Scalability of Predictive Modeling Algorithms
Master's thesis presentation

Ing. Tomáš Frýda

Supervisor: Ing. Pavel Kordík, Ph.D.
Outline

FAKE GAME
- Base Models
- Ensembles
- Evolution
Outline

FAKE GAME
   Base Models
   Ensembles
   Evolution

H2O
   Overview
   Architecture
   Usage
   Related Software
Outline

FAKE GAME
  Base Models
  Ensembles
  Evolution

H2O
  Overview
  Architecture
  Usage
  Related Software

Hyper-parameter Optimization
  SMBO
Outline

**FAKE GAME**
- Base Models
- Ensembles
- Evolution

**H2O**
- Overview
- Architecture
- Usage
- Related Software

**Hyper-parameter Optimization**
- SMBO

**Implementation**
Outline

FAKE GAME
  Base Models
  Ensembles
  Evolution

H2O
  Overview
  Architecture
  Usage
  Related Software

Hyper-parameter Optimization
  SMBO

Implementation

Experiments
Outline

FAKE GAME
  Base Models
  Ensembles
  Evolution

H2O
  Overview
  Architecture
  Usage
  Related Software

Hyper-parameter Optimization
  SMBO

Implementation

Experiments

Results

References
Motivation

- Model selection usually does not depend solely on predictive performance
- I took into account time, which gives me two basic use cases:
  - Good enough model trained using limited computational resources
  - Highly accurate model trained using as much computational resources as needed
- Make FAKE GAME usable on big data
Originally created for small data
Base models
  - Decision tree, KNN, etc
  - Regression models
Ensembles
  - Bagging
  - Boosting
  - Stacking, Cascade Correlation, ...
Genetic programming-based ensemble creation
Base Models

- regression models
  - linear
  - polynomial
  - sigmoid
  - sine, ...

- regression models are used as discriminant functions for classification

- decision trees

- k-NN

- ...

Scalability of Predictive Modeling Algorithms

Ing. Tomáš Frýda

FAKE GAME

Base Models

Ensembles

Evolution

H2O

Overview

Architecture

Usage

Related Software

Hyper-parameter Optimization

SMBO

Implementation

Experiments

Results

References
Ensembles

- Arbitrating\(^1\)
- Bagging
- Boosting
- Cascade Generalization
- Cascading\(^1\)
- Delegating\(^1\)
- Stacking

\(^1\)used only for classification
Evolution

Genetic programming used for evolving templates that can be expanded to hierarchical ensembles
H2O

- framework for distributed machine learning based on MapReduce
- support for preprocessing and data manipulation
- RESTful API used by various language bindings (R, Python, ...)

FAKE GAME

- Base Models
- Ensembles
- Evolution

H2O

- Overview
- Architecture
- Usage
- Related Software
- Hyper-parameter Optimization
- SMBO
- Implementation
- Experiments
- Results
- References
Overview

Machine Learning algorithms included in H2O:

- Deep Learning
- Distributed Random Forest
- Gradient Boosting Machines
- Generalized Linear Model
- Naïve Bayes
- K-Means
- PCA
- GLRM
- ...
Architecture

- in-memory MapReduce
- uses distributed key-value storage
- tries to keep related data in the same or nearby node in order to minimize network usage
- columns are compressed and lazily decompressed just in time of usage in CPU registers
- parallel data load

See more at
http://blog.h2o.ai/2014/03/h2o-architecture/
How are decision trees built in H2O?

