Convex Optimization and Max Flow

Optimization and Machine Learning Seminar, NTUA December 2018

These notes are intended as reference notes for participants of the Optimization and Machine Learning Seminar in NTUA. Thus, they are in no way complete or free of errors. Please exercise critical thinking and don't take anything for granted.

1. Convex Optimization

In Convex Optimization one wishes to (approximately) minimize a convex function. Specifically, given $f : X \subseteq \mathbb{R}^m \to \mathbb{R}$ such that f is convex and some allowed error $\epsilon > 0$, the goal is to compute an x such that $f(x) \leq f(x^*) + \epsilon$, where $x^* = \underset{x}{\operatorname{argmin}}\ f(x)$.

One of the most common procedures to achieve this goal is to use an iterative algorithm, in which one produces a sequence of estimates $x^0, \ldots, x^T$, each one given as a function of the previous ones. More specifically, the most well known such algorithm is probably the *gradient descent algorithm*, given by

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

where $\eta \in \mathbb{R}$ is called a *step size* and is there to scale the gradient appropriately. Despite its simplicity, the gradient descent algorithm has proven to be extremely versatile.

Most algorithms of the form described above require f to satisfy some extra properties in order to get a meaningful bound on the number of iterations. The most well known such properties are the following:

- **G-Lipschitzness**:

  $$|f(y) - f(x)| \leq G\|y - x\|_2 \ \forall x, y \in X$$

  or equivalently, if f is differentiable,

  $$\|\nabla f(x)\|_2 \leq G \ \forall x \in X$$

- **L-smoothness**:

  $$\|\nabla f(y) - \nabla f(x)\|_2 \leq L\|y - x\|_2 \ \forall x, y \in X$$

  or equivalently, if f is twice differentiable,

  $$z^T \nabla^2 f(x) z \leq L \ \forall x \in X, z \in \mathbb{R}^m$$

- **ℓ-strong convexity**:

  $$\|\nabla f(y) - \nabla f(x)\|_2 \geq \ell\|y - x\|_2 \ \forall x, y \in X$$

  or equivalently, if f is twice differentiable,

  $$z^T \nabla^2 f(x) z \geq \ell\|z\|_2^2 \ \forall x \in X, z \in \mathbb{R}^m$$

- **Bounded radius**:

  $$\|x^0 - x^*\|_2 \leq D$$

(Note: Some of the above definitions require a (twice) differentiable function but can be appropriately modified for functions that are not)

The main mathematical tool behind these algorithms is *Taylor's theorem*, from which we use the following corollary:

Theorem 1.1. If f is twice differentiable and for appropriately small $\|y-x\|_2$, we have

$$f(y) = f(x) + \nabla f(x)^T(y-x) + \frac{1}{2}(y-x)^T\nabla^2 f(x)(y-x) + o(\|y-x\|_2^2)$$

The following are some properties that directly follow from the above definitions and Taylor's theorem.

Lemma 1.2. f is convex iff $f(y) \geq f(x) + \nabla f(x)^T(y-x)$ for all $x, y \in X$.

Lemma 1.3. f is L-smooth iff $f(y) \leq f(x) + \nabla f(x)^T(y-x) + \frac{1}{2}L\|y-x\|_2^2$ for all $x, y \in X$.

Lemma 1.4. f is ℓ-strongly convex iff $f(y) \geq f(x) + \nabla f(x)^T(y-x) + \frac{1}{2}\ell\|y-x\|_2^2$ for all $x, y \in X$.

2. Results

2.1. **Smooth functions.** Let's begin by assuming that our function is L-smooth wrt the ℓ_2 norm. We will see that we will also need a bounded radius.

This means that

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

and naturally, given x^t, one would want to minimize $f(x^{t+1})$, or instead its upper bound

$$f(x^t) + \nabla f(x^t)^T(x^{t+1}-x^t) + \frac{L}{2}\|x^{t+1}-x^t\|_2^2$$

Therefore setting $x^{t+1} = x^t + \delta$, δ will be given by

$$\min_\delta f(x^t)^T\delta + \frac{L}{2}\|\delta\|_2^2 \Leftrightarrow \nabla f(x^t) + L\delta = 0 \Leftrightarrow \delta = -\frac{1}{L}\nabla f(x^t)$$

so the step is

$$x^{t+1} = x^t - \frac{1}{L}\nabla f(x^t)$$

which is gradient descent with $\eta = \frac{1}{L}$.

