0.1 Philosophy

Optimization methods often follow the following framework:

Algorithm 1: `OptimizationFramework`

for $k = 0, 1, \cdots$ **do**
  Approximate f by a simpler function f_k according to the current point $x^{(k)}$
  Do something using f_k (such as set $x^{(k+1)} = \arg\min_x f_k(x)$)
end

The runtime depends on the number of iterations and the cost per iteration. Philosophically, the difficulties of a problem can never be created nor destroyed, only converted from one form of difficulty to another. When we decrease the number of iterations, the cost per iteration often increase. The gain of new methods often come from avoiding some wasted computation, utilizing some forgotten information or giving a faster but tailored algorithm for a sub-problem. This is of course just an empirical observation.

One key question to answer in designing an optimization algorithm is what the problem looks like (or how can we approximate f by a simpler function). Here are some approximations we will use in this textbook:

- First-order Approximation: $f(y) \approx f(x) + \langle \nabla f(x), y - x \rangle$ (Section 0.2).
- Second-order Approximation: $f(y) \approx f(x) + \langle \nabla f(x), y - x \rangle + (y - x)^\top \nabla^2 f(x)(y - x)$ (Section ??).
- Stochastic Approximation: $\sum_i f_i(x) \approx f_j(x)$ for a random j (Section ??).
- Matrix Approximation: Approximate A by a simpler B with $\frac{1}{2}A \preceq B \preceq 2A$ (Section ?? and Section ??).
- Set Approximation: Approximate a convex set by an ellipsoid or a polytope (Section ??).
- Barrier Approximation: Approximate a convex set by a smooth function that blows up on the boundary (Section ??).
- Polynomial Approximation: Approximate a function by a polynomial (Section ??).
- Partial Approximation: Split the problem into two parts and approximate only one part.
- Taylor Approximation: $f(y) \approx \sum_{k=0}^{K} D^k f(x)[y - x]^k$.

Here are other approximation not covered:

- Mixed ℓ^2-ℓ^p Approximation: $f(y) \approx f(x) + \langle \nabla f(x), y - x \rangle + \sum_{i=1}^{n} \alpha_i (y_i - x_i)^2 + \beta_i (y_i - x_i)^p$
- Stochastic Matrix Approximation: Approximate A by a simpler random B with $B \preceq 2A$ and $B \succeq \frac{1}{2}A$
- Homotopy Method: Approximate a function by a family of functions
- ...(Please give me more examples here)...

The second question to answer is how to maintain different approximations created in different steps. One simple way would be forget the approximation we got in previous steps, but this is often not optimal. Another way is to keep all previous approximations/information (such as Section ??). Often the best way will be combining previous and current approximation carefully to a better approximation (such as Section ??).

0.2 Basic Algorithm

Perhaps the most natural algorithm for optimization is gradient descent. In fact it has many variants with different guarantees. Assume that the function f to be optimized is continuously differentiable. By basic calculus, either the minimum (or point achieving the minimum) is unbounded or the gradient is zero at a minimum. So we try to find a point with gradient close to zero (which, of course, does not guarantee global optimality). The basic algorithm is the following:

Algorithm 2: `GradientDescent` (GD)

Input: Initial point $x^{(0)} \in$, step size $h > 0$.
for $k = 0, 1, \cdots$ **do**
  if $\|\nabla f(x^{(k)})\|_2 \le \epsilon$ **then return** $x^{(k)}$;
  `// Alternatively, one can use` $x^{(k+1)} \leftarrow \texttt{argmin}_{x=x^{(k)}+t\nabla f(x^{(k)})} f(x)$.
  $x^{(k+1)} \leftarrow x^{(k)} - h \cdot \nabla f(x^{(k)})$.
end

One can view gradient descent as a greedy method for solving $\min_{x\in} f(x)$. At a point x, gradient descent goes to the minimizer of

$$\min_{\|\Delta\|_2 \le h} f(x) + \nabla f(x)^\top \Delta.$$

The term $f(x) + \nabla f(x)^\top \Delta$ is simply the first-order approximation of $f(x + \Delta)$. Note that in this problem, the current point x is fixed and we are optimizing the step Δ. Certainly, there is no inherent reason for using first-order approximation and the Euclidean norm $\|x\|_2$. For example, if you use second-order approximation, then you would get a method involving Hessian of f.

