Statistical Methods

Samy Braik

We denote by p the target distribution and q an easy-to-sample distribution, for example a centered Gaussian. The goal is to learn the distribution p and sample from it.

Diffusion

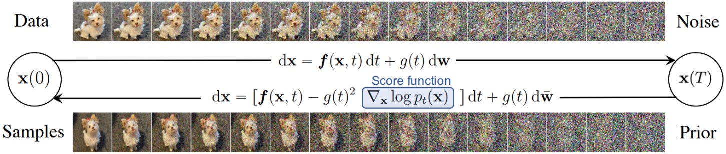
Let $X_0 \sim p$. We want to add noise until we reach pure noise, and denoise it afterward. We choose an horizon of time $T \in \mathbb{N}^*$ and a noise schedule $\beta : [0, T] \to \mathbb{R}^*$, continuous and non decreasing.

Forward process

$$d\overrightarrow{X}_t = \frac{-\beta(t)}{2\sigma^2}\overrightarrow{X}_t dt + \sqrt{\beta(t)} dB_t, \quad \overrightarrow{X}_0 \sim p \tag{1}$$

Backward process

$$\begin{aligned} d\overleftarrow{X}_t = &\left(\frac{\beta(T-t)}{2\sigma^2}\overleftarrow{X}_t + \beta(T-t)\nabla \log p_{T-t}\left(\overleftarrow{X}_t\right)\right) dt \\ &+ \sqrt{\beta(T-t)} dB_t, \quad \overleftarrow{X}_0 \sim p_T \end{aligned} \tag{2}$$



We learn the score by using score-matching techniques

Score matching

$$\mathcal{L}_{\text{score}}(\theta) = \mathbb{E}\left[\left\| s_\theta\left(\tau, \overrightarrow{X}_\tau\right) - \log p_\tau\left(\overrightarrow{X}_\tau | X_0\right)\right\|^2\right] \tag{3}$$

where s_θ is a neural network approximating the score function.

Plug it in the backward process and generate by discretizing the dynamics.

Normalizing flow

Let $x_0 \sim q$ and $f : \mathbb{R}^d \to \mathbb{R}^d$ an invertible and differentiable function an set $x_1 := f(x_0)$ such that $x_1 \sim p$.
We can write the density p as

$$p(x_1) = q(f^{-1}(x_1)) \left| \det \frac{\partial f^{-1}}{\partial x_1}(x_1) \right| \tag{4}$$

$$= q(f^{-1}(x_1)) \left| \det \frac{\partial f}{\partial x_0}(f^{-1}(x_1)) \right|^{-1} \tag{5}$$

We can then write the log-likelihood as

$$\log p(x_1) = \log p(f^{-1}(x_1)) - \log \left| \det \frac{\partial f}{\partial x_0}(f^{-1}(x_1)) \right| \tag{6}$$

Normalizing flow

Let $x_0 \sim q$ and $x_1 \sim p$. We want to learn f_θ such that $x_1 \simeq f_\theta(x_0) = z \sim p_Z$. To do that, we set a structure on f_θ, with $\phi_1, \ldots, \phi_k$ simpler function (all parametrized by θ) such that

$$f_\theta = \phi_k \circ \ldots \circ \phi_1$$

We determine f_θ by minimizing

$$\mathcal{L}_{\mathsf{NF}}(\theta) = \mathbb{E}\left[-\log p_Z(f_\theta(x)) - \log \left|\det \frac{\partial f_\theta}{\partial x}(x)\right|\right]$$

Normalizing flow

An early instances of normalizing flow is the planar flow

$$\phi_k(x) = x + \sigma(b_k^\intercal x + c)a_k$$

where $a_k, b_k \in \mathbb{R}^d, c \in \mathbb{R}$ and $\sigma : \mathbb{R} \to \mathbb{R}$ is a non-linear function.

