Lecture notes on Optimization for ML

CS295 Optimization for Machine Learning

Instructor: Ioannis Panageas
Scribed by: Hadi Khodabandeh, Amisha Priyadarshini, Yasaman Razheghi

Lecture 1, 2. Convex Optimization and Gradient Descent.

1 Introduction

The optimization problems arises in many disciplines such as computer science and machine learning, Engineering, operational research, economics and transportation. It is simply a computational problem in which the objective is to find the best of all the feasible solutions.

2 What is optimization?

In simple description we have give a function like $f : R^n \to R$ and we want to find the minimum point of the function f, i.e we want to solve the $min_{x\in\chi} f(x)$. Mathematically speaking an optimization problem has the form of,

minimize $f_0(x)$
subject to $f_i(x) \leq b_i$, where $i = 1, ..., m$

x is a vector in the above equation and is known as the *optimization variable* of the problem. The function $f_0(x) : R^n \to R$ is the $objective function$, the functions $f_i : R^n \to R$ are the *constraint* $functions$ and the constants, $b_1, ..., b_m$ are the bounds for the constraints.
Generally, we can solve the optimization problem in two scenarios based on the type of the constraints used in the problem,

1: Unconstrained, when $\chi = R^n$
2: Constrained, when $\chi \subset R^n$.

3 Why should we study optimization?

Optimization problems give us a tool to solve many real-work applications like prediction of scenarios in stock-market, finding the fastest route in transportation, working with neural networks, operations supply chain management and so on. In Machine Learning, optimization is a procedure of adjusting the hyper-parameters in order to minimize the cost function by using one of the optimization techniques. This minimization is relevant as it describes the discrepancy between the true value of the estimated parameter and the model's prediction.

4 Convex Optimization and Gradient Descent: Basics

4.1 Goal in Machine learning Problems:

Many machine learning problems involve learning parameters $\theta \in \Theta$ of a function, toward achieving an *objective*. Usually in machine learning we define the *Objectives* by a loss function $L : \Theta \to R$

Supervised learning example In supervised learning we are usually given n i.i.d samples (x_i, y_i) where x is the input and y is the *gold label*. We have a prediction function $f(x_i, \theta)$ which predicts the output given the input x_i. We define a loss function for each sample as $l(f(x_i, \theta), y_i)$ and the goal is to find the parameters $\theta \in \Theta$ which minimizes the overall loss function $L(\theta) = 1/n \sum_{i=1}^{n} l(f(x_i, \theta), y_i)$. This means that we want to minimize the distance between the model's prediction and the actual label of the input.

Formally, we can define the goal as $min_{\theta \in \Theta} L(\theta)$. Note that typically solving the $min_{x \in \chi}$ is NP-hard(computationally intractable, most likely it needs exponential time to be solved).

However, for some certain classes of function f, there exist some strong theoretical guarantees and efficient optimization algorithms. Like in case of the Convex function f which we can solve efficiently with Gradient Descent algorithm.

5 Definitions

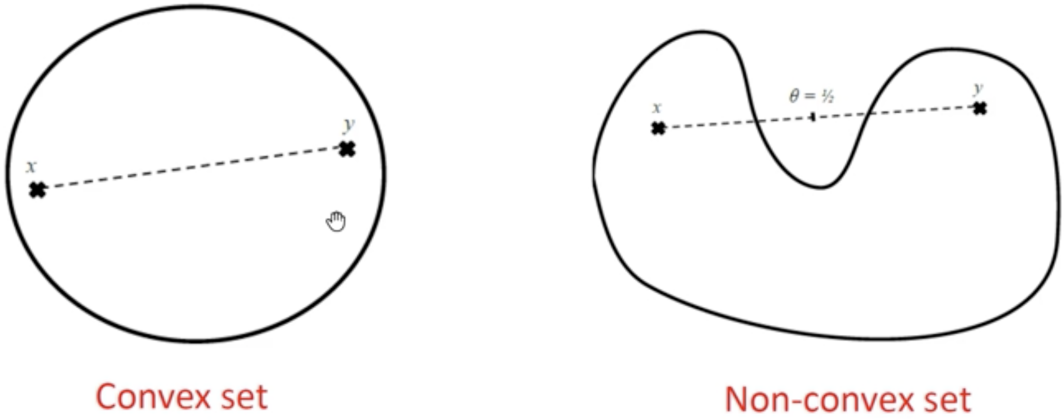
5.1 Definition Convex combination

$z \in R^D$ is a convex combination of $x_1, x_2, x_n \in R^d$ if $z = \sum \lambda_i x_i, \lambda_i \geq 0 for$ all i $and \sum_i \lambda_i = 1$

5.2 Definition Convex set

χ is a convex set if the convex combination of any two points in χ belongs also in χ. An example of convex set is shown in Figure 1.

Figure 1: Definition of convex set, on the left we can see a convex set, for all x and y points in the set, the line connecting these two points should also belong to the convex set. On the right there is a non-convex set, since we can see that the median of the points x and y is outside of the convex set.



5.3 Definition Convex function

A function $f(x)$ is convex if and only if the domain $dom(f)$ is a convex set and $\forall x, y \in dom(f), \theta \in [0, 1]$:

$$f(\theta x + (1 - \theta)y) \leq \theta f(x) + (1 - \theta)f(y)$$

In Figure 2 we can see an illustration of a convex and non-convex function. The line connecting any two points in the domain of f should be above the function as it is shown in Figure 2 on the left side.

5.4 Definition Concave function

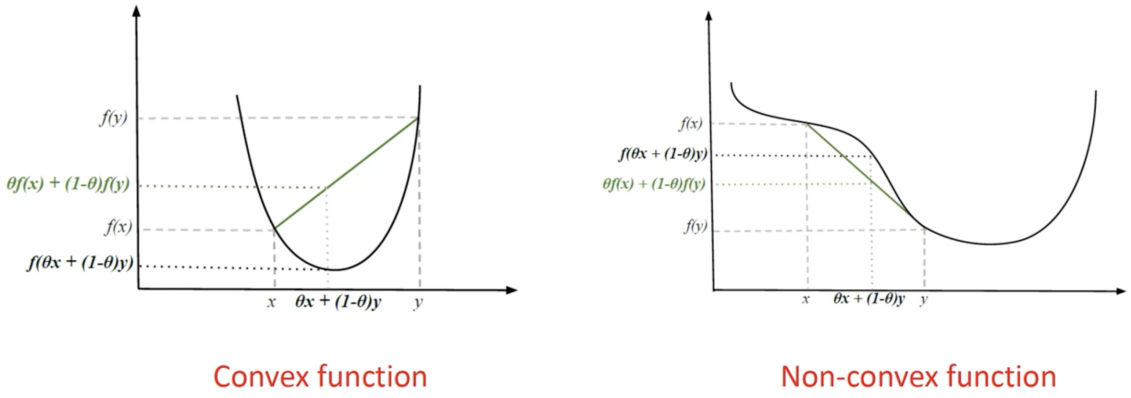
A function f is *Concave* as long as the function $-f$ is *convex*. In other words, a function $f : -f$ should be convex in order for f to be *concave*, i.e the inequality in equation 5.3 should be reversed! Moreover, if the inequality is strict, f is called *strictly convex*.

6 Basic Facts

Lemma 6.1 (First order condition for convexity) A *differentiable* function $f(x)$ is *convex* if and only if the domain $dom(f)$ is a convex set $\forall$x,y$\in$dom(f).

$$f(y) \leq f(x) + \nabla f(x)^T (y - x)$$

Figure 2: Definition of convex function, on the left we can see a convex function, for all x and y points in the domain of f, the line connecting these two points should be above the function f values. On the right there is a non-convex function.



The above equation also means that the value at $f(y)$ dominated the first order approximation of the Taylor expansion with respect to point x. Also, note that the above lemma is *if and only if* which means that if the above condition is met for all x and $y \in dom(f)$ then the function f is convex; also, if the function f is *convex*, then the inequality holds $\forall\ x,\ y \in$dom(f).

Note that the *differentiable* condition should be met because we need to know that the $\nabla f(x)$ exists.

Proof:[$\Longrightarrow$]By convexity we have (for all $t > 0$) :

$$f(ty + (1-t)x) \leq tf(y) + (1-t)f(x).$$

Rearranging the above equation:

$$f(x + t(y-x)) \leq t(f(y) - f(x)) + f(x)$$

Dividing by t we conclude:

$$f(y) - f(x) \leq \frac{f(x + t(y-x)) - f(x)}{t}$$

We need to take the limit as t goes to 0 to get the directional derivative and can have the derivative in the left side of lemma 6.1.

$$f(y) - f(x) \geq \underbrace{\lim_{t \to 0} \frac{f(x + t(y-x)) - f(x)}{t}}_{\text{directional derivative}} = \nabla f(x)^T (y-x)$$

■

Now we want to prove the other side which means that we know the inequality is valid for all x and y in $dom(f)$ and we want to prove that the function f is convex.

Proof:[$\Longleftarrow$]

We choose two points x and y in $dom(x)$. We define z as a convex combination of x and y,

$$z = tx + (1-t)y \text{ for } t \in (0,1)$$

Moreover it holds that,

$$f(x) \geq f(z) + \nabla f(z)^T (x - z).$$

Multiplying by t, we obtain:

tf(x) $\geq tf(z) + t\nabla f(z)^T(x-z) f(y) \geq f(z) + \nabla f(z)^T(y-z)$.

Multiplying by $(1-t)$, we obtain:

(1-t)f(y) $\geq (1-t)f(z) + (1-t)\nabla f(z)^T(y-z)$

Adding them both up, it holds that:

$$tf(x) + (1-t)f(y) \geq tf(z) + t\nabla f(z)^T(x-z) + (1-t)f(z) + (1-t)\nabla f(z)^T(y-z)$$

■

Lemma 6.2 (Second order condition for convexity) *A twice differentiable function f(x) is convex if and only if the domain $dom(f)$ is a convex set $\forall x \in dom(f)$:*

$$\nabla^2 f(x) \geq 0$$

In words, the Hessian of f should be positive semi-definite.

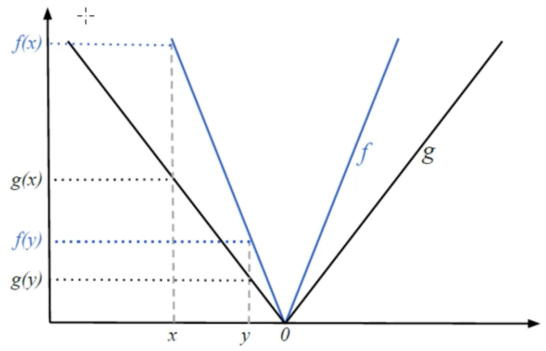
Definition 6.1 *The Hessian function is a second order partial derivative of f as in a Matrix:*

$$Hf = \begin{bmatrix} \frac{\partial f}{\partial {x_1}^2} & \frac{\partial f}{\partial x_1 \partial x_2} & \frac{\partial f}{\partial x_1 \partial x_3} & \cdots & \frac{\partial f}{\partial x_1 \partial x_n} \\ \frac{\partial f}{\partial x_2 \partial x_1} & \frac{\partial f}{\partial {x_2}^2} & \frac{\partial f}{\partial x_2 \partial x_3} & \cdots & \frac{\partial f}{\partial x_2 \partial x_n} \\ \cdot & & & & \\ \cdot & & & & \\ \cdot & & & & \\ \frac{\partial f}{\partial x_n \partial x_1} & \frac{\partial f}{\partial x_n \partial x_1} & \frac{\partial f}{\partial x_n \partial x_3} & \cdots & \frac{\partial f}{\partial {x_n}^2} \end{bmatrix}$$

Note that *Positive demi-definite* meas that all eigen-values of the above matrix should be greater or equal to 0. The Hessian is a symmetric function and it has real eigen-values.

Definition 6.2 (Lipschitz function) *A function $f : R^d \to R^{d'}$ is L-Lipschitz continuous iff for all $L > 0$, $\forall x, y \in dom(f)$,remains bounded when moved from x to y, i.e., $||f(x) - f(y)||_2 \leq L||x - y||_2$. Meaning that the perturbations would make small and bounded changes in the output. In other words, small changes in the input would result in small changes in the output.*

Figure 3: L_f Lipschitz continuous function f and a L_g Lipschitz continuous function g with $L_f > L_g$, here the slope of the g function is less than the slope of the f function. Note that for the linear functions, the Lipschitz constant is their slope.



Definition 6.3 (Smoothness) *A continuously differentiable function $f(x)$ is L-smooth if its gradient is L-Lipschits, i.e., $\exists\ L > 0$, $\forall x, y \in dom(f)$:*

$$||\nabla f(x) - \nabla f(y)||_2 \leq L||x - y||_2$$

Meaning the gradient of the function should be L-Lipschits.

Definition 6.4 (Strong convexity) *A function $f(x)$ is α-strongly convext if for $\alpha > 0$ and $\forall x \in dom(f)$:*

$$f(x) - \frac{\alpha}{2}||x||_2^2 \textit{is convex}$$

Note that a strongly-convex function is also a convex function.

7 Minimizing Convex Fucntions

Lemma 7.1 (Gradient zero) *Let $f : R^d \to R$ be differentiable and convex. x^* is a minimizer if and only if $\nabla f(x*) = 0$. Hence all the minimizers give the same f-values.*

Proof:[⟵]By First Order Condition for convexity we have $\forall x \in dom(f)$

$$f(x) \geq f(x*) + \nabla f(x*)T(x - x*) = f(x*)$$

■

Proof:[⟶]Choose $t > 0$ small enough $\ni y := x * -t\nabla f(x*)$ is in $dom(f)$.

By Taylor approximations we have:

$$f(y) - f(x*) = \nabla f(x*)^T(y - x*) + o(||y - x * ||_2)$$

$$= -t||\nabla f(x*)||_2^2 + o(||t\nabla f(x*)||_2)$$

For very small value of t, $f(y) - f(x*) < 0$ if $\nabla f(x*) \neq 0$

This is a contradiction, since we assumed that $x*$ is a minimizer and the left hand should be positive. ■

For the above proof 7 we use the equality of $\nabla f(x*)^T f(x*) = ||\nabla f(x*)||_2$.

With the lemma pointed in 7.1 when we need to find the minimizer of a function, we need to find the point where the gradient of the function is equal to 0.

8 Gradient Descent (GD)

Definition 8.1 (Gradient Descent) *Let $f : R^D \to R$ be* differentiable *which we want to minimize. Iteratively, using the equation $x_{k+1} = x_k - \alpha \nabla f(x_k)$, is called the* gradient descent algorithm.

8.1 Remarks on GD

- *α is called the* stepsize*(Explanation: Intuitively, the smaller the stepsize, the slower the algorithm).*
- *α may or may not depend on k.*
- *If GD converges, it means that $(x) \to 0$, so we should have convergence to the minimizer (for the convex f).*
- *the minimizer of f are fixed points of GD. (Explanation: if you start from the minimizer, you will remain on that point.)*

9 Analysis of GD for L-Lipschits

Theorem 9.1 (Gradient Descent) *Let $f : R^d \to R$ be differentiable, convex and L-Lipschitz which we want to minimize. Let $R = ||x_1 - x * ||_2$ (R represents the distance between the initial point x_0) and the minimizer $x*$. It holds for $T = \frac{R^2 L^2}{\epsilon^2}$,then:*

$$f(\frac{1}{T}\sum_{t=1}^{T} x_t) - f(x^*) \leq \epsilon$$

for an appropriate value of $\alpha = \frac{\epsilon}{L^2}$.

9.1 Remarks

- The speed of convergence (with number of iterations equal to T) is independent of the dimension, d. We call these types of algorithms *dimension free* algorithms.
- This result give a convergence rate of $O(\frac{1}{\epsilon^2})$. In the theorem 10.1 we will see that with smoothness we can have a faster convergence rate of $O(\frac{1}{\epsilon})$.
- There is Nesterov's acccelerated method that can achieve the convergence rate of $O(\frac{1}{\sqrt{\epsilon}})$ with the smoothness assumption.

- If the f function also meet the *smoothness* and *strong-convexity*, then the f can achieve the convergence rate of $O(ln\frac{1}{\epsilon})$
- The theorem does not imply pointwise convergence $f(x_T) \to f(x)$. This means that the last iteration of the GD algorithm is *not* necessarily the closest point to the minimizer and as it is stated in the left side of the above equation, it is important to calculate the average of all the points that GD visited.

10 Gradient Decent for L-smooth Functions

Now we will cover the gradient decent for L-smooth functions. As we mentioned earlier, a L-smooth function is a function that its gradient is L-Lipschitz, meaning that if we perturb the input by a small amount the gradient does not change by much.

So assuming that f is a L-smooth function, we can show the following theorem.

Theorem 10.1 (Gradient Decent) *Let $f : \mathbb{R}^d \to \mathbb{R}$ be differentiable, convex (want to minimize) and L-smooth. Let $R = \|x_1 - x^*\|_2$. It holds for $T = \frac{2R^2L}{\epsilon}$*

$$f(x_{T+1}) - f(x^*) \leq \epsilon$$

with appropriately choosing $\alpha = \frac{1}{L}$.

This theorem shows that after around $\frac{L}{\epsilon}$ steps, where L is the Lipschitz constant, we are going to reach to ϵ-neighborhood of the minimizer. Also note that the choice of the step size is different from the case that the function was L-Lipschitz. Now the step size is constant $\alpha = \frac{1}{L}$ and does not depend on ϵ. Again, this theorem is independent of the dimension of our function, so we call it dimension-free.

Another important aspect of Theorem 10.1 is that it does not require taking the average of the values of the function, and it implies convergence of the value of the last iteration to the value of the minimizer.

Now we intend to prove Theorem 10.1. First we show a claim about L-smooth functions,

Claim 10.2 *Let f be a differentiable and L-smooth, then*

$$f(y) - f(x) - \nabla f(x)^\top (y - x) \leq \frac{L}{2}\|x - y\|_2^2$$

Note that this claim does not require the function to be convex. This claim basically shows that if we take the value of the function at some point y and subtract the first order approximation of it with respect to another point x, then the result would be within $\frac{L}{2}$ factor of the l-2 norm squared of x and y. A similar inequality might be achievable using Taylor expansion, but the constant $\frac{L}{2}$ that we prove here turns out to be important. Now we prove this claim

Proof: The idea is to express the difference on the left side of the inequality as an integral along the direction $y - x$,

$$\begin{aligned} f(y) - f(x) - \nabla f(x)^\top (y - x) &= \int_0^1 \nabla f(y + t(x - y))^\top (x - y) dt - \nabla f(y)^\top (x - y) \\ &= \left(\int_0^1 \nabla f(y + t(x - y)) dt - \nabla f(y) \right)^\top (x - y) \\ &= \left(\int_0^1 \{\nabla f(y + t(x - y)) - \nabla f(y)\} dt \right)^\top (x - y) \\ &\leq L \int_0^1 t dt \|x - y\|_2^2 = \frac{L}{2} \|x - y\|_2^2 \end{aligned}$$

The last inequality follows directly from the definition of L-smoothness. ■

The next claim that we want to show uses convexity as well as the other two conditions that we had in the previous claim.

Claim 10.3 *Let f be a differentiable, convex and L-smooth, then*

$$f(x^*) - f(x) \leq f(x - \frac{1}{L} \nabla f(x)) - f(x) \leq -\frac{1}{2L} \|\nabla f(x)\|_2^2$$

Note that in the middle term in the inequality, $x - \frac{1}{L} \nabla f(x)$ is simply the next point in gradient decent if we start from x and we choose a step size of $\frac{1}{L}$. So the middle part of the inequality is basically the difference between the values of two consecutive steps of gradient decent. Given the fact that the right side of inequality is always negative (the norm is always positive), this inequality is showing that the value of the value of the function is decreasing after each step, by a factor of the gradient of the function at that step.

Another observation is that if the gradient at some point is zero, it means that we have reached the minimizer. It is also worth to mention that if the gradient is small, we will have small improvement, and if the gradient is large, we will have large improvement. Now we prove this claim,

Proof: Set $z = x - \frac{1}{L} \nabla f(x)$. The first inequality is trivial by the definition of the minimizer (we have $f(x^*) \leq f(z)$). According to Claim 10.2,

$$\begin{aligned} f(z) - f(x) &\leq \nabla f(x)^\top (z - x) + \frac{L}{2} \|z - x\|_2^2 \\ &= -\frac{1}{L} \nabla f(x)^\top \nabla f(x) + \frac{L}{2} \frac{1}{L^2} \|\nabla f(x)\|_2^2 \\ &= \frac{1}{2L} \|\nabla f(x)\|_2^2 \end{aligned}$$

We simply replaced $z = x - \frac{1}{L} \nabla f(x)$ in order to achieve the second equality. ■

Note that this claim is the main reason that proves gradient decent converges. For example, if we have a compact space and the points are decreasing according to this claim, then we know that they are converging to the minimizer.

Now we can prove the main theorem of this section, Theorem 10.1.

Proof: Assume $\|x_t - x^*\|_2$ is decreasing in t (Exercise 4 to prove). Using Claim 10.3,

$$f(x_{t+1}) - f(x_t) \leq -\frac{1}{2L}\|\nabla f(x_t)\|_2^2$$

From convexity we get,

$$\begin{aligned} f(x_t) - f(x^*) &\leq \nabla f(x_t)^\top (x_t - x^*) \\ &\leq \|\nabla f(x_t)\|_2 \|x_t - x^*\|_2 \text{ (C-S inequality)} \\ &\leq \|\nabla f(x_t)\|_2 \|x_0 - x^*\|_2 \text{ (Assumption)} \end{aligned}$$

We used convexity for the first inequality and Cauchy-Schwarz inequality on the dot product to obtain the second inequality. If we sum up the last two inequalities,

$$f(x_{t+1}) - f(x^*) - (f(x_t) - f(x^*)) \leq -\frac{1}{2L}\frac{(f(x_t) - f(x^*))^2}{R^2}$$

Setting $\delta_t = f(x_t) - f(x^*)$, we get $\delta_{t+1} \leq \delta_t - \frac{\delta_t^2}{2LR^2}$. Now we want to find an upper bound on δ_t. Let's divide both sides of the equivalent inequality $\delta_{t+1} - \delta_t \leq -\frac{\delta_t^2}{2LR^2}$ by $\delta_t \delta_{t+1}$,

$$\frac{1}{\delta_t} - \frac{1}{\delta_{t+1}} \leq -\frac{\delta_t}{2LR^2\delta t + 1}$$

Note that $\delta_{t+1} \leq \delta_t - \frac{\delta_t^2}{2LR^2} \leq \delta_t$, so $\frac{\delta_t}{\delta_{t+1}} \geq 1$ and because the right side of the above inequality is negative,

$$\frac{1}{\delta_t} - \frac{1}{\delta_{t+1}} \leq -\frac{1}{2LR^2}$$

If we sum up all these inequalities from 1 to t, we get

$$\frac{1}{\delta_1} - \frac{1}{\delta_{t+1}} \leq -\frac{t}{2LR^2}$$

But δ_1 is a non-negative constant, so we conclude that $\delta_t \leq \frac{2LR^2}{t-1}$ which completes the proof of Theorem 10.1. ■

11 Gradient Decent for μ-convex Functions

Now we consider gradient decent for L-smooth and strongly convex functions. So the gradient is L-Lipschitz and $f(x) - \frac{\mu}{2}\|x\|_2^2$ is also convex. The following theorem holds for such function f.

Theorem 11.1 (Gradient Decent) *Let $f : \mathbb{R}^d \to \mathbb{R}$ be differentiable, μ-strongly convex (want to minimize) and L-smooth. Let $R = \|x_0 - x^*\|_2$. It holds for $T = \frac{2L}{\mu}\ln\left(\frac{R}{\epsilon}\right)$*

$$\|x_T - x^*\|_2 \leq \epsilon$$

with appropriately choosing $\alpha = \frac{1}{L}$.

So the choice of the step size does not depend on the convexity parameter, but only on the Lipschitz constant, and the convergence is exponentially faster than what we discussed for L-smooth functions in the previous section. Another important observation is that we not only have convergence on the value, but also the actual points are converging to the minimizer. In the case of regular convex functions there could be an interval of minimizers, making the minimizer non-unique, but with the assumption of strongly convexity, we know that the minimizer is also unique. So we have the strongest notion of convergence here.

In order to prove this theorem, we not only show that the distance between the points and the minimizer decreases, but we also show that it decreases by a constant factor, which makes the convergence to happen exponentially faster. The proof, as we will see, is much simpler than what we had for L-smooth functions.