The step size of the algorithm usually either uses a fixed constant, or follows a predetermined schedule, or determined using a line search.

If the iteration stops, we get a point with $\nabla f(x)_2 \le \epsilon$. Why is this good? The hope is that x is a near-minimum in the neighborhood of x. However, this might not be true if the gradient can fluctuate wildly:

Definition 0.1. We say f has L-Lipschitz gradient if ∇f is L-Lipschitz, namely, $\|\nabla f(x) - \nabla f(y)\|_2 \le L\|x - y\|_2$ for all $x, y \in^n$.

Similar to Theorem **??**, we have the following equivalent:

Theorem 0.2. *Let $f \in \mathcal{C}^2()$. For any $L \ge 0$, the following are equivalent:*

1. $\|\nabla f(x) - \nabla f(y)\|_2 \le L\|x - y\|_2$ *for all* $x, y \in$.
2. $-LI \preceq \nabla^2 f(x) \preceq LI$ *for all* $x \in$.
3. $\left|f(y) - f(x) - \nabla f(x)^\top (y - x)\right| \le \frac{L}{2}\|y - x\|_2^2$ *for all* $x, y \in$.

Proof. Suppose (1) holds. By the definition of $\nabla^2 f$, we have

$$\nabla^2 f(x)v = \lim_{h\to 0} \frac{\nabla f(x + hv) - \nabla f(x)}{h}.$$

Since $\|\frac{\nabla f(x+hv)-\nabla f(x)}{h}\|_2 \le \frac{L}{h}\|hv\|_2 = L\|v\|_2$, we have $\|\nabla^2 f(x)v\|_2 \le L\|v\|_2$. This proves (2).

Suppose (2) holds. Since by the fundamental theorem of calculus,

$$\nabla f(x) - \nabla f(y) = \int_0^1 \nabla^2 f(y + t(x - y))(x - y)dt,$$

we have that

$$\|\nabla f(x) - \nabla f(y)\|_2 \le \int_0^1 \|\nabla^2 f(y + t(x - y))\|\|x - y\|_2 dt \le L\|x - y\|_2.$$

This gives (1).

Suppose (2) holds. By integration, we have

$$f(y) = f(x) + \nabla f(x)^\top (y - x) + \int_0^1 (1 - t)(y - x)^\top \nabla^2 f(x + t(y - x))(y - x)dt.$$

Since $-LI \preceq \nabla^2 f(x) \preceq LI$, we have

$$\left|(y - x)^\top \nabla^2 f(x + t(y - x))(y - x)\right| \le L\|y - x\|^2.$$

This gives (3).

Suppose (3) holds, then $g(x) = f(x) + \frac{L}{2}\|x\|^2$ satisfies $g(y) \geq g(x) + \nabla g(x)^\top (y - x)$ for all $x, y \in$. So Theorem ?? shows that g is convex and $\nabla^2 g(x) \succeq 0$. Hence, $\nabla^2 f(x) \succeq -LI$. Similarly, by taking $g(x) = \frac{L}{2}\|x\|^2 - f(x)$, we have $\nabla^2 f(x) \preceq LI$. This gives (2). □

With this equivalent definition, we can have an alternative view of gradient descent. Each step, we perform

$$x^{(k+1)} = \arg\min_y f(x^{(k)}) + \left\langle \nabla f(x^{(k)}), y - x^{(k)} \right\rangle + \frac{L}{2}\|y - x^{(k)}\|_2^2.$$

To see this is the same step, we let

$$g(y) = f(x^{(k)}) + \left\langle \nabla f(x^{(k)}), y - x^{(k)} \right\rangle + \frac{L}{2}\|y - x^{(k)}\|_2^2.$$

The optimality condition shows that $0 = \nabla g(x^{(k+1)}) = \nabla f(x^{(k)}) + L(x^{(k+1)} - x^{(k)})$. Hence, this gives the step $x^{(k+1)} = x^{(k)} - \frac{1}{L}\nabla f(x^{(k)})$.

