Automating the Design of Decision Tree Algorithms with Evolutionary Computation

Alex Freitas
University of Kent, UK

work with Rodrigo Barros, Marcio Basgalupp, Andre de Carvalho, University of Sao Paulo, Brazil
Outline of the talk

• Introduction to the classification task of data mining and decision trees

• Objective and motivation for this work

• Related research topics

• The proposed evolutionary algorithm for evolving decision tree algorithms

• Experiments and results

• Conclusions / Research Directions
Introduction to Classification

• Each instance (example, object) belongs to a predefined class
  – An instance represents a customer, a patient, etc. (application dependent)

• Each instance consists of two parts:
  – < predictor attributes (features), class attribute >, e.g.:
  – < Gender = M, Age = 25, Salary = 35,000, credit = good >

• Goal: to discover a classification model which allows us to predict the class of an instance, given its predictor attributes

• Discovered classification model (e.g. in the form of a decision tree) is evaluated on a test set, separated from the training set
Data Partitioning for Classification

<table>
<thead>
<tr>
<th>Training set</th>
<th>Test set</th>
</tr>
</thead>
<tbody>
<tr>
<td>(known-class instances)</td>
<td>(unknown-class instances)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>class</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td></td>
<td>good</td>
</tr>
<tr>
<td></td>
<td>bad</td>
</tr>
<tr>
<td></td>
<td>bad</td>
</tr>
<tr>
<td></td>
<td>good</td>
</tr>
<tr>
<td></td>
<td>good</td>
</tr>
<tr>
<td></td>
<td>bad</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>class</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>?</td>
</tr>
<tr>
<td></td>
<td>?</td>
</tr>
<tr>
<td></td>
<td>?</td>
</tr>
<tr>
<td></td>
<td>?</td>
</tr>
<tr>
<td></td>
<td>?</td>
</tr>
</tbody>
</table>

prediction
Training Set for Classification: a very simple example

Class attribute = will the customer buy a given product?

<table>
<thead>
<tr>
<th>Age</th>
<th>Gender</th>
<th>Salary</th>
<th>Buy?</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td>M</td>
<td>medium</td>
<td>yes</td>
</tr>
<tr>
<td>21</td>
<td>M</td>
<td>high</td>
<td>yes</td>
</tr>
<tr>
<td>23</td>
<td>F</td>
<td>medium</td>
<td>yes</td>
</tr>
<tr>
<td>34</td>
<td>F</td>
<td>high</td>
<td>yes</td>
</tr>
<tr>
<td>30</td>
<td>F</td>
<td>medium</td>
<td>no</td>
</tr>
<tr>
<td>21</td>
<td>M</td>
<td>low</td>
<td>no</td>
</tr>
<tr>
<td>20</td>
<td>M</td>
<td>low</td>
<td>no</td>
</tr>
<tr>
<td>18</td>
<td>F</td>
<td>low</td>
<td>no</td>
</tr>
<tr>
<td>34</td>
<td>F</td>
<td>medium</td>
<td>no</td>
</tr>
<tr>
<td>55</td>
<td>M</td>
<td>medium</td>
<td>no</td>
</tr>
</tbody>
</table>
Decision Tree Structure - example

- Internal nodes: predictor attributes
- Leaf nodes: predicted class
- To classify a new instance (customer), push it down the tree, until it reaches a leaf node

```
  Salary?
  /     \
low    medium    high
   |     |          |
  no   Age?   yes
         |     |
        ≤ 25 > 25
         |
        yes no
```
Objective of This Work

• To **automatically “invent” new decision tree algorithms** for classification – algorithms that output a decision tree model

• To do this automatic algorithm invention, we propose a new Evolutionary Algorithm (EA)

• Note: there are many EAs for building **decision trees** for the dataset at hand, but we are doing something very different:

• We are proposing an EA for creating new rule decision tree **algorithms**, i.e., algorithms that in principle can be used to build decision trees **for any given classification dataset**
Automating the design of decision tree algorithms