**Implementation #1**

Build independent trees per machine local data

- RVotes approach
- Each node builds a subset of forest

---

Implementation #1

✔ Fast - trees are independent and can be built in parallel

❌ Data have to fit into memory

❌ Possible accuracy decrease if each node can see only subset of data

₀xdata
Implementation #2

Build a distributed tree over all data
Implementation #2

Each data point has assigned a tree node

- stored in a temporary vector

Pass over data points means visiting tree nodes

Dataset points

0xdata
Implementation #2

Each data point has in/out of bag flag

- stored in a temporary vector

Trick for on-the-fly scoring: position out-of-bag rows inside a tree is tracked as well
Implementation #2

Tree is built per layer

- Each node prepares histogram for splitting

![Tree is built per layer diagram](image)

In/Out of bag flags

Dataset points

Active tree layer

0xdata
Implementation #2

Tree is built per layer

- Histograms are reduced and a new layer is prepared
Implementation #2

☑️ Exact solution - no decrease of accuracy

☑️ Elegant solution merging tree building and OOB scoring

☒ More data transfers to exchange histograms

☒ Can produce huge trees (since tree size depends on data)
Python

```python
import h2o

h2o.init()
data = h2o.import_file("data.csv")

# Create test/train split
s = data["Year"].runif()
train = data[s <= 0.75]
test = data[s > 0.75]

# Create an estimator
dl = H2ODeepLearningEstimator(
    hidden=[10,10],
epochs=5,
balance_classes=True)

# Train an estimator
dl.train(
    x=myX,
y=myY,
    training_frame=train,
    validation_frame=test)
```

R

```r
library("h2o")

h2o.init()
dt <- h2o.importFile("data.csv")

data = dt
data = dt.splitFrame(data = dt, ratios = 0.75)
train <- dt.split[[1]]
test <- dt.split[[2]]

# Create an estimator and # train it
dl <- h2o.deeplearning(
    x = myX,
y = myY,
    training_frame = train,
    validation_frame = test,
    hidden = c(10,10))
```
Scalability of Predictive Modeling Algorithms

Ing. Tomáš Frýda

FAKE GAME

Base Models
Ensembles
Evolution

H2O

Overview
Architecture
Usage
Related Software
Hyper-parameter Optimization
SMBO
Implementation
Experiments
Results
References

H2O Flow
Scalability of Predictive Modeling Algorithms

Ing. Tomáš Řýda

FAKE GAME
Base Models
Ensembles
Evolution

H2O
Overview
Architecture
Usage
Related Software
Hyper-parameter Optimization
SMBO
Implementation
Experiments
Results
References

H2O Flow

Welcome to H2O Flow:
Flow has a *hybrid* user interface.

importFiles
setupParse [ "nfs://Users/prithw...]
parseRaw srcs: ["nfs://Users/pr...]
getJob "$0381ac18025832d4f.......
getFrame "allyears2k_headers.hex"
inspect getColumnSummary "allyea...
plot data: inspect 'distribution', 
grid inspect "distribution", get...
grid inspect "summary", getCol...
grid inspect "characteristics", ...
grid inspect 'data', getFrame "a...
buildModel
buildModel 'deeplearning', ["tra...
getModel "DeepLearningModel_89a...
predict "DeepLearningModel_88aa...
inspect getPrediction 'DeepLearn...
grid inspect metrics", getPredi...
grid inspect "prediction", getPr...
### Get Model

```
getModel "DeepLearningModel__89aaf0804239efe2521844cddcb417e"
```

### Model

**Key:** DeepLearningModel__89aaf0804239efe2521844cddcb417e  
**Algorithm:** deeplearning  
**Actions:** Predict, Clone model, Inspect

