Sequential Line Search for Generative Adversarial Networks

Masahiro Kazama
Recruit Technologies Co.,Ltd.
Tokyo, Japan
masahiro_kazama@r.recruit.co.jp

Viviane Takahashi
Recruit Technologies Co.,Ltd.
Tokyo, Japan
viviane@r.recruit.co.jp

Abstract

Generative Adversarial Networks (GAN) is a popular method to generate realistic images. The method is used for creating new designs and artwork. However, it takes time to generate the desired image because the relationship between the GAN's input and output is unclear. The user generally has to try several input vectors into the GAN in order to obtain the desired output image. In this paper, we propose a method that can efficiently generate the desired image by applying sequential line search to the GAN. Using this technique, the user only needs to manipulate a single slider a few times in order to obtain the desired output image. The utility of our method is experimented through crowdsourced experiments.

1 Introduction

Generative Adversarial Networks (GAN) [1] has recently gained attention because of their capability of generating very realistic images. GAN is widely used for creative tasks like creating new designs and artwork. The problem in using GAN is the uncertainty of the relationship between input and output. In a typical system, the input is a d-dimensional vector and the output is an image. We propose the method of applying sequential line search to GAN, in which the user merely needs to manipulate a single slider to obtain the desired image, as described in Figure 1. This allows the user to create the desired image quickly. Users can use our method to create artwork.

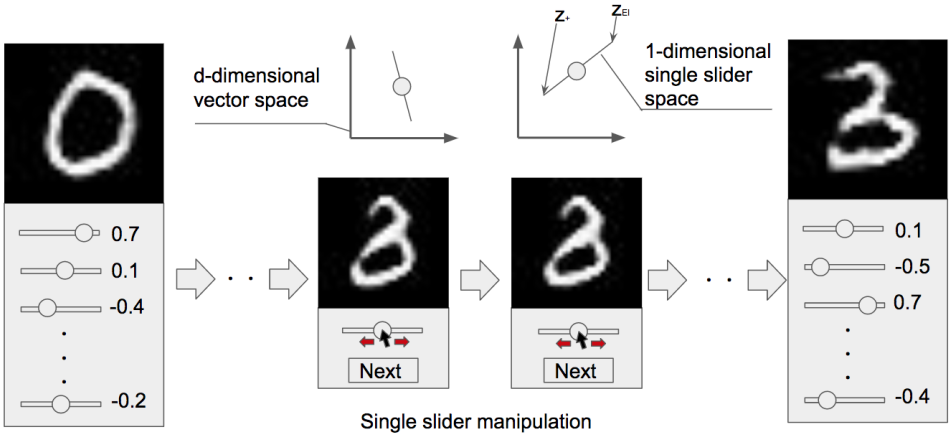

Figure 1: Summary of the proposed method. Users manipulate a single slider several times in order to generate the desired image (i.e., to find an optimal d-dimensional input vector).

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

2 Related works

Koyama et al. [2] proposed the "sequential line search" method to improve aesthetic characteristics of images. Using this method, users are required to manipulate a single slider several times in order to get the desired image appearance, instead of manipulating many parameters like "contrast" or "brightness". In this paper, we apply the sequential line search method to GAN.

3 Method

The purpose of our proposed method is to identify the best input vector to generate the desired image. In our proposed method, the user moves a single slider a few times in order to obtain the output image. The sequential line search consists of 2 steps. 1) In the first step, the user manipulates a slider. The image changes dynamically as the slider is moved. The user should stop the manipulation at the point when the generated image is most similar to the user's desired image. 2) We generate a new single slider according to the user's previous choice by using the Bayesian optimization technique. We have the information of the user's past action, which indicates that the image the user chose is preferable over other images (i.e., other points on the slider). With this information, Bayesian optimization is used to identify which points should be searched next and create a new single slider. Repeating this process, users can get the desired output. The detailed process is described in the supplementary material.

4 Experiments

We evaluate our proposed method by crowdsourced experiments. To examine our proposed method's utility, we ask users to choose the image they judge to be most similar to a predefined target image for 5 trials. After 5 trials, we calculate the Euclidean distance between the final obtained image and the target image. We use this distance as an evaluation metric. For the experiments, we use the MNIST dataset, which consists of handwritten digit data. We train the GAN for the MNIST dataset. Our method is then compared with "Random" methods where the single slider is generated randomly.

