Computer Vision

CSE 455
Learning I

Linda Shapiro
Professor of Computer Science & Engineering
Professor of Electrical & Computer Engineering
Learning

- AI/Vision systems are complex and may have many parameters.
- It is impractical and often impossible to encode all the knowledge a system needs.
- Different types of data may require very different parameters.
- Instead of trying to hard code all the knowledge, it makes sense to learn it.
Learning from Observations

- **Supervised Learning** – learn a function from a set of training examples which are preclassified feature vectors.

<table>
<thead>
<tr>
<th>feature vector</th>
<th>class</th>
</tr>
</thead>
<tbody>
<tr>
<td>(shape, color)</td>
<td></td>
</tr>
<tr>
<td>(square, red)</td>
<td>I</td>
</tr>
<tr>
<td>(square, blue)</td>
<td>I</td>
</tr>
<tr>
<td>(circle, red)</td>
<td>II</td>
</tr>
<tr>
<td>(circle blue)</td>
<td>II</td>
</tr>
<tr>
<td>(triangle, red)</td>
<td>I</td>
</tr>
<tr>
<td>(triangle, green)</td>
<td>I</td>
</tr>
<tr>
<td>(ellipse, blue)</td>
<td>II</td>
</tr>
<tr>
<td>(ellipse, red)</td>
<td>II</td>
</tr>
</tbody>
</table>

Given a previously unseen feature vector, what is the rule that tells us if it is in class I or class II?

- (circle, green)  
- (triangle, blue)
%Training set of Calenouria and Dorenouria
@DATA
0,1,1,0,0,0,0,0,0,1,1,2,3,0,1,2,0,0,0,0,0,0,0,0,0,1,0,0,1,
0,2,0,0,0,0,1,1,1,0,1,8,0,7,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,
3,3,4,0,2,1,0,1,1,1,0,0,0,0,1,0,0,1,1,cal 0,1,0,0,0,1,0,0,0,4,1,2
,2,0,1,0,0,0,0,1,0,0,3,0,2,0,0,1,1,0,0,1,0,0,0,1,0,1,6,1,8,2,0,0,
0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,1,2,0,5,0,0,0,0,0,0,0,0,0,1,3,0,0,0,0,0,0,0,0,cal
0,0,1,0,1,0,0,1,0,1,0,0,1,0,3,0,1,0,0,2,0,0,0,0,1,3,0,0,0,0,0,0,0,0,1,0,
2,0,2,0,1,8,0,5,0,1,0,1,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,2,2,0,0,3,0,0,2,1,1,
5,0,0,0,2,1,3,2,0,1,0,0,cal 0,0,0,0,0,0,0,0,0,2,0,0,1,2,0,1,1,0,0,0,1
,0,0,0,0,0,0,0,0,1,0,0,0,0,1,0,0,3,0,0,0,4,1,8,0,0,0,0,1,0,0,0,0,0,0,0,1,0,1
,0,1,0,0,0,0,0,0,0,4,2,0,2,1,1,2,1,1,0,0,0,0,2,0,0,2,2,cal
...
Learning from Observations

- **Unsupervised Learning** – No classes are given. The idea is to find patterns in the data. This generally involves *clustering*.

- **Reinforcement Learning** – learn from feedback after a decision is made.
Topics to Cover

• Inductive Learning
  – decision trees
  – ensembles
  – neural nets
  – kernel machines

• Ensembles

• The Fergus Algorithm with Kadir’s interest operator
<table>
<thead>
<tr>
<th>Decision Trees</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Theory is well-understood.</td>
</tr>
<tr>
<td>• Often used in pattern recognition problems.</td>
</tr>
<tr>
<td>• Has the nice property that you can easily understand the decision rule it has learned.</td>
</tr>
</tbody>
</table>
If the features are continuous, internal nodes may test the value of a feature against a threshold.

