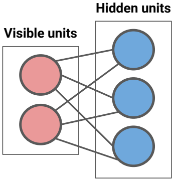
$$p_{\theta_1,\theta_2,\theta_3}(\mathbf{x}) = \frac{1}{Z(\theta_1, \theta_2, \theta_3)} q_{\theta_1}(\mathbf{x})r_{\theta_2}(\mathbf{x})t_{\theta_3}(\mathbf{x})$$

- Note: similar to an AND operation (e.g., probability is zero as long as one model gives zero probability), unlike mixture models which behave more like OR

Example: Restricted Boltzmann machine (RBM)

- RBM: energy-based model with latent variables
- Two types of variables:
  1. $\mathbf{x} \in \{0,1\}^n$ are visible variables (e.g., pixel values)
  2. $\mathbf{z} \in \{0,1\}^m$ are latent ones
- The joint distribution is

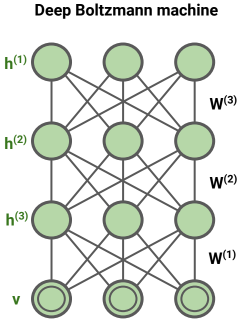
$$p_{W,b,c}(\mathbf{x},\mathbf{z}) = \frac{1}{Z}\exp\left(\mathbf{x}^T W\mathbf{z} + b\mathbf{x} + c\mathbf{z}\right) = \frac{1}{Z}\exp\left(\sum_{i=1}^{n}\sum_{j=1}^{m} x_i z_j w_{ij} + b\mathbf{x} + c\mathbf{z}\right)$$



- Restricted because there are no visible-visible and hidden-hidden connections, i.e., $x_i x_j$ or $z_i z_j$ terms in the objective

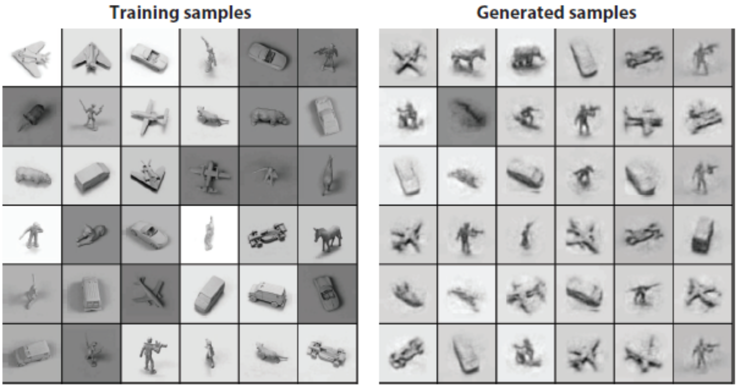
Deep Boltzmann Machines

Stacked RBMs are one of the first deep generative models:



Bottom layer variables **v** are pixel values. Layers above (**h**) represent "higher-level" features (corners, edges, etc). Early deep neural networks for *supervised learning* had to be pre-trained like this to make them work.

Boltzmann Machines: samples



Energy based models: learning and inference

$$p_\theta(\mathbf{x}) = \frac{1}{\int \exp(f_\theta(\mathbf{x}))} \exp(f_\theta(\mathbf{x})) = \frac{1}{Z(\theta)} \exp(f_\theta(\mathbf{x}))$$

Pros:

1. can plug in pretty much any function $f_\theta(\mathbf{x})$ you want

Cons (lots of them):

1. Sampling is hard
2. Evaluating likelihood (learning) is hard
3. No feature learning

Curse of dimensionality: The fundamental issue is that computing $Z(\theta)$ numerically (when no analytic solution is available) scales exponentially in the number of dimensions of $\mathbf{x}$.

Computing the normalization constant is hard

- As an example, the RBM joint distribution is

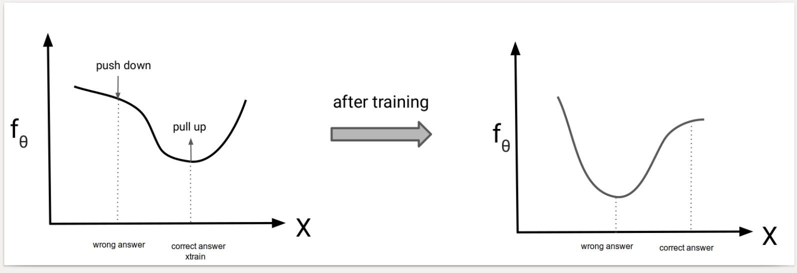
$$p_{W,b,c}(\mathbf{x}, \mathbf{z}) = \frac{1}{Z} \exp\left(\mathbf{x}^T W \mathbf{z} + b\mathbf{x} + c\mathbf{z}\right)$$

