Basics of Energy Based Models

July 18, 2022

Useful papers - [SK21], https://yang-song.github.io/blog/

1 Introduction

Here we will discuss another approach to the generative modelling problem. So, our main problem is to approximate the distribution of the real data $P_{data}(X)$ having some samples from it, i.e. $x_1, ..., x_n$ - our training set. As always, we define some parameterized model $P_\theta(X)$ to approximate $P_{data}(X)$ by varying the parameter θ. The energy based approach assumes the following form of the model distribution:

$$P_\theta(X) = \frac{e^{-E_\theta(X)}}{Z_\theta} \tag{1}$$

It is possible to ask a fair question: why so? Before i give some of my stupid thoughts about this question, first of all, let's say that Z_θ is a normalizing constant

$$Z_\theta = \int e^{-E_\theta(X)} dX \tag{2}$$

Because of this we satisfy the main property of the probability distribution

$$\int P_\theta(X) dX = \int \frac{e^{-E_\theta(X)}}{Z_\theta} dX = \frac{\int e^{-E_\theta(X)} dX}{\int e^{-E_\theta(X)} dX} = 1 \tag{3}$$

Let's return to the question. The first answer is related with the physics. This distribution is the Boltzmann (Gibbs) distribution defining the probability that the system (any physic system: micro system like atom or macro sytem like gas) is in a state of with energy E and temperature T:

$$P_i = \frac{e^{\frac{-E_i}{k_B T_i}}}{Z} \tag{4}$$

As it can be seen from the formula, the probability that the system is in a low-energy state is much higher than the probability of being in a high-energy state. That is, $P \uparrow$ when $E \downarrow$. This reflects one of the basic physical principles: **every system seeks to minimum energy**. Any system, even a person :). For example, if we consider an atom, electrons inside it tend to return to their main state from energized condition. Moreover, the exponent in this formula is necessary to provide the positivity of the probability density (because of $e > 0$). Note that we could write any positive function in the numerator:

$$P_i = \frac{F_i}{Z} \tag{5}$$

But if we want the energy (to make physics analogy) instead of F_i, we have to write the exponent due to the fact that the energy might be lower that zero. Ok, we have physics analogies, besides that it is possible to find some computational advantages of such density presentation. The main advantage is that the energy E_θ can be presented by any neural network (i mean with any architecture, any form, any shape, ...). This is extremely good achievement compared to others generative models: in VAE we have to define encoder and decoder (that is, we resticted form of the model), in GAN it is necessary to consider generator and discriminator, in normalizing flows we restricted by type of neural network layers and so on. However, we have a serious problem with the normalizing constant Z_θ, i.e. this integral over the space of data from the neural network in exponent is intractable. Below different approaches to solve this problem will be considered (Maximum Likelihood Training with MCMC, Score matching, Noise Contrastive Estimation).

2 Maximum Likelihood Training with MCMC

[DM19] De facto standard for learning probabilistic models from i.i.d. data is maximum likelihood estimation:

$$\mathbb{L}(\theta) = \mathbb{E}_{P_{data}(X)} \log P_\theta(X) \tag{6}$$

Maximizing this by θ gives us approximation of the P_{data}. Lets consider it in more detail, because we still have a problem with Z_θ

$$\begin{aligned}\mathbb{L}(\theta) = \int P_{data}(X) \log P_\theta(X) dX = \int P_{data}(X) \log \frac{e^{-E_\theta(X)}}{Z_\theta} dX \\ = -\int P_{data}(X) E_\theta(X) dX - \int P_{data}(X) \log Z_\theta dX \\ = -\int P_{data}(X) E_\theta(X) dX - \log Z_\theta \\ = -\int P_{data}(X) E_\theta(X) dX - \log \int e^{-E_\theta(X)} dX\end{aligned} \tag{7}$$

Lets take the gradient, because we have to solve the optimization problem.

$$\begin{aligned}\frac{\partial \mathbb{L}}{\partial \theta}(\theta) = -\int P_{data}(X) \frac{\partial E_\theta}{\partial \theta}(X) dX + \frac{1}{\int e^{-E_\theta(X)} dX} \int e^{-E_\theta(X)} \frac{\partial E_\theta}{\partial \theta}(X) dX \\ = -\int P_{data}(X) \frac{\partial E_\theta}{\partial \theta}(X) dX + \int P_\theta(X) \frac{\partial E_\theta}{\partial \theta}(X) dX\end{aligned} \tag{8}$$

So, finally we have

$$\frac{\partial \mathbb{L}}{\partial \theta}(\theta) = -\mathbb{E}_{P_{data}(X)} \frac{\partial E_\theta}{\partial \theta}(X) + \mathbb{E}_{P_\theta(X)} \frac{\partial E_\theta}{\partial \theta}(X) \tag{9}$$

Lets discuss this final formula. First of all, we have a gradient of the energy by its parameters $\frac{\partial E_\theta}{\partial \theta}$. This term can be easily calculated by autograd frameworks (backward propagation). Also, we have the expectation by the data distribution. As always, we approximate it by Monte-Carlo estimation (we can do it, because we have samples - it is our training set):

$$\mathbb{E}_{P_{data}(X)} \frac{\partial E_\theta}{\partial \theta}(X) \approx \frac{1}{n}\sum_{i=1}^{n} \frac{\partial E_\theta}{\partial \theta}(x_i),\ i \sim \mathbb{U}(1, ..., n) \tag{10}$$

But, we have another expectation by the model distribution. Here the problem comes in. It is non trivial to obtain samples from the model distribution. So, we have to use Monte-Carlo Markov Chain to obtain these samples. Lets consider it in more detail.

2.1 Monte-Carlo Markov Chain

First of all, let's start with the Monte-Carlo. We used this approach above, to approximate the expectation. This is one of the main goal of the MC methods (approximation of the integral by a probability measure)

$$\mathbf{I} = \int P(X)f(X)dx \approx \frac{1}{n}\sum_{i=1}^{n} f(x_i) = \hat{\mathbf{I}},\ x_i \sim P(X) \tag{11}$$

Note that $\hat{I}$ is a random variable (as sum of random variables). We can consider some of its properties:

1. **Unbiasedness**. $\mathbb{E}\,\hat{\mathbf{I}} = \frac{1}{n}\sum \mathbb{E}\,f(x_i) = \mathbb{E}\,f(X) = \mathbf{I}$. This means that M-C estimation somehow converges to the true integral.

2. **Approximation error**. $\mathbb{D}\hat{\mathbf{I}} = \frac{1}{n^2}\sum \mathbb{D}f(x_i) = \frac{1}{n}\mathbb{D}f(X)$. Taking into account the central limit theorem, we can write that $\hat{\mathbf{I}} \sim \mathbb{N}(\hat{\mathbf{I}}|\,\mathbb{E}\,f(X), \frac{1}{n}\mathbb{D}f(X))$. Based on this, we can highlight the following important fact: The standard error of the normal distribution $\left(\sqrt{\frac{1}{n}\mathbb{D}f(X)}\right)$ is the **approximation error** of the initial integral. This follows from the fact that we want $\hat{\mathbf{I}} = \mathbf{I} = \mathbb{E}\,f(X)$, so if the variance of the normal distribution $(\mathbb{N}(\hat{\mathbf{I}}|\,\mathbb{E}\,f(X), \frac{1}{n}\mathbb{D}f(X)))$ is equal to zero, then we will obtain $\mathbb{E}\,f(X)$ during sampling (that's what we need). So, to reduce the approximation error we have to increase the number of samples n and also it would be nice if the function $f(X)$ is not fluctuating. The high fluctuation leads to high variance. If the function is constant $(\mathbb{D}(f(X)) = 0)$, then we can take only one sample and our estimation will have 100% accuracy.

