Машинное обучение (Machine Learning)

Глубокие порождающие модели: соперничающие сети (GAN)

Уткин Л.В.

Санкт-Петербургский политехнический университет Петра Великого

Соперничающие сети Generative adversarial networks (GAN)

Соперничающие сети - Generative adversarial networks (GAN)

Мотивация:

- Порождающие модели в основном обучаются при помощи метода максимума правдоподобия, котоый может быть неприменимым благодаря нормализации.
- Порождающая соперничающая сеть - это метод для обучения порождающих моделей при помощи нейронных сетей, обученных при помощи стохастического градиентного спуска вместо метода максимума правдоподобия.

Предварительное

- Дивергенция KL (Кульбака-Лейблера) измеряет, насколько одно распределение вероятностей p отличается от второго распределения вероятностей q.
- $$D_{KL}(p\|q) = \int_x p(x) \log \frac{p(x)}{q(x)} dx$$
- KL-дивергенция асимметрична.
- Дивергенция JS (Дженсена-Шеннона) - симметрична и более плавная

$$D_{KL}p(x)D_{JS}(p\|q) = \frac{1}{2}D_{KL}(p\|\frac{p+q}{2}) + \frac{1}{2}D_{KL}(q\|\frac{p+q}{2})$$

Соперничающие сети

Две задачи в машинном обучении:

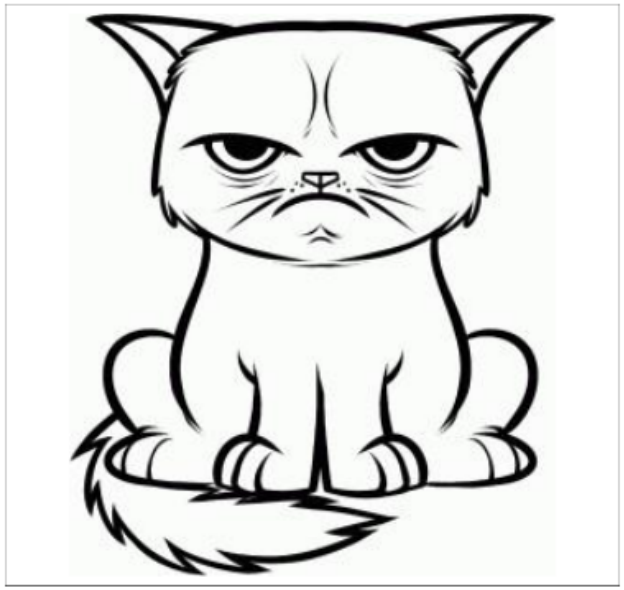
1. дискриминативная (discriminative): классифицирует входные данные
2. порождающая (generative): пытается создать модель, которая может генерировать данные, похожие на обучающие данные

Соперничающие сети

- Две сети:
  1. порождающая (generative): $y_G = G(z)$ - на входе z (шум), на выходе y_G
  2. дискриминативная (discriminative): $y_D = D(x)$ - на входе x (данные), на выходе y_D
- G-сеть должна научиться генерировать такие образцы y_G, что D-сеть D не сможет их отличить от эталонных образцов
- D-сеть должна научиться отличать эти сгенерированные образцы от настоящих
- Игра с нулевой суммой

Данные и шум

x



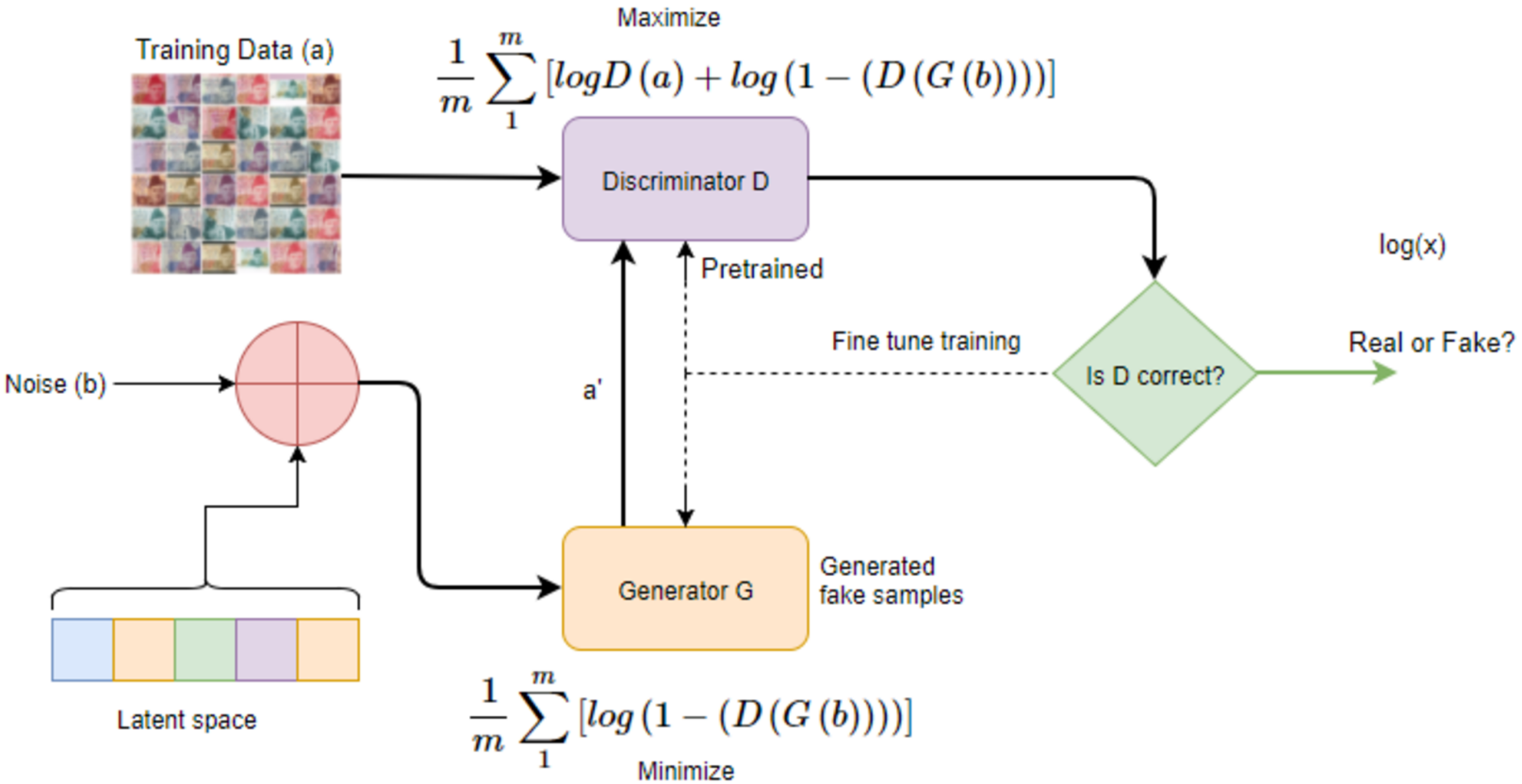
$G(z)$

http://www.lab41.org/lab41-reading-group-generative-adversarial-nets/

Соперничающие сети - две модели

- Две модели: порождающая (фальшивомонетчик) и дискриминативная (банкир)
- Фальшивомонетчик пытается построить на выходе подделку на настоящие деньги, а банкир — отличить подделку от оригинала (обе модели начинают с рандомных условий, и в начале выдают в качестве результатов шум).
- Цель фальшивомонетчика - сделать такой продукт, который банкир не мог бы отличить от настоящего.
- Цель банкира — максимально эффективно отличать подделки от оригиналов.
- Обе модели начинают игру друг против друга, где останется только один.

