Generative models

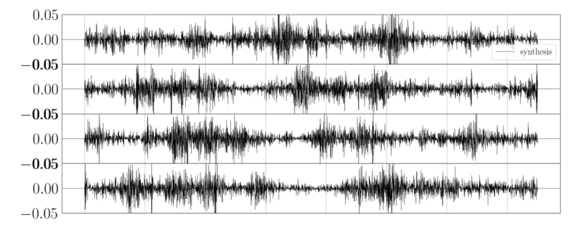
how they work and how to train them

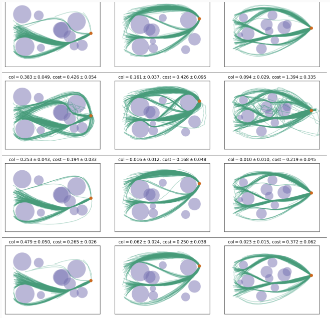
Simon Coste

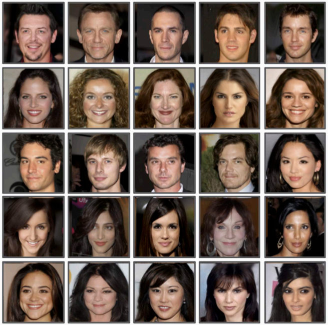
June 25, 2024

Intro: Generative Modelling

$x_*^1, \ldots, x_*^n$: dataset drawn from an unknown distribution ρ_* ("target")







The two goals of generative modelling:

1. Generate ‘new’ samples from ρ_* (direct problem)
2. Find a ‘good’ estimator $\hat{\rho}_*$ for ρ_* (inverse problem)

Examples of generative models: Energy-Based Models, Generative Adversarial Networks, Variational Auto-Encoders, Normalizing Flows, Neural ODEs, Probabilistic PCA, Gaussian Mixtures, Diffusions-based models, Flow matching, Consistency models...

The two goals of generative modelling:

1. Generate 'new' samples from ρ_* (direct problem)
2. Find a 'good' estimator $\hat{\rho}_*$ for ρ_* (inverse problem)

Examples of generative models: Energy-Based Models, Generative Adversarial Networks, Variational Auto-Encoders, Normalizing Flows, Neural ODEs, Probabilistic PCA, Gaussian Mixtures, Diffusions-based models, Flow matching, Consistency models...

"variational inference"

Energy-Based Models

Defining EBMs

$U_\theta : \mathbb{R}^d \to \mathbb{R}_+$ = parametrized family of functions ("model energies")

Definition of the model densities:

$$\rho_\theta(x) = \frac{e^{-U_\theta(x)}}{Z_\theta} \qquad Z_\theta = \int e^{-U_\theta(x)} dx.$$

Defining EBMs

$U_\theta : \mathbb{R}^d \to \mathbb{R}_+$ = parametrized family of functions ("model energies")

Definition of the model densities:

$$\rho_\theta(x) = \frac{e^{-U_\theta(x)}}{Z_\theta} \qquad\qquad Z_\theta = \int e^{-U_\theta(x)} dx.$$

Examples:

- $U_\theta(x) = \langle x, \theta x \rangle$ with θ a square matrix: centered Gaussian distributions
- $U_\theta(x) = |x - \theta|$: family of Laplace distributions
- $U_\theta(x) =$ a complicated neural network with parameters θ: deep EBMs

Training an EBM

The goal is to find the optimal θ_* achieving the best 'fit' between the model ρ_θ and the true unknown density ρ_*.

$$\theta_* \in \operatorname{arg\,min} \operatorname{dist}(\rho_*, \rho_\theta)$$

Q: how do we choose the distance?

Using an EBM

Once U_{θ_*} has been trained, new synthetic samples are obtained by sampling from the distribution

$$\hat{\rho}_* = \rho_{\theta_*} = \frac{e^{-U_{\theta_*}}}{Z_{\theta_*}}.$$

This step typically needs MCMC methods such as Langevin:

$$X_{\tau+1} = X_\tau - \eta \nabla_x U_{\theta_*}(X_\tau) + \sqrt{\eta}\xi_\tau \qquad \xi_\tau \sim \mathcal{N}(0, I)$$

This is called “implicit generation” [Du and Mordatch 19].

