HPC for Big Remote Sensing Data Analytics (Support Vector Machines)

Dr. Gabriele Cavallaro
Postdoctoral Researcher
High Productivity Data Processing Group
Jülich Supercomputing Centre

Ernir Erlingsson
PhD student
University of Iceland

JÜLICH Forschungszentrum | JÜLICH SUPERCOMPUTING CENTRE

UNIVERSITY OF ICELAND
SCHOOL OF ENGINEERING AND NATURAL SCIENCES
FACULTY OF INDUSTRIAL ENGINEERING, MECHANICAL ENGINEERING AND COMPUTER SCIENCE

Outline

- Machine Learning Backgrounds
  - Big Remote Sensing Data
  - Machine Learning Basics
  - Supervised Learning using parallel SVM
- Parallel SVM
  - Adapting Software for HPC
  - Introducing PiSVM
  - The DEEP Projects
  - Running Multiple Tasks Simultaneously
- Practicals with PiSVM
  - Indian Pines dataset and Experimental Setup
  - Submit Train and Test Job Scripts
  - Cross Validation Exercise

Machine Learning Backgrounds

Big Data Motivation: Intertwine HPC and Machine Learning

- Rapid advances in data collection and storage technologies in the last decade
  - **Extracting useful information** is a challenge for ever increasing massive datasets
  - **Traditional data analysis methods cannot be used** in growing cases (e.g. **memory, speed**, etc.)

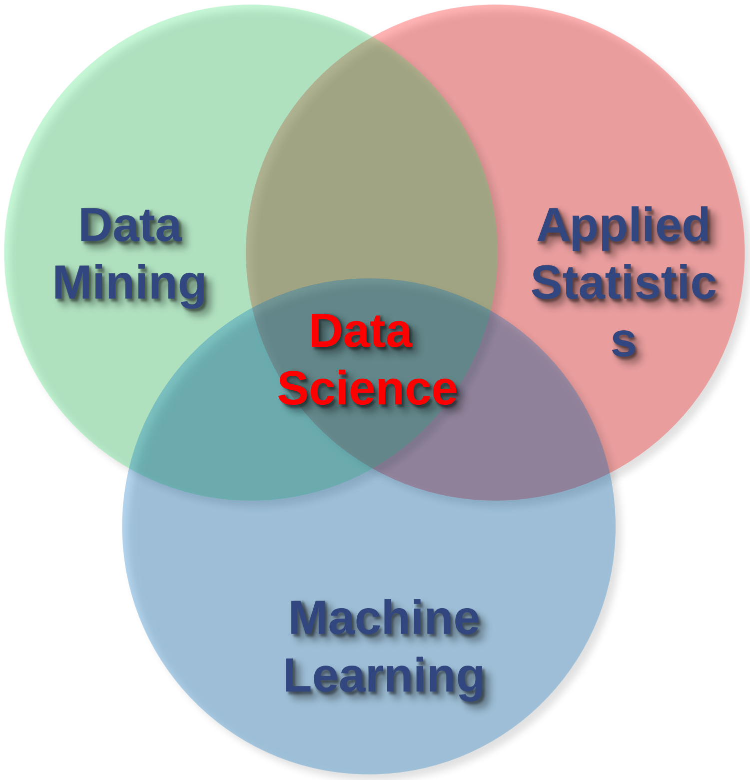
- Machine learning / Data Mining
  - Blends traditional analysis methods with complex algorithms for processing large volumes of data

Modified from [1] Introduction to Data Mining

- Machine Learning & **Statistical** Data Mining
  - Traditional **statistical** approaches are still very useful to consider

Before using HPC: Machine Learning Prerequisites

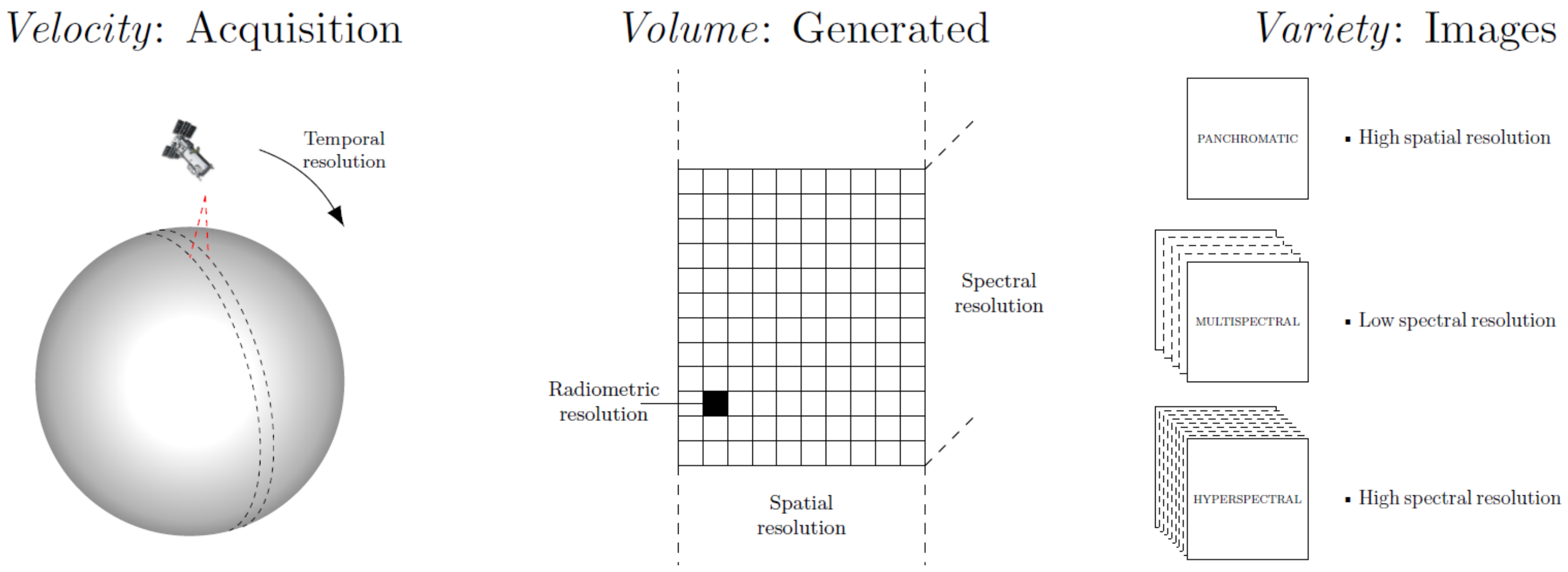
1. Some pattern exists
2. No exact mathematical formula
3. **Data exists**



- Idea **‘Learning from Data‘**
  - Shared with a wide variety of other disciplines
  - E.g., signal processing, data mining
- Challenge: Data is often complex
- Machine learning is a very broad subject and goes from very abstract theory to extreme practice (‘rules of thumb‘)

Big Remote Sensing Data

- Rapid advances in data collection and storage technologies in the last decade



- Extracting useful information is a challenge
- Traditional data analysis techniques cannot be used (e.g., memory, speed, etc.)

Sentinel 2 Mission

- **Platform:** Twin polar-orbiting satellites, phased at 180° to each other
- **Temporal resolution:** 5 days at the equator in cloud-free conditions

~23 TB data stored per day

Spatial Resolution (m)	Band Number	S2A Central Wavelength (nm)	S2A Bandwidth (nm)	S2B Central Wavelength (nm)	S2B Bandwidth (nm)
10	2	496.6	98	492.1	98
	3	560.0	45	559	46
	4	664.5	38	665	39
	8	835.1	145	833	133
20	5	703.9	19	703.8	20
	6	740.2	18	739.1	18
	7	782.5	28	779.7	28
	8a	864.8	33	864	32
	11	1613.7	143	1610.4	141
	12	2202.4	242	2185.7	238
60	1	443.9	27	442.3	45
	9	945.0	26	943.2	27
	10	1373.5	75	1376.9	76

[2] Earth Observation Mission Sentinel 2

- Images for mapping land-use change, land-cover change, biophysical variables
- Monitor the coastal and inland waters and help with risk and disaster mapping

Learning Approaches – What means Learning?

- Basic meaning of learning: ‘use a set of observations to uncover an underlying process’

- **Supervised Learning**
  - Majority of methods follow this approach when annotated data are available
  - Example: credit card approval based on previous customer applications

- **Unsupervised Learning**
  - Often applied before other learning → higher level data representation
  - Example: Coin recognition in vending machine based on weight and size

- **Reinforcement Learning**
  - Typical ‘human way‘ of learning
  - Example: toddler tries to touch a hot cup of tea (again and again)

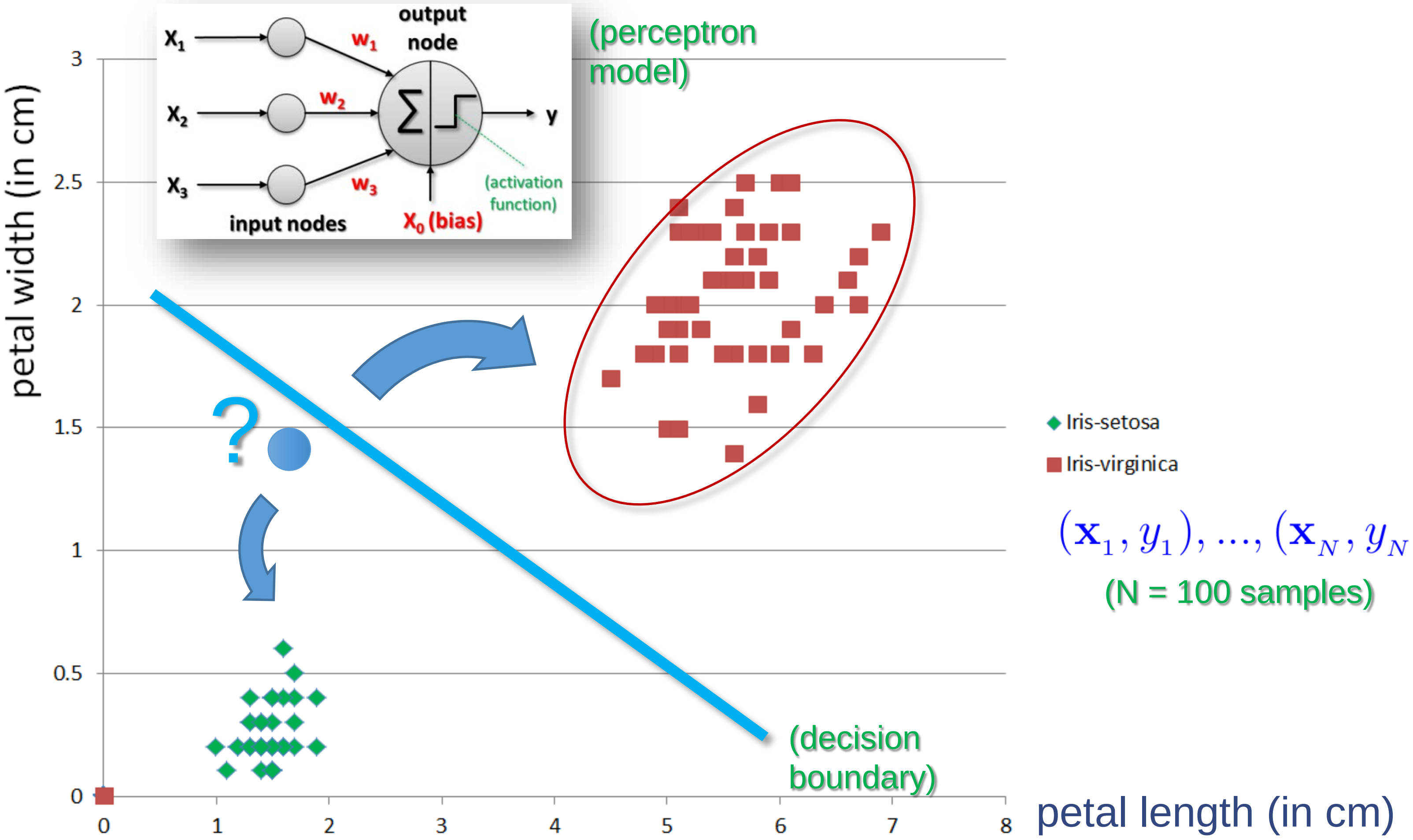
Supervised Learning

- Each observation of the predictor measurement(s) has **an associated response measurement**:
  - Input $\mathbf{x} = x_1, \ldots, x_d$
  - Output $y_i, i = 1, .., n$
  - Data $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)$
- Goal: fit a model that relates the response to the predictors
  - **Prediction**: Aims of accurately predicting the response for future observations
  - **Inference**: Aims to better understanding the relationship between the response and the predictors
- **Supervised learning approaches are used in classification algorithms such as SVMs**

