Review on Adagrad avoids saddle points

Weiliang Sun
2020533010
ShanghaiTech University
sunwl1@shanghaitech.edu.cn

Haoyuan Tian
2020533013
ShanghaiTech University
tianhy@shanghaitech.edu.cn

Jiaye Zheng
2022233389
ShanghaiTech University
zhengjy2022@shanghaitech.edu.cn

Abstract

Adaptive first-order methods in optimization are prominent in machine learning and data science owing to their ability to automatically adapt to the landscape of the function being optimized. However, their convergence guarantees are typically stated in terms of vanishing gradient norms, which leaves open the issue of converging to undesirable saddle points. In this paper, we want to focus on the $AdaGrad$ algorithm to find whether the method's trajectories avoid saddle points. We refer to the paper: *Adagrad avoids saddle points* and employ stable manifold techniques to show that the induced trajectories avoid saddle points from almost any initial condition.

1 Introduction

Deep learning architectures have brought forth a revolution in numerous application areas ranging from computer vision to speech recognition and natural language processing. Its main tool is gradient descent method. In this method, the learning rate is an important hyperparameter that controls the magnitude of parameter updates at each step. Many learning rates need to be adjusted manually. However, when manually adjusting it, a larger learning rate can speed up the convergence of the algorithm but may lead to parameter oscillations or failure to converge near the optimal solution. On the other hand, a smaller learning rate can improve the stability of the algorithm, but may require more iterations to converge to the optimal solution.

In this way, eliminating the need to manually tune the learning rate has emerged as a problem of seminal practical as well as theoretical importance. Then it lead to the emergence of adaptive gradient algorithms which perform more informed gradient steps in later iterations and are able to adapt efficiently to the landscape of the function being optimized.

Most adaptive gradient algorithms are based on the $AdaGrad$ algorithm, with a matrix step-size defined recursively by taking the square root of the sum of squares of past gradients -outer or inner, depending on the specific variant of the method. The $AdaGrad$ algorithm has been shown to attain an $\mathcal{O}(1/T)$ convergence rate when the optimizer has access to a perfect, deterministic gradient oracle, and an $\mathcal{O}(1/\sqrt{T})$ convergence rate when only stochastic gradients are available. Importantly, the merit function in both cases no longer the value of the objective function, but the sum of gradient norms squared; as a result, the above guarantees translate to a convergence rate for the method's "best iterate". In this regard, $AdaGrad$ is an order-optimal method in the non-convex case - which, coupled with its simplicity and the capability of exploiting sparse gradients-makes it an ideal choice for many problems with moderate-to-high dimensionality and a sparse solution structure.

However, there comes the origin of the problem in our paper. Since the only guarantee provided in non-convex settings is that of a vanishing gradient, not a value minimization certificate (local or global). It leaves open not only the question of global versus local optimality, but an even more fundamental one: Do the trajectories of $AdaGrad$ avoid saddle points?

2 Problem background and notations

2.1 Notations

1) saddle points: In mathematics, a saddle point or minimax point is a point on the surface of the graph of a function where the slopes in orthogonal directions are all zero (a critical point), but which is not a local extremum of the function.

2) $AdaGrad$ algorithm: It is a modified stochastic gradient descent algorithm with per-parameter learning rate, first published in 2011. Informally, this increases the learning rate for sparser parameters and decreases the learning rate for ones that are less sparse.

2.2 Background and set up

2.2.1 Problem set

The formal $AdaGrad$ algorithm is described by following recursive formula:

$$x_{t+1} = x_t - \Gamma_t \nabla f(x_t)$$

and the step size Γ_t has following three variants:

- AdaGrad-Norm: $\Gamma_t = \frac{1}{\sqrt{\delta_0^2 + \sum_{s=0}^{t} ||\nabla f(x_s)||^2}}$
- AdaGrad-Diag: $\Gamma_t = G_t^{-\frac{1}{2}}$ where $G_t = \delta_0^2 I + diag(\sum_{s=0}^{t} \nabla f(x_s) \nabla f(x_i)^T)$
- Full AdaGrad: $\Gamma_t = G_t^{-\frac{1}{2}}$ where $G_t = \delta_0^2 I + \sum_{s=0}^{t} \nabla f(x_s) \nabla f(x_i)^T$

2.2.2 Diffeomorphism

A differentiable mapping $\Phi : R^d \to R^d$ is a local diffeomorphism at x if the Jacobian matrix $D\Phi(x)$ is invertible. A typical example of a diffeomorphism is gradient descent with small constant step-size. In our setting, the step-size is time-dependent and the mapping $\Phi(t, x) = x - \Gamma_t \nabla f(x)$ is expected to be a diffeomorphism for all t.

