Computational Learning Theory – PAC Learning

Lecture Outline

- Introduction to Computational Learning Theory – What are the Questions?
- The Probably Approximately Correct (PAC) Learning Model
- Sample Complexity for Finite Hypothesis Spaces
  - Consistent Learners
  - Learning Conjunctions of Boolean Literals
  - Agnostic Learning and Inconsistent Hypotheses

Reading

Mitchell, Chapter 7.1-7.3

Introduction to Computational Learning Theory

- We have studied various algorithms for machine learning.
- Reasonable to ask: Are there laws that underlie/govern all machine/non-machine learners?

Introduction to Computational Learning Theory

- We have studied various algorithms for machine learning.
- Reasonable to ask: Are there laws that underlie/govern all machine/non-machine learners?
- In particular:
  1. Can we identify classes of learning problems that are inherently difficult/easy, regardless of ML algorithm?

Introduction to Computational Learning Theory

- We have studied various algorithms for machine learning.
- Reasonable to ask: Are there laws that underlie/govern all machine/non-machine learners?
- In particular:
  1. Can we identify classes of learning problems that are inherently difficult/easy, regardless of ML algorithm?
  2. Can we discover general laws that tell us how many examples must be supplied to ensure successful learning?

Introduction to Computational Learning Theory

- We have studied various algorithms for machine learning.
- Reasonable to ask: Are there laws that underlie/govern all machine/non-machine learners?
- In particular:
  1. Can we identify classes of learning problems that are inherently difficult/easy, regardless of ML algorithm?
  2. Can we discover general laws that tell us how many examples must be supplied to ensure successful learning?
  3. Does this number depend on whether the learner can ask questions, or whether training examples are presented to it at random?

Introduction to Computational Learning Theory

- We have studied various algorithms for machine learning.
- Reasonable to ask: Are there laws that underlie/govern all machine/non-machine learners?
- In particular:
  1. Can we identify classes of learning problems that are inherently difficult/easy, regardless of ML algorithm?
  2. Can we discover general laws that tell us how many examples must be supplied to ensure successful learning?
  3. Does this number depend on whether the learner can ask questions, or whether training examples are presented to it at random?
  4. Can we determine how many mistakes a learner will make in the course of learning a target function?

Introduction to Computational Learning Theory

- We have studied various algorithms for machine learning.
- Reasonable to ask: Are there laws that underlie/govern all machine/non-machine learners?
- In particular:
  1. Can we identify classes of learning problems that are inherently difficult/easy, regardless of ML algorithm?
  2. Can we discover general laws that tell us how many examples must be supplied to ensure successful learning?
  3. Does this number depend on whether the learner can ask questions, or whether training examples are presented to it at random?
  4. Can we determine how many mistakes a learner will make in the course of learning a target function?
- Fully general answers to all these questions are not yet known.

  But, progress is being made towards establishing a **computational theory of learning**

Introduction to Computational Learning Theory (cont)

- We shall focus on two of these questions, not for specific learning algorithms, but for classes of algorithms characterised by
  - the hypothesis spaces they consider
  - the manner of presentation of examples

Introduction to Computational Learning Theory (cont)

- We shall focus on two of these questions, not for specific learning algorithms, but for classes of algorithms characterised by
  - the hypothesis spaces they consider
  - the manner of presentation of examples

  These questions are:
  1. *Sample Complexity*: How many examples are needed to successfully learn the target function?

Introduction to Computational Learning Theory (cont)

- We shall focus on two of these questions, not for specific learning algorithms, but for classes of algorithms characterised by
  - the hypothesis spaces they consider
  - the manner of presentation of examples

  These questions are:

  1. *Sample Complexity*: How many examples are needed to successfully learn the target function?
  2. *Mistake bound*: How many mistakes will the learner make before succeeding?

Introduction to Computational Learning Theory (cont)

- We shall focus on two of these questions, not for specific learning algorithms, but for classes of algorithms characterised by
  - the hypothesis spaces they consider
  - the manner of presentation of examples

  These questions are:
  1. *Sample Complexity*: How many examples are needed to successfully learn the target function?
  2. *Mistake bound*: How many mistakes will the learner make before succeeding?

- Quantitative bounds can be put on these measures given quantitative measures of:
  - size/complexity of the hypothesis space examined by the learner
  - accuracy to which we wish to approximate the target concept
  - probability that the learner will output a successful hypothesis
  - manner in which training examples are presented to the learner

The PAC Learning Model: General Setting

- Initially we consider only the case of learning boolean-valued functions from noise-free data.

