Chap. 18

Learning from Examples

Agents can improve their behavior
through diligent study of their own experiences.

1

Outline

- Learning Agents
- Inductive Learning
- Decision Tree Learning
- Measuring Learning Performance

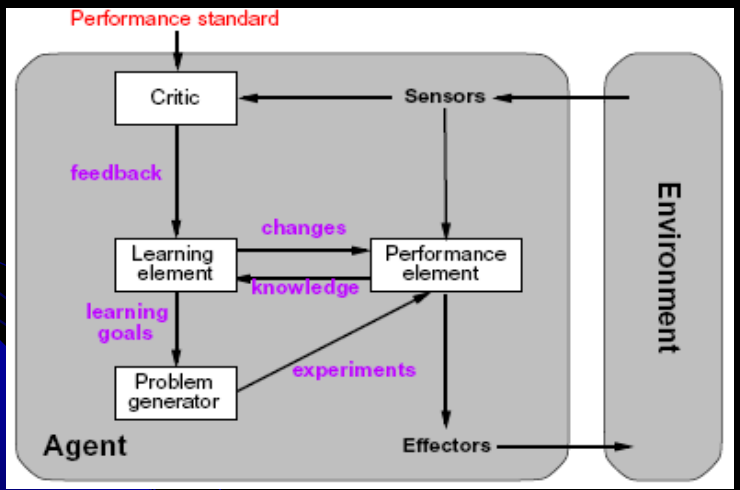
2

Learning

- Learning is essential for unknown environments,
  i.e., when designer lacks omniscience
- Learning is useful as a system construction method,
  i.e. expose the agent to reality rather than trying to write it down
- Learning modifies the agent's decision mechanisms
  to *improve performance* in the future.

3

Learning Agents



4

Learning Element

- Design of learning element for the improvement depends on:
  - what type of performance element is used
  - which functional component is to be learned
  - what prior knowledge the agent already has
  - how that component & data are represented
  - what kind of feedback is available
- Example scenarios

Performance element	Component	Representation	Feedback
Alpha–beta search	Eval. fn.	Weighted linear function	Win/loss
Logical agent	Transition model	Successor–state axioms	Outcome
Utility–based agent	Transition model	Dynamic Bayes net	Outcome
Simple reflex agent	Percept–action fn	Neural net	Correct action

5

… continued

- Design of learning element is dictated by
  - Type of performance element
  - Functional components to be learned
    1. Direct mapping from conditions on the current state to actions
    2. A means to infer relevant properties of the world from the percept sequence
    3. Information about the way the world evolves and about the results of possible actions the agent can take.
    4. Utility information indicating the desirability of the world states
    5. Action-value information indicating the desirability of actions
    6. Goals that describe classes of states whose achievement maximizes the agent's utility.

    Each of these components can be learned from appropriate feedback.

6

.. continued

- *Representation and Prior Knowledge*
  - Factored representation (a vector of attribute values) and outputs
  - Linear weighted polynomials for utility function
  - Propositional/first-order logical sentences,
  - probabilistic description: e.g. Bayesian Network
  - Types of learning: inductive learning, deductive learning.
    Generalization/Rules $\leftarrow/\rightarrow$ Specific Examples/Activities
- *Types of feedback*
  - **Supervised learning**: correct answers for each instance
    - Learning a function from examples of its inputs/outputs
  - **Reinforcement learning**: occasional rewards
    - Learning how the environment works
  - **Unsupervised learning:**
    - Learning patterns in the input with no specific output values
    - E.g.) clustering

7

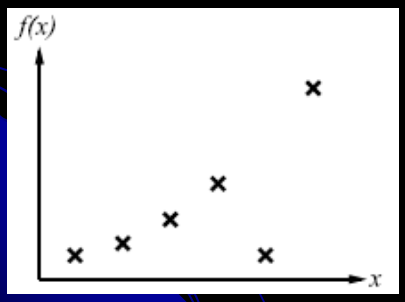
Inductive Learning in Supervised Learning

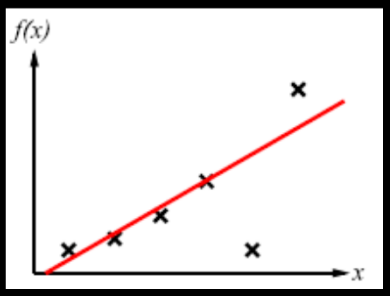
- Simplest form: learn a function from examples
- f is the target function s.t. $y = f(x)$
- An example is a pair $\langle x, f(x) \rangle$, e.g.
- Task of pure inductive inference (or induction):
  find a hypothesis h
  such that $h \approx f$, given a training set of examples
- The fundamental problem of Induction: how good $h \approx f$?
  A good hypothesis will generalize well; i.e. predict unseen examples correctly.
- Classification: when the output y is one of a finite set of values
- Regression: when y is a number, i.e. to find a conditional expectation of y.

8

Inductive Learning Method

- Construct/adjust *h* ($\in H$) to agree with *f* on training set
  (*h* is consistent if it agrees with *f* on all examples)
- Hypothesis space H: the set of hypotheses
- E.g.) curve fitting:





9

.. continued

- Construct/adjust *h* to agree with *f* on training set
  (*h* is consistent if it agrees with f on all examples)
- Hypothesis space H: the set of hypotheses
  - *How to choose from among multiple consistent hypotheses?*
- E.g.) curve fitting:

