Анализ неструктурированных данных

Семинар 10
GAN-ы для текстов

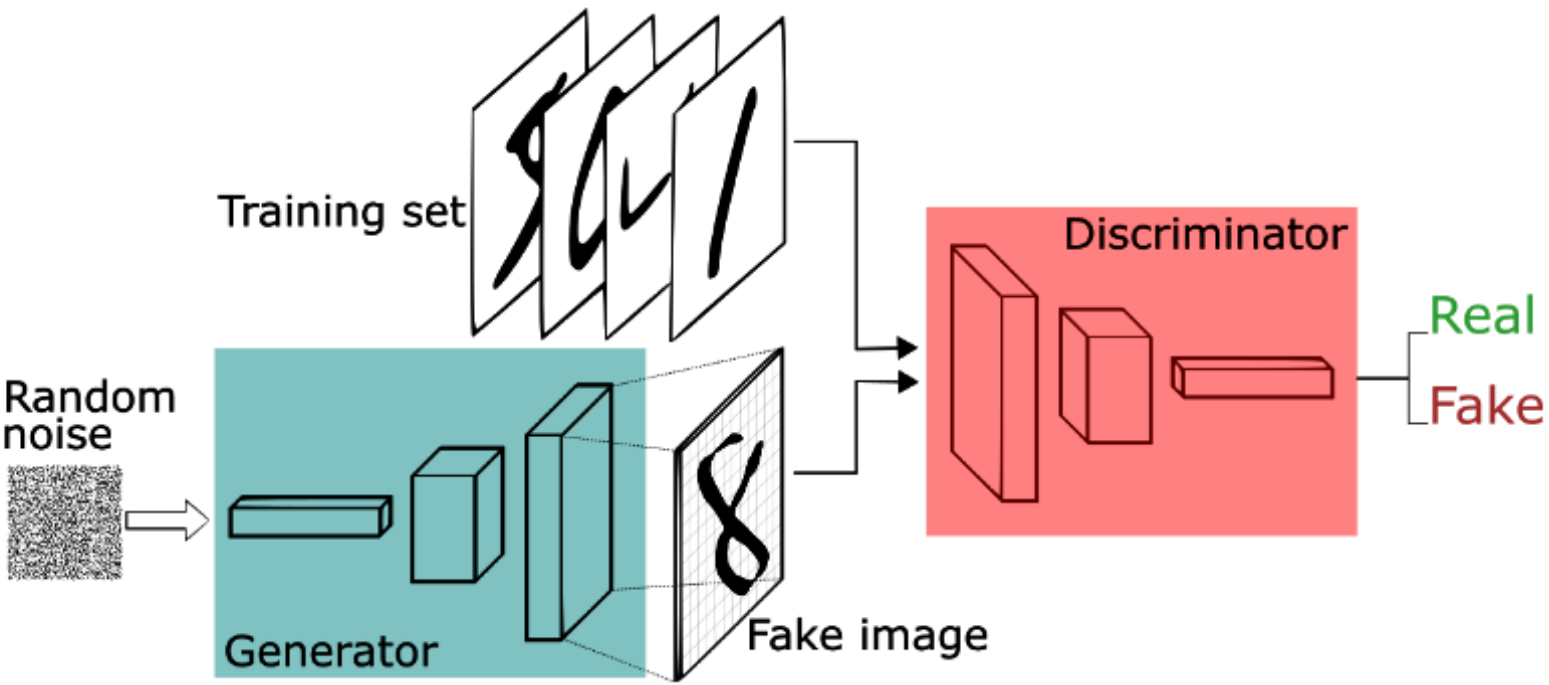
Национальныи Исследовательский Университет
Высшая Школа Экономики

14 - 15 ноября 2018

Generative Adversarial Net

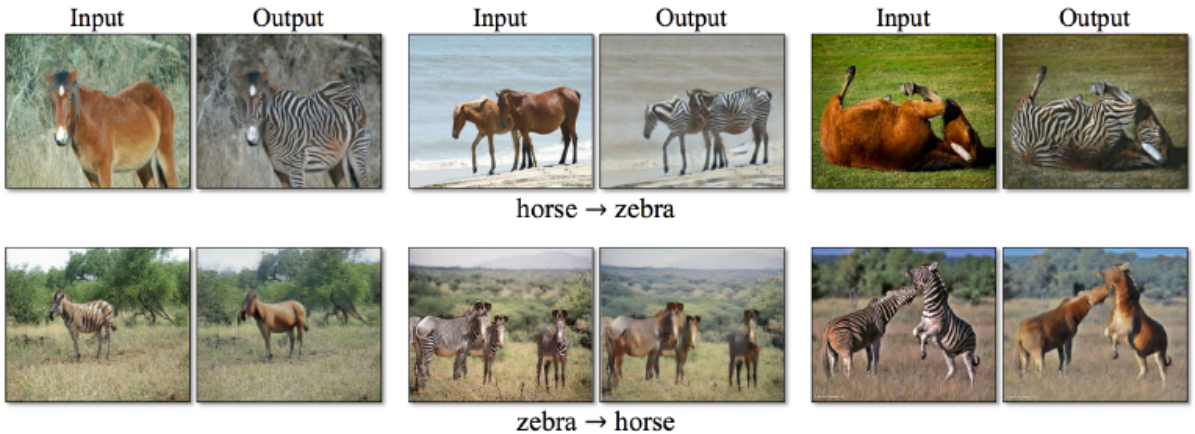
- система, состоящая из generator-a, discriminator-a. Generator порождает примеры, discriminator пытается отличить примеры, порожденные generator-ом от примеров, взятых их обучающей выборки (выдает вероятность того, что данный пример - реальный пример, а не "фальшивка").
- Процесс обучения можно описать как игру с нулевой суммой, в которой функция $v(\theta^g, \theta^d)$ - платеж discriminator-a, а генератор получает $-v(\theta^g, \theta^d)$ в качестве своего платежа.
- В процессе игры каждый пытается максмизировать свой платеж $g^* = argmin_g max_d v(g, d)$ (Идея в том, что не минимизируем лосс(y, p), а решаем другую оптимизационную задачу - максимизации функции успеха)