Proof: It holds that

$$\begin{aligned}\|x_T - x^*\|_2^2 &= \left\|x_{T-1} - \frac{1}{L}\nabla f(x_{T-1}) - x^*\right\|_2^2 \\ &= \|x_{T-1} - x^*\|_2^2 + \frac{1}{L^2}\|\nabla f(x_{T-1})\|_2^2 - 2\frac{1}{L}\nabla f(x_{T-1})^\top (x_{T-1} - x^*)\end{aligned}$$

We intend to bound the last two terms in terms of the first term. From Exercise 2 and then Claim 10.3 we get

$$\begin{aligned}\frac{2}{L}\nabla f(x_{T-1})^\top (x^* - x_{T-1}) &\leq \frac{2}{L}(f(x^*) - f(x_{T-1})) - \frac{\mu}{L}\|x^* - x_{T-1}\|_2^2 \\ &\leq -\frac{1}{L^2}\|\nabla f(x_{T-1})\|_2^2 - \frac{\mu}{L}\|x^* - x_{T-1}\|_2^2\end{aligned}$$

Putting together this inequality and the equality we had earlier,

$$\|x_T - x^*\|_2^2 \leq (1 - \frac{\mu}{L})\|x_{T-1} - x^*\|_2^2$$

Writing this inequality for 1 to T and combining them together,

$$\|x_T - x^*\|_2^2 \leq (1 - \frac{\mu}{L})^T R^2 \leq e^{-\frac{\mu T}{L}} R^2$$

■

So far we considered three cases of gradient decent on functions with different assumptions. Hence we completed the analysis of gradient decent.

12 Projected Gradient Decent (PGD)

In this section we consider projected gradient decent, where we can have some constraints on the minimizer. We define the projected gradient decent formally in the following way,

Definition 12.1 (Projected Gradient Decent) *Let $f : \mathbb{R}^d \to \mathbb{R}$ be differentiable (want to minimize) in some compact convex set $\mathcal{X}$. The algorithm below is called projected gradient decent*

$$x_{k+1} = \Pi_{\mathcal{X}}(x_k - \alpha\nabla f(x_k))$$

As you see gradient decent is not applicable on this problem, as the iterations may go outside the set $\mathcal{X}$ and we may find a minimizer that does not belong to the set. So we somehow need to force the gradient decent to remain inside the compact set, and we do that by projecting. So the algorithm is basically the same, we find the next step of gradient decent, $x_k - \alpha \nabla f(x_k)$, and then we project the result using the operator $\Pi_{\mathcal{X}}$. Note that this implies that the gradient is not necessarily zero at the minimizer we find in $\mathcal{X}$, because the actual minimizer where the gradient is zero at can lie outside the set $\mathcal{X}$.

It is worth to mention that typically projection is not an easy task to do, as it might be inefficient or intractable, and the projection itself can be a separate minimization problem. Intuitively, projection means to find a point in the set that has the minimum distance to our point. It surely depends on the metric that we use, but here we only consider Euclidean distance. So projection here is an l_2 minimization problem.

Given this, we assume that the projection $\Pi_{\mathcal{X}}$ works as a black-box. And we intend to analyse the performance of the projected gradient decent given this black-box. We will cover the analysis of PGD for L-Lipschitz functions in this section, the analysis for the cases of L-smoothness and strongly convexity would be exactly the same as the ones that we discussed in previous sections.

We have the following theorem for projected gradient decent for L-Lipschitz functions,

Theorem 12.1 (Projected Gradient Decent) *Let $f : \mathbb{R}^d \to \mathbb{R}$ be differentiable, convex (want to minimize in some compact set $\mathcal{X}$) and L-Lipschitz. Let $R = \|x_1 - x^*\|_2$, the distance between the initial point x_0 and the minimizer x^*. It holds for $T = \frac{R^2 L^2}{\epsilon^2}$*

$$f\left(\frac{1}{T}\sum_{t=1}^{T} x_t\right) - f(x^*) \leq \epsilon$$

with appropriately choosing $\alpha = \frac{\epsilon}{L^2}$.

We can observe that we have exactly the same guarantees as in the unconstrained case. Now we prove the theorem.

Proof: Set $y_t := x_t - \alpha \nabla f(x_t)$. It holds that

$$\begin{aligned}
f(x_t) - f(x^*) &\leq \nabla f(x_t)^\top (x_t - x^*) \ \text{(FOC from convexity)} \\
&= \frac{1}{\alpha}(x_t - y_t)^\top (x_t - x^*) \ \text{(definition of GD)} \\
&= \frac{1}{2\alpha}\left(\|x_t - x^*\|_2^2 + \|x_t - y_t\|_2^2 - \|y_t - x^*\|_2^2\right) \ \text{(law of Cosines)} \\
&= \frac{1}{2\alpha}\left(\|x_t - x^*\|_2^2 - \|y_t - x^*\|_2^2\right) + \frac{\alpha}{2}\|\nabla f(x_t)\|_2^2 \ \text{(Def of } y_t\text{)} \\
&= \frac{1}{2\alpha}\left(\|x_t - x^*\|_2^2 - \|y_t - x^*\|_2^2\right) + \frac{\alpha L^2}{2}
\end{aligned}$$

So far we had the same approach as in the unconstrained case. But note that y_t might not be in $\mathcal{X}$. The following claim helps to resolve this issue.

Claim 12.2 *It is true that*

$$(\Pi_{\mathcal{X}}(y) - x)^\top (\Pi_{\mathcal{X}}(y) - y) \leq 0$$

Proof: By picture.

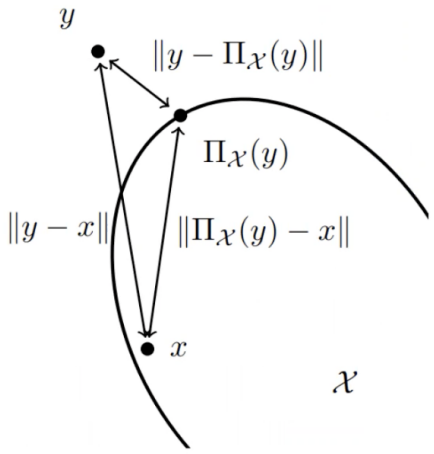


Figure 4: Proof of the claim

■

Corollary 12.3 *It is true that (law of Cosines)*

$$\|y - x\|_2^2 \geq \|\Pi_{\mathcal{X}}(y) - y\|_2^2 + \|\Pi_{\mathcal{X}}(y) - x\|_2^2$$

If we use this corollary for $y = y_t$ and $x = x^*$ we get

$$\begin{aligned} \|y_t - x^*\|_2^2 &\geq \|x_{t+1} - y_t\|_2^2 + \|x_{t+1} - x^*\|_2^2 \\ &\geq \|x_{t+1} - x^*\|_2^2 \end{aligned}$$

which basically proves that projection is helping to get closer to the minimizer, not further. So we have the following inequality

$$f(x_t) - f(x^*) \leq \frac{1}{2\alpha} \left(\|x_t - x^*\|_2^2 - \|x_{t+1} - x^*\|_2^2 \right) + \frac{\alpha L^2}{2}$$

Now we can create the same telescopic sum as we did in the unconstrained case to obtain the result.

■

As we mentioned earlier the same trick of substituting y_t by its projection can be used to prove the bounds for L-smooth functions and strongly convex functions.

13 Conclusion

We covered the basics and we also proved the following convergence rates for gradient decent for different functions,

- GD has a rate of convergence $\mathcal{O}(L^2/\epsilon^2)$ for L-Lipschitz functions.
- GD has a rate of convergence $\mathcal{O}(L/\epsilon)$ for L-smooth functions.
- GD has a rate of convergence $\mathcal{O}(L/\mu \ln \frac{1}{\epsilon})$ for L-smooth μ-convex functions.

We also proved the same bounds for projected gradient decent.

References

[1] Stephen Boyd, Dept. of EE, Stanford University. *Convex Optimization.* Cambridge University Press, 2009.

[2] Shai Shalev-Shwartz Shai Ben-David. *Understanding Machine Learning from Theory to Algorithms.* Cambridge University Press, 2014.

[3] Sebastian Bubeck. *Convex Optimization: Algorithms and Complexity.* Foundations and Trends in Machine Learning, 2015.

CS295 Optimization for Machine Learning

Instructor: Ioannis Panageas Scribed by: Berken Utku Demirel, Oyku Deniz Kose

Lecture 3. Subgradients and Stochastic Gradient Descent.

1 Introduction

Definition (Subgradients): Let $f(x) : \mathcal{X} \to \mathbb{R}$ be a function, with $\mathcal{X} \subset \mathbb{R}^d$. $g_x \in \mathbb{R}^d$ is called a subgradient of f at x if for all $y \in \mathcal{X}$ following holds:

$$f(y) - f(x) \geq g_x^\top (y - x).$$

The set of subgradients at x can be defined, and it is denoted by $\partial f(x)$.
Example: $|x|$, which is not differentiable at $x = 0$.

$$f(y) - f(0) \geq g_0(y - 0),$$
$$|y| \geq g_0.y \to g \in [-1, 1].$$

Therefore, the subgradient of $|x|$ at 0 is a set.

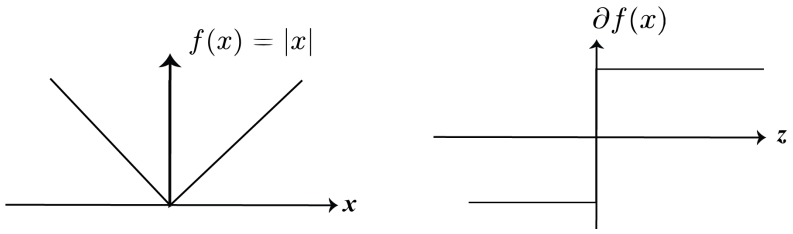


Figure 1: The absolute value function $|x|$ (left), and its subdifferential $\partial f(x)$ as a function of x (right).

Remark: If $f(x)$ is differentiable, g_x coincides with the gradient.

Lemma 1.1 *(Existence and convexity). Let $f : \mathcal{X} \to \mathbb{R}$ be a function such that $\partial f(x) \neq \varnothing$ for all x. It holds that f is convex*

Proof: Since $\partial f(x) \neq \varnothing$, it holds that there exists a vector g such that

$$f(ty + (1-t)x) - f(x) \leq g^\top t(y - x) \quad (1)$$
$$f(ty + (1-t)x) - f(y) \leq g^\top (1-t)(x - y) \quad (2).$$

Multiplying inequality (1) with $(1 - t)$ and inequality (2) with (t), and summing the resulted inequalities, the following inequality can be written:

$$f(ty + (1-t)x) \leq (1-t)f(x) + tf(y).$$

Converse is also true, and can be shown utilizing Supporting Hyperplane Theorem.

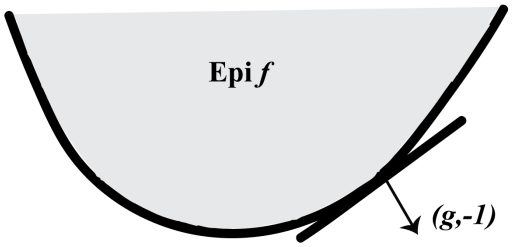


Figure 2: A vector $g \in \mathbf{R}^n$ is a subgradient of f at x if and only if vector $(g, -1)$ defines a supporting hyperplane to epi f at $(x, f(x))$

Lemma 1.2 *(Local minima are global minima). Let $f : \mathcal{X} \to \mathbb{R}$ be a convex function. If x is a local minimum then it is a global minimum. This happens if and only if $\underline{0} \in \partial f(x)$.*

Proof: If x is global minimizer, using the definition of subgradient:

$$f(y) - f(x) \geq g_x^\top (y - x),$$

it is obvious that $\underline{0}$ is the trivial solution for g_x. Therefore, if x is global minimizer, $\underline{0} \in \partial f(x)$. Moreover, from convexity, for $t > 0$ small enough,

$$f(x) \leq f(tx + (1-t)y) \leq tf(x) + (1-t)f(y).$$

Hence $f(x) \leq f(y)$.

Definition (Revisited Gradient Descent): Let the function $f : \mathbb{R}^d \to \mathbb{R}$ be convex and not necessarily differentiable in some convex set $\mathcal{X}$. Gradient descent is defined iteratively:

$$x_{k+1} = x_k - \alpha g_{x_k}.$$

Remarks

- $g_{x_k} \in \partial f\,(x_k)$ is the subgradient computed at x_k.
- Same guarantees as the classic and projected gradient descent apply.

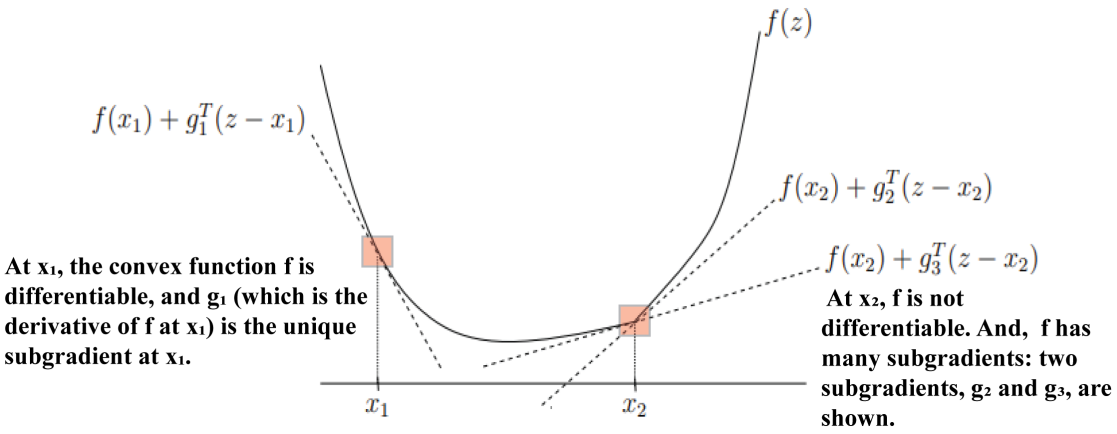


Figure 3: An example of subdifferentiable function f

2 Analysis of Gradient Descent for L-Lipschitz

Theorem 2.1 *(Gradient Descent). Let $f : \mathbb{R}^d \to \mathbb{R}$ be not necessarily differentiable, convex (want to minimize) and L-Lipschitz. Let $R = \|x_1 - x^*\|_2$ be the distance between the initial point x_1 and minimizer x^*. It holds for $T = \frac{R^2L^2}{\epsilon^2}$,*

$$f\left(\frac{1}{T}\sum_{t=1}^{T} x_t\right) - f(x^*) \leq \epsilon,$$

with appropriately choosing $\alpha = \frac{\epsilon}{L^2}$.

Proof: It holds that

$$\begin{aligned}
f(x_t) - f(x^*) &\leq g_{x_t}^\top (x_t - x^*) \text{ from the definition of subgradient,} \\
&= \frac{1}{\alpha}(x_t - x_{t+1})^\top (x_t - x^*) \text{ from the definition of gradient descent,} \\
&= \frac{1}{2\alpha}\left(\|x_t - x^*\|_2^2 + \|x_t - x_{t+1}\|_2^2 - \|x_{t+1} - x^*\|_2^2\right) \text{ from law of Cosines,} \\
&= \frac{1}{2\alpha}\left(\|x_t - x^*\|_2^2 - \|x_{t+1} - x^*\|_2^2\right) + \frac{\alpha}{2}\|g_{x_t}\|_2^2 \text{ from the definition of gradient descent,} \\
&\leq \frac{1}{2\alpha}\left(\|x_t - x^*\|_2^2 - \|x_{t+1} - x^*\|_2^2\right) + \frac{\alpha L^2}{2} \text{ from Exercise 3.}
\end{aligned}$$

(Exercise 3 (General case): Suppose $f(x)$ is L -Lipschitz, continous, and $\partial f(x) \neq \varnothing$. Then $\forall x \in \text{dom}(f)$

$$\|g_x\|_2 \leq L \text{ where } g_x \in \partial f(x).)$$

Following

$$f(x_t) - f(x^*) \leq \frac{1}{2\alpha}\left(\|x_t - x^*\|_2^2 - \|x_{t+1} - x^*\|_2^2\right) + \frac{\alpha L^2}{2}$$

and taking the telescopic sum we have

$$\frac{1}{T}\sum_{t=1}^{T} f(x_t) - f(x^*) \leq \frac{1}{2\alpha T}\left(\|x_1 - x^*\|_2^2 - \|x_{t+1} - x^*\|_2^2\right) + \frac{\alpha L^2}{2}.$$
$$\leq \frac{R^2}{2\alpha T} + \frac{\alpha L^2}{2} = \epsilon \text{ by choosing } \alpha, T \text{ appropriately .}$$

The claim follows by convexity since $\frac{1}{T}\sum_{t=1}^{T} f(x_t) \geq f\left(\frac{1}{T}\sum_{t=1}^{T} x_t\right)$ (from Jensen's inequality).

3 Stochastic Gradient Descent (SGD)

Definition (SGD): Let $f : \mathbb{R}^d \to \mathbb{R}$ be convex (want to minimize). The algorithm below is called stochastic gradient descent

$$x_{k+1} = x_k - \alpha_k v_k$$

where $\mathbb{E}[v_k \mid x_k] \in \partial f(x_k)$.

Remarks

- α_k is called the stepsize. Intuitively the smaller values lead to a slower algorithm.
- α_k must depend on k (must be vanishing to talk about convergence).
- v_k and moreover x_k are random vectors!

4 Analysis of SGD for μ-convex

Theorem 4.1 *(Stochastic Gradient Descent). Let $f : \mathbb{R}^d \to \mathbb{R}$ be μ -strongly convex (want to minimize). Moreover, assume that $\mathbb{E}\left[\|v_k\|^2\right] \leq \rho^2$. Let x^* be a minimizer. It holds for $\alpha_k = \frac{1}{\mu k}$,*

$$\mathbb{E}\left[f\left(\frac{1}{T}\sum_t x_t\right)\right] - f(x^*) \leq \frac{\rho^2}{2\mu T}(1 + \log T)$$

Remarks

- α_k scales with $\frac{1}{k}$ and is vanishing for convergence.
- For $T = \Theta\left(\frac{1}{\epsilon}\log\frac{1}{\epsilon}\right)$ we get error ϵ.
- [1] derives a convergence rate in which the $\log T$ is eliminated for a variant.
- [2] shows theorem above for the last iterate x_T!

Proof: Set $\nabla^t = \mathbb{E}[v_t \mid x_t]$. Then, from strong convexity we get

$$\mathbb{E}\left[(x_t - x^*)^\top \nabla^t\right] \geq \mathbb{E}\left[f(x_t) - f(x^*) + \frac{\mu}{2}\|x_t - x^*\|_2^2\right]. \quad (3) \quad \text{(This should be true no matter what)}$$

Claim: $\mathbb{E}\left[(x_t - x^*)^\top \nabla^t\right] \leq \frac{\mathbb{E}\left[\|x_t - x^*\|_2^2 - \|x_{t+1} - x^*\|_2^2\right]}{2\alpha_t} + \frac{\alpha_t}{2}\rho^2 \quad (4)$

Proof of Claim: Law of Cosines gives

$$\begin{aligned}
\|x_t - x^*\|_2^2 - \|x_{t+1} - x^*\|_2^2 &\geq 2\alpha_t (x_t - x^*)^\top v_t - \alpha_t^2 \|v_t\|_2^2, \\
\mathbb{E}\left[\|x_t - x^*\|_2^2 - \|x_{t+1} - x^*\|_2^2\right] &\geq \mathbb{E}\left[2\alpha_t (x_t - x^*)^\top v_t - \alpha_t^2 \|v_t\|_2^2\right], \\
&= \mathbb{E}\left[\mathbb{E}\left[2\alpha_t (x_t - x^*)^\top v_t - \alpha_t^2 \|v_t\|_2^2 | x_t\right]\right] \text{ (from Tower property)}, \\
&= 2\alpha_t \mathbb{E}\left[(x_t - x^*)^\top \mathbb{E}\left[v_t | x_t\right]\right] - \alpha_t^2 \rho^2, \\
&= 2\alpha_t \mathbb{E}\left[(x_t - x^*)^\top \nabla^t\right] - \alpha_t^2 \rho^2.
\end{aligned}$$

Combining (3) & (4) and utilizing the linearity of expectation operator, we can write:

$$\mathbb{E}\left[f(x_t) - f(x^*)\right] \leq \frac{\mathbb{E}\left[\|x_t - x^*\|_2^2 (1 - \alpha_t \mu) - \|x_{t+1} - x^*\|_2^2\right]}{2\alpha_t} + \frac{\alpha_t}{2}\rho^2$$

Therefore, again utilizing the linearity of expectation operator, and recalling $\alpha_t = \frac{1}{t\mu}$,

$$\begin{aligned}
\mathbb{E}\left[\frac{1}{T}\sum_t f(x_t)\right] - f(x^*) \leq& \mathbb{E}\left[-\mu T \|x_{T+1} - x^*\|_2^2\right] + \frac{\rho^2}{2\mu}\frac{1}{T}\sum_t \frac{1}{t}, \\
&\leq \frac{\rho^2}{2\mu}\left(\frac{1 + \log T}{T}\right) (\text{using} \sum_{k=1}^{n} \frac{1}{k} \approx logn < logn + 1)).
\end{aligned}$$

5 Analysis of SGD (general)

Theorem 5.1 *(Stochastic Gradient Descent). Let $f : \mathbb{R}^d \to \mathbb{R}$ be a convex function (want to minimize). Moreover, assume that $\|v_k\|_2 \leq \rho$ with probability 1. Let x^* be a minimizer. It holds for $\alpha = \frac{R}{\rho\sqrt{k}}$,*

$$\mathbb{E}\left[f\left(\frac{1}{T}\sum_t x_t\right)\right] - f(x^*) \leq \frac{R\rho}{\sqrt{T}}$$

Remarks

- α scales with $\sqrt{\frac{1}{k}}$ (k is the total number of iterations) and is vanishing for convergence, but fixed!
- For $T = \Theta\left(\frac{1}{\epsilon^2}\right)$, we get error ϵ.

Proof:

$$\begin{aligned}
\mathbb{E}_{1:T}\left[f\left(x_t\right)-f\left(x^*\right)\right] &\leq \mathbb{E}_{1:T}\left[\left(x_t-x^*\right)^\top \nabla^t\right], \text{(due to convexity)} \\
&= \mathbb{E}_{1:t-1}\left[\mathbb{E}_{1:T}\left[\left(x_t-x^*\right)^\top \nabla^t \mid v_1,\ldots,v_{t-1}\right]\right], \\
&= \mathbb{E}_{1:t-1}\left[\left(x_t-x^*\right)^\top \mathbb{E}_{1:T}\left[\nabla^t \mid v_1,\ldots,v_{t-1}\right]\right], \\
&= \mathbb{E}_{1:t-1}\left[\left(x_t-x^*\right)^\top v_t\right], \quad (\text{ Recall } ||v_t|| \leq \rho) \\
&\leq \mathbb{E}_{1:T}\left[\frac{1}{2\alpha}\left(\|x_t-x^*\|_2^2-\|x_{t+1}-x^*\|_2^2\right)\right]+\frac{\alpha\rho^2}{2}.
\end{aligned}$$

Taking the telescopic sum, we have

$$\mathbb{E}_{1:T}\left[\frac{1}{T}\sum_{t=1}^{T} f\left(x_t\right)-f\left(x^*\right)\right] \leq \frac{R^2}{2\alpha T}+\frac{\alpha\rho^2}{2}.$$

Example: Coordinate Descent (Special case of SGD)
Definition (Coordinate Descent): Let $f:\mathbb{R}^d \to \mathbb{R}$ be convex, differentiable function in some convex set $\mathcal{X}$. Coordinate descent is defined iteratively:

Choose coordinate $i \in [d]$ and update $x_{k+1} = x_k - \alpha_k \frac{\partial f(x_k)}{\partial x_i} \cdot e_i$.

Remarks

- Similar guarantees with gradient descent can be provided as long as each coordinate is taken often.
- If coordinate i is chosen uniformly at random, then instantination of SGD.

6 Conclusion

- An introduction is made to Subgradients and SGD.
  - It has been shown that the same guarantees as for differentiable functions apply with the employment of subgradient concept.
  - SGD has a rate of convergence $\Theta\left(\frac{1}{\epsilon}\ln\frac{1}{\epsilon}\right)$ for μ -convexity.

References

[1] Rakhlin, Shamir & Sridharan, *Relax and randomize: From value to algorithms*, Proceedings of the 25th International Conference on Neural Information Processing Systems-Volume 2, 2012.

[2] Shamir & Zhang , *Stochastic gradient descent for non-smooth optimization: Convergence results and optimal averaging schemes*, International conference on machine learning, 2013.

CS295 Optimization for Machine Learning

Instructor: Ioannis Panageas Scribed by: Emily Gallagher-Brunelli, Alex Konrad

Lecture 4. Stochastic Gradient Descent (Examples)

1 Introduction

Consider examples that one can face in regression problems. Eventually an optimization problem appears and you want to minimize some function–that's when stochastic gradient descent appears.

2 Optimization in ML, SGD to the rescue

Definition Let $l(x, z) : \mathcal{X} \times \mathcal{Z} \to \mathcal{R}$ be a risk function (x, z are vectors) and D some unknown distribution we can get samples from. We are interested in solving:

$$\min_{x \in \mathcal{X}} L(x), \text{where } L(x) := \mathbb{E}_{Z \sim D}[l(x, z)]$$

I.e. we want to find some x that minimizes our risk. Approach one:

1. Take enough (say n) samples z_i *independently* and consider the *estimate* L, the empirical mean of our loss. By **Law of Large Numbers** this is a close enough with high probability.
2. Run a first order optimization algorithm like gradient descent on $\bar{L}(x)$.

