

## Learning Theory

Hal Daumé III

CS5350: Machine Learning

08 October 2009

1. Why learning theory?
2. The PAC learning model
3. Occam's razor
4. VC dimension

CS5350

Hal Daumé III (U Utah)

1 / 26

CS5350

Hal Daumé III (U Utah)

2 / 26

## Why learning theory?

We've put a lot of effort into **models** of learning and **algorithms** for solving those problems.

How can we tell if we've done a good job?

- ▶ Experimental results
- ▶ Theoretical analysis

Why theory?

- ▶ I can only run so many experiments
- ▶ Experiments rarely tell me what will go wrong
- ▶ I want to be able to deploy my algorithm on Mars

## Why learning theory?

What can we hope for in terms of an algorithm? (Lets even allow ourselves infinite data)

- ▶ Always is optimal?
- ▶ Frequently is optimal?
- ▶ Always is almost-optimal?
- ▶ Frequently is almost-optimal?  $\Leftarrow$  **Our Goal**

**No Free Lunch Theorem** (Wolpert, 2001) states:

*In a **noise-free scenario** where the **loss function** is the **misclassification rate**, if one is **interested in off-training-set error**, then **all algorithms are equivalent, on average**.*

# Why?

CS5350

Hal Daumé III (U Utah)

3 / 26

CS5350

Hal Daumé III (U Utah)

4 / 26

## Formal (classification) problem:

Some terminology:

- ▶ **Hypothesis** ( $h$ ) – the thing we learn
- ▶ **Hypothesis class** ( $\mathcal{H}$ ) – the space of all  $h$
- ▶ **Instance space** ( $\mathcal{X}$ ) – the inputs
- ▶ **Target distribution** ( $\mathcal{D}$ ) – any probability distribution over  $\mathcal{X}$
- ▶ **Target concept** ( $c$ ) – a mapping from  $\mathcal{X}$  to  $\{0, 1\}$ , what we learn

**Given:**  $x_1, \dots, x_N \sim \mathcal{D}$ ,  $y_1 = c(x_1), \dots, y_N = c(x_N)$  and  $\mathcal{H}$

**Find:**  $h \in \mathcal{H}$  to minimize error:

$$\underbrace{\mathbb{E}_{x \sim \mathcal{D}} [\mathbf{1}(c(x) \neq h(x))]}_{\text{expected loss}} = \int_{\mathcal{X}} \mathbf{1}(c(x) \neq h(x)) d\mathcal{D}(x) \\ \neq \underbrace{\sum_n \mathbf{1}(c(x_n) \neq h(x_n))}_{\text{training loss}}$$

CS5350

Hai Daumé III (U Utah)

5 / 26

CS5350

Hai Daumé III (U Utah)

6 / 26

## Probably approximately correct learning

Idea is that I want a learning algorithm that:

- ▶ Finds a solution on most data sets  $\Leftarrow$  "probably"
- ▶ Finds a solution that's close to optimal  $\Leftarrow$  "approximately"

A concept class  $\mathcal{C}$  is **PAC learnable** using  $\mathcal{H}$  if there exists an algorithm  $\mathcal{L}$  with the following property. Choose:

- ▶ Some concept  $c \in \mathcal{C}$
- ▶ Some distribution over inputs,  $\mathcal{D}$  over  $\mathcal{X}$
- ▶ An error parameter  $0 < \epsilon < \frac{1}{2}$
- ▶ A confidence parameter  $0 < \delta < \frac{1}{2}$

Then  $\mathcal{L}$ , given access to labeled examples, must with probability at least  $1 - \delta$  output a hypothesis with error at most  $\epsilon$ .

$\mathcal{C}$  is **efficiently PAC learnable** if  $\mathcal{L}$  needs time polynomial in  $\epsilon^{-1}$ ,  $\delta^{-1}$ .

CS5350

Hai Daumé III (U Utah)

5 / 26

CS5350

Hai Daumé III (U Utah)

6 / 26

## Learning boolean conjunctions

Input space is a sequence of bits. Concept is a conjunction of these bits or negations thereof.