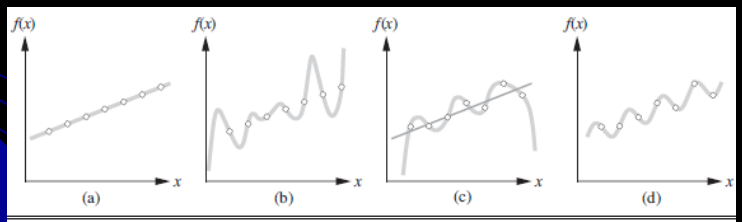
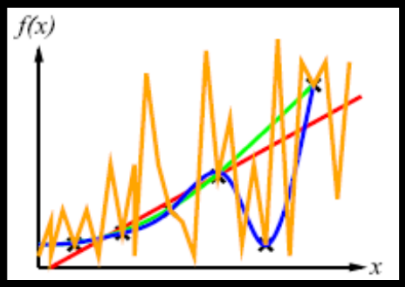


Figure 18.1 **FILES: figures/xy-plot.eps (Tue Nov 3 16:24:13 2009).** (a) Example $(x, f(x))$ pairs and a consistent, linear hypothesis. (b) A consistent, degree-7 polynomial hypothesis for the same data set. (c) A different data set, which admits an exact degree-6 polynomial fit or an approximate linear fit. (d) A simple, exact sinusoidal fit to the same data set.

10

.. continued

- Construct/adjust *h* to agree with *f* on training set (*h* is consistent if it agrees with f on all examples)
- E.g.) curve fitting:



- Hypothesis space H: the set of hypotheses
  - *How to choose from among multiple consistent hypotheses?*
  - **Ockham's razor**: maximize a combination of consistency and *simplicity*
- Tradeoff between the complex hypotheses that fit the data well and simpler hypotheses that may generalize better.

11

.. continued

- Possibility/Impossibility of finding a simple, consistent hypothesis depends on the *hypothesis space chosen*
  - E.g.) a polynomial function vs. a sinusoidal function
  - A learning is *realizable* if the hypothesis space contains the *true function*; otherwise, *unrealizable*.
    - To use prior knowledge to derive a hypothesis space in which we know the true function must lie.
    - To use the largest possible hypothesis space => *computational complexity of learning*
    - How probable a hypothesis is: Choose *h** that is most probable given the data.

$$h^* = \underset{h \in \mathcal{H}}{\operatorname{argmax}} P(h|data) = \underset{h \in \mathcal{H}}{\operatorname{argmax}} P(data|h)\, P(h)$$

  - A tradeoff b/t the expressiveness of a hypothesis space and the complexity of finding a simple, consistent hypothesis within that space.

12

Decision Tree Representations:
Attribute-based representation

- Examples described by the attribute values (Boolean, discrete, continuous, etc.)
  - Input: an object/situation described by a vector of attributes
  - Output: a decision – the predicted output value for the input
- E.g.) situations where I will/won't wait for a table:

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

- **Classification** of examples is positive (T) or negative (F)
  - Boolean classification

13

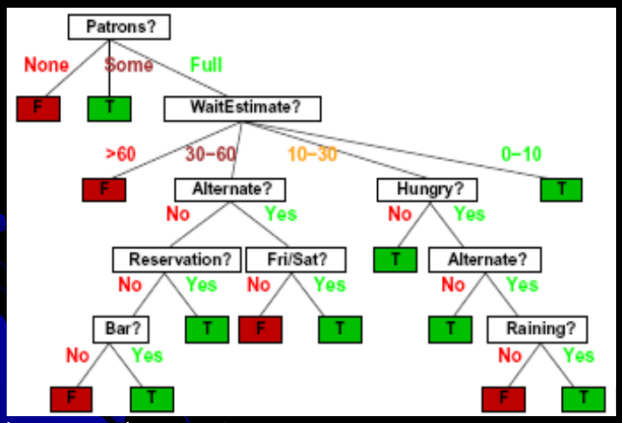
Decision Trees
as Learning Elements

- One possible representation for hypotheses in inductive learning
- *Attribute-based representation*
- Classification vs. regression:
  - discrete/continuous output value
  - Classification: learning a discrete-valued function
    - E.g.) Boolean classification (true/false, positive/negative)
  - Regression: learning a continuous function
- Decision by performing a sequence of tests:
  - an internal node: a test of the value of one of the properties
  - branches: labeled with the possible values of the test
  - Leaf: the output value

14

.. continued

- E.g.) a decision tree for deciding whether to wait.
  - Aim: To learn a definition for the *goal predicate* (*WillWait*).



  - Irrelevant attributes: price, type

15

Expressiveness of Decision Tree

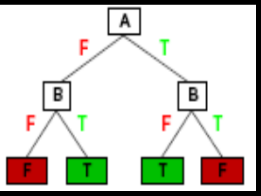
- Decision trees can express *any (boolean) function* of the input attributes: propositional language

  $Goal \Leftrightarrow (Path_1 \vee Path_2 \vee \ldots..)$

- E.g) for Boolean functions, truth table row $\rightarrow$ path to leaf:
  - $\forall s\ WillWait(s) \Leftrightarrow (P_1(s) \vee P_2(s) \vee \ldots \vee P_n(s))$

    s: object, *WillWait*: goal predicate, P_i: a path to a leaf with a positive output
  - XOR(A,B) $\leftrightarrow (\neg a \wedge b) \vee (a \wedge \neg b)$
- DT describes a relationship b/t a goal predicate and some logical combination of attribute values.

A	B	A xor B
F	F	F
F	T	T
T	F	T
T	T	F



- Prefer to find more compact decision trees

16

Hypothesis Spaces of Decision Tree

- How many distinct decision trees with *n* Boolean attributes??

  = number of Boolean functions

  = number of distinct truth tables with 2^n rows = 2^{2^n}

  (consider the answer column of the table as 2^n-bit number that defines the fn.)

- E.g.) with 6 Boolean attributes, there are 19,446,744,073,709,551,616 trees.
- When each attribute can be in (positive), in (negative), or out

  $\Rightarrow 3^n$ distinct conjunctive hypotheses
- More expressive hypothesis space
  - Increases chance that target function can be expressed (pros.)
  - Increases number of hypotheses consistent w/ training set

    $\Rightarrow$ may get worse predictions (cons.)