3. **Approximation error does not depend on the space dimension**. As can be seen from $\left(\sqrt{\frac{1}{n}\mathbb{D}f(X)}\right)$ that our error does not depend on the space dimension, we have only number of samples and properties of the function. It is really useful property, because if we take a look to errors of other numerical integration methods we will see:

   - Trapezoidal rule. Error $\sim n^{-\frac{2}{d}}$ (here n is the number of nodes of the quadrature formula)
   - Simpson's rule. Error $\sim n^{-\frac{4}{d}}$

- Monte-Carlo. Error $\sim n^{-\frac{1}{2}}$

  As can be seen, the Monte-Carlo approach is losing to classical methods in the low dimension spaces. However, in the high dimension spaces (it is our case in the deep learning) it works much better.

So, the M-C is quite good. But we missed the most important thing: and can we even sample from $P(X)$. In the deep generative modelling (when $P(X)$ is a model) it is non-trivial problem. To do so, we have to consider Markov Chains.

2.1.1 Basics of Markov Chain

Let's give a definition of the Markov Chain:

Definition 1 *(Markov Chain)*

The Markov Chain is a ordered process of generation of the random variables. The joint distribution of the generated variables has the following form:

$$P(x_1, ..., x_n) = P_0(x_1)P_1(x_2|x_1)\dots P_n(x_n|x_{n-1}) \tag{12}$$

Here, P_0 is a base distribution, we sample x_1 from it. After, this sample start to participate in the process (Markov Chain). In other words, we move to another sample $x_2 \sim P_1(x_2|x_1)$ and so on. Doing this process several times we obtain x_n. As was said, our main goal is to produce samples from the distribution $P(X)$, so under some conditions it can be assumed that x_n is a sample from $P(X)$:). The first condition is a homogeneity:

Definition 2 *(Homogeneous Markov Chain)*

The Markov Chain is called homogeneous if and only if

$$P_i(x_i|x_{i-1}) = P(x_i|x_{i-1}), \forall i \tag{13}$$

It turns out that such Markov Chain (if $P(x_i|x_{i-1}) > 0, \forall x_i, x_{i-1}$ (such property called Irreducible) and the system never returns to the same state with a fixed period (called Aperiodic)) has a stationary state (distribution). In simple words it means that using the process of generating the random variables corresponding to Homogeneous Markov

Chain at some point we will obtain samples from stationary distribution. More formally:

$$\begin{aligned}
x_1 &\sim P_0(x_1) \\
x_2 &\sim P_1(x_2) = \int P(x_2|x_1)P_0(x_1)dx_1 \\
x_3 &\sim P_2(x_3) = \int P(x_3|x_2)P_1(x_2)dx_2 \\
&\dots \\
x_n &\sim P_{n-1}(x_n) = \int P(x_n|x_{n-1})P_{n-2}(x_{n-1})dx_{n-1} \\
x_{n+1} &\sim P_{n-1}(x_{n+1}) = \int P(x_{n+1}|x_n)P_{n-1}(x_n)dx_n \leftarrow \text{stationary distribution} \\
x_{n+2} &\sim P_{n-1}(x_{n+2}) = \int P(x_{n+2}|x_{n+1})P_{n-1}(x_{n+1})dx_{n+1} \leftarrow \text{stationary distribution}
\end{aligned} \tag{14}$$

It is pretty good, because if the stationary distribution is equal to our target distribution, then we will obtain samples from the target distribution. But how can we make sure that stationary distribution is equal to our target distribution. We can use the following theorem:

Theorem 2.1 *(Detailed Balance Equation)*

If the distribution $P(x)$ satisfies the following equation

$$P(x)P(x'|x) = P(x')P(x|x') \tag{15}$$

then the distribution $P(x)$ is stationary

So, it means that we to have check this equation for our target distribution, then the Markov Chain will converge to samples from the target distribution. Lets consider some of the MCMC algorithms.

2.1.2 Metropolis-Hastings method

Let $P(x) = \frac{\hat{P}(x)}{Z}$ - target distribution, $q(x|y) > 0$ - Markov Chain. (The support of $q(x|y)$ should match with the support of the $P(x)$ for the algorithm, im not sure that it is necessary). The Metropolis-Hastings algorithms has the following form:

1. Draw initial value $x_0 \sim P_0(x)$
2. for i = 1, ..., m , repeat:
   - Draw candidate $x^* \sim q(x^*|x_{i-1})$
   - Calculate $\alpha = \min\left(1, \frac{\hat{P}(x^*)q(x_{i-1}|x^*)}{q(x_*|x_{i-1})\hat{P}(x_{i-1})}\right)$
   - $x_i = x^*$ with probability α else $x_i = x_{i-1}$

Starting from some $n < m$ we will obtain samples from $P(x)$. We wander through the distribution area, choosing a point with a probability of improvement over the previous point ($\alpha = \frac{\hat{P}(x^*)}{\hat{P}(x_{i-1})}$ for symmetric case). If we only took steps with improvements, then sooner or later we would have stabilized in the distribution mode, and we want samples, but not optimal values. Lets consider a toy example, as in figure below. The target

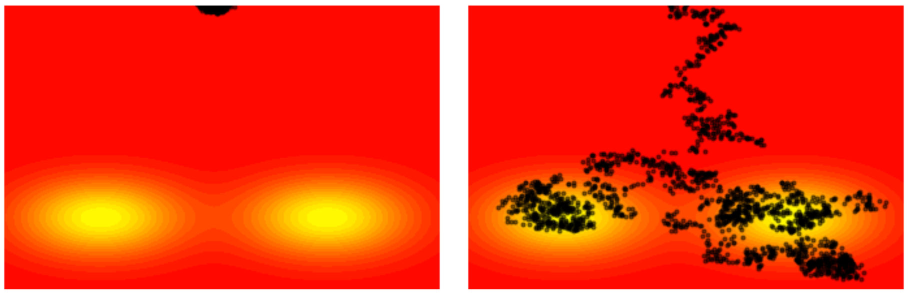

Figure 1: Target and base distribution (left). Samples from Markov Chain (right).

distribution is a mix of two gaussians. Base distribution is also gaussian, but it putted much higher than modes of the target distribution. As can be seen from the right figure, the Markov Chain converges to target distribution and gives nice samples.

