Supervised Learning Methods

- k-nearest-neighbors (k-NN)
- **Decision trees** (Chapter 18.3)
- Neural networks (ANN)
- Support vector machines (SVM)

1

Inductive Concept Learning by Learning Decision Trees

- Goal:
  Build a decision tree for classifying examples as one of a discrete set of possible values
  - a form of *supervised learning*
  - uses *batch processing* of training examples
  - uses a *preference bias*
    - Learning can be viewed as searching the Hypothesis Space H of possible h functions, $y = h(x)$
    - **Preference bias**: define a metric for comparing h's so as to determine whether one is better than another

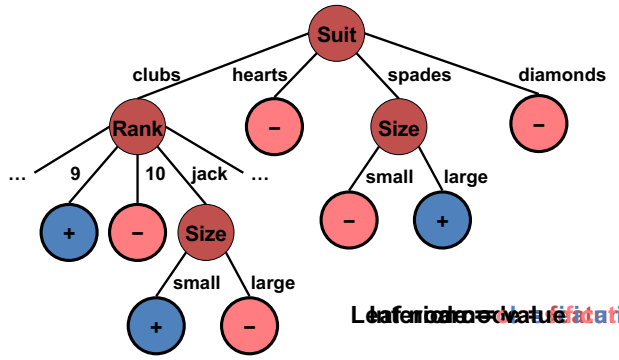
2

Inductive Concept Learning by Learning Decision Trees

- A **decision tree** is a tree in which:
  - each ***non-leaf node*** has associated with it an attribute (aka feature)
  - each ***leaf node*** has associated with it a discrete classification value (aka **class** or **label**, e.g., + or −)
  - each ***arc*** has associated with it one of the possible values of the (categorical) attribute of its parent node (i.e., node from where the arc is directed)

3

Inductive Concept Learning by Learning Decision Trees



4

Using a Decision Tree

A Decision Tree is used as a **classifier** by taking a given input example (aka instance), which is given by its feature vector, and:

1. The attribute at the root node of the tree is interpreted as a question, and the answer is determined by the value of that attribute in the input example
2. Answer determines to which child to move
3. Repeat until a leaf node is reached; class label at leaf is the classification predicted for the input example

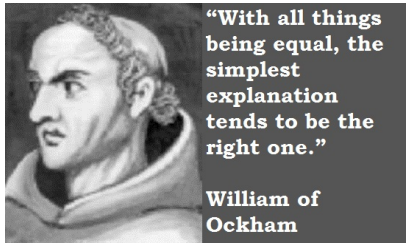
7

Inductive Concept Learning by Learning Decision Trees

- What is the best decision tree?
- Preference Bias: **Ockham's Razor**
  - The ***simplest hypothesis*** that is consistent with all observations is most likely
  - The **smallest decision tree** that correctly classifies all of the training examples is best
- Finding the provably smallest decision tree is an NP-Hard problem, so instead construct one that is "pretty small"

8

Ockham's Razor



(1287-1347)

"Everything should be made as simple as possible, but not simpler." – Albert Einstein

Ockham chooses a razor

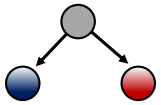
9

Decision Tree Construction using a Greedy Algorithm

- Aka **Decision-Tree-Learning** or **ID3** or **C5.0**
- Top-down (greedy) construction of the decision tree:
  1. Select the "best attribute" to use for the current node in the tree
  2. For each possible value of the selected attribute:
     a) Partition the examples using the possible values of this attribute, and assign these disjoint subsets of the examples to the appropriate child node
     b) Recursively generate each child node until (ideally) all examples for a node have same label (class)

10

Building a Decision Tree

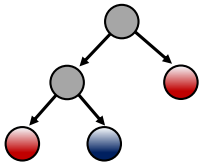


- Select an attribute and split the data into its children in a tree

Slide by Intel Software

11

Building a Decision Tree

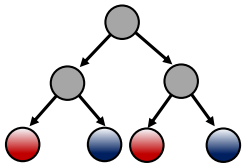


- Select an attribute and split the data into its children in a tree
- Continue splitting with available attributes

Slide by Intel Software

12

Building a Decision Tree

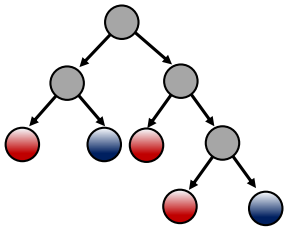


- Select an attribute and split the data into its children in a tree
- Continue splitting with available attributes

Slide by Intel Software

13

How Long to Keep Splitting?



Until:

- Leaf node(s) are pure (only one class remains)
- A maximum depth is reached
- A performance metric is achieved

Slide by Intel Software

14

Decision-Tree-Learning Algorithm

buildtree(*examples*, *attributes*, *default-label*)
if empty(*examples*) then return *default-label*
if (*examples* all have same label *y*) then return *y*
if empty(*attributes*) then return majority-class of *examples*
q = best_attribute(*examples, attributes*)
tree = create-node with attribute *q*
foreach value *v* of attribute *q* do
  v-ex = subset of examples with *q* == *v*
  subtree = **buildtree**(*v-ex*, *attributes* - {*q*}, *majority-class*(*examples*))
  add arc from *tree* to *subtree*
return *tree*

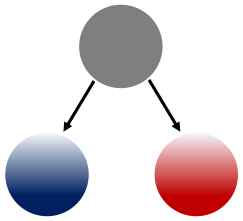
15

Real-Valued Attributes

- Not a discrete set of possible values
- Instead use a threshold to split data into 2 children
  - Height >= 168 and Height < 168
- Try multiple thresholds and pick the "best" threshold for this attribute

16

Building the Best Decision Tree



What defines the best split?

Slide by Intel Software

17

Decision Tree Algorithm

- Which is the "best attribute" ?
  - **Random:** an attribute chosen at random
  - **Least-Values:** the attribute with the *smallest* number of possible values
  - **Most-Values:** the attribute with the *largest* number of possible values
  - **Max-Gain:** the attribute that has the largest expected **information gain**

18

Information Gain

- **Goal:** Select the attribute that will result in the *smallest expected tree size*
- How?

  Use **information theory**

19

Information Theory

- How many yes/no questions would you expect to ask to determine which number I'm thinking of in the range 1 to 100?

  7
- Why?

  With each yes/no question at most 1/2 of the elements remaining can be eliminated

  $log_2 100 = 6.64$

20

Information Theory

- Given a set S of size $|S|$, the expected work required to determine a specific element is

  $log_2 |S|$
- Call this value the information value of being *told* the element rather than having to work for it (by asking questions)

21

Information Theory

Given $S = P \cup N$, where P and N are two disjoint sets, how hard is it to determine which element I am thinking of in S?

if x is in P,
then $log_2 p$ questions needed, where $p = |P|$

if x is in N,
then $log_2 n$ questions needed, where $n = |N|$

22

Information Theory

So, the expected number of *questions* that have to be asked is:

$(Pr(x \text{ in } P) * log_2 p) + (Pr(x \text{ in } N) * log_2 n)$

or, equivalently,

$(p/(p+n))\ log_2 p + (n/(p+n))\ log_2 n$

23

Entropy

- In general, say there are $n = n_1 + \ldots + n_k$ examples
  - n_1 examples have label y_1
  - n_2 examples have label y_2
  - …
  - n_k examples have label y_k
- What's the **impurity/inhomogeneity/disorder** of the examples?
- Turn it into a game: If I put these examples in a bag, and pick one at random, what is the probability the example has label y_i?

24

Entropy

- Probability estimated from the given samples:
  - probability $p_1 = n_1/n$ the example has label y_1
  - probability $p_2 = n_2/n$ the example has label y_2
  - …
  - probability $p_k = n_k/n$ the example has label y_k
- $p_1 + p_2 + \ldots + p_k = 1$
- The "outcome" of the draw is a random variable Y with probabilities $(p_1, p_2, \ldots, p_k)$
- What's the impurity/disorder of the examples? ➔

  What's the uncertainty of Y in a random drawing?