How to measure progress? The natural thing is to track the decrease in function value after every step, i.e. establish a lower bound on $f(x^t) - f(x^{t+1})$. From the L-smoothness property, we have

$$\begin{aligned} f(x^t) - f(x^{t+1}) &\geq -\nabla f(x^t)^T(x^{t+1}-x^t) - \frac{L}{2}\|x^{t+1}-x^t\|_2^2 \\ &= \frac{1}{2L}\|\nabla f(x^t)\|_2^2 \end{aligned}$$

This is good news, since it is saying "Large gradients imply fast convergence". What happens when the gradient is small? Convexity should imply that we are already close to optimality. Specifically,

$$f(x^t) - f(x^*) \leq \nabla f(x^t)^T(x^t - x^*) \leq \|\nabla f(x^t)\|_2\|x^t - x^*\|_2$$

As we will show, $\|x^t - x^*\|_2 \leq D$, and so

$$f(x^t) - f(x^{t+1}) \geq \frac{1}{2L}\|\nabla f(x^t)\|_2^2 \geq \frac{(f(x^t) - f(x^*))^2}{2LD^2}$$

Denoting $y(t) = f(x^t) - f(x^*)$ and extending it linearly to real values of t it suffices to solve

$$\frac{dy(t)}{dt} = -\frac{(y(t))^2}{2LD^2} \Leftrightarrow \frac{dy(t)}{(y(t))^2} = -\frac{dt}{2LD^2} \Leftrightarrow \frac{1}{y(t)} = \frac{t}{2LD^2} + c$$

which gives us

$$f(x^t) - f(x^*) \leq \frac{2LD^2}{t}$$

Since we want ϵ error it's enough to guarantee

$$\frac{2LD^2}{T} \leq \epsilon \Leftrightarrow T \geq \frac{2LD^2}{\epsilon}$$

so $O\left(\frac{LD^2}{\epsilon}\right)$ iterations are enough for optimization of smooth functions.

Note: This bound can be improved to $O\left(\sqrt{\frac{LD^2}{\epsilon}}\right)$ using Nesterov's *accelerated gradient descent*.

To finish off, we need to show that $\|x^t - x^*\|_2 \leq D$ for all t. We will show something stronger, i.e. that $\|x^{t+1} - x^*\|_2 \leq \|x^t - x^*\|_2$ and the result will follow by induction. Specifically,

$$\begin{aligned}
&\|x^{t+1} - x^*\|_2^2 - \|x^t - x^*\|_2^2 \\
&= \|x^{t+1} - x^t\|_2^2 + 2(x^{t+1} - x^t)^T(x^t - x^*) \text{ (cosine theorem)} \\
&\leq \|x^{t+1} - x^t\|_2^2 + \frac{2}{L}(f(x^*) - f(x^t)) \text{ (smoothness and choice of gradient)} \\
&\leq \|x^{t+1} - x^t\|_2^2 + \frac{2}{L}(f(x^{t+1}) - f(x^t)) \\
&\leq \|x^{t+1} - x^t\|_2^2 - \frac{1}{L^2}\|\nabla f(x^t)\|_2^2 \text{ (choice of gradient)} \\
&= 0
\end{aligned}$$

2.2. **Smooth and strongly convex functions.** If our function is ℓ-strongly convex in addition to L-smooth, we get much faster convergence. In particular, we run the same algorithm as for the previous section so we know that

$$f(x^t) - f(x^{t+1}) \geq \frac{1}{2L}\|\nabla f(x^t)\|_2^2$$

but now we can get a much better lower bound for the RHS because of ℓ-strong convexity:

$$\|\nabla f(x^t)\|_2^2 \geq \frac{(f(x^t) - f(x^*))^2}{\|x^t - x^*\|_2^2} \geq \frac{\ell}{2}(f(x^t) - f(x^*))$$

Therefore

$$f(x^t) - f(x^{t+1}) \geq \frac{\ell}{4L}(f(x^t) - f(x^*)) = \frac{1}{4\kappa}(f(x^t) - f(x^*))$$

where $\kappa = \frac{L}{\ell}$ is called the *condition number* of f. So now

$$\begin{aligned}
f(x^{t+1}) - f(x^*) \leq &(1 - \frac{1}{4\kappa})(f(x^t) - f(x^*)) \\
\leq &\cdots \\
\leq &(1 - \frac{1}{4\kappa})^{t+1}(f(x^0) - f(x^*)) \\
\leq &e^{-\frac{t}{4\kappa}}(f(x^0) - f(x^*))
\end{aligned}$$