**Decision Tree Hypothesis Space**

Classic ML example: decision tree for “Shall I play tennis today?”

from Tom Mitchell’s ML book
The Stonefly Problem

Calenouria Dorenouria
Imaging Setup at OSU

Transportation Apparatus

Imaging Setup
Original Image Analysis

Feature vector: histogram of SIFT descriptors in a bag of words type of approach.
A Real Decision Tree (WEKA)

```
part23 < 0.5
| part29 < 3.5
| | part34 < 0.5
| | | part8 < 2.5
| | | | part2 < 0.5
| | | | | part63 < 3.5
| | | | | | part20 < 1.5 : dor (53/12) [25/8]
| | | | | | part20 >= 1.5
| | | | | | | part37 < 2.5 : cal (6/0) [5/2]
| | | | | | | part37 >= 2.5 : dor (3/1) [2/0]
| | | | | | | part63 >= 3.5 : dor (14/0) [3/0]
| | | | | | part2 >= 0.5 : cal (21/8) [10/4]
| | | | | part8 >= 2.5 : dor (14/0) [14/0]
| | | | part34 >= 0.5 : cal (38/12) [18/4]
| | | part29 >= 3.5 : dor (32/0) [10/2]
| | part23 >= 0.5
| | | part29 < 7.5 : cal (66/8) [35/12]
| | | part29 >= 7.5
| | | | part24 < 5.5 : dor (9/0) [4/0]
| | | | part24 >= 5.5 : cal (4/0) [4/0]
```
Evaluation

Correctly Classified Instances 281  73.5602 %
Incorrectly Classified Instances 101  26.4398 %
Kappa statistic 0.4718
Mean absolute error 0.3493
Root mean squared error 0.4545
Relative absolute error 69.973 %
Root relative squared error 90.7886 %
Total Number of Instances 382

=== Detailed Accuracy By Class ===

<table>
<thead>
<tr>
<th>Class</th>
<th>TP Rate</th>
<th>FP Rate</th>
<th>Precision</th>
<th>Recall</th>
<th>F-Measure</th>
<th>ROC Area</th>
</tr>
</thead>
<tbody>
<tr>
<td>cal</td>
<td>0.77</td>
<td>0.297</td>
<td>0.713</td>
<td>0.77</td>
<td>0.74</td>
<td>0.747</td>
</tr>
<tr>
<td>dor</td>
<td>0.703</td>
<td>0.23</td>
<td>0.761</td>
<td>0.703</td>
<td>0.731</td>
<td>0.747</td>
</tr>
<tr>
<td>Wg Avg.</td>
<td>0.736</td>
<td>0.263</td>
<td>0.737</td>
<td>0.736</td>
<td>0.735</td>
<td>0.747</td>
</tr>
</tbody>
</table>

=== Confusion Matrix ===

a  b  <-- classified as
144 43 | a = cal
58 137 | b = dor

Precision = TP/(TP+FP)
Recall = TP/(TP+FN)
F-Measure = 2 x Precision x Recall
             ---------------------------------------
                             Precision + Recall
Properties of Decision Trees

- They divide the decision space into axis parallel rectangles and label each rectangle as one of the k classes.
- They can represent Boolean functions.
- They are variable size and deterministic.
- They can represent discrete or continuous parameters.
- They can be learned from training data.
Learning Algorithm for Decision Trees

Growtree(S) \ /* Binary version */\n  if (y==0 for all (x,y) in S) return newleaf(0)
  else if (y==1 for all (x,y) in S) return newleaf(1)
  else
    choose best attribute \( x_j \)
    \( S_0 = (x,y) \) with \( x_j = 0 \)
    \( S_1 = (x,y) \) with \( x_j = 1 \)
    return new node(\( x_j \), Growtree(S_0), Growtree(S_1))

How do we choose the best attribute?
What should that attribute do for us?
Shall I play tennis today?
Which attribute should be selected?

“training data”

- **outlook**
  - sunny
    - yes
      - yes
      - yes
      - yes
      - yes
    - no
    - yes
    - yes
    - yes
    - yes
  - overcast
    - yes
    - yes
    - yes
    - yes
    - yes
    - no
    - no
  - rainy
    - yes
    - yes
    - yes
    - yes
    - no
    - no

- **windy**
  - false
    - yes
    - yes
    - yes
    - yes
    - yes
    - no
    - no
  - true
    - yes
    - yes
    - yes
    - yes
    - yes
    - no
    - no

- **humidity**
  - high
    - yes
    - yes
    - yes
    - yes
    - yes
    - no
    - no
  - normal
    - yes
    - yes
    - yes
    - yes
    - yes
    - yes
    - no
    - no