2.2.3 Matrix preliminaris

For the adaptive step-size matrix Γ_t, we can have the following two lemmas:

- Lowner-Heinz inequality: If $A \geq B \geq 0$ and $0 \leq r \leq 1$ then $A^r \geq B^r$
- Weyl's Monotonicity: If H is positive, and the eigenvalues of $A + H$ and A are ordered as: $|\lambda_1(A+H)| \geq \cdots \geq |\lambda_d(A+H)|$ and $|\lambda_1(A) \geq \cdots \geq |\lambda_d(A)|$. Then $\lambda_i(A+H) \geq \lambda_i(A)$ for all i

2.2.4 Banach fixed point theorem

Let (X, d) be a complete metric space, then each contraction map $T : X \to X$ has unique fixed point. The property uniqueness means that the sequence (generated by $AdaGrad$) converging to the saddle point under discussion is unique.

3 Implementation

3.1 The sequence of adaptive matrices converges to positive scalar or positive definite matrix

First we denote the γ_t as $\frac{1}{\sqrt{\delta_0^2 + \sum_{s=0}^{t} ||\nabla f(x_s)||^2}}$, which emphasize that γ_t is the scalar step-size in $AdaNorm$.

We can easily know that γ_t is decreasing and bounded as: $lim_{t\to\infty}\gamma_t = inf_{t\in N}\gamma_t = \gamma_\infty \geq 0$ Without loss of generality, we can assume $\gamma_\infty = 0$, because f is smooth, so we have:

$$f(x_{t+1}) \leq f(x_t) + < \nabla f(x_t), x_{t+1} - x_t > + \frac{L}{2}||x_{t+1} - x_t||^2.$$

According to the definition of $AdaNorm$: $x_{t+1} - x_t = -\gamma_t \nabla f(x_t)$, we can get the conclusion that:

$$\frac{1}{2}\gamma_t||\nabla f(x_t)||^2 \leq f(x_t) - f(x_{t+1}) + \frac{\gamma_t}{2}(L\gamma_t - 1)||\nabla f(x_t)||^2$$

Summing up all the value of t and the fact that $f(x_T) \geq inf_t f(x_t)$, we get:

$$\frac{1}{2}\sum_{t=0}^{T}\gamma_t||\nabla f(x_t)||^2 \leq f(x_1) - inf_t f(x_t) + \sum_{t=0}^{T}\frac{\gamma_t}{2}(L\gamma_t - 1)||\nabla f(x_t)||^2$$

When $t \to \infty$ we assume $\gamma_t \to 0$ so there must exist t_0 for all $t > t_0$ satisfying $L\gamma_t - 1 < 0$. Therefore, because $\sum_{t=0}^{T}\frac{\gamma_t}{2}(L\gamma_t - 1)||\nabla f(x_t)||^2$ when $T = t_0$ spikes, meaning that $\sum_{t=0}^{\infty}\gamma_t||f(x_t)||^2 < +\infty$. To put the upper bound and lower bound together, we can have:

$$\frac{1}{2\gamma_T} - \frac{\delta_0}{2} \leq \sum_{t=0}^{T}\frac{\gamma}{2}||\nabla f(x_t)||^2 < +\infty$$

With our assumption we can conclude the result:

$$+\infty - \frac{\delta_0}{2} \leq \sum_{t=0}^{\infty}\gamma_t||\nabla f(x_t)||^2 < +\infty$$

which is a contradiction. Therefore, we have $\gamma_\infty > 0$ and the result follows.