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training Frame</td>
<td>allyears2k_headers.hex</td>
<td>Training frame</td>
</tr>
<tr>
<td>Response Column</td>
<td>isDeepDelayed</td>
<td>Response column</td>
</tr>
<tr>
<td>DO Classification</td>
<td>true</td>
<td>Convert the response column to an enum (forcing a classification instead of a regression) if needed.</td>
</tr>
<tr>
<td>MAX_AFTER_BALANCE_SIZE</td>
<td>infinity</td>
<td>Maximum relative size of the training data after balancing class counts (can be less than 1.0)</td>
</tr>
<tr>
<td>SEED</td>
<td>881048674552106414</td>
<td>Seed for random numbers (affects sampling). Note: only reproducible when running single threaded</td>
</tr>
<tr>
<td>LOSS</td>
<td>MeanSquare</td>
<td>Loss function</td>
</tr>
<tr>
<td>MAX_AFTER_BALANCE_SIZE</td>
<td>infinity</td>
<td>Maximum relative size of the training data after balancing class counts (can be less than 1.0)</td>
</tr>
<tr>
<td>REPLICATE_TRAINING_DATA</td>
<td>false</td>
<td>Replicate the entire training dataset onto every node for faster training on small datasets</td>
</tr>
</tbody>
</table>

### H2O Flow

- **Flow Demo!**
  - Flow: H2O Flow
  - Edit: Edit
  - View: View
  - Format: Format
  - Run: Run
  - Help: Help

### Outline

- Welcome to H2O Flow!
- Flow has a hybrid user interface assist
- importFiles
- importFiles ['../smalldata/airline','../smalldata/benchmarks']
- setupParse ['nfs://Users/prithv/...']
- parseRaw srcs ['nfs://Users/prithv/...']
- getJob '$8381c188253832d4fffffff'
- getFrame 'allyears2k_headers.hex'
- inspect getColumnSummary 'allyears2k_headers.hex'
- plot data: inspect 'distribution'
- grid inspect 'distribution', getFrame
- grid inspect 'summary', getFrame
- grid inspect 'characteristics', getFrame
- grid inspect 'data', getFrame
- buildModel
- buildModel 'deeplearning', {'train...'}
- getModel 'DeepLearningModel__89aaf0804239efe2521844cddcb417e'
- predict 'DeepLearningModel__89aaf0804239efe2521844cddcb417e'
- inspect getPrediction 'DeepLearning...'
- grid inspect 'metrics', getPrediction
- grid inspect 'prediction', getPrediction

### References
## Scalability of Predictive Modeling Algorithms

Ing. Tomáš Frýda

### FAKE GAME
- Base Models
- Ensembles
- Evolution

### H2O
- Overview
- Architecture
- Usage
- Related Software
- Hyper-parameter Optimization
- SMBO
- Implementation
- Experiments
- Results
- References

### H2O Flow

**Flow Demo!**

---

```groovy
grid inspect "metrics", getPredictions model: "DeepLearningModel__89aaf0804239efe2521844cddcbb417e", frame: "allyears2k_headers.hex"
```

#### Metrics for the selected predictions

<table>
<thead>
<tr>
<th>CRITERIA</th>
<th>THRESHOLD</th>
<th>F1</th>
<th>F2</th>
<th>F0POINTS</th>
<th>ACCURACY</th>
<th>ERROR</th>
<th>PRECISION</th>
</tr>
</thead>
<tbody>
<tr>
<td>maximum F1</td>
<td></td>
<td>0.68657443</td>
<td>0.8468044</td>
<td>0.5801671</td>
<td>0.525058</td>
<td>0.47494203</td>
<td>0.525058</td>
</tr>
<tr>
<td>maximum F2</td>
<td></td>
<td>0.68657443</td>
<td>0.8468044</td>
<td>0.5801671</td>
<td>0.525058</td>
<td>0.47494203</td>
<td>0.525058</td>
</tr>
<tr>
<td>maximum F0point5</td>
<td></td>
<td>0.68657443</td>
<td>0.8468044</td>
<td>0.5801671</td>
<td>0.525058</td>
<td>0.47494203</td>
<td>0.525058</td>
</tr>
<tr>
<td>maximum Accuracy</td>
<td></td>
<td>0.68657443</td>
<td>0.8468044</td>
<td>0.5801671</td>
<td>0.525058</td>
<td>0.47494203</td>
<td>0.525058</td>
</tr>
<tr>
<td>maximum Precision</td>
<td></td>
<td>0.0014023929</td>
<td>0.0063689488</td>
<td>0.004002683</td>
<td>0.015820755</td>
<td>0.47662467</td>
<td>0.52375303</td>
</tr>
<tr>
<td>maximum Recall</td>
<td></td>
<td>0.68657443</td>
<td>0.8468044</td>
<td>0.5801671</td>
<td>0.525058</td>
<td>0.47494203</td>
<td>0.525058</td>
</tr>
<tr>
<td>maximum Specificity</td>
<td></td>
<td>0.0014023929</td>
<td>0.0063689488</td>
<td>0.004002683</td>
<td>0.015820755</td>
<td>0.47662467</td>
<td>0.52375303</td>
</tr>
<tr>
<td>maximum absolute MCC</td>
<td></td>
<td>0.00034111855</td>
<td>0.37031785</td>
<td>0.3361132</td>
<td>0.4122729</td>
<td>0.43464914</td>
<td>0.56535083</td>
</tr>
<tr>
<td>minimizing maxper class Error</td>
<td></td>
<td>0.00029882442</td>
<td>0.45200276</td>
<td>0.44506764</td>
<td>0.4591574</td>
<td>0.43910593</td>
<td>0.5608941</td>
</tr>
</tbody>
</table>

