COMS 4995-2: Advanced Algorithms (Spring'20) March 12, 2020

Lecture 15: Ellipsoid Algorithm & Gradient Descent

Instructor: *Alex Andoni* Scribes: *Andrew Quijano, Isabel Ehrhardt*

1 Project

There will be 3 deliverables for the projects, which are two project reports (PR) and a final paper. The instructor will upload a document to Courseworks with more detailed information on both the expectations of the deliverables and potential project ideas.

1.1 Progress Report 1 - due 3/26

- 1-2 pages
- Include team members, 2-4 people (1 person projects discouraged, talk to Professor)
- Specify area(s) of interest
- Specify type of project (see nest sub-section)
- Include some relevant literature for professor to help guide interest/give suggestions
- Identify some potential problem/goal for your final report

1.2 Progress Report 2 - due 4/16

- 3 pages
- Expand on PR 1 by formally defining the problem, the goal of the project, and what related works you will be referencing
- Include current progress and remaining work

1.3 Final Report - due 5/11

- Minimum 10 pages
- A complete report on the project you set for your team in the first 2 PRs.

1.4 Types of Projects

1.4.1 Reading/Survey

Read 2 - 3 papers or a very long paper e. g. a thesis. The goal is to synthesize the data from the sources into one concise paper. For example, it can be reading about 2 approaches to same problem and or viewing both an old and more modern approach to a specific problem. The survey will discuss approaches and the main core ideas of the papers, and synthesize their ideas and how they are related. The final project report should be 10+ pages. Avoid re-writing the papers, you want to discuss the core ideas of the papers and be able to clearly explain it and compare/contrast both papers. The description should follow a lecture format that you would be able to explain it to your classmates.

1.4.2 Implementation

Implement some algorithm that is theoretically backed, such as the algorithms seen in class. It would have a theorem both proving the algorithm works and has specified run-time.
You should understand the core idea of the paper and run the algorithm on your a graph/dataset potentially. There should be some hypothesis you want to prove or disprove. An example would be you implement a standard algorithm and compare with a new algorithm and compare which runs faster. You should be able to explain your results e. g. an algorithm works better on certain kind of datasets than another algorithm.

1.4.3 Research-based

You want to try push bounds of what we have learned in class to create some new theoretical work. It is connected to the reading/survey project as you will need to understand the current state of the problem you are investigating in algorithms to devise a new algorithm. If you struggle to complete your goal in time, you should have a plan to fall back to a reading/survey project. You can also narrow the scope, such as improving an algorithm's run-time for a very specific use case (e. g. making assumptions on a potential dataset your new algorithm would be analyzing).

1.5 Last remarks

You are free to e-mail/ask the instructor on Piazza about the project before PR1 deadline about the direction of your project. It is best to ask the instructor directly rather than the TAs with your project inquiries. PR1 will be returned with the instructor's comments about your project and a list of updated related works that may better fit your interests.

2 Ellipsoid Algorithm - Khachiyan, 1979

2.1 Background

- Our 1st polynomial time algorithm for LP.
  (We have talked about the Simplex algorithm that was smarter in enumerating the vertices as it would get closer and closer to the solution. While no bound can be proved for the Simplex algorithm, in practice it runs in polynomial time assuming your LP is not adversarial, etc.)

- The Ellipsoid Algorithm isn't used practically for LP, but is more versatile for other problems beside LP.
- It solves the Feasibility Problem: which is defined as finding an x ∈ F, where F is the feasible region e. g. Ax ≥ b.

2.2 Reducing LP to Feasibility

Fact 1. *You can reduce general LP into solving feasibility.*

General Linear Program:

$$\begin{array}{ll} \text{minimize} & c^\top x \\ \text{subject to} & Ax \geq b \end{array}$$

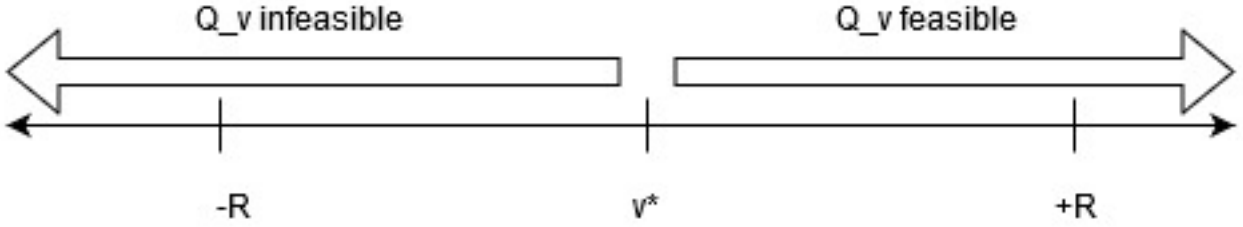
Proof. Suppose we have a guess v for LP. Then, we can solve the following feasibility problem for x:

$$Q_v = \begin{cases} Ax \geq b \\ c^\top x \leq v \end{cases}$$

If you don't have a guess, execute binary search to find v.
Start with v^0 = -R, where R is a really large number and is likely to be infeasible.
If Q_{v^0} is infeasible $\rightarrow v^1 = \frac{-R}{2}$. If Q_{v^1} is infeasible $\rightarrow v^2 = \frac{v^1}{2}$. If

$$v^* = \begin{cases} Ax \geq b \\ c^\top x \leq v \end{cases}$$

Then we just do a binary search for v^* starting from interval [-R, +R] for large enough R.