25

Entropy

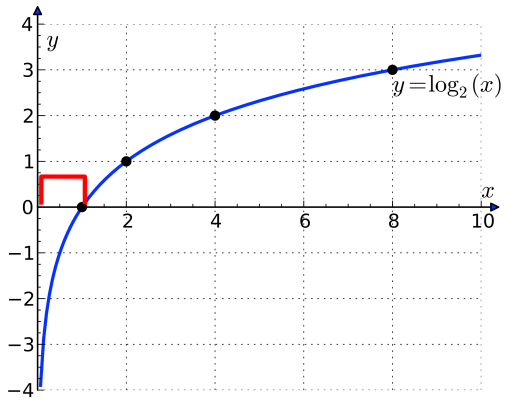
$$H(Y) = \sum_{i=1}^{k} -\Pr(Y = y_i) \log_2 \Pr(Y = y_i)$$

$$= \sum_{i=1}^{k} -p_i \log_2 p_i$$

Definition

Interpretation: The number of yes/no questions (in bits) needed on average to determine the value of Y in a random drawing

27



28

Entropy: $H(Y)$

- H measures the **information content** (in **bits**) associated with a set of examples
- $H(Y) \geq 0$

  where 0 is no information, and 1 is maximum information (for a 2-class *Y*)
- Bit
  - information needed to answer a yes/no question
  - a real-valued scalar, *not binary bits*

29

Information Extremes

- 2 classes: + and −
- Perfect Balance (Maximum Inhomogeneity):

  given $p_+ = p_- = ½$

  $H(Y) = H(½, ½) = -½ \log_2 ½ - ½ \log_2 ½$

  $= -½ (\log_2 1 - \log_2 2) - ½ (\log_2 1 - \log_2 2)$

  $= -½ (0 - 1) - ½ (0 - 1)$

  $= ½ + ½$

  $= 1$ (➔ the entropy is large)

  A histogram of the frequency distribution of values of *Y* is nearly flat
- "High Entropy" means *Y* is from a nearly ***uniform*** distribution

30

Information Extremes

- 2 classes: + and −
- Perfect Homogeneity:

  given $p_+ = 1$ and $p_- = 0$

  $H(Y) = H(1, 0) = -1 \log_2 1 - 0 \log_2 0$

  $= -1 (0) - ???$

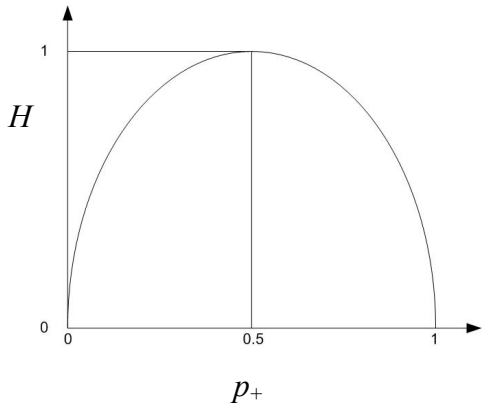
  $= 0 - 0$

  $= 0$ (➔ no information content)

  A histogram of the frequency distribution of values of *Y* has many low values and a few high values
- "Low Entropy" means *Y* is from a varied (peaks and valleys) distribution

31

Entropy



32

Entropy


Pr(head) = 0.5
Pr(tail) = 0.5
H = 1

Pr(head) = 0.51
Pr(tail) = 0.49
H = 0.9997

33

Entropy

When there are *k* classes, entropy is defined as

$$H(Y) = \sum_{i=1}^{k} -p_i \log_2 p_i$$

- p_i is the proportion of *Y* that belong to class *i*
- Log is still base 2 because entropy is a measure of expected encoding length measured in bits
- Maximum value of *H* is $\log_2 k$
  - For example, when $k = 3$, $0 \le H \le 1.58$

34

Example

- 3 classes: *Color* = R, G, B
- 10 examples: 3 R, 2 G, 5 B

- *H*(*Color*) = *H*(3/10, 2/10, 5/10)
  = (-3/10) $\log_2$ (3/10) + (-2/10) $\log_2$ (2/10) +
  (-5/10) $\log_2$ (5/10)
  = (-.3)(-1.74) + (-.2)(-2.32) + (-.5)(-1)
  = 1.486

35

Entropy

Entropy will be used as a *heuristic* for estimating the (relative) tree size rooted at a node given a set of examples associated with that node

- Small entropy predicts a small tree size
- Large entropy predicts a large tree size

36

Conditional Entropy

$$H(Y \mid X = v) = \sum_{i=1}^{k} -\Pr(Y = y_i \mid X = v) \log_2 \Pr(Y = y_i \mid X = v)$$

$$H(Y \mid X) = \sum_{v \in \text{values of } X} \Pr(X = v) H(Y \mid X = v)$$

called "Specific Conditional Entropy"

- *Y*: a label (or attribute)
- *X*: an attribute (i.e., feature or question)
- *v*: a value of the attribute *X*
- *Pr*(*Y*|*X*=*v*): conditional probability
- Textbook calls *H*(*Y*|*X*) the ***Remainder***(*X*)

38

Conditional Entropy: $H(Y \mid X)$

- Weighted sum of the entropies of each subset of the examples partitioned by the possible values of attribute *X*
- Expected value of entropy ***after*** splitting on *X*
- Measures the total "impurity," "disorder" or "inhomogeneity" of *all* the ***children nodes***
- $0 \leq H(Y \mid X) \leq 1$

39

Specific Conditional Entropy: *H*(*Y*|*X*=*v*)

X = College Major

Y = Likes "Gladiator"

Major	Likes G
Math	Yes
History	No
CS	Yes
Math	No
Math	No
CS	Yes
History	No
Math	Yes

My training data

Let's try to predict if someone likes a given movie when we know their major

41

Specific Conditional Entropy: $H(Y|X=v)$

X = College Major

Y = Likes "Gladiator"

Major	Likes G
Math	Yes
History	No
CS	Yes
Math	No
Math	No
CS	Yes
History	No
Math	Yes

Definition of Specific Conditional Entropy:

$H(Y \mid X=v)$ = entropy of Y for *only* those records in which X has value v

- $H(Y \mid X=History) = 0$
- $H(Y \mid X=CS) = 0$
- $H(Y \mid X=Math) =$

H(Y | X=Math) = -P(Y=Yes | X=Math) log P(Y=Yes | X=Math) – P(Y=No | X=Math) log P(Y=No | X=Math)
= -(2/4 * log (2/4)) – (2/4 * log (2/4))
= (-.5 * -1) + (-.5 * -1)
= 1

42

Conditional Entropy: $H(Y|X)$

X = College Major

Y = Likes "Gladiator"

Major	Likes G
Math	Yes
History	No
CS	Yes
Math	No
Math	No
CS	Yes
History	No
Math	Yes

Definition of Conditional Entropy:

$H(Y \mid X)$ = *total* specific conditional entropy of Y

= if you choose a record at random what will be the entropy of Y, conditioned on that row's value of X

= Expected number of bits to transmit Y if **both sides know the value of X**

$= \sum_j Pr(X=v_j)\, H(Y \mid X = v_j)$

43

Conditional Entropy

X = College Major

Y = Likes "Gladiator"

Major	Likes G
Math	Yes
History	No
CS	Yes
Math	No
Math	No
CS	Yes
History	No
Math	Yes

Definition of Conditional Entropy:

$H(Y \mid X)$ = total conditional entropy of Y

$= \sum_j Pr(X=v_j)\, H(Y \mid X = v_j)$

v_j	$Pr(X=v_j)$	$H(Y \mid X=v_j)$
Math	0.5	1
History	0.25	0
CS	0.25	0

$H(Y|X)$ = 0.5 * 1 + 0.25 * 0 + 0.25 * 0
= 0.5

44

Conditional Entropy: $H(Y|X=v)$

Y = College Major

X = Likes "Gladiator"

Major	Likes G
Math	Yes
History	No
CS	Yes
Math	No
Math	No
CS	Yes
History	No
Math	Yes

Note: We could have alternatively tried to predict someone's major given whether or not they like the movie Gladiator

Here the class variable, Y, is their major and the feature, X, is whether or not they like the movie "Gladiator"

45

Information Gain

- **Information gain**, or **mutual information**

$$I(Y;X) = H(Y) - H(Y \mid X)$$

- Measures the ***difference* in entropy** of a node and the entropy *remaining* after the node's examples are "**split**" between the children using a chosen attribute
- *I(Y; X)* means *I* must transmit *Y.* How many bits on average would it save me if both ends know *X*?
- Choose the attribute (i.e., feature or question) *X* that ***maximizes*** *I*(*Y*; *X*)
- Textbook calls *I*(*Y*; *X*) the ***Gain***(*X*)

46

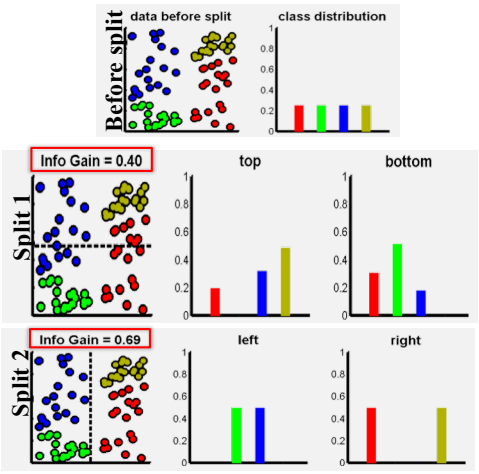
Using Information Gain to Select the Best Attribute

Goal: Construct a ***small*** decision tree that correctly classifies the training examples

- Why would high information gain be desirable?
  - means more of the examples are the *same class* in each child node
  - so, the decision trees rooted at each child that are needed to differentiate between the classes are likely to be small