- “Manual evolution” of the design of decision tree algorithms

<table>
<thead>
<tr>
<th>Breiman et al.</th>
<th>Quinlan</th>
<th>Quinlan</th>
<th>Yildiz &amp; Alpaydin</th>
</tr>
</thead>
<tbody>
<tr>
<td>CART</td>
<td>ID3</td>
<td>C4.5</td>
<td>omnivariate D.T.s</td>
</tr>
<tr>
<td>... 1984</td>
<td>1986</td>
<td>1993</td>
<td>2001 ...</td>
</tr>
</tbody>
</table>

- The idea is to replace the above manual, “ad-hoc” evolution with an automatic, “data-driven” evolution of DT algorithms
Manual invention of Decision Tree Algorithms

Full DT algorithm

while (...) 
  ... 
  eval. features() 
  select feature() 
  expand tree() 
  ... 
end while 
prune tree()

Human-designed generic decision tree program

DTA

datasets

DTA

DTA

DTA

decision trees

...
Automatic invention of Decision Tree Algorithms

"building blocks" of DT algorithms

components of eval. features()
components of select feature()
components of expand tree
components of prune tree()

Iteratively mix building blocks

Machine-designed generic Decision Tree Algorithms

DTA

datasets

DTA

DTA

DTA

decision trees
Motivations for automating the design of data mining algorithms (1)

- New level of automation in data mining
  - Increasing the degree of “computational intelligence” (AI’s perspective)
  - Study the differences between human-designed and machine-designed algo.’s

- Avoid algorithm biases introduced by the human algorithm designer (e.g., typical decision tree algorithms are greedy)

- No classification algorithm is “the best” across all datasets
  - New machine-designed algorithm can be useful for types of datasets where human-designed classification algorithms do not very perform well

- In this work we focus on Decision Tree Algorithms, due to the comprehensibility of their output classification models
Motivations for automating the design of data mining algorithms (2)

• Possibility of automatically designing an algorithm customized to the data at hand
  – Research on meta-learning (automated algorithm recommendation) involves selecting the best algorithm out of a predefined list of algorithms

    BY CONTRAST

  – The proposed method can construct an algorithm customized to the data, out of a very large search space of possible candidate algorithms

• In principle algorithm construction is more flexible than algorithm selection
Related Research Topics

• Different levels of algorithm “configuration”:

- Automatic full algorithm design
- Automatic design of (one or a few) algorithm components/procedures
- Automatic parameter optimization
In the context of hyper-heuristics (HHs)...

• A DT algorithm is a “set of heuristics”
  – Heuristic feature evaluation function
  – Heuristic data splitting criteria
  – Heuristic missing value strategies
  – Heuristic pruning procedures

• A hyper-heuristic searches in the space of heuristics
  – E.g. (Vella et al. 2009) evolves a heuristic evaluation function

• Here, the EA evolves the entire DT algorithm, i.e., it searches in the space of entire DT algorithms

• Note: most HHs focus on optimization, not classification
Maximizing accuracy in training set does not necessarily lead to maximizing accuracy in test set (due to “overfitting / oversearching”)
Related Forthcoming Events

• GECCO-2015
  – Track on Search-Based Software Engineering and Self-* Search (Ochoa & Kessentini)
  – 5th Workshop on Evolutionary Computation for the Automated Design of Algorithms (Woodward & Tauritz)
  – Workshop on Automatically Configurable Algorithmic Frameworks (Simons et al.)

• This LION conference – tutorial on:
  – Automatic Algorithm Configuration: From Parameter Tuning to Automatic Design (Lopez-Ibanez & Stutzle)
Basic Flowchart of Evolutionary Algorithms

1. Generate a random initial population $P(0)$
2. Compute the fitness $f(p)$ of each individual of the current population $P(t)$
3. Select individuals from $P(t)$ with probability proportional to their fitness values
4. Generate $P(t+1)$ by applying variation operators to individuals selected from $P(t)$

