AN INTRODUCTION TO FLOW MATCHING

diffusion model generative modelling normalising flows

By Tor Fjelde (https://retiredparkingguard.com/about.html), Emile Mathieu (www.emilemathieu.fr), Vincent Dutordoir (https://vdutor.github.io/)

TABLE OF CONTENTS

INTRODUCTION

Flow matching (FM) is a recent generative modelling paradigm which has rapidly been gaining popularity in the deep probabilistic ML community. Flow matching combines aspects from *Continuous Normalising Flows (CNFs)* and *Diffusion Models (DMs)*, alleviating key issues both methods have. In this blogpost we'll cover the main ideas and unique properties of FM models starting from the basics.

GENERATIVE MODELLING

Let's assume we have data samples $\boldsymbol{x_1, x_2, \ldots, x_n}$ from a distribution of interest $\boldsymbol{q_1(x)}$, whose density is unknown. We're interested in using these samples to learn a probabilistic model approximating $\boldsymbol{q_1}$. In particular, we want efficient generation of new samples (approximately) distributed from $\boldsymbol{q_1}$. This task is referred to as **generative modelling**.

The advancement in generative modelling methods over the past decade has been nothing short of revolutionary. In 2012, Restricted Boltzmann Machines, then the leading generative model, were just about able to generate MNIST digits (https://physics.bu.edu/~pankajm/ML-Notebooks/HTML/NB17_CXVI_RBM_mnist.html). Today, state-of-the-art methods

are capable of generating high-quality images (https://openai.com/dall-e-3), audio (https://deepmind.google/discover/blog/transforming-the-future-of-music-creation/) and language (https://arxiv.org/pdf/2305.14671.pdf), as well as model complex biological (https://www.nature.com/articles/s41586-023-06415-8) and physical (https://deepmind.google/discover/blog/nowcasting-the-next-hour-of-rain/) systems. Unsurprisingly, these methods are now venturing into video generation (https://imagen.research.google/video/).

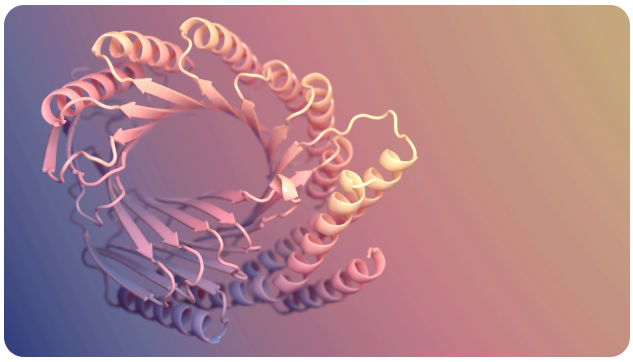

Figure 1: Protein generated by RFDiffusion (Watson et al., 2023).

Figure 2: Image from DALL-E 3 (Betker et al., 2023).

OUTLINE

Flow Matching (FM) models are in nature most closely related to (Continuous) Normalising Flows (CNFs). Therefore, we start this blogpost by briefly recapping the core concepts behind CNFs. We then continue by discussing the difficulties of CNFs and how FM models address them.

NORMALISING FLOWS

Let $\phi : \mathbb{R}^d \to \mathbb{R}^d$ be a continuously differentiable function which transforms elements of $\mathbb{R}^d$, with a continously differentiable inverse $\phi^{-1} : \mathbb{R}^d \to \mathbb{R}^d$. Let $q_0(x)$ be a density on $\mathbb{R}^d$ and let $p_1(\cdot)$ be the density induced by the following sampling procedure

$$x \sim q_0$$
$$y = \phi(x),$$

which corresponds to transforming the samples of q_0 by the mapping ϕ. Using the change-of-variable rule we can compute the density of p_1 as

$$p_1(y) = q_0(\phi^{-1}(y)) \left| \det \left[\frac{\partial \phi^{-1}}{\partial y}(y) \right] \right| \tag{1}$$

$$= \frac{q_0(x)}{\left| \det \left[\frac{\partial \phi}{\partial x}(x) \right] \right|} \quad \text{with } x = \phi^{-1}(y) \tag{2}$$

where the last equality can be seen from the fact that $\phi \circ \phi^{-1} = \mathrm{Id}$ and a simple application of the chain rule[1]. The quantity $\frac{\partial \phi^{-1}}{\partial y}$ is the Jacobian of the inverse map. It is a matrix of size $d \times d$ containing $J_{ij} = \frac{d\phi_i^{-1}}{dx_j}$. Depending on the task at hand, evaluation of likelihood or sampling, the formulation in (1) or (2) is preferred (Friedman, 1987; Chen & Gopinath, 2000).

Example: Transformation of 1D Gaussian variables by linear map ▶️

Transforming a base distribution q_0 into another p_1 via a transformation ϕ is interesting, yet its direct application in generative modelling is limited. In generative modelling, the aim is to approximate a distribution using only the available samples. Therefore, this task requires the transformation ϕ to map samples from a "simple" distribution, such as $\mathcal{N}(0, I)$, to approximately the data distribution. However, a straightforward linear transformation, as in the previous example, is inadequate due to the highly non-Gaussian nature of the data distribution. This brings us to a neural network as a flexible transformation ϕ_θ. The key task then becomes optimising the neural net's parameters θ.

LEARNING FLOW PARAMETERS BY MAXIMUM LIKELIHOOD

Let's denote the induced parametric density by the flow ϕ_θ as $p_1 \triangleq [\phi_\theta]_{\#} p_0$.

A natural optimisation objective for learning the parameters $\theta \in \Theta$ is to consider maximising the probability of the data under the model:

$$\text{argmax}_\theta \ \mathbb{E}_{x\sim\mathcal{D}}[\log p_1(x)].$$

Parameterising ϕ_θ as a deep neural network leads to several constraints:

- How do we enforce **invertibility**?
- How do we compute its **inverse**?
- How do we compute the **jacobian** efficiently?

Designing flows ϕ therefore requires trading-off expressivity (of the flow and thus of the probabilistic model) with the above mentioned considerations so that the flow can be trained efficiently.

RESIDUAL FLOW

In particular, computing the determinant of the Jacobian is in general very expensive (as it would require d automatic differentation passes in the flow) so we impose structure in ϕ[2].

Full-rank residual (Behrmann et al., 2019; Chen et al., 2010)

Expressive flows relying on a residual connection have been proposed as an interesting middle-ground between expressivity and efficient determinant estimation. They take the form:

$$\phi_k(x) = x + \delta\, u_k(x), \tag{8}$$

where unbiased estimate of the log likelihood can be obtained[3]. As opposed to auto-regressive flows (Huang et al., 2018, Larochelle and Murray, 2011, Papamakarios et al., 2017), and low-rank residual normalising flows (Van Den Berg et al. 2018), the update in (8) has *full rank* Jacobian, typically leading to more expressive transformations.

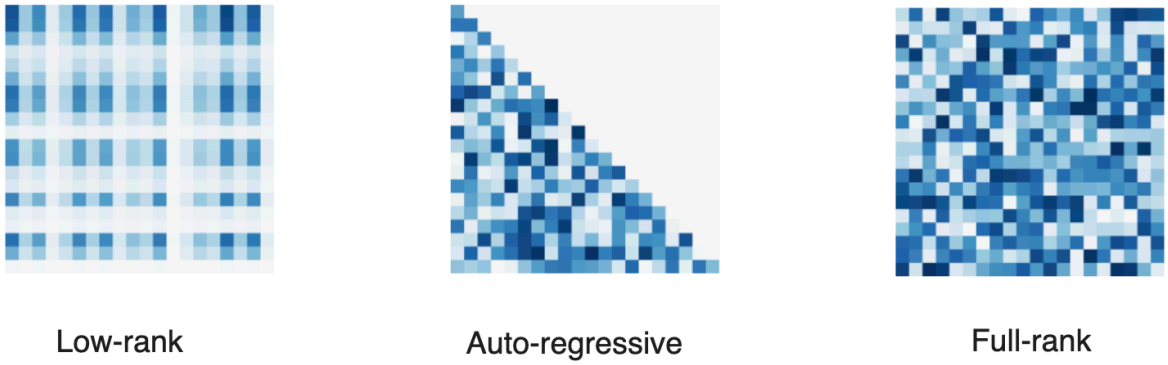

Figure 4: Jacobian structure of different normalising flows.

We can also compose such flows to get a new flow:

$$\phi = \phi_K \circ \ldots \circ \phi_2 \circ \phi_1.$$

This can be a useful way to construct move expressive flows! The model's log-likelihood is then given by summing each flow's contribution

$$\log q(y) = \log p(\phi^{-1}(y)) + \sum_{k=1}^{K} \log \det \left[\frac{\partial \phi_k^{-1}}{\partial x_{k+1}}(x_{k+1}) \right]$$

with $x_k = \phi_K^{-1} \circ \ldots \circ \phi_k^{-1}(y)$.

CONTINUOUS TIME LIMIT

As mentioned previously, residual flows are transformations of the form $\phi(x) = x + \delta\, u(x)$ for some $\delta > 0$ and Lipschitz residual connection u. We can re-arrange this to get

$$\frac{\phi(x) - x}{\delta} = u(x)$$

which is looking awfully similar to u being a derivative. In fact, letting $\delta = 1/K$ and taking the limit $K \to \infty$ under certain conditions[4], a composition of residual flows $\phi_K \circ \cdots \circ \phi_2 \circ \phi_1$ is given by an ordinary differential equation (ODE):

$$\frac{\mathrm{d}x_t}{\mathrm{d}t} = \lim_{\delta \to 0} \frac{x_{t+\delta} - x_t}{\delta} = \frac{\phi_t(x_t) - x_t}{\delta} = u_t(x_t)$$

where the *flow* of the ODE $\phi_t : [0,1] \times \mathbb{R}^d \to \mathbb{R}^d$ is defined such that

$$\frac{d\phi_t}{dt} = u_t(\phi_t(x_0)).$$

That is, ϕ_t maps initial condition x_0 to the ODE solution at time t:

$$x_t \triangleq \phi_t(x_0) = x_0 + \int_0^t u_s(x_s)\,\mathrm{d}s.$$

Continuous change-in-variables

Of course, this only defines the map $\phi_t(x)$; for this to be a useful normalising flow, we still need to compute the log-abs-determinant of the Jacobian!

