Scientific Big Data Analytics by HPC -

Parallel and Scalable Machine Learning on JURECA

Dr.-Ing. Morris Riedel et al.
Head of Research Group, Juelich Supercomputing Centre
Adjunct Associated Professor, University of Iceland

JURECA Porting and Tuning Workshop
6th - 8th June 2016, Jülich Supercomputing Centre
Learning from Data – Different to Simulation Science

1. Some pattern exists
2. **No exact mathematical formula**
3. Data exists

Idea ‘Learning from Data‘ shared with a wide variety of other disciplines
- E.g. signal processing, etc.

- Statistical data mining and machine learning is a very broad subject and goes from very abstract theory to extreme practice (‘rules of thumb’)

Using HPC resources like JURECA useful
- Reasoning: parallel I/O, mature inter-process communication (MPI), OpenMP, GPGPUs, etc.
Context Juelich Supercomputing Centre

- Research data-intensive science and engineering applications
- Explore computing that is more intertwined with data analysis
- Tackle Inverse Problems
- Sharing, re-use, towards reproducability

Federated Data Management, Preservation, Security & Access

Communities
Exascale

Parallel Data Analytics
Data Mining Methods
Scientific Community Applications
Generic Data Methods

Machine Learning Algorithms
Data Analysis Tools

Data Science

Parallelization Demand

Serial data analysis techniques/tools increasingly show limits

- Traditional methods still relevant, but need to scale for ‘big data’
- Big Data: e.g. high number of dimensions/classes or ‘data points’

Concrete ‘big data’: large health data

Concrete ‘big data’: large earth science data

Classification++

Regression++

Data Mining Methods

Data Science

Machinae Learning Algorithms

Parallel Data Analytics

Scientific Community Applications

Generic Data Methods

Data Analysis Tools
Clustering Technique

Classification
- Groups of data exist
- New data classified to existing groups

Clustering
- No groups of data exist
- Create groups from data close to each other

Regression
- Identify a line with a certain slope describing the data
Selected Clustering Methods

K-Means Clustering – Centroid based clustering
- Partitions a data set into K distinct clusters (centroids can be artificial)

K-Medoids Clustering – Centroid based clustering (variation)
- Partitions a data set into K distinct clusters (centroids are actual points)

Sequential Agglomerative hierarchic nonoverlapping (SAHN)
- Hierarchical Clustering (create tree-like data structure → ‘dendrogram’)

Clustering Using Representatives (CURE)
- Select representative points / cluster; as far from one another as possible

Density-based spatial clustering of applications + noise (DBSCAN)
- Reasoning: density similarity measure helpful in our driving applications
- Assumes clusters of similar density or areas of higher density in dataset
Technology Review of Available ‘Big Data ‘Tools

- JSC courses 'parallel programming' useful: Introduction to parallel programming with MPI and OpenMP, Advanced parallel programming with MPI and OpenMP

<table>
<thead>
<tr>
<th>Technology</th>
<th>Platform</th>
<th>Approach</th>
<th>Analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td>HPDBSCAN (authors implementation)</td>
<td>C; MPI;</td>
<td>Parallel, hybrid, DBSCAN</td>
<td></td>
</tr>
<tr>
<td>Apache Mahout</td>
<td>Java; Hadoop</td>
<td>K-means variants, spectral, no DBSCAN</td>
<td></td>
</tr>
<tr>
<td>Apache Spark/MLlib</td>
<td>Java; Spark</td>
<td>Only k-means clustering, No DBSCAN</td>
<td></td>
</tr>
<tr>
<td>scikit-learn</td>
<td>Python</td>
<td>No parallelization strategy for DBSCAN</td>
<td></td>
</tr>
<tr>
<td>Northwestern University PDSDBSCAN-D</td>
<td>C++; MPI; OpenMP</td>
<td>Parallel DBSCAN</td>
<td></td>
</tr>
</tbody>
</table>

DBSCAN

DBSCAN Algorithm
- Introduced 1996 by Martin Ester et al.
- Groups number of similar points into clusters of data
- Similarity is defined by a distance measure (e.g. euclidean distance)

