Московский государственный университет
имени М. В. Ломоносова
Факультет вычислительной математики и кибернетики

Отчёт по заданию для поступления на кафедру ММП

«Диффузионные модели»

Подготовил:
студент 205 группы
Морозов К. О.

Москва
2025

Оглавление

Постановка задачи

Задание заключалось в том, чтобы разобраться с теоретической и практической составляющими диффузионных моделей. Для этого было предложено ознакомиться со следующими статьями:

- Deep Unsupervised Learning using Nonequilibrium Thermodynamics
- Denoising Diffusion Probabilistic Models
- Generative Modeling by Estimating Gradients of the Data Distribution

В практической части была реализована диффузионная модель для генерации изображений из датасета $MNIST$.

Диффузионные модели и их связь с физикой

Прежде чем переходить к описанию алгоритмов диффузионных моделей, стоит понять, а какую задачу мы пытаемся решить.

Пусть у нас есть набор данных $\mathbb{X} = X_1, X_2, ..., X_n$ из какого-то многомерного пространства Ω с плотностью распределения $q(x)$. Мы хотим научиться создавать объекты, максимально похожие на объекты из $\mathbb{X}$. Эта задача эквивалентна созданию объектов из распределения $q(x)$.

Таким образом, мы можем ввести понятие генеративного обучения, или генеративного моделирования — направления в машинном обучении, суть которого заключается в поиске алгоритмов, занимающихся генерацией данных, схожих с теми, на которых была обучена модель.

В чём заключается проблема? А в том, что зачастую распределение данных нам не известно. То есть всё упирается в поиск того самого распределения $q(x)$, либо его приближении.

Теперь попробуем представить модель одного из алгоритмов генеративного обучения, основываясь на физическом понятии диффузии.

Рассмотрим простой эксперимент: в стакан с водой добавим несколько капель чернил. Что происходит дальше? Чернила постепенно распределяются по всему объёму воды, пока система не достигнет равновесного состояния. Этот процесс называется диффузией. В физике подобные процессы являются неравновесными, так как система переходит из упорядоченного состояния в хаос.

Вдохновившись идеями неравновесной статической физики, попробуем перенести физическую модель диффузии на нашу задачу генерации данных. Мы будем систематически медленно разрушать структуру распределения данных через итеративный прямой процесс диффузии. Далее же наша цель будет в обучении обратного процесса диффузии для восстановления данных. В конечном итоге, мы сможем подавать произвольный шум на вход модели обратного процесса диффузии и получать на выходе данные, максимально похожие на те, на которых была обучена модель.

Математическая основа

Сформулируем нашу задачу из прошлой главы через математическую основу диффузионных моделей по статье «Deep Unsupervised Learning using Nonequilibrium Thermodynamics».

Пусть у нас есть данные $x^{(0)}$ из распределения $q(x^{(0)})$. Давайте преобразуем распределение в хорошо определённое $\pi(y)$, которое можно легко вычислить, используя марковское ядро диффузии $T_\pi(y|y', \beta)$ (в статье рассматриваются случаи, когда марковское ядро совпадает с гауссовским или биномиальным распределением):

$$\pi(y) = \int T_\pi(y|y', \beta) \cdot \pi(y') dy', \;\; q(x^{(t)}|x^{(t-1)}) = T_\pi(x^{(t)}|x^{(t-1)}, \beta)$$

Теперь заметим важное свойство: так как полученная после T шагов алгоритма последовательность данных $x^{(0)}, x^{(1)}, ..., x^{(T)}$ представляет собой цепь Маркова — последовательность случайных событий, таких что вероятность наступления очередного события зависит только от состояния, достигнутого в предыдущем событии, то, используя определение условной вероятности, получаем:

$$q(x^{0:T}) = q(x^{(0)}) \cdot \prod q(x^{(t)}|x^{(t-1)})$$

Таким образом, мы провели прямой процесс диффузии, так что можем переходить к обратному.