- **temperature**
  - hot
    - yes
    - yes
    - yes
    - yes
    - yes
    - no
    - no
  - mild
    - yes
    - yes
    - yes
    - yes
    - yes
    - yes
    - yes
    - yes
  - cool
    - yes
    - yes
    - yes
    - yes
    - yes
    - yes
    - no
    - no
Criterion for attribute selection

• Which is the best attribute?
  – The one that will result in the smallest tree
  – **Heuristic:** choose the attribute that produces the “purest” nodes

• Need a good measure of purity!
  – Maximal when?
  – Minimal when?
Information Gain

Which test is more informative?

Split over whether Balance exceeds 50K

- Less or equal 50K
- Over 50K

Split over whether applicant is employed

- Unemployed
- Employed
Information Gain

Impurity/Entropy (informal)

– Measures the level of impurity in a group of examples
Impurity

Very impure group

Less impure

Minimum impurity
Entropy: a common way to measure impurity

- Entropy = \( \sum_{i} - p_i \log_2 p_i \)
  
  \( p_i \) is the probability of class \( i \)
  
  Compute it as the proportion of class \( i \) in the set.

16/30 are green circles; 14/30 are pink crosses

\[ \log_2(16/30) = -0.9; \quad \log_2(14/30) = -1.1 \]

Entropy = \(- (16/30)(-0.9) - (14/30)(-1.1) = 0.99 \)

- Entropy comes from information theory. The higher the entropy the more the information content.

What does that mean for learning from examples?
2-Class Cases:

• What is the entropy of a group in which all examples belong to the same class?
  – entropy = -1 \log_2 1 = 0
  
  not a good training set for learning

• What is the entropy of a group with 50% in either class?
  – entropy = -0.5 \log_2 0.5 - 0.5 \log_2 0.5 = 1
  
  good training set for learning
Information Gain

- We want to determine which attribute in a given set of training feature vectors is most useful for discriminating between the classes to be learned.

- Information gain tells us how important a given attribute of the feature vectors is.

- We will use it to decide the ordering of attributes in the nodes of a decision tree.
Calculating Information Gain

Information Gain = \( \text{entropy(parent)} - [\text{average entropy(children)}] \)

\[ \text{parent entropy} = -\left( \frac{14}{30} \cdot \log_2 \frac{14}{30} \right) - \left( \frac{16}{30} \cdot \log_2 \frac{16}{30} \right) = 0.996 \]

\[ \text{child entropy} = -\left( \frac{13}{17} \cdot \log_2 \frac{13}{17} \right) - \left( \frac{4}{17} \cdot \log_2 \frac{4}{17} \right) = 0.787 \]

Entire population (30 instances)

(Weighted) Average Entropy of Children = \( \left( \frac{17}{30} \cdot 0.787 \right) + \left( \frac{13}{30} \cdot 0.391 \right) = 0.615 \)

Information Gain = \( 0.996 - 0.615 = 0.38 \) for this split
Entropy-Based Automatic Decision Tree Construction

Training Set $S$

$x_1 = (f_{11}, f_{12}, \ldots, f_{1m})$

$x_2 = (f_{21}, f_{22}, f_{2m})$

. . .

$x_n = (f_{n1}, f_{n2}, f_{2m})$

Node 1

What feature should be used?

What values?

Quinlan suggested information gain in his ID3 system and later the gain ratio, both based on entropy.
Using Information Gain to Construct a Decision Tree

1. Choose the attribute $A$ with highest information gain for the full training set at the root of the tree.

2. Construct child nodes for each value of $A$. Each has an associated subset of vectors in which $A$ has a particular value.

