# Convex Optimization Course Notes

Instructor: Yuanzhi Li
Notes by Yan Pan*
Carnegie Mellon University

There lecture notes were written for the course 10-725 *Convex Optimization* (*a.k.a. Optimization for Machine Learning/Non-Convex Optimization*) at Carnegie Mellon University in Spring 2021.

## Contents



*Email: ypan2@andrew.cmu.edu

# 1 Introduction

We begin with some basic definitions of convexity.

**Definition 1.1** (Convex Set)**.** A set $\mathcal{D}$ is *convex* if for every $x, y \in \mathcal{D}$ and $0 \leq \lambda \leq 1$, we have

$$(1-\lambda)x + \lambda y \in \mathcal{D}.$$

**Definition 1.2** (Convex Function)**.** A function $f$ is *convex* over a convex set $\mathcal{D}$ if for every $x, y \in \mathcal{D}$ and $0 \leq \lambda \leq 1$, we have

$$f\left((1-\lambda)x + \lambda y\right) \leq (1-\lambda)f(x) + \lambda f(y).$$

**Remark 1.2.1.** A convex function does not need to be differentiable, such as $f(x) = |x|$.

**Remark 1.2.2** (Alternative Definition of Convexity)**.** A second order differentiable function over a convex set $\mathcal{D}$ is *convex* if and only if for every vector $v$ and every $x \in \mathcal{D}$, we have

$$v^\top \nabla^2 f(x) v \geq 0.$$

**Definition 1.3** (Convex Optimization)**.** An optimization problem is considered as *convex optimization* if $f$ is a convex function and $\mathcal{D}$ is a convex set.

Convex optimization is a very common class of optimization problems. Examples of convex optimization problems include

- Linear regression: $\min_x \|y - Ax\|_2^2$.
- Ridge regression: $\min_x \|y - Ax\|_2^2 + \lambda \|x\|_2^2$.
- Logistic regression: $\min_x \sum_i -\log \frac{1}{1+e^{-y_i \langle A_i, x \rangle}}$.

We examine some key properties of convex optimization problems mathematically.

**Lemma 1.4** (Lower Linear Bound)**.** *For every differentiable convex function $f$ over a convex set $\mathcal{D}$ and every $x, y \in \mathcal{D}$, we have*

$$f(y) \geq f(x) + \langle \nabla f(x), y - x \rangle.$$

*Proof.* Proof by contradiction. Assume for contradiction that for any $\varepsilon > 0$ exists $y \in \mathcal{D}$ such that $f(y) \leq f(x) + \langle \nabla f(x), y - x \rangle - \varepsilon$. Then by convexity, for all $0 \leq \lambda \leq 1$, we have

$$f((1-\lambda)x + \lambda y) \leq (1-\lambda)f(x) + \lambda f(y) \leq f(x) + \lambda \langle \nabla f(x), y - x \rangle - \lambda\varepsilon,$$

$$f(x + \lambda(y-x)) \leq f(x) + \lambda \langle \nabla f(x), y - x \rangle - \lambda\varepsilon$$

which is (for non-zero $\lambda$)

$$\frac{f(x + \lambda(y-x)) - f(x)}{\lambda} \leq \langle \nabla f(x), y - x \rangle - \varepsilon.$$

By definition,

$$\lim_{\lambda \to 0} \frac{f(x + \lambda(y-x)) - f(x)}{\lambda} = \langle \nabla f(x), y - x \rangle.$$

So let $\lambda \to 0^+$ we have $\langle \nabla f(x), y - x \rangle \leq \langle \nabla f(x), y - x \rangle - \varepsilon$, which is a contradiction. □

**Theorem 1.5.** *For a first order differentiable convex function $f$, $\nabla f(x^*) = 0$ iff $f(x^*) = \min_x f(x)$.*

*Proof.* ($\Rightarrow$) By the lower linear bound, for every $x \in \mathcal{D}$, $f(x) \geq f(x^*)$ since $\nabla f(x^*) = 0$.

($\Leftarrow$) Next lecture. □

**Corollary 1.5.1.** *For general Lipschitz function, $\exists y \in \partial f(x^*), y = 0$ iff $f(x^*) = \min_x f(x)$.*

## 2 Gradient Descent

**Definition 2.1** (Gradient Descent Algorithm)**.** Given a starting point $x_0$ and a learning rate $\eta$, the *gradient descent* algorithm to find the minimizer of a function $f$ is defined as follows: For every iteration $t = 0, 1, 2, \cdots$, update

$$x_{t+1} = x_t - \eta \nabla f(x_t).$$

The selection of the learning rate $\eta$ is important in the gradient descent algorithm. Intuitively, we might want to choose large learning rates for "smooth" functions and small learning rates for "steep" functions. This gives us the motivation to define the *$L$-smoothness* of functions. With the definition of $L$-smoothness, we can now derive a lemma for gradient descent about how much the function decreases.

**Definition 2.2** ($L$-smoothness)**.** A first order differentiable function (not necessarily convex) $f$ over a set (not necessarily convex) $\mathcal{D}$ is called *$L$-smooth* for some $L > 0$ if for every $x, y \in \mathcal{D}$, we have

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

The above formula is also referred to as the *upper quadratic bound*. Now for convex function $f$, we have both the lower linear bound and upper quadratic bound, giving us for every $x, y \in \mathcal{D}$,

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

There are some other properties of $L$-smoothness, given below.

**Remark 2.2.1** (Alternative Definition of $L$-smoothness)**.** A second order differentiable function over a convex set $\mathcal{D}$ is *$L$-smooth* if for all $x \in \mathcal{D}$, we have

$$L \geq \|\nabla^2 f(x)\|_{sp}$$

where $\| \cdot \|_{sp}$ is the *spectral norm*, defined as the largest singular value of the matrix. Or equivalently for every unit vector $v$

$$v^\top \nabla^2 f(x) v \leq L.$$

**Corollary 2.2.2.** *Every third order differentiable function over a closed, bounded convex set $\mathcal{D}$ is $L$-smooth for some finite $L$.*

With $L$-smoothness, we have a formal definition of smoothness. Then, we can derive an useful lemma for analyzing the convergence rate of gradient descent.

**Lemma 2.3** (Gradient Descent Lemma)**.** *For any $L$-smooth function $f$, as long as $\eta \leq \frac{1}{L}$, we have*

$$f(x_{t+1}) \leq f(x_t) - \frac{\eta}{2} \|\nabla f(x_t)\|_2^2.$$

*Proof.* Notice that $x_{t+1} = x_t - \eta \nabla f(x_t)$ in gradient descent. By definition of $L$-smoothness, we have

$$f(x_{t+1}) \leq f(x_t) - \langle \nabla f(x_t), \eta \nabla f(x_t) \rangle + \eta^2 \frac{L}{2} \|\nabla f(x_t)\|_2^2.$$

Since $\eta^2 \frac{L}{2} \leq \eta$ for $\eta \leq \frac{1}{L}$, we have

$$f(x_{t+1}) \leq f(x_t) - \frac{\eta}{2} \|\nabla f(x_t)\|_2^2$$

as desired. □

**Lemma 2.4** (Convergence Rate of Gradient)**.** *In gradient descent, for every $\varepsilon > 0$, let*

$$T_\varepsilon = \frac{2(f(x_0) - \min_x f(x))}{\eta\varepsilon},$$

*then there exists $t \le T_\varepsilon$ such that $\|\nabla f(x_t)\|_2^2 \le \varepsilon$.*

*Proof.* By contradiction. Suppose for every $t \le T_\varepsilon$, $\|\nabla f(x_t)\|_2^2 > \varepsilon$. Then,

$$f(x_{t+1}) < f(x_t) - \frac{\eta}{2}\varepsilon$$

which implies that

$$f(x_{T_\varepsilon}) < f(x_0) - \frac{\eta}{2}T_\varepsilon\varepsilon = f(x_0) - (f(x_0) - \min_x f(x)) = \min_x f(x)$$

which is a contradiction. □

**Remark 2.4.1.** Lemma 2.4 shows that the learning rate does not depend on how large the gradient is, although in practice we do want to tune the learning rate for faster convergence.