Distinct Algorithm Features
- Clusters a variable number of clusters
- Forms arbitrarily shaped clusters
- Identifies outliers/noise

Understanding Parameters for MPI/OpenMP tool
- Looks for a similar points within a given search radius → Parameter $\epsilon$
- A cluster consist of a given minimum number of points → Parameter $minPoints$
Parallel & Scalable HP-DBSCAN Tool on JURECA (1)

Parallelization Strategy
- Smart ‘Big Data‘ Preprocessing into Spatial Cells (‘indexed‘)
- OpenMP standalone
- MPI (+ optional OpenMP hybrid)

Preprocessing Step
- Spatial indexing and redistribution according to the point localities
- Data density based chunking of computations

Computational Optimizations
- Caching of point neighborhood searches
- Cluster merging based on comparisons instead of zone reclustering

Parallel & Scalable HP-DBSCAN Tool on JURECA (2)

Usage via jobsript
- Using job scheduler
- Important: module load hdf5/1.8.13
- Important: library gcc-4.9.2/lib64
- np = number of processors
- t = number of threads
- Uses parallel/IO

DBSCAN Parameters

module load hdf5/1.8.13
export LD_LIBRARY_PATH=/home/zam/analytic/bigdata/hpdbscan/gcc-4.9.2/lib64:$LD_LIBRARY_PATH
DBSCAN=/home/zam/analytic/bigdata/hpdbscan/jsc_mpi/dbscan
SMALLBREMENDATA=/home/zam/analytic/bigdata/hpdbscan/jsc_mpi/mriruns/bremenSmall.h5

cd /home/zam/analytic/bigdata/hpdbscan/jsc_mpi/mriruns
mpiexec -rp 1 $DBSCAN -es 300 -m 100 -t 12 $SMALLBREMENDATA

JSC courses ‘Parallel I/O’ useful: Parallel I/O and portable data formats
Clustering Applications – Large Point Clouds

‘Big Data‘: 3D/4D laser scans
- Captured by robots or drones
- Millions to billion entries
- Inner cities (e.g. Bremen inner city)
- Whole countries (e.g. Netherlands)

Selected Scientific Cases
- Filter noise to better represent real data
- Grouping of objects (e.g. buildings)
- Different level of details (e.g. trees)

Research activities in collaboration with the Netherlands e-Science Centre & TU Delft
Clustering Applications – Many Time Series & Events

Earth Science Data Repository

- Time series measurements (e.g. salinity)
- Millions to billions of data items/locations
- Less capacity of experts to analyse data

Selected Scientific Case

- Data from Koljöfjords in Sweden (Skagerrak)
- Each measurement small data, but whole sets are ‘big data‘
- Automated water mixing event detection & quality control (e.g. biofouling)
- Verification through domain experts

Research activities in collaboration with MARUM in Bremen and University of Gothenburg
Clustering Applications – Neuro Science Image Analysis

Large Brain Images
- High resolution scans of post mortem brains
- Rare ‘groundtruth available’

Selected Scientific Case
- Cell nuclei detection and tissue clustering
- Detect various layers (colored)
- Layers seem to have different density distribution of cells
- Extract cell nuclei into 2D/3D point cloud
- Cluster different brain areas by cell density

Research activities in collaboration with Institute of Medicine and Neuroscience (T. Dickscheid)
Classification Technique

- **Classification**
  - Groups of data exist
  - New data classified to existing groups

- **Clustering**
  - No groups of data exist
  - Create groups from data close to each other

- **Regression**
  - Identify a line with a certain slope describing the data
Selected Classification Methods

Perceptron Learning Algorithm – simple linear classification
- Enables binary classification with ‘a line’ between classes of separable data

Support Vector Machines (SVMs) – non-linear (‘kernel’) classification
- Enables non-linear classification with maximum margin (best ‘out-of-the-box’)

Reasoning: achieves often better results than other methods in tackled application domain

Decision Trees & Ensemble Methods – tree-based classification
- Grows trees for class decisions, ensemble methods average n trees