Flow matching

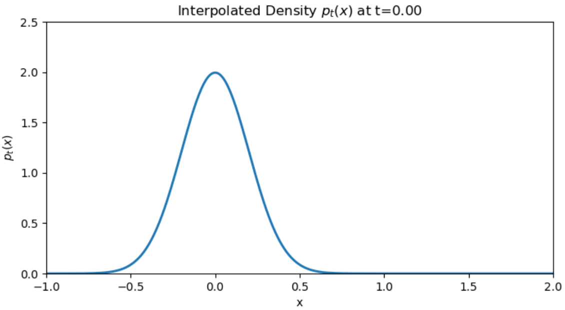
We start by defining a probability density path

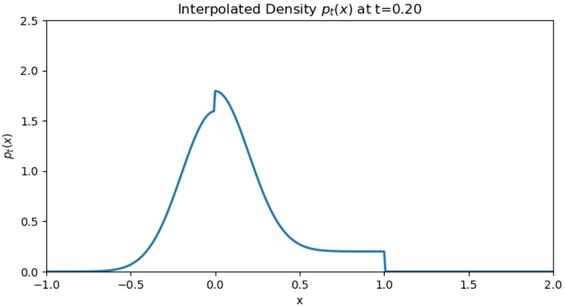
Probability density path

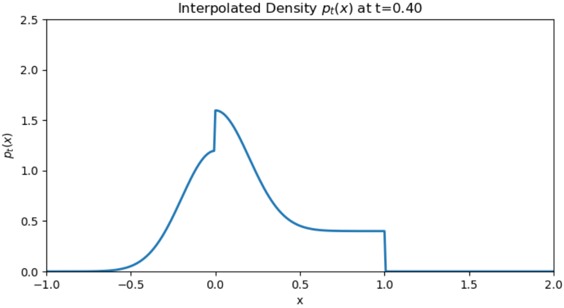
A probability path $p : [0,1] \times \mathbb{R}^d \to \mathbb{R}^d$ meaning that for each time t, p_t is density function i.e. $\int p_t(x)dx = 1$.

A simple example of such a path is a path p interpolating two density p_0 and p_1 with $p_t = tp_1 + (1-t)p_0$

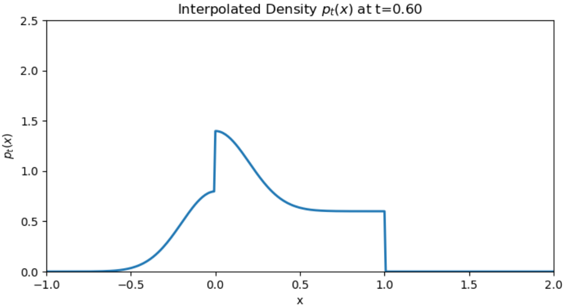
Figure: A probability path interpolating $\mathcal{N}(0, 0.2)$ and $\mathcal{U}([0,1])$


Interpolated Density $p_t(x)$ at t=0.00
2.5
2.0
1.5
1.0
0.5
0.0
$p_t(x)$
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
x

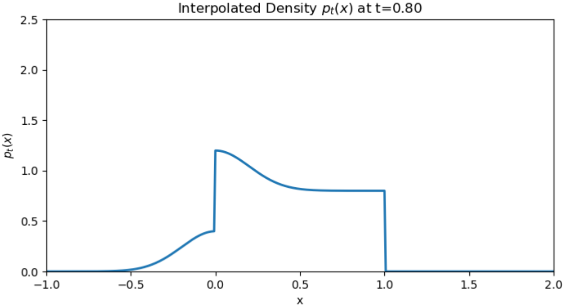

Interpolated Density $p_t(x)$ at t=0.20
2.5
2.0
1.5
1.0
0.5
0.0
$p_t(x)$
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
x

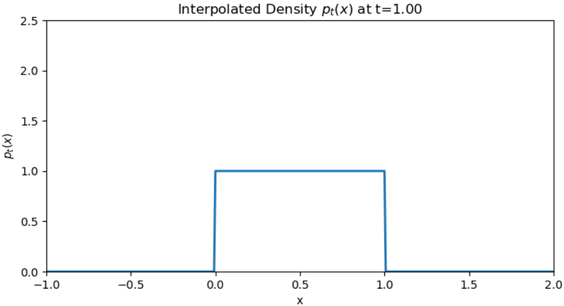


Interpolated Density $p_t(x)$ at t=0.40
2.5
2.0
1.5
1.0
0.5
0.0
$p_t(x)$
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
x



Interpolated Density $p_t(x)$ at t=0.60
2.5
2.0
1.5
1.0
0.5
0.0
$p_t(x)$
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
x


