Announcements

Assignments

- HW10 (programming + “written”)
  - Due Thu 4/30, 11:59 pm

Introduction to Machine Learning

Learning Theory

Instructor: Pat Virtue

Questions For Today

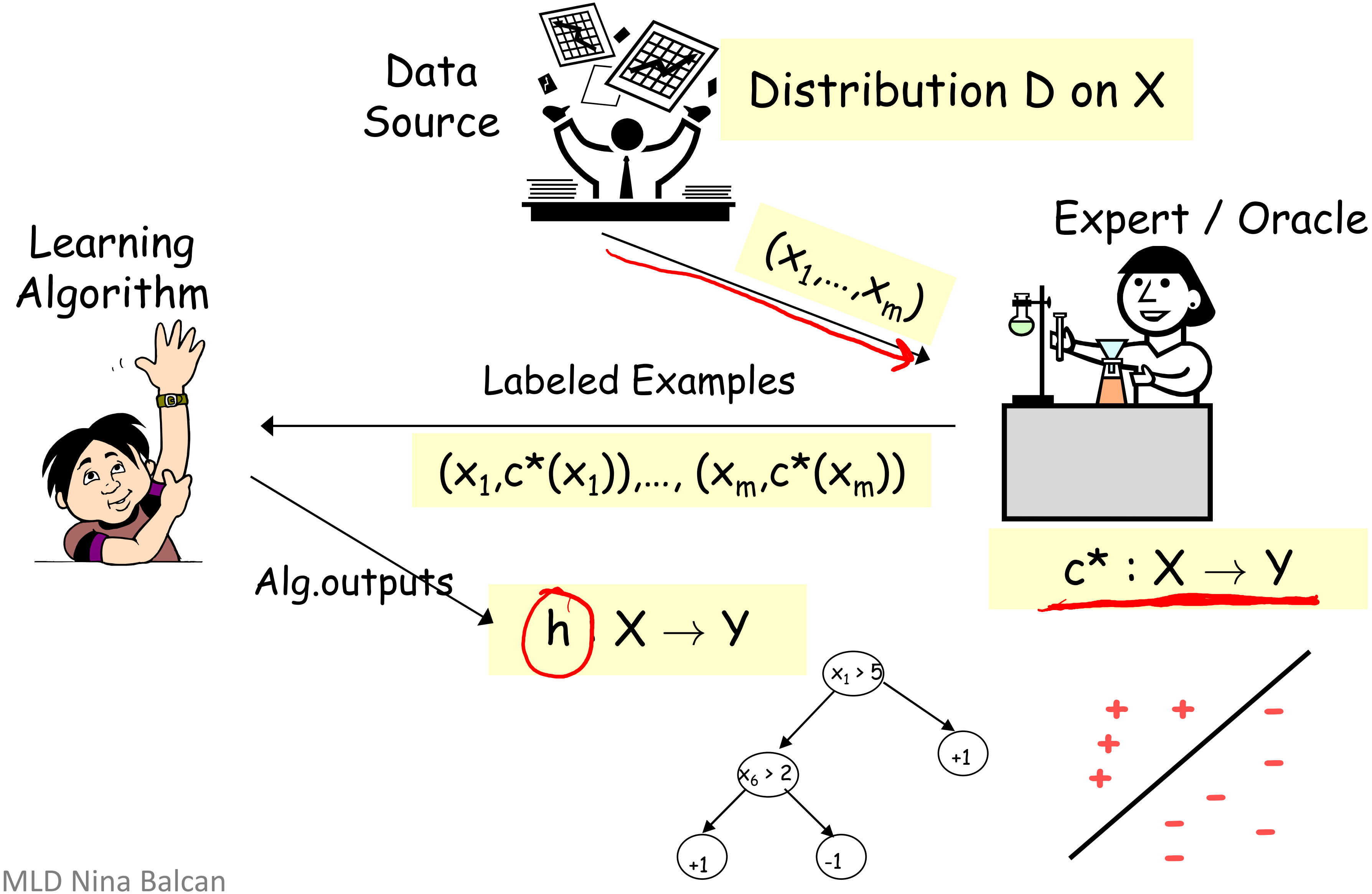
1. Given a classifier with **zero training error**, what can we say about **true error** (aka. generalization error)?
   (Sample Complexity, Realizable Case)

