Growing Decision Trees Genetically and in Parallel

Robert Keller
Brian Bentow
Joseph Malone
Harvey Mudd College
keller@cs.hmc.edu

Outline

- What is a decision tree?
- Why are decision trees of interest?
- Project history
- Pre-existing software
- How is genetic programming used?
- Why/How to use parallel computing?
- Our software
- Related work
- Future directions

General Application Area

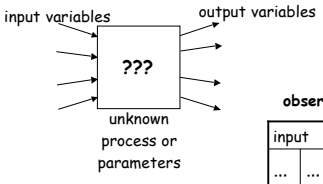
"Data Mining"

Extraction of meaningful models from collections of data

Data Mining

- Given data collected from an unknown **process** (physical, biological, economic, …)
- Devise a model that **explains** the data
- So that accurate **predictions** can be made
- (predict output from input data)

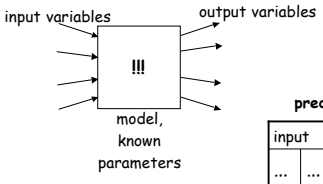
Modeling/Prediction



observations

input		output	
…	…	…	…
…	…	…	…
…	…	…	…

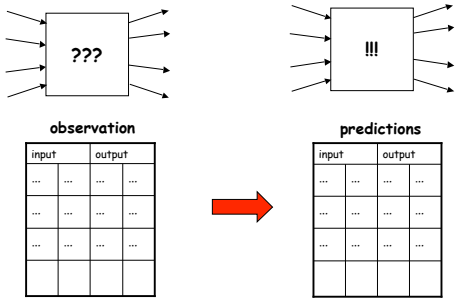
Modeling/Prediction



predictions

input		output	
…	…	…	…
…	…	…	…
…	…	…	…

Modeling/Prediction



Model Validation

- How closely do predictions made by the model agree with actual output?

Types of Models

- Statistical
  - Regression equations
  - Factor analysis
- Neural network
  - Multi-level perceptron
  - Counter propagation network
  - Support Vector Machine
- Decision tree

What is a decision tree?

- A tree of questions to be asked about the input data, resulting in a prediction about the output data.
- There are numerous varieties of questions:
  - Equality: $x_i = c$?
  - Single variable inequality: $x_i \geq c$?
  - Multi-variable inequality: $a_i x_i + a_j x_j \geq c$?
  - Boolean combinations: $x_i \geq c_i$ □ $x_j \geq c_j$?
  - etc.

Data types

- Continuous data:
  - Finite or infinite set of values
  - Numeric inequality comparison meaningful
- Categorical data:
  - Finite set of values
  - Numeric inequality comparison not meaningful

Example of a Decision Tree

- Given a set of measurements on a banknote, determine whether the banknote is a forgery.
- Input measurements (continuous):
  - diagonal
  - height on the right
  - height on the left
  - inner frame to upper border
  - inner frame to lower border

- Output category (categorical):
  - {forgery, genuine}

Example of a Decision Tree

- Given a set of measurements on a banknote, determine whether the banknote is a forgery.
- A Decision tree:

```
diagonal > 140.3
|   inner_frame_to_lower_border > 9.6
|   |   height_on_right > 130.8 forgery
|   |   height_on_right <= 130.8 genuine
|   inner_frame_to_lower_border <= 9.6 genuine
diagonal <= 140.3  forgery
```

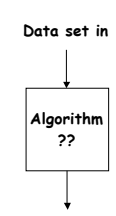
Problem: Given data observations, determine a good decision tree