Lemma 2.4 shows that we can decrease the objective, however if the gradient is too small, it might take forever for gradient to reach exactly zero. Take the function $f(x) = \varepsilon^2 x^2$ as an example, when $x = \frac{1}{\varepsilon}$ we have $|\nabla f(x)| = \varepsilon$ but $f(x) = 1$. Therefore, we also need to study the convergence rate of gradient descent.

**Lemma 2.5** (Mirror Descent Lemma)**.** *In gradient descent, for any L-smooth function $f$ and $y \in \mathcal{D}$, we have*

$$f(x_t) \le f(y) + \frac{1}{2\eta}\left(\|y - x_t\|_2^2 - \|y - x_{t+1}\|_2^2 + \|x_{t+1} - x_t\|_2^2\right).$$

*Proof.* By lower linear bound we have

$$f(y) \ge f(x_t) + \langle \nabla f(x_t), y - x_t\rangle = f(x_t) + \frac{1}{\eta}\langle x_t - x_{t+1}, y - x_t\rangle.$$

We observe that

$$\begin{aligned}\langle x_t - x_{t+1}, y - x_t\rangle &= \langle x_{t+1}, x_t\rangle + \langle y, x_t\rangle - \langle x_{t+1}, y\rangle - \langle x_t, x_t\rangle \\ &= -\frac{1}{2}\left(\|y - x_t\|_2^2 - \|y - x_{t+1}\|_2^2 + \|x_{t+1} - x_t\|_2^2\right).\end{aligned}$$

Hence we have the desired result. □

**Remark 2.5.1.** The mirror descent lemma implies that if the current function value $f(x_t)$ is much larger than $f(x^*)$, then since $\|x_{t+1} - x_t\|_2^2$ is small, the values $\|x^* - x_{t+1}\|_2^2$ must be much smaller than $\|x^* - x_t\|_2^2$, so $x_{t+1}$ will be much closer to $x^*$ compared to $x_t$. So the mirror descent lemma looks at decreasing the distance between $x_t$ to $x^*$.

**Theorem 2.6** (Convergence Rate of Gradient Descent)**.** *In gradient descent, as long as $\eta \le \frac{1}{L}$, we have*

$$f(x_T) \le f(x^*) + \frac{\|x^* - x_0\|_2^2}{\eta T}.$$

*Proof.* Let $y = x^*$, summing up the mirror descent lemma for $t = 0, \cdots, t-1$, we have the *telescoping sum* inequality

$$\sum_{t=0}^{T-1} f(x_t) \le Tf(x^*) + \frac{1}{2\eta}\left(\|x^* - x_0\|_2^2 - \|x^* - x_T\|_2^2 + \sum_{t=0}^{T-1}\|x_{t+1} - x_t\|_2^2\right)$$

$$\frac{1}{T}\sum_{t=0}^{T-1} f(x_t) \leq f(x^*) + \frac{1}{2\eta T}\left(\|x^* - x_0\|_2^2 - \|x^* - x_T\|_2^2 + \sum_{t=0}^{T-1}\|x_{t+1} - x_t\|_2^2\right)$$

$$\frac{1}{T}\sum_{t=0}^{T-1} f(x_t) \leq f(x^*) + \frac{1}{2\eta T}\left(\|x^* - x_0\|_2^2 + \sum_{t=0}^{T-1}\|x_{t+1} - x_t\|_2^2\right)$$

which implies using $x_{t+1} = x_t - \eta\nabla f(x_t)$

$$\frac{1}{T}\sum_{t=0}^{T-1} f(x_t) \leq f(x^*) + \frac{1}{2\eta T}\|x^* - x_0\|_2^2 + \frac{\eta}{2T}\sum_{t=0}^{T-1}\|\nabla f(x_t)\|_2^2.$$

When $\eta \leq \frac{1}{L}$, by Gradient Descent Lemma we have

$$\frac{\eta}{2}\sum_{t=0}^{T-1}\|\nabla f(x_t)\|_2^2 \leq f(x_0) - f(x_T) \leq f(x_0) - f(x^*)$$

since $x^*$ is a minimizer of $f$. Then,

$$\frac{1}{T}\sum_{t=0}^{T-1} f(x_t) \leq f(x^*) + \frac{1}{T}\left(\frac{1}{2\eta}\|x^* - x_0\|_2^2 + f(x_0) - f(x^*)\right).$$

Using the $L$-smoothness of $f$ that shows $f(x_0) - f(x^*) \leq \frac{L}{2}\|x^* - x_0\|_2^2$ and the gradient descent lemma that shows $f(x_T) \leq f(x_t)$, we have

$$f(x_T) \leq f(x^*) + \frac{\|x^* - x_0\|_2^2}{T}\left(\frac{1}{2\eta} + \frac{L}{2}\right)$$

which implies when $\eta \leq \frac{1}{L}$

$$f(x_T) \leq f(x^*) + \frac{\|x^* - x_0\|_2^2}{\eta T}.$$

□

# 3 Momentum

The intuition behind *accelerated gradient descent* is to use larger learning rate $\eta > \frac{1}{L}$ without entering zig-zag. Some functions are only non-smooth at some corners, and using a low learning rate $\frac{1}{L}$ might not be the most efficient option.

The key idea of momentum is to use a universal large learning rate and use the "weighted" sum of the gradients from the previous iterations to update the current point. When gradients point to the same direction, the sum will be large. Otherwise, when the gradients bump back and forth, the sum will be small. The weighted sum of the past gradients is called the *momentum*.

**Definition 3.1** (Momentum)**.** Use a learning rate $\eta > \frac{1}{L}$, and update using

$$x_{t+1} = x_t - \eta g_t.$$

**Definition 3.2** (Nesterov's Accelerated Gradient Descent)**.** For a $L$-smooth function $f$, in each iteration, we compute

$$z_{t+1} = x_t - \eta \nabla f(x_t)$$
$$x_{t+1} = (1-\gamma_t) z_{t+1} + \gamma_t z_t$$
$$\lambda_0 = 0, \lambda_t = \frac{1+\sqrt{1+4\lambda_{t-1}^2}}{2}, \qquad \gamma_t = \frac{1-\lambda_t}{\lambda_{t+1}}.$$

**Remark 3.2.1.** Nesterov's accelerated gradient descent is guaranteed to work mathematically, but in practice it's not the best momentum. In contrast, people use the *heavy ball momentum* more often, for example in PyTorch `optim.SGD(momentum=0.9)`, but it doesn't have theoretical guarantee.

**Definition 3.3** (Heavy Ball Momentum)**.** The update can be approximated by

$$x_{t+1} \approx x_t - \eta g_t$$
$$g_t = \gamma \sum_{s \le t} (1-\gamma)^{t-s} \nabla f(x_s)$$

where the last step can be updated easily using

$$g_{t+1} = g_t(1-\gamma) + \gamma \nabla f(x_{t+1}).$$

Therefore, we can choose $\eta = \frac{1}{\gamma L} > \frac{1}{L}$.

**Remark 3.3.1.** There is no proof for why heavy ball momentum works, but we can do a thought experiment to show why this works intuitively. The key observation is that when gradient is smaller than usual, we can use a larger learning rate. We fix a value $K > 0$, if $\|f(x_t)\|_2^2 \ge K$ holds for *every* $t$, using $\eta = \frac{1}{L}$ and the gradient descent lemma, we have

$$f(x_{t+1}) \le f(x_t) - \frac{K}{2L}$$

so we need at most $\frac{Lf(x_0)}{K}$ iterations to find a point $x_T$ with $f(x_T) \le \frac{f(x_0)}{2}$. If $\|f(x_t)\|_2^2 < K$ holds for *every* $t$, using the telescoping sum of mirror descent lemma, we have

$$\frac{1}{T}\sum_{t=0}^{T-1} f(x_t) \le \frac{1}{2\eta T}\|x^* - x_0\|_2^2 + \frac{\eta K}{2}$$

with the assumption that $f(x^*) = 0$. With $\eta = \frac{f(x_0)}{2K}$, we need at most $\frac{4K\|x_0 - x^*\|_2^2}{f(x_0)^2}$ iterations to find a point $x_T$ with $f(x_T) \le \frac{f(x_0)}{2}$.

