

CmpE 540

Principles of Artificial Intelligence

Pinar Yolum
pinar.yolum@boun.edu.tr

Department of
Computer Engineering
Boğaziçi University

1

Machine Learning

Chapter 18
(Based mostly on the course slides from
<http://aima.cs.berkeley.edu/> and
<http://www.cmpe.boun.edu.tr/~akin/>)

2

What is Learning?

- "Learning denotes changes in a system that ... enable a system to do the same task more efficiently the next time." --Herbert Simon
- "Learning is constructing or modifying representations of what is being experienced." --Ryszard Michalski
- "Learning is making useful changes in our minds." --Marvin Minsky

3

What is Machine Learning?

There are two ways that a system can improve:

1. By acquiring new knowledge
For example: acquiring new facts or skills
2. By adapting its behavior
For example: solving problems more accurately or efficiently

4

Machine Learning

- Why is learning necessary?
 - learning is the hallmark of intelligence; a system that cannot learn is arguably not intelligent.
 - without learning, everything is new; a system that cannot learn is not efficient because it rederives each solution and repeatedly makes the same mistakes.
 - Can be used to understand and improve efficiency of human learning. For example, use to improve methods for teaching and tutoring people, as done in CAI -- Computer-aided instruction
 - Discover new things or structure that is unknown to humans. Example: Data mining
 - Fill in skeletal or incomplete specifications about a domain. Large, complex AI systems cannot be completely derived by hand and require dynamic updating to incorporate new information. Learning new characteristics expands the domain or expertise and lessens the "brittleness" of the system
- Why is learning possible?
 - because there are regularities in the world.

5

Learning

Definition: A computer program is said to **learn** from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience.

6

Major Paradigms of Machine Learning

- **Rote Learning:**
 - One-to-one mapping from inputs to stored representation.
 - "Learning by memorization."
 - Association-based storage and retrieval.
- **Learning by being told.**
- **Induction:**
 - Use specific examples to reach general conclusions
- **Reinforcement:**
 - Only feedback (positive or negative reward) given at end of a sequence of steps.
 - Requires assigning reward to steps by solving the credit assignment problem--which steps should receive credit or blame for a final result?
- **Clustering:**
 - Analogy.
 - Determine correspondence between two different representations
- **Discovery:**
 - Unsupervised, specific goal not given

7

Examples of Machine Learning

- **Learn to recognize speech:**
 - **Task:** recognizing and classifying words in a speech signal
 - **Performance:** average number of words correctly identified
 - **Experience:** speech signal segmented and labeled by word
- **Learn to recognize faces:**
 - **Task:** recognizing and classifying bit-map images of faces
 - **Performance:** percentage of faces correctly classified
 - **Experience:** faces and their classification (e. g., name of person)
- **Learning to drive:**
 - **Task:** driving on four-lane highways using vision sensors
 - **Performance:** average distance traveled before error
 - **Experience:** sequence of images and steering commands.

8

Brief History of Machine Learning

- 1950's: Samuels checker player
- 1960's: Neural networks, perceptron; pattern recognition; learning in the limit theory; Minsky & Papert.
- 1970's: Symbolic concept induction; Winston's arch learner; knowledge acquisition bottleneck; Quinlan's ID3; Michalski's AQ and soybean diagnosis results; Scientific discovery with BACON; mathematical discovery with AM.
- 1980's: Continued progress on decision-tree and rule learning; Explanation-based learning; speedup learning; utility problem, analogy; resurgence of connectionism (PDP, ANN); Valiant's PAC learning theory; experimental evaluation.
- 1990's: Data mining; adaptive software agents; reinforcement learning; theory refinement; inductive logic programming; voting, bagging, boosting, and stacking; learning Bayesian networks.

