Generative modelling witch score function*

Score-Based model

Florentin Coeurdoux

May 2, 2022

Contents

*florentin.coeurdoux@irit.fr

1 Generative modelling witch score function

The key idea is to model the gradient of the log-probability density function also known as (stein) the score function:

$$\nabla_{\mathbf{x}} \log p(\mathbf{x})$$

Likelyhood based model (autoregressive, normalizing flows, energy based model and variational auto-encoders) also they allow density estimation, they are constrained by the model architecture. They usualy required strong restrictions on the model architecture to ensure a tractable noramalizing constant, or must rely on surrogate objectives to approximate likelyhood. Adversarial model (GAN) also they allows good sampling quality, they suffer from difficult training procedure, lead to mode collaps and do not allows density estimation. Score based model allows both quality sampling and density estimation, by creating a map to navigate the data space. We can train score based model to minimize Fisher divergence, this objective function is known as the score matching. Sampling is made using Langevin Dynamics and improved by adding noise perturbations in the training procedure

When the number of noise scales approaches infinity, we essentially perturb the data distribution with continuously growing levels of noise. In this case, the noise perturbation procedure is a continuous-time stochastic process, which are are solutions of SDE. Such models are named score-based SDE or Probability flow ODE Score-based model also allows conditional sampling with controllable generation

2 Score function

Likelihood based model are trained to directly fit the parameter of probability density function :

$$p_\theta(\mathbf{x}) = \frac{e^{-f_\theta(\mathbf{x})}}{\underbrace{Z_\theta}_{\text{normalizing const}}}$$

$Z_\theta > 0$ such that $\int p_\theta(\mathbf{x}) \mathrm{d}\mathbf{x} = 1$.

Unfortunately Z_θ is not tractable. Usually the used techniques is to restrict the model architecture to make Z_θ tractable (normalizing flows) or approximate Z_θ (variational auto-encoders)

Score function do not need normalizing constant to be tractable

$$\mathbf{s}_\theta(\mathbf{x}) = \nabla_{\mathbf{x}} \log p_\theta(\mathbf{x}) = -\nabla_{\mathbf{x}} f_\theta(\mathbf{x}) - \underbrace{\nabla_{\mathbf{x}} \log Z_\theta}_{=0} = -\nabla_{\mathbf{x}} f_\theta(\mathbf{x})$$

3 Score matching function

We can train score based model to minimize Fisher divergence (see **fischer information** in crlb between the model and the data distributions) :

$$\mathbb{E}_{p(\mathbf{x})}\left[\|\nabla_{\mathbf{x}} \log p(\mathbf{x}) - \mathbf{s}_\theta(\mathbf{x})\|_2^2\right]$$

Unfortunately this is not directly computable because $p(x)$ and thus $\nabla_{\mathbf{x}} \log p(\mathbf{x})$ is unknown. Fortunately, it exist score matching techniques such as **Implicit Score Matching** and **Denoising Score Matching** that minimises the Fisher distance without needed $\nabla_{\mathbf{x}} \log p(\mathbf{x})$ ground truth (Song and Ermon 2019).

3.1 Implicit score matching

A computable version of the Fisher Divergence also known as Score matching has been introduce by (Hyvärinen 2005). Using a series of integration by part Hyvarinen managed to derived a computable score matching

Although it is computable, it does not scale with dimension as the computation as a complexity of $O(D)$ with D the dimension of the input data x.

This is exactly why (Song et al. 2020) proposed a sliced version of the score matching to treat it like multiples one dimensional problems (just like the sliced Wasserstein).

