VOLGOGRAD STATE TECHNICAL UNIVERSITY

Машинное обучение и нейросетевые модели

Лекция 9. Генеративно-состязательные сети

Лектор: Кравченя Павел Дмитриевич

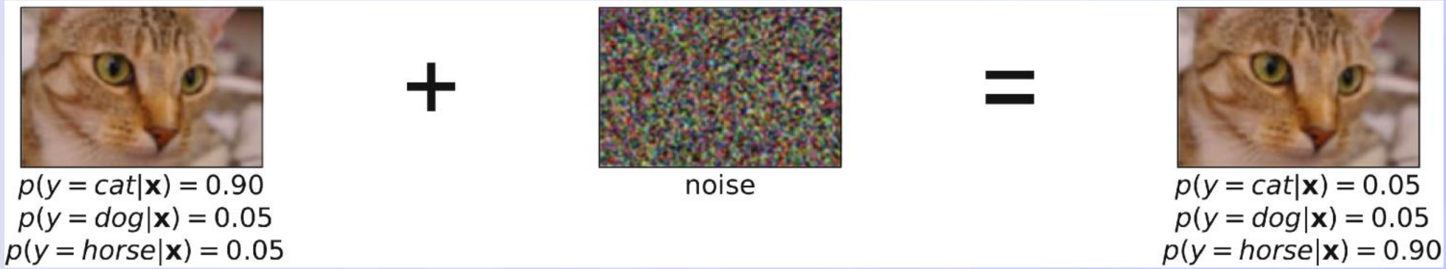
Волгоград 2025

План лекции

1. Понятие дискриминативных и генеративных моделей, их свойства.
2. Классификация глубоких генеративных моделей.
3. Генеративно-состязательные сети. Генеративно-состязательный процесс.
4. Функции ошибки генератора и дискриминатора. Минимаксная игра.
5. Оценка оптимального значения дискриминатора. Обучение генератора.
6. Дивергенция Йенсена-Шеннона.
7. Обучение GAN. Наивный алгоритм и его модификации. Условный GAN.
8. Проблемы при обучении GAN. Метрики качества работы GAN.
9. Манипуляции в латентном пространстве. Обеспечение «распутанности».
10. Практические советы по обучению генеративно-состязательных сетей.

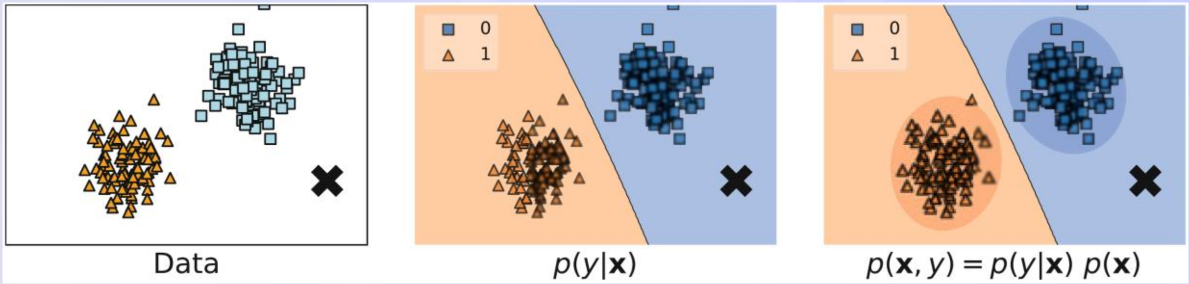
VOLGOGRAD STATE TECHNICAL UNIVERSITY

- Нейронные сети могут демонстрировать *неожиданные результаты*. Известно, например, что добавление шума к изображению может *сдвинуть* предсказанные вероятности классов для ранее обученного классификатора.



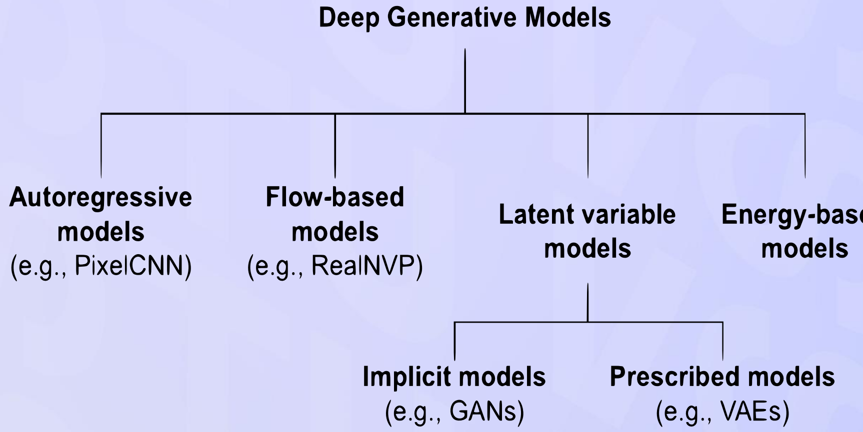
- Подобное поведение говорит о том, что модели не удалось «понять» изображение. Как правило, это справедливо для моделей, которые обучаются предсказывать условное распределение $p(\mathbf{y}\,|\,\mathcal{D})$. Такие модели носят название **дискриминативных** (от *лат. discriminatio* — «*обособление*», «*различение*»).

- Для более *надёжной работы* алгоритмы машинного обучения должны «*понимать*» данные, с которыми они работают. Им требуется не только научиться принимать решения, но и количественно их оценивать с помощью *распределения вероятностей*.
- Для этого модели необходимо выучить совместное распределение вероятностей $p(\boldsymbol{y}, \boldsymbol{\mathcal{D}})$. Такие модели называют **генеративными**. Часто они используются для *генерации новых данных*, однако, могут применяться и для решения *других задач*.