2.1.3 Hamiltonian dynamics

Hamiltonian Monte Carlo (HMC) is a physics motivated algorithm for sampling from unnormalized distribution. Here we associate random variable with some particle because of the fact that they have a lot in common. In exact, random variable tends to be in places with the highest probability density, opposite a particle more likely to be in places with the lowest potential energy (as we said, everything seeks to minimum energy). These properties are shown below in Figure 2. Thus, we have an accordance

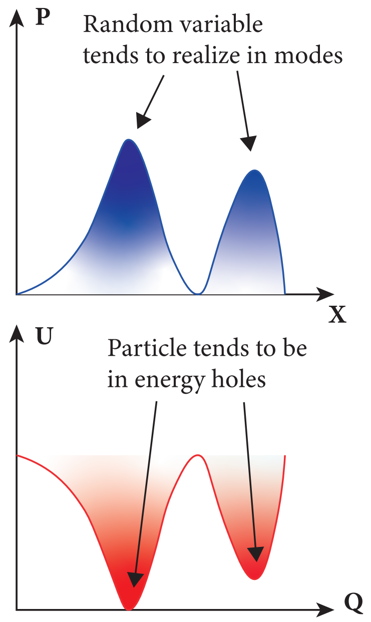

Figure 2: Comparison between behaviour of random variable and physics particle

between the behaviour of random variable and physics particle. So, lets move to physics.

The core of the Hamiltonian physics is Hamiltonian (something like total energy of the system):

$$H(p,q) = K(p) + U(q) \tag{16}$$

where q, p - coordinate (it is our random variable) and momentum, $K(p)$ and $U(q)$ - kinetic and potential energy. As we said, the potential energy is correspond to minus probability density function $U(q) = -\log P(q) = -\log \frac{\hat{P}(q)}{Z}$. The standard form of the kinetic energy is $K(p) = \sum_i \frac{p_i^2}{2m_i}$. As was said, we are working with the energy based distribution (we want to sample from it):

$$\begin{gathered}
P(q,p) = \frac{e^{-H(p,q)}}{Z} = \frac{1}{Z} e^{-\sum_i \frac{p_i^2}{2m_i} + \log \frac{\hat{P}(q)}{Z_1}} = \frac{1}{Z_1} e^{-\sum_i \frac{p_i^2}{2m_i}} \frac{1}{Z_2} e^{+\log \hat{P}(q)} = P(q)P(p) \\
P(q) = \frac{1}{Z_2} e^{\log \hat{P}(q)} = \frac{1}{Z_2} \hat{P}(q) \\
P(p) = \frac{1}{Z_1} e^{-\sum_i \frac{p_i^2}{2m_i}} = \prod_i \mathbb{N}(p_i|0, m_i) - \text{fully factorized normal distribution}
\end{gathered} \tag{17}$$

In terms of statistical physics the distribution $P(q,p)$ define the canonical ensemble. This means that system of particles is in interaction with heat bath (environment). The interaction is manifest via changing the energy between particles and heat bath. So, we have a two types of interactions: internal (particles interact with each other and thus create potential field) and external (particles interact with external environment - heat bath). Worth noting that the first interactions do not change the total energy, i.e. $H(p,q)$ remains constant while particle moves in potential field. However, the external interactions changes the total energy. Because of this interactions the system of particles can be in different energy states. More precisely, the heat bath change the momentum of particle by giving it acceleration (not sure). Lets connect physics with the previous notations

$$\hat{H}(X,p) = -\log \hat{P}(X,p) = \sum_i \frac{p_i^2}{2m_i} - \log \hat{P}_\theta(X) \tag{18}$$

where X is our target random variable, $\hat{P}_\theta(X) = e^{-E_\theta(X)}$ - neural network approximating the unnormalized density. Our main goal is to make samples of X random variable. To do this we have to model two mentioned interactions. To model external interactions we should create sample from $p \sim P(p)$. The modelling of the internal interactions more harder. We have to use Hamiltonian equations:

$$\begin{gathered}
\frac{\partial X_k}{\partial t} = \frac{\partial H}{\partial p_k} = \frac{p_i}{m_i} \\
\frac{\partial p_k}{\partial t} = -\frac{\partial H}{\partial X_k} = \left(\nabla_X \log \hat{P}_\theta(X)\right)_k
\end{gathered} \tag{19}$$

Unfortunately, these equations cannot be solve analytical for every cases. So, we have to use numerical methods, the most common is Leapfrog step:

$$\begin{aligned} p_k^{(t+\frac{\epsilon}{2})} &= p_k^{(t)} + \frac{\epsilon}{2}\left(\nabla_X \log \hat{P}_\theta(X^{(t)})\right)_k \\ X_k^{(t+\epsilon)} &= X_k^{(t)} + \epsilon \frac{p_k^{(t+\frac{\epsilon}{2})}}{m_k} \\ p_k^{(t+\epsilon)} &= p_k^{(t+\frac{\epsilon}{2})} + \frac{\epsilon}{2}\left(\nabla_X \log \hat{P}_\theta(X^{(t+\epsilon)})\right)_k \end{aligned} \tag{20}$$

This method is preferable because it saves the useful properties of the Hamilton equations (reversibility, volume preservation). The Hamiltonian MCMC has the following form

1. Generate momentum $p^{(t)} \sim \mathbb{N}(p|0, M)$
2. Generate proposal points p', X' solving Hamilton equations using Leapfrog step
3. Use Metropolis step:
   - Calculate $\alpha = min\left(1, \frac{P(X',p')}{P(X^{(t)},p^{(t)})}\right) = min\left(1, \frac{\hat{P}(X',p')}{\hat{P}(X^{(t)},p^{(t)})}\right)$ $= min\left(1, \exp\left[-\hat{H}(X',p') + \hat{H}(X^{(t)},p^{(t)})\right]\right)$
   - $X^{(t+1)}, p^{(t+1)} = X', p'$ with probability α else $X^{(t+1)}, p^{(t+1)} = X^{(t)}, p^{(t)}$

The first step model external interactions, the second step model internal interactions. The third step is necessary due to numerical integration (Leapfrog step) and called Metropolis correction. It is possible to say that Hamiltonian Monte Carlo (HMC) improves the computational efficiency of the Metropolis-Hastings algorithm by reducing its random walk behaviour. Lets consider applied example with Rosenbrock function 3 As can be seen, Hamiltonian Monte-Carlo warm up quite faster (only one step) than

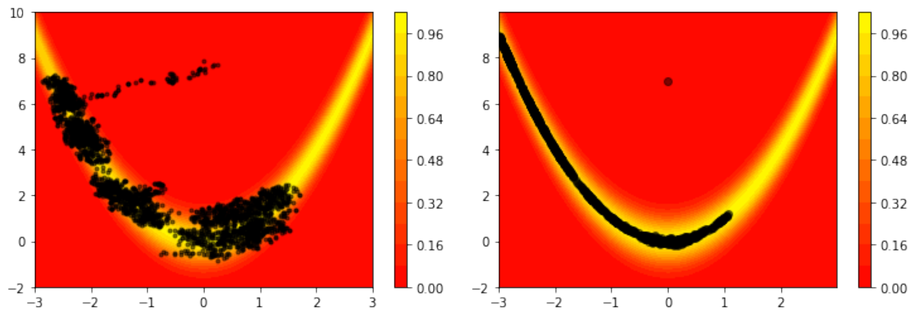

Figure 3: Metropolis-Hastings for Rosenbrock function (left). Hamiltonian Monte Carlo for Rosenbrock function (right).

