6.S978 Graphs, Linear Algebra, and Optimization – Fall 2015 September 22, 2015

Lecture 3

Lecturer: Aleksander Mądry *Scribe: Tal Wagner*

1 Overview

In this lecture, we analyze the gradient descent algorithm. This is a first-order optimization method that enables us to leverage certain smoothness property of the minimized function to achieve a convergence rate that is better than the one delivered by the (projected) sub-gradient descent algorithm we analyzed last time.

2 Maximum Flow Problem as an Optimization Problem

Our motivating combinatorial optimization problem is the maximum flow problem. Given an input undirected graph $G(V, E)$ with $|V| = n$ vertices, $|E| = m$ edges, and $s, t \in V$, we formulated it as follows:

$$\begin{aligned} \min \quad & \|f\|_\infty \\ \text{s.t.} \quad & Bf = \chi_{s,t}, \end{aligned} \tag{1}$$

where $B \in \mathbb{R}^{n \times m}$ is a *signed edge-vertex incidence matrix* of G, defined as

$$B_{v,e} := \begin{cases} -1 & v \text{ is the head of } e \\ 1 & v \text{ is the tail of } e \\ 0 & \text{otherwise} \end{cases}, \tag{2}$$

under an arbitrary orientation of the edges of G, and $\chi_{s,t} \in \mathbb{R}^V$ is defined as

$$\chi_{s,t}(v) := \begin{cases} -1 & v = s \\ 1 & v = t \\ 0 & \text{otherwise} \end{cases}. \tag{3}$$

At this point, we want to abstract away the specifics of the the maximum flow problem, and view it as a general convex program:

$$\begin{aligned} \min \quad & f(x) \\ \text{s.t.} \quad & x \in \mathcal{K}. \end{aligned} \tag{4}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is a convex function, and $K \subset \mathbb{R}^n$ is a convex set.

As mentioned earlier, in principle, we could apply generic convex programming machinery, such as the Ellipsoid algorithm to solve this general problem. However, this would result in a rather slow algorithm. Instead, we attempt to use gradient descent–type strategies to get a much improved performance, but at a price of delivering worse quality of approximation.

3 Gradient Descent and Projected Gradient Descent Algorithms

Recall that if we are dealing with an unconstrained minimization problem, i.e., have $\mathcal{K} = \mathbb{R}^n$, the *gradient descent* (GD) algorithm takes the form presented in Algorithm 1.

Algorithm 1 Gradient descent algorithm

```
$x_1 \leftarrow \vec{0}$
for $s = 1, \ldots, T-1$ do
    $x_{s+1} \leftarrow x_s - \eta \nabla f(x_s)$
end for
return $x_T$
```

In words, we initialize our estimate x_1 to an arbitrary feasible point, e.g., an all-zeros vector $\vec{0}$, and then make a sequence of T local improvements. Each of these improvements corresponds to taking a step of in the direction opposite to the gradient of the current estimate, $-\nabla f(x_s)$, with the step size modulated by the parameter η. Since the gradient points in the steepest direction upwards, we know that taking a step in the opposite direction yields the best local improvement toward minimization.

Observe that the final answer returned by gradient descent is simply the last point x_T and not the average $\bar{x}_T := \frac{1}{T}\sum_{s=1}^{T} x_s$ of all the computed points, as was the case in the subgradient descent algorithm we analyzed last time. In fact, one could have a variant of the gradient descent in which one also returns $\bar{x}_T$ instead of x_T but its convergence would be slower. After all, we know that if a sequence converges then the sequence of averages converges as well but at a slower rate. So, as long as we can control the direct convergence – which we couldn't do in the setting of subgradient descent but will be able to do now – then working with that convergence is always preferable.

Now, if we want to adapt gradient descent to the constrained setting, i.e., when $\mathcal{K}$ is a proper (convex) subset of $\mathbb{R}^n$, we again need to employ the notion of projection to keep each successive steps within our feasible set $\mathcal{K}$. To this end, recall the following definition

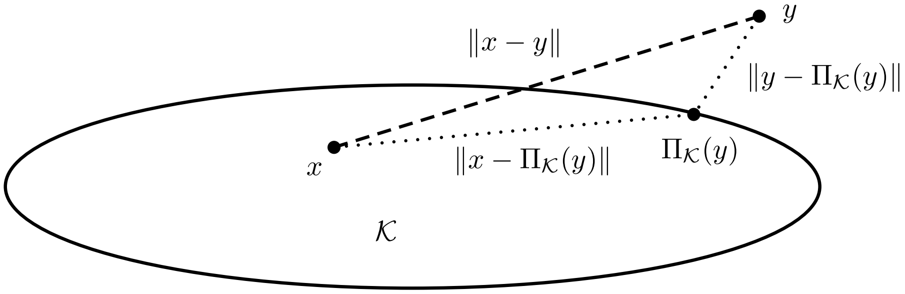
Definition 1 *(ℓ_2-) projection For a convex set $\mathcal{K} \subseteq \mathbb{R}^n$ and a point $y \in \mathcal{K}$, let us define*

$$\Pi_{\mathcal{K}}(y) := \operatorname{argmin}_{x \in \mathcal{K}} \|x - y\|_2.$$

The crucial property of the projection is captured by the following fact.