Соперничающие сети



Соперничающие сети - обучение G-сети

- G-сеть: максимизация функционала $D(G(z))$, т.е. G-сеть максимизирует результат работы D-сети.
- G-сеть должна научиться для любого значения z на входе сгенерировать на выходе такое значение y_G, подав которое на вход D-сети, получим максимальное значение на ее выходе y_D (сделать так, чтобы банкир был уверен, что подделки — настоящие.)

Соперничающие сети

- p_z - распределение вероятностей шума z (обычно неизвестно)
- p_g - распределение вероятностей генератора на данных x
- p_r - распределение вероятностей реальных данных x

Продолжение

- С одной стороны, мы хотим убедиться, что решения дискриминатора D относительно реальных данных точны, максимизируя $\mathbb{E}_{x\sim p_r(x)}[\log D(x)]$. Между тем, для фальшивой выборки $G(z)$, $z \sim p_z(z)$ ожидается, что дискриминатор выдаст вероятность $D(G(z))$, близкую к нулю, максимизируя $\mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))]$.
- С другой стороны, генератор обучен повышать вероятность D создания фальшивого примера с высокой вероятностью, тем самым сводя к минимуму $\mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))]$.

Минимаксная игра

- При объединении обоих аспектов вместе получаем **минимаксную игру**, в которой должны оптимизировать следующую функцию потерь:

$$\begin{aligned}\min_G \max_D L(D,G) &= \mathbb{E}_{x\sim p_r(x)}[\log D(x)] \\ &+ \mathbb{E}_{z\sim p_z(z)}[\log(1-D(G(z)))] \\ &= \mathbb{E}_{x\sim p_r(x)}[\log D(x)] \\ &+ \mathbb{E}_{x\sim p_g(x)}[\log(1-D(x)]\end{aligned}$$

- ($\mathbb{E}_{x\sim p_r(x)}[\log D(x)]$ не влияет на обновления во время градиентного спуска)

Какое D оптимально?

- Есть функция потерь. Что является лучшим значением для D?

$$L(G, D)$$
$$= \int_x (p_r(x) \log(D(x)) + p_g(x) \log(1 - D(x))) \, dx$$

- Нас интересует, какое значение D максимизирует $L(G, D)$ наилучшим образом
- Обозначим $\tilde{x} = D(x), A = p_r(x), B = p_g(x)$

Какое D оптимально?

- И тогда то, что находится внутри интеграла:

$$
\begin{aligned}
f(\tilde{x}) &= A \cdot \log \tilde{x} + B \cdot \log(1 - \tilde{x}) \\
\frac{df(\tilde{x})}{d\tilde{x}} &= A \cdot \frac{1}{\ln 10} \frac{1}{\tilde{x}} - B \cdot \frac{1}{\ln 10} \frac{1}{1 - \tilde{x}} \\
&= \frac{1}{\ln 10} \left(\frac{A}{\tilde{x}} - \frac{B}{1 - \tilde{x}} \right) \\
&= \frac{1}{\ln 10} \cdot \frac{A - (A + B) \cdot \tilde{x}}{\tilde{x} \cdot (1 - \tilde{x})}
\end{aligned}
$$

Какое D оптимально?

- Таким образом, установив

$$\frac{df(\tilde{x})}{d\tilde{x}} = 0,$$

получим наилучшее значение дискриминатора

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

- Как только генератор натренирован до оптимального p_g, становится очень близко к p_r. когда $p_g = p_r$, $D^*(x)$ становится $1/2$.

Глобальный оптимум?

- Когда оба G и D находятся в своих оптимальных значениях, имеем $p_g = p_r$ и $D^*(x) = 1/2$ и функция потерь становится:

$$
\begin{aligned}
L(G, D^*) &= \int_x \left[p_r(x) \log(D^*(x)) + p_g(x) \log(1 - D^*(x)) \right] dx \\
&= \log \frac{1}{2} \int_x p_r(x) dx + \log \frac{1}{2} \int_x p_g(x) dx \\
&= -2 \log 2
\end{aligned}
$$

Что представляет функция потерь?

- JS-дивергенция между p_g и p_r:

$$
\begin{aligned}
D_{JS}(p_r \| p_g) =& \frac{1}{2} D_{KL}\left(p_r || \frac{p_r + p_g}{2}\right) + \frac{1}{2} D_{KL}\left(p_g || \frac{p_r + p_g}{2}\right) \\
=& \frac{1}{2}\left(\log 2 + \int_x p_r(x) \log \frac{p_r(x)}{p_r + p_g(x)} dx\right) + \\
& \frac{1}{2}\left(\log 2 + \int_x p_g(x) \log \frac{p_g(x)}{p_r + p_g(x)} dx\right) \\
=& \frac{1}{2}\left(\log 4 + L(G, D^*)\right)
\end{aligned}
$$

Что представляет функция потерь?

- И $L(G, D^*) = 2D_{JS}(p_r \| p_g) - 2\log 2$
- Функция потерь GAN количественно определяет сходство между генеративным распределением данных и распределением реальной выборки по JS-дивергенции, когда дискриминатор оптимален.
- Лучшее, что воспроизводит реальное распределение данных, приводит к минимуму $L(G^*, D^*) = -2\log 2$, который согласуется с приведенными выше уравнениями

Проблемы: сложно достичь равновесия Нэша

- Две модели одновременно обучаются нахождению равновесия по Нэшу в некооперативной игре с двумя игроками.
- Однако каждая модель обновляет свой лосс независимо от другого игрока в игре. Одновременное обновление градиента обеих моделей не может гарантировать сходимость.
- Рассмотрим пример. Пусть один игрок обновляет xдля минимизации $f_1(x) = xy$, а другой игрок обновляет y для минимизации $f_2(x) = -xy$.
- Так как $\frac{\partial f_1}{\partial x} = y$ и $\frac{\partial f_2}{\partial y} = -x$, обновляем x: $x - \eta y$ и y: $y + \eta x$ одновременно в одной итерации, где η скорость обучения. Как только xи yимеют разные знаки, каждое последующее обновление градиента вызывает огромные колебания, и нестабильность со временем ухудшается.

