Bulldozers II

Tomasek, Hajic, Havranek, Taufer
Building database

- CSV -> SQL
  - Table trainRaw
    - 1:1 parsed csv data input
Building database

- **table trainRaw**
  - Columns need transformed into the form for effective use
  - Foreach column:
    - Distinct analysis
    - String data into separate table
    - Replace with int indexed value
    - Create relation constraint
Structuring database

- trainRaw -> train table
  - fast access to data
  - useless data in detached tables
Using database

- SQL stored & compiled procedures and functions for data analysis
  - getBestEnum
  - GetMedian
  - GetAvgVar
First solution - Decision tree

- solution for generic categorization problem
- category
  - = price interval
- tree nodes
  - switches for Enums
Decision tree - example
Our data

- 39 different enums
- avg value range = 10
- choosing best enum for node
  - Variance vs. Count
  - counted in sql
    - first iteration takes ~10 min
Categories

- how big?
  - no categories
  - fix size
    - according to fit function is $100 ok
  - variable size
    - 100 bulldozers for cat
  - use some genetic to find best size :)


Decision tree results

- depth 4
- runtime 2:34:02
- result 0.5431
To do

● don't use irrelevant enums
● sql optimizations
● multi core processing
● find some very strong Machine
Statistics

- Some columns give no usable information
- About half columns are machine type specific
Second solution - genetic

- Population member
  - Expression tree
    - Evaluates price
  - Nodes
    - [Price] -> Price
    - Constant, Arithmetic, Sql Aggregation, Switch

- Fit function
  - Challenge official: RMSLE

- Reproduction
  - switching subtrees between father and mother
Second solution - genetic

- Mutation
  - Specific per node type
    - Only few types can mutate
  - Random added members
    - Avoids of local extremes
Genetics - use & experience

● **Original input parameters**
  ○ Population size
    ■ Very large is not needed
    ■ For performance
    ■ Actual value 50 members
  ○ Max depth
    ■ For performance and convergence
    ■ 10 seems to be enough, actual value 12
  ○ Train data sample
    ■ Size
      ● Performance & miscellany, actual 25%
      ■ Select every generation / Keep same
Genetics - use & experience

● Added parameters
  ○ Min depth
    ■ Avoidance of
      ● One-node trees
      ● Train data specific expressions
  ○ Action probabilities
    ■ Reproduction
      ● Makes variety, actually 0.6
    ■ Clone
      ● Not important, actually 0.3
    ■ Mutation
      ● Important is high value, actually 0.7
      ● Helps with convergence
      ● Needs to upgrade in several node types
Genetics - use & experience

- Node type implementation & specific tuning
  - Abstract node
    - Mutation is called recursively to children
  - Constant
    - Finite universum
      - \{ k / 100 | k in \mathbb{N} U \{0\} & k < 101 \} U \{ \pi, e \}
    - Mutation
      - + d where d is from \{-0.01, 0, 0.01\}
  - Arithmetic
    - +-*/ only binary
Genetics - use & experience

● Node type implementation & specific tuning
  ○ SqlAgg
    ■ Defined by agg. function and selected data columns
    ■ Returns aggregated price of database table rows what have same values in selected columns
    ■ Mutation changes agg. function
      ● Maybe change of selected column is needed
  ○ Switch
    ■ Defined by one data column
    ■ k children
      ● k is loaded only once by column variety
Genetics - use & experience

● Evolution process implementation
  ○ For every generation
    ■ Selection
    ■ Reproduction & cloning
    ■ Mutation
  ○ Evaluating train data sample by each member
  ○ Fit calculation
  ○ Best one serialisation
  ○ GC.Collect()

● Genetic process is very slow
  ○ Threadpool implemented
    ■ by member
Genetics - results

● First whole night run on full train data
  ○ 294 generations
  ○ Best result fit 0.49 (381 / 454)
    ■ Challenge leader has 0.22
    ■ Median benchmark has 0.74
  ○ Lesson
    ■ Min depth constraint
      ● Very simple data specific nodes broke population development
    ■ Smaller train data sample
      ● More data-specific results and more performance
    ■ More mutation
    ■ More sql query parallelism
    ■ Sql results caching
Genetics - example
Neural networks

What have we tried?
- single MLP
- 10 classes of equal magnitude
- 18 / 51 features
- network structure 18 - 10 - 10
Neural networks

What have we tried? (cont.)
- backpropagation learning algorithm
- different minimization techniques
  - gradient descent
  - conjugated gradient ((C) Andrew Ng)
- different values of regularization
- different training set sizes
How did it go?
Actual results

Best experiment:
- trained 10000 samples
- 50 iterations of Conjugate gradient
- classification accuracy on all training data:

0.224

RMSLE on Validation data:
0.773

(mean benchmark: 0.74745)
What went wrong?

Non-numerical features
- overall: 8 comparable features
- 43 non-numerical features

Missing features
- not all features are available for all samples
- sometimes less than half
- \Rightarrow inaccurate guesswork

Time constraints
- not trained on all data (100k samples \sim 1 night)
What do we do about it?

Non-comparable features
- set up indicator variables

Missing features
- better guesswork (mean of class)
Further work

- multiple MLPs + agreement algorithm
- vary amount of classes
- vary class size
  (equal magnitude vs. equal width)
- more detailed (class -> price) conversion
- different cost function
  - factor in cost of misclassification
- different learning algorithm?