Parallel & Scalable Machine Learning
Introduction to Machine Learning Algorithms

Dr. Gabriele Cavallaro
Postdoctoral Researcher, Juelich Supercomputing Centre, Germany

Ernir Erlingsson
PhD Student, University of Iceland & DEEP-EST Application Engineer

LECTURE 12

Parallelization Benefits and Cross-Validation Practicals

March 8th, 2018
JSC, Germany
Outline of the Course

1. Introduction to Machine Learning Fundamentals  
2. PRACE and Parallel Computing Basics  
3. Unsupervised Clustering and Applications  
4. Unsupervised Clustering Challenges & Solutions  
5. Supervised Classification and Learning Theory Basics  
6. Classification Applications, Challenges, and Solutions  
7. Support Vector Machines and Kernel Methods  
8. Practicals with SVMs  
9. Validation and Regularization Techniques  
10. Practicals with Validation and Regularization  
11. Deep Learning Introduction  
12. Parallelization Benefits and Cross-Validation Practicals

Day One – beginner  
Day Two – moderate  
Day Three – expert
Outline

- Parallelization Benefits
  - Regularization Parameters Revisited
  - Cross-Validation Technique Revisited
  - First Level of Parallelization in piSVM
  - Second Level of Parallelization in GridSearch

- Cross-Validation Practicals
  - Gridsearch as a team
  - Assign teams combinations of parameters
  - Parameter C and Gamma on Indian Pines
  - Gridsearch Benefits Revisited
Outline

- Short Course Summary
  - Machine Learning Fundamentals
  - Unsupervised Clustering with HPDBSCAN
  - Supervised Classification with PISVM
  - Statistical Learning Theory Basics
  - Regularization & Validation
Regularization Revisited & Kernel Parameter Gamma

- $C = 0$ (too restrictive, potentially bad for $E_{out}$)
  - No budget/costs for violations: comparable to maximal margin classifier
  - Further constraint: only works in linearly separable cases (less in practice)

- $C > 0$ (flexible option, better for $E_{out}$)
  - No more than $C$ data points can be on the wrong side of the hyperplane (‘how much misclassifications allowed‘)
  - Reasoning: if an observation is on the wrong side then $\epsilon_i > 1$

- Kernel Parameter Gamma
  - RBF Kernel element, example Iris Dataset

[3] SVM and Parameters
Validation Technique – Cross-Validation – Revisited

- 10-fold cross validation is mostly applied in practical problems by setting $K = N/10$ for real data
- Having $N/K$ training sessions on $N - K$ points each leads to long runtimes ($\rightarrow$ use parallelization)

- Leave-one-out
  - N training sessions on $N - 1$ points each time

- Leave-more-out
  - Break data into number of folds
  - $N/K$ training sessions on $N - K$ points each time (fewer training sessions than above)
  - Example: ‘10-fold cross-validation’ with $K = N/10$ multiple times ($N/K$)
    (use 1/10 for validation, use 9/10 for training, then another 1/10 ... $N/K$ times)
Parallelization Benefit: Lower-Time-To-Solution

- Major speed-ups; ~interactive (<1 min); same accuracy;

### (1) Scenario 'unprocessed data'
- Training time (in min)

<table>
<thead>
<tr>
<th>Number of cores</th>
<th>(1,14,06)</th>
<th>(2,01,10)</th>
<th>(4,65,4)</th>
<th>(6,36,8,2)</th>
<th>(8,00,55)</th>
</tr>
</thead>
</table>

### (2) Scenario 'pre-processed data'
- Training time (in min)

<table>
<thead>
<tr>
<th>Number of cores</th>
<th>(1,33,48)</th>
<th>(2,23,6)</th>
<th>(4,20,4)</th>
<th>(6,10,2)</th>
<th>(8,0,31)</th>
</tr>
</thead>
</table>

### (1) Scenario 'unprocessed data'
- Testing time (in min)

<table>
<thead>
<tr>
<th>Number of cores</th>
<th>(1,29,46)</th>
<th>(2,11,50)</th>
<th>(4,05,62)</th>
<th>(8,31,41)</th>
<th>(16,51,6)</th>
<th>(32,8,41)</th>
<th>(64,4,46)</th>
<th>(8,0,409)</th>
</tr>
</thead>
</table>

