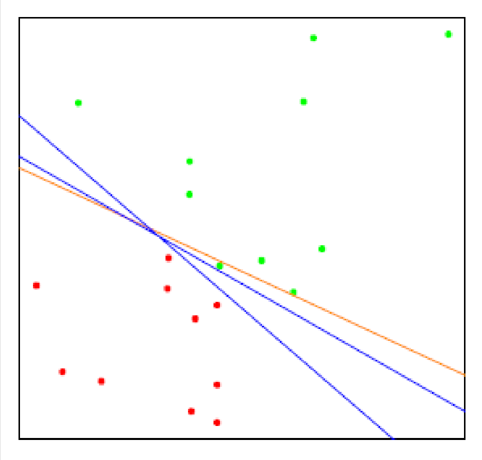


Parallel & Scalable Machine Learning

Introduction to HPC-driven Machine Learning Models

Prof. Dr. – Ing. Morris Riedel
Adjunct Associated Professor
School of Engineering and Natural Sciences, University of Iceland
Research Group Leader, Juelich Supercomputing Centre, Germany

INVITED LECTURE

Tutorial on Machine Learning & Data Analytics

June 27th, 2018
International Supercomputing Conference (ISC), STEM Day, Frankfurt, Germany

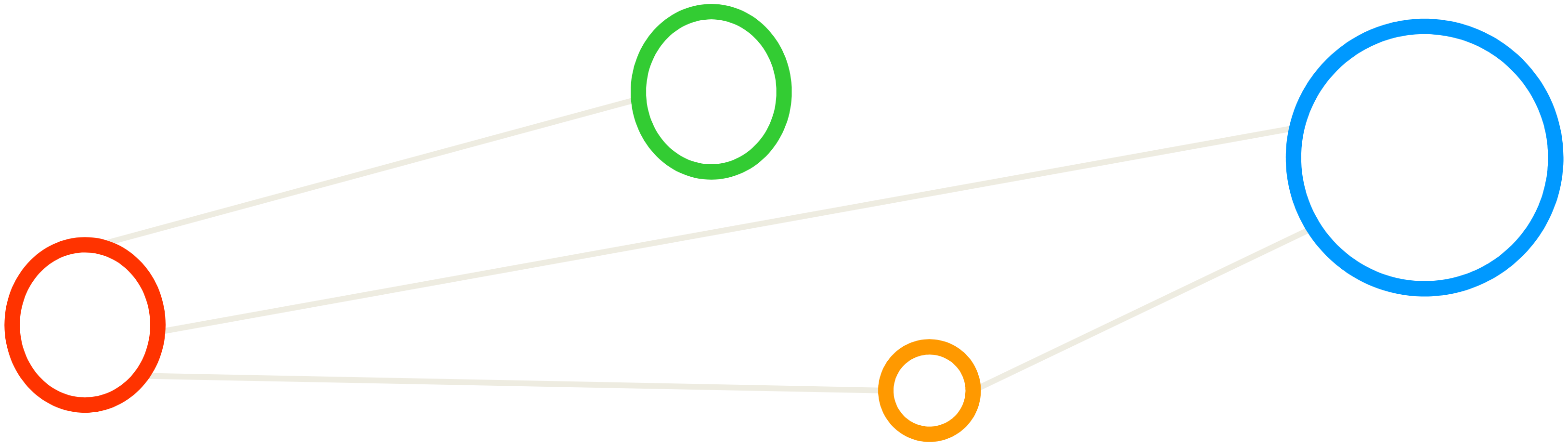
UNIVERSITY OF ICELAND
SCHOOL OF ENGINEERING AND NATURAL SCIENCES
FACULTY OF INDUSTRIAL ENGINEERING, MECHANICAL ENGINEERING AND COMPUTER SCIENCE

JÜLICH Forschungszentrum

HELMHOLTZ RESEARCH FOR GRAND CHALLENGES

DEEP Projects

Outline

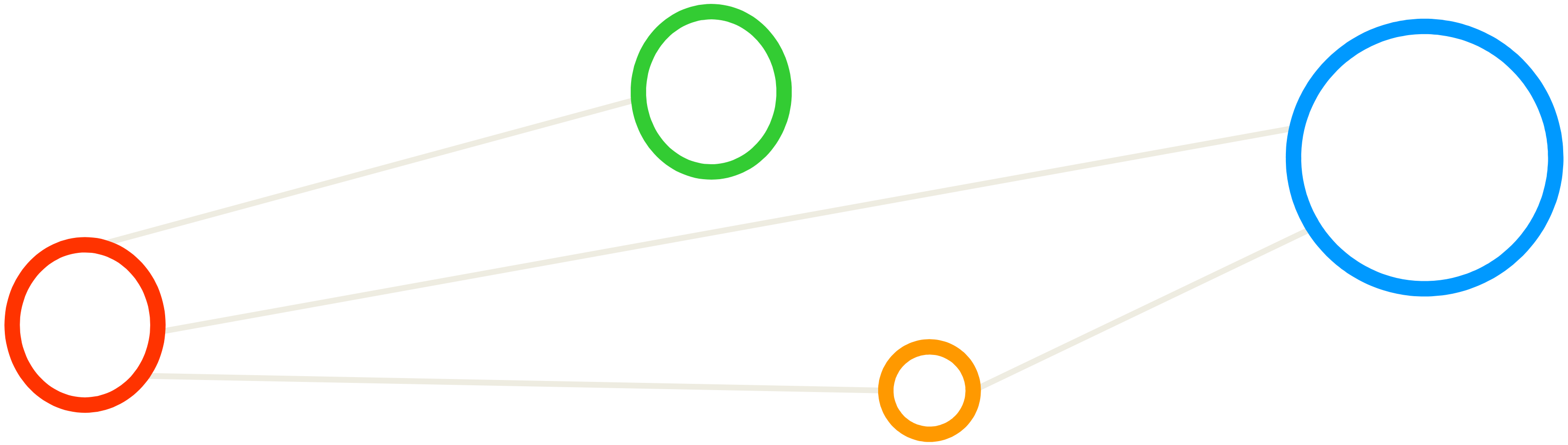


Outline

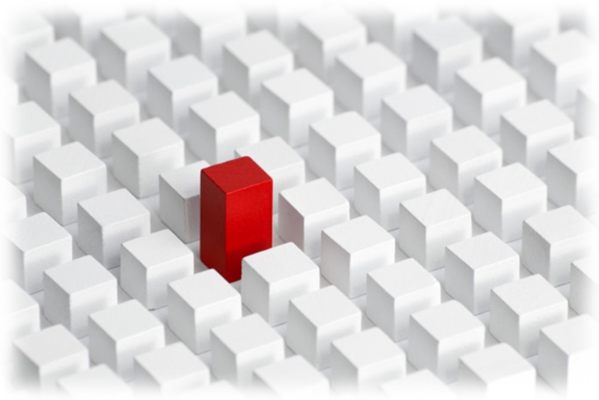
- Machine Learning Foundations
  - Motivation & Methods Overview
  - Simple Classification Application Example
  - Perceptron Model & Learning Algorithm
  - Decision Boundary & Linear Seperability
  - Training vs. Testing & Overfitting
- HPC-driven Data Analytics
  - Supervised Learning using parallel SVMs
  - Parallelization Benefits using Cross-Validation
  - Unsupervised Learning using parallel DBSCAN
  - Short Introduction to Deep Learning using GPGPU
  - Comparisons Machine Learning & Deep Learning
- Summary
- Appendix A, B & C: Selected In-depth Topics

- **Machine Learning requires a full university course covering topics beyond modeling & algorithms like statistical learning theory, regularization & validation techniques**
- **Using High Performance Computing (HPC) adds another level of complexity requiring a full HPC university course**

Machine Learning Foundations



‘Big Data‘ Motivation: Intertwine HPC & Machine Learning



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

> - **Machine learning / Data Mining is a technology that blends traditional data analysis methods with sophisticated algorithms for processing large volumes of data**
> - **Machine Learning / Data Mining is the process of automatically discovering useful information in large data repositories ideally following a systematic process**

modified from [1] Introduction to Data Mining

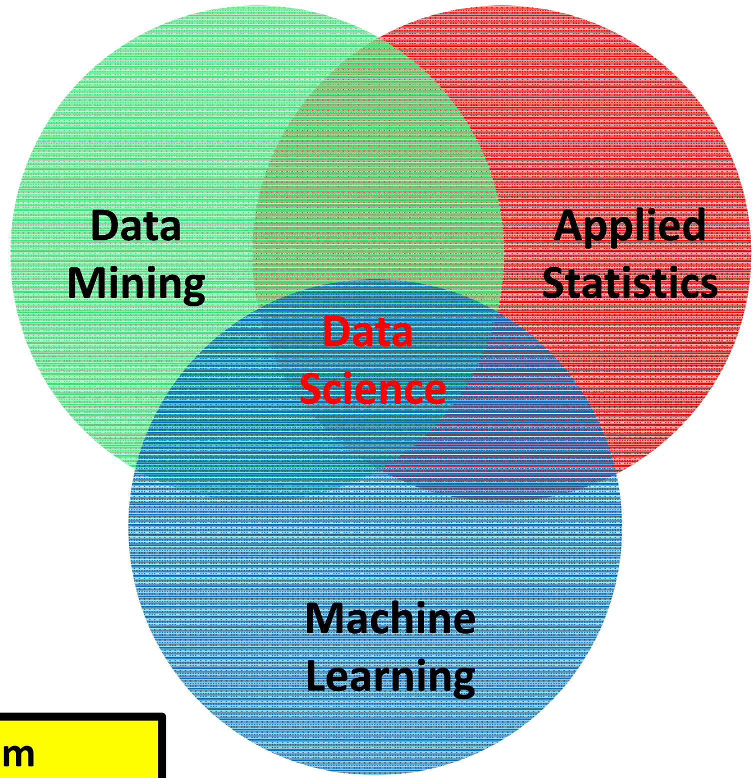
- Machine Learning & Statistical Data Mining
  - Traditional statistical approaches are still very useful to consider

> ➢ **Link to Talk by Bernd Mohr – HPC & Parallel Computing to solve memory limits and increase speed**

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, etc.
- Challenge: Data is often complex

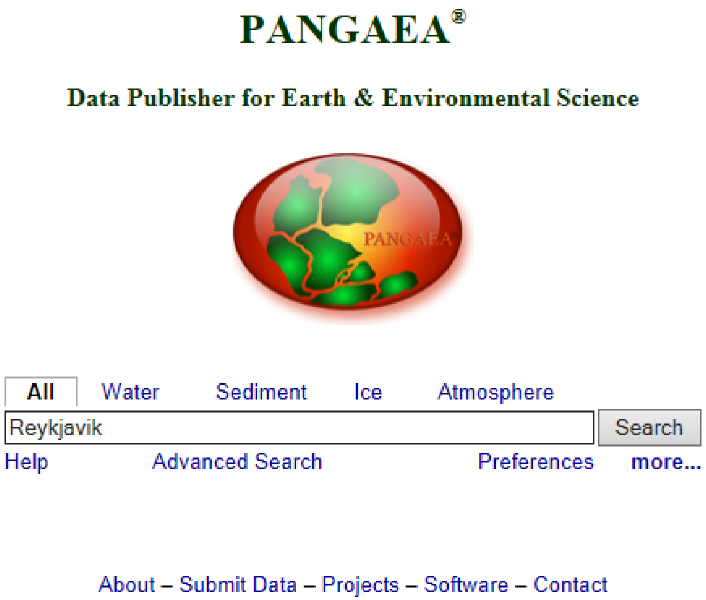


- **Machine learning is a very broad subject and goes from very abstract theory to extreme practice ('rules of thumb')**

Examples of Real Data Collections

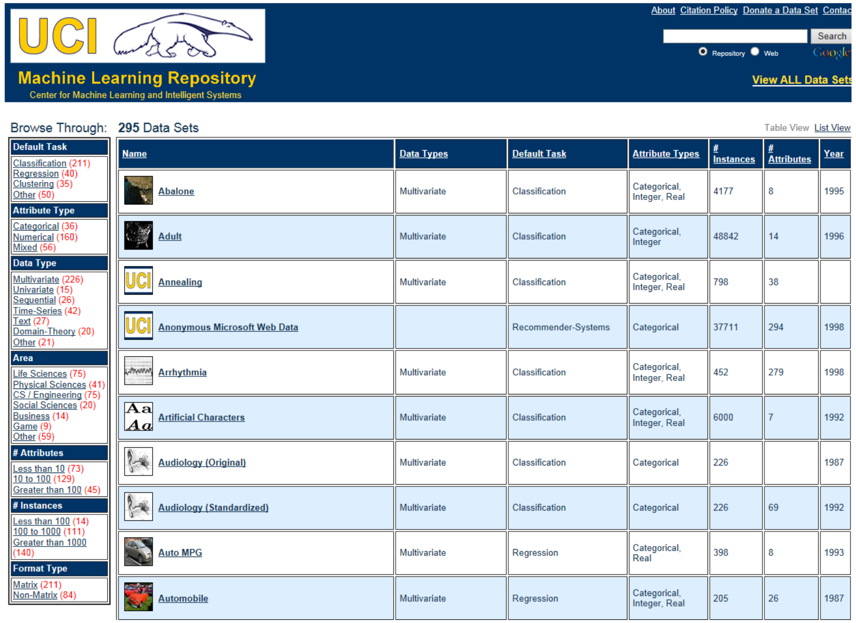
- Data collection of the earth and environmental science domain
  - Different from the known 'UCI machine learning repository examples'

(real science datasets examples)



[2] PANGAEA data collection

(examples for learning & comparisons)

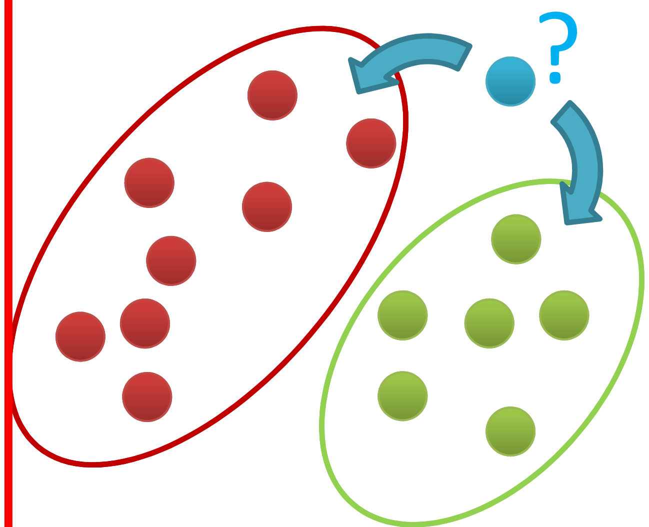


[3] UCI Machine Learning Repository

Methods Overview

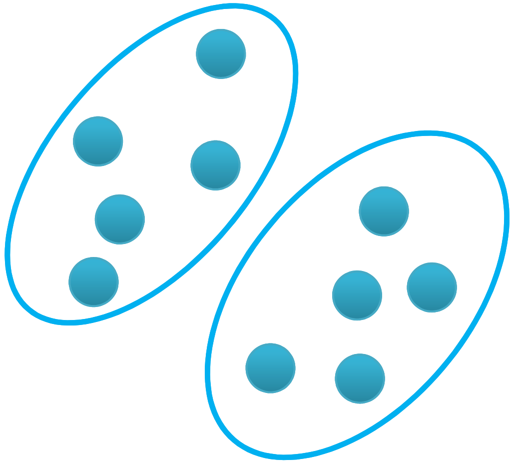
- **Machine learning methods can be roughly categorized in classification, clustering, or regression augmented with various techniques for data exploration, selection, or reduction**

Classification



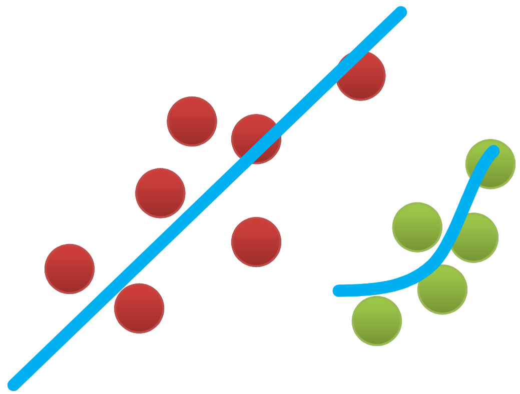
- Groups of data exist
- New data classified to existing groups

Clustering



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

Regression

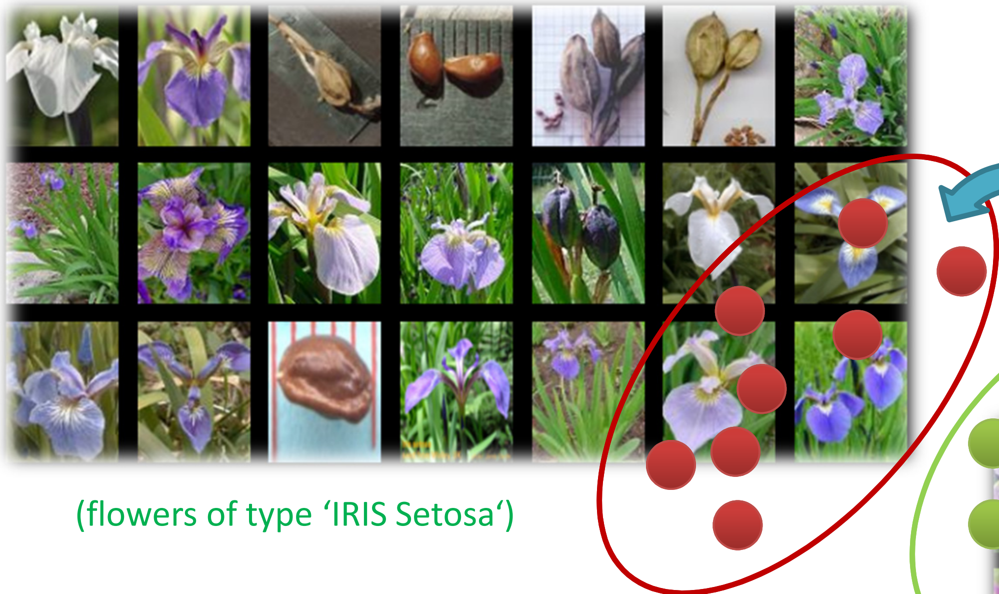


- Identify a line with a certain slope describing the data

Simple Application Example: Classification of a Flower

(1) Problem Understanding Phase

(what type of flower is this?)



(flowers of type ‘IRIS Setosa‘)

(flowers of type ‘IRIS Virginica‘)

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

[4] Image sources: Species Iris Group of North America Database, www.signa.org

The Learning Problem in the Example

(flowers of type ‘IRIS Setosa‘)

(flowers of type ‘IRIS Virginica‘)

[4] Image sources: Species Iris Group of North America Database, www.signa.org

Learning problem: A prediction task

- Determine whether a new Iris flower sample is a “Setosa” or “Virginica”
- Binary (two class) classification problem
- What attributes about the data help?

(what type of flower is this?)

Feasibility of Machine Learning in this Example

1. Some pattern exists:
   - Believe in a 'pattern with 'petal length' & 'petal width' somehow influence the type
2. No exact mathematical formula
   - To the best of our knowledge there is no precise formula for this problem
3. Data exists
   - Data collection from UCI Dataset „Iris"
   - 150 labelled samples (aka 'data points')
   - Balanced: 50 samples / class

[5] Image source: Wikipedia, Sepal

(four data attributes for each sample in the dataset)

- sepal length in cm
- sepal width in cm
- petal length in cm
- petal width in cm

(one class label for each sample in the dataset)

- class: Iris Setosa, or Iris Versicolour, or Iris Virginica

(2) Data Understanding Phase

[6] UCI Machine Learning Repository Iris Dataset

Understanding the Data – Check Metadata

- First: Check metadata if available (metadata is not always available in practice)
  - Example: Downloaded `iris.names` includes metadata about data

```
1. Title: Iris Plants Database
     Updated Sept 21 by C.Blake - Added discrepency information

2. Sources:
     (a) Creator: R.A. Fisher
     (b) Donor: Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)
     (c) Date: July, 1988

                   ...

5. Number of Instances: 150 (50 in each of three classes)

6. Number of Attributes: 4 numeric, predictive attributes and the
class

7. Attribute Information:
   1. sepal length in cm
   2. sepal width in cm
   3. petal length in cm
   4. petal width in cm
   5. class:
      -- Iris Setosa
      -- Iris Versicolour
      -- Iris Virginica
```

(Subject, title, or context)

(author, source, or creator)

(number of samples, instances)

(attribute information)

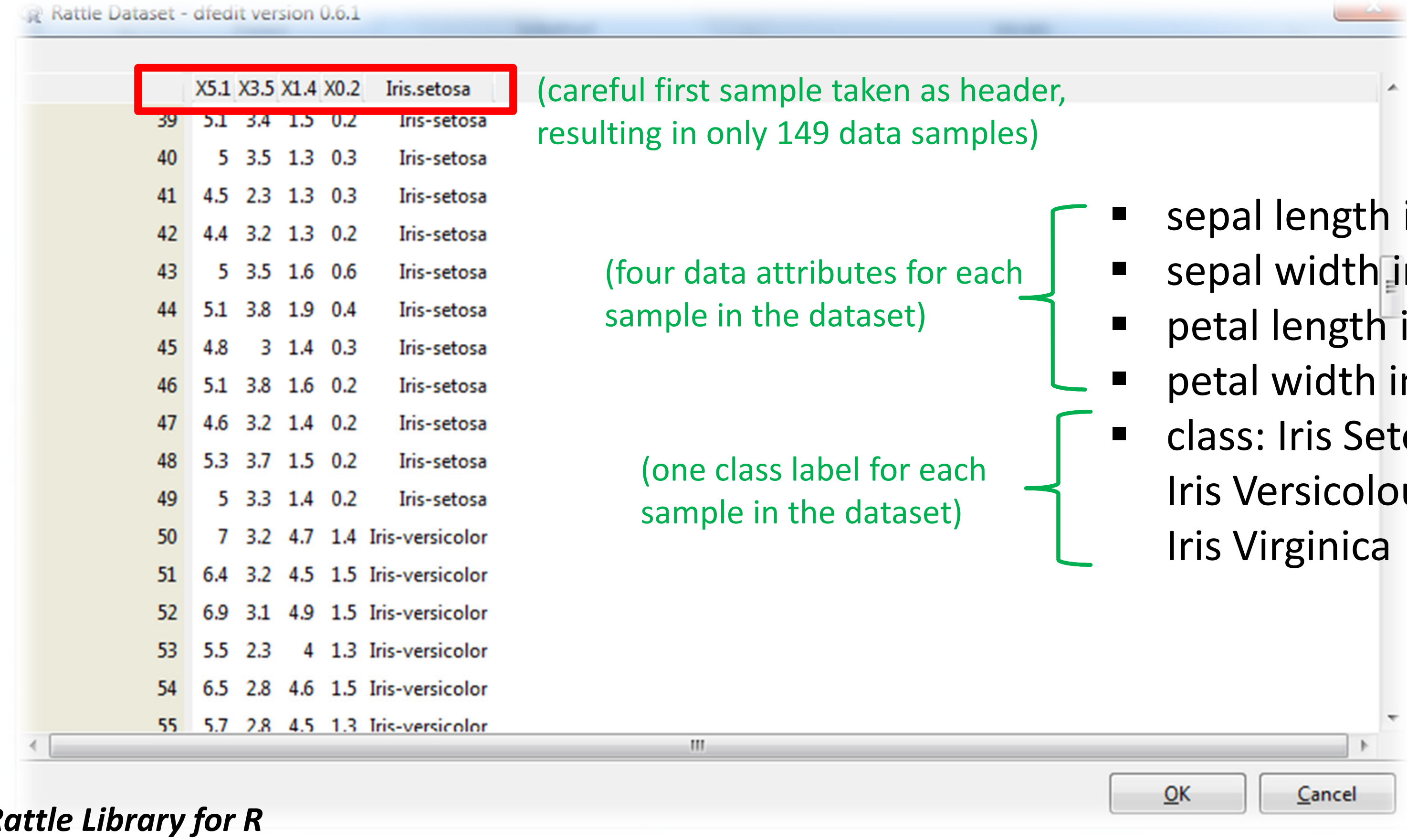
(detailed attribute information)

(detailed attribute information)

[6] UCI Machine Learning Repository Iris Dataset

Understanding the Data – Check Table View

- Second: Check table view of the dataset with some samples
  - E.g. Using a GUI like 'Rattle' (library of R), or Excel in Windows, etc.
  - E.g. Check the first row if there is header information or if is a sample



Rattle Dataset - dfedit version 0.6.1

	X5.1	X3.5	X1.4	X0.2	Iris.setosa
39	5.1	3.4	1.5	0.2	Iris-setosa
40	5	3.5	1.3	0.3	Iris-setosa
41	4.5	2.3	1.3	0.3	Iris-setosa
42	4.4	3.2	1.3	0.2	Iris-setosa
43	5	3.5	1.6	0.6	Iris-setosa
44	5.1	3.8	1.9	0.4	Iris-setosa
45	4.8	3	1.4	0.3	Iris-setosa
46	5.1	3.8	1.6	0.2	Iris-setosa
47	4.6	3.2	1.4	0.2	Iris-setosa
48	5.3	3.7	1.5	0.2	Iris-setosa
49	5	3.3	1.4	0.2	Iris-setosa
50	7	3.2	4.7	1.4	Iris-versicolor
51	6.4	3.2	4.5	1.5	Iris-versicolor
52	6.9	3.1	4.9	1.5	Iris-versicolor
53	5.5	2.3	4	1.3	Iris-versicolor
54	6.5	2.8	4.6	1.5	Iris-versicolor
55	5.7	2.8	4.5	1.3	Iris-versicolor

OK Cancel

(careful first sample taken as header, resulting in only 149 data samples)

(four data attributes for each sample in the dataset)

- sepal length in cm
- sepal width in cm
- petal length in cm
- petal width in cm

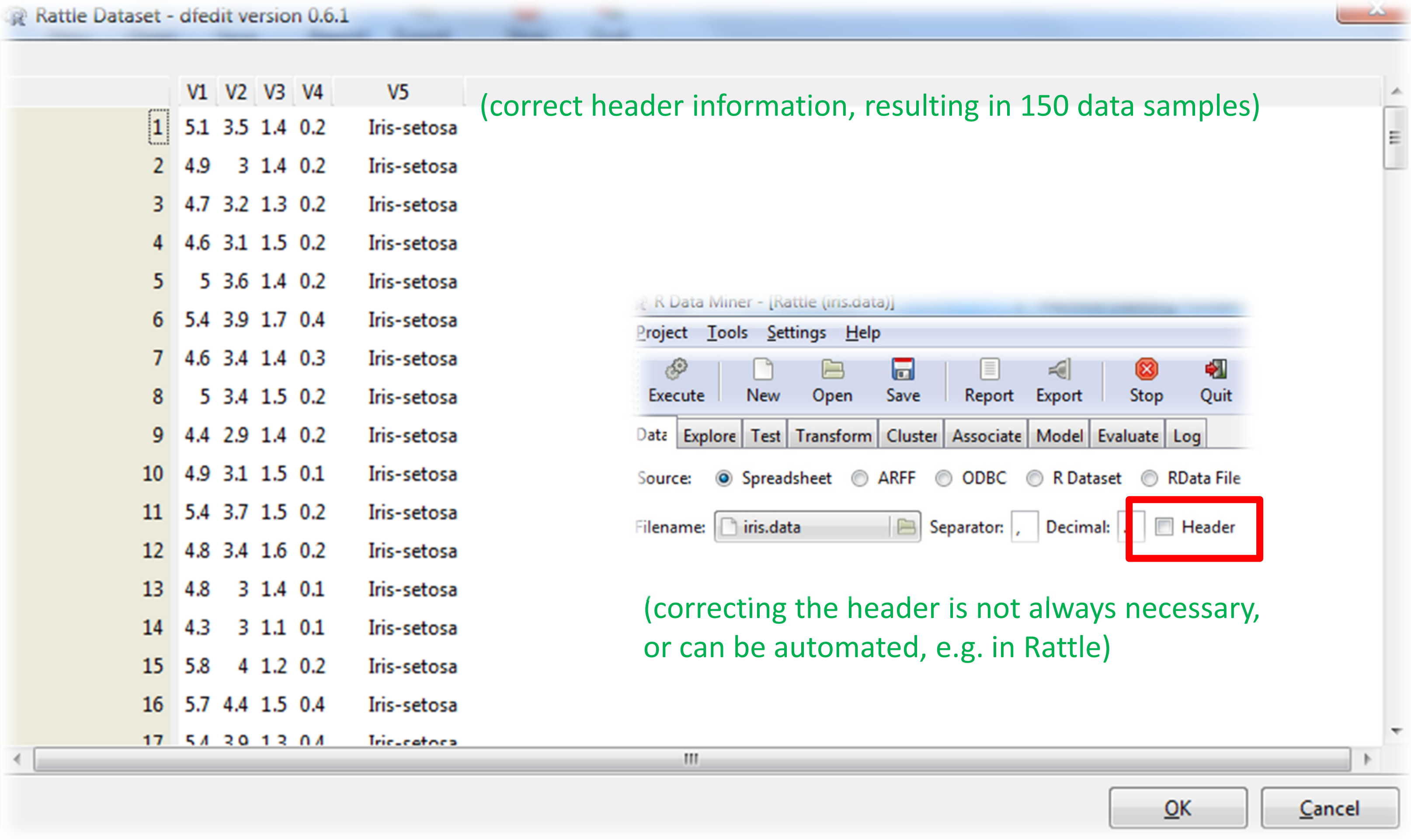
(one class label for each sample in the dataset)

- class: Iris Setosa, or Iris Versicolour, or Iris Virginica

[7] Rattle Library for R

Preparing the Data – Corrected Header

(3) Data Preparation Phase

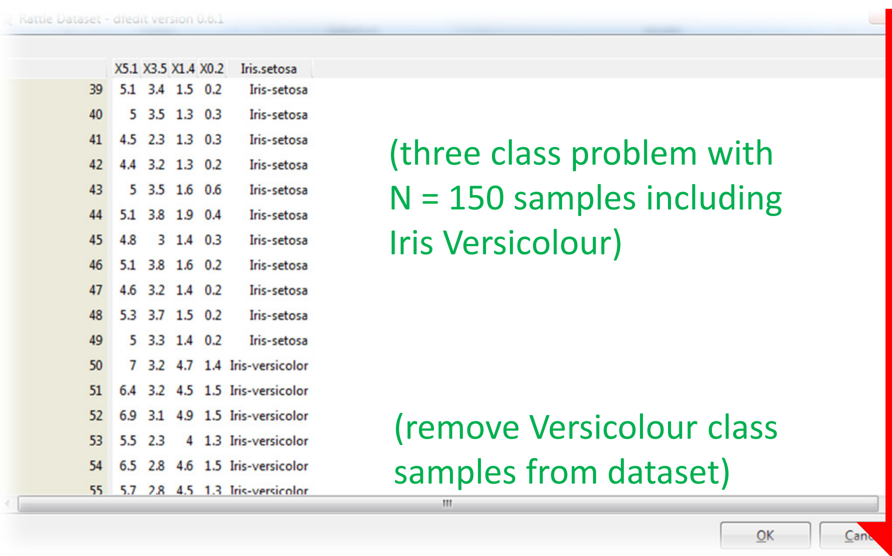


(correct header information, resulting in 150 data samples)

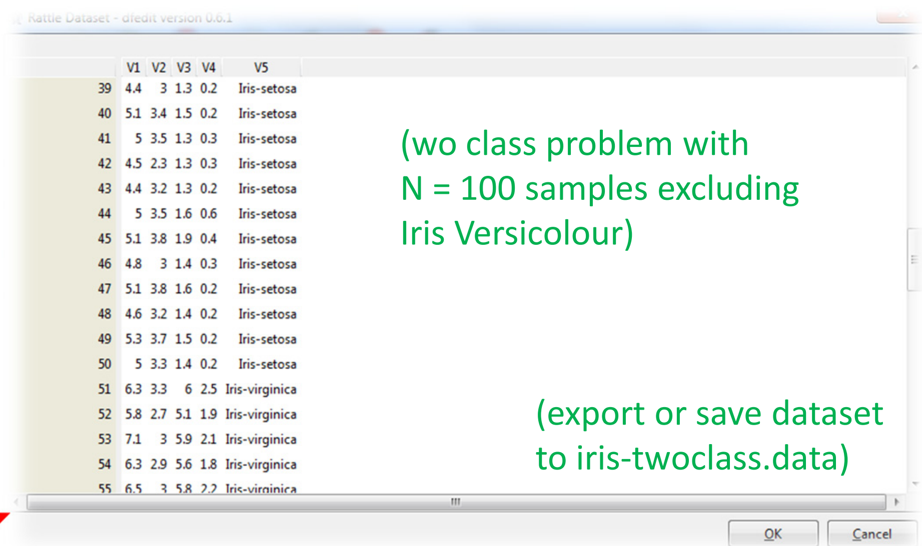
(correcting the header is not always necessary, or can be automated, e.g. in Rattle)

Preparing the Data – Remove Third Class Samples

- Data preparation means to prepare our data for our problem
  - In practice the whole dataset is rarely needed to solve one problem
  - E.g. apply several sampling strategies (but be aware of class balance)
- Recall: Our learning problem
  - Determine whether a new Iris flower sample is a “Setosa” or “Virginica”
  - Binary (two class) classification problem : ‘Setosa‘ or ‘Virginica‘



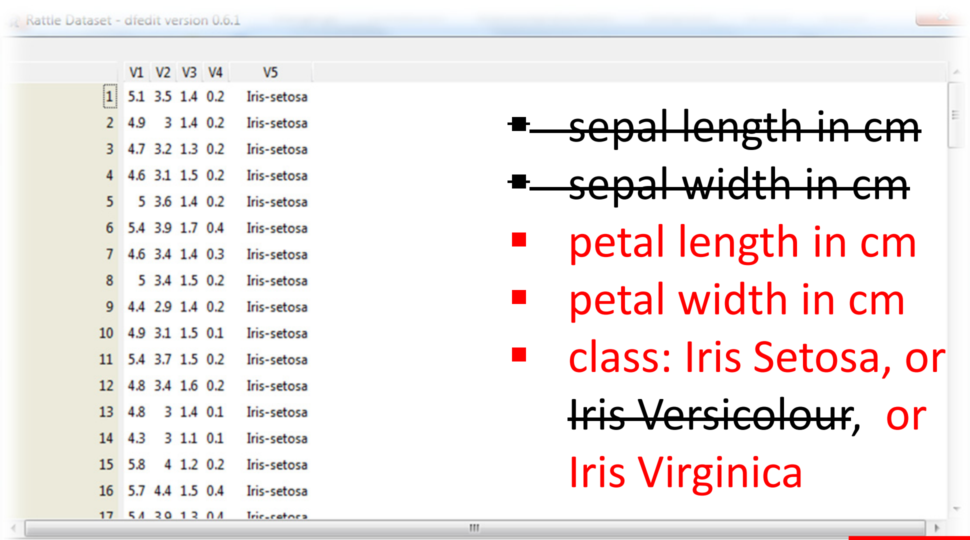
	X5.1	X3.5	X1.4	X0.2	Iris.setosa
39	5.1	3.4	1.5	0.2	Iris-setosa
40	5	3.5	1.3	0.3	Iris-setosa
41	4.5	2.3	1.3	0.3	Iris-setosa
42	4.4	3.2	1.3	0.2	Iris-setosa
43	5	3.5	1.6	0.6	Iris-setosa
44	5.1	3.8	1.9	0.4	Iris-setosa
45	4.8	3	1.4	0.3	Iris-setosa
46	5.1	3.8	1.6	0.2	Iris-setosa
47	4.6	3.2	1.4	0.2	Iris-setosa
48	5.3	3.7	1.5	0.2	Iris-setosa
49	5	3.3	1.4	0.2	Iris-setosa
50	7	3.2	4.7	1.4	Iris-versicolor
51	6.4	3.2	4.5	1.5	Iris-versicolor
52	6.9	3.1	4.9	1.5	Iris-versicolor
53	5.5	2.3	4	1.3	Iris-versicolor
54	6.5	2.8	4.6	1.5	Iris-versicolor
55	5.7	2.8	4.5	1.3	Iris-versicolor



	V1	V2	V3	V4	V5
39	4.4	3	1.3	0.2	Iris-setosa
40	5.1	3.4	1.5	0.2	Iris-setosa
41	5	3.5	1.3	0.3	Iris-setosa
42	4.5	2.3	1.3	0.3	Iris-setosa
43	4.4	3.2	1.3	0.2	Iris-setosa
44	5	3.5	1.6	0.6	Iris-setosa
45	5.1	3.8	1.9	0.4	Iris-setosa
46	4.8	3	1.4	0.3	Iris-setosa
47	5.1	3.8	1.6	0.2	Iris-setosa
48	4.6	3.2	1.4	0.2	Iris-setosa
49	5.3	3.7	1.5	0.2	Iris-setosa
50	5	3.3	1.4	0.2	Iris-setosa
51	6.3	3.3	6	2.5	Iris-virginica
52	5.8	2.7	5.1	1.9	Iris-virginica
53	7.1	3	5.9	2.1	Iris-virginica
54	6.3	2.9	5.6	1.8	Iris-virginica
55	6.5	3	5.8	2.2	Iris-virginica

Preparing the Data – Feature Selection Process

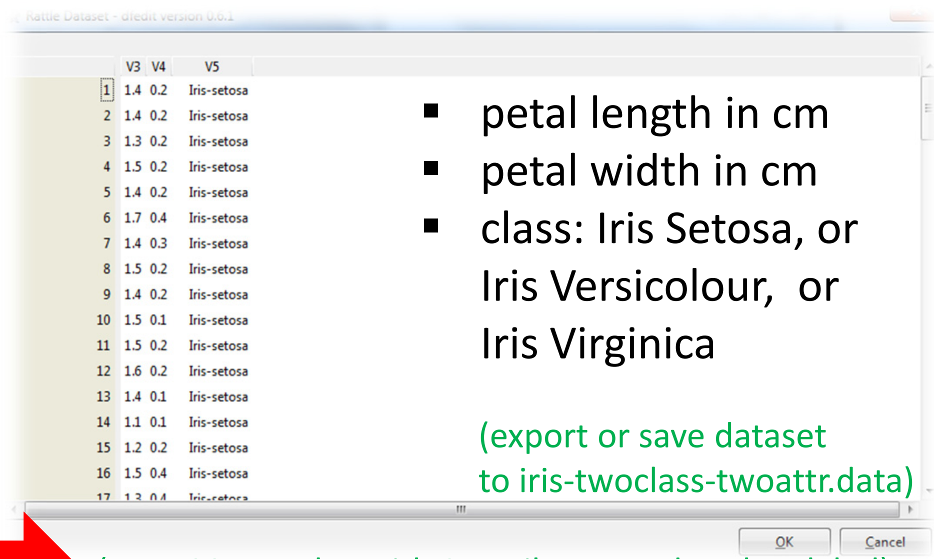
- Data preparation means to prepare our data for our problem
  - In practice the whole dataset is rarely needed to solve one problem
  - E.g. perform feature selection (aka remove not needed attributes)
- Recall: Our believed pattern in the data
  - A 'pattern with 'petal length' & 'petal width' somehow influence the type



