Learning Theory

John Augustine (IIT Madras)

Krishna Palem (Rice University)

Source:
These slides are based on source material (including notation, concepts, and presentation approach) from "An introduction to Computational Learning theory" by Michael J. Kearns and Umesh V. Vazirani and "Analysis of Boolean Functions" by Ryan O'Donnell. The PAC learning game is borrowed from Jeremy Kun's blog www.jeremykun.com/primers

Acknowledgement: We thank Shreyas Shetty M (IIT Madras) for collaboration that resulted in these slides.

Topics in this Lecture

Overview

Background

- Boolean functions map n bit binary vectors to $\mathbb{R}$.
- An important special case is

$$f: \{-1, +1\}^n \to \{-1, +1\}$$

- Given a function, we can evaluate its value for different inputs.
- Fourier expansion gave us a perspective on the structure of the function.
  - It helps identify parts of the inputs that contribute more to the output when compared to other parts.

Motivation

- Let us now consider the inverse question.
  - Can we determine the function given its evaluation on a few inputs?

1. Can we do better if we know the nature of the function?
   - What if we know that the function is sparse ?
     - Of great interest in processing ``big data'' efficiently.
2. Why do we care about being able to do so ?

- We intend to discuss answers to these preceding questions in this lecture.
- We start with the question of why we care about this problem.

Why do we care ?

- Machine learning
  - Computers perform tasks without explicitly programming them.
- Examples include spam detection, face recognition, recommendation systems etc.
- Learning is achieved by presenting examples to an algorithm which then deciphers underlying patterns.
- The algorithm uses these patterns to predict the output on inputs that have not been seen so far.
  - Probability and statistics play an important role.

Why do we care ? (contd.)

- Our goal is to design efficient machine learning algorithms.
- To do so, we need to first answer a few questions.
  - What does it mean *formally* to learn to do a task ?
  - How many input examples are needed to learn a task?
- Design and analysis of algorithms need precise answers to the above questions.
- Hence we need mathematical models which model the notion of ***learning***.

PAC model

- Probably Approximately Correct (PAC) model is one such model, proposed by Leslie G. Valiant in 1984.
- This model resulted in a new field known as Computational Learning Theory or COLT.
- This work among other outstanding contributions were the basis for honouring Leslie Valiant with the Turing Award.

PAC learning model

Modeling Learning

- The PAC model represents the abstract notion of learning formally.
- This is analogous to algorithmic complexity where we modeled computation using a Turing machine.
- Recall that our computational models were used to answer interesting questions such as
  - What can be computed efficiently by a computer ?
  - What are the limitations of a computer ? and so on.
- PAC models do something similar in the learning context.

Introducing PAC Models

1. We will first define the constituents of the model and related notation.
2. We will then describe the model in a preliminary way.
3. This will lead us to some further parameters to be defined to explain the model in detail.
4. We will then discuss learning Boolean functions specifically in the PAC setting.

Definitions and notation

- Let $\boldsymbol{X}$ denote the instance space
  - the set of all encodings of objects or instances in the learners world.
    - These are examples to be used to learn the ``concept''.
  - One potential encoding of instances is to use binary strings.
- A concept is a subset of the instance space $\boldsymbol{X}$. Let $\boldsymbol{c}$ denote a concept over $\boldsymbol{X}$.
  - Since it is not interesting to use all $\boldsymbol{X}$ to learn the non-trivial case is when $\boldsymbol{c} \subset \boldsymbol{X}$.

Definition and notation (contd.)

- Alternatively $\boldsymbol{c}$ can be thought of as a map represented by an indicator function that identifies instances that belong to the concept.

$$\boldsymbol{c}(x) = \begin{cases} 1, & x \in \boldsymbol{c} \\ 0, & x \notin \boldsymbol{c} \end{cases}$$

- A concept class, is a collection of concepts denoted by $\boldsymbol{C}$.
- Further let $\boldsymbol{A}$ be our learning algorithm.

An example

- Let us consider the task of finding an interval that will classify points on a line into two groups.
  - Within the interval as 1.
  - Outside the interval as 0.
  - Note that our concept is the interval's definition the denoted by by $\boldsymbol{c}(x)$ earlier.

An example (contd.)