Generative Adversarial Net



- $v(\theta^g, \theta^d) = E_{x \sim P_{data}} \log d(x) + E_{x \sim P_{model}} \log(1 - d(x))$,
- $L^d(\theta^g, \theta^d) = -\frac{1}{2} E_{x \sim P_{data}} \log d(x) - \frac{1}{2} E_z \log(1 - d(g(z)))$, где z - input noize.
- $L^g(\theta^g, \theta^d) = -L^d(\theta^g, \theta^d)$
- Проблема: процедура в общем случае не сходится => недообучение.
- Generator пытается увеличить логарифм вероятности того, что дискриминатор допустит ошибку (а не уменьшить логарифм вероятности, что дискриминатор сделает правильное предсказание)
- $L^g(\theta^g, \theta^d) = -\frac{1}{2} E_z \log(1 - d(g(z)))$

- Успешно применяются для картинок



- Тут можно поиграться с ганами в браузере
- Получится ли применить для текстов?

GAN-ы для текстов: проблемы?

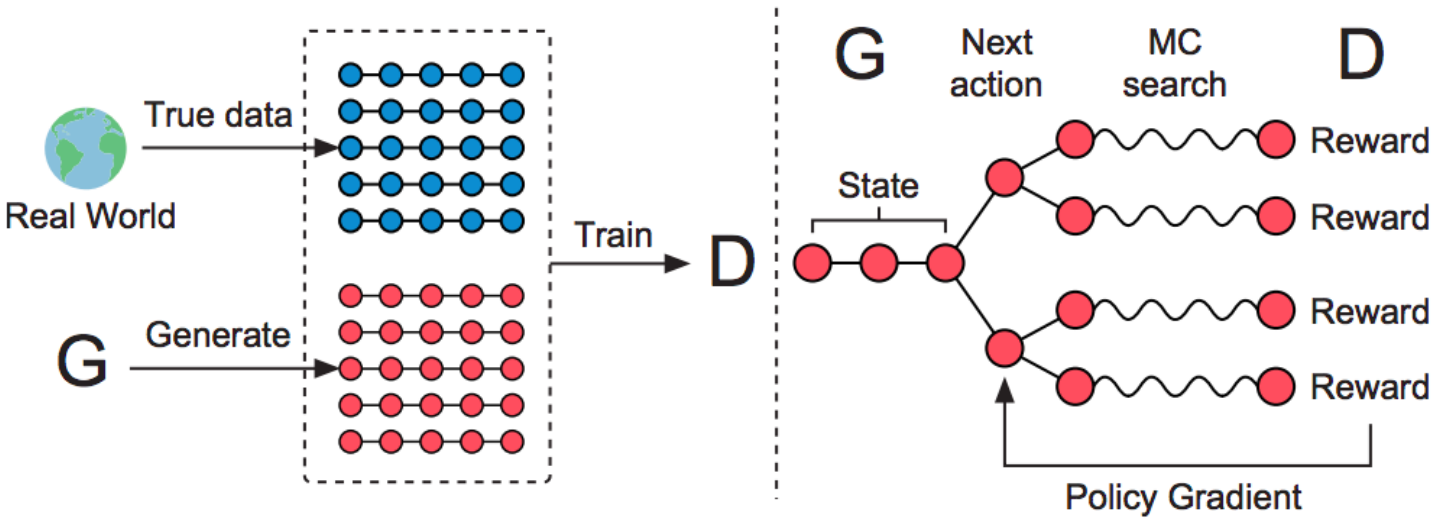
- есть ограничения, когда хотим генерировать последовательности дискретных токенов (не можем пробросить градиент лосса от discriminator-a w.r.t. выхода generator-a)
- дискриминативная модель может оценить, насколько хорошо сгенерировано полное предложение, а по сгенерированной части нельзя понять ни текущий скор, ни будущий (когда полное предложение будет сгенрировано))

Sequence generation procedure as a Sequential decision Making Process

- Generator - агент in RL
- State - уже сгенерированные токены. Action - токен, который мы хотим сгенерировать.
- Reward приходит от discriminator-a (оцененный по полной последовательности)
- Поскольку в силу дискретности аутпут-а мы не можем пробросить градиент к генератору, исопльзуем stochastic parametrized policy. (С Monte-Carlo search для аппроксимации search-value action.)

