Lecture Notes for EC525: Optimization for Machine Learning

Ashok Cutkosky
Boston University
ashok@cutkosky.com

Contents

1 Machine Learning Basics

Machine learning is all about identifying some pattern in data and applying those patterns to new data. Formally, let us suppose that you have bunch of datapoints or observations $\mathbf{z}_1, \ldots, \mathbf{z}_N$. For example, each $\mathbf{z}_i$ could be a tuple $(\mathbf{x}_i, \mathbf{y}_i)$ where $\mathbf{x}_i$ is some object like a picture or the state of a board game or information about a medical patient, and $\mathbf{y}_i$ is some label like an object in the picture, the best move in the board game, or the diagnosis of the patient. The typical machine learning workflow consists of two main choices:

1. Choose some kind of *model* to explain the data. In supervised learning in which $\mathbf{z} = (\mathbf{x}, \mathbf{y})$, typically we pick some function f and use the model $\mathbf{y} \approx f(\mathbf{x}, \mathbf{w})$ where $\mathbf{w}$ is the *parameter* of the model. We will let W be the set of acceptable values for $\mathbf{w}$.
2. Fit the model to the data. That is, using our dataset $\mathbf{z}_1, \ldots, \mathbf{z}_N$, choose some value $\hat{\mathbf{w}}$. This is often called *training* the model.

For most of this course, will consider the case that $W = \mathbb{R}^d$ for some d so that we are attempting to minimize a real-valued function of a real vector.

For example, in a binary classification task if $\mathbf{z} = (\mathbf{x}, \mathbf{y})$ where $\mathbf{x} \in \mathbb{R}^d$ and $\mathbf{y} \in \{-1, 1\}$, we might choose the simple linear classifier model

$$\mathbf{y} \approx \text{sign}(\langle \mathbf{x}, \mathbf{w} \rangle)$$

After completing step 2, we will need some way to measure how good our choice of $\hat{\mathbf{w}}$ is. This is accomplished through a *loss function* $\mathcal{L}(\mathbf{w})$. Typically, the loss function takes the form of some kind of expectation:

$$\mathcal{L}(\mathbf{w}) = \mathop{\mathbb{E}}_{\mathbf{z}}[\ell(\mathbf{w}, \mathbf{z})]$$

where the expectation is over some unknown distribution of observations $\mathbf{z} \sim P_{\mathbf{z}}$. In this course, we will consider solely losses of this form. For example, in the binary classification task, we might write:

$$\ell(\mathbf{w}, \mathbf{z}) = \mathbb{1}[\mathbf{y} \neq \text{sign}(\langle \mathbf{x}, \mathbf{w} \rangle)]$$

This loss is 1 if we misclassify the example $\mathbf{z} = (\mathbf{x}, \mathbf{y})$ and zero otherwise. The full loss $\mathcal{L}(\mathbf{w}) = \mathbb{E}[\ell(\mathbf{w}, \mathbf{z})]$ is simply the probability of misclassifying a random example. In general, when $\mathbf{z} = (\mathbf{x}, \mathbf{y})$ and we have some model $y \approx f(\mathbf{w}, \mathbf{z})$, it is common for the loss to take the form:

$$\ell(\mathbf{w}, \mathbf{z}) = l(f(\mathbf{w}, \mathbf{z}), \mathbf{y})$$

However, in order to allow for greater generality we will not use this notion of l and f and consider just the composition ℓ.

With this idea, we can formally characterize the task of step 2: we are trying to find a $\hat{\mathbf{w}}$ that minimizes the loss $\mathcal{L}(\mathbf{w})$. Thus, step two is trying to identify:

$$\mathbf{w}_\star = \mathop{\text{argmin}}_{\mathbf{w} \in W} \mathcal{L}(\mathbf{w})$$

We can measure our ability to find the argmin by the *suboptimality gap*:

$$\mathcal{L}(\hat{\mathbf{w}}) - \mathcal{L}(\mathbf{w}_\star)$$

If we have a small suboptimality gap, then we have done a good job of training the model. Note that this does not mean that the model is actually good: it might be that $\mathcal{L}(\mathbf{w}_\star)$ is actually quite high because we made a poor choice of model in the first place.

This problem is difficult for two main reasons. The first of these is a *computational* barrier. It turns out that minimizing arbitrary functions $\mathcal{L}$ is computationally intractable (in fact, even the example just provided for linear classification is NP-hard to minimize). This issue is usually addressed by choosing some *surrogate* loss ℓ^S with $\mathcal{L}^S(w) = \mathbb{E}_z[\ell^S(w, z)]$. Then, we need to instead solve the problem:

$$\mathop{\text{argmin}}_{w \in W} \mathcal{L}^S(w)$$

The surrogate should be chosen such that:

1. $\mathcal{L}^S(w) \approx \mathcal{L}(w)$.

2. It is computationally feasible to minimize $\mathcal{L}^S$.

There is some natural tension here: surrogates that more accurately capture the original loss $\mathcal{L}$ may be harder to minimize.

The second reason the problem is difficult is that usually we do not even know how to evaluate the function $\mathcal{L}$!. In particular, we do not know the distribution over z in $\mathcal{L}(w) = \mathbb{E}_z[\ell(w, z)]$. This makes the problem of minimizing $\mathcal{L}(w)$ not just an optimization problem but also a *stochastic* optimization problem ("stochastic" is really just a fancy word for "involving randomness").

The classical way to deal with this is through *empirical risk minimization*. Here, we make use of a dataset $\mathbf{z}_1, \ldots, \mathbf{z}_N$ to define:

$$\hat{\mathcal{L}}^S(\mathbf{w}) = \frac{1}{N} \sum_{i=1}^{N} \ell^S(\mathbf{w}, \mathbf{z}_i)$$

We make a critical assumption: suppose that each $\mathbf{z}_i$ is drawn independently from the distribution $P_{\mathbf{z}}$. Intuitively, we should expect $\hat{\mathcal{L}}^S \approx \mathcal{L}^S$ since averages typically approach their mean. Then, instead of trying to minimize $\mathcal{L}^S$, which we do not know, we can minimize $\hat{\mathcal{L}}^S$. The function $\hat{\mathcal{L}}^S(\mathbf{w})$ is the empirical loss. This leads to the overall suboptimality gap:

$$\mathcal{L}(\hat{\mathbf{w}}) - \mathcal{L}(\mathbf{w}_\star) = \underbrace{\mathcal{L}(\hat{\mathbf{w}}) - \mathcal{L}^S(\hat{\mathbf{w}})}_{\text{surrogate quality}} + \underbrace{\mathcal{L}^S(\hat{\mathbf{w}}) - \hat{\mathcal{L}}^S(\hat{\mathbf{w}})}_{\text{empirical approximation}} + \underbrace{\hat{\mathcal{L}}^S(\hat{\mathbf{w}}) - \hat{\mathcal{L}}^S(\mathbf{w}_\star)}_{\text{suboptimality from algorithm}} + \hat{\mathcal{L}}^S(\mathbf{w}_\star) - \mathcal{L}(\mathbf{w}_\star)$$

This idea is called empirical risk minimization because in statistics the loss function is often called the "risk", and so $\hat{\mathcal{L}}$ is the "empirical risk".

Empirical risk minimization comes with the important caveat that the point $\hat{\mathbf{w}}_\star = \operatorname{argmin} \hat{\mathcal{L}}^S(\mathbf{w})$ may not be very close to the actual minimizer $\mathbf{w}_\star$. As a result, we can end up with a situation in which

$$\hat{\mathcal{L}}^S(\hat{\mathbf{w}}) < \mathcal{L}^S(\mathbf{w}_\star)$$

This is known as *overfitting*. There is an entire field of statistical learning theory dedicated to understand which properties of ℓ^S lead to overfitting and which do not, and there are still many open problems in this area. A similar amount of work has gone into identifying conditions under which there are reasonable surrogate losses ℓ^S that are computationally tractable while also providing sufficiently good approximations to the true loss ℓ. However, for our purposes whenever we perform empirical risk minimization we will simply assume that the surrogate ℓ^S is such that the overfitting issue is not as important as actually solving the optimization problem. Also, we will generally assume that the surrogate ℓ^S is a good enough approximation to the loss we care about ℓ.

We will usually analyze exclusively the surrogate loss $\mathcal{L}^S(\mathbf{w}) = \mathbb{E}_{\mathbf{z}}[\ell^S(\mathbf{w}, \mathbf{z})]$. In order to keep notation simpler, we will drop the S and simply refer to this as the loss $\mathcal{L}$ and ℓ. If we ever actually need the true losses, it will be made clear on a case-by-case basis.

There is second way of dealing with the problem of unknown $\mathcal{L}^S$ that becomes viable when the size N of the dataset is extremely large, as may happen in some modern language modeling tasks for which data is scraped en masse off of the internet. In this case, N might be so large that it is computationally infeasible to look at any one datapoint $\mathbf{z}_i$ more than once. In this case we will be able to design algorithms that provably cannot overfit.

Assuming the overfitting problem is negligible, we are left with the managing the quality of the surrogate loss. Remember that we want to choose the surrogate loss ℓ^S such that $\mathcal{L}(\mathbf{w}) = \mathbb{E}_{\mathbf{z}}[\ell(\mathbf{w}, \mathbf{z})] \approx \mathbb{E}_{\mathbf{z}}[\ell^S(\mathbf{w}, \mathbf{z})] = \mathcal{L}^S(\mathbf{w})$ and also such that we are able to design algorithms that minimize $\mathcal{L}^S$ effectively. A common strategy to ensure that $\mathcal{L}^S$ is minimizable easily is to make $\ell(\mathbf{w}, \mathbf{z})$ a *convex* function of $\mathbf{w}$ for all $\mathbf{z}$. A convex function is one for which the *epigraph*:

$$\{(\mathbf{w}, k) \in \mathbb{R}^{d+1} : k \geq \ell(\mathbf{w}, \mathbf{z})\}$$

is a convex set. Algebraically, when ℓ is differentiable, this implies the inequality for all $\mathbf{w}, \mathbf{w}'$

$$\ell(\mathbf{w}', \mathbf{z}) \geq \ell(\mathbf{w}, \mathbf{z}) + \langle \nabla \ell(\mathbf{w}, \mathbf{z}), \mathbf{w}' - \mathbf{w} \rangle$$

In words, the value of the first-order Taylor approximation to ℓ at any point $\mathbf{w}$ is smaller than the true function value at all other points. This means that *local* information at a point $\mathbf{w}$ in terms of the gradient $\nabla\ell(\mathbf{w}, \mathbf{z})$ actually provides a *global* bound on the value of $\ell(\mathbf{w}, \mathbf{z})$ at every other point. This property has engendered a vast body of literature on algorithms for optimizing such losses.

Unfortunately, essentially all of the popular deep learning models used to obtain impressive results today do not appear to allow for surrogates ℓ^S that are both good approximations of the true loss ℓ and also convex. Thus, we will focus much of our study on what can be done without using convexity.

2 Gradient Descent for Convex Losses

The most commonly used algorithm in machine learning today is (stochastic) gradient descent. Let's consider this algorithm first in the *deterministic* setting to gain some idea for how it works.

Algorithm 1 Gradient Descent

Input: Initial Point $\mathbf{w}_1$, learning rate η, time horizon T:
for $t = 1 \dots T$ **do**
  Set $\mathbf{g}_t = \nabla \mathcal{L}(\mathbf{w}_t)$.
  Set $\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \mathbf{g}_t$.
end for

Recall the definition of a convex function:

Definition 2.1. *A function $\mathcal{L}$ is* convex *if for all x, there exists some g_x such that for all y:*

$$\mathcal{L}(y) \geq \mathcal{L}(x) + \langle g_x, y - x \rangle$$

When $\mathcal{L}$ is differentiable, $g_x = \nabla \mathcal{L}(x)$.

The definition of convexity has an interesting consequence: if $\mathbf{w}_\star = \operatorname{argmin} \mathcal{L}(\mathbf{w})$, then we have:

$$\mathcal{L}(\mathbf{w}) - \mathcal{L}(\mathbf{w}_\star) \leq \langle \nabla \mathcal{L}(\mathbf{w}), \mathbf{w} - \mathbf{w}_\star \rangle$$

So, if we can understand the *linear* functions $\langle \nabla \mathcal{L}(\mathbf{w}), \mathbf{w} - \mathbf{w}_\star \rangle$, then we will be able to bound the non-linear suboptimality gap $\mathcal{L}(\mathbf{w}) - \mathcal{L}(\mathbf{w}_\star)$. In order to do this, we'll need to assume that our loss is *Lipschitz*:

Definition 2.2. *A function $\mathcal{L} : \mathbb{R}^d \to \mathbb{R}$ is G-Lipschitz if for all $x, y \in \mathbb{R}^d$,*

$$|\mathcal{L}(x) - \mathcal{L}(y)| \leq G \|x - y\|$$

This definition may be a little strange looking, but it's actually fairly likely to hold. The intuition here is that the G in G-Lipschitzness is actually measuring the degree to which $\mathcal{L}$ is continuous. In fact, if W is a compact set, then any continuous $\mathcal{L}$ must be G-Lipschitz for some G. When $\mathcal{L}$ is differentiable, we can phrase G-Lipschitzness in the following way:

Proposition 2.3. *If $\mathcal{L}$ is differentiable, then $\mathcal{L}$ is G-Lipschitz if and only if $\|\nabla \mathcal{L}(x)\| \leq G$ for all x.*

Proof. First, suppose $\|\nabla \mathcal{L}(x)\| > G$ for some $x \in \mathbb{R}^d$. By definition of gradient for any vector v, we have:

$$\lim_{\delta \to 0} \frac{\mathcal{L}(x + \delta v) - \mathcal{L}(x) - \delta \langle v, \nabla \mathcal{L}(x) \rangle}{\delta} = 0$$

Let $v = \frac{\nabla \mathcal{L}(x)}{\|\nabla \mathcal{L}(x)\|}$. Then this implies:

$$\lim_{\delta \to 0} \frac{\mathcal{L}(x + \delta v) - \mathcal{L}(x)}{\delta} = \|\nabla \mathcal{L}(x)\|$$

Thus, if $\|\nabla \mathcal{L}(x)\| > G$, there must be some δ such that

$$|\mathcal{L}(x + \delta v) - \mathcal{L}(x)| > G\delta$$

and so $\mathcal{L}$ is not G-Lipschitz. Therefore G-Lipschitzness implies $\|\nabla \mathcal{L}(x)\| \leq G$. Now suppose $\|\nabla \mathcal{L}(x)\| \leq G$ for all x. Then by the fundamental theorem of calculus,

$$\begin{aligned}
\mathcal{L}(x) - \mathcal{L}(y) &= \int_0^1 \frac{d}{dt} \mathcal{L}(y + t(x - y))\, dt \\
&= \int_0^1 \langle \nabla \mathcal{L}(y + t(x - y)), x - y \rangle\, dt \\
&\leq \int_0^1 \|\nabla \mathcal{L}(y + t(x - y))\| \|x - y\|\, dt \\
&\leq G \|x - y\|
\end{aligned}$$

so that $\mathcal{L}$ is G-Lipschitz. □

Here is the standard convergence result for gradient descent:

Theorem 2.4. *Suppose $\mathcal{L}$ is G-Lipschitz and convex. Suppose $\|\mathbf{w}_\star - \mathbf{w}_1\| \leq D$. Set $\eta = \frac{D}{G\sqrt{T}}$. Then*

$$\sum_{t=1}^{T} \mathcal{L}(\mathbf{w}_t) - \mathcal{L}(\mathbf{w}_\star) \leq GD\sqrt{T}$$

In particular, if $\hat{\mathbf{w}}$ is selected uniformly at random from $\mathbf{w}_1, \ldots, \mathbf{w}_T$,

$$\mathbb{E}[\mathcal{L}(\hat{\mathbf{w}}) - \mathcal{L}(\mathbf{w}_t)] \leq \frac{GD}{\sqrt{T}}$$

Proof. The proof is quite short, although the technique may seem a bit magical:

$$\begin{aligned}\|\mathbf{w}_{t+1} - \mathbf{w}_\star\|^2 &= \|\mathbf{w}_t - \eta\nabla\mathcal{L}(\mathbf{w}_t) - \mathbf{w}_\star\|^2 \\ &= \|\mathbf{w}_t - \mathbf{w}_\star\|^2 - 2\eta\langle\nabla\mathcal{L}(\mathbf{w}_t), \mathbf{w}_t - \mathbf{w}_\star\rangle + \eta^2\|\nabla\mathcal{L}(\mathbf{w}_t)\|^2\end{aligned}$$

rearranging terms:

$$\langle\nabla\mathcal{L}(\mathbf{w}_t), \mathbf{w}_t - \mathbf{w}_\star\rangle \leq \frac{\|\mathbf{w}_t - \mathbf{w}_\star\|^2 - \|\mathbf{w}_{t+1} - \mathbf{w}_\star\|^2}{2\eta} + \frac{\eta\|\nabla\mathcal{L}(\mathbf{w}_t)\|^2}{2}$$

using convexity:

$$\mathcal{L}(\mathbf{w}_t) - \mathcal{L}(\mathbf{w}_\star) \leq \frac{\|\mathbf{w}_t - \mathbf{w}_\star\|^2 - \|\mathbf{w}_{t+1} - \mathbf{w}_\star\|^2}{2\eta} + \frac{\eta\|\nabla\mathcal{L}(\mathbf{w}_t)\|^2}{2}$$

Use G-Lipschitzness:

$$\leq \frac{\|\mathbf{w}_t - \mathbf{w}_\star\|^2 - \|\mathbf{w}_{t+1} - \mathbf{w}_\star\|^2}{2\eta} + \frac{\eta G^2}{2}$$

Now, sum over all t:

$$\sum_{t=1}^{T} \mathcal{L}(\mathbf{w}_t) - \mathcal{L}(\mathbf{w}_\star) \leq \sum_{t=1}^{T} \frac{\|\mathbf{w}_t - \mathbf{w}_\star\|^2 - \|\mathbf{w}_{t+1} - \mathbf{w}_\star\|^2}{2\eta} + \frac{\eta G^2}{2}$$

The sum telescopes:

$$\begin{aligned}&= \frac{\|\mathbf{w}_1 - \mathbf{w}_\star\|^2 - \|\mathbf{w}_{T+1} - \mathbf{w}_\star\|^2}{2\eta} + \frac{\eta T G^2}{2} \\ &\leq \frac{D^2}{2\eta} + \frac{\eta T G^2}{2}\end{aligned}$$

Now, use our setting for η:

$$= DG\sqrt{T}$$

Now, if $\hat{\mathbf{w}}$ is chosen uniformly at random from $\mathbf{w}_1, \ldots, \mathbf{w}_T$, then by definition:

$$\mathbb{E}[\mathcal{L}(\hat{\mathbf{w}}) - \mathcal{L}(\mathbf{w}_\star)] = \frac{1}{T}\sum_{t=1}^{T} \mathcal{L}(\mathbf{w}_t) - \mathcal{L}(\mathbf{w}_\star) \leq \frac{GD}{\sqrt{T}}$$

□

This result has a few lessons for us:

1. This is an *upper bound*. This means that the actual performance of the algorithm might be much better than we have shown here. In practice, this is often true.
2. We only make a statement about a randomly chosen iterate, rather than the last iterate. This is very common in analysis of optimization methods (at least in the stochastic case). In practice however, it is common to simply take $\hat{\mathbf{w}} = \mathbf{w}_T$.
3. Most of the analysis is done without using the value of η - instead it is only substituted at the end. You can see from the analysis that we chose $\eta = \frac{D}{G\sqrt{T}}$ specifically to minimize the final expression. Other choices would still converge.
4. The learning rate is $O(1/\sqrt{T})$. This will show up in the non-convex case as well.

3 Stochastic Convex Losses

Now that we've seen how to analyze gradient descent for deterministic convex losses, let's consider the more practical *stochastic* case. Remember in the stochastic setting we assume:

$$\mathcal{L}(\mathbf{w}) = \mathop{\mathbb{E}}_{z \sim P_z}[\ell(\mathbf{w}, z)]$$

Let's assume that $\ell(\mathbf{w}, z)$ is differentiable as a function of $\mathbf{w}$ for all z. Then, we have the following:

Proposition 3.1. *If $\mathcal{L}(\mathbf{w}) : \mathbb{R}^d \to \mathbb{R}$ is given by $\mathcal{L}(\mathbf{w}) = \mathbb{E}_{z \sim P_z}[\ell(\mathbf{w}, z)]$ and $\ell(\mathbf{w}, z)$ is differentiable as a function of $\mathbf{w}$ for all z, then:*

$$\nabla \mathcal{L}(\mathbf{w}) = \mathop{\mathbb{E}}_{z}[\nabla \ell(\mathbf{w}, z)]$$

Proof. Define $p(z)$ to be the probability density function of z. Then we have:

$$\begin{aligned}\nabla \mathcal{L}(\mathbf{w}) &= \nabla \, \mathbb{E}[\ell(\mathbf{w}, z)] \\ &= \nabla \int \ell(\mathbf{w}, z) p(z) dz\end{aligned}$$

interchanging integration and differentation:

$$\begin{aligned}&= \int \nabla \ell(\mathbf{w}, z) p(z) dz \\ &= \mathbb{E}[\nabla \ell(\mathbf{w}, z)]\end{aligned}$$

For the very technical minded reader, this proof has a couple holes: we assumed z has a density $p(z)$, and we assumed that it is indeed possible to exchange integration and differentiation. Both of these are not fatal issues: instead of $p(z)dz$, we may integrate with respect to the probability measure $d\mu(z)$ if there is not density. For the interchange of integration and differentiation, we can overcome this by adding the assumption that for any $\mathbf{w}$ $\nabla \ell(\mathbf{w}, z)$ is bounded on some closed ball centered at $\mathbf{w}$. This assumption is true for all functions you are likely to ever encounter in practice. □

This Lemma gives us a way to provide an *unbiased estimate* of the gradient $\nabla \mathcal{L}(\mathbf{w})$ at any point $\mathbf{w}$: simply sample some $z \sim P_z$ and return $\nabla \ell(\mathbf{w}, z)$. An *unbaised estimator* for a quantity X is a random variable Y such that $\mathbb{E}[Y] = X$. Unbiased estimators are useful because if you can generate many independent unbiased estimates $Y_1, \ldots, Y_N$, then their average $\frac{1}{N}\sum_{i=1}^{N} Y_i$ will converge to X as N grows.

So, our stochastic gradient descent algorithm will be essentially the same as regular gradient descent, but we will substitute an unbiased gradient estimate instead of the true gradient (Algorithm 2).

Let's analyze the convergence of this algorithm under the assumption that $\mathcal{L}$ is convex:

Algorithm 2 Stochastic Gradient Descent

Input: Initial Point $\mathbf{w}_1$, learning rate η, time horizon T:
for $t = 1 \dots T$ **do**
  Sample $z_t \sim P_z$
  Set $\mathbf{g}_t = \nabla \ell(\mathbf{w}_t, z_t)$.
  Set $\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \mathbf{g}_t$.
end for

Theorem 3.2. *Suppose that $\mathcal{L}$ is convex and that $\ell(\mathbf{w}, z)$ is G-Lipschitz. Suppose $\|\mathbf{w}_1 - \mathbf{w}_\star\| \leq D$. Then with $\eta = \frac{D}{G\sqrt{T}}$,*

$$\mathbb{E}\left[\sum_{t=1}^{T} \mathcal{L}(\mathbf{w}_t) - \mathcal{L}(\mathbf{w}_\star)\right] \leq DG\sqrt{T}$$

If $\hat{\mathbf{w}}$ is selected uniformly at random from $\mathbf{w}_1, \dots, \mathbf{w}_T$,

$$\mathbb{E}[\mathcal{L}(\hat{\mathbf{w}}) - \mathcal{L}(\mathbf{w}_\star)] \leq \frac{DG}{\sqrt{T}}$$

Proof. The proof is actually very similar to the proof for the deterministic case, but we put expectations around many quantities and use the unbiasedness property $\mathbb{E}[\nabla \ell(\mathbf{w}_t, z)|\mathbf{w}_t] = \nabla \mathcal{L}(\mathbf{w}_t)$. All expectations presented here are over the randomness of the algorithms (i.e. over the choices $z_1, \dots, z_T$) unless otherwise specified:

$$\begin{aligned}\mathbb{E}[\|\mathbf{w}_{t+1} - \mathbf{w}_\star\|^2] &= \mathbb{E}[\|\mathbf{w}_t - \eta \mathbf{g}_t - \mathbf{w}_\star\|^2] \\ &= \mathbb{E}[\|\mathbf{w}_t - \mathbf{w}_\star\|^2 - 2\eta \langle \mathbf{g}_t, \mathbf{w}_t - \mathbf{w}_\star \rangle + \eta^2 \|\mathbf{g}_t\|^2]\end{aligned}$$

rearranging:

$$\mathbb{E}[\langle \mathbf{g}_t, \mathbf{w}_t - \mathbf{w}_\star \rangle] = \frac{\mathbb{E}[\|\mathbf{w}_t - \mathbf{w}_\star\|^2 - \|\mathbf{w}_{t+1} - \mathbf{w}_\star\|^2]}{2\eta} + \frac{\eta\, \mathbb{E}[\|\mathbf{g}_t\|^2]}{2}$$

Now, from unbiasedness and convexity, we have:

$$\mathbb{E}[\langle \mathbf{g}_t, \mathbf{w}_t - \mathbf{w}_\star \rangle] = \mathop{\mathbb{E}}_{z_t, \dots, z_{t-1}}[\mathbb{E}[\langle \mathbf{g}_t, \mathbf{w}_t - \mathbf{w}_\star \rangle | z_1, \dots, z_{t-1}]]$$

now, observe that w_t is a deterministic function of $z_1, \dots, z_{t-1}$, and that $\mathbf{g}_t$ is independent of $z_1, \dots, z_{t-1}$ given w_t:

$$\begin{aligned}&= \mathop{\mathbb{E}}_{z_t, \dots, z_{t-1}}[\mathbb{E}[\langle \mathbf{g}_t, \mathbf{w}_t - \mathbf{w}_\star \rangle | w_t]] \\ &= \mathop{\mathbb{E}}_{z_1, \dots, z_{t-1}}[\langle \nabla \mathcal{L}(\mathbf{w}_t), \mathbf{w}_t - \mathbf{w}_\star \rangle]\end{aligned}$$

Further, $\nabla \mathcal{L}(w_t)$ and w_t are independent of $z_t, \dots, z_T$:

$$\begin{aligned}&= \mathbb{E}[\langle \nabla \mathcal{L}(\mathbf{w}_t), \mathbf{w}_t - \mathbf{w}_\star \rangle] \\ &\geq \mathbb{E}[\mathcal{L}(\mathbf{w}_t) - \mathcal{L}(\mathbf{w}_\star)]\end{aligned}$$

So, plugging this in:

$$\mathbb{E}[\mathcal{L}(\mathbf{w}_t) - \mathcal{L}(\mathbf{w}_\star)] \leq \frac{\mathbb{E}[\|\mathbf{w}_t - \mathbf{w}_\star\|^2 - \|\mathbf{w}_{t+1} - \mathbf{w}_\star\|^2]}{2\eta} + \frac{\eta\, \mathbb{E}[\|\mathbf{g}_t\|^2]}{2}$$

$$\mathbb{E}\left[\sum_{t=1}^{T} \mathcal{L}(\mathbf{w}_t) - \mathcal{L}(\mathbf{w}_\star)\right] \leq \sum_{t=1}^{T} \frac{\mathbb{E}[\|\mathbf{w}_t - \mathbf{w}_\star\|^2 - \|\mathbf{w}_{t+1} - \mathbf{w}_\star\|^2]}{2\eta} + \frac{\eta\, \mathbb{E}[\|\mathbf{g}_t\|^2]}{2}$$

Telescoping the first sum, and using $\|\mathbf{g}_t\| \leq G$:

$$\begin{aligned} &\leq \frac{\mathbb{E}[\|\mathbf{w}_1 - \mathbf{w}_\star\|^2}{2\eta} + \frac{\eta T G^2}{2} \\ &\leq DG\sqrt{T} \end{aligned}$$

Now, for the last statement in the Theorem, again use the fact that

$$\mathbb{E}[\mathcal{L}(\hat{\mathbf{w}}) - \mathcal{L}(\mathbf{w}_\star)] = \mathbb{E}\left[\frac{1}{T}\sum_{t=1}^{T} \mathcal{L}(\mathbf{w}_t) - \mathcal{L}(\mathbf{w}_\star)\right]$$

□

There are a couple interesting points about this proof:

1. The result appears essentially the same as in the deterministic case.
2. The expectation in the final statement is now over both the randomness in the choice of $\hat{\mathbf{w}}$ and also the randomness in the z_ts.
3. The G here is a bound on $\|\nabla \ell(\mathbf{w}, z)\|$, which could be much larger than the G in Theorem 2.4, which is only a bound on $\|\nabla \mathcal{L}(\mathbf{w})\|$.

Let's compare the *total computation* required by the deterministic and the stochastic versions of gradient descent. Suppose we are performing empirical risk minimization, so that the objective is

$$\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N} \ell(\mathbf{w}, z_i)$$

Then, to compute $\nabla \mathcal{L}(\mathbf{w}) = \frac{1}{N}\sum_{i=1}^{N} \nabla \ell(\mathbf{w}, z_i)$ takes N total gradient computations, so $O(N)$ time.
On the other hand, we can re-write $\mathcal{L}$ as:

$$\mathcal{L} = \mathop{\mathbb{E}}_{z}[\ell(\mathbf{w}, z)]$$

where P_z is the uniform distribution over $z_1, \ldots, z_N$. Then, to the gradient estimate $\mathbf{g}_t$, we choose some z uniformly at random from $z_1, \ldots, z_N$ and compute $\mathbf{g}_t = \nabla \ell(\mathbf{w}, z)$. This takes only one gradient evaluation, so $O(1)$ time.

As a result, after T iterations of gradient descent, we have spent $O(NT)$ gradient computations, but with stochastic gradient descent, we have spent only $O(T)$ gradient computations. Since in both cases the converge rate is only a function of T and not N, This makes it seem like stochastic gradient descent is significantly better than deterministic gradient descent.

However, there is a caveat: the G in Theorem 2.4 is potentially much smaller than the G in Theorem 3.2, so that it is possible that one would be better off with gradient descent. However, in general this is unlikely: G would have to be $\sqrt{N}$ times larger and as we collect more data (so N grows), this becomes less and less plausible.

3.1 Generalization of SGD

Remember that one issue that can come up when using ERM to solve machine learning problems is *overfitting*, in which performance on the empirical risk $\frac{1}{N}\sum_{i=1}^{N} \ell(\hat{\mathbf{w}}, z_i)$ is much better than the true loss $\mathcal{L}(\hat{\mathbf{w}}) = \mathbb{E}[\ell(\hat{\mathbf{w}}, z)]$.

One of the magical properties of SGD is that it can in some sense provably avoid overfitting. The procedure is the following: given an i.i.d. dataset $z_1, \ldots, z_N$, run SGD for N iterations (and no more!) using each datapoint one after the other with no repeating (sample without replacement). Then a random selected iterate will satisfy:

$$\mathbb{E}[\mathcal{L}(\hat{\mathbf{w}}) - \mathcal{L}(\mathbf{w}_\star)] \leq \frac{DG}{\sqrt{N}}$$

Notice that this is a statement about the *true* loss $\mathcal{L}$, not the empirical risk! Not only that, the dependency on N is actually statistically optimal up to constant factors in many cases although this is not so easy to see.

3.2 SGD for ERM

However, in practice it seems that minimizing the empirical risk frequently is superior to a single-pass over the dataset. So how can we use SGD to minimize the empirical risk? We have already discussed how if $\hat{\mathcal{L}}$ is the ERM-loss, $\hat{\mathcal{L}}(\mathbf{w}) = \mathbb{E}_{z\sim\hat{P}}[\ell(\mathbf{w}, z)]$, where $\hat{P}$ is the uniform distribution over $z_1, \ldots, z_N$. So you could just run SGD using this distribution.

In practice, however, it is more common to run the following procedure:

Algorithm 3 Stochastic Gradient Descent for ERM with shuffling

Input: Initial Point $\mathbf{w}_1$, learning rate η, time horizon T, dataset $z_1, \ldots, z_N$:
Initialize $t = 1$.
for $e = 1 \ldots T/N$ **do**
  for $i = 1 \ldots N$ **do**
    Set $\mathbf{g}_t = \nabla\ell(\mathbf{w}_t, z_i)$.
    Set $\mathbf{w}_{t+1} = \mathbf{w}_t - \eta\mathbf{g}_t$.
    $t \leftarrow t + 1$.
  end for
  Optionally shuffle the dataset into a new random order.
end for

This algorithm is typically much better in practice than using with-replacement sampling, but the reasons are still somewhat mysterious. Sampling at random for every iteration is much simpler to analyze, and analyses of the in-order scheme are only recently showing improvements. For recent references, see Nguyen et al. 2020; Mishchenko, Khaled, and Richtárik 2020.

4 Non-Convex Gradient Descent on Smooth Losses

Starting in this section, and for much of this course, we will consider exclusively losses $\mathcal{L}$ that are *smooth*:

Definition 4.1. *A function $\mathcal{L} : \mathbb{R}^d \to \mathbb{R}$ is H-smooth if $\mathcal{L}$ is differentiable at all $x \in \mathbb{R}^d$, and for all $x, y \in W$,*

$$\|\nabla\mathcal{L}(x) - \nabla\mathcal{L}(y)\| \leq H\|x - y\|$$

Let's be clear: there are plenty of losses out there that are not smooth, or are only smooth with a very large value of H. However, without *some* kind of assumption about the loss it is very difficult to prove anything about how an algorithm will operate. Similarly to the way that G-Lipschitzness is the same as the gradient being bounded by G when $\mathcal{L}$ is differentiable, H-smoothness is the same as the Hessian being bounded by H when $\mathcal{L}$ is twice-differentiable:

Proposition 4.2. *Suppose $\mathcal{L}$ is twice differentiable. Then $\mathcal{L}$ is H-smooth if and only if $\|\nabla^2\mathcal{L}(\mathbf{w})\|_{op} \leq H$ for all $\mathbf{w}$.*

Proof. The proof is essentially the same as in the Lipschitz case. Suppose $\mathcal{L}$ is H-smooth. Let x be some arbitrary point and let v be an arbitary unit vector. Define $f : \mathbb{R} \to \mathbb{R}$ by $f(t) = \mathcal{L}(x + tv)$. Then by definition of directional derivatives, $f'(0) = \langle\nabla\mathcal{L}(x + tv), v\rangle$ and $f''(0) = \nabla^2\mathcal{L}(x + tv)v$. Now we have:

$$\begin{aligned}
\|\nabla^2\mathcal{L}(x)v\| &= |f''(0)| \\
&= \left|\lim_{\delta\to 0} \frac{f'(\delta) - f'(0)}{\delta}\right| \\
&= \left|\lim_{\delta\to 0} \frac{\langle\nabla\mathcal{L}(x + \delta v) - \nabla\mathcal{L}(x), v\rangle}{\delta}\right| \\
&\leq \lim_{\delta\to 0} \frac{\|\nabla\mathcal{L}(x + \delta v) - \nabla\mathcal{L}(x)\|\|v\|}{\delta} \\
&\leq H
\end{aligned}$$

Now since x and v were arbitrary, $\|\mathcal{L}(\mathbf{w})\|_{\text{op}} \leq H$ for all $\mathbf{w}$.

On the other hand, suppose $\|\mathcal{L}(\mathbf{w})\|_{\text{op}} \leq H$ for all $\mathbf{w}$. Then by mean-value theorem, for any x and y:

$$\nabla\mathcal{L}(x) = \nabla\mathcal{L}(y) + \nabla^2\mathcal{L}(z)(x - y)$$

for some z on the line segment connecting x and y. Then, since $\|\nabla^2\mathcal{L}(z)\| \leq H$, we have $\|\nabla\mathcal{L}(x) - \nabla\mathcal{L}(y)\| \leq H\|x - y\|$. □

Smoothness is at least approximately satisfied by essentially any continuous function, in the sense that any continuous function can be replaced with an approximation that is smooth. The quality of the approximation can be traded off for the amount of smoothness. Formally, the following theorem (which intuitively captures the idea of "gaussian blur") holds:

Theorem 4.3. *Let $\mathcal{L} : \mathbb{R}^d \to \mathbb{R}$ be a G-Lipschitz function. Then for any $\gamma > 0$, consider*

$$\hat{\mathcal{L}}(x) = \mathop{\mathbb{E}}_{v\sim N(0,I)}[L(x + \gamma v)]$$

where $v \sim N(0, I)$ indicates that v is sampled from Gaussian distribution. $\hat{\mathcal{L}}$ is G-Lipschitz and G/γ-smooth and for all $x \in W$,

$$|\hat{\mathcal{L}}(x) - \mathcal{L}(x)| \leq \gamma\sqrt{d}$$

Because we will not be assuming that $\mathcal{L}$ is convex, in general we will not be able to actually show that our algorithms in fact minimizes $\mathcal{L}$. Instead, we will often simply try to approach a *critical point*:

Definition 4.4. *A* critical point*, or* first-order stationary point *of $\mathcal{L}$ is a point x such that $\nabla\mathcal{L}(x) = 0$.*

The search for critical points is motivated by the observation that any minimizer of $\mathcal{L}$ must also be a critical point, so that finding a critical point is a necessary but not sufficient condition for actually minimizing $\mathcal{L}$. There is some active research investigating the degree to which this condition may actually be sufficient. In fact, one of the key desirable properties of convex functions is that any critical point must also minimize $\mathcal{L}$.

Since finding a point where the gradient is *exactly* zero may be difficult, we will instead relax the goal slightly to find an *approximate* critical point:

Definition 4.5. *A ϵ-approximate critical point of $\mathcal{L}$ is a point x such that $\|\nabla \mathcal{L}(x)\| \leq \epsilon$*

Now let's re-analyze gradient descent assuming smoothness, but not convexity.

Algorithm 2 Gradient Descent

Input: Initial Point $\mathbf{w}_1$, learning rate η, time horizon T:
for $t = 1 \dots T$ **do**
  Set $\mathbf{g}_t = \nabla \mathcal{L}(\mathbf{w}_t)$.
  Set $\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \mathbf{g}_t$.
end for

In order to do this, we need the following critical Lemma about how the gradient influences the local behavior of a smooth function:

Lemma 4.6. *Suppose $\mathcal{L}$ is H-smooth. Then for any $\mathbf{w}$ and $\mathbf{v}$,*

$$\mathcal{L}(\mathbf{w} + \mathbf{v}) \leq \mathcal{L}(\mathbf{w}) + \langle \nabla \mathcal{L}(\mathbf{w}), \mathbf{v} \rangle + \frac{H}{2} \|\mathbf{v}\|^2$$

Proof. Since $\mathcal{L}$ is differentiable (since all smooth functions are differentiable), we can apply the fundamental theorem of calculus to get:

$$\begin{aligned}
\mathcal{L}(\mathbf{w} + \mathbf{v}) &= \mathcal{L}(\mathbf{w}) + \int_0^1 \frac{d}{dt} \mathcal{L}(\mathbf{w} + t\mathbf{v}) \, dt \\
&= \mathcal{L}(\mathbf{w}) + \int_0^1 \langle \nabla \mathcal{L}(\mathbf{w} + t\mathbf{v}), \mathbf{v} \rangle \, dt \\
&= \mathcal{L}(\mathbf{w}) + \int_0^1 \langle \nabla \mathcal{L}(\mathbf{w}), \mathbf{v} \rangle \, dt + \langle \nabla \mathcal{L}(\mathbf{w} + t\mathbf{v}) - \nabla \mathcal{L}(\mathbf{w}), \mathbf{v} \rangle \, dt \\
&= \mathcal{L}(\mathbf{w}) + \langle \nabla \mathcal{L}(\mathbf{w}), \mathbf{v} \rangle + \int_0^1 \langle \nabla \mathcal{L}(\mathbf{w} + t\mathbf{v}) - \nabla \mathcal{L}(\mathbf{w}), \mathbf{v} \rangle \, dt \\
&\leq \mathcal{L}(\mathbf{w}) + \langle \nabla \mathcal{L}(\mathbf{w}), \mathbf{v} \rangle + \int_0^1 \|\nabla \mathcal{L}(\mathbf{w} + t\mathbf{v}) - \nabla \mathcal{L}(\mathbf{w})\| \|\mathbf{v}\| \, dt
\end{aligned}$$

Now use the fact that $\mathcal{L}$ is H-smooth:

$$\begin{aligned}
&\leq \mathcal{L}(\mathbf{w}) + \langle \nabla \mathcal{L}(\mathbf{w}), \mathbf{v} \rangle + \int_0^1 H \|t\mathbf{v}\| \|\mathbf{v}\| \, dt \\
&\leq \mathcal{L}(\mathbf{w}) + \langle \nabla \mathcal{L}(\mathbf{w}), \mathbf{v} \rangle + H \|\mathbf{v}\|^2 \int_0^1 t \, dt \\
&\leq \mathcal{L}(\mathbf{w}) + \langle \nabla \mathcal{L}(\mathbf{w}), \mathbf{v} \rangle + \frac{H}{2} \|\mathbf{v}\|^2
\end{aligned}$$

□

This Lemma is a foundational result that enables almost all analysis of gradient-descent based algorithms today. Let us see it in action by using it to understand what happens in one step of gradient descent:

Lemma 4.7. *Suppose $\mathcal{L}$ is H-smooth. Then every step of gradient descent (Algorithm 1) satisfies:*

$$\mathcal{L}(\mathbf{w}_{t+1}) \leq \mathcal{L}(\mathbf{w}_t) - \left(1 - \frac{H\eta}{2}\right)\eta\|\nabla\mathcal{L}(\mathbf{w}_t)\|^2$$

Moreover, if $0 \leq \eta \leq \frac{1}{H}$,

$$\mathcal{L}(\mathbf{w}_{t+1}) \leq \mathcal{L}(\mathbf{w}_t) - \frac{\eta}{2}\|\nabla\mathcal{L}(\mathbf{w}_t)\|^2$$

Proof. The second statement of the Lemma follows immediately from the first statement, so let's just prove the first statement. This turns out to be a nearly immediate consequence of the key Lemma 4.6:

$$\mathcal{L}(\mathbf{w}_{t+1}) \leq \mathcal{L}(\mathbf{w}_t) + \langle\nabla\mathcal{L}(\mathbf{w}_t), \mathbf{w}_{t+1} - \mathbf{w}_t\rangle + \frac{H}{2}\|\mathbf{w}_{t+1} - \mathbf{w}_t\|^2$$

now use the fact that $\mathbf{w}_{t+1} - \mathbf{w}_t = -\eta\nabla\mathcal{L}(\mathbf{w}_t)$:

$$\begin{aligned}
&= \mathcal{L}(\mathbf{w}_t) - \eta\|\nabla\mathcal{L}(\mathbf{w}_t)\|^2 + \frac{H}{2}\eta^2\|\nabla\mathcal{L}(\mathbf{w}_t)\|^2 \\
&= \mathcal{L}(\mathbf{w}_t) - \left(1 - \frac{H\eta}{2}\right)\eta\|\nabla\mathcal{L}(\mathbf{x}_t)\|^2
\end{aligned}$$

□

This Lemma tells us that the *function progress* of gradient descent is related to the *gradient magnitude*. Instead of being directly proportional, it is proportional to the *square* of the gradient magnitude. This is because there are two compounding effects that make function progress slow down when the gradient is small: first, a small gradient means that $\mathbf{w}_{t+1}$ is closer to $\mathbf{w}_t$. Second, a small gradient means that the change in function value produced by a change in input $\mathbf{w}$ is small.

So, let's use this knowledge to show that gradient descent will find an approximate critical point, and quantify the number of gradient computations required to do so:

Theorem 4.8. *Suppose $\mathcal{L}$ is H-smooth, and $\eta \leq \frac{1}{H}$. Define $\Delta = \mathcal{L}(\mathbf{w}_1) - \inf_{\mathbf{w}}\mathcal{L}(\mathbf{w})$. Then gradient descent guarantees:*

$$\sum_{t=1}^{T}\|\nabla\mathcal{L}(\mathbf{w}_t)\|^2 \leq \frac{2\Delta}{\eta}$$

In particular, if $\eta = \frac{1}{H}$ and $\hat{\mathbf{w}}$ is selected from $\mathbf{w}_1, \ldots, \mathbf{w}_T$ uniformly at random, we have:

$$\mathbb{E}[\|\nabla\mathcal{L}(\hat{\mathbf{w}})\|] \leq \frac{\sqrt{2H\Delta}}{\sqrt{T}}$$

Proof. By Lemma 4.7, we have:

$$\begin{aligned}
\mathcal{L}(\mathbf{w}_{t+1}) &\leq \mathcal{L}(\mathbf{w}_t) - \frac{\eta}{2}\|\nabla\mathcal{L}(\mathbf{w}_t)\|^2 \\
\mathcal{L}(\mathbf{w}_{t+1}) - \mathcal{L}(\mathbf{w}_t) &\leq -\frac{\eta}{2}\|\nabla\mathcal{L}(\mathbf{w}_t)\|^2
\end{aligned}$$

now, sum over t from 1 to T. Notice that we never compute $\mathbf{w}_{T+1}$, but it still exists for analysis:

$$\sum_{t=1}^{T}\mathcal{L}(\mathbf{w}_{t+1}) - \mathcal{L}(\mathbf{w}_t) \leq -\sum_{t=1}^{T}\frac{\eta}{2}\|\nabla\mathcal{L}(\mathbf{w}_t)\|^2$$

The left hand side is a telescoping sum:

$$\mathcal{L}(\mathbf{w}_{T+1}) - \mathcal{L}(\mathbf{w}_1) \leq -\sum_{t=1}^{T} \frac{\eta}{2} \|\nabla \mathcal{L}(\mathbf{w}_t)\|^2$$

$$\sum_{t=1}^{T} \|\nabla \mathcal{L}(\mathbf{w}_t)\|^2 \leq \frac{2}{\eta} \left(\mathcal{L}(\mathbf{w}_1) - \mathcal{L}(\mathbf{w}_{T+1})\right)$$

$$\sum_{t=1}^{T} \|\nabla \mathcal{L}(\mathbf{w}_t)\|^2 \leq \frac{2\Delta}{\eta}$$

Now for the second part of the Theorem, notice that $\hat{\mathbf{w}}$ is chosen at randomfrom $\mathbf{w}_1, \ldots, \mathbf{w}_T$, then:

$$\mathbb{E}[\|\nabla \mathcal{L}(\hat{\mathbf{w}})\|^2] = \frac{1}{T} \sum_{t=1}^{T} \|\nabla \mathcal{L}(\mathbf{w}_t)\|^2 \leq \frac{2\Delta}{\eta T}$$

Next, by Jensen inequality,

$$\mathbb{E}[\|\nabla \mathcal{L}(\hat{\mathbf{w}})\|] \leq \sqrt{\mathbb{E}[\|\nabla \mathcal{L}(\hat{\mathbf{w}})\|^2]}$$

so that plugging in the value $\eta = \frac{1}{H}$ finishes the proof. □

4.1 How to Invent Gradient Descent

Although the preceeding analysis shows that gradient descent has some favorable properties, one might wonder why we consider this algorithm at all. Is it just based on some weird physical intuition? There isn't actually any physics happening here, so there's no particular reason to believe that physical intuition is actually useful - maybe we can actually make algorithms that perform better than gradient descent! It turns out that often one *can* indeed do better than gradient descent, and we will look a bit at such algorithms later, but regardless there is a little more reason in the design of the gradient descent algorithm than simply pulling it out of a hat.

Let's try to design an iterative algorithm for optimizing $\mathcal{L}$ from first principles. One desirable property might be that we want the function to continuously decrease as the algorithm progresses. That is, $\mathcal{L}(\mathbf{w}_{t+1}) \leq \mathcal{L}(\mathbf{w}_t)$. Further, we want the decrease to happen as fast as possible, so that $\mathcal{L}(\mathbf{w}_t) - \mathcal{L}(\mathbf{w}_{t+1})$ is as large as possible.

To do this, we'll consider a *local upper bound* on $\mathcal{L}$. That is, given $\mathbf{w}_t$, we will define a new function $U_{\mathbf{w}_t}$ such that:

1. $U_{\mathbf{w}_t}(\mathbf{w}_t) = \mathcal{L}(\mathbf{w}_t)$.
2. $U_{\mathbf{w}_t}(\mathbf{w}) \geq \mathcal{L}(\mathbf{w})$ for all $\mathbf{w}$.
3. $U_{\mathbf{w}_t}$ is "simple" in the sense that we can easily compute the minimizer $\operatorname{argmin}_{\mathbf{w}} U_{\mathbf{w}_t}(\mathbf{w})$ in closed form.

The idea is that $U_{\mathbf{w}_t}$ should somehow encapsulate the information available to the algorithm about $\mathcal{L}$ near the point $\mathbf{w}_t$.

Given such an upper bound, our algorithm will be:

1. Form $U_{\mathbf{w}_t}$.
2. Set $\mathbf{w}_{t+1} = \operatorname{argmin} U_{\mathbf{w}_t}(\mathbf{w})$.

This algorithm is guaranteed to have $\mathcal{L}(\mathbf{w}_{t+1}) \leq \mathcal{L}(\mathbf{w}_t)$ because:

$$\mathcal{L}(\mathbf{w}_{t+1}) \leq U_{\mathbf{w}_t}(\mathbf{w}_{t+1}) \leq U_{\mathbf{w}_t}(\mathbf{w}_t) = \mathcal{L}(\mathbf{w}_t)$$

Furthermore, the "gap" between $\mathcal{L}(\mathbf{w}_{t+1})$ and $\mathcal{L}(\mathbf{w}_t)$ can be lower bounded by $U_{\mathbf{w}_t}(\mathbf{w}_t) - U_{\mathbf{w}_t}(\mathbf{w}_{t+1})$. Thus, by setting $\mathbf{w}_{t+1} = \operatorname{argmin} U_{\mathbf{w}_t}$, we are maximizing this gap. It only remains to find a reasonable candidate for $U_{\mathbf{w}_t}$. To this end, we again apply the key Lemma 4.6:

Proposition 4.9. *Suppose $\mathcal{L}$ is H-smooth. For any $\mathbf{x}$, define $U_{\mathbf{x}}(\mathbf{y}) = \mathcal{L}(\mathbf{x}) + \langle \nabla \mathcal{L}(\mathbf{x}), \mathbf{y} - \mathbf{x} \rangle + \frac{H}{2}\|\mathbf{x} - \mathbf{y}\|^2$. Then for all $\mathbf{y}$:*

$$\mathcal{L}(\mathbf{x}) = U_{\mathbf{x}}(\mathbf{x})$$
$$\mathcal{L}(\mathbf{y}) \leq U_{\mathbf{x}}(\mathbf{y})$$

Furthermore,

$$\operatorname*{argmin}_{\mathbf{y}} U_{\mathbf{x}}(\mathbf{y}) = \mathbf{x} - \frac{\nabla \mathcal{L}(\mathbf{x})}{H}$$

Proof. First, it is clear from the definition that $\mathcal{L}(\mathbf{x}) = U_{\mathbf{x}}(\mathbf{x})$. Next, the statement $\mathcal{L}(\mathbf{y}) \leq U_{\mathbf{x}}(\mathbf{y})$ follows directly from Lemma 4.6. Finally, differentiating with respect to $\mathbf{y}$ shows that to compute the argmin we need:

$$0 = \nabla \mathcal{L}(\mathbf{x}) - H(\mathbf{x} - \mathbf{y})$$

so solving for $\mathbf{y}$ yields the last part of the result. □

Notice that this Proposition actually recovers the gradient descent algorithm, and automatically is using the best learning rate! This strategy of defining a bounding function U is a common theme in optimization of which gradient descent is simply the first example. This should immediately suggest a way to perhaps improve the algorithm: if we could come up with a better bound U, maybe we could make a better algorithm.

5 Non-Convex Stochastic Gradient Descent and Decreasing Learning Rates

Previously, we showed that gradient descent can find critical points of smooth functions in the deterministic setting. Now, we'll consider the stochastic setting. Let's recall the SGD procedure.

Algorithm 3 Stochastic Gradient Descent

Input: Initial Point $\mathbf{w}_1$, learning rate η, time horizon T:
for $t = 1 \dots T$ **do**
  Sample $z_t \sim P_z$
  Set $\mathbf{g}_t = \nabla \ell(\mathbf{w}_t, z_t)$.
  Set $\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \mathbf{g}_t$.
end for

The classic analysis of stochastic gradient descent is as follows:

Theorem 5.1. *Suppose $\mathcal{L}$ is H-smooth, and $\nabla \ell(\mathbf{w}, z)$ is G-Lipschitz. Let us consider SGD with a* fixed *learning rate $\eta_t = \eta$ for all t. Then Algorithm 2 guarantees:*

$$\sum_{t=1}^{T} \mathbb{E}[\|\nabla \mathcal{L}(\mathbf{w}_t)\|^2] \leq \frac{\Delta}{\eta} + \frac{HT\eta G^2}{2}$$

Further, if we set $\eta = \frac{\sqrt{2\Delta}}{G\sqrt{HT}}$, then:

$$\sum_{t=1}^{T} \mathbb{E}[\|\nabla \mathcal{L}(\mathbf{w}_t)\|^2] \leq G\sqrt{2\Delta HT}$$

, then if $\hat{\mathbf{w}}$ is selected uniformly at random from $\mathbf{w}_1, \dots, \mathbf{w}_T$,

$$\mathbb{E}[\|\nabla \mathcal{L}(\hat{\mathbf{w}})\|] \leq \frac{\sqrt{G\sqrt{2\Delta H}}}{T^{1/4}}$$

In comparison to the rate for *deterministic* problems, we have moved from $O(1/\sqrt{T})$ to $O(1/T^{1/4})$, so that in the non-convex setting there is a significant price to be paid for stochasticity.

Proof. From the main lemma on smooth losses:

$$\begin{aligned}
\mathcal{L}(\mathbf{w}_{t+1}) &\leq \mathcal{L}(\mathbf{w}_t) + \langle \nabla \mathcal{L}(\mathbf{w}_t), \mathbf{w}_{t+1} - \mathbf{w}_t \rangle + \frac{H}{2}\|\mathbf{w}_{t+1} - \mathbf{w}_t\|^2 \\
&= \mathcal{L}(\mathbf{w}_t) - \eta \langle \nabla \mathcal{L}(\mathbf{w}_t), \mathbf{g}_t \rangle + \frac{H\eta^2 \|\mathbf{g}_t\|^2}{2} \\
&\leq \mathcal{L}(\mathbf{w}_t) - \eta \langle \nabla \mathcal{L}(\mathbf{w}_t), \mathbf{g}_t \rangle + \frac{H\eta^2 G^2}{2} \\
\mathbb{E}[\mathcal{L}(\mathbf{w}_{t+1})] &\leq \mathbb{E}[\mathcal{L}(\mathbf{w}_t) - \eta \|\nabla \mathcal{L}(\mathbf{w}_t)\|^2 + \frac{H\eta^2 G^2}{2}]
\end{aligned}$$

sum over t and rearrange:

$$\sum_{t=1}^{T} \mathbb{E}[\|\nabla \mathcal{L}(\mathbf{w}_t)\|^2] \leq \frac{\Delta}{\eta} + \frac{H\eta T G^2}{2}$$

This shows the first part of the theorem. The second parts follow from plugging in the given value for η and Jensen inequality. □

Theorem 5.2. *Suppose $\mathcal{L}$ is H-smooth, and $\nabla \ell(\mathbf{w}, z)$ has variance at most σ^2 for all $\mathbf{w}$ (that is for all $\mathbf{w}$, $\mathbb{E}_z[\|\nabla \ell(\mathbf{w}, z) - \nabla \mathcal{L}(\mathbf{w})\|^2] \leq \sigma^2$). Let us consider SGD with a* fixed *learning rate $\eta_t = \eta$ for all t. Then so long as $\eta \leq \frac{1}{H}$, Algorithm 2 guarantees:*

$$\sum_{t=1}^{T} \mathbb{E}[\|\nabla \mathcal{L}(\mathbf{w}_t)\|^2] \leq \frac{2\Delta}{\eta} + HT\eta\sigma^2$$

Further, if we set $\eta = \min\left(\frac{1}{H}, \frac{\sqrt{\Delta}}{\sigma\sqrt{HT}}\right)$, then:

$$\sum_{t=1}^{T} \mathbb{E}[\|\nabla \mathcal{L}(\mathbf{w}_t)\|^2] \leq 2\Delta H + 3\sigma\sqrt{\Delta HT}$$

, then if $\hat{\mathbf{w}}$ is selected uniformly at random from $\mathbf{w}_1, \ldots, \mathbf{w}_T$,

$$\mathbb{E}[\|\nabla \mathcal{L}(\hat{\mathbf{w}})\|] \leq \frac{\sqrt{2\Delta H}}{\sqrt{T}} + \frac{\sqrt{3\sigma\sqrt{\Delta H}}}{T^{1/4}}$$

Notice that the second statement of the Theorem bounds the expected gradient norm by a sum of two terms. The first term is identical to the bound for non-stochastic gradient descent, while the second term depends on the variance σ and has a slower $O(1/T^{1/4})$ rate of dependence on the time T.

Proof. Again, let's use our understanding of smooth losses to bound the progress made in one step of stochastic gradient descent:

$$\begin{aligned}
\mathcal{L}(\mathbf{w}_{t+1}) &\leq \mathcal{L}(\mathbf{w}_t) + \langle \nabla \mathcal{L}(\mathbf{w}_t), \mathbf{w}_{t+1} - \mathbf{w}_t \rangle + \frac{H}{2}\|\mathbf{w}_{t+1} - \mathbf{w}_t\|^2 \\
&= \mathcal{L}(\mathbf{w}_t) - \eta \langle \nabla \mathcal{L}(\mathbf{w}_t), \mathbf{g}_t \rangle + \frac{H\eta^2}{2}\|\mathbf{g}_t\|^2
\end{aligned}$$

Now, in deference to the randomness of our situation, we take the expected value of both sides:

$$\mathbb{E}[\mathcal{L}(\mathbf{w}_{t+1})] \leq \mathbb{E}[\mathcal{L}(\mathbf{w}_t)] - \eta\, \mathbb{E}[\nabla \mathcal{L}(\mathbf{w}_t), \mathbf{g}_t \rangle] + \frac{H\eta^2}{2}\, \mathbb{E}[\|\mathbf{g}_t\|^2]$$

Now, use the fact that $\mathbb{E}[\mathbf{g}_t | \mathbf{w}_t] = \nabla \mathcal{L}(\mathbf{w}_t)$, so that:

$$= \mathbb{E}[\mathcal{L}(\mathbf{w}_t)] - \eta\, \mathbb{E}[\|\nabla \mathcal{L}(\mathbf{w}_t)\|^2] + \frac{H\eta^2}{2}\, \mathbb{E}[\|\mathbf{g}_t\|^2]$$

From bias variance decomposition:

$$\begin{aligned}
&\leq \mathbb{E}[\mathcal{L}(\mathbf{w}_t)] - \eta\, \mathbb{E}[\|\nabla \mathcal{L}(\mathbf{w}_t)\|^2] + \frac{H\eta^2}{2}\, \mathbb{E}[\|\nabla \mathcal{L}(\mathbf{w}_t)\|^2 + \sigma^2] \\
&= \mathbb{E}[\mathcal{L}(\mathbf{w}_t)] - \eta(1 - \eta H/2)\, \mathbb{E}[\|\nabla \mathcal{L}(\mathbf{w}_t)\|^2] + \frac{H\eta^2\sigma^2}{2}
\end{aligned}$$

Since $\eta \leq \frac{1}{H}$:

$$\leq \mathbb{E}[\mathcal{L}(\mathbf{w}_t)] - \frac{\eta}{2}\, \mathbb{E}[\|\nabla \mathcal{L}(\mathbf{w}_t)\|^2] + \frac{H\eta^2\sigma^2}{2}$$

Summing over t and telescoping:

$$\begin{aligned}
\mathbb{E}[\mathcal{L}(\mathbf{w}_{T+1}) - \mathcal{L}(\mathbf{w}_1)] &\leq -\sum_{t=1}^{T} \frac{\eta}{2}\, \mathbb{E}[\|\nabla \mathcal{L}(\mathbf{w}_t)\|^2] + \frac{H\eta^2\sigma^2}{2} \\
\sum_{t=1}^{T} \mathbb{E}[\|\nabla \mathcal{L}(\mathbf{w}_t)\|^2] &\leq \frac{2\Delta}{\eta} + HT\eta\sigma^2
\end{aligned}$$

This proves the first part of the Theorem. Now, for the second part we consider the provided setting for η. There are two cases, either $\eta = \frac{1}{H} \leq \frac{\sqrt{\Delta}}{\sigma\sqrt{HT}}$ or not. If $\eta = \frac{1}{H}$, then:

$$\begin{aligned}\sum_{t=1}^{T} \mathbb{E}[\|\nabla \mathcal{L}(\mathbf{w}_t)\|^2] &\leq \frac{2\Delta}{\eta} + HT\eta\sigma^2 \\ &= 2\Delta H + HT\eta\sigma^2\end{aligned}$$

Further, since $\eta \leq \frac{\sqrt{\Delta}}{\sigma\sqrt{HT}}$,

$$HT\eta\sigma^2 \leq \sigma\sqrt{\Delta HT}$$

so altogether:

$$\sum_{t=1}^{T} \mathbb{E}[\|\nabla \mathcal{L}(\mathbf{w}_t)\|^2] \leq 2\Delta H + \sigma\sqrt{\Delta HT}$$

Next, consider the case $\eta = \frac{\sqrt{\Delta}}{\sigma\sqrt{HT}}$. Then, by plugging in η, we have:

$$\begin{aligned}\sum_{t=1}^{T} \mathbb{E}[\|\nabla \mathcal{L}(\mathbf{w}_t)\|^2] &\leq \frac{2\Delta}{\eta} + HT\eta\sigma^2 \\ &= 3\sigma\sqrt{\Delta HT}\end{aligned}$$

so overall, we have that $\sum_{t=1}^{T} \mathbb{E}[\|\nabla \mathcal{L}(\mathbf{w}_t)\|^2]$ is bounded by the maximum of these quantities, which is

$$\sum_{t=1}^{T} \mathbb{E}[\|\nabla \mathcal{L}(\mathbf{w}_t)\|^2] \leq 2\Delta H + 3\sigma\sqrt{\Delta HT}$$

Now, for the final statement of the theorem, divide by T and apply Jensen's inequality:

$$\begin{aligned}\mathbb{E}[\|\nabla \mathcal{L}(\hat{\mathbf{w}})\|] &\leq \sqrt{\frac{1}{T}\sum_{t=1}^{T} \mathbb{E}[\|\nabla \mathcal{L}(\mathbf{w}_t)\|^2]} \\ &\leq \sqrt{\frac{2\Delta H}{T} + \frac{3\sigma\sqrt{\Delta H}}{\sqrt{T}}} \\ &\leq \frac{\sqrt{2\Delta H}}{\sqrt{T}} + \frac{\sqrt{3\sigma\sqrt{\Delta H}}}{T^{1/4}}\end{aligned}$$

□

Theorem 5.2 has the problem that the learning rate η is set based on various parameters like H, σ and T, which are presumably not actually known. In practice, the common strategy is to simply guess the learning rate. That is:

1. Try several learning rates out.
2. Choose the one that resulted in best performance on a validation set.

However, it *is* possible to make some guarantees without requiring detailed settings for η. The standard approach is to set $\eta_t \propto \frac{1}{\sqrt{t}}$, and to rely on some kind of Lipschitz assumption. For example, one could assume that $\|\nabla \ell(\mathbf{w}, z)\| \leq G$ always. This would be implied it $\ell(\mathbf{w}, Z)$ is G-Lipschitz as a function of $\mathbf{w}$. In fact, we can make do with a slightly weaker assumption that $\mathbb{E}[\|\nabla \ell(\mathbf{w}, z)\|^2] \leq G^2$. Note that we do not need to *know* G in order to guarantee convergence, although knowing it might allow us to set the c coefficient in η_t more optimally.