Fact 2 *For any $x \in \mathcal{K}$ and $y \in \mathbb{R}^n$, $(\Pi_{\mathcal{K}}(y) - x)^T (\Pi_{\mathcal{K}}(y) - y) \leq 0$.*

Geometrically speaking, the above fact tells us that the angle between the lines formed between x and the projection of y; and y and its projection is always obtuse – see Figure 1.



y
$\|x - y\|$
$\|y - \Pi_{\mathcal{K}}(y)\|$
x
$\|x - \Pi_{\mathcal{K}}(y)\|$
$\Pi_{\mathcal{K}}(y)$
$\mathcal{K}$

Figure 1: Illustration for Fact 2.

Now, armed with the notion of projection we can "fix" the gradient descent strategy from Algorithm 1 to make it work with the constraints imposed by the feasible set $\mathcal{K}$. The resulting *projected gradient descent* (abbrev. PGD) algorithm is presented as Algorithm 2.

4 L-Smoothness

Clearly, to make the (projected) gradient descent algorithms well-defined, we need to assume that the objective function f is differentiable. Otherwise, the gradients $\nabla f(x)$ might not exist sometime. In fact,

Algorithm 2 Projected gradient descent algorithm

```
H
  $x_1 \leftarrow \vec{0}$
  for $s = 1, \ldots, T-1$ do
      $x_{s+1} \leftarrow \Pi_K \left( x_s - \eta \nabla f(x_s) \right)$
  end for
  return $x_T$
```

in order to develop rigorous quantitative bounds on the convergence guarantees of these algorithms, we will need to work with the following quantitative version of differentiability

Definition 3 (L-smoothness) *We say f is* L-smooth*, for some $L \geq 0$ iff*

$$\forall\, x, y, \quad \|\nabla f(x) - \nabla f(y)\|_2 \leq L \cdot \|x - y\|_2.$$

The above property should be compared with the property of f that we needed for the analysis of the projected subgradient descent algorithm. There, we needed a bound G that acted as a Lipschitz constant of f (first-order condition), whereas here L is a Lipschitz constant of ∇f, which is a second-order condition.

How can leverage L-smoothness for our needs? In general, when analyzing the local improvements approach of GD-type algorithms, we need to use *local* information about f to infer *global* information about it. For example, the convexity of f tells us that if the gradient is zero *locally* at a point x, then x is a *global* optimum. We would like to get stronger implication of this type, and the implication we get from L-smoothness is encompassed by the following lemma.

Lemma 4 *If f is convex and L-smooth, then*

$$\forall\, x, y \in \mathcal{D}, \quad 0 \leq f(y) - f(x) - \nabla f(x)^T (y - x) \leq \frac{L}{2} \cdot \|x - y\|_2^2.$$

To get some intuition for this technical statement, let us recall the Taylor expansion of $f(y)$ around x:

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

The convexity of f implies that if we omit all the terms beyond the linear one from the right-hand side, it can only decrease

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

(To see this, recall that $\nabla f(x)$ is a subgradient of f.) This gives a lower bound on $f(y) - f(x)$ that acts as a measure for the global quality of x as an estimate of the optimum. The left-hand side in Lemma 4 is just the difference between the quantity $f(y) - f(x)$ and its lower bound $\nabla f(x)^T (y - x)$ (which is a *first-order* bound), and the lemma states a bound on this difference (and thus a *second-order* bound), in terms of the smoothness constant L.

Proof [of Lemma 4] The left-hand side inequality follows directly from the convexity of f, as explained above.

To establish the right-hand side inequality, let us note that by Cauchy-Schwarz inequality we have that

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

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

5 The Analysis of the Gradient Descent Algorithm

With the notion of L-smoothness and Lemma 4 at hand, we are ready to analyze the performance of GD approaches. For now, we restrict our attention to the unconstrained version of the program Eq. (4), $\mathcal{K} = \mathbb{R}^n$ and analyze the corresponding gradient descent algorithm as presented in Algorithm 1. We will prove the following guarantee for GD in this setting:

Theorem 5 *Suppose f is L-smooth. If we set $\eta = \frac{1}{L}$, then the output x_T of the Algorithm 1 satisfies*

$$f(x_T) - f(x^*) \leq O\left(\frac{L \cdot R^2}{T}\right),$$

where $R := \|x_1 - x^*\|_2$.