3.1.1 Estimation of Non-Normalized Statistical Models by Score Matching

This is a quick summary of the paper (Hyvärinen 2005). Recall, our goal is to compute the Fisher divergence :

$$\begin{aligned}
&\frac{1}{2}\mathbb{E}_{p_{\text{data}}}\left[\left(\nabla_x \log p_{\text{data}}\,(x) - \nabla_x \log p_\theta(x)\right)^2\right] \\
=&\frac{1}{2}\int p_{\text{data}}\,(x)\left(\nabla_x \log p_{\text{data}}\,(x) - \nabla_x \log p_\theta(x)\right)^2\,\mathrm{d}x
\end{aligned}$$

The proposed approach used integration by parts

$$\begin{aligned}
=&\frac{1}{2}\int p_{\text{data}}\,(x)\left(\nabla_x \log p_{\text{data}}\,(x) - \nabla_x \log p_\theta(x)\right)^2\,\mathrm{d}x \\
=&\frac{1}{2}\int p_{\text{data}}\,(x)\left(\nabla_x \log p_{\text{data}}\,(x)\right)^2 + \frac{1}{2}\int p_{\text{data}}\,(x)\left(\nabla_x \log p_\theta(x)\right)^2\,\mathrm{d}x \\
&-\int p_{\text{data}}\,\nabla_x \log p_{\text{data}}\,(x)\nabla_x \log p_\theta(x)\mathrm{d}x
\end{aligned}$$

- LHS is constant because it does not depends on θ
- MHS is truncable because is does not involve the score of the data distribution $\nabla_x \log p_{\text{data}}\,(x)$

- RHS is the only term involving both scores

Let's then do another integration by part by only on the RHS:

$$
\begin{aligned}
& -\int p_{\text{data}}\ \nabla_x \log p_{\text{data}}\ (x) \nabla_x \log p_\theta(x) \mathrm{d}x \\
=& -\int p_{\text{data}}\ \frac{1}{p_{\text{data}}\ (x)} \nabla_x p_{\text{data}}\ (x) \nabla_x \log p_\theta(x) \mathrm{d}x \\
=& -\ p_{\text{data}}\ (x) \nabla_x \log p_\theta(x)|_{x=-\infty}^{\infty} + \int p_{\text{data}}\ (x) \nabla_x^2 \log p_\theta(x) \mathrm{d}x
\end{aligned}
$$

$$
\begin{aligned}
& -\int p_{\text{data}}\ \nabla_x \log p_{\text{data}}\ (x) \nabla_x \log p_\theta(x) \mathrm{d}x \\
=& -\int p_{\text{data}}\ \frac{1}{p_{\text{data}}\ (x)} \nabla_x p_{\text{data}}\ (x) \nabla_x \log p_\theta(x) \mathrm{d}x \\
=& -\ p_{\text{data}}\ (x) \nabla_x \log p_\theta(x)|_{x=-\infty}^{\infty} + \int p_{\text{data}}\ (x) \nabla_x^2 \log p_\theta(x) \mathrm{d}x
\end{aligned}
$$

- The LHS goes to zeros because $\lim_{x \to +\infty} p_\theta(x) = 0$

Now we can replace the RHS on the previous equation, and we obtain :

$$
\begin{aligned}
& \frac{1}{2} \mathbb{E}_{p_{\text{data}}}\ \left[\left(\nabla_x \log p_{\text{data}}\ (x) - \nabla_x \log p_\theta(x) \right)^2 \right] \\
=& \frac{1}{2} \int p_{\text{data}}\ (x) \left(\nabla_x \log p_{\text{data}}\ (x) \right)^2 + \frac{1}{2} \int p_{\text{data}}\ (x) \left(\nabla_x \log p_\theta(x) \right)^2\ \mathrm{d}x \\
& \int p_{\text{data}}\ (x) \nabla_x^2 \log p_\theta(x) \mathrm{d}x \\
& = \text{const} + \frac{1}{2} \mathbb{E}_{p_{\text{data}}}\ \left[\left(\nabla_x \log p_\theta(x) \right)^2 \right] + \mathbb{E}_{p_{\text{data}}}\ \left[\nabla_x^2 \log p_\theta(x) \right]
\end{aligned}
$$

