Learning From Examples Part A: 18.1 – 18.6

By Graham Foster

CISC 453

March 24, 2010

Learning?

- "In which we describe agents that can improve their behavior through diligent study of their own experiences"

- Types of algorithms examined:
  - Boolean Decision Trees
  - Boolean Decision Lists
  - Linear Classification

Types of Learning

- Machine accomplishes learning by:
  - Perceiving information
    - Factored representation: vector of attribute values
  - Choosing an appropriate action
    - Based on the result of testing for certain attributes
  - Receiving feedback on chosen action
    - To adjust the testing process and improve future performance

Feedback

- There are several types of learning that can take place based on the type of feedback gathered:
  - Unsupervised
  - Reinforcement
  - Supervised
    - vs semi-supervised

Unsupervised Learning

- No explicit feedback provided. Agent clusters similar groups of inputs that react the same way
  - Taxi driver may begin to recognize good traffic days vs bad traffic days without being explicitly told
  - Not necessarily preferring good traffic to bad traffic simply noticing the difference in situations.

Reinforcement Learning

- Learns from series of reinforcements
  - Taxi agent may not get tip at the end of a journey indicating it did something wrong.
  - Chess agent may receive two points at the end of a game indicating it did something right

- It is up to the agent to determine what actions were responsible for reinforcement.

Supervised Learning

- Agent observes input-output pairs which act as guide towards making decisions
- Both input situation as well as desired output value (after the fact) are provided by percepts.
  - Semi-supervised
    - May be systematic errors in provided examples (such as guessing peoples age or weight).
    - To learn from this is an unsupervised learning task

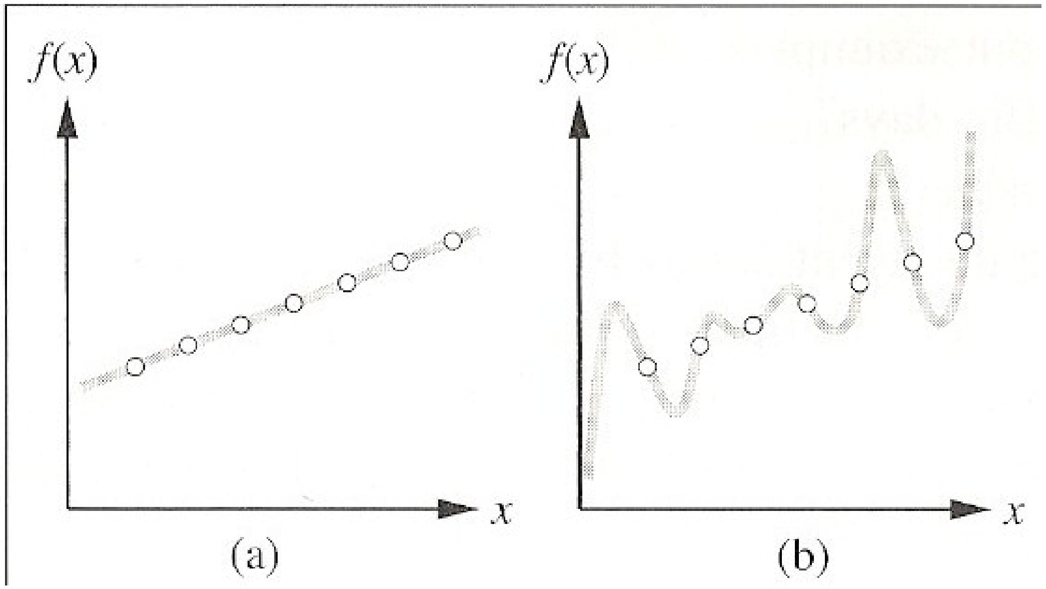
Supervised Learning Overview

- Training Set
  - Data with input-output pairs (x, y) are provided
  - Formation of decision making process based on these
  - f(x) = y
- Hypothesis
  - Logical function that decision making process creates and employs
  - h(x) = y`
- Test Set
  - New sets of data pairs are used to test performance of hypothesis
  - If h(x) = f(x)

Hypothesis Space

- Hypothesis space contains all possible logical functions that use n attributes
  - f(x) is assumed to be contained in this space
- The space could be exhaustively searched, however complexity is 2^{2^n}
  - 2^n rows in truth table, and answer column is a statement of length 2^n bits
  - This means with 10 attributes, as in the following example, there are 2^{1024} or about 10^{308} functions to choose from.