Interpolated Density $p_t(x)$ at t=0.80
$p_t(x)$
x
2.5
2.0
1.5
1.0
0.5
0.0
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0


Interpolated Density $p_t(x)$ at t=1.00
$p_t(x)$
x
2.5
2.0
1.5
1.0
0.5
0.0
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0

Flow matching

Next we introduce a core object, a time dependent vector field $v : [0, 1] \times \mathbb{R}^d \to \mathbb{R}^d$ which is used to construct a map $\phi : [0, 1] \times \mathbb{R}^d \to \mathbb{R}^d$, called a flow, by the following ODE

$$\begin{aligned} \frac{d}{dt}\phi_t(x) &= v_t(\phi_t(x)) \\ \phi_0(x) &= x \end{aligned} \tag{7}$$

The link between the flow and the probability path is given by the change of variable formula

$$p_t(x) = q(\phi_t^{-1}(x)) \det \left[\frac{\partial \phi_t^{-1}}{\partial x}(x) \right] \tag{8}$$

That coincides with the normalizing flow case.

The link between the vector field and the probability path is given by the continuity equation

$$\frac{d}{dt}p_t(x) + \mathsf{div}\left[v_t(x)p_t(x)\right] = 0 \tag{9}$$

It said that the vector field v_t generates the probability path p_t if it satisfies the continuity equation.

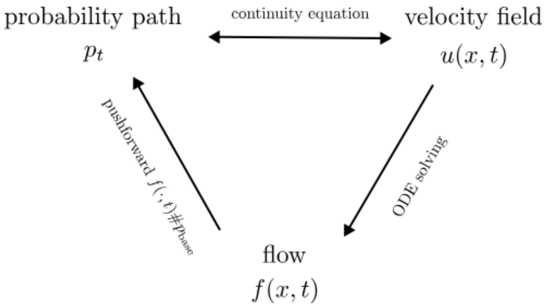


Figure: Link between marginal flow matching objects

Flow matching

Given a probability path p_t and a vector field v_t, the naïve flow matching loss is defined by

$$\mathcal{L}_{\mathsf{FM}}(\theta) = \mathbb{E}_{t,p_t(x)}\left[\|v_t^\theta(x) - v_t(x)\|^2\right] \tag{10}$$

but we don't have access to v_t and p_t. To adress this problem, we

introduce new objects.

Flow matching

Given a particular data sample x_1 from p, we introduce a conditional probability path $p_t(x|x_1)$ such that at time $t = 0$ $p_0(x|x_1) = q(x)$ and by marginalizing over x_1 we can recover the marginal probability path

$$p_t(x) = \int p_t(x|x_1)p(x_1)dx_1 \tag{11}$$

Instead of working with a probability density, we consider samples from our distributions and define how to go from one to the other.

Flow matching

In the same vein, we can define a conditional vector field, assuming $p_t(x) > 0$ for all t and x

$$v_t(x) = \int v_t(x|x_1) \frac{p_t(x|x_1)p(x_1)}{p_t(x)} dx_1 \tag{12}$$

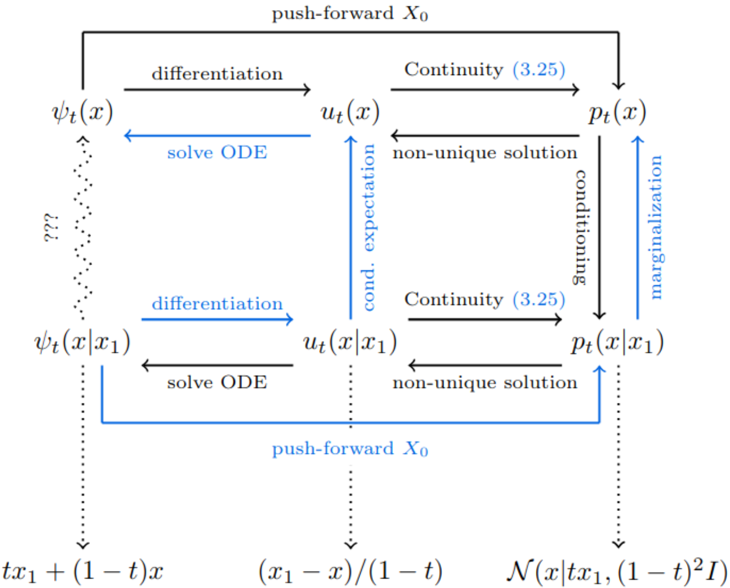


Figure: Link between all the flow matching objects

Flow matching

The new loss function, called conditional flow matching loss, is defined as

$$\mathcal{L}_{\mathsf{CFM}}(\theta) = \mathbb{E}_{t,p_t(x|x_1)} \left[\left\| v_t^{\theta}(x|x_1) - v_t(x|x_1) \right\|^2 \right] \tag{13}$$

