Computational Learning Theory: Positive and negative learnability results

Machine Learning

THE UNIVERSITY OF UTAH

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

Computational Learning Theory

- The Theory of Generalization
- Probably Approximately Correct (PAC) learning
- Positive and negative learnability results
- Agnostic Learning
- Shattering and the VC dimension

This lecture: Computational Learning Theory

- The Theory of Generalization
- Probably Approximately Correct (PAC) learning
- Positive and negative learnability results
- Agnostic Learning
- Shattering and the VC dimension

$$m > \frac{1}{\epsilon}\left(\ln(|H|) + \ln\frac{1}{\delta}\right)$$

What can be learned

General conjunctions are PAC learnable

- $|H|$ = Number of conjunctions of n variables = ??

$$m > \frac{1}{\epsilon}\left(\ln(|H|) + \ln\frac{1}{\delta}\right)$$

What can be learned

General conjunctions are PAC learnable

- $|H|$ = Number of conjunctions of n variables = 3^n

$$\ln|H| = n\ln(3)$$

$$m > \frac{1}{\epsilon}\left(\ln(|H|) + \ln\frac{1}{\delta}\right)$$

What can be learned

General conjunctions are PAC learnable

- $|H|$ = Number of conjunctions of n variables = 3^n

$$\ln|H| = n\ln(3)$$

- Number of examples needed $m > \frac{1}{\epsilon}\left(n\ln(3) + \ln\frac{1}{\delta}\right)$

$$m > \frac{1}{\epsilon}\left(\ln(|H|) + \ln\frac{1}{\delta}\right)$$

What can be learned

General conjunctions are PAC learnable

- $|H|$ = Number of conjunctions of n variables = 3^n

$$\ln|H| = n\ln(3)$$

- Number of examples needed $m > \frac{1}{\epsilon}\left(n\ln(3) + \ln\frac{1}{\delta}\right)$

- If we want to guarantee a 95% chance of learning a hypothesis of at least 90% accuracy, with $n = 10$ Boolean variables, we need $\mathrm{m} > \frac{\left(\ln\left(\frac{1}{0.05}\right)+10\ln(3)\right)}{0.1} = 140$ examples

$$m > \frac{1}{\epsilon}\left(\ln(|H|) + \ln\frac{1}{\delta}\right)$$

What can be learned

General conjunctions are PAC learnable

- $|H|$ = Number of conjunctions of n variables = 3^n

$$\ln|H| = n\ln(3)$$

- Number of examples needed $m > \frac{1}{\epsilon}\left(n\ln(3) + \ln\frac{1}{\delta}\right)$

- If we want to guarantee a 95% chance of learning a hypothesis of at least 90% accuracy, with $n = 10$ Boolean variables, we need $\mathrm{m} > \frac{\left(\ln\left(\frac{1}{0.05}\right)+10\ln(3)\right)}{0.1} = 140$ examples

- If $n = 100$, this goes to 1129. (linearly increases with n)

$$m > \frac{1}{\epsilon}\left(\ln(|H|) + \ln\frac{1}{\delta}\right)$$

What can be learned

General conjunctions are PAC learnable

– $|H|$ = Number of conjunctions of n variables = 3^n

$$\ln|H| = n\ln(3)$$

– Number of examples needed $m > \frac{1}{\epsilon}\left(n\ln(3) + \ln\frac{1}{\delta}\right)$

- If we want to guarantee a 95% chance of learning a hypothesis of at least 90% accuracy, with $n = 10$ Boolean variables, we need $\mathrm{m} > \frac{\left(\ln\left(\frac{1}{0.05}\right)+10\ln(3)\right)}{0.1} = 140$ examples

- If $n = 100$, this goes to 1129. (linearly increases with n)

These results hold for ***any*** consistent learner

$$m > \frac{1}{\epsilon}\left(\ln(|H|) + \ln\frac{1}{\delta}\right)$$

What can be learned

General conjunctions are PAC learnable

- $|H|$ = Number of conjunctions of n variables = 3^n

$$\ln|H| = n\ln(3)$$

- Number of examples needed $m > \frac{1}{\epsilon}\left(n\ln(3) + \ln\frac{1}{\delta}\right)$

- If we want to guarantee a 95% chance of learning a hypothesis of at least 90% accuracy, with $n = 10$ Boolean variables, we need $\mathrm{m} > \frac{\left(\ln\left(\frac{1}{0.05}\right) + 10\ln(3)\right)}{0.1} = 140$ examples
- If $n = 100$, this goes to 1129. (linearly increases with n)
- Increasing the confidence to 99% will cost 1145 examples (logarithmic with δ)

