EXPLORATION OF MINIMIZATION-MAXIMIZATION METHODS

CONVEX OPTIMIZATION - FINAL PROJECT

Facundo Sapienza
Department of Statistics, UC Berkeley
fsapienza@berkeley.edu

ABSTRACT

Minimization-maximization or saddle-point problems play a central role in optimization, applied mathematics and recently in machine learning. In this work, we will explore different first order methods (gradient descent, optimistic mirror descent) and second order methods (implicit twisted descent and variants of it). We will introduce some new methods for the minimax problem and we will study their performance for different cost functions.

1 INTRODUCTION

Given a function $L : \mathbb{R}^n \times \mathbb{R}^m \mapsto \mathbb{R}$, we are going to study the following form of unconstrained minimization-maximization problem (minimax)

$$\min_{x\in\mathbb{R}^n} \max_{y\in\mathbb{R}^m} L(x, y), \tag{1}$$

where n and m are the dimensions of the variables x and y, respectively. Finding solutions to this problem is equivalent to finding a pair (x^*, y^*) such that $L(x^*, y) \leq L(x^*, y^*) \leq L(x, y^*)$ for all (x, y) in some neighborhood of (x^*, y^*). This last formulation of the minimax problem resembles the notion of Nash equilibrium in zero-sum games. We can think that x and y are two players, where the former is trying to minimize his/her loss and the second trying to maximize his/her profit [1]. A solution of the minimax problem corresponds to a point where either x or y can improve his current situation changing his strategy while the other player does not change his position.

Applications of minimax problems include:

- *Two players zero-sum games*. As we just mentioned, a minimization-maximization problem is a formulation for finding Nash Equilibrium when both variables x and y are continuous.
- *Constarined Optimization*. As we saw during this course, constrained optimization problems can be formulated as a minimization-maximization problem on the Lagrangian. For example, consider the problem

  $$\min_{x\in\mathbb{R}^n} f(x) \quad \text{subject to } g_i(x) = 0 \text{ for all } i = 1, 2, \ldots, m. \tag{2}$$

  Instead of solving this problem, we can attempt to solve the saddle point formulation given by

  $$\min_{x\in\mathbb{R}^n} \max_{\lambda\in\mathbb{R}^m} f(x) + \langle \lambda, g(x)\rangle. \tag{3}$$

  In Essid et al. (2019a), the authors showed how to use this approach in order to find the optimal transport strategy to map a distribution of points into another one.
- *Generative Adversarial Neural Networks*. A current living area of research in machine learning is the training and design of GANs, Goodfellow et al. (2014). The core of this kind of networks consists on solving a minimax problem, where minimization and maximization are performed on the weights of a generative and discriminative neural network.

[1] Notice that since $L(x, y)$ can take negative values, the player x can also earn a positive profit.

2 METHODS

In this section we are going to introduce some of the standard and state of the art algorithms used to solve minimax optimization problems. The simplest approach is running **gradient descent** (GD) for each player. Given some initial values $x^0 \in \mathbb{R}^n$, $y^0 \in \mathbb{R}^m$ and $t = 1, \dots, T$, and assuming that L is differentiable, we iterate the following rule

$$x^{t+1} = x^t - \alpha^t \nabla_x^t, \tag{4}$$

$$y^{t+1} = y^t + \alpha^t \nabla_y^t, \tag{5}$$

where $\nabla_x^t = \nabla_x L(x^t, y^t)$, $\nabla_y^t = \nabla_y L(x^t, y^t)$ and α^t is the stepsize. Notice that we are performing gradient ascent in the variable y, even when for simplicity we decided to call this method simply GD. This first method have several problems:

1. There are no guarantees about the convergence even for the convex-concave case, that is, when $L(\cdot, y)$ is a convex function and $L(x, \cdot)$ is concave for all values of x and y.
2. In applications, many solutions have periodic orbits that do not converge.

In order to generalize this method, let us first observe that the gradient descent update is equivalent to the Follow-the-Regularized-Leader formulation of the problem used in the online optimization framework, (Shalev-Shwartz (2012)). Given the initial value $x^0 = 0$, it is easy to check that equation (4) is equivalent to

$$x^{t+1} = \text{argmin}_{x \in \mathbb{R}^n} \left\{ \sum_{s=1}^{t} \alpha^t \langle \nabla_x^s, x \rangle + \frac{1}{2} \|x\|_2^2 \right\}. \tag{6}$$

Notice that this is equivalent to what happens with the mirror descent method with the use of the ℓ_2 divergence. Inspired by this last expression of GD, we can consider to augment the cost function in (6) adding a predictor M_x^{t+1} of the gradient at the next iteration:

$$x^{t+1} = \text{argmin}_{x \in \mathbb{R}^n} \left\{ \sum_{s=1}^{t} \alpha^s \langle \nabla_x^s, x \rangle + \alpha^{t+1} \langle M_x^{t+1}, x \rangle + \frac{1}{2} \|x\|_2^2 \right\}, \tag{7}$$

which leads to the following updating rule

$$x^{t+1} = x^t - \alpha^t \nabla_x^t - \alpha^{t+1} M_x^{t+1} + \alpha^t M_x^t. \tag{8}$$

If we use the gradient of the previous update as predictor, that is, $M_x^{t+1} = \nabla_x^t$, and we assume $\alpha^t \approx \alpha^{t+1}$, then we arrive to the update rule for **optimistic gradient descent** (OGD) method

$$x^{t+1} = x^t - 2\alpha^t \nabla_x^t + \alpha^t \nabla_x^{t-1} \tag{9}$$

$$y^{t+1} = y^t + 2\alpha^t \nabla_y^t - \alpha^t \nabla_y^{t-1}. \tag{10}$$

In Daskalakis et al. (2017), the authors showed the improvements of training a Wasserstein GAN using OGD instead of ordinary GD. Notice that OMD updates x^t in the direction of the vector $\nabla^t + (\nabla^t - \nabla^{t-1})$. If we think the succession x^t as the position of a particle in space, then the first vector corresponds to the instantaneous velocity, while the difference $\nabla^t - \nabla^{t-1}$ is approximately proportional to the acceleration of the particle. Then, intuitively, we can understand why OMD avoid the problem of periodic orbits in GD: if a particle is in circular motion, it experiences an acceleration that points to the center of the circle.