As it turns out, the density induced by ϕ_t (or equivalently u_t) can be computed via the following equation[5]

$$\frac{\partial}{\partial_t} p_t(x_t) = -(\nabla \cdot (u_t p_t))(x_t).$$

This statement on the time-evolution of p_t is generally known as the *Transport Equation*. We refer to p_t as the probability path induced by u_t.

Computing the *total* derivative (as x_t also depends on t) in log-space yields[6]

$$\frac{\mathrm{d}}{\mathrm{d}t} \log p_t(x_t) = -(\nabla \cdot u_t)(x_t)$$

resulting in the log density

$$\log p_t(x) = \log p_0(x_0) - \int_0^t (\nabla \cdot u_s)(x_s)\,\mathrm{d}s.$$

Parameterising a vector field neural network $u_\theta : \mathbb{R}_+ \times \mathbb{R}^{\mathbb{d}} \to \mathbb{R}^{\mathbb{d}}$ therefore induces a parametric log-density

$$\log p_\theta(x) \triangleq \log p_1(x) = \log p_0(x_0) - \int_0^1 (\nabla \cdot u_\theta)(x_t)\mathrm{d}t. \tag{9}$$

In practice, to compute $\log p_t$ one can either solve both the time evolution of x_t and its log density $\log p_t$ jointly

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{pmatrix} x_t \\ \log p_t(x_t) \end{pmatrix} = \begin{pmatrix} u_\theta(t, x_t) \\ -\operatorname{div} u_\theta(t, x_t) \end{pmatrix},$$

or solve only for x_t and then use quadrature methods to estimate $\log p_t(x_t)$.

Feeding this (joint) vector field to an adaptive step-size ODE solver allows us to control both the error in the sample x_t and the error in the $\log p_t(x)$.

One may legitimately wonder why should we bother with such *time-continuous* flows versus *discrete* residual flows. There are a couple of benefits:

1. CNFs can be seen as an automatic way of choosing the number of residual flows K to use, which would otherwise be a hyperparameter we would have to tune. In the time-continuous setting, we can choose an error threshold ϵ and the adapative solver would give us a the discretisation step size δ, effectively yielding $K = 1/\delta$ steps. Using an explicit first order solver, each step is of the form $x \leftarrow x + \delta\, u_\theta(t_k, x)$, akin to a residual flow, where the residual connection parameters θ are *shared* for each discretisation step, since u_θ is amortised over t, instead of having a different θ_k for each layer.
2. In residual flows, during training we need to ensure that u_θ is $1/\delta$ Lipschitz; otherwise the resulting flow will not be invertible and thus not a valid normalising flow. With CNFs, we still require the vector field $u_\theta(t, x)$ to be Lipschitz in x, *but* we don't have to worry about exactly what this Lipschitz constant is, which is obviously much easier to satisfy and enforce in the neural architecture.

Now that you know why CNFs are cool, let's have a look at what such a flow would be for a simple example.

Example: Gaussian to a Gaussian (1D) ▶️

Training CNFs

Similarly to any flows, CNFs can be trained by maximum log-likelihood

$$\mathcal{L}(\theta) = \mathbb{E}_{x\sim q_1}[\log p_1(x)], \tag{10}$$

where the expectation is taken over the data distribution and p_1 is the parameteric distribution. This involves integrating the time-evolution of samples x_t and log-likelihood $\log p_t$, both terms being a function of the parametric vector field $u_\theta(t, x)$. This requires

- ⚠ Expensive numerical ODE simulations at training time!
- ⚠ Estimators for the divergence to scale nicely with high dimension. [7]

CNFs are very expressive as they parametrise a large class of flows, and therefore of probability distributions. Yet training can be *extremely* slow due to the ODE integration at each iteration. One may wonder whether a 'simulation-free', i.e. *not* requiring any integration, training procedure exists for training these CNFs.

FLOW MATCHING

And that is exactly where Flow Matching (FM) comes in!

Flow matching is a simulation-free way to train CNF models where we directly formulate a regression objective w.r.t. the parametric vector field u_θ of the form

$$\mathcal{L}(\theta) = \mathbb{E}_{t\sim\mathcal{U}[0,1]}\mathbb{E}_{x\sim p_t}\left[\|u_\theta(t,x) - u(t,x)\|^2\right].$$

In the equation above, $u(t, x)$ would be a vector field inducing a *probability path* (or bridge) p_t interpolating the reference p_0 to p_1, i.e.

$$\log p_1(x) = \log p_0 - \int_0^1 (\nabla \cdot u_t)(x_t)\,\mathrm{d}t.$$

In words: we're just performing regression on $u_t(x)$ for all $t \in [0, 1]$.

Of course, this requires knowledge of a *valid* $u(t, x)$, and if we already have access to u_t, there's no point in learning an approximation $u_\theta(t, x)$ in the first place! But as we will see in the next section, we can leverage this formulation to construct a useful

target for $u_\theta(t, x)$ witout having to compute explicitly $u(t, x)$.

This is where *Conditional* Flow Matching (CFM) comes to the rescue.

Non-uniqueness of vector field

We say *a valid* u_t because there is no *unique* vector field u_t; there are indeed many valid choices for u_t inducing maps $p_0 \overset{\phi}{\longleftrightarrow} p_1$ as illustrated in the figure below. As we will see in what follows, in practice we have to pick a particular target u_t, which has practical implications.

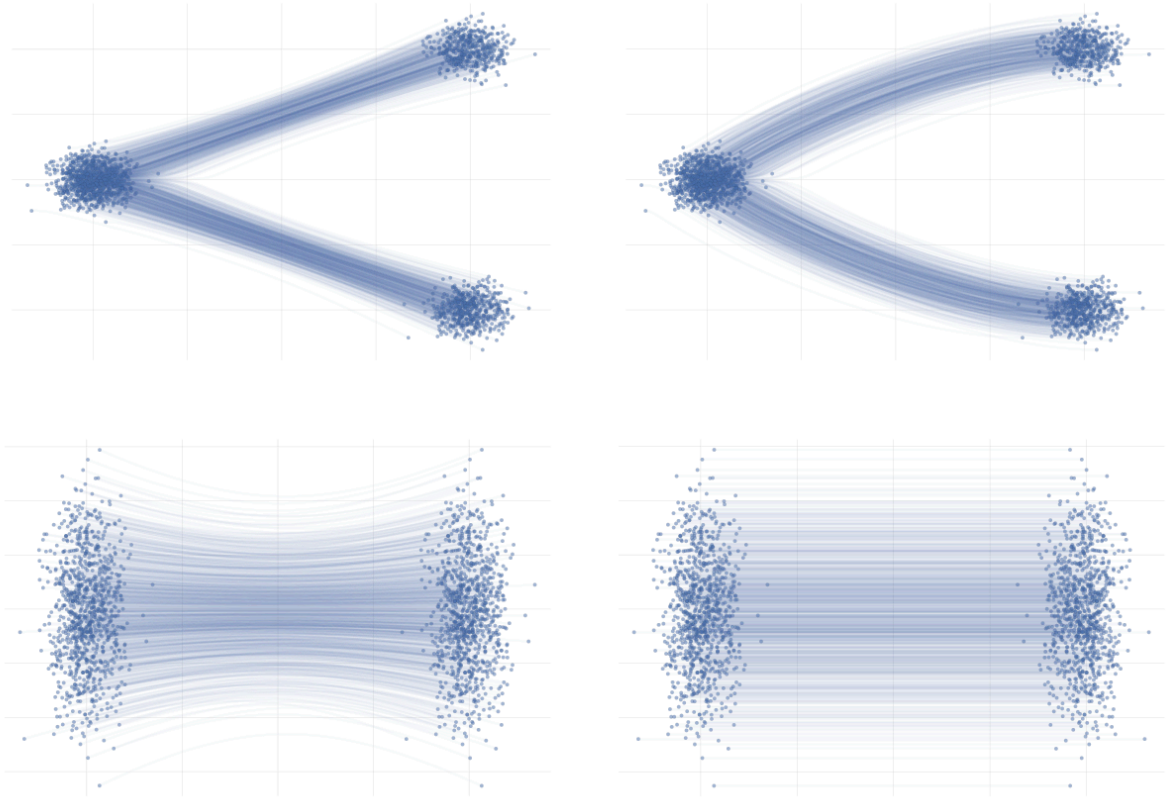

Figure 7: *Different paths with the same endpoints marginals*[8].

CONDITIONAL FLOWS

First, let's remind ourselves that the transport equation relates a vector field u_t to (the time evolution of) a probability path p_t

$$\frac{\partial p_t(x)}{\partial t} = -\nabla \cdot \big(u_t(x) p_t(x)\big),$$

thus constructing p_t or u_t is *equivalent*. One key idea (Lipman et al., 2023 and Albergo & Vanden-Eijnden, 2022) is to express the probability path as a marginal over a joint involving a latent variable z: $p_t(x_t) = \int p(z)\, p_{t|z}(x_t \mid z) \mathrm{d}z$. The $p_{t|z}(x_t \mid z)$ term being a **conditional probability path**, satisfying some boundary conditions at $t = 0$ and $t = 1$ so that p_t be a valid path interpolating between q_0 and q_1. In addition, as opposed to the marginal p_t , the conditional $p_{t|1}$ could be available in closed-form.

In particular, as we have access to data samples $x_1 \sim q_1$, it sounds pretty reasonable to condition on $z = x_1$, leading to the following marginal probabilithy path

$$p_t(x_t) = \int q_1(x_1)\, p_{t|1}(x_t \mid x_1)\, \mathrm{d}x_1.$$

In this setting, the conditional probability path $p_{t|1}$ needs to satisfy the boundary conditions

$$p_0(x \mid x_1) = p_0 \quad \text{and} \quad p_1(x \mid x_1) = \mathcal{N}(x; x_1, \sigma_{\min}^2 I) \xrightarrow[\sigma_{\min} \to 0]{} \delta_{x_1}(x)$$

with $\sigma_{\min} > 0$ small, and for whatever reference p_0 we choose, typically something "simple" like $p_0(x) = \mathcal{N}(x; 0, I)$, as illustrated in the figure below.