Therefore, picking $K = \sqrt{\frac{Lf^3(x_0)}{4\|x_0-x^*\|_2^2}}$, in both cases we need at most $\frac{2\|x_0-x^*\|_2\sqrt{L}}{\sqrt{f(x_0)}}$ iterations to find a point $x_T$ with $f(x_T) \leq \frac{f(x_0)}{2}$. In the second case, when $f(x_0) \approx \|x_0 - x^*\|_2 \approx 1$, the learning rate is indeed much larger $\eta = \frac{f(x_0)}{K} \approx \frac{1}{\sqrt{L}} > \frac{1}{L}$. Therefore, the convergence of the thought experiment is, for $\varepsilon > 0$, we need at most $\frac{\sqrt{2L}}{\sqrt{\varepsilon}}$ iterations to find a point $x_{T_\varepsilon}$ with $f(T_\varepsilon) \leq \varepsilon$.

**Definition 3.4** (Linear Coupling)**.** For a $0 \leq \tau \leq 1$, at every iteration, we compute

$$\begin{aligned} s_{t+1} &= x_t - \frac{1}{L}\nabla f(x_t) && \text{(gradient descent)} \\ l_{t+1} &= l_t - \eta\nabla f(x_t) && \text{(momentum)} \\ x_{t+1} &= (1-\tau)s_{t+1} + \tau l_{t+1}. && \text{(updated value)} \end{aligned}$$

**Remark 3.4.1.** Linear coupling is a combination of gradient descent and momentum. We need this because in the thought experiment, it might be the case that neither $\|\nabla f(x_t)\|_2^2 \geq K$ and $\|\nabla f(x_t)\|_2^2 < K$ holds for *every* $t$.

# 4 Constraint Optimization

**Definition 4.1** (Constraint Convex Optimization)**.** A convex optimization function is a *constraint convex optimization* if the set $\mathcal{D}$ is a strict subset of $\mathbb{R}^d$.

**Remark 4.1.1.** A problem with gradient descent in the constraint convex optimization problem is $x_t \in \mathcal{D}$ does not necessarily imply $x_{t+1} \in \mathcal{D}$.

**Definition 4.2** (Algorithmic Projected Gradient Descent)**.** The update rule for convex set $\mathcal{D}$ is

$$x_{t+1} = \Pi_{\mathcal{D}}(x_t - \eta \nabla f(x_t)),$$

where

$$\Pi_{\mathcal{D}}(x) = \arg\min_z \|z - x\|_2^2.$$

**Remark 4.2.1.** For many convex sets $\mathcal{D}$, the projection is not easy to compute. For example, the polytope defined as

$$\mathcal{D} = \left\{x \in \mathbb{R}^d \mid \forall i \in [m], \langle w_i, x\rangle \leq 0; \forall j \in [n], \langle v_j, x\rangle + b_i = 0\right\}$$

There will be methods such as the *min-max optimization algorithm* and the *interior point method* to solve them. We will also cover *duality* in Lecture 7.

**Lemma 4.3.** *For a convex set $\mathcal{D}$, for every $x \in \mathbb{R}^d$, the projection $\Pi_{\mathcal{D}}(x)$ is unique.*

*Proof.* If $x \in \mathcal{D}$, $\arg\min_z \|z - x\|_2^2 = x$. If $x \notin \mathcal{D}$, suppose $z_1, z_2 \in \mathcal{D}$, $z_1 \neq z_2$ such that $\|z_1 - x\|_2^2 = \|z_2 - x\|_2^2$, then $z' = \frac{z_1+z_2}{2} \in \mathcal{D}$ by convexity of $\mathcal{D}$. By convexity of $f(z) = \|z - x\|_2^2$,

$$\|z' - x\|_2^2 < \frac{\|z_1 - x\|_2^2 + \|z_2 - x\|_2^2}{2}.$$

Therefore $z_1$ and $z_2$ are not the projection. □

**Definition 4.4** (Gradient Mapping)**.** In the algorithmic projected gradient descent, the *gradient mapping* is defined as

$$g(x_t) = \frac{1}{\eta}\left(x_t - \Pi_{\mathcal{D}}(x_t - \eta \nabla f(x_t))\right)$$

and the update rule becomes

$$x_{t+1} = x_t - \eta g(x_t).$$

**Lemma 4.5** (Lower Linear Bound for Convex Set)**.** *For every $y' \notin \mathcal{D}$ and $y \in \mathcal{D}$,*

$$\langle y' - \Pi_{\mathcal{D}}(y'), \Pi_{\mathcal{D}}(y') - y\rangle \geq 0.$$

**Lemma 4.6** (Gradient Descent Lemma for Gradient Mapping)**.** *For convex set $\mathcal{D}$ and $L$-smooth function $f$, when $\eta \leq \frac{1}{L}$,*

$$f(x_{t+1}) \leq f(x_t) - \frac{\eta}{2}\langle \nabla f(x_t), g(x_t)\rangle$$

$$f(x_{t+1}) \leq f(x_t) - \frac{\eta}{2}\|g(x_t)\|_2^2.$$

*Proof.* We apply the lower linear bound for convex set

$$\langle x_t - \eta \nabla f(x_t) - x_{t+1}, x_{t+1} - x_t\rangle \geq 0$$

$$\langle -\eta \nabla f(x_t) + \eta g(x_t), -\eta g(x_t)\rangle \geq 0$$

which implies that

$$\langle \nabla f(x_t), g(x_t) \rangle \geq \|g(x_t)\|_2^2.$$

When the function is $L$-smooth, we have

$$f(x_{t+1}) \leq f(x_t) - \eta \langle \nabla f(x_t), g(x_t) \rangle + \frac{L\eta^2}{2} \|g(x_t)\|_2^2.$$

Therefore, using $\langle \nabla f(x_t), g(x_t) \rangle \geq \|g(x_t)\|_2^2$ we derive the desired result. □

**Lemma 4.7** (Mirror Descent Lemma for Gradient Mapping)**.** *For any $y \in \mathcal{D}$, we have*

$$f(x_t) \leq f(y) + \frac{1}{2\eta} \left( \|y - x_t\|_2^2 - \|y - x_{t+1}\|_2^2 + 2\eta^2 \langle \nabla f(x_t), g(x_t) \rangle \right).$$

# 5 Mirror Descent

Mirror descent is a generalization of gradient descent algorithm. Although it is rarely used in deep learning, it is more often used in online learning, which we will talk about later. We start by introducing the concept of *Bregman divergence*, a class of distances, that is crucial in the mirror descent algorithm.

**Definition 5.1** (Bregman Divergence)**.** For a differentiable convex function $g$, the *Bregman divergence* is defined as

$$D_g(x, y) = g(x) - g(y) - \langle \nabla g(y), x - y \rangle.$$

Bregman divergence is a large class of distances, covering many common distance metrics. For example, when $g(x) = \|x\|_2^2$, then $D_g(x, y) = \|x - y\|_2^2$, which is the *Euclidean distance*. When $g(x) = \sum_{i\in[d]} x_i \log x_i$ where each $x_i \geq 0$ and $\sum_{i\in[d]} x_i = \sum_{i\in[d]} y_i = 1$,

$$\begin{aligned} D_g(x, y) &= g(x) - g(y) - \langle \nabla g(y), x - y \rangle \\ &= \sum_{i\in[d]} (x_i \log x_i - y_i \log y_i) - \sum_{i\in[d]} (\log y_i + 1)(x_i - y_i) \\ &= \sum_{i\in[d]} \left( x_i \log \frac{x_i}{y_i} + (x_i - y_i) \right) \\ &= \sum_{i\in[d]} x_i \log \frac{x_i}{y_i} \end{aligned}$$

which is the *KL-divergence*.