Metropolis-Hastings algorithm and gives more accurate samples.

2.1.4 Langevin dynamics

Now we again start with physics. Lets write Langevin equation and describe it

$$m\mathbf{a} = \mathbf{F}(x) - \gamma\mathbf{v} + \eta(t) \tag{21}$$

This equation describes the Brownian motion of the particle, i.e. changing of acceleration of the particle with mass m in time. This motion defines by viscous friction, $\gamma\mathbf{v}$, stochastic term, $\eta(t)$, due to continuous collisions of a particle with liquid molecules and systematic force, $\mathbf{F}(x)$, arising from intramolecular and intermolecular interactions. Let the mass be negligible and use the fact that $\mathbf{v} = \frac{\partial \mathbf{x}}{\partial t}$

$$0 = \mathbf{F}(x)dt - \gamma d\mathbf{x} + \eta(t)dt \tag{22}$$

Also, we can write that force is a gradient of potential energy with minus:

$$\begin{gathered}\mathbf{F}(x) = -\nabla U(x) \\ d\mathbf{x} = -\nabla U(x)dt + \eta(t)dt\end{gathered} \tag{23}$$

We obtain the equation that gives us the position of the particle that moves in potential field U and randomly interacts with other particles (it can be collisions). Is this equation useful for sampling from some distributions? Yes! As was said before, we can match random variable (tends to realize in mode of distribution) and particle (tends to be in minimum of potential field). For these equation it is also true: x is our random variable and $U(x) = -\log P(x)$, $P-$ target distribution. Stochastic term is also useful, because without it we would obtain samples only from mode of distribution P, but we want diverse samples.

So, the question may arise: if we will move random variable $\mathbf{x}$ corresponding to Langevin equation will we obtain samples from target distribution $P(\mathbf{x})$? In other words, is stationary distribution of Markov Chain generated by Langevin equation will equal to $P(\mathbf{x})$? To answer this question we have to derive Fokker-Planck equation. Lets discretize Langevin equation:

$$\begin{gathered}\mathbf{F}(\mathbf{x}) = -\nabla U(\mathbf{x}) \\ \mathbf{x}_{k+1} = \mathbf{x}_k - \nabla U(\mathbf{x}_k)dt + d\eta(t) \\ d\eta(t) \sim \mathbb{N}(0, dt)\end{gathered} \tag{24}$$

So, we can write that

$$\mathbf{x} \sim \mathbb{N}\left(\mathbf{x}|\mathbf{x}' - \nabla U(\mathbf{x}')dt, dt\right) \tag{25}$$

Standard formula for changing the density of a random variable in Markov Chain:

$$\begin{gathered}P_t(\mathbf{x}) = \int d\mathbf{x}' P(\mathbf{x}|\mathbf{x}')P_{t-dt}(\mathbf{x}') \\ P_{t'}(\mathbf{x}) = \int d\mathbf{x}' P(\mathbf{x}|\mathbf{x}')P_{t'}(\mathbf{x}'), \text{ then } P_{t'} - \text{stationary distribution}\end{gathered} \tag{26}$$

where $P(\mathbf{x}|\mathbf{x}')-$ transition distribution, $P_t(\mathbf{x})-$ distribution of random variable $\mathbf{x}$ that depends on time. Our main task is to check will there be a stationary distribution equals to our target distribution $P(\mathbf{x})$. Based on previous formula we can write $P(\mathbf{x}|\mathbf{x}') = \mathbb{N}\left(\mathbf{x}|\mathbf{x}' - \nabla U(\mathbf{x}')dt, dt\right)$. However we have a problem: to calculate integral in 26 we need samples from $P_{t-dt}(\mathbf{x}')$, however $P_{t-dt}(\mathbf{x}')$ might be complex. Would be better to sample from $P(\mathbf{x}|\mathbf{x}')$ because it much easier (normal distribution). To do this we have to change variables in the integral.

$$\begin{gathered}\mathbb{N}\left(\mathbf{x}|\mathbf{x}' - \nabla U(\mathbf{x}')dt, dt\right) = \frac{1}{(2\pi dt)^{\frac{n}{2}}} \exp\left(\frac{-(-\mathbf{x} + \mathbf{x}' - \nabla U(\mathbf{x}')dt)^2}{2dt}\right) = \\ = [\mathbf{y} = -\mathbf{x} + \mathbf{x}' - \nabla U(\mathbf{x}')dt] = \mathbb{N}(\mathbf{y}|0, dt)\end{gathered} \tag{27}$$

$$P_t(\mathbf{x}) = \int d\mathbf{y} \left|\frac{\partial \mathbf{x}'}{\partial \mathbf{y}}\right| \mathbb{N}(\mathbf{y}|0, dt) P_{t-dt}(\mathbf{x}'(\mathbf{y})) \tag{28}$$

where $\left|\frac{\partial \mathbf{x}'}{\partial \mathbf{y}}\right|$ - determinant of Jacobian. Here we have to calculate $\mathbf{x}'(\mathbf{y})$ and determinant.

$$\begin{aligned}
\mathbf{y} = \mathbf{x}' - \mathbf{x} - \nabla U(\mathbf{x}')dt \approx \mathbf{x}' - \mathbf{x} - \left(\nabla U(\mathbf{x}) + \frac{\partial \nabla U(\mathbf{x})}{\partial x}(\mathbf{x}' - \mathbf{x}) + O(\mathbf{x}' - \mathbf{x})^2\right) dt \\
\mathbf{x}' = \left(I - \frac{\partial \nabla U(\mathbf{x})}{\partial x} dt\right)^{-1} \left(\mathbf{y} + \mathbf{x} + \nabla U(\mathbf{x})dt - \frac{\partial \nabla U(\mathbf{x})}{\partial x}\mathbf{x}dt + O(dt)\right) \approx \\
\approx \left(I + \frac{\partial \nabla U(\mathbf{x})}{\partial x} dt + O(dt)\right) \left(\mathbf{y} + \mathbf{x} + \nabla U(\mathbf{x})dt - \frac{\partial \nabla U(\mathbf{x})}{\partial x}\mathbf{x}dt + O(dt)\right) = \\
= \mathbf{x} + \mathbf{y} + \nabla U(\mathbf{x})dt - \frac{\partial \nabla U(\mathbf{x})}{\partial x}\mathbf{x}dt + \frac{\partial \nabla U(\mathbf{x})}{\partial x}\mathbf{y}dt + \frac{\partial \nabla U(\mathbf{x})}{\partial x}\mathbf{x}dt + O(dt) = \\
= \mathbf{x} + \mathbf{y} + \nabla U(\mathbf{x})dt + \frac{\partial \nabla U(\mathbf{x})}{\partial x}\mathbf{y}dt + O(dt)
\end{aligned} \tag{29}$$

We know that

$$\begin{aligned}
\mathbf{y}dt = (-\mathbf{x} + \mathbf{x}' - \nabla U(\mathbf{x}')dt)dt \\
\mathbf{x} - \mathbf{x}' = -dt\nabla U(\mathbf{x}') + \mathbb{N}(0, dt)
\end{aligned} \tag{30}$$