3. Repeat recursively till when?
Simple Example

Training Set: 3 features and 2 classes

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>Z</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>I</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>I</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>II</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>II</td>
</tr>
</tbody>
</table>

How would you distinguish class I from class II?
\[
\begin{array}{cccc}
X & Y & Z & C \\
1 & 1 & 1 & I \\
1 & 1 & 0 & I \\
0 & 0 & 1 & \text{II} \\
1 & 0 & 0 & \text{II} \\
\end{array}
\]

\[E_{\text{parent}} = 1\]

Split on attribute X

If X is the best attribute, this node would be further split.

\[
\begin{align*}
E_{\text{child}_1} &= -(1/3)\log_2(1/3) - (2/3)\log_2(2/3) \\
&= .5284 + .39 \\
&= .9184 \\
\end{align*}
\]

\[
\begin{align*}
E_{\text{child}_2} &= 0 \\
\end{align*}
\]

\[
\text{GAIN} = 1 - (3/4)(.9184) - (1/4)(0) = .3112
\]
<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>Z</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>I</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>I</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>II</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>II</td>
</tr>
</tbody>
</table>

$E_{\text{parent}} = 1$

Split on attribute $Y$

$\text{GAIN} = 1 - \left(\frac{1}{2}\right) 0 - \left(\frac{1}{2}\right) 0 = 1$; BEST ONE
<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>Z</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>I</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>I</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>II</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>II</td>
</tr>
</tbody>
</table>

\[ E_{\text{parent}} = 1 \]

Split on attribute Z

\[ \text{GAIN} = 1 - \left( \frac{1}{2} \right) \left( 1 \right) - \left( \frac{1}{2} \right) \left( 0 \right) = 0 \]

ie. NO GAIN; WORST

\[ E_{\text{child}1} = 1 \]

\[ E_{\text{child}2} = 1 \]
feature vector
(square, red) | class | I
(square, blue) | I
(circle, red) | II
(circle blue) | II
(triangle, red) | I
(triangle, green) | I
(ellipse, blue) | II
(ellipse, red) | II

Try the shape feature

Entropy?

GAIN?
Try the color feature

```
<table>
<thead>
<tr>
<th>feature vector</th>
<th>class</th>
</tr>
</thead>
<tbody>
<tr>
<td>square, red</td>
<td>I</td>
</tr>
<tr>
<td>square, blue</td>
<td>I</td>
</tr>
<tr>
<td>circle, red</td>
<td>II</td>
</tr>
<tr>
<td>circle blue</td>
<td>II</td>
</tr>
<tr>
<td>triangle, red</td>
<td>I</td>
</tr>
<tr>
<td>triangle, green</td>
<td>I</td>
</tr>
<tr>
<td>ellipse, blue</td>
<td>II</td>
</tr>
<tr>
<td>ellipse, red</td>
<td>II</td>
</tr>
</tbody>
</table>
```

Many-Valued Features

- Your features might have a large number of discrete values.
  Example: pixels in an image have (R,G,B) which are each integers between 0 and 255.
- Your features might have continuous values.
  Example: from pixel values, we compute gradient magnitude, a continuous feature.
One Solution to Both

- We often group the values into bins

What if we want it to be a binary decision at each node?

Use thresholds in a binary tree.
Training and Testing

- Divide data into a **training set** and a separate **testing set**.
- Construct the decision tree using the training set only.
- Test the decision tree on the training set to see how it’s doing.
- Test the decision tree on the **testing set** to report its real performance.
Measuring Performance

• Given a test set of labeled feature vectors e.g. (square,red) I
• Run each feature vector through the decision tree
• Suppose the decision tree says it belongs to class X and the real label is Y
• If (X=Y) that’s a correct classification
• If (X<>Y) that’s an error
Measuring Performance

- In a 2-class problem, where the classes are positive or negative (ie. for cancer)
  - # true positives TP
  - # true negatives TN
  - # false positives FP
  - # false negatives FN
- **Accuracy** = #correct / #total = \( \frac{TP + TN}{TP + TN + FP + FN} \)
- **Precision** = \( \frac{TP}{TP + FP} \)
  
  How many of the ones you said were cancer really were cancer?
- **Recall** = \( \frac{TP}{TP + FN} \)
  
  How many of the ones who had cancer did you call cancer?
More Measures

• F-Measure = $2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$

Gives us a single number to represent both precision and recall.

In medicine:

• Sensitivity = $\frac{\text{TP}}{\text{TP} + \text{FN}} = \text{Recall}$

The sensitivity of a test is the proportion of people who have a disease who test positive for it.

• Specificity = $\frac{\text{TN}}{\text{TN} + \text{FP}}$

The specificity of a test is the number of people who DON’T have a disease who test negative for it.
Measuring Performance

- For multi-class problems, we often look at the **confusion matrix**.

