Computational Learning Theory

Nakul Gopalan

Slides based on material from Vivek Srikumar, Dan Roth, Avrim Blum, Tom Mitchell and others

Contents

- Computational Learning theory
- Probably Approximately correct
- Vapnik-Chervonenkis (VC) dimension

Computational Learning Theory

- Large sub-field
- Conference on Learning theory
- What problems are solvable?
- How many samples do we need to solve a novel problem?
- How well will the algorithm generalize?

- Slides largely from materials developed by Vivek Srikumar.

PAC learning

- For batch learning
- Asks how well will your *learner* generalize to unsee data in the wild

Problem setup

- Instance Space: X, the set of examples
- Concept Space: C, the set of possible target functions: $f \in C$ is the hidden target function
  - Example: all n-conjunctions; all n-dimensional linear functions…
- Hypothesis Space: H, set of possible hypotheses
  - Set of functions the learning algorithm considers
  - Different from C, whose form might not be known!!
- Training instances: $S \times \{-1,1\}$: positive and negative examples of the target concept. (S is a finite subset of X)
  - $(x_1, f(x_1)), (x_2, f(x_2)), (x_3, f(x_3)), \ldots\ldots\ldots\ldots (x_n, f(x_n))$
- What we want: A hypothesis $h \in H$ such that $h(x) = f(x)$
  - For x in S???
  - For x in X???

Problem setup

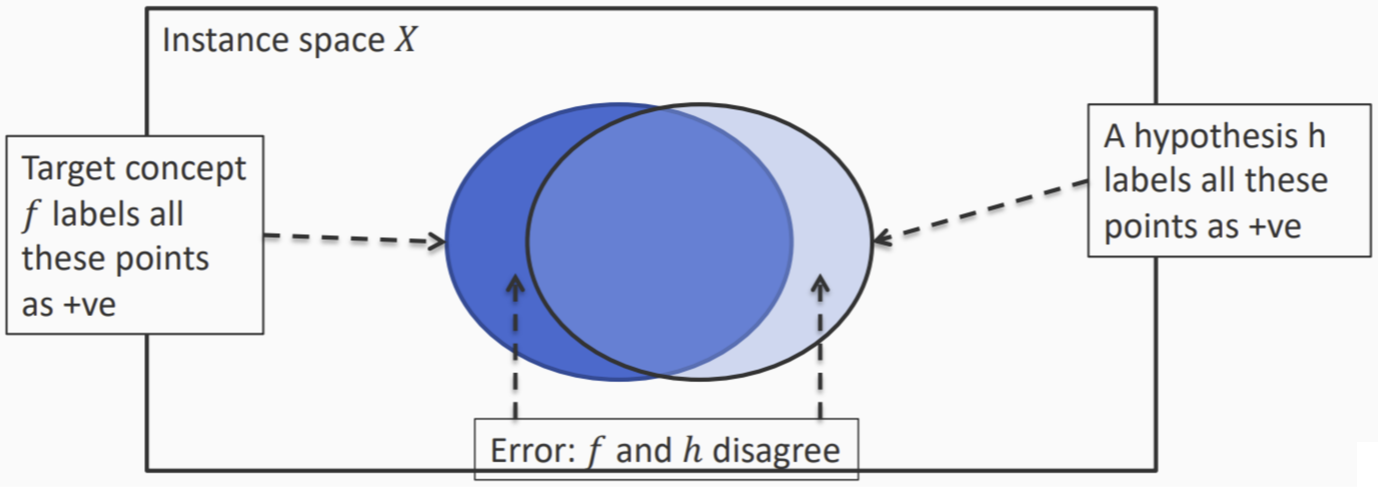
- Instance Space: X, the set of examples
- Concept Space: C, the set of possible target functions: $f \in C$ is the hidden target function
  - Example: all n-conjunctions; all n-dimensional linear functions…
- Hypothesis Space: H, set of possible hypotheses
  - Set of functions the learning algorithm considers
- Training instances: $S \times \{-1,1\}$: positive and negative examples of the target concept. (S is a finite subset of X)
  - S sampled from X using a distribution D
- What we want: A hypothesis $h \in H$ such that $h(x) = f(x)$
  - Evaluation on more samples from X using D

True Error of a hypothesis (not empirical)

Definition:

Given a distribution D over examples, the error of a hypothesis h with respect to a target concept f is:

$$E_D(h) = Pr_D[h(x) \neq f(x)]$$



Contents

- Computational Learning theory
- Probably Approximately correct
- Vapnik-Chervonenkis (VC) dimension

Theoretical Questions?

