**CME 323: Distributed Algorithms and Optimization, Spring 2020**
`http://stanford.edu/~rezab/dao`.
Instructor: Reza Zadeh, Matroid and Stanford.

**Lecture 8, 4/23/2020. Lecture notes by Robin Brown**

# 8 Introduction to Optimization for Machine Learning

We will now shift our focus to unconstrained problems with a separable objective function, which is one of the most prevalent setting for problems in machine learning. Formally stated, we wish to solve the following problem:

$$\underset{w}{\text{minimize}} \quad F(w) := \sum_{i=1}^{n} F_i(w, x_i, y_i) \tag{1}$$

where we can interpret $x_i$'s as the input data, the $y_i$'s as the output data, and $w$ as the parameter we wish to optimize over. The $F_i$'s are objective functions ("loss functions") that are designed for the problem at hand. A few commonly used examples include:

- **Least squares regression** Here the hypothesis is that the output data $y_i$ is explained by the input data $x_i$ by a linear equation and some noise. Specifically, we postulate that $y_i = +\epsilon_i$. The squared error loss function is given by $F_i(w, x_i, y_i) = ||w^T x_i + b - y_i||_2^2 = \|\epsilon_i\|_2^2$
- **Support vector machine (SVM)** Here the outputs mark each of the $x_i$'s as coming from one of two categories, $-1$ or 1, and the goal is to build a model that assigns new examples to one of the two categories. If the goal is "maximum-margin" classification, the hinge loss is appropriate. The hinge loss is given by $F_i(w, x_i, y_i) = \max\{0, 1 - y_i(w^T x_i)\}$. When $w^T x_i$ and $y_i$ have opposite signs (meaning incorrect classification), the loss increases linearly with $|w^T x_i|$. When the have the same sign (meaning correct classification) and $|w^T x_i| \geq 1$ the loss is zero, but if $|w^T x_i| < 1$ a loss is still incurred because the point is not classified with enough margin (hence why it is good for "maximum-margin" classification). (convex)

In this setting, the optimization problem has some aspects that are suited for distributed computing, such as regularization and hyperparameter tuning, but these are quite straightforward and not particular interesting from an algorithmic or distributed design perspective. The two quantities we are interested in with regards to the scaling such algorithms are:

- **Data parallelism**: How does the algorithm scale with $n$ (number of training points)?
- **Model parallelism**: How does the algorithm scale with $d$ (number of parameters)?

Gradient descent, and stochastic gradient descent are some of the more widely used methods for solving this optimization problem. In this lecture, we will first prove the convergence rate of gradient descent (in the serial setting); the number of iterations needed to reach a desired error tolerance

will inherently determine the total depth of the (stochastic) gradient descent algorithm. We will then discuss how gradient descent and stochastic gradient descent trade-off work per iteration and number of iterations needed for convergence. This section will conclude with HOGWILD!, an algorithm that exploits sparsity to circumvent the inherently sequential aspect of computing iterations.

## 8.1 Gradient Descent

The general idea of gradient descent is based on initializing $w$ at random and performing the sequential updates:

$$w_{k+1} \leftarrow w_k - \alpha \nabla f(w_k),$$

where the convergence depends on $\alpha$ and the structural properties of $F$ itself.

## 8.2 Convergence of Gradient Descent

In general, function minimization is impossible unless we make some sort of structural assumption about the function itself (imagine a discontinuous function that jumps around arbitrarily at each point!). Fortunately, for most practice problems it's reasonable to impose some assumptions about the structure of the problem, namely convexity[1] and smoothness[2]. Theorems 8.3 and 8.7 state the convergence rates of gradient descent on $L$-smooth and $L$-smooth, $\mu$-strongly convex functions respectively. This section will be dedicated to building up the necessary background for proving these theorems.

**Definition 8.1 ($L$-smooth)** *A differentiable function $f : \mathbb{R}^n \to \mathbb{R}$ is said to be $L$-smooth is for all $x, y \in \mathbb{R}^n$, we have that*

$$\|\nabla f(x) - \nabla f(y)\|_2 \leq L \|x - y\|$$

The gradient of a functions measures how the function changes when we move in a particular direction from a point. If the gradient were to change arbitrarily quickly, the old gradient does not give us much information at all even if we take a small step. In contrast, smoothness assures us that the gradient cannot change too quickly—therefore we have an assurance that the gradient information is informative within a region around where it is taken. The implication is that we can decrease the function's value by moving the direction opposite of the gradient.

