# Lecture 9: Introduction to Optimization

*Instructor: Aditya Bhaskara* *Scribe: Aravinda Kanchana Ruwanpathirana*

**CS 5966/6966: Theory of Machine Learning**

*February 8th, 2022*

**Abstract**

In the last few lectures we saw how empirical risk minimization allows us to get good generalization bounds. In this lecture, we will briefly recap what we have seen and introduce the ideas of loss functions and optimizations. We will introduce the idea of using loss functions to approximate the errors, basics of convexity and the idea of approximating the error bounds using convex functions.

## 1 Introduction

Previously, in our course, we established a formal framework about learning. We studied the idea of learning and generalization over a distribution where the distribution captures all the data we have to learn from and we hope to learn a good enough hypothesis using a small set of samples. We also studied no-free lunch theorem which gave the notion of inductive bias and the necessity of choosing a hypothesis class for learning. We followed this with PAC learnability of a hypothesis class and the notions of agnostic learning and an improper learner. We also introduced the notion of a growth function of a hypothesis class and showed how this relates to the number of examples a random sample should contain for it to be a representative sample and thus lead to class being learnable from the specific sample. This enabled us to introduce the notion of shattering and VC dimensions, which as we saw, captured the notion of learnability. Formally, we saw that a hypothesis class $\mathcal{H}$ is PAC learnable (or agnostic PAC learnable) implies it has a finite VC dimension (and vice versa). Note that one of the implications of this is, if VC dimension of $\mathcal{H}$ is infinite, then it is not PAC learnable.

*PAC learning captures the idea of learning from examples where you are given a set of examples and you try to find the best hypothesis. PAC learning does not contain a notion of how the sampling is carried out and it does not assume that the examples are chosen carefully.*

*Agnostic learning implies that the target hypothesis does not belong to the hypothesis class we have chosen. Improper learner is an algorithm such that the output of the algorithm is outside of the hypothesis class (Ex: Nearest Neighbor Classifier).*

While what we have studied so far, allows us to get good generalization bounds, one thing we can see is that the bounds we have derived so far only applies to the case of Emperical Risk Minimization (ERM) algorithm, where you do proper learning. What we were able to show was that a small sample is a representative sample, and for any hypothesis in the class the training error over the representative sample and test error over the distribution are close enough. If the hypothesis is outside of your class, this does not imply anything. Therefore, the results would not hold true for an improper learner for instance. While ERM is a very strong algorithm that can give you the best hypothesis given enough samples, it turns out doing ERM is very inefficient in practice. Therefore, it is necessary to come up with an alternative approach that is efficient as well as that is able to give good enough guarantees. The people have come up with for this, is the idea of loss minimization and optimization, which will be the main topic of our discussion in today's class.

## 2 Loss Minimization

Assume we are given the task of finding the best linear classifier for a given set of points. It turns out this problem is NP hard even in this simplest setting. In order to handle this, we introduce the idea of loss functions and loss minimization. In the case of linear classifiers, if we use ERM, our goal would be to minimize the number of mistakes which we can see, is a discrete problem. With loss functions what we do is to relax this discrete problem in to continuous problem, which could allows us to approximately solve it efficiently.

Consider the setting where we are given $m$ examples $(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)$, where $x_i \in \mathbb{R}^n$ and $y_i \in \{-1, 1\}$ for all $i \in [m]$. Given the $m$ examples, the goal of ERM is to find the hypothesis that minimizes the loss (or error), which can be formally stated as,

$$\text{(1)} \quad \textsc{Minimize} \ \frac{1}{m} \sum_{i=1}^{m} \mathbb{1}\left[h(x_i) \neq y_i\right]$$

The idea of loss minimization can be seen to follow from the same idea. Now instead of minimizing the number of mistakes, we try to minimize the "loss" from each example. Let $l$ be some loss function (that is selected by us), then the goal in loss minimization is to,

$$\text{(2)} \quad \textsc{Minimize} \ \frac{1}{m} \sum_{i=1}^{m} l\left(h(x_i), y_i\right)$$

We can see that depending on the choice of loss functions we would be able to use this method to find good enough hypothesis efficiently whereas finding the best hypothesis for ERM (which is equivalent to setting loss function $l(h(x_i), y_i) = \mathbb{1}\left[h(x_i) \neq y_i\right]$) turns out to be NP-hard even for the linear classifiers.

Commonly used loss functions include,

1. Square loss, i.e. $l(h(x_i), y_i) = (h(x_i) - y_i)^2$
2. $\ell_1$ loss, i.e. $l(h(x_i), y_i) = |h(x_i) - y_i|$
3. Logistic loss, i.e. $l(h(x_i), y_i) = \log\left(\frac{1}{1+e^{-h(x_i)\cdot y_i}}\right)$

Generally, given a hypothesis $h$, we parameterize the hypothesis $h$ and then use loss functions to find the best parameters (the best hypothesis). We will see this in the example of the use of square loss function in finding the best linear classifier. Let $h(x) = \langle w, x \rangle + b$ where $w, x \in \mathbb{R}^n$ and $b \in \mathbb{R}$ (where $w, b$ are the parameters). Our goal is to find $w, b$ such that the "loss" is minimized. Given $m$ examples, this can be stated in terms of square loss as,

$$\textsc{Minimize} \ \frac{1}{m} \sum_{i=1}^{m} \left(y_i - \langle w, x \rangle - b\right)^2$$

*While this approach gives us a linear classifier, note that this might not result in the same linear classifier you might get using ERM. Loss minimization does not solve ERM exactly, but gives a way to solve it approximately. Since ERM is NP hard for even in the case of linear classifiers, we use loss minimization to solve it approximately.*

This turns out to be the least-squares regression problem, which is a well-studied problem in machine learning. This problem can be solved efficiently to find the best linear classifier in terms of squared loss.