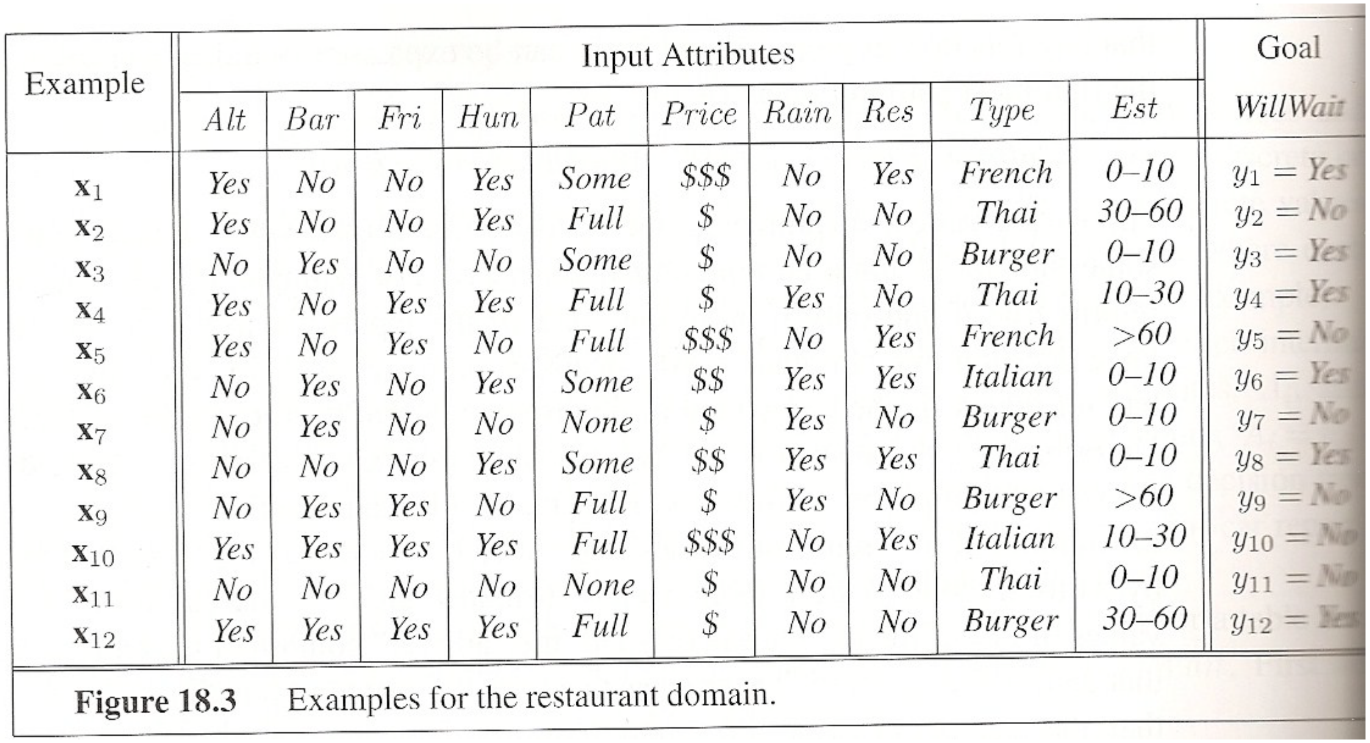
Hypothesis Space



- More than one hypothesis can fit the provided data and mimic the real function
- In general, simpler hypotheses are preferred

Restaurant Example

- Consider the decision process deciding whether to wait for a table at a restaurant



Example	Input Attributes										Goal
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
$\mathbf{x}_1$	*Yes*	*No*	*No*	*Yes*	*Some*	*\$\$\$*	*No*	*Yes*	*French*	*0–10*	$y_1 =$ *Yes*
$\mathbf{x}_2$	*Yes*	*No*	*No*	*Yes*	*Full*	*\$*	*No*	*No*	*Thai*	*30–60*	$y_2 =$ *No*
$\mathbf{x}_3$	*No*	*Yes*	*No*	*No*	*Some*	*\$*	*No*	*No*	*Burger*	*0–10*	$y_3 =$ *Yes*
$\mathbf{x}_4$	*Yes*	*No*	*Yes*	*Yes*	*Full*	*\$*	*Yes*	*No*	*Thai*	*10–30*	$y_4 =$ *Yes*
$\mathbf{x}_5$	*Yes*	*No*	*Yes*	*No*	*Full*	*\$\$\$*	*No*	*Yes*	*French*	*>60*	$y_5 =$ *No*
$\mathbf{x}_6$	*No*	*Yes*	*No*	*Yes*	*Some*	*\$\$*	*Yes*	*Yes*	*Italian*	*0–10*	$y_6 =$ *Yes*
$\mathbf{x}_7$	*No*	*Yes*	*No*	*No*	*None*	*\$*	*Yes*	*No*	*Burger*	*0–10*	$y_7 =$ *No*
$\mathbf{x}_8$	*No*	*No*	*No*	*Yes*	*Some*	*\$\$*	*Yes*	*Yes*	*Thai*	*0–10*	$y_8 =$ *Yes*
$\mathbf{x}_9$	*No*	*Yes*	*Yes*	*No*	*Full*	*\$*	*Yes*	*No*	*Burger*	*>60*	$y_9 =$ *No*
$\mathbf{x}_{10}$	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*\$\$\$*	*No*	*Yes*	*Italian*	*10–30*	$y_{10} =$ *No*
$\mathbf{x}_{11}$	*No*	*No*	*No*	*No*	*None*	*\$*	*No*	*No*	*Thai*	*0–10*	$y_{11} =$ *No*
$\mathbf{x}_{12}$	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*\$*	*No*	*No*	*Burger*	*30–60*	$y_{12} =$ *Yes*

Figure 18.3 Examples for the restaurant domain.

Restaurant Example

- Goal is to predict the value of the WillWait predicate by testing for certain attributes

