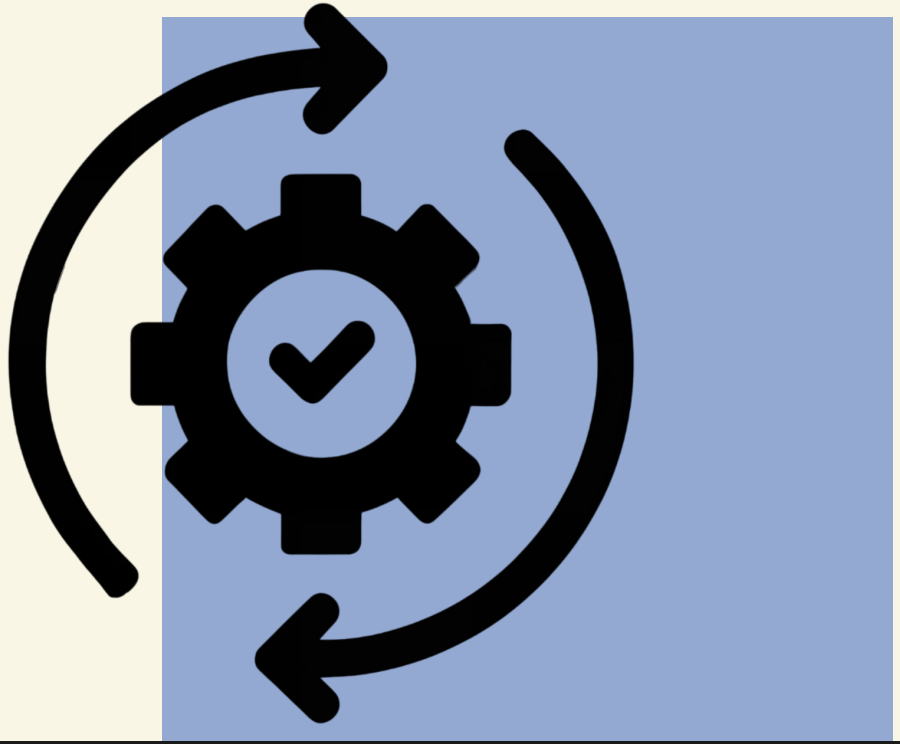
Эффективные системы машинного обучения



Лекция 9

Дискретные и непрерывные диффузионные модели

Преподаватель

Оганов Александр

22 ноября 2024

ВМК МГУ

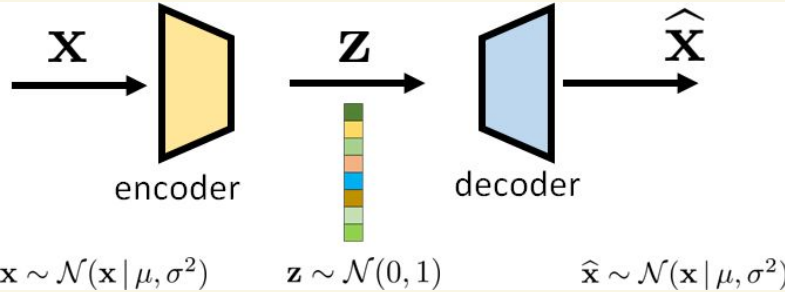
Генеративные модели

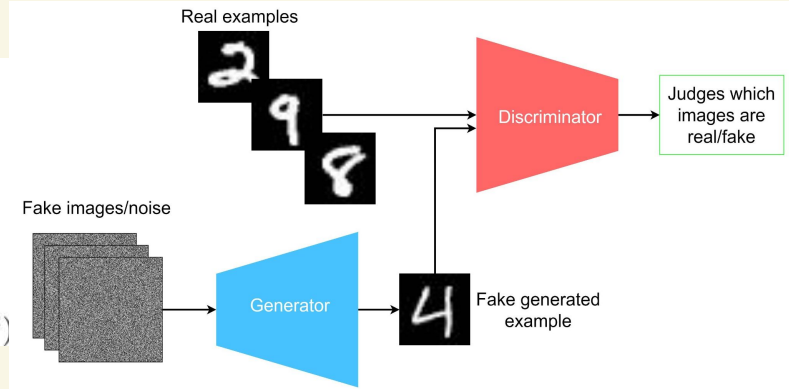
Пусть у нас есть выборка $X_1, \dots, X_n \sim p_{data}$

Наша цель научиться получать новые сэмплы, то есть $X \sim p_{data}$

Как решаем?

- VAE (Variational AutoeEncoder 2013г., https://arxiv.org/abs/1312.6114)
- GAN (Generative Adversarial Networks 2014г., https://arxiv.org/abs/1406.2661)





Какие проблемы?

Генеративная трилемма:

1) Качественная генерация
2) Разнообразная генерация
3) Быстрая генерация

Вопрос:

Чего не хватает VAE? Чего не хватает GAN?

