Final Report for Structural Machine Learning Models and Their Applications

林子涵

Department of Applied Mathematics
National Chung Hsing University

June 21, 2021

Outline

The Numerics of GANs

Algorithm

Algorithm 1 Simultaneous Gradient Ascent (SimGA)

while not converged **do**
  $v_\phi \leftarrow \nabla_\phi f(\theta, \phi)$
  $v_\theta \leftarrow \nabla_\theta g(\theta, \phi)$
  $\phi \leftarrow \phi + h v_\phi$
  $\theta \leftarrow \theta + h v_\theta$
end while

Algorithm 2 Consensus optimization

while not converged **do**
  $v_\phi \leftarrow \nabla_\phi (f(\theta, \phi) - \gamma L(\theta, \phi))$
  $v_\theta \leftarrow \nabla_\theta (g(\theta, \phi) - \gamma L(\theta, \phi))$
  $\phi \leftarrow \phi + h v_\phi$
  $\theta \leftarrow \theta + h v_\theta$
end while

Comparison

	Old method	New method
vector field	$v(\phi,\theta)=\begin{pmatrix}\nabla_\phi f(\phi,\theta)\\ \nabla_\theta g(\phi,\theta)\end{pmatrix}$	$w(\phi,\theta)=v(\phi,\theta)-\gamma\nabla L(\phi,\theta)$
Jacobian	$v'(\phi,\theta)=\begin{pmatrix}\nabla^2_\phi f(\phi,\theta) & \nabla_{\phi\theta} f(\phi,\theta)\\ \nabla_{\theta\phi} g(\phi,\theta) & \nabla^2_\theta g(\phi,\theta)\end{pmatrix}$	$w'(\phi,\theta)=v'(\phi,\theta)-\gamma v'(\phi,\theta)^T v'(\phi,\theta)$
Numerics form	$F(\phi,\theta)=\begin{pmatrix}\phi\\ \theta\end{pmatrix}+hv(\phi,\theta)$	$F(\phi,\theta)=\begin{pmatrix}\phi\\ \theta\end{pmatrix}+h\left(I-\gamma v'(\phi,\theta)^T\right)v(\phi,\theta)$

where $L(\phi,\theta)=\frac{1}{2}\|v(\phi,\theta)\|^2$,$\gamma>0$ and $h>0$.

Objective function and Constrain of h

Our objective(loss) function is:

$$\min_{\phi} \max_{\theta} g(\phi, \theta) \tag{1}$$

where $f = -g$ when two player game is a zero-sum game.

Lemma (4)

Assume that $A \in \mathbb{R}^{n \times n}$ only has eigenvalues with negative real-part and let $h > 0$. Then the eigenvalues of the matrix $I + hA$ lie in the unit ball if and only if

$$h < \frac{1}{|\Re(\lambda)|} \frac{2}{1 + \left(\frac{\Im(\lambda)}{\Re(\lambda)}\right)^2} \tag{2}$$

for all eigenvalues λ of A.

Why the algorithm will converge?

Proposition (3)

Let $F : \Omega \to \Omega$ be a continuously differential function on an open subset Ω of $\mathbb{R}^n$ and let $\bar{x} \in \Omega$ be so that

1. $F(\bar{x}) = \bar{x}$, *and*
2. *the absolute values of the eigenvalues of the Jacobian $F'(\bar{x})$ are all smaller than 1.*

Then there is an open neighborhood U of $\bar{x}$ so that for all $x_0 \in U$, the iterates $F^{(k)}(x_0)$ converge to $\bar{x}$. The rate of convergence is at least linear. More precisely, the error $||F^{(k)}(x_0) - \bar{x}||$ is in $\mathcal{O}(|\lambda_{max}|^k)$ for $k \to \infty$ where λ_{max} is the eigenvalue of $F'(\bar{x})$ with the largest absolute value.

Experiment in CIFAR-10

I have do some experiment in CIFAR-10 dataset, the loss of discriminator(generator) and the generate image is shown as below:

