Introduction to Deep Generative Modeling

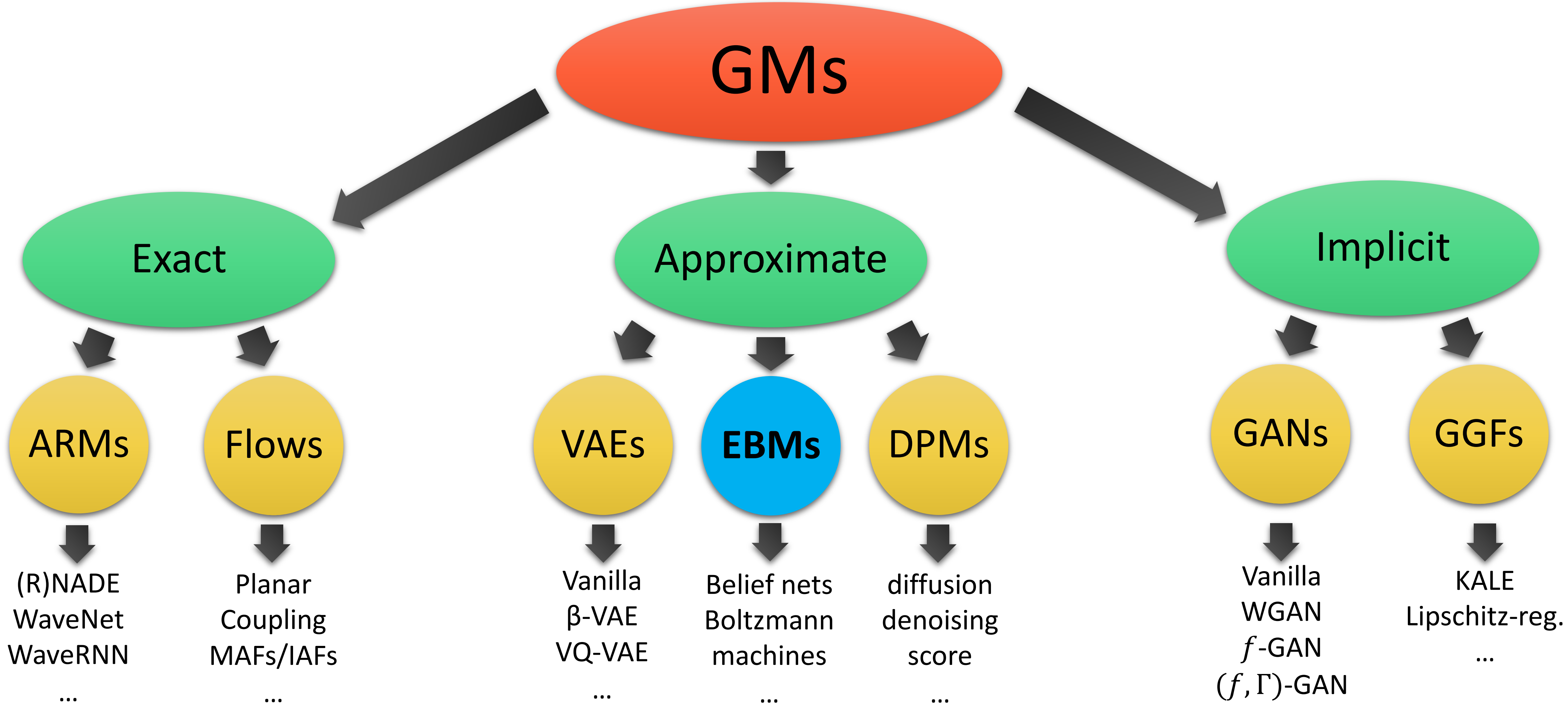
Lecture #11

HY-673 – Computer Science Dep., University of Crete

Professors: Yannis Pantazis, Yannis Stylianou

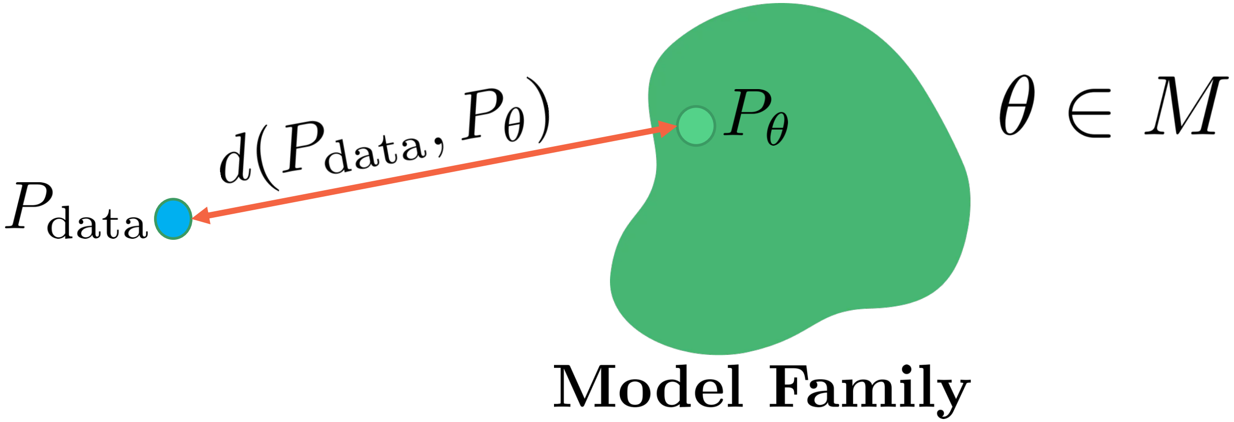
Teaching Assistant: Michail Raptakis

Taxonomy of GMs



Recap

$$x_i \sim P_{\text{data}}$$
$$i = 1, 2, \ldots, n$$

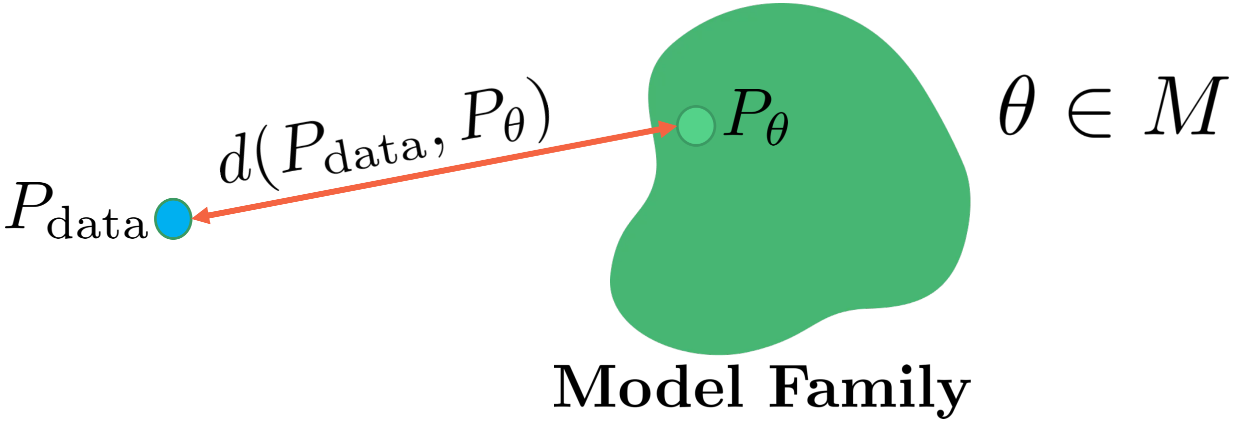


- Autoregressive models: $p_\theta(x_1, x_2, \ldots, x_n) = \prod_{i=1}^{n} p_\theta(x_i | x_{<i})$.
- Normalizing flow models: $p_\theta(\mathbf{x}) = p(\mathbf{z})|\det J_{f_\theta}|$, where $\mathbf{z} = f_\theta(\mathbf{x})$.
- Variational autoencoders: $p_\theta(\mathbf{x}) = \int p(\mathbf{z}) p_\theta(\mathbf{x}|\mathbf{z}) d\mathbf{z}$.

Cons: Model architectures are restricted.

Today's Lecture

$$x_i \sim P_{\text{data}}$$
$$i = 1, 2, \ldots, n$$



- Energy-Based Models (EBMs):
  - Very flexible model architectures.
  - Stable training.
  - Relatively high sample quality.
  - Flexible composition.

Parametrizing Probability Distributions

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

1. non-negative: $p(x) \geq 0$.