By Theorem 0.2, we know that g is an upper bound of f, namely $g(y) \geq f(y)$ for all y. In general, many optimization methods involves minimizing some upper bound function every step. Note that the progress we made for f is at least the progress we made for g. If g is exactly f, we can get all the progress we can make in one step. Hence, we should believe if g is a better approximation of f, then we are making more progress. For gradient descent, it uses the simplest first-order approximation. Although this is not the best approximation one can come up with, it is robust enough to use in all sorts of applications.

0.2.1 Analysis for general functions

Gradient descent works for both convex and non-convex functions. For non-convex function, we can only find a point with small gradient (called an approximate saddle point).

Theorem 0.3. *Let f be a function with L-Lipschitz gradient and x^* be any minimizer of f. The* `GradientDescent` *with step size $h = \frac{1}{L}$ outputs a point x such that $\|\nabla f(x)\|_2 \leq \epsilon$ in $\frac{2L}{\epsilon^2}\left(f(x^{(0)}) - f(x^*)\right)$ iterations.*

The next lemma shows that the function value must decrease along the GD path for a sufficiently small step size, and the magnitude of the decrease depends on the norm of the current gradient.

Lemma 0.4. *For any $f \in \mathcal{C}^2(^n)$ with L-Lipschitz gradient, we have*

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

Proof. Lemma ?? shows that

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

for some $z \in [x, x - \frac{1}{L}\nabla f(x)]$. Since $\|\nabla^2 f(x)\| \leq L$, we have that

$$\nabla f(x)^\top \nabla^2 f(z) \nabla f(x) \leq L \cdot \|\nabla f(x)\|_2^2.$$

Putting this back above, we have

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

□

We can now prove the theorem.

Proof of Theorem 0.3. We observe that either $\|\nabla f(x)\|_2 \leq \epsilon$, or $\|\nabla f(x)\|_2 \geq \epsilon$ and so by Lemma 0.4, the function value $f(x)$ decreases by at least $\frac{\epsilon^2}{2L}$. Since the function value can decrease by at most $f(x^{(0)}) - f(x^*)$, this bounds the number of iterations — each step of gradient descent decreases f by at least $\frac{\epsilon^2}{2L}$ and since we can decrease f by at most $f(x^{(0)}) - f^*$, we have the result. □

Despite the simplicity of the algorithm and the proof, it is known that this is the best one can do via any algorithm in this general setting [?].

0.3 Analysis for convex functions

Assuming the function is convex, we can prove that gradient descent in fact converges to the global minimum. In particular, when $\|\nabla f(x)\|_2$ is small, convexity shows that $f(x) - f^*$ is small (Theorem ??).

Lemma 0.5. *For any convex $f \in \mathcal{C}^1()$, we have that $f(x) - f(y) \leq \|\nabla f(x)\|_2 \cdot \|x - y\|_2$ for all x, y.*

Proof. Theorem ?? and Cauchy-Schwarz inequality shows that

$$f(x) - f(y) \leq \langle \nabla f(x), x - y \rangle \leq \|\nabla f(x)\|_2 \cdot \|x - y\|_2.$$

□

This in turns give a better bound on the number of iterations because the bound in Theorem 0.3 is affected by $f(x) - f^*$.

Theorem 0.6. *Let $f \in \mathcal{C}^2(^n)$ be convex with L-Lipschitz gradient and x^* be any minimizer of f. With step size $h = \frac{1}{L}$, the sequence $x^{(k)}$ in* `GradientDescent` *satisfies*

$$f(x^{(k)}) - f(x^*) \leq \frac{2LR^2}{k+4} \text{ where } R = \max_{f(x) \leq f(x^{(0)})} \|x - x^*\|_2.$$

