How to Train your Energy-Based Model

Yang Song and Diederik Kingma, 2021

(Chapter 24 in Probabilistic Machine Learning: Advanced Topics)

Energy-Based Models (High-Level)

- Non-normalized probabilistic models that specify a probability density or mass function up to an unknown normalization constant
- No restriction on the tractability of normalizing constant
  - Allows for more flexibility and ability to model a broader family of probability distributions
- Unknown normalization constant makes training difficult
- How do we train such models?
  - Three major ways
    1. Maximum Likelihood Training with MCMC sampling
    2. Score Matching
    3. Noise Contrastive Estimation

Energy-Based Models (EBMs)

Assume unconditional EBMs over a single dependent variable $\boldsymbol{x}$. The density of an EBM is given by

$$p_{\boldsymbol{\theta}}(\mathbf{x}) = \frac{\exp(-E_{\boldsymbol{\theta}}(\mathbf{x}))}{Z_{\boldsymbol{\theta}}}$$

where $E_\theta(\boldsymbol{x})$ (the energy) is a nonlinear regression function with parameters θ and Z_θ denotes the normalizing constant (partition function)

$$Z_{\boldsymbol{\theta}} = \int \exp(-E_{\boldsymbol{\theta}}(\mathbf{x}))\ \mathrm{d}\mathbf{x}$$

which is constant w.r.t. $\boldsymbol{x}$ but is a function of θ which results in intractability for evaluation and differentiation of $\log p_\theta(\boldsymbol{x})$ w.r.t. its parameters.

Maximum Likelihood Training with MCMC

- Defacto standard for learning probabilistic models from i.i.d. data is MLE so we start here.
- Let $p_\theta(\boldsymbol{x})$ be a probabilistic model parameterized by θ and $p_{\text{data}}(\boldsymbol{x})$ be the underlying data distribution of a dataset.
- We fit $p_\theta(\boldsymbol{x})$ to $p_{\text{data}}(\boldsymbol{x})$ by maximizing the expected log-likelihood over the data distribution

$$\mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})}[\log p_{\boldsymbol{\theta}}(\mathbf{x})]$$

- Maximizing the likelihood is equivalent to minimizing the KL divergence between $p_{\text{data}}(\boldsymbol{x})$ and $p_\theta(\boldsymbol{x})$

$$\begin{aligned}
-\mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})}[\log p_{\boldsymbol{\theta}}(\mathbf{x})] &= D_{KL}(p_{\text{data}}(\mathbf{x}) \parallel p_{\boldsymbol{\theta}}(\mathbf{x})) - \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})}[\log p_{\text{data}}(\mathbf{x})] \\
&= D_{KL}(p_{\text{data}}(\mathbf{x}) \parallel p_{\boldsymbol{\theta}}(\mathbf{x})) - \text{constant},
\end{aligned}$$

Maximum Likelihood Training with MCMC

- We cannot compute the likelihood of an EBM due to the intractability in the normalizing constant Z_θ.
- We can estimate the gradient of the log-likelihood with MCMC allowing for likelihood maximization with gradient ascent.
- The gradient of the log-probability of an EBM can be decomposed as two sums

$$\nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\mathbf{x}) = -\nabla_{\boldsymbol{\theta}} E_{\boldsymbol{\theta}}(\mathbf{x}) - \nabla_{\boldsymbol{\theta}} \log Z_{\boldsymbol{\theta}}$$

- The first term is straight forward with modern auto-differentiation. We must figure out how to approximate the second term which is intractable.
- We can rewrite this gradient as an expectation

