A Mini-Course on Convex Optimization

with a view toward designing fast algorithms

Nisheeth K. Vishnoi

Preface

These notes are largely based on lectures delivered by the author in the Fall of 2014. The first three lectures are meant to be an introduction to the core techniques in convex optimization which are increasingly being deployed to design fast algorithms for discrete problems. Many thanks to the scribes: Elisa Celis, Slobodan Mitrovic, Damian Straszak and Jakub Tarnawski. The material presented in these lectures is inspired from several excellent sources such as Boyd and Vanderberghe's *Convex optimization*, Nesterov's *Introductory lectures on convex optimization* and Arora *et al.*'s survey on the *Multiplicative weight update method.* Thanks to Sushant Sachdeva for many enlightening discussions about Interior point methods which have influenced the last part of these notes. Finally, these notes are quite preliminary and will be revised. Please feel free to email your comments and suggestions.

Nisheeth K. Vishnoi
7 Oct. 2015, EPFL
nisheeth.vishnoi@epfl.ch

Contents

1

Basics, Gradient Descent and Its Variants

1.1 Overview

Convex optimization is about minimizing a convex function over a convex set. For a convex set K, and a convex function f whose domain contains K, the goal is to solve the following problem:

$$\inf_{x \in K} f(x).$$

Convex optimization is a classical area with a long and rich history and diverse applications. Traditionally, a large fraction of algorithms in TCS have been obtained by formulating the underlying discrete problem (for example: Shortest Path; Maximum Flow; Matching, Graph Partitioning) as a linear (or a semi-definite) program and, consequently, deploying standard methods such as the ellipsoid method or the interior point methods from convex programming to solve these programs. However, with the growth in the sizes of the real-world instances, there has been a pressing need to design faster and faster algorithms, ideally in almost the same time as it takes to read the data. In the last decade or so TCS researchers are coming remarkably close to what would have been considered a pipe-dream a few years ago. And tools and techniques from convex optimization are playing a bigger and bigger role in this goal. Since general methods of solving convex programs have runtimes which are far from linear, often this goal requires adapting known methods from convex optimization and, more often than not, coming up with novel problem-specific solutions. Thus, techniques from convex optimization have become an indispensable tool in the toolkit of any algorithm designer and in these three lectures, we cover a large ground in this regard. The methods we present include gradient descent and its many variants, multiplicative weight update methods and their application to approximately solving linear programs and semi-definite programs quickly, Newton's method and its application to path-following interior point methods. Through these methods, we will see a very intimate connection between convex optimization and *online convex optimization*. With a good understanding of the material covered in these three lectures, a student should be well-equipped to understand many recent breakthrough works in TCS and, hopefully, push the state-of-the-art.

1.2 Basic Notions

1.2.1 Notation

We are concerned with functions $f : \mathbb{R}^n \mapsto \mathbb{R}$. By $x, y, \ldots$ we typically mean vectors in $\mathbb{R}^n$ and we use $x_1, x_2, \ldots$ to denote its coordinates. When the function is smooth enough, we can talk about its gradients and Hessians. The gradient is denoted by

$$\nabla f(x) \stackrel{\text{def}}{=} \left(\frac{\partial f}{\partial x_1}(x), \ldots, \frac{\partial f}{\partial x_n}(x) \right).$$

The Hessian of f at a point x is a matrix denoted by $\nabla^2 f(x)$ whose (i,j)-th entry is $\frac{\partial^2 f}{\partial x_i \partial x_j}(x)$. We say that a symmetric matrix $M \succeq lI$, if all its eigenvalues are at least l. When $l \geq 0$, the matrix is said to be positive semi-definite (psd). Similarly, we denote by $M \preceq LI$ the fact that all eigenvalues of M are at-most L. $\langle x, y \rangle$ denotes the inner product between x and y. $\|x\|$ denotes the ℓ_2 norm unless stated otherwise. An important but easy to prove result is the Cauchy-Schwarz inequality which states that

$$\langle x, y\rangle \leq \|x\| \cdot \|y\|.$$

An important tool from calculus is Taylor expansion of a function f at y around a point x

$$f(y) = f(x) + \langle \nabla f(x), y - x\rangle + \frac{1}{2}(y-x)^\top \nabla^2 f(x)(y-x) + \cdots$$

when the function is infinitely differentiable. The k-th order Taylor series approximation is obtained by truncating the above expression at k. However, if the function has $k+1$ derivatives, then one can use the Mean Value Theorem to truncate the above expression at k by a suitable modification and know the exact error. For instance, when $k = 1$

$$f(y) - (f(x) + \langle \nabla f(x), y - x\rangle) = \frac{1}{2}(y-x)^\top \nabla^2 f(\zeta)(y-x)$$

for some ζ in the line segment joining x and y.

1.2.2 Convexity

We start by defining the basic notions of a convex set and a convex function.

Definition 1.1. A set $K \subseteq \mathbb{R}^n$ is *convex* if, for every two points in K, the line segment connecting them is contained in K, i.e., for any $x, y \in K$ and $\lambda \in [0,1]$ we have

$$\lambda x + (1-\lambda)y \in K. \tag{1.1}$$

Definition 1.2. A function $f : K \to \mathbb{R}$ is *convex* if it satisfies

$$f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda)f(y) \tag{1.2}$$

for any $x, y \in K$ and $\lambda \in [0,1]$. If inequality (1.2) is strict for all $x \neq y$ and $\lambda \in (0,1)$, then we say that f is *strictly convex.*

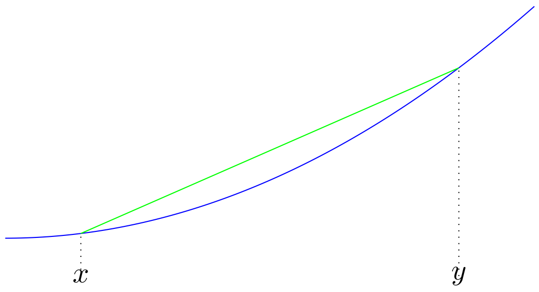

Fig. 1.1 A convex function.

Geometrically, this means that for any $x, y \in K$, the segment connecting points $(x, f(x))$ and $(y, f(y))$ lies above the graph of f (in the space $\mathbb{R}^n \times \mathbb{R}$): see Fig. 1.1.
While this is the most general definition of convexity, but not always the most useful. When f is smooth enough, we can give other equivalent definitions:

Proposition 1.3. If $f : K \to \mathbb{R}$ is differentiable, then it is convex iff

$$f(y) \geq f(x) + \langle \nabla f(x), y - x \rangle \tag{1.3}$$

for any $x, y \in K$.

The geometric interpretation of this condition is that for any $x \in K$, the tangent space of f at x should lie below the graph of f: see Fig. 1.2. In other words, the right-hand side of Eq. (1.3), which is the first-order Taylor approximation of f at x, is an *under-estimation* for the value of f at every other point $y \in K$.

Proof. We prove the one-dimensional case first.

Suppose f is convex, and fix any $x, y \in K$. Then for every $\lambda \in (0, 1]$ we have

$$(1 - \lambda) f(x) + \lambda f(y) \geq f(\lambda y + (1 - \lambda) x) = f(x + \lambda(y - x)).$$

Subtracting $(1 - \lambda) f(x)$ and dividing by λ yields

$$f(y) \geq f(x) + \frac{f(x + \lambda(y - x)) - f(x)}{\lambda}$$

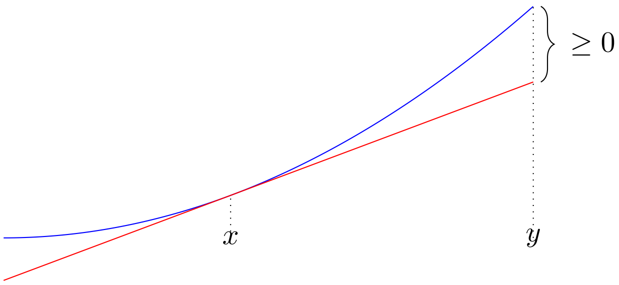

Fig. 1.2 The first-order condition for convexity.

and now it suffices to take the limit $\lambda \to 0$. (When $\lambda = 0$, there is nothing to prove.)

Conversely, suppose f satisfies Eq. (1.3) and fix $x, y \in K$ and $\lambda \in [0,1]$. Let $z = \lambda x + (1-\lambda)y$. The first-order approximation of f at z underestimates both $f(x)$ and $f(y)$; the difference between the linear approximation of f at z using the values $f(x)$ and $f(y)$ and the actual value $f(z)$ is a weighted average of these underestimations and thus also nonnegative. Formally, we have

$$f(x) \geq f(z) + f'(z)(x - z), \tag{1.4}$$

$$f(y) \geq f(z) + f'(z)(y - z), \tag{1.5}$$

and multiplying (1.4) by λ and (1.5) by $1 - \lambda$ yields

$$\lambda f(x) + (1-\lambda) f(y) \geq f(z).$$

To extend the proof to many dimensions, just note that after fixing points $x, y \in K$ it is enough to restrict our attention to the line segment connecting them. □

If the function has second derivatives, we can provide another characterization of convexity below. We leave the proof as an exercise.

Proposition 1.4. Suppose K is convex and open. If $f : K \to \mathbb{R}$ is twice differentiable, then it is convex iff

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

for any $x \in K$. If the inequality is strict for all $x \in K$, the function is called strictly convex.

In the rest of this lecture we will most often use the definition of Theorem 1.3.

1.3 Gradient Descent

We now introduce perhaps the simplest but extremely powerful *gradient descent* method to solve a convex program. Actually, it is not a single method, but a general framework with many possible realizations. We describe some concrete variants and analyze their performance. The performance guarantees that we are going to obtain will depend on assumptions that we make about f.

We describe the core ideas of the gradient descent methods in the *unconstrained setting*, i.e., $K = \mathbb{R}^n$. In Section 1.7.7 we discuss the constrained setting. Also, let us assume for simplicity that f has a unique minimum $x^\star$ (this follows if f is strictly convex) and that it is differentiable.

What does the gradient have to do with optimizing a convex function? We discuss this informally below. Suppose that we are at a point x, along with a value $f(x)$ that we want to decrease as much as possible. We want to accomplish this by moving to a point y at some pre-specified small distance from x. If we consider points y in a near neighbourhood of x, we should expect that the first-order Taylor approximation

$$f(y) \approx f(x) + \langle \nabla f(x), y - x \rangle$$

will be very tight: essentially an equality. Thus, if we set $\Delta x \overset{\text{def}}{=} y - x$, then we are tasked with minimizing the scalar product

$$\langle \nabla f(x), y - x \rangle = \langle \nabla f(x), \Delta x \rangle$$

where x is fixed, and $\|\Delta x\|$ is also given. The optimal solution of this problem has the form[1]

$$\Delta x = -\eta \cdot \nabla f(x)$$

[1] To see this, consider the problem of minimizing $\langle v, w \rangle$ over all v with $\|v\| = 1$ for a fixed w. One cannot do better than $-\|w\|$, because $|\langle v, w \rangle| \leq \|v\| \cdot \|w\| = \|w\|$ by Cauchy-Schwartz.

and it yields

$$f(y) \approx f(x) - \eta \|\nabla f(x)\|^2 .$$

Thus we see that the norm of the gradient controls the rate at which we can decrease f. More vividly, the gradient of f at x is the direction in which f grows the fastest, so if we are looking to minimize f, it makes the most sense to go in the opposite direction.

Any gradient descent method will work by constructing a sequence of points $x_1, x_2, \dots, x_T$ with the objective of getting very close to the optimum $x^\star$ after a small number of iterations. Usually, it will try to ensure that $f(x_0) \geq \cdots \geq f(x_T)$ (although the first method that we will show does not guarantee this). We will not be able to find the exact optimum; we can only hope to get ε-close to it, for a given accuracy requirement ε.

The first nice property of convex functions that we can discover in this context, and a reason why gradient descent does not work for a general f, is the following: if we keep decreasing the value of f, we will not get "stuck" in a local minimum which is not a global minimum. This is guaranteed by the following fact, which again points out the role of the gradient in optimizing f.

Proposition 1.5. For a differentiable convex function $f : \mathbb{R}^n \to \mathbb{R}$ and a point x, the following conditions are equivalent:

(a) x is a global minimum of f,
(b) x is a local minimum of f,
(c) $\nabla f(x) = 0$.

Proof. The direction (a)–(b) is trivial, and (b)–(c) holds for all f. For (c)–(a), just note that for any $y \in K$,

$$f(y) \geq f(x) + \langle \nabla f(x), y - x \rangle = f(x).$$

□

Remark 1.6. The above proposition generalizes to the case $K \subseteq \mathbb{R}^n$ as follows: for a closed convex set K and a convex function $f : K \to \mathbb{R}$, a point $x \in K$ minimizes f iff

$$\langle \nabla f(x), y - x \rangle \geq 0$$

for all $y \in K$. In other words, when there is no direction in K where the gradient decreases.

Let us proceed to the description of a general variant of gradient descent.

The algorithm will construct a sequence of points $x_1, x_2, \ldots, x_T$, for a value T which will depend on the assumptions we make about the structure of f and will be specified later. At the t-th iteration, knowing x_t, the algorithm takes x_{t+1} to be

$$x_{t+1} \stackrel{\text{def}}{=} x_t - \eta_t \nabla f(x_t)$$

where η_t is the *step size* (which will also depend on f and will be determined as per the setting.). For a given ε, our objective is to get ε-close to the optimum $f(x^\star)$ in as less iterations as possible. We want to understand what values of η_t will enable us to do achieve this goal.

1.4 Convex Functions, Oracles and their Properties

Till now we have ignored a very important detail: how is f given to the algorithm. In particular, to use the gradient algorithmically, one must be able to compute it. Here we will assume that the algorithm has access to an oracle which returns values of $f(x)$ and $\nabla f(x)$ for a query point $x \in K$. Such methods, which work in this oracle setting are referred to as *first-order methods.* Later, we will consider the setting where we would need access to a *second-order oracle* for f : given x, y output $(\nabla^2 f(x))^{-1} y$. Such methods are referred to as *second order methods.*

We also assume that we have a bound on the distance between the starting point and the optimum:

$$\|x_1 - x^\star\| \leq D.$$

If K were a bounded set, then $D = \operatorname{diam}(K)$ would be a valid choice; in the unconstrained setting, we must assume something about how far from the optimum we are starting.

Unfortunately, in general, it is still difficult to optimize f if we have no control at all over the magnitude of its gradient. We have to impose additional conditions on f.

We propose three such conditions. In each of these settings, we will obtain a guarantee on the convergence rate of our method, as well as a way to set the step lengths η_t.

1.4.1 Bounded gradient (Lipschitz f)

The simplest condition that we can propose is that the gradient be bounded from above: there should exist a $G > 0$ such that for all $x \in K$,

$$\|\nabla f(x)\| \leq G.$$

Why is this useful? Intuitively, a large gradient means that the function decreases very fast in the direction in which we are moving, which should be desirable. However, as we will see, this would make us unable to control the step size well enough.

Remark 1.7. This condition is equivalent to saying that f is G-Lipschitz, that is,

$$\|f(y) - f(x)\| \leq G \cdot \|y - x\|$$

for all $x, y \in K$. We will use this term in the sequel.

In this case we can get the following bound, which we prove in Section 1.5.1:

Theorem 1.8. There is a gradient-descent method which, given an ε, a starting point x_1 satisfying $\|x_1 - x^\star\| \leq D$, and a function f which is G-Lipschitz, produces a sequence of points $x_1, \ldots, x_T$ such that

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

where

$$T = \left(\frac{DG}{\varepsilon}\right)^2.$$

We mostly focus on the dependence of T on ε, and in this case it is

$$T = O\left(\varepsilon^{-2}\right), \quad \varepsilon = O\left(\frac{1}{\sqrt{T}}\right).$$

Note that we did not get any guarantee for the points $x_1, \ldots, x_T$ – just for their running averages.

Remark 1.9. We are using the Euclidean norm here and assuming that $\|\nabla f\|_2 = O(1)$. But sometimes we might find ourselves in a setting where $\|\nabla f\|_\infty = O(1)$, and a naive bound would give us just $\|\nabla f\|_2 = O(\sqrt{n})$. We will see in later lectures how to deal with such situations (see also Section 1.7.6).

1.4.2 Lipschitz gradient (smooth f)

We could also demand that the gradient be Lipschitz continuous.

Definition 1.10. We say that a function f is L-smooth (with a constant $L > 0$) if for all $x, y \in K$,

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

Remark 1.11. L is the Lipschitz constant of ∇f. We can equivalently say that $\nabla^2 f(x) \preceq LI$ for all $x \in K$.

To understand the usefulness of this condition, note that it implies the following crucial property:

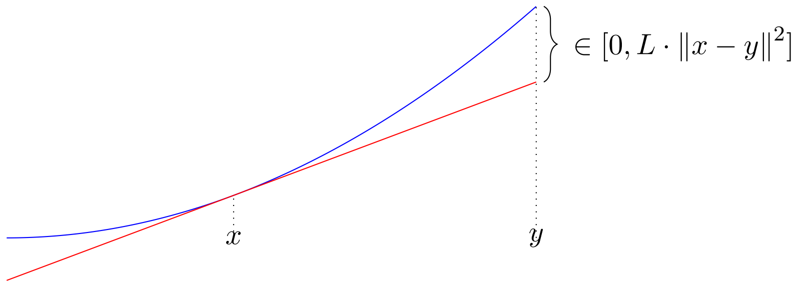

Fig. 1.3 The Bregman divergence.

Lemma 1.12. If f is L-smooth, then for any $x, y \in K$ we have

$$0 \leq f(y) - f(x) - \langle \nabla f(x), y-x \rangle \leq L \cdot \|x-y\|^2 .$$

This means that the distance of $f(y)$ from its first-order Taylor approximation at x is between 0 and $L \cdot \|x-y\|^2$, which is to be thought of as small; see Fig. 1.3. This distance is called the *Bregman divergence* with respect to the ℓ_2-norm.

Proof. We begin from the definition of convexity: $f(x) \geq f(y) + \langle \nabla f(y), x-y \rangle$, and use Cauchy-Schwarz along the way:

$$\begin{aligned} f(y) - f(x) &\leq \langle \nabla f(y), y-x \rangle \\ &= \langle \nabla f(x), y-x \rangle + \langle \nabla f(y) - \nabla f(x), y-x \rangle \\ &\leq \langle \nabla f(x), y-x \rangle + \|\nabla f(y) - \nabla f(x)\| \cdot \|y-x\| \\ &\leq \langle \nabla f(x), y-x \rangle + L \cdot \|x-y\|^2 . \end{aligned}$$

On the other hand we have (again from convexity)

$$f(y) - f(x) \geq \langle \nabla f(x), y-x \rangle .$$

□

This will enable us to obtain a better dependence of T on ε. The following bound is proved in Section 1.5.2:

Theorem 1.13. There is a gradient-descent method which, given an ε, a starting point x_1 and an L-smooth function f, produces a sequence of points $x_1, \dots, x_T$ such that

$$f(x_T) - f(x^\star) \le \varepsilon$$

and

$$T = O\left(\frac{LD^2}{\varepsilon}\right),$$

where $D = \|x_1 - x^\star\|$.

Again suppressing the constants, we see the improved dependence

$$T = O\left(\varepsilon^{-1}\right), \quad \varepsilon = O\left(\frac{1}{T}\right).$$

Moreover, this time we can show that we are really getting closer and closer to the optimum (not just on average).

While we will not cover it in the lectures, it is possible to attain a quadratic improvement using the important *accelerated gradient method* of Nesterov:

Theorem 1.14. There is an algorithm which, given an ε, a starting point x_1 satisfying $\|x_1 - x^\star\| \le D$, and an L-smooth function f, produces a sequence of points $x_1, \dots, x_T$ such that

$$f(x_T) - f(x^\star) \le \varepsilon$$

and

$$T = O\left(\frac{\sqrt{L}D}{\sqrt{\varepsilon}}\right).$$

1.4.3 Strong convexity

Another natural restriction would be to *strongly convex* functions, i.e., those that have all eigenvalues of the Hessian bounded below by a constant $\ell > 0$. In other words:

Definition 1.15 (strong convexity). We say that f is *ℓ-strongly convex* (with $\ell > 0$) if for all $x \in K$ we have

$$\nabla^2 f(x) \succeq \ell I.$$

Remark 1.16. This is equivalent to saying that

$$f(y) \geq f(x) + \langle \nabla f(x), y - x \rangle + \frac{\ell}{2} \|x - y\|^2 \tag{1.6}$$

for all $x, y \in K$. In other words, the corresponding Bregman divergence is lower bounded by $\frac{\ell}{2} \|x - y\|^2$.

If we also assume that f is G-Lipschitz, then we are able to get the following bound, whose proof appears in Section 1.5.3.

Theorem 1.17. There is an algorithm which, given an ε, a starting point x_1, and a function f which is both G-Lipschitz and ℓ-strongly convex, produces a sequence of points $x_1, \ldots, x_T$ such that

$$f(x_T) - f(x^\star) \leq \varepsilon$$

and

$$T = O\left(\frac{G^2}{\ell\varepsilon}\right).$$

1.5 Proofs of Convergence Rates

Now we prove the first two theorems stated in the previous section.

1.5.1 Lipschitz functions – proof of Theorem 1.8

Recall that we are guaranteed that f is G-Lipschitz. We will fix our step size η to be constant (independent of t).

We will be interested in understanding the quantity $f(x_t) - f(x^\star)$ – in particular, how fast it decreases with t. We begin by applying convexity and get

$$\begin{aligned} f(x_t) - f(x^\star) &\leq \langle \nabla f(x_t), x_t - x^\star \rangle \\ &= \frac{1}{\eta} \langle x_t - x_{t+1}, x_t - x^\star \rangle \\ &= \frac{1}{2\eta} \left(\|x_t - x_{t+1}\|^2 + \|x_t - x^\star\|^2 - \|x_{t+1} - x^\star\|^2 \right), \end{aligned} \tag{1.7}$$

where we used the well-known equality $2\langle a, b \rangle = \|a\|^2 + \|b\|^2 - \|a - b\|^2$. We can now take advantage of our assumption about f and write

$$\|x_t - x_{t+1}\|^2 = \eta^2 \|\nabla f(x)\|^2 \leq \eta^2 G^2. \tag{1.8}$$

From (1.7) and (1.8) we get

$$f(x_t) - f(x^\star) \leq \frac{1}{2\eta} \left(\eta^2 G^2 + \|x_t - x^\star\|^2 - \|x_{t+1} - x^\star\|^2 \right)$$

for every $t = 1, \ldots, T$. Notice that if we add all these inequalities, we will get a telescoping sum:

$$\sum_{t=1}^{T} (f(x_t) - f(x^\star)) \leq \frac{1}{2\eta} \left(T\eta^2 G^2 + \|x_1 - x^\star\|^2 - \|x_{T+1} - x^\star\|^2 \right).$$

We may bound the last term by simply zero. For the middle one, recall that we have introduced D which bounds the distance from the starting point to the optimum. Thus, using $\|x_1 - x^\star\|^2 \leq D^2$, we obtain

$$\sum_{t=1}^{T} (f(x_t) - f(x^\star)) \leq \frac{1}{2\eta} \left(T\eta^2 G^2 + D^2 \right) = \frac{1}{2} \left(T\eta G^2 + \frac{D^2}{\eta} \right).$$

The minimizing choice of η here is the one which satisfies $T\eta G^2 = \frac{D^2}{\eta}$, so that

$$\eta = \frac{D}{G\sqrt{T}}.$$

We have bounded the quantity $\frac{1}{T}\sum_{t=1}^{T} f(x_t) - f(x^\star)$. But from convexity of f we have[2] that $f\left(\frac{1}{T}\sum_{t=1}^{T} x_t\right) \leq \frac{1}{T}\sum_{t=1}^{T} f(x_t)$. Thus we get

$$f\left(\frac{1}{T}\sum_{t=1}^{T} x_t\right) - f(x^\star) \leq \frac{1}{2T}\left(T\eta G^2 + \frac{D^2}{\eta}\right) = \frac{DG}{\sqrt{T}}.$$

In order to get $\frac{DG}{\sqrt{T}} \leq \varepsilon$ we need to set $T \geq \left(\frac{DG}{\varepsilon}\right)^2$.

1.5.2 Smooth functions – proof of Theorem 1.13

Recall that we are assuming that f is L-smooth, i.e., that ∇f is L-Lipschitz. This implies Theorem 1.12 (a bound on the Bregman divergence).

We prove this theorem for $D = \max\{\|x - x^\star\| : f(x) \leq f(x_1)\}$. To prove the above theorem with $D = \|x_1 - x^\star\|$ one needs to show that in all the subsequent steps $\|x_t - x^\star\| \leq \|x_1 - x^\star\|$. This requires us to prove the following inequality which follows from lower bounding the Bregman divergence. We omit its proof.

$$\|\nabla f(x) - \nabla f(y)\|^2 \leq L \cdot \langle x - y, \nabla f(x) - \nabla f(y)\rangle.$$

Once again, we will keep the step length η constant and take

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

Let us see how the value of f changes as we iterate. Use Theorem 1.12 to get

$$\begin{aligned} f(x_{t+1}) - f(x_t) &\leq \langle \nabla f(x_t), x_{t+1} - x_t\rangle + L \cdot \|x_{t+1} - x_t\|^2 \\ &= -\eta \|\nabla f(x_t)\|^2 + L\eta^2 \|\nabla f(x_t)\|^2 . \end{aligned}$$

The minimizing value of η is $\frac{1}{2L}$. By substituting it, we get

$$f(x_{t+1}) - f(x_t) \leq -\frac{1}{4L} \|\nabla f(x_t)\|^2 . \tag{1.9}$$

Intuitively, this means that, at every step, either the gradient is large and we are making good progress, or it is small, which means that we

[2] Extend Eq. (1.2) to T points.

are already close to the optimum. Indeed, if we are at a distance θ from the optimum value:

$$f(x_t) - f(x^\star) \geq \theta,$$

then by convexity and Cauchy-Schwarz,

$$\theta \leq f(x_t) - f(x^\star) \leq \langle \nabla f(x_t), x_t - x^\star \rangle \leq \|\nabla f(x_t)\| \cdot \|x_t - x^\star\|$$

and if we bound $\|x_t - x^\star\|$ by D (this follows since $f(x_t) \leq f(x_1)$), then we get

$$\|\nabla f(x_t)\| \geq \frac{\theta}{D}$$

and, by (1.9),

$$f(x_{t+1}) - f(x_t) \leq -\frac{\theta^2}{4LD^2}.$$

Until our distance from the optimum goes down below $\frac{\theta}{2}$, we will make a progress of $\Omega\left(\frac{\theta^2}{LD^2}\right)$ at every step, so we will take at most $O\left(\frac{LD^2}{\theta}\right)$ steps before this happens. Then this process of halving is repeated, and so on, until we reach the required distance ε. Bringing the distance down from $\frac{\theta}{2^i}$ to $\frac{\theta}{2^{i+1}}$ requires $O\left(\frac{LD^2 2^i}{\theta}\right)$ steps, so we get

$$\sum_{i=0}^{\log \frac{\theta}{\varepsilon}} O\left(\frac{LD^2 2^i}{\theta}\right) = O\left(\frac{LD^2 2^{\log \frac{\theta}{\varepsilon}}}{\theta}\right) = O\left(\frac{LD^2}{\varepsilon}\right)$$

steps before we are ε-close to the optimum.