Artificial Neural Networks (ANNs) – brain-inspired classification
- Combine multiple linear perceptrons to a strong network for non-linear tasks

Naive Bayes Classifier – probabilistic classification
- Use of the Bayes theorem with strong/naive independence between features
### Technology Review of Available ‘Big Data ‘Tools

- **JSC courses ‘GPU programming‘ useful: Vectorisation and portable programming using OpenCL, GPU programming with OpenACC, GPU programming with CUDA**

<table>
<thead>
<tr>
<th>Technology</th>
<th>Platform</th>
<th>Approach</th>
<th>Analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apache Mahout</td>
<td>Java; Hadoop</td>
<td></td>
<td>No parallelization strategy for SVMs</td>
</tr>
<tr>
<td>Apache Spark/MLlib</td>
<td>Java; Spark</td>
<td></td>
<td>Parallel linear SVMs (no multi-class)</td>
</tr>
<tr>
<td>Twister/ParallelSVM</td>
<td>Java; Twister; Hadoop 1.0</td>
<td></td>
<td>Parallel SVMs, open source; developer version 0.9 beta</td>
</tr>
<tr>
<td>scikit-learn</td>
<td>Python</td>
<td></td>
<td>No parallelization strategy for SVMs</td>
</tr>
<tr>
<td>piSVM 1.2 &amp; piSVM 1.3</td>
<td>C; MPI</td>
<td></td>
<td>Parallel SVMs; stable; not fully scalable</td>
</tr>
<tr>
<td>GPU LibSVM</td>
<td>CUDA</td>
<td></td>
<td>Parallel SVMs; hard to programs, early versions</td>
</tr>
<tr>
<td>pSVM</td>
<td>C; MPI</td>
<td></td>
<td>Parallel SVMs; unstable; beta version</td>
</tr>
</tbody>
</table>

SVMs

SVM Algorithm

- Introduced 1995 by C. Cortes & V. Vapnik et al.
- Creates a ‘maximal margin classifier’ to get future points (‘more often’) right and take advantage of kernel methods
- Uses quadratic programming & Lagrangian method with $N \times N$

![Graph showing SVM Algorithm](image)

$maximal\ margin\ classifier\ example$

$\mathcal{L}(\alpha) = \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} y_n y_m \alpha_n \alpha_m x_n^T x_m$

(kernel trick, quadratic coefficients – Computational Complexity & Big Data Impact)

(Linear example)

(linear example)

(‘maximal margin classifier’ example)

$\min_{w, \xi_i, b} \left\{ \frac{1}{2} \|w\|^2 + C \sum_{i} \xi_i \right\}$

$y_i (w \cdot x_i - b) \geq 1 - \xi_i, \quad \xi_i \geq 0$

(use of soft-margin approach for better generalization)

(maximizing hyperplane turned into optimization problem, minimization, allow some errors)

Parallel and Scalable Machine Learning on JURECA
Parallel & Scalable piSVM Tool on JURECA (1)

Original parallel piSVM tool 1.2
- Open-source and based on libSVM library, C, 2011
- Message Passing Interface (MPI)
- New version appeared 2014-10 v. 1.3 (no major improvements)
- Lack of ‘big data‘ support (memory, layout, etc.)

Tuned scalable parallel piSVM tool 1.2.1
- Highly scalable version maintained by Juelich
- Based on original piSVM 1.2 tool
- Open-source (repository to be created)
- Optimizations: load balancing; MPI collectives
Usage via jobscript

- Using job scheduler
- np = number of processors;
- o/q = problem partitioning
- c = cost (soft margin SVM)
- g = RBF kernel parameter
- T = type of SVM (here C-SVC)
- Example: train phase submit

```bash
### location
PISVM=/homeb/zam/mriedel/pisvm-1.2/pisvm-1.2/pisvm-train

TRAINDATA=/homeb/zam/mriedel/bigdata/86-romeok/sdap_area_all_training.el

### submit
mpisec -np $NSLOTS $PISVM -o 1024 -q 512 -c 10000 -g 16 -t 2 -m 1024 -a 0 $TRAINDATA
```

