Двойственный подход к обучению генеративных состязательных сетей

М.Г.Бочко, Е.А.Жестов, А.Р.Хуснутдинов

Московский физико-технический институт (национальный исследовательский университет)

141701, Московская область, г. Долгопрудный, Институтский переулок, д.9.

Abstract

В данной работе будет проанализирован новый метод тренировки генеративно-состязательных нейронных сетей. В его основе лежит переформулирование исходной минимаксной задачи как задачи нахождения седловой точки лагранжиана выпуклой оптимизации. В двойственной задаче значения функции дискриминатора будут играть роль прямых переменных, а сгенерированное распределение - роль двойственной переменной. Новый взгляд на задачу обучения приводит к модификации правил обновления параметра генератора. Будут проведены эксперименты, мотивирующие исследовать границы применимости предложенного метода.

1. Введение

Генеративно-состязательная сеть (GAN) представляет собой алгоритм машинного обучения, построенный на комбинации из двух нейронных сетей. Первая нейронная сеть называется *генератором*, а вторая *дискриминатором*. Генератор сэмплирует данные из случайного шума. Дискриминатор же классифицирует данные, т.е старается отличить сгенерированные от подлинных.

Целью генератора является повысить процент ошибок дискриминатора, а целью дискриминатора является, наоборот, улучшение точности распознавания. Такие нейронные сети впервые были предложены в статье [1].

Задача обучения GAN формулируется как следующая минимаксная задача:

$$\min_G \max_D \mathbb{E}_{x \backsim p_d(x)}\{\log D(x)\} + \mathbb{E}_{z \backsim p_z(z)}\{\log(1 - D(G(z)))\}, \qquad (1)$$

где $D(x) : \mathbb{R}^n \to \{1, 0\}$ - функция дискриминатора, определяющая метку класса по входному вектору x, p_d — истинное распределение данных, $G(z) : \mathbb{R}^k \to \mathbb{R}^n$ — функция генератора для создания данных из пространства с некоторым распределением $p_z(z)$.

Дискриминатор и генератор определяются параметрами θ_d и θ_g соответственно. Параметры генератора и дискриминатора обновляются итеративно с помощью градиентного спуска.

В работе [2] описана проблема, связанная с обучением, которая состоит в том, что в какой-то момент генератор начинает предлагать одну и ту же модель, лучше всего подходящую под дискриминатор. Эта проблема называется *коллапсом моды*. Для корректной работы GAN необходимо, чтобы генерировались все классы данных, в противном случае обучение дискриминатора происходит на несбалансированной выборке.

В работе рассматривается замена задачи на эквивалентную: вместо нахождения оптимальных параметров сети мы ищем седловую точку лагранжиана задачи выпуклой оптимизации. Данная задача специально подобрана так, чтобы её решение было эквивалентно решению исходной минимаксной задачи. Нами ожидается, что это поможет избежать проблемы коллапса моды и приведет к улучшению работы сети.

1.1. Обзор литературы

Впервые GAN были предложены в статье [1]. Практическая польза GAN заключается в возможности создания реалистичных изображений, текстов, музыки [3]. Тем не менее, они обладают рядом недостатков, главными из них являются сложность обучения и анализ сходимости. Общий обзор решений этой проблемы дается в статье [2], авторы которой отдельно фокусируются на рассмотрении задачи с оптимизационной точки зрения. В этой работе рассматриваются три типа решений. Первый тип – это использование непараметрических моделей генератора и дискриминатора ([4]). Второй тип – использование так называемой «развернутой оптимизации», при которой дискриминатор остается оптимальным или почти оптимальным в процессе генерации ([5]). Третий тип – рассмотрение оптимизации сетей через прямо-двойственную задачу. Данная работа опирается на статью [6]. Наша задача заключается в проведении модифицированных экспериментов, проверяющих эффективность прямо-двойственного подхода.