1.5.3 Strongly convex functions – proof of Theorem 1.17

Recall that f is ℓ-strongly convex and G-Lipschitz. As a start, we will try to mimic the proof from Section 1.5.1. However, we will not fix the step size η_t to be constant, and we use the first-order condition for convexity in the strong form (Eq. (1.6)). We easily get

$$f(x_t) - f(x^\star) \leq \frac{1}{2\eta_t}\left(\eta_t^2 G^2 + \|x_t - x^\star\|^2 - \|x_{t+1} - x^\star\|^2\right) - \frac{\ell}{2}\|x_t - x^\star\|^2 \tag{1.10}$$

for every $t = 1, \ldots, T$. Now, to obtain a better bound than previously, we must make good use of the new term (the last one). Intuitively, if $\|x_t - x^\star\|$ is large, then it is very helpful, but as t grows and $\|x_t - x^\star\|$

decreases, its importance is diminished. We will try to prevent this from happening by multiplying our inequality by t. And we will still aim to obtain a telescoping sum. So let us sum all the t-multiples of Eq. (1.10):

$$\sum_{t=1}^{T} t\left(f(x_t) - f(x^\star)\right) \leq \frac{G^2}{2}\sum_{t=1}^{T} t\eta_t + \sum_{t=2}^{T} \|x_t - x^\star\|^2 \cdot \left(\frac{t}{2\eta_t} - \frac{\ell t}{2} - \frac{t-1}{2\eta_{t-1}}\right)$$
$$+ \|x_1 - x^\star\|^2 \cdot \left(\frac{1}{2\eta_1} - \frac{\ell}{2}\right) - \|x_{T+1} - x^\star\|^2 \cdot \frac{T}{2\eta_T}.$$

As before, we bound the last term by just zero. Now, to make the sum telescoping, we would like to get $\frac{t}{2\eta_t} - \frac{\ell t}{2} - \frac{t-1}{2\eta_{t-1}} = 0$ for every $t = 2, \ldots, T$. As for the term $\frac{1}{2\eta_1} - \frac{\ell}{2}$, we would also prefer to remove it, so as not to have any dependence on $\|x_1 - x^\star\|$. Solving this system of equations (beginning with $\frac{1}{2\eta_1} - \frac{\ell}{2} = 0$) yields the following setting of η_ts:

$$\eta_t = \frac{2}{\ell(t+1)}.$$

We are thus left with:

$$\sum_{t=1}^{T} t\left(f(x_t) - f(x^\star)\right) \leq \frac{G^2}{2}\sum_{t=1}^{T} t\eta_t = \sum_{t=1}^{T} \frac{G^2 t}{\ell(t+1)} \leq \sum_{t=1}^{T} \frac{G^2}{\ell} = \frac{G^2 T}{\ell}.$$

The rest is straightforward: we normalize by $(1 + \cdots + T) = \frac{T(T+1)}{2}$ and use convexity of f to get

$$f\left(\frac{2}{T(T+1)}\sum_{t=1}^{T} t \cdot x_t\right) - f(x^\star) \leq \frac{2G^2}{\ell(T+1)},$$

and bringing this down to ε requires setting

$$T = O\left(\frac{G^2}{\ell\varepsilon}\right).$$

1.6 Strong Convexity - Solving PSD linear systems

Suppose we have a linear system with a positive definite constraint matrix, i.e. we want to find a vector x satisfying $Ax = b$, where $A \succ 0$.

Let us formulate it as a convex problem. Define

$$f(x) = \frac{1}{2}x^\top Ax - x^\top b.$$

Then f is strongly convex:

$$\nabla^2 f(x) = A \succ 0$$

and one can verify that

$$\nabla f(x) = Ax - b$$

so that $x^\star$ minimizes f iff it is a solution to the linear system $Ax = b$. So solving the system amounts to minimizing f, and we will do this using gradient descent. We will construct a sequence of points $x_1, \ldots, x_T$ using the formula

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

as previously. However, this time we will not use the same step length η_t for every iteration, but instead (**exercise**) analytically find a closed-form expression for the best η_t, i.e.

$$\eta_t = \operatorname{argmin}_\eta \, f\left(x_t - \eta \cdot \nabla f(x_t)\right).$$

So at each step we will decrease f as much as possible by going in the direction opposite to the gradient. This is called the *steepest descent method.*

If we do this, then one can show (**exercise**) that the convergence rate of this method (to a predefined accuracy ε: $f(x_T) - f(x^\star) \le \varepsilon$) is given by

$$T = O\left(\kappa(A) \cdot \log \frac{1}{\varepsilon}\right),$$

where $\kappa(A) = \frac{L}{\ell}$, the ratio between the largest and the smallest eigenvalue of A, is the *condition number* of the matrix A.[3]

[3] For more information on this and the related *Conjugate Gradient method*, see Chapter 15 of the monograph [Vis13].

1.7 Discussion

1.7.1 Gradient vs. Sub-gradient

Throughout this lecture we assumed the function f to be differentiable or twice differentiable as and when we needed. However, in many applications (practical and theoretical), f is not differentiable everywhere. Could any of the methods presented in this section work in this non-differentiable setting? Start by noting that if f is convex then it is continuous. Hence, we need to only consider the issue of non-differentiable f. Non-differentiability poses the problem that there may be some points at which the gradient is not unique. In general, at a non-differentiable point x, the set of gradients forms a set which which we denote by $\partial f(x)$ and an element of it is referred to as a *sub-gradient*. First we note that the convexity of f implies that for any x, y

$$f(y) \geq f(x) + \langle g, y - x \rangle$$

for all $g \in \partial f(x)$. Thus, we could modify the gradient descent in a natural manner:

$$x_{t+1} = x_t - \eta_t g$$

for any $g \in \partial f(x_t)$ given to us by the first order oracle for f. Further, it can be easily checked that the guarantee claimed in Theorem 1.8 and its corollaries carry over with the following parameter:

$$G \stackrel{\text{def}}{=} \sup_x \sup_{g \in \partial f(x)} \|g\|.$$

1.7.2 Dependence on ε

We have seen in Theorems 1.8, 1.13, 1.14 and 1.17 that our methods converge to the ε-approximate solution in a number of iterations which is $O\left(\text{poly}\left(\frac{1}{\varepsilon}\right)\right)$, with the exponent equal to 2, 1 or $\frac{1}{2}$. However, a polynomial dependence on ε does not look very good if we hope to use these methods in computer science. For such applications, the right answer should be $O\left(\text{poly}\left(\log \frac{1}{\varepsilon}\right)\right)$. And, if the function is both strongly convex and smooth, it is possible to obtain a convergence rate similar to the one in Section 1.6: see e.g. Section 3.4.2 of [Bub14].

1.7.3 Dependence on n

Note that n does not appear anywhere in the oracle complexity[4] of the presented algorithms. This is an important feature of first-order methods. (Of course, the time complexity of a single iteration will still depend on n.)

1.7.4 Coordinate Descent

Instead of moving in the exact direction in which f decreases the fastest, we may limit ourselves only to moving along the standard basis vectors, i.e. to changing only one coordinate of x at a time.

For example, rather than computing the whole gradient

$$\nabla f(x_t) = \frac{\partial f}{\partial x_1}(x_t) \cdot e_1 + \cdots + \frac{\partial f}{\partial x_n}(x_t) \cdot e_n$$

(where $e_1, \ldots, e_n$ are standard basis vectors), we can only pick one *random* coordinate i and update

$$x_{t+1} = x_t - \eta_t \cdot \frac{\partial f}{\partial x_i}(x_t) \cdot e_i.$$

We can analyze this *random coordinate descent* by examining the expected decrease in the function value. One can prove the following theorem. We omit its simple proof.

Theorem 1.18. There is an algorithm which, given an ε, a starting point x_1 satisfying $\|x_1 - x^\star\| \leq D$, and a function f which is G-Lipschitz, produces a sequence of (random) points $x_1, \ldots, x_T$ such that

$$\mathbb{E}\left[f\left(\frac{1}{T}\sum_{t=1}^{T} x_t\right)\right] - f(x^\star) \leq \varepsilon$$

and

$$T = O\left(\left(\frac{DG}{\varepsilon}\right)^2 \cdot n\right).$$

[4] The number of queries for $f(x)$, $\nabla f(x)$ etc.

As we would expect, changing only one coordinate at a time comes at a cost of an n-fold increase in the number of iterations (compared to our first method of Theorem 1.8).

1.7.5 Online Convex Optimization

Consider the following game against an adversary. Given are: a convex set of strategies K, and a family of convex functions $\mathcal{F}$ (for example, all convex functions which are G-Lipschitz). The game proceeds in rounds. In the t-th round, the player picks a point $x_t \in K$, and then the adversary picks a function $f_t \in \mathcal{F}$. The value $f_t(x_t)$ is considered to be the player's loss at time t. The objective of the game is to minimize the *regret* of the player up to a time T:

Definition 1.19. For a sequence of points $x_1, \ldots, x_T$ and functions $f_1, \ldots, f_T$, the *regret* up to time T is defined as

$$\text{Regret}_T = \sum_{t=1}^{T} f_t(x_t) - \min_{x \in K} \sum_{t=1}^{T} f_t(x).$$

I.e., we compare the player's loss to the minimum loss which could be achieved if one knew all the functions f_t beforehand, but were only allowed to pick a single argument x for all of them.[5] The player is trying to minimize the regret, and the adversary is trying to maximize it. This model has many practical applications, such as stock pricing and portfolio selection, see [Haz14].

One can describe a strategy for the player which uses gradient descent:

$$x_{t+1} = x_t - \eta_t \cdot \nabla f_t(x_t)$$

for a suitable choice of η_t. Even though the functions f_t change from iteration to iteration, one can still obtain a similar guarantee to the one from our first analysis. Namely, we are able to make the *average regret* go to 0 at a rate inversely proportional to the square root of T:

$$\frac{\text{Regret}_T}{T} = O\left(\frac{1}{\sqrt{T}}\right).$$

[5] For some choices of $x_1, \ldots, x_t, f_1, \ldots, f_t$, regret can be negative.

Indeed, note that in the proof of Section 1.5.1 we never used the fact that the function f remains the same between iterations. We actually proved the following:

Corollary 1.20. There is an algorithm which, given a starting point x_1, parameters ε and D, and a family $f_1, \dots, f_T$ of convex functions which are G-Lipschitz, produces a sequence of points $x_1, \dots, x_T$ such that

$$\frac{1}{T}\sum_{t=1}^{T}(f_t(x_t) - f_t(x^\star)) \le \frac{DG}{\sqrt{T}}$$

for any $x^\star$ satisfying $\|x_1 - x^\star\| \le D$, and T being

$$T = \left(\frac{DG}{\varepsilon}\right)^2.$$

Moreover, it produces each point x_{t+1} knowing only the functions $f_1, \dots, f_t$.

However, we will see in the next lecture that multiplicative weight update methods are often able to perform much better in the online learning setting.

There are also other optimization scenarios where access to information is even more limited (and such models are thus closer to real life applications). For example, we might be only given oracle access to the function (i.e., we do not see any ”formula” for f). In particular, we cannot compute the gradient directly. (This is called the *bandit setting.*) Even then, methods based on gradient descent are very competitive.

1.7.6 Non-Euclidean Norms

Our algorithm for the L-smooth case can be easily adapted to work with functions which are smooth with respect to any pair of dual norms.[6]

[6] Let $\|\cdot\|$ be any norm. We define the dual norm $\|\cdot\|_\star$ as follows:

$$\|y\|_\star = \sup_{x\in\mathbb{R}^n:\|x\|=1} |\langle x, y\rangle|.$$

Then we say that a function f is L-smooth with respect to $\|\cdot\|$ if for any $x, y \in K$, $\|\nabla f(x) - \nabla f(y)\|_\star \le L \cdot \|x - y\|$.

Now, moving in the direction of the gradient makes progress which we can measure using the Euclidean norm, because then

$$\langle \nabla f(x_t), x_{t+1} - x_t \rangle = \langle \nabla f(x_t), -\eta \nabla f(x_t) \rangle = -\eta \left\| \nabla f(x_t) \right\|_2^2 .$$

If we want to utilize a different pair of dual norms, we must adapt our direction of descent to the geometry that they induce. To this end, we move instead in the direction

$$x_{t+1} = x_t - \eta \left(\nabla f(x_t) \right)^{\#} ,$$

where for any $x \in \mathbb{R}^n$ we define $x^{\#}$ to be the minimizer of

$$\frac{1}{2} \left\| x^{\#} \right\|^2 - \left\langle x^{\#}, x \right\rangle . \tag{1.11}$$

(Note that for $\|\cdot\| = \|\cdot\|_2$ we have $x^{\#} = x$.) Theorem 1.12 still holds in the new setting, and one can show that a step size of $\eta = \frac{1}{L}$ allows us to get

$$f(x_{t+1}) - f(x_t) \leq -\frac{1}{2L} \left\| \nabla f(x_t) \right\|_\star^2 .$$

Then the analysis proceeds as in Section 1.5.2, giving us the following generalization of Theorem 1.13:

Theorem 1.21. There is an algorithm which, given an ε, a starting point x_1 satisfying $\|x_1 - x^\star\| \leq D$, and a function f which is L-smooth with respect to the norm $\|\cdot\|$, produces a sequence of points $x_1, \ldots, x_T$ such that

$$f(x_T) - f(x^\star) \leq \varepsilon$$

and

$$T = O\left(\frac{LD^2}{\varepsilon} \right) .$$

To produce each point x_t, it makes one call to a routine which computes $x^{\#}$ for a given x, i.e. minimizes an expression of the form (1.11).

1.7.7 Constrained setting – projection

If K is not the whole space $\mathbb{R}^n$, then our choice of the next iterate x_{t+1} in the gradient descent method might fall outside of the convex body

K. In this case we need to project it back onto K: to find the point in K with the minimum distance to x, and take it to be our new iterate instead:

$$x_{t+1} = \mathrm{proj}_K \left(x_t - \eta_t \cdot \nabla f(x_t)\right).$$

The convergence rates remain the same (for example, the first proof just carries over to this new setting, since the value of f at the new point will be at least as good as the bound that we had derived for its value at the original point). However, depending on K, the projection may or may not be difficult (or computationally expensive) to perform.

More precisely, as long as the algorithm has access to an oracle which, given a query point x, returns the projection $\mathrm{proj}_K(x)$ of x onto K, then for the G-Lipschitz case we have the following analogue of Theorem 1.8:

Theorem 1.22. There is an algorithm which, given an ε, a function $f : K \to \mathbb{R}$ which is G-Lipschitz and has a minimum $x^\star$, and a starting point x_1 satisfying $\|x_1 - x^\star\| \le D$, produces a sequence of points $x_1, \dots, x_T \in K$ such that

$$f\left(\frac{1}{T}\sum_{t=1}^{T} x_t\right) - f(x^\star) \le \varepsilon$$

and

$$T = \left(\frac{DG}{\varepsilon}\right)^2.$$

To compute each point x_t, it uses one gradient query and one projection query.

And for the L-smooth case we get the following analogue of Theorem 1.13.

Theorem 1.23. There is an algorithm which, given an ε, a function $f : K \to \mathbb{R}$ which is L-smooth and has a minimum $x^\star$, and a starting point x_1 satisfying $\|x_1 - x^\star\| \le D$, produces a sequence of points $x_1, \dots, x_T \in K$ such that

$$f(x_T) - f(x^\star) \le \varepsilon$$

and

$$T = O\left(\frac{LD^2 + f(x_1) - f(x^\star)}{\varepsilon}\right).$$

To compute each point x_t, it uses one gradient query and one projection query.

1.7.8 Constrained setting – the Frank-Wolfe algorithm

In cases where computing projections is prohibitively difficult (for example, harder than the original problem), one can use another variant of gradient descent: the *Frank-Wolfe algorithm*[7]. At the t-th step, it first considers the first-order Taylor approximation of f around x_t, and tries to minimize this function over K, obtaining a minimizer y_t. Then it takes the new point x_{t+1} to be a weighted average of x_t and y_t, which is guaranteed to also be in K. More precisely, it first minimizes the function

$$f(x_t) + \langle \nabla f(x), y_t - x_t \rangle$$

over $y_t \in K$. But this function is linear in y_t, and the task amounts to finding the minimizer of $\langle y_t, \nabla f(x_t) \rangle$ over $y_t \in K$. The new point is selected as

$$x_{t+1} = (1 - \gamma_t)x_t + \gamma_t y_t$$

with $\gamma_t \in [0, 1]$ being a parameter[8]. Because this is a convex combination, we get that $x_{t+1} \in K$ (as long as the starting point x_1 was in K).

This way, instead of having to project onto K, we need to be able to optimize linear functions over K, which may very well be easier. For example, if K is a polytope, then we are left with a linear programming subproblem.

Using this method one can obtain the same bound on the number of iterations as in Theorem 1.13: (see also Theorem 3.4 in [Bub14]):

[7] Also called the conditional gradient descent method.

[8] A good choice is $\gamma_t = \frac{2}{t+1}$.

Theorem 1.24. There is an algorithm which, given parameters ε and D, where $D = \operatorname{diam}(K)$, a function $f : K \to \mathbb{R}$ which is L-smooth (with respect to any norm $\|\cdot\|$) and has a minimum $x^\star$, and any starting point $x_1 \in K$, produces a sequence of points $x_1, \ldots, x_T \in K$ such that

$$f(x_T) - f(x^\star) \leq \varepsilon$$

and

$$T = O\left(\frac{LD^2}{\varepsilon}\right).$$

To compute each point x_t, it uses one gradient query and one call to a routine which optimizes a linear function over K.

This algorithm has another important feature: if K is a polytope and x_1 is a vertex of K, then any iterate x_t can be written as a convex combination of t vertices of K. In constrast, Caratheodory's theorem guarantees that any $x \in K$ can be written as a convex combination of at most $n+1$ vertices of K. Because T is independent of n (and will often be much smaller), we can say that x_T is sparse in the polytope-vertex representation.

2

Multiplicative Weights Update vs. Gradient Descent

2.1 Overview

In the previous lecture we discussed gradient descent-type methods from convex optimization and showed how they can be straightforwardly translated into regret-minimizing algorithms in the online convex optimization setting. The key observation was that several gradient-descent type methods are oblivious to the fact that the same function is used in every round. Today we turn this idea on its head: namely, we begin with an algorithm that solves an online problem, and end up with a (new) convex optimization method. As a result, we can get improved dependence on the number of iterations when we are in settings other than the Euclidean space or have guarantees on the function on the Euclidian norm. For instance, suppose we know that $\|\nabla f(x)\|_\infty = O(1)$ which, at best, implies a bound $\|\nabla f(x)\|_2 = O(\sqrt{n})$. We obtain a gradient-descent type algorithm that takes $O({}^{\log n}/_{\varepsilon^2})$ iterations as opposed to $O({}^{\sqrt{n}}/_{\varepsilon^2})$ iterations if we use the algorithm from the previous lecture. The journey we take in deducing this result is long and rich. The central player is the old, famous and versatile: the Multiplicative Weights Update (MWU) method. In the process we also explain how to bet on stocks, how to solve linear programs approximately and quickly, and how MWU can be thought of as a method in convex optimization. We also hint how one may solve semi-definite programs approximately using a matrix variant of the MWU.

2.2 The Multiplicative Weights Update Method

MWU has appeared in the literature at least as early as the 1950s and has been rediscovered many times in many fields since then. It has applications in optimization (solving LPs), game theory, finance (portfolio optimization), machine learning (Winnow, AdaBoost, Hedge), theoretical computer science (devising fast algorithms for LPs and SDPs), and many more. We refer the reader to the survey of Arora, Hazan, and Kale [AHK12] for a survey on Multiplicative Weight Update method.

To describe MWU, consider the following game. We want to invest in the stock market, i.e., to buy/sell stocks and make money. To simplify, assume that each day we can either buy or sell one share of one particular kind of stock. Now, our goal is to trade stock each day in order to maximize our revenue.

Unfortunately, we do not know much about the stock market. However, we have access to n *experts* that know something (or a lot) about the stock market. At the beginning of each day, each expert i advices us on whether we should buy or sell. Once we hear all of the advice, we decide what to do. At the end of each day we observe whether our profit went up or down. If the advice of an expert was incorrect (i.e., they said "buy" when the best option was "sell"), we say that the expert made a *mistake*.

Now, of course, experts may make mistakes, or may even be adversarial and collude against us. Nevertheless, our goal is to, based on advice given by the experts, do as *well as possible*; i.e., consider the expert $\star$ who gave the most correct advice, our aim is to do almost as good as expert $\star$.

To formalize the setup, we define m_i^t and M^t, for each expert i and each day t, as follows

$$m_i^t \stackrel{\text{def}}{=} \text{number of mistakes made by expert } i \text{ until day } t,$$

and

$$M^t \overset{\text{def}}{=} \text{number of mistakes made by us until day } t.$$

Having these definitions in hand, our goal is to ensure that the *regret*

$$\text{Regret}_t \overset{\text{def}}{=} M^t - m^t_\star$$

is minimized, where $m^t_\star = \min_i m^t_i$. Note that M^t may even be smaller than $m^t_\star$.

Let us first consider some obvious approaches to regret-minimization given the advice of experts. One natural strategy is to take advice from an expert who was right on the previous day. However, one can very easily design examples where this strategy will perform very poorly.[1] Alternately, we can consider *Majority Vote* in which we simply see the advice for a given day, and take the majority option. In other words, we will buy if and only if the majority of the experts say to buy. However, in this case we can also design examples where this algorithm performs poorly.

2.2.1 The Weighted Majority Algorithm

In the two aforementioned approaches we were either too radical (i.e., ignoring experts who just made a mistake) or too forgiving (i.e., treating all experts equally regardless of their history). Instead, we could consider a strategy which combines the best from both the worlds by considering *Weighted Majority.* Here, we take an expert's past advice into consideration using weights, and take the weighted majority in order to determine which advice to follow.

Let w^t_i be the weight of expert i at the beginning of day t. We can also think of w^t_i as our confidence in expert i – the larger w^t_i the more confidence we have. We are unbiased in the beginning, and let

$$w^1_i = 1, \quad \forall i.$$

If expert i is right on day t, then we do not change w^t_i, but otherwise

[1] For example, if half the experts are right on even days and wrong on odd days, and the other half or the experts are wrong on even days and right on odd days, we will always make the wrong choice.

penalise that expert. Formally, define f_i^t to be

$$f_i^t \stackrel{\text{def}}{=} \begin{cases} 1 & \text{if expert } i \text{ makes a mistake on day } t \\ 0 & \text{otherwise} \end{cases}$$

Then, we set

$$w_i^{t+1} \stackrel{\text{def}}{=} w_i^t(1 - \varepsilon f_i^t)$$

for an $\varepsilon > 0$. This introduction of ε is *crucial* and is a reflection of our trust in the predictions of the experts. $\varepsilon = 1$ would correspond to the setting when we know that there is some expert who is always correct. On the other hand, if we set $\varepsilon = 0$, we are discarding the history. Morally, this parameter plays the same role as the η-parameter in gradient-descent type methods and could depend on t and, in principle, be different for each expert. For now, we just think of ε as a small enough number.

How do we make a decision given these weights? Towards this, let $\Phi^t \stackrel{\text{def}}{=} \sum_i w_i^t$ denote the sum of all weights of the experts. On day t we decide to buy if the sum of the weights of experts saying to buy is at least $\Phi^t/2$, and sell otherwise. This completes the description of the Weighted Majority Algorithm (WMA).

How well does WMA perform? Before we delve into the proofs, let us state two fact that we will use repeatedly.

Fact 2.1. For any x the following holds

$$(1 - x) \leq e^{-x}.$$

Fact 2.2. For any $|x| \leq 1/2$ we have

$$-\ln(1 - x) \leq x + x^2.$$

Now we are ready to prove the following theorem.

Theorem 2.3. After t steps, let m_i^t denote the number of mistakes made by expert i and M^t the number of mistakes WMA had made, and let $0 < \varepsilon \leq 1/2$. Then, for every i we have

$$M^t - 2(1+\varepsilon)m_i^t \leq \frac{2\ln n}{\varepsilon}.$$

In other words, the performance of the WMA is within a factor of $2(1+\varepsilon)$ of the best expert up to an additive factor of about $\ln n/\varepsilon$. If we consider the average after time t, i.e., M^t/t, the above theorem implies that if $t \geq 2\ln n/\varepsilon^2$, then the difference between the average mistakes made by the WMA is no more than an additive ε of that of the best expert. This is formalized in the following corollary:

Corollary 2.4. After T days, let $m_\star^T$ be the number of mistakes the best expert has made and let M^T be the number of mistakes WMA has made. If $T \geq \frac{2\ln n}{\varepsilon^2}$ where $0 < \varepsilon \leq 1/2$, then

$$\frac{1}{T}\left(M^T - 2(1+\varepsilon)m_\star^T\right) \leq \varepsilon.$$

Note that the corollary says that in the long run, i.e., for large T, the algorithm essentially makes at most two times more mistakes than the best expert in hindsight![2]

The proof is easy and relies on observing that every time the WMA makes a mistake, the total weight on the experts reduces by a multiplicative factor of $(1 - \varepsilon/2)$.

