

Machine Learning: Model Class Selection

15-381/781

Fall 2016

Emma Brunskill & Ariel Procaccia

Material includes slides from and modified from Manuela Veloso, Andrew Moore, Dan Klein, Tom Mitchell and Ariel Procaccia

Carnegie Mellon

A supervised learning pipeline

Training Data

$$\begin{pmatrix} 2 \\ 0 \\ 8 \\ 5 \\ \vdots \end{pmatrix}, \begin{pmatrix} 2 \\ 0 \\ 8 \\ 5 \\ \vdots \end{pmatrix}$$

Machine Learning

→ Hypothesis
function
 h_{θ}

Deployment

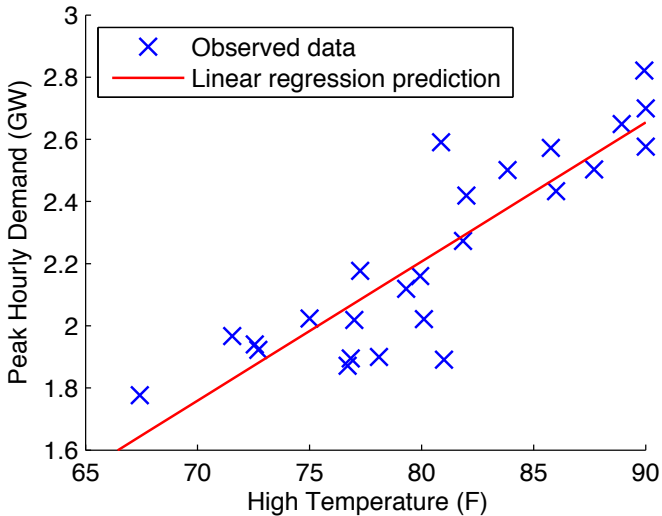
$$\begin{aligned} \text{Prediction} &= h_{\theta} \begin{pmatrix} 2 \end{pmatrix} \\ \text{Prediction} &= h_{\theta} \begin{pmatrix} 5 \end{pmatrix} \\ &\vdots \end{aligned}$$

A simple example: predicting electricity use

What will peak power consumption be in the Pittsburgh area tomorrow?

Collect data of past high temperatures and peak demands

High Temperature (F)	Peak Demand (GW)
76.7	1.87
72.7	1.92
71.5	1.96
86.0	2.43
90.0	2.69
87.7	2.50
⋮	⋮



A supervised learning pipeline

Training Data

$$\left(\begin{array}{c} \text{2} \\ \end{array} , 2 \right)$$

$$\left(\begin{array}{c} \text{0} \\ \end{array} , 0 \right)$$

$$\left(\begin{array}{c} \text{8} \\ \end{array} , 8 \right)$$

$$\left(\begin{array}{c} \text{5} \\ \end{array} , 5 \right)$$

⋮

Machine Learning

→ Hypothesis
function
 h_{θ}

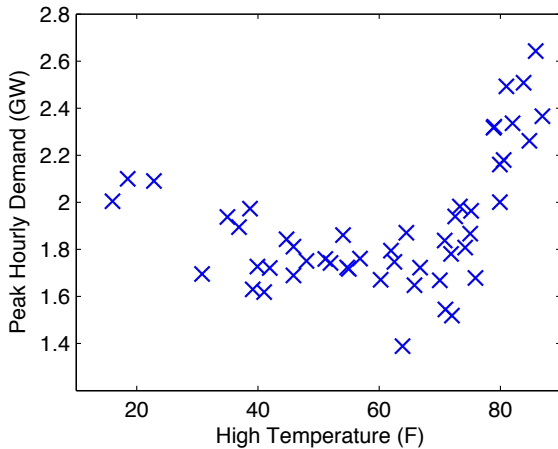
Deployment

$$\text{Prediction} = h_{\theta} \left(\begin{array}{c} \text{2} \\ \end{array} \right)$$

$$\text{Prediction} = h_{\theta} \left(\begin{array}{c} \text{5} \\ \end{array} \right)$$

⋮

Last time: Given a hypothesis class and data,
how fit parameters theta? Analytically, gradient descent,...