2. Постановка задачи

Мы предлагаем альтернативный подход к постановке задачи: сформулировать задачу как нахождение седловой точки лагранжиана задачи выпуклой оптимизации. Предположим, что исходные данные и сгенерированные принадлежат конечному множеству $\{x_1, \ldots, x_n\}$ заданного размера n. Реальные данные имеют конечный размер, поэтому нам интересен именно этот случай. Поставим следующую задачу оптимизации

$$\begin{aligned} &\max \sum_{i=1}^{n} p_d(x_i) \log(D_i) \\ &s.t. \log(1 - D_i) \geq \log(1/2),\ i = \overline{1, n} \\ &\mathbf{D} \in \mathcal{D}, \end{aligned}$$

где $\mathcal{D}$ некоторое выпуклое множество. Прямые переменные в ней: $\mathbf{D} = (D_1, \ldots, D_n)$, где $D_i = D(x_i)$. Пусть $p_g = (p_g(x_1), \ldots, p_g(x_n))$, где $p_g(x_i)$ — i-ая двойственная переменная. Функция Лагранжа в этом случае запишется как

$$L(\mathbf{D}, p_g) = \sum_{i=1}^{n} p_d(x_i) \log(D_i) + \sum_{i=1}^{n} p_g(x_i) \log(2(1 - D_i)), \tag{2}$$

где $\mathbf{D} \in \mathcal{D}$. В случае $\mathcal{D} = \{\mathbf{D} : 0 \leq D_i \leq 1, \forall i\}$ поиск седловой точки лагранжиана в точности эквивалентен решению задачи (1). Это свойство позволяет переработать прямо-двойственный субградиентый метод для обновления значений $D(x)$ и $p_g(x)$, так как они будут сходиться к седловой точке.

3. Методы

В этом разделе мы представим псевдокод предложенного метода обучения GAN (**Algorithm 1**) и дадим его описание. Обновление параметра дискриминатора в предложенном алгоритме такое же , как и в стандартном методе обучения. Отличие от стандартного метода заключается в процедуре обновления генератора.

Значение функции сгенерированного распределения в точке $\mathbf{x}$ есть $p_g(\mathbf{x}) = \frac{1}{m}\sum\limits_{i=1}^{m} \mathcal{I}(G(z_j) = \mathbf{x})$, где $\mathcal{I}(\cdot)$ – индикаторная функция. В алгоритме будет использована гладкая аппроксимация индикаторной функции

$$k_\sigma(\mathbf{x}) = \exp(-\frac{\|x\|_2^2}{\sigma^2}), \tag{3}$$

где σ – положительный гиперпараметр. При $\sigma \to 0$, $k_\sigma(G(z_j) - \mathbf{x}) \to \mathcal{I}(G(z_j) = \mathbf{x})$. Перед обновлением параметра генератора производится обновление генерируемого распределения (6). Обновление двойственной переменной, в данном случае $p_g(x_i)$, в прямо-двойственном алгоритме производится по правилу

$$\tilde{p}_g(\mathbf{x}_i) = p_g(\mathbf{x}_i) - \alpha \frac{\partial L(\mathbf{D}, \mathbf{p_g})}{\partial p_g(\mathbf{x}_i)} = p_g(\mathbf{x}_i) - \alpha f_1\left(D(\mathbf{x}_i)\right)$$

Добавление второго слагаемого в функцию потерь (7) заставляет GAN генерировать образцы из распределения все более и более близкого к истинному распределению данных.

Algorithm 1: Training GAN via Primal-Dual Subgradient Methods

---

Initialization: : Choose the objective function $f_0(\cdot)$ and constraint function $f1(\cdot)$ according to the GAN realization. For the original GAN based on Jensen-Shannon divergence, $f_0(D) = \log(D)$ and $f_1(D) = \log(2(1-D))$.

while *the stopping criterion is not met* **do**

Sample minibatch m_1 data samples $x_1, \ldots, x_{m_1}$

Sample minibatch m_2 noise samples $z_1, \ldots, z_{m_2}$.;

for $k = 1, \ldots, k_0$ **do**

Update the discriminator parameters with gradient ascent:

$$\nabla_{\theta_d} \left[\frac{1}{m_1} \sum_{i=1}^{m_1} f_0\left(D(\mathbf{x}_i)\right) + \frac{1}{m_2} \sum_{j=1}^{m_2} f_1\left(D(G(\mathbf{z}_j))\right) \right]. \tag{4}$$

end

Update the target generated distribution as:

$$\tilde{p}_g(\mathbf{x}_i) = p_g(\mathbf{x}_i) - \alpha f_1\left(D(\mathbf{x}_i)\right), i = 1, \ldots, m_1, \tag{5}$$

where α is some step size and

$$p_g(\mathbf{x}_i) = \frac{1}{m_2} \sum_{j=1}^{m_2} k_\sigma\left(G(\mathbf{z}_j) - \mathbf{x}_i\right) \tag{6}$$