---

### Results

Connections: 0
Scalability of Predictive Modeling Algorithms

Ing. Tomáš Frýda

FAKE GAME
Base Models Ensembles Evolution

H2O
Overview Architecture Usage
Related Software
Hyper-parameter Optimization SMBO Implementation Experiments Results References

H2O Flow

```
dump getFrame "allyears2k_headers.hex"

```

```
> dump: {schema_version: 2, schema_name: FrameV2, schema_type: Frame, key: Object, off: 0, ...}
  schema_version: 2
  schema_name: FrameV2
  schema_type: Frame
  key: {name: allyears2k_headers.hex}
    off: 0
    len: 100
    checksum: 1989216397660266000
    rows: 43978
    byteSize: 1939554
    isText: false
  default_pcts: [0.01, 0.1, 0.25, 0.3333333333333333, 0.5, ...]
  columns: [Object, Object, Object, Object, ...]
    0: {label: Year, missing: 0, zeros: 0, pins: 0, ninfs: 0, ...}
    1: {label: Month, missing: 0, zeros: 0, pins: 0, ninfs: 0, ...}
    2: {label: DayOfMonth, missing: 0, zeros: 0, pins: 0, ninfs: 0, ...}
    3: {label: DayOfWeek, missing: 0, zeros: 0, pins: 0, ninfs: 0, ...}
    4: {label: DayOfWeek, missing: 0, zeros: 0, pins: 0, ninfs: 0, ...}
      label: DepTime
      missing: 1086
      zeros: 1086
      pins: 0
      ninfs: 0
      mins: [1, 1, 1, 1, 4]
      max: [2400, 2400, 2400, 2400, 2359]
      mean: 1345.8466613820758
      sigma: 465.3408991242345
```
Related Software

- Sparkling Water
- Deep Water
- Steam — H2O deployment
Sparkling Water
Deep Water

Deep Water Architecture

Data Science API: R, Python, Flow GUI

REST API
Web server

H2O
Java
Execution Engine

H2O RPC

other comm.

H2O
Java
Execution Engine

H2O
Java
Execution Engine

Scala
Spark

Node 1

Node N

TensorFlow/mxnet/Caffe
C++

GPU
CPU

GPU
CPU

H2O.ai

FAKE GAME
Base Models
Ensembles
Evolution

H2O
Overview
Architecture
Usage
Related Software

Hyper-parameter
Optimization
SMBO
Implementation
Experiments
Results
References

Scalability of
Predictive
Modeling
Algorithms

Ing. Tomáš Frýda
Hyper-Parameter Optimization

- Grid Search
- Random Search
- Bayesian optimization (SMAC)
Sequential Model-based Bayesian Optimization (SMBO)