Proof. Let $\epsilon_k = f(x^{(k)}) - f(x^*)$. Lemma 0.4 shows that

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

Subtracting $f(x^*)$ from both sides, we have $\epsilon_{k+1} \leq \epsilon_k - \frac{1}{2L}\|\nabla f(x^{(k)})\|_2^2$. Lemma 0.5 shows that

$$\epsilon_k \leq \|\nabla f(x^{(k)})\|_2 \cdot \|x^{(k)} - x^*\|_2 \leq \|\nabla f(x^{(k)})\|_2 \cdot R.$$

Therefore, we have that

$$\epsilon_{k+1} \leq \epsilon_k - \frac{1}{2L}\left(\frac{\epsilon_k}{R}\right)^2.$$

Now, we need to solve the recursion. We note that

$$\frac{1}{\epsilon_{k+1}} - \frac{1}{\epsilon_k} = \frac{\epsilon_k - \epsilon_{k+1}}{\epsilon_k \epsilon_{k+1}} \geq \frac{\epsilon_k - \epsilon_{k+1}}{\epsilon_k^2} \geq \frac{1}{2LR^2}.$$

Also, we have that

$$\epsilon_0 = f(x^{(0)}) - f^* \leq \nabla f(x^*)^\top (x^{(0)} - x^*) + \frac{L}{2}\|x^{(0)} - x^*\|_2^2 \leq \frac{LR^2}{2}.$$

Therefore, after k iterations, we have

$$\frac{1}{\epsilon_k} \geq \frac{1}{\epsilon_0} + \frac{k}{2LR^2} \geq \frac{2}{LR^2} + \frac{k}{2LR^2} = \frac{k+4}{2LR^2}.$$

□

This style of proof is typical in optimization. It shows that when the gradient is large, then we make large progress and when the gradient is small, we are close to optimal.

This proof above does not use any property of ℓ_2 or inner product space. Therefore, it works for general norms if the gradient descent step is defined using that norm. For the case of ℓ_2, one can prove that $\|x^{(k)} - x^*\|_2$ is in fact decreasing.

Lemma 0.7. *For $h \leq \frac{2}{L}$, we have that $\|x^{(k+1)} - x^*\|_2 \leq \|x^{(k)} - x^*\|_2$. Therefore, for an L-gradient Lipschitz convex function f, for GD with $h = \frac{1}{L}$, we have*

$$f(x^{(k)}) - f(x^*) \leq \frac{2L\|x^{(0)} - x^*\|_2^2}{k+4}.$$

Proof. We compute the distance to an optimal point as follows, noting that $\nabla f(x^*) = 0$:

$$\begin{aligned}\|x^{(k+1)} - x^*\|_2^2 &= \|x^{(k)} - x^* - h\nabla f(x^{(k)})\|_2^2 \\ &= \|x^{(k)} - x^*\|_2^2 - 2h\left\langle \nabla f(x^{(k)}), x^{(k)} - x^* \right\rangle + h^2\|\nabla f(x^{(k)})\|^2 \\ &= \|x^{(k)} - x^*\|_2^2 - 2h\left\langle \nabla f(x^{(k)}) - \nabla f(x^*), x^{(k)} - x^* \right\rangle + h^2\|\nabla f(x^{(k)}) - \nabla f(x^*)\|^2.\end{aligned}$$

To handle the term $\nabla f(x) - \nabla f(x^*)$, we note that

$$\nabla f(x^{(k)}) - \nabla f(x^*) = H(x^{(k)} - x^*)$$

with $H = \int_0^1 \nabla^2 f(x^* + t(x^{(k)} - x^*))dt$. Since $0 \preceq H \preceq LI$ and that $H \succeq \frac{1}{L}H^2$, we have

$$\begin{aligned}\left\langle \nabla f(x^{(k)}) - \nabla f(x^*), x^{(k)} - x^* \right\rangle &= (x^{(k)} - x^*)^\top H(x^{(k)} - x^*) \\ &\geq \frac{1}{L}(x^{(k)} - x^*)^\top H^2(x^{(k)} - x^*) \\ &= \frac{1}{L}\|\nabla f(x^{(k)}) - \nabla f(x^*)\|^2.\end{aligned}$$