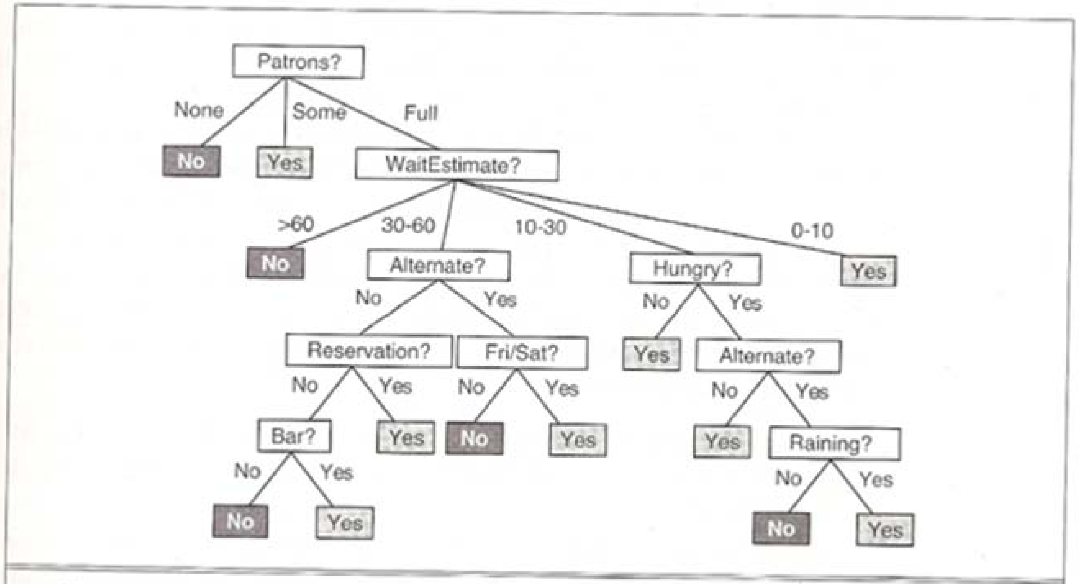
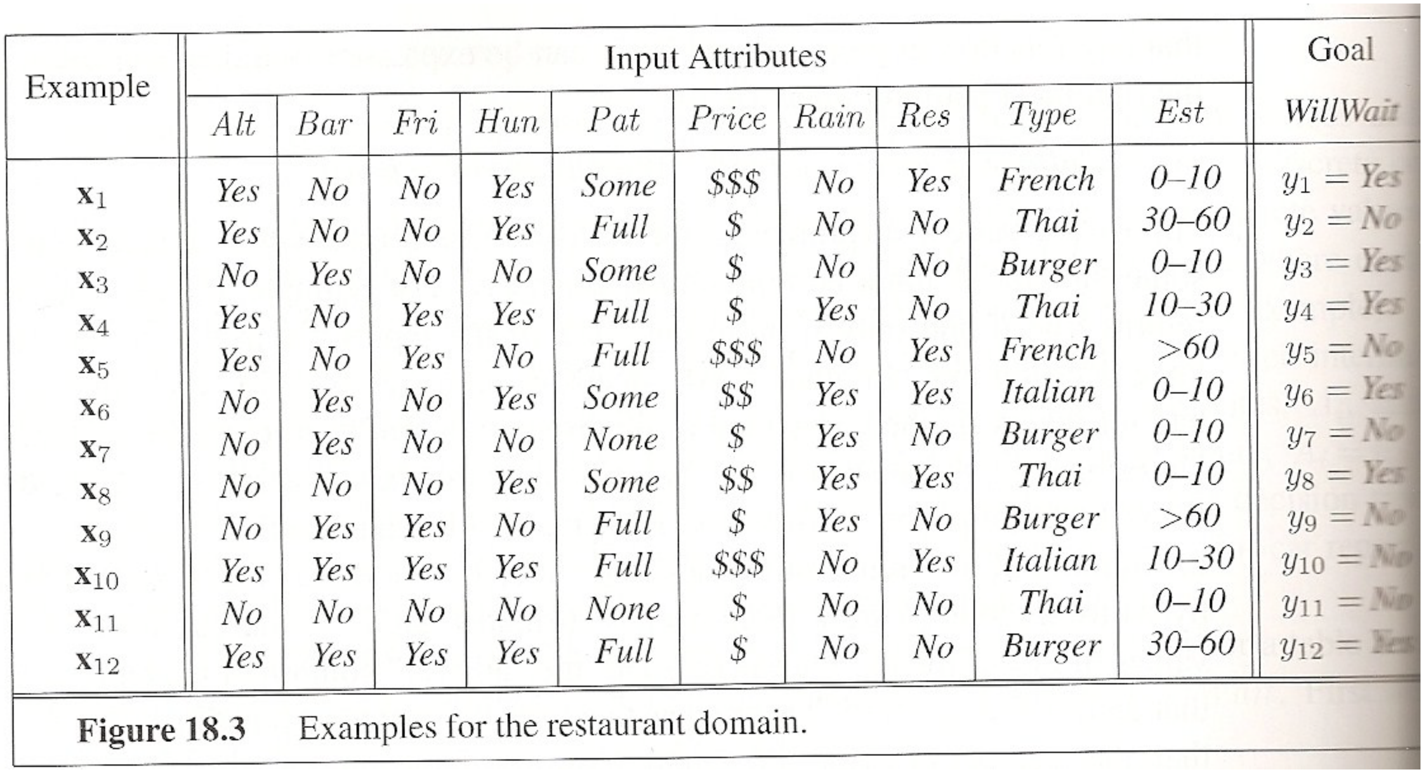


Figure 18.2 A decision tree for deciding whether to wait for a table.

- Input vector is supplied, determining which path to take, and leaf node is returned.

Testing Input Attributes

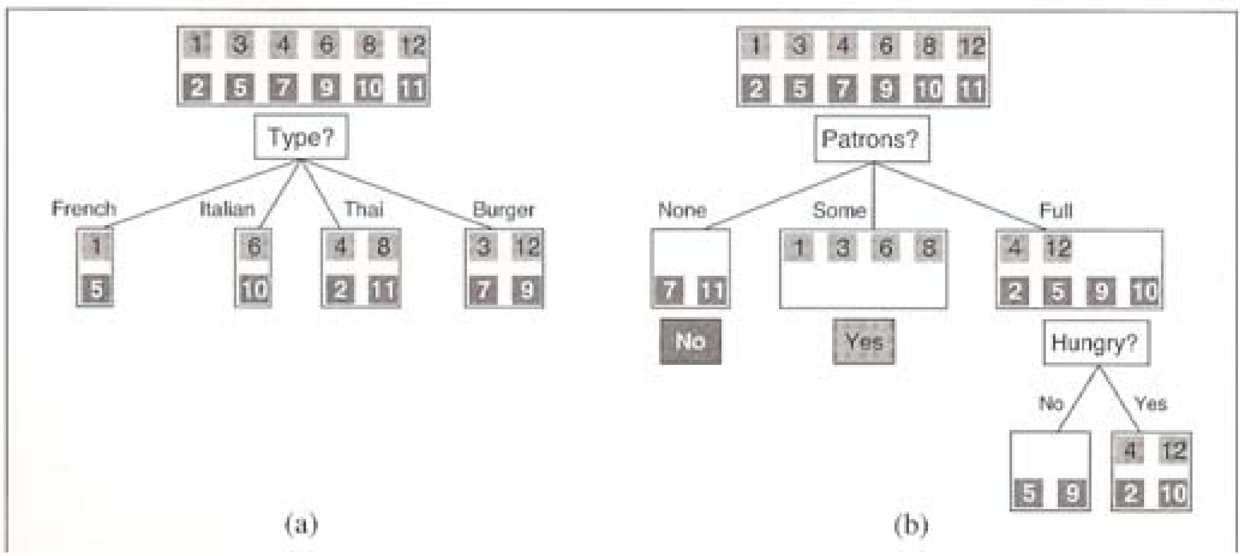
- Learning occurs as the agent decides which attributes to test in order to produce the desired value.