**Lemma 8.1** *Let $f : \mathbb{R}^n \to \mathbb{R}$ be differentiable, then for all $x, y \in \mathbb{R}^n$ and $x_t = x + t(x - y)$ for $t \in [0, 1]$. Then,*

$$f(x) - f(y) = \int_0^1 \nabla f(x_t)^T (y - x) dt$$

*And so,*

$$f(x) - f(y) - \nabla f(x)^T (y - x) = \int_0^1 (\nabla f(x_t) - \nabla f(x))^T (y - x) dt$$

[1] Although most problems in machine learning are not convex, convex functions are among the easiest to minimize, making their study interesting

[2] We can also often forgo the smoothness assumption by using subgradients instead of gradients. We will assume smoothness for illustrative purposes, because extensions to the nonsmooth case are straightfoward

**Lemma 8.2** *If $f : \mathbb{R}^n \to \mathbb{R}$ be L-smooth. Then for all $x, y \in \mathbb{R}^n$ we have that*

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

We can now analyse the convergence of gradient descent on $L$-smooth functions.

**Theorem 8.3** *Gradient descent on L-smooth functions, with a fixed step-size of $\frac{1}{L}$ achieve an $\epsilon$-critical point in $\frac{2L(f(x_0)-f_*)}{\epsilon^2}$ iterations.*
**Proof:** *Applying Lemma 8.10, we get*

$$|f(w_{k+1}) - (f(w_k) + \alpha \|\nabla f(w_k)\|_2^2)| \leq \frac{\alpha^2 L}{2} \|\nabla f(w_k)\|_2^2$$

*So if we pick $\alpha = \frac{1}{L}$, then $f(w_{k+1}) \leq f(w_k) - \frac{1}{2L} \|\nabla f(w_k)\|_2^2$. So, in k gradient descent steps, we have an iterate $w_k$ such that*

$$f_* - f(w_0) \leq f(w_k) - f(w_0) \leq \frac{1}{2L} \sum_{i=0}^{k-1} \|\nabla f(w_i)\|_2^2 .$$

*Rearranging, we get*

$$\frac{1}{k} \sum_{i=0}^{k-1} \|\nabla f(w_i)\|_2^2 \leq \frac{2L(f(w_0) - f_*)}{k}$$

*So for one of the $w_i$'s, $\|\nabla f(w_i)\|_2^2 \leq \frac{2L(f(w_0)-f_*)}{k}$. So, if we want to find an $\epsilon$-critical point i.e., a point where $\|\nabla f(w)\|_2 \leq \epsilon$, we need at least $\frac{2L(f(x_0)-f_*)}{\epsilon^2}$ iterations.* ■

Remember in the previous lecture we mentioned that guarantees that we are close to a global optimum are hard to come by? This is one of those cases, so we need to settled for an $\epsilon$-critical point instead.

One of the assumptions that will allow us to obtain global optimality guarantees is convexity.

**Definition 8.2 ($\mu$-strongly convex)** *A function $f : \mathbb{R}^n \to \mathbb{R}$ is said to be $\mu$-strongly convex for $\mu \geq 0$ if for all $x, y \in \mathbb{R}^n$ and $t \in [0, 1]$ we have that*

$$f(ty - (1 - t)x) \leq tf(y) + (1 - t)f(x) - \frac{\mu}{2} t(1 - t) \|x - y\|_2^2$$

A function is said to be *convex* if the above relation holds for $\mu = 0$, and typically *strongly convex* implies that $\mu > 0$. We will first state a couple helper lemmas that will be will be useful in analyzing gradient descent on $\mu$-strongly convex functions (their proofs are included in the appendix for the interested reader).