17

Decision Tree Learning

- Aim: Find the *smallest tree* consistent with the training examples
- Idea: (recursively) choose "most significant" attribute as root of (sub)tree

  -- the one that makes the most difference to the classification

function DECISION-TREE-LEARNING(*examples*, *attributes*, *parent_examples*) **returns** tree

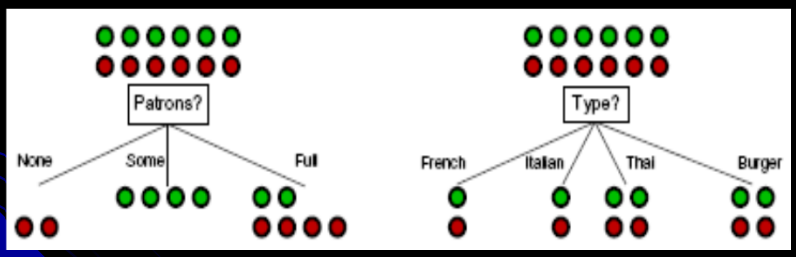
if *examples* is empty **then return** PLURALITY-VALUE(*parent_examples*)
else if all *examples* have the same classification **then return** the classification
else if *attributes* is empty **then return** PLURALITY-VALUE(*examples*)
else
  $A \leftarrow \text{argmax}_{a \in attributes}$ IMPORTANCE(a,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

- The basis of classification and regress tree (CART) algorithm, ID3 algorithm and its extension C4.5 algorithm.

18

Choosing an Attribute

- Idea: a good attribute splits the examples into subsets that are (ideally) "all positive" or "all negative"



- *Patrons?* Is a better choice – gives information about the classification
- A need of a formal measure:
  its maximum/minimum value when the attribute is perfect/of no use.

19

Information

- The need of a *formal measure* to evaluate the goodness of attribute.
  - The maximum value when the attribute is perfect.
  - The *expected amount of information* provided by the attribute
- Information answers questions
  - The information contained in the answer depends on one's prior knowledge.
  - The most clueless I am about the answer initially, the more information is contained in the answer
- *Entropy*: a measure of the uncertainty of a random variable; acquisition of information corresponds to a reduction in entropy. -- *impurity measure*
- Scale: Information theory measures information content in *bits*.
  - 1 bit = answer to Boolean question with prior <0.5, 0.5>
- Entropy of a random variable V with values v_k, each with prior $P(v_k)$ is

  $$H(V) = \sum_{k=1..n} P(v_k) \log_2 \frac{1}{P(v_k)} = -\sum_{k=1..n} P(v_k) \log_2 P(v_k)$$

  - *Shannon's* ***entropy*** of the prior

  – the average information (in bits) associated with the prediction of outcomes in a random experiment.

  e.g.) for the tossing of a fair coin, $H(<½, ½>) = -½\log_2 ½ - ½\log_2 ½ = 1$ bit.
- The correct classification is answered by a correct decision tree.

20

.. continued

- An estimate of the probabilities of the possible answers before any of the attributes have been tested is given by the proportions of positive/negative examples in the training set.
  - Suppose we have p positive and n negative examples at the root(=goal)
    $\Rightarrow H(goal) = B(p/(p+n))$ bits needed to classify a new example

    where $B(q)$ is defined as the entropy of a Boolean random variable that is true with prob. q.
  - E.g.) for 12 restaurant examples, $p=n=6$ so we need 1 bit

.. continued

- An attribute splits the examples E into subsets E_k, each of which (we hope) needs less information to complete the classification
  - Let E_k have p_k positive and n_k negative examples
    $\Rightarrow B(p_k / (p_k+n_k))$ bits needed to classify a new example
    $\Rightarrow$ expected number of bits per example over all branches is:

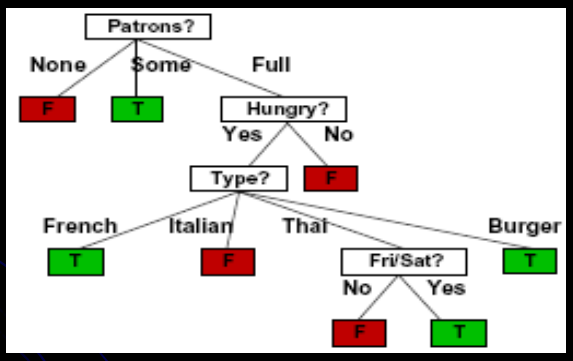
    Remainder(A) = $\sum_{k=1}^{d} \frac{p_k + n_k}{p+n} B(\frac{p_k}{(p_k+n_k)})$: the expected entropy remaining after testing attribute A.

    -- the need of bits of information to classify the example.
- **Information Gain:** $Gain(A) = B(p/(p+n)) - Remainder(A)$
  - Expected reduction in entropy
  - For *Patrons?*, remainder is 0,459 bits, for *Type* this is (still) 1 bit

$\Rightarrow$ choose the attribute that minimizes the remaining information (*Remainder*) needed
i.e. the attribute that maximizes the information gain (*Gain*).

Example

- Suppose tree learned from the 12 examples:



- Substantially simpler than "true" tree – a more complex hypothesis isn't justified by small amount of data

23

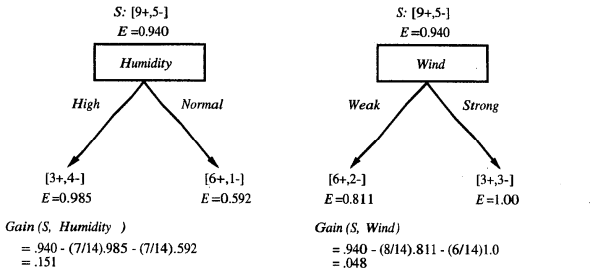
Example: Decision Tree Learning

Day	*Outlook*	*Temperature*	*Humidity*	*Wind*	*PlayTennis*
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

TABLE 3.2
Training examples for the target concept *PlayTennis*.