with the property : $\mathcal{L}_{\mathsf{FM}}(\theta) = \mathcal{L}_{\mathsf{CFM}}(\theta)$ up to a constant independant of θ.

Flow matching

With in mind

$$v_t(x|x_1), p_t(x|x_1) \iff \phi_t(x|x_1)$$

Algorithm Flow matching training

Input: dataset p, noise q
Initialized v^θ
while not converged **do**
  $t \sim \mathcal{U}([0,1])$
  $x_1 \sim p(x_1)$
  $x_0 \sim q(x_0)$
  $x_t = \phi_t(x_0|x_1)$
  Gradient step with $\nabla_\theta \|v_t^\theta(x_t) - \dot{x}_t\|^2$
Output: v^θ

Flow matching

Algorithm Flow matching sampling

Input: Trained v^θ
$x_0 \sim q(x_0)$
Solve numerically the ODE $\dot{x}_t = v_t^\theta(x_t)$
Output: x_1

Choice of conditional flow

Linear flow

The simplest flow is a linear flow defined by

$$\phi_t(x|x_1) = \alpha_t x + \sigma_t x_1 \tag{14}$$

where α_t and σ_t are two differentiable functions and satisfy the constraint $\alpha_0 = \sigma_1 = 1$ and $\alpha_1 = \sigma_0 = 0$.

In particular, the simplest setting is to choose $\alpha_t = 1 - t$ and $\sigma_t = t$. This is called the **rectified flow**.

Link between diffusion and flow matching

Considering the forward SDE

$$\frac{dx_t}{dt} = f(x,t)dt + g(t)B_t$$

and the associated reverse SDE

$$\frac{dx_t}{dt} = \left(f(x,t) + \frac{1}{2}g(t)^2\nabla \log p_t(x)\right)dt + g(t)dB_t$$

which has the same marginals p_t as the probability flow ODE

$$\frac{dx_t}{dt} = f(x,t) - \frac{1}{2}g(t)^2\nabla_x \log p_t(x)$$

Algorithm Diffusion training

Input: dataset p, noise q

Initialized s^θ

while not converged **do**

$t \sim \mathcal{U}([0,1])$

$x_1 \sim p(x_1)$

$x_t = p_t(x_t|x_1)$

Gradient step with $\nabla_\theta \| s_t^\theta(x_t) - \nabla_{x_t} \log p_t(x_t|x_1)\|^2$

Output: s^θ

Algorithm Diffusion sampling

Input: Trained s^θ

$x_0 \sim p(x_0)$

Solve numerically the SDE or the probability flow ODE

Output: x_1

Link between normalizing flow and flow matching

When considering a transformation $\phi(x) = x + \frac{1}{K}v(x)$ in the normalizing flow context, we can re-arrange the equation to obtain

$$\frac{\phi(x) - x}{1/K} = v(x)$$

Taking the limit $K \to \infty$ gives us the ODE

$$\frac{dx_t}{d} = \lim_{K\to\infty} \frac{x_{t+1/K} - x_t}{1/K} = \frac{\phi_t(x_t) - x_t}{1/K} = v_t(x_t)$$

where the flow $\phi : [0,1] \times \mathbb{R}^d \to \mathbb{R}^d$ is defined by the ODE

$$\frac{d\phi_t}{d} = v_t(\phi_t(x_0))$$

Comparison

Models	Pros	Cons
Diffusion	Easy to train	Hard to sample (solve a SDE) Only works with Gaussian
Normalizing flow	Exact density estimation	Computationally intensive Less expressive
Flow matching	Exact density estimation Simulation free training Easy to sample	
Kernel estimator	Flexible Easy to exploit	Slow rate of convergence Hard to evaluate at new data point Hard to choose tuning parameters Need a lot of data