Maximum Likelihood Training with MCMC

$$
\begin{aligned}
\nabla_{\boldsymbol{\theta}} \log Z_{\boldsymbol{\theta}} &= \nabla_{\boldsymbol{\theta}} \log \int \exp(-E_{\boldsymbol{\theta}}(\mathbf{x})) d\mathbf{x} \\
&\overset{(i)}{=} \left(\int \exp(-E_{\boldsymbol{\theta}}(\mathbf{x})) d\mathbf{x} \right)^{-1} \nabla_{\boldsymbol{\theta}} \int \exp(-E_{\boldsymbol{\theta}}(\mathbf{x})) d\mathbf{x} \\
&= \left(\int \exp(-E_{\boldsymbol{\theta}}(\mathbf{x})) d\mathbf{x} \right)^{-1} \int \nabla_{\boldsymbol{\theta}} \exp(-E_{\boldsymbol{\theta}}(\mathbf{x})) d\mathbf{x} \\
&\overset{(ii)}{=} \left(\int \exp(-E_{\boldsymbol{\theta}}(\mathbf{x})) d\mathbf{x} \right)^{-1} \int \exp(-E_{\boldsymbol{\theta}}(\mathbf{x}))(-\nabla_{\boldsymbol{\theta}} E_{\boldsymbol{\theta}}(\mathbf{x})) d\mathbf{x} \\
&= \int \left(\int \exp(-E_{\boldsymbol{\theta}}(\mathbf{x})) d\mathbf{x} \right)^{-1} \exp(-E_{\boldsymbol{\theta}}(\mathbf{x}))(-\nabla_{\boldsymbol{\theta}} E_{\boldsymbol{\theta}}(\mathbf{x})) d\mathbf{x} \\
&\overset{(iii)}{=} \int \frac{\exp(-E_{\boldsymbol{\theta}}(\mathbf{x}))}{Z_{\boldsymbol{\theta}}} (-\nabla_{\boldsymbol{\theta}} E_{\boldsymbol{\theta}}(\mathbf{x})) d\mathbf{x} \\
&\overset{(iv)}{=} \int p_{\boldsymbol{\theta}}(\mathbf{x})(-\nabla_{\boldsymbol{\theta}} E_{\boldsymbol{\theta}}(\mathbf{x})) d\mathbf{x} \\
&= \mathbb{E}_{\mathbf{x} \sim p_{\boldsymbol{\theta}}(\mathbf{x})} \left[-\nabla_{\boldsymbol{\theta}} E_{\boldsymbol{\theta}}(\mathbf{x}) \right],
\end{aligned}
$$

(i) Gradient Chain Rule

(ii) Gradient Chain Rule

(iii) EBM Definition

(iv) EBM Definition

Maximum Likelihood Training with MCMC

- Thus, we can obtain an unbiased one-sample Monte Carlo estimate of the log-likelihood gradient by

$$\nabla_{\boldsymbol{\theta}} \log Z_{\boldsymbol{\theta}} \simeq -\nabla_{\boldsymbol{\theta}} E_{\boldsymbol{\theta}}(\tilde{\mathbf{x}}),$$

where $\tilde{\boldsymbol{x}} \sim p_\theta(\boldsymbol{x})$ is a random sample from the distribution over $\boldsymbol{x}$ given by the EBM.

- As long as we can sample the model, we can estimate the log-likelihood gradient allowing for easy optimization

Maximum Likelihood Training with MCMC

- Drawing samples is not trivial, so we focus on efficient MCMC sampling of EBMs
  - Langevin MCMC and Hamiltonian Monte Carlo both use the fact that the gradient of the log-probability w.r.t. $\boldsymbol{x}$ (the score) is equal to the negative gradient of the energy

$$\nabla_{\mathbf{x}} \log p_{\boldsymbol{\theta}}(\mathbf{x}) = -\nabla_{\mathbf{x}} E_{\boldsymbol{\theta}}(\mathbf{x}) - \underbrace{\nabla_{\mathbf{x}} \log Z_{\boldsymbol{\theta}}}_{=0} = -\nabla_{\mathbf{x}} E_{\boldsymbol{\theta}}(\mathbf{x}).$$