- Observations from actual banknotes (partial)

```
diagonal,    height_on_left, height_on_right, …,        category
214.80000,131.00000,131.10000,9.00000,9.70000,141.00000,genuine
214.80000,129.70000,129.70000,8.70000,9.60000,142.20000,genuine
215.00000,129.60000,129.70000,10.40000,7.70000,141.80000,genuine
215.50000,129.50000,129.70000,7.90000,9.60000,141.60000,genuine
214.90000,129.40000,129.70000,8.20000,11.00000,141.90000,genuine
215.30000,130.40000,130.30000,7.90000,11.70000,141.80000,genuine
215.20000,130.80000,129.60000,7.90000,10.80000,141.40000,genuine
215.10000,129.90000,129.70000,7.70000,10.80000,141.80000,genuine

215.10000,130.30000,130.00000,10.60000,10.80000,139.70000,forgery
214.70000,130.60000,130.10000,11.80000,10.50000,139.80000,forgery
214.80000,130.50000,130.20000,11.00000,11.00000,140.00000,forgery
214.80000,130.30000,130.40000,10.10000,12.10000,139.60000,forgery
215.30000,130.80000,131.10000,11.60000,10.60000,140.20000,forgery
214.70000,130.50000,130.50000,9.90000,10.30000,140.10000,forgery
215.00000,130.40000,130.40000,9.40000,11.60000,140.20000,forgery
```

Our Problem




```
diagonal,    height_on_left, height_on_right, …, category
214.80000,131.00000,131.10000,9.00000,9.70000,141.00000,genuine
214.80000,129.70000,129.70000,8.70000,9.60000,142.20000,genuine
215.00000,129.60000,129.70000,10.40000,7.70000,141.80000,genuine
215.50000,129.50000,129.70000,7.90000,9.60000,141.60000,genuine
214.90000,129.40000,129.70000,8.20000,11.00000,141.90000,genuine
215.30000,130.40000,130.30000,7.90000,11.70000,141.80000,genuine
215.20000,130.80000,129.60000,7.90000,10.80000,141.40000,genuine
215.10000,129.90000,129.70000,7.70000,10.80000,141.80000,genuine

215.10000,130.30000,130.00000,10.60000,10.80000,139.70000,forgery
214.70000,130.60000,130.10000,11.80000,10.50000,139.80000,forgery
214.80000,130.50000,130.20000,11.00000,11.00000,140.00000,forgery
214.80000,130.30000,130.40000,10.10000,12.10000,139.60000,forgery
215.30000,130.80000,131.10000,11.60000,10.60000,140.20000,forgery
214.70000,130.50000,130.50000,9.90000,10.30000,140.10000,forgery
215.00000,130.40000,130.40000,9.40000,11.60000,140.20000,forgery
```

```
diagonal > 140.3
|   inner_frame_to_lower_border > 9.6
|   |   height_on_right > 130.8 forgery
|   |   height_on_right <= 130.8 genuine
|   inner_frame_to_lower_border <= 9.6 genuine
diagonal <= 140.3  forgery
```

Two Varieties of Tree

- Classification tree:
  - Determine one of a set of discrete categories for the output variable
- Regression tree:
  - Determine a predicted mean value for the output variable

Value of Trees

- Trees vs. Regression formulae:
  - Tree provides discrete explanation, whereas regression formula is a monolithic function
- Trees vs. Neural nets:
  - Neural nets don't necessarily display their "rationale" well

Some Difficulties

- Real-world data may be noisy and have missing observations.
- Data may have conflicting outputs for the same input.
- There is **no unique answer**:
  - Tree classification **accuracy** can be increased,
  - at the expense of increasing tree **complexity**.
- **Validation** requires additional data not already used as input

Tree Pruning

- "Pruning" a tree is a process of cutting out selected sub-trees:
  - The size gets smaller, but
  - the error usually increases.
- The trick is to prune the sub-trees that have the least impact.

Algorithmic Approaches

- Based on choosing questions that give the best discrimination with the least error, e.g.
- **Greedy algorithm**: At a node in the tree, construct a question that divides the data into two sets, such that the category error is minimum. Repeat for child nodes.
- The problem of constructing minimal decision tree has been shown **NP-complete**.

Hyafil, R. and Rivest, R.L.. Constructing optimal binary trees is NP-complete, Information processing letters, 5, 15-17, 1976.