Проблемы: затухание градиента

- Когда дискриминатор сильный, т.е. $D(x) = 1, \forall x \in p_r$ и $D(x) = 0, \forall x \in p_g$. Функция потерь L падает до нуля, и нет градиента для обновления потерь во время итераций.
- В результате обучение GAN сталкивается с дилеммой:
  - Если дискриминатор ведет себя “слабо”, генератор не имеет точной обратной связи, и функция потерь не может отображать реальность
  - Если дискриминатор отлично справляется, градиент функции потерь падает почти до нуля, и обучение становится очень медленным или даже тормозит.
- Эта дилемма явно способна сделать обучение GAN очень сложным.

Wasserstein GAN (WGAN) (1)

Расстояние Вассерштайна (WD) — это мера расстояния между двумя распределениями вероятностей. Его также называют расстоянием движения Земли, потому что неформально его можно интерпретировать как минимальные затраты энергии на перемещение и преобразование грунта в форме одного распределения вероятностей в форму другого распределения. Стоимость определяется количественно: количество перемещенного грунта $\times$ расстояние перемещения.

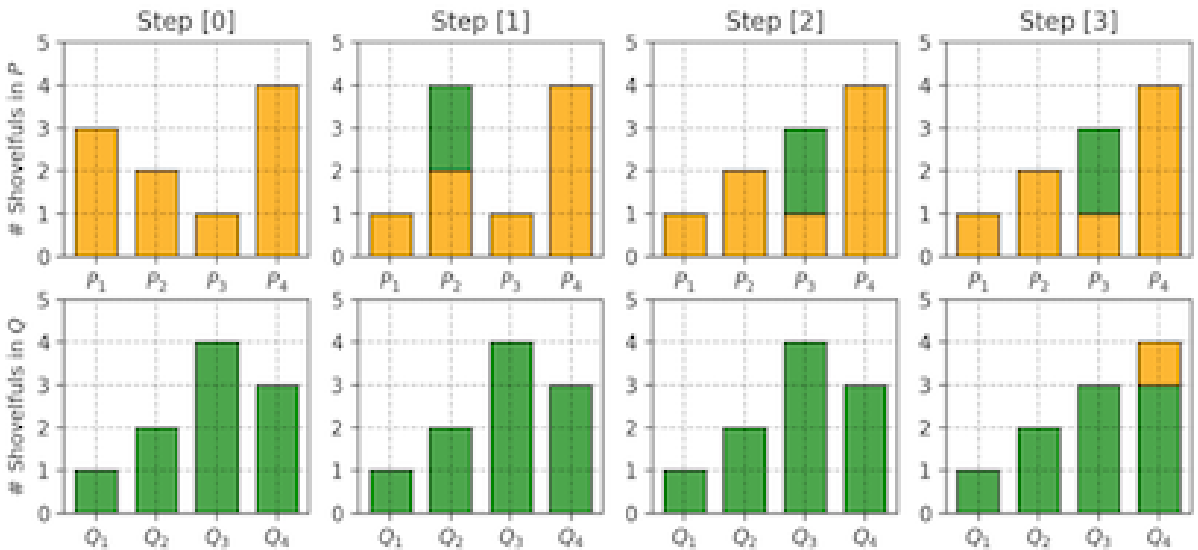
Wasserstein GAN (WGAN) (2)

Рассмотрим простой случай, когда область вероятностей является дискретной. Например, предположим, что у нас есть два распределения P и Q, каждое имеет по 4 кучи грунта, а в сумме у обоих десять лопат грунта. Количество лопат в каждой куче грунта распределяется следующим образом:

$$P_1 = 3, P_2 = 2, P_3 = 1, P_4 = 4,$$
$$Q_1 = 1, Q_2 = 2, Q_3 = 4, Q_4 = 3$$

Wasserstein GAN (WGAN) (3)

- Первый ход 2 лопаты из P_1 в $P_2 => (P_1, Q_1)$ совпадают
- Затем 2 лопаты из P_2 в $P_3 => (P_2, Q_2)$ совпадают
- В завершение 1 лопата из Q_3 в $Q_4 => (P_3, Q_3)$ и (P_4, Q_4) совпадают



Wasserstein GAN (WGAN) (4)

- Если обозначить затраты, чтобы P_i и Q_i совпадали с δ_i, то получим и $\delta_{i+1} = \delta_i + P_i - Q_i$:

$$\delta_0 = 0$$
$$\delta_1 = 0 + 3 - 1 = 2$$
$$\delta_2 = 2 + 2 - 2 = 2$$
$$\delta_3 = 2 + 1 - 4 = -1$$
$$\delta_4 = -1 + 4 - 3 = 0$$

- Наконец, WD: $W = \sum |\delta_i| = 5$

Wasserstein GAN (WGAN) (5)

- При непрерывных распределениях вероятностей WD:

$$W(p_r, p_g) = \inf_{\gamma \sim \Pi(p_r, p_g)} \mathbb{E}_{(x,y)\sim\gamma}[\|x - y\|]$$

- $\Pi(p_r, p_g)$ - множество всех возможных совместных распределений вероятностей между p_r и p_g.
- Одно совместное распределение $\gamma \in \Pi(p_r, p_g)$ описывает один план транспортировки грунта.
- $\gamma(x, y)$ указывает процент грунта, который необходимо транспортировать из точки x в точку y, чтобы получить x, имеющее одно и то же распределение вероятностей y.

Wasserstein GAN (WGAN) (6)

$$W(p_r, p_g) = \inf_{\gamma \sim \Pi(p_r, p_g)} \mathbb{E}_{(x,y)\sim\gamma}[\|x - y\|]$$

- Вот почему маргинальное распределение по x в сумме составляет p_g, $\sum_x \gamma(x, y) = p_g(y)$ (как только закончим перемещение запланированного количества грунта из всех возможных x в целевое y, мы получим именно то, что y имеет согласно p_g) и наоборот $\sum_y \gamma(x, y) = p_r(x)$.