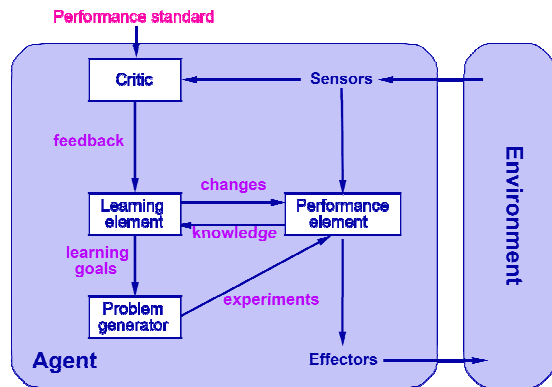
9

Applications of ML

- Learning to recognize spoken words
 - SPHINX (Lee 1989)
- Learning to drive an autonomous vehicle
 - ALVINN (Pomerleau 1989)
- Learning to classify celestial objects
 - (Fayyad et al 1995)
- Learning to play world-class backgammon
 - TD-GAMMON (Tesauro 1992)
- Designing the morphology and control structure of electro-mechanical artefacts
 - GOLEM (Lipton, Pollock 2000)

10

Components of a Learning System



11

Learning element

- Design of learning element is dictated by
 - what type of performance element is used
 - which functional component is to be learned
 - how that functional component is 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

Supervised learning: correct answers for each instance
 Reinforcement learning: occasional rewards

12

Evaluating Performance

- Several possible criteria for evaluating a learning algorithm:
 - Predictive accuracy of classifier
 - Speed of learner
 - Speed of classifier
 - Space requirements
 - Most common criterion is **predictive accuracy**

13

Input to an ML System

- Most ML algorithms use a training set of examples as the basis for learning.
- Each example is encoded using the instance representation chosen for the problem. The representation is important!
- **Supervised Learning**: the learning program is given training examples with the correct classification for each example.
- **Unsupervised Learning**: the learning program is given training examples without classifications

14

Training Examples' Representation

- Examples presented to a machine learning algorithm are typically represented as attribute-value pairs.
- An *attribute* is a general property associated with an object.
 - For example, we might describe animals with the attributes: size, color, temp, has tail, has beak, covering
- A *value* is one possible instantiation of the attribute.
- A CANARY might be represented as:
 - size= small, color= yellow, temp= warm blooded, has tail= true, has beak= true, covering= feathers
- A TERRIER might be represented as:
 - size= medium, color= brown, temp= warm blooded, has tail= true, has beak= false, covering= fur

15

Learning Problem Examples

- Credit card applications
 - Task T: Distinguish "good" applicants from "risky" applicants.
 - Performance measure P : ?
 - Experience E : ? (direct/indirect)
 - Target function : ?

16

Performance Measure P:

- Error based: minimize percentage of incorrectly classified customers : $P = (N_{fp} + N_{fn}) / N$

N_{fp} : # false positives (rejected good customers)

N_{fn} : # false negatives (accepted bad customers)

- Utility based: maximize expected profit of credit card business: $P = N_{cp} * U_{cp} + N_{fn} * U_{fn}$

U_{cp} : expected utility of an accepted good customer

U_{fn} : expected utility/loss of an accepted bad customer

17

Experience E:

- Direct: Decisions on credit card applications made by a human financial expert

Training data: <customer inf., reject/accept>

- Direct: Actual customer behavior based on previously accepted customers

Training data: <customer inf., good/bad>

Problem: Distribution of applicants $P_{\text{applicant}}$ is not identical with training data P_{train}

- Indirect: Evaluate a decision policy based on the profit you made over the past N years.

18

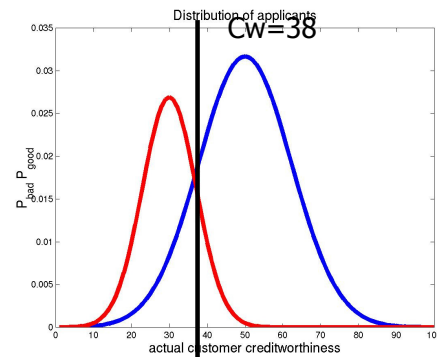
Distribution of Applicants

Good customers

Bad customers

Assume we want to minimize classification error:

What is the optimal decision boundary?



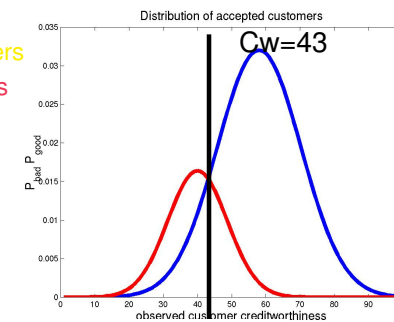
19

Distribution of Accepted Customers

Good customers

Bad customers

What is the optimal decision boundary?