Remark: One of the weaknesses of this approach is that we need to have the closed form of our function. If we don't know the form of $l(x, z)$ and we only have oracle access it's impossible (oracle access meaning, if point x is given to you, you can give back the value of the function at point x). Secondly, there are a large number of computations per iteration of gradient descent.

2.1 Stochastic Gradient Descent

1. For each iteration $t+1$, take *a fresh* sample z_t *independently* from $z_1, \ldots, z_t$ and consider the *unbiased estimate* $\nabla_x l(x_t, z_t)$ (the gradient of the risk at the point z_t, which is the random sample, and x_t, the current iteration). This is as opposed to taking samples at the beginning of the process like we did in the case above.
2. Update $x_{t+1} = x_t - \alpha_t \nabla_x l(x_t, z_t)$. When we do an update, we use only the current sample and the current iteration (different from gradient descent where we take all samples in the beginning). Likewise, we DON'T need the closed form for the risk minimization function. Every iteration we just have one basically one computation (no need to sum over all the gradients and take the average).

One upside is we can have a smaller batch size.

In general this is faster but there are some downsides. For one, the variance of the gradient increases. Also, we need to make sure the estimate of the gradient is really unbiased.

Less cost per iteration, but you need to "pay" the variance.

Because the z's are independent, the sample is an unbiased estimate of the gradient. Because of the linearity of expectation.

For $t := 1$ to T, do
1. sample $z \sim D$
2. Pick $v_t \in \partial l(x_t, z)$
3. $x_{t+1} = x_t - \alpha_t v_t$ (remember, we choose the stepsize to be $\frac{1}{t}$ if our function is strongly convex, and if our function is Lipschitz then $\alpha = \frac{1}{\sqrt{t}}$)

Return $\frac{1}{T}\sum x_t$, the average of all our points we visited.

Forget that you have randomness in the objective/loss function, and supposed the our loss function can be written as the sum of other functions:
$L(x) = \frac{1}{n}\sum_{i=1}^{n} g_i(x)$

Instead of just running gradient descent on this function, we can:

1. For each iteration $t + 1$, take uniformly at random *independently index* i from 1 to n and consider the (unbiased estimate) $\nabla_x g_i(x)$.
2. Update $x_{t+1} = x_t - \alpha_t \nabla_x g_i(x)$

Claim: this is an instantiation of stochastic gradient descent. The cost per iteration is sped up by n times. If all the g_i are close to each other in value, then this version of gradient descent should converge fast. But if all the g_i are spread out and have high variance, then it could be even slower than gradient descent.

2.2 An example using Stochastic Gradient Descent

Definition (MLE for Gaussian) Let $z \sim \mathcal{N}(\mu, 1)$ and $l(x, z) := -\log p_x(z)$ denote the log-likelihood of $\mathcal{N}(x, 1)$. We don't know the mean μ and are interested in finding it. One way to do so is to solve the following risk-minimization problem, because maximizing the log likelihood is the same as minimizing the negative log likelihood.

$$\min_{x \in \mathbb{R}} \mathbb{E}_{Z \sim \mathcal{N}(\mu,1)}[-\log p_x(z)]$$

I.e., minimize over x the expectation of the negative log likelihood of our distribution. z represents our normally distributed data points, and $p_x(z)$ represents the PDF of our Gaussian. What is the minimizer of this function?

It is minimized with $x^* = \mu$ (note, this is the true expectation, not the empirical expectation).

Remarks on Maximum (log)-Likelihood:

1. Standard approach for parameter estimation of parametric families of distributions, i.e., create an optimization problem!
2. Under assumptions, the maximum (log) Likelihood Estimator is consistent.
3. Above boils down to $\min_{x\in\mathbb{R}}$ least squares estimate, i.e.

$$\min_{x\in\mathbb{R}} \mathbb{E}_{Z\sim\mathcal{N}(\mu,1)} \frac{(z-x)^2}{2}$$

which is gotten from just taking the log of the Gaussian PDF.

The derivative is just $(x-z)$ and $\mathbb{E}[(x-z)^2] = 1 + (x-\mu)^2$. The second derivative is 1, hence, *1-strongly convex*. Start from $x_0 = 0$. At iteration $t+1$, get a fresh sample z_t, and we have $x_{t+1} = x_t - \alpha_t(x_t - z_t)$.

Choosing $\alpha_t = \frac{1}{t}$ (check SGD theorem), what is x_T? x_T = empirical mean.

$$\alpha_t = \frac{1}{t} \implies x_T = \frac{\sum_{i=1}^{T} z_i}{T}$$

If you choose a different step size it will not be the empirical mean, but it will be a weighted average and will have the same convergence guarantees.

Recall for $T = \Theta(\frac{1}{\epsilon}\log\frac{1}{\epsilon})$.

If $T = \frac{1}{\epsilon}\log\frac{1}{\epsilon}$, then $f(\frac{1}{T}\sum x_i) - f(x^*) \le \epsilon$.

Want to compare the empirical average $\frac{\sum x_i}{T}$ vs μ. Same to compare $-\log p(z)$ vs $-\log p(z)$.

How can I get from the mean to the true parameter. The answer is that the function is strongly convex.

So we have a function $F(y) - F(\mu)$. I claim because it is strongly convex, it should be at least the gradietn of the function at μ times $y - \mu$

$$F(y) - F(\mu) \ge \nabla F(\mu)(y-\mu) + \frac{1}{2}(y-\mu)^2$$

We know that $\nabla F(\mu)$ is 0 so the whole term is 0. This tells us that the second term is less than ϵ. That tells us $y - \mu \in \mathbb{O}(\sqrt{\epsilon})$. So

$$\|\|\frac{\sum x_i}{T} - \mu\|\| \le \mathbb{O}(\sqrt{\epsilon})$$

You can get ϵ-close to the true mean μ after $\frac{1}{\epsilon^2}\ln\frac{1}{\epsilon^2}$ steps. This bound is not tight, but we can make it better.

How many samples do you need to approximate mu with error epsilon?

Chernoff bounds (informal)

$\Pr(|\bar{z} - \mu| > \frac{c}{\sqrt{T}}) \le \epsilon$

With high probability, $\frac{\sum z_i}{T} \in [\mu - \frac{3}{\sqrt{T}}, \mu + \frac{3}{\sqrt{T}}]$.

So you can get rid of $\ln\frac{1}{\epsilon^2}$.

2.3 An example – Bias of a coin

Assume you are given a coin that gives H with probability $p \in (0,1)$ and T with probability $1-p$. You want to tell whether or not the coin is biased (does $p = \frac{1}{2}$?). How many tosses do you need to get an estimate $\tilde{p}$ about p and be sure with probability 99% that $|p - \tilde{p}| \leq \epsilon$? How can you create a test to determine if p and q are close/far from each other? (This is called goodness of fit testing. You might want to check if p is unimodal, for example. Overall, you want to check if some property is true.)

Hint: Density of binomial distribution is $f_p(z) = p^z(1-p)^{1-z}$

A discrete probabilist will use Chernoff bounds or Chebyshev. A statistician/optimization person will solve $\min_x \mathbb{E}[-\log f_x(z)$, i.e. minimizing the negative log likelihood.

We would like to solve (of course $x^* = p$ is the solution but we don't know p). We can do this using SGD.

$$min_x \mathbb{E}[-z \log x - (1-z)\log(1-x)]$$

Note: the above function is note defined when x = 0 or x = 1, so we assume that the coins toss probabilities are in $(0,1)$, not $[0,1]$.

The derivative of l is just $-\frac{z}{x} + \frac{(1-z)}{1-x} = \frac{x-z}{x(1-x)}$, which is in absolute value at most $\frac{1}{\epsilon}$ for $x \in (\epsilon, 1-\epsilon)$.

The second derivative of L is $\frac{p}{x^2} + \frac{1-p}{(1-x^2)}$, hence $4(p-p^2)$-strongly convex in (0, 1). Notice that $p - p^2$, or $p(1-p)$, is the variance of the Bernoulli distribution, so it's very natural that this appears here in the 2nd derivative (I think because of its connection to the 2^{nd} moment?).

Start from $x_0 = \frac{1}{2}$
At iteration $t+1$, get a fresh sample z_t and we have $x_{t+1} = x_t - \alpha_t \frac{(x_t - z_t)}{x_t(1-x_t)}$.
You can get ϵ-close to p after $\frac{1}{4(p-p^2)\epsilon^6} ln \frac{1}{e^2}$ iterations.

How many samples do you need? You need enough samples that you beat the standard deviation. Standard deviation in this case is $\frac{p-p^2}{\sqrt{T}}$ (where T is the number of samples). Consider p to be a constant. Empirical: count the number of heads and divide by the total number of tosses.

Result: if you deviate from a Gaussian (if you do SGD for a Binomial or Bernoulli distribution for example), SGD gives you worse convergence guarantees. In more challenging examples, the empirical mean will not work anymore and you really do need to use SGD. .

The theorem from previous lecture can be used for all these examples.

2.4 Example

Problem (Mixture of Gaussians). Assume you have access to i.i.d. samples from a Gaussian with unknown mean, $z \sim \mathcal{N}(\mu, 1)$. However, there is an adversary that with probability 1/2 corrupts z and gives you $-z$. Can you infer or estimate μ?

This problem is similar to mixture of Gaussian with symmetry and around the mean and 1/2 probability of sampling from either.

Suppose you take the sample average, then in the limit it will converge to 0 because of the symmetry (but this is clearly not a normal distribution centered around 0). This is one example of a problem where the empirical mean cannot give you the right answer. 0 will not be a minimizer.

We need to minimize the negative log likelihood of the mixture of two Gaussians:

$$\min_{x \in \mathbb{R}} \mathbb{E}_{z \sim \mathcal{N}(\mu,1)} \left[-\log(\frac{1}{2\sqrt{2\pi}} e^{(z-x)^2/2}) + \log(\frac{1}{2\sqrt{2\pi}} e^{(z+x)^2/2}) \right]$$

Is the function convex (this is problem 5 on the HW)? There are two minimizers, μ and $-\mu$.

No, it's not convex (intuitively, the function has two peaks, corresponding to the two minimizers), but all the minimizers have the same value. And using some recent machinery we can get some guarantees. Specifically that SGD converges to either μ or $-\mu$.

CS295 Optimization for Machine Learning

Instructor: Ioannis Panageas Scribed by: Tamanna Hossain, Ruoyu Lin

Lecture 5. Online Optimization and Online Learning.

1 Playing the experts game

We start our exploration of online optimization and online learning through the example of *playing the experts games.*

Example 1.1 *(**Playing the Experts Game**) For each day $t = 1 \dots T$, you have to choose between alternatives A,B (e.g. rain or not rain)*

- *Choose A or B according to some rule (eg. you choose to predict it will rain, or not rain).*
- *One of the alternatives realizes (eg. it actually rains, or it is actually does not rain).*
- *If you choose **correctly** you are **not penalized** otherwise you **lose one point**. (eg. if it rains and you carry an umbrella, then you are not penalized; but it rains and you did not carry an umbrella then you are penalized etc.)*
- *Imagine that there are n **experts** (eg. weather forecasters) who on each day t, recommend either A or B.*

Goal: The goal is not to minimize the number of mistakes made, but to perform as close to the best expert as possible, which means, for example, we do the experiments for T days, after which among the n experts, there is a best expert who made the least amount of mistakes. So our goal is to design a rule that performs as close as possible to the best expert did.

Algorithm: To learn the best expert we must run the whole experiment for T days and see who the expert is at the end. We can use the **Weighted Majority** algorithm for this purpose (see Algorithm [1] below). Everyday we get the prediction of each expert, and our prediction is the prediction of the majority of experts, i.e., if more than 50% predict rain then we will get an umbrella, otherwise if more than 50% of experts predict it will not rain then we will not get an umbrella. The 'weighted' aspect of the algorithm is that the credibility or weight given to each expert is lowered when they make mistakes. A *stepsize* of ϵ is used to decrease the weight of experts that make mistakes.

Note that the *majority vote* through this algorithm is not the classical majority in terms of count of experts but the majority weighted prediction. For the first iteration every expert has the same credibility so the weighted majority vote corresponds to the classical majority count. However, for subsequent iterations the weights are adjusted based on credibility so the majority vote can be different from the classic majority count. It is possible for a minority of experts in terms of counts to get a majority in terms of the weighted vote if their credibility is much higher than the

experts voting opposite to them.

Algorithm 1: Weighted Majority

Initialize $w_i^0 = 1$ for all $i \in [n]$;
for *t=1... T* **do**
  if $\sum_{i \text{ choose } A} w_i^{t-1} \geq \sum_{i \text{ choose } B} w_i^{t-1}$ **then**
    Choose A, otherwise B.;
  end
  for *expert i that made a mistake* **do**
    $w_i^t = (1-\epsilon) w_i^{t-1}$
  end
  for *expert i that did not made a mistake* **do**
    $w_i^t = w_i^{t-1}$
  end
end

Theorem 1.2 *(**Weighted Majority**). Let M_T, M_T^B be the total number of mistakes the algorithm and best expert make until step T, respectively. It holds that*

$$M_T \leq 2(1+\epsilon) M_t^B + \frac{\log n}{\epsilon}$$

Proof: Let's define the **potential** function $\phi_t = \sum_i w_i^t$. Note that potential functions are like energy fictions that capture the performance of an algorithm.

- $\phi_0 = n$
- $\phi_{t+1} \leq \phi_t$. Intuitively, see that this is true because the potential function is the sum of weights, and weights can either remain the same or decrease. We have equality only if all experts were correct up to iteration $t+1$.

Observe that if we make a mistake at time t then the weighted majority was wrong, that is at least $\frac{\phi_t}{2}$ will be multiplied by $1-\epsilon$. Hence, if we make a mistake at time t then

$$\phi_{t+1} \leq (1-\epsilon)\frac{\phi_t}{2} + \frac{\phi_t}{2} = (1 - \frac{\epsilon}{2})\phi_t$$

.

That is $\phi_{t+1} \leq (1-\frac{\epsilon}{2})\phi_t$ when we do make a mistake at time t, otherwise just $\phi_{t+1} \leq \phi_t$.

After every mistake, the energy function is shrinked by a factor of $1-\frac{\epsilon}{2}$. So since we have M_T mistakes at the end after T iterations, the energy function at that time has to have been decreased by at least $(1-\frac{\epsilon}{2})^{M_T}$, that is,

$$\phi_T \leq \left(1 - \frac{\epsilon}{2}\right)^{M_T} \phi_0$$

We have shown an upper bound for ϕ_T. We also want to bound ϕ_T from below, which we will do as follows.

Let the best expert be i^*. Since weights are non-negative we know that,

$$\phi_T = \sum_i w_i^T > w_{i^*}^T$$

Now assume the best expert, i^*, makes M_T^B mistakes. Since all weights are 1 in the beginning, we know $w_{i*}^0 = 1$. So $w_{i^*}^T = (1-\epsilon)^{M_T^B}$. Then,

$$\phi_T > w_{i^*}^T = (1-\epsilon)^{M_T^B}$$

Then using the upper and lower bounds of the potential functions,

$$(1-\epsilon)^{M_T^B} < \phi_T \leq \left(1 - \frac{\epsilon}{2}\right)^{M_T} \phi_0$$

We can then conclude that,

$$(1-\epsilon)^{M_T^B} < \left(1 - \frac{\epsilon}{2}\right)^{M_T} n$$

By taking the log,

$$M_T^B \log(1-\epsilon) < \log(1 - \frac{\epsilon}{2}) M_T + \log n$$

Since $x - x^2 < \log(1-x) < -x$,

$$M_T^B(\epsilon - \epsilon^2) < -M_T \frac{\epsilon}{2} + \log n$$

By rearranging,

$$M_T < 2(1+\epsilon)M_t^B + \frac{\log n}{\epsilon}$$

■

2 Playing the experts game (randomized)

Now let's consider the example of *playing the experts games (randomized).*

Definition 2.1 *For each day $t = 1 \ldots T$, you have to choose between alternatives A,B (e.g. rain or not rain)*

- *Choose A or B with some probability (e.g. In the beginning of each day, you have a prediction that it will rain with the probability of 70%).*
- *One of the alternatives realizes*
- *If you choose **correctly** you are **not penalized** otherwise you **lose one point**.*

- *Imagine that there are n* ***experts*** *who on each day t, recommend either A or B.*

Goal: The goal is to design a rule that performs in expectation as close to the best expert as possible.

Algorithm: We define the following algorithm [2] (The algorithm is also called **Multiplicative Weights Update** and performs almost as good as the "**best**" expert (fewest mistakes), where ϵ is the **stepsize** to be chosen later.)

Note that if the expert i made a mistake at time t, we need to update the weights of all the experts instead of just the expert i's weight.

Algorithm 2: Randomized Weighted Majority

Initialize $w_i^0 = 1$ for all $i \in [n]$;
for *t=1... T* **do**
  Randomly Choose the expert i's recommendation with probability $p_i^t = \frac{w_i^{t-1}}{\sum_{j=1}^{n} w_j^{t-1}}$
  if *the expert i made a mistake* **then**
    $w_j^t = (1-\epsilon)w_j^{t-1}, \forall j = 1, \ldots, n$
  end
  if *the expert i did not made a mistake* **then**
    $w_j^t = w_j^{t-1}, \forall j = 1, \ldots, n$
  end
end

Theorem 2.1 *(**Randomized Weighted Majority**). Let M_T, M_T^B be the total number of mistakes the algorithm and best expert make until step T, respectively. It holds that*

$$\mathbf{E}[M_T] \leq (1+\epsilon)M_t^B + \frac{\log n}{\epsilon}$$

Proof: Let's define the **potential** function $\phi_t = \sum_i w_i^t$.
Using the exact same argument, if the best expert (say i^*) did M_T^B mistakes, we have

$$\phi_T = \sum_i w_i^T > w_{i^*}^T = (1-\epsilon)^{M_T^B} * w_{i^*}^0$$

Since $w_{i^*}^0 = 1$, then we have

$$\phi_T > (1-\epsilon)^{M_T^B}$$

Thus the lower bound of ϕ_T is exactly the same as the deterministic case. Now let's consider the upper bound of it. Fitst, let's define the indicator function $\mathbf{1}_i$

$$\mathbf{1}_i = \begin{cases} 1, & \text{if i wrong at t} \\ 0, & \text{if i correct at t} \end{cases}$$

then we have

$$\begin{aligned}\phi_{t+1} &= \sum w_i^{t+1} \\ &= \sum w_i^t(1-\epsilon \mathbf{1}_{\text{i wrong at t}})\end{aligned}$$

Since

$$p_i^{t+1} = \frac{w_i^t}{\sum_{j=1}^{n} w_j^t} = \frac{w_i^t}{\phi_t}$$

then we have

$$w_i^t = \phi_t p_i^{t+1}$$

thus

$$\begin{aligned}\phi_{t+1} &= \sum \phi_t p_i^{t+1}(1-\epsilon \mathbf{1}_{\text{i wrong at t}}) \\ &= \phi_t \sum p_i^{t+1}(1-\epsilon \mathbf{1}_{\text{i wrong at t}}) \\ &= \phi_t(1-\epsilon \sum p_i^{t+1}\mathbf{1}_{\text{i wrong at t}})\end{aligned}$$

Since the probability $P_r[x \in A] = \mathbf{E}[\mathbf{1}_{x\in A}]$, and according to the inequality $1-x \leq e^{-x}$, we have

$$\begin{aligned}\phi_{t+1} &= \phi_t(1-\epsilon \mathbf{E}[\mathbf{1}_{\text{i wrong at t}}]) \\ &\leq \phi_t e^{-\epsilon \mathbf{E}[\mathbf{1}_{\text{i wrong at t}}]}\end{aligned}$$

Using the telescopic product

$$\begin{aligned}\phi_T &= \phi_{T-1} \cdot e^{-\epsilon \mathbf{E}[\mathbf{1}_{\text{i wrong at t}}]} \\ \phi_{T-1} &= \phi_{T-2} \cdot e^{-\epsilon \mathbf{E}[\mathbf{1}_{\text{i wrong at t}}]} \\ &\vdots \\ \phi_2 &= \phi_1 e^{-\epsilon \mathbf{E}[\mathbf{1}_{\text{i wrong at t}}]}\end{aligned}$$

we have

$$\phi_T \leq \phi_1 e^{-\epsilon \mathbf{E}[\sum \mathbf{1}_{\text{i wrong at t}}]} = \phi_1 e^{-\epsilon \mathbf{E}[M_T]}$$

Therefore

$$(1-\epsilon)^{M_T^B} \leq e^{-\epsilon \mathbf{E}[M_T]} n$$

By taking the log,

$$M_T^B(-\epsilon - \epsilon^2) \leq \log n - \epsilon \mathbf{E}[M_T]$$

By rearranging,

$$\mathbf{E}[M_T] \leq (1+\epsilon) M_t^B + \frac{\log n}{\epsilon}$$

■

3 The general online learning setting

Now let's consider the general setting (unlike the precious two cases, the actions here can also be continuous).

Definition 3.1 *At each time step $t = 1 \dots T$.*

- ***Player*** *chooses $x_t \in \mathcal{K} \subset \mathbb{R}^n$ (some closed convex set).*
- ***Adversary*** *(e.g. the weather in precious cases) chooses the lost function $l_t \in \mathcal{F}$ (set of convex functions).*
- ***Player*** *suffers loss $l_t(x_t)$ and observes feedback.*

Goal: The goal is to minimize the (time average) **Regret**, which is

$$\frac{1}{T}[\sum_{t=1}^{T} l_t(x_t) - \min_{u \in \mathcal{K}} \sum_{t=1}^{T} l_t(u)]$$

Note that: if the Regret tends to zero as T tends to infinity, the algorithm is called **no-regret**. And the term **Regret** can be viewed as a generalized notion of the best expert.

Now let's look at a special case - Convex Optimization:

Definition 3.2 *At each time step $t = 1 \dots T$.*

- ***Player*** *chooses $x_t \in \mathcal{K} \subset \mathbb{R}^n$ (some closed convex set).*
- ***Adversary*** *chooses* ***same*** *l (convex function).*
- ***Player*** *suffers loss $l_t(x_t)$ and observes feedback.*

Goal: The goal is to minimize the (time average) **Regret**, which is (According to the Jensen's inequality):

$$\frac{1}{T}[\sum_{t=1}^{T} l(x_t) - \min_{u \in \mathcal{K}} \sum_{t=1}^{T} l(u)] \geq l(\frac{1}{T} \sum_{t=1}^{T} x_t) - l(x^*)$$

Now let's look at the Regret for the Experts problem, which is also a special case of the general setting:

Goal: The goal is to minimize the (time average) **Regret**, which is just:

$$\frac{\mathbf{E}[M_T] - M_T^B}{T}$$

Explanation: We choose x_t as the probability distribution at time t over experts and l_t is the probability to do a mistake.

Recall that

$$\mathbf{E}[M_T] \leq (1+\epsilon)M_t^B + \frac{\log n}{\epsilon}$$

Choosing $\epsilon = (\frac{\log n}{T})^{\frac{1}{2}}$ gives average regret $2(\frac{\log n}{T})^{\frac{1}{2}}$, which is of $\mathcal{O}(T^{-\frac{1}{2}})$ since $(\mathbf{E}[M_T] - M_T^B)$ is of $\mathcal{O}(T^{\frac{1}{2}})$. Note that we can NOT do better.

Consider just **two** experts that choose one A and B respectively at all times. The adversary chooses **uniformaly ar random** A or B. The **expected number of mistakes** of an online algorithm is $\frac{T}{2}$. One of the two fixed statrgies will have $\frac{T}{2} - \Theta(\sqrt{T})$ mistakes **with high probability**.

CS295 Optimization for Machine Learning

Instructor: Ioannis Panageas Scribed by: Pouya M Ghari

Lecture 7. Faster than GD: Accelerated Methods.

1 Introduction

This lecture studies solving the optimization problem of minimizing $f(\boldsymbol{x})$ where $\boldsymbol{x} \in \mathbb{R}^d$. In this context, gradient descent algorithm exhibits well-documented performance in practice as well as great theoretical properties. Let $\boldsymbol{x}^*$ be the optimal solution to minimize $f(\boldsymbol{x})$ and $\boldsymbol{x}_t$ be the obtained solution by gradient descent at the t-th iteration. It is proved that if the function $f(\cdot)$ is a differentiable, convex and L-smooth function, $f(\boldsymbol{x}_{T+1}) - f(\boldsymbol{x}^*) \leq \epsilon$ when the step size is appropriately chosen as $\alpha = \frac{1}{L}$ and we have

$$T = \frac{2\|\boldsymbol{x}_1 - \boldsymbol{x}^*\|_2^2 L}{\epsilon}. \tag{1}$$

This shows that the speed of convergence is independent of the dimension d while gradient descent achieves convergence rate of $\mathcal{O}(\frac{L}{\epsilon})$. Furthermore, it is proved that if the function $f(\cdot)$ is a differentiable μ-strongly convex function and L-smooth, $\|\boldsymbol{x}_T - \boldsymbol{x}^*\| \leq \epsilon$ holds for

$$T = \frac{2L}{\mu} \ln\left(\frac{\|\boldsymbol{x}_1 - \boldsymbol{x}^*\|_2}{\epsilon}\right) \tag{2}$$

when the step size is appropriately chosen as $\alpha = \frac{1}{L}$. In this case, it can be concluded that the speed of convergence is independent of dimension d while the rate of convergence is $\mathcal{O}(\frac{L}{\mu}\log(\frac{1}{\epsilon}))$.