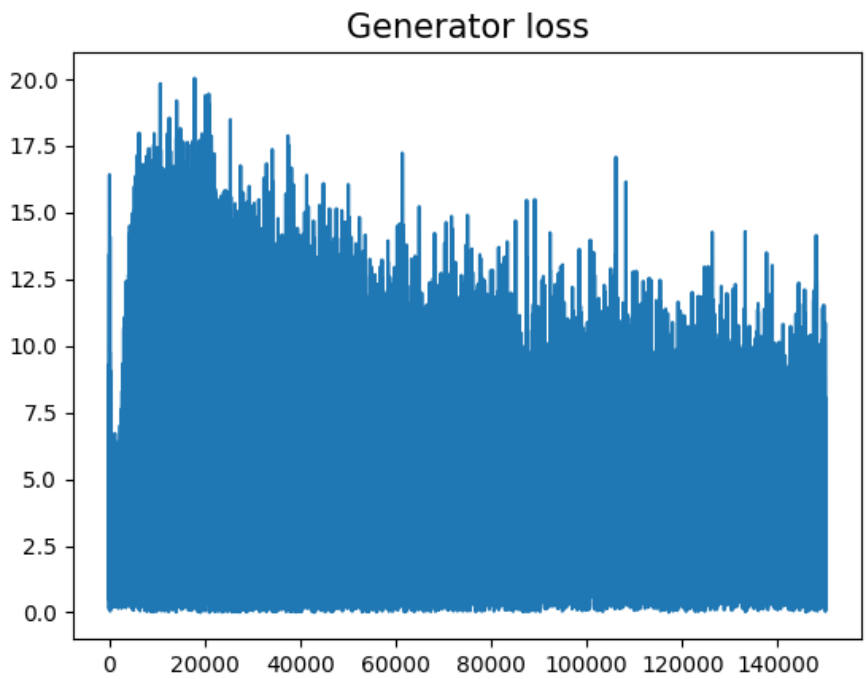


Figure: The generator loss of SimGD

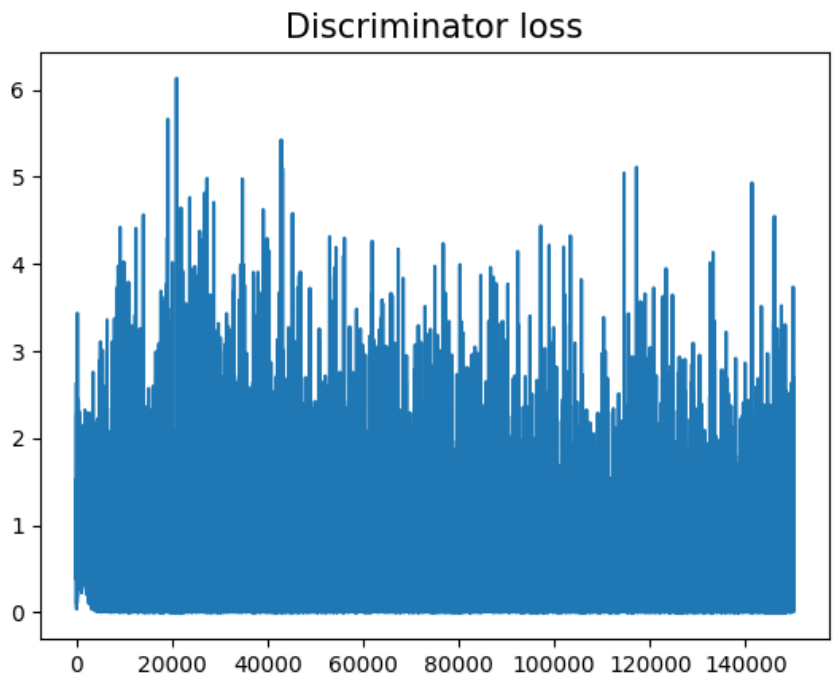


Figure: The discriminator loss of SimGD

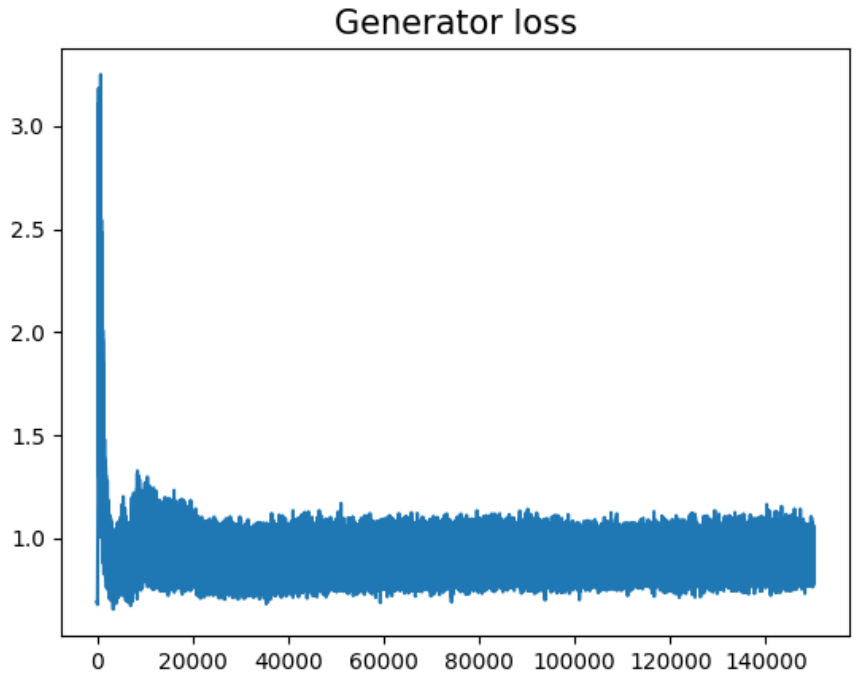


Figure: The discriminator loss of Consensus Optimization

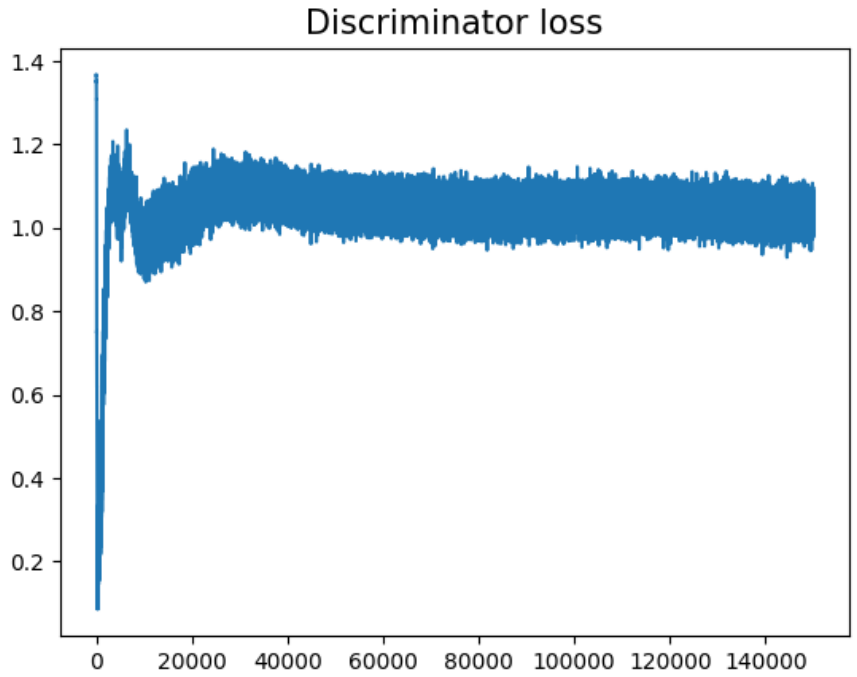


Figure: The discriminator loss of Consensus Optimization

