Scalable Machine Learning to Exploit Big Data for Knowledge Discovery

Una-May O’Reilly
MIT

MIT ILP-EPOCH Taiwan Symposium
Big Data: Technologies and Applications
Lots of Data Everywhere

Knowledge Mining Opportunities
In an ICU environment, physiologic data is collected at high frequency but is ignored because of the need for immediate focus of attention.

### MIMIC Waveform database

<table>
<thead>
<tr>
<th>Feature</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total size</td>
<td>4 Terra Bytes</td>
</tr>
<tr>
<td>Waveform types</td>
<td>22</td>
</tr>
<tr>
<td>Signal sampling frequency</td>
<td>125 samples/sec</td>
</tr>
<tr>
<td>Number of samples</td>
<td>500m</td>
</tr>
</tbody>
</table>
GigaBeats Project

Informative Micro-structure (features)

Latent Variable Hidden Conditions

Interpretable Forecasting Rules

Beats DB

Feature Extraction

Feature Discovery

State Space Modeling

Consensus Driven Clustering

Hidden Condition Indicators

SCALE

ECSTAR

Scalable Machine Learning
Machine Learning Primer

Exemplars

Data

Training set

Learner

Model or Classifier

Testing set

new observation

prediction class label pattern
Agenda
Distributed Computation + Scalable Machine Learning
SCALE
FlexGP
EC-Star
SCALE
SCALE System Layer

Algorithm unification
Constrained Parameter Sweep

Networked Task Management (server side)

Networked Task Management (client side)
DCAP Protocol

Task Server

Task List

Request for Task

Task Results

DCAP

Classifier

Client
SCALE Demonstration: WD-1

Forecasting ABP

- 10 levels
- 95K exemplars
- 67K/28K training/test split
- 7 dimensions
  - Stats, trends on MAP
- 1225 learner tasks (10 fold cross validation)
- 80 nodes, ~2 days, ~4000 node hours

<table>
<thead>
<tr>
<th>Classifier</th>
<th>Number of Instances</th>
<th>Errors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Neural Networks</td>
<td>720</td>
<td>0</td>
</tr>
<tr>
<td>Discrete Bayes</td>
<td>18</td>
<td>0</td>
</tr>
<tr>
<td>Naive Bayes</td>
<td>64</td>
<td>0</td>
</tr>
<tr>
<td>Support Vector Machines (SVM)</td>
<td>66</td>
<td>34</td>
</tr>
<tr>
<td>Decision Trees</td>
<td>324</td>
<td>0</td>
</tr>
</tbody>
</table>
SCALE: WD-1 Running time

Elapsed Time Plot

Time (s) vs. x 10^4

- Neural Network
- Decision Tree
- SVM
- Discrete Bayes
- Naive Bayes

Classifier Types

Neural Network  DecisionTree  SVM  DiscreteBayes  NaiveBayes
SCALE: WD-1 Accuracy results

F1 Score over all Classifiers, All Classes
SCALE: WD-1 Comparison SVM v DT

SVM

Decision Tree

F1 Score

class

F1 Score

class
Scaling up: from SCALE to FlexGP

SCALE
- Modest 10’s of features
- Assumes all training data fits into RAM

FlexGP
- 100’s of features
- Big Data
- Big Data requires multidimensional factoring, filtering then fusion
FlexGP ML Layer

FACTORING

$\Pi$: Probability of feature, objective function, operator

$\mathcal{D}$: factoring of the data
FlexGP Learner

Goal: Model $y = f(x_1, x_2, \ldots, x_n)$

Form GP Trees

$$\{+, -, *, /, \log, \sqrt{}, \exp\}$$

$$\{x_1, x_2, \ldots, x_n\}$$

Execute trees

New trees

Selection and variation of trees

$$\hat{y}$$

$$y_{true}$$

Transparent expression

$$\left(2.2 - \left(\frac{x_2}{11}\right)\right) + (7 \times \cos(x_1))$$

Genetic Programming Symbolic Regression
FlexGP Filter and Fusion

\[ x_1 \sin(x_5) + x_2 \sqrt{x} \]
\[ \cos(x_4)/\sin(x_2) + \sqrt{x_3 - x_4} \]
\[ \frac{x_1}{\exp(x_2)} + x_5 x_3 + \frac{x_2}{x_3} \]
\[ \frac{\cos(x_4)}{\tan(x_2) + x_2} + \sqrt{x_3} \]

Filter to select diverse models

X

\[ \overline{X} \rightarrow y \]

Fusion to derive an ensemble prediction

Adaptive Regression Mixing

FlexGP Overview
Factoring is Better

- Each learning from 100% of data
  - 412,00 exemplars
  - 21 hours each

- Each learning from 10% of data
  - 2 hours each

- Fused factored models
FlexGP: Data Factoring Size Study

MSE vs. Size of Training Dataset

Sweet Spot
36000 exemplars

Dataset | $|D_{gp}|$ | $|D_{f}|$ | $|D_t|$ | Total | $|x_j|$ | Range of $z$
--- | --- | --- | --- | --- | --- | ---
MSD | 398,924 | 67204 | 49436 | 515564 | 90 | [1922 2011]
Resource and System Management Layer