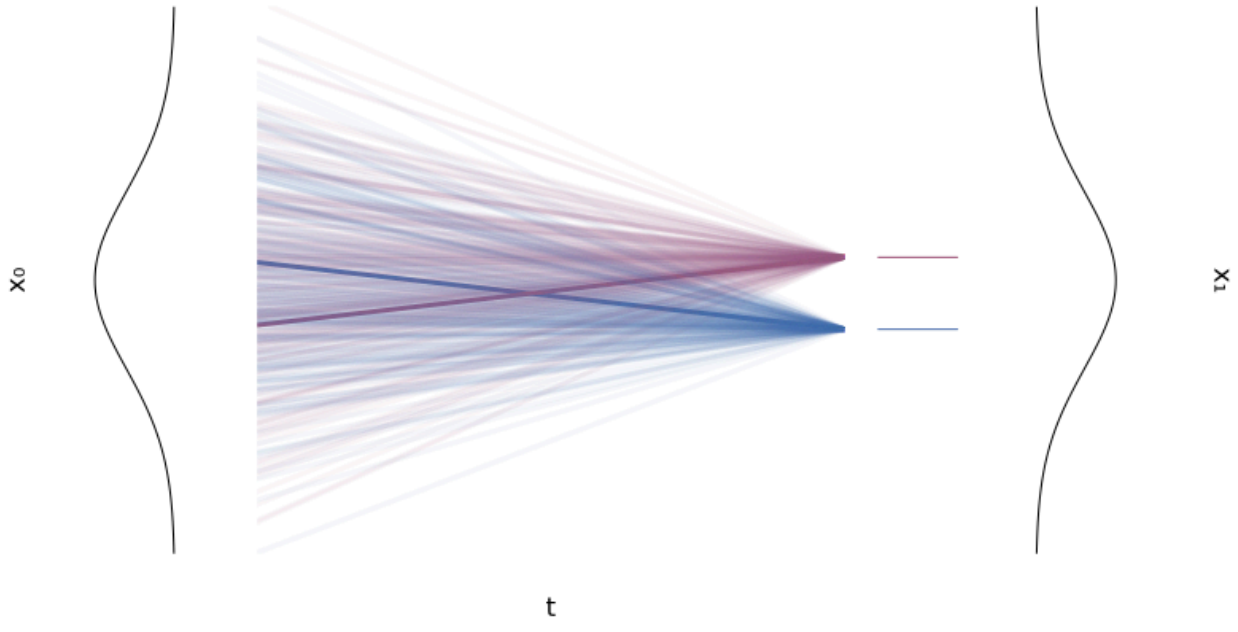

Figure 8: Two conditional flows $\phi_t(x \mid x_1)$ for two univariate Gaussians.

The conditional probability path also satisfies the transport equation with the **conditional vector field** $u_t(x \mid x_1)$:

$$\frac{\partial p_t(x \mid x_1)}{\partial t} = -\nabla \cdot \big(u_t(x \mid x_1) p_t(x \mid x_1)\big). \tag{11}$$

Lipman et al. (2023) introduced the notion of **Conditional Flow Matching (CFM)** by noticing that this *conditional* vector field $u_t(x \mid x_1)$ can express the *marginal* vector $u_t(x)$ of interest via the conditional probability path $p_{t|1}(x_t \mid x_1)$ as

$$\begin{aligned} u_t(x) &= \mathbb{E}_{x_1 \sim p_{1|t}}\left[u_t(x \mid x_1)\right] \\ &= \int u_t(x \mid x_1) \frac{p_t(x \mid x_1) q_1(x_1)}{p_t(x)} \mathrm{d}x_1. \end{aligned} \tag{12}$$

To see why this u_t the same the vector field as the one defined earlier, i.e. the one generating the (marginal) probability path p_t, we need to show that the expression above for the marginal vector field $u_t(x)$ satisfies the transport equation

$$\frac{\partial p_t(x)}{\partial t} = -\nabla \cdot \big(u_t(x) p_t(x)\big).$$

Writing out the left-hand side, we have

$$\begin{aligned} \frac{\partial p_t(x)}{\partial t} &= \frac{\partial}{\partial t} \int p_t(x \mid x_1) q(x_1) \, \mathrm{d}x_1 \\ &= \int \frac{\partial}{\partial t} \big(p_t(x \mid x_1)\big) q(x_1) \, \mathrm{d}x_1 \\ &= -\int \nabla \cdot \big(u_t(x \mid x_1) p_t(x \mid x_1)\big) q(x_1) \, \mathrm{d}x_1 \\ &= -\int \nabla \cdot \big(u_t(x \mid x_1) p_t(x \mid x_1) q(x_1)\big) \, \mathrm{d}x_1 \\ &= -\nabla \cdot \int u_t(x \mid x_1) p_t(x \mid x_1) q(x_1) \, \mathrm{d}x_1 \\ &= -\nabla \cdot \left(\int u_t(x \mid x_1) \frac{p_t(x \mid x_1) q(x_1)}{p_t(x)} p_t(x) \, \mathrm{d}x_1 \right) \\ &= -\nabla \cdot \left(\int u_t(x \mid x_1) \frac{p_t(x \mid x_1) q(x_1)}{p_t(x)} \, \mathrm{d}x_1 \, p_t(x) \right) \\ &= -\nabla \cdot \big(u_t(x) p_t(x)\big) \end{aligned}$$

where in the **first highlighted step** we used (11) and in the **last highlighted step** we used the expression of $u_t(x)$ in (12).

The relation between $\phi_t(x_0)$, $\phi_t(x_0 \mid x_1)$ and their induced densities are illustrated in the Figure 9 below. And since $\phi_t(x_0)$ and $\phi_t(x_0 \mid x_1)$ are solutions corresponding to the vector fields $u_t(x)$ and $u_t(x \mid x_1)$ with $x(0) = x_0$, Figure 9 is equivalent to Figure 10, but note the difference in the expectation taken to go from $u_t(x_0 \mid x_1) \longrightarrow u_t(x_0)$ compared to $\phi_t(x_0 \mid x_1) \longrightarrow \phi_t(x_0)$.

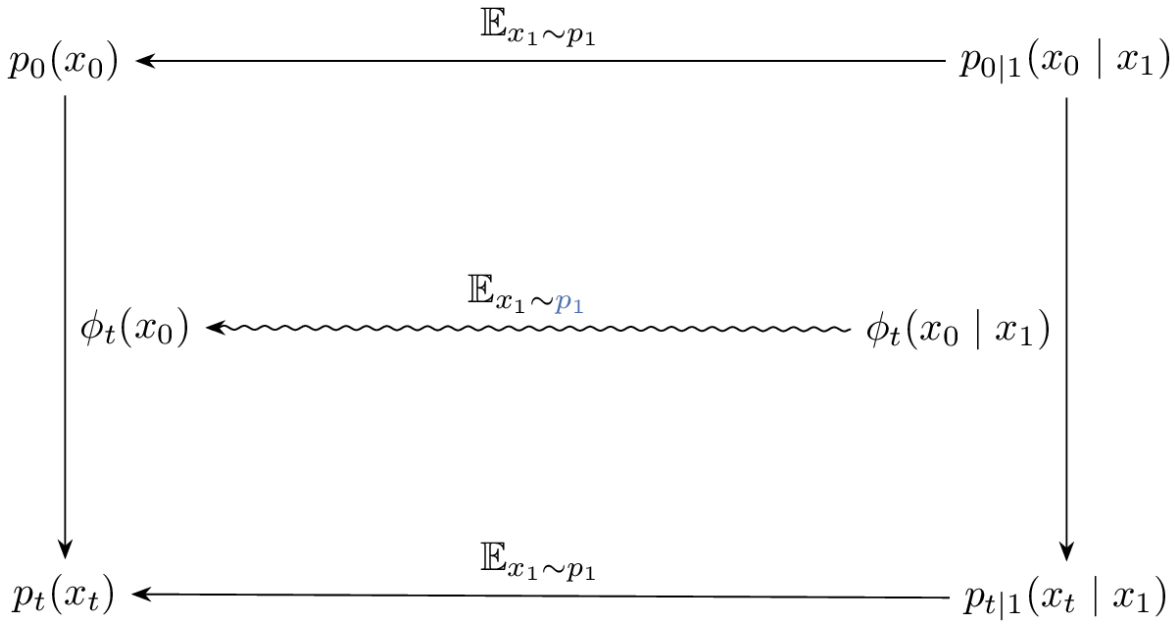

Figure 9: Diagram illustrating the relation between the paths $\phi_t(x_0)$, $\phi_t(x_0 \mid x_1)$, and their induced marginal and conditional densities.

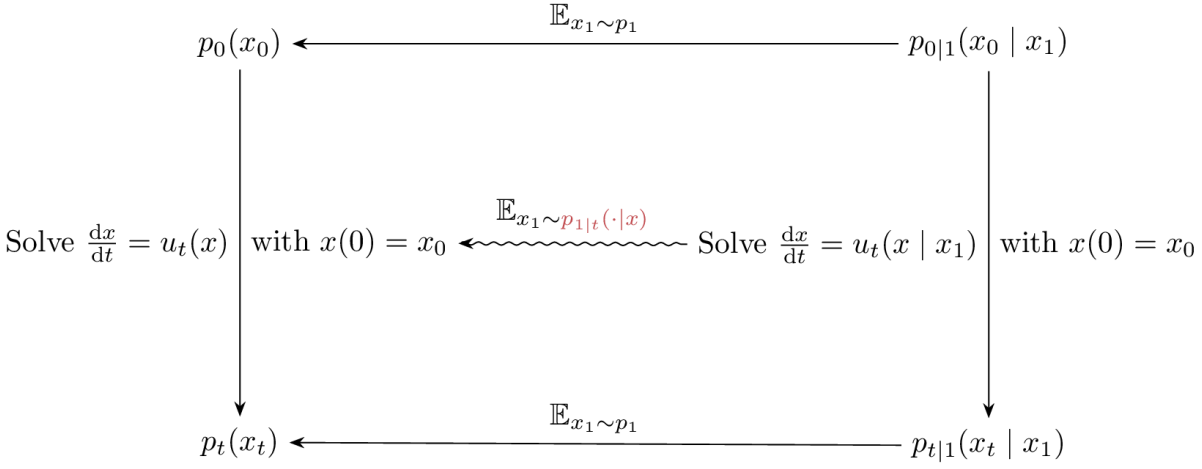

Figure 10: Diagram illustrating the relation between the vector fields $u_t(x_0)$, $u_t(x_0 \mid x_1)$, and their induced marginal and conditional densities.

