UNIVERSITY OF SOUTHERN CALIFORNIA

EE546 FINAL PROJECT

Literature Review: On Graduated Optimization for Stochastic Non-Convex Problems by Elad Hazan, Kfir Y. Levy and Shai Shalev-Swartz

DHRUV PARIKH
12-9-2020
USC ID 2522583608

1. Why Graduated Optimization?

The paper discusses graduated optimization scheme, performed stochastically, for a function that may be non-convex. This warrants discussion of the graduated optimization scheme, before novel points of the paper are discussed.

Functions that are difficult to optimize (non-convex functions for instance) are optimized in steps/epochs. In each step/epoch, we optimize over a 'smooth' version of the function. The smooth version of the function can be built in several ways, however, no matter the smoothing technique, as the epoch number increases, smoothing becomes more and more local.

Initially, we start with the smoothest version of the function and optimize over this version of the original function. The result of this optimization is then used to optimize over a 'less smooth' version of the original function, in the subsequent epoch. With an increase in the number of epochs, the function over which optimization is performed gets closer and closer to the original function, and the optimal point estimate gets closer and closer to the true global optimum.

Thus, graduated optimization breaks down the original optimization problem into several parts. Each part is solved as its own separate optimization problem, and the solution of each part hastens the solution of the subsequent part.

Smooth versions of non-convex functions may have certain properties which can be exploited to analyze convergence behavior of optimization of such functions.

2. Smoothing

Smoothing the original non-convex function is an important step to the optimization strategy presented in the paper. Smoothing a function simply means averaging the original function at each point, over a local neighborhood/vicinity surrounding that point. Of the different ways that smoothing can be performed, the approach used here is as follows:

Smoothing. Given $f : R^d \rightarrow R$, its δ smoothed version is given by

$$\hat{f}_\delta(\boldsymbol{x}) = \boldsymbol{E}[f(\boldsymbol{x} + \delta \boldsymbol{u})]$$

Where $\boldsymbol{u}$ is a random variable distributed uniformly over a ball of unit radius, centered at the origin.

$$\boldsymbol{u} \sim B(\mathbf{0}, 1)$$

The above notation will be used to denote a random variable which is distributed over a certain region, uniformly.

Thus, the parameter δ controls the size of the neighborhood over which the function is smoothed. A larger δ corresponds to a larger neighborhood surrounding $\boldsymbol{x}$, and thus corresponds to an even smoother function (estimate of the original function).

In graduated optimization, once a smooth version of the original function is constructed, we then optimize over this smooth function. The minimizer estimate (optimal solution) of this smooth version is then obtained, via an optimization strategy. In the subsequent iteration/epoch, the value of δ is reduced, the new smooth function built again, and this new smooth function is optimized via the same optimization strategy which is now initiated via the minimizer estimate (optimal solution) of the previous epoch.

3. Function Class

The non-convex function for which optimization has been performed and analyzed in the paper has two properties:

i. L-Lipschitz
ii. σ-nice

Non-convex function which are L-Lipschitz and σ-nice have convergence of poly(1/ε), as has been shown in the paper. Prior to discussing the optimization strategy, we will first discuss what it means for the function to be L-Lipschitz and σ-nice.

3.1. L-Lipschitz Functions

Given a function $f: S \to R, S \subseteq R^d$, if $\forall \boldsymbol{x}, \boldsymbol{y} \in S$,

$$|f(\boldsymbol{x}) - f(\boldsymbol{y})| \leq L\|\boldsymbol{x} - \boldsymbol{y}\|$$

Then the function is said to be L-Lipschitz. The norm is the Euclidean norm.

Additionally, if the function is differentiable ($or\ C^1$), the above condition is equivalent to,

$$\|\nabla f(\boldsymbol{x})\| \leq L$$

Also, if the gradient norm of a function is bounded (by L), then it implies that,

$$|f(\boldsymbol{x}) - f(\boldsymbol{y})| \leq L\|\boldsymbol{x} - \boldsymbol{y}\|$$

Thus,

$$|f(\boldsymbol{x}) - f(\boldsymbol{y})| \leq L\|\boldsymbol{x} - \boldsymbol{y}\| \ \leftrightarrow\ \|\nabla f(\boldsymbol{x})\| \leq L$$

Given, obviously, that the function is differentiable.

Bias between $\hat{f}_\delta(x)$ ***and*** $f(x)$**.** After discussing L-Lipschitz functions, we can now bound the bias between $f(x)$ and its smooth version, with parameter δ, $\hat{f}_\delta(x)$.