Example	Input Attributes										Goal
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
$\mathbf{x}_1$	*Yes*	*No*	*No*	*Yes*	*Some*	*\$\$\$*	*No*	*Yes*	*French*	*0–10*	$y_1 =$ *Yes*
$\mathbf{x}_2$	*Yes*	*No*	*No*	*Yes*	*Full*	*\$*	*No*	*No*	*Thai*	*30–60*	$y_2 =$ *No*
$\mathbf{x}_3$	*No*	*Yes*	*No*	*No*	*Some*	*\$*	*No*	*No*	*Burger*	*0–10*	$y_3 =$ *Yes*
$\mathbf{x}_4$	*Yes*	*No*	*Yes*	*Yes*	*Full*	*\$*	*Yes*	*No*	*Thai*	*10–30*	$y_4 =$ *Yes*
$\mathbf{x}_5$	*Yes*	*No*	*Yes*	*No*	*Full*	*\$\$\$*	*No*	*Yes*	*French*	*>60*	$y_5 =$ *No*
$\mathbf{x}_6$	*No*	*Yes*	*No*	*Yes*	*Some*	*\$\$*	*Yes*	*Yes*	*Italian*	*0–10*	$y_6 =$ *Yes*
$\mathbf{x}_7$	*No*	*Yes*	*No*	*No*	*None*	*\$*	*Yes*	*No*	*Burger*	*0–10*	$y_7 =$ *No*
$\mathbf{x}_8$	*No*	*No*	*No*	*Yes*	*Some*	*\$\$*	*Yes*	*Yes*	*Thai*	*0–10*	$y_8 =$ *Yes*
$\mathbf{x}_9$	*No*	*Yes*	*Yes*	*No*	*Full*	*\$*	*Yes*	*No*	*Burger*	*>60*	$y_9 =$ *No*
$\mathbf{x}_{10}$	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*\$\$\$*	*No*	*Yes*	*Italian*	*10–30*	$y_{10} =$ *No*
$\mathbf{x}_{11}$	*No*	*No*	*No*	*No*	*None*	*\$*	*No*	*No*	*Thai*	*0–10*	$y_{11} =$ *No*
$\mathbf{x}_{12}$	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*\$*	*No*	*No*	*Burger*	*30–60*	$y_{12} =$ *Yes*

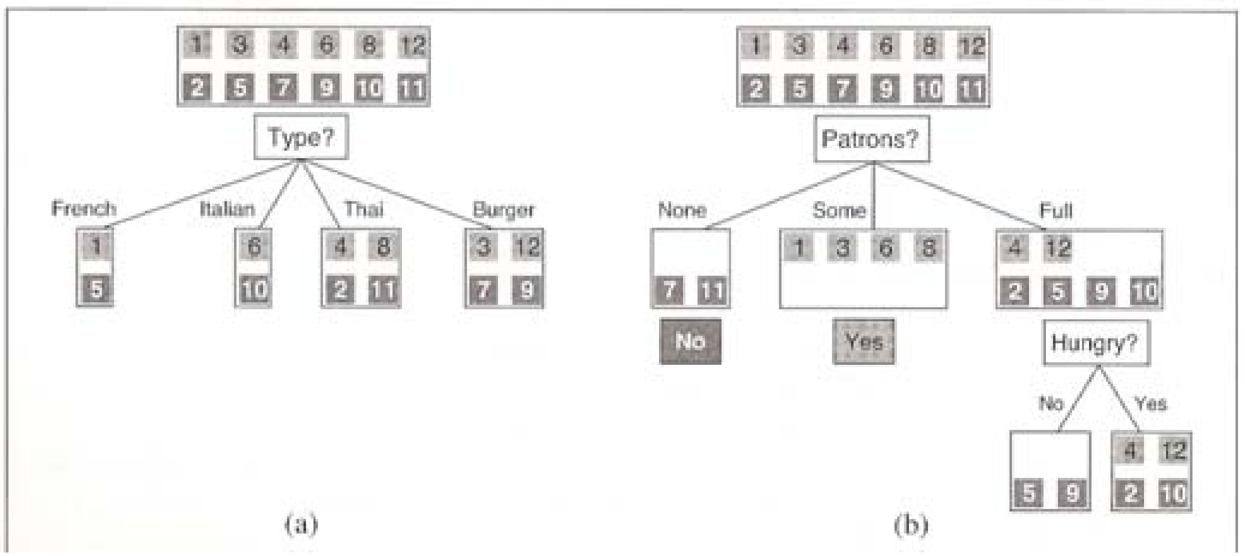
Figure 18.3 Examples for the restaurant domain.