	V1	V2	V3	V4	V5
1	5.1	3.5	1.4	0.2	Iris-setosa
2	4.9	3	1.4	0.2	Iris-setosa
3	4.7	3.2	1.3	0.2	Iris-setosa
4	4.6	3.1	1.5	0.2	Iris-setosa
5	5	3.6	1.4	0.2	Iris-setosa
6	5.4	3.9	1.7	0.4	Iris-setosa
7	4.6	3.4	1.4	0.3	Iris-setosa
8	5	3.4	1.5	0.2	Iris-setosa
9	4.4	2.9	1.4	0.2	Iris-setosa
10	4.9	3.1	1.5	0.1	Iris-setosa
11	5.4	3.7	1.5	0.2	Iris-setosa
12	4.8	3.4	1.6	0.2	Iris-setosa
13	4.8	3	1.4	0.1	Iris-setosa
14	4.3	3	1.1	0.1	Iris-setosa
15	5.8	4	1.2	0.2	Iris-setosa
16	5.7	4.4	1.5	0.4	Iris-setosa
17	5.4	3.9	1.3	0.4	Iris-setosa

- ~~sepal length in cm~~
- ~~sepal width in cm~~
- petal length in cm
- petal width in cm
- class: Iris Setosa, or ~~Iris Versicolour~~, or Iris Virginica

(N = 100 samples with 4 attributes and 1 class label)



	V3	V4	V5
1	1.4	0.2	Iris-setosa
2	1.4	0.2	Iris-setosa
3	1.3	0.2	Iris-setosa
4	1.5	0.2	Iris-setosa
5	1.4	0.2	Iris-setosa
6	1.7	0.4	Iris-setosa
7	1.4	0.3	Iris-setosa
8	1.5	0.2	Iris-setosa
9	1.4	0.2	Iris-setosa
10	1.5	0.1	Iris-setosa
11	1.5	0.2	Iris-setosa
12	1.6	0.2	Iris-setosa
13	1.4	0.1	Iris-setosa
14	1.1	0.1	Iris-setosa
15	1.2	0.2	Iris-setosa
16	1.5	0.4	Iris-setosa
17	1.3	0.4	Iris-setosa

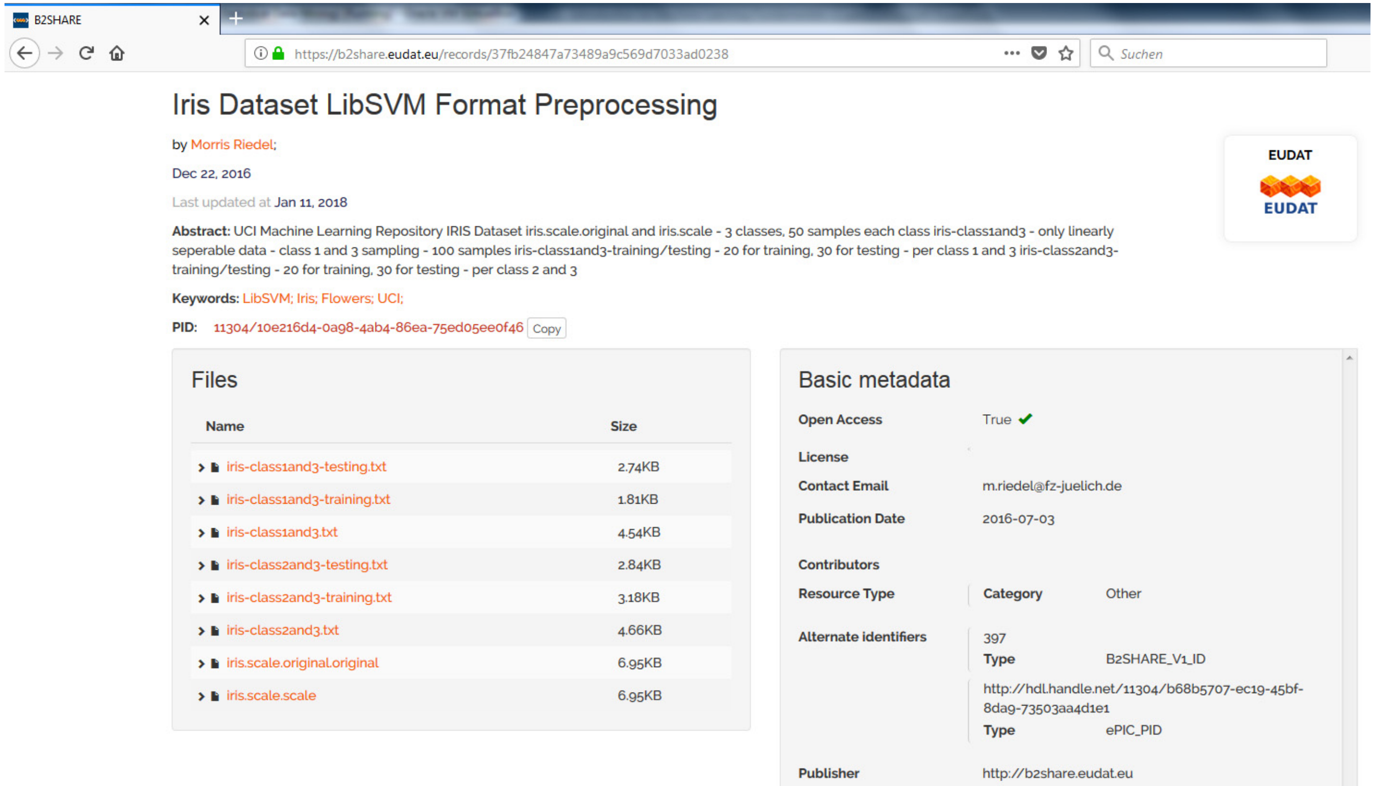
- petal length in cm
- petal width in cm
- class: Iris Setosa, or Iris Versicolour, or Iris Virginica

(export or save dataset to iris-twoclass-twoattr.data)

(N = 100 samples with 2 attributes and 1 class label)

Iris Dataset – Open Data

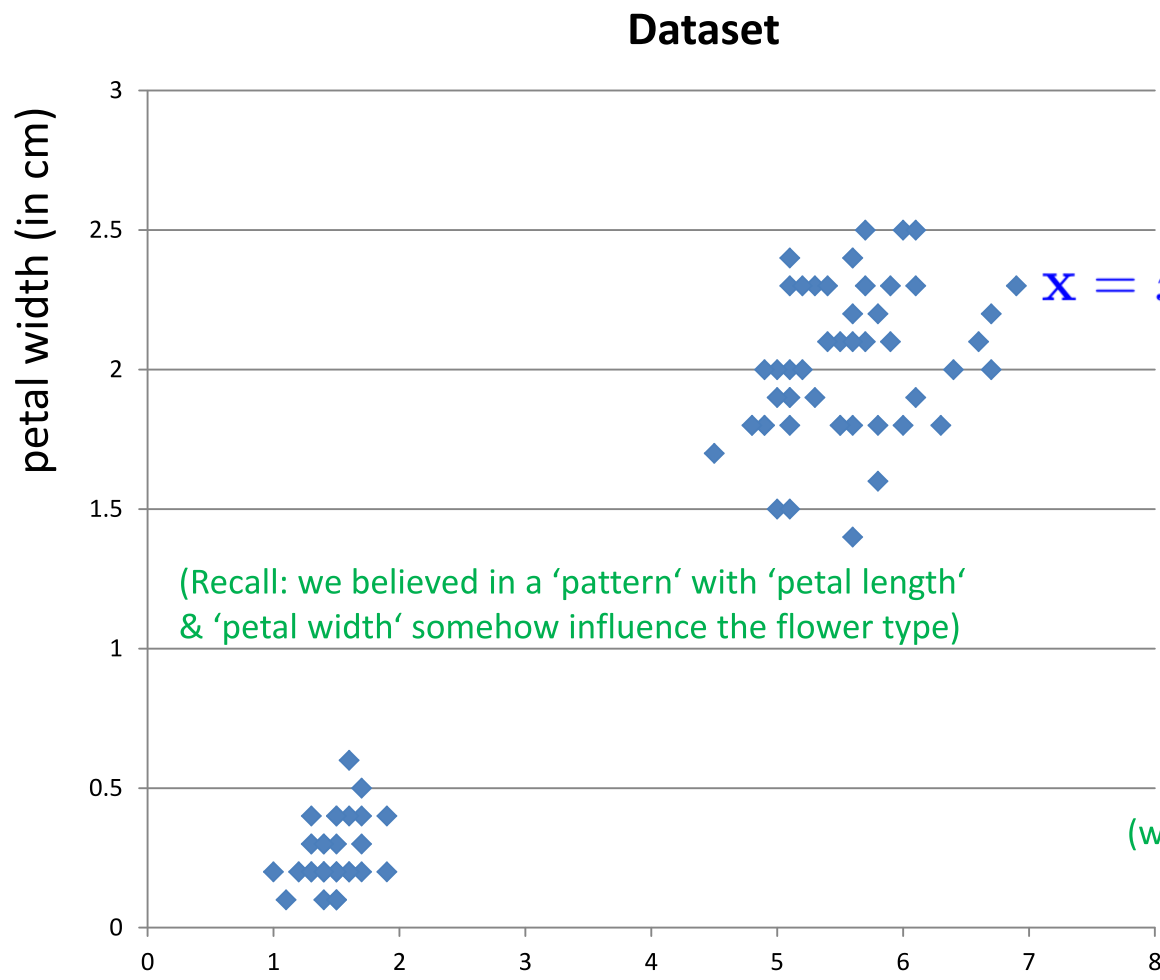
- Different samples of the original Iris dataset
  - Created for linear seperability and non-linear seperability



[14] Iris Dataset

B2SHARE Store and Share Research Data

Check Preparation Phase: Plotting the Data



(attributes with d=2)

$\mathbf{x} = x_1, \ldots, x_d$

(x1 is petal length,
x2 is petal width)

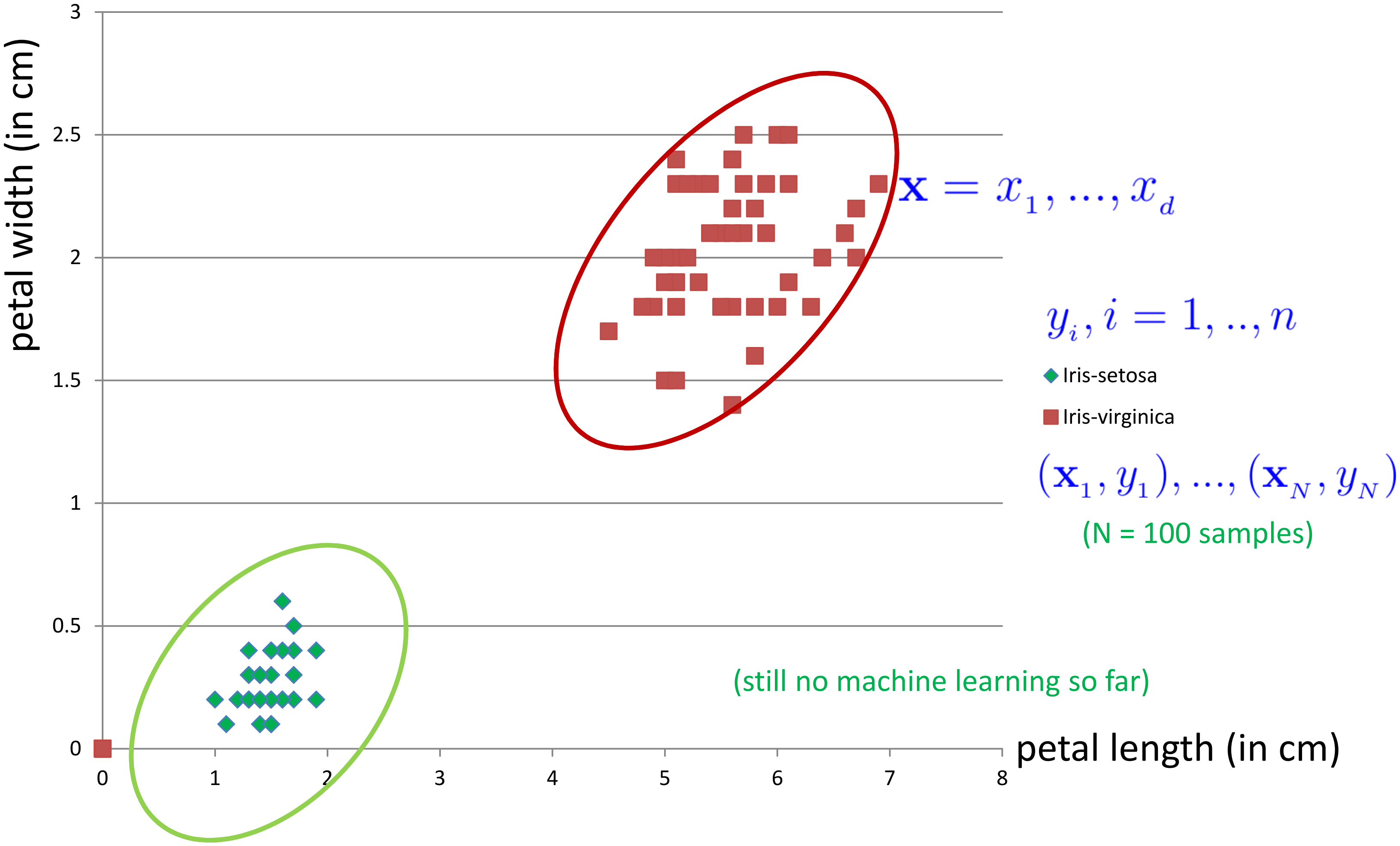
$(\mathbf{x}_1), \ldots, (\mathbf{x}_N)$

(N = 100 samples)

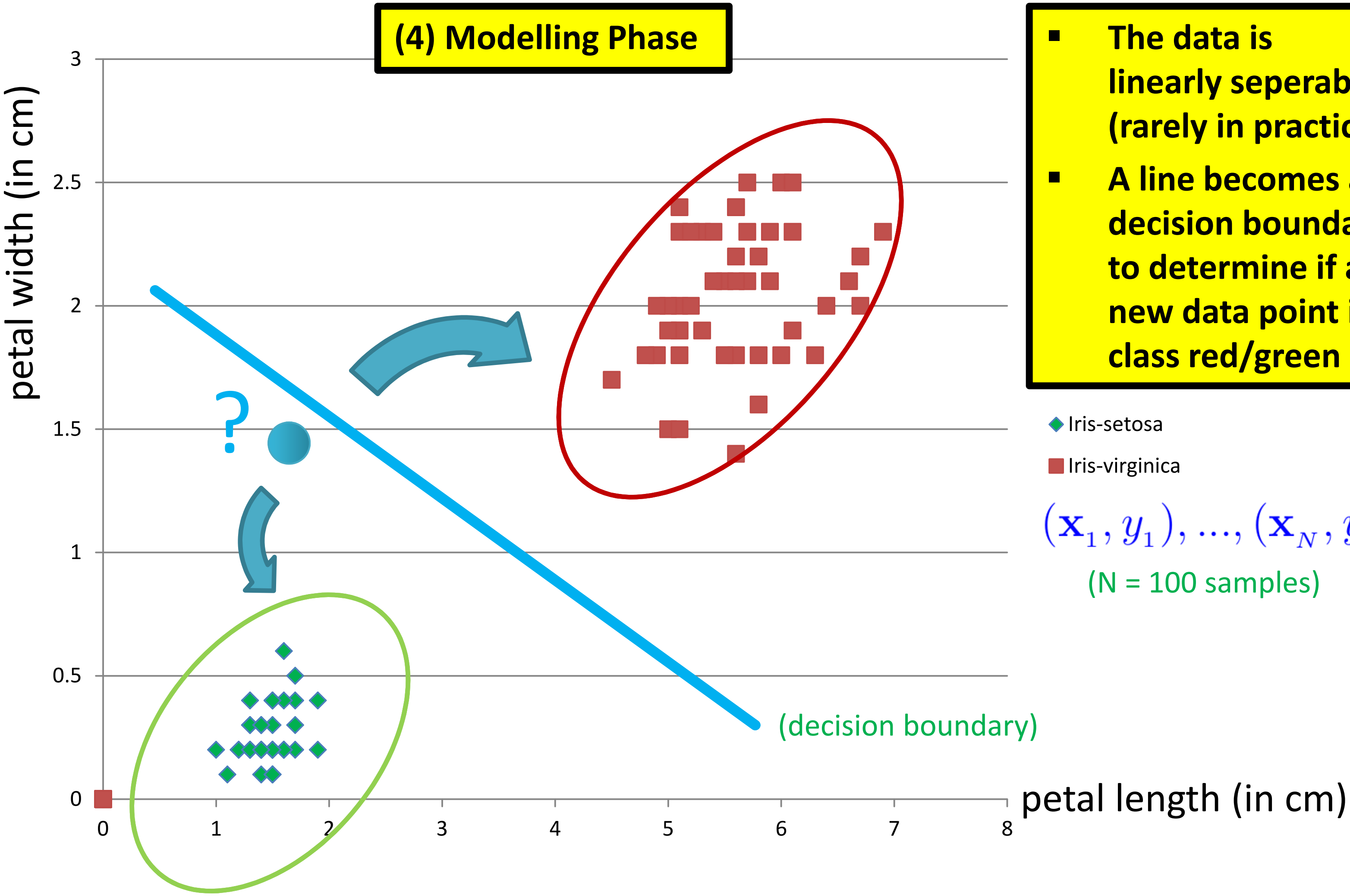
(what about the class labels?)

petal length (in cm)

Check Preparation Phase: Class Labels



Linearly Seperable Data & Linear Decision Boundary



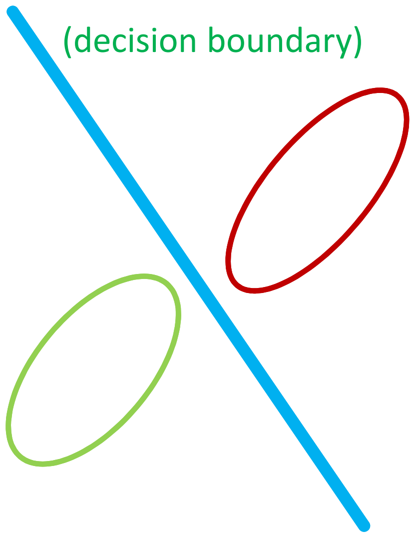
- The data is linearly seperable (rarely in practice)
- A line becomes a decision boundary to determine if a new data point is class red/green

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

(N = 100 samples)

Separating Line & Mathematical Notation

- Data exploration results
  - A line can be crafted between the classes since linearly seperable data
  - All the data points representing Iris-setosa will be below the line
  - All the data points representing Iris-virginica will be above the line
- More formal mathematical notation
  - Input: $\mathbf{x} = x_1, \ldots, x_d$ (attributes of flowers)
  - Output: class +1 (Iris-virginica) or class -1 (Iris-setosa)



Iris-virginica if $\sum_{i=1}^{d} w_i x_i > threshold$

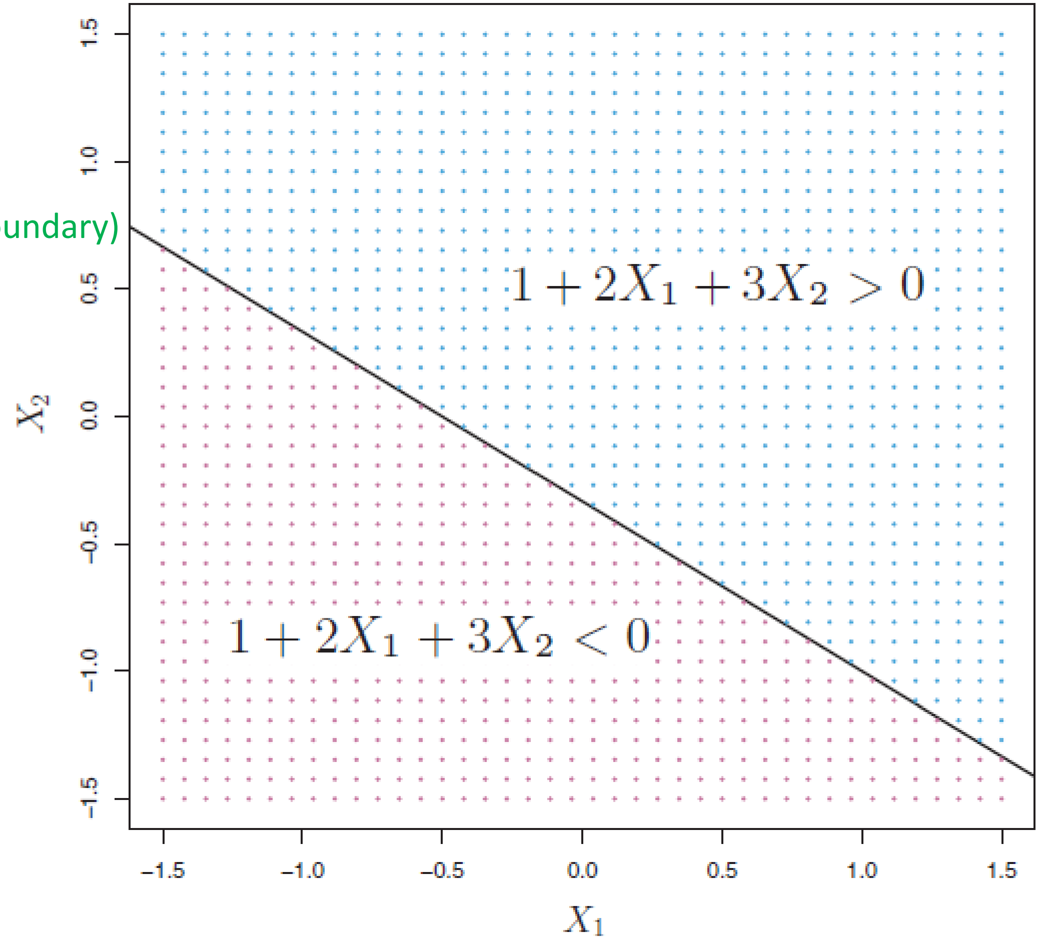
Iris-setosa if $\sum_{i=1}^{d} w_i x_i < threshold$

(w_i and threshold are still unknown to us)

$$sign\left(\left(\sum_{i=1}^{d} w_i x_i\right) - threshold\right)$$ (compact notation)

Separating Line & 'Decision Space' Example

(decision boundary)



(equation of a line)

$$1 + 2X_1 + 3X_2 = 0$$

(all points X_i on this line have to satisfy this equation)

modified from [13] An Introduction to Statistical Learning

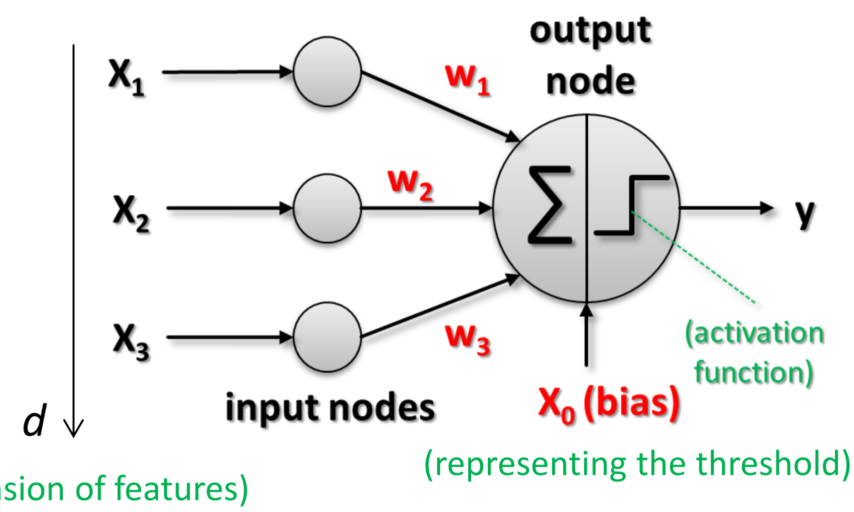
A Simple Linear Learning Model – The Perceptron

- Human analogy in learning *[8] F. Rosenblatt, 1957*
  - Human brain consists of nerve cells called neurons
  - Human brain learns by changing the strength of neuron connections (w_i) upon repeated stimulation by the same impulse (aka a 'training phase')
  - Training a perceptron model means adapting the weights w_i
  - Done until they fit input-output relationships of the given 'training data'

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

(training data)

$$\underset{\text{(activation function, +1 or -1)}}{sign}\Bigg(\underbrace{\left(\sum_{i=1}^{d} w_i x_i\right) - \overset{\text{(modelled as bias term)}}{threshold}}_{\text{(the signal)}}\Bigg)$$



Perceptron – Example of a Boolean Function

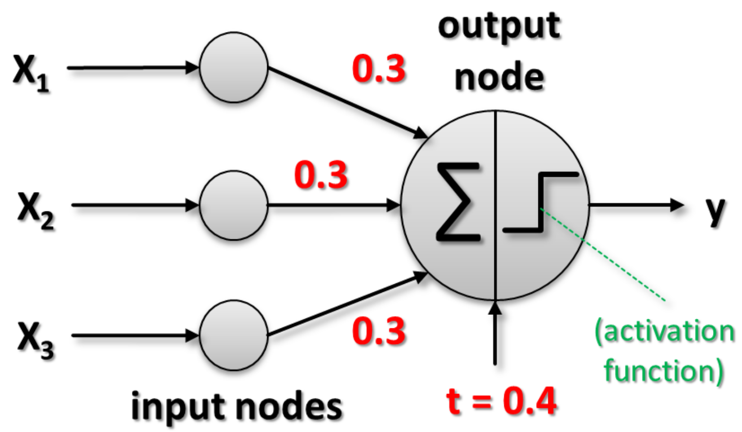
	x1	x2	x3	y
1	1	0	0	-1
2	1	0	1	1
3	1	1	0	1
4	1	1	1	1
5	0	0	1	-1
6	0	1	0	-1
7	0	1	1	1
8	0	0	0	-1

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

(training data)

(training phase)

$$\boxed{sign}\left(\left(\sum_{i=1}^{d} w_i x_i\right) - threshold\right)$$



(trained perceptron model)

- Output node interpretation
  - More than just the weighted sum of the inputs – threshold (aka bias)
  - Activation function sign (weighted sum): takes sign of the resulting sum

$$y = 1, \textbf{if } 0.3x_1 + 0.3x_2 + 0.3x_3 - 0.4 > 0$$

(e.g. consider sample #3, sum is positive (0.2) → +1)

$$y = -1, \textbf{if } 0.3x_1 + 0.3x_2 + 0.3x_3 - 0.4 < 0$$

(e.g. consider sample #6, sum is negative (-0.1) → -1)

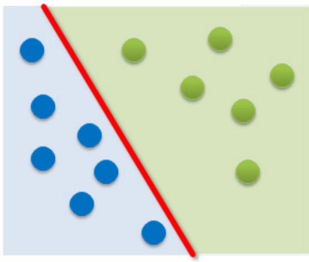
Summary Perceptron & Hypothesis Set h(x)

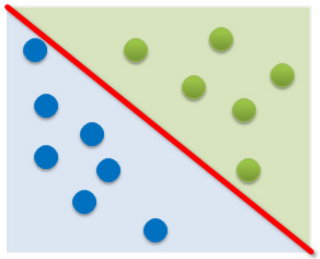
- When: Solving a linear classification problem *[8] F. Rosenblatt, 1957*
  - Goal: learn a simple value (+1/-1) above/below a certain threshold
  - Class label renamed: Iris-setosa = -1 and Iris-virginica = +1
- Input: $\mathbf{x} = x_1, \ldots, x_d$ (attributes in one dataset)
- Linear formula (take attributes and give them different weights – think of 'impact of the attribute')
  - All learned formulas are different hypothesis for the given problem

$$h(\mathbf{x}) = \boxed{sign}\left(\left(\sum_{i=1}^{d} w_i x_i\right) - threshold\right); h \in \mathcal{H}$$

(parameters that define one hypothesis vs. another)

(each green space and blue space are regions of the same class label determined by sign function)





(red parameters correspond to the redline in graphics)

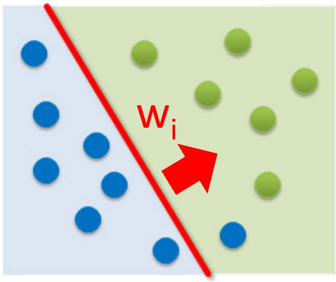
(but question remains: how do we actually learn w_i and threshold?)

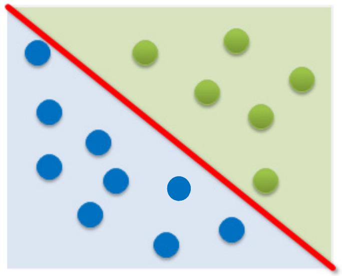
Perceptron Learning Algorithm – Understanding Vector W

- When: If we believe there is a linear pattern to be detected
  - Assumption: Linearly seperable data (lets the algorithm converge)
  - Decision boundary: perpendicular vector w_i fixes orientation of the line

$$\mathbf{w}^T\mathbf{x} = 0$$
$$\mathbf{w} \cdot \mathbf{x} = 0$$

(points on the decision boundary satisfy this equation)





$$h(\mathbf{x}) = sign(\mathbf{w}^T\mathbf{x})$$

(vector notation, using T = transpose)

$$\mathbf{w}_i = (w_{i1}, w_{i2}, \ldots, w_d)$$

$$\mathbf{w}_i^T = \begin{bmatrix} w_{i1} \\ w_{i2} \\ \cdots \\ w_{id} \end{bmatrix}$$

- Possible via simplifications since we also need to learn the threshold:

$$h(\mathbf{x}) = sign\left(\left(\sum_{i=1}^{d} w_i x_i\right) + w_0\right); w_0 = -threshold$$

$$\mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_d)$$

$$h(\mathbf{x}) = sign\left(\left(\sum_{i=0}^{d} w_i x_i\right)\right); x_0 = 1$$

$$h(\mathbf{x}) = sign(\mathbf{w} \cdot \mathbf{x})$$

(equivalent dotproduct notation)

[9] Rosenblatt, 1958

(all notations are equivalent and result is a scalar from which we derive the sign)

Understanding the Dot Product – Example & Interpretation

- 'Dot product'

$$\mathbf{u} \cdot \mathbf{v} = \sum_{i=1}^{n} u_i v_i$$

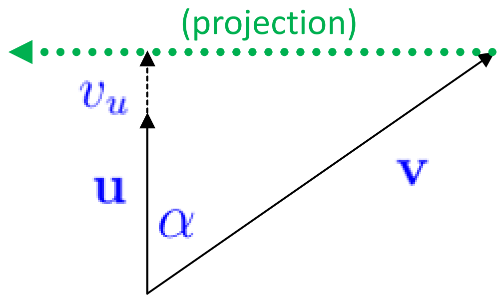
$$h(\mathbf{x}) = sign\left(\left(\sum_{i=0}^{d} w_i x_i\right)\right); x_0 = 1$$

$$h(\mathbf{x}) = sign(\mathbf{w} \cdot \mathbf{x})$$

(our example)

  - Given two vectors
  - Multiplying corresponding components of the vector
  - Then adding the resulting products
  - Simple example: $(2,3) \cdot (4,1) = 2 * 4 + 3 * 1 = \boxed{11}$ (a scalar!)
  - Interesting: Dot product of two vectors is a scalar
- 'Projection capabilities of Dot product' (simplified)
  - Orthogonal projection of vector $\mathbf{v}$ in the direction of vector $\mathbf{u}$

$$\mathbf{u} \cdot \mathbf{v} = (\|v\| cos(\alpha))\|u\| = v_u \|u\|$$



- Normalize using length of vector

$$\frac{\mathbf{u}}{\|\mathbf{u}\|} \quad \|\mathbf{u}\| = length(\mathbf{u}) = L_2 norm = \sqrt{\mathbf{u} \cdot \mathbf{u}}$$

> ➢ **Dot Products are important in machine learning, e.g. in Support Vector Machines, see Appendix B**

Perceptron Learning Algorithm – Learning Step

- Iterative Method using (labelled) training data $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)$

   (one point at a time is picked)

1. Pick one misclassified training point where:

   $$sign(\mathbf{w}^T \mathbf{x}_n) \neq y_n$$

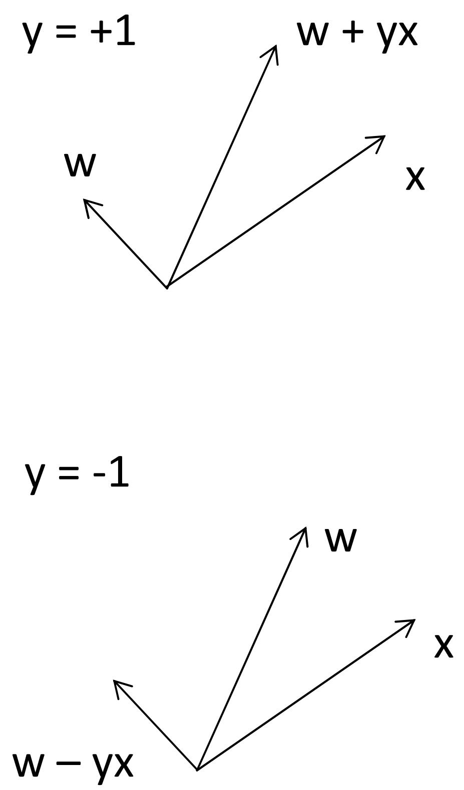
2. Update the weight vector:

   $$\mathbf{w} \leftarrow \mathbf{w} + y_n \mathbf{x}_n$$

   (y_n is either +1 or -1)

   (a) adding a vector or
   (b) subtracting a vector

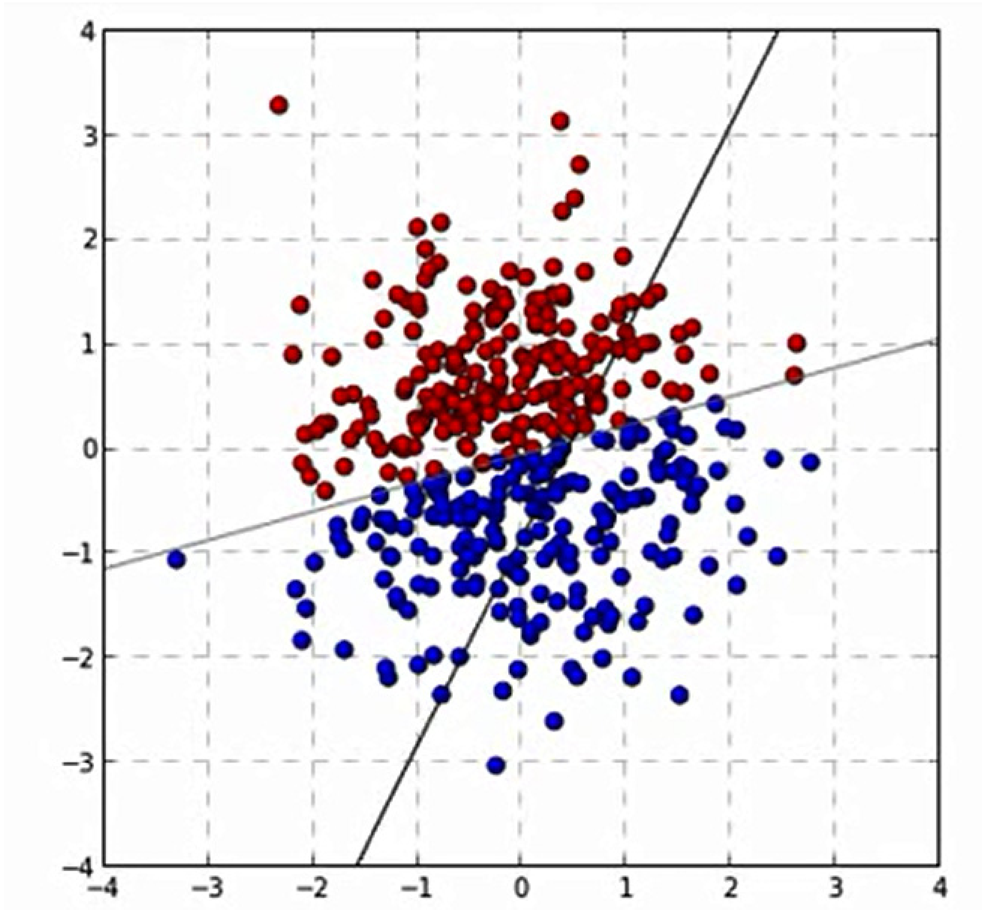
(a) (b)



- Terminates when there are no misclassified points

  (converges only with linearly seperable data)

[Video] Perceptron Learning Algorithm



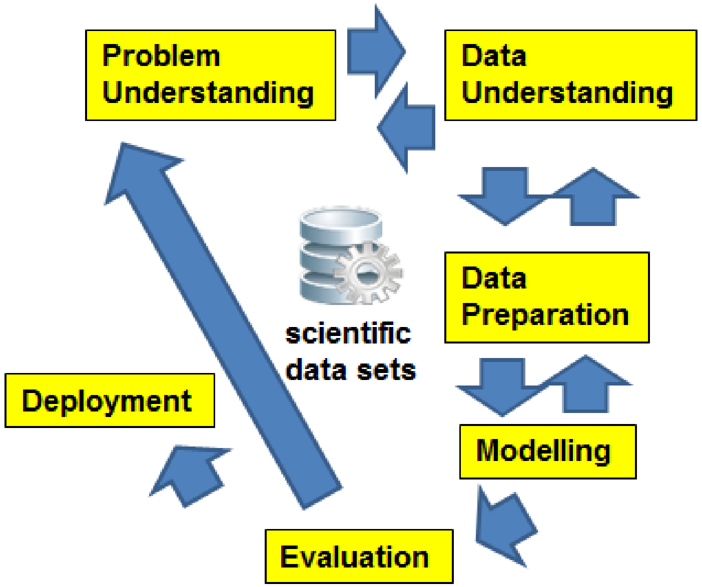
[10] PLA Video

Systematic Process to Support Learning From Data

- Systematic data analysis guided by a 'standard process'
  - Cross-Industry Standard Process for Data Mining (CRISP-DM)

- **A data mining project is guided by these six phases:**
  (1) Problem Understanding;
  (2) Data Understanding;
  (3) Data Preparation;
  (4) Modeling;
  (5) Evaluation;
  (6) Deployment

(learning takes place)



- Lessons Learned from Practice
  - Go back and forth between the different six phases

[11] C. Shearer, CRISP-DM model, Journal Data Warehousing, 5:13

> ➢ **A more detailed description of all six CRISP-DM phases is in the Appendix A of the slideset**

