

Machine Learning

Burr H. Settles
CS-540, UW-Madison
www.cs.wisc.edu/~cs540-1
Summer 2003

1

Announcements (7/15)

- If you haven't already, read Sections 18.1-18.3 in *AI: A Modern Approach*
- Homework #3 due tomorrow
 - The handin directories set up for you to submit your prolog programs
- Homework #4 will be out soon
 - Will have a programming portion

2

Announcements (7/16)

- Homework #3 due today
- Read Sections 20.4 and 20.5 in *AI: A Modern Approach* for next time
- This week's discussion topic: describe real-world inductive learning task
 - Is it a classification or regression problem?
 - What are a good set of features?

3

Comments on the Midterm

- Skolemizing
 - Forget what I said yesterday about a predicate connecting two variables (brain fart, grrr...)
 - Instead: work from the outside in, and substitute each existentially quantified variable with a Skolem function dependent on the universally quantified variables on the left (see p. 296 of *AIMA*)
- There was a typo in the exam (5.a.iii):
 - i.e. *not* UR: $((A \vee B) \wedge \neg B) \Leftrightarrow A$
 - UR: $((A \vee B) \wedge \neg B) \Rightarrow A$
 - The first isn't a tautology, but it doesn't change the answer to the question! (still *true*: each question has 4 interpretations, 3 models)

4

Agents that Don't Learn

- So far, all the types of intelligent agents we've discussed are quite "hard-wired"
 - Search through a problem space (perhaps using defined heuristics, or randomness) to find a good solution
 - Use expert-written logical knowledge
- These approaches are good for well-understood or definable environments, but what if things are too novel or more complex?

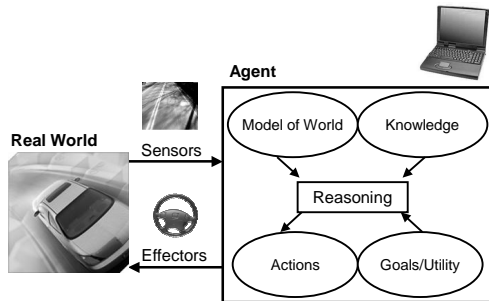
5

Agents that Learn

- Learning is essential for unknown environments
 - Too complex/rich to represent in a search space, or to search efficiently
 - Programmer doesn't know enough to write a sufficient knowledge base
- Learning is also a useful construction method
 - Expose the agent to reality and let it sort the problem out rather than programming it
- * *Learning modifies the agent's decision-making mechanisms to improve performance*

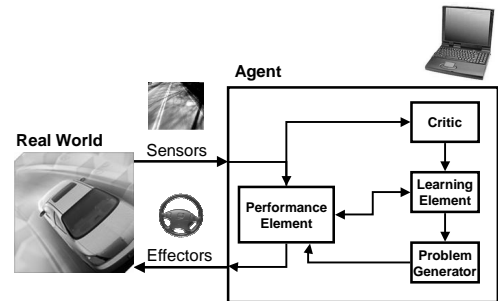
6

Old Agent Architecture



7

Learning Agent Architecture



8





Inductive Learning

- Inductive learning is the simplest form of learning (can also be considered *science*)
 - Learn a function from examples
- Scaled-down model of real learning
 - Ignores prior knowledge
 - Assumes deterministic, observable environment
 - Assumes training examples are available
 - Assumes that the agent *wants* to learn...?

9

Inductive Learning

- Problem framework:
 - Given a set of training examples as pairs: $\langle x, f(x) \rangle$
 - x is the example itself, and $f(x)$ is the concept to be learned
 - Find a hypothesis function $h(x)$ such that $h(x) \approx f(x)$

$x =$		$f(x) = \text{mammal}$
$x =$		$f(x) = \text{mammal}$
$x =$		$f(x) = \text{bird}$
$x =$		$f(x) = \text{bird}$

10

Representing Examples

- The main issue for inductive learning is how to *represent* the example x as data
 - Example x must somehow be mapped to input(s) for the hypothesis function $h(x)$
 - Must still capture the nature and the important features of the example
- We typically represent an example as a vector of features (or attributes), e.g. $x = \langle x_1, x_2, x_3, \dots \rangle$

11

Feature-Vector Representation

- Imagine you're in the circus, and company policy says that if more than 1,000 people attend in a day, you need extra security guards
- However, if you hire extra guards on a day with less than 1,000 customers, you lose money!
- You must also notify the extra guards 24 hours in advance... so you want to be able to predict if over 1,000 will attend or not