### (2) Scenario 'pre-processed data'
- Testing time (in min)

<table>
<thead>
<tr>
<th>Number of cores</th>
<th>(1,47,16)</th>
<th>(2,24,26)</th>
<th>(4,14,07)</th>
<th>(8,7,12)</th>
<th>(16,4,05)</th>
<th>(32,2,05)</th>
<th>(64,1,34)</th>
<th>(8,0,1,05)</th>
</tr>
</thead>
</table>

---

<table>
<thead>
<tr>
<th>manual &amp; serial activities (in min)</th>
<th>kPCA</th>
<th>ESap</th>
<th>SWfe</th>
<th>10x CSV</th>
<th>Training</th>
<th>Test</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1) Scenario</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>$4.47 \times 10^3$</td>
<td>10.45</td>
<td>71.08</td>
<td>$4.55 \times 10^3$</td>
</tr>
<tr>
<td>(2) Scenario</td>
<td>5</td>
<td>15.38</td>
<td>1</td>
<td>529.55</td>
<td>1.37</td>
<td>23.25</td>
<td>575.55</td>
</tr>
</tbody>
</table>

---

'big data' is not always better data

<table>
<thead>
<tr>
<th>Number of features</th>
<th>200</th>
<th>30</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overall Accuracy (%)</td>
<td>40.68</td>
<td>77.96</td>
</tr>
</tbody>
</table>

(cf. Importance of feature engineering above)


(aka first level of parallelism)
Parallelization Benefit: Parallel 10-Fold Cross-Validation

- Example: 2 Parameters, 10-fold cross-validation
  - 2 x benefits of parallelization possible in a so-called ‘gridsearch’
  - (1) Compute parallel; (2) Do all cross-validation runs in parallel (all cells)
  - Evaluation between Matlab (aka ‘serial laptop’) & parallel (80 cores)

<table>
<thead>
<tr>
<th>γ/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></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>32</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

In Matlab one after another

- (2) Scenario ‘pre-processed data’, 10xCV serial: accuracy (min)
  - (1) First Result: best parameter set from 14.41 min to 1.02 min
  - (2) Second Result: all parameter sets from ~9 hours to ~35 min

- Evaluation between Matlab (aka ‘serial laptop’) & parallel (80 cores)

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

‘(1) each cell inherent parallel’

‘(2) all cells in parallel’

- 10-fold cross-validation achieves parallelization benefits (1) in each grid cell and (2) across all cells

Parallelization Summary – aka ‘GridSearch’

- **Parallelization benefits** are enormous for complex problems
  - Enables feasibility to tackle **extremely large datasets & high dimensions**
  - Provides functionality for a high number of classes (e.g. #k SVMs)
  - Achieves a massive reduction in time → **lower time-to-solution**

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

<table>
<thead>
<tr>
<th>γ/C</th>
<th>1</th>
<th>10</th>
<th>100</th>
<th>1000</th>
<th>10 000</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>32</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Second level of parallelism

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>γ/C</th>
<th>1</th>
<th>10</th>
<th>100</th>
<th>1000</th>
<th>10 000</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>32</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Second level of parallelism

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>γ/C</th>
<th>1</th>
<th>10</th>
<th>100</th>
<th>1000</th>
<th>10 000</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>32</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Second level of parallelism

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

<table>
<thead>
<tr>
<th>γ/C</th>
<th>1</th>
<th>10</th>
<th>100</th>
<th>1000</th>
<th>10 000</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>32</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Second level of parallelism

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

Complex Application Example in Industry – Netflix

- ~2009 - Netflix Prize Challenge 2009
  - **Data:** Netflix company provided data to learn from previous movie rentals
  - **Challenge:** Improve Netflix in-house movie recommender system
  - **Prize:** 1,000,000 US $ for team with 10% improvements
  - **Approaches:** Machine learning algorithms and collaborative filterings
  - **Winner:** Prize received by working with Artificial Neural Network (ANNs)

Complex Application Example in Industry – Windpower

- Predictive & Instant Maintenance Workforce Management

Business Value Driver
- Increase average time between inspections
- Decrease lost power generation factor
- Decrease cost of spare parts

Big Data Impact
- Ad-hoc analysis on large data volumes to predict, monitor and optimize performance and component breakdown
- Enable “Process-to-device”

Value Chain
- Generation

Process
- Turbine Maintenance

Value

Feasibility

Phase
1. 2. 3.