- For this example of learning an interval, our instance space $\boldsymbol{X} = \mathbb{R}$.
  - The intervals could arise anywhere along the real number line.
- An example concept $\boldsymbol{c}$ is the interval between 1 and 6 usually represented as (1,6).
- A concept class $\boldsymbol{C}$ is the set all intervals of given width, say 5.
  - $\boldsymbol{c}$ is one such concept in $\boldsymbol{C}$.

Model

- Our learning algorithm $\boldsymbol{A}$ can draw examples from a set of instances.
  - The examples are meant to learn an unknown target concept $\boldsymbol{c} \in \boldsymbol{C}$ that $\boldsymbol{A}$ must "learn".
    - The interval (1,6) in our example.
- The algorithm $\boldsymbol{A}$ outputs a hypothesis $\boldsymbol{h}$
  - $\boldsymbol{h}$ which is a concept over $\boldsymbol{X}$.
  - The hypothesis belongs to the concept class $\boldsymbol{C}$ $\boldsymbol{h} \in \boldsymbol{C}$.
    - This is some interval of width 5.
- Informally we say that $\boldsymbol{A}$ has learned $\boldsymbol{c}$ if it outputs a hypothesis $\boldsymbol{h} \in \boldsymbol{C}$ that is "close" to $\mathbf{c}$.

Model (contd.)

- What does it mean for hypothesis $\boldsymbol{h}$ to be close to $\boldsymbol{c}$?
- The "closeness" of $\boldsymbol{h}$ to $\boldsymbol{c}$ is measured quantitatively
  - Formally represented as $\boldsymbol{error(h)}$ between $\boldsymbol{h}$ and $\boldsymbol{c}$.
- The algorithm outputs a hypothesis concept $\boldsymbol{h}$ such that $\boldsymbol{error(h)}$ is small, say $\leq \epsilon$
  - ϵ is our tolerance.
- The confidence on the error being small should be high
  - say $\geq 1 - \delta$ where δ refers to a confidence parameter.

Example (contd.)

- Back to our interval learning example, we have
  - the instance space $\boldsymbol{X}$
  - the concept $\boldsymbol{c}$
  - and the concept class $\boldsymbol{C}$.
- Learning algorithm $\boldsymbol{A}$ has access to examples.
- Needs to compute a hypothesis which is close in terms of reducing error to the true concept.
  - Ideally a hypothesis that has the smallest error.
- Intuitively, in our example, this is an interval that is close to the true interval.

Modelling error and computation

- Let us now introduce probabilities into our model.
- Let $\boldsymbol{D}$ denote a fixed probability distribution over the instance space $\boldsymbol{X}$
  - Referred to as the target distribution.
- If $\boldsymbol{h}$ is any concept over $\boldsymbol{X}$ error between $\boldsymbol{h}$ and $\boldsymbol{c}$ is given by
  - $\boldsymbol{error}(\boldsymbol{h}) = \boldsymbol{P}_{\boldsymbol{x \sim D}}\,(\boldsymbol{h}(\boldsymbol{x}) \neq \boldsymbol{c}(\boldsymbol{x}))$ where $\boldsymbol{P}_{\boldsymbol{x \sim D}}$ denotes the probability of mismatch given $\boldsymbol{x}$ is drawn randomly from $\boldsymbol{D}$.
- Learning algorithm $\boldsymbol{A}$ has access to a procedure $\boldsymbol{EX}(\boldsymbol{c}, \boldsymbol{D})$ which magically produces instances $\boldsymbol{x}$ and its indicator value (whether it belongs to the concept) given $\boldsymbol{c}$ and $\boldsymbol{D}$
  - In this model $\boldsymbol{x}$ is drawn randomly and independently according to the distribution $\boldsymbol{D}$.

PAC model

For every concept $\boldsymbol{c} \in \boldsymbol{C}$, for every distribution $\boldsymbol{D}$ on $\boldsymbol{X}$, and for all $\mathbf{0} < \boldsymbol{\epsilon} < \frac{\mathbf{1}}{\mathbf{2}}$, $\mathbf{0} < \boldsymbol{\delta} < \frac{\mathbf{1}}{\mathbf{2}}$, if $\boldsymbol{A}$ is given access to $\boldsymbol{EX}(\boldsymbol{c}, \boldsymbol{D})$ and inputs $\boldsymbol{\epsilon}, \boldsymbol{\delta}$, then with probability at least $\mathbf{1} - \boldsymbol{\delta}$, $\boldsymbol{A}$ outputs a hypothesis concept $\boldsymbol{h} \in \boldsymbol{C}$, satisfying $\boldsymbol{error}(\boldsymbol{h}) \leq \boldsymbol{\epsilon}$.