2. Given a classifier with **low training error**, what can we say about **true error** (aka. generalization error)?
   (Sample Complexity, Agnostic Case)

3. Is there a **theoretical justification for regularization** to avoid overfitting?
   (Structural Risk Minimization)

Slide credit: CMU MLD Matt Gormley

Model for Supervised Learning



Slide credit: CMU MLD Nina Balcan

Optimal Classification Function

Find the best $h(x) \to \hat{y}$ by searching in the space of hypothesis functions $h \in \mathcal{H}$.

Optimal classifier:

train

$$h^*(x) = \underset{y}{\operatorname{argmax}} \underbrace{P(Y = y \mid X = x)}$$

But why?

Goal: find a prediction function $h^*: \mathcal{X} \to \mathcal{Y}$ that minimizes the expected loss for randomly drawn test data (X, Y)

$$h^* = \underset{h}{\operatorname{argmin}} \mathbb{E}_{XY}[L(Y, h(X))] \quad R(h) = E \text{ Loss}$$

$L(y, \hat{y})$ is the loss or cost of predicting $\hat{y}$ when the true value is y.

Loss Functions

$$h^* = \underset{h}{\operatorname{argmin}}\ \mathbb{E}_{XY}\left[\underline{L\big(Y, h(X)\big)}\right]$$

Loss function:

$L\colon \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$

Classification:

- Two-class, 0,1 loss

	$\hat{y}=0$	$\hat{y}=1$
$y=0$	0	1
$y=1$	1	0

- Two-class, arbitrary loss

Regression:

Optimal Classification Function

Expected loss is also called risk:

$$R(h) = \mathbb{E}_{XY}\left[L\big(Y, h(X)\big)\right]$$
$$h^* = \underset{h}{\operatorname{argmin}}\, R(h)$$
$$= \underset{h}{\operatorname{argmin}}\, \mathbb{E}_{XY}\left[L\big(Y, h(X)\big)\right]$$

For 0,1 loss classification, risk is also error:

$$R(h) = E_{XY}\left[L(Y, h(X))\right] \leftarrow L(y, \hat{y}) = \begin{cases} 1 & \text{if } h(x) = y \\ 0 & \text{ow} \end{cases} \quad \mathbb{1}(h(x) \neq y)$$

$$= \sum_{x,y} p(x,y)\, L(y, h(x))$$

$$\hat{R}(h) = \sum_{i=1}^{N} \frac{1}{N} \mathbb{1}(y^{(i)} \neq h(x^{(i)})) = \frac{1}{N} \sum \#\text{error}$$

Two Types of Error

1. True Error (aka. **expected risk**)

$$R(h) = P_{\mathbf{x} \sim p^*(\mathbf{x})}(c^*(\mathbf{x}) \neq h(\mathbf{x}))$$

$\hat{y}$

This quantity is always **unknown**

2. Train Error (aka. **empirical risk**)

$$\hat{R}(h) = P_{\mathbf{x} \sim \mathcal{S}}(c^*(\mathbf{x}) \neq h(\mathbf{x}))$$
$$= \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}(c^*(\mathbf{x}^{(i)}) \neq h(\mathbf{x}^{(i)}))$$
$$= \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}(y^{(i)} \neq h(\mathbf{x}^{(i)}))$$

We can **measure** this on the training data

where $\mathcal{S} = \{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(N)})\}_{i=1}^{N}$ is the training data set, and $\mathbf{x} \sim \mathcal{S}$ denotes that $\mathbf{x}$ is sampled from the empirical distribution.

