Learning Theory
18.5

Family Cars

Learning a Class from Examples

- Class C of a "family car"
  - Prediction: Is car *x* a family car?
  - Knowledge extraction: What do people expect from a family car?
- Output:

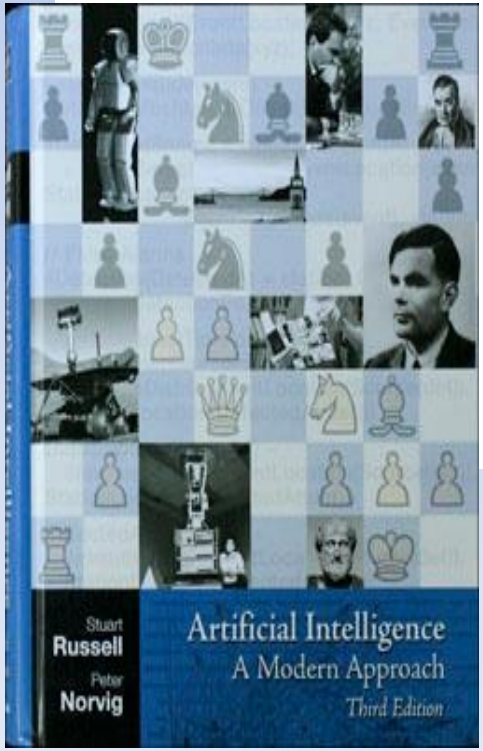
  Positive (+) and negative (–) examples
- Input representation:

  x_1: price, x_2 : engine power

Theory of Learning

18 LEARNING FROM EXAMPLES

18.5 THE THEORY OF LEARNING

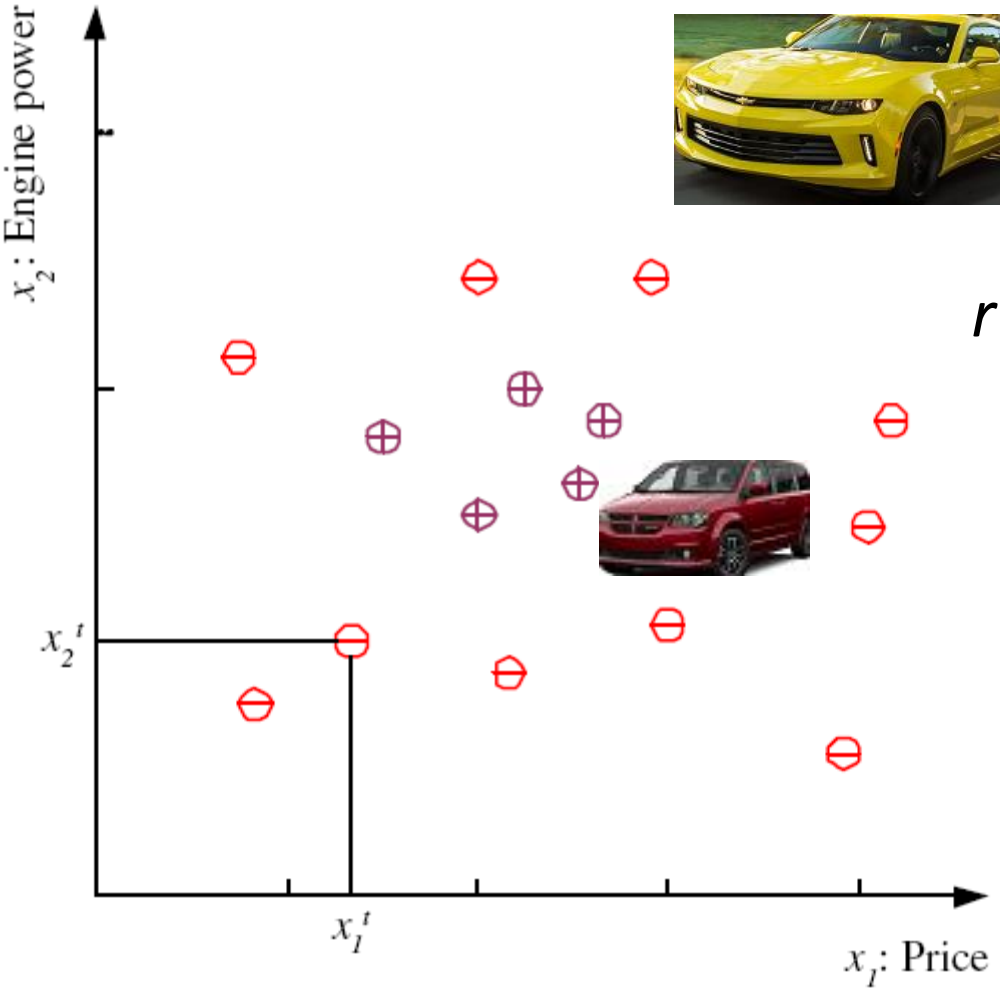


The main unanswered question in learning is this: How can we be sure that our learning algorithm has produced a hypothesis that will predict the correct value for previously unseen inputs? In formal terms, how do we know that the hypothesis h is close to the target function f if we don't know what f is? These questions have been pondered for several centuries. In more recent decades, other questions have emerged: how many examples do we need to get a good h? What hypothesis space should we use? If the hypothesis space is very complex, can we even find the best h, or do we have to settle for a local maximum in the

18.5: The Theory of Learning

- What statements can we prove about our learning algorithms:
  - How do we know that the hypothesis h is close to the target function f if we don't know what f is?
  - How many examples do we need to get a good h?
  - What hypothesis space should we use?
  - If the hypothesis space is very complex, can we even find the best h, or do we have to settle for a local maximum in the space of hypotheses?
  - How complex should h be?
  - How do we avoid overfitting?

Training set $\mathcal{X}$



$$\mathcal{X} = \{\mathbf{x}^t, r^t\}_{t=1}^N$$

$$r = \begin{cases} 1 \text{ if } \mathbf{x} \text{ is positive} \\ 0 \text{ if } \mathbf{x} \text{ is negative} \end{cases}$$