Theorem 5.3. *Suppose $\mathcal{L}$ is H-smooth, and $\nabla \ell(\mathbf{w}, z)$ satisfies $\mathbb{E}_z[\|\nabla \ell(\mathbf{w}, z)\|^2] \leq G^2$ for all $\mathbf{w}$. Let $\eta_1 \geq \cdots \geq \eta_T$ be an arbitrary* deterministic *and decreasing learning rate schedule. Then:*

$$\frac{1}{T}\sum_{t=1}^{T} \mathbb{E}\left[\|\nabla \mathcal{L}(\mathbf{w}_t)\|^2\right] \leq \frac{\Delta}{T\eta_T} + \frac{HG^2}{2T\eta_T}\sum_{t=1}^{T}\eta_t^2$$

Next, set $\eta_t = \frac{c}{\sqrt{t}}$ for some c. Then:

$$\frac{1}{T}\sum_{t=1}^{T} \mathbb{E}\left[\|\nabla \mathcal{L}(\mathbf{w}_t)\|^2\right] \leq \frac{\Delta}{c\sqrt{T}} + \frac{HG^2 c(1+\log(T))}{2\sqrt{T}}$$

In particular, if $\hat{\mathbf{w}}$ is randomly selected from $\mathbf{w}_1, \ldots, \mathbf{w}_T$, then

$$\mathbb{E}[\|\nabla \mathcal{L}(\hat{\mathbf{w}})\|] \leq \frac{\sqrt{\Delta/c + G^2 c H \frac{1+\log(T)}{2}}}{T^{1/4}}$$

Proof. Again we have

$$\begin{aligned}
\mathcal{L}(\mathbf{w}_{t+1}) &\leq \mathcal{L}(\mathbf{w}_t) + \langle \nabla \mathcal{L}(\mathbf{w}_t), \mathbf{w}_{t+1} - \mathbf{w}_t\rangle + \frac{H}{2}\|\mathbf{w}_{t+1} - \mathbf{w}_t\|^2 \\
&= \mathcal{L}(\mathbf{w}_t) - \eta_t \langle \mathcal{L}(\mathbf{w}_t), \mathbf{g}_t\rangle + \frac{H\eta_t^2}{2}\|\mathbf{g}_t\|^2 \\
\mathbb{E}[\mathcal{L}(\mathbf{w}_{t+1})] &\leq \mathbb{E}[\mathcal{L}(\mathbf{w}_t)] - \eta_t\, \mathbb{E}[\|\nabla \mathcal{L}(\mathbf{w}_t)\|^2] + \frac{H\eta_t^2 G^2}{2}
\end{aligned}$$

Again we sum over t and telescope:

$$\mathbb{E}[\mathcal{L}(\mathbf{w}_{T+1})] \leq \mathbb{E}[\mathcal{L}(\mathbf{w}_1)] - \sum_{t=1}^{T}\eta_t\, \mathbb{E}[\|\nabla \mathcal{L}(\mathbf{w}_t)\|^2] + \frac{H\eta_t^2 G^2}{2}$$

Use the fact that $\eta_T \leq \eta_t$ for all t:

$$\leq \mathbb{E}[\mathcal{L}(\mathbf{w}_1)] - \eta_T \sum_{t=1}^{T}\mathbb{E}[\|\nabla \mathcal{L}(\mathbf{w}_t)\|^2 + \frac{HG^2}{2}\sum_{t=1}^{T}\eta_t^2$$

rearrange terms:

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[\|\nabla \mathcal{L}(\mathbf{w}_t)\|^2] \leq \frac{\mathbb{E}[\mathcal{L}(\mathbf{w}_1) - \mathcal{L}(\mathbf{w}_{T+1})]}{T\eta_T} + \frac{HG^2}{2T\eta_T}\sum_{t=1}^{T}\eta_t^2$$

so that we have shown the first part of the Theorem. Now, we get to use an identity that will become useful time and time again:

$$\begin{aligned}
\sum_{t=1}^{T}\frac{1}{t} &= 1 + \sum_{t=2}^{T}\frac{1}{t} \\
&\leq 1 + \int_1^T \frac{dt}{t} \\
&= 1 + \log(T)
\end{aligned}$$

Therefore, we have:

$$\begin{aligned}\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[\|\nabla\mathcal{L}(\mathbf{w}_t)\|^2] &\leq \frac{\mathbb{E}[\mathcal{L}(\mathbf{w}_1)-\mathcal{L}(\mathbf{w}_{T+1})]}{T\eta_T}+\frac{HG^2}{2T\eta_T}\sum_{t=1}^{T}\eta_t^2\\ &= \frac{\Delta}{c\sqrt{T}}+\frac{cHG^2}{2\sqrt{T}}\sum_{t=1}^{T}\frac{1}{t}\\ &\leq \frac{\Delta}{c\sqrt{T}}+\frac{cHG^2(1+\log(T))}{2\sqrt{T}}\end{aligned}$$

The final statement follows from taking square roots and using Jensen inequality. □

6 Minibatch SGD

While SGD is the basis for almost all the popular algorithms in use for training neural networks today, most of the time a number of different modifications are put in place. By far the most common change is the use of *minibatches* (in fact, minibatches are almost never *not* used). Minibatching is a way to leverage parallelism to speed up the total amount of time taken to train a model. The entire training set is called the "batch", and a random small subset of the training set is called a "minibatch". Using a minibatch is straightforward: any time you wish to call the stochastic gradient oracle, instead call it B times and return the average of those B vectors. B is called the minibatch size. The code for the most basic version of minibatch SGD is below:

Algorithm 4 Minibatch Stochastic Gradient Descent

Input: Initial Point $\mathbf{w}_1$, learning rates $\eta_1,\dots,\eta_T$, time horizon T, batch size B.
for $t = 1\dots T$ **do**
  for $i = 1\dots B$ **do**
    $\mathbf{g}_{t,i} = \nabla\ell(\mathbf{w}_t, z_{(t-1)B+i})$
  end for
  Set $\mathbf{g}_t = \frac{1}{B}\sum_{i=1}^{B}\mathbf{g}_{t,i}$.
  Set $\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t\mathbf{g}_t$.
end for

Let's analyze this algorithm. In order to do so, we are going to re-use the analysis of Theorem 5.2 by considering $\mathbf{g}_t$ as the output of a stochastic gradient oracle with variance $\frac{\sigma^2}{B}$. Specifically:

Proposition 6.1. *In Algorithm 4, suppose* $\mathbb{E}_{\mathbf{z}}[\|\nabla\ell(\mathbf{w},\mathbf{z})-\nabla\mathcal{L}(\mathbf{w})\|^2] \leq \sigma^2$ *for all* $\mathbf{w}$. *Then* $\mathbb{E}[\|\mathbf{g}_t-\nabla\mathcal{L}(\mathbf{w}_t)\|^2] \leq \frac{\sigma^2}{B}$.

Proof. This is a standard property of averages: they decrease the variance by the number of averaged items.

$$\begin{aligned}\mathbb{E}[\|\mathbf{g}_t-\nabla\mathcal{L}(\mathbf{w}_t)\|^2] &= \mathbb{E}\left[\left\|\frac{1}{B}\left(\sum_{i=1}^{B}\mathbf{g}_{t,i}-\nabla\mathcal{L}(\mathbf{w}_t)\right)\right\|^2\right]\\ &= \frac{1}{B^2}\left(\mathbb{E}\left[\sum_{i=1}^{B}\|\mathbf{g}_{t,i}-\nabla\mathcal{L}(\mathbf{w}_t)\|^2\right]+\mathbb{E}\left[\sum_{i\neq j}\langle\mathbf{g}_{t,i}-\nabla\mathcal{L}(\mathbf{w}_t),\mathbf{g}_{t,j}-\nabla\mathcal{L}(\mathbf{w}_t)\rangle\right]\right)\end{aligned}$$

Using $\mathbb{E}[\|\mathbf{g}_{t,i}-\nabla\mathcal{L}(\mathbf{w}_t)\|^2] \leq \sigma^2$ and $\mathbb{E}[\mathbf{g}_{t,i}-\nabla\mathcal{L}(\mathbf{w}_t)] = 0$,

$$\leq \frac{B\sigma^2}{B^2} = \frac{\sigma^2}{B}$$

□

Now, we notice that Algorithm 4 is actually the same as Algorithm 2, with the difference that the gradient estimates $\mathbf{g}_t$ have variance decreased to σ^2/B. Therefore by Theorem 5.2:

Theorem 6.2. *Suppose $\mathcal{L}$ is H-smooth, and $\hat{\nabla}\ell(\mathbf{w})$ has variance at most σ^2 for all $\mathbf{w}$ (that is for all $\mathbf{w}$, $\mathbb{E}[\|\hat{\nabla}\ell(\mathbf{w}) - \nabla\mathcal{L}(\mathbf{w})\|^2] \leq \sigma^2$). Let us consider a* fixed *learning rate $\eta_t = \eta$ for all t. Then so long as $\eta \leq \frac{1}{H}$, Algorithm 4 guarantees:*

$$\sum_{t=1}^{T} \mathbb{E}[\|\nabla\mathcal{L}(\mathbf{w}_t)\|^2] \leq \frac{2\Delta}{\eta} + \frac{HT\eta\sigma^2}{B}$$

Further, if we set $\eta = \frac{1}{\max(H, \sigma\sqrt{HT/B})}$, then if $\hat{\mathbf{w}}$ is selected uniformly at random from $\mathbf{w}_1, \ldots, \mathbf{w}_T$,

$$\mathbb{E}[\|\nabla\mathcal{L}(\hat{\mathbf{w}})\|] \leq \frac{\sqrt{2\Delta H}}{\sqrt{T}} + \frac{\sqrt{\sigma(1+2\Delta)}H^{1/4}}{(BT)^{1/4}}$$

Let's take a moment to appreciate how the learning rates changed when we incorporated the minibatch. For large enough T, the learning rate suggested by the theory is:

$$\eta \propto \frac{\sqrt{B}}{\sqrt{T}}$$

and the gradient size identified is:

$$\mathbb{E}[\|\nabla\mathcal{L}(\hat{\mathbf{w}})\|] \leq O(\frac{1}{(BT)^{1/4}})$$

When considering the *total computational cost*, we notice that when using a minibatch of size B, each iteration takes B times more compute since we need to compute B gradients. Thus the cost is $C = TB$. This is also the *oracle complexity* - the number of calls to the stochastic gradient oracle. Re-writing these results:

$$\eta \propto \frac{B}{\sqrt{C}}$$
$$\mathbb{E}[\|\nabla\mathcal{L}(\hat{\mathbf{w}})\|] \leq O(\frac{1}{C^{1/4}})$$

The first lesson here is that *the gradient size is independent of the batch size*. In fact, if we are more careful we would see that the best thing to optimize the constants hiding in this analysis is to set $B = 1$.

In one point of view, this is a bad thing: we increased the batch size, but the performance may not get any better! Fortunately, although we may not save on total compute cost, we might actually save in terms of *total time*. Specifically, the B computations $\mathbf{g}_{t,i} = \nabla\ell(\mathbf{w}_t, z_{(t-1)B+i})$ in Algorithm 4 can all be done in parallel. So, in theory if we had access to M machines and ignore a plethora of issues involving communication overheads, we might be able to have the total time spent by the algorithm equal to $\tau = \frac{C}{M}$. Thus, in time τ we have:

$$\mathbb{E}[\|\nabla\mathcal{L}(\hat{\mathbf{w}})\|] \leq O(\frac{1}{(M\tau)^{1/4}})$$

and so there is a clear advantage to increasing the batch size. However, there is a caveat here: all of this analysis only holds for sufficiently large T. If we have $B \geq T$, then we hit a point of diminishing returns:

Exercise 6.3. *Show that for $B \geq T$, the optimal value for η obtains only:*

$$\mathbb{E}[\|\nabla\mathcal{L}(\hat{\mathbf{w}})\|] \leq O(\frac{1}{\sqrt{T}}) = O(\frac{\sqrt{B}}{\sqrt{C}})$$

so that increasing B may be actively harmful. Can you think of an intuitive reason why this should be expected?

7 Automatic Differentiation

Note: throughout this section, we will usually use the notation $x[a]$ to indicate the ath component of a vector x, $M[a, b]$ to indicate the a, bth entry of a matrix M and so on in order to make correspondence between formulas and code as close as possible.

One of the most important advances in machine learning engineering of recent years is actually based on very old classical ideas: automatic differentiation. The development of robust automatic differentiation packages over the last decade (e.g. Tensorflow, Pytorch), has significantly simplified and accelerated the development of machine learning models and training algorithms. Since almost all optimization algorithms in use today are based on derivatives, the ability to easily compute derivatives is critical. Prior to the widespread adoption of automatic differentiation, one needed to manually code up derivative formulas, which was not only tedious but also very error-prone and increasingly difficult as models become more complex. Automatic differentiation significantly simplifies this process.

Some of the techniques in automatic differentiation go back to Leibniz, and have been refined (and rediscovered) several times in recent decades, most famously in the "backpropogation" algorithm Rumelhart, Hinton, and Williams 1986. In this section we will cover the most common form of automatic differentiation used in machine learning, which is called "reverse-mode differentiation". It can be viewed as a generalization of the backpropogation algorithm that you may be familiar with, and is the underlying method implemented in popular packages like Tensorflow and Pytorch. Please also see Baydin et al. 2018 for an excellent survey of this topic, including a description of the "forward-mode differentiation" that we will not cover here.

What AD does

In order to understand how automatic differentiation works, we need to first understand what it will accomplish. Automatic differentiation will allow us to write some code describing a function $f : \mathbb{R} \to \mathbb{R}$, and then given an input x, we will be able to compute the derivative $f'(x)$. Now, for one-dimensional functions like this there is already an intuitively obvious way to compute an approximate derivative: simply take some small h and set $f'(x) \approx \frac{f(x+h)-f(x)}{h}$. This method, which we will call *numerical differentiation*, has three main issues:

1. Numerical differentiation is less exact as h gets bigger.
2. Numerical differentiation suffers larger and larger floating point round-off error as h gets smaller.
3. Numerical differentiation requires $O(d)$ evaluations of f to compute a gradient when $f : \mathbb{R}^d \to \mathbb{R}$ (need to compute each component of of $\nabla f(x)$ individually by $\nabla f(x)_i \approx \frac{f(x+h\cdot e_i)-f(x)}{h}$).

The first two issues mean that numerical differentiation is fundamentally a little bit inaccurate. This is not necessarily a fatal flaw - small errors in the gradient computation may be tolerable in our optimization algorithm. The last issue is much more important as in machine learning we typically consider models with a very large number of parameters (high d), and so computing the gradient would be very slow. Automatic differentiation fixes the first two issues, and usually allows us to compute a gradient in roughly twice the amount of time it takes to compute f, which is a huge saving at high dimensions.

AD is NOT symbolic differentiation

You may have experience with symbolic algebra packages (e.g. mathematica, or sufficiently advanced graphing calculators). These systems can "symbolically" differentiate: given a function $f(x) = x^2 + 3x$, they will describe the derivative as a symbolic expression $f'(x) = 2x + 3$. Thus, these systems actually mimic the way we as humans would compute derivatives. However, this is more work than really we need: no optimization algorithm we will consider actually needs the full expression for the derivative, we just need to be able to evaluate $f'(x)$ for *specific values of* x. Automatic differentiation provides this capacity without actually computing the symbolic expression for the derivative. In fact, computing the symbolic derivative is *also* slow just like numerical differentiation.

7.1 Derivatives

Now that we know a bit about what our goal will be, we need to take another step back and understand exactly what a derivative is. In these notes, when we speak of the derivative, we mean the "total derivative", which is defined as follows:

Definition 7.1. *A function $f : \mathbb{R}^n \to \mathbb{R}^m$ is differentiable at a point $x \in \mathbb{R}^n$ if there exists a linear map $D_{f,x} : \mathbb{R}^n \to \mathbb{R}^m$ such that:*

$$\lim_{h \to 0} \frac{f(x+h) - f(x) - D_{f,x}(h)}{\|h\|} = 0$$

The linear map $D_{f,x}$ is called the Total Derivative (or just derivative) of f at x.

In words, the derivative is the linear approximation to the function f that is valid near x. As an exercise, take a moment to see why the above definition agrees with the standard definition you may have learned in a basic calculus class, where we would write the linear map $D_{f,x}$ as $D[f, x](h) = f'(x) \cdot h$. Further, note that the *domain* and *range* of f and the derivative $D_{f,x}$ are identical. For a more formal treatment, when f is a function of smooth manifolds rather than vector spaces, technically the domain and range of $D_{f,x}$ should be the *tangent spaces* to the domain and range of f at x and $f(x)$ respectively, but in our scenario for which the domain and range of f are both vector spaces this distinction is not relevant. For a nice introduction to the formalism of functions and derivatives on manifolds see Tu 2011.

Remark on Notation: We use the particular notation $D_{f,x}$ rather than the potentially more familiar notation $\frac{df}{dx}$ since the relevant expression is a linear map and even less related to a fraction than the standard scalar derivative is (which was anyway already not that related to a fraction). Later, we will discuss partial derivatives using a similar notation instead of the more familiar $\frac{\partial f}{\partial x_i}$. While our notation allows for a little more precision, once you become comfortable with the ideas it is fine to use the fraction-style notation to keep things a little more compact with fewer subscripts.

This definition may seem a little abstract. How can we actually compute the derivative? Notice that $D_{f,x}$ is a linear map, and so can be represented by a matrix in $\mathbb{R}^{m \times n}$. It turns out that so long as all partial derivatives of f exist and are continuous, we can write the i, jth component of $D_{f,x}$ as:

$$D_{f,x}[i, j] = \frac{\partial f(x)[i]}{\partial x[j]}$$

where we use $f(x)[i]$ to indicate the ith component of $f(x) \in \mathbb{R}^m$. This matrix is called the *Jacobian*. For essentially all practical purposes, you may assume that the Jacobian is equal to the total derivative: we will never consider any situation in which this is not the case.

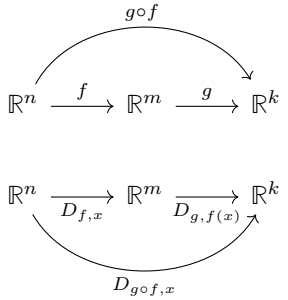
Finally, notice that in the special case that $f : \mathbb{R}^n \to \mathbb{R}$, the total derivative is a linear map $\mathbb{R}^n \to \mathbb{R}$, which can be represented as a $1 \times n$ matrix, or a transposed vector. Inspecting the formula for the total derivative, one sees that this is exactly the same as the familiar *gradient*. Thus, after we develop machinery to compute general derivatives, we can easily apply the techniques to compute gradients, which are of particular interest in the algorithms used in machine learning.

The total derivative defined this way satisfies familiar properties from scalar calculus:

1. The derivative is linear: for functions f and g and a scalar c, $D_{cf,x} = cD_{f,x}$ and $D_{f+g,x} = D_{f,x} + D_{g,x}$.
2. The derivative satisfies the following generalization of the chain rule: If $f : \mathbb{R}^n \to \mathbb{R}^m$ and $g : \mathbb{R}^m \to \mathbb{R}^k$, then $D_{g \circ f,x} = D_{g,f(x)} \circ D_{f,x}$.

The chain rule is particularly important for automatic differentiation. Let's take a moment to appreciate what's happening here, since the composition notation can be a bit confusing. Suppose $f : \mathbb{R}^n \to \mathbb{R}^m$ and $g : \mathbb{R}^m \to \mathbb{R}^k$ are two different functions, then we can draw the following diagram[1]:

[1] this sort of diagram, in which nodes are labeled by sets and arrow functions and any two paths between nodes represent the same function is called *commutative diagram* in mathematics.



Thus, $g \circ f$ is a function $\mathbb{R}^n \to \mathbb{R}^k$, so the derivative $D_{g\circ f,x}$ should be a *linear* function $\mathbb{R}^n \to \mathbb{R}^k$. The chain rule then says that the linear function obtained by first moving from $\mathbb{R}^n$ to $\mathbb{R}^m$ via the linear map $D_{f,x}$ and then moving to $\mathbb{R}^k$ via the linear map $D_{g,f(x)}$ is equal to the derivative $D_{g\circ f,x}$.

Finally, to make this more concrete, remember that for all practical settings, the derivative is given by the Jacobian matrix of partial derivatives. Further, composition of linear functions is the same as multiplication of the corresponding matrices. Let's consider a particular example: suppose $n = 2, m = 2, k = 3$. Define f and g as:

$$f(x) = (x[1], 1 + x[0])$$
$$g(z) = (z[0]z[1], z[0], z[1])$$

This example is simple enough that we can compute by hand:

$$(g \circ f)(x) = (x[0]x[1] + x[1], x[1], 1 + x[0])$$

$$D_{g\circ f,x} = \begin{pmatrix} x[1] & x[0]+1 \\ 0 & 1 \\ 1 & 0 \end{pmatrix}$$

Let's verify this via the chain rule:

$$D_{f,x} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

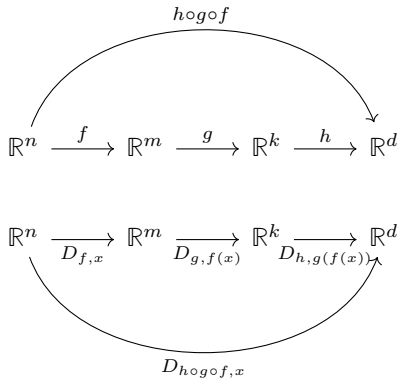
$$D_{g,z} = \begin{pmatrix} z[1] & z[0] \\ 1 & 0 \\ 0 & 1 \end{pmatrix}$$

$$D_{g,f(x)} = \begin{pmatrix} 1+x[0] & x[1] \\ 1 & 0 \\ 0 & 1 \end{pmatrix}$$

Now, we multiply the matrices:

$$D_{g,f(x)} D_{f,x} = \begin{pmatrix} x[1] & x[0]+1 \\ 0 & 1 \\ 1 & 0 \end{pmatrix}$$

Now that we've seen this simpler example, let's think about what would happen if we composed *three* functions: $h \circ g \circ f$. Using the chain rule (twice), we can draw a very similar diagram:



7.2 Partial derivatives

Note: Much of the material in this section is essentially a re-stating of familiar partial-derivative facts in our language of total derivatives. There should be no surprises as everything is completely analogous to the standard facts one learns in multivariable calculus. The key result that is useful for automatic differentiation is that when computing the derivative of a function with multiple outputs, we need to add up all the partials. This is encapsulated in the standard multivariable calculus chain-rule formula:

$$\frac{d}{dt} f(x_0(t), \ldots, x_{k-t}(t)) = \sum_{i=0}^{k-1} \frac{\partial f}{x_i} \frac{dx_i}{dt}$$

This section simply provides the generalization of this rule, which has essentially the same form.

Now that we have defined the total derivative, let us turn to an analog of partial differentiation that will be useful in describing automatic differentiation. Let $f : \mathbb{R}^{d_0} \times \mathbb{R}^{d_2} \times \cdots \times \mathbb{R}^{d_{k-1}} \to \mathbb{R}^n$ be a function taking k arguments, of which the ith argument is in $\mathbb{R}^{d_i}$:

$$x_i \in \mathbb{R}^{d_i}$$
$$f(x_0, \ldots, x_{k-1}) \in \mathbb{R}^n$$

Then we define the partial derivative of f with respect to the ith argument x_i at a point $x = (x_0, \ldots, x_{k-1})$ as the total derivative of the function $f^{i,x} : \mathbb{R}^{d_i} \to \mathbb{R}^n$ given by $f^{i,x}(z) = f(x_1, \ldots, x_{i-1}, z, x_{i+1}, x_k)$:

$$\partial_{f,i,x} = D_{f^{i,x},x_i}$$

Thus, $\partial_{f,i,x}$ is a linear map from $\mathbb{R}^{d_i}$ to $\mathbb{R}^n$, and in matrix form the a, bth entry is:

$$\partial_{f,i,x}[a, b] = \frac{\partial f[a]}{\partial x_i[b]}$$

Let's make this concrete: Consider a function $f : \mathbb{R}^2 \to \mathbb{R}^1$ given by $f(x) = x[0]x[1]$, $g : \mathbb{R}^1 \to \mathbb{R}^1$ given by $g(x) = x^2$ and $h : \mathbb{R}^1 \to \mathbb{R}^1$ given by $h(x) = 3x$. Putting these functions together, we might have the composition $p(x) = f(g(x), h(x)) = 3x^4$. Then the partial derivative $\partial_{f,0,x}$ is a linear map from $\mathbb{R} \to \mathbb{R}$ (i.e. a scalar), that is equal to the ordinary partial derivative $\frac{\partial f}{\partial x_0} = x[1]$.

Note: We use the notation $\partial_{f,i,x}$ to indicate that x is a variable with a value (where the partial derivative is evaluated), while i is the index of the argument to differentiate with respect to (an integer). However, the more familiar calculus notation is $\frac{\partial f}{\partial x_i}$, which combines the index and the variable value into one. There are tradeoffs here: our notation is more precise as it explicitly separates these two concerns, but occasionally we may wish to refer to an argument of f not by a name rather than an integer index: for example, for a two-input function $f(x, y)$ we will occasionally also use the notation $\partial_{f,x,(x,y)}$ rather than $\partial_{f,0,(x,y)}$. The meaning should always be clear from context.

This notion of partial differentiation has an important "associativity" property: if $f : \mathbb{R}^{d_0} \times \mathbb{R}^{d_1} \times \mathbb{R}^{d_2} \to \mathbb{R}^n$ is a 3-argument function, we could "combine" the last two arguments to consider instead a 2-argument function $\bar{f} : \mathbb{R}^{d_0} \times \mathbb{R}^{d_1 \cdot d_2} \to \mathbb{R}^n$. Now, we have the identities:

$$\partial_{\bar{f},0,x} = \partial_{f,0,x} \tag{1}$$

$$\partial_{\bar{f},1,x} = (\partial_{f,1,x}, \partial_{f,2,x}) \tag{2}$$

where the last notation $(\partial_{f,1,x}, \partial_{f,2,x})$ indicates concatenation of the a $n \times d_1$ matrix with an $n \times d_2$ matrix. In other words, the "partial derivative" is essentially just chopping up the total derivative matrix. In particular, observe that the matrix form of the total derivative of a k-argument function can be written as:

$$D_{f,x} = (\partial_{f,0,x},\ \partial_{f,1,x},\ \ldots\ ,\ \partial_{f,k-1,x})$$

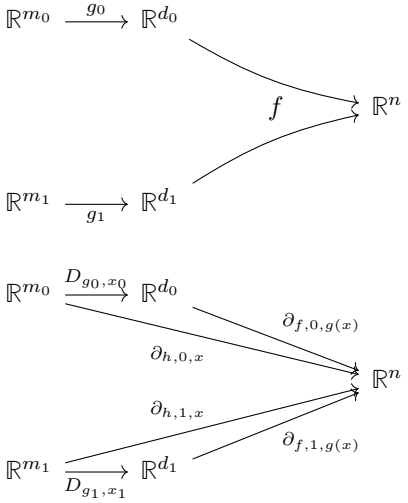
The partial differentiation satisfies the following chain rule : Consider k functions $g_i : \mathbb{R}^{m_i} \to \mathbb{R}^{d_i}$ for $i = 0, \ldots, k-1$, and a k-argument function $f : \mathbb{R}^{d_0} \times \mathbb{R}^{d_2} \times \cdot \times \mathbb{R}^{d_{k-1}} \to \mathbb{R}^n$. We will write $g(x)$ to indicate $(g_0(x_0), \ldots, g_{k-1}(x_{k-1}))$. Let $h : \mathbb{R}^{m_0} \times \cdots \times \mathbb{R}^{m_{k-1}} \to \mathbb{R}^n$ be the composition given by

$$h(x_0, \ldots, x_{k-1}) = f(g(x)) = f(g_0(x_0), \ldots, g_{k-1}(x_{k-1}))$$

Then

$$\partial_{h,i,x} = \partial_{f,i,g(x)} \circ D_{g_i,x_i}$$

A diagram for the $k = 2$ case might be the following, where we use curved arrows sharing the same label "f" to indicate that the function f takes both starting points of the arrows as input.



7.3 Multiple Outputs

The previous section discussed how to deal with multiple inputs using partial derivatives. We saw that the partial derivative is just a way to organize the total derivative by dividing it into groups of columns. Next, we will consider how to deal with multiple *outputs* using an entirely analogous idea that divides the total derivative into groups of *rows*.

Consider a k-input, p-output function $f(x_0, \ldots, x_{k-1}) = (f^0(x), \ldots, f^{p-1}(x))$, where $f^i(x) \in \mathbb{R}^{s_i}$ and $x_i \in \mathbb{R}^{d_i}$. Then we can form the partial derivative of f^a with respect to x_b, $\partial_{f^a,b,x}$ by simply considering the function $f^a : \mathbb{R}^{d_0} \times \cdots \times \mathbb{R}^{d_{k-1}} \to \mathbb{R}^{s_a}$ in isolation. Using this terminology, we can write the total derivative of the function $f : \mathbb{R}^{d_0 \cdots d_{k-1}} \to \mathbb{R}^{s_0 \cdots s_{p-1}}$ as a block-matrix in the following way:

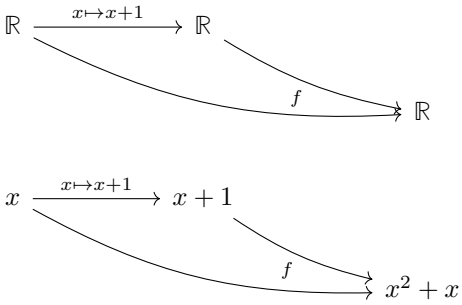
$$D_{f,x} = \begin{pmatrix} \partial_{f_0,0,x} & \cdots & \partial_{f_0,k-1,x} \\ \vdots & \ddots & \vdots \\ \partial_{f_{p-1},0,x} & \cdots & \partial_{f_{p-1},k-1,x} \end{pmatrix}$$

Thus, we can see that this notion of partial differentiation, taken to the extreme in which $d_i = s_i = 1$, is completely consistent with our identification of the derivative matrix with the Jacobian of f.

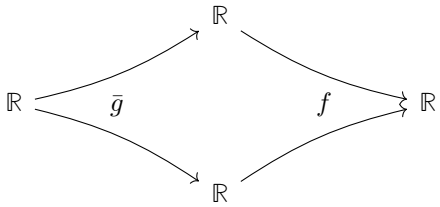
The chain rule for partial derivatives including both multi-output and multi-input is the following: Consider a p-input and k-output function $g : \mathbb{R}^{t_0,\ldots,t_{p-1}} \to \mathbb{R}^{d_0} \times \cdots \times \mathbb{R}^{d_{k-1}}$ and a k-input function $f : \mathbb{R}^{d_0} \times \cdots \times \mathbb{R}^{d_{k-1}} \to \mathbb{R}^n$. Then

$$\partial_{f\circ g,i,x} = \sum_{j=0}^{k-1} \partial_{f,j,g(x)} \circ \partial_{g^j,i,x}$$

Now, finally, we can add one more twist to the diagram: what if one of the domains in the diagram is used *twice*? For example, suppose $f : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ is given by $f(x_0, x_1) = x_0 x_1$, but we actually call f on x and $x + 1$, where we are *reusing* the 1-dimensional quantity x - that is, $f(x, x + 1) = x^2 + x$. Formally, we could model this using a function $g : \mathbb{R} \to \mathbb{R} \times \mathbb{R}$ given by $g(x) = (x, x + 1)$ and then $h(x) = f(g(x))$. However, diagrammatically the picture is perhaps more intuitive:



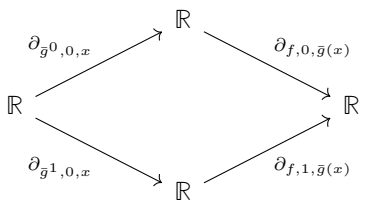
To use the chain rule here, we can re-write this calculation in terms of a 2-output function $\bar{g}(x) = (x, x + 1)$:



Using this diagram, we can compute the derivative using the chain rule:

$$\begin{aligned}\frac{dh}{dx} &= \partial_{f,0,\bar{g}(x)}\partial_{\bar{g}^0,0,x} + \partial_{f,1,\bar{g}(x)}\partial_{\bar{g}^1,0,x} \\ &= \bar{g}^1(x) \cdot 1 + \bar{g}^0(x) \cdot 1 \\ &= 2x + 1\end{aligned}$$

which we can easily see is equal to the derivative of $h(x) = x^2 + x$ as desired. Note that we can write a partial derivative diagram in the following way:



Then, the derivative of the final output with respect to the input can be computed by first *multiplying partial derivatives along all paths* and then *adding up* these products for all paths from the input to the output.

7.4 Forming a Computation Graph

Now, let's put all this chain rule formalism together into a concrete algorithm. For this, we will lean heavily into the notion of a *computation graph*, which is basically a computerized representation of the diagrams in the previous section. Our computation graphs will have the following properties (not all of these are strictly required, but will make our algorithm simpler - feel free to try to see how to generalize the method to remove restrictions).:

- All computation graphs are *directed acyclic graphs*.
- Since the graph is directed, for any node a we say b is a *child* of a if there is an edge $a \to b$. Similarly, we say b is a parent of a if there is an edge $b \to a$. If there is a sequence of edges $a \to \cdots \to b$, we say that a is an ancestor of b, or that b is *downstream* of a. Note that since the graph is acyclic, it cannot be that both a is downstream of b and b is downstream of a simultaneously.
- All nodes in the computation graph are either labeled as *variable* nodes, or *operation* nodes.
- The graph is bi-partite: all neighbors of operation nodes are variable nodes and vice-versa.
- All operation nodes have exactly one child node.
- All variable nodes have at most one parent node.
- All source nodes (i.e. nodes that have no parents) and all sink nodes (nodes that have no children) are variable nodes. In particular, all operation nodes have at least one parent.
- There is only one sink node.

A computation graph is intended to describe how one could compute a particular function. The *source* nodes represent *inputs* to the function, while the *sink* node represents the function's output. For example, consider the function $h(x) = f(g(x))$ where $f(x) = x^2$ and $g(x) = x + 1$. When written as a computation graph, we have two *operation* nodes, labeled f and g, and three variable nodes which we will label x, y and z as follows (we will put boxes around operation nodes to distinguish them from variable nodes):

$$x \longrightarrow \boxed{g} \longrightarrow y \longrightarrow \boxed{f} \longrightarrow z$$

In order to make the use of the computation graph clearer, we can also label the operation nodes with the actual calculations that they perform, although we may sometimes omit this information in order to save space:

$$x \longrightarrow \boxed{g : x \mapsto x + 1} \longrightarrow y \longrightarrow \boxed{f : x \mapsto x^2} \longrightarrow z$$

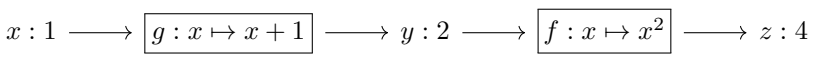
Thus, the *parents* of an operation node indicate the *arguments* that should be provided to that operation, while the *child* of an operation node indicates the output of that function. To compute the overall function (or "run" the computation graph), we first provide an initial value to the source variable nodes. This provides enough information to compute the output of some of the operation nodes, which are then passed on as the values of their children, which are then used to compute downstream operations and so on. For example, we could first provide the value of 1 to the source node x, which would be stored as some extra value in the same data structure that is holding the node x, diagrammed as follows;

$$x : 1 \longrightarrow \boxed{g : x \mapsto x + 1} \longrightarrow y \longrightarrow \boxed{f : x \mapsto x^2} \longrightarrow z$$

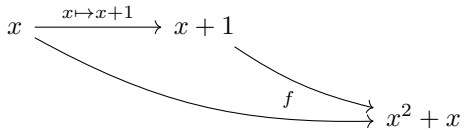
Now, all of the parents of the operation node g have values attached to them, so we can compute the output value of 2 and attach that value to the child node:

$$x : 1 \longrightarrow \boxed{g : x \mapsto x + 1} \longrightarrow y : 2 \longrightarrow \boxed{f : x \mapsto x^2} \longrightarrow z$$

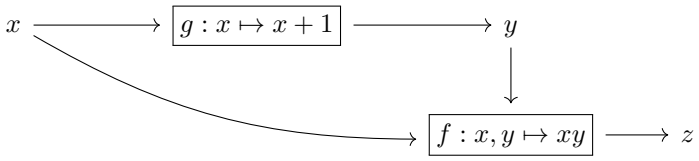
Continuing in the same way:



The point of splitting nodes into variable and operation nodes is to avoid the clunky curved arrow diagrams in the previous section for functions that take multiple inputs. For example, considering again the diagram:



we would write this as a computation graph in the following way:

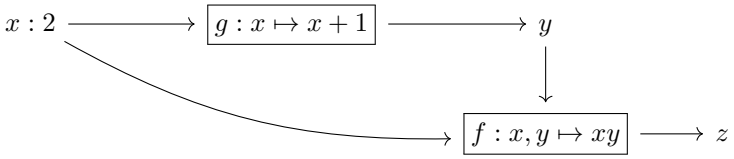


Now, we are ready to walk through the two phases of reverse-mode differentiation on this computation graph.

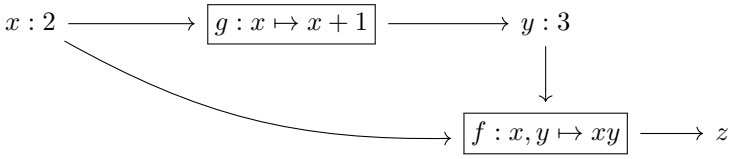
Phase 1: Forward Pass

In the forward pass phase, we actually compute the values associated with the variable node x, y and z. We *also* will store some extra information that will be useful eventually in the second phase.

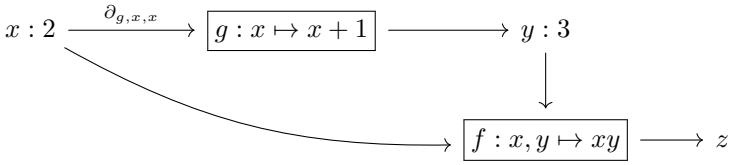
To perform this phase, we need to provide input values for all source nodes in the graph, which in this case is just the x node. Say the value is 2. Then we start by labeling the source node with its value:



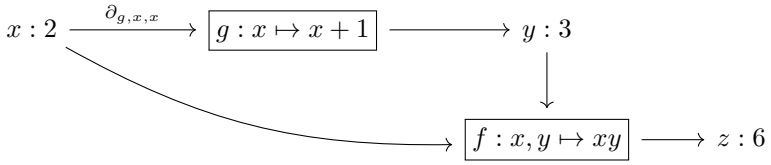
Then we step through all operation nodes for which all the parent variable nodes have been provided a value (in this case just g), and label the corresponding child nodes with the ouputs of the operations (in this case, y):



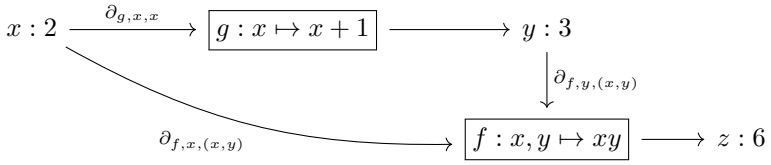
At the same time, we need to record some extra information. Let us label all of the *edges* from parent variable nodes to child operation nodes with the *partial derivatives* of the operation's output with respect to the parent variable input:



Now we repeat the process. Since we have labeled the value for y, we can now perform the operation f:



and we label the edges with appropriate partial derivative computations:

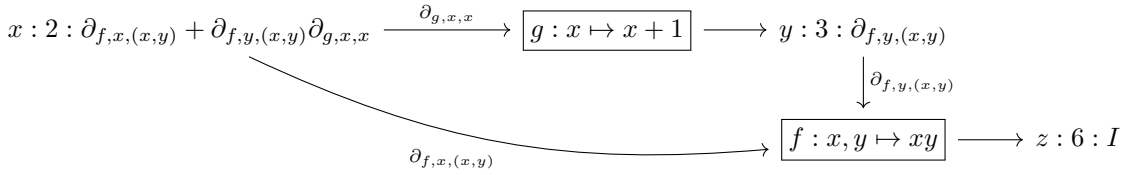


Now the forward pass is complete. Notice that the sink node (z) is labeled with the final output of the computation (6). It's time to move on to the second phase.

Phase 2: backward pass

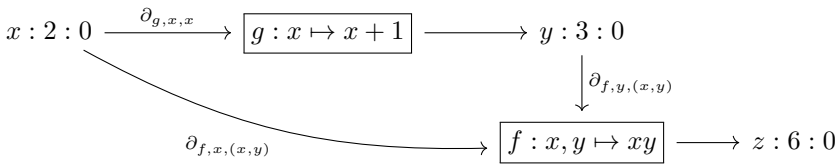
In the backward pass, we use the information in the forward pass to compute derivatives. In this phase, we start at the *sink* node (z), and follow the arrows backward towards the source nodes (in this case just one, x). While doing so, we will *multiply* the partial derivatives we encounter and accumulate running products in the variable nodes as derivatives.

In the end, the graph will be labeled as follows (I indicates the identity matrix):



By the chain rule, the label for each source node represents the partial derivative of the final output z with respect to that variable.

To accomplish this algorithmically, we perform a *backwards depth-first search* of the graph starting at the sink node z and moving towards the source node z. First, we label each variable node with an accumulator set to 0:



Let us call the value of the accumulator for a node V as $V.derivative$. Then, the backward pass is described by Algorithm 8 and Algorithm 7 below. Note that Algorithm 8 can lead to a extremely slow operation on graphs with many branches (try to think why!). We will later discuss how to make the algorithm efficient.

Algorithm 5 VariableNodeBackward (simpler, less efficient version)

Input: Variable Node V, Child Operation node C and downstream partial derivative product D,
Add D to $V.derivative$: $V.derivative \leftarrow V.derivative + D$.
Let O be the parent operation node of V (if there is one).
Call OperationNodeBackward(O, D).

Then, to start the backward pass, we call VariableNodeBackward(z, I), where I is the identity matrix.

Algorithm 6 VariableNodeBackward

Input: Variable Node V, Child Operation node C and downstream partial derivative product D,
Add D to $V.derivative$: $V.derivative \leftarrow V.derivative + D$.
if all child Operation nodes of V have provided a partial derivative **then**
  Let O be the parent operation node of V (if there is one).
  Call OperationNodeBackward(O, D).
end if

Algorithm 7 OperationNodeBackward

Input: Operation Node O, Downstream partial derivative product D
Let P be the set of parent variable nodes of O:
for V in P **do**
  Let ∂ be the partial derivative labeling on the edge $V \to O$.
  Call VariableNodeBackward(V, O, $D\partial$) (the product indicates matrix product).
end for

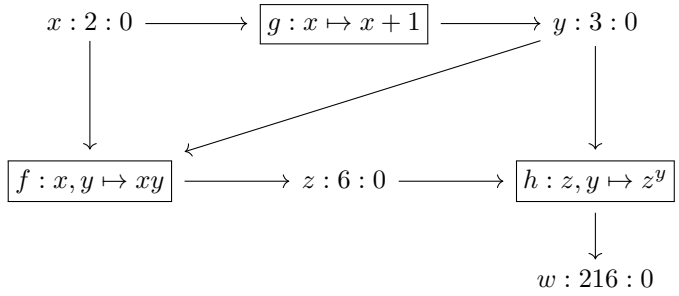
Formally, this will cause each variable node V to satisfy:

$$V.derivative = \sum_{\text{paths } V = V_0 \to O_1 \to V_1 \to O_2 \dots O_N \to V_N = Z \text{ from } V \text{ to output node } Z} \partial_{O_N, V_{N-1}} \partial_{O_{N-1}, V_{N-2}} \cdots \partial_{O_1, V_0}$$

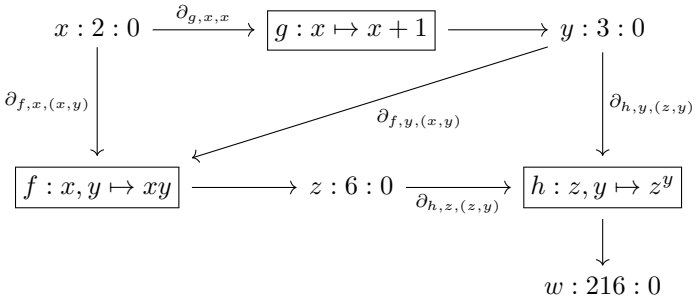
Where here we have use $\partial_{O_i, V_{i-1}}$ to indicate the partial derivative labeling on the edge $V_{i-1} \to O_i$.

More complicated example

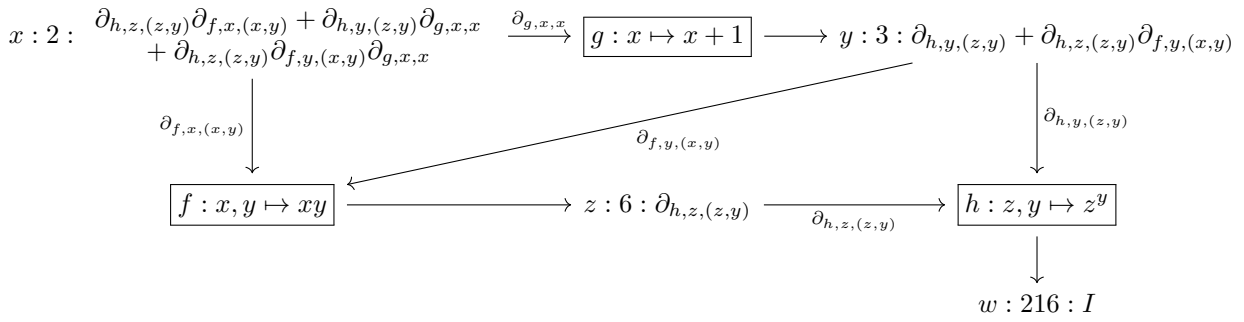
Let's work through a more complicated example. Here is the computation graph before the forward pass:



Now, after the forward pass:



Finally, after the backward pass we have:



7.5 Dynamic Graphs: Simultaneous Forward-pass and Graph Definition

The procedure as described above is known as the "static graph" method of automatic differentiation: we first define the computation graph, and then compute the forward pass and then the backward pass. This method is roughly what was employed in initial versions of Tensorflow.

Nowadays, many frameworks by default utilize a slightly different system called "dynamic graph" creation. This was arguably popularized by Pytorch, and results in significantly easier debugging (it is called "eager mode" in Tensorflow, and has been the default mode since Tensorflow 2.0). One of the problems with the static graph method is that if you make an error in defining the graph (i.e. you try to multiply matrices of the wrong shape), the error may only be detected during the forward pass, rather than at the time that the actual operation node is defined by the program. This makes it harder to trace which lines of code is causing a problem.

To fix this, the dynamic graph approach is to define the computation graph while performing the forward pass. To do this, we provide the programmer with two kinds of objects corresponding to variable nodes and operation nodes. The programmer is allowed to define a new variable node initialized with a value, which will be a source node in the graph. Then, the programmer may define an operation node and call a "forwardpass" function on the operation node, which takes as input the operation node and some variable nodes to use as parents. This immediately performs the forward pass, computing and saving the relevant partial derivatives, and then providing as output a new variable node that already has its value set to be the result of the operation.

As an example, consider constructing the $x^2 + x$ graph we have been using as an example:

1. Initialize a variable node Vx = Variable(value=2). At the moment Vx has no parents and no children and its derivative accumulator Vx.derivative is 0. It is the only node in the graph:

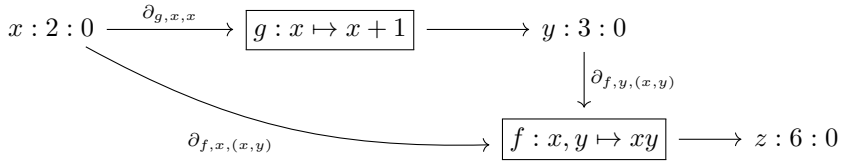
$$x : 2 : 0$$

2. Initialize a new operation node corresponding to the "add a constant" operation with the constant set to 1: Og = PlusConstantOperation(constant=1).

3. Tell this new operation node that its parent is Vx using the forward function: Vy = Og.forward(Vx). This will produce a new variable node Vy and run the forward pass, so that the graph now looks like:

$$x : 2 : 0 \xrightarrow{\partial_{g,x,x}} \boxed{g : x \mapsto x + 1} \longrightarrow y : 3 : 0$$

4. Next, we initialize a new operation node corresponding to the multiply operation: Of = MultiplyOperation(), and call forward on it: Vz = Of.forward(Vx, Vy) so that the graph now looks like the completed forward pass:

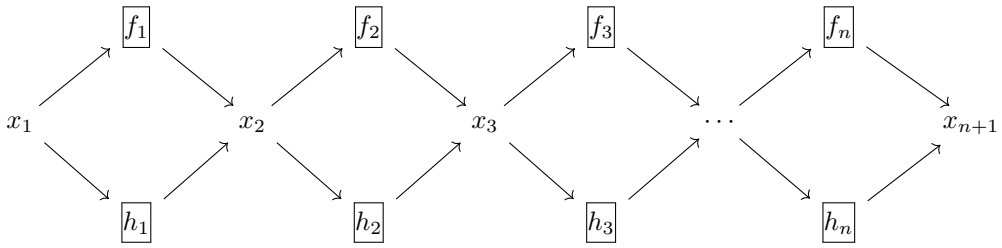


Notice that if we had tried some forbidden operation (e.g. maybe g was a square root operation and x was negative), the error would have been detected immediately rather than waiting for a separate forward call later on.

One further advantage of the dynamic graph formulation is slightly simpler coding: if we were to first construct the graph and then perform the forward pass separately, we would need to build the graph using both forward and backward pointers for the edges so that we could find the children of a node during the forward pass and also find the parents of a node during the backward pass. In the dynamic graph formulation, we don't really need to store the children of a node since the forward pass is performed at the same time as graph construction. We do still need to store pointers to the parents of each node for the backward pass however.

7.6 Efficient implementations: avoiding an exponential slowdown

Consider the following computational graph:



How long will it take for our automatic differentation algorithm to compute the derivative on this graph? It turns out the answer is $O(2^n)$ time! To see this, let us count the number of time backward will be called on any given variable not x_i. Clearly, it is called only once on x_n. Suppose it is called p times on x_{n-k} for some k. Then, each of these p backward calls will call backward on the operation nodes f_{n-k} and h_{n-k}, and each of these operation node backward calls will call backward on x_{n-k-1}, for a total of $2p$ calls. Thus, one can argue by induction that the number of backward calls on variable node x_{n-k} is 2^k, for a total of $O(2^n)$ when we get to x_1.

This is obviously very bad and absolutely needs to be fixed in any practical implementation. Fortunately, it can be avoided by the following simple modification to Algorithm 8:

Algorithm 8 VariableNodeBackward (faster version)

Input: Variable Node V, Child Operation node C and downstream partial derivative product D,
Add D to $V.derivative$: $V.derivative \leftarrow V.derivative + D$.
if all child Operation nodes of V have provided a partial derivative **then**
  Let O be the parent operation node of V (if there is one).
  Call OperationNodeBackward($O, V.derivative$).
end if

The idea behind this change is that each variable node should only call backward on its parent operation node *once*. This call should happen after the variable node has finished accumulating all of the partial derivatives from its children, so that $V.derivative$ is indeed the complete partial derivative $\frac{\partial F}{\partial V}$. This does not change the final answer since the derivative is a linear map. Since operation nodes have only one child, this means that each operation node will only have backward called on it once. As a result, the total time taken will be on the order of the number of edges in the computation graph.

For those familiar with graph algorithms, an equivalent approach is to first *topologically sort* the graph, and then call backwards on nodes in the order described by the topological sort.

7.7 Efficient implementations: no need to store entire partial derivatives

We make one final note here: for mathematical simplicity we have labeled edges with the full partial derivatives of the operations. However, in practice it is frequently unnecessary to actually compute this value. Notice that in the backward pass for a operation node (Algorithm 7), we never actually need the real partial derivative ∂. Instead, we just need to be able to compute the *product* $D\partial$, where D is the downstream derivative. Frequently, it is possible to make this computation much much faster than by actually forming the matrix ∂ and computing the matrix product.

As an example, consider the operation $x \mapsto x + 1$. The partial derivative for this operation is simply the identity matrix I. This is an $d \times d$ matrix if $x \in \mathbb{R}^d$. Let's suppose the final computation output is 1-dimensional, so that the downstream derivative is a $1 \times d$ matrix. Naively to compute the produce $D\partial$, we would need $d \times d$ space to store the identity matrix I, and then $O(d)$ time to compute the matrix product $D\partial$. However, obviously the product of any matrix with the identity is just itself. Thus, we do not actually need to store anything at all, and we also don't even need to do any computation: we can just immediately pass on D as the product $D\partial$. This is a significant saving since in machine learning we frequently work with very large dimensions d.

Of course, in practice the partial derivative is rarely the identity. Nevertheless, typically there *is* some structure in the partial derivative that makes it especially easy to compute the product $D\partial$ without having to store a very large matrix or make a full matrix-product calculation. Sometimes one might also be able to leverage some structure in D as well: for example, a simple version of the reverse-mode differentiation might assume that the final computational output of the graph is always 1-dimensional, in which case we know that D is $1 \times d$ matrix, and this could also be useful for saving computation time.

7.8 Beyond matrices: higher order tensors

It is frequently convenient to organize data into "higher-dimensional" matrices - for example a 4-dimensional array M with dimensions $1000 \times 32 \times 32 \times 3$ might represent 1000 32×32 color images, where $M[b, x, y, 0], M[b, x, y, 1], M[b, x, y, 2]$ indicates the red, green and blue color intensities respectively of the pixel at coordinates (x, y) in the bth image. When working with functions of such data, we might be tempted to "flatten" the array into a $1000 \cdot 32 \cdot 32 \cdot 3$-dimensional vector so as to be able to store it in a "variable" node and apply all the automatic differentiation machinery above. While this is certainly a potential solution, it can become very onerous to manage all this flattening and unflattening. Instead, it is frequently possible to maintain objects in their native shape through the use of *tensor* notation and *tensor contraction*.

Although tensors have a rich mathematical interpretation, for our purposes we will simply consider a tensor to be an n-dimensional array of real numbers. We will say that the *shape* of a tensor is a an n-tuple of numbers describing the dimensions of the array. So, our image dataset is represented by a 4-dimensional tensor with shape $(1000, 32, 32, 3)$. We will also frequently write the shape in a Cartesian product notation $1000 \times 32 \times 32 \times 3$. Thus, the total number of entries in a tensor T is the product of its dimensions.

Examples: A d-dimensional *vector* is usually represented as a 1-dimensional tensor of shape (d), or as a 2-dimensional tensor of shape $d \times 1$. The *transpose* of a vector is usually the 2-dimensional tensor of shape $1 \times d$. A $n \times m$ matrix is just a 2-dimensional tensor of shape $n \times m$.

Tensor Contraction

In the same way that a $n \times m$ matrix represents a linear map from $\mathbb{R}^m$ to $\mathbb{R}^n$, tensors also represent linear maps between different shapes of tensors. Formally, a tensor T of shape (X, Y, Z, W) can be considered a linear function taking tensors of shape (Z, W, A) to tensors of shape (X, Y, A). The formula is purely analogous to matrix product. If V is a (Z, W, A)-shape tensor, then:

$$(TV)[x, y, a] = \sum_{z=1}^{Z} \sum_{w=1}^{W} T[x, y, z, w]V[z, w, a]$$

This generalizes the familiar matrix product notation. Notice that T has multiple interpretations: it could *also* represent a linear map from tensors of shape (W, A) to tensors of shape (X, Y, Z, A) for example, or as a linear map from tensor of shape (W) to tensors of shape (X, Y, Z). This is analogous to the way that a $n \times m$ matrix can map vectors in $\mathbb{R}^m$ to $\mathbb{R}^n$, but also could be viewed as a way to map matrices in $\mathbb{R}^{m \times k}$ to matrices in $\mathbb{R}^{n \times k}$ via matrix products. This procedure of combining tensors to produce new tensors is called *tensor contraction*.

All of the preceding discussion of derivatives generalizes completely seamlessly to the tensor case. For example, if $f : \mathbb{R}^{Z \times W \times A} \to \mathbb{R}^{B \times C}$ is a function taking tensors as both input and output, then the derivative needs to be a *linear* function from $\mathbb{R}^{Z \times W \times A} \to \mathbb{R}^{B \times C}$. Such a linear function can always be represented by a tensor of shape $B \times C \times Z \times W \times A$, and has the formula[2]:

[2]In fact, *any* linear function taking tensors to tensors can be represented by a tensor of appropriate shape, just as any linear map of vector spaces can be represented by a matrix.

$$D_{f,x}[b, c, z, w, a] = \frac{\partial f(x)[b, c]}{\partial x[z, w, a]}$$

The chain rule holds exactly as before, only now the matrix-multiply is replaced with tensor contraction. Thus, the entire automatic differentiation procedure is completely identical: all partial derivatives simply are now tensors instead of matrices, and all matrix multiplies are tensor contractions.

8 Tuning Learning Rates

So far, we have discussed setting the learning rate η using what is sometimes called "oracle tuning". That is, the tuning is in some way magical: we do not actually know the various parameters Δ, H, G, σ or T that are required to set the learning rate, so in reality we probably cannot use these theoretically motivated learning rates. In practice, one usually simply guesses the learning rate. That is, you try out a few learning rates and see how well the resulting outputs behave on some test set. Then, you just hope that you tried a learning rate that is good.

- When tuning learning rates (or regularization constants or usually any real-valued parameter), it is practically effective to tune on a *logarithmic scale*. That is, you might test the learning rates $10, 1.0, 0.1, 0.01, 0.001$.
- It is a good idea to make sure that the "best" learning rate is not on the extreme edge of the range you tuned over.
- If you are tuning more than just the learning rate (e.g. also tuning a regularization constant), you might want to tune via *random search*. That is, choose some values $x_1, x_2, \dots$ at random from $[-3, 1]$ and test learning rate of 10^{x_i}. This is helpful for avoiding issue in which the learning rate and the regularization parameter have correlated effects (e.g, increasing regularization may be similar to decreasing learning rates).

8.1 Learning Rate Schedules

We have discussed setting the learning rate for SGD as $\propto \frac{1}{\sqrt{t}}$ for optimal theoretical bounds. However, in practice there are a wide range of options used. We'll just discuss a few prominent ones.

Many of these schedules are set based on the number of *epochs* that have elapsed, where an epoch is one whole pass over the training set. However, this is not always the case, as some datasets are so large that the total number of epochs may be very small (GPT3 was trained for less than a single epoch). In these cases, one might consider setting the schedule as a function of the number of iterations instead.

- **Exponential Decay**: In this schedule, after a certain number of epochs, we decrease the learning rate by a constant factor. For example, when training models on the ImageNet dataset for object recognition, it is common to employ SGD with a constant learning rate, and decrease the learning rate by a factor of 10 every 30 epochs for 90 total epochs.

  Intuitively, the idea is that after 30 epochs the optimization has converged roughly to the limit of how well it can converge using the given learning rate. By decreasing the learning rate, it is possible for the optimizer to explore higher-resolution features of the loss surface. Typically, one observes a rapid decrease in loss in the first iterations after decreasing the learning rate.

- **Cosine Annealing**. This schedule was introduced by Loshchilov and Hutter 2016 and uses the cosine function to set the learning rate:

$$\eta_t = \eta_{\min} + \frac{1}{2}\left(\eta_{\max} - \eta_{\min}\right)\left(1 + \cos\left(\frac{t}{T}\pi\right)\right)$$

  where $\eta_{\min}$ and $\eta_{\max}$ are some chosen minimum and maximum learning rates. It is also possible to combine cosine annealing with exponential decay. In this setup, one would divide the training into N phases. Let $T_1, \dots, T_N$ be the lengths of the phases. Then in phase i, we re-index the step-counter t so that phase i starts with $t = 1$, and use the learning rate:

$$\eta_t = \eta_{\min,i} + \frac{1}{2}\left(\eta_{\max,i} - \eta_{\min}\right)\left(1 + \cos\left(\frac{t}{T_i}\pi\right)\right)$$

  where $\eta_{\min,i}$ and $\eta_{\max,i}$ are potentially decaying learning rate ranges for each phase. This approach means that the learning rate will actually *increase* at the start of each phase, leading to a kind of oscillatory behavior in

the loss. Thus, one should take care to only return the parameters $\hat{w}$ that occur at the end of the last completed phase.

The cosine annealing schedule is quite effective empirically, but unfortunately there is essentially no good mathematical justification for it. While it is in some sense inspired by a notion of "restarting" that is useful in convex optimization, there is no formal connection, and indeed the use of the cosine function is essentially unheard of in convex optimization. It's also not clear if the cosine function itself is particularly important here.

- **Scaling learning rate when increasing batch size**. This rule is fairly simple: if you scale the batch-size by a factor B, but keep the same number of total gradient computations (i.e. *epochs*), then you should scale the learning rate also by a factor B.

  One of the main points of increasing the batch size is to train "faster" in the sense of spending less time by parallelizing the computation of the batch gradients. This is only useful if we keep the total computation constant, so if we keep the number of passes over the training set constant (each pass is called one "epoch"), then increasing the batch size from 1 to B is accompanied by a corresponsing *decrease* in the total number of SGD update from T to T/B. Thus, looking back at the previous lecture notes, we see that if the optimal learning rate with batch size 1 and T iterations is $O(1/\sqrt{T})$, then with batch size B and T/B iterations, the optimal learning rate is $O(\sqrt{B}/\sqrt{T/B}) = O(B/\sqrt{T})$ because the variance of the gradients goes down by a factor of B.

- **Warm-up**. See Goyal et al. 2017. This schedule is empirically motivated in the large batch-size regime. In this idea, when training with a very large batch size, one should actually start with a very small learning rate and slowly increase the learning rate during the earlier iterations. Once a certain small number of epochs has elapsed, then the learning rate is allowed to remain constant or decrease following some other more typical schedule. For example, it is common to apply *linear warm-up:*

$$\eta_t = \begin{cases} \frac{t\eta_0}{T_{\text{warm-up}}} & t \leq T_{\text{warm-up}} \\ \text{some other schedule (e.g. } \propto 1/\sqrt{t}) & t > T_{\text{warm-up}} \end{cases}$$

  Here η_0 is some base or "target" learning rate, that we are slowly increasing η_t until this rate is reached. Afterwards, we follow some arbitrary other schedule, typically such that the first learning rate of the other schedule is η_0.

8.2 Some more technical intuition for warm-up

Let's try to gain some intuition behind why this might be a good idea. The essential claim is that this procedure roughly causes the iterates produced by SGD using a large batch of size B to be close to the iterates produced via SGD using a batch size of 1 but a B-times smaller learning rate. In the following, we'll flesh out this argument more formally. Note that we will *not* show a strong theorem here, instead we'll just try to provide a little intuition about what might be going on. So far as I am aware, a truly complete formal understanding of warm-up is not yet known.

We have already discussed how increasing the batch size should be generally accompanied by an increase in the learning rate, so that it seems reasonable that these two learning rate/batch size schemes should have similar guarantees about convergence to critical points.

Intuitively, however, there is an alternative way to justify scaling the learning rate when the batch size is increased. Instead of trying to reason about the function value $\mathcal{L}(\mathbf{w}_t)$, we could try to understand the actual iterates $\mathbf{w}_t$ themselves. Specifically, consider two scenarios, a batch-size of 1 and a learning rate of η, and a batch size of B and a learning rate $\eta^{(B)}$. Let $\mathbf{w}_t^{(1)}$ be the tth iterate with batch size 1 and $\mathbf{w}_t^{(B)}$ be the tth iterate with batch size B. Suppose both batch sizes start at the same initial point $\mathbf{w}_1$. Then after B steps with batch-size 1, we have:

$$\mathbf{w}_{B+1}^{(1)} = \mathbf{w}_1 - \eta \sum_{t=1}^{B} \nabla \ell(\mathbf{w}_t^{(1)}, z_t)$$

while after 1 step of batch-size B,

$$\mathbf{w}_2^{(B)} = \mathbf{w}_1 - \eta^{(B)} \frac{1}{B} \sum_{t=1}^{B} \nabla \ell(\mathbf{w}_1, z_t)$$

Now, if it happened to be true that $\nabla \ell(\mathbf{w}_1, z_t) \approx \nabla \ell(\mathbf{w}_t^{(1)}, z_t)$, and $\eta^{(B)} = B\eta$, then we would have $\mathbf{w}_{B+1}^{(1)} \approx \mathbf{w}_2^{(B)}$, so that after B gradient evaluations, the two procedures have reached nearly the same point.