Note the substantial gain over the analogous $\frac{RL}{\sqrt{T}}$ convergence bound of subgradient descent. The inverse-dependence of T is improved from square-root to linear. Also, note that the definition of R is modified from the radius of $\mathcal{K}$ to the distance from the initial estimate, since we are now in the setting $\mathcal{K} = \mathbb{R}^n$, so $\mathcal{K}$ is not compact and has no finite radius.

Before delving into the details of the proof, let us get some sense of what we are aiming at. We begin with the following observation:

Observation 6 *For every s,*

$$f(x_s) - f(x_{s+1}) \geq \frac{1}{2L}\|\nabla f(x_s)\|_2^2.$$

Proof Apply Lemma 4 with $x = x_s$ and $y = x_{s+1}$, and recall that $x_s - x_{s+1} = \eta \nabla f(x_s) = \frac{1}{L}\nabla f(x_s)$. The observation follows by simple manipulations. ■

This is a lower bound on the progress of GD in each step, in terms the gradient norm at that step. Indeed, we have already mentioned several times that the gradient norm at a point x acts as a measure for the distance of x from the optimum x^*. If the norm is large, then we might be far from the optimum, but Observation 6 guarantees we make large progress in the current step; if the norm is small, then we have a poor guarantee for the progress of the current step, but this is okay since we are already quite close to the optimum. This is the trade-off that we wish to build on.

5.1 Proof of Theorem 5

For $s = 1, \dots, T-1$, define

$$\delta_s := f(x_s) - f(x^*).$$

We have

$$\delta_s = f(x_s) - f(x^*) \leq \nabla f(x_s)^T (x_s - x^*) \leq \|\nabla f(x_s)\|_2 \cdot \|x_s - x^*\|_2, \tag{5}$$

where the first inequality follows from the left-hand-side inequality of Lemma 4, and the second inequality is Cauchy-Schwartz. Hence

$$\delta_s - \delta_{s+1} \geq \frac{1}{2L} \|\nabla f(x_s)\|_2^2 \geq \frac{1}{2L} \cdot \frac{\delta_s^2}{\|x_s - x^*\|_2^2}, \tag{6}$$

where the first inequality is Observation 6 and the second inequality is by Eq. (5). The difficulty now is to handle the term $\|x_s - x^*\|_2^2$ in the denominator. It is tempting to bound $\|x_s - x^*\|_2$ by R, but this turns out to be a subtle point. The definition $R := \|x_1 - x^*\|_2^2$ does not generally imply $\|x_s - x^*\|_2^2 \leq R$ for every s. In our setting, however, we can obtain this bound by relying on the L-smoothness of f and on the careful choice of the step size $\eta = \frac{1}{L}$. Let us record it as a lemma:

Lemma 7 *For every s, $\|x_s - x^*\|_2 \leq R$.*

To keep the presentation clear, we now complete the proof of Theorem 5 relying on the Lemma 7. The lemma will be proven in the next subsection.

Plugging Lemma 7 into Eq. (6), we get

$$\delta_s - \delta_{s+1} \geq \frac{\delta_s^2}{2LR^2},$$

and hence

$$\frac{1}{\delta_{s+1}} - \frac{1}{\delta_s} = \frac{\delta_s - \delta_{s+1}}{\delta_s \delta_{s+1}} \geq \frac{\delta_s - \delta_{s+1}}{\delta_s^2} \geq \frac{1}{2LR^2}.$$

Summing over all $s = 1, \dots, T-1$, the left-hand side telescopes and we get

$$\frac{1}{\delta_T} - \frac{1}{\delta_1} \geq \frac{T-1}{2LR^2}. \tag{7}$$

We bound δ_1:

$$\delta_1 = f(x_1) - f(x^*) \leq \nabla f(x^*)^T (x_1 - x^*) + \frac{L}{2} \|x_1 - x^*\|_2^2 \leq \frac{LR^2}{2},$$

where the first inequality is by Lemma 4, and the second inequality by noting that $\nabla f(x^*) = 0$ and $R = \|x_1 - x^*\|$. By plugging this back into Eq. (7) and rearranging, the proof of Theorem 5 is complete:

$$\delta_T \leq O\left(\frac{LR^2}{T}\right).$$

5.2 Proof of Lemma 7

We will prove the following statement which immediately implies Lemma 7:

$$\forall s, \quad \|x_{s+1} - x^*\|_2 \leq \|x_s - x^*\|_2. \tag{8}$$

We highlight the difference between Eq. (8) and the convergence guarantee of the algorithm (which is stated in Theorem 5): the latter states that the sequence of evaluations $\{f(x_s)\}$ converges to the evaluation $f(x^*)$ (in $\mathbb{R}$), whereas Eq. (8) is concerned with distance of the actual points $\{x_s\}$ from x^* (in $\mathbb{R}^n$).

