

Parallel & Scalable Machine Learning

Machine Learning Models using High Performance Computing

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

Parallel Machine Learning & Deep Learning Driven by HPC

July 3rd, 2018

4th International Summer School on Big Data & Machine Learning, Leipzig, Germany



UNIVERSITY OF ICELAND
SCHOOL OF ENGINEERING AND NATURAL SCIENCES

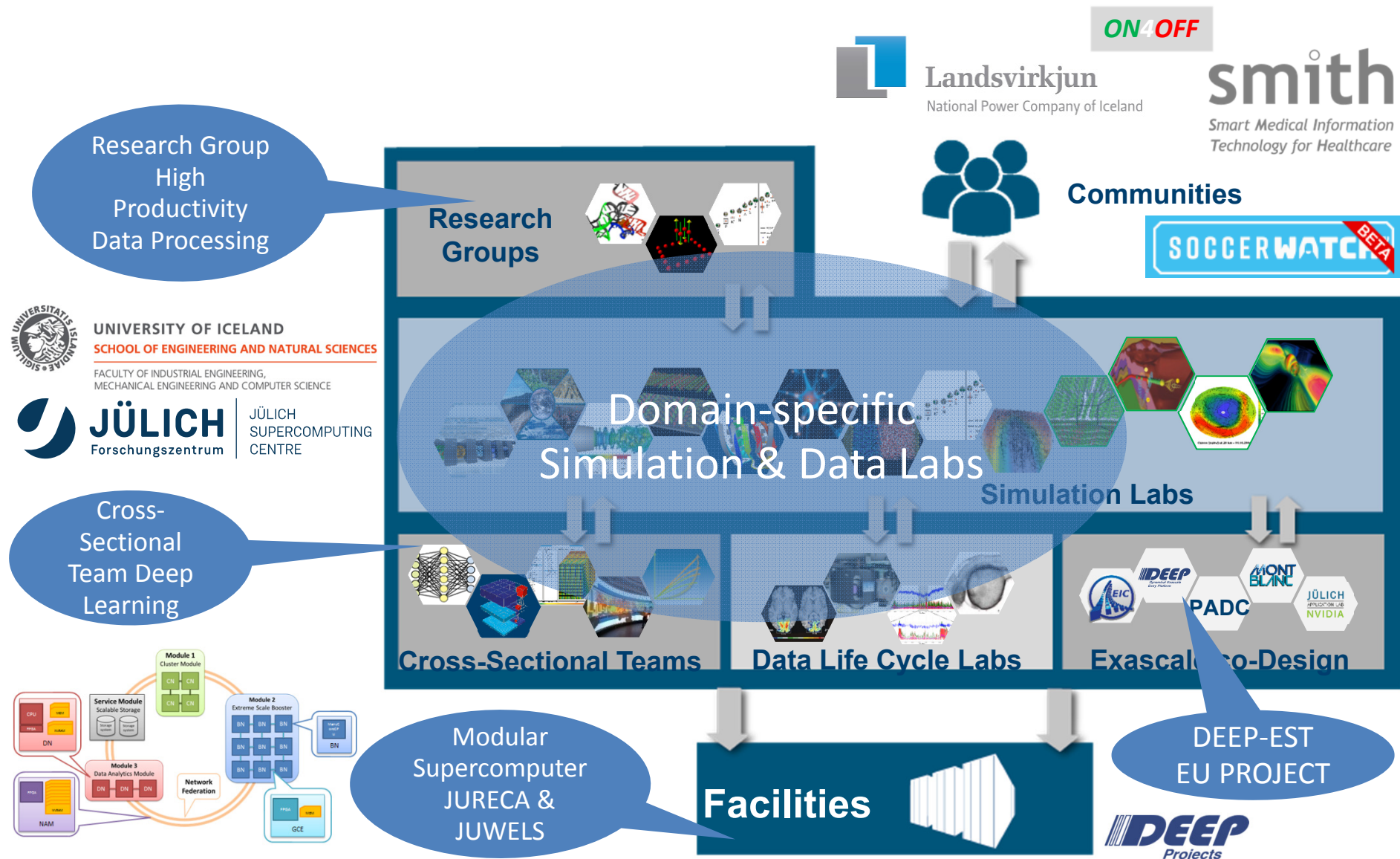
FACULTY OF INDUSTRIAL ENGINEERING,
MECHANICAL ENGINEERING AND COMPUTER SCIENCE



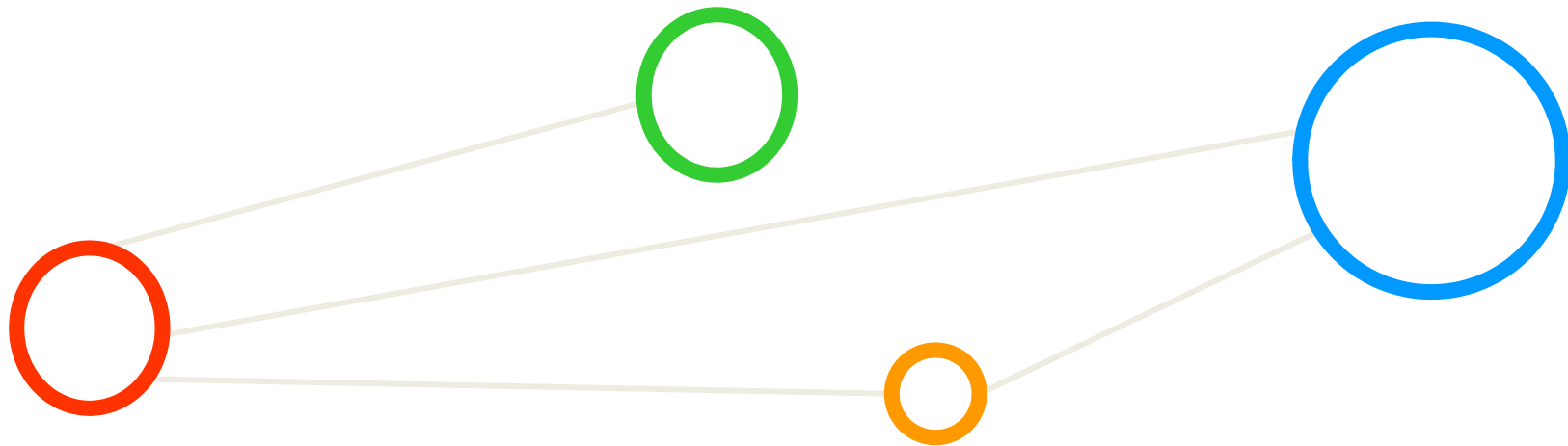
HELMHOLTZ
RESEARCH FOR GRAND CHALLENGES



Juelich Supercomputing Centre: Intertwine Data Analysis/HPC



Outline



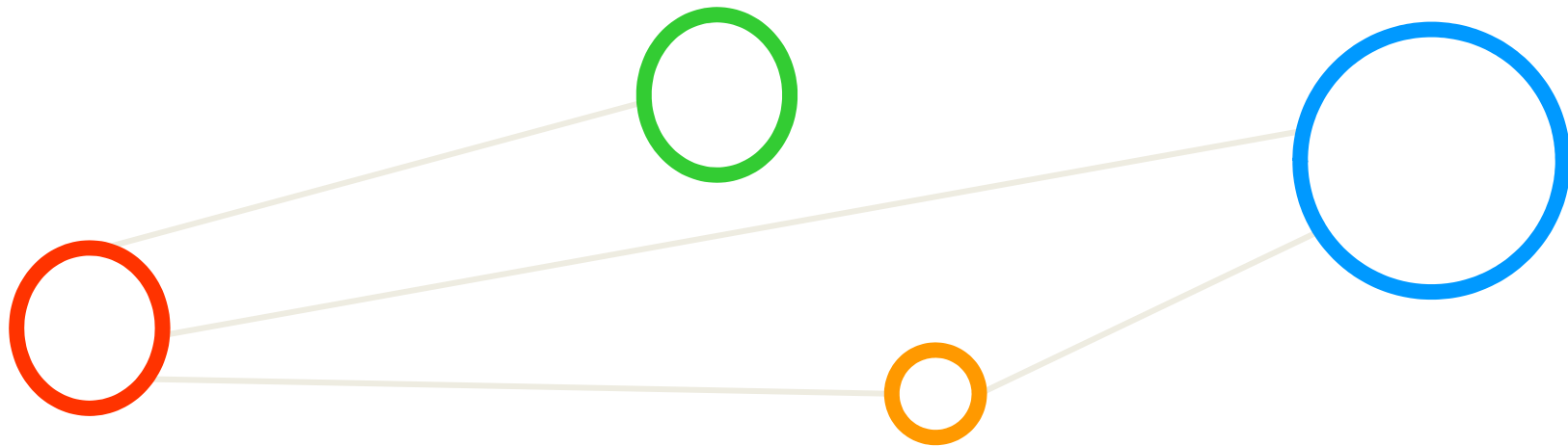
Outline

- Traditional Machine Learning Models
 - Big Data & Machine Learning Introduction
 - Supervised & Unsupervised Learning
 - Supervised Learning using parallel SVMs
 - Parallelization Benefits using Cross-Validation
 - Unsupervised Learning using parallel DBSCAN
- Selected Deep Learning Models
 - Short Introduction to Deep Learning
 - Role of Accelerators & GPGPUs
 - Comparisons Machine Learning & Deep Learning
 - Convolutional Neural Networks (CNNs) Models
 - Long Short-Term Memory (LSTM) Networks
- Open Challenges & Summary
- Appendix A – E: Selected In-depth Topics

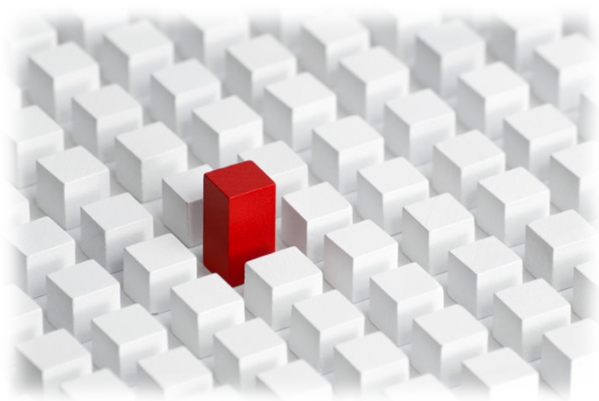
- Machine Learning requires a full university course covering topics beyond modeling & algorithms like statistical learning theory, regularization & validation techniques
- Using High Performance Computing (HPC) adds another level of complexity requiring a full HPC university course



Traditional Machine Learning Models



'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

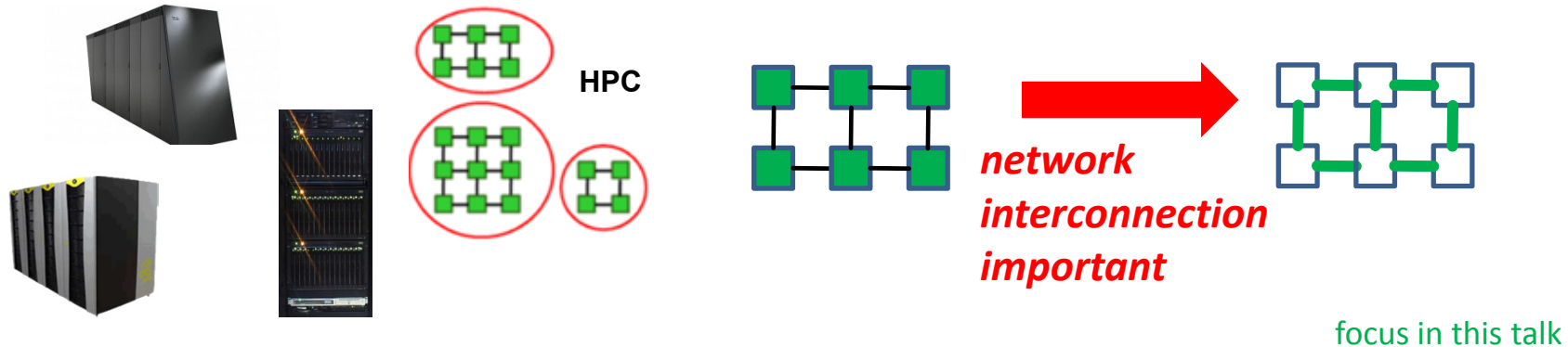
modified from [1] Introduction to Data Mining

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

➤ Link to talk this morning by J. Bungartz – HPC Meets Big Data: Analytics & HPC examples

Understanding High Performance Computing

- High Performance Computing (HPC) is based on computing resources that enable the efficient use of parallel computing techniques through specific support with dedicated hardware such as high performance cpu/core interconnections.



- High Throughput Computing (HTC) is based on commonly available computing resources such as commodity PCs and small clusters that enable the execution of 'farming jobs' without providing a high performance interconnection between the cpu/cores.



➤ [Link to talk this morning by J. Bungartz – HPC Meets Big Data: What is HPC & parallel efficiency](#)

PRACE as Persistent pan-European HPC Infrastructure

Mission:

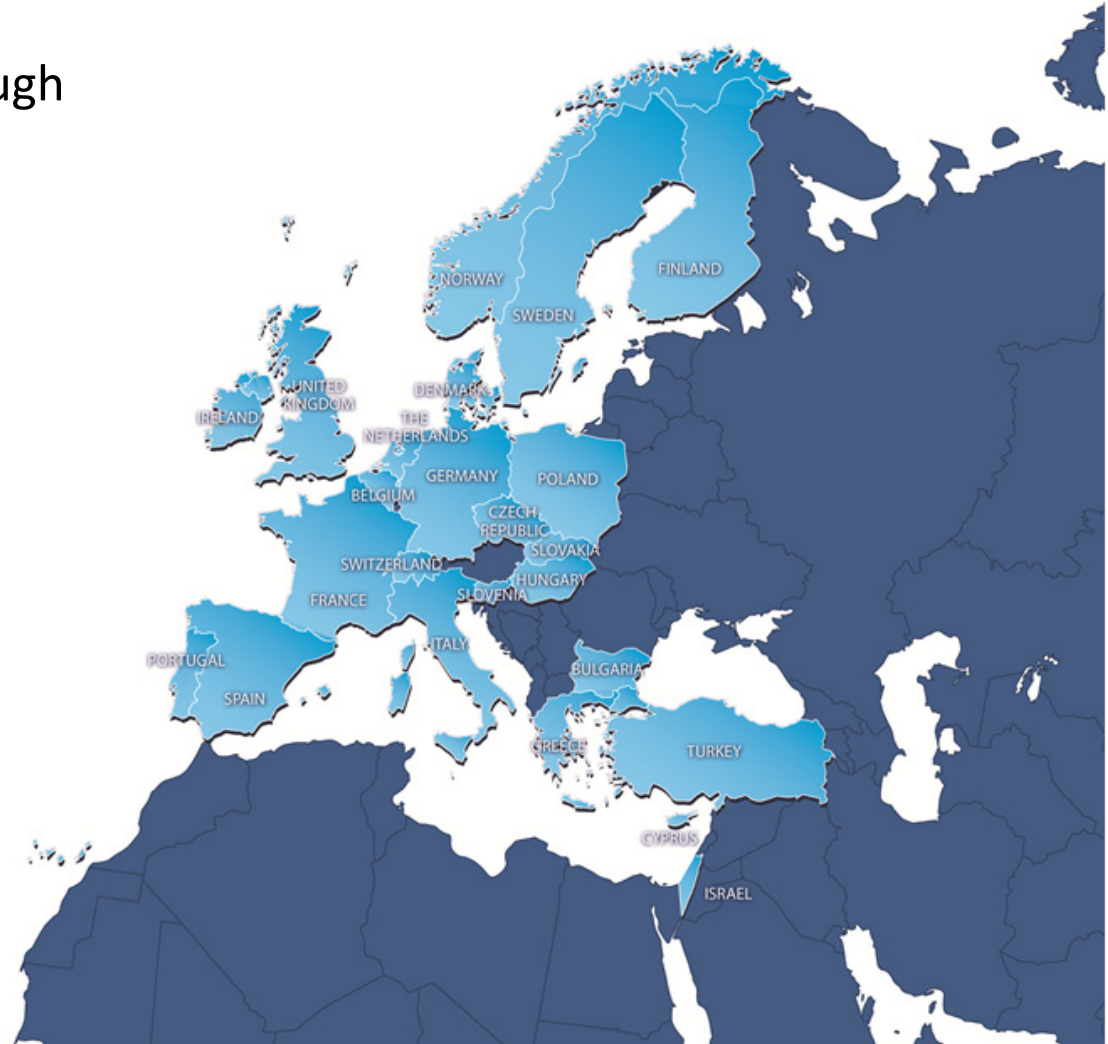
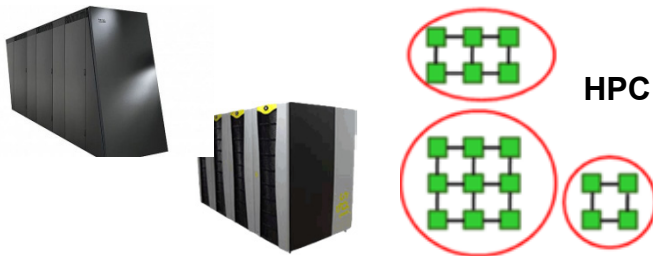
enabling world-class science through
large scale simulations

Offering:

HPC resources on leading edge
capability systems

Resource award:

through a single and fair pan-
European peer review process for
open research



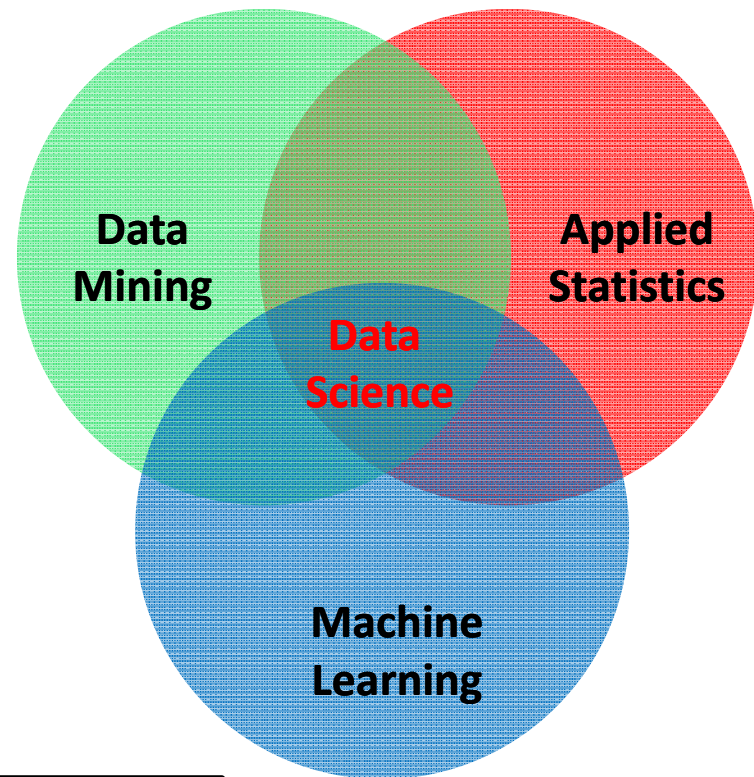
Before using HPC: Machine Learning Prerequisites

1. Some pattern exists
2. No exact mathematical formula

3. Data exists

- Idea ‘**Learning from Data**’ shared with a wide variety of other disciplines
 - E.g. signal processing, data mining, 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’)



➤ Link to talk by U. Leser – Web-Scale Domain-Specific Information Extraction: Data Science?!

➤ Machine Learning is introduced in Appendix A of the slides with a simple classification example

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)



[About](#) – [Submit Data](#) – [Projects](#) – [Software](#) – [Contact](#)

[2] PANGAEA data collection

[40] M. Goetz, PhD Thesis, University of Iceland

(examples for learning & comparisons)

The image shows the UCI Machine Learning Repository logo, which is a stylized dinosaur. Below the logo is a table of data sets. The table has columns for Name, Data Types, Default Task, Attribute Types, # Instances, # Attributes, and Year. The table lists 295 data sets, with the first 10 shown in the image.

Name	Data Types	Default Task	Attribute Types	# Instances	# Attributes	Year
Abalone	Multivariate	Classification	Categorical, Integer, Real	4177	8	1995
Adult	Multivariate	Classification	Categorical, Integer	48842	14	1996
Annealing	Multivariate	Classification	Categorical, Integer, Real	798	38	
Anonymous Microsoft Web Data		Recommender-Systems	Categorical	37711	294	1998
Arrhythmia	Multivariate	Classification	Categorical, Integer, Real	452	279	1998
Artificial Characters	Multivariate	Classification	Categorical, Integer, Real	6000	7	1992
Audiology (Original)	Multivariate	Classification	Categorical	226		1987
Audiology (Standardized)	Multivariate	Classification	Categorical	226	69	1992
Auto MPG	Multivariate	Regression	Categorical, Real	398	8	1993
Automobile	Multivariate	Regression	Categorical, Integer, Real	205	26	1987

[3] UCI Machine Learning Repository

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)

➤ This invited lecture focus on supervised and unsupervised learning applications & examples

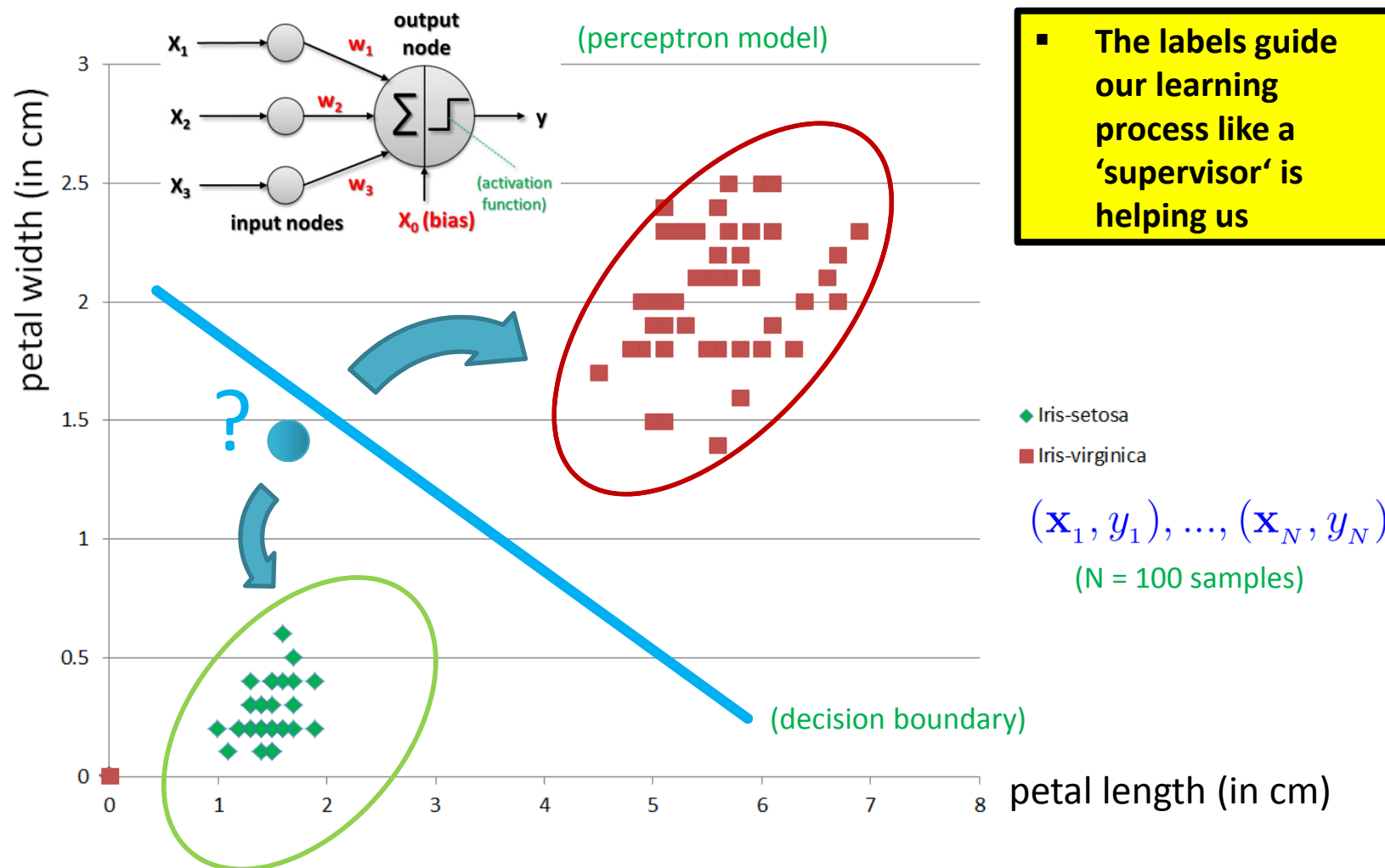
Learning Approaches – Supervised Learning

- Each observation of the predictor measurement(s) has an associated response measurement:
 - Input $\mathbf{x} = x_1, \dots, x_d$
 - Output $y_i, i = 1, \dots, n$
 - 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

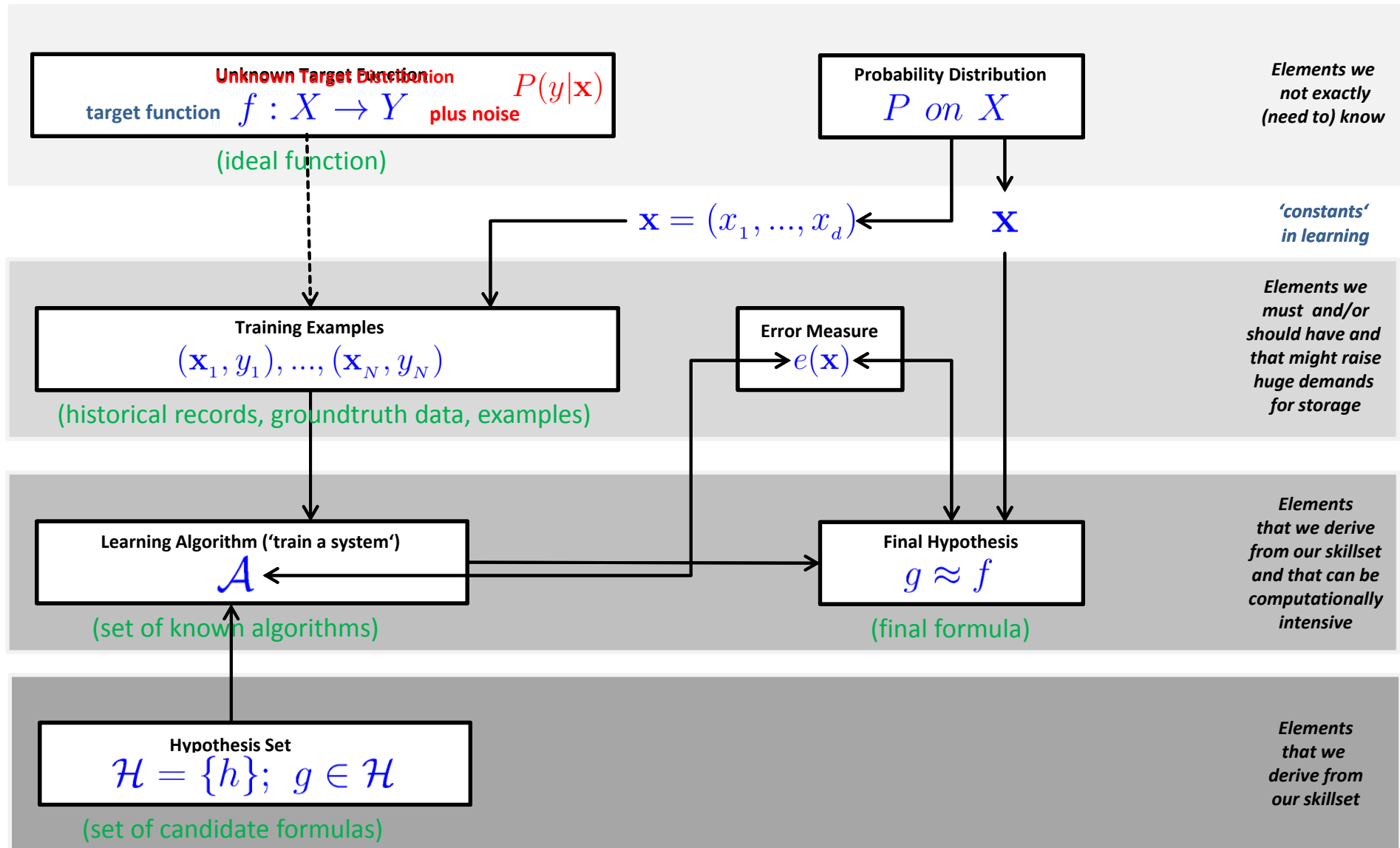
Learning Approaches – Supervised Learning Example



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

➤ Full example of this linear perceptron learning model is introduced in Appendix A of the slides

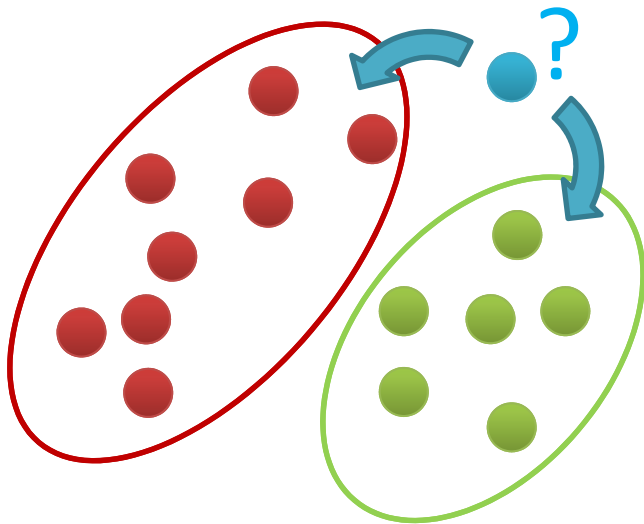
Supervised Learning – Overview & Summary



Methods Overview – Advanced Example

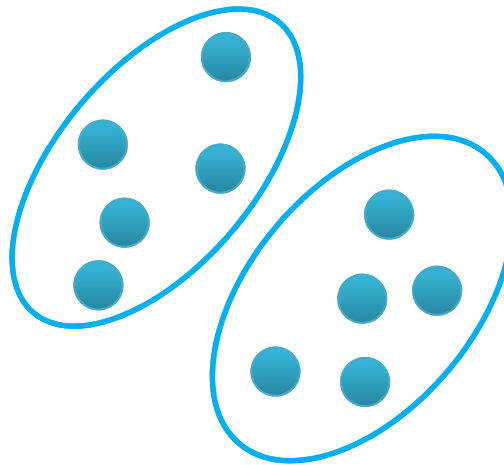
- Statistical data mining methods can be roughly categorized in classification, clustering, or regression augmented with various techniques for data exploration, selection, or reduction

Classification



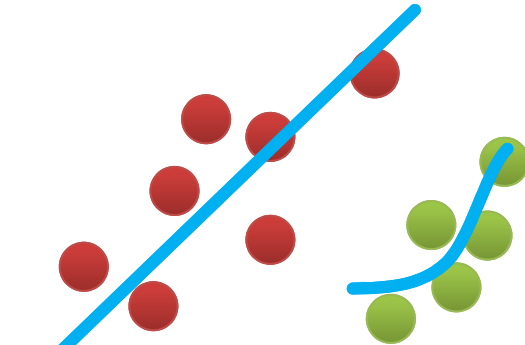
- Groups of data exist
- New data classified to existing groups

Clustering



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

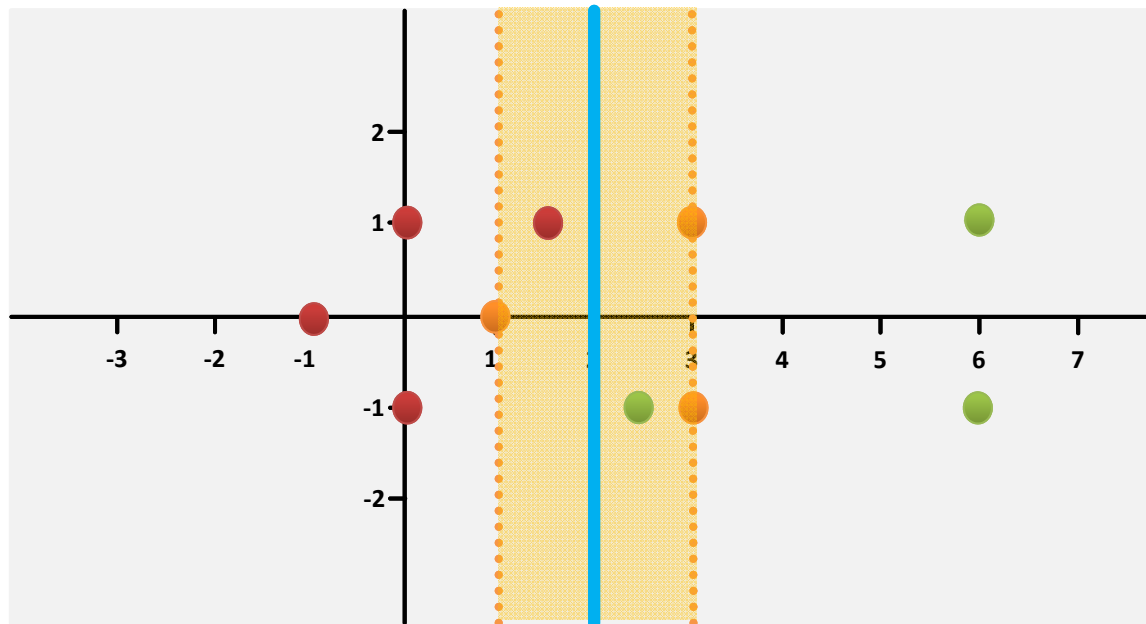
Regression



- Identify a line with a certain slope describing the data

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 C of the slideset

Term Support Vector Machines Refined

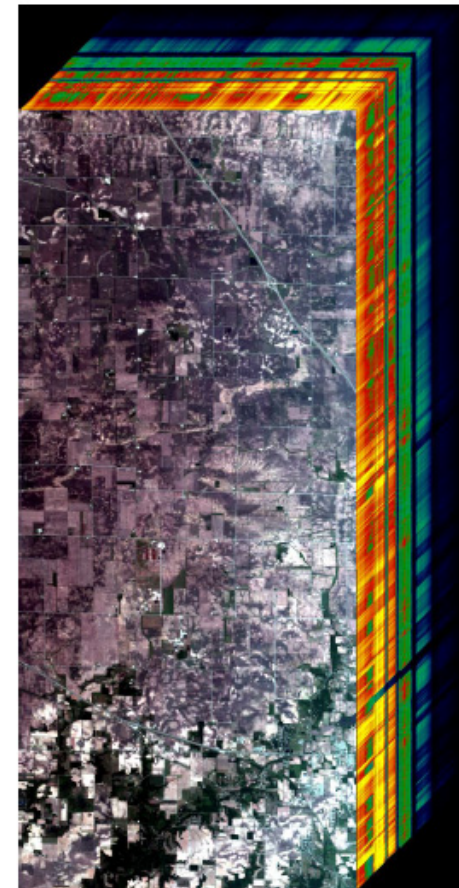
- Support Vector Machines (SVMs) are a classification technique developed ~1990
- SVMs perform well in many settings & are considered as one of the best 'out of the box classifiers'