The PAC Learning Model: General Setting

- Initially we consider only the case of learning boolean-valued functions from noise-free data.
- The general learning setting we consider is as follows:
  - Let X be the set of all possible instances over which target functions are to be defined

The PAC Learning Model: General Setting

- Initially we consider only the case of learning boolean-valued functions from noise-free data.
- The general learning setting we consider is as follows:
  - Let X be the set of all possible instances over which target functions are to be defined
  - C is the set of target concepts the learner may be asked to learn
    - for each $c \in C$, $c \subseteq X$
    - alternatively, c may be viewed as a boolean-valued function $c : X \rightarrow \{0, 1\}$ if x is a positive example $c(x) = 1$; if x is a negative example $c(x) = 0$.

The PAC Learning Model: General Setting

- Initially we consider only the case of learning boolean-valued functions from noise-free data.
- The general learning setting we consider is as follows:
  - Let X be the set of all possible instances over which target functions are to be defined
  - C is the set of target concepts the learner may be asked to learn
    - for each $c \in C$, $c \subseteq X$
    - alternatively, c may be viewed as a boolean-valued function $c : X \rightarrow \{0, 1\}$ if x is a positive example $c(x) = 1$; if x is a negative example $c(x) = 0$.
  - Examples are drawn at random from X according to a probability distribution $\mathcal{D}$.

The PAC Learning Model: General Setting

- Initially we consider only the case of learning boolean-valued functions from noise-free data.
- The general learning setting we consider is as follows:
  - Let X be the set of all possible instances over which target functions are to be defined
  - C is the set of target concepts the learner may be asked to learn
    - for each $c \in C$, $c \subseteq X$
    - alternatively, c may be viewed as a boolean-valued function $c : X \rightarrow \{0, 1\}$ if x is a positive example $c(x) = 1$; if x is a negative example $c(x) = 0$.
  - Examples are drawn at random from X according to a probability distribution $\mathcal{D}$.
  - A learner L considers a set of hypotheses H and, after observing some sequence of training examples, outputs a hypothesis $h \in H$ which is its estimate of c.

The PAC Learning Model: General Setting

- Initially we consider only the case of learning boolean-valued functions from noise-free data.
- The general learning setting we consider is as follows:
  - Let X be the set of all possible instances over which target functions are to be defined
  - C is the set of target concepts the learner may be asked to learn
    - for each $c \in C$, $c \subseteq X$
    - alternatively, c may be viewed as a boolean-valued function $c : X \rightarrow \{0, 1\}$ if x is a positive example $c(x) = 1$; if x is a negative example $c(x) = 0$.
  - Examples are drawn at random from X according to a probability distribution $\mathcal{D}$.
  - A learner L considers a set of hypotheses H and, after observing some sequence of training examples, outputs a hypothesis $h \in H$ which is its estimate of c.
  - L is evaluated by testing the performance of h over new instances drawn randomly from X according to $\mathcal{D}$.

The PAC Learning Model: General Setting

- Initially we consider only the case of learning boolean-valued functions from noise-free data.
- The general learning setting we consider is as follows:
  - Let X be the set of all possible instances over which target functions are to be defined
  - C is the set of target concepts the learner may be asked to learn
    - for each $c \in C$, $c \subseteq X$
    - alternatively, c may be viewed as a boolean-valued function $c : X \rightarrow \{0, 1\}$ if x is a positive example $c(x) = 1$; if x is a negative example $c(x) = 0$.
  - Examples are drawn at random from X according to a probability distribution $\mathcal{D}$.
  - A learner L considers a set of hypotheses H and, after observing some sequence of training examples, outputs a hypothesis $h \in H$ which is its estimate of c.
  - L is evaluated by testing the performance of h over new instances drawn randomly from X according to $\mathcal{D}$.
- Given this setting computational learning theory attempts to characterise
  - various learners L
  - using various hypothesis spaces H
  - learning target concepts drawn from various classes C
  - drawn according to various instance distributions $\mathcal{D}$

The PAC Learning Model: Example Setting

- **Given:**
  - Instances X: e.g. possible days, each described by the attributes *Sky, AirTemp, Humidity, Wind, Water, Forecast*
  - Target function c: e.g. $EnjoySport : X \rightarrow \{0,1\}$
  - Hypotheses H: Conjunctions of literals, e.g.

$$\langle ?, Cold, High, ?, ?, ? \rangle.$$

  - Training examples D: Positive and negative examples of the target function

$$\langle x_1, c(x_1) \rangle, \ldots \langle x_m, c(x_m) \rangle$$

The PAC Learning Model: Example Setting

- **Given:**
  - Instances X: e.g. possible days, each described by the attributes *Sky, AirTemp, Humidity, Wind, Water, Forecast*
  - Target function c: e.g. $EnjoySport : X \rightarrow \{0,1\}$
  - Hypotheses H: Conjunctions of literals, e.g.

$$\langle ?, Cold, High, ?, ?, ? \rangle .$$

  - Training examples D: Positive and negative examples of the target function

$$\langle x_1, c(x_1) \rangle, \ldots \langle x_m, c(x_m) \rangle$$

- **Determine:**
  - A hypothesis h in H such that $h(x) = c(x)$ for all x in D?
  - A hypothesis h in H such that $h(x) = c(x)$ for all x in X?