Advantages of EBMs

- **Simplicity.** Only one neural network U_θ
  $\rightarrow$ *VAEs and GANs require at least two!*
- **Flexibility.** We can exploit the tradeoff between quality and cost
  $\rightarrow$ *impossible with feed-forward generators such as GANs or NFs*
- **Compositionality.** Combining different EBMs is as simple
  $\rightarrow$ *just add the energies*
- **Reusability.** Can be used to help various other tasks
  $\rightarrow$ *inpainting, importance sampling, OOD detection...*

Choosing the right loss for EBM learning

Kullback-Leibler divergence $\leftrightarrow$ max-likelihood

$$\begin{aligned}\mathrm{dist}(\rho_\theta, \rho_*) &= \mathbb{E}_{X\sim\rho_*} \log \rho_*(X)) - \log \rho_\theta(X) \\ &\approx \mathrm{cst} - \frac{1}{n}\sum \log \rho_\theta(x_i^*)\end{aligned}$$

Choosing the right loss for EBM learning

Kullback-Leibler divergence $\leftrightarrow$ max-likelihood

$$\begin{aligned}\mathrm{dist}(\rho_\theta, \rho_*) &= \mathbb{E}_{X\sim\rho_*} \log \rho_*(X)) - \log \rho_\theta(X) \\ &\approx \mathrm{cst} - \frac{1}{n}\sum \log \rho_\theta(x_i^*)\end{aligned}$$

Fisher divergence

$$\begin{aligned}\mathrm{dist}(\rho_\theta, \rho_*) &= \mathbb{E}_{X\sim\rho_*} \left|\nabla \log \rho_*(X) - \nabla \log \rho_\theta(X)\right|^2 \\ &\approx \frac{1}{n}\sum \left|\nabla \log \rho_*(x_i^*) - \nabla \log \rho_\theta(x_i^*)\right|^2\end{aligned}$$

Choosing the right loss for EBM learning

Kullback-Leibler divergence $\leftrightarrow$ max-likelihood

$$\mathrm{dist}(\rho_\theta, \rho_*) = \mathbb{E}_{X\sim\rho_*} \log \rho_*(X)) - \log \rho_\theta(X)$$
$$\approx \mathrm{cst} - \frac{1}{n}\sum \log \rho_\theta(x_i^*)$$

Fisher divergence

$$\mathrm{dist}(\rho_\theta, \rho_*) = \mathbb{E}_{X\sim\rho_*} |\nabla \log \rho_*(X) - \nabla \log \rho_\theta(X)|^2$$
$$\approx \frac{1}{n}\sum |\nabla \log \rho_*(x_i^*) - \nabla \log \rho_\theta(x_i^*)|^2$$

Other losses?

Bregman, Wasserstein, etc.

Training procedures I: max-likelihood

Gradient ascent on Energy-Based Models

Goal: maximize $L(\theta) = \mathbb{E}_*[\log \rho_\theta] = -\mathbb{E}_*[U_\theta + \log Z_\theta]$

$$\nabla_\theta L(\theta) = -\mathbb{E}_*[U_\theta] - \nabla \log Z_\theta$$

Computation of $\nabla_\theta \log Z_\theta$:

$$\frac{\nabla_\theta Z_\theta}{Z_\theta} = \int -\nabla_\theta U_\theta(x) e^{-U_\theta(x)} \frac{1}{Z_\theta} dx = -\mathbb{E}_\theta[\nabla_\theta U_\theta]$$

Gradient of the log-likelihood

$$\nabla_\theta L(\theta) = \mathbb{E}_\theta[\nabla_\theta U_\theta] - \mathbb{E}_*[\nabla_\theta U_\theta]$$

Gradient ascent with stepsize $\eta > 0$:

$$\theta_{t+1} - \theta_t = \eta \times \left(\mathbb{E}_{\theta_t}[\nabla_\theta U_{\theta_t}] - \mathbb{E}_*[\nabla_\theta U_{\theta_t}]\right)$$

$$\nabla_\theta L(\theta) = (\mathbb{E}_\theta[\nabla_\theta U_\theta(X)] - \mathbb{E}_*[\nabla_\theta U_\theta(X)])$$

