10-425/625: Introduction to Convex Optimization **(Fall 2023)**

Lecture 19: Proximal Gradient Method

Instructor:[1] *Matt Gormley* *November 13, 2023*

19.1 Proximal Gradient Descent

The particular problem class we'll focus on now will be: we want to minimize an unconstrained function f which can be written as the sum of a "nice", convex, function g and a potentially non-smooth convex function h, i.e.

$$\min_{x \in \mathbb{R}^d} g(x) + h(x).$$

The prox. GD algorithm alternates the following steps:

1. We compute $y^{t+1} = x^t - \eta_t \nabla g(x^t)$.
2. We then compute our next iterate by solving:

$$x^{t+1} = \arg\min_{x \in \mathbb{R}^d} \left[h(x) + \frac{1}{2\eta_t} \|x - y^{t+1}\|_2^2 \right].$$

(Note that at this point, the only other solution we have for such a problem is the subgradient method, with $O(1/\sqrt{k})$ convergence rate. However, we'll see this proximal gradient descent method can achieve a $O(1/k)$ rate—with the usual assumptions. Relatedly, recall that the projected gradient algorithm on problems of the form $\min_{x \in \mathcal{C}} f(x)$ achieved rates of convergence that were inhereited from the unconstrained optimization of f.)

19.1.1 The Proximal Operator

For a convex function f the proximal operator is defined to be:

$$\text{prox}_f(v) = \arg\min_x \left(f(x) + \frac{1}{2} \|x - v\|_2^2 \right).$$

[1]These notes were originally written by Siva Balakrishnan for 10-725 Spring 2023 (original version: here) and were edited and adapted for 10-425/625.

One can interpret this as some type of generalized projection operation, i.e. if you take a point v outside the domain of f the prox operator will return a point which is in the domain close to the original point (where the function f is small). If you are inside the domain then prox operator will move you towards the minimum of f, and if you're at the minimizer of f you will stay there.

More directly, if $f(x) = \mathbb{I}_C(x)$ is a convex indicator function, then $\text{prox}_f(v)$ simply projects v onto C.

With this notation, one can write the prox GD algorithm as:

1. We compute $y^{t+1} = x^t - \eta_t \nabla g(x^t)$.
2. We then compute our next iterate by solving:

$$x^{t+1} = \text{prox}_{\eta_t h}(y^{t+1}).$$

19.1.2 Step Sizes

As with any of the first-order algorithms we have studied so far an important practical choice will be that of the step-size. We'll as usual see some theoretical choices in the future. In practice, often backtracking is used. Usually it is used to ensure sufficient decrease in the smooth part of the objective. Describing the precise backtracking line search will be a bit easier once we've introduced the so-called gradient map so we'll return to it later.

19.1.3 An Example – ISTA

There are many nice examples of proximal algorithms that are popular in practice. Part of the revival of interest in prox GD came from its application to solving the LASSO – in this case the algorithm goes by the name Iterative Shrinkage Thresholding Algorithm (ISTA).

If we want to use the prox GD algorithm to solve the LASSO:

$$\min_{x \in \mathbb{R}^d} \frac{1}{2}\|Ax - b\|_2^2 + \lambda \|x\|_1,$$

we would:

1. Compute $y^{t+1} = x^t - \eta_t A^T(Ax^t - b)$.

2. Then we would need to solve the prox problem:

$$x^{t+1} = \arg\min_{x \in \mathbb{R}^d} \frac{1}{2\eta_t} \|x - y^{t+1}\|_2^2 + \lambda \|x\|_1.$$

   Fortunately, we seen the solution to this problem in closed form. We set

$$x^{t+1} = S_{\lambda\eta_t}(y^{t+1}),$$

   where S_λ is the soft-thresholding operation we have seen before.

Definition 19.1. *The soft-thresholding operation for a scalar y_i, and $\lambda > 0$,*

$$S_\lambda(y_i) = \begin{cases} y_i - \lambda & \textit{if } y_i > \lambda \\ 0 & \textit{if } -\lambda \leq y_i \leq \lambda \\ y_i + \lambda & \textit{if } y_i < -\lambda. \end{cases}$$

This algorithm is very natural – we simply take a gradient step (ignoring the non-smooth part) and then soft-threshold the iterates (to account for the non-smooth part). It should however be pretty mysterious as to why this algorithm works (is the LASSO solution even a fixed point?).