Let's try to characterize when this happens more formally. We already should suspect that the learning rate should never exceed $\frac{1}{H}$ when $\mathcal{L}$ is an H-smooth function. Let's assume a slightly stronger statement: $\eta^{(B)} \leq \frac{1}{2H}$ so that $\eta^{(1)} \leq \frac{1}{2BH}$. Further, define $r_t^{(1)} = \nabla \ell(\mathbf{w}_t^{(1)}, z_t) - \nabla \mathcal{L}(\mathbf{w}_t^{(1)})$. Notice that $\mathbb{E}[r_t^{(1)}] = 0$. Let's also suppose $\mathbb{E}[\|r_t^{(1)}\|^2] \leq \sigma^2$ for some σ. Similarly, define $r_t^{(B)} = \nabla \ell(\mathbf{w}_1, z_t) - \nabla \mathcal{L}(\mathbf{w}_1)$, and assume $\mathbb{E}[\|r_t^{(B)}\|^2] \leq \sigma^2$. Now we can write:

$$\begin{aligned}
\mathbf{w}_{B+1}^{(1)} - \mathbf{w}_1 &= -\eta \sum_{t=1}^{B} \nabla \ell(\mathbf{w}_t^{(1)}, z_t) \\
&= -\eta \sum_{t=1}^{B} (\nabla \ell(\mathbf{w}_t^{(1)}, z_t) - \nabla \mathcal{L}(\mathbf{w}_t^{(1)})) + (\nabla \mathcal{L}(\mathbf{w}_t^{(1)}) - \nabla \mathcal{L}(\mathbf{w}_1)) + \nabla \mathcal{L}(\mathbf{w}_1) \\
&= -\eta B \nabla \mathcal{L}(\mathbf{w}_1) - \eta \sum_{t=1}^{B} r_t^{(1)} - \eta \sum_{t=1}^{B} (\nabla \mathcal{L}(\mathbf{w}_t^{(1)}) - \nabla \mathcal{L}(\mathbf{w}_1))
\end{aligned}$$

Also, we have:

$$\begin{aligned}
\mathbf{w}_2^{(B)} - \mathbf{w}_1 &= -\eta^{(B)} \frac{1}{B} \sum_{t=1}^{B} \nabla \ell(\mathbf{w}_1, z_t) \\
&= -\eta \sum_{t=1}^{B} \nabla \mathcal{L}(\mathbf{w}_1) + r_t^{(B)} \\
&= e - \eta B \nabla \mathcal{L}(\mathbf{w}_1) - \eta \sum_{t=1}^{B} r_t^{(B)}
\end{aligned}$$

Thus, we subtract to find the distance between $\mathbf{w}_{B+1}^{(1)}$ and $\mathbf{w}_2^{(B)}$:

$$\mathbf{w}_{B+1}^{(1)} - \mathbf{w}_2^{(B)} = -\eta \sum_{t=1}^{B} (r_t^{(1)} - r_t^{(B)}) - \eta \sum_{t=1}^{B} (\nabla \mathcal{L}(\mathbf{w}_t^{(1)}) - \nabla \mathcal{L}(\mathbf{w}_1))$$

Now, let us define $p_t = r_t^{(1)} - r_t^{(B)}$. Notice that $\mathbb{E}[p_t] = 0$ and $\mathbb{E}[\|p_t\|^2] \leq \mathbb{E}[2\|r_t^{(1)}\|^2 + 2\|r_t^{(B)}\|^2] \leq 4\sigma^2$. Then, we have shown:

$$\begin{aligned}
\mathbf{w}_{B+1}^{(1)} - \mathbf{w}_2^{(B)} &= -\eta \left[\sum_{t=1}^{B} p_t + \sum_{t=1}^{B} (\nabla \mathcal{L}(\mathbf{w}_t^{(1)}) - \nabla \mathcal{L}(\mathbf{w}_1)) \right] \\
\|\mathbf{w}_{B+1}^{(1)} - \mathbf{w}_2^{(B)}\| &\leq \eta \left[\left\| \sum_{t=1}^{B} p_t \right\| + \sum_{t=1}^{B} \|\nabla \mathcal{L}(\mathbf{w}_t^{(1)}) - \nabla \mathcal{L}(\mathbf{w}_1)\| \right] \\
&\leq \eta \left[\left\| \sum_{t=1}^{B} p_t \right\| + H \sum_{t=1}^{B} \|\mathbf{w}_t^{(1)} - \mathbf{w}_1\| \right]
\end{aligned}$$

Now, observe that we have p_t are mean zero, variance at most $4\sigma^2$ and also p_t and $p_{t'}$ are uncorrelated for any $t \neq t'$. Therefore:

$$\mathbb{E}\left[\left\|\sum_{t=1}^{B} p_t\right\|\right] \leq \sqrt{\mathbb{E}\left[\left\|\sum_{t=1}^{B} p_t\right\|^2\right]} \leq 2\sigma\sqrt{B}$$

Thus:

$$\mathbb{E}[\|\mathbf{w}_{B+1}^{(1)} - \mathbf{w}_2^{(B)}\|] \leq 2\sigma\eta\sqrt{B} + \eta H\, \mathbb{E}\left[\sum_{t=1}^{B} \|\mathbf{w}_t^{(1)} - \mathbf{w}_1\|\right]$$

Now, let's turn to bounding $\mathbb{E}\left[\sum_{t=1}^{B} \|\mathbf{w}_t^{(1)} - \mathbf{w}_1\|\right]$. By previous arguments, we have:

$$\mathbf{w}_t^{(1)} - \mathbf{w}_1 = -\eta(t-1)\nabla\mathcal{L}(\mathbf{w}_1) - \eta\sum_{t'=1}^{t-1} r_{t'}^{(1)} - \eta\sum_{t'=1}^{t-1}(\nabla\mathcal{L}(\mathbf{w}_{t'}^{(1)}) - \nabla\mathcal{L}(\mathbf{w}_1))$$

$$\|\mathbf{w}_t^{(1)} - \mathbf{w}_1\| \leq \eta(t-1)\|\nabla\mathcal{L}(\mathbf{w}_1)\| + \eta\left\|\sum_{t'=1}^{t-1} r_{t'}^{(1)}\right\| + \eta\sum_{t'=1}^{t-1} H\|\mathbf{w}_{t'}^{(1)} - \mathbf{w}_1\|$$

summing over t:

$$\sum_{t=1}^{B}\|\mathbf{w}_t^{(1)} - \mathbf{w}_1\| \leq \frac{\eta B(B-1)}{2}\|\nabla\mathcal{L}(\mathbf{w}_1)\| + \eta\sum_{t=1}^{B}\left\|\sum_{t'=1}^{t-1} r_{t'}^{(1)}\right\| + \eta HB\left(\sum_{t=1}^{B}\|\mathbf{w}_t^{(1)} - \mathbf{w}_1\|\right)$$

Using $\eta \leq \frac{1}{2BH}$:

$$\leq \frac{\eta B(B-1)}{2}\|\nabla\mathcal{L}(\mathbf{w}_1)\| + \eta\sum_{t=1}^{B}\left\|\sum_{t'=1}^{t-1} r_{t'}^{(1)}\right\| + \frac{1}{2}\sum_{t=1}^{B}\|\mathbf{w}_t^{(1)} - \mathbf{w}_1\|$$

$$\sum_{t=1}^{B}\|\mathbf{w}_t^{(1)} - \mathbf{w}_1\| \leq \eta B(B-1)\|\nabla\mathcal{L}(\mathbf{w}_1)\| + 2\eta\sum_{t=1}^{B}\left\|\sum_{t'=1}^{t-1} r_{t'}^{(1)}\right\|$$

$$\mathbb{E}\left[\sum_{t=1}^{B}\|\mathbf{w}_t^{(1)} - \mathbf{w}_1\|\right] \leq \eta B^2\|\nabla\mathcal{L}(\mathbf{w}_1)\| + 4\eta\sigma B^{3/2}$$

where in the last line, we have use $\mathbb{E}\left[\left\|\sum_{t'=1}^{t-1} r_{t'}^{(1)}\right\|\right] \leq 2\sigma\sqrt{t}$ and $\sum_{t=1}^{B}\sqrt{t} \leq 2B^{3/2}$. To see the latter identity, notice that since $\sqrt{x}$ is an increasing function:

$$\int_t^{t+1}\sqrt{x}\,dx \geq \sqrt{t}$$

So that

$$\sum_{t=1}^{B}\sqrt{t} \leq \int_1^{B+1}\sqrt{x}\,dx = \frac{2}{3}(B+1)^{3/2} - \frac{2}{3}$$
$$\leq 2B^{3/2}$$

Therefore:

$$\mathbb{E}[\|\mathbf{w}_{B+1}^{(1)} - \mathbf{w}_2^{(B)}\|] \leq 2\sigma\eta\sqrt{B} + \eta H\,\mathbb{E}\left[\sum_{t=1}^{B}\|\mathbf{w}_t^{(1)} - \mathbf{w}_1\|\right]$$
$$\leq 2\sigma\eta\sqrt{B} + \eta^2 HB^2\|\nabla\mathcal{L}(\mathbf{w}_1)\| + 4\eta^2 H\sigma B^{3/2}$$

Using $\eta \leq \frac{1}{2HB}$:

$$\leq 4\sigma\eta\sqrt{B} + \frac{\eta B}{2}\|\nabla\mathcal{L}(\mathbf{w}_1)\|$$
$$\leq \frac{2\sigma}{H\sqrt{B}} + \frac{\eta B}{2}\|\nabla\mathcal{L}(\mathbf{w}_1)\|$$

Now, the first term above is $O(1/\sqrt{B})$, and so is actually small as B increases. However, the second term may be as large as $O(\|\nabla\mathcal{L}(\mathbf{w}_1)\|)$ and so could actually be quite large, *unless* $\|\nabla\mathcal{L}(\mathbf{w}_1)\|$ *is somewhat small*.

This provides some alternative view of linearly scaling the learning rate: if the gradient is already somewhat small, then we should expect that B steps of batch-size 1 is approximately *equal* to 1 step of batch size B. If the gradient is *not* small, this will only be true if we set the learning rate much smaller than the $\frac{1}{H}$ limit suggested by simply looking at the function decrease calculations we performed previously.

As a result, in the earlier iterations, when we might intuitively expect $\nabla\mathcal{L}(\mathbf{w})$ to be large, we might try using an artificially small learning rate instead of $\eta^{(B)} = \frac{1}{2H}$, and slowly increase as $\nabla\mathcal{L}(\mathbf{w})$ (hopefully) decreases, until we reach a point when $\nabla\mathcal{L}(\mathbf{w})$ is small enough that we can use our original target learning rate of $\frac{1}{2H}$.

9 Adaptive Learning Rate

Recall that we adopt the notation:

$$\mathbf{g}_t = \nabla \ell(\mathbf{w}_t, z_t)$$

to make the equations look a little simpler.

Previously, we saw that the "optimal" learning rate for SGD is roughly $\min\left(\frac{1}{H}, \frac{\sqrt{\Delta}}{\sigma\sqrt{TH}}\right)$. This is nice in theory, but in reality you probably don't actually know the values for H or Δ or σ or even T. We saw that it's possible to avoid needing to know T via a varying learning rate of $\eta_t = O(1/\sqrt{t})$ at the cost of a logarithmic factor, but how would we be able to deal with the other unknown values?

This is objective of *adaptive* algorithms. Adaptive algorithms "adapt" to the problem at hand, achieving faster convergence on "easier" problems while gracefully degrading on harder problems. Concretely, we can think of adaptive algorithms as figuring out some parameters of the problem (such as σ of H) "on-the-fly".

In the *convex* setting, this area has been heavily studied, and there are very sophisticated adaptive algorithms available that can adapt to all sorts of different types of "easy" problems. See Elad Hazan, Rakhlin, and Bartlett 2008; J. Duchi, E. Hazan, and Singer 2010; Erven and Koolen 2016; Cutkosky and Orabona 2018 for some examples of such algorithms. In the non-convex setting, however, things are a little less developed. Today we will discuss a way to adapt well to σ, and to some extent H. There is still no good way to adapt to Δ.

9.1 Intuition for Adaptive Learning Rates

Let's take a look at the learning rate setting we saw previously:

$$\eta = \min\left(\frac{1}{H}, \frac{\sqrt{\Delta}}{\sigma\sqrt{TH}}\right)$$

A simple way to try to design an adaptive algorithm is to "guess" this value of η on the fly. To start, let's make a simplification: for large enough T, we should always expect the $1/\sqrt{T}$ part to be dominant, so let's focus on estimating $\sigma\sqrt{T}$.

Ignoring the other constants for now, we want to make a learning rate like

$$\eta_t = \frac{c}{\sigma\sqrt{t}}$$

So now we need only estimate σ. How can we estimate the variance? The simplest method might be to try some form of empirical estimate. Since the variance of a random quantity Z is $\mathbb{E}[\|Z\|^2 - \|\mathbb{E}[Z]\|^2]$, we could try to estimate σ^2 by:

$$\sigma^2 \approx \frac{1}{t}\sum_{i=1}^{t} \|\mathbf{g}_i\|^2 - \left\|\frac{1}{t}\sum_{i=1}^{t}\mathbf{g}_i\right\|^2$$

Or, we could use the slightly more advance *unbiased* estimate:

$$\overline{\mathbf{g}} = \frac{1}{t}\sum_{i=1}^{t}\mathbf{g}_i \sigma^2 \approx \frac{1}{t-1}\sum_{i=1}^{t}\|\mathbf{g}_i - \overline{\mathbf{g}}\|^2$$

where here $\mathbf{g}_i = \nabla\ell(\mathbf{w}_i, z_i)$, the ith output of the stochastic gradient oracle.

However, it turns out that it will be much simpler algebraically to analyze instead the following estimate:

$$\sigma^2 \approx \frac{1}{t}\sum_{i=1}^{t}\|\mathbf{g}_i\|^2$$

This is a kind of "optimistic" guess for σ^2: if the gradients $\nabla\mathcal{L}(\mathbf{w}_t)$ are getting smaller, we might expect $\frac{1}{t}\sum_{i=1}^t \mathbf{g}_i$ to also be small, and so these two estimates are nearly the same. This leaves us with the learning rate:

$$\eta_t = \frac{c}{\sqrt{\sum_{i=1}^t \|\mathbf{g}_i\|^2}}$$

This is almost identical to the popular AdaGrad learning rate J. Duchi, E. Hazan, and Singer 2010 that we will discuss more later.

Strangely, while the above learning rate can analyzed without too much struggle in the *convex* setting, in the non-convex setting there is a technical issue in the algebra that makes it surprisingly difficult to work with. In order to appreciate this technical issue, let us start our performing the analysis and see what might go wrong.

We might think to start with our standard bounds on smooth losses (e.g Theorem 4.6), which tell us that:

$$\begin{aligned}\mathbb{E}[\mathcal{L}(\mathbf{w}_{t+1})] &\leq \mathbb{E}[\mathcal{L}(\mathbf{w}_t) - \langle\nabla\mathcal{L}(\mathbf{w}_t), \eta_t\mathbf{g}_t\rangle + \frac{H\eta_t^2\|\mathbf{g}_t\|^2}{2}] \\ &= \mathbb{E}[\mathcal{L}(\mathbf{w}_t)] - \mathbb{E}[\eta_t\langle\nabla\mathcal{L}(\mathbf{w}_t), \mathbf{g}_t\rangle] + \mathbb{E}[\frac{H\eta_t^2\|\mathbf{g}_t\|^2}{2}]\end{aligned}$$

Now, in the past we have simplified the term $-\mathbb{E}[\eta_t\langle\nabla\mathcal{L}(\mathbf{w}_t), \mathbf{g}_t\rangle]$ as $-\eta_t\,\mathbb{E}[\langle\nabla\mathcal{L}(\mathbf{w}_t), \mathbf{g}_t\rangle] = -\eta_t\,\mathbb{E}[\|\nabla\mathcal{L}(\mathbf{w}_t)\|^2]$. However, this was only possible because η_t was a deterministic quantity. In our current setting, η_t actually depends on $\mathbf{g}_t$, and so we cannot simply pull it out of the expectation. In fact, X. Li and Orabona 2019 shows that is might actually be that $-\mathbb{E}[\eta_t\langle\nabla\mathcal{L}(\mathbf{w}_t), \mathbf{g}_t\rangle] > 0$, which suggests that the gradient step might actually make *negative* progress!

There are roughly two ways to deal with this. Probably the simplest way, as suggested in X. Li and Orabona 2019, is to simply change the learning rate to:

$$\eta_t = \frac{c}{\sqrt{\epsilon^2 + \sum_{i=1}^{t-1} \|\mathbf{g}_i\|^2}}$$

Note the two major differences here: the presence of ϵ, and the fact that the sum now only goes up to $t-1$. The ϵ is a *small* constant that is now necessary because otherwise $\eta_1 = \infty$. The sum going only up to $t-1$ fixes the problem with expectations, because now η_t is independent of z_t. As a result, we would have:

$$\begin{aligned}\mathbb{E}[\langle\nabla\mathcal{L}(\mathbf{w}_t), \eta_t\nabla\ell(\mathbf{w}_t, z_t)\rangle] &= \mathbb{E}[\eta_t \mathop{\mathbb{E}}_{z_t}[\langle\nabla\mathcal{L}(\mathbf{w}_t), \nabla\ell(\mathbf{w}_t, z_t)\rangle|z_t]] \\ &= \mathbb{E}[\eta_t\|\nabla\mathcal{L}(\mathbf{w}_t)\|^2]\end{aligned}$$

However, this is not the learning rate used in practice, so we will instead consider the following rate:

$$\eta_t = \frac{c}{\sqrt{\epsilon^2 + \sum_{i=1}^{t} \|\mathbf{g}_i\|^2}}$$

Notice that while the ϵ is still there, the sum now again goes up to t, so that the above argument with expectations is *invalid*. In the convex setting, the ϵ is not needed, but we will see that it actually plays a role in our analysis to get around the expectation issues. In practice, the ϵ is usually added to avoid numerical instability issues: when computing η_t, it might be that the first gradient is zero or very small simply by chance. As a result, we would have $\eta_t = \infty$. Now, technically, we always have:

$$\|\eta_t\mathbf{g}_t\| \leq c$$

so that it should not matter if $\mathbf{g}_1 = 0$, but when actually implementing this we will run into numerical issues (in the worst case, you will get a divide by zero error), so the small ϵ is added to the denominator to avoid this. It just happens as a happy coincidence to help in the theory as well.

10 Adaptive SGD Algorithm

So, to summarize, we are considering the following Algorithm:

We are able to prove the following general Theorem. Notice that this result applies to *any* sequence of learning rates, not just the ones in Algorithm 9:

Algorithm 9 Adaptive Stochastic Gradient Descent

Input: Initial Point $\mathbf{w}_1$, learning rates scaling c, small constant ϵ (e.g. 1e-4).
for $t = 1 \dots T$ **do**
  Sample $z_t \sim P_z$.
  Set $\mathbf{g}_t = \nabla \ell(\mathbf{w}_t, z_t)$.
  Set $\eta_t = \frac{c}{\sqrt{\epsilon^2 + \sum_{i=1}^t \|\mathbf{g}_i\|^2}}$
  Set $\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t \mathbf{g}_t$.
end for

Theorem 10.1. *Suppose $\mathcal{L}$ is an H-smooth function. Defining $\mathbf{g}_t = \nabla \ell(\mathbf{w}_t, z_t)$, suppose that $\mathbb{E}[\max_{t \le T} \|\mathbf{g}_t\|] \le G_\ell$ (note this is true if $\|\nabla \ell(\mathbf{w}_t, z_t)\| \le G_\ell$ with probability 1, but would be also implied by more technical conditions such as subgaussianity). Further, suppose that $\mathcal{L}$ is $G_{\mathcal{L}}$-Lipschitz. Finally, let $\eta_1, \dots, \eta_T$ be any sequence of learning rates such that (1) $\eta_t \ge 0$ for all t, (2) $\eta_1 \ge \eta_2 \ge \dots \ge \eta_T$ and also (3) the sequence is "causal" in the sense that η_t is not allowed to depend on $\mathbf{g}_{t+1}, \mathbf{g}_{t+2}, \dots, \mathbf{g}_T$. Let η_0 be a deterministic quantity such that $\eta_0 \ge \eta_1$. Consider the SGD update:*

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t \mathbf{g}_t$$

Then we have:

$$\mathbb{E}[\mathcal{L}(\mathbf{w}_{T+1})] \le \mathbb{E}\left[\mathcal{L}(\mathbf{w}_1) - \sum_{t=1}^T \eta_{t-1} \|\nabla \mathcal{L}(\mathbf{w}_t)\|^2 + \frac{H}{2} \sum_{t=1}^T \eta_t^2 \|\mathbf{g}_t\|^2\right] + \eta_0 G_{\mathcal{L}} G_\ell$$

This Theorem is a somewhat technical-looking result, so before we prove it, let's see what it implies about the adaptive learning rate scheme we are using in Algorithm 9. To start with, notice that this learning rate scheme satisfies the conditions of Theorem 10.1: the learning rates are monotonically decreasing, and η_t does not depend on $\mathbf{g}_{t+1}$ or later values of $\mathbf{g}_t$. Further, we can set $\eta_0 = \frac{c}{\epsilon}$. For simplicity, let us suppose that $\|\mathbf{g}_t\| \le G$ with probability 1 and just take $G_{\mathcal{L}} = G_\ell = G$. Thus, Theorem 10.1 tells us that:

$$\mathbb{E}[\mathcal{L}(\mathbf{w}_{T+1})] \le \mathbb{E}\left[\mathcal{L}(\mathbf{w}_1) - \sum_{t=1}^T \eta_t \|\nabla \mathcal{L}(\mathbf{w}_t)\|^2 + \frac{H}{2} \sum_{t=1}^T \eta_t^2 \|\mathbf{g}_t\|^2\right] + \frac{cG^2}{\epsilon}$$

Now, let us do some back-of-the-envelope calculations to gain intuition for what this means. Note that these are *NOT CORRECT*, but will be "nearly correct", and we will make the arguments rigorous later.

First, let's do some rearrangement and use the fact $\eta_T \le \eta_t$ for all t:

$$\mathbb{E}\left[\sum_{t=1}^T \eta_T \|\nabla \mathcal{L}(\mathbf{w}_t)\|^2\right] \le \mathbb{E}\left[\Delta + \frac{H}{2} \sum_{t=1}^T \eta_t^2 \|\mathbf{g}_t\|^2\right] + \frac{cG^2}{\epsilon}$$

Next, let's consider the sum

$$\sum_{t=1}^T \eta_t^2 \|\mathbf{g}_t\|^2 = c^2 \sum_{t=1}^T \frac{\|\mathbf{g}_t\|^2}{\epsilon^2 + \sum_{i=1}^t \|\mathbf{g}_t\|^2}$$

We will soon see that for roughly the same reasons that $\sum_{t=1}^T \frac{1}{t} \le O(\log(T))$, this sum is $O(c^2 \log(T/\epsilon^2))$. Let's just pretend it is actually bounded by $c^2 \log(T/\epsilon^2)$ for now. Then, we perform another technically-illegal operation: let's divide by η_T inside the expectations:

$$\begin{aligned}
\mathbb{E}\left[\sum_{t=1}^T \|\nabla \mathcal{L}(\mathbf{w}_t)\|^2\right] &\le \mathbb{E}\left[\frac{\Delta}{\eta_T} + \frac{c^2 \log(T/\epsilon^2)}{\eta_T} + \frac{cG^2}{\epsilon \eta_T}\right] \\
&= \mathbb{E}\left[\left(\frac{\Delta}{c} + c \log(T/\epsilon^2) + \frac{G^2}{\epsilon}\right) \sqrt{\epsilon^2 + \sum_{t=1}^T \|\mathbf{g}_t\|^2}\right]
\end{aligned}$$

Now, let's again make a leap of faith and assume that it will actually hold that

$$\mathbb{E}\left[\sum_{t=1}^{T}\|\nabla\mathcal{L}(\mathbf{w}_t)\|^2\right] = O(\sigma\log(T)\sqrt{T})$$

where $\sigma^2 \geq \mathbb{E}[\|\mathbf{g}_t - \nabla\mathcal{L}(\mathbf{w}_t)\|^2]$ for all t. Note that this is definitely not logically okay - we would need to prove this. But just to get some intuition, let's make this assumption and at least see that we don't hit a contradiction. Now, we have (using the bias-variance decomposition Proposition 10.4):

$$\begin{aligned}\mathbb{E}\left[\sum_{t=1}^{T}\|\mathbf{g}_t\|^2\right] &= \sum_{t=1}^{T}\mathbb{E}[\|\nabla\mathcal{L}(\mathbf{w}_t)\|^2] + \mathbb{E}[\|\mathbf{g}_t - \nabla\mathcal{L}(\mathbf{w}_t)\|^2] \\ &= O(\sigma\log(T)\sqrt{T} + \sigma^2 T) = O(\sigma^2 T)\end{aligned}$$

where we've assume T is sufficiently large that $\sigma\log(T)\sqrt{T} \leq \sigma^2 T$. Further, by Jensen inequality:

$$\begin{aligned}\mathbb{E}\left[\sqrt{\epsilon^2 + \sum_{t=1}^{T}\|\mathbf{g}_t\|^2}\right] &\leq \sqrt{\epsilon^2 + \mathbb{E}\left[\sum_{t=1}^{T}\|\mathbf{g}_t\|^2\right]} \\ &\leq \epsilon + O\left(\sigma\sqrt{T}\right)\end{aligned}$$

from which we are able to recover our assumption $\mathbb{E}\left[\sum_{t=1}^{T}\|\nabla\mathcal{L}(\mathbf{w}_t)\|^2\right] = O(\sigma\log(T)\sqrt{T})$ by absorbing Δ, G and ϵ into the big-Oh notation.

Now that we have this sketch for how to use the Theorem, let's proceed to prove Theorem 10.1.

Proof of Theorem 10.1. As usual, be we begin by using our key result about smooth functions:

$$\mathcal{L}(\mathbf{w}_{t+1}) \leq \mathcal{L}(\mathbf{w}_t) - \eta_t\langle\nabla\mathcal{L}(\mathbf{w}_t), \mathbf{g}_t\rangle + \frac{H}{2}\eta_t^2\|\mathbf{g}_t\|^2$$

now, when we take expectations, the $\eta_t\langle\nabla\mathcal{L}(\mathbf{w}_t), \mathbf{g}_t\rangle$ will cause trouble since η_t depends on $\mathbf{g}_t$. So let's instead write in terms of $\eta_{t-1}\langle\nabla\mathcal{L}(\mathbf{w}_t), \mathbf{g}_t\rangle$:

$$= \mathcal{L}(\mathbf{w}_t) - \eta_{t-1}\langle\nabla\mathcal{L}(\mathbf{w}_t), \mathbf{g}_t\rangle + (\eta_{t-1} - \eta_t)\langle\nabla\mathcal{L}(\mathbf{w}_t), \mathbf{g}_t\rangle + \frac{H}{2}\eta_t^2\|\mathbf{g}_t\|^2$$

Iterating for all t:

$$\mathcal{L}(\mathbf{w}_{T+1}) \leq \mathcal{L}(\mathbf{w}_1) - \sum_{t=1}^{T}\eta_{t-1}\langle\nabla\mathcal{L}(\mathbf{w}_t), \mathbf{g}_t\rangle + \sum_{t=1}^{T}(\eta_{t-1} - \eta_t)\langle\nabla\mathcal{L}(\mathbf{w}_t), \mathbf{g}_t\rangle + \sum_{t=1}^{T}\frac{H}{2}\eta_t^2\|\mathbf{g}_t\|^2$$

using the fact that η_t is decreasing:

$$\leq \mathcal{L}(\mathbf{w}_1) - \sum_{t=1}^{T}\eta_{t-1}\langle\nabla\mathcal{L}(\mathbf{w}_t), \mathbf{g}_t\rangle + \max_{t\leq T}|\langle\nabla\mathcal{L}(\mathbf{w}_t), \mathbf{g}_t\rangle|\sum_{t=1}^{T}(\eta_{t-1} - \eta_t) + \sum_{t=1}^{T}\frac{H}{2}\eta_t^2\|\mathbf{g}_t\|^2$$

Using Cauchy-Schwarz:

$$\begin{aligned}&\leq \mathcal{L}(\mathbf{w}_1) - \sum_{t=1}^{T}\eta_{t-1}\langle\nabla\mathcal{L}(\mathbf{w}_t), \mathbf{g}_t\rangle + \max_{t\leq T}\|\nabla\mathcal{L}(\mathbf{w}_t)\|\max_{t\leq T}\|\mathbf{g}_t\|\sum_{t=1}^{T}(\eta_{t-1} - \eta_t) + \sum_{t=1}^{T}\frac{H}{2}\eta_t^2\|\mathbf{g}_t\|^2 \\ &\leq \mathcal{L}(\mathbf{w}_1) - \sum_{t=1}^{T}\eta_{t-1}\langle\nabla\mathcal{L}(\mathbf{w}_t), \mathbf{g}_t\rangle + \max_{t\leq T}\|\nabla\mathcal{L}(\mathbf{w}_t)\|\max_{t\leq T}\|\mathbf{g}_t\|\eta_0 + \sum_{t=1}^{T}\frac{H}{2}\eta_t^2\|\mathbf{g}_t\|^2\end{aligned}$$

Taking expectations:

$$\mathbb{E}[\mathcal{L}(\mathbf{w}_{T+1})] \leq \mathbb{E}\left[\mathcal{L}(\mathbf{w}_1) - \sum_{t=1}^{T} \eta_{t-1}\langle \nabla \mathcal{L}(\mathbf{w}_t), \mathbf{g}_t\rangle + G_{\mathcal{L}} G_\ell \eta_0 + \sum_{t=1}^{T} \frac{H}{2}\eta_t^2 \|\mathbf{g}_t\|^2\right]$$

Finally, we can pull $G_{\mathcal{L}} G_\ell \eta_0$ out of the expectation since this is actually a constant. □

10.1 Convergence of Algorithm 9

Now that we have proven the technical result of Theorem 10.1, we can turn to formalizing the intuitive argument for how this implies a convergence result for Algorithm 9. In order to do this, we will need the following important technical Lemma:

Lemma 10.2. *Suppose $x_0, \ldots, x_T$ are arbitrary non-negative values. And let $f(x) : \mathbb{R} \to \mathbb{R}$ be an arbitary* decreasing *function. Then:*

$$\sum_{t=1}^{T} x_t f\left(\sum_{i=0}^{t} x_i\right) \leq \int_{x_0}^{\sum_{i=0}^{T} x_i} f(x)\, dx$$

Proof. Notice that since f is decreasing, for all intervals $[a, b]$ we have:

$$(b-a)f(b) \leq \int_a^b f(x)\, dx$$

In particular:

$$\begin{aligned} x_t f\left(\sum_{i=0}^{t} x_i\right) &\leq \int_{\sum_{i=0}^{t-1} x_i}^{\sum_{i=0}^{t} x_i} f(x)\, dx \\ \sum_{t=1}^{T} x_t f\left(\sum_{i=0}^{t} x_i\right) &\leq \sum_{t=1}^{T} \int_{\sum_{i=0}^{t-1} x_i}^{\sum_{i=0}^{t} x_i} f(x)\, dx \\ &= \int_{x_0}^{\sum_{i=0}^{T} x_i} f(x)\, dx \end{aligned}$$

□

As an immediate corollary of this Theorem, we have:

$$\begin{aligned} \sum_{t=1}^{T} \frac{\|\mathbf{g}_t\|^2}{\epsilon^2 + \sum_{i=1}^{t} \|\mathbf{g}_t\|^2} &\leq \int_{\epsilon^2}^{\sum_{t=1}^{T} \|\mathbf{g}_t\|^2} \frac{dx}{x} \\ &= \log\left(\frac{\sum_{t=1}^{T} \|\mathbf{g}_t\|^2}{\epsilon^2}\right) \end{aligned}$$

Now, we are ready to state and prove the following:

Theorem 10.3. *Suppose $\mathcal{L}$ is H-smooth and $G_{\mathcal{L}}$ Lipschitz, and let $\Delta = \mathcal{L}(\mathbf{w}_t) - \mathcal{L}(\mathbf{w}_\star)$. Suppose $\mathbb{E}[\max_{t \leq T} \|\mathbf{g}_t\|] \leq G_\ell$ and $\mathbb{E}[\|\mathbf{g}_t - \nabla \mathcal{L}(\mathbf{w}_t)\|^2] \leq \sigma^2$ for all t. Define*

$$K = \frac{\Delta}{c} + \frac{Hc \log\left(\frac{T(G_{\mathcal{L}}^2 + \sigma^2)}{\epsilon^2}\right)}{2} + \frac{G_{\mathcal{L}} G_\ell}{\epsilon} = O(\log(T))$$

Then Algorithm 9 guarantees:

$$\frac{1}{T}\,\mathbb{E}\left[\sum_{t=1}^{T} \|\nabla \mathcal{L}(\mathbf{w}_t)\|\right] \leq \frac{K\sqrt{2} + \sqrt{K\epsilon}}{\sqrt{T}} + \frac{\sqrt{K\sigma}}{T^{1/4}}$$

Proof. First, note that that by the bias-variance decomposition (Proposition 10.4), we have:

$$\begin{aligned}\mathbb{E}[\|\mathbf{g}_t\|^2] &\leq \mathbb{E}[\|\nabla\mathcal{L}(\mathbf{w}_t)\|^2] + \mathbb{E}[\|\mathbf{g}_t - \nabla\mathcal{L}(\mathbf{w}_t)\|^2] \\ &\leq \mathbb{E}[\|\nabla\mathcal{L}(\mathbf{w}_t)\|^2] + \sigma^2 \\ &\leq G_{\mathcal{L}}^2 + \sigma^2 \end{aligned} \tag{3}$$

Applying Theorem 10.1, with $\eta_0 = \frac{c}{\epsilon}$, we have:

$$\mathbb{E}[\mathcal{L}(\mathbf{w}_{T+1})] \leq \mathbb{E}\left[\mathcal{L}(\mathbf{w}_1) - \sum_{t=1}^{T}\eta_{t-1}\|\nabla\mathcal{L}(\mathbf{w}_t)\|^2 + \frac{H}{2}\sum_{t=1}^{T}\eta_t^2\|\mathbf{g}_t\|^2\right] + \frac{cG_{\mathcal{L}}G_\ell}{\epsilon}$$

Now, applying the definition of η_t and Lemma 10.2, we can see that

$$\begin{aligned}\sum_{t=1}^{T}\eta_t^2\|\mathbf{g}_t\|^2 &= c^2\sum_{t=1}^{T}\frac{\|\mathbf{g}_t\|^2}{\epsilon^2 + \sum_{i=1}^{t}\|\mathbf{g}_t\|^2} \\ &\leq c^2\log\left(\frac{\sum_{t=1}^{T}\|\mathbf{g}_t\|^2}{\epsilon^2}\right)\end{aligned}$$

Taking expectations and using Jensen inequality:

$$\mathbb{E}\left[\sum_{t=1}^{T}\eta_t^2\|\mathbf{g}_t\|^2\right] \leq c^2\log\left(\frac{\mathbb{E}[\sum_{t=1}^{T}\|\mathbf{g}_t\|^2]}{\epsilon^2}\right)$$

Using equation (3):

$$\leq c^2\log\left(\frac{T(G_{\mathcal{L}}^2 + \sigma^2)}{\epsilon^2}\right)$$

Thus, we have:

$$\mathbb{E}[\mathcal{L}(\mathbf{w}_{T+1})] \leq \mathbb{E}\left[\mathcal{L}(\mathbf{w}_1) - \sum_{t=1}^{T}\eta_{t-1}\|\nabla\mathcal{L}(\mathbf{w}_t)\|^2\right] + \frac{Hc^2\log\left(\frac{T(G_{\mathcal{L}}^2+\sigma^2)}{\epsilon^2}\right)}{2} + \frac{cG_{\mathcal{L}}G_\ell}{\epsilon}$$

rearranging:

$$\mathbb{E}\left[\sum_{t=1}^{T}\eta_{t-1}\|\nabla\mathcal{L}(\mathbf{w}_t)\|^2\right] \leq \mathbb{E}[\mathcal{L}(\mathbf{w}_1) - \mathcal{L}(\mathbf{w}_{T+1})] + \frac{Hc^2\log\left(\frac{T(G_{\mathcal{L}}^2+\sigma^2)}{\epsilon^2}\right)}{2} + \frac{cG_{\mathcal{L}}G_\ell}{\epsilon}$$

using $\eta_T \leq \eta_t$ for all t:

$$\mathbb{E}\left[\sum_{t=1}^{T}\eta_T\|\nabla\mathcal{L}(\mathbf{w}_t)\|^2\right] \leq \Delta + \frac{Hc^2\log\left(\frac{T(G_{\mathcal{L}}^2+\sigma^2)}{\epsilon^2}\right)}{2} + \frac{cG_{\mathcal{L}}G_\ell}{\epsilon}$$

Now, we come to a primary difficulty in the proof. In our sketch before, we just divided by η_T at this point. In previous arguments this was fine, because in all our previous algorithms, η_T was just a deterministic constant. Now, however, η_T is unfortunately a *random variable*, and so it is not allowed to divide it out from both sides. We will have to be a bit more tricky. The key idea is to use Cauchy-Schwarz for random variables (see Proposition 10.6). In particular, if we define random variables

$$\begin{aligned}A^2 &= \sum_{t=1}^{T}\eta_T\|\nabla\mathcal{L}(\mathbf{w}_t)\|^2 \\ B^2 &= \frac{1}{\eta_T}\end{aligned}$$

Then by Proposition 10.6, we have

$$\begin{aligned}
\mathbb{E}[AB] &\leq \sqrt{\mathbb{E}[A^2]\,\mathbb{E}[B^2]} \\
\frac{\mathbb{E}[AB]^2}{\mathbb{E}[B^2]} &\leq \mathbb{E}[A^2] \\
\frac{\mathbb{E}\left[\sqrt{\sum_{t=1}^T \|\nabla\mathcal{L}(\mathbf{w}_t)\|^2}\right]^2}{\mathbb{E}[\eta_T^{-1}]} &\leq \mathbb{E}\left[\sum_{t=1}^T \eta_T \|\nabla\mathcal{L}(\mathbf{w}_t)\|^2\right]
\end{aligned}$$

This is a somewhat magical trick: by doing a little manipulation with Proposition 10.6, we are able to get an expression in terms of $\mathbb{E}\left[\sqrt{\sum_{t=1}^T \|\nabla\mathcal{L}(\mathbf{w}_t)\|^2}\right]$, which *does not* have the η_T inside the expectation! So, continuing:

$$\begin{aligned}
\frac{\mathbb{E}\left[\sqrt{\sum_{t=1}^T \|\nabla\mathcal{L}(\mathbf{w}_t)\|^2}\right]^2}{\mathbb{E}[\eta_T^{-1}]} &\leq \Delta + \frac{Hc^2 \log\left(\frac{T(G_{\mathcal{L}}^2+\sigma^2)}{\epsilon^2}\right)}{2} + \frac{cG_{\mathcal{L}}G_\ell}{\epsilon} \\
\mathbb{E}\left[\sqrt{\sum_{t=1}^T \|\nabla\mathcal{L}(\mathbf{w}_t)\|^2}\right]^2 &\leq \left(\Delta + \frac{Hc^2 \log\left(\frac{T(G_{\mathcal{L}}^2+\sigma^2)}{\epsilon^2}\right)}{2} + \frac{cG_{\mathcal{L}}G_\ell}{\epsilon}\right)\mathbb{E}[\eta_T^{-1}] \\
&= \left(\frac{\Delta}{c} + \frac{Hc \log\left(\frac{T(G_{\mathcal{L}}^2+\sigma^2)}{\epsilon^2}\right)}{2} + \frac{G_{\mathcal{L}}G_\ell}{\epsilon}\right)\mathbb{E}\left[\sqrt{\epsilon^2 + \sum_{t=1}^T \|\mathbf{g}_t\|^2}\right]
\end{aligned}$$

Now, to avoid carrying around too much algebra, let's just define $K = \frac{\Delta}{c} + \frac{Hc \log\left(\frac{T(G_{\mathcal{L}}^2+\sigma^2)}{\epsilon^2}\right)}{2} + \frac{G_{\mathcal{L}}G_\ell}{\epsilon}$. Then, we can write this as:

$$\mathbb{E}\left[\sqrt{\sum_{t=1}^T \|\nabla\mathcal{L}(\mathbf{w}_t)\|^2}\right]^2 \leq K\,\mathbb{E}\left[\sqrt{\epsilon^2 + \sum_{t=1}^T \|\mathbf{g}_t\|^2}\right]$$

The next part of the proof is an important trick that frequently arises when analyzing adaptive algorithms: we are going to use a *self-bounding argument*. The idea is that if we define $X = \mathbb{E}\left[\sqrt{\sum_{t=1}^T \|\nabla\mathcal{L}(\mathbf{w}_t)\|^2}\right]$, which is the quantity that we are interested in bounding, we will try to obtain an expression of the form:

$$X^2 \leq f(X)$$

for some function f. Then, we will use the shape of the function f to argue that if $X^2 \leq f(X)$, X cannot be too large. Specifically, $f(X)$ will eventually be a linear function, so we will have $X^2 \leq aX + b$, from which we can use the quadratic formula to bound X. So far, we have obtained:

$$X^2 \leq K\,\mathbb{E}\left[\sqrt{\epsilon^2 + \sum_{t=1}^T \|\mathbf{g}_t\|^2}\right] \tag{4}$$

So, now we need to find some way to see X in the RHS of this expression. Our starting point is the identity $\|a+b\|^2 \leq 2\|a\|^2 + 2\|b\|^2$. To see this identity, observe $\|a+b\|^2 = \|a\|^2 + \|b\|^2 + 2\langle a, b\rangle$, and then apply Young inequality (Proposition 10.5) to bound $2\langle a, b\rangle \leq \|a\|^2 + \|b\|^2$. From this, we have:

$$\|\mathbf{g}_t\|^2 = \|\mathbf{g}_t - \nabla\mathcal{L}(\mathbf{w}_t) + \nabla\mathcal{L}(\mathbf{w}_t)\|^2 \leq 2\|\nabla\mathcal{L}(\mathbf{w}_t)\|^2 + 2\|\mathbf{g}_t - \nabla\mathcal{L}(\mathbf{w}_t)\|^2$$

Next, we use the identity $\sqrt{a+b} \leq \sqrt{a} + \sqrt{b}$. This can be seen by simply squaring both sides. Thus, we have:

$$\begin{aligned} K\,\mathbb{E}\left[\sqrt{\epsilon^2 + \sum_{t=1}^{T} \|\mathbf{g}_t\|^2}\right] &\leq K\,\mathbb{E}\left[\sqrt{\epsilon^2 + 2\sum_{t=1}^{T} \|\mathbf{g}_t - \nabla\mathcal{L}(\mathbf{w}_t)\|^2 + 2\sum_{t=1}^{T} \|\nabla\mathcal{L}(\mathbf{w}_t)\|^2}\right] \\ &\leq K\,\mathbb{E}\left[\sqrt{\epsilon^2 + 2\sum_{t=1}^{T} \|\mathbf{g}_t - \nabla\mathcal{L}(\mathbf{w}_t)\|^2}\right] + K\sqrt{2}\,\mathbb{E}\left[\sqrt{\sum_{t=1}^{T} \|\nabla\mathcal{L}(\mathbf{w}_t)\|^2}\right] \\ &= K\,\mathbb{E}\left[\sqrt{\epsilon^2 + 2\sum_{t=1}^{T} \|\mathbf{g}_t - \nabla\mathcal{L}(\mathbf{w}_t)\|^2}\right] + K\sqrt{2}X \end{aligned}$$

using Jensen inequality:

$$\begin{aligned} &\leq K\sqrt{\epsilon^2 + 2\,\mathbb{E}\left[\sum_{t=1}^{T} \|\mathbf{g}_t - \nabla\mathcal{L}(\mathbf{w}_t)\|^2\right]} + K\sqrt{2}X \\ &\leq K\sqrt{\epsilon^2 + T\sigma^2} + K\sqrt{2}X \end{aligned}$$

So, now going back to equation (4), we have:

$$X^2 \leq K\sqrt{\epsilon^2 + T\sigma^2} + K\sqrt{2}X$$

Now, by quadratic formula, we have:

$$\begin{aligned} X &\leq \frac{K\sqrt{2} + \sqrt{2K^2 + 4K\sqrt{\epsilon^2 + T\sigma^2}}}{2} \\ &\leq K\sqrt{2} + \sqrt{K}\left(\epsilon^2 + T\sigma^2\right)^{1/4} \\ &\leq K\sqrt{2} + \sqrt{K\epsilon} + \sqrt{K\sigma}T^{1/4} \end{aligned}$$

Now, finally, recall our definition of X and use Cauchy-Schwarz to conclude:

$$\begin{aligned} X &= \mathbb{E}\left[\sqrt{\sum_{t=1}^{T} \|\nabla\mathcal{L}(\mathbf{w}_t)\|^2}\right] \\ &\geq \frac{1}{\sqrt{T}}\,\mathbb{E}\left[\sum_{t=1}^{T} \|\nabla\mathcal{L}(\mathbf{w}_t)\|\right] \end{aligned}$$

From which we can conclude:

$$\frac{1}{T}\,\mathbb{E}\left[\sum_{t=1}^{T} \|\nabla\mathcal{L}(\mathbf{w}_t)\|\right] \leq \frac{K\sqrt{2} + \sqrt{K\epsilon}}{\sqrt{T}} + \frac{\sqrt{K\sigma}}{T^{1/4}}$$

□

10.2 More refined analysis

The analysis we've presented has a somwhat poor dependence on ϵ: it is $O\left(\frac{\sqrt{\sigma}}{\sqrt{\epsilon}T^{1/4}}\right)$, which could be quite bad. However, it is possible to obtain a more refined bound that does not have this poor dependence on ϵ. See Ward, Wu, and Bottou 2019 for a proof that achieves this. Note their proof is along a slightly different lines: instead of bounding the $(\eta_{t-1} - \eta_t)\langle\nabla\mathcal{L}(\mathbf{w}_t), \mathbf{g}_t\rangle$ terms by using decreasing learning rates, as we did in Theorem 10.1, instead they carefully bound the gap $|\eta_{t-1} - \eta_t|$ to show that each term is quite small.

10.3 Auxiliary technical results (also in appendix)

Proposition 10.4. *[Bias-variance Decomposition] Suppose $X \in \mathbb{R}^d$ is some random variable. Then for any deterministic value Y:*

$$\mathbb{E}[\|X - Y\|^2] = \|Y - \mathbb{E}[X]\|^2 + \mathbb{E}[\|X - \mathbb{E}[X]\|^2]$$

Proof. Expanding the square we have:

$$\begin{aligned}\mathbb{E}[\|X - Y\|^2] &= \mathbb{E}[\|X - \mathbb{E}[X] + \mathbb{E}[X] - Y\|^2] \\ &= \mathbb{E}[\|Y - \mathbb{E}[X]\|^2] + \mathbb{E}[\|X - \mathbb{E}[X]\|^2] + 2\,\mathbb{E}[\langle X - \mathbb{E}[X], \mathbb{E}[X] - Y\rangle] \\ &= \|Y - \mathbb{E}[X]\|^2 + \mathbb{E}[\|X - \mathbb{E}[X]\|^2] + 2\langle \mathbb{E}[X - \mathbb{E}[X]], \mathbb{E}[X] - Y\rangle \\ &= \|Y - \mathbb{E}[X]\|^2 + \mathbb{E}[\|X - \mathbb{E}[X]\|^2]\end{aligned}$$

where □

Another useful inequality is the Young inequality:

Proposition 10.5. *[Young inequality] Suppose X and Y are arbitrary vectors. Then for any $\lambda > 0$ and any non-negative real numbers p and q satisfying $\frac{1}{p} + \frac{1}{q} = 1$,*

$$\langle X, Y\rangle \leq \frac{\|X\|^p}{p\lambda^p} + \frac{\lambda^q \|Y\|^q}{q}$$

Proof. Notice that we must have both p and q at least 1 (otherwise $1/p + 1/q > 1$, which is not allowed). Now, differentiating the RHS with respect to λ yields:

$$-\frac{\|X\|^p}{\lambda^{p+1}} + \lambda^{q-1}\|Y\|^q$$

Differentiating again yields:

$$(p+1)\frac{\|X\|^p}{\lambda^{p+2}} + (q-1)\lambda^{q-2}\|Y\|^q \geq 0$$

where we have used $q \geq 1$. Thus, we can find a minimum value for the RHS by setting the first derivative to 0. Solving $-\frac{\|X\|^p}{\lambda^{p+1}} + \lambda^{q-1}\|Y\|^q = 0$ for λ then gives $\lambda = \frac{\|X\|^{p/(p+q)}}{\|Y\|^{q/(p+q)}}$, which yields a minimum value of

$$\frac{\|X\|^{p-\frac{p^2}{p+q}}\|Y\|^{\frac{pq}{p+q}}}{p} + \frac{\|Y\|^{q-\frac{q^2}{p+q}}\|X\|^{\frac{pq}{p+q}}}{q} = \frac{\|X\|^{\frac{pq}{p+q}}\|Y\|^{\frac{pq}{p+q}}}{p} + \frac{\|Y\|^{\frac{pq}{p+q}}\|X\|^{\frac{pq}{p+q}}}{q}$$

Now, notice that

$$1 = \frac{1}{p} + \frac{1}{q} = \frac{p+q}{pq}$$

So that the minimum of the RHS simplifies to:

$$\frac{\|X\|\|Y\|}{p} + \frac{\|Y\|\|X\|}{q} = \|X\|\|Y\|$$

Now to conclude observe that $\langle X, Y\rangle \leq \|X\|\|Y\|$ by Cauchy-Schwarz. □

Finally, we will also often need the following generalization of Cauchy-Schwarz:

Proposition 10.6. *[Cauchy-Schwarz for random variables] Suppose $A \in \mathbb{R}$ and $B \in \mathbb{R}$ are arbitrary random variables. Then:*

$$\mathbb{E}[AB] \leq \sqrt{\mathbb{E}[A^2]\,\mathbb{E}[B^2]}$$

Proof. By Young inequality (Proposition 10.5), we have

$$\mathbb{E}[AB] \leq \frac{\mathbb{E}[A^2]}{2\lambda^2} + \frac{\lambda^2\,\mathbb{E}[B^2]}{2}$$

for any $\lambda > 0$. Now, set $\lambda = \frac{\mathbb{E}[A^2]}{\mathbb{E}[B^2]}$ to conclude the desired result. □

11 Finer-Grained Adaptive SGD analysis

Previously, we examined the learning rates schedule:

$$\eta_t = \frac{c}{\sqrt{\epsilon^2 + \sum_{i=1}^t \|\mathbf{g}_i\|^2}}$$

with the corresponding Algorithm 9, restated below:

Algorithm 10 Adaptive Stochastic Gradient Descent

Input: Initial Point $\mathbf{w}_1$, learning rates scaling c, small constant ϵ (e.g. 1e-4).
for $t = 1 \dots T$ **do**
  Sample $z_t \sim P_z$.
  Set $\mathbf{g}_t = \nabla \ell(\mathbf{w}_t, z_t)$.
  Set $\eta_t = \frac{c}{\sqrt{\epsilon^2 + \sum_{i=1}^t \|\mathbf{g}_i\|^2}}$
  Set $\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t \mathbf{g}_t$.
end for

And we were able to prove the following result:

Theorem 10.3. *Suppose $\mathcal{L}$ is H-smooth and $G_{\mathcal{L}}$ Lipschitz, and let $\Delta = \mathcal{L}(\mathbf{w}_t) - \mathcal{L}(\mathbf{w}_\star)$. Suppose $\mathbb{E}[\max_{t \le T} \|\mathbf{g}_t\|] \le G_\ell$ and $\mathbb{E}[\|\mathbf{g}_t - \nabla \mathcal{L}(\mathbf{w}_t)\|^2] \le \sigma^2$ for all t. Define*

$$K = \frac{\Delta}{c} + \frac{Hc \log\left(\frac{T(G_{\mathcal{L}}^2 + \sigma^2)}{\epsilon^2}\right)}{2} + \frac{G_{\mathcal{L}} G_\ell}{\epsilon} = O(\log(T))$$

Then Algorithm 9 guarantees:

$$\frac{1}{T}\,\mathbb{E}\left[\sum_{t=1}^T \|\nabla \mathcal{L}(\mathbf{w}_t)\|\right] \le \frac{K\sqrt{2} + \sqrt{K\epsilon}}{\sqrt{T}} + \frac{\sqrt{K\sigma}}{T^{1/4}}$$

However, it turns out an even more fine-grained result is true:

Theorem 11.1. *Suppose $\mathcal{L}$ is H-smooth and $G_{\mathcal{L}}$ Lipschitz, and let $\Delta = \mathcal{L}(\mathbf{w}_t) - \mathcal{L}(\mathbf{w}_\star)$. Suppose $\mathbb{E}[\max_{t \le T} \|\mathbf{g}_t\|] \le G_\ell$ and define $\sigma_t^2 = \mathbb{E}[\|\mathbf{g}_t - \nabla \mathcal{L}(\mathbf{w}_t)\|^2]$ for all t. Then*

$$K = \frac{\Delta}{c} + \frac{Hc \log\left(\frac{T(G_{\mathcal{L}}^2 + \sigma^2)}{\epsilon^2}\right)}{2} + \frac{G_{\mathcal{L}} G_\ell}{\epsilon} = O(\log(T))$$

Then Algorithm 9 guarantees:

$$\frac{1}{T}\,\mathbb{E}\left[\sum_{t=1}^T \|\nabla \mathcal{L}(\mathbf{w}_t)\|\right] \le \frac{1}{\sqrt{T}}\,\mathbb{E}\left[\sqrt{\sum_{t=1}^T \|\nabla \mathcal{L}(\mathbf{w}_t)\|^2}\right] \le \frac{K\sqrt{2} + \sqrt{K\epsilon}}{\sqrt{T}} + \frac{\sqrt{K\sqrt{\frac{1}{T}\,\mathbb{E}\left[\sum_{t=1}^T \sigma_t^2\right]}}}{T^{1/4}}$$

The proof is essentially the same - the only difference is that we have not performed the last Cauchy-Schwarz argument to bound $\frac{1}{T}\,\mathbb{E}\left[\sum_{t=1}^T \|\nabla \mathcal{L}(\mathbf{w}_t)\|\right] \le \frac{1}{\sqrt{T}}\,\mathbb{E}\left[\sqrt{\sum_{t=1}^T \|\nabla \mathcal{L}(\mathbf{w}_t)\|^2}\right]$, and also at a certain point in the proof of Theorem 10.3, we used the bound

$$\sqrt{\mathbb{E}\left[\sum_{t=1}^T \|\mathbf{g}_t - \nabla \mathcal{L}(\mathbf{w}_t)\|^2\right]} \le \sigma\sqrt{T}$$

and instead we should have used the more fine-grained bound:

$$\sqrt{\mathbb{E}\left[\sum_{t=1}^{T}\|\mathbf{g}_t - \nabla\mathcal{L}(\mathbf{w}_t)\|^2\right]} = \sqrt{\frac{1}{T}\mathbb{E}\left[\sum_{t=1}^{T}\sigma_t^2\right]}\sqrt{T}$$

The result of Theorem 11.1 is much more appealing than that of Theorem 10.3 because it is written in terms of the varying variances σ_t. How might this be helpful? Consider a situation in which $\mathbb{E}[\|\nabla\ell(\mathbf{w},z) - \nabla\mathcal{L}(\mathbf{w})\|^2] \leq D\|\nabla\mathcal{L}(\mathbf{w})\|^2$ for all $\mathbf{w}$ for some d. That is, points with small gradients also have small variance. This is not entirely unlikely: in the case that it is possible to obtain $\mathcal{L}(\mathbf{w}_\star) = 0$ and $\ell(\mathbf{w},z) \geq 0$ for all $\mathbf{w}$ and z, we would have $\mathbb{E}[\|\nabla\ell(\mathbf{w}_\star,z) - \nabla\mathcal{L}(\mathbf{w}_\star)\|^2] = 0$. If in addition we have that $\ell(\mathbf{w},z)$ is H-smooth in $\mathbf{w}$ for all z, This would also imply that $\mathbb{E}[\|\nabla\ell(\mathbf{w},z) - \nabla\mathcal{L}(\mathbf{w})\|^2] \leq H^2\|\mathbf{w}-\mathbf{w}_\star\|^2$. Thus, if it happened to be that $\|\nabla\mathcal{L}(\mathbf{w})\| \geq \|\mathbf{w}-\mathbf{w}_\star\|$ (this would be implied by strong-convexity for example), then indeed it would hold that $\mathbb{E}[\|\nabla\ell(\mathbf{w},z) - \nabla\mathcal{L}(\mathbf{w})\|^2] \leq K\|\nabla\mathcal{L}(\mathbf{w})\|^2$ for all $\mathbf{w}$ for some D.

Regardless, under this assumption, we have:

$$\begin{aligned}\frac{1}{\sqrt{T}}\mathbb{E}\left[\sqrt{\sum_{t=1}^{T}\|\nabla\mathcal{L}(\mathbf{w}_t)\|^2}\right] &\leq \frac{K\sqrt{2}+\sqrt{K\epsilon}}{\sqrt{T}} + \frac{\sqrt{K\sqrt{\frac{1}{T}\mathbb{E}\left[\sum_{t=1}^{T}\sigma_t^2\right]}}}{T^{1/4}} \\ &\leq \frac{K\sqrt{2}+\sqrt{K\epsilon}}{\sqrt{T}} + \frac{\sqrt{KD\sqrt{\frac{1}{T}\mathbb{E}\left[\sum_{t=1}^{T}\|\nabla\mathcal{L}(\mathbf{w}_t)\|^2\right]}}}{T^{1/4}}\end{aligned}$$

Now, we can do another self-bounding style argument. Let $X = \sqrt{\frac{1}{T}\mathbb{E}\left[\sum_{t=1}^{T}\|\nabla\mathcal{L}(\mathbf{w}_t)\|^2\right]}$. Then, we have:

$$X \leq \frac{K\sqrt{2}+\sqrt{K\epsilon}}{\sqrt{T}} + \frac{\sqrt{KDX}}{T^{1/4}}$$

Now, we can use the quadratic formula again to obtain:

$$\sqrt{X} \leq \frac{\sqrt{KD}/T^{1/4} + \sqrt{\frac{KD}{\sqrt{T}} + 4\frac{K\sqrt{2}+\sqrt{K\epsilon}}{\sqrt{T}}}}{2}$$

using $\sqrt{a+b} \leq \sqrt{a} + \sqrt{b}$:

$$\leq \frac{\sqrt{KD}}{T^{1/4}} + \frac{\sqrt{K\sqrt{2}+\sqrt{K\epsilon}}}{T^{1/4}}$$

Using $(a+b)^2 \leq 2a^2 + 2b^2$:

$$X \leq \frac{2KD + 2K\sqrt{2} + 2\sqrt{K\epsilon}}{\sqrt{T}}$$

using $\frac{1}{T}\mathbb{E}\left[\sum_{t=1}^{T}\|\nabla\mathcal{L}(\mathbf{w}_t)\|\right] \leq X$:

$$\frac{1}{T}\mathbb{E}\left[\sum_{t=1}^{T}\|\nabla\mathcal{L}(\mathbf{w}_t)\|\right] \leq \frac{2KD + 2K\sqrt{2} + 2\sqrt{K\epsilon}}{\sqrt{T}}$$

So that we actually converge at an $O(1/\sqrt{T})$ rate rather than an $O(1/T^{1/4})$ rate - there is a kind of "virtuous cycle" in which making progress makes the variance smaller, which allows for faster progress.

12 Tracking the variance

Let's take a look at one step of stochastic gradient descent:

$$\mathbb{E}[\mathcal{L}(\mathbf{w}_{t+1})] \leq \mathbb{E}\left[\mathcal{L}(\mathbf{w}_t) - \eta\langle \mathbf{g}_t, \nabla\mathcal{L}(\mathbf{w}_t)\rangle + \frac{H\eta^2}{2}\|\mathbf{g}_t\|^2\right]$$

Using bias-variance decomposition, and assuming η is independent of $\mathbf{g}_t$:

$$\leq \mathbb{E}\left[\mathcal{L}(\mathbf{w}_t) - \left(\eta - \frac{H\eta^2}{2}\right)\|\nabla\mathcal{L}(\mathbf{w}_t)\|^2 + \frac{H\eta^2}{2}\|\mathbf{g}_t - \nabla\mathcal{L}(\mathbf{w}_t)\|^2\right]$$

assuming $\eta \leq \frac{1}{H}$, and writing $\sigma_t^2 = \mathbb{E}[\|\mathbf{g}_t - \nabla\mathcal{L}(\mathbf{w}_t)\|^2]$:

$$\leq \mathbb{E}\left[\mathcal{L}(\mathbf{w}_t) - \frac{\eta}{2}\|\nabla\mathcal{L}(\mathbf{w}_t)\|^2 + \frac{H\eta^2\sigma_t^2}{2}\right]$$

Now, from this we should expect that the right learning rate *at this iterate* is something like $\eta = \min\left(\frac{1}{H}, \frac{\|\nabla\mathcal{L}(\mathbf{w}_t)\|^2}{2H\sigma_t^2}\right)$ (to see this, differentiate the above with respect. to η). After a little case-work, this yields:

$$\mathbb{E}[\mathcal{L}(\mathbf{w}_{t+1})] \leq \mathbb{E}\left[\mathcal{L}(\mathbf{w}_t) - \min\left(\frac{\|\nabla\mathcal{L}(\mathbf{w}_t)\|^2}{4H}, \frac{\|\nabla\mathcal{L}(\mathbf{w}_t)\|^4}{8H\sigma_t^2}\right)\right]$$

Following now the usual train logic, we end up with:

$$\mathbb{E}\left[\frac{1}{T}\sum_{t=1}^{T}\min\left(\frac{\|\nabla\mathcal{L}(\mathbf{w}_t)\|^2}{4H}, \frac{\|\nabla\mathcal{L}(\mathbf{w}_t)\|^4}{8H\sigma_t^2}\right)\right] \leq \frac{\Delta}{T}$$

Therefore, if $\hat{\mathbf{w}}$ is a randomly selected iterate, we have

$$\mathbb{E}\left[\min\left(\frac{\|\nabla\mathcal{L}(\hat{\mathbf{w}})\|^2}{4H}, \frac{\|\nabla\mathcal{L}(\hat{\mathbf{w}})\|^4}{8H\,\mathbb{E}_z[\|\nabla\ell(\hat{\mathbf{w}}, z) - \nabla\mathcal{L}(\hat{\mathbf{w}})\|^2}\right)\right] \leq \frac{\Delta}{T}$$

This suggests that the point $\hat{\mathbf{w}}$ has $\|\nabla\mathcal{L}(\hat{\mathbf{w}})\| = O(1/T^{1/4})$, but also that the proportionality constant is now related to the variance *at the same point* $\hat{\mathbf{w}}$ rather than a sum of variance terms over all iterates. This might enable us to obtain similar "virtuous cycles" as found in the previous section with much weaker conditions on the losses.

The key problem here of course is that we do not actually know the "instantaneous" variance σ_t, nor the true gradient norm $\|\nabla\mathcal{L}(\mathbf{w}_t)\|$. The latter sounds very difficult to obtain, but what if we were able to estimate the former? Could we still obtain an improved bound?

Let's consider learning rates:

$$\eta_t = \min\left(\frac{1}{H}, \frac{\sqrt{\Delta}}{\sigma_t\sqrt{TH}}\right)$$

Then, following the same logic as usual, we obtain:

$$\mathbb{E}\left[\mathcal{L}(\mathbf{w}_{t+1})\right] \leq \mathbb{E}\left[\mathcal{L}(\mathbf{w}_t) - \frac{\eta_t}{2}\|\nabla\mathcal{L}(\mathbf{w}_t)\|^2 + \frac{\Delta}{2T}\right]$$

$$\mathbb{E}\left[\sum_{t=1}^{T}\eta_t\|\nabla\mathcal{L}(\mathbf{w}_t)\|^2\right] \leq 3\Delta$$

From which a randomly selected $\hat{\mathbf{w}}$ obtains:

$$\mathbb{E}\left[\min\left(\frac{1}{H}, \frac{\sqrt{\Delta}}{\sqrt{\mathbb{E}_z[\|\nabla\ell(\hat{\mathbf{w}}, z) - \nabla\mathcal{L}(\hat{\mathbf{w}})\|^2]TH}}\right)\|\nabla\mathcal{L}(\hat{\mathbf{w}})\|^2\right] \leq \frac{3\Delta}{T}$$

Thus, for large T we should expect

$$\mathbb{E}\left[\frac{\|\nabla \mathcal{L}(\hat{\mathbf{w}})\|^2}{\mathbb{E}_z[\|\nabla \ell(\hat{\mathbf{w}}, z) - \nabla \mathcal{L}(\hat{\mathbf{w}})\|^2}\right] \leq O\left(\frac{1}{\sqrt{T}}\right)$$

This is a similar result, but we still have an issue: how did we know the instantaneous variance σ_t? A standard heuristic used in practice that might achieve this goal is the *exponentially weighted moving average*. In this scheme, we choose some parameter $\beta \in (0, 1)$ and then set

$$\eta_t = \frac{c}{\sqrt{\epsilon^2 + \sum_{i=1}^t \beta^{t-1} \|\mathbf{g}_i\|^2}}$$

This can be computed efficiently, as demonstrated in the following pseudocode:

Algorithm 10 Adaptive Stochastic Gradient Descent with EMA

Input: Initial Point $\mathbf{w}_1$, learning rates scaling c, small constant ϵ (e.g. 1e-4), averaging factor β:
Set $A_0 = 0$.
for $t = 1 \dots T$ **do**
  Sample $z_t \sim P_z$.
  Set $\mathbf{g}_t = \nabla \ell(\mathbf{w}_t, z_t)$.
  Set $A_t = \beta A_{t-1} + \|\mathbf{g}_t\|^2$.
  Set $\eta_t = \frac{c}{\sqrt{\epsilon^2 + A_t}}$
  Set $\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t \mathbf{g}_t$.
end for

Intuitively, this setting of η_t is allowed to "forget" about previous gradients, so that if σ_t happens to be large in the first iterates, this will not have as big an effect on later iterates.