3.2 Local structure of AdaGrad iterative dynamics

To be precise, we review our updating form of the $AdaGrad$ algorithm: $x_{t+1} = x_t - \Gamma_t \nabla f(x_t)$ Then we can rewrite the formula with the replacement $\Gamma_t = \Gamma + \Gamma_t - \Gamma$ with the assumption: $||\Gamma||_2 \leq ||\Gamma_0||_2 \leq \frac{1}{\delta_0} \leq \frac{1}{L}$:

$$x_{t+1} = x_t - \Gamma_t \nabla f(x_t) = x_t - \Gamma \nabla f(x_t) - (\Gamma_t - \Gamma)\nabla f(x_t).$$

Now without loss of generality we assume 0 is a strict saddle point meaning that $\nabla f(0) = 0$, then we can apply Taylor expansion to the formula:

$$\nabla f(x) = \nabla f(0) + \nabla^2 f(0)x + \theta(x) = \nabla^2 f(0)x + \theta(x).$$

With the Taylor expansion in a neighborhood of 0, we replace the first $\nabla f(x_t)$ with:

$$\nabla f(x_t) = \nabla^2 f(0)x_t + \theta(x_t).$$

Then the dynamical system can be represented the form of a stabilized linear transformation and a residual term:

$$x_{t+1} = (I - \Gamma\nabla^2 f(0))x_t + \eta(t, x_t),$$

where the residual $\eta(t, x) = -\Gamma\theta(x) - (\Gamma_t - \Gamma)\nabla f(x_t)$. With the Lipschitzness of $\theta(x)$ from Taylor expansion, Lipschitzness of $\nabla f(x_t)$ by assumption, and considering the convergence of Γ_t as well, we can intuitively derive the Lipschitzness of $\eta(t, x)$, that is for any $\epsilon > 0$, we have

$$||\eta(t, x) - \eta(t, y)|| \leq \epsilon||x - y||.$$

So far we have decomposed the dynamical system of $AdaGrad$ into a stabilized matrix and a residual term, with the Lipschitz type condition of the remainder, which is a crucial property to the following proof of the existence of local stable manifold.

3.3 Local stable manifold theorem

We will briefly follow the proving pipeline of the $Theorem\ 1$ in this paper, following the Lyapunov-Perron method [2], which states the local stable-manifold theorem corresponding to the $AdaGrad$ family of algorithms.

Consider the dynamical system

$$\frac{dx}{dt} = A(t)x + R(t, x),$$

where $A(t)$ is a time-dependent matrix. If the solution $u(t, x_0)$ generated by the dynamical system with some initial condition x_0 converges to an unstable fixed point, then it must hold that for the integral operator T that

$$Tu(t, x_0) = U(t)x_0 + \int_0^t U(t-s)R(s, u(s, x_0))ds - \int_0^t V(t-s)R(s, u(s, x_0))ds.$$

That derives the transformed sequence, which is the discrete version of T acting on space of sequences,

$$(Tx)_{t+1} = B(t, 0)x_0^+ + \sum_{i=0}^{t} B(t, i+1)\eta^+(i, x_0, x_i) - \sum_{i=0}^{\infty} C(t+1+i, t+1)^{-1}\eta^-(t+1+i, x_0, x_{t+1+i}).$$

Then the paper prove the following properties of operator T,

1. $\{(Tx)_t\}_{t\in\mathbb{N}}$ converges to the fixed point 0 as long as $\{x_t\}_{t\in\mathbb{N}}$ does,
2. $\{x_t\}_{t\in\mathbb{N}}$ converging to 0 whose initial points have the same stable component form a complete metric space,
3. T is a contraction mapping on the space of sequence converging to saddle point.

Together with Banach Fixed Point Theorem, that ensures the existence and uniqueness of local stable manifold of the dynamical system, we can conclude that there exists a unique sequence converging to 0 for each fixed stable component of the initial condition x_0.

It is followed by the statement that $\Gamma\nabla^2 f(0)$ is diagonalizable, given that we can derive that $\Gamma\nabla^2 f(0)$ has the same eigenvalues as $\Gamma^{\frac{1}{2}}\nabla^2 f(0)\Gamma^{\frac{1}{2}}$, while $\Gamma^{\frac{1}{2}}\nabla^2 f(0)\Gamma^{\frac{1}{2}}$ has the same number of positive and negative eigenvalues as $\nabla^2 f(0)$ does. So we state the diagonal decomposition of $\Gamma\nabla^2 f(0)$ being

$$\Gamma\nabla^2 f(0) = Q^{-1}HQ,$$

then the dynamical system can be derived to

$$x_{t+1} = (I - \Gamma\nabla^2 f(0))x_t + \eta(t, x_t) = Q^{-1}(I - H)Qx_t + \eta(t, x_t),$$

equivalent to

$$y_{t+1} = (I - H)y_t + Q\eta(t, Q^{-1}y_t),$$

where $y_t = Q_t$, and the Lipschitzness of the residual $Q\eta(t, Q^{-1}y_t)$ is proved.