(where we used $1 - x \leq e^{-x}$) and so to achieve ϵ error $T = O\left(\kappa \log \frac{f(x^0) - f(x^*)}{\epsilon}\right)$ iterations are enough. Note: This bound can be improved to $O\left(\sqrt{\kappa} \log \frac{f(x^0) - f(x^*)}{\epsilon}\right)$ using Nesterov's *accelerated gradient descent*.

2.3. **Lipschitz functions.** If our function is not smooth but is G-Lipschitz lower bounding the function value decrease in each iteration as above is not possible. Instead, we use a potential argument in the lines of "While the function value is large, we are moving fast towards the optimum". The algorithm will be (sub)gradient descent (because f might not be differentiable, $\nabla f(x)$ denotes any vector g such that $f(y) \geq f(x) + g^T(y-x)$ for all y), given by $x^{t+1} = x^t - \eta \nabla f(x^t)$, for some still undetermined step size η.

Now, convexity gives

$$\begin{aligned} f(x^t) - f(x^*) \leq & \nabla f(x^t)^T (x^t - x^*) \\ = & \frac{1}{\eta}(x^t - x^{t+1})^T (x^t - x^*) \end{aligned}$$

On the other hand, the decrease of the distance to the optimum is

$$\begin{aligned} \|x^t - x^*\|_2^2 - \|x^{t+1} - x^*\|_2^2 = & \|x^t - x^{t+1}\|_2^2 + 2(x^{t+1} - x^*)^T (x^t - x^{t+1}) \\ = & \|x^t - x^{t+1}\|_2^2 + 2(x^t - x^*)^T (x^t - x^{t+1}) + 2(x^{t+1} - x^t)^T (x^t - x^{t+1}) \\ = & -\|x^t - x^{t+1}\|_2^2 + 2(x^t - x^*)^T (x^t - x^{t+1}) \\ \geq & -\eta^2 \|\nabla f(x^t)\|_2^2 + \eta (f(x^t) - f(x^*)) \end{aligned}$$

so after T iterations this gives

$$\begin{aligned} \frac{1}{T} \sum_t (f(x^t) - f(x^*)) \leq & \frac{1}{\eta T} (\|x^0 - x^*\|_2^2 - \|x^T - x^*\|_2^2) + \eta G^2 \\ \leq & \frac{D^2}{\eta T} + \eta G^2 \\ \leq & \frac{DG}{\sqrt{T}} \end{aligned}$$

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

Since f is convex, setting $\bar{x} = \frac{1}{T} \sum_t x^t$ we get

$$f(\bar{x}) - f(x^*) \leq \frac{1}{T} \sum_t (f(x^t) - f(x^*)) \leq \frac{DG}{\sqrt{T}} \leq \epsilon$$

so $T = O(\frac{D^2 G^2}{\epsilon^2})$ iterations are enough.

2.4. **Lipschitz and strongly convex functions.** We proceed similarly to the previous section, only with a variable step size η_t, and get $T = O(\frac{G^2}{\ell \epsilon})$ iterations.

2.5. **Minimization over convex domain.** All of the results that we have seen so far can be adapted to work with *constrained* optimization problems, where x is constrained in some convex set X. This can be done by projecting to the feasible region after each gradient descent step. Note that it is often non-trivial whether this projection can be computed efficiently.

2.6. **Gradient Descent for general norms.** Gradient descent can be adapted to cope with functions which are smooth/strongly convex with respect to a different norm. The only thing that changes from the definitions is that gradients are measured in the *dual* norm and from the statements that the radius bound is not $\|x^0 - x^*\|$ but $\max_{x: f(x) \leq f(x^0)} \|x - x^*\|$. The algorithm also changes, since now the step is given by the minimization problem

$$\min_\delta f(x^t)^T \delta + \frac{L}{2} \|\delta\|^2$$

so for different norms $\|\cdot\|$, the step is in a different direction. For example, for the ℓ_∞ norm, we have

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

3. Max Flow using Gradient Descent

3.1. **Problem.** Given a graph $G(V, E)$ and a pair of designated nodes s and t, the Max Flow problem asks to send the maximum amount of flow from s to t while not exceeding the edge capacities. Here we will concentrate on the case of an undirected graph G (i.e. flow can go both ways) with unit capacities. However note that for convenience we will assign an arbitrary reference direction to edges in E, so we will be treating E as a set directed edges from now on.