[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 separable 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**;

Remote Sensing Application Example – Indian Pines Dataset

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



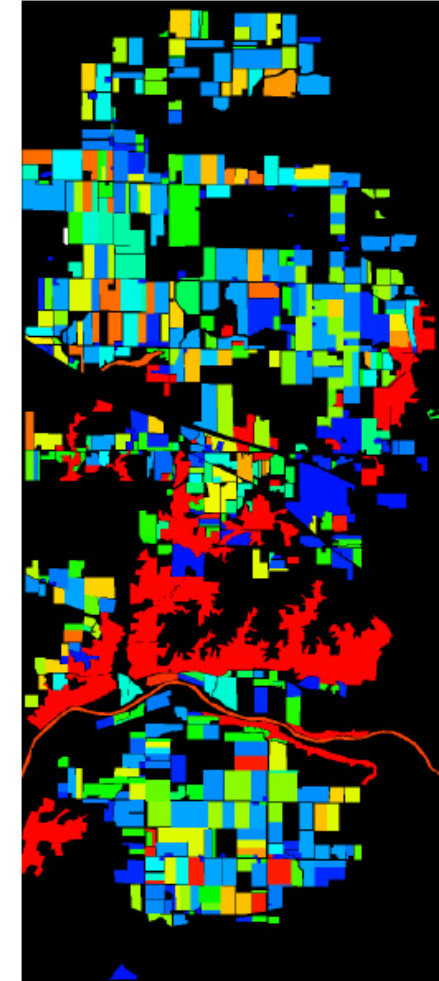
[39] Indian Pines dataset

Indian Pines Dataset – Preprocessing

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

Class				Class			
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-CleanTill-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



Publicly Available Datasets – Open Data

■ *Indian Pines Dataset Raw and Processed*



[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)

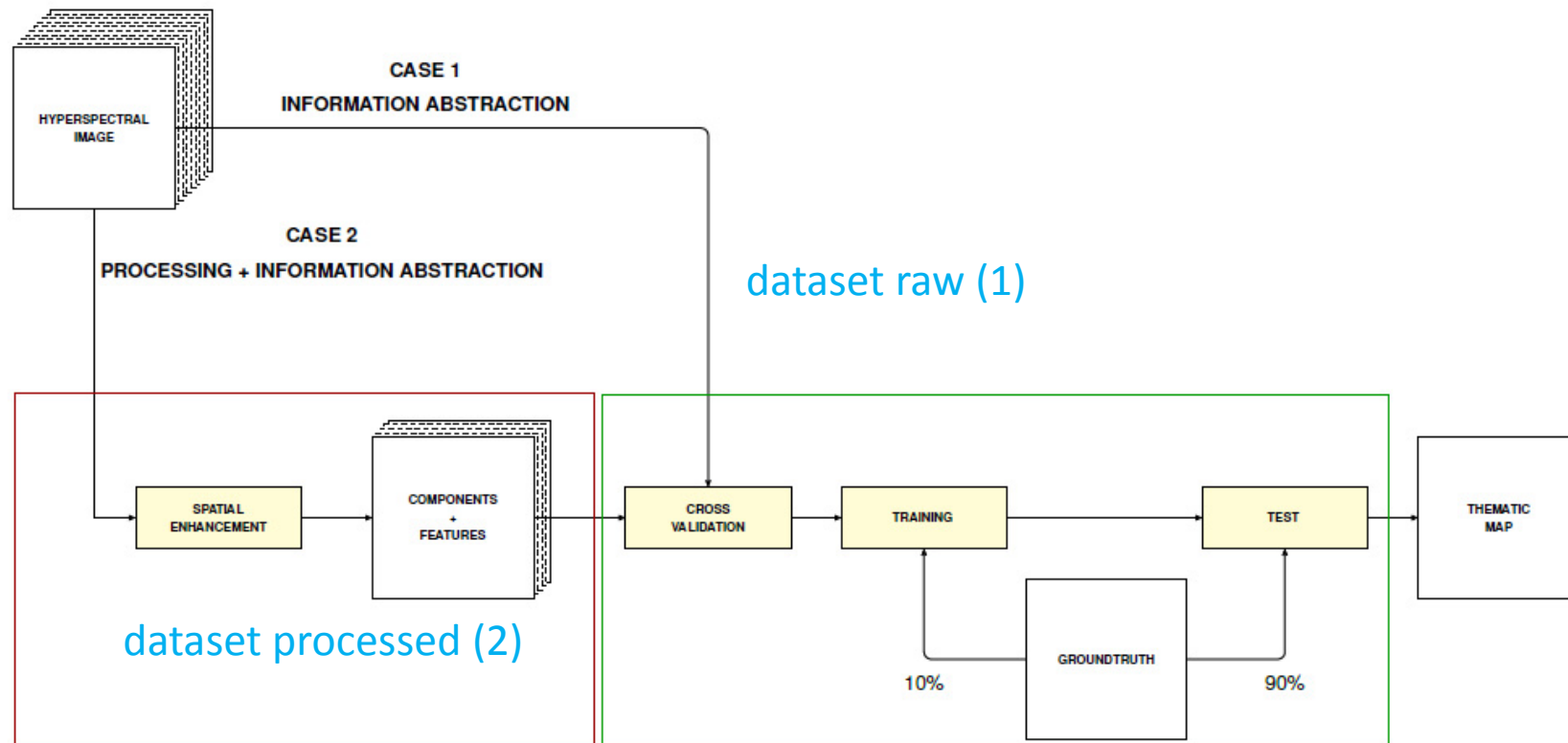
PID: [11304/7e8eec8e-ad61-11e4-ac7e-860aa0063d1f](https://hdl.handle.net/11304/7e8eec8e-ad61-11e4-ac7e-860aa0063d1f) [Copy](#)



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/11304/7e8eec8e-ad61-11e4-ac7e-860aa0063d1f Type ePIC_PID
Publisher	https://b2share.eudat.eu
Language	en

Indian Pines – ‘pure’ Big Data vs. Feature Engineering



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

Review of Open Source Parallel SVM Implementations

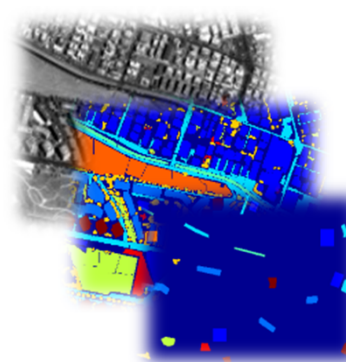
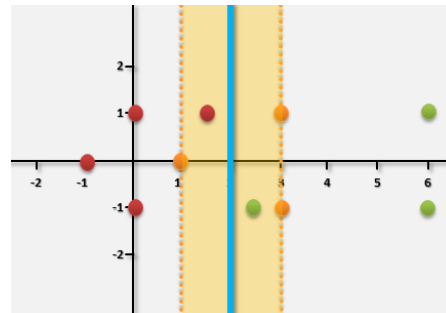
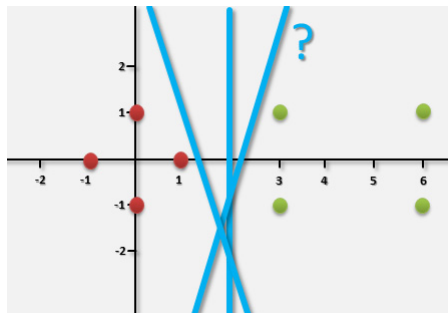
Technology	Platform Approach	Analysis
Apache Mahout	Java; Hadoop	No parallelization strategy for SVMs
Apache Spark/MLlib	Java; Spark	Parallel linear SVMs (no multi-class)
Twister/ParallelSVM	Java; Twister; Hadoop 1.0	Parallel SVMs, open source; developer version 0.9 beta
scikit-learn	Python	No parallelization 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

➤ Work in progress: Recent related work analysis reveals no new results; evaluations pending...

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



[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*

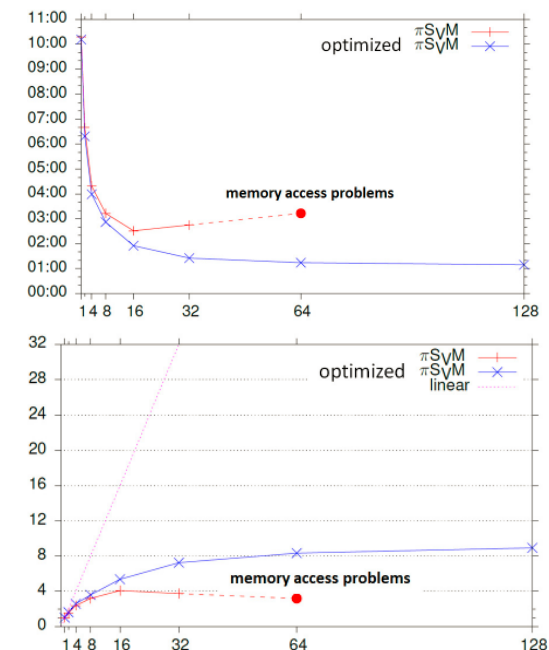
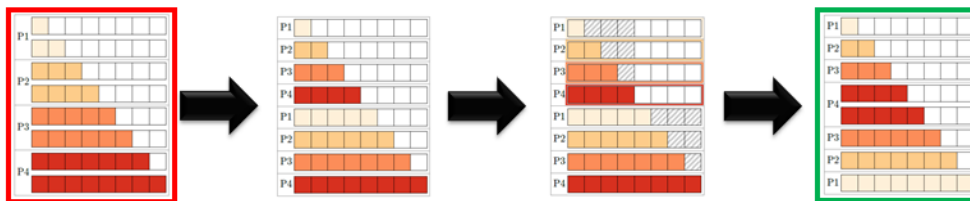
➤ Open source code publicly available at: <https://github.com/mricherzhagen/pisvm>

Parallel SVM with MPI Technique – piSVM Implementation



[19] piSVM on SourceForge, 2008

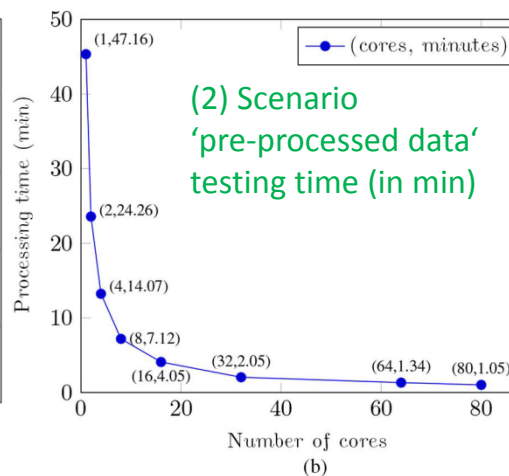
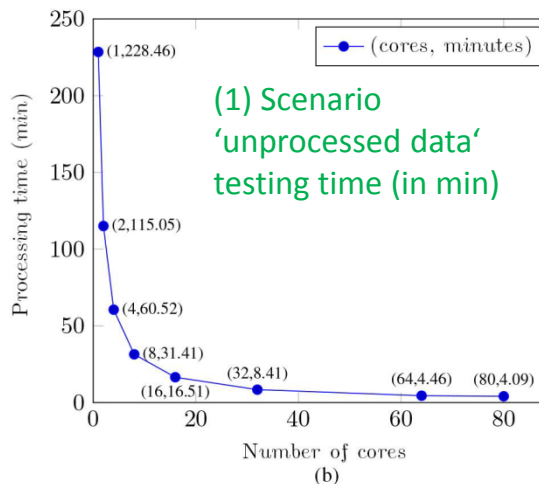
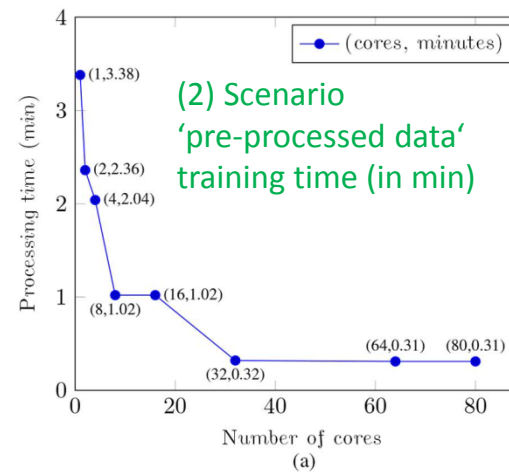
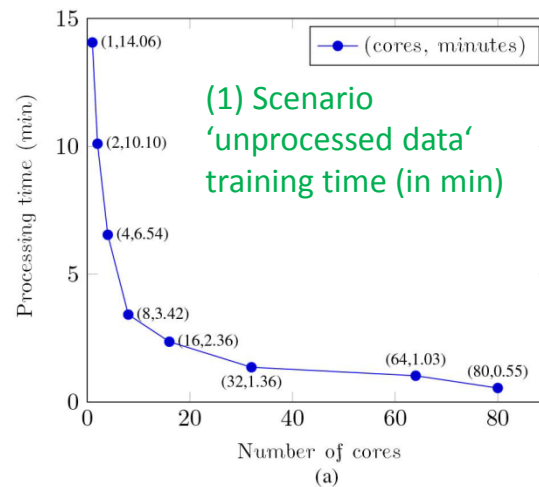
- Original piSVM 1.2 version (2011)
 - Open-source and based on libSVM library, C
 - Message Passing Interface (MPI)
 - New version appeared 2014-10 v. 1.3 (no major improvements)
 - Lack of 'big data' support (e.g. memory, layout)
- Tuned scalable parallel piSVM tool 1.2.1
 - Highly scalable version maintained by Juelich
 - Based on original piSVM 1.2 tool
 - Open-source (repository to be created)
 - Optimizations: load balancing; MPI collectives



➤ Open source code publicly available at: <https://github.com/mricherzhagen/pisvm>

Parallelization Benefit: Lower-Time-To-Solution

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



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)

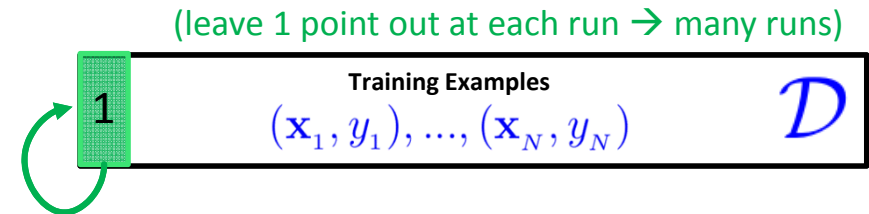


Validation Technique – Cross-Validation for Model Selection

- 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 (\rightarrow use parallelization)

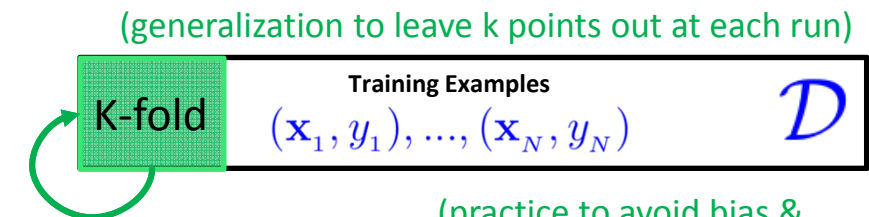
- Leave-one-out

- N training sessions on $N - 1$ points each time

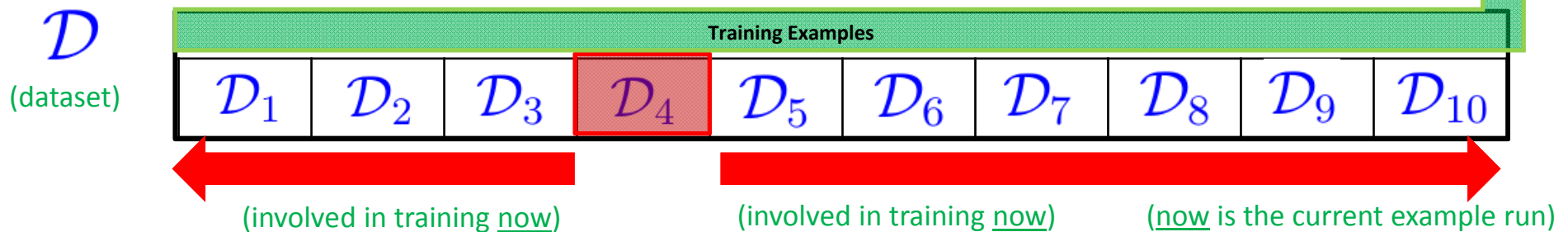


- Leave-more-out

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



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



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**)
 - Massive reduction in time → **lower time-to-solution – keeping accuracy!**

(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)

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

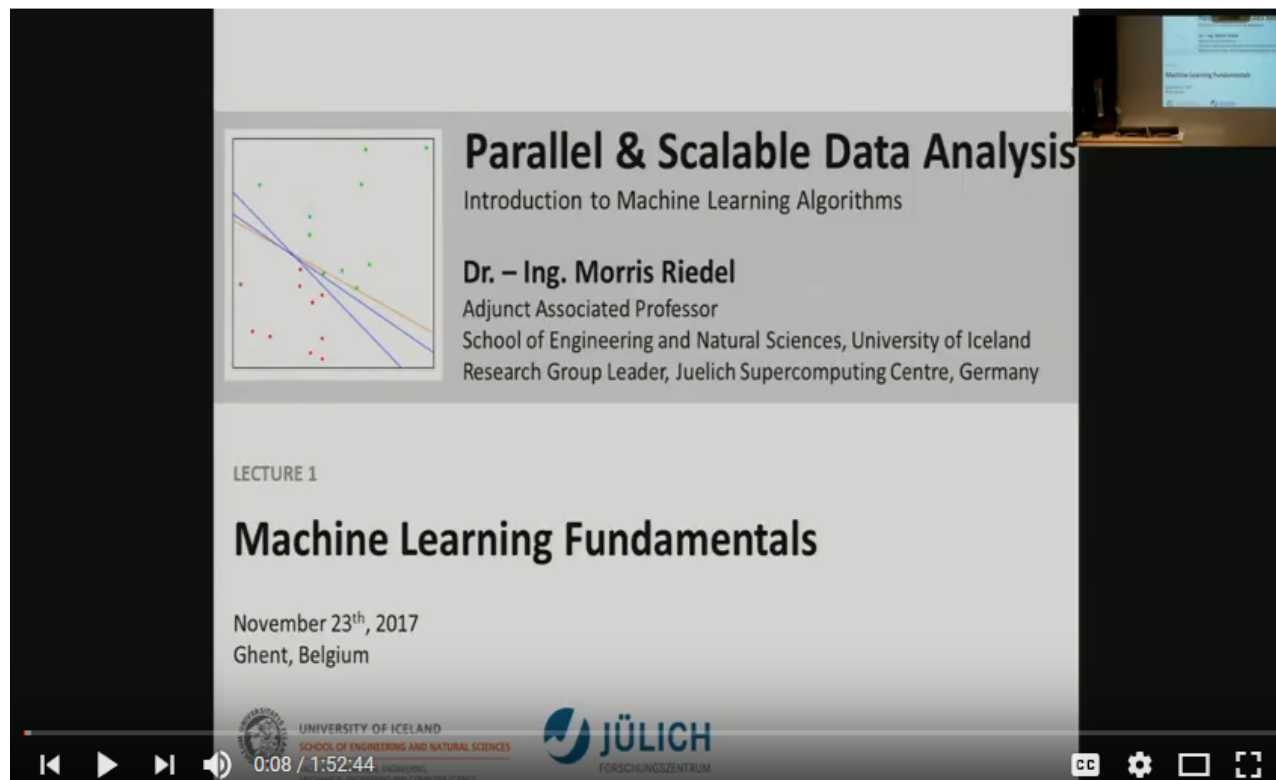
γ/C	1	10	100	1000	10 000
2	75.26 (1.02)	65.12 (1.03)	73.18 (1.33)	75.76 (2.35)	74.53 (4.40)
4	57.60 (1.03)	70.88 (1.02)	75.87 (1.03)	76.01 (1.33)	74.06 (2.35)
8	64.17 (1.02)	74.52 (1.03)	77.02 (1.02)	75.79 (1.04)	74.42 (1.34)
16	68.57 (1.33)	76.07 (1.33)	76.40 (1.34)	75.26 (1.05)	74.53 (1.34)
32	70.21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

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

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

[YouTube Lectures] More about parallel SVMs & HPC

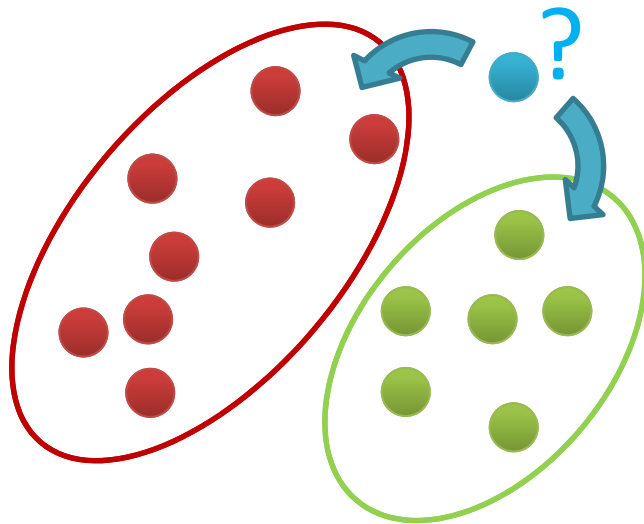


[20] Morris Riedel, 'Introduction to Machine Learning Algorithms', Invited YouTube Lecture, six lectures, University of Ghent, 2017

Methods Overview – Introduction to Deep Learning

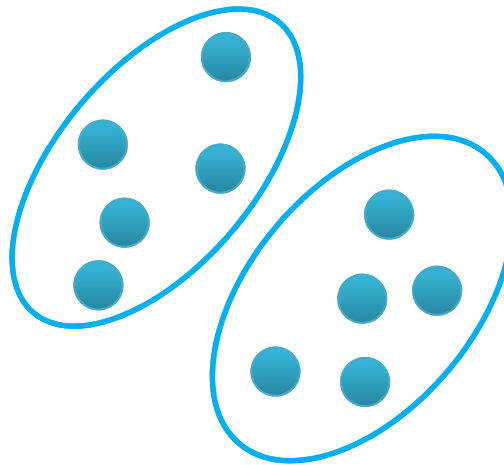
- Statistical data mining methods can be roughly categorized in classification, clustering, or regression augmented with various techniques for data exploration, selection, or reduction

Classification



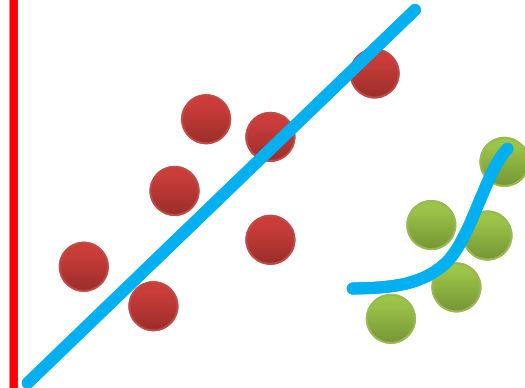
- Groups of data exist
- New data classified to existing groups

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

Learning Approaches – Unsupervised Learning

- Each observation of the predictor measurement(s) has **no associated response measurement**:
 - Input $\mathbf{x} = x_1, \dots, x_d$
 - **No output**
 - Data $(\mathbf{x}_1), \dots, (\mathbf{x}_N)$
- 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

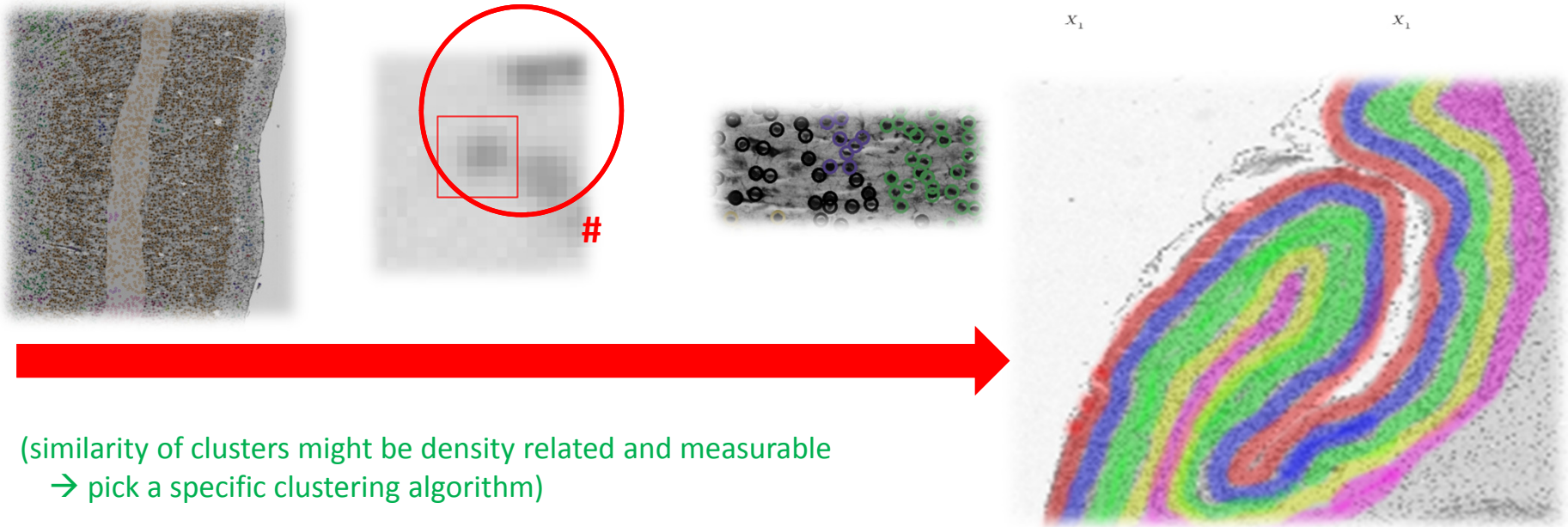
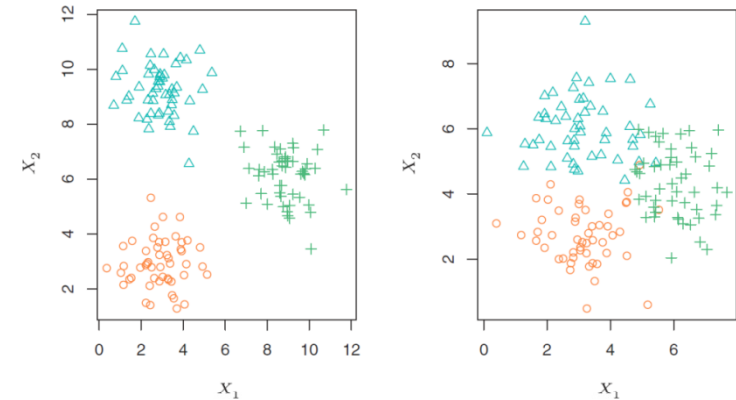
- Unsupervised learning approaches seek to understand relationships between the observations
- Unsupervised learning approaches are used in clustering algorithms such as k-means, etc.
- Unsupervised learning works with data = [input, ---]

[13] An Introduction to Statistical Learning

Learning Approaches – Unsupervised Learning Example

- Lessons learned from practice
 - The number of clusters are often ambiguities / no hard boundaries

[13] An Introduction to Statistical Learning



➤ Collaboration with Forschungszentrum Juelich - INM-1 Timo Dickscheid & Katrin Amunts

Selected Clustering Methods

- **K-Means Clustering** – Centroid based clustering
 - Partitions a data set into K distinct clusters (centroids can be artificial)
- **K-Medoids Clustering** – Centroid based clustering (variation)
 - Partitions a data set into K distinct clusters (centroids are actual points)
- Sequential Agglomerative hierarchic nonoverlapping (**SAHN**)
 - Hierarchical Clustering (create tree-like data structure → 'dendrogram')
- Clustering Using Representatives (**CURE**)
 - Select representative points / cluster – as far from one another as possible
- Density-based spatial clustering of applications + noise (**DBSCAN**)
 - Assumes clusters of similar density or areas of higher density in dataset

DBSCAN Algorithm

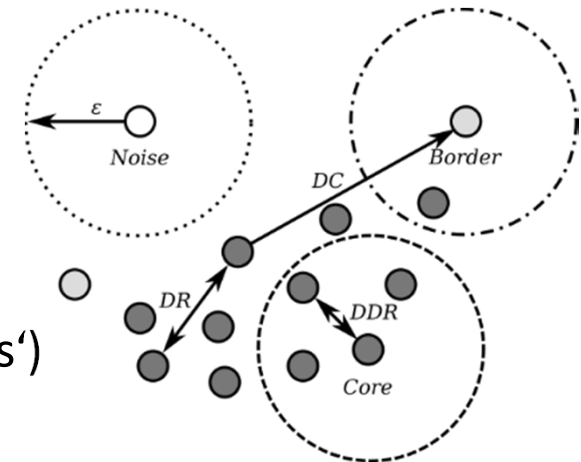
- DBSCAN Algorithm

[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*)

- Distinct Algorithm Features

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



(MinPoints = 4)

- 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?



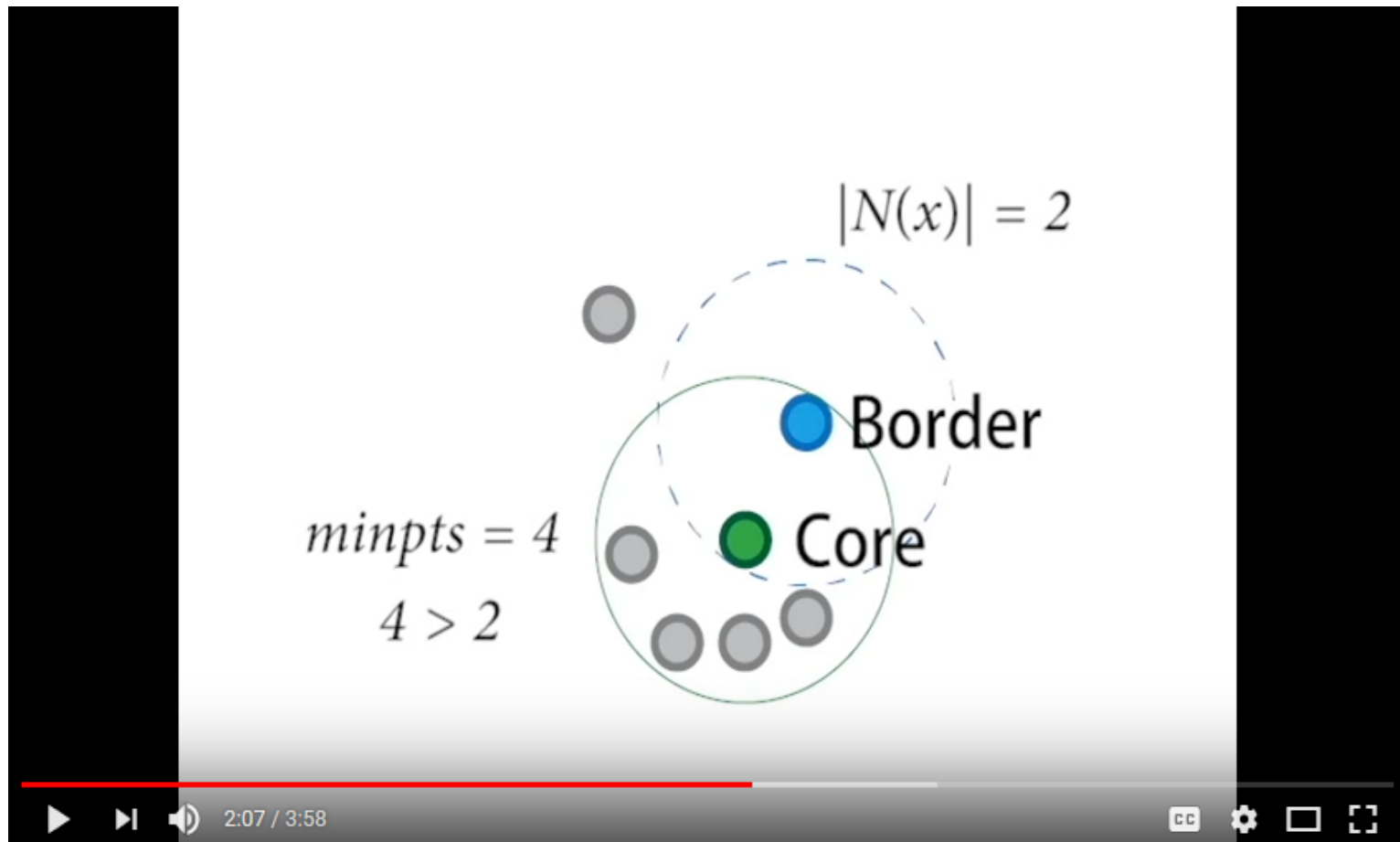
Unclustered
Data



Clustered
Data

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

[Video] DBSCAN Clustering

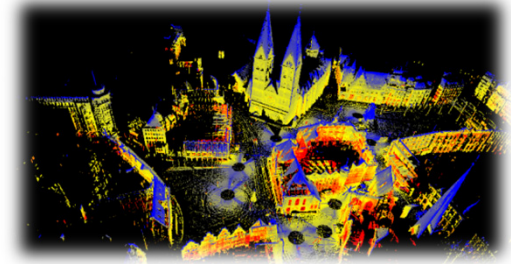


[6] DBSCAN, YouTube Video

'Big Data' Example – Point Cloud Applications

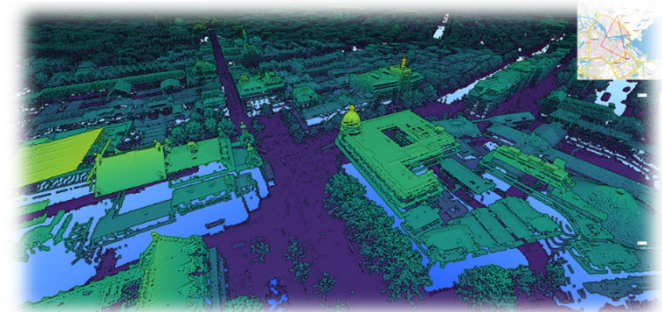
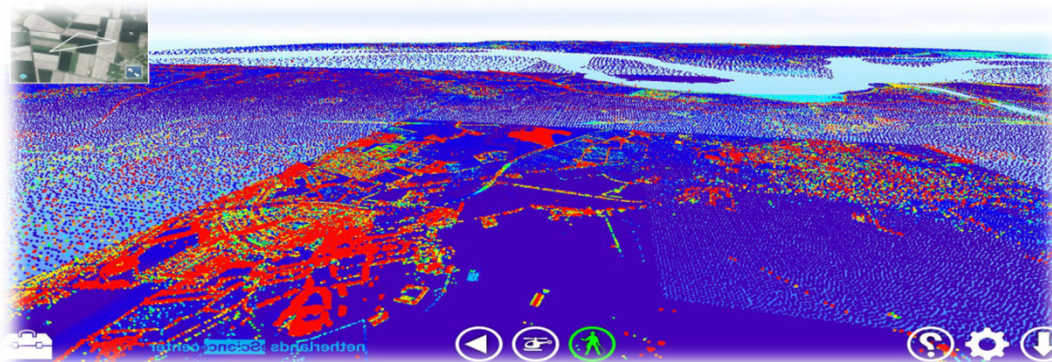
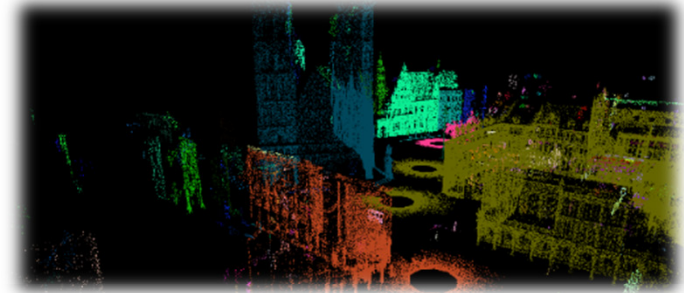
- 'Big Data': 3D/4D laser scans