Given an L-Lipschitz function $f: R^d \to R$, we have previously defined its smooth version $\hat{f}_\delta(x)$ which is,

$$\hat{f}_\delta(\boldsymbol{x}) = \boldsymbol{E}[f(\boldsymbol{x} + \delta\boldsymbol{u})]$$

Then, $\forall\, x \in R^d$, we have the following bound on the bias:

$$\left|\hat{f}_\delta(x) - f(x)\right| \leq \delta L$$

The proof for this bound is quite straightforward and uses the Jensen's Inequality in combination with the L-Lipschitz nature of the function and the fact that $\|u\| \leq 1$.

3.2. σ-Nice Functions

An important class of non-convex functions, defined in the paper, is the class of σ-nice functions.

σ-nice. Given a function $f: \mathcal{K} \longrightarrow R$, where $\mathcal{K} \subseteq R^d$, is a convex set, the function is called σ-nice if it satisfies the following properties:

Centering property. (Property 1)

Consider δ and $\hat{f}_\delta(\boldsymbol{x})$ as defined previously. Say,

$$x_\delta^* = argmin\, \hat{f}_\delta(\boldsymbol{x})\ (over\ \mathcal{K})$$

Thus, x_δ^* is the minimizer for $\hat{f}_\delta(\boldsymbol{x})$.

Additionally, also consider $\delta/2$ and $\hat{f}_{\delta/2}(\boldsymbol{x})$. Say,

$$x_{\delta/2}^* = argmin\, \hat{f}_{\delta/2}(\boldsymbol{x})\ (over\ \mathcal{K})$$

Thus, $x_{\delta/2}^*$ is the minimizer for $\hat{f}_{\delta/2}(\boldsymbol{x})$.

The centering property states that ∀ such $\delta > 0$, ∀ such x_δ^*, there must exist ∃ $x_{\delta/2}^*$, such that the following property is satisfied:

$$\left\|x_\delta^* - x_{\delta/2}^*\right\| \leq \delta/2$$

Thus, the distance between minimizers of smoothed version of the (original) function, with smoothing parameters δ and $\delta/2$ should be upper bounded by $\delta/2$.

Local strong convexity (for the smoothed function). (Property 2)

In addition to the centering property, σ-nice functions should also satisfy the property of their smooth versions being strongly convex, locally.

Define $r_\delta = 3\delta$ and $x_\delta^* = argmin\, \hat{f}_\delta(\boldsymbol{x})\ (over\ \mathcal{K})$.

Let $B(x_\delta^*, r_\delta)$ be a ball of radius r_δ centered at x_δ^*.

This property, then, states that ∀ such $\delta > 0$,

$\hat{f}_\delta(\boldsymbol{x})$ must be σ-strongly convex over $B(x_\delta^*, r_\delta)$.

Thus, the smoothed version of the original function must be σ-strongly convex locally, around its minimizer, for all smoothing parameters greater than 0.

3.2.1 Strongly Convex Functions

Since the smoothed version of the original function, given that the function is σ-nice, must be locally strongly convex, it is a good sidenote to discuss the properties associated with strongly convex functions.

Strong convexity. A function $F: R^n \to R$ is σ-strongly convex over the convex set $\mathcal{K}$ if $\forall \boldsymbol{x}, \boldsymbol{y} \in \mathcal{K}$, the following property holds:

$$F(y) \geq F(x) + \nabla F(x)^T (y - x) + \frac{\sigma}{2}\|x - y\|^2$$

Also, ∃ x^* that minimizes $F(x)$ over the convex set $\mathcal{K}$, the following property holds for these functions (σ-strongly convex functions):

$$\frac{\sigma}{2}\|x - x^*\|^2 \leq F(x) - F(x^*)$$

This property is obvious from the definition of σ-strongly convex functions combined with the fact that at x^* the function gradient is 0 ($\nabla F(x^*) = 0$) since x^* is a global minimum and the function is convex (σ-strongly convex functions are also convex) and we assume that the function is continuously differentiable.

4. Building Oracles

As has been discussed before, once smoothing of the original function has been performed and the smoothed version $\hat{f}_\delta(\boldsymbol{x})$ built, $\exists\ \delta > 0$, we now attempt to optimize this smoothed function. As such, we could directly optimize using gradients of $\hat{f}_\delta(\boldsymbol{x})$, however this may be computationally expensive. Since δ changes in every single iteration, the function that is to be optimized changes in every single iteration, and thus we need to evaluate a new set of gradients for new functions in every single iteration. Thus optimizing $\hat{f}_\delta(\boldsymbol{x})$ directly seems computationally taxing.