Lead User(s)

Slide courtesy of Dr. S. Fischer, Global Head of Applied Research – SAP AG, Germany SDIL
Complex Application Examples in Science & Engineering

- **Classification** of Abnormalities in Brain MRI Images
  - Using **Support Vector Machines (SVMs)**
  - ‘Classify images between normal and abnormal along with type of disease depending upon features.’

  ![Brain MRI Images](image1)

- **Classification** of buildings from multi-spectral satellite data
  - Using **Support Vector Machines (SVMs)**
  - Classify land cover using image data & data preprocessing methods

  ![Satellite Image](image2)
JURECA System – SSH Login

- Use your account train004 - train050
- Windows: use mobaxterm or putty
- UNIX: ssh trainXYZ@jureca.fz-juelich.de
- Example

    adminuser@linux-8djg:~$ ssh train001@jureca.fz-juelich.de
    Warning: the ECDSA host key for 'jureca.fz-juelich.de' differs from the key for the IP address '134.94.33.9'
    Offending key for IP in /home/adminuser/.ssh/known_hosts:12
    Matching host key in /home/adminuser/.ssh/known_hosts:19
    Are you sure you want to continue connecting (yes/no)? yes
    Last login: Mon Aug 21 14:29:03 2017 from zam2036.zam.kfa-juelich.de
    ***************************************************************************************
    *                              Welcome to JURECA                                  *
    *                                                                                *
    * Information about the system, latest changes, user documentation and FAQs: *
    *                      http://www.fz-juelich.de/ias/jsc/jureca                     *
    ***************************************************************************************
    ### Known Issues ###
    *
    * An up-to-date list of known issues on the system is maintained at
    *    http://www.fz-juelich.de/ias/jsc/jureca-known-issues
    *
    * Open issues:
    *    - Intel compiler error with std::valarray and
    *
    * optimized headers, added 2016-03-20
    *

➤ Remember to use your own trainXYZ account in order to login to the JURECA system
**Indian Pines Dataset – Preprocessing**

Corrected by JPL

- **1417 × 617 pixels (~600 MB)**
- **200 bands (20 discarded, with low SNR)**
- **58 classes (6 discarded, with ≤ 100 samples)**

<table>
<thead>
<tr>
<th>Class</th>
<th>Number of samples</th>
<th>Class</th>
<th>Number of samples</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>training</td>
<td>test</td>
<td>number</td>
</tr>
<tr>
<td>1</td>
<td>Buildings</td>
<td>1720</td>
<td>13475</td>
</tr>
<tr>
<td>2</td>
<td>Corn</td>
<td>1778</td>
<td>16005</td>
</tr>
<tr>
<td>3</td>
<td>Corn?</td>
<td>16</td>
<td>142</td>
</tr>
<tr>
<td>4</td>
<td>Corn-EW</td>
<td>51</td>
<td>465</td>
</tr>
<tr>
<td>5</td>
<td>Corn-NS</td>
<td>236</td>
<td>2120</td>
</tr>
<tr>
<td>6</td>
<td>Corn-CleanTill</td>
<td>1240</td>
<td>11164</td>
</tr>
<tr>
<td>7</td>
<td>Corn-CleanTill-EW</td>
<td>2649</td>
<td>23837</td>
</tr>
<tr>
<td>8</td>
<td>Corn-CleanTill-NS</td>
<td>3968</td>
<td>35710</td>
</tr>
<tr>
<td>9</td>
<td>Corn-CleanTill-NS-Irrigated</td>
<td>80</td>
<td>720</td>
</tr>
<tr>
<td>10</td>
<td>Corn-CleanTill-NS?</td>
<td>173</td>
<td>1555</td>
</tr>
<tr>
<td>11</td>
<td>Corn-MiniTill</td>
<td>105</td>
<td>944</td>
</tr>
<tr>
<td>12</td>
<td>Corn-MiniTill-EW</td>
<td>563</td>
<td>5066</td>
</tr>
<tr>
<td>13</td>
<td>Corn-MiniTill-NS</td>
<td>886</td>
<td>7976</td>
</tr>
<tr>
<td>14</td>
<td>Corn-NoTill</td>
<td>438</td>
<td>3943</td>
</tr>
<tr>
<td>15</td>
<td>Corn-NoTill-EW</td>
<td>121</td>
<td>1085</td>
</tr>
<tr>
<td>16</td>
<td>Corn-NoTill-NS</td>
<td>209</td>
<td>2110</td>
</tr>
<tr>
<td>17</td>
<td>Fescue</td>
<td>11</td>
<td>102</td>
</tr>
<tr>
<td>18</td>
<td>Grass</td>
<td>115</td>
<td>1032</td>
</tr>
<tr>
<td>19</td>
<td>Grass/Trees</td>
<td>233</td>
<td>2098</td>
</tr>
<tr>
<td>20</td>
<td>Hay</td>
<td>113</td>
<td>1015</td>
</tr>
<tr>
<td>21</td>
<td>Hay?</td>
<td>219</td>
<td>1966</td>
</tr>
<tr>
<td>22</td>
<td>Hay-Alfalfa</td>
<td>226</td>
<td>2032</td>
</tr>
<tr>
<td>23</td>
<td>Lake</td>
<td>32</td>
<td>202</td>
</tr>
<tr>
<td>24</td>
<td>NorCropped</td>
<td>194</td>
<td>1746</td>
</tr>
<tr>
<td>25</td>
<td>Oats</td>
<td>174</td>
<td>1568</td>
</tr>
<tr>
<td>26</td>
<td>Oats?</td>
<td>34</td>
<td>301</td>
</tr>
</tbody>
</table>