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>A</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>B</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>C</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>D</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>E</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>F</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>G</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

C(i,j) = number of times (or percentage) class i is given label j.
Overfitting

• Suppose the classifier $h$ has error (1-accuracy) of $\text{error}_{\text{train}}(h)$
• And there is an alternate classifier (hypothesis) $h'$ that has $\text{error}_{\text{train}}(h')$
• What if $\text{error}_{\text{train}}(h) < \text{error}_{\text{train}}(h')$
• But $\text{error}_{D}(h) > \text{error}_{D}(h')$ for full test set $D$
• Then we say $h$ overfits the training data
What happens as the decision tree gets bigger and bigger?

**Overfitting in Decision Tree Learning**

Error on training data goes down, on testing data goes up
Reduced Error Pruning

• Split data into training and validation sets

• Do until further pruning is harmful

  1. Evaluate impact on validation set of pruning each possible node (and its subtree)

  2. Greedily remove the one that most improves validation set accuracy

• Then you need an additional independent testing set.
The tree is pruned back to the red line where it gives more accurate results on the test data.

Effect of Reduced-Error Pruning
• The WEKA example with Calenouria and Dorenouria I showed you used the REPTree classifier with 21 nodes.

• The classic decision tree for the same data had 65 nodes.

• Performance was similar for our test set.

• Performance increased using a random forest of 10 trees, each constructed with 7 random features.
## Decision Trees: Summary

- **Representation** = decision trees
- **Bias** = preference for small decision trees
- **Search algorithm** = none
- **Heuristic function** = information gain or information content or others
- **Overfitting and pruning**
- **Advantage** is simplicity and easy conversion to rules.
An ensemble is a set of classifiers whose combined results give the final decision.
MODEL* ENSEMBLES

• Basic Idea
  • Instead of learning one model
  • Learn several and combine them

• Often this improves accuracy by a lot

• Many Methods
  • Bagging
  • Boosting
  • Stacking

*A model is the learned decision rule. It can be as simple as a hyperplane in n-space (ie. a line in 2D or plane in 3D) or in the form of a decision tree or other modern classifier.
Bagging

• Generate bootstrap replicates of the training set by sampling with replacement

• Learn one model on each replicate

• Combine by uniform voting
Boosting

• Maintain a vector of weights for samples
• Initialize with uniform weights
• Loop
  – Apply learner to weighted samples
  – Increase weights of misclassified ones
• Combine models by weighted voting
Idea of Boosting
1. Set all E weights to 1, and learn H1.
2. Repeat m times: increase the weights of misclassified Es, and learn H2,…Hm.
3. H1..Hm have “weighted majority” vote when classifying each test
Weight(H) = accuracy of H on the training data
ADABOOST

- ADABOOST boosts the accuracy of the original learning algorithm.

- If the original learning algorithm does slightly better than 50% accuracy, ADABOOST with a large enough number of classifiers is guaranteed to classify the training data perfectly.
/* First find the sum of the weights of the misclassified samples */
for j = 1 to N do /* go through training samples */
    if h[m](x_j) <> y_j then error <- error + w_j

/* Now use the ratio of error to 1-error to change the weights of the correctly classified samples */
for j = 1 to N do
    if h[m](x_j) = y_j then w[j] <- w[j] * error/(1-error)
Example

• Start with 4 samples of equal weight .25.
• Suppose 1 is misclassified. So error = .25.
• The ratio comes out .25/.75 = .33
• The correctly classified samples get weight of .25*.33 = .0825

.2500
.0825
.0825
.0825

What’s wrong? What should we do?

We want them to add up to 1, not .4975.