Then

$$\mathbf{y}dt = \mathbb{N}(0, dt)dt = \mathbb{N}(0, I)(dt)^{\frac{3}{2}} = O(dt) \tag{31}$$

Finally, we have

$$\begin{aligned}
\mathbf{x}' = \mathbf{x} + \mathbf{y} + \nabla U(\mathbf{x})dt + O(dt) \\
\left|\frac{\partial \mathbf{x}'}{\partial \mathbf{y}}\right| = 1 + \mathbf{div}\nabla U(\mathbf{x})dt + O(dt) \\
\mathbf{div} f(x) = \sum_i \frac{\partial f_i}{\partial x_i}
\end{aligned} \tag{32}$$

Then

$$\begin{aligned}
P_t(\mathbf{x}) = \int d\mathbf{y}\mathbb{N}(\mathbf{y}|0, dt) P_{t-dt}(\mathbf{x} + \mathbf{y} + \nabla U(\mathbf{x})dt) = \\
= (1 + \mathbf{div}\nabla U(\mathbf{x})dt)\, \mathbb{E}_{\mathbb{N}(\mathbf{y}|0,dt)} \left[(P_{t-dt}(\mathbf{x} + \mathbf{y} + \nabla U(\mathbf{x})dt)\right]
\end{aligned} \tag{33}$$

Finally, we obtain Fokker-Planck equation

$$\frac{\partial}{\partial t} P_t(\mathbf{x}) = \nabla_\mathbf{x} P_t(\mathbf{x})^T \nabla U(\mathbf{x}) + P_t(\mathbf{x})\mathbf{div}\nabla U(\mathbf{x}) + \frac{1}{2}\nabla^2 P_t(\mathbf{x}) \tag{34}$$

This equation shows how distribution $P_t(\mathbf{x})$ of random variable $\mathbf{x}$ changing over time. So, we want to find stationary distribution, that is, that do not depend on time.

$$0 = \nabla_\mathbf{x} P_t(\mathbf{x})^T \nabla U(\mathbf{x}) + P_t(\mathbf{x})\mathbf{div}\nabla U(\mathbf{x}) + \frac{1}{2}\nabla^2 P_t(\mathbf{x}) \tag{35}$$

It is possible to show that Boltzmann distribution $P_{st}(\mathbf{x}) = \frac{1}{Z}\exp(-2U(\mathbf{x}))$ satisfies this equation. It is easy show this in one-dimension

$$0 = \frac{\partial P}{\partial x}\frac{\partial U}{\partial x} + P\frac{\partial^2 U}{\partial x^2} + \frac{1}{2}\frac{\partial^2 P}{\partial x^2} \tag{36}$$

Lets summarize what we have done. If we will transform random variable $\mathbf{x}$ according to Langevin equation:

$$\begin{gathered}\mathbf{x}_{k+1} = \mathbf{x}_k - \nabla U(\mathbf{x}_k)\epsilon^2 + \mu\epsilon \\ \mu \sim \mathbb{N}(0,1)\end{gathered} \tag{37}$$

Then, after some large step ($n > k$) we will obtain samples from stationary Boltzmann distribution, $P_{st}(\mathbf{x}) = \frac{1}{Z}\exp\left(-2U(\mathbf{x})\right)$. However we need samples from target distribution. We remember that we can match potential energy and target distribution like $U(\mathbf{x}) = -logP(\mathbf{x})$. Then

$$\begin{gathered}P_{st}(\mathbf{x}) = \frac{1}{Z}\exp\left(2logP(\mathbf{x})\right) = \frac{1}{Z}\exp(2)P(\mathbf{x}) = P(\mathbf{x}) \\ Z = \int d\mathbf{x}\exp(2)\exp(\log P(\mathbf{x})) = \exp(2)\end{gathered} \tag{38}$$

Thats great, stationary distribution equals to our target.

3 Score Matching

Finally, we have sorted out the MLT with MCMC. But such approach is extremely computational expensive, because we need to run Markov Chain at each step of training. We want to avoid this.

We know that maximization of the log-likelihood equivalent to minimization of the KL-divergence

$$\max_\theta \mathbb{E}_{P_{data}(X)} \log P_\theta(X) \Longleftrightarrow \min_\theta \mathbb{KL}(P_{data}||P_\theta) \tag{39}$$

But lets consider another divergence called Fisher divergence.

$$\begin{aligned}\mathbb{F}(P_{data}||P_\theta) &= \frac{1}{2}\int P_{data}(X)\|\nabla_X \log P_{data}(X) - \nabla_X \log P_\theta(X)\|^2 \\ &= \frac{1}{2}\mathbb{E}_{P_{data}(X)}\|\nabla_X \log P_{data}(X) - \nabla_X \log P_\theta(X)\|^2\end{aligned} \tag{40}$$

Here we use gradient of distributions with respect to data, it allows to avoid normalizing constant, $\nabla_X \log P_\theta(X) = -\nabla_X E_\theta(X)$. That‘s nice, but two questions may arise: 1) how to deal with $\nabla_X \log P_{data}(X)$ and 2) why this divergence gives $P_{data} = P_{\theta^*}$. Lets start with the latter. We can see that

$$\begin{gathered}\mathbb{F}(P_{data}||P_\theta) = 0 \Longleftrightarrow \nabla_X \log P_\theta(X) = \nabla_X \log P_{data}(X) \\ \to \log P_\theta(X) = \log P_{data}(X) + C \\ \exp(\log P_\theta(X)) = \exp(\log P_{data}(X) + C) = C' \exp(\log P_{data}(X)) \\ \int dX P_\theta(X) = 1 = C' \int dX P_{data}(X) = C'\end{gathered} \tag{41}$$

So, we show that, when F-divergence equals to zero, $P_\theta = P_{data}$. Lets deal with the first question. To do this consider one dimension case:

$$\begin{aligned} &\int dx P_{data}(x)\left(\frac{\partial}{\partial x}\log P_{data}(x)-\frac{\partial}{\partial x}\log P_\theta(x)\right)^2 \\ =&\int dx P_{data}(x)\left(\frac{\partial}{\partial x}\log P_{data}(x)\right)^2-2\frac{\partial}{\partial x}\log P_{data}(x)\frac{\partial}{\partial x}\log P_\theta(x)+\left(\frac{\partial}{\partial x}\log P_\theta(x)\right)^2 \\ =&\,C-2\int dx P_{data}(x)\frac{\partial}{\partial x}\log P_{data}(x)\frac{\partial}{\partial x}\log P_\theta(x)+\int dx P_{data}(x)\left(\frac{\partial}{\partial x}\log P_\theta(x)\right)^2 \end{aligned} \tag{42}$$

Here we have a constant C with respect to θ. The third is computable, we need to consider only the second one

$$\begin{aligned} &\int dx P_{data}(x)\frac{\partial}{\partial x}\log P_{data}(x)\frac{\partial}{\partial x}\log P_\theta(x) \\ =&\int dx\frac{\partial P_{data}(x)}{\partial x}\frac{\partial}{\partial x}\log P_\theta(x)=\int dP_{data}(x)\frac{\partial}{\partial x}\log P_\theta(x) \\ =&\,P_{data}(x)\frac{\partial}{\partial x}\log P_\theta(x)|_{-\infty}^{+\infty}-\int dx P_{data}(x)\frac{\partial^2}{\partial x^2}\log P_\theta(x) \\ &P_{data}(x)\to_{x\to\infty}0, \end{aligned} \tag{43}$$