Several days of peak demand vs. high temperature in Pittsburgh over all months

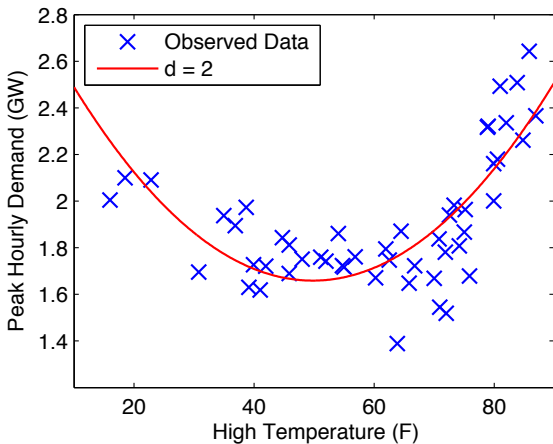
Overfitting

Though they may seem limited, linear hypothesis classes are very powerful, since the input features can themselves include non-linear features of data

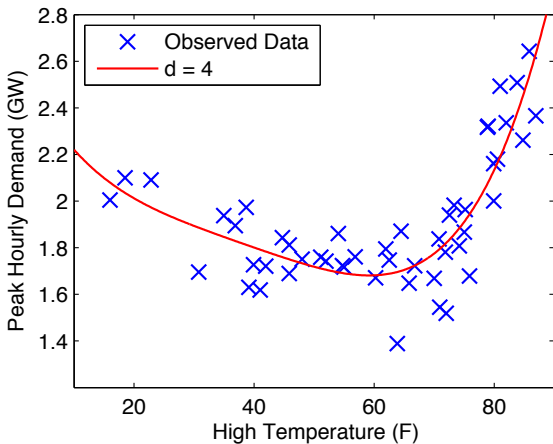
$$x^{(i)} \in \mathbb{R}^3 = \begin{bmatrix} (\text{high temperature for day } i)^2 \\ \text{high temperature for day } i \\ 1 \end{bmatrix}$$

In this case, $h_{\theta}(x) = x^T \theta$ will be a non-linear function of “original” data (i.e., predicted peak power is a non-linear function of high temperature)

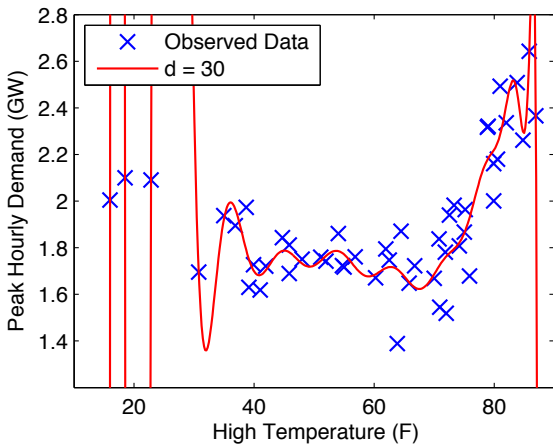
Same solution method as before, gradient descent or (for squared loss) analytical solution



Linear regression with second degree polynomial features



Linear regression with fourth degree polynomial features



Linear regression with 30th degree polynomial features

What Hypothesis/ Model Class Should We Choose?

Training and validation loss

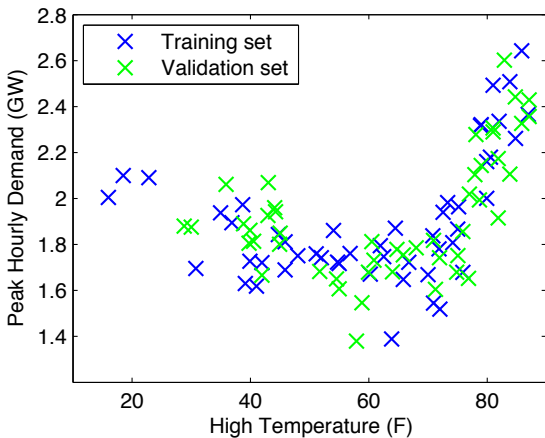
Fundamental problem: we are looking for parameters that optimize

$$\underset{\theta}{\text{minimize}} \sum_{i=1}^m \ell(h_{\theta}(x^{(i)}), y^{(i)})$$

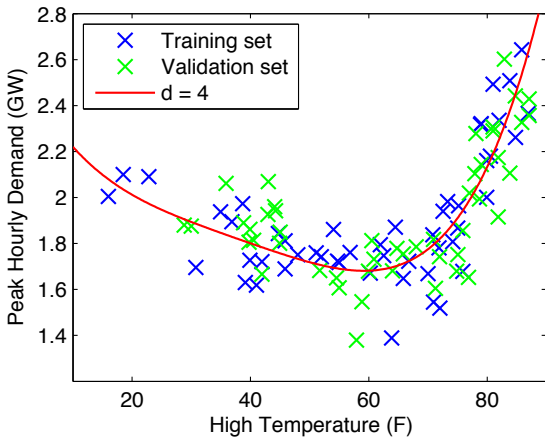
but what we really care about is loss of prediction on *new* examples (x', y') (also called *generalization error*)