12

Feature-Vector Representation

- Let's say you have the nightly weather forecast and attendance records for the last 2 weeks
- We can think of each day as an example x :
 - Each of the weather measurements (outlook, temperature, humidity, etc.) are features of x
 - Whether or not there were >1,000 customers is our binary concept function $f(x)$
- If we can learn the concept well enough, we can predict the attendance for the next day based on the nightly forecast information

13

Feature-Vector Representation

Day	Outlook	Temperature	Humidity	Wind	>1,000?
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

14

A Hypothesis for the Circus

- The feature-vector corresponds to the set of all the agent's percepts
- Try hand-writing a series of if-then rules that characterizes what is observed in the previous set of examples
 - For instance: $Outlook=Sunny \wedge Humidity=High \Rightarrow No$
- ★ This set of rules is comprises a hypothesis function $h(x)$

15

A Hypothesis for the Circus

Day	Outlook	Temperature	Humidity	Wind	>1,000?
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

16

A Hypothesis for the Circus

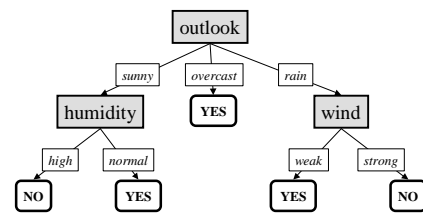
- One possible set of rules is:

$Outlook=Sunny \wedge Humidity=High \Rightarrow No$
 $Outlook=Sunny \wedge Humidity=Normal \Rightarrow Yes$
 $Outlook=Overcast \Rightarrow Yes$
 $Outlook=Rain \wedge Wind=Weak \Rightarrow Yes$
 $Outlook=Rain \wedge Wind=Strong \Rightarrow No$

17

Decision Trees

- Notice that the previous agent can also be represented as a logical tree:



18

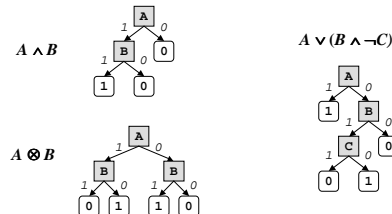
Decision Trees

- Decision trees are graphical representations of logical functions
 - Often very compact compared to truth tables
- They are one possible representation for the hypothesis function $h(x) \approx f(x)$:
 - Leaves (terminal nodes) are the results of $h(x)$
 - In this example, both $h(x)$ and $f(x)$ are the Boolean function "will more than 1,000 people attend?"

19

Expressiveness of D-Trees

- Decision trees can express any logical function of the input attributes:



20

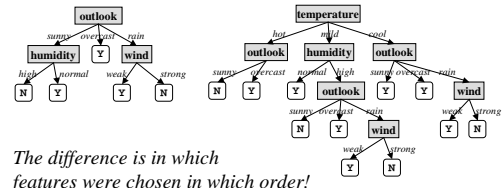
Decision Tree Induction

- * It would be nice to be able to induce the decision tree automatically from data, rather than trying to hand-write the rules*
- Fairly trivial to induce a decision tree from training data in a feature-vector representation:
 - Pick some feature x_i as the root node
 - Create an edge for each possible value of x_i
 - If all the examples that flow down the path from the root to this edge have the same $f(x)$ value, add a leaf for that value
 - Else, pick another feature x_j and add a node here
 - Recursively repeat until you can add a leaf

21

Decision Tree Induction

- However, note that both of these trees are consistent with the circus training data:



The difference is in which features were chosen in which order!

22

Hypothesis Spaces

- A hypothesis space is the set of *all* the possible hypothesis functions (in this case, decision trees) for a given problem description
- How big is a hypothesis space for decision trees?
 - Consider n Boolean features
 - The size of this hypothesis space = number of distinct decision trees over n features

23

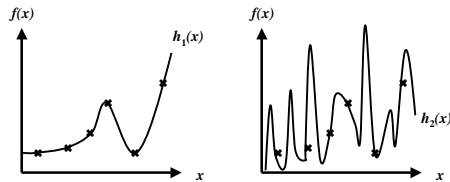
Hypothesis Spaces

- How many decision trees with n Boolean features?
 - # of Boolean functions
 - # of distinct truth tables with 2^n rows = 2^{2^n}
 - e.g., for 6 Boolean features there can be up to 18.4×10^{18} trees!!
 - Not all are necessarily consistent with training data, of course
- How many purely conjunctive hypotheses (e.g. $A \wedge \neg B$) are there for n Boolean features?
 - Each feature is in (1), in (0), or out
 - 3^n distinct conjunctive hypotheses (e.g. path from root to leaf)
- * More expressive hypothesis spaces increase the chance of fitting the function, but also increase complexity!*

24

Inductive Bias

- If there are several hypotheses that are all consistent with the training data, which should we prefer?