Теперь наша цель – построить распределение $\rho(x)$, относящееся к нашей последовательности. Мы знаем, что данные $x^{(T)}$ имеют распределение $\pi(x^{(T)})$ по построению. Так что теперь мы можем применить аналогичную формулу для условных вероятностей, только для распределения $\rho(x)$:

$$\rho(x^{0:T}) = \pi(x^{(T)}) \cdot \prod \rho(x^{(t-1)}|x^{(t)})$$

Далее заметим важный момент: если значение β достаточно мало, то $x^{(t-1)}$ мало отличается от $x^{(t)}$. Поэтому мы можем сказать, что распределение $\rho(x^{(t-1)}|x^{(t)})$ будет иметь тот же вид, что и распределение $q(x^{(t)}|x^{(t-1)})$ в случае гауссова или биномиального распределения второго, что полностью соответствует нашим условиям. Таким образом, нам необходимо оценить среднее $f_\mu(x^{(t)}, t)$ и ковариацию $f_\Sigma(x^{(t)}, t)$ в случае гауссова распределения, или вероятность переключения бита $f_b(x^{(t)}, t)$ в случае биномиального распределения.

Мы готовы к формулировке задачи с точки зрения математической статистики. Мы имеем реальное распределение начальных данных $q(x^{(0)})$ и хотим приблизить его распределением $\rho(x^{(0)})$. Оно вычисляется по формуле, используя выведенные выше факты, а также совместное распределение данных $x^{0:T}$:

$$\rho(x^{(0)}) = \int \rho(x^{0:T}) \cdot dx^{1:T} = \int \rho(x^{(T)}) \cdot q(x^{1:T}|x^{(0)}) \cdot \prod \frac{\rho(x^{(t-1)}|x^{(t)})}{q(x^{(t)}|x^{(t-1)})} \cdot dx^{1:T}$$

Другими словами, мы хотим достигнуть максимума правдоподобия распределения $\rho(x^{(0)})$, или же, максимизировать логарифм правдоподобия этого распределения:

$$\mathbb{L} = \mathbb{E}_q \log \rho(x^{(0)}) = \int q(x^{(0)}) \cdot \log \rho(x^{(0)}) dx^{(0)}$$

Оценим эту величину снизу. Для начала, подставим формулу для $\rho(x^{(0)})$:

$$\mathbb{L} = \int q(x^{(0)}) \cdot \log \left(\int \rho(x^{(T)}) \cdot q(x^{1:T}|x^{(0)}) \cdot \prod \frac{\rho(x^{(t-1)}|x^{(t)})}{q(x^{(t)}|x^{(t-1)})} \cdot dx^{1:T} \right) dx^{(0)}$$

Далее пользуемся неравенством Йенсена: для вогнутой функции $f(x)$ справедливо $f(\mathbb{E}X) \geq \mathbb{E}f(X)$. В нашем функция $\mathbb{E}_{q(x^{1:T}|x^{(0)})} \left(\rho(x^{(T)}) \cdot \prod \frac{\rho(x^{(t-1)}|x^{(t)})}{q(x^{(t)}|x^{(t-1)})} \right)$ находится внутри логарифма, являющегося вогнутой функцией, поэтому $\mathbb{L}$ можно оценить снизу:

$$\mathbb{L} \geq \mathbb{K} = \int q(x^{(0:T)}) \cdot \log \left(\rho(x^{(T)}) \cdot \prod \frac{\rho(x^{(t-1)}|x^{(t)})}{q(x^{(t)}|x^{(t-1)})} \right) dx^{(0:T)}$$

Разобьём K на 2 слагаемых, используя формулу логарифма произведения:

$$\mathbb{K} = \int q(x^{(0:T)}) \cdot \sum_{t=1}^{T} \log \left(\frac{\rho(x^{(t-1)}|x^{(t)})}{q(x^{(t)}|x^{(t-1)})} \right) dx^{(0:T)} + \int q(x^{(T)}) \cdot \log(\rho(x^{(T)})) dx^{(T)}$$

Вспомним, что такое энтропия. Энтропия в математической статистике и теории вероятности - мера неопределённости распределения. Она определяется формулой $H_\rho(\xi) = -\int \rho_\xi(x) \cdot \log(\rho_\xi(x)) dx$. Далее, по построению, в нашей задаче второе слагаемое равно $\int q(x^{(T)}) \cdot \log(\pi(x^{(T)})) dx^{(T)}$, что является энтропией $-H_\rho(x^{(T)})$.