- When using Langevin MCMC, to sample from $p_\theta(\boldsymbol{x})$, we first draw initial sample $\boldsymbol{x}^0$ from some simple prior and simulate an (overdamped) Langevin diffusion process for K steps with step size $\epsilon > 0$

$$\mathbf{x}^{k+1} \leftarrow \mathbf{x}^k + \frac{\epsilon^2}{2} \underbrace{\nabla_{\mathbf{x}} \log p_{\boldsymbol{\theta}}(\mathbf{x}^k)}_{=-\nabla_{\mathbf{x}} E_{\boldsymbol{\theta}}(\mathbf{x})} + \epsilon \mathbf{z}^k, \quad k = 0, 1, \cdots, K-1.$$

- When $\epsilon \longrightarrow 0$ and $K \longrightarrow \infty$, $\boldsymbol{x}^K$ is guaranteed to distribute as $p_\theta(\boldsymbol{x})$

Score Matching

- We can additionally learn an EBM by approximately matching the first derivatives of its log-PDF to the first derivatives to the log-PDF of the data distribution.
- If the derivatives match, then the EBM captures the data distribution exactly.
- We call the first order gradient of a log-PDF the *score* of that distribution.

$$\underbrace{\nabla_{\mathbf{x}} \log p_{\boldsymbol{\theta}}(\mathbf{x})}_{\text{Score}} = -\nabla_{\mathbf{x}} E_{\boldsymbol{\theta}}(\mathbf{x})$$

- It is useful to use equivalence of scores because the score of an EBM does not involve the typically intractable normalizing constant

Score Matching

- Let $p_{\text{data}}(\boldsymbol{x})$ be the underlying data distribution, but we do not know its PDF.
- The score matching objective minimizes the discrepancy between two distribution called the Fisher divergence

$$D_F(p_{\text{data}}(\mathbf{x}) \parallel p_{\boldsymbol{\theta}}(\mathbf{x})) = \mathbb{E}_{p_{\text{data}}(\mathbf{x})}\left[\frac{1}{2}\|\nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x}) - \nabla_{\mathbf{x}} \log p_{\boldsymbol{\theta}}(\mathbf{x})\|^2\right]$$

- The expectation in this objective allows for unbiased Monte Carlo estimation using the empirical mean of samples $\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x})$.
- The second term is generally intractable since it requires knowing the true gradient of the log-data distribution.
- Rewrite the Fisher divergence using integration by parts

$$D_F(p_{\text{data}}(\mathbf{x}) \parallel p_{\boldsymbol{\theta}}(\mathbf{x})) = \mathbb{E}_{p_{\text{data}}(\mathbf{x})}\left[\frac{1}{2}\sum_{i=1}^{d}\left(\frac{\partial E_{\boldsymbol{\theta}}(\mathbf{x})}{\partial x_i}\right)^2 + \frac{\partial^2 E_{\boldsymbol{\theta}}(\mathbf{x})}{(\partial x_i)^2}\right] + \text{constant}$$

where d is dimensionality of $\boldsymbol{x}$

- In general, computation of second derivatives is quadratic with d, thus it does not scale well with high-dimensional data. Thus, this can only be applied to relatively simple energy function.

Denoising Score Matching (DSM)

- Previous score matching objective requires several regularity conditions (continuously differentiable, finite everywhere), but these may not hold in practice (e.g., images).
- We can alleviate this issue by adding noise to each datapoint: $\tilde{\boldsymbol{x}} = \boldsymbol{x} + \boldsymbol{\epsilon}$
  - As long as $p(\boldsymbol{\epsilon})$ is smooth, the resulting noise data distribution $q(\tilde{\boldsymbol{x}}) = \int q(\tilde{\boldsymbol{x}}|\boldsymbol{x})\, p_{\text{data}}(\boldsymbol{x}) d\boldsymbol{x}$ is also smooth and thus $D_F(q(\tilde{\boldsymbol{x}})||p_\theta(\tilde{\boldsymbol{x}}))$ is a proper objective
- We still need second order derivatives if using the Fisher divergence, but we can circumvent this by showing