We want to introduce an inductive bias to prefer h_1 over other hypothesis, since it seems to generalize more

25

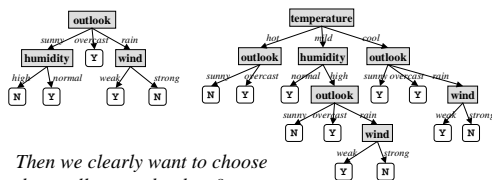
Occam's Razor

- English philosopher William of Occam was the first to address the question in 1320
 - Apparently while shaving?
- The inductive bias is called Occam's Razor:
 - Prefer the *simplest* hypothesis that fits the data
- But how do we define *simple*?

26

Occam's Razor and D-Trees

- We could say that, for decision trees, the simplest hypothesis is the tree with the fewest nodes



Then we clearly want to choose the smaller tree, but how?

27

Occam's Razor and D-Trees

- One way to find the smallest (i.e. simplest, or most general) decision tree is to enumerate all of them and choose the one with the fewest nodes
 - But the hypothesis space is too large!
- Alternatively: use the induction algorithm from slide 21 (or page 658 of *AIMA*), using some heuristic to choose the best feature x_i to add

28

ID3: Efficient Tree Induction

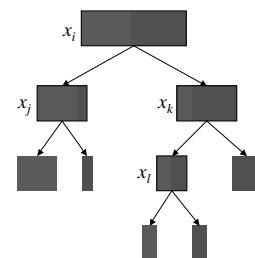
■ J.R. Quinlan, "Induction of Decision Trees," *Machine Learning*, 1986

- With the ID3 algorithm, there are many ways to choose the "best feature" for adding at a node
 - In general, we will use information theory
 - First developed by Shannon & Weaver at AT&T labs (used in digitizing telephone signals)
 - Information gain: amount of information (in bits) that is added by a certain feature

29

Information Theory Illustration

Say we're learning a Boolean concept, and have Boolean features:



Begin with the entire training set

Choose the feature that, when added, partitions the training set into the "purest" subsets

Do this recursively until nodes are totally pure (leaves)

30

Entropy

- To define information gain, we must first define entropy, which characterizes the (im)purity of set of examples S , in “bits”:

$$Entropy(S) = -p_+ \log_2 p_+ - p_- \log_2 p_-$$

- Where p_+ is the proportion of positive examples in S and p_- is the proportion of negatives
- Note: we will consider $0 \log_2 0 = 0$ (not *undefined*)

31

Entropy Example

- For example, the circus domain has a set S of 14 examples: 9 positives ($f(x) = Yes$) and 5 negatives ($f(x) = No$):

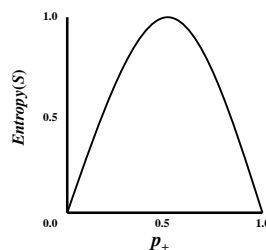
$$\begin{aligned} Entropy([9+,5-]) &= -(9/14) \log_2 (9/14) - (5/14) \log_2 (5/14) \\ &= -(0.64) \log_2 (0.64) - (0.36) \log_2 (0.36) \\ &= -(-0.41) - (-0.53) \\ &= 0.94 \end{aligned}$$

32

Entropy

Entropy reflects the lack of “purity” of some particular set S

As the proportion of positives p_+ approaches 0.5 (very impure), the Entropy of S converges to 1.0



33

Information Gain

- Now we can compute the information gain of adding a particular feature F on the set S in terms of the entropy:

$$InfoGain(F, S) = Entropy(S) - \sum_{v \in values(F)} |S_v|/|S| Entropy(S_v)$$

- Where $values(F)$ is the set of possible values for the feature F (e.g. $values(Wind) = \{Weak, Strong\}$)