VOLGOGRAD STATE TECHNICAL UNIVERSITY

Классификация и свойства глубоких генеративных моделей

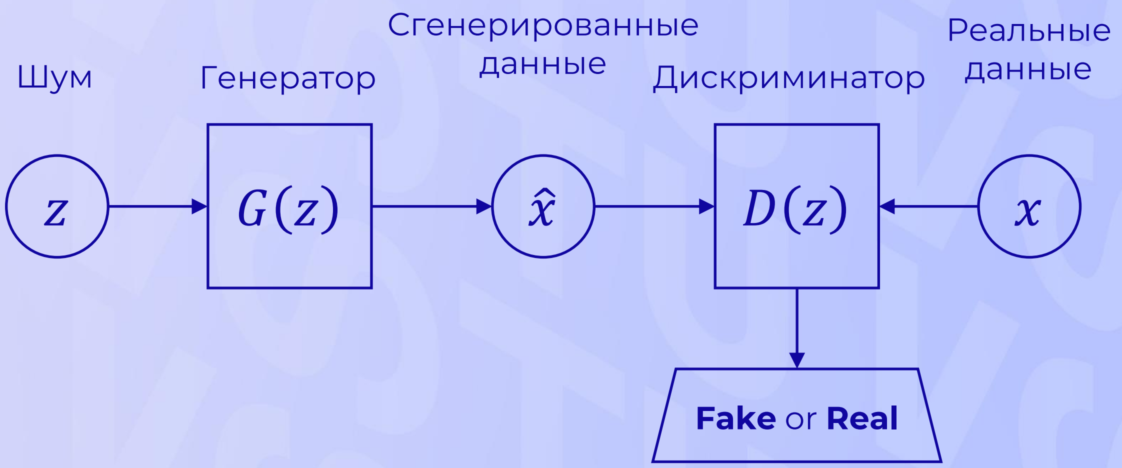


Генеративная модель	Процесс обучения	Оценка правдоподобия	Семплирование	Сжатие информации	Обучение представлений
Autoregressive models	Стабильный	Точная	Медленное	Без потерь	Нет
Flow-based models	Стабильный	Точная	Медленное / быстрое	Без потерь	Да
Implicit models	Нестабильный	Отсутствует	Быстрое	Отсутствует	Нет
Prescribed models	Стабильный	Приближенная	Быстрое	С потерями	Да
Energy-based models	Стабильный	Ненормализованная	Медленное	Скорее отсутствует	Да

- **Генеративно-состязательные сети** (англ. *Generative Adversarial Nets*, *GAN*) представляют собой алгоритм *машинного обучения*, входящий в семейство порождающих моделей и построенный на комбинации двух нейронных сетей:
  - *генеративной модели* G для *генерации «фейковых»* данных, *максимально приближенных* к реальным;
  - *дискриминативной модели* D для *оценки вероятности* того, что образец взят *из обучающих данных*, а не был *сгенерирован* моделью G.
- GAN позволяют определить распределение реальных данных с помощью *состязательной конкуренции* между *генератором* и *дискриминатором*.
- *Впервые* такие сети были предложены Яном Гудфеллоу в 2014 году.

Генеративно-состязательный процесс

- $p_z(z)$ – плотность вероятности *шума* $z \in Z$;
- $p_r(x)$ – плотность вероятности *реальных* данных $x \in X$;
- $p_g(x)$ – плотность вероятности данных $\hat{x} \in \hat{X}$, созданных *генератором*;
- $G_{\gamma_g}:\ Z \to \hat{X}$;
- $D_{\gamma_d}:\ X \times \hat{X} \to \{0, 1\}$.
- В процессе обучения *генератор* старается «*обмануть*» дискриминатор, а *дискриминатор* старается научиться *отличать* сгенерированные данные от реальных.



Бинарная кросс-энтропия

- *Дискриминатор* в *GAN* является *бинарным классификатором*, который определяет принадлежность поданных на его вход данных к *реальным*.
- Функция ошибки бинарного классификатора – бинарная кросс-энтропия:

$$\mathrm{BCE}(y,t) = -t\log y - (1-t)\log(1-y)\,, \qquad y \in [0,1], \qquad t \in \{0,1\}.$$

- Или, для всего множества Y:

$$\mathrm{BCE}(Y,T) = -\frac{1}{m}\sum_{i=1}^{m} t_i \log y_i - \frac{1}{m}\sum_{i=1}^{m}(1-t_i)\log(1-y_i)$$

- Если $y_i \sim p(y)$, то при $m \to \infty$ данная запись *эквивалентна* следующим:

$$\mathrm{BCE}(Y,1) = -\mathbb{E}_{y\sim p(y)}[\log y];$$
$$\mathrm{BCE}(Y,0) = -\mathbb{E}_{y\sim p(y)}[\log(1-y)].$$

- При этом, в процессе *обучения* классификатора стремятся $\mathrm{BCE} \to \min$.

Функции ошибки генератора и дискриминатора

- Нужно, чтобы дискриминатор формировал как можно более *высокую вероятность*, если на его вход подаются *реальные* данные:

$$\mathbb{E}_{x\sim p_r(x)}[\log D(x)] \to \max_D;$$

- и как можно более *низкую*, если эти данные *созданы генератором*:

$$\mathbb{E}_{z\sim p_z(z)}[\log(D(G(z)))] \to \min_D \quad \Longrightarrow \quad \mathbb{E}_{z\sim p_z(z)}\left[\log\left(1 - D\big(G(z)\big)\right)\right] \to \max_D.$$

- С другой стороны, генератор обучается предсказывать *новые данные* так, чтобы *дискриминатор* считал их *реальными* с высокой вероятностью:

$$\mathbb{E}_{z\sim p_z(z)}[\log(D(G(z)))] \to \max_G \quad \Longrightarrow \quad \mathbb{E}_{z\sim p_z(z)}\left[\log\left(1 - D\big(G(z)\big)\right)\right] \to \min_G.$$

- Тогда итоговая функция ошибки соответствует **минимаксной игре**:

$$\min_G \max_D L(D,G) = \min_{\gamma_g} \max_{\gamma_d} \left[\mathbb{E}_{x\sim p_r(x)}[\log D_{\gamma_d}(x)] + \mathbb{E}_{z\sim p_z(z)}\left[\log\left(1 - D_{\gamma_d}\left(G_{\gamma_g}(z)\right)\right)\right]\right].$$

Оценка оптимального значения дискриминатора

- Функцию ошибки для *GAN* можно переписать следующим образом:

$$L(D,G) = \mathbb{E}_{x\sim p_r(x)}\left[\log D_{\gamma_d}(x)\right] + \mathbb{E}_{x\sim p_g(x)}\left[\log\left(1 - D_{\gamma_d}(x)\right)\right].$$

$$L(D,G) = \int_x \left(p_r(x)\cdot\log D_{\gamma_d}(x) + p_g(x)\cdot\log\left(1 - D_{\gamma_d}(x)\right)\right)dx\,.$$

- Попробуем оценить *оптимальное* значение *дискриминатора*, которое соответствует $\max\limits_D L(D,G)$.