In a more recent paper the authors introduced **implicit twisted-gradient descent** (ITG), (Essid et al. (2019b)). The idea in this paper is really elegant: if we think the minimax problem as a zero-sum game, where the player x wants to minimize his/her loss and y wants to maximize his/her profit, we can think in a scheme were both players can anticipate the state of the game at the next step, instead of just looking at the current state of the game. The update of both x^t and y^t takes into account the future position of both players:

$$x^{t+1} = x^t - \alpha^t \nabla_x^{t+1}, \tag{11}$$

$$y^{t+1} = y^t + \alpha^t \nabla_y^{t+1}. \tag{12}$$

Notice that this formulation is equivalent to (7) for $M_x^{t+1} = \nabla_x^{t+1}$, instead of the ∇_x^t of OMD. For simplicity, let us introduce the following notation

$$z^t = \begin{pmatrix} x^t \\ y^t \end{pmatrix} \in \mathbb{R}^{n+m}, \quad \nabla^t = \begin{pmatrix} \nabla_x^t \\ \nabla_y^t \end{pmatrix} \in \mathbb{R}^{n+m}, \quad J = \begin{pmatrix} I_{n\times n} & 0_{n\times m} \\ 0_{m\times n} & -I_{m\times m}, \end{pmatrix} \in \mathbb{R}^{(n+m)\times(n+m)} \tag{13}$$

where $I_{n\times n}$ is the identity matrix in $\mathbb{R}^{n\times n}$. Then, equations (11) and (12) can been written in a more compact form as

$$z^{t+1} = z^t - \alpha J \nabla^{t+1}. \tag{14}$$

Solving these equations implies having access to the value of the gradients of $L(x, y)$ at (x^{t+1}, y^{t+1}). If $L(x, y)$ is twice differentiable, then we can approximate these quantities at first order using the Taylor expansion

$$\nabla^{t+1} \approx \nabla^t + H^t(z^{t+1} - z^t), \tag{15}$$

where H^t is the Hessian at step t. Under this approximation, the updating scheme is given by

$$z^{t+1} = z^t - \alpha(J + \alpha H^t)^{-1}\nabla^t. \tag{16}$$

First of all, notice that ITD is a second order method. In the original paper, the authors also introduce a quasi implicit ITD that avoids the computation and inversion of the Hessian.

In order to better understand this iteration rule, let us see what happens in both regimes $\alpha \approx 0$ and $\alpha \to \infty$. In the first case, the iteration rule of ITD simplifies to the GD update $z^{t+1} \approx z^t - \alpha J \nabla^t$. On the other hand, for large values of α we have that $z^{t+1} \approx z^t - (H^t)^{-1}\nabla^t$, which corresponds to the Newton method. However, notice that in this case the updating rule coincides with the problem of minimizing $L(x, y)$ in both x and y, reason why we have to be careful about the value of the stepsize we use in practice. Furthermore, all this approximations are valid when $z^{t+1} \approx z^t$, which is a condition that in general is not going to be satisfied for large values of α. We are going to discuss the choice of the stepsize in Section 4.

2.1 Exploration of new methods for the minimax problem

In this section we are going to introduce some new methods for the minimax optimization problem. The first immediate generalization we can think is a mix of Optimistic Mirror Descent and Implicit Twisted Descend, were we replace the gradient ∇^t of ITD by $2\nabla^t - \nabla^{t-1}$. Now, in the limit when α is small, ITD coincides with OMD instead of GD. We call this method **Implicit Optimistic Descent** (IOD) and it is defined by the scheme

$$z^{t+1} = z^t - \alpha(J + \alpha H^t)^{-1}(2\nabla^t - \nabla^{t-1}). \tag{17}$$

In practice, I did not observe any improvement of this method with respect to ITD or OMD. However, it is a valid generalization that for some problems can behave better.

Let us now analyze other different variants for the minimax problem. The first method arises from the wish of combining the mirror method with Implicit Twisted Descent. The second one consists on a game theoretical approach to the problem.

2.1.1 Mirror descent generalization of ITD

It is easy to check that the scheme given by ITD (equation (16)) is equivalent to the other following scheme:

1. Consider that the prediction for the gradient at step $t + 1$ does not depend on z^{t+1}. Then

$$\begin{aligned} z^{t+1} &= \text{argmin}_z \left\{ \alpha\langle JM^{t+1}, z - z^t\rangle + \frac{1}{2}\|z - z^t\|_2^2 \right\} \\ &= z^t - \alpha J M^{t+1} \end{aligned} \tag{18}$$

2. Now, plug in $M^{t+1} = \nabla^{t+1}$ in the last expression and obtain $z^{t+1} = z^t - \alpha J \nabla^{t+1}$.
3. Finally, approximate $\nabla^{t+1} \approx \nabla^t + H^t(z^{t+1} - z^t)$ and solve.