Hence, we have

$$\begin{aligned}\|x^{(k+1)} - x^*\|_2^2 &\leq \|x^{(k)} - x^*\|_2^2 - (\frac{2h}{L} - h^2)\|\nabla f(x^{(k)}) - \nabla f(x^*)\|^2 \\ &\leq \|x^{(k)} - x^*\|_2^2.\end{aligned}$$

The error estimate follows from $\|x^{(k)} - x^*\|_2^2 \leq \|x^{(0)} - x^*\|_2^2$ for all k and the proof in Theorem 0.6. □

Rewriting the bound, Theorem 0.6 shows it takes $\frac{2L\|x^{(0)} - x^*\|_2^2}{\epsilon}$ iterations. Compare to the bound $\frac{2L}{\epsilon^2}\left(f(x^{(0)}) - f^*\right)$ in Theorem 0.3, it seems the new result has a strictly better dependence on ϵ. However, this is not true because one measures the error in terms of $\|\nabla f(x)\|_2$ while the other is in terms of $f(x) - f^*$. For $f(x) = x^2/2$, we have $f(x) - f^* = \|\nabla f(x)\|_2^2$ and hence both have the same dependence on ϵ for this particular function. So, the real benefit of Theorem 0.6 is its global convergence.

■ 0.4 Strongly Convex Functions

We note that the convergence rate ϵ^{-1} or ϵ^{-2} is not great if we need to solve the problem up to machine accuracy. Getting to machine accuracy is sometimes important if the optimization problem is used as a subroutine. We note that for the case $f(x) = \frac{1}{2}\|x\|_2^2$, gradient descent with step size $h = 1$ takes exactly 1 step. Therefore, it is natural to ask if one can improve the bound for functions close to quadratics. This motivates the following assumption:

Definition 0.8. We call a function $f \in \mathcal{C}^1()$ is μ-strongly convex if for any $x, y \in$

$$f(y) \geq f(x) + \nabla f(x)^\top(y - x) + \frac{\mu}{2}\|y - x\|_2^2.$$

Similar to the convex case (Theorem **??**), we have the following:

Theorem 0.9. *Let $f \in \mathcal{C}^2()$. For any $\mu \geq 0$, the following are equivalent:*

- $f(tx + (1-t)y) \leq tf(x) + (1-t)f(y) - \frac{1}{2}\mu t(1-t)\|x - y\|_2^2$ *for all $x, y \in$ and $t \in [t0, 1]$.*
- $f(y) \geq f(x) + \nabla f(x)^\top(y - x) + \frac{\mu}{2}\|y - x\|_2^2$ *for all $x, y \in$.*
- $\nabla^2 f(x) \succeq \mu I$ *for all $x \in$*

Proof. It follows from applying Theorem **??** on the function $g(x) = f(x) - \frac{\mu}{2}\|x\|^2$. □

Next, we study gradient descent for μ-strongly convex functions.

Lemma 0.10. *Let $f \in \mathcal{C}^2(^n)$ be μ-strongly convex. Then, for any $x, y \in^n$,we have*

$$\nabla f(x)^2 \geq 2\mu \left(f(x) - f(y)\right).$$

Proof. By the definition of μ strong convexity, we have

$$f(y) \geq f(x) + \langle \nabla f(x), y - x \rangle + \frac{\mu}{2} x - y^2.$$

Rearranging, we have

$$f(x) - f(y) \leq \langle \nabla f(x), x - y \rangle - \frac{\mu}{2} x - y^2 \leq \max_{\Delta} \nabla f(x)^\top \Delta - \frac{\mu}{2} \Delta^2 = \frac{1}{2\mu} \nabla f(x)^2.$$

□

This will lead to the following guarantee. Note that the error now decreases geometrically rather than additively.

Theorem 0.11. *Let $f \in \mathcal{C}^2(^n)$ be μ-strongly convex with L-Lipschitz gradient and x^* be any minimizer of f. With step size $h = \frac{1}{L}$, the sequence $x^{(k)}$ in* `GradientDescent` *satisfies*

$$f(x^{(k)}) - f(x^*) \leq \left(1 - \frac{\mu}{L}\right)^k \left(f(x^{(0)}) - f(x^*)\right).$$

In a later chapter, we will see that an *accelerated* variant of gradient descent improves this further by replacing the $\frac{\mu}{L}$ term with $\sqrt{\frac{\mu}{L}}$.