Project history

- Wood's Hole Marine Biology Laboratory (MBL) Computer Science Clinic Project, 1999-2000 and 2000-2001, Dr. Michael Cummings, sponsor, Z. Sweedyk, advisor, James Benham, Susan Bowers, Charlie Garod, Michelle Velea, Greg Mulert, Chris Lundberg, Titus Winters students.
- As consultant on the 1999-2000 project, I suggested applying **Genetic Programming** concepts.
- Summer 2002

Pre-existing software

- c4.5 (formerly c3.0)
  Ross Quinlan, Machine learning background
  Now a commercial product c5.
- CART (Classification-and-Regression-Trees)
  Leo Breiman, et al., Statistics background
  Also a commercial product
- R-PART (Recursive Partitioning) library in S-plus
  Public domain implementation of CART methods
- Terry M Therneau and Beth Atkinson, Mayo Clinic
- Numerous others

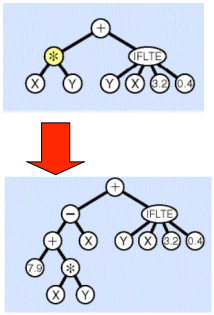
What is Genetic Programming?

- GP is a form of genetic algorithm in which the **genomes are programs**, rather than, say, bit strings.
- **Genetic operators** such as mutation and crossover are defined on programs.
- **Fitness** of a program is how well it carries out a specific function.

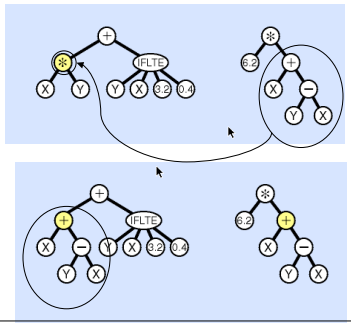
Trees as Genomes

- Every program in a given language has a **syntax tree**.
- The **genetic operators** are expressed based on syntax trees.

Mutation Operator



Crossover Operator



How is genetic programming used?

- A decision tree is a primitive form of program.
- GP seemed like a natural selection of an algorithm for producing decision trees.

Project Objectives

- We set out to design a program that would evolve decision trees based on input data.
- We chose to use the c4.5 data format.
- We used c4.5 as a performance benchmark.
- We wanted to be able to use parallel processors to speed up the computation.

First Milestone: gentree

- We produced a program that evolves both classification and regression trees (c4.5 only does classification).
- In some cases, our program produced better quality results than c4.5, in other cases, lower quality.
  We are still exploring ways to improve the results.
- Our program invariably runs longer than c4.5, especially on large data sets.
- Our program can display lots of options
  (accuracy vs. size tradeoffs).

```
Gentree output, example mbl-factor, run ID 102

Run parameters were:
1000 Chromosomes. Training dataset size: 310
Dataset is: /research/keller/gentree/data/factor/factor.data
Fitness function was: % correct - 0.001 * treeSize - 0.005 * nVars
The top 10% were replicated each generation
Target error was: 0   Target size was: 0
-----------------------------------------------------

Evolution progress (Only changes of most-fit individuals are shown):
   gen  size  error  nCorrect  fitness  mean  stddev
     0    21 13.55%   268/310     0.84  0.63  0.042
     1    15 13.55%   268/310     0.85  0.67  0.067
     2    11 13.23%   269/310     0.86  0.70  0.088
     3    29 11.29%   275/310     0.86  0.72  0.102
     4    23 11.61%   274/310     0.86  0.73  0.111
     5    23  9.68%   280/310     0.88  0.73  0.113
     7    17  9.68%   280/310     0.89  0.74  0.116
     9    11 10.00%   279/310     0.89  0.75  0.112
    10    19  9.03%   282/310     0.89  0.75  0.117
    11    23  9.03%   282/310     0.89  0.74  0.124
    12    25  8.39%   284/310     0.89  0.74  0.124
    14    29  8.06%   285/310     0.90  0.76  0.118
    15    43  7.74%   286/310     0.90  0.77  0.116
    17    25  7.74%   286/310     0.90  0.77  0.115
    18    59  7.42%   287/310     0.90  0.78  0.110
    19    49  7.42%   287/310     0.90  0.79  0.108
    21    25  7.74%   286/310     0.91  0.78  0.113
    24    21  7.74%   286/310     0.91  0.79  0.105
    26    19  7.74%   286/310     0.91  0.77  0.123
    28    17  7.74%   286/310     0.91  0.76  0.128
```

Examples

Example	method	tree size	training error %	test error %
iris	unpruned C4.5	5	2.6	5.4
iris	pruned C4.5	5	2.6	5.4
iris	gentree	5	2.6	5.4
mbl-factor	unpruned C4.5	289	1.6	test = training
mbl-factor	pruned C4.5	53	20.9	
mbl-factor	gentree	25	1.6	

