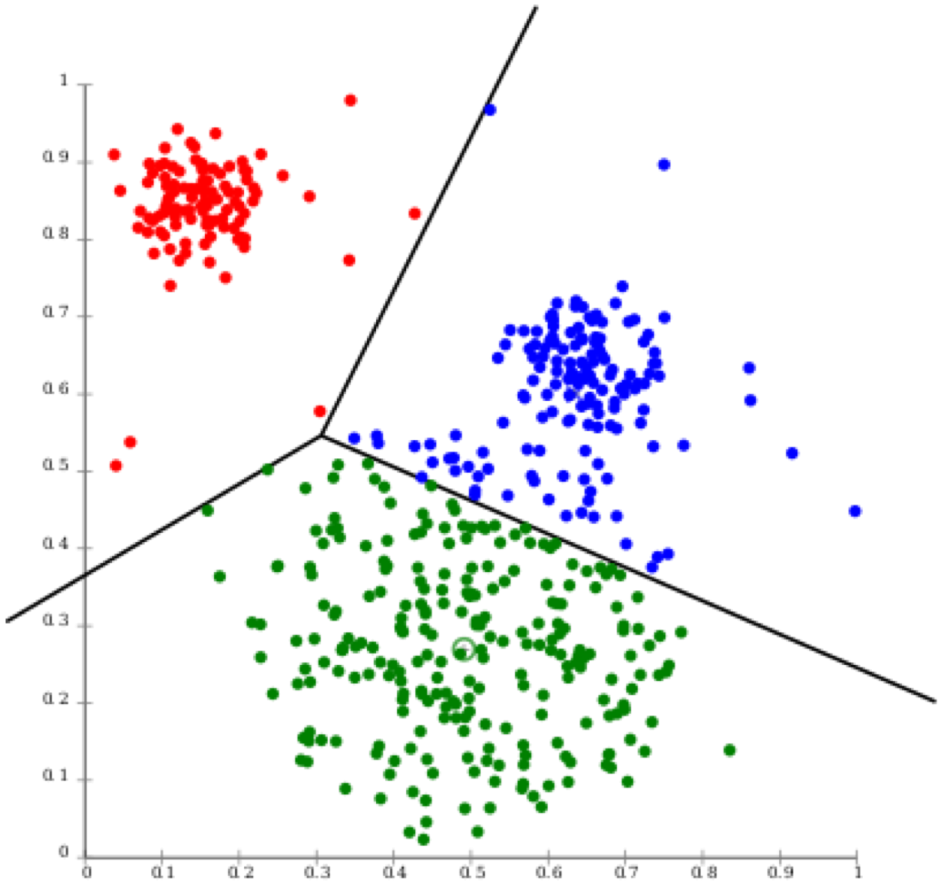


SUPERVISED MACHINE LEARNING

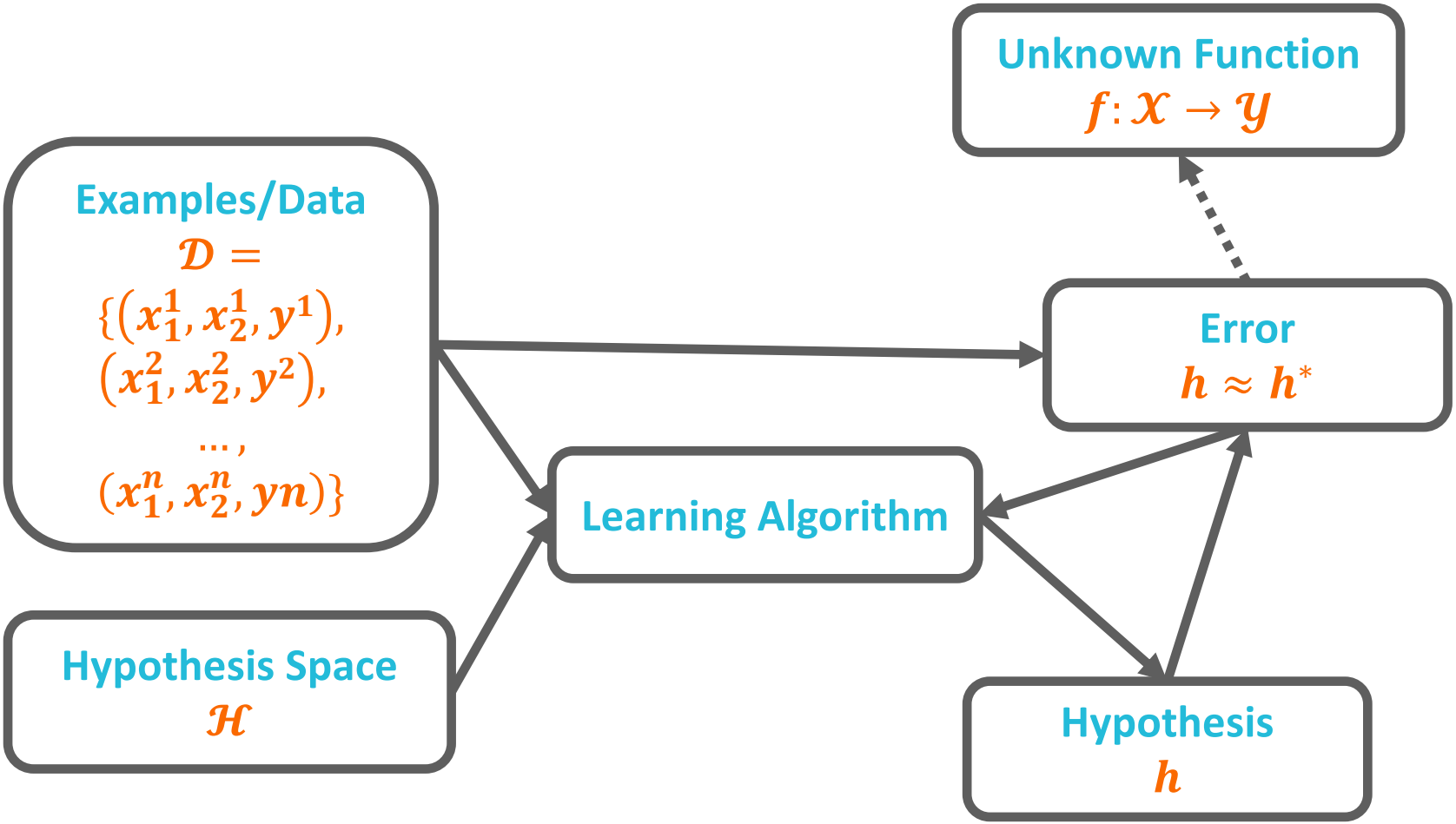
ECS170 Spring 2018

Josh McCoy, @deftjams

Machine Learning Formalization

- **Input**: $x \in \mathcal{X}$
  - Ex. input with two features: $x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$
- **Output**: $y \in \mathcal{Y}$
- **Approximation** of the unknown function
  - $f: \mathcal{X} \rightarrow \mathcal{Y}$
- Labeled **Data**
  - $\mathcal{D} = \{(x_1^1, x_2^1, y^1), (x_1^2, x_2^2, y^2), \ldots, (x_1^n, x_2^n, yn)\}$