**Definition 5.2** (Mirror Descent Algorithm)**.** Given a distance $D_g(x, y)$ defined by a differentiable convex function $g$, the *mirror descent* algorithm to minimize a function $f$ is defined as follows: At every iteration $t$, update

$$\nabla g(x_{t+1}) = \nabla g(x_t) - \eta \nabla f(x_t).$$

**Corollary 5.2.1.** *As long as $g(x) = \omega(\|x\|_2)$ when $\|x\|_2 \to \infty$, such $x_{t+1}$ can always be found. The criteria is sufficient but not necessary.*

We can see that mirror descent is a generalization of the gradient descent algorithm. When we use the Euclidean distance $g(x) = \frac{1}{2}\|x\|_2^2$, we have $\nabla g(x) = x$, so the mirror descent algorithm becomes the gradient descent algorithm. This is why we have $\|x^* - x_0\|_2^2$ in the convergence rate of gradient descent.

**Definition 5.3** ($L$-Lipschitzness)**.** A function $f$ is called $L$-Lipschitz with respect to a distance $D_g$ if for every $x, y$, we have

$$|\langle \nabla f(x), y - x \rangle| \leq L\sqrt{D_g(y, x)}.$$

Specifically, for $D_g(y, x) = \|y - x\|_2^2$, we have

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

or equivalently,

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

**Lemma 5.4** (Mirror Descent Lemma)**.** *In mirror descent, for $L$-Lipschitz convex function $f$ and any $y \in \mathcal{D}$, we have*

$$f(x_t) \leq f(y) + \frac{1}{\eta} \left( D_g(y, x_t) - D_g(y, x_{t+1}) + D_g(x_t, x_{t+1}) \right).$$

*Proof.* We first apply the lower linear bound

$$f(x_t) \leq f(y) - \langle \nabla f(x_t), y - x_t \rangle.$$

By the mirror descent update, we have

$$\nabla f(x_t) = \frac{1}{\eta} \left( \nabla g(x_t) - \nabla g(x_{t+1}) \right)$$

which implies

$$f(x_t) \leq f(y) - \frac{1}{\eta} \langle \nabla g(x_t) - \nabla g(x_{t+1}), y - x_t \rangle.$$

By definition of the distance $D_g$, we have

$$\begin{aligned} D_g(y, x_t) &= g(y) - g(x_t) - \langle \nabla g(x_t), y - x_t \rangle \\ -D_g(y, x_{t+1}) &= -g(y) + g(x_{t+1}) + \langle \nabla g(x_{t+1}), y - x_{t+1} \rangle \\ D_g(x_t, x_{t+1}) &= g(x_t) - g(x_{t+1}) - \langle \nabla g(x_{t+1}), x_t - x_{t+1} \rangle \end{aligned}$$

Hence,

$$-\langle \nabla g(x_t) - \nabla g(x_{t+1}), y - x_t \rangle = D_g(y, x_t) - D_g(y, x_{t+1}) + D_g(x_t, x_{t+1}).$$

Therefore, we have the mirror descent lemma

$$f(x_t) \leq f(y) + \frac{1}{\eta} \left( D_g(y, x_t) - D_g(y, x_{t+1}) + D_g(x_t, x_{t+1}) \right).$$

□

# 6 Stochastic Gradient Descent

In supervised learning, the problem is often an *empirical risk minimization*, where we were given training examples $\{x_i, y_i\}_{i=1}^N$ and we want to find a model $h$ such that $h(x_i) \approx y_i$. The key problem is that $N$ is usually extremely large here, and when $h$ is a giant neural network computing the exact gradient in gradient descent can be very costly. We want an algorithm that is *faster* and also *as good as possible*. Therefore we have the *stochastic gradient descent* algorithm.

**Definition 6.1** (Empirical Risk Minimization)**.** Given training data set $\{x_i, y_i\}_{i=1}^N$ where $x_i$'s are the *training data*, $y_i$'s are the *training labels*, the *ERM* type of problem is given as

$$\min_W \frac{1}{N} \sum_{i=1}^N \ell(h(x_i, W), y_i) + R(W)$$

where $h$ is a *parameterized model*, $W$ is the *trainable parameters*, $\ell$ is the *loss function*, and $R$ is the *regularizer*.

**Remark 6.1.1.** There is a mismatch of notations in machine learning and optimization. In machine learning, we use $W$ to denote the weight that we want to optimize and $x_i$ to denote the training examples, but in optimization the weight is actually $x$.

**Definition 6.2** (Stochastic Gradient Descent)**.** In the same setting as gradient descent, the *stochastic gradient descent* minimizes a function as

$$x_{t+1} = x_t - \eta \tilde{\nabla} f(x_t)$$

where $\mathbb{E}\left[\tilde{\nabla} f(x_t)\right] = \nabla f(x_t)$.

**Lemma 6.3** (Expectation of Stochastic Gradient)**.** *If we sample a subset $\mathcal{S}_t$ uniformly at random, the expectation of the stochastic gradient is the true gradient.*

*Proof.* Observe that $\mathbb{E}\left[\mathbf{1}_{i \in \mathcal{S}_t}\right] = \mathbb{P}(i \in \mathcal{S}_t) = \frac{m}{N}$ for any $i$ since the subset is sampled uniformly at random. Then,

$$\begin{aligned}
\mathbb{E}\left[\tilde{\nabla} f(W_t)\right] &= \mathbb{E}\left[\frac{1}{m} \sum_{i \in \mathcal{S}_t} \nabla \ell(h(x_i, W_t), y_i) + \nabla R(W_t)\right] \\
&= \frac{1}{m} \mathbb{E}\left[\sum_{i \in \mathcal{S}_t} \nabla \ell(h(x_i, W_t), y_i)\right] + \nabla R(W_t) \\
&= \frac{1}{m} \mathbb{E}\left[\sum_{i \in [N]} \mathbf{1}_{i \in \mathcal{S}_t} \nabla \ell(h(x_i, W_t), y_i)\right] + \nabla R(W_t) \\
&= \frac{1}{m} \sum_{i \in [N]} \mathbb{E}\left[\mathbf{1}_{i \in \mathcal{S}_t}\right] \nabla \ell(h(x_i, W_t), y_i) + \nabla R(W_t) \\
&= \frac{1}{m} \sum_{i \in [N]} \frac{m}{N} \nabla \ell(h(x_i, W_t), y_i) + \nabla R(W_t) \\
&= \frac{1}{N} \sum_{i \in [N]} \nabla \ell(h(x_i, W_t), y_i) + \nabla R(W_t) \\
&= \nabla f(W_t).
\end{aligned}$$

□

**Definition 6.4** (Variance of Stochastic Gradient)**.** The *variance* of stochastic gradient is defined as $\mathbb{E}\left[\|\tilde{\nabla} f(x_t)\|_2^2\right]$.

**Lemma 6.5** (Gradient Descent Lemma)**.** *In stochastic gradient descent with $L$-smooth function $f$, as long as $\mathbb{E}\left[\|\tilde{\nabla} f(x_t)\|_2^2\right] \leq G$, we have*

$$\mathbb{E}\left[f(x_{t+1})\right] \leq f(x_t) - \eta\|\nabla f(x_t)\|_2^2 + \frac{L}{2}\eta^2 G.$$

*Proof.* When $f$ is $L$-smooth, we have

$$f(x_{t+1}) \leq f(x_t) - \eta\langle \nabla f(x_t), \tilde{\nabla} f(x_t)\rangle + \frac{L}{2}\eta^2\|\tilde{\nabla} f(x_t)\|_2^2.$$

Since $\mathbb{E}\left[\tilde{\nabla} f(x_t)\right] = \nabla f(x_t)$, we have

$$\begin{aligned}\mathbb{E}[f(x_{t+1})] &\leq f(x_t) - \eta\|\nabla f(x_t)\|_2^2 + \frac{L}{2}\eta^2\,\mathbb{E}\left[\|\tilde{\nabla} f(x_t)\|_2^2\right] \\ &\leq f(x_t) - \eta\|\nabla f(x_t)\|_2^2 + \frac{L}{2}\eta^2 G.\end{aligned}$$

□

**Lemma 6.6** (Mirror Descent Lemma)**.** *In stochastic gradient descent, for every $y$ we have*

$$f(x_t) \leq f(y) + \mathbb{E}\left[\frac{1}{2\eta}\left(\|x_t - y\|_2^2 - \|x_{t+1} - y\|_2^2 + \|x_t - x_{t+1}\|_2^2\right)\right].$$

*Proof.* TODO □

**Theorem 6.7** (Convergence Rate of Stochastic Gradient Descent)**.** *In stochastic gradient descent, as long as $\eta = \sqrt{\frac{\|x_0 - x^*\|_2^2}{GT}}$, we have*

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}[f(x_t)] \leq f(x^*) + \frac{\|x_0 - x^*\|_2\sqrt{G}}{\sqrt{T}}.$$

*Proof.* TODO □

Finally, since we've shown that the convergence rate depends on the variance, we look at how to reduce the *variance* of the stochastic gradient.