Comparison of the Result

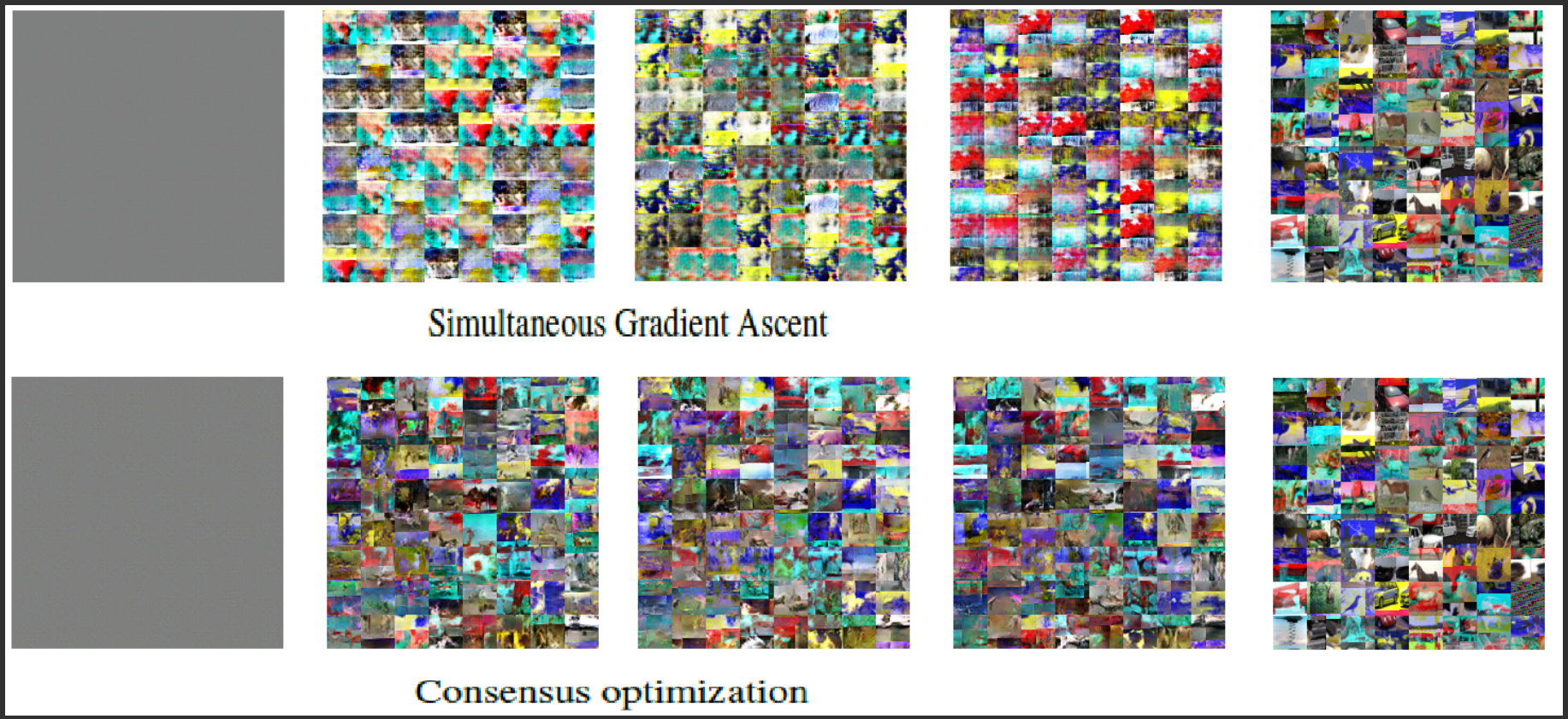


Figure: Comparison of Simultaneous Gradient Ascent and Consensus optimization on CIFAR-10 dataset.The generate images depect from left to right the resulting densities of the algorithm after 0, 50000, 100000 and 150000 iterations and the real image in last.