О чем рассказ?

Сегодня мы начнем с дискретных диффузионных моделей (Denoising Diffusion Probabilistic Models https://arxiv.org/abs/2006.11239). Посмотрим какие результаты и параметризации существуют

Дальше воспользуемся непрерывными диффузионными моделями для другого взгляда (Score-Based Generative Modeling through Stochastic Differential Equations https://arxiv.org/abs/2011.13456)

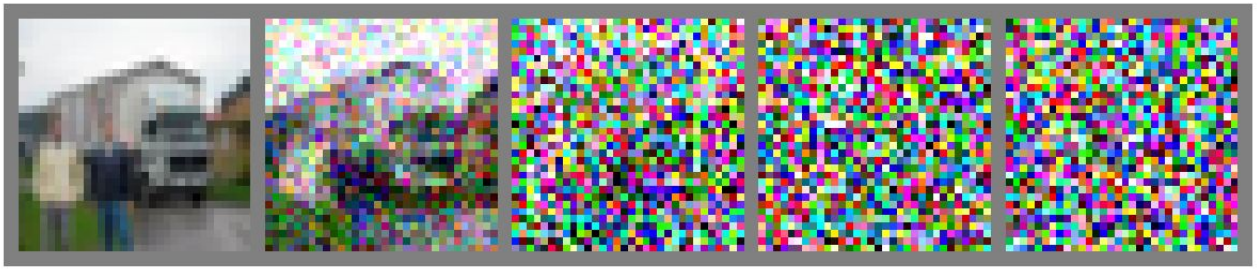
Диффузионные модели

Прежде, чем научиться создавать изображения, нужно уметь их разрушать. Процесс разрушения должен быть простым и приводить к полному отсутствию информации, например, в шум. Далее мы могли бы придумать как найти “обратный” процесс и получить красивые картинки

Вопрос:

Какие простые процессы вы знаете?

(подсказка: у каких процессов нет памяти, то есть переход зависит от настоящего?)



Процесс разрушения (идея)

Мы хотим придумать марковский процесс, который гарантированно за T шагов приведет нас в стандартное нормальное распределение.
Обычно считают, что T = 1000
Требования:

- Марковский
- Достаточно простой
- X_T из стандартного нормального распределения

Для марковских процессов достаточно задать переходные вероятности, пусть они будут нормальными распределениями с какими-либо параметрами. Так будет проще с выводом формул (байесовские методы ура ура ура) и это разумно если хотим прийти в нормальное

Процесс разрушения (формулы)

В основных работах используют 2 процесса зашумления: VP и VE. В первом случае дисперсии сохраняются, во втором случае дисперсии взрываются. Далее будем считать $x_0 \sim p_{data}$, а все переходные вероятности через $q(x_t | x_{t-1})$ или $q(x_t | x_0)$

variance preserving (VP)

$$q(x_t \mid x_{t-1}) = \mathcal{N}(x_{t-1} \mid \sqrt{\alpha_t} x_{t-1}, (1-\alpha_t) I)$$
$$q(x_t \mid x_0) = \mathcal{N}(x_0 \mid \sqrt{\bar{\alpha}_t} x_0, (1-\bar{\alpha}_t) I)$$
$$\bar{\alpha}_t \coloneqq \prod_{i=1}^{t} \alpha_i$$

variance exploding (VE)

$$q(x_t \mid x_{t-1}) = \mathcal{N}(x_{t-1} \mid x_0, (\sigma_t^2 - \sigma_{t-1}^2) I)$$
$$q(x_t \mid x_0) = \mathcal{N}(x_0 \mid x_0, (\sigma_t^2 - \sigma_0^2) I)$$

Вопрос:

Что нам дает шаг из X_0 в X_T ?

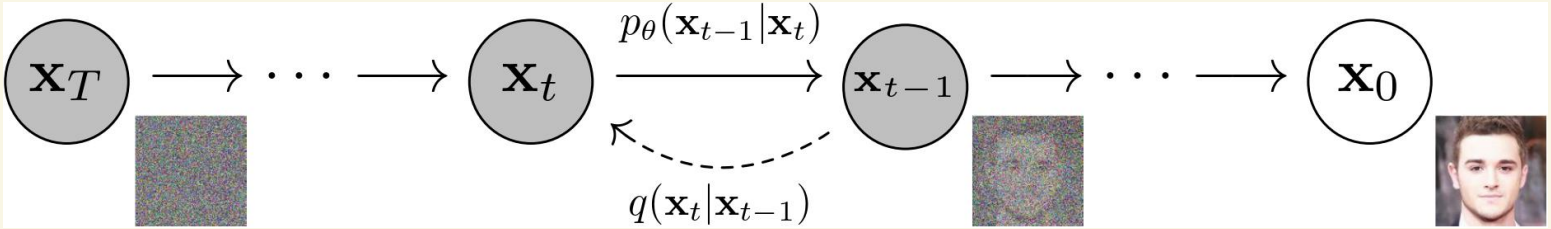
Основные статьи: https://arxiv.org/abs/2006.11239, https://arxiv.org/pdf/2011.13456

Замечание

Дальше мы будем говорить только про вариант VP, а почему вы узнаете в следующей серии)))

Процесс генерации

Мы все начали ради генерации, каким будет процесс генерации?
Как его стоит задать?