- Captured by robots or drones
- Millions to billion entries
- Inner cities (e.g. Bremen inner city)
- Whole countries (e.g. Netherlands, USA per state)



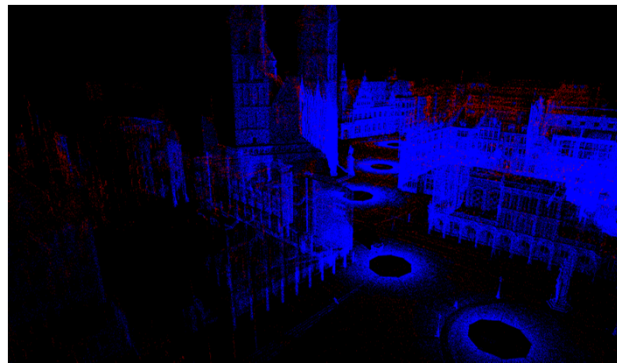
- Selected Scientific Cases

- Filter noise to better represent real data
- Grouping of objects (e.g. buildings)
- Study level of continuous details (complex)

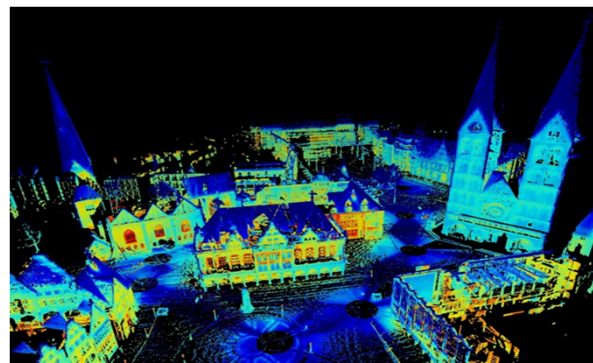


Open Bremen Dataset using Hierarchical Data Format (HDF)

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



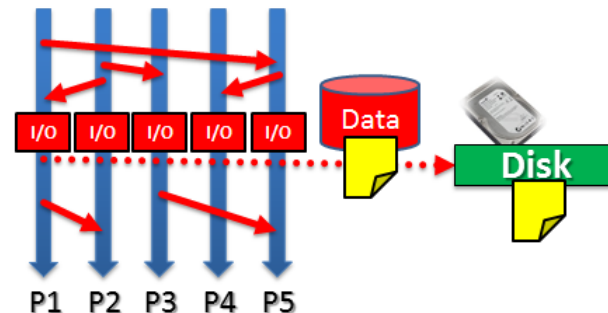
[22] Bremen Dataset



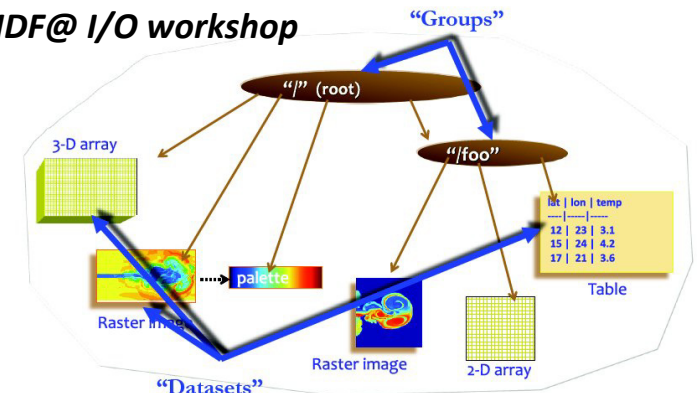
- The Bremen Dataset is encoded in the HDF5 parallel file format
- Enables efficient parallel I/O in HPC



(read & write : read point cloud data and assign cluster – IDs or mark noise)



[49] HDF@ I/O workshop



➤ Power of parallel I/O in HPC for 'big data' is often underestimated in machine learning community

Review of Open Source Parallel DBSCAN Implementations

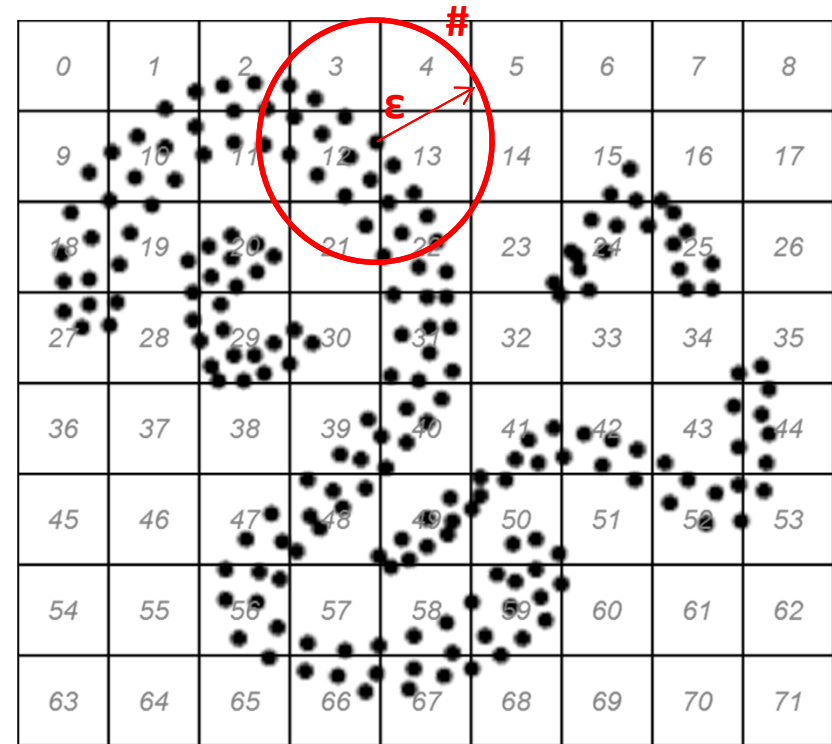
Technology	Platform Approach	Analysis
HPDBSCAN (authors implementation)	C; MPI; OpenMP	Parallel, hybrid, DBSCAN
Apache Mahout	Java; Hadoop	K-means variants, spectral, no DBSCAN
Apache Spark/MLlib	Java; Spark	Only k-means clustering, No DBSCAN
scikit-learn	Python	No parallelization strategy for DBSCAN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

➤ Work in progress: Spark/MLlib & ~10 DBSCAN codes not so good; other MPI code 2D only, ...

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

➤ Open source code publicly available at: <https://bitbucket.org/markus.goetz/hpdbscan>

HPDBSCAN – Smart Domain Decomposition Example

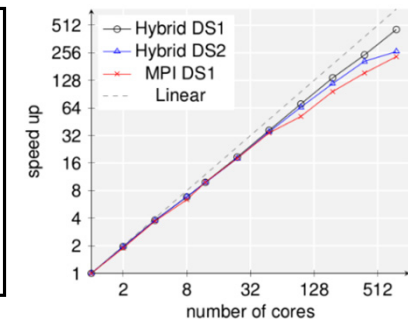
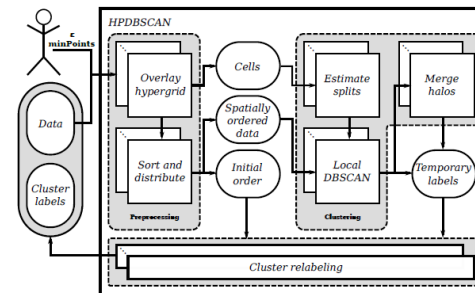
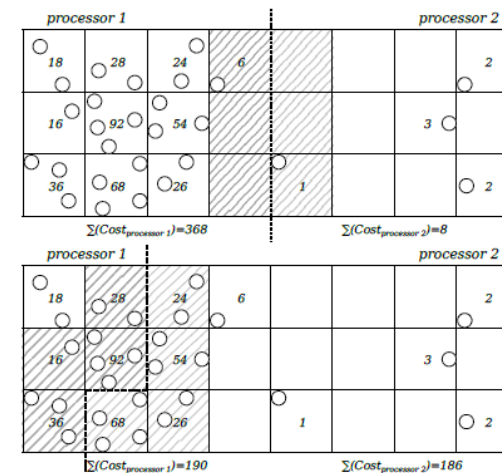
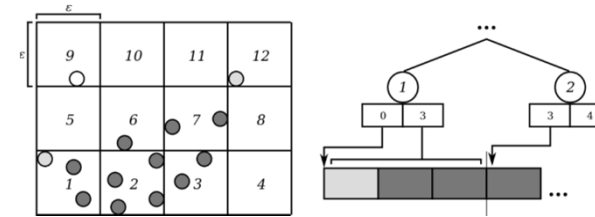
■ Parallelization Strategy

- 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

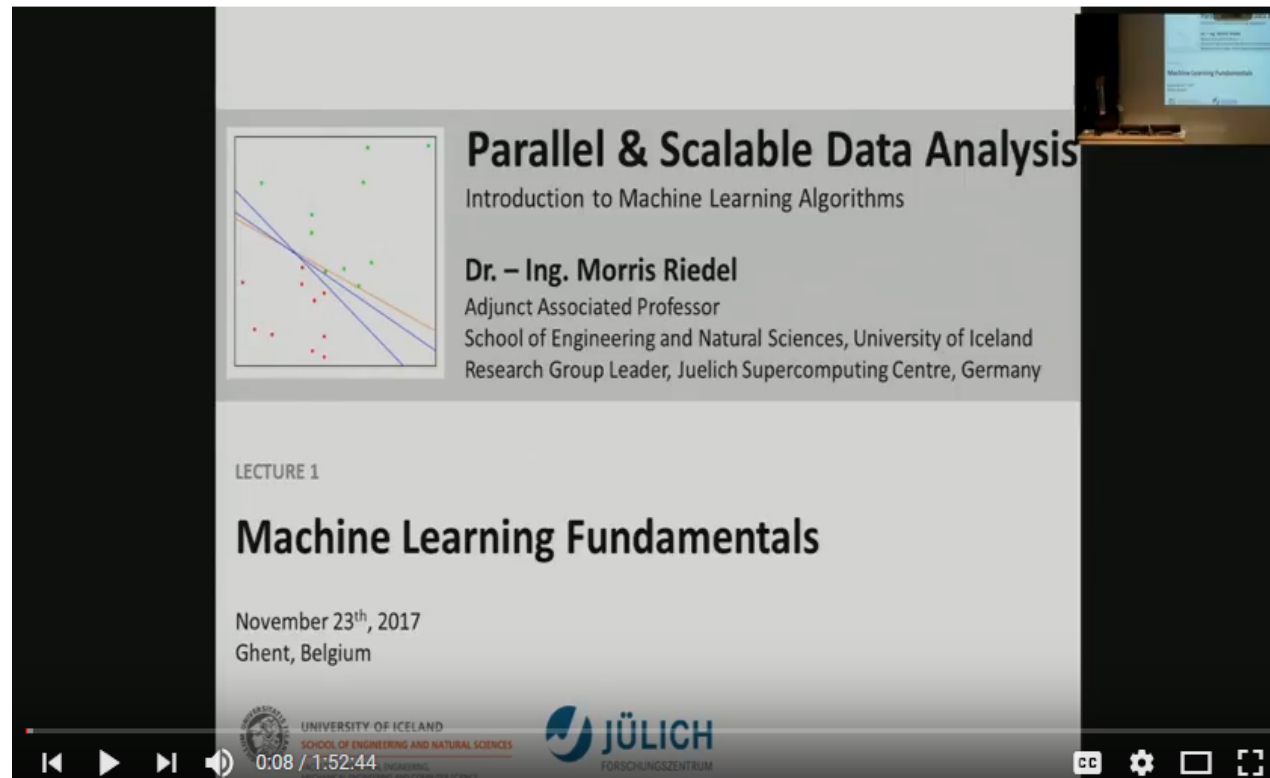
- Use of **HDF5** (stores noise ID / cluster ID)

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



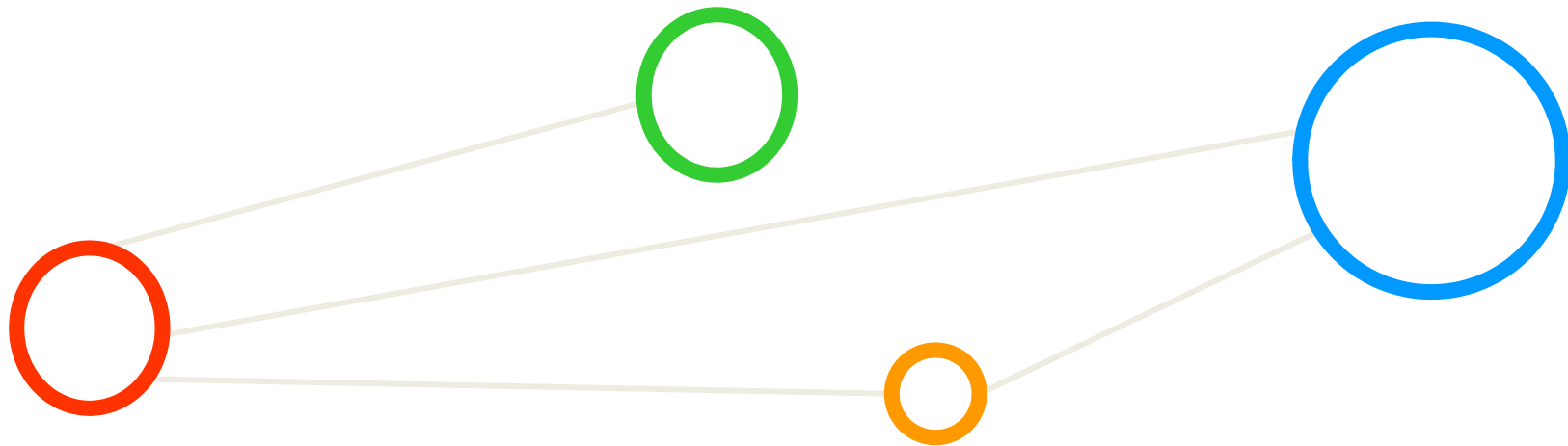
➤ Open source code publicly available at: <https://bitbucket.org/markus.goetz/hpdbscan>

[YouTube Lectures] More about parallel DBSCANs & HPC



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

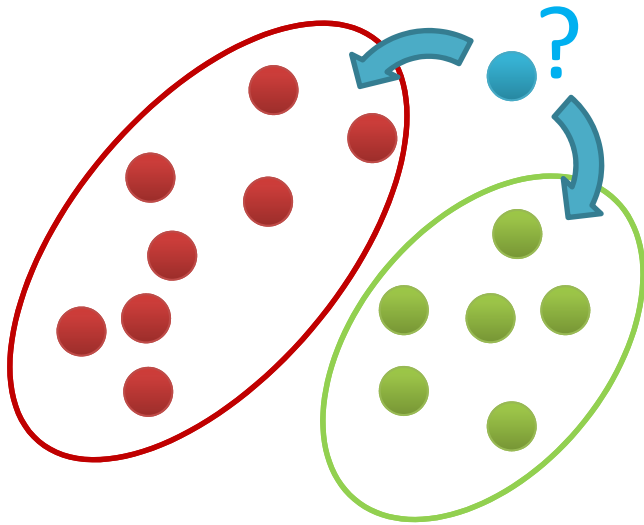
Selected Deep Learning Models



Methods Overview – Introduction to Deep Learning

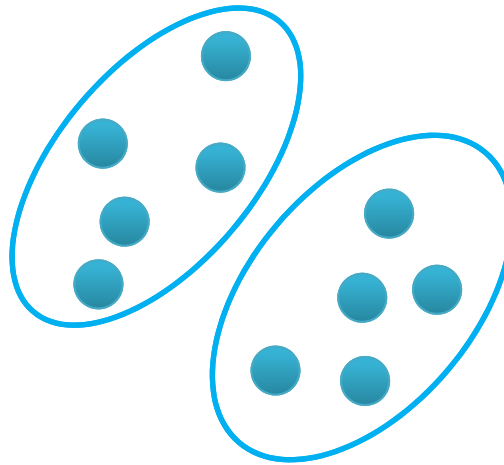
- Statistical data mining methods can be roughly categorized in classification, clustering, or regression augmented with various techniques for data exploration, selection, or reduction

Classification



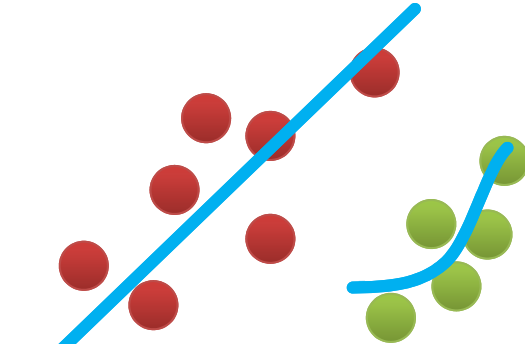
- Groups of data exist
- New data classified to existing groups

Clustering



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

Regression

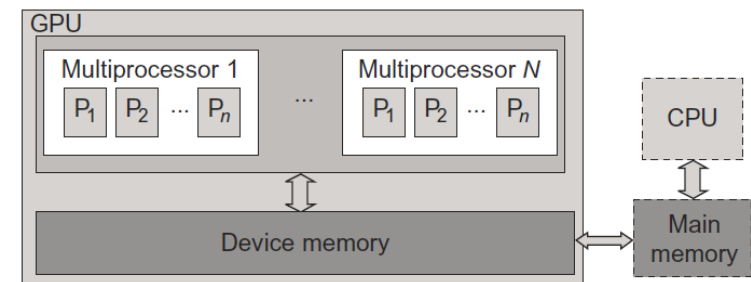
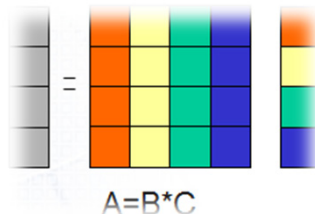


- Identify a line with a certain slope describing the data

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

➤ HPC Impact: Top500 #1 Summit (ORNL) 6 GPUs/node; 1st time more flop/s added by GPUs vs. CPUs

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

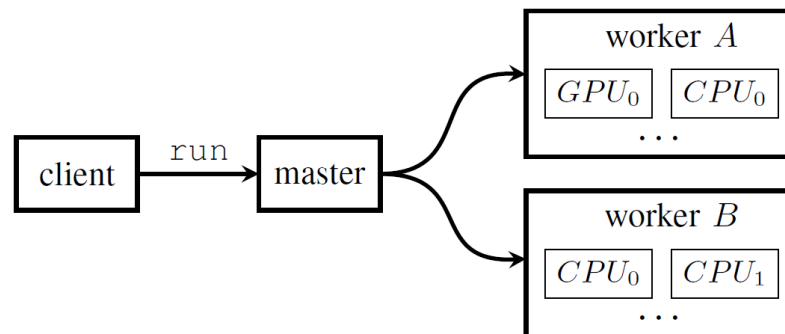


Keras

[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



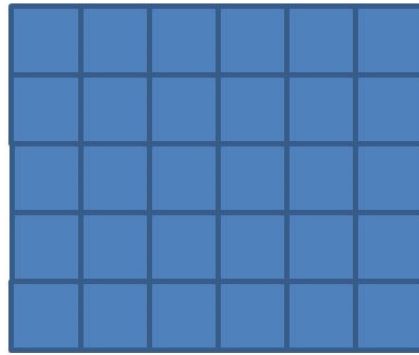
What is a Tensor?

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

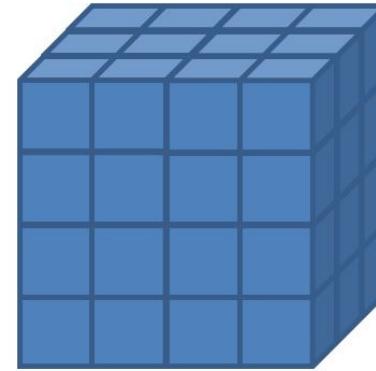
*[33] Big Data Tips,
What is a Tensor?*



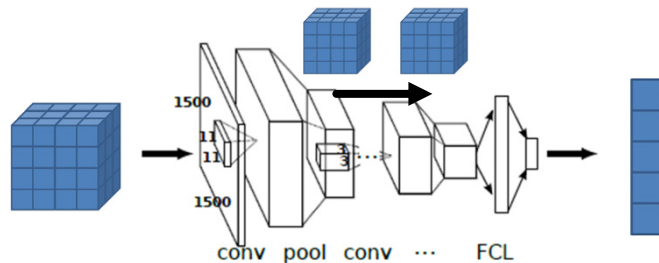
(one dimensional tensor)
(vector of dimension [5])



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



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



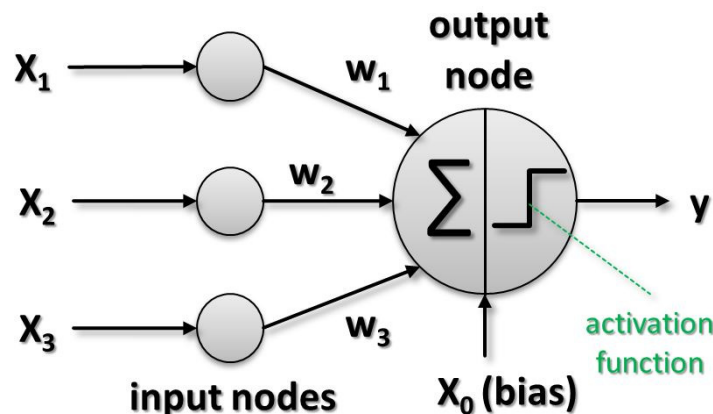
(‘tensors flow through the
deep learning networks)

(note: learned weighted connections
often omitted from many deep
learning network visualizations)

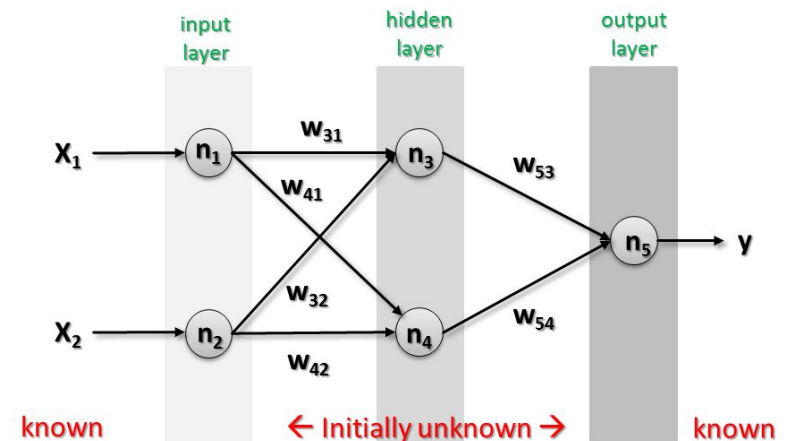
Artificial Neural Network – Feature Engineering & Layers

- 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 recognition ('state-of-the-art in your phone')

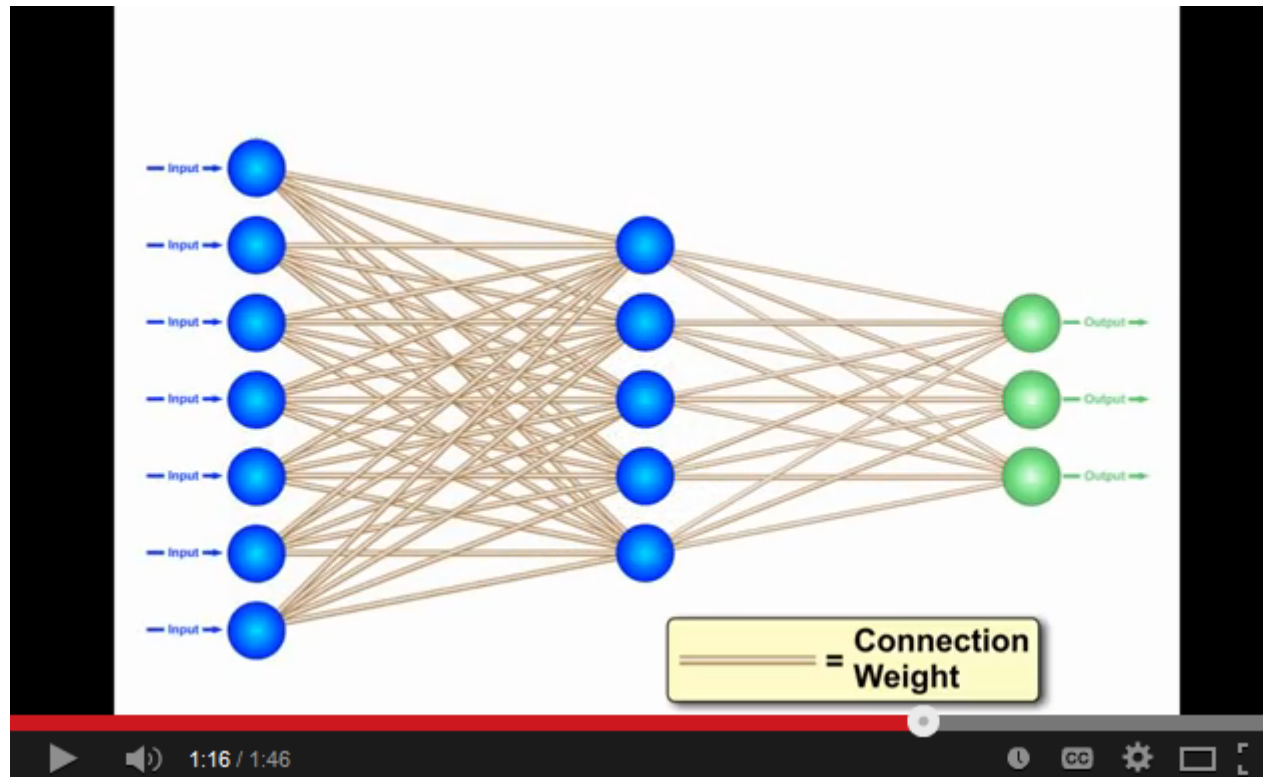


(Perceptron model: designed after human brain neuron)



(Artificial neural network two layer feed – forward)

[Video] Towards Multi-Layer Perceptrons



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

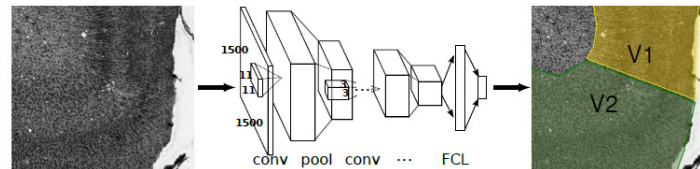
Deep Learning Architectures

- Deep Neural Network (DNN)

- 'Shallow ANN' approach with many hidden layers between input/output

- Convolutional Neural Network (CNN, sometimes ConvNet)

- Connectivity pattern between neurons is like animal visual cortex



- Deep Belief Network (DBN)

- Composed of multiple layers of variables; only connections between layers

- Recurrent Neural Network (RNN)

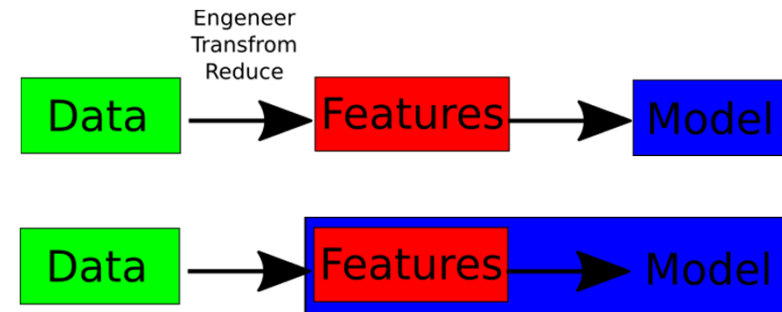
- 'ANN' but connections form a directed cycle; state and temporal behaviour

- Deep Learning architectures can be classified into Deep Neural Networks, Convolutional Neural Networks, Deep Belief Networks, and Recurrent Neural Networks all with unique characteristics
- Deep Learning needs 'big data' to work well & for high accuracy – works not well on sparse data

Deep Learning – Feature Learning & More Smart Layers

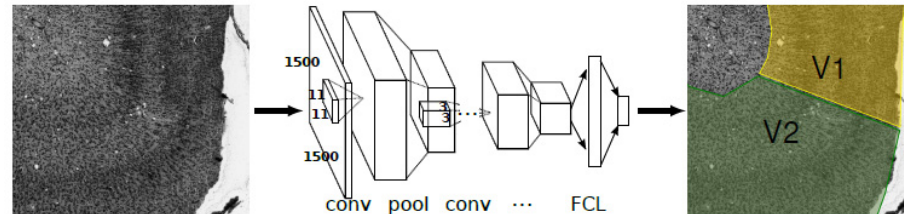
- Approach: Learn Features

- Classical Machine Learning
- (Powerful computing evolved)
- Deep (Feature) Learning

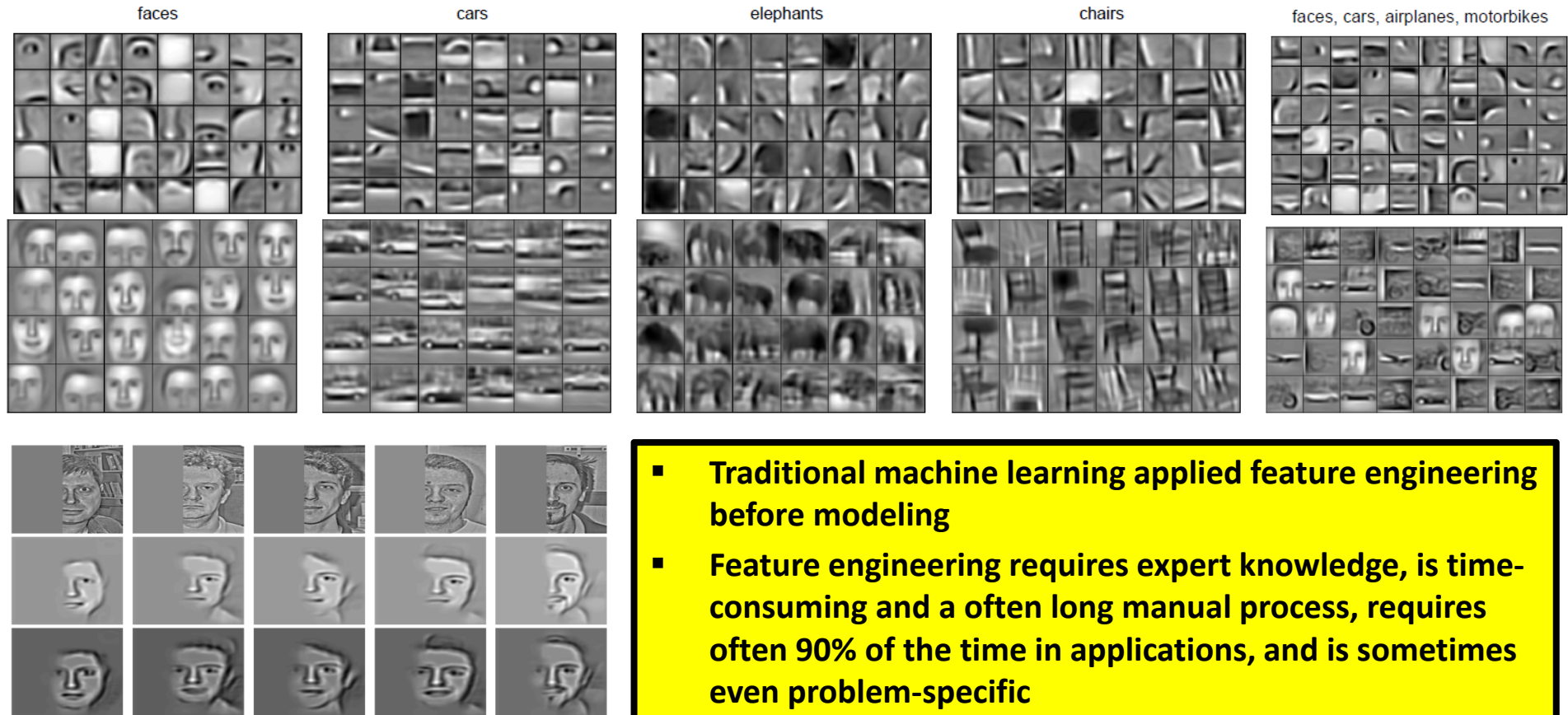


- Very successful for image recognition and other emerging areas
- Assumption: data was generated by the interactions of many different factors on different levels (i.e. form a hierarchical representation)
- Organize factors into multiple levels, corresponding to different levels of abstraction or composition (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

➤ More background information about CNN and its key elements are provided in Appendix D

HPC Machine: JSC JURECA System – CLUSTER Module

■ Characteristics

- Login nodes with 256 GB memory per node
- 45,216 CPU cores
- 1.8 (CPU) + 0.44 (GPU) Petaflop/s peak performance
- Two Intel Xeon E5-2680 v3 Haswell CPUs per node: 2 x 12 cores, 2.5 GHz
- 75 compute nodes equipped with two NVIDIA K80 GPUs (2 x 4992 CUDA cores)

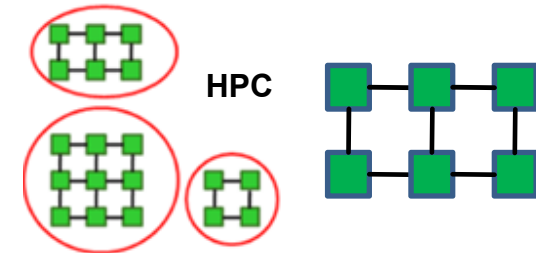
■ Architecture & Network

- Based on T-Platforms V-class server architecture
- Mellanox EDR InfiniBand high-speed network with non-blocking fat tree topology
- 100 GiB per second storage connection to JUST



[48] JURECA HPC System

- Use our ssh keys to get an access and use reservation
- Put the private key into your `./ssh` directory (UNIX)
- Use the private key with your putty tool (Windows)