(non-linearly separable) dataset
Publicly Available Datasets – Location

- **Indian Pines Dataset Raw and Processed**

  **Indian pines: raw and processed**

  by [Unknown]

  Dec 22, 2016

  Last updated at Jan 11, 2018

  **Abstract:** 1) Indian raw: 1417x614x200 (training 10% and test) 2) Indian processed: 1417x614x30 (training 10% and test)

  **PID:** 11304/7e8eece8e-ad61-11e4-ac7e-860aa0063df | Copy

  **Files**

<table>
<thead>
<tr>
<th>Name</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>indian_processed.test.el</code></td>
<td>105.59MB</td>
</tr>
<tr>
<td><code>indian_processed_training.el</code></td>
<td>11.73MB</td>
</tr>
<tr>
<td><code>indian_raw.el</code></td>
<td>74713MB</td>
</tr>
<tr>
<td><code>indian_raw_training.el</code></td>
<td>83.01MB</td>
</tr>
</tbody>
</table>

  ![Indian Pines Raw and Processed](image)

  ![EUDAT](image)

  ![B2SHARE](image)
Exercises – Gridsearch Cross-Validation Runs – Indian Pines
## Exercises – Cross-Validation Runs – Gridsearch Rome

<table>
<thead>
<tr>
<th>GAMMA</th>
<th>C</th>
<th>0.1</th>
<th>1</th>
<th>10</th>
<th>100</th>
<th>1000</th>
<th>10000</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>32</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>64</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Homework:
Please fill the grid with your values and find out the best parameters.
## Exercises – Cross-Validation Runs – Gridsearch Indian Pines

<table>
<thead>
<tr>
<th>GAMMA</th>
<th>C</th>
<th>0.1</th>
<th>1</th>
<th>10</th>
<th>100</th>
<th>1000</th>
<th>10000</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>32</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>64</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Lecture 1 – Introduction to Machine Learning Fundamentals

- Machine Learning Prerequisites
  1. Some pattern exists
  2. No exact mathematical formula
  3. Data exists

- Linearly separable dataset Iris
  - Perceptron Model as hypothesis
    (simplest linear learning model, linearity in learned weights $w_i$)
  - Understood Perceptron Learning Algorithm (PLA)

- Learning Approaches
  - Supervised Learning
  - Unsupervised Learning
  - Reinforcement Learning
Lecture 2 – PRACE and Parallel Computing Basics

- **PRACE**
  - High Performance Computing (HPC)
  - Pan-European supercomputing infrastructure
  - Offers HPC resources on leading edge capability systems
  - Peer-review of research proposals

- **JURECA (Mon, Wed) & JUROPA3 (Tue)**
  - Tutorial HPC systems
  - Accounts train004-train050

- **Scheduler & Job scripts**
  - Define parallel job properties
  - Reservation in job scripts change daily
  - Job submit with sbatch <jobscript>; check status with squeue –u <userid>

[7] LLView Tool

Network interconnection important
Lecture 3 – Unsupervised Clustering and Applications

- Unsupervised Clustering
  - Input $\mathbf{x} = x_1, \ldots, x_d$
  - No output (unsupervised)
  - Data $(x_1), \ldots, (x_N)$