2. sum-to-one: $\sum_x p(x) = 1$, or $\int_{\mathcal{X}} p(x)dx = 1$ for continuous variables.

Coming up with a non-negative function $p_\theta(\mathbf{x})$ is not hard. Given any funtion $f_\theta(\mathbf{x})$, we can choose:

- $g_\theta(\mathbf{x}) = f_\theta(\mathbf{x})^2$
- $g_\theta(\mathbf{x}) = |f_\theta(\mathbf{x})|$
- $g_\theta(\mathbf{x}) = \exp\left(f_\theta(\mathbf{x})\right)$
- $g_\theta(\mathbf{x}) = \log((1 + \exp\left(f_\theta(\mathbf{x})\right)))$

Parametrizing Probability Distributions

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

1. non-negative: $p(x) \geq 0$.
2. sum-to-one: $\sum_x p(x) = 1$, or $\int_{\mathcal{X}} p(x)dx = 1$ for continuous variables.

- *Sum-to-one is key:* Total "volume" is fixed: Increasing $p(x_i)$ guarantees that x_i becomes relatively more likely compared to the rest.

Problem:

- $g_\theta(x)$ is easy, but $g_\theta(x)$ might not sum to one.
- $Z_\theta := \sum_x g_\theta(x) \neq 1$ in general, so $g_\theta(x)$ is not a valid PMF or PDF.

Parametrizing Probability Distributions

Problem: $g_\theta(x) \geq 0$ is easy, but $g_\theta(x)$ might not be normalized.