**Lemma 8.4** *1. A differentiable function is $\mu$-strongly convex if and only for all $x, y \in \mathbb{R}^2$,*

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

*2. A twice differentiable function $f$ is $\mu$-strongly convex if and only if for all $x \in \mathbb{R}^n$*

$$z^T \nabla^2 f(x) z \geq \mu \|z\|_2^2$$

Ultimately, convexity will allow us to translate gradient information into distance from the optimum and suboptimality of the function value. Previously, we were able to find an $\epsilon$-critical point using gradient descent only assuming $L$-smoothness. This ends up not being very informative because when a function is non-convex, its gradients can equal zero at a saddle point or a maximum—these points are certainly not optimal. However, this cannot be the case for convex functions. In particular, if we define $X_*(f) := \{x | f(x) = f_*\}$ to be the set of minimizers of $f$, then $x_* \in X_*(f)$ iff and only if $\nabla f(x_*) = 0$.

**Lemma 8.5** *If $f : \mathbb{R}^n \to \mathbb{R}$ is $L$-smooth, then for all $x_* \in X_*(f)$ and $x_* \in \mathbb{R}^n$,*

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

We can prove similar bounds if $f$ is $\mu$-strongly convex.

**Lemma 8.6** *If $f : \mathbb{R}^n \to \mathbb{R}$ is smooth and $\mu$-strongly convex for $\mu > 0$, then for all $x_* \in X_*(f)$ and $x_* \in \mathbb{R}^n$,*

$$\frac{1}{2\mu} \|\nabla f(x)\|_2^2 \geq f(x) - f_* \geq \frac{\mu}{2} \|x - x_*\|_2^2$$

Now that we can relate gradient information to suboptimality and distance from an optimum, we can determine the convergence rate of gradient descent for strongly convex functions.

**Theorem 8.7 (Strongly Convex Gradient Descent)** *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a $L$- smooth, $\mu$-strongly convex function for $\mu > 0$. Then for $x_0 \in \mathbb{R}^n$ let $x_{k+1} = x_k - \frac{1}{L}\nabla f(x_k)$ for all $k \geq 0$. Then,*

$$f(x_k) - f_* \leq \left(1 - \frac{\mu}{L}\right)^k (f(x_0) - f_*).$$

*Consequently, we require $\frac{L}{\mu} \log\left(\frac{f(x_0)-f_*}{\epsilon}\right)$ iterations to find an $\epsilon$-optimal point.*

**Proof:** *We have previously shown that $f(x_{k+1}) \leq f(x_k) - \frac{1}{2L} \|\nabla f(x_k)\|_2^2$. From Lemma 8.6, we know that $\|\nabla f(x_k)\|_2^2 \geq 2\mu(f(x_k) - f_*)$. Putting this all together,*

$$\begin{aligned} f(x_{k+1}) - f_* &\leq f(x_k) - f_* - \frac{1}{2L} \|\nabla f(x_k)\|_2^2 \\ &\leq f(x_k) - f_* - \frac{\mu}{L}(f(x_k) - f_*) \\ &= (1 - \frac{\mu}{L})(f(x_k) - f_*) \end{aligned}$$

*So, applying this bound repeatedly gives us*

$$f(x_k) - f_* \leq \left(1 - \frac{\mu}{L}\right)^k (f(x_0) - f_*).$$

*Using the fact that $1 + x \leq e^x$, and picking $k \geq \frac{L}{\mu} \log\left(\frac{f(x_0)-f_*}{\epsilon}\right)$ gives the convergence rate.* ■

We now have convergence rates for gradient descent. The next important step is understanding the complexity of computing the gradient, and with that some of the challenges of parallelizing gradient descent. We will use linear least squares as an illustrative example.

### 8.3 Analyzing least squares gradient descent

The linear least squares objective is given by

$$F(w) = \sum_{i=1}^{n} F_i(w, x_i, y_i) = \sum_{i=1}^{n} ||x_i^T w - y_i||_2^2.$$

This has work $O(nd)$ and depth $O(\log n + \log d)$. This object function has gradient

$$\nabla_w F = \sum_{i=1}^{n} \nabla_w F_i(w) = \sum_{i=1}^{n} 2(x_i^T w - y_i)x_i \quad \in R^d$$

Here, we see that the gradient is just a re-scaling of $x_i$. Computing the gradient for a single datapoint can be done in $O(\log d)$ depth (the computation being dominated by the dot product operation). The global gradient $\nabla_w F$ then requires $O(\log d) + O(\log n)$ depth. The previous analysis shows that to achieve $\epsilon$ error, gradient descent requires $\mathcal{O}(\log \frac{1}{\epsilon})$ iterations. Thus total depth for gradient descent is $\mathcal{O}(\log \frac{1}{\epsilon} (\log n + \log d))$, which means that our algorithm is slow even with multiple processors. This is because each of the iterations is inherently sequential, and thus not easily parallelized.