Testing Input Attributes



- Testing "Type" attribute is not as beneficial as testing "Patrons"
- Attribute chosen based on greedy selection of attribute with highest entropy.
- Algorithm then recursively chooses new attribute from smaller set of examples to generate subtree

Testing Input Attributes



- All remaining examples are either Yes or No
- Some of each leads to further recursion
- No examples follow this path
- No more attributes remain untested

Decision Tree

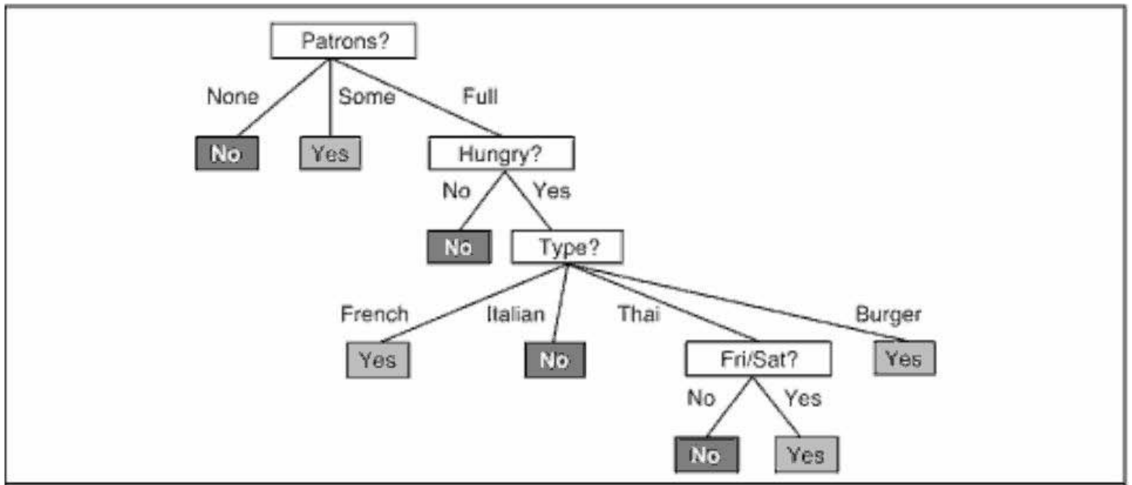


Figure 18.6 The decision tree induced from the 12-example training set.

- This tree was generated following the greedy selection of attributes algorithm.
- This tree is complete and simpler than the earlier tree
- System has stumbled on the realization that "Rain" and "Reservation" are not necessary for complete classification of the training set.

Decision Tree Performance

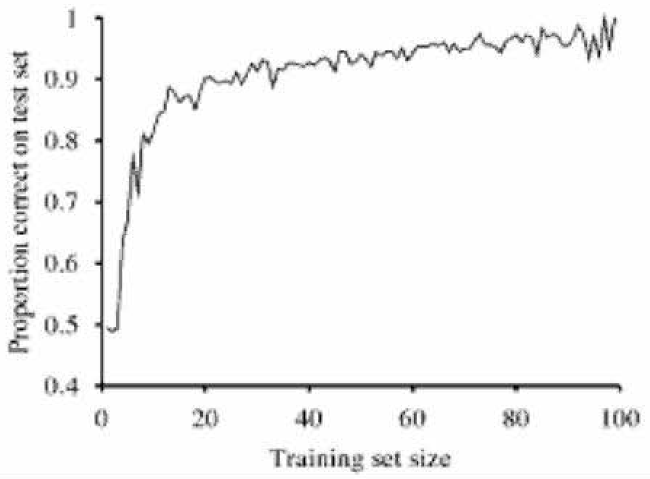


Figure 18.7 A learning curve for the decision tree learning algorithm on 100 randomly generated examples in the restaurant domain. Each data point is the average of 20 trials.

- A larger training set encompasses a wider variety of situations and therefore better prepares the system for new input

Generalization and Overfitting

- Overfitting is caused by the inappropriate classification of input based on an attribute value
  - Caused by too small a training set
    - Generalization is not meaningful
  - Caused by too many attributes
    - Algorithm infers causal relationship among irrelevant attributes
      - Die rolling example (colour, size, fingers crossed, previous roll)

Tree Pruning

- Goal of pruning is to prevent overfitting
  - To discover irrelevant attributes, one would expect that the result of sorting by this would not affect ratio of positive to negative results.
  - This determined by near 0 entropy of selected attribute

Significance Testing

- Basis of significance testing covered in STAT 263
- Start assuming there is no relationship between attribute value and result
  - Null hypothesis
- Perform χ^2 Testing
  - Commonly at 1% or 5% level
- Prune only in bottom-up direction

Best of Both Worlds?

- The similarity of greedy entropy selection, and significance testing seems to suggest using both.
    - This ignores situations where there may be no immediate attribute which is highly significant to results
        - Ex. XOR function

- Must first generate tree, and then prune when appropriate

Decision Tree Limitations

- Drawbacks:
  - Missing Data
    - Some attributes not provided
  - Multivalued Attributes
    - Worst case – exact time of data provides highest entropy
  - Continuous Values
    - Must be grouped into discrete ranges
  - Continuous Valued Output
    - Out put is not boolean

Error Checking

- Error checking methods:
    - Error rate
        - Train first, then test using training data
    - Holdout Cross-Validation
        - Randomly divide data set into training and testing portions
    - k-Fold Cross-Validation
        - Divide data into k-subsets and repeat Holdout Cross Validation for each
    - Leave-One-Out-Cross-Validation
        - Train with all but one item of data
        - Use remaining item for testing
        - Repeat until each item has been used as test item

Choosing Complexity

- As seen earlier, more than one hypothesis can fit a given set of data
- Simpler functions are preferred, but complex ones are available
- Algorithm modified to take tree depth as argument
  - Depth iteratively increases it until a suitably accurate hypothesis is generated