Completely decentralized launch
- Start up
  - Cascading launch is decentralized and allows elastic retraction and expansion

Completely decentralized network support
- IP discovery and launch of gossip protocol added to start up cascade
  - Launch IP discovery and ongoing gossip protocol enables communication network among nodes at algorithm level
  - Support for Monitoring/reporting/harvesting current best

Remarks
- Essential design for resilience to node failure
  - Launch or running node loss will not halt computation,
  - Lost launch branch or node can be integrated seamlessly
  - Node loss in a communication network won’t break the network

Statistical oversight of learning algorithm’s execution parms and data
- Distribution for parms and data added to start up cascade, negotiated between parent and child at launch of child

FlexGP
# System Layer: SCALE vs FlexGP

<table>
<thead>
<tr>
<th>SCALE:</th>
<th>FlexGP</th>
</tr>
</thead>
<tbody>
<tr>
<td>– LAMPs pre-defines the tasks</td>
<td>– Autonomous task specification</td>
</tr>
<tr>
<td>– Every learner has to know IP of task handler</td>
<td>– Learners gossip to learn each others’ IP</td>
</tr>
<tr>
<td>– Task handler is a bottleneck and central point of failure</td>
<td>– No central task handler or point of failure</td>
</tr>
</tbody>
</table>
FlexGP System Layer

Introduction
FlexGP Launch
ECStar

- **Goal:** compute very cost effectively on *VAST* number of nodes...with a lot of training data
  - Runs on thousand to 10’Ks 100K’s million nodes
  - Vast requires cost effective -> volunteer

- **Domain:** learn from time series
  - Finance, medical signals domain

- **Solution is strategy or classifier expressed as rule sets**
Use Case #1

EC-Star Paradigm
Digital Directed Evolution of Models

- Data selection
- Updated variables
- New objectives
- Current models
EC-Star

BigData factoring strategy

- Use fewer training samples and cull poor models
- Increase training samples on better models
- Local and distributed randomization
- Oversampling on superior models

EC-Star Divide and Conquer
# System Layer Comparison

<table>
<thead>
<tr>
<th></th>
<th>Scale</th>
<th>FlexGP</th>
<th>EC-Star</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ML domain</strong></td>
<td>Classification</td>
<td>Regression</td>
<td>Rule Learning</td>
</tr>
<tr>
<td><strong>Resource Scale</strong></td>
<td>10’s to 100</td>
<td>100’s to 1000</td>
<td>10^3 to 10^6</td>
</tr>
<tr>
<td><strong>Resource Type</strong></td>
<td>Cloud</td>
<td>Cloud</td>
<td>Volunteer and</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Dedicated</td>
</tr>
<tr>
<td><strong>Fusion</strong></td>
<td>External</td>
<td>External</td>
<td>Integrated</td>
</tr>
<tr>
<td><strong>Local Algorithm</strong></td>
<td>Different</td>
<td>Same</td>
<td>Same</td>
</tr>
<tr>
<td><strong>Server:Client ratio</strong></td>
<td>1: many</td>
<td>Decentralized</td>
<td>Few: many</td>
</tr>
</tbody>
</table>
## Scalable Machine Learning Comparison

<table>
<thead>
<tr>
<th></th>
<th>SCALE</th>
<th>FlexGP</th>
<th>EC-Star</th>
</tr>
</thead>
<tbody>
<tr>
<td>Factor</td>
<td>Algorithm</td>
<td>Algorithm and Data</td>
<td>Data: Under to oversampling</td>
</tr>
<tr>
<td>Filter</td>
<td></td>
<td>Correlation Accuracy</td>
<td>Layered competition</td>
</tr>
<tr>
<td>Fuse</td>
<td></td>
<td>Non-parametric output space approaches</td>
<td>Migration and ancestral properties</td>
</tr>
</tbody>
</table>
Automation

- "In the end, the biggest bottleneck is not data or CPU cycles, but human cycles."

Looking Forward
ML requires a lot of Human Effort

- Domain Knowledge Analysis and Transfer
- Problem Definition
- Data Preconditioning
- Feature Identification
- Feature Extraction
- Algorithm Selection
- Algorithm Customization
- Parameter Selection
- Training and Test Data Selection
- Results Evaluation
- Solution Deployment

Looking Forward
Compressing the ML Endeavor

- From desktop
  - Multi-core
- To GPU
- To Cloud
  Seamlessly!
  Rapidly!
  Flexibly!
  Scalably!

- Preprocessing of data
- Feature Engineering
- Structuring into Data Packages
- Generation of decision boundaries
- Generating a library of conditions
- Modify or build an evaluation/fitness function
  - Specify ML system parms
  - Run ML system
- Analyze results
Wrap Up

DCAP is open source: https://github.com/byterial/dcap
FlexGP is documented in publications and thesis
  Owen Derby, MEng, 2013
Thanks to…

• ALFA group members
  – Large team of students
  – Postdoc: Dr. Erik Hemberg
  – Research Scientist: Dr. Kalyan Veeramachaneni
• CSAIL Quanta cloud
• Our collaborators and sponsors