Solution: $p_\theta(x) = \frac{1}{\text{Volume}(g_\theta)} g_\theta(x) = \frac{1}{Z_\theta} g_\theta(x), \quad \int p_\theta(x)dx = 1.$

$\implies$ by definition: $\int p_\theta(x)dx = 1.$

Example: Choose $g_\theta(\mathbf{x})$ so that we know the volume *analytically* as a function of θ:

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

Parametrizing Probability Distributions

Problem: $g_\theta(x) \geq 0$ is easy, but $g_\theta(x)$ might not be normalized.

Solution: $p_\theta(x) = \frac{1}{\text{Volume}(g_\theta)} g_\theta(x) = \frac{1}{Z_\theta} g_\theta(x), \quad \int p_\theta(x)dx = 1.$

3. $g_\theta(x) = \exp\left(\theta^T t(x)\right) h(x)$, **volume:** $\exp\left(A(\theta)\right)$, where $A(\theta) := \log \int \exp\left(\theta^T t(x)\right) h(x)dx \rightarrow$ **Exponential family of distributions**.

   - Normal, Poisson, exponential
   - Bernoulli, Beta, Gamma, Dirichlet, Wishart, etc.

Function $g_\theta(\mathbf{x})$ needs to allow *analytical* integration.
Despite being restrictive, they are useful as <u>*building blocks*</u> for more complex distributions.

Parametrizing Probability Distributions

Problem: $g_\theta(x) \geq 0$ is easy, but $g_\theta(x)$ might not be normalized.

Solution: $p_\theta(x) = \frac{1}{\text{Volume}(g_\theta)} g_\theta(x) = \frac{1}{Z_\theta} g_\theta(x), \quad \int p_\theta(x)dx = 1.$

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

1. **Autoregressive:** Products of normalized objects $p_\theta(x)p_{\theta'}(y)$:

$$\int_x \int_y p_\theta(x)p_{\theta'}(y)dxdy = \int_x p_\theta(x) \underbrace{\int_y p_{\theta'}(y)dydx}_{=1} = \int_x p_\theta(x)dx = 1.$$

Parametrizing Probability Distributions

Problem: $g_\theta(x) \geq 0$ is easy, but $g_\theta(x)$ might not be normalized.

Solution: $p_\theta(x) = \frac{1}{\text{Volume}(g_\theta)} g_\theta(x) = \frac{1}{Z_\theta} g_\theta(x), \quad \int p_\theta(x)dx = 1.$

2. **Latent Variables:** Mixtures of normalized objects $\alpha p_\theta(\mathbf{x}) + (1-\alpha)p_{\theta'}(\mathbf{x})$:

$$\int_x \alpha p_\theta(x) + (1-\alpha)p_{\theta'} dx = \alpha + (1-\alpha) = 1.$$

How about using models where the "volume"/normalization constant of $g_\theta(\mathbf{x})$ is not easy to compute analytically?

Energy-Based Model

Definition: $$p_\theta(x) = \frac{1}{\int \exp\left(f_\theta(x)\right) dx} \exp\left(f_\theta(x)\right) = \frac{1}{Z_\theta} \exp\left(f_\theta(x)\right).$$

- The volume/normalization constant $Z_\theta = \int \exp((f_\theta(x))\, dx$, 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. Many common distributions can be written in the exponential family 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

Definition: $$p_\theta(x) = \frac{1}{\int \exp\left(f_\theta(x)\right) dx} \exp\left(f_\theta(x)\right) = \frac{1}{Z_\theta} \exp\left(f_\theta(x)\right).$$

Pros:

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

Cons:

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

Curse of Dimensionality: The fundamental issue is that numerically computing Z_θ (when no analytic solution is available) scales *exponentially* in the number of dimensions of x.

Nevertheless, **some tasks do not require knowing** Z_θ.

Applications of EBMs

Definition: $$p_\theta(x) = \frac{1}{\int \exp\left(f_\theta(x)\right) dx} \exp\left(f_\theta(x)\right) = \frac{1}{Z_\theta} \exp\left(f_\theta(x)\right).$$

- Given $x,\ x'$, evaluating $p_\theta(x)$ or $p_\theta(x')$ requires Z_θ.
- However, their **ratio** does not depend on Z_θ:

$$\frac{p_\theta(x)}{p_\theta(x')} = \exp\left(f_\theta(x) - f_\theta(x')\right).$$

- This means we can easily check which one is more likely. Applications include:
    1. Anomaly Detection.
    2. Denoising.