Choosing Complexity

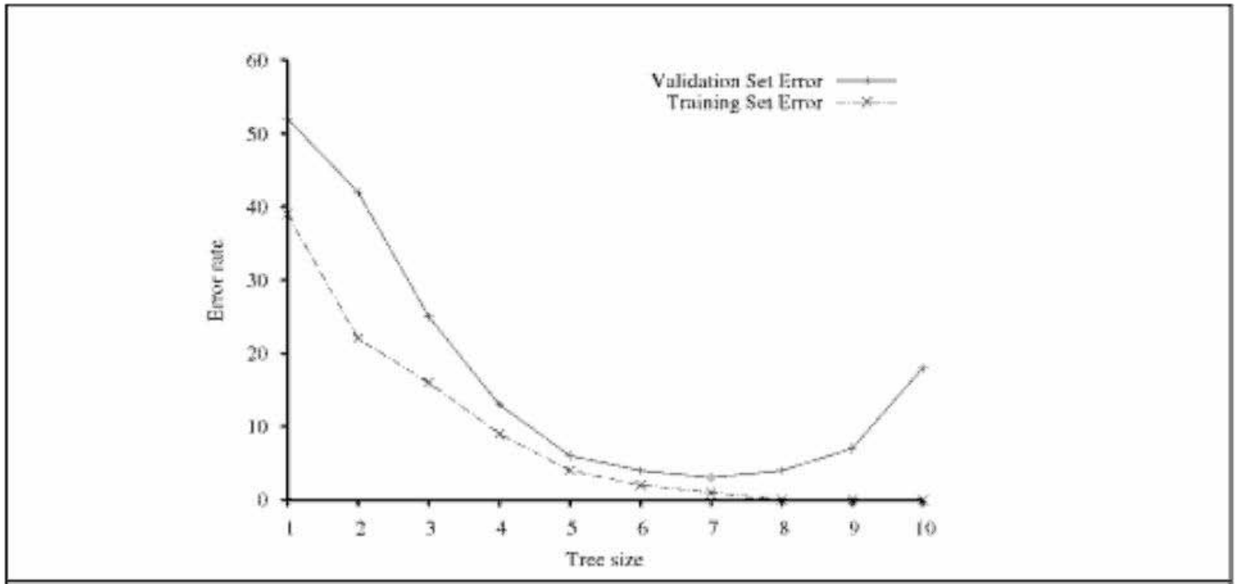


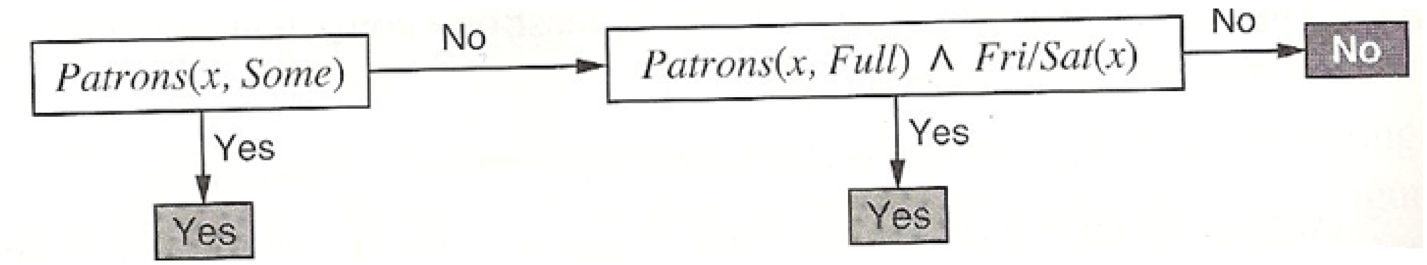
Figure 18.9 Error rates on training data (lower, dashed line) and validation data (upper, solid line) for different size decision trees. We stop when the training set error rate asymptotes, and then choose the tree with minimal error on the validation set; in this case the tree of size 7 nodes.

- Optimal balance of accuracy and simplicity lies at depth = 7

PAC Algorithms

- Nomenclature for an algorithm which is *probably approximately correct*
  - Probably:
    - "…any hypothesis that is seriously wrong will almost certainly be 'found out' with high probability after a small number of examples…"
  - Approximately
    - Amount of error determined through error testing to be below pre-determined constant

Decision Lists



- Algorithm inspects data for:
    - Which *combination* of attributes yields highest entropy
    - Must generate at least one leaf node
- Recurse on remaining data with example(s) in leaf node removed

Decision Lists

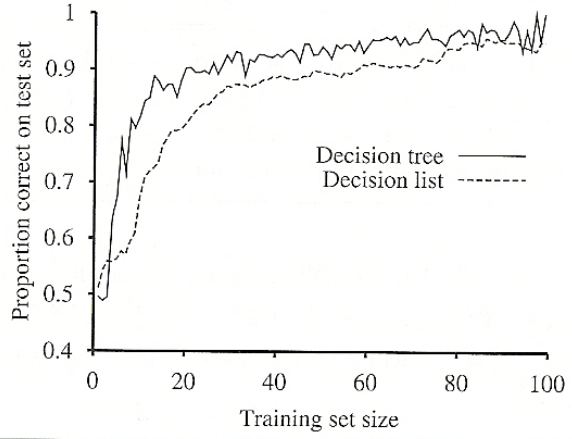


Figure 18.12 Learning curve for DECISION-LIST-LEARNING algorithm on the restaurant data. The curve for DECISION-TREE-LEARNING is shown for comparison.