5 Results and Discussion

The result of our experiment is depicted in Table 1. We calculate the Euclidean distance between the finally obtained image and the target image. We count the number of images under each threshold. Hence, our proposed method can generate images which are fairly close to what the user has in mind and we believe artists could use this method as a new creation tool.

Table 1: Results of the crowdsourced experiments. We calculate the Euclidean distance between the finally obtained image and the target image. We count the number of images under each threshold.

	Random method	Proposed method
Lower than 6.0	0	1
Lower than 7.0	1	3
Lower than 8.0	2	5
Lower than 9.0	13	20
Larger than 9.0	35	28

References

[1] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[2] Yuki Koyama, Issei Sato, Daisuke Sakamoto, and Takeo Igarashi. Sequential line search for efficient visual design optimization by crowds. *ACM Transactions on Graphics (TOG)*, 36(4):48, 2017.

A Supplementary material

A.1 Detail of method

We use the GAN generator $G(\boldsymbol{z})$ as the image generator where $\boldsymbol{z} \in R^d$ is the d-dimensional input vector and $\boldsymbol{x} = G(\boldsymbol{z})$ is the output image. Our task is to find the optimal $\boldsymbol{z}$ that generates the desired image $\boldsymbol{x}$. The sequential line search process for GAN is as follows.

Firstly we choose 2 random input vectors $\boldsymbol{z}_l^1, \boldsymbol{z}_r^1$. Next we generate a single slider whose left endpoint is $\boldsymbol{z}_l^1$ and right end point is $\boldsymbol{z}_r^1$. An interval point in the slider is represented as $\boldsymbol{z}_m^1 = a\boldsymbol{z}_l^1 + (1-a)\boldsymbol{z}_r^1$, where $0 \leq a \leq 1$.

When the user manipulates the slider, the image $G(\boldsymbol{z_m})$ is dynamically generated. The user should stop the slider at the point where the output image is the most similar to the desired image. The point and image are represented as $\boldsymbol{z}_+^1$ and $\boldsymbol{x}_+^1 = G(\boldsymbol{z}_+^1)$ respectively.

After the user chooses a point in the slider, a new single slider will be generated. The left endpoint $\boldsymbol{z}_l^2$ in the new slider is the point chosen by the user at the previous trial (i.e., $\boldsymbol{z}_l^2 = \boldsymbol{z}_+^1$). The right endpoint will be decided from the Bradley-Terry-Luce (BTL) model and Bayesian optimization.

A.1.1 BTL model

We use the BTL model to model the user's preference toward images.

When the user chooses the image $\boldsymbol{x_j} = G(\boldsymbol{z_j})$ from the set of images $\boldsymbol{S} = \{\boldsymbol{x_i} | \boldsymbol{x_i} = G(\boldsymbol{z_i})\}_{i=1}^m$, we represent it as $\boldsymbol{z_j} \succ \boldsymbol{S} \setminus \boldsymbol{z_j}$. The likelihood of the situation is modeled as

$$P(\boldsymbol{z_j} \succ \boldsymbol{S} \setminus \boldsymbol{z_j}) = \frac{\exp(q_j/s)}{\sum_{i=1}^m \exp(q_i/s)}, \tag{1}$$

where q_i is the user's preference value toward image $\boldsymbol{x_i}$ and $\boldsymbol{z_i}$ and s is the scaling parameter. In our single slider situation, we can consider a slider as the set of a finite number of points. In the experiments, we noticed that the results were not altered by a change in the number of points. So we choose the minimal representation where we just use the endpoints $\boldsymbol{z}_r, \boldsymbol{z}_l$ and the chosen point $\boldsymbol{z}_+$, thus, $\boldsymbol{S} = \{\boldsymbol{z}_r, \boldsymbol{z}_l, \boldsymbol{z}_+\}$. Therefore, at each turn n we can obtain the information $\boldsymbol{D}^n = \{\boldsymbol{z_+^n} \succ \boldsymbol{z_l^n}, \boldsymbol{z_r^n}\}$. Using the information $\boldsymbol{D} = [\boldsymbol{D}^1, \boldsymbol{D}^2, ..., \boldsymbol{D}^n]$, we will estimate the preference values $\boldsymbol{q} = [q_l^1, q_r^1, q_+^1, q_r^2, ...]$. The estimation method is explained in the next section.

