Generative Adversarial Network (GAN)

Бурштеин Денис

МОП 162

Что хотим

- Есть какое-то множество объектов, например картинки или музыка
- Хотим модель, которая может генерировать новые объекты
- Эти новые объекты должны быть такими, что даже эксперт не сможет распознать, где настоящие объекты, а где сгенерированные

GAN

- 2014 год - Generative Adversarial Network от Google
- Идея проста: есть 2 модели - генератор и дискриминатор
- Дискриминатор – тот самый эксперт, который умеет отличать настоящие объекты от подделок
- Генератор – модель, генерирующая новые объекты
- «Противостояние эксперта и мошенника»

GAN формально

- p_r - распределение настоящих объектов (r – real)
- p_z - шумовое распределение (обычно просто равномерное)
- p_g - распределение генератора (g – generator)
- $D(x)$ - дискриминатор, оценивает вероятность того, что x - настоящий
- $G(z)$ - генератор, переводит шум $z \sim p_z(z)$ в объекты

GAN формально

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

$$= \mathbb{E}_{x\sim p_r(x)}[\log \underbrace{D(x)}] + \mathbb{E}_{x\sim p_g(x)}[\log \underbrace{(1-D(x)}]$$

Вероятность на настоящих объектах

Обратная вероятность на подделках

Оптимальный дискриминатор

- Оптимальный дискриминатор при фиксированном генераторе:

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

- Если генератор обучен до оптимума, т.е. p_g практически совпадает с p_r, то оптимальный дискриминатор просто равен $1/2$

Оптимальный дискриминатор

- Значение функции потерь при оптимальном дискриминаторе:

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

Дивергенции KL и JS:

$$D_{KL}(p\|q) = \int_x p(x) \log \frac{p(x)}{q(x)} dx$$

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

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

for number of training iterations **do**

  for k steps **do**

    • Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.

    • Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.

    • Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$$

  end for

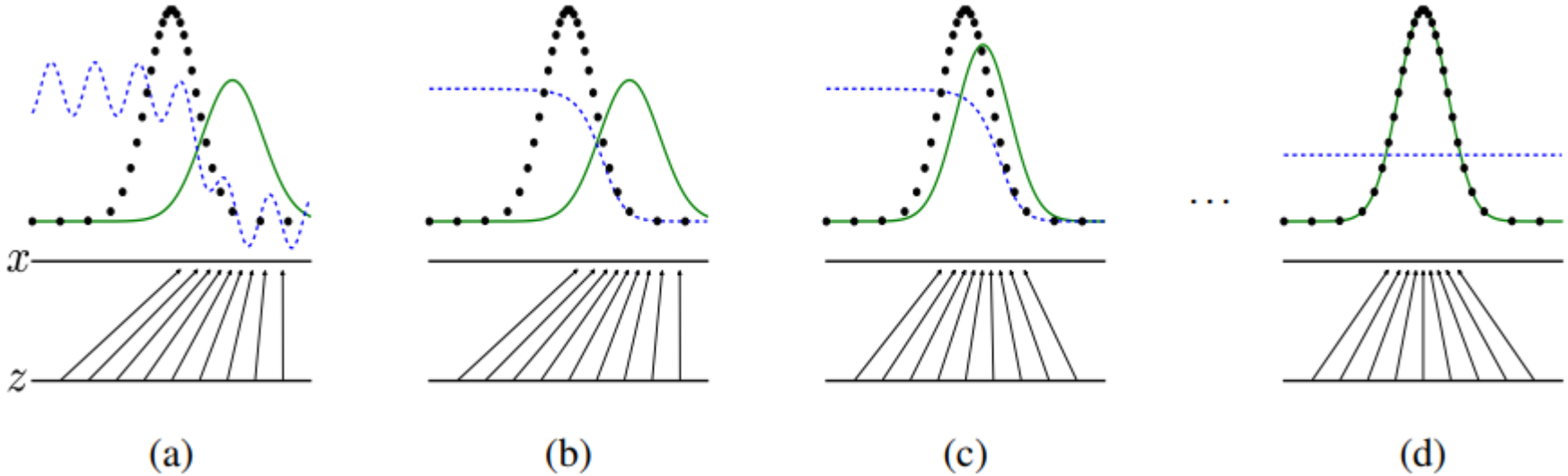
  • Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.

  • Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

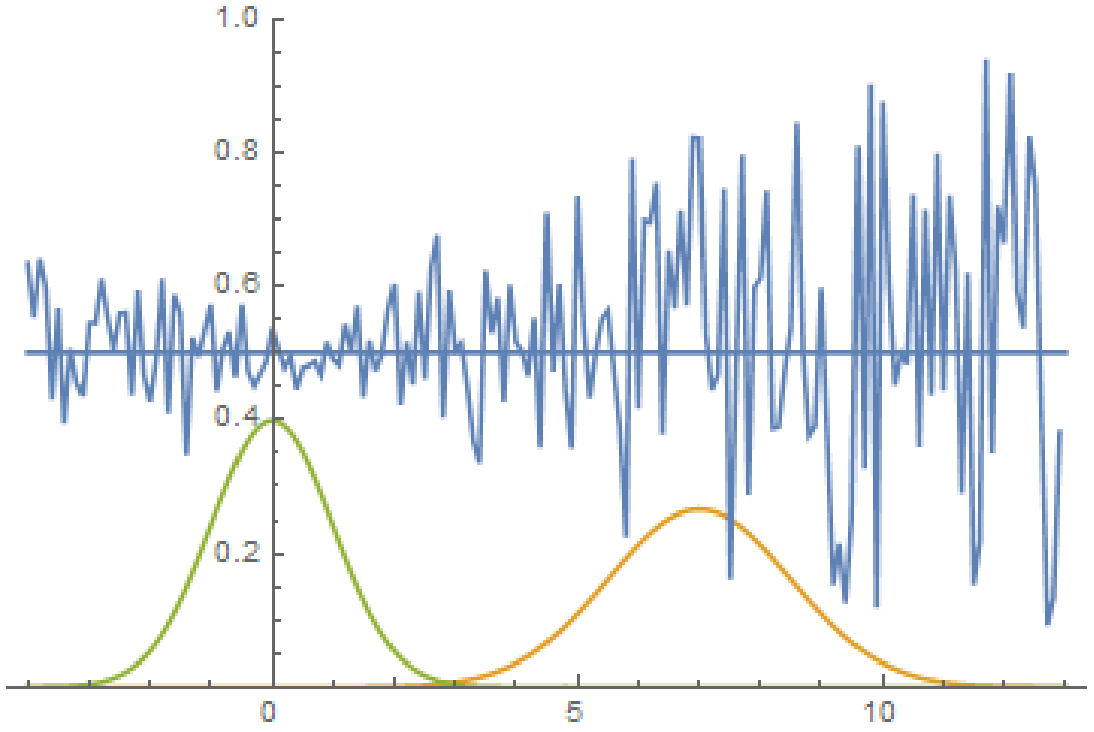
end for

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

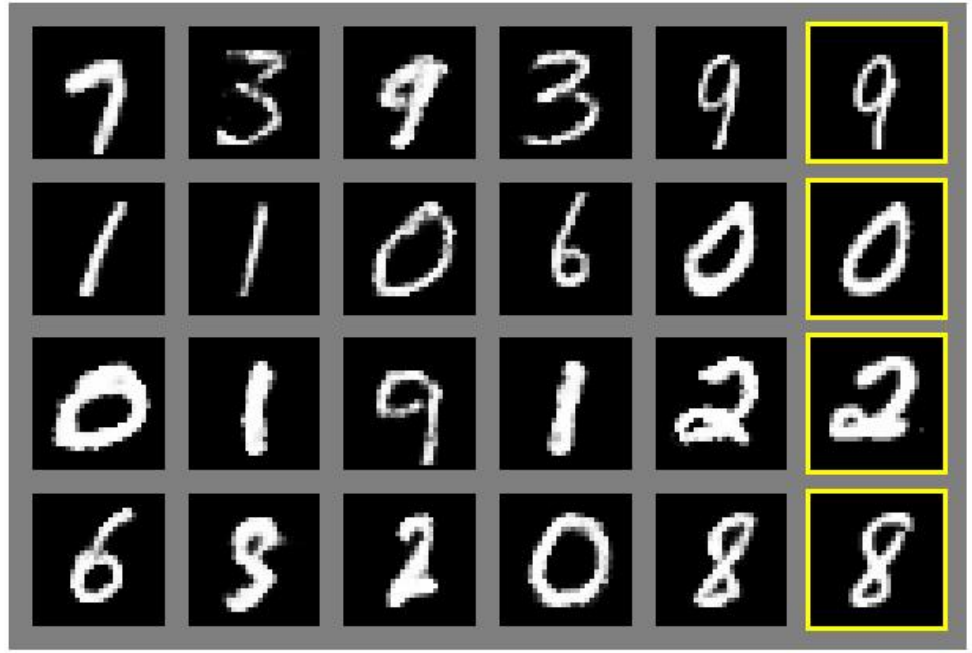


Изменение дискриминатора и распределения генератора по ходу обучения

Синий – график дискриминатора, чёрный – распределение настоящих объектов, зелёный – распределение генератора (генерируемых объектов)