# 7 Duality and Min-Max Optimization

As mentioned in Remark 4.2.1, $\Pi_{\mathcal{D}}$ is not always easily computable, but we still want to optimize over constraint sets.

**Definition 7.1** (Lagrange Duality)**.** For differentiable functions $h_1, ..., h_m, l_1, ..., l_n$, given the constraint set

$$\mathcal{D} = \left\{x \in \mathbb{R}^d \mid \forall i \in [m], h_i(x) \leq 0; \forall j \in [n], l_j(x) = 0\right\},$$

we define the *Lagrangian*

$$L(x, u, v) = f(x) + \sum_{i\in[m]} u_i h_i(x) + \sum_{j\in[n]} v_j l_j(x),$$

then the *Lagrange dual function* is defined as

$$g(u, v) = \min_{x\in\mathbb{R}^d} L(x, u, v)$$

**Theorem 7.2.** *To solve* $\min_{x\in\mathcal{D}} f(x)$*, where* $\mathcal{D}$ *defined the same as above, one can alternatively solve*

$$\min_{x\in\mathbb{R}^d} \max_{u\geq 0,v} L(x, u, v).$$

*Proof.* Clearly, if $x \notin \mathcal{D}$, then $\max_{u\geq 0,v} L(x, u, v) = \infty$, otherwise $\max_{u\geq 0,v} L(x, u, v) = f(x)$. □

However, solving this is still as difficult as the original problem. In particular, $\max_{u\geq 0,v} L(x, u, v)$ is not continuous. It is more like solving

$$\min_{x\in\mathbb{R}^d} f(x) + \lambda \mathbf{1}_{x\in\mathcal{D}}$$

for $\lambda \to \infty$. In contrast, solving

$$\max_{u\geq 0,v} g(u, v) = \max_{u\geq 0,v} \min_{x\in\mathbb{R}^d} L(x, u, v)$$

will be much easier. The good point of this problem is that $g(u, v)$ is typically a differentiable function of $u, v$, in particular the minimizer is finite. An analogy is that the latter form is like the professor first decides the distribution of scores in an exam and punish more the harder questions, and then the students try to maximize their score according to the scores. However, in the previous form, it's like the professor trys to minimize the scores of students after they submit the exam. In this case, if the students get anything wrong, the professor will make all the penalty on this problem. This is ridiculous and the only way you can maximize your score is to get everything correct.

However, why does it make sense? What can we get by solving $g(u, v)$? How to solve it efficiently? To see this, we need the following weak and strong duality argument.

**Lemma 7.3** (Weak Duality)**.** *For every* $u \geq 0, v$*,* $g(u, v) \leq f(x^*)$*, where* $x^*$ *is a minimizer of* $f(x)$*.*

*Proof.* By definition,

$$g(u, v) = \min_{x\in\mathbb{R}^d} L(x, u, v) \leq L(x^*, u, v) = f(x^*) + \sum_{i\in[m]} u_i h_i(x^*) + \sum_{j\in[n]} v_j l_j(x^*) \leq f(x^*).$$

where the last step holds since $x^* \in \mathcal{D}$ and $h_i(x^*) \leq 0$ and $l_j(x^*) = 0$. □

**Lemma 7.4.** $g(u, v)$ *is concave, regardless of* $f, h, l$*.*

*Proof.* For every $u, u', v, v'$ and $0 \leq \lambda \leq 1$,

$$
\begin{aligned}
& g\left((1-\lambda)u + \lambda u', (1-\lambda)v + \lambda v'\right) \\
&= \min_{x \in \mathbb{R}^d} \left\{ f(x) + \sum_{i \in [m]} \left((1-\lambda)u_i + \lambda u_i'\right) h_i(x) + \sum_{j \in [n]} \left((1-\lambda)v_j + \lambda v_j'\right) l_j(x) \right\} \\
&\leq (1-\lambda) \min_{x \in \mathbb{R}^d} \left\{ f(x) + \sum_{i \in [m]} u_i h_i(x) + \sum_{j \in [n]} v_j l_j(x) \right\} + \lambda \min_{x \in \mathbb{R}^d} \left\{ f(x) + \sum_{i \in [m]} u_i' h_i(x) + \sum_{j \in [n]} v_j' l_j(x) \right\} \\
&= (1-\lambda) g(u, v) + \lambda g(u', v').
\end{aligned}
$$

Hence $g(u, v)$ is always concave. □

**Lemma 7.5** (Slater's Condition)**.** *When $h_i, f$ are all convex and $l_j$ are all linear, and if there is an $x$ such that $\forall i \in [m]$, $h_i(x) < 0$ and $\forall j \in [n]$, $l_j(x) = 0$, let $u^*, v^*$ be a maximizer of $g(u, v)$, then*

$$g(u^*, v^*) = f(x^*).$$

*Proof.* Let us consider the set $\mathcal{S} \in \mathbb{R}^{m+n+1}$ defined as all points $(u, v, t)$ where $u \in \mathbb{R}^m, v \in \mathbb{R}^n, t \in \mathbb{R}$ such that $\exists x \in \mathbb{R}^d, \forall i \in [m], h_i(x) \leq u_i$ and $\forall j \in [n], l_j(x) = v_j, f(x) \leq t$. Then, we observe that $\mathcal{S}$ is a convex set, and $(0, 0, f(x^*))$ is on the boundary of the set $\mathcal{S}$. By the lower linear bound of the convex set, there exists $\lambda, \nu, \mu, \alpha$ such that $\forall (u, v, t) \in \mathcal{S}$,

$$\sum_{i \in [m]} \lambda_i u_i + \sum_{j \in [n]} \nu_j v_j + \mu t \geq \alpha$$

$$\mu f(x^*) \leq \alpha.$$

This implies that

$$\sum_{i \in [m]} \lambda_i u_i + \sum_{j \in [n]} \nu_j v_j + \mu t \geq \mu f(x^*).$$

We further observe that $(\infty, 0, f(x^*)) \in \mathcal{S}$, so $\lambda_i \geq 0$. By Slater's condition, $(< 0, 0, > f(x^*)) \in \mathcal{S}$, so $\mu > 0$. This implies that $\forall (u, v, t) \in \mathcal{S}$,

$$\sum_{i \in [m]} \frac{\lambda_i}{\mu} u_i + \sum_{j \in [n]} \frac{\nu_j}{\mu} v_j + t \geq f(x^*)$$

□

**Theorem 7.6** (Karush-Kuhn-Tucker Condition)**.** *Let $x^*, u^*, v^*$ be a solution of*

$$\max_{u \geq 0, v} \min_{x \in \mathbb{R}^d} L(x, u, v)$$

*then the following conditions must hold:*

- *Stationarity:* $\nabla_x L(x^*, u^*, v^*) = \nabla f(x^*) + \sum_{i \in [m]} u_i^* \nabla h_i(x^*) + \sum_{j \in [n]} v_j^* \nabla l_j(x^*) = 0$.
- *Complementary slackness:* $u_i^* h_i(x^*) = 0$ *for all* $i \in [m]$.
- *Primal feasibility:* $x^* \in \mathcal{D}$.
- *Dual feasibility:* $u^* \geq 0$.

*Proof. Stationarity*: Since $x^* = \arg\min_x L(x, u^*, v^*)$, we have $\nabla_x L(x^*, u^*, v^*) = 0$.