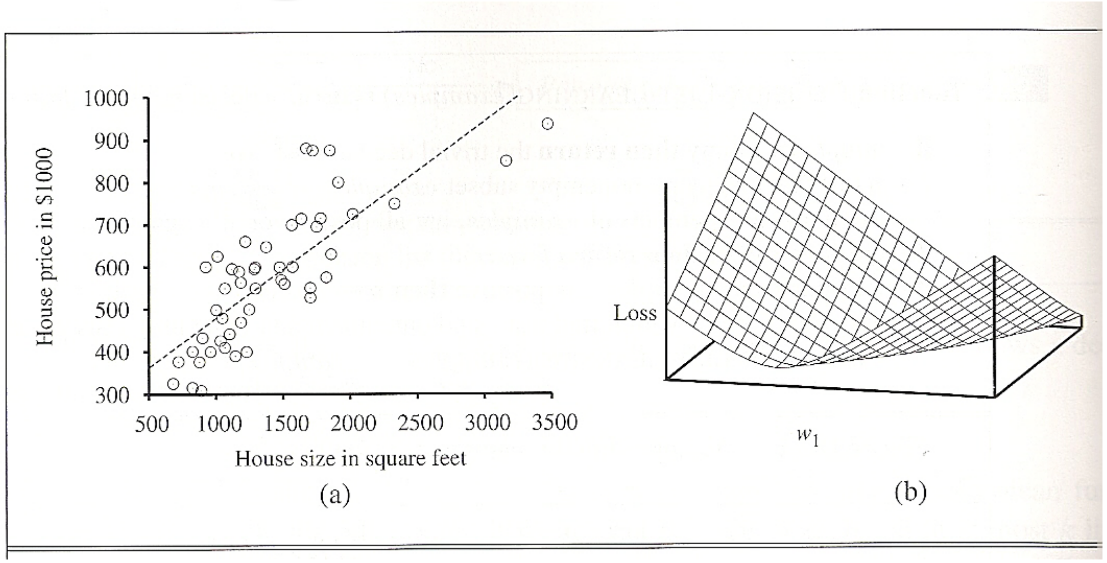
- Poorer performance when the training set size is small, due to overfitting
- Can recognize higher order patterns in data

Linear Regression

- To deal with problems of continuous input in the real world, must accept approximate hypotheses
  - Real Estate

- Generate loss function which sums the difference between generated result and desired result

Linear Regression

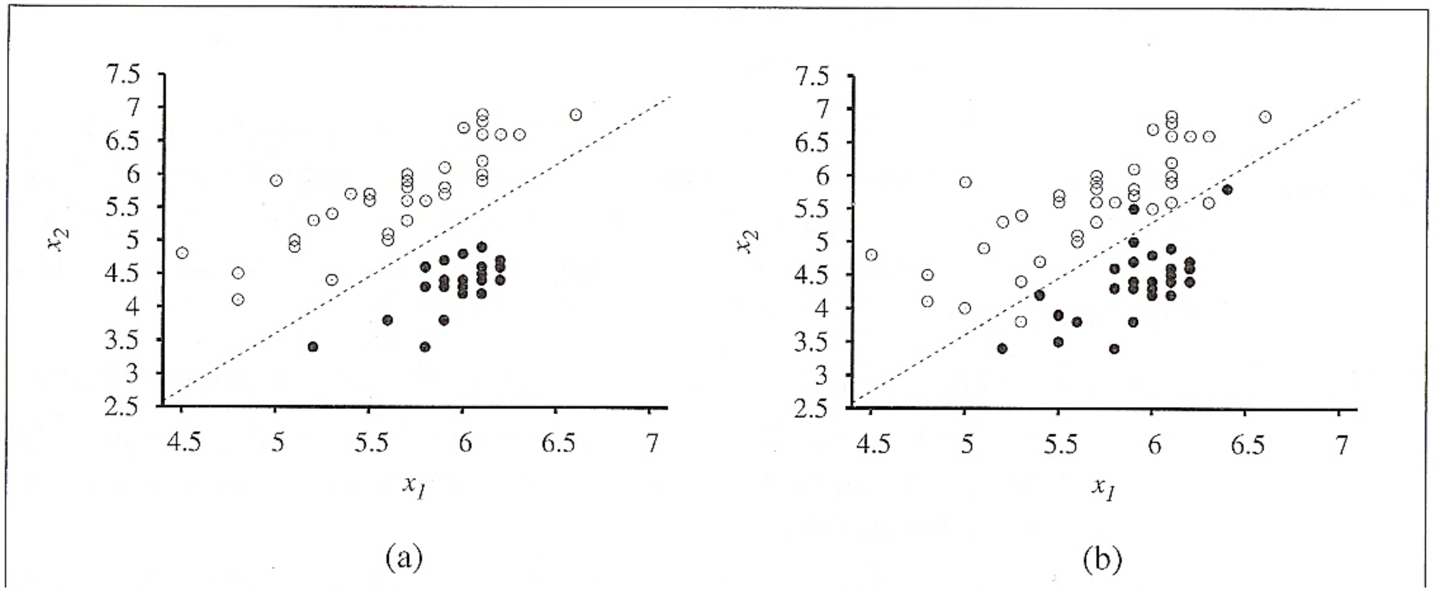


- Generates line of best fit
- Graphing the loss function generates a convex plane (b)
- $y = w_1(x_1) + w_2(x_2) + c$
- Adjusts w values to step “downhill” at either constant or decreasing rate

Linear Regression

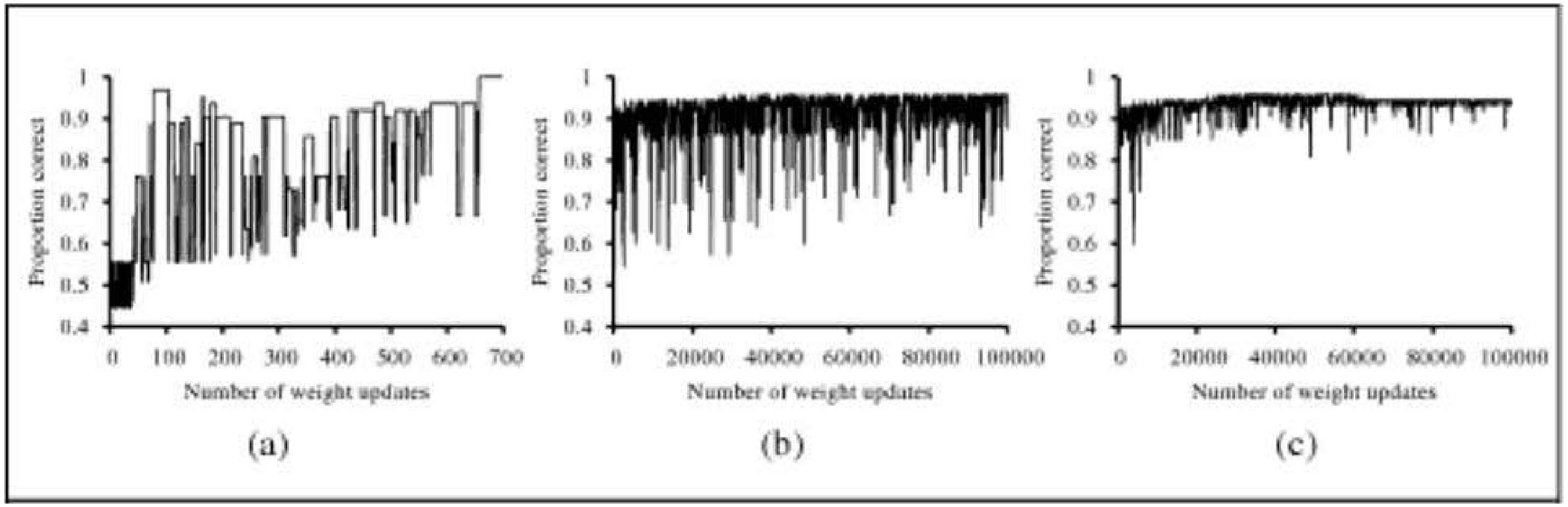
- When adjusting hypothesis point by point appropriate, consistent function is a slow process
  - Exact conformation to data may overfit due to coincidental local minima
  - Convergence on an accurate function can take many iterations through the training set to be appropriate