В первом слагаемом в K вынесем знак суммы из-под интеграла, после чего вынесем ещё одно слагаемое, воспользовавшись заранее зафиксированным фактом, что $\rho(x^{(0)}|x^{(1)}) = q(x^{(1)}|x^{(0)}) \cdot \frac{\pi(x^{(0)})}{\pi(x^{(1)})} = T_\pi(x^{(1)}|x^{(0)}; \beta_1)$:

$$\mathbb{K} = \sum_{t=2}^{T} \int q(x^{(0:T)}) \cdot \log \frac{\rho(x^{(t-1)}|x^{(t)})}{q(x^{(t)}|x^{(t-1)})} dx^{(0:T)} + \int q(x^{(0)}, x^{(1)}) \cdot \log \frac{\pi(x^{(0)})}{\pi(x^{(1)})} dx^{(0)} dx^{(1)} - H_\rho(x^{(T)})$$

Слагаемое в центре равно 0, так как распределения не сильно различаются:

$$K = \sum_{t=2}^{T} \int q(x^{(0:T)}) \cdot \log \frac{\rho(x^{(t-1)}|x^{(t)})}{q(x^{(t)}|x^{(t-1)})} dx^{(0:T)} - H_\rho(x^{(T)})$$

Для упрощения суммы, можем использовать теорему Байеса для смены порядка в условной вероятности:

$$q(x^{(t)}|x^{(t-1)}, x^{(0)}) = \frac{q(x^{(t-1)}|x^{(t)}, x^{(0)}) \cdot q(x^{(t)}|x^{(0)})}{q(x^{(t-1)}|x^{(0)})}$$

.

$$\mathbb{K} = \sum_{t=2}^{T} \int q(x^{(0:T)}) \cdot \log \left(\frac{\rho(x^{(t-1)}|x^{(t)})}{q(x^{(t-1)}|x^{(t)}, x^{(0)})} \cdot \frac{q(x^{(t-1)}|x^{(0)})}{q(x^{(t)}|x^{(0)})} \right) dx^{(0:T)} - H_\rho(x^{(T)})$$

Разобьём логарифм на 2 составляющие по соответствующим дробям и заметим, что от второго логарифма останется после суммирования по t только $\log q(x^{(1)}|x^{(0)}) - \log q(x^{(T)}|x^{(0)})$. Тогда получим:

$$\mathbb{K} = \int q(x^{(0)}, x^{(1)}) \cdot \log q(x^{(1)}|x^{(0)}) dx^{(0)} dx^{(1)} - \int q(x^{(0)}, x^{(T)}) \cdot \log q(x^{(T)}|x^{(0)}) dx^{(0)} dx^{(T)} + \sum_{t=2}^{T} \int q(x^{(0:T)}) \cdot \log \left(\frac{\rho(x^{(t-1)}|x^{(t)})}{q(x^{(t-1)}|x^{(t)}, x^{(0)})} \right) dx^{(0:T)} - H_\rho(x^{(T)})$$

Заметим, что первые 2 слагаемых являются условными энтропиями, а под знаком суммы скрываются математические ожидания KL-дивергенций, поэтому можем прийти к нашей окончательной оценке:

$$\mathbb{K} = \sum_{t=2}^{T} \int q(x^{(0)}, x^{(t)}) \cdot D_{KL} \left(q(x^{(t-1)}|x^{(t)}, x^{(0)}) \,||\, \rho(x^{(t-1)}|x^{(t)}) \right) dx^{(0)} dx^{(t)} + H_q(x^{(T)}|x^{(0)}) - H_q(x^{(1)}|x^{(0)}) - H_\rho(x^{(T)}).$$

Здесь, $\mathbb{D}_{KL}$, или KL-дивергенция, или дивергенция Кульбака-Лейблера, — это мера различия между двумя распределениями, имеющая вид:

$$D_{KL}(q(x) \,||\, \rho(x)) = \int q(x) \cdot \log \frac{q(x)}{\rho(x)} dx$$

Таким образом, мы получили нижнюю оценку логарифма правдоподобия модели $\mathbb{L}$. Важно, что все энтропии и KL-дивергенции могут быть вычислены аналитически. Наша задача в конечном счёте приходит к нахождению распределения $\hat{\rho}(x^{(t-1)}|x^{(t)}) = \arg \max_{\rho(x^{(t-1)}|x^{(t)})} \mathbb{K}$. Данная задача уже решаема с помощью тех же нейронных сетей.

Обучение параметров и умножение распределений

Теперь отметим некоторые преимущества диффузионных моделей из статьи «Deep Unsupervised Learning using Nonequilibrium Thermodynamics» в случаях умножения распределений, а также посмотрим на обучение параметров.