Answer: To normalize, divide each one by their sum (.4975 or .5).
Sample Application: Insect Recognition

Using circular regions of interest selected by an interest operator, train a classifier to recognize the different classes of insects.
Boosting Comparison

- **ADTree classifier only** (alternating decision tree)

- Correctly Classified Instances: 268 (70.1571 %)
- Incorrectly Classified Instances: 114 (29.8429 %)
- Mean absolute error: 0.3855
- Relative absolute error: 77.2229 %

<table>
<thead>
<tr>
<th>Classified as -&gt;</th>
<th>Hesperperla</th>
<th>Doroneuria</th>
</tr>
</thead>
<tbody>
<tr>
<td>Real Hesperperlas</td>
<td>167</td>
<td>28</td>
</tr>
<tr>
<td>Real Doroneuria</td>
<td>51</td>
<td>136</td>
</tr>
</tbody>
</table>
Boosting Comparison

AdaboostM1 with ADTree classifier

- Correctly Classified Instances  303  79.3194 %
- Incorrectly Classified Instances  79  20.6806 %
- Mean absolute error  0.2277
- Relative absolute error  45.6144 %

<table>
<thead>
<tr>
<th>Classified as -&gt;</th>
<th>Hesperperla</th>
<th>Doroneuria</th>
</tr>
</thead>
<tbody>
<tr>
<td>Real Hesperperlas</td>
<td>167</td>
<td>28</td>
</tr>
<tr>
<td>Real Doroneuria</td>
<td>51</td>
<td>136</td>
</tr>
</tbody>
</table>
Boosting Comparison

- **RepTree classifier only** *(reduced error pruning)*

  - Correctly Classified Instances: 294, 75.3846%
  - Incorrectly Classified Instances: 96, 24.6154%
  - Mean absolute error: 0.3012
  - Relative absolute error: 60.606%

<table>
<thead>
<tr>
<th>Classified as -&gt;</th>
<th>Hesperperla</th>
<th>Doroneuria</th>
</tr>
</thead>
<tbody>
<tr>
<td>Real Hesperperlas</td>
<td>169</td>
<td>41</td>
</tr>
<tr>
<td>Real Doroneuria</td>
<td>55</td>
<td>125</td>
</tr>
</tbody>
</table>
Boosting Comparison

AdaboostM1 with RepTree classifier

- Correctly Classified Instances: 324 (83.0769 %)
- Incorrectly Classified Instances: 66 (16.9231 %)
- Mean absolute error: 0.1978
- Relative absolute error: 39.7848 %

<table>
<thead>
<tr>
<th>Classified as -&gt;</th>
<th>Hesperperla</th>
<th>Doroneuria</th>
</tr>
</thead>
<tbody>
<tr>
<td>Real Hesperperlas</td>
<td>180</td>
<td>30</td>
</tr>
<tr>
<td>Real Doroneuria</td>
<td>36</td>
<td>144</td>
</tr>
</tbody>
</table>
References


Stacking

- Apply multiple base learners (e.g.: decision trees, naive Bayes, neural nets)
- Meta-learner: Inputs = Base learner predictions
- Training by leave-one-out cross-validation: Meta-L. inputs = Predictions on left-out examples
Random Forests

- **Tree bagging** creates decision trees using the bagging technique. The whole set of such trees (each trained on a random sample) is called a decision forest. The final prediction takes the average (or majority vote).

- **Random forests** differ in that they use a modified tree learning algorithm that selects, at each candidate split, a random subset of the features.
Back to Stone Flies

Random forest of 10 trees, each constructed while considering 7 random features. Out of bag error: 0.2487. Time taken to build model: 0.14 seconds

Correctly Classified Instances 292 76.4398 % (81.4 with AdaBoost)
Incorrectly Classified Instances 90 23.5602 %
Kappa statistic 0.5272
Mean absolute error 0.344
Root mean squared error 0.4069
Relative absolute error 68.9062 %
Root relative squared error 81.2679 %
Total Number of Instances 382

<table>
<thead>
<tr>
<th>TP Rate</th>
<th>FP Rate</th>
<th>Precision</th>
<th>Recall</th>
<th>F-Measure</th>
<th>ROC Area</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.69</td>
<td>0.164</td>
<td>0.801</td>
<td>0.69</td>
<td>0.741</td>
<td>0.848</td>
<td>cal</td>
</tr>
<tr>
<td>0.836</td>
<td>0.31</td>
<td>0.738</td>
<td>0.836</td>
<td>0.784</td>
<td>0.848</td>
<td>dor</td>
</tr>
<tr>
<td>WAvg.</td>
<td>0.764</td>
<td>0.239</td>
<td>0.769</td>
<td>0.764</td>
<td>0.763</td>
<td>0.848</td>
</tr>
</tbody>
</table>

a b <-- classified as
129 58 | a = cal
32 163 | b = dor