In order to improve the convergence properties of gradient descent based algorithms, accelerated gradient descent has been proposed in the literature. This lecture studies the accelerated gradient descent algorithm and its theoretical properties. Section 2 presents the accelerated gradient descent algorithm. In section 3 the convergence properties of accelerated gradient descent are analyzed. Specifically, it is proved that when the function $f(\cdot)$ is a twice differentiable μ-strongly convex function and L-smooth, the convergence rate of accelerated gradient descent is $\mathcal{O}(\sqrt{\frac{L}{\mu}}\log(\frac{1}{\epsilon}))$ which exhibits significant improvement relative to that of conventional gradient descent algorithm. Furthermore, it is proved that when the function $f(\cdot)$ is a twice differentiable L-smooth function, accelerated gradient descent achieves convergence rate of $\mathcal{O}(\sqrt{\frac{L}{\epsilon}})$ which shows that accelerated gradient descent results in significant improvement in convergence rate compared to the conventional gradient descent.

2 Accelerated Gradient Descent

This section presents accelerated gradient descent algorithm proposed by Nesterov. Let $f : \mathbb{R}^d \to \mathbb{R}$ be a differentiable function. The goal is to find $\boldsymbol{x}^* \in \mathbb{R}^d$ which minimizes $f(\cdot)$. Let $\boldsymbol{x}_t \in \mathbb{R}^d$ denote

Algorithm 1 Accelerated Gradient Descent

Initialize: $\boldsymbol{x}_1$, $\boldsymbol{y}_1 = \boldsymbol{x}_1$, stepsize η.
for $t = 1, \ldots, T$ **do**
  $\boldsymbol{y}_{t+1} = \boldsymbol{x}_t - \eta \nabla f(\boldsymbol{x}_t)$.
  $\boldsymbol{x}_{t+1} = \boldsymbol{y}_{t+1} + \gamma_t(\boldsymbol{y}_{t+1} - \boldsymbol{y}_t)$.
end for

the obtained solution by accelerated gradient descent algorithm at the t-th iteration. Define $\boldsymbol{y}_{t+1}$ as

$$\boldsymbol{y}_{t+1} = \boldsymbol{x}_t - \eta \nabla f(\boldsymbol{x}_t) \tag{3}$$

where η is the stepsize. Then the solution $\boldsymbol{x}_{t+1}$ is updated as

$$\boldsymbol{x}_{t+1} = (1 + \gamma_t)\boldsymbol{y}_{t+1} - \gamma_t \boldsymbol{y}_t = \boldsymbol{y}_{t+1} + \gamma_t(\boldsymbol{y}_{t+1} - \boldsymbol{y}_t) \tag{4}$$

where γ_t is a sequence independent of $\boldsymbol{x}_t$ such that $\gamma_t \geq 0$ for all $1 \leq t \leq T$. Introduced by Nesterov, $\boldsymbol{y}_{t+1} - \boldsymbol{y}_t$ is called momentum. Algorithm 1 summarizes accelerated gradient descent algorithm. In what follows, we study the theoretical properties of this algorithm.

3 Analysis

This section analyzes the convergence of accelerated gradient descent algorithm presented in Algorithm 1. The ensuing theorem studies the convergence of Algorithm 1 when $f(\cdot)$ is strongly convex and smooth.

Theorem 3.1 *Let $f(\cdot) : \mathbb{R}^d \to \mathbb{R}$ be a twice differentiable, L-smooth and μ-strongly convex function. Assume that $\boldsymbol{x}^*$ is the minimizer and set $\gamma_t = \frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}$ and $\eta = \frac{1}{L}$ where $\kappa = \frac{L}{\mu}$ is called condition number. Then it holds that*

$$f(\boldsymbol{y}_{t+1}) - f(\boldsymbol{x}^*) \leq \frac{L+\mu}{2}\|\boldsymbol{x}_1 - \boldsymbol{x}^*\|_2^2 e^{-\frac{t}{\sqrt{\kappa}}} \tag{5}$$

hence we reach ϵ-close in ℓ_2 after $T := \sqrt{\frac{L}{\mu}}\log(\frac{\|\boldsymbol{x}_1 - \boldsymbol{x}^*\|_2^2(L+\mu)}{\epsilon})$ iterations.

Proof: We define the following sequence of functions:

$$\Phi_1(\boldsymbol{x}) = f(\boldsymbol{x}_1) + \frac{\mu}{2}\|\boldsymbol{x} - \boldsymbol{x}_1\|_2^2 \tag{6a}$$

$$\Phi_{s+1}(\boldsymbol{x}) = \left(1 - \frac{1}{\sqrt{\kappa}}\right)\Phi_s(\boldsymbol{x}) + \frac{1}{\sqrt{\kappa}}\left(f(\boldsymbol{x}_s) + \nabla f(\boldsymbol{x}_s)^\top(\boldsymbol{x} - \boldsymbol{x}_s) + \frac{\mu}{2}\|\boldsymbol{x} - \boldsymbol{x}_s\|_2^2\right) \tag{6b}$$

Claim 3.2 *The function $\Phi_{s+1}(\boldsymbol{x})$ is bounded from above as:*

$$\Phi_{s+1}(\boldsymbol{x}) \leq f(\boldsymbol{x}) + \left(1 - \frac{1}{\sqrt{\kappa}}\right)^s (\Phi_1(\boldsymbol{x}) - f(\boldsymbol{x})) \tag{7}$$

Proof: For $\Phi_{t+1}(\boldsymbol{x})$ we find

$$\Phi_{t+1}(\boldsymbol{x}) = \left(1 - \frac{1}{\sqrt{\kappa}}\right) \Phi_t(\boldsymbol{x}) + \frac{1}{\sqrt{\kappa}} \left(f(\boldsymbol{x}_t) + \nabla f(\boldsymbol{x}_t)^\top (\boldsymbol{x} - \boldsymbol{x}_t) + \frac{\mu}{2} \|\boldsymbol{x} - \boldsymbol{x}_t\|_2^2\right). \tag{8}$$

According to the fact that $f(\cdot)$ is a strongly convex function, from (8) we obtain

$$\Phi_{t+1}(\boldsymbol{x}) \leq \left(1 - \frac{1}{\sqrt{\kappa}}\right) \Phi_t(\boldsymbol{x}) + \frac{1}{\sqrt{\kappa}} f(\boldsymbol{x}) = f(\boldsymbol{x}) + \left(1 - \frac{1}{\sqrt{\kappa}}\right) \left(\Phi_t(\boldsymbol{x}) - f(\boldsymbol{x})\right). \tag{9}$$

Therefore, we conclude that

$$\Phi_{t+1}(\boldsymbol{x}) - f(\boldsymbol{x}) \leq \left(1 - \frac{1}{\sqrt{\kappa}}\right) \left(\Phi_t(\boldsymbol{x}) - f(\boldsymbol{x})\right). \tag{10}$$

Applying the inequality in (10) recursively, we get

$$\Phi_{t+1}(\boldsymbol{x}) - f(\boldsymbol{x}) \leq \left(1 - \frac{1}{\sqrt{\kappa}}\right)^t \left(\Phi_1(\boldsymbol{x}) - f(\boldsymbol{x})\right) \tag{11}$$

which proves the claim 3.2. ■

In the following claim, we find the upper bound for $f(\boldsymbol{y}_s)$ in terms of $\Phi_s(\cdot)$.

Claim 3.3 *The value of $f(\boldsymbol{y}_s)$ is bounded from above as*

$$f(\boldsymbol{y}_s) \leq \min_{\boldsymbol{x}} \Phi_s(\boldsymbol{x}) \tag{12}$$

Proof: In order to prove this claim we use induction. For $\Phi_1(\boldsymbol{x})$, we find

$$\Phi_1(\boldsymbol{x}) = f(\boldsymbol{x}_1) + \frac{\mu}{2} \|\boldsymbol{x} - \boldsymbol{x}_1\|_2^2 \geq f(\boldsymbol{x}_1) \tag{13}$$

which holds for all $\boldsymbol{x} \in \mathbb{R}^d$. Therefore, it can be concluded that $\min_{\boldsymbol{x}} \Phi_1(\boldsymbol{x}) \geq f(\boldsymbol{x}_1)$. Since $\boldsymbol{y}_1$ is initialized as $\boldsymbol{y}_1 = \boldsymbol{x}_1$, we conclude that

$$f(\boldsymbol{y}_1) \leq \min_{\boldsymbol{x}} \Phi_1(\boldsymbol{x}). \tag{14}$$

Set $\min_{\boldsymbol{x}} \Phi_s(\boldsymbol{x}) = \Phi_s^*$. Moreover, according to the descent lemma, we can write

$$f(\boldsymbol{y}_{s+1}) \leq f(\boldsymbol{x}_s) + \nabla f(\boldsymbol{x}_s)^\top (\boldsymbol{y}_{s+1} - \boldsymbol{x}_s) + \frac{L}{2} \|\boldsymbol{y}_{s+1} - \boldsymbol{x}_s\|_2^2. \tag{15}$$

Based on (15) and the facts that $\boldsymbol{y}_{s+1} = \boldsymbol{x}_s - \eta \nabla f(\boldsymbol{x}_s)$ and $\eta = \frac{1}{L}$, we can conclude that

$$f(\boldsymbol{y}_{s+1}) \leq f(\boldsymbol{x}_s) - \frac{1}{2L} \|\nabla f(\boldsymbol{x}_s)\|_2^2 \tag{16}$$

which can be rewritten as

$$f(\boldsymbol{y}_{s+1}) \leq \left(1 - \frac{1}{\sqrt{\kappa}}\right) f(\boldsymbol{y}_s) + \left(1 - \frac{1}{\sqrt{\kappa}}\right) \left(f(\boldsymbol{x}_s) - f(\boldsymbol{y}_s)\right) + \frac{1}{\sqrt{\kappa}} f(\boldsymbol{x}_s) - \frac{1}{2L} \|\nabla f(\boldsymbol{x}_s)\|_2^2. \tag{17}$$

To prove the claim using the induction, assume that we have $f(\boldsymbol{y}_s) \leq \Phi_s^*$. Therefore, according to (17) we can write

$$f(\boldsymbol{y}_{s+1}) \leq \left(1 - \frac{1}{\sqrt{\kappa}}\right) \Phi_s^* + \left(1 - \frac{1}{\sqrt{\kappa}}\right) (f(\boldsymbol{x}_s) - f(\boldsymbol{y}_s)) + \frac{1}{\sqrt{\kappa}} f(\boldsymbol{x}_s) - \frac{1}{2L} \|\nabla f(\boldsymbol{x}_s)\|_2^2. \tag{18}$$

Applying the first order convexity condition associated with $f(\boldsymbol{x}_s) - f(\boldsymbol{y}_s)$ leads to

$$f(\boldsymbol{y}_{s+1}) \leq \left(1 - \frac{1}{\sqrt{\kappa}}\right) \Phi_s^* + \left(1 - \frac{1}{\sqrt{\kappa}}\right) \nabla f(\boldsymbol{x}_s)^\top (\boldsymbol{x}_s - \boldsymbol{y}_s) + \frac{1}{\sqrt{\kappa}} f(\boldsymbol{x}_s) - \frac{1}{2L} \|\nabla f(\boldsymbol{x}_s)\|_2^2. \tag{19}$$

From (6a) it can be observed that $\nabla^2 \Phi_1(\boldsymbol{x}) = \mu \boldsymbol{I}_d$ where $\boldsymbol{I}_d$ denote d-by-d identity matrix. Taking the second derivative with respect to $\boldsymbol{x}$ from (6b) we get

$$\nabla^2 \Phi_{s+1}(\boldsymbol{x}) = \left(1 - \frac{1}{\sqrt{\kappa}}\right) \nabla^2 \Phi_s(\boldsymbol{x}) + \frac{\mu}{\sqrt{\kappa}} \boldsymbol{I}_d. \tag{20}$$

Assume that $\nabla^2 \Phi_s(\boldsymbol{x}) = \mu \boldsymbol{I}_d$. Then using (20) we obtain that $\nabla^2 \Phi_{s+1}(\boldsymbol{x}) = \mu \boldsymbol{I}_d$. Considering the fact that $\nabla^2 \Phi_1(\boldsymbol{x}) = \mu \boldsymbol{I}_d$, based on the mathematical induction we conclude that $\nabla^2 \Phi_s(\boldsymbol{x}) = \mu \boldsymbol{I}_d$ holds. Therefore, we can write $\Phi_s(\boldsymbol{x}) = \Phi_s^* + \frac{\mu}{2} \|\boldsymbol{x} - \boldsymbol{v}_s\|_2^2$ for some $\boldsymbol{v}_s \in \mathbb{R}^d$. Therefore, (6b) can be rewritten as

$$\Phi_{s+1}(\boldsymbol{x}) = \left(1 - \frac{1}{\sqrt{\kappa}}\right) (\Phi_s^* + \frac{\mu}{2} \|\boldsymbol{x} - \boldsymbol{v}_s\|_2^2) + \frac{1}{\sqrt{\kappa}} \left(f(\boldsymbol{x}_s) + \nabla f(\boldsymbol{x}_s)^\top (\boldsymbol{x} - \boldsymbol{x}_s) + \frac{\mu}{2} \|\boldsymbol{x} - \boldsymbol{x}_s\|_2^2\right). \tag{21}$$

Taking derivative from (21) with respect to $\boldsymbol{x}$ we obtain

$$\nabla \Phi_{s+1}(\boldsymbol{x}) = \mu \left(1 - \frac{1}{\sqrt{\kappa}}\right) (\boldsymbol{x} - \boldsymbol{v}_s) + \frac{1}{\sqrt{\kappa}} \nabla f(\boldsymbol{x}_s) + \frac{\mu}{\sqrt{\kappa}} (\boldsymbol{x} - \boldsymbol{x}_s). \tag{22}$$

Note that $\Phi_s(\boldsymbol{v}_s) = \Phi_s^*$ which shows that $\boldsymbol{v}_s$ is a minimizer of $\Phi_s(\cdot)$. Therefore, we conclude that $\boldsymbol{v}_{s+1}$ is a minimizer of $\Phi_{s+1}(\cdot)$ and as a result $\nabla \Phi_{s+1}(\boldsymbol{v}_{s+1}) = 0$. Therefore, substituting $\boldsymbol{v}_{s+1}$ into (22) we get

$$\mu \left(1 - \frac{1}{\sqrt{\kappa}}\right) (\boldsymbol{v}_{s+1} - \boldsymbol{v}_s) + \frac{1}{\sqrt{\kappa}} \nabla f(\boldsymbol{x}_s) + \frac{\mu}{\sqrt{\kappa}} (\boldsymbol{v}_{s+1} - \boldsymbol{x}_s) = 0 \tag{23}$$

by which we can express $\boldsymbol{v}_{s+1}$ as

$$\boldsymbol{v}_{s+1} = \left(1 - \frac{1}{\sqrt{\kappa}}\right) \boldsymbol{v}_s + \frac{1}{\sqrt{\kappa}} \boldsymbol{x}_s - \frac{1}{\mu \sqrt{\kappa}} \nabla f(\boldsymbol{x}_s). \tag{24}$$

Furthermore, evaluating $\Phi_{s+1}(\cdot)$ at $\boldsymbol{x}_s$ we have

$$\Phi_{s+1}^* + \frac{\mu}{2} \|\boldsymbol{x}_s - \boldsymbol{v}_{s+1}\|_2^2 = \left(1 - \frac{1}{\sqrt{\kappa}}\right) \Phi_s^* + \frac{\mu}{2} \left(1 - \frac{1}{\sqrt{\kappa}}\right) \|\boldsymbol{x}_s - \boldsymbol{v}_s\|_2^2 + \frac{1}{\sqrt{\kappa}} f(\boldsymbol{x}_s) \tag{25}$$

According to (24), $\|\boldsymbol{x}_s - \boldsymbol{v}_{s+1}\|_2^2$ can be expressed as

$$\begin{aligned} \|\boldsymbol{x}_s - \boldsymbol{v}_{s+1}\|_2^2 &= \left\| \left(1 - \frac{1}{\sqrt{\kappa}}\right) (\boldsymbol{x}_s - \boldsymbol{v}_s) + \frac{1}{\mu \sqrt{\kappa}} \nabla f(\boldsymbol{x}_s) \right\|_2^2 \\ &= \left(1 - \frac{1}{\sqrt{\kappa}}\right)^2 \|\boldsymbol{x}_s - \boldsymbol{v}_s\|_2^2 + \frac{1}{\mu^2 \kappa} \|\nabla f(\boldsymbol{x}_s)\|_2^2 - \frac{2}{\mu \sqrt{\kappa}} \left(1 - \frac{1}{\sqrt{\kappa}}\right) \nabla f(\boldsymbol{x}_s)^\top (\boldsymbol{v}_s - \boldsymbol{x}_s) \end{aligned} \tag{26}$$

Therefore, (25) can be rewritten as

$$\Phi^*_{s+1} = \left(1 - \frac{1}{\sqrt{\kappa}}\right)\Phi^*_s + \frac{\mu}{\sqrt{\kappa}}\left(1 - \frac{1}{\sqrt{\kappa}}\right)\|\boldsymbol{x}_s - \boldsymbol{v}_s\|_2^2 + \frac{1}{\sqrt{\kappa}}f(\boldsymbol{x}_s) - \frac{1}{2L}\|\nabla f(\boldsymbol{x}_s)\|_2^2 + \frac{1}{\sqrt{\kappa}}\left(1 - \frac{1}{\sqrt{\kappa}}\right)\nabla f(\boldsymbol{x}_s)^\top(\boldsymbol{v}_s - \boldsymbol{x}_s). \tag{27}$$

Moreover, according to (6a) it can be inferred that $\boldsymbol{v}_1 = \boldsymbol{x}_1$ since $\Phi^*_1 = f(\boldsymbol{x}_1)$. Now we aim to prove that $\boldsymbol{v}_s - \boldsymbol{x}_s = \sqrt{\kappa}(\boldsymbol{x}_s - \boldsymbol{y}_s)$ by induction. Note that $\boldsymbol{v}_1 = \boldsymbol{x}_1 = \boldsymbol{y}_1$ which shows that $\boldsymbol{v}_1 - \boldsymbol{x}_1 = \sqrt{\kappa}(\boldsymbol{x}_1 - \boldsymbol{y}_1) = 0$. Now assume that $\boldsymbol{v}_s - \boldsymbol{x}_s = \sqrt{\kappa}(\boldsymbol{x}_s - \boldsymbol{y}_s)$ holds true. Then based on (24) we can write

$$\begin{aligned}\boldsymbol{v}_{s+1} - \boldsymbol{x}_{s+1} &= \left(1 - \frac{1}{\sqrt{\kappa}}\right)\boldsymbol{v}_s + \frac{1}{\sqrt{\kappa}}\boldsymbol{x}_s - \frac{1}{\mu\sqrt{\kappa}}\nabla f(\boldsymbol{x}_s) - \boldsymbol{x}_{s+1} \\ &= \sqrt{\kappa}\boldsymbol{x}_s - (\sqrt{\kappa} - 1)\boldsymbol{y}_s - \frac{\sqrt{\kappa}}{L}\nabla f(\boldsymbol{x}_s) - \boldsymbol{x}_{s+1}.\end{aligned} \tag{28}$$

Taking into account that $\boldsymbol{y}_{s+1} = \boldsymbol{x}_s - \eta\nabla f(\boldsymbol{x}_s) = \boldsymbol{x}_s - \frac{1}{L}\nabla f(\boldsymbol{x}_s)$, (28) can be rewritten as

$$\boldsymbol{v}_{s+1} - \boldsymbol{x}_{s+1} = \sqrt{\kappa}\boldsymbol{y}_{s+1} - (\sqrt{\kappa} - 1)\boldsymbol{y}_s - \boldsymbol{x}_{s+1}. \tag{29}$$

According to the Algorithm 1 and the fact that $\gamma_t = \frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}$, we have

$$\boldsymbol{y}_s = \frac{1 + \gamma_t}{\gamma_t}\boldsymbol{y}_{s+1} - \frac{1}{\gamma_t}\boldsymbol{x}_{s+1} = \frac{2\sqrt{\kappa}}{\sqrt{\kappa} - 1}\boldsymbol{y}_{s+1} - \frac{\sqrt{\kappa} + 1}{\sqrt{\kappa} - 1}\boldsymbol{x}_{s+1}. \tag{30}$$

Therefore, using (30), the equality (29) can be rewritten as

$$\boldsymbol{v}_{s+1} - \boldsymbol{x}_{s+1} = \sqrt{\kappa}(\boldsymbol{x}_{s+1} - \boldsymbol{y}_{s+1}). \tag{31}$$

Therefore, by induction we prove that $\boldsymbol{v}_s - \boldsymbol{x}_s = \sqrt{\kappa}(\boldsymbol{x}_s - \boldsymbol{y}_s)$. Hence, (27) can be rewritten as

$$\Phi^*_{s+1} = \left(1 - \frac{1}{\sqrt{\kappa}}\right)\Phi^*_s + \mu\left(\sqrt{\kappa} - 1\right)\|\boldsymbol{x}_s - \boldsymbol{y}_s\|_2^2 + \frac{1}{\sqrt{\kappa}}f(\boldsymbol{x}_s) - \frac{1}{2L}\|\nabla f(\boldsymbol{x}_s)\|_2^2 + \left(1 - \frac{1}{\sqrt{\kappa}}\right)\nabla f(\boldsymbol{x}_s)^\top(\boldsymbol{x}_s - \boldsymbol{y}_s). \tag{32}$$

Combining (19) with (32), we conclude that

$$f(\boldsymbol{y}_{s+1}) \leq \Phi^*_{s+1} - \mu\left(\sqrt{\kappa} - 1\right)\|\boldsymbol{x}_s - \boldsymbol{y}_s\|_2^2 \tag{33}$$

which proves that $f(\boldsymbol{y}_{s+1}) \leq \min_{\boldsymbol{x}} \Phi_{s+1}(\boldsymbol{x})$. Therefore, using the induction we conclude that the claim 3.3 is proved. ■

Combining claim 3.2 with claim 3.3, we obtain

$$f(\boldsymbol{y}_{t+1}) \leq \Phi_{t+1}(\boldsymbol{x}^*) \leq f(\boldsymbol{x}^*) + \left(1 - \frac{1}{\sqrt{\kappa}}\right)^t (\Phi_1(\boldsymbol{x}^*) - f(\boldsymbol{x}^*)) \tag{34}$$

Using (6a) and (34), we can write

$$f(\boldsymbol{y}_{t+1}) - f(\boldsymbol{x}^*) \leq \left(1 - \frac{1}{\sqrt{\kappa}}\right)^t \left(f(\boldsymbol{x}_1) - f(\boldsymbol{x}^*) + \frac{\mu}{2}\|\boldsymbol{x}_1 - \boldsymbol{x}^*\|_2^2\right). \tag{35}$$

Furthermore, based on the descent lemma we have

$$f(\boldsymbol{x}_1) \leq f(\boldsymbol{x}^*) + \nabla f(\boldsymbol{x}^*)^\top (\boldsymbol{x}_1 - \boldsymbol{x}^*) + \frac{L}{2}\|\boldsymbol{x}_1 - \boldsymbol{x}^*\|_2^2. \tag{36}$$

Since $\nabla f(\boldsymbol{x}^*)^\top (\boldsymbol{x}_1 - \boldsymbol{x}^*) = 0$, we get

$$f(\boldsymbol{x}_1) - f(\boldsymbol{x}^*) \leq \frac{L}{2}\|\boldsymbol{x}_1 - \boldsymbol{x}^*\|_2^2. \tag{37}$$

Combining (35) with (37) we obtain

$$f(\boldsymbol{y}_{t+1}) - f(\boldsymbol{x}^*) \leq \left(1 - \frac{1}{\sqrt{\kappa}}\right)^t \frac{L+\mu}{2}\|\boldsymbol{x}_1 - \boldsymbol{x}^*\|_2^2. \tag{38}$$

Using the inequality $\left(1 - \frac{1}{\sqrt{\kappa}}\right)^t \leq e^{-\frac{t}{\sqrt{\kappa}}}$, the inequality (38) can be relaxed to

$$f(\boldsymbol{y}_{t+1}) - f(\boldsymbol{x}^*) \leq \frac{L+\mu}{2}\|\boldsymbol{x}_1 - \boldsymbol{x}^*\|_2^2 e^{-\frac{t}{\sqrt{\kappa}}} \tag{39}$$

which proves theorem 3.1. ■

Theorem 3.1 shows that when the function $f(\cdot)$ is strongly convex and L-smooth, the accelerated gradient descent algorithm converges with the convergence rate of $\mathcal{O}\left(\sqrt{\frac{L}{\mu}}\log\frac{1}{\epsilon}\right)$. The following theorem studies the convergence of Algorithm 1 when $f(\cdot)$ is L-smooth.