Note that the choice of the loss function is important and depending on the problem, the choice of loss function would determine whether the hypothesis that minimizes the loss function can be found efficiently.

Loss minimization can be seen to part of the general mathematical framework of optimization. Next we will see a special class of optimization that would be useful to us when it comes to minimization tasks.

## 3 Convex Optimization

Following the idea of loss minimization, we will go over a class of optimization known as convex optimization. The idea of convex optimization is to, given a convex function defined on a convex domain, find the minimizer. To properly understand this we need to define what convex domain and convex functions are.

3.1 DEFINITION. (Convex Domain) Consider a domain $\mathcal{D} \in \mathbb{R}^d$. We call $\mathcal{D}$ a convex domain, if $\forall x, y \in \mathcal{D}$, we get that $\alpha x + \beta y \in \mathcal{D}\ \forall \alpha, \beta$ such that $\alpha + \beta = 1$.

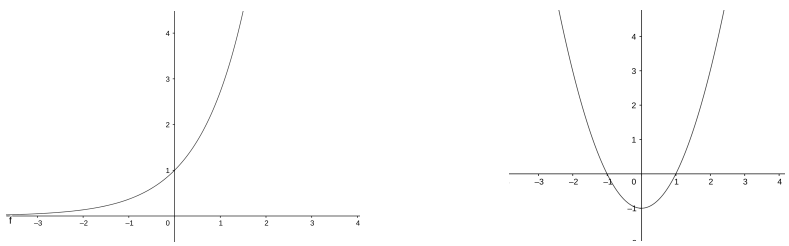
*We can also define a domain $\mathcal{D}$ to be convex if the line segment $xy$ is contained in $\mathcal{D}$ which is the same as Definition 3.1*

We can see that the domain $\mathbb{R}^d$ itself is a convex domain. Also note that we have not assumed anything about the boundedness of $\mathcal{D}$. $\mathcal{D}$ could be either bounded or unbounded.

3.2 DEFINITION. (Convex Function) Given a function $f : \mathbb{R}^d \to \mathbb{R}$, we call it a convex function if $\forall x, y \in \mathcal{D}omain(f)$,

$$f\left(tx + (1-t)y\right) \leq t \cdot f(x) + (1-t) \cdot f(y),\ \forall t \in [0,1] \tag{3}$$

*We can also define convex functions as function $f : \mathbb{R}^d \to \mathbb{R}$ such that $\forall x, y \in \mathcal{D}omain(f)$ $f\left(\frac{x+y}{2}\right) \leq \frac{f(x)+f(y)}{2}$. Also, in real analysis for functions $f : \mathbb{R} \to \mathbb{R}$, convexity can be defined as $f''(x) \geq 0$, $\forall x$ when $f$ is twice differentiable. The definition of 3 works for any function $f$.*



(a) $f(x) = e^x$ (b) $f(x) = x^2 - 1$

Figure 1: Some examples of convex functions

Convex functions defined on convex domains, adhere to certain properties that allow us to use them efficiently to solve minimization problems, as can be seen in the following lemma.

*While minimizing a convex function $f$ over a convex domain $\mathcal{D}$ is easy, the maximization of a convex function $f$ turns out to be hard. While the maximum of a convex function is always attained at the boundary of the domain, depending on the geometry of the boundary this could be hard.*

3.3 LEMMA. *Given a convex function $f$ defined on a convex domain $\mathcal{D}$, if $x \in \mathcal{D}$ is a local optimum for function $f$, then $x$ is also a global optimum, i.e.*

$$\min_{z \in \mathcal{D}} f(z) = f(x)$$

*$x \in \mathcal{D}$ is a local minimum if $\exists$ a ball of radius $r > 0$ such that $\forall y \in Ball(x,r)$ (where $Ball(x,r)$ is the ball of radius $r$ centered at $x$), $f(x) \leq f(y)$*

*Proof.* Let $z$ be the global optimum. Assume $f(z) < f(x)$. We can see that $\exists y \in Ball(x,r)$ and $t \in [0,1]$ such that $y = t \cdot z + (1-t) \cdot x$ (a point inside the ball on the line segment $xz$).

Since $f$ is convex we can see that,

$$f(y) \leq t \cdot f(z) + (1-t) \cdot f(x) < t \cdot f(x) + (1-t) \cdot f(x) = f(x)$$

However, this is a contradiction since $x$ is a local optimum and therefore

$f(x) \leq f(y)$. Therefore, we can see that $f(z) \geq f(x)$. Since $z$ is a global optimum, this implies $f(z) = f(x)$ so $x$ is also a global optimum. $\square$

The implication of this result is that if the domain and the function are convex, we could hope to use local search methods to find a solution. We could start with a point $x$ and then move to a neighboring point that results in a smaller $f$ value. Iteratively doing this would allow us to find a optimal solution (if you cannot move to a neighboring point to reduce the $f$ value, then you have found a local optimum and since the function and domain are convex, it is also a global optimum).

In the next lecture, we will introduce the gradient descent algorithm and its variants, which are built upon the aforementioned local search idea.