Process


Unknown Function
$f: X \rightarrow Y$
Examples/Data
$\mathcal{D} = \{(x_1^1, x_2^1, y^1), (x_1^2, x_2^2, y^2), \ldots, (x_1^n, x_2^n, yn)\}$
Error
$h \approx h^*$
Learning Algorithm
Hypothesis Space
$\mathcal{H}$
Hypothesis
h

Supervised Learning

- **Classification**
  - Output is a label.
- **Regression**
  - Output is a number.
- **Labeled** Data Sets

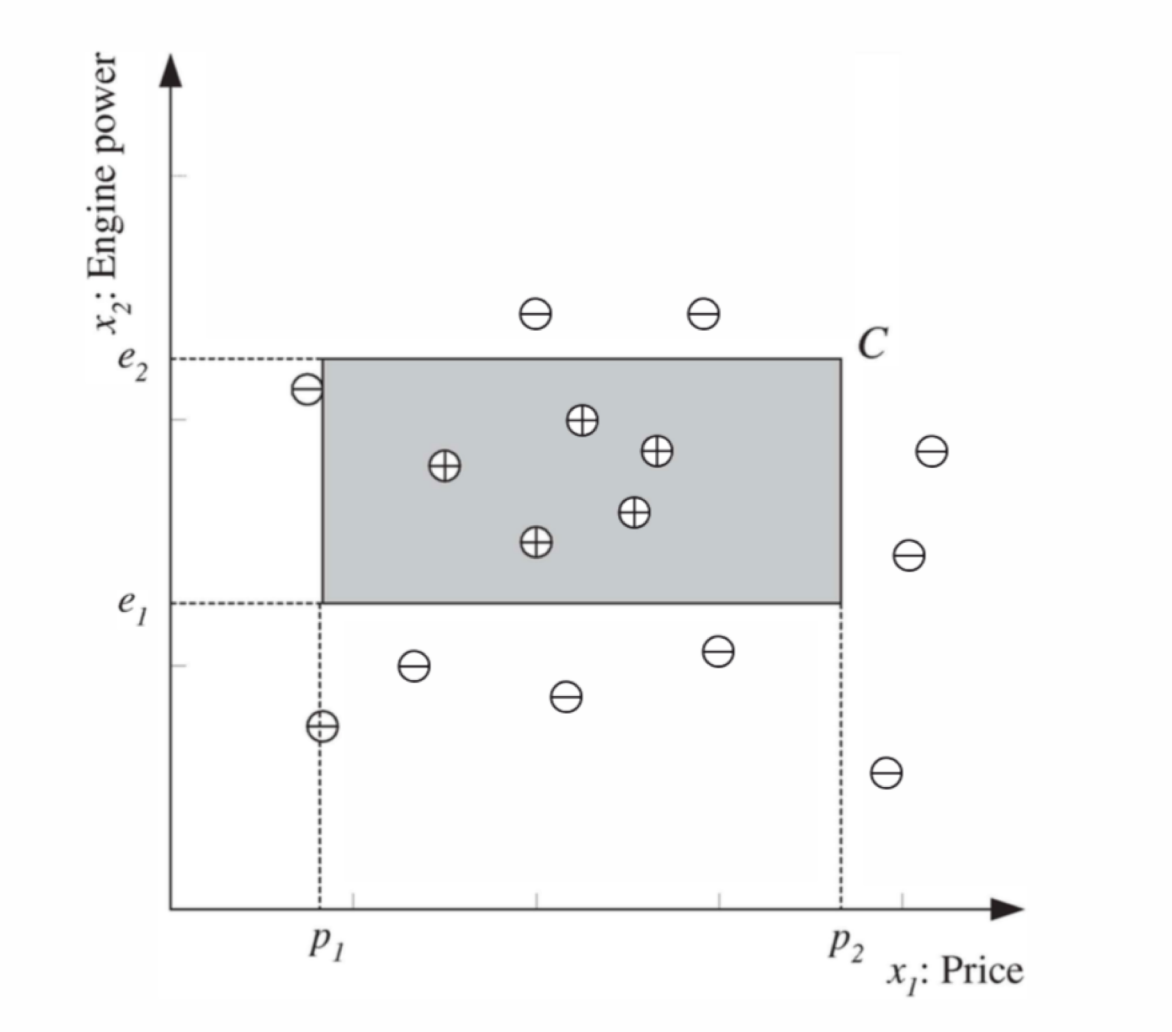
- Goal: find $x \rightarrow y$ with as little error as possible given the labels in the data set.

Supervised

- Classification
  - $f(x) \rightarrow$ labels
  - Boolean or binary.
- Regression
  - $f(x) \rightarrow \mathbb{R}$