$\mathbb{E}_*[\nabla_\theta U_\theta]$

$x_i^* =$ "positive samples"

from ρ_*

$\approx \frac{1}{n}\sum_i U_{\theta_t}(x_*^i)$

$\mathbb{E}_{\theta_t}[\nabla_\theta U_\theta]$

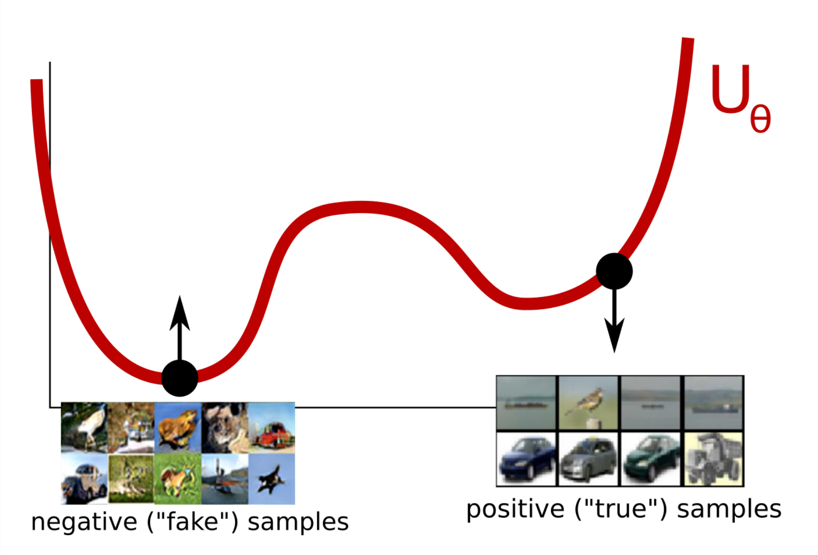
$y_i =$ "negative samples"

from ρ_{θ_t}

$\approx \frac{1}{n}\sum_i U_{\theta_t}(y_i)$

"contrastive learning" :

- pull down the energy of positive samples, $\mathbb{E}_*[U_\theta]$
- pull up the energy of negative samples, $\mathbb{E}_{\theta_t}[U_\theta]$


U_θ
negative ("fake") samples
positive ("true") samples

MCMC sampling is too costly

Q: at each gradient step, how do we get the negative samples for computing $\mathbb{E}_\theta[\nabla_\theta U_\theta]$?

A: using MCMC/Langevin methods...

At step t, initialize X_0^i ("walkers"), then for $\tau = 0, \ldots, T_{mix}$,

$$X_{\tau+1}^i = X_\tau^i - \eta \nabla_x U_\theta(X_\tau^i) + \sqrt{2\eta}\xi_\tau$$

and estimate

$$\mathbb{E}_{\theta_t}[U_{\theta_t}] \approx \frac{1}{N_{walkers}} \sum_{i=1}^{N_{walkers}} U_{\theta_t}(X_{T_{mix}}^i).$$

MCMC sampling is too costly

Q: at each gradient step, how do we get the negative samples for computing $\mathbb{E}_\theta[\nabla_\theta U_\theta]$?

A: using MCMC/Langevin methods...

At step t, initialize X_0^i ("walkers"), then for $\tau = 0, \ldots, T_{mix}$,

$$X_{\tau+1}^i = X_\tau^i - \eta \nabla_x U_\theta(X_\tau^i) + \sqrt{2\eta}\xi_\tau$$

and estimate

$$\mathbb{E}_{\theta_t}[U_{\theta_t}] \approx \frac{1}{N_{walkers}} \sum_{i=1}^{N_{walkers}} U_{\theta_t}(X_{T_{mix}}^i).$$

If T_{mix} is large, this is too costly.

Each gradient ascent step will consume T_{mix} MCMC sampling steps for each of the $N_{walkers}$ chains!