34

Information Gain Example

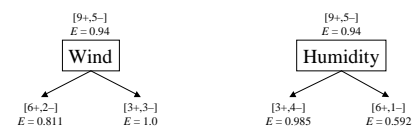
- Again, the circus example:

$$S = [9+,5-] \quad S_{Weak} = [6+,2-] \quad S_{Strong} = [3+,3-]$$

$$\begin{aligned} InfoGain(Wind, S) &= Entropy(S) - \sum_{v \in \{Weak, Strong\}} |S_v|/|S| Entropy(S_v) \\ &= Entropy(S) - (8/14)Entropy(S_{Weak}) \\ &\quad - (6/14)Entropy(S_{Strong}) \\ &= 0.94 - (0.57)0.81 - (0.43)1.00 \\ &= 0.048 \end{aligned}$$

35

Which Feature is Better?



$$\begin{aligned} InfoGain(Wind, S) &= 0.94 - (8/14)0.811 - (6/14)1.00 \\ &= 0.048 \end{aligned}$$

$$\begin{aligned} InfoGain(Humidity, S) &= 0.94 - (7/14)0.985 - (7/14)0.592 \\ &= 0.151 \end{aligned}$$

Humidity provides greater information gain (more pure subsets) than *Wind* on the training set as a whole. This makes it the better choice at this point in the tree

36

Issues with Information Gain

- Consider adding the feature *Date* to the feature vector in the circus problem
 - Each example would have a unique date
 - Therefore, each value of feature *Date* would perfectly purify the training set
 - But this won't be very useful in predicting in the future!
- To remedy this we can alternatively use the gain-ratio measure, which is a normalized information gain that discourages features with more or less uniformly distributed values
- Section 3.7.3 in *Machine Learning* covers more advanced decision tree heuristics in more detail

37

Generalizing Information Gain

- As presented, *Entropy* and thus *InfoGain* only work for learning Boolean concepts
 - The circus problem is Yes/No
- We may want to generalize this to more than two classes (e.g. Labeling objects as animal, vegetable, or mineral)

$$Entropy(S) = \sum_i -p_i \log_2 p_i$$

- Where i ranges over all the labels in the concept

38

Types of Features

- There are three main kinds of features we can use in inductive learning:
 - Boolean (2 values, e.g. *Wind*)
 - Discrete (>2 fixed values, e.g. *Outlook*)
 - Continuous (real numbers, e.g. what *Temperature* perhaps should be)
 - Difficult for decision trees to deal with (not a *logical* construct)
 - Must partition the training set on some value
 - But there are a potentially *infinite* number of thresholds for splitting up a continuous domain!

39

Handling Continuous Features

- One way of dealing with a continuous feature F is to treat them like Boolean features, partitioned on a dynamically chosen threshold t :
 - Sort the examples in S according to F
 - Identify adjacent examples with differing class labels
 - Compute *InfoGain* with t equal to the average of the values of at these boundaries
 - Can also be generalized to multiple thresholds
 - U. Fayyad and K. Irani, "Multi-interval discretization of continuous-valued attributes for classification learning," *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, 1993

40

Handling Continuous Features

- There are two candidates for threshold t in this example:

Temperature	40	48	60	72	80	90
>1,000?	No	No	Yes	Yes	Yes	No

$$t = (48+60)/2 = 54$$

$$t = (80+90)/2 = 85$$

- The dynamically-created Boolean features $Temp_{>54}$ and $Temp_{>85}$ can now compete with the other Boolean and discrete features in the dataset

41

Dealing with Noise

- Consider two or more examples that all have the exact same feature descriptions, but have different labels
 - e.g. The concept is whether or not you find someone is attractive... two people might have the same height, weight, haircolor, etc., but you think one is cute and the other isn't
- This is called noise in the data
- Encountered in ID3 when all features are exhausted, but the examples are not homogenous
 - Solve by adding a leaf with the majority class label value
 - Break ties randomly

42

Tree Induction as Search

- We can think of inducing the “best tree” as an optimization search problem:
 - **States:** possible (sub-)trees
 - **Actions:** add a feature as a node of the tree
 - **Objective Function:** increase the overall information gain of the tree
- Essentially, ID3 is a hill-climbing search through the hypothesis space, where the heuristic picks features that are likely to lead to small trees

43

Evaluating Learning Agents

- Recall that we want the learned hypothesis $h(x)$ to approximate the real concept function $f(x)$
- Therefore, a reasonable evaluation metric for a learned agent is percent accuracy on some set of labeled examples $\langle x, f(x) \rangle$
- * *But we don't want to evaluate on the set of examples we trained on (that would be cheating)!*