Example:

- ▶  $\mathcal{X} = \{0, 1\}^4$
- ▶  $c(x_1, x_2, x_3, x_4) = x_1 \wedge \neg x_3 \wedge x_4$
- ▶  $c(0, 0, 0, 0) = 0 \wedge 1 \wedge 0 = 0$
- ▶  $c(1, 1, 0, 1) = 1 \wedge 1 \wedge 1 = 1$
- ▶  $c(1, 1, 1, 1) = 1 \wedge 0 \wedge 1 = 0$

**Algorithm:**

- ▶ Initialize  $h = x_1 \wedge \neg x_1 \wedge x_2 \wedge \neg x_2 \wedge \dots \wedge x_D \wedge \neg x_D$
- ▶ Ignore negative examples (those with  $c(x) = 0$ )
- ▶ On positive example with  $x_d = 1$ , remove  $\neg x_d$ ; with  $x_d = 0$ , remove  $x_d$

CS5350

Hai Daumé III (U Utah)

7 / 26

CS5350

Hai Daumé III (U Utah)

8 / 26

## Learning boolean conjunction

Note that  $h$  always contains at least as many literals as  $c$ .

Take some literal  $z$ :

- ▶  $z$  causes  $h$  to err only on positive examples with  $z = 0$
- ▶ these are *exactly* those examples that would "fix" this error
- ▶ let  $p(z)$  be the probability of these happening
- ▶ then the error is  $\leq \sum_{z \in \mathcal{H}} p(z)$
- ▶ define a literal as "bad" if  $p(z) \geq \epsilon / (2D)$
- ▶ no bad literals implies success

**Goal: bound the probability that a bad literal appears.**

**Recall:**  $z$  is bad if  $p(z) \geq \epsilon/(2D)$ , where  $p(z)$  is the probability that we draw a positive example with  $z = 0$ .

- ▶ The probability that **some** bad  $z$  is not deleted is at most  $(1 - \epsilon/(2D))^N$
- ▶ The probability that **any** bad  $z$  is not deleted is at most  $2D(1 - \epsilon/(2D))^N$
- ▶ Bound this by  $\delta$  and apply  $1 - x \leq e^{-x}$

$$N \geq \frac{2D}{\epsilon} \left( \log(2D) + \log \frac{1}{\delta} \right)$$

- ▶ For  $\epsilon = \delta = 0.05$ ,  $D = 20$ , we get  $N \geq 5348$

## On to Occam's razor...

### What and Why of Occam's razor?

- ▶ Named after William of Occam
- ▶ Idea: prefer simple hypotheses that explain the data
- ▶ In learning: want a model that fits the data but has a small representation
- ▶ Main result: Occam algorithms are also PAC algorithms

### Occam's razor and PAC learning

Given:

- ▶ A concept class  $\mathcal{C}$
- ▶ A hypothesis space  $\mathcal{H}$
- ▶ A distribution  $\mathcal{D}$
- ▶ A fixed by unknown concept  $c \in \mathcal{C}$
- ▶  $N$  labeled examples sampled from  $\mathcal{D}$
- ▶ A polytime algorithm  $\mathcal{L}$  that outputs  $h \in \mathcal{H}$  consistent with the sample

Then:  $\mathcal{L}$  PAC-learns the concept class so long as:

$$\log |\mathcal{H}| \leq bN\epsilon - \log \frac{1}{\delta}$$

for some constant  $b$ .

In other words:

$$N \geq \frac{1}{b\epsilon} \left[ \log |\mathcal{H}| + \log \frac{1}{\delta} \right]$$

## Learning boolean conjunctions, revisited

### Remember our Algorithm:

- ▶ Initialize  $h = x_1 \wedge \neg x_1 \wedge x_2 \wedge \neg x_2 \wedge \dots \wedge x_D \wedge \neg x_D$
- ▶ Ignore negative examples (those with  $c(x) = 0$ )
- ▶ On positive with  $x_d = 1$ , remove  $\neg x_d$ ; with  $x_d = 0$ , remove  $x_d$

We obtained  $N \geq (2D/\epsilon)[\log(2D) + \log(1/\delta)]$

Clearly this algorithm is always consistent. Occam says:

$$\begin{aligned} N &\geq \frac{1}{b\epsilon} \left[ \log |\mathcal{H}| + \log \frac{1}{\delta} \right] \\ &= \frac{1}{b\epsilon} \left[ \log(3^D) + \log \frac{1}{\delta} \right] \\ &= \frac{1}{b\epsilon} \left[ D \log 3 + \log \frac{1}{\delta} \right] \end{aligned}$$

## What if we're inconsistent?

Another way of seeing the Occam bound is as a bound on the expected loss of a consistent hypothesis  $h$ :

$$\text{ExpectedLoss}(h) \leq \sqrt{\frac{\log |\mathcal{H}| + \log \frac{1}{\delta}}{2N}}$$

What happens if we don't have a consistent hypothesis? I.e., if  $\text{TrainLoss}(h) > 0$ ?