This does not involve p_{data}

The final score matching can thus be rewritten :

$$
\mathbb{E}_{p_{\text{data}}}\ \left[\text{tr}(\ \underbrace{\nabla_{\mathbf{x}}^2 \log p_\theta(\mathbf{x})}_{\text{Hessian of } \log p_\theta(\mathbf{x})}\ ) + \frac{1}{2} \left\| \nabla_{\mathbf{x}} \log p_\theta(\mathbf{x}) \right\|_2^2 \right] + \text{const}
$$

Recall, our neural network take as input a vector x and output a function $f_\theta(\mathbf{x})$:

$$
p_\theta(\mathbf{x}) = \frac{e^{-f_\theta(\mathbf{x})}}{Z_\theta}
$$

Or we have :

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

So we can directly compute the RHS of the score matching function : $\frac{1}{2}\left\|\nabla_{\mathbf{x}} \log p_\theta(\mathbf{x})\right\|_2^2$ using simple backprop.

The LHS required the diagonal of the Hessian, which is also cumputable by computing backpop indenpently for each entrie of the vector x and sum this element to obtain the trace of the Hessian $\operatorname{tr}\left(\nabla_{\mathbf{x}}^2 \log p_\theta(\mathbf{x})\right)$. Although it is computable, it does not scale with dimention as the computation as a complexity oof $O(D)$ with D the dimention of the input data x.

This is eactly why Song Yang proposed a sliced version of the score matching to treat it like multiples one dimentional problems (just like the sliced Wasserstein).

3.1.2 Score matching using sliced score matching

(Song et al. 2020) reused the findings of (Hyvärinen 2005) to reduce the computation complexity of the implicit score matching.

The idea of the paper is:

- One dimensional problems should be easier.
- Projections onto random directions.

He proposed a Sliced version of the Fisher Divergence

$$\frac{1}{2}\mathbb{E}_{p_{\mathbf{v}}}\mathbb{E}_{p_{\text{data}}}\left[\left(\mathbf{v}^\top \nabla_{\mathbf{x}} \log p_\theta(\mathbf{x}) - \mathbf{v}^\top \nabla_{\mathbf{x}} \log p_{\theta\,\text{data}}\,(\mathbf{x})\right)^2\right]$$

where $\mathbf{v} \sim \mathcal{N}(0, \mathbf{I}) \in \mathbb{R}^d$ is a standard multivariate gaussian RV.

This objective can be efficiently compute the vector-jacobian product (VJP), anemely the LHS:

$$\mathbf{v}^\top \nabla_{\mathbf{x}}^2 \log p_\theta(\mathbf{x}) \mathbf{v} = \mathbf{v}^\top \nabla_{\mathbf{x}}\left(\mathbf{v}^\top \nabla_{\mathbf{x}} \log p_\theta(\mathbf{x})\right) = -\mathbf{v}^\top \nabla_{\mathbf{x}}\left(\mathbf{v}^\top \nabla_{\mathbf{x}} f_\theta(\mathbf{x})\right)$$

The required only one path of back propagation using only one randomly chosen dimension, the algorithm know has a complexity of $O(1)$

He also shows some theoretical results:

- Consistency for any number of projections
- Asymptotic convergence for any number of random projections

He proposed experiments using Normalizing Flows (NICE) and a modified version of Wasserstein auto-encoder.

3.2 Denoising Score Matching (DSM)

A different approach (Vincent 2011) investigated the "unsuspected link" between Score Matching and Denoising Autoencoders. This work is still used in the cutting edge score based model.

To get rid of :

$$\begin{aligned}&\frac{1}{2}\mathbb{E}_{p_{\text{data}}}\left[\left(\nabla_x \log p_{\text{data}}\left(x\right)-\nabla_x \log p_\theta(x)\right)^2\right]\\=&\frac{1}{2}\int p_{\text{data}}\left(x\right)\left(\nabla_x \log p_{\text{data}}\left(x\right)-\nabla_x \log p_\theta(x)\right)^2\ \mathrm{d}x\end{aligned}$$