Wasserstein GAN (WGAN) (7)

- При рассмотрении x в качестве отправной точки и y как пункта назначения общее количество перемещенного грунта $= \gamma(x, y)$, а пройденное расстояние $= |x - y|$, и стоимость $= \gamma(x, y) \cdot |x - y|$. Ожидаемая стоимость, усредненная по всем парам (x, y), равна

$$\sum_{x,y} \gamma(x, y)\|x - y\| = \mathbb{E}_{x,y\sim\gamma}\|x - y\|$$

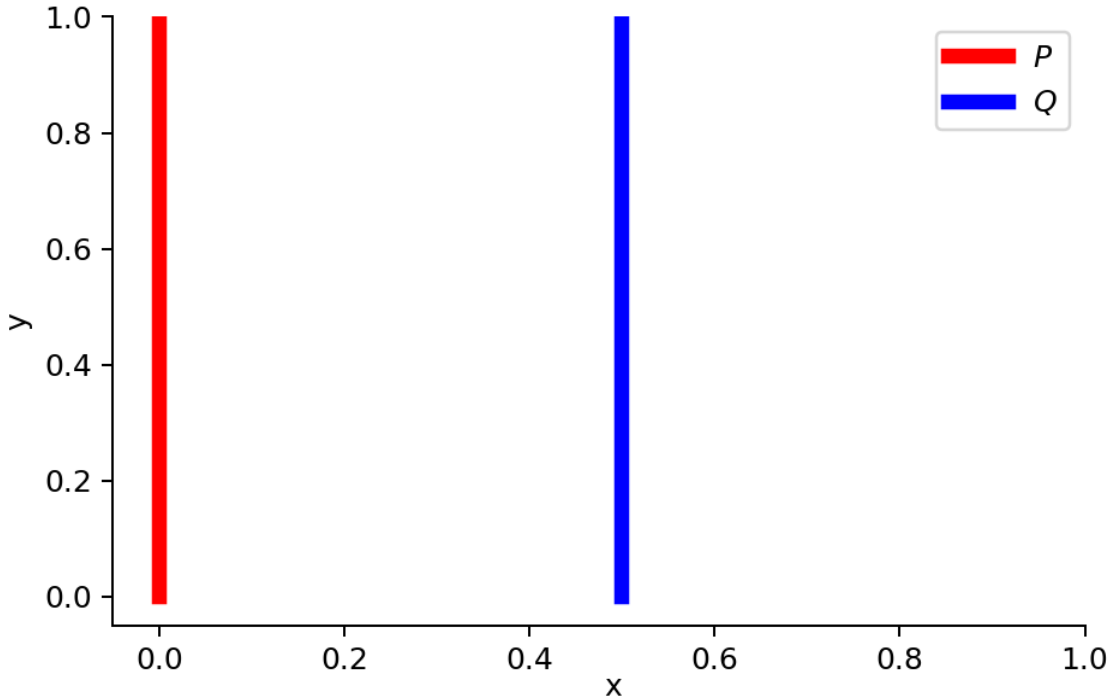
- Наконец, принимаем минимальную стоимость всех решений по перемещению грунта в качестве WD, которое (нижняя граница, также известная как наибольшая нижняя граница) указывается, что нас интересует только наименьшая стоимость.

Почему WD лучше KL и JS (1)

- Даже когда два распределения расположены в многообразиях более низкой размерности без пересечения, WD может обеспечить значимое и гладкое представление расстояния между ними.
- Простой пример: предположим, есть два распределения вероятностей, P и Q:

$$\forall (x, y) \in P,\ x = 0 \text{ и } y \sim U(0, 1)$$
$$\forall (x, y) \in Q,\ x = \theta, 0 \leq \theta \leq 1 \text{ и } y \sim U(0, 1)$$

Почему WD лучше KL и JS (2)



Почему WD лучше KL и JS (3)

Когда $\theta \neq 0$:

$$D_{KL}(P\|Q) = \sum_{x=0, y\sim U(0,1)} 1 \cdot \log \frac{1}{0} = +\infty$$

$$D_{KL}(Q\|P) = \sum_{x=\theta, y\sim U(0,1)} 1 \cdot \log \frac{1}{0} = +\infty$$

$$D_{JS}(P, Q) = \frac{1}{2}\left(\sum_{x=0, y\sim U(0,1)} 1 \cdot \log \frac{1}{1/2} + \sum_{x=0, y\sim U(0,1)} 1 \cdot \log \frac{1}{1/2}\right)$$

$$= \log 2$$

$$W(P, Q) = |\theta|$$

Почему WD лучше KL и JS (4)

- Когда $\theta = 0$, то оба распределения полностью перекрываются:

$$D_{KL}(P\|Q) = D_{KL}(Q\|P) = D_{JS}(P, Q) = 0$$
$$W(P, Q) = 0 = |\theta|$$

- $D_{KL} \to \infty$, когда два распределения не пересекаются. Значение D_{JS} имеет скачок, не дифференцируемый при $\theta = 0$. Только WD обеспечивает плавное измерение, что очень полезно для стабильного процесса обучения с использованием градиентного спуска.

WD в функции потерь GAN

- Невозможно исчерпать все возможные совместные распределения $\Pi(p_r, p_g)$ для вычисления $\inf_{\gamma \sim \Pi(p_r, p_g)}$.
- Преобразование формулы, основанное на двойственности Канторовича-Рубинштейна:

$$W(p_r, p_g) = \frac{1}{K} \sup_{\|f\|_L \leq K} \left(\mathbb{E}_{x \sim p_r}[f(x)] - \mathbb{E}_{x \sim p_g}[f(x)] \right)$$

где sup противоположен inf ; измеряем наименьшую верхнюю границу.

Непрерывность по Липшицу (1)

- Требуется, чтобы функция f в новой форме WD удовлетворяла $|f|_L \leq K$, т.е. должна быть K-непрерывной по Липшицу.
- Функция $f : \mathbb{R} \to \mathbb{R}$ называется K-непрерывной по Липшицу, если существует вещественная константа $K \geq 0$ такая, что для всех $x_1, x_2 \in \mathbb{R}$,

$$|f(x_1) - f(x_2)| \leq K|x_1 - x_2|.$$

- Здесь K - постоянная Липшица для функции f. Функции, всюду непрерывно дифференцируемые, непрерывны по Липшицу, так как производная, оцениваемая как $\frac{|f(x_1)-f(x_2)|}{|x_1-x_2|}$, имеет границы. Однако функция, непрерывная по Липшицу, может быть не всюду дифференцируемой, например $f(x) = |x|$.

Непрерывность по Липшицу (2)

- Пусть f - из семейства K-липшицевых функций $\{f_w\}_{w \in W}$, параметризованных w. В WGAN дискриминатор используется для обучения w, чтобы найти хорошую f_w, и функция потерь построена как WD между p_r и p_g:

$$\begin{aligned} L(p_r, p_g) &= W(p_r, p_g) \\ &= \max_{w \in W} \left(\mathbb{E}_{x \sim p_r}[f_w(x)] - \mathbb{E}_{z \sim p_r(z)}[f_w(g_\theta(z))] \right) \end{aligned}$$

- Т.о. дискриминатор больше не является критиком отличия поддельных примеров от настоящих. Вместо этого он обучается изучать K-липшицеву непрерывную функцию, чтобы помочь вычислить WD. По мере того, как функция потерь уменьшается при обучении, WD становится меньше, а выходные данные модели генератора приближаются к реальному распределению данных.