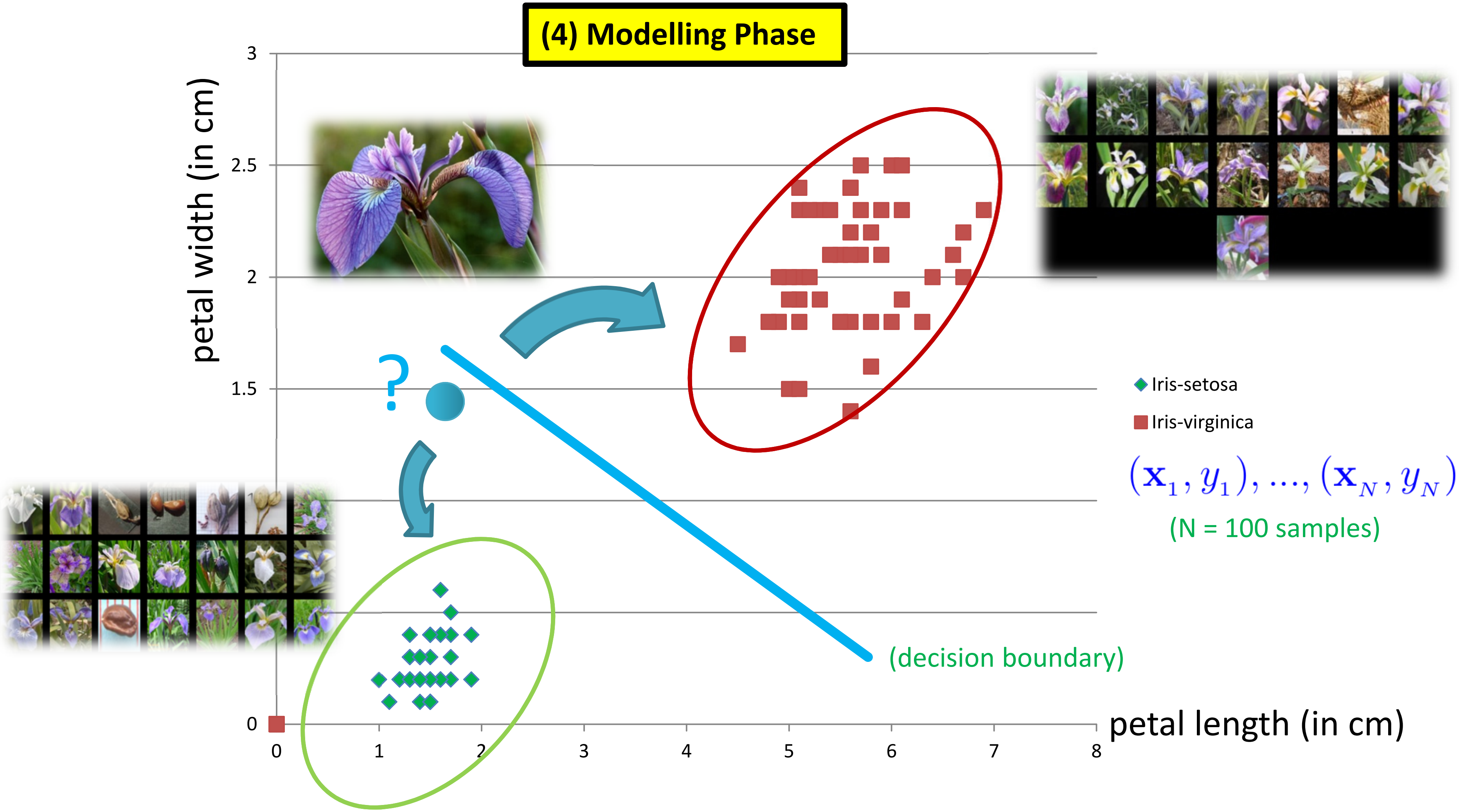
Machine Learning & Data Mining Tasks in Applications

- **Machine learning tasks can be divided into two major categories: Predictive and Descriptive Tasks**

[1] Introduction to Data Mining

- Predictive Tasks
  - Predicts the value of an attribute based on values of other attributes
  - Target/dependent variable: attribute to be predicted
  - Explanatory/independent variables: attributed used for making predictions
  - E.g. predicting the species of a flower based on characteristics of a flower
- Descriptive Tasks
  - Derive patterns that summarize the underlying relationships in the data
  - Patterns here can refer to correlations, trends, trajectories, anomalies
  - Often exploratory in nature and frequently require postprocessing
  - E.g. credit card fraud detection with unusual transactions for owners

Predicting Task: Obtain Class of a new Flower 'Data Point'



[4] Image sources: Species Iris Group of North America Database, www.signa.org

Summary Terminologies & Different Dataset Elements

- Target Function $f : X \rightarrow Y$
  - Ideal function that 'explains' the data we want to learn
- Labelled Dataset (samples)
  - 'in-sample' data given to us: $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)$
- Learning vs. Memorizing
  - The goal is to create a system that works well 'out of sample'
  - In other words we want to classify 'future data' (ouf of sample) correct
- Dataset Part One: Training set

  (4) Modelling Phase
  - Used for training a machine learning algorithms
  - Result after using a training set: a trained system
- Dataset Part Two: Test set

  (5) Evaluation Phase
  - Used for testing whether the trained system might work well
  - Result after using a test set: accuracy of the trained model

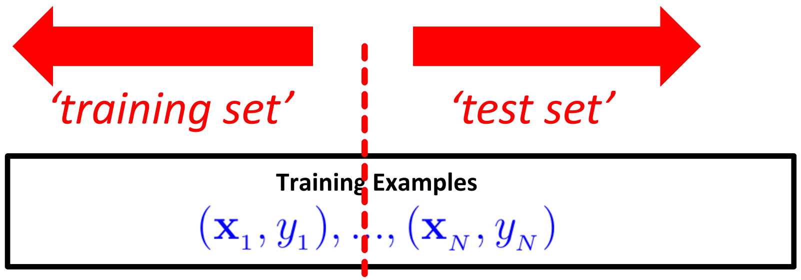
Model Evaluation – Training and Testing Phases

- Different Phases in Learning
  - Training phase is a hypothesis search
  - Testing phase checks if we are on right track (once the hypothesis clear)
- Work on 'training examples'
  - Create two disjoint datasets
  - One used for training only (aka training set)
  - Another used for testing only (aka test set)
  - Exact seperation is rule of thumb per use case (e.g. 10 % training, 90% test)
  - Practice: If you get a dataset take immediately test data away ('throw it into the corner and forget about it during modelling')
  - Reasoning: Once we learned from training data it has an 'optimistic bias'

(4) Modelling Phase

(5) Evaluation Phase

(e.g. student exam training on examples to get E_{in} ‚down', then test via exam)



(historical records, groundtruth data, examples)

Model Evaluation – Testing Phase & Confusion Matrix

(5) Evaluation Phase

- Model is fixed
  - Model is just used with the testset
  - Parameter w_i are set and we have a linear decision function
- Evaluation of model performance
  - Counts of test records that are incorrectly predicted $sign(\mathbf{w}^T \mathbf{x}_n) \neq y_n$
  - Counts of test records that are correctly predicted $sign(\mathbf{w}^T \mathbf{x}_n) = y_n$
  - E.g. create confusion matrix for a two class problem

Counting per sample		Predicted Class	
		Class = 1	Class = 0
Actual Class	Class = 1	f_{11}	f_{10}
	Class = 0	f_{01}	f_{00}

(serves as a basis for further performance metrics usually used)

Model Evaluation – Testing Phase & Performance Metrics

Counting per sample		Predicted Class	
		Class = 1	Class = 0
Actual Class	Class = 1	f_{11}	f_{10}
	Class = 0	f_{01}	f_{00}

(5) Evaluation Phase

(100% accuracy in learning often points to problems using machine learning methos in practice)

- Accuracy (usually in %)

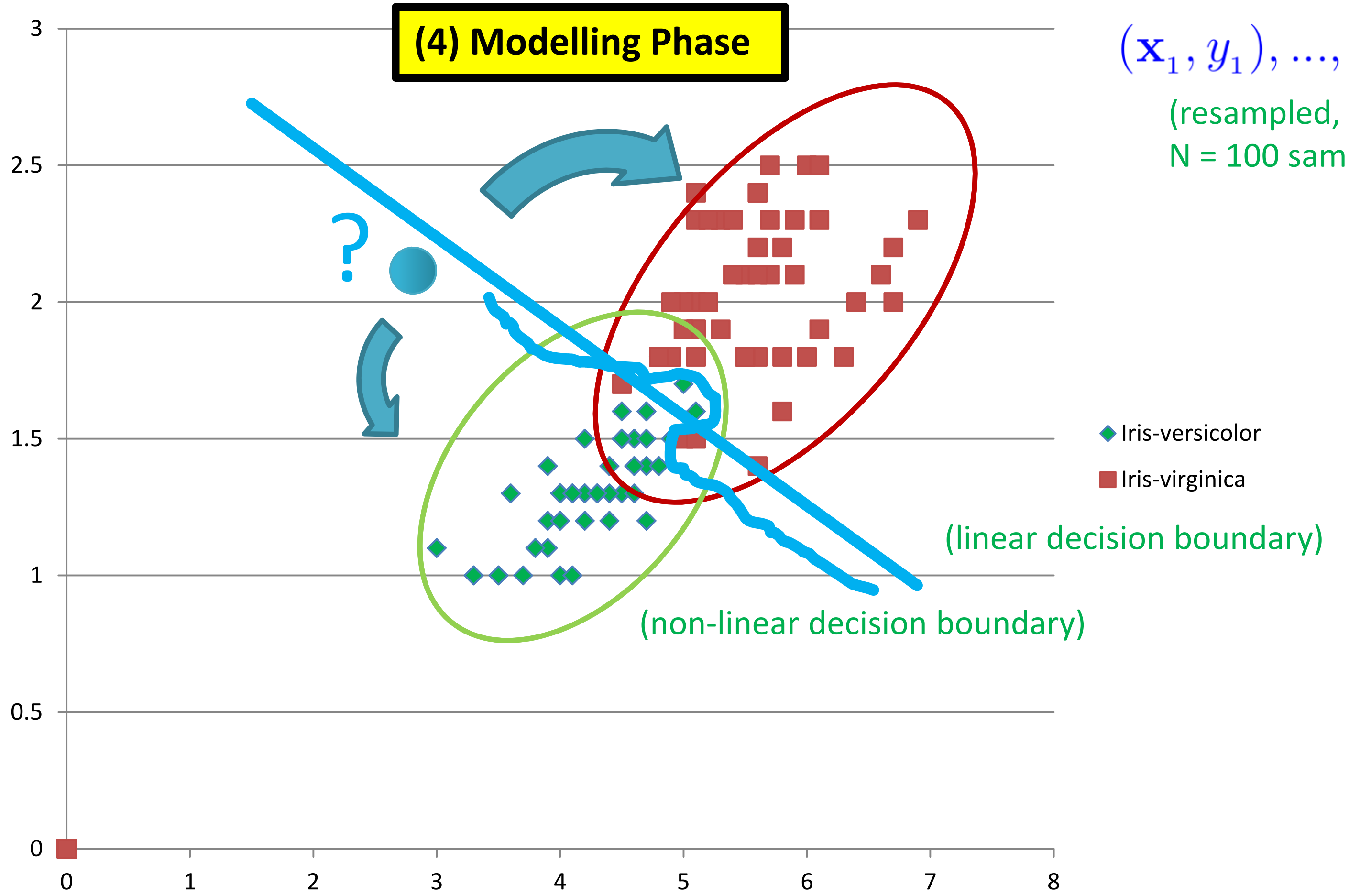
$$Accuracy = \frac{number\ of\ correct\ predictions}{total\ number\ of\ predictions}$$

- Error rate

$$Error\ rate = \frac{number\ of\ wrong\ predictions}{total\ number\ of\ predictions}$$

- If model evaluation is satisfactory: **(6) Deployment Phase**

Non-linearly Seperable Data in Practice – Which model?



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

(resampled, again
N = 100 samples)

(lessons learned from practice: requires soft-thresholds to allow for some errors being overall better for new data
→ Occams razor – 'simple model better')

(lessons learned from practice: requires non-linear decision boundaries)

Key Challenges: Why is it not so easy in practice?

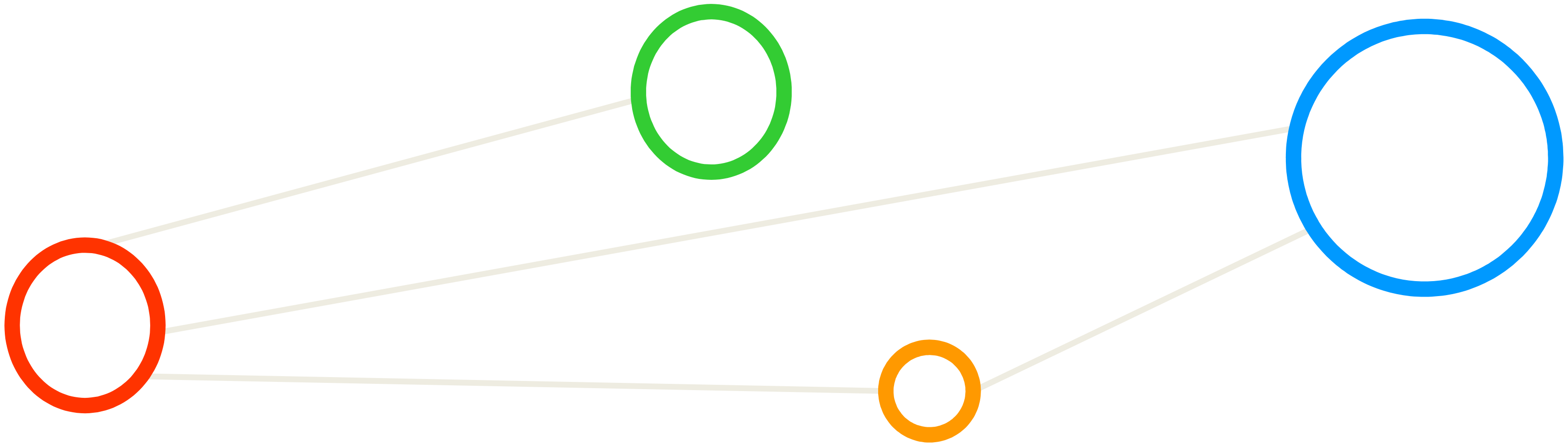
- Scalability
  - Gigabytes, Terabytes, and Petabytes datasets that fit not into memory
  - E.g. algorithms become necessary with out-of-core/CPU strategies
- High Dimensionality
  - Datasets with hundreds or thousand attributes become available
  - E.g. bioinformatics with gene expression data with thousand of features
- Heterogenous and Complex Data
  - More complex data objects emerge and unstructured data sets
  - E.g. Earth observation time-series data across the globe
- Data Ownership and Distribution
  - Distributed datasets are common (e.g. security and transfer challenges)

- **Key challenges faced when doing traditional data analysis and machine learning are scalability, high dimensionality of datasets, heterogenous and complex data, data ownership & distribution**
- **Combat ‘overfitting‘ is the key challenge in machine learning using validation & regularization**

Prevent Overfitting for better ‘ouf-of-sample‘ generalization

[15] Stop Overfitting, YouTube

HPC-Driven Data Analytics



Learning Approaches – What means Learning?

- **The basic meaning of learning is ‘to use a set of observations to uncover an underlying process‘**
- **The three different learning approaches are supervised, unsupervised, and reinforcement learning**

- Supervised Learning
  - Majority of methods follow this approach in this course
  - 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)

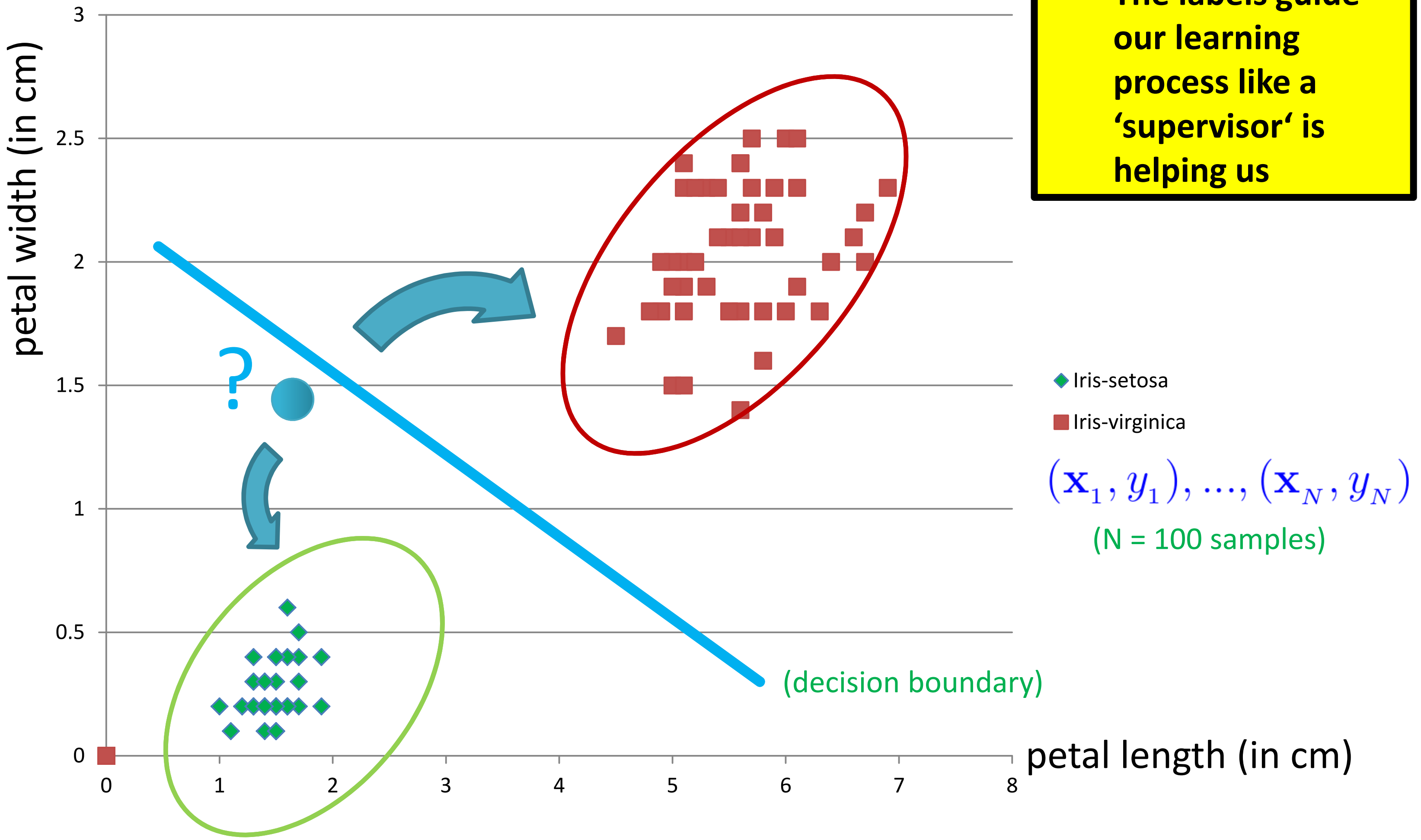
Learning Approaches – Supervised Learning

- Each observation of the predictor measurement(s) has an associated response measurement:
  - Input $\mathbf{x} = x_1, \dots, x_d$
  - Output $y_i, i = 1, .., n$
  - Data $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)$
- 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 fits a model that related the response to the predictors**
- **Supervised learning approaches are used in classification algorithms such as SVMs**
- **Supervised learning works with data = [input, correct output]**

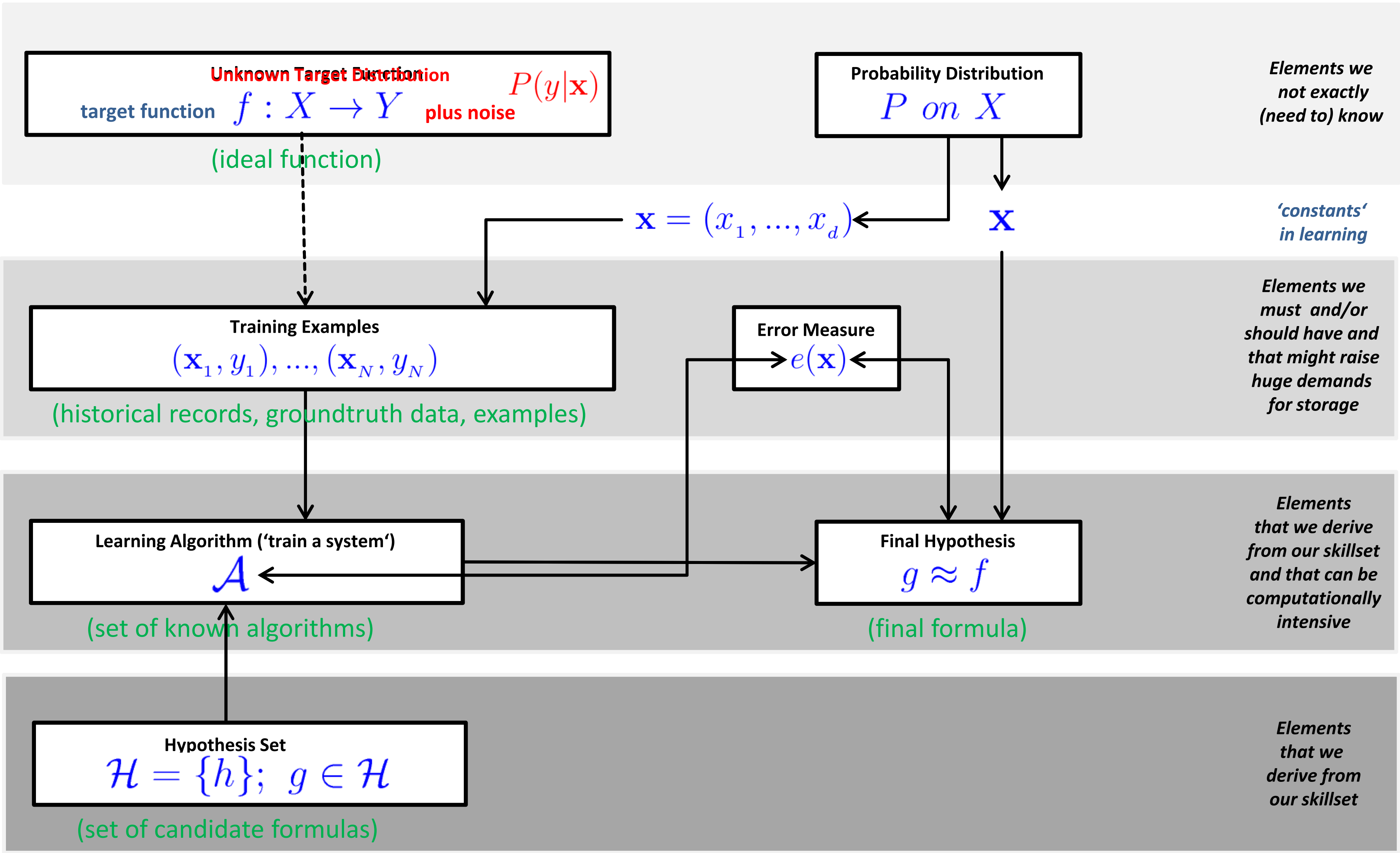
[13] An Introduction to Statistical Learning

Learning Approaches – Supervised Learning Example



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

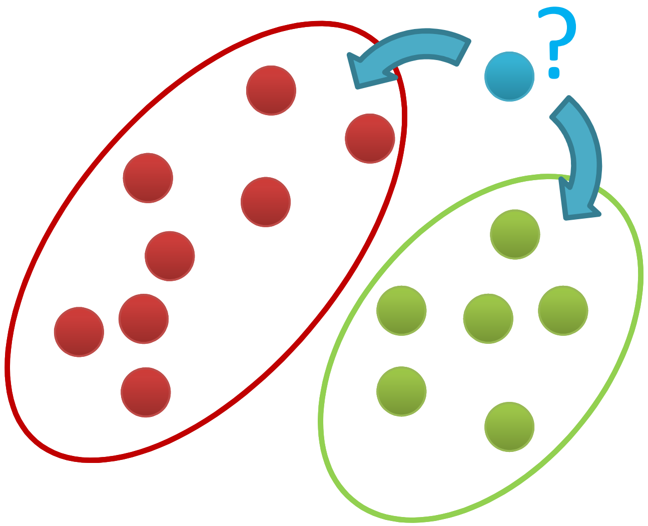
Supervised Learning – Overview & Summary



Methods Overview – Advanced Example

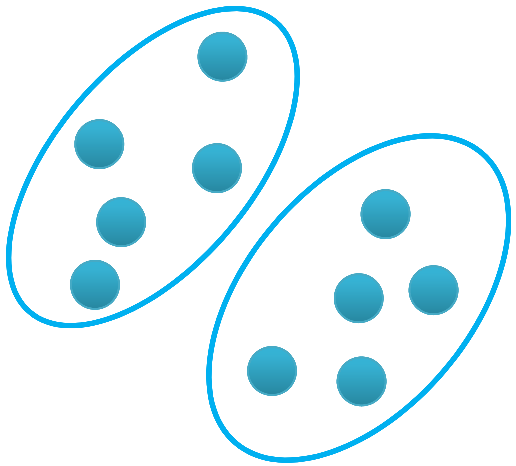
- **Statistical data mining methods can be roughly categorized in classification, clustering, or regression augmented with various techniques for data exploration, selection, or reduction**

Classification



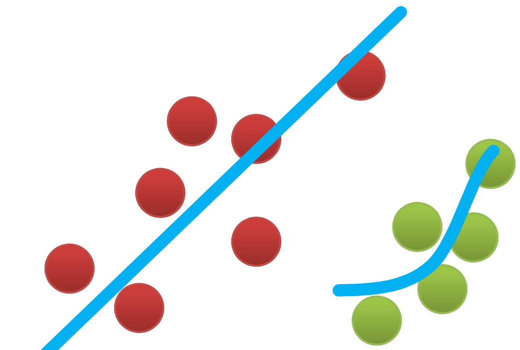
- Groups of data exist
- New data classified to existing groups

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 Refined

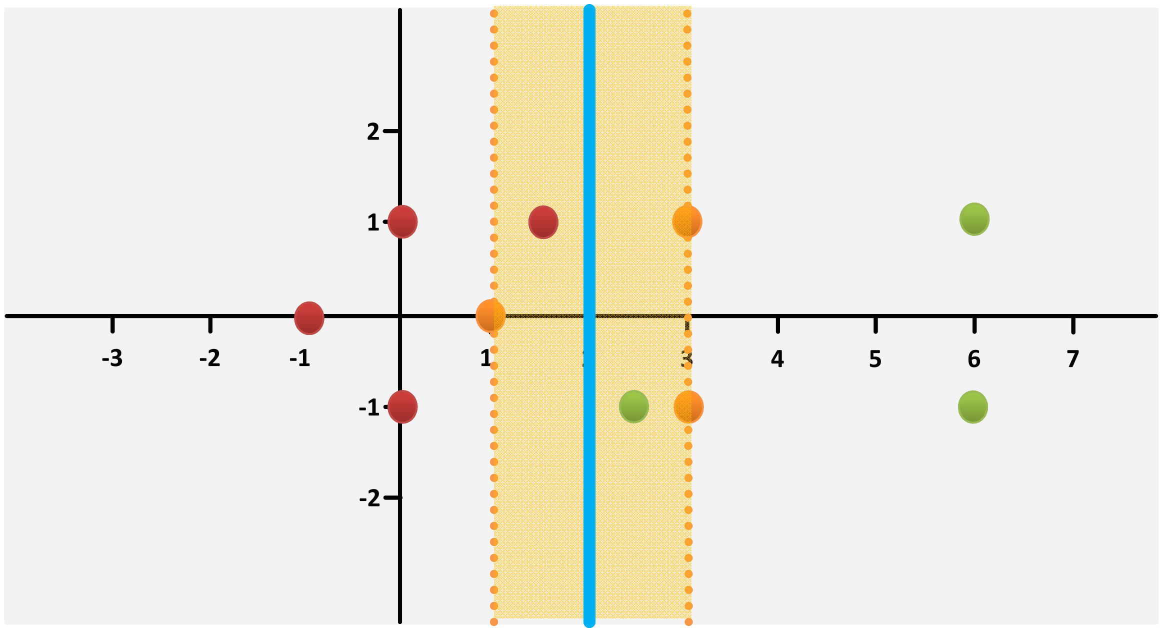
- **Support Vector Machines (SVMs) are a classification technique developed ~1990**
- **SVMs perform well in many settings & are considered as one of the best <u>‘out of the box classifiers‘</u>**

[13] An Introduction to Statistical Learning

- Term detailed refinement into ‘three separate techniques’
  - Practice: applications mostly use the SVMs with kernel methods
- ‘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;

Expected Out-of-Sample Performance for 'Best Line'

- The line with a 'bigger margin' seems to be better – but why?
  - Intuition: chance is higher that a new point will still be correctly classified
  - Fewer hypothesis possible: constrained by sized margin
  - Idea: achieving good 'out-of-sample' performance is goal



(e.g. better performance compared to PLA technique)

(simple line in a linear setup as intuitive decision boundary)

(Question remains: how we can achieve a bigger margin)

> Support Vector Machines (SVMs) are mathematically established in Appendix B of the slideset

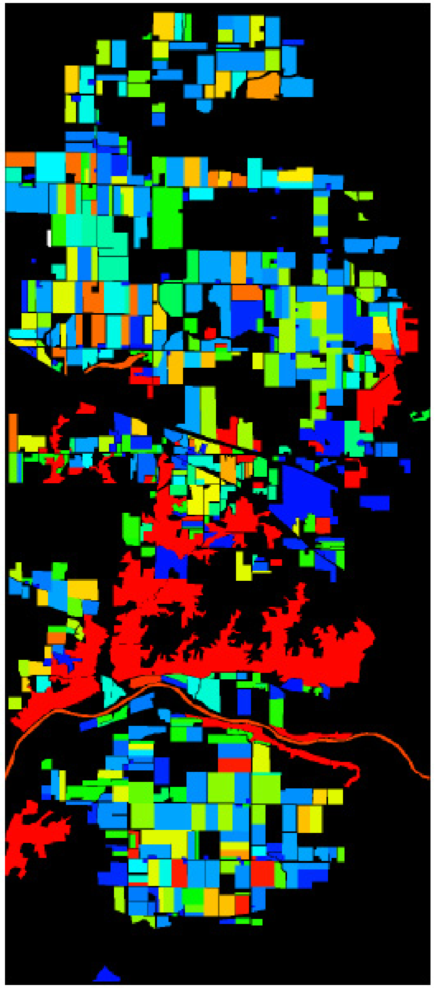
Indian Pines Dataset – Preprocessing

Corrected by JPL

- 1417×617 pixels ($\sim$600 MB)
- 200 bands (20 discarded, with low SNR)
- 58 classes (6 discarded, with $\leq$ 100 samples)

Class		Number of samples		Class		Number of samples	
number	name	training	test	number	name	training	test
1	Buildings	1720	15475	27	Pasture	1039	9347
2	Corn	1778	16005	28	pond	10	92
3	Corn?	16	142	29	Soybeans	939	8452
4	Corn-EW	51	463	30	Soybeans?	89	805
5	Corn-NS	236	2120	31	Soybeans-NS	111	999
6	Corn-CleanTill	1240	11164	32	Soybeans-CleanTill	507	4567
7	Corn-CleanTill-EW	2649	23837	33	Soybeans-CleanTill?	273	2453
8	Corn-CleanTill-NS	3968	35710	34	Soybeans-CleanTill-EW	1180	10622
9	Corn-CleanTill-NS-Irrigated	80	720	35	Soybeans-CleanTill-NS	1039	9348
10	Corn-CleanTilled-NS?	173	1555	36	Soybeans-CleanTill-Drilled	224	2018
11	Corn-MinTill	105	944	37	Soybeans-CleanTill-Weedy	54	489
12	Corn-MinTill-EW	563	5066	38	Soybeans-Drilled	1512	13606
13	Corn-MinTill-NS	886	7976	39	Soybeans-MinTill	267	2400
14	Corn-NoTill	438	3943	40	Soybeans-MinTill-EW	183	1649
15	Corn-NoTill-EW	121	1085	41	Soybeans-MinTill-Drilled	810	7288
16	Corn-NoTill-NS	569	5116	42	Soybeans-MinTill-NS	495	4458
17	Fescue	11	103	43	Soybeans-NoTill	216	1941
18	Grass	115	1032	44	Soybeans-NoTill-EW	253	2280
19	Grass/Trees	233	2098	45	Soybeans-NoTill-NS	93	836
20	Hay	113	1015	46	Soybeans-NoTill-Drilled	873	7858
21	Hay?	219	1966	47	Swampy Area	58	525
22	Hay-Alfalfa	226	2032	48	River	311	2799
23	Lake	22	202	49	Trees?	58	522
24	NotCropped	194	1746	50	Wheat	498	4481
25	Oats	174	1568	51	Woods	6356	57206
26	Oats?	34	301	52	Woods?	14	130

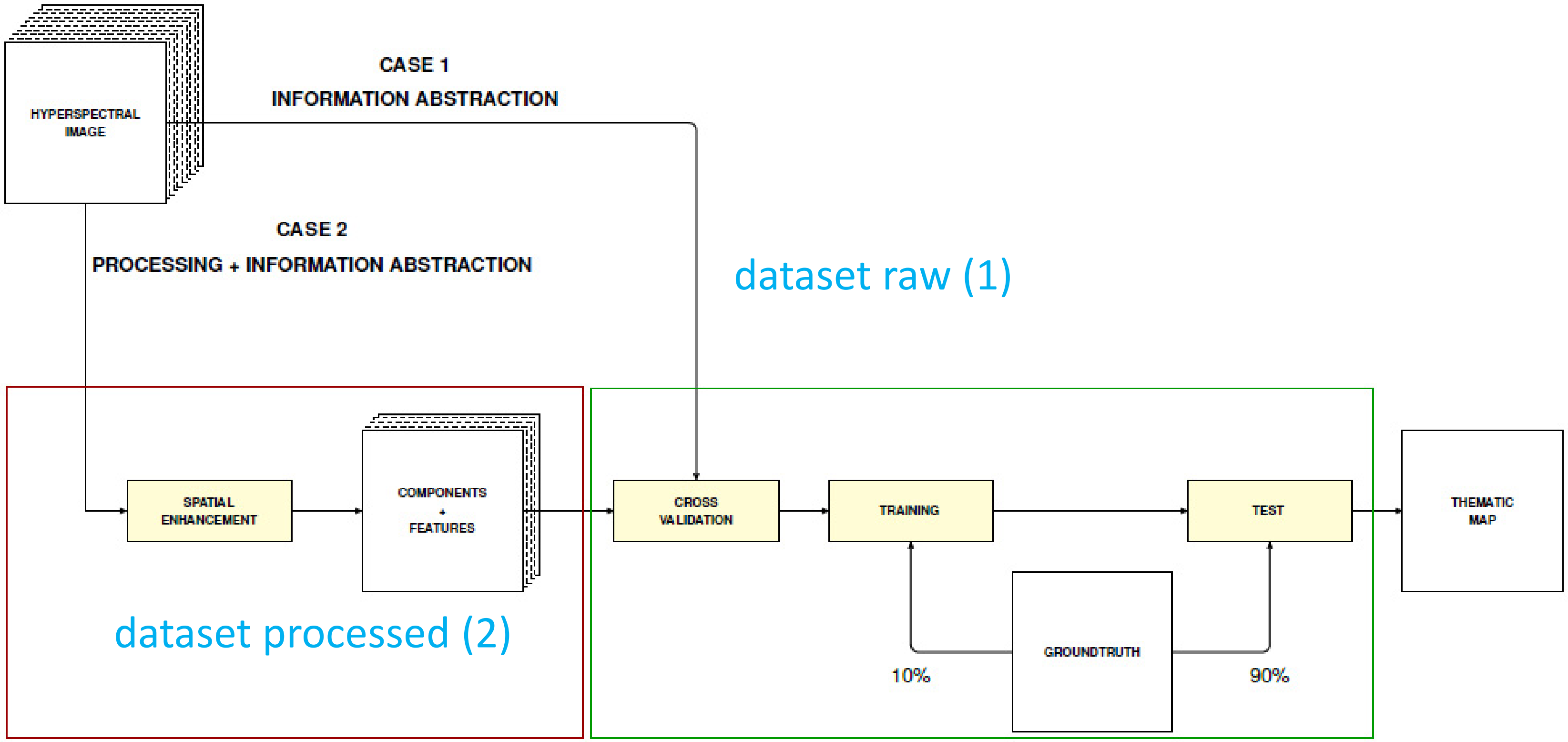
[16] G. Cavallaro and M. Riedel, et al. , 2015



(non-linearly separable) dataset

Indian Pines – Experimental Setup

Two Cases



Feature Enhancement & Selection

- Kernel Principle Component Analysis (KPCA)
- Extended Self-Dual Attribute Profile (ESDAP)
- Nonparametric weighted feature extraction (NWFE)

[16] G. Cavallaro and M. Riedel, et al. , 2015

Publicly Available Datasets – Open Data

- *Indian Pines Dataset Raw and Processed*

B2SHARE Store and Share Research Data

[17] 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		

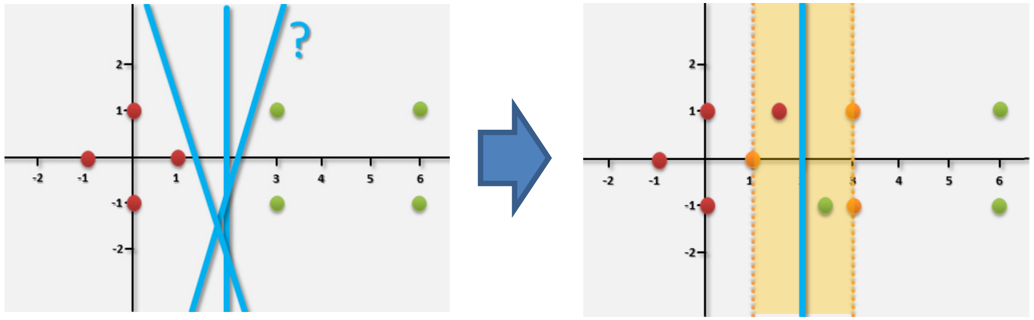
Review of Parallel SVM Implementations

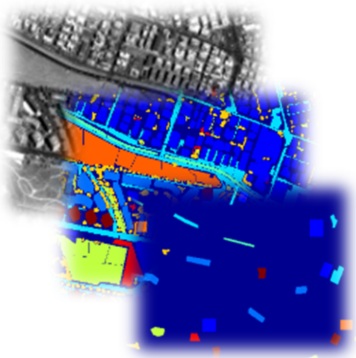
Technology	Platform Approach	Analysis
Apache Mahout	Java; Hadoop	No parallelization strategy for SVMs
Apache Spark/MLlib	Java; Spark	Parallel linear SVMs (no multi-class)
Twister/ParallelSVM	Java; Twister; Hadoop 1.0	Parallel SVMs, open source; developer version 0.9 beta
scikit-learn	Python	No parallelization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

[18] M. Goetz, M. Riedel et al., ‘On Parallel and Scalable Classification and Clustering Techniques for Earth Science Datasets’, *6th Workshop on Data Mining in Earth System Science, International Conference of Computational Science*

Parallel and Scalable Machine Learning – piSVM

- ‘Different kind‘ of parallel algorithms
  - Goal is to ‘learn from data‘ instead of modelling/approximate the reality
  - Parallel algorithms often useful to reduce ‘overall time for data analysis‘
- E.g. Parallel Support Vector Machines (SVMs) Technique
  - Data classification algorithm PiSVM using MPI to reduce ‘training time‘
  - Example: classification of land cover masses from satellite image data





Class	Training	Test
Buildings	18126	163129
Blocks	10982	98834
Roads	16353	147176
Light Train	1606	14454
Vegetation	6962	62655
Trees	9088	81792
Bare Soil	8127	73144
Soil	1506	13551
Tower	4792	43124
Total	77542	697859

πSVM

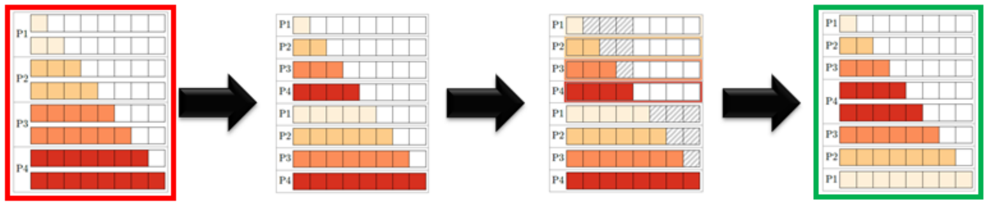
[16] G. Cavallaro & M. Riedel et al., ‘On Understanding Big Data Impacts in Remotely Sensed Image Classification Using Support Vector Machine Methods’, Journal of Applied Earth Observations and Remote Sensing