Applications of EBMs

$f_\theta(x, y)$

x ↑ 

y ↑ cat

Object Recognition

$f_\theta(x, y)$

x ↑ "class"

y ↑ noun

Sequence Labeling

$f_\theta(x, y)$

x ↑ 

y ↑ 

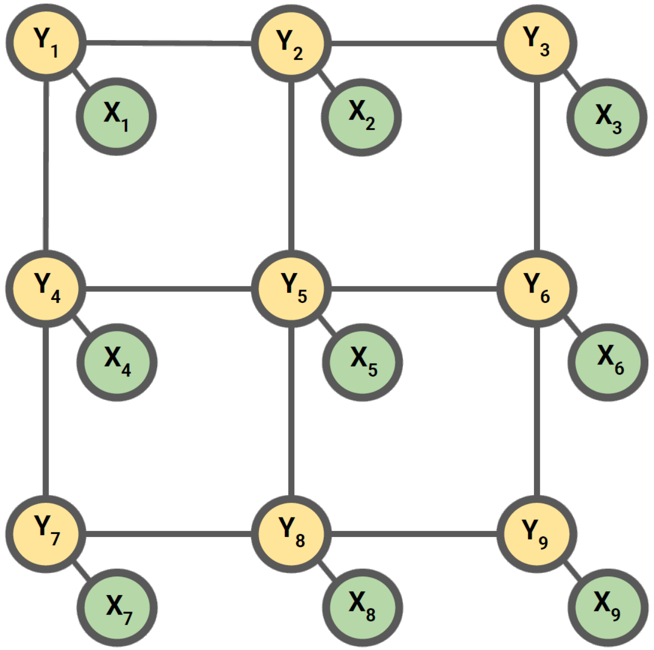
Image Restoration

Given a trained model, many applications require relative comparisons. Hence, Z_θ is not needed.

Example: Ising Model

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

Markov Random Field (MRF)



x_i: noisy pixels

y_i: "true" pixels

Example: Ising Model

- We model the joint distribution $p(x, y)$ as:

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

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

- How did the original image y look like?
  Answer: Maximize $p(y|x)$, or, equivalently, maximize $p(x, y)$.

Example: Product of Experts

- Suppose you have trained several models $q_{\theta_1}(x), r_{\theta_2}(x), t_{\theta_3}(x)$. They can be different models (e.g., PixelCNN, Flow, etc.)
- Each one is like an <u>*expert*</u> that can be used to score how likely an input x is.
- Assuming the experts make their judgement independently, it is tempting to ensemble them as:
$$q_{\theta_1}(x) r_{\theta_2}(x) t_{\theta_3}(x).$$
- To get a valid probability distribution, we need to normalize:
$$p_{\theta_1,\theta_2,\theta_3}(x) = \frac{1}{Z_{\theta_1,\theta_2,\theta_3}} q_{\theta_1}(x) r_{\theta_2}(x) t_{\theta_3}(x).$$
- Node: 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: Product of Experts

Image source:
Du et al., 2020.

Young (EBM)

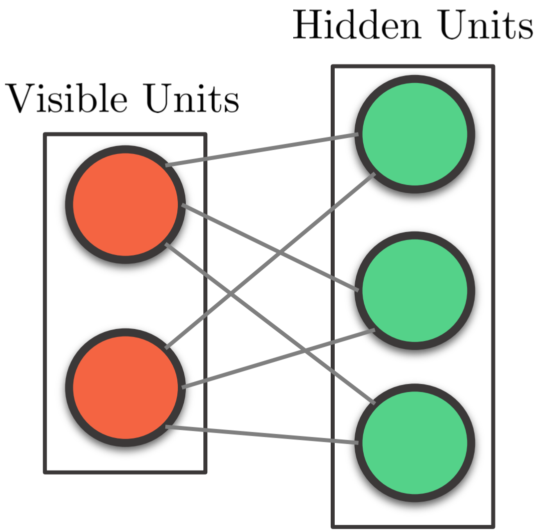
Young AND Female (EBM)

Young AND Female
AND Smiling (EBM)

Young AND Female
AND Smiling
AND Wavy Hair (EBM)

Example: Deep Boltzmann Machines (DBMs)

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



- The joint distribution is:

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

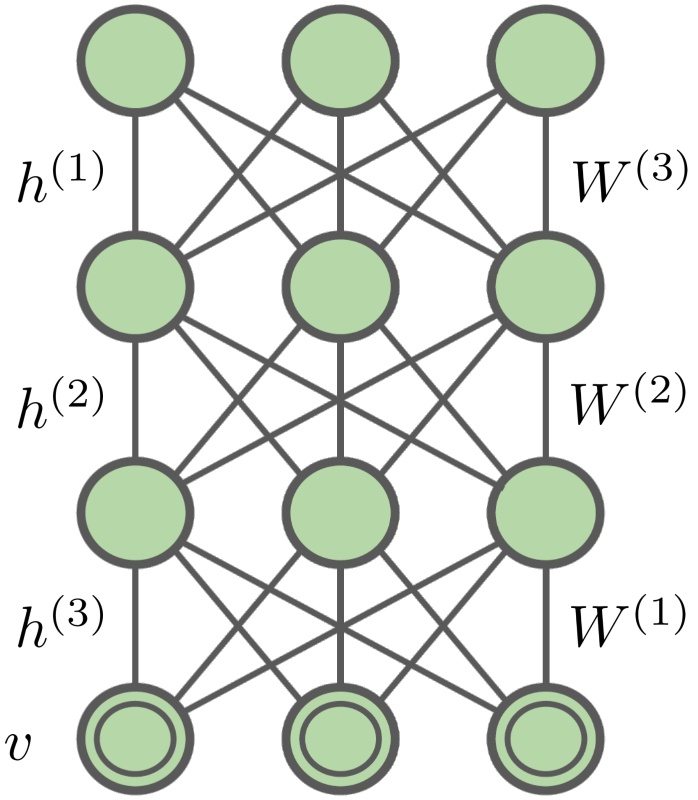
- Restricted because there are no visible-visible and hidden-hidden connections, i.e., $x_i x_j$ or $z_i z_j$ terms in the objective.