*Primal feasibility*: Proof by contradiction. Suppose $x^* \notin \mathcal{D}$, then either $h_i(x^*) > 0$ or $l_i(x^*) \neq 0$. If it's the first case, suppose we let $\delta > 0$ be sufficiently small and update $u_i' = u_i^* + \delta$, then we have

$$\begin{aligned} L(x(u', v^*), u', v^*) &\geq L(x(u^*, v^*), u^*, v^*) + \delta h_i(x^*) + \delta \left\langle \nabla_x L(x(u^*, v^*), u^*, v^*) m, \frac{\partial x(u^*, v^*)}{\partial u_i} \right\rangle - o(\delta) \\ &= L(x(u^*, v^*), u^*, v^*) + \delta h_i(x^*) - o(\delta). \end{aligned}$$

When $\delta$ is sufficiently small, we know $L(x(u', v^*), u', v^*) > L(x^*, u^*, v^*)$, which is a contradiction. The proof is similar for the second case.

*Complementary slackness*: Proof by contradiction. We know that $h_i(x^*) \leq 0$. Suppose $h_i(x^*) < 0$ and $u_i^* > 0$, we let $\delta > 0$ be sufficiently small and update $u_i' = u_i^* - \delta$, then we have

$$\begin{aligned} L(x(u', v^*), u', v^*) &\geq L(x(u^*, v^*), u^*, v^*) - \delta h_i(x^*) - \delta \left\langle \nabla_x L(x(u^*, v^*), u^*, v^*) m, \frac{\partial x(u^*, v^*)}{\partial u_i} \right\rangle - o(\delta) \\ &= L(x(u^*, v^*), u^*, v^*) - \delta h_i(x^*) - o(\delta). \end{aligned}$$

which is a contradiction.

*Dual feasiblity*: Trivial since $h_i(x) \leq 0$. □

**Theorem 7.7** (Karush-Kuhn-Tucker Theorem)**.** *Let $x^*, u^*, v^*$ be a solution of*

$$\max_{u \geq 0, v} \min_{x \in \mathbb{R}^d} L(x, u, v),$$

*then $x^*$ is a global minimal of $f$ in $\mathcal{D}$.*

*Proof.* By KKT condition, we know that

$$L(x^*, u^*, v^*) = f(x^*) + \sum_{i \in [m]} u_i h_i(x^*) + \sum_{j \in [n]} v_i l_i(x^*) = f(x^*)$$

By the weak duality, $L(x^*, u^*, v^*) \leq \min_{x \in \mathcal{D}} f(x)$. □

After we got the KKT condition/theorem, in order to solve it, we want to solve the *min-max optimization* problem, which more generally has the form

$$\max_{y \in \mathcal{S}_1} \min_{x \in \mathcal{S}_2} \phi(x, y).$$

How to solve the simple constraint min-max optimization problem?

Min-max optimization problems are very popular, for example the *generative adversarial networks (GANs)*, where we construct a generator $G$ and discriminator $D$ and solve

$$\max_D \min_G \mathop{\mathbb{E}}_{\omega \sim N(0,1)} [D(G(\omega))] - \mathop{\mathbb{E}}_{X \sim p^*} [D(X)].$$

**Definition 7.8** (Gradient Descent Ascent)**.** The *gradient descent ascent* algorithm to solve the min-max optimization problem $\max_y \min_x \phi(x, y)$ is defined as follows: At every iteration, update

$$\begin{aligned} x_{t+1} &= x_t - \eta \nabla_x \phi(x_t, y_t) \\ y_{t+1} &= y_t + \eta \nabla_y \phi(x_t, y_t). \end{aligned}$$

**Corollary 7.8.1.** *The convergence rate of gradient descent ascent is $O\left(\sqrt{\frac{L}{T}}\right)$ when $\phi$ is $L$-smooth, convex in $x$ and concave in $y$.*

**Definition 7.9** (Mirror-Prox)**.** The algorithm *mirror-prox* to solve the min-max optimization problem is defined as follows: At every iteration, update

$$u_{t+1} = x_t - \eta \nabla_x \phi(x_t, y_t)$$
$$v_{t+1} = y_t + \eta \nabla_y \phi(x_t, y_t)$$

according to *mirror descent ascent* and

$$x_{t+1} = x_t - \eta \nabla_x \phi(u_{t+1}, v_{t+1})$$
$$y_{t+1} = y_t + \eta \nabla_y \phi(u_{t+1}, v_{t+1})$$

according to *proximal descent ascent.*

**Corollary 7.9.1.** *The convergence rate of mirror prox is* $O\left(\frac{L}{T}\right)$ *when* $\phi$ *is* $L$*-smooth, convex in* $x$ *and concave in* $y$*. In particular, for every* $x, y$*,*

$$\frac{1}{T} \sum_{t \in [T]} (\phi(x, u_t) - \phi(v_t, y)) \leq O\left(\frac{L}{T}\right)$$

This bound can be lower in theory. We will not prove this convergence rate here, but the key is to apply the mirror descent lemma on the two updates.

# 8 Distributed Optimization

*Distributed optimization* is the opposite of *stochastic gradient descent*, where we assume that the machine has close to infinite computational power and can optimize any function very fast. Then we ask, what optimization algorithm do we need or do we even need any algorithm in this context. The key challenge is, then $N$ is extremely large, the data can only be stored on different machines. Each individual machine has strong computational power, but the communication between the machines are very limited. Memory constraint is a crucial problem in modern machine learning, as the computational power of GPUs are strong but the memory is limited.

Besides memory constraint, privacy problem might be another issue to consider in distributed optimization. For example, Alice, Bob, and Charlie each have $N/3$ data, but the data contains private information that they do not want to share to others. Is there a way to optimize without sharing these information?

**Definition 8.1** (Distributed Optimization)**.** In the *ERM* setting, the data is stores on $m$ machines, the machines each having a *disjoint data set* $S_1, S_2, \cdots, S_M$, where $\bigcup_{j\in[m]} S_j = [N]$. The goal is to find the minimizer of the problem and minimize the *communication* between the machines.

In order to solve for this problem, we first try a naïve method: each machine minimizes

$$\min_W \frac{1}{|S_j|} \sum_{i\in S_j} \ell(h(x_i, W), y_i) + R(W).$$

However, this won't work. Why because the disjoint datasets are not sampled uniformly. For example, if we do classification problem, each subset can contain exactly all the data with one label, so optimizing on the subsets will not work in this case. In fact, we have Remark 8.1.1.

**Remark 8.1.1.** The disjoint data sets are not sampled uniformly, and might be even biased. Therefore, there is no way to do optimization separately and combine them in some way.

A second approach is, we compute the gradient separately, and do gradient descent on a main machine. At each iteration, each machine computes

$$\nabla_j = \frac{1}{N} \sum_{i\in S_j} \nabla\ell(h(x_i, W), y_i)$$

and send to the central server. The central server computes

$$\nabla f(W_t) = \sum_{j\in[m]} \nabla_j + \nabla R(W).$$

This works, and the *per iteration* communication cost is $O(md)$, with the same convergence rate as the gradient descent. However, this is still relatively slow as the communication cost depends on the convergence rate of gradient descent. When the smoothness or Lipschitzness of $f$ is not very good, this is going to be slow. Can we do better?

We look at the *ADMM* algorithm, that does not depend on the convergence rate of the gradient descent.

**Definition 8.2** (ADMM)**.** The *alternating direction method of multiplier (ADMM)* algorithm is defined as follows: At each iteratin $t$, maintain a shared $W^{(t)}$ on across different machines, and local $\{W_j^{(t)}, \alpha_j^{(t)}\}$ on each machine. Each machine locally compute

$$W_j^{(t+1)} = \arg\min_{W_j} \left( f_j(W_j) + \lambda \|W_j - W^{(t)}\|_2^2 + \langle \alpha_j^{(t)}, W_j - W^{(t)} \rangle \right).$$

Then, all the machine together computed

$$W^{(t+1)} = \frac{1}{2m\lambda} \sum_{j\in[m]} \alpha_j^{(t)} + \frac{1}{m} \sum_{j\in[m]} W_j^{(t+1)}$$

and each machine update

$$\alpha_j^{(t+1)} = \alpha_j^{(t)} - \eta \left( W^{(t+1)} - W_j^{(t+1)} \right).$$

**Lemma 8.3.** *If we choose $\alpha_j^{(0)} = 0$ for all $j$, the update rule for $W^{(t+1)}$ is*

$$W^{(t+1)} = \frac{1}{m} \sum_{j\in[m]} W_j^{(t+1)}.$$

*Proof.* TODO □

We then show the derivation of the ADMM algorithm using duality.

