Deep Learning & Optimization (ENSAE 3A-MS)- Project

ENSAE 2021/2022

Professeur: Marco Cuturi

Mastère Spécialisé - Data Science

Prisca BAHI Dounia CHEKEMBOU Jeffrey VERDIERE

Score-Based Generative Modeling through Stochastic Differential Equations

Song Sohl-Dickstein Kingma Kumar Ermon Poole

ICLR 2021

Contents

1 The problematic

We have (or plan to) introduce generative models in the lectures, notably GANs. In this work, you will study a recent, related, yet different approach to be able to synthesize new datapoints. This process does not rely on a low dimensional subset of vectors, but is rather aimed at generating new points in a certain space using noise sampled from the same space. This process was described in a recent ICLR'21 paper and summarized in the following blog post. In this memoir I ask you to summarize these findings and apply their method to generate points using a different dataset that those considered in the paper. While you can reuse parts of their code, I expect you to write original functions to test these methods on the dataset of your choice, and explain how you were able to tune hyperparameters.

2 Introduction

"What I cannot create, I do not understand." —Richard Feynman

In the world of statistics, we distinguish two types of models, the discriminative and generative models. While the first ones give the probability of the target according to the observation, the second ones tend to give the observations distribution in many cases without any labels or other information. Generative models are part of the second category. These algorithms can be considered as unsupervised learning methods and are often used to create new samples according to given data. Basically, their aim is to replicate similar data after studying its tendencies and pattern, but this process is hard. Indeed, when an algorithm is trying to draw human faces for example, finding the distributions that tell that eyes shouldn't be on foreheads is not a simple task. That's why they need to train more.

To learn this data probability distribution two type of algorithm exist: Likelihood-based models and implicit generative models. Both models work pretty much well on generating data but they do present some limits making their implementation harder. For instance, the likelihood model requires to have a knowledge on the normalizing constant,which not always easy to retrieve. Among the existing implicit methods there are the generative adversarial networks or GAN.

In this document, we present another way to study the probability distribution of a data set based on the score function. These models are more convenient because they can be trained independently of the normalizing constant and without any unstable adversarial training. Also, they are able to solve inverse problems without any re-training.

We will first present the score generative method introduced by Yang song in his research article Score-Based Generative Modeling through Stochastic Differential Equations then try to implement it on a new dataset.

3 First naive approach: score matching and Langevin dynamics sampling

Before diving into stochastic differential equations let's introduce the general setting of score matching and define some useful functions and notations.

3.1 The score function

Considering a dataset $(x_i)_{i=1..N}$ that follows a distribution that we denote $p(x)$, we want to find the appropriate and closest distribution to $p(x)$ in order to sample according to it. To do so, we define a real-valued function $f_\theta(x)$ and define:

$$p_\theta(x) = \frac{\exp -f_\theta(x)}{Z_\theta}$$

Where Z_θ is the normalizing constant that depends on θ, the learnable parameter to use in the algorithms to find the optimal distribution.

The score function is defined by:

$$\mathbf{s}(x) = \nabla_x log p(x)$$

Taking into account our learning rate θ, our model score function will be noted:

$$\mathbf{s}_\theta(x) = \nabla_{\mathbf{x}} \log p_\theta(\mathbf{x})$$

By construction it does only depend on $f_\theta(x)$. Indeed:

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

We are therefore training a score model $s_\theta(x)$ such that it gets closer to $\nabla_x \log\, p(x)$. We usually choose to train a neural network with the given loss function:

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

This is called the Fisher divergence.

Since we don't have access to the real value of the score function we use score matching methods that allow us to minimize the Fisher divergence without any knowledge of it. The training is done by stochastic gradient descent.

3.2 Langevin dynamics to sample data

Langevin dynamics is an iterative algorithm that allows us to sample according to a distribution by only knowing its score function. The algorithm goes like this:

1. We start with an arbitrary distribution $\mathbf{x}_0 \sim \pi(\mathbf{x})$ that we call the prior distribution.
2. for $i = 0$ to K we compute $\mathbf{x}_{i+1} \leftarrow \mathbf{x}_i + \epsilon \nabla_{\mathbf{x}} \log p(\mathbf{x}) + \sqrt{2\epsilon}\ \mathbf{z}_i$ with $\mathbf{z}_I \sim \mathcal{N}(0, I)$

This summerized first introduction to score generative methods helps us understand how the model is built and the density estimated. It is in a way better than GANs because it doesn't require adversial optimization but we will see later that it is not enough to generate accurate data.

3.3 Limits of the first naive approach

The first method we presented has been successful in modelling data samples but has however some limits.

Indeed, in some regions where data points are limiting the algorithm has a hard time minimizing the loss function. This issue seems logical since the fisher divergence is a ℓ_2 distance that is weighted by $p(x)$.

This approach is therefore not the optimal way to generate accurate data.