Local Saddle Point Optimization: A Curvature Exploitation Approach

Introduction

- Consider the problem to solve an optimization problem of the form

$$\min_{x \in \mathbb{R}^k} \max_{y \in \mathbb{R}^d} f(x, y) \tag{3}$$

  where f is smooth in x and y but not neccessarily convex in x or concave in y.
- This particular problem arises in many applications, such as generative adversarial networks (GAN).

Iteration Form

- First-order methods are commonly used to solve problem (1) as they have a cheap per-iteration cost and are therefore easily scalable. One particular method of choice is simultaneous gradient descent/ascent, which performs the following iterative updates,

$$(x_{t+1}, y_{t+1}) = (x_t, y_t) + \eta_t(-\nabla_x f_t, \nabla_y f_t); f_t := f(x_t, y_t) \quad (4)$$

  where $\eta_t > 0$ is a chosen step size($\eta_t = \eta$ if η is bounded constant).
- It is known that the gradient method is locally asymptotically stable[The numerics of GANs]; but stability alone is not sufficient to guarantee convergence to a locally optimal saddle point.
- Throughout the paper refer to a desired local saddle point as a local minimum in x and maximum in y.

Some definition

Definition

(Locally Optimal Saddles) Let us define a γ-neighbourhood around the point (x^*, y^*) as

$$\mathcal{K}_\gamma^* = \{(x, y) | \|x - x^*\| \leq \gamma, \|y - y^*\| \leq \gamma\} \tag{5}$$

with a sufficiently small $\gamma > 0$.

Definition (1)

The point (x^*, y^*) is locally optimal saddle point of the problem in (3) if

$$f(x^*, y) \leq f(x^*, y^*) \leq f(x, y^*) \tag{6}$$

hold for $\forall (x, y) \in \mathcal{K}_\gamma^*$.

Extreme Curvature Direction

- Let $\lambda_x(\lambda_y)$ be the minimum(maximum) eigenvalue and $\rho_x(\rho_y)$ be the Lipschitz constant of $\nabla_x^2 f(z)(\nabla_y^2 f(z))$ with its associated eigenvector $v_x(v_y)$. Then we define

$$v_z^{(-)} = \mathbb{1}_{\{\lambda_x<0\}} \frac{\lambda_x}{2\rho_x} \operatorname{sgn}(v_x^\top \nabla_x f(z)) v_x \tag{7}$$

$$v_z^{(+)} = \mathbb{1}_{\{\lambda_y>0\}} \frac{\lambda_y}{2\rho_y} \operatorname{sgn}(v_y^\top \nabla_y f(z)) v_y \tag{8}$$

  where sgn:$\mathbb{R} \to \{-1, 1\}$ is the sign function,
  $\mathbb{1}_A(x) = \begin{cases} 1 & \text{if } x \in A \\ 0 & \text{if } x \notin A \end{cases}$ and $\left\|\nabla_x^2 f(z) - \nabla_x^2 f(\tilde{z})\right\| \leq \rho_x \|z - \tilde{z}\| (\left\|\nabla_y^2 f(z) - \nabla_y^2 f(\tilde{z})\right\| \leq \rho_y \|z - \tilde{z}\|)$

- We define $v_z := (v_z^{(-)}, v_z^{(+)})$ as the extreme curvature direction at z.

Algorithm

- Using the extreme curvature direction, author modify the gradient steps as follows:

$$(x_{t+1}, y_{t+1}) = (x_t, y_t) + v_{z_t} + \eta(-\nabla_x f_t, \nabla_y f_t), f_t := f(x_t, y_t). \quad (9)$$

- This new update step is constructed by adding the extreme curvature direction to the gradient method of (4).
- This algorithm is called curvature exploitation for the saddle point problem(CESP) method and used to update (x, y).