The PAC Learning Model: True Error of a Hypothesis

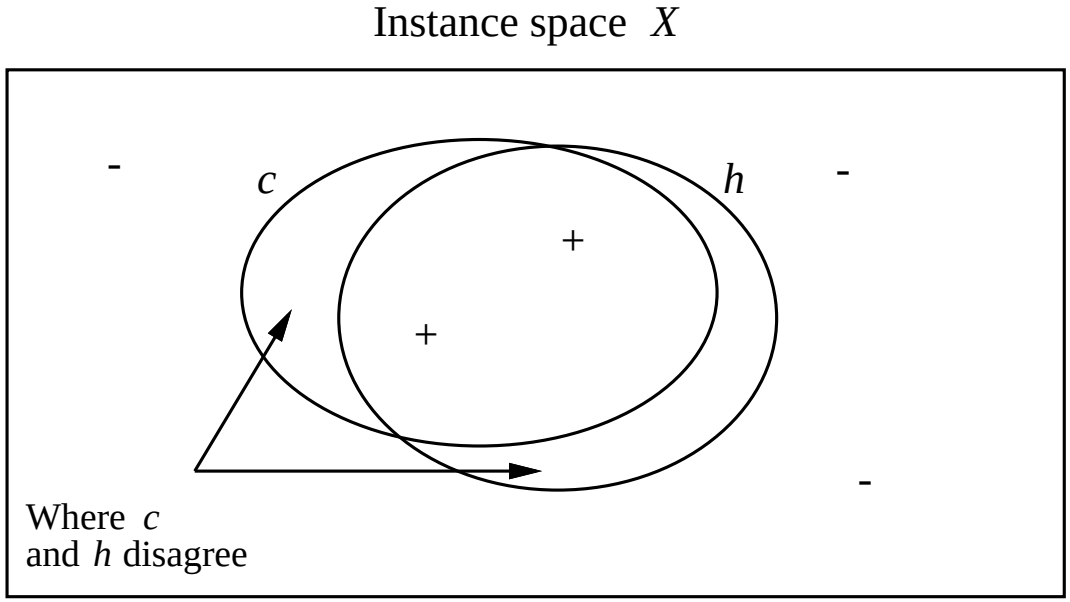
- Need to recall notion of **true error** of a hypothesis.
- **Definition:** The **true error** (denoted $error_{\mathcal{D}}(h)$) of hypothesis h with respect to target concept c and distribution $\mathcal{D}$ is the probability that h will misclassify an instance drawn at random according to $\mathcal{D}$.

$$error_{\mathcal{D}}(h) \equiv \Pr_{x \in \mathcal{D}}[c(x) \neq h(x)]$$

The PAC Learning Model: True Error of a Hypothesis

- Need to recall notion of **true error** of a hypothesis.
- **Definition:** The **true error** (denoted $error_{\mathcal{D}}(h)$) of hypothesis h with respect to target concept c and distribution $\mathcal{D}$ is the probability that h will misclassify an instance drawn at random according to $\mathcal{D}$.

$$error_{\mathcal{D}}(h) \equiv \Pr_{x \in \mathcal{D}}[c(x) \neq h(x)]$$



Source: Mitchell, Chapter 7

- $+$ and $-$ denote positive/negative training examples.
- Note c and h have non-zero error despite agreeing on all training examples so far.

The PAC Learning Model: Training Error of a Hypothesis

- Cannot observe **true error**, but can observe **training error** (training error = errors h makes over training examples).

- Interested in question: how probable is it that observed training error for h gives a misleading estimate of $error_{\mathcal{D}}(h)$?

The PAC Learning Model: Training Error of a Hypothesis

- Cannot observe **true error**, but can observe **training error** (training error = errors h makes over training examples).

- Interested in question: how probable is it that observed training error for h gives a misleading estimate of $error_{\mathcal{D}}(h)$?

- Training error analogous to **sample error** (considered in Lecture 8).
  - sample error of h with respect to a set of examples S is fraction of S misclassified by h
  - training error = sample error when S is the training examples

The PAC Learning Model: Training Error of a Hypothesis

- Cannot observe **true error**, but can observe **training error** (training error = errors h makes over training examples).

- Interested in question: how probable is it that observed training error for h gives a misleading estimate of $error_{\mathcal{D}}(h)$?