PAC / SLT Model

1. Generate instances from *unknown* distribution p^*

$$\mathbf{x}^{(i)} \sim p^*(\mathbf{x}), \ \forall i \quad (1)$$

2. Oracle labels each instance with *unknown* function c^*

$$y^{(i)} = c^*(\mathbf{x}^{(i)}), \ \forall i \quad (2)$$

3. Learning algorithm chooses hypothesis $h \in \mathcal{H}$ with low(est) training error, $\hat{R}(h)$ $h^* = \text{argmin } R(h)$

$$\hat{h} = \underset{h}{\text{argmin}} \, \hat{R}(h) \quad (3)$$

4. Goal: Choose an h with low generalization error $R(h)$

Slide credit: CMU MLD Matt Gormley

Three Hypotheses of Interest

The **true function** c^* is the one we are trying to learn and that labeled the training data:

$$y^{(i)} = c^*(\mathbf{x}^{(i)}),\ \forall i \tag{1}$$

The **expected risk minimizer** has lowest true error:

$$h^* = \underset{h \in \mathcal{H}}{\operatorname{argmin}}\ R(h)$$

Question:
True or False:
h* and c* are always equal.

The **empirical risk minimizer** has lowest training error:

$$\hat{h} = \underset{h \in \mathcal{H}}{\operatorname{argmin}}\ \hat{R}(h) \tag{3}$$

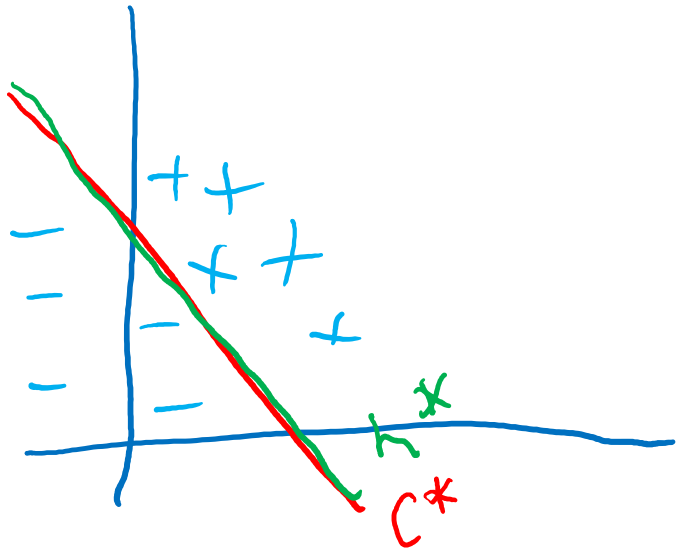
Slide credit: CMU MLD Matt Gormley

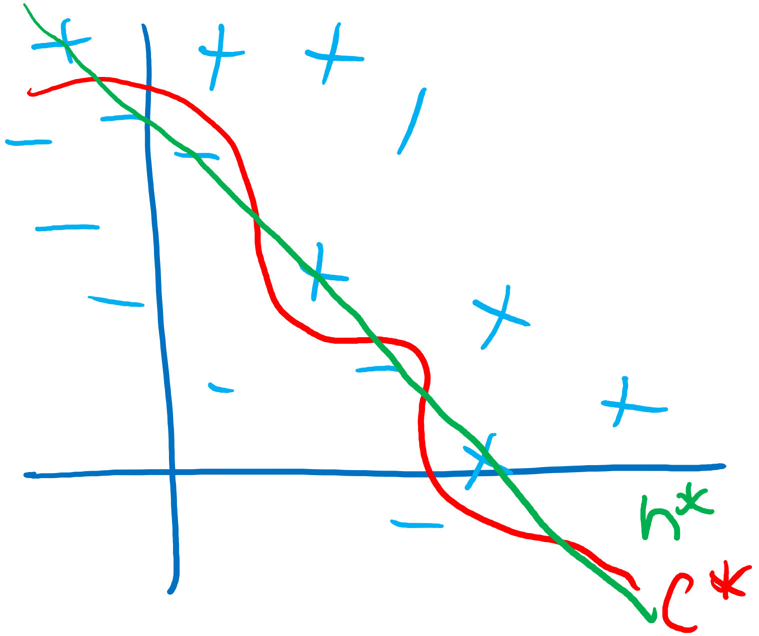
Piazza Poll 1

H linear sep

True or False: h* and c* are always equal.

A. True
B. False
C. Calamity





PAC Learning

unknown known

Can we bound $R(h)$ in terms of $\hat{R}(h)$?