### 8.4 Stochastic Gradient Descent

Stochastic gradient descent (SGD) reduces the work of gradient descent by "sampling" the gradient. Rather than computing all of $\nabla F = \sum_{i=1}^{n} \nabla F_i(w, x_i, y_i)$, at each iteration, we randomly pick some subset of the $F_i(w, x_i, y_i)$ and compute their gradients. Why does this work? If we sample uniformly at random, in expectation, the sampled gradients will equal the true gradients. Let $s_k$ be index uniformly sampled at time step $k$ (we use one training point for illustrative purposes. However, you can sample batches of gradients as well). Then, the SGD iteration as follows:

$$w_{k+1} \leftarrow w_k - \alpha \nabla F_{s_k}(w_k).$$

We have reduced the work per iterations by $n$, which is a drastic reduction if our training set is very large (as they tend to be). However, this comes at the cost of a slower convergence rate (in terms of the number of iterations) than gradient descent.

**Theorem 8.8** *If there exists a constant $G$ such that $\mathbb{E}[\|\nabla f_i(x)\|^2] \leq G^2$ and $f(x)$ is $\mu$-strongly convex. Then, with step-sizes $\gamma_k = \frac{1}{\mu k}$, we have*

$$\mathbb{E}[\|x_k - x_*\|^2] \leq \frac{\max\{\|x_1 - x_*\|^2, \frac{G^2}{\mu^2}\}}{k}.$$

*This means that SGD achieves $\epsilon$ error after $\mathcal{O}(\frac{1}{\epsilon})$ iterations.*

**Proof:** *The proof will be given as an exercise in the homework, and has been omitted from these notes.* ■

Notice the $\mathcal{O}(\cdot)$ notation hides constants such as the Lipschitz constant, depending on $F$, which can be pretty large.

The table below summarizes the performance of gradient descent vs stochastic gradient descent[3] where $\epsilon$ is our tolerance for convergence. (To get this table, we have assumed $F_i$ has a reasonable Lipschitz constant, and that evaluating each $F_i$ requires $O(d)$ work and $\log(d)$ depth.)

| | # iterations | work/iter | depth/iter | total work | total depth |
|---|---|---|---|---|---|
| GD | $O(\log \frac{1}{\epsilon})$ | $O(nd)$ | $O(\log d + \log n)$ | $O(nd \log \frac{1}{\epsilon})$ | $O((\log \frac{1}{\epsilon})(\log d + \log n))$ |
| SGD | $O(\frac{1}{\epsilon})$ | $O(d)$ | $O(\log d)$ | $O(\frac{d}{\epsilon})$ | $O(\frac{\log d}{\epsilon})$ |

In the case of Gradient Descent, the number of iterations is trapped in a log expression, so total depth is as well (this is desirable). However, if we want an error of $\epsilon = 10^{-6}$, this becomes problematic in SGD, which requires many more iterations than GD. On the other hand, the total work for SGD does not depend on $n$, which is powerful if we are working with massive data sets. Because the iterations of SGD are inherently serial, they generally cannot be parallelized. However, when problems are sufficiently sparse[4] we can parallelize these updates.

We now consider a more refined statement of our problem:

$$\underset{w}{\text{minimize}} \quad F(w) := \sum_{i=1}^{n} F_i(w_e, x_i, y_i) \tag{2}$$

where $e \subseteq [d]$ denotes a subset of $\mathbb{R}^d$ and $w_e$ represents the appropriate components of $w$. We now just have to manage conflicts on $F_i$'s that share the same components of $w$.

Managing conflicts is tricky because CPU's do not have direct access to the memory; instead computations can only be done on CPU cache. Data is read from the memory to CPU cache, and after the computations are finished, results are pushed back to memory. Therefore, a processor has no way of knowing whether other processors have local updates that have not yet been pushed to memory.

Conflicts are typically dealt with with a "coordinate" or "memory lock" approach. An exclusive lock essentially tells other processors that one of the processors is currently updating a variable and prevents them from making changes until it is done. This can provide some speedup over serial SGD, and it is straightforward to prove correctness of the algorithm. However load imbalances result in idle-time of the processors, rendering the algorithm inefficient.