20

Target Function

- Customer record:
 - ↗ income, owns house, credit history, age, employed, accept
 - ↗ \$40000, yes, good, 38, full-time, yes
 - ↗ \$25000, no, excellent, 25, part-time, no
 - ↗ \$50000, no, poor, 55, unemployed, no
- T: Customer data → accept/reject
- T: Customer data → probability good customer
- T: Customer data → expected utility/profit

21

Learning methods

- Decision rules:
 - ↗ If income < \$30.000 then reject
- Bayesian network:
 - ↗ $P(\text{good} \mid \text{income, credit history, ...})$
- Neural Network:
- Nearest Neighbor:
 - ↗ Take the same decision as for the customer in the data base that is most similar to the applicant

22

Learning Problem Examples

- Obstacle Avoidance Behavior of a Mobile Robot
 - ↗ Task T: Navigate robot safely through an environment.
 - ↗ Performance measure P : ?
 - ↗ Experience E : ?
 - ↗ Target function : ?

23

Performance Measure P:

- P: Maximize time until collision with obstacle
- P: Maximize distance traveled until collision with obstacle
- P: Minimize rotational velocity, maximize translational velocity
- P: Minimize error between control action of a human operator and robot controller in the same situation

24

Training Experience E:

- **Direct:** Monitor human operator and use her control actions as training data:
 - $E = \{ \langle \text{perception}_i, \text{action}_i \rangle \}$
- **Indirect:** Operate robot in the real world or in a simulation. Reward desirable states, penalize undesirable states
 - $V(b) = +1$ if $v > 0.5$ m/s
 - $V(b) = +2$ if $\omega < 10$ deg/s
 - $V(b) = -100$ if bumper state = 1
 - Question: Internal or external reward ?

25

Target Function

- Choose action:
 - A: perception \rightarrow action
 - Sonar readings: $s_1(t) \dots s_n(t) \rightarrow \langle v, \omega \rangle$
- Evaluate perception/state:
 - $V: s_1(t) \dots s_n(t) \rightarrow V(s_1(t) \dots s_n(t))$
 - Problem: states are only partially observable therefore world seems non-deterministic
 - Markov Decision Process : successor state $s(t+1)$ is a probabilistic function of current state $s(t)$ and action $a(t)$
- Evaluate state/action pairs:
 - $V: s_1(t) \dots s_n(t), a(t) \rightarrow V(s_1(t) \dots s_n(t), a(t))$

26

Concept Learning

Task: classification

Goal: to learn a procedure for assigning objects to predefined categories. The categories are usually mutually exclusive.

Example classification tasks:

1. Given a set of symptoms for a patient, classify the set of symptoms according to possible diseases.
2. Given a text, classify it according to its topic (e. g., sports, crime, economics, entertainment).
3. Given the weather conditions today, predict the weather tomorrow (e. g., sunny, partly cloudy, rain, snow).

27

Concept Learning Terminology

- An **instance** is a description of a specific item. X is the space of all instances (instance space).
- The **target concept**, $c(x)$, is a binary function over instances.
- A **training example** is an instance labeled with its correct value for $c(x)$ (positive or negative). D is the set of all training examples.
- The **hypothesis space**, H , is the set of functions, $h(x)$, that the learner can consider as possible definitions of $c(x)$.
- The goal of concept learning is to find an h in H such that for all $\langle x, c(x) \rangle$ in D , $h(x) = c(x)$.

28

Sample Hypothesis Space

- Consider a hypothesis language defined by a conjunction of constraints.
- For instances described by n features consider a vector of n constraints, $\langle c_1, c_2, \dots, c_n \rangle$ where each c_i is either:
 - $?$, indicating that any value is possible for the i th feature
 - A specific value from the domain of the i th feature
 - \emptyset , indicating no value is acceptable
- Sample hypotheses in this language:
 - $\langle \text{big, red, ?} \rangle$
 - $\langle ?, ?, ? \rangle$ (most general hypothesis)
 - $\langle \emptyset, \emptyset, \emptyset \rangle$ (most specific hypothesis)

29

Inductive Learning Hypothesis

- “Any hypothesis that is found to approximate the target function well over a sufficiently large set of training examples will also approximate the target function well over other unobserved examples.”
- Assumes that the training and test examples are drawn from the same general distribution.
- This is fundamentally an unprovable hypothesis unless additional assumptions are made about the target concept.