Для начала, авторы статьи предлагают обучать методом градиентного подъёма на $\mathbb{K}$ параметры дисперсий $\beta_{2:T}$ в случае гауссова распределения, а параметр β_1 фиксировать изначально достаточно малым для предотвращения переобучения. Тогда в зависимости выборок из распределения $q(x^{(1:T)}|x^{(0)})$ от $\beta_{1:T}$ будет появляться "замороженный шум". То есть при прямом диффузионном процессе мы будем фиксировать шум, добавленный к исходным данным, чтобы уменьшить трату вычислительного ресурса при градиентном подъёме для обучения параметров β. При этом, в случае биномиального распределения фиксировать шум не получится из-за дискретности состояний, поэтому часто берут $\beta_t = \frac{1}{T-t+1}$.

Важнейшим преимуществом над другими генеративными моделями у диффузионной модели является умножение распределений и вычисление апостериорных вероятностей. Апостериорная вероятность — условная вероятность события при условии, что все апостериорные данные получены из опыта. Примеры задач: шумоподавление или восстановление пропущенных значений. В таких задачах нам надо умножить распре-

деление $\rho(x^{(0)})$ на другое распределение или положительную функцию $r(x^{(0)})$, создавая новое распределение $\tilde{\rho}(x^{(0)})$. В диффузионных моделях распределение $r(x^{(0)})$ можно рассматривать как малое возмущение каждого шага диффузии, либо просто умножать на каждом шаге в алгоритме.

Диффузионные модели: «Denoising Diffusion Probabilistic Models»

Перейдём ко второй статье из списка «Denoising Diffusion Probabilistic Models». Авторы хотят показать, что диффузионные модели способны генерировать высококачественные данные. Постановка задачи, в целом, остаётся та же: строится цепь Маркова, и мы пытаемся параметризовать распределения $\rho_\theta(x)$ обратного диффузионного процесса. Единственное, рассматривается частный случай гауссова распределения для марковского ядра диффузии.

Авторы статьи рассматривают цепь Маркова как входные данные x_0 и латентные переменные $x_{1:T}$. Латентные переменные — это скрытые переменные, которые мы не наблюдаем явно, но они несут важную информацию об устройстве данных. Тогда $\rho_\theta(x_0) = \int \rho_\theta(x_{0:T}) dx_{1:T}$, где совместное распределение $\rho_\theta(x_{0:T})$ также называется обратным процессом, и также, как и в прошлой статье, определяется как цепь Маркова:

$$\rho_\theta(x_{0:T}) = \rho_\theta(x_T) \cdot \prod_{t=1}^{T} \rho_\theta(x_{t-1}|x_t), \;\; \rho_\theta(x_T) = \mathcal{N}(x_T; 0, I),$$

$$\rho_\theta(x_{t-1}|x_t) = \mathcal{N}\left(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t)\right)$$

Прямой процесс диффузии также будем характеризовать распределением $q(x)$. Для цепи Маркова по построению будет верно: $q(x_{1:T}|x_0) = \prod_{t=1}^{T} q(x_t|x_{t-1})$, где каждое условное распределение будем формировать как $q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t} \cdot x_{t-1}, \beta_t I)$. Последняя запись, фактически, эквивалентна постепенному размытию данных x_{t-1} и добавлению к ним гауссова шума:

$$x_t = \sqrt{1-\beta_t} \cdot x_{t-1} + \sqrt{\beta_t} \cdot \epsilon, \text{ где } \epsilon = \mathcal{N}(0, I)$$

Важным свойством прямого процесса диффузии является возможность сэмплировать x_t на шаге t в замкнутой форме:

$$q(x_t|x_0) = \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t} \cdot x_0, (1-\bar{\alpha}_t) \cdot I), \text{ где } \alpha_t = 1 - \beta_t \text{ и } \bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$$

Также в этот раз будем оценивать $\mathbb{E}_q[\log \rho_\theta(x_0)]$, только с противоположным знаком. Соответственно, наша цель будет минимизировать это значение (так как хотим максимизировать правдоподобие $\rho_\theta(x_0)$):

$$\mathbb{E}_{q(x_0)}[-\log \rho_\theta(x_0)] \leq \mathbb{E}_{q(x_{0:T})}[-\log \frac{\rho_\theta(x_{0:T})}{q(x_{1:T}|x_0)}] = \mathbb{L}$$

После схожих с предыдущей статьёй преобразований, получаем:
$\mathbb{L} = \mathbb{E}_q[\mathbb{D}_{KL}(q(x_T|x_0) \,||\, \rho_\theta(x_T)) + \sum\limits_{t>1} \mathbb{D}_{KL}(q(x_{t-1}|x_t, x_0) \,||\, \rho_\theta(x_{t-1}|x_t)) - \log \rho_\theta(x_0|x_1)] = \mathbb{L}_T + \sum\limits_{t>1} \mathbb{L}_{t-1} + \mathbb{L}_0$.