Для начала попробуем развернуть процесс разрушения (зашумления)

Мы знаем:

$q(x_t \mid x_{t-1}), q(x_t \mid x_0)$

Хотим найти:

$q(x_{t-1} \mid x_t, x_0)-?$

Процесс генерации

Вопрос:
Что мы знаем про $q(x_t | x_{t-1}, x_0)$?

Процесс генерации

Вопрос:
Что мы знаем про $q(x_t | x_{t-1}, x_0)$?

Ответ:
Вообще говоря мы знаем всё, достаточно использовать теорему Байеса :)

$$q(x_{t-1} \mid x_t, x_0) = \mathcal{N}(x_{t-1} \mid \tilde{\mu}_t(x_t, x_0), \tilde{\sigma}_t I)$$

$$\tilde{\mu}_t(x_t, x_0) := \frac{\sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)}{1-\bar{\alpha}_t} x_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t} x_t, \quad \tilde{\sigma}_t := \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}(1-\alpha_t)$$

Вопрос:
Если мы все знаем, то почему три лекции про понятную модель? Где тут нейронные сети?

Как учим и какой смысл?

В нейронках нам достаточно задать лосс от которого можно взять градиент, а об остальном не стоит думать. Определим лосс так, чтобы процесс генерации был близок с процессом разрушения:

$$\mathbb{E}_q\left[-\log\frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\right]=\mathbb{E}_q\left[-\log p(\mathbf{x}_T)-\sum_{t\geq 1}\log\frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_t|\mathbf{x}_{t-1})}\right]=:L$$

Чем-то похоже на KL дивергенции, перепишем L таким образом:

$$\mathbb{E}_q\Bigg[\underbrace{D_{\mathrm{KL}}(q(\mathbf{x}_T|\mathbf{x}_0)\parallel p(\mathbf{x}_T))}_{L_T}+\sum_{t>1}\underbrace{D_{\mathrm{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)\parallel p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t))}_{L_{t-1}}\underbrace{-\log p_\theta(\mathbf{x}_0|\mathbf{x}_1)}_{L_0}\Bigg]$$

$$\mathbb{E}\left[-\log p_\theta(\mathbf{x}_0)\right]\leq\mathbb{E}_q\left[-\log\frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\right]$$

Как и что учим?

Вообще говоря, L_T – константа, L_0 – можем считать за ошибку квантизации (так как датасет из изображений, а это целые числа). То есть достаточно учить сумму KL дивергенций, ура! осталось понять что учить… Мы же не умеем учить функции распределения…

$$\mathbb{E}_q \Bigg[\underbrace{D_{\mathrm{KL}}(q(\mathbf{x}_T|\mathbf{x}_0) \parallel p(\mathbf{x}_T))}_{L_T} + \sum_{t>1} \underbrace{D_{\mathrm{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) \parallel p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t))}_{L_{t-1}} \underbrace{- \log p_\theta(\mathbf{x}_0|\mathbf{x}_1)}_{L_0} \Bigg]$$

Как и что учим?

Вообще говоря, L_T – константа, L_0 – можем считать за ошибку квантизации (так как датасет из изображений, а это целые числа). То есть достаточно учить сумму KL дивергенций, ура! осталось понять что учить… Мы же не умеем учить функции распределения…

Нам достаточно учить параметры нормального распределения (так как именно его мы и пытаемся приблизить)

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t))$$

$$L_{t-1} = \frac{1}{2\sigma_t^2}\|\mu_\theta(x_t, t) - \tilde{\mu}(x_t, x_0, t)\|^2 + \text{const}$$

$$\mathbb{E}_q\left[\underbrace{D_{\mathrm{KL}}(q(\mathbf{x}_T|\mathbf{x}_0) \parallel p(\mathbf{x}_T))}_{L_T} + \sum_{t>1}\underbrace{D_{\mathrm{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) \parallel p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t))}_{L_{t-1}}\underbrace{- \log p_\theta(\mathbf{x}_0|\mathbf{x}_1)}_{L_0}\right]$$

А точно ли мы то учим?