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

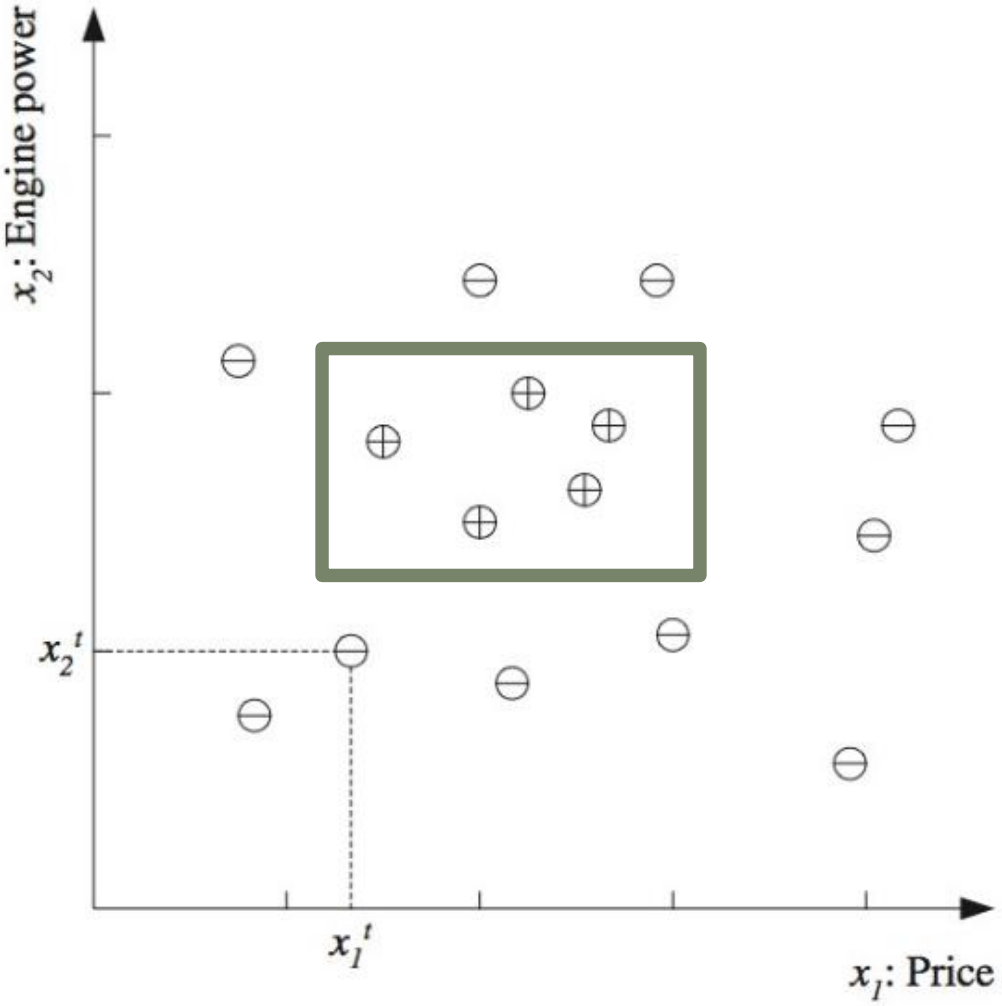
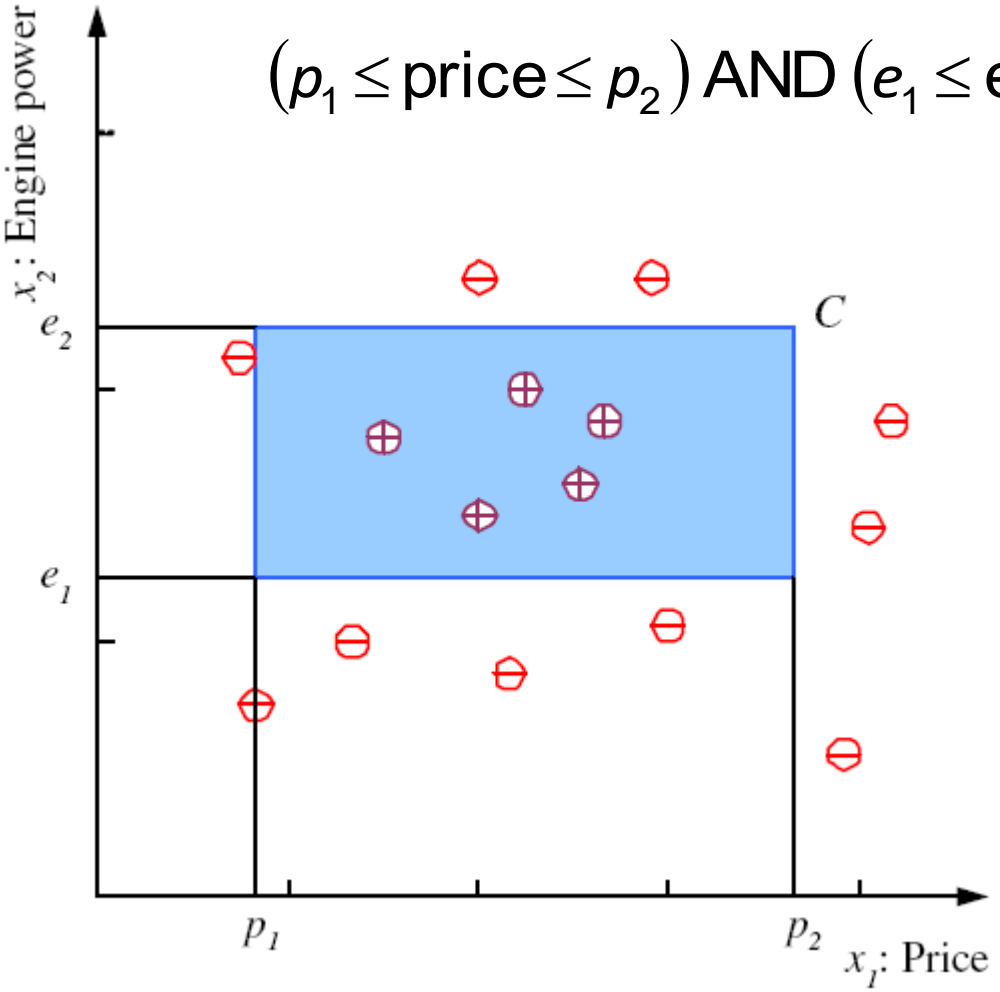


Figure 2.1 Training set for the class of a "family car." Each data point corresponds to one example car, and the coordinates of the point indicate the price and engine power of that car. '+' denotes a positive example of the class (a family car), and '–' denotes a negative example (not a family car); it is another type of car.

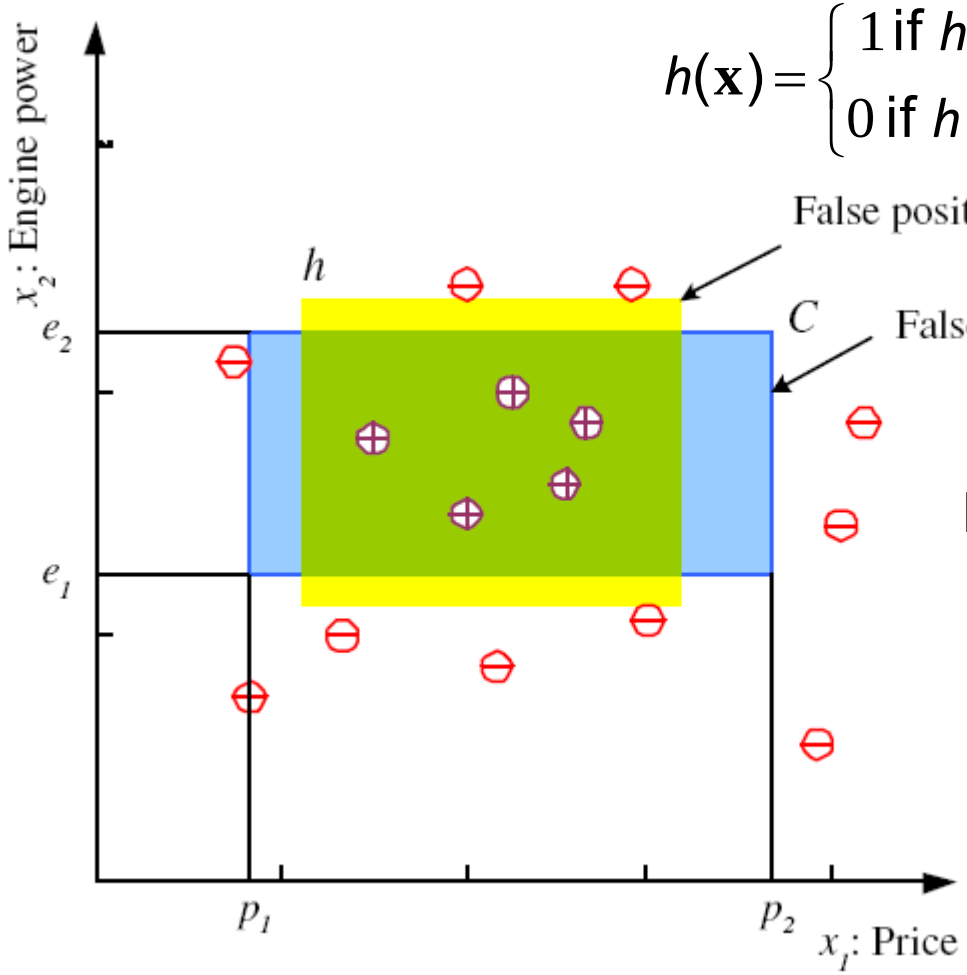
Class C

$$(p_1 \leq \text{price} \leq p_2) \text{ AND } (e_1 \leq \text{engine power} \leq e_2)$$



Hypothesis class $\mathcal{H}$

$$h(\mathbf{x}) = \begin{cases} 1 \text{ if } h \text{ says } \mathbf{x} \text{ is positive} \\ 0 \text{ if } h \text{ says } \mathbf{x} \text{ is negative} \end{cases}$$



Error of h on $\mathcal{H}$

$$E(h \mid \mathcal{X}) = \sum_{t=1}^{N} 1\left(h\left(\mathbf{x}^t\right) \neq r^t\right)$$

Computational Learning Theory:

- Lies at the intersection of AI, Statistics, and Theoretical Computer Science.

Conference on Learning Theory

IMPORTANT DATES

- **Submission deadline**: January 31, 4:00 PM PST
- **Author feedback**: March 28–April 3
- **Author notification**: May 1
- **Conference**: July 9–12

THIRTY-THIRD ANNUAL CONFERENCE ON LEARNING THEORY

The 33rd Annual Conference on Learning Theory (COLT 2020) takes place in Graz, Austria from July 9–12. It is conveniently close in space and time to ICML 2020, which takes place in Vienna immediately after COLT.