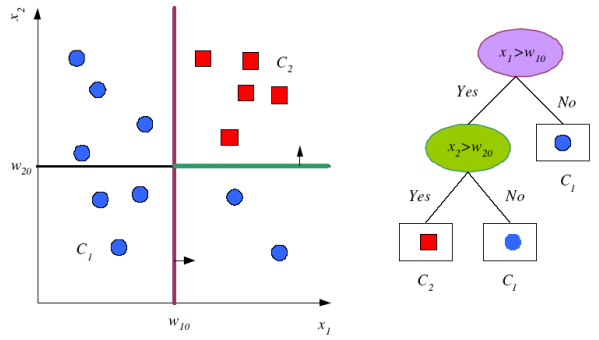
47

Using Information Gain



48

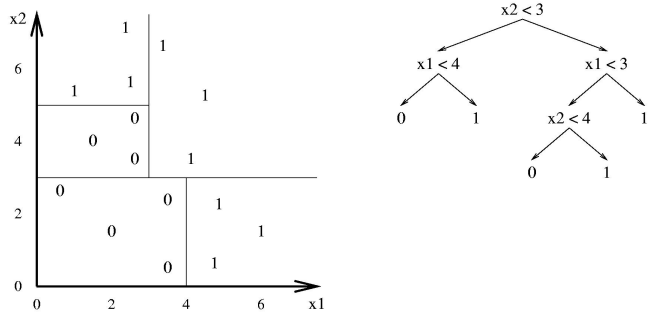
Decision Boundaries



49

Decision Tree Decision Boundaries

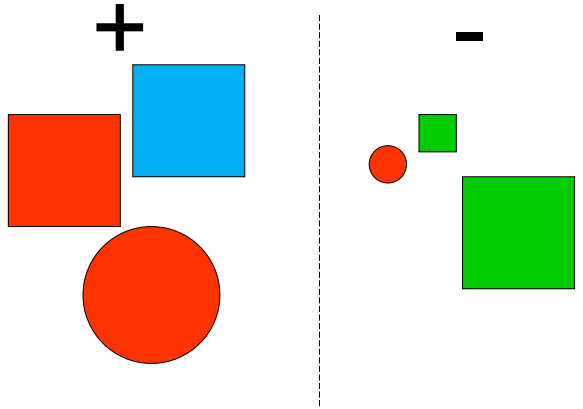
Decision trees divide the feature space into axis-parallel rectangles, and label each rectangle with one of the K classes.



50

Example

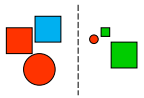
- Attributes: *Color*, *Shape*, *Size*
- What's the best attribute for the root?



51

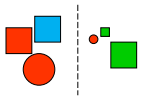
The Training Set

Example	*Color*	*Shape*	*Size*	*Class*
1	Red	Square	Big	+
2	Blue	Square	Big	+
3	Red	Circle	Big	+
4	Red	Circle	Small	−
5	Green	Square	Small	−
6	Green	Square	Big	−



52

Example	*Color*	*Shape*	*Size*	*Class*
1	Red	Square	Big	+
2	Blue	Square	Big	+
3	Red	Circle	Big	+
4	Red	Circle	Small	−
5	Green	Square	Small	−
6	Green	Square	Big	−



$H(Class) = H(3/6, 3/6) = 1$

$H(Class \mid Color) = 3/6\ H(2/3,1/3) + 2/6\ H(0/2, 2/2) + 1/6\ H(1/1, 0/1)$

3 out of 6 are red

2 of the red are +

2 out of 6 are green

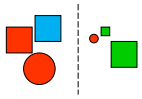
both green are −

1 out of 6 is blue

blue is +

53

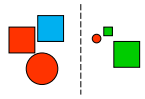
Example	*Color*	*Shape*	*Size*	*Class*
1	Red	Square	Big	+
2	Blue	Square	Big	+
3	Red	Circle	Big	+
4	Red	Circle	Small	–
5	Green	Square	Small	–
6	Green	Square	Big	–



$H(Class) = H(3/6, 3/6) = 1$
$H(Class \mid Color) = 3/6\ H(2/3,1/3) + 1/6\ H(1/1, 0/1) + 2/6\ H(0/2, 2/2)$
$I(Class; Color) = H(Class) - H(Class \mid Color) = 0.54$ bits

54

Example	*Color*	*Shape*	*Size*	*Class*
1	Red	Square	Big	+
2	Blue	Square	Big	+
3	Red	Circle	Big	+
4	Red	Circle	Small	–
5	Green	Square	Small	–
6	Green	Square	Big	–

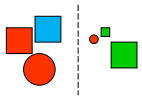


$H(Class) = H(3/6, 3/6) = 1$
$H(Class \mid Shape) = 4/6 * H(2/4, 2/4) + 2/6 * H(1/2,1/2) = 1$
$I(Class; Shape) = H(Class) - H(Class \mid Shape) = 0$ bits

Shape tells us nothing about *Class*!

55

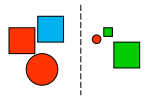
Example	*Color*	*Shape*	*Size*	*Class*
1	Red	Square	Big	+
2	Blue	Square	Big	+
3	Red	Circle	Big	+
4	Red	Circle	Small	–
5	Green	Square	Small	–
6	Green	Square	Big	–



$H(Class) = H(3/6, 3/6) = 1$
$H(Class \mid Size) = 4/6 * H(3/4, 1/4) + 2/6 * H(0/2, 2/2) = 0.54$
$I(Class; Size) = H(Class) - H(Class \mid Size) = 0.46$ bits

56

Example	*Color*	*Shape*	*Size*	*Class*
1	Red	Square	Big	+
2	Blue	Square	Big	+
3	Red	Circle	Big	+
4	Red	Circle	Small	–
5	Green	Square	Small	–
6	Green	Square	Big	–

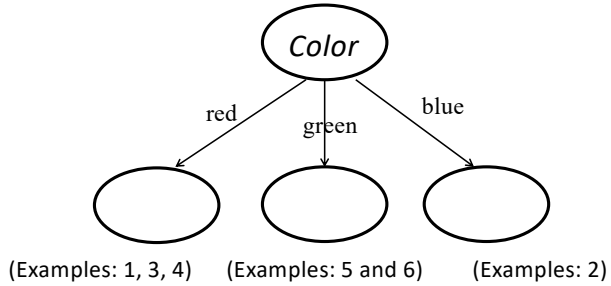


$I(Class; \mathbf{Color}) = H(Class) - H(Class \mid Color) = 0.54$ bits
$I(Class; \mathbf{Shape}) = H(Class) - H(Class \mid Shape) = 0$ bits
$I(Class; \mathbf{Size}) = H(Class) - H(Class \mid Size) = 0.46$ bits

➔ Select ***Color*** as the best attribute at the root

57

What's the Next Step?

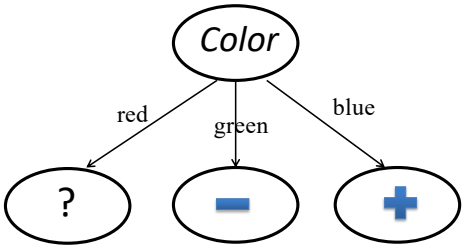


58

- Blue child is a leaf since it has only one example, which is +
  - Make its class +
- Green child is a leaf since both its examples are –
  - Make its class –
- Red child is *not* a leaf since its examples are 2 + and 1 -

59

What's the Next Step?



60

Which Attribute for Red Child?

- Compute *I*(*Class*; *Size*), *I*(*Class*; *Shape*), and *I*(*Class*, *Color*)
- *H*(*Class*) =

  H(2/3, 1/3) = (-2/3)log 2/3 + (-1/3)log 1/3 = 0.92

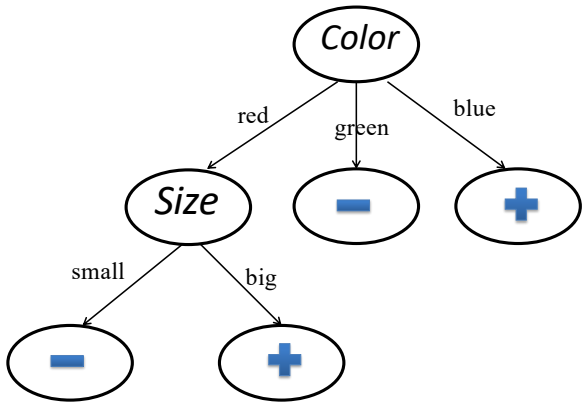
61

Which Attribute for Red Child?

- $H(Class|Size) = 2/3\ H(2/2, 0/2) + 1/3\ H(0/1, 1/1)$
  $= (2/3)(0) + (1/3)(0) = 0$
  So, $I(Class; Size) = 0.92 - 0 = 0.92$
- $H(Class|Shape) = 1/3\ H(1/1, 0/1) + 2/3\ H(½, ½)$
  $= (1/3)(0) + (2/3)(1) = 0.67$
  So, $I(Class; Shape) = 0.92 - 0.67 = 0.25$
- $H(Class|Color) = 3/3\ H(2/3, 1/3) + 0/3\ H(0/0, 0/0)$
  $= H(2/3, 1/3) = 0.92$
  So, $I(Class; Color) = 0$ Must be 0!
- So, ***Size*** is the best attribute

62

Final Decision Tree



63

Selecting the Best Attribute

The best attribute for a node is the attribute (of those candidates available for that node, which are all attributes ***except*** those on path back to the root) with:

Class variable — Attribute

Maximum Information Gain, $I(Y; X)$

or, equivalently,

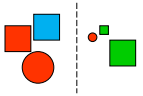
Minimum Conditional Entropy, $H(Y \mid X)$

since at a given node $H(Y)$ is fixed

64

The Training Set

Example	*Color*	*Shape*	*Size*	*Class*
1	Red	Square	Big	+
2	Blue	Square	Big	+
3	Red	Circle	Big	+
4	Red	Circle	Small	–
5	Green	Square	Small	–
6	Green	Square	Big	–