### 8.5 Hogwild!

HOGWILD! works by simply removing the locks. Processors have access to shared memory, and can potentially overwrite each others' work. This works if the problem is sufficiently sparse.

Many machine learning problems are sparse: for example matrix factorization, matrix completion, graph cuts, graph or text classification etc. Generally, this means if multiple processors try

[3] There's a hidden $O(\log(n))$ cost to sampling from a uniform random variable

[4] Although, the convergence HOGWILD! has only been proven for sparse problems, it has seems to work well for dense problems as well, and we do not really know why

to write to $w$ at the same time, "most" of the time they will be writing to different components of $w$. The proof of HOGWILD! then relies of proving that the instances where multiple processors try write to the same components of $w$ will not completely derail the convergence of SGD.

The main idea/insight in the proof is that HOGWILD! can be interpreted as a noisy serial SGD. The noise comes from the instances where multiple processors try write to the same components of $w$, and can be bounded using the sparsity of the "conflict graph", which represents instances where updates might try to write to the same component.

**Theorem 8.9** *Let $F$ be $\mu$-strongly convex, with $\|\nabla F_i\|^2 \leq M^2$. If the number of samples that overlap in time with a single sample during the execution of HOGWILD! is bounded as*

$$\tau = \mathcal{O}\left(\min\left\{\frac{n}{\overline{\Delta}_C}, \frac{M^2}{\epsilon\mu^2}\right\}\right)$$

*HOGWILD!, with stepsize $\gamma = \frac{\epsilon\mu}{2M^2}$ reaches an accuracy of $\mathbb{E}[\|x_k - x_*\|^2] \leq \epsilon$ after*

$$T \geq \mathcal{O}(1)\frac{M^2 \log\left(\frac{a_0}{\epsilon}\right)}{\epsilon\mu^2}$$

*iterations. Where $\overline{\Delta}_C$ is the average degree in the conflict graph.*

```
Initialize w in shared memory // in parallel, do
for i = {1, ..., p} do
    while TRUE do
        if stopping criterion met then
            break
        end
        Sample j from 1, ..., n uniformly at random.
        Compute f_j(w) and ∇f_j(w) using whatever w is currently available.
        Let e_j denote non-zero indices of x_i
        for k ∈ e_j do
            w_(k) ← w_(k) − α [∇F_j(w)]_(k)
        end
    end
end
```

**Algorithm 1:** HOGWILD!

Communication in GD/SGD happens at each iteration: $O(\frac{1}{\varepsilon})$ vs $O(\log \frac{1}{\varepsilon})$. Thus large differences in the cost of communication between a single machine with shared memory and distributed cluster may change the economics of the choice between GD and SGD. In practice, we see single machine architectures using SGD on big beefy GPUs for things like deep learning whereas we may see distributed systems running GD.

## 8.6 Appendix

**Lemma 8.10** *Let $f : \mathbb{R}^n \to \mathbb{R}$ be differentiable, then for all $x, y \in \mathbb{R}^n$ and $x_t = x + t(x - y)$ for $t \in [0, 1]$. Then,*

$$f(x) - f(y) = \int_0^1 \nabla f(x_t)^T (y - x) dt$$

*And so,*

$$f(x) - f(y) - \nabla f(x)^T (y - x) = \int_0^1 (\nabla f(x_t) - \nabla f(x))^T (y - x) dt$$

**Proof:** *Let $g(t) := f(x + t(x - y))$. Then, because $g'(t) = \nabla f(x_t)^T (y - x)$, the first claim follows from the fundamental theorem of calculus. The second claim follows from the first by subtracting $\nabla f(x)^T (y - x)$ from both sides of the equation.* ■

**Lemma 8.11** *If $f : \mathbb{R}^n \to \mathbb{R}$ be $L$-smooth. Then for all $x, y \in \mathbb{R}^n$ we have that*

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

**Proof:** *Let $x_t = x + t(x - y)$, then we can apply the previous lemma to get*

$$\begin{aligned} |f(y) - (f(x) + \nabla f(x)^T (y - x))| &\leq \left| \int_0^1 (\nabla f(x_t) - \nabla f(x))^T (y - x) dt \right| \\ &\leq \int_0^1 |(\nabla f(x_t) - \nabla f(x))^T (y - x)| dt \\ &\leq \int_0^1 \|\nabla f(x_t) - \nabla f(x)\| \, \|y - x\| \, dt \end{aligned}$$