Theorem 3.4 *Let $f(\cdot) : \mathbb{R}^d \to \mathbb{R}$ be a twice differentiable and L-smooth function. Assume that $\boldsymbol{x}^*$ is the minimizer and set $\eta = \frac{1}{L}$, $\gamma_t = \frac{\lambda_t - 1}{\lambda_t + 1}$ where $\lambda_0 = 0$ and $\lambda_t = \frac{1+\sqrt{1+4\lambda_{t-1}^2}}{2}$. Then it holds that*

$$f(\boldsymbol{y}_t) - f(\boldsymbol{x}^*) \leq \frac{2L\|\boldsymbol{x}_1 - \boldsymbol{x}^*\|_2^2}{t^2} \tag{40}$$

hence we reach ϵ-close in value after $T := \sqrt{\frac{2LR^2}{\epsilon}}$ iterations.

The proof of the theorem 3.4 follows similar arguments to the classic GD smooth case. This theorem shows that accelerated gradient descent algorithm converges with a convergence rate of $\mathcal{O}(\sqrt{\frac{L}{\epsilon}})$.

CS295 Optimization for Machine Learning

Instructor: Ioannis Panageas Scribed by: Will Overman

Lecture 8. Intro to non-convex optimization. GD avoids saddle points.

1 Introduction

Up to this point in the course we have only considered convex functions. These are relatively easy to optimize due to either having a unique minimizer or having minimizers that all give the same function value. However, often in practice the function we would like to optimize does not have this uniqueness property and are in fact *non-convex*.

2 Understanding Gradient Descent on Quadratic Functions as a Linear Dynamical System

2.1 Linear Dynamical Systems

Definition 2.1 *Let A be an $n \times n$ matrix. Then the dynamics of the n-dimensional vectors described by $x_{t+1} = Ax_t$, with initial vector x_0, is referred to as a* ***linear dynamical system****.*

Remark 2.2 *Since A does not depend on time we clearly have that $x_t = A^t x_0$.*

Observe that the 0 vector is a fixed point of these dynamics for any $n \times n$ matrix A. So naturally we can ask whether the dynamics for a given matrix A will converge to 0? This depends on the eigenvalues of A.

Lemma 2.1 *Let A be a symmetric matrix of size $n \times n$. Assume that $||A||_2 < 1$, that is, the singular values of A are all less than 1. Then for all $x_0 \in \mathbb{R}^n$ we have that*

$$\lim_{t \to 0} x_t = 0.$$

Proof: Since A is symmetric all of its eigenvalues $\lambda_1, ..., \lambda_n$ are real and their corresponding eigenvectors $v_1, ..., v_n$ span the whole space $\mathbb{R}^n$. Thus we can write the initial vector as

$$x_0 = \sum_{k=1}^{n} c_k v_k.$$

Then using the definition of eigenvectors this gives us

$$A^t x_0 = \sum_{k=1}^{n} c_k \lambda_k^t v_k.$$

Now using $||A||_2 < 1$ we know all the singular values of A are less than 1, and since A is symmetric this implies that $|\lambda_k| < 1$ for all eigenvalues. Thus we have $\lim_{t\to 0} \lambda_k^t = 0$ for all k. This gives us the desired result

$$\lim_{t\to 0} A^t x_0 = \lim_{t\to 0} x_t = 0.$$

■

Remark 2.3 *We can prove the same result as lemma 2.1 without assuming that A is symmetric although this requires considering the spectral radius and Jordan decomposition of the matrix A.*

If $||A||_2 \geq 1$, then we can still have convergence to $\mathbf{0}$, but only for restricted cases of the initial vector x_0. In particular,

Lemma 2.2 *Let A be a symmetric $n \times n$ matrix and assume $v_1, ..., v_k$ are the eigenvectors of A with eigenvalues of magnitude less than 1. Then if $x_0 \in span(v_1, ..., v_k)$, then $\lim_{t\to\infty} x_t = 0$.*

Proof: Same as proof of 2.1 except we only sum over the k eigenvectors $x_0 = \sum_{i=1}^{k} c_i v_i$. ■

If the initial vector x_0 does not sit within the span of these eigenvectors, then x_t will diverge as $t \to \infty$.

2.2 Gradient Descent on Quadratic Functions

Definition 2.4 *We say that $f(x)$ is a quadratic function if it is expressible as $f(x) = x^T Ax$.*

Remark 2.5 *We can assume that A is symmetric since for any A we have $f(x) = \frac{1}{2}x^T Ax + \frac{1}{2}x^T A^T x = \frac{1}{2}x^T(A + A^T)x$, and we know $A + A^T$ is always symmetric.*

Consider performing gradient descent on a quadratic function $f(x) = x^T Ax$. Then the gradient is given by $\nabla f(x_t) = Ax_t$. Thus we have that the update rule is

$$\begin{aligned} x_{t+1} &= x_t - \epsilon A x_t \\ &= (I - \epsilon A)x_t \end{aligned}$$

Now we can see that since A is symmetric and the identity matrix is of course symmetric,that the matrix $I - \epsilon A$ is symmetric as well. This suggests that we can interpret gradient descent on quadratic functions as a linear dynamical system with matrix $I - \epsilon A$.

Lemma 2.3 *Let A be a symmetric matrix of size $n \times n$ and let L be the maximum magnitude of any eigenvalue of A. Set $\epsilon < \frac{1}{L}$. Suppose $x = 0 \in \mathbb{R}^n$ is a strict local minimum of f; then gradient descent converges to 0 for all initializations $x_0 \in \mathbb{R}^n$.*

Proof: Observe that if A had a negative eigenvalue then moving in that direction would decrease the value of the function $f(x) = x^T Ax$, which would contradict 0 being a strict local minimum. Hence A must be positive definite. Now since we defined $\epsilon < 1/L$ we know that we must have that all of the eigenvalues of ϵA lie in the open interval $(0, 1)$.

Then clearly the eigenvalues of $-\epsilon A$ lie in $(-1, 0)$ and for $I - \epsilon A$ they then must again lie in $(0, 1)$. Thus we have a linear dynamical system $x_{t+1} = (I - \epsilon A)x_t$ with eigenvalues less than 1 and hence we are guaranteed convergence of this process by Lemma 2.1. In particular,

$$\lim_{t\to\infty} x_t = \lim_{t\to\infty} (I - \epsilon A)^t x_0 = 0.$$

■

This result is not particularly surprising though since defining the function $f(x) = x^T Ax$ for symmetric A with only positive eigenvalues gives a convex function $f(x)$, so we already knew that gradient descent would converge. So what if A has negative eigenvalues? Then $I - \epsilon A$ will have eigenvalues greater than 1, and 0 will no longer be a local minimum but instead a *saddle point*, which brings us to our next section.

3 Transitioning to non-Convexity

Definition 3.1 *A point x^* is a critical or first-order stationary point of f. if $\nabla f(x^*) = 0$.*

Definition 3.2 *A critical point x^* of f is a saddle point if for all neighborhoods $\mathcal{U}$ around x^* there are $y, z \in \mathcal{U}$ such that $f(z) \leq f(x^*) \leq f(y)$.*

Definition 3.3 *A critical point x^* of f is a strict saddle if $\lambda_{\min}(\nabla^2 f(x^*)) < 0$, that is, the minimum eigenvalue of the Hessian is negative.*

When we are interested in minimizing a function, we will find its critical points and then try to determine which of thee are mininimizers, which are maximiziers, and which are saddle points. The maximizers are generally easy to identity, leaving the mian difficulty in distinguishing minimizers from saddle points. By Definition 3.3, we can see that a strict saddle point can be found by checking the eigenvalues of the Hessian. However, it is computationally difficult in general to distinguish saddle points from minimizers. Thus it is useful to understand under which conditions we converge to a saddle point.

Lemma 3.1 *Let A be an invertible symmetric matrix of size $n \times n$ and let L be the maximum magnitude of an eigenvalue of A. Set $\epsilon < 1/L$. Let $E^s = \{v_1, ..., v_k\}$ be the eigenvectors of A that correspond to eigenvalues greater than 0, and $E^u = \{v_{k+1}, ..., v_n\}$ be the remaining eigenvectors. Then*

$$\lim_t x_t = \lim_t (I - \epsilon A)^t x_0 = 0$$

if and only if $x_0 \in span(v_1, ..., v_k)$.

Lemma 3.2 *Let A be a symmetric invertible matrix of maximum eigenvalue magnitude L such that E^s has dimension $k < n$ (which implies that $x = 0$ is a strict saddle for the quadratic function $f(x) = \frac{1}{2}x^T Ax$). Set $\epsilon < 1/L$. For any continuous distribution D, if we sample initialization x_0 from D, GD converges to $x = 0$ with probability zero.*

Theorem 3.3 *For any $\epsilon > 0$, assume the differentiable quadratic function f is L-smooth and let $\alpha = 1/L$. Moreover, let $f(x^*)$ be the global minimum of f. Then the gradient descent process*

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

will visit an ϵ-stationary point at least once in at most $T = \frac{2L(f(x_0)-f(x^*))}{\epsilon^2}$ iterations.

Proof: Recall the descent lemma

$$f(x - \frac{1}{L}\nabla f(x)) - f(x) \leq -\frac{1}{2L}||\nabla f(x)||_2^2.$$

Now assume for the sake of contradiction that we have not visited an ϵ-stationary point in T iterations. Then we have $||\nabla f(x_t)||_2 > \epsilon$ for all $t = 1, ..., T$. Then we get that

$$f(x_T) - f(x_{T-1}) + f(x_{T-1}) - f(x_{T-2}) + ... + f(x_1) - f(x_0) < -\frac{\epsilon^2 T}{2L}.$$

Therefore

$$f(x^*) - f(x_0) \leq f(x_T) - f(x_0) < -\frac{\epsilon^2 T}{2L} = f(x^*) - f(x_0)$$

which gives us a contradiction, hence proving the theorem. ■

This result above is only for quadratic functions f, but the result can in fact be extended to general $f : \mathbb{R}^n \to \mathbb{R}$.

Theorem 3.4 *Let $f : \mathbb{R}^n \to \mathbb{R}$ ve a twice differentiable function which is L-smooth and let 0 be a strict saddle point. Let $\epsilon < 1/L$. For any continuous distribution D. If we sample initialization x_0 from D, then GD always converged to 0 with probaiblity 0.*

Proof: We will prove a proof sketch. Clearly GD on a general function f is still a dynamical system $x_{t+1} = x_t - \epsilon \nabla f(x_t)$, although not a linear dynamical system. But, we can perform the linearization $x_{t+1} = (I - \epsilon \nabla^2 f(0))x_t + \text{error}(t)$, with $\text{error}(t) = O(||x_t||_2^2)$. But formally the proof requires the use of the stable manifold theorem. ■

CS295 Optimization for Machine Learning

Instructor: Ioannis Panageas Scribed by: Gavin Kerrigan, Stephen McAleer

Lecture 10. Introduction to Min-Max Optimization.

1 Generative Adversarial Networks (GANs)

Generative adversarial networks are a class of deep generative models, where G_θ is a generator parameterized by θ and D_w is a discriminator parameterized by w.

Let the true data distribution be Q and let F be some distribution of noise. The general form of the optimization objective in GANs is

$$\min_\theta \max_w \mathbb{E}_{x\sim Q}\left[D_w(x)\right] - \mathbb{E}_{z\sim F}\left[D_w(G_\theta(z))\right] \tag{1}$$

In the original GAN paper [1], the optimization objective used was

$$\min_\theta \max_w \mathbb{E}_{x\sim p_{\text{data}}}\left[\log D_w(x)\right] + \mathbb{E}_{z\sim p_{\text{noise}}}\left[\log(1 - D_w(G_\theta(z)))\right] \tag{2}$$

Lemma 1.1 (Optimaliy.) *For a fixed generator G, the optimal discriminator D has density*

$$D_{w^*}(x) = \frac{p_{data}(x)}{p_{data}(x) + p_G(x)} \tag{3}$$

where $p_G(x)$ is the implicit distribution of the generator.

Proof: For a fixed G_θ , the discriminator tries to maximize (Equation (2)):

$$\mathbb{E}_{x\sim p_{\text{data}}}\left[\log D_w(x)\right] - \mathbb{E}_{z\sim p_{\text{noise}}}\left[\log(1 - D_w(G_\theta(x)))\right] \tag{4}$$

$$= \int_x \log D_w(x) p_{\text{data}}(x)dx + \int_x \log(1 - D_w(G_\theta(z)))p_{\text{noise}}(z)dz \tag{5}$$

$$= \int_x \log D_w(x) p_{\text{data}}(x)dx + \int_x \log(1 - D_w(x))p_{\text{G}}(x)dx \qquad \text{(substitue } x = G_\theta(z)) \tag{6}$$

Note that $f(y) = a\log y + b\log(1-y)$ is maximized at $y^* = \frac{a}{a+b}$. Hence, the optimal discriminator is given by

$$D_{w^*}(x) = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_{\text{G}}(x)}. \tag{7}$$

■

We are also interested in the form of the optimal generator. For the optimal discriminator (solved for above), we want to minimize the cost function

$$C(G) = \max_w \mathbb{E}_{x\sim p_{\text{data}}}\left[\log D_w(x)\right] + \mathbb{E}_{z\sim p_{\text{noise}}}\left[\log(1 - D_w(G_\theta(z)))\right] \tag{8}$$

$$= \mathbb{E}_{x\sim p_{\text{data}}}\left[\log \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_{\text{G}}(x)}\right] + \mathbb{E}_{x\sim p_{\text{G}}}\left[\log \frac{p_{\text{G}}(x)}{p_{\text{data}}(x) + p_{\text{G}}(x)}\right] \tag{9}$$

Theorem 1.2 (Global Solution.) *The global minimum of $C(G)$ is achieved if and only if*

$$p_G(x) = p_{data}(x) \tag{10}$$

Proof: Note that if $p_{\text{data}} = p_{\text{G}}$, we can evaluate Equation (9) to obtain $C(G) = -\log 4$.

Recall that the KL-divergence between distributions p and q is defined as $\text{KL}(p||q) = \mathbb{E}_{x\sim p}\left[\log \frac{p(x)}{q(x)}\right]$. Moreover, $\text{KL}(p||q) \geq 0$, and equality is achieved if and only if $p = q$.

Now, the cost $C(G)$ can be expressed as

$$\begin{aligned} C(G) &= -\log 4 + \text{KL}\left(p_{\text{data}}||\frac{p_{\text{data}} + p_{\text{G}}}{2}\right) + \text{KL}\left(p_{\text{G}}||\frac{p_{\text{data}} + p_{\text{G}}}{2}\right) \\ &\geq -\log 4 \end{aligned} \tag{11}$$

Hence, $-\log(4)$ is the global minimum of $C(G)$ and is achieved if and only if $p_{\text{data}} = \frac{p_{\text{data}}+p_{\text{G}}}{2}$, i.e. $p_{\text{G}}(x) = p_{\text{data}}(x)$. ■

2 Min-Max Optimization

In the previous section, we saw how GANs motivate the study of optimization problems of the form

$$\min_{x\in\mathcal{X}} \max_{y\in\mathcal{Y}} f(x, y) \tag{12}$$

where f is a continuous function $f : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$, and $\mathcal{X}$ and $\mathcal{Y}$ are compact. In general, $\min_{x\in\mathcal{X}} \max_{y\in\mathcal{Y}} f(x, y) \neq \max_{y\in\mathcal{Y}} \min_{x\in\mathcal{X}} f(x, y)$. A counterexample showing that equality does not always hold is $f(x, y) = \sin(x + y)$. However, the next theorem will show that this holds in the special case of f being convex-concave.

One direction is always true, even in the case of f being non-convex non-concave.

Lemma 2.1 (Min-Max Inequality) *For any function $f : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$,*

$$\inf_{x\in\mathcal{X}} \sup_{y\in\mathcal{Y}} f(x, y) \geq \sup_{y\in\mathcal{Y}} \inf_{x\in X} f(x, y) \tag{13}$$

Proof: Let $g(y) = \inf_{x\in X} f(x, y)$. By definition, $g(y) \leq f(x, y)$ for any $(x, y) \in \mathcal{X} \times \mathcal{Y}$. Hence, $\sup_{y\in\mathcal{Y}} g(y) \leq \sup_{y\in\mathcal{Y}} f(x, y)$ for any $x \in \mathcal{X}$. Since this holds for each x, it must also hold for the infimum: $\sup_{y\in\mathcal{Y}} g(y) \leq \inf_{x\in\mathcal{X}} \sup_{y\in\mathcal{Y}} f(x, y)$. By the choice of g, this last line is equivalent to $\sup_{y\in\mathcal{Y}} \inf_{x\in\mathcal{X}} f(x, y) \leq \inf_{x\in\mathcal{X}} \sup_{y\in\mathcal{Y}} f(x, y)$. ■

Definition 2.1 *We say that $f(x, y)$ is a convex-concave function if $g_y(x) = f(x, y)$ is convex for each fixed y and $h_x(y) = f(x, y)$ is concave for each fixed x.*

Theorem 2.2 (Minimax Theorem (John von Neumann)) *Let $\mathcal{X} \subset \mathbb{R}^n$ and $\mathcal{Y} \subset \mathbb{R}^m$ be compact, convex sets. If f is a continuous function that is convex-concave, we have*

$$\min_{x\in\mathcal{X}} \max_{y\in\mathcal{Y}} f(x, y) = \max_{y\in\mathcal{Y}} \min_{x\in\mathcal{X}} f(x, y) \tag{14}$$

Proof: Let's use no-regret learning for both players. Let $x_1, ..., x_T$ and $y_1, ..., y_T$ be the iterates as advised by some no-regret algorithm and define $\hat{x} = \frac{1}{T}\sum_{i=1}^{T} x_i$ and $\hat{y} = \frac{1}{T}\sum_{i=1}^{T} y_i$ and $T = \Theta(\frac{1}{\epsilon^2}$. Choose any x, then from the no-regret property for x we get that

$$\begin{aligned} \frac{1}{T}\sum_t f(x_t, y_t) &\leq \frac{1}{T}\sum_t f(x, y_t) + \epsilon \\ &\leq f(x, \hat{y}) + \epsilon \end{aligned} \tag{15}$$

Where the second inequality follows by concavity. Now choose any y, then from the no-regret property for y we get that

$$\begin{aligned} \frac{1}{T}\sum_t f(x_t, y_t) &\geq \frac{1}{T}\sum_t f(x_t, y) - \epsilon \\ &\geq f(\hat{x}, y) - \epsilon \end{aligned} \tag{16}$$

Where the second inequality follows by convexity. We conclude that for all x, y we have

$$\max_y f(\hat{x}, y) - 2\epsilon \leq \min_x f(x, \hat{y}) \tag{17}$$

Finally, we get

$$\begin{aligned} \max_y \min_x f(x, y) &\geq \min_x f(x, \hat{y}) \\ &\geq \max_y f(\hat{x}, y) - 2\epsilon \\ &\geq \min_x \max_y f(x, y) - 2\epsilon \end{aligned} \tag{18}$$

Setting $\epsilon \to 0$, we are done. ■

References

[1] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio. *Generative Adversarial Networks.*

CS295 Optimization for Machine Learning

Instructor: Ioannis Panageas Scribed by: Thorben Tröbst

Lecture 11. Min-max Optimization: Local Nash and Last Iterate Convergence.

1 Min-Max with Bilinear Objectives

In the last lecture we discussed that we are often interested in *last iterate convergence*. Unfortunately, Gradient Descent Ascent (GDA) may in general divergece with respect to the last iterate, even if the objective is bilinear, i.e. of the form $x^T Ay$.

Consider the unconstrained problem

$$\min_{x\in\mathbb{R}^n} \max_{y\in\mathbb{R}^m} x^T Ay$$

and run the *continuous* GDA algorithm on it which effectively corresponds to GDA with an infinitesimal step-size. This means we consider $x(t)$ and $y(t)$ as solutions to the system of ODEs

$$\begin{aligned} x'(t) &= -\eta Ay(t), \\ y'(t) &= \eta A^T x(t). \end{aligned}$$

Lemma 1.1 *During continuous GDA, $||x(t)||_2^2 + ||y(t)||_2^2$ is constant with respect to t.*

Proof: Simply compute the time-derivative:

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t}||x(t)||_2^2 &= 2\langle x(t), x'(t)\rangle \\ &= -2\eta x(t)^T Ay(t) \end{aligned}$$

and similarly

$$\frac{\mathrm{d}}{\mathrm{d}t}||y(t)||_2^2 = 2\eta x(t)^T Ay(t)$$

which ultimately implies that

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(||x(t)||_2^2 + ||y(t)||_2^2\right) = 0.$$

■

Note that the bilinear min-max problem always has a saddle point at 0. Depending on A, this may in fact be the *unique* saddle point (in particular if A is invertible). Thus, continuous GDA will never converge in this setting.

1.1 Optimism

Consider the general convex-concave min-max problem

$$\min_{x\in\mathbb{R}^n}\max_{y\in\mathbb{R}^m} f(x,y).$$

Recall that the GDA update step with step size η was defined as

$$x^{(t+1)} = x^{(t)} - \eta\nabla_x f(x^{(t)}, y^{(t)}),$$
$$y^{(t+1)} = y^{(t)} + \eta\nabla_y f(x^{(t)}, y^{(t)}).$$

As we saw in the last section, this update step may cycle or diverge on bilinear objectives. As a fix, we can add an additional *negative momentum* term. This so-called *optimistic* GDA is given by

$$x^{(t+1)} = x^{(t)} - \eta\nabla_x f(x^{(t)}, y^{(t)}) + \frac{\eta}{2}\nabla_x f(x^{(t-1)}, y^{(t-1)}),$$
$$y^{(t+1)} = y^{(t)} + \eta\nabla_y f(x^{(t)}, y^{(t)}) - \frac{\eta}{2}\nabla_y f(x^{(t-1)}, y^{(t-1)}).$$

For an intuitive understanding of this new update, consider Figure 1.

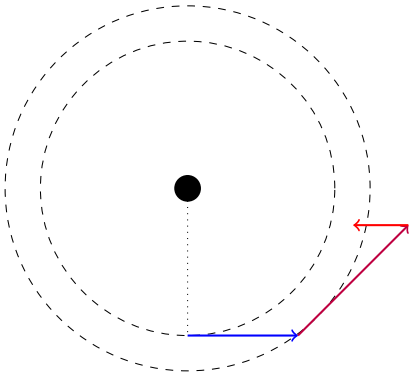


Figure 1: The blue and purple arrows represents normal GDA update steps without optimism. The red arrow represents the negative momentum term. One can see that if one follows the red arrow after the purple arrow (i.e. OGDA), then one avoids the cycling / outwards spiraling behavior.

Theorem 1.2 *Consider the bilinear game*

$$\min_{x\in\mathbb{R}^n}\max_{y\in\mathbb{R}^m} x^T A y$$

where A has full rank. Optimistic GDA converges pointwise (wrt. last iterate) and reaches an ϵ-neighborhood of the saddle point 0 in

$$O\left(\frac{\lambda_{\max}(AA^T)}{\lambda_{\min}(AA^T)}\log\frac{1}{\epsilon}\right)$$

iterations with learning rate $\eta = \frac{1}{4\sqrt{\lambda_{\max}(AA^T)}}$.

Proof: Note that due to the bilinear nature of the objective, we can write the update step as

$$\begin{pmatrix} x^{(t+1)} \\ y^{(t+1)} \end{pmatrix} = \left(I - \begin{pmatrix} 0 & 2\eta A \\ -2\eta A^T & 0 \end{pmatrix} \right) \begin{pmatrix} x^{(t)} \\ y^{(t)} \end{pmatrix} + \begin{pmatrix} 0 & \eta A \\ -\eta A^T & 0 \end{pmatrix} \begin{pmatrix} x^{(t-1)} \\ y^{(t-1)} \end{pmatrix}.$$

This is not a first-order recurrence, but we can rewrite it as one via the classical trick of increasing the dimension:

$$\begin{pmatrix} x^{(t+1)} \\ y^{(t+1)} \\ x^{(t)} \\ y^{(t)} \end{pmatrix} = \begin{pmatrix} I & -2\eta A & 0 & \eta A \\ 2\eta A^T & I & -\eta A^T & 0 \\ I & 0 & 0 & 0 \\ 0 & I & 0 & 0 \end{pmatrix} \begin{pmatrix} x^{(t)} \\ y^{(t)} \\ x^{(t-1)} \\ y^{(t-1)} \end{pmatrix}.$$

Then by the standard theory of matrix powers, we know that this convergences if all the eigenvalues of the above matrix are at most 1. One can show that this is the case if η is small enough, in particular for $\eta \leq \frac{1}{4\sqrt{\lambda_{\max)}(AA^T)}}$. ■

Theorem 1.2 shows that negative momentum terms allow us to avoid cycling / divergence behavior with *bilinear* objectives. However, there are extensions of this result to more general, differentiable convex-concave objectives since these functions are at least locally bilinear.

2 Bilinear Min-Max in Simplices

In the previous sections we saw how to achieve last-iterate convergence in the unconstrained setting. But that do we do if x and y are constrained to be in convex sets? In particular, consider the problem

$$\min_{x \in \Delta_n} \max_{y \in \Delta_m} x^T A y$$

where x and y must lie in the n and m-simplices respectively.

