Discrete Flow Matching

References

① Liu et. al., Flowing straight and fast: Learning to generate and transfer data with rectified flow, ICLR'22

② Lipman et. al., Flow matching for generative modeling, 2022

③ Albergo and Vanden-Eijnden, Building normalizing flows with stochastic interpolants, ICLR'23

④ Lipman et. al., Flow matching guide and code, arXiv: 2412.06264, December 2024.

⑤ Campbell et. al., Generative flows on discrete state spaces: Enabling multimodal flows with application to protein co-design, arXiv: 2402.04997, 2024.

⑥ Gat et. al., Discrete flow matching, arXiv: 2407.15595, 2024.

⑦ Shaul et. al., Flow matching with general discrete paths: A kinetic-optimal perspective, arXiv: 2412.03487, 2024.

We closely follow sections 6, 7 in [47]

Continuous Time (Finite State) Markov Chains

State space $S = \mathcal{T}^L$, $\mathcal{T} = \{1, 2, \ldots, K\}$ (vocabulary)

$x = (x^1\ x^2 \ldots x^L)$ (L = # of tokens)

The CTMC model: For $t \in [0,1]$, and h small,

$$P_{t+h|t}(y|x) := P(X_{t+h} = y \mid X_t = x)$$

$$= \delta(y,x) + h u_t(y,x) + o(h)$$

$\hookrightarrow = \begin{cases} 1 & \text{if } y = x \\ 0 & \text{o.w.} \end{cases}$

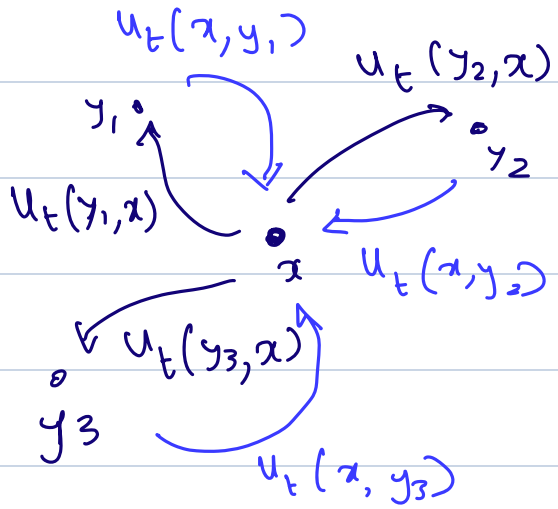
$u_t(y,x)$ is the rate of going to y from x (velocity field)

$$u_t(y,x) \geq 0, \quad y \neq x$$

$$u_t(x,x) < 0$$

s.t.

$$u_t(x,x) = -\sum_{y \neq x} u_t(y,x)$$



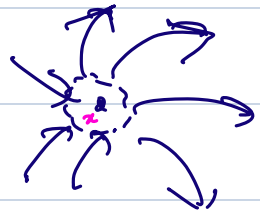
→ (1)

Divergence Review:

ρ: flux → measures the rate of flow from a vector field through a surface (e.g. prob. flux is the volume of prob. per unit time going through a surface).

mass per unit time

$\nabla \cdot \rho$: Divergence is a measure of "accumulation" of flux at a point, given by the sum of flux changes across directions