4 Second approach: using multiple noise perturbations in addition to the first approach

In order to give an accurate score estimation in regions that are low represented we can use noises to perturb the data. To avoid over corrupting the data with higher noises or no perturbation at all with small noises we apply many different noises to our distribution $p(x)$. Let's give an example:

Suppose we are given $\sigma_1 < \sigma_2 < \cdots < \sigma_L$ Gaussian noises. For $i = 1, ..., L$ we have:

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

Sampling through this formula can be done by simply sampling $\mathbf{x} + \sigma_i \mathbf{z}$ with $x \sim p(x)$ and $z \sim \mathcal{N}(0, I)$

The next step is to estimate the score function for each $p_{\sigma_i}(\mathbf{x})$. This can be done thanks to Noise Conditional Score-Based Models. $\mathbf{s}_\theta(\mathbf{x}, i)$ is usually parameterized with neural networks with a weighted sum of Fisher divergences as a loss function associated. It is defined as below:

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

with $\lambda(i)$ usually chosen as ${\sigma_i}^2$

Once the $\mathbf{s}_\theta(\mathbf{x}, i)$ estimated we run several Langevin dynamics introduced in section (3.2).

To summarize, adding multiple noises allow us to cover larger region and deal with the lack of data points sometimes.

5 Third approach: using stochastic differential equations (SDEs) in score-based generative modeling

5.1 Stochastic differential equations and their reverse

As we discussed earlier, adding noise is essential for the success of score-based generative models. Therefore, increasing the number of noises,improves the samples quality. Let's suppose we want to distort our data with an infinity of noises, this can be done thanks to stochastic processes.

A stochastic process is often the solution of the following stochastic differential equations:

$$\mathrm{d}\mathbf{x} = \mathbf{f}(\mathbf{x}, t)\mathrm{d}t + g(t)\mathrm{d}\mathbf{w},$$

where:

- $\mathbf{f}(\cdot, t) : \mathbb{R}^d \to \mathbb{R}^d$ is the drift
- $g(t) \in \mathbb{R}$ the diffusion coefficent
- $\mathbf{w}$ a standard brownian motion and $\mathrm{d}\mathbf{w}$ the white noise associated

As the solution to SDEs is the set of time indexed variables $\{\mathbf{x}(t)\}_{t\in[0,T]}$ we can denote $p_t(\mathbf{x})$ analogically to p_{σ_i} the probability density of $\mathbf{x}(t)$ in the case of infinite noise scales. We therefore have $p_0(\mathbf{x}) = p(\mathbf{x})$ the density at $t = 0$ and $p_T(\mathbf{x})$ the data distribution at T. With T large enough $p_T(\mathbf{x})$ tend to $\pi(\mathbf{x})$, the prior distribution

To very SDE we will then associate the corresponding reverse SDE that has the following form:

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

Reverse SDEs will allow us to sample data from noise.

5.2 Infinite noise scale perturbations

Before getting into the sample generation process let's use a simple example to explain how one can perturb data with infinite noise scales. For example:

$$\mathrm{d}\mathbf{x} = e^t \mathrm{d}\mathbf{w}$$

perturbs data with a gaussian noise of mean 0 and variance e^t

In finite noise scales setting we would use annealed Langevin dynamics to sample data. But when the perturbation are continuous, this algorithm is no longer appropriate.Indeed, reverse SDEs are applied to go from x_T which follows the prior distribution $\pi(x)$ to x_0 that will gives us the data distribution p_0 or p. To do so, we need to estimate $\nabla_{\mathbf{x}} \log p_t(\mathbf{x})$ and also to have the terminal distribution $p_T(\mathbf{x})$.

To find the perfect estimation of $\nabla_{\mathbf{x}} \log p_t(\mathbf{x})$ we use a time dependent score based model $\mathbf{s}_\theta(\mathbf{x}, t)$ such that $\mathbf{s}_\theta(\mathbf{x}, t) \approx \nabla_{\mathbf{x}} \log p_t(\mathbf{x})$ with the following objective function:

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

where:

- $\mathcal{U}(0, T)$ is a uniform distribution on [0,T]
- $\lambda : \mathbb{R} \to \mathbb{R}_{>0}$ is a positive weighting function

This model is trained thanks to score matching methods (sliced score matching for instance).

The score obtained will then be used in the reverse SDE to sample new data following the distribution of $x(0)$

5.3 Numerical SDEs solvers

There are several algorithms used to solve reverse SDEs. The one we will be presenting in this document is the Euler-Maruyala method. This numerical solver works by discretizing the equation. The steps are the following:

1. $\Delta\mathbf{x} \leftarrow [\mathbf{f}(\mathbf{x}, t) - g^2(t)\mathbf{s}_\theta(\mathbf{x}, t)]\Delta t + g(t)\sqrt{|\Delta t|}\mathbf{z}_t$
2. $\mathbf{x} \leftarrow \mathbf{x} + \Delta\mathbf{x}$
3. $t \leftarrow t + \Delta t$

Here, z_t is the gaussian noise applied at each step and Δt is a negative discrete time step.