Deep Learning – Scaling Example on JURECA HPC System

- Simple Image Benchmark on JURECA JSC HPC System

- 75 x 2 NVIDIA Tesla K80/node – dual GPU design

- 1.2 mio images with 224 x 224 pixels

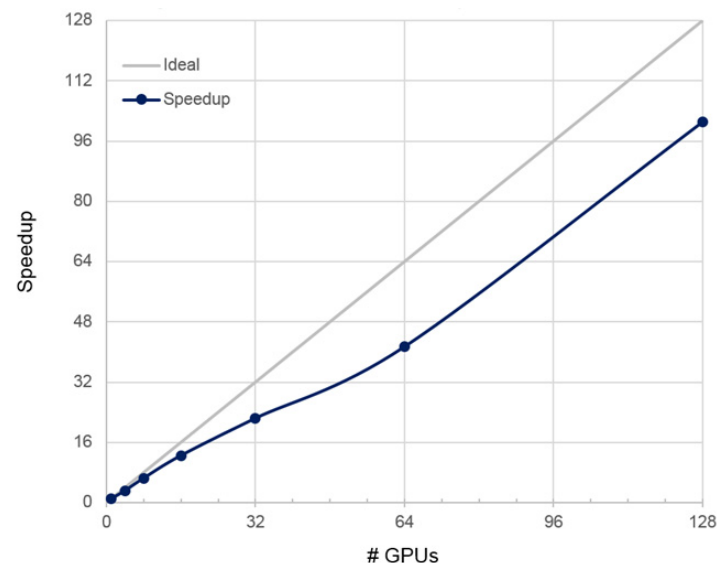
#GPUs	images/s	speedup	Performance per GPU [images/s]
1	55	1.0	55
4	178	3.2	44.5
8	357	6.5	44.63
16	689	12.5	43.06
32	1230	22.4	38.44
64	2276	41.4	35.56
128	5562	101.1	43.45

(absolute number of images per second and relative speedup normalized to 1 GPU are given)

[41] A. Sergeev, M. Del

Balso, 'Horovod', 2018

(setup: TensorFlow 1.4, Python 2.7, CUDA 8, cuDNN 6, Horovod 0.11.2, MVAPICH-2.2-GDR)



- Open source tool Horovod enables distributed deep learning with TensorFlow / Keras
- Machine & Deep Learning: speed-up is just secondary goal after 1st goal accuracy
- Speed-up & parallelization good for faster hyperparameter tuning, training, inference
- Third goal is to avoid much feature engineering through 'feature learning'

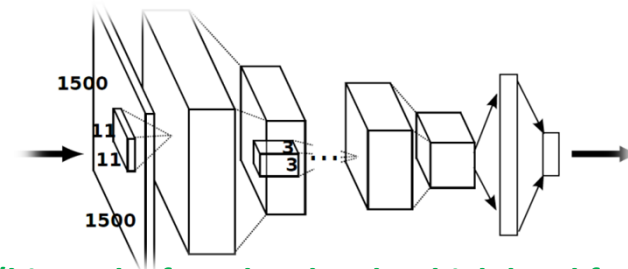
Deep Learning – Key Properties & Application Areas

- 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’)

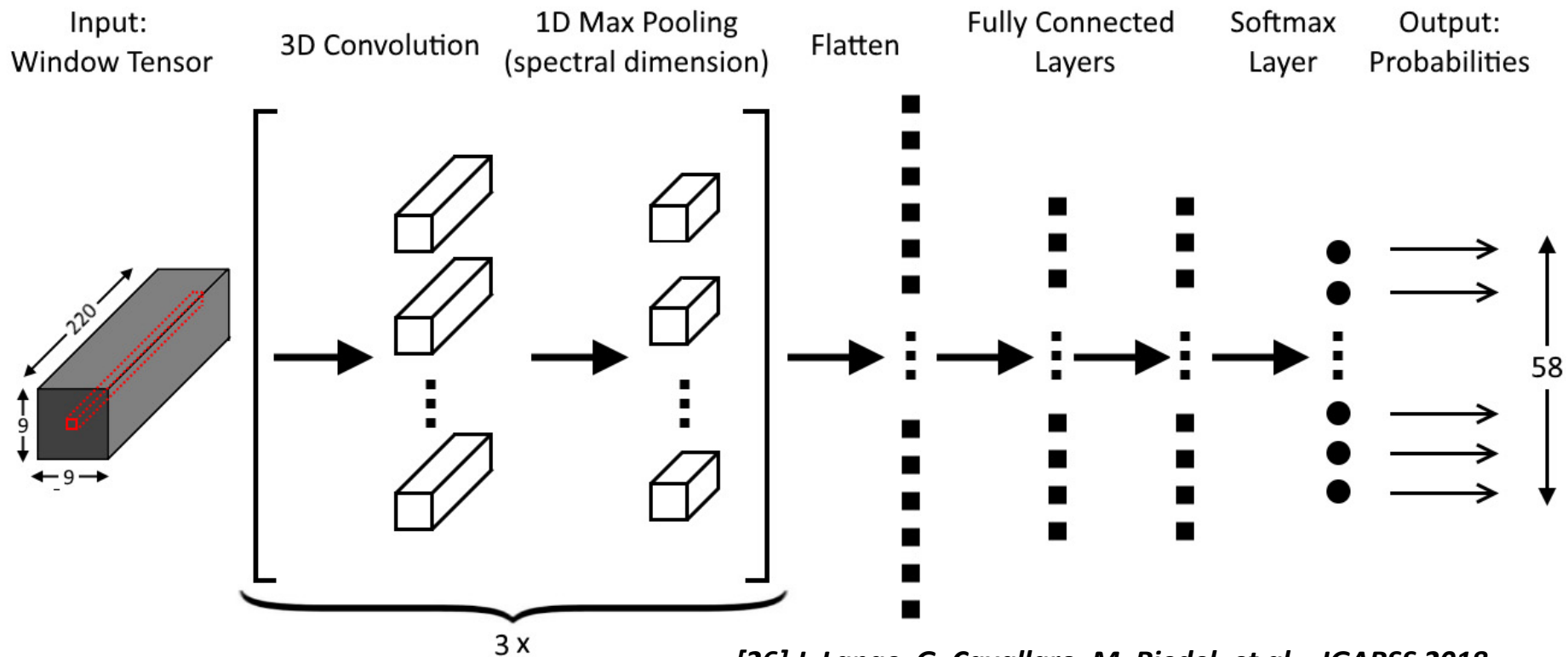
(Deep Learning is often ‘just’ called rebranding of traditional neural networks)



(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)

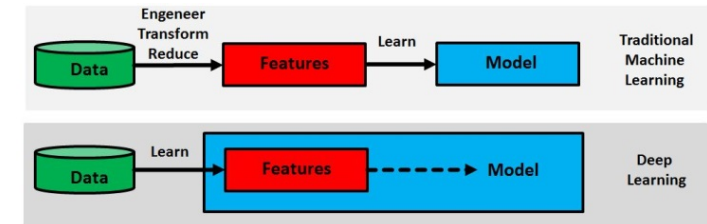


[26] J. Lange, G. Cavallaro, M. Riedel, et al., IGARSS 2018

Comparison: Traditional Machine Learning vs. Deep Learning

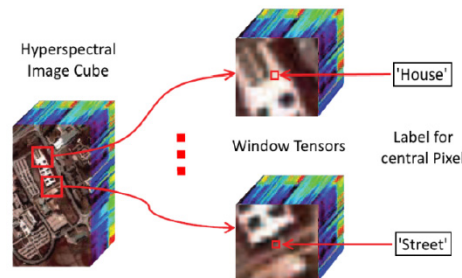
■ Traditional Methods

- 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
- Warning: optimistic bias –
careful data sampling vs. 'big data'!



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

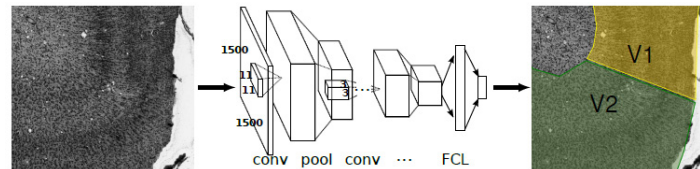
Deep Learning Architectures – Revisited

- Deep Neural Network (DNN)

- ‘Shallow ANN’ approach with many hidden layers between input/output

- Convolutional Neural Network (CNN, sometimes ConvNet)

- Connectivity pattern between neurons is like animal visual cortex



- Deep Belief Network (DBN)

- Composed of multiple layers of variables; only connections between layers

- Recurrent Neural Network (RNN)

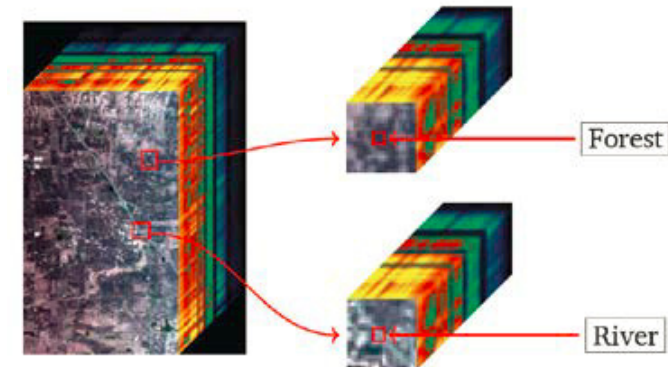
- ‘ANN’ but connections form a directed cycle; state and temporal behaviour

- Deep Learning architectures can be classified into Deep Neural Networks, Convolutional Neural Networks, Deep Belief Networks, and Recurrent Neural Networks all with unique characteristics
- Deep Learning needs ‘big data’ to work well & for high accuracy – works not well on sparse data

Revisit CNNs vs. RNNs – Different Type of Neural Networks

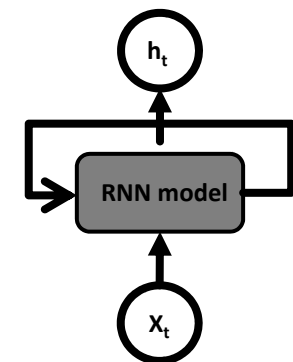
- CNNs → Spatial

- Example: remote sensing application domain, **hyperspectral datasets**
- Neural network key property: **exploit spatial geometry of inputs**
- Approach: **Apply convolution & pooling** (height x width x feature) dimensions



- RNNs → Temporal

- Examples: **texts, speech, time series datasets**
- Neural network key property: **exploit sequential nature of inputs**
- Approach: **Train a graph of 'RNN cells' & each cell performs the same operation on every element** in the given sequence



- **RNNs are used to create sequence models whereby the occurrence of an element in the sequence (e.g. text, speech, time series) is dependent on the elements that appeared before it**

Sequence Models

- Sequence models enable various sequence predictions that are inherent different to other more traditional predictive modeling techniques or supervised learning approaches
- In contrast to mathematical sets often used, the 'sequence' model imposes an explicit order on the input/output data that needs to be preserved in training and/or inference
- Sequence models are driven by application goals and include sequence prediction, sequence classification, sequence generation, and sequence-to-sequence prediction

- Model Categorization

- Based on different inputs/outputs to/from the sequence models

- Practical 'standard dataset' perspective

- Often the order of samples is not important
 - Training/testing datasets and their samples have often no explicit order (i.e. 'sets')

- Practical 'sequence dataset' perspective

- Order of samples is important: sequence learning/inference needs order

➤ More background information about RNNs & LSTMs is in the Appendix E in this slideset

Recurrent Neural Network (RNN)

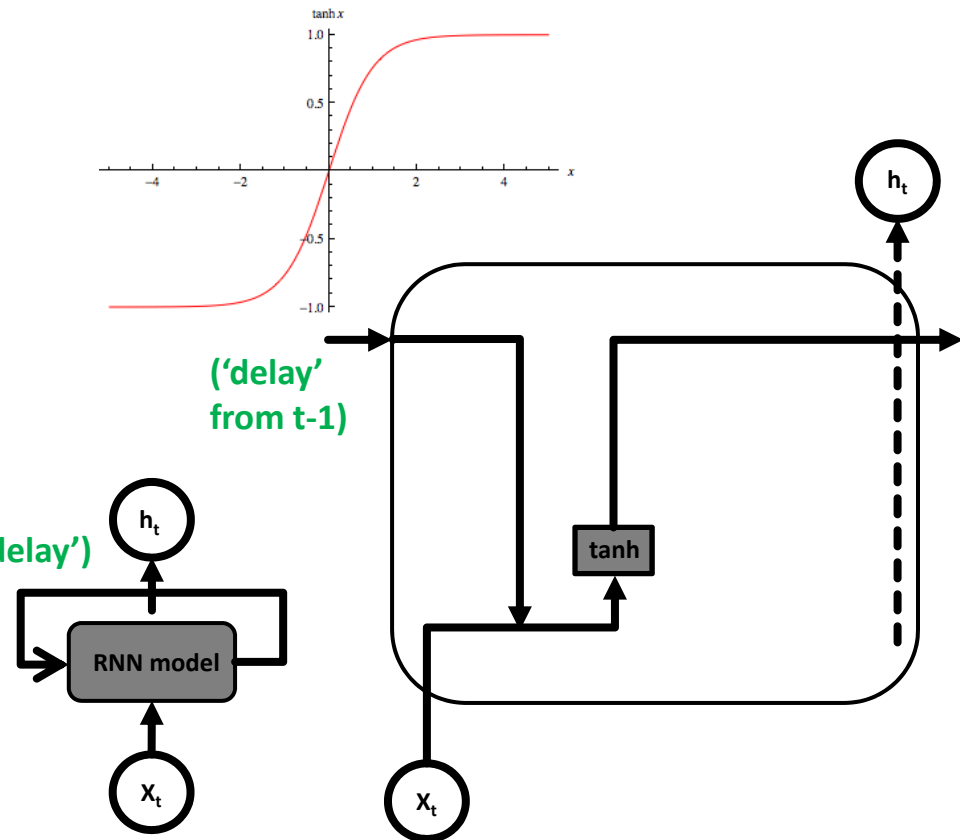
- A Recurrent Neural Network (RNN) consists of cyclic connections that enable the neural network to better model sequence data compared to a traditional feed forward artificial neural network (ANN)
- RNNs consists of 'loops' (i.e. cyclic connections) that allow for information to persist while training
- The repeating RNN model structure is very simple whereby each has only a single layer (e.g. tanh)

- Selected applications

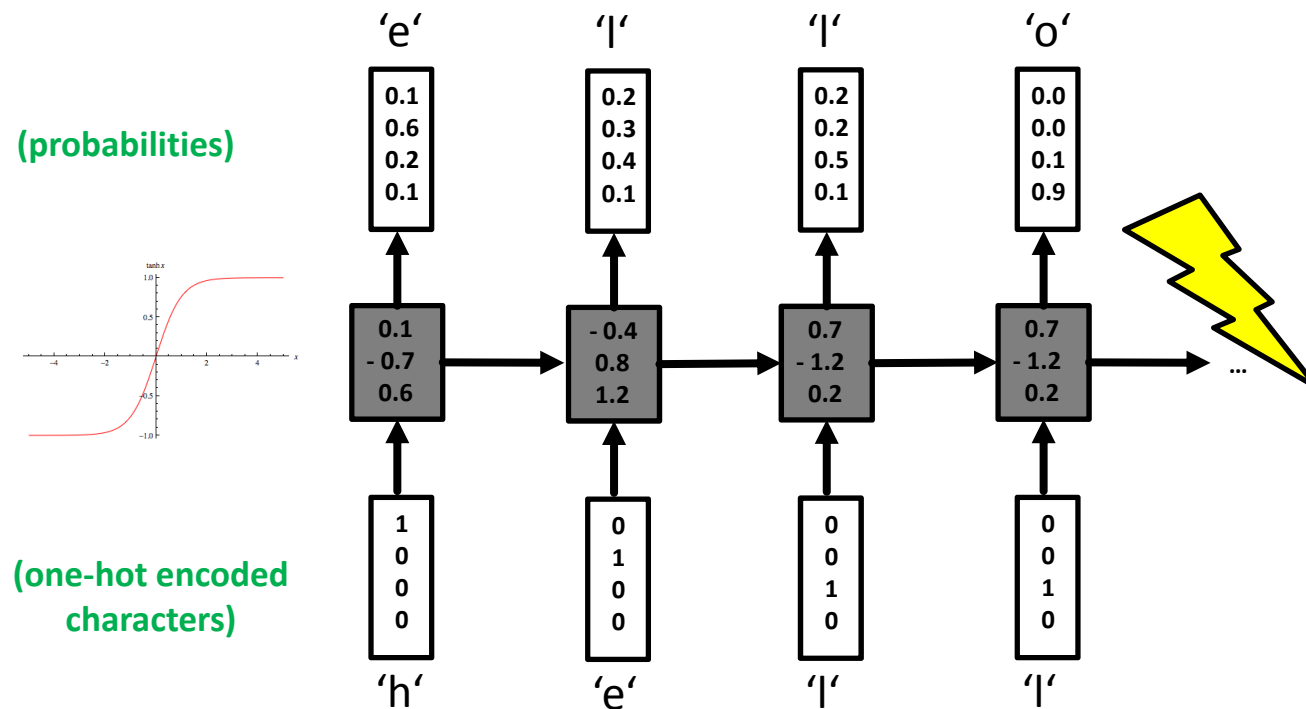
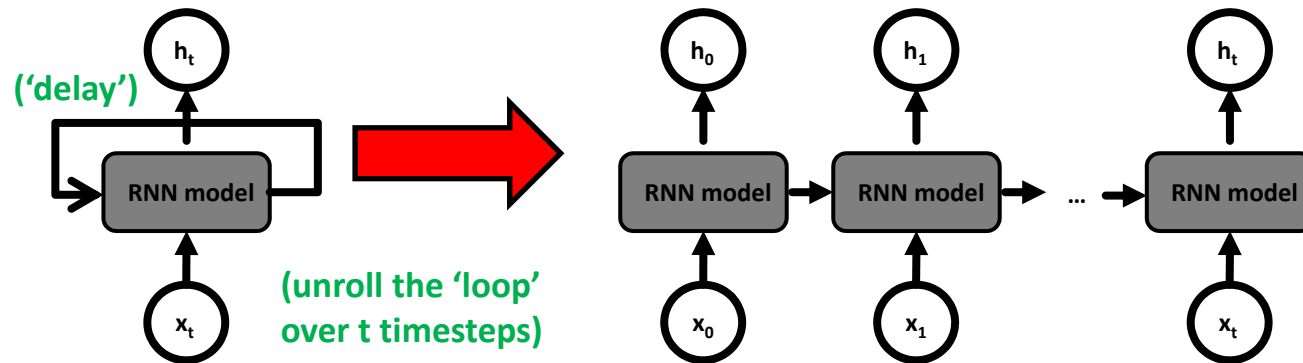
- Sequence labeling
- Sequence prediction tasks
- E.g. handwriting recognition
- E.g. language modeling

- Loops / cyclic connections

- Enable to pass information ('delay') from one step to the next iteration
- Remember 'short-term' data dependencies

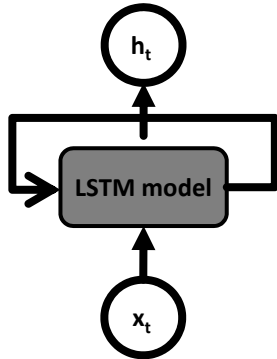


RNN Model – Simple Example – Predict Next Character

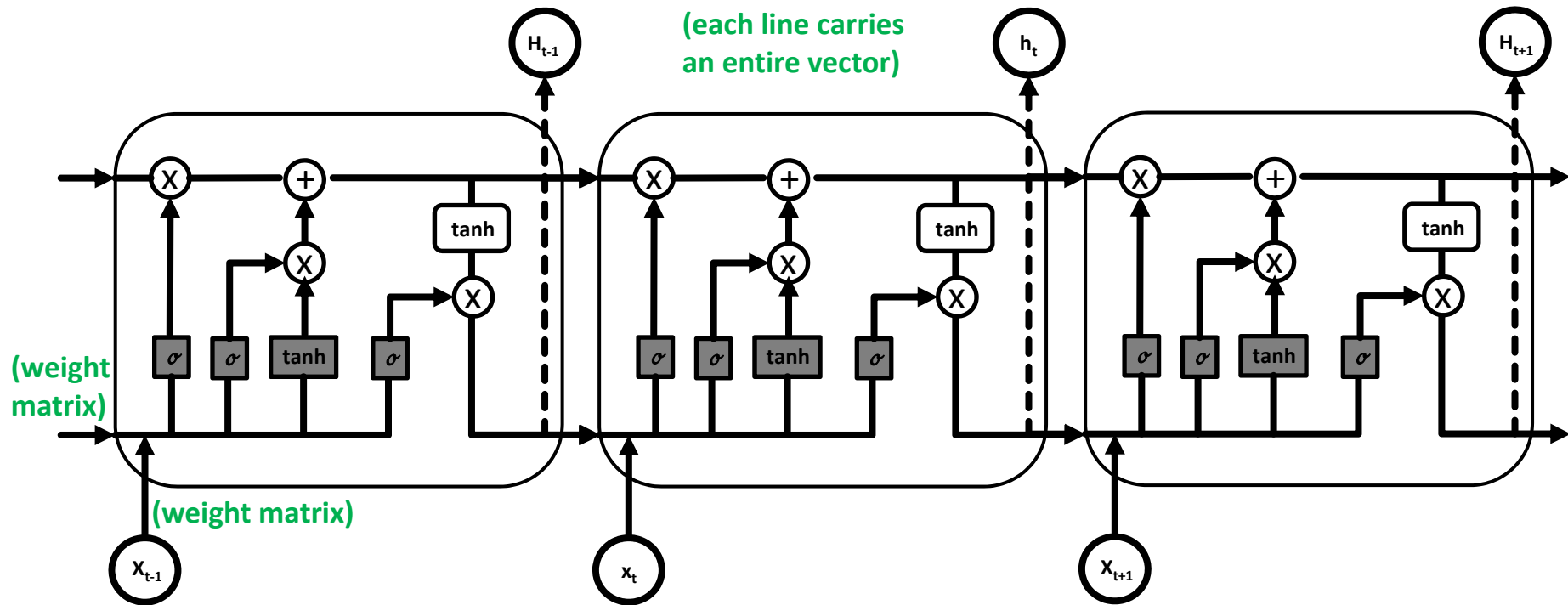
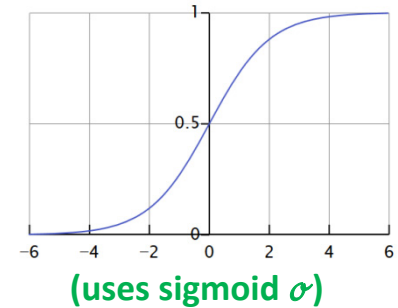


- Sequence values that are separated by a significant number of words (i.e. deep RNN) leads to the vanishing gradient problem
- Reasoning is that small gradients or weights with values than 1 are multiplied many times through the multiple time steps, i.e. gradients shrink asymptotically to zero
- Effect is that weights of those earlier layers are not changed significantly and the network will not learn long-term dependencies

Long Short Term Memory (LSTM) Model

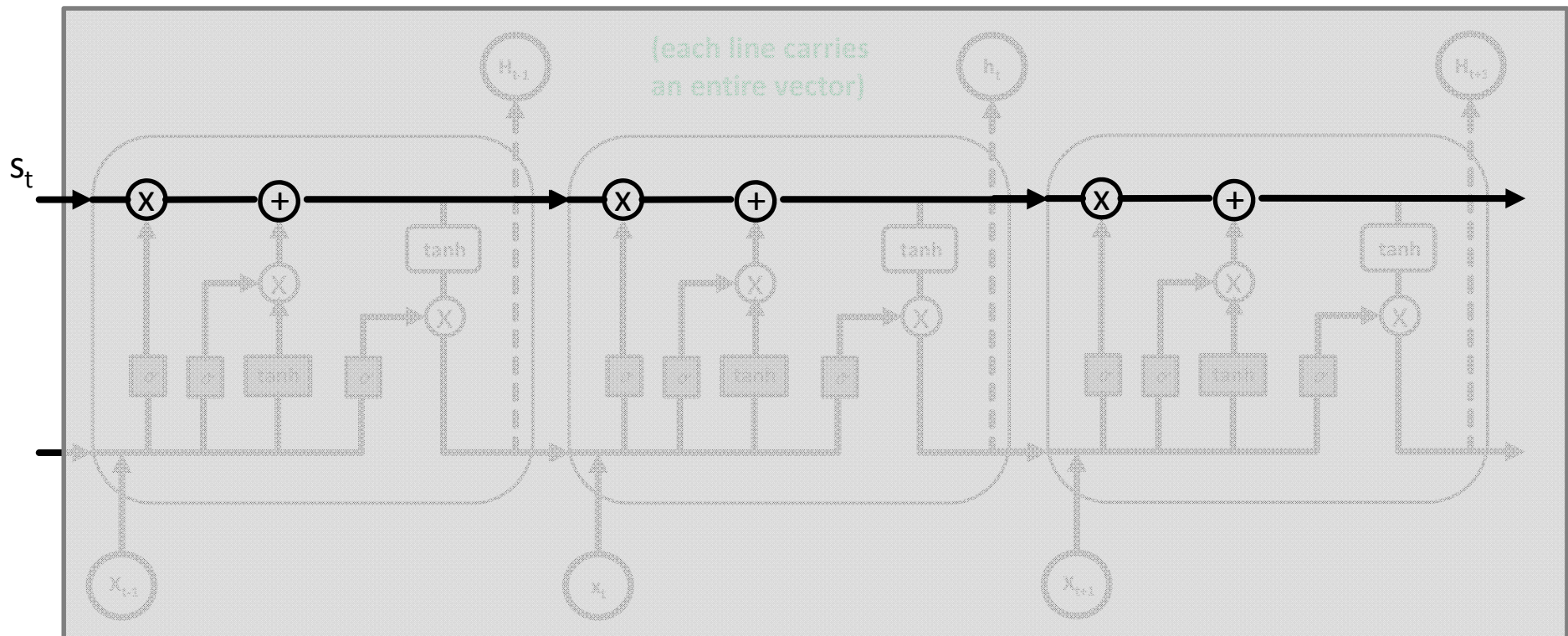


- Long Short Term Memory (LSTM) networks are a special kind of Recurrent Neural Networks (RNNs)
- LSTMs learn long-term dependencies in data by remembering information for long periods of time
- The LSTM chain structure consists of four neural network layers interacting in a specific way



LSTM Model – Memory Cell & Cell State

- LSTM introduce a 'memory cell' structure into the underlying basic RNN architecture using four key elements: an input gate, a neuron with self-current connection, a forget gate, and an output gate
- The data in the LSTM memory cell flows straight down the chain with some linear interactions (x,+)
- The cell state s_t can be different at each of the LSTM model steps & modified with gate structures
- Linear interactions of the cell state are pointwise multiplication (x) and pointwise addition (+)
- In order to protect and control the cell state s_t three different types of gates exist in the structure



Deep Learning for Sequence Data: Long Short-Term Memory

- Standard LSTM

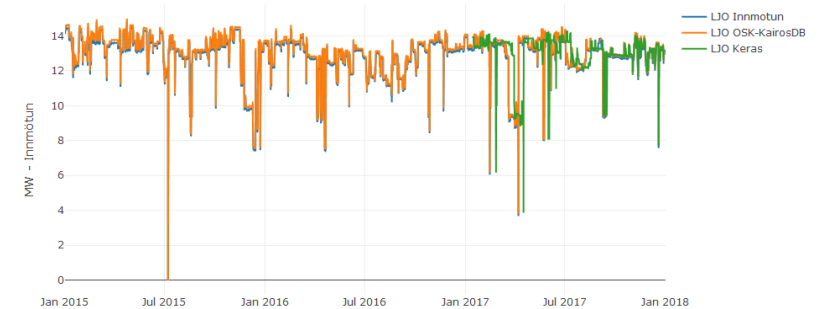
```
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from keras.layers import Dropout
```

```
# 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..")

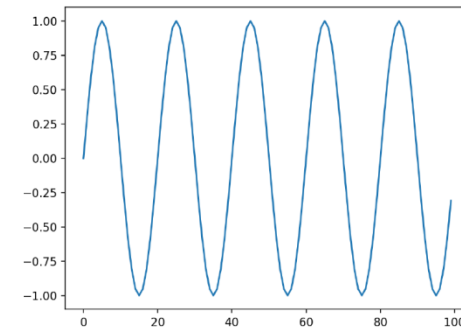
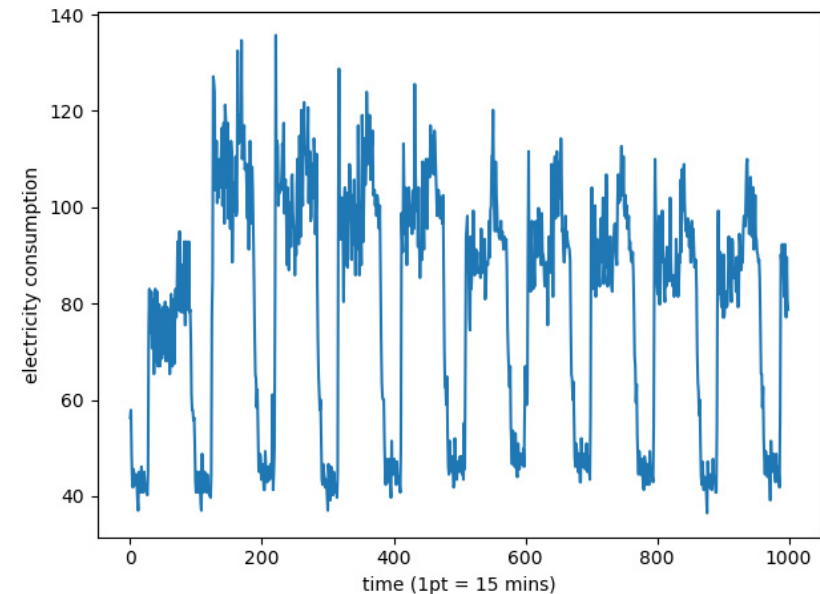
history = model.fit(
    train_X,
    train_y,
    epochs=config['epochs'],
    batch_size=config['batchsize'],
    validation_data=(test_X, test_y),
    verbose=2,
    shuffle=config['shuffle']
)
```



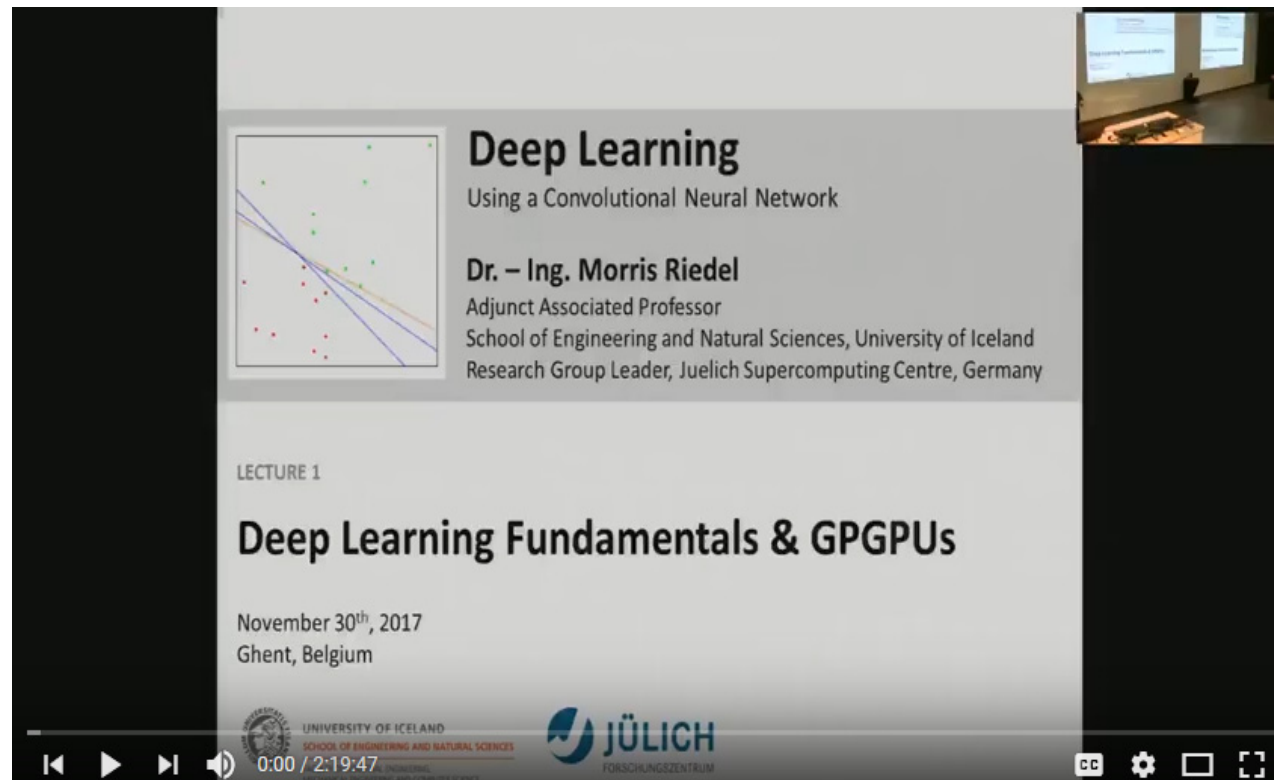
- LSTM models work quite well to predict power but needs to be trained and tuned for different power stations
- Observing that some peaks can not be 'learned' although robust model
- Requires much longer time to train (i.e. more HPC time or GPUs/node)

Different Useful LSTM Models – Stacked LSTMs

- E.g. predicting electricity consumption / customer
 - Stacked LSTM cells
 - Periodic elements can take advantage of state
 - Needs to be carefully tuned
 - Requires through use of state more computing
- E.g. damped sine wave prediction
 - Stacked LSTM cells since again periodic character
 - Depending on wave the pattern might be not able to be detected w/o LSTMs