Example: Deep Boltzmann Machines (DBMs)

Stacked RBMs are one of the first deep generative models:

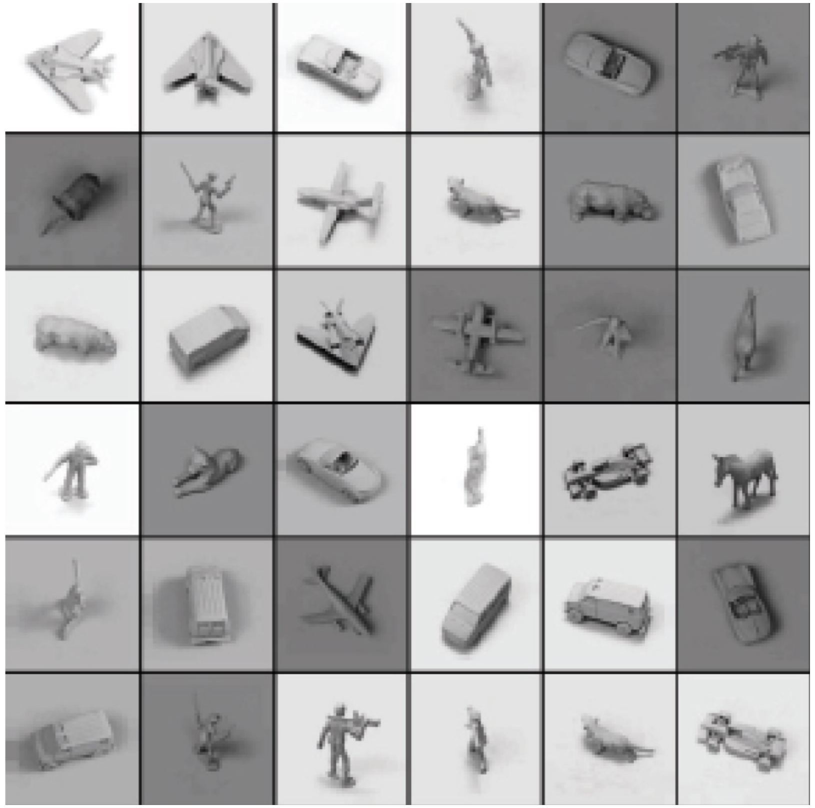
- Bottom layer variables v are pixel values. Layers above (h) represent "higher level" features (e.g., corners, edges, etc.)
- Early deep neural networks for *supervised learning* had to be pre-trained like this to make them work.
- Very similar to *deep belief networks* (one of the first deep learning models with an effective training algorithm).

Deep Boltzmann Machine



Deep Boltzmann Machines: Samples

Training Samples

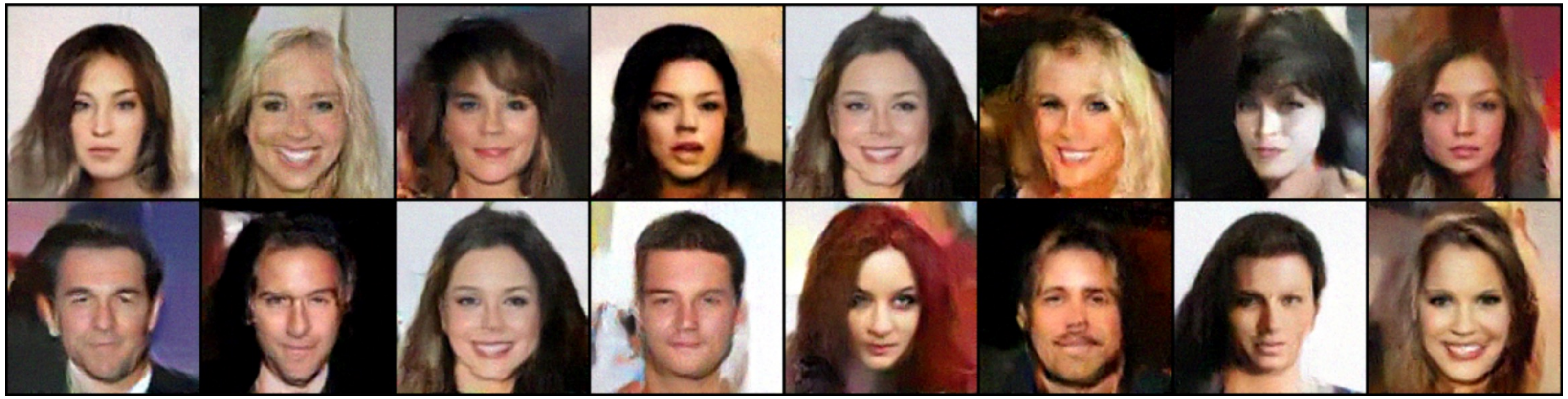


Generated Samples

Image source: Salakhutdinov and Hinton, 2009.