In order to avoid this, we build oracles that estimate the gradients of $\hat{f}_\delta(\boldsymbol{x})$ using the original function $f(x)$ and its gradients. By ensuring that these estimates are unbiased, and using these estimates instead, we make the algorithm much more efficient (at a small cost to gradient estimation accuracy).

Two oracles have been discussed in the paper, built from the original function via random sampling, which we see thus.

4.1. Gradient Feedback Oracle (Oracle 1)

In this particular oracle, the gradient of the smooth function is estimated via the gradient of the original function, via random sampling the gradient of the original function.

Gradient feedback oracle. $\nabla\hat{f}_\delta(x)|_{estimate} = \nabla f(x + \delta\boldsymbol{u})$

Again, $\boldsymbol{u} \sim B(\mathbf{0}, 1)$

Thus, the gradient of the smooth function is estimated via the gradient of the original function, at a randomly sampled point in a neighborhood around x.

Given that the function $f: R^d \to R$, $\delta \geq 0$ and that f is L-Lipschitz, the estimator as defined above satisfies the following properties,

$$\boldsymbol{E}[\nabla f(x + \delta\boldsymbol{u})] = \nabla\hat{f}_\delta(x)$$

$$\|\nabla f(x + \delta\boldsymbol{u})\| \leq L$$

Thus, the gradient feedback oracle is an unbiased and bounded estimator for the gradient of the smooth version of the original function.

Differentiating $\hat{f}_\delta(x) = \boldsymbol{E}[f(x + \delta\boldsymbol{u})]$ directly gives the first result and the second result is due to f being L-Lipschitz.

Algorithm 1 shown in the next page algorithmizes this procedure and creates a subroutine to use in our main optimization algorithm. The oracle that uses the gradient feedback from the original function has been named SGO_G.

SGO$_G$
Input: $x \in R^d$ and δ
Output: $\nabla f(x + \delta \boldsymbol{u})$, $\boldsymbol{u} \sim B(\mathbf{0}, 1)$

Algorithm 1: Gradient Feedback Oracle

4.2. Value Feedback Oracle (Oracle 2)

Rather than using the function gradient, this particular oracle directly uses the function value to estimate the gradient of the smooth function.

Value feedback oracle. $\nabla \hat{f}_\delta(x)|_{estimate} = \frac{d}{\delta} f(x + \delta \boldsymbol{v}) \boldsymbol{v}$

Here d is the d in $x \in R^d$ and, $\boldsymbol{v}$ is a random vector defined uniformly over the surface of a ball centered at the origin and of unit radius. Thus,

$$\boldsymbol{v} \sim S$$

S is the surface of $B(\mathbf{0}, 1)$, defined previously.

Given that $x \in \mathcal{K}$, where $\mathcal{K}$ is a convex set, $\mathcal{K} \subseteq R^d$, $\delta \geq 0$, and say,

$$|f(x)| \leq C, over\ \mathcal{K}$$

The above value feedback estimator satisfies the following properties,

$$\boldsymbol{E}\left[\frac{d}{\delta} f(x + \delta \boldsymbol{v}) \boldsymbol{v}\right] = \nabla \hat{f}_\delta(x)$$

$$\left\| \frac{d}{\delta} f(x + \delta \boldsymbol{v}) \boldsymbol{v} \right\| \leq \frac{Cd}{\delta}$$

The latter property is obvious while the former is a bit more involved to prove (uses Stokes Theorem).

We thus see that even the value feedback oracle is both, unbiased and bounded, in its estimation of the gradient of the smooth function. As SGO_G was defined, we also define SGO_V subroutine.

SGO$_V$
Input: $x \in R^d$ and δ
Output: $\frac{d}{\delta} f(x + \delta \boldsymbol{v}) \boldsymbol{v}$, $\boldsymbol{v} \sim S$

Algorithm 2: Value Feedback Oracle

It can be seen that the value of f is also randomly sampled. This random sampling of the value of the function, to estimate the gradient of the smooth function, is done over a sphere (surface of a ball) centered at x with a radius of δ.