So far $Theorem\ 1$ i.e. new local stable manifold theorem as mentioned in the paper is proved, indicating that the set of initial points converging to saddle point is of measure zero. We will next extend the conclusion from local to the whole Euclidean space.

3.4 AdaGrad is diffeomorphism map

After proving the local property that $AdaGrad$ family avoids saddle points, the paper proceeds to extends the property to $\mathbb{R}^d$ by proving that the mapping defined by $AdaGrad$ are diffeomorphisms. Since each iteration of the algorithms is dependent to all previous iterations, it is difficult to understand $AdaGrad$ algorithms are diffeomorphism. However, it is sufficient to show that for each iteration, i.e.

$$\mathbf{x}_{t+1} = \mathbf{x}_t - \Gamma_t\nabla f(\mathbf{x}_t)$$

We need to show that the map

$$\varphi(\mathbf{x}) = \mathbf{x} - \Gamma_t\nabla f(\mathbf{x})$$

where Γ_t is preconditioned step matrix from $AdaGrad$ family, is diffeomorphism. While it is trivial to show that $\varphi(\mathbf{x})$ is a bijection, the main task is to show that $\varphi(\mathbf{x})$ is invertible function . For simplicity, take $AdaFull$ as an example,

$$\mathbf{x}_{t+1} = \mathbf{x}_t - \Gamma_t \nabla f(\mathbf{x}_t)$$

where

$$\Gamma_t = (\delta_0^2 I + \sum_{s=0}^{t} \nabla f(\mathbf{x}_s) f(\mathbf{x}_s)^\top)^{-\frac{1}{2}}$$

, to show that $\varphi(x)$ is a diffeomorphism on the t iteration, we can split the Γ_t into:

$$\Gamma_t = (\delta_0^2 + S + \nabla f(\mathbf{x}_t) \nabla f(\mathbf{x}_t)^\top)^{-\frac{1}{2}}$$

where

$$S = \sum_{s=0}^{t-1} \nabla f(\mathbf{x}_s)$$

Thus the mapping

$$\varphi(\mathbf{x}) = \mathbf{x} - (\delta_0 I + S + \nabla f(\mathbf{x}_t) \nabla f(\mathbf{x}_t)^\top)^{-\frac{1}{2}} \nabla f(\mathbf{x})$$

is expected to be a diffeomorphism as long as δ_0 is properly tuned. An intuition on φ is that if δ_0 is large enough, $D\varphi(\mathbf{x})$ arbitrarily close to 1.

3.5 Extend Local Result to Global

Since the φ is provably a diffeomorphism, extending the result to global is the last step to complete the proof. The dynamical system defined by $AdaGrad$ algorithms determines each iterate based on time t and the initial condition $\mathbf{x}_0$. Thus, the $\mathbf{x}_{t+1}$ can be denoted as the image of a mapping depending on t and $\mathbf{x}_0$, i.e.

$$\mathbf{x}_{t+1} = \psi(t, \mathbf{x}_0).$$

We generalize the procedure between iteration m, n into

$$\tilde{\psi}(m, n, \mathbf{x}) = \psi(m, ..., \psi(n+1, \psi(n, \mathbf{x})))$$

We denote the fixed point set of dynamical system $\psi(t, x)$ as $\mathcal{A}^*$ and the stable set of $\mathcal{A}^*$ as $W^s(\mathcal{A}^*)$ which defined as

$$W^s(\mathcal{A}^*) = \{\mathbf{x}_0 : \lim_{k \to \infty} \psi(k, 0, \tilde{\mathbf{x}_0}) \in \mathcal{A}^*\}$$

Given a initial point $\mathbf{x}_0 \in W^s(A^*)$. Since

$$\tilde{\psi}(k, 0, \mathbf{x}_0) \to \mathbf{x}^* \in \mathcal{A}^*$$

there exists some non-negative integer T and all $t \geq T$, such that

$$\tilde{\psi}(t, 0, \mathbf{x}_0) \in \bigcup_{\mathbf{x}^* \in \mathcal{A}^*} U_{\mathbf{x}^*} = \bigcup_{i=1}^{\infty} U_{\mathbf{x}^*} https://www.overleaf.com/project/646f4f74dfef5cab0ece06fb$$

which equivalent to

$$\tilde{\psi}(T + k, T, \psi(T, \tilde{0}, \mathbf{x}_0)) \in U_{\mathbf{x}^*}$$

for all $k \geq 0$, and this implies that

$$\tilde{\psi}(T, 0, \mathbf{x}_0) \in \tilde{\psi}^{-1}(T + k, T, U_{\mathbf{x}^*})$$

for all $k \geq 0$. And then we have

$$\tilde{\psi}(T, 0, \mathbf{x}_0) \in \bigcap_{k=0}^{\infty} \tilde{\psi^{-1}}(T + k, T, U_{\mathbf{x}_i^*})$$

