Практический анализ данных и машинное обучение: искусственные нейронные сети

Соловей Влад иУльянкин Филипп

1 июня 2019 г.

Обзор современных архитектур

Agenda

- Генеративные нейросетки
- Вариационные автоэнкодеры
- GAN + VAE autoencoder

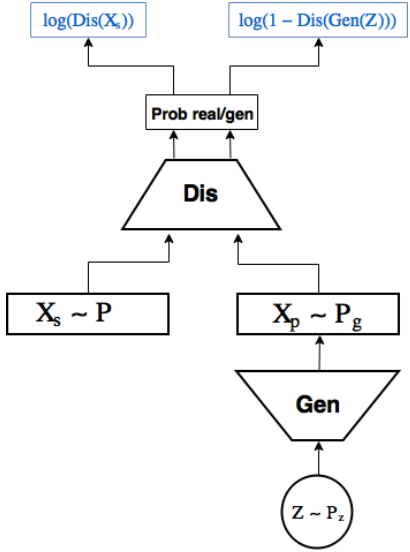
GAN

https://thispersondoesnotexist.com

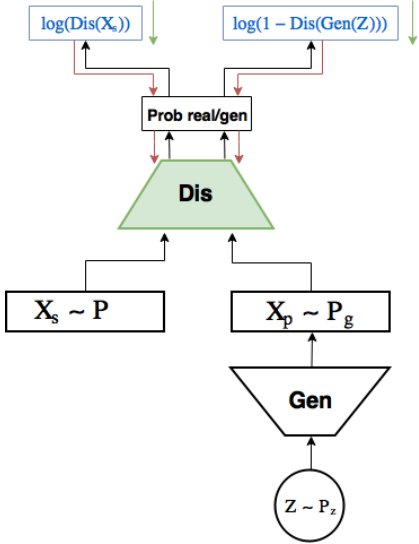
https://thiscatdoesnotexist.com

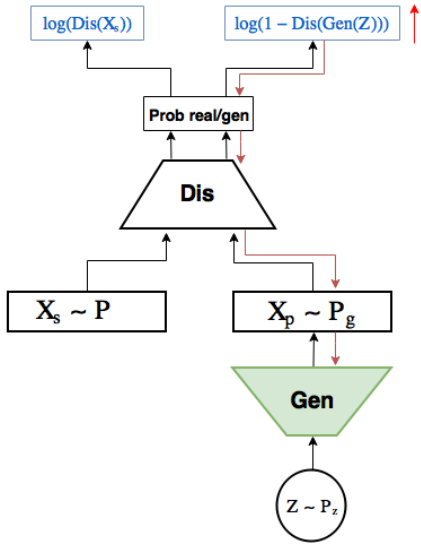
https://arxiv.org/pdf/1411.5928.pdf
https://rb.ru/story/chair-design/

Генеративные сети



Генеративные сети

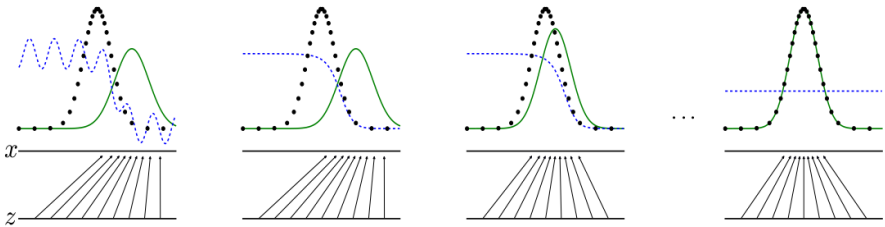




Генеративные сети

$$L_D = -y_i \cdot \ln \hat{p}_i - (1 - y_i) \cdot \ln(1 - \hat{p}_i)$$
$$L_G = -\ln(1 - \hat{p}_i)$$

Генеративные сети



Собираем свой GAN

VAE-autoencoder

Вариационные автоэнкодеры

- Придумано в 2014 году.
- Продолжение идей разреженного автоэнкодера, добавляем новых хотелок на наше признаковое пространство.
- По своей сути является генеративной моделью. Мы хотим найти удобное нам пространство, откуда удобно сэмплить точки для создания нового.

Немного формул

$$P(X) = \int_x P(X|Z)P(Z)dZ,$$

$P(X)$ -- вероятностное распределение изображений цифр на картинках, т.е. вероятность конкретного изображения цифры в принципе быть нарисованным; $P(Z)$ -- вероятностное распределение скрытых факторов, например, распределение толщины штриха; $P(X|Z)$ -- вероятность быть нарисованной конкретную картинку, при условии скрытых факторов. Можно представить следующим образом: $P(X|Z) = f(Z) + \epsilon$.