Теперь же мы переходим к минимизации $\mathbb{L}$. Перед этим отметим важное отличие от первой статьи: в статье про «DDPM» авторы предлагают фиксировать дисперсии β_t перед обучением, а не обучать в том числе и эти параметры. Из-за этого, $\mathbb{L}_T$ будет оставаться константой на всём этапе обучения.

Рассмотрим сумму $\sum\limits_{t>1} \mathbb{L}_{t-1}$. Для начала, из теоремы Байеса получим:
$q(x_{t-1}|x_t, x_0) = \mathcal{N}(x_{t-1}; \tilde{\mu}_t(x_t, x_0), \tilde{\beta}_t \cdot I)$, где $\tilde{\mu}_t(x_t, x_0) = \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t} \cdot x_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t} \cdot x_t$ и $\tilde{\beta}_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t} \cdot \beta_t$.

Будем искать распределение $\rho_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t))$. Идея: на каждом шаге выбираем удобную параметризацию функций $\mu_\theta(x_t, t)$ и $\Sigma_\theta(x_t, t)$ и, соответственно, упрощаем каждое слагаемое. Преимуществом является простой вид KL-дивергенции в случае нормальных распределений:
$\mathbb{D}_{KL}(\rho_1 \,||\, \rho_2) = \frac{1}{2}\left(Tr(\Sigma_2^{-1}\Sigma_1) + (\mu_2 - \mu_1)^T\Sigma_2^{-1}(\mu_2 - \mu_1) + \ln\frac{det(\Sigma_2)}{det(\Sigma_1)} - d\right)$, где $\rho_i(x) = \mathcal{N}(x; \mu_i, \Sigma_i)$ и d – размерность обеих плотностей распределения вероятностей.

Нам необходимо оценить $\mathbb{E}_q[\mathbb{D}_{KL}(q(x_{t-1}|x_t, x_0) \,||\, \rho_\theta(x_{t-1}|x_t)]$. Для начала, положим во втором распределении $\Sigma_\theta(x_t, t) = \sigma_t^2 \cdot I$, где $\sigma_t^2 = \tilde{\beta}_t$ (то есть приравняем дисперсии распределений), или $\sigma_t^2 = \beta_t$. Экспериментально оба варианта дали схожие результаты в статье. Тогда каждое слагаемое $\mathbb{L}_{t-1}$ из суммы примет вид:

$$\mathbb{L}_{t-1} = \mathbb{E}_q[\mathbb{D}_{KL}(q(x_{t-1}|x_t, x_0) \,||\, \rho_\theta(x_{t-1}|x_t)] = \mathbb{E}_q[\frac{1}{2\sigma_t^2}\|\tilde{\mu}_t(x_t, x_0) - \mu_\theta(x_t, t)\|^2] + \mathrm{C}$$

Теперь наша цель – минимизация квадрата нормы разности между математическими ожиданиями распределений $q(x_{t-1}|x_t, x_0)$ и $\rho_\theta(x_{t-1}|x_t)$. Пойдём ещё дальше и представим x_t как функцию от параметров x_0 и ϵ: $x_t(x_0, \epsilon) = \sqrt{\bar{\alpha}_t} \cdot x_0 + \sqrt{1-\bar{\alpha}_t} \cdot \epsilon$, где $\epsilon = \mathcal{N}(0, I)$. Тогда, расписывая величину $\tilde{\mu}_t$ получаем:

$$\mathbb{L}_{t-1} = \mathbb{E}_q[\frac{1}{2\sigma_t^2}\|\frac{1}{\sqrt{\alpha_t}}\left(x_t(x_0, \epsilon) - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}} \cdot \epsilon\right) - \mu_\theta(x_t(x_0, \epsilon), t)\|^2] + \mathrm{C}$$