- Submission of test phase similar but using labelled dataset + trained SVM model
Classification Applications – Remote Sensing Images

Challenges: high number of classes, less samples, mixed pixels

<table>
<thead>
<tr>
<th>Number</th>
<th>Class</th>
<th>Number of samples</th>
<th>Class</th>
<th>Number of samples</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Buildings</td>
<td>1720</td>
<td>Pasture</td>
<td>1039</td>
</tr>
<tr>
<td>2</td>
<td>Corn</td>
<td>1778</td>
<td>Pond</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>Corn?</td>
<td>16</td>
<td>Soybeans</td>
<td>939</td>
</tr>
<tr>
<td>4</td>
<td>Corn-EW</td>
<td>51</td>
<td>Soybeans?</td>
<td>89</td>
</tr>
<tr>
<td>5</td>
<td>Corn-NS</td>
<td>236</td>
<td>Soybeans-NS</td>
<td>111</td>
</tr>
<tr>
<td>6</td>
<td>Corn-CleanTill</td>
<td>1240</td>
<td>Soybeans-CleanTill</td>
<td>507</td>
</tr>
<tr>
<td>7</td>
<td>Corn-CleanTill-EW</td>
<td>2649</td>
<td>Soybeans-CleanTill?</td>
<td>273</td>
</tr>
<tr>
<td>8</td>
<td>Corn-CleanTill-NS</td>
<td>3968</td>
<td>Soybeans-CleanTill-EW</td>
<td>1180</td>
</tr>
<tr>
<td>9</td>
<td>Corn-CleanTill-NS-Irrigated</td>
<td>80</td>
<td>Soybeans-CleanTill-NS</td>
<td>1039</td>
</tr>
<tr>
<td>10</td>
<td>Corn-CleanTilled-NS</td>
<td>173</td>
<td>Soybeans-CleanTilled-NS</td>
<td>224</td>
</tr>
<tr>
<td>11</td>
<td>Corn-MinTill</td>
<td>105</td>
<td>Soybeans-MinTill</td>
<td>152</td>
</tr>
<tr>
<td>12</td>
<td>Corn-MinTill-EW</td>
<td>563</td>
<td>Soybeans-MinTill-EW</td>
<td>267</td>
</tr>
<tr>
<td>13</td>
<td>Corn-MinTill-NS</td>
<td>886</td>
<td>Soybeans-MinTill-NS</td>
<td>495</td>
</tr>
<tr>
<td>14</td>
<td>Corn-NoTill</td>
<td>438</td>
<td>Soybeans-NoTill</td>
<td>216</td>
</tr>
<tr>
<td>15</td>
<td>Corn-NoTill-EW</td>
<td>121</td>
<td>Soybeans-NoTill-EW</td>
<td>253</td>
</tr>
<tr>
<td>16</td>
<td>Corn-NoTill-NS</td>
<td>569</td>
<td>Soybeans-NoTill-NS</td>
<td>93</td>
</tr>
<tr>
<td>17</td>
<td>Fescue</td>
<td>11</td>
<td>Swampy Area</td>
<td>58</td>
</tr>
<tr>
<td>18</td>
<td>Grass</td>
<td>115</td>
<td>River</td>
<td>311</td>
</tr>
<tr>
<td>19</td>
<td>Grass/Trees</td>
<td>233</td>
<td>Trees?</td>
<td>58</td>
</tr>
<tr>
<td>20</td>
<td>Hay</td>
<td>113</td>
<td>Wheat</td>
<td>498</td>
</tr>
<tr>
<td>21</td>
<td>Hay-Alfa</td>
<td>219</td>
<td>Woods</td>
<td>6356</td>
</tr>
<tr>
<td>22</td>
<td>Hay-Alfa</td>
<td>226</td>
<td>Woods?</td>
<td>14</td>
</tr>
<tr>
<td>23</td>
<td>Lake</td>
<td>22</td>
<td>Woods?</td>
<td>14</td>
</tr>
<tr>
<td>24</td>
<td>NotCropped</td>
<td>194</td>
<td>Woods</td>
<td>14</td>
</tr>
<tr>
<td>25</td>
<td>Oats</td>
<td>174</td>
<td>Woods</td>
<td>14</td>
</tr>
<tr>
<td>26</td>
<td>Oats</td>
<td>34</td>
<td>Woods</td>
<td>14</td>
</tr>
</tbody>
</table>