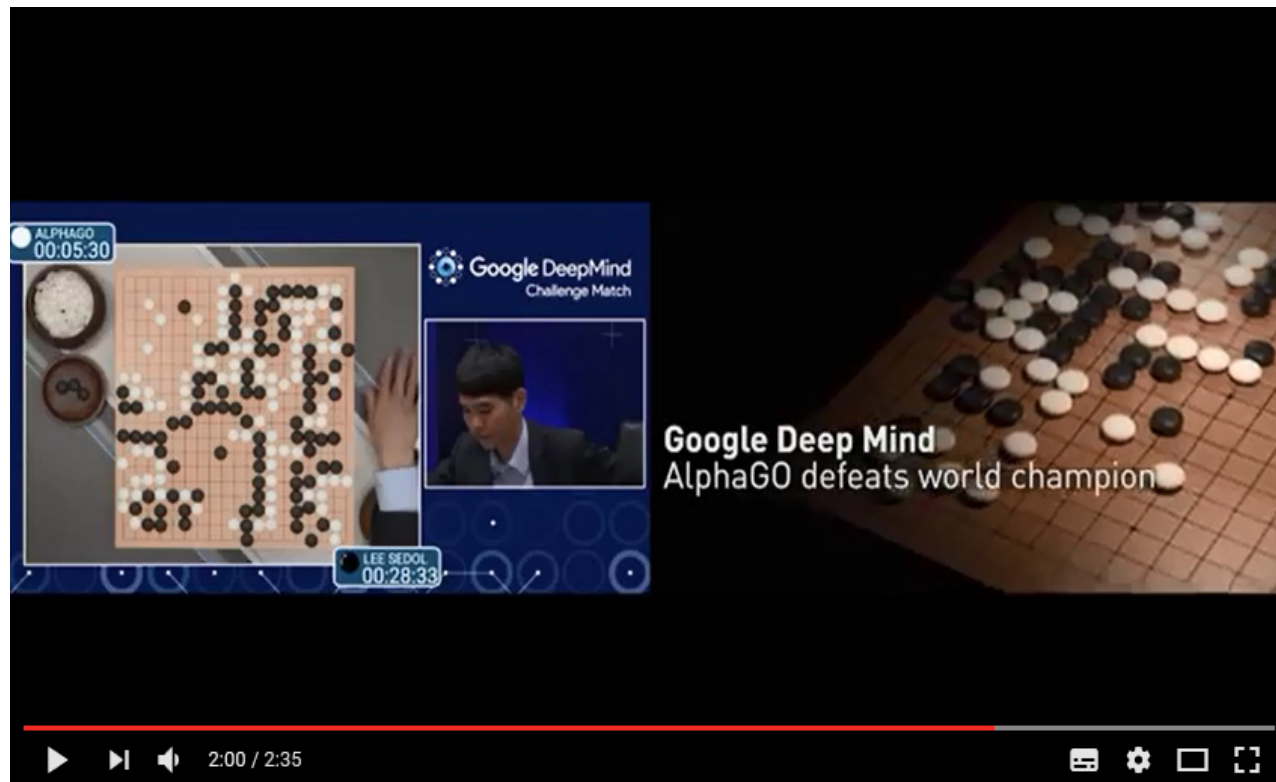


[YouTube Lectures] More about Deep Learning & HPC



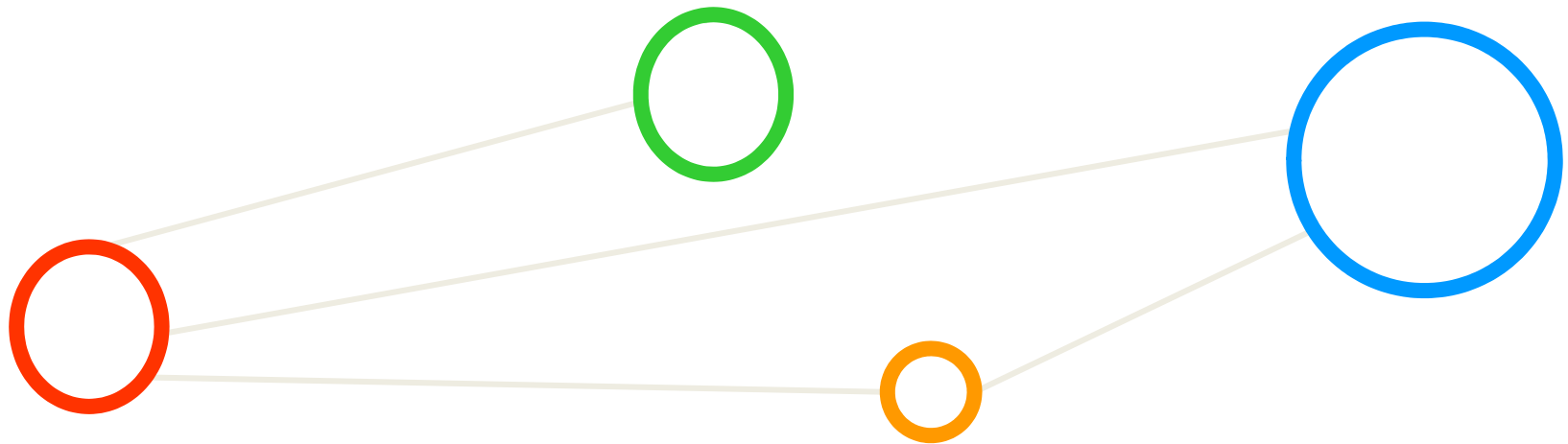
***[21] Morris Riedel, 'Deep Learning - Using a Convolutional Neural Network',
Invited YouTube Lecture, six lectures, University of Ghent, 2017***

[Video] Deep Learning ‘Revolution’

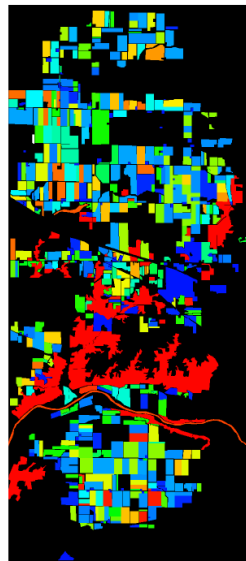
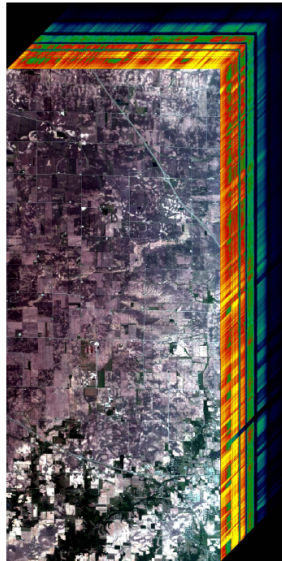


[27] The Deep Learning Revolution, YouTube

Open Challenges & Summary



Number of Parameters – Challenges on the Horizon



Blue: correctly classified
Red: incorrectly classified

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

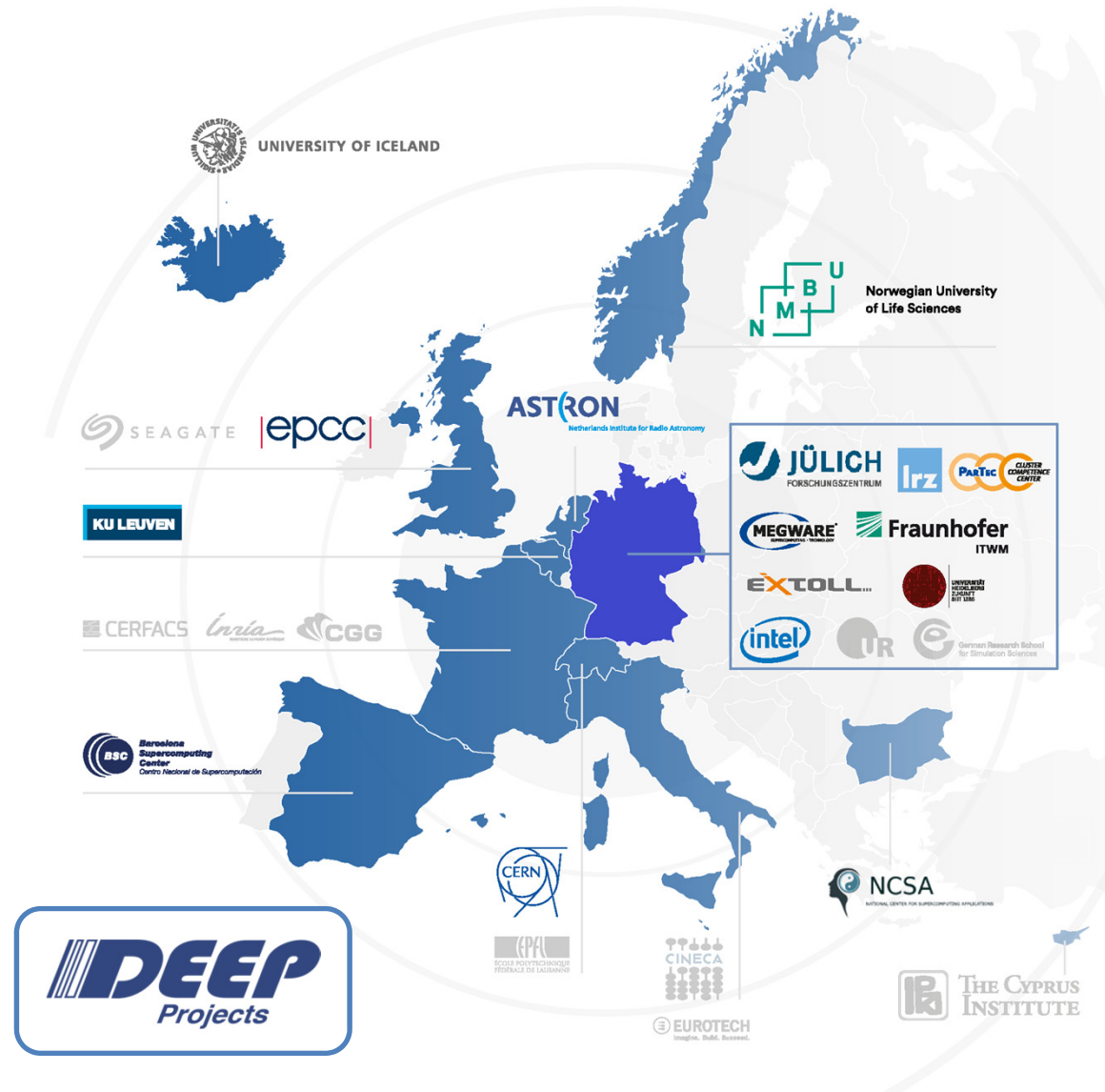
Feature	Representation / Value
Conv. Layer Filters	48, 32, 32
Conv. Layer Filter size	(3, 3, 5), (3, 3, 5), (3, 3, 5)
Dense Layer Neurons	128, 128
Optimizer	SGD
Loss Function	mean squared error
Activation Functions	ReLU
Training Epochs	600
Batch Size	50
Learning Rate	1
Learning Rate Decay	5×10^{-6}

- 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

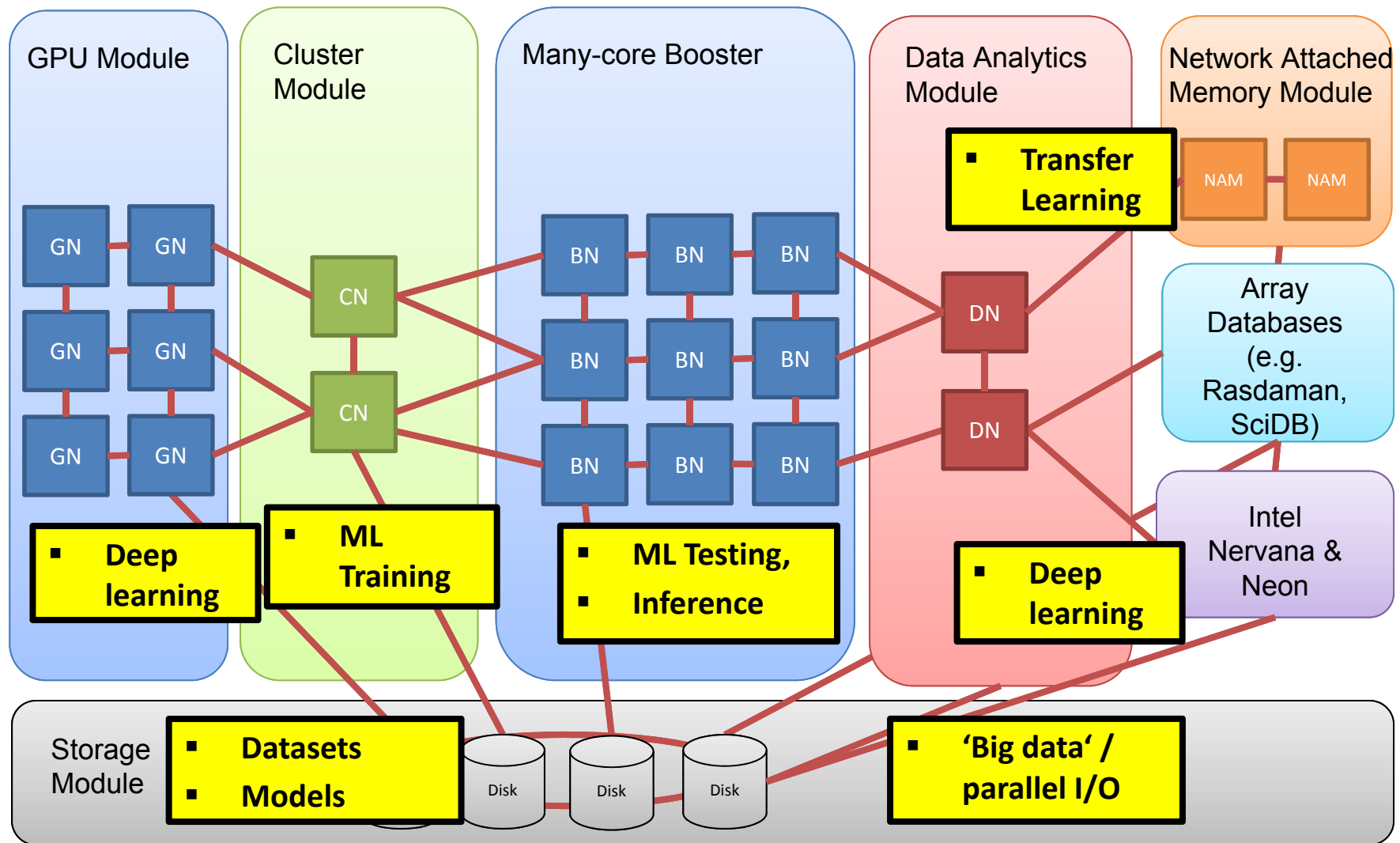
➤ [Link to ISC 2018 Machine Learning Track Keynote by Frank Hutter about hyper-parameter problems](#)

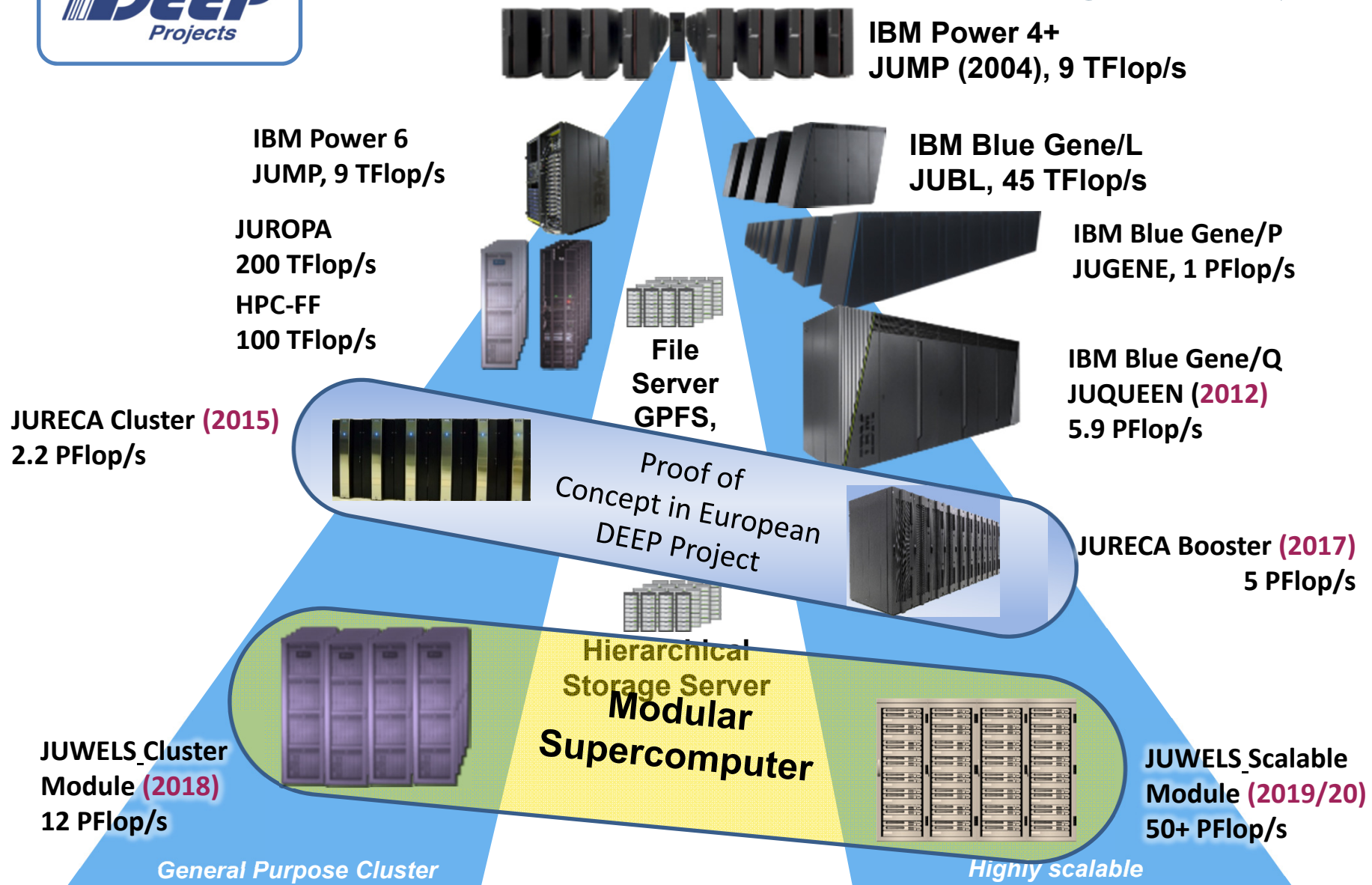
DEEP Projects & Partners

- 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-EST EU Project & Modular Supercomputing





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



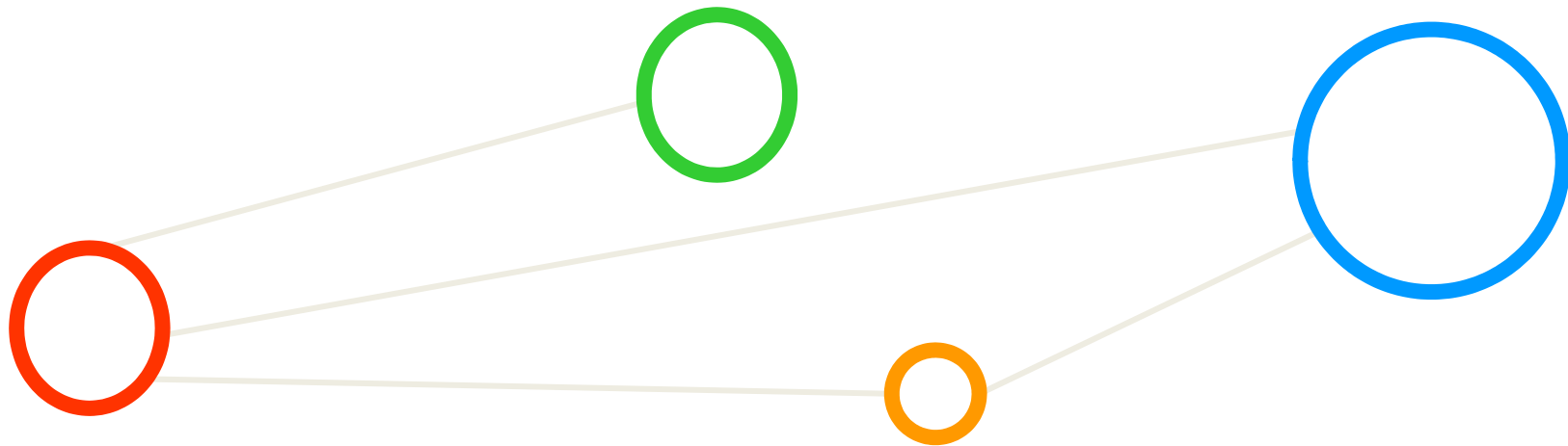
■ Toolset

- 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



➤ **Challenges: intertwine physical models with machine learning & finding good hyperparameters**

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 organization 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_YlosYk&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 discovering clusters 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/>

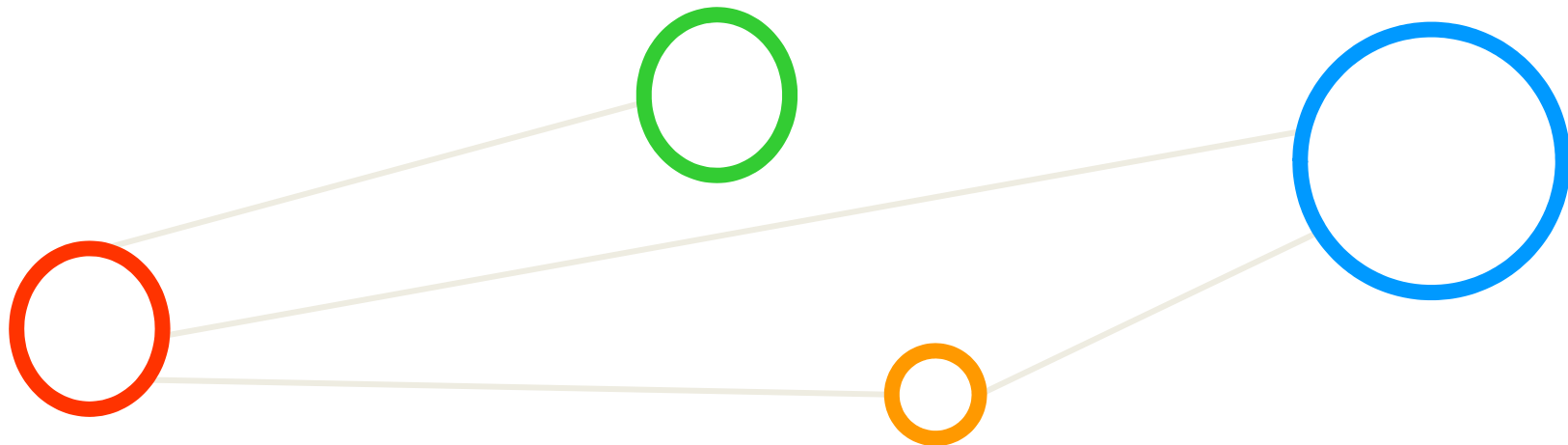
Lecture Bibliography (4)

- [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>
- [39] Indian Pines dataset: 220 Band AVIRIS Hyperspectral Image
Online: <https://purrr.purdue.edu/publications/1947/1>
- [40] Markus Goetz, PhD Thesis University of Iceland, 'Scalable Data Analysis in High Performance Computing',
Online:
<https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=4&cad=rja&uact=8&ved=0ahUKEwilv6jDjYLcAhWwsaQKHTTEB7QQFgg9MAM&url=https%3A%2F%2Fopinvisindi.is%2Fbitstream%2Fhandle%2F20.500.11815%2F472%2Fthesis.pdf%3Fsequence%3D1%26isAllowed%3Dy&usg=AOvVaw2Rs3jgLDQ4PU4SY24flmvQ>
- [41] A. Sergeev, M. Del Balso, 'Horovod: fast and easy distributed deep learning in TensorFlow', 2018
Online: <https://arxiv.org/abs/1802.05799>

Lecture Bibliography (5)

- [42] Timeless Texts, Cutting-Edge Code: Free downloads of Shakespeare from Folger Digital Texts,
Online: <http://www.folgerdigitaltexts.org/download/>
- [43] A. Karpathy and F. Li, 'Deep Visual-Semantic Alignments for Generating Image Descriptions', Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015,
Online: <https://cs.stanford.edu/people/karpathy/cvpr2015.pdf>
- [44] Adventures in Machine Learning, Keras LSTM tutorial,
Online: <http://adventuresinmachinelearning.com/keras-lstm-tutorial/>
- [45] Kaggle, 'UMICH SI650 – Sentiment Classification',
Online: <https://www.kaggle.com/c/si650winter11>
- [46] YouTube Video, 'Recurrent Neural Networks - Ep. 9 (Deep Learning SIMPLIFIED)',
Online: <https://www.youtube.com/watch?v=aCuOwF1ZjU&t=7s>
- [47] YouTube Video, 'Sequence Models and the RNN API (TensorFlow Dev Summit 2017)',
Online: <https://www.youtube.com/watch?v=RIR-Xlbp7s>
- [48] JURECA HPC System @ JSC,
Online: http://www.fz-juelich.de/ias/jsc/EN/Expertise/Supercomputers/JURECA/JURECA_node.html
- [49] Michael Stephan, 'Portable Parallel IO - 'Handling large datasets in heterogeneous parallel environments',
Online: http://www.fz-juelich.de/SharedDocs/Downloads/IAS/JSC/EN/slides/parallel-io-2014/parallel-io-hdf5.pdf?__blob=publicationFile

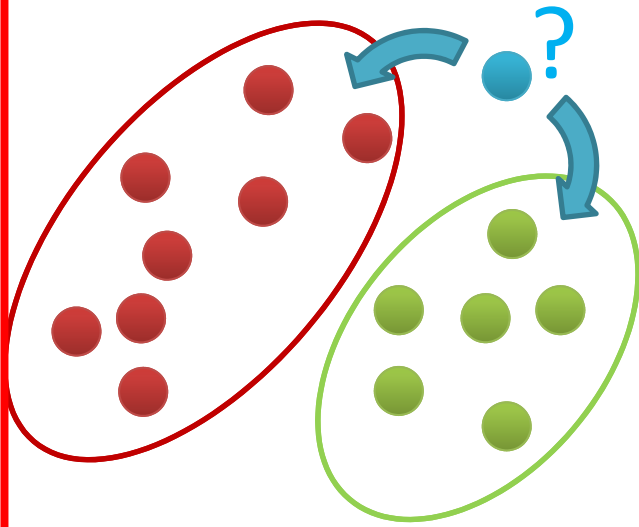
Appendix A: Introduction to Machine Learning



Methods Overview

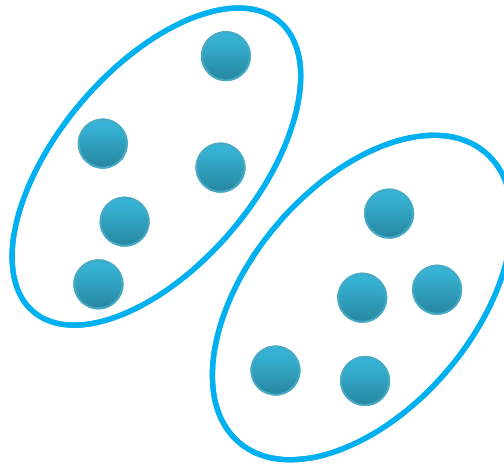
- Machine learning methods can be roughly categorized in classification, clustering, or regression augmented with various techniques for data exploration, selection, or reduction

Classification



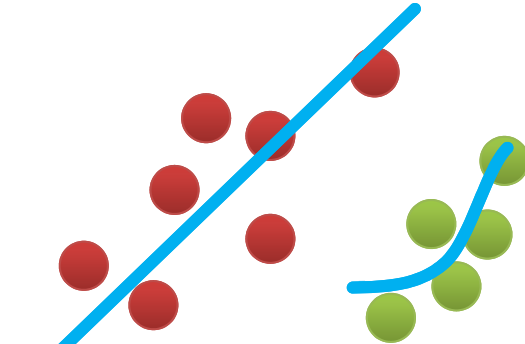
- Groups of data exist
- New data classified to existing groups

Clustering



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

Regression

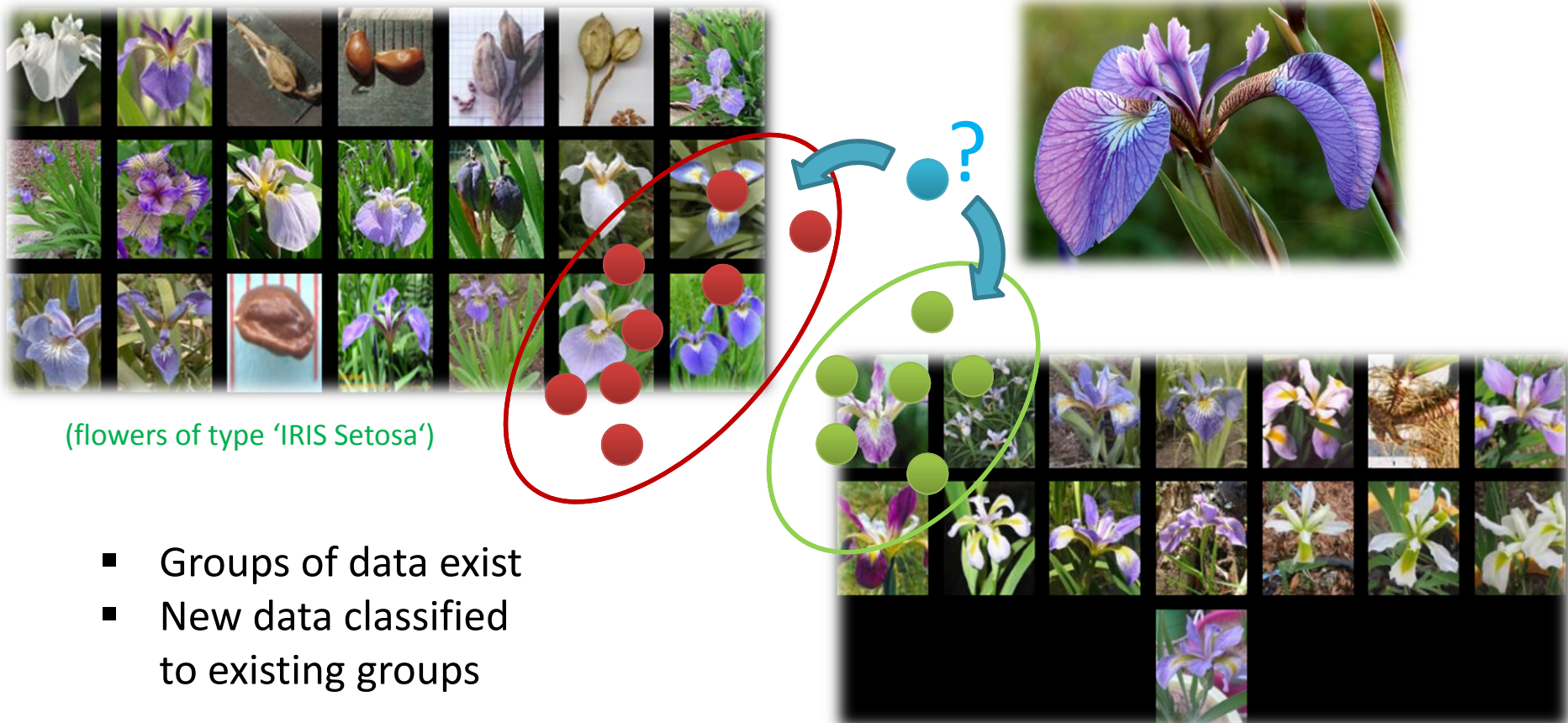


- Identify a line with a certain slope describing the data

Simple Application Example: Classification of a Flower

(1) Problem Understanding Phase

(what type of flower is this?)



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

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

(flowers of type 'IRIS Virginica')

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

(2) Data Understanding Phase

[6] UCI Machine Learning
Repository Iris Dataset

(four data attributes for each
sample in the dataset)

(one class label for each
sample in the dataset)



[5] Image source: Wikipedia, Sepal

- 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

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 discrepancy information
   (Subject, title, or context)

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

   ...

5. Number of Instances: 150 (50 in each of three classes)
   (number of samples, instances)

6. Number of Attributes: 4 numeric, predictive attributes and the
   class
   (attribute information)

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
   (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

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

(four data attributes for each sample in the dataset)

(one class label for each sample in the dataset)

- 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

OK Cancel

[7] Rattle Library for R

Preparing the Data – Corrected Header

(3) Data Preparation Phase

Rattle Dataset - dfedit version 0.6.1

	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	5.1	3.0	1.3	0.4	Iris-setosa

(correct header information, resulting in 150 data samples)

R Data Miner - [Rattle (iris.data)]

Project Tools Settings Help

Execute New Open Save Report Export Stop Quit

Data Explore Test Transform Cluster Associate Model Evaluate Log

Source: ☒ Spreadsheet ☐ ARFF ☐ ODBC ☐ R Dataset ☐ RData File

Filename: Separator: Decimal: ☒ Header

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

OK Cancel

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’

The image shows two screenshots of the Rattle Dataset window, illustrating the process of removing the third class (Iris-Versicolour) from the dataset.

Left Window (Initial Dataset): Shows a table with columns X5.1, X3.5, X1.4, X0.2, and Iris.setosa. The data includes 150 samples, including Iris-Versicolour. A green text overlay states: "(three class problem with N = 150 samples including Iris Versicolour)". A green text overlay at the bottom states: "(remove Versicolour class samples from dataset)".

Right Window (Filtered Dataset): Shows a table with columns V1, V2, V3, V4, and V5. The data includes 100 samples, excluding Iris-Versicolour. A green text overlay states: "(wo class problem with N = 100 samples excluding Iris Versicolour)". A green text overlay at the bottom states: "(export or save dataset to iris-twoclass.data)".

A large red arrow points from the left window to the right window, indicating the transformation of the dataset.

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

Left window (Full Dataset):

	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

Annotations for Left window:

- ~~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

Right window (Selected Dataset):

	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

Annotations for Right window:

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

Iris Dataset – Open Data

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

The screenshot displays the B2SHARE web interface for the 'Iris Dataset LibSVM Format Preprocessing' record. The browser address bar shows the URL: <https://b2share.eudat.eu/records/37fb24847a73489a9c569d7033ad0238>. The record is by Morris Riedel, dated Dec 22, 2016, and last updated on Jan 11, 2018. The abstract describes the UCI Machine Learning Repository IRIS Dataset, providing details on the data classes and sampling. The keywords are LibSVM, Iris, Flowers, and UCI. The PID is 11304/10e216d4-0a98-4ab4-86ea-75ed05ee0f46. The interface is divided into two main sections: 'Files' and 'Basic metadata'. The 'Files' section lists eight files with their names and sizes. The 'Basic metadata' section provides details on open access, license, contact email, publication date, contributors, resource type, alternate identifiers, and publisher.