Once we have the estimated preference values $\boldsymbol{q}$, we can use this for Bayesian optimization, being able to calculate which points should be searched in the next trial.

A.1.2 Bayesian Optimization

Bayesian optimization used for optimizing the black box function $f(\boldsymbol{z})$ is as follows:

$$\max_{\boldsymbol{z} \in \mathcal{Z}} f(\boldsymbol{z}). \tag{2}$$

Usually, the derivative $f^{'}(\boldsymbol{z})$ cannot be easily obtained and evaluating each point of $f(\boldsymbol{z})$ is costly. In this scenario, Bayesian optimization is useful once it indicates which point should be evaluated next based on the current n observed values. Therefore, optimal points can be efficiently found after processing a few observations. f is usually modelled as a Gaussian process:

$$f \sim GP(m_f(\boldsymbol{z}), k_f(\boldsymbol{z}, \boldsymbol{z}^{'})), \tag{3}$$

where m_f is a mean function. We assume that $m_f = \mathbf{0}$ because we do not have any domain specific knowledge. k_f is a kernel function:

$$k_f(\boldsymbol{z}, \boldsymbol{z}^{'}) = \theta_{d+1} \exp\left\{ -\frac{1}{2} \sum_{i=1}^{d} \frac{(z_i - z_i^{'})^2}{\theta_i^2} \right\}, \tag{4}$$

where $\boldsymbol{\theta}$ is a hyperparameter, which is determined by MAP estimation, given the observed data $\boldsymbol{D}$. However, in this paper we predetermined the hyperparameter in advance to reduce computational complexity.

The predictive distribution of $f(\boldsymbol{z})$ is a Gaussian distribution and can be calculated analytically. $f(\boldsymbol{z})$ is used to construct the acquisition function $a(\boldsymbol{z})$, which determines the next point that should be evaluated. Although there are several kinds of acquisition functions, the expected improvement (EI) is commonly used. The EI function is defined as

$$a(\boldsymbol{z}) = E[\max\{f(\boldsymbol{z}) - f^+, 0\}], \tag{5}$$

where f^+ is the best value among current observed data. We should evaluate $\boldsymbol{z}$ as the point that maximizes $a(\boldsymbol{z})$.

In the sequential line search, we use the estimated preference value q towards $\boldsymbol{z}$ as the $f(\boldsymbol{z})$ and calculate the acquisition function $a(\boldsymbol{z})$. Then we obtain the next $\boldsymbol{z}_{EI}$. Finally, we set $\boldsymbol{z}_{EI}$ as the right endpoint of the slider.

To perform the search, we need to estimate the preference value $\boldsymbol{q}$. The parameter can be estimated through MAP estimation:

$$\begin{aligned}
\boldsymbol{q}^{MAP} &= \arg\max_{\boldsymbol{q}} p(\boldsymbol{q}|\boldsymbol{D}, \boldsymbol{\theta}) && (6)\\
&= \arg\max_{\boldsymbol{q}} p(\boldsymbol{D}|\boldsymbol{q}, \boldsymbol{\theta}) p(\boldsymbol{q}|\boldsymbol{\theta}) && (7)\\
&= \arg\max_{\boldsymbol{q}} p(\boldsymbol{D}|\boldsymbol{q}) p(\boldsymbol{q}|\boldsymbol{\theta}). && (8)
\end{aligned}$$

The last equation is derived from the idea that, given the preference value $\boldsymbol{q}$, $\boldsymbol{D}$ and $\boldsymbol{\theta}$ are conditionally independent.

The conditional probability $p(\boldsymbol{D}|\boldsymbol{q})$ is calculated by the BTL model as

$$p(\boldsymbol{D}|\boldsymbol{q}) = \prod_n p(\boldsymbol{z}_+^n \succ \boldsymbol{z}_r^n, \boldsymbol{z}_l^n). \tag{9}$$

The conditional probability $p(\boldsymbol{q}|\boldsymbol{\theta})$ is calculated with the GP as

$$p(\boldsymbol{q}|\boldsymbol{\theta}) = N(\boldsymbol{q}; \boldsymbol{0}, \boldsymbol{K}), \tag{10}$$

where $\boldsymbol{0}$ is the mean function and $\boldsymbol{K}$ is the covariance matrix.