- If $\boldsymbol{A}$ runs in time polynomial in $\frac{\mathbf{1}}{\boldsymbol{\epsilon}}, \frac{\mathbf{1}}{\boldsymbol{\delta}}$, then we say that $\boldsymbol{C}$ is efficiently PAC learnable.

PAC learning of Boolean functions

So far..

- We have described the PAC model for general functions.
- We have a framework to argue about the efficiency of learning algorithms, assuming we have set up the instance space, concept class and so on.
- We now go ahead and consider our goal of learning Boolean functions.

Boolean functions Recap

$$f\colon \{-1,+1\}^n \to \{-1,+1\}$$

- f maps an n bit string to ± 1.
- f can be expressed in terms of its Fourier expansion.

$$\boldsymbol{f(x) = \sum_{S\subseteq[1\cdots n]} \hat{f}(S)x^S}$$

$$\boldsymbol{x^S = \prod_{i\in S} x_i}$$

To Understand PAC learning of Boolean Functions

- We will first map our definitions from the general PAC model to the one for Boolean functions.
- We then define what it means when we say that a Boolean function is efficiently PAC ``learnable''.
- Finally we state and prove a theorem about efficient PAC learnability of sparse Boolean functions.

Some results and definitions that we will use..

- Let $\mathcal{F}$ be a collection of subsets $S \subseteq [n]$.
- We say that the Fourier spectrum of $f: \{-1,1\}^n \to \mathbb{R}$ is ϵ - *concentrated* on $\mathcal{F}$ provided
  - $\sum_{\substack{S \subseteq [n] \\ S \notin \mathcal{F}}} \hat{f}(S)^2 \leq \epsilon$
- Note that $f(x) = \sum_{S \subseteq [1 \cdots n]} \hat{f}(S) x^S$, was a weighted sum of the Fourier coefficients $\hat{f}(S)$.

Some results and definitions that we will use (Contd.)

- The definition of ϵ-concentration says that the squared sum of coefficients on a given collection is small.
- Consequently ϵ – concentration captures the idea that the remaining collection of subsents $\mathcal{F}$ contains all the high value cases.

Mapping definitions from the PAC model

- Earlier, we had defined an error function error(h) to measure the error between the hypothesis h and the true concept c.
- Building on this we define the distance(error) between the hypothesis h and the true function f as
  - $\boldsymbol{dist(h,f)} = \mathbb{E}\left[\left(\boldsymbol{h(x)} - \boldsymbol{f(x)}\right)^2\right]$
  - Note that the probability is over an x chosen uniformly at random from $\{-1,1\}^n$.

Learning Boolean Functions

- Now, the goal of our algorithm is to figure out (learn) $\boldsymbol{f}$ given some of its evaluations using inputs drawn uniformly at random.
- We know that a Boolean function $\boldsymbol{f}: \{-1, +1\}^n \rightarrow \mathbb{R}$ can be represented as

$$\boldsymbol{f}(x) = \sum_{S \subseteq [1 \cdots n]} \hat{\boldsymbol{f}}(S) x^S$$

- The Fourier coefficients $\hat{\boldsymbol{f}}(\boldsymbol{S})$ on all possible subsets S, specify the function completely.
- There are 2^n subsets S
  - Computing all of them is expensive.

A significant result

Theorem. Assume learning algorithm A has acces to examples drawn uniformly at random for

$$f : \{-1,1\}^n \to \{-1,1\}$$

and suppose that A can (using an oracle) identify a collection $\boldsymbol{\mathcal{F}}$ of subsets on which f's Fourier spectrum is $\boldsymbol{\frac{\epsilon}{2}}$ - concentrated, then using poly($|\mathcal{F}|$, n, $1/\epsilon$) additional time, A can output a hypothesis h that is ϵ-close to f with high probability.

What does it mean ?