Now let *Color* be the "class" variable ➔ 3 class problem

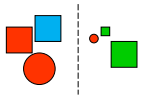
$H(Color) = ?$

$H(Color \mid Class) = ?$

65

The Training Set

Example	*Color*	*Shape*	*Size*	*Class*
1	Red	Square	Big	+
2	Blue	Square	Big	+
3	Red	Circle	Big	+
4	Red	Circle	Small	–
5	Green	Square	Small	–
6	Green	Square	Big	–



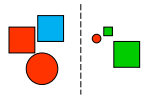
Now let *Color* be the "class" variable ➔ 3 class problem

$H(Color) = H(3/6, 2/6, 1/6)$
$= (-3/6)\log_2(3/6) + (-2/6)\log_2(2/6) + (-1/6)\log_2(1/6)$
$= 1.46$

66

The Training Set

Example	*Color*	*Shape*	*Size*	*Class*
1	Red	Square	Big	+
2	Blue	Square	Big	+
3	Red	Circle	Big	+
4	Red	Circle	Small	–
5	Green	Square	Small	–
6	Green	Square	Big	–



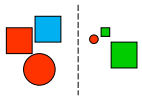
$H(Color \mid Class) = 3/6\ H(2/3, 0/3, 1/3) + 3/6\ H(1/3, 2/3, 0/3)$
$H(2/3, 0, 1/3) = (-.667)(-.584) + 0 + (-.333)(-1.586)$
$= .918$
$H(1/3, 2/3, 0) = .918$

So, $H(Color \mid Class) = 0.918$

67

The Training Set

Example	*Color*	*Shape*	*Size*	*Class*
1	Red	Square	Big	+
2	Blue	Square	Big	+
3	Red	Circle	Big	+
4	Red	Circle	Small	–
5	Green	Square	Small	–
6	Green	Square	Big	–



$I(Color; Class) = H(Color) - H(Color \mid Class)$
$= 1.96 - 0.918$
$= 1.04$ bits

68

Extensions to Decision Tree Learning: **Integer-Valued Attributes**

- In HW3, all attributes are integer-valued with values 1, …, 10
- Consider *all* possible thresholds, $t \le 1$ and $t > 1$; $t \le 2$ and $t > 2$; …, $t \le 9$ and $t > 9$
- For each attribute, find the best threshold, i.e., the threshold that gives the max InfoGain
- Then select the best attribute (one with max InfoGain for its best threshold)
- Create 2 children by splitting on the selected best attribute and associated threshold

69

Extensions to Decision Tree Learning: **Integer-Valued Attributes**

- An attribute selected at a node in the decision tree ***CAN*** be used again in the subtree rooted at the node containing this attribute, but using a *different threshold*
- Your implementation can simply try *all attributes and all associated thresholds* when creating each non-leaf node in the tree. Why?

70

Extensions to Decision Tree Learning: **Real-Valued Attributes**

- What if some of the attributes, x_1, x_2, …, x_k, are real-valued numerical attributes?
- Example: x_1 = height (in inches)
- Idea 1: Branch on each possible numerical value
  - Over-fragments the training data and is likely to "overfit"

71

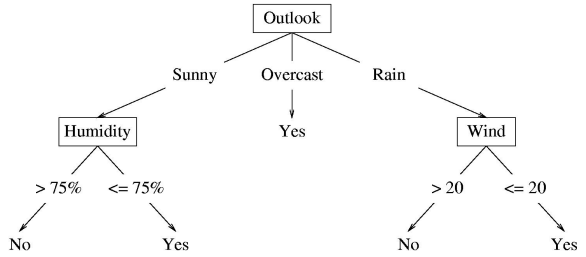
Extensions to Decision Tree Learning: **Real-Valued Attributes**

- What if some of the attributes, x_1, x_2, …, x_k, are real-valued?
- Example: x_1 = height (in inches)
- Idea 2: Use questions of the form: $\boldsymbol{x_1 > t_1}$? where t_1 is a threshold
  - How many thresholds?
    - 1 for each pair of consecutive examples that are classified *differently*
  - How to set threshold values?
    - Midpoint between 2 consecutive examples' values

72

Decision Tree Hypothesis Space

If the features are continuous, internal nodes may test the value of a feature against a threshold.



73

Example

5 training examples, 2 attributes (*X*1 and *X*2) and 2 classes (T and F)

	X1	X2	Class
x1	0.5	4.5	F
x2	2.2	1.5	T
x3	3.9	3.5	F
x4	2.1	1.9	T
x5	1.1	4	T

74

Example

Sort examples by values of selected attribute, *X*2

	X1	X2	Class
x2	2.2	1.5	T
x4	2.1	1.9	T
x3	3.9	3.5	F
x5	1.1	4	T
x1	0.5	4.5	F

75

Find pairs of *consecutive* examples that have *different* class labels, and define a candidate threshold as the average of these two examples' values for *X*2

$$\text{Candidate Thresholds} = \{\frac{1.9+3.5}{2} = 2.7, \frac{3.5+4}{2} = 3.75, \frac{4+4.5}{2} = 4.25\}$$

76

- Examples with $X2 \leq 2.7$

	X1	X2	Class
x2	2.2	1.5	T
x4	2.1	1.9	T

- Examples with $X2 > 2.7$

	X1	X2	Class
x3	3.9	3.5	F
x5	1.1	4	T
x1	0.5	4.5	F

77

Compute Info Gain for *X*2:2.7

$I(Class) = H(3/5, 2/5)$

$= -3/5 \log 3/5 - 2/5 \log 2/5$

$= 0.9709$

$H(Class \mid X2) = 2/5\ H(2/2, 0/2) + 3/5\ H(1/3, 2/3)$

$= (.4)(0) + (.6)[-1/3 \log 1/3 - 2/3 \log 2/3]$

$= 0.5509$

$I(\text{Class}; X2) = 0.9709 - 0.5509 = 0.4199$

78

- $I(Class; X2{:}2.7) = 0.4199$
- Similarly, compute

  $I(\text{Class}; X2{:}3.75)$

  $I(\text{Class}; X2{:}4.25)$
- Pick the threshold that has maximum Information Gain as the best threshold for *X*2

79

Decision-Tree-Learning Algorithm

buildtree(*examples*, *attributes*, *default-label*)
/* examples: a list of training examples
attributes: a set of candidate questions, e.g., "what's the value of attribute x_i?"
default-label: default label prediction, e.g., over-all majority vote */
if empty(*examples*) then return *default-label*
if (*examples* have same label *y*) then return *y*
if empty(*attributes*) then return majority vote in *examples*
q = best_attribute(*examples, attributes*)
tree = create-node with attribute *q*
foreach value *v* of attribute *q* do
    v-ex = subset of examples with *q* == *v*
    subtree = **buildtree**(*v-ex*, *attributes* - {*q*}, *majority-class*(*examples*))
    add arc from *tree* to *subtree*
return *tree*

only for categorical attributes

80

Decision Tree Applet

You are in a basketball pool, currently betting on the outcome of next week's basketball game between the MallRats and the Chinooks. You have to decide which team will win, then bet on that team.

http://www.cs.ualberta.ca/~aixplore/learning/DecisionTrees/index.html

81

Applications

- Credit-card companies and banks use DTs to decide whether to grant a card or loan
- Diagnose breast cancer
  - humans correct 65% of the time
  - decision tree classified 72% correct
- BP designed a decision tree for gas-oil separation for offshore oil platforms
- Cessna designed a flight controller using 90,000 examples and 20 attributes per example

82

Expressiveness of Decision Trees

- Assume all attributes are Boolean and all classes are Boolean (i.e., 2 classes)
- Which Boolean functions can be represented by decision trees?
- Answer: **All Boolean functions!**

Proof:

1. Take any Boolean function
2. Convert it into a truth table
3. Construct a decision tree in which each row of the truth table corresponds to one path through the decision tree from root to a leaf

84

Evaluating Performance

How might performance be evaluated?

- Speed of learner
- Speed of classifier
- Space requirements
- **Predictive accuracy of classifier**

85

Training Set Error

- For each example in the **training set**, use the decision tree to see what class it predicts

  What % of the examples does the decision tree's prediction *disagree* with the known *true* value?
- This quantity is called the ***Training Set Error***
  - The smaller the better
- But why are we doing learning anyway?
  - More important to assess how well the decision tree predicts output for ***future*** **data**

86

Test Set Error

- We hide some data away, called the **test set**, when we learn the decision tree
- After the tree is learned, check how well the tree predicts that held-back data: % classified incorrectly
- This is a good simulation of what happens when we try to predict future data
- Called the ***Test Set Error***

87

Evaluating Classifiers

- During training
  - Train a classifier from a training set (x_1,y_1), (x_2,y_2), …, (x_n,y_n)
- During testing
  - For new test data, x_{n+1}, …, x_{n+m}, your classifier generates predicted labels y'_{n+1}, …, y'_{n+m}
- Test set accuracy:
  - You need to know the true test labels: y_{n+1}, …, y_{n+m}
  - Test set **error**: $$err = \frac{1}{m}\sum_{i=n+1}^{n+m}(y_i \neq y_i')$$ 0-1 loss
  - Test set **accuracy**: $1 - err$

88

Evaluating Performance Accuracy

Use ***separate*** test examples to estimate accuracy!

