Artificial Intelligence

Hai Thi Tuyet Nguyen

Outline

CHAPTER 1: INTRODUCTION (CHAPTER 1)

CHAPTER 2: INTELLIGENT AGENTS (CHAPTER 2)

CHAPTER 3: SOLVING PROBLEMS BY SEARCHING (CHAPTER 3)

CHAPTER 4: INFORMED SEARCH (CHAPTER 3)

CHAPTER 5: LOGICAL AGENT (CHAPTER 7)

CHAPTER 6: FIRST-ORDER LOGIC (CHAPTER 8, 9)

CHAPTER 7: QUANTIFYING UNCERTAINTY(CHAPTER 13)

CHAPTER 8: PROBABILISTIC REASONING (CHAPTER 14)

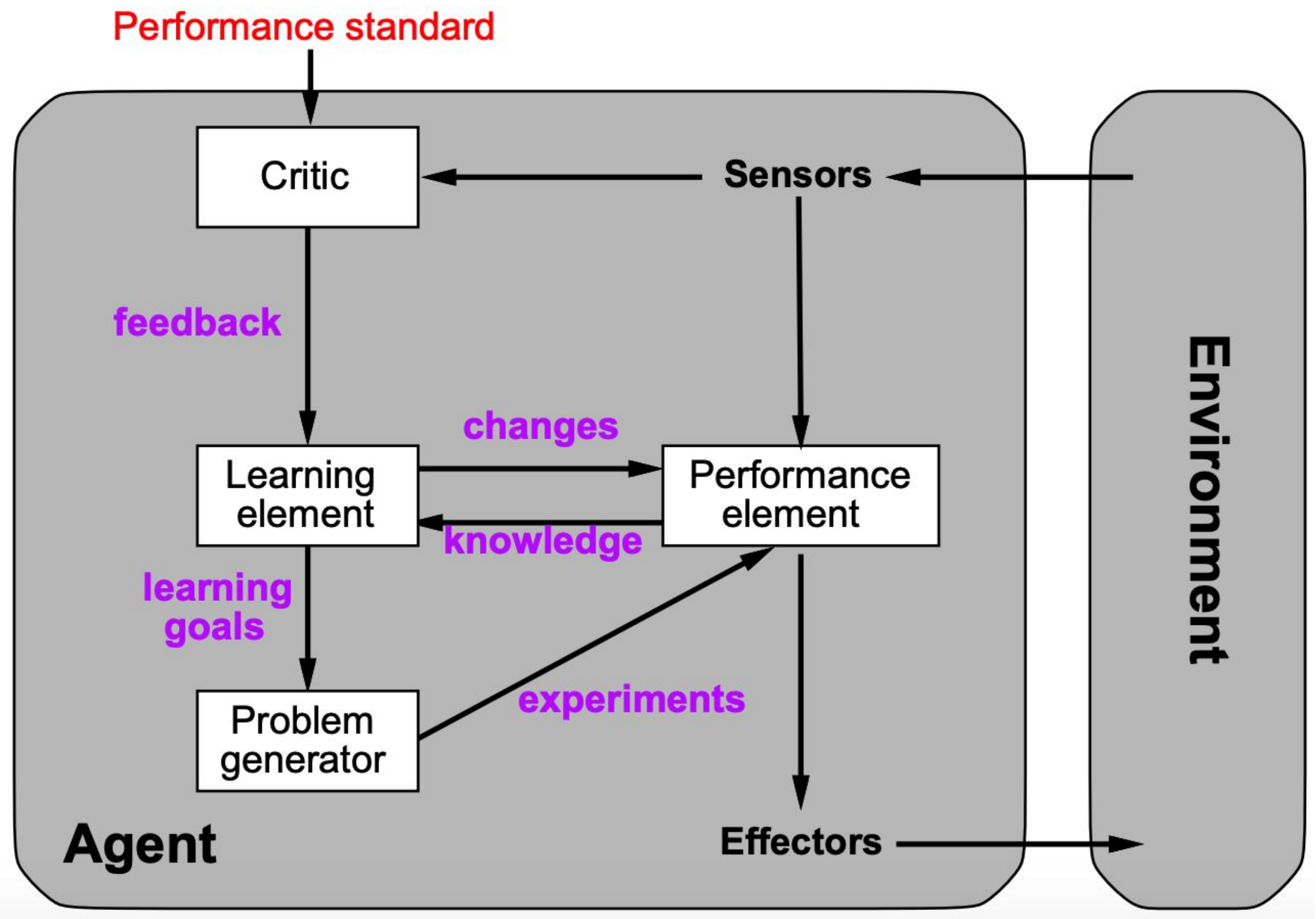
CHAPTER 9: LEARNING FROM EXAMPLES (CHAPTER 18)

CHAPTER 9: LEARNING FROM EXAMPLES

9.1 Forms Of Learning
9.2 Supervised Learning
9.3 Learning Decision Trees
9.4 Evaluating And Choosing The Best Hypothesis
9.5 Regression And Classification With Linear Models
9.6 Artificial Neural Networks

9.1 Forms Of Learning

Learning agents



Learning element

Design of learning element is dictated by

- what type of performance element is used
  - E.g., logical agent
- how that functional component is represented
  - E.g., state transition model
- which functional component is to be learned
  - E.g., logic language
- what kind of feedback is available
  - E.g., outcome

9.1 Forms Of Learning

- What feedback is available to learn from: it depends on three types of feedback
  - In **reinforcement learning** the agent learns from a series of reinforcements - rewards or punishments.
    - E.g.: the points for a win at the end of a chess game tells the agent it did something right.
  - In **unsupervised learning** the agent learns patterns in the input even though no explicit feedback is supplied.
    - E.g.: a taxi agent might gradually develop a concept of “good traffic days” and “bad traffic days” without ever being given labeled examples of each by a teacher.
  - In **supervised learning** the agent observes some example input–output pairs and learns a function that maps from input to output.