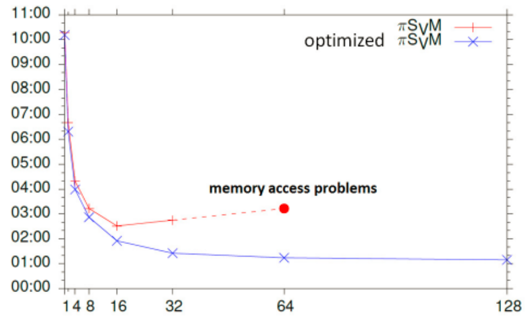
Parallel SVM with MPI Technique – piSVM Implementation

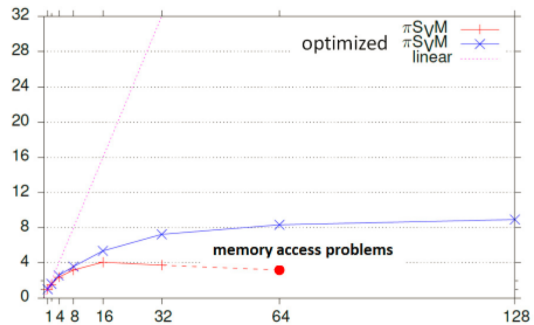
- Original piSVM 1.2 version (2011)
  - Open-source and based on libSVM library, C
  - Message Passing Interface (MPI)
  - New version appeared 2014-10 v. 1.3 (no major improvements)
  - Lack of 'big data' support (e.g. memory, layout)

[19] piSVM on SourceForge, 2008

- 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



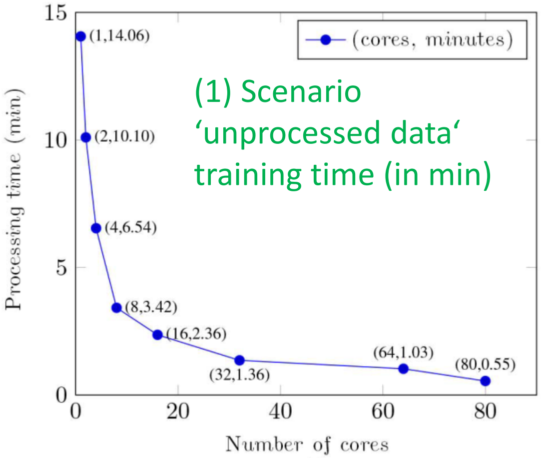




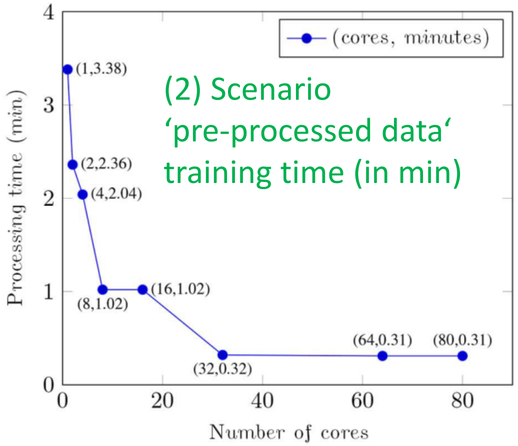
> ➢ **Link to Talk by Bernd Mohr – Performance analysis crucial → e.g. load balance & MPI collectives**

Parallelization Benefit: Lower-Time-To-Solution

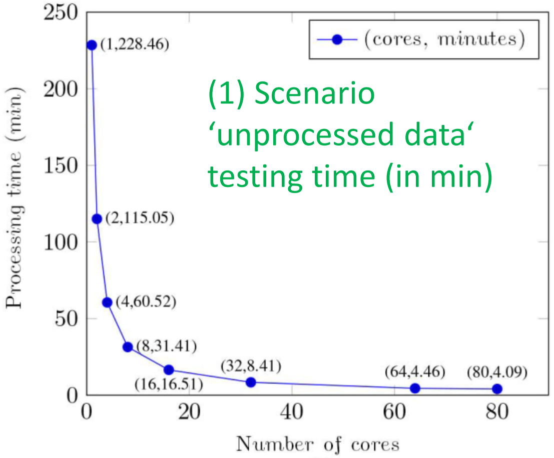
- Major speed-ups; ~interactive (<1 min); same accuracy;



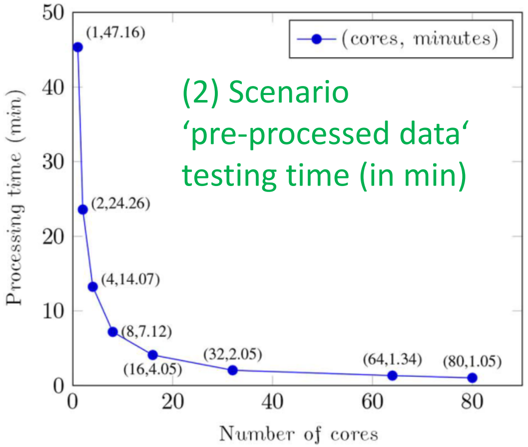
(a)



(a)



(b)



(b)

manual & serial activities (in min)

	kpca	esdap	nwfe	10x CSV	Training	Test	Total
(1) Scenario	0	0	0	4.47×10^3	10.45	71.08	4.55×10^3
(2) Scenario	5	15.38	1	529.55	1.37	23.25	575.55

'big data' is not always better data

	(1) Scenario	(2) Scenario
Number of features	200	30
Overall Accuracy (%)	40.68	77.96

(cf. Importance of feature engineering above)

[16] G. Cavallaro, M. Riedel, J.A. Benediktsson et al., Journal of Selected Topics in Applied Earth Observation and Remote Sensing, 2015

(aka first level of parallelism)

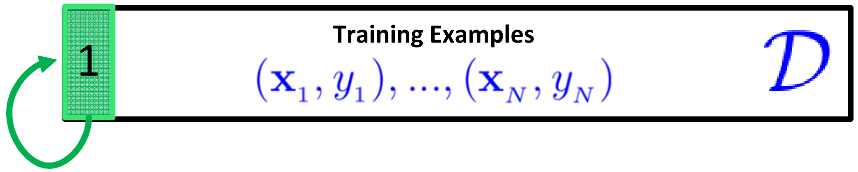
B2SHARE Store and Share Research Data

Validation Technique – Cross-Validation for Model Selection

- **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 (→ use parallelization)**

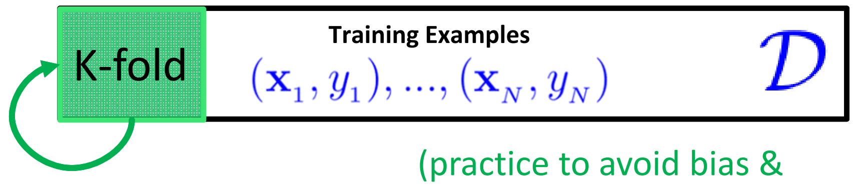
- Leave-one-out
  - N training sessions on N – 1 points each time

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

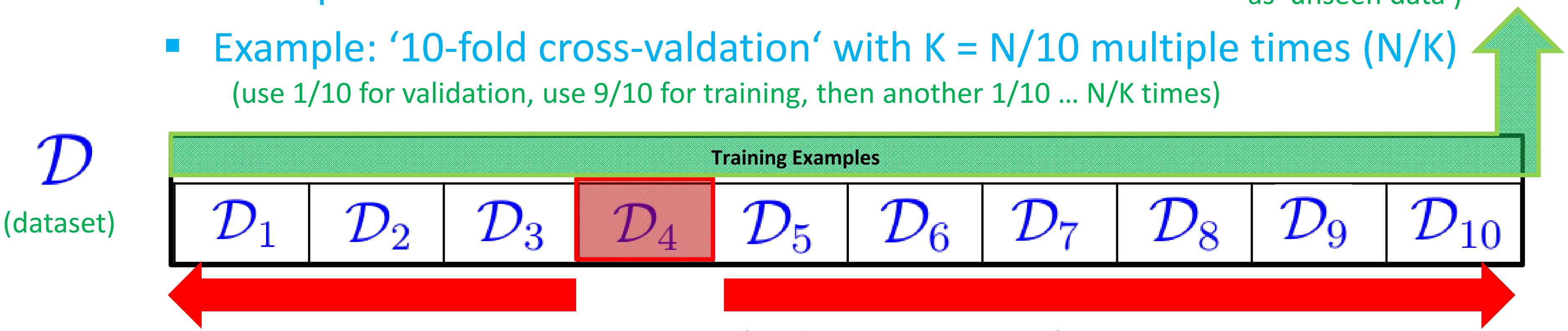


- Leave-more-out
  - Break data into number of folds
  - N/K training sessions on N – K points each time (fewer training sessions than above)
  - Example: '10-fold cross-valdation' with K = N/10 multiple times (N/K)
    (use 1/10 for validation, use 9/10 for training, then another 1/10 … N/K times)

(generalization to leave k points out at each run)



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



(involved in training now) (involved in training now)

(now is the current example run)

Parallelization Benefits using Cross-Validation & Parameters

- 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)

γ/C	1	10	100	1000	10 000
2	27.30 (109.78)	34.59 (124.46)	39.05 (107.85)	37.38 (116.29)	37.20 (121.51)
4	29.24 (98.18)	37.75 (85.31)	38.91 (113.87)	38.36 (119.12)	38.36 (118.98)
8	31.31 (109.95)	**39.68** (118.28)	39.06 (112.99)	39.06 (190.72)	39.06 (872.27)
16	33.37 (126.14)	39.46 (171.11)	39.19 (206.66)	39.19 (181.82)	39.19 (146.98)
32	34.61 (179.04)	38.37 (202.30)	38.37 (231.10)	38.37 (240.36)	38.37 (278.02)

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

γ/C	1	10	100	1000	10 000
2	48.90 (18.81)	65.01 (19.57)	73.21 (20.11)	75.55 (22.53)	74.42 (21.21)
4	57.53 (16.82)	70.74 (13.94)	75.94 (13.53)	76.04 (14.04)	74.06 (15.55)
8	64.18 (18.30)	74.45 (15.04)	**77.00** (14.41)	75.78 (14.65)	74.58 (14.92)
16	68.37 (23.21)	76.20 (21.88)	76.51 (20.69)	75.32 (19.60)	74.72 (19.66)
32	70.17 (34.45)	75.48 (34.76)	74.88 (34.05)	74.08 (34.03)	73.84 (38.78)

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

γ/C	1	10	100	1000	10 000
2	27.26 (3.38)	34.49 (3.35)	39.16 (5.35)	37.56 (11.46)	37.57 (13.02)
4	29.12 (3.34)	37.58 (3.38)	38.91 (6.02)	38.43 (7.47)	38.43 (7.47)
8	31.24 (3.38)	**39.77 (4.09)**	39.14 (5.45)	39.14 (5.42)	39.14 (5.43)
16	33.36 (4.09)	39.61 (4.56)	39.25 (5.06)	39.25 (5.27)	39.25 (5.10)
32	34.61 (5.13)	38.37 (5.30)	38.36 (5.43)	38.36 (5.49)	38.36 (5.28)

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)

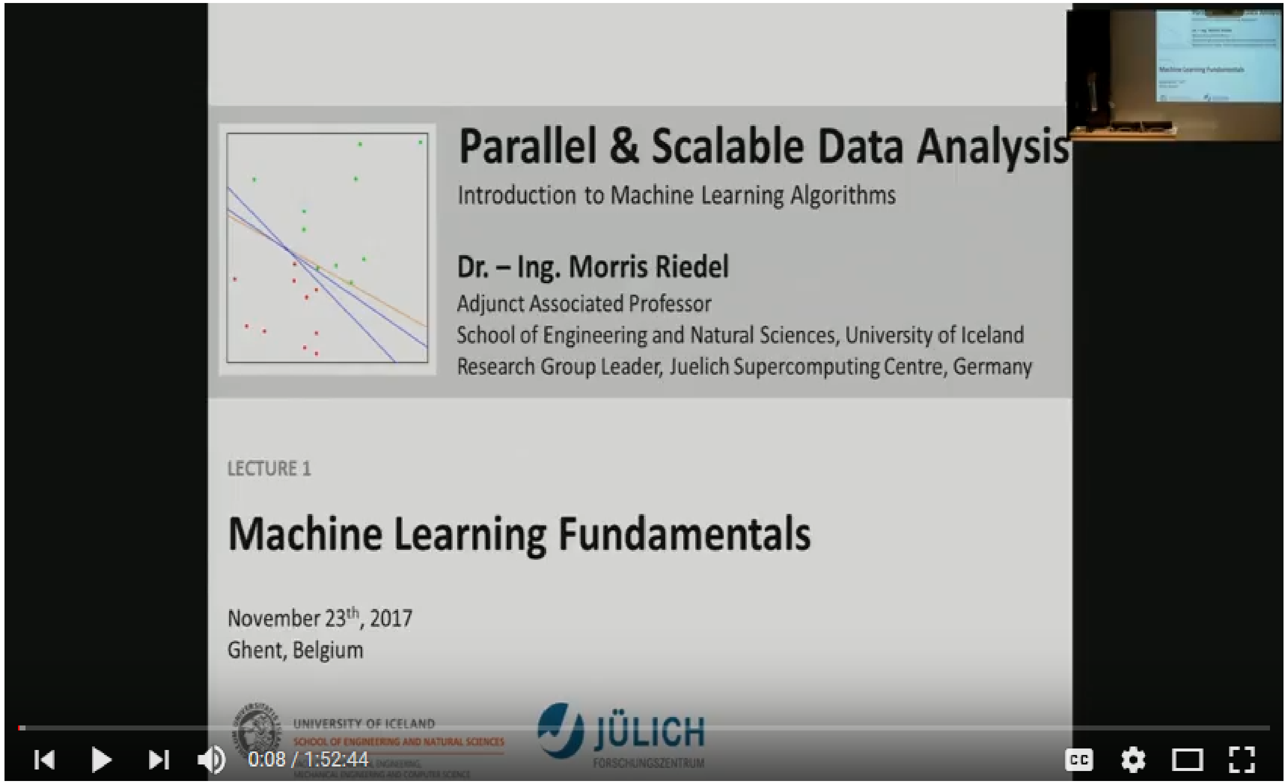
γ/C	1	10	100	1000	10 000
2	75.26 (1.02)	65.12 (1.03)	73.18 (1.33)	75.76 (2.35)	74.53 (4.40)
4	57.60 (1.03)	70.88 (1.02)	75.87 (1.03)	76.01 (1.33)	74.06 (2.35)
8	64.17 (1.02)	74.52 (1.03)	**77.02 (1.02)**	75.79 (1.04)	74.42 (1.34)
16	68.57 (1.33)	76.07 (1.33)	76.40 (1.34)	75.26 (1.05)	74.53 (1.34)
32	70.21 (1.33)	75.38 (1.34)	74.69 (1.34)	73.91 (1.47)	73.73 (1.33)

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

[16] G. Cavallaro, M. Riedel, J.A. Benediktsson et al., Journal of Selected Topics in Applied Earth Observation and Remote Sensing, 2015

B2SHARE Store and Share Research Data

[YouTube Lectures] More about parallel SVMs & HPC



[20] Morris Riedel, 'Introduction to Machine Learning Algorithms', Invited YouTube Lecture, six lectures, University of Ghent, 2017

Learning Approaches – Unsupervised Learning

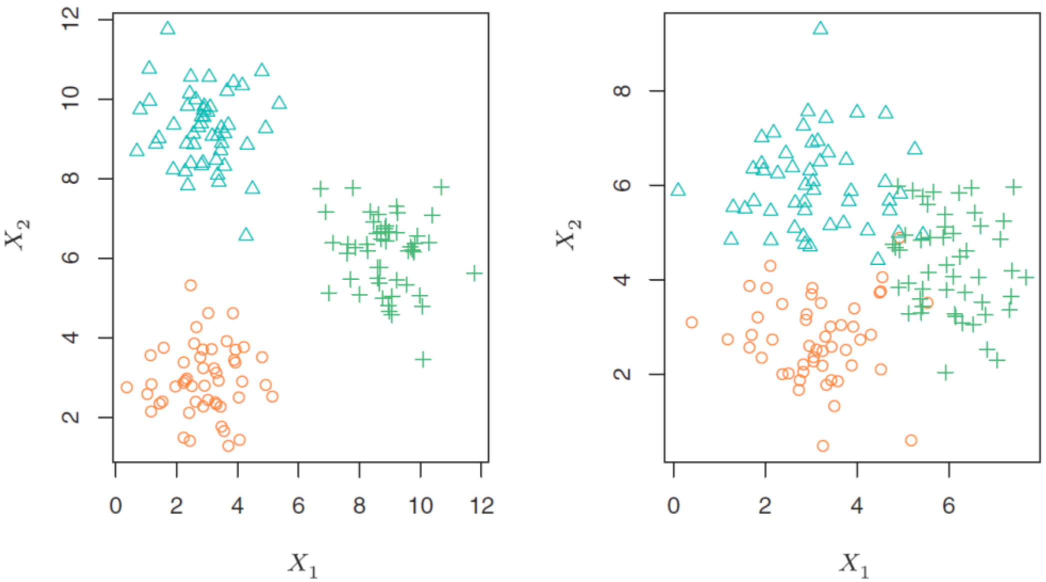
- Each observation of the predictor measurement(s) has no associated response measurement:
  - Input $\mathbf{x} = x_1, \ldots, x_d$
  - No output
  - Data $(\mathbf{x}_1), \ldots, (\mathbf{x}_N)$
- Goal: Seek to understand relationships between the observations
  - Clustering analysis: check whether the observations fall into distinct groups
- Challenges
  - No response/output that could supervise our data analysis
  - Clustering groups that overlap might be hardly recognized as distinct group

- **Unsupervised learning approaches seek to understand relationships between the observations**
- **Unsupervised learning approaches are used in clustering algorithms such as k-means, etc.**
- **Unupervised learning works with data = [input, ---]**

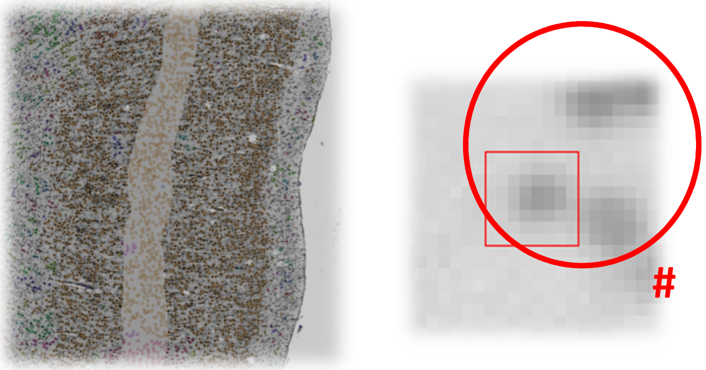
[13] An Introduction to Statistical Learning

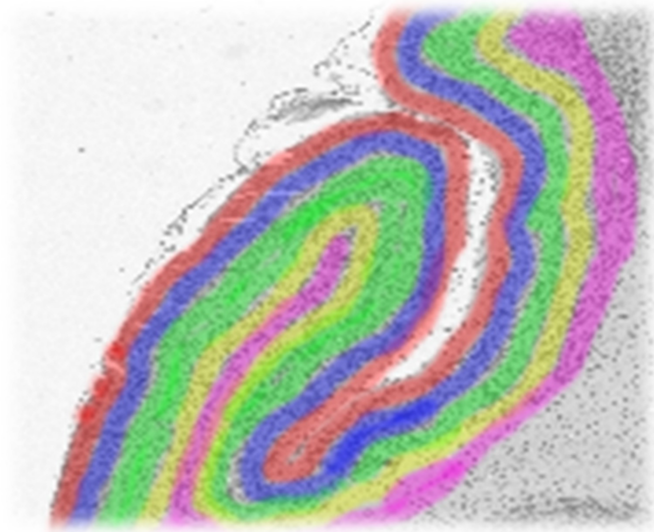
Learning Approaches – Unsupervised Learning Example

- Practice: The number of clusters can be ambiguities



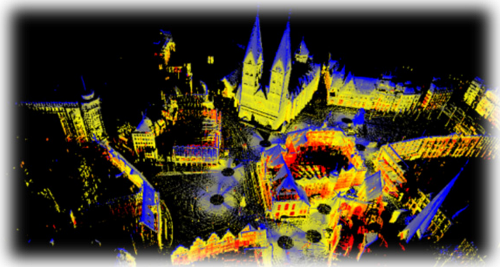
[13] An Introduction to Statistical Learning

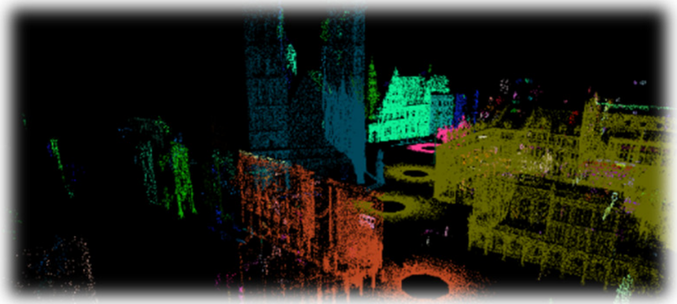


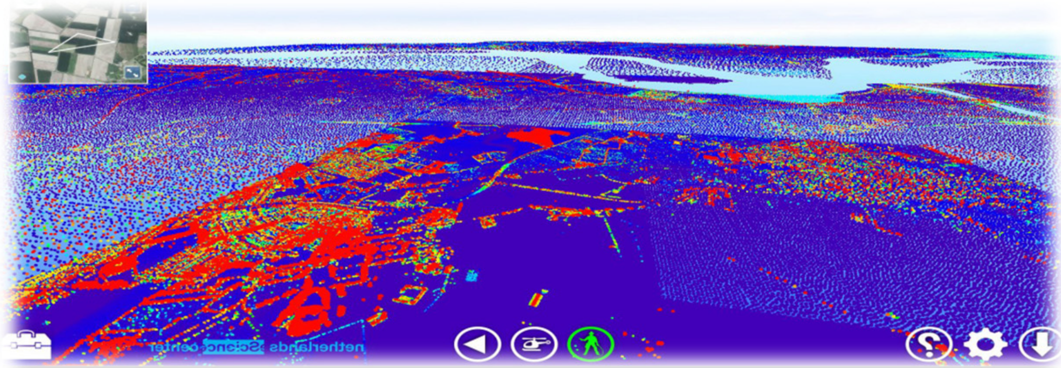


Point Cloud Applications

- '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)
  - Study level of continous details

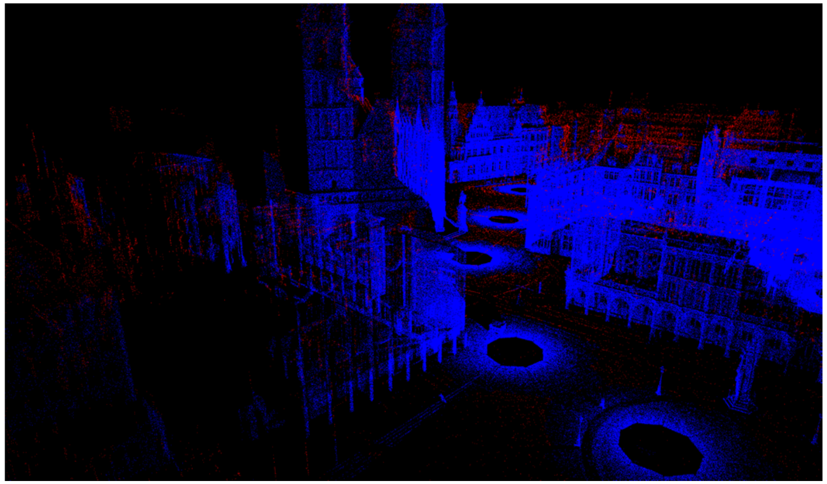


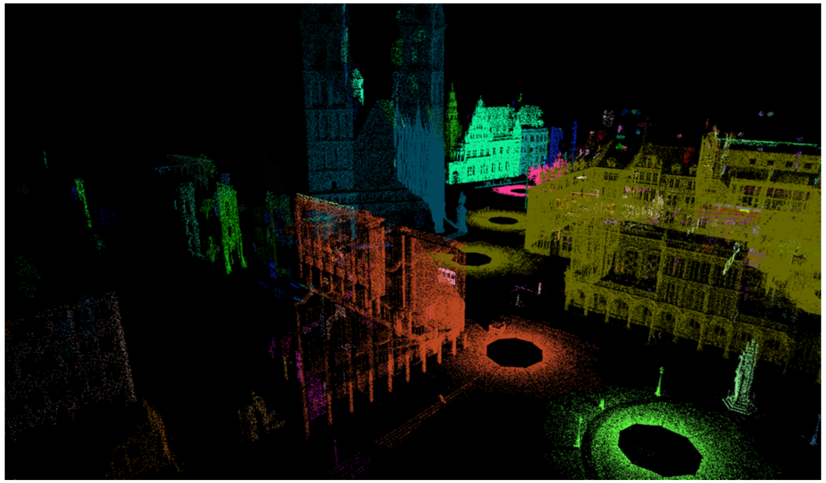




Bremen Dataset & Locations

- Different clusterings of the inner city of Bremen
  - Using smart visualizations of the point cloud library (PCL)





- **The Bremen Dataset is encoded in the HDF5 parallel file format**

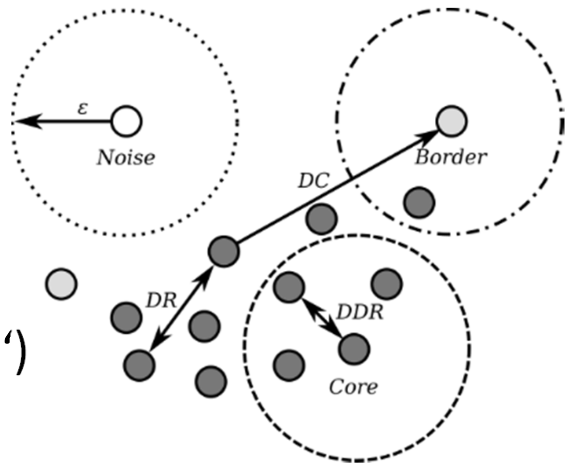
[22] Bremen Dataset

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)
  - Hiearchical 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)
  - Assumes clusters of similar density or areas of higher density in dataset

DBSCAN Algorithm

- DBSCAN Algorithm *[15] Ester et al.*
  - Introduced 1996 and most cited clustering algorithm
  - Groups number of similar points into clusters of data
  - Similarity is defined by a distance measure (e.g. *euclidean distance*)



(MinPoints = 4)

- Distinct Algorithm Features
  - Clusters a variable number of clusters
  - Forms arbitrarily shaped clusters (except 'bow ties')
  - Identifies inherently also outliers/noise

- Understanding Parameters
  - Looks for a similar points within a given search radius → Parameter *epsilon*
  - A cluster consist of a given minimum number of points → Parameter *minPoints*

(DR = Density Reachable)

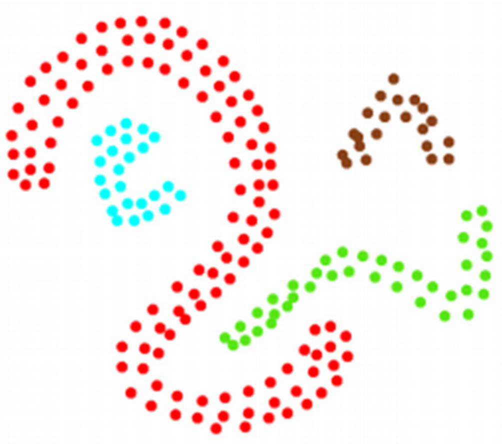
(DDR = Directly Density Reachable)

(DC = Density Connected)

DBSCAN Algorithm – Non-Trivial Example

- Compare K-Means vs. DBSCAN – How would K-Means work?

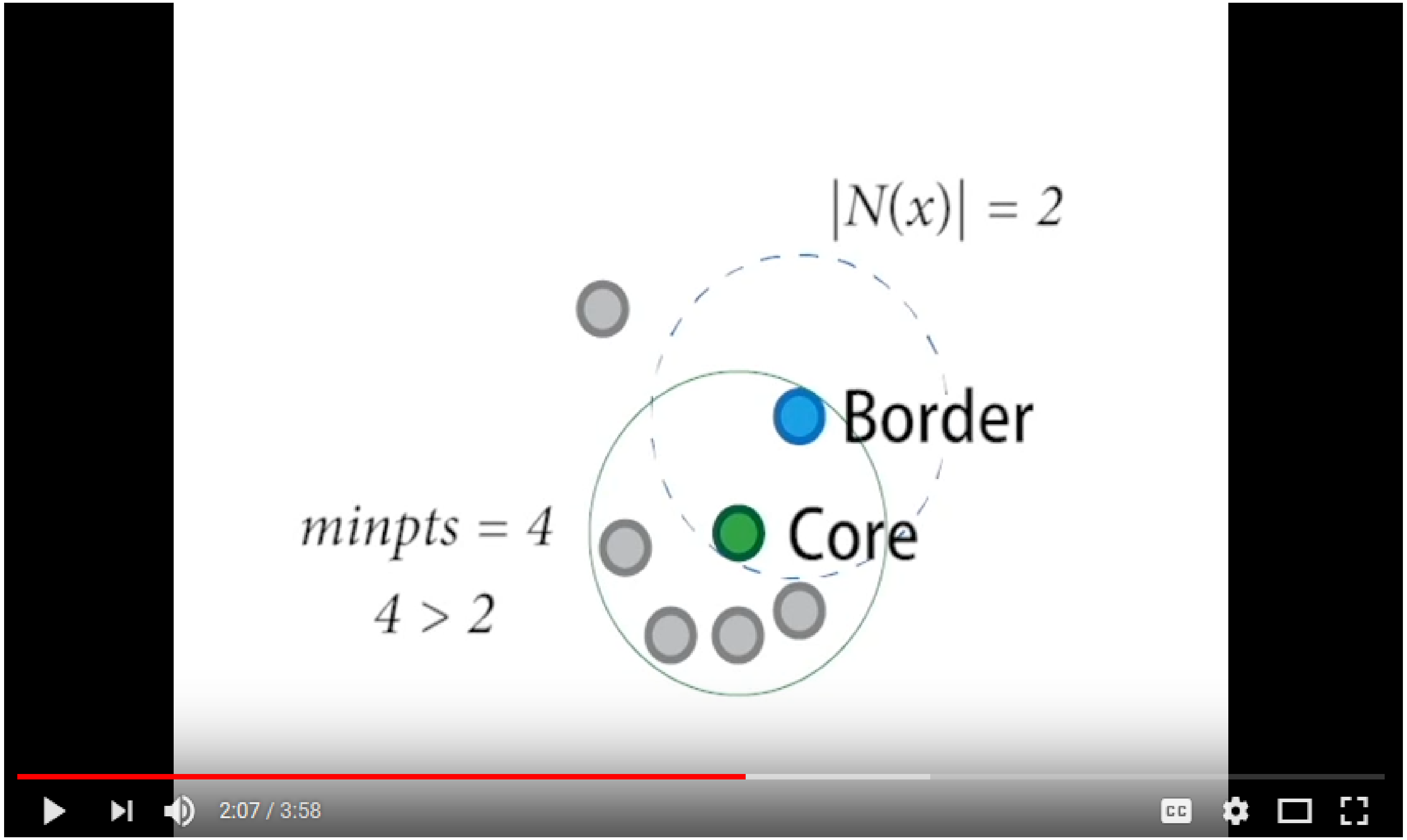
Unclustered
Data



Clustered
Data

- **DBSCAN forms arbitrarily shaped clusters (except 'bow ties') where other clustering algorithms fail**

[Video] DBSCAN Clustering



[6] DBSCAN, YouTube Video

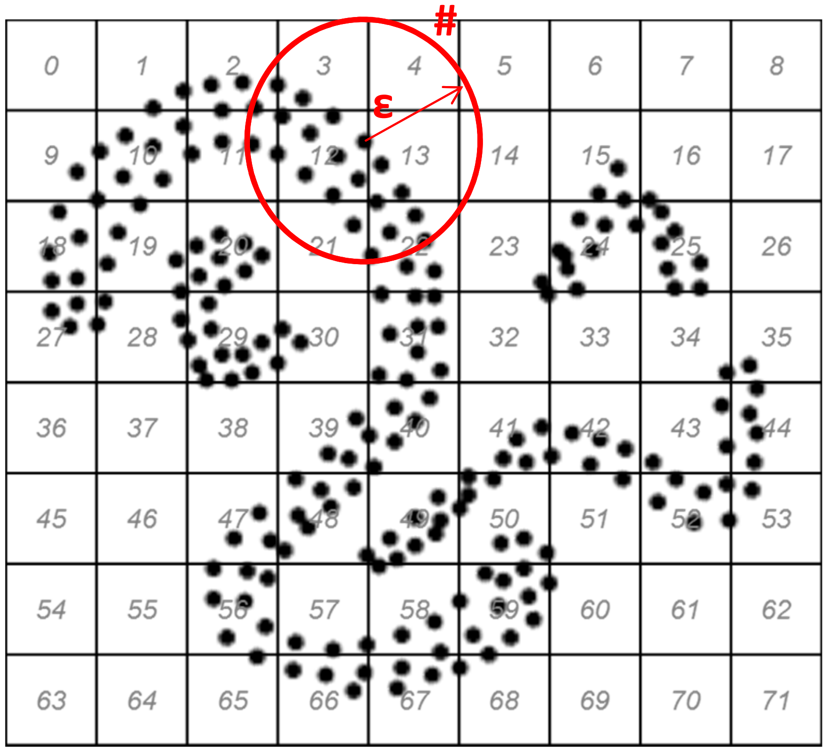
Review of Parallel DBSCAN Implementations

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

[18] M. Goetz, M. Riedel et al., **'On Parallel and Scalable Classification and Clustering Techniques for Earth Science Datasets',** ***6th Workshop on Data Mining in Earth System Science, International Conference of Computational Science (ICCS)***

HDBSCAN Algorithm Details

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

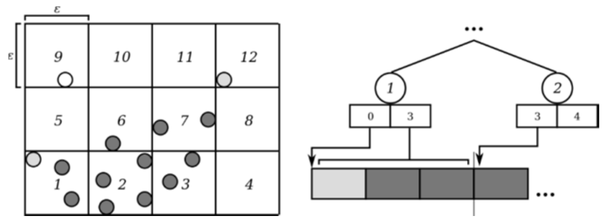


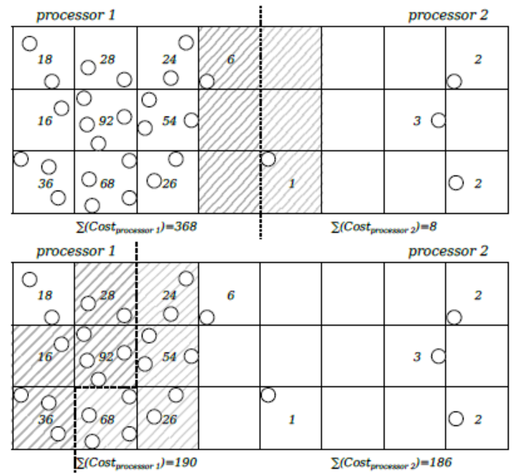
[24] M.Goetz, M. Riedel et al., 'HPDBSCAN – Highly Parallel DBSCAN', MLHPC Workshop at Supercomputing 2015

➢ **Link to Talk by Bernd Mohr – Distributing the work across processors: data domain decomposition**

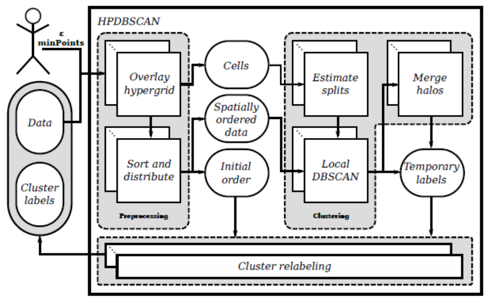
HPDBSCAN – Smart Domain Decomposition Example

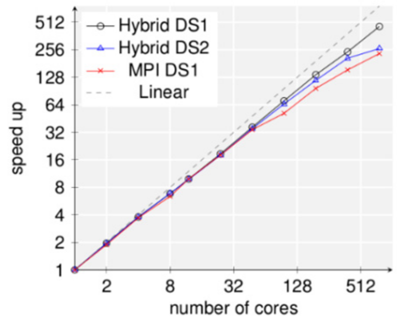
- Parallelization Strategy
  - Chunk data space equally
  - Overlay with hypergrid
  - Apply cost heuristic
  - Redistribute points (data locality)
  - Execute DBSCAN locally
  - Merge clusters at chunk edges
  - Restore initial order
- Data organization
  - Use of HDF5 (stores noise ID / cluster ID)





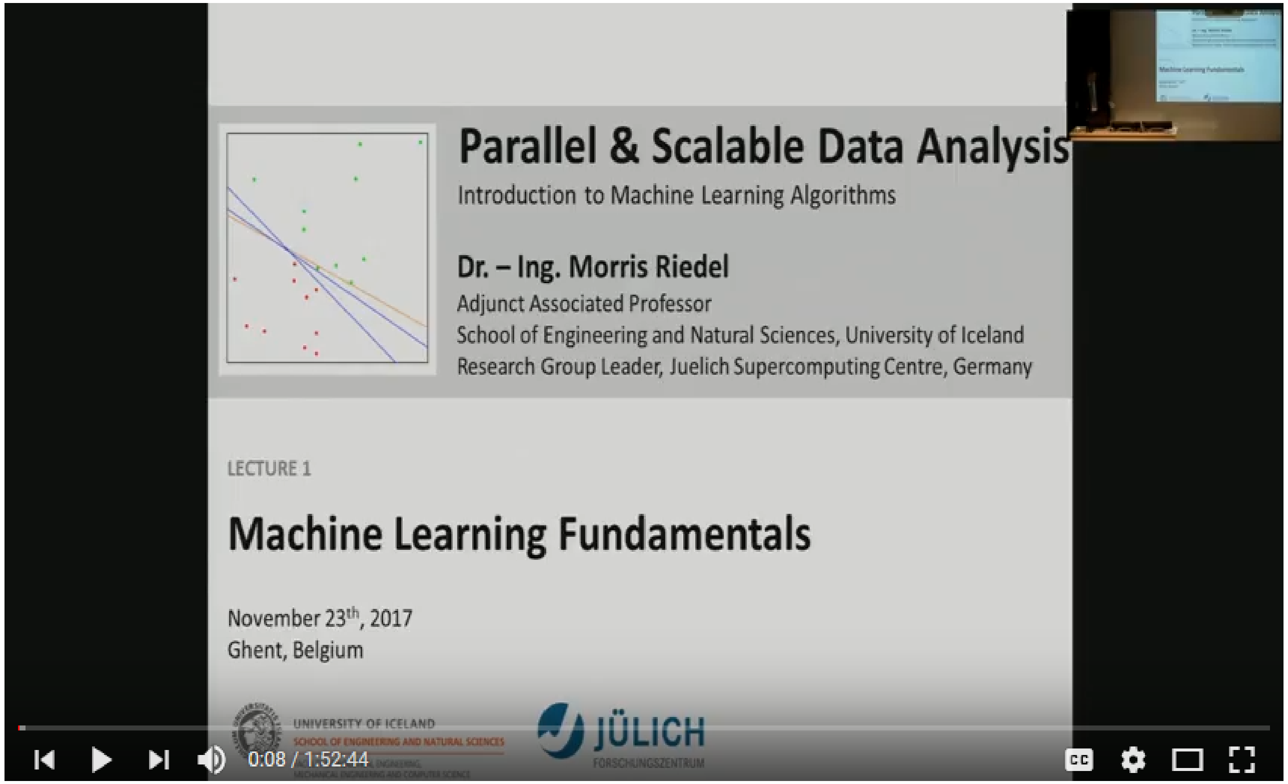
[24] M.Goetz, M. Riedel et al., 'HPDBSCAN – Highly Parallel DBSCAN', MLHPC Workshop at Supercomputing 2015