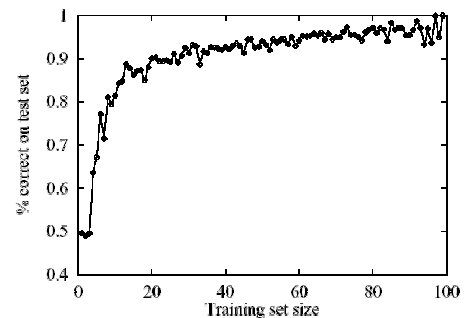
44

Experimental Methodology

- To conduct a reasonable evaluation of how well the agent has learned a concept:
 - Collect a set of labeled examples
 - Randomly partition it into two disjoint subsets: the training set and the test set
 - Apply the learning algorithm (e.g. ID3) to the training set to generate a hypothesis h
 - Measure the percent of examples in the test set accurately labeled by h
- This can be repeated for different, increasing sizes of the training set to construct a learning curve

45

Example Learning Curve



46

Cross-Validation

- One problem with a simple train/test split of the data is that the test set may happen to contain a particularly easy (or difficult) set of examples
- Cross-validation is a way to get a better estimate of an algorithm's performance
- Leave-one-out validation:
 - Train on all but one example in the dataset, and predict the one example that was held out
 - Repeat over the entire dataset and compute accuracy over all of the held-out predictions
 - Time consuming... if there are n examples, we must running the learning algorithm $n-1$ times!

47

k -Fold Cross-Validation

- k -fold cross-validation is a simplified version of leave-one-out:
 - Partition the data into k random, equally sized “folds” with no redundancy
 - Run the learning algorithm on all but one of the folds (effectively the training set), and evaluate accuracy on the held-out fold (test set)
 - Repeat over all k folds and average the performance
 - Leave-one-out is k -fold validation with $k = n$
 - The standard in the ML community is 10-fold cross-validation (results usually close to leave-one-out)

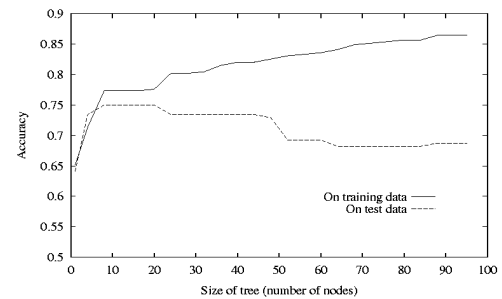
48

Overfitting

- There is a tradeoff that comes with having an expressive hypothesis space:
 - It is more likely that our hypothesis $h(x)$ will fit (or approximate) the actual $f(x)$ exactly
 - But because our training set is a representative sample of $f(x)$, we run the risk of overfitting the training data
- * *Overfitting causes the agent to “memorize” the training data, keeping it from generalizing well to new examples*

49

Overfitting



50

Overfitting Avoidance

- To deal with overfitting in decision trees, we can try two things:
 - Stop growing when the information gain stops being statistically significant
 - Difficult to gauge, doesn't work well in practice
 - Grow the full tree on training data, and then prune the tree
 - Remember Occam's razor: simplify!
- But how do we know what to prune?

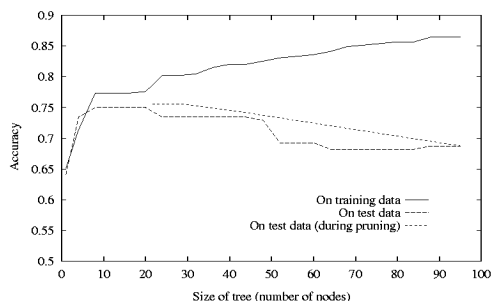
51

Decision Tree Pruning

- The answer is to take the training set and break it up into a sub-training set, and a tuning set, on which we will “fine-tune” (or prune) our hypothesis
 - Induce a tree on the sub-training set
 - Consider pruning each node (and those below it) and evaluate impact on the tuning set
 - Greedily remove the one that most improves performance on the tuning set
- Why don't we want to prune on the test set?
 - The algorithm isn't supposed to be allowed to know the class labels for the test set!

52

Decision Tree Pruning



53

Properties of Decision Trees

- Decision tree learning is fast in practice
- Applied to many real-world problems,
 - Part-picking robots
 - Financial decision-making software
- Another bonus: comprehensibility
 - It is easy to look at the structure and/or the rules of a learned d-tree and understand the concept that has been learned
 - After all, they're basically logical rules!