However, we can think of a new scheme where we change the order of these operations. Indeed, instead of using $M^{t+1} = \nabla^{t+1}$ after the first minimization in equation (18) and then using the Taylor approximation, we can directly plug in

$$M^{t+1} = \nabla L(z) \approx \nabla^t + H^t(z - z^t) \tag{19}$$

and formulate a new updating rule given by

$$z^{t+1} = \text{argmin}_z \left\{ \alpha\langle J\nabla^t, z\rangle + \alpha\langle JH^t(z - z^t), z - z^t\rangle + \frac{1}{2}\|z - z^t\|_2^2 \right\}. \tag{20}$$

This argument is a quadratic function in z and it is an easy optimization to solve. The optimality condition leads to

$$\alpha J\nabla^t + 2\alpha JH^t(z^{t+1} - z^t) + z^{t+1} - z^t = 0. \tag{21}$$

Multiplying both sides by J, noticing that $J = J^{-1}$ we obtain

$$z^{t+1} = z^t - \alpha(J + 2\alpha H^t)^{-1}\nabla^t. \tag{22}$$

Notice that this expression is the same as in equation (16) with an extra 2 factor multiplying the Hessian. Also notice that both equations (21) and (22) are equivalent to

$$z^{t+1} = z^t - \alpha(J + \alpha H^t)^{-1}(\nabla^t + H^t(z^{t+1} - z^t)). \tag{23}$$

Then, we can undestand the extra 2 factor as the result of using a first order approximation for the gradient at step $t + 1$ in the scheme given by ITD instead of just using the gradient. In general, we can write the two models (16) and (22) as

$$z^{t+1} = z^t - \alpha(J + g\alpha H^t)^{-1}\nabla^t, \tag{24}$$

with g another parameter. For $g = 1$, we have the original ITG method. For other values of g, we are going to define the method defined by (24) as g-ITD. Notice that in general, for a fixed value of g we should not expect g-ITD to behave much differently than ITD. However, I think this last formulation is more intuitive and useful in practice when we need to choose the stepsize at each step of the algorithm. We can think that the product $g\alpha$ measures the importance we give to the Hessian in each iteration, which could lead to faster convergence rates; while α corresponds to the real stepsize and quantifies how much do we move in the direction given by the vector $(J + g\alpha H^t)^{-1}\nabla^t$.

2.1.2 227 METHOD

If we observe the form of the implicit updates proposed for ITD, we observe that both players x and y are trying to anticipate the state of the game when both move simultaneously to (x^{t+1}, y^{t+1}). However, from a game theory point of view, it also makes sense that the player x tries to update it value from x^t to x^{t+1} just looking at the gradient of the cost function at (x^t, y^{t+1}), that is, looking how to update the value of x^t looking at the future position of y, but were the value of x is fixed (since this is the current position of the player). This idea leads to the following scheme for the minimax problem:

$$x^{t+1} = x^t - \alpha\nabla_x L(x^t, y^{t+1}), \tag{25}$$

$$y^{t+1} = y^t + \alpha\nabla_y L(x^{t+1}, y^t). \tag{26}$$

Using the same ideas as before, we propose a first order approximation of the gradient based on the Hessian of the cost function $L(x, y)$. Let us denote $H_{xx} \in \mathbb{R}^{n\times n}$,$H_{xy} \in \mathbb{R}^{n\times m}$, $H_{yx} \in \mathbb{R}^{m\times n}$ and $H_{yy} \in \mathbb{R}^{m\times m}$ the submatrices of the Hessian such that

$$H = \left[\begin{array}{c|c} H_{xx} & H_{xy} \\ \hline H_{yx} & H_{yy} \end{array}\right]. \tag{27}$$

Notice that if L has continuous second partial derivatives , then $H_{xy} = H_{yx}^T$. We have that

$$\nabla_x L(x^t, y^{t+1}) \approx \nabla_x^t + H_{xy}(y^{t+1} - y^t), \quad \nabla_y L(x^{t+1}, y^t) \approx \nabla_y^t + H_{yx}(x^{t+1} - x^t). \tag{28}$$

Replacing these approximations on equations (25) and (26) we obtain the following scheme of iterations

$$x^{t+1} = x^t - \alpha\left(\mathbb{I}_{n_x\times n_x} + \alpha^2 H_{xy}^t H_{yx}^t\right)^{-1}\left(\nabla_x^t + \alpha H_{xy}^t\nabla_y^t\right), \tag{29}$$

$$y^{t+1} = y^t + \alpha\left(\mathbb{I}_{n_y\times n_y} + \alpha^2 H_{yx}^t H_{xy}^t\right)^{-1}\left(\nabla_y^t - \alpha H_{yx}^t\nabla_x^t\right). \tag{30}$$

Notice that this scheme does not depend of H_{xx} or H_{yy}. Just as we did with the ITD method, let us get some intuition about what this method is doing in the extreme regimes. For $\alpha \approx 0$, this iterations reduced to the simple GD scheme:

$$x^{t+1} \approx x^t - \alpha \nabla_x^t, \quad y^{t+1} \approx y^t + \alpha \nabla_y^t. \tag{31}$$

On the other hand, for α large enough we have that (29) reduces to

$$x^{t+1} \approx x^t - (H_{xy}^t (H_{xy}^t)^T)^{-1} H_{xy}^t \nabla_y^t. \tag{32}$$

This last equation resembles to the ordinary least square estimator from linear regression. Effectively, simple calculations show that this last equation is equivalent to

$$x^{t+1} = \text{argmin}_x \, \|\nabla_y^t + H_{xy}^t (x - x^t)\|_2^2 \approx \text{argmin}_x \, \|\nabla_y L(x, y^t)\|_2^2, \tag{33}$$

where in the last expression we use the Taylor approximation again. Then, in the limit when $\alpha \to \infty$, x tries to update its value to move to a point where the gradient of the cost function with respect to y is as small as possible. In the ideal case, x^{t+1} will be such that $\nabla_y L(x^{t+1}, y^t) = 0$, making it impossible for y to improve its profit. Since I do not have any clever name for this method, I am going to refer to is as the **227 method**.