(1) Scenario ‘unprocessed data’
(2) Scenario ‘preprocessed data’

Challenges in automation

remote sensing cube & ground reference
Classification Applications – SDAP Feature Extraction

Key importance

- Use feature extraction/enhancement
- Apply dimensionality reduction techniques (e.g. principle components)

Example: Self-Dual Attribute Profile (SDAP)

- Use different filtering strategies for morphological attributes as additional inputs
- Sequentially apply attribute filters on tree-based image representations

Research activities in collaboration with University of Iceland (G. Cavallaro, J.A. Benediktsson)

Classification Applications – Lower time to Solution

Example dataset: high number of classes & mixed pixels

- Parallelization benefits: major speed-ups, ~interactive (<1 min) possible

<table>
<thead>
<tr>
<th>Scenarios</th>
<th>‘unprocessed data’</th>
<th>‘pre-processed data’</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training</td>
<td>(1) Scenario</td>
<td>(2) Scenario</td>
</tr>
<tr>
<td>Time (in min)</td>
<td>(1) Scenario</td>
<td>(2) Scenario</td>
</tr>
<tr>
<td>Processing</td>
<td>(a)</td>
<td>(a)</td>
</tr>
<tr>
<td></td>
<td>(1,14,06)</td>
<td>(1,33,8)</td>
</tr>
<tr>
<td></td>
<td>(4,6,54)</td>
<td>(4,2,04)</td>
</tr>
<tr>
<td></td>
<td>(8,3,42)</td>
<td>(8,1,02)</td>
</tr>
<tr>
<td></td>
<td>(16,2,16)</td>
<td>(16,1,02)</td>
</tr>
<tr>
<td></td>
<td>(64,1,03)</td>
<td>(64,0,31)</td>
</tr>
<tr>
<td></td>
<td>(80,6,55)</td>
<td>(80,6,31)</td>
</tr>
</tbody>
</table>

- Testing time (in min)

<table>
<thead>
<tr>
<th>Scenarios</th>
<th>‘unprocessed data’</th>
<th>‘pre-processed data’</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time (in min)</td>
<td>(1) Scenario</td>
<td>(2) Scenario</td>
</tr>
<tr>
<td>Processing</td>
<td>(b)</td>
<td>(b)</td>
</tr>
<tr>
<td></td>
<td>(1,228,46)</td>
<td>(1,47,16)</td>
</tr>
<tr>
<td></td>
<td>(2,115,05)</td>
<td>(2,24,26)</td>
</tr>
<tr>
<td></td>
<td>(4,66,52)</td>
<td>(4,14,07)</td>
</tr>
<tr>
<td></td>
<td>(8,31,41)</td>
<td>(8,7,12)</td>
</tr>
<tr>
<td></td>
<td>(16,4,16)</td>
<td>(16,4,03)</td>
</tr>
<tr>
<td></td>
<td>(64,4,46)</td>
<td>(64,1,34)</td>
</tr>
<tr>
<td></td>
<td>(80,4,09)</td>
<td>(80,1,05)</td>
</tr>
</tbody>
</table>

‘big data’ is not always better data

<table>
<thead>
<tr>
<th>manual &amp; serial activities (in min)</th>
</tr>
</thead>
<tbody>
<tr>
<td>kPCA</td>
</tr>
<tr>
<td>------</td>
</tr>
<tr>
<td>(1) Scenario</td>
</tr>
<tr>
<td>(2) Scenario</td>
</tr>
</tbody>
</table>

Overall Accuracy (%)

<table>
<thead>
<tr>
<th>(1) Scenario</th>
<th>(2) Scenario</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of features</td>
<td>200</td>
</tr>
<tr>
<td>Overall Accuracy (%)</td>
<td>40.68</td>
</tr>
</tbody>
</table>