These results hold for ***any*** consistent learner

$$m > \frac{1}{\epsilon}\left(\ln(|H|) + \ln\frac{1}{\delta}\right)$$

What can be learned

3-CNF

$$(l_{11} \vee l_{12} \vee l_{13}) \wedge (l_{21} \vee l_{22} \vee l_{23})$$

Subset of CNFs: Each conjunct can have at most three literals (i.e a variable or its negation)

$$m > \frac{1}{\epsilon}\left(\ln(|H|) + \ln\frac{1}{\delta}\right)$$

What can be learned

3-CNF

$$(l_{11} \vee l_{12} \vee l_{13}) \wedge (l_{21} \vee l_{22} \vee l_{23})$$

Subset of CNFs: Each conjunct can have at most three literals (i.e a variable or its negation)

$$m > \frac{1}{\epsilon}\left(\ln(|H|) + \ln\frac{1}{\delta}\right)$$

What can be learned

3-CNF

$$(l_{11} \vee l_{12} \vee l_{13}) \wedge (l_{21} \vee l_{22} \vee l_{23})$$

Subset of CNFs: Each conjunct can have at most three literals (i.e a variable or its negation)

$$m > \frac{1}{\epsilon}\left(\ln(|H|) + \ln\frac{1}{\delta}\right)$$

What can be learned

3-CNF $$(l_{11} \vee l_{12} \vee l_{13}) \wedge (l_{21} \vee l_{22} \vee l_{23})$$

Subset of CNFs: Each conjunct can have at most three literals (i.e a variable or its negation)

What is the sample complexity?

That is, if we had a consistent learner, how many examples will it need to guarantee PAC learnability?

$$m > \frac{1}{\epsilon}\left(\ln(|H|) + \ln\frac{1}{\delta}\right)$$

What can be learned

3-CNF $\quad (l_{11} \vee l_{12} \vee l_{13}) \wedge (l_{21} \vee l_{22} \vee l_{23})$

Subset of CNFs: Each conjunct can have at most three literals (i.e a variable or its negation)

What is the sample complexity?

That is, if we had a consistent learner, how many examples will it need to guarantee PAC learnability?

We need the size of the hypothesis space. How many 3CNFs are there?

$$m > \frac{1}{\epsilon}\left(\ln(|H|) + \ln\frac{1}{\delta}\right)$$

What can be learned

3-CNF $\quad (l_{11} \vee l_{12} \vee l_{13}) \wedge (l_{21} \vee l_{22} \vee l_{23})$

Subset of CNFs: Each conjunct can have at most three literals (i.e a variable or its negation)

What is the sample complexity?

That is, if we had a consistent learner, how many examples will it need to guarantee PAC learnability?

We need the size of the hypothesis space. How many 3CNFs are there?

- Number of conjuncts = $O\left((2n)^3\right)$

$$m > \frac{1}{\epsilon}\left(\ln(|H|) + \ln\frac{1}{\delta}\right)$$

What can be learned

3-CNF

$$(l_{11} \lor l_{12} \lor l_{13}) \land (l_{21} \lor l_{22} \lor l_{23})$$

Subset of CNFs: Each conjunct can have at most three literals (i.e a variable or its negation)

What is the sample complexity?

That is, if we had a consistent learner, how many examples will it need to guarantee PAC learnability?

We need the size of the hypothesis space. How many 3CNFs are there?

- Number of conjuncts = $O\left((2n)^3\right)$
- A 3-CNF is a conjunction with these many variables.

$$m > \frac{1}{\epsilon}\left(\ln(|H|) + \ln\frac{1}{\delta}\right)$$

What can be learned

3-CNF

$$(l_{11} \vee l_{12} \vee l_{13}) \wedge (l_{21} \vee l_{22} \vee l_{23})$$

Subset of CNFs: Each conjunct can have at most three literals (i.e a variable or its negation)

What is the sample complexity?

That is, if we had a consistent learner, how many examples will it need to guarantee PAC learnability?

We need the size of the hypothesis space. How many 3CNFs are there?