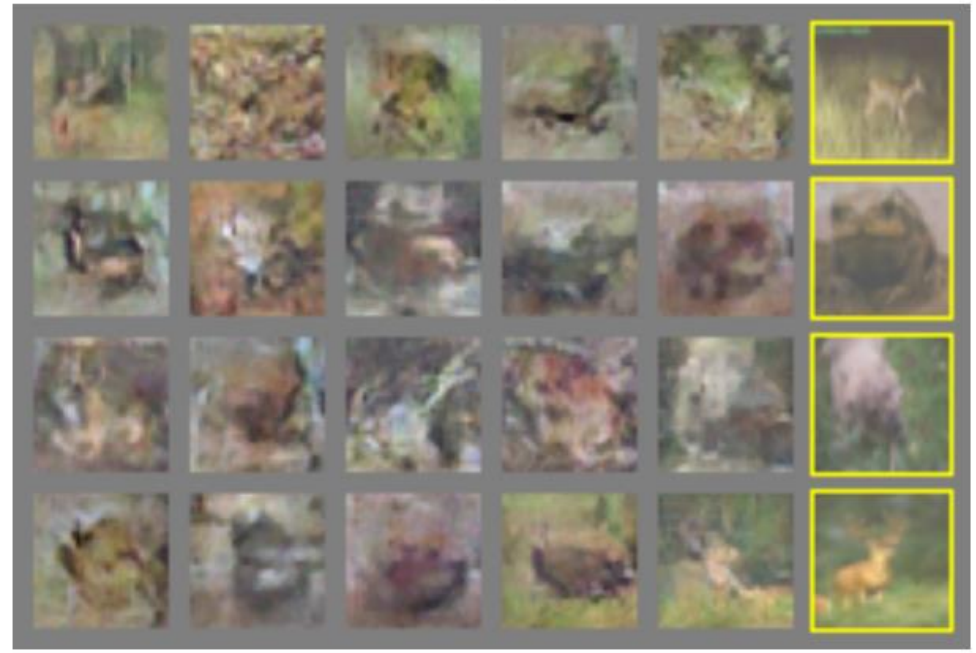
Оранжевый – настоящее распределение, зелёный – распределение генератора, синий – дискриминатор



a)

b)

c)



d)

Применение

- Генерация новых качественных объектов для других задач машинного обучения
- Ещё один шаг к полноценному ИИ
- Генерация реалистичных лиц для видеоигр
- Всяческие развлекательные программы