$$
\begin{aligned}
D_F(q(\tilde{\mathbf{x}}) \parallel p_{\boldsymbol{\theta}}(\tilde{\mathbf{x}})) &= \mathbb{E}_{q(\tilde{\mathbf{x}})}\left[\frac{1}{2}\|\nabla_{\mathbf{x}} \log q(\tilde{\mathbf{x}}) - \nabla_{\mathbf{x}} \log p_{\boldsymbol{\theta}}(\tilde{\mathbf{x}})\|_2^2\right] \\
&= \mathbb{E}_{q(\mathbf{x},\tilde{\mathbf{x}})}\left[\frac{1}{2}\|\nabla_{\mathbf{x}} \log q(\tilde{\mathbf{x}}|\mathbf{x}) - \nabla_{\mathbf{x}} \log p_{\boldsymbol{\theta}}(\tilde{\mathbf{x}})\|_2^2\right] + \text{constant},
\end{aligned}
$$

- Here we have avoided the unknown $p_{\text{data}}(\boldsymbol{x})$ and expensive second order derivatives.

Denoising Score Matching (DSM)

- If $p_{\text{data}}(\boldsymbol{x})$ is already well-behaved (i.e., satisfies regularity constraints), then $D_F(q(\tilde{\boldsymbol{x}})||p_\theta(\tilde{\boldsymbol{x}})) \neq D_F(p_{\text{data}}(\boldsymbol{x})||p_\theta(\boldsymbol{x}))$ and DSM is not a consistent objective.
  - This inconsistency is non-negligible when $q(\tilde{\boldsymbol{x}})$ significantly differs from $p_{\text{data}}(\boldsymbol{x})$
- We can attenuate this inconsistency if we choose $q \approx p_{\text{data}}(\boldsymbol{x})$ (i.e., use a small noise perturbation)
  - This comes at the cost of significantly increasing the variance of the objective values

Denoising Score Matching (DSM): Example

- Suppose $q(\tilde{\boldsymbol{x}}|\boldsymbol{x}) = \mathcal{N}(\tilde{\boldsymbol{x}}; \boldsymbol{x}, \sigma^2 I)$ and $\sigma \approx 0$. The corresponding DSM objective is

$$D_F(q(\tilde{\mathbf{x}}) \parallel p_{\boldsymbol{\theta}}(\tilde{\mathbf{x}})) = \mathbb{E}_{p_{\text{data}}(\mathbf{x})}\mathbb{E}_{\mathbf{z}\sim\mathcal{N}(0,I)}\left[\frac{1}{2}\left\|\frac{\mathbf{z}}{\sigma} + \nabla_{\mathbf{x}}\log p_{\boldsymbol{\theta}}(\mathbf{x}+\sigma\mathbf{z})\right\|_2^2\right]$$
$$\simeq \frac{1}{2N}\sum_{i=1}^{N}\left\|\frac{\mathbf{z}^{(i)}}{\sigma} + \nabla_{\mathbf{x}}\log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}+\sigma\mathbf{z}^{(i)})\right\|_2^2,$$

- When $\sigma \longrightarrow 0$, we can leverage Taylor series expansion to rewrite the Monte Carlo estimator as

$$\frac{1}{2N}\sum_{i=1}^{N}\left[\frac{2}{\sigma}(\mathbf{z}^{(i)})^{\mathsf{T}}\nabla_{\mathbf{x}}\log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) + \frac{\|\mathbf{z}^{(i)}\|_2^2}{\sigma^2}\right] + \text{constant}.$$

- When estimating with samples, the variance of summation terms will grow unbounded as $\sigma \longrightarrow 0$
- We construct a variable that is, for small σ, positively correlated with the DSM objective

$$c_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z}) = \frac{2}{\sigma}\mathbf{z}^{\mathsf{T}}\nabla_{\mathbf{x}}\log p_{\boldsymbol{\theta}}(\mathbf{x}) + \frac{\|\mathbf{z}\|_2^2}{\sigma^2} - \frac{d}{\sigma^2}.$$