Так как x_t доступен как вход модели, предсказывающей μ_θ, положим $\mu_\theta(x_t, t) = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}} \cdot \epsilon_\theta(x_t, t)\right)$. Подставим это выражение в наши расчёты, получим:

$$\mathbb{L}_{t-1} = \mathbb{E}_q[\frac{\beta_t^2}{2\sigma_t^2\alpha_t(1-\bar{\alpha}_t)}\|\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t} \cdot x_0 + \sqrt{1-\bar{\alpha}_t} \cdot \epsilon, t)\|^2] + \mathrm{C}$$

Осталось оценить слагаемое $\mathbb{L}_0$. Здесь стоит отметить, что перед обучением входные данные нужно нормализировать для стабильности, то есть линейно масштабируем исходные данные на отрезок $[-1; 1]$. Если распределение гауссово, то есть непрерыв-

ное, можем пользоваться $\rho_\theta(x_0|x_1) = \mathcal{N}(x_0; \mu_\theta(x_1, 1), \sigma_1^2 I)$. Но в случае конечного числа возможных вариаций данных, плотность условного распределения $\rho_\theta(x_0|x_1)$ можно будет оценить, поочерёдно пройдя по всем составляющим данных (например, в случаях изображений проходимся по каждому пикселю).

Таким образом, наш алгоритм будет заключаться в обучении модели «предсказывать» добавленный к изначальным данным x_0 шум на шаге t. В процессе обучения мы будем подавать модели картинки с известным добавленным шумом ϵ и шаг t. После получения ϵ_θ будем минимизировать квадрат нормы разности между реально добавленым шумом и предсказанным моделью шумом. Для этого будем запускать стандартный градиентный спуск для оптимизации параметров θ модели. Подобный подход эксперементально схож с обучение модели предсказанию $\tilde{\mu}_t$, но лучше себя проявляется именно при минимизации невзвешенной функции потерь (без коеффициентов вида $\frac{\beta_t^2}{2\sigma_t^2\alpha_t(1-\bar{\alpha}_t)}$, которые могут при обучении сильно уменьшать некоторые слагаемые из функции $\mathbb{L}$):

$$\mathbb{L}_{simple}(\theta) = \mathbb{E}_{t,x_0,\epsilon}[\|\epsilon - \epsilon_\theta(x_t, t)\|^2]$$

Практическая часть

Практическая часть состояла из следующих этапов:

- Обобщение нужных алгоритмов
- Реализация диффузионной модели

Пойдём по порядку и начнём с формулировки алгоритмов. Для начала, заострим своё внимание на прямом процессе. На вход принимаем данные x_0 и шаг t. Далее вычисляем значение $x_t = \sqrt{\bar{\alpha}_t} \cdot x_0 + \sqrt{1 - \bar{\alpha}_t} \cdot \epsilon$, где $\epsilon = \mathcal{N}(x; 0, I)$ и возвращаем из алгоритма найденный x_t и добавленный шум ϵ.

Далее необходимо обучить нейронную сеть, предсказывающую шум $\epsilon_\theta(x_t, t)$: для всех данных вычисляем x_t и ϵ на некотором (произвольном) шаге t, используя прямой процесс. По получившимся x_t и t находим шум $\epsilon_\theta(x_t, t)$. Потом минимизируем величину $\mathbb{L} = \mathbb{E}_q[\|\epsilon - \epsilon_\theta(x_t, t)\|^2]$, то есть MSELoss. Для этого высчитываем градиенты функции $\mathbb{L}$ по параметрам θ и запускаем процесс оптимизации параметров. Повторяем этот процесс несколько эпох. Стоит отметить, что в статье про «Denoising Diffusion Probabilistic Models» предлагается для предсказания шума использовать архитектуру нейронной сети $UNet$. Поэтому в практической части мы будем использовать её реализацию из библиотеку $diffusers$.