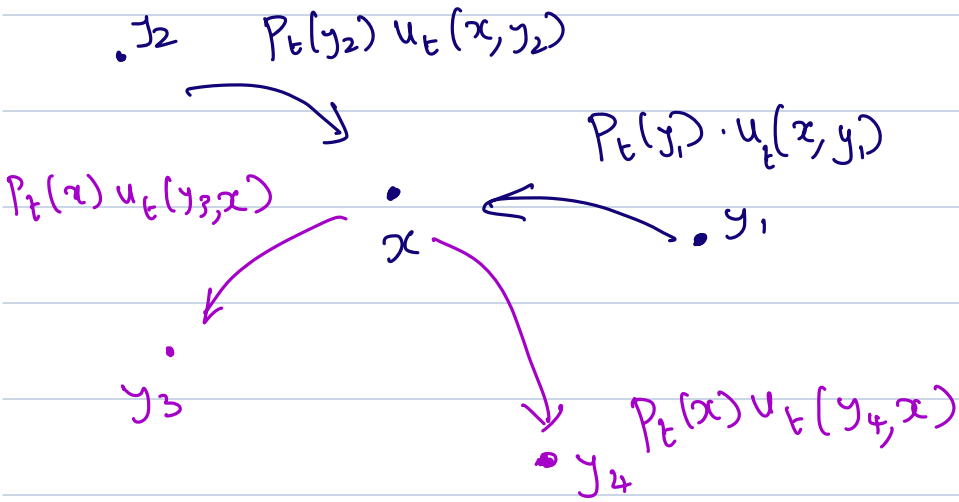


$$\nabla \cdot \rho = \sum_{i=1}^{L} \frac{\partial \rho}{\partial x_i}$$

Recall in the Continuity Eqn: $(p_t u_t)$ was the prob. flux, and $\partial_t p_t = \nabla \cdot (p_t u_t)$

↳ velocity field

A similar analogy holds for CTMC and finite vocabulary:



Incoming prob. flux to state x at time t =

$$\sum_{\{y:\, y \neq x\}} u_t(x, y) \cdot P_t(y)$$

Outgoing flux from x

$$= \sum_{\{y:\, y \neq x\}} u_t(y, x)\, P_t(x)$$

(Discete) Continuity equation:

$$\frac{\partial P_t(x)}{\partial t} = \left(\text{Incoming flux} - \text{Outgoing flux}\right)$$

$$= \sum_{\{y:\, y \neq x\}} u_t(x, y) \cdot P_t(y)$$

$$\sum_{\{y:\, y \neq x\}} u_t(y, x)\, P_t(x)$$

→ (2a)

$$= \nabla \cdot (p_t u_t)$$

The discrete divergence for a CTMC = sum of incoming prob flux − sum of outgoing prob flux.

From (2):

$$\frac{\partial p_t(x)}{\partial t} = \underbrace{\sum_{\{y: y \neq x\}} u_t(x,y) \cdot p_t(y)}_{\text{incoming}} - \underbrace{\sum_{\{y: y \neq x\}} u_t(y,x)\, p_t(x)}_{\text{outgoing}}$$

From (1): $u_t(x,x) = - \sum_{y \neq x} u_t(y,x)$

$$= \sum_{\{y: y \neq x\}} u_t(x,y) \cdot p_t(y) - p_t(x) \underbrace{\sum_{\{y: y \neq x\}} u_t(y,x)}_{= -u_t(x,x)}$$

$$= \sum_{y} u_t(x,y)\, p_t(y)$$

i.e., $$\frac{\partial p_t(x)}{\partial t} = \sum_y u_t(x,y)\, p_t(y)$$

(Kolmogorov backward equation for CTMC)

↳ (2b)

Summary: Analogous to continuous state space diffusions, we have the (discrete) divergence and continuity equation for CTMCs. We can now study density evolution for DLMs in a manner analogous to flow models for continuous state spaces.

Thus, we can characterize the CTMC either by the density evolution or the velocity field, analogous to the cont. statespace flow (see Note 19).

Thm: (Thm 13 in [47])

(p_t, u_t) satisfies continuity equation (2)

$\Leftrightarrow$

u_t generates p_t for the CTMC.

Comment: If a velocity field is divergence-free, i.e., $\tilde{v}_t(y,x)$ s.t. $\sum_x p_t(x) v_t(y,x) = 0$,

then, the velocity fields $\{u_t(y,x)\}$ and $\{u_t(y,x) + \tilde{v}_t(y,x)\}$ both result in the same density evolution.