In addition to it, we can also adjust and optimize the obtained trajectories thanks to correctors (Langevin dynamics for instance). Here is how it goes:

1. we predict $\mathbf{x}(t + \Delta t)$ thanks to our solver
2. we correct it by running several times our correcting method thanks to the score-based model $\mathbf{s}_\theta(\mathbf{x}, t + \Delta t)$
3. we obtain higher quality sample following the density probability $p_{t+\Delta t}(\mathbf{x})$

This is how stochastic differential equations are used for score-based generative models in a nutshell. Now that we have introduced the methods we will try to implement it on a new data set.

6 Implementation of the score based generative model through SDEs

In this section we propose an implementation of the score based generative model through SDEs.

6.1 Data

We chose to implement the studied model on a Zalando's data set that contains a training set of 60,000 examples and a test test of 10,000 examples of 28x28 grayscale image. [1]

6.2 Algorithm

We implemented the algorithm introduced by the author but applied some modification. Indeed, in order to solve the reverse SDE we use the Runge-Kutta method [2]. The order 1 Runge-Kutta method is a numerical solver defined as below :

$$w_0 = X_0$$

$$-w_{i+1} = w_i = a(w_i)\Delta t_i + b(w_i)\Delta W_i$$

$$+\frac{1}{2}[b(w_i + b(w_i)\sqrt{\Delta t_i} - b(w_i))](\Delta W_i^2 - \Delta t_i)/\sqrt{\Delta t_i}$$

We wanted to compare both Euler-Maruyama and Runge Kutta in terms of precision and maybe time execution.

6.3 Results

Both Euler-Maruyama and Runge Kutta provided the same computation time (approximately 3 seconds) on our shrunked data set. However the distribution of the generated data is different because the sampler give a different inputs.

Figure 1: Generate samples using the Euler method sampler with $\sigma = 25.0$

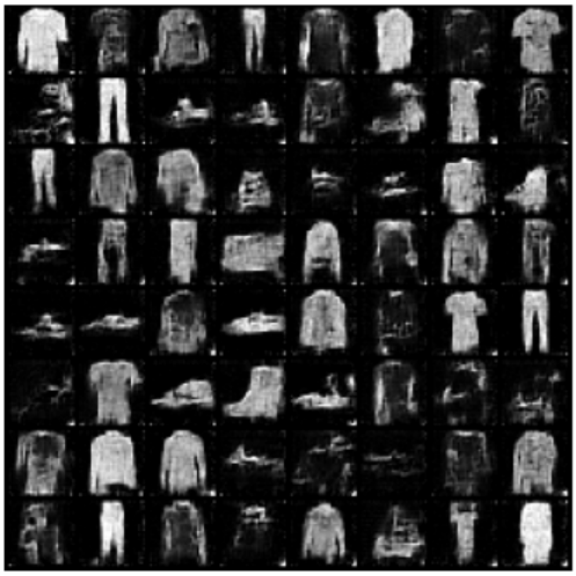

Figure 2: Generate samples using the Euler method sampler $\sigma = 25.0$

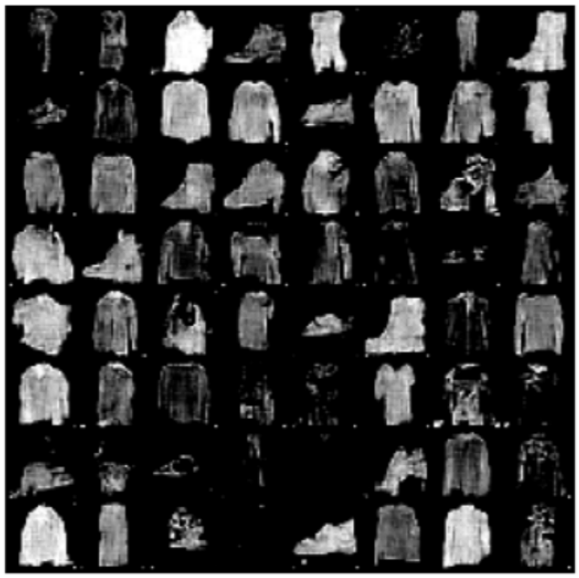

Figure 3: Generate samples using the Euler method sampler $\sigma = 50.0$

7 Conclusion

In this project, we worked on implementing score-based generative model based on the paper of Yann Song. We focused on implementing the paper on fashion MNIST data set to demonstrate its robustness on generating new data.

Particullary,we wanted to evaluate the impact of the solver methodology: Euler or Runge Kutta on the

speedness of the data generation.

References

[1] https://www.kaggle.com/datasets/zalando-research/fashionmnist. “Fashion MNIST Data Set”. In: ().

[2] Sauer. “Numerical Solution of Stochastic Differential Equations in Finance”. In: (2011).