- Training error analogous to **sample error** (considered in Lecture 8).
  - sample error of h with respect to a set of examples S is fraction of S misclassified by h
  - training error = sample error when S is the training examples

- However, significant difference when determining probability that sample/training error is a good estimate of true error
  - when (Lecture 8/9) calculating probability that sample error reflects true error assumed sample S was drawn independently from training data used to form h
  - clearly this assumption is false when considering training error ...

The PAC Learning Model: PAC-Learnability

- Would like to determine how many training examples we need to learn a hypothesis h with $error_{\mathcal{D}}(h) = 0$

The PAC Learning Model: PAC-Learnability

- Would like to determine how many training examples we need to learn a hypothesis h with $error_{\mathcal{D}}(h) = 0$
- Pointless trying to do this, for two reasons:
  1. since for any set of training examples $T \subset X$ there may be multiple hypotheses consistent with T, learner cannot be guaranteed to choose one corresponding to the target concept, unless it is trained on every instance in X (unrealistic)

The PAC Learning Model: PAC-Learnability

- Would like to determine how many training examples we need to learn a hypothesis h with $error_{\mathcal{D}}(h) = 0$
- Pointless trying to do this, for two reasons:
  1. since for any set of training examples $T \subset X$ there may be multiple hypotheses consistent with T, learner cannot be guaranteed to choose one corresponding to the target concept, unless it is trained on every instance in X (unrealistic)
  2. since training examples drawn at random, must be a non-zero probability that they will be misleading

The PAC Learning Model: PAC-Learnability

- Would like to determine how many training examples we need to learn a hypothesis h with $error_{\mathcal{D}}(h) = 0$
- Pointless trying to do this, for two reasons:
  1. since for any set of training examples $T \subset X$ there may be multiple hypotheses consistent with T, learner cannot be guaranteed to choose one corresponding to the target concept, unless it is trained on every instance in X (unrealistic)
  2. since training examples drawn at random, must be a non-zero probability that they will be misleading
- Therefore, weaken demands on learner:
  1. Don't require learner to output a zero error hypothesis – only require that error be bounded by a constant ε that can be made arbitrarily small

The PAC Learning Model: PAC-Learnability

- Would like to determine how many training examples we need to learn a hypothesis h with $error_{\mathcal{D}}(h) = 0$
- Pointless trying to do this, for two reasons:
  1. since for any set of training examples $T \subset X$ there may be multiple hypotheses consistent with T, learner cannot be guaranteed to choose one corresponding to the target concept, unless it is trained on every instance in X (unrealistic)
  2. since training examples drawn at random, must be a non-zero probability that they will be misleading
- Therefore, weaken demands on learner:
  1. Don't require learner to output a zero error hypothesis – only require that error be bounded by a constant ε that can be made arbitrarily small
  2. Don't require learner to succeed for every randomly drawn sequence of training examples – only require that its probability of failure be bounded by a constant δ that can also be made arbitrarily small

The PAC Learning Model: PAC-Learnability

- Would like to determine how many training examples we need to learn a hypothesis h with $error_{\mathcal{D}}(h) = 0$
- Pointless trying to do this, for two reasons:
  1. since for any set of training examples $T \subset X$ there may be multiple hypotheses consistent with T, learner cannot be guaranteed to choose one corresponding to the target concept, unless it is trained on every instance in X (unrealistic)
  2. since training examples drawn at random, must be a non-zero probability that they will be misleading
- Therefore, weaken demands on learner:
  1. Don't require learner to output a zero error hypothesis – only require that error be bounded by a constant ε that can be made arbitrarily small
  2. Don't require learner to succeed for every randomly drawn sequence of training examples – only require that its probability of failure be bounded by a constant δ that can also be made arbitrarily small
- Put otherwise, only assume learner will **probably** learn a hypothesis that is **approximately** correct – hence **probably approximately correct** (PAC) learning

The PAC Learning Model: PAC-Learnability

- Consider a class C of possible target concepts defined over a set of instances X of length n, and a learner L using hypothesis space H.

  Definition: C is **PAC-learnable** by L using H if for all $c \in C$, distributions $\mathcal{D}$ over X, ε such that $0 < \varepsilon < 1/2$, and δ such that $0 < \delta < 1/2$, learner L will with probability at least $(1-\delta)$ output a hypothesis $h \in H$ such that $error_{\mathcal{D}}(h) \leq \varepsilon$, in time that is polynomial in $1/\varepsilon$, $1/\delta$, n and $size(c)$.

  Here

  – n measures length of individual instances – e.g. if instances are conjunctions of k boolean features $n = k$

  – $size(c)$ is a length encoding of c – so if concepts in C are conjunctions of 1 to k boolean features then $size(c)$ is number of boolean features used to describe c

The PAC Learning Model: PAC-Learnability

- Consider a class C of possible target concepts defined over a set of instances X of length n, and a learner L using hypothesis space H.