We would like to encode this as a convex program. If we denote the flow on the graph by a vector $x \in \mathbb{R}^m$, where x_i is the flow on edge i, the capacity constraint is precisely $\|x\|_\infty \leq 1$. Now, given the max flow value F^*, the flow constraints are as follows:

$$\sum_{(u,v)\in E} x_{uv} - \sum_{(v,u)\in E} x_{vu} = \begin{cases} F^*, \text{ if } u \equiv s \\ -F^*, \text{ if } u \equiv t \\ 0, \text{ otherwise} \end{cases}$$

for all $u \in V$. These (linear) constraints can be written succinctly in the form $B^T x = F^* \chi_{st}$, where $B \in \mathbb{R}^{m\times n}$ is the "edge-vertex" incidence matrix of G, given by

$$B_{eu} = B_{(a,b),u} = \begin{cases} 1, \text{ if } u \equiv a \\ -1, \text{ if } u \equiv b \\ 0, \text{ otherwise} \end{cases}$$

and $\chi_{st} \in \mathbb{R}^n$ is the characteristic vector of the demand, given by

$$\chi_{st}(u) = \begin{cases} 1, \text{ if } u \equiv s \\ -1, \text{ if } u \equiv t \\ 0, \text{ otherwise} \end{cases}$$

Therefore we have reduced the problem to finding an $x \in \mathbb{R}^m$ such that $\|x\|_\infty \leq 1$ and $B^T x = F^* \chi_{st}$. If we divide everything by F^* equivalently we are asked to find an x such that $\|x\|_\infty \leq \frac{1}{F^*}$ and $B^T x = \chi_{st}$, or equivalently solving

$$\min \|x\|_\infty$$
$$B^T x = \chi_{st}$$

Note that since we are computing an approximate solution x, this will correspond to an approximate max flow, so to set up the problem appropriately we are asking for a $(1+\gamma)$-approximate max flow in the sense that $F \geq \frac{F^*}{1+\gamma}$. Note $\frac{1}{F} = \|x\|_\infty \leq \|x^*\|_\infty + \epsilon = \frac{1}{F^*} + \epsilon$ therefore setting $\epsilon = \frac{\gamma}{F^*}$ suffices.

3.2. **Computing the projection.** In order to be able to solve this with an iterative algorithm, we have to be able to project any $x \in \mathbb{R}^m$ on the affine constraint $B^T x = \chi_{st}$. In other words, we need to be able to solve

$$\min_{x'} \|x' - x\|_2^2$$
$$B^T x' = \chi_{st}$$

This can be solved by solving a linear system. However, our linear system has special structure, as it is a *Laplacian* system. By employing the seminal result of Spielman and Teng [ST04], we can

solve Laplacian systems in $\widetilde{O}(m \log \frac{1}{\epsilon})$ time, therefore we can compute these projections efficiently. (Note: Here ϵ is the error of the linear system solution, in some appropriately defined norm)

3.3. **Algorithm 1: Subgradient Descent.** Since $\|x\|_\infty$ is not a smooth or a strongly convex function, the only black box algorithm we can apply from the previous section is the Subgradient Descent. Remember that the number of iterations for to get an ϵ-approximate solution is $O(\frac{D^2G^2}{\epsilon^2})$.

We have

$$x^0 = \{\operatorname*{argmin}_x \|x - 0\|_2^2, \text{ s.t. } B^T x = \chi_{st}\}$$

and

$$D^2 = \|x^0 - x^*\|_2^2 \leq 2(\|x^0\|_2^2 + \|x^*\|_2^2) \leq 4\|x^*\|_2^2 \leq 4m\|x^*\|_\infty^2 \leq O\left(\frac{m}{(F^*)^2}\right)$$

Furthermore, since the absolute function is 1-Lipschitz, we have $G^2 = \sum_{i=1}^{m} 1^2 \leq m$. Combining everything together, we get

$$T = O\left(\frac{\frac{m}{(F^*)^2}m}{(\frac{\gamma}{F^*})^2}\right) = O\left(\frac{m^2}{\gamma^2}\right)$$

iterations and a total runtime of

$$\widetilde{O}\left(\frac{m^3}{\gamma^2}\right)$$

This matches (up to the fact that it is an $(1+\gamma)$-approximate max flow) the Edmonds-Karp algorithm for sparse graphs.