With $\tilde{p}_g(\mathbf{x}_i)$ fixed, update the generator parameters with gradient descent:

$$\nabla_{\theta_g} \left[\frac{1}{m_2} \sum_{j=1}^{m_2} f_1\left(D\left(G(\mathbf{z}_j)\right)\right) + \frac{1}{m_1} \sum_{i=1}^{m_1} \left(\tilde{p}_g(\mathbf{x}_i) - \frac{1}{m_2} \sum_{j=1}^{m_2} k_\sigma\left(G(\mathbf{z}_j) - \mathbf{x}_i\right) \right)^2 \right]. \tag{7}$$

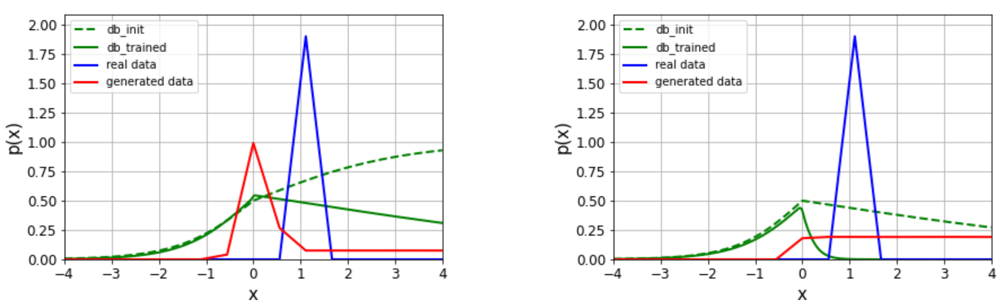
end

---

4. Эксперименты

4.1. Модельный эксперимент

В данном разделе приведён эксперимент, аналогичный описанному в п. 7.3 приложения к [6].В оригинальном эксперименте истинное распределение вырожденное: $p_d(x) = 1$ $(x = 1)$ (с вероятностью $p = 1$ элемент выборки x принимает значение 1). Мы модифицировали эксперимент, взяв истинные данные из распределения $x \sim \mathcal{N}(1, 10^{-3})$. Архитектура GAN для этого эксперимента приведена в Таблице 1. Параметры сети в прямо-двойственном методе обновляются с помощью ADAM ([6]). Результаты обучения GAN для стандартного и предложенного метода представлены на рис. 1. Код доступен в репозитории по ссылке.



a) b)

Рис. 1: Результаты обучения GAN в модельном эксперименте: a) для стандартного метода тренировки, b) для прямо-двойственного субградиентного метода.

	Генератор	Дискриминатор
Вход	$\mathbb{R}$	$\mathbb{R}$
Скрытый слой	32 нейрона + ReLU	32 нейрона + ReLU
Выход	$\mathbb{R}$	$\sigma(\mathbb{R})$

Таблица 1: Архитектура GAN в модельном эксперименте

4.2. Эксперимент с MNIST

В данном разделе приведены результаты тестирования предложенного метода на датасете MNIST. Размер обучающей выборки – 60 000 сэмплов.