**An individual**
- a candidate solution
- (a candidate DT algorithm)

**Fitness function**
- = evaluation function
EA-based invention of decision tree algorithms

1. **Create initial population of algorithms**
2. **Compute accuracy of each algorithm**
3. **Choose best algorithms to be used as “parents”, and modify them a little, producing new “child” algorithms**
4. **Compute accuracy of each algorithm**
5. **Stop?**
   - If no: go back to step 3
   - If yes: Return best DT algorithm

```
Compute accuracy of each algorithm
```

```
Choose best algorithms to be used as “parents”, and modify them a little, producing new “child” algorithms
```
Two different scenarios for automatic algorithm evolution

- **Key point**: the fitness function measures the predictive performance of a candidate DT algorithm

1. Evolve a *robust* DT algorithm (which performs well across many datasets)
   - Fitness function computed on meta-training set with many different datasets

2. Evolve a DT algorithm *tailored* to a specific dataset
   - Fitness function computed on meta-training set which is a subset of the target dataset
   - **We focus on this scenario**
Evolving a decision-tree induction algorithm tailored to one specific data set

Stats = classification accuracy (%)
The Proposed Evolutionary Algorithm (EA) for Creating Decision Tree Algorithms

- It uses a linear encoding to represent a decision tree algo.
  - An individual’s genes encode encapsulated procedures/functions

- Motivation for this encoding
  - Simple to implement, uses standard genetic (variation) operators

- Fitness function: classification accuracy of the decision tree algorithm on validation set (part of meta-training set)
  - Training or Building set (70% of meta-training set): used to build the tree
  - Validation set (30% of meta-training set): used to measure the tree’s accuracy

**Note:** this EA evolves a DT algorithm *tailored to a single dataset*
Linear representation of a DT algorithm

- Each individual specifies how to perform 4 types of "building blocks", each one consisting of 2 or 3 genes

<table>
<thead>
<tr>
<th>split</th>
<th>stopping</th>
<th>missing values</th>
<th>pruning</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

2 split genes: split criterion – 16 options
- binary/multi-way split flag

2 stopping genes: stopping criterion – 5 options
- stopping parameter – adjusted to criterion

3 missing value genes: split evaluation – 4 options
- instance distribution across children (7)
- classification of test instance - 3 options

2 pruning genes: pruning method – 5 options
- parameter – adjusted to method
Building Block 1: node splitting (for tree expansion)

• We need some criteria to split the instances at the current tree node into subsets of instances (child nodes)

<table>
<thead>
<tr>
<th>split</th>
<th>stopping</th>
<th>missing values</th>
<th>pruning</th>
</tr>
</thead>
</table>

2 split genes:

binary/multi-way split flag

split criterion (feature selection criterion) – 16 options …
The 16 options for the split criterion gene

- Information gain (Quinlan, 1986)
- Gini index (Breiman et al., 1984)
- Global mutual information (Gleser & Colleen, 1972)
- G statistics (Mingers, 1987)
- Mántaras criterion (Mántaras, 1981)
- Hypergeometric distribution (Martin, 1997)
- Chanda-Varghese criterion (Chandra & Varghese, 2009)
- DCSM (Chandra et al, 2010)
- $\chi^2$ (Mingers, 1989)
- Mean posterior improvement (Taylor & Silverman, 1993)
- Normalized gain (Jun et al., 1997)
- Orthogonal criterion (Fayyad & Irani, 1992)
- Twoing (Breiman et al., 1984)
- CAIR (Ching et al., 1995)
- Gain Ratio (Quinlan, 1993)
Building Block 2: tree-expansion stopping criteria

• We need some criteria to decide when to stop the tree expansion (node splitting) process

<table>
<thead>
<tr>
<th>split</th>
<th>stopping</th>
<th>missing values</th>
<th>pruning</th>
</tr>
</thead>
</table>

2 stopping genes:
Stopping criterion – 5 options:
- class purity (all instances in a node have the same class)
- maximum tree depth
- minimum number of instances for a non-leaf node
- minimum percentage of instances for a non-leaf node
- minimum accuracy within a node

A parameter that adjusts each criterion to a valid range
Building Block 3: coping with missing values

- Missing value = unknown value of a feature for some instance (e.g. unknown Age)

<table>
<thead>
<tr>
<th>split</th>
<th>stopping</th>
<th>missing values</th>
<th>pruning</th>
</tr>
</thead>
</table>