  Definition: C is **PAC-learnable** by L using H if for all $c \in C$, distributions $\mathcal{D}$ over X, ε such that $0 < \varepsilon < 1/2$, and δ such that $0 < \delta < 1/2$, learner L will with probability at least $(1-\delta)$ output a hypothesis $h \in H$ such that $error_{\mathcal{D}}(h) \leq \varepsilon$, in time that is polynomial in $1/\varepsilon$, $1/\delta$, n and $size(c)$.

  Here

  - n measures length of individual instances – e.g. if instances are conjunctions of k boolean features $n = k$
  - $size(c)$ is a length encoding of c – so if concepts in C are conjunctions of 1 to k boolean features then $size(c)$ is number of boolean features used to describe c

- Note definition of PAC-learnability requires

  1. L must output with arbitrarily high probability $(1-\delta)$ a hypothesis with arbitrarily low error ε.

The PAC Learning Model: PAC-Learnability

- Consider a class C of possible target concepts defined over a set of instances X of length n, and a learner L using hypothesis space H.

  Definition: C is **PAC-learnable** by L using H if for all $c \in C$, distributions $\mathcal{D}$ over X, ε such that $0 < \varepsilon < 1/2$, and δ such that $0 < \delta < 1/2$, learner L will with probability at least $(1-\delta)$ output a hypothesis $h \in H$ such that $error_{\mathcal{D}}(h) \leq \varepsilon$, in time that is polynomial in $1/\varepsilon$, $1/\delta$, n and $size(c)$.

  Here

  - n measures length of individual instances – e.g. if instances are conjunctions of k boolean features $n = k$
  - $size(c)$ is a length encoding of c – so if concepts in C are conjunctions of 1 to k boolean features then $size(c)$ is number of boolean features used to describe c

- Note definition of PAC-learnability requires

  1. L must output with arbitrarily high probability $(1-\delta)$ a hypothesis with arbitrarily low error ε.
  2. L must output such a hypothesis efficiently – in time that grows at most polynomially in
     - $1/\varepsilon$ and $1/\delta$ (demands on hypothesis)
     - n (demand on complexity of instance space X)
     - $size(c)$ (demand on complexity of concept class C)

The PAC Learning Model: PAC-Learnability

- Note definition of PAC-learnability does not explicitly mention number of training examples required – only mentions computational resources (time) required.
- However, overall time required and number of examples are closely related.
  - If L requires minimum time per training example, then for C to be PAC-learnable by L, L must learn from a number of examples which is polynomial in $1/\varepsilon$, $1/\delta$, n and $size(c)$.
  - Typical approach to proving some class C is PAC-learnable is to show
    - each concept in C can be learned from a polynomial number of training examples
    - processing time per training example is polynomially-bounded

The PAC Learning Model: PAC-Learnability (cont)

- Note definition of PAC-learnability assumes H contains a hypothesis with arbitrarily small error for each concept in C
  - Difficult to know this, if C is not known in advance (e.g. what is C for problem of recognising NL sentences?)
  - Could know this if H is unbiased – i.e. powerset of X – but have shown this an unrealistic setting for ML
  - Still, PAC learning provides valuable insights into complexity of learning problems + relationship between generalisation accuracy and number of training examples
  - Can also modify framework to remove this assumption

Sample Complexity and Consistent Learners

- Growth in number of required training examples with problem size is called **sample complexity**

Sample Complexity and Consistent Learners

- Growth in number of required training examples with problem size is called **sample complexity**
- What is bound on sample complexity for *consistent learners*?
  - Remember: a consistent learner is one that outputs hypotheses that perfectly fit the training data, if such is possible

  Interested in such a bound *independently of any specific algorithm*

Sample Complexity and Consistent Learners

- Growth in number of required training examples with problem size is called **sample complexity**
- What is bound on sample complexity for *consistent learners*?
  - Remember: a consistent learner is one that outputs hypotheses that perfectly fit the training data, if such is possible

  Interested in such a bound *independently of any specific algorithm*

- Recall definition of version space (lecture 3). Given hypothesis space H, target concept c, and training data D

$$VS_{H,D} = \{h \in H | (\forall \langle x, c(x) \rangle \in D)(h(x) = c(x))\}$$

  - Version space contains every consistent hypothesis
  - Therefore, every conistent learner outputs a hypothesis in the version space
  - So, to bound number of examples needed by a consistent learner need only bound number of examples needed to ensure the version space contains no unacceptable hypotheses

Sample Complexity and Consistent Learners (cont)

- More precisely

  Definition: The version space $VS_{H,D}$ is said to be **ε-exhausted** with respect to target concept c and instance distribution $\mathcal{D}$, if every hypothesis h in $VS_{H,D}$ has error less than ε with respect to c and $\mathcal{D}$.