- Can we describe or bound the true error (E_D) given the empirical error (E_S)?
- Is a concept class C learnable?
- Is it possible to learn C using only the functions in H using the supervised protocol?
- How many examples does an algorithm need to guarantee good performance?

Expectations of learning

- We cannot expect a learner to learn a concept exactly
  - There will generally be multiple concepts consistent with the available data (which represent a small fraction of the available instance space)
  - Unseen examples could potentially have any label
  - Let's "agree" to misclassify uncommon examples that do not show up in the training set
- We cannot always expect to learn a close approximation to the target concept
  - Sometimes (hopefully only rarely) the training set will not be representative (will contain uncommon examples)

What we can expect

A learner will with high probability learn a close approximation of the target concept.

Probably approximately correct???

- Provide small parameters ε and δ,
- With probability at least $1 - \delta$, a learner produces a hypothesis with error at most ε
- The only reason we can hope for this is the consistent distribution assumption

PAC definition

Consider a concept class C defined over an instance space X (containing instances of length n), and a learner L using a hypothesis space H

The concept class C is PAC learnable by L using H if:

for all $f \in C$,

for all distribution D over X and fixed $0 < \varepsilon, \delta < 1$

given m examples sampled independently according to D, with probability at least $(1 - \delta)$, the algorithm L produces a hypothesis $h \in H$ that has error at most ε,

where m is polynomial in $1/\varepsilon$, $1/\delta$, n and size(H).

PAC definition

Consider a concept class C defined over an instance space X (containing instances of length n), and a learner L using a hypothesis space H

The concept class C is PAC learnable by L using H if:

for all $f \in C$,

for all distribution D over X and fixed $0 < \varepsilon , \delta < 1$

given m examples sampled independently according to D, with probability at least $(1 - \delta)$, the algorithm L produces a hypothesis $h \in H$ that has error at most ε,

where m is polynomial in $1/\varepsilon$, $1/\delta$, n and size(H).

Given a small number of examples

PAC definition

Consider a concept class *C* defined over an instance space *X* (containing instances of length *n*), and a learner *L* using a hypothesis space *H*

The concept class *C* is PAC learnable by *L* using *H* if:

for all $f \in C$,

for all distribution *D* over *X* and fixed $0 < \varepsilon, \delta < 1$

With High Probability

given *m* examples sampled independently according to *D*, with probability at least $(1 - \delta)$, the algorithm *L* produces a hypothesis $h \in H$ that has error at most ε,

where *m* is polynomial in $1/\varepsilon$, $1/\delta$, n and size(*H*).

PAC definition

Consider a concept class *C* defined over an instance space *X* (containing instances of length *n*), and a learner *L* using a hypothesis space *H*

The concept class *C* is PAC learnable by *L* using *H* if:

for all $f \in C$,

for all distribution *D* over *X* and fixed $0 < \varepsilon, \delta < 1$

given *m* examples sampled independently according to *D*, with probability at least $(1 - \delta)$, the algorithm *L* produces a hypothesis $h \in H$ that has error at most ε,

where *m* is polynomial in $1/\varepsilon$, $1/\delta$, n and size(*H*).

The learner will produce a "good enough" classifier

PAC definition

Consider a concept class *C* defined over an instance space *X* (containing instances of length *n*), and a learner *L* using a hypothesis space *H*

The concept class *C* is PAC learnable by *L* using *H* if:

for all $f \in C$,

for all distribution *D* over *X* and fixed $0 < \varepsilon , \delta < 1$

given *m* examples sampled independently according to *D*, with probability at least $(1 - \delta)$, the algorithm *L* produces a hypothesis $h \in H$ that has error at most ε,

where *m* is polynomial in $1/\varepsilon$, $1/\delta$, n and size(*H*).