19.1.4 Motivating Prox GD

There are many different ways to motivate the proximal gradient descent algorithm:

1. **As a generalization of projected GD:** In projected GD, we wanted to solve the constrained problem:

$$\min_{x \in C} f(x),$$

   where f was a differentiable function which could be written as:

$$\min_{x \in \mathbb{R}^d} f(x) + \mathbb{I}_C(x),$$

   which is exactly of the form "nice function" + non-smooth function. Minimizing this with the prox GD algorithm will simply recover the projected GD algorithm (because the prox operation is simply a projection). So one can view prox GD as a natural extension of projected GD to other types of smooth, non-smooth sums.

2. **The Local Approximation Viewpoint:** Another viewpoint on the proximal GD algorithm is that in order to minimize a sum $f = g + h$ one can locally approximate just the smooth part and carry out the minimization, i.e. we can attempt to minimize:

$$x^{t+1} = \arg\min_{x \in \mathbb{R}^d} \left[g(x^t) + \nabla g(x^t)^T (x - x^t) + \frac{1}{2\eta_t} \|x - x^t\|_2^2 + h(x) \right].$$

   In the above expression we have approximated the smooth part (by a linear approximation) and left the non-smooth part untouched.

   Now, it is easy to see that this is equivalent to writing:

$$\begin{aligned} x^{t+1} &= \arg\min_{x \in \mathbb{R}^d} \left[\frac{1}{2\eta_t} \|x - (x^t - \eta_t \nabla g(x^t))\|_2^2 + h(x) \right] \\ &= \text{prox}_{\eta h}(x^t - \eta_t \nabla g(x^t)). \end{aligned}$$

   This is precisely the prox GD algorithm.

3. **Connection to Expectation-Maximization (EM)/Majorization-Minimization (MM) algorithms:** If we have a β-smooth function g, and we choose $\eta_t \leq 1/\beta$ then we know that,

$$g(x) \leq g(x^t) + \nabla g(x^t)^T (x - x^t) + \frac{1}{2\eta_t} \|x - x^t\|_2^2.$$

   As a consequence, our prox GD algorithm in this case is simply finding a "nice" convex upper bound on our function, which is tight at $x = x^t$, and minimizing this upper bound. Algorithms that do this are called MM algorithms, and have lots of nice properties. The EM algorithm is a special case.

19.1.5 Some Special Cases

In the prox algorithm:

1. When $h = 0$ this is just regular GD.
2. When $h = \mathbb{I}_C$ this is just projected GD (which we have some understanding of).

3. When $g = 0$ this is a different algorithm (that pre-dates prox GD) known as proximal minimization, where to minimize some function f, we regularize it a bit and minimize it iteratively, i.e. we solve

   $$x^{t+1} = \text{prox}_{\eta f}(x^t).$$

   This iteration will converge to a minimizer of f under weak conditions (and is sometimes easier to solve than the original problem because of the regularization).

Now that we've derived the algorithm and understood some simple special cases, it remains to really make sense of it.

Our first goal for today will be to try to better understand the prox. algorithm we defined in the previous lecture. Then we will turn our attention to another popular example of proximal GD.

19.2 Some Properties of the Proximal Operator

Recall, for a function h,

$$\text{prox}_h(x) = \arg\min_u \frac{1}{2}\|x - u\|_2^2 + h(u).$$

19.2.1 The proximal operator is a contraction

The first fact that we will show is that (like a projection) the prox operation is a contraction, i.e.

Lemma 19.2. *For a convex function h,*

$$\|prox_h(x) - prox_h(y)\|_2 \leq \|x - y\|_2.$$

Proof: We'll prove the stronger claim that,

$$\|\text{prox}_h(x) - \text{prox}_h(y)\|_2^2 \leq \langle x - y, \text{prox}_h(x) - \text{prox}_h(y)\rangle,$$

from which the original claim follows (by applying the Cauchy-Schwarz inequality to the RHS).