Проблемы GAN

- Затухание градиента

Проблемы GAN

- Затухание градиента
- Mode collapse

Проблемы GAN

- Затухание градиента
- Mode collapse

- Отсутствие каких-то метрик качества

Проблемы GAN

- Затухание градиента
- Mode collapse

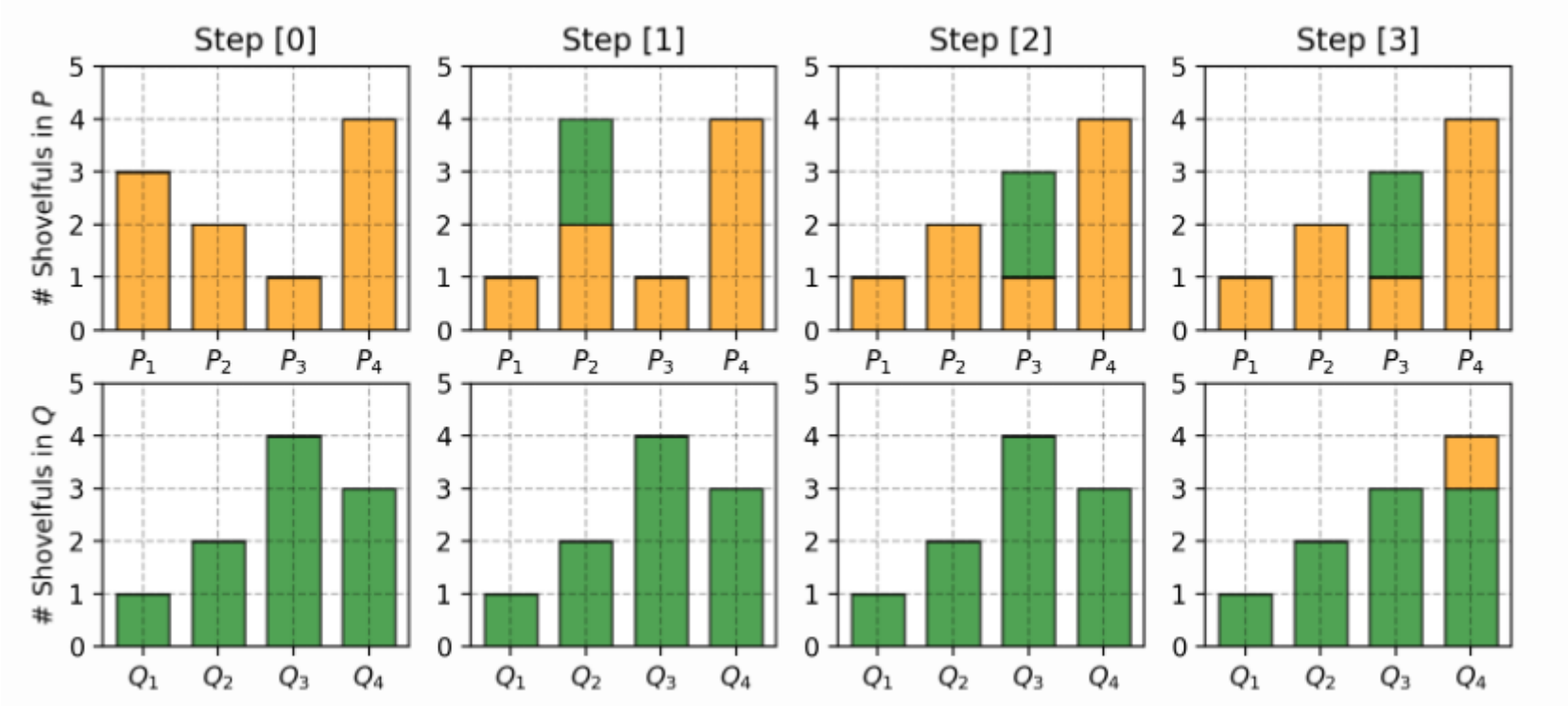
- Отсутствие каких-то метрик качества
- Нестабильность

Wasserstein GAN (WGAN)

- WGAN – одно из возможных улучшений идеи GAN, было предложено в 2017 году.
- Основная идея – перейти от задачи оптимизации дивергенции Йенсена-Шеннона к оптимизации расстояния Вассерштейна, которое показывает себя лучше в случае, когда реальное распределение и распределение генератора слишком сильно «разделены».