> ➢ **Link to Talk by Bernd Mohr – Evaluating parallel programs & code performance analysis is crucial**

[YouTube Lectures] More about parallel DBSCANs & HPC



[20] Morris Riedel, 'Introduction to Machine Learning Algorithms', Invited YouTube Lecture, six lectures, University of Ghent, 2017

Learning Approaches – Reinforcement Learning

- Each observation of the predictor measurement(s) has some associated response measurement:
  - Input $\mathbf{x} = x_1, \ldots, x_d$
  - Some output & grade of the output
  - Data $(\mathbf{x}_1), \ldots, (\mathbf{x}_N)$
- Goal: Learn through iterations
  - Guided by output grade: check learning and compare with grade
- Challenge:
  - Iterations may require lots of CPU time (e.g. backgammon playing rounds)
- (Rarely tackled in this course, just for the sake of completion)

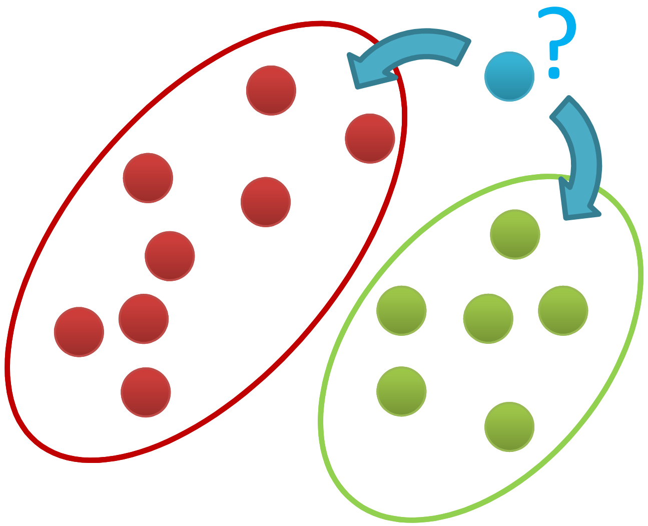
> - **Reinforcement learning approaches learn through iterations using the grading output as guide**
> - **Reinforcement learning approaches are used in playing game algorithms (e.g backgammon)**
> - **Unupervised learning works with data = [input, some output, grade for this output]**

[13] An Introduction to Statistical Learning

Methods Overview – Introduction to Deep Learning

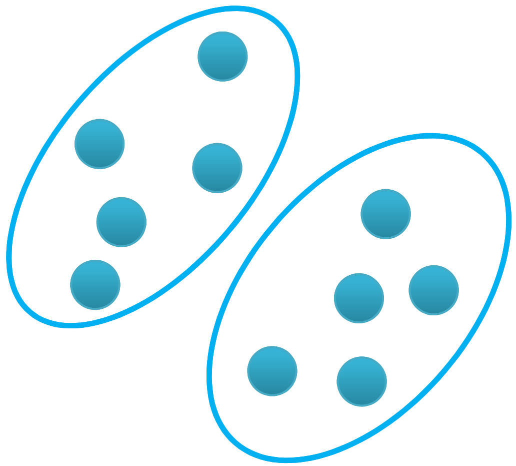
- **Statistical data mining methods can be roughly categorized in classification, clustering, or regression augmented with various techniques for data exploration, selection, or reduction**

Classification



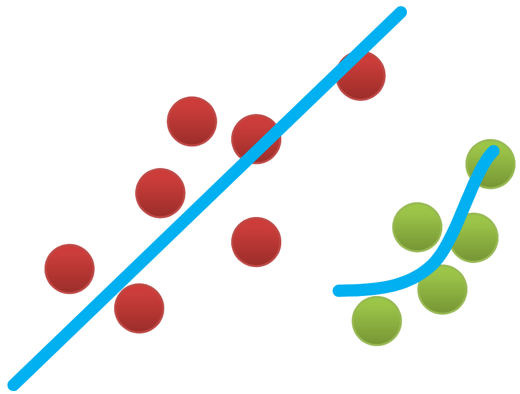
- Groups of data exist
- New data classified to existing groups

Clustering



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

Regression

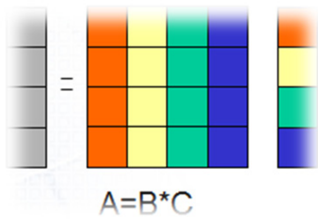


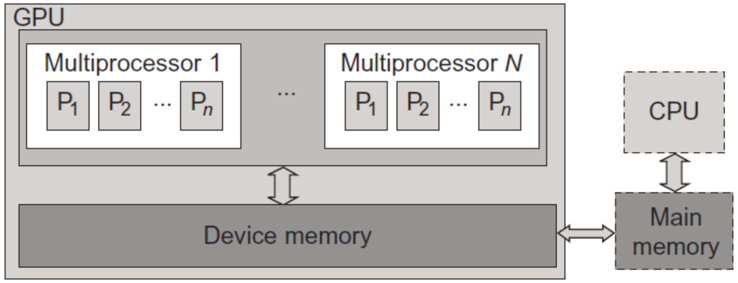
- Identify a line with a certain slope describing the data

More Recent HPC Developments: GPU Acceleration

- **CPU acceleration means that GPUs accelerate computing due to a massive parallelism with thousands of threads compared to only a few threads used by conventional CPUs**
- **GPUs are designed to compute large numbers of floating point operations in parallel**

- GPU accelerator architecture example (e.g. NVIDIA card)
  - GPUs can have 128 cores on one single GPU chip
  - Each core can work with eight threads of instructions
  - GPU is able to concurrently execute 128 * 8 = 1024 threads
  - Interaction and thus major (bandwidth) bottleneck between CPU and GPU is via memory interactions
  - E.g. applications that use matrix – vector multiplication



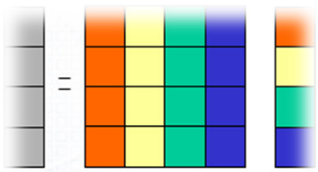


[29] Distributed & Cloud Computing Book

➢ **Link to Talk by Bernd Mohr – Accelerator architectures leveraging many-core and GPGPUs**

GPU Application Example – Matrix-Vector Multiplication

- What are the benefits of using GPUs in this application?



A=B*C

$$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} b_{0,0}c_0 + b_{0,1}c_1 + b_{0,2}c_2 + b_{0,3}c_3 \\ b_{1,0}c_0 + b_{1,1}c_1 + b_{1,2}c_2 + b_{1,3}c_3 \\ b_{2,0}c_0 + b_{2,1}c_1 + b_{2,2}c_2 + b_{2,3}c_3 \\ b_{3,0}c_0 + b_{3,1}c_1 + b_{3,2}c_2 + b_{3,3}c_3 \end{bmatrix}$$

P0 P1 P2 P3

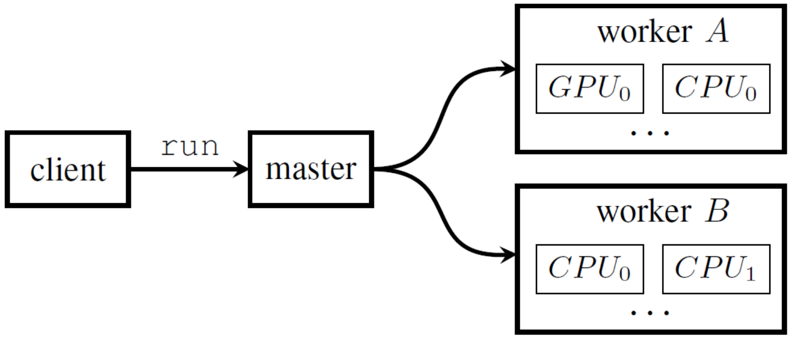
Keras with Tensorflow Backend – GPU Support

- **Keras is a high-level deep learning library implemented in Python that works on top of existing other rather low-level deep learning frameworks like Tensorflow, CNTK, or Theano**
- **The key idea behind the Keras tool is to enable faster experimentation with deep networks**
- **Created deep learning models run seamlessly on CPU and GPU via low-level frameworks**

[30] Keras Python Deep Learning Library

- **Tensorflow is an open source library for deep learning models using a flow graph approach**
- **Tensorflow nodes model mathematical operations and graph edges between the nodes are so-called tensors (also known as multi-dimensional arrays)**
- **The Tensorflow tool supports the use of CPUs and GPUs (much more faster than CPUs)**
- **Tensorflow work with the high-level deep learning tool Keras in order to create models fast**

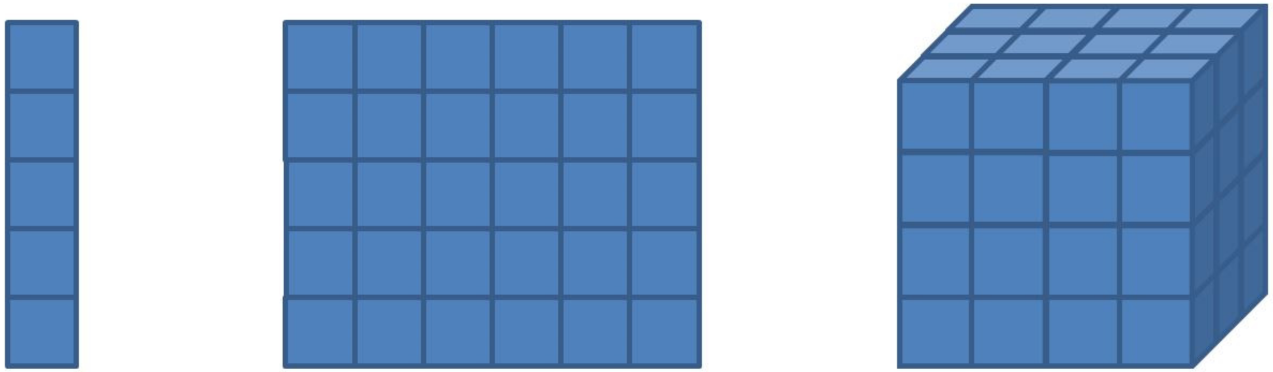
[31] Tensorflow Deep Learning Framework



[32] A Tour of Tensorflow

What is a Tensor?

- Meaning
  - Multi-dimensional array used in big data analysis often today
  - Best understood when comparing it with vectors or matrices



(one dimensional tensor) (vector of dimension [5])

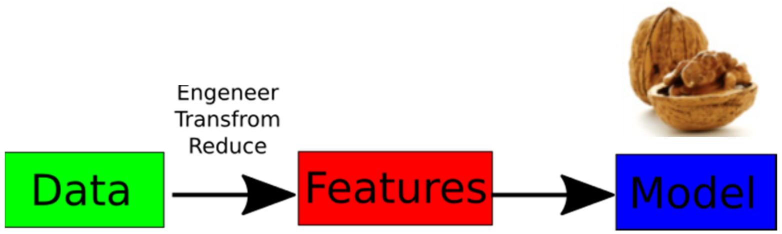
(two dimensional tensor) (matrix of dimensions [5,6])

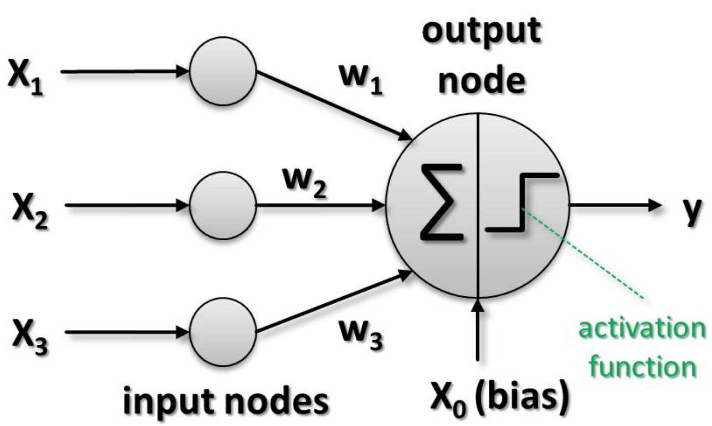
(three dimensional tensor) (tensor of dimension [4,4,3])

[33] Big Data Tips, What is a Tensor?

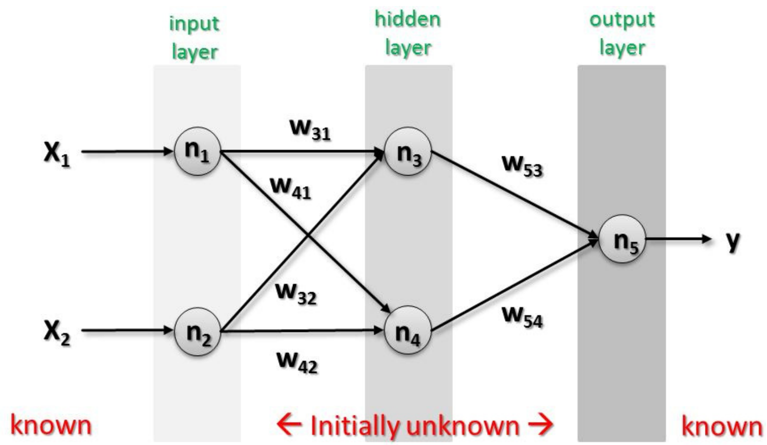
Artificial Neural Network – Feature Engineering & Layers

- Approach: Prepare data before
  - Classical Machine Learning
  - Feature engineering
  - Dimensionality reduction techniques
  - Low number of layers (many layers computationally infeasible in the past)
  - Very succesful for speech recognitition (‘state-of-the-art in your phone‘)



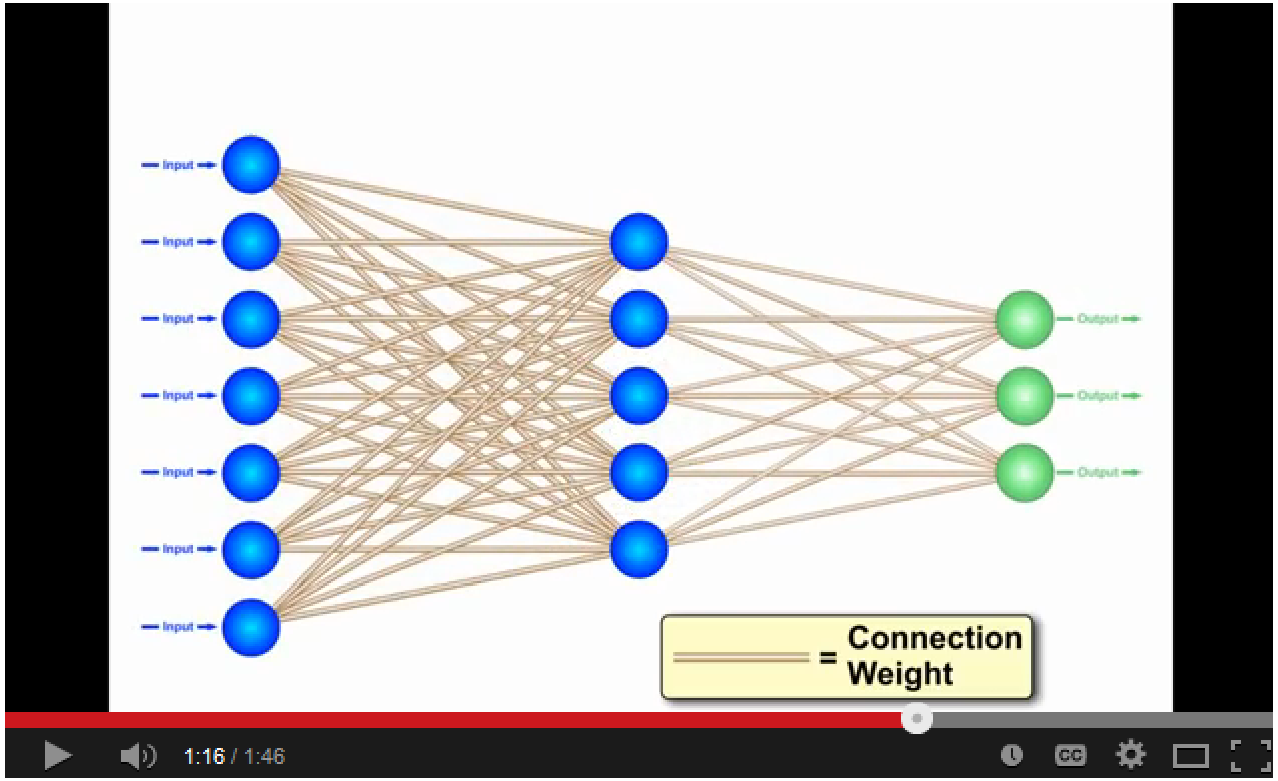


(Perceptron model: designed after human brain neuron)



(Artificial neural network two layer feed – forward)

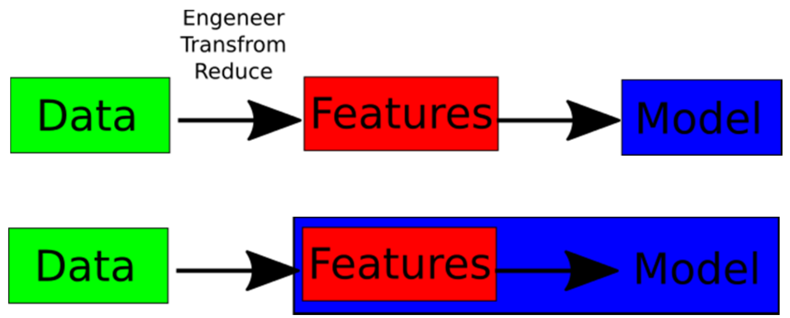
[Video] Towards Multi-Layer Perceptrons



[34] YouTube Video, Neural Networks – A Simple Explanation

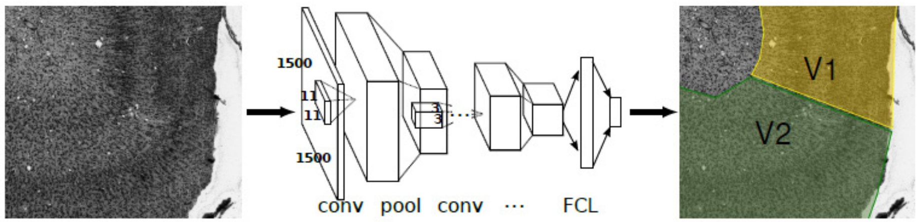
Deep Learning – Feature Learning & More Smart Layers

- Approach: Learn Features
  - Classical Machine Learning
  - (Powerful computing evolved)
  - Deep (Feature) Learning

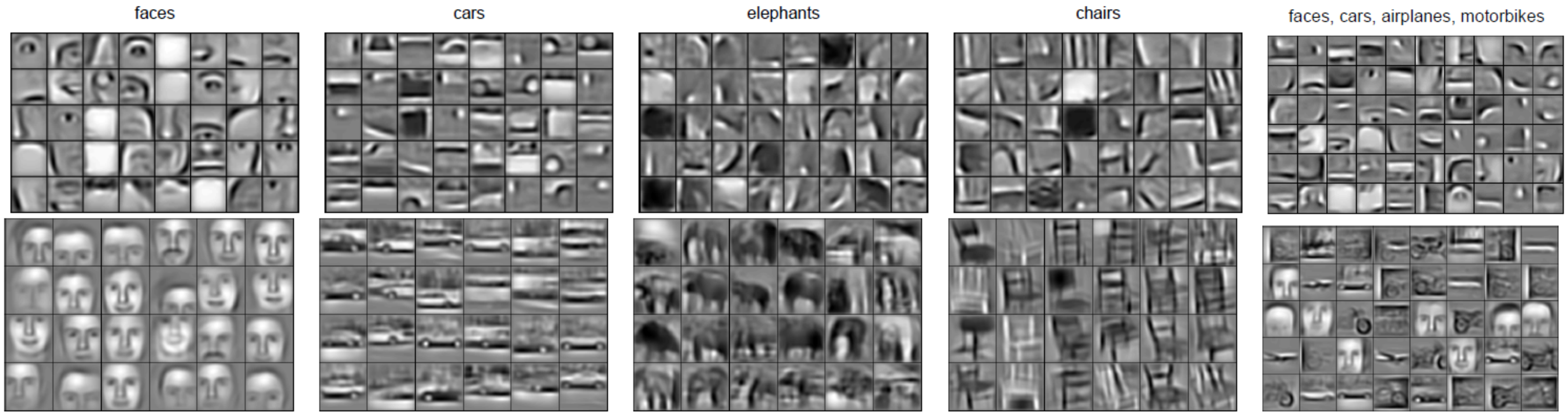


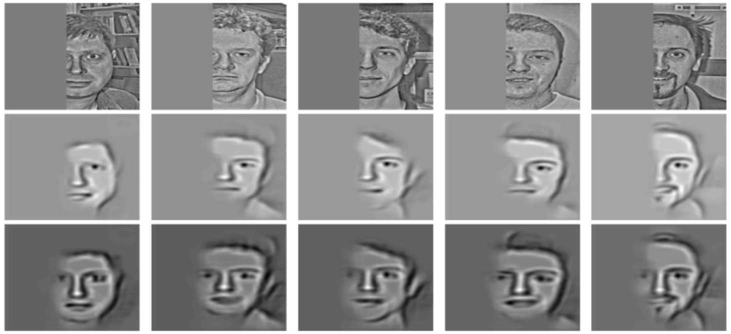
  - Very succesful for image recognition and other emerging areas
  - Assumption: data was generated by the interactions of many different factors on different levels (i.e. form a hierarchical representation)
  - Organize factors into multiple levels, corresponding to different levels of abstraction or composition(i.e. first layers do some kind of filtering)
  - Challenge: Different learning architectures: varying numbers of layers, layer sizes & types used to provide different amounts of abstraction

(Example: Parcellation of cytoarchitectonic cortical regions in the human brain)



Deep Learning – Feature Learning Benefits





- **Traditional machine learning applied feature engineering before modeling**
- **Feature engineering requires expert knowledge, is time-consuming and a often long manual process, requires often 90% of the time in applications, and is sometimes even problem-specific**
- **Deep Learning enables feature learning promising a massive time advancement**

[25] H. Lee et al., 'Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representations'

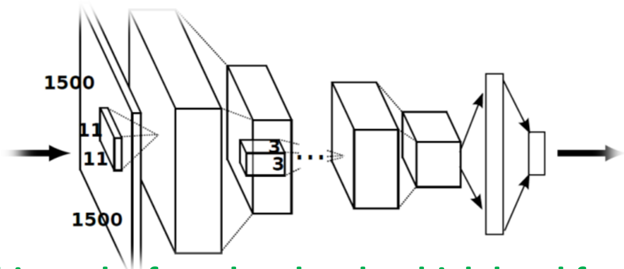
Deep Learning – Key Properties & Application Areas

- **In Deep Learning networks are many layers between the input and output layers enabling multiple processing layers that are composed of multiple linear and non-linear transformations**
- **Layers are not (all) made of neurons (but it helps to think about this analogy to understand them)**
- **Deep Learning performs (unsupervised) learning of multiple levels of features whereby higher level features are derived from lower level features and thus form a hierarchical representation**

- Application before modeling data with other models (e.g. SVM)
  - Create better data representations and create deep learning models to learn these data representations from large-scale unlabeled data
- Application areas
  - Computer vision
  - Automatic speech recognition
  - Natural language processing
  - Bioinformatics
  - …

(Deep Learning is often characterized as 'buzzword')

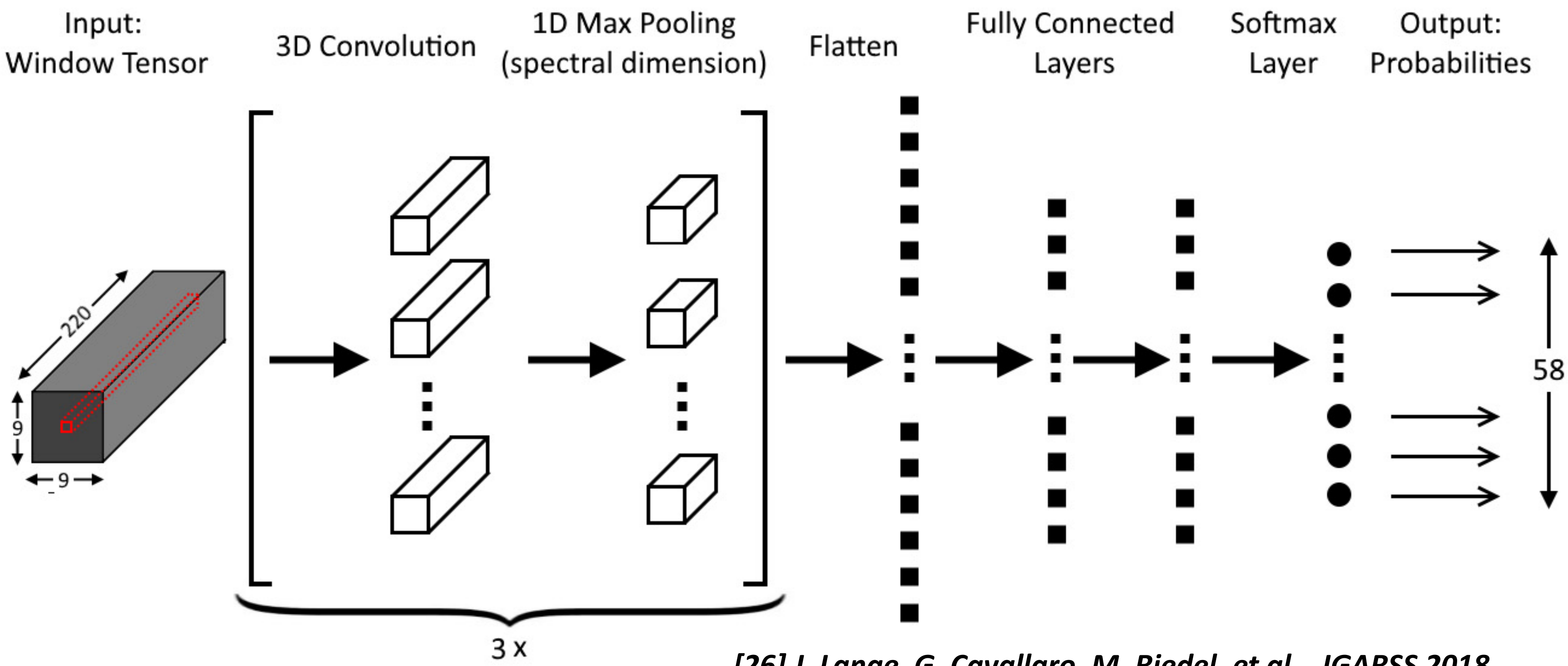
(Deep Learning is often 'just' called rebranding of traditional neural networks)



(hierarchy from low level to high level features)

CNN Architecture for Application – Land Cover Classification

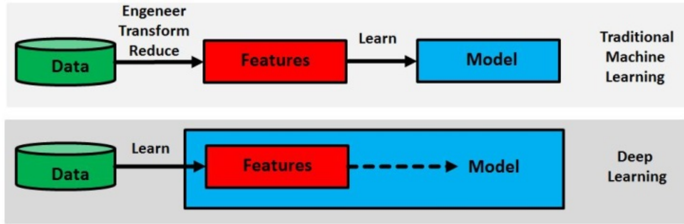
- Classify pixels in a hyperspectral remote sensing image having groundtruth/labels available
- Created CNN architecture for a specific hyperspectral land cover type classification problem
- Used dataset of Indian Pines (compared to other approaches) using all labelled pixels/classes
- Performed no manual feature engineering to obtain good results (aka accuracy)

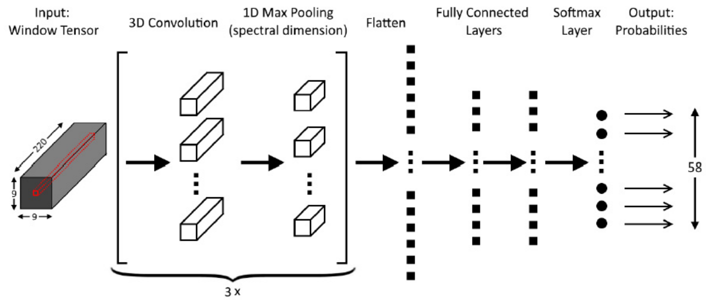


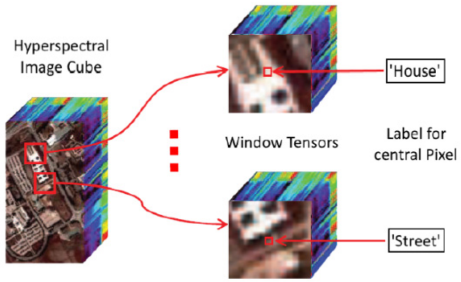
[26] J. Lange, G. Cavallaro, M. Riedel, et al. , IGARSS 2018

Comparison: Traditional Machine Learning vs. Deep Learning

- Traditional Methods
  - C MPI-based Support Vector Machine (SVM)
  - Substantial manual feature engineering
  - 10-fold cross-validation for model selection
  - Achieved 77,02 % accuracy (subsambled classes of 52 classes)
- Convolutional Neural Networks (CNNs)
  - Python/TensorFlow/Keras
  - Automated feature learning
  - Achieved 84,40 % accuracy on all 58 classes



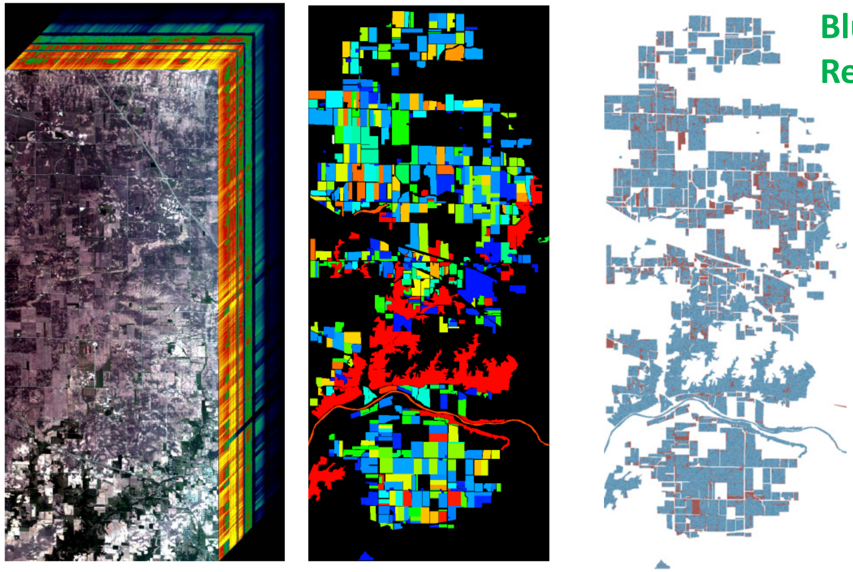




[26] J. Lange, G. Cavallaro, M. Riedel, et al. , 2018

> ➢ **More background information about CNN and its key elements are provided in Appendix C**

Number of Parameters – Challenges on the Horizon



Blue: correctly classified
Red: incorrectly classified

[26] J. Lange, G. Cavallaro, M. Riedel, et al. , IGARSS 2018

Feature	Representation / Value
Conv. Layer Filters	48, 32, 32
Conv. Layer Filter size	(3,3,5), (3,3,5), (3,3,5)
Dense Layer Neurons	128, 128
Optimizer	SGD
Loss Function	mean squared error
Activation Functions	ReLU
Training Epochs	600
Batch Size	50
Learning Rate	1
Learning Rate Decay	5×10^{-6}

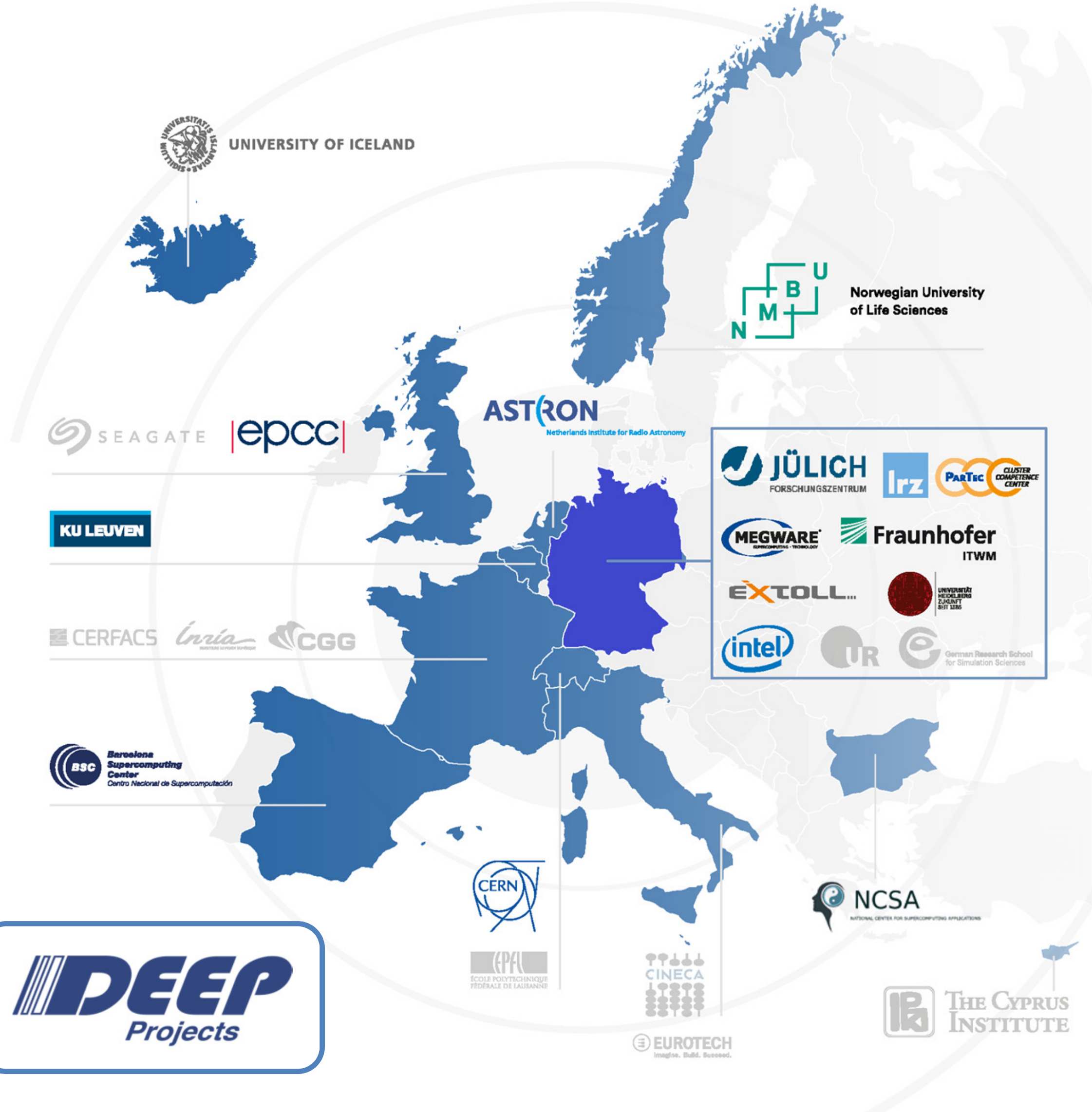
- **Using Python with TensorFlow & Keras easily enables changes in hyper-parameter tuning**
- **Various runs on different topologies add up to computational demand of GPUs**
- **Need for HPC machines with good GPUs and good deep learning software stacks required**
- **Key challenge remains in the number of parameters for deep learning networks & configuration**

➢ **Link to ISC 2018 Machine Learning Track Keynote by Frank Hutter about hyper-parameter problems**

DEEP Projects & Partners

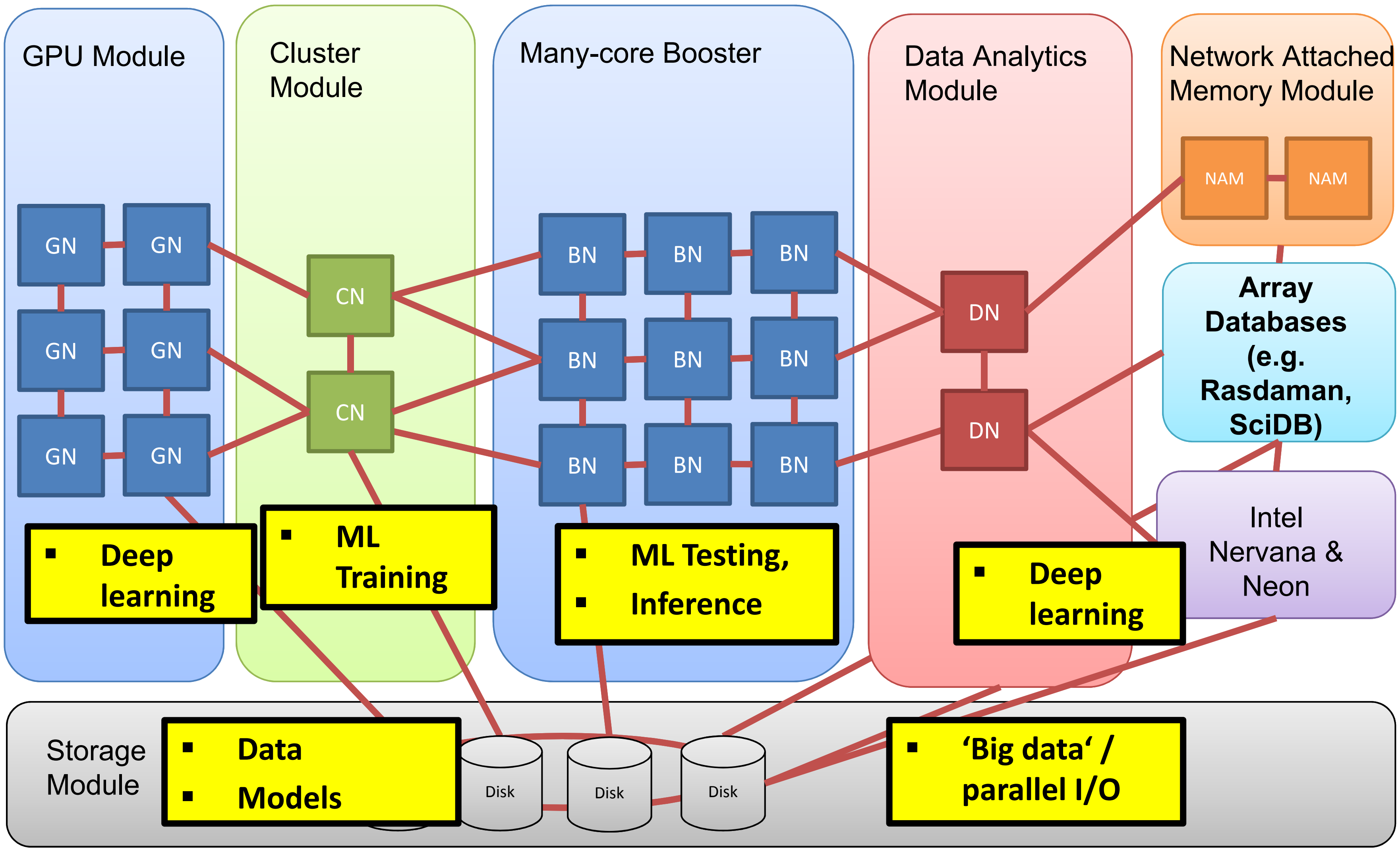
- DEEP
  - Dynamic Exascale Entry Platform
- 3 EU Exascale projects
  DEEP
  DEEP-ER
  DEEP-EST
- 27 partners
  Coordinated by JSC
- EU-funding: 30 M€
  JSC-part > 5,3 M€
- Nov 2011 – Jun 2020