To see this we simply use subgradient optimality conditions. Let $u = \text{prox}_h(x)$, $v = \text{prox}_h(y)$, then we know that by subgradient optimality conditions,

$$\begin{aligned} 0 \in u - x + \partial h(u) \Rightarrow x - u \in \partial h(u), \\ 0 \in v - y + \partial h(v) \Rightarrow y - v \in \partial h(v). \end{aligned}$$

By monotonicity of the gradient of h we know that for any elements (a, b) of the subdifferential of h at u, v,

$$\langle a - b, u - v \rangle \geq 0.$$

Applying this to the two vectors above we see that,

$$\langle x - u - y + v, u - v \rangle \geq 0,$$

from which the stronger claim (and hence the lemma) follows. ■

19.2.2 Gradient Mapping

Another important property of the prox. update is that it only has one fixed point, i.e. only if we're already at an optimal point will the prox. algorithm stop. Before we can show this, we'll need to define an important quantity which is the *gradient mapping.* We'd like to analyze prox. GD but it differs from GD in that it's a bit difficult to make sense of the prox. GD updates. Define,

$$G_\eta(x) = \frac{1}{\eta}\left[x - \text{prox}_{\eta h}(x - \eta \nabla g(x))\right].$$

This might seem quite unintuitive a quantity (it is), but it allows us to write,

$$x^{t+1} = x^t - \eta_t G_{\eta_t}(x^t).$$

This in turn makes prox. GD updates look a bit more familiar. Provided that we can get some handle on the gradient mapping we might then be able to analyze prox. GD in a similar fashion to how we analyzed GD.

Now, lets use the gradient mapping to argue that the only prox. GD fixed points are optimal solutions to our original program.

Lemma 19.3.

$$G_\eta(x^*) = 0 \iff 0 \in \nabla g(x^*) + \partial h(x^*)$$

Proof: For any point $\widetilde{x}$, we know that,

$$G_\eta(\widetilde{x}) = \frac{1}{\eta}\left[\widetilde{x} - \mathrm{prox}_{\eta h}(\widetilde{x} - \eta \nabla g(\widetilde{x}))\right],$$

and this in turn (by optimality conditions for prox) means that,

$$\widetilde{x} - \eta \nabla g(\widetilde{x}) - \widetilde{x} + \eta G_\eta(\widetilde{x}) \in \eta \partial h(\widetilde{x} - \eta G_\eta(\widetilde{x})),$$

i.e. for any point $\widetilde{x}$ we must have that,

$$G_\eta(\widetilde{x}) \in \nabla g(\widetilde{x}) + \partial h(\widetilde{x} - \eta G_\eta(\widetilde{x})). \tag{19.1}$$

It is worth thinking about this equation for a bit and noticing how it differs from a usual sub-gradient, i.e. $v_{\widetilde{x}}$ would be a valid subgradient if it was in the collection $\nabla g(\widetilde{x}) + \partial h(\widetilde{x})$, but the proximal gradient mapping satisfies a slightly different condition.

Now we see that if $G_\eta(x^*) = 0$ for some x^* then x^* must satisfy,

$$0 \in \nabla g(x^*) + \partial h(x^*).$$

Conversely, if $0 \in \nabla g(x^*) + \partial h(x^*)$, then we have that,

$$x^* - \eta \nabla g(x^*) - x^* \in \eta \partial h(x^*),$$

which in turn means that,

$$x^* = \mathrm{prox}_{\eta h}(x^* - \eta \nabla g(x^*)).$$

This final expression yields that, $G_\eta(x^*) = 0$ as desired. ■

19.3 Convergence Analysis

19.3.1 Main Descent Lemma

Our eventual goal will be to analyze the prox algorithm when g is β-smooth, and h is convex The main technical hurdle will be to prove a descent lemma which works when we replace gradients by generalized gradients.

Lemma 19.4. *For any $\eta \leq 1/\beta$, and any z*

$$f(x - \eta G_\eta(x)) \leq f(z) + G_\eta(x)^T(x - z) - \frac{\eta}{2}\|G_\eta(x)\|_2^2.$$

Once we prove this lemma the analysis will mirror the smooth case, but notice that this is a very non-obvious statement since f is not smooth, and since G_η is not a gradient map (or even a valid subgradient).