Divide data into *training set* (used to find parameters for a fixed hypothesis class h_{θ}), and *validation set* (used to choose hypothesis class)

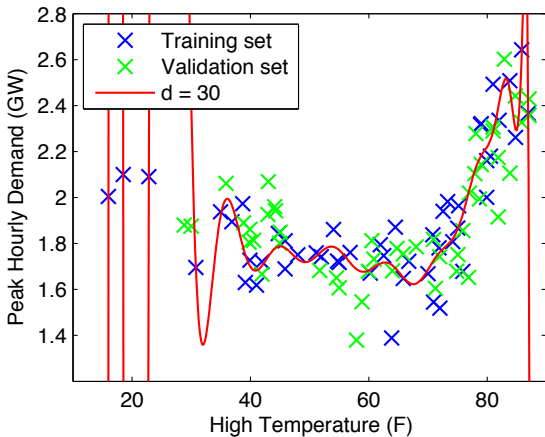
- (Slightly abusing notation here, we're going to wrap the "degree" of the input features into the hypothesis class h_{θ})



Training set and validation set

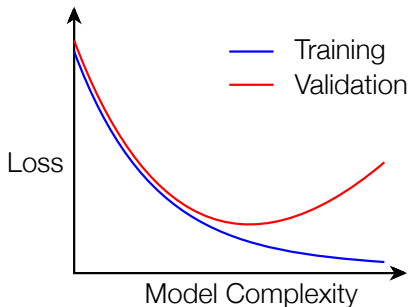


Training set and validation set, fourth degree polynomial

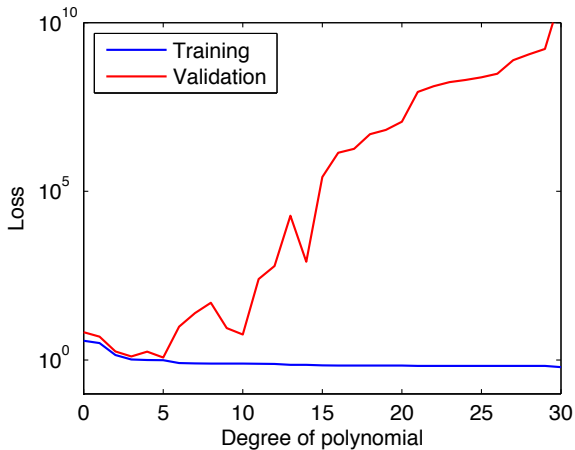


Training set and validation set, 30th degree polynomial

General intuition for training and validation loss



We would like to choose hypothesis class that is at the “sweet spot” of minimizing validation loss



Training and validation loss on peak demand prediction

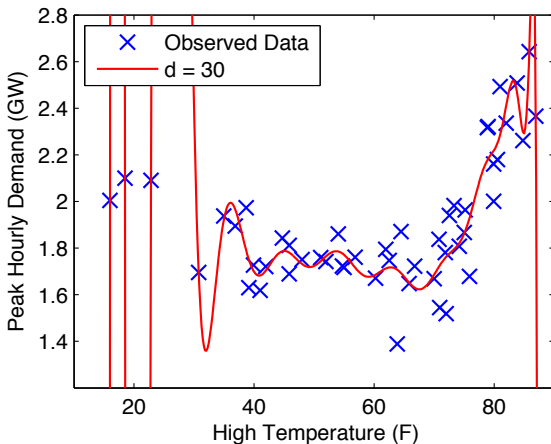
Model complexity and regularization

A number of different ways to control “model complexity”

An obvious one we have just seen: keep the number of features (number of parameters) low

A less obvious method: keep the *magnitude* of the parameters small

Intuition: a 30th degree polynomial that passes exactly through many of the data points requires very large entries in θ

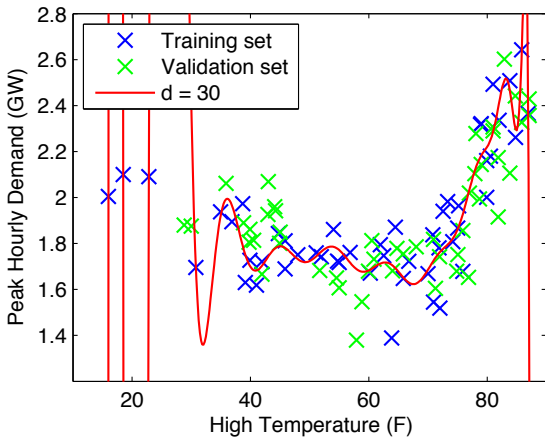


We can directly prevent large entries in θ by penalizing the magnitude of its entries