Gaussian to Gaussian (2D) using a conditional flow

Let's try to gain some intuition behind (12) and the relation between $u_t(x)$ and $u_t(x \mid x_1)$. We do so by looking at the following scenario

$$p_0 = \mathcal{N}([-\mu, 0], I) \quad \text{and} \quad p_1 = \mathcal{N}([+\mu, 0], I) \\ \text{with} \quad \phi_t(x_0 \mid x_1) = (1-t)x_0 + t x_1 \tag{G-to-G}$$

with $\mu = 10$ unless otherwise specified. We're effectively transforming a Gaussian to another Gaussian using a simple time-linear map, as illustrated in the following figure.

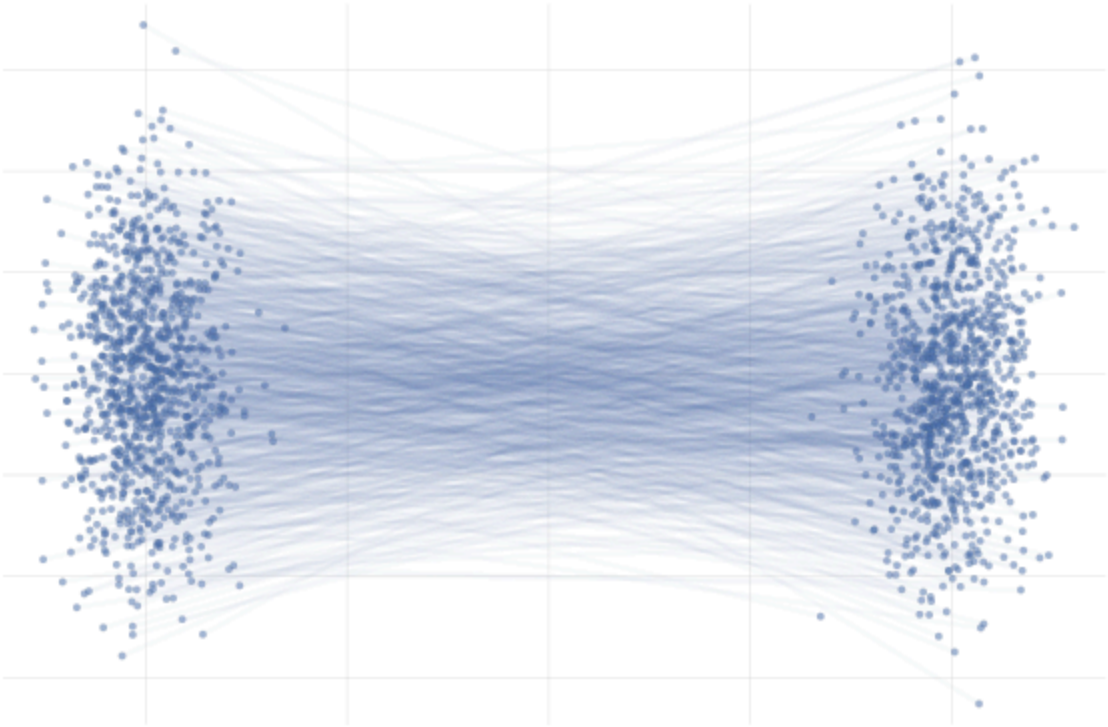

Figure 11: Example conditional paths $\phi_t(x_0 \mid x_1)$ of (G-to-G) with $\mu = 10$.

In the end, we're really just interested in learning the *marginal* paths $\phi_t(x_0)$ for initial points x_0 that are probable under p_0, which we can then use to generate samples $x_1 = \phi_1(x_0)$. In this simple example, we can obain closed-form expressions for $\phi_t(x_0)$ corresponding to the conditional paths $\phi_t(x_0 \mid x_1)$ of (G-to-G), as visualised below.

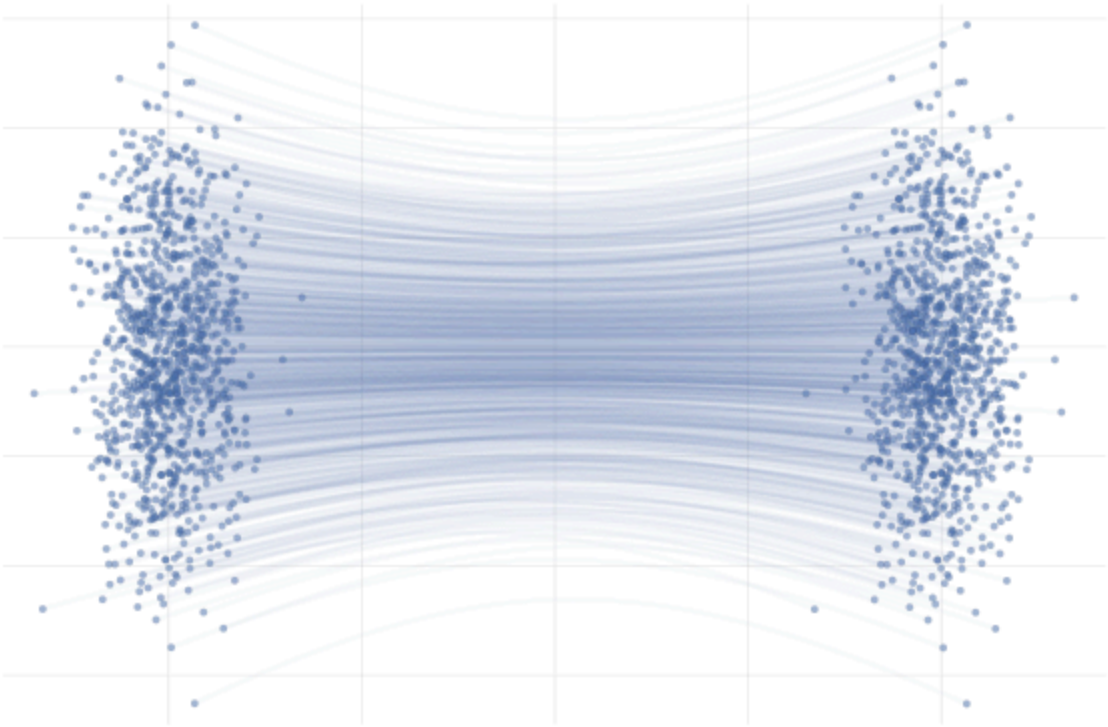

Figure 12: Example marginal paths $\phi_t(x_0)$ of (G-to-G) with $\mu = 10$.

With that in mind, let's pick a random initial point x_0 from p_0, and then compare a MC estimator for $u_t(x_0)$ at different values of t along the path $\phi_t(x_0)$, i.e. we'll be looking at

$$\begin{aligned} u_t\big(\phi_t(x_0)\big) &= \mathbb{E}_{p_{1|t}}\left[u_t\big(\phi_t(x_0) \mid x_1\big)\right] \\ &\approx \frac{1}{n}\sum_{i=1}^{n} u_t\big(\phi_t(x_0) \mid x_1^{(i)}\big) \text{ with } x_1^{(i)} \sim p_{1|t}(x_1 \mid \phi_t(x_0)). \end{aligned}$$

In practice we don't have access to the posterior $p_{1|t}(x_1 \mid x_t)$, but in this specific setting we do have closed-form expressions for everything (Albergo & Vanden-Eijnden, 2022), and so we can visualise the marginal vector field $u_t\big(\phi_t(x_0)\big)$ and the conditional vector fields $u_t\big(\phi_t(x_0) \mid x_1^{(i)}\big)$ for all our "data" samples $x_1^{(i)}$ and see how they compare. This is shown in the figure below.

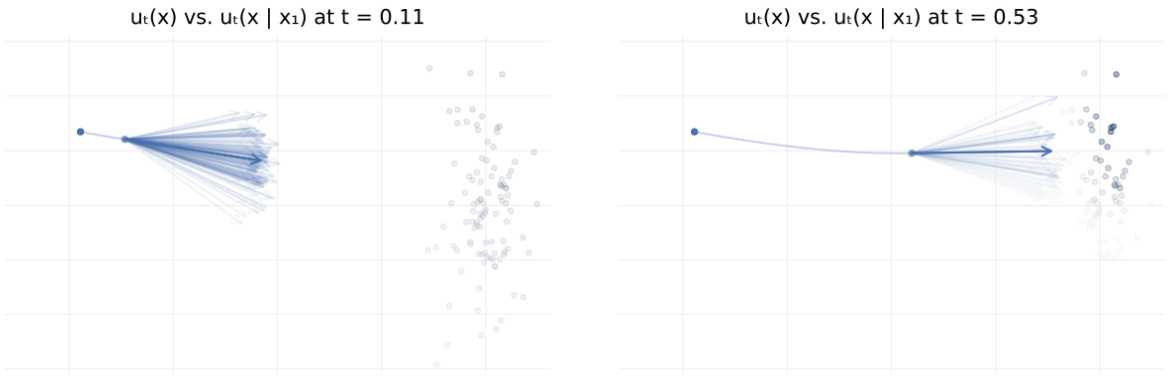

Figure 13: Marginal vector field $u_t(x)$ vs. conditional vector field $u_t(x \mid x_1)$ for samples $x_1 \sim p_1$. Here $p_0 = p_1 = \mathcal{N}(0, 1)$ and the two trajectories are according to the marginal vector field $u_t(x)$. Samples x_1 transparency is given by the IS weight $p_t(x \mid x_1)/p_t(x)$.

From the above figures, we can immediately see how for small t, i.e. near 0, the posterior $p_{1|t}(x_1 \mid x_t)$ is quite scattered so the marginalisation giving u_t involves many equally likely data samples x_1. In contrast, when t increases and get closer to 1, $p_{1|t}(x_1 \mid x_t)$ gets quite concentrated over much fewer samples x_1.