Proof. Assume that we make a mistake on day t. Then, at least half of the total weight of the experts will get decreased by $(1-\varepsilon)$, otherwise the weighted majority would have told us to take the other option. Hence,

$$\Phi^{t+1} \leq \Phi^t\left(\frac{1-\varepsilon}{2} + \frac{1}{2}\right) = \Phi^t\left(1 - \frac{\varepsilon}{2}\right).$$

[2] Interestingly, the factor of 2 in the bound is optimal for a deterministic strategy. We will later see that randomization can help.

Therefore, every time we make a mistake the total weight of the experts decreases by the multiplicative factor of $\left(1 - \frac{\varepsilon}{2}\right)$. Hence, we can upper bound Φ^{t+1} as follows

$$\Phi^{t+1} \leq \Phi^1 \left(1 - \frac{\varepsilon}{2}\right)^{M^t} = n \left(1 - \frac{\varepsilon}{2}\right)^{M^t} \leq n e^{-\varepsilon M^t/2}, \tag{2.1}$$

where we recall that all weights were initialized to 1 so $\Phi^1 = n$ and use Fact 2.1.

On the other hand, since we have that Φ^{t+1} is the sum of all the weights of the experts on day $t+1$, we also have the following lower bound $\forall i$

$$\begin{aligned} \Phi^{t+1} &\geq w_i^{t+1} \\ &= w_i^1 \left(1 - \varepsilon\right)^{m_i^t} \\ &= \left(1 - \varepsilon\right)^{m_i^t}, \end{aligned} \tag{2.2}$$

where again we use the fact that $w_i^1 = 1$.

Now, putting together the upper bound (2.1) and lower bound (2.2), we obtain

$$\left(1 - \varepsilon\right)^{m_i^t} \leq n e^{-\varepsilon M^t/2}.$$

Taking logarithms on both sides we get that

$$m_i^t \ln\left(1 - \varepsilon\right) \leq \ln n - \frac{\varepsilon}{2} M^t. \tag{2.3}$$

Using Fact 2.2, from (2.3) we conclude that

$$-\varepsilon(1 + \varepsilon) m_i^t \leq \ln n - \frac{\varepsilon}{2} M^t,$$

which further implies that

$$M^t - 2(1 + \varepsilon) m_i^t \leq \frac{2 \ln n}{\varepsilon}.$$

This gives a bound on the regret as desired. □

2.2.2 The Multiplicative Weights Update Algorithm

The algorithm we saw so far makes, up to a factor of two, optimal choices about trading a single item. Let us now consider a more general setting. As before, there are n experts, and following solely expert i's advice on day t incurs a cost f_i^t where $f^t : [n] \to [-1,1]$. However, in many cases, as with stocks, we can take fractional decisions. Hence, for a given vector $p^t \in \Delta_n$ which represents a convex combination of experts,[3] we incur a *loss* of $\langle p^t, f^t \rangle$.

Equivalently, we can think of p^t as a probability distribution from where we select a single expert x^t, and incur a loss of $f^t(x^t)$. In this case, the expected loss $\mathbb{E}_{x^t \sim p^t}[f^t(x^t)] = \langle p^t, f^t \rangle$. Furthermore, we compete against the best convex combination of the experts, and are interested in minimizing the *regret*

$$\sum_{t=1}^{T} \langle p^t, f^t \rangle - \min_{p \in \Delta_n} \sum_{t=1}^{T} \langle p, f^t \rangle.$$

How do we update p^t? The strategy is similar to the WMA: maintain a weight w_i^t for the i-th expert where we start with $w_t^1 = 1$. Given the loss function f^t, the weights are updated as

$$w_i^{t+1} \stackrel{\text{def}}{=} w_i^t(1 - \varepsilon f_i^t)$$

for a parameter $\varepsilon > 0$ which has the same intention as before. Since f_i^t could be negative or positive, the weight can increase or decrease. However, since $\|f^t\|_\infty \leq 1$ for all t, the weights always remain non-negative. Finally, since we need a probability distribution (or convex combination) $p^t \in \Delta_n$, we normalize the weight vector w^t by the total weight $\Phi^t \stackrel{\text{def}}{=} \sum_i w_i^t$ to obtain

$$p^t \stackrel{\text{def}}{=} \frac{w^t}{\Phi^t}.$$

We call this method the Multiplicative Weights Update (MWU) algorithm. We are now ready to prove the main theorem of this section. The only difference from Theorem 2.3 is that the factor of 2 goes away! The proof is similar to that of Theorem 2.3.

[3] A point p is in Δ_n if $\|p\|_1 = 1$ and $p_i \geq 0$ for all i.

Theorem 2.5. Assume that $\|f^t\|_\infty \leq 1$ for every t, and let $0 < \varepsilon \leq 1/2$. Then, the MWU algorithm produces a sequence of probability distributions $p^1, \ldots, p^T$ such that

$$\sum_{t=1}^{T} \left\langle p^t, f^t \right\rangle - \inf_{p \in \Delta_n} \sum_{t=1}^{T} \left\langle p, f^t \right\rangle \leq \frac{\ln n}{\varepsilon} + \varepsilon T. \tag{2.4}$$

Thus, the number of iterations after which the average regret becomes less than 2ε is at most $\frac{\ln n}{\varepsilon^2}$.

Proof. We have

$$\Phi^{t+1} = \sum_i w_i^{t+1} = \sum_i w_i^t \left(1 - \varepsilon f_i^t\right).$$

Now, using the facts that $p_i^t = w_i^t/\Phi^t$, $\|p^t\|_1 = 1$ and $p_i^t \geq 0$, we rewrite the above equation as follows

$$\begin{aligned} \Phi^{t+1} &= \sum_i \left(p_i^t \Phi^t\right) \left(1 - \varepsilon f_i^t\right) \\ &= \Phi^t - \varepsilon \Phi^t \sum_i p_i^t f_i^t \\ &= \Phi^t \left(1 - \varepsilon \left\langle p^t, f^t \right\rangle\right). \end{aligned} \tag{2.5}$$

Following Fact 2.1, equality (2.5) can be written as

$$\Phi^{t+1} \leq \Phi^t e^{-\varepsilon \left\langle p^t, f^t \right\rangle}.$$

Therefore, after T rounds we have

$$\begin{aligned} \Phi^{T+1} &\leq \Phi^1 e^{-\varepsilon \sum_{t=1}^{T} \left\langle p^t, f^t \right\rangle} \\ &= n e^{-\varepsilon \sum_{t=1}^{T} \left\langle p^t, f^t \right\rangle}, \end{aligned} \tag{2.6}$$

since $\Phi^1 = n$.

Our next step is to provide a lower bound on Φ^{T+1}. As before, we observe that $\Phi^t \geq w_i^t$ for any t and any i, and obtain

$$\Phi^{T+1} \geq w_i^{T+1} = \prod_{t=1}^{T} (1 - \varepsilon f_i^t).$$

Using Fact 2.2 we can further write

$$\Phi^{T+1} \geq e^{-\varepsilon \sum_{t=1}^{T} f_i^t - \varepsilon^2 \sum_{t=1}^{T} \left(f_i^t\right)^2}. \tag{2.7}$$

Putting together the lower and the upped bound on Φ^{T+1}, i.e. equations (2.7) and (2.6), and taking logarithm we obtain

$$\ln n - \varepsilon \sum_{t=1}^{T} \left\langle p^t, f^t \right\rangle \geq -\varepsilon \sum_{t=1}^{T} f_i^t - \varepsilon^2 \sum_{t=1}^{T} \left(f_i^t\right)^2,$$

which after rearranging and dividing by ε gives

$$\sum_{t=1}^{T} \left\langle p^t, f^t \right\rangle - \sum_{t=1}^{T} f_i^t \leq \frac{\ln n}{\varepsilon} + \varepsilon \sum_{t=1}^{T} \left(f_i^t\right)^2 \leq \frac{\ln n}{\varepsilon} + \varepsilon T.$$

The last inequality is true since $\|f^t\|_\infty \leq 1$. Since this is true for any i, by convexity we obtain that

$$\sum_{t=1}^{T} \left\langle p^t, f^t \right\rangle - \sum_{t=1}^{T} \left\langle p, f^t \right\rangle \leq \frac{\ln n}{\varepsilon} + \varepsilon T$$

for all $p \in \Delta_n$. This completes the proof of the theorem. □

The width. What happens if instead of $\|f^t\|_\infty \leq 1$, we have $\|f^t\|_\infty \leq \rho$ (for all t) some $\rho > 1$? This parameter is often referred to as the *width* of the loss function. In this case, to maintain the non-negativity of the weights, the update rule must be modified to

$$w_i^{t+1} \stackrel{\text{def}}{=} w_i^t \left(1 - \frac{\varepsilon f_i^t}{\rho}\right).$$

In this case, everything goes through as before except that the term $\frac{\ln n}{\varepsilon}$ in the regret bound in Theorem 2.3 becomes $\frac{\rho^2 \ln n}{\varepsilon}$. Thus, the number of iterations after which the average regret becomes less than 2ε is $\frac{\rho^2 \ln n}{\varepsilon^2}$.

2.2.3 Solving Linear Programs Approximately using MWU

In this section we illustrate one of many of the applications of the MWU algorithm: solving linear programs. Rather than solving the optimization version of linear programming, we chose a very simple setting to illustrate the powerful idea of using the MWU algorithm. The method has been used in numerous ways to speed up solutions to linear programs for fundamental problems. We consider the following feasibility problem: given an $m \times n$ matrix A, and $b \in \mathbb{R}^n$ does there exist an x such that $Ax \geq b$?

$$\exists ? x : Ax \geq b. \tag{2.8}$$

For this problem we give an algorithm that, given an error parameter $\varepsilon > 0$, either outputs a point x such that $Ax \geq b - \varepsilon \mathbb{1}$ or proves that there is no solution to this linear system of inequalities. We also assume the existence of an *oracle* that, given vector $p \in \Delta_n$, solves the following relaxed problem

$$\exists ? x : p^\top Ax \geq p^\top b. \tag{2.9}$$

Note that the above feasibility problem involves only *one* inequality. This, often, may be significantly easier algorithmically than the original feasibility problem. In any case, we will assume that. Clearly, if there is an x that satisfies (2.8), then x satisfies (2.9) as well for all $p \in \Delta_n$. On the other hand, if there is no solution to (2.9), then the system (2.8) is infeasible. Finally, we assume that when the oracle returns a feasible solution for a p, the solution x that it returns is not arbitrary but has bounded *width*:

$$\max_i |(Ax)_i - b_i| \leq 1.$$

In this setting, we can prove the following theorem:

Theorem 2.6. If there is a solution to (2.8), then there is an algorithm that outputs a x which satisfies the system (2.8) up to an additive error of 2ε. The algorithm makes at most $\frac{\ln m}{\varepsilon^2}$ calls to a width-bounded oracle for the problem 2.9. On the other hand, if there is no solution to (2.8), then the algorithm says so.

The algorithm in the proof of the theorem is the MWU: we just have to identify the *loss* function at time t. We maintain a probability distribution p^t at any time t and pass it to the oracle. As is usual, the starting probability distribution, p^1, is uniform. We pass this p^t to the oracle. If the oracle returns that the system 2.9 is infeasible, the algorithm returns that the system (2.8) is infeasible. As feasibility is preserved under taking convex combinations, we know that the algorithm is correct. On the other hand, if the oracle returns an x^t, we set the loss function to

$$f^t \stackrel{\mathrm{def}}{=} Ax^t - b.$$

Since $|(Ax^t)_i - b_i| \leq 1$, $\|f^t\|_\infty \leq 1$. Finally, if the algorithm succeeds for $T \stackrel{\mathrm{def}}{=} \frac{\ln m}{\varepsilon^2}$ iterations, then we let

$$\tilde{x} \stackrel{\mathrm{def}}{=} \frac{1}{T} \sum_t x^t.$$

Thus, to prove the theorem, it is sufficient to prove that $\tilde{x}$ satisfies (2.8) up to an additive error of 2ε.

Towards that end, we begin by observing that, since x^t is feasible for the system (2.9) for p^t for every t, we have

$$\left\langle p^t, f^t \right\rangle = \left\langle Ax^t - b, p^t \right\rangle = \left(p^t\right)^\top Ax^t - \left(p^t\right)^\top b \geq 0.$$

Thus, the loss at every step is at least 0. Hence, from Theorem 2.5, for the choice of $T = \ln m/\varepsilon^2$ we obtain that for every i, $-\frac{1}{T}\sum_{t=1}^T f_i^t \leq 2\varepsilon$. This is the same as

$$-\frac{1}{T} \sum_{t=1}^{T} \big((Ax^t)_i - b_i\big) \leq 2\varepsilon.$$

Now, using the definition of $\tilde{x} = \frac{1}{T}\sum_t x^t$, we obtain that for all i,

$$(A\tilde{x})_i \geq b_i - 2\varepsilon.$$

Thus, $\tilde{x}$ is 2ε-feasible for (2.8). This concludes the proof of Theorem 2.6.

2.3 Multiplicative Weights Update as Gradient Descent

Now we begin our journey towards using the ideas developed in the previous sections to give a new algorithm for convex optimization. In this section we first see how the MWU algorithm is in fact a form of gradient descent. The connection is via the entropy function.

For a non-negative vector w, let $H(w)$ denote the function

$$H(w) \stackrel{\text{def}}{=} \sum_i w_i \ln w_i.$$

(This function is the negative entropy when $w \in \Delta_n$.) The gradient of $H(w)$ is the vector $x(w)$ where

$$x_i = (1 + \ln w_i)_i.$$

Thus, if w varies over the non-negative orthant, $x(w)$ varies over $\mathbb{R}^n$. Moreover, this map is invertible: given $x \in \mathbb{R}^n$, one can find a w such that $x = x(w)$:

$$w_i = e^{x_i - 1}.$$

Thus, at iteration t, for w^t we let $x^t \stackrel{\text{def}}{=} x(w^t)$. The loss function is f^t, and the loss is $F^t(w) \stackrel{\text{def}}{=} \langle f^t, w \rangle$. Thus, $\nabla_w F^t = f^t$. Thus, our assumption that $\|f^t\|_\infty \leq 1$ is the same as

$$\|\nabla_w F^t\|_\infty \leq 1.$$

Suppose we take the gradient step of size η in the x-space with respect to $\nabla_w F^t$. Then, we obtain a new point in the x-space:

$$x^{t+1} = x^t - \eta \nabla_w F^t = x^t - \eta f^t.$$

The corresponding w^{t+1} is obtained by equating, for each i,

$$1 + \ln w_i^{t+1} = 1 + \ln w_i^t - \eta f_i^t.$$

Exponentiating, the update step in the w-space is

$$w_i^{t+1} = w_i^t e^{-\eta f_i^t}.$$

While this is not quite the update in the MWU algorithm, it is the *Hedge* variant of the MWU. For these updates we can derive the same regret bound as Theorem 2.5. It is the same as the MWU algorithm when $\eta f_i^t \ll 1$ for all t, i since we can then use the approximation $e^{-x} \approx 1 - x$, giving us the familiar update step:

$$w_i^{t+1} = w_i^t (1 - \eta f_i^t).$$

Thus, if we allow going back and forth to a space via the gradient, there is a function, namely the negative entropy function, for which the MWU algorithm is nothing but a gradient-descent type method! How general is this paradigm?

2.3.1 A Gradient Descent Method Inspired from MWU

It turns out that there is a method for convex optimization that can be abstracted out from the previous section which works well beyond the MWU setting. As we mentioned in the beginning of this lecture, the method we will derive could give better convergence rate than doing the usual gradient descent in the ambient space. We are back in the setting where we are given a convex function f and a convex set $K \subseteq K'$, and the goal is to find a point $x \in K$ such that

$$f(x) - f(x^{\star}) \leq \varepsilon.$$

(The reader can keep in mind $K = \Delta_n$ and $K' = \mathbb{R}^n_{\geq 0}$ to draw the analogy with the previous section.) Assume that the gradient of f is bounded by 1 with respect to some norm $\|\cdot\|$. (This corresponds to the assumption $\|f^t\|_\infty \leq 1$ for all t in the MWU setting.) Just like we chose the negative entropy map H in the previous section, the method depends on a map M which is assumed to be continuously differentiable and strictly convex function M over K'. Further, similar to H, the map M should have additional properties (we do not spell them all out): the map $\nabla M : K' \mapsto S$ should be one-to-one and invertible. ($S = \mathbb{R}^n$ in the previous section.) We can now generalize the algorithm in the previous section.

The initial point x^1 is chosen to be

$$x^1 \stackrel{\text{def}}{=} \arg\inf_{x \in K} M(x).$$

(For negative entropy, this results in the choice of the vector where all the coordinates are the same, i.e., the uniform distribution over the experts or the vector $\frac{1}{n}\nVdash$). Let $x^t \in K$. Once we map x^t to $\nabla M(x^t)$ and do the gradient step, we may move out of K but should remain in K'. Let y^{t+1} be the point in K' which corresponds to the resulting point in S. In other words, given x^t and ∇f, for $t \geq 1$ we let y^{t+1} be the point such that

$$\nabla M\left(y^{t+1}\right) = \nabla M\left(x^t\right) - \eta \nabla f\left(x^t\right). \tag{2.10}$$

As y^{t+1} might lie outside of K, we project it back to K and let x^{t+1} be its projection. We obtain the projection by minimizing the Bregman divergence as follows

$$x^{t+1} \stackrel{\text{def}}{=} \arg \inf_{x \in K} D_M\left(x, y^{t+1}\right).$$

Recall that the *Bregman divergence* $D_M(x, y)$ associated with M for points $x, y \in K'$ is defined to be

$$D_M(x, y) \stackrel{\text{def}}{=} M(x) - M(y) - \langle \nabla M(y), x - y \rangle.$$

(In the MWU setting, the Bregman divergence of the negative entropy function corresponds to the Kullback-Liebler divergence and projecting according to it is nothing but normalizing a non-negative vector to make its ℓ_1 norm 1.) In this setting, we can prove the following theorem which is a generalization the MWU method (and from which we can recover the MWU method):

Theorem 2.7. Let the gradient of f be bounded in a norm $\|\cdot\|$ by G. Let M be a map satisfying the properties listed above and, in addition, is l-strongly convex with respect to norm $\|\cdot\|_\star$. Let $D \stackrel{\text{def}}{=} \sup_{x \in K} M(x) - M(x^1)$. Then, for $\eta \stackrel{\text{def}}{=} \frac{D}{\sqrt{T}}$, the algorithm above produces a sequence of points $x_1, \ldots, x_T$ such that

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

where

$$T = \frac{1}{l} \cdot \left(\frac{DG}{\varepsilon}\right)^2.$$

To complete the analogy to the MWU method in the previous section, we note that if we let $x^1 \stackrel{\text{def}}{=} \frac{1}{n}\mathbb{1}$, then $D = \sqrt{\ln n}$. Further, Pinsker's inequality implies that the negative entropy function is 1-strongly convex with respect to the $\|\cdot\|_1$ norm, the dual norm to $\|\cdot\|_\infty$. The proof of this theorem is left as an exercise. No new idea, other than those discussed in this lecture and the last are required to complete the proof.

Thus, we have achieved the goal promised in the beginning of the lecture. In conclusion, we started with the MWU algorithm in the online convex optimization setting, interpreted it as a gradient descent method and then abstracted the essence to obtain a new method in convex optimization. This method is often referred to as *Mirror Descent* or *Dual Averaging* algorithm. In the appendix we present a matrix version of the MWU method which has applications to approximately solving SDPs quickly.

2.4 Appendix: A Multiplicative Weights Update for Matrices

In this section we discuss the generalization of our MWU framework to the matrix world. As before we have n experts. So far, the decision for round t was essentially the index i of the expert we follow in this round. After this decision is made, the adversary reveals the cost vector $f^t \in [-1,1]^n$, and we pay f_i^t. This is equivalent to choosing on every round t a vector $v^t \in \{e_1, e_2, ..., e_n\}$ and having loss $\langle v^t, f^t \rangle$. Recall that the strategy we devised for the game was randomized and our v^t was a random vector chosen according to the distribution $P[v^t = e_i] = p_i^t$, where p^t was the probability distribution at round t. Then the expected loss was simply $\sum_i p_i f_i^t = \langle p^t, f^t \rangle$.

In the matrix case we allow the decision v^t to be any vector of ℓ_2-norm 1 (i.e. vector from unit sphere $\mathbb{S}^{n-1}$). The loss at round t is given by

$$\langle v^t, F^t v^t \rangle = (v^t)^\top F v^t,$$

where F^t is the loss matrix chosen by the adversary right after our decision at round t. We assume that this matrix is symmetric. Similarly to the basic case, where we had the requirement $|f_i^t| \leq 1$, we impose a bound on the cost matrix: $\|F^t\| \leq 1$ ($\|\cdot\|$ is the spectral norm, i.e. we require that the eigenvalues of F^t lie all in the interval $[-1,1]$). Observe that in our previous setting all the loss matrices were diagonal and we could pick our decision only from a finite set $\{e_1, e_2, ..., e_n\} \subseteq \mathbb{S}^{n-1}$. Like before we will give a randomized strategy for this online problem. Our goal is, of course, to minimize the expected total loss comparing to the best expert. However, now the set of experts is much bigger, it consists of all the unit length vectors.

Suppose our algorithm chooses the vector $v \stackrel{\text{def}}{=} v^t$ according to distribution $\mu \stackrel{\text{def}}{=} \mu^t$ and the cost matrix is $F \stackrel{\text{def}}{=} F^t$. We calculate the loss at round t:

$$\mathbb{E}_{v\sim\mu}[\langle v, Fv\rangle] = \mathbb{E}_{v\sim\mu}[(vv^T) \bullet F] = \mathbb{E}_{v\sim\mu}[vv^T] \bullet F$$

where $A \bullet B \overset{\text{def}}{=} \mathrm{Tr}(A^T B)$ is the usual matrix inner product. So if we denote $P^t = \mathbb{E}_{v\sim\mu}[vv^T]$, then the loss at round t is simply $P^t \bullet F^t$. Observe that P^t is PSD and $\mathrm{Tr}(P^t) = 1$. Our algorithm, instead of working directly with a distribution over $\mathbb{S}^{n-1}$, will keep a matrix P^t. Our objective is to ensure that $\sum_{t=1}^T P^t \bullet F_t$ as small as possible, compared to $\min_{\|w\|_2=1} \sum_{t=1}^T w^T F^t w$. The latter is nothing but the smallest eigenvalue of $\sum_{t=1}^T F^t$. We proceed with the description of our algorithm based on MWU.

Matrix Multiplicative Weights Update (MMWU):

(1) Initialize $Q^1 = I$.

(2) In round t, use the matrix

$$P^t = \frac{Q^t}{\Phi^t},$$

where $\Phi^t \overset{\text{def}}{=} \mathrm{Tr}(Q^t)$.

(3) Observe the loss matrix F^t, and update Q^t as follows

$$Q^{t+1} = e^{-\varepsilon \sum_{k=1}^t F^k}.$$

where: $e^A \overset{\text{def}}{=} \sum_{k=0}^\infty \frac{A^k}{k!}$

We will now prove an analogue of Theorem 2.5 in the matrix world.

Theorem 2.8. Assume that $-I \preceq F^t \preceq I$ for every t, and let $0 < \varepsilon \leq 1$. Then, the algorithm MMWU produces a sequence of matrices $P^1, \ldots, P^T$ with $\mathrm{Tr}(P^t) = 1$ such that:

$$\sum_{t=1}^T P^t \bullet F^t - \inf_{v\in\mathbb{S}^{n-1}} \sum_{t=1}^T v^T F^t v \leq \frac{\ln n}{\varepsilon} + \varepsilon T. \tag{2.11}$$

Thus, after $T = \frac{\ln n}{\varepsilon^2}$,

$$\frac{1}{T}\sum_{t=1}^{T} P^t \bullet F^t \leq \lambda_{\min}\left(\frac{1}{T}\sum_{t=1}^{T} F^t\right) + 2\varepsilon.$$

Thus, if we know that $P^t \bullet F^t \geq 0$ for all t, then

$$-2\varepsilon I \preceq \frac{1}{T}\sum_{t=1}^{T} F^t.$$

This is a powerful implication and has far reaching consequences when one carefully constructs the setting, just as in Section 2.2.3. For instance, this theorem is sufficient to construct fast and approximate SDP solvers and show the existence of near-linear spectral sparsifiers. We omit the details.

The proof has the same structure as the proof of Theorem 2.5. However, in the matrix world where we are plagued with non-commutativty: $AB \neq BA$ in general for matrices. For instance it would have been great if $e^{A+B} = e^A e^B$. But this is false in general. What is true instead is the following fact which is known as the Golden-Thompson inequality:

Fact 2.9. Let A, B be symmetric matrices, then:

$$\operatorname{Tr}\left(e^{A+B}\right) \leq \operatorname{Tr}\left(e^A e^B\right).$$

We note that this is not true for 3 matrices A, B, C! We also need some simple inequalities necessary in the proof which essentially are a consequence of the corresponding scalar inequalities.