3 RESULTS

3.1 IMPLEMENTATION

All the previously mentioned methods were implemented in `julia`. Since I did not have experience working with julia before the project, I decided to spend some time learning julia (not much, since it is quite similar to Python). All the code and Jupyter Notebooks with the experiments included in this report are available in the following repository: `https://github.com/facusapienza21/minimax-optimization.git`. The results can be reproduced from any local computer. The function `minimax_solver` includes a series of optimization methods for minimax problems. Their arguments are:

- `L`: real valued cost fuction. The gradient and Hessian are obtained using the `ForwardDiff` package that implements automatic differentiation.
- `method`: string specifying the method. Options implemented are `GD` (Gradient descent), `OMD` (Optimisitic mirror descent), `ITD` (Implicit twisted descend), `227` (227 method), `adaptive ITD`, `adaptive 227`, `IOD` (Implicit optimistic descent), `adaptive IOD`.
- `nx, ny`: integers. The first `nx` coordinates in the argument of the cost function correspond to the variable x and the second, `ny`, to y.
- `alpha`: stepsize. For the adaptive methods, this coefficient has to be larger than 1 and quantifies a different magnitude (see Section 4).
- `initialization`: initial value of the method. If `false`, then it selects a vector with all entries equal to one.
- `gfactor`: value of g for the second order methods.
- `mu0, mumax`: parameters of the backtracking (see Section 4)

The output of this algorithm is an object of a class `minimax`, which contains the sequence of values of x^t, y^t, z^t and α^t.

3.2 SIMPLE EXAMPLES WITH FIXED STEPSIZE

Let's start our analysis with a simple one dimensional example ($x, y \in \mathbb{R}$) to illustrate how all these methods compare to each other. For now, we are going to think that the value of the stepsize α is fixed. Consider the function

$$L_1(x, y) = xy. \tag{34}$$

It is easy to check that the only possible solution to the minimax problem is $(x^*, y^*) = (0, 0)$. If for example $x^* \neq 0$, then y can increase his/her absolute value in order to make the cost function go to

infinity. The GD method gives iterations $x^{t+1} = x^t - \alpha y^t$, $y^{t+1} = y^t + \alpha x^t$, which correspond to the Forward Euler method for the pair of differential equations given by

$$\begin{cases} \dot{x} = -y \\ \dot{y} = x. \end{cases} \tag{35}$$

This pair of differential equations has solutions that look like orbits around $(0, 0)$, and the quantity $x^2 + y^2$ is conserved during the dynamic. It is known that Implicit Euler (GD) gives solutions that are orbits and consequently do not converge to $(0, 0)$ and actually they do not correspond to the real solution of the differential equation. Furthermore, a semi-implicit scheme given by $x^{t+1} = x^t - \alpha y^t$, $y^{t+1} = y^t + \alpha x^{t+1}$ also gives orbits, reason why we do not explore this variant for minimax solvers, Ascher (2008).

Figure 1 shows the trajectories of the different methods for the function L_1 for the same fixed value of stepsize $\alpha = 0.5$, same number of iterations and initialization. Notice that the method 227 coincides with ITD for this simple cost function. The trajectory of GD diverges. The rest of the methods converge to zero with the number of steps, just as expected.

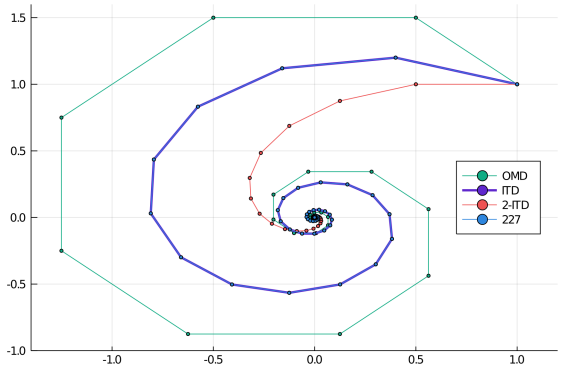


Figure 1: *Trajectories for $L_1(x, y)$. Notice that the trajectories of ITD and 227 coincide.*

In order to quantify how well this method performed for different choices of α, given a desired precision $\epsilon > 0$ we can define the following iteration complexity

$$T(\alpha; \epsilon) = \min \left\{ t : \|(x^t, y^t) - (x^*, y^*)\|_2 < \epsilon, (x^*, y^*) \text{ a solution of the minimax} \right\}. \tag{36}$$

Figure 2 shows this iteration complexity as a function of α for these different methods for $\epsilon = 10^{-5}$ and for the function L_1. Remember that 227 and ITD coincide for this example. We can observe that both ITD and g-ITD converge much faster as we increase the value of α. This makes sense, since the function L_1 is quadratic and in this case one single Newton step converges to the optimal solution. We also observe that there is a critical value of α above which OMD performs very bad. This regime corresponds to trajectories that diverge. This phase transition shows that it could be difficult to choose a right value for the stepsize in OMD. On the behalf of OMD, remember that OMD is the only first order method showed on this plot, and for small values of α it performs as well as ITD.