Modern EBMs

Langevin sampling

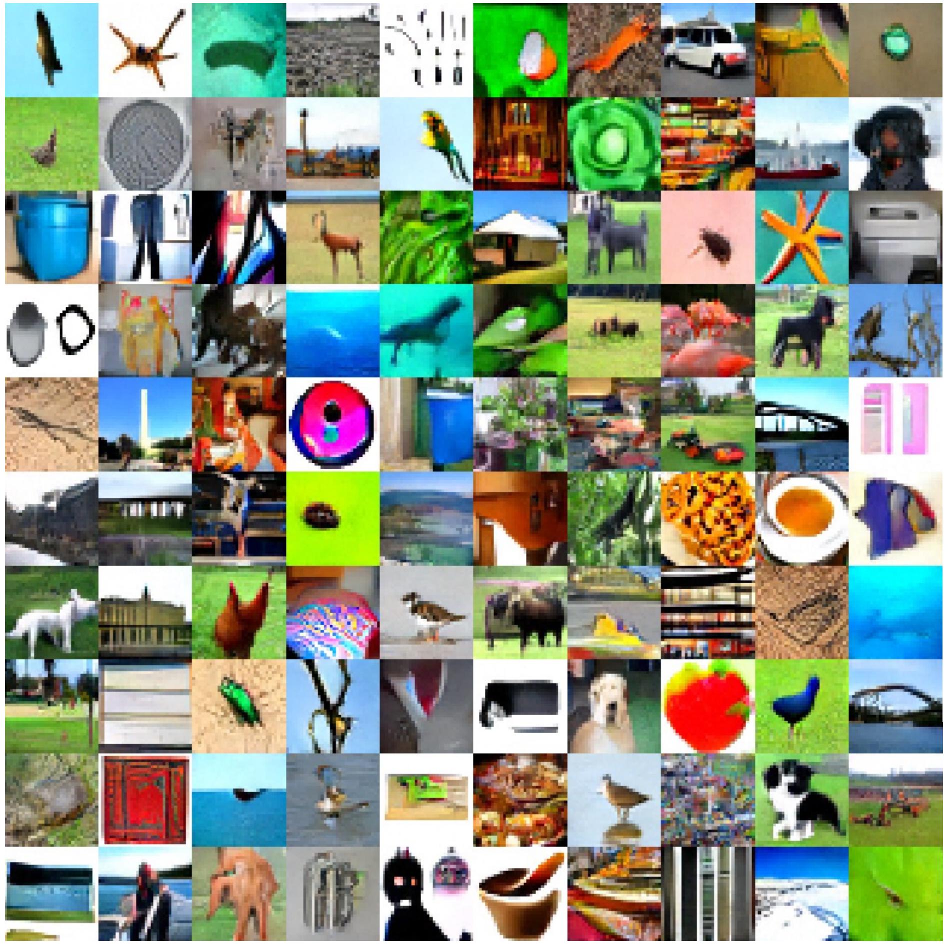


Face samples

Images source: "Learning Non-Convergent Non-Persistent Short-Run MCMC Toward Energy-Based Model", Nijkamp et al. 2019.

Modern EBMs

ImageNet sample generation



Images source: "Implicit Generation and Modeling with Energy-Based Models" Du et al., 2019.

EBMs: Learning and Inference

$$p_\theta(x) = \frac{1}{\int \exp\left(f_\theta(x)\right) dx} \exp\left(f_\theta(x)\right) = \frac{1}{Z_\theta} \exp\left(f_\theta(x)\right).$$

<u>Pros:</u>

1. Can plug in pretty much any function $f_\theta(x)$ you want.

<u>Cons</u> (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 numerically computing Z_θ (when no analytic solution is available) scales <u>*exponentially*</u> in the number of dimensions of x.

Computing the Normalization Constant is Hard

- As an example, the RBM joint distribution is:

$$p_{W,b,c}(x,z) = \frac{1}{Z}\exp\left(x^T W z + bx + cz\right), \text{ where :}$$

1. $x \in \{0,1\}^n$ are visible variables (e.g., binary pixel values).
2. $z \in \{0,1\}^m$ are latent ones.