Relation to DLM in Note 22

In the rest of this note, we discuss discrete flow matching to couple between a noise distribution and a data distribution. Similar to Note 19, this setup can be general; however, in this note, we specialize to the uniform noising model.

Further, analogous to discrete-time DDPM derived through the ELBO bound (Note 10), and the continuous-time SDE (Note 9), we can view this flow model as the continuous-time analog to the DLM derived through ELBO in our

previous note (Note 22).

The setting here is a sequence of tokens, each from the vocabulary $\{1, 2, \ldots, K\}$. The reverse process is initialized from the noise distribution (e.g. here, iid uniform across all tokens). A CTMC evolves this backwards to result in a token sequence sampled from the data distribution.

This continuous time process can be viewed as a time limit of the discrete time evolution that we studied in Note 22 (DLM). Specifically, in the limit where the step-size (h) goes to 0, the ELBO based derivation (which uses a conditional product form reverse process) becomes tight. This is because in this limit, for most of the time-steps no change occurs (the CTMC state is unchanged), and infrequently, atmost one token changes state, with multiple token state changes being a $o(h^2)$ term that is negligible. Thus, the error

due to assuming conditional product form in the reverse process (equation ⊛ on pp. 14, Note 22) is vanishingly small in this limiting regime.

It seems plausible that we can derive concrete discretization error bounds, very similar to that in Note 14 (DDPM Discretization through I-MMSE); this is left for future work.

Discrete Flow Matching

$x_0 \sim p$ (noise distribution: example $p(x_0) = \delta(x_0, m)$, i.e., $x_0 = (m\ m\ \dots\ m)$ or $p(x_0)$ uniformly distributed over vocab., iid across tokens

$\downarrow$

$(x_0^1 \dots x_0^L)$

$x_1 \sim q$ (data distribution)

$(x_0, x_1) \sim p(x_0)\, q(x_1)$ (indep. coupling)

(other dependent couplings allowed in this framework, continuing from Note 19; same document but with different languages [4]).

Sketch of Approach

① Construct a prob. path $p_t(x)$, along with velocity fields $\{u_t(y,x)\}$ to transport from x_0 to x_1 (in this note $x_0 \sim p_0(\cdot)$ uniform)

② To do so, define a conditioning variable $z = (x_0, x_1)$, and derive expressions for $p_{t|z}(x|z)$ and $u_t(y,x|z)$, which are conditional prob. paths.

As in continuous flow matching, we need to specify a vector field $u_t(\cdot,\cdot)$ and associated prob. path $p_t(\cdot)$ to implement the coupling in step ①. However, we do not have a closed form

expression for $u_t(\cdot,\cdot)$. Instead, just like Note 19, we defined conditional vector fields and the associated conditional prob. paths that have a closed form expression. As we will see in step ③, this closed form expression is used in a training loss to solve the unconditional flow matching problem for CTMCs.

Specifically, for $z = (x_0, x_1)$, $t \in [0,1]$,

$$p_t(x) = \sum_z p_{t|z}(x|z)\, p_z(z)$$

$$u_t(y,x) = \sum_z u_t(y,x|z)\, p_{z|t}(z|x)$$

$$= E\left[u_t(y, X_t|z) \,\middle|\, X_t = x\right]$$

Then analogous to pages 6-8, Note 19, from Bayes rule and the continuity equations (for the unconditional and conditional flows), we have that

$u_t(y, x)$ generates $p_t(x)$

↳ meaning $p_t(x)$ defined through the continuity eqn: 2a, 2b

$u_t(y, x|z)$ generates $p_{t|z}(x|z)$

Parameterization: We now parameterize the conditional flow and the associated conditional probs, in a manner that enables scalable implementation.