- If we subtract this from the DSM objective, we obtain an estimator with reduced variance for DSM training

$$\frac{1}{2N}\sum_{i=1}^{N}\left\|\frac{\mathbf{z}^{(i)}}{\sigma} + \nabla_{\mathbf{x}}\log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}+\sigma\mathbf{z}^{(i)})\right\|_2^2 - c_{\boldsymbol{\theta}}(\mathbf{x}^{(i)},\mathbf{z}^{(i)}).$$

Sliced Score Matching (SSM)

- Recall that DSM does not give a consistent estimator of the data distribution
  - One cannot directly obtain an EBM that exactly matches the data distribution even with unlimited data
- Instead of minimizing the Fisher divergence between two vector-valued scores, randomly sample a projection vector $\boldsymbol{v}$, take the inner product between $\boldsymbol{v}$ and the two scores, and then compare the resulting two scalars
  - Sliced Score Matching (SSM) minimizes the sliced Fisher divergence

$$D_{SF}(p_{\text{data}}(\mathbf{x}) || p_{\boldsymbol{\theta}}(\mathbf{x})) = \mathbb{E}_{p_{\text{data}}(\mathbf{x})} \mathbb{E}_{p(\mathbf{v})} \left[\frac{1}{2} (\mathbf{v}^\mathsf{T} \nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x}) - \mathbf{v}^\mathsf{T} \nabla_{\mathbf{x}} \log p_{\boldsymbol{\theta}}(\mathbf{x}))^2 \right]$$

where $p(\boldsymbol{v})$ denotes a projection distribution such that $\mathbb{E}_{p(\boldsymbol{v})}[\boldsymbol{v}\boldsymbol{v}^T]$ is positive definite.

- Sliced Fisher divergence has an implicit form that does not involve the true log-likelihood given by

$$\begin{aligned} &D_{SF}(p_{\text{data}}(\mathbf{x}) \| p_{\boldsymbol{\theta}}(\mathbf{x})) \\ &= \mathbb{E}_{p_{\text{data}}(\mathbf{x})} \mathbb{E}_{p(\mathbf{v})} \left[\frac{1}{2} \sum_{i=1}^{d} \left(\frac{\partial E_{\boldsymbol{\theta}}(\mathbf{x})}{\partial x_i} v_i \right)^2 + \sum_{i=1}^{d} \sum_{j=1}^{d} \frac{\partial^2 E_{\boldsymbol{\theta}}(\mathbf{x})}{\partial x_i \partial x_j} v_i v_j \right] + \text{constant.} \end{aligned}$$

Sliced Score Matching (SSM)

- We still have second order derivative terms, but this can be computed efficiently with linear cost in dimensionality d because

$$\sum_{i=1}^{d}\sum_{j=1}^{d}\frac{\partial^2 E_{\boldsymbol{\theta}}(\mathbf{x})}{\partial x_i \partial x_j}v_i v_j = \sum_{i=1}^{d}\frac{\partial}{\partial x_i}\underbrace{\left(\sum_{j=1}^{d}\frac{\partial E_{\boldsymbol{\theta}}(\mathbf{x})}{\partial x_j}v_j\right)}_{:=f(\mathbf{x})} v_i,$$

- Many choices of $p(\boldsymbol{v})$ yield a partly closed form solution to the SSM objective leading to lower variance. For example, when $p(\boldsymbol{v})$ is a standard normal

$$\mathbb{E}_{p_{\text{data}}(\mathbf{x})}\mathbb{E}_{p(\mathbf{v})}\left[\frac{1}{2}\sum_{i=1}^{d}\left(\frac{\partial E_{\boldsymbol{\theta}}(\mathbf{x})}{\partial x_i}v_i\right)^2\right] = \mathbb{E}_{p_{\text{data}}(\mathbf{x})}\left[\frac{1}{2}\sum_{i=1}^{d}\left(\frac{\partial E_{\boldsymbol{\theta}}(\mathbf{x})}{\partial x_i}\right)^2\right]$$