Непрерывность по Липшицу (3)

- Проблема - обеспечить K-липшицеву непрерывность f_w в процессе обучения.
- Есть простой, но практичный прием: после каждого обновления градиента ограничивать веса, например $[-0.01, 0.01]$, что приводит к компактному пространству параметров W и, таким образом, f_w получает свою нижнюю и верхнюю границы.

WGAN vs. GAN

- По сравнению с GAN в WGAN внесены изменения:
  - После каждого обновления градиента в функции f_w веса фиксируются в небольшом интервале $[-c, c]$.
  - Новая функция потерь, полученная из WD, используется без логарифма.
  - Дискриминатор выступает не как прямой “критик”, а как помощник для оценки WD между реальным и сгенерированным распределением данных.

Соперничающие сети - обучение G-сети

- Если на выходе D-сети стоит сигмоид, то она возвращает вероятность $D(x)$ того, что на вход сети подано “правильное” значение, т.е. G-сеть стремится максимизировать вероятность того, что D-сеть не отличит результат работы порождающей сети от “эталонных” образцов (а это означает, что порождающая сеть порождает правильные образцы).
- G-сеть учится по градиенту результата работы D-сети.

Соперничающие сети - обучение D-сети

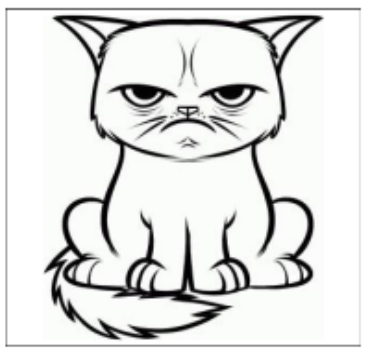
- D-сеть учится два раза за одни шаг обучения: первый раз - на вход подается эталонный образец, а второй раз - результат G-сети.
- D-сеть: максимизация функционала $D(x)(1 - D(G(z)))$. Цель банкира - одновременно положительно опознавать оригиналы $D(x)$, и отрицательно - подделки $(1 - D(G(z)))$.
- D-сеть ничего не знает о том, что ей подано на вход: эталон или подделка. Об этом “знает” только функционал. Однако сеть учится в сторону градиента этого функционала.

Соперничающие сети - обучение D-сети

D-сеть: возвращает 1 для реального образа, 0 - для шума

$$D(\ \ ) = 1$$

$$D(\ \ ) = 0$$



http://www.lab41.org/lab41-reading-group-generative-adversarial-nets/

Соперничающие сети - процесс обучения

1. На вход G-сети на каждом шаге подаются какие-то значения, например, совершенно случайные числа.
2. На вход D-сети на каждом шаге подается очередной эталонный образец и очередной результат работы G-сети.

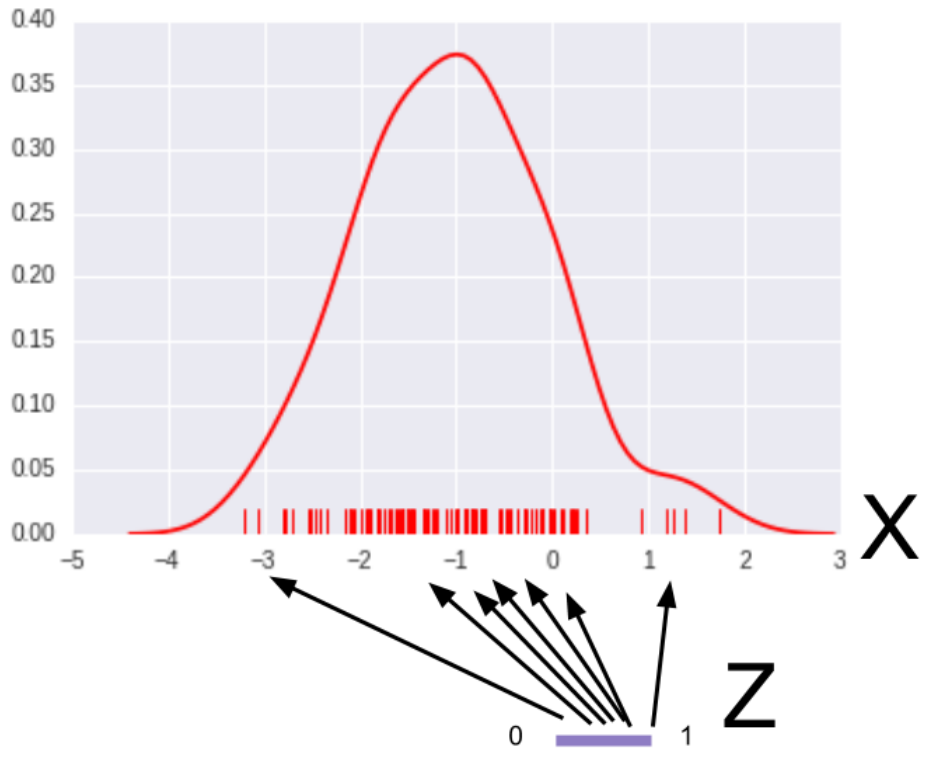
Результат:

1. G-сеть должна научиться генерировать образцы как можно более близкие к эталонным.
2. D-сеть должна научиться отличать эталонные образцы от порожденных.

Возвращаясь к порождающим моделям (практика)

- Нейронная сеть обучается генерированием простых нормально распределенных $\mathcal{N}(-1, 1)$ сл.величин.
- На входе G одиночное случайное число из равномерного распределения шума: $z \sim \mathrm{uniform}(0, 1)$.
- Мы хотим, чтобы G отобразила точки $z_1, ..., z_M$ в $x_1, ..., x_M$ так, чтобы густота точек $x_i = G(z_i)$ соответствовала “сжатости” функции $p_{data}(X)$.
- Таким образом, G берет z и генерирует поддельные x'.