Example: Decision Tree Learning

Which attribute is the best classifier?



Gain (S, Humidity)
= .940 - (7/14).985 - (7/14).592
= .151

Gain (S, Wind)
= .940 - (8/14).811 - (6/14)1.0
= .048

FIGURE 3.3
Humidity provides greater information gain than *Wind*, relative to the target classification. Here, E stands for entropy and S for the original collection of examples. Given an initial collection S of 9 positive and 5 negative examples, $[9+, 5-]$, sorting these by their *Humidity* produces collections of $[3+, 4-]$ ($Humidity = High$) and $[6+, 1-]$ ($Humidity = Normal$). The information gained by this partitioning is .151, compared to a gain of only .048 for the attribute *Wind*.

Example: Decision Tree Learning

Which attribute is the best classifier at the root?

B(<9/14, 5/14>) = - 9/14 $\log_2$ 9/14 – 5/14 $\log_2$ 5/14 = 0.94

Gain (Outlook) *= B(<9/14, 5/14>)* – Remainder*(Outlook)*
= *0.94* – [5/14·B(<2/5, 3/5>) + 4/14·B(<4/4, 0/4>) + 5/14·B(<3/5, 2/5>]
= *0.246;*

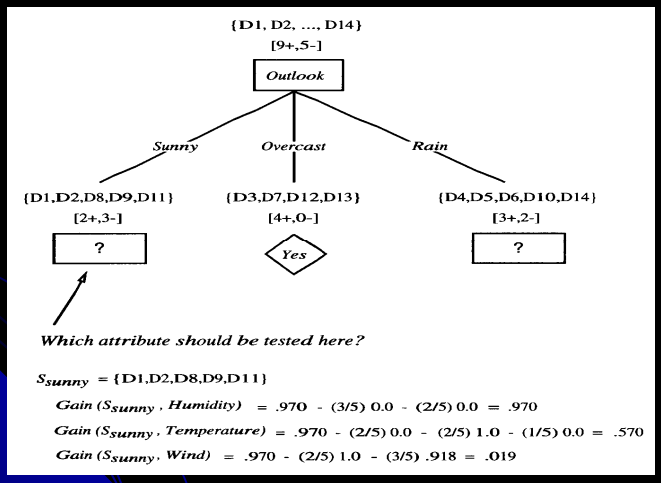
Gain (Humidity) *= 0.94* – Remainder*(Humidity)*
= *0.94* – [7/14·B(<3/7, 4/7>) + 7/14·B(<6/7, 1/7>)
= *0.151;*

Gain (Wind) *= 0.94* – Remainder*(Wind)*
= *0.94* – [8/14·B(<6/8, 2/8>) + 6/14·B(<3/6, 3/6>)]
= 0.94 - [8/14·(- 6/8 $\log_2$ 6/8 – 2/8 $\log_2$ 2/8)
+ 6/14·(-3/6 $\log_2$ 3/6 – 3/6 $\log_2$ 3/6)]
= 0.94 - [8/14 · 0.811 + 6/14 · 1]
= *0.048;*

Gain (Temperature) *= 0.0029*

Example: Decision Tree Learning

Which attribute is the best classifier at the subtree of <outlook=Sunny>?



Example: Decision Tree Learning

Which attribute is the best classifier at the subtree of <outlook=Sunny>?

$B(<2/5, 3/5>) = -2/5 \log_2 2/5 - 3/5 \log_2 3/5 = 0.97$

Gain (Humidity) = 0.97 – Remainder(*Humidity*)
= 0.97 – [3/5·B(<3/3, 0/3>) + 2/5·B(<2/2, 0/2>)] = 0.97;

Gain (Wind) = 0.97 – Remainder(*Wind*)
= 0.97 – [3/5·B(<1/3, 2/3>) + 2/5·B(<1/2, 1/2>)]
= 0.19;

Gain (Temperature) = 0.97 – Remainder(*Wind*)
= 0.97 – [2/5·B(<0/2, 2/2>) + 2/5·B(<1/2, 1/2>) + 1/5·B(<1/1, 0/1>)]
= 0.570

Which attribute is the best classifier at the subtree of <outlook=Rain>?

$B(<3/5, 2/5>) = -3/5 \log_2 3/5 - 2/5 \log_2 2/5 = 0.97$

Gain (Wind) = 0.97 – Remainder(*Wind*)
= 0.97 – [3/5·B(<3/3, 0/3>) + 2/5·B(<0/2, 2/2>)] = 0.97;

Gain (Temperature) = 0.97 – Remainder(*Wind*)
= 0.97 – [0/5·B(<0/2, 2/2>) + 3/5·B(<2/3, 1/3>) + 2/5·B(<1/2, 1/2>)]
= 0.19;

Example: Decision Tree Learning

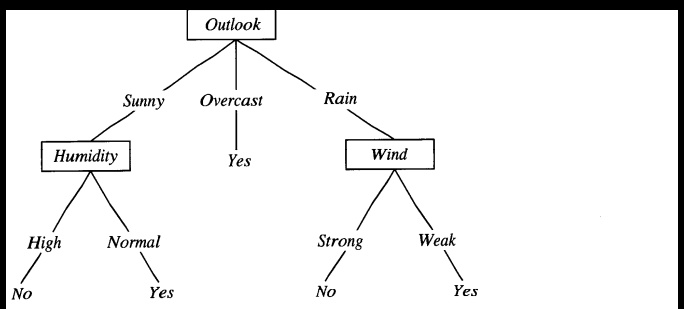


FIGURE 3.1
A decision tree for the concept *PlayTennis*. An example is classified by sorting it through the tree to the appropriate leaf node, then returning the classification associated with this leaf (in this case, *Yes* or *No*). This tree classifies Saturday mornings according to whether or not they are suitable for playing tennis.