3 missing value genes:
- split evaluation – 4 options:
  - ignore all instances with missing attribute value
  - missing value imputation: mean or mode in node
  - missing value imputation: mean or mode in class
  - weight splitting criterion value by % of missing values
3 missing value genes (continuation):

instance distribution across children – 7 options:
  ignore instance
  imputation: mode or mean, regardless of class
  imputation: mode or mean of the instance’s class
  assign instance to all child nodes
  assign instance to largest child node
  assign instance to largest child node considering class
  weight instance according to child node sizes

classification of test instance - 3 options:
  explore all branches, combining results
  take the route to most probably partition
  halt classification, assign instance to majority class of node
Building Block 4: tree pruning

- We need criteria to specify how to prune the decision tree, to avoid that it “overfits” the training set

<table>
<thead>
<tr>
<th>split</th>
<th>stopping</th>
<th>missing values</th>
<th>pruning</th>
</tr>
</thead>
</table>

2 pruning genes:
Pruning criterion – 5 options:
- reduced error pruning
- pessimistic error pruning
- minimum error pruning
- cost-complexity pruning
- error-based pruning

A parameter that adjusts each criterion to a valid range
Experimental Set Up (1)

• Parameters of the EA:
  – Population size: 100;
  – Maximum number of generations: 100;
  – Selection: tournament selection with size $t = 2$;
  – Elitism rate: 10 individuals;
  – Crossover
    • One-point Crossover, with 90% probability;
  – Mutation
    • Random uniform gene mutation with 10% probability.
Experimental Set Up (2)

• We run the EA for one data set at a time
  – Recall: the goal is to evolve a DT algorithm \textit{tailored} to a specific dataset

• 10 fold cross-validation
  – Meta-Training Set (90\%) – accessed by the EA
    • Sub-training (building) set (70\%)
    • Validation set (30\%)
  – Meta-Test Set (10\%) – \textit{not} accessed by the EA

For details of experiments and results:
## 20 Benchmarking Classification Data Sets

<table>
<thead>
<tr>
<th>Data Set</th>
<th># Instances</th>
<th># Attributes</th>
<th># Numeric Attributes</th>
<th># Nominal Attributes</th>
<th>% Missing Values</th>
<th># Classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>abalone</td>
<td>4177</td>
<td>9</td>
<td>7</td>
<td>1</td>
<td>0</td>
<td>30</td>
</tr>
<tr>
<td>anneal</td>
<td>898</td>
<td>39</td>
<td>6</td>
<td>32</td>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td>arrhythmia</td>
<td>452</td>
<td>280</td>
<td>206</td>
<td>73</td>
<td>0.32</td>
<td>16</td>
</tr>
<tr>
<td>audiology</td>
<td>226</td>
<td>70</td>
<td>0</td>
<td>69</td>
<td>2.03</td>
<td>24</td>
</tr>
<tr>
<td>bridges_version1</td>
<td>107</td>
<td>13</td>
<td>3</td>
<td>9</td>
<td>5.53</td>
<td>6</td>
</tr>
<tr>
<td>car</td>
<td>1728</td>
<td>7</td>
<td>0</td>
<td>6</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>cylinder_bands</td>
<td>540</td>
<td>40</td>
<td>18</td>
<td>21</td>
<td>4.74</td>
<td>2</td>
</tr>
<tr>
<td>glass</td>
<td>214</td>
<td>10</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td>hepatitis</td>
<td>155</td>
<td>20</td>
<td>6</td>
<td>13</td>
<td>5.67</td>
<td>2</td>
</tr>
<tr>
<td>iris</td>
<td>150</td>
<td>5</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>kdd_synthetic_control</td>
<td>600</td>
<td>62</td>
<td>60</td>
<td>1</td>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td>segment</td>
<td>2310</td>
<td>20</td>
<td>19</td>
<td>0</td>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td>semeion</td>
<td>1593</td>
<td>266</td>
<td>265</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>shuttle_landing_control</td>
<td>15</td>
<td>7</td>
<td>0</td>
<td>6</td>
<td>28.89</td>
<td>2</td>
</tr>
<tr>
<td>sick</td>
<td>3772</td>
<td>30</td>
<td>7</td>
<td>22</td>
<td>5.54</td>
<td>2</td>
</tr>
<tr>
<td>tempdiag</td>
<td>120</td>
<td>8</td>
<td>1</td>
<td>6</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>tep.fea</td>
<td>3572</td>
<td>8</td>
<td>7</td>
<td>0</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>vowel</td>
<td>990</td>
<td>14</td>
<td>10</td>
<td>3</td>
<td>0</td>
<td>11</td>
</tr>
<tr>
<td>winequality_red</td>
<td>1599</td>
<td>12</td>
<td>11</td>
<td>0</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>winequality_white</td>
<td>4898</td>
<td>12</td>
<td>11</td>
<td>0</td>
<td>0</td>
<td>10</td>
</tr>
</tbody>
</table>
Analysis of results based on ranking according to F-measure (to be max.), Accuracy (to be max.), and Tree Size (to be min.)