- Output either
  - Estimate/Label of an input
  - Prediction of future values

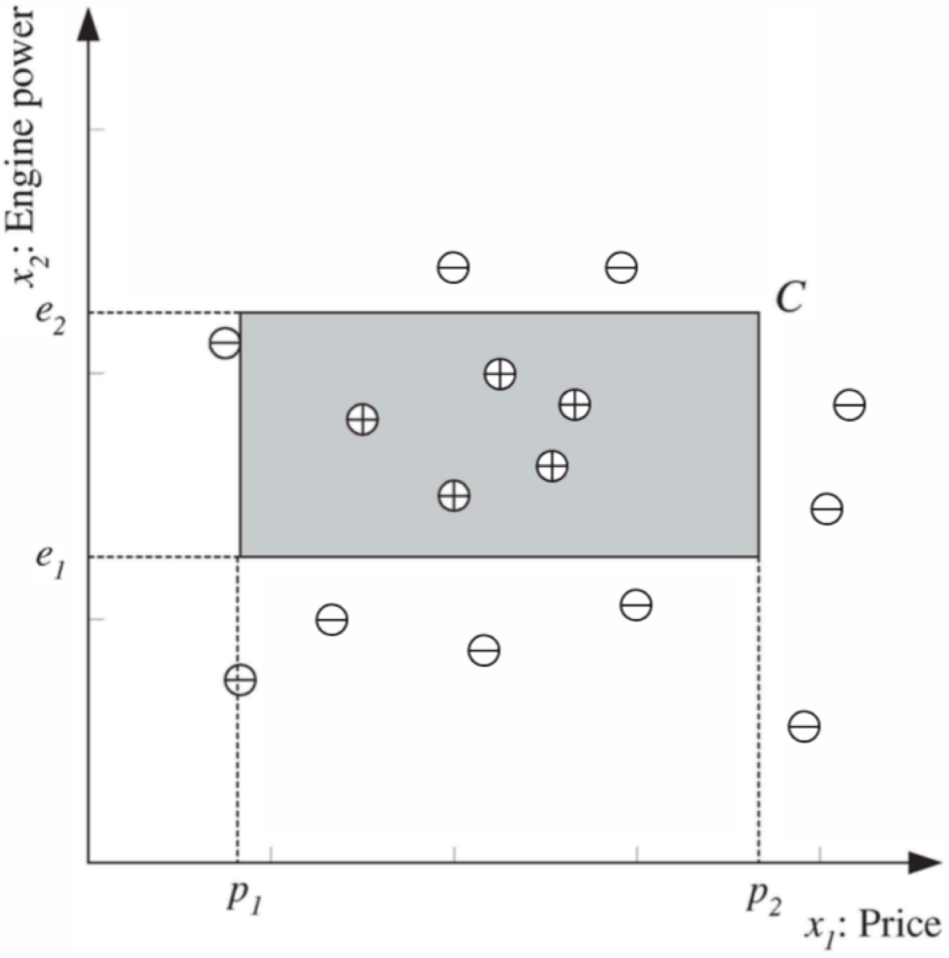
Classify a Family Car



Formalize

- $x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$ — x_1: Cost; x_2: Engine Size
- $y = \begin{cases} 1 \; if \; x \; is \; a \; positive \; example \\ 0 \; if \; x \; is \; a \; negative \; example \end{cases}$
- $\mathcal{D} = \{x^t, y^t\}_{t=1}^{N}$
- $\mathcal{H} = (p_1 \leq price \leq p_2) \; and \; (e_1 \leq engine \; power \leq e_2)$
- $h \in \mathcal{H}$ are bindings of (p_1, p_2, e_1, e_2) that match the data as closely as possible.

Classify a Family Car

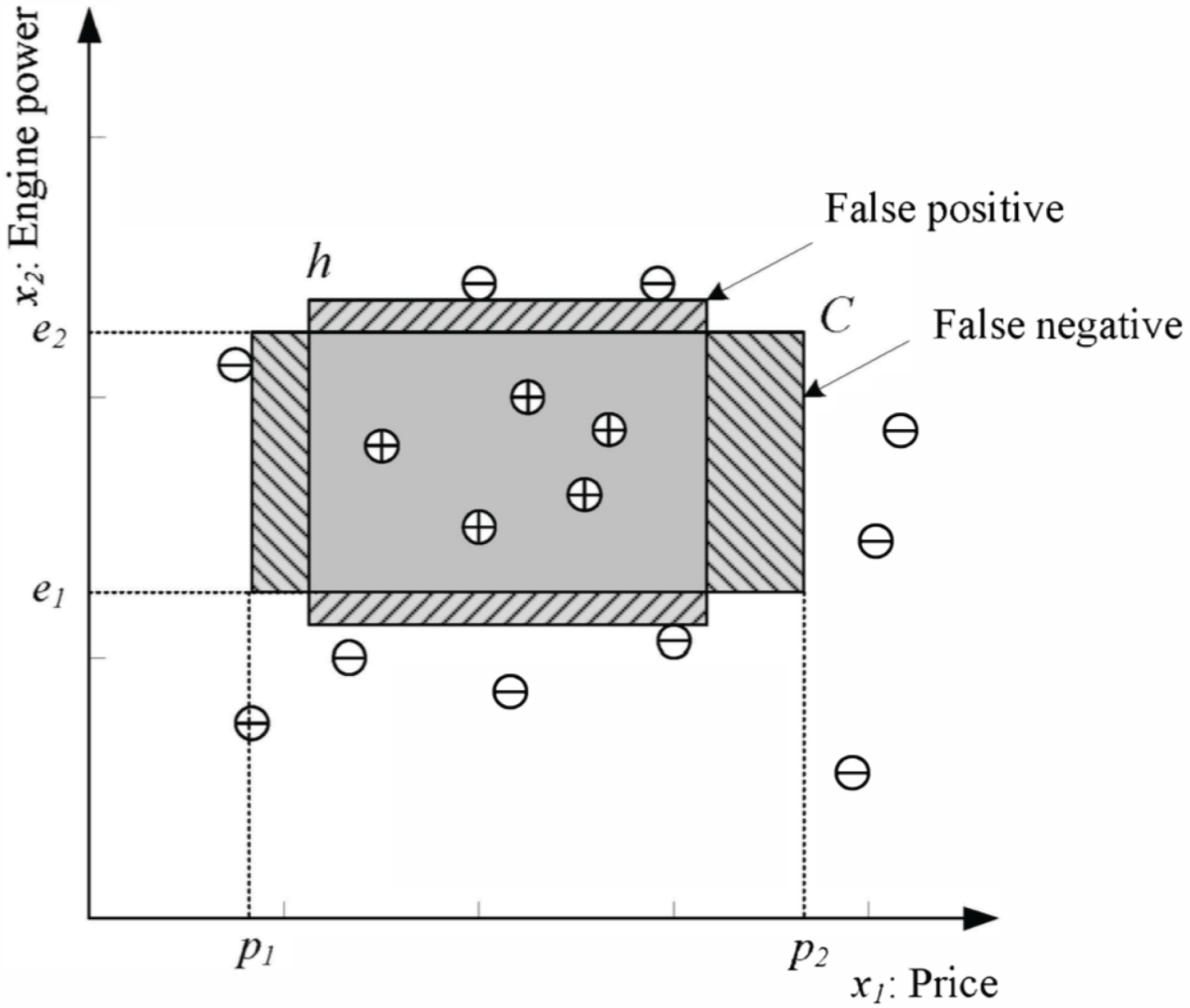


The unknown function is C.