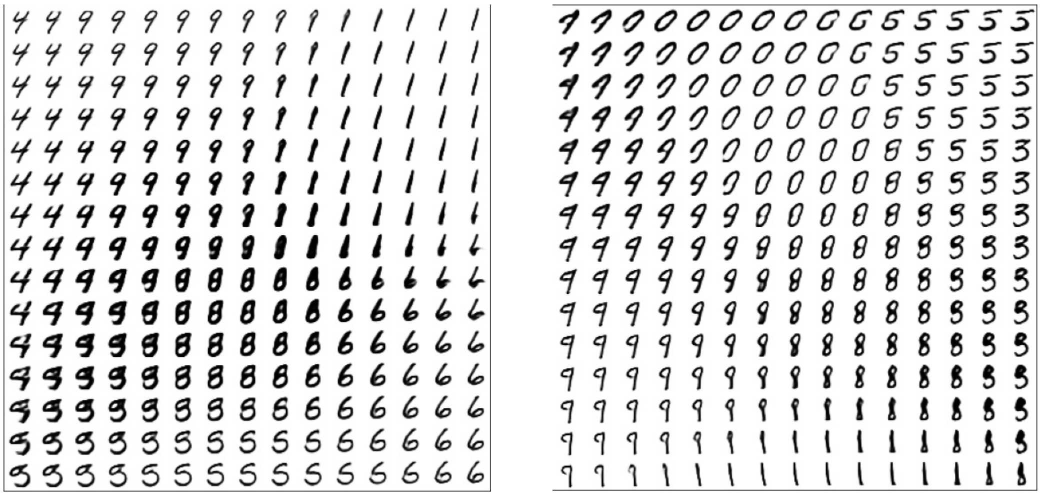
На Рис. 2 представлены изображения, которые генерирует GAN в результате обучения.В результате обучения обоими методами GAN генерирует сопоставимые по качеству изображения. Архитектура нейросетей также представлена в Таблице 2.

	Дискриминатор	Генератор
Вход	Картинка $(28 \times 28 \times 1)$	Массив из $\mathbb{R}^4$
Слой 1	Conv2D $(14 \times 14 \times 128)$	Dense 3136
Слой 2	Leaky ReLU	Dropout 3136
Слой 3	Dropout	Reshape $(7 \times 7 \times 64)$
Слой 4	MaxPooling 2D $(7 \times 7 \times 128)$	UpSampling2D $(14 \times 14 \times 64)$
Слой 5	Conv2D $(7 \times 7 \times 128)$	Conv2D $(14 \times 14 \times 64)$
Слой 6	Leaky ReLU	Dropout $(14 \times 14 \times 64)$
Слой 7	Dropout	Conv2D $(14 \times 14 \times 32)$
Слой 8	Flatten 6272	Dropout $(14 \times 14 \times 32)$
Слой 9	-	UpSampling2D $(28 \times 28 \times 32)$
Выход	Dense $(\{0, 1\})$	Conv2D $(28 \times 28 \times 1)$

Таблица 2: Архитектура GAN в эксперименте MNIST

5. Заключение

В данной работе проведено тестирование предложенного в статье [6] прямо-двойственного метода на задаче, в которой истинные данные распределены как $x \sim \mathcal{N}(1, 10^{-3})$. Как видно из Рис. 1, в результате обучения с помощью прямо-двойственного субградиентного метода GAN начинает генерировать данные, распределение которых значительно отличается от истинного. Мы пока не знаем, как интерпретировать полученный результат. По какой-то причине субградиентный прямо-двойственный метод в случае гауссовского распределения $(x \sim \mathcal{N}(1, 10^{-3}))$ оказывается менее предпочтительным с точки зрения качества в сравнении с градиентным спуском. Нашей возможной дальнейшей задачей может быть изучение границ применимости предложенного в статье [6] метода. Также интересен вопрос выбора оптимизатора, который мы в данной работе не затрагиваем. Важным направлением дальнейшней работы является попытка дать интерпретацию Алгоритму 1 (см. раздел 3).



a) b)

Рис. 2: Результаты обучения GAN в случае тренировки a) стандартным методом, b) прямо-двойственным субградиентным методом.

Список литературы

[1] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial nets, in: Advances in neural information processing systems, 2014, pp. 2672–2680.

[2] S. Lu, R. Singh, X. Chen, Y. Chen, M. Hong, Understand the dynamics of gans via primal-dual optimization (2018).

[3] J. Engel, K. K. Agrawal, S. Chen, I. , Gulrajani, C. Donahue, A. Roberts, Adversarial neural audio synthesis (2019).

[4] M. Arjovsky, S. Chintala, L. Bottou, Wasserstein gan (2017).

[5] L. Metz, B. Poole, D. Pfau, J. Sohl-Dickstein, Unrolled generative adversarial network (2017).

[6] X. Chen, J. Wang, H. Ge, Training generative adversarial networks via primal-dual subgradient methods: A lagrangian perspective on gan, arXiv preprint arXiv:1802.01765 (2018).

[6] [2] [1]