- Мы не знаем x_0 во время генерации, тогда может стоит учить именно его а не математическое ожидание?
- Вообще говоря мы знаем x_t и если бы знали, каким шумом был зашумлен объект, то смогли бы восстановить x_0

Получаем три параметризации:

- предсказывать шум
- предсказывать чистый объект (денойзер)
- предсказывать математическое ожидание

Спойлер:
Хоть все постановки эквивалентные, но все либо предсказывают шум во время обучения, либо в поздних работах – чистый объект

Параметризация шума

Функция потерь:

$$\mathbb{E}_{\mathbf{x}_0,\boldsymbol{\epsilon}}\left[\frac{\beta_t^2}{2\sigma_t^2\alpha_t(1-\bar{\alpha}_t)}\left\|\boldsymbol{\epsilon}-\boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0+\sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon},t)\right\|^2\right]$$

Функция потерь на практике:

$$L_{\text{simple}}(\theta) := \mathbb{E}_{t,\mathbf{x}_0,\boldsymbol{\epsilon}}\left[\left\|\boldsymbol{\epsilon}-\boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0+\sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon},t)\right\|^2\right]$$

Обучение и генерация

Algorithm 1 Training

1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \text{Uniform}(\{1, \dots, T\})$
4: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: Take gradient descent step on
$$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2$$
6: **until** converged

Algorithm 2 Sampling

1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: **for** $t = T, \dots, 1$ **do**
3: $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
4: $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$
5: **end for**
6: **return** $\mathbf{x}_0$

Как измерить качество?

К сожалению измерить расстояние между распределениями в общем случае очень трудно, но между многомерными нормальными распределениями посчитать можно!

- KL и иные дивергенции
- Расстояние Фреше

$$d_F(\mathcal{N}(\mu, \Sigma), \mathcal{N}(\mu', \Sigma'))^2 = \|\mu - \mu'\|_2^2 + \mathrm{tr}\left(\Sigma + \Sigma' - 2(\Sigma\Sigma')^{\frac{1}{2}}\right)$$

Fréchet inception distance (FID)

Как же получить нормальное распределение для картинок? Можем взять эмбединги с последних слоев хорошего классификатора, например, с архитектурой inception.

Усреднить все эмбединги и поверить в силу ЦПТ! (обычно берут 50 тысяч изображений) Получим какие-то многомерные нормальные распределения, далее посчитаем расстояние Фреше.

Такая метрика называется FID, то есть расстояние Фреше в латентном пространстве хорошо обученного классификатора.

Чем меньше FID, тем лучше

Результаты (дискретные модели)

Иногда стоит отбросить что-то сложное, возможно, выйдут результаты лучше!

Objective	IS	FID
$\tilde{\mu}$ prediction (baseline)		
L, learned diagonal Σ	7.28 ± 0.10	23.69
L, fixed isotropic Σ	8.06 ± 0.09	13.22
$\|\tilde{\mu} - \tilde{\mu}_\theta\|^2$	–	–
ϵ prediction (ours)		
L, learned diagonal Σ	–	–
L, fixed isotropic Σ	7.67 ± 0.13	13.51
$\|\tilde{\epsilon} - \epsilon_\theta\|^2$ (L_{simple})	$\mathbf{9.46 \pm 0.11}$	**3.17**

Промежуточные итоги

Мы умеем:

1) Строить процесс разрушения
2) Знаем на что должен быть похож процесс генерации
3) Можем задать 3 разные параметризации процесса генерации
4) Знаем разные лоссы и как учить
5) Знаем метрику качества

Что не умеем:

1) Генерировать быстро, сейчас мы делаем T шагов, а T=1000 в оригинальной статье. Как менять число шагов не переучивая нейронную сеть – не понятно
2) Как запустить условную генерацию?
3) А что там с латентным пространством?

Ответы

Ответы на почти все эти вопросы были даны в оригинальной статье (https://arxiv.org/abs/2006.11239), но мы не будем думать об этом, так как хотим изучить непрерывное время. Возможно, там некоторые вещи будут проще или более понятны... хотелось бы верить...

Любители дискретных моделей могут обратиться к статье Denoising Diffusion Implicit Models (https://arxiv.org/abs/2010.02502)

Непрерывное время

Часто бывает, что непрерывные модели проще анализировать и оптимизировать, чем дискретные. Кроме того, добавляя непрерывность мы можем найти много интересных свойств.

Часто мы не знаем чему равна сумма ряда, но посчитать аналогичный интеграл бывает сильно проще. Возможно, свести все к интегралам получится и тут!