9.2 Supervised Learning

9.2 Supervised Learning

- The task of supervised learning:

  Given a **training set** of N example input–output pairs

  $(x_1,y_1), (x_2,y_2),\ldots(x_N,y_N)$,

  where each y_j was generated by an unknown function $y = f(x)$,

  Problem: find a **hypothesis** h

  such that $h \approx f$

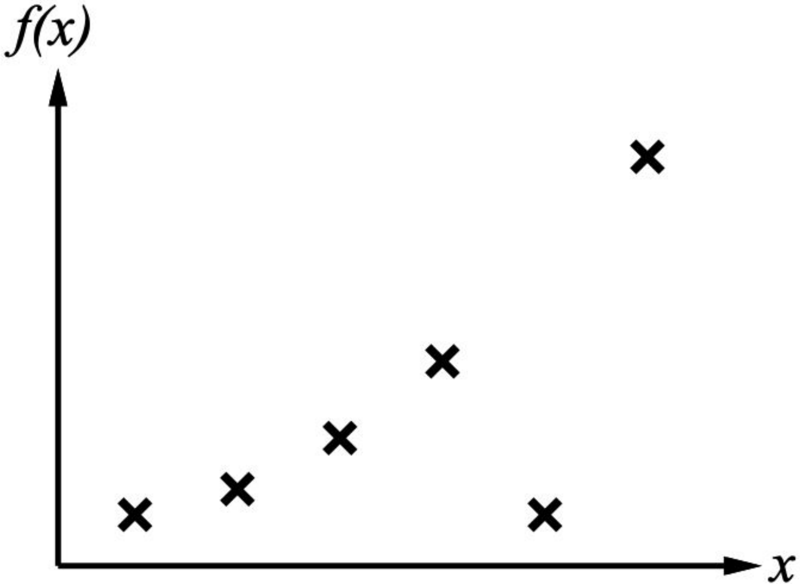
  given a training set of examples

9.2 Supervised Learning

Inductive learning method

Construct/adjust h to agree with f on training set

(h is **consistent** if it agrees with f on all examples)

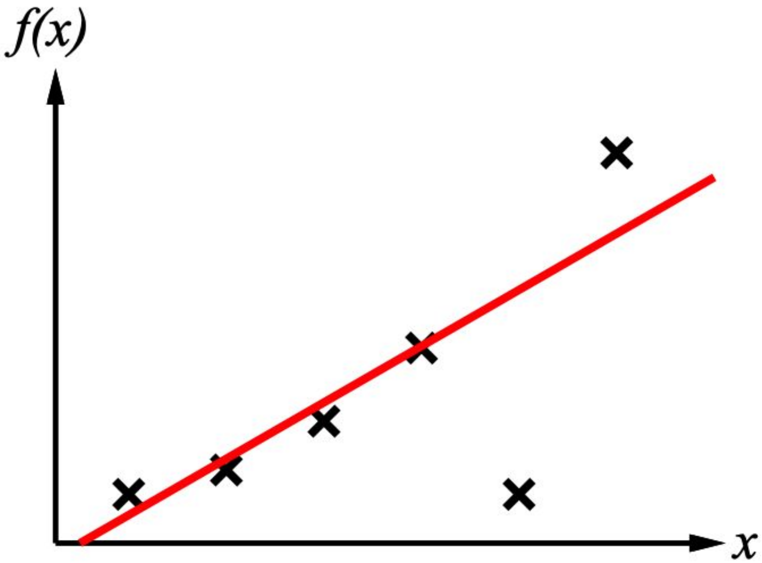


9.2 Supervised Learning

Inductive learning method

Construct/adjust h to agree with f on training set

(h is **consistent** if it agrees with f on all examples)

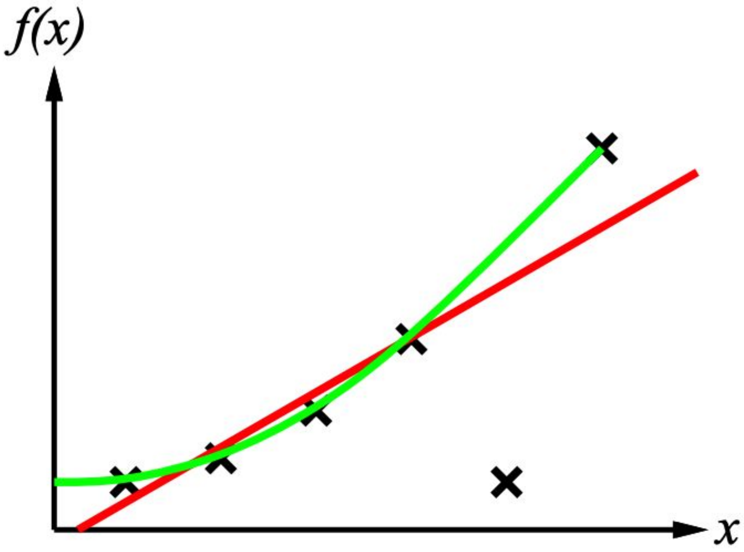


9.2 Supervised Learning

Inductive learning method

Construct/adjust h to agree with f on training set

(h is **consistent** if it agrees with f on all examples)

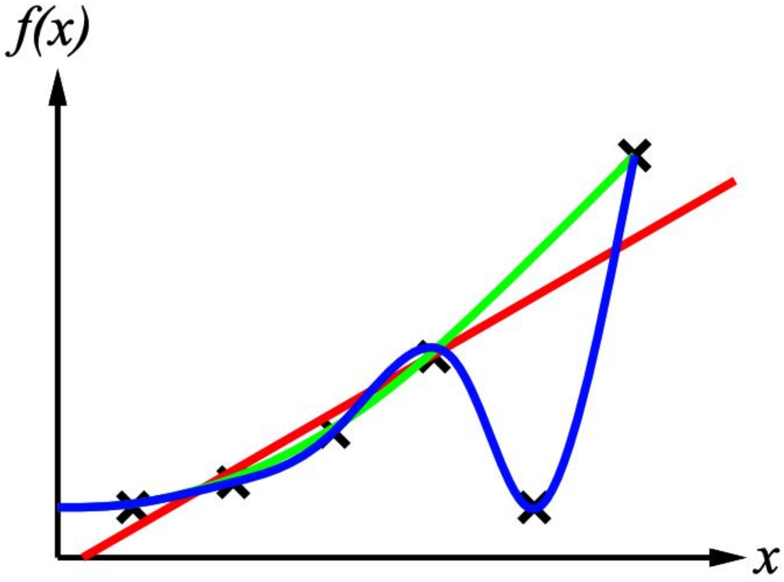


9.2 Supervised Learning

Inductive learning method

Construct/adjust h to agree with f on training set

(h is **consistent** if it agrees with f on all examples)