Comparison

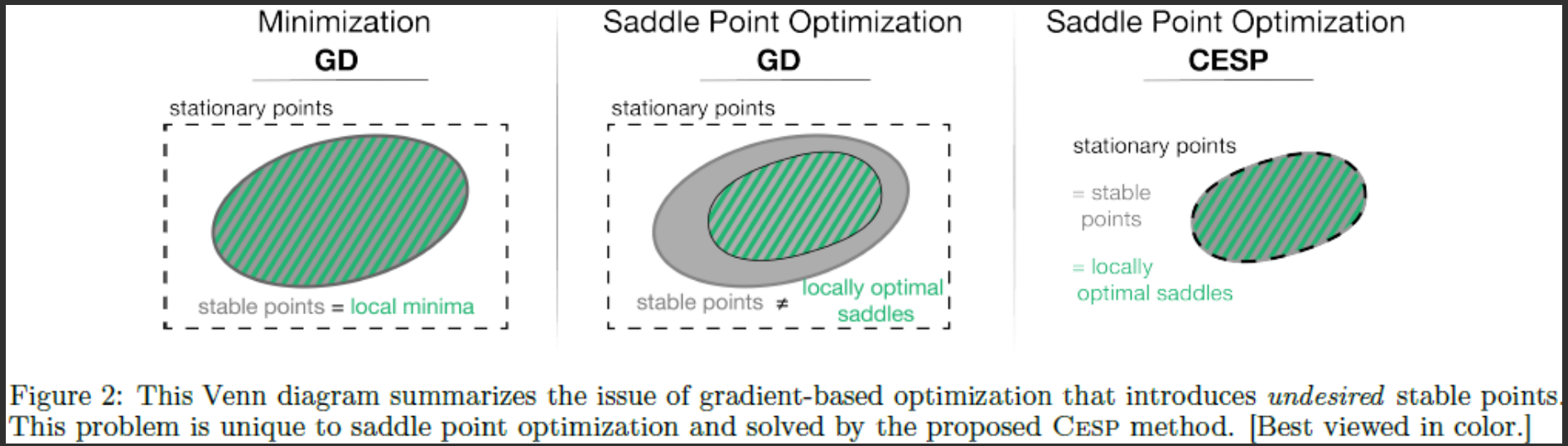


Figure 2: This Venn diagram summarizes the issue of gradient-based optimization that introduces *undesired* stable points. This problem is unique to saddle point optimization and solved by the proposed CESP method. [Best viewed in color.]

Experiment

Generative Adversarial Networks

- This experiment evaluates the performance of the CESP method for training a Generative Adversarial Network (GAN), which reduces to solving the saddle point problem

$$\min_{x} \max_{y} [f(x, y) = \mathbb{E}_{\theta \sim p_d} log(D_x(\theta)) + \mathbb{E}_{z \sim p_z} log(1 - D_x(G_y(z)))] \tag{10}$$

  where the functions $D : \mathbb{R}^n \to [0, 1]$ and $G : \mathbb{R}^m \to \mathbb{R}^n$ are represented by neural networks parameterized with the variables x and y, respectively.
- Here use the MNIST data set and a simple GAN architecture with 1 hidden layer and 100 units.

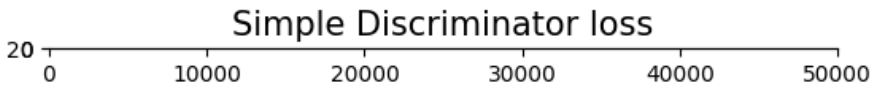
Network Parameter

Table 2: Parameters of the single-layer GAN model.

	Discriminator	Generator
Input Dimension	784	10
Hidden Layers	1	1
Hidden Units / Layer	100	100
Activation Function	Leaky ReLU	Leaky ReLU
Output Dimension	1	784
Batch Size	1000	
Learning Rate η	0.01	
Learning Rate $\alpha := \frac{1}{2\rho_{\mathbf{x}}} = \frac{1}{2\rho_{\mathbf{y}}}$	0.05	

Result

- I compare the result of SimGA and CESP.
- First, the loss of SimGA is shown as below:



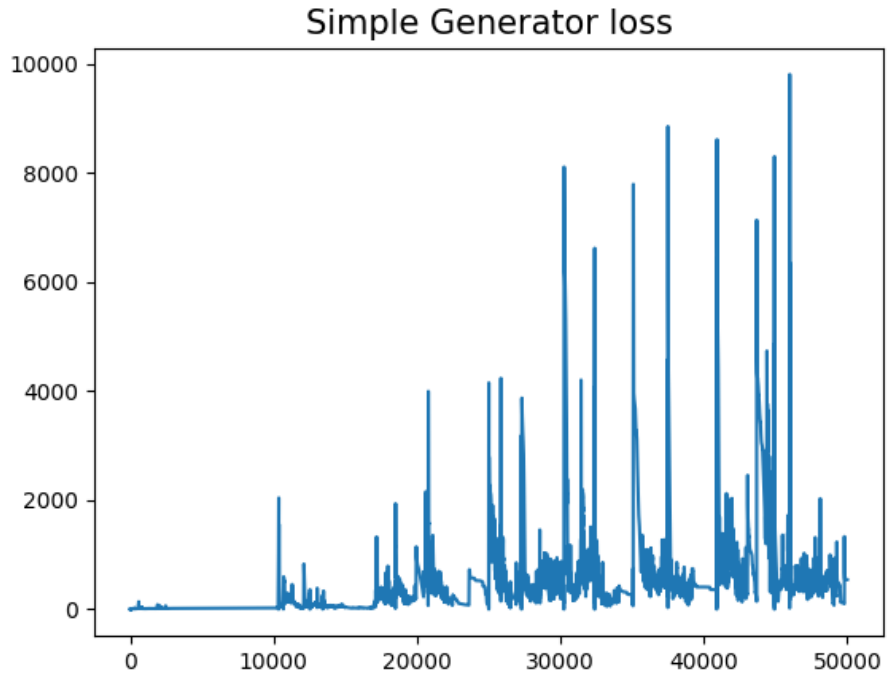


Figure: The discriminator loss of Simultaneous Gradient Ascent

Figure: The generator loss of Simultaneous Gradient Ascent

- Next, the loss of CESP is shown as below:

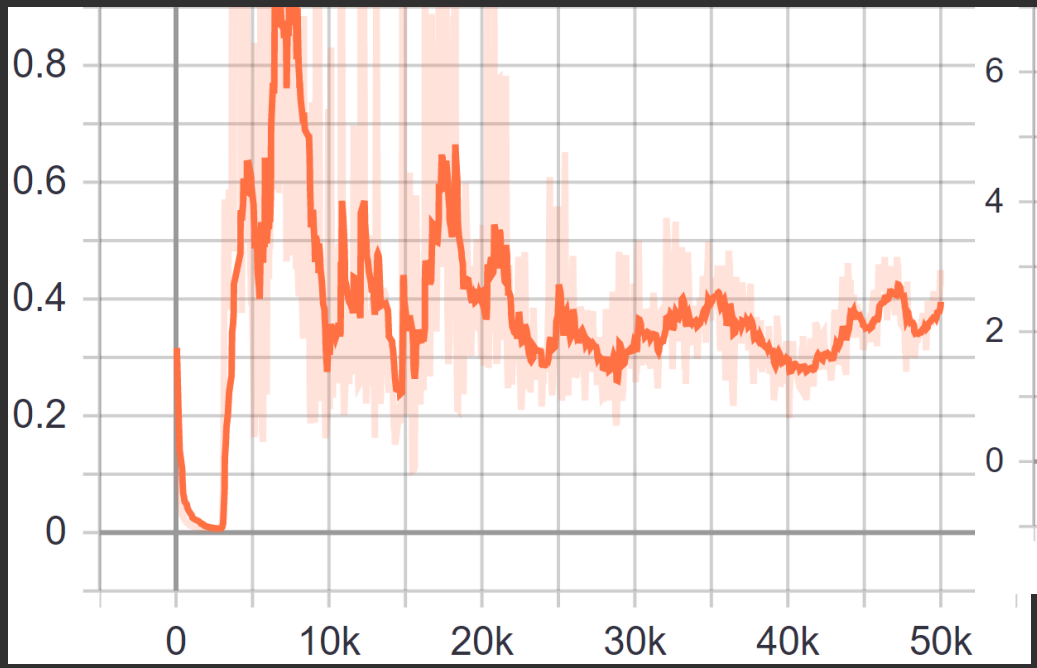


Figure: The discriminator loss of CESP

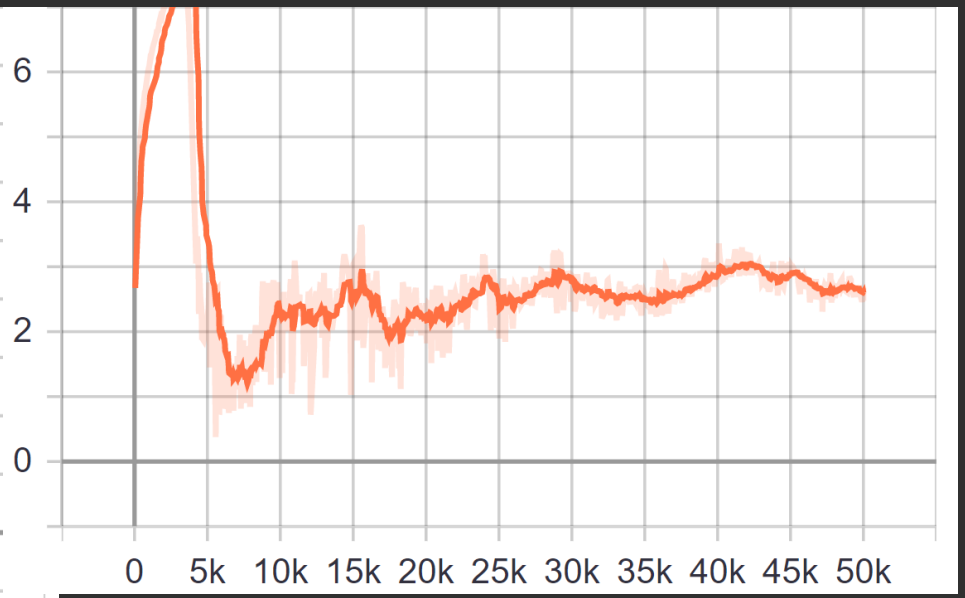


Figure: The generator loss of CESP

How escape?