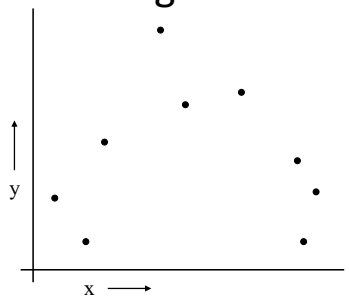
1. Randomly partition training examples
   TRAIN set (say ~80% of all training examples)
   TEST set (say ~20% of all training examples)
2. Generate decision tree using the TRAIN set
3. Use TEST set to evaluate accuracy
   `accuracy = #correct / #total`

89

- In HW3, Problem 2(a), use the provided "Tuning Set" containing 5 examples to compute the classification accuracy of the given decision tree:
  # correctly classified / 5
- In Problem 2(b), create 4 trees as described and then compute the accuracy for each using the given "Tuning Set"

90

The Overfitting Problem: A Regression Problem Example

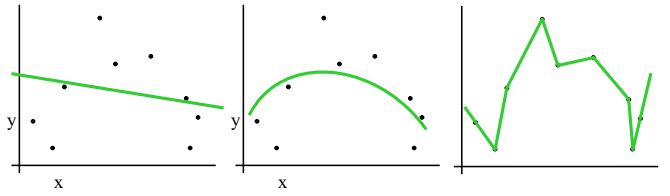


$y = f(x) + \text{noise}$

Can we learn f from this data?

92

Which is Best?



Linear f | Quadratic f | Piecewise Linear f

Fits the data best, including any noise!

93

The **Overfitting Problem** Example: Predicting U.S. Population

x=Year	*y*=Million
1900	75.99
1910	91.97
1920	105.71
1930	123.2
1940	131.67
1950	150.7
1960	179.32
1970	203.21
1980	226.51
1990	249.63
2000	281.42

- We have some training data (n=11)
- What will the population be in 2010?

94

Regression: Polynomial Fit

- The degree d (complexity of the model) is important

$$f(x) = c_d x^d + c_{d-1} x^{d-1} + \cdots + c_1 x + c_0$$

- Fit (i.e., learn) coefficients c_d, … c_0 to minimize Mean Squared Error (MSE) on training data

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - f(x_i))^2$$

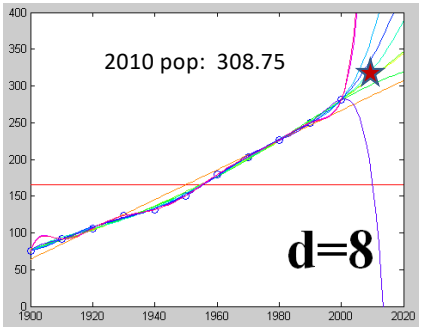
"Squared loss" common for regression

95

Overfitting

- As *d* increases, the MSE on the *training* data improves, but **prediction** on *test* data **worsens**

degree=0 MSE=4181.451643
degree=1 MSE=79.600506
degree=2 MSE=9.346899
degree=3 MSE=9.289570
degree=4 MSE=7.420147
degree=5 MSE=5.310130
degree=6 MSE=2.493168
degree=7 MSE=2.278311
degree=8 MSE=1.257978
degree=9 MSE=0.001433
degree=10 MSE=0.000000



96

Overfitting a Decision Tree

In general, **overfitting means finding "meaningless" regularity in training data**

Especially a problem with "noisy" data due to:

- Class associated with an example is wrong
  - May mean two examples have all the same attribute values but different classes
- Attribute values are incorrect because of errors getting or preprocessing the data
- Irrelevant attributes

97

Overfitting a Decision Tree: Example

Five attributes, all binary, are generated in all 32 possible combinations

Class ***y*** = copy of ***e***, except a random 25% of the records have ***y*** set to the *opposite* of ***e***

32 records

a	*b*	*c*	*d*	*e*	*y*
0	0	0	0	0	0
0	0	0	0	1	0
0	0	0	1	0	0
0	0	0	1	1	1
0	0	1	0	0	1
:	:	:	:	:	:
1	1	1	1	1	1

Training set

98

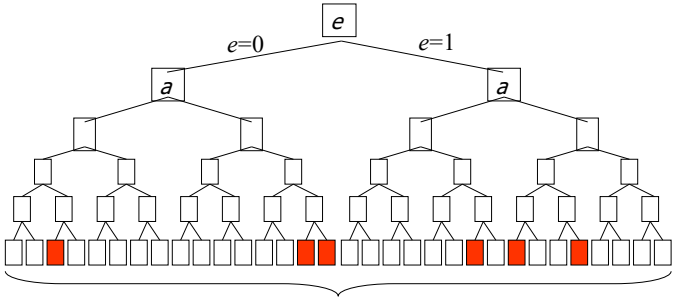
Overfitting a Decision Tree

- The **test set** is constructed similarly
  - ***y*** = ***e***, but 25% the time we corrupt it by ***y*** = **1 - *e***
  - Assume the corruption in training and test sets are *independent*
- The training and test sets are the same, except
  - Some ***y***'s are corrupted in training, but not in test
  - Some ***y***'s are corrupted in test, but not in training

99

Overfitting a Decision Tree

Suppose we build a full tree on the **training set**

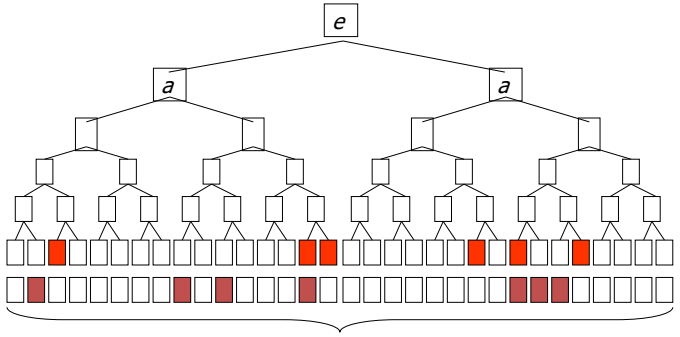


Training set accuracy = 100% (all leaf nodes contain exactly 1 example)
25% of these training leaf node labels will be corrupted (i.e., ≠***e***)

100

Overfiting a Decision Tree

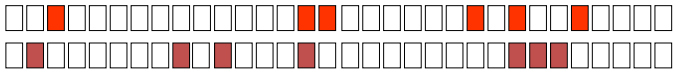
Next, classify the **test data** with the tree



25% of the test examples are corrupted – independent of training data

101

Overfitting a Decision Tree



On average:

- ¾ training data *uncorrupted*
  - ¾ of these are uncorrupted in test – correct labels
  - ¼ of these are corrupted in test – wrong
- ¼ training data *corrupted*
  - ¾ of these are uncorrupted in test – wrong
  - ¼ of these are also corrupted in test – correct labels
- **Test accuracy =** (¾ * ¾) + (¼ * ¼) = 5/8 = **62.5%**

102

Overfitting a Decision Tree

But if we *knew* ***a***,***b***,***c***,***d*** are **irrelevant attributes** and don't use them in the tree …

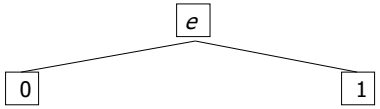
Pretend they don't exist

a	***b***	***c***	***d***	**e**	***y***
0	0	0	0	0	0
0	0	0	0	1	0
0	0	0	1	0	0
0	0	0	1	1	1
0	0	1	0	0	1
:	:	:	:	:	:
1	1	1	1	1	1

103

Overfitting a Decision Tree

The tree would be:



In training data, about ¾ ***y***'s are 0 here. Majority vote predicts ***y*** = 0

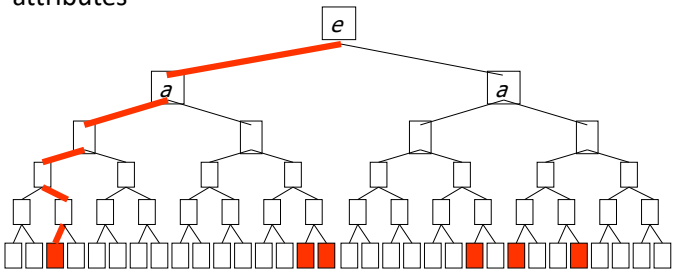
In training data, about ¾ ***y***'s are 1 here. Majority vote predicts ***y*** = 1

In test data, ¼ ***y***'s are different from ***e*** because they were corrupted, and ¾ ***y***'s will be correct, so **test set accuracy = 75%**, which is *better* than when using **meaningless attributes** (= 62.5%)

104

Overfitting a Decision Tree

The full tree ***overfit*** by learning meaningless (noise) attributes



105

Overfitting a Decision Tree

Overfitting

- meaningless regularity is found in the data
- irrelevant attributes confound the true, important, distinguishing features
- fix by **pruning** some nodes in the decision tree

106

Avoiding Overfitting: **Pruning**

Pruning with a ***Tuning Set***

1. Randomly split the training data into TRAIN and TUNE, say 70% and 30%
2. Build a full tree using *only* the TRAIN set
3. Prune the tree using the TUNE set

107

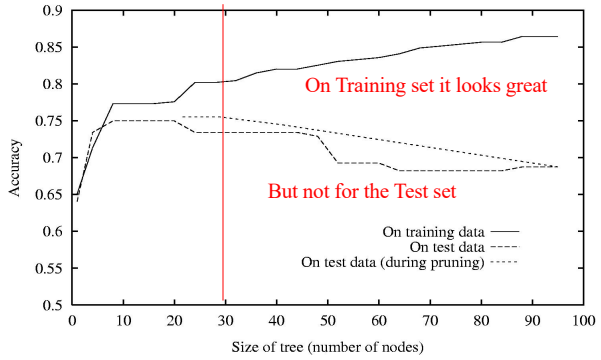
Pruning using a Greedy Algorithm

Prune(tree T, TUNE set)