The obvious thing:

$$\text{ExpectedLoss}(h) \leq \text{TrainLoss}(h) + \sqrt{\frac{\log |\mathcal{H}| + \log \frac{1}{\delta}}{2N}}$$

## Why VC dimension

Recall from PAC learning that Occam's bound states:

$$N \geq \frac{1}{b\epsilon} \left[ \log |\mathcal{H}| + \log \frac{1}{\delta} \right]$$

Where:

- ▶  $N$  is the sample complexity
- ▶  $b$  is some constant
- ▶  $|\mathcal{H}|$  is the size of the hypothesis space
- ▶  $\epsilon, \delta$  are user-set parameters (error and tolerance)

**Big problem:** what happens when  $|\mathcal{H}|$  is infinite (as in most learners)?

**Need a way to measure complexity of infinite hypothesis classes.**

## On to VC dimension...

## Shattering

**Idea:** measure complexity by the effective number of distinct functions that can be defined.

A hypothesis class is more complex if it can represent lots of possible classification decisions.

**Def:** A set of points is *shattered* by  $\mathcal{H}$  if for all possible *binary labelings* of the points, there exists  $h \in \mathcal{H}$  that can represent this decision.

We will measure complexity by the number of points that can be shattered by  $\mathcal{H}$ .

CS5350

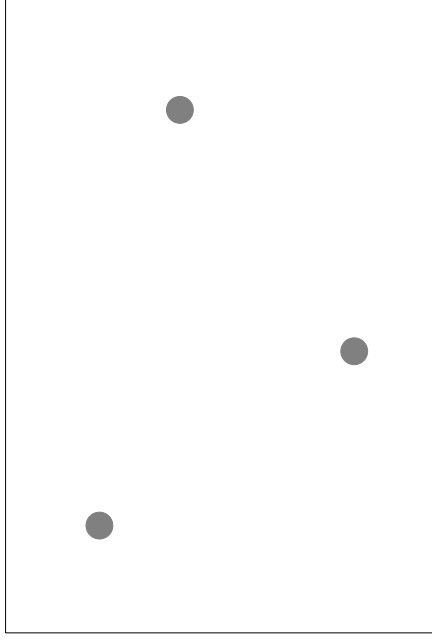
Hai Daumé III (U Utah)

17 / 26

## Shattering

**Def:** A set of points is *shattered* by  $\mathcal{H}$  if for all possible *binary labelings* of the points, there exists  $h \in \mathcal{H}$  that can represent this decision.

Consider  $\mathcal{H} =$ linear classifiers in  $\mathbb{R}^2$



CS5350

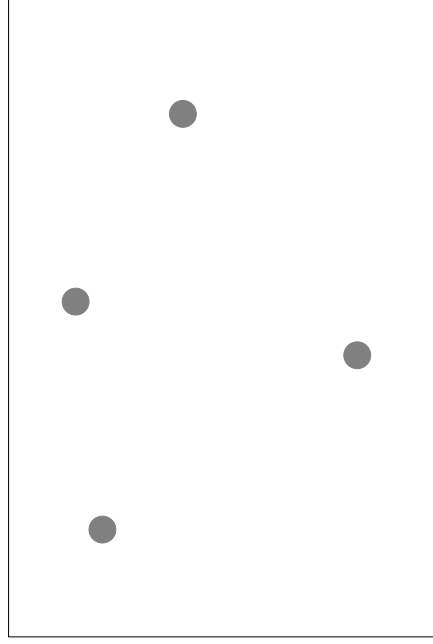
Hai Daumé III (U Utah)

18 / 26

## Shattering

**Def:** A set of points is *shattered* by  $\mathcal{H}$  if for all possible *binary labelings* of the points, there exists  $h \in \mathcal{H}$  that can represent this decision.

Consider  $\mathcal{H} =$ linear classifiers in  $\mathbb{R}^2$



CS5350

Hai Daumé III (U Utah)

19 / 26

## VC Dimension: The Game

VC dimension is a shattering game between us and an adversary.

To show that  $\mathcal{H}$  has VC dimension  $\geq d$ :

1. I choose  $d$  points positioned however I want
2. **Adversary chooses** a labeling of these  $d$  points
3. I choose  $h \in \mathcal{H}$  that labels the points properly

The VC dimension of  $\mathcal{H}$  is the maximum  $d$  I can choose so that I can always succeed in the game.