The keynote speakers for COLT 2020 are David Blei (Columbia University), Salil Vadhan (Harvard University) and Rebecca Willett (University of Chicago).

International Conference on Machine Learning

ICML | 2020

Thirty-seventh International Conference on Machine Learning

Year (2020)

- Help
- My Registrations
- Profile
- Contact ICML
- Code of Conduct
- Future Meetings
- Diversity & Inclusion

Dates | Submit | Attend | Organization

Messe Wien Exhibition & Congress Center, Vienna AUSTRIA

Sun Jul 12th through Sat the 18th, 2020
(Sun is a full day industry expo)

Registration Information

Registration information will appear here closer to the opening of registration, about 3 months prior to the meeting.

Disabilities

Please contact us with any assistance needs at the conference.

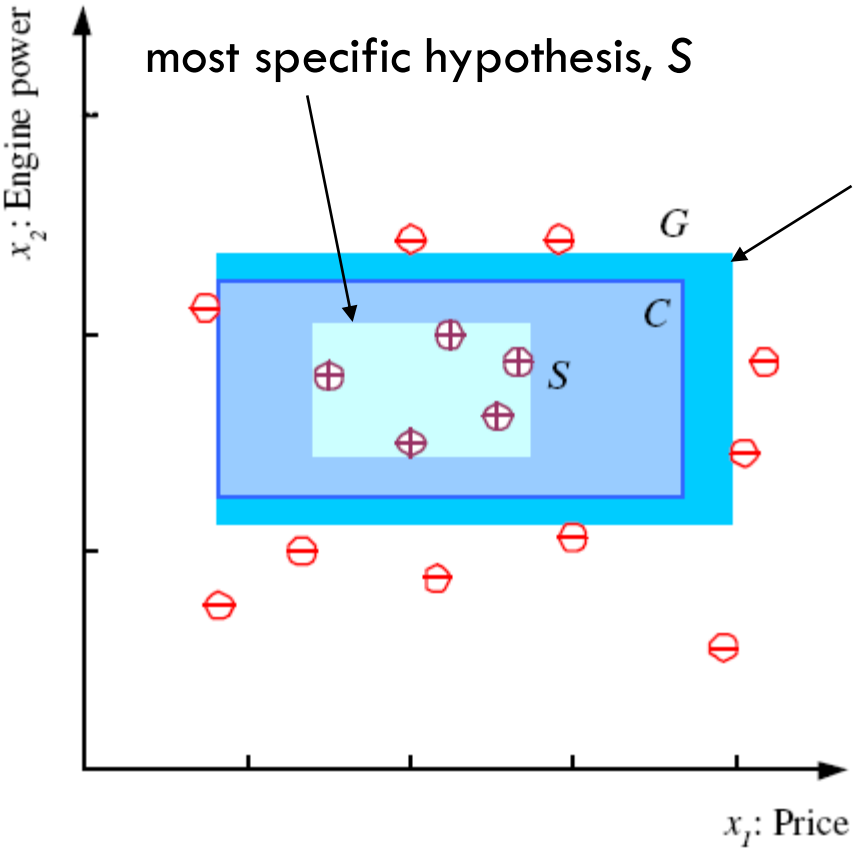
Sponsors

The ICML 2020 Sponsor Portal will open in the first quarter of 2020. New sponsors to ICML, please email us at 2020 ICML Sponsor Prospectus Request, provide your contact information and you will receive the ICML 2020 Sponsor Prospectus when it becomes available. If you were a sponsor at ICML 2019 then you will receive the ICML 2020 Sponsor Prospectus via email when it becomes available as well.

Important Dates

Expo (Industry) Day	Sun July 12th
Tutorials	Mon Jul 13th
Conference Sessions	Tue Jul 14th through Thu the 16th
Workshops	Fri Jul 17th through Sat the 18th

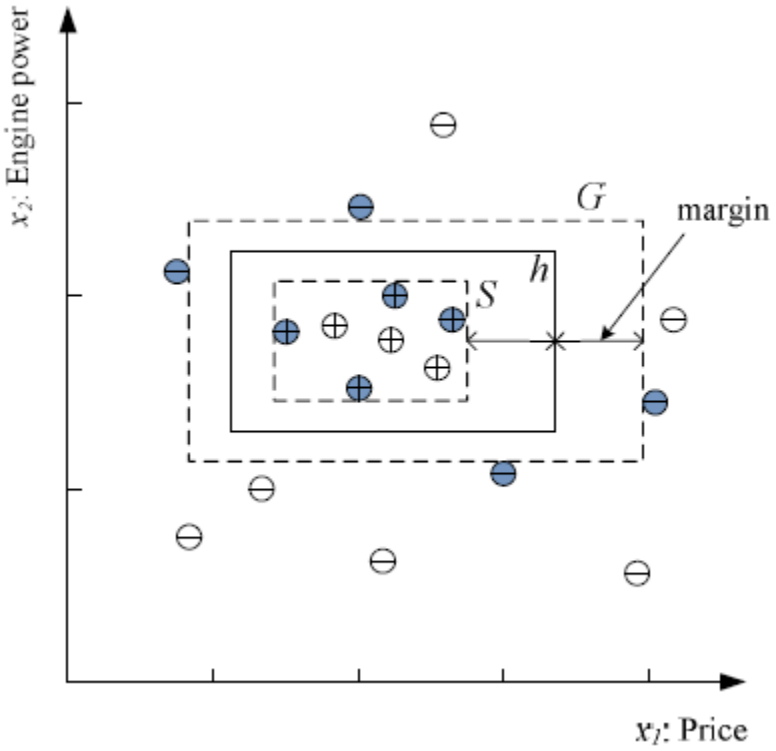
S, G, and the Version Space



$h \in H$, between S and G is consistent and make up the version space
(Mitchell, 1997)

Margin

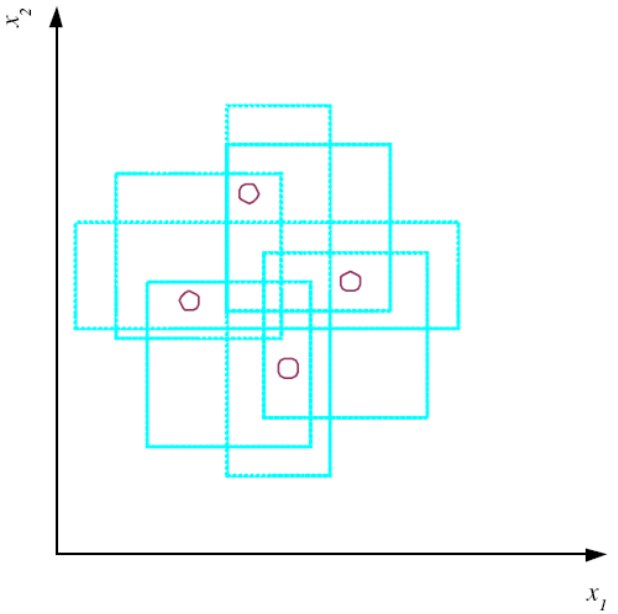
- Choose *h* with largest margin



VC Dimension

- N points can be labeled in 2^N ways as +/–
- $\mathcal{H}$ shatters N if there exists $h \in \mathcal{H}$ consistent for any of these:

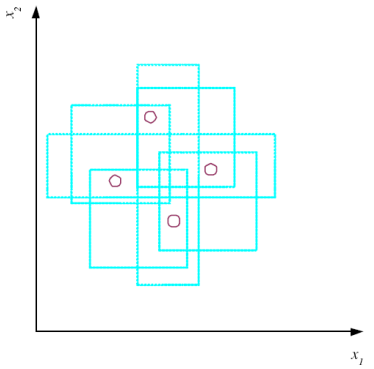
  $VC(\mathcal{H}) = N$



An axis-aligned rectangle shatters 4 points only !

VC Dimension

- N points can be labeled in 2^N ways as +/−
- $\mathcal{H}$ shatters N if there exists $h \in \mathcal{H}$ consistent for any of these: $VC(\mathcal{H}) = N$



- NOTE: It is enough that we find four points that can be shattered
  - it is not necessary that we be able to shatter any four points in two dimensions.
  - For example, four points placed on a line cannot be shattered by rectangles.
  - Five points cannot be placed in two dimensions anywhere such that a rectangle can separate the positive and negative examples for all possible labelings.