- Thus, we have

$$\begin{aligned} &D_{SF}(p_{\text{data}}(\mathbf{x})\|p_{\boldsymbol{\theta}}(\mathbf{x})) \\ &= \mathbb{E}_{p_{\text{data}}(\mathbf{x})}\mathbb{E}_{\mathbf{v}\sim\mathcal{N}(\mathbf{0},\boldsymbol{I})}\left[\frac{1}{2}\sum_{i=1}^{d}\left(\frac{\partial E_{\boldsymbol{\theta}}(\mathbf{x})}{\partial x_i}\right)^2 + \sum_{i=1}^{d}\sum_{j=1}^{d}\frac{\partial^2 E_{\boldsymbol{\theta}}(\mathbf{x})}{\partial x_i \partial x_j}v_i v_j\right] + \text{constant}. \end{aligned}$$

Score-Based Generative Models

- Goal: Use an EBM to create new samples that are similar to training data.
- Solution: Train an EBM with Score Matching, and then sample from it with MCMC approaches
  - We only need a model for score when training Score Matching and sampling with score-based MCMC and do not have to model the energy explicitly.
  - Score models share weights and are implemented with a single neural network conditioned on noise scale (Noise-Conditional Score Network)

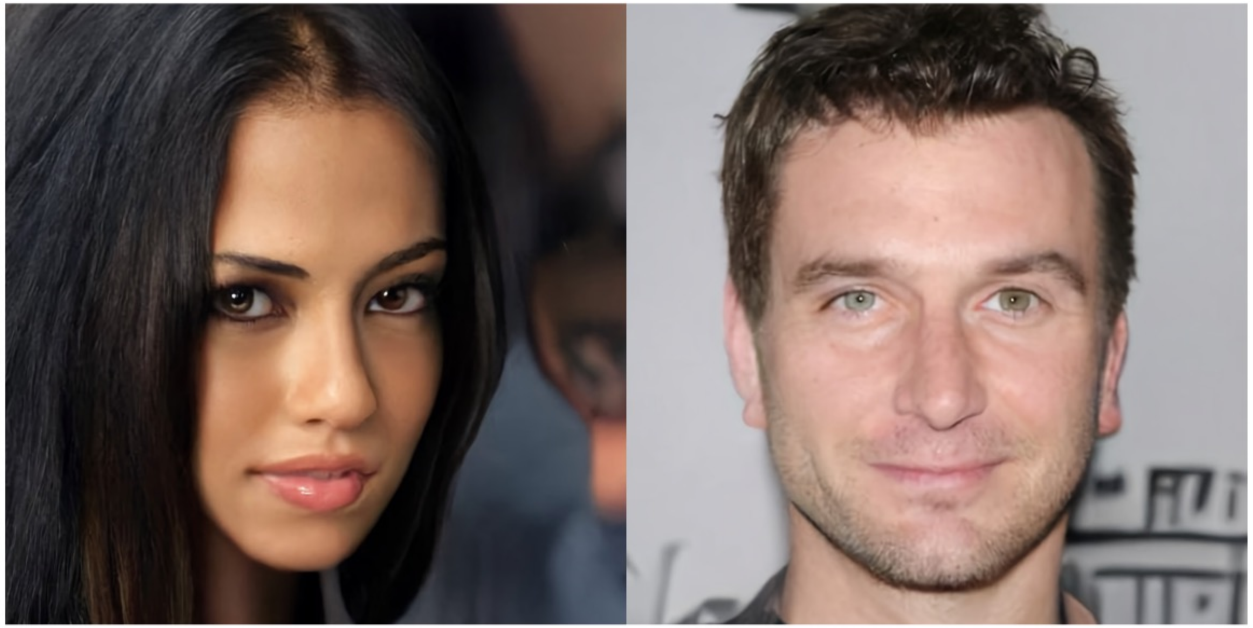

Figure 1: Samples from a score-based generative model trained with multiple scales of noise perturbations (resolution 1024×1024). Image credit to Song et al. (2021).