Moreover, equipped with the knowledge of (12), we can replace

$$\mathcal{L}_{\text{FM}}(\theta) = \mathbb{E}_{t\sim\mathcal{U}[0,1],x\sim p_t}\left[\|u_\theta(t,x) - u(t,x)\|^2\right], \tag{13}$$

where $u_t(x) = \mathbb{E}_{x_1\sim p_{1|t}}\left[u_t(x \mid x_1)\right]$, with an equivalent loss regressing the *conditional* vector field $u_t(x \mid x_1)$ and marginalising x_1 instead:

$$\mathcal{L}_{\text{CFM}}(\theta) = \mathbb{E}_{t\sim\mathcal{U}[0,1],x_1\sim q,x_t\sim p_t(x|x_1)}\left[\|u_\theta(t,x) - u_t(x \mid x_1)\|^2\right].$$

These losses are equivalent in the sense that

$$\nabla_\theta \mathcal{L}_{\text{FM}}(\theta) = \nabla_\theta \mathcal{L}_{\text{CFM}}(\theta),$$

which implies that we can use $\mathcal{L}_{\text{CFM}}$ instead to train the parametric vector field u_θ. We defer the full proof to the footnote[9], but show the key idea below. By developing the squared norm in both losses, we can easily show that the squared terms are equal or

independent of θ. Let's develop the inner product term for $\mathcal{L}_{\text{FM}}$ and show that it is equal to the inner product of $\mathcal{L}_{\text{CFM}}$:

$$\begin{aligned}
\mathbb{E}_{x\sim p_t}\left\langle u_\theta(t,x), u_t(x)\right\rangle &= \int \langle u_\theta(t,x), \int u_t(x\mid x_1)\frac{p_t(x\mid x_1)q(x_1)}{p_t(x)}dx_1\rangle p_t(x)\mathrm{d}x \\
&= \int \left\langle u_\theta(t,x), \int u_t(x\mid x_1)p_t(x\mid x_1)q(x_1)dx_1\right\rangle \mathrm{d}x \\
&= \int\int \langle u_\theta(t,x), u_t(x\mid x_1)\rangle p_t(x\mid x_1)q(x_1)dx_1\,\mathrm{d}x \\
&= \mathbb{E}_{q_1(x_1)p(x|x_1)}\left\langle u_\theta(t,x), u_t(x\mid x_1)\right\rangle
\end{aligned}$$

where in the first highlighted step we used the expression of $u_t(x)$ in (12).

The benefit of the CFM loss is that once we define the conditional probability path $p_t(x\mid x_1)$, we can construct an unbiased Monte Carlo estimator of the objective using samples $\left(x_1^{(i)}\right)_{i=1}^n$ from the data target q_1!

This estimator can be efficiently computed as it involves an expectation over the joint $q_1(x_1)p_t(x\mid x_1)$, of the conditional vector field $u_t(x\mid x_1)$ both being available as opposed to the marginal vector field u_t which involves an expectation over the posterior $p_{1|t}(x_1\mid x)$.

We note that, as opposed to the log-likelihood maximisation loss of CNFs which does not put any preference over which vector field u_t can be learned, the CFM loss does specify one via the choice of a conditional vector field, which will be regressed by the neural vector field u_θ.

GAUSSIAN PROBABILITY PATHS

Let's now look at practical example of conditional vector field and the corresponding probability path. Suppose we want conditional vector field which generates a path of Gaussians, i.e.

$$p_t(x\mid x_1) = \mathcal{N}(x;\mu_t(x_1),\sigma_t(x_1)^2\mathrm{I})$$

for some mean $\mu_t(x_1)$ and standard deviation $\sigma_t(x_1)$.

One conditional vector field inducing the above-defined conditional probability path is given by the following expression:

$$u_t(x \mid x_1) = \frac{\dot{\sigma}_t(x_1)}{\sigma_t(x_1)}(x - \mu_t(x_1)) + \dot{\mu}_t(x_1) \tag{18}$$

as shown in the proof below.

Proof ▶️

Example: Linear interpolation 🔽

A simple choice for the mean $\mu_t(x_1)$ and std. $\sigma_t(x_1)$ is the linear interpolation for both, i.e.

$$\begin{aligned} \mu_t(x_1) &\triangleq t x_1 \qquad \text{and} \quad \sigma_t(x_1) \triangleq (1-t) + t\sigma_{\min} \\ \dot{\mu}_t(x_1) &\triangleq x_1 \qquad \text{and} \quad \dot{\sigma}_t(x_1) \triangleq -1 + \sigma_{\min} \end{aligned}$$

so that

$$\big(\mu_0(x_1) + \sigma_0(x_1)x_1\big) \sim p_0 \quad \text{and} \quad \big(\mu_1(x_1) + \sigma_1(x_1)x_1\big) \sim \mathcal{N}(x_1, \sigma_{\min}^2 I)$$

In addition, letting $p_0 = \mathcal{N}([-\mu, 0], I)$ and $p_1 = \mathcal{N}([+\mu, 0], I)$ for some $\mu > 0$, we're back to the G-to-G example from earlier.

We can then plug this choice of $\mu_t(x_1)$ and $\sigma_t(x_1)$ into (18) to obtain the conditional vector field, writing $\sigma_t(x_1) = 1 - (1 - \sigma_{\min})t$ to make our lives simpler,

$$\begin{aligned} u_t(x \mid x_1) &= \frac{-(1-\sigma_{\min})}{1 - (1-\sigma_{\min})t}(x - t x_1) + x_1 \\ &= \frac{1}{(1-t) + t\sigma_{\min}}\Big(- (1-\sigma_{\min})(x - t x_1) + \big(1 - (1-\sigma_{\min})t\big)x_1 \Big) \\ &= \frac{1}{(1-t) + t\sigma_{\min}}\Big(- (1-\sigma_{\min})x + x_1 \Big) \\ &= \frac{x_1 - (1-\sigma_{\min})x}{1 - (1-\sigma_{\min})t}. \end{aligned}$$

Below you can see the difference between $\phi_t(x_0)$ (top figure) and $\phi_t(x_0 \mid x_1)$ (bottom figure) for pairs (x_0, x_1) with $x_0 \sim p_0$ and $x_1 = \phi_t(x_0)$. The paths are coloured by the sign of the 2nd vector component of x_0 to more clearly highlight the difference between the marginal and conditional flows.

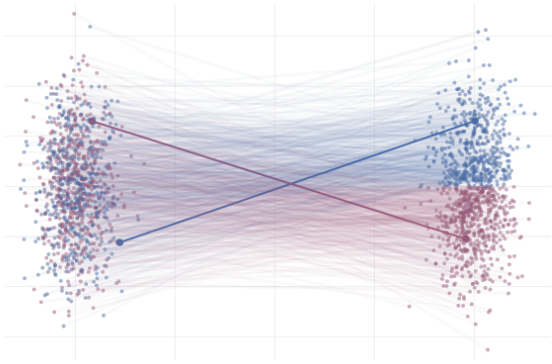

Figure 14: Realizations of paths from p_0 to p_1 following conditional vector fields $u_t(x \mid x_1)$. Paths are highlighted by the sign of the 2nd vector component at time $t = 1$.

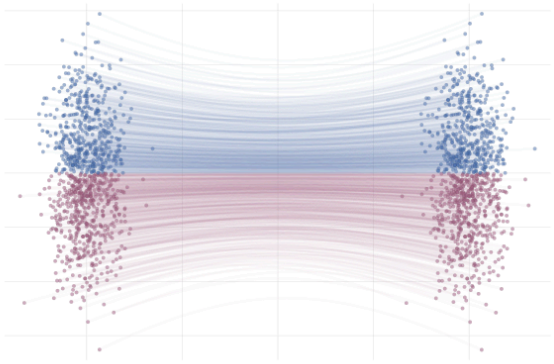

Figure 15: Paths from p_0 to p_1 following the true marginal vector field $u_t(x)$. Paths are highlighted by the sign of the 2nd vector component.

BUT IS CFM REALLY ALL RAINBOWS AND UNICORNS?

Unfortunately not, no. There are two issues arising from *crossing conditional paths*. We will explain this just after, but now we stress that this leads to

1. Non-straight marginal paths ⇒ ODE hard to integrate ⇒ slow sampling at inference.
2. Many possible x_1 for a noised x_t ⇒ high CFM loss variance ⇒ slow training convergence.

To get a better understanding of what these two points above, let's revisit the G-to-G example once more. As we see in the figures below, realizations of the conditional vector field $u_t(x \mid x_1)$, i.e. sampling from the process

$$x_1 \sim q$$
$$x_t \triangleq \phi_t(x \mid x_1)$$

result in paths that are quite different from the marginal paths as illustrated in the figures below.

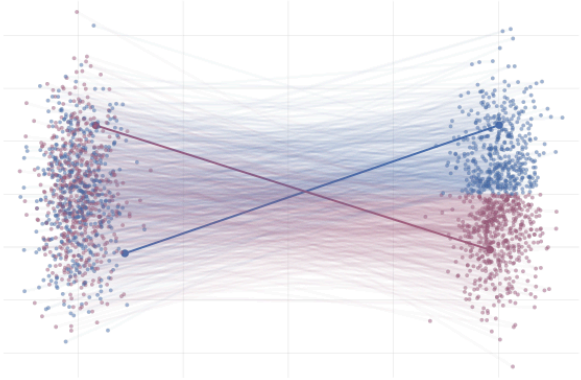

Figure 16: Realizations of conditional paths from $p_0 = p_1 = \mathcal{N}(0, 1)$ for two different $x_1^{(i)}, x_1^{(2)} \sim q$ with conditional vector field given by $u_t(x \mid x_1) = (1 - t)x + tx_1$.

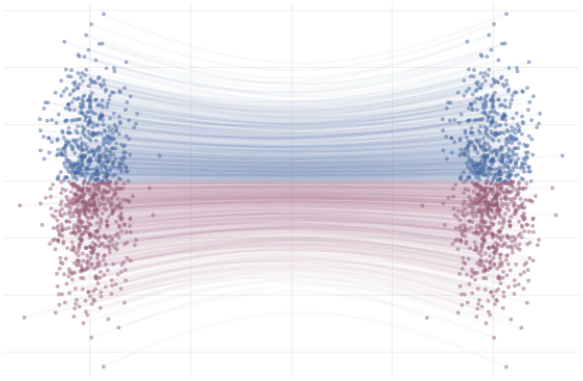

Figure 17: Paths from p_0 to p_1 following the true marginal vector field $u_t(x)$. Paths are highlighted by the sign of the 2nd vector component.