***h* = (Outlook=Sunny ∧ Humidity=Normal) ∨(Outlook=Overcast) ∨ (Outlook=Rain ∧ Wind = Weak)**

Prediction:

For a test data <*Outlook=sunny, Temperature=cool, Humidity=high, Wind=strong*>,
PlayTennis? *PlayTennis = NO !*

Performance Measurement

- A learning algorithm is good if it produces hypotheses that do a good job of *predicting* the classifications of unseen examples.
- Assess the quality (accuracy) of a hypothesis by checking its predictions against the correct classification on the test data set.
  - Collect a large set of examples
  - Divide it 2 disjoint sets: the *training set* & *the test set*.
  - Apply the learning algorithm to the training set, generating a hypothesis h.
  - Measure the percentage of examples, (x, y), in the test set that are correctly classified by h.: $\frac{\#\ of\ test\ examples\ s.t. h(x) = y}{total\ \#\ of\ test\ examples} * 100$
  - Repeat the above steps for different sizes of training sets and different randomly selected sets of each size.
- k-fold Cross-Validation
- Result: a set of data that can be processed to give the average prediction quality as a *function* of the size of the training set.

30

Performance measurement: Error rate to Loss

- The quality of hypothesis may be measured by the error rate, but utility!
- In ML, utility is expressed by means of a loss function.
- Loss function, $L(x, y, \hat{y})$ is defined as the *amount of utility lost by predicting* $h(x) = \hat{y}$ when the correct answer is $f(x) = y$:

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

- E.g.) $L(spam,\ pam) = 1,\ L(pam, spam) = 10$: 10 times worse to classify non-spam as spam than vice versa.
- 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.
- Learning agent can maximize its expected utility by choosing the hypothesis that minimizes expected loss over all (x, y).

31

Cont…. Error rate to Loss

- Let *E* be the set of all possible (x, y) examples with a prior distribution P(X, Y).
- The Expected Generalization of Loss for a hypothesis h (w.r.t. loss fn L)

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

- The best hypothesis, h^*, is the one with the minimum expected generalization loss: $h^* = \underset{h\in H}{argmin}\ GenLoss_{L(h)}$.
- Since P(x, y) is unknown, the L-agent can only estimate gen. loss with empirical loss on *E*:
  - $EmpLoss_{L,E}(h) = \frac{1}{N}\sum_{(x,y)\in E} L(y, h(x))$
- The estimated best hypothesis $\hat{h}^* = \underset{h\in H}{argmin}\ EmpLoss_{L,E(h)}$.
- Why $\hat{h}^* \neq f$?
  - Unrealizability, variance, noise and computational complexity
- In Small-scale learning, the gen. error comes from the approximation error of $f \notin$ H, and from estimation error of not enough training examples.
- Large-scale learning with millions of example: gen. error by limits of computation. → sub-optimal approximation.

32

Performance measurement: Regularization

- Search for a hypothesis that directly minimizes the weighted sum of empirical loss and the complexity of the hypothesis, called *total cost*

  $$Cost(h) = EmpLoss(h) + \lambda \cdot Complexity(h).$$

  where λ (> 0) is a parameter that rates b/t loss and hypothesis complexity.
- The best hypothesis, $\hat{h}^* = \underset{h \in H}{argmin}\ Cost(h)$
- Still needs a cross-validation to find the hypothesis that generalizes best, but with different λ, rather than size. Then, select the best λ* with the best validation set score.
- This process of penalizing complex hypothesis is called *regularization*.

33

Theory of Learning

- How do we know that $h \approx f$? (Hume's Problem of Induction)
  1. Use theorems of *computational/statistical learning theory*
  2. Try h on a new **test set** of examples

     (use same distribution over example space as training set)
- How many examples do we need to get a good h?
- What hypothesis space should be used? , etc.
- Any hypothesis that is seriously wrong will almost certainly be found out with high probability after a small number of examples, because it'll make an incorrect prediction. Thus, any hypothesis that is consistent with a sufficiently large set of training examples is unlikely to be seriously wrong: i.e. it must be *p*robably *a*pproximately *c*orrect.
- PAC learning algorithm: any learning algorithm that returns hypotheses that are probably approximately correct.

34

Cont..

- The simplest PAC theorems deal with Boolean function with 0/1 loss.
- Error rate of a hypothesis h:

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

  -- the probability that h misclassifies a new example.
- A hypothesis h is called *approximately correct*

  if $\text{error}(h) < \varepsilon$, where ε is a small constant.
- Find N, the # examples s.t. all consistent hypotheses will be approximately correct after seeing N examples with high probability: $h \approx f$ inside ε-ball around f. Cf) $\mathcal{H}_{bad}$
- P($h_b \in \mathcal{H}_{bad}$ is consistent with N examples)?

  Since $\text{error}(h_b) > \varepsilon$, P($h_b$ agrees with a given example) $\leq 1 - \varepsilon$.
- The bound for N independent examples

  P(h_b agrees with N example) $\leq (1-\varepsilon)^N$
- P($\mathcal{H}_{bad}$ contains at least one consistent h) $\leq |\mathcal{H}_{bad}| \cdot (1-\varepsilon)^N \leq |\mathcal{H}|(1-\varepsilon)^N$ ➔ $\leq \delta$.
- Given $1-\varepsilon \leq e^{-\varepsilon}$, it's achievable if $N \geq \frac{1}{\varepsilon}(\ln\frac{1}{\delta} + \ln|\mathcal{H}|)$.
- Thus, if a learning algorithm returns a consistent h with N examples,

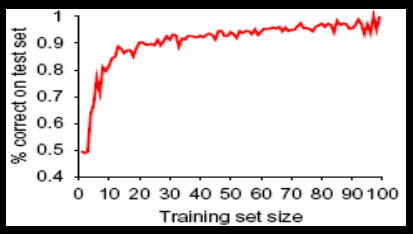
  $\text{error}(h) \leq \varepsilon$ with P(error)$\geq 1-\delta$. – h is probably approximately correct.