Classification Applications – Cross-Validation Benefits

2x benefits of parallelization (shown in n-fold cross validation)

- (1) Compute parallel; (2) Do all cross-validation runs in parallel (all cells)
- Evaluation between Matlab (aka ‘serial laptop’) & parallel piSVM (80 cores)
- 10x cross-validation (RBF kernel parameter $\gamma$ and $C$, aka ‘gridsearch’)

(1) Scenario ‘unprocessed data’, 10xCV serial: accuracy (min)

<table>
<thead>
<tr>
<th>$\gamma$/C</th>
<th>1</th>
<th>10</th>
<th>100</th>
<th>1000</th>
<th>10000</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>27.30 (109.78)</td>
<td>34.59 (124.46)</td>
<td>39.05 (107.85)</td>
<td>37.38 (116.29)</td>
<td>37.20 (121.51)</td>
</tr>
<tr>
<td>4</td>
<td>29.24 (98.18)</td>
<td>37.75 (85.31)</td>
<td>38.91 (113.87)</td>
<td>38.36 (119.12)</td>
<td>38.36 (118.98)</td>
</tr>
<tr>
<td>8</td>
<td>31.31 (109.95)</td>
<td>39.68 (118.28)</td>
<td>39.06 (112.99)</td>
<td>39.06 (190.72)</td>
<td>39.06 (872.27)</td>
</tr>
<tr>
<td>16</td>
<td>33.37 (126.14)</td>
<td>39.46 (171.11)</td>
<td>39.19 (206.66)</td>
<td>39.19 (181.82)</td>
<td>39.19 (146.98)</td>
</tr>
<tr>
<td>32</td>
<td>34.61 (179.04)</td>
<td>38.37 (202.30)</td>
<td>38.37 (231.10)</td>
<td>38.37 (240.36)</td>
<td>38.37 (278.02)</td>
</tr>
</tbody>
</table>

First Result: best parameter set from 118.28 min to 4.09 min
Second Result: all parameter sets from ~3 days to ~2 hours

(2) Scenario ‘pre-processed data’, 10xCV serial: accuracy (min)

<table>
<thead>
<tr>
<th>$\gamma$/C</th>
<th>1</th>
<th>10</th>
<th>100</th>
<th>1000</th>
<th>10000</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>48.90 (18.81)</td>
<td>65.01 (19.57)</td>
<td>73.21 (20.11)</td>
<td>75.55 (22.53)</td>
<td>74.42 (21.21)</td>
</tr>
<tr>
<td>4</td>
<td>57.53 (16.82)</td>
<td>70.74 (13.94)</td>
<td>75.94 (13.53)</td>
<td>76.04 (14.04)</td>
<td>74.06 (15.55)</td>
</tr>
<tr>
<td>8</td>
<td>64.18 (18.30)</td>
<td>74.45 (15.04)</td>
<td>77.00 (14.41)</td>
<td>75.78 (14.65)</td>
<td>74.58 (14.92)</td>
</tr>
<tr>
<td>16</td>
<td>68.37 (23.21)</td>
<td>76.20 (21.88)</td>
<td>76.51 (20.69)</td>
<td>75.32 (19.60)</td>
<td>74.72 (19.66)</td>
</tr>
<tr>
<td>32</td>
<td>70.17 (34.45)</td>
<td>75.48 (34.76)</td>
<td>74.88 (34.05)</td>
<td>74.08 (34.03)</td>
<td>73.84 (38.78)</td>
</tr>
</tbody>
</table>

First Result: best parameter set from 14.41 min to 1.02 min
Second Result: all parameter sets from ~9 hours to ~35 min

(1) Scenario ‘unprocessed data’, 10xCV parallel: accuracy (min)