In particular, we can see that the marginal paths $\phi_t(x)$ *do not cross*; this is indeed just the uniqueness property of ODE solutions. A realization of the conditional vector field $u_t(x \mid x_1)$ also exhibits the "non-crossing paths" property, similar to the marginal flows $\phi_t(x)$, however paths $\phi_t(x \mid x_1)$ corresponding to *different* realizations $x_1 \sim q_1$ may intersect, as highlighted in the figure above.

Consider two highlighted paths in the visualization of $u_t(x \mid x_1)$, with data samples $x_1^{(1)}$ and $x_1^{(2)}$. When learning a parameterized vector field $u_\theta(t, x)$ via stochastic gradient descent (SGD), we approximate the CFM loss as:

$$\mathcal{L}_{\text{CFM}}(\theta) \approx \frac{1}{2}\left\| u_\theta(t, x_t^{(1)}) - u(t, x_t^{(1)} \mid x_1^{(1)}) \right\| + \frac{1}{2}\left\| u_\theta(t, x_t^{(2)}) - u(t, x_t^{(2)} \mid x_1^{(2)}) \right\|$$

where $t \sim \mathcal{U}[0, 1]$, $x_1^{(1)}, x_1^{(2)} \sim q_1$, and $x_t^{(1)} \sim p_t(\cdot \mid x_1^{(1)}), x_t^{(2)} \sim p_t(\cdot \mid x_1^{(2)})$. We compute the gradient with respect to θ for a gradient step.

In such a scenario, we're attempting to align $u_\theta(t, x)$ with two different vector fields whose corresponding paths are impossible under the marginal vector field $u(t, x)$ that we're trying to learn! This fact can lead to increased variance in the gradient estimate, and thus slower convergence.

In slightly more complex scenarios, the situation becomes even more striking. Below we see a nice example from Liu et al. (2022) where our reference and target are two different mixture of Gaussians in 2D differing only by the sign of the mean in the x-component. Specifically,

$$\begin{aligned} p_0 &= (1/2)\mathcal{N}([-\mu, -\mu], I) + (1/2)\mathcal{N}([-\mu, +\mu], I) \\ \textbf{and} \quad p_1 &= (1/2)\mathcal{N}([+\mu, -\mu], I) + (1/2)\mathcal{N}([+\mu, +\mu], I) \\ \textbf{with} \quad \phi_t(x_0 \mid x_1) &= (1-t)x_0 + tx_1 \end{aligned}$$

where we set $\mu = 10$, unless otherwise specified.

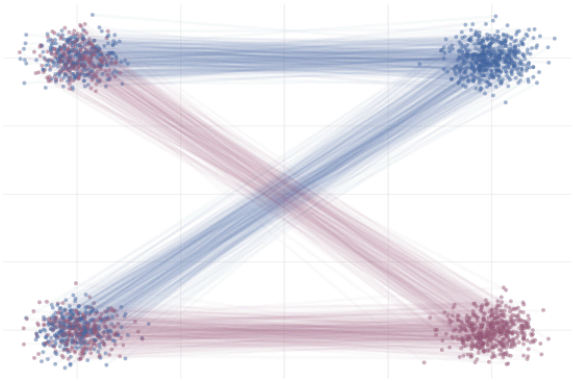

Figure 18: Realizations of conditional paths following conditional vector field $u_t(x \mid x_1)$ from (MoG-to-MoG). Paths are highlighted by the sign of the 2nd vector component.

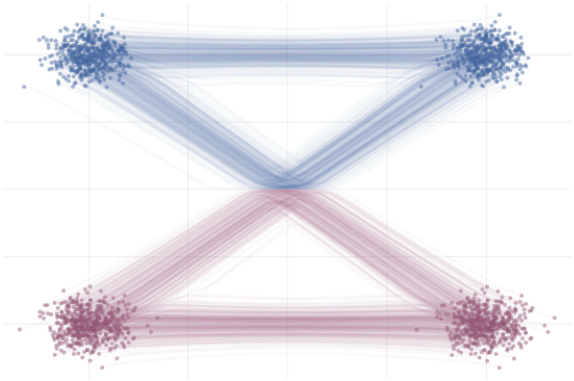

Figure 19: Realizations of marginal paths following the marginal vector field $u_t(x)$ from (MoG-to-MoG). Paths are highlighted by the sign of the 2nd vector component.

Here we see that marginal paths (bottom figure) end up looking *very* different from the conditional paths (top figure). Indeed, at training time paths may intersect, whilst at sampling time they cannot (due to the uniqueness of the ODE solution). As such we see on the bottom plot that some (marginal) paths are quite curved and would therefore require a greater number of discretisation steps from the ODE solver during inference.

We can also see how this leads to a significant variance of the CFM loss estimate for $t \approx 0.5$ in the figure below. More generally, samples from the reference distribution which are arbitrarily close to eachothers can be associated with either target modes, leading to high variance in the vector field regression loss.

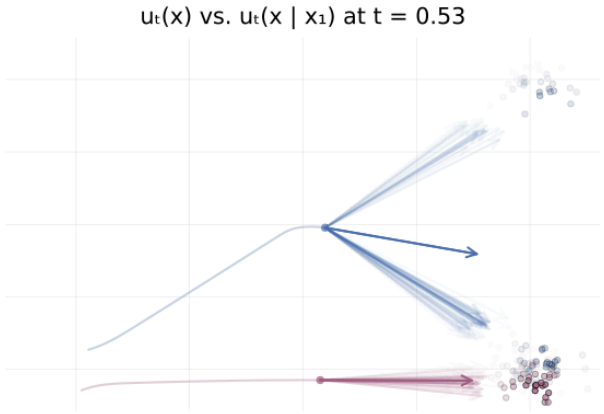

Figure 20: Realizations of conditional paths $\phi_t(x_0 \mid x_1)$ following the conditional vector field $u_t(x \mid x_1)$ for (MoG-to-MoG).

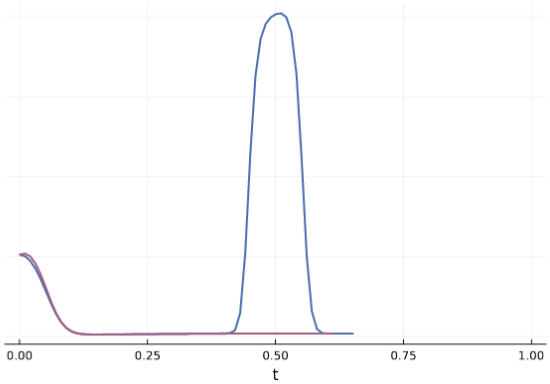

Figure 21: Variance of conditional vector field over $p_{1|t}$ for both blue and red trajectories for (MoG-to-MoG).

An intuitive solution would be to associate data samples with reference samples which are close instead of some arbitrary pairing. We'll detail this idea next via the concept of couplings and optimal transport.

COUPLING

So far we have constructed the vector field u_t by conditioning and marginalising over data points x_1. This is referred to as a *one-sided conditioning*, where the probability path is constructed by marginalising over $z = x_1$:

$$p_t(x_t) = \int p_t(x_t \mid z) q(z) \, \mathrm{d}z = \int p_t(x_t \mid x_1) q(x_1) \, \mathrm{d}x_1 \tag{20}$$

e.g. $p(x_t \mid x_1) = \mathcal{N}(x_t \mid x_1, (1-t)^2)$.

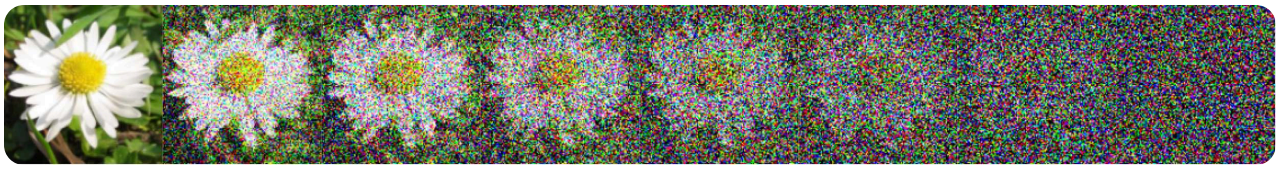

Figure 22: One sided interpolation. Source: Figure (2) in Albergo & Vanden-Eijnden (2022).

Yet, more generally, we can consider conditioning and marginalising over latent variables z, and minimising the following loss:

$$\mathcal{L}_{\mathrm{CFM}}(\theta) = \mathbb{E}_{(t,z,x_t) \sim \mathcal{U}[0,1] q(z) p(\cdot \mid z)} [\| u_\theta(t, x_t) - u_t(x_t \mid z) \|^2]. \tag{21}$$

As suggested in Liu et al. (2023), Tong et al. (2023), Albergo & Vanden-Eijnden (2022) and Pooladian et al. (2023) one can condition on *both* endpoints $z = (x_1, x_0)$ of the process, referred as *two-sided conditioning*. The marginal probability path is defined as:

$$p_t(x_t) = \int p_t(x_t \mid z) q(z) \, \mathrm{d}z = \int p_t(x_t \mid x_1, x_0) q(x_1, x_0) \, \mathrm{d}x_1 \, \mathrm{d}x_0. \tag{22}$$

The following boundary condition on $p_t(x_t \mid x_1, x_0)$: $p_0(\cdot \mid x_1, x_0) = \delta_{x_0}$ and $p_1(\cdot \mid x_1, x_0) = \delta_{x_1}$ is required so that the marginal has the proper conditions $p_0 = q_0$ and $p_1 = q_1$.

For instance, a deterministic linear interpolation gives $p(x_t \mid x_0, x_1) = \delta_{(1-t)x_0 + t x_1}(x_t)$ and the simplest choice regarding the coupling $z = (x_1, x_0)$ is to consider independent samples: $q(x_1, x_0) = q_1(x_1) q_0(x_0)$.

Figure 23: Two sided interpolation. Source: Figure (2) in Albergo & Vanden-Eijnden (2022).