Немного формул

Мы хотим настроить свой генерирующий процесс.

$$P(X|Q) = \int_x P(X|Z;Q)P(Z)dZ.$$

Где $P(X|Z;Q) = f(Z;Q) + \epsilon$. Если выбрать метрику для сравнения близости объектов как расстояние, то получим красивый вывод --
$P(X|Z;Q) = N(X|f(Z;Q), \sigma^2 I)$.
Напрямую оптимизировать интеграл мы не можем. Но для каждого X подмножество хороших Z не очень велико. Введем $Q(Z|X)$ -- распределение хороших Z для каждого конкретного X.

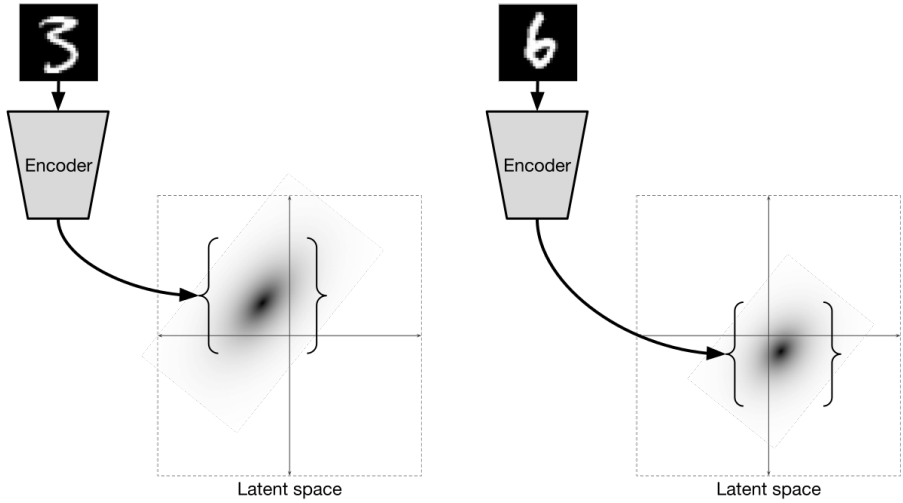
Немного формул

Сим салабим магия переходов, расстояния Кульбака-Лейблера и остальное очень важной математики. Получим следующее тождество.

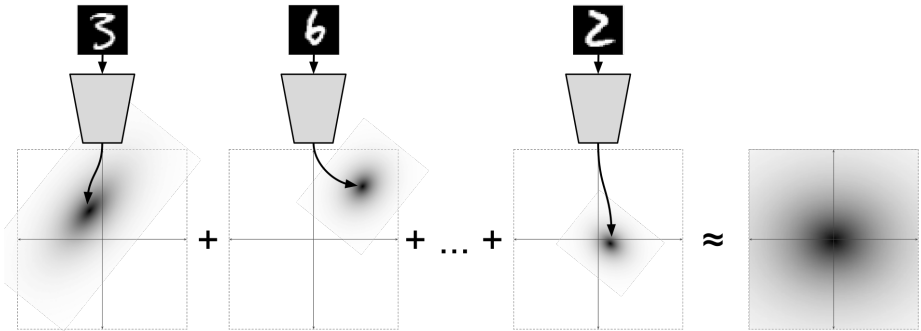
$$\log P(X;\theta_2) - KL[Q(Z|X;\theta_1)||P(Z|X;\theta_2)] =$$
$$\mathbb{E}_{Z\sim Q}[\log P(X|Z;\theta_2)] - KL[Q(Z|X;\theta_1)||N(0,I)]$$

Где $P(Z)$-имеет нормальное распределение. $Q(Z|X;Q_1)$ -- очень похож на энкодер, а $P(X|Z;Q_2) = f(Z;Q_2) + \epsilon$ -- похоже на декодер.Если $Q(Z|X;Q_1) = N(\mu(X;Q_1), \Sigma(X;Q_1))$ -- мы можем оптимизировать всю нашу красоту.