$$(\forall h \in VS_{H,D})\ error_{\mathcal{D}}(h) < \varepsilon$$

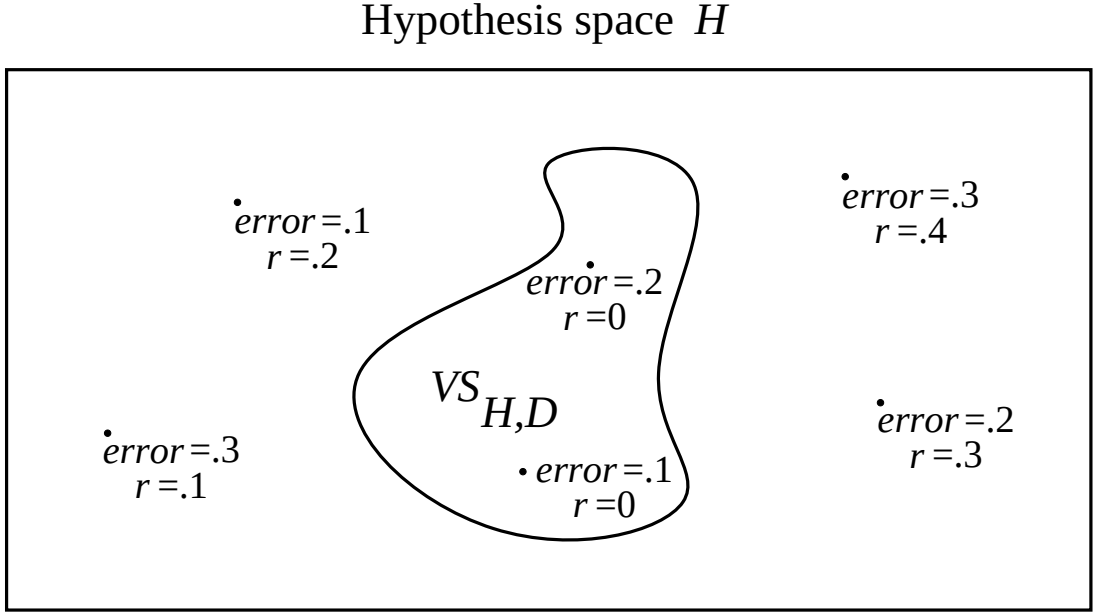
Sample Complexity and Consistent Learners (cont)

- More precisely

  Definition: The version space $VS_{H,D}$ is said to be **ε-exhausted** with respect to target concept c and instance distribution $\mathcal{D}$, if every hypothesis h in $VS_{H,D}$ has error less than ε with respect to c and $\mathcal{D}$.

$$(\forall h \in VS_{H,D})\ error_{\mathcal{D}}(h) < \varepsilon$$

- Example:



Source: Mitchell, Chapter 7

  - $VS_{H,D}$ is subset of $h \in H$ such that h has no training error ($r = 0$)
  - Note that $error_{\mathcal{D}}(h)$ may be non-zero, even when h makes no errors on training data
  - Version space is ε-exhausted when all hypotheses remaining in $VS_{H,D}$ have $error_{\mathcal{D}}(h) < \varepsilon$

Sample Complexity and Consistent Learners (cont)

- Can demonstrate a bound on probability that version space will be ε-exhausted after a given number of training examples, even without knowing
  - identity of target concept
  - distribution from which training examples are drawn

  Theorem: **ε-exhausting the version space**

  If the hypothesis space H is finite, and D is a sequence of $m \geq 1$ independent random examples of some target concept c, then for any $0 \leq \varepsilon \leq 1$, the probability that the version space with respect to H and D is not ε-exhausted (with respect to c) is less than

$$|H|e^{-\varepsilon m}$$

  (see Mitchell, p. 209 for proof; based D. Haussler, 1988)

Sample Complexity and Consistent Learners (cont)

- This theorem bounds the probability that any consistent learner will output a hypothesis h with $error_{\mathcal{D}}(h) \geq \varepsilon$
- If we want to this probability to be below δ

$$|H|e^{-\varepsilon m} \leq \delta$$

then

$$m \geq \frac{1}{\varepsilon}(\ln|H| + \ln(1/\delta))$$

  - This gives a general bound on the number (m) of training examples needed to learn any target concept in H for any δ, ε

Sample Complexity:
Learning Conjunctions of Boolean Literals

- Given preceding theorem, can use it to determine sample complexity and PAC-learnability of specific concept classes
- Consider concept class C consisting of concepts describable by conjunctions of boolean literals (boolean variables and negations of boolean variables).
- Suppose H contains conjunctions of constraints on up to n boolean attributes (i.e., n boolean literals). Then $|H| = 3^n$ (for each literal a hypothesis may either include it, include its negation, not include it).
- How many examples are sufficient to assure with probability at least $(1-\delta)$ that every h in $VS_{H,D}$ satisfies $error_{\mathcal{D}}(h) \leq \varepsilon$?
- Using the theorem:

$$\begin{aligned} m &\geq \frac{1}{\varepsilon}(\ln|H| + \ln(1/\delta)) \\ &\geq \frac{1}{\varepsilon}(\ln 3^n + \ln(1/\delta)) \\ &\geq \frac{1}{\varepsilon}(n\ln 3 + \ln(1/\delta)) \end{aligned}$$