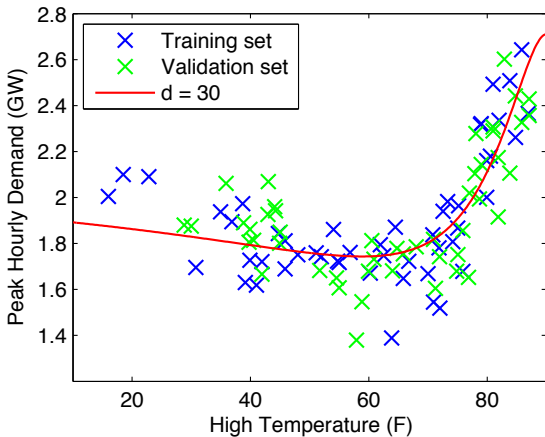
Leads to *regularized loss minimization* problem

$$\underset{\theta}{\text{minimize}} \sum_{i=1}^m \ell \left(h_{\theta}(x^{(i)}), y^{(i)} \right) + \lambda \sum_{i=1}^n \theta_i^2$$

where $\lambda \in \mathbb{R}_+$ is a *regularization parameter* that weights the relative penalties of the size of θ and the loss



Degree 30 polynomial, with $\lambda = 0$ (unregularized)



Degree 30 polynomial, with $\lambda = 1$

Evaluating ML algorithms

The proper way to evaluate an ML algorithm:

1. Break all data into training/testing sets (e.g., 70%/30%)
2. Break training set into training/validation set (e.g., 70%/30% again)
3. Choose hyperparameters using validation set
4. (Optional) Once we have selected hyperparameters, retrain using all the training set
5. Evaluate performance on the testing set

Generalization Bounds & Sample Complexity Bounds

- Generalization Bound
 - Given a finite amount of data, if we learn a classifier, can we have a guarantee on how well that classifier will do on future data? (Bound on generalization error, e.g. accuracy on the future)?

Generalization Bounds & Sample Complexity Bounds

- Generalization Bound
 - Given a finite amount of data, if we learn a classifier, can we have a guarantee on how well that classifier will do on future data? (Bound on generalization error, e.g. accuracy on the future)?
- Sample Complexity
 - If we require a bound on the the classifier accuracy on future data points, can we bound how many training samples we need to get such a classifier?

THE PAC MODEL

- **PAC** = probably approximately correct
- Introduced by Valiant [1984]
- Learner can do well on training set but badly on new samples
- Establish guarantees on accuracy of learner when **generalizing** from examples

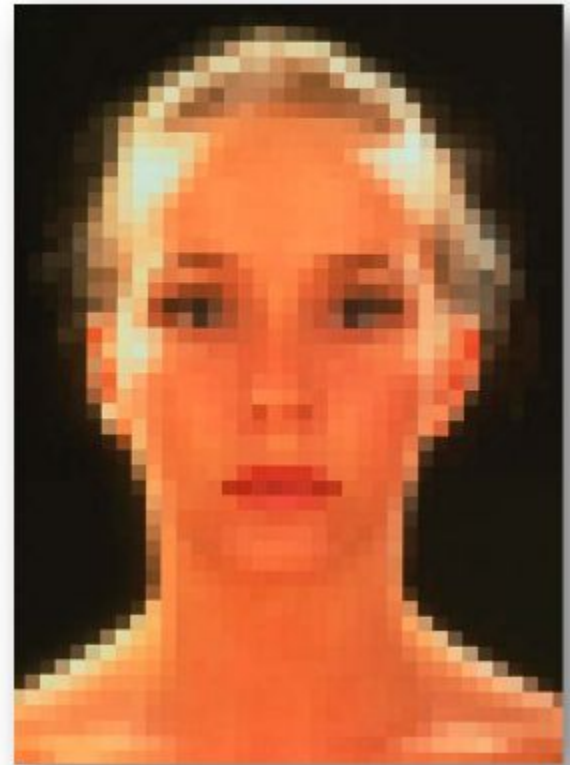


THE PAC MODEL

- **Input space** X
- D distribution over X : unknown but fixed
- Learner receives a set S of m instances x_1, \dots, x_m , independently sampled according to D
- **Concept class** C of functions $h: X \rightarrow \{+, -\}$
- Assume **target function** $c_t \in C$
- **Training examples** $Z = \{(x_i, c_t(x_i))\}$

