LECTURE #3

FEBRUARY 9, 2016

LECTURER: ELAD HAZAN SCRIBE: JOE ZHONG

1 Review: No Free Lunch Theorem

Recall that in lecture 2, we proved the No Free Lunch (NFL) Theorem:

Theorem 1.1. *For any positive integer m, there exists a domain X with cardinality $2m$, such that for any algorithm A over sample S with size m that outputs $A(S) : X \longrightarrow \{0,1\}$, there exists a distribution D and a concept f such that*
(1) $\text{err}(f) = 0$,
(2) $\text{err}(A(S)) \geq 1/10$ *with probability greater than* $1/10$.

The key idea is the *probabilistic method*. If we can prove

$$E_{f:X\to\{0,1\}}\left[E_{S\sim D}[\text{err}(A(S))]\right] \geq \frac{1}{4},$$

then the theorem follows by Markov's inequality.

Now we will present topics for today's lecture: agnostic learning and an introduction to convex optimization.

2 Agnostic Learning

The motivation for agnostic learning is to remove "realizability" assumption in PAC learning, which basically assumes that the concept f we are trying to learn is in the hypothesis class $\mathcal{H}$. This leads to the following definition.

Definition 2.1. A learning problem (X, Y, l) is agnostically learnable iff there exists a function $m : [0,1]^2 \to \mathbb{N}$ and an algorithm A, such that for any $\varepsilon, \delta > 0$, and any distribution D on (X, Y), given $m(\varepsilon, \delta)$ samples $S \sim D$, the algorithm A returns a hypothesis $A(S) \in \mathcal{H}$ such that with probability greater than $1 - \delta$,

$$\text{err}(A(S)) \leq \min_{h\in\mathcal{H}} \text{err}(h) + \varepsilon.$$

Similar to PAC learning, we have the empirical risk minimization (ERM) algorithm for agnostic learning. For samples $S \sim D^N$, the ERM returns hypothesis h_{ERM} given by

$$h_{\text{ERM}} = \operatorname*{argmin}_{h\in\mathcal{H}} \Big\{ \sum_{(x,y)\sim S} l(h(x), y) \Big\},$$

where l is the loss function.

The next theorem is in parallel with the result in PAC learning. We will not prove it in this lecture.

Theorem 2.2. *All finite hypothesis classes are learnable agnostically by the ERM algorithm, with a sample complexity*

$$m(\varepsilon, \delta) \leq \frac{1}{\varepsilon^2} \log \frac{|\mathcal{H}|}{\delta}.$$

Note that this theorem provides a nice bound in terms of error ε and the size of $|\mathcal{H}|$. The main issue for agnostic learning is computational efficiency. The goal for us is to learn agnostically within time $O(\text{poly}(1/\varepsilon, \log(1/\delta), \log|\mathcal{H}|))$.

3 Convex Optimization: An Introduction

The objective of convex optimization is to minimize a convex function subject to constraints, which are some forms of convex sets. First let us recall the definition of convex functions and convex sets.

Definition 3.1. A function $f : \mathbb{R}^d \to \mathbb{R}$ is convex iff for any $x, y \in \mathbb{R}^d$, and any $\alpha \in [0,1]$, it holds that

$$f(\alpha x + (1-\alpha)y) \leq \alpha f(x) + (1-\alpha) f(y).$$

Similarly, a set $K \subset \mathbb{R}^d$ is convex iff for any $x, y \in \mathbb{R}^d$, and any $\alpha \in [0,1]$, it holds that if $x, y \in K$, then $\alpha x + (1-\alpha)y \in K$.

Let the *sub-level set* S_t be $S_t = \{x \mid f(x) \leq t\}$. It can be easily checked that if f is convex, then all sub-level sets of f are convex. A simple example of a convex function is $f(x) = \|x\|^2 = x_1^2 + x_2^2$.

We say an optimization problem

$$\min_{x \in K \subset \mathbb{R}^d} f(x)$$

is a convex optimization problem if f and K are convex. We will see a few examples below.

NP hard example: max-cut problem.

$$\begin{array}{ll} \min & \sum_{(i,j)\in E} (x_i x_j - 1)/2 \\ \text{s.t.} & x_i^2 = 1,\ \forall i = 1, \ldots, d \end{array}$$

Convex examples:
(1) Support vector machine (2) LASSO (3) regression
(4) Linear programming (5) SDP (6) max flow problem

3.1 Computational Efficiency

Convex optimization has good computational guarantee. Suppose our convex optimization problem has input representation with
(1) an evaluation oracle and a membership oracle, which computes $f(x)$ or decides '$x \in K$' efficiently;
(2) explicit representation. (e.g. in LP, store A, b and c)
Then there are algorithms that output $x \in K$, such that

$$f(x) \leq \min_{x \in K} f(x) + \varepsilon,$$

with running time polynomial in dimension d, input representation and $\log(1/\varepsilon)$.

Note that we can only have approximate solutions for some optimization problems, such as regression problems that have real variables.

Algorithms for convex optimization include the ellipsoid method, the random-walk method, and the interior-point method, which run in polynomial time. However, they may be slow and impractical in machine learning.

3.2 Basic Properties of Convex Functions

The *gradient* of a (smooth) convex function f is denoted as $\nabla f(x) \in \mathbb{R}^d$, where

$$\nabla f(x)_i = \frac{\partial}{\partial x_i} f(x).$$

If f is not smooth, we can define the *subgradient* of f at $y \in \mathbb{R}^d$ as any vector $\alpha \in \mathbb{R}^d$ such that

$$f(x) \geq f(y) + \alpha^\top (x - y), \quad \forall\, x \in \mathbb{R}^d.$$

A function f is called *L-Lipschitz* iff

$$f(x) - f(y) \leq L\|x - y\|, \quad \forall\, x, y \in \mathbb{R}^d.$$

For a convex function f, we have

$$f(y) - f(x) \leq \nabla f(y)^\top (y - x) \leq \|\nabla f(y)\| \|x - y\|.$$

To finish this lecture, we review some equivalent definitions of convex functions. In the scalar case, if a function f is twice differentiable, then f is convex iff $f''(x) \geq 0$. This can be checked by using Taylor expansion.

$$f(x) = f(y) + f'(y)(x - y) + \frac{1}{2}(x - y)^2 f''(\xi).$$

If $x \in \mathbb{R}^d$ and f is twice differentiable, then f is convex iff $\nabla^2 f$, the Hessian matrix of f, satisfies $\nabla^2 f \succeq 0$, where the Hessian matrix is defined as

$$\nabla^2 f(x)_{ij} = \frac{\partial}{\partial x_i} \frac{\partial}{\partial x_j} f(x).$$

It is also equivalent to all eigenvalues of $\nabla^2 f$ being nonnegative.