If A, b are all integers then $log_2(\text{R}) = n^{O(1)}$ * b, where b is the bit complexity of the numbers.

v^* = ratio of 2 numbers with bit-complexity b.

# of iterations in the binary search is $n^{0(1)}$ * b. □

2.3 Definitions

Definition 2. ***Ball*** *B, for r > 0, is defined as* $B_r(y) = \{ x: ||x - y|| \leq r\}$

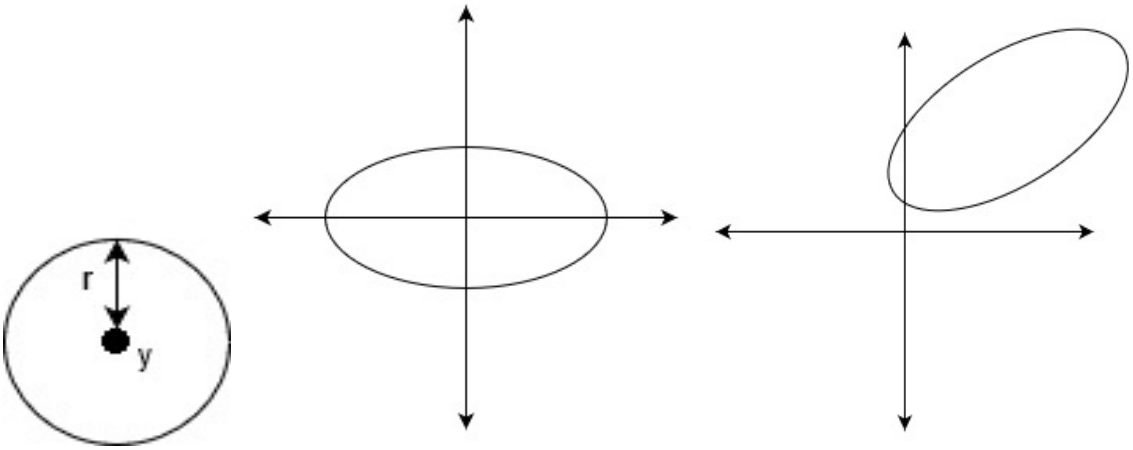
Definition 3. ***Axis-Aligned Ellipsoid*** *E for* $\lambda_1, \lambda_2, ..., \lambda_n > 0$ *is defined as* $E(y) = \{x : \frac{\Sigma_i (x_i - y_i)^2}{\lambda^2} \leq r\}$
Note: if all $\lambda = 1$*, this is a ball*

Definition 4. ***Generic Ellipsoid*** *E for full-rank n x n matrix A is defined as*

$$E(y) = \{x : (y - x)^T A^T A(y - x) \leq r^2\}$$

y is the center and the rotation is determined by A.
Note: if A is the diagonal matrix composed of $\frac{1}{\lambda_i}$*, then this is an axis aligned ellipsoid.*

A ball, an Axis-Aligned Ellipsoid, and a Generic Ellipsoid in 2D:



2.4 Ellipsoid Algorithm - Solving the Feasibility Problem

Feasibility problem:

$$F = \{x : Dx \geq e\} = \begin{cases} Ax \geq b \\ c^\top x \leq v \end{cases}$$

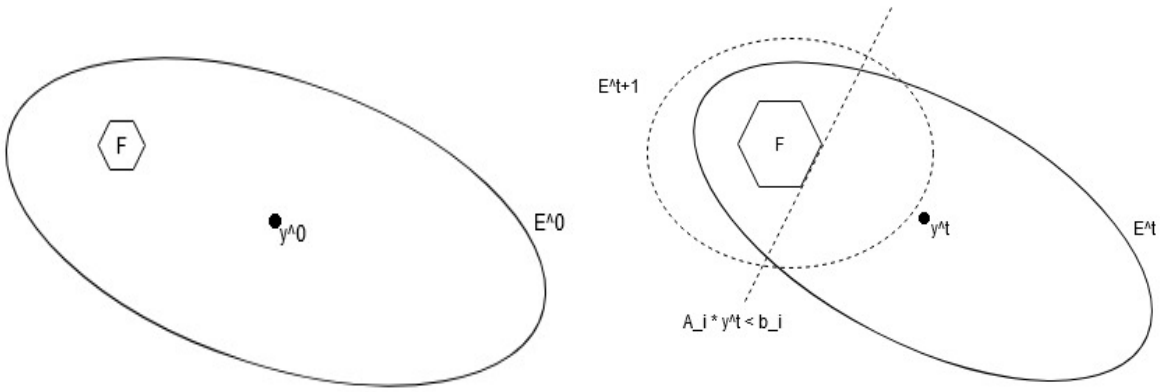
Goal: Find x ∈ F

Algorithm idea: We want to search for x ∈ F, where F is the feasible region, keeping F as a subset of some larger ellipsoid that we make tighter. We proceed in iterations, where in iteration t, y^t is center of ellipsoid E^t such that F ⊆ E^t, and try to make E^{t+1} smaller than E^t