Contrastive Divergence with k steps (CD-k), [Hinton 2005]

- don't let the chain reach T_{mix} steps. Use only k steps ($k = 1$).
- initialize each chain directly at the training points $\{x_*^i\}$.

Contrastive Divergence with k steps (CD-k), [Hinton 2005]

- don't let the chain reach T_{mix} steps. Use only k steps ($k = 1$).
- initialize each chain directly at the training points $\{x_*^i\}$.

[Hyvarinen 2007]
in the limit of small noise $\eta \to 0$, CD-1 = score matching.

[Yair and Michaeli 20] CD-1 is an adversarial game

[Agoritsas et al 23] Effect of non-convergent sampling

Persistent Contrastive Divergence (PCD), [Tieleman 2008]

- don't let the chain reach T_{mix} steps. Use only k steps ($k = 1$).
- ~~Initialize each chain directly at the training points $\{x_*^i\}$.~~
- initialize each chain directly where the previous chain ended.

Persistent Contrastive Divergence (PCD), [Tieleman 2008]

- don't let the chain reach T_{mix} steps. Use only k steps ($k = 1$).
- ~~Initialize each chain directly at the training points $\{x_*^i\}$.~~
- initialize each chain directly where the previous chain ended.

Practically: maintain a set of *walkers* X_t^i. At step $t+1$,
1) approximate $\mathbb{E}_{\theta_t}[U_{\theta_t}] \approx \frac{1}{n}\sum_{i=1}^{N} U_{\theta_t}(X_t^i)$,
2) compute θ_{t+1} using the approximation,
3) move the walkers with $X_{t+1} = X_t - \eta \nabla U_{\theta_{t+1}}(X_t) + \sqrt{2\eta}\xi$

$\Rightarrow$ leads to mode collapse, we'll see why in the last section.

Replay buffer techniques [Du and Mordatch 2019]

- don't let the chain reach T_{mix} steps. Use only k steps ($k = 1$).
- ~~Initialize each chain directly at the training points $\{x_*^i\}$.~~
- ~~initialize each chain directly where the previous chain ended.~~
- initialize, sometimes from the past, sometimes from pure noise

+ many other methods (ask Davide Carbone for our method using Jarzynski's identity !)

Training procedures
II: alternative losses

a Noise Contrastive methods

b GANs

c Score Matching

d Denoising score matching

a. Noise Contrastive Estimation [Gutmann & Hyvarinen 2010]

Idea: - get another dataset y_i, of `fake` samples from a known distribution μ.

- train a binary classifier to distinguish between `true` samples x_i^* and `fake` samples y_i.

Bayes' rule gives the optimal classifier D_{opt}:

$$D_{\mathrm{opt}}(x) = \mathbb{P}(\mathtt{true} \mid x) = \frac{p(x \mid \mathtt{true})}{p(x \mid \mathtt{fake}) + p(x \mid \mathtt{true})}$$
$$= \frac{\rho_*(x)}{\rho_*(x) + \mu(x)}$$

a. Noise Contrastive Estimation [Gutmann & Hyvarinen 2010]

Idea: - get another dataset y_i, of fake samples from a known distribution μ.

- train a binary classifier to distinguish between true samples x_i^* and fake samples y_i.

Bayes' rule gives the optimal classifier D_{opt}:

$$D_{\mathrm{opt}}(x) = \mathbb{P}(\texttt{true} \mid x) = \frac{p(x \mid \texttt{true})}{p(x \mid \texttt{fake}) + p(x \mid \texttt{true})}$$
$$= \frac{\rho_*(x)}{\rho_*(x) + \mu(x)}$$

Reminder: the *logistic regression* loss for training a classifier D_θ is

$$R(\theta) = -\mathbb{E}_{x\sim\texttt{true}} \log D_\theta(x) - \mathbb{E}_{y\sim\texttt{fake}} \log(1 - D_\theta(y))$$

NCE Strategy

1) Set your discriminator as

$$D_\theta(x) = \frac{F_\theta(x)}{(F_\theta(x) + \mu(x))}$$

with $F_\theta(x) = e^{-U_\theta(x)}$

2) then train using the logistic regression loss

$$R(\theta) = \frac{1}{n}\sum_{i=1}^{n} \log D_\theta(x_i^*) + \log(1 - D_\theta(y_i))$$

$$D_{\theta_*} \approx D_{\mathrm{opt}} \quad \Rightarrow \quad F_{\theta_*} \approx \rho_*$$

Extra rizz: the normalization $\int e^{-U_\theta} = 1$ is automatic!