Непрерывное время (VP)

Вспомним оригинальную модель:

$$t \in \{0, \dots, T-1\}, dt = 1$$
$$q(x_t \mid x_{t-1}) = \mathcal{N}(x_{t-1} \mid \sqrt{\alpha_t} x_{t-1}, (1-\alpha_t) I)$$
$$\text{Пусть } \beta(t) \coloneqq 1 - \alpha_t = At + B$$
$$q(x_t \mid x_{t-1}) = \mathcal{N}(x_{t-1} \mid \sqrt{1-\beta(t)} x_{t-1}, \beta(t) I)$$
$$x_t = \sqrt{1-\beta(t)} x_{t-1} + \sqrt{\beta(t)} z_t, z_t \sim \mathcal{N}(z_t \mid 0, I)$$

Попробуем, представить что число шагов стремиться к бесконечности и чуть поменяем запись:

$$t \in \{\frac{0}{T}, \dots, \frac{T-1}{T}\}, dt = \frac{1}{T}$$
$$\tilde{\beta}(t) = T\beta(t)$$
$$x_t = \sqrt{1-\beta_t} x_{t-1} + \sqrt{\beta_t} z_t$$
$$x_t = \sqrt{1 - \frac{\tilde{\beta}(t)}{T}} x_{t-1} + \sqrt{\frac{\tilde{\beta}(t)}{T}} z_t$$
$$x_t = \sqrt{1 - \tilde{\beta}(t) dt} x_{t-1} + \sqrt{\tilde{\beta}(t) dt} z_t$$
$$\text{Пусть } T \to \infty, \text{ тогда } dt \to 0$$
$$x_t \approx (1 - \frac{1}{2}\tilde{\beta}(t)) dt x_{t-1} + \sqrt{\tilde{\beta}(t)} \sqrt{dt} z_t$$

Что получим?

Перейдем к непрерывному представлению, а именно к дифференциальному уравнению, которое называют VP-SDE

$$x_t \approx (1 - \frac{1}{2}\tilde{\beta}(t))dtx_{t-1} + \sqrt{\tilde{\beta}(t)}\sqrt{dt}z_t$$

$$x_t - x_{t-1} \approx -\frac{1}{2}\tilde{\beta}(t)dtx_{t-1} + \sqrt{\tilde{\beta}(t)}\sqrt{dt}z_t$$

$$dx = -\frac{1}{2}\tilde{\beta}(t)x(t)dt + \sqrt{\tilde{\beta}(t)}dW_t$$

Вопрос:

Что такое dW_t?

Стохастические дифференциальные уравнения

Если бы динамика была бы детерминированная (без случайного шума), то получилось бы обыкновенное дифференциальное уравнение (ОДУ/ODE).
Если же динамика стохастическая, то ее описывает аппарат стохастических дифференциальных уравнений (СДУ/SDE).
Желающие могут изучить подробнее в книге Оксендаль Б. “Стохастические дифференциальные уравнения”. Часто СДУ возникают в финансовой математике и используют достаточно много знаний теории вероятности и уравнений в частных производных. На этом курсе, мы будем пользоваться СДУ как удобным представлением.

$$dx = f(x,t)dt + g(t)dW_t$$

W_t — винеровский процесс

Процесс зашумления

Пусть процесс зашумления (прямой процесс) задается через СДУ:

$$dx = f(x,t)dt + g(t)dW_t$$

W_t — винеровский процесс

При достаточно простых функциях мы можем найти аналитическое решение, а x(0) чистый объект, то есть мы можем найти x(t) за константное время (радостно)

Но наша цель задать x(1) в виде шума, решить СДУ (хоть как-то) и получить изображение. Что же решать?

Обратный СДУ

Если обращать во времени ОДУ мы умеем (заменили начальное условие и заменили dt на -dt), то обращать СДУ математика не умеет...

К счастью, нам и не надо обращать СДУ!

Вопрос:
Что мы хотим от обратного СДУ?

Обратный СДУ

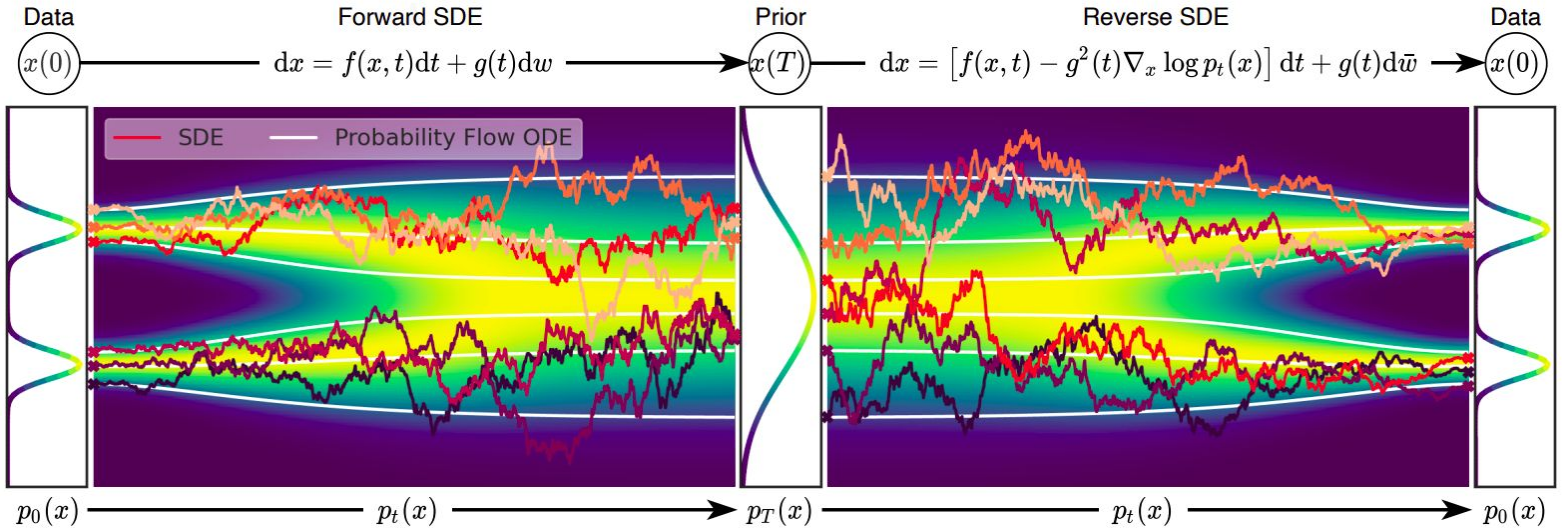
Заметим, что решение любого СДУ – случайная величина с распределением, которое зависит от параметров СДУ.