Sample Complexity:
Learning Conjunctions of Boolean Literals – Example

- Suppose
  - concept to be learned can be described by conjunctions of up to 10 boolean literals
  - we desire with 95% probability that a hypothesis be learned with error less than .1.

Sample Complexity:
Learning Conjunctions of Boolean Literals – Example

- Suppose
  - concept to be learned can be described by conjunctions of up to 10 boolean literals
  - we desire with 95% probability that a hypothesis be learned with error less than .1.

- Using the equation

$$\begin{aligned} m &\geq \frac{1}{\varepsilon}(n\ln 3 + \ln(1/\delta)) \\ &\geq \frac{1}{.1}(10\ln 3 + \ln(1/.05)) \\ &\geq 10((10 * 1.0986) + 2.9957) \\ &\geq 140 \end{aligned}$$

- So, 140 training examples will suffice for a consistent learner to learn with 95% probability a hypothesis of length up to 10 boolean literals which has true error less than 10 %.

Sample Complexity: Learning Conjunctions of Boolean Literals – PAC Learnability

- Overall computational effort is dependent on specific learning algorithm.

Sample Complexity: Learning Conjunctions of Boolean Literals – PAC Learnability

- Overall computational effort is dependent on specific learning algorithm.
- Recall FIND-S algorithm of Lecture 3.
  - FIND-S incrementally finds most specific hypothesis consistent with training examples
  - for each new positive training example FIND-S computes the intersection of literals in the current hypothesis and the training example
  - each such example is processed in time linear in number of literals, n

Sample Complexity: Learning Conjunctions of Boolean Literals – PAC Learnability

- Overall computational effort is dependent on specific learning algorithm.
- Recall FIND-S algorithm of Lecture 3.
  - FIND-S incrementally finds most specific hypothesis consistent with training examples
  - for each new positive training example FIND-S computes the intersection of literals in the current hypothesis and the training example
  - each such example is processed in time linear in number of literals, n
- *Theorem:* **PAC-learnability of boolean conjunctions** The class C of conjunctions of boolean literals is PAC-learnable by the FIND-S algorithm using $H = C$.

Sample Complexity: Learning Conjunctions of Boolean Literals – PAC Learnability

- Overall computational effort is dependent on specific learning algorithm.

- Recall FIND-S algorithm of Lecture 3.
  - FIND-S incrementally finds most specific hypothesis consistent with training examples
  - for each new positive training example FIND-S computes the intersection of literals in the current hypothesis and the training example
  - each such example is processed in time linear in number of literals, n

- *Theorem:* **PAC-learnability of boolean conjunctions** The class C of conjunctions of boolean literals is PAC-learnable by the FIND-S algorithm using $H = C$.

  Proof: The earlier theorem concerning ε-exhausting the version space shows, when applied to learning boolean conjunctions, that the sample complexity of this concept class is polynomial in n, $1/\delta$, and $1/\varepsilon$ and independent of $size(c)$.

  To process each training example FIND-S requires time linear in n and independent of $1/\delta$, $1/\varepsilon$ and $size(c)$.

  Therefore C is PAC-learnable by FIND-S.

Sample Complexity: Agnostic Learning and Inconsistent Hypotheses

- So far, we have assumed $c \in H$

  I.e., there is at least one h with zero error over the training data

- Suppose this is not the case. What do we want then?
  - The hypothesis h that makes fewest errors on training data

  Such a learner, one that does not assume $c \in H$ is called an **agnostic learner**

- Let $error_D(h)$ be training error of h over training examples D
  - Note $error_D(h)$ may well differ from true error $error_{\mathcal{D}}(h)$

- Want to determine how many training examples are necessary to ensure with high probability that true error of best hypothesis $error_{\mathcal{D}}(h_{best})$ will be no more than $\varepsilon + error_D(h_{best})$

Sample Complexity: Agnostic Learning and Inconsistent Hypotheses (cont)

- Use "Hoeffding bounds" which assert that for training data D containing m randomly drawn examples:

$$Pr[error_{\mathcal{D}}(h) > error_{D}(h) + \varepsilon] \leq e^{-2m\varepsilon^2}$$

Sample Complexity: Agnostic Learning and Inconsistent Hypotheses (cont)

- Use "Hoeffding bounds" which assert that for training data D containing m randomly drawn examples:

$$Pr[error_{\mathcal{D}}(h) > error_D(h) + \varepsilon] \leq e^{-2m\varepsilon^2}$$

- This bounds probability that an arbitrarily chosen single hypothesis has a very misleading training error.