Unfortunately, the algorithm as described here does not actually converge. The problem is that the learning rates η_t may increase quite rapidly during the training. Instead, as suggested by Reddi, Kale, and Kumar 2018, we could try the following:

Algorithm 11 Corrected Adaptive Stochastic Gradient Descent with EMA

Input: Initial Point $\mathbf{w}_1$, learning rates scaling c, small constant ϵ (e.g. 1e-4), averaging factor β:
Set $A_0 = 0$.
Set $B_0 = \epsilon^2$.
for $t = 1 \dots T$ **do**
  Sample $z_t \sim P_z$.
  Set $\mathbf{g}_t = \nabla \ell(\mathbf{w}_t, z_t)$.
  Set $A_t = \beta A_{t-1} + \|\mathbf{g}_t\|^2$.
  Set $B_t = \max(B_{t-1}, tA_t)$
  Set $\eta_t = \frac{c}{\sqrt{B_t}}$
  Set $\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t \mathbf{g}_t$.
end for

This algorithm has the following convergence guarantee:

Theorem 12.1. *Suppose $\mathcal{L}$ is H-smooth and $G_{\mathcal{L}}$ Lipschitz, and let $\Delta = \mathcal{L}(\mathbf{w}_t) - \mathcal{L}(\mathbf{w}_\star)$. Suppose $\mathbb{E}[\max_{t \leq T} \|\mathbf{g}_t\|] \leq G_\ell$ for all t. Then Algorithm 11 guarantees:*

$$\mathbb{E}[\mathcal{L}(\mathbf{w}_{T+1})] \leq \mathbb{E}\left[\mathcal{L}(\mathbf{w}_1) - \sum_{t=1}^T \eta_{t-1} \|\nabla \mathcal{L}(\mathbf{w}_t)\|^2\right] + \frac{H}{2}(1 + \log(T)) + \eta_0 G_{\mathcal{L}} G_\ell$$

Moreover, it is also guaranteed that:

$$\mathbb{E}\left[\sum_{t=1}^T \|\nabla \mathcal{L}(\mathbf{w}_t)\|^2\right] \leq \left(\Delta + \frac{H}{2}(1 + \log(T)) + \eta_0 G_{\mathcal{L}} G_\ell\right) \frac{\epsilon + G\sqrt{T}}{\sqrt{1 - \beta}}$$

Note that this result actually doesn't imply any obvious gain over the previous results for adaptive gradient descent. While intuitively it seems reasonable that the algorithm will perform better (and in practice it frequently does), the analysis is sufficiently complicated that there is no solid result characterizing when Algorithm 11 is actually better than 9.

Proof. Recall from Theorem 1 of the previous notes 6 (which applies since η_t is now guaranteed to be decreasing), that we can write:

$$\mathbb{E}[\mathcal{L}(\mathbf{w}_{T+1})] \leq \mathbb{E}\left[\mathcal{L}(\mathbf{w}_1) - \sum_{t=1}^{T} \eta_{t-1} \|\nabla \mathcal{L}(\mathbf{w}_t)\|^2 + \frac{H}{2} \sum_{t=1}^{T} \eta_t^2 \|\mathbf{g}_t\|^2\right] + \eta_0 G_{\mathcal{L}} G_\ell$$

Now, to bound $\sum_{t=1}^{T} \eta_t^2 \|\mathbf{g}_t\|^2$, we have:

$$\begin{aligned}
\sum_{t=1}^{T} \eta_t^2 \|\mathbf{g}_t\|^2 &\leq \sum_{t=1}^{T} \frac{\|\mathbf{g}_t\|^2}{t A_t} \\
&\leq \frac{\|\mathbf{g}_t\|^2}{t \|\mathbf{g}_t\|^2} \\
&\leq \frac{1}{t}
\end{aligned}$$

Therefore

$$\begin{aligned}
\mathbb{E}[\mathcal{L}(\mathbf{w}_{T+1})] &\leq \mathbb{E}\left[\mathcal{L}(\mathbf{w}_1) - \sum_{t=1}^{T} \eta_{t-1} \|\nabla \mathcal{L}(\mathbf{w}_t)\|^2\right] + \frac{H}{2}(1 + \log(T)) + \eta_0 G_{\mathcal{L}} G_\ell \\
\mathbb{E}\left[\sum_{t=1}^{T} \eta_{t-1} \|\nabla \mathcal{L}(\mathbf{w}_t)\|^2\right] &\leq \Delta + \frac{H}{2}(1 + \log(T)) + \eta_0 G_{\mathcal{L}} G_\ell \\
\mathbb{E}\left[(\min_t \eta_t) \sum_{t=1}^{T} \|\nabla \mathcal{L}(\mathbf{w}_t)\|^2\right] &\leq \Delta + \frac{H}{2}(1 + \log(T)) + \eta_0 G_{\mathcal{L}} G_\ell
\end{aligned}$$

Now, observe that

$$A_t \leq G^2 \sum_{i=0}^{t-1} \beta^i \leq \frac{G^2}{1-\beta}$$

Therefore, $\min_t \eta_t \geq \frac{1}{\epsilon + \sqrt{T \max A_t}} \geq \frac{\sqrt{1-\beta}}{\epsilon + G\sqrt{T}}$. This implies

$$\mathbb{E}\left[\sum_{t=1}^{T} \|\nabla \mathcal{L}(\mathbf{w}_t)\|^2\right] \leq \left(\Delta + \frac{H}{2}(1 + \log(T)) + \eta_0 G_{\mathcal{L}} G_\ell\right) \frac{\epsilon + G\sqrt{T}}{\sqrt{1-\beta}}$$

□

13 Smooth Convex Losses and Acceleration

Let's consider again the case of convex losses. Previously, we studied losses $\mathcal{L}$ that are convex and Lipschitz. Now, we will consider $\mathcal{L}$ that are convex, Lipschitz, and *smooth*. We'll see that significantly better results are sometimes possible in this setting. The key idea is smoothness provides an *upper bound* on the function value via our standard smoothness lemma:

$$\mathcal{L}(\mathbf{w}+\delta) \leq \mathcal{L}(\mathbf{w}) + \langle \nabla \mathcal{L}(\mathbf{w}), \delta \rangle + \frac{H}{2}\|\delta\|^2$$

On the other hand, convexity provides an *lower bound* on the function value:

$$\mathcal{L}(\mathbf{w}+\delta) \geq \mathcal{L}(\mathbf{w}) + \langle \nabla \mathcal{L}(\mathbf{w}), \delta \rangle$$

By carefully combining these operations, we can provide improved convergence rates on convex losses. Let's start by revisiting our analysis of (non-stochastic) gradient descent for convex losses. Suppose that we start at the origin, so that $\mathbf{w}_1 = 0$. Then, with a constant learning rate η so that $\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \nabla \mathcal{L}(\mathbf{w}_t)$, we previously proved:

Theorem 2.4. *Suppose $\mathcal{L}$ is G-Lipschitz and convex. Suppose $\|\mathbf{w}_\star - \mathbf{w}_1\| \leq D$. Set $\eta = \frac{D}{G\sqrt{T}}$. Then*

$$\sum_{t=1}^{T} \mathcal{L}(\mathbf{w}_t) - \mathcal{L}(\mathbf{w}_\star) \leq GD\sqrt{T}$$

In particular, if $\hat{\mathbf{w}}$ is selected uniformly at random from $\mathbf{w}_1, \dots, \mathbf{w}_T$,

$$\mathbb{E}[\mathcal{L}(\hat{\mathbf{w}}) - \mathcal{L}(\mathbf{w}_t)] \leq \frac{GD}{\sqrt{T}}$$

In fact, the proof of this theorem involved a more general inequality, which we generalized even further on the homework. Specifically, we have:

Theorem 13.1. *Suppose $\mathcal{L}$ is* any *differentiable function. Then gradient descent with learning rate η guarantees:*

$$\sum_{t=1}^{T} \langle \nabla \mathcal{L}(\mathbf{w}_t), \mathbf{w}_t - \mathbf{w}_\star \rangle \leq \frac{\|\mathbf{w}_\star - \mathbf{w}_1\|^2}{2\eta} + \frac{\eta}{2}\sum_{t=1}^{T} \|\nabla \mathcal{L}(\mathbf{w}_t)\|^2$$

If $\mathcal{L}$ is also convex:

$$\sum_{t=1}^{T} \mathcal{L}(\mathbf{w}_t) - \mathcal{L}(\mathbf{w}_\star) \leq \frac{\|\mathbf{w}_\star - \mathbf{w}_1\|^2}{2\eta} + \frac{\eta}{2}\sum_{t=1}^{T} \|\nabla \mathcal{L}(\mathbf{w}_t)\|^2$$

Note that the first part of this theorem does not actually rely on convexity!

Proof. Let's again bound the change in $\|\mathbf{w}_t - \mathbf{w}_\star\|^2$:

$$\begin{aligned}\|\mathbf{w}_{t+1} - \mathbf{w}_\star\|^2 &= \|\mathbf{w}_t - \eta \nabla \mathcal{L}(\mathbf{w}_t) - \mathbf{w}_\star\|^2 \\ &= \|\mathbf{w}_t - \mathbf{w}_\star\|^2 - 2\eta \langle \nabla \mathcal{L}(\mathbf{w}_t), \mathbf{w}_t - \mathbf{w}_\star \rangle + \eta^2 \|\nabla \mathcal{L}(\mathbf{w}_t)\|^2\end{aligned}$$

rearranging:

$$\langle \nabla \mathcal{L}(\mathbf{w}_t), \mathbf{w}_t - \mathbf{w}_\star \rangle \leq \frac{\|\mathbf{w}_t - \mathbf{w}_\star\|^2}{2\eta} - \frac{\|\mathbf{w}_{t+1} - \mathbf{w}_\star\|^2}{2\eta} + \frac{\eta}{2}\|\nabla \mathcal{L}(\mathbf{w}_t)\|^2$$

sum over t, and telescope the RHS:

$$\begin{aligned}\sum_{t=1}^{T} \langle \nabla \mathcal{L}(\mathbf{w}_t), \mathbf{w}_t - \mathbf{w}_\star \rangle &\leq \frac{\|\mathbf{w}_1 - \mathbf{w}_\star\|^2}{2\eta} - \frac{\|\mathbf{w}_{T+1} - \mathbf{w}_\star\|^2}{2\eta} + \frac{\eta}{2}\sum_{t=1}^{T} \|\nabla \mathcal{L}(\mathbf{w}_t)\|^2 \\ &\leq \frac{\|\mathbf{w}_\star - \mathbf{w}_1\|^2}{2\eta} + \frac{\eta}{2}\sum_{t=1}^{T} \|\nabla \mathcal{L}(\mathbf{w}_t)\|^2\end{aligned}$$

And now the final result follows by convexity, since $\mathcal{L}(\mathbf{w}_t) - \mathcal{L}(\mathbf{w}_\star) \leq \langle \nabla \mathcal{L}(\mathbf{w}_t), \mathbf{w}_t - \mathbf{w}_\star \rangle$ □

Now, previously, we assumed that $\mathcal{L}$ was G-Lipschitz, and used this to bound the $\|\nabla\mathcal{L}(\mathbf{w}_t)\|^2$ terms in the RHS of Theorem 13.1. This time, we'll use smoothness to make a more refined statement. Smoothness implies:

Lemma 13.2. *Suppose $\mathcal{L} : \mathbb{R}^d \to \mathbb{R}$ is an H-smooth function. Then for any $\mathbf{w}$:*

$$\frac{\|\nabla\mathcal{L}(\mathbf{w})\|^2}{2H} \leq \mathcal{L}(\mathbf{w}) - \mathcal{L}(\mathbf{w}_\star)$$

Warning: *the proof of this theorem may not work if $\mathcal{L}$ is only defined on a bounded subset of $\mathbb{R}^d$ - see if you can spot what would go wrong.*

Proof. From smoothness, we have for any δ:

$$\mathcal{L}(\mathbf{w}+\delta) \leq \mathcal{L}(\mathbf{w}) + \langle\nabla\mathcal{L}(\mathbf{w}), \delta\rangle + \frac{H}{2}\|\delta\|^2$$

Set $\delta = -\frac{\nabla\mathcal{L}(\mathbf{w})}{H}$ to obtain:

$$\mathcal{L}(\mathbf{w} - \nabla\mathcal{L}(\mathbf{w})/H) \leq \mathcal{L}(\mathbf{w}) - \frac{\|\nabla\mathcal{L}(\mathbf{w})\|^2}{2H}$$
$$\mathcal{L}(\mathbf{w}_\star) \leq \mathcal{L}(\mathbf{w}) - \frac{\|\nabla\mathcal{L}(\mathbf{w})\|^2}{2H}$$

□

In words, this tells us that when $\mathcal{L}$ is smooth, then small function values imply small gradient values. This will help us prove faster convergence rates for convex functions because in Theorem 13.1, we notice that the error becomes smaller when $\nabla\mathcal{L}(\mathbf{w}_t)$ gets smaller. This will set up a nice feedback cycle: as GD converges, $\nabla\mathcal{L}(\mathbf{w}_t)$ will get smaller, which will cause GD to converge even faster, which will let $\nabla\mathcal{L}(\mathbf{w}_t)$ get even smaller, which again causes faster convergence and so on.

Let's see how we can use this Lemma 13.2 in concert with Theorem 13.1:

Theorem 13.3. *Suppose $\mathcal{L}$ is an H-smooth convex function. Set $\eta = \frac{1}{2H}$. Then gradient descent guarantees:*

$$\sum_{t=1}^{T} \mathcal{L}(\mathbf{w}_t) - \mathcal{L}(\mathbf{w}_\star) \leq 2H\|\mathbf{w}_\star - \mathbf{w}_1\|^2$$

In particular, if $\hat{\mathbf{w}} = \frac{1}{T}\sum_{t=1}^{T}\mathbf{w}_t$, then

$$\mathcal{L}(\hat{\mathbf{w}}) - \mathcal{L}(\mathbf{w}_\star) \leq \frac{2H\|\mathbf{w}_\star - \mathbf{w}_1\|^2}{T}$$

Proof. From Theorem 13.1, we have

$$\sum_{t=1}^{T} \mathcal{L}(\mathbf{w}_t) - \mathcal{L}(\mathbf{w}_\star) \leq \frac{\|\mathbf{w}_\star - \mathbf{w}_1\|^2}{2\eta} + \frac{\eta}{2}\sum_{t=1}^{T}\|\nabla\mathcal{L}(\mathbf{w}_t)\|^2$$

now, from Lemma 13.2:

$$\leq \frac{\|\mathbf{w}_\star - \mathbf{w}_1\|^2}{2\eta} + \frac{\eta}{2}\sum_{t=1}^{T} 2H(\mathcal{L}(\mathbf{w}_t) - \mathcal{L}(\mathbf{w}_\star))$$

Using our definition of η:

$$= H\|\mathbf{w}_\star - \mathbf{w}_1\|^2 + \frac{1}{2}\sum_{t=1}^{T}\mathcal{L}(\mathbf{w}_t) - \mathcal{L}(\mathbf{w}_\star)$$

rearranging:

$$\sum_{t=1}^{T} \mathcal{L}(\mathbf{w}_t) - \mathcal{L}(\mathbf{w}_\star) \leq 2H\|\mathbf{w}_\star - \mathbf{w}_1\|^2$$

Now, we need to show the last part of the Theorem. For this, note that if X is a random variable that takes on each values $\mathbf{w}_t$ with probability $1/T$, then $\mathbb{E}[X] = \hat{\mathbf{w}}$. Therefore, by Jensen:

$$\mathcal{L}(\hat{\mathbf{w}}) = \mathcal{L}(\mathbb{E}[X]) \leq \mathbb{E}[\mathcal{L}(X)] = \frac{1}{T}\sum_{t=1}^{T} \mathcal{L}(\mathbf{w}_t)$$

Thus,

$$\mathcal{L}(\hat{\mathbf{w}}) - \mathcal{L}(\mathbf{w}_\star) \leq \frac{1}{T}\sum_{t=1}^{T} \mathcal{L}(\mathbf{w}_t) - \mathcal{L}(\mathbf{w}_\star)$$

and so comparing the RHS to the bound we have already obtained, the result follows. □

This result is *significantly* better than the previous results we had for convex optimization: here we obtained $O(1/T)$ suboptimality, while previously it was only $O(1/\sqrt{T})$. The difference in assumptions is that this time we assumed H-smoothness, while last time we assumed G-Lipschitzness. It turns out that the $O(1/\sqrt{T})$ rate is *tight* for G-Lipschitz losses. That is, given any algorithm, there exists a G-Lipschitz convex function such that after T iterations, the algorithm cannot do better than $O(1/\sqrt{T})$ (see Bubeck et al. 2015).

However, in the smooth case we can actually do even better than $O(1/T)$, it is possible to obtain $O(1/T^2)$! Algorithms that achieve this rate (which is optimal) are called *accelerated* optimization algorithms. This was famously first achieved in 1983 by Yuri Nesterov in a much-studied algorithm. The original paper is in Russian, but there are texts that cover it in various levels of detail Bubeck et al. 2015; Nesterov n.d.

14 Momentum

Before getting into the technical details of acceleration, let's discuss a possibly more intuitive algorithm. This is the idea of *momentum*. Momentum is probably the single most ubiquitous modification added to SGD/GD in machine learning. The basic idea can be viewed via a kind of "physical intuition". If we consider the objective function $\mathcal{L}(\mathbf{w})$ as a kind of "contour map" in which $\mathcal{L}(\mathbf{w})$ indicates the height of the landscape at some "d-dimensional" GPS coordinates $\mathbf{w}$, then we might view gradient descent as an algorithm that finds a local minimum of $\mathcal{L}$ by repeatedly taking a step in the direction of steepest incline of the landscape.

However, if you were to drop a spherical bowling ball at some point on the landscape, it actually would not always take the direction of steepest incline: as it rolls it would pick up some *momentum* that would keep it going uphill occasionally. Although going uphill seems bad, overall we might feel this is a good thing because the momentum will allow the ball to speed up while rolling down a hill. An important point here is the idea of *friction* - without some friction to decrease the kinetic energy of the bowling ball, it will happily roll uphill exactly as far as it has rolled downhill and make zero progress. By adding an appropriate amount of friction, we could hope to limit the uphill motion while still getting the gains for the downhill motion. Intuitively, the regular gradient descent is like a bowling ball that has a near-infinite amount of friction associated with its motion so that it cannot pick up any momentum.

This idea is captured by defining an auxiliary variable $\mathbf{v}$, which represents the "velocity" of the parameter $\mathbf{w}$. $\mathbf{v}$ is updated using the gradient of $\mathcal{L}$, so that $\mathcal{L}$ is playing the role of a "potential energy":

$$\mathbf{v}_t = \beta\mathbf{v}_{t-1} - \eta\nabla\mathcal{L}(\mathbf{w}_t) \tag{5}$$

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \mathbf{v}_t \tag{6}$$

Here the coefficient $\beta \in [0, 1]$ plays the role of "friction". $\beta = 0$ corresponds to "infinite friction" - all of the kinetic energy is immediately removed and so no velocity gets to carry over to the next iterate. Inspecting the equations shows that with $\beta = 0$, the update is identical to regular gradient descent. $\beta = 1$ corresponds to zero friction. Typically β is set to some large value like 0.9 or 0.99.

14.1 Better integrators

Algorithm 12 Gradient Descent with Momentum

Input: Initial Point $\mathbf{w}_1$, learning rates $\eta_1, \dots, \eta_T$, momentum parameters $\beta_1, \dots, \beta_T$:
Set $\mathbf{v}_0 = 0$.
for $t = 1 \dots T$ **do**
  if Using two-step integrator **then**
    Set $\mathbf{v}_{t-\frac{1}{2}} = \beta_t \mathbf{v}_{t-1}$.
    Set $\mathbf{w}_{t+\frac{1}{2}} = \mathbf{w}_t - \mathbf{v}_{t-\frac{1}{2}}$
    Set $\mathbf{g}_t = \nabla \mathcal{L}(\mathbf{w}_{t+\frac{1}{2}})$.
  else
    Set $\mathbf{g}_t = \nabla \mathcal{L}(\mathbf{w}_t)$
  end if
  Set $\mathbf{v}_t = \beta_t \mathbf{v}_{t-1} - \eta_t \mathbf{g}_t$.
  Set $\mathbf{w}_{t+1} = \mathbf{w}_t - \mathbf{v}_t$.
end for

We can view the momentum update as numerically integrating the following physical "equations of motion":

$$\mathbf{v} = \frac{d\mathbf{w}}{dt}$$
$$\frac{d\mathbf{v}}{dt} = -(1-\beta)\mathbf{v} + \eta \nabla \mathcal{L}(\mathbf{w})$$

The momentum update described in equations (5) and (6) correspond to the most basic Euler integration step. However, there are more accurate ways to simulate this differential equation. Specifically, consider the following "two-step" integrator:

$$\mathbf{v}_{t-\frac{1}{2}} = \beta \mathbf{v}_{t-1}$$
$$\mathbf{w}_{t+\frac{1}{2}} = \mathbf{w}_t + \mathbf{v}_{t-\frac{1}{2}}$$
$$\mathbf{v}_t = \beta \mathbf{v}_{t-1} - \eta \nabla \mathcal{L}(\mathbf{w}_{t+\frac{1}{2}})$$
$$\mathbf{w}_{t+1} = \mathbf{w}_t + \mathbf{v}_t$$

The intuition here is that the equation $\frac{d\mathbf{v}}{dt} = -(1-\beta)\mathbf{v} + \eta \nabla \mathcal{L}(\mathbf{w})$ containts two terms, a $-(1-\beta)\mathbf{v}$ and a $\eta \nabla \mathcal{L}(\mathbf{w})$. If we naively evaluate $\nabla \mathcal{L}(\mathbf{w})$ at $\mathbf{w}_t$, we might miss some behavior in which the ball starts rolling up the hill during this step. By instead evaluating at the "lookahead" point $\mathbf{w}_t + \beta \mathbf{v}_t$, we can help to counteract this by detecting if just the momentum on its own would cause the ball to start rolling uphill.

This leads to Algorithm 12 in which we generalize the setting to allow for time-varying β and η.

Unfortunately, in order to rigorously analyze this algorithm in the convex setting and show acceleration, we will completely abandon this intuition and make an argument that follows a mess of algebra. Further, we will need to set a very delicate schedule for β_t, and re-write the algorithm into very different looking (but totally equivalent) form. This is a big issue with accelerated algorithms: although there are reasonable intuitive explanations in terms of physics like the above, the analyses are typically very strange-looking.

15 Acceleration

Now, let us forget for a momentum about momentum and just think about some equations we've seen so far at a very high level and how we might be able to use them to get a faster convergence. Our approa ch is a streamlined presentation of *linear coupling* Allen-Zhu and Orecchia 2017, which is a more recent technique for designing accelerated algorithms.

To get some intuition behind linear coupling, observe that our ordinary analysis of gradient descent without using convexity shows that as long as $\eta \leq \frac{1}{H}$:

$$\mathcal{L}(\mathbf{w}_{t+1}) \leq \mathcal{L}(\mathbf{w}_t) - \frac{\eta}{2}\|\nabla \mathcal{L}(\mathbf{w}_t)\|^2$$

That is, *we make faster progress when* $\|\nabla \mathcal{L}(\mathbf{w}_t)\|$ *is large*. In contrast, Theorem 13.1 showed:

$$\sum_{t=1}^{T} \mathcal{L}(\mathbf{w}_t) - \mathcal{L}(\mathbf{w}_\star) \leq \frac{\|\mathbf{w}_\star - \mathbf{w}_1\|^2}{2\eta} + \frac{\eta}{2}\sum_{t=1}^{T} \|\nabla \mathcal{L}(\mathbf{w}_t)\|^2$$

so that if $\hat{\mathbf{w}} = \frac{1}{T}\sum_{t=1}^{T} \mathbf{w}_t$,

$$\mathcal{L}(\hat{\mathbf{w}}) - \mathcal{L}(\mathbf{w}_\star) \leq \frac{\|\mathbf{w}_\star - \mathbf{w}_1\|^2}{2\eta t} + \frac{\eta}{2T}\sum_{t=1}^{T} \|\nabla \mathcal{L}(\mathbf{w}_t)\|^2$$

That is, *we make faster progress when* $\|\nabla \mathcal{L}(\mathbf{w}_t)\|$ *is small*. Further, the analysis of gradient descent relating $\mathcal{L}(\mathbf{w}_{t+1})$ directly to $\mathcal{L}(\mathbf{w}_t)$ does not involve any averaging, but the analysis using convexity in Theorem 13.1 does involve averaging.

Intuitively, we might then expect to obtain a kind of "best of both worlds" result by combining these analysis styles: if the gradients are big, then the first style of analysis will give us some fast progress, but if the gradients are small, then the second style might be more helpful. In particular, we'll be able to show that the "function progress" proportional to $\|\nabla \mathcal{L}(\mathbf{w}_t)\|^2$ achieved by the first style of analysis can be used with clever algebra to "cancel out" the growing sum proportional to $\|\nabla \mathcal{L}(\mathbf{w}_t)\|^2$ in the second analysis.

In order to put this together properly, it will be helpful to note an even more general version of Theorem 13.1. This next result is the starting point of the field of *online learning*, although we will not need to go any further in this direction here.

Theorem 15.1. *Suppose* $\mathbf{g}_1, \dots, \mathbf{g}_T$ *is a* completely arbitrary *sequence of vectors. Define* $\mathbf{w}_t$ *by* $\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \mathbf{g}_t$. *Then for any* $\mathbf{w}_\star$*:*

$$\sum_{t=1}^{T} \langle \mathbf{g}_t, \mathbf{w}_t - \mathbf{w}_\star \rangle \leq \frac{\|\mathbf{w}_\star - \mathbf{w}_1\|^2}{2\eta} + \frac{\eta}{2}\sum_{t=1}^{T} \|\mathbf{g}_t\|^2$$

This result is remarkable general: it allows $\mathbf{g}_t$ to really be anything at all (even something generated by some evil adversary intent on messing you up in some way), and also allows $\mathbf{w}_\star$ to be anything at all. Note that while we are kind of doing gradient descent here, it's not really clear that the $\mathbf{g}_t$s actually represent gradients of anything. The proof of this result is actually identical to the proof of Theorem 13.1. Let's just spell it out below to make sure:

Proof. Again, we consider $\|\mathbf{w}_{t+1} - \mathbf{w}_\star\|^2$. Note that this is only done "in analysis", at no point does the algorithm actually compute this quantity. That's a good thing, because $\mathbf{w}_\star$ could be anything!

$$\begin{aligned}\|\mathbf{w}_{t+1} - \mathbf{w}_\star\|^2 &= \|\mathbf{w}_t - \eta \mathbf{g}_t - \mathbf{w}_\star\|^2 \\ &= \|\mathbf{w}_t - \mathbf{w}_\star\|^2 - 2\eta \langle \mathbf{g}_t, \mathbf{w}_t - \mathbf{w}_\star \rangle + \eta^2 \|\mathbf{g}_t\|^2\end{aligned}$$

rearrange:

$$\langle \mathbf{g}_t, \mathbf{w}_t - \mathbf{w}_\star \rangle \leq \frac{\|\mathbf{w}_t - \mathbf{w}_\star\|^2 - \|\mathbf{w}_{t+1} - \mathbf{w}_\star\|^2}{2\eta} + \frac{\eta}{2}\|\mathbf{g}_t\|^2$$

sum over t, and telescope:

$$\sum_{t=1}^{T} \langle \mathbf{g}_t, \mathbf{w}_t - \mathbf{w}_\star \rangle \leq \frac{\|\mathbf{w}_1 - \mathbf{w}_\star\|^2 - \|\mathbf{w}_{T+1} - \mathbf{w}_\star\|^2}{2\eta} + \frac{\eta}{2}\sum_{t=1}^{T} \|\mathbf{g}_t\|^2$$

and finally drop the negative term to conclude the desired result. □

This theorem will in some sense "free" us to do a lot of useful algebraic manipulations without worrying about the relationship between various vectors we produce and actual gradients of $\nabla\mathcal{L}(\mathbf{w}_t)$. Without further ado, let us describe our accelerated gradient descent algorithm:

Algorithm 13 Accelerated Gradient Descent via Momentum

Input: Initial Point $\mathbf{w}_1$, smoothness constant H, time horizon T, learning rate η
Set $\mathbf{y}_1 = \mathbf{w}_1$
Set $\alpha_0 = 0$, $\alpha_1 = 1$.
for $t = 1 \dots T$ **do**
  Set $\tau_t = \frac{\alpha_t}{\sum_{i=1}^t \alpha_t}$
  Set $\mathbf{x}_t = (1-\tau_t)\mathbf{w}_t + \tau_t \mathbf{y}_t$
  Set $\mathbf{g}_t = \alpha_t \nabla\mathcal{L}(\mathbf{x}_t)$.
  Set $\mathbf{y}_{t+1} = \mathbf{y}_t - \eta \mathbf{g}_t$.
  Set $\mathbf{w}_{t+1} = \mathbf{x}_t - \eta\nabla\mathcal{L}(\mathbf{x}_t)$
  Set α_{t+1} to satisfy $\alpha_{t+1}^2 - \alpha_{t+1} = \sum_{i=1}^t \alpha_i$ (use quadratic formula).
end for

Now, this algorithm looks extremely weird, but with a little work we can massage it to look like the momentum algorithm. Specifically, set

$$\mathbf{v}_t = \frac{1}{\alpha_t}(\mathbf{y}_t - \mathbf{w}_t) - \eta\nabla\mathcal{L}(\mathbf{x}_t)$$

Observe that we have:

$$\begin{aligned}\mathbf{w}_{t+1} &= \mathbf{x}_t - \eta\nabla\mathcal{L}(\mathbf{x}_t) \\ &= (1-\tau_t)\mathbf{w}_t + \tau_t\mathbf{y}_t - \eta\nabla\mathcal{L}(\mathbf{x}_t) \\ &= \mathbf{w}_t + \tau_t(\mathbf{y}_t - \mathbf{w}_t) - \eta\nabla\mathcal{L}(\mathbf{x}_t)\end{aligned}$$

using the fact that $\alpha_t = \sum_{i=1}^t \alpha_i$, so that $\tau_t = \frac{1}{\alpha_t}$:

$$\begin{aligned}&= \mathbf{w}_t + \frac{1}{\alpha_t}(\mathbf{y}_t - \mathbf{w}_t) - \eta\nabla\mathcal{L}(\mathbf{x}_t) \\ &= \mathbf{w}_t + \mathbf{v}_t\end{aligned}$$

Further, we have the important relationship:

$$\begin{aligned}\mathbf{y}_{t+1} - \mathbf{w}_{t+1} &= \mathbf{y}_t - \eta\alpha_t\nabla\mathcal{L}(\mathbf{x}_t) - \mathbf{w}_{t+1} \\ &= \mathbf{y}_t - \eta\alpha_t\nabla\mathcal{L}(\mathbf{x}_t) - \mathbf{x}_t + \eta\nabla\mathcal{L}(\mathbf{x}_t) \\ &= \mathbf{y}_t - (1-\tau_t)\mathbf{w}_t + \tau_t\mathbf{y}_t - \eta(\alpha_t - 1)\nabla\mathcal{L}(\mathbf{x}_t) \\ &= (1-\tau_t)(\mathbf{y}_t - \mathbf{w}_t) - \eta(\alpha_t - 1)\nabla\mathcal{L}(\mathbf{x}_t)\end{aligned}$$

Now, observe two more magical properties of our definition for α_t:

$$\begin{aligned}\alpha_t - 1 &= \frac{\alpha_t^2 - \alpha_t}{\alpha_t} = \frac{\sum_{i=1}^{t-1}\alpha_i}{\alpha_t} = \frac{\alpha_{t-1}^2}{\alpha_t} \\ 1 - \tau_t &= \frac{\sum_{i=1}^{t-1}\alpha_i}{\sum_{i=1}^t \alpha_i} = \frac{\alpha_{t-1}^2}{\alpha_t^2}\end{aligned}$$

Combining this with the above implies:

$$\mathbf{y}_{t+1} - \mathbf{w}_{t+1} = \frac{\alpha_{t-1}^2}{\alpha_t}\mathbf{v}_t$$

or, with t instead of $t+1$ for all $t \geq 2$:

$$\begin{aligned} \mathbf{y}_t - \mathbf{w}_t &= \frac{\alpha_{t-2}^2}{\alpha_{t-1}} \mathbf{v}_{t-1} \\ \mathbf{v}_t &= \frac{\alpha_{t-2}^2}{\alpha_t \alpha_{t-1}} \mathbf{v}_{t-1} - \eta \nabla \mathcal{L}(\mathbf{x}_t) \end{aligned} \tag{7}$$

Finally, observe that

$$\begin{aligned} \mathbf{x}_t &= \mathbf{w}_t + \tau_t(\mathbf{y}_t - \mathbf{w}_t) \\ &= \mathbf{w}_t + \frac{1}{\alpha_t}(\mathbf{y}_t - \mathbf{w}_t) \end{aligned}$$

using equation (7):

$$= \mathbf{w}_t + \frac{\alpha_{t-2}^2}{\alpha_{t-1}\alpha_t} \mathbf{v}_{t-1}$$

Thus, defining $\beta_1 = 0$ and $\beta_t = \frac{\alpha_{t-2}^2}{\alpha_t \alpha_{t-1}}$ for all $t \geq 2$ yields the updates for all $t \geq 1$:

$$\begin{aligned} \mathbf{x}_t &= \mathbf{w}_t + \beta_t \mathbf{v}_{t-1} \\ \mathbf{v}_t &= \beta_t \mathbf{v}_{t-1} - \eta \nabla \mathcal{L}(\mathbf{x}_t) \\ \mathbf{w}_{t+1} &= \mathbf{w}_t + \mathbf{v}_t \end{aligned}$$

which is exactly the form of the two-step momentum integrator used in Algorithm 12.

15.1 Acceleration analysis

Theorem 15.2. *Suppose $\mathcal{L}$ is an H-smooth convex function. Let $\eta \leq \frac{1}{H}$. Then if $\mathbf{w}_\star = \operatorname{argmin} \mathcal{L}(\mathbf{w})$, Algorithm 13 guarantees:*

$$\mathcal{L}(\mathbf{w}_{T+1}) - \mathcal{L}(\mathbf{w}_\star) \leq \frac{9\|\mathbf{w}_\star - \mathbf{w}_0\|^2}{2T^2\eta}$$

In particular, with $\eta = \frac{1}{H}$, we have:

$$\mathcal{L}(\mathbf{w}_{T+1}) - \mathcal{L}(\mathbf{w}_\star) \leq O\left(\frac{H\|\mathbf{w}_\star - \mathbf{w}_0\|^2}{T^2}\right)$$

Before proving the Theorem, let's show a small proposition to gain some intuition for how the α_t behave:

Proposition 15.3. *For all $t \geq 1$,*

$$\frac{t^2}{9} \leq \sum_{i=1}^{t} \alpha_i \leq t^2$$

Further, $\alpha_t = \sum_{i=1}^{t} \alpha_i$.

Proof. The second part of the Proposition is immediate from the definition of α_t. The tricky part is proving the first statement. For this, we proceed by induction. The base case for $t = 1$ is clear from definition of α_1. Let us assume the statement holds for some t. By definition of α_{t+1}, we have:

$$\begin{aligned} \alpha_{t+1} &= \frac{1 + \sqrt{1 + 4\sum_{i=1}^{t} \alpha_i}}{2} \\ &\leq 1 + \sqrt{\sum_{i=1}^{t} \alpha_i} \end{aligned}$$

So therefore:

$$\sum_{i=1}^{t+1} \alpha_i \leq \sum_{i=1}^{t} \alpha_i + 1 + \sqrt{\sum_{i=1}^{t} \alpha_i}$$

using the induction assumption:

$$\begin{aligned} &\leq t^2 + 1 + t \\ &\leq (t+1)^2 \end{aligned}$$

For the lower bound, we have

$$\alpha_{t+1} \geq \sqrt{\sum_{i=1}^{t} \alpha_i}$$

so that:

$$\begin{aligned} \sum_{i=1}^{t+1} \alpha_i &\geq \sum_{i=1}^{t} \alpha_i + \sqrt{\sum_{i=1}^{t} \alpha_i} \\ &\geq \frac{t^2}{9} + \frac{t}{3} \end{aligned}$$

using $t \geq 1$:

$$\begin{aligned} &\geq \frac{t^2}{9} + \frac{2t}{9} + \frac{1}{9} \\ &= \frac{(t+1)^2}{9} \end{aligned}$$

□

This shows that $\sum_{i=1}^{t} \alpha_i$ grows quadratically in t. Now, we are ready to prove Theorem 15.2:

Proof of Theorem 15.2. Let's start by examining the quantity $\sum_{t=1}^{T} \alpha_t(\mathcal{L}(\mathbf{x}_t) - \mathcal{L}(\mathbf{w}_\star))$. By convexity, we have $\mathcal{L}(\mathbf{x}_t) - \mathcal{L}(\mathbf{w}_\star) \leq \langle \nabla \mathcal{L}(\mathbf{x}_t), \mathbf{x}_t - \mathbf{w}_\star \rangle$, so

$$\begin{aligned} \sum_{t=1}^{T} \alpha_t(\mathcal{L}(\mathbf{x}_t) - \mathcal{L}(\mathbf{w}_\star)) &\leq \sum_{t=1}^{T} \alpha_t \langle \nabla \mathcal{L}(\mathbf{x}_t), \mathbf{x}_t - \mathbf{w}_\star \rangle \\ &= \sum_{t=1}^{T} \alpha_t \langle \nabla \mathcal{L}(\mathbf{x}_t), \mathbf{x}_t - \mathbf{y}_t \rangle + \sum_{t=1}^{T} \alpha_t \langle \nabla \mathcal{L}(\mathbf{x}_t), \mathbf{y}_t - \mathbf{w}_\star \rangle \\ &= \sum_{t=1}^{T} \langle \nabla \mathcal{L}(\mathbf{x}_t), \alpha_t(\mathbf{x}_t - \mathbf{y}_t) \rangle + \sum_{t=1}^{T} \langle \mathbf{g}_t, \mathbf{y}_t - \mathbf{w}_\star \rangle \end{aligned}$$

Now, notice that the second sum $\sum_{t=1}^{T} \langle \mathbf{g}_t, \mathbf{y}_t - \mathbf{w}_\star \rangle$ can be bounded by Theorem 15.1:

$$\sum_{t=1}^{T} \langle \mathbf{g}_t, \mathbf{y}_t - \mathbf{w}_\star \rangle \leq \frac{\|\mathbf{w}_\star - \mathbf{y}_1\|^2}{2\eta} + \frac{\eta}{2} \sum_{t=1}^{T} \|\mathbf{g}_t\|^2$$

Note that even though the relationship between $\mathbf{g}_t$ and $\mathbf{y}_t$ is somewhat complicated, *this is totally irrelevant* because Theorem 15.1 works for *any* sequence of $\mathbf{g}_t$.

Next, look at the term $\alpha_t(\mathbf{x}_t - \mathbf{y}_t)$. Let's do some tricky algebra using the definition of $\mathbf{x}_t$:

$$\mathbf{x}_t = (1-\tau_t)\mathbf{w}_t + \tau_t\mathbf{y}_t = \left(1 - \frac{\alpha_t}{\sum_{i=1}^t \alpha_i}\right)\mathbf{w}_t + \frac{\alpha_t}{\sum_{i=1}^t \alpha_i}\mathbf{y}_t$$
$$\left(\sum_{i=1}^t \alpha_i\right)\mathbf{x}_t = \left(\left(\sum_{i=1}^t \alpha_i\right) - \alpha_t\right)\mathbf{w}_t + \alpha_t\mathbf{y}_t$$

subtract $\left(\sum_{i=1}^{t-1}\alpha_i\right)\mathbf{x}_t$ and $\alpha_t\mathbf{y}_t$ from both sides:

$$\begin{aligned}\alpha_t\mathbf{x}_t - \alpha_t\mathbf{y}_t &= \left(\left(\sum_{i=1}^t \alpha_i\right) - \alpha_t\right)\mathbf{w}_t - \left(\sum_{i=1}^{t-1}\alpha_i\right)\mathbf{x}_t \\ &= \left(\sum_{i=1}^{t-1}\alpha_i\right)(\mathbf{w}_t - \mathbf{x}_t)\end{aligned}$$

Therefore, we have:

$$\langle \nabla\mathcal{L}(\mathbf{x}_t), \alpha_t(\mathbf{x}_t - \mathbf{y}_t)\rangle = \left(\sum_{i=1}^{t-1}\alpha_i\right)\langle \nabla\mathcal{L}(\mathbf{x}_t), \mathbf{w}_t - \mathbf{x}_t\rangle$$

Now, let's use convexity again: we have $\mathcal{L}(\mathbf{w}_t) \geq \mathcal{L}(\mathbf{x}_t) + \langle \nabla\mathcal{L}(\mathbf{x}_t), \mathbf{w}_t - \mathbf{x}_t\rangle$, so:

$$\langle \nabla\mathcal{L}(\mathbf{x}_t), t(\mathbf{x}_t - \mathbf{y}_t)\rangle \leq \left(\sum_{i=1}^{t-1}\alpha_i\right)(\mathcal{L}(\mathbf{w}_t) - \mathcal{L}(\mathbf{x}_t))$$

Going back and putting this all together, we have shown:

$$\sum_{t=1}^T \alpha_t(\mathcal{L}(\mathbf{x}_t) - \mathcal{L}(\mathbf{w}_\star)) \leq \frac{\|\mathbf{w}_\star - \mathbf{y}_1\|^2}{2\eta} + \frac{\eta}{2}\sum_{t=1}^T \|\mathbf{g}_t\|^2 + \sum_{t=1}^T\left(\sum_{i=1}^{t-1}\alpha_i\right)(\mathcal{L}(\mathbf{w}_t) - \mathcal{L}(\mathbf{x}_t))$$

Now, eventually we are going to want the last sum to telescope. So far there are two obstacles. First, there is a $\mathbf{w}$ instead of a $\mathbf{x}$, and second the coefficients on the $\mathcal{L}(\mathbf{w}_t)$ and $\mathcal{L}(\mathbf{x}_t)$ are the same. Let's fix the second problem first: subtract $\sum_{t=1}^T \alpha_t\mathcal{L}(\mathbf{x}_t)$ from both sides to get,

$$-\sum_{t=1}^T \alpha_t\mathcal{L}(\mathbf{w}_\star) \leq \frac{\|\mathbf{w}_\star - \mathbf{y}_1\|^2}{2\eta} + \frac{\eta}{2}\sum_{t=1}^T \|\mathbf{g}_t\|^2 + \sum_{t=1}^T\left(\sum_{i=1}^{t-1}\alpha_i\right)\mathcal{L}(\mathbf{w}_t) - \left(\sum_{i=1}^{t}\alpha_i\right)\mathcal{L}(\mathbf{x}_t)$$

Now, we can see that if it weren't for the $\mathbf{x}$ vs $\mathbf{w}$ mismatch, the last sum would indeed telescope. Let's use smoothness to relate $\mathcal{L}(\mathbf{x}_t)$ to $\mathcal{L}(\mathbf{w}_{t+1})$. So long as $\eta \leq \frac{1}{H}$, we have:

$$\begin{aligned}\mathcal{L}(\mathbf{w}_{t+1}) &\leq \mathcal{L}(\mathbf{x}_t) - \frac{\eta}{2}\|\nabla\mathcal{L}(\mathbf{x}_t)\|^2 \\ -\mathcal{L}(\mathbf{x}_t) &\leq -\mathcal{L}(\mathbf{w}_{t+1}) - \frac{\eta}{2}\|\nabla\mathcal{L}(\mathbf{x}_t)\|^2\end{aligned}$$

Thus, we have:

$$-\sum_{t=1}^T \alpha_t\mathcal{L}(\mathbf{w}_\star) \leq \frac{\|\mathbf{w}_\star - \mathbf{y}_1\|^2}{2\eta} + \frac{\eta}{2}\sum_{t=1}^T \|\mathbf{g}_t\|^2 + \sum_{t=1}^T\left[\left(\sum_{i=1}^{t-1}\alpha_i\right)\mathcal{L}(\mathbf{w}_t) - \left(\sum_{i=1}^{t}\alpha_i\right)\mathcal{L}(\mathbf{w}_{t+1}) - \left(\sum_{i=1}^{t}\alpha_i\right)\frac{\eta}{2}\|\nabla\mathcal{L}(\mathbf{x}_t)\|^2\right]$$

Finally, the sum $\sum_{t=1}^T\left[\left(\sum_{i=1}^{t-1}\alpha_i\right)\mathcal{L}(\mathbf{w}_t) - \left(\sum_{i=1}^{t}\alpha_i\right)\mathcal{L}(\mathbf{w}_{t+1})\right]$ telescopes, yielding:

$$-\sum_{t=1}^{T}\alpha_t\mathcal{L}(\mathbf{w}_\star) \leq \frac{\|\mathbf{w}_\star-\mathbf{y}_1\|^2}{2\eta}+\frac{\eta}{2}\sum_{t=1}^{T}\|\mathbf{g}_t\|^2-\sum_{t=1}^{T}\alpha_t\mathcal{L}(\mathbf{w}_{T+1})-\sum_{t=1}^{T}\left(\sum_{i=1}^{t}\alpha_i\right)\frac{\eta}{2}\|\nabla\mathcal{L}(\mathbf{x}_t)\|^2$$

rearrange:

$$\left(\sum_{t=1}^{T}\alpha_t\right)(\mathcal{L}(\mathbf{w}_{T+1})-\mathcal{L}(\mathbf{w}_\star)) \leq \frac{\|\mathbf{w}_\star-\mathbf{y}_1\|^2}{2\eta}+\frac{\eta}{2}\sum_{t=1}^{T}\|\mathbf{g}_t\|^2-\sum_{t=1}^{T}\left(\sum_{i=1}^{t}\alpha_i\right)\frac{\eta}{2}\|\nabla\mathcal{L}(\mathbf{x}_t)\|^2$$

Now, let's finally start to use the definition of α_t. Note that we have $\alpha_t^2=\sum_{i=1}^{t}\alpha_i$. Thus:

$$\left(\sum_{t=1}^{T}\alpha_t\right)(\mathcal{L}(\mathbf{w}_{T+1})-\mathcal{L}(\mathbf{w}_\star)) \leq \frac{\|\mathbf{w}_\star-\mathbf{y}_1\|^2}{2\eta}+\frac{\eta}{2}\sum_{t=1}^{T}\|\mathbf{g}_t\|^2-\sum_{t=1}^{T}\alpha_t^2\frac{\eta}{2}\|\nabla\mathcal{L}(\mathbf{x}_t)\|^2$$

insert the definition of $\mathbf{g}_t$:

$$\begin{aligned}&= \frac{\|\mathbf{w}_\star-\mathbf{y}_1\|^2}{2\eta}+\frac{\eta}{2}\sum_{t=1}^{T}\alpha_t^2\|\nabla\mathcal{L}(\mathbf{x}_t)\|^2-\sum_{t=1}^{T}\alpha_t^2\frac{\eta}{2}\|\nabla\mathcal{L}(\mathbf{x}_t)\|^2\\ &= \frac{\|\mathbf{w}_\star-\mathbf{y}_1\|^2}{2\eta}\end{aligned}$$

using Proposition 15.3:

$$\frac{T^2}{9}(\mathcal{L}(\mathbf{w}_{T+1})-\mathcal{L}(\mathbf{w}_\star)) \leq \frac{\|\mathbf{w}_\star-\mathbf{y}_1\|^2}{2\eta}$$

from which the result follows by observing that $\mathbf{y}_1=\mathbf{w}_1$. □

16 Lower Bounds

In the previous section we have concentrated on algorithms that achieve provable convergence guarantees. Some of these algorithms were "better" than others: for example, in the case of deterministic smooth convex optimization, we saw that accelerated gradient descent appears to be strictly superior to ordinary gradient descent. This suggests a natural question: is there something better out there? All of our work so far has been on variants of gradient descent. Maybe if we were a little more creative, it would be possible to design a significantly better algorithm!

It turns out that in broad strokes the answer is "no": when considering the types of optimization problems we have looked at so far, it is impossible to design an algorithm that is *uniformly better* than gradient descent. Here, we consider an algorithm A to be *uniformly better* than algorithm B if *for all* optimization problems in some class of interest (e.g. smooth convex losses), A performs much better than B. Of course, this still allows for algorithms that are *non-uniform* improvements. That is, it might be that A performs about the same as B on *some* optimization problems, but on others it is able to perform much better. Since we will show that the algorithms we have seen so far are already optimal, much of the recent progress in optimization algorithm is focused on designing methods that have such non-uniform improvement, with the hope that problems that appear in reality lie in the set of problems that the new algorithm has an advantage on.

Let us take a minute to appreciate what it means to prove that an optimization algorithm cannot be improved. Our convergence guarantees all look like *upper bounds* of the following form: for all distributions over functions ℓ satisfying some conditions C, our algorithm A is outputs a point $\hat{w}$ such that satisfies some convergence property $F(\hat{w}) \leq B$. For example, in the case of stochastic convex optimization, C is the condition that ℓ must be G-Lipschitz, convex and satisfy $\|w_\star\| \leq R$, $F(\hat{w}) = \mathbb{E}[\mathcal{L}(\hat{w}) - \mathcal{L}(w_\star)]$ and $B = \frac{GR}{\sqrt{T}}$. In the case of non-stochastic non-convex smooth optimization, C is the set of deterministic distributions (i.e. $\ell = \mathcal{L}$) such that $\mathcal{L}$ is H-smooth and satisfies $\mathcal{L}(w_1) - \mathcal{L}(w_\star) \leq \Delta$, $F(\hat{w}) = \mathbb{E}[\|\nabla \mathcal{L}(\hat{w})\|]$, and $B = \frac{\sqrt{\Delta H}}{\sqrt{T}}$.

Thus, in order to prove the optimality of our optimization algorithms we need a *lower bound* of the form: *for all* algorithms A, *there exists* a distribution over functions ℓ satisfying condition C such that A's output $\hat{w}$ satisfies $F(\hat{w}) \geq B$.

We will tackle this goal first for *deterministic smooth convex losses*, and then for *stochastic smooth convex losses*. On the homework you will extend the first result to deterministic non-smooth convex losses.

Similar optimality results also hold for the non-convex case, but bounds were only discovered a few years ago (see Arjevani, Carmon, J. C. Duchi, Foster, Srebro, et al. 2019; Carmon et al. 2019; Carmon et al. 2021). The high-level ideas of these proofs are very similar to the convex case, but the technical details are significantly more involved.

16.1 Deterministic Smooth Stochastic Convex Losses

In order to prove a lower bound for the deterministic case, we need to formalize some assumptions about how our algorithm is able to access the loss function $\mathcal{L}$. Otherwise, it is difficult to be precise about how the algorithm can or cannot proceed. That is, we need to specify what the optimization algorithm can take as input. Can it see a lookup table of pairs $(x, \mathcal{L}(x))$? Does it get as input some C-source code describing how to compute the function $\mathcal{L}$? The way in which $\mathcal{L}$ is provided to the algorithm can have drastic impact on how easy it is to optimize. For example, if $\mathcal{L}$ is provided as some source for computing $\mathcal{L}$, then it is conceivable that a very complicated algorithm could quickly perform some analysis on this code to compute the minimizer very fast.

We formalize our access to $\mathcal{L}$ via the *oracle model*. That is, our algorithm has access to an *oracle* function $\mathcal{O}$ such that $\mathcal{O}(w) = (\mathcal{L}(w), \nabla \mathcal{L}(w))$. $\mathcal{O}$ is sometimes also called a *first order oracle*, in contrast to a *second order oracle* which might return $(\mathcal{L}(w), \nabla \mathcal{L}(w), \nabla^2 \mathcal{L}(w))$. To measure the performance of our algorithm, we will count the *number of oracle calls* required to find a $\hat{w}$ with small $\mathcal{L}(\hat{w})$. We say that an algorithm A that accesses $\mathcal{L}$ only through a first order oracle is a *first order algorithm*. Clearly, every algorithm we have discussed so far in class is a first order algorithm.

With this formalism, it makes sense to define the *iterates* of a first-order algorithm A as:

Definition 16.1. *Let A be a first-order algorithm and let $\mathcal{O}$ be a first-order oracle. Then the* iterates *of A are the sequence of inputs $w_1, \ldots, w_T$ that A provides to $\mathcal{O}$.*

We will further make another restriction on algorithms A: we will assume that A is *zero respecting*, defined formally below:

Definition 16.2. *Let A be a first-order algorithm. Given a first-order oracle $\mathcal{O}$, let $w_1, \dots, w_T$ be the iterates of A and let the outputs be $\mathcal{O}(w_t) = (\mathcal{L}_t, g_t)$. We say that A is zero-respecting if for all $\mathcal{O}$, $w_t[i] = 0$ for all i such that $g_\tau[i] = 0$ for all $\tau < t$.*

More informally, an algorithm is zero-respecting if it starts at the origin $w_1 = 0$, and from then on only changes an ith coordinate of w_t to be non-zero if it has encountered a gradient g_τ whose ith coordiante is non-zero. Intuitively, the algorithm is not allowed to randomly "guess" values of w - it can only make a coordinate non-zero if there is some "evidence" that it should be non-zero. While this may seem restrictive, notice that every single algorithm we have considered so far (and in fact every algorithm we are going to consider later in the course) is a zero-respecting algorithm. Nevertheless, it is possible without too much trouble to extend proofs that only hold for zero-respecting algorithms to arbitrary deterministic algorithms: because such algorithms are still not allowed to "guess", one can cleverly rotate coordinate systems to view any such algorithms as essentially zero-respecting. With somewhat more trouble, it is also possible to extend the techniques to randomized algorithms.

Formalizing a component of the previous definition, we define the *iterates* of an algorithm A as the sequence of inputs $w_1, \dots, w_T$ to the first-order

Our lower bound now takes the following form:

Theorem 16.3. *Let T, H and R be given. Then there is a convex function $\mathcal{L} : \mathbb{R}^T \to \mathbb{R}$ that is H-smooth and satisfies $\|w_\star\| \leq R$ for $w_\star = \operatorname{argmin} \mathcal{L}$ such that for any first-order zero-respecting algorithm A, the first T iterates $w_1, \dots, w_T$ all satisfy:*

$$\mathcal{L}(w_t) - \mathcal{L}(w_\star) \geq \frac{HR^2}{8T^2}$$

Before proving this theorem, notice that it is actually stronger than we really require for a lower bound: the function $\mathcal{L}$ is *universal* in the sense that it *does not depend on* A. That is, this function is in some sense so "difficult" that any first-order zero-respecting algorithm will have trouble optimizing it. However, if we were to drop the zero-respecting condition, we would need to retreat to the easier task of defining a function that is tailored to the specific operation of A.

Notice also one peculiarity of THeorem 16.3: the "hard" function $\mathcal{L}$ is a T-dimensional function. If we force $\mathcal{L}$ to be a d-dimensional function for some bounded $d \leq T$, it turns out that the Theorem is false and that in fact there *is* an algorithm that outperforms gradient descent.

Without further ado, let's see the proof.

Proof. Define the function $\mathcal{L} : \mathbb{R}^d \to \mathbb{R}$ to be the following quadratic:

$$\mathcal{L}(w) = \frac{H}{8}(w[1] - \frac{R}{\sqrt{d}})^2 + \frac{H}{8}(w[2] - w[1])^2 + \frac{H}{8}(w[3] - w[2])^2 + \dots + \frac{H}{8}(w[d] - w[d-1])^2$$

We will later set $d = T$, but for now let us just consider arbitrary d to get a handle on how the function behaves.

Let us verify that $\mathcal{L}$ is convex, H-smooth and satisfies $\|w_\star\| \leq R$. To this end, the gradient of $\mathcal{L}$ is:

$$\nabla \mathcal{L}(w) = \frac{H}{4}(2w[1] - \frac{R}{\sqrt{d}} - w[2], 2w[2] - w[1] - w[3], \dots, 2w[d-1] - w[d] - w[d-2], w[d] - w[d-1])$$

Clearly, the gradient is zero when $w[1] = w[2] = \dots = w[d] = \frac{R}{\sqrt{d}}$. Thus $w_\star = (R/\sqrt{d}, R/\sqrt{d}, \dots)$ and so $\|w_\star\| \leq \sqrt{dR^2/d} = R$. Further, clearly we have $\mathcal{L}(w_\star) = 0$.

Next, the Hessian of $\mathcal{L}$ is:

$$\nabla^2 \mathcal{L}(w) = \frac{H}{2} \begin{pmatrix} 1 & -\frac{1}{2} & 0 & 0 & 0 & \dots \\ -\frac{1}{2} & 1 & -\frac{1}{2} & 0 & 0 & \dots \\ 0 & -\frac{1}{2} & 1 & -\frac{1}{2} & 0 & \dots \\ \vdots & & \ddots & \ddots & & \vdots \\ \dots & 0 & 0 & -\frac{1}{2} & 1 & -\frac{1}{2} \\ \dots & 0 & 0 & 0 & -\frac{1}{2} & 1 \end{pmatrix}$$

Let's consider an arbitrary vector v and compute $v^\top \nabla^2 \mathcal{L}(w) v$:

$$\begin{aligned} v^\top \nabla^2 \mathcal{L}(w) v &= \frac{H}{2}(v[1]^2 - v[1]v[2] + v[2]^2 - v[2]v[3] + \cdots - v[d-1]v[d] + v[d]^2) \\ &= \frac{H}{2}(v[1]^2 + (v[2] - v[1])^2 + (v[2] - v[3])^2 + \cdots + (v[d] - v[d-1])^2) \end{aligned}$$

This clearly implies $v^\top \nabla^2 \mathcal{L}(w) v \geq 0$ so that $\mathcal{L}$ is convex.

Furthermore by Young inequality, $-v[i]v[i+1] \leq \frac{v[i]^2}{2} + \frac{v[i+1]^2}{2}$ so that:

$$\begin{aligned} v^\top \nabla^2 \mathcal{L}(w) v &= \frac{H}{2}(v[1]^2 - v[1]v[2] + v[2]^2 - v[2]v[3] + \cdots - v[d-1]v[d] + v[d]^2) \\ &\leq \frac{H}{2}(v[1]^2 + \frac{v[1]^2}{2} + \frac{v[2]^2}{2} + v[2]^2 + \frac{v[2]^2}{2} + \frac{v[3]^2}{2} + \cdots + \frac{v[d]^2}{2} + v[d]^2) \\ &\leq \frac{H}{2} \sum_{i=1}^{d} 2v[i]^2 \\ &\leq H\|v\|^2 \end{aligned}$$

so that $\mathcal{L}$ is H-smooth.

Now, need a final important property of $\mathcal{L}$: for any w such that $w[d] = 0$:

$$\mathcal{L}(w) - \mathcal{L}(w_\star) \geq \frac{H}{8d^2}$$

Let us prove this property. If w satisfies $w[d] = 0$, then we can write $\mathcal{L}(w)$ as:

$$\mathcal{L}(w) = \frac{H}{8}(w[1] - \frac{R}{\sqrt{d}})^2 + \frac{H}{8}(w[2] - w[1])^2 + \frac{H}{8}(w[3] - w[2])^2 + \cdots + \frac{H}{8}(w[d-1] - w[d-2])^2 + \frac{H}{8}w[d-1]^2$$

The gradient subject to $w[d] = 0$ is:

$$\nabla \mathcal{L}(w) = \frac{H}{4}(2w[1] - \frac{R}{\sqrt{d}} - w[2], 2w[2] - w[1] - w[3], \ldots, 2w[d-2] - w[d-1] - w[d-3], 2w[d-1] - w[d-2])$$

This is zero when $w[d-i] = \frac{iR}{d\sqrt{d}}$: this can be checked by observing that for $i \in [2, d-2]$,

$$2w[d-i] - w[i-1] - w[i+1] = 2\frac{iR}{d\sqrt{d}} - \frac{(i-1)R}{d\sqrt{d}} - \frac{(i+1)R}{d\sqrt{d}} = 0$$

and

$$2w[d-1] - w[d-2] = 2\frac{iR}{d\sqrt{d}} - \frac{2iR}{d\sqrt{d}} = 0$$

and

$$\begin{aligned} 2w[1] - \frac{R}{\sqrt{d}} - w[2] &= 2\frac{(d-1)R}{d\sqrt{d}} - \frac{dR}{d\sqrt{d}} - \frac{(d-2)R}{d\sqrt{d}} \\ &\quad \frac{(2d - 2 - d - d + 2)R}{d\sqrt{d}} \\ &= 0 \end{aligned}$$

For this value of w, we have

$$\begin{aligned} \mathcal{L}(w) &= \frac{HR^2}{8d^3}(((d-1) - d)^2 + ((d-3) - (d-2))^2 + \cdots + (1-2)^2 + 1^2) \\ &= \frac{HR^2}{8d^2} \end{aligned}$$

The bound on $\mathcal{L}(w) - \mathcal{L}(w_\star)$ now follows from observing that $\mathcal{L}(w_\star) = 0$.

With these properties out of the way, we can proceed to show the lower bound. The strategy will be to show that for any $t \leq d$, we must have $w_t[i] = 0$ for all $i \geq t$ as a consequence of the zero-respecting property. Therefore, if $T \leq d$, we have $w_t[d] = 0$ for all $t \leq d$ and as a result $\mathcal{L}(w) - \mathcal{L}(w_\star) \geq \frac{H}{8d^2}$. Thus, by setting $d = T$, we conclude the desired result.

To show this, we now leverage the peculiar structure of $\mathcal{L}$. Notice that by assumption $w_1 = 0$. Next, for purposes of induction, suppose that for some t, $w_\tau[i] = 0$ for all $i \geq \tau$ for all τ $let < d$. Then observe that if $t+1 \leq i \leq d-1$, we have $w_\tau[i] = w_\tau[i+1] = w_\tau[i-1] = 0$ since $i - 1 \geq t$. Thus:

$$\nabla \mathcal{L}(w_\tau)[i] = 2w_\tau[i] - w_\tau[i-1] - w_\tau[i+1] = 0$$

Further, if $t + 1 = d$, we still have $w_\tau[t+1] = w_\tau[t] = 0$ so that:

$$\nabla \mathcal{L}(w_\tau)[t+1] = w_\tau[t+1] - w_\tau[t] = 0$$

So no matter what, $\nabla \mathcal{L}(w_\tau)[i] = 0$ for all $t + 1 \leq i \leq d$ for all $\tau \leq t$. Thus, by the zero-respecting property, $w_{t+1}[i] = 0$ for all $i \geq t + 1$ as desired. □

16.2 deterministic non-zero respecting algorithms

In this section, we briefly sketch how to extend the above techniques to any deterministic algorithm regardless of whether it is zero-respecting or not. Here by deterministic we mean that the algorithm has no *internal* randomness. That is, the first t outputs of the oracle $(l_1, g_1), \ldots, (l_t, g_t)$ uniquely determine the $t = 1$st iterate w_{t+1}. In contrast, a randomized algorithm might choose w_{t+1} from e.g. a Gaussian centered at some particular mean μ_{t+1}.

For such an algorithm, we consider a *family* of possible loss functions in a $2T$ dimensional space:

$$\mathcal{L}_v(w) = \frac{H}{8}\left(\langle w, v_1\rangle - \frac{R}{\sqrt{T}}\right)^2 + \frac{H}{8}\sum_{i=1}^{T-1}(\langle w, v_{i+1} - v_i\rangle)^2$$

where $v_1, \ldots, v_T$ are a sequence of orthogonal unit vectors. By essentially the same arguments as in Theorem 16.3, we have that $w_\star^v = \operatorname{argmin} \mathcal{L}_v$ satisfies $\|w_\star^v\| \leq R$ for all v, and $\mathcal{L}_v$ is convex and H-smooth.

We claim that for any deterministic first-order algorithm A, there is a $v_1, \ldots, v_T$ such that when A is provided a first-order oracle for $\mathcal{L}_v$, the first T iterates of A satisfy:

$$\mathcal{L}_v(w_t) - \mathcal{L}_v(w_\star^v) \geq \frac{HR^2}{8T^2}$$

Notice now that the loss $\mathcal{L}_v$ depends on A, while previously it was a universal loss that stymied any zero-respecting algorithm.