It is a generalization of the descent lemma. If we plug in $z = x$, then we recover the more familiar expression:

$$f(x - \eta G_\eta(x)) \leq f(x) - \frac{\eta}{2}\|G_\eta(x)\|_2^2.$$

This justifies referring to this algorithm as a descent algorithm (at least for appropriate step-size choices the function value reduces in each iteration).

Proof: We begin by observing,

$$\begin{aligned} f(x - \eta G_\eta(x)) &= g(x - \eta G_\eta(x)) + h(x - \eta G_\eta(x)) \\ &\leq g(x) - \eta \nabla g(x)^T G_\eta(x) + \frac{\eta^2 \beta}{2}\|G_\eta(x)\|_2^2 + h(x - \eta G_\eta(x)) \\ &\leq g(z) + \nabla g(x)^T(x - z) - \eta \nabla g(x)^T G_\eta(x) + \frac{\eta^2 \beta}{2}\|G_\eta(x)\|_2^2 + h(x - \eta G_\eta(x)) \end{aligned}$$

just by using smoothness on the β-smooth function h. Now, just by convexity of h we know that,

$$h(x - \eta G_\eta(x)) \leq h(z) - \theta^T(z - x + \eta G_\eta(x)),$$

where θ is any element of $\partial h(x - \eta G_\eta(x))$. In (19.1) we showed that, $G_\eta(x) - \nabla g(x) \in h(x - \eta G_\eta(x))$, so we obtain the bound,

$$h(x - \eta G_\eta(x)) \leq h(z) - \eta\|G_\eta(x)\|_2^2 + \eta \nabla g(x)^T G_\eta(x) - (G_\eta(x) - \nabla g(x))^T(z - x).$$

Putting the pieces together we obtain the desired claim. ■

19.3.2 Analyzing the Prox. Algorithm

With the descent lemma in place its straightforward to analyze prox. GD when applied to β-smooth g (and convex h).

Theorem 19.5. *For β-smooth g, convex h the prox. GD algorithm with step-size $\eta = 1/\beta$ achieves the following guarantee:*

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

Proof: As usual we begin with the familiar manipulations,

$$\|x^{t+1} - x^*\|_2^2 = \|x^t - x^*\|_2^2 + 2\eta \left(\frac{\eta}{2} \|G_\eta(x^t)\|_2^2 - (x^t - x^*)^T G_\eta(x^t) \right).$$

Our lemma (applied with $z = x^*$) gives an upper bound on the last term, which in turn yields:

$$\|x^{t+1} - x^*\|_2^2 \leq \|x^t - x^*\|_2^2 + 2\eta(f(x^*) - f(x^{t+1})).$$

Re-arranging and telescoping we obtain the final result. ■

What should be surprising is that we've attained a $1/k$ rate of convergence for a class of non-smooth functions (i.e. f is certainly not smooth). As always though, we're exploiting a very particular type of structure (that arises often in regularized loss minimization problems). We also no longer operate in the first-order model (and are not bound by the oracle lower bounds discussed earlier).

19.3.3 Descent Without Smoothness

For some more intuition about why the proximal algorithm is so powerful – consider the case when $g = 0$, i.e. we're just doing proximal minimization of a potentially non-smooth function h. Since the function h is not smooth, we could apply the subgradient method, but the crucial observation we belaboured was that an arbitrary subgradient would in general not be a descent direction, and we could never hope to prove an analogue of our "main descent lemma" (which we proved for GD under smoothness).

Now, lets try to understand the proximal minimization steps (this is really a special case of the facts we proved above setting $g = 0$ but the magic is much clearer). We simply iterate, for any fixed $\eta > 0$,

$$x^{t+1} = \text{prox}_{\eta h}(x^t) := \arg\min_x \frac{1}{2}\|x - x^t\|_2^2 + \eta h(x).$$

Of course, we could take η very large, and this would converge to a minimizer in one step (but that problem is as hard to solve as the minimization of h directly). In practice, one might hope taking a smaller value of η might yield easier to solve sub-problems which still converge to a minimizer of h.