Возвращаясь к порождающим моделям (практика)

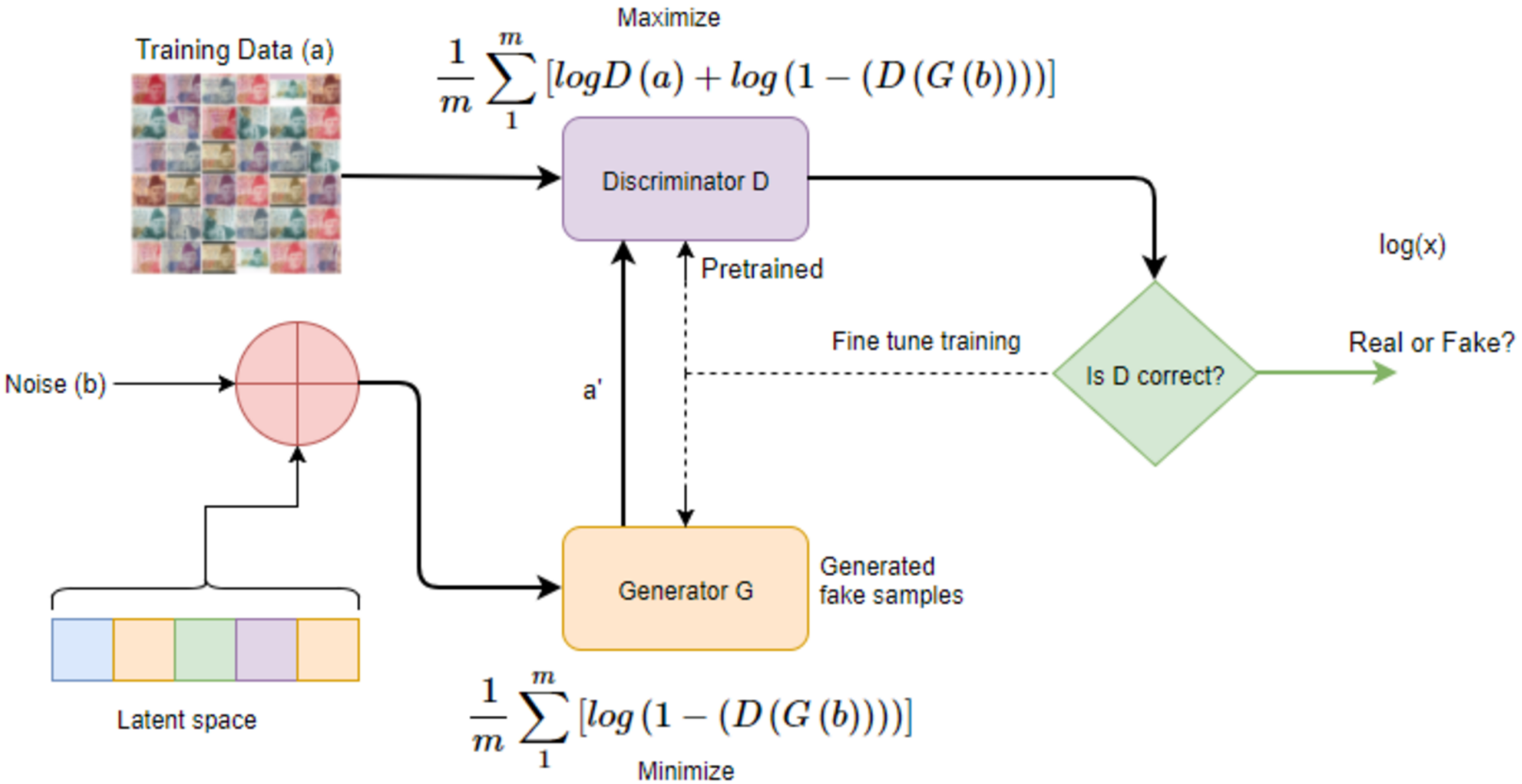


http://blog.evjang.com/2016_06_01_archive.html

Возвращаясь к порождающим моделям (практика)

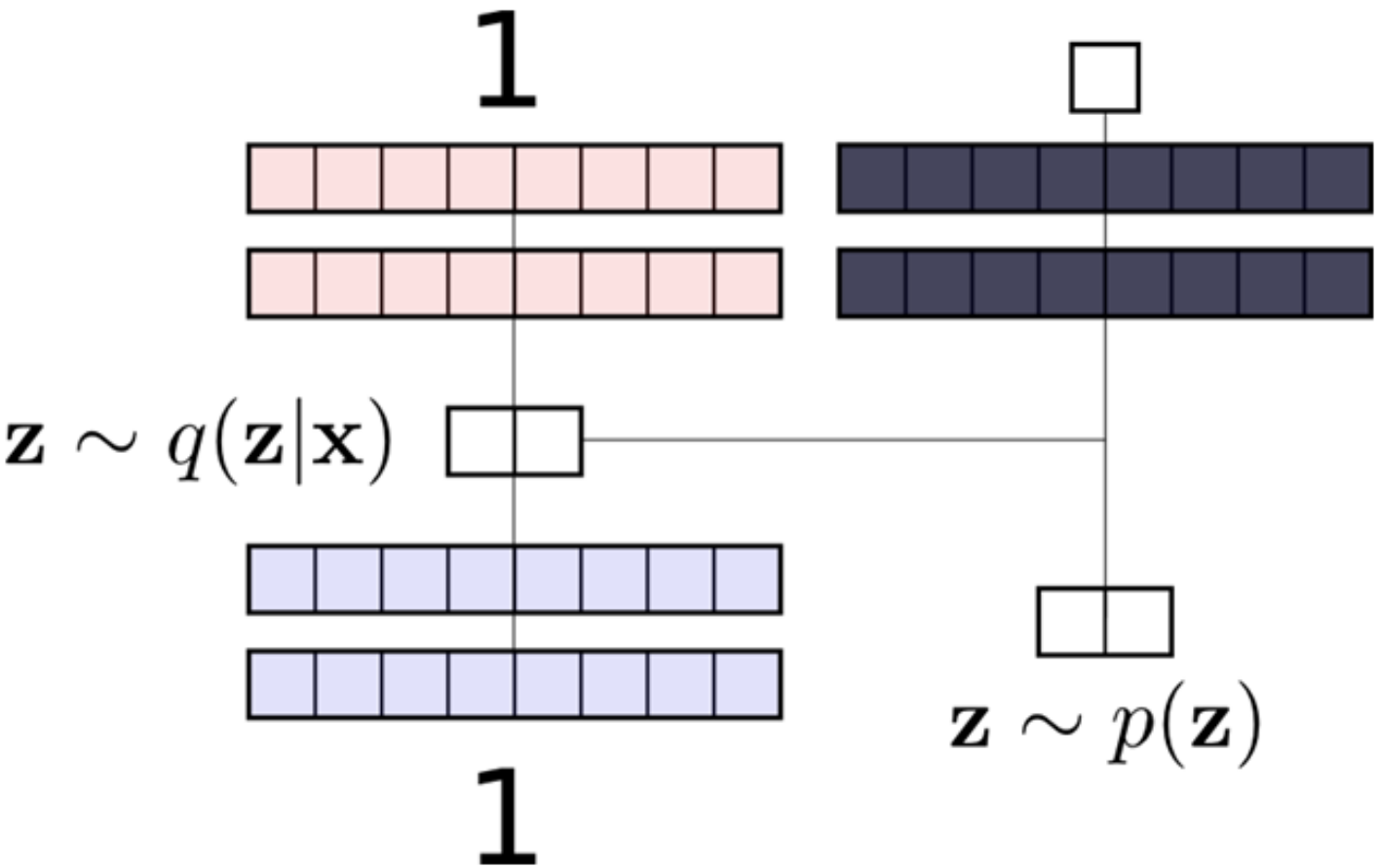
Тем временем на вход дискриминатора D подается x и на выходе - вероятность того, что вход принадлежит p_{data}. Пусть D_1 и D_2 копии D (они разделяют одни и те же параметры $D_1(x) = D_2(x)$). Вход D_1 - один сгенерированный вектор из "истинного" распределения $x \sim p_{data}$ так, чтобы максимизировать $D_1(x)$. На входе D_2 - вектор x' (поддельный вектор, сгенерированный сетью G) так, что, для оптимизации D мы минимизируем $D_2(x')$.