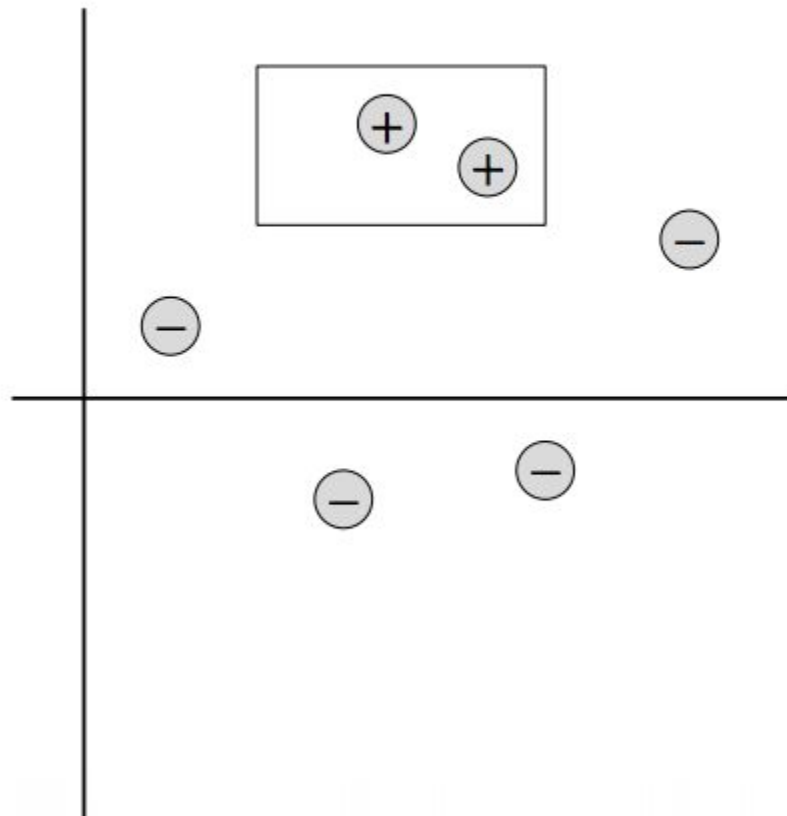
EXAMPLE: FACES

- $X = \mathbb{R}^n$
- Each $x \in X$ is a vector of colors, one per pixel
- $c_t(x) = +$ iff x is a picture of a face
- Training examples: Each is a picture labeled “face” or “not face”



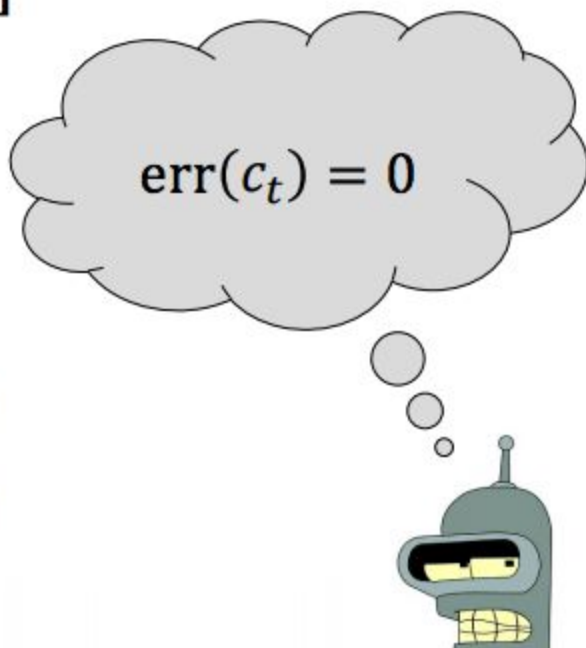
EXAMPLE: RECTANGLE LEARNING

- $X = \mathbb{R}^2$
- $C =$ axes-aligned rectangles
- $h(x) = +$ iff x is contained in h



THE PAC MODEL

- The **error** of concept h is
$$\text{err}(h) = \Pr_{x \sim D} [x: c_t(x) \neq h(x)]$$
- Given **accuracy** parameter $\epsilon > 0$, would like to find concept h with $\text{err}(h) \leq \epsilon$
- Given **confidence** parameter $\delta > 0$, would like to achieve $\Pr[\text{err}(h) \leq \epsilon] \geq 1 - \delta$