DSM proposed to replace $p_{\text{data}}(x)$ by a noisy version of it denoted $p^\sigma_{\text{data}}(\tilde{\mathbf{x}}) =:$

$$p^\sigma_{\text{data}}(\tilde{\mathbf{x}}) = \int p^\sigma_{\text{data}}(\tilde{\mathbf{x}}, \mathbf{x}) d\mathbf{x}$$

with $p^\sigma_{\text{data}}(\tilde{\mathbf{x}}, \mathbf{x}) = p^\sigma_{\mathcal{N}}(\tilde{\mathbf{x}} \mid \mathbf{x})$

The denoising objective thus became:

$$J_{\text{D}}(\theta) = \mathbb{E}_{p^\sigma_{\text{data}}(\tilde{\mathbf{x}})}\left[\left\|\mathbf{s}_\theta(\tilde{\mathbf{x}}) - \nabla_{\tilde{\mathbf{x}}} \log p^\sigma_{\text{data}}\left(\tilde{\mathbf{x}}\right)\right\|_2^2\right]$$

The paper shows (it has mathematical proof) that we can have an equivalent of $J_{\text{D}}(\theta)$ as :

$$J_{\text{D}}(\theta) = \mathbb{E}_{p^\sigma_{\text{data}}\left(\tilde{\mathbf{x}}, \mathbf{x}\right)}\left[\left\|\mathbf{s}_\theta(\tilde{\mathbf{x}}) - \nabla_{\tilde{\mathbf{x}}} \log p^\sigma_{\mathcal{N}}(\tilde{\mathbf{x}} \mid \mathbf{x})\right\|_2^2\right]$$

Note that we now need original-corrupt data pairs $(\tilde{\mathbf{x}}, \mathbf{x})$ in order to compute the expectation, which is quite trivial to do. Also realize that the term $\nabla_{\tilde{\mathbf{x}}} \log p^\sigma_{\mathcal{N}}(\tilde{\mathbf{x}} \mid \mathbf{x})$ is not the data score but related only to the pre-specified noise model with quite an easy analytic form

$$\nabla_{\tilde{\mathbf{x}}} \log p^\sigma_{\mathcal{N}}(\tilde{\mathbf{x}} \mid \mathbf{x}) = -\frac{1}{\sigma^2}(\tilde{\mathbf{x}} - \mathbf{x})$$

The score function we learn this way isn't actually for our original data distribution $p_{\text{data}}\left(\mathbf{x}\right)$, but rather for the corrupted data distribution $p^\sigma_{\text{data}}\left(\tilde{\mathbf{x}}\right)$. The strength σ decided how well it aligns with the original distribution.

This can be interpreted as a unit vector from corrupted sample toward the real sample. The score is trying to learn how to denoise a corrupted sample, which is essentially what denoising auto-encoder do.

4 Naive score-based generative modeling and its pitfalls

Once the score-based model trained, we can use Langevin Dynamics to sample from it - Langevin techniques use the score function $\nabla_{\mathbf{x}} \log p(\mathbf{x})$, more specifically it is done by iterating a chain with arbitrary prior $\mathbf{x}_0 \sim \pi(\mathbf{x})$, and then iterates the following

$$\mathbf{x}_{i+1} \leftarrow \mathbf{x}_i + \epsilon \nabla_{\mathbf{x}} \log p(\mathbf{x}) + \sqrt{2\epsilon}\mathbf{z}_i, \quad i = 0, 1, \cdots, K$$

where $\mathbf{z}_i \sim \mathcal{N}(0, I)$. When $\epsilon \to 0$ and $K \to \infty, \mathbf{x}_K$ obtained from the procedure in (6) converges to a sample from $p(\mathbf{x})$ under some regularity conditions. In practice, the error is negligible when ϵ is sufficiently small and K is sufficiently large.

4.1 Pitfalls of classical score based sampling

Song shows in (Song and Ermon 2020) that sampling using Langevin Dynamics is not efficient when the training is done only using "clean" data, as the iterative sampling procedure get stuck when not located in high density region, which is really frequent when starting the markov chain.