[3] An Introduction to Statistical Learning

Supervised Learning Example

- The labels guide our learning process like a ‘supervisor‘ is helping us



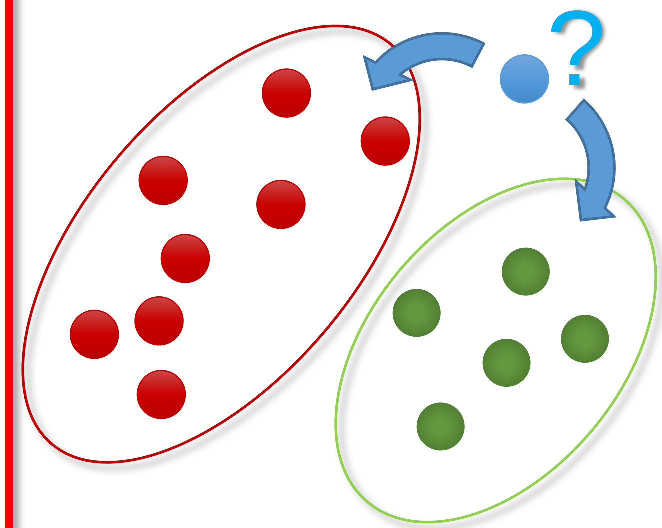
$(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)$

(N = 100 samples)

Methods Overview – Advanced Example

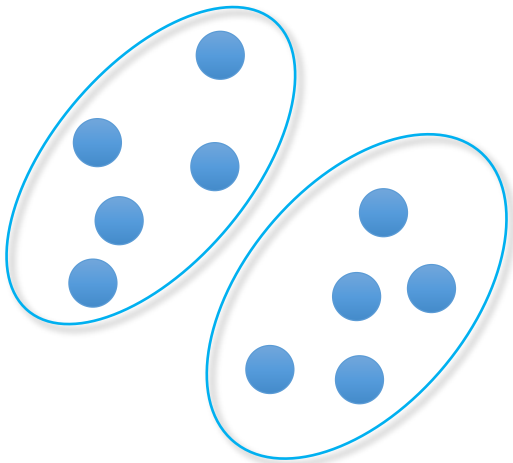
- Statistical data mining methods can be roughly categorized in 3 main classes
- Augmented with various techniques for data exploration, selection, or reduction

Classification



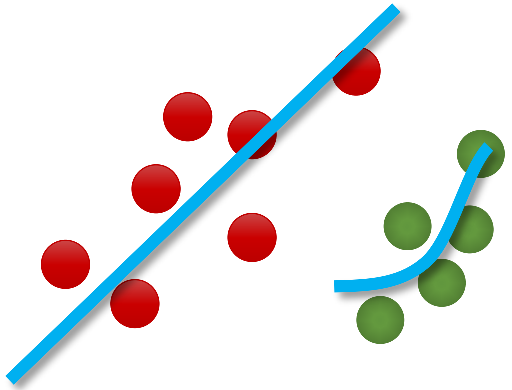
- Groups of data exist
- New data classified to existing groups

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

Term Support Vector Machines

- Support Vector Machines (SVMs) are a classification technique developed ~1990
- Term detailed refinement into **‘three separate techniques’**
  – Practice: applications mostly use the SVMs with kernel methods

[3] An Introduction to Statistical Learning

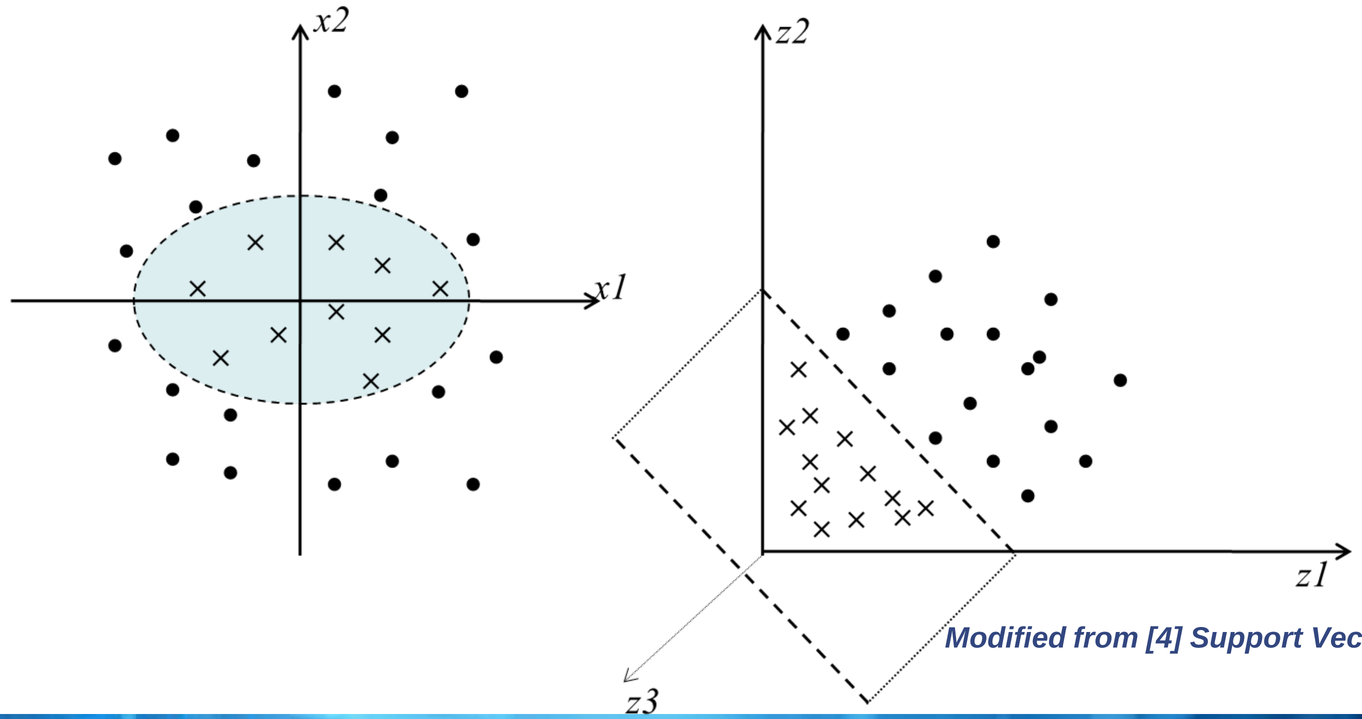
- **‘Maximal margin classifier‘**
  – A simple and intuitive classifier with a ‘best‘ linear class boundary
  – Requires that data is **‘linearly separable‘**

- **‘Support Vector Classifier‘**
  – Extension to the maximal margin classifier for non-linearly seperable data
  – Applied to a broader range of cases, idea of **‘allowing some error‘**

- **‘Support Vector Machines‘ → Using Non-Linear Kernel Methods**
  – Extension of the support vector classifier
  – Enables non-linear class boundaries & via **kernels**

Support Vector Machines

- In linear non-separable cases:
  - Map the data in higher dimensional feature space
  - With **Kernel Functions**, e.g., *Polynomial, Radial Basis Function* (RBF)
  - Fit a hyperplane, which is non-linear in the original feature space



Modified from [4] Support Vector Machines and Kernel Algorithms

SVM Model Selection with RBF kernel

- During the training process we have to estimate penalizing term C and kernel parameters (e.g., γ)

- *C:* regularization parameter to handle misclassified samples in non-separable cases, e.g., samples on the wrong side of the hyperplane (or within the margin). It controls the shape of the solution.
  - Large value of *C* might cause an over-fitting to the training data

- γ: inverse of the RBF standard deviation and used as similarity measure between two points
  - Small γ (RBF with large variance): two points are considered similar even if are far from each other
  - Large γ (RBF with small variance): two points are considered similar just if close to each other

- ***Grid search*** in feature space $C \times \gamma$ is widely used

- Error rate is determined by **cross validation**

Cross-Validation for Model Selection

- With N/K training sessions on N – K points each leads to long runtimes **(use parallelization)**

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

(leave 1 point out at each run → many runs)

1 | Training Examples $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)$ | $\mathcal{D}$

- Leave-more-out
  - Break data into number of folds (fewer training sessions than above)
  - N/K training sessions on **N – K points each time**
  - **Example: '10-fold cross-valdation' 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)

(generalization to leave k points out at each run)

K-fold | Training Examples $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)$ | $\mathcal{D}$

(practice to avoid bias & contamination: some rest for test as 'unseen data')

$\mathcal{D}$ (dataset)

Training Examples

$\mathcal{D}_1$	$\mathcal{D}_2$	$\mathcal{D}_3$	$\mathcal{D}_4$	$\mathcal{D}_5$	$\mathcal{D}_6$	$\mathcal{D}_7$	$\mathcal{D}_8$	$\mathcal{D}_9$	$\mathcal{D}_{10}$

(involved in training now) (involved in training now) (now is the current example run)

Parallel SVM

Adapting Software Applications for HPC

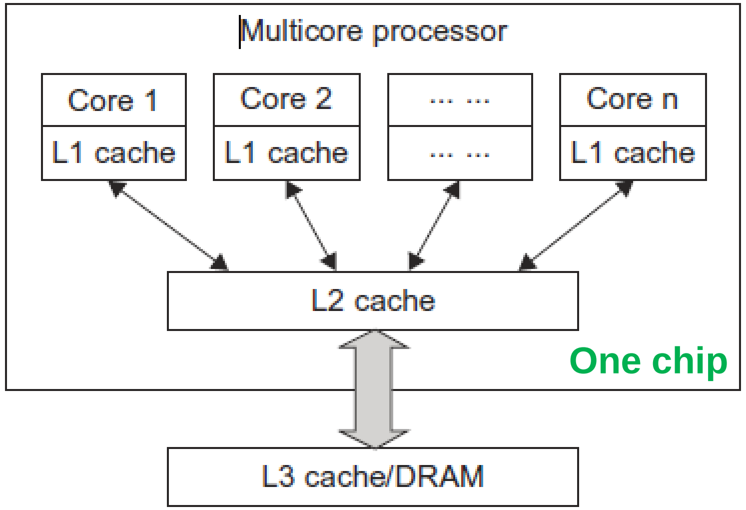
- Applications need to be adapted for HPC in order to take advantage of the shared or distributed memory parallelism.
  - Code parallelism is the key to success
  - Application must scale well over multiple nodes & cores

Processor Cores

As it became increasing difficult to pack more transistors into CPUs; manufacturers started using a **multi-core architecture** to speed up computing, with dual, quad, or n processing cores.