*where the second and thrid inequality follow from Cauchy Schwartz.*
*We can now use the definition of smoothness to get a bound on $\|\nabla f(x_t) - \nabla f(x)\|$. Note that,*

$$x_t - x = x + t(x - y) - x = t(x - y).$$

*So, by smoothness, $\|\nabla f(x_t) - \nabla f(x)\| \leq t \|x - y\|$. Plugging this back into the integral,*

$$\begin{aligned} |f(y) - (f(x) + \nabla f(x)^T (y - x))| &\leq \int_0^1 \|\nabla f(x_t) - \nabla f(x)\| \, \|y - x\| \, dt \\ &\leq L \|x - y\|_2^2 \int_0^1 t dt \\ &= \frac{L}{2} \|x - y\|_2^2 \end{aligned}$$

*yields the result.* ■

**Lemma 8.12** *8.4*

1. *A differentiable function is $\mu$-strongly convex if and only for all $x, y \in \mathbb{R}^2$,*

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

2. *A twice differentiable function $f$ is $\mu$-strongly convex if and only if for all $x \in \mathbb{R}^n$*

$$z^T \nabla^2 f(x) z \geq \mu \|z\|_2^2$$

**Proof:** *Suppose that for all $x, y \in \mathbb{R}^2$,*

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

*Then, for a given $x, y \in \mathbb{R}^2$, let $x_t = x + t(y - x)$ where $t \in [0, 1]$. Then,*

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

*Similarly,*

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

*Taking $t$ times the first expression and $1 - t$ times the second expression, and combining the two yields,*

$$f(ty - (1 - t)x) \leq tf(y) + (1 - t)f(x) - \frac{\mu}{2} t(1 - t) \|x - y\|_2^2 .$$

*Now we assume*

$$f(ty - (1 - t)x) \leq tf(y) + (1 - t)f(x) - \frac{\mu}{2} t(1 - t) \|x - y\|_2^2$$

*to be true. Rearranging,*

$$\begin{aligned} tf(y) \geq f(ty - (1 - t)x) &\leq -(1 - t)f(x) + \frac{\mu}{2} t(1 - t) \|x - y\|_2^2 \\ f(y) &\geq \frac{f(ty - (1 - t)x) \leq -(1 - t)f(x) + \frac{\mu}{2} t(1 - t) \|x - y\|_2^2}{t} \\ &= f(x) + \frac{\mu}{2} (1 - t) \|y - x\|_2^2 + \frac{f(x_t) - f(x)}{t} \end{aligned}$$

*Since $\frac{f(x_t) - f(x)}{t}$ converges to $\nabla f(x)(y - x)$ as $t \to 0$. Taking the limit of expression above as $t \to 0$ yields,*

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

*Assume that for all $x \in \mathbb{R}^n$*

$$z^T \nabla^2 f(x) z \geq \mu \|z\|_2^2$$

*Then, for $x, y \in \mathbb{R}^n$ let $x_\alpha = x + \alpha(y - x)$ with $\alpha \in [0, 1]$,*

$$\begin{aligned} f(y) &= f(x) + \nabla f(x)^T (y - x) + \int_0^1 \int_0^t (y - x)^T \nabla^2 f(x_\alpha)(y - x) d\alpha dt \\ &\geq f(x) + \nabla f(x)^T (y - x) + \int_0^1 \int_0^t \mu \left\| y - x \right\|_2 d\alpha dt \\ &\geq f(x) + \nabla f(x)^T (y - x) + \frac{\mu}{2} \left\| y - x \right\|_2 \end{aligned}$$