We may without loss of generality assume $w_1 = 0$ by potentially translating coordinate systems. Then, to construct $v_1, \ldots, v_T$, first set $v_1 = (1, 0, 0, \ldots)$. Next, notice that since $w_1 = 0$, $\nabla \mathcal{L}_v(w_1)$ and $\mathcal{L}_v(w_1)$ do not depend on the choices $v_2, \ldots, v_T$. Thus, w_2 is purely a function of v_1. In particular, if v_j is orthogonal to w_t for all $t \leq j$, then $\langle w_t, v_j\rangle = 0$ for all $t \leq j$ and so

$$\begin{aligned}\langle \nabla \mathcal{L}_v(w_t), v_j\rangle &= \frac{H}{4}\left(\langle w, v_1\rangle - \frac{R}{\sqrt{T}}\right)\langle v_1, v_j\rangle + \frac{H}{4}\sum_{i=1}^{T-1}(\langle w, v_{i+1} - v_i\rangle)\langle v_{i+1} - v_i, v_j\rangle \\ &= \frac{H}{4}\left(\langle w, v_1\rangle - \frac{R}{\sqrt{T}}\right)\langle v_1, v_j\rangle + \frac{H}{4}\sum_{i=1}^{t-1}(\langle w, v_{i+1} - v_i\rangle)\langle v_{i+1} - v_i, v_j\rangle\end{aligned}$$

In particular $\langle \nabla \mathcal{L}_v(w_t), v_j\rangle = 0$ for all $j > t$. Thus, in general w_t cannot depend on $v_t, \ldots, v_T$.

Thus suggests a simple way to choose v_t: set v_t to be an arbitrary unit vector orthogonal to both $v_1, \ldots, v_{t-1}$ and $w_1, \ldots, w_t$. Since the space is $2T$ dimensional there is always such a vector, and w_{t+1} will then be a function only of $v_1, \ldots, v_t$ so that the process is well-defined. We then maintain the property $\langle w_t, v_T\rangle = 0$ for all T and $\mathcal{L}_v(w) \geq \frac{HR^2}{8T^2}$ for any w with $\langle w, v_T\rangle = 0$ so that the lower-bound argument proceeds nearly identically.

17 Lower bounds for stochastic convex optimization

With the deterministic setting out of the way, let's now consider the *stochastic* setting. Here, we will adopt a rather different proof strategy. Instead of exhibiting a fixed hard function $\mathcal{L}$, we will actually consider a relatively "easy" looking function $\mathcal{L}$, but have the *distribution* of the ℓs have so much noise that it is difficult to identify $\mathcal{L}$ accurately. As a result, we will actually be able to be much much less restrictive on the types of algorithms we allow. Rather than restricting ourselves to the oracle model of access and zero-respecting algorithms, we consider an algorithm to be *any* function that takes as input a dataset of T samples $z_1, \ldots, z_T$ and a convex, Lipschitz and smooth loss function $\ell(w, z)$ and outputs a point $\hat{w}$. As a result, our lower bound is actually significantly more powerful than simply a bound on optimization algorithms: it can be viewed as a bound on how much data is required for learning in general.

The key idea behind the lower bound is construct two different distributions, D^+ and D^- that are *almost the same* in the sense that an i.i.d. sample $(z_1, \ldots, z_T) \sim D^+$ is very difficult to distinguish from an i.i.d. sample $(z_1, \ldots, z_T) \sim D^-$, and yet the expected losses $\mathcal{L}^+(w) = \mathbb{E}_{z \sim D^+}[\ell(w, z)]$ and $\mathcal{L}^- = \mathbb{E}_{z \sim D^-}[\ell(w, z)]$ are very different in the sense that their respective minima are very far apart. If we could build two such distributions, then intuitively no algorithm would be able to tell based on the data which distribution was the "true" one. Thus the algorithm is in some sense forced to "guess" whether the true loss $\mathcal{L}$ is equal to $\mathcal{L}^+$ or $\mathcal{L}^-$, and so it must output an poorly performing $\hat{w}$ on at least one of $\mathcal{L}^+$ or $\mathcal{L}^-$.

In order to pursue this strategy, we need some formal descriptions of "closeness" of distributions. The most relevant metric is the *total variation distance*:

Definition 17.1. *Suppose P and Q are two probability distributions on a common set of events E. Then the* total variation distance *between P and Q is:*

$$TV(P, Q) = \sup_{A \in E} |P(A) - Q(A)|$$

In the case that we are considering distributions with densities, the following is a more typical characterization of total variation distance:

Proposition 17.2. *If P and Q are probability distributions with densities $p(x)$ and $q(x)$, then*

$$TV(P, Q) = \frac{1}{2} \int_X |p(x) - q(x)| dx$$

If instead P and Q are distributions over a countable set with probability mass functions p and q, then:

$$TV(P, Q) = \frac{1}{2} \sum_X |p(x) - q(x)|$$

The value of the total variation distance is that it provides a way to quantify how "different" any function can be when provided data from two similar distributions. In particular, consider random variables $X_P \sim P$ and $X_Q \sim Q$ and an arbitrary function $F(X)$ such that $F(X) \in [-1, 1]$. Then:

$$|\mathbb{E}[F(X_P)] - \mathbb{E}[F(X_Q)]| = \left| \int (F(x)p(x) - F(x)q(x)) dx \right| \leq \int |F(x)p(x) - F(x)q(x)| dx \leq \int |p(x) - q(x)| dx \leq 2TV(P, Q)$$

Thus, if we let F represent the output of our algorithm, we will be able to show that if the distributions D^+ and D^- have small total variation distance, the the algorithm must output similar points.

In order to bound the total variation distance, we introduce another "distance like" function of probability measures that is sometimes easier to compute: the Kullback–Leibler divergence (KL divergence):

Definition 17.3. *Suppose P and Q are distributions over a countable set X with probability mass functions p and q. Then the KL-divergence is defined as:*

$$KL(P, Q) = \sum_X p(x) \log \left(\frac{p(x)}{q(x)} \right)$$

The KL divergence, unlike the total variation distance, is not a "metric" in a formal sense because it is not symmetric: $KL(P,Q) \neq KL(Q,P)$. However, it is "distance-like" in the intuitive sense that $KL(P,Q) > 0$ for $P \neq Q$ and $KL(P,P) = 0$. More importantly for our purposes, the KL divergence satisfies the following useful property, which is a special case of what is more generally called the "chain rule" for KL divergence:

Proposition 17.4. *Suppose P_1, and Q_1 be distributions over a countable set X and suppose P_2 and Q_2 are distributions over a countable set Y with probability mass functions p_1, p_2, q_1, q_2. Let $P_1 \times P_2$ be the product distribution over the sample $(z_1, z_2) \in X \times Y$ with $z_1 \sim P_1$ and $z_2 \sim P_2$ and z_1 independent of z_2, and similarly for $Q_1 \times Q_2$. Then:*

$$KL(P_1 \times P_2, Q_1 \times Q_2) = KL(P_1, Q_1) + KL(P_2, Q_2)$$

Proof. The probability mass function for $P_1 \times P_2$ is $p(x,y) = p_1(x)p_2(y)$. Similarly, the probability mass function for $Q_1 \times Q_2$ is $q(x,y) = q_1(x)q_2(y)$. Thus:

$$\begin{aligned}
KL(P_1 \times P_2, Q_2 \times Q_2) &= \sum_X \sum_Y p(x,y) \log\left(\frac{p(x,y)}{q(x,y)}\right) \\
&= \sum_X \sum_Y p_1(x)p_2(y) \log\left(\frac{p_1(x)p_2(y)}{q_1(x)q_2(y)}\right) \\
&= \sum_X \sum_Y p_1(x)p_2(y) \left(\log\left(\frac{p_1(x)}{q_1(x)}\right) + \log\left(\frac{p_2(x)}{q_2(y)}\right)\right) \\
&= \sum_Y p_2(y) \sum_X p_1(x) \log\left(\frac{p_1(x)}{q_1(x)}\right) + \sum_X p_1(x) \sum_Y p_2(y) \log\left(\frac{p_2(x)}{q_2(y)}\right) \\
&= KL(P_1, Q_1) + KL(P_2, Q_2)
\end{aligned}$$

□

This proposition makes it much easier to compute the KL divergence between the distributions of samples of size n. Specifically, it implies that if P^n and Q^n are the product distributions for i.i.d. samples $(z_1, \ldots, z_n)$, we have:

$$KL(P^n, Q^n) = n \cdot KL(P, Q)$$

The KL divergence is also connected to the total variation distance via the *Pinsker's inequality*:

Proposition 17.5 (Pinsker's inequality)**.**

$$TV(P,Q) \leq \sqrt{\frac{1}{2} KL(P,Q)}$$

Proof. Let us consider $KL(P,Q)$ as a function P for a fixed Q:

$$F(p) = \sum_X p(x) \log\left(\frac{p(x)}{q(x)}\right)$$

Now, for any $t \in [0,1]$, consider the interpolating distribution $R_t = Q(1-t) + tP$ with mass function $r_t(x) = (1-t)q(x) + tp(x)$ and the corresponding function $f : [0,1] \to \mathbb{R}$:

$$f(t) = F(r_t) = \sum_x r_t(x) \log\left(\frac{r_t(x)}{q(x)}\right)$$

Notice that $f(0) = KL(Q,Q) = \sum_x q(x) \log(1) = 0$. The derivative is:

$$f'(t) = \sum_x \frac{dr_t(x)}{dt} (\log(r_t(x)) + 1 - \log(q(x))) = \sum_x (p(x) - q(x))(\log(r_t(x)) + 1 - \log(q(x)))$$

Notice that $f'(0) = \sum_x p(x) - q(x) = 0$. The second derivative is:

$$f''(t) = \sum_x \frac{(p(x) - q(x)^2}{r_t(x)} = \sum_x \frac{(p(x) - q(x)^2}{(1-t)q(x) + tp(x)}$$

Now, we have by Cauchy-Schwarz for any t:

$$\begin{aligned}
\sum_x |p(x) - q(x)| &= \sum_x \frac{|p(x) - q(x)|}{\sqrt{(1-t)q(x) + tp(x)}} \sqrt{(1-t)q(x) + tp(x)} \\
&\leq \sqrt{\sum_x \frac{(p(x) - q(x)^2}{(1-t)q(x) + tp(x)}} \sqrt{\sum_x (1-t)q(x) + tp(x)} \\
&= \sqrt{f''(t)}
\end{aligned}$$

Therefore, by the mean value theorem there is some $t \in [0, 1]$ such that:

$$\begin{aligned}
KL(P, Q) = f(1) &= f(0) + f'(0) + \frac{f''(t)}{2} = \frac{f''(t)}{2} \\
&\geq \frac{\left(\sum_x |p(x) - q(x)|\right)^2}{2} \\
&= 2(TV(P, Q))^2
\end{aligned}$$

□

Putting these facts together, we have the following Corollary:

Corollary 17.6. *Suppose P and Q are arbitrary distributions and P^n and Q^n are the associated product distributions. Then:*

$$TV(P^n, Q^n) \leq \sqrt{\frac{n}{2} KL(P, Q)}$$

That is, the total variation distance between datasets of size n sampled from distributions P and Q only grows as $O(\sqrt{n})$, rather than say $O(n)$. At first this might seem surprising as it seems to suggest that large samples from different distributions are less different looking than one might have expected. However, if you are familiar with the Central Limit Theorem, this may make a bit more sense. The CLT says that the average of a large sample from two distributions tends to look like a Gaussian, and so it may be more reasonable to think that the large samples are not as far apart in total variation distance as one might expect naively.

Now that we have these preliminary results out of the way, we can begin to prove our main lower bound:

Theorem 17.7. *Given R, H and G, there exists a function $\ell : \mathbb{R} \times \{1, -1\} \to \mathbb{R}$ such that $\ell(w, z)$ is G-Lipschitz, H-smooth and convex for all z and two distributions D^+ and D^- over $\{\pm 1\}$ such that if we define $\mathcal{L}^+(w) = \mathbb{E}_{z \sim D^+}[\ell(w, z)]$ and $\mathcal{L}^- = \mathbb{E}_{z \sim D^-}[\ell(w, z)]$ with $w_\star^+ = \operatorname{argmin} \mathcal{L}^+$ and $w_\star^- = \operatorname{argmin} \mathcal{L}^-$, then $\|w_\star^\pm\| \leq R$ and any function $F : (z_1, \ldots, z_T) \to \mathbb{R}$ satisfies either*

$$\mathop{\mathbb{E}}_{(z_i) \sim D^+} [\mathcal{L}^+(F(z_1, \ldots, z_T)) - \mathcal{L}^+(w_\star^+)] \geq \frac{G(R - G/H)^2}{2R\sqrt{6T}}$$

or

$$\mathop{\mathbb{E}}_{(z_i) \sim D^-} [\mathcal{L}^-(F(z_1, \ldots, z_T)) - \mathcal{L}^-(w_\star^+)] \geq \frac{G(R - G/H)^2}{2R\sqrt{6T}}$$

Further, if $H \geq 2G/R$, then $\frac{G(R-G/H)^2}{2R\sqrt{6T}} \geq \frac{GR}{8\sqrt{6T}}$.

Proof. Let us start by defining the function ℓ. To do so, we define:

$$f(x) = \begin{cases} 0 & \text{if } x \geq R \\ \frac{H}{2}(x-R)^2 & \text{if } x \in [R - G/H, R] \\ -G\,(x-R) - \frac{G^2}{2H} & \text{if } x \leq R - G/H \end{cases}$$

Observe that f is H-smooth, convex and G-Lipschitz. We then define $\ell(w, z) = f(wz)$ for $z \in \{1, -1\}$ and $w \in \mathbb{R}$.

Next, we specify the distributions D^+ and D^-. Let D^+ be such that $P[z = 1] = \frac{1+L}{2}$ and $P[z = -1] = \frac{1-C}{2}$ for some constant $C \in [0, 1/2]$ that we will specify later in the proof. Similarly, let D^- set $P[z = 1] = \frac{1-C}{2}$ and $P[z = -2] = \frac{1+C}{2}$ for the same constant L.

Now, for ease of notation let's define $Z^+ = (z_1^+, \ldots, z_T^+) \sim (D^+)^T$ and similarly for Z^-. We also define $\hat{w}^+ = F(Z^+)$ and $\hat{w}^- = F(Z^-)$

Let us define $\mathcal{L}^+(w) = \mathbb{E}_{z \sim D^+}[\ell(w, z)]$ and $\mathcal{L}^-(w) = \mathbb{E}_{z \sim D^-}[\ell(w, z)]$. Then we have for any $w > R$:

$$(\mathcal{L}^+)'(w) = G\frac{1-C}{2}$$

and for any $w < R - \frac{G}{H}$:

$$(\mathcal{L}^+)'(w) = -G\frac{1+C}{2} + \frac{1-C}{2}f'(-w) < -GC < 0$$

where we have used G-Lipschitzness of f. Therefore, $w_\star^+ \in [R - G/H, R]$. Similarly, we have $w_\star^- \; in[-R, -R + G/H]$. Thus, $|w_\star^\pm| \leq R$. Further, for any $w \in [-R, R]$, since $(\mathcal{L}^+)'(w) < -GC$ for $w < R - G/H$ and $(\mathcal{L}^-)'(w) > GC$ for $w > -R + G/H$, we have:

$$\mathcal{L}^+(w) - \mathcal{L}^+(w_\star^+) \geq GC(R - w - G/H)$$
$$\mathcal{L}^-(w) - \mathcal{L}^-(w_\star^-) \geq GC(w + R - G/H)$$

This establishes the important properties of $\mathcal{L}^\pm$ characterizing how they are sufficiently different from each other: their minimizers are roughly $2R$ apart, and choosing an incorrect point makes either loss grow linearly.

Now, let us characterize the difference between the distributions D^+ and D^-. First we calculute the KL divergence:

$$KL(D^+, D^-) = \frac{1+C}{2}\log\left(\frac{1+C}{1-L}\right) + \frac{1-C}{2}\log\left(\frac{1-C}{1+C}\right) = C\log(1+C) - C\log(1-C)$$

Now, since $0 \leq C \leq 1/2$, we have

$$\frac{1}{1-C} \leq 1 + 2C$$

(to verify this, notice that it is equivalent to $1 \leq 1 + C - 2C^2$). Therefore:

$$\begin{aligned} KL(D^+, D^-) &\leq C\log(1+C) + C\log(1+2C) \\ &\leq 3C^2 \end{aligned}$$

where in the last line we have used $\log(1 + x) \leq x$ for all $x \geq 0$.

Therefore, if we define $(D^\pm)^T$ to be the product distributions over the i.i.d. sample $(z_1, \ldots, z_T)$, we have:

$$\frac{1}{2}\sum_Z |P[Z^+ = Z] - P[Z^- = Z]| = TV((D^+)^T, (D^-)^T) \leq \sqrt{\frac{T}{2}KL(D^+, D^-)} \leq C\sqrt{\frac{3}{2}T}$$

Now, we make a simplification: define

$$\bar{F}(z_1, \ldots, z_T) = \Pi_{[-R,R]}F(z_1, \ldots, z_T)$$

notice that since $|w_\star^\pm| \leq R$, $\bar{F}$ will satisfy

$$\mathop{\mathbb{E}}_{Z^+}[\mathcal{L}^+(\bar{F}(Z^+)) - \mathcal{L}^+(w^+\star)] \leq \mathop{\mathbb{E}}_{Z^+}[\mathcal{L}^+(F(Z^+)) - \mathcal{L}^+(w_\star^+)]$$

and similarly for Z^- and $\mathcal{L}^-$. Thus without loss of generality we may prove the theorem for $\bar{F}$ instead of F. This allows us to assume that $\bar{F}$ has a *bounded* output value. Thus, for the remainder of the proof, we assume $|F(Z)| \leq R$ for all Z.

Therefore

$$\begin{aligned}
\mathbb{E}[\hat{w}^+] - \mathbb{E}[\hat{w}^-] &= \sum_Z P[Z^+ = Z]F(Z) - P[Z^- = Z]F(Z) \\
&\leq R\sum_Z |P[Z^+ = Z] - P[Z^- = Z]| \\
&\leq 2R \cdot TV((D^+)^T, (D^-)^T) \\
&\leq RC\sqrt{6T}
\end{aligned}$$

Now, we have:

$$\begin{aligned}
\mathbb{E}[\mathcal{L}^+(\hat{w}^+) - \mathcal{L}^+(w_\star^+)] + \mathbb{E}[\mathcal{L}^-(\hat{w}^-) - \mathcal{L}^-(w_\star^-)] &\geq GC\left(\mathbb{E}[R - \hat{w}^+ - G/H] + \mathbb{E}[\hat{w}^- + R - G/H]\right) \\
&= GC(2R - 2G/H - (\mathbb{E}[\hat{w}^+] - \mathbb{E}[\hat{w}^-])) \\
&\geq G\left(2(R - G/H)C - RC^2\sqrt{6T}\right)
\end{aligned}$$

Now, we can optimize for C: set $C = \frac{R - G/H}{R\sqrt{6T}}$ to obtain:

$$\mathbb{E}[\mathcal{L}^+(\hat{w}^+) - \mathcal{L}^+(w_\star^+)] + \mathbb{E}[\mathcal{L}^-(\hat{w}^-) - \mathcal{L}^-(w_\star^-)] \geq \frac{G(R - G/H)^2}{R\sqrt{6T}}$$

Now, since for any positive a and b, $2\max(a, b) \geq a + b$, we must have

$$\max(\mathbb{E}[\mathcal{L}^+(\hat{w}^+) - \mathcal{L}^+(w_\star^+)], + \mathbb{E}[\mathcal{L}^-(\hat{w}^-) - \mathcal{L}^-(w_\star^-)]) \geq \frac{G(R - G/H)^2}{2R\sqrt{6T}}$$

Now, if $H \geq 2G/R$, we have $R - G/H \geq R/2$, so that $\frac{G(R - G/H)^2}{2R\sqrt{6T}} \geq \frac{GR}{8\sqrt{6T}}$

□

18 Per-Coordinate Learning Rates

Let's consider a simple linear regression problem:

$$\hat{y} = \langle w, x \rangle$$

Suppose for example that we are trying to predict the identity of a speaker from two options given what they said. We'll say that positive $\hat{y}$ indicates that the speaker is person A and negative $\hat{y}$ indicates that the speaker is person B. Suppose that x is be a bag-of-words feature vector. This means that x is a vector with one coordinate for every possible english word, and the value of x at coordinate i is the number of times word i appears in whatever sentence or paragraph x is representing.

In this setting, the ith coordinate of w represents a kind of "score" for how much we believe the ith word is associated with either speaker. Large positive values indicate that the ith word is more likely to be said by speaker A, and large negative values indicate it is more likely to be said by speaker B, while near-zero values indicatres that this is in some sense a "neutral" word.

Now, we may try to learn w using the logistic loss:

$$\ell(w, (x, y)) = \log(1 + \exp(-y\langle w, x \rangle))$$

This loss is 2-smooth, and the gradient is:

$$\nabla \ell(w, (x, y)) = \frac{-yx \exp(-y\langle w, x \rangle)}{1 + \exp(-y\langle w, x \rangle)}$$

From this, if the feature vectors x satisfy $\|x\| \leq r$, we might expect the variance to be bounded by r^2, so we should look for a learning rate of the form

$$\eta_t \propto \frac{r}{\sqrt{t}}$$

However, we might be able to do better. Specifically, language has a long tail of rare words. That is, there are so many different words that any given paragraph is likely to have a few words that do not appear again for a very long time. As a result, in our data many of the coordinates of x will only rarely be non-zero. However, some of these coordinates may be very discriminative. For example, a biologist might use the word "polymerase" occasionally, while a physicist uses the word "quark" occasionally, and it is extremely rare for either person to use the other person's word.

Thus, when we see an example x that encodes a paragraph using the word "polymerase", we would ideally take a large step of gradient descent in order to quickly learn the fact that "polymerase" is a biologist-indicative word.

We will accomplish this via *per-coordinate learning rates*. That is, instead of the standard gradient descent update:

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t \nabla \ell(\mathbf{w}_t, z_t)$$

we will instead update each coordinate using it's own specific learning rate:

$$\mathbf{w}_{t+1}[i] = \mathbf{w}_t[i] - \eta_t[i] \nabla \ell(\mathbf{w}_t, z_t)[i]$$

where we use $v[i]$ to indicate the ith coordinate of a vector v. Thus, η_t is no longer a scalar but in fact a *vector* of learning rates. Then, we will try to set η_t in such a way that $\eta_t[i]$ is larger for coordinates for which the ith word is rarer. This way we can quickly adapt to the uncommon words like polymerase by taking a large step when they appear.

However, there is still a problem: how can we know ahead of time which coordinates are going to be rare? The solution is to just count them. We could set:

$$\eta_t[i] \propto \frac{1}{\sqrt{n_t[i]}}$$

where $n_t[i]$ is the number of times the ith word has appeared after t iterations.

However, we will be even fancier. Taking inspiration from our adaptive methods, we set:

$$\mathbf{g}_t = \nabla \ell(\mathbf{w}_t, z_t)$$
$$\eta_t[i] = \frac{c}{\sqrt{\epsilon^2 + \sum_{\tau=1}^{t} \mathbf{g}_\tau[i]^2}}$$

Note that

$$\mathbf{g}_t[i] = \left[\frac{-y_t \exp(-y_t \langle \mathbf{w}_t, \mathbf{x}_t \rangle)}{1 + \exp(-y \langle \mathbf{w}_t, \mathbf{x}_t \rangle)}\right] \mathbf{x}_t[i]$$

so that roughly speaking, $\sum_{\tau=1}^{t} \mathbf{g}_\tau[i]^2$ is a count of the number of times $\mathbf{x}_t[i]$ is non-zero. However, it is a little better than that - this sum is really more like an estimate of the *per-coordinate variance*:

$$\sigma^2[i] = \mathbb{E}[(\nabla \ell(\mathbf{w}_t, z_t)[i] - \nabla \mathcal{L}(\mathbf{w}_t)[i])^2$$

In much the same way that our previous adaptive learning rate $\eta_t \propto \frac{1}{\sqrt{\sum_{\tau=1}^{t} \|\mathbf{g}_t\|^2}}$ is an estimate of $\frac{1}{\sigma\sqrt{t}}$, this new learning rate is approximately:

$$\eta_t[i] \approx \frac{c}{\sigma[i]\sqrt{t}}$$

That is, the vector of learning rates is measuring how much we should "trust" each individual coordinate of the gradient. This is the famous AdaGrad algorithm J. Duchi, E. Hazan, and Singer 2010 (which was essentially simultaneously described by McMahan and Streeter 2010). As this algorithm was first proposed in the convex setting, we will analyze it there first.

Also, in order to make the algebra look a little cleaner, we will introduce the following notation:

Definition 18.1. *Given vectors $x_1, \ldots, x_t$, we use the notation $x_{a:b}$ to indicate the sum $\sum_{t=a}^{b} x_t$. Similarly, we use $x_{a:b}^2[i]$ to indicate the sum $\sum_{t=a}^{b} x_t^2[i]$ and $\|x\|_{a:b}^2$ for the sum $\sum_{t=a}^{b} \|x_t\|^2$.*

Using this notation, our learning rates can be re-written as:

$$\eta_t[i] = \frac{c}{\sqrt{\epsilon^2 + g_{1:t}^2[i]}}$$

To start with a good baseline, remember that while doing the previous homework, you proved the following result:

Theorem 18.2 (Adaptive Gradient Descent)**.** *Suppose that $\mathcal{L}$ is a convex function satisfying $\|\mathbf{w}_\star\|_2 \leq D_2$. Then with $c = \sqrt{2}D_2$, the algorithm:*

$$\mathbf{g}_t = \nabla \ell(\mathbf{w}_t, z_t)$$
$$\mathbf{w}_{t+1} = \Pi_{\|\mathbf{w}\|_2 \leq D_2} \mathbf{w}_t - \frac{c\mathbf{g}_t}{\sqrt{\|\mathbf{g}\|_{1:t}^2}}$$

satisfies:

$$\mathbb{E}\left[\sum_{t=1}^{T} \mathcal{L}(\mathbf{w}_t) - \mathcal{L}(\mathbf{w}_\star)\right] \leq 2\sqrt{2}D_2 \, \mathbb{E}\left[\sqrt{\sum_{t=1}^{T} \|\mathbf{g}_t\|^2}\right]$$

Note that in this theorem, we have emphasized the fact that we are measuring the norm of $\mathbf{w}_\star$ using the standard Euclidean 2-norm $\|\mathbf{w}_\star\|_2 = \sqrt{\sum_{i=1}^{d} \mathbf{w}_\star[i]^2}$.

In contrast, we will be able to prove the following result about AdaGrad:

Theorem 18.3 (AdaGrad Convergence)**.** *Suppose that $\mathcal{L}$ is a convex function satisfying $\|\mathbf{w}_\star\|_\infty \leq D_\infty$. Then with $c = \sqrt{2}D_\infty$, the algorithm:*

$$\mathbf{g}_t = \nabla \ell(\mathbf{w}_t, z_t)$$
$$\mathbf{w}_{t+1}[i] = \Pi_{|\mathbf{w}[i]| \leq D_\infty} \mathbf{w}_t[i] - \frac{c\mathbf{g}_t[i]}{\sqrt{\mathbf{g}_{1:t}^2[i]}}$$

satisfies:

$$\mathbb{E}\left[\sum_{t=1}^{T}\mathcal{L}(\mathbf{w}_t)-\mathcal{L}(\mathbf{w}_\star)\right]\leq 2\sqrt{2}D_\infty\,\mathbb{E}\left[\sum_{i=1}^{d}\sqrt{\sum_{t=1}^{T}\mathbf{g}_t[i]^2}\right]$$

This algorithm is called AdaGrad *(or sometimes* diagonal AdaGrad*).*

Before proving this theorem, let's take a minute to discuss some of the implications.

Note the differences here: now we are measuring $\mathbf{w}_\star$ using the infinity norm rather than the 2-norm: $\|\mathbf{w}_\star\|_\infty = \max_i |\mathbf{w}_\star[i]|$. The infinity norm is always smaller than the 2-norm, so at first this seems like a purely good thing. However, we may pay for this improvement by having

$$\sum_{i=1}^{d}\sqrt{\sum_{t=1}^{T}\mathbf{g}_t[i]^2} \tag{8}$$

in the bound rather than

$$\sqrt{\sum_{t=1}^{T}\|\mathbf{g}_t\|^2} \tag{9}$$

It is possible for (8) to be bigger than (9). But how much bigger? Let use Cauchy-Schwarz:

$$\begin{aligned}\sum_{i=1}^{d}\sqrt{\sum_{t=1}^{T}\mathbf{g}_t[i]^2} &\leq \sqrt{d}\sqrt{\sum_{i=1}^{d}\sum_{t=1}^{T}\mathbf{g}_t[i]^2}\\ &= \sqrt{d}\sqrt{\sum_{t=1}^{T}\sum_{i=1}^{d}\mathbf{g}_t[i]^2}\\ &= \sqrt{d\sum_{t=1}^{T}\|\mathbf{g}_t\|^2}\end{aligned}$$

So that (8) is at most $\sqrt{d}$ times bigger than (9).

On the other hand, consider the case that $\mathbf{w}_\star = (\pm w, \pm w, \ldots, \pm w)$ for some constant w. In this case we have:

$$\|\mathbf{w}_\star\|_2 = w\sqrt{d} = \|\mathbf{w}_\star\|_\infty\sqrt{d}$$

Thus, when $\mathbf{w}_\star = (\pm w, \pm w, \ldots, \pm w)$ we would have (assuming the c constants are set to their best values):

$$D_\infty\sum_{i=1}^{d}\sqrt{\sum_{t=1}^{T}\mathbf{g}_t[i]^2}\leq D_2\sqrt{\sum_{t=1}^{T}\|\mathbf{g}_t\|^2}$$

Of course, this is a strong restriction on $\mathbf{w}_\star$, but it does still allow for 2^d possible different values so some non-trivial optimization must happen to learn $\mathbf{w}_\star$.

Finally, let us consider a setting in which some of the coordinate $\mathbf{g}_t[i]$ are very rare, similar to our original motivating example. This setting is adapted from J. Duchi, E. Hazan, and Singer 2010. Suppose that $\mathbf{g}_t$ is always a 1-hot vector (i.e. has one non-zero coordinate) such that the ith coordinate $\mathbf{g}_t[i]$ is ± 1 with probability proportional to $1/i^2$ and 0 otherwise. Then we have:

$$\begin{aligned}\mathbb{E}[D_\infty\sum_{i=1}^{d}\sqrt{\mathbf{g}_{1:t}^2[i]}] &\leq D_\infty\sum_{i=1}^{d}\sqrt{T/i^2}\\ &= D_\infty\sqrt{T}\sum_{i=1}^{d}\frac{1}{i} = O(D_\infty\log(d)\sqrt{T})\end{aligned}$$

In contrast:

$$\mathbb{E}[D_2\sqrt{\|\mathbf{g}\|_{1:T}^2}] = D_2\sqrt{T}$$

Thus, so long as D_∞ is significantly less than D_2, the bound will be significantly better using AdaGrad.

In general, it is not obvious apriori which bound is better: the algorithm you use should depend on some empirical observations/specific knowledge about the data.

Now, let's try proving Theorem 18.3.

Proof of Theorem 18.3. Remember that our previous proofs for gradient descent on convex functions made a lot of use of the inequality:

$$\mathbb{E}[\mathcal{L}(\mathbf{w}_t) - \mathcal{L}(\mathbf{w}_\star)] \le \mathbb{E}[\langle \nabla\mathcal{L}(\mathbf{w}_t), \mathbf{w}_t - \mathbf{w}_\star\rangle] = \mathbb{E}[\langle \mathbf{g}_t, \mathbf{w}_t - \mathbf{w}_\star\rangle]$$

In fact, as we say in our analysis of acceleration (Notes 8, Theorem 5), this was the only part of the analysis that actually used the fact that $\mathbf{g}_t$ was at all related to $\mathcal{L}$: the rest of the analysis was purely algebraic manipulation and held for any $\mathbf{g}_t$ and and $\mathbf{w}_\star$. The key idea behind the analysis of AdaGrad is to break the analysis of $\langle \mathbf{g}_t, \mathbf{w}_t - \mathbf{w}_\star\rangle$ using the following simple but surprisingly useful observation:

$$\langle \mathbf{g}_t, \mathbf{w}_t - \mathbf{w}_\star\rangle = \sum_{i=1}^d g_t[i](\mathbf{w}_t[i] - \mathbf{w}_\star[i])$$

This means that we can write:

$$\begin{aligned}
\mathbb{E}\left[\sum_{t=1}^T \mathcal{L}(\mathbf{w}_t) - \mathcal{L}(\mathbf{w}_\star)\right] &\le \sum_{t=1}^T \mathbb{E}[\langle \mathbf{g}_t, \mathbf{w}_t - \mathbf{w}_\star\rangle] \\
&= \sum_{t=1}^T \sum_{i=1}^d \mathbb{E}[\mathbf{g}_t[i](\mathbf{w}_t[i] - \mathbf{w}_\star[i])] \\
&= \sum_{i=1}^d \mathbb{E}\left[\sum_{t=1}^T \mathbf{g}_t[i](\mathbf{w}_t[i] - \mathbf{w}_\star[i])\right]
\end{aligned}$$

So now, it suffices to just bound each sum $\sum_{t=1}^T \mathbf{g}_t[i](\mathbf{w}_t[i] - \mathbf{w}_\star[i])$ *individually*. These are just d 1-d versions of the same adaptive analysis you did on the homework. We'll do it again here for completeness. First, using the fact that $|\mathbf{w}_\star[i]| \le D_\infty$:

$$\begin{aligned}
(\mathbf{w}_{t+1}[i] - \mathbf{w}_\star[i])^2 &\le (\mathbf{w}_t[i] - \eta_t[i]\mathbf{g}_t[i] - \mathbf{w}_\star[i])^2 \\
&= (\mathbf{w}_t[i] - \mathbf{w}_\star[i])^2 - 2\eta_t[i]\mathbf{g}_t[i](\mathbf{w}_t[i] - \mathbf{w}_\star[i]) + \eta_t[i]^2\mathbf{g}_t[i]^2 \\
\mathbf{g}_t[i](\mathbf{w}_t[i] - \mathbf{w}_\star[i]) &\le \frac{(\mathbf{w}_t[i] - \mathbf{w}_\star[i])^2 - (\mathbf{w}_{t+1}[i] - \mathbf{w}_\star[i])^2}{2\eta_t[i]} + \frac{\eta_t[i]\mathbf{g}_t[i]^2}{2}
\end{aligned}$$

summing over t and reordering:

$$\begin{aligned}
\sum_{t=1}^T \mathbf{g}_t[i](\mathbf{w}_t[i] - \mathbf{w}_\star[i]) &\le \frac{(\mathbf{w}_1[i] - \mathbf{w}_\star[i])^2}{2\eta_1[i]} - \frac{(\mathbf{w}_{T+1}[i] - \mathbf{w}_\star[i])^2}{2\eta_T[i]} \\
&\quad + \sum_{t=2}^T (\mathbf{w}_t[i] - \mathbf{w}_\star[i])^2\left(\frac{1}{2\eta_t[i]} - \frac{1}{2\eta_{t-1}[i]}\right) + \sum_{t=1}^T \frac{\eta_t[i]\mathbf{g}_t[i]^2}{2} \\
&\le \frac{D_\infty^2}{2\eta_1[i]} + \sum_{t=2}^T D_\infty^2\left(\frac{1}{2\eta_t[i]} - \frac{1}{2\eta_{t-1}[i]}\right) + \sum_{t=1}^T \frac{\eta_t[i]\mathbf{g}_t[i]^2}{2} \\
&\le \frac{D_\infty^2}{2\eta_T[i]} + \sum_{t=1}^T \frac{\eta_t[i]\mathbf{g}_t[i]^2}{2}
\end{aligned}$$

Now, we observe that from our sum-to-integration lemma (Notes 6 Lemma 2), we have

$$\sum_{t=1}^{T} \eta_t[i]\mathbf{g}_t[i]^2 = \sum_{t=1}^{T} \frac{\mathbf{g}_t[i]^2}{\sqrt{\sum_{\tau=1}^{t} \mathbf{g}_\tau[i]^2}} \leq \int_0^{\sum_{t=1}^{T} \mathbf{g}_t[i]^2} \frac{dx}{\sqrt{x}} = \sqrt{2\sum_{t=1}^{T} \mathbf{g}_t[i]^2}$$

Thus, plugging in the definition of c and η_t now shows:

$$\sum_{t=1}^{T} \mathbf{g}_t[i](\mathbf{w}_t[i] - \mathbf{w}_\star[i]) \leq 2\sqrt{2}D_\infty \sqrt{\sum_{t=1}^{T} \mathbf{g}_t[i]^2}$$

now using the fact that

$$\begin{aligned}\mathbb{E}\left[\sum_{t=1}^{T} \mathcal{L}(\mathbf{w}_t) - \mathcal{L}(\mathbf{w}_\star)\right] &\leq \sum_{i=1}^{d} \mathbb{E}\left[\sum_{t=1}^{T} \mathbf{g}_t[i](\mathbf{w}_t[i] - \mathbf{w}_\star[i])\right] \\ &\leq 2\sqrt{2}D_\infty \sum_{i=1}^{d} \sqrt{\sum_{t=1}^{T} \mathbf{g}_t[i]^2}\end{aligned}$$

as desired. □

19 Adam

At this point, we have seen 3 (maybe 4) distinct components of stochastic optimization algorithms based on gradient descent:

1. **Adaptive learning rates**: these are lsearning rate schedules that depend on the observed statistics of the gradients. These make the optimizer robust to various unknown parameters, like the variance and sometimes the smoothness constant.

1.5. Related to adaptive learning rates, we had the notion of *exponentially weighted moving averages* for collecting the statistics of the gradients. Intuitively, this modifies the adaptive learning rates to "track" the relevant statistics of the losses over time.

2. **Momentum:** we saw two forms of momentum, both of which can be motivated as integrating some physical equations of motion. The first form of momentum simply does an Euler integration, while the second evaluates the gradient at an kind of "auxiliary point" in order to improve the integration. We saw that for deterministic smooth convex losses, appropriate settings for momentum result in *accelerated* convergence rates.

3. **Per-coordinate updates:** we have just seen that by essentially running a independent parallel copy of an adaptive algorithm on each coordinate, we can adapt to varying scales of the different coordinates and sometimes achieve improved guarantees.

The *Adam* algorithm Kingma and Ba 2014 simple combines all of these ideas together into one big pot:

Note that in this algorithm we use $\mathbf{m}$ to refer to the momentum and $\mathbf{v}$ to refer to the denominator. Don't be confused by this! Previously, we used $\mathbf{v}$ (for velocity) to refer to the momentum. We are presenting Adam in this way now to be consistent with the literature for this algorithm. the v in $\mathbf{v}$ is supposed to stand for "variance".

In addition to incorporating the above ideas, Adam adds one more wrinkle: dividing by $1 - \beta^t$. This is meant to *de-bias* some of the estimates. To see why, let us expand out the expression for $\mathbf{v}_t[i]$. Since we focus on just one coordinate, we'll drop the $[i]$'s for simpler equations:

$$\begin{aligned}\mathbf{v}_t &= \beta_2 \mathbf{v}_{t-1} + (1-\beta_2) g_t^2 \\ &= \beta_2^2 \mathbf{v}_{t-2} + \beta_2(1-\beta_2) g_{t-1}^2 + (1-\beta_2) g_t^2\end{aligned}$$

Algorithm 14 Adam

Input: Initial Point $\mathbf{w}_1$, learning rates η_t, momentum parameters β_1, β_2 (defaults typically 0.9 and 0.999).
Set $\mathbf{v}_0 = 0$.
Set $A_0[i] = \epsilon^2$
for $t = 1 \dots T$ **do**
  Set $\mathbf{g}_t = \nabla \ell(\mathbf{w}_t, z_t)$
  Set $\mathbf{m}_t = \beta_1 \mathbf{m}_{t-1} - (1-\beta_1)\mathbf{g}_t$.
  Set $\hat{\mathbf{m}}_t = \frac{\mathbf{m}_t}{1-\beta_1^t}$.
  Set $\mathbf{v}_t[i] = \beta_2 \mathbf{v}_{t-1} + (1-\beta_2)\mathbf{g}_t[i]^2$.
  Set $\hat{\mathbf{v}}_t[i] = \frac{\mathbf{v}_t[i]}{1-\beta_2^t}$
  Set $\mathbf{w}_{t+1}[i] = \mathbf{w}_t[i] - \eta_t \frac{\hat{\mathbf{m}}_t[i]}{\sqrt{\hat{\mathbf{v}}_t[i]}}$.
end for

expanding for t steps:

$$= \beta_2^t \mathbf{v}_0 + \beta_2^{t-1}(1-\beta_2)g_1^2 + \dots + \beta_2(1-\beta_2)g_{t-1}^2 + (1-\beta_2)g_t^2 \qquad = \beta_2^t \epsilon^2 + (1-\beta_2)\sum_{\tau=0}^{t-1} \beta_2^\tau g_{t-\tau}^2$$

Now, let us assume $\beta_2^t \epsilon^2 \approx= 0$. This is reasonable: β_2^t already decreases exponentially fast, and ϵ^2 was already very small to begin with. Further, let us assume that each g_t satisfies $\mathbb{E}[g_t^2] = g^2$ for some *fixed* g^2. This is not true - is only somehow approximately true in the sense that the we have $\mathbb{E}[g_t^2] = \nabla \mathcal{L}(\mathbf{w}_t)^2 + \sigma_t^2$, and we might expect both $\nabla \mathcal{L}(\mathbf{w}_t)$ and σ_t to be a slowly varying function of t. However, under this assumption:

$$\begin{aligned}
\mathbb{E}[\mathbf{v}_t] &= (1-\beta_2)\sum_{\tau=0}^{t-1} \beta_2^\tau g^2 \\
&= (1-\beta_2)\frac{1-\beta_2^t}{1-\beta_2} g^2 \\
&= (1-\beta_2^t) g^2
\end{aligned}$$

Therefore, by dividing by $1-\beta_2^t$, we obtain an "unbiased" estimate of g. Of course, for any reasonably large t, $1-\beta_2^t$ is nearly 1, so it is unclear how and whether this is actually important in theory (or even in practice for that matter, as you will see on the homework).

In the case of the momentum parameter m_t, dividing by $1-\beta_1^t$ has a similar de-biasing effect. This also motivates a second way to think about momentum beyond the physical intuition we discussed earlier.

19.1 Momentum as Averaging

The momentum parameter in Adam can be interpreted as artificially increasing the batch size, To see this, let's follow the same "unraveling" of the recursive formula for m_t as we did for $\mathbf{v}_t$:

$$\begin{aligned}
m_t &= \beta_1^t \epsilon^2 + (1-\beta_1)\sum_{\tau=0}^{t-1} \beta_1^\tau g_{t-\tau} \\
&= (1-\beta_1)\sum_{\tau=0}^{t-1} \beta_1^\tau g_{t-\tau}
\end{aligned}$$

Now, writing $\frac{1-\beta_1^t}{1-\beta_1} = 1 + \beta_1 + \beta_1^2 + \dots \beta_t^{t-1}$, we have:

$$\hat{m}_t = \frac{\sum_{\tau=0}^{t-1} \beta_1^\tau g_{t-\tau}}{\sum_{\tau=0}^{t-1} \beta_1^\tau}$$

That is, just like $\mathbf{v}_t$, m_t is a weighted average of the gradient values g_t. Let us write:

$$g_t = \nabla \mathcal{L}(\mathbf{w}_t) + r_t$$

where r_t is some mean-zero random value. Further, let's assume that w_t is varying slowly enough that we can approximate $\mathcal{L}(\mathbf{w}_{t'}) = \mathcal{L}(\mathbf{w}_t)$ for $t \leq t'$. Then we have:

$$\begin{aligned}\hat{m}_t &= \nabla \mathcal{L}(\mathbf{w}_t) + \frac{\sum_{\tau=0}^{t-1} \beta_1^\tau r_{t-\tau}}{\sum_{\tau=0}^{t-1} \beta_1^\tau} \\ \mathbb{E}[\hat{m}_t] &= \nabla \mathcal{L}(\mathbf{w}_t)\end{aligned}$$

What about the variance of $\hat{m}_t$? Supposing each g_t has variance at most σ^2 we have $\mathbb{E}[r_t^2] \leq \sigma^2$ and so:

$$\begin{aligned}\text{Var}(\hat{m}_t) &= \frac{\sum_{\tau=0}^{t-1} \beta_1^{2\tau} \mathbb{E}[r_{t-\tau}^2]}{(\sum_{\tau=0}^{t-1} \beta_1^\tau)^2} \\ &= \sigma^2 \frac{(1-\beta_1)(1-\beta_1^{2t-1})}{(1-\beta_1^2)(1-\beta_1^t)^2}\end{aligned}$$

Now, before analyzing this, let's try to gain some quick back-of-the-envelope intuition for what will happen. For large t, we can hopefully ignore the β_1^t and β_t^{2t-1} terms, so that we are left with:

$$\sigma^2 \frac{(1-\beta_1)^2}{1-\beta_1^2} = \sigma^2 \frac{1-\beta_1}{1+\beta_1}$$

Now, by first-order taylor expansion we might expect that this is approximately:

$$\sigma^2(1-\beta_1)$$

So that if $\beta_1 = \frac{N-1}{N}$, we would have variance of about $\frac{\sigma^2}{N}$, which corresponds to what would happen we we had a batch-size of N. Note that this should seem intuitively reasonable from the initial formula:

$$\begin{aligned}m_t &= \beta_1 m_{t-1} + (1-\beta_1) g_t \\ &= \frac{N-1}{N} m_{t-1} + \frac{1}{N} g_t\end{aligned}$$

If m_{t-1} happened to be the average of $N-1$ elements, then this formula would set m_t to be the average of those $N-1$ elements and g_t, a total of N elements.

Let's try to see this without the taylor approximations. We need to control $\frac{(1-\beta_1)^2(1-\beta_1^{2t-1})}{(1-\beta_1^2)(1-\beta_1^t)^2}$. We'll start by looking at just one part, $\frac{(1-\beta_1)^2}{1-\beta_1^2}$. Let's write $\beta_1 = 1-\alpha$. Then we have:

$$\frac{(1-\beta_1)^2}{1-\beta_1^2} = \frac{\alpha^2}{2\alpha - \alpha^2} = \frac{\alpha}{2-\alpha} \leq \alpha$$

Further, so long as $t \geq \log(1/2)/\log(\beta_1)$, we will have $\beta_1^t \leq 1/2$ so that:

$$\frac{(1-\beta_1)^2(1-\beta_1^{2t-1})}{(1-\beta_1^2)(1-\beta_1^t)^2} \leq 4\alpha = 4(1-\beta)$$

which (up to the factor 4), is the same result we had with the taylor series.

On the surface, this seems really good - we decreased the variance by a large amount! Unfortunately, this decrease in variance is actually accompanied by a corresponding increase in *bias*. The issue is with our assumption that $\mathbf{w}_t$ is constant. In order to deal with the variation of $\mathbf{w}_t$ properly, it turns out that m_t becomes a *biased* estimate of the gradient. This extra bias will almost exactly cancel out the decreased variance so that in the end it is hard to show that we gain anything. Indeed, *no current analysis of Adam or similar algorithms shows any advantage over adagrad in smooth losses*. Nevertheless, in practice these algorithms perform much better, so clearly this is a interesting hole in our understanding of the properties of the loss functions and how the algorithms work.

19.2 Convergence failure of Adam

It turns out that the Adam algorithm actually can fail to converge. To get some idea of what could go wrong, notice that the definition of $\mathbf{v}_t$ is essentially an average of previous gradients. Therefore, if there is some constant variance so that previous gradients always have $|g_t| = 1$ for example, regardless of the true value of $\nabla\mathcal{L}(\mathbf{w}_t)$, then $\hat{\mathbf{v}}_t$ will always be 1. Thus, we can see that it must be important for η_t to decrease or be very small in order to guarantee convergence (unlikely with adagrad). However, even when η_t decreases like $1/\sqrt{t}$, there can still be a problem, as observed by Reddi, Kale, and Kumar 2018, in which an example is provided for which Adam makes consistent progress *in the wrong direction*.

To see why this might be, consider an extreme case in which $\beta_2 = \beta_1 = 0$. This corresponds to the update:

$$\mathbf{w}_{t+1}[i] = \mathbf{w}_t[i] - \eta_t \frac{\mathbf{g}_t[i]}{|\mathbf{g}_t[i]|}$$

That is, we update update using the *sign* of the gradient rather than the actual gradient. In a previous homework problem, you already provided a distribution for $\mathbf{g}_t$ such that the sign of $\mathbf{g}_t$, on average, is different than the sign of $\mathbb{E}[\mathbf{g}_t]$, so that the above update will actually make negative progress in expectation.

On the other hand, it might be that with more realistic values for β_2 and β_1 (which are usually very close to 1), it is possible to show some convergence guarantee. So far as I know, this is still open.

20 Momentum in Stochastic Optimization

When discussing Adam, we briefly motivated the idea of momentum as related to artificially increasing the batch size rather than as a method connected to physical simulation. Now, let's try to make this connection more formal. We will consider a simple SGD with momentum algorithm:

Algorithm 15 SGD with Momentum

Input: Initial Point $\mathbf{w}_1$, learning rate η, momentum parameters α, time horizon T:
Sample z_1.
Set $\mathbf{m}_1 = \nabla \ell(\mathbf{w}_1, z_1)$.
Set $\mathbf{w}_2 = \mathbf{w}_1 - \mathbf{m}_1$
for $t = 2 \dots T$ **do**
  Sample z_t.
  Set $\mathbf{m}_t = (1-\alpha)\mathbf{m}_{t-1} + \alpha \nabla \ell(\mathbf{w}_t, z_t)$.
  Set $\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \mathbf{m}_t$.
end for

The difficulty in analyzing SGD with momentum is that it is not the case that

$$\mathbb{E}[\mathbf{m}_t] = \nabla \mathcal{L}(\mathbf{w}_t)$$

Thus, our standard strategy of writing:

$$\mathbb{E}[\mathcal{L}(\mathbf{w}_{t+1})] \leq \mathbb{E}[\mathcal{L}(\mathbf{w}_t)] - \eta \, \mathbb{E}[\langle \nabla \mathcal{L}(\mathbf{w}_t), \mathbf{m}_t \rangle] + \frac{\eta^2 H}{2} \, \mathbb{E}[\|\mathbf{m}_t\|^2]$$

$$\begin{aligned} &= \mathbb{E}[\mathcal{L}(\mathbf{w}_t)] - \eta \, \mathbb{E}[\|\nabla \mathcal{L}(\mathbf{w}_t)\|^2] + \frac{\eta^2 H}{2} \, \mathbb{E}[\|\mathbf{m}_t\|^2] \\ &\leq \mathbb{E}[\mathcal{L}(\mathbf{w}_t)] - \eta(1 - H\eta/2) \, \mathbb{E}[\|\nabla \mathcal{L}(\mathbf{w}_t)\|^2] + \frac{\eta^2 H \sigma^2}{2} \end{aligned}$$

is not valid because $\mathbb{E}[\langle \nabla \mathcal{L}(\mathbf{w}_t), \mathbf{m}_t \rangle] \neq \mathbb{E}[\|\nabla \mathcal{L}(\mathbf{w}_t)\|^2]$. Instead, we will need to do something a little more subtle. In order to facilitate this analysis, we'll define the quantity:

$$\epsilon_t = \mathbf{m}_t - \nabla \mathcal{L}(\mathbf{w}_t)$$

Then, ϵ_t is measuring the amount of "error" in the estimate $\mathbf{g}_t$. Notice that $\mathbb{E}[\epsilon_t] \neq 0$.

Also, let's define:

$$r_t = \nabla \ell(\mathbf{w}_t, z_t) - \nabla \mathcal{L}(\mathbf{w}_t)$$

Thus, r_t is the error in the ordinary non-momentum gradient estimates, and has $\mathbb{E}[r_t] = 0$. We'll also assume as usual that $\mathbb{E}[\|r_t^2\|] \leq \sigma^2$.

With this notation, we can produce an analog of our one-step progress lemma for SGD, now incorporating momentum:

Lemma 20.1. *Suppose that $\mathcal{L}$ is H-smooth. Then so long as $\eta_t \leq \frac{1}{4H}$,*

$$\mathbb{E}[\mathcal{L}(\mathbf{w}_{t+1}) - L(\mathbf{w}_t)] \leq -\frac{\eta}{4} \, \mathbb{E}[\|\nabla L(\mathbf{w}_t)\|^2] + \frac{3\eta}{4} \, \mathbb{E}[\|\epsilon_t\|^2]$$

Proof. From the standard smoothness lemma:

$$\begin{aligned} H(\mathbf{w}_{t+1}) &\leq \mathcal{L}(\mathbf{w}_t) + \langle \nabla \mathcal{L}(\mathbf{w}_t), \mathbf{w}_{t+1} - \mathbf{w}_t \rangle + \frac{H}{2} \|\mathbf{w}_{t+1} - \mathbf{w}_t\|^2 \\ &= \mathcal{L}(\mathbf{w}_t) - \eta \langle \nabla \mathcal{L}(\mathbf{w}_t), \mathbf{m}_t \rangle + \frac{\eta^2 H \|\mathbf{m}_t\|^2}{2} \end{aligned}$$

taking expectation of both sides:

$$\begin{aligned}\mathbb{E}[\mathcal{L}(\mathbf{w}_{t+1})] &\leq \mathbb{E}[\mathcal{L}(\mathbf{w}_t)] - \eta\,\mathbb{E}[\langle\nabla\mathcal{L}(\mathbf{w}_t), \mathbf{m}_t\rangle] + \frac{\eta^2 H\,\mathbb{E}[\|\mathbf{m}_t\|^2]}{2} \\ &\leq \mathbb{E}[\mathcal{L}(\mathbf{w}_t)] - \eta\,\mathbb{E}[\|\nabla\mathcal{L}(\mathbf{w}_t)\|^2] - \eta\,\mathbb{E}[\langle\nabla\mathcal{L}(\mathbf{w}_t), \epsilon_t\rangle] + \frac{\eta^2 H\,\mathbb{E}[\|\mathbf{m}_t\|^2]}{2}\end{aligned}$$

Using Young inequality $\langle x, y\rangle \leq \frac{\|x\|^2}{2\lambda} + \frac{\lambda\|y\|^2}{2}$ with $\lambda = 1$:

$$\begin{aligned}&\leq \mathbb{E}[\mathcal{L}(\mathbf{w}_t)] - \eta\,\mathbb{E}[\|\nabla\mathcal{L}(\mathbf{w}_t)\|^2] + \frac{\eta}{2}\,\mathbb{E}[\|\nabla\mathcal{L}(\mathbf{w}_t)\|^2] + \frac{\eta}{2}\,\mathbb{E}[\|\epsilon_t\|^2] + \frac{\eta^2 H\,\mathbb{E}[\|\mathbf{m}_t\|^2]}{2} \\ &\leq \mathbb{E}[\mathcal{L}(\mathbf{w}_t)] - \frac{\eta_t}{2}\,\mathbb{E}[\|\nabla\mathcal{L}(\mathbf{w}_t)\|^2] + \frac{\eta}{2}\,\mathbb{E}[\|\epsilon_t\|^2] + \frac{\eta^2 H\,\mathbb{E}[\|\nabla\mathcal{L}(\mathbf{w}_t) + \epsilon_t\|^2]}{2}\end{aligned}$$

Using $\|a + b\|^2 \leq 2\|a\|^2 + 2\|b\|^2$:

$$\leq \mathbb{E}[\mathcal{L}(\mathbf{w}_t)] - \frac{\eta_t}{2}\,\mathbb{E}[\|\nabla\mathcal{L}(\mathbf{w}_t)\|^2] + \frac{\eta_t}{2}\,\mathbb{E}[\|\epsilon_t\|^2] + \frac{\eta_t^2 H\,\mathbb{E}[2\|\nabla\mathcal{L}(\mathbf{w}_t)\| + 2\|\epsilon_t\|^2]}{2}$$

Using $\eta_t \leq \frac{1}{4H}$:

$$\begin{aligned}&\leq \mathbb{E}[\mathcal{L}(\mathbf{w}_t)] - \frac{\eta_t}{2}\,\mathbb{E}[\|\nabla\mathcal{L}(\mathbf{w}_t)\|^2] + \frac{\eta_t}{2}\,\mathbb{E}[\|\epsilon_t\|^2] + \frac{\eta_t\,\mathbb{E}[\|\nabla\mathcal{L}(\mathbf{w}_t)\| + \|\epsilon_t\|^2]}{4} \\ &= \mathbb{E}[\mathcal{L}(\mathbf{w}_t)] - \frac{\eta_t}{4}\,\mathbb{E}[\|\nabla\mathcal{L}(\mathbf{w}_t)\|^2] + \frac{3\eta_t}{4}\,\mathbb{E}[\|\epsilon_t\|^2]\end{aligned}$$

□

Let's compare this result to our standard gradient descent result when $\mathbb{E}[\mathbf{m}_t] = \nabla\mathcal{L}(\mathbf{w}_t)$. This required only $\eta \leq 1/H$ and achieved:

$$\mathbb{E}[\mathcal{L}(\mathbf{w}_{t+1}) - \mathcal{L}(\mathbf{w}_t)] \leq -\frac{\eta}{2}\,\mathbb{E}[\|\nabla\mathcal{L}(\mathbf{w}_t)\|^2] + \frac{\eta^2 H}{2}\,\mathbb{E}[\|\epsilon\|^2]$$

The dependency on $\mathbb{E}[\|\nabla\mathcal{L}(\mathbf{w}_t)\|^2]$ is similar - just the constants have changed. However, the dependency on $\|\epsilon\|^2$ has changed quite a bit: the power of η has decreased. As a result, *this lemma is not sufficient to guarantee convergence unless ϵ becomes small.*

How small does ϵ need to be? Notice that if $\|\epsilon_t\|^2 \geq \frac{1}{3}\|\nabla\mathcal{L}(\mathbf{w}_t)\|^2$, then Lemma 20.1 does not actually imply that $\mathcal{L}(\mathbf{w}_{t+1}) \leq \mathcal{L}(\mathbf{w}_t)$ anymore. Therefore, we should expect to make progress on the function value only when $\|\epsilon_t\|^2 \leq \frac{1}{3}\|\nabla\mathcal{L}(\mathbf{w}_t)\|^2$. Further, if our goal is to at least match the performance of regular non-momentum SGD, then we'd hope to make progress so long as $\|\nabla\mathcal{L}(\mathbf{w}_t)\|^2 > \Omega(1/\sqrt{T})$. This suggests that we need to get $\|\epsilon_t\|^2$ down to $O(1/\sqrt{T})$.

This is the crux of the momentum idea: if we view $\mathbf{m}_t$ as running average of gradient estimates, it is reasonable to believe that $\mathbf{m}_t$ would be a very close estimate of $\nabla\mathcal{L}(\mathbf{w}_t)$, and so the value of ϵ_t would indeed be small. Let's try to gain some intuition behind how this should happen.

Instead of considering the exponentially weighted moving average, let's just consider a simple *windowed* average:

$$\mathbf{m}_t = \frac{1}{K}\sum_{i=0}^{K-1}\nabla\ell(\mathbf{w}_{t-i}, z_{t-i})$$

What is ϵ_t in this setting?

$$\begin{aligned}\epsilon_t &= \frac{1}{K}\sum_{i=0}^{K-1}(\nabla\ell(\mathbf{w}_{t-i}, z_{t-i}) - \nabla\mathcal{L}(\mathbf{w}_t)) \\ &= \frac{1}{K}\sum_{i=0}^{K-1}(\nabla\ell(\mathbf{w}_{t-i}, z_{t-i}) - \nabla\mathcal{L}(\mathbf{w}_{t-i})) + \frac{1}{K}\sum_{i=0}^{K-1}(\nabla\mathcal{L}(\mathbf{w}_{t-i}) - \nabla\mathcal{L}(\mathbf{w}_t))\end{aligned}$$

Now, the first term above, $\frac{1}{K}\sum_{i=0}^{K-1}(\nabla\ell(\mathbf{w}_{t-i}, z_{t-i}) - \nabla\mathcal{L}(\mathbf{w}_{t-i}))$, is an average of K mean-zero values, and so will be on average $O(1/\sqrt{K})$. The next term, on the other hand is a kind of "bias" that is harder to control. To gain some rough idea of how we could make it small, notice that by smoothness, $\|\nabla\mathcal{L}(\mathbf{w}_{t-i}) - \nabla\mathcal{L}(\mathbf{w}_t)\| \leq H\|\mathbf{w}_{t-i} - \mathbf{w}_t\|$. Further, if we expect $\|\mathbf{w}_{t-1} - \mathbf{w}_t\| = O(\eta)$ since the learning rate is η, we can argue:

$$\begin{aligned}\left\|\frac{1}{K}\sum_{i=0}^{K-1}(\nabla\mathcal{L}(\mathbf{w}_{t-i}) - \nabla\mathcal{L}(\mathbf{w}_t))\right\| &\leq O\left(\frac{1}{K}\sum_{i=0}^{K-1}\eta H\right)\\ &= O\left(K\eta H\right)\end{aligned}$$

So overall, we have:

$$\|\epsilon_t\| = O(\frac{1}{\sqrt{K}} + \eta K)$$

Setting K optimally yields:

$$\|\epsilon_t\| = O(\eta^{1/3})$$

Now, let's try to apply Lemma 20.1:

$$\begin{aligned}\mathbb{E}[\mathcal{L}(\mathbf{w}_{t+1}) - \mathcal{L}(\mathbf{w}_t)] &\leq O(-\eta\,\mathbb{E}[\|\nabla\mathcal{L}(\mathbf{w}_t)\|^2] + \eta^{5/3})\\ \sum_{t=1}^{T}\mathbb{E}[\|\nabla\mathcal{L}(\mathbf{w}_t)\|^2] &\leq O\left(\frac{\Delta}{\eta} + T\eta^{2/3}\right)\end{aligned}$$

So, optimizing η yields:

$$\sum_{t=1}^{T}\mathbb{E}[\|\nabla\mathcal{L}(\mathbf{w}_t)\|^2] \leq O\left(T^{3/5}\right)$$

In contrast, with regular SGD we obtained $\sum_{t=1}^{T}\mathbb{E}[\|\nabla\mathcal{L}(\mathbf{w}_t)\|^2] \leq O\left(T^{1/2}\right)$, so it seems like things have actually gotten worse!

The issue here was that our bound $\|\epsilon_t\| = O(\eta^{1/3})$ was actually a bit loose. If we are more careful, we will be able to obtain $\|\epsilon_t\| = O(\eta^{1/2})$ instead. The key observation is that the bound $\|\mathbf{w}_{t-1} - \mathbf{w}_t\| = O(\eta)$ was also a bit loose: really, we have $\|\mathbf{w}_{t-1} - \mathbf{w}_t\| = O(\eta\mathbf{m}_{t-1}) = O(\eta\|\epsilon_{t-1}\| + \eta\|\nabla\mathcal{L}(\mathbf{w}_{t-1})\|)$. Thus, if ϵ_t and $\nabla\mathcal{L}(\mathbf{w}_t)$ are both getting smaller, we should expect $\|\mathbf{w}_{t-1} - \mathbf{w}_t\|$ to also be getting smaller, which in turn makes ϵ even smaller. This positive feedback cycle is what will eventually yield our improved bounds.

Next, we have the following fact about the error in the momentum estimates. This is the Lemma that will inform our choices for α and η, and will need to be modified for alternative momentum schemes.

Lemma 20.2. *For any $\alpha \leq 1/2$, setting $\eta = \frac{\alpha}{\sqrt{48}H}$ guarantees:*

$$\frac{\sqrt{3}}{8H}\,\mathbb{E}[\|\epsilon_{t+1}\|^2 - \|\epsilon_t\|^2] \leq -\frac{3\eta}{4}\,\mathbb{E}[\|\epsilon_t\|^2] + \frac{\eta}{8}\,\mathbb{E}[\|\nabla\mathcal{L}(\mathbf{w}_t)\|^2] + 3\sqrt{3}H\eta^2\sigma^2$$

Proof. Recall our notation:

$$r_t = \nabla\ell(\mathbf{w}_t, z_t) - \nabla\mathcal{L}(\mathbf{w}_t)$$

Note that we have the important property:

$$\mathbb{E}[r_t] = 0$$

Note however that $\mathbb{E}[\epsilon_t] \neq 0$.