3.4. **Algorithm 2: Smoothing.** In order to exploit the GD analysis for smooth functions, it makes sense to replace $\|x\|_\infty$ by a smooth approximation of it. Specifically, we will use the *softmax function*, defined by

$$\text{smax}_\eta(x) = \eta \log\left(\frac{\sum_i \left(e^{x_i/\eta} + e^{-x_i/\eta}\right)}{2m}\right)$$

One can easily prove that smax_η is $\frac{1}{\eta}$-smooth with respect to the ℓ_2 norm. Furthermore,

$$\|x\|_\infty - \eta \log(2m) \leq \text{smax}_\eta(x) \leq \|x\|_\infty$$

This means that setting $\eta = \epsilon / \log(2m) = \frac{\gamma}{F^* \log(2m)}$ is enough to let us optimize $\text{smax}_\eta(x)$ instead of $\|x\|_\infty$.

Now we can apply the analysis for smooth functions, which gives

$$T = O\left(\frac{LD^2}{\epsilon}\right) = O\left(\frac{\frac{F^* \log(2m)}{\gamma}\frac{m}{(F^*)^2}}{\frac{\gamma}{F^*}}\right) = \widetilde{O}\left(\frac{m}{\gamma^2}\right)$$

for a total runtime of $\widetilde{O}\left(\frac{m^2}{\gamma^2}\right)$. Note, however, that we can improve this by using the guarantee of *accelerated* gradient descent! This gives us a runtime of $\widetilde{O}\left(\frac{m^{3/2}}{\gamma}\right)$ and matches (up to the fact that it is an $(1+\gamma)$-approximate max flow) Goldberg and Rao's long-standing runtime of $\widetilde{O}\left(m^{3/2}\right)$! [GR98]

3.5. **Algorithm 3: ℓ_∞-based Gradient Descent.** To further improve the runtime, the algorithm has to capture the geometry of the problem in a better way. In particular, smax_η is $\frac{1}{\eta}$-smooth with respect to the ℓ_∞ norm (a stronger condition than with respect to ℓ_2 norm), so we might as well run ℓ_∞-based gradient descent.

However, one problem is that computing an ℓ_∞ projection on the affine constraint $B^T x = \chi_{st}$ is hard (if we could do it efficiently, we would immediately solve our initial problem since x^* is the ℓ_∞ projection of 0 on the affine constraint). Fortunately, it turns out that there exists a certain linear projection operator P that performs an *approximate* ℓ_∞ projection onto the constraint $B^T x = \chi_{st}$. This operator is called an oblivious routing and has the property that $\|P\|_\infty \leq m^{o(1)}$.

Putting everything together carefully, one finally gets an almost-linear runtime of $\widetilde{O}\left(\frac{m^{1+o(1)}}{\gamma^2}\right)$ [KLOS14]. With more work it is also possible to get a nearly-linear runtime of $\widetilde{O}\left(\frac{m}{\gamma}\right)$ for this problem [Pen16, She17].

References

[GR98] Andrew V Goldberg and Satish Rao. Beyond the flow decomposition barrier. *Journal of the ACM (JACM)*, 45(5):783–797, 1998.

[KLOS14] Jonathan A Kelner, Yin Tat Lee, Lorenzo Orecchia, and Aaron Sidford. An almost-linear-time algorithm for approximate max flow in undirected graphs, and its multicommodity generalizations. In *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*, pages 217–226. SIAM, 2014.

[Pen16] Richard Peng. Approximate undirected maximum flows in o (m polylog (n)) time. In *Proceedings of the twenty-seventh annual ACM-SIAM symposium on Discrete algorithms*, pages 1862–1867. Society for Industrial and Applied Mathematics, 2016.

[She17] Jonah Sherman. Area-convexity, l &infty; regularization, and undirected multicommodity flow. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 452–460. ACM, 2017.

[ST04] Daniel A Spielman and Shang-Hua Teng. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pages 81–90. ACM, 2004.