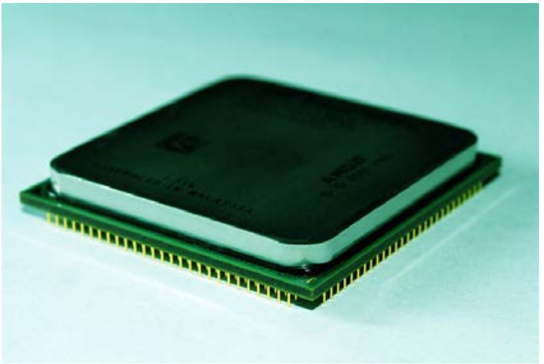
- Processing cores are all on chip

- Cache hierarchy: L1 core private, L2 shared on-chip, L3 off-chip Dynamic Random Access Memory (DRAM)



[13] ***Distributed & Cloud Computing Book***

[11] An Intel Core 2 Duo E6750 dual-core processor

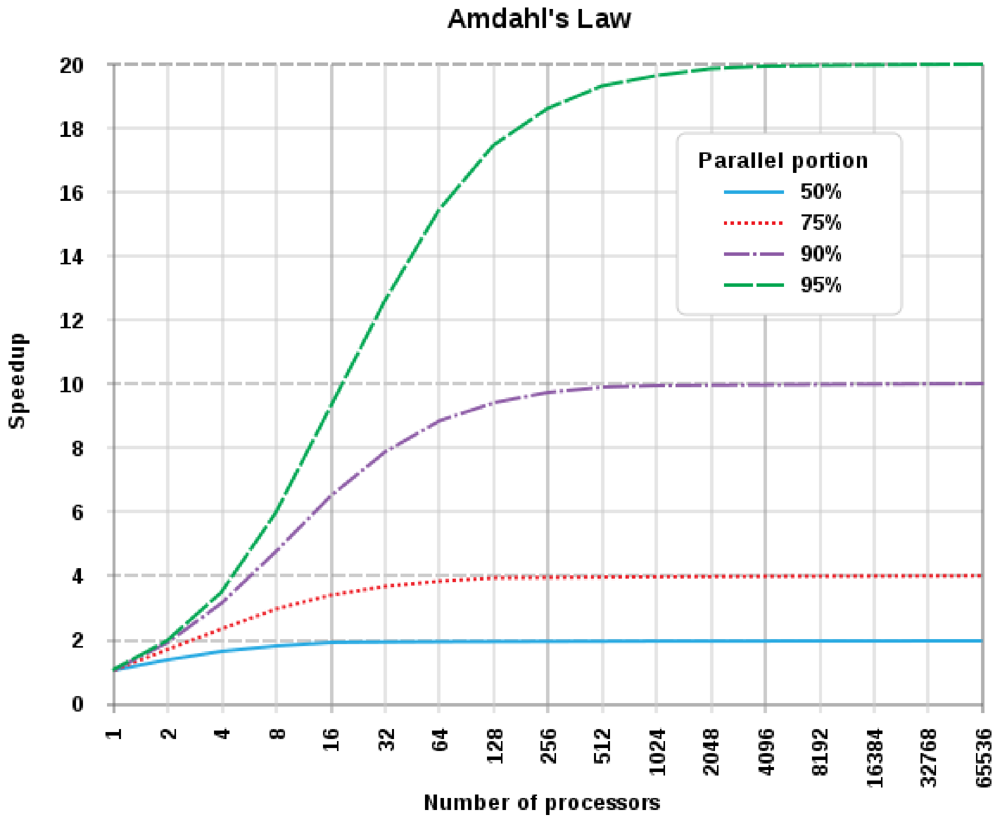


[12] An AMD Athlon X2 6400+ dual-core processor.

Parrallel Computing Theory

Two simple “laws” provide the theoretical foundation for speedup gained with code parallelism

- Amdahl‘s law, i.e. strong scaling, is the speedup with a *fixed problem size* and an increasing number of cores.
- Gustafson‘s law, i.e. weak scaling, is the speedup with a *fixed execution time* and an increasing number of cores.



[14] Amdahl’s law

Introducing PiSVM

- PiSVM is a parallel implemention of SVMs for HPC.
  - It‘s fork of PiSvM, which in turn is derived from LIBSVM, a popular open source library written in C/C++

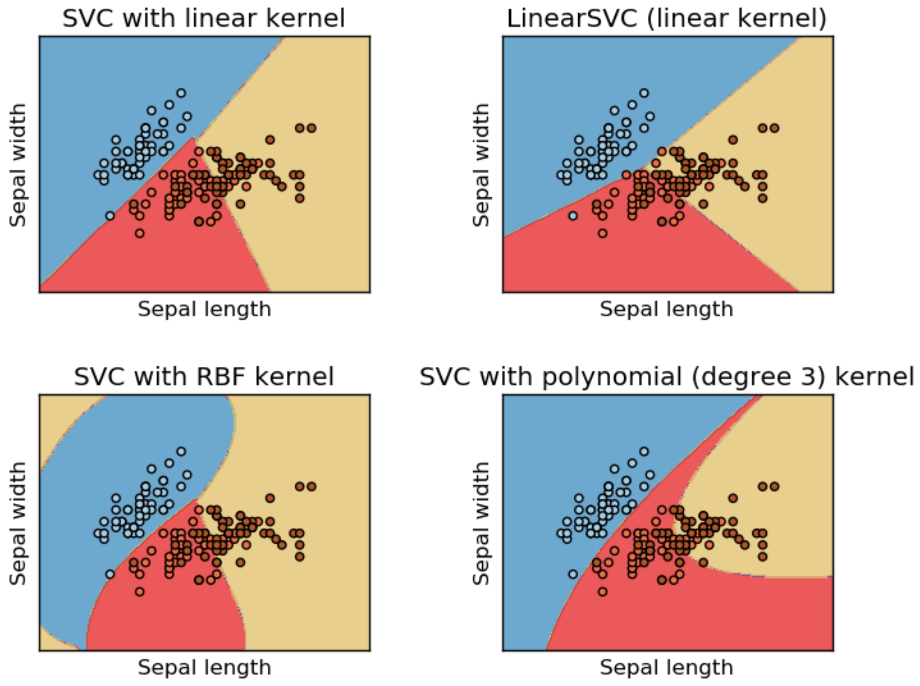
  PiSvM code repository
  http://pisvm.sourceforge.net/

  LIBSVM
  https://www.csie.ntu.edu.tw/~cjlin/libsvm/

PiSVM Details

PiSVM has two executables

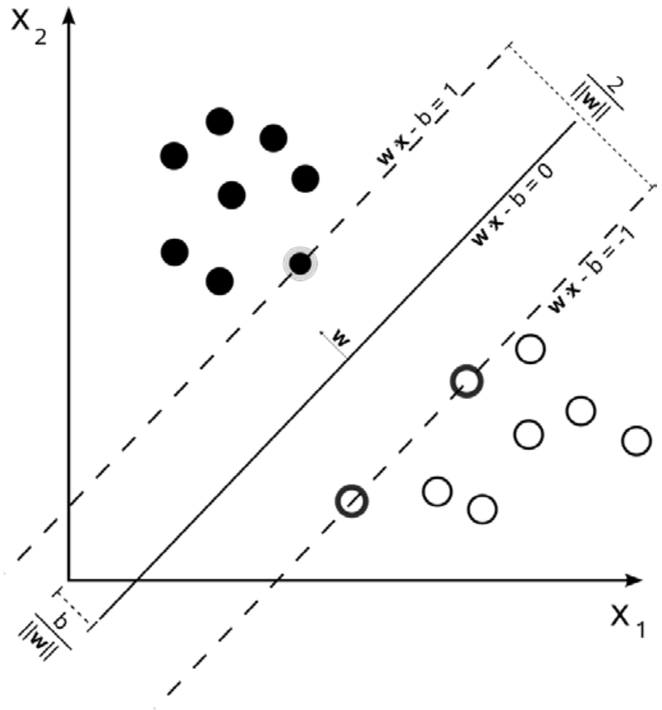
- **pisvm-train**, which is used for model training.
  - types supported: C-SVC, nu-SVC, one-class SVM, epsilon-SVR, nu-SVR
  - Kernels: linear, polynomial, RBF, sigmoid
- **pisvm-predict**, which is used for classifications on unseen data with a trained model.



[15] In depth parameter tuning for SVC

PiSVM Training

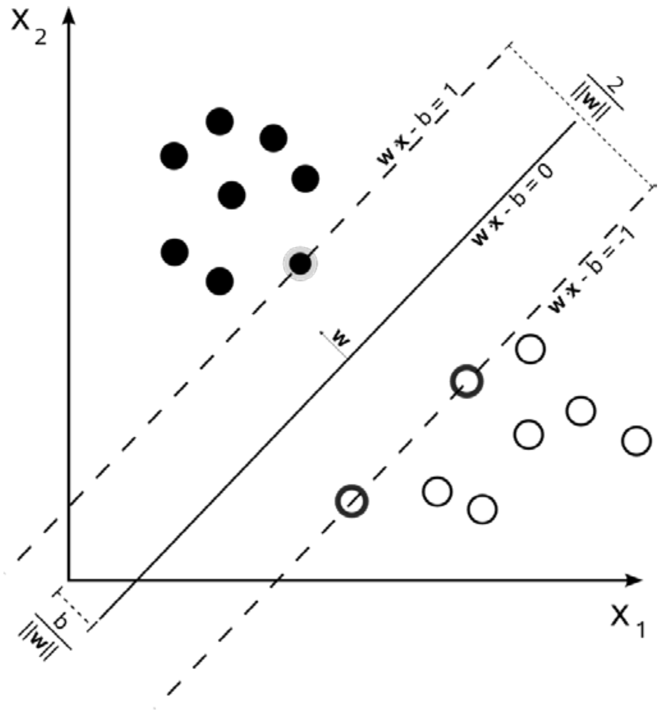
- Can parallelize vector calculations performed during max margin optimization.
  - Each core gets a part of the features and performs the respective vector computation
  - Parallel computations are merged back together and the best margin value is determined.
- Additionally, when performing grid-seach; it is possible to parallelize each cross-validation with a unique pair of C and γ.