1. Compute T's accuracy on TUNE; call it Acc(T)
2. For every non-leaf node N in T:
   a) New tree T_N = copy of T, but prune (delete) the subtree under N
   b) N becomes a leaf node in T_N. The class is the majority vote of TRAIN examples reaching N
   c) Acc(T_N) = T_N's accuracy on TUNE
3. Let T* be the tree (among the T_N's and T) with the largest Acc() Set T = T^* /* prune */
4. If no improvement then return T else goto Step 1

108

Effect of Reduced-Error Pruning



The tree is pruned back to the red line where it gives more accurate results on the Test set

109

Extensions to Decision Tree Learning: **Missing Data (i.e., Attributes without Values)**

Some possible approaches:

- During learning: replace with *most likely value*
- During learning: use *Unknown* as a value
- During classification: follow arcs for *all values* and weight each by the frequency of the examples along that arc

110

Extensions to Decision Tree Learning: **Interpreting Trees as Rules**

Generate a set of rules from a DT:

For each path from the root to a leaf

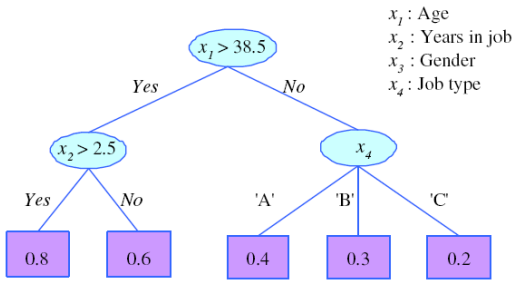
- the rule's antecedent is all the attribute values
- the consequent is the classification at leaf

```
if (Size=small && Suit=hearts) then class = +
```

- Constructing these rules yields an interpretation of the tree's meaning

111

Rule Extraction from Trees



R1: IF (age>38.5) AND (years-in-job>2.5) THEN y =0.8
R2: IF (age>38.5) AND (years-in-job≤2.5) THEN y =0.6
R3: IF (age≤38.5) AND (job-type='A') THEN y =0.4
R4: IF (age≤38.5) AND (job-type='B') THEN y =0.3
R5: IF (age≤38.5) AND (job-type='C') THEN y =0.2

112

Extensions to Decision Tree Learning

Setting parameters

- most learning algorithms require setting various parameters
- they must be set *without* looking at the Test data
- Common approach: use a ***Tuning Set***

  (aka ***Validation Set***)

113

Extensions to Decision Tree Learning

Use a **Tuning set** for setting parameters:

1. Partition given examples into TRAIN, TUNE and TEST sets
2. For each candidate parameter value, generate a decision tree using the TRAIN set
3. Use the TUNE set to evaluate error rates and determine which parameter value is best
4. Compute the final decision tree using the selected parameter values and *both* TRAIN and TUNE sets
5. Use TEST to compute performance accuracy

114

Experimental Evaluation of Performance

Test Set Method

1. Randomly choose say 30% of the data to be the Test set, and the remaining 70% is the Training set
2. Build a (decision tree) classifier using the Training set
3. Estimate future performance using the Test set

115

Test Set Method

- Wastes data because the Test set is *not used* to construct the best classifier
- If we don't have much data, our Test set might be lucky or unlucky in terms of what's in it, making the results on the Test set a "high variance" estimator of the real performance

116

Experimental Evaluation of Performance

Cross-Validation

Often use K = 3, 5 or 10

1. Divide all examples into K disjoint subsets $E = E_1, E_2, ..., E_K$
2. For each $i = 1, ..., K$
   - let TEST set = E_i and TRAIN set = $E - E_i$
   - build decision tree using TRAIN set
   - determine accuracy Acc_i using TEST set
3. Compute ***K*-fold cross-validation** estimate of performance = mean accuracy = $(Acc_1 + Acc_2 + ... + Acc_K)/K$

117

Experimental Evaluation of Performance

Leave-One-Out Cross Validation

For $i = 1$ to N do // N = number of examples

1. Let (x_i, y_i) be the i^{th} example
2. Remove (x_i, y_i) from the dataset
3. Train on the remaining N-1 examples
4. Compute accuracy on i^{th} example

- Accuracy = mean accuracy on all N runs
- Doesn't waste data but is expensive
- Use when you have a small dataset (< ~100)

118

Decision Trees Summary

- One of the most widely used learning methods in practice
- Can out-perform human experts in many applications

119

Decision Trees Summary

Strengths

- fast
- simple to implement
- well founded in information theory
- can convert result to a set of easily interpretable rules
- empirically valid in many commercial products
- handles noisy data
- scales well

120

Decision Trees Summary

Weaknesses

- Univariate splits/partitions using only one attribute at a time, which limits types of decision boundaries
- Each split is a hyperplane orthogonal to 1 axis (i.e., attribute)
- large decision trees may be hard to understand
- requires fixed-length feature vectors
- non-incremental (i.e., it's a batch method)

121

Combining Classifiers: Ensemble Methods

- **Aggregate the predictions of *multiple classifiers* with the goal of improving accuracy by *reducing the variance* of an estimated prediction function**
- "**Mixture of experts**"
- Combining multiple classifiers often produces *higher accuracy* than any individual classifier

122

Example: *Netflix Prize Competition*

October 2006

- Supervised learning task
  - Training data is a set of users and ratings (1,2,3,4,5 stars) those users have given to movies
  - Construct a classifier that given a user and an unrated movie, correctly classifies that movie as either 1, 2, 3, 4, or 5 stars
- $1 million prize for a 10% improvement over Netflix's then-current movie recommender/classifier (MSE = 0.9514)

Slide by T. Holloway

123

Just 3 weeks after it began, at least 40 teams could beat the Netflix classifier

Top teams showed about 5% improvement

Netflix Prize

me | Rules | Leaderboard | Register | Update | Submit | Download

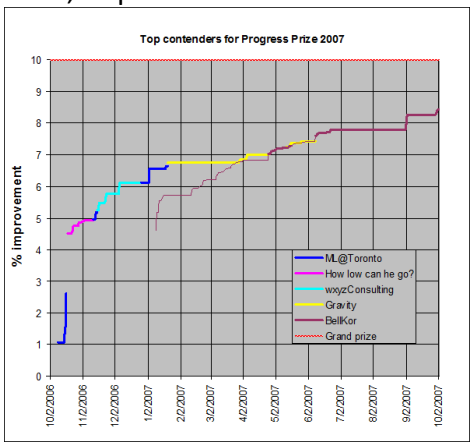
Leaderboard

Team Name	Best Score	% Improvement
No Grand Prize candidates yet	--	--
Grand Prize - RMSE <= 0.8563		
How low can he go?	0.9046	4.92
ML@UToronto A	0.9046	4.92
ssorkin	0.9089	4.47
wxyzconsulting.com	0.9103	4.32
The Thought Gang	0.9113	4.21
NIPS Reject	0.9118	4.16
simonfunk	0.9145	3.88
Bozo_The_Clown	0.9177	3.54
Elliptic Chaos	0.9179	3.52
datcracker	0.9183	3.48
Foreseer	0.9214	3.15
bsdfish	0.9229	3.00
Three Blind Mice	0.9234	2.94
Bocsimacko	0.9238	2.90
Remco	0.9252	2.75
karmatics	0.9301	2.24
Chapelator	0.9314	2.10
Flmod	0.9325	1.99
mthrox	0.9328	1.96

Slide by T. Holloway

124

However, improvement slowed…



from http://www.research.att.com/~volinsky/netflix/

Slide by T. Holloway

125

Ensemble methods to the rescue…

Rookies

"Thanks to Paul Harrison's collaboration, a simple mix of our solutions improved our result from 6.31 to 6.75"

--	No Progress Prize candidates yet	--	--
Progress Prize - RMSE <= 0.8625			
1	BellKor	0.8705	8.50
Progress Prize 2007 - RMSE = 0.8712 - Winning Team: KorBell			
2	KorBell	0.8712	8.43
3	When Gravity and Dinosaurs Unite	0.8717	8.38
4	Gravity	0.8743	8.10
5	basho	0.8746	8.07
6	Dinosaur Planet	0.8753	8.00
7	ML@UToronto A	0.8787	7.64
8	Arek Paterek	0.8789	7.62
9	NIPS Reject	0.8808	7.42
10	Just a guy in a garage	0.8834	7.15
11	Ensemble Experts	0.8841	7.07
12	mathematical capital	0.8844	7.04
13	HowLowCanHeGo2	0.8847	7.01
14	The Thought Gang	0.8849	6.99
15	Reel Ingenuity	0.8855	6.93
16	strudeltamale	0.8859	6.88
17	NIPS Submission	0.8861	6.86
18	Three Blind Mice	0.8869	6.78
19	TrainOnTest	0.8869	6.78
20	Geoff Dean	0.8869	6.78
21	Rookies	0.8872	6.75
22	Paul Harrison	0.8872	6.75
23	ATTEAM	0.8873	6.74
24	wxyzconsulting.com	0.8874	6.73
25	ICMLsubmission	0.8875	6.72
26	Efratko	0.8877	6.70
27	Kitty	0.8881	6.65
28	SecondaryResults	0.8884	6.62
29	Birgit Kraft	0.8885	6.61