- Number of conjuncts = $O\left((2n)^3\right)$
- A 3-CNF is a conjunction with these many variables.
- $|H|$ = Number of 3-CNFs = $O\left(2^{(2n)^3}\right)$

$$m > \frac{1}{\epsilon}\left(\ln(|H|) + \ln\frac{1}{\delta}\right)$$

What can be learned

3-CNF

$$(l_{11} \vee l_{12} \vee l_{13}) \wedge (l_{21} \vee l_{22} \vee l_{23})$$

Subset of CNFs: Each conjunct can have at most three literals (i.e a variable or its negation)

What is the sample complexity?

That is, if we had a consistent learner, how many examples will it need to guarantee PAC learnability?

We need the size of the hypothesis space. How many 3CNFs are there?

- Number of conjuncts = $O\left((2n)^3\right)$
- A 3-CNF is a conjunction with these many variables.
- $|H|$ = Number of 3-CNFs = $O\left(2^{(2n)^3}\right)$

- $\log(|H|) = O(n^3)$

$$m > \frac{1}{\epsilon}\left(\ln(|H|) + \ln\frac{1}{\delta}\right)$$

What can be learned

3-CNF

$$(l_{11} \vee l_{12} \vee l_{13}) \wedge (l_{21} \vee l_{22} \vee l_{23})$$

Subset of CNFs: Each conjunct can have at most three literals (i.e a variable or its negation)

What is the sample complexity?

That is, if we had a consistent learner, how many examples will it need to guarantee PAC learnability?

We need the size of the hypothesis space. How many 3CNFs are there?

- Number of conjuncts = $O\left((2n)^3\right)$
- A 3-CNF is a conjunction with these many variables.
- $|H|$ = Number of 3-CNFs = $O\left(2^{(2n)^3}\right)$

- $\log(|H|) = O(n^3)$

$\log(|H|)$ is polynomial in n
$\Rightarrow$ the sample complexity is also polynomial in n

$$m > \frac{1}{\epsilon}\left(\ln(|H|) + \ln\frac{1}{\delta}\right)$$

What can be learned

3-CNF

$$(l_{11} \vee l_{12} \vee l_{13}) \wedge (l_{21} \vee l_{22} \vee l_{23})$$

Subset of CNFs: Each conjunct can have at most three literals (i.e a variable or its negation)

What is the sample complexity?

That is, if we had a consistent learner, how many examples will it need to guarantee PAC learnability?

We need the size of the hypothesis space. How many 3CNFs are there?

- Number of conjuncts = $O\left((2n)^3\right)$
- A 3-CNF is a conjunction with these many variables.
- $|H|$ = Number of 3-CNFs = $O\left(2^{(2n)^3}\right)$

- $\log(|H|) = O(n^3)$

$\log\,(|H|)$ is polynomial in n
$\Rightarrow$ the sample complexity is also polynomial in n

For PAC learnability, we still need an efficient algorithm that will find a consistent hypothesis.
Exercise: Find one

$$m > \frac{1}{\epsilon}\left(\ln(|H|) + \ln\frac{1}{\delta}\right)$$

What can be learned

General Boolean functions

- How many Boolean functions exist with n variables?

$$m > \frac{1}{\epsilon}\left(\ln(|H|) + \ln\frac{1}{\delta}\right)$$

What can be learned

General Boolean functions

- How many Boolean functions exist with n variables? 2^{2^n}

$$m > \frac{1}{\epsilon}\left(\ln(|H|) + \ln\frac{1}{\delta}\right)$$

What can be learned

General Boolean functions

- How many Boolean functions exist with n variables? 2^{2^n}

  So $\log(|H|)$ is exponential.
- General Boolean functions are not PAC learnable

Sample Complexity

- k-CNF: Conjunctions of any number of clauses where each disjunctive clause has at most k literals.
- k-clause-CNF: Conjunctions of at most k disjunctive clauses.

$$f = C_1 \wedge C_2 \wedge \cdots \wedge C_k$$
$$C_i = l_1 \vee l_2 \vee \cdots \vee l_m$$

$$\ln(|\text{k−clause−CNF}|) = O(kn)$$

Sample Complexity

- k-CNF: Conjunctions of any number of clauses where each disjunctive clause has at most k literals.
- k-clause-CNF: Conjunctions of at most k disjunctive clauses.

$$f = C_1 \wedge C_2 \wedge \cdots \wedge C_k$$
$$C_i = l_1 \vee l_2 \vee \cdots \vee l_m$$

$$\ln(|\text{k−clause−CNF}|) = O(kn)$$

- k-DNF: Disjunctions of any number of terms where each conjunctive term has at most k literals.

$$f = T_1 \vee T_2 \vee \cdots \vee T_m$$
$$C_i = l_1 \wedge l_2 \wedge \cdots \wedge l_k$$

- k-term-DNF: Disjunctions of at most k conjunctive terms.