Соперничающие сети

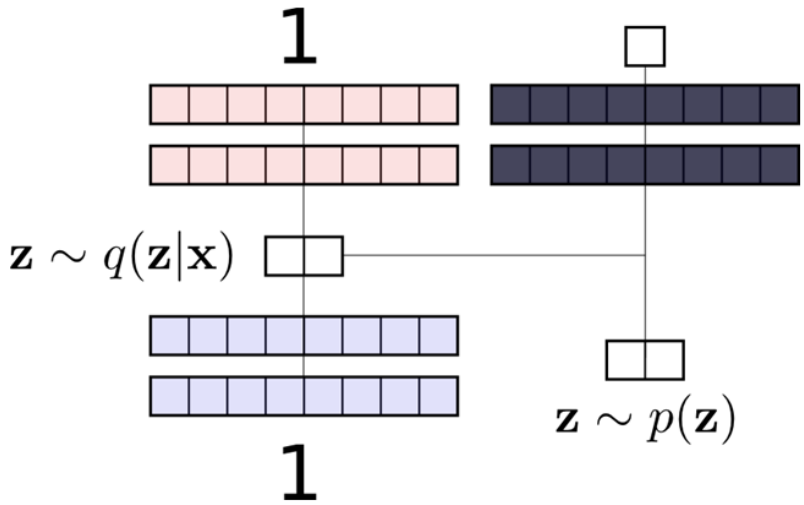


Соперничающие автокодеры

Соперничающие автокодеры - Adversarial autoencoders

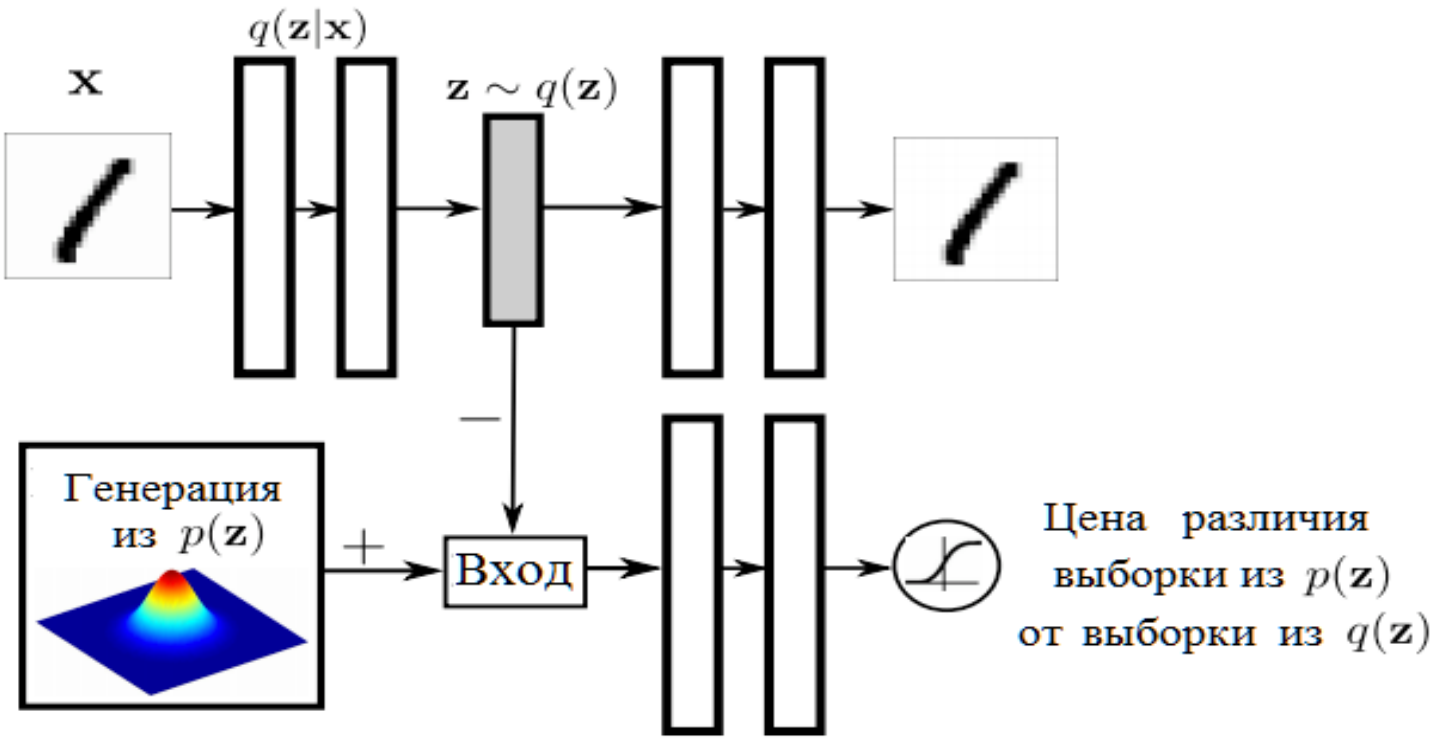


Соперничающие автокодеры

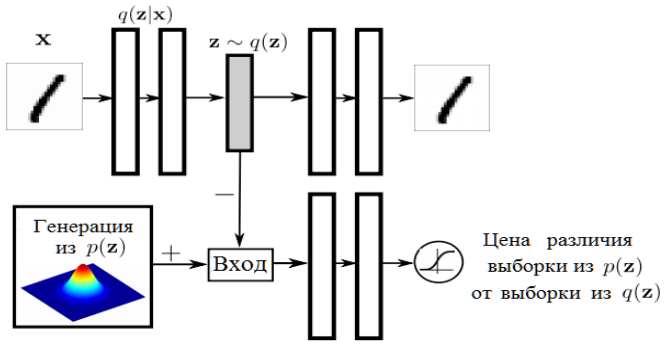


- Слева: входной вектор x (цифра “1”) преобразуется в код z кодером и поступает на декодер.
- Справа: выборочный вектор z генерируется в соответствии с априорным распределением $p(z)$. Дискриминатор оптимизируется, чтобы разделить выборки из $p(z)$ и из $q(z|x)$.