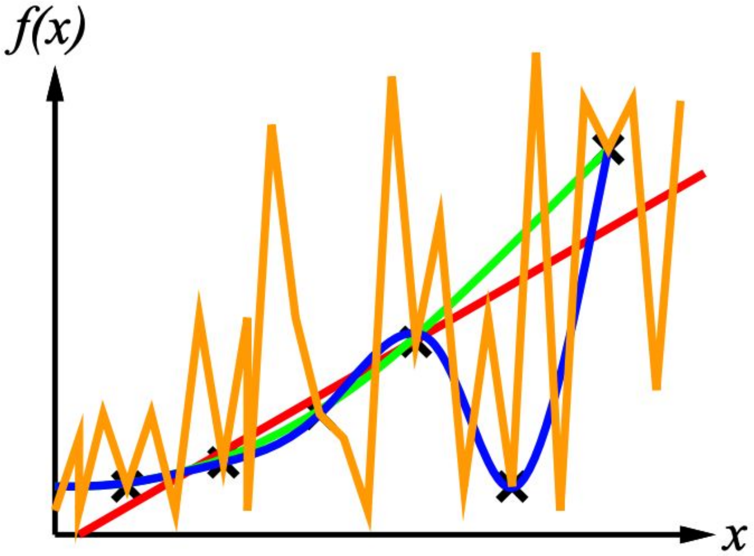
9.2 Supervised Learning

Inductive learning method

Construct/adjust h to agree with f on training set

(h is **consistent** if it agrees with f on all examples)

Ockham's razor: maximize a combination of consistency and simplicity



9.2 Supervised Learning

- To measure the accuracy of a hypothesis h, we give it a **test set** of examples that are distinct from the training set.
- A hypothesis **generalizes** h well if it correctly predicts the value of y for novel examples.

9.2 Supervised Learning

- Types of supervised learning:
  - **Classification**: **y** is one of a finite set of values (e.g., sunny, cloudy or rainy)
  - **Regression**: **y** is a number, the learning problem is called regression.

9.3 Learning Decision Trees

9.3 The decision tree representation

A decision tree represents a function that takes as input a vector of attribute values and returns a "decision"—a single output value.

9.3 The decision tree representation

A decision tree reaches its decision by performing a sequence of tests.

- Each internal node: a test of the values of the attribute, A_i
- The branches from the node: the values of the attribute, $A_i = v_{ik}$.
- Each leaf node: a value to be returned by the function.

Expressiveness of decision trees: $Goal \Leftrightarrow (Path_1 \lor Path_2 \lor \cdots)$

9.3 Inducing decision trees from examples

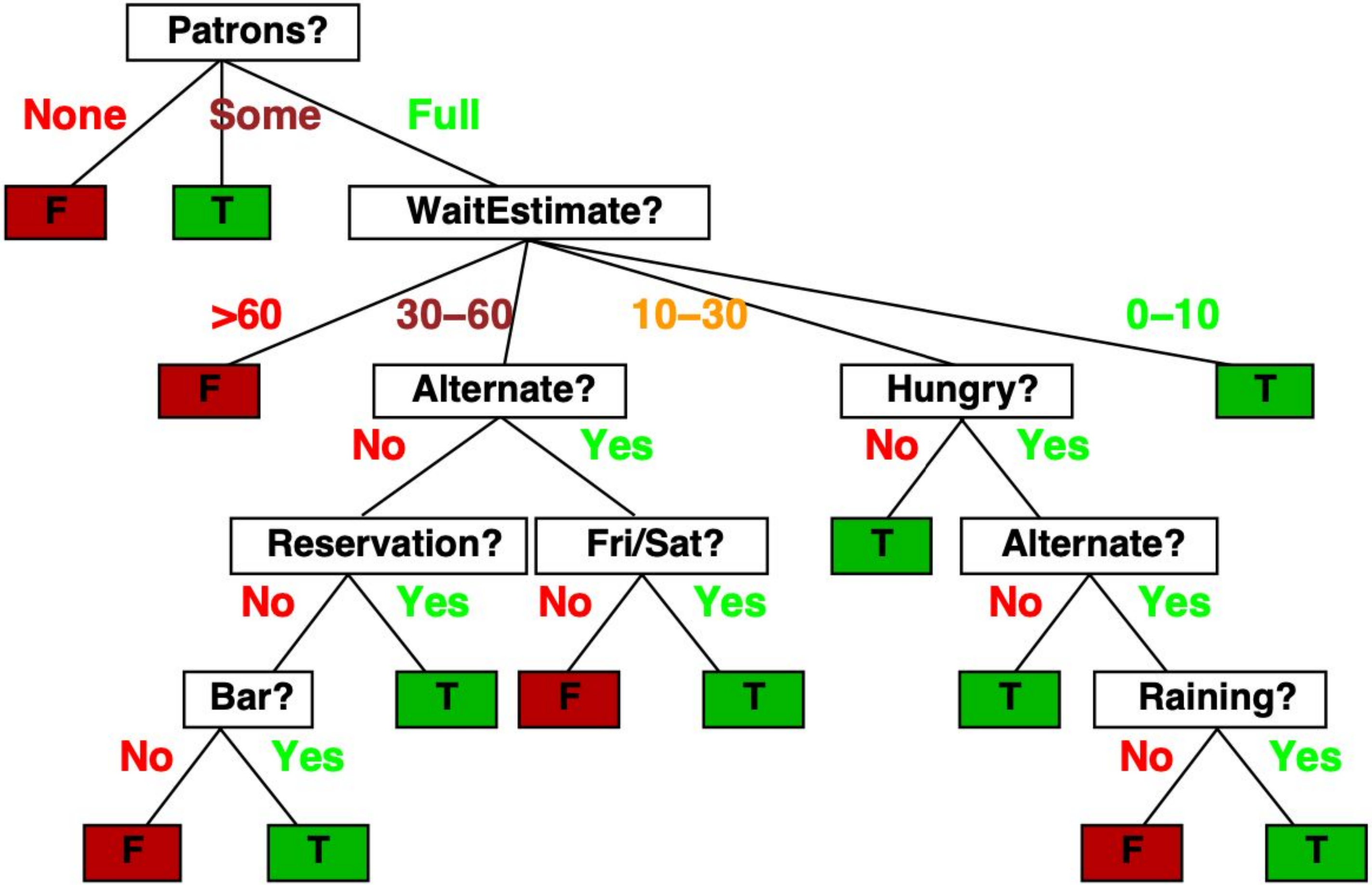
Build a decision tree to decide whether to wait for a table at a restaurant with the following attributes:

1. Alternate: whether there is a suitable alternative restaurant nearby.
2. Bar: whether the restaurant has a comfortable bar area to wait in.
3. Fri/Sat: true on Fridays and Saturdays.
4. Hungry: whether we are hungry.
5. Patrons: how many people are in the restaurant (values are None, Some, and Full).
6. Price: the restaurant's price range ($, $$, $$$).
7. Raining: whether it is raining outside.
8. Reservation: whether we made a reservation.
9. Type: the kind of restaurant (French, Italian, Thai, or burger).
10. WaitEstimate: the wait estimated by the host (0–10 minutes, 10–30, 30–60, or >60).

9.3 Attribute-based representations

Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
X_1	T	F	F	T	Some	$$$	F	T	French	0–10	T
X_2	T	F	F	T	Full	$	F	F	Thai	30–60	F
X_3	F	T	F	F	Some	$	F	F	Burger	0–10	T
X_4	T	F	T	T	Full	$	F	F	Thai	10–30	T
X_5	T	F	T	F	Full	$$$	F	T	French	>60	F
X_6	F	T	F	T	Some	$$	T	T	Italian	0–10	T
X_7	F	T	F	F	None	$	T	F	Burger	0–10	F
X_8	F	F	F	T	Some	$$	T	T	Thai	0–10	T
X_9	F	T	T	F	Full	$	T	F	Burger	>60	F
X_{10}	T	T	T	T	Full	$$$	F	T	Italian	10–30	F
X_{11}	F	F	F	F	None	$	F	F	Thai	0–10	F
X_{12}	T	T	T	T	Full	$	F	F	Burger	30–60	T

9.3 Inducing decision trees from examples



9.3 Decision tree learning

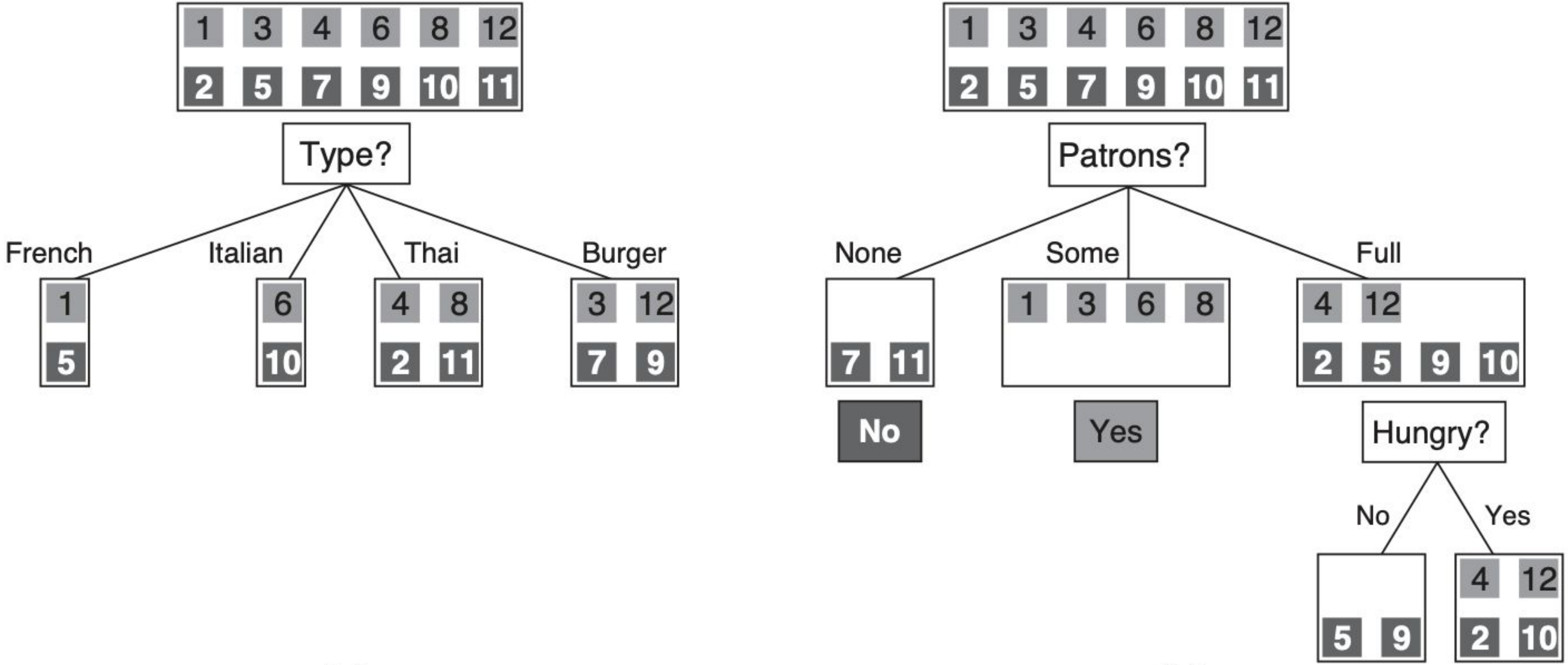
- Aim: find a small tree consistent with the training examples
- Idea: (recursively) choose "most significant" attribute as root of (sub)tree
- PLURALITY-VALUE: selects the most common output value among a set of examples.

