CS 343: Artificial Intelligence Machine Learning

Raymond J. Mooney
University of Texas at Austin

1

What is Learning?

- Herbert Simon: "Learning is any process by which a system improves performance from experience."
- What is the task?
  - Classification
  - Problem solving / planning / control

2

Classification

- Assign object/event to one of a given finite set of categories.
  - Medical diagnosis
  - Credit card applications or transactions
  - Fraud detection in e-commerce
  - Worm detection in network packets
  - Spam filtering in email
  - Recommended articles in a newspaper
  - Recommended books, movies, music, or jokes
  - Financial investments
  - DNA sequences
  - Spoken words
  - Handwritten letters
  - Astronomical images

3

Problem Solving / Planning / Control

- Performing actions in an environment in order to achieve a goal.
  - Solving calculus problems
  - Playing checkers, chess, or backgammon
  - Balancing a pole
  - Driving a car or a jeep
  - Flying a plane, helicopter, or rocket
  - Controlling an elevator
  - Controlling a character in a video game
  - Controlling a mobile robot

4

Sample Category Learning Problem

- Instance language: <size, color, shape>
  - size $\in$ {small, medium, large}
  - color $\in$ {red, blue, green}
  - shape $\in$ {square, circle, triangle}
- *C* = {positive, negative}
- *D*:

Example	Size	Color	Shape	Category
1	small	red	circle	positive
2	large	red	circle	positive
3	small	red	triangle	negative
4	large	blue	circle	negative

5

Hypothesis Selection

- Many hypotheses are usually consistent with the training data.
  - red & circle
  - (small & circle) or (large & red)
  - (small & red & circle) or (large & red & circle)
  - not [(red & triangle) or (blue & circle)]
  - not [(small & red & triangle) or (large & blue & circle)]
- Bias
  - Any criteria other than consistency with the training data that is used to select a hypothesis.

6

Generalization

- Hypotheses must generalize to correctly classify instances not in the training data.
- Simply memorizing training examples is a consistent hypothesis that does not generalize.
- *Occam's razor*:
  - Finding a *simple* hypothesis helps ensure generalization.

7

Hypothesis Space

- Restrict learned functions a priori to a given ***hypothesis space***, H, of functions $h(x)$ that can be considered as definitions of $c(x)$.
- For learning concepts on instances described by n discrete-valued features, consider the space of conjunctive hypotheses represented by a vector of n constraints
  $<c_1, c_2, \ldots c_n>$ where each c_i is either:
  - ?, a wild card indicating no constraint on the ith feature
  - A specific value from the domain of the ith feature
  - Ø indicating no value is acceptable
- Sample conjunctive hypotheses are
  - <big, red, ?>
  - <?, ?, ?> (most general hypothesis)
  - < Ø, Ø, Ø> (most specific hypothesis)

8

Inductive Learning Hypothesis

- Any function that is found to approximate the target concept well on a sufficiently large set of training examples will also approximate the target function well on unobserved examples.
- Assumes that the training and test examples are drawn independently from the same underlying distribution.
- This is a fundamentally unprovable hypothesis unless additional assumptions are made about the target concept and the notion of "approximating the target function well on unobserved examples" is defined appropriately (cf. computational learning theory).

9

Evaluation of Classification Learning

- Classification accuracy (% of instances classified correctly).
  - Measured on an independent test data.
- Training time (efficiency of training algorithm).
- Testing time (efficiency of subsequent classification).

10

Category Learning as Search

- Category learning can be viewed as searching the hypothesis space for one (or more) hypotheses that are consistent with the training data.
- 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: Ø, T, F, ?, so there are 4^n syntactically distinct hypotheses.
- However, all hypotheses with 1 or more Øs are equivalent, so there are 3^n+1 semantically distinct hypotheses.
- The target binary categorization function in principle could be any of the possible $2^{2^{\wedge}n}$ functions on n input bits.
- Therefore, conjunctive hypotheses are a small subset of the space of possible functions, but both are intractably large.
- All reasonable hypothesis spaces are intractably large or even infinite.