The dataset does not cover the entire filed of possible values. Thus for the first steps of the Langevin Dynamics where the prior has hight chance to sample close to the visited space during training, $\mathbf{s}_\theta(\mathbf{x}) \approx \nabla_{\mathbf{x}} \log p(\mathbf{x})$ will give a random direction and it will be very hard to converge to good sample. (Song and Ermon 2020)

This phenomenon can be explained by the fact that the Fisher divergence is weighted by $p(x)$ as we can see on the following :

$$\mathbb{E}_{p(\mathbf{x})}\left[\|\nabla_{\mathbf{x}} \log p(\mathbf{x}) - \mathbf{s}_\theta(\mathbf{x})\|_2^2\right] = \int p(\mathbf{x}) \|\nabla_{\mathbf{x}} \log p(\mathbf{x}) - \mathbf{s}_\theta(\mathbf{x})\|_2^2 \, \mathrm{d}\mathbf{x}$$

Thus we largely ignore low density region where $p(x)$ is small even though it is very probable to start our Langevin-based sampling in this areas.

5 Score-based generative modeling with multiple noise perturbations

In order to get better quality samples Song propose to add noise perturbations in the training procedure Adding multiple noise scales is critical for the success of the method as it allows better sample quality, exact log-likelihood computation and controllable generation for inverse problem (Song and Ermon 2020).

The solution lies in the choice of the noise scale (o big and we learn nothing, to small and we keep our problem) To achieve good results, the training is performed at multiple scale. They use isotropic Gaussian of increasing variance $\sigma_1 < \sigma_2 < \cdots < \sigma_L$ and perturb the data distribution $p(x)$ with each Gaussian $\mathcal{N}\left(0, \sigma_i^2 I\right), i = 1, 2, \cdots, L$ to obtain a noise-perturbed distribution

$$p_{\sigma_i}(\mathbf{x}) = \int p(\mathbf{y}) \mathcal{N}\left(\mathbf{x}; \mathbf{y}, \sigma_i^2 I\right) \mathrm{d}\mathbf{y}$$

Note that we can easily draw samples from $p_{\sigma_i}(\mathbf{x})$ by sampling $\mathbf{x} \sim p(\mathbf{x})$ and computing $\mathbf{x} + \sigma_i \mathbf{z}$, with $\mathbf{z} \sim \mathcal{N}(0, I)$.

The new training procedure is a Noise Conditional Score-Based such that $\mathbf{s}_\theta(\mathbf{x}, i) \approx \nabla_{\mathbf{x}} \log p_{\sigma_i}(\mathbf{x})$, and is trained with the following weighted Fisher distance :

$$\sum_{i=1}^{L} \lambda(i) \mathbb{E}_{p_{\sigma_i}(\mathbf{x})} \left[\|\nabla_{\mathbf{x}} \log p_{\sigma_i}(\mathbf{x}) - \mathbf{s}_\theta(\mathbf{x}, i)\|_2^2 \right]$$

where $\lambda(i) = \sigma_i^2$.

After sampling we can sample from the learned model using Langevin dynamics from $i = L$ to 1 in sequence. This method is called annealed Langevin dynamics.

They are recommendations regarding the noise scale choice :

- Choose $\sigma_1 < \sigma_2 < \cdots < \sigma_L$ as a geometric progression ($L \simeq 1000$).
- $s_\theta(x, i)$ parametrized as a U-Net skip connections.
- Apply exponential moving average on the weights of a score-based model when used at test time.

Adding multiple noise scales is critical for the success of the method as it allows better sample quality, exact log-likelihood computation and controllable generation for inverse problem (Song and Ermon 2020).

6 Learning Scored-based model with stochastic differential equations (SDEs)

An SDE is an ODE (ordinary differential equation) plus Gaussian white noise:

$$\mathrm{d}\mathbf{x} = \mathbf{f}(\mathbf{x}, t)\mathrm{d}t + \underbrace{g(t)}_{\text{diffu coef}} \underbrace{\mathrm{d}\mathbf{w}}_{noise}$$

where $\mathbf{f}(\cdot, t) : \mathbb{R}^d \to \mathbb{R}^d$ is a vector-valued function called the drift coefficient, $g(t) \in \mathbb{R}$ is a real-valued function called the diffusion coefficient, we notes a standard Brownian motion, and $\mathrm{d}\mathbf{w}$ can be viewed as infinitesimal white noise. The solution of a stochastic differential equation is a continuous collection of random variables $\{\mathbf{x}(t)\}_{t \in [0,T]}$.