Noise Contrastive Estimation (NCE)

- Learn an EBM by contrasting it with another distribution with known density
- Let $p_{\mathrm{data}}(\boldsymbol{x})$ be the data distribution and $p_{\mathrm{n}}(\boldsymbol{x})$ be a chosen distribution with know density, called the noise distribution.
  - Usually pick $p_{\mathrm{n}}(\boldsymbol{x})$ to be simple with known PDF, such as standard normal
- Let y be a binary variable with Bernoulli Distribution used to define a mixture distribution of noise and data:
  - $p_{\mathrm{n,data}}(\boldsymbol{x}) = p(y = 0)\, p_{\mathrm{n}}(\boldsymbol{x}) + p(y = 1)\, p_{\mathrm{data}}(\boldsymbol{x})$
- Given a sample $\boldsymbol{x}$ from this mixture, the posterior probability of $y = 0$ is

$$p_{\mathrm{n,data}}(y = 0 \mid \mathbf{x}) = \frac{p_{\mathrm{n,data}}(\mathbf{x} \mid y = 0)p(y = 0)}{p_{\mathrm{n,data}}(\mathbf{x})} = \frac{p_{\mathrm{n}}(\mathbf{x})}{p_{\mathrm{n}}(\mathbf{x}) + \nu p_{\mathrm{data}}(\mathbf{x})}$$

where $v = p(y = 1)/p(y = 0)$

Noise Contrastive Estimation (NCE)

- Suppose we define our EBM as previously.
- We will now treat Z_θ as a learnable (scalar) parameter
- Given this EBM, we define a mixture of noise and the model distribution
  - $p_{n,\theta}(\boldsymbol{x}) = p(y = 0)p_n(\boldsymbol{x}) + p(y = 1)p_\theta(\boldsymbol{x})$
- Similarly, the posterior of $y = 0$ from this mixture model is

$$p_{n,\boldsymbol{\theta}}(y = 0 \mid \mathbf{x}) = \frac{p_n(\mathbf{x})}{p_n(\mathbf{x}) + \nu p_{\boldsymbol{\theta}}(\mathbf{x})}$$

- We indirectly fit $p_\theta(\boldsymbol{x})$ to $p_{\text{data}}(\boldsymbol{x})$ by fitting $p_{n,\theta}(y|\boldsymbol{x})$ to $p_{n,\text{data}}(y|\boldsymbol{x})$ through conditional maximum likelihood objective via SGD

$$\begin{aligned}\boldsymbol{\theta}^* &= \arg\min_{\boldsymbol{\theta}} \mathbb{E}_{p_{n,\text{data}}(\mathbf{x})}[D_{KL}(p_{n,\text{data}}(y \mid \mathbf{x}) \parallel p_{n,\boldsymbol{\theta}}(y \mid \mathbf{x}))] \\ &= \arg\max_{\boldsymbol{\theta}} \mathbb{E}_{p_{n,\text{data}}(\mathbf{x},y)}[\log p_{n,\boldsymbol{\theta}}(y \mid \mathbf{x})],\end{aligned}$$

- When the model is sufficiently powerful, $p_{n,\theta^*}(y|\boldsymbol{x})$ will match $p_{n,\text{data}}(y|\boldsymbol{x})$ at the optimum

$$\begin{aligned} & p_{n,\boldsymbol{\theta}^*}(y = 0 \mid \mathbf{x}) \equiv p_{n,\text{data}}(y = 0 \mid \mathbf{x}) \\ \iff & \frac{p_n(\mathbf{x})}{p_n(\mathbf{x}) + \nu p_{\boldsymbol{\theta}^*}(\mathbf{x})} \equiv \frac{p_n(\mathbf{x})}{p_n(\mathbf{x}) + \nu p_{\text{data}}(\mathbf{x})} \\ \iff & p_{\boldsymbol{\theta}^*}(\mathbf{x}) \equiv p_{\text{data}}(\mathbf{x}) \end{aligned}$$