Sample Complexity

- k-CNF: Conjunctions of any number of clauses where each disjunctive clause has at most k literals.
- k-clause-CNF: Conjunctions of at most k disjunctive clauses.

$$f = C_1 \wedge C_2 \wedge \cdots \wedge C_k$$
$$C_i = l_1 \vee l_2 \vee \cdots \vee l_m$$

$$\ln(|\text{k-clause-CNF}|) = O(kn)$$

- k-DNF: Disjunctions of any number of terms where each conjunctive term has at most k literals.

$$f = T_1 \vee T_2 \vee \cdots \vee T_m$$
$$C_i = l_1 \wedge l_2 \wedge \cdots \wedge l_k$$

- k-term-DNF: Disjunctions of at most k conjunctive terms.

All these classes can be learned using a polynomial size sample

Exercise: Prove that the above four classes of functions have polynomial sample complexity

Sample Complexity

- k-CNF: Conjunctions of any number of clauses where each disjunctive clause has at most k literals.
- k-clause-CNF: Conjunctions of at most k disjunctive clauses.

$$f = C_1 \wedge C_2 \wedge \cdots \wedge C_k$$
$$C_i = l_1 \vee l_2 \vee \cdots \vee l_m$$

$$\ln(|\text{k−clause−CNF}|) = O(kn)$$

- k-DNF: Disjunctions of any number of terms where each conjunctive term has at most k literals.

$$f = T_1 \vee T_2 \vee \cdots \vee T_m$$
$$C_i = l_1 \wedge l_2 \wedge \cdots \wedge l_k$$

- k-term-DNF: Disjunctions of at most k conjunctive terms.

All these classes can be learned using a polynomial size sample

Suppose we want to learn a 2-term-DNF
What should our hypothesis class be?

Computational Complexity

Suppose we want to learn a 2-term-DNF

- Though sample complexity is polynomial, the computational complexity is prohibitive in this case

Computational Complexity

Suppose we want to learn a 2-term-DNF

- Though sample complexity is polynomial, the computational complexity is prohibitive in this case
  - Determining whether there is a 2-term DNF consistent with a set of training data is NP-hard
  - That is, the class of k-term-DNF is not *efficiently* PAC learnable due to computational complexity

Computational Complexity

Suppose we want to learn a 2-term-DNF

- Though sample complexity is polynomial, the computational complexity is prohibitive in this case
  - Determining whether there is a 2-term DNF consistent with a set of training data is NP-hard
  - That is, the class of k-term-DNF is not *efficiently* PAC learnable due to computational complexity
- But, we have seen an algorithm for learning k-CNF (It was an exercise a few slides back.)

Computational Complexity

Suppose we want to learn a 2-term-DNF

- Though sample complexity is polynomial, the computational complexity is prohibitive in this case
  - Determining whether there is a 2-term DNF consistent with a set of training data is NP-hard
  - That is, the class of k-term-DNF is not *efficiently* PAC learnable due to computational complexity
- But, we have seen an algorithm for learning k-CNF
  - And, k-CNF is a superset of k-term-DNF

(That is, every k-term-DNF can be written as a k-CNF)

$$T_1 \vee T_2 \vee T_3 = \bigwedge_{x \in T_1, y \in T_2, z \in T_3} x \vee y \vee z$$

Computational Complexity

Suppose we want to learn a 2-term-DNF

- Though sample complexity is polynomial, the computational complexity is prohibitive in this case
  - Determining whether there is a 2-term DNF consistent with a set of training data is NP-hard
  - That is, the class of k-term-DNF is not *efficiently* PAC learnable due to computational complexity
- But, we have seen an algorithm for learning k-CNF
  - And, k-CNF is a superset of k-term-DNF

  (That is, every k-term-DNF can be written as a k-CNF)

$$T_1 \vee T_2 \vee T_3 = \bigwedge_{x \in T_1, y \in T_2, z \in T_3} x \vee y \vee z$$

Example:

$(a \wedge b \wedge c) \vee (d \wedge e \wedge f) = (a \vee d) \wedge (a \vee e) \wedge (a \vee f) \wedge (b \vee d) \wedge \cdots \wedge (c \vee f)$