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
  $A \leftarrow \operatorname{argmax}_{a \in attributes}$ IMPORTANCE(a, *examples*)
  tree ← 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 ← DECISION-TREE-LEARNING(*exs*, *attributes* − A, *examples*)
    add a branch to *tree* with label $(A = v_k)$ and subtree *subtree*
  return *tree*

9.3 Choosing attribute tests

- Idea: a good attribute splits the examples into subsets that are "all positive" or "all negative"
- Patrons? is a better choice—gives information about the classification



(a) (b)

9.3 Choosing attribute tests

- *The more clueless I am about the answer initially, the more information is contained in the answer*
- Scale: 1 bit = an answer to Boolean question with prior probability $\langle 0.5, 0.5\rangle$
- *Information in an answer* or *entropy of the prior* when prior probability is $\langle P_1, \ldots, P_n\rangle$:

$$H(\langle P_1, \ldots, P_n\rangle) = \Sigma_{i=1}^{n} - P_i \log_2 P_i$$

9.3 Choosing attribute tests

Suppose we have p positive and n negative examples at the root

⇒ bits needed to classify a new example

$$H(\langle p/(p+n), n/(p+n)\rangle)$$

An attribute splits the examples E into subsets E_i, each E_i has p_i positive and n_i negative examples

⇒ bits needed to classify a new example

$$H(\langle p_i/(p_i+n_i), n_i/(p_i+n_i)\rangle)$$

⇒ expected number of bits per example over all branches:

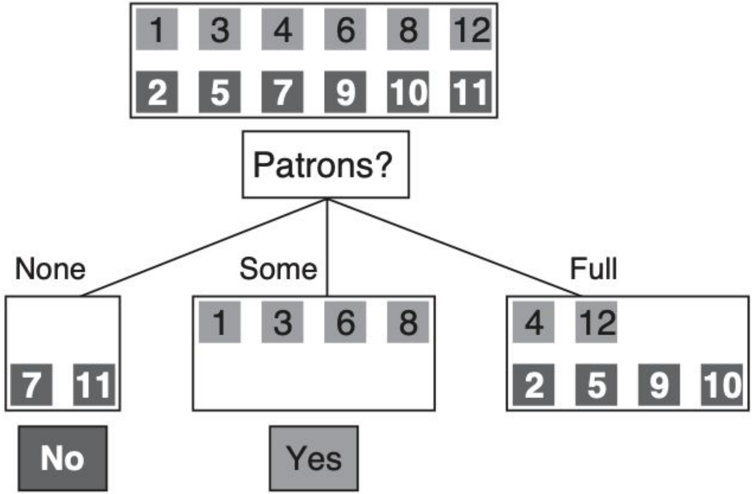
$$\Sigma_i \; \frac{p_i + n_i}{p + n} H(\langle p_i/(p_i + n_i), n_i/(p_i + n_i)\rangle)$$

9.3 Choosing attribute tests

For Patrons?, this is 0.459 bits, for Type this is (still) 1 bit

$\Rightarrow$ choose the attribute that minimizes the remaining information needed

$$\frac{2}{12}H(<0,1>) + \frac{4}{12}H(<1,0>) + \frac{6}{12}H(<\frac{2}{6},\frac{4}{6}>)$$
$$= -\frac{1}{2}(\frac{1}{3}log_2\frac{1}{3} + \frac{2}{3}log_2\frac{2}{3})$$
$$= 0.4591$$

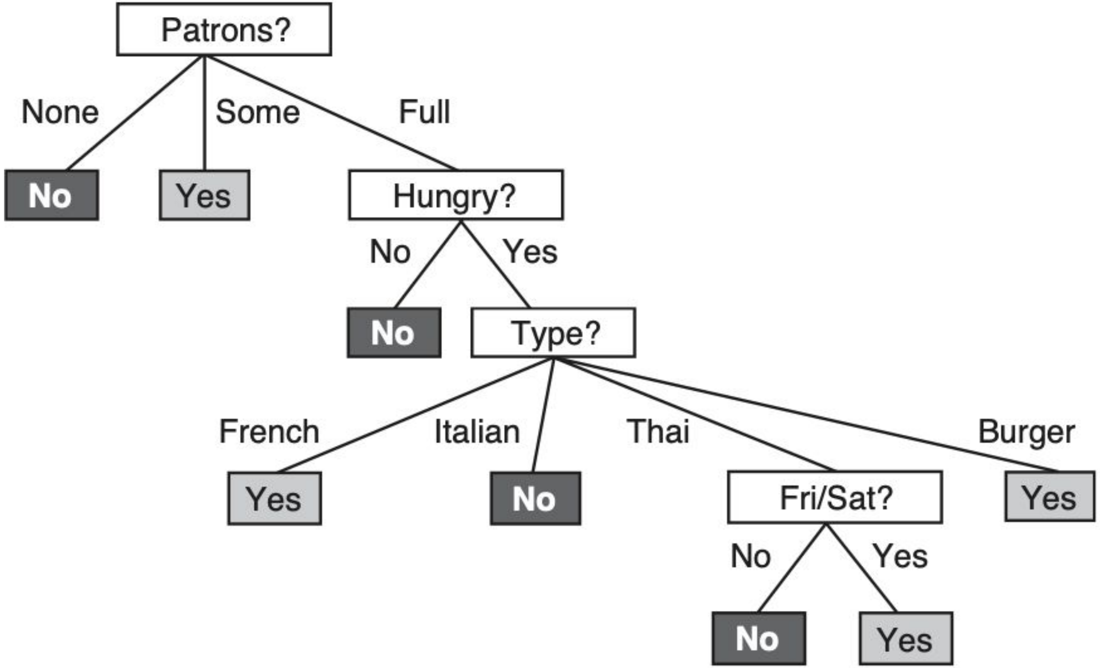


$$\Sigma_i \; \frac{p_i + n_i}{p+n} \; H(\langle p_i/(p_i+n_i), n_i/(p_i+n_i)\rangle)$$