- *Максимальное* значение интеграла можно достигнуть, если все слагаемые интегральной суммы *максимальны*:

$$f\big(D(x)\big) = p_r(x)\cdot\log D(x) + p_g(x)\cdot\log\big(1 - D(x)\big) \to \max_{D(x)}.$$

- Для определения *максимума* посчитаем *производную* $f\big(D(x)\big)$ по $D(x)$ и *приравняем* её к нулю.

Оценка оптимального значения дискриминатора

$$\frac{df\big(D(x)\big)}{dD(x)} = p_r(x) \cdot \frac{1}{D(x)} - p_g(x) \cdot \frac{1}{1 - D(x)} = 0;$$

$$\frac{p_r(x)}{D(x)} = \frac{p_g(x)}{1 - D(x)}.$$

- Отсюда можно получить выражение для *оптимального* значения $D(x)$:

$$D^*(x) = \frac{p_r(x)}{p_r(x) + p_g(x)} \in [0, 1].$$

- Можно заметить, что если некоторое значение x является истинным, то $p_r(x) \to 1$, а $p_g(x) \to 0$, и тогда $D(x) \to 1$. А если значение x создано генератором, то $p_r(x) \to 0$, а $p_g(x) \to 1$, и тогда $D(x) \to 0$.
- Если генератор *хорошо обучен*, то $p_g(x) \to p_r(x)$, и тогда $D^*(x) = \frac{1}{2}$.

- Величина:

$$\mathrm{JS}(p||q) = \frac{1}{2} \cdot \mathrm{KL}\left(p \,||\, \frac{p+q}{2}\right) + \frac{1}{2} \cdot \mathrm{KL}\left(q \,||\, \frac{p+q}{2}\right), \qquad \mathrm{JS}(p||q) \in [0, 1]$$

называется дивергенцией **Йенсена-Шеннона**.

- *Дивергенция Йенсена-Шеннона* является еще одной *мерой схожести* двух *распределений* $p(x)$ и $q(x)$.
- Дивергенция $\mathrm{JS}(p||q)$ является *симметричной*: $\mathrm{JS}(p||q) = \mathrm{JS}(q||p)$.
- Дивергенция $\mathrm{JS}(p||q)$ всегда имеет *конечное значение*. *Например*, если при вычислении $\mathrm{JS}(p||q)$ *логарифм* в KL-дивергенции имеет *основание 2*, то:

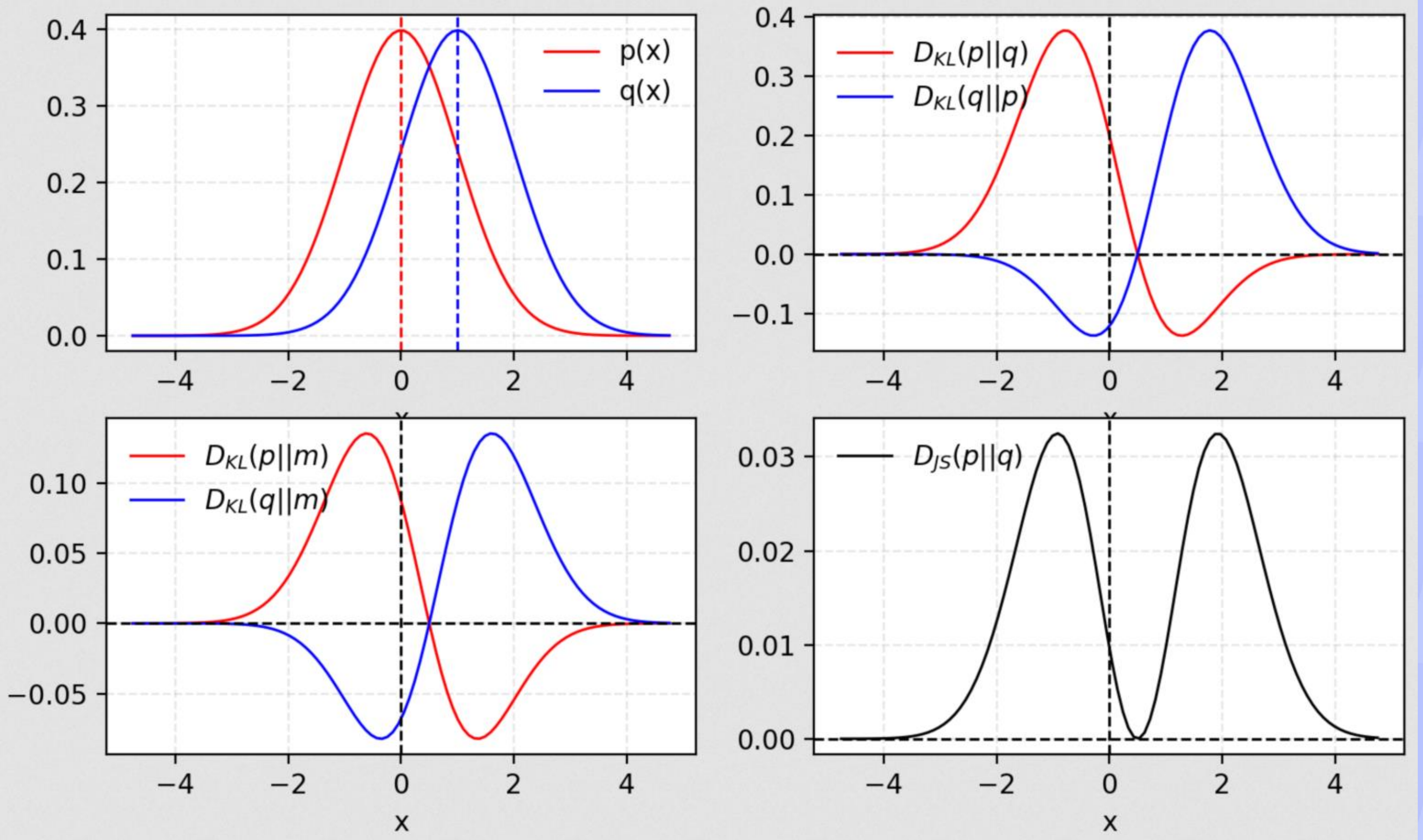
$$0 \le JS(p||q) \le 1.$$

- Чем ближе $\mathrm{JS}(p||q)$ к нулю, тем лучше $q(x)$ аппроксимирует $p(x)$.