Let us analyze now some more complicated functions. Following Mertikopoulos et al. (2019), consider the functions

$$L_2(x, y) = xy + \frac{1}{3} e^{-x^2 - y^2}, \tag{37}$$

$$L_3(x, y) = (x^4 y^2 + x^2 + 1)(x^2 y^4 - y^2 + 1). \tag{38}$$

They both have a single saddle point at $(0, 0)$. Notice that $L_3(x, y)$ is an example of a function with just one saddle point that is not convex-concave. Figure 3 displays the same analysis we did for L_1 but now on this functions. We observe also that all these methods (except GD) converge to zero. A

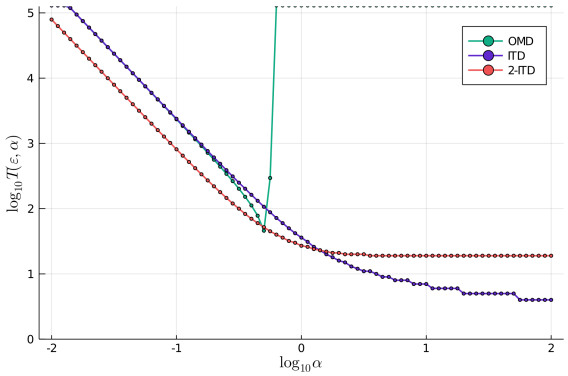


Figure 2: *Complexity for $L_1(x, y)$ as a function of α. Notice that both scales are logarithmic.*

more interesting example is the following function, also considered in Mertikopoulos et al. (2019) Essid et al. (2019b):

$$L_4(x, y) = (x - 0.5)(y - 0.5) + \frac{1}{3}e^{-(x-0.5)^2-(y-0.75)^2}. \tag{39}$$

This function has a saddle point near $(0.5, 0.5)$ and a local maximum near $(0.5, 0.75)$. Then, it is interesting to see if any of these methods converge to the real saddle point or if they get stuck in the local maximum, where the gradient vanishes. Notice that it is particularly difficult to ensure that any of these methods will converge to a saddle point instead of a point were $\nabla L(x, y) = 0$. The first condition demands that $\nabla L(x, y) = 0$, but also $H_{xx}(x, y)$ and $H_{yy}(x, y)$ have to be semi-positive and semi-negative definite, respectively. However, we observe that the methods we use converge to the true saddle point.

Notice that some of them do it faster than others, but this is a not fair comparison because α is fixed at the same value for all of these methods. On the right column of Figure 3, we have the value of $T(\alpha, \epsilon)$ for $\epsilon = 10^{-5}$ as a function of α. Even when these functions are not quadratic, we observe that the convergence of ITD and g-ITD is faster as we increase α. We also observe that there is a critical value of α after which OMD diverges. Something interesting to observe is that there is a range of values of α for which the 227 method reaches a distance at most ϵ in less steps than all the other methods for any value of α for the functions L_3 and L_4. Also, here there is a critical value of α just as in OMD.

4 Stepsize Selection

Until now, we just explored these algorithms for a fixed value of the stepsize α. However, as we observed in the previous examples, the behaviour of the method is quite sensitive to this parameter. In this section we are going to introduce an adaptive version of ITD, OMD and the 227 method, where the step size α^t changes between iterations. The intuition behind this is simple: when possible, we want to use large values of α in order to converge faster to a point where the gradient vanishes. For ITD, we can see that

$$\nabla^{t+1} \approx \nabla^t + H^t(z^{t+1} - z^t) = \left(I - \alpha H^t(J + \alpha H^t)^{-1}\right)\nabla^t = (I + \alpha H^t J)^{-1}\nabla^t. \tag{40}$$