VC Dimension

- WOW: using rectangles as our hypothesis class, we can learn only datasets containing four points and not more.
- A learning algorithm that can learn datasets of four points is not very useful.
- This is because the VC dimension is independent of the probability distribution from which instances are drawn.
- World is smoothly changing,
  - instances close by most of the time have the same labels,
  - we need not worry about all possible labelings.
  - There are a lot of datasets containing many more data points than four that are learnable by our hypothesis class (figure 2.1).
- So even hypothesis classes with small VC dimensions are applicable and are preferred over those with large VC dimensions,
  - lookup table that has infinite VC dimension.

VC Dimension

Machine Learning, Tom Mitchell, McGraw Hill, 1997.

Machine Learning is the study of computer algorithms that improve automatically through experience. Applications range from datamining programs that discover general rules in large data sets, to information filtering systems that automatically learn users' interests.

This book provides a single source introduction to the field. It is written for advanced undergraduate and graduate students, and for developers and researchers in the field. No prior background in artificial intelligence or statistics is assumed.

Chapter Outline: (or see the detailed table of contents (postscript))

- 1. Introduction
- 2. Concept Learning and the General-to-Specific Ordering
- 3. Decision Tree Learning
- 4. Artificial Neural Networks
- 5. Evaluating Hypotheses
- 6. Bayesian Learning
- 7. Computational Learning Theory
- 8. Instance-Based Learning
- 9. Genetic Algorithms
- 10. Learning Sets of Rules
- 11. Analytical Learning
- 12. Combining Inductive and Analytical Learning
- 13. Reinforcement Learning

Tom Mitchell

Tom Mitchell

E. Fredkin University Professor
Machine Learning Department
School of Computer Science
Carnegie Mellon University

Tom.Mitchell@cmu.edu, 412 268 2611, GHC 8203 Resume, A personal interview
Assistant: Mary Stech, 412 268-6869

Mathematics Genealogy Project

Home
Search
Extrema
About MGP
Links
FAQs
Posters
Submit Data
Contact
Mirrors
Donate

Tom Michael Mitchell

MathSciNet

Ph.D. Stanford University 1979 

Dissertation: *Version Spaces: An Approach to Concept Learning*

Advisor: Bruce Gardner Buchanan

Students:
Click here to see the students ordered by family name.

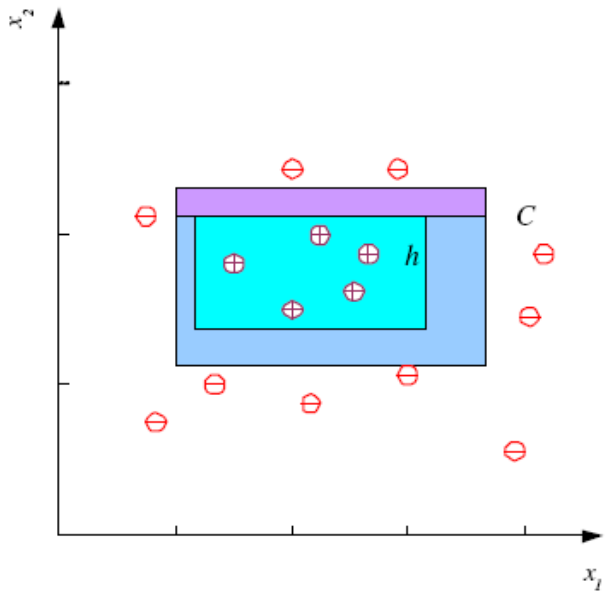
Name	School	Year	Descendants
Biesel, Heiner	Rutgers University, New Brunswick	1984	
Utgoff, Paul	Rutgers University, New Brunswick	1984	10
Keller, Richard	Rutgers University, New Brunswick	1987	
Kedar-Cabelli, Smadar	Rutgers University, New Brunswick	1988	
Mahadevan, Sridhar	Rutgers University, New Brunswick	1990	3
Tadepalli, Prasad	Rutgers University, New Brunswick	1990	5
Etzioni, Oren	Carnegie Mellon University	1991	12
Tan, Ming	Carnegie Mellon University	1991	
Christiansen, Alan	Carnegie Mellon University	1992	
Lin, Long-Ji	Carnegie Mellon University	1992	
Cheng, John	Carnegie Mellon University	1995	
McCallum, Andrew	University of Rochester	1995	2
Thrun, Sebastian	Rheinische Friedrich-Wilhelms-Universität Bonn	1995	19
Chrisman, Lonnie	Carnegie Mellon University	1996	
Freitag, Dayne	Carnegie Mellon University	1998	
Gordon, Geoffrey	Carnegie Mellon University	1999	2
Nigam, Kamal	Carnegie Mellon University	2001	
Roy, Nicholas	Carnegie Mellon University	2003	3
Rustandi, Indrayana	Carnegie Mellon University	2010	

Probably Approximately Correct (PAC) Learning

- How many training examples N should we have, such that with probability at least $1 - \delta$, h has error at most ε ?

  (Blumer et al., 1989)

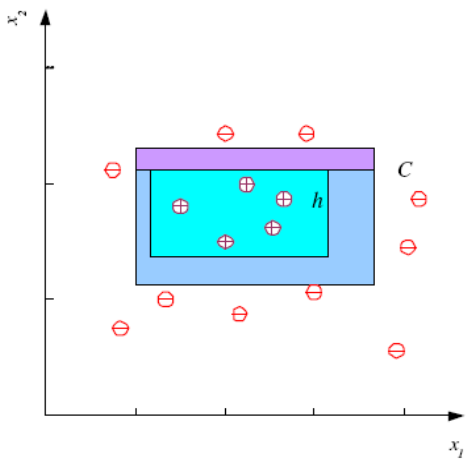
- Each strip is at most $\varepsilon/4$
- Pr that we miss a strip $1 - \varepsilon/4$
- Pr that N instances miss a strip $(1 - \varepsilon/4)^N$
- Pr that N instances miss 4 strips $4(1 - \varepsilon/4)^N$
- $4(1 - \varepsilon/4)^N \leq \delta$ and $(1 - x) \leq \exp(-x)$
- $4\exp(-\varepsilon N/4) \leq \delta$ and $N \geq (4/\varepsilon)\log(4/\delta)$



Russell & Norvig
Probably Approximately Correct (PAC)

- Underlying Principle :
  - any hypothesis seriously wrong will almost certainly be "found out" with high probability after a small number of examples.
  - Because, it will make an incorrect prediction.
- Thus,
  - any hypothesis that is consistent with a sufficiently large set of training examples is unlikely to be seriously wrong:
  - It must be Probably Approximately Correct.
- PAC Learning Algorithm returns hypotheses that are probably approximately correct.
- PAC allows performance bounds of various learning algorithms.

Probably Approximately Correct (PAC) Learning



- How many training examples N should we have, such that with probability at least $1 - \delta$, h has error at most ε ?
- First – Our Learning Algorithm:
  - Hypothesis is the smallest rectangle covering all positives and no negatives

PAC-Learning Theorems & Stationarity Assumption

- Assumes Stationary Distribution
  - Probability distribution over examples remains stationary over time.
- Each example data point:
  - is a random variable E_j whose observed value $e_j = (x_j, y_j)$ is sampled from that distribution,
  - and is independent of the previous examples:
    - $P(E_j | E_{j-1}, E_{j-2}, \ldots) = P(E_j)$,
- Each example has an identical prior probability distribution:
  - $P(E_j) = P(E_{j-1}) = P(E_{j-2}) = \cdots$.
- Examples that satisfy these assumptions are called:
  - independent and identically distributed or i.i.d..