So, we obtain that

$$\begin{aligned} \mathbb{F}(P_{data}||P_\theta)&=\frac{1}{2}\,\mathbb{E}_{P_{data}(X)}\left\|\nabla_X\log P_{data}(X)-\nabla_X\log P_\theta(X)\right\|^2 \\ &=\mathbb{E}_{P_{data}(X)}\left[\sum_{j=1}^{n}\left(\frac{1}{2}\left(\frac{\partial}{\partial x_i}\log P_\theta(X)\right)^2+\frac{\partial^2}{\partial x_i^2}\log P_\theta(X)\right)\right]+const \\ &=\mathbb{E}_{P_{data}(X)}\left[\frac{1}{2}\|\nabla_X\log P_\theta(X)\|^2+\mathbf{tr}\left(\mathbf{H}_{\log P_\theta(X)}\right)\right]+const \end{aligned} \tag{44}$$

Here $\mathbf{tr}\left(\mathbf{H}_{\log P_\theta(X)}\right)$ is a trace of the Hessian matrix:

$$\mathbf{tr}\begin{pmatrix} \frac{\partial^2}{\partial x_1^2}\log P_\theta(X) & ... & \frac{\partial^2}{\partial x_n\partial x_1}\log P_\theta(X) \\ ... & ... & ... \\ \frac{\partial^2}{\partial x_1\partial x_n}\log P_\theta(X) & ... & \frac{\partial^2}{\partial x_n^2}\log P_\theta(X) \end{pmatrix} \tag{45}$$

Both terms in the final formula are calculable. It is necessary to highlight following significant facts:

- Based on the Fisher-divergence, we require that $\nabla_X \log P_{data}(X)$ exists. However, these conditions may not always hold in practice. For example, a distribution of digital images is typically discrete and therefore Score Matching is not directly applicable.

- Calculation of the Hessian has $O(n^2)$ complexity. However, we need only trace of it, it is much easier and requires $O(n)$ operations. Lets show that in autograd framework we cannot obtain trace directly (we need to calculate Hessian first):

  First of all we need to do forward pass Figure 4. After this to obtain gradients

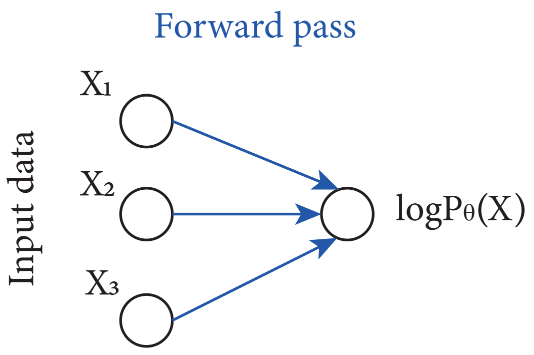

Figure 4: Forward pass

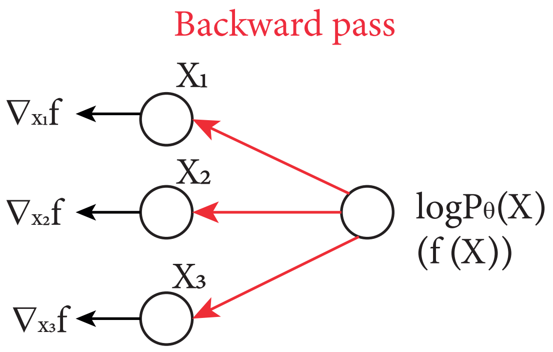

Figure 5: Backward pass

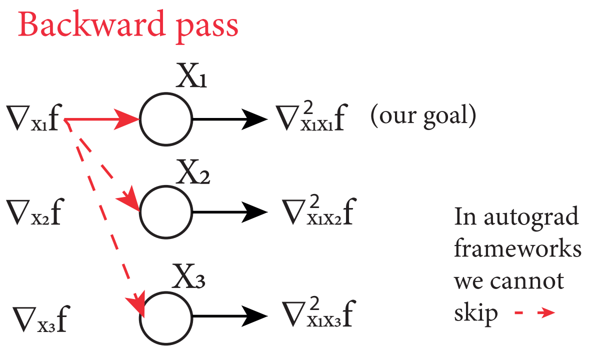

Figure 6: Backward pass to obtain second derivativies

we have to do backward pass Figure 5. And finally, most interesting is another backward Figure 6. For each component of the gradient (in figure it is shown like $\nabla_{x_i} f$) we need to calculate backward pass with respect to input. In such a way we will obtain a vector for each component of the gradient (in figure it is shown as $\frac{\partial^2 f}{\partial x_1}, \frac{\partial^2 f}{\partial x_1 \partial x_2}, \frac{\partial^2 f}{\partial x_1 \partial x_3}$). For calculation of the trace we only need $\frac{\partial^2 f}{\partial x_1}$, but autograd frameworks do not allow to obtain it directly, first of all we need to calculate gradient of $\nabla_{x_i} f$ with respect to input. This limitation arise from the fact that when we calculate our forward pass we create a computational graph with all variables, i.e. with X_1, X_2, X_3. And when we do second backward we cannot drop part of variables (X_2, X_3 in case of Figure 6) from the graph. I think it is possible to solve this problem if we consider own graph for each variable, but it is very tricky, especially for high dimension data :).

So, we have skipped intractable normalizing constant in model distribution, but obtained solution still has mentioned limitations. We need to solve them.

3.1 Denoising Score Matching

[LCS19] One way is to consider noised data, i.e. $X^* = X + \epsilon, \epsilon \sim \mathbb{N}(\epsilon|0, \sigma^2)$. In such case distribution of X^* is smooth, $Q(X^*) = \int dX Q(X^*|X) P_{data}(X)$, where $Q(X^*|X) = \mathbb{N}(X^*|X, \sigma^2)$ - transition between original object X and noised object X^*. So now we can approximate noised distribution via Fisher divergence

$$\mathbb{F}(Q||P_\theta) = \frac{1}{2}\mathbb{E}_{Q(X^*)} \|\nabla_{X^*} \log Q(X^*) - \nabla_{X^*} \log P_\theta(X^*)\|^2 \tag{46}$$