One main advantage is that this allows for non Gaussian reference distribution q_0. Choosing a standard normal as noise distribution $q(x_0) = \mathcal{N}(0, \mathrm{I})$ we recover the same *one-sided* conditional probability path as earlier:

$$p(x_t \mid x_1) = \int p(x_t \mid x_0, x_1) q(x_0) \, \mathrm{d}x_0 = \mathcal{N}(x_t \mid t x_1, (1-t)^2). \tag{23}$$

Optimal Transport (OT) coupling

Now let's go back to the idea of *not* using an independent coupling (i.e. pairing) but instead to correlate pairs (x_1, x_0) with a joint $q(x_1, x_0) \neq q_1(x_1) q_0(x_0)$. Tong et al. (2023) and Pooladian et al. (2023) suggest using the *optimal transport coupling*

$$q(x_1, x_0) = \pi(x_1, x_0) \in \arg\inf_{\pi \in \Pi} \int \|x_1 - x_0\|_2^2 \mathrm{d}\pi(x_1, x_0) \tag{OT}$$

which minimises the optimal transport (i.e. Wasserstein) cost (Monge, 1781, Peyré and Cuturi 2020). The OT coupling π associates samples x_0 and x_1 such that the total distance is minimised.

This OT coupling is illustrated in the right hand side of the figure below, adapted from Tong et al. (2023). In contrast to the middle figure which an independent coupling, the OT one does not have paths that cross. This leads to lower training variance and faster sampling[10].

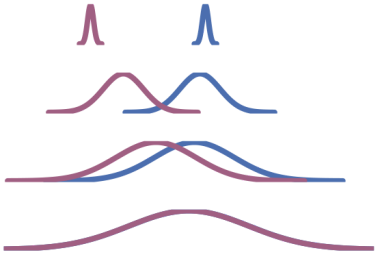

Figure 24: One-sided conditioning (Lipman et al., 2022)

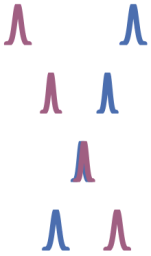

Figure 25: Two-sided conditioning (Tong et al., 2023)

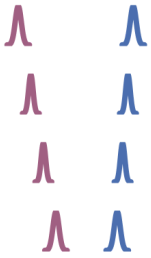

Figure 26: OT coupling (Tong et al., 2023)

In practice, we cannot compute the optimal coupling π between $x_1 \sim q_1$ and $x_0 \sim q_0$, as algorithms solving this problem are only known for finite distributions. In fact, finding a map from q_0 to q_1 is the generative modelling problem that we are trying to solve in the first place!

Tong et al. (2023) and Pooladian et al. (2023) propose to approximate the OT coupling π by computing such optimal coupling only over each mini-batch of data and noise samples, coined **mini-batch OT** (Fatras et al., 2020). This is scalable as for finite collection of samples the OT problem can be computed with quadratic complexity via the Sinkhorn algorithm (Peyre and Cuturi, 2020). This results in a *joint* distribution $\gamma(i, j)$ over "inputs" $\left(x_0^{(i)}\right)_{i=1,\dots,B}$ and "outputs" $\left(x_1^{(j)}\right)_{j=1,\dots,B}$ such that the expected distance is (approximately) minimised. Finally, to construct a mini-batch from this γ which we can subsequently use for training, we can either compute the expectation

wrt. $\gamma(i, j)$ by considering all n^2 pairs (in practice, this can often boil down to only needing to consider n disjoint pairs[11]) or sample a new collection of training pairs $(x_0^{(i')}, x_1^{(j')})$ with $(i', j') \sim \gamma$[12].

For example, we can apply this to the (G-to-G) example from before, which almost completely removes the crossing paths behaviour described earlier, as can be seen in the figure below.

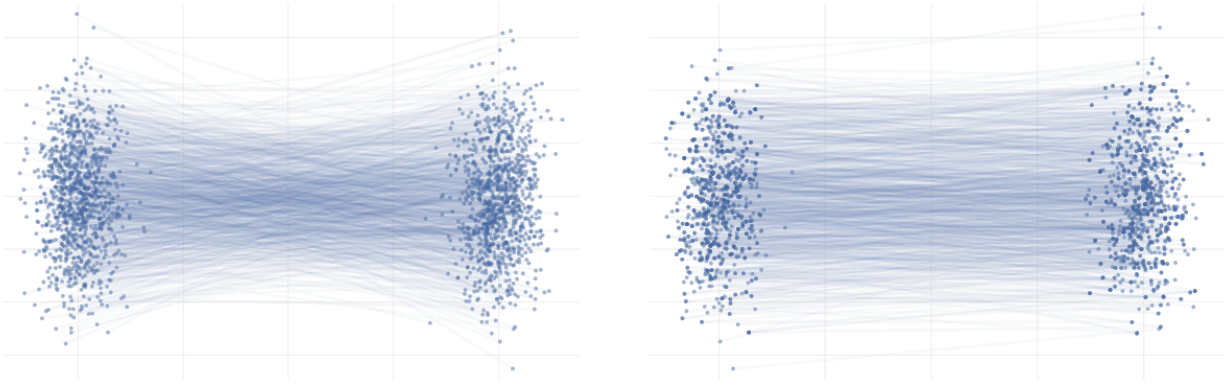

Figure 27: (G-to-G) with uniformly sampled pairings (left) and with OT pairings (right).

We also observe similar behavior when applying this the more complex example (MoG-to-MoG), as can be seen in the figure below.

Figure 28: (MoG-to-MoG) with uniformly sampled pairings (left) and with OT pairings (right).

All in all, making use of mini-batch OT seems to be a strict improvement over the uniform sampling approach to constructing the mini-batch in the above examples and has been shown to improve practical performance in a wide range of applications (Tong et al., 2023; Klein et al., 2023).

It's worth noting that in (OT) we only considered choosing the coupling $\gamma(i, j)$ such that we minimize the expected squared Euclidean distance. This works well in the examples (G-to-G) and (MoG-to-MoG), but we could also replace squared Euclidean distance with some other distance metric when constructing the coupling $\gamma(i, j)$. For example, if we were modeling molecules using CNFs, it might also make sense to pick (i, j) such that $\boldsymbol{x}_0^{(i)}$ and $\boldsymbol{x}_1^{(j)}$ are also rotationally aligned as is done in the work of Klein et al. (2023).

QUICK SUMMARY

In short, we've shown that flow matching is an efficient approach to training continuous normalising flows (CNFs), by directly regressing over the vector field instead of explicitly training by maximum likelihood. This is enabled by constructing the target vector field as the marginalisation of simple conditional vector fields which (marginally) interpolate between the reference and data distribution, but crucially for which we can evaluate and integrate over time. A neural network parameterising the vector field can then be trained by regressing over these conditional vector fields. Similarly to CNFs, samples can be obtained at inference time by solving the ODE associated with the neural vector field.

In this post we have not talked about diffusion (i.e. score based) models on purpose as they are not necessary for understanding flow matching. Yet these are deeply related and even exactly the same in some settings. We are planning to explore these connections, along with generalisations in a follow-up post!

CITATION

Please cite us as:

```
@misc{mathieu2024flow,
  title   = "An Introduction to Flow Matching",
  author  = "Fjelde, Tor and Mathieu, Emile and Dutordoir, Vincent",
  journal = "https://mlg.eng.cam.ac.uk/blog/",
  year    = "2024",
  month   = "January",
  url     = "https://mlg.eng.cam.ac.uk/blog/2024/01/20/flow-matching.html
}
```

ACKNOWLEDGMENTS

We deeply thank Michael Albergo, Valentin Debortoli and James Thornton for giving insightful feedback! And thank you to Andrew Foong for pointing out several typos and rendering issues!

REFERENCES

- Albergo, Michael S. & Vanden-Eijnden, Eric (2023) Building Normalizing Flows with Stochastic Interpolants (https://openreview.net/pdf?id=li7qeBbCR1t).
- Behrmann, Jens and Grathwohl, Will and Chen, Ricky T. Q. and Duvenaud, David and Jacobsen, Joern-Henrik (2019). Invertible Residual Networks (https://proceedings.mlr.press/v97/behrmann19a.html).
- Betker, James, Gabriel Goh, Li Jing, TimBrooks, Jianfeng Wang, Linjie Li, LongOuyang, JuntangZhuang, JoyceLee, YufeiGuo, WesamManassra, PrafullaDhariwal, CaseyChu, YunxinJiao and Aditya Ramesh (2023). Improving Image Generation with Better Captions (https://cdn.openai.com/papers/dall-e-3.pdf).
- Chen & Gopinath (2000). Gaussianization (https://proceedings.neurips.cc/paper_files/paper/2000/file/3c947bc2f7ff007b86a9428b74654de5-Paper.pdf).
- Chen & Lipman (2023). Riemannian Flow Matching on General Geometries (http://arxiv.org/abs/2302.03660v2).
- Chen, Ricky T. Q. and Behrmann, Jens and Duvenaud, David K and Jacobsen, Joern-Henrik (2019). Residual flows for invertible generative modeling (http://arxiv.org/abs/1906.02735).

- De Bortoli, Mathieu & Hutchinson et al. (2022). Riemannian Score-Based Generative Modelling (http://arxiv.org/abs/2202.02763v3).
- Dupont, Doucet & Teh (2019). Augmented Neural Odes (http://arxiv.org/abs/1904.01681v3).
- Friedman (1987). Exploratory projection pursuit (https://www.jstor.org/stable/pdf/2289161.pdf).
- George Papamakarios, Theo Pavlakou, Iain Murray (2018). Masked Autoregressive Flow for Density Estimation (https://proceedings.neurips.cc/paper/2017/file/6c1da886822c67822bcf3679d04369fa-Paper.pdf).
- Huang, Chin-Wei and Krueger, David and Lacoste, Alexandre and Courville, Aaron (2018). Neural Autoregressive Flows (http://arxiv.org/abs/1804.00779).
- Klein, Krämer & Noé (2023). Equivariant Flow Matching (http://arxiv.org/abs/2306.15030v2).
- Lipman, Yaron and Chen, Ricky T. Q. and Ben-Hamu, Heli and Nickel, Maximilian and Le, Matt (2022). Flow Matching for Generative Modeling (http://arxiv.org/abs/2210.02747).
- Liu, Xingchao and Gong, Chengyue and Liu, Qiang (2022). Flow Straight and Fast: Learning to Generate and Transfer Data with Rectified Flow (http://arxiv.org/abs/2209.03003).
- Monge, Gaspard (1781). Mémoire Sur La Théorie Des Déblais et Des Remblais.
- Peyré, Gabriel and Cuturi, Marco (2020). Computational Optimal Transport (http://arxiv.org/abs/1803.00567).
- Pooladian, Aram-Alexandre and {Ben-Hamu}, Heli and {Domingo-Enrich}, Carles and Amos, Brandon and Lipman, Yaron and Chen, Ricky T. Q. (2023). Multisample Flow Matching: Straightening Flows With Minibatch Couplings (http://arxiv.org/abs/2304.14772).
- Song, Sohl-Dickstein & Kingma et al. (2020). Score-Based Generative Modeling Through Stochastic Differential Equations (http://arxiv.org/abs/2011.13456v2).