  To ensure best hypothesis has an error bounded this way, must consider that probability that any of $|H|$ hypotheses could have large error:

$$Pr[(\exists h \in H) error_{\mathcal{D}}(h) > error_D(h) + \varepsilon] \leq |H| e^{-2m\varepsilon^2}$$

Sample Complexity: Agnostic Learning and Inconsistent Hypotheses (cont)

- Use "Hoeffding bounds" which assert that for training data D containing m randomly drawn examples:

$$Pr[error_{\mathcal{D}}(h) > error_D(h) + \varepsilon] \leq e^{-2m\varepsilon^2}$$

- This bounds probability that an arbitrarily chosen single hypothesis has a very misleading training error.

  To ensure best hypothesis has an error bounded this way, must consider that probability that any of $|H|$ hypotheses could have large error:

$$Pr[(\exists h \in H) error_{\mathcal{D}}(h) > error_D(h) + \varepsilon] \leq |H| e^{-2m\varepsilon^2}$$

- Call this probability δ and determine how many training examples are necessary to hold δ to a desired value (i.e. to keep $\delta \leq |H| e^{-2m\varepsilon^2}$):

$$m \geq \frac{1}{2\varepsilon^2}(\ln |H| + \ln(1/\delta))$$

Sample Complexity: Agnostic Learning and Inconsistent Hypotheses (cont)

- Use "Hoeffding bounds" which assert that for training data D containing m randomly drawn examples:

$$Pr[error_{\mathcal{D}}(h) > error_D(h) + \varepsilon] \leq e^{-2m\varepsilon^2}$$

- This bounds probability that an arbitrarily chosen single hypothesis has a very misleading training error.

  To ensure best hypothesis has an error bounded this way, must consider that probability that any of $|H|$ hypotheses could have large error:

$$Pr[(\exists h \in H) error_{\mathcal{D}}(h) > error_D(h) + \varepsilon] \leq |H| e^{-2m\varepsilon^2}$$

- Call this probability δ and determine how many training examples are necessary to hold δ to a desired value (i.e. to keep $\delta \leq |H| e^{-2m\varepsilon^2}$):

$$m \geq \frac{1}{2\varepsilon^2}(\ln |H| + \ln(1/\delta))$$

- This is analogue of previous equation where learner was assumed to pick a hypothesis with zero training errors.

  Now assume learner picks "best hypotheses", but may still contain training errors.

Sample Complexity: Agnostic Learning and Inconsistent Hypotheses (cont)

- Use "Hoeffding bounds" which assert that for training data D containing m randomly drawn examples:

$$Pr[error_{\mathcal{D}}(h) > error_D(h) + \varepsilon] \leq e^{-2m\varepsilon^2}$$

- This bounds probability that an arbitrarily chosen single hypothesis has a very misleading training error.

  To ensure best hypothesis has an error bounded this way, must consider that probability that any of $|H|$ hypotheses could have large error:

$$Pr[(\exists h \in H) error_{\mathcal{D}}(h) > error_D(h) + \varepsilon] \leq |H| e^{-2m\varepsilon^2}$$

- Call this probability δ and determine how many training examples are necessary to hold δ to a desired value (i.e. to keep $\delta \leq |H| e^{-2m\varepsilon^2}$):

$$m \geq \frac{1}{2\varepsilon^2}(\ln |H| + \ln(1/\delta))$$

- This is analogue of previous equation where learner was assumed to pick a hypothesis with zero training errors.

  Now assume learner picks "best hypotheses", but may still contain training errors.

- Note m now grows as square of $1/\varepsilon$ rather than linearly with $1/\varepsilon$

Summary

- Computational learning theory attempts to discover laws/generalization that govern all learning systems. In particular it addresses questions such as:
  1. *Sample Complexity*: How many examples needed to successfully learn target function?
  2. *Mistake bound*: How many mistakes will the learner make before succeeding?
- One theoretical approach is the **PAC – probably approximately correct – learning model**
  - determine quantitative bounds on the probability that a learner will output a hypothesis which is approximately correct, i.e. whose true error is no greater than some bound
- For consistent learners operating over finite hypothesis spaces it is possible to demonstrate a bound on the probability that the version space will contain hypotheses with error greater than some other bound given a certain number of training examples.
  - This results lets us determine the sample complexity and PAC learnability of specific concepts classes. E.g.
    - the class of concepts C describable by conjunctions of boolean literals
- Can extend results to agnostic learners, i.e. for cases where there may be no hypotheses consistent with the training data in the hypothesis space, by using Hoeffding bounds