Then denote $S_{i,T} := \bigcap_{k=0}^{\infty} \tilde{\psi^{-1}}(T+k, T, U_{\mathbf{x}_i^*})$ and the above relation is equivalent to $\mathbf{x}_0 \in \tilde{\psi}^{-1}(T, 0, S_{i,T})$. Take the union for all $T \geq 0$, we have

$$\mathbf{x}_0 \in \bigcup_{i=1}^{\infty} \bigcup_{T=0}^{\infty} \tilde{\psi}^{-1}(T, 0, S_{i,T})$$

implying that

$$W^s(\mathcal{A}^*) \subset \bigcup_{i=1}^{\infty} \bigcup_{T=0}^{\infty} \tilde{\psi}^{-1}(T, 0, S_{i,T})$$

Since $S_{i,T} \subset W_n(\mathbf{x}^*)$ and $W_n(\mathbf{x}^*)$ has codimention with $\mathbb{R}^n$ at least 1(by Bananch fixed-point theorem). Thus, S_i, T has measure 0. Since the image of set of measure 0 under diffeomorphism is of measure 0 and countable union of 0 measure sets is still measure 0, we have that $W^s(\mathcal{A}^*)$ is of measure 0.

4 Conclusion and Lesson Learned

Conclusion This paper interprets the saddle point avoidance of $AdaGrad$ family using stable set analysis. Despite the paradigm status of dynamical system analysis for Gradient Descent saddle point interpretation problems, the paper overcomes the difficulty that the iterative algorithm has dependent variable.

Innovation The innovation we think of this paper is that it proves that $AdaGrad$ algorithm can avoid saddle points and converge to second-order stationary points in non-convex optimization problems. This result is important for understanding and improving the performance of adaptive gradient methods in machine learning applications. Meanwhile, the paper also provides a new state-space perspective for analyzing the convergence of $AdaGrad$ and its variants such as Adam by using a recursive representation of the $AdaGrad$ preconditioner and employs center-stable techniques to show that the trajectories of the algorithm avoid saddle points from almost any initial condition.

Lesson Learned Thanks to its concise mathematical language to perform the proof, the paper makes it easy to understand. However, to understand the proof in the paper, we took efforts to study some basic knowledge of dynamical system, matrix analysis, and topology. Since the wide deployment of $AdaGrad$ family in deep learning, the paper will be a basis for the community to understand the performance of deep learning as well as any other non-convex optimization.

5 Code Implementation

In this part, we write a code to verify that the $AdaGrad$ algorithm can avoid saddle points. We use a simple function $f(x, y) = x^2 - y^2$ and visualize the final result.