Расстояние Вассерштейна

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



Расстояние Вассерштейна

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

Из двойственности Канторовича-Рубинштейна:

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

К-Липшицева функция

Функция потерь WGAN

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

$\{f_w\}_{w \in W}$ - параметрическое семейство К-Липшицевых функций с параметром w (так называемый критик, замена дискриминатору)

Теперь наша задача: $\min_{g_\theta} L(p_r, p_g), g_\theta$ - генератор

«Weight clipping»

- Вся эта задумка работает только при условии Липшицевости функции
- Чтобы она была таковой авторы «сжимают» параметр в окно $[-c, c]$

Algorithm 1 WGAN, our proposed algorithm. All experiments in the paper used the default values $\alpha = 0.00005$, $c = 0.01$, $m = 64$, $n_{\text{critic}} = 5$.

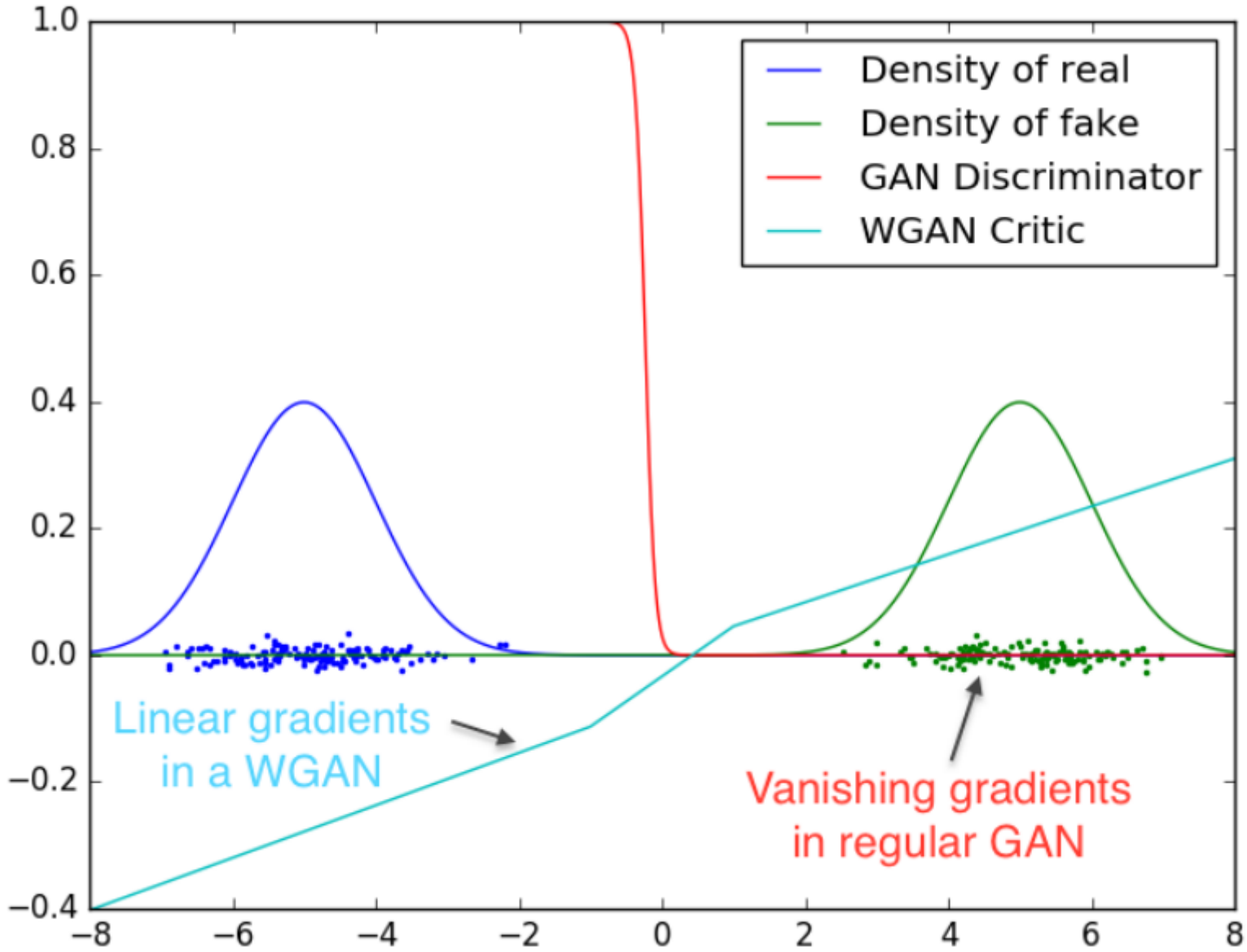
Require: : α, the learning rate. c, the clipping parameter. m, the batch size. n_{critic}, the number of iterations of the critic per generator iteration.