*We can apply the previous lemma to get the result.*
*Now assume that $f$ is $\mu$-strongly convex, and for $x, t \in \mathbb{R}^n$, define $x_t = x + tz$ for all $t \in \mathbb{R}$. Let $g(t) = f(x_t)$. Then $g'(t) = \nabla f(x_t)^T z$ and $g''(t) = z^T \nabla^2 f(x_t) z$. Furthermore,*

$$g''(0) = \lim_{t \to 0} \frac{g'(t) - g'(0)}{t} = \lim_{t \to 0} \frac{(\nabla f(x_t) - \nabla f(x))^T z}{t} = \lim_{t \to 0} \frac{(\nabla f(x_t) - \nabla f(x))^T (x_t - x)}{t^2}$$

*Applying the previous lemma, we have that*

$$f(x_t) \geq f(x) + \nabla f(x)^T (x_{tx}) + \frac{\mu}{2} \left\| x_t - x \right\|^2$$

*and*

$$f(x) \geq f(x_t) + \nabla f(x_t)^T (x - x_t) + \frac{\mu}{2} \left\| x_t - x \right\|^2$$

*By definition, $x_t - x = tz$, so, adding these together yields*

$$(\nabla f(x_t) - (x))^T (x_t - x) \geq \mu t^2 \left\| z \right\|_2^2,$$

*proving our result.* ■

***Lemma 8.13*** *If $f : \mathbb{R}^n \to \mathbb{R}$ is $L$-smooth, then for all $x_* \in X_*(f)$ and $x_* \in \mathbb{R}^n$,*

$$\frac{1}{2L} \left\| \nabla f(x) \right\|_2^2 \leq f(x) - f_* \leq \frac{L}{2} \left\| x - x_* \right\|_2^2$$

**Proof:** *We will use the inequality,*

$$f(y) \leq f(x) + \nabla f(x)^T (y - x) + \frac{L}{2} \left\| y - x \right\|_2^2$$

*for all $x, y \in \mathbb{R}^n$. Consider $x, y$ where $y = x - \frac{1}{L} \nabla f(x)$. Then, from our previous analysis we have that $f(y) \leq f(x) - \frac{1}{2L} \left\| \nabla f(x) \right\|_2^2$. By definition of optimality, $f_* \leq f(y)$. So,*

$$f_* \leq f(y) \leq f(x) - \frac{1}{2L} \left\| \nabla f(x) \right\|_2^2$$

*and we have the left hand side of the inequality, $\frac{1}{2L} \left\| \nabla f(x) \right\|_2^2 \leq f(x) - f_*$.*

*Now, we plug in $x$ for $y$, and $x_*$ for $x$ to get,*

$$f(x) \leq f(x_*) + \nabla f(x_*)^T (x - x_*) + \frac{L}{2} \left\| x - x_* \right\|_2^2.$$

*Since $\nabla f(x_*) = 0$, this inequality becomes*

$$f(x) \leq f(x_*) + \frac{L}{2} \left\| x - x_* \right\|_2^2.$$

*Thus, proving the right side of the inequality.* ■

***Lemma 8.14*** *If $f : \mathbb{R}^n \to \mathbb{R}$ is smooth and $\mu$-strongly convex for $\mu > 0$, then for all $x_* \in X_*(f)$ and $x_* \in \mathbb{R}^n$,*

$$\frac{1}{2\mu} \|\nabla f(x)\|_2^2 \geq f(x) - f_* \geq \frac{\mu}{2} \|x - x_*\|_2^2$$

**Proof:** *The proof follows analogously to the L-smooth case. Here we use the relation*

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

*Plugging in $x$ for $y$, and $x_*$ for $x$ we get,*

$$f(x) \geq f(x_*) + \nabla f(x_*)^T (x - x_*) + \frac{\mu}{2} \|x_* - x\|_2^2$$

*Using the fact that $\nabla f(x_*) = 0$ yields,*

$$f(x) \geq f(x_*) + \frac{\mu}{2} \|x_* - x\|_2^2$$

*which is the right side of the inequality. To prove the left side, note that*

$$f(x_*) \geq \min_y f(y) + \nabla f(x)^T (y - x) + \frac{\mu}{2} \|x - y\|_2^2 \geq f(x) - \frac{1}{2\mu} \|\nabla f(x)\|_2^2$$

*since $y = x - \frac{1}{\mu} \|\nabla\| f(x)$ minimizes the middle expression.* ■

# References


*[1] B. Recht, C. Re, S. Wright, F. Niu* Hogwild: A Lock-Free Approach to Parallelizing Stochastic Gradient Descent. *NIPS, 2011.*