Slide by T. Holloway

126

Arek Paterek

"My approach is to **combine the results of many methods** (also two-way interactions between them) using linear regression"

http://rainbow.mimuw.edu.pl/~ap/ap_kdd.pdf

--	No Progress Prize candidates yet	--	--
Progress Prize - RMSE <= 0.8625			
1	BellKor	0.8705	8.50
Progress Prize 2007 - RMSE = 0.8712 - Winning Team: KorBell			
2	KorBell	0.8712	8.43
3	When Gravity and Dinosaurs Unite	0.8717	8.38
4	Gravity	0.8743	8.10
5	basho	0.8746	8.07
6	Dinosaur Planet	0.8753	8.00
7	ML@UToronto A	0.8787	7.64
8	Arek Paterek	0.8789	7.62
9	NIPS Reject	0.8808	7.42
10	Just a guy in a garage	0.8834	7.15
11	Ensemble Experts	0.8841	7.07
12	mathematical capital	0.8844	7.04
13	HowLowCanHeGo2	0.8847	7.01
14	The Thought Gang	0.8849	6.99
15	Reel Ingenuity	0.8855	6.93
16	strudeltamale	0.8859	6.88
17	NIPS Submission	0.8861	6.86
18	Three Blind Mice	0.8869	6.78
19	TrainOnTest	0.8869	6.78
20	Geoff Dean	0.8869	6.78
21	Rookies	0.8872	6.75
22	Paul Harrison	0.8872	6.75
23	ATTEAM	0.8873	6.74
24	wxyzconsulting.com	0.8874	6.73
25	ICMLsubmission	0.8875	6.72
26	Efratko	0.8877	6.70
27	Kitty	0.8881	6.65
28	SecondaryResults	0.8884	6.62
29	Birgit Kraft	0.8885	6.61

Slide by T. Holloway

127

BellKor / KorBell

The winning team was from AT&T:

"Our final solution (RMSE=0.8712) consists of blending 107 individual results. "

An 8.5% improvement over Netflix

	No Progress Prize candidates yet		
Progress Prize - RMSE <= 0.8625			
1	BellKor	0.8705	8.50
Progress Prize 2007 - RMSE = 0.8712 - Winning Team: KorBell			
2	KorBell	0.8712	8.43
3	When Gravity and Dinosaurs Unite	0.8717	8.38
4	Gravity	0.8743	8.10
5	basho	0.8746	8.07
6	Dinosaur Planet	0.8753	8.00
7	ML@UToronto A	0.8787	7.64
8	Arek Paterek	0.8789	7.62
9	NIPS Reject	0.8808	7.42
10	Just a guy in a garage	0.8834	7.15
11	Ensemble Experts	0.8841	7.07
12	mathematical capital	0.8844	7.04
13	HowLowCanHeGo2	0.8847	7.01
14	The Thought Gang	0.8849	6.99
15	Reel Ingenuity	0.8855	6.93
16	strudeltamale	0.8859	6.88
17	NIPS Submission	0.8861	6.86
18	Three Blind Mice	0.8869	6.78
19	TrainOnTest	0.8869	6.78
20	Geoff Dean	0.8869	6.78
21	Rookies	0.8872	6.75
22	Paul Harrison	0.8872	6.75
23	ATTEAM	0.8873	6.74
24	wxyzconsulting.com	0.8874	6.73
25	ICMLsubmission	0.8875	6.72
26	Efratko	0.8877	6.70
27	Kitty	0.8881	6.65
28	SecondaryResults	0.8884	6.62
29	Birgit Kraft	0.8885	6.61

Slide by T. Holloway

128

Why Combine Classifiers?

- **Statistical**: When the amount of training data is small relative to size of the feature space, there are many possible classifiers that fit the data; "averaging" their results *reduces risk of picking a poor classifier*
- **Computational**: Small training set + local search means it's hard to find the "true" classifier; ensemble *simulates multiple random restarts* to obtain multiple classifiers
- **Representational**: True classifier may not be representable in the feature space of a single method, but some (weighted) combination of hypotheses *expands the space of representable functions*

129

How to Combine Classifiers?

Given a test example, classify it using *each* classifier and report as the output class the **majority** (for a 2-class problem) or **mode** classification (for *k*-class problems)

130

Intuition

Majority Vote Classifier

Suppose we have 5 completely *independent* classifiers

– If accuracy is 70% for each, combined accuracy is: $10(.7^3)(.3^2) + 5(.7^4)(.3) + (.7^5)$

$\binom{5}{3}$

- **83.7% majority vote accuracy**

Using 101 independent classifiers

- **99.9% majority vote accuracy**

Slide by T. Holloway

131

When is an Ensemble Better?

Necessary and sufficient condition for an ensemble to be *more accurate* than individual classifiers, is when each individual classifier is:

- **Accurate**: error rate is better than random guessing
- **Diverse**: Classifiers make errors on *different* examples, i.e., they are at least *somewhat uncorrelated*

132

Ensemble Strategies

Boosting

- Sequential production of classifiers, where each classifier is *dependent on the previous one*
- Make examples misclassified by current classifier *more* important in the next classifier

Bagging

- Create classifiers using *different training sets*, where each training set is created by "bootstrapping," i.e., drawing examples (with replacement) from all possible training examples

Slide by T. Holloway

133

Bagging

- Given *N* training examples, generate separate training sets by choosing *n* examples *with replacement* from all *N* examples
- Called "**taking a bootstrap sample**" or "randomizing the training set"
- Construct a separate classifier using the *n* examples in *each* training set

134

Bagging Example (Opitz, 1999)

$N = 8, n = 8$

Original	1	2	3	4	5	6	7	8
Training set 1	2	7	8	3	7	6	3	1
Training set 2	7	8	5	6	4	2	7	1
Training set 3	3	6	2	7	5	6	2	2
Training set 4	4	5	1	4	6	4	3	8

135

Bagging with Decision Trees

- For each training set, build a separate decision tree
- Take majority/mode vote from all the decision trees to determine the output classification of a given testing example

136

Random Forests

aka Decision Forest, Classification Forest

2 Main Ideas:

1. **Bagging**: Use random samples of the training examples to construct each classifier
2. **Randomized Node Optimization**: Each time a node is split, only a *randomly chosen subset of the attributes is used*

137

Random Forests

For each tree,

1. Build a training set by choosing *n* times with replacement from all *N* available training examples
2. At each node of decision tree during construction, choose a *random subset* of *m* ***attributes*** from the total number, *M*, of possible attributes (*m* << *M*)
3. Select the best attribute at node using Max-Gain

- No tree pruning
- Doesn't overfit!

138

Test Set Classification Error (%)

Data set	Adaboost	Selection	Forest-RI single input	One tree
Glass	22.0	20.6	21.2	36.9
Breast cancer	3.2	2.9	2.7	6.3
Diabetes	26.6	24.2	24.3	33.1
Sonar	15.6	15.9	18.0	31.7
Vowel	4.1	3.4	3.3	30.4
Ionosphere	6.4	7.1	7.5	12.7
Vehicle	23.2	25.8	26.4	33.1
German credit	23.5	24.4	26.2	33.3
Image	1.6	2.1	2.7	6.4
Ecoli	14.8	12.8	13.0	24.5
Votes	4.8	4.1	4.6	7.4
Liver	30.7	25.1	24.7	40.6
Letters	3.4	3.5	4.7	19.8
Sat-images	8.8	8.6	10.5	17.2
Zip-code	6.2	6.3	7.8	20.6
Waveform	17.8	17.2	17.3	34.0
Twonorm	4.9	3.9	3.9	24.7
Threenorm	18.8	17.5	17.5	38.4
Ringnorm	6.9	4.9	4.9	25.7

~5 attributes 1 attribute

100 trees in forest

Breiman, Leo (2001). "Random Forests". Machine Learning 45 (1): 5-32

139

Classification Forests in Practice

Microsoft Kinect

J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake, **Real-Time Human Pose Recognition in Parts from a Single Depth Image**, *Proc. IEEE CVPR*, June 2011

140

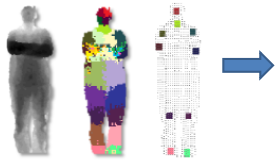
Kinect for Xbox One

- aka "Kinect 2.0" (2013)
- RGB-D Camera: Time-of-Flight Camera + Color Camera
- 30 fps
- Depth resolution: 2.5cm at 4m measured at ~300,000 points

141

What the Kinect Does

Compute Depth Image

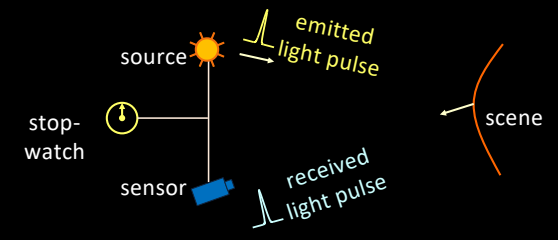


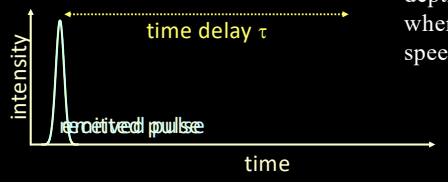
Application (e.g., games)