VOLGOGRAD STATE TECHNICAL UNIVERSITY

Дивергенция Йенсена-Шеннона

- $p(x) \sim \mathcal{N}(x; 0, 1)$;
- $q(x) \sim \mathcal{N}(x; 1, 1)$;
- $m(x) = \frac{p(x)+q(x)}{2}$;
- $\mathrm{KL}(p||q)$ является несимметричной;
- $\mathrm{JS}(p||q)$ является симметричной;
- *Обе дивергенции* равны нулю, если $p(x) = q(x)$.



- Для обучения *генератора* требуется зафиксировать *дискриминатор*. Пусть *дискриминатор* обучен до своего оптимального значения $D^*(x)$. Тогда:

$$L(D^*, G) = \mathbb{E}_{x\sim p_r(x)}[\log D^*(x)] + \mathbb{E}_{x\sim p_g(z)}\left[\log\left(1 - D^*(x)\right)\right] =$$

$$= \mathbb{E}_{x\sim p_r(x)}\left[\log\frac{p_r(x)}{p_r(x)+p_g(x)}\right] + \mathbb{E}_{x\sim p_g(z)}\left[\log\left(\frac{p_g(x)}{p_r(x)+p_g(x)}\right)\right] =$$

$$= \mathbb{E}_{x\sim p_r(x)}\left[\log p_r(x) - \log\left(p_r(x)+p_g(x)\right)\right] + \mathbb{E}_{x\sim p_g(z)}\left[\log\left(p_g(x)\right) - \log\left(p_r(x)+p_g(x)\right)\right] =$$

$$= \mathbb{E}_{x\sim p_r(x)}\left[\log p_r(x) - \log\frac{p_r(x)+p_g(x)}{2} - \log 2\right] + \mathbb{E}_{x\sim p_g(z)}\left[\log\left(p_g(x)\right) - \log\frac{p_r(x)+p_g(x)}{2} - \log 2\right] =$$

$$= -2\log 2 + \mathbb{E}_{x\sim p_r(x)}\left[\log p_r(x) - \log\frac{p_r(x)+p_g(x)}{2}\right] + \mathbb{E}_{x\sim p_g(z)}\left[\log\left(p_g(x)\right) - \log\frac{p_r(x)+p_g(x)}{2}\right].$$

- В таком случае, рассматриваемую *функцию ошибки GAN* можно записать:

$$L(D^*, G) = -2\log 2 + \mathrm{KL}\left(p_r(x) \,||\, \frac{p_r(x) + p_g(x)}{2}\right) + \mathrm{KL}\left(p_g(x) \,||\, \frac{p_r(x) + p_g(x)}{2}\right).$$

- Или, с использованием *дивергенции Йенсена-Шеннона*:

$$L(D^*, G) = -2\log 2 + 2 \cdot \mathrm{JS}(p_r \,||\, p_g).$$

- При обучении генератора требуется $L(D^*, G) \to \min\limits_G$, что приводит к минимизации дивергенции Йенсена-Шеннона между распределениями *истинных* и *созданных генератором* данных.
- Таким образом, эффективно обученный *генератор* должен очень хорошо *имитировать* реальные данные. Если *сгенерированные* данные *не отличаются* от реальных, тогда $L(D^*, G^*) = -2\log 2$.

Наивный алгоритм обучения GAN

• *Наивный алгоритм* обучения *GAN* предполагает, что в процессе обучения требуется делать *два шага оптимизации поочередно*:

1. Решить задачу максимизации ошибки по γ_d, повторяя следующие шаги до сходимости параметров дискриминатора к оптимальному значению γ_d^*:
   - Составить *мини-батч* семплов шума $\{z_1, z_2, \dots, z_n\}$ из $p_z(z)$.
   - Составить *мини-батч* семплов данных $\{x_1, x_2, \dots, x_n\}$ из $p_r(x)$.
   - Обновить *дискриминатор*, выполнив шаг вверх по его градиенту:

$$\nabla_{\gamma_d} \frac{1}{n} \sum_{i=1}^{n} \left[\log D_{\gamma_d}(x_i) + \log\left(1 - D_{\gamma_d}(G_{\gamma_g}(z_i))\right) \right].$$

Наивный алгоритм обучения GAN

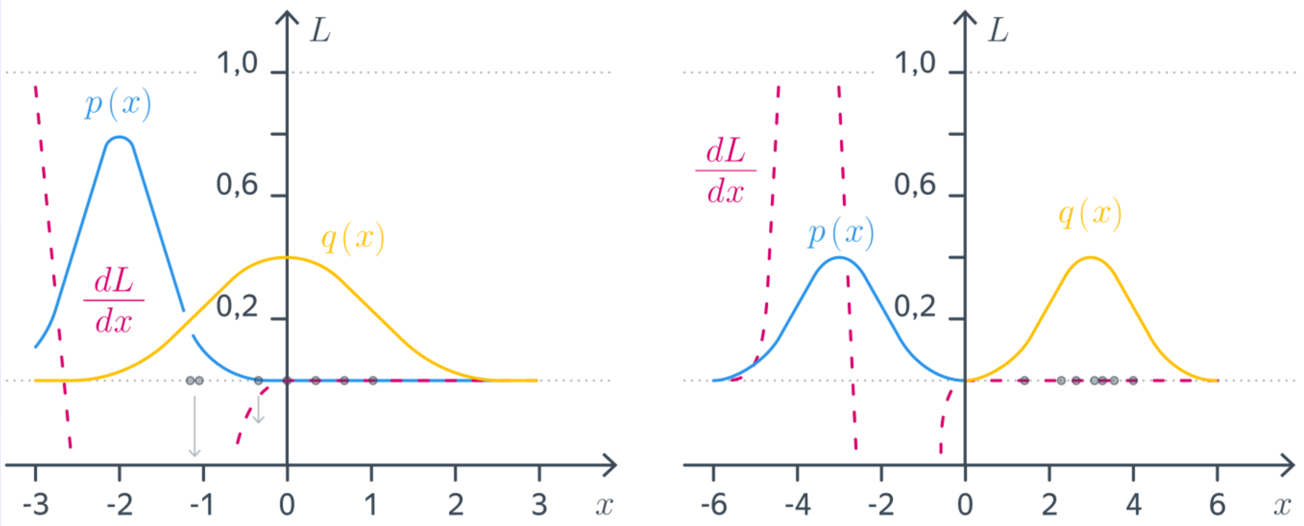
2. Выполнить шаг *стохастического градиентного спуска* для решения задачи минимизации ошибки по параметрам генератора γ_g:
   - Составить *мини-батч* семплов шума $\{z_1, z_2, \dots, z_n\}$ из $p_z(z)$.
   - Обновить *генератор*, выполнив шаг вверх по его градиенту:

$$\nabla_{\gamma_g} \frac{1}{n} \sum_{i=1}^{n} \left[\log\left(1 - D_{\gamma_d^*}(G_{\gamma_g}(z_i))\right) \right] = \frac{1}{n} \sum_{i=1}^{n} - \frac{\nabla_{\gamma_g} D_{\gamma_d^*}(G_{\gamma_g}(z_i))}{1 - D_{\gamma_d^*}(G_{\gamma_g}(z_i))}.$$