Comparison Trees: iris

gentree

```
5 nodes  generation 0  error 2.632%  correct 74/76  fitness 0.9712
a4 > 0.6  (51 of 76)
|   a4 > 1.6  (25 of 51)   category  Iris-virginica  (24/1)
|   a4 <= 1.6  (26 of 51)   category  Iris-versicolor  (25/1)
a4 <= 0.6  (25 of 76)   category  Iris-setosa  (0/25)
```

c4.5

```
a4 <= 0.6 : Iris-setosa (25.0)
a4 > 0.6 :
|   a4 <= 1.6 : Iris-versicolor (26.0/1.0)
|   a4 > 1.6 : Iris-virginica (25.0/1.0)


Evaluation on training data (76 items):

          Before Pruning           After Pruning
          ---------------  --------------------------
          Size      Errors   Size      Errors   Estimate

             5    2( 2.6%)      5    2( 2.6%)    ( 8.5%)
```

Comparison Trees: mbl-factor

gentree

```
13 nodes  generation 33  error 8.387%  correct 284/310  fitness 0.9005
pos5 == D  (11 of 310)   category  0  (11/0)
pos5 != D  (299 of 310)
    pos5 == P  (20 of 299)   category  0  (17/3)
    pos5 != P  (279 of 299)
        pos8 == P  (19 of 279)   category  0  (19/0)
        pos8 != P  (260 of 279)
            pos1 == T  (34 of 260)   category  1  (25/9)
            pos1 != T  (226 of 260)
                pos1 == Y  (17 of 226)   category  1  (17/0)
                pos1 != Y  (209 of 226)
                    pos1 == S  (138 of 209)   category  1  (130/8)
                    pos1 != S  (71 of 209)   category  0  (65/6)
```

c4.5

```
Evaluation on training data (310 items):

          Before Pruning           After Pruning
          ---------------  --------------------------
          Size      Errors   Size      Errors   Estimate

           218    5( 1.6%)     39   29( 9.4%)    (20.9%)
```

Comparison Trees: mbl-factor

c4.5

```
Simplified Decision Tree:

pos1 = S: 1 (149.0/20.4)
pos1 = Y: 1 (18.0/2.5)
pos1 = V: 0 (3.0/2.1)
pos1 = Q: 0 (7.0/3.4)
pos1 = A: 0 (9.0/3.5)
pos1 = N: 0 (4.0/2.2)
pos1 = L: 0 (12.0/1.3)
pos1 = G: 0 (4.0/1.2)
pos1 = F: 0 (5.0/1.2)
pos1 = D: 0 (11.0/1.3)
pos1 = R: 0 (8.0/1.3)
pos1 = P: 0 (21.0/1.3)
pos1 = H: 0 (3.0/1.1)
pos1 = I: 0 (3.0/1.1)
pos1 = C: 0 (4.0/1.2)
pos1 = E: 0 (4.0/1.2)
pos1 = K: 0 (3.0/1.1)
```

```
pos1 = T:
|   pos5 = H: 1 (1.0/0.8)
|   pos5 = F: 1 (10.0/2.4)
|   pos5 = P: 0 (5.0/2.3)
|   pos5 = Y: 1 (13.0/3.6)
|   pos5 = M: 1 (0.0)
|   pos5 = N: 1 (0.0)
|   pos5 = L: 1 (2.0/1.0)
|   pos5 = V: 0 (3.0/1.1)
|   pos5 = A: 1 (1.0/0.8)
|   pos5 = I: 0 (2.0/1.8)
|   pos5 = G: 0 (1.0/0.8)
|   pos5 = D: 0 (1.0/0.8)
|   pos5 = E: 0 (1.0/0.8)
|   pos5 = S: 0 (1.0/0.8)
|   pos5 = K: 1 (0.0)
|   pos5 = Q: 1 (0.0)
|   pos5 = R: 0 (1.0/0.8)
|   pos5 = W: 1 (0.0)
|   pos5 = T: 1 (0.0)
|   pos5 = C: 1 (0.0)
```

gentree trade-offs: iris

```
Cached trees evaluated on training data:
   gen  size  error  nCorrect  fitness  mean stddev
     1     1 65.79%    26/76      0.34
     0     3 32.89%    51/76      0.67
     0     5  2.63%    74/76      0.97
     2     7  1.32%    75/76      0.99

Cached trees evaluated with test data:
   gen  size  error  nCorrect  fitness  mean stddev
     1     1 66.22%    25/74      0.34
     0     3 32.43%    50/74      0.68
     0     5  5.41%    70/74      0.95
     2     7  4.05%    71/74      0.96
```