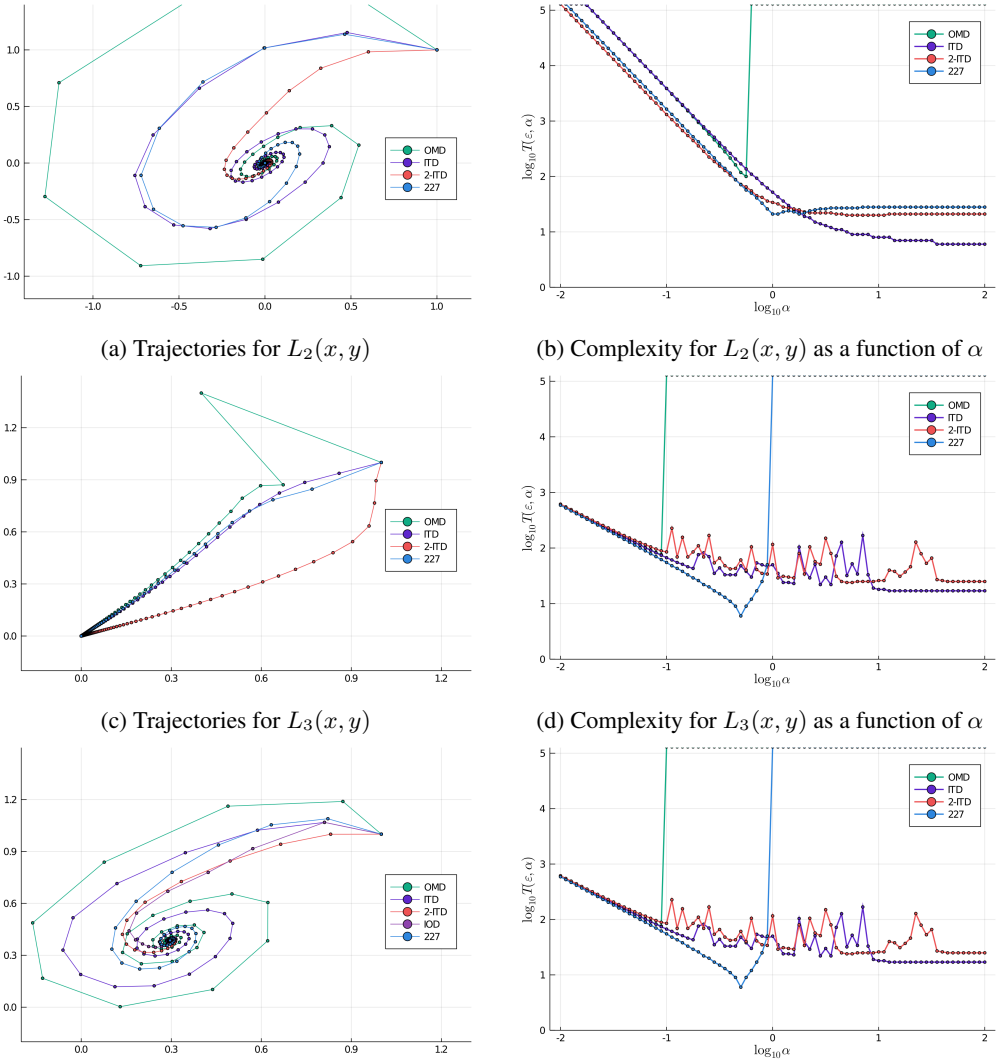
Let us assume that there exists a constant $m > 0$ such that $\langle z, H^t J z\rangle \geq m\|z\|^2$. For example, it is easy to check that this is satisfied in the strongly convex-concave case. Then

$$\|I + \alpha H^t J\|_2 = \sup_{z:\|z\|_2=1} \langle z, (I + \alpha H^t J)z\rangle \geq 1 + m\alpha. \tag{41}$$

Consequently, we will have the following bound between the norm of the gradients at consecutive steps:[2]

$$\|\nabla^{t+1}\|_2 \leq \frac{1}{1 + \alpha m}\|\nabla^t\|_2. \tag{42}$$

[2]Notice that this bound in better than the one found in Essid et al. (2019b), equation 11.



(a) Trajectories for $L_2(x, y)$

(b) Complexity for $L_2(x, y)$ as a function of α

(c) Trajectories for $L_3(x, y)$

(d) Complexity for $L_3(x, y)$ as a function of α

(e) Trajectories for $L_4(x, y)$

(f) Complexity for $L_4(x, y)$ as a function of α

Figure 3: *Trajectories given for different schemes. Here we consider Optimistic Mirror Descent, Implicit Twisted Descent for $g = 1$ and $g = 2$, 227 method and we also try with Implicit Optimistic Descent. In the three cases, the initial point is $(1, 1)$ and $\alpha = 0.5$. The GD method gives solutions that diverge.*

Based on this, our intuition dictates that we should chose a value of α as large as possible in order to reduce the norm of the gradient. However, vanishing gradient does not necessarily mean saddle point. Furthermore, the first approximation in (40) relies on the fact that z^t and z^{t+1} are close to each other, which is not the case for large values of α, reason why we need some kind of condition that ensures we are approaching to a saddle point. The strategy followed in Essid et al. (2019b) and that we are going to use here is to chose α^t such that at each step the condition

$$L(x^{t+1}, y^t) \leq L(x^{t+1}, y^{t+1}) \leq L(x^t, y^{t+1}) \tag{43}$$

is satisfied. This agrees with the definition of saddle point and with the anticipatory game idea behind the method. To change the value of α^t, we are going to use the following strategy. Given an initial value μ^0, at each step we increase the value of α^t by $\mu^{t+1} = \min\{\nu\mu^t, \mu_{max}\}$, with $\nu > 1$ and $\mu_{max} \gg 1$ two parameters of the method. Then, we update z^t using $\alpha^t = \mu^t / \|\nabla^t\|^2$. The reason behind this last normalization is because we want to scale the stepsize by the magnitude of the gradient at each step. If condition (43) is not satisfied, then we reduce the value of α^t (for

example, dividing by 2) and we update z^t again until the condition will be eventually satisfied for α^t small enough. We use this strategy to train all the adaptive models we use here, namely ITD, g-ITD, 227 and IOD.

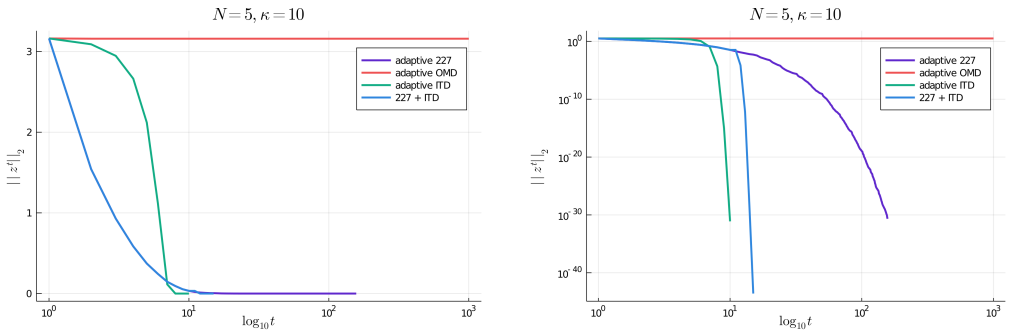
4.1 Examples with adaptive stepsize

Let us study now the optimization problem in the case where both x and y are vectors in more than one dimension. Given a matrix $A \in \mathbb{R}^{(n+m)\times(n+m)}$, let us consider the cost function given by