35

Performance measurement

- **Learning curve** = % correct on test as a function of training set size

  - the average prediction quality



- As the training set grows, the prediction quality increases.

36

Noise and Overfitting

- Noise
  - Examples may contain conflicting descriptions.
  - Naïve Solution:
    - majority classification if a deterministic hypothesis; or
    - the estimated probabilities of each classification using the relative frequencies
- Overfitting
  - Resulting decision tree may find meaningless regularity in the data
    though an exact hypothesis will be found as there are more attributes.
  - A general phenomenon for every kind of learning algorithm
  - Example: (Roll of die)
    Day : The day on which the die was rolled.
    Month : The month in which the die is rolled.
    Color : The color of the die.
  - We expect a decision tree with single leaf node for each roll with probabilities close to 1/6.

37

Techniques to treat Noise/Overfitting

- Decision Tree Pruning
  - Prevention of a recursive splitting on the (possibly) irrelevant attributes.
  - How to detect them? *Information Gain*(irr. attr.) ≈ 0.
- Null Hypothesis
  - How large IG should we require for splitting on a particular attribute?
  - A statistical Significance test with the assumption of null hypothesis:
    no underlying pattern.
  - Calculate the extent of deviation from a perfect absence of pattern: if the deg(deviation) is unlikely, good evidence for the presence of a significant pattern in the data.
- Chi-Square(χ^2) Pruning
  - Under the null hypothesis of irrelevance of attribute, the value of deviation is distributed according to χ^2 distribution.

38

Techniques to treat Noise/Overfitting

- Cross Validation
  - To reduce overfitting: it's applicable to any learning algorithm.
  - To estimate how well each hypothesis will predict unseen data.
  - Set aside some fraction of the known data and using it to test the prediction performance of a hypothesis.
  - *K*-fold cross-validation:
    run k experiments, each time setting aside a different 1/k of the data to test on, and average the result.

39

Broadening the Applicability of Decision Trees

- Missing Data
  - In many domains, not all the attribute values will be known for every example.
  - Given a complete DT, how to classify an object with missing test attributes?
  - How to modify the IG formula when some examples with unknown values for the attributes?
- Multivalued Attributes
  - When an attribute has a large number of possible values, the information gain gives inappropriate indications.
  - Singleton sets have highest information gain! - use gain ratio: Select attributes acc. To the ratio b/t their gain and their intrinsic information content, i.e. the amount of information contained in the answer to the question. – how efficiently an attribute provides information on the correct classification of an example?
- Continuous-valued input Attributes
  - Attributes such as height has large set of possible values.
  - Discretization with a split point which gives the highest IG is a solution for continuous attributes.

40

Summary

- Learning needed for unknown environments, lazy designers
- Learning agent = performance element + learning element
- Learning method depends on type of performance element, available feedback, type of component to be improved, and its representation
- For supervised learning, the aim is to find a simple hypothesis that is approximately consistent with training examples
- Decision tree learning using information gain
- Learning performance
  = prediction accuracy measured on test set

41

Regression and Classification with Linear Models

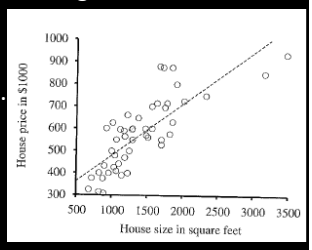
The class of linear functions of continuous-valued inputs

Regression with a Univariate Linear Function

- $y=w_1x + w_0$, where w_0, w_1 are real-valued coefficients/weights to be learned.

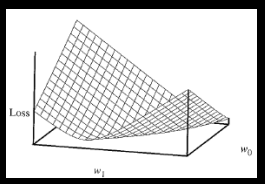
  $h_w(x)=w_1x + w_0$ $\mathbf{w}=[w_1, w_0]$

- Find the best h_w that fits the data: linear regression.
  → Find the values of weight $\mathbf{w}=[w_1, w_0]$ that minimize the empirical loss.
- Use the squared *Loss function*: no local minima.
- Find $\mathbf{w}^* = argmin_w Loss(h_w)$:



$$Loss(h_{\mathbf{w}}) = \sum_{j=1}^{N} L_2(y_j, h_{\mathbf{w}}(x_j)) = \sum_{j=1}^{N} (y_j - h_{\mathbf{w}}(x_j))^2 = \sum_{j=1}^{N} (y_j - (w_1 x_j + w_0))^2$$

$$\frac{\partial}{\partial w_0} \sum_{j=1}^{N} (y_j - (w_1 x_j + w_0))^2 = 0 \qquad \frac{\partial}{\partial w_1} \sum_{j=1}^{N} (y_j - (w_1 x_j + w_0))^2 = 0$$



$$\Leftrightarrow \quad w_1 = \frac{N(\sum x_j y_j) - (\sum x_j)(\sum y_j)}{N(\sum x_j^2) - (\sum x_j)^2}; \quad w_0 = (\sum y_j - w_1(\sum x_j))/N$$

In Non-linear models, a general optimization search problem in a continuous weight space.

Cont..