Соперничающие автокодеры

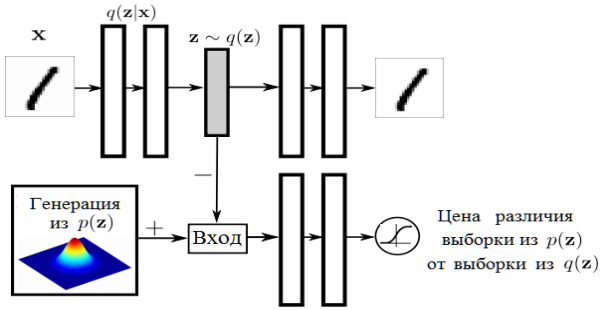


Соперничающие автокодеры



- Верхняя часть рисунка - вероятностный автокодер. Для данного входа x генерируется скрытый код z из кодирующего распределения $q(z|x)$ (моделируется как выход глубокой нейронной сети).
- В обычном автокодере этот кодер - детерминированный. Здесь он может быть вероятностным.
- Декодирующая сеть затем обучается, чтобы декодировать z и реконструировать исходный входной вектор x.
- Цель обучения - минимизация ошибки реконструкции (обычно квадратичная норма).

Соперничающие автокодеры



- Ошибка реконструкции обеспечивает то, что процесс кодирования сохраняет информацию о входном изображении, но она не осуществляет ничего другого, соответствующего скрытым векторам z.
- В общем, их распределение описывается как агрегированное апостериорное распределение $q(z) = \mathbb{E}_x q(z|x)$.

Соперничающие автокодеры

- Мы хотим, чтобы $q(z)$ совпадало с априорным $p(z)$. Это достигается введением дополнительного слагаемого в функцию потерь автокодера, который измеряет дивергенцию между q и p.
- Это можно сделать при помощи соперничающего обучения: обучаем разделяющую сеть, которая постоянно обучается, чтобы различать реальные кодовые векторы z, образованные кодированием реальных данных, от случайных кодовых векторов, сгенерированных из p. Если q почти полностью совпадает с p, то оптимальная разделяющая сеть должна иметь большую ошибку классификации.

Соперничающие автокодеры - события

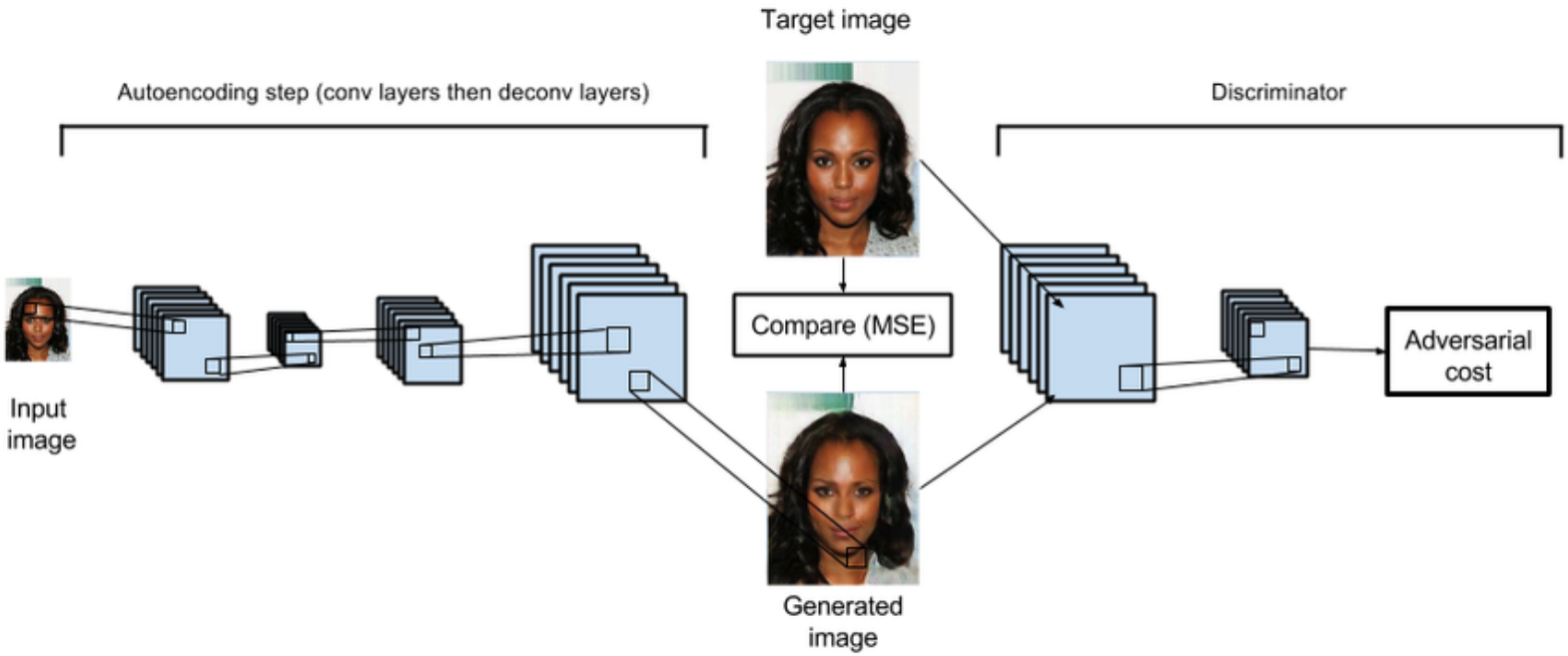
Для каждого блока данных есть три события:

1. Блок входных векторов кодируется и декодируется, после чего сеть модифицируется на основе стандартной ошибки реконструирования.
2. Блок входных векторов преобразуется кодером, после чего он соединятся с вектором, сгенерированным из априорного распределения $p(z)$. Дискриминатор затем модифицируется, используя двоичный кросс-энтропийный функционал потерь, основанный его способности разделять эти два вектора.

Соперничающие автокодеры - события (продолжение)

3. Блок входных векторов преобразуется кодером, источник этих данных прогнозируется дискритминатором, и генератор (кодер) модифицируется, используя двоичный кросс-энтропийный функционал потерь, основанный на способности “обмануть” дискриминатор в предположении, что данные - из априорного распределения.

Соперничающие автокодеры и сверточная сеть



https://swarbrickjones.wordpress.com/2016/01/24/generative-adversarial-autoencoders-in-theano/

Вопросы

?