The concept class *C* is efficiently learnable if *L* can produce the hypothesis in time that is polynomial in $1/\varepsilon$, $1/\delta$, n and size(*H*).

PAC Learnability

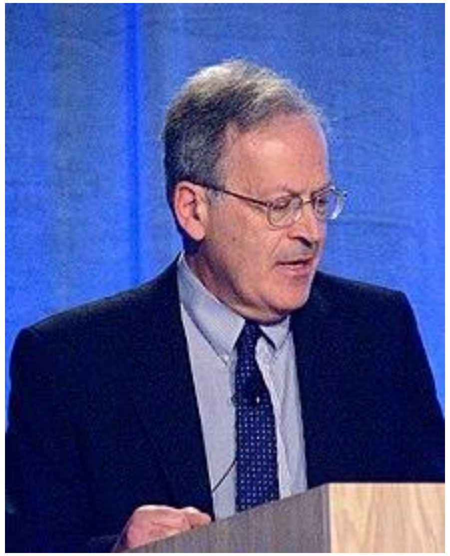
- Imposes two limitations
  - Polynomial sample complexity (information theoretic constraint)
    - Is there enough information in the sample to distinguish a hypothesis h that approximates f ?
  - Polynomial time complexity (computational complexity)
    - Is there an efficient algorithm that can process the sample and produce a good hypothesis h ?

- To be PAC learnable, there must be a hypothesis $h \in H$ with arbitrary small error for every $f \in C$. We assume $H \supseteq C$. (Properly PAC learnable if $H=C$)

- Worst Case definition: the algorithm must meet its accuracy
  - for every distribution (The distribution free assumption)
  - for every target function *f* in the class *C*

Results with PAC learnability

- General conjunctions are PAC learnable!!!
  - a ∧ b ∧ c ∧ d ∧ e
  - Sample complexity linear in in *n* the number of variables
- 3-CNFs are PAC learnable
  - Example – (*a* ∨ *b* ∨ *c*) ∧ (*x* v *y* v *z*)
  - Sample complexity polynomial in *n* the number of 3 conjuncts
- General Boolean functions not PAC learnable
  - Number of possible Boolean functions with n variables: 2^{2^n}
  - Size of *H* is super-exponential
- Turing Award for Leslie Valiant ☺



Negative result strategies

Generally two types of non-learnability results

1. Complexity Theoretic (computational complexity bad) –

   Showing that various concepts classes cannot be learned, based on well accepted assumptions from computational complexity theory – Takes the form "A concept class C cannot be learned unless P=NP"

2. Information Theoretic (sample complexity bad) –

   The concept class is sufficiently rich that a polynomial number of examples may not be sufficient to distinguish a particular target concept – The proof typically shows that a given class cannot be learned by algorithms using hypotheses from the same class. (Is this always a problem?)

Contents

- Computational Learning theory
- Probably Approximately correct
- Vapnik-Chervonenkis (VC) dimension

Problem

- After training a model we have some training error
- How do we know what kind of test error to expect?
- How do we know which of the possible models is the best?
- How do we know if one hypothesis class is better than the other?

A Measure of Model Complexity

- Pick *n* points
- Assign labels to them randomly (+ve and –ve)
- Can our hypothesis class separate the data points?

Two points and linear hypothesis class

- Can a linear classifier split any two points?