Limitations of NCE

- if μ is too close to ρ_* then training the classifier is too difficult
- if μ is too different to ρ_* then classifying is too easy, there are near-optimal classifiers very different than the optimal one

the GAN idea

$\Rightarrow$ also train a "fake sample generator", say μ_β, instead of using always the same fixed generator μ

b. Generative Adversarial Networks [Goodfellow 2014]

GAN objective

$$\max_{\theta} \min_{\beta} \mathbb{E}_{*}[\log D_{\theta}(x)] + \mathbb{E}_{y \sim \mu_{\beta}}[\log(1 - D_{\theta}(y))]$$

b. Generative Adversarial Networks [Goodfellow 2014]

GAN objective

$$\max_{\theta} \min_{\beta} \mathbb{E}_{*}[\log D_{\theta}(x)] + \mathbb{E}_{y \sim \mu_{\beta}}[\log(1 - D_{\theta}(y))]$$

Min-Max optimization is very hard to stabilize:

1. The gradients with respect to β can easily vanish as long as the two distributions are slightly different, due to the log. That led to Wasserstein GANs [Arjovsky et al., 2017].
2. "Mode colapse" phenomena, where entire regions of the true distribution are forgotten, are very frequent.
3. A tiny change in architecture can completely break a working, stable training procedure. Hyperparameter fine-tuning is hard.

See [Salimans et al. 2016] for many training tips, mostly empirical.

c. Score Matching [Hyvarinen 2005]

c. Score Matching [Hyvarinen 2005]

Goal: minimize $SM(\theta) = \mathbb{E}_*[|\nabla \log \rho_\theta - \nabla \log \rho_*|^2]$.

Hyvarinen 2005

$$SM(\theta) = \mathrm{cst} + \mathbb{E}_*[|\nabla \log \rho_\theta|^2 + 2\Delta \log \rho_\theta].$$

- Parametrize the score $\nabla \log \rho_\theta$ with a neural network s_θ
- Minimize $\mathbb{E}_*[|s_\theta|^2 + 2\nabla_x \cdot (s_\theta)]$ using gradient descent

Problem 1: for $\nabla_\theta SM(\theta)$ we need to compute "double derivatives" like

$$\nabla_\theta \nabla_x \cdot s_\theta(x).$$

Problem 2: inferring $\log \rho$ from $s_\theta \approx \nabla \log \rho_\theta$?

Proof of Hyvarinen's identity: it's just an integration by parts.

For p, q two smooth densities with fast decay at ∞,

$$
\begin{aligned}
\mathbb{E}_p[|\nabla \log p - \nabla \log q|^2] &= \int p|\nabla \log p - \nabla \log q|^2 \\
&= \int p|\nabla p/p - \nabla q/q|^2 \\
&= c_p + \int p|\nabla q/q|^2 - 2\int p\frac{\nabla p}{p} \cdot \nabla \log q \\
&= c_p + \int p|\nabla q/q|^2 - 2\int \nabla p \cdot \nabla \log q \\
&= c_p + \int p|\nabla \log q|^2 + 2\int p\nabla \cdot \nabla \log q \\
&= c_p + \mathbb{E}_p[|\nabla \log q|^2 + 2\Delta \log q]
\end{aligned}
$$

d. Denoising Score Matching [Vincent 2009]

Let us corrupt the original samples with noise:

$$x_i^{\texttt{noisy}} = x_i^* + \epsilon_i \qquad \epsilon_i \sim \mathcal{N}(0, \sigma^2)$$

distribution of $x_i^{\texttt{noisy}}$ is a convolution $\rho_{\texttt{noisy}} = \rho_* * \mathcal{N}$.