  where
  1. $\mathbf{x} \in \{0, 1\}^n$ are visible variables (e.g., pixel values)
  2. $\mathbf{z} \in \{0, 1\}^m$ are latent ones
- The normalization constant (the "volume") is

$$Z(W, b, c) = \sum_{\mathbf{x}\in\{0,1\}^n} \sum_{\mathbf{z}\in\{0,1\}^m} \exp\left(\mathbf{x}^T W \mathbf{z} + b\mathbf{x} + c\mathbf{z}\right)$$

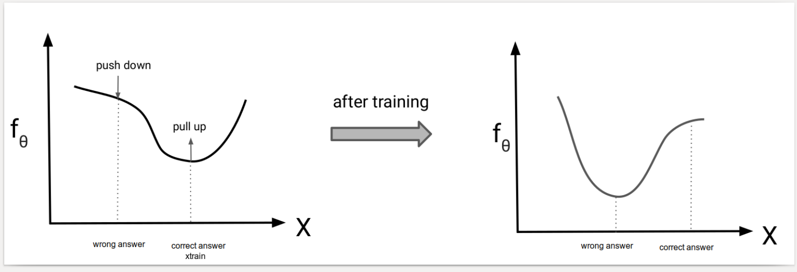
- Note: it is a well defined function of the parameters W, b, c, but no simple closed-form. Takes time exponential in n, m to compute. This means that *evaluating* the objective function $p_{W,b,c}(\mathbf{x}, \mathbf{z})$ for likelihood based learning is hard.
- Optimizing the un-normalized probability $\exp\left(\mathbf{x}^T W \mathbf{z} + b\mathbf{x} + c\mathbf{z}\right)$ is easy (w.r.t. trainable parameters W, b, c), but *optimizing* the likelihood $p_{W,b,c}(\mathbf{x}, \mathbf{z})$ is also difficult..

Training intuition



- Goal: maximize $\frac{f_\theta(x_{train})}{Z(\theta)}$. Increase numerator, decrease denominator.
- **Intuition**: because the model is not normalized, increasing the un-normalized probability $f_\theta(x_{train})$ by changing θ does **not** guarantees that x_{train} becomes relatively more likely (compared to the rest).
- We also need to take into account the effect on other "wrong points" and try to "push them down" to *also* make $Z(\theta)$ small.

Contrastive Divergence



- Goal: maximize $\frac{f_\theta(x_{train})}{Z(\theta)}$
- **Idea**: Instead of evaluating $Z(\theta)$ exactly, use a Monte Carlo estimate.
- **Contrastive divergence algorithm**: sample $x_{sample} \sim p_\theta$, take step on $\nabla_\theta \left(f_\theta(x_{train}) - f_\theta(x_{sample})\right)$. Make training data more likely than typical sample from the model. Recall comparisons are easy in energy based models!
- Looks simple, but wow to sample? Unfortunately, sampling is hard

Sampling from Energy based models

$$p_\theta(\mathbf{x}) = \frac{1}{\int \exp(f_\theta(\mathbf{x}))} \exp(f_\theta(\mathbf{x})) = \frac{1}{Z(\theta)} \exp(f_\theta(\mathbf{x}))$$

- No direct way to sample like in autoregressive or flow models. Main issue: cannot easily compute how likely each possible sample is
- However, we can easily compare two samples $\mathbf{x}, \mathbf{x}'$.
- Use an iterative approach called Markov Chain Monte Carlo:
  1. Initialize x^0 randomly, $t = 0$
  2. Let $x' = x^t + \text{noise}$
     1. If $f_\theta(x') > f_\theta(x^t)$, let $x^{t+1} = x'$
     2. Else let $x^{t+1} = x'$ with probability $\exp(f_\theta(x') - f_\theta(x^t))$
  3. Go to step 2
- Works in theory, but can take a very long time to converge

Conclusion

- Energy-based models are another useful tool for modeling high-dimensional probability distributions.
- Very flexible class of models. Currently less popular because of computational issues.
- Energy based GANs: energy is represented by a discriminator. Contrastive samples (like in contrastive divergence) from a GAN-styke generator.
- Reference: LeCun et. al, *A Tutorial on Energy-Based Learning* [Link]