[16] Support Vector Machines

PiSVM Classifications

- One of the disadvantages of using SVMs is that classification predictions must be calculated before they can be determined.
  - The time required to make classifications is dependent on the number of decision boundaries.
- Making SVM classification is an embarrasingly parallel operation, i.e. nicely parallel, which is optimal for accelerators.



[16] Support Vector Machines

Hardware Accelerators

- Hardware acceleration is the use of hardware to perform some functions more efficiently than is possible in software running on a more general-purpose CPU.

Common accelerators include:

- General Purpose Graphic Processing Units (GPGPUs)
  - NVIDIA has the lead with CUDA
- Field Programmable Gate Arrays (FPGAs)
- Specialized CPUs offering more parallelism
  - e.g. Intel Xeon Phi (R.I.P.)

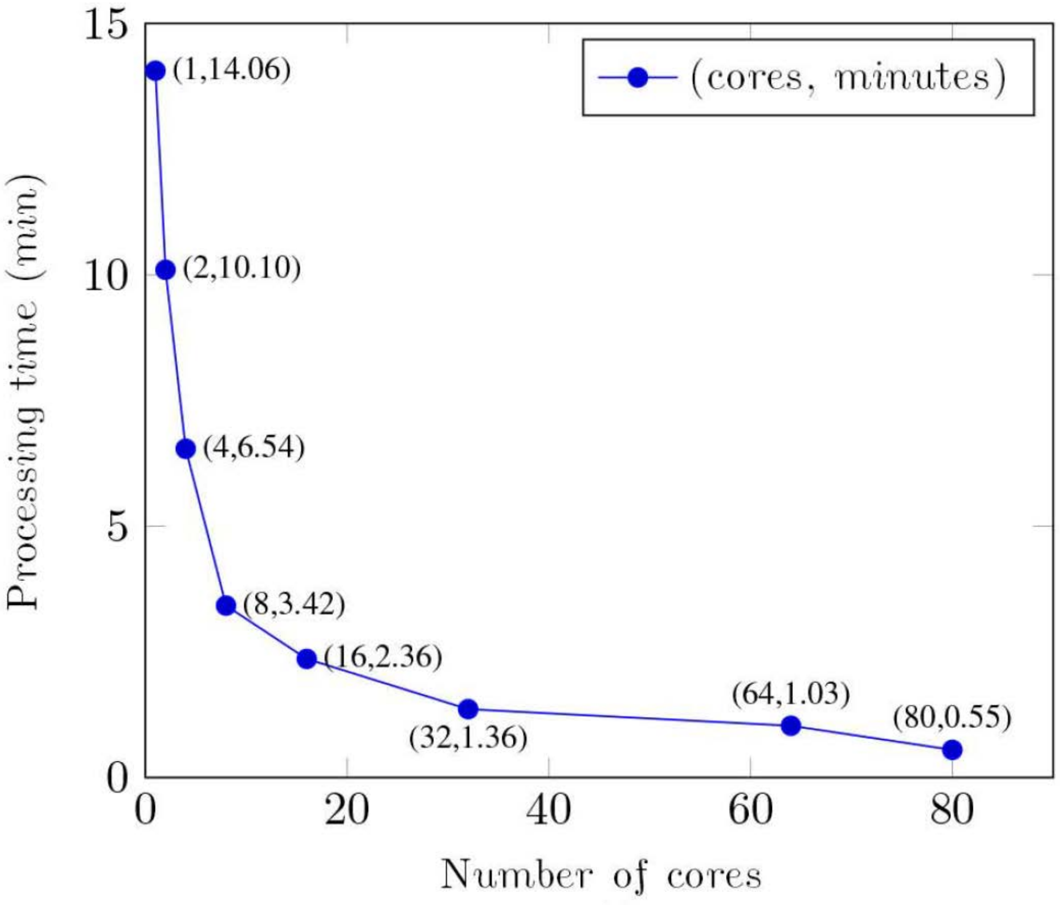
Note that hardware accelerators require at least one CPU, but not necessarily a good one.

Software Optimizations – Moving Target

- HPC software often to be optimized for new hardware
- Processing bottlenecks can change as processing vs memory vs interconnect speeds vary.

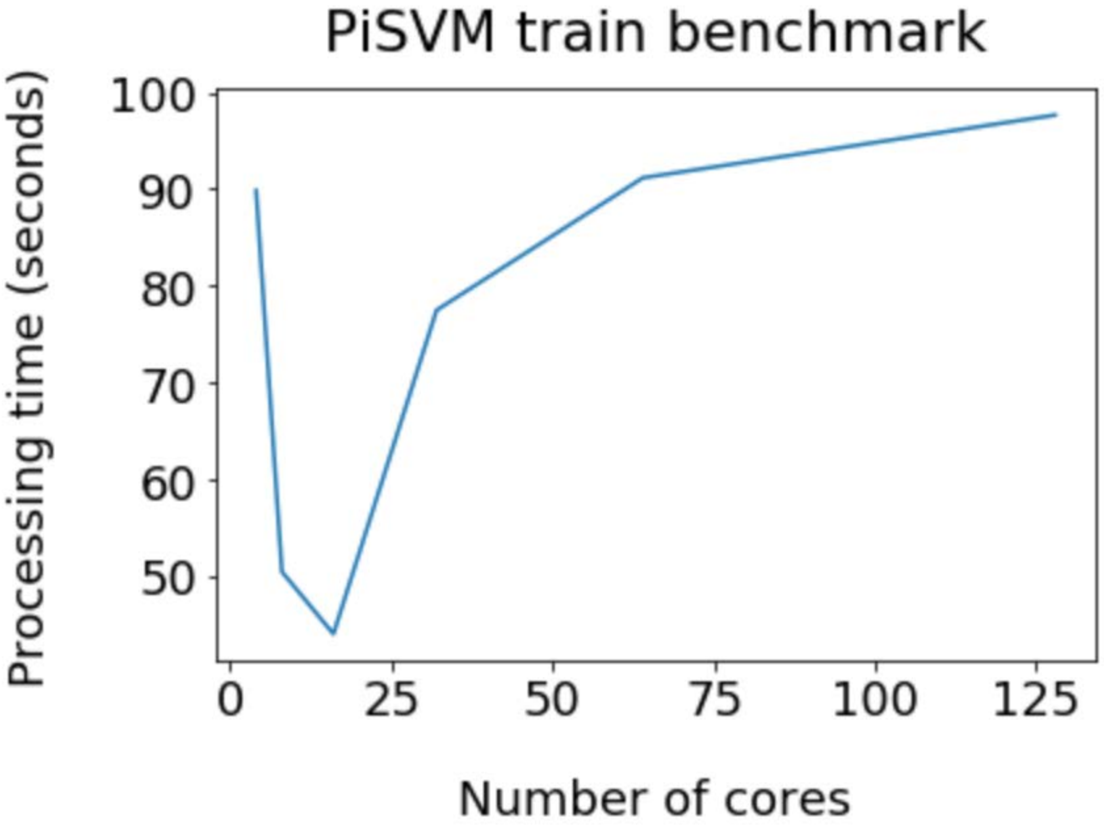
Images from https://www.pexels.com/

Software Optimizations – PiSVM



Intel® Xeon® X5650

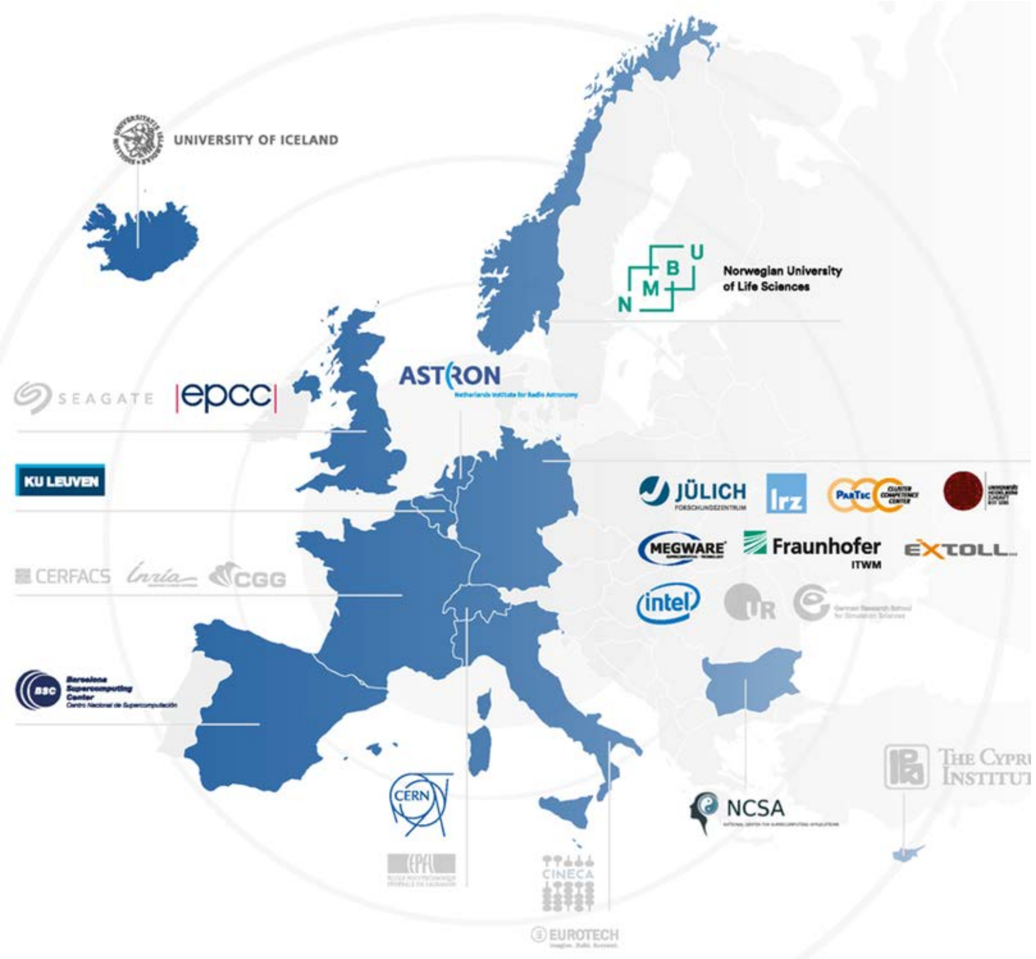
[7]



Intel® Xeon® E5-2680

The DEEP Projects

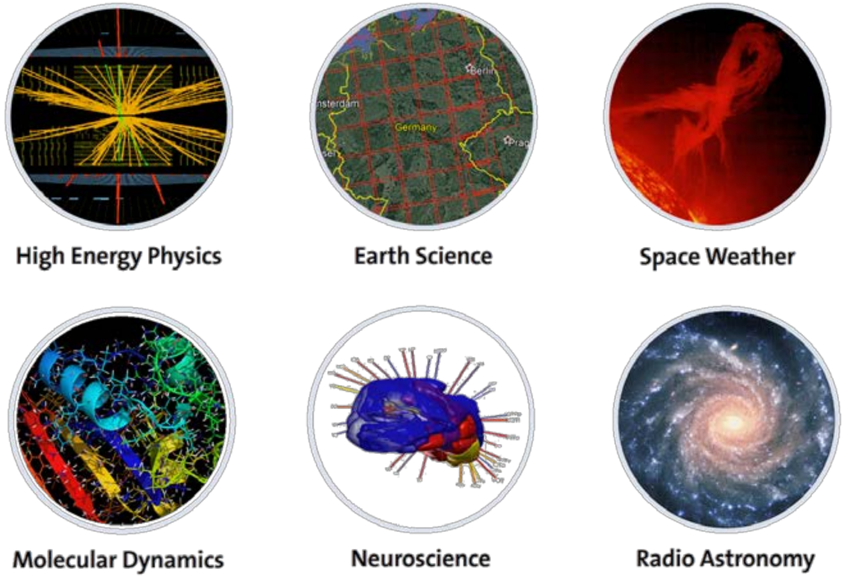
- Research & innovation projects co-funded by the European Union
  - DEEP (2012–2015): Cluster/Booster concept
  - DEEP-ER (2013–2017): scalable I/O and resiliency
  - DEEP-EST (2017–2020): generalized modular supercomputing architecture, support for data analytics and ML
- Co-design leading to novel, highly efficient, heterogeneous HPC architectures
- One of only two Exascale project series funded



[17] The DEEP projects

DEEP-EST Project Applications

- CERN – High Energy Physics
- NEST – Brain simulation
- University of Iceland – Earth Science
- ASTRON – Radio Astronomy
- KU Leuven – Space Weather
- GROMACS – Molecular Dynamics



Focus on co-design!