- Однако, данный подход к обучению имеет *пару серьёзных проблем*:
  - Он очень *медленный*, потому что необходимо обучать дискриминатор *до сходимости*, чтобы сделать всего один шаг по градиенту генератора.
  - *Функция потерь* генератора может *насыщаться* и выдавать близкие к нулю градиенты.

Проблема насыщения функции потерь генератора

- Визуализируем $p_r(x) = p(x)$ и $p_g(x) = q(x)$, а также <u>*градиент*</u> *по семплам* из генератора.
- В случае, когда пики распределений <u>*плохо пересекаются*</u> друг с другом, градиент будет <u>равен нулю</u> на *большинстве* семплов, которые выдаёт *генератор*.
- Обучение происходит <u>*недостаточно эффективно*</u>: тратится время на вычисление сэмплов, которые <u>не делают никакой вклад</u> в обновление параметров генератора.



Способы решения проблемы насыщения

- Из-за наличия *проблемы насыщения* описанная функция потерь генератора называется «*сатурирующей*».
- Существует *два способа решения* проблемы насыщения:
  - ➢ Обучать дискриминатор на каждой итерации не до сходимости, а с *небольшим* *фиксированным* числом шагов (на практике чаще всего используется $k \leq 2$).
  - ➢ Изменить *функцию ошибки генератора*, преобразовав её таким образом, чтобы её градиент в области *несоответствия* распределений был отличен от нуля. Например, вместо $\log\left(1 - D_{\gamma_d^*}(G_{\gamma_g}(z_i))\right)$ применять $-\log\left(D_{\gamma_d^*}(G_{\gamma_g}(z_i))\right)$, поскольку:

$$\gamma_g^* = \arg\min_{\gamma_d} L(D^*, G) = \arg\min_{\gamma_d} \log\left(1 - D_{\gamma_d^*}(G_{\gamma_g}(z_i))\right) = \arg\min_{\gamma_d} \left[-\log\left(D_{\gamma_d^*}(G_{\gamma_g}(z_i))\right)\right].$$

Метрики качества работы GAN

- Для оценки *качества работы* *GAN* применяют следующие метрики:

1. User study – *экспертная оценка*, в ходе которой эксперт должен *сравнить* сгенерированный и реальный образец и определить фейковый. Опрос экспертов направлен на *оценку реализма* полученных результатов.
2. Frechet Inception Distance – основан на сравнении двух распределений *высокоуровневых* признаков для реальных и сгенерированных объектов. Признаки обычно формируются с выходов *глубоких слоёв* нейросети, обученной на *датасете*, который используется для генерации. При работе с изображениями практически во всех случаях используется модель *Inception v3*, предобученная на данных *ImageNet*.

Для измерения *схожести* между *распределениями* используется метрика Вассерштейна:

$$\mathrm{FID} = |\mu - \hat{\mu}|^2 + \mathrm{Tr}\left(\Sigma + \hat{\Sigma} - 2\sqrt{\Sigma\hat{\Sigma}}\right).$$

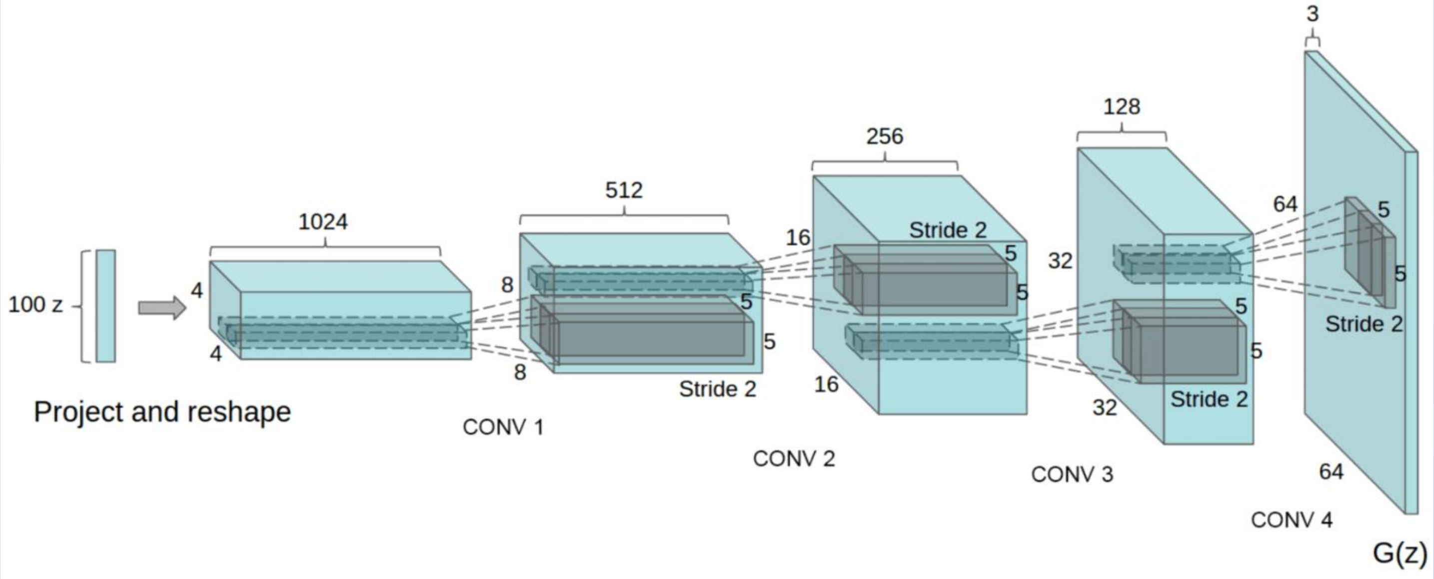
Здесь $\mu \in \mathbb{R}^C$ и $\Sigma \in \mathbb{R}^{C\times C}$ – вектор средних и матрица ковариаций глубоких признаков $\{F_i \in \mathbb{R}^{C\times HW}\}_{i=1}^N$, которые рассчитываются по выборке из N реальных изображений: $F \in \mathbb{R}^{C\times NHW}$. Так же вычисляются и $\hat{\mu}$, $\hat{\Sigma}$ для сгенерированных картинок.