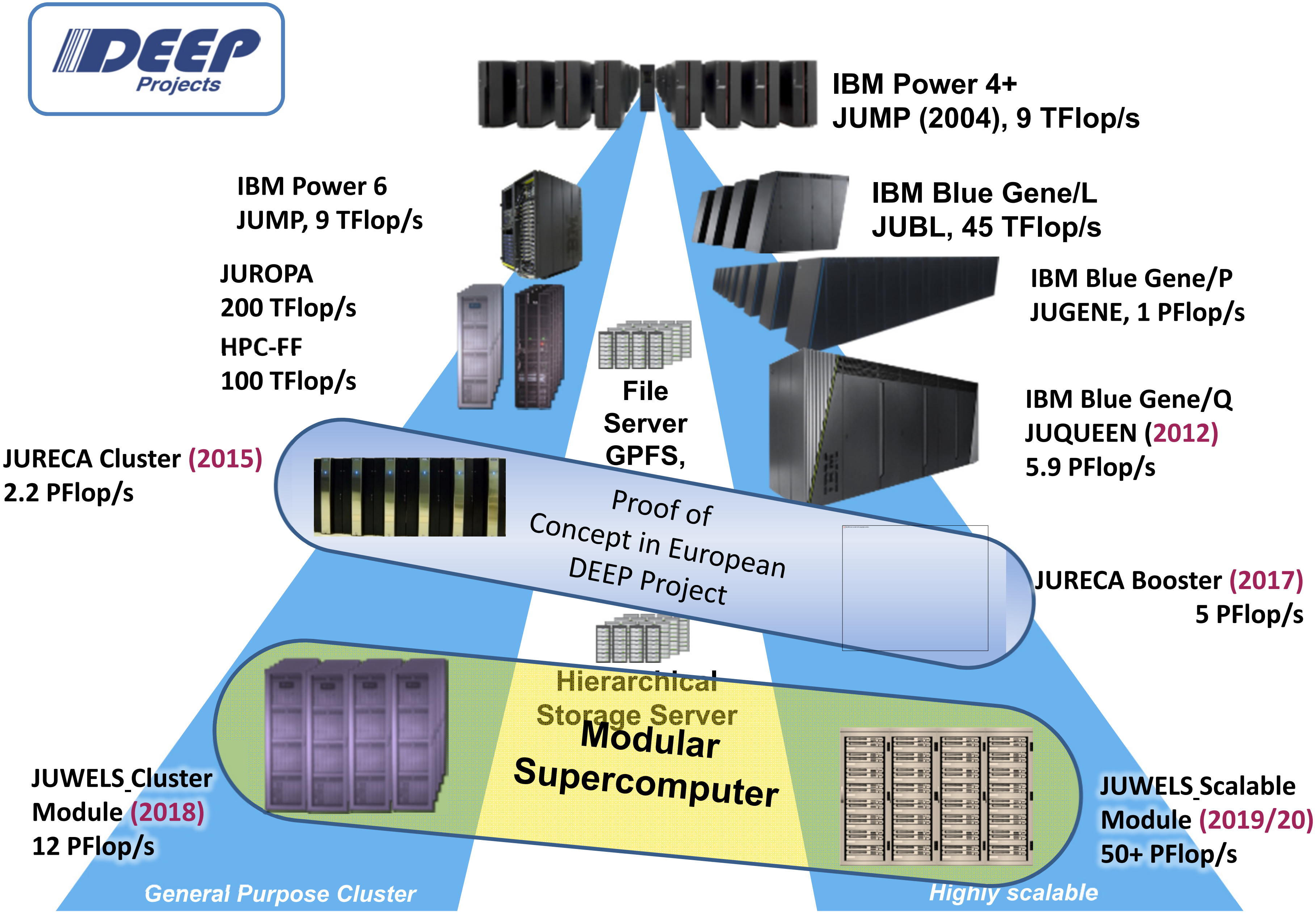
[28] DEEP-EST EU Project



DEEP Projects

DEEP-EST EU Project & Modular Supercomputing





DEEP Projects
IBM Power 4+
JUMP (2004), 9 TFlop/s
IBM Power 6
JUMP, 9 TFlop/s
IBM Blue Gene/L
JUBL, 45 TFlop/s
JUROPA
200 TFlop/s
HPC-FF
100 TFlop/s
IBM Blue Gene/P
JUGENE, 1 PFlop/s
File
Server
GPFS,
IBM Blue Gene/Q
JUQUEEN (2012)
5.9 PFlop/s
JURECA Cluster (2015)
2.2 PFlop/s
Proof of
Concept in European
DEEP Project
JURECA Booster (2017)
5 PFlop/s
Hierarchical
Storage Server
Modular
Supercomputer
JUWELS Cluster
Module (2018)
12 PFlop/s
JUWELS Scalable
Module (2019/20)
50+ PFlop/s
General Purpose Cluster
Highly scalable

Deep Learning for Sequence Data: Long Short-Term Memory

- Standard LSTM

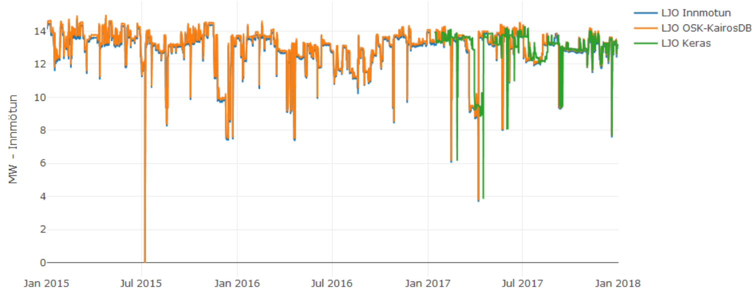
```
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from keras.layers import Dropout
```

```
# design network
model = Sequential()
model.add(LSTM(
    units=config['units'],
    input_shape=(train_X.shape[1], train_X.shape[2])
))
model.add(Dense(1, activation=config['activation']))

model.compile(loss=config['loss'], optimizer=config['optimizer'])

# fit network
print("Fitting model..")

history = model.fit(
    train_X,
    train_y,
    epochs=config['epochs'],
    batch_size=config['batchsize'],
    validation_data=(test_X, test_y),
    verbose=2,
    shuffle=config['shuffle']
)
```

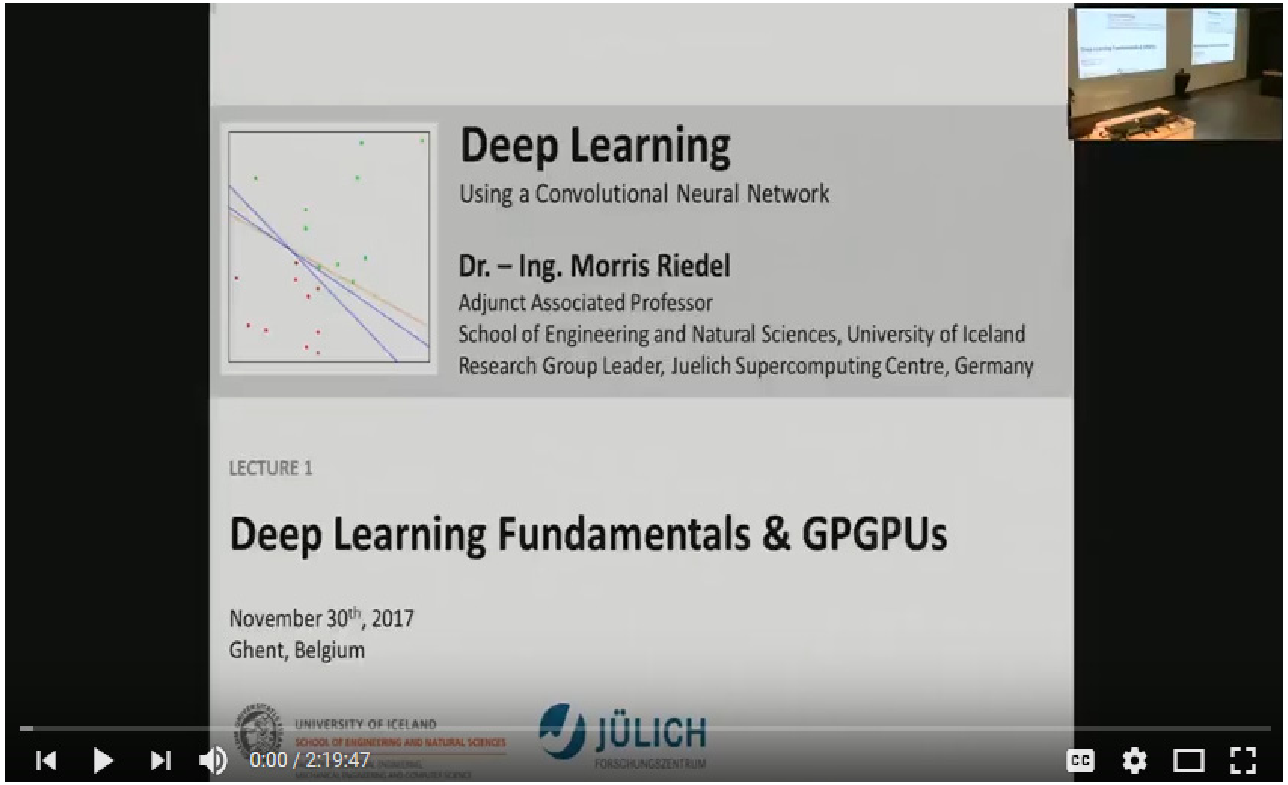
- **LSTM models work quite well to predict power but needs to be trained and tuned for different power stations**
- **Observing that some peaks can not be ‘learned‘**

Landsvirkjun

National Power Company of Iceland

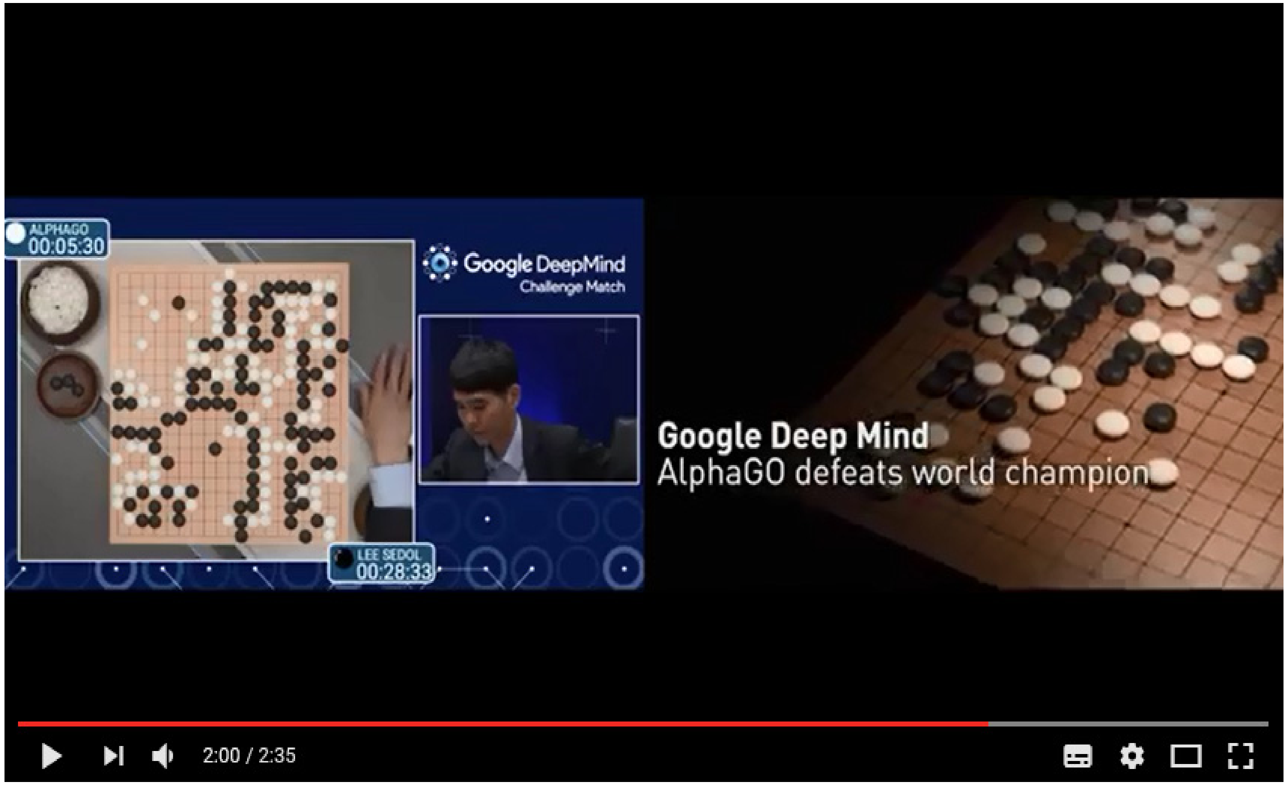


[YouTube Lectures] More about Deep Learning & HPC



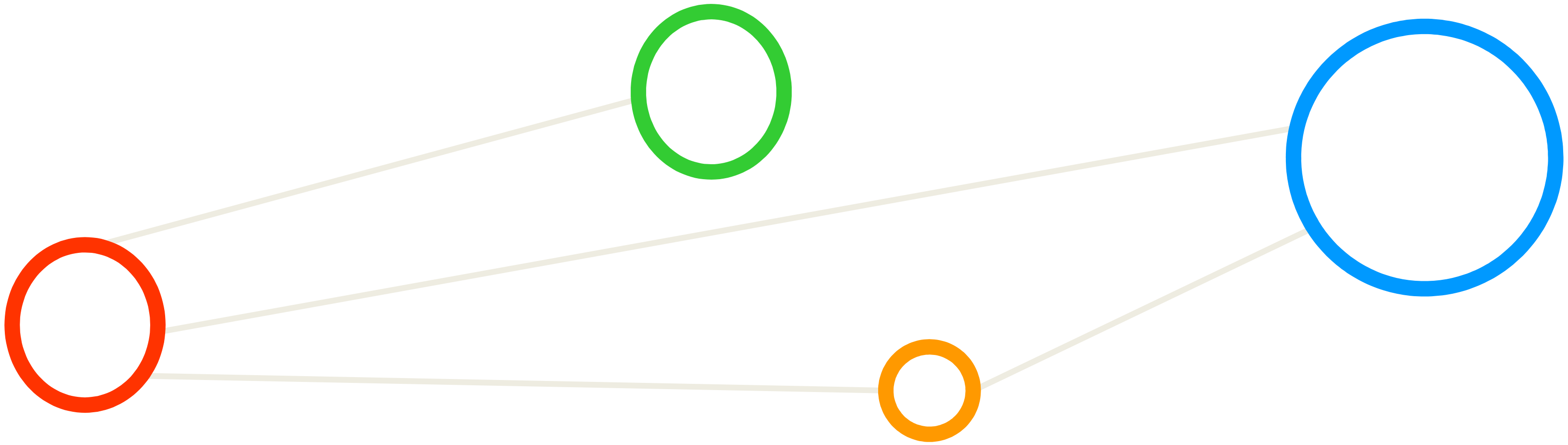
[21] Morris Riedel, 'Deep Learning - Using a Convolutional Neural Network', Invited YouTube Lecture, six lectures, University of Ghent, 2017

[Video] Deep Learning 'Revolution'



[27] The Deep Learning Revolution, YouTube

Summary



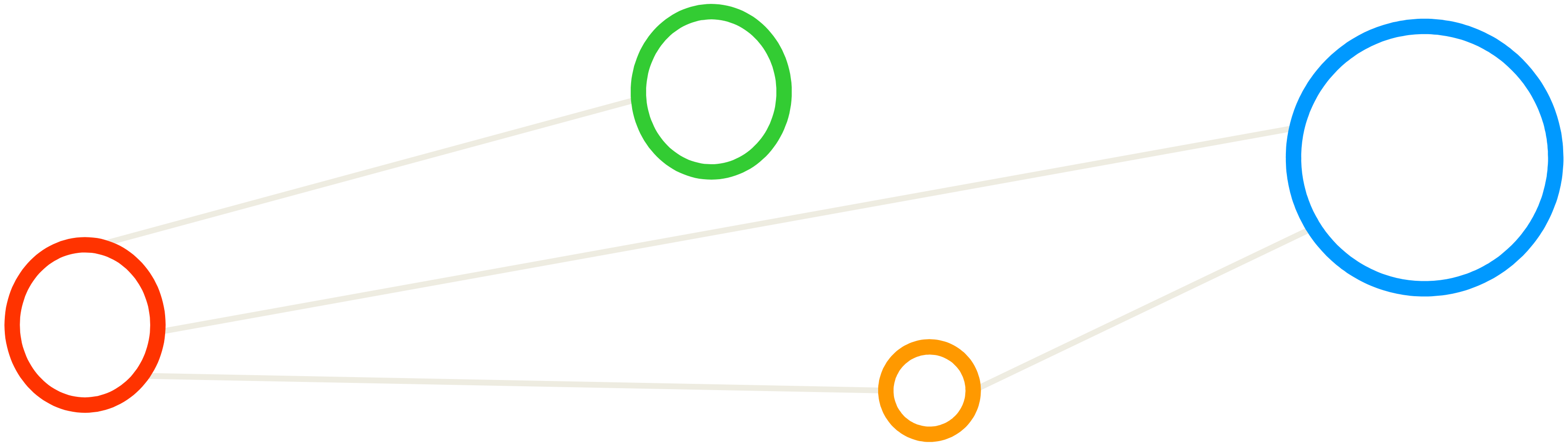
Summary

- Mindset
  - Think traditional machine learning still relevant for deep learning
  - Using interpreted languages like Python is 'modus operandi'
  - Selected new specific deep learning methods (CNN, LSTM, etc.)

- Skillset
  - Basic knowledge of machine learning required for deep learning
  - Validation (i.e. model selection) and regularization still valid(!)
  - Many job offers for specialists in machine/deep learning & HPC

- Toolset
  - Parallel versions of machine learning methods exist (piSVM, HPDBSCAN)
  - Python, Tensorflow & Keras often used for deep learning
  - Explore technology trends, e.g. specific chips for deep learning

Lecture Bibliography



Lecture Bibliography (1)

- [1] Introduction to Data Mining, Pang-Ning Tan, Michael Steinbach, Vipin Kumar, Addison Wesley, ISBN 0321321367, English, ~769 pages, 2005
- [2] PANGAEA Data Collection, Data Publisher for Earth & Environmental Science, Online: http://www.pangaea.de/
- [3] UCI Machine Learning Repository, Online: http://archive.ics.uci.edu/ml/datasets.html
- [4] Species Iris Group of North America Database, Online: http://www.signa.org
- [5] UCI Machine Learning Repository Iris Dataset, Online: https://archive.ics.uci.edu/ml/datasets/Iris
- [6] Wikipedia 'Sepal', Online: https://en.wikipedia.org/wiki/Sepal
- [7] Rattle Library for R, Online: http://rattle.togaware.com/
- [8] F. Rosenblatt, 'The Perceptron--a perceiving and recognizing automaton', Report 85-460-1, Cornell Aeronautical Laboratory, 1957
- [9] Rosenblatt,'The Perceptron: A probabilistic model for information storage and orgainzation in the brain', Psychological Review 65(6), pp. 386-408, 1958
- [10] PLA Algorithm, YouTube Video, Online:
- [11] C. Shearer, CRISP-DM model, Journal Data Warehousing, 5:13
- [12] Pete Chapman, 'CRISP-DM User Guide', 1999, Online: http://lyle.smu.edu/~mhd/8331f03/crisp.pdf

Lecture Bibliography (2)

- [13] An Introduction to Statistical Learning with Applications in R, Online: http://www-bcf.usc.edu/~gareth/ISL/index.html
- [14] B2Share, 'Iris Dataset LibSVM Format Preprocessing', Online: https://b2share.eudat.eu/records/37fb24847a73489a9c569d7033ad0238
- [15] Udacity, 'Overfitting', Online: https://www.youtube.com/watch?v=CxAxRCv9WoA
- [16] 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.
- [17] Indian Pines Raw and Processed
  Online: http://hdl.handle.net/11304/9ec5eac8-61b4-4617-ae1c-1f8c8cd3cd74
- [18] M. Goetz, M. Riedel et al.,' On Parallel and Scalable Classification and Clustering Techniques for Earth Science Datasets' 6th Workshop on Data Mining in Earth System Science, Proceedings of the International Conference of Computational Science (ICCS), Reykjavik, Online: http://www.proceedings.com/26605.html
- [19] Original piSVM tool, Online: http://pisvm.sourceforge.net/
- [20] Morris Riedel, 'Introduction to Machine Learning Algorithms', Invited YouTube Lecture, six lectures University of Ghent, 2017
  Online: https://www.youtube.com/watch?v=KgiuUZ3WeP8&list=PLrmNhuZo9sgbcWtMGN0i6G9HEvh08JG0J
- [21] Morris Riedel, 'Deep Learning - Using a Convolutional Neural Network', Invited YouTube Lecture, six lectures University of Ghent, 2017
  Online: https://www.youtube.com/watch?v=gOL1_YIosYk&list=PLrmNhuZo9sgZUdaZ-f6OHK2yFW1kTS2qF

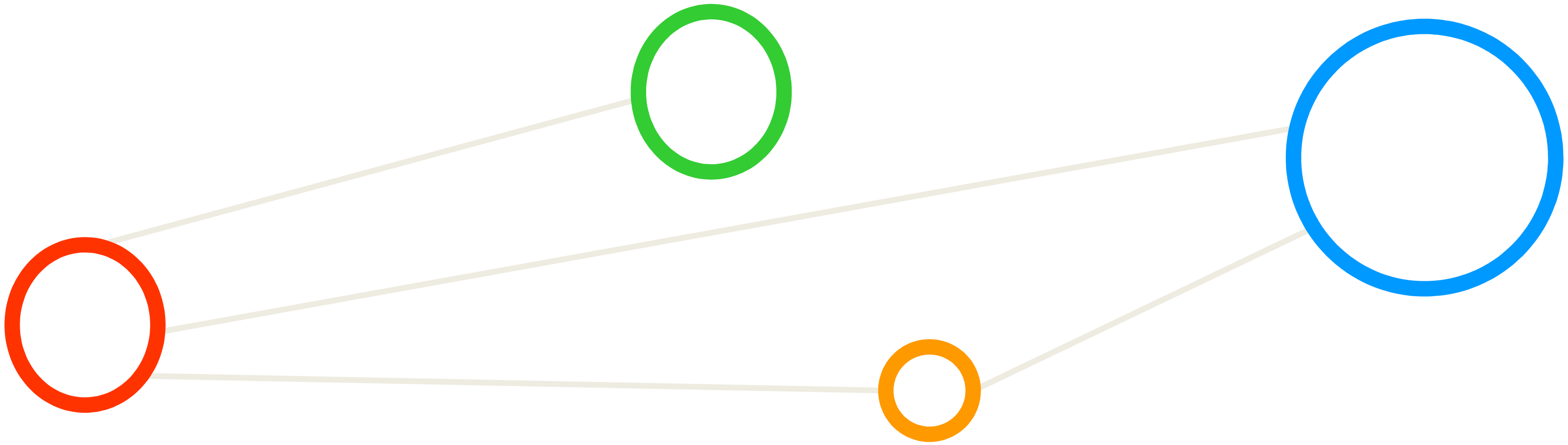
Lecture Bibliography (3)

- [22] B2SHARE, ‘HPDBSCAN Benchmark test files’,
  Online: http://hdl.handle.net/11304/6eacaa76-c275-11e4-ac7e-860aa0063d1f
- [23] Ester, Martin, et al. "A density-based algorithm for discoveringclusters in large spatial databases with noise." Kdd. Vol. 96. 1996.
- [24] M.Goetz, M. Riedel et al.,‘HPDBSCAN – Highly Parallel DBSCAN‘, MLHPC Workshop at Supercomputing 2015,
  Online: https://dl.acm.org/citation.cfm?id=2834894
- [25] H. Lee et al., ‘Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representations’, Proceedings of the 26th annual International Conference on Machine Learning (ICML), ACM, 2009
- [26] J. Lange, G. Cavallaro, M. Goetz, E. Erlingsson, M. Riedel, ‘The Influence of Sampling Methods on Pixel-Wise Hyperspectral Image Classification with 3D Convolutional Neural Networks’, Proceedings of the IGARSS 2018 Conference, to appear
- [27] YouTube Video, ‘The Deep Learning Revolution’,
  Online: https://www.youtube.com/watch?v=Dy0hJWltsyE
- [28] DEEP-EST EU Project,
  Online: http://www.deep-projects.eu/
- [29] 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
- [30] Keras Python Deep Learning Library,
  Online: https://keras.io/
- [31] Tensorflow Deep Learning Framework,
  Online: https://www.tensorflow.org/

Lecture Bibliography (4)

- [32] A Tour of Tensorflow,
  Online: https://arxiv.org/pdf/1610.01178.pdf
- [33] Big Data Tips, ‘What is a Tensor?‘,
  Online: http://www.big-data.tips/what-is-a-tensor
- [34] YouTube Video, ’Neural Networks, A Simple Explanation’,
  Online: http://www.youtube.com/watch?v=gcK_5x2KsLA
- [35] M. Nielsen, ‘Neural Networks and Deep Learning‘,
  Online: http://neuralnetworksanddeeplearning.com/
- [36] A. Rosebrock, ‘Get off the deep learning bandwagon and get some perspective‘, Online:
  http://www.pyimagesearch.com/2014/06/09/get-deep-learning-bandwagon-get-perspective/
- [37] A. Gulli and S. Pal, ‘Deep Learning with Keras‘ Book, ISBN-13 9781787128422, 318 pages,
  Online: https://www.packtpub.com/big-data-and-business-intelligence/deep-learning-keras
- [38] D. Kingma and Jimmy Ba, ‘Adam: A Method for Stochastic Optimization‘,
  Online: https://arxiv.org/abs/1412.6980

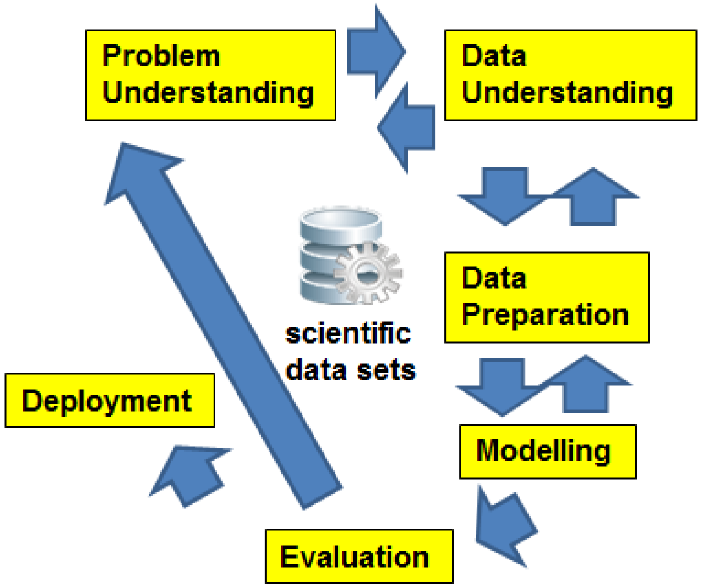
Appendix A: CRISP-DM Process



Summary: Systematic Process

- Systematic data analysis guided by a ‘standard process‘
  - Cross-Industry Standard Process for Data Mining (CRISP-DM)

- **A data mining project is guided by these six phases:**
  (1) Problem Understanding;
  (2) Data Understanding;
  (3) Data Preparation;
  (4) Modeling;
  (5) Evaluation;
  (6) Deployment



[11] C. Shearer, CRISP-DM model, Journal Data Warehousing, 5:13

- Lessons Learned from Practice
  - Go back and forth between the different six phases

1 – Problem (Business) Understanding

- **The Business Understanding phase consists of four distinct tasks: (A) Determine Business Objectives; (B) Situation Assessment; (C) Determine Data Mining Goal; (D) Produce Project Plan**

[12] CRISP-DM User Guide

- Task A – Determine Business Objectives
  - Background, Business Objectives, Business Success Criteria
- Task B – Situation Assessment
  - Inventory of Resources, Requirements, Assumptions, and Contraints
  - Risks and Contingencies, Terminology, Costs & Benefits
- Task C – Determine Data Mining Goal
  - Data Mining Goals and Success Criteria
- Task D – Produce Project Plan
  - Project Plan
  - Initial Assessment of Tools & Techniques

2 – Data Understanding

- **The Data Understanding phase consists of four distinct tasks:**
  (A) Collect Initial Data; (B) Describe Data; (C) Explore Data; (D) Verify Data Quality

[12] CRISP-DM User Guide

- Task A – Collect Initial Data
  - Initial Data Collection Report
- Task B – Describe Data
  - Data Description Report
- Task C – Explore Data
  - Data Exploration Report
- Task D – Verify Data Quality
  - Data Quality Report

3 – Data Preparation

- **The Data Preparation phase consists of six distinct tasks: (A) Data Set; (B) Select Data; (C) Clean Data; (D) Construct Data; (E) Integrate Data; (F) Format Data**

[12] CRISP-DM User Guide

- Task A – Data Set
  - Data set description
- Task B – Select Data
  - Rationale for inclusion / exclusion
- Task C – Clean Data
  - Data cleaning report
- Task D – Construct Data
  - Derived attributes, generated records
- Task E – Integrate Data
  - Merged data
- Task F – Format Data
  - Reformatted data

4 – Modeling

- **The Data Preparation phase consists of four distinct tasks: (A) Select Modeling Technique; (B) Generate Test Design; (C) Build Model; (D) Assess Model;**

[12] CRISP-DM User Guide

- Task A – Select Modeling Technique
  - Modeling assumption, modeling technique
- Task B – Generate Test Design
  - Test design
- Task C – Build Model
  - Parameter settings, models, model description
- Task D – Assess Model
  - Model assessment, revised parameter settings

5 – Evaluation

- **The Data Preparation phase consists of three distinct tasks: (A) Evaluate Results; (B) Review Process; (C) Determine Next Steps**

[12] CRISP-DM User Guide

- Task A – Evaluate Results
  - Assessment of data mining results w.r.t. business success criteria
  - List approved models
- Task B – Review Process
  - Review of Process
- Task C – Determine Next Steps
  - List of possible actions, decision

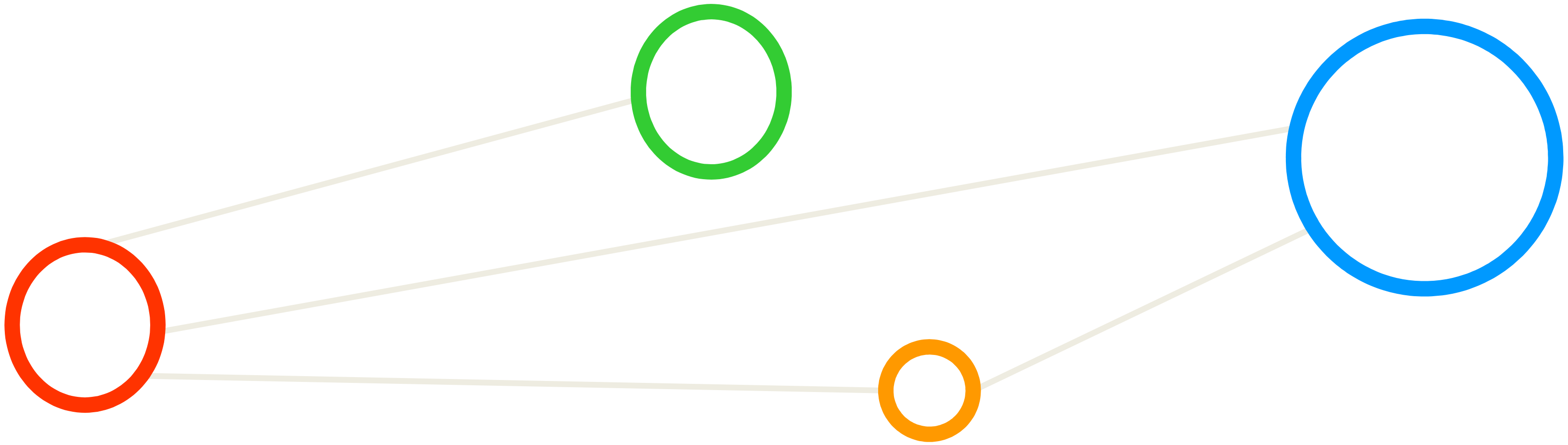
6 – Deployment

- **The Data Preparation phase consists of three distinct tasks: (A) Plan Deployment; (B) Plan Monitoring and Maintenance; (C) Produce Final Report; (D) Review Project**

[12] CRISP-DM User Guide

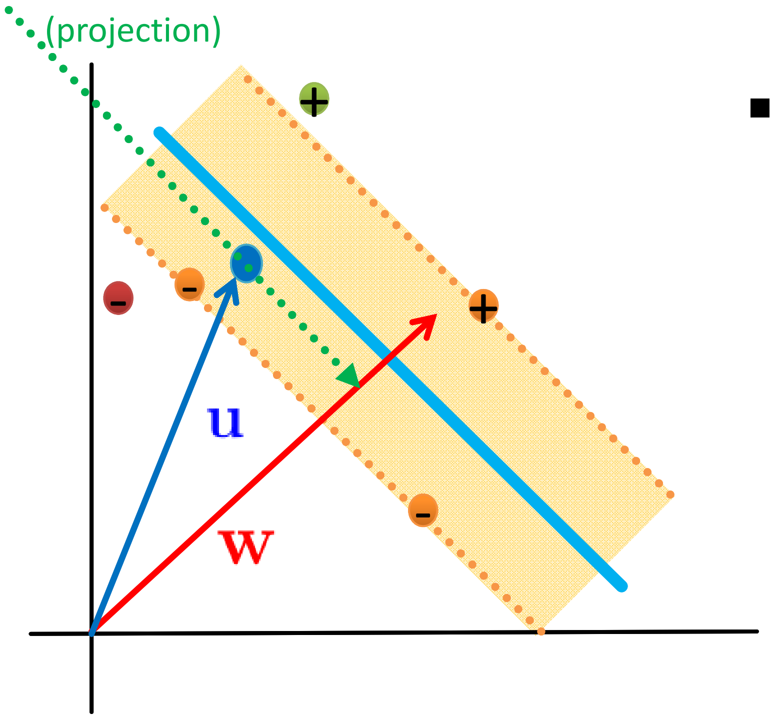
- Task A – Plan Deployment
  - Establish a deployment plan
- Task B – Plan Monitoring and Maintenance
  - Create a monitoring and maintenance plan
- Task C – Product Final Report
  - Create final report and provide final presentation
- Task D – Review Project
  - Document experience, provide documentation

Appendix B: Geometric Interpretation of SVMs & Kernels



Geometric SVM Interpretation and Setup (1)

- Think ‘simplified coordinate system‘ and use ‘Linear Algebra‘
  - Many other samples are removed (red and green not SVs) - +
  - Vector $\mathbf{w}$ of ‘any length‘ perpendicular to the decision boundary
  - Vector $\mathbf{u}$ points to an unknown quantity (e.g. new sample to classify)
  - Is $\mathbf{u}$ on the left or right side of the decision boundary?