Implicit Smoothing. The above methods of building oracles are referred as methods to 'implicitly smooth' the function, in the paper. In the initial discussions, the importance of building 'smooth versions' of the original function has been repeatedly highlighted, as it relates to graduated optimization. The above oracles facilitate information extraction from the smoothed function, without having to

explicitly build it. Since the original function is never explicitly smoothed in the optimization algorithm, we say that it has been 'implicitly smoothed'. This also improves the computational efficiency of the algorithm. Thus, despite the fact that the statement, 'building the smoothed version of the original function', appears quite frequently, we never actually explicitly build it! This is why the oracles as discussed above are so important to the functioning of the optimization algorithm.

Noisy feedbacks. Assuming that the noise ξ is zero mean and bounded, even if the oracles are noisy, the estimators still stay unbiased and bounded. Thus, all the preceding discussions still hold, and in our subsequent analysis we work with formulas that exclude ξ, for simplicity. As such, since we assume zero-mean and bounded noise, they don't affect our discussions and the discussions still stay valid.

5. Graduated Optimization with Gradient Feedback Oracle

Having discussed all the preliminaries, we now discuss the graduated optimization algorithm which uses the gradient feedback oracle.

Algorithm 3: Graduated Optimization via SGO_G

Input: error ε, probability p, convex set $\mathcal{K}$, f

Initiation: Select $\bar{x}_1$ from $\mathcal{K}$ at random, uniformly, δ_1, M, $\tilde{p} = \frac{p}{M}$, α_0

for m=1 to M

$\varepsilon_m = \frac{\sigma\delta_m^2}{32}$

$T_f = \frac{12480L^2}{\sigma\varepsilon_m}\log(\frac{2}{\tilde{p}} + 2\log\frac{12480L^2}{\sigma\varepsilon_m})$

Updating Decision Set,

$\mathcal{K}_m = \mathcal{K} \cap B(\bar{x}_m, 1.5\delta_m)$ // reducing the decision set

Calling Gradient Feedback Oracle,

GradOracle = SGO_G (δ_m) // defining a function GradOracle which is SGO_G with δ_m

Perform SGD,

$\bar{x}_{m+1} = SGD(T_f, \mathcal{K}_m, \bar{x}_m, GradOracle)$ // function call to SGD with the above parameters

Update δ_m,

$\delta_{m+1} = \delta_m/2$

end

Return: $\bar{x}_{M+1}$

Algorithm 4: SGD (Stochastic Gradient Descent)
Input: $T_f, \mathcal{K}_m = \mathcal{K}, \bar{x}_m = x_1, GradOracle$ for t=1 to T_f $\eta_t = \frac{1}{\sigma t}$ $g_t = GradOracle(x_t)$ $x_{t+1} = \Pi_{\mathcal{K}}(x_t - \eta_t g_t)$ end Return: x_{T_f+1}

The above two algorithms show how graduated optimization is performed on f, again, with a gradient feedback oracle.

Algorithm 3. The error ε is the final optimization error that we feed as input to the algorithm. The algorithm ensures that the final output is ε optimal with a probability greater than $1 - p$. This p, which is the failure probability, is fed as input to the algorithm as well. The optimization is to be performed for a function f which is L-Lipschitz and σ-nice over a convex set $\mathcal{K}$.

$$\delta_1 = \frac{diam(\mathcal{K})}{2} \; where \; diam(\mathcal{K}) \; is \; the \; diameter \; of \; the \; convex \; set$$

$$M = \log_2 \frac{1}{\alpha_0 \varepsilon} \; where \; M \; is \; the \; total \; epochs$$

$$\alpha_0 = min\left\{\frac{1}{2Ldiam(\mathcal{K})}, \frac{2\sqrt{2}}{\sqrt{\sigma} diam(\mathcal{K})}\right\}$$

The above parameters define the initiation portion of the algorithm, which weren't assigned in the algorithm itself for brevity.

In each of the M epochs for algorithm 3, we first calculate a local error ε_m. This ε_m then defines the number of iterations T_f that will be performed for the SGD over $\hat{f}_\delta(x)$. In each of the M epochs, the SGD performed is initiated with the outcome of the SGD performed in the previous epoch. Also, note that in each epoch the smoothing parameter δ reduces. Thus, we reach closer and closer to x^*, the minimizer for $f(x)$.

Algorithm 4. This algorithm takes as its input the iterations T_f, the 'reduced convex set $\mathcal{K}_m$' over which we are performing the projected stochastic gradient descent, the GradOracle function which is the SGO_G oracle as defined prior and finally the initial point x_1 which is the optimized point obtained in the previous epoch (output of SGD in the previous epoch).