Пусть $x(t)$ решение прямого СДУ

Наша цель:
Найти такое СДУ, с решением $y(t)$, чтобы $p(y(t)) = p(x(1 - t))$

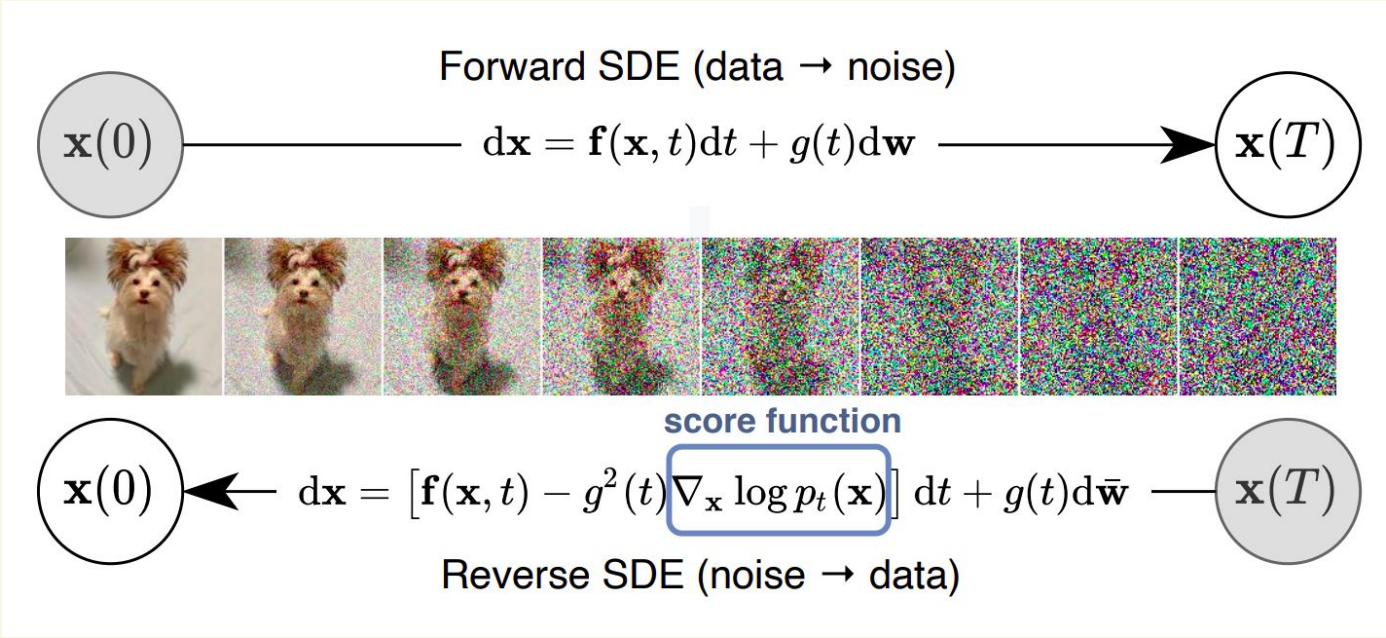
Обратный СДУ

Оказывается, есть статья 1982 года (Reverse-time diffusion equation models) с небольшим использованием уравнений в частных производных, которая дает аналитический вид обратного СДУ



Как выглядит на практике?

Для генерации нам нужно решить СДУ, поэтому на практике мы запускаем численные схемы (например, метод Эйлера).



В чем проблема?