HPC – One Size does NOT fit All

HPC systems come in two very different flavours

- General purpose Clusters with
  - *High flexibility & reliable performance*
  - *Preferred by many applications since “good enough "performance is easy to achieve*
  - *Relatively high power consumption*
- Dedicated, highly scalable HPC systems (MPPs)
  - *Highest degree of parallelism, specialized fabrics*
  - *Few (highly parallelizable) codes can fully exploit them*
  - *Highly energy efficient*

The Deep projects combine the two flavours with the Modular Supercomputing Architecture (MSA)

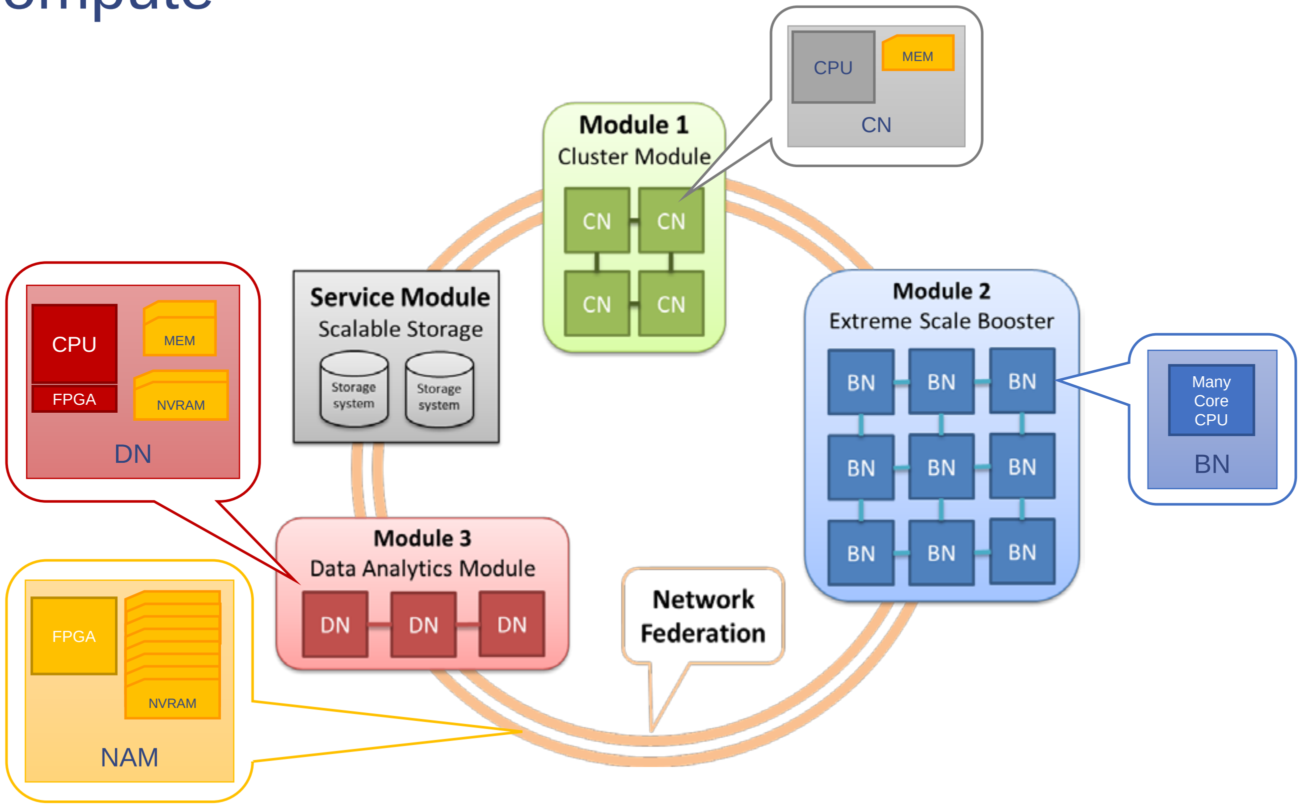
JSC JURECA Cluster
Intel® Xeon® Haswell
~ 2.2 PFlop/s in 1.3 MW

JUQUEEN
IBM Blue Gene/Q
5.9 PFlop/s in 2.3 MW

DEEP Prototype
Intel Xeon + Intel Xeon Phi
0.6 PFlop/s in 0.150 MW

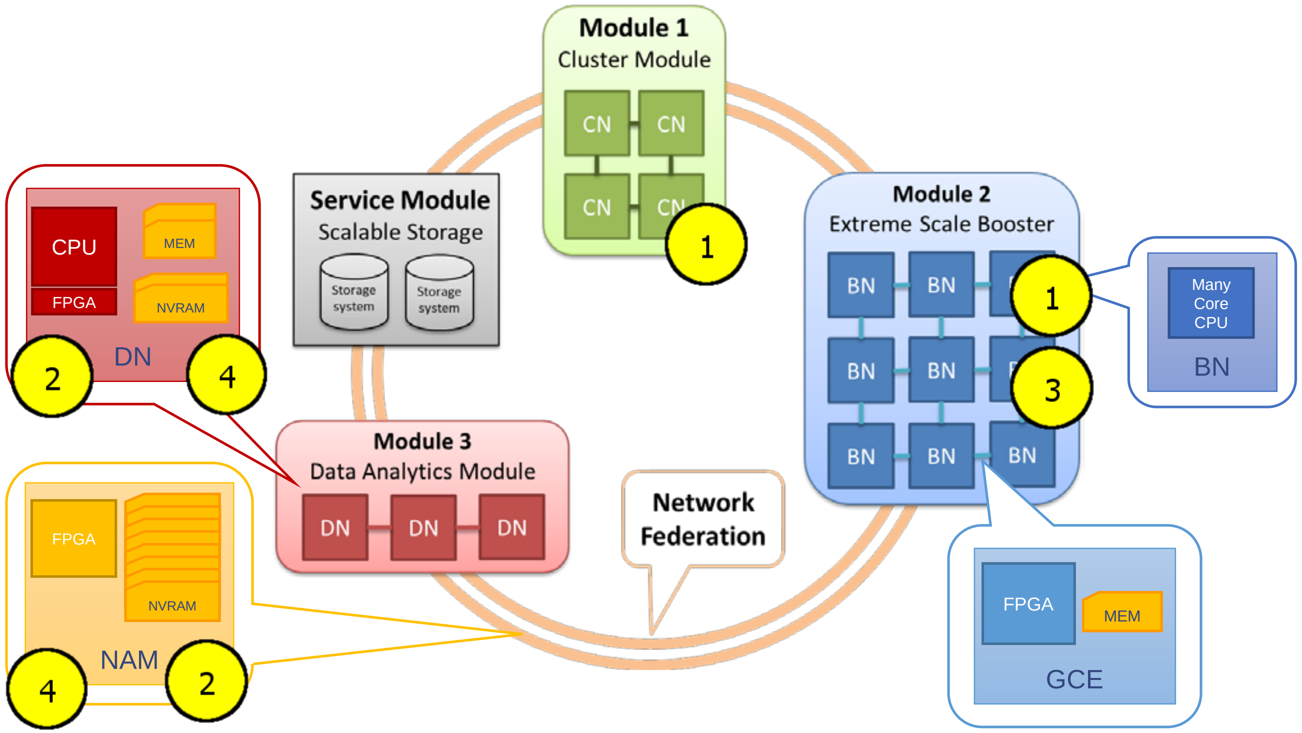
DEEP-EST MSA Architecture

- DEEP-EST prototype includes 3 compute modules
  - Cluster Module
  - Extreme Scale Booster
  - Data Analytics Module
- Storage module handles workflow data
- Local storage and the network attached memory (NAM) for hot application data
- Global communication engine (GCE) accelerates MPI collective operations
- Network federation binds all parts together



Application Analysis

PiSvM – Cross-validation



(1) Initial experiments performed with training and testing shows that a parameter space search is required in order to perform model selection (i.e. validation)

(2) Validation requires a validation dataset or again the training dataset when using cross-validation (low bias) but instead of reading the training data from a file again and again it can be placed in the DEEP-EST NAM **or the DAM´s NVRAM**