Joe Malone's Optimization

- Rather than completely build a tree, then evaluate its fitness, only build it down to the nodes before the leaves.
- Then assign categories to the leaves that minimize the overall error.

Examples run by gentree

agaricus-lepiota
anneal
balance-scale
banknotes
breast-cancer
breast-cancer-wisconsin
bupa
car
clean1
crx
dsilk
factor
german
glass
golf
hayes-roth
hepatitis
imports-85
income-food
ionosphere
iris
krkopt
kyphosis
labor-neg
lung-cancer
lymphography
monk1
monk2
monk3
parity3
parity4
pima-indians-diabetes
post-operative
primary-tumor
segal
servo
soybean
tic-tac-toe
vote
wine
yeast
zoo

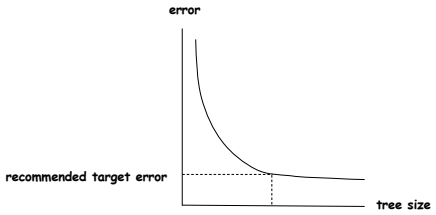
Some Challenges

- Fitness is based on a combination of **size** and **accuracy**. What is the best way to weight the two?
- If just one tree has to be recommended, which one?
- What is the best way to validate the results?

Cross-Validation

- We implemented n-way cross validation (user selects n):
  - Data set is divided into n disjoint sets.
  - n training runs are done, with (n-1)/n of the data used as training and 1/n as test.
  - The average error vs. size is plotted.
  - We expect that the "knee" in the curve indicates the most robust error value.
  - Given the target error value, a tree of corresponding size can be generated.

n-Way Cross-Validation



cross-validation notes

- Traditionally, cross-validation has been used to prune down the tree size,
- but we aren't pruning in the traditional sense.
- I am not clear on whether this is the right way to go, but it seems plausible.
- Advice is welcome.
- The knee-selection idea has not been automated.

Another Challenge: Missing Data Values

- There are several known methods for dealing with missing data values, none universally accepted.
- c4.5 uses a "fractional re-distribution" method
- We implemented the "surrogate variable" method, which prescribes a value for the missing variable based on proximity to other data points.

Another Challenge: Maintaining Population Diversity

- A common difficulty with genetic approaches is that, because of natural selection, the population of solutions **loses diversity** over time.
- This means that fewer nuances are introduced, and the trend may move toward a **local fitness minimum**.
- We have been working on ways to prevent loss of diversity, **without sacrificing quality** of the most fit solutions.
- One approach involves defining an **equivalence relation** among the individuals and only keeping one member of each equivalence class.

Choice of Equivalence Relation

- Numerous equivalence relations are possible.
- Identity is too fine, and same-error-value may be too coarse.
- We are working on using "confusion matrices" for equivalence.

```
 (a)   (b) <-classified as
----  ----
 108    21 (a): class 0
   8   173 (b): class 1
```

Why/How to use parallel computing?

- Parallel computing offers a means to **speed up** computation when other algorithmic and technology-based approaches saturate.
- Genetic programming seems to be a reasonable candidate for parallel computing: evolution **in nature** takes place in parallel.

Broad Parallel Computing Models

- **Shared Memory**: Several processors sharing variables in a common memory.
- **Distributed Memory**: Processors working independently, communicating results through messages.