30

Concept Learning as Search

- Concept learning can be viewed as searching the space of hypotheses for one (or more) consistent with the training instances.
- Consider an instance space consisting of n binary features, which therefore has 2^n instances.
- For conjunctive hypotheses, there are 4 choices for each feature: T, F, \emptyset , $?$, so there are 4^n syntactically distinct hypotheses, but any hypothesis with a \emptyset is the empty hypothesis, so there are $3^n + 1$ semantically distinct hypotheses.
- The target concept could in principle be any of the 2^{2^n} (2 to the 2^n) possible binary functions on n binary inputs.
- Frequently, the hypothesis space is very large or even infinite and intractable to search exhaustively.

31

Learning by Enumeration

- For any finite or countably infinite hypothesis space, one can simply enumerate and test hypotheses one by one until one is found that is consistent with the training data.
- This algorithm is guaranteed to terminate with a consistent hypothesis if there is one; however it is obviously intractable for most practical hypothesis spaces, which are at least exponentially large.

32

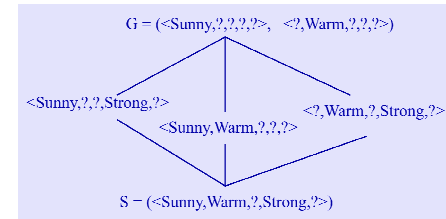
Version Spaces

- General idea:
 - - Organize hypotheses in a lattice based on generality
 - - Maintain sets of most general & specific hypotheses
- Definition:
 - A version space $VS(H,E)$ with respect to hypothesis space H and training examples E is the subset of H that are consistent with E (i.e., if H classifies all examples in E correctly). It is represented by its general boundary G (its set of most general members) and its specific boundary S (its set of most specific members).
- Strength
 - Training algorithm converges to correct target concept, assuming error-free training set and target concept is in H . Otherwise, collapses to empty set.
- Limitation:
 - S and G grow exponentially for some hypothesis languages H (i.e., disjunctive concepts). Brittle.

33

Version Spaces Example

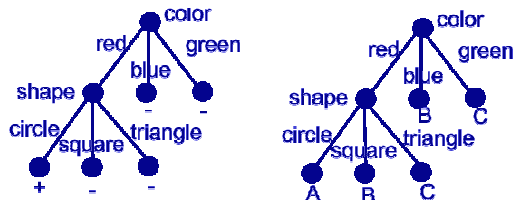
Class	Sky	Air	Humidity	Wind	Water
+	Sunny	Warm	Normal	Strong	Warm
+	Sunny	Warm	High	Strong	Warm
-	Rainy	Cold	High	Strong	Warm
+	Sunny	Warm	High	String	Cool



34

Decision Trees

- Decision trees are classifiers for instances represented as feature vectors.
- Nodes test features, there is one branch for each value of the feature, and leaves specify categories.



35

Properties of Decision Trees

- Can represent arbitrary disjunction and conjunction and therefore can represent any discrete function on discrete features.
- Can categorize instances into multiple disjoint categories.
- Can be rewritten as rules in disjunctive normal form (DNF)

36

Decision Tree Learning

- Instances are represented as attribute-value pairs.
- Discrete values are simplest, thresholds on numerical features are also possible for splitting nodes.
- Output is a discrete category. Real valued outputs are possible with additions (regression trees).
- Algorithms are efficient for processing large amounts of data.
- Methods are available for handling noisy data (category and attribute noise).
- Methods are available for handling missing attribute values.