Fact 2.10. Let A, B, C be symmetric matrices, let A be PSD and $B \preceq C$, then:

$$\operatorname{Tr}(AB) \leq \operatorname{Tr}(AC).$$

Fact 2.11. Let A be a symmetric matrix and v be a vector of unit length, then:

$$e^{v^\top Av} \leq \mathrm{Tr}\left(e^A\right).$$

Proof. Suppose the eigenvalues of A are $\lambda_1 \leq \cdots \leq \lambda_n$. The eigenvalues of e^A are precisely $e^{\lambda_1}, \ldots, e^{\lambda_n}$. We know that $v^\top Av \leq \lambda_n$. Therefore $e^{v^\top Av} \leq e^{\lambda_n} \leq \mathrm{Tr}\left(e^A\right)$. □

Fact 2.12. Let A be a symmetric matrix satisfying $\|A\| \leq 1$, then:

$$e^{-A} \preceq I - A + A^2.$$

This is easy to see because the eigenspaces of all the matrices $A, A^2, A^3, ..., e^A$ are the same, so it is enough to prove the above inequality for A being a number in the interval $[-1, 1]$. Now we are ready to prove the theorem.

Proof. As in the proof of Theorem 2.5 we try to obtain upper and lower bounds on the potential function Φ^{t+1}, which in our case is defined to be $\Phi^{t+1} \stackrel{\text{def}}{=} \mathrm{Tr}(Q^{t+1})$. Let us start with the upper bound:

$$\begin{aligned}
\Phi^{t+1} &= \mathrm{Tr}(Q^{t+1}) = \mathrm{Tr}\left(e^{-\varepsilon\sum_{k=1}^{t} F^k}\right) \\
&\overset{G-T}{\leq} \mathrm{Tr}\left(e^{-\varepsilon\sum_{k=1}^{t-1} F^k} e^{-\varepsilon F^t}\right) \\
&\leq \mathrm{Tr}\left(e^{-\varepsilon\sum_{k=1}^{t-1} F^k}\left(I - \varepsilon F^t + \varepsilon^2 (F^t)^2\right)\right) \\
&= \mathrm{Tr}\left(Q^t\right) - \varepsilon \mathrm{Tr}\left(Q^t F^t\right) + \varepsilon^2 \mathrm{Tr}\left(Q^t (F^t)^2\right).
\end{aligned}$$

Now we use the fact that $Q^t = P^t \mathrm{Tr}(Q^t)$:

$$\mathrm{Tr}\left(Q^t F^t\right) = \mathrm{Tr}(Q^t)\mathrm{Tr}\left(P^t F^t\right) = \Phi^t (P^t \bullet F^t)$$

Similarly:

$$\mathrm{Tr}\left(Q^t (F^t)^2\right) = \mathrm{Tr}(Q^t)\mathrm{Tr}\left(P^t F^t\right) = \Phi^t (P^t \bullet (F^t)^2).$$

Therefore we can conclude that:

$$\Phi^{t+1} \leq \Phi^t(1 - \varepsilon(P^t \bullet F^t) + \varepsilon^2(P^t \bullet (F^t)^2)) \leq \Phi^t e^{-\varepsilon(P^t \bullet F^t) + \varepsilon^2(P^t \bullet (F^t)^2)}.$$

Expanding further,

$$\Phi^{t+1} \leq ne^{-\varepsilon \sum_{k=1}^t (P^k \bullet F^k) + \varepsilon^2 \sum_{k=1}^t (P^k \bullet (F^k)^2)}. \tag{2.12}$$

It remains to obtain a suitable lower bound for Φ^{t+1}. To this end let us fix any vector v with $\|v\|_2 = 1$. We use Fact 2.11 with $A = -\varepsilon \sum_{k=1}^t F^k$:

$$e^{-\varepsilon \sum_{k=1}^t v^\top F^k v} \leq \operatorname{Tr}\left(e^{-\varepsilon \sum_{k=1}^t F^k}\right) = \Phi^{t+1}. \tag{2.13}$$

Putting inequalities 2.12 and 2.13 together and taking logarithms of both sides yields

$$-\varepsilon \sum_{k=1}^t v^\top F^k v \leq -\varepsilon \sum_{k=1}^t (P^k \bullet F^k) + \varepsilon^2 \sum_{k=1}^t (P^k \bullet (F^k)^2) + \ln n.$$

After dividing by ε and rearranging, we get for every $v \in \mathbb{S}^{n-1}$:

$$\sum_{k=1}^t (P^k \bullet F^k) \leq \sum_{k=1}^t v^T F^k v + \varepsilon \sum_{k=1}^t (P^k \bullet (F^k)^2) + \frac{\ln n}{\varepsilon}.$$

□

3

Newton's Method and the Interior Point Method

3.1 Overview

In the last of the three foundational lectures we continue our journey towards faster (and better) algorithms for convex programs. The algorithms introduced till now assume access only to an oracle for the value of the convex function and its gradient at a specified point. In this lecture we assume that we are also given a *second order* access to f: namely, given vectors x and y, we could obtain $(\nabla^2 f(x))^{-1}y$. The resulting method would be Newton's method for solving unconstrained programming and will have this property that if one starts close enough to the optimal solution the convergence would be in $\log\log {}^1\!/\!\varepsilon$ iterations! Finally, we present the application of Newton's method to solving constrained convex programs. This is achieved by moving from constrained to unconstrained optimization via a *barrier* function. The resulting methods are broadly termed as interior point methods. We analyze one such method, referred to as the *primal path-following interior point method*, for linear programs. We end this lecture by a discussion on *self-concordant* barrier functions which allow us to go beyond linear programs to more general convex programs.

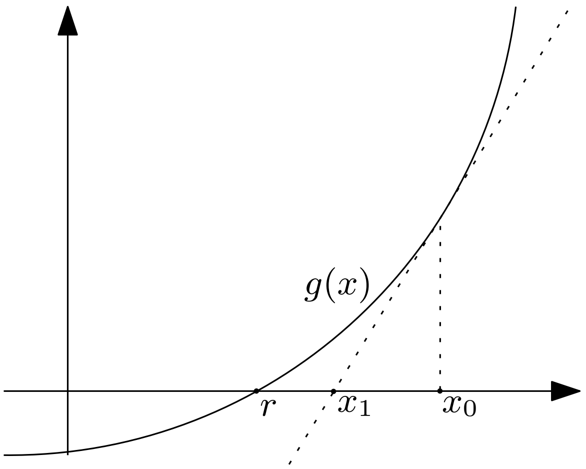

Fig. 3.1 One step of Newton's Method

3.2 Newton's Method and its Quadratic Convergence

Our starting point is the versatile Newton's method which we first explain in the simplest setting of finding a root for a univariate polynomial.

3.2.1 Newton-Raphson method

In numerical analysis, Newton's method (also known as the Newton-Raphson method), named after Isaac Newton and Joseph Raphson, is a method for finding successively better approximations to the roots (or zeroes) of a real-valued function. Suppose we are given a function $g : \mathbb{R} \mapsto \mathbb{R}$ and we want to find its root (or one of its roots). Assume we are given a point x_0 which is likely to be close to a zero of g. We consider the point $(x_0, g(x_0))$ and draw a line through it which is tangent to the graph of g. Let x_1 be the intersection of the line with the x-axis (see Figure 3.2.1). Then it is reasonable (at least if one were to believe the figure above) to suppose that by moving from x_0 to x_1 we have made progress in reaching a zero of g. First note that:

$$x_1 \overset{\text{def}}{=} x_0 - \frac{g(x_0)}{g'(x_0)}$$

From x_1, by the same method we obtain x_2, then x_3 etc. Hence, the general formula is:

$$x_{k+1} \overset{\text{def}}{=} x_k - \frac{g(x_k)}{g'(x_k)} \qquad \text{for all } k \geq 0. \tag{3.1}$$

Of course we require differentiability of g, in fact we will assume even more – that g is twice continuously differentiable. Let us now analyze how fast the distance to the root decreases with k.

Let r, be the root of g, that is $g(r) = 0$. Expand g into Taylor series at the point x_k and, use the Mean Value Theorem, to obtain the following:

$$g(r) = g(x_k) + (r - x_k)g'(x_k) + \frac{1}{2}(r - x_k)^2 g''(\theta)$$

for some θ in the interval $[r, x_k]$. From (3.1) we know that

$$g(x_k) = g'(x_k)(x_k - x_{k+1}).$$

Recall also that $g(r) = 0$. Hence, we get:

$$0 = g'(x_k)(x_k - x_{k+1}) + (r - x_k)g'(x_k) + \frac{1}{2}(r - x_k)^2 g''(\theta)$$

which implies that

$$g'(x_k)(r - x_{k+1}) = \frac{1}{2}(r - x_k)^2 g''(\theta).$$

This gives us the relation between the new distance from the root in terms of the old distance from it:

$$|r - x_{k+1}| = \left|\frac{g''(\theta)}{2g'(x_k)}\right| |r - x_k|^2.$$

This can be summarized in the following theorem:

Theorem 3.1. Suppose $g : \mathbb{R} \mapsto \mathbb{R}$ is a $\mathcal{C}^2$ function, [1] $r \in \mathbb{R}$ is a root of g, $x_0 \in \mathbb{R}$ is a starting point and $x_1 = x_0 - \frac{g(x_0)}{g'(x_0)}$, then:

$$|r - x_1| \leq M|r - x_0|^2$$

where $M = \sup_{x \in [r,x_0]} \left| \frac{g''(\theta)}{2g'(x)} \right|$.

Thus, assuming that M is a small constant, say $M \leq 1$ (and remains so throughout the execution of this method) and that $|x_0 - r| < 1$, we obtain *quadratically* fast convergence of x_k to r. For the error $|x_k - r|$ to became less then ε one needs to take $k = \log\log {}^1\!/\!_\varepsilon$. As one can imagine, for this reason Newton's Method is very efficient and powerful. In practice, it gives very good results even when no reasonable bounds on M or $|x_0 - r|$ are available.

3.2.2 Newton's Method for Convex Optimization

How could the benign looking Newton-Raphson method be useful to solve convex programs? The key lies in the observation from the first lecture that the task of minimization of a differentiable convex function in the unconstrained setting is equivalent to *finding a root of its derivative.* In this section we abstract out the method from the previous section and present Newton's method for convex programming.

Recall that the problem is to find

$$x_\star \stackrel{\text{def}}{=} \arg \inf_{x \in \mathbb{R}^n} f(x).$$

where f is a convex (smooth enough) function. The gradient ∇f of f is a function $\mathbb{R}^n \mapsto \mathbb{R}^n$ and its derivative $\nabla^2 f$ maps $\mathbb{R}^n$ to $n \times n$ symmetric matrices. Hence, the right analog of the update formula (3.1) to our setting can be immediately seen to be:

$$x_{k+1} \stackrel{\text{def}}{=} x_k - (\nabla^2 f(x_k))^{-1} \nabla f(x_k) \qquad \text{for all } k \geq 0. \tag{3.2}$$

For notational convenience we define *the Newton step* at point x to be

$$n(x) \stackrel{\text{def}}{=} -(\nabla^2 f(x))^{-1} \nabla f(x),$$

[1] The function is twice differentiable and the second derivative is continuous.

then (3.2) gets abbreviated to $x_{k+1} = x_k + n(x_k)$. One may convince themselves that (3.2) is meaningful by applying it to f being a strictly convex quadratic function (i.e. $f(x) = x^\top Mx$ for M positive definite). Then, no matter which point we start, after one iteration we land in the unique minimizer. This phenomenon can be explained as follows: suppose $\tilde{f}$ is the second order approximation of f at point x,

$$\tilde{f}(y) = f(x) + (y-x)^\top \nabla f(x) + \frac{1}{2}(y-x)^\top \nabla^2 f(x)(y-x)$$

If f is sctrictly convex then its Hessian is positive definite, hence the minimizer of $\tilde{f}(y)$ is

$$y^\star = x - (\nabla^2 f(x))^{-1}\nabla f(x) = x + n(x)$$

For this reason Newton's method is called a second-order method, because it takes advantage of the second order approximation of a function to make a step towards the minimum. All the algorithms we looked at in the previous lecture were first-order methods. They used only the gradient (first order approximation) to perform steps. However, computationally our task has increased as now we would need a second order oracle to the function: given x and y, we would need to solve the system of equations $\nabla^2 f(x)y = \nabla f(x)$.

The next question is if, and, under what conditions x_k converges to the minimizer of f. It turns out that it is possible to obtain a similar quadratic convergence guarantee assuming that x_0 is sufficiently close to the minimizer $x^\star$. We can prove the following theorem whose hypothesis and implications should be compared to Theorem 3.1.

Theorem 3.2. Let $f : \mathbb{R}^n \mapsto \mathbb{R}$ be a $\mathcal{C}^2$ function and $x^\star$ be its minimizer. Denote the gradient $\nabla^2 f(x)$ by $H(x)$ and assume that the following hold:

- There is some constant $h > 0$ and a ball $B(x^\star, r)$ around $x^\star$ such that, whenever $x \in B(x^\star, r)$, $\|H(x)^{-1}\| \leq \frac{1}{h}$.
- There is some constant $L > 0$ and a ball $B(x^\star, r)$ around $x^\star$ such that, whenever $x, y \in B(x^\star, r)$, $\|H(x) - H(y)\| \leq L\|x - y\|$.

If x_0 is a starting point, sufficiently close to $x^\star$ and $x_1 = x_0 + n(x)$ then:

$$\|x_1 - x^\star\| \leq M\|x_0 - x^\star\|^2$$

for some constant M. For example $M = \frac{L}{2h}$ will do.

Thus, if $M \leq 1$, then with a starting point close enough to the optimal solution, Newton's method converges quadratically fast. One can present a rough analogy of this theorem with Theorem 3.1. There, for the method to have quadratic convergence, $|g'(x)|$ should be bigger in comparison to $|g''(x)|$ (to end up with small M). Here, the role of g is played by the derivative f' of f (the gradient in the one-dimensional case). The first condition on $H(x)$ says basically that $|f''(x)|$ is big. The second condition may be a bit more tricky to decipher, it says that $f''(x)$ is Lipschitz-continuous, and upper-bounds the Lipschitz constant. Assuming f is thrice continuously differentiable, this essentially gives an upper bound on $|f'''(x)|$. Note that this intuitive explanation does not make any formal sense, since $f'(x), f''(x), f'''(x)$ are not numbers, but vectors, matrices and 3-tensors respectively. We only wanted to emphasize that the spirit of Theorem 3.2 still remains the same as Theorem 3.1.

Proof. [Proof of Theorem 3.2] The basic idea of the proof is the same as in 3.1. We need a similar tool as the Taylor expansion used in the previous chapter. To obtain such, we consider the function $\phi : [0,1] \to \mathbb{R}^n$, $\phi(t) = \nabla f(x + t(y-x))$. Applying the fundamental theorem of calculus to ϕ (to every coordinate separately) yields:

$$\begin{aligned} \phi(1) - \phi(0) &= \int_0^1 \nabla\phi(t)dt \\ \nabla f(y) - \nabla f(x) &= \int_0^1 H(x + t(y-x))(x-y)dt. \qquad (3.3) \end{aligned}$$

Let $x = x_0$ for notational convenience and write $x_1 - x^\star$ in a convenient form:

$$\begin{aligned}
x_1 - x^\star &= x - x^\star + n(x) \\
&= x - x^\star - H(x)^{-1}\nabla f(x) \\
&= x - x^\star + H(x)^{-1}(\nabla f(x^\star) - \nabla f(x)) \\
&= x - x^\star + H(x)^{-1}\int_0^1 H(x + t(x^\star - x))(x - x^\star)dt \\
&= H(x)^{-1}\int_0^1 (H(x + t(x^\star - x)) - H(x))(x - x^\star)dt.
\end{aligned}$$

Now take norms:

$$\begin{aligned}
\|x_1 - x^\star\| &\leq \|H(x)^{-1}\| \int_0^1 \|(H(x + t(x^\star - x)) - H(x))(x - x^\star)\| dt \\
&\leq \|H(x)^{-1}\| \|x - x^\star\| \int_0^1 \|(H(x + t(x^\star - x)) - H(x))\| dt.
\end{aligned} \tag{3.4}$$

We use the Lipschitz condition on H to bound the integral:

$$\begin{aligned}
\int_0^1 \|(H(x + t(x^\star - x)) - H(x))\| dt &\leq \int_0^1 L\|t(x^\star - x)\| dt \\
&\leq L\|x^\star - x\| \int_0^1 t dt \\
&= \frac{L}{2}\|x^\star - x\|.
\end{aligned}$$

Together with (3.4) this implies:

$$\|x_1 - x^\star\| \leq \frac{L\|H(x)^{-1}\|}{2}\|x^\star - x\|^2 \tag{3.5}$$

which completes the proof. We can take $M = \frac{L\|H(x)^{-1}\|}{2} \leq \frac{L}{2h}$. □

3.3 Constrained Convex Optimization via Barriers

In this section return to constrained convex optimization problems of the form:

$$\inf_{x \in K} f(x) \tag{3.6}$$

where f is a convex, real valued function and $K \subseteq \mathbb{R}^n$ is a convex set. [2] In the first lecture we discussed how gradient-descent type methods could be adapted in this setting by projecting onto K at every step. We took an improvement step $x_k \mapsto x'_{k+1}$ with respect to f and then we projected x'_{k+1} onto K to obtain x_{k+1}. There are a few problems with this method. One of them is that in most cases computing projections is prohibitively hard. Furthermore, even if we ignore this issue, the projection-based methods are not quiet efficient. To get an ε-approximation to the solution, the number of iterations depends polynomially on ε^{-1} (i.e. the number of iterations is proportional to $1/\varepsilon^{O(1)}$). Unfortunately such dependence on ε is often unsatisfactory. For example, to obtain the optimal solution for a linear program we need to take ε of the form 2^{-L}, where L is the size of the instance.[3] Hence, to get a polynomial time algorithm, we need the running time dependence on ε to be $\log^{O(1)}(1/\varepsilon)$. Today we will see an interior point algorithm which achieve such a guarantee.

3.3.1 Following the Central Path

We are going to present one very general idea for solving constrained optimization problems. Recall that our aim is to minimize a given convex function $f(x)$ subject to $x \in K$. To simplify our discussion, we assume that the objective function is linear, [4] i.e. $f(x) = c^\top x$ and the convex body K is bounded and full-dimensional (it has positive volume).

Suppose we have a point $x_0 \in K$ and we want to perform an improvement step maintaining the condition of being inside K. The simplest idea would be to move in the direction $-c$ to decrease our objective value as much as possible. Our step will then end up on the boundary of K. The second and further points would lie very close to the

[2] K is given to us either explicitly – by a collection of constraints defining it, or by a separation oracle.

[3] One should think of L as the total length of all the binary encodings of numbers in the description of the linear program. In the linear programming setting, $L^{O(1)}$ can be shown to bound the number of binary digits required to represent the coordinates of the optimal solution.

[4] Actually we are not losing on generality here. Every convex problem can be stated equivalently with a linear objective function.

boundary, which will force our steps to be short and thus inefficient. In case of K being a polytope, such a method would be equivalent to the simplex algorithm, which as known to have an exponential worst case running time. For this reason we need to set some force, which would repel us from the boundary of K. More formally, we want to move our constraints to the objective function and consider $c^\top x + F(x)$,[5] where $F(x)$ can be regarded as a "fee" for violating constraints. $F(x)$ should become big for x close to ∂K. Of course, if we would like the methods developed in the previous sections for unconstrained optimization to be applicable here, we would also like f to be strongly convex. Thus, this is another route we could take to convert a constrained minimization problem into unconstrained minimization, but with a slightly altered objective function. To formalize this approach we introduce the notion of a *Barrier Function.* Instead of giving a precise definition, we list some properties, which we wish to hold for a barrier function F:

- F is defined in the interior of K, i.e. $\mathrm{dom}(F) = \mathrm{int}(K)$, (3.7)
- for every point $b \in \partial K$ we have: $\lim_{x \to b} F(x) = +\infty$, (3.8)
- F is strictly convex. (3.9)

Suppose F is such a barrier function, let us define a perturbed objective function f_η, where $\eta > 0$ is a real parameter:

$$f_\eta(x) \stackrel{\text{def}}{=} \eta c^\top x + F(x) \tag{3.10}$$

We may imagine that f_η is defined on all of $\mathbb{R}^n$ but attains finite values only on $\mathrm{int}(K)$. Intuitively, making η bigger and bigger reduces the influence of $F(x)$ on the optimal value of $f_\eta(x)$. Furthermore, observe that since $c^\top x$ is a linear function, the second order behavior of f_η is completely determined by F, that is $\nabla^2 f_\eta = \nabla^2 F$. In particular, f_η is strictly convex and it has a unique minimizer $x^\star_\eta$. The set

$$\{x^\star_\eta : \eta \geq 0\}$$

[5] One seemingly perfect choice of F would be a function which is 0 on K and $+\infty$ on the complement of K. This reduces our problem to unconstrained minimization of $c^\top x + F(x)$. However, note that we have not gained anything by this reformulation, even worse: our objective is not continuos anymore.

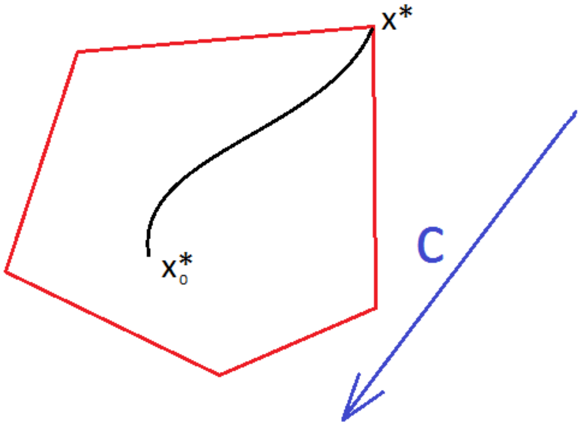

Fig. 3.2 Example of a central path

can be seen to be continuous due to the Implicit Function Theorem and is referred to as a *central path* starting at $x_0^\star$ and approaching $x^\star$ – the solution to our convex problem 3.6. In other words:

$$\lim_{\eta \to \infty} x_\eta^\star = x^\star.$$

A method which follows this general approach is called a *path-following interior point method.* Now, the key question is: "how fast is this convergence?". Needless to say, this would depend on the choice of the barrier function and the method for solving the unconstrained optimization problem. We answer it in the next section for linear programs: using the logarithmic barrier function along with Newton's method from the previous section we can give an algorithm to solve linear programs in time polynomial in the encoding length.

3.4 Interior Point Method for Linear Programming

3.4.1 A Brief History of IPMs

Interior point methods (IPMs), as alluded to in the previous section, first appear in a recognizable form in the Ph.D. thesis of Ilya Dikin in the 60s. However, there was no analysis for the time needed for convergence. In 1984 Narendra Karmarkar announced a polynomial-time algorithm to solve linear programs using an IPM. At that point there was already a known polynomial time algorithm for solving LPs, namely the Ellipsoid Algorithm from 1979. Further, the method of choice in practice, despite known to be inefficient in the worst case, was the Simplex method. However, in his paper, he also presented empirical results which showed that his algorithm was consistently 50 times faster than the simplex method. This event, which received publicity around the world throughout the popular press and media, marks the beginning of the interior-point revolution. For a nice historical perspective on this revolution, we refer the reader to the survey of Wright [Wri05].

Karmarkar's algorithm needed roughly $O(m \log {}^1\!/\!_\varepsilon)$ iterations to find a solution (which is optimal up to an additive error of ε) to a linear program, where m is the number of constraints. Each such iteration involved solving a linear system of equations, which could be easily performed in polynomial time. Thanks to the logarithmic dependence on the error ε it can be used to find the exact, optimal solution to a linear program in time polynomial with respect to the encoding size of the problem. Thus, Karmarkar's algorithm was the first efficient algorithm with provable worst case polynomial running time. Subsequently, James Renegar proposed an interior point algorithm with reduced number of iterations: $O(\sqrt{m}\log({}^1\!/\!_\varepsilon))$. Around the same time, Nesterov and Nemirovski abstracted out the essence of interior point methods and came up with the the notion of *self-concordance*, which in turn was used to provide efficient, polynomial time algorithms for many nonlinear convex problems such as semi-definite programs.

We begin our discussion by introducing the logarithmic barrier function and then Newton's Method, which is at a core of all interior-point algorithms. Then we present Renegar's primal path following method. We give a full analysis of this algorithm, i.e. we prove that after $\widetilde{O}(\sqrt{m}\log(1/\varepsilon))$ it outputs a solution which is ε-close to the optimum.

3.4.2 The Logarithmic Barrier

In this section we switch from a general discussion on constrained convex optimization to a particular problem: linear programming. We will use ideas from the previous section to obtain an efficient algorithm for solving linear programs in the form:

$$\begin{aligned} &\min_{x\in\mathbb{R}^n} c^\top x \\ &\text{s.t. } Ax \leq b \end{aligned} \tag{3.11}$$

where x is the vector of variables of length n, $c \in \mathbb{R}^n$, $b \in \mathbb{R}^m$ and $A \in \mathbb{R}^{m\times n}$. We will denote the rows of A by $a_1, a_2, \ldots, a_m$ (but treat them as column vectors). We will assume that the set of constraints $Ax \leq b$ defines a bounded polytope P of nonzero volume in $\mathbb{R}^n$.[6]

We are going to use the following barrier function which is often called the *logarithmic barrier* function:

$$F(x) \stackrel{\text{def}}{=} -\sum_{i=1}^{m} \log(b_i - a_i^\top x)$$

It is easy to see that $F(x)$ is well defined on $\mathrm{int}(P)$ and tends to infinity when approaching the boundary of P. Let us now write down formulas for the first and second derivative of F, they will prove useful a couple of times in the analysis. To simplify the notation, we will often write $s_i(x)$ for $b_i - a_i^\top x$.

Fact 3.3. If x is a point in the interior of P, then:

(1) $\nabla F(x) = \sum_{i=1}^m \frac{a_i}{s_i(x)}$
(2) $\nabla^2 F(x) = \sum_{i=1}^m \frac{a_i a_i^\top}{s_i(x)^2}$

Using the above formulas we can investigate the convexity of F. It is clear that $\nabla^2 F(x) \succeq 0$, which implies convexity. Strong convexity of F

[6] Of course this is not always the case for general linear programs, however if the program is feasible then we can perturb the constraints by an exponentially small $\varepsilon > 0$ to force the feasible set to be full dimensional. Moreover, it will turn out soon that infeasibility is not a serious issue.