The summary of sequential line search for GAN is bellow.

Algorithm 1 Sequential line search for GAN

for $n = 1, 2, ...$ **do**
  $\boldsymbol{q}^{MAP} = compute\_MAP\_estimate(\boldsymbol{D}^n)$
  $\boldsymbol{z}_{EI} = \arg\max_{\boldsymbol{z}} a(\boldsymbol{z})$
  Create a new single slider with the endopoints $\boldsymbol{z}_l^n = \boldsymbol{z}_+^{n-1}, \boldsymbol{z}_r^n = \boldsymbol{z}_{EI}$
  User manipulates the slider
  $\boldsymbol{D}^{n+1} = \boldsymbol{D}^n \cup \{\boldsymbol{z}_+^n \succ \boldsymbol{z}_l^n, \boldsymbol{z}_r^n\}$
end for

A.2 Detail of experiments

We use the GAN generator $G(\boldsymbol{z})$ and train it for the MNIST dataset. We set the dimension of input vector $\boldsymbol{z}$ as 5 and the dimension of the output image as 28×28. We set the hyperparameter of Bayesian optimization as $\theta_i = 0.5 (i = 1, ..., d + 1)$ and the scaling parameter of the BTL model as $s = 1$. We use crowdsourcing to compare our method with the random method. The summary of the random and proposed methods is described in Table 2.

We used Amazon Mechanical Turk and 48 workers performed the task. Our task is described in Figure 2.

Table 2: Comparison of random and proposed method.

	Random	Proposed
N=1	Two endpoints are randomly chosen	Two endpoints are randomly chosen
N=2,3,..	Left endpoint is the previously chosen one Right endpoint is randomly chosen	Left endpoint is the previously chosen one Right endpoint is z_{EI}

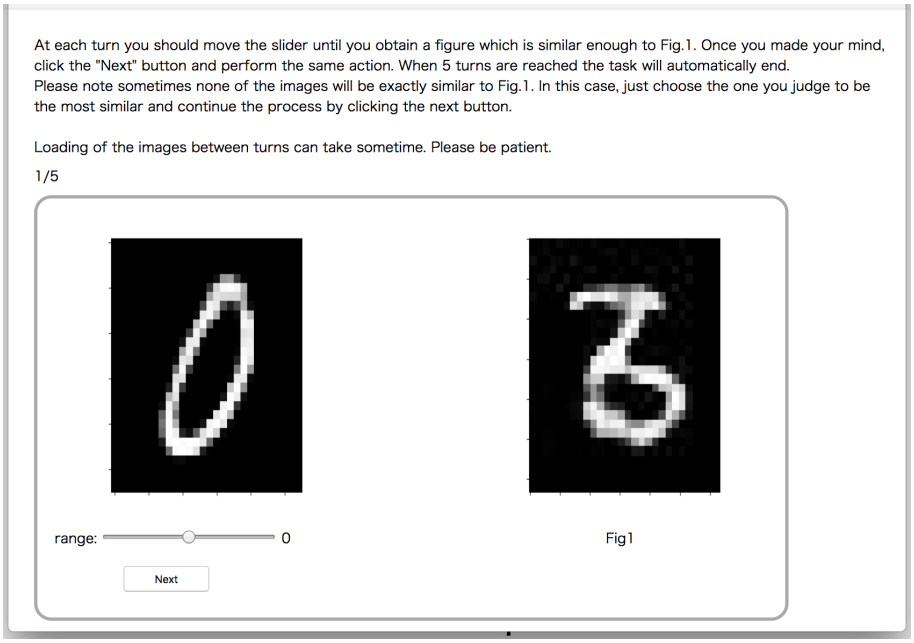

Figure 2: Crowdsourced experiment.

A.3 Detail of results

Example of one user's trial on our proposed method is described in Figure 3. We can observe that the images generated are similar to the target image.

Figure 3: Example of one user's trial on our proposed method.