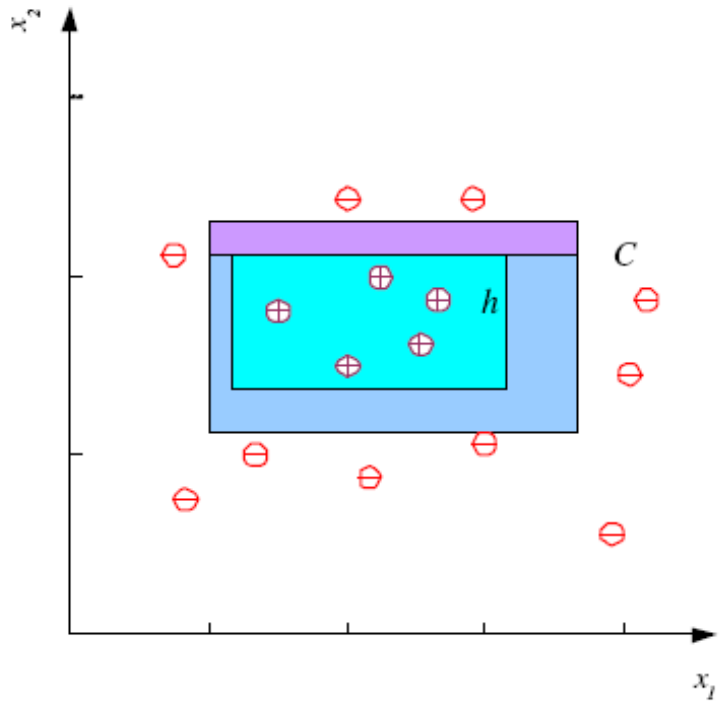
PAC-Learning Theorems & Stationarity Assumption

- An i.i.d. assumption connects the past to the future:
  - without some such connection, all bets are off—the future could be anything.
- Note: Learning can still occur if there are slow changes in the distribution
  - (i.e., discounting the past)

PAC Learning Theorems

- Stationarity Assumption
  - Sample distribution unchanging
- Also assume true function f is:
  - deterministic
  - member of the hypothesis class H that is being considered
- Simplest Theorems assume f is a boolean function.
  - Rectangle for Illustration!

Probably Approximately Correct (PAC) Learning



- Strip is 0.1 of concept.
- 0.9 chance of missing strip.

Probably Approximately Correct

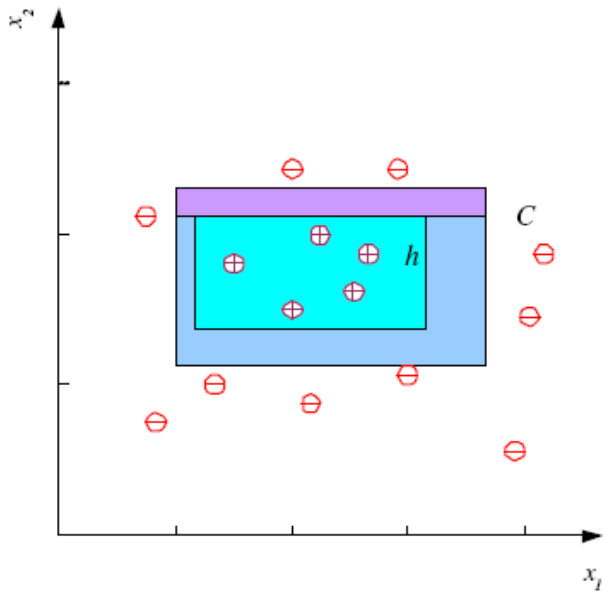
- What is the probability of an incorrect hypothesis (h_b) getting lucky on N examples:
  - Error (h_b) > ε
  - P(h_b agrees with N examples) = $(1 - \varepsilon)^N$
- Given bad hypothesis set H_b, probability that any are consistent on N examples is:
  - P(H_b contains a hypothesis h_b that agrees with N examples)
  - = $|H_b| * (1 - \varepsilon)^N \leq |H| * (1 - \varepsilon)^N$
- Bound using size of hypothesis space |H|

Probably Approximately Correct (PAC) Learning

- How many training examples N should we have, such that with probability at least $1 - \delta$, h has error at most ε ?

  (Blumer et al., 1989)

- Each strip is at most $\varepsilon/4$
- Pr that we miss a strip $1 - \varepsilon/4$
- Pr that N instances miss a strip $(1 - \varepsilon/4)^N$
- Pr that N instances miss 4 strips $4(1 - \varepsilon/4)^N$
- $4(1 - \varepsilon/4)^N \leq \delta$ and $(1 - x) \leq \exp(-x)$
- $4\exp(-\varepsilon N/4) \leq \delta$ and $N \geq (4/\varepsilon)\log(4/\delta)$



Approximately Correct

$$GenLoss_L(h) = \sum_{(x,y)\in\mathcal{E}} L(y, h(x))\, P(x, y)\ ,$$

$$\text{error}(h) = GenLoss_{L_{0/1}}(h) = \sum_{x,y} L_{0/1}(y, h(x))\, P(x, y)\ .$$

Absolute value loss: $L_1(y, \hat{y}) = |y - \hat{y}|$
Squared error loss: $L_2(y, \hat{y}) = (y - \hat{y})^2$
0/1 loss: $L_{0/1}(y, \hat{y}) = 0$ if $y = \hat{y}$, else 1

- Hypothesis *h* is approximately correct if:
  - Error(h) < ε
  - ε is small constant
- We can calculate N such that if a hypothesis is consistent over N examples it is approximately correct.

Probably Approximately Correct

- What is the probability of an incorrect hypothesis (h_b) getting lucky and being correct on N examples:
  - Error (h_b) > ε
  - P(h_b agrees with N examples) = $(1 - \varepsilon)^N$
- Given the set of all bad hypothesis H_b, the probability that any of them are consistent on N examples is
  - P(H_b contains a hypothesis h_b that agrees with N examples)
  - = $|H_b| * (1 - \varepsilon)^N \leq |H| * (1 - \varepsilon)^N$

Probably Approximately Correct

- We want the probability for the existence of a bad hypothesis to be less than some small number δ
  - $|H|*(1-\varepsilon)^N \leq \delta$

Given that $1-\epsilon \leq e^{-\epsilon}$, we can achieve this if we allow the algorithm to see

$$N \geq \frac{1}{\epsilon}\left(\ln\frac{1}{\delta}+\ln|\mathcal{H}|\right) \quad (18.1)$$

- So if a learning algorithm returns an hypothesis that is consistent with this many examples:
  - With probability at least 1− δ (probably), it has error at most ε (approximately correct)
  - It is Probably Approximately Correct!

Probably Approximately Correct

- If our hypothesis space is the set of all boolean functions
  - $|H| = 2^{2^n}$
  - The number of possible example is 2^n
  - Learning a boolean function would require seeing nearly all of the examples.
- Our Hypothesis Space is Too Complex…

2^{2^n}: Our Hypothesis Space is Too Large!?! What can we do?!?

1. Maybe we need prior knowledge about the problem?
   - AI: Knowledge Representation
2. Weight algorithm to prefer SIMPLE hypothesis.
   - Tree Pruning
   - Later More on Regularization
3. Restrict Hypothesis SpaceFacilitate Analysis
   - Facilitate Analysis
   - Book's Approach w/ Theoretical Analysis

Restricted Hypothesis Space
Decision Lists

- Consider a new Hypothesis Space
- Hypothesis H will be a list of tests, where each test is a conjunction of literals
  - If the length of the tests are unbounded, we're back where we started.
  - If the length of the tests are bounded to **k** literals… Learnable on reasonable number of examples!

Decision Lists

- Consider a new Hypothesis Space using Conj(n, k)=
  - Conjunction of K Literals from a set of N attributes
- Decision list will consist of a list of items from Conj(n,k).
- Given each test is Yes or No or absent from the decision list, there are at most $3^{|Conj(n,k)|}$ distinct sets of component
- $|k\text{-}DL(n)| \leq 3^{|Conj(n,k)|} * |Conj(n,k)|!$
  - $|Conj(n,k)|!$ factor included since the elements can be in any order.