- The normalization constant (the "volume") is:

$$Z_{W,b,c} := \sum_{x\in\{0,1\}^n} \sum_{z\in\{0,1\}^m} \exp\left(x^T W z + bx + cz\right).$$

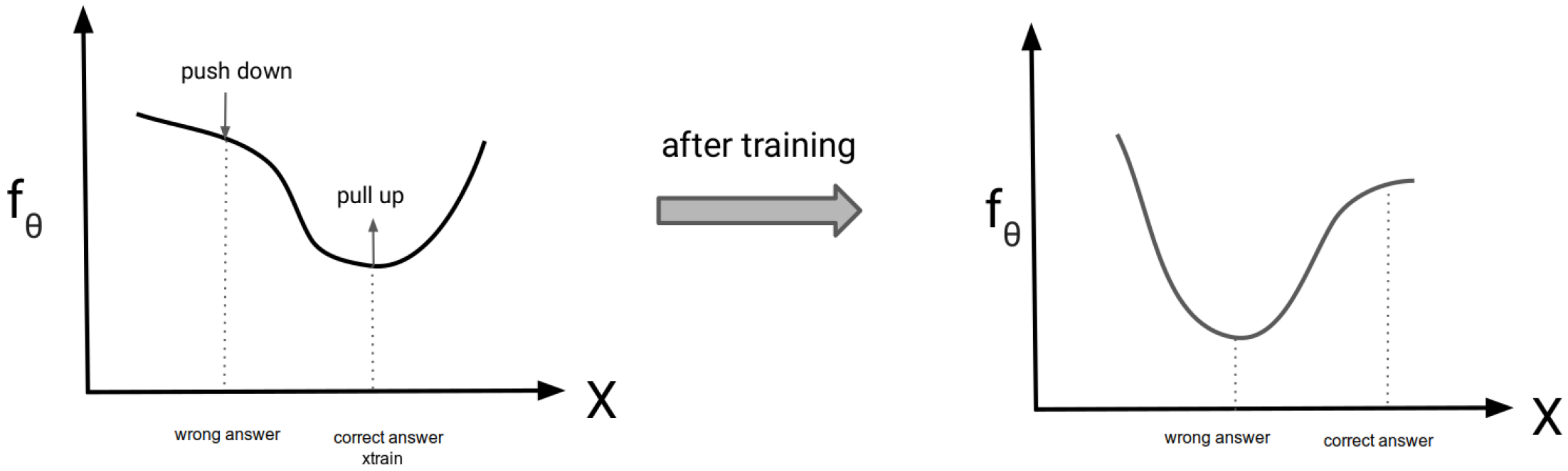
Computing the Normalization Constant is Hard

$$\text{Joint distribution : } \; p_{W,b,c}(x,z) = \frac{1}{Z}\exp\left(x^T W z + b^T x + c^T z\right).$$

$$\text{Volume : } \; Z_{W,b,c} := \sum_{x\in\{0,1\}^n}\sum_{z\in\{0,1\}^m}\exp\left(x^T W z + b^T x + c^T z\right).$$

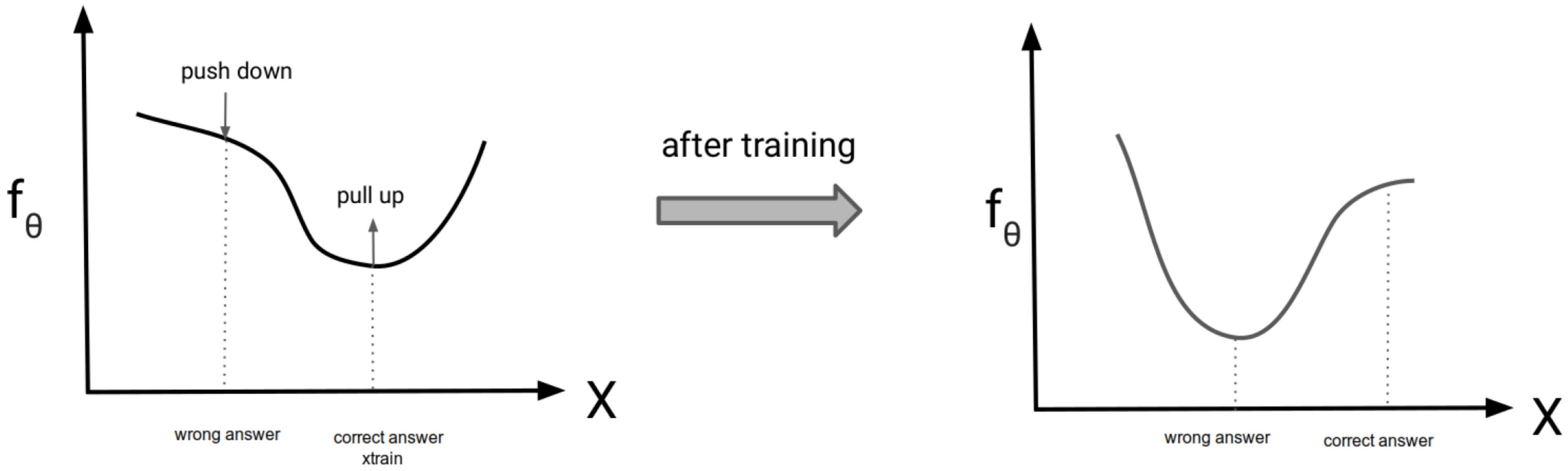
- **Note:** It is a well-defined function of the parameters W, b, c, but no simple closed form. Takes time, exponential in n, m to compute. This means that *evaluating* the objective function $p_{W,b,c,}(x,z)$ for likelihood-based learning is hard.
- **Observation:** Optimizing the likelihood $p_{W,b,c}(x,z)$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

Training Intuition



- **Goal:** Maximize $\frac{1}{Z_\theta}\exp\left(f_\theta(x_{\text{train}})\right)$. Increase numerator, decrease denominator.
- **Intuition:** Because the model is not normalized, increasing the un-normalized log-probability $f_\theta(x_{\text{train}})$ by changin θ does **not** guarantee that x_{train} becomes relatively more likely (compared to the rest).
- We also need to take into account the effect on the other "wrong points" and try to "push them down" to *also* make Z_θ small.