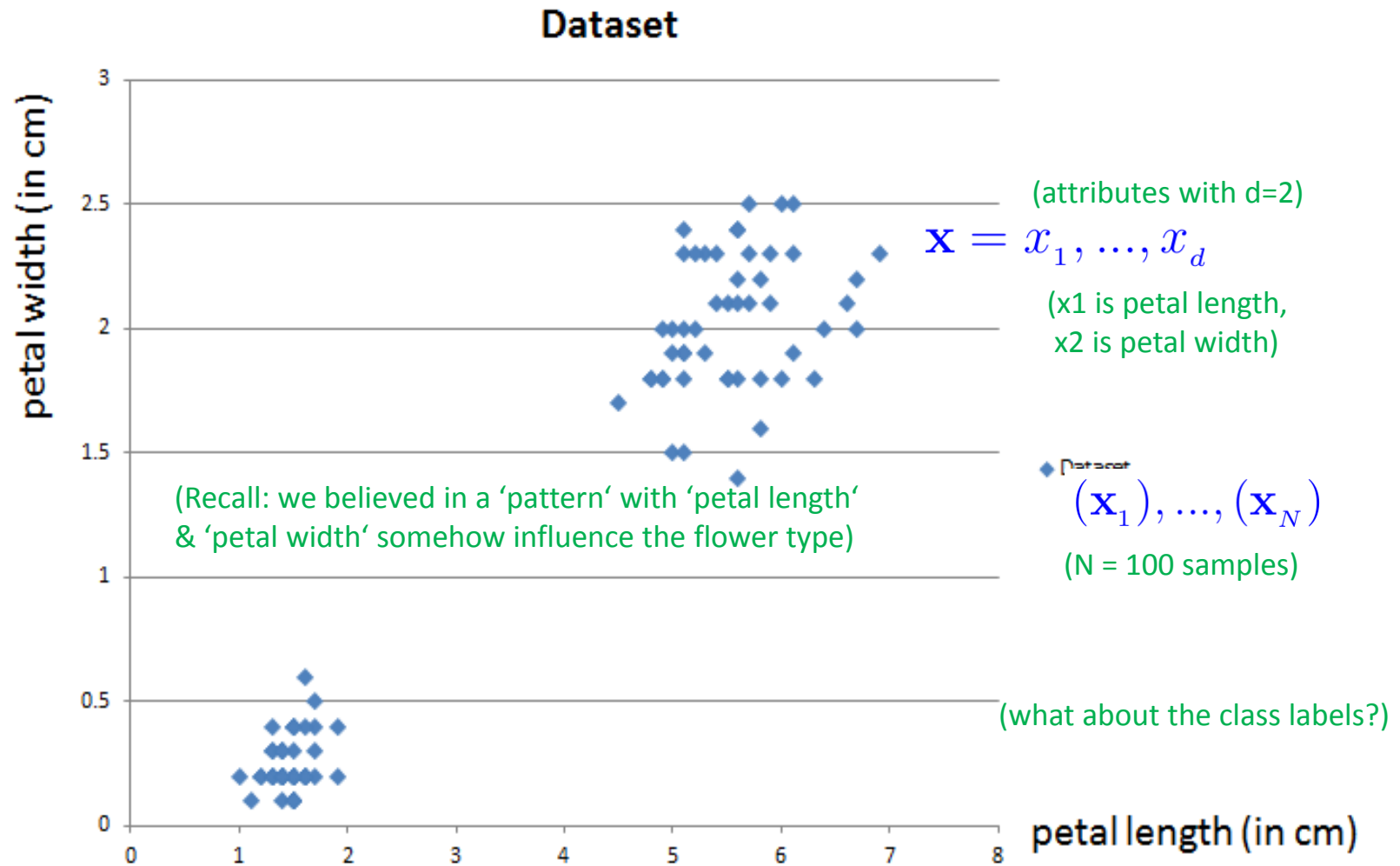
Name	Size
iris-class1and3-testing.txt	2.74KB
iris-class1and3-training.txt	1.81KB
iris-class1and3.txt	4.54KB
iris-class2and3-testing.txt	2.84KB
iris-class2and3-training.txt	3.18KB
iris-class2and3.txt	4.66KB
iris.scale.original.original	6.95KB
iris.scale.scale	6.95KB

Basic metadata		
Open Access	True ✓	
License		
Contact Email	m.riedel@fz-juelich.de	
Publication Date	2016-07-03	
Contributors		
Resource Type	Category	Other
Alternate identifiers	397	
	Type	B2SHARE_V1_ID
	http://hdl.handle.net/11304/b68b5707-ec19-45bf-8da9-73503aa4d1e1	
	Type	ePIC_PID
Publisher	http://b2share.eudat.eu	

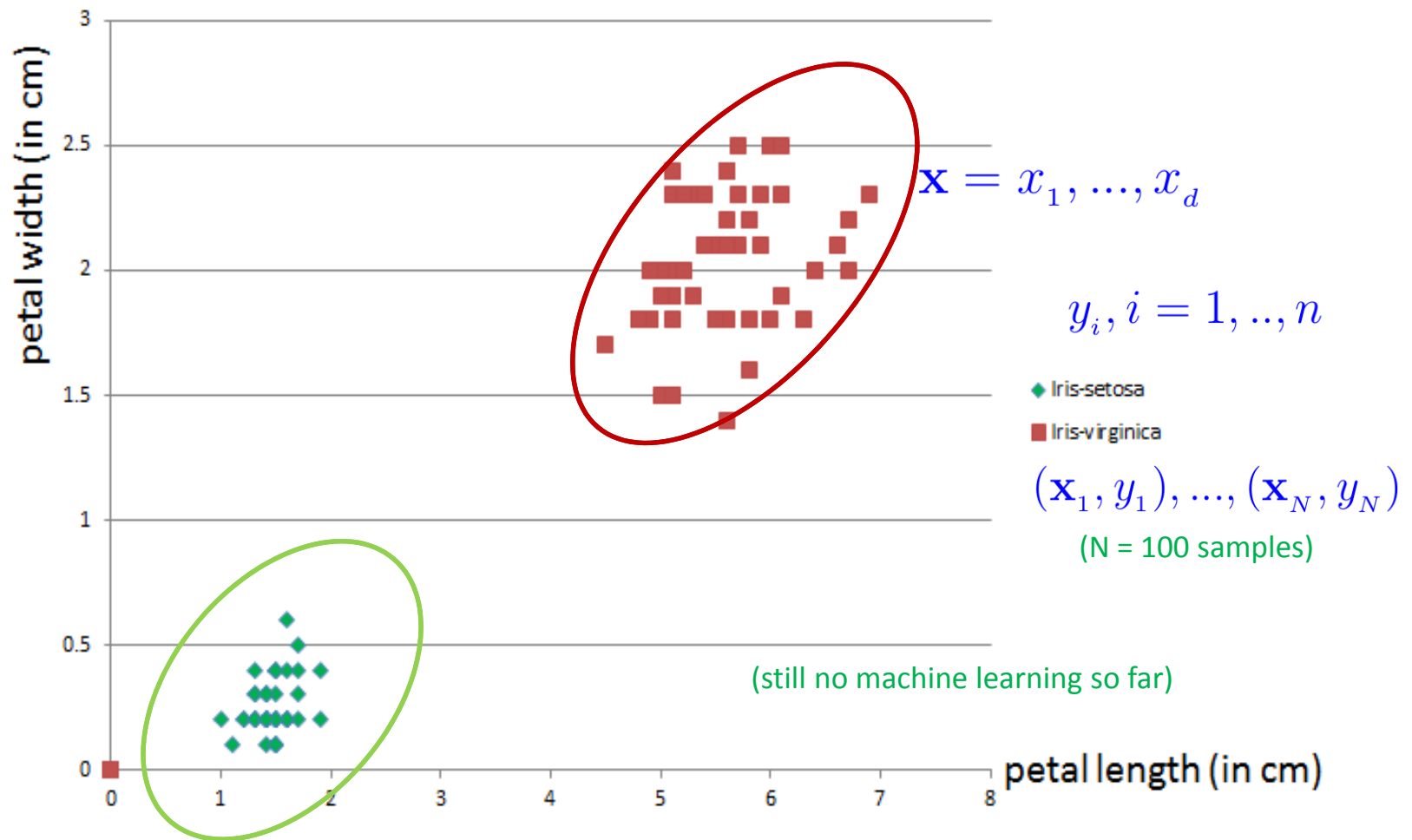
[14] Iris Dataset



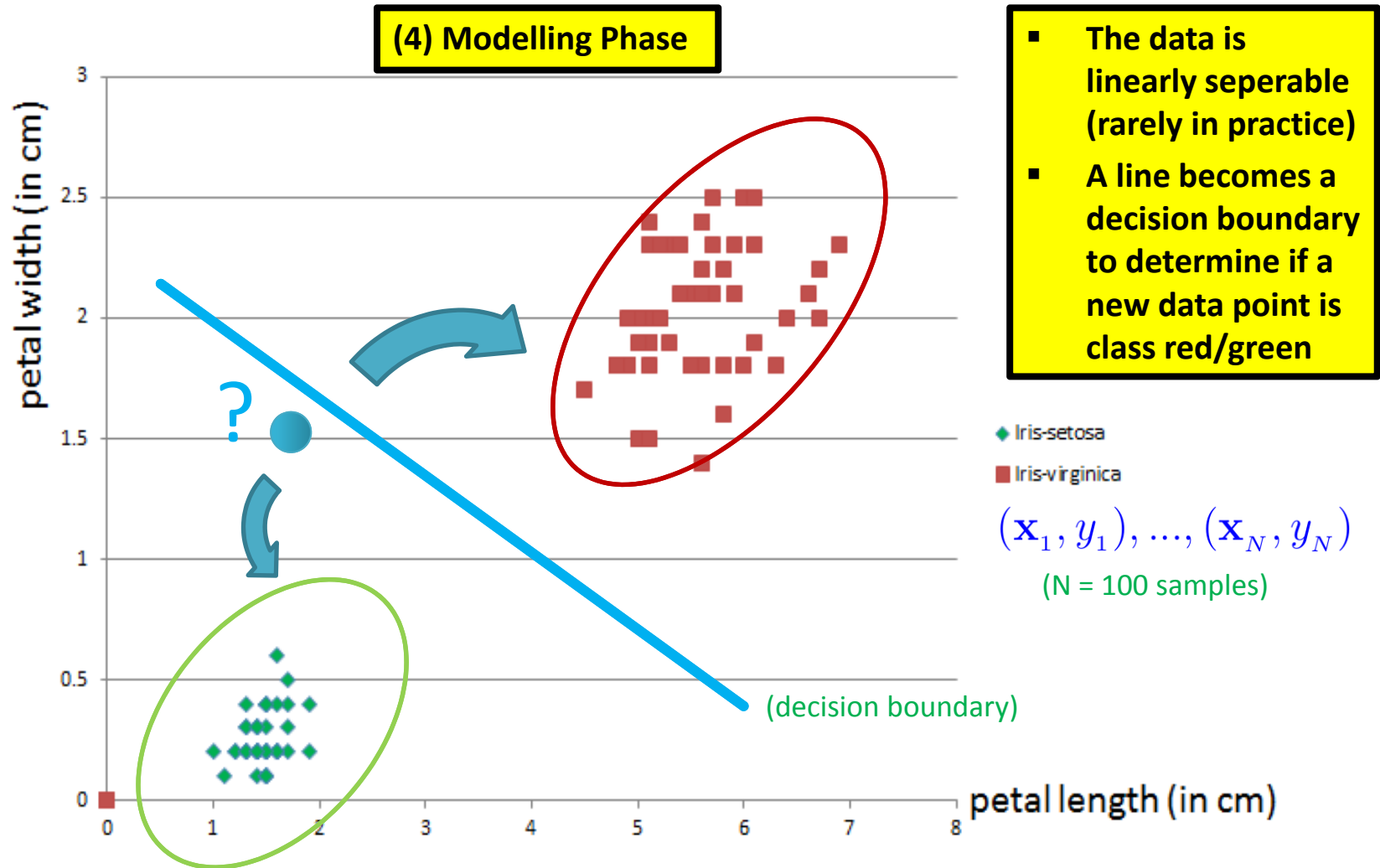
Check Preparation Phase: Plotting the Data



Check Preparation Phase: Class Labels

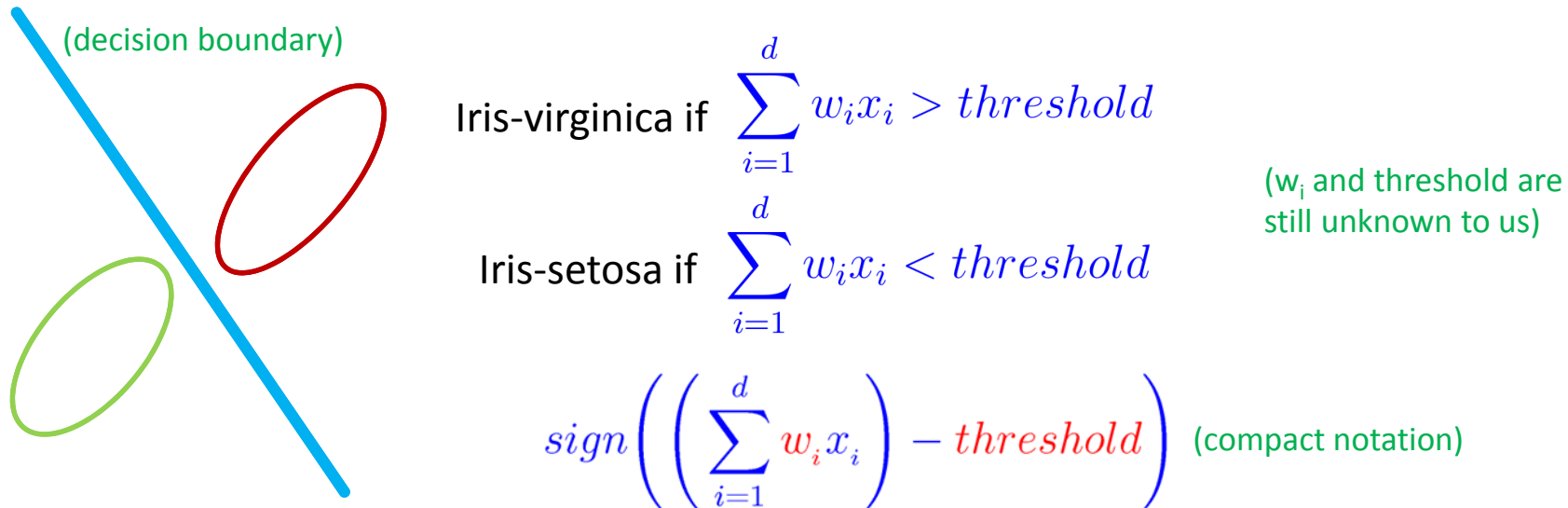


Linearly Seperable Data & Linear Decision Boundary

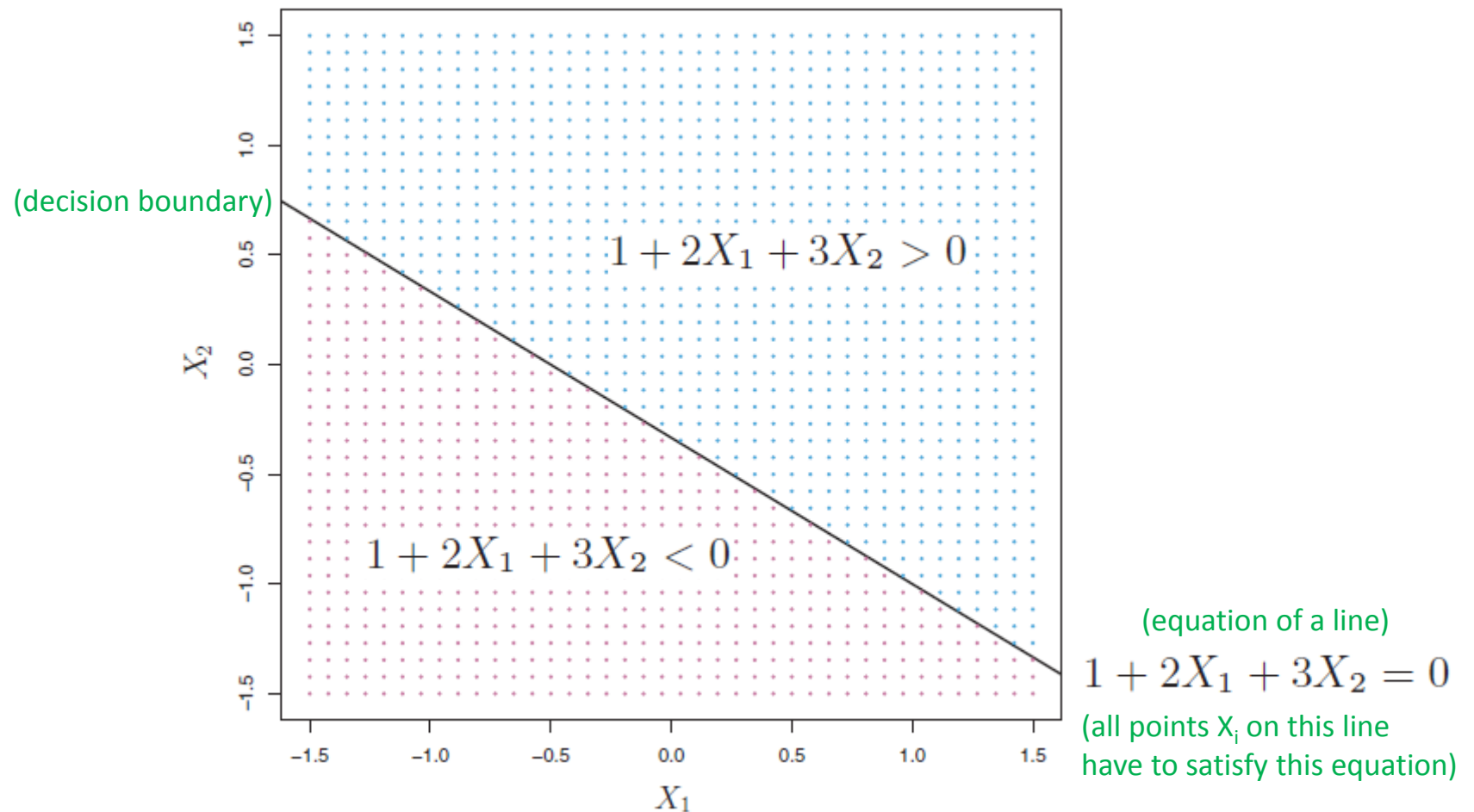


Separating Line & Mathematical Notation

- Data exploration results
 - A line can be crafted between the classes since linearly separable 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: $\mathbf{X} = x_1, \dots, x_d$ (attributes of flowers)
 - Output: class +1 (Iris-virginica) or class -1 (Iris-setosa)



Separating Line & 'Decision Space' Example



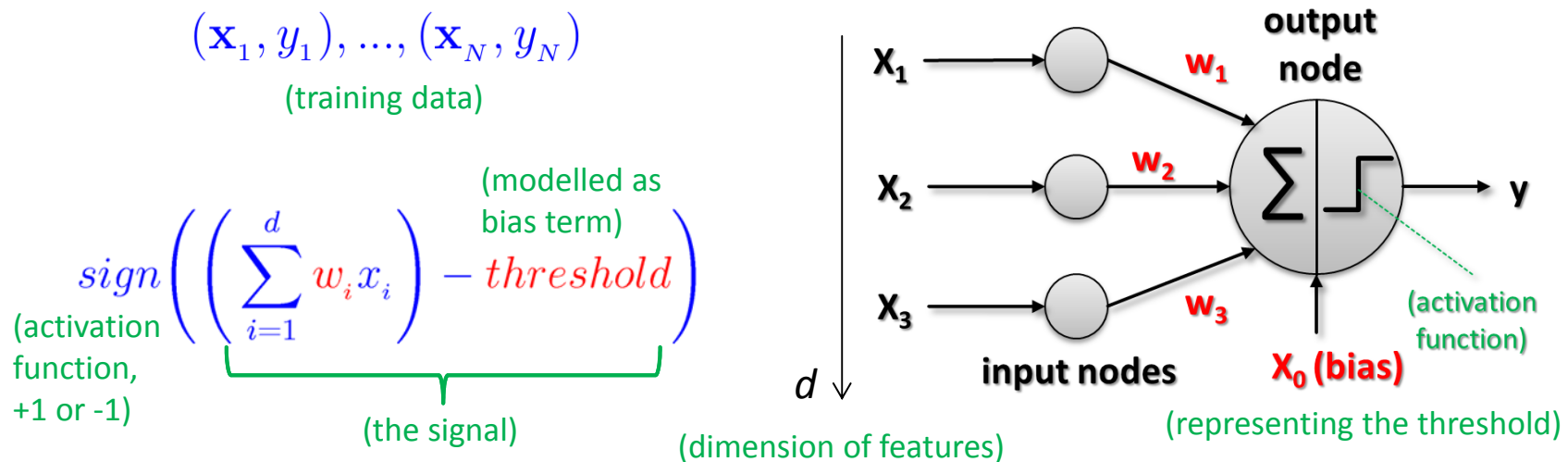
modified from [13] An Introduction to Statistical Learning

A Simple Linear Learning Model – The Perceptron

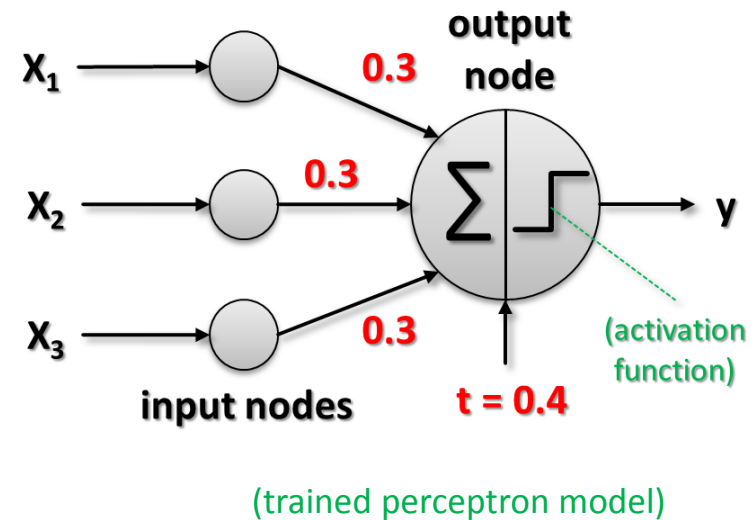
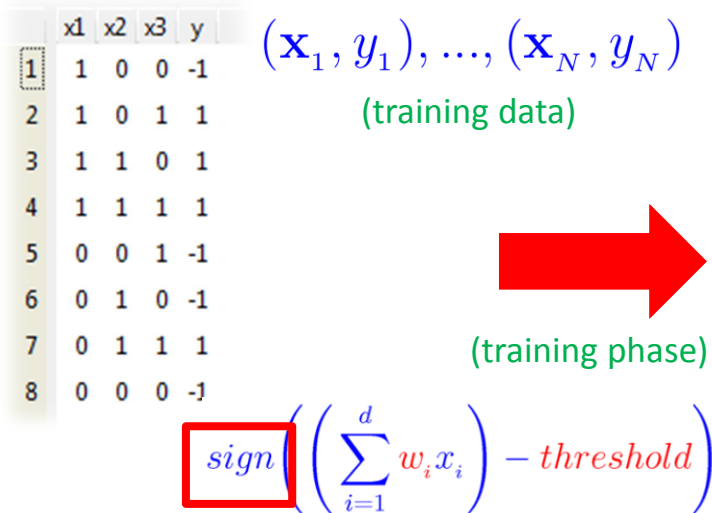
- Human analogy in learning

[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'



Perceptron – Example of a Boolean Function



■ 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

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

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

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

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

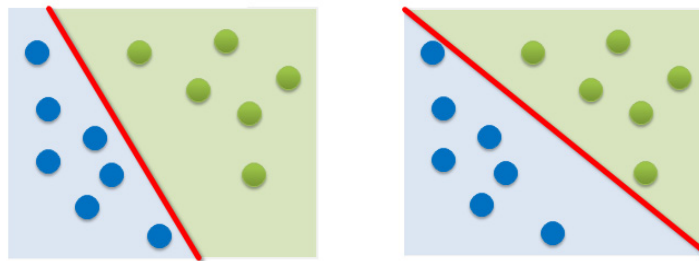
Summary Perceptron & Hypothesis Set $h(\mathbf{x})$

- 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: $\mathbf{X} = x_1, \dots, x_d$ (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**

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

(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 red line in graphics)

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

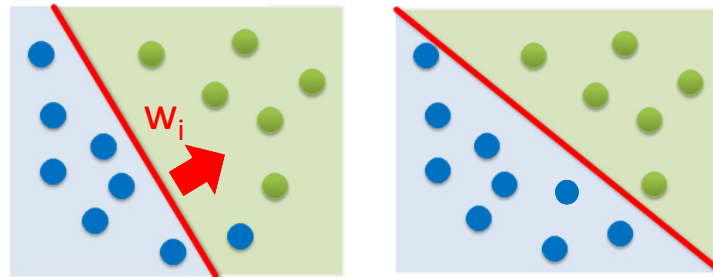
Perceptron Learning Algorithm – Understanding Vector W

- When: If we believe there is a **linear pattern** to be detected
 - Assumption: **Linearly seperable data** (lets the algorithm converge)
 - Decision boundary: perpendicular vector \mathbf{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)



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

(vector notation, using T = transpose)

$$\mathbf{w}_i = (w_{i1}, w_{i2}, \dots, w_{id})$$

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

$$\mathbf{x}_i = (x_{i1}, x_{i2}, \dots, x_{id})$$

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

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

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

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

(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’

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

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

(our example)

- Given two vectors

- Multiplying corresponding components of the vector

- Then adding the resulting products

- Simple example: $(2, 3) \cdot (4, 1) = 2 * 4 + 3 * 1 = 11$ (a scalar!)

- Interesting: Dot product of two vectors is a scalar

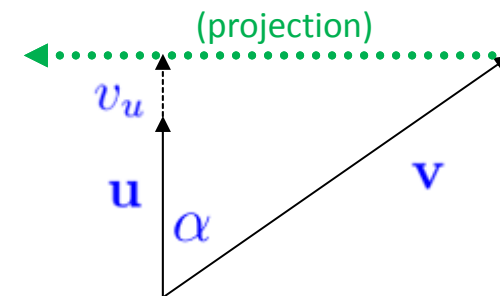
- ‘Projection capabilities of Dot product’ (simplified)

- 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

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



➤ Dot Products are important in machine learning, e.g. in Support Vector Machines, see Appendix C

Perceptron Learning Algorithm – Learning Step

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

(one point at a time is picked)

- Pick one misclassified training point where:

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

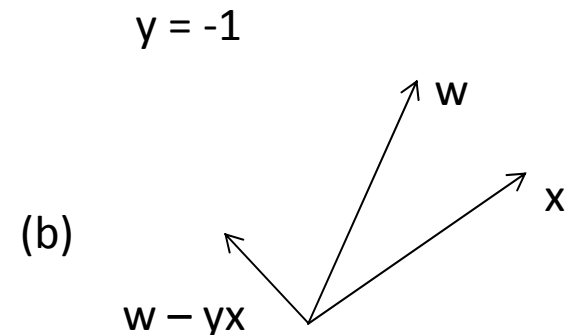
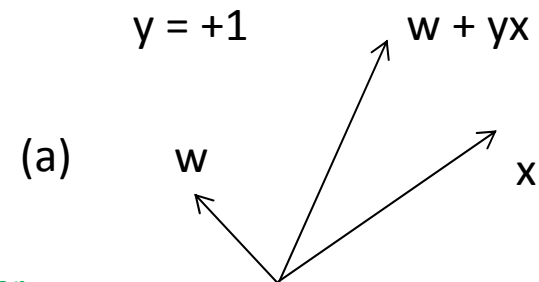
- Update the weight vector:

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

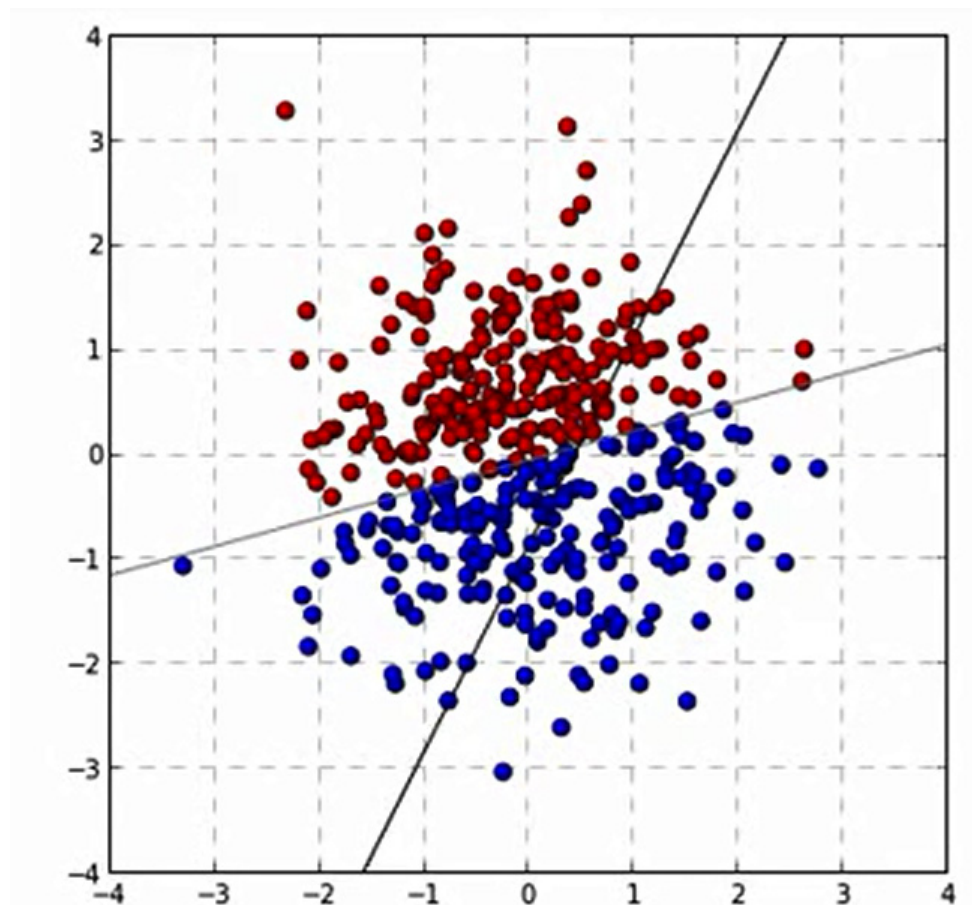
(y_n is either +1 or -1)

- Terminates when there are no misclassified points

(converges only with linearly separable data)



[Video] Perceptron Learning Algorithm



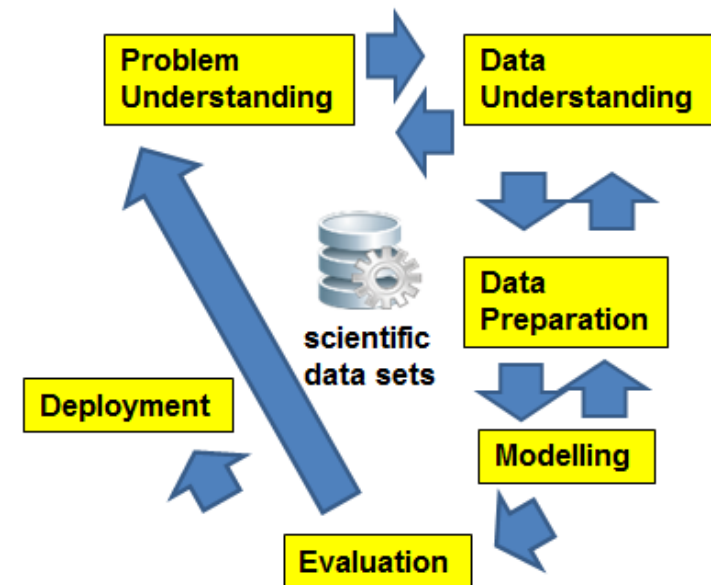
[10] PLA Video

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)



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

- Lessons Learned from Practice

- Go back and forth between the different six phases

➤ A more detailed description of all six CRISP-DM phases is in the Appendix B 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

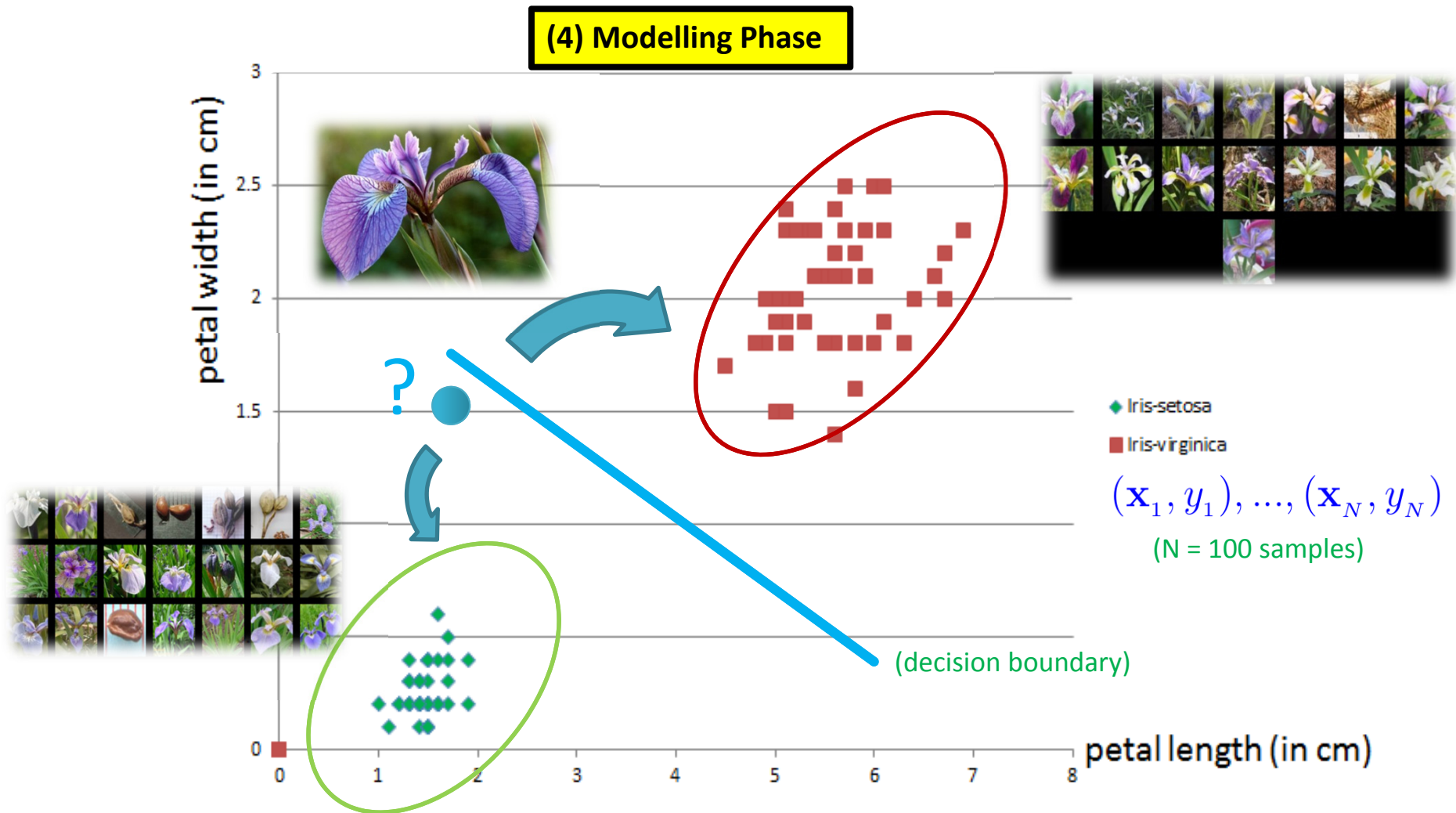
- Predictive Tasks

- 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: $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)$
- **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’ (out 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)

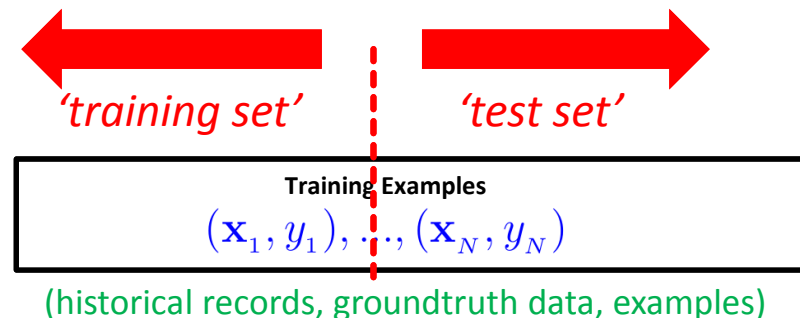
(4) Modelling Phase

(5) Evaluation Phase

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

- 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 separation 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’



Model Evaluation – Testing Phase & Confusion Matrix

- 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
 - Counts of test records that are correctly predicted
 - E.g. create confusion matrix for a two class problem

(5) Evaluation Phase

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

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

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 methods in practice)

- Accuracy (usually in %)

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

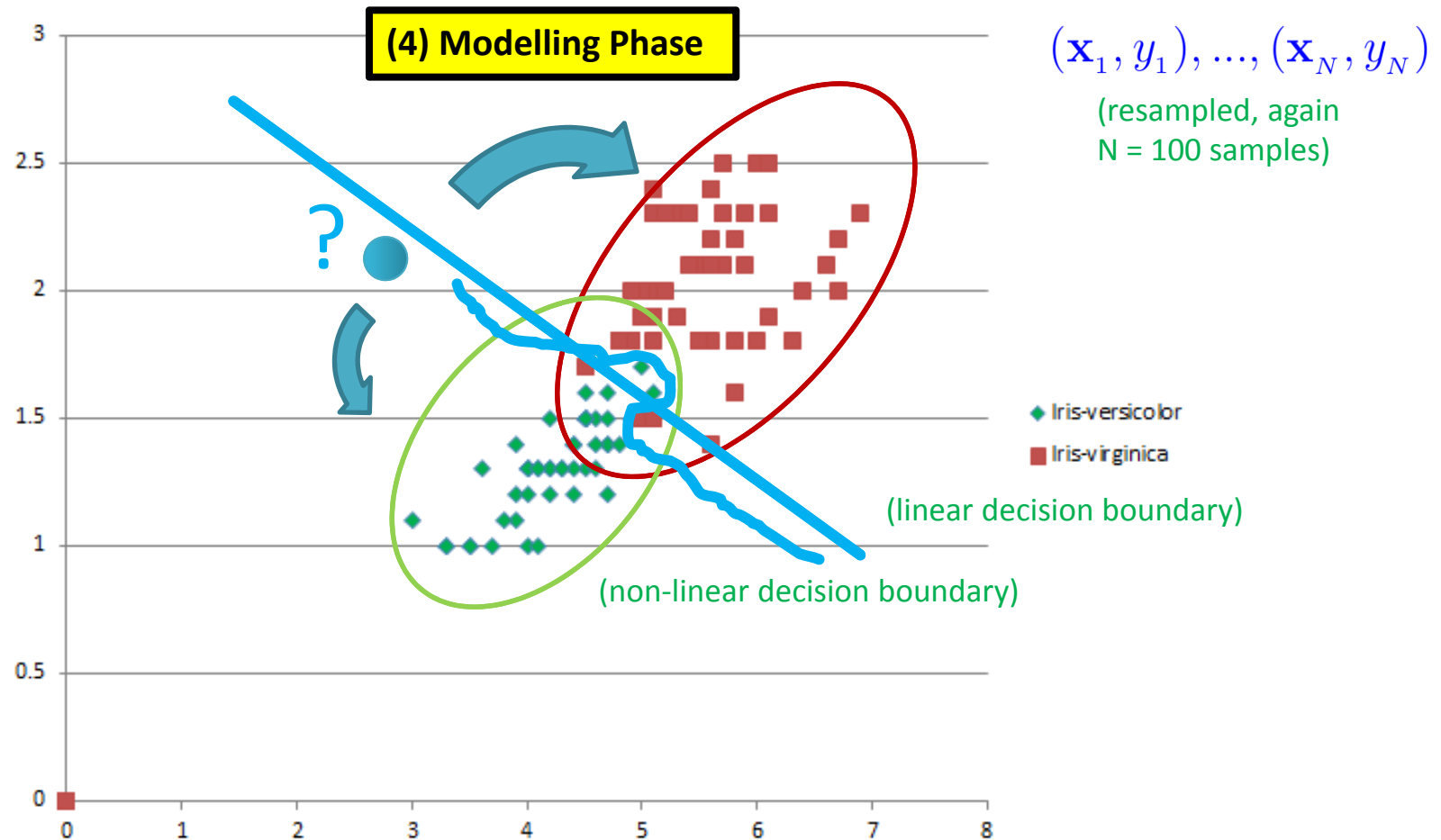
- Error rate

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

- If model evaluation is satisfactory:

(6) Deployment Phase

Non-linearly Seperable Data in Practice – Which model?