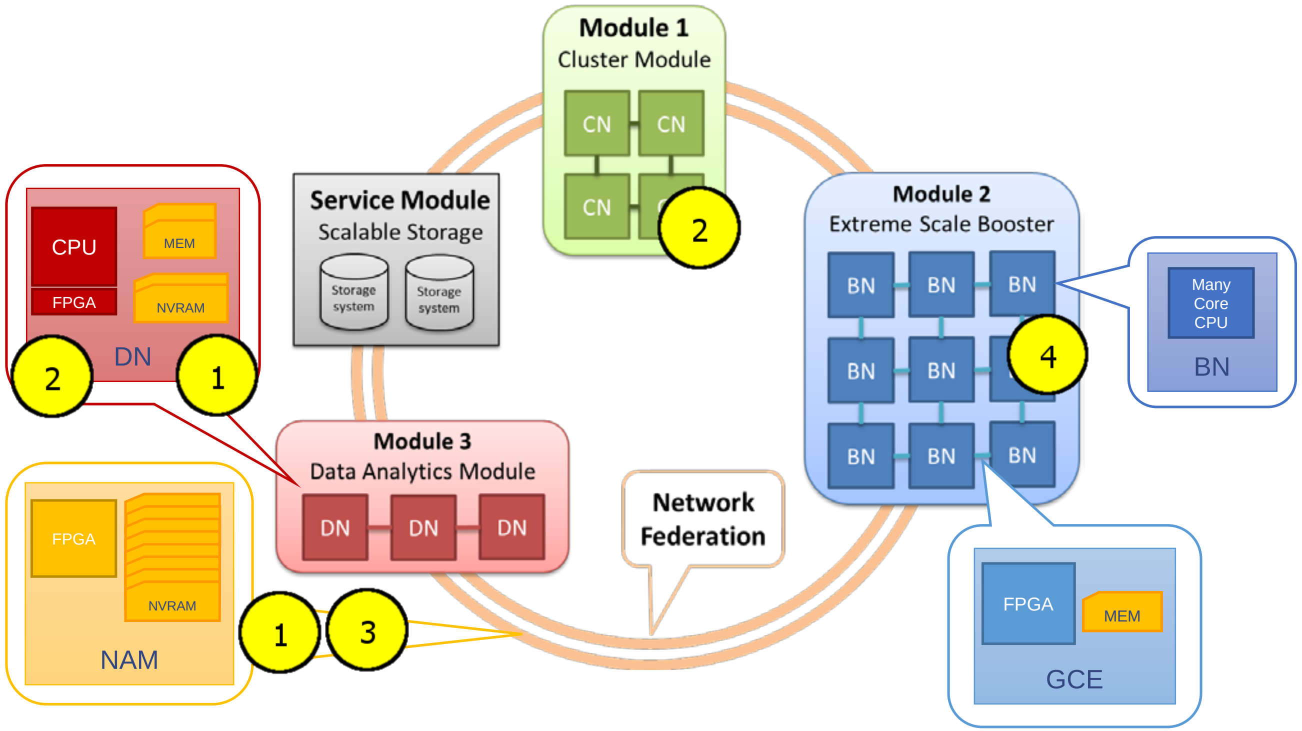
(3) n-fold cross-validation over a grid of parameters (kernel, cost) performs an estimate of the out-of-sample performance and performs n-times independent training process on a “folded” subset of the dataset (use of training data in folds). n-fold Cross-Validation (e.g. 10-fold often used) with piSVM is partly computational intensive whereby each fold can be nicely parallelized without requiring a good interconnection and thus can take advantage of the DEEP-EST CLUSTER module (use of training data in folds) whereby results of each fold per parameter can be put in the DEEP-EST NAM module **or the DAM´s NVRAM**

(4) The best parameters w.r.t. MAXIMUM accuracy in all the folds across all the parameter spaces can be computed using the DEEP-EST NAM **or DAM** module (FPGA computing maximum)

(5) The best parameter set that resides in the DEEP-EST NAM is given as input to the training/test pipeline (see PiSvM Tranining)

Application Analysis

PiSvM – Training



(1) The training dataset and testing dataset of the remote sensing application is used many times in the process and make sense to put into the DEEP-EST Network Attached Memory (NAM) module **or the NVRAM in the DAM.**

(2) Training with piSVM in order to generate a model requires powerful CPUs with good interconnection for the inherent optimization process and thus can take advantage of the DEEP-EST CLUSTER module (use of training dataset, requires piSVM parameters for kernel and cost). **Optionally, the DAM can be used in the case when its NVRAM is employed**.

(3) Instead of dropping the trained SVM model (i.e. file with support vectors) to disk it makes sense to put this model into the DEEP-EST NAM module

(4) Testing with piSVM in order to evaluate the model accuracy requires not powerful CPUs and not a good interconnection but scales perfectly (i.e. nicely parallel) and thus can take advantage of the BOOSTER module (use of testing dataset & model file residing in NAM)

(5) If accuracy too low back to (2) to change parameters

Further Information

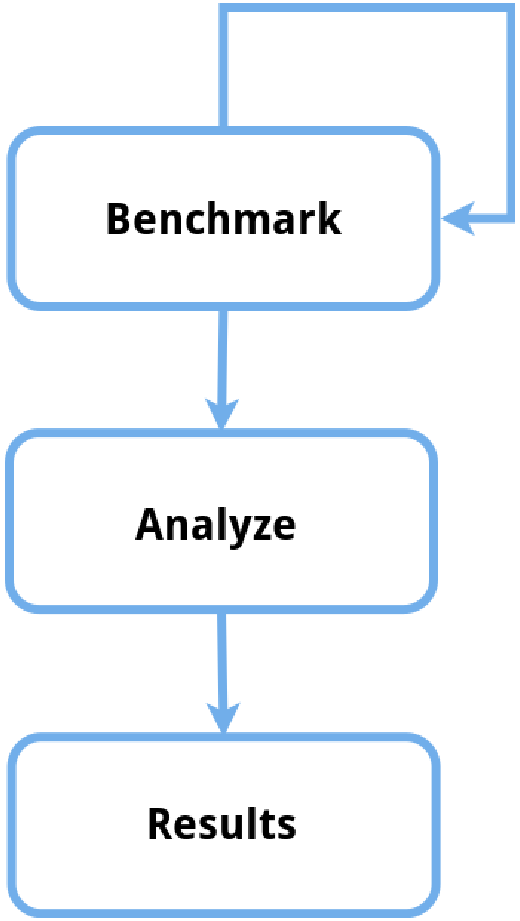
Interested in learning more about PiSVM on the DEEP system ?

Come and check out my talk on Tuesday!

Paper Code: TU2.R8.5 **Paper Number:** 4143
Title: SCALING SUPPORT VECTOR MACHINES TOWARDS EXASCALE COMPUTING FOR CLASSIFICATION OF LARGE-SCALE HIGH-RESOLUTION REMOTE SENSING IMAGES
Topic: Special Themes: Big machine learning in remote sensing
Session: TU2.R8: Big Machine Learning IV
Time: Tuesday, July 24, 11:10 - 12:50
Authors: Ernir Erlingsson, Gabriele Cavallaro, Morris Riedel, Helmut Neukirchen

Running Multiple Tasks - JUBE

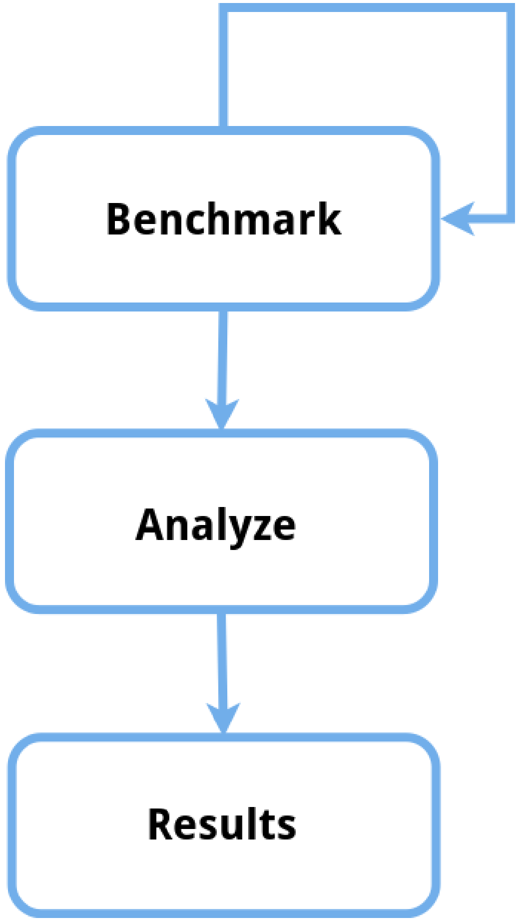
- HPC users rarely run single tasks, but rather batches of multiple tasks; and possibly using different computational resources. Examples include PiSVM grid search and the training evaluation phases of PiSVM.
- The naïve approach is to launch every single job manually, but better is to use tools such as the **JUBE** benchmarking environment developed by the Juelich Supercomputing Centre in Germany.
- **JUBE** can configure, compile and run a benchmark suite on several platforms, including an additional results collection and analysis.



[18] JUBE

Running Multiple Tasks - JUBE

- **JUBE** scripts are written with XML, using a set of custom tags and specifying serial execution steps, which can be *synchronous* or *asynchronous*.
- Regular expressions can be used to collect data from any output, and present it in the console or export as a csv or xml.
- Conditions are relative to the supercomputer used, which partitions, etc.; i.e. the conditions are defined by the admins.



[18] JUBE

Running Multiple Tasks - JUBE

```
<?xml version="1.0" encoding="UTF-8"?>
<jube>
    <benchmark name="pisvm_train" outpath="./pisvm_sdv_train">
        <comment>
            A JUBE script that compiles and trains models using PiSVM
        </comment>

        <parameterset name="pisvm_train_executeset">
            <parameter name="partition">sdv</parameter>
            <parameter name="nodes">${used_nodes}</parameter>
            <parameter name="tasks">${used_tasks}</parameter>
            <parameter name="walltime">03:00:00</parameter>
            <parameter name="pisvm">~/git/UoI-piSVM/source/pisvm-train</parameter>
            <parameter name="train_data">${input_dir}${train}</parameter>
            <parameter name="process">srun $pisvm -D -o 1024 -q 512 -c 8 -g 10 -t 2 -m 1024 -s 0 $train_data</parameter>
        </parameterset>

        <patternset name="pattern">
            <pattern name="total_time_pat">^total time [0-9]*\.[0-9]*</pattern>
        </patternset>

        <step name="modules">
            <do>
                module purge;
                module load $modules
            </do>
        </step>
        <step name="compile" depend="modules">
            <do>
                make -C ~/git/UoI-piSVM/source/
            </do>
        </step>
        <step name="train" depend="compile">
```

Practicals

Example - Indian Pines Dataset

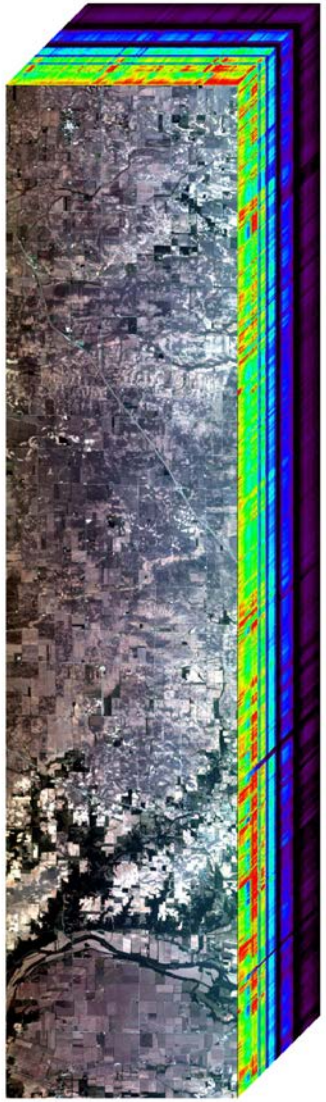
- Nasa – Jet Propulsion Laboratory (JPL)
- Airbone Visible/Infrared Imaging Spectrometer (AVIRIS)
- 224 contiguous spectral channels
- Spectral range of 0.4μm to 2.5μm (10 $\mathrm{n}m$ spectral resolution)
- Spatial resolution of 20m

[5] AVIRIS

Indian Pines Dataset

- Agricultural fields with a variety of crops
- Challenging classification problem
- Similar spectral classes and mixed pixels