(i.e. $\nabla^2 F(x) \succ 0$) is equivalent to the fact that $a_1, a_2, \ldots, a_n$ span the whole $\mathbb{R}^n$, in our case this is true, because of the assumption that P is full dimensional and bounded. Which is which should be clear from the context.

From now on, whenever we talk about a function F we will write $H(x)$ for its Hessian $\nabla^2 F(x)$ and $g(x)$ for the gradient $\nabla F(x)$. Note however that at some places in the text we refer to F as a general function and sometimes we fix F to be a specific one: the logarithmic barrier.

3.4.3 Local Norms

In this section we introduce an important concept of a local norm. It plays a crucial role in understanding interior point methods. Rather than working with the Euclidean norm, the analysis of our algorithm will be based on bounding the local norm of the Newton step. Let $A \in S^n$ be a positive definite matrix, we associate with it the inner product $\langle \cdot, \cdot \rangle_A$ defined as:

$$\langle x, y \rangle_A \stackrel{\text{def}}{=} x^\top A y$$

and the norm $\| \cdot \|_A$:

$$\|x\|_A \stackrel{\text{def}}{=} \sqrt{x^\top A x}$$

Note that a ball of radius 1 with respect to such a norm corresponds to an ellipsoid in the Euclidean space. Formally, we define the ellipsoid associated with the matrix A, centered at $x_0 \in \mathbb{R}^n$ as:

$$\mathcal{E}_{x_0}(A) \stackrel{\text{def}}{=} \{x : (x - x_0)^\top A (x - x_0) \leq 1\}.$$

Matrices of particular interest are for us Hessians of strictly convex functions $F : \mathbb{R}^n \mapsto \mathbb{R}$ (such as the logarithmic barrier). For them we usually write in short:

$$\|z\|_x \stackrel{\text{def}}{=} \|z\|_{H(x)}$$

whenever the function F is clear from the context. This is called a local norm at x with respect to F. From now on, let $F(x) = \sum_{i=1}^m -\log(b_i - a_i^\top x)$ be the logarithmic barrier function. For the logarithmic barrier, $\mathcal{E}_x(\nabla^2 F(x))$ is called the Dikin Ellipsoid centered at x. An important property of this ellipsoid is that it is contained in P and that the Hessian does not change much. Thus, the curvature of the central path with respect to the local norm does not change much and, hence, it is close to a straight line. Thus, taking a short Newton step inside this ellipsoid from the center keeps one close to the central path.

We return to this intimate connection between the Dikin Ellipsoid and IPMs in the appendix. We conclude this section by noting that there is another way to motivate measuring progress in the local norm: that Newton's method is *affinely invariant*. This means that if, for a invertible linear transformation A, we do a change of variables $x = Ay$, then the Newton step is just a transformation by A: $n(x) = An(y)$. On the other hand the Lipschitz condition on the Hessian in Theorem 3.2 is not affine invariant w.r.t. the Euclidean norm. However, it would be if we redefine it as

$$\|H(x) - H(y)\| \leq L\|x - y\|_x,$$

this remains affine invariant.

3.4.4 A Primal Path-Following IPM

We come back to the description of the path following algorithm. Recall that we defined the central path as: $\{x^\star_\eta : \eta > 0\}$ consisting of optimal solutions to the perturbed linear program. We want to start at $x_0 = x^\star_{\eta_0}$ for some small $\eta_0 > 0$ and move along the path by taking discrete steps of certain length. This corresponds essentially to increasing η in every step. So one idea for our iterative procedure would be to produce a sequence of pairs $(x_0, \eta_0), (x_1, \eta_1), \ldots$ such that $\eta_0 < \eta_1 < \cdots$ and $x_k = x^\star_{\eta_k}$ for every k. We should finish at the moment when we are close enough to the optimum, that is when $c^\top x_k < c^\top x^\star + \varepsilon$. Our first lemma gives a bound on when to stop:

Lemma 3.4. For every $\eta > 0$ we have $c^\top x^\star_\eta - c^\top x^\star < \frac{m}{\eta}$.

Proof. Calculate first the derivative of $f_\eta(x)$:

$$\nabla f_\eta(x) = \nabla(\eta c^\top x + F(x)) = \eta c + \nabla F(x) = \eta c + g(x)$$

The point $x_\eta^\star$ is the minimum of f_η, hence $\nabla f_\eta(x_\eta^\star) = 0$ and so:

$$g(x_\eta^\star) = -\eta c \tag{3.12}$$

Using this observation we obtain that

$$c^\top x_\eta^\star - c^\top x^\star = -\left\langle c, x^\star - x_\eta^\star \right\rangle = \frac{1}{\eta}\left\langle g(x_\eta^\star), x^\star - x_\eta^\star \right\rangle$$

To complete the proof it remains to argue that $\left\langle g(x_\eta^\star), x^\star - x_\eta^\star \right\rangle < m$. We will show even more, that for every two points x, y in the interior of P, we have $\langle g(x), y - x\rangle < m$. This follows by a simple calculation:

$$\begin{aligned}
\langle g(x), y - x\rangle &= \sum_{i=1}^{m} \frac{a_i^\top (y-x)}{s_i(x)} \\
&= \sum_{i=1}^{m} \frac{(b_i - a_i^\top x) - (b_i - a_i^\top y)}{s_i(x)} \\
&= \sum_{i=1}^{m} \frac{s_i(x) - s_i(y)}{s_i(x)} = m - \sum_{i=1}^{m} \frac{s_i(y)}{s_i(x)} < m
\end{aligned}$$

Where in the last inequality we make use of the fact that our points x, y are strictly feasible, i.e. $s_i(x), s_i(y) > 0$ for all i.

□

This lemma tells us that if want an ε-additive solution, we could stop our path following procedure when

$$\eta = \Omega\left(\frac{m}{\varepsilon}\right).$$

At this point one may wonder why instead of working in an iterative fashion we do not just set $\eta = {}^{m}\!/\!_{\varepsilon}$ and solve the perturbed linear problem to optimality. It turns out that it is hard to compute $x_\eta^\star$ when some arbitrary $\eta > 0$ is given (essentially, it is at least as hard as solving linear optimization problems). What we are able to do is given $x_\eta^\star$ for some $\eta > 0$ calculate $x_{\eta'}^\star$ for η' a little bit larger than η. This immediately rouses another question: then how do we find $x_{\eta_0}^\star$ at the very beginning of the algorithm? We will discuss this problem in detail later, for now let us assume that it is possible to provide some $\eta_0 > 0$ and the corresponding point $x_0 = x^\star_{\eta_0}$.

One step of our algorithm will essentially correspond to one step of Newton's method. Before we give a full description of the algorithm recall that $n(x)$ is the Newton step at point x with respect to some underlying function f. Whenever we write $n(x_k)$, we have the function $f_{\eta_k}(x) = \eta_k c^\top x + F(x)$ in mind. So

$$n(x_k) = -H(x_k)^{-1}\nabla f_{\eta_k}(x_k).$$

We now give the algorithm.

```
Primal Path Following IPM for Linear Programming:

  (1) Find an initial $\eta_0$ and $x_0$ with $\|n(x_0)\|_{x_0} \leq \frac{1}{2}$.
  (2) At iteration $k$ ($k = 0, 1, 2, \ldots$):
        • compute $x_{k+1}$ according to the rule:
              $x_{k+1} \overset{\text{def}}{=} x_k + n(x_k)$
        • set $\eta_{k+1} \overset{\text{def}}{=} \eta_k \left(1 + \frac{1}{8\sqrt{m}}\right)$.
  (3) Stop when for the first time $K$, $\eta_K > \frac{m}{\varepsilon}$.
  (4) Calculate $\hat{x} \overset{\text{def}}{=} x^\star_{\eta_K}$ by Newton's method (starting at
      $x_K$), output $\hat{x}$.
```

In this algorithm we do not ensure that $x_k = x^\star_{\eta_k}$ at every step. This means that as opposed to what we have said before, our points do not lie on the central path. All we care about is that the points are *close enough to the central path.* By close enough, we mean $\|n(x_k)\|_{x_k} \leq 1/2$. It may not be clear why this is the right notion of the distance from central path. We will discuss this in the Section 3.5. Let us conclude the description of the algorithm by the following remark.

Remark 3.5. At step 4. of the *Primal Path Following IPM* we apply Newton's method to obtain a point on the central path. At this moment it is not obvious that it will converge and, if yes, how quickly. We will see later that the point x_K is in the quadratic convergence region of Newton's method, so in fact we will need only a few iterations to reach $x^\star_{\eta_K}$. Instead of doing this final computation, one could also output simply x_K. It can be shown that the optimality guarantee at point x_K is $O(\varepsilon)$, see Appendix 3.7

We summarize what we prove about the algorithm above:

Theorem 3.6. The primal path following algorithm, given a linear program with m constraints and a precision parameter $\varepsilon > 0$, performs $O(\sqrt{m} \log {}^{m}\!/\!{}_{\eta_0 \cdot \varepsilon})$ iterations and outputs a point $\hat{x}$ satisfying:

$$c^\top \hat{x} \leq c^\top x^\star + \varepsilon.$$

3.4.5 Analysis of the Algorithm

This section is devoted to the proof of Theorem 3.6. The statement does not provide any bounds on the cost of a single iteration. It is easy to see that the only expensive operation we perform at every iteration is computing the Newton step. This computation can be viewed as solving a linear system of the form $H(x)z = g(x)$, where z is the vector of variables. Of course, a trivial bound would be $O(n^3)$ or $O(n^\omega)$, but these may be significantly improved for specific problems. For example, computing a Newton step for an LP max-flow formulation can be done in $\widetilde{O}(|E|)$ time.

Recall that we initialize our algorithm with some $\eta_0 > 0$. Then, at every step we increase η by a factor of $(1 + \frac{1}{8\sqrt{m}})$, thus after $O(\sqrt{m} \log {}^{m}\!/\!{}_{\eta_0 \cdot \varepsilon})$ we reach $\Omega({}^{m}\!/\!{}_{\varepsilon})$. This establishes the bound on the number of iterations. It remains to prove the correctness (the optimality guarantee).

The following lemma plays a crucial role in the proof of correctness. It asserts that the points we produce lie close to the central path:

Lemma 3.7. For every $k = 0, 1, \ldots, K$ it holds that $\|n(x_k)\|_{x_k} \leq \frac{1}{2}$.

To prove this, we consider one iteration and show that, if started with $\|n(x_k)\|_{x_k} \leq 1/2$, then we will end up with $\|n(x_{k+1})\|_{x_{k+1}} \leq 1/2$. Every iteration consists of two steps: the Newton step w.r.t. f_{η_k} and the increase of η_k to η_{k+1}. We formulate another two lemmas explaining what happens when performing those steps.

Lemma 3.8. After taking one Newton step at point x w.r.t. the function f_η , the new point x' satisfies:

$$\|n(x')\|_{x'} \leq \|n(x)\|_x^2.$$

Lemma 3.9. For every two positive $\eta, \eta' > 0$, we have:

$$\|H^{-1}(x)\nabla f_{\eta'}(x)\|_x \leq \frac{\eta'}{\eta}\|H^{-1}(x)\nabla f_\eta(x)\|_x + \sqrt{m}\left|\frac{\eta'}{\eta} - 1\right|.$$

It is easy to verify that the above two lemmas together imply that if $\|n(x_k)\|_{x_k} \leq \frac{1}{2}$, then

$$\|n(x_{k+1})\|_{x_{k+1}} \leq \frac{1}{4} + \frac{1}{8} + o(1) < \frac{1}{2}.$$

Hence, Lemma 3.7 follows by induction.

It remains to prove Lemmas 3.8 and 3.9. We start with the latter, since its proof is a bit simpler.

Proof. [Proof of Lemma 3.9] We have

$$\begin{aligned} H^{-1}(x)\nabla f_{\eta'}(x) &= H^{-1}(x)(\eta' c + g(x)) \\ &= \frac{\eta'}{\eta} H^{-1}(x)(\eta c + g(x)) + (1 - \frac{\eta'}{\eta})H^{-1}g(x) \\ &= \frac{\eta'}{\eta} H^{-1}(x)\nabla f_\eta(x) + \left(1 - \frac{\eta'}{\eta}\right) H^{-1}g(x). \end{aligned}$$

After taking norms and applying triangle inequality w.r.t. $\|\cdot\|_x$:

$$\|H^{-1}(x)\nabla f_{\eta'}(x)\|_x \leq \frac{\eta'}{\eta}\|H^{-1}(x)\nabla f_\eta(x)\|_x + \left|1 - \frac{\eta'}{\eta}\right| \|H^{-1}g(x)\|_x.$$

Let us stop here for a moment and try to understand what is the significance of the specific terms in the last expression. In our analysis of the algorithm, the term $\|H^{-1}(x)\nabla f_\eta(x)\|_x$ is a small constant. The goal is to bound the left hand side by a small constant as well. We should think of η' as $\eta(1+\delta)$ for some small $\delta > 0$. In such a setting $\frac{\eta'}{\eta}\|H^{-1}(x)\nabla f_\eta(x)\|_x$ will be still a small constant, so what prevents us from choosing a large δ is the second term $(1-\frac{\eta'}{\eta})\|H^{-1}g(x)\|_x$. We need to derive an upper bound on $\|H^{-1}g(x)\|_x$. We show that $\|H^{-1}g(x)\|_x \leq \sqrt{m}$. Let us denote $z \overset{\text{def}}{=} H^{-1}g(x)$. We get:

$$\|z\|_x^2 = z^\top g(x) = \sum_{i=1}^m \frac{z^\top a_i}{s_i(x)} \leq \sqrt{m}\sqrt{\sum_{i=1}^m \frac{(z^\top a_i)^2}{s_i(x)^2}}. \tag{3.13}$$

The last inequality follows from Cauchy-Schwarz. Further:

$$\sum_{i=1}^m \frac{(z^\top a_i)^2}{s_i(x)^2} = z^\top \left(\sum_{i=1}^m \frac{a_i a_i^\top}{s_i(x)^2}\right) z = z^\top H(x) z = \|z\|_x^2. \tag{3.14}$$

Putting (3.13) and (3.14) together we obtain that $\|z\|_x^2 \leq \sqrt{m}\|z\|_x$, so in fact $\|z\|_x \leq \sqrt{m}$.

□

Before we proceed with the proof of Lemma 3.8 let us remark that it can be seen as an assertion that x belongs to the quadratic convergence region of Newton's method.

Proof. [Proof of Lemma 3.8] We know that the point x' is the minimizer of the second order approximation of f_η at point x, hence:

$$\nabla f_\eta(x) + H(x)(x'-x) = 0.$$

which implies that:

$$\sum_{i=1}^m \frac{a_i}{s_i(x)} + \sum_{i=1}^m \frac{a_i a_i^\top}{s_i(x)^2}(x'-x) = -\eta c.$$

We use it to compute $\nabla f_\eta(x')$:

$$\begin{aligned}
\nabla f_\eta(x') &= \eta c + \sum_{i=1}^{m} \frac{a_i}{s_i(x')} \\
&= -\left(\sum_{i=1}^{m} \frac{a_i}{s_i(x)} + \sum_{i=1}^{m} \frac{a_i a_i^\top}{s_i(x)^2}(x'-x) \right) + \sum_{i=1}^{m} \frac{a_i}{s_i(x')} \\
&= \sum_{i=1}^{m} \left(\frac{a_i}{s_i(x')} - \frac{a_i}{s_i(x)} - \frac{a_i a_i^\top (x'-x)}{s_i(x)^2} \right) \\
&= \sum_{i=1}^{m} \left(\frac{a_i a_i^\top (x'-x)}{s_i(x)s_i(x')} - \frac{a_i a_i^\top (x'-x)}{s_i(x)^2} \right) \\
&= \sum_{i=1}^{m} \frac{a_i (a_i^\top (x'-x))^2}{s_i(x)^2 s_i(x')}.
\end{aligned}$$

Our goal is to show that $\|n(x')\|_{x'} \leq \|n(x)\|_x^2$. Instead [7] , we will prove that for every vector z, we have that $\langle z, n(x')\rangle_{x'} \leq \|z\|_{x'}\|n(x)\|_x^2$. Indeed:

$$\begin{aligned}
\left\langle z, n(x')\right\rangle_{x'} = z^\top \nabla f_\eta(x') &= \sum_{i=1}^{m} \frac{z^\top a_i (a_i^\top (x'-x))^2}{s_i(x)^2 s_i(x')} \\
&\overset{\text{Cauchy}-\text{Schwarz}}{\leq} \left(\sum_{i=1}^{m} \frac{(z^\top a_i)^2}{s_i(x')^2} \right)^{1/2} \cdot \left(\sum_{i=1}^{m} \frac{(a_i^\top (x'-x))^4}{s_i(x)^4} \right)^{1/2} \\
&\leq \|z\|_{x'} \cdot \left(\sum_{i=1}^{m} \frac{(a_i^\top (x'-x))^2}{s_i(x)^2} \right) = \|z\|_{x'} \|n(x)\|_x^2
\end{aligned}$$

which completes the proof.

□

3.4.6 The Starting Point

In this section we give a method for finding a valid starting point. More precisely, we show how to find efficiently some $\eta_0 > 0$ and x_0 such that

[7] The following fact is true for every Hilbert space $\mathcal{H}$: the norm of an element $u \in \mathcal{H}$ is given by the formula $\|u\| = \max\{\frac{\langle z,u \rangle}{\|z\|} : z \in \mathcal{H} \setminus \{0\}\}$. We work with $\mathcal{H} = \mathbb{R}^n$ but with nonstandard inner product: $\langle u, v\rangle_{x'} = u^\top H(x')v$

$\|n(x_0)\|_{x_0} \le 1/2$.

Before we start, we would like to remark that this discussion provides a very small η_0, of order 2^{-L}. This enables us to prove that in fact IPM can solve linear programming in polynomial time, but does not seem promising when trying to apply IPM to devise fast algorithms for combinatorial problems. Indeed, there is a factor of $\log \eta_0^{-1}$ in the bound on number of iterations, which translates to L for such a tiny η_0. To make an algorithm fast, we need to have $\eta_0 = \Omega(1/\mathrm{poly(m)})$. However, it turns out that for specific problems (such as maximum flow) we can often devise some specialized methods for finding satisfying η_0 and x_0 and thus solve this issue.

First, we will show how given a point $x' \in \mathrm{int}(P)$ we can find some starting pair (η_0, x_0). Then finally we show how to obtain such point $x' \in \mathrm{int}(P)$. Let us assume now that such a point is given. Furthermore, we assume that each of its coordinates is written using $O(L)$ bits and each constraint is satisfied with slack at least 2^{-L}, that is $b_i - a_i^\top x' \ge 2^{-L}$. Our procedure for finding x' will provide such an x' based on our assumption that P is full dimensional. [8]

Recall that we want to find a point x_0 close to the central path $\Gamma_c = \{x_\eta^\star : \eta \ge 0\}$, which corresponds to the objective function $c^\top x$. Note that as $\eta \to 0$, $x_\eta^\star \to x_0^\star = x_c$, the analytic center of P. So finding a point x_0, very close to the analytic center and choosing η_0 to be some very tiny number should be a good strategy. In fact it is, but how to find a point close to x_c?

The central path Γ_c is of main interest for us, because it tends to the optimal solution to our linear program. In general, if $d \in \mathbb{R}^n$ we may define Γ_d to be the path consisting of minimizers to the functions $\nu d^\top x + F(x)$ for $\nu \ge 0$. What do all the paths Γ_d have in common? The origin! They all start at the same point: analytic center of P. Our strategy will be to pick one such path on which x' lies and traverse it backwards to reach a point very close to the origin of this path, which at the same time will be a good choice for x_0.

Recall that g is the gradient of the logarithimic barrier F and define

[8] In this presentation we will not discuss some details, e.g. the full-dimensionality assumption. These are easy to deal with and can be found in any book on linear programming.

$d = -g(x')$. Now it turns out that $x' \in \Gamma_d$. Why is this? Denote $f'_\nu(x) = d^\top x + F(x)$ and let $x'^\star_\nu$ be the minimizer of f'_ν. Then $x'^\star_1 = x'$, since $\nabla f'_1(x') = 0$. We will use $n'(x)$ for the Newton step at x with respect to f'_ν. We see in particular that $n'(x') = 0$.

As mentioned above, our goal is to move along the Γ_d path in the direction of decreasing ν. We will use exactly the same method as in the *Primal Path Following*. We perform steps, in each of them we make one Newton step w.r.t. current ν and then decrease ν by a factor of $(1 - {}^1\!/_{8\sqrt{m}})$. At each step it holds that $\|n'(x)\|_x \le {}^1\!/_2$, by an argument identical to the proof of Lemma 3.7. It remains to see, how small ν we need to have (how many iterations we need to perform).

Lemma 3.10. If $\|H(x)^{-1}g(x)\|_x \le {}^1\!/_4$ and we take $\eta_0 = (4\|H(x)^{-1}c\|_x)^{-1}$, then the point x is a good candidate for x_0. The Newton step $n(x)$ w.r.t. f_{η_0} satisfies:

$$\|n(x)\|_x \le \frac{1}{2}.$$

Proof. This is just a simple calculation: $\|n(x)\|_x =$

$$\|H^{-1}(x)(\eta_0 c + g(x))\|_x \le \eta_0\|H(x)^{-1}c\|_x + \|H(x)^{-1}g(x)\|_x \le \frac{1}{4} + \frac{1}{4}.$$

□

Suppose now we have some very small $\nu > 0$ and a point x such that $\|n'(x)\|_x \le {}^1\!/_2$. By performing two further Newton steps, we may assume $\|n'(x)\|_x \le {}^1\!/_{16}$. We have:

$$n'(x) = H(x)^{-1}(-\nu g(x') + g(x)),$$

hence:

$$\|H(x)^{-1}g(x)\|_x \le \nu\|H(x)^{-1}g(x')\|_x + \|n'(x)\|_x \le \nu\|H(x)^{-1}g(x')\|_x + \frac{1}{16}$$

So it turns out, by Lemma 3.10 that it is enough to make $\nu\|H(x)^{-1}g(x')\|_x$ smaller then a constant. We can make ν as small as we want, but what with $\|H(x)^{-1}g(x')\|_x$? Let us present a technical claim, which we leave without proof:

Claim 3.11. All the calculations during the backward walk along Γ_d can be performed with $2^{O(L)}$ precision.

This means that we may assume that all the points x computed during the procedure have only $O(L)$ places after the comma, and are of absolute value at most $2^{O(L)}$. This allows us to provide an upper bound on $\|H(x)^{-1}g(x')\|_x$.

Claim 3.12. If the description sizes of x' and x are $O(L)$, then $\|H(x)^{-1}g(x')\|_x \leq 2^{\text{poly(L)}}$.

This follows from the fact that a solution to a linear system is of polynomial size. Now it remains to take ν so small that $\nu\|H(x)^{-1}g(x')\|_x \leq {}^1\!/_8$. By our previous reasoning this is enough to get suitable (η_0, x_0). To reach such a ν we need to perform $\widetilde{O}(\sqrt{m}\log {}^1\!/_\nu)$ iterations, but $\log {}^1\!/_\nu$ is polynomial in L, so we are done.

To complete our considerations we show how to find a suitable $x' \in \text{int}(P)$. To this end, we consider an auxiliary linear program:

$$\begin{aligned} &\min_{(t,x)\in\mathbb{R}^{n+1}} t \\ &a_i^\top x \leq b_i + t, \text{ for all } i \end{aligned} \tag{3.15}$$

Taking $x = 0$ and t big enough ($t = 1 + \sum_i |b_i|$), we get a strictly feasible solution. Having such, we may solve 3.15 up to an additive error of $2^{-O(L)}$ in polynomial time (by IPM). This will give us a solution (t', x') with $t' = -2^{-\Omega(L)}$, so $x' \in \text{int}(P)$ is a suitable starting point for our walk along Γ_d.

3.5 Self-Concordant Barrier Functions

Finally, in this section we present the definition of a *self-concordant barrier function* for a convex set P. Armed with this definition, the task of bounding the number of iterations of the path following method for minimizing a linear function over P reduces to analyzing certain structural properties of the self-concordant barrier function. We omit the straightforward details about how, once we have a self-concordant barrier function for a convex set, one can design a path following IPM for it.

Definition 3.13. Let $F : \mathrm{int}(P) \mapsto \mathbb{R}$ be a $\mathcal{C}^3$ function. We say that F is self-concordant with parameter ν if:

(1) F is a barrier function: $F(x) \to \infty$ as $x \to \partial P$.
(2) F is strictly convex.
(3) For all $x \in \mathrm{int}(P)$, $\nabla^2 F(x) \succeq \frac{1}{\nu} \nabla F(x) \nabla F(x)^\top$.
(4) For all $x \in \mathrm{int}(P)$, $\left|\nabla^3 F(x)[h,h,h]\right| \leq 2\|h\|_x^3 = 2\left|\nabla^2 F(x)[h,h]\right|^{3/2}$.[9]

With this definition in hand, it is not difficult to give an IPM for P which takes about $\sqrt{\nu} \log {}^1\!/\!_\varepsilon$ iterations. In fact, a careful reader would already have observed that Lemmas 3.9 and 3.8 prove exactly (3) and (4) for the log-barrier function with the complexity parameter m.

Similar to the log-barrier function for the positive real-orthant, for semidefinite programs over the cone of positive-definite matrices (or an affine transformation of it), one can use the following barrier function:

$$F(X) = -\log \det X$$

for $X \succ 0$.

In the next lecture we will discuss self-concordant barrier functions with complexity parameters $\ll m$. The main object of study will be the *volumetric barrier* of Vaidya.