**Lemma 8.4** (Fenchel Duality)**.** *For a differentiable function $h$ and a matrix $A$, if*

$$G(\alpha) = -\min_X \{h(X) + \langle \alpha, AX \rangle\},$$

*we have for $X^* = \arg\min\{h(X) + \langle \alpha, AX\rangle\}$,*

$$\nabla G(\alpha) = -AX^*.$$

*Proof.* TODO □

**Theorem 8.5.** *The ADMM algorithm finds the minimizer of the function $f$.*

*Proof.* TODO □

Now we analyze the convergence rate of ADMM and show that it does not depend on the smoothness or Lipschitzness of the function $f$. We first derive the *mirror descent lemma* for ADMM.

**Lemma 8.6** (Mirror Descent Lemma for ADMM)**.** *For every $W$ and $\eta$, if $\eta = 2\lambda$,*

$$\frac{1}{m} \sum_{j\in[m]} f_j \left( W_j^{(t+1)} \right) \le f(W) + \frac{\eta}{2} \left( \|W - W^{(t)}\|_2^2 - \|W - W^{(t+1)}\|_2^2 - \|W^{(t)} - W^{(t+1)}\|_2^2 \right)$$
$$+ \frac{1}{m} \sum_{j\in[m]} \frac{1}{2\eta} \left( \|\alpha_j^{(t)}\|_2^2 - \|\alpha_j^{(t+1)}\|_2^2 - \|\alpha_j^{(t+1)} - \alpha_j^{(t)}\|_2^2 \right)$$

*Proof.* TODO □

**Theorem 8.7** (Convergence Rate of ADMM)**.** *In ADMM, as long as $\eta = 2\lambda$, we have*

$$\frac{1}{m} \sum_{j\in[m]} f_j \left( W_j^{(t+1)} \right) \le f(W) + \frac{1}{\sqrt{T}} \|W - W^{(0)}\|_2^2$$

*Proof.* TODO □

# 9 Proximal Algorithms

**Definition 9.1** (Proximal Mapping)**.** Given a function $h$, its *proximal mapping* is defined as

$$\text{prox}_h(x) = \arg\min_z \left\{ h(z) + \frac{1}{2}\|z - x\|_2^2 \right\}.$$

In our setting, $\text{prox}_h(x)$ is assumed to be easily computable, the computation time is ignorable. This often has a closed form solution. Some examples of proximal mappings are in Remark 9.1.1.

**Remark 9.1.1.** When $h(x) = \langle w, x \rangle$ is a linear function of $x$, then the proximal mapping is $\text{prox}_h(x) = x - w$. In constraint optimization, if we define

$$h(x) = \begin{cases} \infty & \text{if } x \notin \mathcal{D} \\ 0 & \text{otherwise} \end{cases}$$

then $\text{prox}_h(x) = \Pi_{\mathcal{D}}(x)$ is the projection of $x$.

**Definition 9.2** (Proximal Gradient Descent)**.** The *proximal gradient descent* algorithm to minimize a function $f(x) = g(x) + h(x)$ is defined as follows: At every iteration $t$, update

$$y_t = x_t - \eta \nabla g(x_t) \qquad \text{(gradient descent step)}$$
$$x_{t+1} = \text{prox}_{\eta h}(y_t) \qquad \text{(proximal step)}$$

When $\eta$ is small, the proximal step does a gradient descent step on $h$ and the gradient descent step does a gradient descent step on $g$.

**Remark 9.2.1** (Iterative Soft-Thresholding Algorithm (ISTA))**.** The *iterative soft-thresholding algorithm (ISTA)* to solve Lasso-type problems is defined as follows:

**Theorem 9.3** (Convergence Rate of Proximal Gradient Descent)**.** *For every $x \in \mathcal{D}$, as long as $\eta \leq \frac{1}{L}$, we have*

$$\frac{1}{T}\sum_{t=0}^{T-1} f(x_{t+1}) \leq f(x) + \frac{\|x - x_0\|_2^2}{2\eta T}.$$

*Proof.* TODO □

# 10 Hessian and Pre-Conditioned Gradient Descent

# 11 Interior Point Method

# 12 Adagrad

# 13 Ellipsoid

# 14 Online Optimization

Online optimizatin studies the fundamental question: What if the function $f$ changes over time? There are two major categories: *online learning* and *reinforcement learning*. Typical scenarios include playing computer games (reinforcement learning) and pushing latest news to the users (online learning).

**Definition 14.1** (Online Optimization)**.** The *online optimization* is the following iterative game: At every iteration $t$, the environment picks a function $f_t : \mathcal{D} \to \mathbb{R}$, and the player chooses a point $x_t \in \mathcal{D}$, without knowing $f_t$. The environment then tells the player some information about $f_t$ at $x_t$: $f_t(x_t)$, and/or $\nabla f_t(x_t)$ and/or $\nabla^2 f_t(x_t)$, etc.

The difference between online learning and reinforcement learning is that in online learning, the environment is independent of the player's action, while in reinforcement learning the environment is changing with the player's action.

**Definition 14.2** (Regret)**.** The *regret* of the player is defined as

$$R := \frac{1}{T}\sum_{t\in[T]} f_t(x_t) - \min_{x\in\mathcal{D}} \frac{1}{T}\sum_{t\in[T]} f_t(x).$$

It seems like online learning is hard, but it turns out that to find the optimal $x^*$ that optimizes the regret is pretty simple.

**Definition 14.3** (Gradient Descent for Online Learning)**.** The *gradient descent* algorithm for online learning is defined as follows: At every iteration $t$, update

$$x_{t+1} = x_t - \eta \nabla f_t(x_t).$$

Or we can do *projected gradient descent*, define the *gradient mapping* as

$$g(x_t) = \frac{1}{\eta}\left(x_t - \Pi_{\mathcal{D}}(x_t - \eta \nabla f(x_t))\right)$$

and the update rule is

$$x_{t+1} = x_t - \eta g(x_t).$$

**Theorem 14.4.** *In gradient descent for online learning, when each $f_t$ is convex and $L$-Lipschitz, let $x^* = \arg\min_x \sum_{t\in[T]} f_t(x)$. Using learning rate $\eta = \frac{\|x^* - x_0\|_2}{L\sqrt{T}}$, the regret is bounded by*

$$\frac{1}{T}\sum_{t\in[T]} f_t(x_t) - \frac{1}{T}\sum_{t\in[T]} f_t(x^*) \le \frac{2\|x^* - x_0\|_2 L}{\sqrt{T}}.$$

*Proof.* When $f_t$ is $L$-Lipschitz, $\|g(x_t)\|_2 \le \|\nabla f(x_t)\|_2 \le L$. By the mirror descent lemma,

$$f_t(x_t) \le f_t(x^*) + \frac{1}{2\eta}\left(\|x^* - x_t\|_2^2 - \|x^* - x_{t+1}\|_2^2 + 2\eta^2 L^2\right)$$

so we can sum them up and get

$$\frac{1}{T}\sum_{t\in[T]} f_t(x_t) \le \frac{1}{T}\sum_{t\in[T]} f_t(x^*) + \frac{1}{2\eta T}\|x^* - x_0\|_2^2 + \eta L^2.$$

Therefore picking $\eta = \frac{\|x^* - x_0\|_2}{L\sqrt{T}}$ we complete the proof. □

**Remark 14.4.1.** Intuitively, why is this true? Because when $f_t$'s are all random, there is no $x^*$ that minimizes all functions, so the baseline is bad. When most of them has a common minimizer, since $f_t$'s are convex, the gradients are still pointing to $x^*$.