Computational Complexity

Suppose we want to learn a 2-term-DNF

- Though sample complexity is polynomial, the computational complexity is prohibitive in this case
  - Determining whether there is a 2-term DNF consistent with a set of training data is NP-hard
  - That is, the class of k-term-DNF is not efficiently PAC learnable due to computational complexity
- But, we have seen an algorithm for learning k-CNF
  - And, k-CNF is a superset of k-term-DNF

  (That is, every k-term-DNF can be written as a k-CNF)

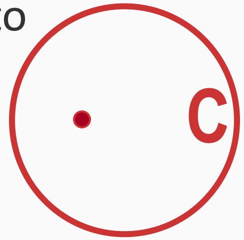
That is, the concept class C = k-term-DNF can be learned using H = k-CNF as the hypothesis space

Computational Complexity

Suppose we want to learn a 2-term-DNF

- Though sample complexity is polynomial, the computational complexity is prohibitive in this case
  - Determining whether there is a 2-term DNF consistent with a set of training data is NP-hard
  - That is, the class of k-term-DNF is not efficiently PAC learnable due to computational complexity
- But, we have seen an algorithm for learning k-CNF
  - And, k-CNF is a superset of k-term-DNF

  (That is, every k-term-DNF can be written as a k-CNF)



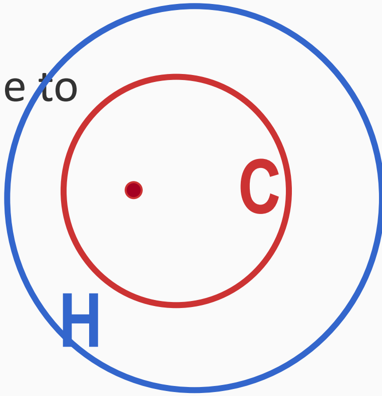
That is, the concept class C = k-term-DNF can be learned using H = k-CNF as the hypothesis space

Computational Complexity

Suppose we want to learn a 2-term-DNF

- Though sample complexity is polynomial, the computational complexity is prohibitive in this case
  - Determining whether there is a 2-term DNF consistent with a set of training data is NP-hard
  - That is, the class of k-term-DNF is not efficiently PAC learnable due to computational complexity
- But, we have seen an algorithm for learning k-CNF
  - And, k-CNF is a superset of k-term-DNF

(That is, every k-term-DNF can be written as a k-CNF)



That is, the concept class C = k-term-DNF can be learned using H = k-CNF as the hypothesis space

Computational Complexity

Suppose we want to learn a 2-term-DNF

- Though sample complexity is polynomial, the computational complexity is prohibitive in this case
  - Determining whether there is a 2-term DNF consistent with a set of training data is NP-hard
  - That is, the class of k-term-DNF is not efficiently PAC learnable due to computational complexity
- But, we have seen an algorithm for learning k-CNF
  - And, k-CNF is a superset of k-term-DNF

  (That is, every k-term-DNF can be written as a k-CNF)

The lesson: Importance of representation Concepts that cannot be learned using one representation can *sometimes* be learned using a different, more expressive, representation

Computational Complexity

Suppose we want to learn a 2-term-DNF

- Though sample complexity is polynomial, the computational complexity is prohibitive in this case
  - Determining whether there is a 2-term DNF consistent with a set of training data is NP-hard
  - That is, the class of k-term-DNF is not efficiently PAC learnable due to computational complexity
- But, we have seen an algorithm for learning k-CNF
  - And, k-CNF is a superset of k-term-DNF

  (That is, every k-term-DNF can be written as a k-CNF)

The lesson: Importance of representation Concepts that cannot be learned using one representation can *sometimes* be learned using a different, more expressive, representation

We have seen this idea before: Linear classifiers for conjunctions

Negative Results – Examples

Two types of non-learnability results

Negative Results – Examples

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”

Negative Results – Examples

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?)

Negative Results for Learning

- Complexity Theoretic
  - k-term DNF, for k>1 (k-clause CNF, k>1)
  - Neural Networks of fixed architecture: 2-layer, 3-nodes, n inputs, threshold activations. (Blum and Rivest, 1988)
  - “read-once” Boolean formulas (Pitt and Valiant, 1988)
  - Quantified conjunctive concepts

- Information Theoretic
  - Arbitrary Boolean functions (DNF Formulas or CNF Formulas)
  - Deterministic Finite Automata
  - Context Free Grammars