We have seen in previous lectures that problems of the form

$$\min_{x \in \Delta_n} f(x)$$

can be solved using the Multiplicative Weights Update (MWU) method. We also already know that if both the min and the max player use MWU, then the average converges.

Just as with GDA, we will not generally have last-iterate convergence using this approach. But we can use a negative-momentum term to get *optimistic MWU*:

$$x_i^{(t+1)} = x_i^{(t)} \frac{1 + 2\eta(Ay^{(t)})_i - \eta(Ay^{(t-1)})_i}{\sum_j x_j^{(t)}(1 + 2\eta(Ay^{(t)})_j - \eta(Ay_j^{(t-1)})},$$
$$y_i^{(t+1)} = y_i^{(t)} \frac{1 - 2\eta(A^T x^{(t)})_i + \eta(A^T x^{(t-1)})_i}{\sum_j y_j^{(t)}(1 - 2\eta(A^T x^{(t)})_j - \eta(A^T x_j^{(t-1)})}.$$

Theorem 2.1 *Let A be the payoff matrix of a zero-sum game and assume that the game has a unique Nash equilibrium. Then for η sufficiently small (may be exponentially small), starting from a uniform distribution, $x^{(t)}$ and $y^{(t)}$ converge to the Nash equilibrium under optimistic MWU.*

3 Min-Max in General Settings and Local Nash Equilibria

Finally, let us consider settings in which the objective function f is not convex-concave. For simple minimization problems, we were able to generalize the convergence results of gradient descent type algorithms: instead of showing convergence to the global minimum one can instead show convergence to some notion of local minimum. We would like to give similar results for min-max problems but first we need to find a good notion of local optimality. A natural one is that of a *local Nash equilibrium*.

Definition 3.1 *A critical point (x^*, y^*) is a* local Nash equilibrium *if there exists a neighborhood U around (x^*, y^*) so that for all $(x, y) \in U$ we have*

$$f(x^*, y) \leq f(x^*, y^*) \leq f(x, y^*).$$

Local Nash equilibria are of particular interest in machine learning contexts because they represent a kind of *robustness*. One can then show that local Nash equilibria are stable under GDA and optimistic GDA updates. Recall that by *stability* near (x^*, y^*) we mean that (optimistic) GDA converges to (x^*, y^*) in a neighborhood of (x^*, y^*).

Theorem 3.1 *Under some mild assumptions on $f(x, y)$ and the step-size, we have*

$$\textit{Local Nash} \subseteq \textit{GDA-stable} \subseteq \textit{OGDA-stable}.$$

Unfortunately, the notion of local Nash equilibria is not nearly as robust as that of local optimality for minimization problems. In particular, the inclusions in Theorem 3.1 are in general strict. Furthermore, local Nash equilibria may not always exist.

Lemma 3.2 *There are functions f with critical points that are stable under GDA but not local Nash equilibria. An example is $f(x, y) = -\frac{1}{8}x^2 - \frac{1}{2}y^2 + \frac{6}{10}xy$.*

Proof: We can compute the Jacobian of the GDA update around 0. This yields

$$J = \begin{pmatrix} 1 + \frac{1}{4}\eta & -\frac{6}{10}\eta \\ \frac{6}{10}\eta & 1 - \eta \end{pmatrix}.$$

Now one may check that for $\eta < 1.34$, J has eigenvalues below 1 and thus GDA is contracting around $(0, 0)$. Therefore this critical point is GDA-stable.

On the other hand, $(0, 0)$ is clearly not a local Nash equilibrium as for any $x \neq 0$ we have $f(x, 0) < f(0, 0)$. ■

CS 295 Optimization for Machine Learning

Instructor: Ioannis Panageas Scribed by: Deepanway Ghosal, Wayne Lin

Lecture 12-13. Introduction to Multi-armed Bandits.

1 Introduction

Multi-armed bandit problems are examples of sequential decision problems where a fixed set of resources has to be allocated between alternative choices to maximize the expected gain. This is a classic framework with exploration-exploitation trade-off, where algorithms make decisions over time under uncertainty. At the beginning, the properties of the different alternative choices are unknown or partially known, and may become better understood as resources are allocated to those choices at the time of each sequential decision. Hence, there is always this balance between staying with the choice which gave the highest reward in the past and exploring new options which may give better rewards in the future. Therefore, the goal is to design algorithms with the expectation that they will be able to choose the optimal sequence of actions and consequently give us higher rewards or payoffs.

More formally, the multi-arm bandit problem is said to be a sequential allocation problem governed by a set of actions. At each time step an action is selected, and a reward is observed. The objective is to maximize the total reward observed for a given time horizon (total number of time steps). We first introduce the framework:

The player is given K different arms (actions) and a total time horizon of T. Both K & T are known. At each time step $t = 1, 2, .., T$:

1. The player chooses an arm a_t.
2. A reward $r_t \in [0, 1]$ is observed.

In this lecture we will analyze *stochastic bandits* and *adversarial bandits*. We have the following set of assumptions for *stochastic bandits*,

- The reward for each arm $a \in [K]$ is independent and identically distributed (IID). For each arm a there is a distribution D_a over reals, called the *reward distribution*. This distribution is unknown to the player. Each time arm a is chosen by the player, the reward is sampled independently from the distribution D_a.
- The player observes only the reward for the selected arm. Rewards for other arms, that were not selected, are not observed.

This assumption of having IID rewards is a feature of *stochastic bandits*. However, in *adversarial bandits*, instead of rewards, we have costs and the costs are not IIDs. In this case we have the following assumptions,

- The costs for all arms and all rounds are chosen in advance by the adversary. Each time arm a is chosen by the player in some round t, the cost $\in [0, 1]$ for that arm a in that round t is revealed to the player.
- The player observes only the cost for the selected arm. Costs for other arms, that were not selected, are not observed.

Notation: We introduce the following set of notations first. Arms are denoted by a and the set of all arms is A. Rounds are denoted by t, and the arm chosen at round t is denoted by a_t. The mean reward of arm a is denoted by $\mu(a) = E[D_a]$. The arm with the best mean reward will then have a reward of $\mu^* = \max_{a \in A} \mu(a)$.

Now, the strategy of the player is to maximize the sum of all collected rewards. The best reward is obtained when the optimal arm (the arm with the best mean reward μ^*) is chosen at every round. However, any information about $\mu(a)$ or D_a is unknown to the player at the very beginning. The strategy of maximizing the sum of rewards, is therefore, equivalent to minimizing the regret, where the regret is defined as,

$$R(T) = \mu^* T - \sum_{t=1}^{T} \mu(a_t)$$

Intuitively, the regret describes how far off are we from the optimal reward at each round and sums up those differences over the whole time horizon T. Choosing a bad arm would mean we will end up with a comparatively higher regret, whereas, choosing the best arm at every round would mean we will have a regret of 0 and the highest possible reward. So, a strategy of minimizing the regret would accomplish our goal of maximizing the reward.

Note that, $R(T)$ is a random variable as the arm a_t chosen at round t is a random quantity and may depend on the randomness in the rewards and the algorithm. So, for analysis we will consider the expected value of the regret: $\mathbb{E}[R(T)]$. We will analyze the asymptotic dependence of regret $\mathbb{E}[R(T)]$ on the time horizon T.

2 Algorithms (Stochastic Setting)

In this section, we will present a number of different algorithms that the player can employ and we will analyze the regret in terms of time horizon T and number of arms K.

2.1 Explore First Algorithm

The most simple strategy could be to first estimate the expected reward for all the arms and then use the arm with the maximum estimated reward for the rest of the rounds. We do this by keeping

the initial few rounds only for exploration. In this phase, we explore all the arms uniformly. Then in the next phase, we exploit the optimal arm for the rest of the rounds.

1. Exploration phase: try each arm N/K times.
2. Select the arm a^* with the highest average reward (break ties arbitrarily).
3. Exploitation phase: play a^* in all remaining $T - N$ rounds.

The parameter N is fixed in advance as a function of T and K. We will choose it in a way such that it minimizes the regret.

Let us denote the average reward for arm a after the exploration phase as $\hat{\mu_a}$. We can use Hoeffding inequality to bound the quantity $|\hat{\mu_a} - \mu_a|$ as following,

$$Pr\{|\hat{\mu_a} - \mu_a| \leq r(a)\} \geq 1 - \frac{2}{T^4} \tag{1}$$

Ideally we would want the average reward after the exploration phase to be a good estimate of the true expected reward. Hence the quantity $|\hat{\mu_a} - \mu_a|$ should be small, and using the Hoeffding inequality we can say that $|\hat{\mu_a} - \mu_a|$ can be made smaller than bounding radius $r(a) = \sqrt{\frac{2K \log T}{N}}$ with a large probability of at least $1 - \frac{2}{T^4}$

Now we define the *clean event* to be a event where inequality (1) holds for all arms. It follows that the probability of the *bad event*, which is the complement of the *clean event*, is going to be very small. So, for the rest of the analysis we would only consider the clean event because it happens with such a high probability.

We start the analysis in the case of a clean event. Suppose a^* is the arm with the best true expected reward. However, after the exploration phase, some other arm $a \neq a*$ is chosen because it has a higher average reward i.e. $\hat{\mu}(a) > \hat{\mu}(a^*)$. From (1), we can thus say,

$$\mu_a + r(a) \geq \hat{\mu_a} > \hat{\mu}(a^*) \geq \mu(a^*) - r(a^*)$$

or,

$$\hat{\mu}(a^*) - \mu_a \leq r(a) + r(a^*) = 2\sqrt{\frac{2K \log T}{N}} \tag{2}$$

Now, the N rounds in exploration phase can be considered to contribute at most N (regret of maximum 1 in each round) regret. For the exploitation phase, each round contributes $\hat{\mu}(a^*) - \mu_a$ which is bounded by $2\sqrt{\frac{2K \log T}{N}}$ from inequality (2). Hence the total regret after N rounds of exploration and $T - N$ rounds of exploitation,

$$\begin{aligned} R(T) &\leq N + 2\sqrt{\frac{2K \log T}{N}}(T - N) \\ &\leq N + \sqrt{\frac{8KT^2 \log T}{N}} \end{aligned} \tag{3}$$

The two summands in inequality (3) are monotonically increasing and decreasing with N. Hence, the sum can be minimized by making them approximately equal, which happens when we choose $N = 2T^{2/3}(K \log T)^{1/3}$. We can now put this value of N back in (3) to obtain

$$R(T) \leq 4T^{2/3}(K \log T)^{1/3} \tag{4}$$

To complete the result, we also need to analyze the case of the *bad events*. For *bad events* the regret can be at most T for T rounds. Moreover, the *bad event* happens when at least one arm doesn't satisfy (1) which happens with probability $\frac{K}{T^4}$ (from union bound of probability) or probability of $O(1/T^3)$. Hence, the overall regret,

$$\begin{aligned}
\mathbb{E}[R(T)] &= \mathbb{E}[R(T)|\text{clean event}] * Pr[\text{clean event}] + \mathbb{E}[R(T)|\text{bad events}] * Pr[\text{bad events}] \\
&\leq 4T^{2/3}(K \log T)^{1/3} + T * O(T^{-3}) \\
&\leq O(T^{2/3}(K \log T)^{1/3})
\end{aligned} \tag{5}$$

Theorem 2.1 *The explore first algorithm achieves regret* $O(T^{2/3}(K \log T)^{1/3})$

2.2 Epsilon Greedy Algorithm

The *explore first* algorithm performs very poorly in the exploration phase. A simple alternative way to alleviate this would be to use a greedy method. In this case we use a greedy action selection method to maximize current reward by exploiting current knowledge. In particular, we behave greedily most of time by selecting the best arm from our knowledge of previous rounds, but once in a while with a small probability ϵ we perform exploration by randomly selecting any one of the possible arms. As a result, the exploration phase becomes more spread over time and we can analyze the regret bounds even for small t. This is performed in the *epsilon greedy* algorithm.

```
for round t = 1, 2, .. T do
    Toss a coin with success probability ε_t;
    if success then
        explore: choose an arm uniformly at random;
    else
        exploit: choose the arm with the highest observed average reward so far;
    end
end
```

We analyze the regret bound of this algorithm by fixing round t. After round t, for each arm a, following Hoeffding inequality, we will have,

$$Pr\{|\hat{\mu_a} - \mu_a| \leq \epsilon\} \geq 1 - 2e^{-2\epsilon^2(t\epsilon_t/K)} \tag{6}$$

Among the first t rounds, the exploration happens only in $t\epsilon_t$ rounds and hence, each arm on average will be explored $t\epsilon_t/K$ times. Naturally, some arms will be chosen more in the exploitation phase. For those arms, we will have even tighter probability bounds and hence, for the asymptotic analysis we can proceed from (6).

Proceeding in a similar way as in (2), for arm a_t chosen at round t, we will have,

$$\hat{\mu}(a^*) - \mu_{a_t} \leq 2\sqrt{\frac{2K \log t}{t\epsilon_t}} \tag{7}$$

Now, the expected value of the regret only at the particular round of t,

$$\begin{aligned} \mathbb{E}[\tilde{R}(t)] &= Pr[\text{coin toss success}] * 1 + Pr[\text{coin toss failure}] * (\hat{\mu}(a^*) - \mu_{a_t}) \\ &= \epsilon_t + (1 - \epsilon_t) * 2\sqrt{\frac{2K \log t}{t\epsilon_t}} \\ &\leq \epsilon_t + 2\sqrt{\frac{2K \log t}{t\epsilon_t}} \end{aligned} \tag{8}$$

In (8), $\mathbb{E}(\tilde{R}(t))$ can be minimized by making the two summands approximately equal, resulting in $\epsilon_t = t^{-1/3}(K \log t)^{1/3}$. Now, the overall regret can be bounded as,

$$\begin{aligned} \mathbb{E}[R(t)] &= \mathbb{E}\Big[\sum_{1}^{t} \tilde{R}(t)\Big] \\ &\leq t * \mathbb{E}[\tilde{R}(t)] \\ &\leq t^{2/3}(K \log t)^{1/3} \end{aligned} \tag{9}$$

Theorem 2.2 *The epsilon greedy algorithm with exploration probabilities* $\epsilon_t = t^{-1/3}(K \log t)^{1/3}$ *achieves regret bound* $\mathbb{E}[R(t)] \leq O(t^{2/3}(K \log t)^{1/3})$ *for each round t.*

2.3 Upper Confidence Bound (UCB) Elimination Algorithm

2.3.1 Two-arm UCB elimination algorithm

The UCB elimination algorithm for two arms involves alternating between the arms until we find out that one arm is, with very high probability, much better than the other, at which point we abandon the inferior arm and pick the superior one forever more. We do this by defining upper and lower confidence bounds (UCB/LCB) for for the mean $\mu(a)$ of each arm a for every $t \geq 2$:

$$UCB_t(a) := \hat{\mu}_t(a) + r_t(a), \quad LCB_t(a) := \hat{\mu}_t(a) + r_t(a),$$

where $\hat{\mu}_t(a)$ is the empirical mean reward for the arm observed thus far, and the confidence radius $r_t(a)$ is defined by

$$r_t(a) = \sqrt{\frac{2 \log T}{n_t(a)}},$$

where $n_t(a) = \lceil \frac{t}{2} \rceil$ is the number of times the algorithm has tried arm a thus far. Observe that the confidence radius $r_t(a)$ only changes at times when the arm a is pulled.

The UCB elimination algorithm for two players is then as follows:

1. Alternate between the two arms a, a' until the two confidence intervals no longer intersect, i.e. $UCB_t(a) < LCB_t(a')$.
2. Eliminate the arm with the lower confidence interval (a), and use the arm (a') forever more.

Let τ be the last round in which we had not invoked the elimination rule. To analyze the expected regret $\mathbb{E}[R(T)] = \mu^*(T) - \sum_{t=1}^{T} \mu(a_t)$, where a_t is the arm chosen at time t, we condition on the "clean" event ϵ that the true means for each arm fall within the confidence interval for every time step, i.e.

$$\begin{aligned}\epsilon &= \Big\{\forall t \in [\tau],\ \text{arm } a: \quad \mu(a) \in [LCB_t(a), UCB_t(a)]\Big\} \\ &= \Big\{\forall t \in [\tau],\ \text{arm } a: \quad |\hat{\mu}_t(a) - \mu(a)| \le r_t(a)\Big\}.\end{aligned}$$

Then by the law of total expectation, we have

$$\mathbb{E}[R(T)] = \mathbb{E}[R(T) \,|\, \epsilon] \cdot \mathbb{P}[\epsilon] + \mathbb{E}[R(T) \,|\, \neg\epsilon] \cdot \mathbb{P}[\neg\epsilon].$$

For the first term (clean event), the eliminated arm cannot be the best arm, and so from time τ onwards we have zero expected regret. Now look at time τ, just before invoking the elimination rule. At time τ the confidence intervals of the two arms still intersect, and thus we have (in the clean event)

$$|\mu(a) - \mu(a')| \le 2(r_\tau(a) + r_\tau(a')).$$

Then, since $n_\tau(a) = n_\tau(a') = \frac{\tau}{2}$, we have

$$r_\tau(a) = r_\tau(a') = \sqrt{\frac{4 \log T}{\tau}}.$$

Thus

$$\mathbb{E}[R(T) \,|\, \epsilon] = |\mu(a) - \mu(a')| \cdot \frac{\tau}{2} = \sqrt{\tau \log T} = O(\sqrt{T \log T}),$$

and since $\mathbb{P}[\epsilon] \le 1$, we have that the first term,

$$\mathbb{E}[R(T) \,|\, \epsilon] \cdot \mathbb{P}[\epsilon] = O(\sqrt{T \log T}).$$

For the second term (unclean event), since the expected regret per timestep is ≤ 1, we have $\mathbb{E}[R(T) \,|\, \neg\epsilon] \le T$. It then remains to bound $\mathbb{P}[\neg\epsilon]$. By Hoeffding's inequality, we have that, for any

given arm a and time t,

$$\begin{aligned}\mathbb{P}\big(|\hat{\mu}_t(a) - \mu(a)| > r_j(a)\big) &\leq 2e^{-2n_t r_t(a)^2} \\ &= 2e^{-2n_t(a)\frac{2\log T}{n_t(a)}} \\ &= \frac{2}{T^4}.\end{aligned}$$

The only confidence radius and empirical mean changing at time t are those of arm a_t. (Thus, at time t, the only way that the event can be made unclean is for $\mu(a_t)$ to move outside of its confidence interval, since the confidence intervals for the other arms are unaffected.) Thus by union bound,

$$\begin{aligned}\mathbb{P}[\neg\epsilon] &\leq \bigcup_{t=1}^{\tau} \mathbb{P}\big(|\hat{\mu}_t(a_t) - \mu(a_t)| > r_t(a_t)\big) \\ &\leq 2\tau \cdot \frac{2}{T^4} \\ &= O(\frac{1}{T^3}).\end{aligned}$$

Thus the second term, $\mathbb{E}[R(T) \mid \neg\epsilon] \cdot \mathbb{P}[\neg\epsilon]$ is $O(\frac{1}{T^3})$.
Combining the two terms, we then get

$$\mathbb{E}[R(T)] = \mathbb{E}[R(T) \mid \epsilon] \cdot \mathbb{P}[\epsilon] + \mathbb{E}[R(T) \mid \neg\epsilon] \cdot \mathbb{P}[\neg\epsilon] = O(\sqrt{T \log T}).$$

Theorem 2.3 *The UCB elimination algorithm for two arms achieves an expected regret of*

$$O(\sqrt{T \log T}).$$

2.3.2 UCB elimination algorithm when there are more than two arms

When there are more than two arms, the UCB elimination algorithm is as follows:

1. Initially set all arms to "active".
2. Try all active arms once.
3. Deaactivate all arms a for which there exists an arm a' with $UCB_t(a) < LCB_t(a')$.
4. Repeat Steps 2 and 3 until there is one arm left, then pick it for the rest of time.

The following theorem can be shown, with proof similar to the two-arm case:

Theorem 2.4 *The UCB elimination algorithm for K arms achieves an expected regret of*

$$O(\sqrt{KT \log T}).$$

2.4 Upper Confidence Bound (UCB) algorithm

The UCB algorithm involves keeping track of (empirical) confidence bounds on the true means of each arm, and always picking the arm with the highest upper confidence bound (UCB). To do this, we first try each arm once to get an empirical mean $\hat{\mu}(a)$ for each arm a. Then, for any subsequent time t, we, similar to the UCB elimination algorithm, define the upper/lower confidence bounds for every arm a:

$$UCB_t(a) := \hat{\mu}_t(a) + r_t(a), \quad LCB_t(a) := \hat{\mu}_t(a) + r_t(a),$$

where $\hat{\mu}_t(a)$ is the empirical mean reward for the arm observed thus far, and the confidence radius $r_t(a)$ is defined by

$$r_t(a) = \sqrt{\frac{2 \log T}{n_t(a)}},$$

where $n_t(a)$ is the number of times the algorithm has tried arm a thus far. Observe that the confidence radius $r_t(a)$ only changes at times when the arm a is pulled.

Where the UCB algorithm differs from the UCB elimination algorithm in the previous section is that, after going one round trying each arm so that we obtain a confidence interval for each arm, the UCB algorithm just continues picking the arm that has the largest UCB at that time. The motivation for this is that there are two possible reasons why an arm might have a high UCB: either the empirical reward is large, which makes it likely that the true reward is large, or the confidence radius is large, which means that the arm has not been explored much yet. Either reason makes this arm worth trying, until its UCB drops below that of another arm.

The UCB algorithm is summarized as follows:

1. Try each arm once.
2. In each round t, pick the arm with the highest upper confidence bound, i.e. $\arg\max_a UCB_t(a)$.

The analysis for the UCB algorithm is similar to that for the UCB elimination algorithm in the last section: we condition on the "clean" event ϵ that the true means always lie within the confidence intervals, i.e.

$$\begin{aligned}\epsilon &= \Big\{\forall t \in [\tau],\ \text{arm } a: \quad \mu(a) \in [LCB_t(a), UCB_t(a)]\Big\} \\ &= \Big\{\forall t \in [\tau],\ \text{arm } a: \quad |\hat{\mu}_t(a) - \mu(a)| \leq r_t(a)\Big\},\end{aligned}$$

and again, by the law of total expectation we have

$$\mathbb{E}[R(T)] = \mathbb{E}[R(T) \,|\, \epsilon] \cdot \mathbb{P}[\epsilon] + \mathbb{E}[R(T) \,|\, \neg\epsilon] \cdot \mathbb{P}[\neg\epsilon].$$

We now bound the second term (contribution by unclean event) first, because it is very similar to the analysis for the UCB elimination algorithm. Again, the expected regret per timestep is ≤ 1, so $\mathbb{E}[R(T) \mid \neg\epsilon] \leq T$. It then remains to bound $\mathbb{P}[\neg\epsilon]$. By Hoeffding's inequality, we have again that, for any given arm a and time t,

$$\mathbb{P}\big(|\hat{\mu}_t(a) - \mu(a)| > r_j(a)\big) \leq \frac{2}{T^4}.$$

Now denote by a_t the arm chosen at time t. The only confidence radius and empirical mean changing at time t are those of arm a_t. Thus, by union bound, we have

$$\begin{aligned}
\mathbb{P}[\neg\epsilon] &\leq \bigcup_{t=1}^{T} \mathbb{P}\big(|\hat{\mu}_t(a_t) - \mu(a_t)| > r_t(a_t)\big) \\
&\leq 2T \cdot \frac{2}{T^4} \\
&= O(\frac{1}{T^3}).
\end{aligned}$$

Thus again, the second term (the contribution from the unclean event),

$$\mathbb{E}[R(T) \mid \neg\epsilon] \cdot \mathbb{P}[\neg\epsilon] = O(\frac{1}{T^3}). \tag{10}$$

Now we analyze the first term. (The contribution from the clean event.) Let a^* be the optimal arm. At time t, if we chose arm a_t, we have:

$$UCB_t(a_t) \geq UCB_t(a^*),$$

by virtue of the fact that we chose a_t over a^* at time t;

$$\mu(a_t) \in [LCB_t(a_t), UCB_t(a_t)] \Longrightarrow \mu(a_t) + 2r_t(a_t) \geq UCB_t(a_t);$$

and

$$UCB_t(a^*) \geq \mu(a^*),$$

which always holds in the clean event. Combining the last three inequalities, we have $\mu(a_t) + 2r_t(a_t) \geq \mu(a^*)$, which can be rearranged to give

$$\mu(a^*) - \mu(a_t) \leq 2r_t(a_t).$$

But we have that

$$r_t(a_t) = \sqrt{\frac{2\log T}{n_t(a_t)}} \leq \sqrt{\frac{2\log T}{n_T(a_t)}} = r_T(a_t),$$

so combining the last two inequalities we obtain

$$\mu(a^*) - \mu(a_t) \leq 2r_T(a_t). \tag{11}$$

Thus the contribution of any arm a to the regret is

$$\big[\mu(a^*) - \mu(a)\big] n_T(a) \leq 2r_T(a) \cdot n_T(a) = 2\sqrt{2\log T n_T(a)},$$

and so we can bound the total regret in the clean event, $\mathbb{E}[R(T) \mid \epsilon]$, by

$$2\sqrt{2\log T}\sum_a \sqrt{n_T(a)}.$$

Since $\sqrt{\cdot}$ is concave, we can use Jensen's inequality to bound the sum

$$\sum_a \sqrt{n_T(a)} \le K\sqrt{\frac{\sum_a n_T(a)}{K}} = \sqrt{TK}.$$

Thus the total regret in the clean event, $\mathbb{E}[R(T) \mid \epsilon]$, is bounded by

$$2\sqrt{2\log T}\sqrt{TK} = O(\sqrt{KT\log T}).$$

It can be seen that this term far outweighs the unclean contribution in (10), and so we have the following theorem:

Theorem 2.5 *The UCB algorithm achieves regret*

$$\mathbb{E}\big[R(T)\big] \le O(\sqrt{KT\log T}).$$

The bound on the regret in Theorem 2.5 is good when arms perform close to each other, because there is no dependence on the differences in means for the different arms. But what if we are guaranteed that the second-best performing arm is significantly worse than the best performing arm a^*? It would seem that we should then be able to eliminate the bad-performing arms earlier. Can we then obtain a better bound for the regret, with reduced dependence on T?