**Definition 14.5** (Online Newton's Method)**.** The *online newton's method*, or *online BFGS* algorithm is defined as follows: At every iteration $t$, update using a quasi-newton step

$$x_{t+1} = x_t - \eta_t B_t \nabla f_t(x_t)$$

where the *inverse Hessian matrix* $B_t$ is given by: for $s_{t+1} := \eta_t B_t \nabla f_t(x_t)$ and

$$y_t := \nabla f_t(x_t) - \nabla f_{t-1}(x_{t-1})$$

update for $\rho_t = \langle s_t, y_t \rangle^{-1}$:

$$B_t = (I - \rho_t s_t y_t^\top) B_{t-1} (I - \rho_t y_t s_t^\top) + \rho_t s_t s_t^\top .$$

There are several extensions to online optimization.

In standard optimization, we can estimate $\nabla f(x)$ using function value

$$\langle \nabla f(x), v \rangle = \lim_{t \to 0} \frac{f(x + tv) - f(x)}{t}$$

We can use *Stoke's formula*, for every $v \geq 0$,

$$\mathbb{E}\left[ \frac{d}{r} f(x + rv) v \right] = \nabla \tilde{f}(x)$$

where

$$\tilde{f}(x) = \int_{\|v\|_2 \leq 1} f(x + rv) \mathrm{d}v$$

# 15 Reinforcement Learning

**Definition 15.1** (Reinforcement Learning)**.** We define an environment as a *state* $s \in \mathbb{R}^S$, a player's *action* $a \in \mathbb{R}^d$, the *reward function* $r : \mathbb{R}^S \times \mathbb{R}^d \to \mathbb{R}$, and the *transition function*

$$P_a(s, s') = \mathbb{P}(\text{environment changes from } s \text{ to } s' \text{ after player taking action } a).$$

Then, the *reinforcement learning* regime is the following game: At each iteration, the player is at the state $s_t$. The player decides to take action $a_t$ based on $s_t$, and observe the reward $r(a_t, s_t)$. The environment then transit to state $s_{t+1}$ according to probability distribution $P_{a_t}(s_t, s_{t+1})$. The total discounted reward that the player collects is given by

$$R = \sum_{t \geq 0} \gamma^t r(s_t, a_t)$$

for some discount factor $0 < \gamma \leq 1$.

**Remark 15.1.1.** We should think of $a_t$ as $x_t$ and $r(s_t, *)$ as the function $-f_t$. $P_{a_t}(s_t, s_{t+1})$ measures the change of the functions according to the player's actions.

# 16 Variance Reduction

We learnt the reinforcement learning setting, the convergence of policy gradient heavily depends on the variance.

**Definition 16.1** (SAGA Algorithm)**.** The *SAGA* algorithm to minimize $f(x) = \frac{1}{N}\sum_{i\in[N]} f_i(x)$ is defined as follows. Initially

**Remark 16.1.1.** SAGA = an improved version of SAG (Stochastic Avergage Gradient).

# 17 Edge of Stability

# 18 Why Non-Convex Optimization?

There are some fundamental limitations of convex models, so it is impossible that we can turn deep learning models into convex models.

**Definition 18.1** (Deep Learning)**.** Given input $x$, for weights $M_L, M_{L-1}, \cdots, M_1$ and activation function $\sigma$, output

$$\mathrm{Deep}(x) = M_L\sigma(M_{L-1}\sigma(M_{L-2}\cdots\sigma(M_1x))).$$

**Definition 18.2** (Shallow Learning)**.** Given input $x$, for weight $M$ and *feature mapping* $\Phi$, output

$$\mathrm{Shallow}(x) = M\Phi(x).$$

**Definition 18.3** (Convex Optimization in Machine Learning)**.** In the *ERM* model,

**Remark 18.3.1.** $h(w,x)$ is a non-convex function of $x$, or has a non-convex decision boundary, does **NOT** imply $f(w)$ is a non-convex function of $x$. For example, when $x \in \mathbb{R}^2$, $h(w,x) = w_1x_1^2 + w_2x_2^2$. Suppose $w = (1,-1)$, the function $h(w,x) = x_1^2 - x_2^2$ is non-convex, non-concave of $x$, but it is a linear function of $w$.

**Definition 18.4** (Main Theorem)**.** Under some regularity conditions, when the loss function $\ell$ is convex, the function

$$f(w) = \sum_{i\in[N]} \ell(h(w,x^{(i)}), y^{(i)})$$

is a convex function *if and only if* $h(w,x) = \langle w, \Phi(x)\rangle$ is convex for some feature mapping $\Phi(x)$.

**Remark 18.4.1.** Increasing the layers of neural networks does not make a neural network more or less convex.

**Claim 18.5.** Empirically, shallow learning requires much *larger* model compared to deep learning models to achieve the same performance.

**Remark 18.5.1.** People tend to believe that the success of deep learning is due to its size. However, this is **NOT** why deep learning is better than shallow learning.

We will see why this works with a concept class

$$H(x) = F(x) + \alpha G(F(x))$$

# 19 Non-Convex Optimization

The goal of non-convex optimization is to find at least one local minima and find the global minima. The former one can be done efficiently, while the latter one is in general difficult, but possible in some settings.

The Taylor expansion of a function $f$ around any point $x$ is

$$f(x+\tau) = f(x) + \langle \nabla f(x), \tau \rangle + \frac{1}{2}\tau^\top \nabla^2 f(x)\tau \pm O(\tau^3).$$

By the Lipschitzness of the Hessian, we have

$$f(x+\tau) = f(x) + \langle \nabla f(x), \tau \rangle + \frac{1}{2}\tau^\top \nabla^2 f(x)\tau \pm \gamma\|\tau\|_2^3.$$

For non-convex functions, $\nabla^2 f(x)$ might not be positive semi-definite, so there might be *local minima* or *saddle points*.

**Definition 19.1** (Local Minima and Saddle Point)**.** A point $x$ is called *local minima* or *second-order local minima* if $\nabla f(x) = 0$ and $\nabla^2 f(x)$ is positive semi-definite. A point $x$ is called *saddle point* if $\nabla f(x) = 0$ and $\nabla^2 f(x)$ is not positive semi-definite, or equivalently there exists $v \in \mathbb{R}^d$ such that $v^\top \nabla^2 f(x) v < 0$.

## 20 Graduate Student Descent

# 21 Over-Parametrization

# 22 Over-Parametrization in Deep Learning

**Definition 22.1** (Neural Tangent Kernel)**.** The *neural tangent kernel* of a neural network $f$ is defined as

$$K(x, y) = \langle \nabla_W f(W_0, x), \nabla_W f(W_0, y) \rangle$$

where $W_0$ is the initialization.

**Theorem 22.2.** *No matter what the neural network $f$ is, the neural tangent kernel $K(x, y)$ is shallow and convex.*

# 23 Algorithmic Regularization

The empirical risk minimization problem asks us to optimize

$$\min_W \frac{1}{N} \sum_{i \in [N]} \ell(h(W, x_i), y_i) + \lambda R(W).$$

However, in the end, what we care more is the generalization error

$$\min_W \mathop{\mathbb{E}}_{x,y \sim \mathcal{D}} [\ell(h(W, x_i), y_i)].$$

Can we use different algorithms to

# 24 Quantum Optimization

**Definition 24.1** (Q-Bit)**.** A $n$-dimensional *q-bit* $z$ is of the following form

$$z = \sum_{s\in\{0,1\}^n} \alpha_s |s_1\rangle|s_2\rangle\cdots|s_n\rangle = \sum_{s\in\{0,1\}^n} \alpha_s |s\rangle$$

where $\sum_s |\alpha_s|^2 = 1$ and if I measure $z$, I have probability $|\alpha_s|^2$

**Definition 24.2** (Quantum Approximate Optimization Algorithm (QAOA))**.** The QAOA algorithm to solve non-convex optimization is defined as follows: Let $z_0 = \frac{1}{2^{n/2}} \sum_{s\in\{0,1\}^n} |s\rangle$, at every step $t$, update

$$|z(t+1)\rangle = e^{-i\beta B} e^{-iF\gamma} |z(t)\rangle.$$

QAOA does converge to global optimal as long as $t \to \infty$ and $\beta, \gamma$ are small enough.