Yes! PAC: Probably Approximately Correct

PAC Learner yield $h \in H$ which is approximately correct: $R(h) \approx 0$

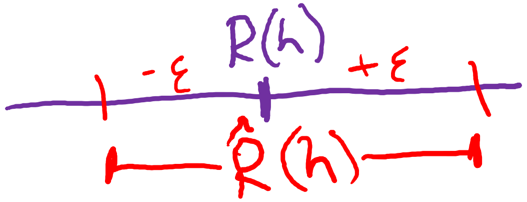
with high probability

$P(R(h) \approx 0) \approx 1$

Definition: PAC Criterion:

small small

$$P\left(\forall h, |R(h) - \hat{R}(h)| \le \varepsilon\right) \ge 1 - \delta$$



Random? $\hat{R}(h)$ based $x_{train} \sim p^*(x)$

PAC Learning

Definition: sample complexity is min. num. of training examples N s.t. the PAC criterion is satisfied for ϵ and δ

$1-\delta = 99\%$

$\epsilon = 0.1\%$

Definition: consistent hypothesis

a $h \in H$ is <u>consistent</u> with training data if $\hat{R}(h) = 0$

PAC Learning

The **PAC criterion** is that our learner produces a high accuracy learner with high probability:

$$P(|R(h) - \hat{R}(h)| \leq \epsilon) \geq 1 - \delta \tag{1}$$

Suppose we have a learner that produces a hypothesis $h \in \mathcal{H}$ given a sample of N training examples. The algorithm is called **consistent** if for every ϵ and δ, there exists a positive number of training examples N such that for any distribution p^*, we have that:

$$P(|R(h) - \hat{R}(h)| > \epsilon) < \delta \tag{2}$$

The **sample complexity** is the minimum value of N for which this statement holds. If N is finite for some learning algorithm, then $\mathcal{H}$ is said to be **learnable.** If N is a polynomial function of $\frac{1}{\epsilon}$ and $\frac{1}{\delta}$ for some learning algorithm, then $\mathcal{H}$ is said to be **PAC learnable.**

PAC Learning

Four types of problems

Two cases c^*

- Realizable: $c^* \in H$
- Agnostic: $c^* \notin H$ or $c^* \in H$

Two cases $|H|$

- Finite $|H| < \infty$
- Infinite $|H| = \infty$

Sample Complexity Results

Definition 0.1. The **sample complexity** of a learning algorithm is the number of examples required to achieve arbitrarily small error (with respect to the optimal hypothesis) with high probability (i.e. close to 1).

Four Cases we care about…

We'll start with the finite case…

	Realizable	Agnostic
Finite $\lvert\mathcal{H}\rvert$	1	2
Infinite $\lvert\mathcal{H}\rvert$	3	4

PAC Learning

Theorem 1: Sample Complexity (Realizable, Finite $|\mathcal{H}|$)

$N \geq \frac{1}{\epsilon}\left(\log |H| + \log 1/\delta\right)$ labelled examples

are sufficient to ensure that with prob $1-\delta$

all $h \in H$ with $\hat{R}(h) = 0$ have $R(h) \leq \epsilon$

PAC Learning

Proof of Theorem 1

{To the whiteboard…}

Sample Complexity Results

Definition 0.1. The **sample complexity** of a learning algorithm is the number of examples required to achieve arbitrarily small error (with respect to the optimal hypothesis) with high probability (i.e. close to 1).

Four Cases we care about...

	Realizable	Agnostic
Finite $\lvert\mathcal{H}\rvert$	**Thm. 1** $N \geq \frac{1}{\epsilon}\left[\log(\lvert\mathcal{H}\rvert) + \log(\frac{1}{\delta})\right]$ labeled examples are sufficient so that with probability $(1-\delta)$ all $h \in \mathcal{H}$ with $\hat{R}(h) = 0$ have $R(h) \leq \epsilon$.	
Infinite $\lvert\mathcal{H}\rvert$		

Piazza Poll 2

Question:

Suppose H = class of conjunctions over **x** in $\{0,1\}^M$

Example hypotheses:

$h(\mathbf{x}) = x_1 (1-x_3) x_5$ $\quad x_1 \wedge \neg x_3 \wedge x_5$

$h(\mathbf{x}) = x_1 (1-x_2) x_4 (1-x_5)$

If M = 10, ε = 0.1, δ = 0.01, how many examples suffice according to Theorem 1?