It turns out that we can. We can rearrange (11) to get, for every arm a with $\mu(a) < \mu(a^*)$ (i.e., every arm whose picking at any time contributes to the expected regret), an upper bound on the number of times we would try arm a in the clean event:

$$n_T(a) \le \frac{8\log T}{[\mu(a^*) - \mu(a)]^2}.$$

We can then rearrange this to get the contribution of arm a to the total regret (in the clean event):

$$n_T(a) \cdot [\mu(a^*) - \mu(a)] \le \frac{8\log T}{\mu(a^*) - \mu(a)}.$$

We can then bound the total regret in the clean event,

$$\sum_{a:\,\mu(a)<\mu(a^*} n_T(a)[\mu(a^*) - \mu(a)] \le O(\log T) \sum_{a:\,\mu(a)<\mu(a^*} \frac{1}{\mu(a^*) - \mu(a)}.$$

Again, the contribution from the unclean event is negligible compared to this, and so we have the following theorem:

Theorem 2.6 *The UCB algorithm achieves regret*

$$\mathbb{E}\big[R(T)\big] \leq O(\log T) \left(\sum_{a:\mu(a)<\mu(a^*)} \frac{1}{\mu(a^*) - \mu(a)} \right).$$

It should be noted that the bounds in Theorems 2.5 and 2.6 both hold, in any instance, for the UCB algorithm. In a given scenario, the UCB algorithm will have a fixed performance (in expected regret), but one bound may be tighter than the other (and hence more useful) depending on the situation.

3 Algorithms (Adversarial Setting)

For the adversarial setting (given in the introduction), the adversary picks costs c_a^t for each arm a and each time t. Thus where a_t denotes the arm picked at time t, the regret is given by

$$R(T) = \sum_{t\in[T]} c_{a_t}^t - \min_a \sum_{t\in[T]} c_a^t,$$

and the objective of an algorithm is to minimize the expected regret, $\mathbb{E}[R(T)]$.

The only algorithm we shall analyze for the adversarial setting shall be the Exp3 algorithm.

3.1 Exp3 Algorithm

The Exp3 algorithm follows the Multiplicative Weights Update algorithm (MWUA):

1. Initialize $w_a^0 = 1$ for all $a \in [K]$.
2. **For** $t = 1, 2, \ldots, T$ **do**
3. **Choose** arm a with probability p_a^t proportional to w_a^{t-1}.
4. **Only for** the chosen arm a, **do**
5. $w_a^t = e^{-\epsilon c_a^t / p_a^t} w_a^{t-1}$.
6. **End For**
7. **End For**

For our analysis, define, for every time t, the K-dimensional vector

$$\hat{\mathbf{c}}^t = \frac{c_{a_t}^t}{p_{a_t}^t} \mathbf{e}_{a_t}.$$

Note that the a-th component of $\hat{\mathbf{c}}^t$ is given by

$$\hat{c}_a^t = \begin{cases} \frac{c_a^t}{p_a^t} & \text{if } a_t = a, \text{ i.e. arm } a \text{ is chosen at time } t, \\ 0 & \text{otherwise.} \end{cases}$$

We can then rewrite the update step in the algorithm (Step 5) as

$$w_a^t = e^{-\epsilon \hat{c}_a^t} w_a^{t-1} \quad \forall\, a \in [K].$$

Note also that

$$\begin{aligned} \mathbb{E}_{a\sim \mathbf{p}^t}\left[\hat{\mathbf{c}}^t\right] &= \sum_{a\in[K]} p_a^t \cdot \left(\frac{c_a^t}{p_a^t}\mathbf{e}_a\right) \\ &= \sum_{a\in[K]} c_a^t \mathbf{e}_a \\ &= \mathbf{c}^t, \end{aligned}$$

and so at any time t, $\hat{\mathbf{c}}^t$ is an unbiased estimator for the true cost vector $\mathbf{c}^t$.

Now define

$$L_a^t := \sum_{\tau=1}^{t} \hat{c}_a^\tau,$$

so that

$$w_a^t = e^{-\epsilon L_a^t}.$$

Define a potential function Φ_t by

$$\Phi_t := -\frac{1}{\epsilon} \log \sum_{a\in[K]} w_a^{t-1}.$$

We have

$$\begin{aligned} \Phi_{t+1} - \Phi_t &= -\frac{1}{\epsilon} \log \frac{w_a^t}{w_a^{t-1}} \\ &= -\frac{1}{\epsilon} \log \frac{w_a^{t-1} \cdot e^{-\epsilon \hat{c}_a^t}}{w_a^{t-1}} \\ &= -\frac{1}{\epsilon} \log \mathbb{E}_{a\sim \mathbf{p}^t}\left[e^{-\epsilon \hat{c}_a^t}\right]. \end{aligned}$$

Then, because $e^{-x} \le 1 - x + \frac{1}{2}x^2$, we have

$$\begin{aligned} \Phi_{t+1} - \Phi_t &= -\frac{1}{\epsilon} \log \mathbb{E}_{a\sim \mathbf{p}^t}\left[e^{-\epsilon \hat{c}_a^t}\right] \\ &\ge -\frac{1}{\epsilon} \mathbb{E}_{a\sim \mathbf{p}^t}\left[1 - \epsilon \hat{c}_a^t + \frac{1}{2}\epsilon^2 (\hat{c}_a^t)^2\right] \\ &= -\frac{1}{\epsilon} \log\left(1 - \mathbb{E}_{a\sim \mathbf{p}^t}\left[\epsilon \hat{c}_a^t - \frac{1}{2}\epsilon^2 (\hat{c}_a^t)^2\right]\right), \end{aligned}$$

and since $-x \geq \log(1-x)$ we have

$$\begin{aligned}
\Phi_{t+1} - \Phi_t &\leq -\frac{1}{\epsilon}\log\left(1 - \mathbb{E}_{a\sim\mathbf{p}^t}\left[\epsilon\hat{c}_a^t - \frac{1}{2}\epsilon^2(\hat{c}_a^t)^2\right]\right) \\
&\geq \frac{1}{\epsilon}\mathbb{E}_{a\sim\mathbf{p}^t}\left[\epsilon\hat{c}_a^t - \frac{1}{2}\epsilon^2(\hat{c}_a^t)^2\right] \\
&= \mathbb{E}_{a\sim\mathbf{p}^t}\left[\hat{c}_a^t\right] - \frac{1}{2}\epsilon\mathbb{E}_{a\sim\mathbf{p}^t}\left[(\hat{c}_a^t)^2\right] \\
&= \sum_{a\in[K]} p_a^t\hat{c}_a^t - \frac{1}{2}\epsilon\sum_{a\in[K]} p_a^t(\hat{c}_a^t)^2 \\
&= \sum_{a\in[K]} p_a^t\hat{c}_a^t - \frac{1}{2}\epsilon\sum_{a\in[K]} c_a^t\hat{c}_a^t.
\end{aligned}$$

Taking expectation, we have

$$\mathbb{E}[\Phi_{t+1} - \Phi_t] \geq \sum_{a\in[K]} p_a^t c_a^t - \frac{1}{2}\epsilon\sum_{a\in[K]}(c_a^t)^2 \geq \sum_{a\in[K]} p_a^t c_a^t - \frac{K\epsilon}{2}.$$

Then, taking the telescopic sum, we have

$$\Phi_{T+1} - \Phi_1 \geq \sum_{t=1}^{T}\sum_{a\in[K]} p_a^t c_a^t - \frac{KT\epsilon}{2}. \tag{12}$$

On the other hand, we have $\Phi_1 = -\frac{1}{\epsilon}\log K$, so, where a^* is the optimal arm, we have

$$\mathbb{E}[\Phi_{T+1} - \Phi_1] \leq \mathbb{E}[L_{a^*}^T - (-\frac{1}{\epsilon}\log K)] = \sum_{t=1}^{T} c_{a^*}^t + \frac{1}{\epsilon}\log K. \tag{13}$$

Combining (12) and (13), we then have that

$$\mathbb{E}[R(T)] = \sum_{t=1}^{T}\sum_{a\in[K]} p_a^t c_a^t - \sum_{t=1}^{T} c_{a^*}^t \leq \frac{KT\epsilon}{2} + \frac{1}{\epsilon}\log K.$$

We can then choose $\epsilon = \sqrt{\frac{2\log K}{TK}}$, from which it would follow that the expected regret $\mathbb{E}[R(T)]$ is $O(\sqrt{TK\log K})$.

Theorem 3.1 *The Exp3 algorithm with $\epsilon = \sqrt{\frac{2\log K}{TK}}$ achieves an expected regret of*

$$O(\sqrt{TK\log K}).$$

4 Conclusion

In these two lectures we were introduced to framework of multi-armed bandits, both the stochastic setting, where each arm gives a reward according to a fixed distribution, and the adversarial setting,

where each arm at each time comes with a different cost, determined by an adversary. In both settings, the aim was to design an algorithm that minimizes the expected regret.

For the stochastic setting, we analyzed the Explore First algorithm, the Epsilon Greedy algorithm, the UCB Elimination algorithm, and the UCB algorithm. For large times T, it was found that the UCB elimination algorithm and the UCB algorithm had a favorable bound of $O(\sqrt{KT \log T})$ on the expected regret.

For the adversarial setting, the Exp3 algorithm was analyzed. The expected regret was found to be $O(\sqrt{TK \log K})$.

CS295 Optimization for Machine Learning

Instructor: Ioannis Panageas Scribed by: Yingxin Cao, Dahai Hao

Lecture 14. Introduction to Markov Decision Processes and RL.

1 Introduction

In this lecture we introduce the formal problem of finite Markov decision processes, or finite MDP's, which is a basic framework for reinforcement learning. We introduce key elements of the problem's mathematical structure, such as value functions, state-action value functions and Bellman equations. In the end, we introduce two iterative methods, policy iteration and value iteration that can be used to compute optimal policies given a perfect model of the environment as a MDP.

2 Markov Decision Process

2.1 The Framework

A finite Markov Decision Process (MDP) is defined as follows: A finite state space S; a finite action space A; a transition model P where $P(s'|s,a)$ is a matrix of size $(S \cdot A) \times S$, representing the probability of transitioning into state s' upon taking action a in state s; a reward function $r : S \times A \to [0,1]$; and a discounted factor $\gamma \in [0,1)$ bounding the value function.

The goal for the reinforcement learning is to find a stationary policy $\pi : S \to A$ such that the value function is maximized.

$$V^{\pi}(S) = (1-\gamma)\mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)|\pi, s_0 = s\right] \tag{1}$$

Above equation shows a infinite time horizon case. In this case, the sum of rewards is bounded with the discounted factor, which is a geometric series. In the finite case, one can omit the discounted factor.

Example 2.1 (Navigation) *Suppose you are given a grid map. The state of the agent is their current location. The four actions might be moving 1 step along each of east, west, north or south. The transitions in the simplest setting are deterministic. There is a goal g that is trying to reach. Reward is one if the agent reaches the goal and zero otherwise.*

The optimal behavior π in this setting corresponds to finding the shortest path from the initial to the goal state.

The value function of a state s, given the aforementioned policy is

$$V^{\pi}(s) = (1-\gamma)\gamma^d \tag{2}$$

where d is the number of steps to reach the goal from s.

3 State-action value function

Definition 3.1 (Q-function) *In discounted infinite horizon problems, for any policy π, the state-action value function $Q : S \times A \to \mathbb{R}$ is given by*

$$Q^{\pi}(s,a) = (1-\gamma)\mathbb{E}\left[\sum_{t=0}^{\infty}\gamma^{t} r(s_t,a_t)|s_0 = s, a_0 = a \text{ and } \pi(s_\tau)\ \forall\tau\right] \tag{3}$$

Q-function gives you the value you can get if you choose a particular action on state s. Value function is the maximizer of Q-functions for all the actions.

By definition, the following equations must be satisfied:

$$V^{\pi}(s) = Q^{\pi}(s,\pi(s)) \tag{4}$$

$$Q^{\pi}(s,a) = (1-\gamma)r(s,a) + \gamma\mathbb{E}_{s'\sim P(.|s,a)}\left[V^{\pi}(s')\right] \tag{5}$$

Equation 4 defines the recursive update of value function and Q-function. By plugging in the transition matrix we have

$$Q^{\pi}(s,a) = (1-\gamma)r(s,a) + \gamma\sum_{s'} P(s'|s,a)V^{\pi}(s') \tag{6}$$

This formula co-relates Q-function, value function and transition matrix P. The proof of equation 5 is shown below.

$$\begin{aligned} Q^{\pi}(s,a) &= (1-\gamma)r(s,a) + \sum_{s'} P(s'|s_0 = s, a_0 = a)\mathbb{E}\left[\sum_{t=1}^{\infty}\gamma^{t} r(s_t,a_t)|s_1 = s',\pi\right] \\ &= (1-\gamma)r(s,a) + \gamma\sum_{s'} P(s'|s_0 = s, a_0 = a)\mathbb{E}\left[\sum_{t=0}^{\infty}\gamma^{t} r(s_{t+1},a_{t+1})|s_1 = s',\pi\right] \\ &= (1-\gamma)r(s,a) + \gamma\sum_{s'} P(s'|s,a)V^{\pi}(s') \end{aligned}$$

Similarly, we can have the following for a value function

$$V^{\pi}(s) = (1-\gamma)r(s,\pi(s)) + \gamma\sum_{s'} P(s'|s,\pi(s))V^{\pi}(s') \tag{7}$$

If you have your values written as a vector, the equation 7 can be then rewrite into the following form,

$$Q^{\pi} = (1-\gamma)r + \gamma P V^{\pi} \tag{8}$$

with expectation written as the product of transition matrix P and value vector v^{π}. Then, by substitution,

$$Q^{\pi} = (1-\gamma)r + \gamma P^{\pi}Q^{\pi} \tag{9}$$

where p^{π} is a $(S\cdot A)\times(S\cdot A)$ that is induced by the P and policy π. Though this, we then have a close form formula of Q^{π},

$$Q^{\pi} = (1-\gamma)(I-\gamma P^{\pi})^{-1}r \tag{10}$$

where $(I - \gamma P^{\pi})^{-1}$ is invertible.

One can prove $(I - \gamma P^{\pi})^{-1}$ is invertible by showing $(\gamma P^{\pi})^{-1}$ has eigenvalues with absolute values equal or less than 1. This is true since P is a stochastic transition matrix with positive entries and each row sums up to one.

4 Bellman Equations

We would like to find the optimal stationary policy, that is we want to

$$V^*(s) = \max_{\pi} V^{\pi}(s) \tag{11}$$

Lemma 4.1 (Bellman Equations) *The following must hold:*

$$V^*(s) = \max_{a \in A} \left\{ (1 - \gamma r(s,a) + \gamma \sum_{s'} P(s'|s, \pi(s)) V^*(s') \right\}. \tag{12}$$

Equivalently for Q-function

$$Q^*(s,a) = (1-\gamma) r(s,a) + \gamma \sum_{s'} P(s'|s,\pi(s)) \max_{b \in A} Q^*(s', b). \tag{13}$$

Proof:

$$\begin{aligned}
V^*(s) &= \max_{\pi} V^{\pi}(s) \\
&= \max_{\pi} (1-\gamma) \mathbb{E}\left[\sum_t \gamma^t r(s_t, a_t | \pi, s, a) \right] \\
&= \max_{a, \pi'} (1-\gamma) r(s,a) + \gamma \sum_{s'} P(s'|s,a) \mathbb{E}\left[\sum_t \gamma^t r(s_t, a_t | \pi', s') \right] \\
&= \max_{a, \pi'} (1-\gamma) r(s,a) + \gamma \sum_{s'} P(s'|s,a) V^{\pi'}(s') \\
&= \max_{a} \left\{ (1-\gamma) r(s,a) + \gamma \sum_{s'} P(s'|s,a) \max_{\pi'} V^{\pi'}(s') \right\} \\
&= \max_{a} \left\{ (1 - \gamma r(s,a) + \gamma \sum_{s'} P(s'|s,\pi(s)) V^*(s') \right\}
\end{aligned}$$

■

Definition 4.1 (Bellman Operator) *Let's define the following operator T:*

$$TW(x) = \max_{a \in A} (1-\gamma) r(s,a) + \gamma \sum_{s'} P(s'|s,a) W(s') \tag{14}$$

Claim 4.2 (Bellman Operator) *V^* is the unique fixed point of the operator.*

Proof: We have already shown it is a fixed point. We will show that T is contracting!(Banach Fixed point Theorem).

Consider f, f' and observe that

$$|\max_a f(a) - \max_{a'} f'(a')| \leq \max_a |f(a) - f'(a)| \tag{15}$$

Assume a maximizes $f(a)$ and moreover $f(a) \geq \max_{a'} f'(a')$ (w.l.o.g. due to symmetry). Then we get

$$f(a) - \max_{a'} f'(a') \leq f(a) - f'(a) \leq \max_b f(b) - f'(b) \tag{16}$$

Therefore

$$\begin{aligned}
\left\|TV - TV'\right\|_\infty &= (1-\gamma)\left[\max_a r(s,a) + \sum_{s'} P(s'|a,s)V(s') - \max_{a'} r(s,a') - \sum_{s'} P(s'|a',s)V'(s')\right]_\infty \\
&\leq (1-\gamma)\max_a \left[r(s,a) + \sum_{s'} P(s'|a,s)V - r(s,a) - \sum_{s'} P(s'|a,s)V'\right]_\infty \\
&= (1-\gamma)\max_a \left[\sum_{s'} P(s'|a,s)(V - V')\right]_\infty \\
&\leq (1-\gamma)\left\|V - V'\right\|_\infty \max_a \left[\sum_{s'} P(s'|a,s)\right] \\
&= (1-\gamma)\left\|V - V'\right\|_\infty
\end{aligned}$$

■

5 Value Iteration

The idea of value iteration algorithm is simple: starting from an initial vector V_0, apply the optimal Bellman operator iteratively so that given V_k at iteration k we compute $V_{k+1} = TV_k$.

Claim 5.1 (Convergence of Value Iteration Algorithm) *After $k = \frac{\log(1/\epsilon)}{\log(1/\gamma)}$ iterations, we have error less than ϵ.*

Proof: Assume V^* is the optimal value then its a fixed point of the optimal Bellman operator T such that $TV^* = V^*$. V_0 is the initial value such that $\|V^* - V_0\|_\infty \leq 1$. So, after k iterations

$$\left\|T^k V^* - T^k V_0\right\|_\infty \leq (1-\gamma)^k \left\|V^* - V_0\right\|_\infty$$

Then,

$$\|V^* - V_k\|_\infty \leq (1-\gamma)^k$$

Plugging in $k = \frac{\log(1/\epsilon)}{\log(1/\gamma)}$, we have

$$\|V^* - V_k\|_\infty \leq \epsilon$$

■

Note that after the algorithm converges, we can use $\pi^* = argmax_a(1-\gamma)r(s,a)+\gamma \sum_{s'} P(s'|s,a)V^*(s')$ to find the optimal policy.

The value iteration algorithm converges very fast in terms of number of iterations, but it's not that efficient during each iteration because in each iteration the algorithm need to explore the whole state and action space.

6 Policy Iteration

The idea of policy iteration is that in each iteration, given the current policy π_k, evaluate the current policy by computing the value vector V^{π_k}. Then do a greedy move by computing $\pi_{k+1}(x)$ as $\pi_{k+1}(x) = argmax_{a \in A} \left[r(x,a) + \gamma \sum_y p(y|x,a)V^{\pi_k}(y) \right]$ for each state x. The algorithm stops when $V^{\pi_k} = V^{\pi_{k+1}}$. Because a finite MDP has only a finite number of policies, this process must converge to an optimal policy and optimal value function in a finite number of iterations.

CS295 Optimization for Machine Learning

Instructor: Ioannis Panageas | Scribed by: Cheng Zhang

Lecture 17. Introduction to Multi-agent RL.

1 Introduction

Single-agent reinforcement learning (RL) systems are typically modelled as Markov Decision Processes (MDPs) as it provides the guarantee of existence of stationery deterministic optimal policies. However, extending the theory to involve multi-agent interactions is not trivial. Despite the notable empirical advancements, there is a lack of understanding about the theoretical convergence guarantees of the existing multi-agent reinforcement learning algorithms.

In this lecture, we discussed about the general concept of multi-agent *Markov Decision Process* and focused on *Markov Potential Games*.

2 Definitions and Theorems

The mathematical details in the definitions may slightly different from what is used in the lecture. I follow the expressions provided in [1]

Markov Decision Process(MDP). We consider a setting with n agents who repeatedly select actions in a shared Markov Decision Process (MDP). The goal of each agent is to maximize their respective value function. Formally, a MDP is defined as a tuple $\mathcal{G} = (\mathcal{S}, \mathcal{N}, \{\mathcal{A}_i, \mathcal{R}_i\}_{i\in\mathcal{N}}, P, \gamma, \rho)$, where

- $\mathcal{S}$ is a finite state space of size $S = |\mathcal{S}|$. We will write $\Delta(\mathcal{S})$ to denote the set of all probability distribution over the set $\mathcal{S}$
- $\mathcal{N} = 1, 2, ..., n$ is the set of the $n \geq 2$ agents in the game
- $\mathcal{A}_i$ is a finite action space for agent $i \in \mathcal{N}$ with generic element $a_i \in \mathcal{A}$. In the report, We also write $\mathcal{A} = \prod_{i\in\mathcal{N}} \mathcal{A}_i$ and $\mathcal{A}_{-i} = \prod_{j\neq i} \mathcal{A}_j$.
- $\mathcal{R}_i : \mathcal{S} \times \mathcal{A} \to [0, 1]$ is the individual reward function of agent $i \in \mathcal{N}$
- P is the transition probability function, for which $P(s'|s, a_1, ...a_n)$ is the probability of transitioning into state s' upon agent i taking action a_i in state s.
- $\gamma \in [0, 1)$ us a discount factor fur future rewards of the MDP, shared by all agents
- $\rho \in \Delta(\mathcal{S})$ is the distribution for the initial state at time $t = 0$

Stochastic Policies For each agent $i \in \mathcal{N}$, a **stochastic**, stationary policy $\pi_i : \mathcal{S} \to \prod_i$ where $\prod_i := \Delta(\mathcal{A}_i)^{\mathcal{S}}$, specifies a probability distribution over the actions of agent i for each state $s \in \mathcal{S}$.

Value Functions The value function, V_s^i gives the expected reward of agent $i \in \mathcal{N}$ when initial state $s_0 = s$ and the agents draw their actions $\boldsymbol{a} = (a_i)_{i\in\mathcal{N}}$ at time $t \geq 0$ from policies $\pi = (\pi_i)_{i\in\mathcal{N}}$

$$V_s^i(\pi) := \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r_{i,t} | s_0 = s \right] \tag{17}$$

Nash Policy. A joint policy, $\pi^* = (\pi_i^*)_{i\in\mathcal{N}}$ is a Nash policy if for each agent i it holds that

$$V_s^i(\pi_i^*, \pi_{-i}^*) \geq V_s^i(\pi_i, \pi_{-i}^*) \tag{18}$$

for all $\pi_i \in \Delta(\mathcal{A}_i)$ and all $s \in \mathcal{S}$.

Policy Gradient and Direct Parameterization We assume that all agents update their policies *independently* according to the *projected gradient ascent* (PGA) or *policy gradient* algorithm on their policies. Independence here refers to the fact that PGA only requires local information to form the updates, i.e., to estimate that agent's policy gradients. The PGA algorithms is given by

$$\pi_i^{(t+1)} := P_{\Delta(\mathcal{A}_i)^{\mathcal{S}}} \left(\pi_i^{(t)} + \eta \nabla_{\pi_i} V_\rho^i(\pi^{(t)}) \right) \tag{19}$$

for each agent $i \in \mathcal{N}$, where $P_{\Delta(\mathcal{A}_i)^{\mathcal{S}}}$ is the projection onto $Delta(\mathcal{A}_i)^{\mathcal{S}}$ in the Euclidean norm. Here the additional argument $t \geq 0$ denotes time. We also assume that all players $i \in \mathcal{N}$ use direct policy parameterizations, i,e.,

$$\pi_i(a|s) = x_{i,s,a} \tag{20}$$

with $x_{i,s,a} \geq 0$ for all $s \in \mathcal{S}$, $a \in \mathcal{A}_i$ and $\sum_{a\in\mathcal{A}_i} x_{i,s,a} = 1$ for all $s \in \mathcal{S}$.