Convergence theorem for SGO_G $\exists\, \varepsilon \in (0,1)\; and \; \exists\, p \in \left(0, \frac{1}{e}\right)$, given that $f: \mathcal{K} \dashrightarrow R$, where $\mathcal{K}$ is a convex set, and f is a L-Lipschitz, σ-nice function. Algorithm 3 (with Algorithm 4 a part of it, obviously) when applied to such an f to gradually optimize f guarantees that after $O(\frac{1}{\sigma^2 \varepsilon^2})$ epochs/iterations the output $\bar{x}_{M+1}$ obtained is ε-optimal with a probability greater than $1 - p$.

The convergence theorem states that for L-Lipschitz, σ-nice functions, using the algorithm as prescribed above, we can obtain an ε-optimal minimizer for such f, with a probability greater than $1-p$, in $O(\frac{1}{\sigma^2\varepsilon^2})$ epochs.

As initially discussed, the algorithm presented above optimizes smooth version of the original function in each epoch (via SGD). The algorithm doesn't need to build this smooth function explicitly. The smoothing parameter becomes smaller and smaller after each epoch, and thus the smoothed function gets closer and closer to the original function (after each epoch). Finally, the optimization is initiated in each epoch with the value of the minimizer obtained in the previous epoch, for the previous smooth function, thereby hastening the optimization in each epoch.

5.1. Analysis of the SGO$_G$ Algorithm

We first refer to a result for σ-strong convex functions:

Convergence of σ-strong convex functions. $\exists\ p \in (0,\frac{1}{e})$ and for a function F, which is σ-strong convex, say the unbiased estimator for $\nabla F(x)$ used in its SGD optimization is upper bounded by $\exists\ G$, then the following property holds true for its optimization, with a probability $\geq 1-p$:

$$F\left(\bar{x}_{T_f}\right) - F(x^*) \leq \frac{6240\log(\frac{2\log T_f}{p})G^2}{\sigma T_f}$$

Where, $\bar{x}_{T_f}$ is the SGD output after no more than T_f iterations and x^* is the minimizer for F over a convex set $\mathcal{K}$. Additionally, if,

$$T_f \geq \frac{12480G^2}{\sigma\varepsilon}\log(\frac{2}{p} + 2\log(\frac{12480G^2}{\sigma\varepsilon}))$$

We will have the excess loss ($F\left(\bar{x}_{T_f}\right) - F(x^*)$) smaller than ε, with a probability $\geq 1-p$.

While this particular property of σ-strong convex functions can't directly be used on our non-convex function f, since f is σ-nice, it implies that smooth versions of f are strongly convex, locally, over a neighborhood surrounding its minimizer (the minimizer of the smooth function, that is). Thus, we can apply this property indirectly on a L-Lipschitz, σ-nice non-convex function f.

The main Lemma, that ties the entire analysis of the algorithm together, has been stated thus:

Key Lemma. In Algorithm 3, we have defined M, $\mathcal{K}_m$ and $\bar{x}_{m+1}$. Say the minimizer of $\hat{f}_{\delta_m}(x)$ is x_m^*. Then the following holds $\forall\ m \in \{1,2,\dots,M\}$ with a probability $\geq 1-\frac{p}{M}$:

1. Over the defined set $\mathcal{K}_m$, $\hat{f}_{\delta_m}$ is σ-strongly convex and $x_m^* \in \mathcal{K}_m$
2. $\hat{f}_{\delta_m}(\bar{x}_{m+1}) - \hat{f}_{\delta_m}(x_m^*) \leq \frac{\sigma\delta_{m+1}^2}{8}$

Discussion of proof of Key Lemma. The above Lemma has been proved in the paper via induction. Note that the input (initial input) to the SGD in $epoch = m$ is $\bar{x}_m$ and the SGD outputs the minimizer estimate for $\hat{f}_{\delta_m}(x)$ which is $\bar{x}_{m+1}$. The actual minimizer for $\hat{f}_{\delta_m}(x)$ is taken to be x_m^*.

For m=1, we have,

$$\mathcal{K}_1 = \mathcal{K}$$

Due to δ_1 = diam ($\mathcal{K}$)/2. Thus, obviously we then have $x_1^* \in \mathcal{K}_1$. Since f is σ-nice, we have $\hat{f}_{\delta_1}(x)$ as σ-strongly convex. Property 2 then follows for m=1, directly via the fact that $\hat{f}_{\delta_1}(x)$ is σ-strongly convex. We have seen the convergence property of σ-strongly convex functions, the same property has been applied to obtain the Property 2 of the Lemma for m=1, noting that,

$$G = L$$

Which again follows from the properties of Gradient Feedback Oracle (which is unbiased and upper bounded by L).