3. Интерполяции в скрытом пространстве – предполагает сравнение сгенерированных объектов для *интерполированных* векторов шума.
4. Поиск ближайших соседей – предполагает визуальную проверку совпадения ближайших соседей из датасета с *созданными семплами*.

- На практике обычно не так часто возникает задача генерации просто *какого-либо* объекта. Чаще требуется сгенерировать *конкретный* объект, удовлетворяющий заданным условиям (как правило, *заданного класса*).
- В таком случае говорят об *условной генерации*. В качестве условия y может выступать любой объект (например, текстовые эмбеддинги).
- Таким образом, задача сводится к *построению генератора*, моделирующего $p_g(x \mid y)$.
- Основной метод *условной генерации* — конкатенация *условия* с *вектором шума*, который *генератор* принимает на вход.
- Также *рекомендуется* подавать *условие* не только в генератор, но и в дискриминатор (Conditional GAN, 2014).

Пример: Deep Convolution GAN

- Наиболее *простая* версия *генеративной модели* для *изображений* — это *Deep Convolution GAN* (*DCGAN*). В её основе лежит *простая идея*: нейросети, основанные на *свёртках*, отлично подходят для *распознавания* изображений, а значит, вполне могут подойти и для их *генерации*.

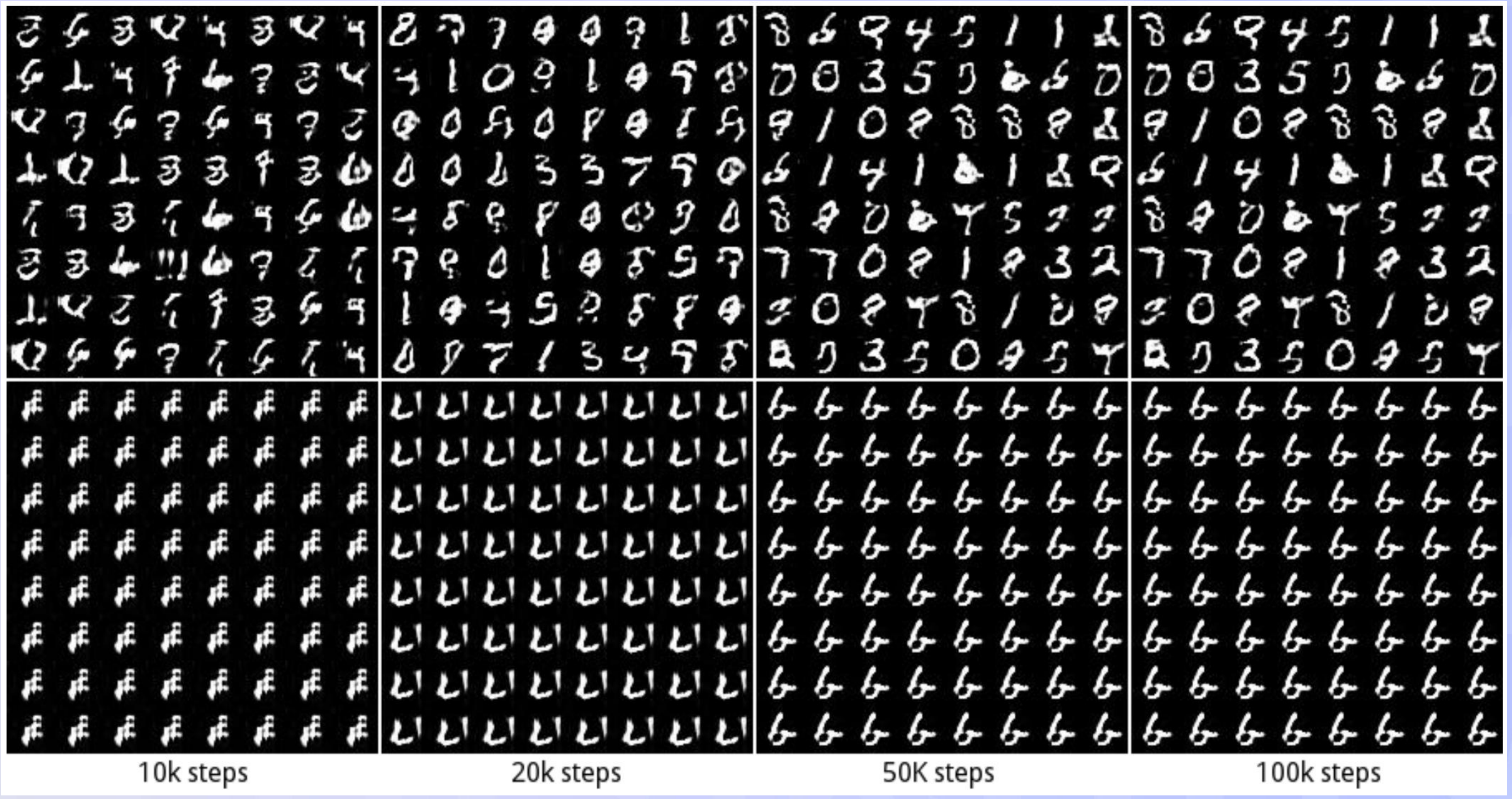


- Большинство *GAN* подвержено возникновению *<u>следующих проблем</u>*:
  - *<u>Коллапс моды распределения</u>* (англ. *mode collapse*): генератор выдает *ограниченное* количество разных *объектов*.
  - *<u>Проблема стабильности обучения</u>* (англ. *non-convergence*): параметры модели *дестабилизируются* и *не сходятся*.
  - *<u>Исчезающий градиент</u>* (англ. *diminished gradient*): *дискриминатор* становится слишком «*сильным*», при этом градиент *генератора* исчезает, и обучение *<u>не происходит</u>*.
  - *<u>Проблема запутывания</u>* (англ. *disentanglement problem*): выявление *корреляции* в признаках, *<u>слабо связанных</u>* с *реальными* факторами.
  - *<u>Высокая чувствительность</u>* GAN к *гиперпараметрам*.