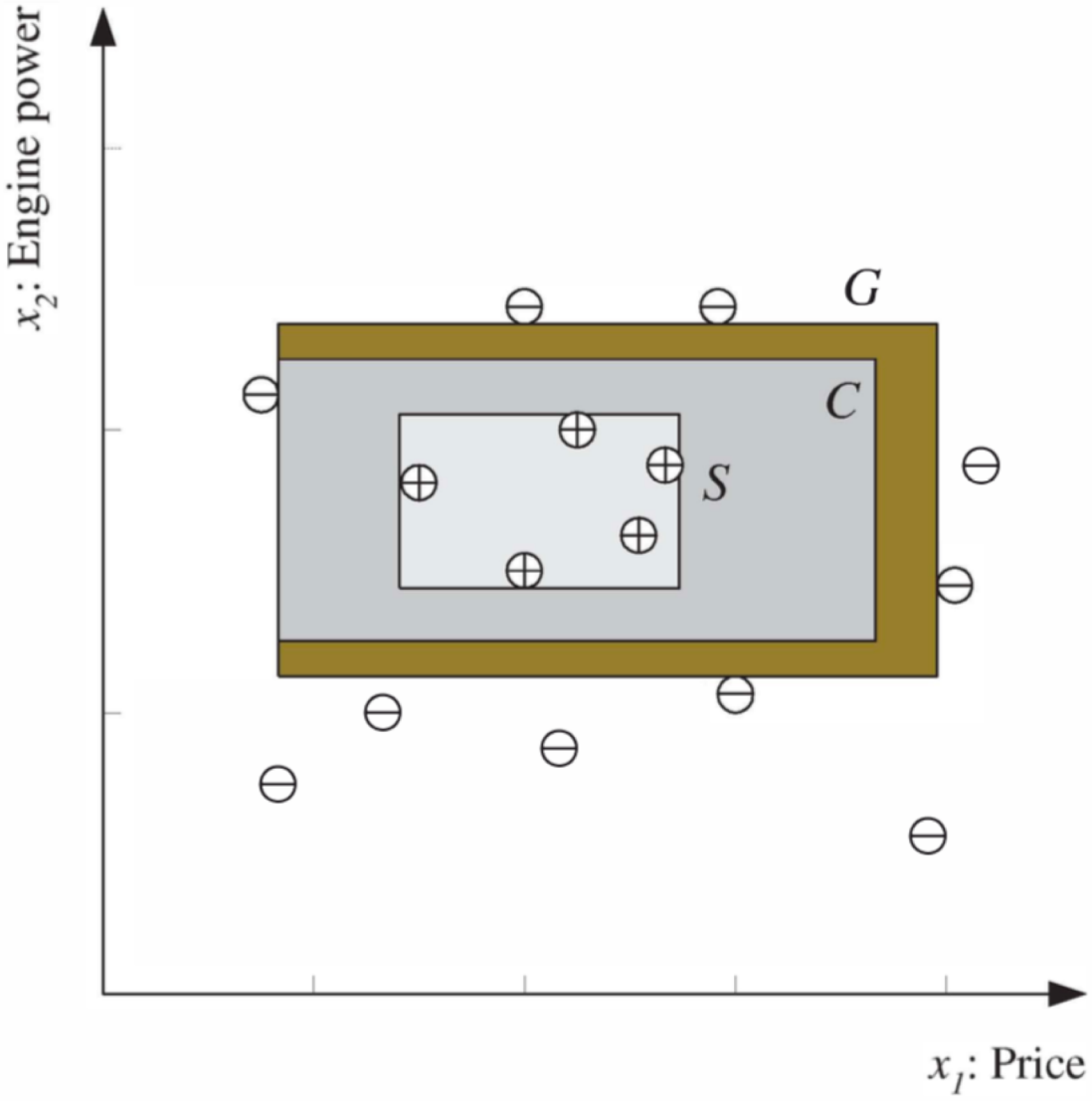
What algorithm should we use to approximate C given the data?

What are the drawbacks and benefits?