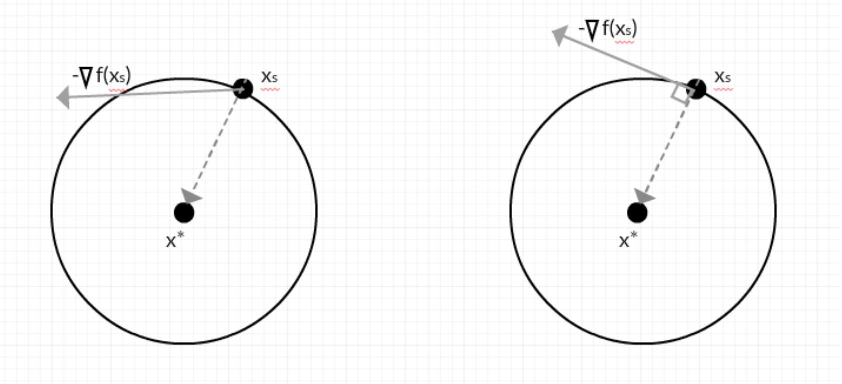
First, let us see why Eq. (8) is not true in general (without relying on the L-smoothness and on the step size). Visualize the ball centered at x^* with radius $\|x_s - x^*\|$. We are standing at x_s, and are about to step to x_{s+1}. Eq. (8) holds if step keeps us inside that ball. The actual direction in which we step is $-\nabla f(x)$, while the "correct" direction is towards x^*, i.e. into the center of the ball. The convexity of f ensures us that the angle between the correct direction and the actual direction cannot be obtuse. Specifically, it follows from the subgradient condition

$$f(x) - f(y) \le \nabla f(x)^T (x - y)$$

after plugging $x = x_s$ and $y = x^*$.

However, the direction might still, in principle, be (close to) orthogonal. In such case, *any* sufficiently large step we take in direction $-\nabla f(x_s)$ will increase the distance from x^* and violate Eq. (8) – see the right diagram in Figure 2 for illustration. So, only sufficiently small steps satisfy Eq. (8).

Figure 2:



The proof of Eq. (8) therefore needs to leverage the L-smoothness to infer that the angle is acute, and that our selected step size $\eta = \frac{1}{L}$ is indeed sufficiently small. The former is encompassed in the following lemma.

Lemma 8 *For every $x, y \in \mathbb{R}^n$,*

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

Proof

■

To prove Eq. (8), let us use the above lemma with $x = x_s$ and $y = x^*$ to obtain

$$\nabla f(x_s)^T (x_s - x^*) \ge \frac{1}{L} \|\nabla f(x_s)\|_2^2, \tag{9}$$

where we used the fact that $\nabla f(x^*) = 0$. We can now use this inequality to conclude that

$$\begin{aligned} \|x_{s+1} - x^*\|_2^2 &= \|x_s - \tfrac{1}{L}\nabla f(x_s) - x^*\|_2^2 \\ &= \|x_s - x^*\|_2^2 + \|\tfrac{1}{L}\nabla f(x_s)\|_2^2 - 2 \cdot \tfrac{1}{L}\nabla f(x_s)^T (x_s - x^*) \\ &\le \|x_s - x^*\|_2^2 + \|\tfrac{1}{L}\nabla f(x_s)\|_2^2 - 2 \cdot \tfrac{1}{L^2}\|\nabla f(x_s)\|_2^2 \\ &\le \|x_s - x^*\|_2^2, \end{aligned}$$

which is exactly (8). Thus, Lemma 7 follows.

6 Conclusion and Future Lectures

Theorem 5 provides an improved bound for GD relying on the additional L-smoothness assumption, but it suffers some limitations. First, we formulated it only to the unconstrained setting $K = \mathbb{R}^n$, which does not cover our motivating the maximum flow problem problem. In order to extend it to the general constrained setting, we need to re-introduce the projection step into the algorithm – that is, to analyze PGD. The difficulty is that the projection might shrink distances, destroying our progress lower bound in Observation 6, which was crucial to the analysis. It turns out, however, that an appropriate extension of the analysis to a projected version can overcome this obstacle.

Once this hurdle is behind us, we are faced with the more challenging limitation of Theorem 5: it relies on the existence of gradients, i.e. on the differentiability of f. The maximum flow problem formulation in Eq. (1) uses the ℓ_∞-norm, which is not differentiable everywhere. Our approach to this issue will be to approximate the ℓ_∞-norm with a smooth objective function, which inevitably will be lossy, since the program we are optimizing will no longer be an accurate formulation of the maximum flow problem. The challenge will be to balance the loss in the approximation step against the gain from the smoothness of the approximate objective function.