(b)

9.3 Choosing attribute tests

- Decision tree learned from the 12 examples:

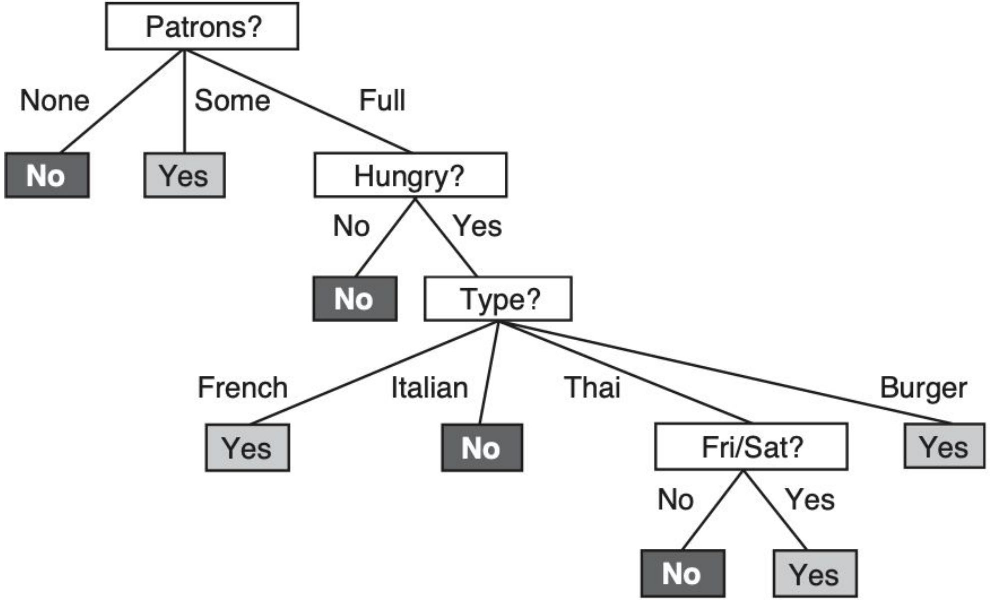


9.3 Choosing attribute tests

- Predict label of new example: X13

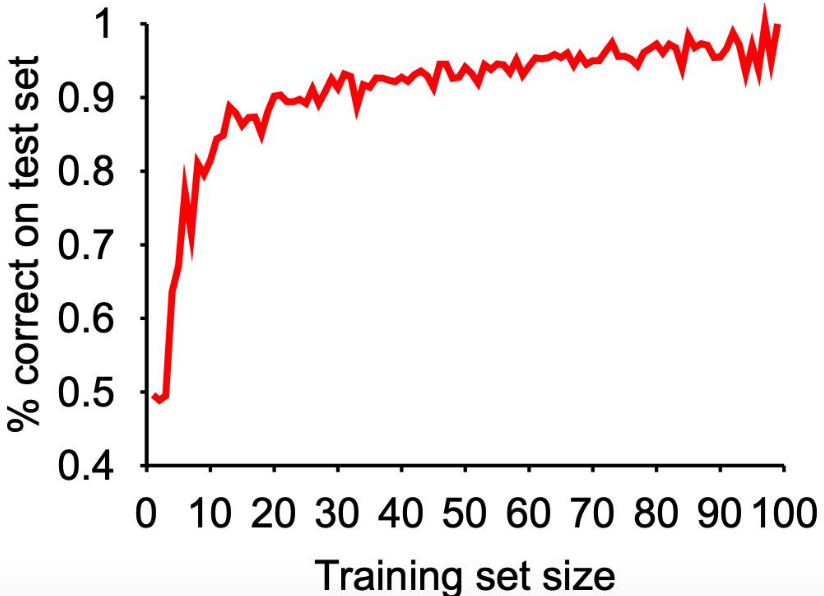
Attributes									
Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*

X13 = [T, T, F, F, Full, $$, F, F, Thai, 10-30]



Performance measurement

- How do we know that h ≈ f?
  - Try h on a new test set of examples
  - (use same distribution over example space as training set)
- **Learning curve** = % correct on test set as a function of training set size



9.3 Broadening the applicability of decision trees

- Overfitting: the model corresponds too closely or exactly to a particular set of data, => it fails to fit to additional data or predict future observations
    - For decision trees, a technique called decision tree pruning combats overfitting
- Missing data: not all the attribute values will be known for every example
- Multivalued attributes: an attribute has many possible values
- Continuous and integer-valued input attributes: find the split point that gives the highest information gain

9.4 Evaluating And Choosing The Best Hypothesis

9.4 Evaluating And Choosing The Best Hypothesis

- We want to learn a hypothesis that fits the future data best.
- "Best fit":
  - the **error rate** of a hypothesis:
    - the proportion of mistakes it makes
    - the proportion of times that $h(x) \neq y$ for an (x, y) example.
  - Lowest error rate on a new data