37

Basic Decision Tree Algorithm

- Recursively build a decision tree top-down through batch processing of the training data.

```
DTree(examples, attributes)
If all examples are in one category, return a leaf node with this
category as a label.
Else if attributes are empty then return a leaf node labeled with
the category which is the most common in examples.
Else Pick an attribute, A, for the root.
For each possible value  $v_i$  for A
Let  $examples_i$  be the subset of examples that have value  $v_i$  for A.
Add a branch out of the root for the test  $A=v_i$  .
If  $examples_i$  is empty
Then create a leaf node labeled with the category which is most
common in examples
Else recursively create a subtree by calling DTree( $examples_i$ ,
attributes - {A})
```

38

Picking the Root Attribute

- Goal is to have the resulting decision tree be as small as possible, following Occam's Razor.
- Finding a minimal decision tree consistent with a set of data is NP-hard.
- Simple recursive algorithm does a greedy heuristic search for a fairly simple tree but cannot guarantee optimality.
- Want a test which creates subsets which are relatively "pure" in one class so that they are closer to being leaf nodes.
- There are various heuristics for picking a good test, the most popular one based on information gain originated with ID3 system of Quinlan (1979).

39

Entropy

- Entropy (impurity, disorder) of a set of examples, S, relative to a binary classification is:
$$\text{Entropy}(S) = -p^+ \log_2(p^+) - p^- \log_2(p^-)$$
- where p^+ is the proportion of positive examples in S and p^- the proportion of negatives.
- If all examples belong to the same category, entropy is 0 (by definition $0 \log_2 0$ is defined to be 0).
- If examples are equally mixed ($p^+ = p^- = 0.5$) then entropy is a maximum at 1.0.

40

Entropy (Cont'd)

- Entropy can be viewed as the number of bits required on average to encode the class of an example in S , where data compression (e.g. Huffman coding) is used to give shorter codes to more likely cases.
- For multiple-category problems with c categories, entropy generalizes to:

$$Entropy(S) = \sum_{i=1}^c -p_i \log_2(p_i)$$

- where p_i is proportion of category i examples in S .

41

Information Gain

- The information gain of an attribute is the expected reduction in entropy caused by partitioning on this attribute:

$$Gain(S, A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

- where S_v is the subset of S for which attribute A has value v and the entropy of the partitioned data is calculated by weighting the entropy of each partition by its size relative to the original set.

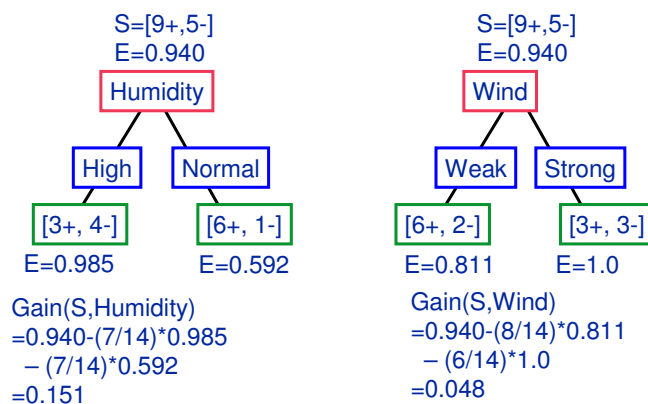
42

Training Examples

Day	Outlook	Temp.	Humidity	Wind	Play Tennis
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	Weak	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cold	Normal	Weak	Yes
D10	Rain	Mild	Normal	Strong	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