54

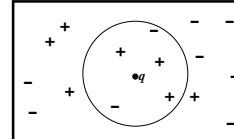
Eager vs. Lazy Learning

- Decision tree induction is called an eager learning method because it actively (eagerly) constructs a model hypothesis function
- There are also lazy learning methods (or instance-based learning) which simply memorize aspects of the training examples and compare new examples to what it's "learned"

55

k -Nearest Neighbors

- The k -nearest neighbors (k -NN) algorithm is the most common form of lazy learning
 - Retain all the training data in memory
 - When a test example is queried, let the k most similar training examples "vote" on the class label



Consider this Venn Diagram with both + and - examples

If we are using 5-NN learning, what is the label for the point q ?

The vote is 3-2 in favor of +

56

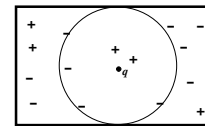
Evaluating Distance

- Given a query (test) example q , we compare it to every x in the training set and let the nearest k vote
- To evaluate which training examples are "nearest" to the query, we need a distance metric!
 - Boolean and discrete features
 - Hamming distance: # of features in x and q that do not match
 - Continuous features
 - Euclidian distance: $distance(x, q) = \sqrt{\sum_i (x_i - q_i)^2}$ where i ranges over all the examples' features
 - The two can be combined if all feature types are present

57

Distance-Weighted k -NN

Consider the following Venn Diagram:



- If we conduct 5-NN learning in this rather sparse problem, we'll probably end up misclassifying q
- To remedy this by conducting a weighted vote
 - Compute a weight w for each example x : $w = 1 / distance(x, q)^2$
 - This assumes that the distances are normalized
 - Now the examples nearest q will have more influence in the vote

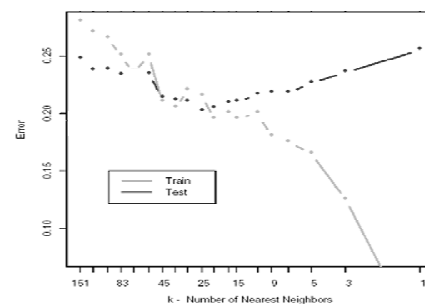
58

The Key to k -NN

- The most important parameter in the k -NN algorithm is the value for k itself: how many neighbors are needed?
 - If k is too low, we consider few examples and don't generalize well (risk overfitting)
 - If k is too high, we over-generalize and lose the sense of relationship between the query and the examples
 - Page 734 has some good illustrations of the tradeoff
- Section 8.2 of *Machine Learning* covers all the k -NN related issues well

59

The Key to k -NN



60

Tuning k

- As we did with decision tree pruning, we can “tune” the value of k by splitting the training set into a sub-training set and a tuning set
 - Consider several values for k , and evaluate performance against the tuning set
 - Choose the value of k that showed the best performance (lowest error)

★ *Tuning the value of k can make or break the utility of k -NN learning agents*

61

Properties of k -Nearest Neighbors

- k -NN can be more robust to noisy data than decision trees
 - If ≥ 2 identical examples have conflicting labels, they aren't the only ones in the neighborhood
- The inductive bias is toward examples with small Euclidian distance from the query
- However, k -NN computes distance based on all features, whereas d-trees don't necessarily
 - Can fix by weighting “important” features higher

62

Regression Learning

- So far, we've assumed the concept function $f(x)$ to be a classification task
 - e.g. yes/no, +/-, animal/vegetable/mineral, etc...
- Sometimes we want the agent to learn real-valued functions, which is called a regression task
 - e.g. Predict the *exact* number of customers at the circus, not just the Boolean $>1,000$

63

Regression Learning

- Because decision trees represent logical functions, it is difficult to extend them to handle such regression problems
 - CART (Classification And Regression Trees)
 - ▣ J. Friedman, “A recursive partitioning decision tree rule for non-parametric classification,” IEEE Transactions on Computers, 1977
- k -NN is a bit better suited to regression problems
 - The estimated label is an average (or weighted average) of its neighbors, instead of a vote
 - This still has problems: what if $f(x)$ is polynomial?

64

Summary

- Learning allows an agent to sort tasks out for itself
 - Helpful for complex problem domains
 - Useful for not-well-understood problems
- Inductive learning is the task of creating a hypothesis which approximates some concept
 - Learning a discrete function is called classification
 - Learning a real-valued function is called regression
- Examples for inductive learning are represented as a feature-vectors (a vector of percepts)

65

Summary

- Decision tree induction is an eager learning method whose hypothesis represents logical functions
- k -Nearest Neighbors is a lazy learning method which compares test examples to recorded training data
- Machine Learning evaluation is typically done using separate training and test sets
- Overfitting the training data can usually be avoided by using a tuning set to tweak the model

66