11

Learning by Enumeration

- For any finite or countably infinite hypothesis space, one can simply enumerate and test hypotheses one at a time until a consistent one is found.

  For each h in H do:

  If h is consistent with the training data D,

  then terminate and return h.
- This algorithm is guaranteed to terminate with a consistent hypothesis if one exists; however, it is obviously computationally intractable for almost any practical problem.

12

Efficient Learning

- Is there a way to learn conjunctive concepts without enumerating them?
- How do human subjects learn conjunctive concepts?
- Is there a way to efficiently find an unconstrained boolean function consistent with a set of discrete-valued training instances?
- If so, is it a useful/practical algorithm?

Conjunctive Rule Learning

- Conjunctive descriptions are easily learned by finding all commonalities shared by all positive examples.

Example	Size	Color	Shape	Category
1	small	red	circle	positive
2	large	red	circle	positive
3	small	red	triangle	negative
4	large	blue	circle	negative

Learned rule: red & circle → positive

- Must check consistency with negative examples. If inconsistent, **no** conjunctive rule exists.

Limitations of Conjunctive Rules

- If a concept does not have a single set of necessary and sufficient conditions, conjunctive learning fails.

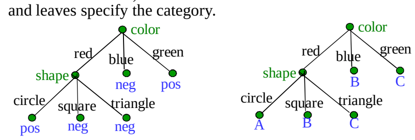
Example	Size	Color	Shape	Category
1	small	red	circle	positive
2	large	red	circle	positive
3	small	red	triangle	negative
4	large	blue	circle	negative
5	medium	red	circle	negative

Learned rule: red & circle → positive

Inconsistent with negative example #5!

Decision Trees

- Tree-based classifiers for instances represented as feature-vectors. Nodes test features, there is one branch for each value of the feature, and leaves specify the category.



- Can represent arbitrary conjunction and disjunction. Can represent any classification function over discrete feature vectors.
- Can be rewritten as a set of rules, i.e. disjunctive normal form (DNF).
  - red ∧ circle → pos
  - red ∧ circle → A
    blue → B; red ∧ square → B
    green → C; red ∧ triangle → C

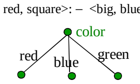
Properties of Decision Tree Learning

- Continuous (real-valued) features can be handled by allowing nodes to split a real valued feature into two ranges based on a threshold (e.g. length < 3 and length ≥3)
- Classification trees have discrete class labels at the leaves, ***regression trees*** allow real-valued outputs at the leaves.
- Algorithms for finding consistent trees are efficient for processing large amounts of training data for data mining tasks.
- Methods developed for handling noisy training data (both class and feature noise).
- Methods developed for handling missing feature values.

Top-Down Decision Tree Induction

- Recursively build a tree top-down by divide and conquer.

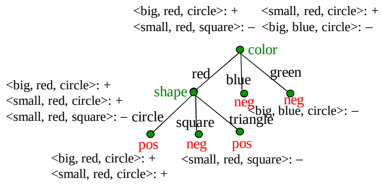
<big, red, circle>: + <small, red, circle>: +
<small, red, square>: – <big, blue, circle>: –



<big, red, circle>: +
<small, red, circle>: +
<small, red, square>: –

Top-Down Decision Tree Induction

- Recursively build a tree top-down by divide and conquer.



19

Decision Tree Induction Pseudocode

DTree(*examples*, *features*) returns a tree
If all *examples* are in one category, return a leaf node with that category label.
Else if the set of *features* is empty, return a leaf node with the category label that is the most common in examples.
Else pick a feature *F* and create a node *R* for it
For each possible value v_i of *F*:
Let $examples_i$ be the subset of examples that have value v_i for *F*
Add an out-going edge *E* to node *R* labeled with the value v_i.
If $examples_i$ is empty
then attach a leaf node to edge *E* labeled with the category that is the most common in *examples*.
else call DTree($examples_i$, *features* – {*F*}) and attach the resulting tree as the subtree under edge *E*.
Return the subtree rooted at *R*.