Answer:

A. 10*(2*ln(10)+ln(100)) ≈ 92
B. 10*(3*ln(10)+ln(100)) ≈ 116
C. 10*(10*ln(2)+ln(100)) ≈ 116
D. 10*(10*ln(3)+ln(100)) ≈ 156
E. ~~100*(2*ln(10)+ln(10)) ≈ 691~~
F. ~~100*(3*ln(10)+ln(10)) ≈ 922~~
G. ~~100*(10*ln(2)+ln(10)) ≈ 924~~
H. ~~100*(10*ln(3)+ln(10)) ≈ 1329~~

$|H| = 3^{10}$

Thm. 1 $N \geq \frac{1}{\epsilon}\left[\log(|\mathcal{H}|) + \log(\frac{1}{\delta})\right]$ labeled examples are sufficient so that with probability $(1-\delta)$ all $h \in \mathcal{H}$ with $\hat{R}(h) = 0$ have $R(h) \leq \epsilon$.

Slide credit: CMU MLD Matt Gormley

Sample Complexity Results

Definition 0.1. The **sample complexity** of a learning algorithm is the number of examples required to achieve arbitrarily small error (with respect to the optimal hypothesis) with high probability (i.e. close to 1).

Four Cases we care about...

	Realizable	Agnostic
Finite $\lvert\mathcal{H}\rvert$	**Thm. 1** $N \geq \frac{1}{\epsilon}\left[\log(\lvert\mathcal{H}\rvert) + \log(\frac{1}{\delta})\right]$ labeled examples are sufficient so that with probability $(1-\delta)$ all $h \in \mathcal{H}$ with $\hat{R}(h) = 0$ have $R(h) \leq \epsilon$.	**Thm. 2** $N \geq \frac{1}{2\epsilon^2}\left[\log(\lvert\mathcal{H}\rvert) + \log(\frac{2}{\delta})\right]$ labeled examples are sufficient so that with probability $(1-\delta)$ for all $h \in \mathcal{H}$ we have that $\lvert R(h) - \hat{R}(h)\rvert \leq \epsilon$.
Infinite $\lvert\mathcal{H}\rvert$		

1. Bound is **inversely linear in epsilon** (e.g. halving the error requires double the examples)
2. Bound is **only logarithmic in |H|** (e.g. quadrupling the hypothesis space only requires double the examples)

1. Bound is **inversely quadratic in epsilon** (e.g. halving the error requires 4x the examples)
2. Bound is **only logarithmic in |H|** (i.e. same as Realizable case)

3^{10}

$10 \log 3$

	Realizable	Agnostic
Finite $\lvert\mathcal{H}\rvert$	**Thm. 1** $N \geq \frac{1}{\epsilon}\left[\log(\lvert\mathcal{H}\rvert) + \log(\frac{1}{\delta})\right]$ labeled examples are sufficient so that with probability $(1-\delta)$ all $h \in \mathcal{H}$ with $\hat{R}(h) = 0$ have $R(h) \leq \epsilon$.	**Thm. 2** $N \geq \frac{1}{2\epsilon^2}\left[\log(\lvert\mathcal{H}\rvert) + \log(\frac{2}{\delta})\right]$ labeled examples are sufficient so that with probability $(1-\delta)$ for all $h \in \mathcal{H}$ we have that $\lvert R(h) - \hat{R}(h)\rvert \leq \epsilon$.
Infinite $\lvert\mathcal{H}\rvert$		

Using a PAC bound

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

- Given ε and δ, yields sample complexity

#training data, $m \geq \dfrac{\ln|H| + \ln\frac{1}{\delta}}{\epsilon}$

- Given m and δ, yields error bound

error, $\epsilon \geq \dfrac{\ln|H| + \ln\frac{1}{\delta}}{m}$

Summary of PAC bounds for finite model classes

With probability $\geq 1-\delta$,

1) For all $h \in H$ s.t. $\text{error}_{\text{train}}(h) = 0$,

$$\text{error}_{\text{true}}(h) \leq \varepsilon = \frac{\ln|H| + \ln\frac{1}{\delta}}{m}$$