THE PAC MODEL

- A **learning algorithm** L is a function from training examples to \mathcal{C} such that: for every $\epsilon, \delta > 0$ there exists $m_0(\epsilon, \delta)$ such that for every $m \geq m_0$ and every D , if m examples Z are drawn from D and $L(Z) = h$ then

$$\Pr[\text{err}(h) \geq \epsilon] \leq 1 - \delta$$

- \mathcal{C} is **learnable** if there is a learning algorithm for \mathcal{C}

$m_0(\epsilon, \delta)$ is
independent of D !



THE PAC MODEL

- A **learning algorithm** L is a function from training examples to \mathcal{C} such that: for every $\epsilon, \delta > 0$ there exists $m_0(\epsilon, \delta)$ such that for every $m \geq m_0$ and every D , if m examples Z are drawn from D and $L(Z) = h$ then

$$\Pr[\text{err}(h) \geq \epsilon] \leq 1 - \delta$$

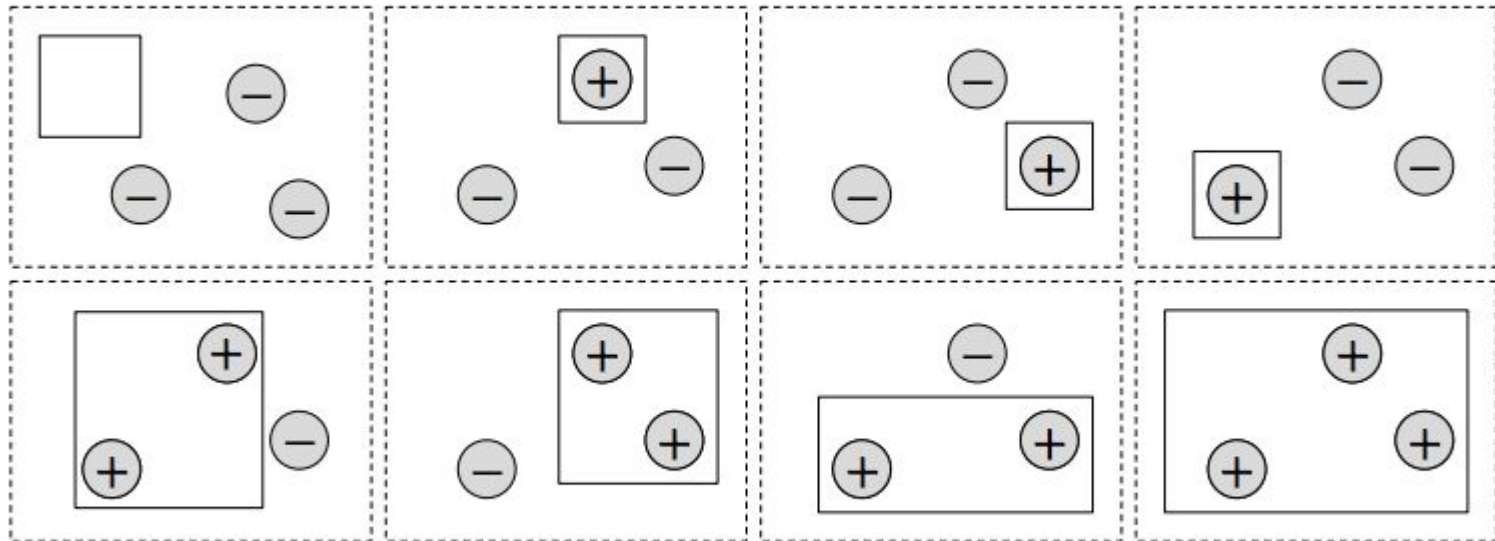
Or in reverse, if we have such a L , it tells us how much data we need to learn a classifier that is PAC

VC DIMENSION

- We would like to obtain a more general result
- Let $S = \{x_1, \dots, x_m\}$
- $\Pi_C(S) = \{(h(x_1), \dots, h(x_m)) \mid h \in C\}$

Ways can label the points using classifiers
in the set C

VC DIMENSION

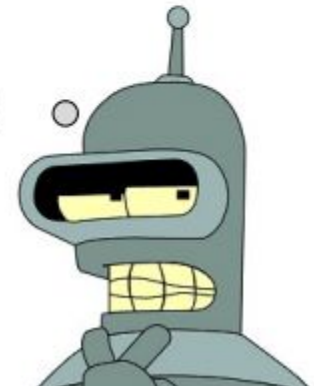


$$\Pi_C(S) = \{(-, -, -), (-, +, -), (-, -, +), (+, -, -), \\ (+, +, -), (-, +, +), (+, -, +), (+, +, +)\}$$