20

Picking a Good Split Feature

- Goal is to have the resulting tree be as small as possible, per Occam's razor.
- Finding a minimal decision tree (nodes, leaves, or depth) is an NP-hard optimization problem.
- Top-down divide-and-conquer method does a greedy search for a simple tree but does not guarantee to find the smallest.
  - General lesson in ML: "Greed is good."
- Want to pick a feature that creates subsets of examples that are relatively "pure" in a single class so they are "closer" to being leaf nodes.
- There are a variety of heuristics for picking a good test, a popular one is based on information gain that originated with the ID3 system of Quinlan (1979).

21

Entropy

- Entropy (disorder, impurity) of a set of examples, S, relative to a binary classification is:

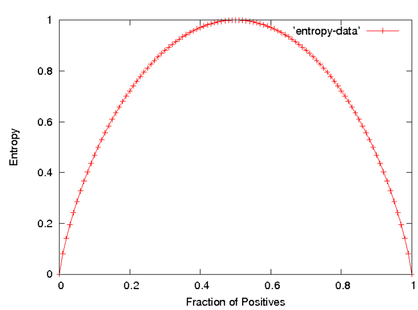
$$Entropy(S) = -p_1 \log_2(p_1) - p_0 \log_2(p_0)$$

where p_1 is the fraction of positive examples in S and p_0 is the fraction of negatives.
- If all examples are in one category, entropy is zero (we define $0 \cdot \log(0)=0$)
- If examples are equally mixed ($p_1=p_0=0.5$), entropy is a maximum of 1.
- 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 multi-class problems with c categories, entropy generalizes to:

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

22

Entropy Plot for Binary Classification



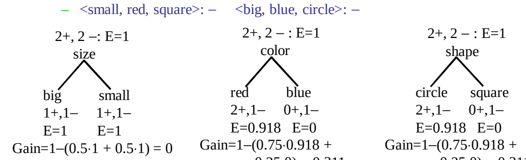
23

Information Gain

- The information gain of a feature *F* is the expected reduction in entropy resulting from splitting on this feature.

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

where S_v is the subset of *S* having value *v* for feature *F*.
- Entropy of each resulting subset weighted by its relative size.
- Example:
  - <big, red, circle>: + <small, red, circle>: +
  - <small, red, square>: – <big, blue, circle>: –



24

Hypothesis Space Search

- Performs ***batch learning*** that processes all training instances at once rather than ***incremental learning*** that updates a hypothesis after each example.
- Performs hill-climbing (greedy search) that may only find a locally-optimal solution. Guaranteed to find a tree consistent with any conflict-free training set (i.e. identical feature vectors always assigned the same class), but not necessarily the simplest tree.
- Finds a single discrete hypothesis, so there is no way to provide confidences or create useful queries.

25

Another Red-Circle Data Set

Example	Size	Color	Shape	Category
1	small	red	circle	positive
2	large	red	circle	positive
3	small	red	square	negative
4	large	blue	circle	negative

26

Weka J48 Trace 1

```
data> java weka.classifiers.trees.J48 -t figure.arff -T figure.arff -U -M 1
Options: -U -M 1
J48 unpruned tree
------------------
color = blue: negative (1.0)
color = red
|   shape = circle: positive (2.0)
|   shape = square: negative (1.0)
|   shape = triangle: positive (0.0)
color = green: positive (0.0)

Number of Leaves  :     5
Size of the tree :      7

Time taken to build model: 0.03 seconds
Time taken to test model on training data: 0 seconds
```

27

A Data Set Requiring Disjunction

Example	Size	Color	Shape	Category
1	small	red	circle	positive
2	large	red	circle	positive
3	small	red	triangle	negative
4	large	blue	circle	negative
5	small	green	circle	positive

28

Weka J48 Trace 2

```
data> java weka.classifiers.trees.J48 -t figure3.arff -T figure3.arff -U -M 1
Options: -U -M 1
J48 unpruned tree
------------------
shape = circle
|   color = blue: negative (1.0)
|   color = red: positive (2.0)
|   color = green: positive (1.0)
shape = square: positive (0.0)
shape = triangle: negative (1.0)

Number of Leaves  :     5
Size of the tree :      7

Time taken to build model: 0.02 seconds
Time taken to test model on training data: 0 seconds
```