Decision Lists

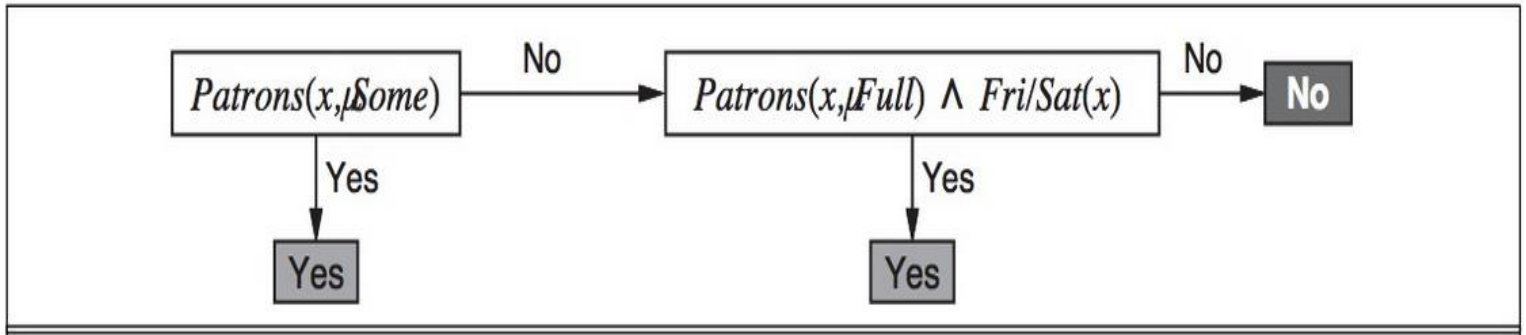


Figure 18.10 A decision list for the restaurant problem.

$$WillWait \Leftrightarrow (Patrons = Some) \vee (Patrons = Full \wedge Fri/Sat) .$$

function DECISION-LIST-LEARNING(*examples*) **returns** a decision list, or *failure*

if *examples* is empty **then return** the trivial decision list *No*
$t \leftarrow$ a test that matches a nonempty subset $examples_t$ of *examples*
such that the members of $examples_t$ are all positive or all negative
if there is no such t **then return** *failure*
if the examples in $examples_t$ are positive **then** $o \leftarrow Yes$ **else** $o \leftarrow No$
return a decision list with initial test t and outcome o and remaining tests given by
DECISION-LIST-LEARNING($examples - examples_t$)

Figure 18.11 An algorithm for learning decision lists.

- Now let's analyze this algorithm.
- How many examples N should we see?

Restricted Hypothesis Space Decision Lists

Given that $1 - \epsilon \leq e^{-\epsilon}$, we can achieve this if we allow the algorithm to see

$$N \geq \frac{1}{\epsilon}\left(\ln\frac{1}{\delta} + \ln|\mathcal{H}|\right) \tag{18.1}$$

- Hypothesis *h* is approximately correct if:
  - Error(h) < ε
  - ε is small constant
- What is our new size of $\mathcal{H}$?

$\mathcal{H}$ = k-DL(n)
Conjunction of at most k literals using n attributes

- $|\text{k-DL(n)}| \leq 3^{|\text{Conj(n,k)}|} * |\text{Conj(n,k)}|!$
  - $|\text{Conj(n,k)}| = O(n^k)$

- $|\text{k-DL(n)}| = 2^{O(n^k \log_2(n^k))}$

k-DL(n)

- $|\text{k-DL(n)}| \leq 3^{|\text{Conj(n,k)}|} * |\text{Conj(n,k)}|!$
  - $|\text{Conj(n,k)}| = O(n^k)$

- $|\text{k-DL(n)}| = 2^{O(n^k \log_2(n^k))}$

Now we are ready to calculate N.

Given that $1 - \epsilon \leq e^{-\epsilon}$, we can achieve this if we allow the algorithm to see

$$N \geq \frac{1}{\epsilon}\left(\ln\frac{1}{\delta} + \ln|\mathcal{H}|\right) \qquad (18.1)$$

k-DL(n)

- $|k\text{-DL}(n)| \leq 3^{|Conj(n,k)|} * |Conj(n,k)|!$
  - $|Conj(n,k)| = O(n^k)$
- $|k\text{-DL}(n)| = 2^{O(n^k \log_2(n^k))}$

$$N \geq \frac{1}{\varepsilon}\left(\ln\frac{1}{\delta} O(nk \log_2(nk))\right) = O(n^k)$$

- Polynomial in the number of attributes!

Joys of Theoretical Computer Science: Corroborating Empirical Data

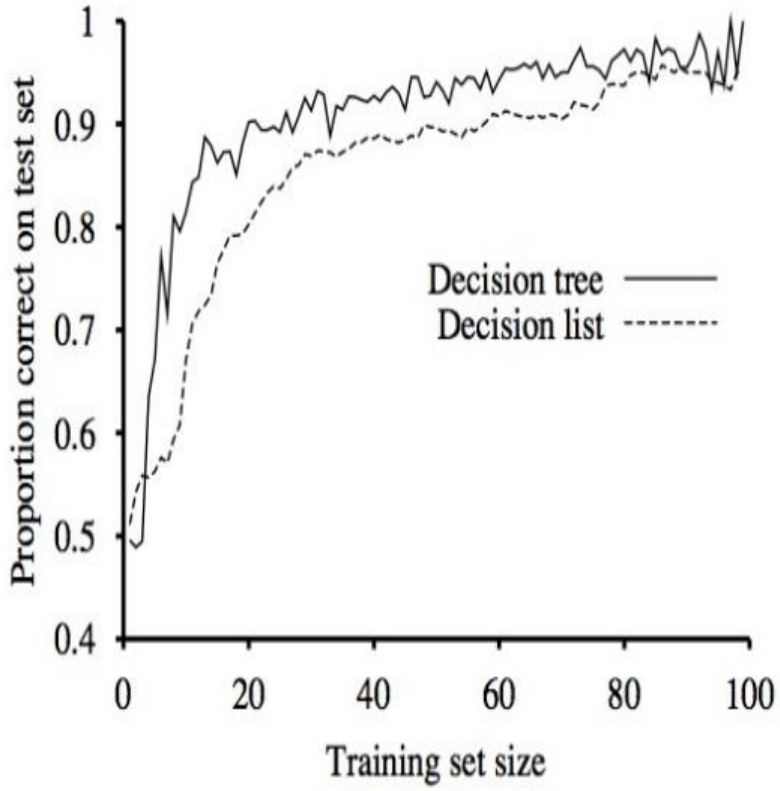
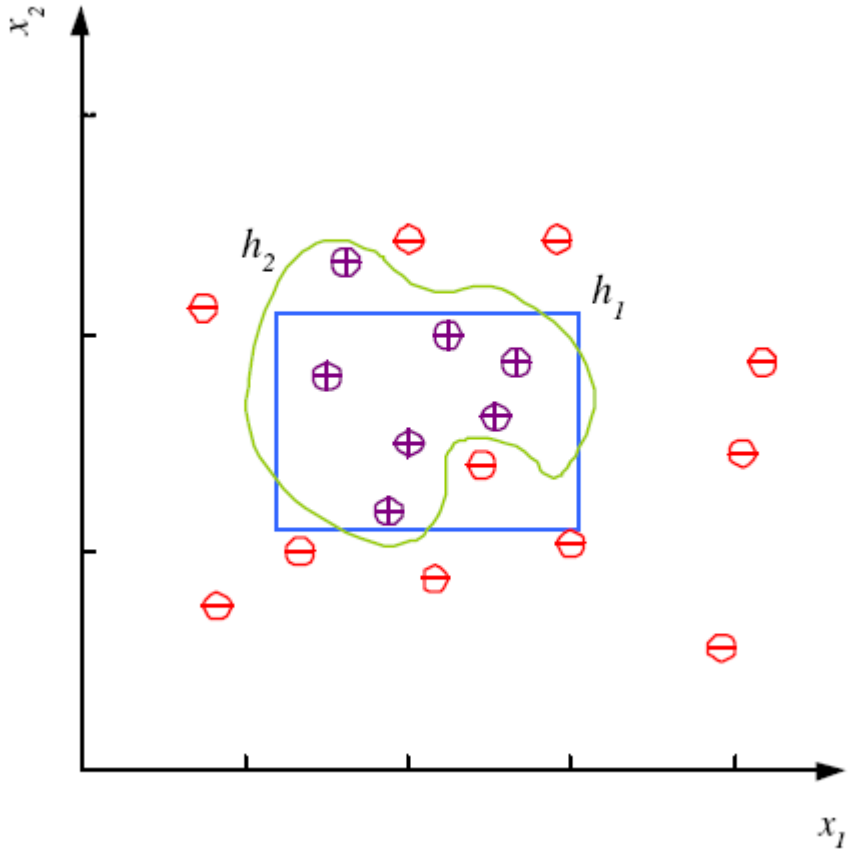


Figure 18.12 Learning curve for DECISION-LIST-LEARNING algorithm on the restaurant data. The curve for DECISION-TREE-LEARNING is shown for comparison.