<table>
<thead>
<tr>
<th>$\gamma$/C</th>
<th>1</th>
<th>10</th>
<th>100</th>
<th>1000</th>
<th>10000</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>27.26 (3.38)</td>
<td>34.49 (3.35)</td>
<td>39.16 (5.35)</td>
<td>37.56 (11.46)</td>
<td>37.57 (13.02)</td>
</tr>
<tr>
<td>4</td>
<td>29.12 (3.34)</td>
<td>37.58 (3.38)</td>
<td>38.91 (6.02)</td>
<td>38.43 (7.47)</td>
<td>38.43 (7.47)</td>
</tr>
<tr>
<td>16</td>
<td>33.36 (4.09)</td>
<td>39.61 (4.56)</td>
<td>39.25 (5.06)</td>
<td>39.25 (5.27)</td>
<td>39.25 (5.10)</td>
</tr>
<tr>
<td>32</td>
<td>34.61 (5.13)</td>
<td>38.37 (5.30)</td>
<td>38.36 (5.43)</td>
<td>38.36 (5.49)</td>
<td>38.36 (5.28)</td>
</tr>
</tbody>
</table>

(2) Scenario ‘pre-processed data’, 10xCV parallel: accuracy (min)

<table>
<thead>
<tr>
<th>$\gamma$/C</th>
<th>1</th>
<th>10</th>
<th>100</th>
<th>1000</th>
<th>10000</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>75.26 (1.02)</td>
<td>65.12 (1.03)</td>
<td>73.18 (1.33)</td>
<td>75.76 (2.35)</td>
<td>74.53 (4.40)</td>
</tr>
<tr>
<td>4</td>
<td>57.60 (1.03)</td>
<td>70.88 (1.02)</td>
<td>75.87 (1.03)</td>
<td>76.01 (1.33)</td>
<td>74.06 (2.35)</td>
</tr>
<tr>
<td>8</td>
<td>64.17 (1.02)</td>
<td>74.52 (1.03)</td>
<td>77.02 (1.02)</td>
<td>75.79 (1.04)</td>
<td>74.42 (1.34)</td>
</tr>
<tr>
<td>16</td>
<td>68.57 (1.33)</td>
<td>76.07 (1.33)</td>
<td>76.40 (1.34)</td>
<td>75.26 (1.05)</td>
<td>74.53 (1.34)</td>
</tr>
<tr>
<td>32</td>
<td>70.21 (1.33)</td>
<td>75.38 (1.34)</td>
<td>74.69 (1.34)</td>
<td>73.91 (1.47)</td>
<td>73.73 (1.33)</td>
</tr>
</tbody>
</table>

First Result: best parameter set from 14.41 min to 1.02 min
Second Result: all parameter sets from ~9 hours to ~35 min

Summary

Scientific Peer Review is essential to progress in the field

- Work in the field needs to be guided & steered by communities
- NIC Scientific Big Data Analytics (SBDA) first step (learn from HPC)
- Towards enabling reproducability by uploading runs and datasets

Selected SBDA by HPC benefit from parallelization

- Statistical data mining techniques able to reduce ‘big data‘ (e.g. PCA, etc.)
- Benefits in n-fold cross-validation & raw data, less on preprocessed data
- Two codes available to use and maintained @JSC: HPDBSCAN, piSVM
- HPDBSCAN and piSVM work on JURECA (less useful on JUQUEEN)

Number of ‘Data Analytics et al.’ technologies incredible high

- (Less) open source & working versions available, often paper studies
- Evaluating approaches hard: HPC, map-reduce, Spark, SciDB, MaTex, …
- Collection of codes in Juelich Machine Learning Library (JUML) started…
References


Acknowledgements

PhD Student Gabriele Cavallaro, University of Iceland
Tómas Philipp Runarsson, Kristján Jonasson, Jón Atli Benediktsson, University of Iceland

Timo Dickscheid, Markus Axer, Stefan Köhnen, Tim Hütz, Institute of Neuroscience & Medicine, Forschungszentrum Juelich

Selected Members of the Research Group on High Productivity Data Processing

Ahmed Shiraz Memon
Mohammad Shahbaz Memon
Markus Goetz
Christian Bodenstein
[Philipp Glock (moved to INM)]
Matthias Richerzhagen
Talk soon available at: www.morrisriedel.de/talks