29

Weka J48 Trace 3

```
data> java weka.classifiers.trees.J48 -t contact-lenses.arff
J48 pruned tree
------------------
tear-prod-rate = reduced: none (12.0)
tear-prod-rate = normal
|   astigmatism = no: soft (6.0/1.0)
|   astigmatism = yes
|   |   spectacle-prescrip = myope: hard (3.0)
|   |   spectacle-prescrip = hypermetrope: none (3.0/1.0)

Number of Leaves  :     4
Size of the tree :      7

Time taken to build model: 0.03 seconds
Time taken to test model on training data: 0 seconds

=== Error on training data ===

Correctly Classified Instances          22               91.6667 %
Incorrectly Classified Instances         2                8.3333 %
Kappa statistic                          0.8447
Mean absolute error                      0.0833
Root mean squared error                  0.2041
Relative absolute error                 22.6257 %
Root relative squared error             48.1223 %
Total Number of Instances               24

=== Confusion Matrix ===

  a  b  c   <-- classified as
  5  0  0 |  a = soft
  0  3  1 |  b = hard
  1  0 14 |  c = none

=== Stratified cross-validation ===

Correctly Classified Instances          20               83.3333 %
Incorrectly Classified Instances         4               16.6667 %
Kappa statistic                          0.71
Mean absolute error                      0.15
Root mean squared error                  0.3249
Relative absolute error                 39.7059 %
Root relative squared error             74.3898 %
Total Number of Instances               24

=== Confusion Matrix ===

  a  b  c   <-- classified as
  5  0  0 |  a = soft
  0  3  1 |  b = hard
  1  2 12 |  c = none
```

30

Evaluating Inductive Hypotheses

- Accuracy of hypotheses on training data is obviously biased since the hypothesis was constructed to fit this data.
- Accuracy must be evaluated on an independent (usually disjoint) test set.
- Average over multiple train/test splits to get accurate measure of accuracy.
- K-fold cross validation averages over K trials using each example exactly once as a test case.

31

K-Fold Cross Validation

Randomly partition data D into k disjoint equal-sized subsets $P_1 \ldots P_k$
For i from 1 to k do:
Use P_i for the test set and remaining data for training
$S_i = (D - P_i)$
$h_A = L_A(S_i)$
$h_B = L_B(S_i)$
$\delta_i = \text{error}_{P_i}(h_A) - \text{error}_{P_i}(h_B)$
Return the average difference in error:

$$\delta = \frac{1}{k}\sum_{i=1}^{k}\delta_i$$

32

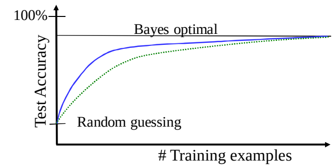
K-Fold Cross Validation Comments

- Every example gets used as a test example once and as a training example k–1 times.
- All test sets are independent; however, training sets overlap significantly.
- Measures accuracy of hypothesis generated for $[(k-1)/k]\cdot|D|$ training examples.
- Standard method is 10-fold.
- If k is low, not sufficient number of train/test trials; if k is high, test set is small and test variance is high and run time is increased.
- If k=|D|, method is called ***leave-one-out*** cross validation.

33

Learning Curves

- Plots accuracy vs. size of training set.
- Has maximum accuracy (Bayes optimal) nearly been reached or will more examples help?
- Is one system better when training data is limited?
- Most learners eventually converge to Bayes optimal given sufficient training examples.



34

Cross Validation Learning Curves

Split data into k equal partitions
For trial i = 1 to k do:
Use partition i for testing and the union of all other partitions for training.
For each desired point p on the learning curve do:
For each learning system L
Train L on the first p examples of the training set and record training time, training accuracy, and learned concept complexity.
Test L on the test set, recording testing time and test accuracy.
Compute average for each performance statistic across k trials.
Plot curves for any desired performance statistic versus training set size.

35