- In Non-linear models, a general optimization search problem in a continuous weight space. – e.g.) hill climbing algorithm.
- Use *gradient descent* to minimize the loss.

w ← any point in the parameter space
loop until convergence **do**
  for each w_i **in w do**
    $w_i \leftarrow w_i - \alpha \frac{\partial}{\partial w_i} Loss(\mathbf{w})$

· Move to a neighboring point that is downhill.

where α (step size) : the learning rate.

- For univariate regression with a quadratic Loss function → its partial derivative is a linear function.

$$\frac{\partial}{\partial w_0} Loss(\mathbf{w}) = -2(y - h_{\mathbf{w}}(x)); \qquad \frac{\partial}{\partial w_1} Loss(\mathbf{w}) = -2(y - h_{\mathbf{w}}(x)) \times x$$

- Then, $w_0 \leftarrow w_0 + \alpha (y - h_{\mathbf{w}}(x)); \quad w_1 \leftarrow w_1 + \alpha (y - h_{\mathbf{w}}(x)) \times x$
- For N training examples, $w_0 \leftarrow w_0 + \alpha \sum_j (y_j - h_{\mathbf{w}}(x_j)); \quad w_1 \leftarrow w_1 + \alpha \sum_j (y_j - h_{\mathbf{w}}(x_j)) \times x_j$
- The batch gradient descent learning rule for univariate linear regression.
- Convergence to the unique global minimum is guaranteed (as long as α is small enough), but very slow.
- Cf) Stochastic gradient descent in online setting: faster but no guarantee in convergence with a fixed $\alpha \Rightarrow$ schedule a decreasing α.

Multivariate Linear Regression

- $\mathbf{x}_j$ is an n-element vector.
- Hypothesis space is the set of functions s.t.

$$h_{sw}(\mathbf{x}_j) = w_0 + w_1 x_{j,1} + \cdots + w_n x_{j,n} = w_0 + \sum_i w_i x_{j,i} .$$

$$h_{sw}(\mathbf{x}_j) = \mathbf{w} \cdot \mathbf{x}_j = \mathbf{w}^\top \mathbf{x}_j = \sum_i w_i x_{j,i} .$$

- The best vector of weights, $\mathbf{w}^*$, minimizes square-error loss over the examples:

$$\mathbf{w}^* = \operatorname*{argmin}_{\mathbf{w}} \sum_j L_2(y_j, \mathbf{w} \cdot \mathbf{x}_j)$$

- Gradient descent will reach the (unique) minimum of the loss function with the update:

$$w_i \leftarrow w_i + \alpha \sum_j x_{j,i}(y_j - h_{\mathbf{w}}(\mathbf{x}_j))$$

- On the other hand, solve it for $\mathbf{w}$ that minimizes loss:

  $\mathbf{w}^* = (\mathbf{X}^\mathrm{T}\mathbf{X})^{-1}\mathbf{X}^\mathrm{T}\mathbf{y}$ where $\mathbf{y}$ is the vector of outputs

  and $\mathbf{X}$ is the data matrix (of n-dim. example per row.).

Cont..

- Problem of overfitting in high-dimensional space:

  Use *regularization* on multivariate linear function.
- Minimize the *total cost* of a hypothesis, counting both the empirical loss and the complexity of the hypothesis:

  $Cost(h) = EmpLoss(h) + \lambda \cdot Complexity(h)$ where

  $EmpLoss(h) = \frac{1}{N}\sum_{(x,y)\in E} L(y, h(x))$, $Complexity(h_w) = L_q(w) = \sum_i |w_i|^q$

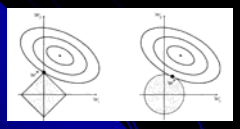
  Then, $\hat{h}^* = \operatorname*{argmin}_{h \in \mathcal{H}} Cost(h)$
- With loss function L,

  $q=1$: L_1 regularization which minimize the sum of absolute values

  -- it tends to produce a sparse model; i.e. many weights = 0

  $\Rightarrow$ the irrelevant attributes.

  $q=2$: L_2 regularization which minimizes the sum of squares.



# of examples required to find a good h is linear in the # of irrelevant features for L_2 regularization , but only logarithmic with L_1 regularization. – empirical evidence.

L_1 regularization vs. L_2 regularization

Linear classifiers with a hard threshold

- Linear function as a classifier
- Given training data, the task of classification is to learn a hypothesis *h* that will classify a new data.

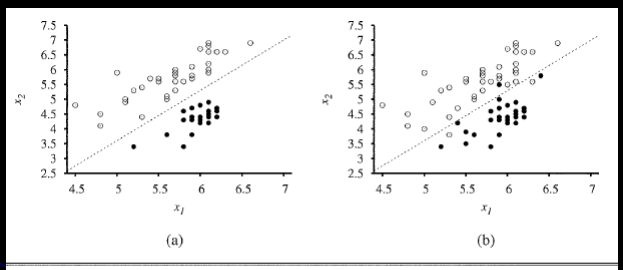


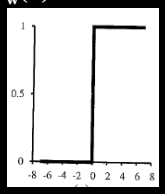
Figure 18.15 (a) Plot of two seismic data parameters, body wave magnitude x_1 and surface wave magnitude x_2, for earthquakes (white circles) and nuclear explosions (black circles) occurring between 1982 and 1990 in Asia and the Middle East (Kebeasy *et al.*, 1998). Also shown is a decision boundary between the classes. (b) The same domain with more data points. The earthquakes and explosions are no longer linearly separable.

- A decision boundary is a line (or a surface in higher dimension) that separates two classes: a linear separator vs. a linearly separable data: $x_2 = 1.7x_1 - 4.9$

Cont..

- The classification hypothesis: $h_{\mathbf{w}}(\mathbf{x}) = 1$ if $\mathbf{w}\cdot\mathbf{x} \geq 0$, 0 otherwise.
- Alternatively, think of *h* as the result of *threshold function* for $\mathbf{w}\cdot\mathbf{x}$:

  $h_{\mathbf{w}}(\mathbf{x}) = Threshold(\mathbf{w}\cdot\mathbf{x})$ where $Threshold(z) = 1$ if $z \geq 0$ and 0 otherwise.