9.4 Evaluating And Choosing The Best Hypothesis

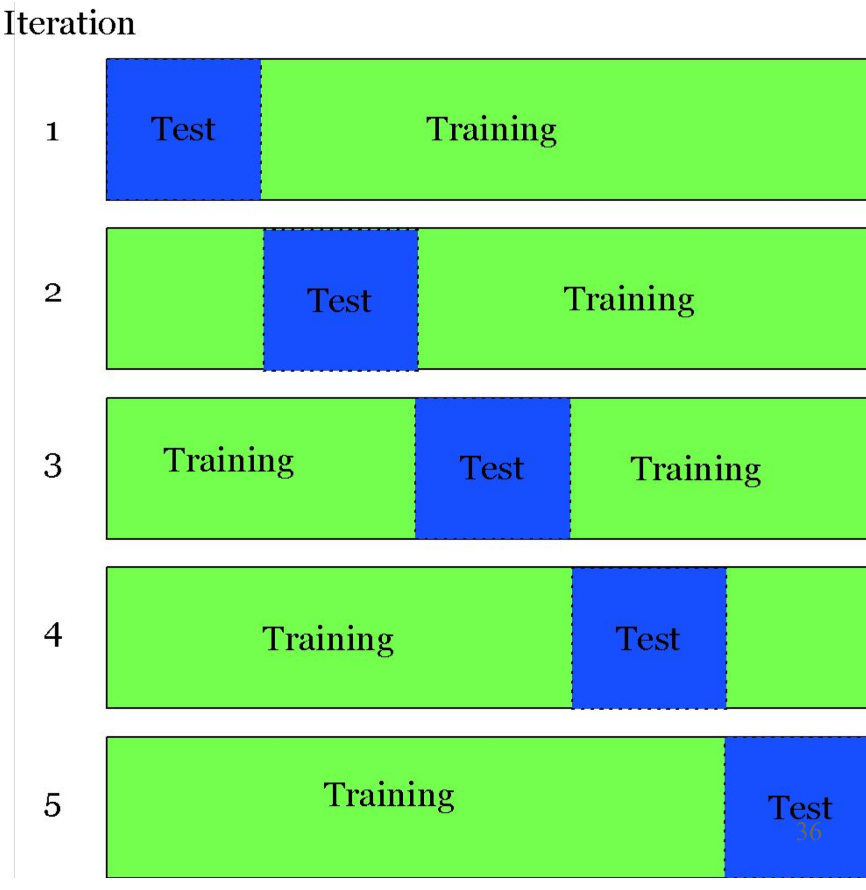
- Rather large number of examples: split examples into 3 sets:
  - training set: train h
  - validation set (development set): choose the best h
  - test set: evaluate h on the unseen data

9.4 Evaluating And Choosing The Best Hypothesis

- Small number of examples: k-fold cross-validation
    - split the data into k equal subsets
    - perform k rounds of learning
    - on each round:
        - $1/k$ of the data is held out as a test set,
        - the remaining examples are used as training data

9.4 Evaluating And Choosing The Best Hypothesis

- Break up data into 10 folds
  - Equal positive and negative inside each fold?
- For each fold
  - Choose the fold as a temporary test set
  - Train on 9 folds, compute performance on the test fold
- Report average performance of the 10 runs



9.4 Model selection

An algorithm to select the model that has the lowest error rate on validation data by

- building models of increasing complexity
- choosing the one with best empirical error rate on validation data.

9.4 From error rates to loss

- It is traditional to express utilities by a loss function.
- The loss function L(x, y, ŷ) is defined as the amount of utility lost by predicting h(x) = ŷ when the correct answer is f(x) = y:

$$
\begin{aligned}
L(x, y, \hat{y}) &= \mathit{Utility}(\text{result of using } y \text{ given an input } x) \\
&- \mathit{Utility}(\text{result of using } \hat{y} \text{ given an input } x)
\end{aligned}
$$

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

NONPARAMETRIC MODELS

NONPARAMETRIC MODELS

- A learning model that summarizes data with a set of parameters of fixed size (independent of the number of training examples) is called a **parametric model**.
- A **nonparametric model** is one that cannot be characterized by a bounded set of parameters.

Nearest neighbor models

- Given a query x_q , find the k examples that are nearest to x_q.
- $NN(k, x_q)$ to denote the set of k nearest neighbors.
- Classification:
  - find $NN(k, x_q)$
  - take the plurality vote of the neighbors
- Regression
  - find $NN(k, x_q)$
  - take the mean or median of the k neighbors
- 5-nearest-neighbors decision boundary is good; higher k would underfit.
- Cross-validation can be used to select the best value of k.

Nearest neighbor models

- Distance metric:

  Minkowski distance or L^p norm

$$L^p(\mathbf{x}_j, \mathbf{x}_q) = (\sum_i |x_{j,i} - x_{q,i}|^p)^{1/p}$$

With $p = 2$ this is Euclidean distance and with $p = 1$ it is Manhattan distance

$(|x_{11} - x_{21}|^2 + |x_{12} - x_{22}|^2)^{½}$

$(|x_{11} - x_{21}|^1 + |x_{12} - x_{22}|^1)^1$

d(A, B)

$((x_A - x_B)^2 + (y_A - y_B)^2)^{1/2}$

Nearest neighbor models

- Low-dimensional spaces with plenty of data, nearest neighbors works very well.
- High-dimensional spaces: the nearest neighbors are usually not very near!
- kNN algorithm:
  - Input: given a set of N examples and a query x_q,
  - Output: NN (k, x_q)
  - kNN algorithm:
    - iterate through the examples
      - measure the distance to xq from each one
    - keep the best k.

Nearest neighbor models

- Predict label of new example: X13

Attributes									
Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*

X13 = [T, T, F, F, Full, $$, F, F, Thai, 10-30]

Fri={F:0, T:1}; Hun = {F:0, T:1}; Pat = {0:Full, 1:None, 2:Some};

Type={0:Burger, 1:French, 2:Italy, 3:Thai}

Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
X_1	T	F	F	T	Some	$$$	F	T	French	0–10	T

$L^2(X_{13}, X_1) = (|0-0|^2 + |0-1|^2 + |0-2|^2 + |3-1|^2)^{1/2} = 3$

$L^2(X_{13}, X_2)$

Naive Bayes models

Naive Bayes models

- Naive Bayes is a probabilistic classifier
- Given a document d, out of all classes c ∈ C, the classifier returns the class ĉ which has the maximum posterior probability given the document.

$$\hat{c} = \underset{c \in C}{\operatorname{argmax}} P(c|d)$$

Naive Bayes models

$$\hat{c} = \underset{c \in C}{\operatorname{argmax}} P(c|d)$$

$$\hat{c} = \underset{c \in C}{\operatorname{argmax}} P(c|d) = \underset{c \in C}{\operatorname{argmax}} \frac{P(d|c)P(c)}{P(d)}$$

$$\hat{c} = \underset{c \in C}{\operatorname{argmax}} P(c|d) = \underset{c \in C}{\operatorname{argmax}} P(d|c)P(c)$$

