Diffusion and Score-based Generative Models

Simone Petruzzi-1811872 Domenico Tersigni-

August 2023

1 Introduction

Given a dataset $\{x_1, x_2, ..., x_n\}$ where each point is drawn independently from an underlying distribution $p(x)$, the goal of generative modelling is to fit a model to the data distribution, such that we can synthetize new data points at will by sampling from the model of the distribution.
Let's say p_{data} is our model and $x_i \sim p_{data}$. We want to find a single probability distribution, minimizing the distance from p_{data} to p_θ. Afterwards we can generate samples from p_θ. In order to estimate the model we can use as starting point a Gaussian distribution (basically a computational graph composed by two layers). However a single Gaussian is too simple and we want to leverage on a bigger and deeper computational graph. Thus we want to use deep neural networks to represent complex data distributions. This kind of architectures typically convert high dimensional input into one dimensional output $f_\theta(x)$, this output should be converted into a probability density and for this reason we take the exponential of the output in order to make it always positive. Then we normalize by means of a constant Z_θ obtaining:

$$p_\theta(x) = \frac{e^{f_\theta(x)}}{Z_\theta} \tag{1}$$

Z_θ by definition should be computed by evaluating the high dimensional integral of the exponential function of out θ, over all possible values of x in the space:

$$Z_\theta = \int e^{f_\theta(x)}\, dx \tag{2}$$

In case of deep neural networks the computation of this integral becomes intractable (NP-complete problem). In order to tackle with the intractability of the normalizing constant, this proposal aims to work with **score functions**.

2 Score-based generative modeling

The score funcion is a vector field that gives the direction where the density fucntion grows most quickly. We define the score of a probability density $p(x)$ to be:

$$p(x) = \nabla_x \ log \ p(x) \tag{3}$$

Score function is good since in anytime given the density function we can compute the score function very easily by simply texting the dervative. Conversely given the score function we can recover the density function in principle by computing the integral. Thus, we argue that score function mantains preseved all mathematical information of the density function, being computationally more easy to deal with. Indeed with score functions we don't have any normalization restriction:

$$\nabla_x \ log \ p_\theta(x) = \nabla_x f_\theta(x) \ - \nabla_x \log Z_\theta \tag{4}$$

the term $\nabla_x \log Z_\theta$ is always 0 because it is the gradient of a constant. Thus the score function is the gradient of the deep neural network $f_\theta(x)$. We define the score model $s_\theta(x)$ as:

$$\nabla_x f_\theta(x) = s_\theta(x) \tag{5}$$

and we want to develop a technique that allows us to train $s_\theta(x)$, in order to estimate the underlying score function from a limited set of training data points

2.1 Score matching

We must train our score model to be close to our ground-truth data score function. Mathematically we can capture the distance between the two vector fields of score by the **Fisher divergence objective**:

$$\frac{1}{2} E_{p_{data}(x)}[||\nabla_x \ log \ p_{data}(x) - s_\theta(x)||_2^2] \tag{6}$$

However Fisher divergence can not be computed since we don't know the ground trith value of the data score function $\nabla_x \ log \ p_{data}(x)$. There is a way for addressing this challenge and the method is called **score matching**. Score matching uses integration by parts of Gauss' theorem to convert Fisher divergence into the following equivalent objective:

$$E_{p_{data}(x)}[\frac{1}{2}||s_\theta(x)||_2^2 + trace(\nabla_x \ s_\theta(x))] \tag{7}$$

where $\nabla_x \ s_\theta(x)$ denotes the Jacobian of $s_\theta(x)$. We call it score matching objective, and it is equivalent to the fisher divergence up to a costant. Since constants do not affect optimization, their score matching objective defines the same optimum as the Fisher divergence. In score matching there is no dependency on the score function of the data distribution anymore. Moreover the expectation in score matching can be efficiently be approximated by using empirical mean:

$$E_{p_{data}(x)}[\frac{1}{2}||s_\theta(x)||_2^2 + trace(\nabla_x \ s_\theta(x))] \approx \frac{1}{N} \sum_{i=1}^{N} [\frac{1}{2}||s_\theta(x)||_2^2 + trace(\nabla_x \ s_\theta(x_i))] \tag{8}$$

again this is not scalable to compute since we need one forward propagation to compute the first element of the score function output and we need a backpropagation to compute the first element on the diagonal of its Jacobian. This procedure has to be repeated multiple times until we recovered all diagonal elements on the Jacobian (trace is given by the sum over the diagonal elements). Thus, number of backpropagations needed is proportional to the dimenstionality of our data points. It follows that the whole procedure when modelling high dimensional data like images, requires a lot of backpropagations making this naive score matching approach not scalable.

2.2 Sliced score matching

This approach aims to solve the problem of naive score matching: the basic intuition is that we want to convert high dimensional problem to a one-dimensional one (since it is much easier to solve).
The idea is to leverage on random projection, this is done in order to approximate $trace(\nabla_x\, s_\theta(x))$ and get one-dimensional scalar fields. Again to capture this intuition we use sliced Fisher divergence:

$$\frac{1}{2}E_{p_v}E_{p_{data}(x)}[(\mathbf{v}^T\nabla_x\ log\ p_{data}(x) - \mathbf{v}^T s_\theta(x))^2] \tag{9}$$

also there we apply integration by parts and obtain sliced score matching:

$$E_{p_v}E_{p_{data}(x)}[\mathbf{v}^T\nabla_x s_\theta(x)\mathbf{v}\ + \frac{1}{2}||s_\theta(x)||_2^2] \tag{10}$$

$\mathbf{v}^T\ \nabla_x s_\theta(x)\mathbf{v}$ is much more scalable to compute. It is not hard to find that we can rewrite:

$$\mathbf{v}^T\nabla_x s_\theta(x)\mathbf{v} = \mathbf{v}^T\nabla_x(\mathbf{v}^T s_\theta(x)) \tag{11}$$

we just need one forward propagation to get the output $s_\theta(x)$ and then we can directly compute the inner product between $\mathbf{v}^T$ and s_θ.
Sliced score matching provides score estimation for the original *unperturbed* data distribution.

2.3 Denoising score matching

Another approach is denoising score matching which estimates the scores of *perturbed* data distribution.
Indeed, the idea here is to perturb the original data distribution employing a perturbation kernel q_σ, then estimate the score function of the noisy data density instead of the original one.

$$p_{data}(x)\ \longrightarrow q_\sigma(\tilde{x}|x)\ \longrightarrow q_\sigma(\tilde{x}) \tag{12}$$

We employ score matching to estimate the score of the perturbed data distribution $q_\sigma(\tilde{x})$, and when the perturbation is very small, we can approximately consider the score function of the noisy data density as equivalent to the score function of the noise-free data density. The denoising score matching objective function has been proved equivalent to the following

$$\frac{1}{2}\mathbb{E}_{p_{data}(x)}\mathbb{E}_{q_\sigma(\tilde{x}|x)}[||s_\theta(\tilde{x}) - \nabla_{\tilde{x}} log q_\sigma(\tilde{x}|x)||_2^2] \tag{13}$$

in this approach we must take care of the trade-off between the quantity of the noise injected. If you want to work well in estimating score functions of noise-free data densities, you need

a very small σ. However when σ is too small, the variance of the objective function becomes bigger and eventually explodes.

3 Langevin dynamics

Now that we have trained our score model to estimate the underlying score function $s_\theta \approx \nabla_x logp(x)$ we want a method for generating new data points from the given vector field of score functions.