3.6 Appendix: The Dikin Ellipsoid

Recall our setting:

$$\begin{aligned} \min\ & \langle c, x \rangle \\ \text{s.t.}\quad & Ax \leq b \end{aligned}$$

where $Ax \leq b$ defines a bounded, nonempty and full-dimensional polytope P. Recall that we can write the constrains at $\langle a_i, x \rangle \leq b_i$ and

[9] Here we interpret $\nabla^3 F(x)[h,h,h]$ as the inner product between the 3-tensor $\nabla^3 F(x)$ and $h \otimes h \otimes h$, namely, $\langle \nabla^3 F(x), h \otimes h \otimes h \rangle)$ and $\nabla^2 F(x)[h,h] = h^\top \nabla^2 F(x) h = \langle \nabla^2 F(x), h \otimes h \rangle$.

then consider the *slacks* $s_i(x) = b_i - \langle a_i, x \rangle$ which capture the extent to which a constraint is satisfied. Clearly, if x is a feasible point, then $s_i(x) \geq 0$ for all i. Note that, when clear from context, we will often write s_i for $s_i(x)$ for conciseness. The log-barrier function (rather its negative) and its differentials are:

$$F(x) = \sum_i \log(s_i(x))$$

where

$$\begin{aligned} \nabla F(x) &= \frac{a_i}{s_i}, \qquad \text{and} \\ \nabla^2 F(x) &= \sum_i \frac{a_i a_i^\top}{s_i^2}. \end{aligned}$$

We often write $H(x) = \nabla^2 F(x)$ for the *Hessian* of x, and note that $H(x) \succeq 0$ (and in fact $H(x) \succ 0$ when A is fully-dimensional).

Now, we can consider the ellipsoid centred at x and defined by H, namely

$$\mathcal{E}_x(H(x)) = \{y : (y - x)^\top H(x)(y - x) \leq 1\}.$$

This is the *Dikin ellipsoid* centred at x.

Definition 3.14. The *Dikin Ellipsoid* at $x \in \text{int}(P)$ is defined as the ball of radius 1 around x in the local norm defined by $H(x)$ at x.

We will show that the Dikin ellipsoid is always completely contained in the polytope P. Furthermore, if we denote by x_c the *Analytic Center* of P, i.e. the unique minimizer of $F(x)$, then the Dikin ellipsoid at x_c inflated m times contains the whole polytope P (we will also show that $\sqrt{m}$ blow-up is sufficient for symmetric polytopes). We start by proving the first claim.

Theorem 3.15. For every $x \in \text{int}(P)$, the Dikin ellipsoid at x is fully contained in P.[10]

[10] In fact, this also holds when P is not bounded for certain cases such as if P is the positive orthant.

Proof. This is easy to see since, for any $y \in \mathcal{E}_x$, all slacks are non-negative, and hence $y \in P$. Formally, let $y \in \mathcal{E}_x(H(x))$. Then,

$$\begin{aligned}
(y-x)^\top H(x)(y-x) &\leq 1 \\
\Rightarrow \sum_i \frac{\langle a_i, y-x\rangle^2}{s_i^2} &\leq 1 \\
\Rightarrow \frac{\langle a_i, y-x\rangle^2}{s_i^2} &\leq 1 \quad \forall i \text{ since all summands are non-negative} \\
\Rightarrow \left(\frac{s_i(x)-s_i(y)}{s_i(x)}\right)^2 &\leq 1 \quad \forall i \\
\Rightarrow \left|1-\frac{s_i(y)}{s_i(x)}\right| &\leq 1 \quad \forall i.
\end{aligned}$$

Hence, for all i we know that $0 \leq {}^{s_i(y)}\!/\!_{s_i(x)} \leq 2$, and, in particular, $s_i(y) \geq 0 \quad \forall i$. □

Let us start with a very interesting lemma describing one crucial property of the analytic center of P:

Lemma 3.16. If x_c is the analytic center of P then for every point $x \in \text{int}(P)$ it holds that:

$$\sum_{i=1}^m \frac{s_i(x)}{s_i(x_c)} = m.$$

Proof. Let us denote $r(x) = \sum_{i=1}^m \frac{s_i(x)}{s_i(x_c)}$. We know that $r(x_c) = m$. To show that a function is constant, it remains to find its derivative and argue that it is zero:

$$\nabla r(x) = \nabla\left(\sum_{i=1}^m \frac{s_i(x)}{s_i(x_c)}\right) = \frac{a_i}{s_i(x_c)} = \nabla F(x_c)$$

but x_c is the minimizer of $F(x)$ so the gradient at this point vanishes. □

Now we are ready to proof the second theorem describing the Dikin ellipsoid. This time we look at the Dikin ellipsoid centered at a specific point x_c:

Theorem 3.17. The Dikin ellipsoid at x_c inflated m times contains P inside: $P \subseteq m\mathcal{E}_{x_c}(H(x_c))$.

Proof. Take any point $x \in P$, the goal is to show that $(x - x_c)^\top H(x_c)(x - x_c) \le m^2$. We compute:

$$\begin{aligned}
&(x-x_c)^\top H(x_c)(x-x_c)\\
=&(x-x_c)^\top \left(\sum_{i=1}^m \frac{a_i a_i^\top}{s_i(x_c)^2}\right)(x-x_c)\\
=&\sum_{i=1}^m \frac{(a_i^\top (x-x_c))^2}{s_i(x_c)^2} = \sum_{i=1}^m \frac{(s_i(x_c)-s_i(x))^2}{s_i(x_c)^2}\\
=&\sum_{i=1}^m \frac{s_i(x)^2}{s_i(x_c)^2} + \sum_{i=1}^m \frac{s_i(x_c)^2}{s_i(x_c)^2} - 2\sum_{i=1}^m \frac{s_i(x_c)s_i(x)}{s_i(x_c)^2}.
\end{aligned}$$

Now, the middle term $\sum_{i=1}^m \frac{s_i(x_c)^2}{s_i(x_c)^2}$ is equal to m and $\sum_{i=1}^m \frac{s_i(x_c)s_i(x)}{s_i(x_c)^2} = \sum_{i=1}^m \frac{s_i(x)}{s_i(x_c)} = m$ by Lemma 3.16. So we obtain:

$$(x-x_c)^\top H(x_c)(x-x_c) = \sum_{i=1}^m \frac{s_i(x)^2}{s_i(x_c)^2} - m.$$

Observe that all the terms in the sum $\sum_{i=1}^m \frac{s_i(x)^2}{s_i(x_c)^2}$ are nonnegative, so we can use the simple bound $\sum_{i=1}^m \frac{s_i(x)^2}{s_i(x_c)^2} \le \left(\sum_{i=1}^m \frac{s_i(x)}{s_i(x_c)}\right)^2 = m^2$ (the last equality again by Lemma 3.16) and obtain finally:

$$(x-x_c)^\top H(x_c)(x-x_c) \le m^2 - m \le m^2.$$

□

The last theorem we want to proof about the Dikin ellipsoid asserts that when we restrict ourselves to symmetric polytopes (i.e. symmetric with respect to the center of P) then the Dikin ellipsoid at the center inflated only $\sqrt{m}$ times contains the polytope.

Theorem 3.18. Let P be a symmetric polytope, then $P \subseteq \sqrt{m}\mathcal{E}_{x_c}(H(x_c))$.

Remark 3.19. Note that the Dikin ellipsoid depends not exactly on P but rather on the description of P (on the set of constraints). In the above theorem we assume that the description of P is reasonable. Since P is symmetric (with respect to the origin), we can assume that for every constraint $x^\top a_i \leq b_i$ there is a corresponding constraint $-x^\top a_i \leq b_i$.

Proof. [Proof of Theorem 3.18] We assume that P is symmetric w.r.t. the origin, so $x_c = 0$. Further, since all the b_i's are nonzero, we may rescale the constraints and assume $b_i = 1$ for all i. After this reductions, our ellipsoid is as follows: $\mathcal{E} = \{x : \sum_{i=1}^m (x^\top a_i)^2 \leq 1\}$.

Take any point x on the boundary of $\mathcal{E}$, that is $\sum_{i=1}^m (x^\top a_i)^2 = 1$. We need to show that $\sqrt{m}x \notin \operatorname{int}(P)$. This will finish the proof.

Since $\sum_{i=1}^m (x^\top a_i)^2 = 1$, there exists i with $|x^\top a_i| \geq \frac{1}{\sqrt{m}}$. The set of constraints is symmetric, so we can assume that $x^\top a_i \geq \frac{1}{\sqrt{m}}$. Hence $(\sqrt{m}x)^\top a_i \geq 1$, so $\sqrt{m}x \notin \operatorname{int}(P)$. □

Theorems 3.17 and 3.18 together with Theorem 3.15 demostrate that the Dikin ellipsoid centered at x_c is a very good approximation of the polytope P. We obtain an ellipsoid with a blow-up ratio of m (that is $\mathcal{E}_{x_c}(H(x_c)) \subseteq P \subseteq m\mathcal{E}_{x_c}(H(x_c)))$ or $\sqrt{m}$ for the symmetric case. One can ask if this is the best we can do. It turns out that we can obtain better ratio by taking the so called John ellipsoid. It is defined as the largest-volume ellipsoid contained in the polytope P. When we inflate it by a factor of n, it contains P inside (similarly, $\sqrt{n}$ is enough for symmetric polytopes). This means that the John Ellipsoid achieves better blow-up ratio, because $n \leq m$ (we need at least n linear inequalities to define a non-degenerate, bounded polytope in n-dimensional space). One can also prove that this is indeed tight, the best possible blow-up ratio for the n-dimensional simplex is exactly n.

3.6.1 The Dikin Algorithm and some Intuition for $\sqrt{m}$ Iterations

The results above imply an particularly simple greedy IPM: start with the analytic centre x_c of a polytope P and move to the boundary of $\mathcal{E}_{x_c}(H(x_c))$ in the direction of $-c$, where c is the cost vector. Repeat the same from the next point. First note that algorithmically this makes sense as Theorem 3.15 implies that throughout the execution of the algorithm, we are always inside P. The cost is also decreasing. However, we can only argue that the cost becomes $1/\sqrt{m}$ in the symmetric case (or $1/m$ in the non-symmetric case) of the optimal cost in the *first* iteration. To see this assume that P is symmetric. Thus, $x_c = 0$ and the cost of this point is 0. Let x be the point on the boundary of $\mathcal{E}_{x_c}(H(x_c))$ in the direction of $-c$. We know from Theorem 3.15 that $x \in P$. However, we also know from Theorem 3.18 that $\langle c, \sqrt{m}x\rangle \leq \langle c, x^\star\rangle = \text{opt}$, where $x^\star$ is the optimal solution to the linear program. This is because $\sqrt{m}x$ lies on the boundary of $\sqrt{m}\mathcal{E}_{x_c}$ which contains P, and is in the direction of $-c$. Thus, $\langle c, x\rangle \leq \frac{1}{\sqrt{m}}\text{opt}$. If this were to magically continue at every step then one would expect the cost to come around opt in about $\sqrt{m}$ iterations. However, we cannot prove this and this analogy ends here.

3.7 Appendix: The Length of Newton Step

In this section we explain the relation between the length of the Newton step at point x (with respect to the function f_η) and the distance to the optimum $x_\eta^\star$. We show that whenever $\|n(x)\|_x$ is sufficiently small, $\|x - x_\eta^\star\|_x$ is small as well. This together with a certain strenghtening of Lemma 3.4 imply that in the last step of *Primal Path Following IPM* we do not need to go with x_K to optimality. In fact, only 2 additional Newton steps bring us (2ε)-close to the optimum.

Let us start by a generalization of Lemma 3.4, which shows that to get a decent approximation of the optimum we do not neccessarily need to be on the central path, but only close enough to it.

Lemma 3.20. For every point $x \in \text{int}(P)$ and every $\eta > 0$, if $\|x -$

$x^\star_\eta\|_x < 1$ then:

$$c^\top x - c^\top x^\star \leq \frac{m}{\eta}(1 - \|x - x^\star_\eta\|_x)^{-1}.$$

Proof. For every $y \in \mathrm{int}(P)$ we have:

$$\begin{aligned} c^\top x - c^\top y &= c^\top (x - y) \\ &= \langle c, x - y \rangle \\ &= \langle c_x, x - y \rangle_x \\ &\leq \|c_x\|_x \|x - y\|_x \end{aligned}$$

Where $c_x = H^{-1}(x)$ and the last inequality follows from Cauchy-Schwarz. Now, we want to bound $\|c_x\|_x$. Imagine we are at point x and we move in the direction of $-c_x$ until hitting the boundary of Dikin's ellipsoid. We will land in the point $x - \frac{c_x}{\|c_x\|_x}$, which is still inside P by Theorem 3.15. Therefore:

$$\left\langle c, x - \frac{c_x}{\|c_x\|_x} \right\rangle \geq \langle c, x^\star \rangle.$$

Since $\langle c, c_x \rangle = \|c_x\|_x^2$, we get:

$$\|c_x\|_x \leq \langle c, x \rangle - \langle c, x^\star \rangle.$$

We have obtained:

$$c^\top x - c^\top y \leq \|x - y\|_x (c^\top x - c^\top x^\star). \tag{3.16}$$

Now, let us express

$$c^\top x - c^\top x^\star = (c^\top x - c^\top x^\star_\eta) + (c^\top x^\star_\eta - c^\top x^\star)$$

and use 3.16 with $y = x^\star_\eta$. We obtain

$$c^\top x - c^\top x^\star \leq (c^\top x - c^\top x^\star)\|x - y\|_x + (c^\top x^\star_\eta - c^\top x^\star).$$

Thus

$$(c^\top x - c^\top x^\star)(1 - \|x - y\|_x) \leq c^\top x^\star_\eta - c^\top x^\star.$$

By applying Lemma 3.4 we get the result. □

Note that in the algorithm we never literally mention the condition that $\|x - x^\star_\eta\|_x$ is small. However, we will show that it follows from $\|n(x)\|_x$ being small. We will need the following simple fact about logarithmic barrier. A proof appears in the notes on Volumetric Barrier in the next lecture.

Fact 3.21. If $x, z \in \mathrm{int}(P)$ are close to each other, i.e. $\|z - x\|_z \leq \frac{1}{4}$ then $H(x)$ and $H(z)$ are close as well:

$$\frac{4}{9}H(x) \preceq H(z) \preceq 4H(x).$$

We are ready to prove the main theorem of this section.

Theorem 3.22. Let x be any point in $\mathrm{int}(P)$ and $\eta > 0$. Consider the Newton step $n(x)$ at point x with respect to f_η. If $\|n(x)\|_x \leq \frac{1}{18}$, then $\|x - x^\star_\eta\|_x \leq 5\|n(x)\|_x$.

Proof. Pick any h such that $\|h\|_x \leq \frac{1}{4}$. Expand $f_\eta(x+h)$ into a Taylor series around x:

$$f_\eta(x+h) = f_\eta(x) + h^\top \nabla f_\eta(x) + \frac{1}{2}h^\top \nabla^2 f_\eta(\theta)h \tag{3.17}$$

for some point θ lying on the segment $[x, x+h]$. We proceed by lower-bounding the linear term. Note that:

$$\left|h^\top \nabla f_\eta(x)\right| = |\langle h, n(x)\rangle_x| \leq \|h\|_x \|n(x)\|_x \tag{3.18}$$

by Cauchy-Schwarz. Next:

$$h^\top \nabla^2 f_\eta(\theta)h = h^\top H(\theta)h \geq \frac{4}{9}h^\top H(x)h \tag{3.19}$$

where we used Fact 3.21. Applying bounds 3.18, 3.19 to the expansion 3.17 results in:

$$f_\eta(x+h) \geq f_\eta(x) - \|h\|_x \|n(x)\|_x + \frac{2}{9}\|h\|_x^2. \tag{3.20}$$

Set $r = \frac{9}{2}\|n(x)\|_x$ and consider points y satysfing $\|x - y\|_x = r$, i.e. points on the boundary of the local norm ball of radius r centered at x. Then 3.20 simplifies to:

$$f_\eta(x+h) \geq f_\eta(x)$$

which implies that $x_\eta^\star$, the unique minimizer of f_η, belongs to the above mentioned ball and the theorem follows.

□

Combining Lemma 3.20 with Theorem 3.22 we get that if $\eta \geq \frac{m}{\varepsilon}$ and $\|n(x)\|_x \leq \frac{1}{18}$ then $c^\top x - c^\top x^\star < 2\varepsilon$.

4

Volumetric Barrier, Universal Barrier and John Ellipsoids

4.1 Overview

In this lecture we take a step towards improving the $\sqrt{m}$ in the number of iterations the path-following IPM presented in the previous lecture to $\sqrt{n}$. This will be achieved by considering more involved barrier functions rather than altering the basic framework of IPMs. The log-barrier function was particularly nice as working with it was computationally not harder than solving linear systems. However, it had many shortcomings. An obvious one is that it has no way to take into account that a constraint could be repeated many time. One way to handle this would be to work with *weighted* barrier functions which have the following form:

$$-\sum_i w_i(x) \log s_i(x)$$

where we could allow the weights to depend on the current point as well. Analyzing such methods, however, poses great technical challeges as now the gradients and Hessians become more unwieldy. Amazingly, Vaidya laid the foundations of analyzing such weighted barrier functions. He introduced one such function, the *volumetric barrrier* and showed how it allows us to improve the $\sqrt{m}$ to $(mn)^{1/4}$. Computationally, the volumetric barrier was no harder than computing determinants.

Subsequently, Nesterov-Nemirovskii discovered the *Universal Barrier* which achieves the stated goal of $\sqrt{n}$ iterations. Their result held far beyond the setting of LPs and worked for almost any convex body. However, computationally the barrier is at least hard as solving the convex program itself!

The search for a $\sqrt{n}$ barrier which is computationally efficient got a major boost by a recent result of Lee-Sidford who demonstrated a computationally efficient barrier (about the same complexity as solving linear systems) which achieves $\tilde{O}(\sqrt{n})$ iterations. In my opinion, their result is inspired by Vaidya's work: while Vaidya considered the volume of the Dikin ellipsoid as a barrier, Lee-Sidford consider an object which can be considered a *smoothening* of John ellipsoid. It remains an outstanding problem to remove the log-factors from the work of Lee-Sidford.

We begin by presenting yet another proof of $\sqrt{m}$ convergence and then show how the volumetric barrier allows us to improve this to $(mn)^{1/4}$. Towards the end, we give a brief presentation (without proofs) of the $\sqrt{\text{rank}}$ Universal Barrier by Nesterov-Nemirovskii for *any* convex set. As for the work of Lee-Sidford, currently, explaining it simply remains a challenge. However, in the appendix we introduce the John ellipsoid and give some basic intuition about why one may expect the $\sqrt{m}$ to be possibly replaced by $\sqrt{n}$.

4.2 Restating the Interior Point Method for LPs

We start by restating the interior point method in a slightly different but equivalent way.

Theorem 4.1. Let F be a barrier function such that there exists δ and θ as follows:

(1) For all $x, z \in \text{int}(P)$ such that $\|x - z\|^2_{\nabla^2 F(z)} \leq \delta$, we have $\nabla^2 F(z) \overset{O(1)}{\approx} \nabla^2 F(x)$,[1] and

[1] Formally, there exist constants c_1, c_2 such that $c_1 \nabla^2 F(z) \preceq \nabla^2 F(x) \preceq c_2 \nabla^2 F(z)$.

(2) $\|\nabla F(x)\|_{(\nabla^2 F(x))^{-1}} \leq \theta$.[2]

Then, there is an interior point method that produces an ε-approximate solution to LPs of the form $\min c^\top x$, such that $Ax \leq b$ in $O(\theta/\delta \log 1/\varepsilon)$ iterations. Here m is the number of rows of A.

Note that the first property captures the fact that, if two points x and z are close in the local norm, then the Hessian does not change by much. These two requirements along with differentiability are equivalent to self-concordance with complexity parameter θ/ε. We leave the proof to the reader.

4.2.1 The Logarithmic Barrier Revisited

Before we proceed to the volumetric barrier, as an exercise, we revisit the log-barrier funtion and derive the $O(\sqrt{m})$-iteration convergence result for it, yet again. Recall that the log-barrier function is:

$$F(x) \stackrel{\text{def}}{=} -\sum_{i=1}^{m} \log(b_i - a_i^\top x).$$

We will show that for this function $\theta = \sqrt{m}$ and $\delta = O(1)$. Hence, via Theorem 4.1, it follows that this results in an IPM which converges in roughly $\sqrt{m}$ iterations.

4.2.1.1 θ for log-barrier

Let S be the diagonal matrix where $S_{ii} = s_i(x)$. We can now restate the gradient and Hessian of F with more convenient notation as follows:

$$\begin{aligned} \nabla F(x) &= A^\top S^{-1} 1, \text{ and} \\ \nabla^2 F(x) &= A^\top S^{-2} A. \end{aligned}$$

[2] Equivalently, we could write $(\nabla F(x))(\nabla F(x))^{-1} \preceq \theta \nabla^2 F(x)$.

Hence,

$$\begin{aligned}\|\nabla F(x)\|^2_{(\nabla^2 F(x))^{-1})} &= (\nabla F(x))^\top (\nabla^2 F(x))^{-1}(\nabla F(x))\\ &= (A^\top S^{-1}1)^\top (A^\top S^{-2}A)^{-1}(A^\top S^{-1}1)\\ &= 1^\top (S^{-1}A(A^\top S^{-2}A)^{-1}A^\top S^{-1})1.\end{aligned}$$

Denote

$$\Pi \stackrel{\text{def}}{=} S^{-1}A(A^\top S^{-2}A)^{-1}A^\top S^{-1},$$

and note that Π is a projection matrix; i.e., $\Pi^2 = \Pi$. Hence,

$$\begin{aligned}\|\nabla F(x)\|^2_{(\nabla^2 F(x))^{-1})} &= 1^\top \Pi 1 = 1^\top \Pi^2 1.\\ &= \|\Pi 1\|_2^2 \leq \|1\|_2^2\\ &= m.\end{aligned}$$

Thus, $\|\nabla F(x)\|_{(\nabla^2 F(x))^{-1})} \leq \sqrt{m}$ as desired.

4.2.1.2 δ for the log-barrier

We will show that $\nabla^2 F(z) \overset{O(1)}{\approx} \nabla^2 F(x)$ for $\delta = 1/4$. Let x, z be such that $\|x - z\|_z \leq \delta$. Hence,

$$\begin{aligned}(x-z)^\top (\nabla^2 F(z))(x-z) &= \sum_i \frac{\langle a_i, x-z\rangle^2}{s_i^2(z)}\\ &= \sum_i \frac{(a_i^\top (x-z))^2}{s_i^2(z)}\\ &= \sum_i \left(\frac{s_i(x) - s_i(z)}{s_i(z)}\right)^2.\end{aligned}$$

Therefore,

$$\left|\frac{s_i(x) - s_i(z)}{s_i(z)}\right| \leq \sqrt{\delta} \quad \forall i.$$

In particular, for all i, $1 - \sqrt{\delta} \leq s_i(x)/s_i(z) \leq 1 + \sqrt{\delta}$, and hence, for $\delta = 1/4$,

$$\frac{s_i^2(z)}{s_i^2(x)} \leq 4 \quad \text{and} \quad \frac{s_i^2(x)}{s_i^2(z)} \leq \frac{9}{4}$$

for all i. Since $\min_i \left(\frac{s_i^2(z)}{s_i^2(x)}\right) \cdot \nabla^2 F(z) \preceq \nabla^2 F(x) \preceq \max_i \left(\frac{s_i^2(z)}{s_i^2(x)}\right) \cdot \nabla^2 F(z)$, we have that

$$\frac{4}{9}\nabla^2 F(z) \preceq \nabla^2 F(x) \preceq 4\nabla^2 F(z)$$

which gives $\nabla F(x) \overset{O(1)}{\approx} \nabla F(z)$ as desired.

4.3 Vaidya's Volumetric Barrier

Now, we go *beyond* $\sqrt{m}$; though we would like to get roughly $\sqrt{n}$ iterations, we will not quite reach this target, but will come somewhere in-between. The problem with the log-barrier function is that it weighs each constraint equally. Thus, if, for instance, a constraint is repeated many times, there is no way for the log-barrier function to know this. Vaidya introduced the *volumetric barrier* function which aims to bypass this limitation by assigning weights to each constraint:

$$-\sum_i w_i \log(b_i - a_i^\top x).$$

Of course, a useful choice of weights must depend on the current point x, and this causes several technical difficulties. His work laid the foundations for and developed many techniques to analyze such weighted barrier functions.

4.3.1 The Volumetric Barrier

The proposed barrier function by Vaidya uses the log-volume of the Dikin Ellipsoid.

Definition 4.2. The *volumetric barrier* is defined to be

$$V(x) \stackrel{\text{def}}{=} -\frac{1}{2} \cdot \log\det\left(\nabla^2 F(x)\right).$$

Note that, whenever we come up with a new barrier function, we must also worry about its computability along with the computability of its first and second order oracle. Using the restatement of the barrier function as the determinant of a positive definite matrix, we see that the volumetric barrier is efficiently computable.

Notation and basic properties. Notation will play an important role in understanding the rather technical sections that are to follow. Towards this, we introduce new notation, some of which may conflict with what we have used previously. Instead of $F(x), V(x), H(x)$, and so on, we denote the same quantities by F_x, V_x, and H_x. Similarly, the slack vector will be denoted by s_x and its components by $s_{x,i} = s_i(x)$. Let the i-th row of A be the vector a_i. For an $x \in \mathrm{int}(P)$, let

$$A_x \stackrel{\text{def}}{=} S_x^{-1} A$$

where S_x is the diagonal matrix corresponding to the vector s_x. Then, the Hessian of the log-barrier function can be written as

$$H_x = A_x^\top A_x = A^\top S^{-2} A.$$

Thus, if $V_x = -\frac{1}{2} \log \det H_x$, then

$$\nabla V_x = A_x^\top \sigma_x \tag{4.1}$$

where

$$\sigma_{x,i} \stackrel{\text{def}}{=} \frac{a_i^\top H_x^{-1} a_i}{s_{x,i}^2}.$$

An informed reader could compare $\sigma_{x,i}$ with *leverage scores*, or *effective resistances* that arise when dealing with graph Laplacians. We note some simple properties of σ_x which will allow us to build towards finding the θ and δ necessary for applying Thoerem 4.1.

Fact 4.3.

(1) $\sigma_x \leq 1$.
(2) $\sigma_x^\top 1 = n$.