Properties of Hypothesis

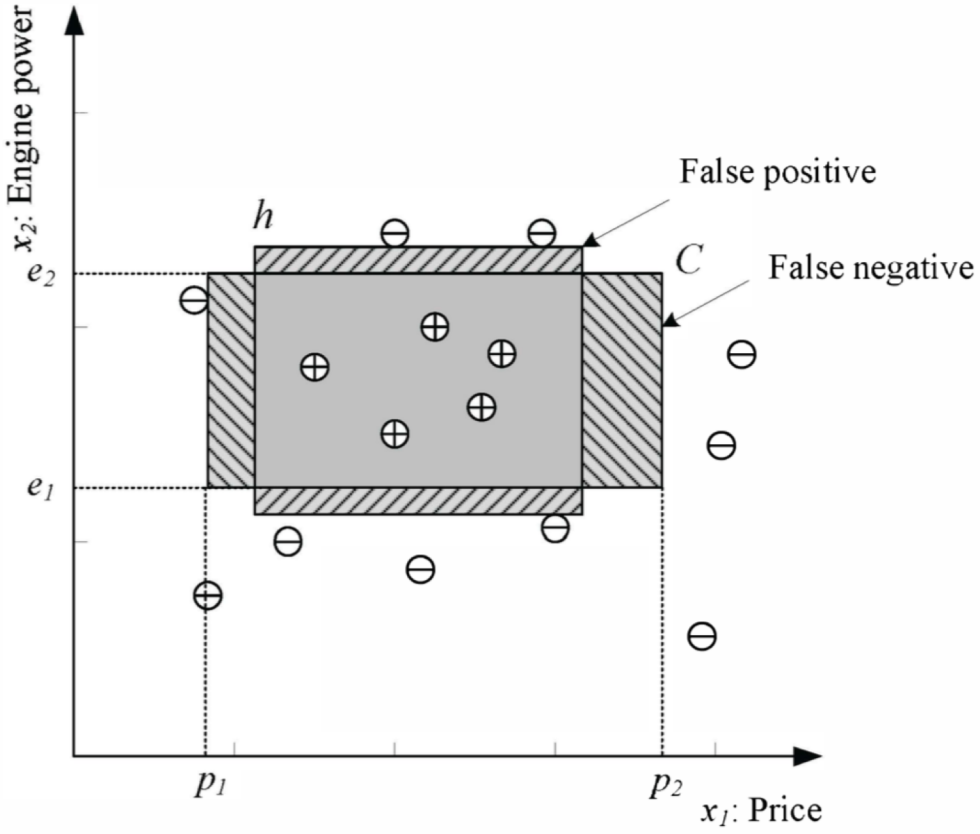


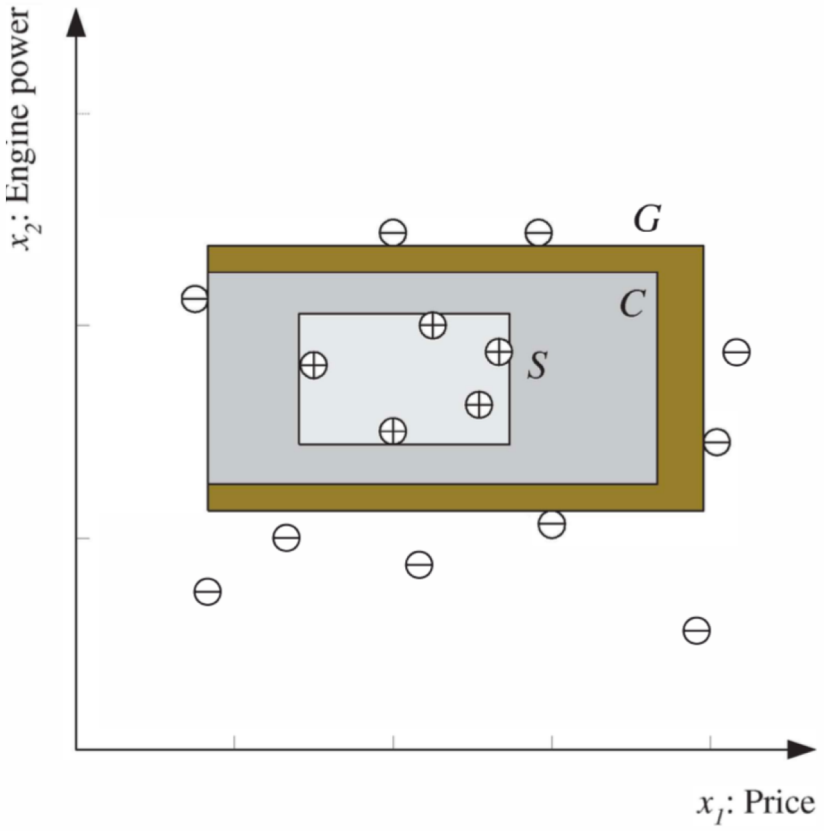
Types of Hypotheses



- C: actual class
- G: most general hypothesis
- S: most specific hypothesis

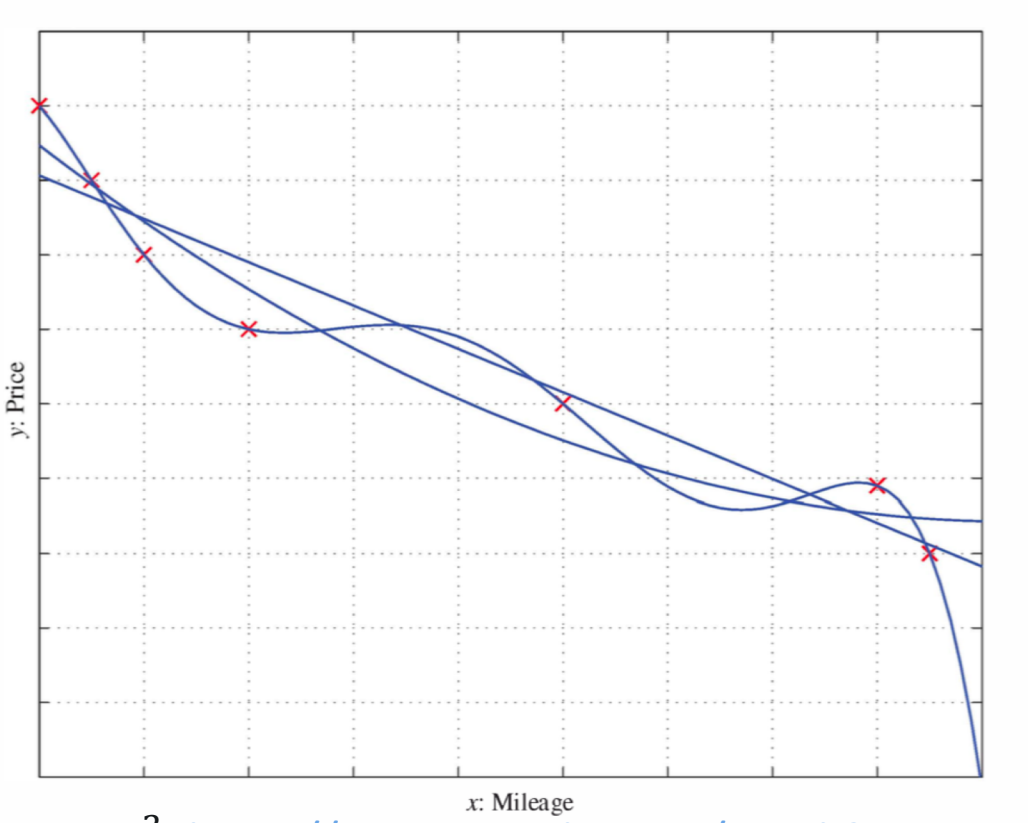
Comparison


x_2: Engine power
e_2
e_1
h
C
False positive
False negative
p_1
p_2
x_1: Price


x_2: Engine power
G
C
S
x_1: Price

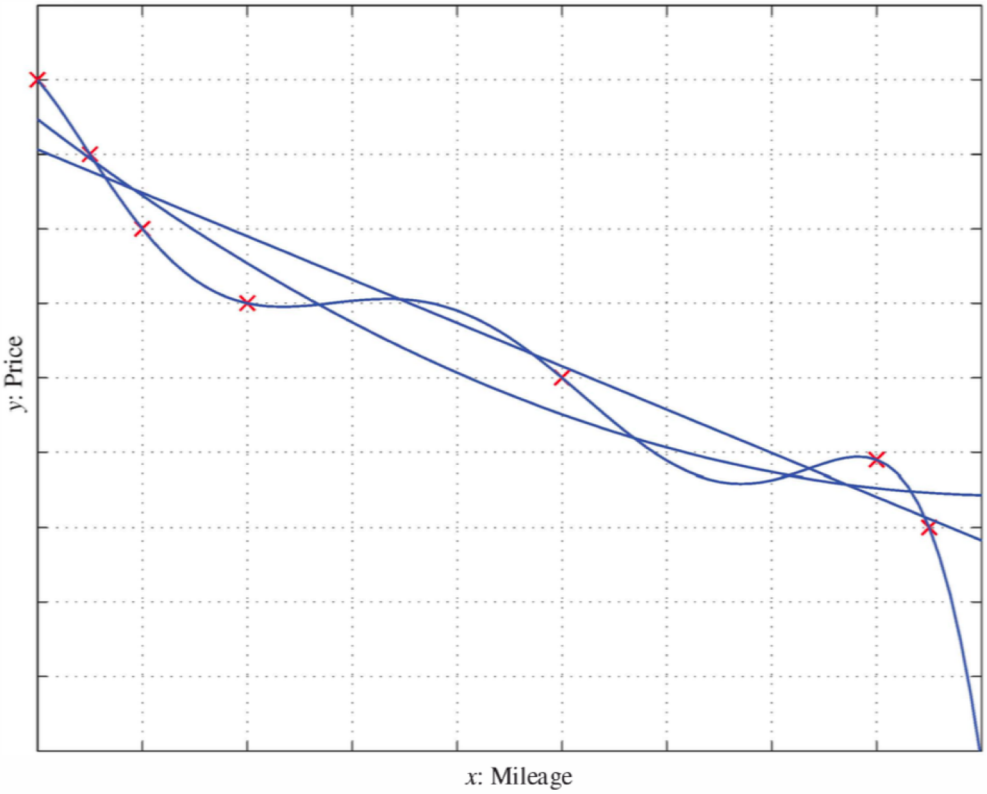
Linear Regression

- $\mathcal{D} = \{x^t, y^t\}_{t=1}^{N}$ where $y^t \in \mathbb{R}$
- $\mathcal{H} = w_0 x + w_1$ (frame: $\mathcal{H} = mx + b$)
  - Find values for $w_0, w_1 \in \mathbb{R}$