But it is not clear how to calculate $\nabla_{X^*} \log Q(X^*)$. Lets consider how to deal with it

$$\begin{gathered}
\frac{1}{2}\mathbb{E}_{Q(X^*)} \|\nabla_{X^*} \log Q(X^*) - \nabla_{X^*} \log P_\theta(X^*)\|^2 \\
= \frac{1}{2}\mathbb{E}_{Q(X^*)} \|\nabla_{X^*} \log Q(X^*)\|^2 - \mathbb{E}_{Q(X^*)} \nabla_{X^*} \log Q(X^*) \cdot \nabla_{X^*} \log P_\theta(X^*) \\
+ \frac{1}{2}\mathbb{E}_{Q(X^*)} \|\nabla_{X^*} \log P_\theta(X^*)\|^2, \text{sign } \cdot \text{ denotes scalar product}
\end{gathered}$$

$$\begin{gathered}
\mathbb{E}_{Q(X^*)} \nabla_{X^*} \log Q(X^*) \cdot \nabla_{X^*} \log P_\theta(X^*) = \int dX^* Q(X^*) \nabla_{X^*} \log Q(X^*) \cdot \nabla_{X^*} \log P_\theta(X^*) \\
= \int dX^* \nabla_{X^*} Q(X^*) \cdot \nabla_{X^*} \log P_\theta(X^*) = \int dX^* \nabla_{X^*} \int dX Q(X^*|X) P_{data}(X) \cdot \nabla_{X^*} \log P_\theta(X^*) \\
= \int\int dX^* dX P_{data}(X) Q(X^*|X) \nabla_{X^*} \log Q(X^*|X) \cdot \nabla_{X^*} \log P_\theta(X^*) \\
= \int\int dX^* dX Q(X^*, X) \nabla_{X^*} \log Q(X^*|X) \cdot \nabla_{X^*} \log P_\theta(X^*) \\
= \mathbb{E}_{Q(X^*, X)} \nabla_{X^*} \log Q(X^*|X) \cdot \nabla_{X^*} \log P_\theta(X^*)
\end{gathered} \tag{47}$$

So, we have that

$$\begin{gathered}
\frac{1}{2}\mathbb{E}_{Q(X^*)} \|\nabla_{X^*} \log Q(X^*) - \nabla_{X^*} \log P_\theta(X^*)\|^2 \\
= C - \mathbb{E}_{Q(X^*, X)} \nabla_{X^*} \log Q(X^*|X) \cdot \nabla_{X^*} \log P_\theta(X^*) + \frac{1}{2}\mathbb{E}_{Q(X^*)} \|\nabla_{X^*} \log P_\theta(X^*)\|^2 \\
= \frac{1}{2}\mathbb{E}_{Q(X, X^*)} \|\nabla_{X^*} \log Q(X^*|X) - \nabla_{X^*} \log P_\theta(X^*)\|^2 + C'
\end{gathered} \tag{48}$$

Finally, we have come to the fact that we need to optimize

$$\frac{1}{2}\mathbb{E}_{Q(X, X^*)} \|\nabla_{X^*} \log Q(X^*|X) - \nabla_{X^*} \log P_\theta(X^*)\|^2 \tag{49}$$

It is called **Denoising Score Matching**. As can be seen both terms in expectation can be calculated. So we solved both problem, because we do not need to calculate second derivatives and $\log Q(X^*|X)$ is smooth. In practice we will write this divergence like

$$\begin{gathered}\frac{1}{2}\,\mathbb{E}_{Q(X,X^*)}\,\|\frac{1}{\sigma^2}(X-X^*)-\nabla_{X^*}\log P_\theta(X^*)\|^2\\ \nabla_{X^*}\log Q(X^*|X)=\frac{1}{\sigma^2}(X-X^*)\end{gathered}\tag{50}$$

We define that

$$\begin{gathered}X^*=X+\epsilon=X+\sigma z\\ \epsilon\sim\mathbb{N}(\epsilon|0,\sigma^2),\,z\sim\mathbb{N}(\epsilon|0,I)\end{gathered}\tag{51}$$

So, **practice formula** has the following form:

$$\begin{gathered}\frac{1}{2}\,\mathbb{E}_{P_{data}(X)}\,\mathbb{E}_{z\sim\mathbb{N}(0,I)}\,\|\frac{z}{\sigma}+\nabla_X\log P_\theta(X+\sigma z)\|^2\\ \approx\frac{1}{2N}\sum_{i=1}^{N}\|\frac{z_i}{\sigma}+\nabla_x\log P_\theta(x_i+\sigma z_i)\|^2\end{gathered}\tag{52}$$

We obtained cool results, however it has his own drawbacks. We approximate $Q(X^*)$ not P_{data}, so we need to use small magnitude of noise, $\sigma\approx 0$. However, in this case formula 52 has large variance and it can lead to computational optimization problems.

3.2 Sliced Score Matching

As was said Denoising Score Matching is a nice result however we are not approximate P_{data} directly. So, lets consider another approach called Sliced Score Matching. In the Fisher Divergence we matched two gradients: $\nabla_X\log P_\theta(X)$ and $\nabla_X\log P_{data}(X)$ minimizing Euclidian norm. What does it mean when two vectors equals to each other? Easy question: every coordinate of both vectors equals. But let consider this question from another point of view. If we will match not a vectors but their projections on all possible directions (units vectors) in the space (scalar product). So, in this case when two vectors equals to each other? Again easy question: when projection on every direction equals.

This is the main idea of the Sliced Score Matching: we will match projections of both vectors to an arbitrary direction.

$$\mathbb{F}_s(P_{data}||P_\theta)=\mathbb{E}_{P_{data}}\,\mathbb{E}_{P(\mathbf{v})}\left[\frac{1}{2}\left(\mathbf{v^T}\nabla_X\log P_{data}(X)-\mathbf{v^T}\nabla_X\log P_\theta(X)\right)\right]^2\tag{53}$$

Lets explain this formula. First of all we have distribution $P(\mathbf{v})$ of directions $\mathbf{v}$. So, this divergence does following procedure: sample random vector $\mathbf{v}\sim P(\mathbf{v})$, project both gradients on this vector via scalar product, calculate difference between projections and repeat for other projections. Note that we have to calculate difference for all projection, in other way we will obtain bad estimator.

We can show that in this case we again can skip gradient of data distribution

$$\begin{aligned}
&\mathbb{E}_{P_{data}(X)}\,\mathbb{E}_{P(\mathbf{v})}\left[\frac{1}{2}\left(\mathbf{v^T}\nabla_X \log P_{data}(X) - \mathbf{v^T}\nabla_X \log P_\theta(X)\right)\right]^2 \\
&= \mathbb{E}_{P(\mathbf{v})}\int dX\, P_{data}(X)\left[\frac{1}{2}\left(\mathbf{v^T}\nabla_X \log P_{data}(X) - \mathbf{v^T}\nabla_X \log P_\theta(X)\right)\right]^2 \\
&\left[\frac{1}{2}\left(\mathbf{v^T}\nabla_X \log P_{data}(X) - \mathbf{v^T}\nabla_X \log P_\theta(X)\right)\right]^2 \\
&= C - \mathbf{v^T}\nabla_X \log P_{data}(X)\mathbf{v}\nabla_X \log P_\theta(X) + \frac{1}{2}(\mathbf{v^T}\nabla_X \log P_\theta(X))^2 \\
&\int dX\, P_{data}(X)\mathbf{v^T}\nabla_X \log P_{data}(X)\mathbf{v}\nabla_X \log P_\theta(X) = \int d(\mathbf{v^T}P_{data}(X))\mathbf{v}\nabla_X \log P_\theta(X) \\
&= \mathbf{v^T}P_{data}(X)\mathbf{v}\nabla_X \log P_\theta(X)|_{-\infty}^{+\infty} - \int d(\mathbf{v^T}\nabla_X \log P_\theta(X))\mathbf{v}P_{data}(X) \\
&= -\int dX\, P_{data}(X)\mathbf{v^T}\nabla_X^2 \log P_\theta(X)\mathbf{v}
\end{aligned} \tag{54}$$