We define the gradient mapping $G_\eta(x)$ as before (though its expression is a bit simpler):

$$G_\eta(x) = \frac{1}{\eta}\left(x - \mathrm{prox}_{\eta h}(x)\right).$$

Lemma 19.6 (Descent Without Smoothness)**.** *For h convex, we have the following guarantees,*

$$\begin{aligned} h(x^{t+1}) &\leq h(x^t) - \eta\|G_\eta(x^t)\|_2^2, \\ h(x^{t+1}) &\leq h(x^*) - \eta\|G_\eta(x^t)\|_2^2 + G_\eta(x^t)^T(x^t - x^*). \end{aligned}$$

It is worth appreciating this result, since it highlights the key idea of the prox. method. Without any smoothness assumptions whatsoever we obtain a descent lemma. The key caveat being that it's no longer an "explicit" step, rather we solve a minimization for each step (taking a so-called "implicit" step).

Proof: We know that for any z, any $u \in \partial h(x^{t+1})$,

$$h(x^{t+1}) \leq h(z) - \eta u^T(z - x^{t+1}).$$

In our setting we know that, $G_\eta(x^t) \in \partial h(x^{t+1})$. Taking $z = x^t$ we recover the first claim, and taking $z = x^*$ we recover the second claim. ■

Theorem 19.7. *After k iterations the proximal method, for convex h, achieves the guarantee:*

$$h(x^k) - h(x^*) \leq \frac{\|x^0 - x^*\|_2^2}{2\eta k}.$$

Proof: As usual,

$$\begin{aligned} \|x^{t+1} - x^*\|_2^2 &= \|x^t - x^*\|_2^2 + \eta^2\|G_\eta(x^t)\|_2^2 - 2\eta G_\eta(x^t)^T(x^t - x^*) \\ &\leq \|x^t - x^*\|_2^2 + 2\eta(h(x^{t+1}) - h(x^*)), \end{aligned}$$

just by applying the second claim of the descent lemma. Re-arranging and summing we obtain that,

$$h(x^k) - h(x^*) \leq \frac{\|x^0 - x^*\|_2^2}{2\eta k}.$$

■

19.4 Another example – Matrix Completion

Another nice example of the proximal GD algorithm comes from the problem of matrix completion. In matrix completion, we observe some subset of indices of a matrix M^* (possibly with some noise), and we would like to "fill in" the matrix. For some subset of indices $\mathcal{I}$, we observe $\{Y_{ij} : (i,j) \in \mathcal{I}\}$.

Of course there are many possible ways to do this, and a typical assumption is that M^* has low rank, and so we'd like to find a matrix that is close to Y on the observed indices but has small rank. The rank of a matrix is a non-convex function, and so we'll instead choose to use a convex relaxation of the rank, which is called the trace norm or nuclear norm.

For a matrix $M \in \mathbb{R}^{n \times d}$ if we write its SVD as $M = U\Sigma V^T$ then the nuclear norm is simply $\|M\|_{\text{tr}} = \sum_{i=1}^{d} \sigma_i(M)$, i.e. the sum of its singular values.

With all of this background, we'd like to minimize the following objective:

$$\min_M \sum_{(i,j)\in\mathcal{I}} (Y_{ij} - M_{ij})^2 + \lambda \|M\|_{\text{tr}}.$$

This is a convex objective, but just like the LASSO the regularizer is not smooth.

We can still hope for a fast (at least compared to the subgradient method) algorithm if we can compute the proximal operator corresponding to the regularizer. You will show in your HW that the program,

$$\min_M \frac{1}{2}\|Y - M\|_F^2 + \lambda \|M\|_{\text{tr}},$$

has a closed form solution. The optimal M comes from soft-thresholding the singular values of Y, i.e. $\text{prox}_{\lambda h}(Y) = U\Sigma_\lambda V^T$, where $Y = U\Sigma V^T$ is the SVD of Y and $\Sigma_\lambda = \max\{0, \Sigma - \lambda\}$ which just subtracts λ from every singular value of Y (and stops if it hits 0).

Now, we have a simple proximal GD algorithm to optimize our original objective:

1. We compute $Z^{t+1} = X^t - \eta(X^t - Y) \circ \Omega$, where $\Omega_{ij} = 1$ if $(i, j) \in \mathbb{I}$ and 0 otherwise.
2. We then compute $X^{t+1} = \text{prox}_{\lambda h}(Z^{t+1})$.

You can take the step-size $\eta = 1$ since the objective is 1-smooth, and then this algorithm is known as soft-impute and is one of the faster ways to solve the matrix completion problem.