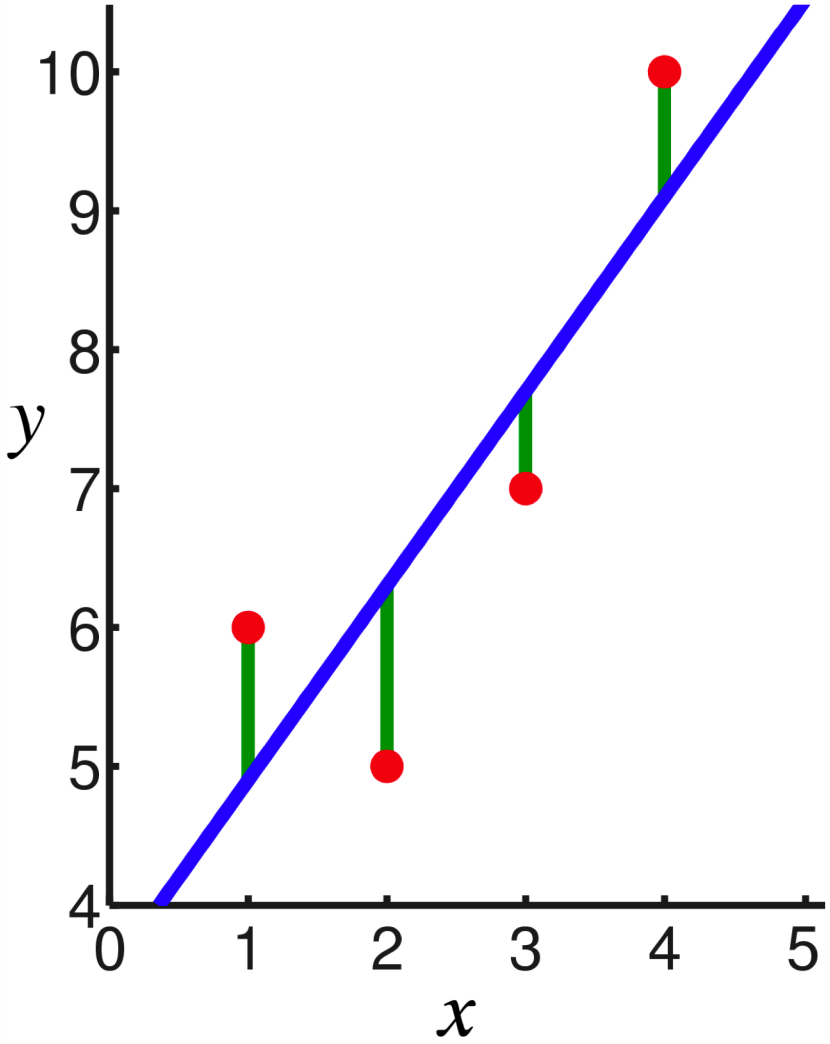


Refresher on R^2: https://www.youtube.com/watch?v=w2FKXOa0HGA

Error of Linear Regressions Models



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



https://en.wikipedia.org/wiki/Linear_regression#/media/File:Linear_least_squares_example2.png

Decision Trees

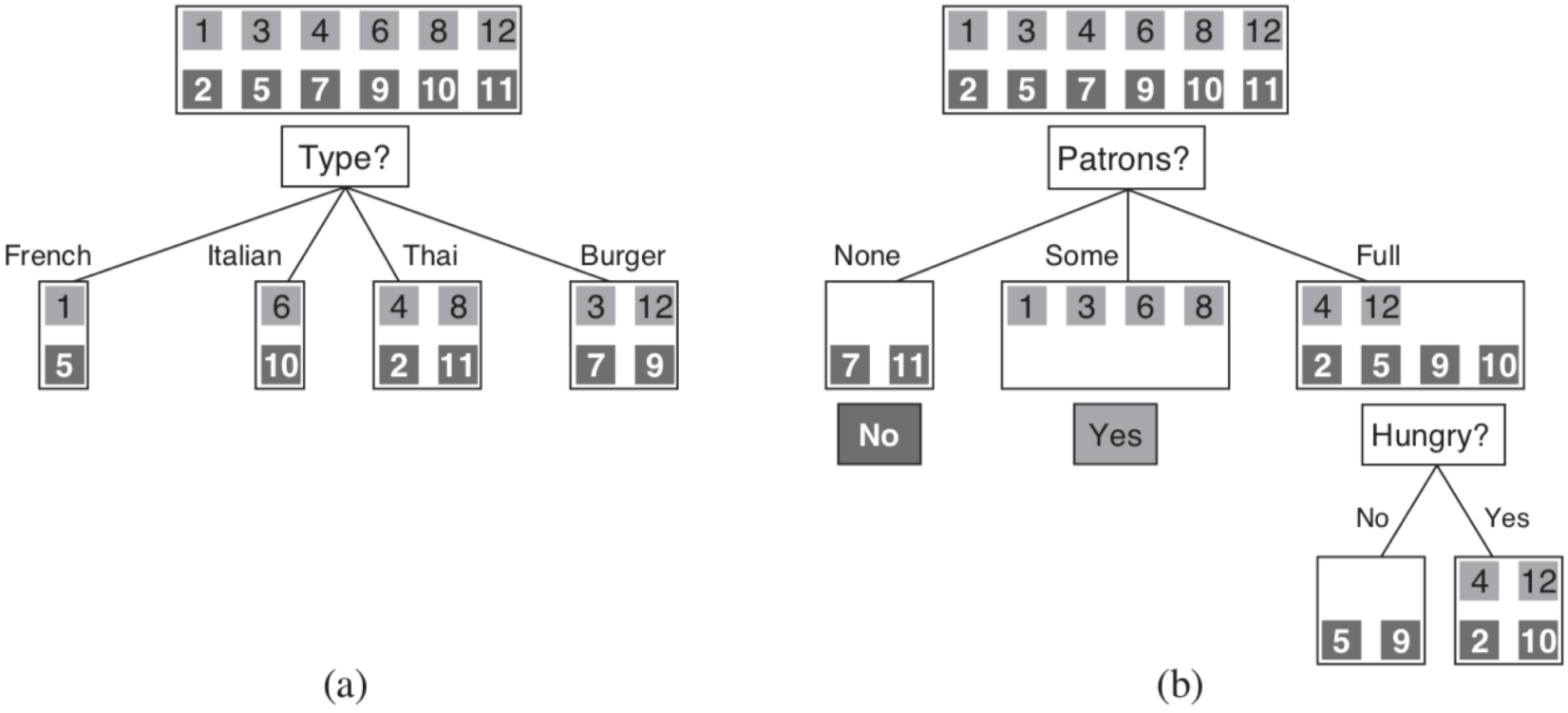


Figure 18.4 Splitting the examples by testing on attributes. At each node we show the positive (light boxes) and negative (dark boxes) examples remaining. (a) Splitting on *Type* brings us no nearer to distinguishing between positive and negative examples. (b) Splitting on *Patrons* does a good job of separating positive and negative examples. After splitting on *Patrons*, *Hungry* is a fairly good second test.