Proof. Lemma 0.4 shows that

$$\begin{aligned} f(x^{(k+1)}) - f^* &\leq f(x^{(k)}) - f^* - \frac{1}{2L} \|\nabla f(x^{(k)})\|_2^2 && (1) \\ &\leq f(x^{(k)}) - f^* - \frac{\mu}{L} \left(f(x^{(k)}) - f^*\right) && (2) \\ &= \left(1 - \frac{\mu}{L}\right) \left(f(x^{(k)}) - f^*\right) && (3) \end{aligned}$$

where we used Lemma 0.10. The conclusion follows. □

It is natural to ask to what extent the assumption of convexity is essential for the bounds we obtained. This is the motivation for the next exercises.

Exercise 0.12. Suppose f satisfies $\langle \nabla f(x), x - x^* \rangle \geq \alpha \left(f(x) - f(x^*)\right)$. Derive a bound similar to Theorem 0.6 for gradient descent.

Exercise 0.13. Suppose f satisfies $\nabla f(x)_2^2 \geq \mu \left(f(x) - f(x^*)\right)$. Derive a bound similar to Theorem 0.11 for gradient descent.

Exercise 0.14. Give examples of nonconvex functions satisfying the above conditions. (Note: convex functions satisfy the first with $\alpha = 1$ and μ-strongly convex functions satisfy the second.)

■ 0.5 Line Search

In practice, we often stop line search early because we can make larger progress in a new direction. A standard stopping condition is called the Wolfe condition.

Definition 0.15. A step size h satisfies the Wolfe condition with respect to direction p if

1. $f(x + hp) \leq f(x) + c_1 h \cdot p^\top \nabla f(x)$,

2. $-p^\top \nabla f(x+hp) \le -c_2 p^\top \nabla f(x)$.

with $0 < c_1 < c_2 < 1$.

Let the objective for progress be $\phi(h) = f(x) - f(x+hp)$. The first condition requires the algorithm makes sufficient progress ($\phi(h) \ge c_1 h \cdot \phi'(0)$). The second condition requires the slope to reduce significantly ($\phi'(h) \le c_2 \phi'(0)$). One can think the first condition gives a upper bound on h while the second condition gives a lower bound on h. In general, a step size satisfying the Wolfe conditions will be larger than the step size $h = \frac{1}{L}$. In particular, one can show the following.

Exercise 0.16. Suppose f is μ-strongly convex and has L-Lipschitz gradient. If a step size h satisfies the Wolfe condition with the direction $p = -\nabla f(x)$, then we have

$$\frac{2(1-c_1)}{\mu} \ge h \ge \frac{1-c_2}{L}.$$

As a corollary, we have that the function value progress given by such step is $\Omega(\|\nabla f(x)\|^2/L)$. Therefore, this gives the same guarantee Theorem 0.6 and Theorem 0.11. A common way to implement this is via a backtracking line search. The algorithm starts with a large step size and decreases it by a constant factor if the Wolfe conditions are violated. For gradient descent, the next step involves exactly computing $\nabla f(x+hp)$ and hence if our line search accepts the step size immediately, the line search almost costs nothing extra. Therefore, if we maintain a step size throughout the algorithm and decreases it only when it violates the condition, the total cost of the line search will be only an additive logarithmic number of gradient calls throughout the algorithm and is negligible.

Finally, we note that for problems of the form $f(x) = \sum_i f_i(a_i^\top x)$, the bottleneck is often in computing Ax. In this case, exact line search is almost free because we can store the vectors Ax and Ah.