Two points and linear hypothesis class

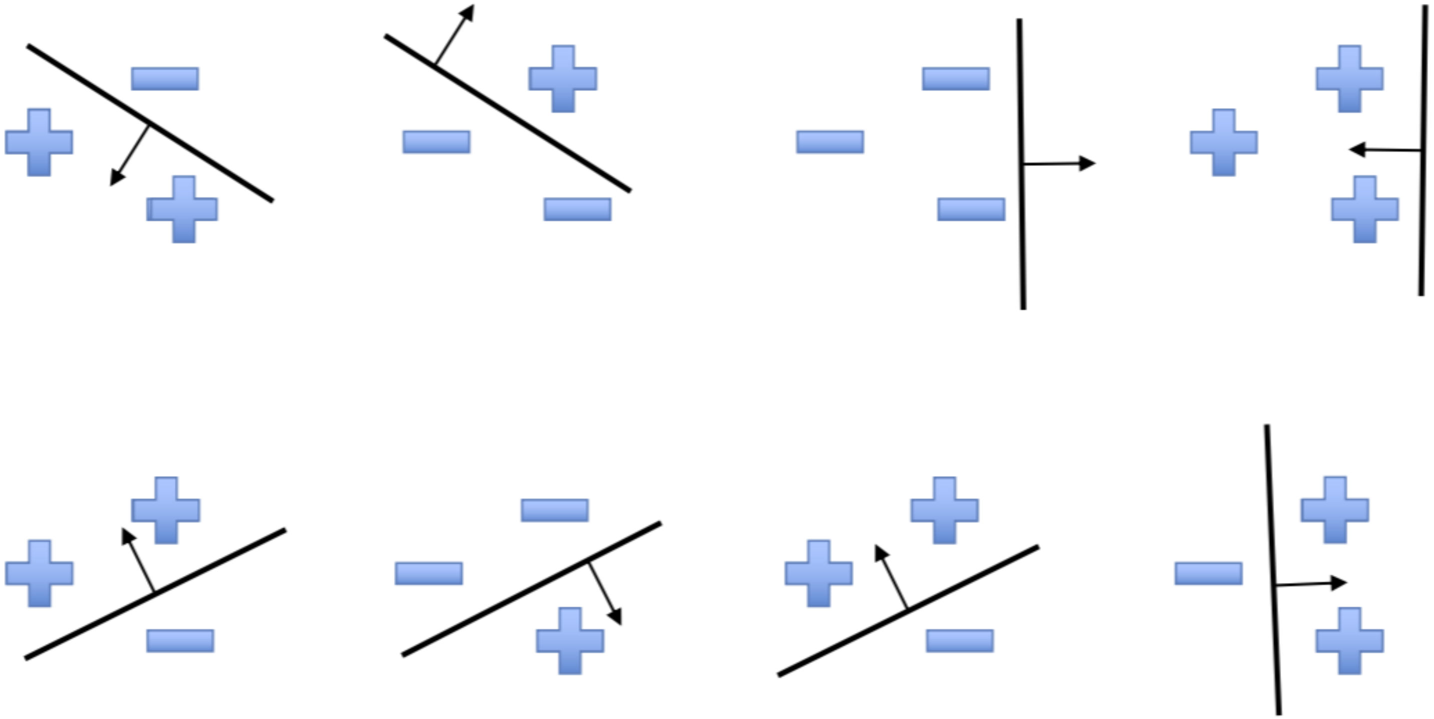
- We say that linear functions are expressive enough to *shatter* two points

Shattering

Definition: A set S of examples is shattered by a set of functions H if for every partition of the examples in S into positive and negative examples there is a function in H that gives exactly these labels to the examples

Intuition: A rich set of functions shatters large sets of points

Three points and linear hypothesis class



Slide inspired from those of Geoff Hinton and Byron Boots

Four or more points??

Vapnik-Chervonenkis (VC) dimension

Definition: The VC dimension of hypothesis space H over instance space X is the size of the largest finite subset of X that is shattered by H

- If there exists any subset of size d that can be shattered, VC(H) >= d – Even one subset will do
- If no subset of size d can be shattered, then VC(H) < d

Example VC dimensions

Concept Class	VC dimension
Linear threshold unit in d dimensions	d + 1
Neural networks	Number of parameters
1 nearest neighbor	Infinite
Sine Wave / Curve	Infinite

VC dimension

- VC dimensions a measure of richness or size of the *H*
- If we have *m* examples, then with probability $1 - \delta$, the true error of a hypothesis *h* with training error $E_S(h)$ is bounded by:

$$E_S(h) \leq E_D(h) + \sqrt{\frac{VC(H)\left(\ln\frac{2m}{VC(H)} + 1\right) + \ln\frac{4}{\delta}}{m}}$$

Take away

- Probably approximately correct: Tells us if a concept class is learnable with high probability and with low generalization error with few examples.
- Allows us to define learnable concepts and distinguish efficient algorithms
- VC dimension presents a measure of the model/ hypothesis complexity
- PAC learning provides a way to create a bound on the test error using VC dimensions of the hypothesis class.