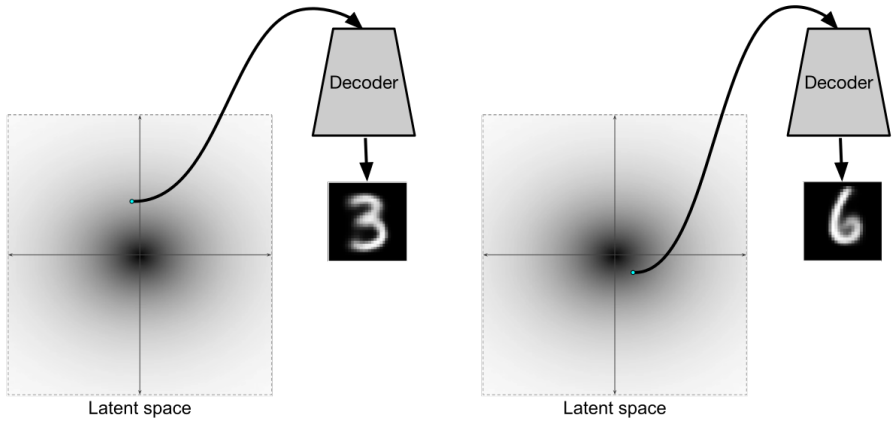
Вариационные автокодировщики



Вариационные автокодировщики

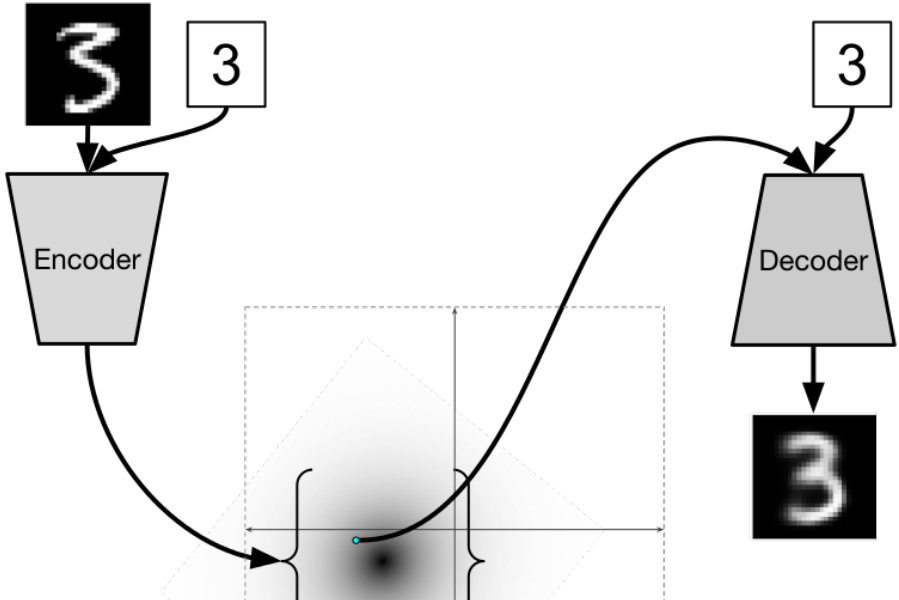


Вариационные автокодировщики



По сути нам нужен сам декодер, который сможет получая вход скрытое представление генерировать нам картинку.

Условный вариационный автокодировщик



Собираем свой VAE-autoencoder

Соединяя VAE и GAN

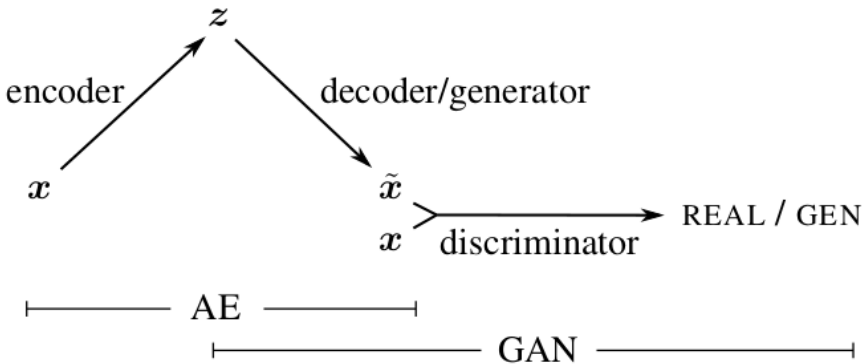
Проблема автокодировщика в том, что он пытается считать ошибку попиксельно.

Для него а, ближе к б,чем к с. Хотя для нас, человеков это не совсем так.

Соединяя VAE и GAN

Генератор GAN выполняет функцию, аналогичную декодеру в VAE: оба сэмплят из априорного распределения $P(Z)$ и переводят его в $P_g(X)$.
Однако роли у них разные: декодер восстанавливает объект, закодированный энкодером, при обучении опираясь на некоторую метрику сравнения; генератор же генерирует случайный объект, который ни с чем не сравнивается, лишь бы дискриминатор не мог отличить, какому из распределений P или P_g он принадлежит.
Идея: добавить в VAE третью сеть — дискриминатор и подавать ей на вход и восстановленный объект и оригинал, а дискриминатор обучать определять, какой из них какой.

Соединяя VAE и GAN



Соединяя VAE и GAN

Нам требуется немного усложнить подход, чтобы мы просто не выучили нашу выборку и не уперлись в проблему слишком умного дискриминатора.