- This result tells us that if the target function f is sparse i.e. has its spectrum concentrated on a collection of sets $\mathcal{F}$, we can efficiently "learn" f.
- The "learning" is restricted to $\mathcal{F}$ and we need to expend effort limited to that collection.
- With high probability we output a good hypothesis
  - This is the (1-δ) confidence factor in the definition of general PAC setting.
- Summary: knowledge that f is sparse gives us an efficient algorithm to learn f.

Let us get a feel for the result..

- By hypothesis, we know that the function is $\frac{\epsilon}{2}$ concentrated on some collection $\mathcal{F}$.
- We only need to estimate the Fourier coefficients on these sets, because the other subsets are guaranteed to have "small" coefficients.
- This gives hope of an designing an efficient learning algorithm if we limit our attention to the set $\mathcal{F}$.

Let us get a feel for the result (Contd.)

- Using the PAC framework, we have access to a procedure similar to $EX(c, D)$ which generates examples.
- In addition, by hypothesis, we can determine the Fourier coefficients on the sets of interest $\mathcal{F}$.
  - We can then construct an "approximation" of the function.
  - Two questions arise.
    - How to construct the approximation ?
    - How good is the approximation ?

Constructing the Approximation

- The Fourier coefficient $\hat{f}(S)$ is given by $\hat{f}(S) = E_x[f(x)x^S]$
  - The expectation is over a uniform distribution on $\{-1,1\}^n$.
- We can calculate this expectation given all 2^n inputs.
  - Using all 2^n possible inputs is expensive.
- A reasonable approximation is to average the values of $f(x)x^S$ on the samples that we have.
- If we can do this, the remaining question then is, how close an approximation of $f(x)$ can we obtain ?

Switching to an example

Constructing the Approximation

- The Fourier coefficient $\hat{f}(S)$ is given by $\hat{f}(S) = E_x[f(x)x^S]$
  - The expectation is over a uniform distribution on $\{-1,1\}^n$.
- We can calculate this expectation given all 2^n inputs.
  - Using all 2^n possible inputs is expensive.
- A reasonable approximation is to average the values of $f(x)x^S$ on the samples that we have.
- If we can do this, the remaining question then is, how close an approximation of $f(x)$ can we obtain ?

Boolean functions Recap

$$\boldsymbol{f}: \{-1,+1\}^n \to \{-1,+1\}$$

- $\boldsymbol{f}$ maps an n bit string to ± 1.
- $\boldsymbol{f}$ can be expressed in terms of its Fourier expansion.

$$\boldsymbol{f(x)} = \sum_{\boldsymbol{S \subseteq [1 \cdots n]}} \boldsymbol{\hat{f}(S) x^S}$$

$$\boldsymbol{x^S} = \prod_{\boldsymbol{i \in S}} \boldsymbol{x_i}$$

Constructing the approximation

- We saw how to construct $\boldsymbol{f}(\boldsymbol{x})$ given $\hat{\boldsymbol{f}}(\boldsymbol{S})$ on all possible subsets.
- Now, consider the complimentary question, i.e. determining $\hat{\boldsymbol{f}}(\boldsymbol{S})$ given $\boldsymbol{f}(\boldsymbol{x})$.
- It can be shown that $\hat{\boldsymbol{f}}(\boldsymbol{S}) = \boldsymbol{E}_{\boldsymbol{x}\sim\{-\mathbf{1},\mathbf{1}\}^{\boldsymbol{n}}}\left[\boldsymbol{f}(\boldsymbol{x})\boldsymbol{x}^{\boldsymbol{S}}\right]$
  - Given a set S we can multiply the function value $\boldsymbol{f}(\boldsymbol{x})$ with $\boldsymbol{x}^{\boldsymbol{S}}$ and take the expectation over all possible $\boldsymbol{x}$ values, i.e. over all possible 2^n input combinations of $\boldsymbol{x}$.
    - Interpret $\boldsymbol{E}_{\boldsymbol{x}\sim\{-\mathbf{1},\mathbf{1}\}^{\boldsymbol{n}}}$ to be this expectation or averaging over inputs $\{-\mathbf{1},\mathbf{1}\}^{\boldsymbol{n}}$of length *n* drawn uniformly at random.
  - Refer to Proposition 1.8,(Chapter 1), "Analysis of Boolean functions", Ryan O'Donnell for a proof.