Here we have $\mathbf{v^T} \in \mathbb{R}^{1,n}$, $\nabla_X^2 \log P_\theta(X) \in \mathbb{R}^{n,n}$. Then

$$\mathbf{v^T}\nabla_X^2 \log P_\theta(X)\mathbf{v} = \sum_{i=1}^{n}\frac{\partial}{\partial x_i}\sum_{j=1}^{n}\left(\frac{\partial}{\partial x_j}\log P_\theta \mathbf{v}_j\right)\mathbf{v}_i \tag{55}$$

Finally,

$$\begin{aligned}
\mathbb{F}(P_{data}||P_\theta) &= \mathbb{E}_{P_{data}(X)}\,\mathbb{E}_{P(\mathbf{v})}\left[\frac{1}{2}(\mathbf{v^T}\nabla_X \log P_\theta(X))^2 - \mathbf{v^T}\nabla_X^2 \log P_\theta(X)\mathbf{v}\right] \\
&= \mathbb{E}_{P_{data}(X)}\,\mathbb{E}_{P(\mathbf{v})}\left[\frac{1}{2}\sum_{k=1}^{n}\left(\frac{\partial \log P_\theta(X)}{\partial x_k}\mathbf{v}_k\right)^2 - \sum_{i=1}^{n}\frac{\partial}{\partial x_i}\sum_{j=1}^{n}\left(\frac{\partial}{\partial x_j}\log P_\theta \mathbf{v}_j\right)\mathbf{v}_i\right]
\end{aligned} \tag{56}$$

As can be seen, we again have second order derivatives. But, lets looks on it more precisely. Is this derivatives better than second derivatives in the standard Score Matching? Yes! This is due to the fact that for calculation of $\sum_{i=1}^{n}\frac{\partial}{\partial x_i}\left[\sum_{j=1}^{n}\frac{\partial \log P_\theta(X)}{\partial x_j}\mathbf{v_j}\right]\mathbf{v}_i$ we have linear complexity. Lets show it in the same way as we did in the previous section (Figure 7): As it can be seen, we need to calculate gradient once, it requires $O(n)$ operations, then we need to calculate sum of this gradient, $O(n)$ complexity, and then we need again calculate gradient from this sum with respect to input X. Thus, our complexity is $O(n + n + n) = O(n)$. It is quite better than $O(n^2)$ that we had in standard Score Matching (in this case we had to calculate second gradients for every coordinate, and thus it required $O(n^2)$).

Lets consider expectations with respect to vector $\mathbf{v}$ more precisely. First of all, what distribution should we choose? We can select standard normal distribution and obtain.

$$\mathbb{F}(P_{data}||P_\theta) = \left[\frac{1}{2}\sum_{k=1}^{n}\left(\frac{\partial \log P_\theta(X)}{\partial x_k}\right)^2 - \sum_{i=1}^{n}\frac{\partial}{\partial x_i}\sum_{j=1}^{n}\left(\frac{\partial}{\partial x_j}\log P_\theta \mathbf{v}_j\right)\mathbf{v}_i\right] \tag{57}$$

So, we obtained some different approaches for training Energy based model via Score Matching. Lately, we will consider Score based generative models. This models

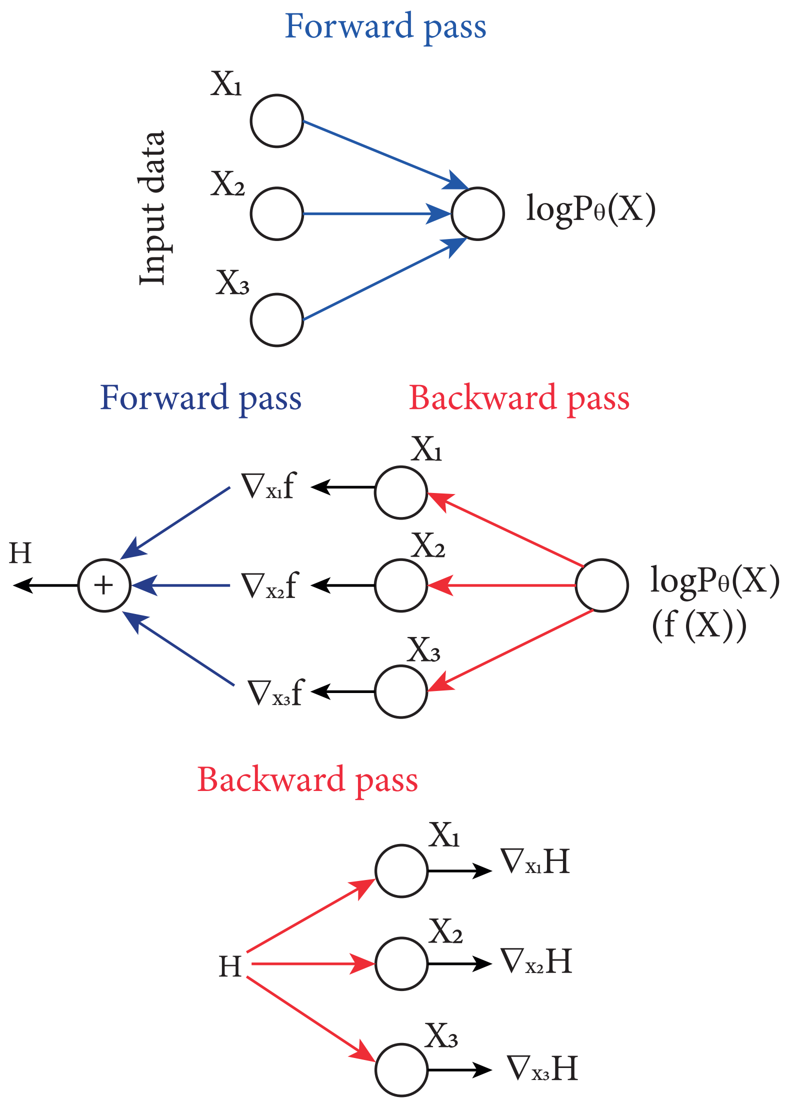

Figure 7: Visualization of the process of taking gradients in denoising score matching.

motivated by the fact that in proposed formulas (with score matching) we only need score, not energy. And at inference time we also need only score. So, we will obtain score by neural network.

References

[DM19] Yilun Du and Igor Mordatch. Implicit generation and modeling with energy based models. *Advances in Neural Information Processing Systems*, 32, 2019.

[LCS19] Zengyi Li, Yubei Chen, and Friedrich T Sommer. Learning energy-based models in high-dimensional spaces with multi-scale denoising score matching. *arXiv preprint arXiv:1910.07762*, 2019.

[SK21] Yang Song and Diederik P Kingma. How to train your energy-based models. *arXiv preprint arXiv:2101.03288*, 2021.