Require: : w_0, initial critic parameters. θ_0, initial generator's parameters.

while θ has not converged **do**
  for $t = 0, ..., n_{\text{critic}}$ **do**
    Sample $\{x^{(i)}\}_{i=1}^{m} \sim \mathbb{P}_r$ a batch from the real data.
    Sample $\{z^{(i)}\}_{i=1}^{m} \sim p(z)$ a batch of prior samples.
    $g_w \leftarrow \nabla_w \left[\frac{1}{m}\sum_{i=1}^{m} f_w(x^{(i)}) - \frac{1}{m}\sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))\right]$
    $w \leftarrow w + \alpha \cdot \text{RMSProp}(w, g_w)$
    $w \leftarrow \text{clip}(w, -c, c)$
  end for
  Sample $\{z^{(i)}\}_{i=1}^{m} \sim p(z)$ a batch of prior samples.
  $g_\theta \leftarrow -\nabla_\theta \frac{1}{m}\sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))$
  $\theta \leftarrow \theta - \alpha \cdot \text{RMSProp}(\theta, g_\theta)$
end while

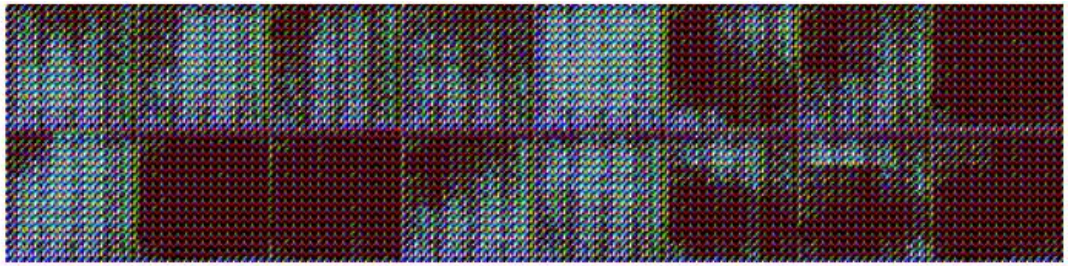
Основные отличия от GAN

- «Weight clipping», позволяющий получать Липшицевы функции
- Используется расстояние Вассерштейна, которое показывает себя лучше, чем дивергенция Йенсена-Шеннона, особенно в случае «разделённых» распределений.
- Как такового дискриминатора больше нет, вместо него Липшицева функция, поэтому не может случиться переобучения дискриминатора, которое приводило к затуханию градиента в случае обычного GAN.


Density of real
Density of fake
GAN Discriminator
WGAN Critic
Linear gradients
in a WGAN
Vanishing gradients
in regular GAN
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−8
−6
−4
−2
0
2
4
6
8

Проблемы WGAN

- Всё та же нестабильность
- «Weight clipping», используемый для гарантирования Липшицевости, достаточный грубый метод. Когда «окно», в которое сжимают параметр, слишком большое, обучение сходится очень медленно, когда же оно слишком маленькое возникает проблема затухания градиента.



WGAN

GAN

Видно, что в этом случае стандартный GAN вообще не сошёлся, а с WGAN всё отлично.

Источники

- https://lilianweng.github.io/lil-log/2017/08/20/from-GAN-to-WGAN.html
- https://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf
- https://arxiv.org/pdf/1701.07875.pdf
- https://habr.com/ru/post/352794/