- K-Means Clustering
  - Problem of assigning $K$ as numbers of clusters
  - Simple, works only sometimes

- DBSCAN Clustering
  - Finds arbitrary shapes and numbers of clusters
  - Still two parameters
  - Parallel version scales for big pointcloud datasets

[4] An Introduction to Statistical Learning
Lecture 4 – Unsupervised Clustering, Challenges & Solutions

- Unsupervised Clustering
  - K-Means & K-Median
  - DBSCAN very effective
  - Applications in Context
  - Parameter Changes minPoints & Epsilon

- Point Cloud Datasets
  - 3D/4D laser scans
  - Cities, Buildings, etc.
  - Bremen small & big datasets
  - Big Data: Whole Countries (e.g. Netherlands)
Lecture 4 – Python Script HDF5 → PCL for PCL Viewer

```python
import h5py as h5
import numpy as np
import sys

if len(sys.argv) < 2:
    INPUT="bremen.h5"
else:
    INPUT = sys.argv[1]
FILE = "bremenClustered.pcd"

print"loading H5"
bremen = h5.File("bremenSmall.h5")
points = bremen["DBSCAN"]
clusters = bremen["Clusters"]
colors = bremen["COLORS"]

print "Transform to numpy"
points = np.array(points)
clusters = np.array(clusters)
colors = np.array(colors)

#print "Remove Noise"
#points = points[clusters!=0]
#clusters = clusters[clusters!=0]

data = np.concatenate((points, colors.reshape((-1,1))), axis=1)
data = np.concatenate((data, clusters.reshape((-1,1))), axis=1)

clusters[clusters!=0]=1
data = np.concatenate((data, clusters.reshape((-1,1))), axis=1)

print "Write PCD"
"H5toPCD.py" 49L, 1028C
```
Lecture 4 – Point Cloud Library – Viewer

[train001@jrl12 pisvm-1.2.1]$ module spider pcl

PCL:

Description:
The Point Cloud Library (PCL) is a standalone, large scale, open project for 2D/3D image and point cloud processing.

Versions:
PCL/1.8.1-Python-2.7.14

For detailed information about a specific "PCL" module (including how to load the modules) use the module's full name. For example:

$ module spider PCL/1.8.1-Python-2.7.14

[train001@jrl04 ~]$ module load Intel/2018.0.128-GCC-5.4.0 ParaStationMPI/5.2.0-1
[train001@jrl04 ~]$ module load PCL

[train001@jrl06 ~]$ pcl_viewer
Lecture 5 – Supervised Classification & Learning Theory Basics

- **Supervised Classification**
  - Finding a **target distribution** for realistic learning situations
  - Assume **unknown probability distribution** over the input space
  - Hypothesis search with \( M \) models and we pick one

- **Statistical Learning Theory**
  
  \[
  \Pr \left[ | E_{in}(g) - E_{out}(g) | > \epsilon \right] \leq 2M e^{-2\epsilon^2 N}
  \]

  \[
  \Pr \left[ | E_{in}(g) - E_{out}(g) | > \epsilon \right] \leq 4m_{\mathcal{H}}(2N) e^{-1/8\epsilon^2 N}
  \]

  Hypothesis Set
  \[ \mathcal{H} = \{ h \}; \ g \in \mathcal{H} \]

  Final Hypothesis
  \[ g \approx f \]

  - **Learning Algorithm** (‘train a system’)
    \[ \mathcal{A} \]

  - **Probability Distribution**
    \[ P \text{ on } X \]

  - **Unknown Target Distribution**
    \[ f : X \to Y \] plus noise

  - **Probability Distribution**
    \[ P(y|x) \]

  - **Hypothesis Set**
    \[ \mathcal{H} = \{ h \}; \ g \in \mathcal{H} \]

  - **Final Hypothesis**
    \[ g \approx f \]

  - **Statistical Learning Theory**
    
    \[ \Pr \left[ | E_{in}(g) - E_{out}(g) | > \epsilon \right] \leq 2M e^{-2\epsilon^2 N} \]

    \[ \Pr \left[ | E_{in}(g) - E_{out}(g) | > \epsilon \right] \leq 4m_{\mathcal{H}}(2N) e^{-1/8\epsilon^2 N} \]
Lecture 6 – Classification Applications, Challenges & Solutions