Linear Regression



- Data shows examples of two types of seismic readings
  - Hollow dots represent those caused by earthquakes
  - Solid dots represent those caused by underground weapons tests
- (a) is linearly separable, (b) is not

Linear Regression



- (a) shows performance of algorithm on linearly separable data
- (b) shows performance on data that is not linearly separable
- (c) shows performance when gradually decreasing learning factor is implemented

Logistic Regression

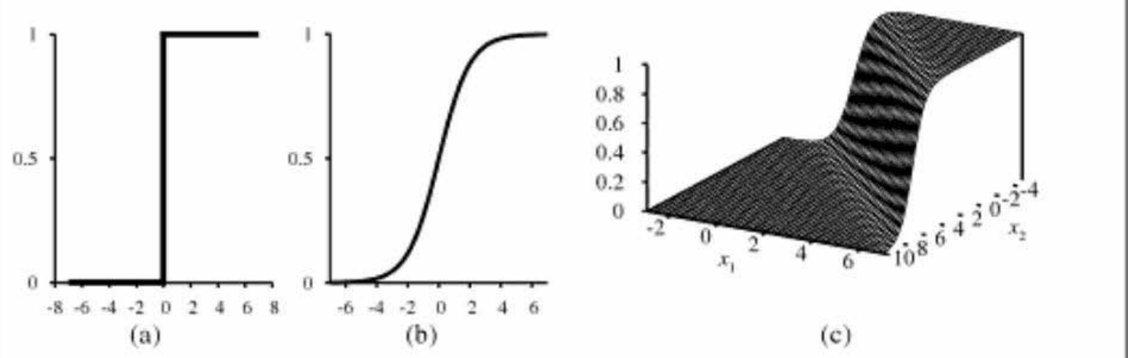


Figure 18.17 (a) The hard threshold function *Threshold*(z) with 0/1 output. Note that the function is nondifferentiable at $z = 0$. (b) The logistic function, *Logistic*$(z) = \frac{1}{1+e^{-z}}$, also known as the sigmoid function. (c) Plot of a logistic regression hypothesis $h_{\mathbf{w}}(\mathbf{x}) = Logistic(\mathbf{w} \cdot \mathbf{x})$ for the data shown in Figure 18.15(b).

- Allows the system to output any value between 0 and 1
- Can be thought of as expressing the degree of confidence by which the data can be classified

Logistic Regression

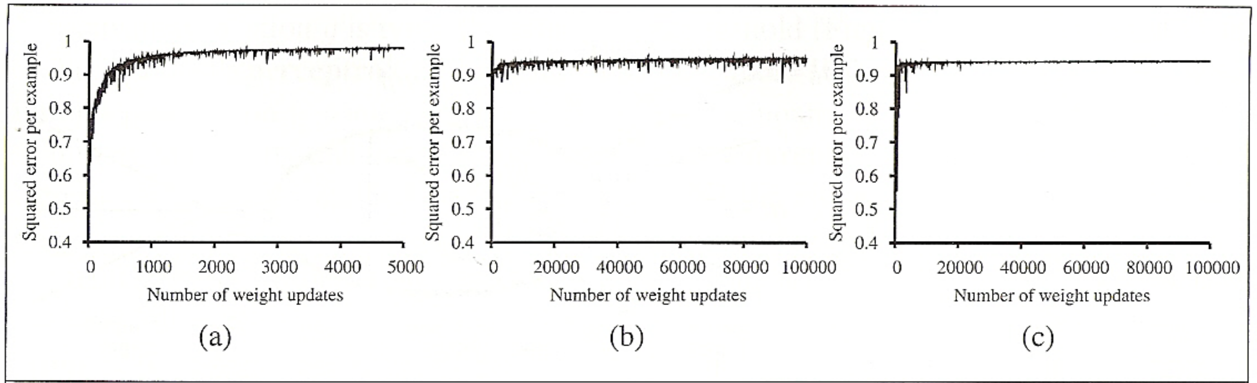


Figure 18.18 Repeat of the experiments in Figure 18.16 using logistic regression and squared error. The plot in (a) covers 5000 iterations rather than 1000, while (b) and (c) use the same scale.

- Although the calculus needed to arrive at each weight adjustment is more complex, the function can be honed in a much more effective way
- The lack of direct contradictions allows the graph to reach stability

Summary

- Decision Trees
    - Start with training set
    - Generate hypothesis
    - Test on new data
- Decision Lists
    - Test more than one attribute at a time
- Linear Regression
    - Line of best fit to classify data
    - Line can be softened to allow reporting of confidence