We just (informally) showed that the VC dimension of linear classifiers in  $\mathbb{R}^2$  is ...3

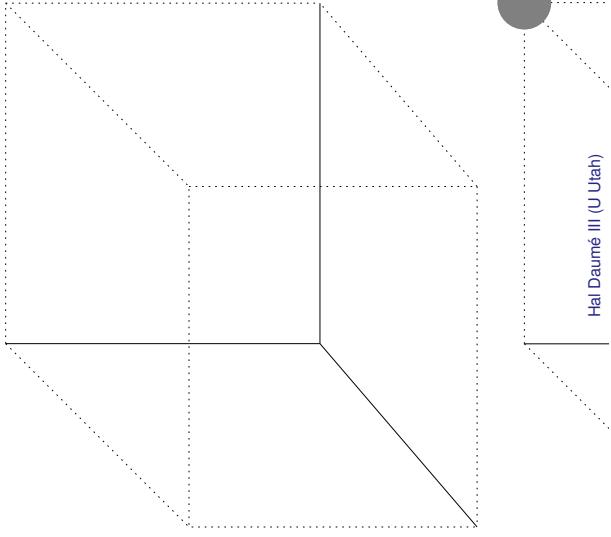
CS5350

Hai Daumé III (U Utah)

20 / 26

## VC Dimension in $\mathbb{R}^3$

What about linear classifiers in  $\mathbb{R}^3$ ?



CS5350

Hai Daumé III (U Utah)

21 / 26

CS5350

Hai Daumé III (U Utah)

22 / 26

## VC Dimension in $\mathbb{R}^D$

What about linear classifiers in  $\mathbb{R}^3$ ?

$VC = 4$  seems like a reasonable guess. . .

What about  $\mathbb{R}^D$ ? Yup,  $VC = D + 1$

What about the VC dimension of a 1-nearest neighbor classifier?

. . . or an SVM with an RBF kernel?

## VC Dimension and Generalization Bounds

Recall from PAC learning that Occam's bound states:

$$\text{ExpectedLoss}(h) \leq \text{TrainLoss}(h) + \sqrt{\frac{\log |\mathcal{H}| + \log \frac{1}{\delta}}{2N}}$$

If  $\mathcal{H}$  is infinite, but has finite VC-dimension  $d$ , then:

$$\text{ExpectedLoss}(h) \leq \text{TrainLoss}(h) + \sqrt{\frac{d \left( \log \frac{2N}{d} + 1 \right) + \log \frac{4}{\delta}}{N}}$$

Again, for linear classifiers, having fewer features is better!

CS5350

Hai Daumé III (U Utah)

23 / 26

CS5350

Hai Daumé III (U Utah)

24 / 26

## Relationship to SVMs. . .

**Thm** (Vapnik, 1982)

Given:

- ▶  $N$  data points  $X = \{x_1, \dots, x_N\}$ ,
- ▶ Living in  $\mathbb{R}^D$ ,
- ▶ All with  $\|x_n\| \leq R$
- ▶  $\mathcal{H}_\gamma =$  set of linear classifiers in  $\mathbb{R}^D$  with margin  $\gamma$  on  $X$

Then, the VC dimension of  $\mathcal{H}_\gamma$  is bounded by:

$$VC(\mathcal{H}_\gamma) \leq \min \left\{ D, \left\lceil \frac{4R^2}{\gamma^2} \right\rceil \right\}$$

Vapnik tells us:

$$VC(\mathcal{H}_\gamma) \leq \min \left\{ D, \left\lceil \frac{4R^2}{\gamma^2} \right\rceil \right\}$$

This means that hypothesis spaces with large margin have small VC dimension.

From before, we know that:

$$\text{ExpectedLoss}(h) \leq \text{TrainLoss}(h) + \sqrt{\frac{VC(\mathcal{H}_\gamma) \left( \log \frac{2N}{VC(\mathcal{H}_\gamma)} + 1 \right) + \log \frac{4}{\delta}}{N}}$$

Together, this means that large margin implies good generalization!  
Hooray!

- ▶ No universally good learning algorithm exists
- ▶ We care about *expected error*, not *training error*
- ▶ The measure of importance is *sample complexity*
- ▶ We can often bound expected error by using the fact that “if it were important, we would have seen it already”
- ▶ Occam’s razor tells us that **consistency** and **small  $\mathcal{H}$**  is sufficient for generalization – we need  $\mathcal{O}((1/\epsilon)[\log |\mathcal{H}| + \log(1/\delta)])$  examples
- ▶ Standard PAC bounds only apply to finite hypothesis classes
- ▶ VC dimension is a measure of complexity of infinite hypothesis classes
- ▶ Based on the idea of shattering: how many points can we always correctly classify?
- ▶ Generalization now scales in terms of VC dimension
- ▶ Large margins imply small VC dimension