So, choose the weights $\mathbf{w}$ to minimize the loss.

- *The perceptron learning rule: a weight update rule* that converges to a solution.

  $w_i \leftarrow w_i + \alpha \cdot (y - h_w(\mathbf{x})) \times x_i$
- 3 possibilities for 0/1 classification problem:
  - Correct output: $y = h_w(\mathbf{x})$: no changes in the weights.
  - Incorrect output:
    - $y = 1$, $h_w(\mathbf{x})=0$:

      w_i is increased/decreased when x_i is positive/negative, respectively.
    - $y = 0$, $h_w(\mathbf{x})=1$:

      w_i is decreased/increased when x_i is positive/negative, respectively.

A Training curve for perceptron learning rule,
measuring the classifier performance
on a fixed training set as the process proceeds on that same training set.

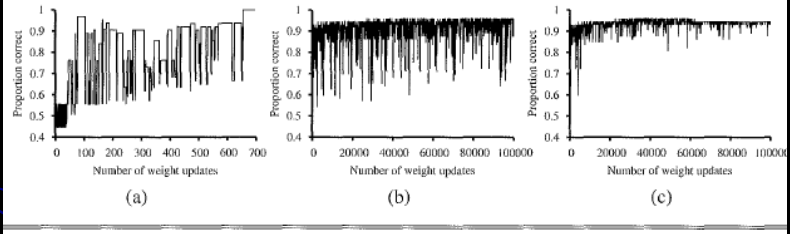
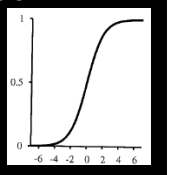


Figure 18.16 (a) Plot of total training-set accuracy vs. number of iterations through the training set for the perceptron learning rule, given the earthquake/explosion data in Figure 18.15(a). (b) The same plot for the noisy, non-separable data in Figure 18.15(b); note the change in scale of the x-axis. (c) The same plot as in (b), with a learning rate schedule $\alpha(t) = 1000/(1000 + t)$.

49

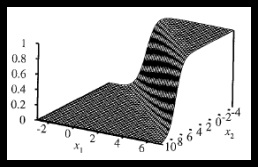
Linear classifiers with Logistic Regression

- If the hypothesis $h_{\mathbf{w}}(\mathbf{x})$ is not differentiable and is a discontinuous function of its input and its weights, learning with the perceptron rule is unpredictable.
- Some data are very close to the boundary
  -- the need of more gradated predictions.
- Use a soft threshold function: the logistic function $(Logistic(z) = \frac{1}{1+e^{-z}})$



$0 < h_{\mathbf{w}}(\mathbf{x}) = Logistic(\mathbf{w}\cdot\mathbf{x}) = \frac{1}{1+e^{-\mathbf{wx}}} < 1$

-- probability of belonging to the class labeled 1.



- Logistic regression:
  the process of fitting the weights of model with logistic function to minimize loss on a data set.
- Use the gradient descent weight update and L_2 loss function.

$$\begin{aligned}\frac{\partial}{\partial w_i} Loss(\mathbf{w}) &= \frac{\partial}{\partial w_i}(y - h_{\mathbf{w}}(\mathbf{x}))^2 \\ &= 2(y - h_{\mathbf{w}}(\mathbf{x})) \times \frac{\partial}{\partial w_i}(y - h_{\mathbf{w}}(\mathbf{x})) \\ &= -2(y - h_{\mathbf{w}}(\mathbf{x})) \times g'(\mathbf{w}\cdot\mathbf{x}) \times \frac{\partial}{\partial w_i}\mathbf{w}\cdot\mathbf{x} \\ &= -2(y - h_{\mathbf{w}}(\mathbf{x})) \times g'(\mathbf{w}\cdot\mathbf{x}) \times x_i\,.\end{aligned}$$

$g'(\mathbf{w}\cdot\mathbf{x}) = g(\mathbf{w}\cdot\mathbf{x})(1-g(\mathbf{w}\cdot\mathbf{x})) = h_{\mathbf{w}}(\mathbf{x})(1-h_{\mathbf{w}}(\mathbf{x}))$
where g is the logistic function with g'.

⇓

weight update:

$w_i \leftarrow w_i + \alpha(y - h_{\mathbf{w}}(\mathbf{x})) \times h_{\mathbf{w}}(\mathbf{x})(1-h_{\mathbf{w}}(\mathbf{x})) \times x_i$

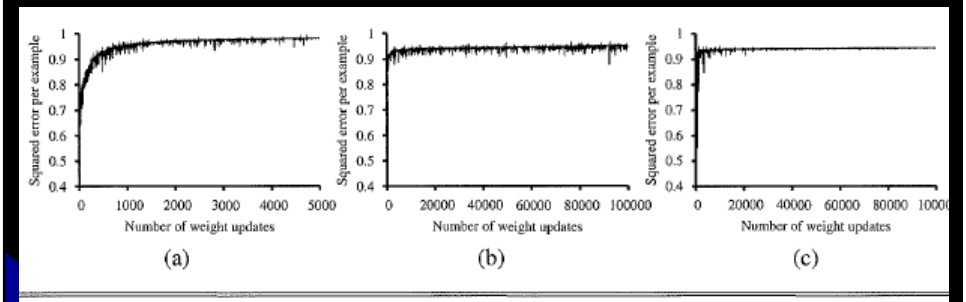


Figure 18.18 Repeat of the experiments in Figure 18.16 using logistic regression and squared error. The plot in (a) covers 5000 iterations rather than 1000, while (b) and (c) use the same scale.

51