SL07. Computational Learning Theory

Introduction:

- The computational learning theory helps us:
  - Define learning problems.
  - Show that specific algorithms work.
  - Show that some problems are fundamentally hard (No algorithm in a particular class will ever be able to solve them).
- The tools used for analyzing learning problems are similar to those used for analyzing algorithms in computing.
- Resources in machine learning:
  - Time.
  - Space.
  - Data.
- Some of the resources useful in analyzing learning algorithms are time and space just like normal algorithms. Furthermore, learning algorithms are measured in terms of the number of samples or data they need to be effective.

Inductive Learning:

- As discussed before, Inductive Learning is basically learning from examples.
- Properties to think of when dealing with Inductive Learning:
  - Probability of successful trainings.
  - Number of training examples.
  - Complexity of hypothesis class (Complexity of *H*): A class of simple hypotheses can will not be sufficient to truly express complex concepts. Whereas a class of complex hypotheses can potentially overfit the data.
  - Accuracy to which the target concept is approximated.
  - Manner in which training examples are presented (Batch/Online).
  - Manner in which training examples are selected.

Selecting Training Examples:

- Several ways to select training examples:
  - The learner asks questions to the teacher.
  - The teacher gives examples to help the learner.
  - Fixed distribution.
  - Evil distribution.

- For a set of hypotheses *H*:

- The teacher can help the learner get to the right answer using only one question.
- The learner would have to ask $\log H$ questions to get to the answer on its own.

Learning with Constrained Queries

- Teacher with constrained queries:
  - The teacher has to show what's irrelevant. This can be achieved using two positive examples.
  - The teacher has to show what's relevant. This can be achieved using *k* negative examples, where *k* is the number of variables.
  - The answer can be achieved in *2+k* questions.
- Learner with constrained queries:
  Can't explain
- Learner with mistake bounds:
  These will be the new rules:
  - Input arrives.
  - Learner guesses answer.
  - Wrong answer charged.
  - Go to 1.

  Using these rules, we'll never make more than *k+1* mistakes.

Definitions

- Computational complexity: How much computational effort the learner needs to converge to a correct hypothesis, or to the best hypothesis in the available hypotheses space.
- Sample complexity: How much training examples the learner needs to create a successful hypothesis.
- Mistake bounds: How many misclassifications a learner can make over an infinite run.
- Version space: A consistent learner is a learner that produces a hypothesis that matches the data it had seen. Version space is the space of all the consistent hypotheses.

PAC (Probably Approximately Correct) Learning

- Training error: Fraction of training examples misclassified by hypothesis *h*.
- True error: Fraction of examples that would be misclassified on a sample drawn from distribution *D*.

$$error_d(h) = Pr_{x\sim D}[c(x) \neq h(x)]$$

- Parameters:
  - C: concept class.
  - L: Learner.
  - H: Hypothesis space.
  - n: |H| → size of hypothesis space.
  - D: distribution over inputs
  - Error goal:

$$0 \leq \varepsilon \leq {}^{1}/_{2}$$

- Certainty goal:

$$0 \leq \delta \leq {}^{1}/_{2}$$

- Since we're drawing our training sample randomly from a distribution *D*, it's possible to get unlucky and have a bad training set that results in higher error but it's okay because the probability of that happening is very low.
- Formal definition of PAC Learning:
  c is PAC-learnable by *L* using *H* if, and only if, learner *L* will, with probability $1 - \delta$, output a hypothesis $h \in H$ such that $error_d(h) \leq \varepsilon$ in time and samples polynomial in ${}^{1}/_{\varepsilon}$, ${}^{1}/_{\delta}$, & n.

Epsilon Exhaustion

- A version space is epsilon exhausted if, and only if, every hypothesis in the version space has an error less than epsilon.

$$VS(s) is\ \varepsilon - exhausted\ iff\ \forall h \in VS(s)\ error_d(h) \leq \varepsilon$$

Haussler Theorem – Bound True Error

- A way of bounding True Error as a function of the number of training examples.
  Can't explain