Now, we derive a recursive formula for ϵ_{t+1} in terms of ϵ_t:

$$\begin{aligned}
\epsilon_{t+1} &= \mathbf{m}_{t+1} - \nabla\mathcal{L}(\mathbf{w}_{t+1}) \\
&= (1-\alpha)\mathbf{m}_t + \alpha_t \nabla\ell(\mathbf{w}_{t+1}, z_{t+1}) - \nabla\mathcal{L}(\mathbf{w}_{t+1}) \\
&= (1-\alpha)(\mathbf{m}_t - \nabla\mathcal{L}(\mathbf{w}_{t+1})) + \alpha(\nabla\ell(\mathbf{w}_{t+1}, z_{t+1}) - \nabla\mathcal{L}(\mathbf{w}_{t+1})) \\
&= (1-\alpha)(\mathbf{m}_t - \nabla\mathcal{L}(\mathbf{w}_t)) + (1-\alpha)(\nabla\mathcal{L}(\mathbf{w}_t) - \nabla\mathcal{L}(\mathbf{w}_{t+1})) + \alpha r_{t+1} \\
&= (1-\alpha)\epsilon_t + (1-\alpha)(\nabla\mathcal{L}(\mathbf{w}_t) - \nabla\mathcal{L}(\mathbf{w}_{t+1})) + \alpha r_{t+1}
\end{aligned}$$

Now, remember that we are actually interested in $\mathbb{E}[\|\epsilon_t\|^2]$, so let us take the norm-squared of both sides in the above, and use the fact that $\mathbb{E}[r_t] = 0$:

$$\begin{aligned}
\mathbb{E}[\|\epsilon_{t+1}\|^2] &= (1-\alpha)^2\,\mathbb{E}[\|\epsilon_t\|^2] + 2(1-\alpha)^2\,\mathbb{E}[\langle\epsilon_t, \nabla\mathcal{L}(\mathbf{w}_t) - \nabla\mathcal{L}(\mathbf{w}_{t+1})\rangle] + (1-\alpha)^2\,\mathbb{E}[\|\nabla\mathcal{L}(\mathbf{w}_t) - \nabla\mathcal{L}(\mathbf{w}_{t+1})\|^2] \\
&\quad + \alpha^2\,\mathbb{E}[\|r_{t+1}\|^2] \\
&\leq (1-\alpha)^2\,\mathbb{E}[\|\epsilon_t\|^2] + 2(1-\alpha)^2\,\mathbb{E}[\langle\epsilon_t, \nabla\mathcal{L}(\mathbf{w}_t) - \nabla\mathcal{L}(\mathbf{w}_{t+1})\rangle] + (1-\alpha)^2\,\mathbb{E}[\|\nabla\mathcal{L}(\mathbf{w}_t) - \nabla\mathcal{L}(\mathbf{w}_{t+1})\|^2] \\
&\quad + \alpha^2\sigma^2
\end{aligned}$$

Notice that by H-smoothness,

$$\begin{aligned}
\|\nabla\mathcal{L}(\mathbf{w}_t) - \nabla\mathcal{L}(\mathbf{w}_{t+1})\| &\leq H\|\mathbf{w}_t - \mathbf{w}_{t+1}\| \\
&\leq H\eta\|\mathbf{m}_t\| \\
&\leq H\eta(\|\nabla\mathcal{L}(\mathbf{w}_t)\| + \|\epsilon_t\|)
\end{aligned}$$

Further, by Young inequality again, for any λ we have:

$$\langle\epsilon_t, \nabla\mathcal{L}(\mathbf{w}_t) - \nabla\mathcal{L}(\mathbf{w}_{t+1})\rangle \leq \frac{\|\epsilon_t\|^2}{2\lambda} + \lambda H^2\eta^2(\|\nabla\mathcal{L}(\mathbf{w}_t)\|^2 + \|\epsilon_t\|^2)$$

Thus, putting this together with $(1-\alpha)^2 \leq 1$, we have:

$$\mathbb{E}[\|\epsilon_{t+1}\|^2] \leq \left[(1-\alpha)^2 + \frac{1}{\lambda} + 2\lambda H^2\eta^2 + 2H^2\eta^2\right]\mathbb{E}[\|\epsilon_t\|^2] + \left[2\lambda H^2\eta^2 + H^2\eta^2\right]\mathbb{E}[\|\nabla\mathcal{L}(\mathbf{w}_t)\|^2] + \alpha^2\sigma^2$$

Let's take a step back and see where we are going with this. Notice that $(1-\alpha)^2 = 1 - 2\alpha + \alpha^2$. Therefore, by setting $\lambda \approx 1/\alpha$, and then setting η in some clever way we might be able to guarantee something like $(1-\alpha)^2 + \frac{1}{\lambda} + 2\lambda L^2\eta^2 + 2L^2\eta^2 \approx 1 - \alpha$. Discarding many constants, and observing that $\lambda\eta^2 \approx \eta^2/\alpha$ is bigger η^2, this would yield something like:

$$\mathbb{E}[\|\epsilon_{t+1}\|^2] \leq (1-\alpha)\,\mathbb{E}[\|\epsilon_t\|^2] + O(\eta^2/\alpha\,\mathbb{E}[\|\nabla\mathcal{L}(\mathbf{w}_t)\|^2] + \alpha^2\sigma^2)$$

This suggests that $\|\epsilon_t\|^2$ will decrease until it reaches some "equilibrium" value when the above recurrence has $\|\epsilon_{t+1}\| = \|\epsilon_t\|$. Solving for this, we obtain:

$$\mathbb{E}[\eta\|\epsilon_t\|^2] \leq O\left(\frac{\eta^3}{\alpha^2}\,\mathbb{E}[\|\nabla\mathcal{L}(\mathbf{w}_t)\|^2] + \alpha\eta\sigma^2\right)$$

So in order to keep the $\eta\|\epsilon_t\|^2$ terms in Lemma 20.1 from dominating the $-\eta\|\nabla\mathcal{L}(\mathbf{w}_t)\|^2$ terms, we should try to set $\eta = \alpha$

Let's see if this is possible. We will try to obtain $(1-\alpha)^2 + \frac{1}{\lambda} + 2\lambda H^2\eta^2 + 2H^2\eta^2 \leq 1 - \frac{1}{2}\alpha$. We can expand the $(1-\alpha)^2$ to $1 - 2\alpha + \alpha^2$. Thus, we have an "extra" $-\frac{3}{2}\alpha$ term of slack, and three positive terms α^2, $\frac{1}{\lambda}$ and $2H^2(\lambda+1)\eta^2$. We will try to get all three positive terms at most $\frac{1}{2}\alpha$ to obtain the entire expression is at most $1 - \frac{1}{2}\alpha$. To accomplish this, we need:

$$\begin{aligned}
\alpha &\leq \frac{1}{2} \\
\lambda &= \frac{2}{\alpha} \\
\eta &\leq \frac{\alpha}{4H}
\end{aligned}$$

The first two equations deal with α^2 and $\frac{1}{\lambda}$. For the final term, notice that $\lambda > 1$, so that $2H^2(\lambda+1)\eta^2 \leq 4H^2\lambda\eta^2 = \frac{8H^2\eta^2}{\alpha}$. So if we set $\eta \leq \frac{\alpha}{4H}$, we will have $(1-\alpha)^2 + \frac{1}{\lambda} + 2\lambda H^2\eta^2 + 2H^2\eta^2 = 1 - \frac{1}{2}\alpha \leq 1 - \frac{1}{2}\alpha$.

Thus overall we have obtained:

$$\begin{aligned}
\mathbb{E}[\|\epsilon_{t+1}\|^2] &\leq (1-\frac{\alpha}{2})\,\mathbb{E}[\|\epsilon_t\|^2] + \left[2\lambda H^2\eta^2 + H^2\eta^2\right]\mathbb{E}[\|\nabla\mathcal{L}(\mathbf{w}_t)\|^2] + \alpha^2\sigma^2 \\
&\leq (1-\frac{\alpha}{2})\,\mathbb{E}[\|\epsilon_t\|^2] + 4\lambda H^2\eta^2\,\mathbb{E}[\|\nabla\mathcal{L}(\mathbf{w}_t)\|^2] + \alpha^2\sigma^2 \\
&\leq (1-\frac{\alpha}{2})\,\mathbb{E}[\|\epsilon_t\|^2] + \frac{8H^2\eta^2}{\alpha}\,\mathbb{E}[\|\nabla\mathcal{L}(\mathbf{w}_t)\|^2] + \alpha^2\sigma^2 \\
\frac{3\eta}{2\alpha}\,\mathbb{E}[\|\epsilon_{t+1}\|^2 - \|\epsilon_t\|^2] &\leq -\frac{3\eta}{4}\,\mathbb{E}[\|\epsilon_t\|^2] + \frac{6H^2\eta^3}{\alpha^2}\,\mathbb{E}[\|\nabla\mathcal{L}(\mathbf{w}_t)\|^2] + \frac{3\eta\alpha}{4}\sigma^2
\end{aligned}$$

Now, if we set $\eta = \frac{\alpha}{\sqrt{48}H}$ (which satisfies $\eta \leq \frac{\alpha}{4H}$), this simplifies to:

$$\frac{3\eta}{2\alpha}\,\mathbb{E}[\|\epsilon_{t+1}\|^2 - \|\epsilon_t\|^2] \leq -\frac{3\eta}{4}\,\mathbb{E}[\|\epsilon_t\|^2] + \frac{\eta}{8}\,\mathbb{E}[\|\nabla\mathcal{L}(\mathbf{w}_t)\|^2] + 3\sqrt{3}H\eta^2\sigma^2$$

And the Lemma follows by observing $\frac{3\eta}{2\alpha} = \frac{\sqrt{3}}{8H}$ □

Theorem 20.3. *Algorithm 15 with* $\eta = \frac{\alpha}{\sqrt{48}H}$ *and* $\alpha = \min\left(\frac{1}{2}, \frac{\sqrt{32\sqrt{3}H\Delta + 12\sigma^2}}{\sigma\sqrt{T}}\right)$ *guarantees*

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[\|\nabla\mathcal{L}(\mathbf{w}_t)\|^2] \leq O\left(\frac{H\Delta}{T} + \frac{\sigma\sqrt{H\Delta} + \sigma^2}{\sqrt{T}}\right)$$

Proof. The proof works using the so-called *potential* method. We define the value

$$\Phi_t = \mathcal{L}(\mathbf{w}_t) + \frac{\sqrt{3}}{8H}\|\epsilon_t\|^2$$

We will then show that Φ_t roughly decreases with t at rate that depends on $\|\nabla\mathcal{L}(\mathbf{w}_t)\|^2$. Specifically:

$$\mathbb{E}[\Phi_{t+1} - \Phi_t] = \mathbb{E}[\mathcal{L}(\mathbf{w}_{t+1}) - \mathcal{L}(\mathbf{w}_t) + \frac{\sqrt{3}}{8H}\|\epsilon_{t+1}\|^2 - \frac{\sqrt{3}}{8H}\|\epsilon_t\|^2]$$

applying Lemmas 20.1 and 20.2:

$$\begin{aligned}
&\leq -\frac{\eta}{4}\,\mathbb{E}[\|\nabla\mathcal{L}(\mathbf{w}_t)\|^2] + \frac{3\eta}{4}\,\mathbb{E}[\|\epsilon_t\|^2] \\
&\quad - \frac{3\eta}{4}\,\mathbb{E}[\|\epsilon_t\|^2] + \frac{\eta}{8}\,\mathbb{E}[\|\nabla\mathcal{L}(\mathbf{w}_t)\|^2] + 3\sqrt{3}H\eta^2\sigma^2
\end{aligned}$$

(Notice that the $\|\epsilon_t\|^2$ terms will magically cancel in the above: this was the point of including this weird-seeming $\frac{\sqrt{3}}{8H}\|\epsilon_t\|^2$ in the potential:

$$= -\frac{\eta}{8}\,\mathbb{E}[\|\nabla\mathcal{L}(\mathbf{w}_t)\|^2] + 3\sqrt{3}\eta^2\sigma^2$$

Now sum over t:

$$\mathbb{E}[\Phi_{T+1} - \Phi_1] \leq -\frac{\eta}{8}\sum_{t=1}^{T}\mathbb{E}[\|\nabla\mathcal{L}(\mathbf{w}_t)\|^2] + 3\sqrt{3}\sigma T H\eta^2$$

rearranging...

$$\sum_{t=1}^{T}\mathbb{E}[\|\nabla\mathcal{L}(\mathbf{w}_t)\|^2] \leq \frac{8}{\eta}\,\mathbb{E}[\Phi_1 - \Phi_{T+1}] + 24\sqrt{3}T\eta H\sigma^2$$

Now, observe that

$$\begin{aligned}\mathbb{E}[\Phi_1 - \Phi_{T+1}] &= \mathbb{E}[\mathcal{L}(\mathbf{w}_1) - \mathcal{L}(\mathbf{w}_{T+1}) + \frac{\sqrt{3}}{8H}\|\epsilon_1\|^2 - \frac{\sqrt{3}}{8H}\|\epsilon_{T+1}\|^2] \\ &\leq \Delta + \frac{\sqrt{3}\sigma^2}{8H}\end{aligned}$$

where in the second line we used $\mathbb{E}[\|\epsilon_1\|^2] = \mathbb{E}[\|r_1\|^2] \leq \sigma^2$. Putting all this together with $\eta = \frac{\alpha}{H\sqrt{48}\alpha}$ yields:

$$\begin{aligned}\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[\|\nabla\mathcal{L}(\mathbf{w}_t)\|^2] &\leq \frac{8}{T\eta}\mathbb{E}[\Phi_1 - \Phi_{T+1}] + 24\sqrt{3}\sigma^2 H\eta \\ &\leq \frac{32\sqrt{3}H\Delta + 12\sigma^2}{T\alpha} + 6\sigma^2\alpha\end{aligned}$$

Now, we would like to balance these terms and set:

$$\alpha = \frac{\sqrt{32\sqrt{3}H\Delta + 12\sigma^2}}{\sigma\sqrt{T}}$$

but we also need to satisfy $\alpha \leq 1/2$. So, this motivates the setting:

$$\alpha = \min\left(\frac{1}{2}, \frac{\sqrt{32\sqrt{3}H\Delta + 12\sigma^2}}{\sigma\sqrt{T}}\right)$$

Now, if $\alpha \neq 1/2$, we have:

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[\|\nabla\mathcal{L}(\mathbf{w}_t)\|^2] \leq \frac{2\sqrt{32\sqrt{3}H\Delta + 12\sigma^2}\sigma\sqrt{6}}{\sqrt{T}} = O\left(\frac{\sigma\sqrt{H\Delta} + \sigma^2}{\sqrt{T}}\right)$$

On the other hand, if $\alpha = 1/2$, this implies

$$\frac{\sqrt{32\sqrt{3}H\Delta + 12\sigma^2}}{\sigma\sqrt{T}} \geq 1/2$$

so that:

$$\frac{\sigma^2}{4} \leq \frac{32\sqrt{3}H\Delta + 12\sigma^2}{T}$$

so that overall:

$$\begin{aligned}\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[\|\nabla\mathcal{L}(\mathbf{w}_t)\|^2] &\leq \frac{64\sqrt{3}H\Delta + 24\sigma^2}{T\alpha} + 3\sigma^2 \\ &\leq O\left(\frac{H\Delta + \sigma^2}{T}\right)\end{aligned}$$

so that adding the two bounds for the two cases yields the desired result. □

21 Distributed/Large-Scale Systems

Today there are many problems for which we are fortunate to have a huge abundance of training data. This in turn has helped fuel the rise of extremely large models with millions or billions of parameters. This brings a new computational and systems dimension to designing training algorithms.

1. In order to process the data in a reasonable amount of time, we really need to process it in parallel.
2. In some cases, the model may be so large that even evaluating a single loss $\ell(\mathbf{w}, z)$ is slow, so that we would like to evaluate this in parallel as well.

These two considerations lead to what are called *data parallel* and *model parallel* paradigms for parallel or distributed machine learning. Model parallelism typically refers to the case in which the computation of $\ell(\mathbf{w}, z)$ is distributed over multiple *different computers* that communicate over some network rather than the case of speeding up matrix multiplies in a neural network by using parallel computation on a GPU. Model parallelism is far less common that data parallelism, so we will spend much more time talking about the latter.

22 Data parallelism

Data parallelism is essentially a fancy name for minibatching. At the most basic, we simply want to calculate a large minibatch gradient:

$$\mathbf{g}_t = \frac{1}{B}\sum_{i=1}^{B} \nabla \ell(\mathbf{w}_t, z_{t,i})$$

and then take a gradient descent step:

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \mathbf{g}_t$$

The issue is that B is extremely large so we would like to compute $\mathbf{g}_t$ in parallel. The standard algorithm for this is the following:

Algorithm 16 Parallel Minibatch SGD

for $t = 1 \dots T$ **do**
  for $i = 0 \dots M-1$ **do**
    (Asynchronously) ask processor i to compute $\mathbf{g}_{t,i} = \sum_{j=iB/M}^{(i+1)B/M-1} \nabla \ell(\mathbf{w}_t, z_{t,j})$.
  end for
  Wait for all processors to finish their tasks.
  Set $\mathbf{g}_t = \frac{1}{B}\sum_{i=0}^{M} \mathbf{g}_{t,i}$.
  $\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \mathbf{g}_t$.
end for

In an ideal world, each iteration of this algorithm would take $O(B/M)$ time, as each processor takes $O(B/M)$ time to compute its individual sum. In reality, however, things are somewhat more complicated. One issue is the step "wait for all processors to finish their tasks". This synchronization step can significantly slow down the computation in practice. Further, just launching the different jobs on the different processors can carry some significant overhead in terms of setting up or scheduling tasks on a thread. As a result, typically there is a limit beyond which making M bigger will actually result in slower computation.

In order to mitigate this problem Niu et al. 2011 introduced a simple idea, which they call Hogwild!: what if you just don't wait for all the processors to finish. That is, each processor performs the following procedure:

In this algorithm, we've carefully written out the update $\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \mathbf{g}_t$ into the individual operations over the coordinates in order to make clearer what can happen. Specifically, if i_1 and i_2 are two different threads, and $\mathbf{w} \in \mathbb{R}^2$, it is possible for the following order of operations to take place:

1. i_1 performs update $\mathbf{w}[0] = \mathbf{w}[0] - \eta \mathbf{g}_{t,i_1}[0]$.

Algorithm 17 Hogwild! SGD worker algorithm

i is the worker id.
$\mathbf{w}$ is stored in a *shared memory location* across all workers.
for $t = 1 \dots T$ **do**
  Sample $z_{t,i}$.
  Compute $\mathbf{g}_{t,i} = \nabla \ell(\mathbf{w}, z_{t,i})$.
  for $j = 0 \dots d-1$ **do**
    $\mathbf{w}[j] = \mathbf{w}[j] - \eta \mathbf{g}_{t,i}[j]$.
  end for
end for

2. i_2 computes gradient $\mathbf{g}_{t,i_2} = \nabla \ell(\mathbf{w}, z_{t,i_2})$.
3. i_2 finishes its gradient descent update with $\mathbf{w}[1] = \mathbf{w}[1] - \eta \mathbf{g}_{t,i_2}$.

In this setting, the gradient $\mathbf{g}_{t,i_2}$ is computed at a *partially udpated* $\mathbf{w}$, so it is really the gradient at the wrong point! This is what the "wait for all processors to finish their tasks" was supposed to prevent. The basic idea of Hogwild! is to just totally ignore this issue and hope that it does not cause any significant problems. This style of updating is called *lock-free* because we are not "locking" the gradient update in order to let one worker completely finish before allowing another worker to operate on the gradient.

That said, we can show that under some reasonable assumptions, this lock-free update actually isn't so bad. Instead of actual Hogwild!, we'll analyze the following easier-to-analyze algorithm:

Algorithm 18 Simpler Hogwild! SGD worker algorithm

i is the worker id.
$\mathbf{w}$ is stored in a *shared memory location* across all workers.
for $t = 1 \dots T$ **do**
  Sample $z_{t,i}$.
  Compute $\mathbf{g}_{t,i} = \nabla \ell(\mathbf{w}, z_{t,i})$.
  Choose an index $j_{t,i}$ uniformly at random from $0, \dots, d-1$:
  $\mathbf{w}[j_{t,i}] = \mathbf{w}[j_{t,i}] - d\eta \mathbf{g}_{t,i}[j_{t,i}]$.
end for

We also need the following terminologies:

- Define $\mathbf{w}_t$ to be the value of $\mathbf{w}$ after t different updates have been performed, regardless of which worker performs the update (if updates occur at the same time, break ties arbitrarily). Note that this index t is different from the worker's internal index t.
- Define $k(t)$ to be the index such that the tth update to $\mathbf{w}$ uses a gradient evaluated at $\mathbf{w}_{k(t)}$.
- Define $p(t)$ to be the thread index from 0 to $M-1$ such that $\mathbf{w}_{t+1}$ generated by thread $p(t)$.
- Define $j(t)$ to be the coordinate that was updated to produce $\mathbf{w}_{t+1}$.

The definition of $k(t)$ implicitly assumes that the computation of $\nabla \ell(\mathbf{w}, z_{t,i})$ really does take place at a single value of $\mathbf{w}_t$. That is, if $\mathbf{w}$ is updated in the middle of computing $\nabla \ell(\mathbf{w}, z_{t,i})$, this should either not effect the computation, or result in the gradient being the gradient at the updated value. An example of a loss ℓ that would have such a property is if ℓ depends on $\mathbf{w}$ only through some inner-product, as in a linear regression problem:

$$\ell(\mathbf{w}, z_{t,i}) = f(\langle \mathbf{x}_{t,i}, \mathbf{x} \rangle, \mathbf{y}_{t,i})$$

for some function f. In this case, when we compute the inner product $\langle \mathbf{x}_{t,i}, \mathbf{x} \rangle$, we look at each coordinate of $\mathbf{w}$ only once. This issues is basically the only reason we are making the "random coordinate" modification to the optimizer, as it makes the analysis much simpler.

Note that in this section we are completely ignoring the issues of *data races*, which could potentially be a serious problem. To mitigate this, all of the operations on the coordinates of $\mathbf{w}$ need to be carefully wrapped in atomic primitives.

Theorem 22.1. *Suppose that ℓ is H-smooth and that $\|\nabla \ell(\mathbf{w}, z)\| \leq G$ for all $\mathbf{w}$ and z. Suppose further $t - k(t) \leq \tau$ for all t for some τ and define $\Delta = \mathcal{L}(\mathbf{w}_1) - \mathcal{L}(\mathbf{w}_\star)$. Then:*

$$\sum_{t=1}^{T} \mathbb{E}[\|\mathcal{L}(\mathbf{w}_t)\|^2] \leq \frac{\Delta}{\eta} + \eta T \frac{2dGH\tau + HG^2}{2}$$

In particular, with $\eta = \frac{\sqrt{2\Delta}}{\sqrt{T(2GH\tau + HG^2}}$, we have:

$$\frac{1}{T}\sum_{t=1}^{T} \mathbb{E}[\|\mathcal{L}(\mathbf{w}_t)\|^2] \leq 2\frac{\sqrt{\Delta(dGH\tau + HG^2/2)}}{\sqrt{T}}$$

Note that this scales in a reasonable way with τ: τ is measuring the degree to which not synchronizing the workers results in a different operation than doing the correct synchronization. Thus, it makes sense that the performance degrades as τ increases. Further, if $\tau = O(T)$, then we should expect essentially no guarantee, which is exactly what the result shows.

Proof. Since $\mathcal{L}$ is H-smooth and $\|\nabla \ell(\mathbf{w}, z)\|_\infty \leq G$, we have that after for any t, t':

$$\begin{aligned}\|\mathbf{w}_t - \mathbf{w}_{t'}\| &\leq \eta dG|t - t'| \\ \|\nabla \mathcal{L}(\mathbf{w}_t) - \nabla \mathcal{L}(\mathbf{w}_{t'})\| &\leq \eta dGH|t - t'|\end{aligned}$$

Further, we have:

$$\mathbf{w}_{t+1}[j(t)] = \mathbf{w}_t[j(t)] - \eta d \nabla \ell(\mathbf{w}_{k(t)}, z_{k(t),p(t)})[j(t)]$$

Therefore:

$$\begin{aligned}\mathbf{w}_{t+1} - \mathbf{w}_t &= -\eta d \nabla \ell(\mathbf{w}_{k(t)}, z_{k(t),p(t)})[j(t)] \\ &= -\eta(d\nabla \ell(\mathbf{w}_{k(t)}, z_{k(t),p(t)})[j(t)] - \nabla \mathcal{L}(\mathbf{w}_{k(t)})) + \eta(\nabla \mathcal{L}(\mathbf{w}_t) - \nabla \mathcal{L}(\mathbf{w}_{k(t)})) - \eta \nabla \mathcal{L}(\mathbf{w}_t)\end{aligned}$$

Now, notice that $\mathbb{E}[d\nabla \ell(\mathbf{w}_t, z_{k(t),p(t)})[j(t)] - \nabla \mathcal{L}(\mathbf{w}_t)] = 0$ and $\|\nabla \ell(\mathbf{w}_t, z_{k(t),p(t)}) - \nabla \ell(\mathbf{w}_{k(t)}, z_{k(t),p(t)})\| \leq \eta dGH|t - k(t)| \leq \eta dGH\tau$. Therefore:

$$\begin{aligned}\mathcal{L}(\mathbf{w}_{t+1}) &\leq \mathcal{L}(\mathbf{w}_t) + \langle \nabla \mathcal{L}(\mathbf{w}_t), \mathbf{w}_{t+1} - \mathbf{w}_t \rangle + \frac{H}{2}\|\mathbf{w}_{t+1} - \mathbf{w}_t\|^2 \\ \mathbb{E}[\mathcal{L}(\mathbf{w}_{t+1})] &\leq \mathbb{E}[\mathcal{L}(\mathbf{w}_t)] - \eta\, \mathbb{E}[\|\mathcal{L}(\mathbf{w}_t)\|^2] + \eta^2 dGH\tau + \frac{H\eta^2 G^2}{2} \\ \sum_{t=1}^{T} \mathbb{E}[\|\mathcal{L}(\mathbf{w}_t)\|^2] &\leq \frac{\Delta}{\eta} + \eta T \frac{2dGH\tau + HG^2}{2}\end{aligned}$$

□

We can make a similar analysis for convex losses:

Theorem 22.2. *Suppose $\mathcal{L}$ is convex and ℓ is G-Lipschitz. Also suppose that $t - k(t) \leq \tau$ for all t and $\mathbf{w}_1 = 0$. Suppose also that $\mathbf{w}_\star = \operatorname{argmin} \mathcal{L}$ satisfies $\|\mathbf{w}_\star\|_\infty \leq R$ Consider the modified version of Algorithm 18 that makes the update:*

$$\mathbf{w}[j_{t,i}] = \Pi_{-R,R}\left[\mathbf{w}[j_{t,i}] - d\eta \mathbf{g}_{t,i}[j_{t,i}]\right]$$

Then:

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[\mathcal{L}(\mathbf{w}_t) - \mathcal{L}(\mathbf{w}_\star)] \leq \frac{dR^2}{2\eta T} + \frac{\eta d^2 G^2}{2} + 2Rd^{3/2}GH\eta\tau$$

So, setting $\eta = \frac{R}{\sqrt{T(dG^2 + R\sqrt{d}GH\tau)}}$ *we obtain:*

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[\mathcal{L}(\mathbf{w}_t) - \mathcal{L}(\mathbf{w}_\star)] \leq O\left(\frac{Rd\sqrt{dG^2 + R\sqrt{d}GH\tau}}{\sqrt{T}}\right)$$

Proof. Again, since $\mathcal{L}$ is H-smooth and ℓ is G-Lipschitz,

$$\begin{aligned}\|\mathbf{w}_t - \mathbf{w}_{t'}\| &\leq \eta dG|t - t'| \\ \|\nabla\mathcal{L}(\mathbf{w}_t) - \nabla\mathcal{L}(\mathbf{w}_{t'})\| &\leq \eta dGH|t - t'|\end{aligned}$$

Further, notice that $\|\mathbf{w}_t - \mathbf{w}_\star\|_\infty \leq 2R$ for all t. Next

$$\begin{aligned}\sum_{t=1}^{T}\mathbb{E}[\mathcal{L}(\mathbf{w}_t) - \mathcal{L}(\mathbf{w}_\star)] &\leq \sum_{t=1}^{T}\mathbb{E}[\langle\nabla\mathcal{L}(\mathbf{w}_t), \mathbf{w}_t - \mathbf{w}_\star\rangle] \\ &\leq \sum_{t=1}^{T}\mathbb{E}[\langle\nabla\mathcal{L}(\mathbf{w}_{k(t)}), \mathbf{w}_t - \mathbf{w}_\star\rangle] + \langle\nabla\mathcal{L}(\mathbf{w}_t) - \nabla\mathcal{L}(\mathbf{w}_{k(t)}), \mathbf{w}_t - \mathbf{w}_\star\rangle] \\ &\leq \sum_{t=1}^{T}\mathbb{E}[\langle\nabla\mathcal{L}(\mathbf{w}_{k(t)}), \mathbf{w}_t - \mathbf{w}_\star\rangle] + 2R\sum_{t=1}^{T}\sum_{i=0}^{d-1}|(\nabla\mathcal{L}(\mathbf{w}_t) - \nabla\mathcal{L}(\mathbf{w}_{k(t)}))[i]| \\ &\leq \sum_{t=1}^{T}\mathbb{E}[\langle\nabla\mathcal{L}(\mathbf{w}_{k(t)}), \mathbf{w}_t - \mathbf{w}_\star\rangle] + 2Rd^{3/2}GH\eta\tau T \\ &= \sum_{i=0}^{d-1}\sum_{t=1}^{T}\mathbb{E}[\nabla\mathcal{L}(\mathbf{w}_{k(t)})[i](\mathbf{w}_t[i] - \mathbf{w}_\star[i])] + 2Rd^{3/2}GH\eta\tau T\end{aligned}$$

So, we now need to understand $\sum_{t=1}^{T}\mathbb{E}[\nabla\mathcal{L}(\mathbf{w}_{k(t)})_i(\mathbf{w}_{t,i} - \mathbf{w}_{\star,i})]$. For this, the analysis looks similar to the convex SGD argument for per-coordinate learning rates. First, define $\mathbf{s}_t$ to be a vector that is all zeros except for:

$$\mathbf{s}_t[j(t)] = d\nabla\ell(\mathbf{w}_{k(t)}, z_{k(t),p(t)})[j(t)]$$

Notice that since j is sampled uniformly at random, we have:

$$\begin{aligned}\mathbb{E}[\mathbf{s}_t] &= \nabla\mathcal{L}(\mathbf{w}_{k(t)}) \\ \mathbb{E}[\mathbf{s}_t[i]^2] &\leq dG^2 \\ \mathbb{E}[s_t[i](\mathbf{w}_t[i] - \mathbf{w}_\star[i])] &= \mathbb{E}[\nabla\mathcal{L}(\mathbf{w}_{k(t)})[i](\mathbf{w}_t[i] - \mathbf{w}_\star[i])]\end{aligned}$$

Further, we have:

$$\mathbf{w}_{t+1}[i] = \Pi_{[-R,R]}\mathbf{w}_t[i] - \eta\mathbf{s}_t[i]$$

So now we proceed:

$$\begin{aligned}
(\mathbf{w}_{t+1}[i] - \mathbf{w}_\star[i]) &= (\mathbf{w}_t[i] - \eta \mathbf{s}_t[i] - \mathbf{w}_\star[i])^2 \\
&= (\mathbf{w}_t[i] - \mathbf{w}_\star[i])^2 - 2\eta \mathbf{s}_t[i](\mathbf{w}_t[i] - \mathbf{w}_\star[i]) + \eta^2 \mathbf{s}_t[i]^2 \\
\sum_{t=1}^{T} \mathbf{s}_t[i](\mathbf{w}_t[i] - \mathbf{w}_\star[i]) &\leq \frac{\mathbf{w}_\star[i]^2}{2\eta} + \frac{\eta}{2} \sum_{t=1}^{T} \mathbf{s}_t[i]^2 \\
\sum_{t=1}^{T} \mathbb{E}[\mathbf{s}_t[i](\mathbf{w}_t[i] - \mathbf{w}_\star[i])] &\leq \frac{R^2}{2\eta} + \frac{\eta T d G^2}{2} \\
\sum_{t=1}^{T} \mathbb{E}[\nabla \mathcal{L}(\mathbf{w}_{k(t)})[i](\mathbf{w}_t[i] - \mathbf{w}_\star[i])] &\leq \frac{\mathbf{w}_\star[i]^2}{2\eta} + \frac{\eta}{2} \sum_{t=1}^{T} \mathbf{s}_t[i]^2 \\
\sum_{t=1}^{T} \mathbb{E}[\mathbf{s}_t[i](\mathbf{w}_t[i] - \mathbf{w}_\star[i])] &\leq \frac{R^2}{2\eta} + \frac{\eta T d G^2}{2}
\end{aligned}$$

Putting this all together, we have:

$$\mathbb{E}\left[\sum_{t=1}^{T} \mathcal{L}(\mathbf{w}_t) - \mathcal{L}(\mathbf{w}_\star)\right] \leq \frac{dR^2}{2\eta} + \frac{\eta T d^2 G^2}{2} + 2Rd^{3/2} G H \eta \tau T$$

□

22.1 Parameter Server architecture

The Hogwild! paradigm is designed for a single computer running multiple threads in parallel using a shared memory. However, this is unlikely to scale to a huge number of threads - typically even a very high-quality computer may only have around O(100) cores. To go further, we need to network together multiple computers. In principle, this allows us to increase the parallelism arbitrarily highly, but there is a significant hurdle: now the different machines must communicate over a network rather than simply using a shared RAM. As a result, the communication overhead can quickly become extremely onerous. This leads to significant new implementation challenges. The standard method to deal with this is the *parameter server* architecture. Please see the following excellent source (which is co-written by some of the original proposers of the parameter server) for details on how this would work `https://d2l.ai/chapter_computational-performance/parameterserver.html`

23 Regularization and Optimization

One of the most common strategies in building machine learning systems is *regularization*. For example, it is typical to encourage model parameter $\mathbf{w}$ to be small by augmenting the loss with a penalty for large $\mathbf{w}$ as in the following:

$$\mathcal{L}(\mathbf{w}) + \frac{\lambda}{2}\|\mathbf{w}\|^2$$

where the λ value is sometimes called the weight-decay parameter, or the l2-regularization constant. Typically, this is justified in machine learning courses by claiming that small $\mathbf{w}$ values are in some sense "simpler" and so avoid overfitting. Making this idea concrete is deceptively difficult, involving a foray into statistical learning theory. However, it turns out that we can also justify regularization from an optimization point of view: adding regularization makes the function *easier to optimize*. This type of justification is in many ways simpler than the statistical learning theory viewpoint - the standard SLT approach would not really suggest why one should use a penalty proportional to $\|\mathbf{w}\|^2$ rather than, say $\|\mathbf{w}\|$ or $\|\mathbf{w}\|^4$ or $\sqrt{\|\mathbf{w}\|}$, all of which still encourage $\mathbf{w}$ to be small and so therefore "simpler".

Let's try to get some intuition behind why adding $\|\mathbf{w}\|^2$ to the objective might make it easier to optimize. First, let us define

$$\mathbf{w}_\star = \operatorname{argmin} \mathcal{L}(\mathbf{w})$$
$$\mathbf{w}_\star^\lambda = \operatorname{argmin} \mathcal{L}(\mathbf{w}) + \frac{\lambda}{2}\|\mathbf{w}\|^2$$

Intuitively, we might expect that $\|\mathbf{w}_\star^\lambda\| \leq \|\mathbf{w}_\star\|$. This is indeed the case:

Lemma 23.1. *Suppose $\lambda_1 \geq \lambda_2$. Then $\|\mathbf{w}_\star^{\lambda_1}\| \leq \|\mathbf{w}_\star^{\lambda_2}\|$.*

Proof.

$$\begin{aligned}
\mathcal{L}(\mathbf{w}_\star^{\lambda_1}) + \frac{\lambda_1}{2}\|\mathbf{w}_\star^{\lambda_1}\|^2 &\leq \mathcal{L}(\mathbf{w}_\star^{\lambda_2}) + \frac{\lambda_1}{2}\|\mathbf{w}_\star^{\lambda_2}\|^2 \\
&\leq \mathcal{L}(\mathbf{w}_\star^{\lambda_2}) + \frac{\lambda_2}{2}\|\mathbf{w}_\star^{\lambda_2}\|^2 + \frac{\lambda_1 - \lambda_2}{2}\|\mathbf{w}_\star^{\lambda_2}\|^2 \\
&\leq \mathcal{L}(\mathbf{w}_\star^{\lambda_1}) + \frac{\lambda_2}{2}\|\mathbf{w}_\star^{\lambda_1}\|^2 + \frac{\lambda_1 - \lambda_2}{2}\|\mathbf{w}_\star^{\lambda_2}\|^2 \\
\frac{\lambda_1 - \lambda_2}{2}\|\mathbf{w}_\star^{\lambda_1}\|^2 &\leq \frac{\lambda_1 - \lambda_2}{2}\|\mathbf{w}_\star^{\lambda_2}\|^2
\end{aligned}$$

now divide by $\frac{\lambda_1 - \lambda_2}{2} \geq 0$ to conclude the result. □

Further, we have the following:

Lemma 23.2. *Suppose that $\mathcal{L}$ is G-Lipschitz and differentiable. Then*

$$\|\mathbf{w}_\star^\lambda\| \leq \frac{G}{\lambda}$$

Proof. Since $\mathcal{L}$ is differentiable, so is $\mathcal{L}(\mathbf{w}) + \frac{\lambda}{2}\|\mathbf{w}\|^2$, and the derivative at $\mathbf{w}_\star^\lambda$ should be zero. Thus:

$$\begin{aligned}
\nabla\mathcal{L}(\mathbf{w}_\star^\lambda) + \lambda\mathbf{w}_\star^\lambda &= 0 \\
\|\mathbf{w}_\star^\lambda\| &= \frac{\|\nabla\mathcal{L}(\mathbf{w}_\star^\lambda)\|}{\lambda} \\
&\leq \frac{G}{\lambda}
\end{aligned}$$

□

This Lemma tells us that as λ increases, there is ball of radius G/λ for which we can certify that the solution lies in the ball. Therefore in some sense the "search space" is decreasing as λ increases, thus making the task of finding $\mathbf{w}_\star^\lambda$ easier.

However, this is far from a rigorous proof. To make things rigorous, we will introduce the notion of *strong convexity*.

Definition 23.3. *A twice-differentiable function $\mathcal{L}$ is μ-strongly convex if $\nabla^2 L(x) \succeq \mu I$ (recall that $A \succeq B$ if $v \top Av \geq v^\top Bv$ for all v).*

Strong convexity can also be characterized by the following:

Lemma 23.4. *A twice differentiable convex function $\mathcal{L}$ is μ-strongly convex if and only if for all x and y,*

$$\mathcal{L}(y) \geq \mathcal{L}(x) + \rangle \nabla \mathcal{L}(x), y - x\rangle + \frac{\mu}{2}\|y - x\|^2$$

Proof. First, suppose that $\mathcal{L}$ is strongly-convex. Then we proceed in a manner very analogous to the smoothness lemma. Apply the fundamental theorem of calculus twice to obtain:

$$\begin{aligned}
\mathcal{L}(y) &= \mathcal{L}(x) + \int_0^1 \langle \nabla \mathcal{L}(x + t(y - x)), (y - x)\rangle \, dt \\
&= \mathcal{L}(x) + \langle \nabla \mathcal{L}(x), (y - x)\rangle + \int_0^1 \int_0^1 t(y - x)^\top \nabla^2 \mathcal{L}(x + kt(y - x))(y - x)\rangle \, dk \, dt \\
&\geq \mathcal{L}(x) + \langle \nabla \mathcal{L}(x), (y - x)\rangle + \mu\|x - y\|^2 \int_0^1 \int_0^1 t \, dk \, dt \\
&= \mathcal{L}(x) + \langle \nabla \mathcal{L}(x), (y - x)\rangle + \frac{\mu\|x - y\|^2}{2}
\end{aligned}$$

For the other direction, suppose $\mathcal{L}(y) \geq \mathcal{L}(x) + \rangle \nabla \mathcal{L}(x), y - x\rangle + \frac{\mu}{2}\|y - x\|^2$ for all x and y. Then consider the function $f(t) = \mathcal{L}(x + t\delta)$ for some arbitrary x and vector δ. We have:

$$\begin{aligned}
f'(t) &= \langle \nabla \mathcal{L}(x + t\delta), \delta\rangle \\
f''(t) &= \delta^\top \nabla^2 \mathcal{L}(x + t\delta)\delta
\end{aligned}$$

Further, the condition $\mathcal{L}(y) \geq \mathcal{L}(x) + \rangle \nabla \mathcal{L}(x), y - x\rangle + \frac{\mu}{2}\|y - x\|^2$ applied to both the pair $x, x + t\delta$ and $x + t\delta, x$ implies:

$$\begin{aligned}
f(h) &\geq f(0) + f'(0)h + \frac{\mu}{2}h^2\|\delta\|^2 \\
f(0) &\geq f(h) - f'(h)h + \frac{\mu}{2}h^2\|\delta\|^2 \\
\frac{f'(h) - f(0)}{h} &\geq \mu\|\delta\|^2
\end{aligned}$$

Now, be definition of derivative:

$$\begin{aligned}
\delta^\top \nabla^2 \mathcal{L}(x)\delta &= f''(0) \\
&= \lim_{h \to 0} \frac{f'(h) - f'(0)}{h} \\
&\geq \mu\|\delta\|^2
\end{aligned}$$

□

This lemma provides a kind of opposite to the standard smoothness lemma: now the inequality is in the other direction. This will enable faster convergence by having the gradient norm be an *upper bound* for the suptimality $\mathcal{L}(\mathbf{w}) - \mathcal{L}(\mathbf{w})_\star$. Specifically, we have the following result:

Lemma 23.5. *Suppose that $\mathcal{L}$ is μ-strongly convex. Then for all* $\mathbf{w}$*,*

$$\mathcal{L}(\mathbf{w}) - \mathcal{L}(\mathbf{w}_\star) \leq \frac{\|\nabla \mathcal{L}(\mathbf{w})\|^2}{2\mu}$$

Proof. By Lemma 23.4, we have:

$$\begin{aligned}\mathcal{L}(\mathbf{w}_\star) &\geq \mathcal{L}(\mathbf{w}) + \langle \nabla \mathcal{L}(\mathbf{w}), \mathbf{w}_\star - \mathbf{w} \rangle + \frac{\mu}{2}\|\mathbf{w}_\star - \mathbf{w}\|^2 \\ &\geq \inf_\delta \mathcal{L}(\mathbf{w}) + \langle \nabla \mathcal{L}(\mathbf{w}), \delta \rangle + \frac{\mu}{2}\|\delta\|^2 \\ &= \mathcal{L}(\mathbf{w}) - \frac{\|\nabla \mathcal{L}(\mathbf{w})\|^2}{2\mu}\end{aligned}$$

rearranging proves the result □

Thus, using our standard smoothness result we have the gradient descent with learning rate $\eta = \frac{1}{H}$ on an H-smooth loss satisfies:

$$\mathcal{L}(\mathbf{w}_{t+1}) \leq \mathcal{L}(\mathbf{w}_t) - \frac{\|\nabla \mathcal{L}(\mathbf{w}_t)\|^2}{2H}$$

Plugging in the result of Lemma 23.5 yields:

$$\begin{aligned}\mathcal{L}(\mathbf{w}_{t+1}) - \mathcal{L}(\mathbf{w}_\star) &\leq \mathcal{L}(\mathbf{w}_t) - \mathcal{L}(\mathbf{w}_\star) - \frac{\|\nabla \mathcal{L}(\mathbf{w}_t)\|^2}{2H} \\ &\leq \mathcal{L}(\mathbf{w}_t) - \mathcal{L}(\mathbf{w}_\star) - \frac{\mu}{H}(\mathcal{L}(\mathbf{w}_t) - \mathcal{L}(\mathbf{w}_\star)) \\ &\leq (1 - \frac{\mu}{H})(\mathcal{L}(\mathbf{w}_t) - \mathcal{L}(\mathbf{w}_\star))\end{aligned}$$

unrolling this for t iterations:

$$\begin{aligned}&\leq \left(1 - \frac{\mu}{H}\right)^t (\mathcal{L}(\mathbf{w}_1) - \mathcal{L}(\mathbf{w}_\star)) \\ &= \exp(t \log\left(1 - \frac{\mu}{H}\right)(\mathcal{L}(\mathbf{w}_1) - \mathcal{L}(\mathbf{w}_\star)) \\ &\leq \exp\left(-\frac{\mu}{H}t\right)(\mathcal{L}(\mathbf{w}_1) - \mathcal{L}(\mathbf{w}_\star))\end{aligned}$$

Thus, we have shown:

Theorem 23.6. *Suppose $\mathcal{L}$ is μ-strongly convex and H-smooth. Then gradient descent with learning rate $\eta = \frac{1}{H}$ guarantees:*

$$\mathcal{L}(\mathbf{w}_{t+1}) - \mathcal{L}(\mathbf{w}_\star) \leq \exp\left(-\frac{\mu}{H}t\right)(\mathcal{L}(\mathbf{w}_1) - \mathcal{L}(\mathbf{w}_\star))$$

This is somewhat remarkable: the error is decreasing *exponentially fast*! All of our previous rates have only had a polynomial dependence on t, so this is truly much much faster than was previously possible. Of course, there is a catch: the exponent depends on the ratio $\frac{\mu}{H}$. This quantity is often called the *condition number* of the problem, and in practice on machine learning problems it can be very very small. In fact, on a dataset of size N it is not unusual to have $\mu \propto \frac{1}{\sqrt{N}}$, so that the condition number is also $\propto 1/\sqrt{N}$. This is so small that in fact the exponential rate may look only polynomial!

Strong convexity and Regularization

The study of strongly-convex functions is linked to regularization via the following important result:

Lemma 23.7. *Suppose that $\mathcal{L}$ is a convex function. Then $\mathcal{L}(\mathbf{w}) + \frac{\lambda}{2}\|\mathbf{w}\|^2$ is λ-strongly convex.*

Proof. Let $F(\mathbf{w}) = \mathcal{L}(\mathbf{w}) + \frac{\lambda}{2}\|\mathbf{w}\|^2$. Then $\nabla^2 F(\mathbf{w}) = \nabla^2 \mathcal{L}(\mathbf{w}) + \lambda I$. Thus for all v, $v^\top \nabla^2 F(\mathbf{w}) v = v^\top \nabla^2 \mathcal{L}(\mathbf{w}) v + \lambda \|v\|^2 \geq \lambda \|v\|^2$, where the final inequality follows since $\mathcal{L}$ is convex. □

This means that we can *force a convex loss to be strongly-convex by adding regularization*. However, there is an inherent tradeoff in doing so: by adding regularization, we bias the objective away from the true optimum. Thus, we want to add as little regularization as possible while still allowing for fast optimization. This is what results in very small condition numbers - the condition number will be $\frac{\lambda}{H}$, and we want to make λ as small as is feasible.

Strong-convexity and distance from optimum

Strong convexity also implies another important property:

Lemma 23.8. *Suppose $\mathcal{L}$ is μ-strongly convex. Then for all $\mathbf{w}$:*

$$\mathcal{L}(\mathbf{w}) - \mathcal{L}(\mathbf{w}_\star) \geq \frac{\mu}{2}\|\mathbf{w} - \mathbf{w}_\star\|^2$$

Proof. From Lemma 23.4:

$$\mathcal{L}(\mathbf{w}) \geq \mathcal{L}(\mathbf{w}_\star) + \langle \nabla\mathcal{L}(\mathbf{w}_\star), \mathbf{w} - \mathbf{w}_\star\rangle + \frac{\mu}{2}\|\mathbf{w} - \mathbf{w}_\star\|^2$$

Now note that $\nabla\mathcal{L}(\mathbf{w}_\star) = 0$ and rearrange to see the result. □

This is a very powerful statement: it allows us show not only convergence in objective value $\mathcal{L}(\mathbf{w}) \to \mathcal{L}(\mathbf{w}_\star)$, but also convergence in parameter value $\mathbf{w} \to \mathbf{w}_\star$.

How much regularization should we add?

Adding regularization makes the objective strongly convex, which makes the optimization process faster, but it also makes our solutions subtly incorrect. Thus there is a natural question of what the "right" amount of regularization is that would optimally tradeoff between having an accurate solution and obtaining the solution quickly.

In particular, we can ask the following question: Given that we are allowed to perform T gradient computations, how much regularization should we add to obtain the smallest possible error?

To address this question, we need to analyze two different sources of error: error from the optimization algorithm, and error from adding regularization. Specifically, we have:

$$\begin{aligned}
\mathcal{L}(\mathbf{w}_t) + \frac{\lambda}{2}\|\mathbf{w}_t\|^2 - (\mathcal{L}(\mathbf{w}_\star) + \frac{\lambda}{2}\|\mathbf{w}_\star\|^2) &\leq \mathcal{L}(\mathbf{w}_t) + \frac{\lambda}{2}\|\mathbf{w}_t\|^2 - (\mathcal{L}(\mathbf{w}_\star^\lambda) + \frac{\lambda}{2}\|\mathbf{w}_\star^\lambda\|^2) \\
\mathcal{L}(\mathbf{w}_t) - \mathcal{L}(\mathbf{w}_\star) &\leq \mathcal{L}(\mathbf{w}_t) + \frac{\lambda}{2}\|\mathbf{w}_t\|^2 - (\mathcal{L}(\mathbf{w}_\star^\lambda) + \frac{\lambda}{2}\|\mathbf{w}_\star^\lambda\|^2) + \frac{\lambda}{2}\|\mathbf{w}_\star\|^2 - \frac{\lambda}{2}\|\mathbf{w}_t\|^2 \\
&\leq \mathcal{L}(\mathbf{w}_t) + \frac{\lambda}{2}\|\mathbf{w}_t\|^2 - (\mathcal{L}(\mathbf{w}_\star^\lambda) + \frac{\lambda}{2}\|\mathbf{w}_\star^\lambda\|^2) + \frac{\lambda}{2}\|\mathbf{w}_\star\|^2
\end{aligned}$$

where the first inequality holds since $\mathbf{w}_\star^\lambda = \operatorname{argmin} \mathcal{L}(w) + \frac{\lambda}{2}\|w\|^2$. Now, let us suppose we start at $\mathbf{w}_1 = 0$ and use gradient descent on the loss $\mathcal{L}(\mathbf{w}) + \frac{\lambda}{2}\|\mathbf{w}\|^2$. Then we have:

$$\begin{aligned}
\mathcal{L}(\mathbf{w}_1) + \frac{\lambda}{2}\|\mathbf{w}_1\|^2 - (\mathcal{L}(\mathbf{w}_\star^\lambda) + \frac{\lambda}{2}\|\mathbf{w}_\star^\lambda\|^2) &\leq \mathcal{L}(\mathbf{w}_1) - \mathcal{L}(\mathbf{w}_\star^\lambda) - \frac{\lambda}{2}\|\mathbf{w}_\star^\lambda\|^2 \\
&\leq \mathcal{L}(\mathbf{w}_1) - \mathcal{L}(\mathbf{w}_\star)
\end{aligned}$$

Further, observe that $\mathcal{L}(\mathbf{w}) + \frac{\lambda}{2}\|\mathbf{w}_\star\|^2$ is $\lambda + H$-smooth if $\mathcal{L}$ is H smooth. To see this last fact, define $F(\mathbf{w}) = \mathcal{L}(\mathbf{w}) + \frac{\lambda}{2}\|\mathbf{w}_\star\|^2$. Then we have $\nabla^2 F(\mathbf{w}) = \nabla^2\mathcal{L}(\mathbf{w}) + \lambda I$, and since $\nabla^2\mathcal{L}(\mathbf{w}) \preceq HI$ since $\mathcal{L}$ is H-smooth, this implies $\nabla^2 F(\mathbf{w}) \leq (H + \lambda)I$. We also have that F is λ-strongly convex. Therefore by Theorem 23.6:

$$\begin{aligned}
F(\mathbf{w}_t) - F(\mathbf{w}_\star^\lambda) &\leq \exp\left(-\frac{\lambda}{H+\lambda}t\right)(F(\mathbf{w}_1) - F(\mathbf{w}_\star^\lambda)) \\
\mathcal{L}(\mathbf{w}_t) + \frac{\lambda}{2}\|\mathbf{w}_t\|^2 - (\mathcal{L}(\mathbf{w}_\star^\lambda) + \frac{\lambda}{2}\|\mathbf{w}_\star^\lambda\|^2) &\leq \exp\left(-\frac{\lambda}{H+\lambda}t\right)(\mathcal{L}(\mathbf{w}_1) - \mathcal{L}(\mathbf{w}_\star))
\end{aligned}$$

Putting this all together:

$$\mathcal{L}(\mathbf{w}_t) - \mathcal{L}(\mathbf{w}_\star) \leq \exp\left(-\frac{\lambda}{H+\lambda}t\right)(\mathcal{L}(\mathbf{w}_1) - \mathcal{L}(\mathbf{w}_\star)) + \frac{\lambda}{2}\|\mathbf{w}_\star\|^2$$

Now, notice that:

$$\begin{aligned}\mathcal{L}(\mathbf{w}_1) &\leq \mathcal{L}(\mathbf{w}_\star) + \langle \nabla \mathcal{L}(\mathbf{w}_\star), \mathbf{w}_1 - \mathbf{w}_\star \rangle + \frac{H}{2}\|\mathbf{w}_1 - \mathbf{w}_\star\|^2 \\ &\leq \mathcal{L}(\mathbf{w}_\star) + \frac{H}{2}\|\mathbf{w}_\star\|^2\end{aligned}$$

where we have used $\nabla \mathcal{L}(\mathbf{w}_\star) = 0$ and $\mathbf{w}_1 = 0$. Thus our bound becomes:

$$\mathcal{L}(\mathbf{w}_t) - \mathcal{L}(\mathbf{w}_\star) \leq \frac{\|\mathbf{w}_\star\|^2}{2}\left(H \exp\left(-\frac{\lambda}{H+\lambda}t\right) + \lambda\right)$$

Now to balance the two terms here exactly is somewhat complicated, but we can essentially just guess. To start, notice that $\lambda \geq H$ is unlikely to be the right answer: in this case just the λ term would already be so big that we would be not convergence. So, let us restrict attention to $\lambda \leq H$. In this case, we have

$$\exp\left(-\frac{\lambda}{H+\lambda}t\right) \leq \exp\left(-\frac{\lambda}{2H}t\right)$$

Next, let us set $c = \exp(\lambda t/2H)$ so that $\lambda = \frac{2H}{t}\log(c)$. Then we have:

$$H \exp\left(-\frac{\lambda t}{2H}\right) \leq \frac{H}{c}$$

Thus, if we consider $\log(c)$ to be an unimportant log factor, when setting $\frac{H}{c} = \lambda$ to balance terms, we would get $c = t/2$. So, if t is large enough that $\frac{2H}{t}\log(t/2) \leq H$, we can set $\lambda = \frac{2H}{t}\log(t/2)$ to obtain:

$$\begin{aligned}\mathcal{L}(\mathbf{w}_t) - \mathcal{L}(\mathbf{w}_\star) &\leq \frac{\|\mathbf{w}_\star\|^2}{2}\left(H \exp\left(-\frac{\lambda}{H+\lambda}t\right) + \lambda\right) \\ &\leq \frac{\|\mathbf{w}_\star\|^2}{2}\left(\frac{2H}{t} + \frac{2H\log(t/2)}{t}\right) \\ &= O\left(\frac{H\|\mathbf{w}_\star\|^2\log(t)}{t}\right)\end{aligned}$$

This is almost the same rate we obtained with ordinary gradient descent, but now there is an extra log factor.

So, what was the point of that? It seems that we actually didn't gain anything from adding the regularization. It's actually not so clear: remember that all of our results so far have been *upper bounds*, so it might easily hold that a small amount of regularization (smaller than suggested by this theory), would actually result in a much better output than suggested here. In particular, it might be that $\mathcal{L}$ is in some sense "almost" already strongly convex except at a few locations $\mathbf{w}$. By adding a small amount of regularization, we may encourage gradient descent to somehow quickly bypass those locations and result in an overall faster convergence rate. However, it is very difficult to predict when this will occur in practice, so typically the optimal value for λ is tuned manually.

24 Stochastic Strongly-Convex Optimization and Almost-Convexity

Previously we considered smooth and strongly-convex losses and showed that *deterministic* gradient descent is able to achieve a convergence rate of:

$$\mathcal{L}(\mathbf{w}_{T+1}) - \mathcal{L}(\mathbf{w}_\star) \leq \exp\left(-\frac{\mu}{H}T\right)(\mathcal{L}(\mathbf{w}_1) - \mathcal{L}(\mathbf{w}_\star))$$

Let's now consider the *stochastic* case in which we cannot trust our gradient evaluations to be completely accurate. Recall that previously we showed that SGD on ordinary convex functions achieves:

$$\frac{1}{T}\mathbb{E}\left[\sum_{t=1}^{T}\mathcal{L}(\mathbf{w}_t) - \mathcal{L}(\mathbf{w}_\star)\right] \leq O\left(\frac{1}{\sqrt{T}}\right)$$

Now, we will show that if $\mathcal{L}$ is known to be μ-strongly convex, then SGD can instead obtain:

$$\frac{1}{T}\mathbb{E}\left[\sum_{t=1}^{T}\mathcal{L}(\mathbf{w}_t) - \mathcal{L}(\mathbf{w}_\star)\right] \leq O\left(\frac{\log(T)}{\mu T}\right)$$

Thus, even in the stochastic case, we see a significant improvement stemming from strong-convexity. However, note that the improvement is quite a bit less dramatic than in the case of deterministic losses. The analysis is remarkably similar to the analysis for ordinary convex losses:

Theorem 24.1. *Suppose that $\mathcal{L}(\mathbf{w}) = \mathbb{E}[\ell(\mathbf{w}, z)]$ is a μ-strongly convex function satisfiying $\|\mathbf{w}_\star\| \leq D$ for some known D, and $\mathbb{E}[\|\nabla\ell(\mathbf{w}, z)\|^2] \leq G^2$ for all $\|\mathbf{w}\| \leq D$. Consider projected stochastic gradient descent with learning rate $\eta_t = \frac{1}{\mu t}$:*

$$\mathbf{w}_{t+1} = \prod_{\|\mathbf{w}\|\leq D}\left[\mathbf{w}_t - \eta_t\nabla\ell(\mathbf{w}_t, z_t)\right]$$

This algorithm satisfies:

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[\mathcal{L}(\mathbf{w}_t) - \mathcal{L}(\mathbf{w}_\star)] \leq \frac{G^2(\log(T)+1)}{2\mu}$$

Proof. The proof begins in exactly the same way as for previous convex analysis. Since $\|\mathbf{w}_\star\| \leq D$, by properties of the projection we have:

$$\begin{aligned}
\|\mathbf{w}_{t+1} - \mathbf{w}_\star\|^2 &\leq \|\mathbf{w}_t - \eta_t\nabla\ell(\mathbf{w}_t, z_t) - \mathbf{w}_\star\|^2 \\
&= \|\mathbf{w}_t - \mathbf{w}_\star\|^2 - 2\eta_t\langle\nabla\ell(\mathbf{w}_t, z_t), \mathbf{w}_t - \mathbf{w}_\star\rangle + \eta_t^2\|\nabla\ell(\mathbf{w}_t, z_t)\|^2 \\
\langle\nabla\ell(\mathbf{w}_t, z_t), \mathbf{w}_t - \mathbf{w}_\star\rangle &\leq \frac{\|\mathbf{w}_t - \mathbf{w}_\star\|^2}{2\eta_t} - \frac{\|\mathbf{w}_{t+1} - \mathbf{w}_\star\|^2}{2\eta_t} + \frac{\eta_t\|\nabla\ell(\mathbf{w}_t, z_t)\|^2}{2}
\end{aligned}$$

Now, previously we simply argued that by convexity, $\mathbb{E}[\mathcal{L}(\mathbf{w}_t) - \mathcal{L}(\mathbf{w}_\star)] \leq \mathbb{E}[\langle\nabla\ell(\mathbf{w}_t, z_t), \mathbf{w}_t - \mathbf{w}_\star\rangle]$. However, now we will instead leverage strong convexity. Recall that an equivalent characterization of strong convexity is the condition:

$$\mathcal{L}(x) \geq \mathcal{L}(y) + \langle\nabla\mathcal{L}(y), x - y\rangle + \frac{\mu}{2}\|x - y\|^2$$

From this, we can conclude:

$$\begin{aligned}
\mathcal{L}(\mathbf{w}_t) - \mathcal{L}(\mathbf{w}_\star) &\leq \langle\nabla\mathcal{L}(\mathbf{w}_t), \mathbf{w}_t - \mathbf{w}_\star\rangle - \frac{\mu}{2}\|\mathbf{w}_t - \mathbf{w}_\star\|^2 \\
\mathbb{E}[\mathcal{L}(\mathbf{w}_t) - \mathcal{L}(\mathbf{w}_\star)] &\leq \mathbb{E}\left[\langle\nabla\ell(\mathbf{w}_t, z_t), \mathbf{w}_t - \mathbf{w}_\star\rangle - \frac{\mu}{2}\|\mathbf{w}_t - \mathbf{w}_\star\|^2\right]
\end{aligned}$$

Plugging this into our previous equation yields:

$$\begin{aligned}
\langle \nabla \ell(\mathbf{w}_t, z_t), \mathbf{w}_t - \mathbf{w}_\star \rangle &\leq \frac{\|\mathbf{w}_t - \mathbf{w}_\star\|^2}{2\eta_t} - \frac{\|\mathbf{w}_{t+1} - \mathbf{w}_\star\|^2}{2\eta_t} + \frac{\eta_t \|\nabla \ell(\mathbf{w}_t, z_t)\|^2}{2} \\
\mathbb{E}\left[\mathcal{L}(\mathbf{w}_t) - \mathcal{L}(\mathbf{w}_\star) + \frac{\mu}{2}\|\mathbf{w}_t - \mathbf{w}_\star\|^2\right] &\leq \mathbb{E}\left[\frac{\|\mathbf{w}_t - \mathbf{w}_\star\|^2}{2\eta_t} - \frac{\|\mathbf{w}_{t+1} - \mathbf{w}_\star\|^2}{2\eta_t} + \frac{\eta_t \|\nabla \ell(\mathbf{w}_t, z_t)\|^2}{2}\right] \\
&\leq \mathbb{E}\left[\frac{\|\mathbf{w}_t - \mathbf{w}_\star\|^2}{2\eta_t} - \frac{\|\mathbf{w}_{t+1} - \mathbf{w}_\star\|^2}{2\eta_t} + \frac{\eta_t G^2}{2}\right] \\
\mathbb{E}[\mathcal{L}(\mathbf{w}_t) - \mathcal{L}(\mathbf{w}_\star)] &\leq \mathbb{E}\left[\frac{\|\mathbf{w}_t - \mathbf{w}_\star\|^2}{2\eta_t} - \frac{\|\mathbf{w}_{t+1} - \mathbf{w}_\star\|^2}{2\eta_t} - \frac{\mu\|\mathbf{w}_t - \mathbf{w}_\star\|^2}{2} + \frac{\eta_t G^2}{2}\right]
\end{aligned}$$

Now, it is time to invoke our particular magical choice for $\eta_t = \frac{1}{\mu t}$:

$$\begin{aligned}
\mathbb{E}[\mathcal{L}(\mathbf{w}_t) - \mathcal{L}(\mathbf{w}_\star)] &\leq \mathbb{E}\left[\frac{\|\mathbf{w}_t - \mathbf{w}_\star\|^2}{2\eta_t} - \frac{\|\mathbf{w}_{t+1} - \mathbf{w}_\star\|^2}{2\eta_t} - \frac{\mu\|\mathbf{w}_t - \mathbf{w}_\star\|^2}{2} + \frac{\eta_t G^2}{2}\right] \\
&= \mathbb{E}\left[\frac{\mu\|\mathbf{w}_t - \mathbf{w}_\star\|^2}{2t} - \frac{\mu\|\mathbf{w}_{t+1} - \mathbf{w}_\star\|^2}{2t} - \frac{\mu\|\mathbf{w}_t - \mathbf{w}_\star\|^2}{2} + \frac{G^2}{2\mu t}\right] \\
&= \mathbb{E}\left[\frac{(t-1)\mu\|\mathbf{w}_t - \mathbf{w}_\star\|^2}{2} - \frac{t\mu\|\mathbf{w}_{t+1} - \mathbf{w}_\star\|^2}{2} + \frac{G^2}{2\mu t}\right]
\end{aligned}$$

now, sum over t and telescope:

$$\begin{aligned}
\sum_{t=1}^{T} \mathbb{E}[\mathcal{L}(\mathbf{w}_t) - \mathcal{L}(\mathbf{w}_\star)] &\leq \sum_{t=1}^{T} \mathbb{E}\left[\frac{(t-1)\mu\|\mathbf{w}_t - \mathbf{w}_\star\|^2}{2} - \frac{t\mu\|\mathbf{w}_{t+1} - \mathbf{w}_\star\|^2}{2} + \frac{G^2}{2\mu t}\right] \\
&= \mathbb{E}\left[-\frac{T\mu\|\mathbf{w}_{T+1} - \mathbf{w}_\star\|^2}{2} + \sum_{t=1}^{T} \frac{G^2}{2\mu t}\right] \\
&\leq \mathbb{E}\left[\sum_{t=1}^{T} \frac{G^2}{2\mu t}\right] \\
&\leq \frac{G^2(\log(T)+1)}{2\mu}
\end{aligned}$$

□

There is one quite subtle point in the above argument: although we assumed $\|\mathbf{w}_\star\| \leq D$, it doesn't seem that it was used anywhere in the analysis! This is because the assumption $\mathbb{E}[\|\nabla \ell(\mathbf{w}, z)\|^2] \leq G^2$ actually *requires* some kind of restriction on $\mathbf{w}$. To see why, recall the following fact about strongly convex functions:

Lemma 23.5. *Suppose that $\mathcal{L}$ is μ-strongly convex. Then for all $\mathbf{w}$,*

$$\mathcal{L}(\mathbf{w}) - \mathcal{L}(\mathbf{w}_\star) \leq \frac{\|\nabla \mathcal{L}(\mathbf{w})\|^2}{2\mu}$$

This result implies that it is actually impossible to have $\mathbb{E}[\|\nabla \ell(\mathbf{w}, z)\|^2] \leq G^2$ for any fixed constant G unless we also guarantee that $\|\mathbf{w} - \mathbf{w}_\star\|$ is bounded. Since it's always possible for the noise in the stochastic gradients to push $\mathbf{w}_t$ very far away from $\mathbf{w}_\star$, we needed to assume $\|\mathbf{w}_\star\| \leq D$ and use projections to keep the distance bounded, which in turn allows for the bounded gradient assumption to hold.