0.6 Generalizing Gradient Descent*

Now, we study what properties of gradient descent are being used for the strongly convex case. There are many ways to generalize it. One way is to view gradient descent is as approximating the function f by splitting it into two terms, one term is the first-order approximation and the second term is just an ℓ_2 norm. More generally, we can split a function into two terms, one that is easy to optimize and another that we need to approximate with some error. More precisely, we consider the following

Definition 0.17. We say $g + h$ is an α-approximation to f at x if

- g is convex, $g(x) = f(x)$,
- $h(x) = 0$ and $h((1-\alpha)x + \alpha\widehat{x}) \le \alpha^2 h(\widehat{x})$ for all $\widehat{x}$,
- $g(y) + \alpha h(y) \le f(y) \le g(y) + h(y)$ for all y.

To understand this assumption, we note that if f is μ-strongly convex with L-Lipschitz gradient, then for any x, we can use $\alpha = \frac{\mu}{L}$ and

$$g(y) = f(x) + \langle \nabla f(x), y - x \rangle,$$
$$h(y) = \frac{L}{2}\|y - x\|^2.$$

The condition requires h converging to 0 quadratically when $y \to x$.

Now, we consider the following algorithm:

Algorithm 3: `GeneralizedGradientDescent`

Input: Initial point $x^{(0)} \in$, approximation factor $\alpha > 0$.
for $k = 0, 1, \cdots$ **do**
  Find an α-approximation of f at $x^{(k)}$ given by $g^{(k)}(x) + h^{(k)}(x)$.
  $x^{(k+1)} \leftarrow \arg\min_y g^{(k)}(y) + h^{(k)}(y)$.
end

Theorem 0.18. *Suppose we are given a convex function f such that we can find an α-approximation at any x. Let x^* be any minimizer of f. Then the sequence $x^{(k)}$ in* `GeneralizedGradientDescent` *satisfies*

$$f(x^{(k)}) - f(x^*) \le (1-\alpha)^k (f(x^{(0)}) - f(x^*)).$$

Proof. Using the fact that $g^{(k)} + h^{(k)}$ is an upper bound on f, we have that our progress on f is larger than the best possible progress on $g^{(k)} + h^{(k)}$:

$$f(x^{(k+1)}) \le \min_y g^{(k)}(y) + h^{(k)}(y).$$

To bound the best possible progress, i.e., the RHS above, we consider $\widehat{x} = \arg\min_y g^{(k)}(y) + \alpha h^{(k)}(y)$ and $z = (1-\alpha)x^{(k)} + \alpha\widehat{x}$. We have that

$$\begin{aligned}
\min_y g^{(k)}(y) + h^{(k)}(y) &\le g^{(k)}(z) + h^{(k)}(z) \\
&\le (1-\alpha) g^{(k)}(x^{(k)}) + \alpha g^{(k)}(\widehat{x}) + \alpha^2 h^{(k)}(\widehat{x}) \\
&\le (1-\alpha) g^{(k)}(x^{(k)}) + \alpha (g^{(k)}(x^*) + \alpha h^{(k)}(x^*))
\end{aligned}$$

where we used $g^{(k)}$ is convex and the assumption on h in the second inequality, we used $\widehat{x}$ minimizes $g^{(k)} + \alpha h^{(k)}$.

Combining both and using the fact that $g^{(k)} + \alpha h^{(k)}$ is a lower bound on f, we have

$$f(x^{(k+1)}) \le (1-\alpha) f(x^{(k)}) + \alpha f(x^*).$$

This gives the result. □

Although Theorem 0.18 looks weird, it captures many well-known theorems. Here we list some of them.

Exercise 0.19. Show that the second condition in the definition of α-approximation can be replaced by

$$g(y) + \alpha h(y/\alpha) \le f(y) \le g(y) + h(y)$$

while maintaining the guarantee for the convergence of generalized GD.