Annealed Langevin process has the ability to reverse the perturbation process. For infinite noise scales, we can analogously reverse the perturbation process for sample generation by using the reverse SDE.

6.1 Reversing the SDE for sample generation

Any SDE has a reverse SDE:

$$\mathrm{d}\mathbf{x} = \left[\mathbf{f}(\mathbf{x}, t) - g^2(t)\nabla_{\mathbf{x}} \log p_t(\mathbf{x})\right] \mathrm{d}t + g(t)\mathrm{d}\mathbf{w}$$

Remarque that the reverse SDE use the score $\nabla_{\mathbf{x}} \log p_t(\mathbf{x})$.

6.2 Estimating the reverse SDE with score-based models and score matching

In order to estimate the reverse SDE, we train a time-dependent score based model : $\mathbf{s}_\theta(\mathbf{x}, t) \approx \nabla_{\mathbf{x}} \log p_t(\mathbf{x})$.

The learning is done by minimizing the following Fisher information :

$$\mathbb{E}_{t\in\mathcal{U}(0,T)}\mathbb{E}_{p_t(\mathbf{x})}\left[\lambda(t)\left\|\nabla_{\mathbf{x}}\log p_t(\mathbf{x}) - \mathbf{s}_\theta(\mathbf{x},t)\right\|_2^2\right]$$

where $\mathcal{U}(0,T)$ denotes a uniform distribution over the time interval $[0,T]$, and $\lambda : \mathbb{R} \to \mathbb{R}_{>0}$ is a positive weighting function. Typically we use $\lambda(t) \propto 1/\mathbb{E}\left[\left\|\nabla_{\mathbf{x}(t)}\log p(\mathbf{x}(t) \mid \mathbf{x}(0))\right\|_2^2\right]$ to balance the magnitude of different score matching losses across time.

6.3 How to solve the reverse SDE

Once trained, the score van be directly plug into the reverse SDE:

$$\mathrm{d}\mathbf{x} = \left[\mathbf{f}(\mathbf{x},t) - g^2(t)\mathbf{s}_\theta(\mathbf{x},t)\right]\mathrm{d}t + g(t)\mathrm{d}\mathbf{w}$$

Solving reverse SDE can be done using different differential equation solver, among them we have : - Euler-Maruyama method :

$$\begin{aligned}\Delta\mathbf{x} &\leftarrow \left[\mathbf{f}(\mathbf{x},t) - g^2(t)\mathbf{s}_\theta(\mathbf{x},t)\right]\Delta t + g(t)\sqrt{|\Delta t|}\mathbf{z}_t\\ \mathbf{x} &\leftarrow \mathbf{x} + \Delta\mathbf{x}\\ t &\leftarrow t + \Delta t\end{aligned}$$

- Milstein method
- Stochastic Runge-Kutta method

We can apply MCMC approaches to fine-tune the trajectories obtained from numerical SDE solver. This technique is called Predictor Connector Sampler. The Predictor can be any SDE solver that can predict $\mathbf{x}(t+\Delta t) \sim p_{t+\Delta t}(\mathbf{x})$. The Corrector can be any MCMC procedure that relies on the score function (i.E Langevin Dynamics or Hammiltonian Monte Carlo). 1. Choose $\Delta t < 0$ 2. Predict $\mathbf{x}(t+\Delta t)$ based on current $\mathbf{x}(t)$ 3. Run MCMC Corrector to improve $\mathbf{x}(t+\Delta t)$ according to $\mathbf{s}_\theta(\mathbf{x}, t+\Delta t)$ to get higher quality sample 4. Repeat

Predictor-Corrector are currently the better architecture of score-based model (outperform StyleGAN2+ADA) even for hight dimensional data 1024x1024)

6.4 Probability flow ODE

Despite their very good generative capabilities, Predictor-Corrector models are not able to compute exact log-likelihood.