$$L_5(x, y) = z^T A z = x^T A_{xx} x + y^T A_{yy} y + 2x^T A_{xy} y, \tag{44}$$

where A_{xx}, A_{yy}, A_{xy} and A_{yx} denote again the sub-matrices of A and we assume i) A_{xx} is a positive semi-definite matrix, ii) A_{yy} is negative semi-definite and iii) $A_{xy} = A_{yx}^T$. For simplicity, we are going to consider the case where both A_{xx} and A_{yy} are diagonal matrices with $A_{xx} = -A_{yy}$, where we can change the values in its diagonal and, consequently, they condition number κ. Let's set $n = m = N$. The matrix A_{xy} is sampled with independent standard Gaussian entries. It is easy to check that then L_5 has almost surely just one solution for the minimax problem at $x = y = 0$.

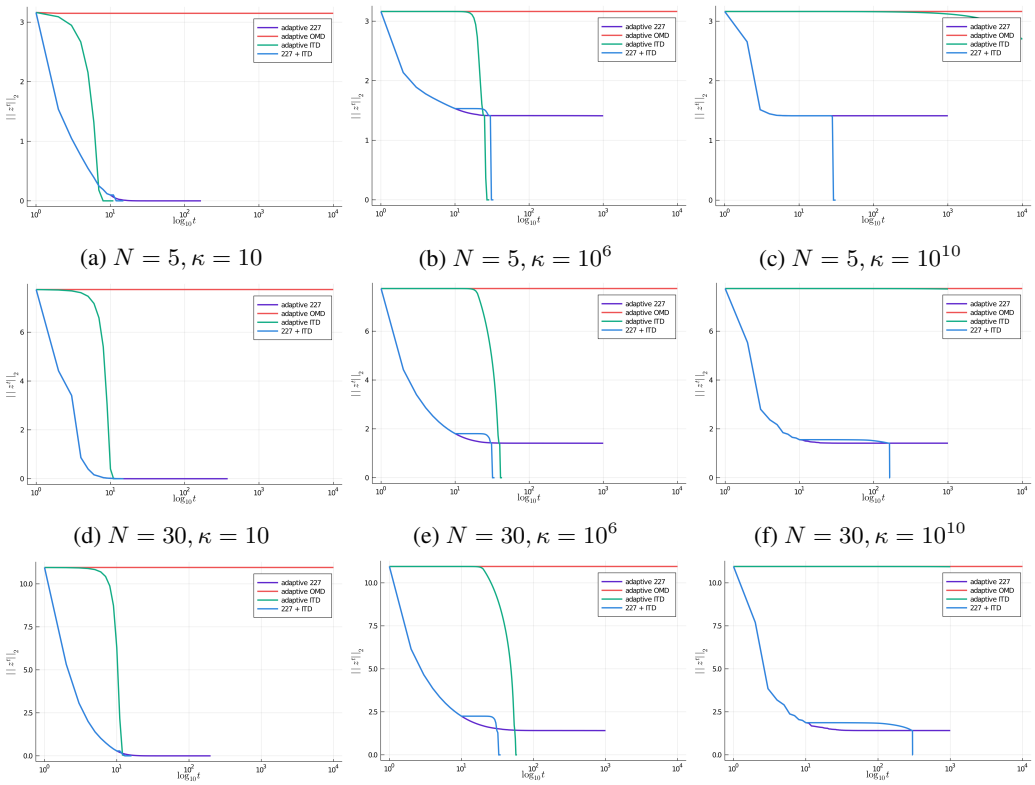
Figure 4 shows the norm $\|z^t\|_2$ as a function of the step for four different methods: Adaptive OMD, Adaptive ITD, Adaptive 227 and a mix of 227 and ITD. In practice, I observed that the 227 method converges slower than ITD, but it makes much more progress in the first few steps. The 227 + ITD method consist on running a few iterations (10 for example) of 227 and then switch to ITD, using the last step of 227 as initialization for ITD. The results displayed in Figure 4 correspond to values of $N = 5$ and $\kappa = 10$. OMD does not use the Hessian and then it is computationally less expensive, but even for this simple problem it does not converge to the right solution. Notice also that ITD does not make much progress in the few steps, but then it converges abruptly to zero.



(a) (b)

Figure 4: *Convergence of the minimax solution for $L_5(x, y)$ for different algorithms with adaptive stepsize. The plot on the right is the same as the one on the left with a logarithm scale in the y axes too.*

Now, it is interesting to analyze what happens when we increase the condition number and the dimension. Since the 227 method does not depend on H_{xx} and H_{yy}, we expect that this method is not going to be too sensitive to the condition number κ. On the other hand, ITD has to perform the inversion of the matrix $(J + \alpha H^t)$, which can affect the performance of the method. Figure 5 shows the performance of these methods for the combinations of values $(N, \kappa) \in \{5, 30, 60\} \times \{10, 10^6, 10^{10}\}$. For all the possible combinations we observe convergence of OMD with adaptive stepsize. For a small condition number ($\kappa = 10$), the three method ITD, 227 and 227 + ITD converge quite fast. For $N = 30$, ITD converges abruptly, in opposition to the 227 method that converges really slowly to zero. For $\kappa = 10^{10}$ ITD does not seem to converge or converges after a lot of iterations. Here is where we take advantage of the fast improvement of the 227 method in the few steps to construct the 227 + ITD method, that shows to converge much faster than the others.