- Review the algorithm
  $(x_{t+1}, y_{t+1}) = (x_t, y_t) + v_{z_t} + \eta(-\nabla_x f_t, \nabla_y f_t), f_t := f(x_t, y_t).$

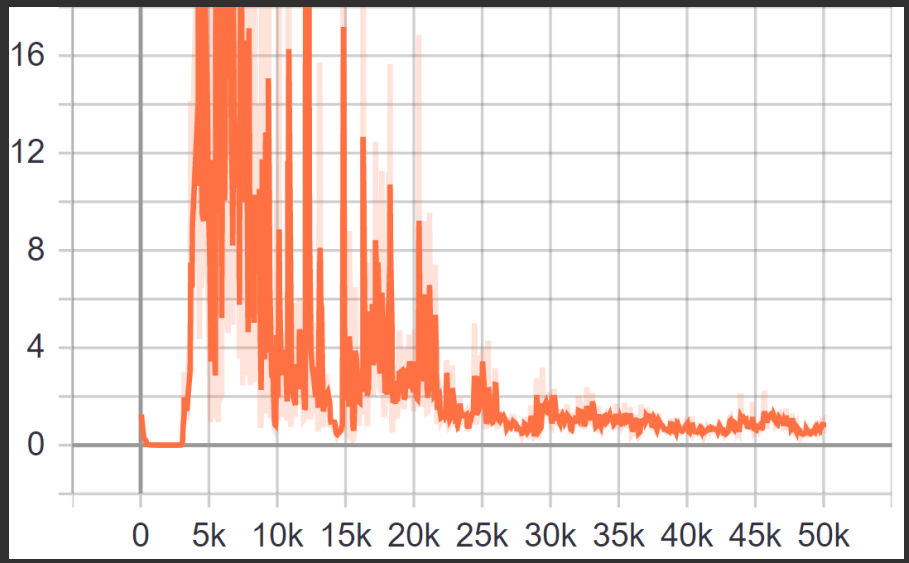


Figure: The gradient of discriminator in CESP

- As this figure above, we observe that CESP help the discriminator escape the undesired stable point.

Generate Image

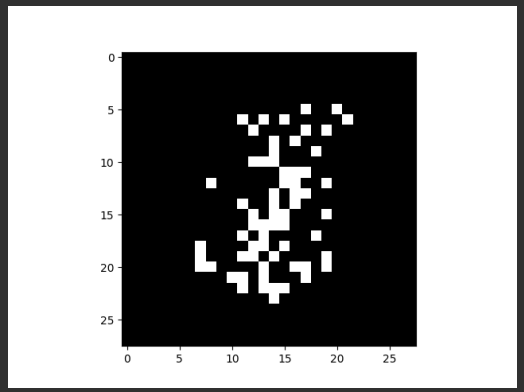


Figure: The image generated by SimGA

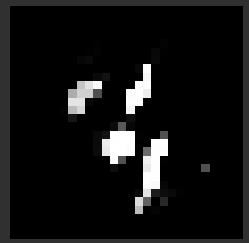


Figure: The image generated by CESP

Supplementary

How about mix up the method proposed in two paper?

Consider the following two-dimensional saddle point problem

$$\min_x \max_y \left[f(x,y) = 2x^2 + y^2 + 4xy + \frac{4}{3}y^3 - \frac{1}{4}y^4 \right] \tag{11}$$

with $x, y \in \mathbb{R}$. The critical points of the function, i.e. points for which $\nabla f(x,y) = 0$, are

$$z_0 = (0,0), z_1 = (-2-\sqrt{2}, 2+\sqrt{2}), z_2 = (-2+\sqrt{2}, 2-\sqrt{2}) \tag{12}$$

Evaluating the Hessians at the three critical points gives rise to the following three matrices:

$$H(z_0) = \begin{bmatrix} 4 & 4 \\ 4 & 2 \end{bmatrix}, H(z_1) = \begin{bmatrix} 4 & 4 \\ 4 & -4\sqrt{2} \end{bmatrix}, H(z_2) = \begin{bmatrix} 4 & 4 \\ 4 & 4\sqrt{2} \end{bmatrix} \tag{13}$$

Lemma

For zero-sum games, $v'(x)$ is negative (semi-)definite if and only if $\nabla^2_\phi f(\phi, \theta)$ is negative (semi-)definite and $\nabla^2_\theta f(\phi, \theta)$ is positive (semi-)definite.

Result

In [7]:
```
gd_list = [init_point]

eta = 0.01
iteration = 10000
x, y = init_point
for i in range(iteration):
    x, y = gd(x, y, eta)
    gd_list.append([x, y])

    if nabla_f_x(x, y)<1e-5 and nabla_f_x(x, y)>-1e-5:
        print(i)
        break
    elif nabla_f_y(x, y)<1e-5 and nabla_f_y(x, y)>-1e-5:
        print(i)
        break
```

868

In [8]:
```
cesp_list = [init_point]
x, y = init_point
for i in range(iteration):
    x, y = cesp(x, y, rho_x, rho_y, eta)
    cesp_list.append([x, y])
    if nabla_f_x(x, y)<1e-5 and nabla_f_x(x, y)>-1e-5:
        print(i)
        break
    elif nabla_f_y(x, y)<1e-5 and nabla_f_y(x, y)>-1e-5:
        print(i)
        break
```