Having proved the properties of the Key Lemma for m=1, we now apply induction to prove it $\forall\, m$. Assume that the properties hold for $m > 1$. We now attempt to prove it for the $m + 1$ case.

Both the properties obviously hold for m. Thus $\hat{f}_{\delta_m}$ is σ-strongly convex over $\mathcal{K}_m$.

Now, using the below property for σ-strong convex functions,

$$\frac{\sigma}{2}\|x - x^*\|^2 \leq F(x) - F(x^*)$$

With the following replacements,

$$x \to \bar{x}_{m+1}$$

$$x^* \to x_m^*$$

$$F \to \hat{f}_{\delta_m}$$

And using the Property 2 of the Key Lemma (assumed true for the epoch m), we obtain the following bound:

$$\|\bar{x}_{m+1} - x_m^*\| \leq \frac{\delta_{m+1}}{2}$$

Also note that the smoothing parameter is halved after every epoch, for use in the subsequent epoch. Thus, using the centering property of σ-nice functions, as below,

$$\left\|x_\delta^* - x_{\delta/2}^*\right\| \leq \delta/2$$

Again, with the following replacements,

$$x_\delta^* \to x_m^*$$

$$x_{\delta/2}^* \to x_{m+1}^*$$

$$\delta \to \delta_m$$

We obtain the following bound,

$$\|x_m^* - x_{m+1}^*\| \leq \delta_m/2$$

Also, noting that $\delta_{m+1} = \delta_m/2$ and applying the triangle inequality to the bounds obtained above on the points: $x_m^*, x_{m+1}^*\ and\ \bar{x}_{m+1}$, we obtain the following bound:

$$\|\bar{x}_{m+1} - x_{m+1}^*\| \leq 1.5\delta_{m+1}$$

The above bound implies the following:

$$x_{m+1}^* \in B(\bar{x}_{m+1}, 1.5\delta_{m+1})$$

This ball defined above is nothing but the convex set $\mathcal{K}_{m+1}$, that is,

$$\mathcal{K}_{m+1} = B(\bar{x}_{m+1}, 1.5\delta_{m+1})$$

Which completes the proof for the fact that $x^*_{m+1} \in \mathcal{K}_{m+1}$

Also, we know that f is σ-nice. Thus f follows the following property, as we have seen before,

$\hat{f}_{\delta_{m+1}}(x)\ over\ B(x^*_{m+1}, r_{\delta_{m+1}})$ is σ-strongly convex. Here,

$$r_{\delta_{m+1}} = 3\delta_{m+1}$$

Also note that,

$$B(\bar{x}_{m+1}, 1.5\delta_{m+1}) \subset B(x^*_{m+1}, 3\delta_{m+1})$$

Which again follows directly from the fact that $x^*_{m+1} \in \mathcal{K}_{m+1}$. The above fact implies that,

$\hat{f}_{\delta_{m+1}}(x)\ over\ \mathcal{K}_{m+1}$ is σ-strongly convex, completing the proof of Property 1 of Key Lemma.

Property 2 then follows from the convergence property of σ-strongly convex functions and the proved property 1 of the Key Lemma. While using the convergence property of σ-strongly convex functions to prove the Property 2 of the Key Lemma, we note the following relationship between ε_m and δ_m in each epoch.

$$\varepsilon_m = \frac{\sigma\delta_m^2}{32}$$

Using this relationship, and the ε optimal property of the convergence of σ-strongly convex functions, as obtained previously, we obtain the Property 2 of the Key Lemma in the form that it appears in, in the paper.

Failure probability. The probability of failure in each epoch is $\frac{p}{M}$.

$$p|F, m \leq \frac{p}{M}$$

Where $p|F, m$ indicates the failure probability in any epoch m.

Overall failure occurs if even a single epoch fails.

Thus,

$$p|F = 1 - p|S$$

Where $p|S$ is the probability of overall success.

Overall success occurs if none of the epochs fail.

Thus,

$$p|S \geq \left(1 - \frac{p}{M}\right)^M$$

Since p is rather small (chosen so) and M is relatively large, $\frac{p}{M} \ll 1$, and thus,

$$p|S \geq 1 - p$$

Thus, the overall probability of success is approximately greater than $1 - p$.