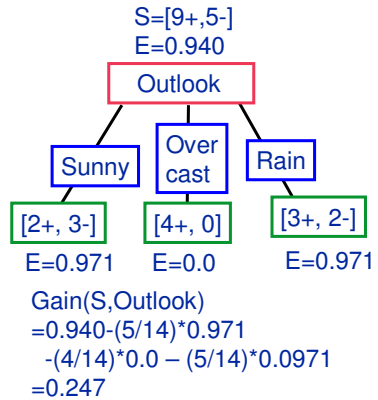
43

Selecting the Next Attribute



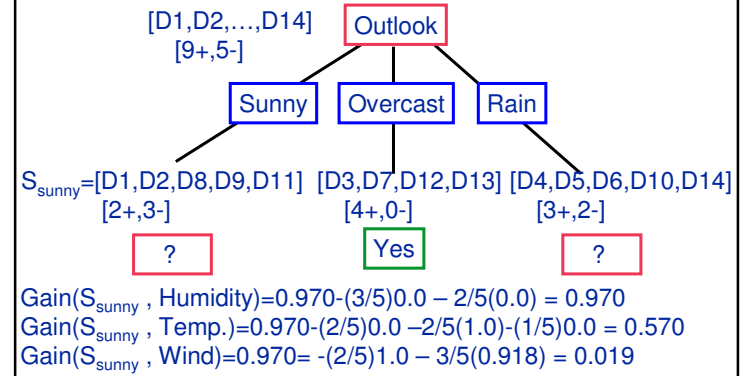
44

Selecting the Next Attribute



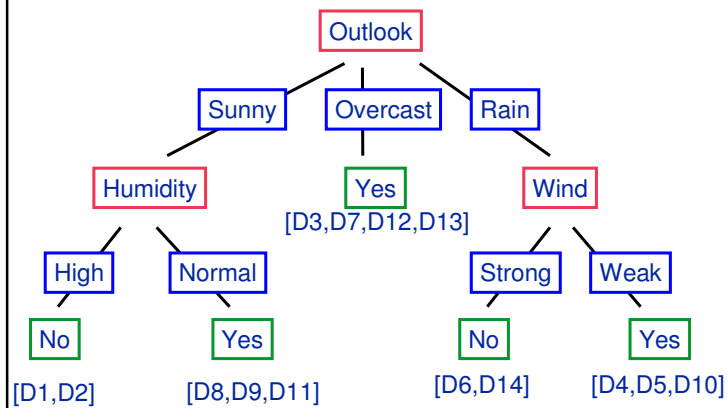
45

ID3 Algorithm



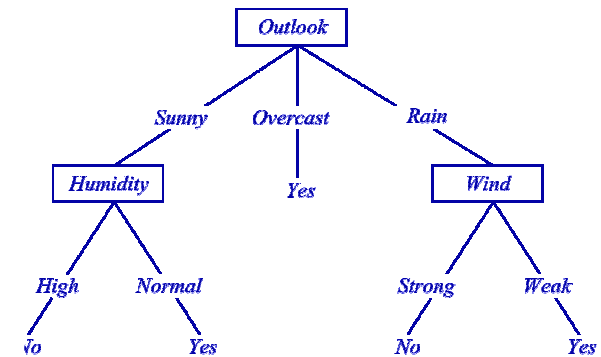
46

ID3 Algorithm



47

Converting A Tree to Rules



48

Converting A Tree to Rules

IF (Outlook = Sunny) and (Humidity = High)

THEN PlayTennis= No

IF (Outlook = Sunny) and (Humidity = Normal)

THEN PlayTennis= Yes

49

Case Studies

- Many case studies have shown that decision trees are at least as accurate as human experts.
- For example, one study for diagnosing breast cancer had humans correctly classifying the examples 65% of the time, and the decision tree classified 72% correct.
- British Petroleum designed a decision tree for gas-oil separation for offshore oil platforms. Replaced a rule-based expert system.
- Cessna designed an airplane flight controller using 90,000 examples and 20 attributes per example.

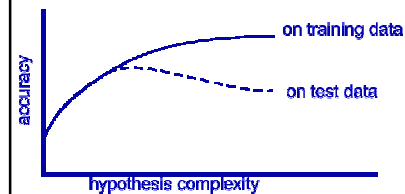
50

Noisy Data

- Two examples have the same attribute, value pairs, but different classifications
- Some values of attributes are incorrect because of errors in the data acquisition process or the preprocessing phase
- The classification is wrong (e.g., + instead of -) because of some error
- Some attributes are irrelevant to the decision-making process. For example, the color of a die is irrelevant to its outcome.

51

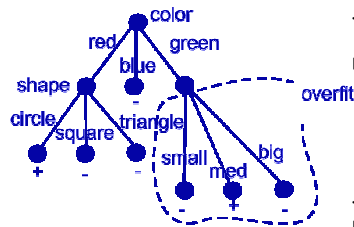
Overfitting and Pruning



- Learning a tree that classifies the training data perfectly may not lead to the tree with the best generalization performance since
 - ↗ There may be noise in the training data that the tree is fitting.
 - ↗ The algorithm might be making some decisions toward the leaves of the tree that are based on very little data and may not reflect reliable trends in the data.
- A hypothesis, h , is said to overfit the training data if there exists another hypothesis, h' , such that h has smaller error than h' on the training data but h' has smaller error on the test data than h .