288

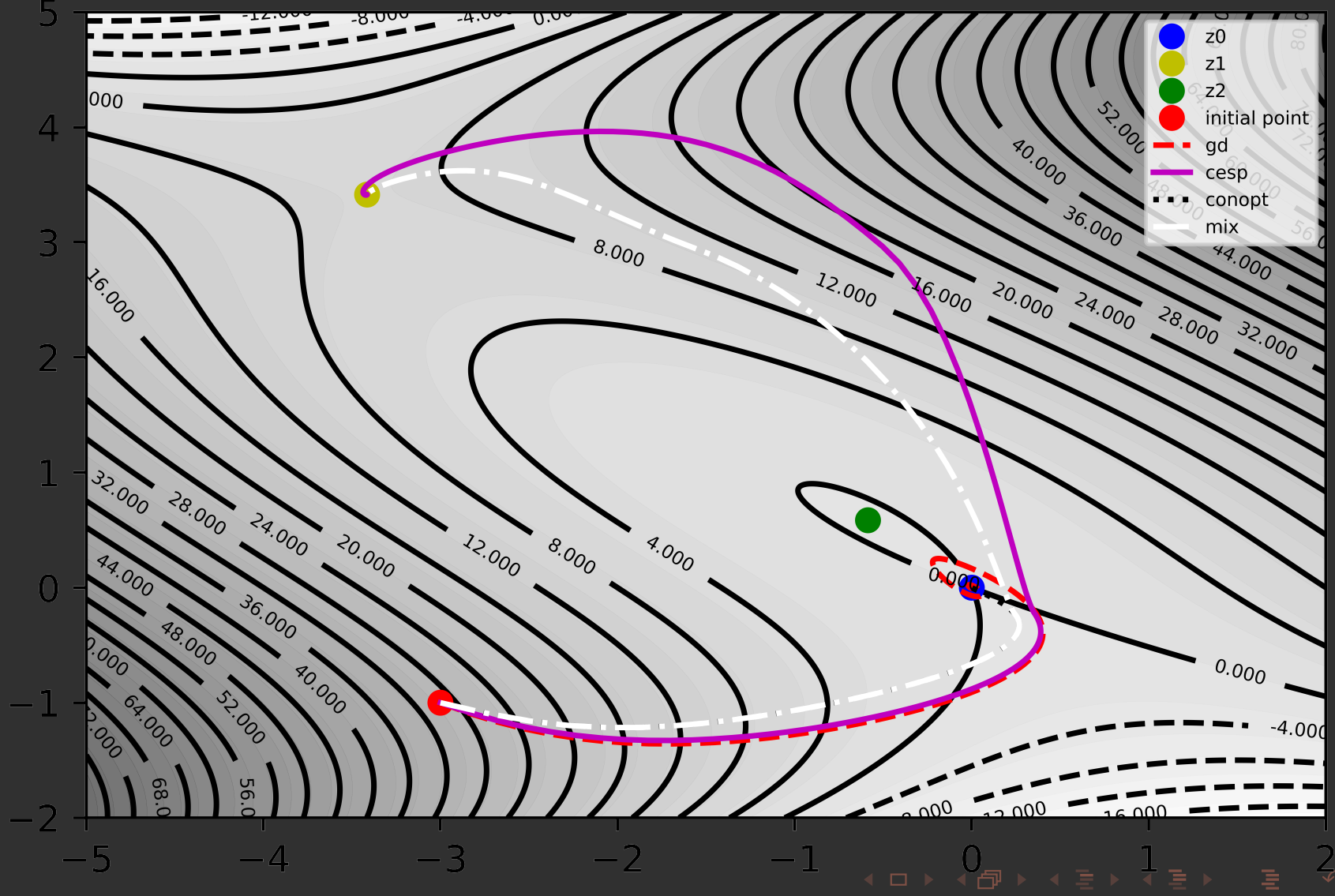
In [9]:
```
conopt_list = [init_point]
x, y = init_point
for i in range(iteration):
    x, y = conopt(x, y, gamma, eta)
    conopt_list.append([x, y])
    if nabla_f_x(x, y)<1e-5 and nabla_f_x(x, y)>-1e-5:
        print(i)
        break
    elif nabla_f_y(x, y)<1e-5 and nabla_f_y(x, y)>-1e-5:
        print(i)
        break
```

534

In [10]:
```
mix_list = [init_point]
x, y = init_point
for i in range(iteration):
    x, y = mix(x, y, rho_x, rho_y, gamma, eta)
    mix_list.append([x, y])
    if nabla_f_x(x, y)<1e-5 and nabla_f_x(x, y)>-1e-5:
        print(i)
        break
    elif nabla_f_y(x, y)<1e-5 and nabla_f_y(x, y)>-1e-5:
        print(i)
        break
```

207

Figure: From top to bottom, from left to right, they are SimGA, CESP, Conopt and Mix


z0
z1
z2
initial point
gd
cesp
conopt
mix
5
4
3
2
1
0
−1
−2
−5
−4
−3
−2
−1
0
1
2
8.000
12.000
16.000
20.000
24.000
28.000
32.000
36.000
40.000
44.000
52.000
4.000
0.000
-4.000

References

1. Lars Mescheder, Sebastian Nowozin, Andreas Geiger;The Numericals of GANs,11 Jun 2018, https://arxiv.org/abs/1705.10461
2. Leonard Adolphs, Hadi Daneshmand, Aurelien Lucchi, Thomas Hofmann; Local Saddle Point Optimization: A Curvature Exploitation Approach

The End