Sequence GAN

- Обучаем модель G_θ генерировать последовательность $Y_{1:T} = (y_1, ..y_T), y_t \in Y$ - весь словарь.
- В момент времени t текущее состояние - это $(y_1, ..y_{t-1})$, а действие - следующий токет y_t
- $G_\theta(y_t|Y_{1:t-1})$ - стохастическая
- State transition, после того, как действие выбрано, - детерминировано.
- Обучаем модель D_ϕ
- $D_\phi(Y_{1:T})$ - вероятность того, что последовательность $Y_{1:T}$ - из исходного обучающей выборки (не сгенерированный генератором)


True data
Real World
Generate
G
Train
D
G
Next action
MC search
D
State
Reward
Reward
Reward
Reward
Policy Gradient

SeqGAN via Policy Gradient

- Цель генератора - сгенерировать последовательность от начального состояние s_0, максимизировав ожидаемый конечный reward

$$J(\theta) = E[R_T|s_0, \theta] = \sum_{s_1 \in Y} G_\theta(y_1|s_0) Q_{D_\phi^\theta}(s_0, y_1),$$

- R_T - reward за все предложение (от дискриминатора).
- $Q_{D_\phi}^{G_\theta}(s_0, y_1)$ - action-value function последовательности (м.о. реворда, если начать в состоянии s, приняв действие a, следую стратегии G_θ)
- Как оценить action-value funciton? REINFORCE алгоритм
- $Q_{D_\phi}^{G_\theta}(a = y_T, s = Y_{1:T-1}) = D_\phi(Y_{1:T})$

SeqGAN via Policy Gradient

- Поскольку нам нужен хороший в долгосрочной перспективе (когда сгенерируем все предложение), будем использовать MC search
- Чтобы уменьшить дисперсию и лучше оценить action value, запустимся N раз:

$$Q_{D_\phi}^{G_\theta}(s = Y_{1:t-1}, a = y_t) = \tag{4}$$

$$\begin{cases} \frac{1}{N}\sum_{n=1}^{N} D_\phi(Y_{1:T}^n),\ Y_{1:T}^n \in \mathrm{MC}^{G_\beta}(Y_{1:t}; N) & \text{for} \quad t < T \\ D_\phi(Y_{1:t}) & \text{for} \quad t = T, \end{cases}$$

SeqGAN via Policy Gradient

- Обучаем дискриминатор:

$$\min_{\phi} -\mathbb{E}_{Y\sim p_{\text{data}}}[\log D_\phi(Y)] - \mathbb{E}_{Y\sim G_\theta}[\log(1 - D_\phi(Y))]$$

- Как только обучили дискриминатор, переходим к генератору:

$$\nabla_\theta J(\theta) = \sum_{t=1}^{T} \mathbb{E}_{Y_{1:t-1}\sim G_\theta}\Big[\sum_{y_t\in\mathcal{Y}} \nabla_\theta G_\theta(y_t|Y_{1:t-1}) \cdot Q_{D_\phi}^{G_\theta}(Y_{1:t-1}, y_t)\Big]$$

SeqGAN via Policy Gradient - Algorithm

Require: generator policy G_θ; roll-out policy G_β; discriminator D_ϕ; a sequence dataset $\mathcal{S} = \{X_{1:T}\}$

Initialize G_θ, D_ϕ with random weights θ, ϕ.
Pre-train G_θ using MLE on $\mathcal{S}$
$\beta \leftarrow \theta$
Generate negative samples using G_θ for training D_ϕ
Pre-train D_ϕ via minimizing the cross entropy
repeat
  for g-steps **do**
    Generate a sequence $Y_{1:T} = (y_1, \ldots, y_T) \sim G_\theta$
    for t in $1 : T$ **do**
      Compute $Q(a = y_t; s = Y_{1:t-1})$ by Eq. (4)
    end for
    Update generator parameters via policy gradient Eq. (8)
  end for
  for d-steps **do**
    Use current G_θ to generate negative examples and combine with given positive examples $\mathcal{S}$
    Train discriminator D_ϕ for k epochs by Eq. (5)
  end for
  $\beta \leftarrow \theta$
until SeqGAN converges

- Генератор - RNN: маппит инпут эмбеддинговых представлений токенов последовательности в последовательность скрытых состояний. Затем softmax маппит скрытые состояния в распределение над выходными токенами
- Дискриминатор - CNN + max pooling, FC-layer, Softmax

В чем проблема SeqGAN?

- Одной чиселки от дискриминатора - мало
- Хотим получать от дискриминатора какой-то набор фичей

Материалы

- Пост на хабре
- Статья про SeqGAN
- Статья про LeakGAN