52

Overfitting and Noise



- Category or attribute noise can cause overfitting.
- Add noisy instance
<<medium, green, circle>, +> (really -)
- Noise can also cause directly conflicting examples with same description and different class. Impossible to fit this data and must label leaf with majority category.
<<big, red, circle>, -> (really +)
- Conflicting examples can also arise if attributes are incomplete and inadequate to discriminate the categories.

53

Methods to Avoid Overfitting

- Two basic approaches to when pruning occurs
 - - **Prepruning**: Stop growing the tree at some point during construction when it is determined that there is not enough data to make reliable choices.
 - - **Postpruning**: Grow the full tree and then remove nodes that seem do not have sufficient evidence.

54

Methods for evaluating which subtrees to prune

- **Cross-validation**: Reserve some of the training data as a hold-out set (validation set, tuning set) to evaluate utility of subtrees.
- **Statistical testing**: Perform some statistical test on the training data to determine if any observed regularity can be dismissed as likely due to random chance.
- **Minimum Description Length (MDL)**: Determine if the additional complexity of the hypothesis is less complex than just explicitly remembering any exceptions.

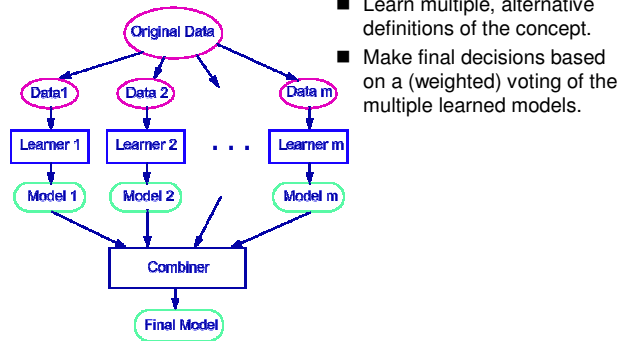
55

Missing Attribute Values

- Many times values are not available for all attributes during either training or test.
- One approach is to “fraction” examples based on the a priori probability of each value determined from the examples at that node that have specified values.
- During training, example counts used to calculate the evaluation heuristic include the fractional counts of examples with missing values. Examples with multiple missing values can be fractioned multiple times into numerous smaller and smaller “pieces.”

56

Multiple Models



57

Bagging

- Create multiple models by training the same learner on different samples of the training data (Breiman, 1996).
- Given a training set of size n , create m different training sets of size n by sampling from the original data with replacement.
- Combine the m models using simple majority vote.
- Can be applied to any learning method.
- Decreases generalization error by reducing variance in the results for unstable learners, i.e. algorithms whose hypothesis can change dramatically when their training data is altered only slightly.

58

Boosting

- Another method for producing multiple models by repeatedly altering the data given to an existing learner.
- Developed by computational learning theorists to guarantee performance improvements for weak learners that only need to generate an hypothesis with an accuracy greater than $1/2$.
- Examples are given weights, and at each iteration a new hypothesis is learned and the examples are reweighted to focus the system on examples that the latest hypothesis gets wrong.

59

Clustering

Task: organizing the data into meaningful clusters (groups)

Goal: to organize the data such that similar items are clustered together to reveal regularities in the data. There are no predefined categories.

Example clustering tasks:

1. Clustering customers into groups that represent similar buying patterns.
2. Clustering politicians into groups that represent similar voting patterns.

60

Instance-based (case-based) learning

- Instance-based reasoning systems keep track of previously seen instances and apply them directly to new ones.
- In general, the system simply stores each “case” that it experiences in a “case base” which represents its memory of previous episodes.
- To reason about a new instance, the system consults its case base and finds the most similar case that it’s seen before. The old case is then applied to the new situation.
- IBL is similar to reasoning by analogy. Many people believe that much of human learning is case- based in nature.

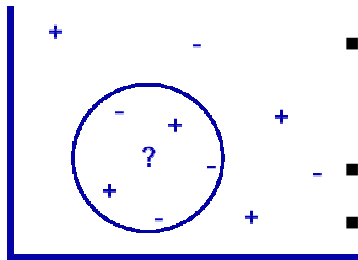
61

Distance Metrics

- Instance-based methods assume a function for calculating the (dis)similarity of two instances.
- For continuous feature vectors, just use Euclidean distance
- For discrete features, just assume distance between two values is 0 if they are the same, 1 if different (e.g. Hamming distance).
- To compensate for differences in units, scale all continuous values to normalize their values to be between 0 and 1.