Haussler's bound

2) For all $h \in H$

$$|\text{error}_{\text{true}}(h) - \text{error}_{\text{train}}(h)| \leq \varepsilon = \sqrt{\frac{\ln|H| + \ln\frac{2}{\delta}}{2m}}$$

Hoeffding's bound

Slide credit: CMU MLD Aarti Singh and Carlos Guestrin

PAC bound and Bias-Variance tradeoff

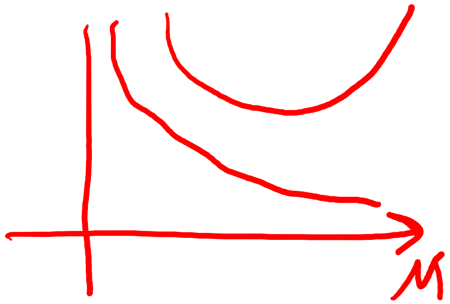
$$P\left(|\text{error}_{true}(h) - \text{error}_{train}(h)| \geq \epsilon\right) \leq 2|H|e^{-2m\epsilon^2} \leq \delta$$

- Equivalently, with probability $\geq 1 - \delta$

$$\underbrace{\text{error}_{true}(h)}_{R(h)} \leq \underbrace{\text{error}_{train}(h)}_{\hat{R}(h)} + \sqrt{\frac{\ln|H| + \ln\frac{2}{\delta}}{2m}}$$

- Fixed |H|

~~Model class~~ Train Size		
m small	small	large
m large	large	small
	bias	variance



Slide credit: CMU MLD Aarti Singh and Carlos Guestrin

PAC bound and Bias-Variance tradeoff

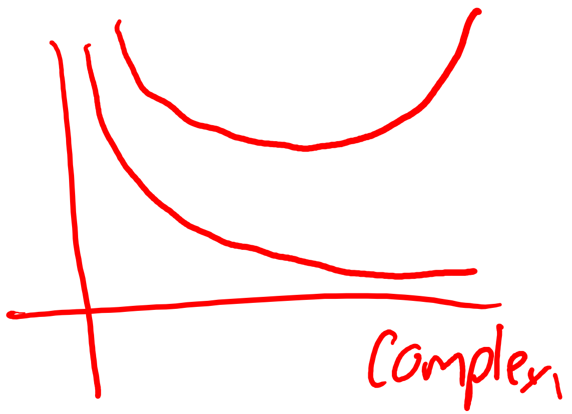
$$P\left(|\text{error}_{true}(h) - \text{error}_{train}(h)| \geq \epsilon\right) \leq 2|H|e^{-2m\epsilon^2} \leq \delta$$

- Equivalently, with probability $\geq 1 - \delta$

$$\text{error}_{true}(h) \leq \underbrace{\text{error}_{train}(h)}_{\downarrow} + \underbrace{\sqrt{\frac{\ln|H| + \ln\frac{2}{\delta}}{2m}}}_{\downarrow}$$

- Fixed m

Model class		
|H| large (complex)	small	large
|H| small (simple)	large	small



Slide credit: CMU MLD Aarti Singh and Carlos Guestrin

Number of decision trees of depth k

$$m \geq \frac{1}{2\epsilon^2}\left(\ln|H| + \ln\frac{2}{\delta}\right)$$

Recursive solution:

Given *n* **binary** attributes

H_k = Number of **binary** decision trees of depth k

$H_0 = 2$

H_k = (#choices of root attribute)

*(# possible left subtrees)

*(# possible right subtrees) $= n * H_{k-1} * H_{k-1}$

Write $L_k = \log_2 H_k$

$L_0 = 1$

$L_k = \log_2 n + 2L_{k-1} = \log_2 n + 2(\log_2 n + 2L_{k-2})$

$= \log_2 n + 2\log_2 n + 2^2\log_2 n + \ldots + 2^{k-1}(\log_2 n + 2L_0)$

So $L_k = (2^k-1)(1+\log_2 n) + 1$

Slide credit: CMU MLD Aarti Singh and Carlos Guestrin

PAC bound for decision trees of depth k

$$m \geq \frac{\ln 2}{2\epsilon^2}\left((2^k - 1)(1 + \log_2 n) + 1 + \log_2 \frac{2}{\delta}\right)$$

- Bad!!!
  - Number of points is exponential in depth k!