Projected Gradient Descent / Proximal Gradient Descent

Given a convex set K, a μ-strongly convex function f with L-Lipschitz gradient, we consider the problem

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

To apply Theorem 0.18 to $F(x)f(x) + \delta_K(x)$, for any x, we consider the functions

$$\begin{aligned}
g(y) &= f(x) + \langle \nabla f(x), y - x \rangle + \delta_K(y), \\
h(y) &= \frac{L}{2}\|y - x\|_2^2.
\end{aligned}$$

Note that $g+h$ is a $\frac{\mu}{L}$-approximation to F. Theorem 0.18 shows the `GeneralizedGradientDescent` converges in $O(\frac{L}{\mu}\log(1/\epsilon))$ steps where each step involves solving the problem

$$\min_{y \in K} f(x) + \langle \nabla f(x), y - x \rangle + \frac{L}{2}\|y - x\|_2^2.$$

More generally, this works for problems of the form

$$\min_x f(x) + \phi(x)$$

for some convex function $\phi(x)$. Theorem 0.18 requires us to solve a sub-problem of the form

$$\min_y f(x) + \langle \nabla f(x), y - x \rangle + \frac{L}{2}\|y - x\|_2^2 + \phi(y).$$

ℓ^p Regression

One can apply this framework for optimizing ℓ^p regression $\min_x \|Ax-b\|_p^p$. For example, one can approximate the function $f(x) = x^p$ by $g(y) = x^p + px^{p-1}(y-x)$ and $h(y) = p2^{p-1}(x^{p-2}(y-x)^2 + (y-x)^p)$. Using this, one can show that one can solve the problem

$$\min_x \|Ax-b\|_p^p$$

using the problem

$$\min_y v^\top y + \|DA(y-x)\|_2^2 + \|A(y-x)\|_p^p \tag{4}$$

for some vector v and some diagonal matrix D. One can show that Theorem 0.18 only need to solve the sub-problem approximately. Therefore, this shows that one can solve ℓ^p regression with $\log(1/\epsilon)$ convergence by solving mixed $\ell_2 + \ell_p$ regression approximately.

Other assumptions

For some special cases, it is possible to analyze this algorithm without convexity. One prominent application is compressive sensing:

$$\min_{\|x\|_0 \le k} \|Ax-b\|_2^2.$$

For matrices A satisfying restricted isometry property, one can apply `GeneralizedGradientDescent` to solve the problem with the splitting $g(x) = 2A^\top(Ax-b) + \delta_{\|x\|_0 \le k}$ and $h(x) = \|x\|^2$. In this case, the algorithm is called iterative hard-thresholding [?] and the sub-problem has a closed form expression.

Exercise 0.20. Give the closed form solution for the sub-problem given by the splitting above.

■ 0.7 Gradient Flow

In continuous time, gradient descent follows the ODE

$$\frac{dx_t}{dt} = -\nabla f(x_t).$$

This can be viewed as the canonical continuous algorithm. Finding the right discretization has lead to many fruitful research directions. One benefit of the continuous view is to simplify some calculations. For example, for strongly convex f, Theorem 0.11 now becomes

$$\frac{d}{dt}(f(x_t) - f(x^*)) = \nabla f(x_t)^\top \frac{dx_t}{dt} = -\|\nabla f(x_t)\|_2^2 \le -2\mu\,(f(x_t) - f(x^*))$$

where we used Lemma 0.10 at the end. Solving this differential inequality, we have

$$f(x_t) - f(x^*) \le e^{-2\mu t}(f(x_0) - f(x^*)).$$

Without the strong convexity assumption, gradient flow can behave wildly. For example, the length of the gradient flow can be exponential in d on an unit ball [?].

We emphasize that this continuous view is mainly useful for understanding, indicative of but not necessarily implying an algorithmic result.

■ 0.8 Discussion

Convex optimization by variants of gradient descent is a very active field with an increasing number of applications. Often, methods that are provable for the convex setting are applied as heuristics to nonconvex problems as well, most notably in deep learning. This is one of the principal features of GD, its wide applicability as an algorithmic paradigm.

Researchers are also using GD to get provably faster algorithms for classical problems. For example, [**?**] and [**?**] applied the decomposition of Eqn. (4) to obtain fast algorithms for ℓ^p regression and the ℓ^p flow problem. [**?**] showed that the ℓ^p flow problem can be used as a subroutine to solve uncapacitied maximum flow problem in $m^{4/3+o(1)}$ time. Instead of assuming $h(x)$ converges to 0 quadratically, [**?**] proved Theorem 0.18 assuming h is given by some divergence function and showed its applications in D-optimal design.