DT Generic Algorithm

function DECISION-TREE-LEARNING($examples$, $attributes$, $parent\_examples$) **returns** a tree

if $examples$ is empty **then return** PLURALITY-VALUE($parent\_examples$)
else if all $examples$ have the same classification **then return** the classification
else if $attributes$ is empty **then return** PLURALITY-VALUE($examples$)
else
  $A \leftarrow \text{argmax}_{a \in attributes}$ IMPORTANCE(a, $examples$)
  $tree \leftarrow$ a new decision tree with root test A
  for each value v_k of A **do**
    $exs \leftarrow \{e : e \in examples$ **and** $e.A = v_k\}$
    $subtree \leftarrow$ DECISION-TREE-LEARNING(exs, $attributes - A$, $examples$)
    add a branch to $tree$ with label $(A = v_k)$ and subtree $subtree$
  return $tree$

Figure 18.5 The decision-tree learning algorithm. The function IMPORTANCE is described in Section 18.3.4. The function PLURALITY-VALUE selects the most common output value among a set of examples, breaking ties randomly.

Entropy and Information Gain

Entropy: $$H(V) = \sum_k P(v_k) \log_2 \frac{1}{P(v_k)} = -\sum_k P(v_k) \log_2 P(v_k) \ .$$

$$B(q) = -(q \log_2 q + (1-q) \log_2 (1-q)) \ .$$

$$H(Goal) = B\left(\frac{p}{p+n}\right) \ .$$

$$Remainder(A) = \sum_{k=1}^{d} \frac{p_k+n_k}{p+n} B(\frac{p_k}{p_k+n_k}) \ .$$

$$Gain(A) = B(\frac{p}{p+n}) - Remainder(A) \ .$$

Generated DT

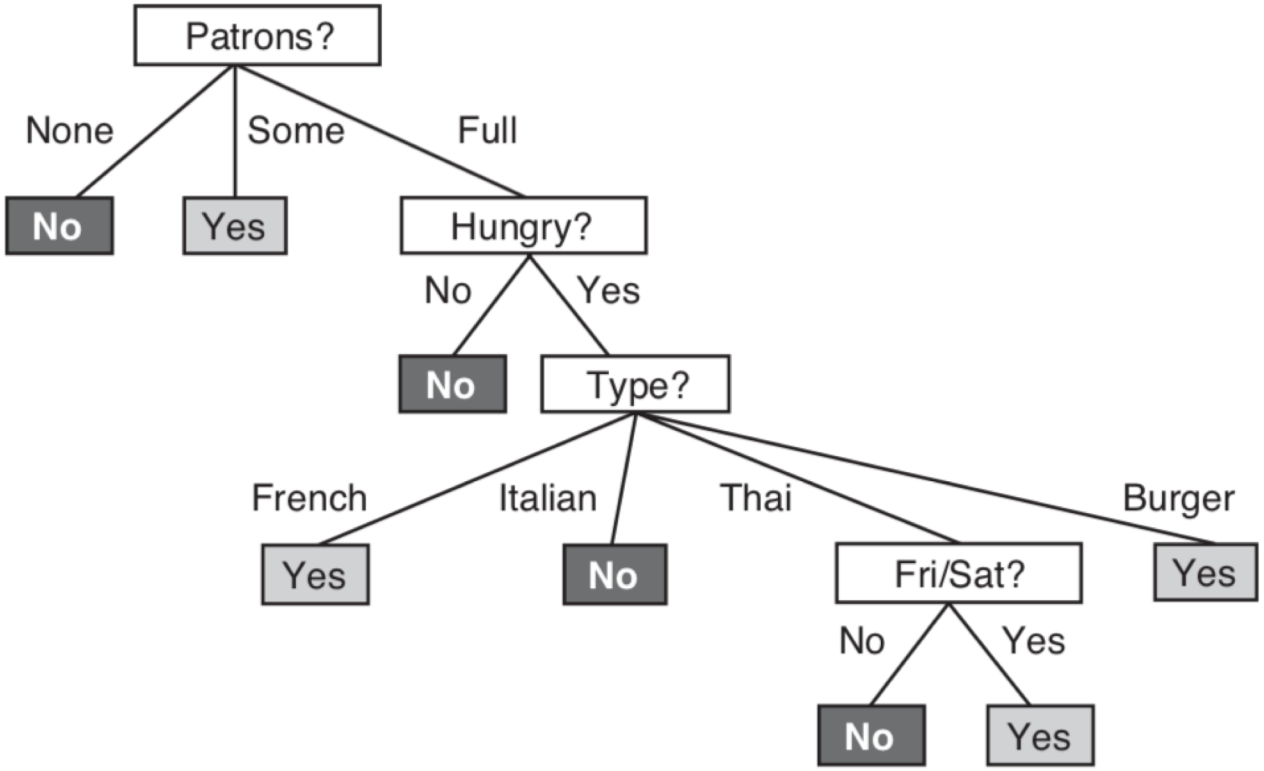
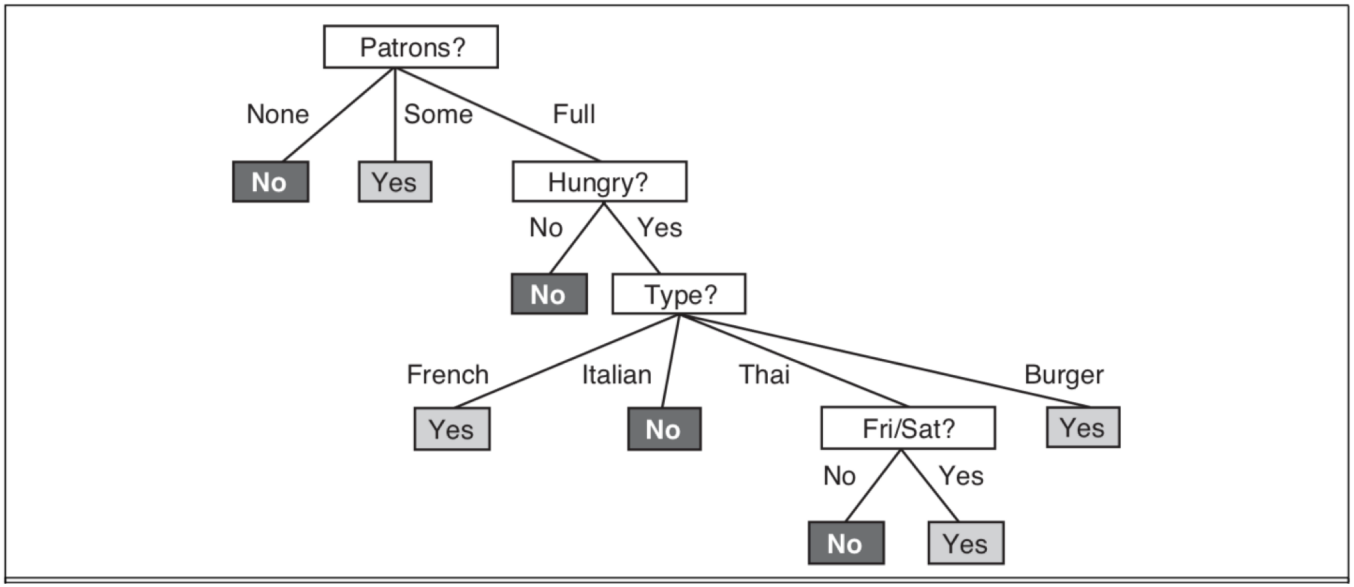
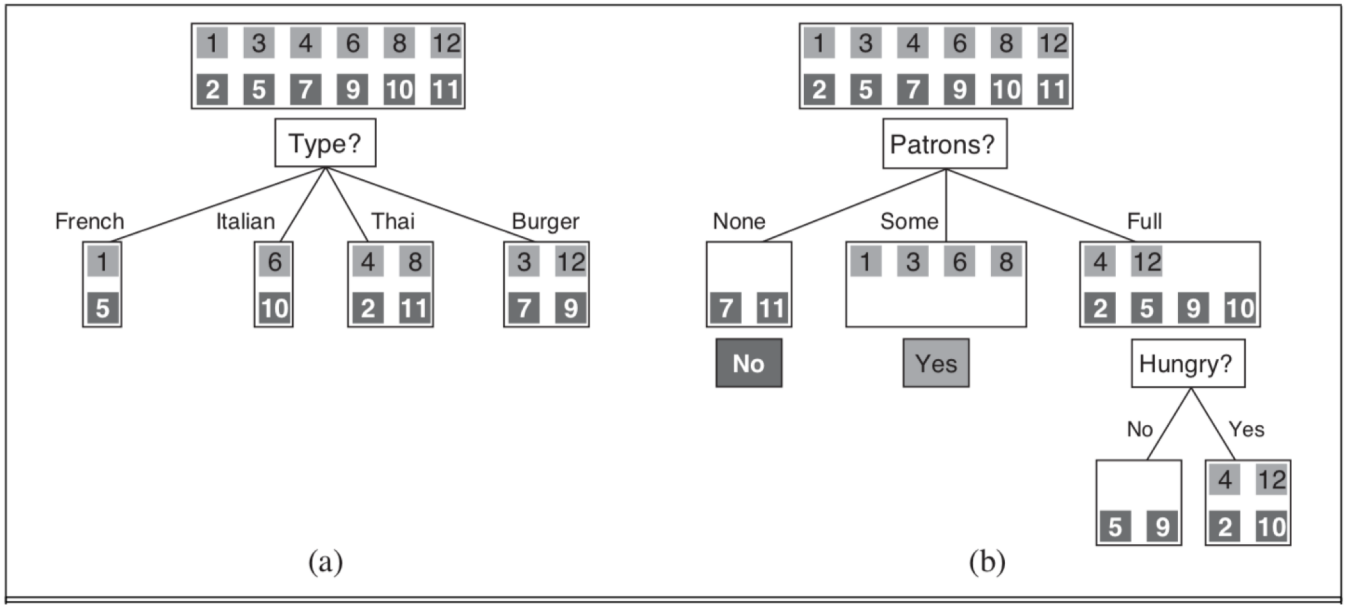