We parameterize $p_{t|z}(\cdot|\cdot)$ in product form across tokens (but conditioned on the entire vector z):

$$p_{t|z}(x|z) = \prod_i p^i_{t|z}(x^i|z) \, ;$$

with $z = (x_0, x_1)$, and with $t \in [0, 1]$,

$$p^i_{t|z}(x^i|z) = \kappa_t\, \delta(x^i, x^i_1) + (1-\kappa_t)\, \delta(x^i, x^i_0)$$

κ_t → time varying interpolation

e.g., $(\kappa_t = t)$ leads to a straight-line interpolation in the conditional dist.

along with the associated generating conditional velocity fields $u^i_t(y^i, x^i|z)$, and characterize their relation using Kolmogorov backward eqn (or equivalently, the continuity eqn):

$$\frac{d}{dt} p^i_{t|z}(y^i|z) = \sum_{x^i} u^i_t(y^i, x^i|z) \cdot p^i_{t|z}(y^i|z)$$

This results in:

$$u^i_t(y^i, x^i|z) = \frac{\dot{\kappa}_t}{1-\kappa_t} \left(\delta(y^i, x^i_1) - \delta(y^i, x^i) \right)$$

$x^i \to (x_0, x_1)$

Importantly, this is a closed form expression, that can be used in the loss function when

learning a velocity field $u^\theta(\cdot,\cdot)$.

Furthermore, observe that the conditional velocity field is factorized (product form across tokens); this implies that we model the process to have jumps for each token conditionally independent of any other token. Furthermore, we can show (see Theorem 16 in [4], pp. 45) that this results in the overall velocity flow being in factorized form:

$$u_t(y,x) = \sum_i \delta(y^{\bar{i}}, x^{\bar{i}})\, u_t^i(y^i, x)$$

$\bar{i} = (1, 2, \dots, i-1, i+1, \dots, 2)$

In summary, this parameterization leads the following (forward) sampling process:

i^{th} token

$$P(X_{t+h}^i = y^i \mid X_t = x) = \delta(y^i, x^i) + h\, u^i(y^i, x) + o(h)$$

note the joint vector in conditioning

$$= \sum_{x_1^i} \left(\delta(y^i, x^i) + h \frac{\dot{\kappa}_t}{1-\kappa_t} \left(\delta(y^i, x_1^i) - \delta(y^i, x^i) \right) + o(h) \right) \cdot p^i_{1|t}(x_1^i | x)$$

(Note that there are two design choices above: $Z = (X_0, X_1)$, and $p_{t|Z}(\cdot|\cdot)$ is product form + interpolation. Please see [4] for the general setup beyond these choices. However, for scalable implementation, these choices are important.)

---

(3) Training via flow matching loss

We use the (generalized) KL divergence as the instance of Bregman divergence for discrete flow matching (recall we used the squared loss as the Bregman divergence in Note 19).

Specifically, let $\Omega_x = \{v : v(y) \geq 0 \ \forall\, y \neq x,\ v(x) = -\sum_{y \neq x} v(y)\}$

$\phi_x : \Omega_x \longrightarrow \mathbb{R}$ a convex function, and $D_{\phi_x}(\cdot,\cdot)$ the associated Bregman Divergence.

Then, analogous to Note 19, in the discrete case, we have:

$$\mathcal{L}_{DFM}(\theta) = \mathbb{E}_{t, x_t \sim p_t}\left[D_{\phi_{x_t}}\left(u_t(\cdot, x_t), u_t^\theta(\cdot, x_t)\right)\right]$$

$$\mathcal{L}_{CDFM}(\theta) = \mathbb{E}_{t, z, x_t \sim p_{t|z}}\left[D^i_{x_t}\left(u^i_t(\cdot, x_t|z), u^{\theta,i}_t(\cdot, x_t)\right)\right]$$

Then, $\nabla_\theta \mathcal{L}_{DFM}(\theta) = \nabla_\theta \mathcal{L}_{CDFM}(\theta)$.

Thus, we can train with conditional discrete flow matching, for which $u^i_t(\cdot, \cdot|z)$ is given

in closed form. We can specialize CDFM loss to either the uniform noising case or the masking case (see [7], Appendix D) which then recovers the ELBO loss that we discussed in Note 22.