<table>
<thead>
<tr>
<th>Data set</th>
<th>F-Measure</th>
<th>Accuracy</th>
<th>Tree Size</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CART</td>
<td>C4.5</td>
<td>DT-EA</td>
</tr>
<tr>
<td>abalone</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>anneal</td>
<td>2.5</td>
<td>2.5</td>
<td>1</td>
</tr>
<tr>
<td>arrhythmia</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>audiology</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>bridges_version1</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>car</td>
<td>2.5</td>
<td>2.5</td>
<td>1</td>
</tr>
<tr>
<td>cylinder_bands</td>
<td>2</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>glass</td>
<td>2.5</td>
<td>2.5</td>
<td>1</td>
</tr>
<tr>
<td>hepatitis</td>
<td>2.5</td>
<td>2.5</td>
<td>1</td>
</tr>
<tr>
<td>iris</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>kdd_synthetic_control</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>segment</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>semeion</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>shuttle_landing_control</td>
<td>2.5</td>
<td>2.5</td>
<td>1</td>
</tr>
<tr>
<td>sick</td>
<td>2.5</td>
<td>2.5</td>
<td>1</td>
</tr>
<tr>
<td>tempdiag</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>tep.fea</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>vowel</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>winequality_red</td>
<td>2</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>winequality_white</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

Average: 2.5 2.2 1.3 2.4 2.2 1.4 1.2 2 2.8

The smaller the average rank, the better the result
Statistical analysis of the ranking results

- We compare multiple algorithms on multiple datasets, using a non-parametric Friedman test with the Nemenyi post-hoc test.

- First, Friedman test compares the average ranks of algorithms.

- If the null hypothesis of similar performance is rejected, we use the Nemenyi test for pairwise comparisons of algorithms.

- Results of Statistical Analysis
  - DT-EA significantly better than CART and C4.5, in terms of both predictive performance measures (F-measure and accuracy).
  - DT-EA significantly worse than CART and C4.5, in terms of tree size.
  - **Note**: predictive performance is considered more important than tree size in classification.
Recent applications of the EA for creating DT algorithms

In Bioinformatics


• Barros et al. Automatic design of decision-tree induction algorithms tailored to flexible-receptor docking data. *BMC Bioinformatics 2012.*

In Software Engineering

Conclusions

• We proposed an EA for automatically designing decision tree induction algorithms

• Encouraging results
  – The evolved DT algorithms significantly outperformed the two most used DT algorithms: C4.5 and CART

• The EA is very time consuming, but…
  – How long a human would take to develop and optimize a new algorithm for a given dataset? Months??
  – Number of application domains >> number of human experts on DT algorithms
  – So, the EA is more cost-effective than manual algorithm design
Some Future Research Directions

- Extension of the current EA:
  - Extend individual representation to encode more options for the building blocks of DT algorithms
  - Speed up the EA with parallel processing

- Developing new EAs for evolving other types of classification algorithms

- Developing new EAs for evolving clustering algorithms

- Developing other types of search methods (not EA) for evolving data mining algorithms
Thanks for listening

Questions?