Finally, using the Key Lemma, we can now prove the convergence theorem for SGO_G

Proof of Convergence Theorem for SGO_G Using Key Lemma, M as defined above, α_0 as defined above, along with the bias result between $\hat{f}_\delta(x)$ and $f(x)$, we can easily prove the following statement for SGO_G, using direct algebraic steps,

$$f(\bar{x}_{M+1}) - f(x^*) \leq \varepsilon$$

Given that,

$$total\ iterations \leq T|total$$

Where,

$$T|total \leq \sum_{i=1}^{M} T_f^i$$

Here, T_f^i are the total iterations required by the SGD in the i^{th} epoch.

After a few algebraic manipulations, we obtain,

$$T|total \leq \left[14 \times 10^4 L^2 \log(\Gamma) \max\left\{16L^2, \frac{\sigma}{2}\right\}\right]\left[\frac{1}{\sigma^2\varepsilon^2}\right]$$

Where,

$$\Gamma = \frac{2\mathrm{M}}{\mathrm{p}} + 2\log\left(\frac{12480L^2}{\sigma\varepsilon_M}\right)$$

$$\Gamma \leq \frac{2\mathrm{M}}{\mathrm{p}} + 2\log\left(\frac{4 \times 10^5 L^2 \max\{16L^2, \frac{\sigma}{2}\}}{\sigma^2\varepsilon^2}\right)$$

Thus, this completes the analysis of Graduated Optimization via SGO_G

6. Graduated Optimization with Value Feedback Oracle

The algorithm for SGD stays the same as before, and is still Algorithm 4, however there are minor modifications in the Algorithm 3. We now use the value feedback oracle as opposed to the gradient feedback oracle, where we have already discussed the value feedback oracle before, and named the subroutine for generating the value feedback oracle as SGO_V.

The Algorithm 5 in the next page shows the algorithm for graduated optimization via the value feedback oracle.

As seen in the algorithm, there isn't much difference from the previous algorithm for SGO_G

Additionally, note that

$$M, \alpha_0, \delta_1\ stay\ the\ same\ for\ SGO_V$$

Thus, these parameters are defined in the same manner here as they were for SGO_G

Algorithm 5: Graduated Optimization via SGO_V

Input: error ε, probability p, convex set $\mathcal{K}$, f

Initiation: Select $\bar{x}_1$ from $\mathcal{K}$ at random, uniformly, δ_1, M, $\tilde{p} = \frac{p}{M}$, α_0

for m=1 to M

$\varepsilon_m = \frac{\sigma\delta_m^2}{32}$

$T_f = \frac{12480d^2C^2}{\sigma\varepsilon_m\delta_m^2}\log(\frac{2}{\tilde{p}} + 2\log\frac{12480d^2C^2}{\sigma\varepsilon_m\delta_m^2})$

Updating Decision Set,

$\mathcal{K}_m = \mathcal{K} \cap B(\bar{x}_m, 1.5\delta_m)$ // reducing the decision set

Calling Value Feedback Oracle,

GradOracle = SGO_V (δ_m) // defining a function GradOracle which is SGO_V with δ_m

Perform SGD,

$\bar{x}_{m+1} = SGD(T_f, \mathcal{K}_m, \bar{x}_m, GradOracle)$ // function call to SGD with the above parameters

Update δ_m,

$\delta_{m+1} = \delta_m/2$

end

Return: $\bar{x}_{M+1}$

Algorithm 5. Again, not much difference exists between functioning of the algorithm 3 and algorithm 5 other than the fact that for the current algorithm, the oracle used to estimate the gradient of the smooth version of the function is obtained via the random sampling of the value of the original function, as has been discussed before. Again, this oracle is both unbiased and bounded, however the bounding parameter now is changed.

Convergence theorem for SGO_V $\exists\ \varepsilon > 0\ and\ \exists\ p \in \left(0, \frac{1}{e}\right)$, given that $f: \mathcal{K} \dashrightarrow R$, where $\mathcal{K}$ is a convex set, and f is a L-Lipschitz, σ-nice function. Assuming that $|f(x)| \leq C$, Algorithm 5 (with Algorithm 4 a part of it, obviously) when applied to such an f to gradually optimize f guarantees that after $O(\frac{d^2}{\sigma^2\varepsilon^4})$ epochs/iterations the output $\bar{x}_{M+1}$ obtained is ε-optimal with a probability greater than $1 - p$.