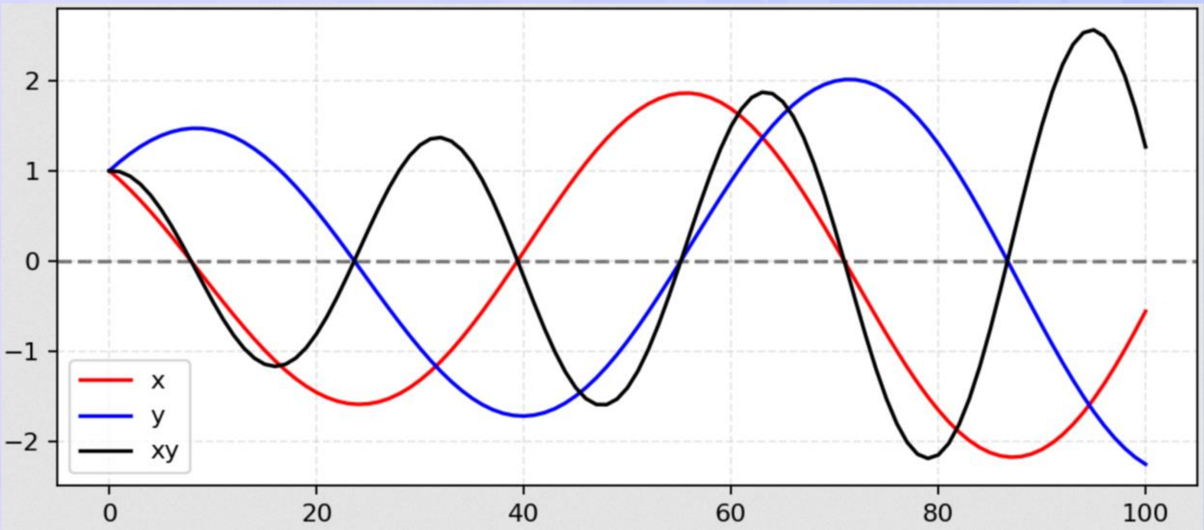
- В процессе обучения _генератор_ может прийти к состоянию, при котором он будет всегда выдавать _ограниченный набор объектов_, существенно _меньший_, чем пространство _реальных объектов_.
- Главная _причина_ коллапса в том, что _генератор_ обучается _обманывать дискриминатор_, а не _воспроизводить_ реальное распределение:
  - Генератор начинает каждый раз выдавать _похожий выход_, который является _максимально правдоподобным_ для дискриминатора.
  - Если дискриминатор _улучшит детектирование_, то в дальнейшем наиболее вероятно, что генератор придет к _другому изображению_, _хорошо обманывающему_ текущий дискриминатор.
- Однако, данный процесс не сходится, а количество мод не увеличивается.

Пример коллапса моды распределения

- Датасет MNIST
- Нижний ряд – обычный GAN
- Верхний ряд – Unrolled GAN с 20 обратными шагами.



- Задача *обучения дискриминатора и генератора* в общем смысле *не является* задачей поиска *локального* или *глобального* экстремума функции, а является задачей поиска *точки равновесия* двух игроков.
- Эта точка называется *точкой равновесия Нэша* (англ. *Nash equilibrium*).
- *Пример*: пусть G старается максимизировать $f(x, y) = xy$, а D – *минимизировать* с помощью *градиентного спуска*.



- $\frac{\partial f}{\partial x} = y,\ x \leftarrow x - \eta \cdot y;$
- $\frac{\partial f}{\partial y} = x,\ y \leftarrow y + \eta \cdot x.$
- Разные *знаки* приращений ведут к *осцилляции* x и y и *нестабильности*.

Интерполяции в латентном пространстве

- В результате обучения GAN получается генератор, который можно рассматривать как функцию, порождающую новый объект: $x = G(z)$. Пространство, в котором располагается z, называется **латентным**.
- Рассмотрим $z_1 \sim p_z(z)$ и $z_2 \sim p_z(z)$. Тогда $x_1 = G(z_1)$ и $x_2 = G(z_2)$ – два объекта, созданные генератором. Все точки на линии, соединяющей z_1 и z_2, будут соответствовать *объектам*. Если *двигаться* по этой линии и использовать точки с неё в качестве входа для генератора, то можно получить *плавно изменяющийся* сгенерированный объект.

Проблема запутывания в GAN. Манипуляции в латентном пространстве

- Неясно, как GAN определяет конкретные характеристики объекта, и связаны ли они между собой.
- Для хорошо обученной сети генератор – это функция $G\colon Z \to X,\ Z \subseteq \mathbb{R}^d$. Пусть существует функция оценки $f_S\colon X \to S$, $S \subseteq \mathbb{R}^m$ – пространство *характеристик* изображения. Тогда связь между точкой в скрытом пространстве (шума) $\mathbf{z}$ и характеристикой изображения $\mathbf{s}$, которому она соответствует, выражается *соотношением*: $\mathbf{s} = f_S(G(\mathbf{z}))$.
- Установлено, что при движении между двумя точками $\mathbf{z}_1$ и $\mathbf{z}_2$ характеристики меняются *плавно*. Тогда по этому направлению в Z можно построить *гиперплоскость*:

$$\{\mathbf{z} \in \mathbb{R}^d : \mathbf{n}^{\mathrm{T}}\mathbf{z} = 0\}.$$

Проблема запутывания в GAN. Манипуляции в латентном пространстве

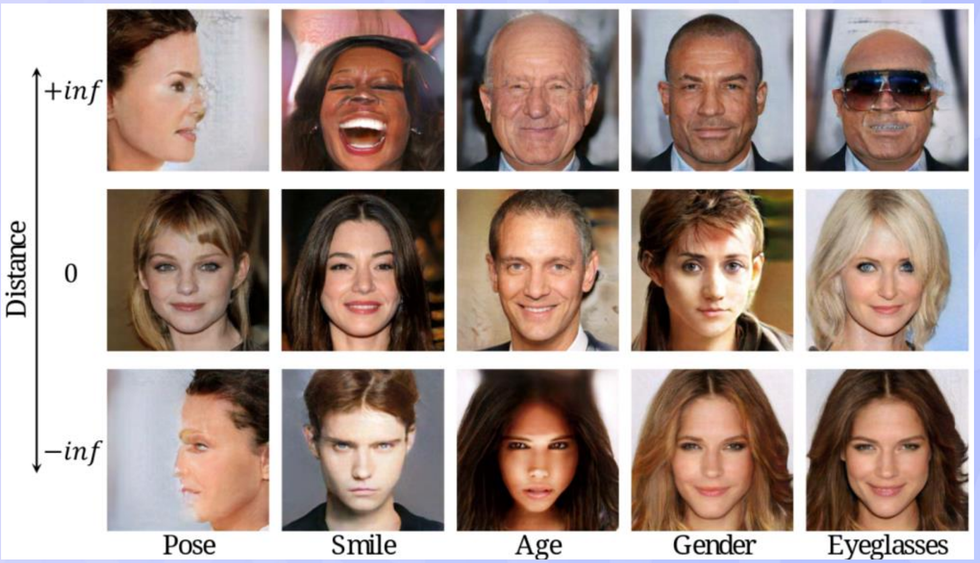
- Сделаем предположение, что для некоторого бинарного параметра существует такая *гиперплоскость*, что все образцы с одной стороны от нее имеют *одинаковое значение* этого параметра.
- Введем функцию «*расстояния*»: $d(\mathbf{n},\mathbf{z}) = \mathbf{n}^{\mathrm{T}}\mathbf{z}$, $\mathbf{n} \in \mathbb{R}^d$ – вектор *нормали* к *гиперплоскости*. Тогда *ожидается*, что:

$$f_S(G(\mathbf{z})) = \lambda d(\mathbf{n},\mathbf{z}).$$

- В случае нескольких параметров можно записать:

$$\mathbf{s} \equiv f_S\big(G(\mathbf{z})\big) = \mathbf{\Lambda}\mathbf{N}^{\mathrm{T}}\mathbf{z}.$$

Здесь $\mathbf{s} = [s_1, s_2, \dots, s_m]^{\mathrm{T}}$, $\mathbf{\Lambda} = \mathrm{diag}(\lambda_1, \lambda_2, \dots, \lambda_m)$, $\mathbf{N} = [\mathbf{n}_1, \mathbf{n}_2, \dots, \mathbf{n}_m]$.



- Если $z \sim \mathcal{N}(\mathbf{0}, \mathbb{I}_d)$, то можно посчитать *матрицы среднего и ковариации* для оценки $\mathbf{s}$:

$$\boldsymbol{\mu}_{\mathbf{s}} = \mathbb{E}[\boldsymbol{\Lambda}\mathbf{N}^{\mathrm{T}}\mathbf{z}] = \boldsymbol{\Lambda}\mathbf{N}^{\mathrm{T}}\mathbb{E}[\mathbf{z}] = \mathbf{0};$$

$$\boldsymbol{\Sigma}_{\mathbf{s}} = \mathbb{E}\left[\boldsymbol{\Lambda}\mathbf{N}^{\mathrm{T}}\mathbf{z}\left(\boldsymbol{\Lambda}\mathbf{N}^{\mathrm{T}}\mathbf{z}\right)^{\mathrm{T}}\right] - \mathbb{E}[\boldsymbol{\Lambda}\mathbf{N}^{\mathrm{T}}\mathbf{z}]\,\mathbb{E}\left[\left(\boldsymbol{\Lambda}\mathbf{N}^{\mathrm{T}}\mathbf{z}\right)^{\mathrm{T}}\right] = \mathbb{E}[\boldsymbol{\Lambda}\mathbf{N}^{\mathrm{T}}\mathbf{z}\mathbf{z}^{\mathrm{T}}\mathbf{N}\boldsymbol{\Lambda}^{\mathrm{T}}] = \boldsymbol{\Lambda}\mathbf{N}^{\mathrm{T}}\mathbb{E}[\mathbf{z}\mathbf{z}^{\mathrm{T}}]\mathbf{N}\boldsymbol{\Lambda}^{\mathrm{T}} = \boldsymbol{\Lambda}\mathbf{N}^{\mathrm{T}}\mathbf{N}\boldsymbol{\Lambda}^{\mathrm{T}}.$$

- Таким образом, получаем, что:

$$\mathbf{s} \sim \mathcal{N}(0, \boldsymbol{\Sigma}_{\mathbf{s}}).$$

- Значения *оценочной функции*, соответствующие различным параметрам, можно считать «*распутанными*» только когда *матрица* $\boldsymbol{\Sigma}_{\mathbf{s}}$ является *диагональной*.
- В этом случае, *вектора* $\{\mathbf{n}_1, \mathbf{n}_2, \dots, \mathbf{n}_m\}$ являются *ортогональными*.

VOLGOGRAD STATE TECHNICAL UNIVERSITY

- Универсального подхода к решению многих проблем при обучении GAN нет. Но существуют практические советы для помощи при обучении:
  - *Нормализация данных* – формирование признаков в диапазоне $[-1, 1]$.
  - *Замена функции ошибки* для генератора с $\min \log(1 - D)$ на $\max \log D$.
  - *Сэмплирование шума* из многомерного *нормального* распределения вместо *равномерного*.
  - *Использование слоёв нормализации* (например, *batch normalization* или *layer normalization*) в *генераторе* и *дискриминаторе*.
  - *Использование меток для данных*, если они *имеются*, в процессе обучения (обучать дискриминатор также *классифицировать образцы*).

Демонстрация практических примеров

VOLGOGRAD STATE TECHNICAL UNIVERSITY

Заключение

1. Дали понятия дискриминативных и генеративных моделей, рассмотрели классификацию и свойства генеративных моделей.
2. Познакомились с генеративно-состязательными сетями и рассмотрели основные принципы их функционирования.
3. Рассмотрели наивный алгоритм обучения GAN и способы его улучшения.
4. Поговорили про основные проблемы, связанные с обучением GAN, и выяснили возможные способы их решения.
5. Рассмотрели манипуляции в латентном пространстве GAN, поговорили о возможностях для генерации, которые они предоставляют.
6. Обговорили основные советы, предлагаемые для обучения GAN, и на практическом примере рассмотрели этапы обучения и генерации GAN.

VOLGOGRAD STATE TECHNICAL UNIVERSITY

Спасибо за внимание!

Волгоград 2025