Theorem 2.1 *It can be shown for* ***one*** *agent that after $\mathcal{O}(1/\epsilon^2)$ iterations an ϵ-optimal policy can be reached. With some modifications, it works for two-player zero-sum games too[2].*

This theorem cannot be extended to the general settings with more than two players.

Markov Potential Game. A Markov Decision Process (MDP), $\mathcal{G}$, is called a *Markov Potential Game* (MPG) if there exists a (state-dependent) function $\phi_s : \prod \to \mathbb{R}$ for $s \in \mathcal{S}$ so that

$$\phi_s(\pi_i, \pi_{-i}) - \phi_s(\pi_i', \pi_{-i}) = V_s^i(\pi_i, \pi_{-i}) - V_s^i(\pi_i', \pi_{-i}) \tag{21}$$

for all agents $i \in \mathcal{N}$, all states $s \in \mathcal{S}$ and all policies $\pi_i, \pi_i' \in \prod, \pi_{-i} \in \prod_{-i}$.

Theorem 2.2 (Deterministic Optimal Policy Profile) *Let $\mathcal{G}$ be a Markov Potential Game (MPG). Then, there exists a Nash policy $\pi^* \in \Delta(\mathcal{A})^{\mathcal{S}}$ which is deterministic, i.e., for each agent $i \in \mathcal{N}$ and each state $s \in \mathcal{S}$, there exists an action $a_i in \mathcal{A}_i$ so that $\pi_i^*(a_i|s) = 1$*

The theorem 2.1 can be proved by exploiting the fact that we can iteratively reduce the non-deterministic components of an arbitrary Nash policy profile that corresponds to a global maximizer of the potential and still retain the Nash profile property at all times. At each iteration, we isolate an agent $i \in \mathcal{N}$, and find a deterministic (optimal) policy for that agent in the (single-agent) MDP in which the policies of all other agents but i remain fixed. The important observation is that the resulting profile is again a global maximizer of the potential and hence, a Nash policy profile. The detailed proof can be found in the appendix of [1].

Theorem 2.3 (PGA for Markov Potential Games) *Suppose all agents run policy gradient (PGA) iteration independently and update simultaneously. It can be shown that after* $\mathbb{O}(1/\epsilon^2)$ *iterations, an* ϵ*-Nash policy can be reached.*

Proof: Check Section 4 in [1] ■

References

[1] Stefanos Leonardos, Will Overman, Ioannis Panageas, Georgios Piliouras *Global Convergence of Multi-Agent Policy Gradient in Markov Potential Games.* arxiv.

[2] Yulai Zhao, Yuandong Tian, Jason D. Lee, and Simon S. Du. *Provably efficient policy gradient methods for two-player zero-sum markov games.* CoRR, abs/2102.08903, 2021.

CS295 Optimization for Machine Learning

Instructor: Ioannis Panageas Scribed by: Yanzhao Zou, Lu Xu

Lecture 15-16. Introduction to Statistical Learning Theory.

1 Introduction

Perceptron is a linear classifier or binary classifier, which is widely used in supervised learning to classify the given input data. The simplest perceptron is a single layer neural network, while multi-layers of perceptron are referred as neural network. Formally, the perceptron is defined as :

$$y = sign(\omega^T x - \theta), \tag{1}$$

where ω is the weight vector and θ is the threshold. And the goal is to compute a vector w that separates the two classes.

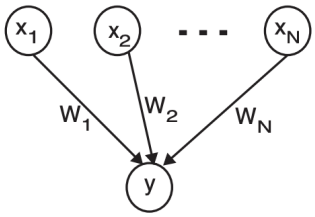


Figure 1: A simple perceptron.

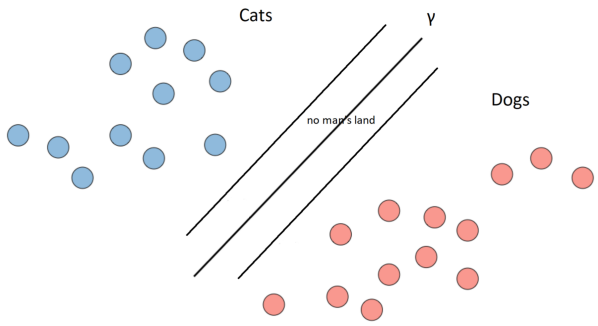


Figure 2: An example of Dogs and Cats classification.

1.1 The Perceptron Algorithm

Given $(x_1, y_1), ..., (x_T, y_T) \in X \times \{\pm 1\}$ where we assume $\| x \| = 1$ for all t. Formally γ is defined:

$$\gamma := max_{\omega:\|\omega\|=1} min_{i\in[T]}(y_i \omega^T x_i)_+, \tag{2}$$

where $(a)_+ = max(a, 0)$.

Consider the following iterative algorithm, where the goal is to iteratively update ω and optimize γ:

1. Initialize $\omega = 0$(hypothesis)
2. On round $t = 1...T$:

   Consider (x_t, y_t) and form prediction $\hat{y}_t = sign(\omega_t^T x_t)$.
   If $\hat{y}_t \neq y_t$:
   $\omega_{t+1} = \omega_t + y_t x_t$
   Else $\omega_{t+1} = \omega_t$.

1.2 Analysis of Perceptron

Theorem 1.1 *Perceptron makes at most $1/\gamma^2$ mistakes and corrections on any sequence with margin γ.*

Proof: Let m be the number of mistakes after T iterations. If a mistake is made at round t then

$$\| \omega_{t+1} \|_2^2 = \| \omega_t + y_t x_t \|_2^2 \tag{3}$$
$$\| \omega_{t+1} \|_2^2 = \| \omega_t \|_2^2 + \| x_t \|_2^2 + 2\, y_t\, x_t^T \omega_t (negative) \tag{4}$$
$$\| \omega_{t+1} \|_2^2 \leq \| \omega_t \|_2^2 + 1 \tag{5}$$

Since the update is only performed when there is a mistake, the total number of updates is equal to the number of mistakes made till step T, which is m in this case. When you sum equation 5 from 0 to m and cancel the same terms, we can get the below formula:

$$\| \omega_t \|_2^2 \leq m, \tag{6}$$

Consider a vector ω^* with margin γ, by definition of γ for all t that there is a mistake:

$$\gamma \leq y_t w^{*T} x_t = w^{*T}(w_{t+1} - w_t) \tag{7}$$

γ by definition is the the max min of $y_t w^{*T} x_t$, thus the $\leq$ relation holds. While by manipulating the iterative update step 2, we establish $y_t w^{*T} x_t = w^{*T}(w_{t+1} - w_t)$.
By adding equation 7 from 0 to m we also have that:

$$m\gamma \leq w^{*T}(w_T - w_1) = w^{*T} w_T, \tag{8}$$
$$= \| w_T \|_2 . \tag{9}$$
$$Therefore \quad m\gamma \leq \| w_T \|_2 \leq \sqrt{m} \tag{10}$$
$$Therefore \quad m \leq \frac{1}{\gamma^2}. \tag{11}$$

■

2 Random Data and 0-1 Loss Function

What we really showed is that given $(x_1, y_1), ..., (x_T, y_T) \in X \times \{\pm 1\}$, where we assume $\| x \| = 1$ for all t it holds:

$$\sum_{t=1}^{T} 1_{y_t \omega_t^T x_t \leq 0} \leq \frac{1}{\gamma^2} \tag{12}$$

Given $(x_1, y_1), ..., (x_T, y_T) \in X \times \{\pm 1\}$ IID from some distribution P. Run perceptron algorithm and consider $\omega_1, ...\omega_n$. Then choose ω.

Theorem 2.1 *IID Data: Let ω be the choice of the algorithm. It hold that:*

$$\mathbb{E}[\frac{1}{n} \sum_{i=1}^{n} 1_{y_i \omega^T x_i \leq 0}] \leq \frac{1}{n} \mathbb{E}[\frac{1}{\gamma^2}] \tag{13}$$

Proof: We have proved from before that (and taking expectation)

$$\mathbb{E}[\frac{1}{n} \sum_{i=1}^{n} 1_{y_i \omega^T x_i \leq 0}] \leq \mathbb{E}[\frac{1}{n\gamma^2}] \tag{14}$$

let $S = ((x_1, y_1), ..., (x_n, y_n))$. The LHS can be expressed as:

$$\mathbb{E}_\tau \, \mathbb{E}_S[1_{y_\tau \omega_\tau^T x_\tau \leq 0}] = \mathbb{E}_S \, \mathbb{E}_\tau[1_{y_\tau \omega_\tau^T x_\tau \leq 0}] \tag{15}$$

Observe now that ω_τ depdends only on $(x_1, y_1), ..., (x_{\tau-1}, y_{\tau-1})$, hence finally we can express the LHS in the form of a 0-1 loss function:

$$\mathbb{E}_S \, \mathbb{E}_\tau[1_{y_\tau \omega_\tau^T x_\tau \leq 0}] = \mathbb{E}_S \, \mathbb{E}_\tau \, \mathbb{E}_{(x,y)\sim P}[1_{y \omega_\tau^T x \leq 0}] = \mathbb{E}_S \, \mathbb{E}_\tau[L_{0-1}(\omega_\tau)] \tag{16}$$

where:

$$L_{0-1}(\omega) = \frac{1}{n} \sum_i 1_{y_i \omega^T x_i \leq 0}. \tag{17}$$

Note that if we keep iterating perceptron algorithm we finally get $L_{0-1}(\omega_\tau) = 0$, providing the two classes are linearly separable. ■

3 PAC Learning

Now that we have understood the definition of the algorithm and how it generalises to random data, let us talk about how we can evaluate the performance of the algorithm.
Assume we are given:

- Domain set $\mathcal{X}$, typically $\mathbb{R}^d$ or $\{0,1\}^d$. Think of 32x32 pixel images.
- Label set $\mathcal{Y}$, typically binary like $\{0,1\}$ or $\{-1,+1\}$
- A concept class $\mathcal{C} = \{h : h : \mathcal{X} \to \mathcal{Y}\}$

Given a learning problem, we analyse the performance of a learning algorithm:

- Training data $\mathcal{S} = (x_1, y_1), ..., (x_m, y_m)$, where samples $\mathcal{S}$ was generated by drawing m IID samples from the distribution $\mathcal{D}$.
- Output a hypothesis from a hypothesis class $\mathcal{H} = \{h : h : \mathcal{X} \to \mathcal{Y}\}$ of target functions.

We measure the performance through generalization error that is

$$err(h) = \mathbb{E}_{(x,y)\sim D}[l_{0-1}(h(x), y)]. \tag{18}$$

Definition 3.1 *(PAC learnable). We call a concept class $\mathcal{C}$ of target function is PAC learnable (w.r.t to $\mathcal{H}$) if there exists an algorithm A and function $m_C^A : (0,1)^2 \to \mathbb{N}$ with the following property:*

Assume $\mathcal{S} = ((x_1, y_1), ..., (x_m, y_m))$ is a smaple of IID examples generated by some arbitrary distribution D such that $y_i = h(x_i)$ for some $h \in \mathcal{C}$ almost surely. If $\mathcal{S}$ is the input of A and $m > m_C^A$ then the algorithm returns a hypothesis $h_s \in \mathcal{H}$ such that, with probability $1 - \delta$ (over the choice of the m training examples):

$$err(h_s) < \epsilon \tag{19}$$

The function m_C^A is referred to as the sample complexity of algorithm A.

To help us understand the definition of concept class, here we list two concrete examples:
Example: (Axis Aligned Rectangles). The first example of a hypothesis class will be of rectangles aligned to the axis. Here we take the domain $\mathcal{X} = \mathbb{R}^2$ and we let $\mathcal{C}$ include be defined by all rectangles that are aligned to the axis. Namely for every (z_1, z_2, z_3, z_4) consider the following function over the pane

$$f_{z_1,z_2,z_3,z_4}(x_1, x_2) = \begin{cases} 1 & z_1 \le x_1 \le z_2, z_3 \le x_2 \le z_4 \\ 0 & else \end{cases} \tag{20}$$

Then $\mathcal{C} = \{f_{z_1,z_2,z_3,z_4} : (z_1, z_2, z_3, z_4) \in \mathbb{R}^4\}$.

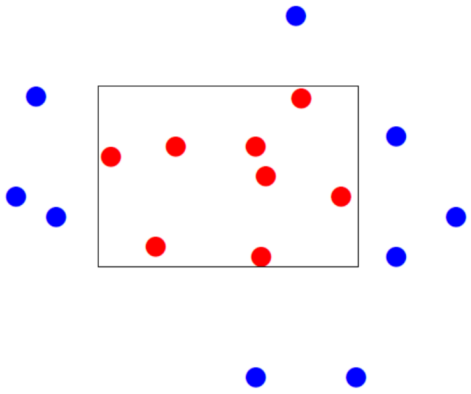


Figure 3: Concept Class of Axis Aligned Rectangles.

Example: (Half-space). A second example that is of some importance is defined by hyperplane. Here we let the domain be $\mathcal{X} = \mathbb{R}^d$ for some integer d. For every $w \in \mathbb{R}^d$, induces a half space by consider all elements $\mathcal{X}$ such that $w \cdot x \geq 0$. Thus, we may consider the class of target functions described as follows:

$$\mathcal{C} = \{f_w : w \in \mathbb{R}^d, f_w(x) = sign(w \cdot x)\} \tag{21}$$

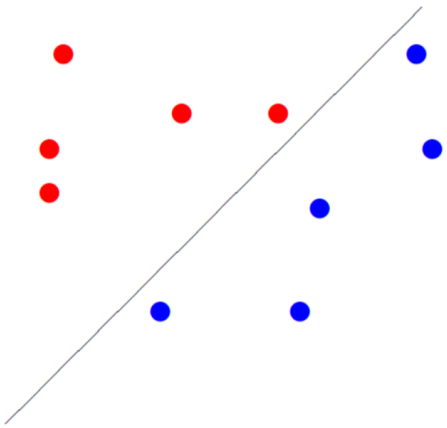


Figure 4: Concept Class of Half-Space.

4 ERM Algorithm

Now even if a concept class is PAC learnable, there might exist multiple hypothesis classes that meet the requirement. For the interest of optimization, our real focus is to find the optimal hypothesis class that gives us the minimum error given all the conditions. And that is where we Empirical Risk

Minimization (ERM) algorithm comes into play. ERM algorithm is defined as follows: Return:

$$arg\ min_{h\in\mathcal{H}}\ err_s(h), \tag{22}$$

where $err_s(h) = \frac{1}{m}\sum l_{0-1}(h(x_i), y_i)$.

Luckily, we have some nice guarantees when the concept class is finite.

Theorem 4.1 *Finite classes are PAC learnable: Consider a finite class of target function $\mathcal{H} = h_1, ..., h_t$ over a domain. Then if size of sample S is $m > \frac{2}{\epsilon^2}\log\frac{2|\mathcal{H}|}{\delta}$ then with probability $1-\delta$ we have that:*

$$max_{h\in\mathcal{H}} \mid err_S(h) - err(h) \mid < \epsilon. \tag{23}$$

Proof: Applying Hoeffding's inequality we obtain that for every S and fixed h, since $err_S(h)$ is sum of IID bernoulli with expectation $err(h)$:

$$Pr_S[\mid err_S(h) - err(h) \mid > \epsilon] \leq 2e^{-2m\epsilon^2} \tag{24}$$

Applying union bound we obtain that:

$$Pr_S[\exists h :\mid err_S(h) - err(h) \mid > \epsilon] \leq 2 \mid \mathcal{H} \mid e^{-2m\epsilon^2} \tag{25}$$

We want the RHS to be less than δ. We can achieve that With the appropriate choice of m. ■

5 VC Dimension

Now that we have see the neat result guarantee when the concept class is finite, what happens when the concept class is infinite? Does the guarantee still hold true? Let's first take a look at a motivating example.

Lemma 5.1 *Threshold Function: Consider the Hypothesis class of threshold function on the real line, that is:*

$$\mathcal{H} = h_a : a \in \mathbb{R}, \tag{26}$$

where $h_a(x) = 1_{x<a}$.$\mathcal{H}$ is PAC learnable using ERM algorithm (even if the class is infinite).

Remarks:

- Therefore, it is not necessary that the hypothesis class is of finite cardinality.
- We will show the lemma above , i.e., (ϵ, δ) - learnable using $\frac{\log\frac{2}{\delta}}{\epsilon}$ samples.

Proof: Let D be the marginal distribution over the domain and fix ϵ, δ. We need to show that taking S samples IID of size $\frac{\log\frac{2}{\delta}}{\epsilon}$ suffices so that with probability ($1-\delta$ the generalization error is at most ϵ.
Let a^* be a number such that h_{a^*} has error zero (perfect fit). Moreover, consider $a_0 < a^* < a_1$ such that:

$$Pr_{x\sim D}[x \in (a_0, a^*)] = Pr_{x\sim D}[x \in (a^*, a_1)] = \epsilon. \tag{27}$$

Observe that we might have to choose $a_0 = -\infty$ or $a_1 = +\infty$.

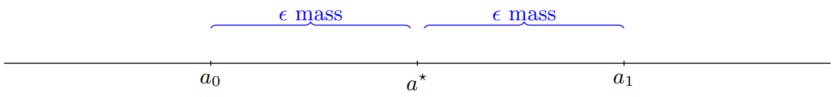


Figure 5: Concept Class of Axis Aligned Rectangles.

Let S be a set of IID samples and assume that the ERM algorithm returns a function h_S with threshold b_S.
If b_0 is the maximum x with label 1 and b_1 the minimum x with label 0 it holds that $b_S \in (b_0, b_1]$.
The error of $h_S$is at most ϵ if and only if $(b_0, b_1] \subseteq (a_0, a_1)$.
Let 's bound the probability of this event. By union bound we have:

$$Pr_{S\sim D^m}[(b_0 < a_0) \cup (b_1 > a_1)] \leq Pr_{S\sim D^m}[(b_0 < a_0)] + Pr_{S\sim D^m}[(b_1 > a_1)] \tag{28}$$

$$Pr_{S\sim D^m}[(b_0 < a_0)] \leq Pr_S[\forall x \in S, x \notin (a_0, a^*)] = (1-\epsilon)^m \leq \epsilon^{-\epsilon m} \tag{29}$$

$$Pr_{S\sim D^m}[(b_1 > a_1)] \leq Pr_S[\forall x \in S, x \notin (a^*, a_1)] = (1-\epsilon)^m \leq \epsilon^{-\epsilon m} \tag{30}$$

By adding equation 29 and 30, we conclude that the error probability is $2\epsilon^{-\epsilon m} = \delta$. Solving for m we get:

$$m = \frac{\log(\frac{2}{\delta})}{\epsilon}. \tag{31}$$

However, note that not all hypothesis classes are learnable. With the help of VC dimension in the next section, we can get more defined conditions for learnable and unlearnable cases.

■

5.1 Definition

Definition 5.1 *(Restriction). Let $\mathcal{H}$ be a class of functions from $\mathcal{X}$ to {0,1} and let $C = c_1, ...c_m$. The restriction of $\mathcal{H}$ to C is the set of functions from C to {0,1} that can be derived from $\mathcal{H}$. That is*

$$\mathcal{H}_C = \{h(c_1), ..., h(c_m)) : h \in \mathcal{H}\}, \tag{32}$$

where we represent each function from C to {0,1} as a vector in $\{0,1\}^{|C|}$

Definition 5.2 *(Shattering). A hypothesis class $\mathcal{H}$ shatters a finite set $C \subset \mathcal{X}$ if the restriction of $\mathcal{H}$ to C is the set of all functions from C to $\{0,1\}$. That is $\mid \mathcal{H} \mid = 2^{|C|}$.*

Definition 5.3 *(VC dimension). The VC-dimension hypothesis class $\mathcal{H}$, denote VCdim($\mathcal{H}$),is the maximal size of a set C that can be shattered by $\mathcal{H}$. If $\mathcal{H}$ can shatter sets of arbitrarily large size we say that $\mathcal{H}$ has infinite VC-dimension.*

Example: Let's see some examples and their intuitions:

- The class of threshold functions on real lines has VC dimension 1.
  Ans: Any point that lies on the same side of the threshold cannot be labelled as 0 and 1 at the same time.

- The class of interval functions on real line has VC dimension of 2.
  Ans: Any point that lies between two given points with the same label, cannot take the other label value.

- The class of aligned rectangle functions on the plane has VC dimension 4.
  Ans: Refering to Figure 6, any axis aligned rectangle cannot label c_5 by 0 and the rest of the points by 1.

- Any infinite class $\mathcal{H}$ hs VC dimension of at most log $\mid \mathcal{H} \mid$.
  Ans: Because by definition of shattering, $\mid \mathcal{H} \mid = 2^{|C|}$

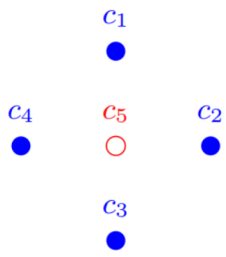


Figure 6: example of VC dimension of 4

5.2 VC Dimension of Halfspaces

Theorem 5.2 *(Halfspaces). The VC dimension of the class $\mathcal{H}$ of homogenous halfspaces in $\mathbb{R}^d$. Note that $\mathcal{H} = \{h_w(x) : h_w(x) := sign(w^T x)\}$.*

Proof: We first need to show that VC dimension is at least d by appropriately choosing a set C. Consider the set of vector $e_1, ..e_d$,where for every i the vector e_i is the all zeros vector except 1 in the i-th coordinate. This set is shattered by the class of homogenous halfspaces because for every binary vector $y_1, ...y_d$ and for $w = (y_1, ...y_d)$, we get that $h_w(e_i) = y_i$.
Next we need to show that VC dimension is less than d+1. Let $x_1, ...x_{d+1}$ be a set of d+1 vectors in $\mathbb{R}^d$.Then, there must exist real numbers $a_1, ..., a_{d+1}$, not all of them are zero, such that:

$$\sum a_i x_i = 0 \text{ linearly dependent.} \tag{33}$$

$$\text{Let } I = \{i : a_i > 0\} \text{ and } J = \{j : a_j < 0\} \tag{34}$$

If both I, J are non-empty then

$$\sum_{i \in I} a_i x_i = \sum_{j \in J} \mid a_j \mid x_j. \tag{35}$$

If $x_1, ..., x_{d+1}$ are shattered then there exists a ω such that $\omega^T x_i > 0$ for $i \in I$ and $\omega^T x_j < 0$ for $j \in J$. If the above is true, we get that:

$$0 < \sum_{i \in I} a_i \omega^T x_i = \omega^T \sum_{i \in I} a_i x_i \tag{36}$$

$$= \omega^T \sum_{j \in J} \mid a_j \mid x_j \tag{37}$$

$$= \sum_{j \in J} \mid a_j \mid \omega^T x_j < 0, \tag{38}$$

which is a contradiction. Thus we can conclude that VC dimension is less than d+1. And the theorem is thereafter proved. ■

5.3 Example of Infinite VC

As mentioned earlier before introducing the formal definition of VC dimension, it helps us define what hypothesis classes are not PAC learnable. Let's see an example of that.

Theorem 5.3 *(since has infinite VC). Consider the real line and let*

$$\mathcal{H} = \{x \to \lceil sin(\theta x) \rceil : \theta \in \mathbb{R}\}. \tag{39}$$

The VC dimension of the hypothesis class above is infinite.

Proof: We need to show that for for every d one can find d points that are shattered by $\mathcal{H}$. consider $x \in (0, 1)$ and let $0.x_1x_2x_3...$, be the binary expansion of x. Then for any natural number m, $\lceil sin(2^m \pi x) \rceil = 1 - x_m$, provided that there exists a $k \geq m$ such that $x_k = 1$.
Fix d and consider $C = \{1/2, 1/4, ..., 1/2^d\}$ and moreover choose any binary vector of labels $(y_1, ..., y_d)$. Set $x = 0.y_1...y_d1$ and use the above.
Intuitively, the sign function of sine function, is a square wave function with amplitude 1 and period given by $2\pi costan^{-1}(\theta)$. Thus by changing the value of θ, the square wave frequency can be manipulated to produce any labeling for a given set of points. Thus, its VC dimension is infinite. ■

5.4 The Importance of VC Dimension

Theorem 5.4 *Fundamental Theorem of Learnability: The following are equivalent:*

- $\mathcal{H}$ *is PAC learnable.*
- *Any ERM rule is a successful PAC learner for* $\mathcal{H}$.
- $\mathcal{H}$ *has finite VC dimension.*

Remarks: The number of samples needed is $O(\frac{d \log \frac{1}{\epsilon} + \log \frac{1}{\delta}}{\epsilon})$, where d is the VC dimension of the hypothesis class.