(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)

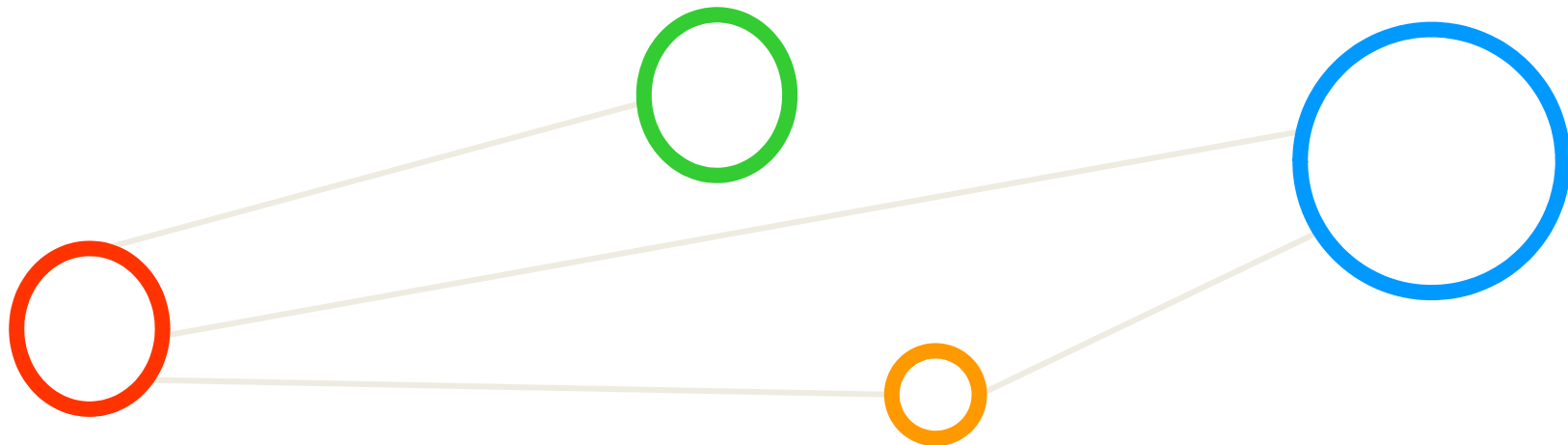
- Key challenges faced when doing traditional data analysis and machine learning are scalability, high dimensionality of datasets, heterogenous and complex data, data ownership & distribution
- Combat 'overfitting' is the key challenge in machine learning using validation & regularization

Prevent Overfitting for better 'out-of-sample' generalization



[15] Stop Overfitting, YouTube

Appendix B: CRISP-DM Process

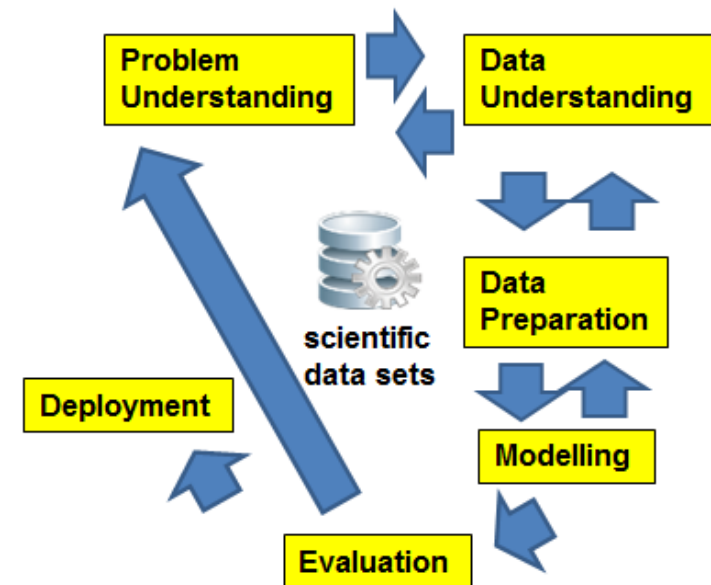


Summary: Systematic Process

- 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

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



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

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

- Task A – Determine Business Objectives

[12] CRISP-DM User Guide

- Background, Business Objectives, Business Success Criteria

- Task B – Situation Assessment

- Inventory of Resources, Requirements, Assumptions, and Constraints
 - 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

- Task A – Evaluate Results

[12] CRISP-DM User Guide

- 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

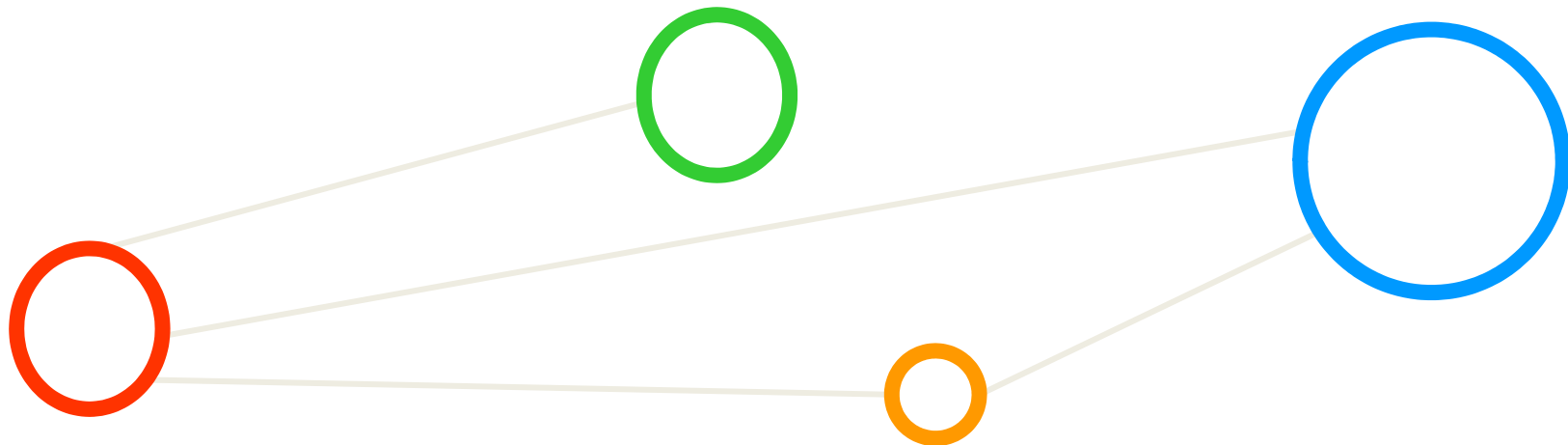
6 – Deployment

- 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

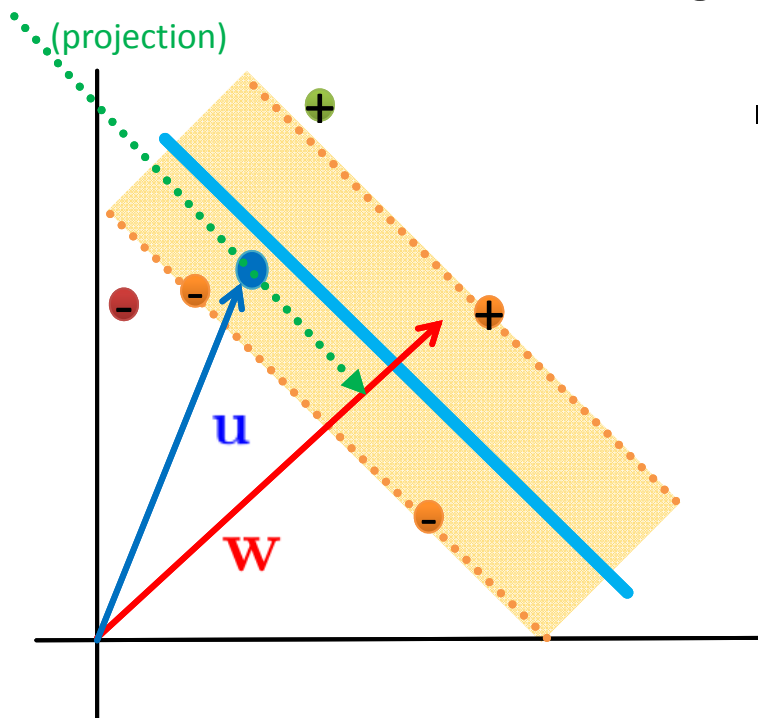
- Task A – Plan Deployment
 - Establish a deployment plan
- Task B – Plan Monitoring and Maintenance
 - Create a monitoring and maintenance plan
- Task C – Product Final Report
 - Create final report and provide final presentation
- Task D – Review Project
 - Document experience, provide documentation

Appendix C: 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?

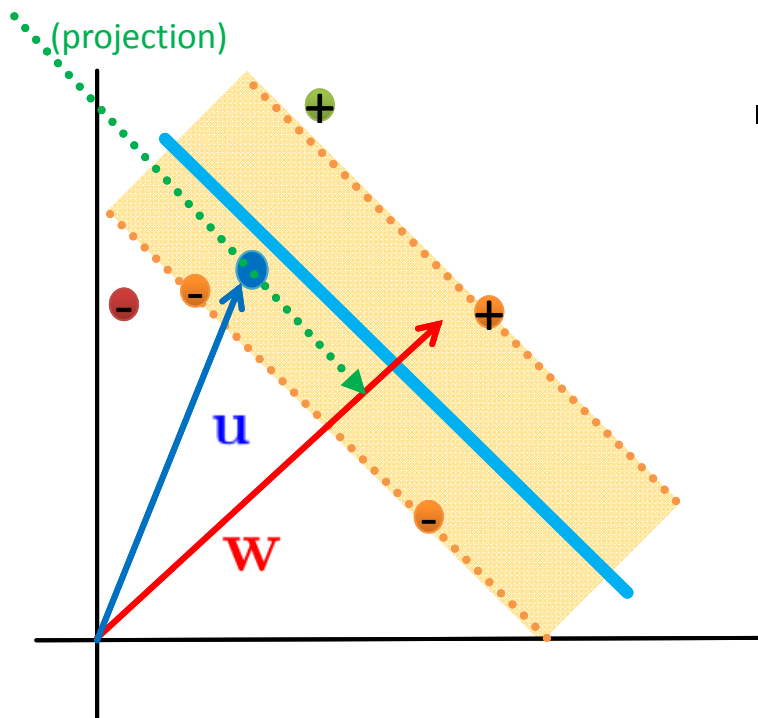


- Dot product $\mathbf{w} \cdot \mathbf{u} \geq C; C = -b$
 - With \mathbf{u} takes the projection on the \mathbf{w}
 - Depending on where projection is it is left or right from the decision boundary
 - Simple transformation brings decision rule:
- ① $\mathbf{w} \cdot \mathbf{u} + b \geq 0 \rightarrow$ 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 \oplus $\mathbf{w} \cdot \mathbf{x}_+ + b \geq 1$
 - Second constraint set for negative samples \ominus $\mathbf{w} \cdot \mathbf{x}_- + b \leq 1$
 - For **mathematical convenience** introduce variables (i.e. **labelled samples**)
 $y_i = +$ for \oplus and $y_i = -$ for \ominus

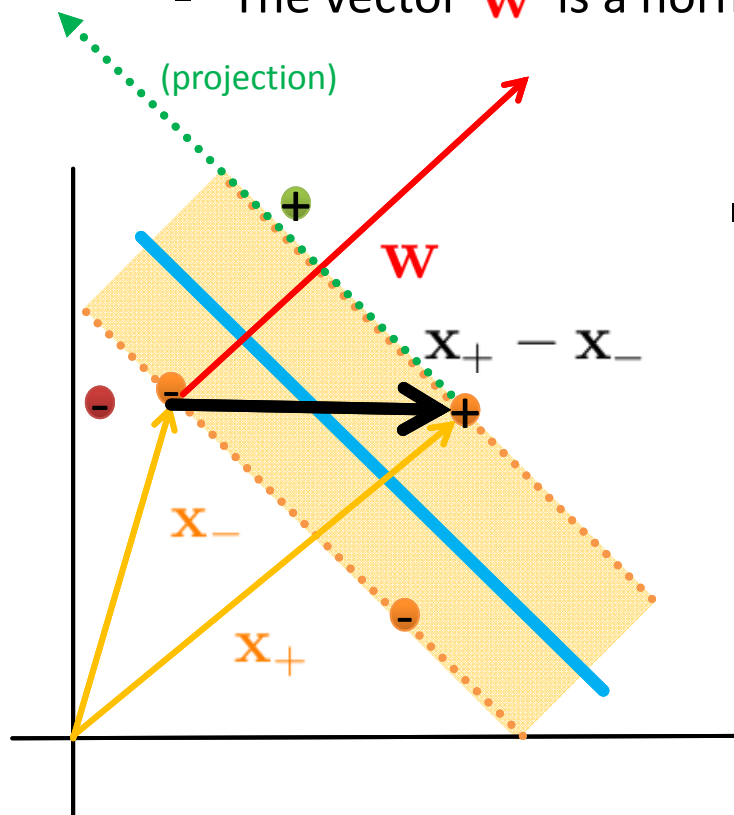


- Multiply equations by y_i
 - Positive samples: $y_i(\mathbf{x}_i \cdot \mathbf{w} + b) \geq 1$
 - Negative samples: $y_i(\mathbf{x}_i \cdot \mathbf{w} + b) \leq -1$
 - Both **same** due to $y_i = +$ and $y_i = -$
 (brings us mathematical convenience often quoted)

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

$$\begin{array}{c}
 \mathbf{x}_+ - \mathbf{x}_- \cdot \boxed{\frac{\mathbf{w}}{\|\mathbf{w}\|}} \quad (\text{unit vector}) \\
 \downarrow \quad \downarrow \\
 1 - b \quad 1 + b \quad \rightarrow \quad \frac{2}{\|\mathbf{w}\|} \quad \textcircled{3}
 \end{array}$$

- When enforce constraint: $y_i = +$ (green dot)

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

Constrained Optimization Steps SVM (1)

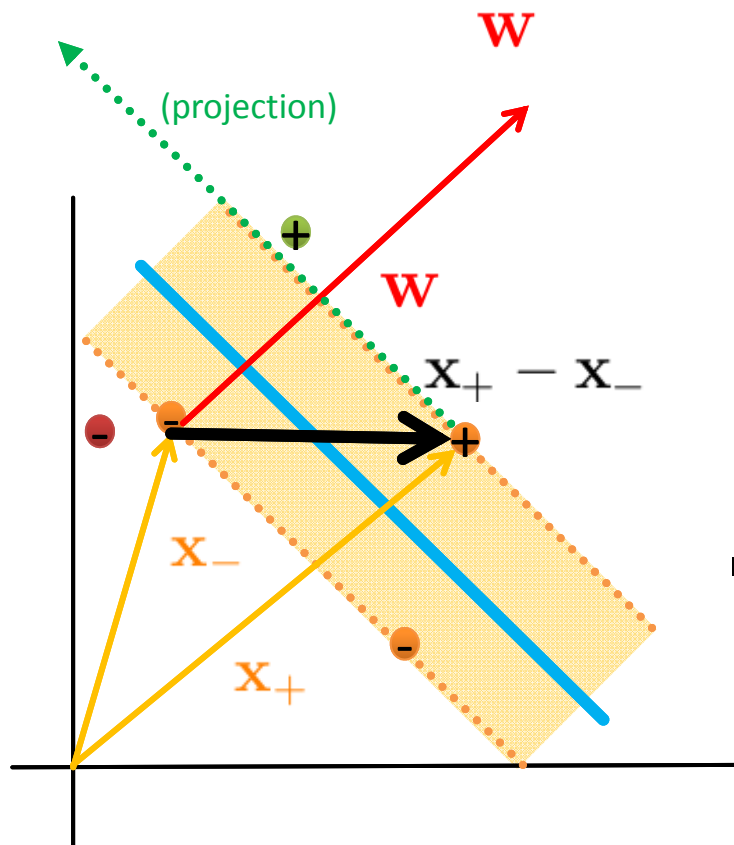
- Use 'constraint optimization' of mathematical toolkit

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

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

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

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



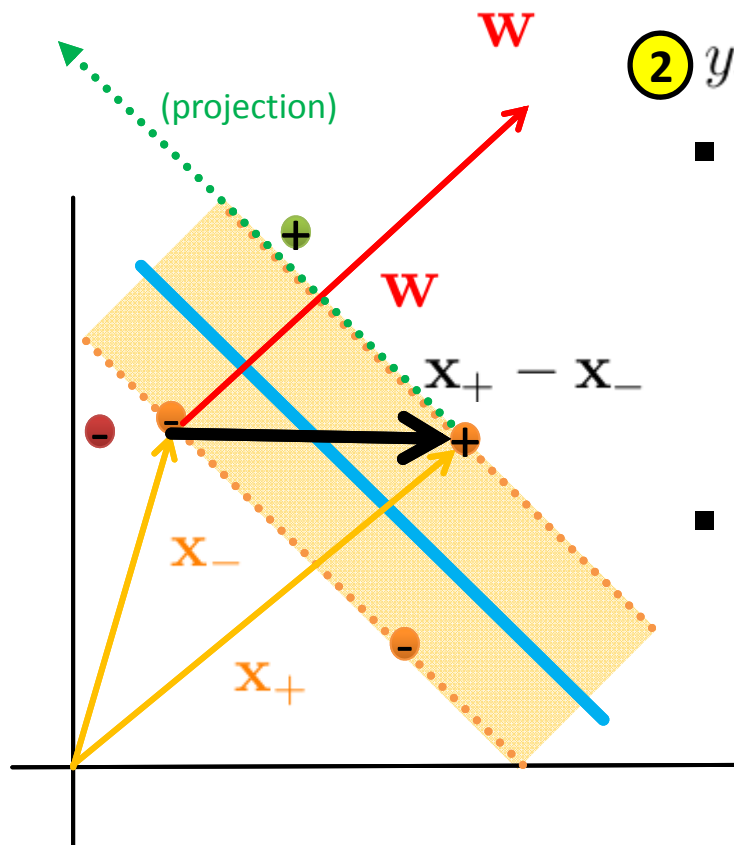
- Next: Find the extreme values

- 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}

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

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

- Simple transformation brings:

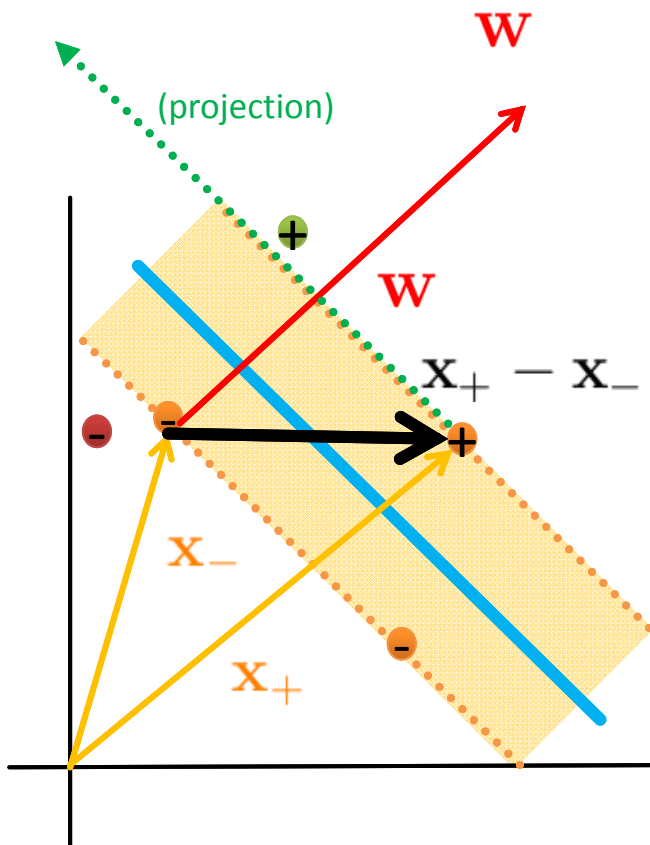
$$\textcircled{5} \mathbf{w} = \sum \alpha_i y_i \mathbf{x}_i$$

(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

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



Constrained Optimization Steps SVM (4)

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

- Find minimum

- Quadratic optimization problem

- Take advantage of $\mathbf{w} = \sum \alpha_i y_i \mathbf{x}_i$

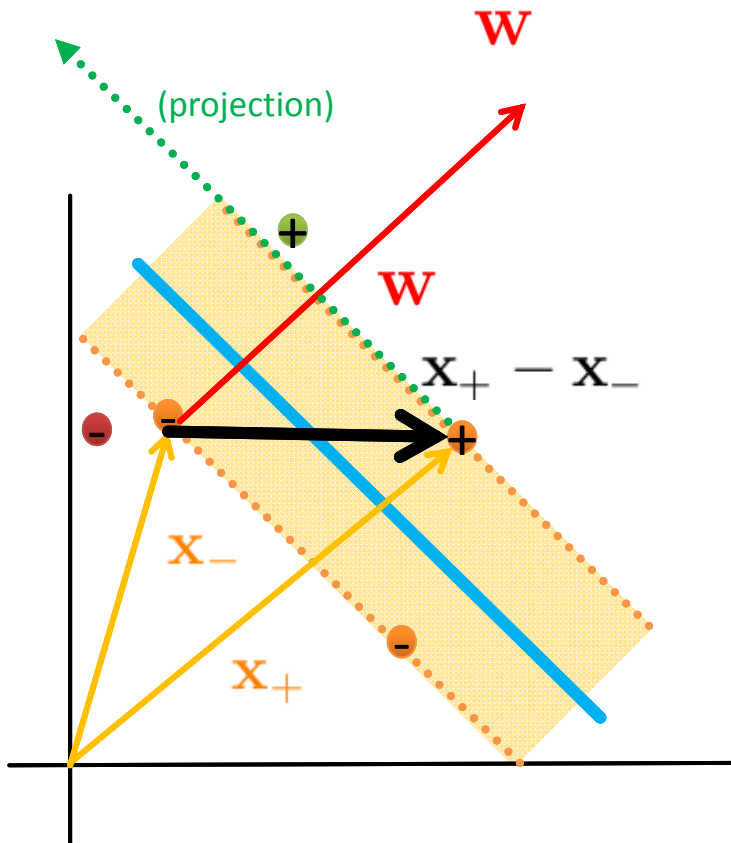
$$\mathcal{L} = \frac{1}{2} \left(\sum \alpha_i y_i \mathbf{x}_i \right) \cdot \left(\sum \alpha_j y_j \mathbf{x}_j \right)$$

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

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

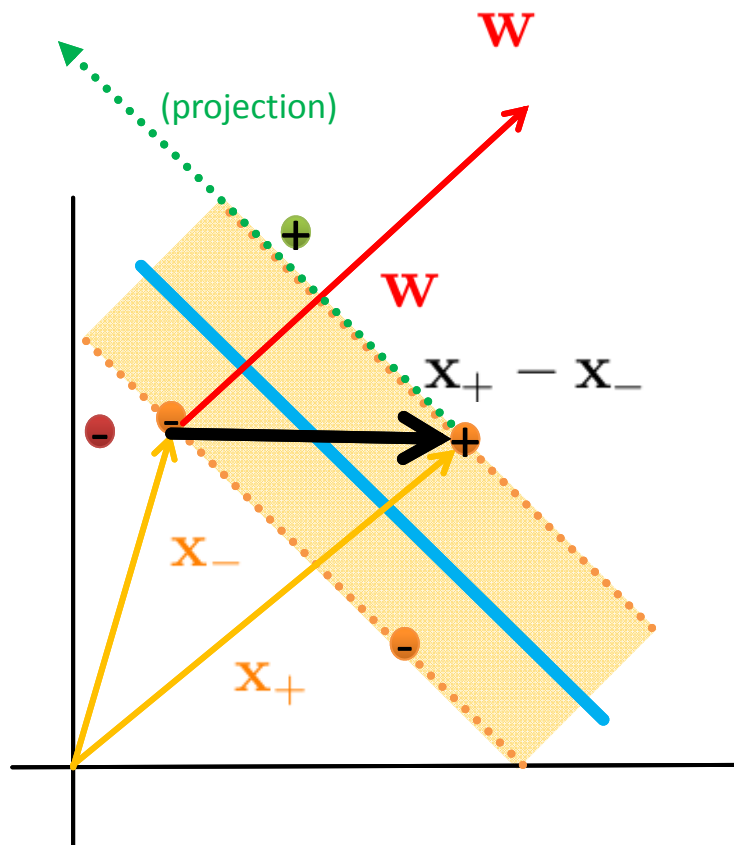
(b constant
in front sum)

$$\sum \alpha_i y_i = 0$$



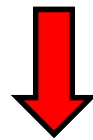
Constrained Optimization Steps SVM (5)

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



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

(was 0)



(results in)

(optimization depends only on dot product of samples)

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

- Equation to be solved by some quadratic programming package

Use of SVM Classifier to Perform Classification

- Use findings for decision rule

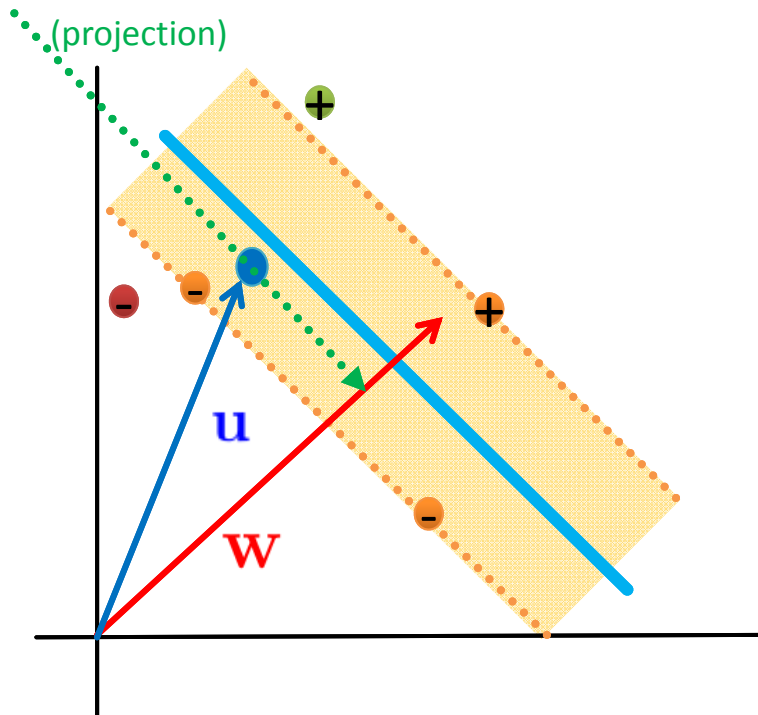
$$\textcircled{5} \mathbf{w} = \sum \alpha_i y_i \mathbf{x}_i$$

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



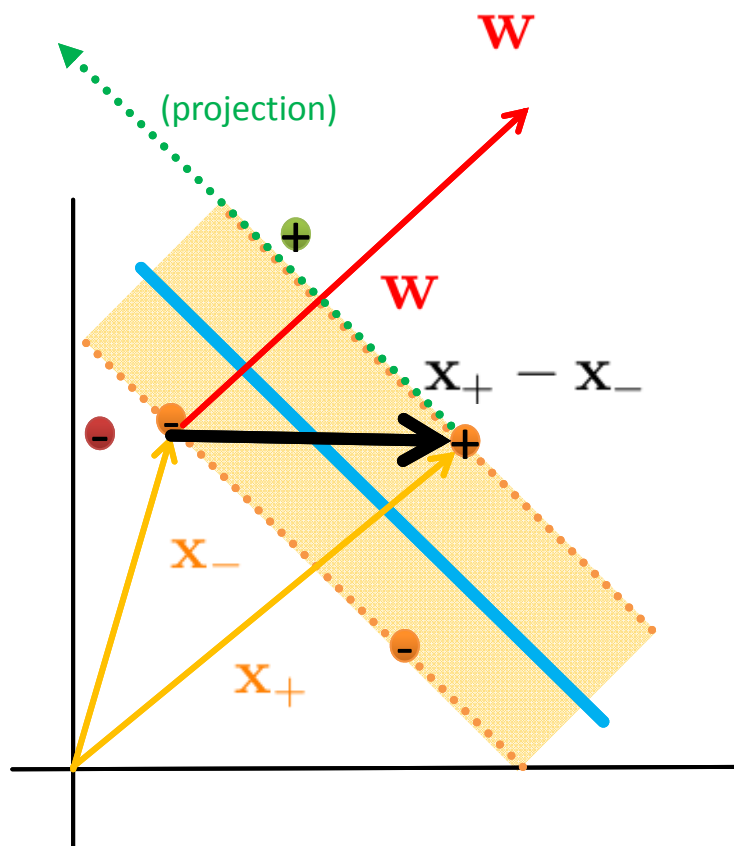
$$\sum \alpha_i y_i \mathbf{x}_i \cdot \mathbf{u}_i + b \geq 0 \quad +$$

(decision rule also
depends on
dotproduct)



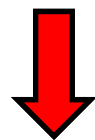
Constrained Optimization Steps SVM & Dot Product

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



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

(was 0)



(results in)

(optimization depends only on dot product of samples)

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

- Equation to be solved by some quadratic programming package

Kernel Methods & Dot Product Dependency

- Use findings for **decision rule**

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



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



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

(decision rule also depends on dotproduct)

- 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**)

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

$$\Rightarrow \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j) \quad (\text{in optimization})$$

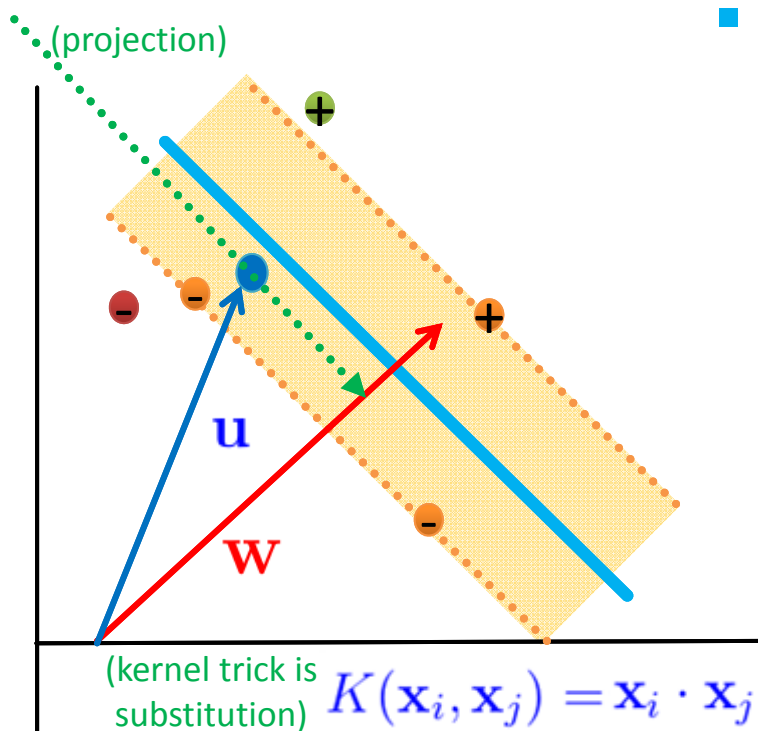
(optimization depends only on dot product of samples)