(a) $N = 5, \kappa = 10$ (b) $N = 5, \kappa = 10^6$ (c) $N = 5, \kappa = 10^{10}$

(d) $N = 30, \kappa = 10$ (e) $N = 30, \kappa = 10^6$ (f) $N = 30, \kappa = 10^{10}$

(g) $N = 60, \kappa = 10$ (h) $N = 60, \kappa = 10^6$ (i) $N = 60, \kappa = 10^{10}$

Figure 5: *Convergence of the minimax solution for $L_5(x, y)$ for different algorithms with adaptive stepsize and different values of the dimension N and the condition number κ of A_{xx} and A_{yy}.*

5 Conclusion

In this project we explore and introduce different methods to solve minimization-maximization optimization problems. From the most basic approach given by gradient descent/ascent to the implicit second order methods, we had covered some theoretical foundations and application to some simple examples. The work done along this project can be divided into the theoretical part, the implementation and the further design and analysis of the different experiment.

From a theoretical perspective, there is still much work to be done regarding guarantees of convergence of these methods. As it was pointed out in Daskalakis et al. (2017), it is difficult to prove convergence beyond the convex-concave case. In Mertikopoulos et al. (2019), the authors generalize some results for a more general class of functions. However, it seems difficult in general to guarantee convergence to the solutions. This is particularly hard for minimax problems since we need to guarantee both a vanishing gradient and x^* has to be a local minimum of $L(\cdot, y^*)$ and y^* a local maximum of $L(x^*, \cdot)$. As we saw before, exploiting the adversarial interpretation of minimax problems could lead to convergence of the solutions that seem to satisfy this property. It would be interesting to explore more schemes that exploit this fact. The implementation of these methods involved learning julia. A series of methods were implemented in this language and are available in a public repository. Experiments with different cost functions showed that ITD is quite robust, but bad initialization can lead to a really slow convergence. In this last case, running the 227 method and then switch to ITD converges much faster.

5.1 Final remarks and open questions

I think there are many interesting open questions about minimax optimization not solved yet. I decided to include some thoughts and ideas around this.

1. Even when ITD and its variants performed much better in the examples we saw here, ITD could be computationally expensive and in many problems the dimensions n_x and n_y could be large enough to make impossible the computation of the Hessian and the further matrix inversion. In this line, it is possible to explore approximations to the method that makes this method more useful in general cases. In Essid et al. (2019b), the author also introduced a Quasi implicit twisted gradient descent algorithm.
2. *Extended 227 method:* in the derivation of the 227 method, we observed that x^{t+1} is such that it minimizes a first order approximation of $\|\nabla_y L(x, y^t)\|_2$. In the same spirit, we can think in the anticipatory version of this given by
$$x^{t+1} = \text{argmin}_x \|\nabla_y L(x, y^{t+1})\|_2. \tag{45}$$
I did not explore this in depth, but it is possible to find a new scheme using the same tools we saw in this project (approximation + optimization).
3. Based on the Follow-the-Leader formulation, and in the same spirit we derived the g-ITD method, we can attempt to change the order in which we optimize and approximate and derive the following rule
$$z^{t+1} = \text{argmin}_z \left\{ \sum_{s=1}^{t} \alpha \langle J\nabla^s, z\rangle + \alpha\langle J\nabla^t, z\rangle + \alpha\langle JH^t(z - z^t), z - z^t\rangle + \frac{1}{2}\|z\|_2^2 \right\}. \tag{46}$$
However, it is not straightforward to obtain an iteration rule that relates z^{t+1} with z^t, something like $z^{t+1} = z^t + S^t$ for some vector S^t. This could be another interesting idea to explore.
4. If we have access to the Hessian, we can try to include a second order approximation of the function inside the argmin in equations (7) and (46), instead of the first order approximation $\langle J\nabla^t, z\rangle$ (notice that optimizing a quantity involving this term with respect to z is the same as optimizing $L(z^t) + \langle J\nabla^t, z - z^t \approx L(z^{t+1})$).

REFERENCES

Uri M Ascher. *Numerical methods for evolutionary differential equations*, volume 5. Siam, 2008.

Constantinos Daskalakis, Andrew Ilyas, Vasilis Syrgkanis, and Haoyang Zeng. Training gans with optimism. *arXiv preprint arXiv:1711.00141*, 2017.

Montacer Essid, Debra F Laefer, and Esteban G Tabak. Adaptive optimal transport. *Information and Inference: A Journal of the IMA*, 8(4):789–816, 2019a.

Montacer Essid, Esteban Tabak, and Giulio Trigila. An implicit gradient-descent procedure for minimax problems. *arXiv preprint arXiv:1906.00233*, 2019b.

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*, pp. 2672–2680, 2014.

Panayotis Mertikopoulos, Bruno Lecouat, Houssam Zenati, Chuan-Sheng Foo, Vijay Chandrasekhar, and Georgios Piliouras. Optimistic mirror descent in saddle-point problems: Going the extra(-gradient) mile. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=Bkg8jjC9KQ`.

Shai Shalev-Shwartz. Online learning and online convex optimization. *Foundations and Trends® in Machine Learning*, 4(2):107–194, 2012. ISSN 1935-8237. doi: 10.1561/2200000018. URL `http://dx.doi.org/10.1561/2200000018`.