- Dot product $\mathbf{w} \cdot \mathbf{u} \geq C; C = -b$
  - With $\mathbf{u}$ takes the projection on the $\mathbf{w}$
  - Depending on where projection is it is left or right from the decision boundary
  - Simple transformation brings decison rule:

1 $\mathbf{w} \cdot \mathbf{u} + b \geq 0$ → means +

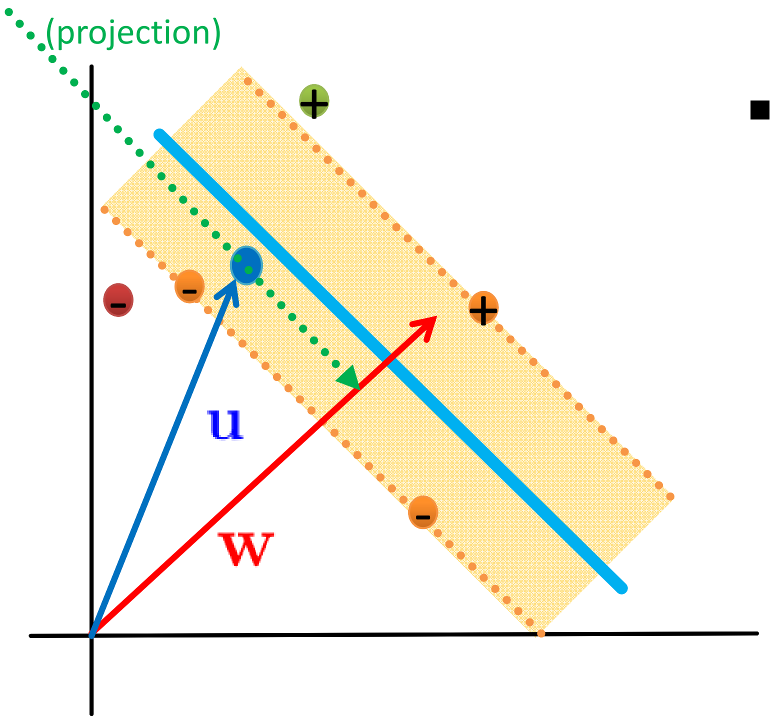
  - (given that b and $\mathbf{w}$ are unknown to us)

    (constraints are not enough to fix particular b or w, need more constraints to calculate b or w)

Geometric SVM Interpretation and Setup (2)

- Creating our constraints to get b or $\mathbf{w}$ computed
  - First constraint set for positive samples ➕ $\mathbf{w} \cdot \mathbf{x}_+ + b \geq 1$
  - Second constraint set for negative samples ➖ $\mathbf{w} \cdot \mathbf{x}_- + b \leq 1$
  - For mathematical convenience introduce variables (i.e. labelled samples)

    $y_i = +$ for ➕ and $y_i = -$ for ➖



- Multiply equations by y_i
  - Positive samples: $y_i(\mathbf{x}_i \cdot \mathbf{w} + b) \geq 1$
  - Negative samples: $y_i(\mathbf{x}_i \cdot \mathbf{w} + b) \geq 1$
  - Both same due to $y_i = +$ and $y_i = -$

    (brings us mathematical convenience often quoted)

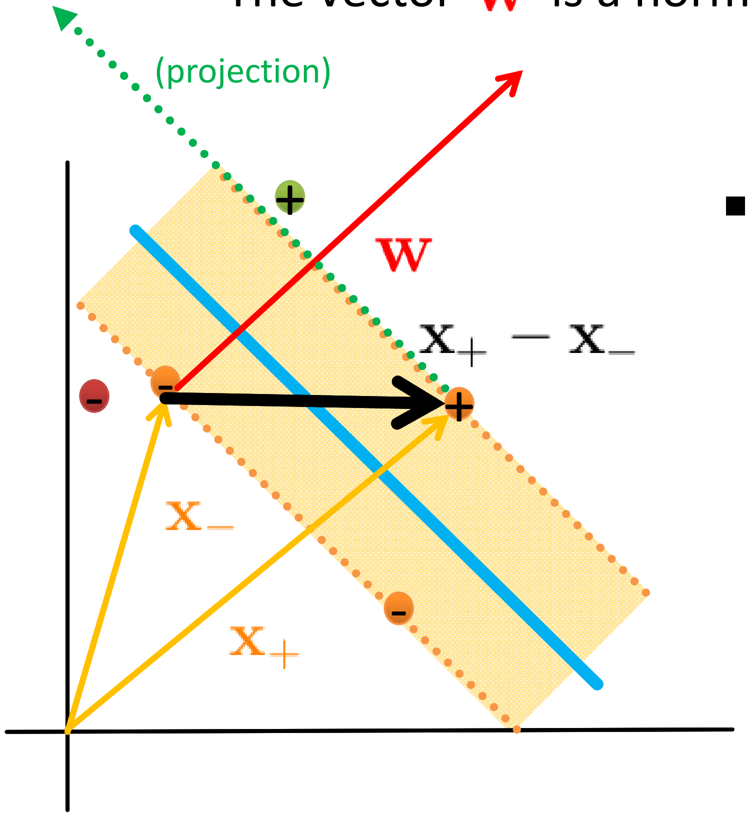
    $$y_i(\mathbf{x}_i \cdot \mathbf{w} + b) - 1 \geq 0$$

    (additional constraints just for support vectors itself helps)

    (2) $$y_i(\mathbf{x}_i \cdot \mathbf{w} + b) - 1 = 0$$

Geometric SVM Interpretation and Setup (3)

- Determine the 'width of the margin'
  - Difference between positive and negative SVs: $\mathbf{x}_+ - \mathbf{x}_-$
  - Projection of $\mathbf{x}_+ - \mathbf{x}_-$ onto the vector $\mathbf{w}$
  - The vector $\mathbf{w}$ is a normal vector, magnitude is $\|\mathbf{w}\|$



(Dot product of two vectors is a scalar, here the width of the margin)

- Unit vector is helpful for 'margin width'
  - Projection (dot product) for margin width:

$$\mathbf{x}_+ - \mathbf{x}_- \cdot \boxed{\frac{\mathbf{w}}{\|\mathbf{w}\|}}$$

(unit vector)

$$1 - b \qquad 1 + b \qquad \Rightarrow \qquad \frac{2}{\|\mathbf{w}\|} \quad (3)$$

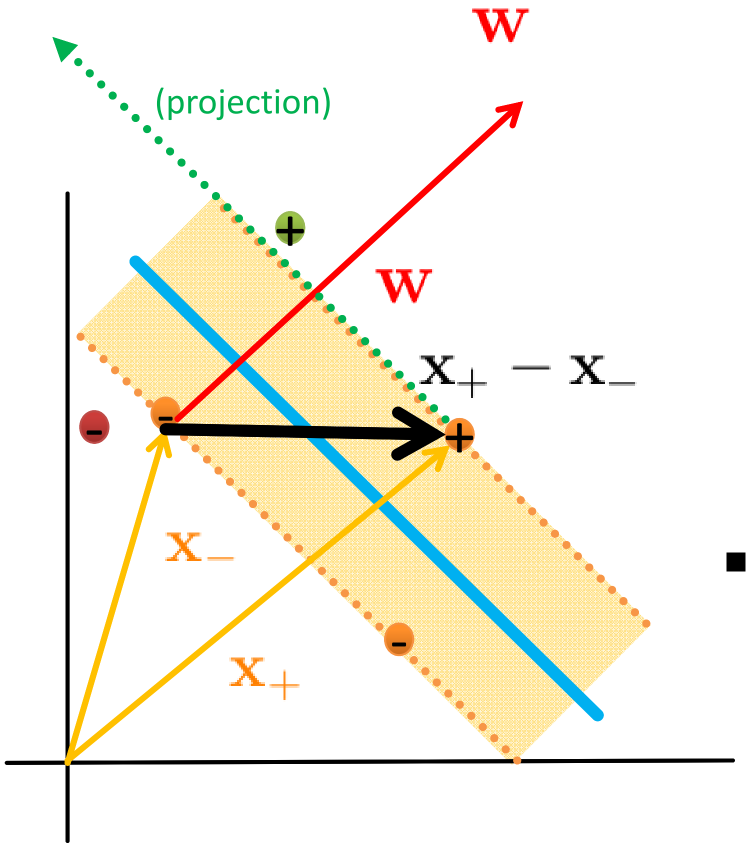
- When enforce constraint:

$$(2) \quad y_i(\mathbf{x}_i \cdot \mathbf{w} + b) - 1 = 0$$

$$y_i = + \qquad y_i = -$$

Constrained Optimization Steps SVM (1)

- Use 'constraint optimization' of mathematical toolkit
  - Idea is to 'maximize the width' of the margin: $max \frac{2}{\|\mathbf{w}\|}$ (drop the constant 2 is possible here)



$$\Rightarrow max \frac{1}{\|\mathbf{w}\|}$$ (equivalent)

$$\Rightarrow min \|\mathbf{w}\|$$ (equivalent for max)

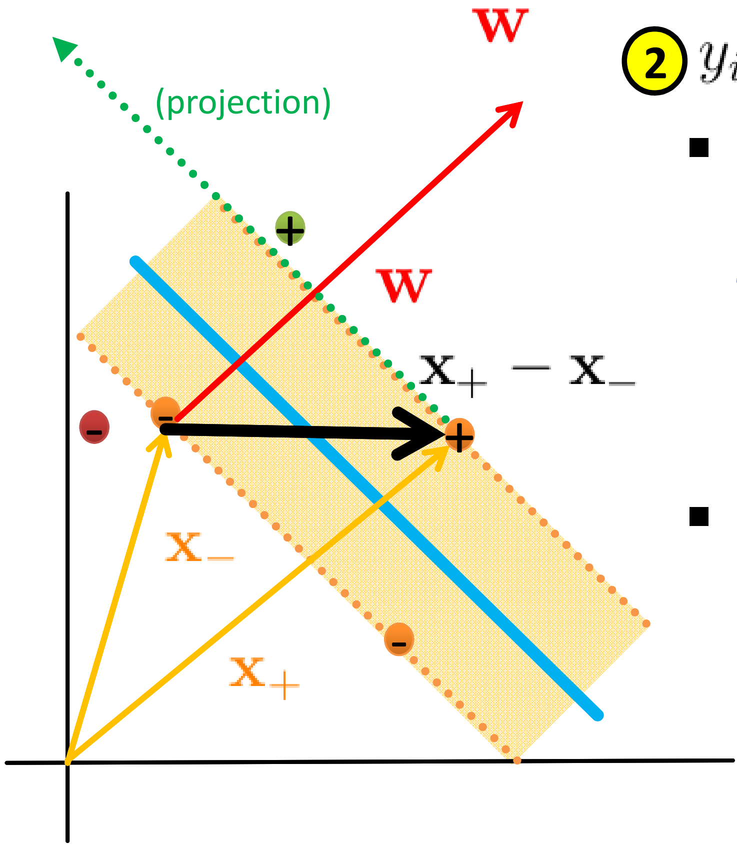
$$\Rightarrow min \frac{1}{2}\|\mathbf{w}\|^2$$ (mathematical convenience) 3

- Next: Find the extreme values
  - Subject to constraints

2 $$y_i(\mathbf{x}_i \cdot \mathbf{w} + b) - 1 = 0$$

Constrained Optimization Steps SVM (2)

- Use 'Lagrange Multipliers' of mathematical toolkit
  - Established tool in 'constrained optimization' to find function extremum
  - 'Get rid' of constraints by using Lagrange Multipliers 4



2 $y_i(\mathbf{x}_i \cdot \mathbf{w} + b - 1) = 0$

- Introduce a multiplier for each constraint

$$\mathcal{L}(\alpha) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum \alpha_i [y_i(\mathbf{x}_i \cdot \mathbf{w} + b) - 1]$$

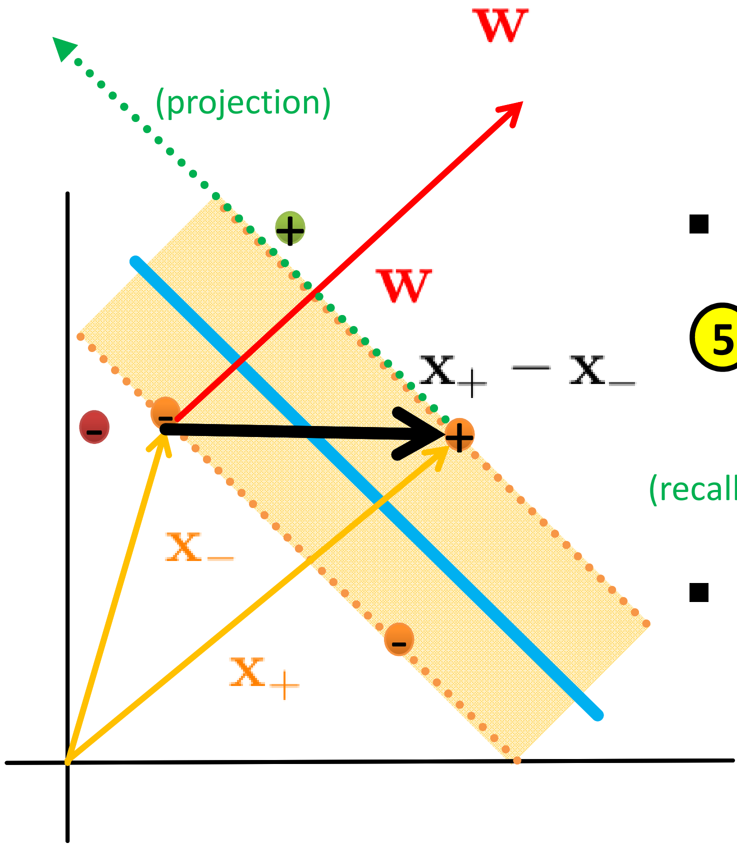
(interesting: non zero for support vectors, rest zero)

- Find derivatives for extremum & set 0
  - But two unknowns that might vary
  - First differentiate w.r.t. $\mathbf{w}$
  - Second differentiate w.r.t. b

(derivative gives the gradient, setting 0 means extremum like min)

Constrained Optimization Steps SVM (3)

- Lagrange gives: $\mathcal{L}(\alpha) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum \alpha_i [y_i(\mathbf{x}_i \cdot \mathbf{w} + b) - 1]$



- First differentiate w.r.t $\mathbf{w}$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = \mathbf{w} - \boxed{\sum \alpha_i y_i \mathbf{x}_i} = 0$$

(derivative gives the gradient, setting 0 means extremum like min)

- Simple transformation brings:

5 $\mathbf{w} = \boxed{\sum \alpha_i y_i \mathbf{x}_i}$ (i.e. vector is linear sum of samples)

(recall: non zero for support vectors, rest zero → even less samples)

- Second differentiate w.r.t. b

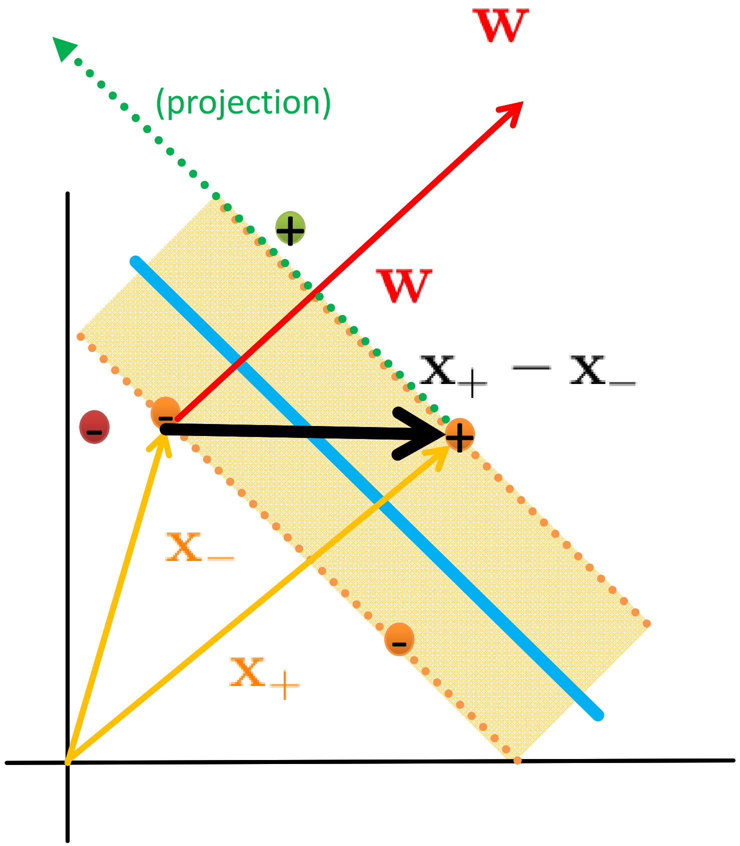
$$\frac{\partial \mathcal{L}}{\partial b} = -\sum \alpha_i y_i = 0 \Rightarrow \sum \alpha_i y_i = 0$$ 5

Constrained Optimization Steps SVM (4)

- Lagrange gives: $\mathcal{L}(\alpha) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum \alpha_i[y_i(\mathbf{x}_i \cdot \mathbf{w} + b) - 1]$

(plug into)

- Find minimum
  - Quadratic optimization problem
  - Take advantage of 5 $\mathbf{w} = \sum \alpha_i y_i \mathbf{x}_i$



$$\mathcal{L} = \frac{1}{2}(\sum \alpha_i y_i \mathbf{x}_i) \cdot (\sum \alpha_j y_j \mathbf{x}_j)$$

$$- \sum \alpha_i y_i \mathbf{x}_i \cdot (\sum \alpha_j y_j \mathbf{x}_j)$$

$$- \sum \alpha_i y_i b + \sum \alpha_i$$

(b constant in front sum)

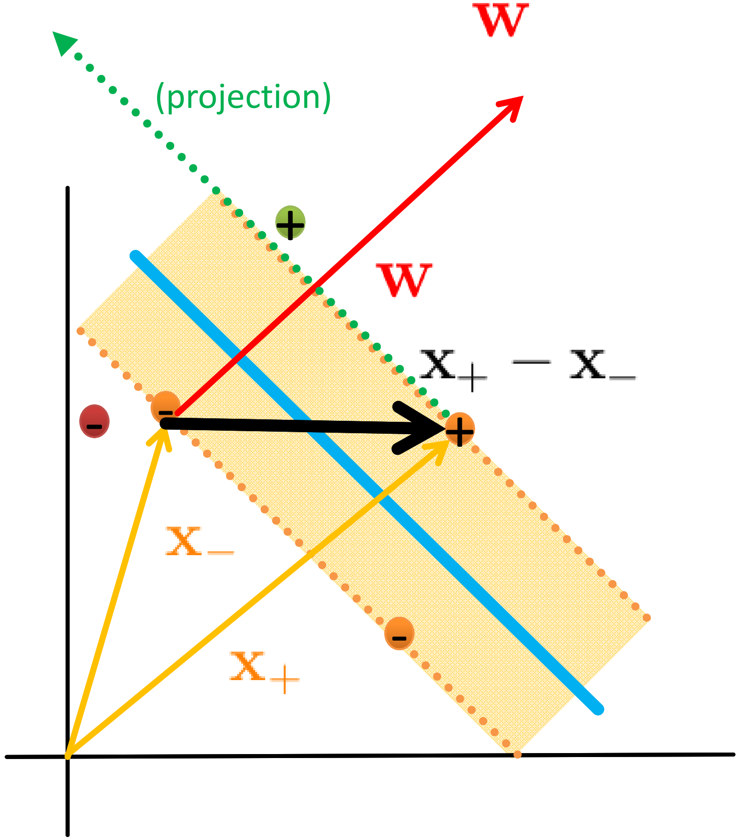
5 $\sum \alpha_i y_i = 0$

Constrained Optimization Steps SVM (5)

- Rewrite formula: $\mathcal{L} = \frac{1}{2}\boxed{(\sum \alpha_i y_i \mathbf{x}_i) \cdot (\sum \alpha_j y_j \mathbf{x}_j) - \sum \alpha_i y_i \mathbf{x}_i \cdot (\sum \alpha_j y_j \mathbf{x}_j)}$ (the same)

$$\cancel{- \sum \alpha_i y_i b} + \sum \alpha_i$$

(was 0)



(results in)

(optimization depends only on dot product of samples)

$$\mathcal{L} = \sum \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j \boxed{\mathbf{x}_i \cdot \mathbf{x}_j} \quad 6$$

- Equation to be solved by some quadratic programming package

Use of SVM Classifier to Perform Classification

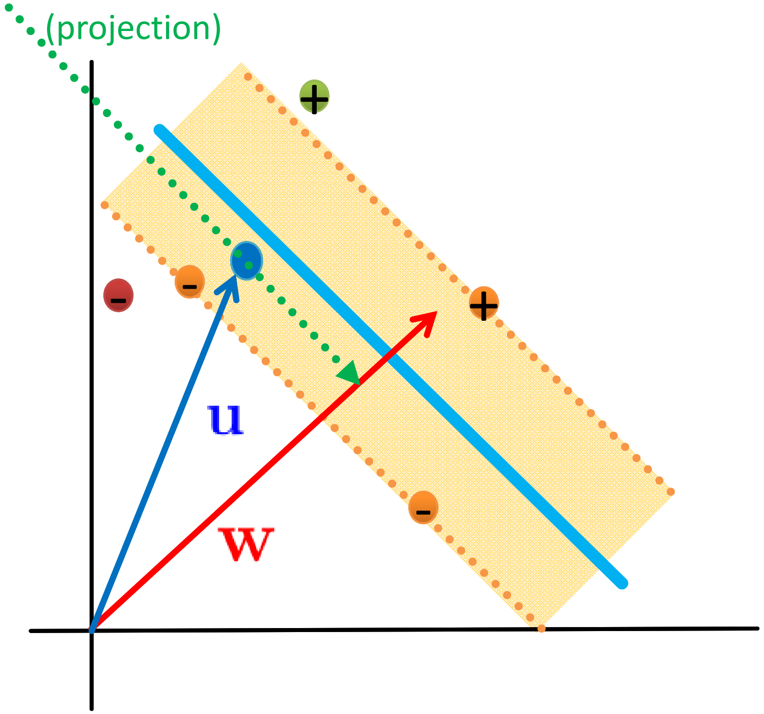
- Use findings for decision rule

5 $\mathbf{w} = \sum \alpha_i y_i \mathbf{x}_i$

1 $\mathbf{w} \cdot \mathbf{u} + b \geq 0$ +

$\sum \alpha_i y_i \boxed{\mathbf{x}_i \cdot \mathbf{u}_i} + b \geq 0$ +

(decision rule also depends on dotproduct)

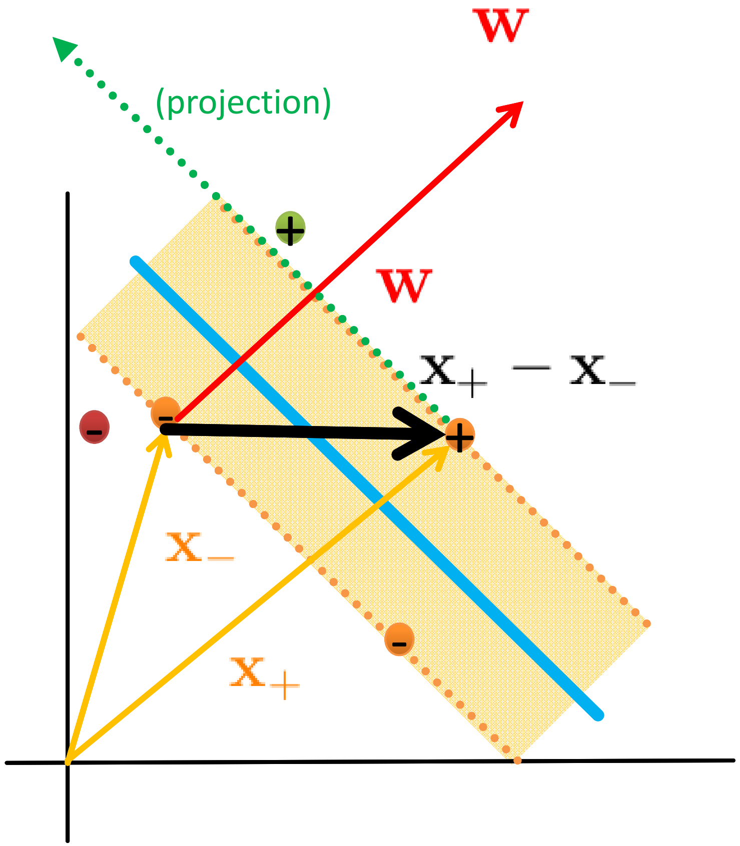


Constrained Optimization Steps SVM & Dot Product

- Rewrite formula: $\mathcal{L} = \frac{1}{2}\boxed{(\sum \alpha_i y_i \mathbf{x}_i) \cdot (\sum \alpha_j y_j \mathbf{x}_j) - \sum \alpha_i y_i \mathbf{x}_i \cdot (\sum \alpha_j y_j \mathbf{x}_j)}$ (the same)

$$- \cancel{\sum \alpha_i y_i b} + \sum \alpha_i$$

(was 0)



(results in)

(optimization depends only on dot product of samples)

$$\mathcal{L} = \sum \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j \boxed{\mathbf{x}_i \cdot \mathbf{x}_j} \quad (6)$$

- Equation to be solved by some quadratic programming package

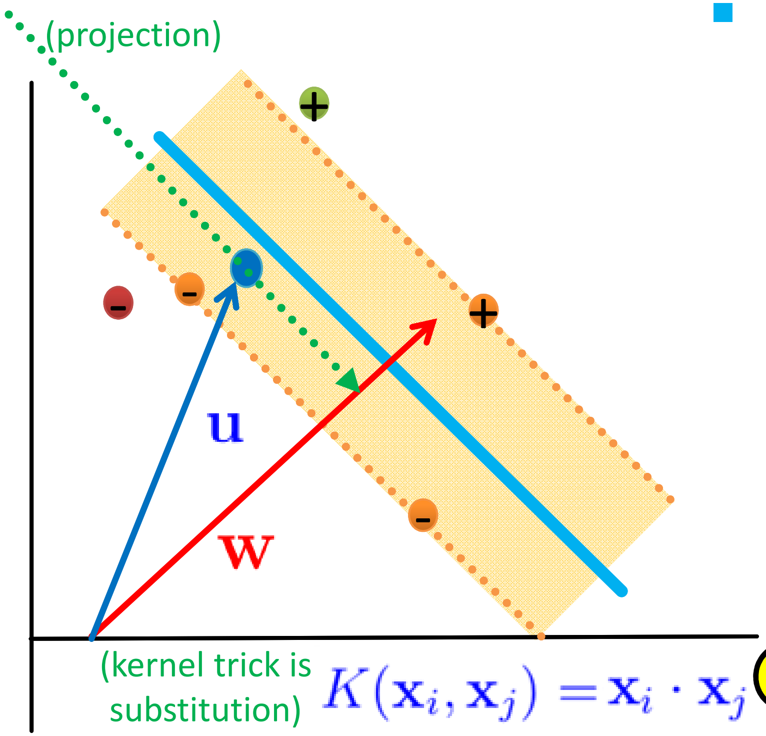
Kernel Methods & Dot Product Dependency

- Use findings for decision rule

5 $\mathbf{w} = \sum \alpha_i y_i \mathbf{x}_i$

(decision rule also depends on dotproduct)

1 $\mathbf{w} \cdot \mathbf{u} + b \geq 0$ + ⇒ $\sum \alpha_i y_i \boxed{\mathbf{x}_i \cdot \mathbf{u}_i} + b \geq 0$ +



- Dotproduct enables nice more elements
  - E.g. consider non linearly seperable data
  - Perform non-linear transformation Φ of the samples into another space (work on features)

$$\mathcal{L} = \sum \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j \boxed{\mathbf{x}_i \cdot \mathbf{x}_j}$$ 6

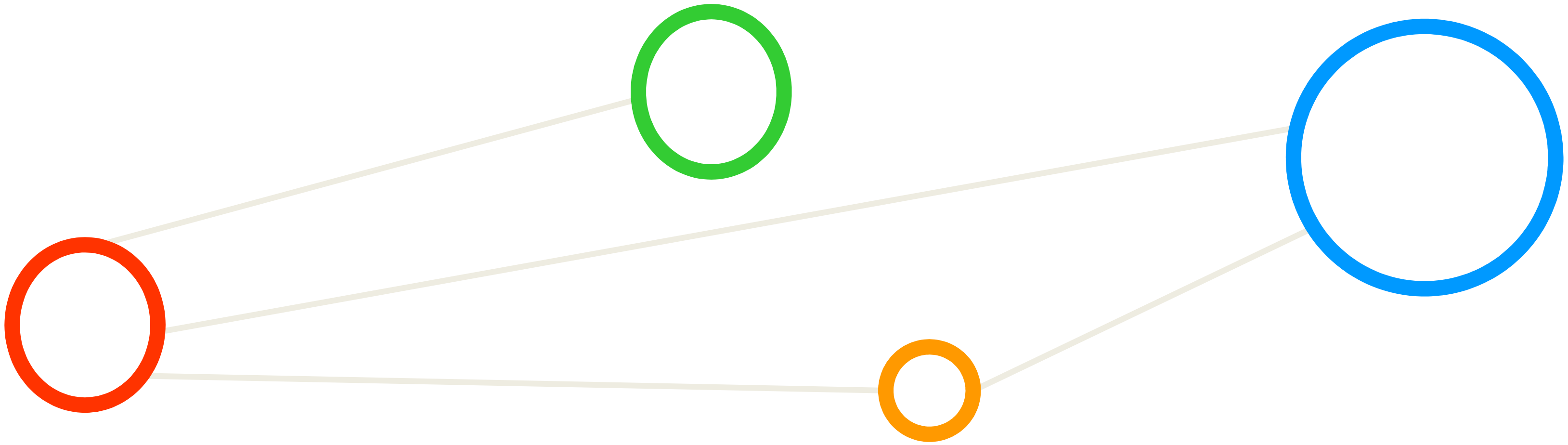
(optimization depends only on dot product of samples)

⇒ $\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$ (in optimization)

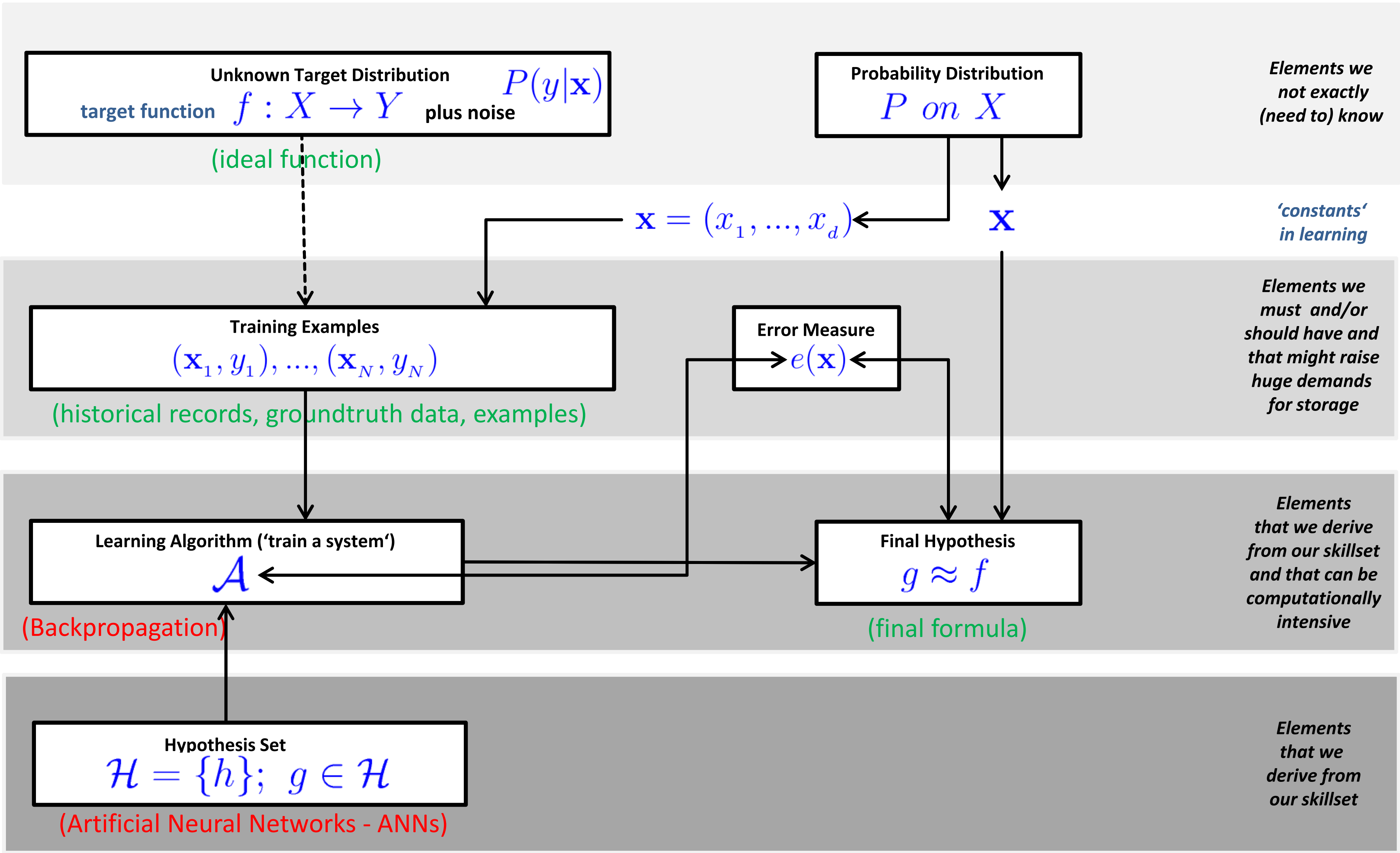
⇒ $\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{u}_i)$ (for decision rule above too)

(kernel trick is substitution) $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i \cdot \mathbf{x}_j$ 7 $\boxed{K(\mathbf{x}_i, \mathbf{x}_j)} = \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$ (trusted Kernel avoids to know Phi)

Appendix C: Convolutional Neural Networks in Keras



Solution Tools: Artificial Neural Network Learning Model



ANN – Handwritten Character Recognition MNIST Dataset

- Metadata
  - Subset of a larger dataset from US National Institute of Standards (NIST)
  - Handwritten digits including corresponding labels with values 0 to 9
  - All digits have been size-normalized to 28 * 28 pixels
    and are centered in a fixed-size image for direct processing
  - Not very challenging dataset, but good for experiments / tutorials

- Dataset Samples
  - Labelled data (10 classes)
  - Two separate files
    for training and test
  - 60000 training samples (~47 MB)
  - 10000 test samples (~7.8 MB)

MNIST Dataset for the Tutorial

- When working with the dataset
  - Dataset is not in any standard image format like jpg, bmp, or gif
  - File format not known to a graphics viewer
  - One needs to write typically a small program to read and work for them
  - Data samples are stored in a simple file format that is designed for storing vectors and multidimensional matrices
  - The pixels of the handwritten digit images are organized row-wise with pixel values ranging from 0 (white background) to 255 (black foreground)
  - Images contain grey levels as a result of an anti-aliasing technique used by the normalization algorithm that generated this dataset.
- Available already for the tutorial
  - Part of the Tensorflow tutorial package and Keras tutorial package

```
# download & unpack MNIST data
from tensorflow.examples.tutorials.mnist import input_data
mnist = input_data.read_data_sets("MNIST_data/", one_hot=True)
```

MNIST Dataset for the Tutorial

- When working with the dataset
  - Dataset is not in any standard image format like jpg, bmp, or gif
  - One needs to write typically a small program to read and work for them
  - Data samples are stored in a simple file format that is designed for storing vectors and multidimensional matrices (here numpy binary files)
  - The pixels of the handwritten digit images are organized row-wise with pixel values ranging from 0 (white background) to 255 (black foreground)
  - Images contain grey levels as a result of an anti-aliasing technique used by the normalization algorithm that generated this dataset.

```
/homea/hpclab/train001/data/mnist
[train001@jrl09 mnist]$ pwd
/homea/hpclab/train001/data/mnist
[train001@jrl09 mnist]$ ls -al
total 53728
drwxr-xr-x  2 train001 hpclab      512 Jun  6 12:17 .
drwxr-xr-x 10 train001 hpclab      512 Jun  6 12:17 ..
-rw-r-----  1 train001 hpclab  7840080 Jun  6 12:17 x_test.npy
-rw-r-----  1 train001 hpclab 47040080 Jun  6 12:17 x_train.npy
-rw-r-----  1 train001 hpclab    10080 Jun  6 12:17 y_test.npy
-rw-r-----  1 train001 hpclab    60080 Jun  6 12:17 y_train.npy
```

MNIST Dataset – Exploration – One Character Encoding

```
[train001@jrl09 mnist]$ python explore-mnist-training.py
Samples of 28 x 28 pixel matrices reserved for training
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   3   18  18  18  126 136 175 26  166 255 247 127 0   0   0   0
0   0   0   0   0   0   0   0   30  36  94  154 170 253 253 253 253 253 225 172 253 242 195 64  0   0   0   0
0   0   0   0   0   0   0   49  238 253 253 253 253 253 253 253 253 251 93  82  82  56  39  0   0   0   0   0
0   0   0   0   0   0   0   18  219 253 253 253 253 253 198 182 247 241 0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   80  156 107 253 253 205 11  0   43  154 0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   14  1   154 253 90  0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   139 253 190 2   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   11  190 253 70  0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   35  241 225 160 108 1   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   81  240 253 253 119 25  0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   45  186 253 253 150 27  0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   16  93  252 253 187 0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   249 253 249 64  0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   46  130 183 253 253 207 2   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   39  148 229 253 253 253 250 182 0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   24  114 221 253 253 253 253 201 78  0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   23  66  213 253 253 253 253 198 81  2   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   18  171 219 253 253 253 253 195 80  9   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   55  172 226 253 253 253 253 244 133 11  0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   136 253 253 253 212 135 132 16  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0


Label:
5
```

MNIST Dataset – Exploration Script Training

```
import numpy as np

# n x 28 x 28 pixel training data
X_train = np.load("/homea/hpclab/train001/data/mnist/x_train.npy")

# n x 1 training labels
Y_train = np.load("/homea/hpclab/train001/data/mnist/y_train.npy")
```

```
print("Samples of 28 x 28 pixel matrices reserved for training")

# function for showing a character
def character_show(character):
    for y in character:
        row = ""
        for x in y:
            row += '{0: <4}'.format(x)
        print row
```

```
# view first 10 characters
for i in range (0,9):
  character_show(X_train[i])
  print("\n")
  print("Label:")
  print(Y_train[i])
  print("\n")
```

- Loading MNIST training datasets (X) with labels (Y) stored in a binary numpy format
- Format is 28 x 28 pixel values with grey level from 0 (white background) to 255 (black foreground)

- Small helper function that prints row-wise one 'hand-written' character with the grey levels stored in training dataset
- Should reveal the nature of the number (aka label)

- Loop of the training dataset and the testing dataset (e.g. first 10 characters as shown here)
- At each loop interval the 'hand-written' character (X) is printed in 'matrix notation' & label (Y)

MNIST Dataset – Exploration – Selected Training Samples

```
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 3 18 18 18 126 136 175 26 166 255 247 127 0 0 0 0
0 0 0 0 0 0 0 0 30 36 94 154 170 253 253 253 253 253 225 172 253 242 195 64 0 0 0 0
0 0 0 0 0 0 0 49 238 253 253 253 253 253 253 253 253 251 93 82 82 56 39 0 0 0 0 0
0 0 0 0 0 0 0 18 219 253 253 253 253 253 198 182 247 241 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 80 156 107 253 253 205 11 0 43 154 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 14 1 154 253 90 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 139 253 190 2 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 11 190 253 70 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 35 241 225 160 108 1 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 81 240 253 253 119 25 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 45 186 253 253 150 27 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 16 93 252 253 187 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 249 253 249 64 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 46 130 183 253 253 207 2 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 39 148 229 253 253 253 250 182 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 24 114 221 253 253 253 253 201 78 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 23 66 213 253 253 253 253 198 81 2 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 18 171 219 253 253 253 253 195 80 9 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 55 172 226 253 253 253 253 244 133 11 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 136 253 253 253 212 135 132 16 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0


Label:
5
```

```
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 51 159 253 159 50 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 48 238 252 252 252 237 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 54 227 253 252 239 233 252 57 6 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 10 60 224 252 253 252 202 84 252 253 122 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 163 252 252 252 253 252 252 96 189 253 167 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 51 238 253 253 190 114 253 228 47 79 255 168 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 48 238 252 252 179 12 75 121 21 0 0 253 243 50 0 0 0 0 0
0 0 0 0 0 0 0 0 38 165 253 233 208 84 0 0 0 0 0 0 253 252 165 0 0 0 0 0
0 0 0 0 0 0 0 7 178 252 240 71 19 28 0 0 0 0 0 0 253 252 195 0 0 0 0 0
0 0 0 0 0 0 0 57 252 252 63 0 0 0 0 0 0 0 0 0 253 252 195 0 0 0 0 0
0 0 0 0 0 0 0 198 253 190 0 0 0 0 0 0 0 0 0 0 255 253 196 0 0 0 0 0
0 0 0 0 0 0 76 246 252 112 0 0 0 0 0 0 0 0 0 0 253 252 148 0 0 0 0 0
0 0 0 0 0 0 85 252 230 25 0 0 0 0 0 0 0 0 7 135 253 186 12 0 0 0 0 0
0 0 0 0 0 0 85 252 223 0 0 0 0 0 0 0 0 7 131 252 225 71 0 0 0 0 0 0
0 0 0 0 0 0 85 252 145 0 0 0 0 0 0 0 48 165 252 173 0 0 0 0 0 0 0 0
0 0 0 0 0 0 86 253 225 0 0 0 0 0 0 114 238 253 162 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 85 252 249 146 48 29 85 178 225 253 223 167 56 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 85 252 252 252 229 215 252 252 252 196 130 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 28 199 252 252 253 252 252 233 145 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 25 128 252 253 252 141 37 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0


Label:
0
```

```
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 67 232 39 0 0 0 0 0
0 0 0 0 62 81 0 0 0 0 0 0 0 0 0 0 0 0 0 0 120 180 39 0 0 0 0 0
0 0 0 0 126 163 0 0 0 0 0 0 0 0 0 0 0 0 0 2 153 210 40 0 0 0 0 0
0 0 0 0 220 163 0 0 0 0 0 0 0 0 0 0 0 0 0 27 254 162 0 0 0 0 0 0
0 0 0 0 222 163 0 0 0 0 0 0 0 0 0 0 0 0 0 183 254 125 0 0 0 0 0 0
0 0 0 46 245 163 0 0 0 0 0 0 0 0 0 0 0 0 0 198 254 56 0 0 0 0 0 0
0 0 0 120 254 163 0 0 0 0 0 0 0 0 0 0 0 0 23 231 254 29 0 0 0 0 0 0
0 0 0 159 254 120 0 0 0 0 0 0 0 0 0 0 0 0 163 254 216 16 0 0 0 0 0 0
0 0 0 159 254 67 0 0 0 0 0 0 0 0 0 14 86 178 248 254 91 0 0 0 0 0 0 0
0 0 0 159 254 85 0 0 0 47 49 116 144 150 241 243 234 179 241 252 40 0 0 0 0 0 0 0
0 0 0 150 253 237 207 207 207 253 254 250 240 198 143 91 28 5 233 250 0 0 0 0 0 0 0 0
0 0 0 0 119 177 177 177 177 177 98 56 0 0 0 0 0 102 254 220 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 169 254 137 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 169 254 57 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 169 254 57 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 169 255 94 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 169 254 96 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 169 254 153 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 169 255 153 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 96 254 153 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0


Label:
4
```

```
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 13 25 100 122 7 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 33 151 208 252 252 252 146 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 40 152 244 252 253 224 211 252 232 40 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 15 152 239 252 252 252 216 31 37 252 252 60 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 96 252 252 252 252 217 29 0 37 252 252 60 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 181 252 252 220 167 30 0 0 77 252 252 60 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 26 128 58 22 0 0 0 0 100 252 252 60 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 157 252 252 60 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 110 121 122 121 202 252 194 3 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 10 53 179 253 253 255 253 253 228 35 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 5 54 227 252 243 228 170 242 252 252 231 117 6 0 0 0 0 0 0 0
0 0 0 0 0 0 0 6 78 252 252 125 59 0 18 208 252 252 252 252 87 7 0 0 0 0 0 0
0 0 0 0 0 0 5 135 252 252 180 16 0 21 203 253 247 129 173 252 252 184 66 49 49 0 0 0
0 0 0 0 0 3 136 252 241 106 17 0 53 200 252 216 65 0 14 72 163 241 252 252 223 0 0 0
0 0 0 0 0 105 252 242 88 18 73 170 244 252 126 29 0 0 0 0 0 89 180 180 37 0 0 0
0 0 0 0 0 231 252 245 205 216 252 252 252 124 3 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 207 252 252 252 252 178 116 36 4 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 13 93 143 121 23 6 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0


Label:
2
```