Proof. The first fact follows from the fact that $\frac{a_i^\top H_x^{-1} a_i}{s_{x,i}^2} \leq 1$ if and only if $\frac{a_i a_i^\top}{s_{x,i}^2} \preceq H_x$. But the latter is true since H_x is the sum over all i of quantites on the left hand side. For the second part note that

$$\begin{aligned}\sigma_x^\top 1 &= \sum_i \frac{a_i^\top H_x^{-1} a_i}{s_{x,i}^2} \\ &= \mathrm{Tr}\left(H_x^{-1} \sum_i \frac{a_i a_i^\top}{s_{x,i}^2}\right) \\ &= \mathrm{Tr}\left(H_x^{-1} H_x\right) \\ &= n.\end{aligned}$$

□

Thus, each $\sigma_{x,i}$ is at most one and they sum up to n. Thus, $\sigma_{x,i}$ assigns a relative importance to each constraint while maintaining a budget of n. This is unlike in the setting of log-barrier where each constraint has a weight of 1 and, hence, requires a budget of m. Further, note the following straightforward fact:

Fact 4.4. $\sigma_{x,i} = \sum_j \frac{(a_i^\top H_x^{-1} a_j)^2}{s_{x,i}^2 s_{x,j}^2}$.

Proof.

$$\begin{aligned}\sigma_{x,i} &= \frac{a_i^\top H_x^{-1} a_i}{s_{x,i}^2} \\ &= \frac{a_i^\top H_x^{-1} H_x H_x^{-1} a_i}{s_{x,i}^2} \\ &= \frac{a_i^\top H_x^{-1} \left(\sum_j \frac{a_j a_j^\top}{s_{x,j}^2}\right) H_x^{-1} a_i}{s_{x,i}^2} \\ &= \sum_j \frac{(a_i^\top H_x^{-1} a_j)^2}{s_{x,i}^2 s_{x,j}^2}.\end{aligned}$$

□

Let Σ_x denote the diagonal matrix corresponding to σ_x. Let

$$P_x \stackrel{\text{def}}{=} A_x H_x^{-1} A_x^\top = A_x (A_x^\top A_x)^{-1} A_x^\top.$$

Note that $P_x \succ 0$, is symmetric and is a projection matrix since

$$P_x^2 = A_x(A_x^\top A_x)^{-1} A_x^\top A_x (A_x^\top A_x)^{-1} A_x^\top = P_x.$$

Further, let $P_x^{(2)}$ denote the matrix whose each entry is the square of the corresponding entry of P_x. Then, Fact 4.4 above can be restated as

$$\sigma_x = P_x^{(2)} 1.$$

Thus, $P_x^{(2)} \geq 0$, $\Sigma_x - P_x^{(2)}$ is symmetric and $(\Sigma_x - P_x^{(2)})1 = 0$. Thus, it is a graph Laplacian. As a consequence we obtain the following fact.

Fact 4.5. $\Lambda_x \stackrel{\text{def}}{=} \Sigma_x - P_x^{(2)} \succeq 0.$

Gradient and the Hessian of the Volumetric Barrier. In the appendix we show the following by a straightforward differentiation of the volumetric barrier:

Lemma 4.6.

(1) $\nabla V_x = A_x^\top \sigma_x$ and
(2) $\nabla^2 V_x = A_x^\top (3\Sigma_x - 2P_x^{(2)}) A_x$.

Thus, using Fact 4.5, we obtain the following corollary which allows us to think of the Hessian of the volumetric barrier as the log-barrier weighted with leverage scores. This lemma greatly simplifies the calculations to come.

Corollary 4.7.

$$A_x^\top \Sigma_x A_x \preceq \nabla^2 V_x \preceq 5 A_x^\top \Sigma_x A_x.$$

Proof. The lower bound is a direct consequence of Fact 4.5.

For the upper bound, note that for a graph Laplacian, it follows from the inequality $(a-b)^2 \leq 2(a^2+b^2)$ that

$$\Lambda_x \preceq 2(\Sigma_x + P_x^{(2)}).$$

Thus,

$$\nabla^2 V_x = A_x^\top (3\Sigma_x - 2P_x^{(2)}) A_x = A_x^\top (\Sigma_x + 2\Lambda_x) A_x \preceq 5 A_x^\top \Sigma_x A_x.$$

□

We now let $Q_x \stackrel{\text{def}}{=} A_x^\top \Sigma_x A_x$. The next lemma relates Q_x back the Hessian of the log-barrier function.

Lemma 4.8. $\frac{1}{4m} H_x \preceq Q_x \preceq H_x$.

This lemma is responsible for us losing the grounds we gained by the weighted barrier function. If we could somehow improve the lower bound to $\frac{1}{100} H_x$, then we would achieve our goal of $\sqrt{n}$. In the next section we show how to alter the barrier function to get a lower bound of $\frac{n}{m} H_x$ which will enable us to get the bound of $(mn)^{1/4}$.

Proof. The upper bound follows straightforwardly from Fact 4.3 which states that $\Sigma_x \preceq I$.

For the lower bound, we first break the sum into two parts as follows

$$\begin{aligned} Q_x &= \sum_i \frac{\sigma_{x,i} a_i a_i^\top}{s_{x,i}^2} \\ &= \sum_{i:\sigma_{x,i} \geq 1/2m} \frac{\sigma_{x,i} a_i a_i^\top}{s_{x,i}^2} + \sum_{\sigma_{x,i} < 1/2m} \frac{\sigma_{x,i} a_i a_i^\top}{s_{x,i}^2} \\ &\succeq \sum_{i:\sigma_{x,i} \geq 1/2m} \frac{\sigma_{x,i} a_i a_i^\top}{s_{x,i}^2}. \end{aligned}$$

Thus,

$$Q_x \succeq \frac{1}{2m} \sum_{i:\sigma_{x,i} \geq 1/2m} \frac{a_i a_i^\top}{s_{x,i}^2} = \frac{1}{2m} \left(H_x - \sum_{i:\sigma_{x,i} < 1/2m} \frac{a_i a_i^\top}{s_{x,i}^2} \right). \tag{4.2}$$

Now note that by the definition of $\sigma_{x,i}$, we obtain for all i,

$$\frac{a_i a_i^\top}{s_{x,i}^2} \preceq \sigma_{x,i} H_x.$$

Thus,

$$\sum_{i:\sigma_{x,i}<1/2m} \frac{a_i a_i^\top}{s_{x,i}^2} \preceq \sum_{i:\sigma_{x,i}<1/2m} \sigma_{x,i} H_x \preceq \frac{1}{2m} \sum_i H_x \preceq \frac{1}{2} H_x.$$

Thus, plugging in this estimate in (4.2), we obtain that

$$Q_x \succeq \frac{1}{2m}\left(H_x - \frac{1}{2}H_x\right) = \frac{1}{4m} H_x.$$

This completes the proof of the lemma. □

This allows us to get the following crucial bounds on σ_x.

Lemma 4.9. $\frac{a_i^\top Q_x^{-1} a_i}{s_{x,i}^2} \leq \min\left\{\frac{1}{\sigma_{x,i}}, 4m\sigma_{x,i}\right\} \leq 2\sqrt{m}.$

Proof. From the lower bound estimate of Lemma 4.8, it follows that

$$\frac{a_i^\top Q_x^{-1} a_i}{s_{x,i}^2} \leq \frac{4m a_i^\top H_x^{-1} a_i}{s_{x,i}^2} = 4m\sigma_{x,i}.$$

The other inequality follows from the following simple inequality

$$\frac{\sigma_{x,i} a_i a_i^\top}{s_{x,i}^2} \preceq Q_x.$$

□

We can now conclude by stating the key properties of the volumetric barrier function necessary to get a desired bound on the number of iterations of the IPM:

Theorem 4.10.

(1) $\nabla V_x^\top (\nabla^2 V_x)^{-1} \nabla V_x \leq n.$
(2) For all $x, y \in \text{int}(P)$ such that $\|x - y\|_x \leq \frac{1}{8m^{1/2}}$,

$$\nabla^2 V_x \overset{O(1)}{\approx} \nabla^2 V_y.$$

Proof. For the first part, note that

$$\begin{aligned}
\zeta^\top \nabla V_x \nabla V_x^\top \zeta &= \langle \zeta, A_x^\top \sigma_x \rangle^2 \\
&= \langle \zeta, A_x^\top \Sigma_x 1 \rangle^2 \\
&= \langle \Sigma_x^{1/2} A_x \zeta, \Sigma_x^{1/2} 1 \rangle^2 \\
&\leq \left(\zeta^\top A_x^\top \Sigma_x A_x \zeta \right) \left(1^\top \Sigma_x 1 \right) \\
&\leq \left(\zeta^\top Q_x \zeta \right) n \\
&\leq n \left(\zeta^\top \nabla^2 V_x \zeta \right).
\end{aligned}$$

For the second part, we start by noting that $\|x-y\|_x \leq \delta$ is the same as $(x-y)^\top Q_x (x-y) \leq \delta^2$. This in turn implies that

$$|a_i^\top (x-y)|^2 \leq \delta^2 a_i^\top Q_x^{-1} a_i \leq \frac{1}{4} \cdot s_{x,i}^2.$$

Thus, an argument similar to what we did for the log-barrier implies that

$$\left| 1 - \frac{s_{y,i}}{s_{x,i}} \right| \leq \frac{1}{2}.$$

This implies that $H_x \approx H_y$ and $\sigma_{x,i} \approx \sigma_{y,i}$ for all i. Thus, $Q_x \approx Q_y$. This implies that $\nabla^2 V_x \approx \nabla^2 V_y$, completing the proof. □

4.3.2 The Hybrid Barrier

Now consider the barrier function

$$G_x = V_x + \frac{n}{m} F_x,$$

where V_x is the volumetric barrier function and F_x the log-barrier function.

$$\nabla^2 G_x = \nabla^2 V_x + \frac{n}{m} H_x.$$

This adding a scaled version of the log-barrier function changes nothing significantly in the previous section. However, crucially this addition implies, via Lemmas 4.7 and 4.8 that

$$\nabla^2 G_x \succeq \frac{n}{m} H_x$$

rather than $\frac{1}{4m}H_x$ in the case of volumetric barrier. This implies that the upper bound in Lemma 4.9 comes down to $\sqrt{m/n}$. This then implies that, while the first part of Theorem 4.10 continues to hold with $4n$ instead of n, the second part holds with $(mn)^{1/4}$ rather than $\sqrt{m}$. Hence, from Theorem 4.1, we obtain an $(mn)^{1/4}$ iteration IPM for LPs.

One can interpret Vaidya's technique as starting with a barrier F and replacing it with

$$-\frac{1}{2}\log\det\nabla^2 F(x) + \frac{n}{m}F(x).$$

One may ask if repeating this helps improve the bound of $(mn)^{1/4}$. In particular, it is an interesting question to understand the *fixed point* of this functional equation and to what extent is it self-concordant.

4.4 Appendix: The Universal Barrier

In this section we introduce Nesterov-Nemirovskii's result that every bounded and convex body K in n-dimensions admits an $O(n)$ -self concordant barrier function. This implies that the number of iterations scale like $\sqrt{n}$. This not only improves the self-concordance results for polytopes we have proved till now, but also significantly generalizes it to *all* convex bodies. However, as we will see, computationally it is not very attractive. Designing a barrier function that does not require much more computation than solving linear system of equations, at least for polytopes, remains an outstanding open problem.

What is their barrier function that works in such a general setting? It is a volumetric barrier; however, not of any ellipsoid, rather of the *polar* of the body K centered at the current point x. More precisely, for a point $x \in \text{int}(K)$, let

$$K_x^\circ \stackrel{\text{def}}{=} \{z : z^\top (y - x) \leq 1 \ \ \forall y \in K\}.$$

Then, the Nesterov-Nemirovskii barrier function is defined to be

$$F(x) \stackrel{\text{def}}{=} \log \text{vol} K_x^\circ.$$

Since K is bounded, this is well-defined in the interior of K. Moreover, it is easy to see that as x approaches the boundary of K from the interior, the volume blows up to infinity. For instance, if K is the interval $[0,1]$ then $K_x^\circ = \left[-\frac{1}{x}, 0\right] \cup \left[0, \frac{1}{1-x}\right]$. Thus, as $x \to 1$, the right end of the polar goes to $+\infty$ and as $x \to 0$, the left end of the polar goes to $-\infty$. In either case, the volume goes to infinity.

In fact this one-dimensional intuition goes a long way in understanding $F(x)$. This is due to the following simple observation which allows us to think of K_x° as a collection of line segments. Let $v \in \mathbb{S}^{n-1}$ be a unit vector and let $p(v)$ denote the maximum over $y \in K$ of $v^\top y$. Thus, we can rewrite the polar as a collection of the following *line segments*; each along a unit vector v.

$$K_x^\circ = \bigcup_{v \in \mathbb{S}^{n-1}} \left[0, \frac{1}{p(v) - v^\top x}\right] v.$$

This representation allows us to easily rewrite the volume of K_x° as

$$f(x) \stackrel{\text{def}}{=} \text{vol } K_x^\circ = \frac{1}{n} \int_{v \in \mathbb{S}^{n-1}} (p(v) - v^\top x)^{-n} dS(v)$$

where $dS(v)$ is the $(n-1)$-dimensional volume of the surface of the unit sphere around v.

What we achieve by this transformation is an understanding of the derivatives of $f(x)$ in terms of *moment integrals* over the polar. In fact the following claim can now be seen easily:

$$D^l f(x)[h, h, \ldots, h] = \frac{(-1)^l (n+l)!}{n!} I_l(h)$$

where

$$I_l(h) \stackrel{\text{def}}{=} \int_{K_x^\circ} (z^\top h)^l dz.$$

However, we are interested in the differentials of $F(x) = \log f(x)$. How do we calculate the differentials of this function? This is also easy since

$$\frac{d \log g(x)}{dx} = \frac{1}{g(x)} \frac{dg(x)}{dx}.$$

This allows us to easily derive the following expressions:

(1) $DF(x)[h] = -(n+1)\frac{I_1(h)}{I_0}$.

(2) $D^2F(x)[h,h] = (n+1)(n+2)\frac{I_2(h)}{I_0} - (n+1)^2\frac{I_1^2(h)}{I_0^2}$.

(3) $D^3F(x)[h,h,h] = -(n+3)(n+2)(n+1)\frac{I_3(h)}{I_0} + 3(n+2)(n+1)^2\frac{I_2(h)I_1(h)}{I_0^2} - 2(n+1)^3\frac{I_1(h)^3}{I_0^3}$.

It follows that $F(x)$ is convex since a straightforward calculation shows that

$$D^2F(x)[h,h] > 0.$$

Further, it is not difficult to see that

$$|DF(x)[h,h]| \le \sqrt{n+1}\sqrt{|D^2F(x)[h,h]|},$$

which establishes the required bound on the complexity parameter. The self-concordance property remains to be proved; that does not seem to have a very illuminating proof and hence we omit the details here.

In summary, we sketched the proof of the fact that log-volume of the polar is a $\sqrt{n}$ self-concordant barrier function. Could this be improved beyond $\sqrt{n}$? The answer turns out to be no in general. This is as far as the theory of self-concordant barriers takes us. However, that is not to say that interior point methods cannot take us further, at least in the case of combinatorial polytopes!

4.5 Appendix: Calculating the Gradient and the Hessian of the Volumetric Barrier

In this section we give a sketch of the gradiant and Hessian calculation of the volumetric barrier. In particular, we prove the followin lemma mentioned earlier.

Lemma 4.11. (1) $\nabla V_x = A_x^\top \sigma_x$ and
(2) $\nabla^2 V_x = A_x^\top(3\Sigma_x - 2P_x^{(2)})A_x$.

Proof. Recall that $H_x = \sum_i \frac{a_i a_i^\top}{s_{x,i}^2}$ and $V_x = \frac{1}{2}\log\det H_x$. Let ζ be a vector and let $t \in \mathbb{R}$ be an infinitesimal. First note that

$$H_{x+t\zeta} - H_x = \sum_i \frac{a_i a_i^\top}{s_{x+t\zeta,i}^2} - \sum_i \frac{a_i a_i^\top}{s_{x,i}^2} = 2t\sum_i \frac{a_i a_i^\top}{s_{x,i}^2}\frac{a_i^\top \zeta}{s_{x,i}} + O(t^2).$$

Let $\Delta \overset{\text{def}}{=} 2t \sum_i \frac{a_i a_i^\top}{s_{x,i}^2} \frac{a_i^\top \zeta}{s_{x,i}}$. Thus, for an infinitesimal t,

$$\begin{aligned}
\frac{1}{2}\log\det H_{x+t\zeta} - \frac{1}{2}\log\det H_x &\approx \frac{1}{2}\log\det(H_x + \Delta) - \frac{1}{2}\log\det H_x \\
&= \frac{1}{2}\log\det H_x^{1/2}(I + H_x^{-1/2}\Delta H_x^{-1/2})H_x^{1/2} \\
&\quad -\frac{1}{2}\log\det H_x \\
&= \frac{1}{2}\log\det(I + H_x^{-1/2}\Delta H_x^{-1/2}) \\
&\approx \frac{1}{2}\mathrm{Tr}(H_x^{-1/2}\Delta H_x^{-1/2}) \\
&= t\mathrm{Tr}\left(H_x^{-1/2}\sum_i \frac{a_i a_i^\top}{s_{x,i}^2}\frac{a_i^\top \zeta}{s_{x,i}} H_x^{-1/2}\right) \\
&= t\sum_i \sigma_{x,i}\frac{a_i^\top \zeta}{s_{x,i}}.
\end{aligned}$$

Thus, $\nabla V_x = A_x^\top \sigma_x$.

To get the formula for the Hessian, start by noting that

$$\begin{aligned}
\frac{\partial^2 V_x}{\partial x_k \partial x_l} &= \frac{\partial}{\partial x_k} e_l^\top A_x^\top \sigma_x = \frac{\partial}{\partial x_k}\sum_i a_i^\top H_x^{-1} a_i \frac{A_{il}}{s_{x,i}^3} \\
&= \sum_i \left(\frac{A_{il}}{s_{x,i}^3}\cdot \frac{\partial}{\partial x_k} a_i^\top H_x^{-1} a_i + a_i^\top H_x^{-1} a_i \cdot \frac{\partial}{\partial x_k}\frac{A_{il}}{s_{x,i}^3}\right) \\
&= \sum_i \left(\frac{A_{il}}{s_{x,i}^3}\cdot \frac{\partial}{\partial x_k} a_i^\top H_x^{-1} a_i + 3 a_i^\top H_x^{-1} a_i \cdot \frac{A_{il}A_{ik}}{s_{x,i}^4}\right).
\end{aligned}$$

It remains to compute $\frac{\partial}{\partial x_k} a_i^\top H_x^{-1} a_i$. Towards this first note that similar to the calculation we did above,

$$H_{x+t\zeta}^{-1} - H_x^{-1} \approx -H_x^{-1}\Delta H_x^{-1}.$$

Thus, $a_i^\top H_{x+t\zeta}^{-1} a_i - a_i^\top H_x^{-1} a_i = -a_i^\top H_x^{-1}\left(2t\sum_j \frac{a_j a_j^\top}{s_{x,j}^2}\frac{a_j^\top \zeta}{s_{x,j}}\right) H_x^{-1} a_i = -2t\sum_j \frac{(a_i^\top H_x^{-1} a_j)^2}{s_{x,j}^2}\frac{a_j^\top \zeta}{s_{x,j}}$. Thus,

$$\frac{A_{il}}{s_{x,i}^3}\cdot \frac{\partial}{\partial x_k} a_i^\top H_x^{-1} a_i = -2\sum_j \frac{(a_i^\top H_x^{-1} a_j)^2}{s_{x,j}^2 s_{x,i}^2}\frac{A_{jk}}{s_{x,j}}\frac{A_{il}}{s_{x,i}} = -2A_x^\top P_x^{(2)} A_x.$$

Thus taking $\zeta = e_k$ and letting $t \to 0$, we obtain

$$\nabla^2 V_x = A_x^\top (3\Sigma_x - 2P_x^{(2)}) A_x.$$

□

4.6 Appendix: John ellipsoid

In this section we introduce the John ellipsoid and contrast it with the Dikin ellipsoid introduced in the previous lecture. In particular, we show a fundamental result that, for symmetric polytopes, the John ellipsoid $\sqrt{n}$ approximates the body (as opposed to the $\sqrt{m}$ achieved by the Dikin ellipsoid). Thus, just as we did with the Dikin ellipsoid, one can in principle define "The John Algorithm" and hope that it converges in $\sqrt{n}$ iterations. Whether it does remains far from clear. Recently, Lee-Sidford show that a *smoothened* version of John ellipsoid actually does converge in $\sqrt{n}\log^{O(1)}(m)$ iterations.

Definition 4.12. Given a bounded polytope $P \overset{\text{def}}{=} \{x \in \mathbb{R}^n : a_i^\top x \leq b_i \text{ for } i = 1, \ldots, m\}$ and a point $x \in \text{int}(P)$, the *John ellipsoid* of P at x is defined to be the ellipsoid of maximum volume which is centered at x and contained in P.

Let us try to describe the John ellipsoid $\mathcal{E}_x$ centered at a point x using a quadratic form. Towards this, let B_x be a PSD matrix such that

$$\mathcal{E}_x = \{y : \|y - x\|_{B_x}^2 \leq 1\}.$$

Then, the constraint that $\mathcal{E}_x$ is contained inside the polytope P would require that for every $i = 1, \ldots, m$

$$\mathcal{E}_x \subseteq \{y : \langle a_i, y \rangle \leq b_i\}.$$

In other words for all i,

$$\max_{y \in \mathcal{E}_x} \langle a_i, y \rangle \leq b_i$$

or equivalently

$$\max_{y : y^\top B_x y \leq 1} \langle a_i, y + x \rangle \leq b_i$$

or equivalently

$$\max_{y: y^\top B_x y \leq 1} \langle a_i, y\rangle \leq b_i - \langle a_i, x\rangle = s_i(x).$$

It can be checked that the left-hand side is equal to $\|a_i\|_{B_x^{-1}}$. Because $s_i(x) \geq 0$, we can square both sides and obtain the constraint

$$a_i^\top B_x^{-1} a_i \leq s_i(x)^2.$$

As for the volume of $\mathcal{E}_x$ (which we are maximizing), we have that $\mathrm{vol}(\mathcal{E}_x)$ equals

$$\mathrm{vol}(\{y : y^\top B_x y \leq 1\}) = \mathrm{vol}(\{B_x^{-1/2} v : \|v\|_2 \leq 1\}) = V_n \left(\det(B_x^{-1/2})\right)$$

where V_n is the volume of the unit ℓ_2 ball in $\mathbb{R}^n$. Ignoring the V_n term (by just redefining the volume relative to V_n) we obtain

$$\log \mathrm{vol}\ \mathcal{E}_x = \frac{1}{2} \log\det(B_x^{-1}).$$

We take the logarithm here because this will allow us to obtain a convex program.

We can now write the following program, called (John-Primal):

$$\begin{aligned} \min \quad & -\log\det B^{-1} \\ \text{s.t.} \quad & \frac{a_i^\top B^{-1} a_i}{s_i(x)^2} \leq 1 \qquad \text{for } i = 1, \ldots, m. \end{aligned}$$

Note that we deal with the constraint $B \succ 0$ (equivalently, $B^{-1} \succ 0$) by encoding it into the domain of the function $\log\det$.

Let us denote $C = B^{-1}$. The program (John-Primal) is convex in the variables $\{C_{ij}\}$.

4.6.1 Duality

Let us now write the dual program (John-Dual). We multiply the i-th constraint by a multiplier $w_i \geq 0$, obtaining the Lagrangian

$$\begin{aligned} L(C, w) &= -\log\det C + \sum_i w_i \left(\frac{a_i^\top C a_i}{s_i(x)^2} - 1\right) \\ &= -\log\det C + C \bullet \left(\sum_i \frac{w_i}{s_i(x)^2} a_i a_i^\top\right) - \sum_i w_i. \end{aligned}$$

The KKT optimality condition $\nabla_C L(C, w) = 0$ yields

$$-C^{-1} + \sum_i \frac{w_i}{s_i(x)^2} a_i a_i^\top = 0$$

and thus

$$B = C^{-1} = \sum_i \frac{w_i}{s_i(x)^2} a_i a_i^\top .$$

The dual objective function is

$$g(w) = \inf_C L(C, w) = -\log\det \left(\sum_i \frac{w_i}{s_i(x)^2} a_i a_i^\top \right)^{-1} + n - \sum_i w_i,$$

with constraints $w_i \geq 0$ and $\sum_i \frac{w_i}{s_i(x)^2} a_i a_i^\top \succ 0$ (the latter arising from our restriction of $\log\det$ to PD matrices).

Slater's condition holds (just consider a small ball certifying that $x \in \operatorname{int}(P)$) and we have strong duality. Thus, at optimal values of C and w:

$$-\log\det C = -\log\det C - \sum_i w_i + n,$$

and hence

$$\sum_i w_i = n.$$

We can interpret w_i as a measure of how strongly the ellipsoid $\mathcal{E}_x$ is supported on the hyperplane defined by the i-th constraint. For example, from the complementary slackness conditions we have that if $a_i^\top B^{-1} a_i < s_i(x)^2$ (i.e., the ellipsoid does not touch the hyperplane), then $w_i = 0$.

4.6.2 Approximating symmetric polytopes

Now we show that the John ellipsoid improves the $\sqrt{m}$ of the Dikin ellipsoid to $\sqrt{n}$ for symmetric polytopes.