- Tong, Alexander and Malkin, Nikolay and Fatras, Kilian and Atanackovic, Lazar and Zhang, Yanlei and Huguet, Guillaume and Wolf, Guy and Bengio, Yoshua (2023). Simulation-Free Schrodinger Bridges via Score and Flow Matching (http://arxiv.org/abs/2307.03672).
- Tong, Malkin & Huguet et al. (2023). Improving and Generalizing Flow-Based Generative Models With Minibatch Optimal Transport (http://arxiv.org/abs/2302.00482v2).
- Watson, Joseph L. and Juergens, David and Bennett, Nathaniel R. and Trippe, Brian L. and Yim, Jason and Eisenach, Helen E. and Ahern, Woody and Borst, Andrew J. and Ragotte, Robert J. and Milles, Lukas F. and Wicky, Basile I. M. and Hanikel, Nikita and Pellock, Samuel J. and Courbet, Alexis and Sheffler, William and Wang, Jue and Venkatesh, Preetham and Sappington, Isaac and Torres, Susana V{'a}zquez and Lauko, Anna and De Bortoli, Valentin and Mathieu, Emile and Ovchinnikov, Sergey and Barzilay, Regina and Jaakkola, Tommi S. and DiMaio, Frank and Baek, Minkyung and Baker, David (2023). De Novo Design of Protein Structure and Function with RFdiffusion (https://www.nature.com/articles/s41586-023-06415-8).

FOOTNOTES

1. The property $\phi \circ \phi^{-1} = \mathbf{Id}$ implies, by the chain rule,
$$\left.\frac{\partial \phi}{\partial x}\right|_{x=\phi^{-1}(y)} \left.\frac{\partial \phi^{-1}}{\partial y}\right|_{y} = 0 \iff \left.\frac{\partial \phi}{\partial x}\right|_{x=\phi^{-1}(y)} = \left(\left.\frac{\partial \phi^{-1}}{\partial y}\right|_{y}\right)^{-1} \quad \forall y \in \mathbb{R}^d$$ ↩
2. **Autoregressive** (Papamakarios et al., 2018; Huang et al., 2018) One strategy is to factor the flow's Jacobian to have a triangular structure by factorising the density as $p_\theta(x) = \prod_d p_\theta(x_d; x_{d<})$ with each conditional $p_\theta(x_d; x_{d<})$ being induced via a flow. **Low rank residual** (Van Den Berg et al., 2018) Another approach is to construct a flow via a residual connection: $\phi(x) = x + Ah(Bx + b)$ with parameters $A \in \mathbb{R}^{d \times m}$, $B \in \mathbb{R}^{m \times m}$ and $b \in \mathbb{R}^m$. Leveraging Sylvester's

determinant identity $\det(I_d + AB) = \det(I_m + BA)$, the determinant computation can be reduced to one of a $m \times m$ matrix which is advantageous if $m \ll d$. ↩

3. A sufficient condition for ϕ_k to be invertible is for u_k to be $1/h$-Lipschitz [Behrmann et al., 2019]. The inverse ϕ_k^{-1} can be approximated via fixed-point iteration (Chen et al., 2019). ↩
4. A sufficient condition for ϕ_t to be invertible is for u_t to be Lipschitz and continuous by Picard–Lindelöf theorem. ↩
5. The *Fokker–Planck equation* gives the time evolution of the density induced by a stochastic process. For ODEs where the diffusion term is zero, one recovers the transport equation. ↩
6. Expanding the divergence in the *transport equation* we have: $\frac{\partial}{\partial_t} p_t(x_t) = -(\nabla \cdot (u_t p_t))(x_t) = -p_t(x_t)(\nabla \cdot u_t)(x_t) - \langle \nabla_{x_t} p_t(x_t), u_t(x_t) \rangle$. Yet since x_t also depends on t, to get the *total derivative* we have
$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t} p_t(x_t) &= \frac{\partial}{\partial_t} p_t(x_t) + \langle \nabla_{x_t} p_t(x_t), \frac{\mathrm{d}}{\mathrm{d}t} x_t \rangle \\ &= -p_t(x_t)(\nabla \cdot u_t)(x_t) - \langle \nabla_{x_t} p_t(x_t), u_t(x_t) \rangle + \langle \nabla_{x_t} p_t(x_t), \frac{\mathrm{d}}{\mathrm{d}t} x_t \\ &= -p_t(x_t)(\nabla \cdot u_t)(x_t).\end{aligned}$$
Where the last step comes from $\frac{\mathrm{d}}{\mathrm{d}t} x_t = u_t$. Hence, $\frac{\mathrm{d}}{\mathrm{d}t} \log p_t(x_t) = \frac{1}{p_t(x_t)} \frac{\mathrm{d}}{\mathrm{d}t} p_t(x_t) = -(\nabla \cdot u_t)(x_t)$. ↩
7. The Skilling-Hutchinson trace estimator is given by $\mathbf{Tr}(A) = \mathbb{E}[v^\top A v]$ with $v \sim p$ isotropic and centred. In our setting we are interested in $\mathbf{div}(u_t)(x) = \mathbf{Tr}\left(\frac{\partial u_t(x)}{\partial x}\right) = \mathbb{E}[v^\top \frac{\partial u_t(x)}{\partial x} v]$ which can be approximated with a Monte-Carlo estimator, where the integrand is computed via automatic forward or backward differentiation. ↩
8. The top row is with reference $p_0 = \mathcal{N}([-a, 0], I)$ and target $p_1 = (1/2)\mathcal{N}([a, -10], I) + (1/2)\mathcal{N}([a, 10], I)$, and the bottom row is the G-to-G example. The left column shows the straight-line solutions for the *marginals* and the right column shows the marginal solutions induced by considering the straight-line *conditional* interpolants. ↩

9. Developing the square in both losses we get:
$\|u_\theta(t,x) - u_t(x \mid x_1)\|^2 = \|u_\theta(t,x)\|^2 + \|u_t(x \mid x_1)\|^2 - 2\langle u_\theta(t,x), u_t(x \mid x_1)\rangle$,
and $\|u_\theta(t,x) - u_t(x)\|^2 = \|u_\theta(t,x)\|^2 + \|u_t(x)\|^2 - 2\langle u_\theta(t,x), u_t(x)\rangle$. Taking the expectation over the last inner product term:
$$\begin{aligned}\mathbb{E}_{x\sim p_t}\,\langle u_\theta(t,x), u_t(x)\rangle &= \int \langle u_\theta(t,x), \int u_t(x|x_1)\frac{p_t(x \mid x_1)q(x_1)}{p_t(x)}dx_1\rangle p_t(x)\\ &= \int \langle u_\theta(t,x), \int u_t(x \mid x_1)p_t(x \mid x_1)q(x_1)dx_1\rangle \,\mathrm{d}x\\ &= \int\int \langle u_\theta(t,x), u_t(x \mid x_1)\rangle p_t(x \mid x_1)q(x_1)dx_1\,\mathrm{d}x\\ &= \mathbb{E}_{q_1(x_1)p(x|x_1)}\,\langle u_\theta(t,x), u_t(x \mid x_1)\rangle.\end{aligned}$$
Then we see that the neural network squared norm terms are equal since:
$\mathbb{E}_{p_t}\|u_\theta(t,x)\|^2 = \int \|u_\theta(t,x)\|^2 p_t(x \mid x_1)q(x_1)\,\mathrm{d}x\,\mathrm{d}x_1 = \mathbb{E}_{q_1(x_1)p(x|x_1)}\|u_\theta(t,x)\|^2$
↩

10. Dynamic optimal transport [Benamou and Brenier, 2000]
$W(q_0,q_1)_2^2 = \inf_{p_t,u_t}\int\int_0^1 \|u_t(x)\|^2 p_t(x)\,\mathrm{d}t\,\mathrm{d}x$ ↩

11. In mini-batch OT, we only work with the empirical distributions over $x_0^{(i)}$ and $x_1^{(j)}$, i.e. they all have weights $1/n$, where n is the size of the mini-batch. This means that we can find a γ matching the $\inf$ in (OT) by solving what's referred to as a linear assignment problem (https://en.wikipedia.org/wiki/Assignment_problem). This results in a sparse matrix with exactly n entries, each then with a weight of $1/n$. In such a scenario, computing the expectation over the joint $\gamma(i,j)$, which has n^2 entries but in this case only n non-zero entries, can be done by only considering n training pairs where every i is involved in exactly one pair and similarly for every j. This is usally what's done in practice. When solving the assignment problem is too computationally intensive, using Sinkhorn and a sampling from the coupling might be the preferable approach. ↩

12. Note the size of the resulting mini-batch sampled from $\gamma(i,j)$ does not necessarily have to be of the same size as the mini-batch size used to construct the mini-batch OT approximation as we can sample from γ with replacement, but using the same size is typically done in practice, e.g. Tong et al. (2023). ↩

« Natural-Gradient Variational Inference 2: ImageNet-scale (/blog/2021/11/24/ngvi-bnns-part-2.html)

Published on 20 January 2024.

This is the blog of the Cambridge Machine Learning Group (https://mlg.eng.cam.ac.uk/). Follow us on Twitter @CambridgeMLG (https://twitter.com/CambridgeMLG) and check out our GitHub cambridge-mlg (https://github.com/cambridge-mlg/).