62

K-Nearest Neighbor



- Calculate the distance between a test instance and every training instance.
- Pick the k closest training examples and assign the test example to the most common category among these “nearest neighbors”
- 1-nearest neighbor says +, 5-nearest neighbor says –
- Voting multiple neighbors helps increase resistance to noise. Odd values of k normally used to avoid ties. $k=1,3,5$ most common values used.

63

Learning Rules

- If-then rules are a standard representation of knowledge and concepts that has proven useful in building expert systems and other AI systems.
- Rules are relatively easy for people to understand and useful in providing insight and understanding of the regularities in data, and are therefore frequently preferred in data mining applications.
- There are a number of methods for inducing rules from data which have been shown to more easily build expert systems than manual knowledge engineering.
- Rule learning methods can be extended to handle relational (structural, first-order) representations such as learning Prolog programs from input/output pairs (Inductive Logic Programming).
- Thus, rule learning has moved beyond just handling “propositional” data that must be represented by a fixed, finite vector of predetermined features.

64

Rule Learning Methods

- Translate decision trees into rules (C4.5).
- Sequential (set) covering algorithms.
 - General to specific (top down) (CN2, FOIL)
 - Specific to general (bottom up) (GOLEM)
 - Hybrid Search (AQ, Chillin, Progol)
- Translate neural-networks into rules (TREPAN).

65

Bayesian Learning

- Bayesian learning techniques apply Bayesian statistics to data.
- Statistical techniques can be extremely powerful, but they rely on very large data sets.
- Bayesian classifiers can be structured to represent independent features, or hierarchical relationships in a Bayesian network.
- Statistical learning has demonstrated great success on many tasks, including speech recognition.

66

Neural Networks

- Neural networks are inspired by the interconnectivity of the brain.
- Connectionist networks typically consist of many nodes that are highly interconnected. When a node is activated, it sends signals to other nodes so that they are activated in turn.
- Using layers of nodes allows connectionist models to learn complex functions.
- Connectionist models are often amenable to parallelism, but it can sometimes be difficult to understand their behavior.

67

Reinforcement learning

- Reinforcement learning is often used for control problems . The model produces a series of responses, and then a reinforcement signal is provided by an external source as feedback.
- However, the learning system only gets delayed reinforcement so it is not told whether each individual response is correct or incorrect. It is only told whether the final result is positive or negative!
- One type of reinforcement learning tries to learn from temporally successive predictions (it is sometimes called temporal- difference learning)

68

Explanation- based Learning

- Explanation- based learning (EBL) systems try to explain why each training instance belongs to the target concept. The resulting “proof” is generalized and saved.
- If a new instance can be explained in the same manner as a previous instance, then it is also assumed to be a member of the target concept.
- One of the strengths of EBL is that the resulting “explanations” are typically easy to understand.
- One of the weaknesses of EBL is that they rely on a domain theory to generate the explanations.

69

Issues in Machine Learning

- What algorithms can approximate functions well and when?
- How does the number of training examples influence accuracy?
- How does the complexity of hypothesis representation impact it?
- How does noisy data influence accuracy?
- What are the theoretical limits of learnability?

70

Limitations of Some ML Research

Technique	Description	Problem
Explanation-Based Learning	Use domain theory to speed up learning process	How to acquire this domain theory?
Decision Tree or Rule Induction	Learn a decision structure from hundreds to thousands of annotated examples	Works only with trivial representations, cannot employ significant domain knowledge, and requires annotations
Inductive Logic Programming	Learn first-order representations	Huge search space requires domain-specific constraints, which must be identified and laboriously hand-coded
Reinforcement Learning (RL)	Learn which actions to perform in any state based on a final reward	Learning is slow! Need models to provide feedback before a final reward is given
Evolutionary Computation	Genetic operators permit search in a huge space	Requires thousands of learning trials, and domain-specific representations must be hand-coded
Neural Networks	Learn non-linear functions	Requires thousands of learning trials, and learning results were opaque (though translation has been frequently studied)
Bayesian Networks	Relate beliefs through probabilistic/causal links	How to obtain probabilities? How can these networks be learned?

(Aha, 2005)

71