ANN –MNIST Dataset – Parameters & Data Normalization

```
import numpy as np
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers.core import Dense, Activation
from keras.utils import np_utils
```

```
# parameters
NB_CLASSES = 10
NB_EPOCH = 200
BATCH_SIZE = 128
VERBOSE = 1
N_HIDDEN = 128
OPTIMIZER = 'SGD'
VALIDATION_SPLIT = 0.2
```

- **NB_CLASSES: 10 Class Problem**
- **NB_EPOCH: number of times the model is exposed to the training set – at each iteration the optimizer adjusts the weights so that the objective function is minimized**
- **BATCH_SIZE: number of training instances taken into account before the optimizer performs a weight update**
- **OPTIMIZER: Stochastic Gradient Descent (‘SGD‘) – only one training sample/iteration**

```
# dataset 28 x 28 pixels = 784 reshaped
(X_train, y_train), (X_test, y_test) = mnist.load_data()
RESHAPED = 784
X_train = X_train.reshape(60000, RESHAPED)
X_test = X_test.reshape(10000, RESHAPED)
X_train = X_train.astype('float32')
X_test = X_test.astype('float32')

# normalization
X_train /= 255
X_test /= 255

# data output
print(X_train.shape[0], 'train samples')
print(X_test.shape[0], 'test samples')
```

- **Data load shuffled between training and testing set**
- **Data preparation, e.g. X_train is 60000 samples / rows of 28 x 28 pixel values that are reshaped in 60000 x 784 including type specification (i.e. float32)**
- **Data normalization: divide by 255 – the max intensity value to obtain values in range [0,1]**

ANN – MNIST Dataset – A Simple Model

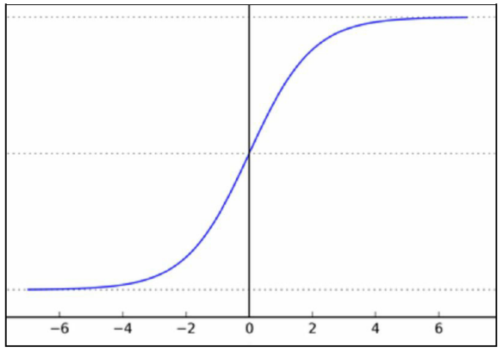
- The Sequential() Keras model is a linear pipeline (aka 'a stack') of various neural network layers including Activation functions of different types (e.g. softmax)

- Dense() represents a fully connected layer used in ANNs that means that each neuron in a layer is connected to all neurons located in the previous layer

- The non-linear Activation function 'softmax' represents a generalization of the sigmoid function – it squashes an n-dimensional vector of arbitrary real values into a n-dimenensional vector of real values in the range of 0 and 1 – here it aggregates 10 answers provided by the Dense layer with 10 neurons

```
# convert vectors to binary matrices of classes
Y_train = np_utils.to_categorical(y_train, NB_CLASSES)
Y_test = np_utils.to_categorical(y_test, NB_CLASSES)

# Simple ANN model
model = Sequential()
model.add(Dense(NB_CLASSES, input_shape=(RESHAPED,)))
model.add(Activation('softmax'))
model.summary()

# Compilation
model.compile(loss='categorical_crossentropy', optimizer=OPTIMIZER, metrics=['accuracy'])

# Fit the model
history = model.fit(X_train, Y_train, batch_size=BATCH_SIZE, epochs=NB_EPOCH, verbose=VERBOSE, validation_split=VALIDATION_SPLIT)

# evaluation
score = model.evaluate(X_test, Y_test, verbose=VERBOSE)
print('Test score:', score[0])
print('Test accuracy:', score[1])
```

$$\mathrm{softmax}(\mathbf{x})_i = \frac{\exp(\mathbf{x}_i)}{\sum_j \exp(\mathbf{x}_j)}$$



- Loss function is a multiclass logarithmic loss: target is *ti,j* and prediction is *pi,j*

$$L_i = -\Sigma_j t_{i,j} \log(p_{i,j})$$

ANN – MNIST Dataset – Job Script

```
#!/bin/bash
#PBS -l nodes=1:ppn=all
#PBS -l walltime=1:0:0
#PBS -N KERAS_MNIST_ANN

module load TensorFlow/1.4.0-intel-2017b-Python-3.6.3
module load Keras/2.1.1-intel-2017b-Python-3.6.3

# make sure Keras is using TensorFlow as backend
export KERAS_BACKEND=tensorflow

export WORKDIR=$VSC_SCRATCH/${PBS_JOBNAME}_${PBS_JOBID}
mkdir -p $WORKDIR
cd $WORKDIR

export OMP_NUM_THREADS=1
python $PBS_O_WORKDIR/KERAS_MNIST_ANN.py

echo "Working directory was $WORKDIR"
```

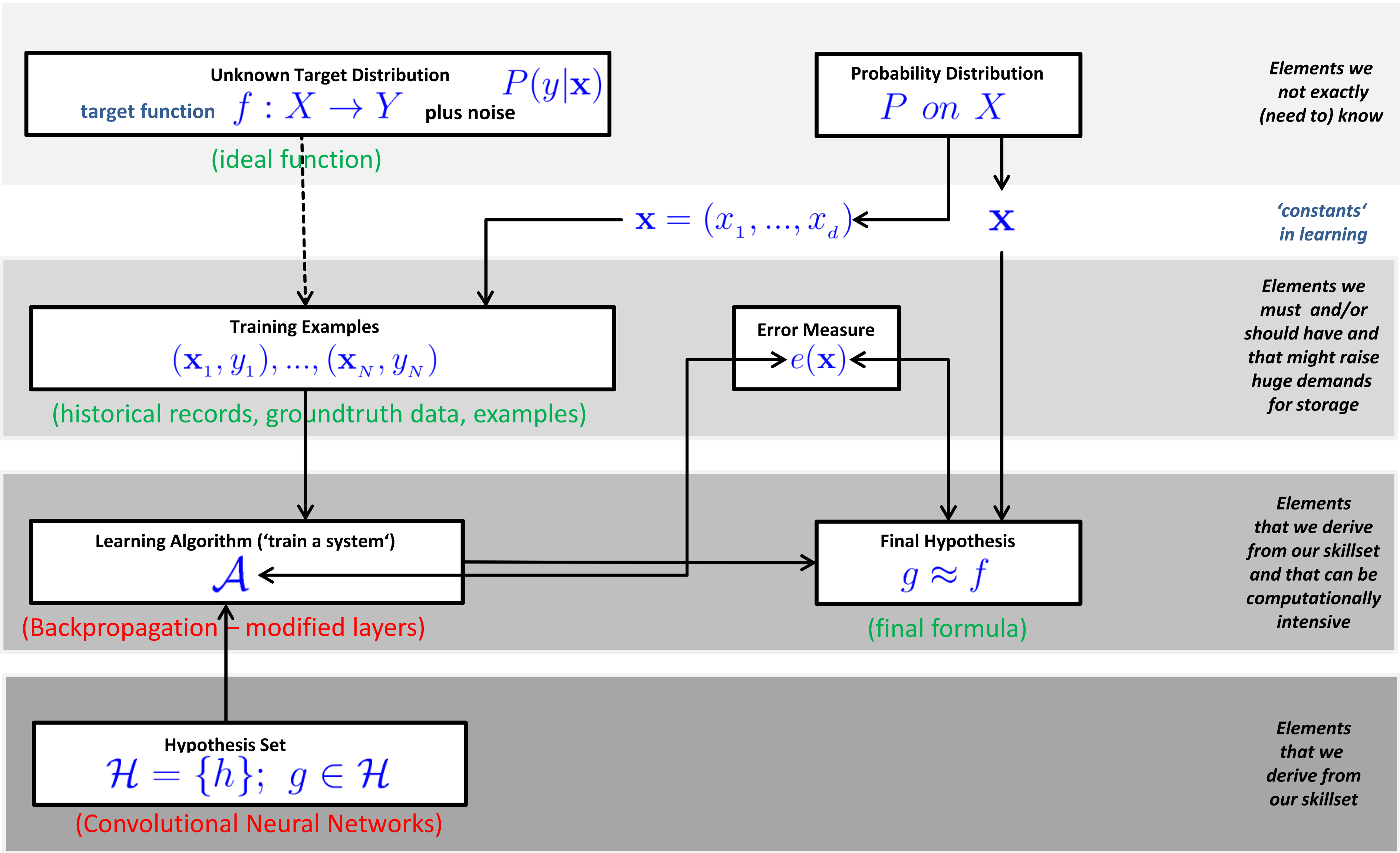
ANN – MNIST Dataset – A Simple Model – Output

```
[vsc42544@gligar03 deeplearning]$ more KERAS_MNIST_ANN.e1179465
Using TensorFlow backend.

[vsc42544@gligar03 deeplearning]$ more KERAS_MNIST_ANN.o1179465
```

```
60000 train samples
10000 test samples
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 10)                7850
_________________________________________________________________
activation_1 (Activation)    (None, 10)                0
=================================================================
Total params: 7,850
Trainable params: 7,850
Non-trainable params: 0
_________________________________________________________________
Train on 48000 samples, validate on 12000 samples
```

```
[vsc42544@gligar03 deeplearning]$ tail KERAS_MNIST_ANN.o1179465
48000/48000 [==============================] - 1s 12us/step - loss: 0.2760 - acc: 0.9227 - val_loss: 0.2747 - val_acc: 0.9234

   32/10000 [..............................] - ETA: 0s
 3104/10000 [========>.....................] - ETA: 0s
 6208/10000 [=================>............] - ETA: 0s
 9344/10000 [===========================>..] - ETA: 0s
10000/10000 [==============================] - 0s 16us/step
Test score: 0.277443544486
Test accuracy: 0.9221
Working directory was /user/scratch/gent/vsc425/vsc42544/KERAS_MNIST_ANN_1179465.master19.golett.gent.vsc
```

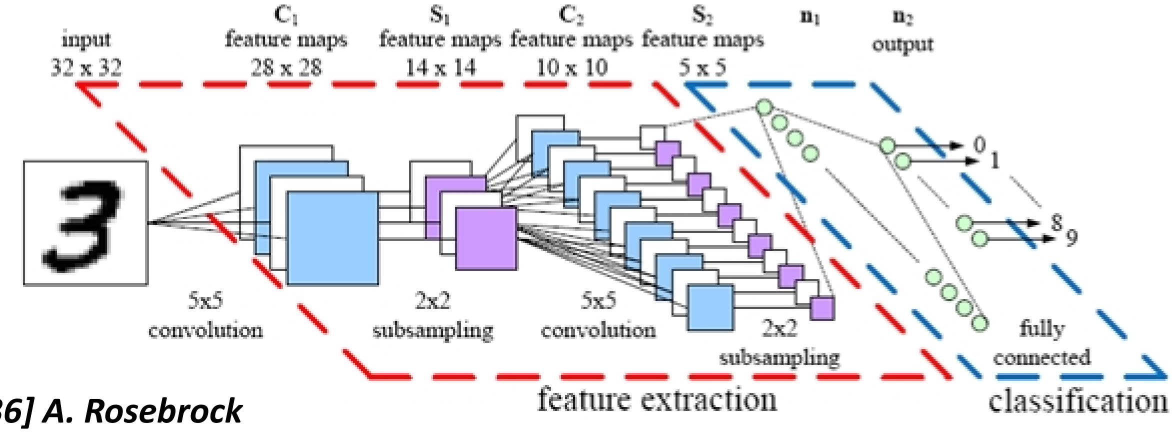
Solution Tools: Convolutional Networks Learning Model



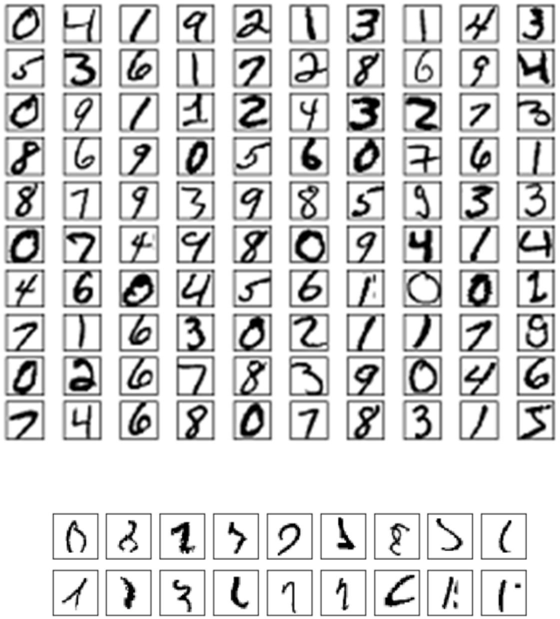
CNNs – Basic Principles

- **Convolutional Neural Networks (CNNs/ConvNets) implement a connectivity pattner between neurons inspired by the animal visual cortex and use several types of layers (convolution, pooling)**
- **CNN key principles are local receptive fields, shared weights, and pooling (or down/sub-sampling)**
- **CNNs are optimized to take advantage of the spatial structure of the data**

- Simple application example
  - MNIST database written characters
  - Use CNN architecture with different layers
  - Goal: automatic classification of characters



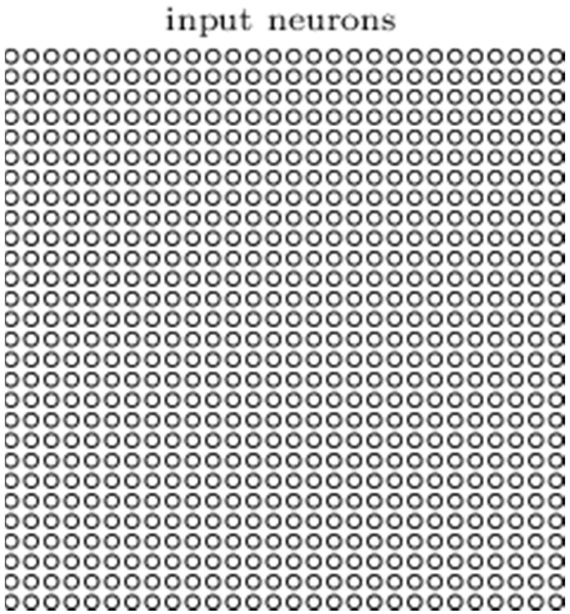
[36] A. Rosebrock



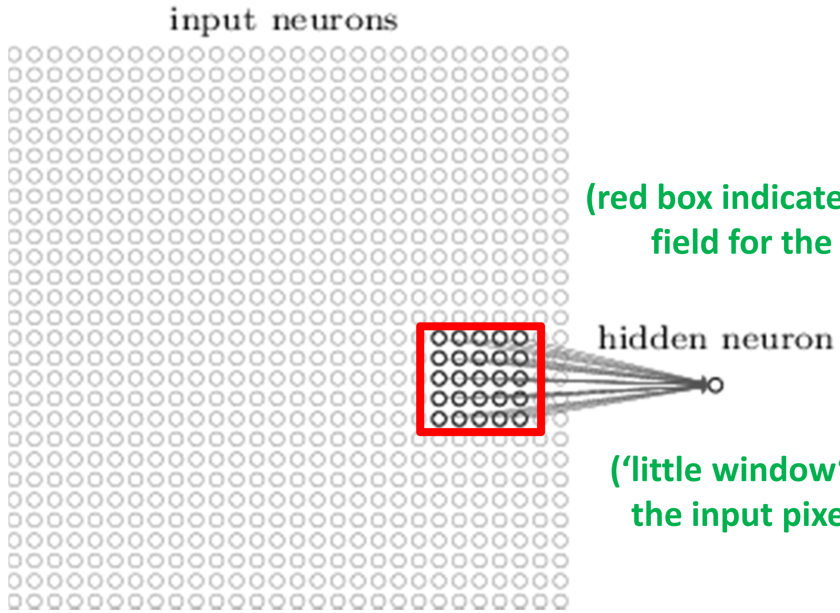
[35] M. Nielsen

CNNs – Principle Local Receptive Fields

- MNIST dataset example
  - 28 * 28 pixels modeled as square of neurons in a convolutional net
  - Values correspond to the 28 * 28 pixel intensities as inputs



(28 * 28 pixel image)



(red box indicate the local receptive field for the hidden neuron)

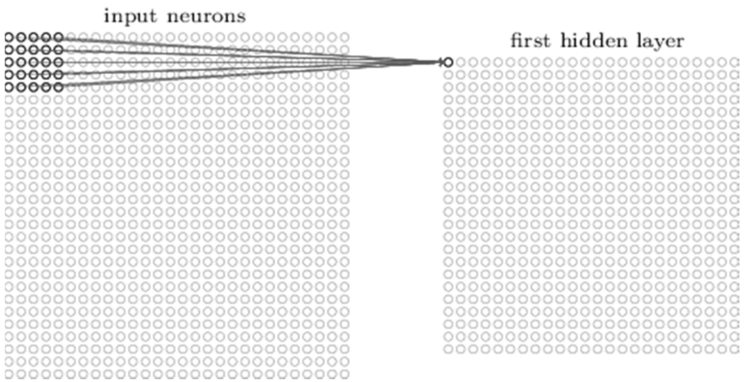
('little window' on the input pixels)

(5 * 5 local connectivity)

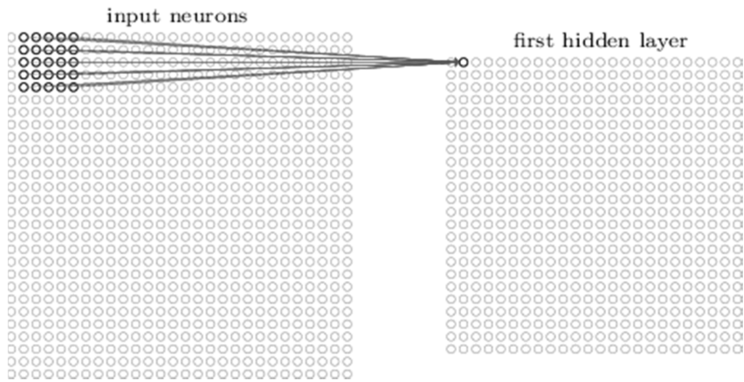
[35] M. Nielsen

CNNs – Principle Local Receptive Fields & Sliding

- MNIST database example
  - Apply stride length = 1
  - Different configurations possible and depends on application goals
  - Creates ‘feature map‘ of 24 * 24 neurons (hidden layer)



(28 * 28 pixel image) **(24 * 24 feature map)**

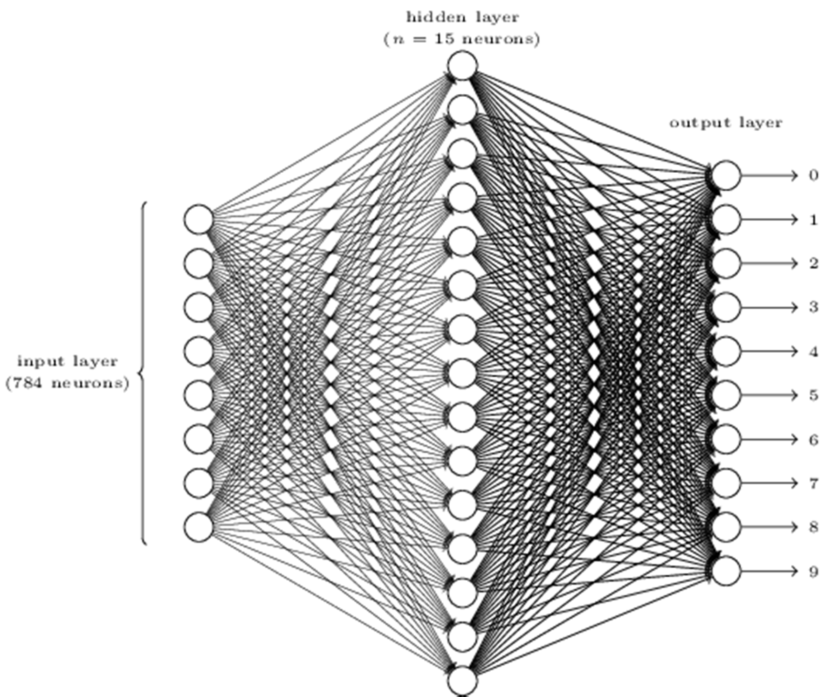


(28 * 28 pixel image) **(24 * 24 feature map)**

[35] M. Nielsen

CNNs –Example with an ANN with risk of Overfitting

- MNIST database example
  - CNN: e.g. 20 feature maps with 5 * 5 (+bias) = 520 weights to learn
  - Apply ANN that is fully connected between neurons
  - ANN: fully connected first layer with 28 * 28 = 784 input neurons
  - ANN: e.g. 15 hidden neurons with 784 * 15 = 11760 weights to learn

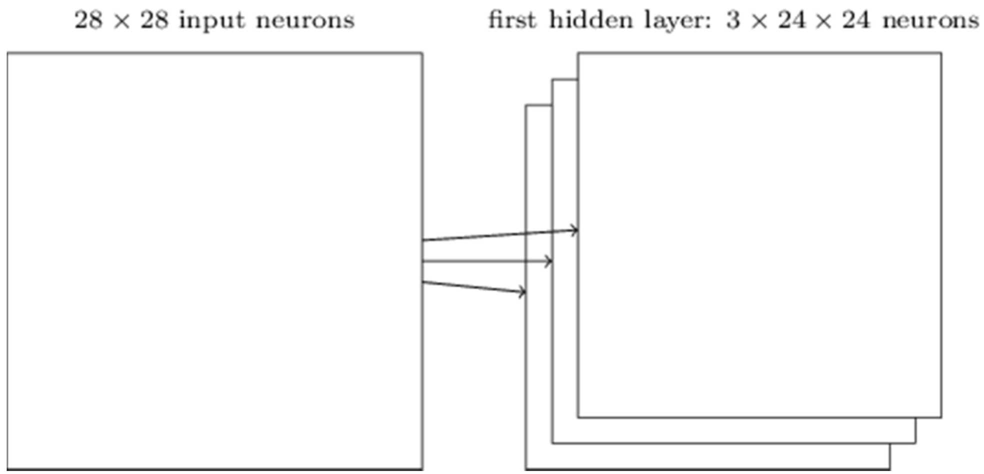


(eventually lead to overfitting and much computing time)

[35] M. Nielsen

CNNs – Principle Shared Weights & Feature Maps

- Approach
  - CNNs use same shared weights for each of the 24 * 24 hidden neurons
  - Goals: significant reduction of number of parameters (prevent overfitting)
  - Example: 5 * 5 receptive field → 25 shared weights + shared bias
- Feature Map
  - Detects one local feature
  - E.g. 3: each feature map is defined by a set of 5 * 5 shared weights and a single shared bias leading to 24 * 24
  - Goal: The network can now detect 3 different kind of features **(many more in practice)**



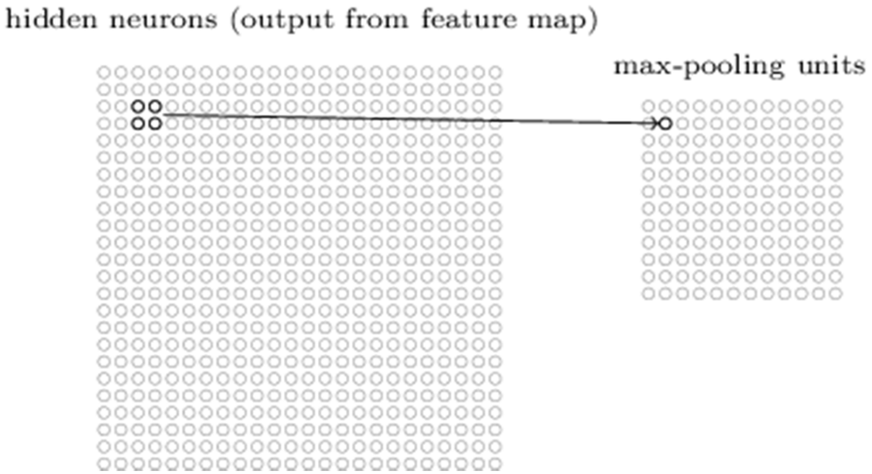
(shared weights are also known to define a kernel or filter)

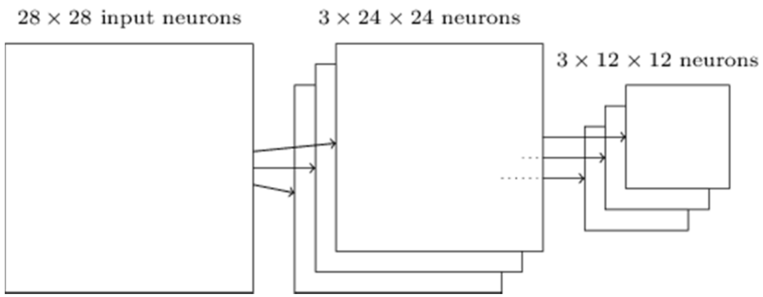
  - Benefit: learned feature being detectable across the entire image

[35] M. Nielsen

CNNs – Principle of Pooling

- ‘Downsampling‘ Approach
  - Usually applied directly after convolutional layers
  - Idea is to simplify the information in the output from the convolution
  - Take each feature map output from the convolutional layer and generate a condensed feature map
  - E.g. Pooling with 2 * 2 neurons using ‘max-pooling‘
  - Max-Pooling outputs the maximum activation in the 2 * 2 region

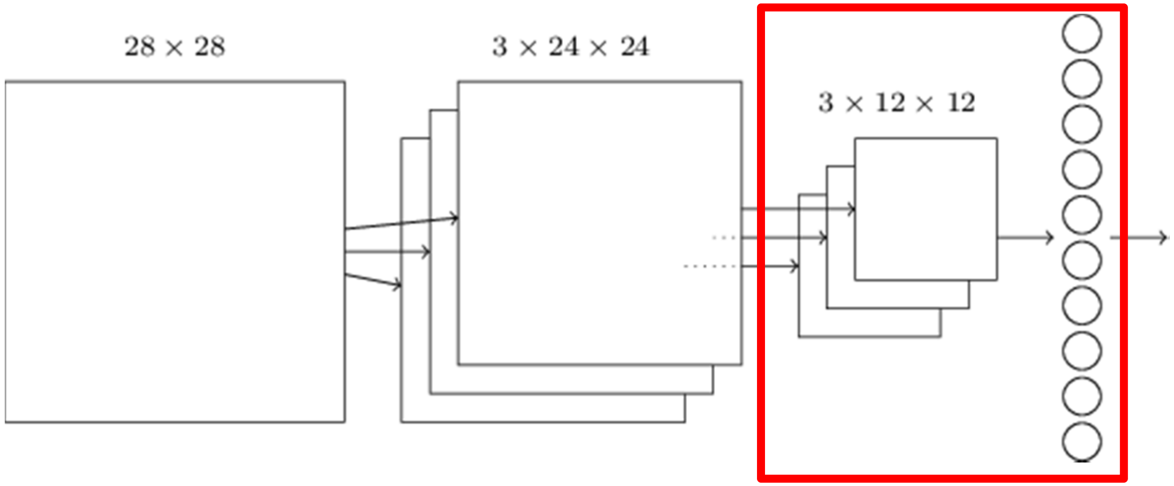




[35] M. Nielsen

CNN – Application Example MNIST

- MNIST database example
  - Full CNN with the addition of output neurons per class of digits
  - Apply 'fully connected layer': layer connects every neuron from the max-pooling outcome layer to every neuron of the 10 out neurons
  - Train with backpropagation algorithm (gradient descent), only small modifications for new layers



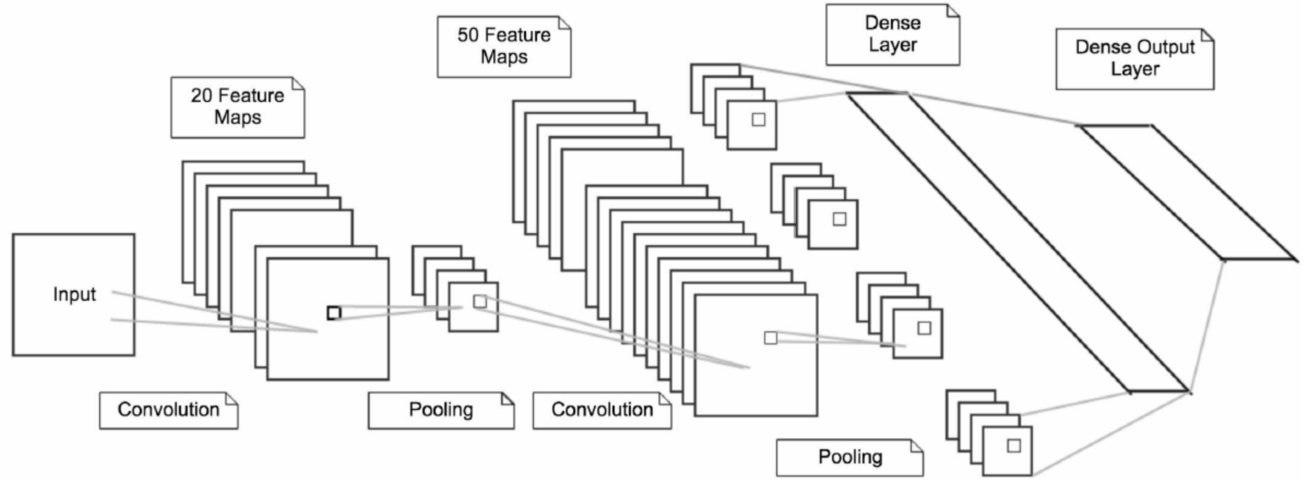
  - Approach works, except for some bad training and test examples

(another indicator that even with cutting edge technology machine learning never achieves 100% performance)

[35] M. Nielsen

MNIST Dataset – CNN Model

```python
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers.core import Dense, Activation, Flatten
from keras.utils import np_utils
from keras import backend as K
from keras.layers.convolutional import Convolution2D, MaxPooling2D
from keras.optimizers import SGD, RMSprop, Adam

# model
class CNN:
  @staticmethod
  def build(input_shape, classes):
    model = Sequential()
    model.add(Convolution2D(20, kernel_size=5, padding="same", input_shape=input_shape))
    model.add(Activation("relu"))
    model.add(MaxPooling2D(pool_size=(2,2), strides=(2,2)))
    model.add(Convolution2D(50, kernel_size=5, border_mode="same"))
    model.add(Activation("relu"))
    model.add(MaxPooling2D(pool_size=(2,2), strides=(2,2)))
    model.add(Flatten())
    model.add(Dense(500))
    model.add(Activation("relu"))
    model.add(Dense(classes))
    model.add(Activation("softmax"
    return model
```



[37] A. Gulli et al.

MNIST Dataset – CNN Python Script

```
# parameters
NB_CLASSES = 10
NB_EPOCH = 20
BATCH_SIZE = 128
VERBOSE = 1
OPTIMIZER = 'Adam'
VALIDATION_SPLIT = 0.2
IMG_ROWS, IMG_COLS = 28, 28
INPUT_SHAPE = (1, IMG_ROWS, IMG_COLS)

# dataset 28 x 28 pixels
(X_train, y_train), (X_test, y_test) = mnist.load_data()
K.set_image_dim_ordering("th")
X_train = X_train.astype('float32')
X_test = X_test.astype('float32')

# normalization
X_train /= 255
X_test /= 255

# input convnet
X_train = X_train[:, np.newaxis, :, :]
X_test = X_test[:, np.newaxis, :, :]

# data output
print(X_train.shape[0], 'train samples')
print(X_test.shape[0], 'test samples')

# convert vectors to binary matrices of classes
Y_train = np_utils.to_categorical(y_train, NB_CLASSES)
Y_test = np_utils.to_categorical(y_test, NB_CLASSES)

# Simple CNN model
model = CNN.build(input_shape=INPUT_SHAPE, classes=NB_CLASSES)

# Compilation
model.compile(loss='categorical_crossentropy', optimizer=OPTIMIZER, metrics=['accuracy'])

# Fit the model
history = model.fit(X_train, Y_train, batch_size=BATCH_SIZE, epochs=NB_EPOCH, verbose=VERBOSE, validation_split=VALIDATION_SPLIT)

# evaluation
score = model.evaluate(X_test, Y_test, verbose=VERBOSE)
print('Test score:', score[0])
print('Test accuracy:', score[1])
```

- **OPTIMIZER: Adam - advanced optimization technique that includes the concept of a momentum (a certain velocity component) in addition to the acceleration component of Stochastic Gradient Descent (SGD)**
- **Adam computes individual adaptive learning rates for different parameters from estimates of first and second moments of the gradients**
- **Adam enables faster convergence at the cost of more computation and is currently recommended as the default algorithm to use (or SGD + Nesterov Momentum)**

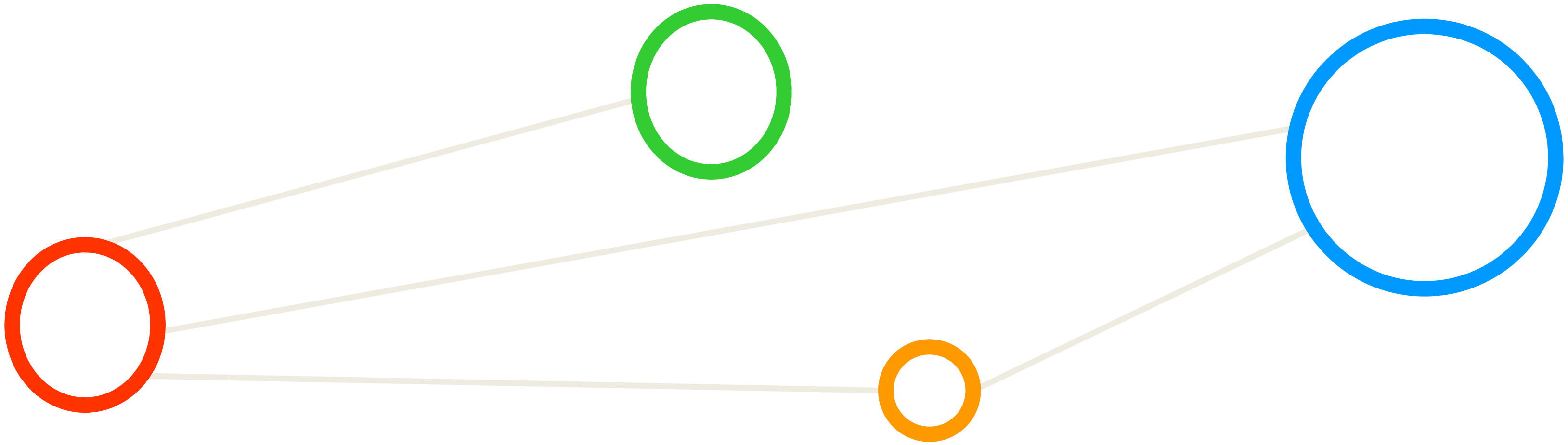
[38] D. Kingma et al., 'Adam: A Method for Stochastic Optimization'

MNIST Dataset – CNN Model – Output

```
[vsc42544@gligar01 deeplearning]$ head KERAS_MNIST_CNN.o1179880
60000 train samples
10000 test samples
Train on 48000 samples, validate on 12000 samples
Epoch 1/20

  128/48000 [..............................] - ETA: 10:06 - loss: 2.2997 - acc: 0.1250
  256/48000 [..............................] - ETA: 7:46 - loss: 2.2578 - acc: 0.1992
  384/48000 [..............................] - ETA: 6:58 - loss: 2.2127 - acc: 0.2083
  512/48000 [..............................] - ETA: 6:35 - loss: 2.1632 - acc: 0.2598
  640/48000 [..............................] - ETA: 6:20 - loss: 2.0934 - acc: 0.3234
```

```
[vsc42544@gligar01 deeplearning]$ tail KERAS_MNIST_CNN.o1179880
 9824/10000 [============================>.] - ETA: 0s
 9856/10000 [============================>.] - ETA: 0s
 9888/10000 [============================>.] - ETA: 0s
 9920/10000 [============================>.] - ETA: 0s
 9952/10000 [============================>.] - ETA: 0s
 9984/10000 [============================>.] - ETA: 0s
10000/10000 [==============================] - 41s 4ms/step
Test score: 0.0483192791523
Test accuracy: 0.99
Working directory was /user/scratch/gent/vsc425/vsc42544/KERAS_MNIST_CNN_1179880.master19.golett.gent.vsc
```

Acknowledgements

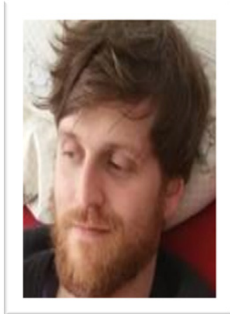


Acknowledgements – Membery of my HPDP Research Group

PD Dr.
G. Cavallaro

Senior PhD
Student A.S. Memon

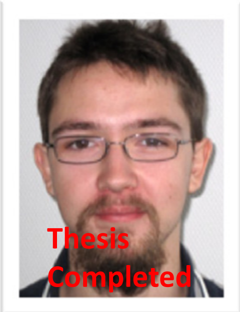
Senior PhD
Student M.S. Memon



PhD Student
E. Erlingsson

PhD Student
C. Bakarat

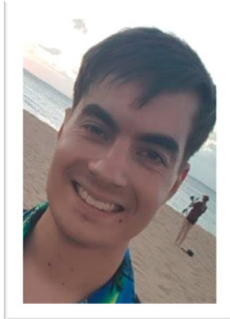
Dr. M. Goetz
(now KIT)



MSc M.
Richerzhagen

MSc
P. Glock
(now INM-1)

MSc
C. Bodenstein
(now Soccerwatch.tv)



MSc Student
G.S. Guðmundsson
(Landsverkjun)

Slides Available at http://www.morrisriedel.de/talks