Naive Bayes models

Without loss of generality, we can represent a document d as a set of features f_1, f_2,..., f_n:

$$\hat{c} = \underset{c \in C}{\operatorname{argmax}} P(c|d) = \underset{c \in C}{\operatorname{argmax}} P(d|c)P(c)$$

$$\hat{c} = \underset{c \in C}{\operatorname{argmax}} \overbrace{P(f_1, f_2, \ldots, f_n|c)}^{\text{likelihood}} \; \overbrace{P(c)}^{\text{prior}}$$

Naive Bayes models

- 1st assumption: position doesn't matter.
- 2nd assumption, Naive Bayes assumption: this is the conditional independence assumption that the probabilities $P(f_i|c)$ are independent given the class c

$$\hat{c} = \underset{c \in C}{\operatorname{argmax}} \overbrace{P(f_1, f_2, \ldots, f_n|c)}^{\text{likelihood}} \; \overbrace{P(c)}^{\text{prior}}$$

$$c_{NB} = \underset{c \in C}{\operatorname{argmax}} P(c) \prod_{f \in F} P(f|c)$$

Naive Bayes models

- Apply the naive Bayes classifier to text, each word in the documents as a feature

$$c_{NB} = \underset{c \in C}{\operatorname{argmax}} P(c) \prod_{f \in F} P(f|c)$$

$$c_{NB} = \underset{c \in C}{\operatorname{argmax}} P(c) \prod_{i \in positions} P(w_i|c)$$

Boolean Multinomial Naïve Bayes: Learning

- Calculate $P(c_j)$ terms
  - For each c_j in C do

    $docs_j \leftarrow$ all docs with class $=c_j$

$$P(c_j) \leftarrow \frac{|docs_j|}{|\text{total \# documents}|}$$

- From training corpus, extract *Vocabulary*

  Remove duplicates in each doc:

  For each word type w in doc_j

  Retain only a single instance of w

Boolean Multinomial Naïve Bayes: Learning

- Calculate $P(w_k \mid c_j)$ terms

  $Text_j \leftarrow$ single doc containing all $docs_j$

  For each word w_k in *Vocabulary*

  $n_k \leftarrow$ # of occurrences of w_k in $Text_j$

$$P(w_k \mid c_j) \leftarrow \frac{n_k + \alpha}{n + \alpha \mid Vocabulary \mid}$$

Boolean Multinomial Naïve Bayes: Testing

- First remove all duplicate words from *d*
- Then compute NB using the same equation:

$$c_{NB} = \underset{c_j \in C}{\operatorname{argmax}} P(c_j) \prod_{i \in positions} P(w_i \mid c_j)$$

Let's do a worked sentiment example!

	Cat	Documents
Training	-	just plain boring
	-	entirely predictable and lacks energy
	-	no surprises and very few laughs
	+	very powerful
	+	the most fun film of the summer
Test	?	predictable with no fun

An Example with add-1 smoothing

	Cat	Documents
Training	-	just plain boring
	-	entirely predictable and lacks energy
	-	no surprises and very few laughs
	+	very powerful
	+	the most fun film of the summer
Test	?	predictable ~~with~~ no fun

1. Prior from training:

$$\hat{P}(c_j) = \frac{N_{c_j}}{N_{total}}$$

P(-) = 3/5
P(+) = 2/5

2. Drop "with"

An Example with add-1 smoothing

	Cat	Documents
Training	-	just plain boring
	-	entirely predictable and lacks energy
	-	no surprises and very few laughs
	+	very powerful
	+	the most fun film of the summer
Test	?	predictable ~~with~~ no fun

3. Likelihoods from training:

$$p(w_i|c) = \frac{count(w_i, c) + 1}{(\sum_{w \in V} count(w, c)) + |V|}$$

$$P(\text{“predictable”}|-) = \frac{1+1}{14+20} \qquad P(\text{“predictable”}|+) = \frac{0+1}{9+20}$$

$$P(\text{“no”}|-) = \frac{1+1}{14+20} \qquad P(\text{“no”}|+) = \frac{0+1}{9+20}$$

$$P(\text{“fun”}|-) = \frac{0+1}{14+20} \qquad P(\text{“fun”}|+) = \frac{1+1}{9+20}$$

An Example with add-1 smoothing

	Cat	Documents
Training	-	just plain boring
	-	entirely predictable and lacks energy
	-	no surprises and very few laughs
	+	very powerful
	+	the most fun film of the summer
Test	?	predictable ~~with~~ no fun

4. Scoring the test set:

$$\boxed{P(-)P(S|-) = \frac{3}{5} \times \frac{2 \times 2 \times 1}{34^3} = 6.1 \times 10^{-5}}$$

$$P(+)P(S|+) = \frac{2}{5} \times \frac{1 \times 1 \times 2}{29^3} = 3.2 \times 10^{-5}$$

Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
X_1	T	F	F	T	Some	$$$	F	T	French	0–10	T
X_2	T	F	F	T	Full	$	F	F	Thai	30–60	F
X_3	F	T	F	F	Some	$	F	F	Burger	0–10	T
X_4	T	F	T	T	Full	$	F	F	Thai	10–30	T
X_5	T	F	T	F	Full	$$$	F	T	French	>60	F
X_6	F	T	F	T	Some	$$	T	T	Italian	0–10	T
X_7	F	T	F	F	None	$	T	F	Burger	0–10	F
X_8	F	F	F	T	Some	$$	T	T	Thai	0–10	T
X_9	F	T	T	F	Full	$	T	F	Burger	>60	F
X_{10}	T	T	T	T	Full	$$$	F	T	Italian	10–30	F
X_{11}	F	F	F	F	None	$	F	F	Thai	0–10	F
X_{12}	T	T	T	T	Full	$	F	F	Burger	30–60	T

X13 = [T, T, F, F, Full, $$, F, F, Thai, 10-30]