Остаётся научиться генерировать новые данные. Этот процесс будет неразрывно связан с обратным диффузионным процессом. Для этого предлагается следующий алгоритм: так как мы знаем, что $\tilde{\mu}_t(x_t, x_0) = \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t} \cdot x_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t} \cdot x_t$, $\sigma_t^2 = \beta_t$ или $\sigma_t^2 = \tilde{\beta}_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t} \cdot \beta_t$ и $q(x_{t-1}|x_t, x_0) = \mathcal{N}(x_{t-1}; \tilde{\mu}_t(x_t, x_0), \tilde{\beta}_t \cdot I)$, можем попробовать выразить x_{t-1} через x_t:

$$x_{t-1} = \frac{x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}} \cdot \epsilon_\theta(x_t, t)}{\sqrt{\alpha_t}} + \sigma_t \cdot z$$

Здесь $z = \mathcal{N}(x; 0, I)$ - случайный шум, причём авторы статьи «Denoising Diffusion Probabilistic Models» предлагают не добавлять этот шум на первом шаге, то есть при $t = 1$, чтобы избежать артефактов, возникающих из стохастичности, на последнем этапе генерации. Именно данные преобразования мы будем делать в обратном процессе.

Таким образом, процесс генерации состоит из изначального получения случайного шума из нормального распределения x_T, далее, используя обратный процесс, находим x_{t-1} при $t = T, T-1, ..., 1$. На последнем шаге получаем данные x_0, которые и будут результатом генерации.

Все вышеперечисленные алгоритмы были реализованы на языке *python* в практической части задания. В качестве датасета был выбран $MNIST$. $JupyterNotebook$ прилагается.

«Generative Modeling by Estimating Gradients of the Data Distribution»

Рассмотрим последнюю статью «Generative Modeling by Estimating Gradients of the Data Distribution». В ней представлен координально другой подход к диффузионным моделям, но при этом эквивалентный рассмотренному выше.

У нас также есть распределение реальных данных $\rho_{data}(x)$. И мы вводим понятие *score* — оценка плотности распределения, определяемая как градиент логарифма этого распределения: $\nabla_x \log \rho_{data}(x)$. Идея заключается в предсказании этой оценки. Если мы научимся это делать, то итерационно будем от белого шума идти в сторону градиента (фактически, градиентный подъём), тем самым на каждом шаге будем получать данные из более близкого к реальному распределения.

Обобщим вышесказанные факты. Для этого поработаем с динамикой Ланджевена — концепцией из физики, которая изначально статистически моделирует молекулярные системы. Но с точки зрения теории вероятности, динамика Ланджевена обобщается следующей формулой:

$$x_t = x_{t-1} + \frac{\epsilon}{2} \nabla_x \log \rho(x_{t-1}) + \sqrt{\epsilon} \cdot z_t$$

Здесь начальные данные x_0 лежат, вообще говоря, в некотором априорном распределении $\pi(x)$, далее имеем score в точке x_{t-1}, и также добавляется шум $z_t = \mathcal{N}(0, I)$ для того, чтобы модель не застревала в локальных максимумах. Ну и $\epsilon > 0$ - некоторый параметр. В конечном итоге, при $\epsilon \to 0$ и $T \to \infty$ получаем генерацию точных выборок. Таким образом, для генерации нам достаточно уметь предсказывать score.

Для предсказания score предлагается обучить модель:

$$s_\theta(x) \approx \nabla_x \log \rho(x)$$

Осталось понять, как мы будем оценивать loss. Для этого минимизируем следующую величину:

$$\frac{1}{2}\mathbb{E}_{\rho_{data}}\|s_\theta(x) - \nabla_x \log \rho_{data}(x)\|^2$$

Если расписать квадрат нормы, то получим 3 слагаемых, среди которых слагаемое с квадратом распределения, которое никак не влияет на s_θ, поэтому получаем:

$$\mathbb{E}_{\rho_{data}}\left(\frac{1}{2}\|s_\theta(x)\|^2 - s_\theta(x)\cdot\nabla_x \log \rho_{data}(x)\right)$$

Разбиваем на 2 слагаемых, и второе слагаемое разбиваем по частям. Перед этим, если выполнены условия регулярности для $\rho_{data}(x)$, то на границах оно будет стремиться к 0, из-за чего получим просто:

$$\mathbb{E}_{\rho_{data}}\left(\frac{1}{2}\|s_\theta(x)\|^2 - tr(\nabla_x s_\theta(x))\right)$$

Проблема теперь в оценке $tr(\nabla_x s_\theta(x))$ — якобиана функции $s_\theta(x)$. Есть 2 решения этой проблемы. Одно из них называется Denoising score matching.