Proposition 4.13. *Let P be a polytope which is symmetric around the origin (i.e., $P = -P$), $0 \in \operatorname{int}(P)$, and let $\mathcal{E}$ be the John ellipsoid of P at 0. Then the ellipsoid $\sqrt{n}\mathcal{E}$ contains P.*

Proof. The proof resembles the one of a similar statement for Dikin ellipsoids. Without loss of generality assume that all $b_i = 1$. Also, because P is symmetric, we assume that for every constraint $\langle a_i, x\rangle \leq 1$ there is a corresponding constraint $\langle -a_i, x\rangle \leq 1$. Now all $s_i(x) = 1$ and the John ellipsoid $\mathcal{E}$ is given by

$$B = \sum_i w_i a_i a_i^\top, \quad \mathcal{E} = \{x : x^\top B x \leq 1\} = \left\{x : \sum_i w_i \langle a_i, x\rangle^2 \leq 1\right\}.$$

Pick a point x on the boundary of $\mathcal{E}$. It is enough to show that $\sqrt{n}x$ is not inside K. Since $x \in \partial\mathcal{E}$, we have

$$\sum_i w_i \langle a_i, x\rangle^2 = 1.$$

If it were the case that $\sqrt{n}x \in \operatorname{int}(P)$, then for every i we would have $\langle a_i, \sqrt{n}x\rangle < 1$ and $\langle -a_i, \sqrt{n}x\rangle < 1$, so that

$$\left\langle a_i, \sqrt{n}x\right\rangle^2 < 1,$$

and multiplying by w_i/n and summing over all i we would get

$$\sum_i w_i \langle a_i, x\rangle^2 = \sum_i \frac{w_i}{n} \left\langle a_i, \sqrt{n}x\right\rangle^2 < \frac{\sum_i w_i}{n} = 1,$$

a contradiction. □

4.6.3 The John Algorithm?

Thus, the John ellipsoid provides us with an alternative to the Dikin ellipsoid and one may ask what is stopping us from defining a barrier similar to Vaidya's volumetric barrier. After all, it does give us a barrier of the form

$$\sum_i w_i(x) \frac{a_i a_i^\top}{s_i(x)^2}$$

with weights summing up to n at every point raising the possibility of a $\sqrt{n}$-iteration algorithm. Unfortunately, the weights $w_i(x)$ are not even continuous (let alone differentiable). For example, consider the square $[-1, 1]^2$ defined by the four natural constraints (suppose w_1 corresponds to $x \leq 1$ and w_2 corresponds to $-x \leq 1$). At a point $(\varepsilon, 0)$, with ε very small, we have $w_1 = 1$ and $w_2 = 0$, but at $(-\varepsilon, 0)$ the converse is true. However, Lee-Sidford were able to find a workaround by *smoothening* the weights at a poly-log cost in the number of iterations. The details are quite involved and omitted here.

5

A Primal-Dual IPM for Linear Programs (without the Barrier)

5.1 Overview

In this lecture we will present an elementary proof of Ye's primal-dual interior point method (IPM) for solving linear programming (LP) problems. The primal-dual approach presented here can be cast in the language of *central path* using a barrier function. The formulation and analysis here are simpler and make no reference to *barrier functions, central paths, Newton step, affine scaling, predictor step, corrector step, centering* etc., however an avid who is familiar with path following methods is encouraged to see the similarities.

5.2 Linear Programming and Duality

It is well-known that any LP problem can be formulated as below, along with the corresponding dual program.

Primal (P)

$$\begin{array}{lr} \min & \langle c, x\rangle \\ \text{s.t.} & \\ & Bx = b \\ \forall e \in [m] & x_e \geq 0 \end{array}$$

Dual (D)

$$\begin{array}{lr} \max & \langle b, y\rangle \\ \text{s.t.} & \\ & B^\top y + s = c \\ \forall e \in [m] & s_e \geq 0 \end{array}$$

Here, m is the number of variables and n is the number of constraints, i.e.,

$$B \in \mathbb{R}^{n\times m}, \quad b \in \mathbb{R}^n, \quad c \in \mathbb{R}^m, \quad x \in \mathbb{R}^m, \quad y \in \mathbb{R}^n, \quad s \in \mathbb{R}^m.$$

Example 5.1. As an important example, if we think of B as a vertex-edge incidence matrix[1] of a directed graph G with edge costs c, then the primal problem above encodes the MinCostUncapacitatedFlow in G with demands given by the vector b and the cost by c. In view of this application, we will be indexing coordinates of x and s with the letter e.

Motivated by this example, we will present our method in the context of solving a combinatorial problem, that is, one where the input data B, b, c contains only entries of bounded length (e.g. only from the set $\{-1, 0, 1\}$). Therefore, the encoding length L of the instance will not be a factor in our considerations. Moreover, we will not solve programs (P) and (D) to optimality; instead, once we are able to bring the duality gap $\langle c, x\rangle - \langle b, y\rangle$ down to $O(m^{-1})$, we will stop and assume that the

[1] That is, $B_{ve} = -1$ and $B_{we} = 1$ for $e = \langle v, w\rangle$.

solution we output will be good enough for the purpose at hand (for example, that it can be rounded to an optimal solution).[2]

Consider an optimal primal-dual solution (x, y, s). It satisfies all four (sets of) conditions from the programs (P) and (D), as well as the following *complementary slackness* condition:

$$\forall e \in [m] \qquad\qquad x_e s_e = 0.$$

Moreover, it follows from the theory of linear programming that any triple (x, y, s) satisfying all these constraints, that is,

$$\begin{aligned} & & Bx &= b \\ & & B^\top y + s &= c \\ \forall e \in [m] & & x_e &\geq 0 \\ \forall e \in [m] & & s_e &\geq 0 \\ \forall e \in [m] & & x_e s_e &= 0 \end{aligned}$$

is in fact an optimal primal-dual solution (i.e. it satisfies $\langle c, x\rangle = \langle b, y\rangle$).

Thus, we have turned the task of finding the optimum solution of a linear program into the task of finding a feasible solution to a quadratic program. At first sight, this does not seem promising, especially given that the new program is not even convex. Nevertheless, we will be able to come up with a polynomial-time algorithm for approximately solving this new program. The following is the main result:

Theorem 5.2. There is an algorithm, which produces a solution (x, y, s) which is feasible for the primal-dual linear program mentioned above such that $\langle c, x\rangle - \langle b, y\rangle \leq \delta$ in $O(\sqrt{m} \log {}^m\!/\!_\delta)$ iterations. Each iteration corresponds to solving an $m \times m$ positive semi-definite (PSD) linear system of equations.

[2] Extremal solution to linear programming problems have this property that they are rational with the denominator bounded by $2^{O(L)}$, where L is the encoding length. Further, for many combinatorial polytopes, the vertices have all coordinates 0/1. Due to this, once the duality gap is small enough, of the order $2^{-O(L)}$, one can *round* the solutions to get an actual vertex solution. For the MinCostUncapacitatedFlow problem, a duality gap of $m^{-O(1)}$ suffices. This is left as an easy exercise.

5.3 A Primal-Dual Interior Point Method

The most important challenge seems to be handling the quadratic constraints $x_e s_e = 0$. To this end, we will introduce a parameter $\mu \in \mathbb{R}_+$, which is going to serve as an approximation of the 0 in these constraints. And we define the relaxed program called KKT(μ) as follows:[3]

$$
\begin{aligned}
& & Bx = b & \qquad (5.1)\\
& & B^\top y + s = c & \qquad (5.2)\\
&\forall e \in [m] & x_e \geq 0 & \qquad (5.3)\\
&\forall e \in [m] & s_e \geq 0 & \qquad (5.4)\\
&\forall e \in [m] & x_e s_e = \mu & \qquad (5.5)
\end{aligned}
$$

It is easy to see that as $\mu \to 0$, the optimal solution to $KKT(\mu)$ tends to the optimal solution of the underlying linear program. The rough idea of our approach is to start with an initial solution (x^0, s^0, y^0) satisfying KKT(μ) for a large value of μ. Then we hope to turn it into solutions (x^t, s^t, y^t) of programs KKT(μ) for smaller and smaller values of μ using some iterative process. When μ is small enough, we can stop. Why?

Fact 5.3. The duality gap $\langle c, x\rangle - \langle b, y\rangle$ is equal to $\sum_e x_e s_e = \langle x, s\rangle$.

Proof.

$$
\begin{aligned}
\langle x, c\rangle - \langle b, y\rangle &= \left\langle x, B^\top y + s\right\rangle - \langle Bx, y\rangle\\
&= \left\langle x, B^\top y\right\rangle + \langle x, s\rangle - \left\langle x, B^\top y\right\rangle\\
&= \langle x, s\rangle .
\end{aligned}
$$

□

Therefore, a setting of $\mu = O(m^{-2})$ would give us a duality gap of $\sum_e x_e s_e = m\mu = O(m^{-1})$, which, as we discussed before, we are assuming to be good enough.

[3] The KKT stands for Karush-Kuhn-Tucker. The reason we call it KKT(μ) is because these *are* the KKT-conditions for the following convex programming problem obtained from our primal LP by removing the $x \geq 0$ constraints and replacing them by the log barrier function: $\inf\langle c, x\rangle - \mu \sum_e \ln x_e$ subject to $Ax = b$.

Unfortunately, this is still too difficult a task. Namely, approximating 0 by μ is not enough: it is unreasonable to expect that we will be able to make all the terms $x_e s_e$ equal to any given parameter (and also, we are still left with a quadratic constraint). Indeed, we will not try to solve KKT(μ) exactly. Instead, we will allow some leeway in the constraints $x_e s_e = \mu$, but we will take care to measure the total relative error in the approximate equality $(x_e s_e)_e \approx \mu \cdot \mathbb{1}$ and keep it under control.

We will encode this measurement using the following potential function:

$$v \overset{\text{def}}{=} v(x, s, \mu) \overset{\text{def}}{=} \left\| \left(\frac{x_e s_e - \mu}{\mu} \right)_e \right\|_2$$

That is,

$$v(x, s, \mu) = \sqrt{\sum_e \left(\frac{x_e s_e}{\mu} - 1 \right)^2}.$$

Our main invariant is that we will keep this quantity bounded from above by $1/2$ at all times. In other words, *we will always be close to the actual solution of KKT(μ).*[4] Hence, the program KKT'(μ) that we will be solving is the following:

$$\begin{array}{lr} & Bx = b \\ & B^\top y + s = c \\ \forall e \in [m] & x_e \geq 0 \\ \forall e \in [m] & s_e \geq 0 \\ & v(x, s, \mu) \leq \dfrac{1}{2} \end{array}$$

It can be checked that one can find an initial solution (x^0, s^0, y^0) to the above problem for some $\mu^0 = m^{O(1)}$.[5] Let us quickly verify that this relaxation is good enough for our purposes.

[4] The solution of KKT(μ) traces a path in the interior of the primal polytope and the dual polytope simultaneoulsy as $\mu \to 0$ continuously. This is the *primal-dual central path.*

[5] Actually, we will need to introduce a constant number of new x, y, s variables (with very high costs) that enable us to find a starting solution.

Fact 5.4. Let $\mu = O(m^{-2})$. If (x, y, s) is a solution of the program KKT'(μ), then the duality gap is $O(m^{-1})$.

Proof. We have that

$$v(x, s, \mu) = \left\| \left(\frac{x_e s_e - \mu}{\mu} \right)_e \right\|_2 \leq \frac{1}{2},$$

hence[6]

$$\left\| \left(\frac{x_e s_e - \mu}{\mu} \right)_e \right\|_1 \leq \frac{\sqrt{m}}{2},$$

and hence,

$$\sum_e (x_e s_e - \mu) \leq \frac{\mu\sqrt{m}}{2}$$

and, using Theorem 5.3, the duality gap is

$$\sum_e x_e s_e \leq \frac{\mu\sqrt{m}}{2} + \mu m = O(m^{-1}).$$

□

Hence, the gist of our approach will be to produce solutions of the program KKT'(μ) for values of μ that decrease geometrically until we reach $\mu = O(m^{-2})$.

We will always maintain the invariant that x and s are strictly positive, i.e., that we are in the interior of the primal and dual bodies. This is why this method can be considered an interior point method. Let (x^t, s^t, y^t) be the solutions at iteration t.

$$\text{Invariant}: \quad x^t \succ 0, \quad s^t \succ 0. \tag{5.6}$$

Again, we are tacitly assuming that we are able to somehow produce an initial solution (x^0, y^0, s^0) for KKT'(μ) satisfying $x^0 \succ 0$, $s^0 \succ 0$, for some setting of μ which is polynomial in m.

[6] We are using the basic fact that $\|v\|_1 \leq \sqrt{m}\|v\|_2$ (actually, a factor of m would suffice).

We will now describe a single update step. It will consist of two stages:

$$\begin{array}{lrll} \text{from} & (x,y,s,\mu) & \overset{\text{def}}{=} (x^t,y^t,s^t,\mu^t) & \text{s.t. } v \le 1/2 \\ \text{through} & (x+\Delta x, y+\Delta y, s+\Delta s, \mu) & \overset{\text{def}}{=} (x^{t+1},y^{t+1},s^{t+1},\mu^t) & \text{s.t. } v \le 1/4 \\ \text{to} & (x+\Delta x, y+\Delta y, s+\Delta s, \mu(1-\gamma)) & \overset{\text{def}}{=} (x^{t+1},y^{t+1},s^{t+1},\mu^{t+1}) & \text{s.t. } v \le 1/2 \end{array}$$

That is: in the first stage, we will try to bring v down as much as possible by changing our solution (but keeping μ intact). In the second stage we will decrease μ as much as possible while still keeping v below $1/2$ (we decrease μ using the multiplicative update $\mu^{t+1} \overset{\text{def}}{=} (1-\gamma)\mu^t$). Thus our task amounts to finding (for the first stage) a way to decrease v, and (for the second stage) a maximum value of γ which will guarantee $v \le 1/2$.

Let us focus on the first stage for now. We have a solution which satisfies all constraints of KKT(μ) except for (5.5), and our goal is to satisfy this constraint (as much as possible). We have denoted by $(\Delta x, \Delta s, \Delta y)$ the change in the solution. Let us write the old and new linear equality constraints:

$$\begin{aligned} Bx &= b \\ B^\top y + s &= c \\ B(x+\Delta x) &= b \\ B^\top(y+\Delta y) + s + \Delta s &= c \end{aligned}$$

Straightforward subtraction shows that the two new ones will be satisfied iff

$$\begin{aligned} B\Delta x &= 0, \\ B^\top \Delta y + \Delta s &= 0. \end{aligned}$$

Ideally, we would want to satisfy (5.5), that is, to have

$$(x_e + \Delta x_e)(s_e + \Delta s_e) = \mu.$$

But this would lead to having to solve a program with a quadratic constraint – precisely what we are trying to avoid. Thus instead, we use the following as an approximation:

$$\forall e \in [m], \quad x_e s_e + \Delta x_e s_e + x_e \Delta s_e = \mu.$$

I.e., we just dropped the quadratic term $\Delta x_e \Delta s_e$.[7]

Thus, the linear system of equations **LS** which we need to solve to find $(\Delta x, \Delta s, \Delta y)$ is:

$$B\Delta x = 0 \tag{5.7}$$

$$B^\top \Delta y + \Delta s = 0 \tag{5.8}$$

$$\forall_e \in [m], \qquad x_e s_e + \Delta x_e s_e + x_e \Delta s_e = \mu. \tag{5.9}$$

Note that we should still enforce that $x + \Delta x, s + \Delta s \succ 0$. But it turns out that this is true for any solution, as Theorem 5.9 will show.

We begin by showing that this system has a solution.

Lemma 5.5. The linear system LS has a solution.

Proof. Let[8] $L \stackrel{\text{def}}{=} BXS^{-1}B^\top$, and let $z \stackrel{\text{def}}{=} -\frac{1}{\mu}Bs^{-1}$, where we denote $\left(s^{-1}\right)_e \stackrel{\text{def}}{=} s_e^{-1}$. Then Δy can be found by solving:

$$L\Delta y = z.$$

It can be checked that this has a solution since $x, s \succ 0$ and, by definition, z lies in the column space of B. Vectors Δs and Δx are now determined by (5.8) and (5.9), respectively:

$$\Delta s = -B^\top \Delta y,$$
$$(\Delta x)_e = \frac{\mu - x_e \Delta s_e}{s_e} - x_e.$$

We are left with verifying that $B\Delta x = 0$, which is an easy exercise. □

[7] This is the Newton step which approximates the function by its first order Taylor series and finds the best direction to move.

[8] We use the notation that for a vector x, let X denote the diagonal matrix with the entries of x in the diagonal.

Example 5.6. Coming back to Theorem 5.1, we can see that the matrix $BXS^{-1}B^\top$ from the proof of Theorem 5.5 is the Laplacian matrix of a graph where each edge e has conductance $\frac{x_e}{s_e}$. Solving the system for Δy (which is the only computationally nontrivial step in solving the system LS) amounts to computing an electrical flow in this graph. It can be done in nearly-linear time using a fast approximate Laplacian solver due to Spielman-Teng. It is important to note that the Laplacian solver of Spielman-Teng can find a solution in $\tilde{O}(m \log {}^1\!/\!_\varepsilon)$ time where ε is the relative error of the solution. It can be checked that an approximate solution with $\varepsilon = {}^1\!/\!_{m^{O(1)}}$ suffices for our purpose. In summary, our method solves the MinCostUncapacitatedFlow problem by computing a sequence of electrical flows where the resistances and the demand vector depend on the current point.

Now let us fix a solution $(\Delta x, \Delta y, \Delta s)$ and proceed to analyzing its properties. We begin with a simple but very useful observation.

Lemma 5.7. If $(\Delta x, \Delta s)$ satisfy LS, then

$$\sum_{e \in [m]} \Delta x_e \Delta s_e = 0.$$

As a corollary, on average the duality gap is satisfied:

$$\frac{1}{m} \sum_e (x_e + \Delta x_e)(s_e + \Delta s_e) = \mu.$$

The proof of the lemma is immediate.

Proof. Start by multiplying (5.8) by $\Delta x^\top$ and use (5.7) to obtain

$$0 = \Delta x^\top B^\top \Delta y + \Delta x^\top \Delta s = (B\Delta x)^\top \Delta y + \Delta x^\top \Delta s = \Delta x^\top \Delta s.$$

□

The following crucial lemma establishes that potential reduces quadratically[9] and lets us analyze the first stage.

[9] This is similar to the analysis of Newton's method (see Lecture 3) and this similarity is not coincidental. The Taylor approximation we performed can be seen as a Newton step

Lemma 5.8. For (x, s, μ) such that $v(x, s, \mu) < 1$ and $\Delta x, \Delta s$ that satisfy LS, we have

$$v(x + \Delta x, s + \Delta s, \mu) \leq \frac{1}{2} \cdot \frac{v(x, s, \mu)^2}{1 - v(x, s, \mu)}.$$

It follows that if $v(x, s, \mu) \leq \frac{1}{2}$, then $v(x + \Delta x, s + \Delta s, \mu) \leq 1/2 \cdot 1/2 \cdot 1/2 \cdot 2 = 1/4$. The proof relies on the Cauchy-Schwarz inequality and Theorem 5.7.

Proof. Define $dx_e \stackrel{\text{def}}{=} \sqrt{\frac{s_e}{\mu x_e}} \Delta x_e$ and $ds_e \stackrel{\text{def}}{=} \sqrt{\frac{x_e}{\mu s_e}} \Delta s_e$. Thus,

$$dx_e + ds_e = -\sqrt{\frac{\mu}{x_e s_e}} \left(\frac{x_e s_e}{\mu} - 1 \right).$$

Hence, $v(x + \Delta x, s + \Delta s, \mu)^2 = \sum_{e \in [m]} dx_e^2 ds_e^2$ which is at most

$$\frac{1}{4} \sum_{e \in [m]} (dx_e^2 + ds_e^2)^2 \leq \frac{1}{4} \left(\sum_{e \in [m]} (dx_e + ds_e)^2 \right)^2 = \frac{1}{4} \max_{e \in [m]} \left(\frac{\mu}{x_e s_e} \right)^2 v(x, s, \mu)^4.$$

Here we have used that

$$\sum_{e \in [m]} (dx_e + ds_e)^2 = \sum_{e \in [m]} dx_e^2 + ds_e^2 + 2 dx_e ds_e = \sum_{e \in [m]} dx_e^2 + ds_e^2$$

since $\sum_{e \in [m]} dx_e ds_e = 0$. Thus, for all e, $1 - \frac{x_e s_e}{\mu} \leq v(x, s, \mu)$. This implies that $1 - v(x, s, \mu) \leq \frac{x_e s_e}{\mu}$. Since $v(x, s, \mu) < 1$, $\max_{e \in [m]} \left(\frac{\mu}{x_e s_e} \right) \leq \frac{1}{1 - v(x, s, \mu)}$. This completes the proof of the lemma. □

As we remarked above, the following fact proves that just solving this linear system (consisting only of equalities) is actually enough to guarantee that the linear inequalities are satisifed as well (i.e. that we are in the interior of the primal and dual bodies).

for the system of quadratic equations we started with and the condition that $v \leq 1/2$ corresponds to the domain of quadratic convergence in Newton's method.

Lemma 5.9. If (x, s, μ) are such that $v(x+\Delta x, s+\Delta s, \mu) < 1$, where $\Delta x, \Delta s$ satisfy LS, then $x + \Delta x > 0$ and $s + \Delta s > 0$.

Proof. $v(x+\Delta x, s+\Delta s, \mu) < 1$ implies that for all e, $-\mu < \Delta x_e \Delta s_e < \mu$. Thus,

$$(x_e + \Delta x_e)(s_e + \Delta s_e) = \mu + \Delta x_e \Delta s_e > 0.$$

Thus, for each e, both $x_e + \Delta x_e$ and $s_e + \Delta s_e$ are either both positive or both negative. If they are both negative then,

$$\Delta x_e < -x_e \quad \text{and} \quad \Delta s_e < -s_e.$$

Since $x_e > 0$ and $s_e > 0$, multiplying the above inequalities by s_e and x_e respectively and adding we obtain

$$\Delta x_e s_e + \Delta s_e x_e < -2 x_e s_e.$$

Thus, $\mu = \Delta x_e s_e + \Delta s_e x_e + x_e s_e < -x_e s_e$ which is a contradiction as $\mu > 0$. □

Finally, the following lemma lets us analyze the second stage. It gives an estimate of the rate of convergence of the whole method. That is, it answers the question of what values of γ are guaranteed to keep v below $1/2$.

Lemma 5.10. For (x, s, μ) such that $v(x, s, \mu) < 1$ and $\Delta x, \Delta s$ that satisfy LS we have

$$v(x+\Delta x, s+\Delta s, (1-\gamma)\mu) \leq \frac{1}{1-\gamma} \cdot \sqrt{v(x+\Delta x, s+\Delta s, \mu)^2 + \gamma^2 m}.$$

Thus, if $v(x, s, \mu) \leq 1/2$ and consequently $v(x+\Delta x, s+\Delta s, \mu) \leq 1/4$, then we must take $\gamma = \Omega\left(1/\sqrt{m}\right)$ in order to maintain the invariant that $v(x+\Delta x, s+\Delta s, (1-\gamma)\mu) \leq 1/2$.

Proof.

$$\begin{aligned} v(x+\Delta x, s+\Delta s, (1-\gamma)\mu)^2 &= \sum_{e\in[m]} \left(\frac{(x_e+\Delta x_e)(s_e+\Delta s_e)}{\mu(1-\gamma)} - 1 \right)^2 \\ &= \frac{1}{(1-\gamma)^2} \left(\sum_{e\in[m]} \left(\frac{\Delta x_e \Delta s_e}{\mu} + \gamma \right)^2 \right) \\ &= \frac{1}{(1-\gamma)^2} \left(\sum_{e\in[m]} \left(\frac{\Delta x_e \Delta s_e}{\mu} \right)^2 \right. \\ &\qquad \left. + \sum_{e\in[m]} \frac{2\Delta x_e \Delta s_e \gamma}{\mu} + m\gamma^2 \right). \end{aligned}$$

Using Theorem 5.7, we obtain

$$\begin{aligned} v(x+\Delta x, s+\Delta s, (1-\gamma)\mu)^2 &= \frac{1}{(1-\gamma)^2} \sum_{e\in[m]} \left(\frac{\Delta x_e \Delta s_e}{\mu} \right)^2 + \frac{m\gamma^2}{(1-\gamma)^2} \\ &= \frac{v(x+\Delta x, s+\Delta s, \mu)^2}{(1-\gamma)^2} + \frac{m\gamma^2}{(1-\gamma)^2}. \end{aligned}$$

□

To finish, we calculate the number of iterations is takes to bring $\mu^0 = \text{poly}(m)$ down to $\mu^T = O(m^{-2})$. Note that after T iterations we have $\mu_T = \mu^0(1-\gamma)^T$, and hence, we need $(1-\gamma)^T = \text{poly}(m^{-1})$. Since $\gamma = \Omega\left(\frac{1}{\sqrt{m}}\right)$, a constant decrease in μ is brought about by $O(\sqrt{m})$ iterations, and we need to iterate this $O(\log m)$ times. We conclude that

$$T = O\left(\sqrt{m}\log m\right).$$

Each iteration involves solving a system of linear equations.

Example. Coming back to Theorems 5.1 and 5.6, notice that our result, together with a fast Laplacian solver, implies that we are able to solve the MinCostUncapacitatedFlow problem in time $\tilde{O}\left(m^{3/2}\right)$.

References

[AHK12] Sanjeev Arora, Elad Hazan, and Satyen Kale. The multiplicative weights update method: a meta-algorithm and applications. *Theory of Computing*, 8(1):121–164, 2012.

[Bub14] Sebastien Bubeck. Theory of convex optimization for machine learning. Internet draft, 2014. URL: `http://arxiv.org/abs/1405.4980`.

[Haz14] Elad Hazan. Introduction to online convex optimization. Internet draft, 2014. URL: `http://ocobook.cs.princeton.edu/`.

[Vis13] Nisheeth K. Vishnoi. Lx = b. *Foundations and Trends in Theoretical Computer Science*, 8(1-2):1–141, 2013. URL: `http://research.microsoft.com/en-us/um/people/nvishno/site/Lxb-Web.pdf`.

[Wri05] Margaret H. Wright. The interior-point revolution in optimization: history, recent developments, and lasting consequences. *Bull. Amer. Math. Soc. (N.S*, 42:39–56, 2005.