На практике мы не знаем чему равна score функция, значит будем приближать ее нейронной сетью.

Вопрос:
Что мы хотим приблизить? Какой лосс надо минимизировать?...

В чем проблема?

На практике мы не знаем чему равна score функция, значит будем приближать ее нейронной сетью.

Вопрос:
Что мы хотим приблизить? Какой лосс надо минимизировать?...

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \mathbb{E}_t \Big\{ \lambda(t) \mathbb{E}_{\mathbf{x}(0)} \mathbb{E}_{\mathbf{x}(t)|\mathbf{x}(0)} \big[\big\| \mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}(t), t) - \nabla_{\mathbf{x}(t)} \log p_{0t}(\mathbf{x}(t) \mid \mathbf{x}(0)) \big\|_2^2 \big] \Big\}$$

$$p_{0t}(\mathbf{x}(t) \mid \mathbf{x}(0)) = \begin{cases} \mathcal{N}\big(\mathbf{x}(t); \mathbf{x}(0), [\sigma^2(t) - \sigma^2(0)]\mathbf{I}\big), & \text{(VE SDE)} \\ \mathcal{N}\big(\mathbf{x}(t); \mathbf{x}(0) e^{-\frac{1}{2}\int_0^t \beta(s)\mathrm{d}s}, \mathbf{I} - \mathbf{I} e^{-\int_0^t \beta(s)\mathrm{d}s}\big) & \text{(VP SDE)} \\ \mathcal{N}\big(\mathbf{x}(t); \mathbf{x}(0) e^{-\frac{1}{2}\int_0^t \beta(s)\mathrm{d}s}, [1 - e^{-\int_0^t \beta(s)\mathrm{d}s}]^2 \mathbf{I}\big) & \text{(sub-VP SDE)} \end{cases}$$

Что-то похожее

Рассмотрим VP-SDE

$$p_{0t}(x(t) \mid x(0)) := \mathcal{N}(x(t) \mid x(0)\sqrt{\bar{\alpha}_t}, (1-\bar{\alpha}_t)I)$$

$$\log p_{0t}(x(t) \mid x(0)) = \text{const} - \frac{1}{2}\frac{(x(t) - \sqrt{\bar{\alpha}_t}x(0))^2}{1-\bar{\alpha}_t}$$

$$\nabla_x \log p_{0t}(x(t) \mid x(0)) = -\frac{x(t) - \sqrt{\bar{\alpha}_t}x(0)}{1-\bar{\alpha}_t}$$

$$\nabla_x \log p_{0t}(x(t) \mid x(0)) = -\frac{\epsilon(x(t), x(0))}{\sqrt{1-\bar{\alpha}_t}}$$

То есть обучение score функции эквивалентно обучению на шум.
Мы можем учиться на шум, а потом пересчитать score функцию

Результаты

Почему качество стало лучше?

- Взяли лучше лосс
- Лучше обучили эмбеденги, раньше учили на дискретные величины, теперь на непрерывные

Table 3: CIFAR-10 sample quality.

Model	FID↓	IS↑
Conditional		
BigGAN (Brock et al., 2018)	14.73	9.22
StyleGAN2-ADA (Karras et al., 2020a)	**2.42**	**10.14**
Unconditional		
StyleGAN2-ADA (Karras et al., 2020a)	2.92	9.83
NCSN (Song & Ermon, 2019)	25.32	$8.87 \pm .12$
NCSNv2 (Song & Ermon, 2020)	10.87	$8.40 \pm .07$
DDPM (Ho et al., 2020)	3.17	$9.46 \pm .11$
DDPM++	2.78	9.64
DDPM++ cont. (VP)	2.55	9.58
DDPM++ cont. (sub-VP)	2.61	9.56
DDPM++ cont. (deep, VP)	2.41	9.68
DDPM++ cont. (deep, sub-VP)	2.41	9.57
NCSN++	2.45	9.73
NCSN++ cont. (VE)	2.38	9.83
NCSN++ cont. (deep, VE)	**2.20**	**9.89**

Что получили нового?

1) Раньше мы знали про VE и VP, но теперь нам достаточно задать СДУ и теория сразу дает обратный СДУ. Меняя f(x, t) и g(t) мы можем рассмотреть сотни разных СДУ (какие имеют смысл можно вывести из теории)
2) Чтобы в генерации сделать меньше шагов, достаточно увеличить dt в методе Эйлера
3) Можем брать не метод Эйлера, а что-то более сложное, но как?

Немного про ОДУ

Оказывается, для каждого СДУ есть ОДУ, который дает такое же распределения для случайной величины, что заданное СДУ.

Мы можем вместо случайных шагов, заложить всю случайность в начальное условие

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

$$\mathrm{d}\mathbf{x} = \left[\mathbf{f}(\mathbf{x}, t) - \frac{1}{2} g(t)^2 \nabla_{\mathbf{x}} \log p_t(\mathbf{x})\right]\mathrm{d}t$$

Как решать ОДУ?