VC DIMENSION

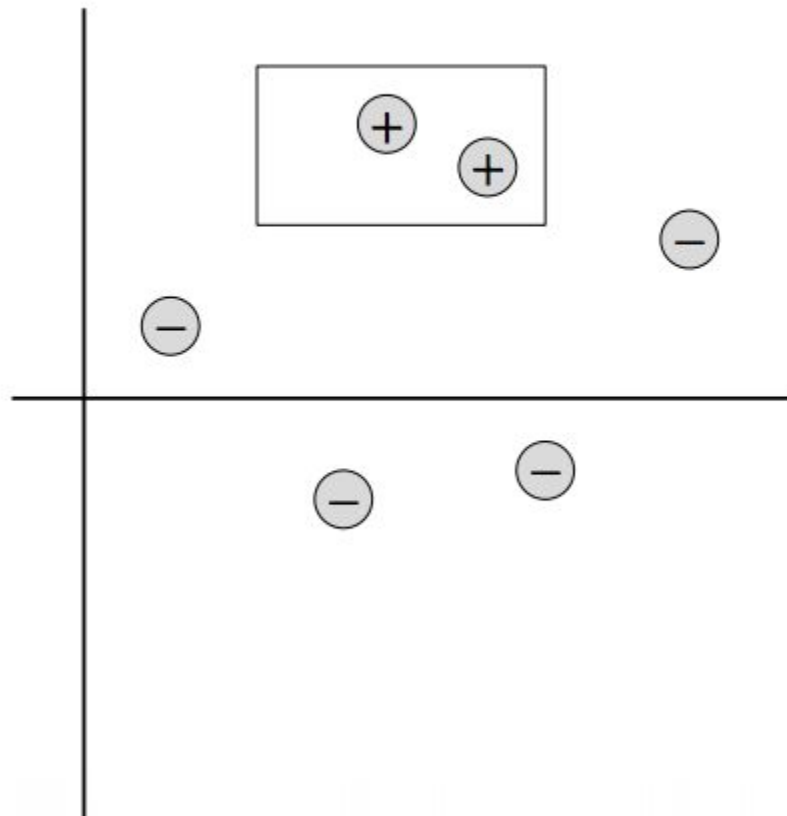
- S is **shattered** by \mathcal{C} if $|\Pi_{\mathcal{C}}(S)| = 2^m$
- The **VC dimension** of \mathcal{C} is the cardinality of the largest set that is shattered by \mathcal{C}

How do we
prove upper and
lower bounds?



EXAMPLE: RECTANGLE LEARNING

- $X = \mathbb{R}^2$
- $C =$ axes-aligned rectangles
- $h(x) = +$ iff x is contained in h



Proving VC Dimension

- Lower Bound
 - Want to show VC dimension is at least N
 - To do so it suffices to provide a set of N points that no matter how they are labeled, we can always create a classifier that correctly labels them
 - E.g. this set of N points is shattered

Proving VC Dimension

- Lower Bound
 - Want to show VC dimension is at least N
 - To do so it suffices to provide a set of N points that no matter how they are labeled, we can always create a classifier that correctly labels them
 - E.g. this set of N points is shattered
- Upper Bound
 - Want to show VC dimension is less than $N+1$
- Show that upper and lower bound yield a unique VC dimension

To show $d_{vc} < D+1$

- We need to show that

- 1) There are $D+1$ points we can't shatter
- 2) There are $D+2$ points we can't shatter
- 3) We cannot shatter any set of $D+1$ points
- 4) We cannot shatter any set of $D+2$ points
- 5) Not sure

VC DIMENSION

- **Poll 3:** $X =$ real line, $\mathcal{C} =$ intervals, what is $\text{VC-dim}(\mathcal{C})$?

1 3

2 ∞

- **Poll 4:** $X =$ real line, $\mathcal{C} =$ unions of intervals, what is $\text{VC-dim}(\mathcal{C})$?