Note that this result does not rely on smoothness. In fact, as you saw on the homework, adding a smoothness assumption enables us to remove the bounded gradient assumption, at the cost of an extra $\frac{H}{\mu}$ factor in the error.

Almost Convexity

At this point, we have considered losses that are H-smooth and non-convex, losses that are convex and H-smooth, and losses that are μ=strongly convex and H-smooth. When written in terms of bounds on the Hessian, these conditions correspond to:

- H-smoothness: $-HI \preceq \nabla^2 \mathcal{L}(\mathbf{w}) \preceq HI$.
- Convexity: $0 \preceq \nabla^2 \mathcal{L}(\mathbf{w})$.
- μ-strong convexity: $\mu I \preceq \nabla^2 \mathcal{L}(\mathbf{w})$.

It seems that something is missing from this categorization of the Hessian: what if $\nabla^2 \mathcal{L}(\mathbf{w})$ is bounded from below by some negative number between $-H$ and 0? This condition has a few different names in the literature, but we will call it γ-almost convexity:

Definition 24.2. *A twice-differentiable function $\mathcal{L}$ is γ-almost convex if for all $\mathbf{w}$:*

$$-\gamma I \preceq \nabla^2 \mathcal{L}(\mathbf{w})$$

Thus, all H-smooth functions are automatically H-almost convex, and all convex functions are 0-almost convex. The natural question now is whether we can take advantage of γ-almost convexity to obtain improved bounds.

On the homework, you will design an *accelerated* algorithm for H-smooth and μ-strongly convex objectives such that after T iterations, such that the algorithm outputs a point $\hat{\mathbf{w}}$ with:

$$\mathcal{L}(\hat{\mathbf{w}}) - \mathcal{L}(\mathbf{w}_\star) \leq \exp\left(-C\sqrt{\frac{\mu}{H}}T\right)\frac{H\|\mathbf{w}_\star - \mathbf{w}_1\|^2}{2}$$

where C is an absolute constant.

From this, we have the following result:

Theorem 24.3. *Run the accelerated gradient descent for strongly-convex functions for $N = \frac{\log(\frac{H^2\|\mathbf{w}_\star\|^2}{\epsilon^2})}{C\sqrt{\frac{\mu}{H}}}$ iterations starting at the origin. Then the output $\hat{\mathbf{w}}$ satisfies:*

$$\mathcal{L}(\hat{\mathbf{w}}) - \mathcal{L}(\mathbf{w}_\star) \leq \frac{\epsilon^2}{2H}$$
$$\|\nabla \mathcal{L}(\hat{\mathbf{w}})\| \leq \epsilon$$

Proof. Recall that smooth functions satisfy: $\|\nabla \mathcal{L}(\mathbf{w})\|^2 \leq 2H(\mathcal{L}(\mathbf{w}) - \mathcal{L}(\mathbf{w}_\star))$. Thus we have

$$\begin{aligned}\|\nabla \mathcal{L}(\hat{\mathbf{w}})\|^2 &\leq 2H(\mathcal{L}(\mathbf{w}) - \mathcal{L}(\mathbf{w}_\star)) \\ &\leq 2H\exp\left(-C\sqrt{\frac{\mu}{H}}\frac{\log(\frac{H^2\|\mathbf{w}_\star - \mathbf{w}_1\|^2}{\epsilon^2})}{C\sqrt{\frac{\mu}{H}}}\right)\frac{H\|\mathbf{w}_\star\|^2}{2} \\ &= \epsilon^2\end{aligned}$$

□

How can we use this fact to optimize a γ-almost convex function? The key idea is regularization. For any given point $\mathbf{w}_t$, consider the function $F_t(\delta) = \mathcal{L}(\mathbf{w}_t + \delta) + \gamma\|\delta\|^2$. Note that so long as $\mathcal{L}$ is γ-almost-convex, the hessian of this function satisfies $\nabla^2 F_t(\delta) = \nabla^2 \mathcal{L}(\mathbf{w}_t + \delta) + 2\gamma I \succeq \gamma I$, so that F_t is γ-strongly convex and $H + 2\gamma \leq 3H$ smooth. Let us further assume that $\mathcal{L}(\mathbf{w})$ is G-Lipschitz. Then if $\delta_\star = \operatorname{argmin} F_t(\delta)$, we have $\|\delta_\star\| \leq \frac{G}{2\gamma}$ because $0 = \nabla F_t(\delta_\star) = \nabla \mathcal{L}(\delta_\star) + 2\gamma\delta_\star$, so $\delta_\star = \frac{-\nabla \mathcal{L}(\delta_\star)}{2\gamma}$.

Therefore, by Theorem 24.3, after $N = \frac{\log(\frac{9H^2G^2}{4\gamma^2\epsilon^2})}{C\sqrt{\frac{\gamma}{6H}}}$ we obtain a point $\hat{\delta}$ such that:

$$F_t(\hat{\delta}) - F_t(\delta_\star) \leq \frac{\epsilon^2}{6H}$$
$$\|\nabla F_t(\hat{\delta})\| \leq \epsilon$$

Our overall algorithm will be take the following idea: given any iterate $\mathbf{w}_t$, we will form the function F_t and optimize it using accelerated gradient descent to get some $\hat{\delta}_t$. Then we set $\mathbf{w}_{t+1} = \mathbf{w}_t + \hat{\delta}_t$ and repeat the process. Why should this be a good idea? From the definition, we have that that $F_t(\hat{\delta}_t) \geq \mathcal{L}(\mathbf{w}_{t+1})$, and we might expect to be able to optimize F_t quickly because F_t is γ-strongly convex. Formally, if $\delta_{t,\star} = \operatorname{argmin} F_t(\delta)$, we have the following:

$$\begin{aligned}
\mathcal{L}(\mathbf{w}_t) - \mathcal{L}(\mathbf{w}_{t+1}) &= \mathcal{L}(\mathbf{w}_t) - \mathcal{L}(\mathbf{w}_t + \hat{\delta}_t) - \gamma\|\hat{\delta}_t\|^2 + \gamma\|\hat{\delta}_t\|^2 \\
&= F_t(0) - F_t(\hat{\delta}_t) + \gamma\|\hat{\delta}_t\|^2 \\
&\geq F_t(\delta_{t,\star}) - F_t(\hat{\delta}_t) + \gamma\|\hat{\delta}_t\|^2 \\
\mathcal{L}(\mathbf{w}_1) - \mathcal{L}(\mathbf{w}_\star) + \sum_{t=1}^{T} F_t(\hat{\delta}_t) - F_t(\delta_{t,\star}) &\geq \gamma \sum_{t=1}^{T} \|\hat{\delta}_t\|^2 \\
\mathcal{L}(\mathbf{w}_1) - \mathcal{L}(\mathbf{w}_\star) + \frac{T\epsilon^2}{6H} &\geq \gamma \sum_{t=1}^{T} \|\hat{\delta}_t\|^2
\end{aligned}$$

Now, we also have:

$$\begin{aligned}
\epsilon &\geq \|\nabla F_t(\hat{\delta}_t)\| \\
&= \|\nabla \mathcal{L}(\mathbf{w}_{t+1}) + 2\gamma\hat{\delta}_t\| \\
\epsilon + 2\gamma\|\hat{\delta}_t\| &\geq \|\nabla \mathcal{L}(\mathbf{w}_{t+1})\| \\
2\epsilon^2 + 8\gamma^2\|\hat{\delta}_t\|^2 &\geq \|\nabla \mathcal{L}(\mathbf{w}_{t+1})\|^2 \\
\frac{\epsilon^2}{4\gamma} + \gamma\|\hat{\delta}_t\|^2 &\geq \frac{\|\nabla \mathcal{L}(\mathbf{w}_{t+1})\|^2}{8\gamma}
\end{aligned}$$

Therefore, we have:

$$\begin{aligned}
\sum_{t=1}^{T} \frac{\|\nabla \mathcal{L}(\mathbf{w}_{t+1})\|^2}{8\gamma} &\leq \mathcal{L}(\mathbf{w}_1) - \mathcal{L}(\mathbf{w}_\star) + \frac{T\epsilon^2}{6H} + \frac{T\epsilon^2}{4\gamma} \\
\sum_{t=1}^{T} \|\nabla \mathcal{L}(\mathbf{w}_{t+1})\|^2 &\leq 8\gamma(\mathcal{L}(\mathbf{w}_1) - \mathcal{L}(\mathbf{w}_\star)) + \frac{4T\gamma\epsilon^2}{3H} + 2T\epsilon^2 \\
\frac{1}{T}\sum_{t=1}^{T} \|\nabla \mathcal{L}(\mathbf{w}_{t+1})\|^2 &\leq \frac{\gamma(\mathcal{L}(\mathbf{w}_1) - \mathcal{L}(\mathbf{w}_\star))}{T} + \frac{4\gamma\epsilon^2}{3H} + 2\epsilon^2 \\
&\leq \frac{\gamma(\mathcal{L}(\mathbf{w}_1) - \mathcal{L}(\mathbf{w}_\star))}{T} + 4\epsilon^2
\end{aligned}$$

Therefore, if we set $\epsilon = \sqrt{\frac{\gamma(\mathcal{L}(\mathbf{w}_1) - \mathcal{L}(\mathbf{w}_\star))}{T}}$, we have

$$\frac{1}{T}\sum_{t=1}^{T} \|\nabla \mathcal{L}(\mathbf{w}_{t+1})\|^2 \leq \frac{5\gamma(\mathcal{L}(\mathbf{w}_1) - \mathcal{L}(\mathbf{w}_\star))}{T}$$

However, the total number of iterations here is $M = NT$, so:

$$\begin{aligned}
\frac{1}{T}\sum_{t=1}^{T} \|\nabla \mathcal{L}(\mathbf{w}_{t+1})\|^2 &\leq \frac{5\gamma(\mathcal{L}(\mathbf{w}_1) - \mathcal{L}(\mathbf{w}_\star))}{M/N} \\
&\leq \frac{5\gamma(\mathcal{L}(\mathbf{w}_1) - \mathcal{L}(\mathbf{w}_\star))}{M} \frac{\log(\frac{H^2G^2}{\gamma^2\epsilon^2})}{C\sqrt{\frac{\gamma}{2H}}} \\
&\leq \frac{4\gamma(\mathcal{L}(\mathbf{w}_1) - \mathcal{L}(\mathbf{w}_\star))}{M} \frac{\log(\frac{H^2G^2T}{\gamma^3(\mathcal{L}(\mathbf{w}_1) - \mathcal{L}(\mathbf{w}_\star))})}{C\sqrt{\frac{\gamma}{2H}}} \\
&= O\left(\frac{\sqrt{\gamma H}\log(HGM/\gamma)}{M}\right)
\end{aligned}$$

There is a bit of a subtlety here: what happens if we send $\gamma \to 0$ (i.e. if the function really is convex)? Since $\lim_{\gamma\to 0}\sqrt{\gamma}\log(1/\gamma) = 0$, this seems to suggest that the algorithm converges infinitely fast! The resolution to this issue is to notice that we have implicitly assumed $T \geq 1$ so that $M \geq N$. This implies that $M \geq \tilde{O}(1/\sqrt{\gamma})$ so that in fact we cannot use $\gamma \leq 1/M^2$. Now, if we set $\gamma = 1/M^2$, then the convergence rate becomes also $1/M^2$, which is in fact consistent with our accelerated gradient descent results.

25 Variance Reduction for Convex Losses

For most of this course we have considered fairly generic stochastic optimization problems. Now, we will consider a particular kind of problem that is of great interest in machine learning: the so-called *finite-sum* objectives:

$$\mathcal{L}(\mathbf{w}) = \frac{1}{N}\sum_{i=1}^{N} \ell(\mathbf{w}, z_i)$$

For example, the *training loss* for most standard machine learning problems takes this form, where $z_1, \ldots, z_N$ represent the training set and $\ell(\mathbf{w}, z)$ is the loss on a training example. Let's consider the case that $\mathcal{L}$ is convex and H-smooth. Without invoking complicated procedures like acceleration, there are roughly two ways one might consider to apply gradient descent in this setting:

1. Use *full gradient* descent: that is, compute a gradient $\nabla\mathcal{L}(\mathbf{w}_t)$ and take a gradient descent step $\mathbf{w}_{t+1} = \mathbf{w}_t - \eta\nabla\mathcal{L}(\mathbf{w}_t)$ for $\eta = \frac{1}{H}$. This guarantees:

$$\mathcal{L}(\mathbf{w}_T) - \mathcal{L}(\mathbf{w}_\star) \leq O(1/T)$$

2. Use *stochastic* gradient descent: sample a training point z_t at random and take a step $\mathbf{w}_{t+1} = \mathbf{w}_t - \eta\nabla\ell(\mathbf{w}_t, z_t)$. Since $\mathbb{E}[\nabla\ell(\mathbf{w}_t, z_t)] = \nabla\mathcal{L}(\mathbf{w}_t)$, by setting $\eta = O(1/\sqrt{T})$, we can guarantee:

$$\frac{1}{T}\sum_{t=1}^{T} \mathbb{E}[\mathcal{L}(\mathbf{w}_t) - \mathcal{L}(\mathbf{w}_\star) \leq O(1/\sqrt{T})$$

As previously discussed, although the dependence on T is worse for the SGD approach, the total computation scales much better with N: every single iterate of the full gradient approach requires $O(N)$ computation, but every iterate of the SGD approach requires $O(1)$ computation. So in terms of total compute C, we are comparing $O(N/C)$ to $O(1/\sqrt(C))$. As long as $N^2 \geq C$, we would prefer the stochastic approach. Even if we allow for accelerated gradient descent, it turns out that regular SGD is still better for sufficiently large N.

Now, the natural question is: can we improve at all on stochastic gradient descent? Surprisingly, by leveraging the finite-sum structure of the problem, the answer is *yes*! The key idea is to create a better gradient estimator than simply sampling one element at random. The method we discuss here has been found in various forms at around the same time, but arguable the simplest presentation is Johnson and Zhang 2013. Other essentially concurrent methods include Roux, Schmidt, and Bach 2012; Shalev-Shwartz and Zhang 2013.

We will create a better gradient estimate by *occasionally* computing a full gradient. Intuitively, if we compute one single full-gradient N iterations, we will still on average use only $O(1)$ computation per iteration. By some clever trickery, we will be able to leverage this perfectly-accurate gradient to improve subsequent gradients.

Let's suppose that we know the true gradient $\nabla\mathcal{L}(\mathbf{v})$ at some point $\mathbf{v}$ which we will call the *checkpoint* value. We wish now to compute an estimate of the gradient at a *nearby* point $\mathbf{w}$. To form the estimate, we randomly sample some z, and compute:

$$\mathbf{g} = \nabla\ell(\mathbf{w}, z) - \nabla\ell(\mathbf{v}z) + \nabla\mathcal{L}(\mathbf{v})$$

The first important property of this estimate is that it is *unbiased*:

$$\begin{aligned}\mathbb{E}[\mathbf{g}] &= \mathbb{E}[\nabla\ell(\mathbf{w}, z) - \nabla\ell(\mathbf{v}z) + \nabla\mathcal{L}(\mathbf{v})] \\ &= \nabla\mathcal{L}(\mathbf{w}) - \nabla\mathcal{L}(\mathbf{v}) + \nabla\mathcal{L}(\mathbf{v}) \\ &= \nabla\mathcal{L}(\mathbf{w})\end{aligned}$$

However, we can hope that the *variance* of $\mathbf{g}$ is rather small. Why might this be? Notice that the only random part of $\mathbf{g}$ is the difference $\nabla\ell(\mathbf{w}, z) - \nabla\ell(\mathbf{v}, z)$. Now, the function ℓ is a smooth function of $\mathbf{w}$ for all z, then we have:

$$\|\nabla\ell(\mathbf{w}, z) - \nabla\ell(\mathbf{v}, z)\| \leq H\|\mathbf{w} - \mathbf{v}\|$$

so that if $\mathbf{w} \approx \mathbf{v}$, the randomness in $\mathbf{g}$ is necessarily quite small. Note however that we have subtly changed assumptions here: *assuming that ℓ is smooth is a stronger assumption than assuming that $\mathcal{L}$ is smooth.*

In fact, by leveraging convexity, we can provide a somewhat more subtle bound on the variance of $\mathbf{g}$. To this end, we have the following Lemma:

Lemma 25.1. *Suppose $\ell(\mathbf{w}, z)$ is an H-smooth convex function of $\mathbf{w}$ for all z. Then:*

$$\mathbb{E}[\|\nabla\ell(\mathbf{w}, z) - \nabla\ell(\mathbf{v}, z) + \nabla\mathcal{L}(\mathbf{v})\|^2] \leq 8H(\mathcal{L}(\mathbf{w}) - \mathcal{L}(\mathbf{w}_\star)) + 4H(\mathcal{L}(\mathbf{v}) - \mathcal{L}(\mathbf{w}_\star))$$

Proof. Recall from previous lectures the fact that for all x, and all H-smooth functions f with $x_\star = \operatorname{argmin} f(x)$:

$$\|\nabla f(x)\|^2 \leq 2H(f(x) - f(x_\star)) \tag{10}$$

Let's consider the function $f(x) = \ell(x, z) - \langle\nabla\ell(\mathbf{w}_\star, z), x - \mathbf{w}_\star\rangle$. Notice that since ℓ is H-smooth and convex, so is f (because we only added a linear function to ℓ). However, also we have that $\nabla f(\mathbf{w}_\star) = \nabla\ell(\mathbf{w}_\star) - \nabla\ell(\mathbf{w}_\star) = 0$, so that f is actually minimized at $\mathbf{w}_\star$ (which is the argmin of $\mathcal{L}$, not ℓ). Therefore:

$$\|\nabla f(x)\|^2 \leq 2H(\nabla f(x) - f(\mathbf{w}_\star))$$
$$\|\nabla\ell(x, z) - \nabla\ell(\mathbf{w}_\star, z)\|^2 \leq 2H(\ell(x, z) - \ell(\mathbf{w}_\star, z) - \langle\nabla\ell(\mathbf{w}_\star, z), x - \mathbf{w}_\star\rangle)$$

Further, by a more direct application of (10):

$$\|\nabla\mathcal{L}(\mathbf{w})\|^2 \leq 2H(\mathcal{L}(\mathbf{w}) - \mathcal{L}(\mathbf{w}_\star))$$

Now, we can compute:

$$\begin{aligned}
\mathbb{E}[\|\nabla\ell(\mathbf{w}, z) - \nabla\ell(\mathbf{v}, z) + \nabla\mathcal{L}(\mathbf{v})\|^2] &= \mathbb{E}[\|\nabla\ell(\mathbf{w}, z) - \nabla\ell(\mathbf{v}, z) + \nabla\mathcal{L}(\mathbf{v}) - \nabla\mathcal{L}(\mathbf{w}) + \nabla\mathcal{L}(\mathbf{w})\|^2] \\
&\leq 2\,\mathbb{E}[\|\nabla\ell(\mathbf{w}, z) - \nabla\ell(\mathbf{v}, z) + \nabla\mathcal{L}(\mathbf{v}) - \nabla\mathcal{L}(\mathbf{w})\|^2 + \|\nabla\mathcal{L}(\mathbf{w})\|^2] \\
&\leq 2\,\mathbb{E}[\|\nabla\ell(\mathbf{w}, z) - \nabla\ell(\mathbf{v}, z) + \nabla\mathcal{L}(\mathbf{v}) - \nabla\mathcal{L}(\mathbf{w})\|^2] + 4H(\mathcal{L}(\mathbf{w}) - \mathcal{L}(\mathbf{w}_\star)
\end{aligned}$$

Now, notice that $\mathbb{E}[\nabla\ell(\mathbf{w}, z) - \nabla\ell(\mathbf{v}, z)] = \nabla\mathcal{L}(\mathbf{w}) - \nabla\mathcal{L}(\mathbf{v})$. Further, for any random variable X, $\mathbb{E}[\|X - \mathbb{E}[X]\|^2] \leq \mathbb{E}[\|X\|^2]$. Therefore the first term in the above expression can be bounded:

$$\begin{aligned}
\mathbb{E}[\|\nabla\ell(\mathbf{w}, z) - \nabla\ell(\mathbf{v}, z) + \nabla\mathcal{L}(\mathbf{v}) - \nabla\mathcal{L}(\mathbf{w})\|^2] &\leq \mathbb{E}[\|\nabla\ell(\mathbf{w}, z) - \nabla\ell(\mathbf{v}, z)\|^2] \\
&= \mathbb{E}[\|(\nabla\ell(\mathbf{w}, z) - \nabla\ell(\mathbf{w}_\star, z)) - (\nabla\ell(\mathbf{v}, z) - \nabla\ell(\mathbf{w}_\star, z))\|^2] \\
&\leq 2\,\mathbb{E}[\|\nabla\ell(\mathbf{w}, z) - \nabla\ell(\mathbf{w}_\star, z)\|^2 + \|\nabla\ell(\mathbf{v}, z) - \nabla\ell(\mathbf{w}_\star, z)\|^2] \\
&\leq 4H\,\mathbb{E}\left[\ell(\mathbf{w}, z) - \ell(\mathbf{w}_\star, z) + \ell(\mathbf{v}, z) - \ell(\mathbf{w}_\star, z) - \langle\nabla\ell(\mathbf{w}_\star, z), \mathbf{w} + \mathbf{v} - 2\mathbf{w}_\star\rangle\right] \\
&= 4H(\mathcal{L}(\mathbf{w}) - \mathcal{L}(\mathbf{w}_\star) + \mathcal{L}(\mathbf{v}) - \mathcal{L}(\mathbf{w}_\star))
\end{aligned}$$

□

This lemma tells us that the estimate $\mathbf{g}$ has low variance if both $\mathbf{w}$ and $\mathbf{v}$ have small function value. We can use this fact to set up a kind of "virtuous cycle": using estimates $\mathbf{g}$, we run N steps of SGD to obtain iterates with smaller function value. Then, we choose a new checkpoint $\mathbf{v}$ from these iterates and recompute the new $\nabla\mathcal{L}(\mathbf{v})$, and then run N more iterates of SGD. Now, as SGD converges, $\mathbf{w}$ and $\mathbf{v}$ will obtain smaller and smaller objective values, which will make the value of $\mathbb{E}[\|\mathbf{g}\|^2]$ small, which will make SGD convergence even faster, making $\mathbb{E}[\|\mathbf{g}\|^2]$ even smaller, and so on.

The algorithm is presented in Algorithm 19:

Theorem 25.2. *Suppose $\mathcal{L} = \frac{1}{N}\sum_{i=1}^N \ell(\mathbf{w}, z_i)$ is such that each $\ell(\mathbf{w}, z)$ is H-smooth and convex in $\mathbf{w}$ for all z_i. Suppose T is a multiple of N. Suppose $\eta = \frac{c}{\sqrt{N}}$ and η satisfies $\eta \leq \frac{1}{8H}$. Then Algorithm 19 guarantees:*

$$\mathbb{E}\left[\frac{1}{T}\sum_{t=1}^T \mathcal{L}(\mathbf{w}_t) - \mathcal{L}(\mathbf{w}_\star)\right] \leq \frac{\sqrt{N}[8cH(\mathcal{L}(\mathbf{w}_1) - \mathcal{L}(\mathbf{w}_\star)) + \frac{2}{c}\|\mathbf{w}_1 - \mathbf{w}_\star\|^2]}{T}$$

Let's take a minute to interpret this result. First, since Algorithm 19 only performs one $O(N)$ full-gradient computation every N iterations, and normally each iteration takes 2 gradient computations, the total computation used by Algorithm 19 is still $O(T)$, same as it would be for SGD. Therefore, if we use C for the total computation, we need to compare the following numbers:

Algorithm 19 SVRG

Input: Initial Point $\mathbf{w}_1$, learning rate η.
Set initial checkpoint $\mathbf{v}_1 = \mathbf{w}_1$.
Compute $\nabla\mathcal{L}(\mathbf{v}_1)$ (takes $O(N)$ time).
Set $e = 1$ //e is an "epoch" counter.
for $t = 1 \dots T$ **do**
  Sample z_t at random from $\{z_1, \dots, z_N\}$.
  Form $\mathbf{g}_t = \nabla\ell(\mathbf{w}_t, z_t) - \nabla\ell(\mathbf{v}_e, z_t) + \nabla\mathcal{L}(\mathbf{v}_e)$
  Set $\mathbf{w}_{t+1} = \mathbf{w}_t - \eta\mathbf{g}_t$.
  if $t \equiv 0 \mod N$ **then**
    //If we've finished one epoch, pick a new $\mathbf{v}$.
    $e = e + 1$.
    $\mathbf{v}_e = \frac{1}{N}\sum_{i=t-N+1}^{t} \mathbf{w}_i$
    Compute $\nabla\mathcal{L}(\mathbf{v}_e)$ (takes $O(N)$ time).
  end if
end for

1. $O\left(\frac{N}{C}\right)$ for full gradient descent.
2. $O\left(\frac{1}{\sqrt{C}}\right)$ for SGD.
3. $O\left(\frac{\sqrt{N}}{C}\right)$ for SVRG.

Now, it is clear here that SVRG always beats full gradient descent. But what about SGD? Let's consider the case that C is fairly large compared to N, say $O(N^{3/2})$. Then the bounds are $O(1/N^{3/4})$ for SGD compared with $O(1/N)$ for SVRG. Thus, SVRG clearly dominates in this case. However, it's also intuitive that SVRG may not do as well when N is large compared to C. By inspecting the equations, we can see that SVRG will always beat SGD (up to constants) so long as $N \leq C$. This is extremely mild: essentially we are asking that we need to look at every point in the dataset at once. Anything less is essentially throwing out data, so the situation in which SGD is better is a bit of a degenerate case.

Now, let's give the proof of Theorem 25.2

Proof. The proof is similar to our bound for smooth convex losses (after reading the proof, you might want to take a minute to think about why we don't need projections here unlike some of our other convex analyses).

We will use e_t to indicate the current value of e at iteration t. So $e_t = \lceil \frac{t}{N} \rceil$.

$$\begin{aligned}
\|\mathbf{w}_{t+1} - \mathbf{w}_\star\|^2 &= \|\mathbf{w}_t - \eta\mathbf{g}_t - \mathbf{w}_\star\|^2 \\
&= \|\mathbf{w}_t - \mathbf{w}_\star\|^2 - 2\eta\langle \mathbf{g}_t, \mathbf{w}_t - \mathbf{w}_\star\rangle + \eta^2\|\mathbf{g}_t\|^2 \\
\langle \mathbf{g}_t, \mathbf{w}_t - \mathbf{w}_\star\rangle &\leq \frac{\|\mathbf{w}_t - \mathbf{w}_\star\|^2 - \|\mathbf{w}_{t+1} - \mathbf{w}_\star\|^2}{2\eta} + \frac{\eta\|\mathbf{g}_t\|^2}{2} \\
\mathbb{E}[\langle \mathbf{g}_t, \mathbf{w}_t - \mathbf{w}_\star\rangle] &\leq \mathbb{E}\left[\frac{\|\mathbf{w}_t - \mathbf{w}_\star\|^2 - \|\mathbf{w}_{t+1} - \mathbf{w}_\star\|^2}{2\eta} + \frac{\eta\|\mathbf{g}_t\|^2}{2}\right]
\end{aligned}$$

Use the fact that $\mathbb{E}[\mathbf{g}_t] = \nabla\mathcal{L}(\mathbf{w}_t)$:

$$\begin{aligned}
\mathbb{E}[\mathcal{L}(\mathbf{w}_t) - \mathcal{L}(\mathbf{w}_\star)] &\leq \mathbb{E}\left[\frac{\|\mathbf{w}_t - \mathbf{w}_\star\|^2 - \|\mathbf{w}_{t+1} - \mathbf{w}_\star\|^2}{2\eta} + \frac{\eta\|\mathbf{g}_t\|^2}{2}\right] \\
\sum_{t=1}^{T}\mathbb{E}[\mathcal{L}(\mathbf{w}_t) - \mathcal{L}(\mathbf{w}_\star)] &\leq \sum_{t=1}^{T}\mathbb{E}\left[\frac{\|\mathbf{w}_t - \mathbf{w}_\star\|^2 - \|\mathbf{w}_{t+1} - \mathbf{w}_\star\|^2}{2\eta} + \frac{\eta\|\mathbf{g}_t\|^2}{2}\right] \\
&\leq \frac{\|\mathbf{w}_1 - \mathbf{w}_\star\|^2}{2\eta} + \frac{\eta}{2}\sum_{t=1}^{T}\mathbb{E}[\|\mathbf{g}_t\|^2]
\end{aligned}$$

now we use Lemma 25.1:

$$\leq \frac{\eta\|\mathbf{w}_1 - \mathbf{w}_\star\|^2}{2\eta} + \frac{\eta}{2}\sum_{t=1}^{T}\mathbb{E}[8H(\mathcal{L}(\mathbf{w}_t) - \mathcal{L}(\mathbf{w}_\star)) + 4H(\mathcal{L}(\mathbf{v}_{e_t}) - \mathcal{L}(\mathbf{w}_\star))]$$

Using $\eta \leq \frac{1}{8H}$:

$$\leq \frac{\|\mathbf{w}_1 - \mathbf{w}_\star\|^2}{2\eta} + \frac{1}{2}\sum_{t=1}^{T}\mathbb{E}[\mathcal{L}(\mathbf{w}_t) - \mathcal{L}(\mathbf{w}_\star)] + \frac{N}{4}\sum_{e=2}^{T/N}\mathbb{E}[\mathcal{L}(\mathbf{v}_e) - \mathcal{L}(\mathbf{w}_\star)] + 2\eta HN\,\mathbb{E}[\mathcal{L}(\mathbf{v}_1) - \mathcal{L}(\mathbf{w}_\star)]$$

Now, let's take a minute to understand $\mathcal{L}(\mathbf{v}_e) - \mathcal{L}(\mathbf{w}_\star)$. By Jensen inequality, for all $e > 1$, we have:

$$\mathcal{L}(\mathbf{v}_e) \leq \frac{1}{N}\sum_{i=1}^{N}\mathcal{L}(\mathbf{w}_{e(N-1)+1+i})$$

Therefore:

$$\begin{aligned} N\sum_{e=2}^{T/N-1}\mathbb{E}[\mathcal{L}(\mathbf{v}_e) - \mathcal{L}(\mathbf{w}_\star)] &\leq \sum_{e=2}^{T/N-1}\sum_{i=1}^{N}\mathcal{L}(\mathbf{w}_{(e-1)N+i}) - \mathcal{L}(\mathbf{w}_\star) \\ &= \sum_{t=N+1}^{T(N-1)/N}\mathbb{E}[\mathcal{L}(\mathbf{w}_t) - \mathcal{L}(\mathbf{w}_\star)] \\ &\leq \sum_{t=1}^{T}\mathbb{E}[\mathcal{L}(\mathbf{w}_t) - \mathcal{L}(\mathbf{w}_\star)] \end{aligned}$$

Further, for $e = 1$, $\mathbf{v}_e = \mathbf{w}_1$. Therefore, we have:

$$\begin{aligned} \sum_{t=1}^{T}\mathbb{E}[\mathcal{L}(\mathbf{w}_t) - \mathcal{L}(\mathbf{w}_\star)] &\leq \frac{\|\mathbf{w}_1 - \mathbf{w}_\star\|^2}{2\eta} + \frac{3}{4}\sum_{t=1}^{T}\mathbb{E}[\mathcal{L}(\mathbf{w}_t) - \mathcal{L}(\mathbf{w}_\star)] + 2\eta HN\,\mathbb{E}[\mathcal{L}(\mathbf{v}_1) - \mathcal{L}(\mathbf{w}_\star)] \\ \frac{1}{4}\sum_{t=1}^{T}\mathbb{E}[\mathcal{L}(\mathbf{w}_t) - \mathcal{L}(\mathbf{w}_\star)] &\leq \frac{\|\mathbf{w}_1 - \mathbf{w}_\star\|^2}{2\eta} + 2\eta HN\,\mathbb{E}[\mathcal{L}(\mathbf{v}_1) - \mathcal{L}(\mathbf{w}_\star)] \\ \frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[\mathcal{L}(\mathbf{w}_t) - \mathcal{L}(\mathbf{w}_\star)] &\leq \frac{2\|\mathbf{w}_1 - \mathbf{w}_\star\|^2}{\eta T} + \frac{8\eta HN(\mathcal{L}(\mathbf{w}_1) - \mathcal{L}(\mathbf{w}_\star))}{T} \end{aligned}$$

Now finally plugging in $\eta = \frac{c}{\sqrt{N}}$ concludes the result. □

26 Normalized Gradient Descent and Non-convex Variance Reduction

We have just seen how to perform variance reduction for finite-sum convex problems. It turns out that variance reduction is in some sense even more powerful for non-convex problems. In the convex case, we are only able to see gains over SGD in the special case that $\mathcal{L}(\mathbf{w})$ is has the form of a finite sum. In contrast, for non-convex problems we will be able to obtain improved convergence to critical points without this extra restriction. For reference, recall that SGD obtained the guarantee:

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[\|\nabla\mathcal{L}(\mathbf{w}_t)\|^2] \leq O\left(\frac{1}{\sqrt{T}}\right)$$

Thus, by selecting an iterate $\hat{\mathbf{w}}$ at random from $\mathbf{w}_1, \dots, \mathbf{w}_T$, we obtain:

$$\mathbb{E}[\|\nabla\mathcal{L}(\hat{\mathbf{w}})\|] \leq O\left(\frac{1}{T^{1/4}}\right)$$

By using variance reduction, we will be able to significantly improve this to:

$$\mathbb{E}[\|\nabla\mathcal{L}(\hat{\mathbf{w}})\|] \leq O\left(\frac{1}{T^{1/3}}\right)$$

This rate discovered simultaneously by two different groups in 2018 Fang, C. J. Li, et al. 2018; Zhou, Xu, and Gu 2018, and in 2020 this was shown to be the optimal rate Arjevani, Carmon, J. C. Duchi, Foster, Srebro, et al. 2019.

Our presentation of the results will look slightly more similar to Fang, C. J. Li, et al. 2018, but somewhat more streamlined borrowing ideas from Cutkosky and Mehta 2020.

In order to derive the algorithm with a good balance of intuition we will need to consider *normalized* updates for SGD. To start, let's look at the following scheme:

$$\begin{aligned}\mathbf{m}_1 &= \nabla\ell(\mathbf{w}_1, z_1)\\ \mathbf{m}_t &= (1-\alpha)\mathbf{m}_{t-1} + \alpha\nabla\ell(\mathbf{w}_t, z_t)\\ \mathbf{w}_{t+1} &= \mathbf{w}_t - \eta\frac{\mathbf{m}_t}{\|\mathbf{m}_t\|}\end{aligned}$$

We'll call these updates *normalized gradient descent with momentum*.

This is almost identical to our previously studied momentum methods, but now instead of writing $\mathbf{w}_{t+1} = \mathbf{w}_t - \eta\mathbf{m}_t$, we normalized the momentum term in the update. This makes the following important identity true:

$$\|\mathbf{w}_{t+1} - \mathbf{w}_1\| = \eta \text{ for all } t$$

This identity is extremely useful for analysis. Recall that when we previously analyzed momentum in the non-convex setting, we tried to view momentum as a form of averaging, and the primary difficulty was trading off some bias caused by the fact that $\mathbf{w}_t$ is changing over time. Accuratly measuring this bias was very technically challenging because there was a complicated relationship between the speed that $\mathbf{w}_t$ is changing and the amount of bias. In the end, we never actually really quantified how much this bias was, but we were able to sidestep the problem through a tricky use of a potential function. With normalized updates, we are going to be able to completely avoid all of these difficulties.

In particular, we have the following result:

Lemma 26.1. *Suppose that $\mathcal{L}$ is an H-smooth function and $\mathbb{E}[\|\nabla\ell(\mathbf{w}, z) - \nabla\mathcal{L}(\mathbf{w})\|^2] \leq \sigma^2$ for all $\mathbf{w}$. Then using the normalized gradient descent with momentum updates, we have:*

$$\mathbb{E}[\|\mathbf{m}_t - \nabla\mathcal{L}(\mathbf{w}_t)\|] \leq (1-\alpha)^t\sigma + \sigma\sqrt{\alpha} + \frac{H\eta}{\alpha}$$

Proof. Let's start by obtaining an expanded expression for $\mathbf{m}_t$. To compactify the notation, set $\mathbf{g}_t = \nabla\ell(\mathbf{w}_t, z_t)$. Further, let's define:

$$\epsilon_t = \mathbf{m}_t - \nabla\mathcal{L}(\mathbf{w}_t)$$
$$r_t = \mathbf{g}_t - \nabla\mathcal{L}(\mathbf{w}_t)$$

Notice that $\mathbb{E}[r_t] = 0$ and $\mathbb{E}[\|r_t\|^2] \leq \sigma^2$, so that by Jensen inequality, $\mathbb{E}[\|r_t\|] \leq \sigma$.

Then we have:

$$\begin{aligned}
\mathbf{m}_t &= (1-\alpha)\mathbf{m}_{t-1} + \alpha\mathbf{g}_t \\
\epsilon_t &= (1-\alpha)(\mathbf{m}_{t-1} - \nabla\mathcal{L}(\mathbf{w}_t)) + \alpha(\mathbf{g}_t - \nabla\mathcal{L}(\mathbf{w}_t)) \\
&= (1-\alpha)(\mathbf{m}_{t-1} - \nabla\mathcal{L}(\mathbf{w}_{t-1})) + (1-\alpha)(\nabla\mathcal{L}(\mathbf{w}_{t-1}) - \nabla\mathcal{L}(\mathbf{w}_t)) + \alpha r_t \\
&= (1-\alpha)\epsilon_{t-1} + (1-\alpha)(\nabla\mathcal{L}(\mathbf{w}_{t-1}) - \nabla\mathcal{L}(\mathbf{w}_t)) + \alpha r_t
\end{aligned}$$

Now, we have generated a recursive expression for ϵ_t. Notice that the third term, αr_t, is zero in expectation, so we might hope that it has a small contribution to ϵ_t. The second term, is bounded by:

$$\|\nabla\mathcal{L}(\mathbf{w}_{t-1}) - \nabla\mathcal{L}(\mathbf{w}_t)\| \leq H\|\mathbf{w}_{t-1} - \mathbf{w}_t\| = H\eta$$

So we can control it by setting η small. Notice that by using normalized updates, we have a very tight control over the difference of the gradients because we know *exactly* how big $\|\mathbf{w}_{t-1} - \mathbf{w}_t\|$ is.

Let's continue expanding the recursive expression for ϵ_t to see how we can leverage these intuitions:

$$\begin{aligned}
\epsilon_t &= (1-\alpha)\epsilon_{t-1} + (1-\alpha)(\nabla\mathcal{L}(\mathbf{w}_{t-1}) - \nabla\mathcal{L}(\mathbf{w}_t)) + \alpha r_t \\
&= (1-\alpha)^2\epsilon_{t-2} + (1-\alpha)^2(\nabla\mathcal{L}(\mathbf{w}_{t-2}) - \nabla\mathcal{L}(\mathbf{w}_{t-1})) + \alpha(1-\alpha)r_{t_1} + (1-\alpha)(\nabla\mathcal{L}(\mathbf{w}_{t-1}) - \nabla\mathcal{L}(\mathbf{w}_t)) + \alpha r_t
\end{aligned}$$

unrolling for t iterations:

$$= (1-\alpha)^{t-1}\epsilon_1 + \alpha(1-\alpha)^{t-2}r_2 + \cdots + \alpha r_t + (1-\alpha)^{t-1}(\nabla\mathcal{L}(\mathbf{w}_1) - \nabla\mathcal{L}(\mathbf{w}_2)) + \cdots + (1-\alpha)(\nabla\mathcal{L}(\mathbf{w}_{t-1}) - \nabla\mathcal{L}(\mathbf{w}_t))$$

recall that $\mathbf{m}_1 = \mathbf{g}_1$ so that $\epsilon_1 = r_1$:

$$\begin{aligned}
&= (1-\alpha)^{t-1}r_1 + \alpha(1-\alpha)^{t-2}r_2 + \cdots + \alpha(1-\alpha)r_t + \sum_{\tau=1}^{t-1}(1-\alpha)^{t-\tau}(\nabla\mathcal{L}(\mathbf{w}_\tau) - \nabla\mathcal{L}(\mathbf{w}_{\tau+1})) \\
&= (1-\alpha)^t r_1 + \alpha(1-\alpha)^{t-1}r_1 + \cdots + \alpha(1-\alpha)r_t + \sum_{\tau=1}^{t-1}(1-\alpha)^{t-\tau}(\nabla\mathcal{L}(\mathbf{w}_\tau) - \nabla\mathcal{L}(\mathbf{w}_{\tau+1})) \\
&= (1-\alpha)^t r_1 + \alpha\sum_{\tau=1}^{t}(1-\alpha)^{t-\tau}r_\tau + \sum_{\tau=1}^{t-1}(1-\alpha)^{t-\tau}(\nabla\mathcal{L}(\mathbf{w}_\tau) - \nabla\mathcal{L}(\mathbf{w}_{\tau+1}))
\end{aligned}$$

do a little reindexing to make the geometric series in the sums clearer:

$$= (1-\alpha)^t r_1 + \alpha\sum_{\tau=0}^{t}(1-\alpha)^{\tau}r_{t-\tau} + \sum_{\tau=1}^{t-1}(1-\alpha)^{\tau}(\nabla\mathcal{L}(\mathbf{w}_{t-\tau}) - \nabla\mathcal{L}(\mathbf{w}_{t-\tau+1}))$$

Now, observe that all of these terms are expected to be small: the first term is of course geometrically decaying in t,

and the other terms involve goemetric series of $(1-\alpha)$. Let's make this concrete by taking expectations:

$$\begin{aligned}
\mathbb{E}[\|\epsilon_t\|] &\leq (1-\alpha)^t\,\mathbb{E}[\|r_1\|] + \alpha\,\mathbb{E}\left[\left\|\sum_{\tau=0}^{t}(1-\alpha)^\tau r_{t-\tau}\right\|\right] + \sum_{\tau=1}^{t-1}(1-\alpha)^\tau\,\mathbb{E}[\|\nabla\mathcal{L}(\mathbf{w}_{t-\tau}) - \nabla\mathcal{L}(\mathbf{w}_{t-\tau+1})\|] \\
&= (1-\alpha)^t\,\mathbb{E}[\|r_1\|] + \alpha\,\mathbb{E}\left[\left\|\sum_{\tau=0}^{t}(1-\alpha)^\tau r_{t-\tau}\right\|\right] + \sum_{\tau=1}^{t-1}(1-\alpha)^\tau H\eta \\
&\leq (1-\alpha)^t\,\mathbb{E}[\|r_1\|] + \alpha\,\mathbb{E}\left[\left\|\sum_{\tau=0}^{t}(1-\alpha)^\tau r_{t-\tau}\right\|\right] + \sum_{\tau=0}^{\infty}(1-\alpha)^\tau H\eta \\
&= (1-\alpha)^t\,\mathbb{E}[\|r_1\|] + \alpha\,\mathbb{E}\left[\left\|\sum_{\tau=0}^{t}(1-\alpha)^\tau r_{t-\tau}\right\|\right] + \frac{H\eta}{\alpha} \\
&\leq (1-\alpha)^t\sigma + \alpha\,\mathbb{E}\left[\left\|\sum_{\tau=0}^{t}(1-\alpha)^\tau r_{t-\tau}\right\|\right] + \frac{H\eta}{\alpha}
\end{aligned}$$

Now, by Jensen inequality:

$$\begin{aligned}
\mathbb{E}\left[\left\|\sum_{\tau=0}^{t}(1-\alpha)^\tau r_{t-\tau}\right\|\right] &\leq \sqrt{\mathbb{E}\left[\left\|\sum_{\tau=0}^{t}(1-\alpha)^\tau r_{t-\tau}\right\|^2\right]} \\
&\leq \sqrt{\mathbb{E}\left[\sum_{\tau=0}^{t}\sum_{\tau'=0}^{t}(1-\alpha)^{\tau\tau'}\langle r_{t-\tau}, r_{t-\tau'}\rangle\right]}
\end{aligned}$$

since $\mathbb{E}[\langle r_t, r_t'\rangle] = 0$ for $t \neq t'$ and $\mathbb{E}[\|r_t\|^2] \leq \sigma^2$:

$$\begin{aligned}
&\leq \sqrt{\sum_{\tau=0}^{t}(1-\alpha)^{2\tau}\sigma^2} \\
&\leq \sigma\sqrt{\sum_{\tau=0}^{t}(1-\alpha)^\tau} \\
&\leq \sigma\sqrt{\sum_{\tau=0}^{\infty}(1-\alpha)^\tau} \\
&= \frac{\sigma}{\sqrt{\alpha}}
\end{aligned}$$

Thus, $\alpha\,\mathbb{E}\left[\left\|\sum_{\tau=0}^{t}(1-\alpha)^\tau r_{t-\tau}\right\|\right] \leq \sigma\sqrt{\alpha}$. So, putting all this together:

$$\mathbb{E}[\|\epsilon_t\|] \leq (1-\alpha)^t\sigma + \sigma\sqrt{\alpha} + \frac{H\eta}{\alpha}$$

□

This Lemma tells us that, by setting α and η appropriately, we will be able to ensure that $\mathbf{m}_t \approx \nabla\mathcal{L}(\mathbf{w}_t)$ in expectation. Now, it remains to see how we can use this property. To do this, we'll need a variation on the lemma for biased gradient descent we established when analyzing SGD with momentum:

Lemma 26.2. *Define $\epsilon_t = \mathbf{m}_t - \nabla\mathcal{L}(\mathbf{w}_t)$. Then we have:*

$$\mathcal{L}(\mathbf{w}_{t+1}) \leq \mathcal{L}(\mathbf{w}_t) - \frac{\eta}{3}\|\nabla\mathcal{L}(\mathbf{w}_t)\| + \frac{13\eta}{6}\|\epsilon_t\| + \frac{H\eta^2}{2}$$

Proof. By the smoothness property:

$$\begin{aligned}\mathcal{L}(\mathbf{w}_{t+1}) &\leq \mathcal{L}(\mathbf{w}_t) - \eta\langle\nabla\mathcal{L}(\mathbf{w}_t), \frac{\mathbf{m}_t}{\|\mathbf{m}_t\|}\rangle + \frac{H\eta^2}{2} \\ &= \mathcal{L}(\mathbf{w}_t) - \eta\left\langle\nabla\mathcal{L}(\mathbf{w}_t), \frac{\nabla\mathcal{L}(\mathbf{w}_t)+\epsilon_t}{\|\nabla\mathcal{L}(\mathbf{w}_t)+\epsilon_t\|}\right\rangle + \frac{H\eta^2}{2}\end{aligned}$$

Now, let's consider two cases, either $\|\epsilon_t\| \geq \frac{1}{2}\|\nabla\mathcal{L}(\mathbf{w}_t)\|$ or not. If $\|\epsilon_t\| \geq \frac{1}{2}\|\nabla\mathcal{L}(\mathbf{w}_t)\|$, then:

$$\begin{aligned}-\left\langle\nabla\mathcal{L}(\mathbf{w}_t), \frac{\nabla\mathcal{L}(\mathbf{w}_t)+\epsilon_t}{\|\nabla\mathcal{L}(\mathbf{w}_t)+\epsilon_t\|}\right\rangle &\leq \|\nabla\mathcal{L}(\mathbf{w}_t)\| \\ &\leq 2\|\epsilon_t\| \\ &\leq -\frac{\|\nabla\mathcal{L}(\mathbf{w}_t)\|}{3} + \frac{13}{6}\|\epsilon_t\|\end{aligned}$$

Alternatively, if $\|\epsilon_t\| \leq \frac{1}{2}\|\nabla\mathcal{L}(\mathbf{w}_t)\|$:

$$\begin{aligned}\|\nabla\mathcal{L}(\mathbf{w}_t)+\epsilon_t\| &\leq \frac{3}{2}\|\nabla\mathcal{L}(\mathbf{w}_t)\| \\ -\langle\nabla\mathcal{L}(\mathbf{w}_t), \epsilon_t\rangle &\leq \frac{1}{2}\|\nabla\mathcal{L}(\mathbf{w}_t)\|^2 \\ -\left\langle\nabla\mathcal{L}(\mathbf{w}_t), \frac{\nabla\mathcal{L}(\mathbf{w}_t)+\epsilon_t}{\|\nabla\mathcal{L}(\mathbf{w}_t)+\epsilon_t\|}\right\rangle &= -\frac{\|\nabla\mathcal{L}(\mathbf{w}_t)\|^2 + \langle\nabla\mathcal{L}(\mathbf{w}_t), \epsilon_t\rangle}{\|\nabla\mathcal{L}(\mathbf{w}_t)+\epsilon_t\|} \\ &\leq -\frac{\|\nabla\mathcal{L}(\mathbf{w}_t)\|^2/2}{3\|\nabla\mathcal{L}(\mathbf{w}_2)\|/2} \\ &= -\frac{\|\nabla\mathcal{L}(\mathbf{w}_t)\|}{3}\end{aligned}$$

Therefore, either way we have:

$$-\eta\left\langle\nabla\mathcal{L}(\mathbf{w}_t), \frac{\nabla\mathcal{L}(\mathbf{w}_t)+\epsilon_t}{\|\nabla\mathcal{L}(\mathbf{w}_t)+\epsilon_t\|}\right\rangle \leq -\frac{\eta}{3}\|\nabla\mathcal{L}(\mathbf{w}_t)\| + \frac{13\eta}{6}\|\epsilon_t\|$$

from which the result follows. □

Now, we're ready to put everything together and analyze this new version of momentum:

Theorem 26.3. *Define $\Delta = \mathcal{L}(\mathbf{w}_1) - \mathcal{L}(\mathbf{w}_\star)$. Suppose $\mathcal{L}$ is H-smooth and $\mathbf{g}_t$ has variance at most σ^2. Then with $\alpha = \min\left(1, \frac{\sqrt{\Delta H}}{\sigma\sqrt{T}}\right) = O(1/\sqrt{T})$ and $\eta = \frac{\sqrt{\Delta\alpha}}{\sqrt{HT}} = O(1/T^{3/4})$,*

$$\begin{aligned}\mathbb{E}\left[\sum_{t=1}^{T}\|\nabla\mathcal{L}(\mathbf{w}_t)\|\right] &\leq 24\sqrt{\Delta HT} + \frac{35(\Delta HT^3\sigma^2)^{1/4}}{2} + \frac{13\sqrt{T}}{2\sqrt{\Delta H}} \\ &\leq O(T^{3/4})\end{aligned}$$

Proof. Applying Lemma 26.2 followed by Lemma 26.1, we have:

$$\begin{aligned}\mathbb{E}[\mathcal{L}(\mathbf{w}_{t+1})] &\leq \mathbb{E}[\mathcal{L}(\mathbf{w}_t) - \frac{\eta}{3}\|\nabla\mathcal{L}(\mathbf{w}_t)\| + \frac{13\eta}{6}\|\epsilon_t\| + \frac{H\eta^2}{2}] \\ &\leq \mathbb{E}\left[\mathcal{L}(\mathbf{w}_t) - \frac{\eta}{3}\|\nabla\mathcal{L}(\mathbf{w}_t)\| + \frac{H\eta^2}{2} + \frac{13\eta}{6}\left((1-\alpha)^t\sigma + \sigma\sqrt{\alpha} + \frac{H\eta}{\alpha}\right)\right]\end{aligned}$$

telescoping over t:

$$\begin{aligned}
\mathbb{E}[\mathcal{L}(\mathbf{w}_{T+1})] &\leq \mathbb{E}\left[\mathcal{L}(\mathbf{w}_1) - \frac{\eta}{3}\sum_{t=1}^{T}\|\nabla\mathcal{L}(\mathbf{w}_t)\| + \frac{HT\eta^2}{2} + \frac{13\eta}{6}\left(T\sigma\sqrt{\alpha} + \frac{HT\eta}{\alpha} + \sum_{t=1}^{T}(1-\alpha)^t\sigma\right)\right] \\
&\leq \mathbb{E}\left[\mathcal{L}(\mathbf{w}_1) - \frac{\eta}{3}\sum_{t=1}^{T}\|\nabla\mathcal{L}(\mathbf{w}_t)\| + \frac{HT\eta^2}{2} + \frac{13\eta}{6}\left(T\sigma\sqrt{\alpha} + \frac{HT\eta}{\alpha} + \frac{\sigma}{\alpha}\right)\right] \\
&= \mathbb{E}\left[\mathcal{L}(\mathbf{w}_1) - \frac{\eta}{3}\sum_{t=1}^{T}\|\nabla\mathcal{L}(\mathbf{w}_t)\| + \frac{HT\eta^2}{2} + \frac{13\eta T\sigma\sqrt{\alpha}}{6} + \frac{13HT\eta^2}{6\alpha} + \frac{13\eta\sigma}{6\alpha}\right] \\
&\leq \mathbb{E}\left[\mathcal{L}(\mathbf{w}_1) - \frac{\eta}{3}\sum_{t=1}^{T}\|\nabla\mathcal{L}(\mathbf{w}_t)\| + \frac{8HT\eta^2}{3\alpha} + \frac{13\eta T\sigma\sqrt{\alpha}}{6} + \frac{13\eta\sigma}{6\alpha}\right]
\end{aligned}$$

Now let's define $\Delta = \mathcal{L}(\mathbf{w}_1) - \mathcal{L}(\mathbf{w}_\star)$ and rearrange:

$$\mathbb{E}\left[\sum_{t=1}^{T}\|\nabla\mathcal{L}(\mathbf{w}_t)\|\right] \leq \frac{3\Delta}{\eta} + \frac{8HT\eta}{\alpha} + \frac{13T\sigma\sqrt{\alpha}}{2} + \frac{13\sigma}{2\alpha}$$

Now, all that remains is to set α and η appropriately. This is a somewhat tricky task. To start, notice that the optimal value for η should balance the $\frac{3\Delta}{\eta}$ and the $\frac{8HT\eta}{\alpha}$ terms. From this, we can get (ignoring the constant factors) $\eta = \frac{\sqrt{\Delta\alpha}}{\sqrt{HT}}$ so that:

$$\mathbb{E}\left[\sum_{t=1}^{T}\|\nabla\mathcal{L}(\mathbf{w}_t)\|\right] \leq \frac{11\sqrt{\Delta HT}}{\sqrt{\alpha}} + \frac{13T\sigma\sqrt{\alpha}}{2} + \frac{13\sigma}{2\alpha}$$

Now, observe that unless $\alpha \leq \frac{1}{T^{2/3}}$, we should expect the $T\sqrt{\alpha}$ term to be larger than the $1/\alpha$ term. Then, to balance the first and second terms, we can set $\alpha = \frac{\sqrt{\Delta H}}{\sigma\sqrt{T}}$. This would yield:

$$\mathbb{E}\left[\sum_{t=1}^{T}\|\nabla\mathcal{L}(\mathbf{w}_t)\|\right] \leq \frac{35(\Delta HT^3\sigma^2)^{1/4}}{2} + \frac{13\sqrt{T}}{2\sqrt{\Delta H}}$$

However, there is a subtlety: this value of α may not be allowed because we must have $\alpha \leq 1$. If it is not allowed, then $\frac{\sqrt{\Delta H}}{\sigma\sqrt{T}} \geq 1$, so that $\sigma \leq \frac{\sqrt{\Delta H}}{\sqrt{T}}$, and we we set $\alpha = 1$ to obtain:

$$\begin{aligned}
\mathbb{E}\left[\sum_{t=1}^{T}\|\nabla\mathcal{L}(\mathbf{w}_t)\|\right] &\leq 11\sqrt{\Delta HT} + \frac{13T\sigma}{2} + \frac{13\sigma}{2} \\
&\leq 11\sqrt{\Delta HT} + 13\sqrt{\Delta HT} \\
&\leq 24\sqrt{\Delta HT}
\end{aligned}$$

Thus, with $\alpha = \min\left(1, \frac{\sqrt{\Delta H}}{\sigma\sqrt{T}}\right)$, we obtain:

$$\begin{aligned}
\mathbb{E}\left[\sum_{t=1}^{T}\|\nabla\mathcal{L}(\mathbf{w}_t)\|\right] &\leq 24\sqrt{\Delta HT} + \frac{35(\Delta HT^3\sigma^2)^{1/4}}{2} + \frac{13\sqrt{T}}{2\sqrt{\Delta H}} \\
&\leq O(T^{3/4})
\end{aligned}$$

□

Now, this just recovers the standard SGD rate we've seen before. However, it turns out that a small tweak to formula will enable us to get the improved variance-reduction rate without too much extra work in the analysis.

26.1 Adding the Variance Reduction

The variance reduction scheme we will describe now is different than SVRG algorithm we saw earlier: we will not assume that the $\mathcal{L}$ has a finite-sum form, and we will never have to evaluate a full batch (this is good, because if $\mathcal{L}$ is not a finite-sum form it's not even possible to evaluate a full batch!). However, we will make the assumption the $\mathcal{L}(\mathbf{w}) = \mathbb{E}[\ell(\mathbf{w}, z)]$ where $\ell(\mathbf{w}, z)$ is H-smooth in $\mathbf{w}$ for all z.

Now, our new variance-reduced momentum scheme will be the following:

$$\begin{aligned}\mathbf{m}_1 &= \nabla\ell(\mathbf{w}_1, z_1)\\ \mathbf{m}_t &= (1-\alpha)(\mathbf{m}_{t-1} + \nabla\ell(\mathbf{w}_t, z_t) - \nabla\ell(\mathbf{w}_{t-1}, z_t)) + \alpha\nabla\ell(\mathbf{w}_t, z_t)\\ \mathbf{w}_{t+1} &= \mathbf{w}_t - \eta\frac{\mathbf{m}_t}{\|\mathbf{m}_t\|}\end{aligned}$$

Let's call this normalized gradient descent with variance-reduced momentum.

This is almost the same as what we had previously, but now there is an extra $\nabla\ell(\mathbf{w}_t, z_t) - \nabla\ell(\mathbf{w}_{t-1}, z_t)$ added into the momentum update. Intuitively, this term is correcting some bias: since $\mathbf{m}_{t-1}$ is an estimate for the gradient at $\nabla\mathcal{L}(\mathbf{w}_{t-1})$ rather than an $\nabla\mathcal{L}(\mathbf{w}_t)$, we picked up some bias terms when analyzing $\|\epsilon_t\|$ in Lemma 26.1. Since $\mathbb{E}[\nabla\ell(\mathbf{w}_t, z_t) - \nabla\ell(\mathbf{w}_{t-1}, z_t)] = \nabla\mathcal{L}(\mathbf{w}_t) - \nabla\mathcal{L}(\mathbf{w}_{t-1})$, adding this term to the $\mathbf{m}_{t-1}$ is attempting to "de-bias" the momentum to mitigate this effect.

Let's see an analog of Lemma 26.1 for this new update:

Lemma 26.4. *Suppose that $\ell(\mathbf{w}, z)$ is an H-smooth function for all z and $\mathbb{E}[\|\nabla\ell(\mathbf{w}, z) - \nabla\mathcal{L}(\mathbf{w})\|^2] \le \sigma^2$ for all $\mathbf{w}$. Then using the normalized gradient descent with variance-reduced momentum updates, we have:*

$$\mathbb{E}[\|\mathbf{m}_t - \nabla\mathcal{L}(\mathbf{w}_t)\|] \le (1-\alpha)^t\sigma + \sigma\sqrt{\alpha} + \frac{H\eta}{\sqrt{\alpha}}$$

Proof. The proof is extremely similar to Lemma 26.1. Set $\mathbf{g}_t = \nabla\ell(\mathbf{w}_t, z_t)$. Define:

$$\begin{aligned}\epsilon_t &= \mathbf{m}_t - \nabla\mathcal{L}(\mathbf{w}_t)\\ r_t &= \mathbf{g}_t - \nabla\mathcal{L}(\mathbf{w}_t)\\ \delta_t &= \mathbf{w}_t - \mathbf{w}_{t-1}\end{aligned}$$

Now, we have:

$$\begin{aligned}\mathbf{m}_t &= (1-\alpha)(\mathbf{m}_{t-1} + \nabla\ell(\mathbf{w}_t, z_t) - \nabla\ell(\mathbf{w}_{t-1}, z_t)) + \alpha\nabla\ell(\mathbf{w}_t, z_t)\\ \epsilon_t &= (1-\alpha)(\mathbf{m}_{t-1}\nabla\ell(\mathbf{w}_t, z_t) - \nabla\ell(\mathbf{w}_{t-1}, z_t) - \nabla\mathcal{L}(\mathbf{w}_t)) + \alpha(\nabla\ell(\mathbf{w}_t, z_t) - \nabla\mathcal{L}(\mathbf{w}_t))\\ &= (1-\alpha)(\mathbf{m}_{t-1} - \nabla\mathcal{L}(\mathbf{w}_{t-1})) + (1-\alpha)(\nabla\ell(\mathbf{w}_t, z_t) - \nabla\ell(\mathbf{w}_{t-1}, z_t) + \nabla\mathcal{L}(\mathbf{w}_{t-1}) - \nabla\mathcal{L}(\mathbf{w}_t)) + \alpha r_t\\ &= (1-\alpha)\epsilon_{t-1} + (1-\alpha)(\nabla\ell(\mathbf{w}_t, z_t) - \nabla\ell(\mathbf{w}_{t-1}, z_t) + \nabla\mathcal{L}(\mathbf{w}_{t-1}) - \nabla\mathcal{L}(\mathbf{w}_t)) + \alpha r_t\end{aligned}$$

Now, notice the critical difference from the proof of Lemma 26.1: the middle term here is now also zero in expectation! Let's define

$$s_t = \nabla\ell(\mathbf{w}_{t+1}, z_{t+1}) - \nabla\ell(\mathbf{w}_t, z_{t+1}) + \nabla\mathcal{L}(\mathbf{w}_t) - \nabla\mathcal{L}(\mathbf{w}_{t+1})$$

then we have $\mathbb{E}[s_t] = 0$, and

$$\begin{aligned}\mathbb{E}[\|s_t\|^2] &\le \mathbb{E}[\|\nabla\ell(\mathbf{w}_{t+1}, z_{t+1}) - \nabla\ell(\mathbf{w}_t, z_{t+1})\|^2]\\ &\le H^2\|\mathbf{w}_{t+1} - \mathbf{w}_t\|^2\\ &= H^2\eta^2\end{aligned}$$

Now, let's proceed to unroll the recursion once again:

$$\epsilon_t = (1-\alpha)^{t-1}\epsilon_1 + \alpha(1-\alpha)^{t-2}r_2 + \cdots + \alpha r_t + (1-\alpha)^{t-1}s_1 + \cdots + (1-\alpha)s_{t-1}$$

recall that $\mathbf{m}_1 = \mathbf{g}_1$ so that $\epsilon_1 = r_1$:

$$= (1-\alpha)^{t-1} r_1 + \alpha(1-\alpha)^{t-1} r_1 + \cdots + \alpha(1-\alpha) r_t + \sum_{\tau=1}^{t-1} (1-\alpha)^{t-\tau} s_\tau$$

$$= (1-\alpha)^t r_1 + \alpha \sum_{\tau=1}^{t} (1-\alpha)^{t-\tau} r_\tau + \sum_{\tau=1}^{t-1} (1-\alpha)^{t-\tau} s_\tau$$

do a little reindexing to make the geometric series in the sums clearer:

$$= (1-\alpha)^t r_1 + \alpha \sum_{\tau=0}^{t} (1-\alpha)^{\tau} r_{t-\tau} + \sum_{\tau=1}^{t-1} (1-\alpha)^{\tau} s_\tau$$

Now, let's take norms and expectations. The first two terms are bounded identically to in the proof of Lemma 26.1.

$$\mathbb{E}[\|\epsilon_t\|] \leq (1-\alpha)^t \sigma + \sigma\sqrt{\alpha} + \mathbb{E}\left[\left\|\sum_{\tau=1}^{t-1} (1-\alpha)^{\tau} s_\tau\right\|\right]$$

Now, for this last term the argument is again familiar:

$$\mathbb{E}\left[\left\|\sum_{\tau=1}^{t-1} (1-\alpha)^{\tau} s_\tau\right\|\right] \leq \sqrt{\mathbb{E}\left[\left\|\sum_{\tau=1}^{t-1} (1-\alpha)^{\tau} s_\tau\right\|^2\right]}$$

using $\mathbb{E}[s_t] = 0$:

$$\leq \sqrt{\sum_{\tau=1}^{t-1} (1-\alpha)^{2\tau}\, \mathbb{E}[\|s_\tau\|^2]}$$

using $\mathbb{E}[\|s_t\|^2] \leq H^2\eta^2$:

$$\leq H\eta \sqrt{\sum_{\tau=1}^{t-1} (1-\alpha)^{2\tau}}$$

$$\leq \frac{H\eta}{\sqrt{\alpha}}$$

So over all we have obtained:

$$\mathbb{E}[\|\epsilon_t\|] \leq (1-\alpha)^t \sigma + \sigma\sqrt{\alpha} + \frac{H\eta}{\sqrt{\alpha}}$$

□

Compare this result with Lemma 26.1: notice that the $\frac{\eta}{\alpha}$ term has improved to $\frac{\eta}{\sqrt{\alpha}}$.