При решении ОДУ, чтобы обойти метод Эйлера не обязательно залезать в сложную математику. Нам будет достаточно применить метод Рунге–Кутта (о более умных подходах поговорим на следующих парах)

Table 2: NLLs and FIDs (ODE) on CIFAR-10.

Model	NLL Test ↓	FID ↓
RealNVP (Dinh et al., 2016)	3.49	-
iResNet (Behrmann et al., 2019)	3.45	-
Glow (Kingma & Dhariwal, 2018)	3.35	-
MintNet (Song et al., 2019b)	3.32	-
Residual Flow (Chen et al., 2019)	3.28	46.37
FFJORD (Grathwohl et al., 2018)	3.40	-
Flow++ (Ho et al., 2019)	3.29	-
DDPM (L) (Ho et al., 2020)	$\leqslant 3.70^{*}$	13.51
DDPM (L_{simple}) (Ho et al., 2020)	$\leqslant 3.75^{*}$	3.17
DDPM	3.28	3.37
DDPM cont. (VP)	3.21	3.69
DDPM cont. (sub-VP)	3.05	3.56
DDPM++ cont. (VP)	3.16	3.93
DDPM++ cont. (sub-VP)	3.02	3.16
DDPM++ cont. (deep, VP)	3.13	3.08
DDPM++ cont. (deep, sub-VP)	**2.99**	**2.92**

Условная генерация

Раньше мы хотели прийти в $p(x)$ и использовали соответствующую score функцию. Теперь мы хотим сделать условную генерацию, то есть прийти в $p(x \mid y)$. К сожалению найти или вычислить даже score функцию такого распределения очень трудно...

Вопрос:
Что делать?

Условная генерация (classifier guidance)

Раньше мы хотели прийти в p(x) и использовали соответствующую score функцию. Теперь мы хотим сделать условную генерацию, то есть прийти в p(x | y). К сожалению найти или вычислить даже score функцию такого распределения очень трудно...

Воспользуемся теоремой Байеса и вспомним, что p(y | x) легко считается, через обучение шумного классификатора

$$p(x \mid y) = \frac{p(y \mid x)p(x)}{p(y)}$$

$$\log p(x \mid y) = \log p(y \mid x) + \log p(x) + \text{const}$$

$$\nabla_x \log p(x \mid y) = \nabla_x \log p(y \mid x) + \nabla_x \log p(x)$$

Classifier guidance на практике

К сожалению, как всегда бывает в машинном обучении, на практике добавляется гипер параметр. Чем больше ɣ, тем более явной получим условную генерации

$$\nabla_x \log p(x \mid y) = \nabla_x \log p(y \mid x) + \nabla_x \log p(x)$$

$$(1+\gamma)\nabla_x \log p(y \mid x) + \nabla_x \log p(x)$$

От classifier guidance к classifier free guidance (CFG)

Обучать шумный классификатор бывает сложно и накладывает лишнии расходы, а это всегда печально... Попробуем избавить от p(y | x) с помощью теоремы байеса еще раз

$$
\begin{aligned}
&(1+\gamma)\nabla_x \log p(y \mid x) + \nabla_x \log p(x) = \\
&= \gamma \nabla_x \log p(y \mid x) + \nabla_x \log p(x \mid y) = \\
&= \nabla_x \log p(x \mid y) + \gamma(\nabla_x \log p(x \mid y) - \nabla_x \log p(x))
\end{aligned}
$$

Если раньше мы использовали левую часть с классификатором, то теперь будем использовать правую.

CFG на практике

Будем учить нейросеть, подавая метки класса для объекта который пытаемся зашумить (если зашумили 1 из MNIST, то подаем 1). Для обучения безусловной score функции создадим dummy переменную и будем использовать любые объекты.

$$\nabla_x \log p(x \mid y) + \gamma(\nabla_x \log p(x \mid y) - \nabla_x \log p(x)) =$$
$$= \nabla_x \log p(x \mid y) + \gamma(\nabla_x \log p(x \mid y) - \nabla_x \log p(x \mid \emptyset))$$

Именно CFG используют сейчас везде, особенно для text2img генерации, y – текстовый промт, а dummy класс – пустая строка

CFG примеры

Итоги

Мы знаем:

1) Как обучить диффузию в непрерывном времени
2) Как свести ОДУ к СДУ и наоборот
3) Как можно через численные методы легко решать СДУ и ОДУ
4) Как задать условную генерацию через обучение шумного классификатора

Мы **не знаем**:

1) Как решать ОДУ/СДУ учитывая условия задачи?
2) Как же правильно задавать лосс?
3) Как выбрать f(x,t) и g(t)?
4) Как не сломать себе голову, когда читаешь статьи и увидеть единый формализм