Figure 18.6 The decision tree induced from the 12-example training set.

Contextualizing Info Gain of Attrs.





First Case of Iterative Learning

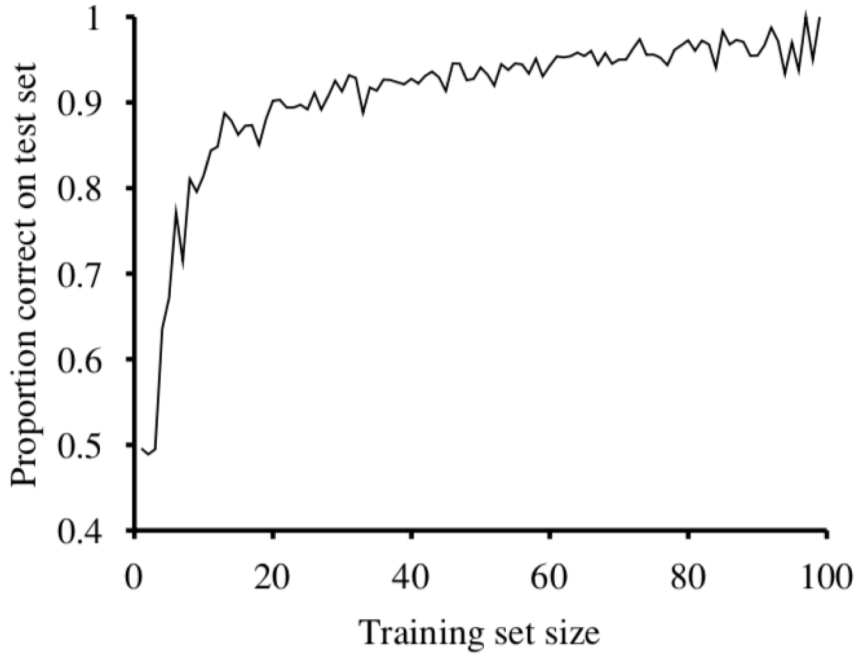


Figure 18.7 A learning curve for the decision tree learning algorithm on 100 randomly generated examples in the restaurant domain. Each data point is the average of 20 trials.

Data Set Partitions

- Data is partitioned into 3 sets
  - Training - generating a h
  - Validation - finetune/determine how good h is
  - Testing - application or final test case for h
    - Benchmark.
- Why this split?
  - Avoid overfitting via cross validation (p. 709).
  - Avoid biased error rates.
- How do I spit up my data?
  - 10%, 10%, 80%
  - 20%, 20%, 60%
  - 80%, 10%, 10% (common for deep methods)

Vapnik-Chervonenkis Dimension

- N points of data.
- 2^N ways ways to label that set as positive or negative.
- If there is $h \in \mathcal{H}$ that separates all positive and negative points of data it **shatters** N points.
- $\mathrm{VC}(\mathcal{H})$ is the number of points that $\mathcal{H}$ can shatter.
- Capacity of $\mathcal{H}$ to classify.