Estimate body parts and joint poses

142

Time-of-Flight Depth Sensing



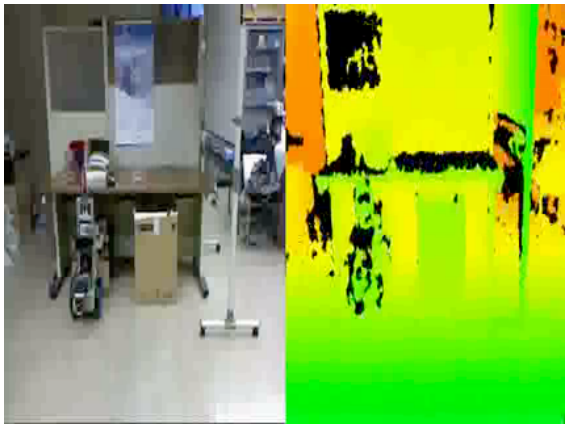


depth = c / 2t, where c = speed of light

[Koechner, 1968]

143

Kinect RGB-D Camera



144

Goal: Estimate Pose from Depth Image

Step 1. Find body parts
Step 2. Compute joint positions

RGB | Depth | Part Label Map | Joint Positions

145

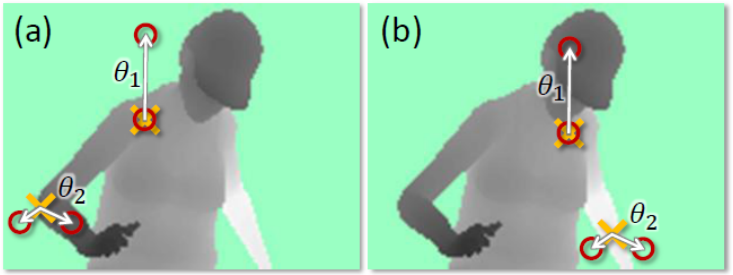
Finding Body Parts

- What should we use for **features**?
  Difference of depths at 2 pixels
- What should we use for a **classifier**?
  Random Forest

148

Features

Difference of depths at two pixels

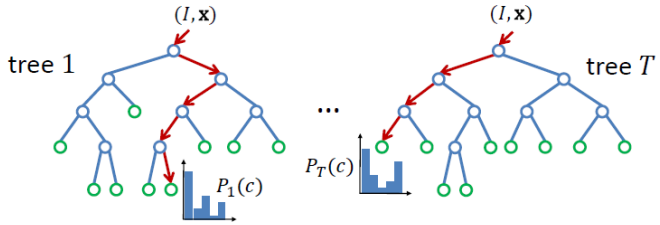


θ = (**u**, **v**) is the offset to second pixel from first pixel

149

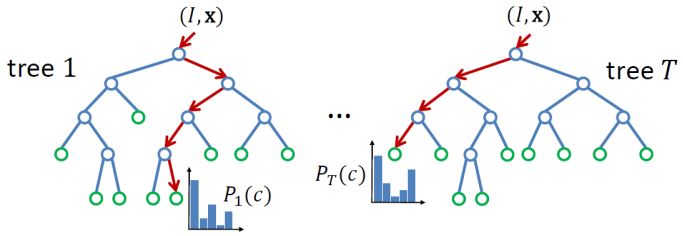
Part Classification with Random Forests

- **Random Forest**: collection of independently-trained binary **decision trees**
- Each tree is a classifier that predicts the likelihood of a pixel **x** belonging to body part class *c* in image *I*
  - Non-leaf node corresponds to a thresholded feature
  - Leaf node corresponds to a conjunction of several features
  - At leaf node store learned distribution $P(c|I, \mathbf{x})$



152

Learning Phase



1. For each tree, pick a randomly sampled subset of training data
2. Randomly choose a subset of features and thresholds at each node
3. Pick the feature and threshold that give the largest information gain
4. Recurse until a certain accuracy is achieved or depth-bound reached

153

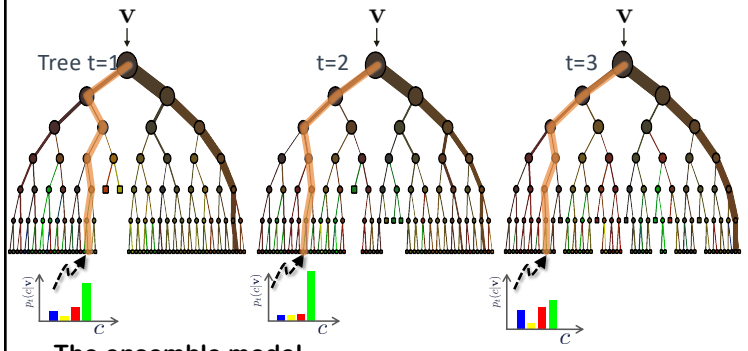
Classification

Classify each pixel **x** in image *I* using *all T* decision trees and ***average*** the results at the leaves:

$$P(c|I, \mathbf{x}) = \frac{1}{T} \sum_{t=1}^{T} P_t(c|I, \mathbf{x})$$

154

Random Forest: An Ensemble Model



155

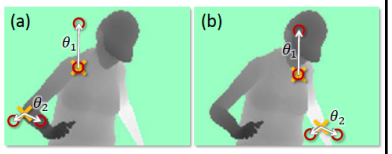
Implementation

- 31 body parts (i.e., 31 classes)
- 3 trees (max depth 20)
- 300,000 training images per tree randomly selected from 1M training images
- 2,000 training example pixels per image
- 2,000 candidate features
- 50 candidate thresholds per feature
- Decision forest constructed in 1 day on 1,000 core cluster

156

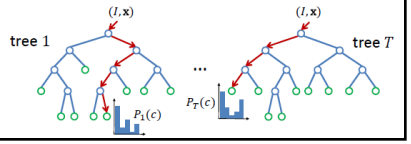
Basic Learning Approach

- Very simple features



- Lots of data

- Flexible classifier



157

Lots of Training Data

- Capture and sample 500K motion capture frames of people kicking, driving, dancing, etc.
- Get 3D models for 15 bodies with a variety of weights, heights, etc.
- Synthesize motion capture data for all 15 body types

158

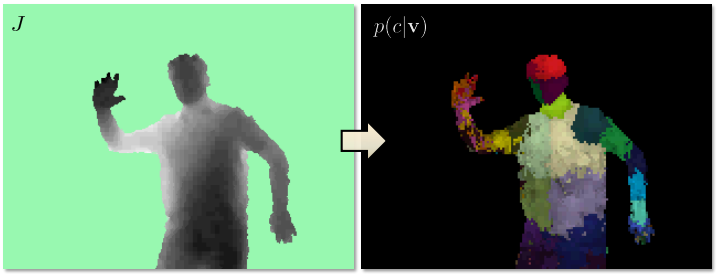
Synthetic Data for Training and Testing

160

Results

162

Body Tracking in Microsoft Kinect for XBox 360



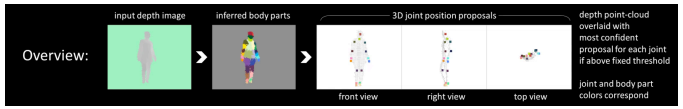
Input depth image (bg removed) Classified body parts

(2 videos here)

163

Results

164



165

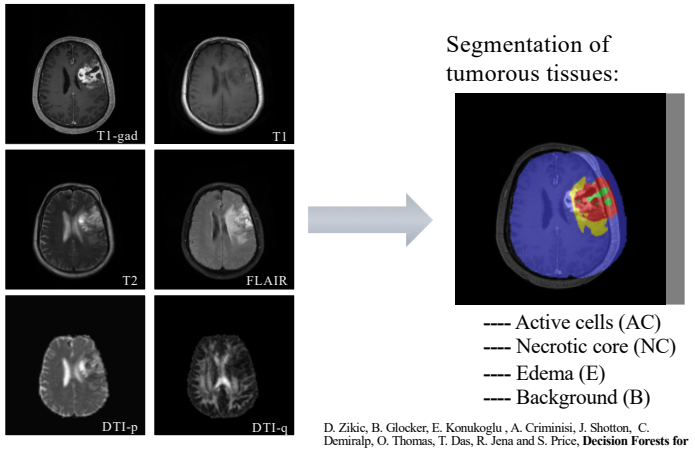
Classification Forests in Practice

Automatic Segmentation of Glioblastoma Brain Tumors

D. Zikic, B. Glocker, E. Konukoglu , A. Criminisi, J. Shotton, C. Demiralp, O. M. Thomas, T. Das, R. Jena and S. J. Price, **Decision Forests for Tissue-specific Segmentation of High-grade Gliomas in Multi-channel MR**, *Proc. MICCAI*, 2012

166

Segmentation of Brain Tumors



3D MRI input data

D. Zikic, B. Glocker, E. Konukoglu , A. Criminisi, J. Shotton, C. Demiralp, O. Thomas, T. Das, R. Jena and S. Price, **Decision Forests for Tissue-specific Segmentation of High-grade Gliomas in Multi-Channel MR,** *Proc. MICCAI,* 2012

167

Random Forest Summary

- Advantages
  - One of the most accurate learning algorithms
  - Efficient
  - Can handle thousands of attributes
- Disadvantages
  - Difficult to interpret (compared to decision trees)

173