Noise Contrastive Estimation (NCE)

- NCE provides the normalizing constant as a by-product of its training procedure
- When the EBM is expressive (e.g., DNN) we can assume it is able to approximate a normalized probability distribution and absorb Z_θ into the parameters of $E_\theta(\boldsymbol{x})$
- The resulting EBM trained via NCE will be self-normalized (normalizing constant is close to 1)
- We must choose $p_{\mathrm{n}}(\boldsymbol{x})$ correctly for success
- Works best when $p_{\mathrm{n}}(\boldsymbol{x})$ is close to data distribution

Adversarial Training

- We can additionally sidestep expensive MCMC sampling by learning an auxiliary model through adversarial training to allow for fast sampling
- We can rewrite the maximum likelihood objective by introducing a variational distribution $q_\phi(\boldsymbol{x})$

$$
\begin{aligned}
\mathbb{E}_{p_{\text{data}}(\mathbf{x})}[\log p_{\boldsymbol{\theta}}(\mathbf{x})] &= \mathbb{E}_{p_{\text{data}}(\mathbf{x})}[-E_{\boldsymbol{\theta}}(\mathbf{x})] - \log Z_{\boldsymbol{\theta}} \\
&= \mathbb{E}_{p_{\text{data}}(\mathbf{x})}[-E_{\boldsymbol{\theta}}(\mathbf{x})] - \log \int e^{-E_{\boldsymbol{\theta}}(\mathbf{x})} \mathrm{d}\mathbf{x} \\
&= \mathbb{E}_{p_{\text{data}}(\mathbf{x})}[-E_{\boldsymbol{\theta}}(\mathbf{x})] - \log \int q_{\boldsymbol{\phi}}(\mathbf{x}) \frac{e^{-E_{\boldsymbol{\theta}}(\mathbf{x})}}{q_{\boldsymbol{\phi}}(\mathbf{x})} \mathrm{d}\mathbf{x} \\
&\overset{(i)}{\leq} \mathbb{E}_{p_{\text{data}}(\mathbf{x})}[-E_{\boldsymbol{\theta}}(\mathbf{x})] - \int q_{\boldsymbol{\phi}}(\mathbf{x}) \log \frac{e^{-E_{\boldsymbol{\theta}}(\mathbf{x})}}{q_{\boldsymbol{\phi}}(\mathbf{x})} \mathrm{d}\mathbf{x} \qquad \text{Jensen's Inequality} \\
&= \mathbb{E}_{p_{\text{data}}(\mathbf{x})}[-E_{\boldsymbol{\theta}}(\mathbf{x})] - \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{x})}[-E_{\boldsymbol{\theta}}(\mathbf{x})] - H(q_{\boldsymbol{\phi}}(\mathbf{x})),
\end{aligned}
$$

- For training, we can first minimize this upper bound w.r.t. $q_\phi(\boldsymbol{x})$ so that it is closer to the likelihood objective, and then maximize w.r.t. $E_\theta(\boldsymbol{x})$ as a surrogate for maximizing likelihood

$$
\max_{\boldsymbol{\theta}} \min_{\boldsymbol{\phi}} \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{x})}[E_{\boldsymbol{\theta}}(\mathbf{x})] - \mathbb{E}_{p_{\text{data}}(\mathbf{x})}[E_{\boldsymbol{\theta}}(\mathbf{x})] - H(q_{\boldsymbol{\phi}}(\mathbf{x})).
$$

- This optimization is similar to GANs and can be achieved by adversarial training