- Remote Sensing Data
  - Pixel-wise spectral-spatial classifiers
  - Feature enhancements
  - Example of Self-Dual Attribute Profile (SDAP) technique

- Classification Challenges
  - Scalability, high dimensionality, etc.
  - Non-linearly seperable data
  - Overfitting as key problem in training machine learning models

![Image of classification challenges]

![Image of classification techniques]

![Graph showing error vs. training time]

Error vs. Training time graph showing the decrease in both training and generalization error with increasing training time, indicating the point of overfitting.
Lecture 7 – Support Vector Machines & Kernel Methods

- **Non-linear Transformations**
  - Use of a mapping function $\Phi$
  - Hyperplane in higher dimensional space possible
  - Mapping back corresponds to non-linear decision boundary in initial input or x space

- **Full Support Vector Machine**
  - Full = use of non-linear kernel
  - Take advantage of mapping into a higher-level/infinite space
  - Apply ‘kernel trick’
  - Kernels quantify similarity
  - Different trusted kernels available (RBF, polynomial, etc.)

\[
\sum \alpha_i y_i \mathbf{x}_i \cdot \mathbf{u}_i + b \geq 0 \quad \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{u}_i)
\]

\[
\mathcal{L} = \sum \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j
\]

\[
K(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)
\]

(trusted Kernel avoids to know Phi)
Lecture 8 – Practicals with SVMs

- Analysing Indian Pines Dataset
  - Remote Sensing Dataset
  - Raw & Processed
  - Using JUROPA3 system with MPI-based piSVM 1.2.1 tool

- Feature Engineering in Machine learning
  - Manual feature enhancements increased accuracy from ~44% to ~77%
  - Some work before ‘learning from data’ is important

Lecture 9 – Validation and Regularization Techniques

- ‘Combat’ Overfitting in Machine Learning
  - Two key approaches

- Validation
  - 10-fold cross-validation mostly applied in practice
  - Rule of thumb: 1/5 of data

- Regularization
  - Rule of thumb: Simpler model is usually better
  - Regularization approaches often inherent in algorithm / optimization
  - E.g. Parameter C as cost parameter in SVMs to violate the margin

Example: ‘10-fold cross-validation’ with $K = N/K$ multiple times ($N/K$)
(use 1/10 for validation, use 9/10 for training, then another 1/10 ... $N/K$ times)
Lecture 10 – Practicals with Validation and Regularization

- **10-fold Cross-Validation**
  - Validation estimates for out of sample performance only on training Dataset is remarkably well
  - Training Dataset is validation Dataset using 10-fold cross-validation
  - Test Dataset kept separate

- **Regularization**
  - Parameter C important to steer how much error allowed

---

### An Introduction to Statistical Learning

[1] An Introduction to Statistical Learning
Exercises – Cross-Validation Runs – Evaluate Gridsearch
Parallelization Benefit: Parallel 10-Fold Cross-Validation

- **Example:** 2 Parameters, 10-fold cross-validation
  - **2 x benefits of parallelization** possible in a so-called ‘gridsearch’
  - (1) Compute parallel; (2) Do all cross-validation runs in parallel (all cells)
  - Evaluation between Matlab (aka ‘serial laptop’) & parallel (80 cores)

<table>
<thead>
<tr>
<th>γ/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><strong>77.00</strong> (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>

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


![Parallelization Diagram]

- **10-fold cross-validation achieves parallelization benefits (1) in each grid cell and (2) across all cells**
Parallelization Summary

- **Parallelization benefits** are enormous for complex problems
  - Enables feasibility to tackle extremely large datasets & high dimensions
  - Provides functionality for a high number of classes (e.g. \( k \) SVMs)
  - Achieves a massive reduction in time \( \Rightarrow \) lower time-to-solution

---

#### (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>8</td>
<td>31.24 (3.38)</td>
<td>37.97 (4.09)</td>
<td>39.14 (5.45)</td>
<td>39.14 (5.42)</td>
<td>39.14 (5.43)</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>

**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 **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

---

Lecture Bibliography (1)

- [1] An Introduction to Statistical Learning with Applications in R. Online: http://www-bcf.usc.edu/~gareth/ISL/index.html
- [6] Indian Pines Raw and Processed Online: http://hdl.handle.net/11304/9ec5eac8-61b4-4617-ae1c-1f8c8cd3cd74