Algorithm:

1: Start with $y_0 = 0$, $E_0 :=$ ball of radius r, such that if $x \in F$ exists, $||x|| \leq R$
2: **if** $y^t \in$ F **then**
3: Success!
4: **else**
5: y^t violated some inequality constraint $A_i x \geq b_i$ where $A_i y^t < b_i$
6: Compute new y^{t+1}, E^{t+1} s. t. F $\subseteq E^{t+1}$ and E^{t+1} is smaller than E
7: To compute this, use the fact that the constraint gives a hyperplane that cuts the ellipsoid, with y^t on one side, and F on the other
8: Find E^{t+1} that contains $E^t \cap \{x_i; Ax_i \geq b\}$
9: If not yet done, continue unless E^t is so small that we need to stop and say it is infeasible.
10: **end if**

2D Visuals:



Main Analysis Claim: There is a method to define E^{t+1} such that vol(E^{t+1}) $\leq$ vol(E^t)(1 - $\frac{1}{2n}$). (where vol is the traditional volume of ellipsoid).

Corollary: if v = vol(E^0) = exp(n log(n)) R^n. Then after T = O(nlog($\frac{v}{\epsilon}$)) iterations. Then val(E^t) $\leq$ vol(E^0)(1 - $\frac{1}{2n}$)^T $\leq \epsilon$

Remark: This algorithm is applicable to in programming/optimization problems beyond LP.
e. g.: LPs where number of constraints is exp(n) and where for given y, find a constraint Ax $\geq$ b that is violated by y in polynomial time with respect to n. This is done by taking the normal given y. it is polynomial because you aren't iterating overall constraints.

3 Gradient Descent (GD)

We so far talked about L. P. for a constrained optimization problem

$$\begin{array}{ll} \text{minimize} & c^\top x \\ \text{subject to} & Ax \geq b \end{array}$$

GD is unconstrained optimization for minimizing some function f(x). Solve

$$min_{x \in \mathbb{R}^n} f(x)$$

3.1 LP can be formatted as Unconstrained Optimization

Define

$$g(x) = \begin{cases} f(x) & x \in F,\ Ax \geq b \\ +\infty & x \notin F \end{cases}$$

Note: F is the feasible region. Now we have a new optimization problem:

$$min_{x \in \mathbb{R}^n} g(x)$$

Goal: Find $min_x\ f(x)$.

General Approach for GD:
Set x_0 as a starting point
Define $x^{t+1} = f(x^t$, values of f or x^t, ...)
Then hope that in each iteration of x^t to x^{t+1} that f(x^{t+1}) is improved over f(x^t).

Computational questions:

1. How to go from x^t to x^{t+1}?
2. How many iterations?

There are similar computational questions brought up when discussing max flow using the Ford-Fulkerson algorithm. The first point matches with selecting better augmenting paths for Ford-Fulkerson. Also, the second point matches with how picking a better alternating path reduces the number of iterations.
Gradient Descent as a natural greedy algorithm, means it moves a little bit $\delta = x^{t+1}$ - x^t such that to minimize f(x^{t+1})

3.2 What should δ be?

Suppose f is a "nice" function so that we can approximate f via a Taylor expansion in $\mathbb{R}^n$. Note that $\nabla f(x)^\top \delta \in \mathbb{R}^n$ and $\nabla^2 f(x)$ is the Hessian Matrix, which can be described as $[\nabla^2 f(x)]_{i,j} = \dfrac{\partial^2 f}{\partial x_i \partial x_j}$

$$f(x+\delta) \approx f(x) + \nabla f(x)^\top \delta + \delta^\top \nabla^2 f(x) \delta$$

Think of δ as very small, as if near infinitesimal small. Note $O(||\delta||^2)$ is negligible. Therefore:

$$f(x+\delta) \approx f(x) + \nabla f(x)^\top \delta + O(||\delta||^2)$$
$$f(x+\delta) \approx f(x) + \nabla f(x)^\top \delta$$

Question: find δ such that:

1. $||\delta|| \leq \epsilon$
2. minimize f(x) + $\nabla f(x)^\top \delta$

Note: You drop f(x) as it is not depending on δ.
Note: For the optimization, the best thing to do is to take δ to be equal to the gradient but the opposite sign!
Note: $\eta^2 ||\nabla f(x)||^2 = \epsilon^2$

$$\begin{aligned}\delta &= argmin_{\delta s.t. ||\delta|| \leq \epsilon} f(x) + \nabla f(x)^\top \delta \\ &= argmin_{\delta s.t. ||\delta|| \leq \epsilon} \nabla f(x)^\top \delta \\ &= -\nabla f(x)^\top \eta \; s.t. \; ||\delta||^2 = \epsilon^2\end{aligned}$$

Finally, we get the result:

$$x^{t+1} = x^t + \delta = x^t - \nabla f(x)^\top \eta$$