Vincent 2009

$$SM(\theta) = \mathrm{cst} + \mathbb{E}_{X \sim \rho_*, \varepsilon \sim g}[|\nabla \log g(\varepsilon) - \nabla \log \rho_\theta(X + \varepsilon)|^2].$$

- Parametrize the score $\nabla \log \rho_{\texttt{noisy}}$ of the noisy distribution with s_θ (NN)
- Minimize $\mathbb{E}[|\epsilon/\sigma - s_\theta(X + \epsilon)|^2]$
- $\Rightarrow$ no double derivatives!
- BUT you don't learn ρ_* but $\rho_{\texttt{noisy}}$. Is it easier to denoise $\rho_{\texttt{noisy}}$ than to learn ρ_*?

Limitations of SM

1) In the presence of high energy barriers, SM methods and variants cannot learn the relative weights of the modes and/or lead to mode collapse.

2) The score $s_{\theta_*} \approx \nabla \log \rho_*$ does not give direct access to the density!

3) Results were good... but not as good as GANs

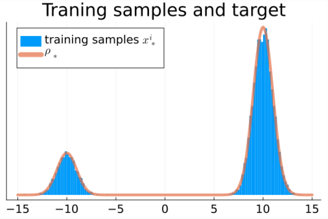
Training procedures III: getting insight from toy models

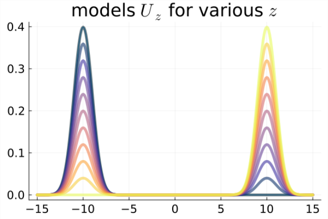
Model: all gaussian mixtures with modes $a = -10, b = 10$:

$$U_\theta(x) = -\log\left(e^{-|x-a|^2/2} + e^{-\theta}e^{-|x-b|^2/2}\right)$$

$$Z_\theta = (1+e^{-\theta})\sqrt{2\pi}$$

$$\rho_\theta(x) = \frac{e^{-|x-a|^2/2} + e^{-\theta}e^{-|x-b|^2/2}}{(1+e^{-\theta})\sqrt{2\pi}}$$





Target: $\rho_* = \rho_{\theta_*}$ for some θ_* with $q_* = \frac{e^{-\theta_*}}{1+e^{-\theta_*}} \approx 0.8$.

Gradient flow (continuous version of the discrete gradient descent):

$$\dot{\theta}(t) = -\nabla_\theta \, \mathrm{loss}(\rho_*, \rho_{\theta(t)})$$

where loss is one of the various objectives above.

We'll see:

- variational loss (KL)
- score matching
- persistent contrastive divergence

Useful approximations

In the following, we will need to compute quantities like $\nabla_\theta U_\theta(x)$ and $\nabla_x U_\theta(x)$. They actually assume a super simple form.

Useful approximations

In the following, we will need to compute quantities like $\nabla_\theta U_\theta(x)$ and $\nabla_x U_\theta(x)$. They actually assume a super simple form.

$$\nabla_x U_\theta(x) = \frac{(x-a)e^{-(x-a)^2/2} + e^{-\theta}(x-b)e^{-(x-b)^2/2}}{e^{-(x-a)^2/2} + e^{-\theta}e^{-(x-b)^2/2}}$$
$$\approx (x-a)1_{x \text{ close to } a} + (x-b)1_{x \text{ close to } b}$$

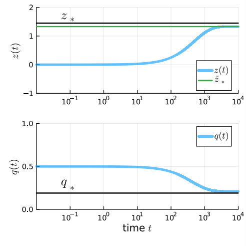
$$\nabla_\theta U_\theta(x) = \frac{e^{-\theta}e^{-(x-b)^2/2}}{e^{-(x-a)^2/2} + e^{-\theta}e^{-(x-b)^2/2}} \approx 1_{x \text{ is close to } b}$$

$$\forall \theta, w \qquad \mathbb{E}_w[\nabla_\theta U_\theta] \approx \mathbb{P}_w(\text{ mode } b) = \frac{e^{-w}}{1+e^{-w}}$$

Success of max-likelihood

$$\dot{\theta}(t) = \mathbb{E}_{\theta(t)}[\nabla_\theta U_{\theta(t)}] - \mathbb{E}_{\theta_*}[\nabla_\theta U_{\theta(t)}]$$
$$\approx \frac{e^{-\theta(t)}}{1+e^{-\theta(t)}} - \frac{e^{-\theta_*}}{1+e^{-\theta_*}}.$$

Clearly this system converges towards its unique FP $\theta(t) = \theta_*$.