Contrastive Divergence



- **Goal:** Maximize $\frac{1}{Z_\theta} \exp\left(f_\theta(x_{\text{train}})\right)$.
- **Idea:** Instead of evaluating Z_θ exactly, use a Monte Carlo estimate.
- **Contrastive Divergence Algorithm:** Sample $x_{\text{sample}} \sim p_\theta(x)$, take step on $\nabla_\theta \left(f_\theta(x_{\text{train}}) - f_\theta(x_{\text{sample}})\right)$. Make training data more likely than typical sample from the model.

Contrastive Divergence

- Maximize log-likelihood: $\max_\theta f_\theta(x_{\text{train}}) - \log Z_\theta$ with the log-likelihood gradient being:

$$\begin{aligned}
\nabla_\theta f_\theta(x_{\text{train}}) - \nabla_\theta \log Z_\theta &= \nabla_\theta f_\theta(x_{\text{train}}) - \frac{1}{Z_\theta}\nabla_\theta Z_\theta \\
&= \nabla_\theta f_\theta(x_{\text{train}}) - \frac{1}{Z_\theta}\int \nabla_\theta \exp\left(f_\theta(x)\right) dx \\
&= \nabla_\theta f_\theta(x_{\text{train}}) - \int \frac{1}{Z_\theta}\exp\left(f_\theta(x)\right) \nabla_\theta f_\theta(x) dx \\
&= \nabla_\theta f_\theta(x_{\text{train}}) - \mathbb{E}_{p_\theta}\left[\nabla_\theta f_\theta(x)\right] \\
&\approx \nabla_\theta f_\theta(x_{\text{train}}) - \nabla_\theta f_\theta(x_{\text{sample}}),
\end{aligned}$$

- How to sample?

where $x_{\text{sample}} \sim p_\theta(x) := \dfrac{1}{Z_\theta}\exp\left(f_\theta(x)\right).$

Sampling from an EBM

- No direct way to sample like in autoregressive or flow models. Main issue: Cannot easily compute how likely each possible sample it.

$$p_\theta(x) = \frac{1}{Z_\theta} \exp\left(f_\theta(x)\right).$$

- However, we can easily compare two samples x, x'.

- Use an iterative approach called Markov Chain Monte Carlo (MCMC):

1. Initialize $x^{(0)}$ randomly, $t = 0$.
2. Let $x' = x^{(t)} + \text{noise}$.
   - 2.1. If $f_\theta(x') > f_\theta\left(x^{(t)}\right)$, let $x^{(t+1)} = x'$.
   - 2.2. Else, let $x^{(t+1)} = x'$ with probability $\exp\left(f_\theta(x') - f_\theta\left(x^{(t)}\right)\right)$.
3. Go to step 2.

$\rightarrow$ Works in theory, but can take a very long time to converge.

Sampling from an EBM

- For any continuous distribution $p_\theta(x)$, suppose we can compute its gradient, i.e., the **score function**, $\nabla_x \log p_\theta(x)$.

- Let $\pi(x)$ be a prior distribution that is easy to sample from.

- Langevin MCMC:

  1. $x^{(0)} \sim \pi(x)$.

  2. Repeat $x^{(t+1)} \sim x^{(t)} + \epsilon \nabla_x \log p_\theta(x^{(t)}) + \sqrt{2\epsilon} z^{(t)}$, for $t = 0, 1, \ldots, T-1$, where $z^{(t)} \sim \mathcal{N}(0, I)$.

  3. If $\epsilon \to 0$ and $T \to \infty$, we have $x_T \sim p_\theta(x)$.

- Note that for energy-based models: $\nabla_x \log p_\theta(x) = \nabla_x f_\theta(x) - \underbrace{\nabla_x \log Z_\theta}_{=0} = \nabla_x f_\theta(x)$.

References

1. Probabilistic Machine Learning: Advanced Topics (*Chapter 23*)
   Kevin P Murphy, The MIT Press (2023)
2. How to Train Your Energy-Based Models
   https://arxiv.org/pdf/2101.03288.pdf
3. https://github.com/yataobian/awesome-ebm
4. Statistical exponential families: A digest with flash cards, Nielsen & Garcia, https://arxiv.org/pdf/0911.4863.pdf

Introduction to Deep Generative Modeling

Lecture #11

HY-673 – Computer Science Dep., University of Crete

Professors: Yannis Pantazis, Yannis Stylianou

Teaching Assistant: Michail Raptakis