2 4

3 ∞

SAMPLE COMPLEXITY

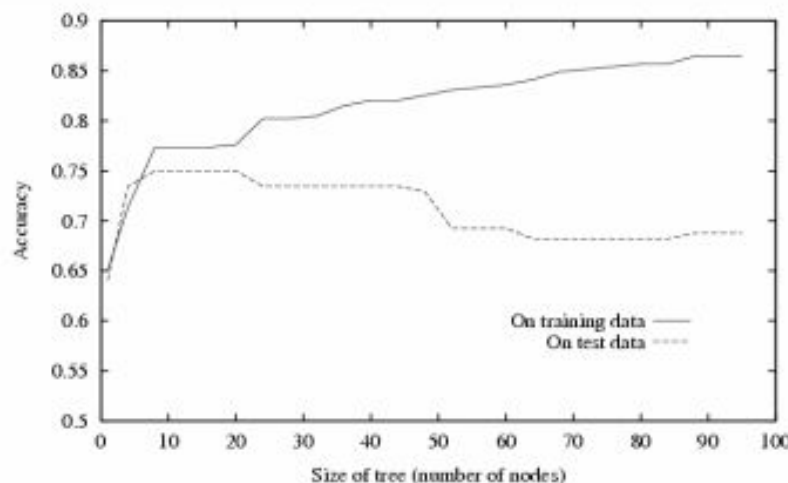
- **Theorem:** a concept class \mathcal{C} with $\text{VC-dim}(\mathcal{C}) = \infty$ is not PAC learnable
- **Theorem:** Let \mathcal{C} with $\text{VC-dim}(\mathcal{C}) = d$. Let L be an algorithm that produces an $h \in \mathcal{C}$ that is **consistent** with the given samples S . Then L is a learning algorithm for \mathcal{C} with $m_0 = c_0 \left(\frac{1}{\epsilon} \log \frac{1}{\delta} + \frac{d}{\epsilon} \log \frac{1}{\epsilon} \right)$

Agnostic Learning: VC Bounds

[Schölkopf and Smola, 2002]

With probability at least $(1-\delta)$ every $h \in H$ satisfies

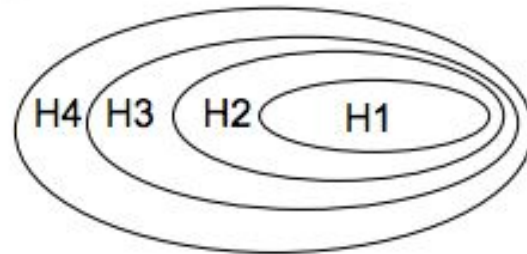
$$error_{true}(h) < error_{train}(h) + \sqrt{\frac{VC(H)(\ln \frac{2m}{VC(H)} + 1) + \ln \frac{4}{\delta}}{m}}$$



Structural Risk Minimization [Vapnik]

Which hypothesis space should we choose?

- Bias / variance tradeoff













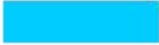
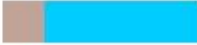






SRM: choose H to minimize bound on true error!

$$error_{true}(h) < error_{train}(h) + \sqrt{\frac{VC(H)(\ln \frac{2m}{VC(H)} + 1) + \ln \frac{4}{\delta}}{m}}$$

* unfortunately a somewhat loose bound...

Structural Risk Minimization

$$error_{true}(h) < error_{train}(h) + \sqrt{\frac{VC(H)(\ln \frac{2m}{VC(H)} + 1) + \ln \frac{4}{\delta}}{m}}$$

i	f_i	TRAINERR	VC-Confidence	Probable upper bound on TESTERR	Choice
1	f_1				
2	f_2				
3	f_3				⊗
4	f_4				
5	f_5				
6	f_6				

ML Model Class Selection: What You Should Know

- Define training error, generalization error and model selection problem
- Be able to apply training set partitioning to both identify a model class expect to yield good generalization error, and provide an estimation of that generalization error (and explain why this procedure is reasonable)
 - Empirical approach:
 - partition data
- Define VC dimension and be able to
 - Prove the VC dimension of a particular model class
 - Use it to obtain a bound on the generalization error
 - Know how many data points are needed to learn a PAC classifier as a function of VC

Online Resources

<http://www.autonlab.org/tutorials/vcdim08.pdf>

<http://www.cs.cmu.edu/~awm/10701/slides/PAC-learning-10-25-05.pdf>

<https://alliance.seas.upenn.edu/~cis520/wiki/index.php?n=Recitations.VCDim>

<http://web.engr.oregonstate.edu/~xfern/classes/cs534/midterm-solutions-07.pdf>

<http://www.cs.cmu.edu/~gustrin/Class/10701/slides/learningtheory-bigpicture.pdf>