Now, look back to the proof of Lemma 26.2: this Lemma actually made zero assumptions whatsoever about how $\mathbf{m}_t$ was generated. Thus, it applies equally well with our new improved way to generate $\mathbf{m}_t$ and so we can applying directly analogously to the proof of Theorem 26.3 to show:

Theorem 26.5. *Define $\Delta = \mathcal{L}(\mathbf{w}_1) - \mathcal{L}(\mathbf{w}_\star)$. Suppose $\ell(\mathbf{w}, z)$ is H-smooth for all z and $\nabla\ell(\mathbf{w}, z)$ has variance at most σ^2. Then with $\alpha = 1/T^{2/3}$ and $\eta = \frac{\sqrt{\Delta\sqrt{\alpha}}}{\sqrt{HT}} = O(1/T^{2/3})$,*

$$\mathbb{E}\left[\sum_{t=1}^{T} \|\nabla\mathcal{L}(\mathbf{w}_t)\|\right] \leq 11\sqrt{\Delta H} T^{2/3} + 13\sigma T^{2/3}$$

$$\leq O(T^{2/3})$$

Proof. Applying Lemma 26.2 followed by Lemma 26.1, we have:

$$\begin{aligned}\mathbb{E}[\mathcal{L}(\mathbf{w}_{t+1})] &\leq \mathbb{E}[\mathcal{L}(\mathbf{w}_t) - \frac{\eta}{3}\|\nabla\mathcal{L}(\mathbf{w}_t)\| + \frac{13\eta}{6}\|\epsilon_t\| + \frac{H\eta^2}{2}] \\ &\leq \mathbb{E}\left[\mathcal{L}(\mathbf{w}_t) - \frac{\eta}{3}\|\nabla\mathcal{L}(\mathbf{w}_t)\| + \frac{H\eta^2}{2} + \frac{13\eta}{6}\left((1-\alpha)^t\sigma + \sigma\sqrt{\alpha} + \frac{H\eta}{\sqrt{\alpha}}\right)\right]\end{aligned}$$

telescoping over t:

$$\begin{aligned}\mathbb{E}[\mathcal{L}(\mathbf{w}_{T+1})] &\leq \mathbb{E}\left[\mathcal{L}(\mathbf{w}_1) - \frac{\eta}{3}\sum_{t=1}^{T}\|\nabla\mathcal{L}(\mathbf{w}_t)\| + \frac{HT\eta^2}{2} + \frac{13\eta}{6}\left(T\sigma\sqrt{\alpha} + \frac{HT\eta}{\sqrt{\alpha}} + \sum_{t=1}^{T}(1-\alpha)^t\sigma\right)\right] \\ &\leq \mathbb{E}\left[\mathcal{L}(\mathbf{w}_1) - \frac{\eta}{3}\sum_{t=1}^{T}\|\nabla\mathcal{L}(\mathbf{w}_t)\| + \frac{HT\eta^2}{2} + \frac{13\eta}{6}\left(T\sigma\sqrt{\alpha} + \frac{HT\eta}{\sqrt{\alpha}} + \frac{\sigma}{\alpha}\right)\right] \\ &= \mathbb{E}\left[\mathcal{L}(\mathbf{w}_1) - \frac{\eta}{3}\sum_{t=1}^{T}\|\nabla\mathcal{L}(\mathbf{w}_t)\| + \frac{HT\eta^2}{2} + \frac{13\eta T\sigma\sqrt{\alpha}}{6} + \frac{13HT\eta^2}{6\sqrt{\alpha}} + \frac{13\eta\sigma}{6\alpha}\right] \\ &\leq \mathbb{E}\left[\mathcal{L}(\mathbf{w}_1) - \frac{\eta}{3}\sum_{t=1}^{T}\|\nabla\mathcal{L}(\mathbf{w}_t)\| + \frac{8HT\eta^2}{3\sqrt{\alpha}} + \frac{13\eta T\sigma\sqrt{\alpha}}{6} + \frac{13\eta\sigma}{6\alpha}\right]\end{aligned}$$

Rearranging:

$$\mathbb{E}\left[\sum_{t=1}^{T}\|\nabla\mathcal{L}(\mathbf{w}_t)\|\right] \leq \frac{3\Delta}{\eta} + \frac{8HT\eta}{\sqrt{\alpha}} + \frac{13T\sigma\sqrt{\alpha}}{2} + \frac{13\sigma}{2\alpha}$$

Now, again we need only to choose the values for η and α. Balancing the first two terms with $\eta = \frac{\sqrt{\Delta\sqrt{\alpha}}}{\sqrt{HT}}$ yields:

$$\mathbb{E}\left[\sum_{t=1}^{T}\|\nabla\mathcal{L}(\mathbf{w}_t)\|\right] \leq 11\frac{\sqrt{\Delta HT}}{\alpha^{1/4}} + \frac{13T\sigma\sqrt{\alpha}}{2} + \frac{13\sigma}{2\alpha}$$

Now, set $\alpha = \frac{1}{T^{2/3}}$ to obtain:

$$\mathbb{E}\left[\sum_{t=1}^{T}\|\nabla\mathcal{L}(\mathbf{w}_t)\|\right] \leq 11\sqrt{\Delta H}T^{2/3} + 13\sigma T^{2/3}$$

If you want to be a little more careful, we can set $\alpha = \min\left(1, \frac{(\Delta H)^{2/3}}{\sigma^{4/3}T^{2/3}}\right)$. Then, if $\alpha = 1$ we have $\sigma \leq$

$$\mathbb{E}\left[\sum_{t=1}^{T}\|\nabla\mathcal{L}(\mathbf{w}_t)\|\right] \leq 34\sqrt{\Delta HT}$$

while otherwise we have:

$$\mathbb{E}\left[\sum_{t=1}^{T}\|\nabla\mathcal{L}(\mathbf{w}_t)\|\right] \leq \frac{35(\Delta H\sigma)^{1/3}T^{2/3}}{2} + \frac{13\sigma^{7/3}T^{2/3}}{2(\Delta H)^{2/3}}$$

□

26.2 Other Uses of Normalized SGD in Practice

In the preceding sections, we primarily introduce normalization into th SGD update for analytical convenience: normalization ensures that $\|\mathbf{w}_t - \mathbf{w}_{t-1}\| = \eta$ for all t, which makes calculations much simpler. In fact, it is possible to prove essentially similar results without normalization.

However, normalization can also be motivated by essentially purely empirical considerations. You, Gitman, and Ginsburg 2017; You, J. Li, et al. 2019 suggest the LARS and LAMB algorithms that incorporate normalization in concert with specific "layer-wise learning rate". To describe this proposal, we need to break the full parameter $\mathbf{w}$ into a small number of "sub-parameters" corresponding to conceptual units in whatever model $\mathbf{w}$ specifies. For example, consider the model:

$$\mathbf{x} \mapsto W_2\sigma(W_1\sigma(W_0\mathbf{x}))$$

where σ is the ReLU activation: $\sigma : \mathbb{R}^n \to \mathbb{R}^n$ via $\sigma(z)[i] = \max(0, z[i])$ and W_0, W_1 and W_2 are all matrices. Then we can consider $\mathbf{w}$ as the flattened vector of all three matrices concatenated together. Most of our analysis would then discard the original structure of the matrices. However, conceptually it makes sense to remember this structure. Thus, we could re-write the standard SGD update as:

$$(W_i)_{t+1} = (W_i)_t - \eta\nabla_{W_i}\ell(\mathbf{w}, \zeta)$$

We then move to a normalized update and allow η to depend on the matrix W_i, to obtain the update:

$$(W_i)_{t+1} = (W_i)_t - \eta\|(W_i)_t\|\frac{\nabla_{W_i}\ell(\mathbf{w}, \zeta)}{\|\nabla_{W_i}\ell(\mathbf{w}, \zeta)\|}$$

This is essentially the LARS update. Now, at this point the update is identical to a "per-matrix" version of normalized gradient descent (without momentum), with the exception of scaling the learning rate by $\|M_i\|$. So far as I am aware, there is no strong theoretical justification for this modification. Nevertheless, You, Gitman, and Ginsburg 2017 showed that this approach yielded extremely good empirical performance when combined with very large batch sizes.

Now, we will describe the LAMB update. First, add momentum:

$$(M_i)_{t+1} = \beta(M_i)_t + (1-\beta)\nabla_{W_i}\ell(\mathbf{w}_t, \zeta_t)$$
$$(W_i)_{t+1} = (W_i)_t - \eta\|(W_i)_t\|\frac{(M_i)_t)}{\|(M_i)_t\|}$$

Now, we give the update an "Adam"-flavor (all addition and squaring below is per-coordinate):

$$(M_i)_{t+1} = \beta(M_i)_t + (1-\beta)\nabla_{W_i}\ell(\mathbf{w}_t, \zeta_t)$$
$$(V_i)_{t+1} = beta(V_i)_t + (1-\beta)\nabla_{W_i}\ell(\mathbf{w}_t, \zeta_t)^2$$
$$(D_i)_{t+1} = \frac{(M_i)_{t+1}}{\sqrt{(V_i)_{t+1}}}$$
$$(W_i)_{t+1} = (W_i)_t - \eta\|(W_i)_t\|\frac{(D_i)_{t+1})}{\|(D_i)_{t+1}\|}$$

This update attempts to normalize the Adam displacement rather than the just a running-average of the gradients. Again, there is no known theoretical justification for this, but it seems to provide strong empirical performance, and was originally reported to significantly improve upon other optimizers like Adam when used with very large batch sizes.

A word of caution: Although LARS and LAMB have become rather popular, recent results suggest that much of their proposed gain of Adam at large batch sizes may be due to poor tuning of standard baselines like Adam Nado et al. 2021. This is a common theme in empirical optimization algorithm work: it is very hard to disentangle improvements that are actually due to conceptual improvements in the optimization algorithm and improvements that are simply due to poorly tuned baselines.

27 Second-Order Smoothness and Cubic Regularization

For most of this course we have consider *smooth* objectives, for which the second derivative $\nabla^2\mathcal{L}(\mathbf{w})$ satisfies the bound $\|\nabla^2\mathcal{L}(\mathbf{w})\|_{\text{op}} \leq H$ for some H. Now, we will consider restricting the class of functions we consider a little more, to see if we can make some improved algorithms. Specifically, let us consider functions for which not just the second derivative, but also the *third* derivative is bounded. Such functions are callled *second-order smooth*:

Definition 27.1. *A function $\mathcal{L} : \mathbb{R}^d \to \mathbb{R}$ is ρ-second-order smooth if $\mathcal{L}$ is twice-differentiable at all $x \in \mathbb{R}^d$, and for all x, y, v:*

$$\|(\nabla^2\mathcal{L}(x) - \nabla^2\mathcal{L}(y))v\| \leq \rho\|x - y\|\|v\|$$

Note that in the literature, the symbol for second-order smoothness is often ρ rather than J. This definition is related to the third derivative (or "jerk" as it is called in physics) in a directly analogous way to how Lipschitzness is related to the first derivative and ordinary smoothness is related to the second derivative. Note that in the literature, the symbol for second-order smoothness is often ρ rather than J. We name it J for "jerk" to help remember the symbol, and also because when writing it is often easy to get a ρ confused with a p.

In order to formalize the relationship between second-order smoothness and the third derivative, we need to think a little bit about what sort of object the third derivative actually is. The third derivative of a function $\mathcal{L} : \mathbb{R}^d \to \mathbb{R}$ can be specified by a three dimensional $d \times d \times d$ matrix whose ijk entry is $\frac{\partial^2\mathcal{L}(\mathbf{w})}{\partial\mathbf{w}[i]\partial\mathbf{w}[j]\partial\mathbf{w}[k]}$. Formally, in the same way that a (2-d) matrix M represents a bilinear function taking vectors v, w to the scalar $v^\top Mw$, a 3-d matrix T represents a trilinear function taking vectors x, y, z to a scalar via the operation:

$$(x, y, z) \mapsto \sum_{i=1}^{d}\sum_{j=1}^{d}\sum_{k=1}^{d} T[i, j, k]x[i]y[j]z[k]$$

If we denote the third derivative of a function $\mathcal{L}$ at a point $\mathbf{w}$ by $\nabla^3\mathcal{L}(\mathbf{w})$ and use $\nabla^3\mathcal{L}(\mathbf{w})(x, y, z)$ to indicate application of the above trilinear form to vectors x, y, z, then we can write the third-order Taylor approximation to $\mathcal{L}$ as:

$$\mathcal{L}(\mathbf{w} + \delta) \approx \mathcal{L}(\mathbf{w}) + \langle\nabla\mathcal{L}(\mathbf{w}), \delta\rangle + \frac{\delta^\top\nabla^2\mathcal{L}(\mathbf{w})\delta}{2} + \frac{\nabla^2\mathcal{L}(\mathbf{w})(\delta, \delta, \delta)}{6}$$

Also, note that just as a 2-d matrix transforms vectors into other vectors, the 3-d matrices transform vectors into matrices. That is, since $\nabla^3\mathcal{L}(\mathbf{w})$ is the third derivative, we can use it to write a first-order Taylor expansion for $\nabla^2\mathcal{L}$:

$$\nabla^2\mathcal{L}(\mathbf{w} + \delta)[i, j] \approx \nabla^2\mathcal{L}(\mathbf{w})[i, j] + \sum_{k=1}^{d}\nabla^3\mathcal{L}(\mathbf{w})[i, j, k]\delta[k]$$

Mathematically, the objects $\nabla\mathcal{L}(\mathbf{w})$, $\nabla^2\mathcal{L}(\mathbf{w})$ and $\nabla^3\mathcal{L}(\mathbf{w})$ are sometimes said to be first, second, and third-order *tensors* respectively. Since it is somewhat cumbersome to continually say 2-d matrix vs 3-d matrix, we will call the first derivative a tensor, the second derivative a matrix, and the first derivative a vector.

We can extend the definition of operator norm from matrices to tensors as follows:

Definition 27.2. *The* operator norm *of a tensor T is denoted by $\|T\|_{op}$ and defined by:*

$$\sup_{\|x\|,\|y\|,\|z\|\leq 1} |T(x, y, z)|$$

Notice that the trilinearity of a tensor T then implies:

$$|T(x, y, z)| \leq \|x\|\|y\|\|z\|\|T\|_{\text{op}}$$

With these preliminaries out of the way, we can provide the formal connection between second-order smoothness and the third derivative:

Proposition 27.3. *Suppose $\mathcal{L}$ is thrice differentiable. Then $\mathcal{L}$ is J-second-order smooth if $\|\nabla^3\mathcal{L}(\mathbf{w})\|_{op} \leq J$ for all $\mathbf{w}$.*

Proof. Let x and y be arbitrary points. Then by the mean value theorem, there is some z such that:

$$\nabla^2 \mathcal{L}(y)[i,j] = \nabla^2 \mathcal{L}(x)[i,j] + \sum_{k=1}^{d} \nabla^3 \mathcal{L}(z)[i,j,k](x-y)[k]$$

Therefore, for any unit vectors v and w:

$$\begin{aligned} v^\top (\nabla^2 \mathcal{L}(y) - \nabla^2 \mathcal{L}(x)) w &= \nabla^3 \mathcal{L}(z)(v, w, (x-y)) \\ &\leq \|v\| \|w\| \|x-y\| \|\nabla^3 \mathcal{L}(z)\|_{\text{op}} \\ &\leq J\|x-y\| \end{aligned}$$

where the last line uses the boundedness of $\nabla^3 \mathcal{L}(z)$ and that v and w are unit-vectors. But $\sup_{\|v\|=1, \|w\|=1} v^\top (\nabla^2 \mathcal{L}(y) - \nabla^2 \mathcal{L}(x)) w = \|\nabla^2 \mathcal{L}(y) - \nabla^2 \mathcal{L}(x)\|_{\text{op}}$, so this implies $\|\nabla^2 \mathcal{L}(y) - \nabla^2 \mathcal{L}(x)\|_{\text{op}} \leq J\|x-y\|$ and $\mathcal{L}$ is J-second-order smooth. □

Next, we can provide an analog of the "key smoothness lemma":

Lemma 27.4. *Suppose $\mathcal{L}$ is J-second-order smooth. Then for any x and δ:*

$$\mathcal{L}(x+\delta) \leq \mathcal{L}(x) + \langle \nabla \mathcal{L}(x), \delta \rangle + \frac{\delta^\top \nabla^2 \mathcal{L}(x) \delta}{2} + \frac{J\|\delta\|^3}{6}$$

Proof. From fundamental theorem of calculus (used twice):

$$\begin{aligned} \mathcal{L}(x+\delta) &= \mathcal{L}(x) + \int_0^1 \langle \nabla \mathcal{L}(x + t\delta), \delta \rangle \, dt \\ &= \mathcal{L}(x) + \int_0^1 \int_0^1 \langle \nabla \mathcal{L}(x), \delta \rangle + t\delta^\top \nabla^2 \mathcal{L}(x + tk\delta)\delta \, dk dt \\ &= \mathcal{L}(x) + \langle \nabla \mathcal{L}(x), \delta \rangle + \int_0^1 \int_0^1 t\delta^\top \nabla^2 \mathcal{L}(x)\delta + t\delta^\top (\nabla^2 \mathcal{L}(x + tk\delta) - \nabla^2 \mathcal{L}(x))\delta \, dk dt \\ &= \mathcal{L}(x) + \langle \nabla \mathcal{L}(x), \delta \rangle + \frac{\delta^\top \nabla^2 \mathcal{L}(x)\delta}{2} + \int_0^1 \int_0^1 t\delta^\top (\nabla^2 \mathcal{L}(x + tk\delta) - \nabla^2 \mathcal{L}(x))\delta \, dk dt \\ &\leq \mathcal{L}(x) + \langle \nabla \mathcal{L}(x), \delta \rangle + \frac{\delta^\top \nabla^2 \mathcal{L}(x)\delta}{2} + \int_0^1 \int_0^1 t\|\delta\|^2 \|\nabla^2 \mathcal{L}(x + tk\delta) - \nabla^2 \mathcal{L}(x)\|_{\text{op}} \, dk dt \\ &\leq \mathcal{L}(x) + \langle \nabla \mathcal{L}(x), \delta \rangle + \frac{\delta^\top \nabla^2 \mathcal{L}(x)\delta}{2} + \int_0^1 \int_0^1 t^2 k J \|\delta\|^3 \, dk dt \\ &= \mathcal{L}(x) + \langle \nabla \mathcal{L}(x), \delta \rangle + \frac{\delta^\top \nabla^2 \mathcal{L}(x)\delta}{2} + \frac{J\|\delta\|^3}{6} \end{aligned}$$

□

We also have the following bound on the change in the gradients, which is also an intuitive consequence of Taylor series ideas:

Lemma 27.5. *Suppose $\mathcal{L}$ is J-second order smooth. Then for any $\mathbf{w}$ and δ:*

$$\|\nabla \mathcal{L}(\mathbf{w} + \delta) - (\nabla \mathcal{L}(\mathbf{w}) + \nabla^2 \mathcal{L}(\mathbf{w})\delta)\| \leq \frac{J\|\delta\|^2}{2}$$

Proof. By the fundamental theorem of calculus:

$$\begin{aligned}
\nabla\mathcal{L}(\mathbf{w}+\delta) &= \nabla\mathcal{L}(\mathbf{w}) + \int_0^1 \nabla^2\mathcal{L}(\mathbf{w}+t\delta)\delta\, dt \\
&= \nabla\mathcal{L}(\mathbf{w}) + \nabla^2\mathcal{L}(\mathbf{w})\delta + \int_0^1 (\nabla^2\mathcal{L}(\mathbf{w}+t\delta) - \nabla^2\mathcal{L}(\mathbf{w}))\delta\, dt \\
\|\nabla\mathcal{L}(\mathbf{w}+\delta) - (\nabla\mathcal{L}(\mathbf{w}) + \nabla^2\mathcal{L}(\mathbf{w})\delta)\| &\le \left\| \int_0^1 (\nabla^2\mathcal{L}(\mathbf{w}+t\delta) - \nabla^2\mathcal{L}(\mathbf{w}))\delta\, dt \right\| \\
&\le \int_0^1 \|(\nabla^2\mathcal{L}(\mathbf{w}+t\delta) - \nabla^2\mathcal{L}(\mathbf{w}))\delta\|\, dt \\
&\le \int_0^1 \|\nabla^2\mathcal{L}(\mathbf{w}+t\delta) - \nabla^2\mathcal{L}(\mathbf{w})\|_{\text{op}}\|\delta\|\, dt \\
&\le \int_0^1 Jt\|\delta\|^2 \\
&\le \frac{J\|\delta\|^2}{2}
\end{aligned}$$

□

What can we do with second-order smooth functions? With regular smooth functions, we were trying to find critical points where the gradient is small. With second-order smooth functions, we can do much better. Not only will we be able to find critical points faster, we will be able to find *second-order stationary points*, otherwise known as (approximate) *local minima*.

Definition 27.6. *A point* $\mathbf{w}$ *is an* (α, β)*-second order stationary point of a function* $\mathcal{L}$ *if:*

$$\|\nabla\mathcal{L}(\mathbf{w})\| \le \alpha$$
$$\lambda_{\min}(\nabla^2\mathcal{L}(\mathbf{w})) \ge -\beta$$

where $\lambda_{\min}(M)$ *indicates the smallest eigenvalue of a matrix* M*.*

Intuitively, if $\beta = 0$ in the above, that means that the hessian is positive semi-definite at this point so that if the gradient is also small, we must be at a local minimum. Thus, allowing a small positive β is a way to relax the notion of local minimum.

One way to try to minimize second-order smooth functions is using the *cubic-regularized newton step* Nesterov and Polyak 2006. This algorithm makes the update:

$$\mathbf{w}_{t+1} = \operatorname*{argmin}_{\mathbf{w}} \mathcal{L}(\mathbf{w}_t) + \langle\mathcal{L}(\mathbf{w}_t), \mathbf{w} - \mathbf{w}_t\rangle + \frac{(\mathbf{w}-\mathbf{w}_t)^\top\nabla^2\mathcal{L}(\mathbf{w}_t)(\mathbf{w}-\mathbf{w}_t)}{2} + \frac{J\|\mathbf{w}-\mathbf{w}_t\|^3}{6}$$

Now, actually solving for the value of $\mathbf{w}_{t+1}$ here is itself a non-convex optimization problem. However, it turns out that since it has this very special cubic form, it is possible to reformulate it into a way that can be solved efficiently. However, let's not worry about that now and instead get some idea for how this update will perform.

Previously we've exploited the fact that a large gradient value means that it is possible to make the function value decrease a lot. Here, we will exploit a different fact: if $\nabla^2\mathcal{L}(\mathbf{w})$ has an eigenvector with large negative eigenvalue, then it is also possible to make the function value decrease a lot. Let's see how. Suppose $\nabla^2\mathcal{L}(\mathbf{w})\mathbf{v} = -\lambda\mathbf{v}$ for some $\lambda > 0$ and unit vector $\mathbf{v}$. Notice that we also have $\nabla^2\mathcal{L}(\mathbf{w})(-\mathbf{v}) = -\lambda(-\mathbf{v})$. Therefore, after possibly replacing $\mathbf{v}$ with $-\mathbf{v}$, we may as well assume that $\langle\mathbf{v}, \nabla\mathcal{L}(\mathbf{w})\rangle \le 0$. Let's consider $\mathbf{w} + \eta\mathbf{v}$ for some to-be-specified η. Then we have:

$$\mathcal{L}(\mathbf{w}+\eta\mathbf{v}) \le \mathcal{L}(\mathbf{w}) + \langle\nabla\mathcal{L}(\mathbf{w}), \eta\mathbf{v}\rangle + \frac{\eta^2\mathbf{v}^\top\nabla^2\mathcal{L}(\mathbf{w})\mathbf{v}}{2} + \frac{J\eta^3\|\mathbf{v}\|^3}{6}$$

using $\langle\nabla\mathcal{L}(\mathbf{w}), \mathbf{v}\rangle \le 0$, $\nabla^2\mathcal{L}(\mathbf{w})\mathbf{v} = -\lambda\mathbf{v}$ and $\|\mathbf{v}\| = 1$:

$$\le \mathcal{L}(\mathbf{w}) - \frac{\eta^2\lambda}{2} + \frac{J\eta^3}{6}$$

Now, set $\eta = \frac{2\lambda}{J}$:

$$\leq \mathcal{L}(\mathbf{w}) - \frac{2\lambda^3}{3J^2}$$

Now, this is similar to how we had a function decrease of $O(-\|\nabla\mathcal{L}(\mathbf{w})\|^2/H)$, but now the decrease is in terms of the eigenvalue of $\nabla^2\mathcal{L}(\mathbf{w})$ and is cubic rather than quadratic.

Thus, we have the following result:

Theorem 27.7. *Suppose $\mathcal{L}$ is J-second-order smooth, and for some given $\mathbf{w}$ there is a unit vector $\mathbf{v}$ such that $\mathbf{v}^\top\nabla^2\mathcal{L}(\mathbf{w})\mathbf{v} < -\lambda$ for some $\lambda \geq 0$. Let $\delta = -\frac{2\lambda}{J}\mathbf{v}sign(\langle\mathbf{v}, \nabla\mathcal{L}(\mathbf{w})\rangle)$, where sign$(x)$ is 1 if $x \geq 0$ and -1 otherwise. Then:*

$$\mathcal{L}(\mathbf{w}+\delta) \leq \mathcal{L}(\mathbf{w}) + \langle\nabla\mathcal{L}(\mathbf{w}), \delta\rangle + \frac{\delta^\top\nabla^2\mathcal{L}(\mathbf{w})\delta}{2} + \frac{\|\delta\|^3 J}{6} \leq \mathcal{L}(\mathbf{w}) - \frac{2\lambda^3}{3J^2}$$

Proof. Notice by definition of δ we have $\langle\nabla\mathcal{L}(\mathbf{w}), \delta\rangle \leq 0$. Further, $\delta^\top\nabla^2\mathcal{L}(\mathbf{w})\delta < -\frac{4\lambda^3}{J^2}$ and $\|\delta\|^3 = \frac{8\lambda^3}{J^3}$. Therefore

$$\begin{aligned}\mathcal{L}(\mathbf{w}+\delta) &\leq \mathcal{L}(\mathbf{w}) + \langle\nabla\mathcal{L}(\mathbf{w}), \delta\rangle + \frac{\delta^\top\nabla^2\mathcal{L}(\mathbf{w})\delta}{2} + \frac{\|\delta\|^3 J}{6} \\ &< \mathcal{L}(\mathbf{w}) - \frac{2\lambda^3}{3J^2}\end{aligned}$$

□

Now, let's define

$$\delta_t^\star = \operatorname*{argmin}_\delta \langle\nabla\mathcal{L}(\mathbf{w}_t), \delta\rangle + \frac{\delta^\top\nabla^2\mathcal{L}(\mathbf{w})\delta}{2} + \frac{\|\delta\|^3 J}{6}$$

and so the cubic regularized newton step algorithm is:

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \delta_t^\star$$

Let's also define the "progress" function:

$$P_t(\delta) = \langle\nabla\mathcal{L}(\mathbf{w}_t), \delta\rangle + \frac{\delta^\top\nabla^2\mathcal{L}(\mathbf{w})\delta}{2} + \frac{\|\delta\|^3 J}{6}$$

so that

$$\mathcal{L}(\mathbf{w}_{t+1}) \leq \mathcal{L}(\mathbf{w}) + P_t(\delta_t^\star)$$

and $\delta_t^\star = \operatorname{argmin} P_t(\delta)$. Now, by Theorem 27.7, we have just seen that if the smallest eigenvalue of $\nabla^2\mathcal{L}(\mathbf{w})$ is denoted by $-\lambda(\mathbf{w})$, then if $\lambda(\mathbf{w}) \geq 0$, we have the bound:

$$P_t(\delta_t^\star) \leq -\frac{2\lambda(\mathbf{w})^3}{3J^2}$$

Our overall goal to analyze the cubic-regularized newton step algorithm is roughly to show that, at each iteration for any ϵ, either we have found a point with both $\|\nabla\mathcal{L}(\mathbf{w})\| \leq \epsilon$ and $\lambda(\mathbf{w}) \leq \sqrt{\epsilon}$, or $\mathcal{L}(\mathbf{w})$ will decrease by at least $\epsilon^{3/2}$. Therefore, since we can only decrease by $\epsilon^{3/2}$ at most $\frac{\mathcal{L}(\mathbf{w}_1)-\mathcal{L}(\mathbf{w}_\star)}{\epsilon^{3/2}}$ times, we see that by setting $\epsilon = O(1/T^{2/3})$, there must be some iteration at which we do not decrease the function value be $\epsilon^{3/2}$, and so we find an $(\epsilon, \sqrt{\epsilon})$-second-order stationary point.

Theorem 27.8. *After T steps of cubic-regularized newton step, there must exist some $t \in \{1, \ldots, T\}$ such that:*

$$\begin{aligned}\|\nabla\mathcal{L}(\mathbf{w}_{t+1})\| &\leq \frac{6^{2/3}\Delta^{2/3}J^{4/3}}{T^{2/3}} \\ \lambda(\mathbf{w}_{t+1}) &\leq 2\frac{6^{1/3}\Delta^{1/3}J^{2/3}}{T^{1/3}}\end{aligned}$$

That is, $\mathbf{w}_{t+1}$ is a $\left(\frac{6^{2/3}\Delta^{2/3}J^{4/3}}{T^{2/3}}, 2\frac{6^{1/3}\Delta^{1/3}J^{2/3}}{T^{1/3}}\right)$ second-order stationary point.

Proof. Let us define $\Delta = \mathcal{L}(\mathbf{w}_1) - \mathcal{L}(\mathbf{w}_\star)$ and $\epsilon = \frac{6^{2/3}\Delta^{2/3}J^{4/3}}{T^{2/3}}$. Now, let's consider some index $t \in \{1, \dots, T\}$ and do some casework.

First, suppose $\lambda(\mathbf{w}_t) > \sqrt{\epsilon}$. Then by Theorem 27.7, we have:

$$\mathcal{L}(\mathbf{w}_{t+1}) < \mathcal{L}(\mathbf{w}_t) - \frac{2\epsilon^{3/2}}{3J^2}$$

Next, let's suppose $\lambda(\mathbf{w}_t) \leq \sqrt{\epsilon}$. Now, notice that since $\delta_t^\star = \operatorname{argmin} P_t(\delta)$, we have $\nabla P_t(\delta_t^\star) = 0$. Therefore:

$$\nabla\mathcal{L}(\mathbf{w}_t) + \nabla^2\mathcal{L}(\mathbf{w}_t)\delta_t^\star + \frac{\delta_t^\star\|\delta_t^\star\|J}{2} = 0 \tag{11}$$

Further, since $\mathbf{w}_{t+1} = \mathbf{w}_t + \delta_t^\star$, by Lemma 27.5, there is some vector $\mathbf{x}$ with $\|\mathbf{x}\| \leq \frac{J\|\delta_t^\star\|^2}{2}$ such that:

$$\nabla\mathcal{L}(\mathbf{w}_t) + \nabla^2\mathcal{L}(\mathbf{w}_t)\delta_t^\star = \nabla\mathcal{L}(\mathbf{w}_{t+1}) + \mathbf{x}$$

Therefore:

$$\nabla\mathcal{L}(\mathbf{w}_{t+1}) + \mathbf{x} + \frac{\delta_t^\star\|\delta_t^\star\|J}{2} = 0$$

which in turn implies:

$$\|\nabla\mathcal{L}(\mathbf{w}_{t+1})\| \leq \|\mathbf{x}\| + \frac{\|\delta_t^\star\|^2 J}{2} \leq J\|\delta_t^\star\|^2$$

Now, let's suppose that $\|\delta_t^\star\| \leq \frac{\sqrt{\epsilon}}{J}$. Then we have

$$\|\nabla\mathcal{L}(\mathbf{w}_{t+1})\| \leq \epsilon$$

And further, since $\lambda(\mathbf{w}) \leq \sqrt{\epsilon}$, we have for all unit vectors $\mathbf{v}$:

$$\begin{aligned}
\mathbf{v}^\top\nabla^2\mathcal{L}(\mathbf{w}_{t+1})\mathbf{v} &= \mathbf{v}^\top(\mathcal{L}(\mathbf{w}_{t+1}) - \mathcal{L}(\mathbf{w}_t))\mathbf{v} + \mathbf{v}^\top\mathcal{L}(\mathbf{w}_t)\mathbf{v} \\
&\geq -J\|\mathbf{w}_{t+1} - \mathbf{w}_t\| + \mathbf{v}^\top\mathcal{L}(\mathbf{w}_t)\mathbf{v} \\
&\geq -J\|\delta_t^\star\| - \sqrt{\epsilon} \\
&\geq -2\sqrt{\epsilon}
\end{aligned}$$

so that $\lambda(\mathbf{w}_{t+1}) \leq 2\sqrt{\epsilon}$ and so $\mathbf{w}_{t+1}$ is an $(\epsilon, 2\sqrt{\epsilon})$ second order stationary point.

Now, suppose instead that $\|\delta_t^\star\| > \frac{\sqrt{\epsilon}}{J}$. Then, taking the inner product of equation (11) with $\delta_t^\star$ on both sides, we have:

$$\begin{aligned}
\langle\nabla\mathcal{L}(\mathbf{w}_t), \delta_t^\star\rangle + \delta_t^{\star\top}\nabla^2\mathcal{L}(\mathbf{w}_t)\delta_t^\star + \frac{\|\delta_t^\star\|^3 J}{2} &= 0 \\
P_t(\delta_t^\star) + \frac{\delta_t^{\star\top}\nabla^2\mathcal{L}(\mathbf{w})\delta_t^\star}{2} + \frac{\|\delta_t^\star\|^3 J}{3} &= 0 \\
P_t(\delta_t^\star) &= -\frac{\delta_t^{\star\top}\nabla^2\mathcal{L}(\mathbf{w})\delta_t^\star}{2} - \frac{\|\delta_t^\star\|^3 J}{3} \\
&\leq \frac{\lambda(\mathbf{w})\|\delta_t^\star\|^2}{2} - \frac{\|\delta_t^\star\|^3 J}{3} \\
&\leq \frac{\sqrt{\epsilon}\|\delta_t^\star\|^2}{2} - \frac{\|\delta_t^\star\|^3 J}{3} \\
&< -\frac{\epsilon^{3/2}}{6J^2}
\end{aligned}$$

In summary we have the following possibilities:

1. $\mathbf{w}_{t+1}$ is an $(\epsilon, 2\sqrt{\epsilon})$ second-order stationary point.

2. $\lambda(\mathbf{w}_t) > \sqrt{\epsilon}$ and $\mathcal{L}(\mathbf{w}_{t+1}) - \mathcal{L}(\mathbf{w}_t) < -\frac{2\epsilon^{3/2}}{3J}$.

3. $\lambda(\mathbf{w}_t) \leq \sqrt{\epsilon}$ and $\mathcal{L}(\mathbf{w}_{t+1}) - \mathcal{L}(\mathbf{w}_t) < -\frac{\epsilon^{3/2}}{6J^2}$.

We can compactify these cases a little bit by noticing that the conclusion of the third case is strictly weaker than the conclusion of the second case:

1. $\mathbf{w}_{t+1}$ is an $(\epsilon, 2\sqrt{\epsilon})$ second-order stationary point.

2. If case 1 does not occur, then $\mathcal{L}(\mathbf{w}_{t+1}) - \mathcal{L}(\mathbf{w}_t) < -\frac{\epsilon^{3/2}}{6J^2}$.

Suppose case 1 never occurs. Then by definition of ϵ, this would imply:

$$-\Delta \leq \mathcal{L}(\mathbf{w}_{T+1}) - \mathcal{L}(\mathbf{w}_1) = \sum_{t=1}^{T} \mathcal{L}(\mathbf{w}_{t+1}) - \mathcal{L}(\mathbf{w}_t) < -\frac{T\epsilon^{3/2}}{6J^2} = -\Delta$$

so that $-\Delta < -\Delta$, which is a contradiction. Therefore there must be at least one iteration in which case 1 happens, which implies the desired result. □

There has been a fair amount of recent work on second-order smooth optimization. In the deterministic setting, see Carmon et al. 2017 for an algorithm that does not require the ability to compute the entire hessian in order to achieve faster convergence rates (it's also a nice application of the almost-convex optimization algorithm we saw earlier). In the stochastic setting, see Fang, Lin, and Zhang 2019 for a proof that SGD actually converges faster on second-order smooth functions than the analysis we provided previously in lecture, or Cutkosky and Mehta 2020 for a slightly more complicated algorithm but much simpler proof of a similar result. The lower bounds in this setting are not as well understood. In the case of stochastic methods, the results of Fang, Lin, and Zhang 2019; Cutkosky and Mehta 2020 are known to be tight, as recently shown by Arjevani, Carmon, J. C. Duchi, Foster, Sekhari, et al. 2020. However, in the deterministic setting there is a bit of a gap between the lower bounds and the best known algorithms. See Carmon et al. 2019 and Carmon et al. 2021 for a description of the lower bounds currently known in this case.

References

Nguyen, Lam M et al. (2020). "A Unified Convergence Analysis for Shuffling-Type Gradient Methods". In: *arXiv preprint arXiv:2002.08246*.

Mishchenko, Konstantin, Ahmed Khaled, and Peter Richtárik (2020). "Random Reshuffling: Simple Analysis with Vast Improvements". In: *arXiv preprint arXiv:2006.05988*.

Rumelhart, David E, Geoffrey E Hinton, and Ronald J Williams (1986). "Learning representations by back-propagating errors". In: *nature* 323.6088, pp. 533–536.

Baydin, Atilim Gunes et al. (2018). "Automatic differentiation in machine learning: a survey". In: *Journal of machine learning research* 18.

Tu, Loring W.. (2011). *An introduction to manifolds*. Springer.

Loshchilov, Ilya and Frank Hutter (2016). "Sgdr: Stochastic gradient descent with warm restarts". In: *arXiv preprint arXiv:1608.03983*.

Goyal, Priya et al. (2017). "Accurate, large minibatch sgd: Training imagenet in 1 hour". In: *arXiv preprint arXiv:1706.02677*.

Hazan, Elad, Alexander Rakhlin, and Peter L Bartlett (2008). "Adaptive online gradient descent". In: *Advances in Neural Information Processing Systems*, pp. 65–72.

Duchi, J., E. Hazan, and Y. Singer (2010). "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization". In: *Conference on Learning Theory (COLT)*.

Erven, Tim van and Wouter M Koolen (2016). "MetaGrad: Multiple Learning Rates in Online Learning". In: *Advances in Neural Information Processing Systems 29*. Ed. by D. D. Lee et al. Curran Associates, Inc., pp. 3666–3674.

Cutkosky, Ashok and Francesco Orabona (2018). "Black-Box Reductions for Parameter-free Online Learning in Banach Spaces". In: *Conference On Learning Theory*, pp. 1493–1529.

Li, Xiaoyu and Francesco Orabona (2019). "On the convergence of stochastic gradient descent with adaptive stepsizes". In: *The 22nd International Conference on Artificial Intelligence and Statistics*. PMLR, pp. 983–992.

Ward, Rachel, Xiaoxia Wu, and Leon Bottou (2019). "AdaGrad stepsizes: Sharp convergence over nonconvex landscapes". In: *International Conference on Machine Learning*. PMLR, pp. 6677–6686.
Reddi, Sashank J., Satyen Kale, and Sanjiv Kumar (2018). "On the Convergence of Adam and Beyond". In: *6th International Conference on Learning Representations, ICLR 2018*.
Bubeck, Sébastien et al. (2015). "Convex Optimization: Algorithms and Complexity". In: *Foundations and Trends® in Machine Learning* 8.3-4, pp. 231–357.
Nesterov, Yurii (n.d.). "Introductory Lectures on Convex Optimization A Basic Course". In: ().
Allen-Zhu, Zeyuan and Lorenzo Orecchia (2017). "Linear Coupling: An Ultimate Unification of Gradient and Mirror Descent". In: *8th Innovations in Theoretical Computer Science Conference (ITCS 2017)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.
Arjevani, Yossi, Yair Carmon, John C Duchi, Dylan J Foster, Nathan Srebro, et al. (2019). "Lower bounds for non-convex stochastic optimization". In: *arXiv preprint arXiv:1912.02365*.
Carmon, Yair et al. (2019). "Lower bounds for finding stationary points i". In: *Mathematical Programming*, pp. 1–50.
— (2021). "Lower bounds for finding stationary points II: first-order methods". In: *Mathematical Programming* 185.1-2.
McMahan, H. Brendan and Matthew Streeter (2010). "Adaptive Bound Optimization for Online Convex Optimization". In: *Proceedings of the 23rd Annual Conference on Learning Theory (COLT)*.
Kingma, Diederik and Jimmy Ba (2014). "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980*.
Niu, Feng et al. (2011). "HOGWILD! a lock-free approach to parallelizing stochastic gradient descent". In: *Proceedings of the 24th International Conference on Neural Information Processing Systems*, pp. 693–701.
Johnson, Rie and Tong Zhang (2013). "Accelerating stochastic gradient descent using predictive variance reduction". In: *Advances in neural information processing systems*, pp. 315–323.
Roux, Nicolas Le, Mark Schmidt, and Francis Bach (2012). "A stochastic gradient method with an exponential convergence rate for finite training sets". In: *Proceedings of the 25th International Conference on Neural Information Processing Systems-Volume 2*, pp. 2663–2671.
Shalev-Shwartz, Shai and Tong Zhang (2013). "Stochastic Dual Coordinate Ascent Methods for Regularized Loss Minimization." In: *Journal of Machine Learning Research* 14.2.
Fang, Cong, Chris Junchi Li, et al. (2018). "Spider: Near-optimal non-convex optimization via stochastic path-integrated differential estimator". In: *Advances in Neural Information Processing Systems*, pp. 689–699.
Zhou, Dongruo, Pan Xu, and Quanquan Gu (2018). "Stochastic nested variance reduction for nonconvex optimization". In: *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pp. 3925–3936.
Cutkosky, Ashok and Harsh Mehta (2020). "Momentum Improves Normalized SGD". In: *International Conference on Machine Learning*.
You, Yang, Igor Gitman, and Boris Ginsburg (2017). "Scaling sgd batch size to 32k for imagenet training". In: *arXiv preprint arXiv:1708.03888* 6, p. 12.
You, Yang, Jing Li, et al. (2019). "Large batch optimization for deep learning: Training bert in 76 minutes". In: *arXiv preprint arXiv:1904.00962*.
Nado, Zachary et al. (2021). "A large batch optimizer reality check: Traditional, generic optimizers suffice across batch sizes". In: *arXiv preprint arXiv:2102.06356*.
Nesterov, Yurii and Boris T Polyak (2006). "Cubic regularization of Newton method and its global performance". In: *Mathematical Programming* 108.1, pp. 177–205.
Carmon, Yair et al. (2017). ""Convex Until Proven Guilty": Dimension-Free Acceleration of Gradient Descent on Non-Convex Functions". In: *International Conference on Machine Learning*. PMLR, pp. 654–663.
Fang, Cong, Zhouchen Lin, and Tong Zhang (2019). "Sharp Analysis for Nonconvex SGD Escaping from Saddle Points". In: *Conference on Learning Theory*, pp. 1192–1234.
Arjevani, Yossi, Yair Carmon, John C Duchi, Dylan J Foster, Ayush Sekhari, et al. (2020). "Second-Order Information in Non-Convex Stochastic Optimization: Power and Limitations". In: *Conference on Learning Theory*, pp. 242–299.

A big-O notation

Big-O notation is a shorthand commonly used in computer science and engineering to simplify complicated equations by focusing on high-level asymptotic rates. Formally, two functions $f(x)$ and $g(x)$ satisfy $f \in O(g)$ if there is are

constants C and X such that for all $x \geq X$ x, $f(x) \leq Cg(x)$. That is,

$$O(g) = \{f | \exists C, X \text{ such that } \forall x \geq X, f(x) \leq Cg(x)$$

In these notes, we frequently use the "$\leq$" symbol rather than the "$\in$" symbol to emphasize the role of big-O as an upper-bound, so we would write $f \leq O(g)$.

Some examples:

1. $1 + x \leq O(x)$.
2. $ax^n + bx^k \leq O(x^k)$ for any constants a and b, n and k with $k \geq n$.
3. $\log(x) + x \leq O(x)$.

Big-O notation also satisfies the following general rules:

1. If $f \leq O(g)$ and $h \leq O(k)$, $fh \leq O(gk)$.
2. If $f \leq O(g)$, then $cf \leq O(g)$ for all constant c.
3. If $f \leq O(g)$ and $h \leq O(k)$, then $f + h \leq O(\max(g, k))$.
4. If $f \leq O(g)$ and $g \leq O(k)$, then $f \leq O(k)$.
5. If $f \leq O(f)$ for all f.
6. If f is bounded away from zero (i.e. there is some ϵ such that $f(x) \geq \epsilon$ for all sufficiently large x), then $1 \leq O(f)$.
7. $f \leq O(1)$ implies that there is a constant C such that $f(x) \leq C$ for all x.

Frequently, we will use big-O notation to hide dependencies on certain values. For example, we might say that $1 + GT^2 \leq O(T^2)$. In this case, we are attempting to draw attention to the way a particular quantity varies *with* T, and assuming G to be constant.

B Probability and Math Background

Proposition B.1 (Markov Inequality)**.** *Suppose X is a random variable such that $X \geq 0$ with probability 1 and $\mathbb{E}[X] < \infty$. Then for all $y > 0$:*

$$P[X \geq y] \leq \frac{\mathbb{E}[X]}{y}$$

Proof. Notice that $P[X \geq y] = \mathbb{E}[\mathbb{1}[X \geq y]]$, where $\mathbb{1}[X \geq y]$ is 1 when $X \geq y$ and 0 otherwise. Further, since $X \geq 0$, $\mathbb{1}[X \geq y] \leq \frac{X}{y}$. Therefore, $P[X \geq y] = \mathbb{E}[\mathbb{1}[X \geq y]] \leq \mathbb{E}[X/y] = \frac{\mathbb{E}[X]}{y}$. □

Proposition B.2 (Chebychev Inequality)**.** *Suppose X is a random variable with variance $\sigma^2 = \mathbb{E}[(X - \mathbb{E}[X])^2]$. Then for all y:*

$$P[|X - \mathbb{E}[X]| \geq y] \leq \frac{\sigma^2}{y^2}$$

Proof. Define $Z = (X - \mathbb{E}[X])^2$. Observe that $Z \geq 0$ with probability 1, and $\mathbb{E}[Z] = \sigma^2$. Further, $|X - \mathbb{E}[X]| \geq y$ if and only if $Z \geq y^2$. The result now follows from Markov inequality. □

Proposition B.3 (Jensen Inequality)**.** *Suppose X is a random variable and f is a convex function. Then*

$$\mathbb{E}[f(X)] \geq f(\mathbb{E}[X])$$

Proof. Since f is convex, there exists some g such that for all y,

$$f(y) \geq f(\mathbb{E}[X]) + \langle g, y - \mathbb{E}[X] \rangle$$

(for example, if f is differentiable, $g = \nabla f(\mathbb{E}[X])$). Then

$$\mathbb{E}[f(X)] \geq f(\mathbb{E}[X]) + \mathbb{E}[\langle g, X - \mathbb{E}[X] \rangle] = f(\mathbb{E}[X])$$

□

Jensen inequality is often used in reverse: if f is a *concave* function (that is, $-f$ is convex), then $\mathbb{E}[f(x)] \leq f(\mathbb{E}[X])$. This can be used, for example, to show that for any random variable X, $\mathbb{E}[|X|] \leq \sqrt{\mathbb{E}[X^2]}$.

Also, the following *discrete* statement of Jensen inequality is often useful (note that this is really the same statement, just reformulated a bit):

Corollary B.4 (Jensen inequality in discrete case)**.** *Suppose $x_1, \ldots, x_N \in \mathbb{R}^d$ and $a_1, \ldots, a_N$ are non-negative real numbers. Define $\overline{x} = \frac{1}{\sum_{i=1}^N a_i} \sum_{i=1}^N x_i$. Then if f is a convex function:*

$$\frac{1}{\sum_{i=1}^N a_i} \sum_{i=1}^N f(x_i) \geq f(\overline{x})$$

Proof. Consider the random variable X with a discrete distribution that takes on value x_i with probability $\frac{a_i}{\sum_{j=1}^N a_j}$. Notice that $\mathbb{E}[X] = \overline{x}$. The result then follows directly from Jensen inequality. □

Proposition B.5 (Union Bound)**.** *Suppose A and B are two events. Then $P[A \text{ or } B] \leq P[A] + P[B]$.*

Proof. Clearly the event represented by A or B can be decomposed into either A or B and not A. The probability of B and not A is at most the probability of B, so the result follows. □

Next, we will often make use of the *bias-variance decomposition*:

Proposition 10.4. *[Bias-variance Decomposition] Suppose $X \in \mathbb{R}^d$ is some random variable. Then for any deterministic value Y:*

$$\mathbb{E}[\|X - Y\|^2] = \|Y - \mathbb{E}[X]\|^2 + \mathbb{E}[\|X - \mathbb{E}[X]\|^2]$$

Proof. Expanding the square we have:

$$\begin{aligned}
\mathbb{E}[\|X - Y\|^2] &= \mathbb{E}[\|X - \mathbb{E}[X] + \mathbb{E}[X] - Y\|^2] \\
&= \mathbb{E}[\|Y - \mathbb{E}[X]\|^2] + \mathbb{E}[\|X - \mathbb{E}[X]\|^2] + 2\,\mathbb{E}[\langle X - \mathbb{E}[X], \mathbb{E}[X] - Y \rangle] \\
&= \|Y - \mathbb{E}[X]\|^2 + \mathbb{E}[\|X - \mathbb{E}[X]\|^2] + 2\langle \mathbb{E}[X - \mathbb{E}[X]], \mathbb{E}[X] - Y \rangle \\
&= \|Y - \mathbb{E}[X]\|^2 + \mathbb{E}[\|X - \mathbb{E}[X]\|^2]
\end{aligned}$$

where □

Another useful inequality is the Young inequality:

Proposition 10.5. *[Young inequality] Suppose X and Y are arbitrary vectors. Then for any $\lambda > 0$ and any non-negative real numbers p and q satisfying $\frac{1}{p} + \frac{1}{q} = 1$,*

$$\langle X, Y \rangle \leq \frac{\|X\|^p}{p\lambda^p} + \frac{\lambda^q \|Y\|^q}{q}$$

Proof. Notice that we must have both p and q at least 1 (otherwise $1/p + 1/q > 1$, which is not allowed). Now, differentiating the RHS with respect to λ yields:

$$-\frac{\|X\|^p}{\lambda^{p+1}} + \lambda^{q-1}\|Y\|^q$$

Differentiating again yields:

$$(p+1)\frac{\|X\|^p}{\lambda^{p+2}} + (q-1)\lambda^{q-2}\|Y\|^q \geq 0$$

where we have used $q \geq 1$. Thus, we can find a minimum value for the RHS by setting the first derivative to 0. Solving $-\frac{\|X\|^p}{\lambda^{p+1}} + \lambda^{q-1}\|Y\|^q = 0$ for λ then gives $\lambda = \frac{\|X\|^{p/(p+q)}}{\|Y\|^{q/(p+q)}}$, which yields a minimum value of

$$\frac{\|X\|^{p-\frac{p^2}{p+q}}\|Y\|^{\frac{pq}{p+q}}}{p} + \frac{\|Y\|^{q-\frac{q^2}{p+q}}\|X\|^{\frac{pq}{p+q}}}{q} = \frac{\|X\|^{\frac{pq}{p+q}}\|Y\|^{\frac{pq}{p+q}}}{p} + \frac{\|Y\|^{\frac{pq}{p+q}}\|X\|^{\frac{pq}{p+q}}}{q}$$

Now, notice that

$$1 = \frac{1}{p} + \frac{1}{q} = \frac{p+q}{pq}$$

So that the minimum of the RHS is in fact:

$$\frac{\|X\|\|Y\|}{p} + \frac{\|Y\|\|X\|}{q} = \|X\|\|Y\|$$

Now to conclude observe that $\langle X, Y\rangle \leq \|X\|\|Y\|$ by Cauchy-Schwarz. □

Finally, we will also often need the following generalization of Cauchy-Schwarz:

Proposition 10.6. *[Cauchy-Schwarz for random variables] Suppose $A \in \mathbb{R}$ and $B \in \mathbb{R}$ are arbitrary random variables. Then:*

$$\mathbb{E}[AB] \leq \sqrt{\mathbb{E}[A^2]\,\mathbb{E}[B^2]}$$

Proof. By Young inequality (Proposition 10.5), we have

$$\begin{aligned}\mathbb{E}[AB] &\leq \mathbb{E}\left[\frac{A^2}{2\lambda^2} + \frac{\lambda^2 B^2}{2}\right] \\ &= \frac{\mathbb{E}[A^2]}{2\lambda^2} + \frac{\lambda^2\,\mathbb{E}[B^2]}{2}\end{aligned}$$

for any constant $\lambda > 0$. Now, set $\lambda = \frac{\mathbb{E}[A^2]}{\mathbb{E}[B^2]}$ to conclude the desired result. □

C Convexity, Smoothness and Lipschitzness

Many classic algorithms work for *convex* functions:

Definition 2.1. *A function $\mathcal{L}$ is* convex *if for all x, there exists some g_x such that for all y:*

$$\mathcal{L}(y) \geq \mathcal{L}(x) + \langle g_x, y - x\rangle$$

When $\mathcal{L}$ is differentiable, $g_x = \nabla\mathcal{L}(x)$.

When $\mathcal{L}$ is twice-differentiable, the following is an equivalent condition for convexity:

Proposition C.1. *Suppose $\mathcal{L}$ is twice-differentiable. Then $\mathcal{L}$ is convex if and only if $\nabla^2\mathcal{L}(\mathbf{w}) \succeq 0$ for all $\mathbf{w}$.*

Next, we will usually assume that loss functions $\mathcal{L}$ iare *Lipschitz*:

Definition 2.2. *A function $\mathcal{L} : \mathbb{R}^d \to \mathbb{R}$ is G-Lipschitz if for all $x, y \in \mathbb{R}^d$,*

$$|\mathcal{L}(x) - \mathcal{L}(y)| \leq G\|x - y\|$$

This definition may be a little strange looking, but it's actually fairly likely to hold. The intuition here is that the G in G-Lipschitzness is actually measuring the degree to which $\mathcal{L}$ is continuous. In fact, if W is a compact set, then any continuous $\mathcal{L}$ must be G-Lipschitz for some G. When $\mathcal{L}$ is differentiable, we can phrase G-Lipschitzness in the following way:

Proposition 2.3. *If $\mathcal{L}$ is differentiable, then $\mathcal{L}$ is G-Lipschitz if and only if $\|\nabla\mathcal{L}(x)\| \leq G$ for all x.*

Proof. First, suppose $\|\nabla\mathcal{L}(x)\| \geq G$ for some $x \in \mathbb{R}^d$. By definition of gradient for any vector v, we have:

$$\lim_{\delta\to 0} \frac{\mathcal{L}(x + \delta v) - \mathcal{L}(x) - \delta\langle v, \nabla\mathcal{L}(x)\rangle}{\delta} = 0$$

Let $v = \frac{\nabla\mathcal{L}(x)}{\|\nabla\mathcal{L}(x)\|}$. Then this implies:

$$\lim_{\delta\to 0} \frac{\mathcal{L}(x + \delta v) - \mathcal{L}(x)}{\delta} = \|\nabla\mathcal{L}(x)\|$$

Thus, if $\|\nabla\mathcal{L}(x)\| \geq G$, there must be some δ such that

$$|\mathcal{L}(x + \delta v) - \mathcal{L}(x)| \geq G\delta$$

and so $\mathcal{L}$ is no G-Lipschitz. Therefore G-Lipschitzness implies $\|\nabla\mathcal{L}(x)\| \leq G$. Now suppose $\|\nabla\mathcal{L}(x)\| \leq G$ for all x. Then by the fundamental theorem of calculus,

$$\begin{aligned}
\mathcal{L}(x) - \mathcal{L}(y) &= \int_0^1 \frac{d}{dt}\mathcal{L}(y + t(x - y))\, dt \\
&= \int_0^1 \langle\nabla\mathcal{L}(y + t(x - y)), x - y\rangle\, dt \\
&\leq \int_0^1 \|\nabla\mathcal{L}(y + t(x - y))\|\|x - y\|\, dt \\
&\leq G\|x - y\|
\end{aligned}$$

so that $\mathcal{L}$ is G-Lipschitz. □

For most of this course, we will consider exclusively losses $\mathcal{L}$ that are *smooth*:

Definition 4.1. *A function $\mathcal{L} : \mathbb{R}^d \to \mathbb{R}$ is H-smooth if $\mathcal{L}$ is differentiable at all $x \in \mathbb{R}^d$, and for all $x, y \in W$,*

$$\|\nabla\mathcal{L}(x) - \nabla\mathcal{L}(y)\| \leq H\|x - y\|$$

Let's be clear: there are plenty of losses out there that are not smooth, or are only smooth with a very large value of H. However, without *some* kind of assumption about the loss it is very difficult to prove anything about how an algorithm will operate. Similarly to the way that G-Lipschitzness is the same as the gradient being bounded by G when $\mathcal{L}$ is differentiable, H-smoothness is the same as the Hessian being bounded by H when $\mathcal{L}$ is twice-differentiable:

Proposition 4.2. *Suppose $\mathcal{L}$ is twice differentiable. Then $\mathcal{L}$ is H-smooth if and only if $\|\nabla^2\mathcal{L}(\mathbf{w})\|_{op} \leq H$ for all $\mathbf{w}$.*

The proof is essentially the same as the proof of Proposition 2.3, although with two integrals instead of one. We will leave it as an exercise.

Smoothness is at least approximately satisfied by essentially any continuous function, in the sense that any contuous function can be replaced with a approximation that is smooth. The quality of the approximation can be traded off for the amount of smoothness. Formally, the following theorem holds:

Theorem 4.3. *Let $\mathcal{L} : \mathbb{R}^d \to \mathbb{R}$ be a G-Lipschitz function. Then for any $\gamma > 0$, consider*

$$\hat{\mathcal{L}}(x) = \mathop{\mathbb{E}}_{v \sim N(0,I)}[L(x + \gamma v)]$$

where $v \sim N(0, I)$ indicates that v is sampled from Gaussian distribution. $\hat{\mathcal{L}}$ is G-Lipschitz and G/γ-smooth and for all $x \in W$,

$$|\hat{\mathcal{L}}(x) - \mathcal{L}(x)| \leq \gamma\sqrt{d}$$

To extend the idea of smoothness, we have the notion of *second-order smoothness*:

Definition 27.1. *A function $\mathcal{L} : \mathbb{R}^d \to \mathbb{R}$ is ρ-second-order smooth if $\mathcal{L}$ is twice-differentiable at all $x \in \mathbb{R}^d$, and for all x, y, v:*

$$\|(\nabla^2\mathcal{L}(x) - \nabla^2\mathcal{L}(y))v\| \leq \rho\|x - y\|\|v\|$$

Just as Lipschitzness relates to the first derivative and smoothness to the second derivative, second-order smoothness means that the *third* derivative is bounded. The third derivative of a function $f : \mathbb{R}^d \to \mathbb{R}$ is a bit of a complicated object. Formally, it is a tensor in $\mathbb{R}^{d\times d\times d}$. Concretely, it can be represented by a three-dimensional matrix whose ijk entry is $\frac{\partial f}{\partial x_i \partial x_j \partial x_k}$. This 3-dimensional matrix turns vectors into regular 2-dimensional matrices: given a 3-dimensional matrix T and a vector v, we write the 2-dimensional matrix Tv as:

$$Tv[i, j] = \sum_k T[i, j, k]v[k]$$

We say $\|T\|_{\text{op}} = \sup_{\|v\|\leq 1} \|Tv\|_{\text{op}}$. With this notation, a thrice-differentiable function is ρ-second-order smooth if and only if its third derivative has operator norm at most ρ.

C.1 Critical points and Local minima

Because we will not be assuming that $\mathcal{L}$ is convex, in general we will not be able to actually show that our algorithms in fact minimizes $\mathcal{L}$. Instead, we will often simply try to approach a *critical point*:

Definition 4.4. *A* critical point*, or* first-order stationary point *of $\mathcal{L}$ is a point x such that $\nabla\mathcal{L}(x) = 0$.*

The search for critical points is motivated by the observation that any minimizer of $\mathcal{L}$ must also be a critical point, so that finding a critical point is a necessary but not sufficient condition for actually minimizing $\mathcal{L}$. There is some active research investigating the degree to which this condition may actually be sufficient. In fact, one of the key desirable properties of convex functions is that any critical point must also minimize $\mathcal{L}$. Sometimes we will be more ambitious and instead try to find a *local minimum*:

Definition C.2. *A* local minimum*, or a* second-order stationary point *of $\mathcal{L}$ is a point x such that for some ϵ and all y with $\|y - x\| \leq \epsilon$, $\mathcal{L}(y) \geq \mathcal{L}(x)$.*

As is typical in optimization, instead of actually identifying critical points or local minima, we will identify approximate critical points or local minima. The notion of an approximate critical point is relatively straightforward: we simply try to make $\|\nabla\mathcal{L}(x)\|$ as small as possible:

Definition 4.5. *A ϵ-approximate critical point of $\mathcal{L}$ is a point x such that $\|\nabla\mathcal{L}(x)\| \leq \epsilon$*

The appropriate way to define an approximate local minimum is less clear however. To think about this, we will need to use some information about the second derivative. One particular idea that is important is the notion of being *positive semidefinite*:

Definition C.3. *A square matrix $M \in \mathbb{R}^{d\times d}$ is positive semidefinite if for all $v \in \mathbb{R}^d$, $v^\top Mv \geq 0$. Further, for any matrices A and B, we use the notation $A \succeq B$ to indicate that $v^\top Av \geq v^\top Bv$ for all v. Then, M is positive semidefinite if and only if $M \succeq 0$.*

We will adopt the following standard definition:

Definition C.4. *Suppose $\mathcal{L}$ is twice-differentiable. Then x is an (ϵ, δ)-approximate local minimum if $\|\nabla \mathcal{L}(x)\| \leq \epsilon$ and $\nabla^2 \mathcal{L}(x) \succeq -\delta I$. Frequently, we will consider the case $\delta = \sqrt{\epsilon}$.*

Let's get some intuition behind why we impose the condition on $\nabla^2 \mathcal{L}(x)$. First, observe that if $\mathcal{L}$ is twice differentiable, then at any local minimum we must have $\nabla^2 \mathcal{L}(x) \succeq 0$. To see this intuitively, observe that by a second-order Taylor expansion, for all v very close to 0 we have:

$$\mathcal{L}(x + v) \approx \mathcal{L}(x) + \langle \nabla \mathcal{L}(x), v \rangle + \frac{v^\top \nabla^2 \mathcal{L}(x) v}{2}$$

We have that $\nabla \mathcal{L}(x) = 0$ at a local minimum, so:

$$= \mathcal{L}(x) + \frac{v^\top \nabla^2 \mathcal{L}(x) v}{2}$$

Now, since we are at a local minimum, $\mathcal{L}(y) \geq \mathcal{L}(x)$ so that we must have $v^\top \nabla^2 \mathcal{L}(x) v \geq 0$ for all v in some neighborhood of 0, which implies that $\nabla^2 \mathcal{L}(x) \succeq 0$. Therefore, the δ part of Definition C.4 is measuring the degree to which this inequality is violated.