To solve this issue, Y.Song proposed in (Song et al. 2021) to convert SDE to ODE. The corresping ODE to an SDE is named **probability flow ODE**, given by:

$$\mathrm{d}\mathbf{x} = \left[\mathbf{f}(\mathbf{x}, t) - \frac{1}{2} g^2(t) \nabla_{\mathbf{x}} \log p_t(\mathbf{x})\right] \mathrm{d}t$$

They observe that trajectories obtained via SDE and ODE have the same marginal distribution.

The previous formulation is an example of continious normalizing flows FFJORD (Grathwohl et al. 2019) and a special case of neural ODE. Indeed, it convert a data posterior $p_0(\mathbf{x})$ to a prior distribution $p_T(\mathbf{x})$ and is fully invertible. As such, probability-flow ODE inherits all properties of continuous NF and thus the ability to explicitly compute exact log-likelihood.

This probability-Flow ODE achieve state of the art :

- density estimation
- controllable generation for inverse problem solving

7 Conditional sampling for inverse problem

Score-based generative models are particularly suitable for solving inverse problems which are Bayesian inference problems. Let $\mathbf{x}$ and $\mathbf{y}$ be two random variables, and suppose we know the forward process of generating $\mathbf{y}$ from $\mathbf{x}$, represented by the transition probability distribution $p(\mathbf{y} \mid \mathbf{x})$. The inverse problem is to compute $p(\mathbf{x} \mid \mathbf{y})$. From Bayes' rule, we have $p(\mathbf{x} \mid \mathbf{y}) = p(\mathbf{x})p(\mathbf{y} \mid \mathbf{x}) / \int p(\mathbf{x})p(\mathbf{y} \mid \mathbf{x})\mathrm{d}\mathbf{x}$. This expression can be greatly simplified by taking gradients with respect to $\mathbf{x}$ on both sides, leading to the following Bayes' rule for score functions:

$$\nabla_{\mathbf{x}} \log p(\mathbf{x} \mid \mathbf{y}) = \underbrace{\nabla_{\mathbf{x}} \log p(\mathbf{x})}_{\simeq s_\theta(x)} + \nabla_{\mathbf{x}} \underbrace{\log p(\mathbf{y} \mid \mathbf{x})}_{\text{known}}$$

Thus it only required a pretrained classifier and plug $p(\mathbf{y} \mid \mathbf{x})$ to the sampling procedure to be able to tackle this types of problem.

7.1 Challenges :

- Sampling speed is slow because of Langevin types iterations.
- Does not work well for discrete data distribution because of the continuous score function.

References

Grathwohl, Will, Ricky T. Q. Chen, Jesse Bettencourt, Ilya Sutskever, and David Duvenaud. 2019. "FFJORD: Free-Form Continuous Dynamics for Scalable Reversible Generative Models." *International Conference on Learning Representations*.

Hyvärinen, Aapo. 2005. "Estimation of Non-Normalized Statistical Models by Score Matching." *J. Mach. Learn. Res.*

Song, Yang, and Stefano Ermon. 2019. "Generative Modeling by Estimating Gradients of the Data Distribution." In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*.

———. 2020. "Improved Techniques for Training Score-Based Generative Models." In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, Virtual*.

Song, Yang, Sahaj Garg, Jiaxin Shi, and Stefano Ermon. 2020. "Sliced Score Matching: A Scalable Approach to Density and Score Estimation." In *Proceedings of the 35th Uncertainty in Artificial Intelligence Conference*. Proceedings of Machine Learning Research. PMLR.

Song, Yang, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. 2021. "Score-Based Generative Modeling Through Stochastic Differential Equations." In *International Conference on Learning Representations*.

Vincent, Pascal. 2011. "A Connection Between Score Matching and Denoising Autoencoders." *Neural Comput.*