We first show the numerical result of the $AdaGrad$ algorithm. We set the initial point as $(10, 0.5)$ and the learning rate as 0.1.

```
Iteration 0: x = 9.90310001373291, y = 0.5048450231552124, f(x, y) = 97.81652069091797
Iteration 100: x = 8.209085464477539, y = 0.6087490320205688, f(x, y) = 67.01850891113281
Iteration 200: x = 7.4605817794799805, y = 0.6697618961334229, f(x, y) = 55.211700439453125
Iteration 300: x = 6.904938220977783, y = 0.7236148715019226, f(x, y) = 47.154556274414006
Iteration 400: x = 6.450438499450684, y = 0.7745651006698608, f(x, y) = 41.008205413818336
Iteration 500: x = 6.061243534088135, y = 0.8242681622505188, f(x, y) = 36.059257507324222
Iteration 600: x = 5.718869209289551, y = 0.8735859990119934, f(x, y) = 31.942310333251953
Iteration 700: x = 5.412284374237061, y = 0.9230433106422424, f(x, y) = 28.440813064575195
Iteration 800: x = 5.134271621704102, y = 0.9729978442192078, f(x, y) = 25.414020538330078
Iteration 900: x = 4.879795551300049, y = 1.0237125158309937, f(x, y) = 22.76441764831543
Iteration 1000: x = 4.645175933837891, y = 1.0753921270370483, f(x, y) = 20.421192169189453
Iteration 1100: x = 4.427624702453613, y = 1.1282045841217041, f(x, y) = 18.331014633178711
Iteration 1200: x = 4.224965095520195, y = 1.1822959184646606, f(x, y) = 16.452507019042969
Iteration 1300: x = 4.035455703735352, y = 1.2377921342849731, f(x, y) = 14.752773284912111
Iteration 1400: x = 3.8576788902282715, y = 1.2948079109191895, f(x, y) = 13.205158233642578
Iteration 1500: x = 3.690459966659546, y = 1.3534504175186157, f(x, y) = 11.787666320800781
Iteration 1600: x = 3.5328128337860107, y = 1.4138187170028687, f(x, y) = 10.481883049011230
Iteration 1700: x = 3.383899211883545, y = 1.4760087728500366, f(x, y) = 9.272171974182129
Iteration 1800: x = 3.242994785308838, y = 1.5401115417480469, f(x, y) = 8.145071983337402
Iteration 1900: x = 3.1094727516174316, y = 1.606215476989746, f(x, y) = 7.088891983032227
Iteration 2000: x = 2.982785940170288, y = 1.6744067668914795, f(x, y) = 6.093373775482178
```

(a) Iteration 2000

```
Iteration 7400: x = 0.5305091142654419, y = 9.408987998962402, f(x, y) = -88.24761199951172
Iteration 7500: x = 0.5196202993392944, y = 9.606115341186523, f(x, y) = -92.0074462890625
Iteration 7600: x = 0.5091590881347656, y = 9.803444862365723, f(x, y) = -95.8482894897461
Iteration 7700: x = 0.4991849684715033, y = 10.000890731811523, f(x, y) = -99.76870727539062
Iteration 7800: x = 0.4894382655620575, y = 10.198396682739258, f(x, y) = -103.76774597167969
Iteration 7900: x = 0.4801404774188995, y = 10.395848274230957, f(x, y) = -107.84312438964844
Iteration 8000: x = 0.4711940884590149, y = 10.593199729919434, f(x, y) = -111.99385833740234
Iteration 8100: x = 0.4625824093818664, y = 10.790369033813477, f(x, y) = -116.21807861328125
Iteration 8200: x = 0.4542895555496216, y = 10.987306594848633, f(x, y) = -120.5145263671875
Iteration 8300: x = 0.4463006556034088, y = 11.183949470520020, f(x, y) = -124.88154602050781
Iteration 8400: x = 0.4386011064052582, y = 11.380248069763184, f(x, y) = -129.3176727294922
Iteration 8500: x = 0.4311774671077728, y = 11.576147079467773, f(x, y) = -133.8212738037109
Iteration 8600: x = 0.4240170121192932, y = 11.771604537963867, f(x, y) = -138.3908843994140
Iteration 8700: x = 0.4171074330806732, y = 11.966574668884277, f(x, y) = -143.0249328613281
Iteration 8800: x = 0.4104372859001597, y = 12.161016464233398, f(x, y) = -147.7218627929687
Iteration 8900: x = 0.4039955437183380, y = 12.354894638061523, f(x, y) = -152.4802093505859
f(x, y) did not reach zero during the iteration.
```

(b) Iteration 9000

Figure 1: 9000 iterations

It shows that the algorithm escaped the saddle points. Now is the visualization of our code:

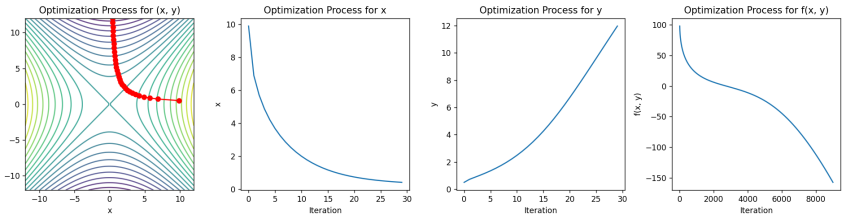

Figure 2: Visualization of the result

References

[1] Kimon Antonakopoulos & P.Mertikopoulos & Xiao Wang (2022) AdaGrad Avoids Saddle Points, *International Conference on Machine Learning*

[2] Perko, L. Differential Equations and Dynamical Systems. Springer, 2001.