# Lecture 14

- Learning
  - Inductive inference
  - Probably approximately correct learning



# What is learning?

Key point: all learning can be seen as learning the *representation* of a function.

Will become clearer with more examples!

Example representations:

- propositional if-then rules
- first-order if-then rules
- first-order logic theories
- decision trees
- neural networks
- Java programs

# Learning: formalism

Come up with some function $f$ such that

- $f(x) = y$
  for all training examples (x,y) and
- $f$ (somehow) generalizes to yet unseen examples.
  - In practice, we don't always do it perfectly.



# Inductive bias: intro

- There has to be some structure apparent in the inputs in order to support generalization.
- Consider the following pairs from the phone book.

| Inputs | Outputs |
|---|---|
| Ralph Student | 941-2983 |
| Louie Reasoner | 456-1935 |
| Harry Coder | 247-1993 |
| **Fred Flintstone** | **???-????** |

There is not much to go on here.
•Suppose we were to add zip code information.
•Suppose phone numbers were issued based on the spelling of a person's last name.
•Suppose the outputs were user passwords?

# Example 2

- Consider the problem of fitting a curve to a set of (x,y) pairs.

```
|    x      x
|-x---------x---
|    x
|___________x_____
```

  - Should you fit a linear, quadratic, cubic, piece-wise linear function?
  - It would help to have some idea of how smooth the target function is or to know from what family of functions (e.g., polynomials of degree 3) to choose from.
  - Does this sound like cheating? What's the alternative?



# Inductive Learning

Given a collection of examples *(x,f(x))*, return a function *h* that approximates *f*.

*h* is called the **<u>hypothesis</u>** and is chosen from the **<u>hypothesis space</u>**.

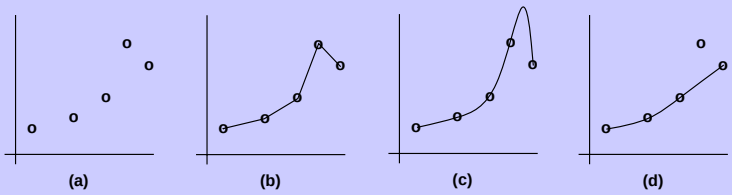
- What if *f* is not in the hypothesis space?

# Inductive Bias: definition

- This "some idea of what to choose from" is called an **inductive bias**.
- Terminology
  - H, **hypothesis space** - a set of functions to choose from
  - C, **concept space** - a set of possible functions to learn
- Often in learning we search for a hypothesis $f$ in H that is consistent with the training examples, i.e., $f(x) = y$ for all training examples $(x,y)$.
- In some cases, any hypothesis consistent with the training examples is likely to generalize to unseen examples. The trick is to find the right bias.



# Which hypothesis?



## Bias explanation

How does learning algorithm decide

**Bias** leads them to prefer one hypothesis over another.

Two types of bias:

- **preference bias** (or search bias) depending on how the hypothesis space is explored, you get different answers
- **restriction bias** (or language bias), the "language" used: Java, FOL, etc. (*h* is not equal to *c*).

- e.g. language: piece-wise linear functions: gives (b)/(d).



## Issues in selecting the bias

Tradeoff (similar in reasoning):

more expressive the language, the harder to find (compute) a good hypothesis.

Compare: propositional Horn clauses with first-order logic theories or Java programs.

- Also, often need more examples.

## Occam's Razor

- Most standard and intuitive preference bias:
  **Occam's Razor**
  (aka **Ockham's Razor)**

*The most likely hypothesis is the **simplest** one that is consistent will all of the observations.*

- Named after Sir William of Ockham.



## Implications

- The world is simple.
- The chances of an accidentally correct explanation are low for a **simple** theory.

# Probably Approximately Correct (PAC) Learning

Two important questions that we have yet to address:

- Where do the training examples come from?
- How do we test performance, i.e., are we doing a good job learning?

- PAC learning is one approach to dealing with these questions.