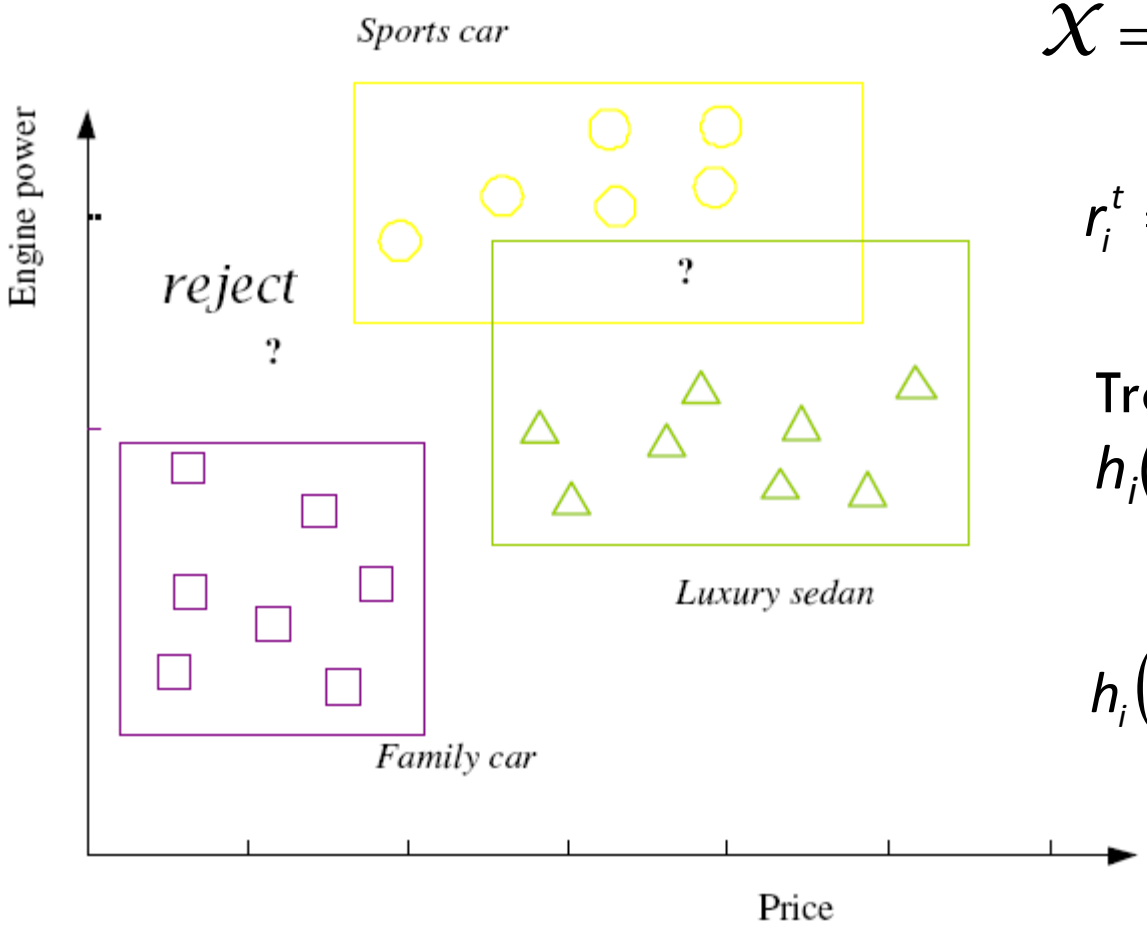
Noise and Model Complexity

Use the simpler one because

- Simpler to use (lower computational complexity)
- Easier to train (lower space complexity)
- Easier to explain (more interpretable)
- Generalizes better (lower variance - Occam's razor)



Multiple Classes, C_i i=1,...,K



$$\mathcal{X} = \{\mathbf{x}^t, r^t\}_{t=1}^N$$

$$r_i^t = \begin{cases} 1 \text{ if } \mathbf{x}^t \in C_i \\ 0 \text{ if } \mathbf{x}^t \in C_j, j \neq i \end{cases}$$

Train hypotheses $h_i(\boldsymbol{x})$, $i = 1,...,K$:

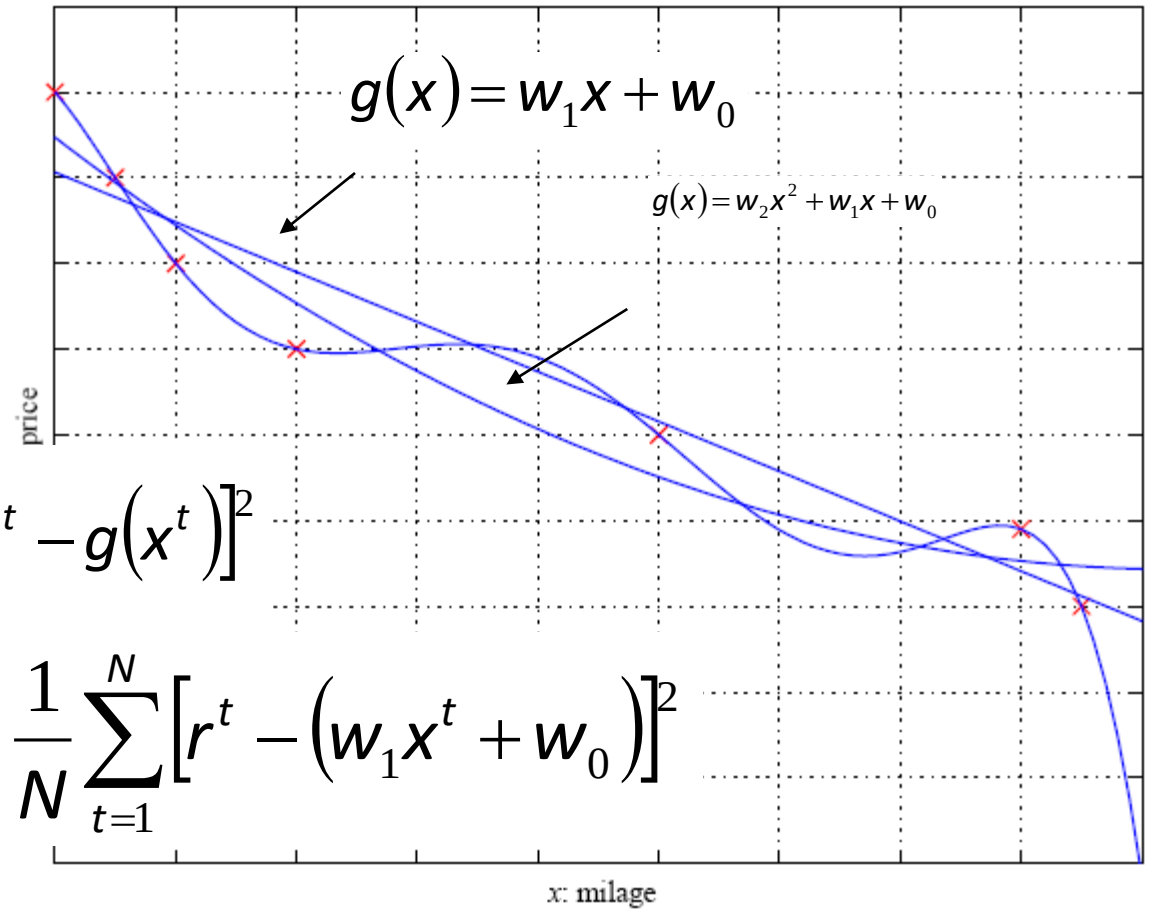
$$h_i\left(\mathbf{x}^t\right) = \begin{cases} 1 \text{ if } \mathbf{x}^t \in C_i \\ 0 \text{ if } \mathbf{x}^t \in C_j, j \neq i \end{cases}$$

Regression

$$\mathcal{X} = \left\{x^t, r^t\right\}_{t=1}^N$$

$$r^t \in \Re$$

$$r^t = f\left(x^t\right) + \varepsilon$$



$$E(g \mid \mathcal{X}) = \frac{1}{N}\sum_{t=1}^{N}\left[r^t - g\left(x^t\right)\right]^2$$

$$E(w_1, w_0 \mid \mathcal{X}) = \frac{1}{N}\sum_{t=1}^{N}\left[r^t - \left(w_1 x^t + w_0\right)\right]^2$$

Model Selection & Generalization

- Learning is an ill-posed problem; data is not sufficient to find a unique solution
- The need for inductive bias, assumptions about $\mathcal{H}$
- Generalization: How well a model performs on new data
- Overfitting: $\mathcal{H}$ more complex than C or *f*
- Underfitting: $\mathcal{H}$ less complex than C or *f*