Note that the d here is the d in $x \in R^d$.

6.1. Analysis of the SGO$_V$ Algorithm

The analytical procedure followed for SGO$_G$ applies here as well.

Note that, in order to use the convergence of σ-strong convex functions here, we need to make the following minor change,

$$G = \frac{Cd}{\delta_m}$$

This, as we've seen before, is the upper bound of the value feedback oracle.

The Key Lemma for SGO$_V$ algorithm, stated without proof due to similarity with the previous proof, is as follows:

Key Lemma. In Algorithm 5, we have defined M, $\mathcal{K}_m$ and $\bar{x}_{m+1}$. Say the minimizer of $\hat{f}_{\delta_m}(x)$ is x_m^*. Then the following holds $\forall\ m\ \in \{1,2,\dots,M\}$ with a probability $\geq 1 - \frac{p}{M}$:

1. Over the defined set $\mathcal{K}_m$, $\hat{f}_{\delta_m}$ is σ-strongly convex and $x_m^* \in \mathcal{K}_m$
2. $\hat{f}_{\delta_m}(\bar{x}_{m+1}) - \hat{f}_{\delta_m}(x_m^*) \leq \frac{\sigma\delta_{m+1}^2}{8}$

The Key Lemma is same as the one obtained previously and so is the proof. Thus, we exclude those details. Discussions of the failure probability are also similar.

Proof of Convergence Theorem for SGO$_V$ Again the steps of this proof follow directly from the steps of the proof for SGO$_G$. The relevant statements go thus,

Algorithm 5 outputs the point $\bar{x}_{M+1}$, such that,

$$f(\bar{x}_{M+1}) - f(x^*) \leq \varepsilon$$

Given that,

$$total\ iterations \leq T|total$$

Where,

$$T|total \leq \sum_{i=1}^{M} T_f^i$$

Here, T_f^i are the total iterations required by the SGD in the i^{th} epoch. Summing the T_f^i over all the epochs, we obtain,

$$T|total \leq \left[6 \times 10^4 C^2 \log(\Gamma) \max\left\{256L^4, \frac{\sigma^2}{4}\right\}\right]\left[\frac{d^2}{\sigma^2\varepsilon^4}\right]$$

Where,

$$\Gamma = \frac{2\mathrm{M}}{\mathrm{p}} + 2\log\left(\frac{12480d^2C^2}{\sigma\varepsilon_M\delta_M^2}\right)$$

$$\Gamma \leq \frac{2\mathrm{M}}{\mathrm{p}} + 2\log\left(\frac{4 \times 10^5 d^2 C^2 \max\{256L^4, \frac{\sigma^2}{4}\}}{\sigma^2 \varepsilon^4}\right)$$

This completes the analysis of Graduated Optimization via SGO_V

7. Discussing the Results in the Paper

The SGO_G algorithm and the MSGD (Mini Batch Stochastic Gradient Descent) algorithm was used to train a neural network with a single hidden layer consisting of 30 neurons. The NN was trained over the popular MNIST dataset (loss function = squared error, activation via ReLU).

An interesting issue highlighted by the paper was the fact that the MSGD algorithm has a tendency to 'stall' in regions where the objective function (to optimize) takes the shape of an extremely narrow valley. MSGD's 'view' of the optimal point is blocked by this valley, thus slowing down its progression towards the optimal.

In contrast, by implicitly smoothing the function (objective function), as discussed earlier, and optimizing over it, the valley has 'been smoothed' allowing the optimization over the smoothed function to progress quickly. The valley is still there, and hasn't completely disappeared, however in the smoothed function the valley is not as steep and thus doesn't 'block the view' of the SGD algorithm as much. It was seen that the larger the smoothing parameter, the more the valley was smoothed and the SGD progressed more quickly.

Comparison between the learning curves of MSGD and SGO_G clearly indicate that SGO_G learns much more quickly, and thus converges to an ε optimal point more quickly than MSGD.

8. Possible Future Work

In the initial project proposal, my plan was to implement the given algorithm in this paper to non-convex lasso optimization problem and compare the results with the traditional lasso convex-optimization problem, along with comparisons to the widely used Sparse-Net algorithm to optimize non-convex lasso. However due to the situations in the current semester, I was unable to fully implement them and had to stick with the literature review format. Having implemented the Sparse-Net algorithm in MATLAB, in future work, I plan on implementing the algorithms presented here to the non-convex lasso and work on a research paper on the same.