1. evaluate random configuration and add it to the probabilistic model

2. select promising configuration based on probabilistic model using an acquisition function\(^2\)

3. evaluate the configuration

4. add the new configuration to the probabilistic model

5. go to step 2

\(^2\)usually Expected Improvement \(EI(x) = E(\max\{0, f_{t+1}(x) - f(x^+)\}|M_t)\)
Instances of SMBO

- Gaussian Process based SMBO
  - no obvious way how to deal with categorical parameters
- Tree-structured Parzen Estimator (TPE)
- Sequential Model-based Algorithm Configuration (SMAC)
- Hyperband
Tree-structured Parzen Estimator

\[ p(x|y) = \begin{cases} 
  l(x) & \text{if } y < y^* \\
  g(x) & \text{if } y \geq y^* 
\end{cases} \]

- easy to sample space of promising values
- EI is proportional to \( \left( \gamma + \frac{g(x)}{l(x)}(1 - \gamma) \right)^{-1} \)

\[ y^* \text{ is chosen as some quantile (e.g., } p(y < y^*) = 0.15 = \gamma) \]
Sequential Model-based Algorithm Configuration

- based upon ROAR — an racing optimization algorithm
- uses random forest as a probabilistic model
- usage of random forest makes it easy to use user-defined cost metric
- configuration to be evaluated is selected by following process
  1. take 10 best previously seen configurations
  2. initialize local search (using one-exchange neighbourhood for categorical, and four random neighbours for numerical variables)
  3. merge resulting 10 best configurations with 10,000 randomly sampled configurations
  4. sort by their EI
  5. interleave with uniformly sampled configurations
Hyperband

- SMBO with enhanced selection and evaluation phase
- uses information from training phase of a model that is being optimized and eventually stops it if it doesn’t converge well $\Rightarrow$ explores more space using the same amount of resources
- iteratively discards the worse half of evaluated configurations
Implementation

- integration of FAKE GAME into H2O framework
- creation of benchmarking environment
  - written in Python
  - supports
    - all supervised machine learning algorithms in H2O
    - H2O Ensemble (implemented in R, based on SuperLearner package)
    - Hyper-Parameter optimization using Random Search and SMAC
  - configurable using YAML and Python
Experiments

- 2 datasets with binomial response class
  - Higgs
  - Airline - 4 different scenarios
- 20+ models benchmarked on each of 5 scenarios
- Hyper-Parameter optimization on each dataset
Overview of Results

**Figure: Higgs dataset**

**Figure: Airline – predicting IsDepDelayed**
Decision Boundary on Airline dataset

**ENS - SigmoidNorm**, probability of class

**ENS - SigmoidNorm**, decision boundary

**DRF**, probability of class

**DRF**, decision boundary
Decision Boundary on Airline dataset

**GBM**, probability of class

**Deep Learning**, probability of class

**GBM**, decision boundary

**Deep Learning**, decision boundary
Hyper-Parameter Optimization
Conclusion

- Successfully integrated FAKE GAME into H2O and created benchmarking environment
- Experiments took over 2000 hours (wall clock), used 12 CPUs and 16 GiB of RAM
- Experiments show that
  - newly integrated FAKE GAME can find better models than those previously present in H2O
  - H2O’s auto-tuning yields good results by default
- Results of those experiments were submitted, as part of an article, to be published in Machine Learning
0xdata,
https://www.slideshare.net/0xdata/rf-brighttalk
https://www.slideshare.net/0xdata/deep-water-gpu-deep-learning-for-h2o-arno-candel

J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl,
“Algorithms for Hyper-Parameter Optimization,”

F. Hutter, H. H. Hoos, and K. Leyton-Brown,
“Sequential model-based optimization for general algorithm configuration”,

L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar,
“Efficient Hyperparameter Optimization and Infinitely Many Armed Bandits,”