# Classifier example

Consider learning the predicate *Flies(Z) = { true, false}*.
We are assigning objects to one of two categories: recall we call this a *classifier*.
Suppose that X = {pigeon,dodo,penguin,747}, Y = {true,false}, and that

```
 Pr(pigeon) = 0.3        Flies(pigeon) = true
   Pr(dodo) = 0.1        Flies(dodo) = false
Pr(penguin) = 0.2        Flies(penguin) = false
    Pr(747) = 0.4        Flies(747) = true
```

`Pr` is the distribution governing the presentation of training examples (how often do we see such examples).
We will use the same distribution for evaluation purposes.

- Note that if we mis-classified dodos but got everything else right, then we would still be doing pretty well in the sense that 90% of the time
- we would get the right answer.

- We formalize this as follows.



- The approximate error associated with a hypothesis f is

- error(f) = $\sum$ {x | f(x) not= Flies(x)} Pr(x)

- We say that a hypothesis is
  **<u>approximately correct with error at most ε</u>**

if

error(f) =< ε

- The chances that a theory is correct increases with the number of consistent examples it predicts.

- Or….

- A badly wrong theory will probably be uncovered after only a few tests.



# PAC: definition

Relax this requirement by not requiring that the learning program necessarily achieve a small error but only that it to keep the error small **with high probability**.

Probably approximately correct (PAC) with probability $\delta$ and error at most $\varepsilon$ if, given any set of training examples drawn according to the fixed distribution, the program outputs a hypothesis f such that

$$\Pr(\text{Error}(f) > \varepsilon) < \delta$$

# PAC

- Idea:
- Consider space of hypotheses.
- Divide these into “good” and “bad” sets.
- Want to assure that we can close in on the set of good hypotheses that are close approximations of the correct theory.



# PAC Training examples

Theorem:

If the number of hypotheses $|H|$ is finite, then a program that returns an hypothesis that is consistent with

$$\ln(\delta / |H|)/\ln(1-\varepsilon)$$

training examples (drawn according to Pr) is guaranteed to be PAC with probability $\delta$ and error bounded by $\varepsilon$.

# PAC theorem: proof

If f is not approximately correct then Error(f) > ε so the probability of f being correct on one example is < 1 - ε and the probability of being correct on m examples is $< (1 - \varepsilon)^m$.

Suppose that H = {f,g}. The probability that f correctly classifies all m examples is $< (1 - \varepsilon)^m$. The probability that g correctly classifies all m examples is $< (1 - \varepsilon)^m$. The probability that one of f or g correctly classifies all m examples is < 2 * (1 - ε )^m.

To ensure that any hypothesis consistent with m training examples is correct with an error at most ε with probability δ, we choose m so that 2 * (1 - ε )^m < δ.



Generalizing, there are |H| hypotheses in the restricted hypothesis space and hence the probability that there is some hypothesis in H that correctly classifies all m examples is bounded by

$$|H|(1- \varepsilon)^m.$$

Solving for m in

$$|H|(1- \varepsilon)^m < \delta$$

we obtain

$$m >= \ln(\delta /|H|)/\ln(1- \varepsilon).$$

QED

## Stationarity

- Key assumption of PAC learning:

Past examples are drawn randomly from *the same distribution* as future examples: stationarity.

The number *m* of examples required is called the **<u>sample complexity.</u>**



A class of concepts C is said to be PAC learnable for a hypothesis space H if (roughly) there exists an polynomial time algorithm such that:

for any c in C, distribution Pr, epsilon, and delta,

if the algorithm is given a number of training examples polynomial in 1/epsilon and 1/delta then with probability 1-delta the algorithm will return a hypothesis f from H such that

Error(f) =< epsilon.

# Overfitting

Consider error in hypothesis *h* over:

- training data: $error_{train}(h)$
- entire distribution *D* of data: $error_D(h)$

- Hypothesis *h \in H* **<u>overfits</u>** training data if
  - there is an alternative hypothesis *h' \in H* such that
    - $error_{train}(h) < error_{train}(h')$

but

    - $error_D(h) > error_D(h')$