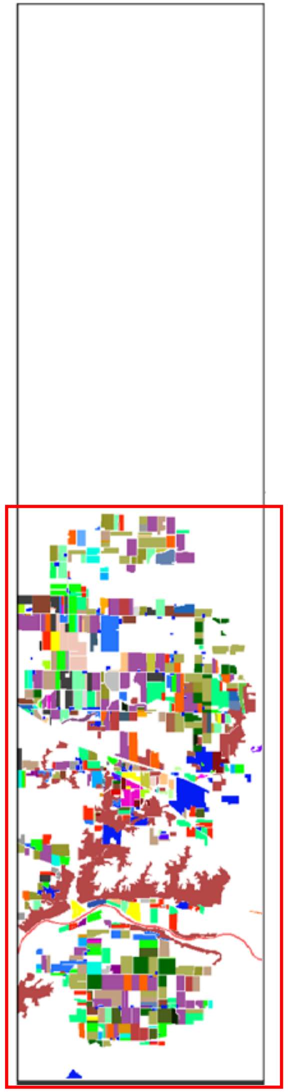
[6] Indian Pines dataset



Preprocessing and Groundtruth

- Crop of 1417 x 617 pixels (~600MB)
- 200 bands (20 discarded, with low SNR)
- 58 classes (6 discarded, with < 100 samples)

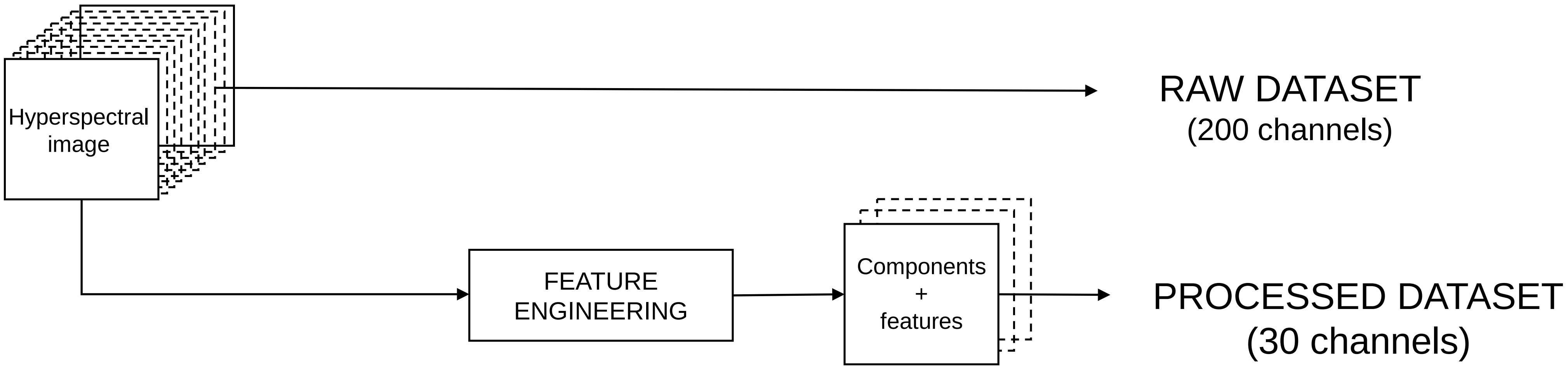
X BareSoil (57)	Corn-MinTill (1049)	Hay (1128)	Soybeans-NS (1110)	Soybeans-NaTill (2157)
Buildings (17195)	Corn-MinTill-EW (5629)	Hay (pre.) (2185)	Soybeans-CleanTill (5074)	Soybeans-NaTill-EW (2533)
X Concrete/Asphalt (69)	Corn-MinTill-NS (8862)	Hay-Alfalfa (2258)	Soybeans-CleanTill (pre.) (2726)	Soybeans-NaTill-NS (929)
Corn (17783)	Corn-Natill (4381)	Lake (224)	Soybeans-CleanTill-EW (11802)	Soybeans-NaTill-Drilled (8731)
Corn (presumed) (158)	Corn-Natill-EW (1206)	NotCropped (1940)	Soybeans-CleanTill-NS (10387)	Swampy Area (583)
Corn-EW (514)	Corn-Natill-NS (5685)	Oats (1742)	Soybeans-CleanTill-Drilled (2242)	River (3110)
Corn-NS (2356)	Fescue (114)	Oats (pre.) (335)	Soybeans-CleanTill-Weedy (543)	Trees (pre.) (580)
Corn-CleanTill (12404)	Grass (1147)	X Orchard (39)	Soybeans-MinTill (15118)	Wheat (4979)
Corn-CleanTill-EW (26486)	Grass/Trees (2331)	Pasture (10386)	Soybeans-Drilled (2667)	Woods (63562)
Corn-CleanTill-NS (39678)	X Grass/Pasture-mowed (19)	Pond (102)	Soybeans-MinTill (1832)	Unknown (144)
Corn-CleanTill-NS-Irrigated (800)	X Grass/Pasture (73)	Soybeans (9391)	Soybeans-MinTill-Drilled (8098)	
Corn-CleanTill-NS (pre.) (1728)	X Grass-runway (37)	Soybeans (pre.) (894)	Soybeans-MinTill-NS (4953)	



[7] G. Cavallaro et al.

Experimental Setup

- Feature Engineering
  - Kernel Principle Component Analysis (KPCA)
  - Extended Self-Dual Attribute Profile (ESDAP) *[7] G. Cavallaro et al.*
  - Nonparametric weighted feature extraction (NWFE)



Publicly Available Datasets – Location

- *Indian Pines Dataset Raw and Processed*

B2SHARE
Store and Share Research Data

[8] Indian Pines 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: 11304/7e8eec8e-ad61-11e4-ac7e-860aa0063d1f Copy

EUDAT

Files

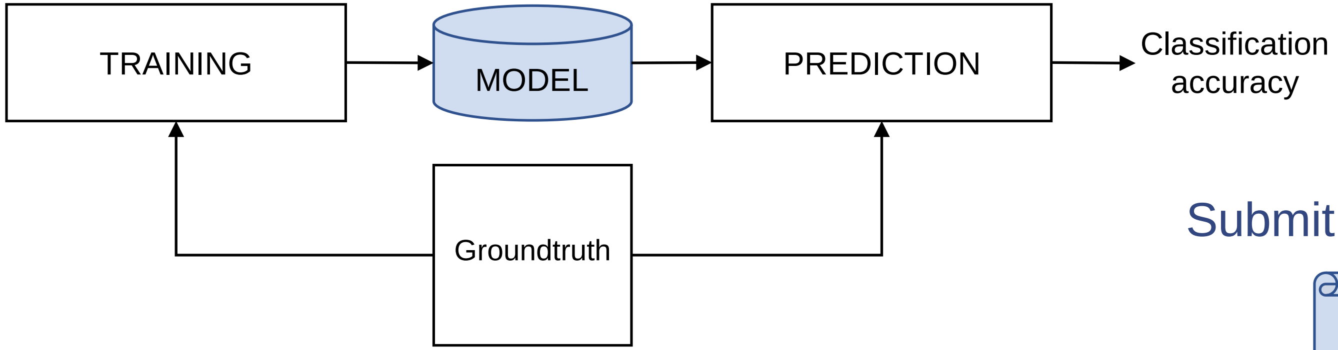
Name	Size
indian_processed_test.el	105.59MB
indian_processed_training.el	11.73MB
indian_raw_test.el	747.13MB
indian_raw_training.el	83.01MB

Basic metadata

Open Access	True ✔
License	
Contact Email	
Publication Date	2015-02-04
Contributors	
Resource Type	**Category** Other
Alternate identifiers	172 **Type** B2SHARE_V1_ID http://hdl.handle.net/11304/9ec5eac8-61b4-4617-ae1c-1f8c8cd3cd74 **Type** ePIC_PID
Publisher	https://b2share.eudat.eu
Language	en

```
[train001@jrl07 indianpines]$ pwd
/homea/hpclab/train001/data/indianpines
[train001@jrl07 indianpines]$ ls -al
total 925344
drwxr-xr-x 2 train001 hpclab       512 Jul  7  2016 .
drwxr-xr-x 5 train001 hpclab       512 Jan 14 13:10 ..
-rw-r--r-- 1 train001 hpclab        36 Jul  7  2016 b2share.txt
-rw-r--r-- 1 train001 hpclab 105594346 Feb  5  2015 indian_processed_test.el
-rw-r--r-- 1 train001 hpclab  11732509 Feb  5  2015 indian_processed_training.el
-rw-r--r-- 1 train001 hpclab 747125597 Feb  5  2015 indian_raw_test.el
-rw-r--r-- 1 train001 hpclab  83014311 Feb  5  2015 indian_raw_training.el
```

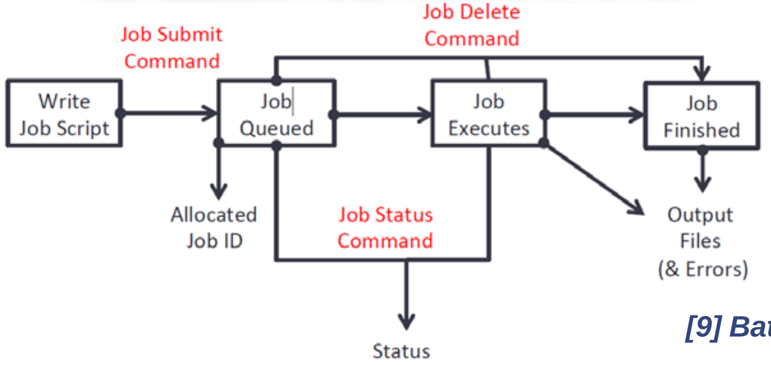
Classification Pipeline with piSVM



Submit the training

submit-train-indian-pines-raw.sh

RAW DATASET
(200 channels)



Submit the prediction

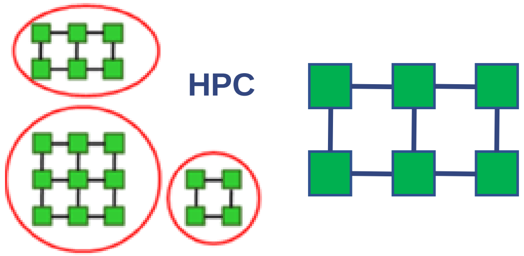
submit-pred-indian-pines-raw.sh

[9] Batch Systems

JURECA HPC System at JSC

- Characteristics
  - Login nodes with 256 GB memory per node
  - 45,216 CPU cores
  - 1.8 (CPU) + 0.44 (GPU) Petaflop/s peak performance
  - **Two Intel Xeon E5-2680 v3 Haswell CPUs per node: 2 x 12 cores, 2.5 GhZ**
  - 75 compute nodes equipped with two NVIDIA K80 GPUs (2 x 4992 CUDA cores)
- Architecture & Network
  - Based on T-Platforms V-class server architecture
  - Mellanox EDR InfiniBand high-speed network with non-blocking fat tree topology
  - 100 GiB per second storage connection to JUST

[10] JURECA HPC System



Location of the Job Scripts

```
[train053@jrl06 ~]$ ls
bin  igarss_tutorial
[train053@jrl06 ~]$ cd igarss_tutorial/
[train053@jrl06 igarss_tutorial]$ ls
3dcnn  mpi_hello_world  pisvm
[train053@jrl06 igarss_tutorial]$ cd pisvm/
[train053@jrl06 pisvm]$ ls
submit-pred-indian-pines-processed.sh  submit-pred-indian-pines-raw.sh  submit-train-indian-pines-processed.sh  submit-train-indian-pines-raw.sh
```

Or, from whenever you are:

- **$ cd ~/igarss_tutorial/pisvm/**
- **$ ls**

Job Scrip for Training (Raw Dataset)

```
#!/bin/bash -x

#SBATCH--job-name=train-indianpines-1-24
#SBATCH--output=train-indianpines-1-24-out.%j
#SBATCH--error=train-indianpines-1-24-err.%j
#SBATCH--mail-user=your_email
#SBATCH--mail-type=ALL

#SBATCH--partition=batch
#SBATCH--nodes=1
#SBATCH--ntasks=24
#SBATCH--time=01:00:00
#SBATCH--reservation=igarss-cpu

### load modules
module load intel-para

### location executable
PISVM=/homea/hpclab/train001/tools/pisvm-1.2.1/pisvm-train

### location data
TRAINDATA=/homea/hpclab/train001/data/indianpines/indian_raw_training.el

### submit
srun $PISVM -D -o 1024 -q 512 -c 100 -g 8 -t 2 -m 1024 -s 0 $TRAINDATA
```

$ sbatch submit-train-indian-pines-raw.sh

```
[train001@jrl06 pisvm]$ sbatch submit-train-indian-pines-raw.sh
Submitted batch job 5812912
```

(ReservationName=igarss-cpu StartTime=2018-07-22T10:45:00 EndTime=2018-07-22T18:15:00)

Output Training Model

- indian_raw_training.el.model

```
[train001@jrl06 pisvm]$ ls
indian_raw_training.el.model              submit-train-indian-pines-raw.sh
submit-pred-indian-pines-processed.sh     train-indianpines-1-24-err.5812912
submit-pred-indian-pines-raw.sh           train-indianpines-1-24-out.5812912
submit-train-indian-pines-processed.sh
```

- $ vi indian_raw_training.el.model

```
svm_type c_svc
kernel_type rbf
gamma 8
nr_class 52
total_sv 32911
```

Job Scrip for Predicting (Raw Dataset)

```
#!/bin/bash -x

#SBATCH--job-name=train-indianpines-1-24
#SBATCH--output=train-indianpines-1-24-out.%j
#SBATCH--error=train-indianpines-1-24-err.%j
#SBATCH--mail-user=your_email
#SBATCH--mail-type=ALL

#SBATCH--partition=batch
#SBATCH--nodes=1
#SBATCH--ntasks=24
#SBATCH--time=01:00:00
#SBATCH--reservation=igarss-cpu

### load modules
module load intel-para

### location executable
PISVMPRED=/homea/hpclab/train001/tools/pisvm-1.2.1/pisvm-predict

### location data
TESTDATA=/homea/hpclab/train001/data/indianpines/indian_raw_test.el

### trained model data
MODELDATA=/homea/hpclab/train001/tools/pisvm-1.2.1/indian_raw_training.el.model

### submit
srun $PISVMPRED $TESTDATA $MODELDATA results.txt
```

$ sbatch submit-predict-indian-pines-raw.sh

```
[train001@jrl06 pisvm]$ sbatch submit-pred-indian-pines-raw.sh
Submitted batch job 5813193
```

(ReservationName=igarss-cpu StartTime=2018-07-22T10:45:00 EndTime=2018-07-22T18:15:00)

Prediction Outputs

```
[train001@jrl06 pisvm]$ ls
indian_raw_training.el.model              submit-train-indian-pines-raw.sh
results.txt                               train-indianpines-1-24-err.5812912
submit-pred-indian-pines-processed.sh     train-indianpines-1-24-err.5813193
submit-pred-indian-pines-raw.sh           train-indianpines-1-24-out.5812912
submit-train-indian-pines-processed.sh    train-indianpines-1-24-out.5813193
```

$ vi results.txt

```
27
48
48
51
36
51
48
27
51
50
51
8
51
51
51
51
51
51
51
51
```

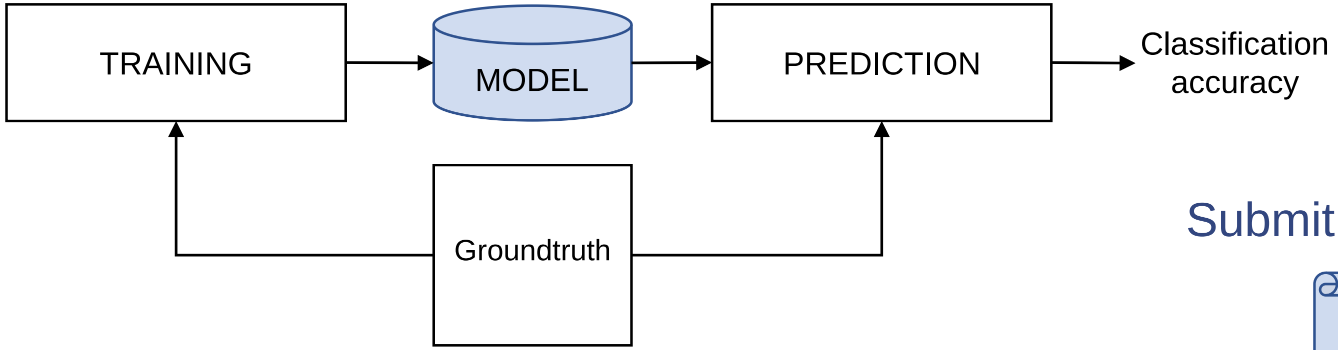
$ vi train-indianpines-1-24-out-5813193

```
Accuracy = 40.0828% (120471/300555) (classification)
Mean squared error = 453.392 (regression)
Squared correlation coefficient = 0.137346 (regression)
```

Exercise 1

- Replicate the previous steps for the **processed dataset**
- Use the job script **submit-train-indian-pines-processed.sh** for training
- Use the job script **submit-pred-indian-pines-processed.sh** for predicting
- Check the classification results

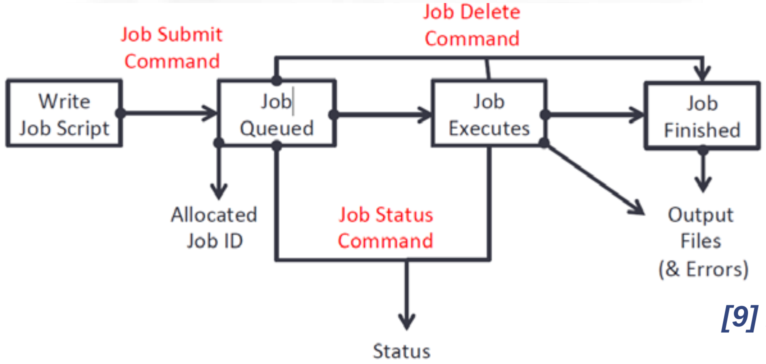
Classification Pipeline with piSVM



Submit the training

submit-train-indian-pines-processed.sh

PROCESSED DATASET
(30 channels)



[9] Batch Systems

Submit the prediction

submit-pred-indian-pines-processed.sh

Exercise 2 – Cross Validation – Processed Dataset

```
#!/bin/bash -x

#SBATCH--job-name=train-indianpines-1-24
#SBATCH--output=train-indianpines-1-24-out.%j
#SBATCH--error=train-indianpines-1-24-err.%j
#SBATCH--mail-user=g.cavallaro@fz-juelich.de
#SBATCH--mail-type=ALL

#SBATCH--partition=batch
#SBATCH--nodes=1
#SBATCH--ntasks=24
#SBATCH--time=01:00:00
#SBATCH--reservation=igarss-cpu

### load modules
module load intel-para

### location executable
PISVM=/homea/hpclab/train001/tools/pisvm-1.2.1/pisvm-train

### location data
TRAINDATA=/homea/hpclab/train001/data/indianpines/indian_raw_training.el

### submit
srun $PISVM -D -o 1024 -q 512 –v 10 -c ??? -g ??? -t 2 -m 1024 -s 0 $TRAINDATA
```

sbatch submit-train-indian-pines-raw.sh

Exercise 2 – Cross Validation – Processed Dataset

γ \ C	1	10	100	1000	10000
1					
2					
4					
8					
16					
32					

Bibliography

- [1] Introduction to Data Mining, Pang-Ning Tan, Michael Steinbach, Vipin Kumar, Addison Wesley, ISBN 0321321367, English, ~769 pages, 2005.
- [2] Earth Observation Mission Sentinel 2, Online: https://sentinel.esa.int/web/sentinel/missions/sentinel-2
- [3] An Introduction to Statistical Learning with Applications in R

  Online: http://www-bcf.usc.edu/~gareth/ISL/index.html
- [4] Support Vector Machines and Kernel Algorithms

  Online https://pdfs.semanticscholar.org/2862/e7b8fefb209cdb4c47a1643f2af71cd67b00.pdf
- [5] Airborne Visible/Infrared Imaging Spectrometer (AVIRIS)

  Online: https://www.jpl.nasa.gov/missions/airborne-visible-infrared-imaging-spectrometer-aviris/
- [6] Indian Pines dataset: 220 Band AVIRIS Hyperspectral Image

  Online: https://purr.purdue.edu/publications/1947/1
- [7] G. Cavallaro, M. Riedel, M. Richerzhagen, J. A. Benediktsson and A. Plaza, "On Understanding Big Data Impacts in Remotely Sensed Image Classification Using Support Vector Machine Methods," in the IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, vol. 8, no. 10, pp. 4634-4646, Oct. 2015.
- [8] Indian Pines Raw and Processed

  Online: http://hdl.handle.net/11304/9ec5eac8-61b4-4617-ae1c-1f8c8cd3cd74
- [9] Batch Systems – archer Running your jobs on an HPC machine

  Online: https://www.archer.ac.uk/training/course-material/2017/07/intro-epcc/slides/L10_Batch_Execution.pdf
- [10] JURECA HPC System at JSC

  Online: http://www.fz-juelich.de/ias/jsc/EN/Expertise/Supercomputers/JURECA/JURECA_node.html

Bibliography

- [11] An Intel Core 2 Duo E6750 dual-core processor.
  Online: https://en.wikipedia.org/wiki/Multi-core_processor
- [12] An AMD Athlon X2 6400+ dual-core processor.
  Online: https://en.wikipedia.org/wiki/Multi-core_processor
- [13] Distributed & Cloud Computing Book] K. Hwang, G. C. Fox, J. J. Dongarra, 'Distributed and Cloud Computing', Book,
  Online: http://store.elsevier.com/product.jsp?locale=en_EU&isbn=9780128002049
- [14] Amdahl's law
  Online: https://en.wikipedia.org/wiki/Amdahl%27s_law
- [15] In depth parameter tuning for SVC
  Online: https://medium.com/@mohtedibf/in-depth-parameter-tuning-for-svc-758215394769
- [16] Support Vector Machines
  Online: https://en.wikipedia.org/wiki/Support_vector_machine
- [17] Deep Projects
  http://www.deep-projects.eu/
- [18] JUBE
  Online: http://www.fz-juelich.de/jsc/jube

DEEP Projects

The DEEP projects have received funding from the European Union's Seventh Framework Programme (FP7) for research, technological development and demonstration and the Horion2020 (H2020) funding framework under grant agreement no. FP7-ICT-287530 (DEEP), FP7-ICT-610476 (DEEP-ER) and H2020-FETHPC-754304 (DEEP-EST).