- But, for *m* data points, decision tree can't get too big…

Number of leaves never more than number data points

Slide credit: CMU MLD Aarti Singh and Carlos Guestrin

Number of decision trees with k leaves

$$m \geq \frac{1}{2\epsilon^2}\left(\ln |H| + \ln \frac{2}{\delta}\right)$$

H_k = Number of binary decision trees with k leaves

H_1 =2

H_k = (#choices of root attribute) *

[(# left subtrees wth 1 leaf)*(# right subtrees wth k-1 leaves)

+ (# left subtrees wth 2 leaves)*(# right subtrees wth k-2 leaves)

+ …

+ (# left subtrees wth k-1 leaves)*(# right subtrees wth 1 leaf)]

$$H_k = n \sum_{i=1}^{k-1} H_i H_{k-i} = n^{k-1} C_{k-1}$$ (C_{k-1} : Catalan Number)

Loose bound (using Sterling's approximation):

$$H_k \leq n^{k-1} 2^{2k-1}$$

Slide credit: CMU MLD Aarti Singh and Carlos Guestrin

Number of decision trees

- With k leaves

$$m \geq \frac{1}{2\epsilon^2}\left(\ln|H| + \ln\frac{2}{\delta}\right)$$

$$\log_2 H_k \leq (k-1)\log_2 n + 2k - 1$$

linear in k

number of points m is linear in #leaves

- With depth k

$\log_2 H_k = (2^k-1)(1+\log_2 n) + 1$

exponential in k

number of points m is exponential in depth

Slide credit: CMU MLD Aarti Singh and Carlos Guestrin

PAC bound for decision trees with k leaves – Bias-Variance revisited

With prob ≥ 1-δ $\quad \text{error}_{true}(h) \leq \text{error}_{train}(h) + \sqrt{\frac{\ln|H| + \ln\frac{2}{\delta}}{2m}}$

With $H_k \leq n^{k-1}2^{2k-1}$, we get

$$\text{error}_{true}(h) \leq \text{error}_{train}(h) + \sqrt{\frac{(k-1)\ln n + (2k-1)\ln 2 + \ln\frac{2}{\delta}}{2m}}$$

	$\downarrow$	$\downarrow$
k = m	0	large (~ > ½)
k < m	>0	small (~ <½)

Slide credit: CMU MLD Aarti Singh and Carlos Guestrin

What did we learn from decision trees?

- Moral of the story:

  Complexity of learning not measured in terms of size of model space, but in maximum *number of points* that allows consistent classification

Sample Complexity Results

Definition 0.1. The **sample complexity** of a learning algorithm is the number of examples required to achieve arbitrarily small error (with respect to the optimal hypothesis) with high probability (i.e. close to 1).

Four Cases we care about...

	Realizable	Agnostic
Finite $\lvert\mathcal{H}\rvert$	**Thm. 1** $N \geq \frac{1}{\epsilon}\left[\log(\lvert\mathcal{H}\rvert) + \log(\frac{1}{\delta})\right]$ labeled examples are sufficient so that with probability $(1-\delta)$ all $h \in \mathcal{H}$ with $\hat{R}(h) = 0$ have $R(h) \leq \epsilon$.	**Thm. 2** $N \geq \frac{1}{2\epsilon^2}\left[\log(\lvert\mathcal{H}\rvert) + \log(\frac{2}{\delta})\right]$ labeled examples are sufficient so that with probability $(1-\delta)$ for all $h \in \mathcal{H}$ we have that $\lvert R(h) - \hat{R}(h)\rvert \leq \epsilon$.
Infinite $\lvert\mathcal{H}\rvert$		