Failure of score matching

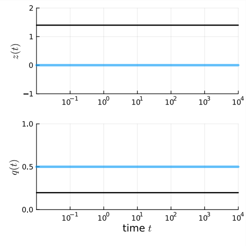
$$\dot{\theta}(t) = -\nabla_\theta SM(\theta) = \nabla_\theta \mathbb{E}_{\theta_*}[|\nabla \log \rho_{\theta(t)} - \nabla \log \rho_{\theta_*}|^2]$$

Remember that

$$\nabla \log \rho_\theta(x) \approx (x-a)1_{x \text{ close to } a} + (x-b)1_{x \text{ close to } b}$$

$\Rightarrow \nabla \log \rho_\theta(x)$ does not depend on θ, hence $\nabla_\theta SM(\theta) \approx 0$.

This leads to the "no learning" phenomenon $\dot{\theta}(t) \approx 0$



NCE fails

Let us choose as fake data generator a Gaussian mixture with modes a, b and parameter $\mu \neq \theta_*$. Recall that $D_\theta(x) = e^{-U_\theta(x)}/(e^{-U_\theta(x)} + e^{-U_\mu(x)})$. The NCE flow is

$$\dot{\theta}(t) = \mathbb{E}_*[\log D_{\theta(t)}(x)] + \mathbb{E}_\mu[\log(1 - D_{\theta(t)}(y))]$$

$$\dot{\theta}(t) = -\frac{e^{-\theta_*}}{1 + e^{-\theta_*}} + \frac{e^{-\theta_*}}{1 + e^{-\theta_*}} \frac{e^{-\theta(t)}}{e^{-\theta(t)} + e^{-\mu}} + \frac{e^{-\mu}}{1 + e^{-\mu}} \frac{e^{-\theta(t)}}{e^{-\theta(t)} + e^{-\mu}}$$

Do the computations: you will find that the only fixed point is given by

$$\theta = \mu + \log\left(\frac{1 + e^{-\theta_*}}{1 + e^{-\mu}}\right) \neq \theta_*$$

Mode collapse in PCD

Here the negative samples are generated using

$$dX_t = -\nabla_x U_{\theta(t)}(X_t)dt + \sqrt{2}dB_t$$

Remember that

$$\nabla_x U_\theta(x) \approx (x-a)1_{x \text{ close to } a} + (x-b)1_{x \text{ close to } b}$$

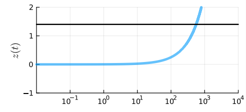
X_t close to $b \Rightarrow dX_t \approx -(X_t - b)dt + \sqrt{2}dB_t$: Ornstein-Uhlenbeck
X_t close to $a \Rightarrow dX_t \approx -(X_t - a)dt + \sqrt{2}dB_t$: Ornstein-Uhlenbeck

There is no transfer of walkers between modes a and b!

The distribution of X_t does not change and is equal to $\rho_{\theta(0)}$:

$$\dot{\theta}(t) \approx \frac{e^{-\theta(0)}}{1+e^{-\theta(0)}} - \frac{e^{-\theta_*}}{1+e^{-\theta_*}} = \mathrm{cst}$$

This leads to mode collapse, $\theta(t) \to \pm\infty$.



That's all for old-fashioned GenAI.

Next lesson: probabilistic paths techniques.

A personal curated list of references

How to train your EBMs (Song & Kingma)

Score Matching (Hyvarinen)

Denoising score matching (Vincent)

Noise contrastive estimation (Gutman & Hyvarinen)

Improved CD (Du & al.)

'The' GAN paper (Goodfellow & al.)

'The' GAN training paper (Salimans & al.)

Wasserstein GANs (Arjovsky & al.)

Efficient training of EBMs (Carbone & al.)

From SM to diffusion models (Song & Ermon)