Parallel Computing Trade-Offs

- **Shared Memory**:
  - Fast, but
    Requires specially-implemented interconnect.
  - Scalability limited.
- **Distributed Memory**:
  - Slower, due to communication delays, but
  - Commodity processors can be used,
  - Highly-scalable (1000's of processors on a network)

Choice for Genetic Programming

- We use distributed memory
- **MPI** library (Multi-Processor Interface)
- Very portable
- Many systems use it:
  - "Beowulf" clusters (16 processors at HMC)
  - Others
- Based on Single-Program Multiple-Data Model
  - One program for all computers
  - Differentiable by processor id

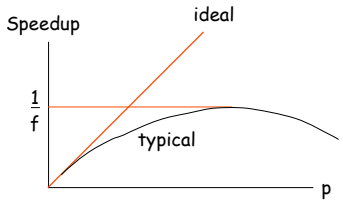
pagentree

- Parallel version of gentree.
- Population is divided into "demes".
- Each deme evolves independently for a while, then passes most fit to its immediate neighbor, called the "island model".
- We currently use a logical "ring" organization to connect the processors.
- Demes can also help with the diversity problem.

Speedup Measurements

- Speedup $S_p = T_1/T_p$
- T_p = Time to execute on p processors
- Ideally $S_p \geq kp$, where k is a constant near 1

Speedup for a fixed problem size



f = fraction of code that must be executed sequentially

Sample Speedup Measurements

running mbl-factor with aggregate population 1000		
processors	run time (sec.)	speedup
1	580.530	1
2	328.650	1.77
3	347.400	1.67
4	167.490	3.47
8	74.230	7.80

Early Termination

- Our implementation includes a distributed load sharing and early-termination capability.
- When a processor is not "making progress" toward better solutions in its deme, it notifies its neighbor. The notification is passed to other processors, until a processor is found that is making progress, which then gives some highly fit individuals to the originating processor.
- If no such processor is found, then a message is sent out to terminate execution.

Added Challenge

- Early termination makes it difficult to determine speedup, because the amount of work done is dependent on the number of processors.
- We want to not delay getting answers to the user, yet we also want to be getting true speedup from the multi-processor version of the system.

Our software

gentree/pagentree is a single source build under CVS on turing.cs.hmc.edu. Less than 1000 out of 5837 are devoted to MPI support. All MPI code was done by Brian Bentow.

```
wc *.cpp *.h
   1333    4220   35197 chromosome.cpp
    121     279    2569 config.cpp
    878    2234   17348 dataSet.cpp
    211     431    3497 directedoutput.cpp
   1824    5629   51635 gentree.cpp
    299     794    5705 question.cpp
    183     537    4682 chromosome.h
    121     634    4062 config.h
    268    1037    7167 dataSet.h
    193     584    4094 directedoutput.h
    207     804    5901 gentree.h
    199     700    4871 question.h
   5837   17883  146728 total
```

Eventually we plan to make it available to the public on the web.

Related work (1)

- There is a fair amount of work on **parallel genetic programming**. I won't list it all here, but it is readily accessible on the web.
- In 1999, almost no references to **genetic programming of decision trees** could be found. An old, cursory, one I discovered this past year was:

  Koza, J.R., *Concept formation and decision tree induction using the genetic programming paradigm*, in Hans-Paul Schwefel and Reinhard Maenner (eds), Parallel problem solving from nature, Springer-Verlag, 1991.

Related work (2)

- At the start of the summer project, we discovered some more recent work, notably:
  H.C. Kennedy, et al., ***The construction and evaluation of decision trees: A comparison of evolutionary and concept learning methods***, Springer, LNCS 1305, 1997 and
- Bot, Martijn and Langdon, ***Application of Genetic Programming to Induction of Linear Classification Trees***, LNCS 1802 Genetic Programming, Springer-Verlag, 2000.
- At the end of the summer, we discovered some very relevant work hiding under "genetic algorithms": Athanasios Papagelis, Dimitris Kalles, ***GATree:Genetically Evolved Decision Trees***, International Conference on Tools in AI, IEEE, Nov. 2001. This work reports extraordinarily good results for tree size (6 times better than C4.5, in some cases). We are still in the process of evaluating it.
- Additional related work is appearing as we speak. See our wiki: http://www.cs.hmc.edu/~jmalone/cgi-bin/research/wiki.pl

Future directions

- We have ideas for several improvements in the existing algorithm.
- An ultimate decision-tree generator can be built by seeding the population with results of C4.5 or other algorithms.
- While parallel processing looks promising, we have more work to do to cogently present its benefits.