Constructing the Approximation

- Using all 2^n possible inputs is expensive.
- A reasonable approximation is to average the values of the product $f(x)x^S$ on the examples that we have.
- We denote this approximation of $\hat{\boldsymbol{f}}(\boldsymbol{S})$ by $\tilde{\boldsymbol{f}}(\boldsymbol{S})$.
- If we can do this, the remaining question then is, how close an approximation of $f(x)$ can we obtain ?

An example function

- For convenience, we give an example Boolean function over $\mathbb{R}$ rather than the restricted ± 1.
- Consider $\boldsymbol{f}: \{-1,1\}^3 \longrightarrow \mathbb{R}$
  - Let $\hat{f}(\{1\}) = 0.86$ and $\hat{f}(\{2\}) = 0.5$.
  - *All the other subsets have zero coefficients*.
- We will access example inputs uniformly at random.
- Say we get three examples.
  - $x_1 = \{-1, -1, 1\}\ f(x_1) = -1.36$
  - $x_2 = \{-1, -1, 1\}\ f(x_2) = -1.36$
  - $x_3 = \{1, -1, 1\}\ \ f(x_3) = 0.36$

An example function

- ϵ – concentration on a collection is defined as sum of $\hat{f}(S)^2$ on all sets *outside* the collection.
- Note that the Fourier spectrum of f is concentrated on $S = \{1\}$.
  - $\frac{\epsilon}{2} \leq (0.5)^2 = 0.25$
  - $\epsilon \leq 0.5$
- Under ϵ-concentration only one set S = $\{1\}$ qualifies.
  - Our other candidate, $\{2\}$ has a concentration $\frac{\epsilon}{2} = (0.86)^2 = 0.75$, resulting in $\epsilon = 1.5 > 0.5,$ outside our tolerable error.
- Hence we approximate by estimating $\hat{f}(\{1\})$.

An example function

- As claimed earlier on precisely computing $\hat{f}(\{1\})$ is not possible.
  - This is the case unless we could look at all examples.
  - Notice however, we are trying to approximate using a few random examples.
  - What form would this take ?

An example function (contd.)

- We estimate $\hat{f}(\{1\})$ using $(x_1, f(x_1))$, $(x_2, f(x_2))$ and $(x_3, f(x_3))$.
- For S $= \{1\}$, $x^S = 1$
  - if the bit at position 1 is 1
  - and -1 otherwise.
- For the examples.
  - $x_1 = \{-1, -1, 1\}\ f(x_1) = -1.36$, $f(x_1)x_1^{\{1\}} = -1.36 * -1 = 1.36$
  - $x_2 = \{-1, -1, 1\}\ f(x_2) = -1.36$, $f(x_2)x_2^{\{1\}} = -1.36 * -1 = 1.36$
  - $x_3 = \{+1, -1, 1\}\ f(x_3) = +0.36$, $f(x_3)x_3^{\{1\}} = -0.36 * 1 = -0.36$

An example function (contd.)

- For the given samples $f(x)x^{\{1\}}$ evaluates to 1.36, 1.36 and -0.36, resulting in an average of 0.786.
- This gives us an approximation $\tilde{f}(\{1\}) = 0.786$
- Once we have $\tilde{f}(\{1\})$
  - we can just approximate f by h
  - where h is now $\tilde{f}(\{1\})x^{\{1\}}$.

An example function (contd.)

- $h(x) = \tilde{f}(\{1\})x^{\{1\}}$
- The expected square error is

$$\mathbb{E}[(f-h)^2] = \mathbb{E}\left[\left(\hat{f}(\{1\})x^{\{1\}} + \hat{f}(\{2\})x^{\{2\}}\right) - \tilde{f}(\{1\})x^{\{1\}}\right)^2\right]$$
$$= \hat{f}(\{1\})^2 + \hat{f}(\{2\})^2 - \tilde{f}(\{1\})^2$$
$$= (0.5)^2 + (0.86)^2 - (0.786)^2 = 0.3822 \leq \epsilon = 0.5$$

- Notice that the theorem promised an error less than 0.5 and we observe that.
- The $(1-\delta)$ parameter is captured (indirectly) by the examples that went into the algorithm.

.

Remarks

- The value of the estimate is dependent on the examples drawn.
- The nature of guarantees that we have is that on expectation, we are “close” to the true value.
- We have defined the error here to be ``slightly'' different from the definition of distance between two Boolean functions.
- We will give a formal proof of the Theorem in the next lecture.

Thank You!