Будем добавлять шум к данным, как в диффузии, которую мы рассматривали ранее. В таком случае, имеем распределение шума $q_\sigma(\tilde{x}|x)$. Тогда оказывается, что мы можем оценивать не градиент логарифма исходных данных, а логирифм распределения шума:

$$\frac{1}{2}\mathbb{E}_{q_\sigma(\tilde{x}|x)\cdot\rho_{data}}\|s_\theta(x) - \nabla_x \log q_\sigma(\tilde{x}|x)\|^2$$

$$s_\theta(x) \approx \nabla_x \log q_\sigma(x)$$

Интересный момент, что если сопоставить распределение из «DDPM» распределению шума, то $q(x_t|x_0) = \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t}\cdot x_0, (1-\bar{\alpha}_t)\cdot I$, или $q(x_t|x_0) = \frac{1}{\sqrt{2\pi}\cdot\sqrt{1-\bar{\alpha}_t}}\cdot \exp -\frac{(x_t - \sqrt{\bar{\alpha}_t}\cdot x_0)^2}{2\cdot(1-\bar{\alpha}_t)}$. Взяв от этого логарифм и продифференцировав по x_t получим:

$$s_\theta(x_t, t) \approx -\frac{x_t - \sqrt{\bar{\alpha}_t}\cdot x_0}{1-\bar{\alpha}_t} = -\frac{\epsilon}{\sqrt{1-\bar{\alpha}_t}}$$

Таким образом, мы видим, что в данном случае предсказание score и шума, добавленного к начальным данным, эквивалентны.

Описанный в статье метод сталкивается с несколькими проблемами. Для начала, мы имеем гипотезу многообразия, что реальные данные лежат в низкоразмерных пространствах, вложенных в высокоразмерные пространства. Таким образом, мы можем вычислять score в окружающем пространстве, но он может быть неопределён в точках x из низкоразмерного многообразия. Также score matching предполагает распределённость данных во всём пространстве, что неверно в случае низкораспределённой природы данных.

Ещё одна проблема связана с предсказание градиента, которая в областях с низкой плотностью распределения даёт плохие результаты в силу отсутствия выборок из этих областей.

Решение описанных проблем заключается в зашумлении данных и аннулированной динамике. Для начала рассмотрим подход Noise Conditional Score Networks (NCSN). Будем зашумлять данные шумом с уровнем σ_i, где σ_i будут образовывать убывающую геометрическую последовательность.

$$q_{\sigma_i}(x) = \int \rho_{data}(t)\mathcal{N}(x; t, \sigma_i^2 \cdot I)dt$$

Наша цель в обучении нейросети: $s_\theta(x, \sigma) \approx \nabla_x q_\sigma(x)$. Оценивать же в таком случае мы будем следующую величину:

$$l(\theta; \sigma) = \frac{1}{2}\mathbb{E}_{\rho_{data}}\mathbb{E}_{\tilde{x}\sim\mathcal{N}(x,\sigma^2\cdot I)}\|s_\theta(\tilde{x}, \sigma) + \frac{\tilde{x} + x}{\sigma^2}\|^2$$

Фактически, мы использовали выведенную ранее оценку в Denoising score matching. Остаётся только пройтись по всем σ_i и усреднить результат, используя балансировочную функцию $\lambda(\sigma)$. Итоговая цель:

$$\mathcal{L}(\theta, \sigma_1, ..., \sigma_L) = \frac{1}{L}\sum_{i=1}^{L}\lambda(\sigma_i) \cdot l(\theta, \sigma_i)$$

Наконец, аннелированная динамика Ланджевена предлагает изначально инициализировать x_0 из равномерного шума и далее проходиться по двойному циклу: по L для расчёта шага $\alpha_i = \epsilon \cdot \frac{\sigma_i^2}{\sigma_L^2}$ и по T для градиентного подъёма по динамике Ланджевена с параметром α_i. Конечный результат в цикле по T будет являться начальным в новой итерации по L и, соответственно, по T. Таким образом, мы сгенерируем данные, обходя изначальные проблемы данного подхода к диффузионным моделям.

Заключение

Таким образом, в ходе изучения диффузионных моделей, были изучены довольно близкие статьи «Deep Unsupervised Learning using Nonequilibrium Thermodynamics» и «Denoising Diffusion Probabilistic Models», далее была рассмотрена статья «Generative Modeling by Estimating Gradients of the Data Distribution», предлагающая новый, но при этом эквивалентный с математической точки зрения взгляд на диффузионные модели. В конечном итоге, был реализован простой пример диффузионной модели для генерации изображений из датасета $MNIST$ на основе статьи «Denoising Diffusion Probabilistic Models».