R&N 18: Consistency vs. simplicity

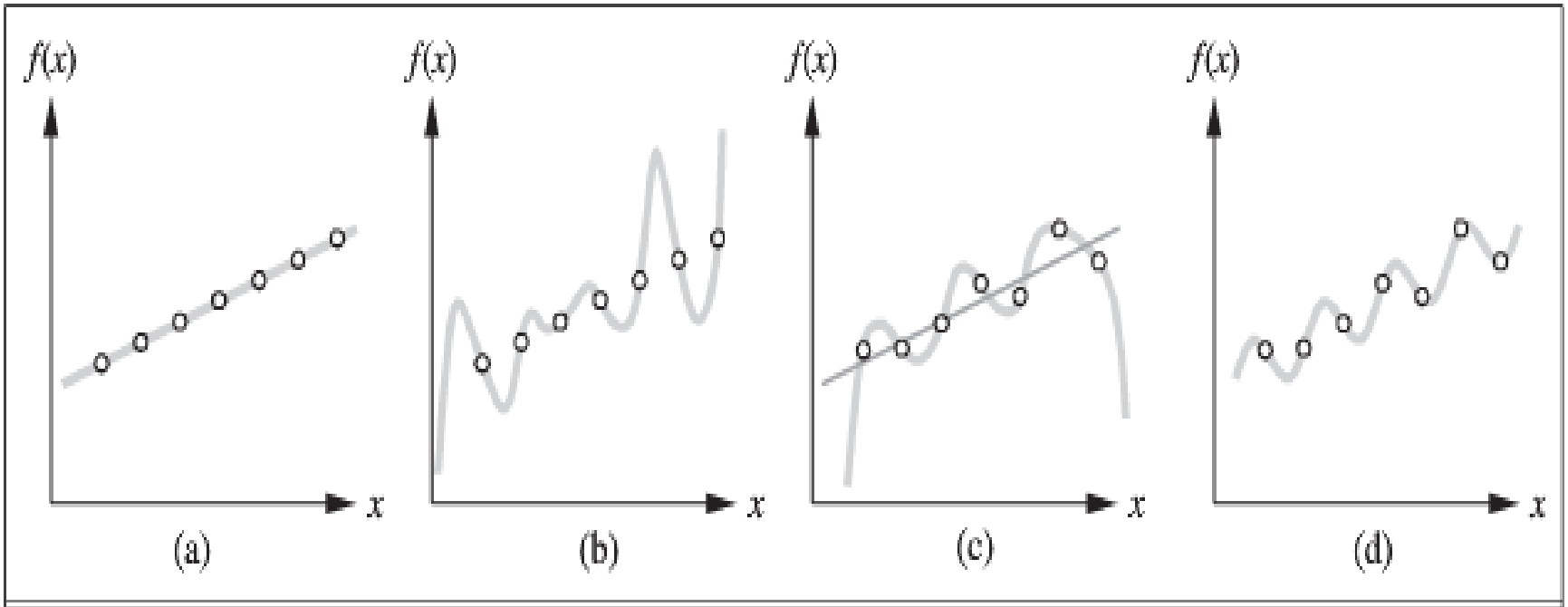
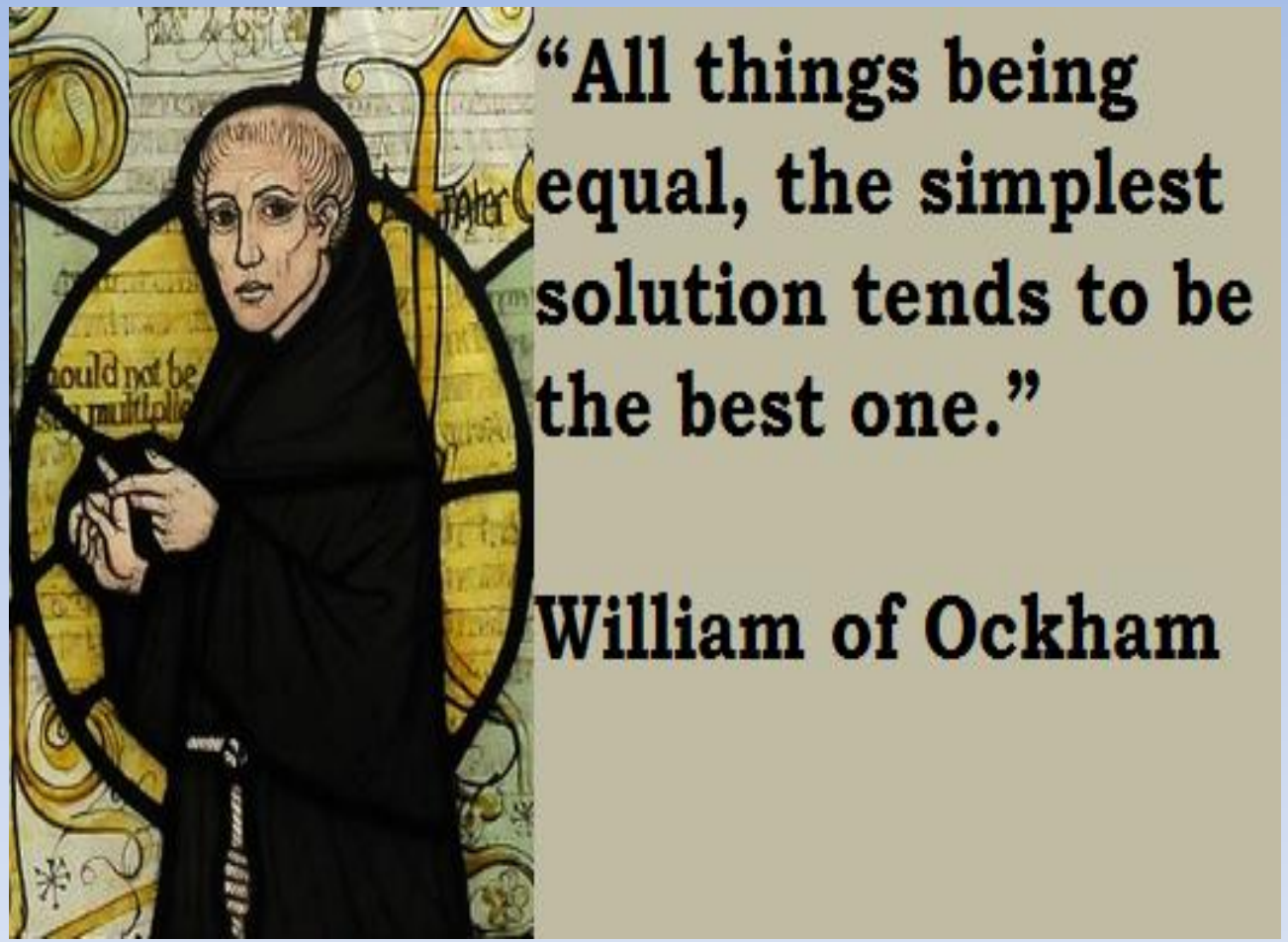


Figure 18.1 (a) Example $(x, f(x))$ pairs and a consistent, linear hypothesis. (b) A consistent, degree-7 polynomial hypothesis for the same data set. (c) A different data set, which admits an exact degree-6 polynomial fit or an approximate linear fit. (d) A simple, exact sinusoidal fit to the same data set.

Ockham's razor



- 14th-century English philosopher William of Ockham

Consistency vs. simplicity

- Usually algorithms prefer consistency by default
- Several ways to operationalize simplicity:
  - Reduce the Hypothesis Space
  - Regularization

Triple Trade-Off

- There is a trade-off between three factors (Dietterich, 2003):
  1. Complexity of $\mathcal{H}$, c ($\mathcal{H}$),
  2. Training set size, N,
  3. Generalization error, E, on new data
- As $N\uparrow$, $E\downarrow$
- As c ($\mathcal{H}$)$\uparrow$, first $E\downarrow$ and then $E\uparrow$
  - Overfitting!

Cross-Validation

- To estimate generalization error, we need data unseen during training. We split the data as
  - Training set (50%)
  - Validation set (25%)
  - Test (publication) set (25%)
- Resampling when there is few data

Dimensions of a Supervised Learner

1. Model: $g(\mathbf{x}|\theta)$

2. Loss function: $E(\theta|\mathcal{X}) = \sum_t L(r^t, g(\mathbf{x}^t|\theta))$

3. Optimization procedure:

$$\theta^* = \arg\min_\theta E(\theta|\mathcal{X})$$