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

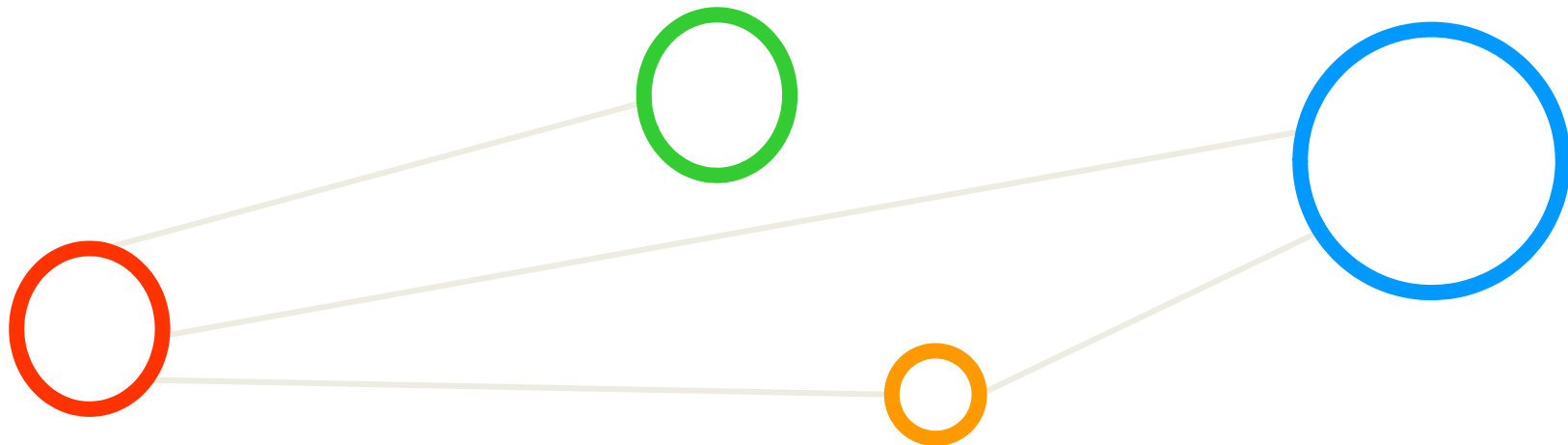
(kernel trick is substitution)

$$\textcircled{7} K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i \cdot \mathbf{x}_j \quad K(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$$

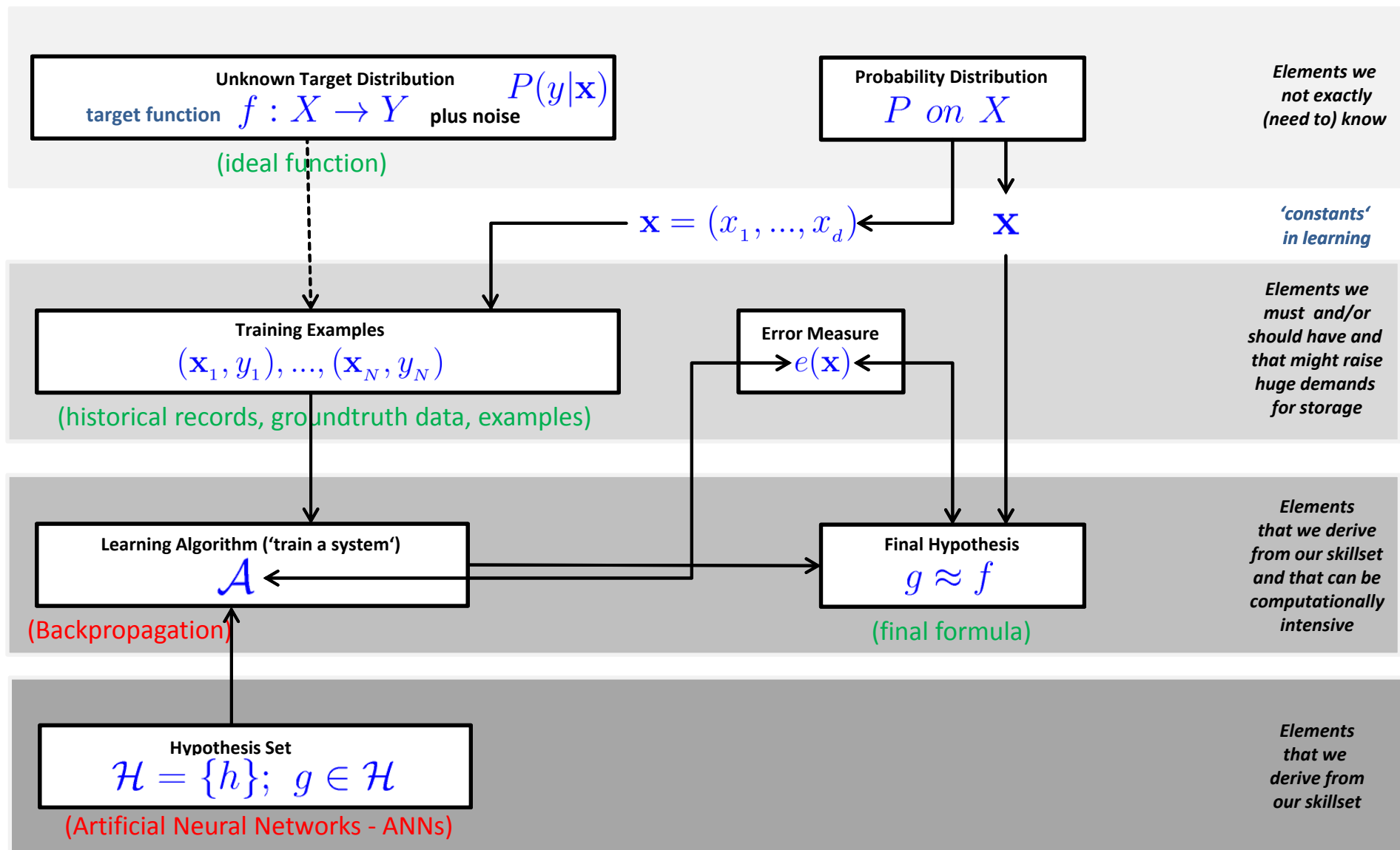
(trusted Kernel avoids to know Phi)



Appendix D: 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 3 18 18 18 126 136 175 26 166 255 247 127 0 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 49 238 253 253 253 253 253 253 253 251 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 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 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 0 139 253 190 2 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 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 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 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 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 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 0 249 253 249 64 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 39 148 229 253 253 253 250 182 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 23 66 213 253 253 253 253 198 81 2 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 55 172 226 253 253 253 253 244 133 11 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

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

[illegible][illegible][illegible][illegible]

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
```

```
# 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')
```

- **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

- 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-dimensional 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)
```

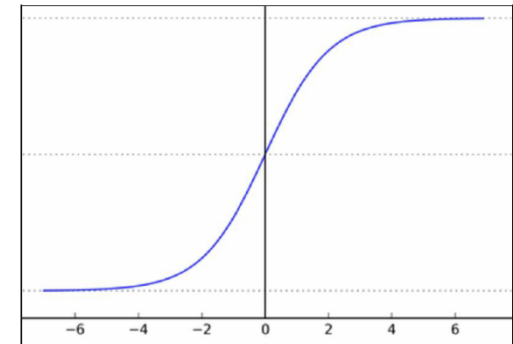
```
# Simple ANN model
model = Sequential()
model.add(Dense(NB_CLASSES, input_shape=(RESHAPED,)))
model.add(Activation('softmax'))
model.summary()
```

```
# 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])
```

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



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

- Loss function is a multiclass logarithmic loss: target is $t_{i,j}$ and prediction is $p_{i,j}$

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
```

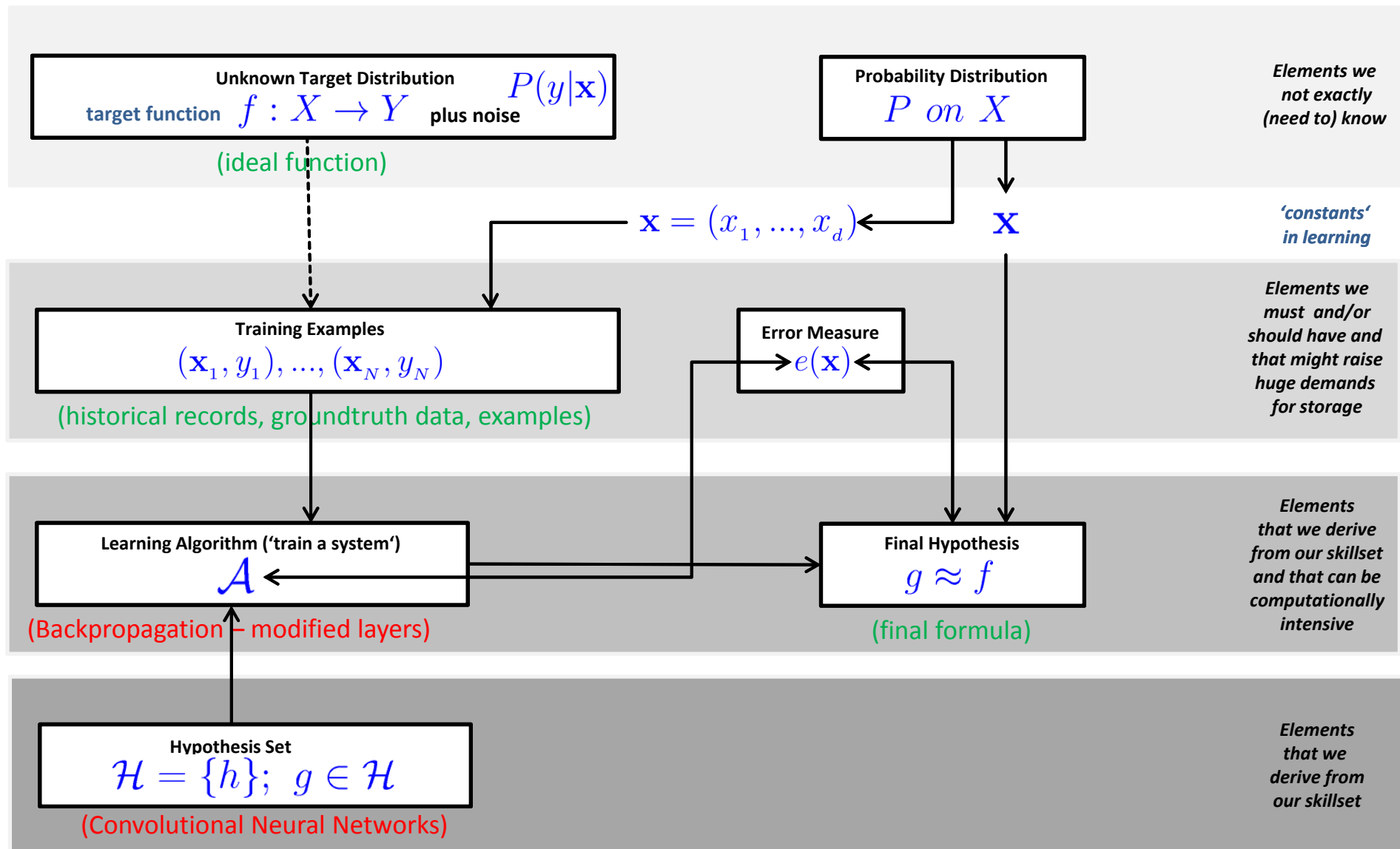
```
32/10000 [.....] - ETA: 0s
3104/10000 [=====>.....] - ETA: 0s
6208/10000 [=====>.....] - ETA: 0s
9344/10000 [=====>.....] - ETA: 0s
10000/10000 [=====] - 0s 16us/step
```

```
Test score: 0.277443544486
```

```
Test accuracy: 0.9221
```

```
Working directory was /user/scratch/gent/vsc425/vsc42544/KERAS_MNIST_ANN_1179465.master19.golett.gent.vsc
```

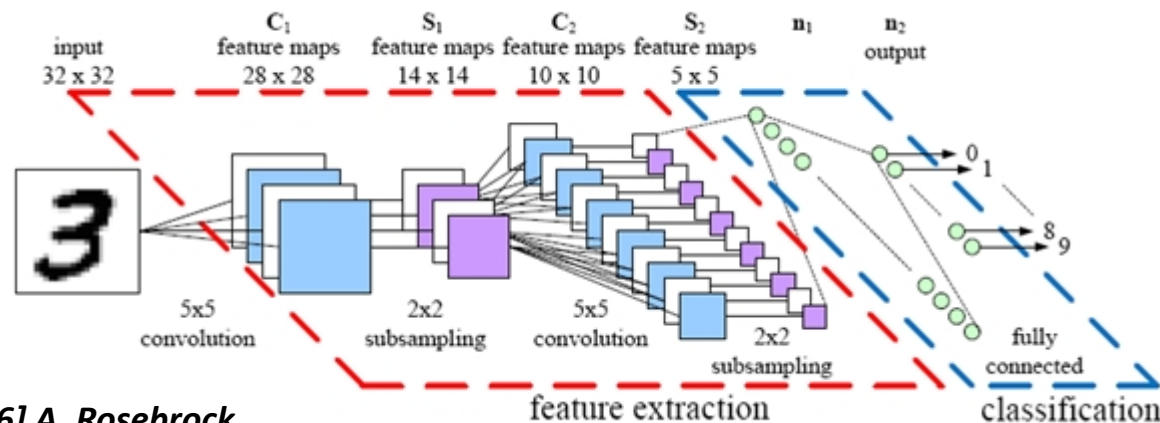
Solution Tools: Convolutional Networks Learning Model



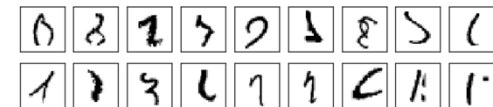
CNNs – Basic Principles

- Convolutional Neural Networks (CNNs/ConvNets) implement a connectivity pattern 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

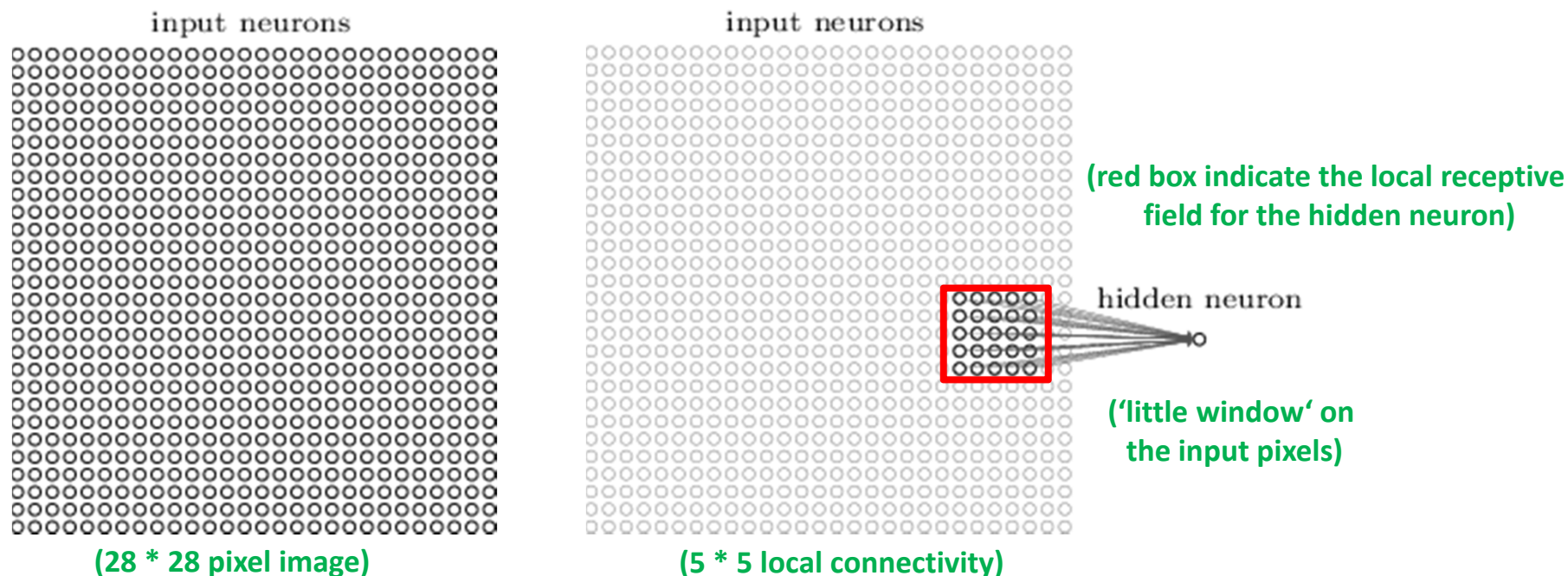


[35] M. Nielsen

CNNs – Principle Local Receptive Fields

- MNIST dataset example

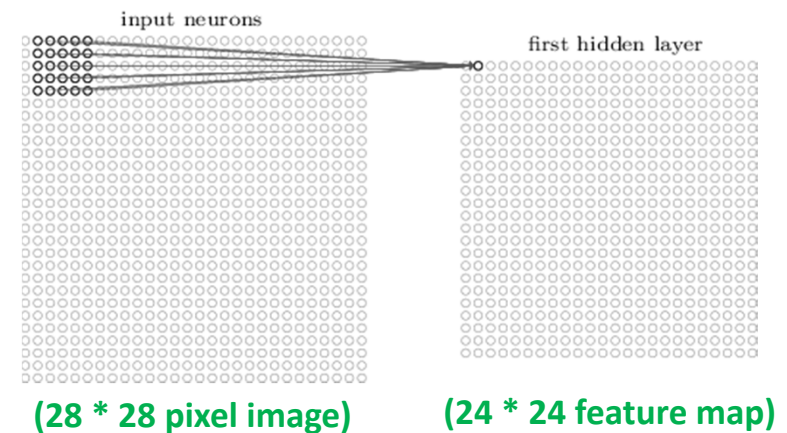
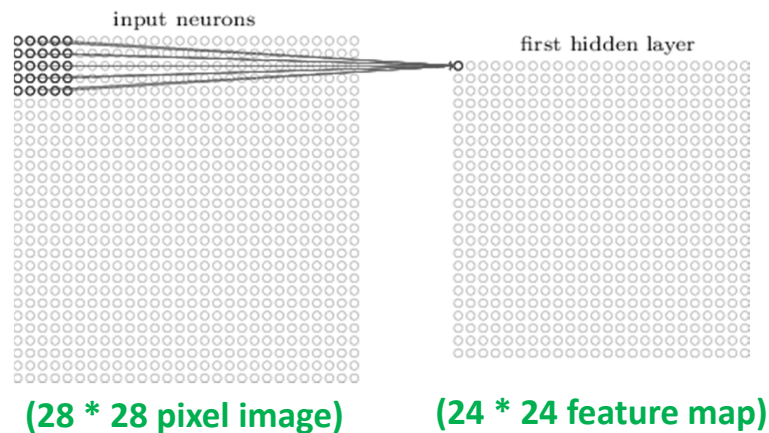
- 28 * 28 pixels modeled as square of neurons in a convolutional net
- Values correspond to the 28 * 28 pixel intensities as inputs



[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)

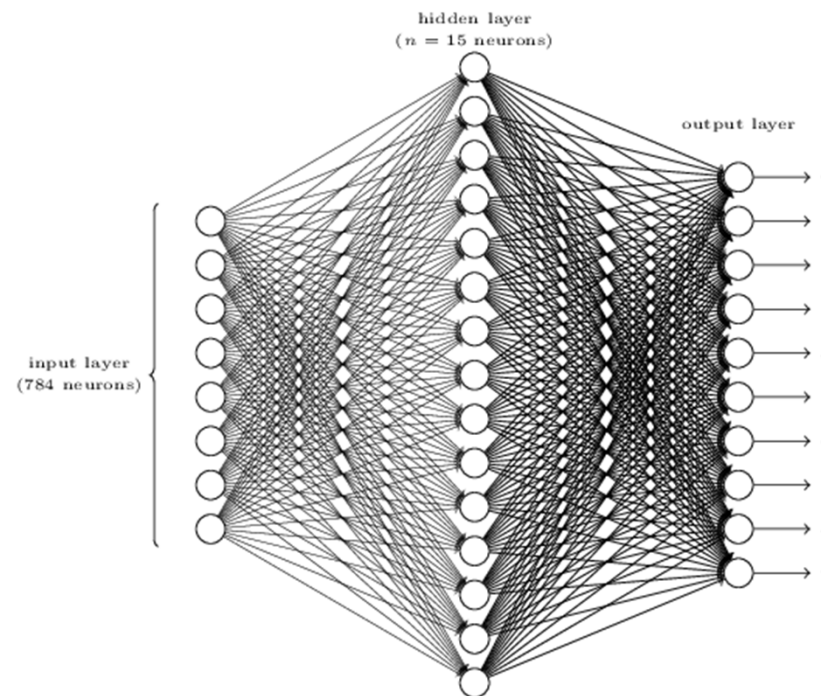


[35] M. Nielsen

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

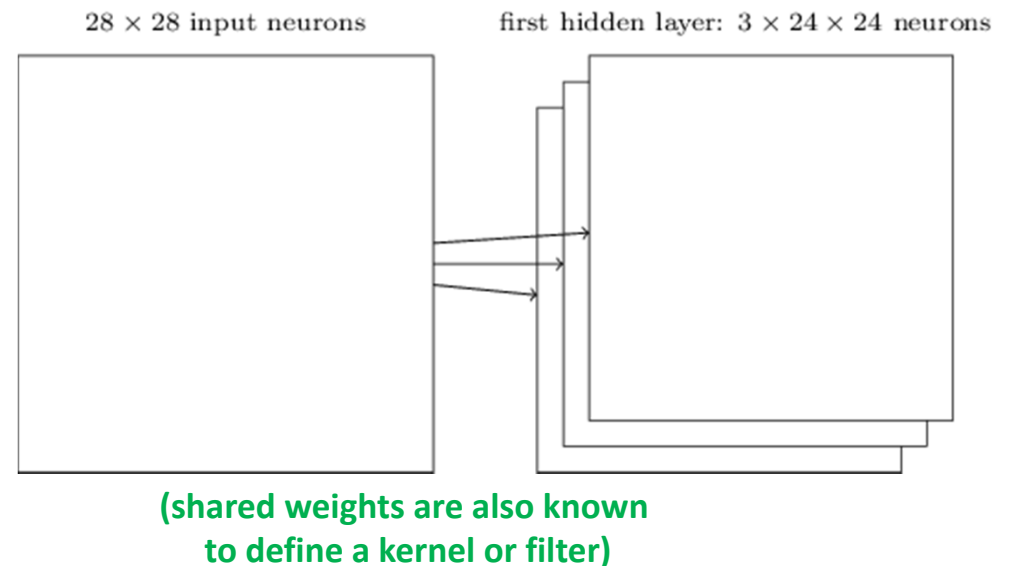
CNNs – Principle Shared Weights & Feature Maps

■ 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 \rightarrow 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)
- Benefit: learned feature being detectable across the entire image



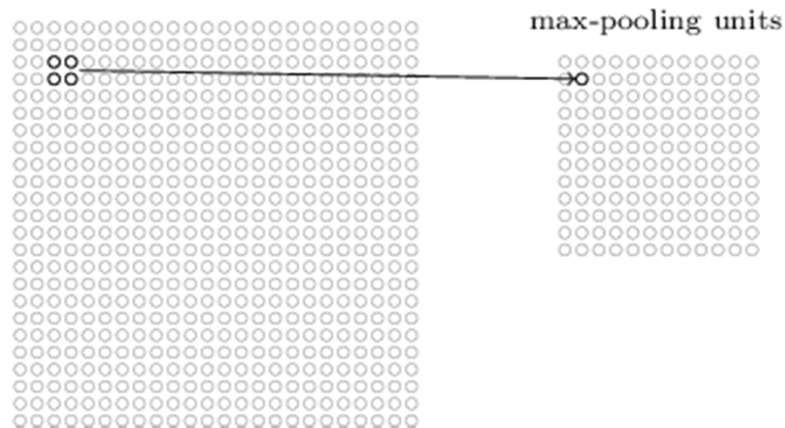
[35] M. Nielsen

CNNs – Principle of Pooling

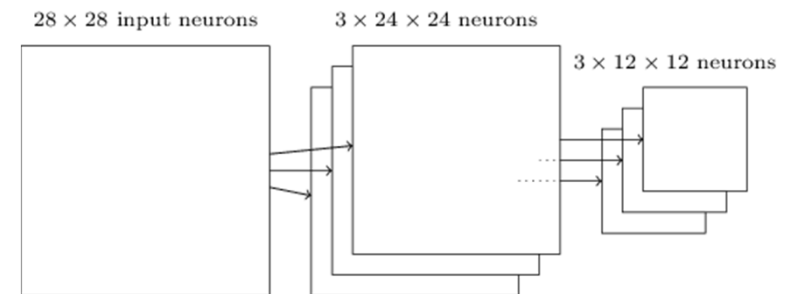
■ ‘Downsampling’ Approach

- 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

hidden neurons (output from feature map)



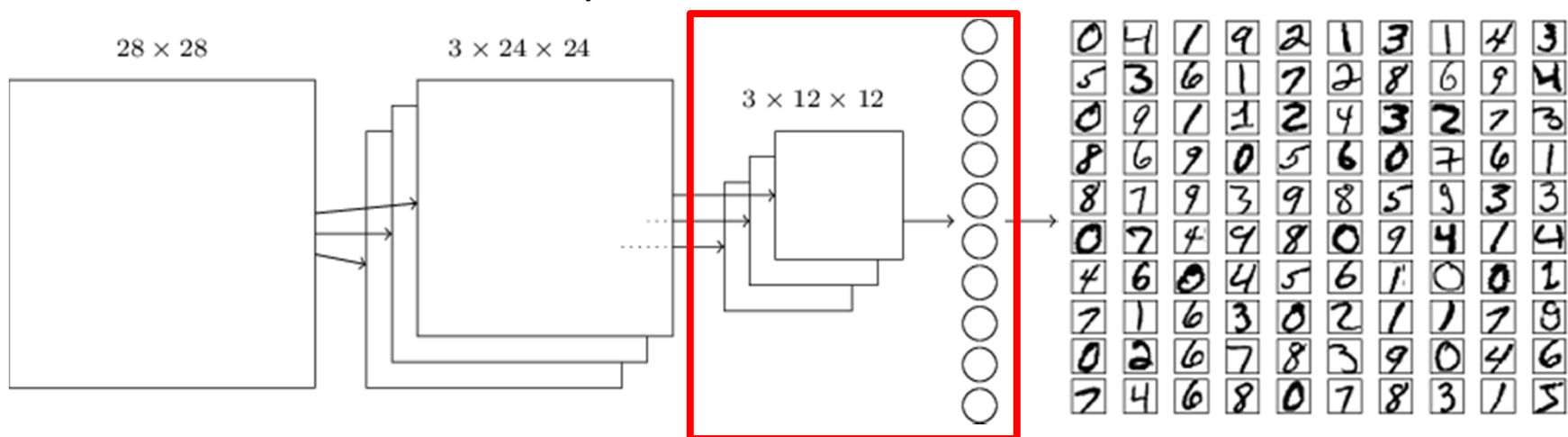
[35] M. Nielsen



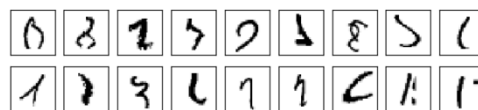
CNN – Application Example MNIST

■ 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**



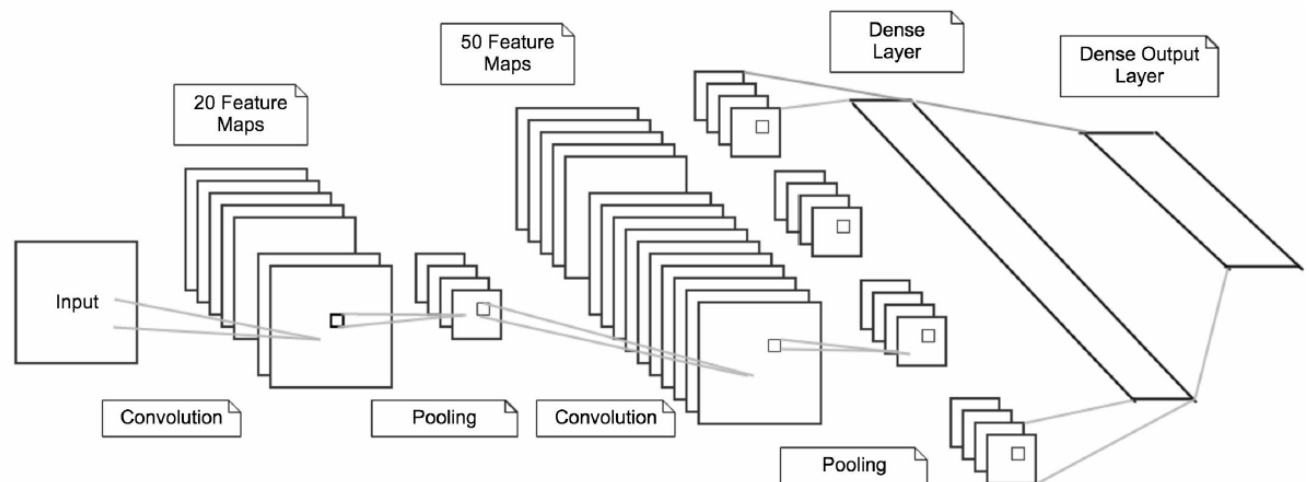
(another indicator that even with cutting edge technology machine learning never achieves 100% performance)

[35] M. Nielsen

MNIST Dataset – CNN Model

```
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
```

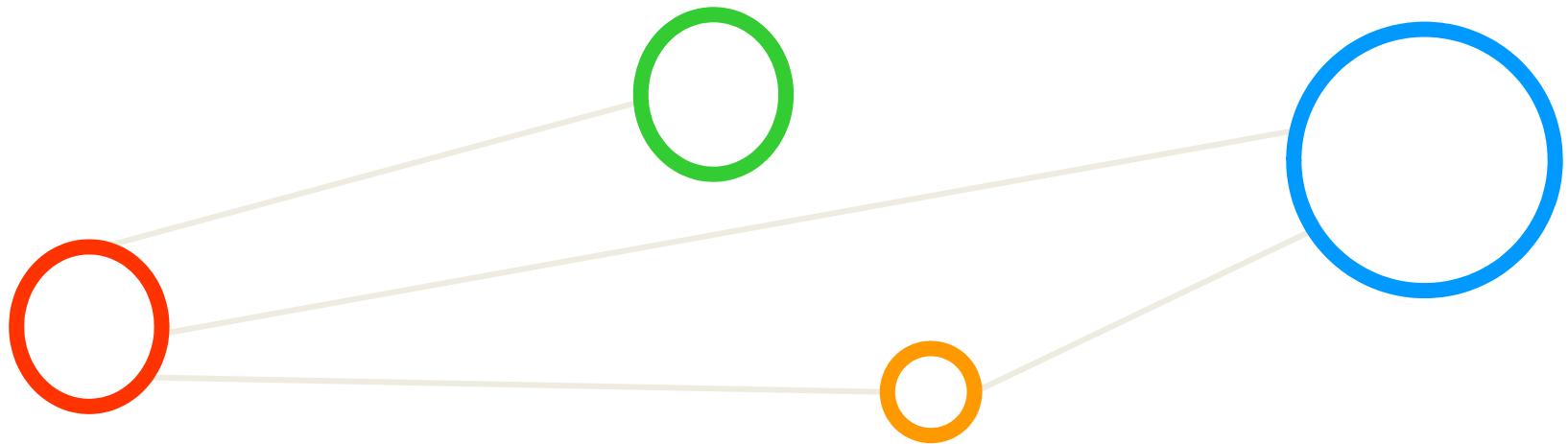
```
9824/10000 [=====>.] - ETA: 0s
9856/10000 [=====>.] - ETA: 0s
9888/10000 [=====>.] - ETA: 0s
9920/10000 [=====>.] - ETA: 0s
9952/10000 [=====>.] - ETA: 0s
9984/10000 [=====>.] - ETA: 0s
10000/10000 [=====] - 41s 4ms/step
```

```
Test score: 0.0483192791523
```

```
Test accuracy: 0.99
```

```
Working directory was /user/scratch/gent/vsc425/vsc42544/KERAS_MNIST_CNN_1179880.master19.golett.gent.vsc
```

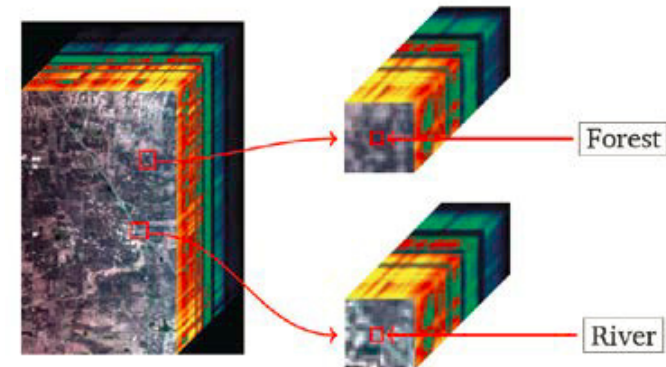
Appendix E: RNNs & LSTMs in Keras



Revisit CNNs vs. RNNs – Different Type of Neural Networks

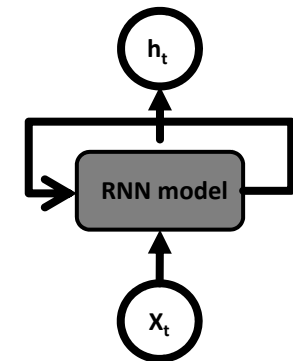
- CNNs → spatial

- Example: remote sensing application domain, **hyperspectral datasets & images**
- Neural network key property: **exploit spatial geometry of inputs**
- Approach: **Apply convolution & pooling** (height x width x feature) dimensions



- RNNs → temporal

- Examples: **texts, speech, time series datasets**
- Neural network key property: **exploit sequential nature of inputs**
- Approach: **Train a graph of 'RNN cells' & each cell performs the same operation on every element** in the given sequence



- **RNNs are used to create sequence models whereby the occurrence of an element in the sequence (e.g. text, speech, time series) is dependent on the elements that appeared before it**

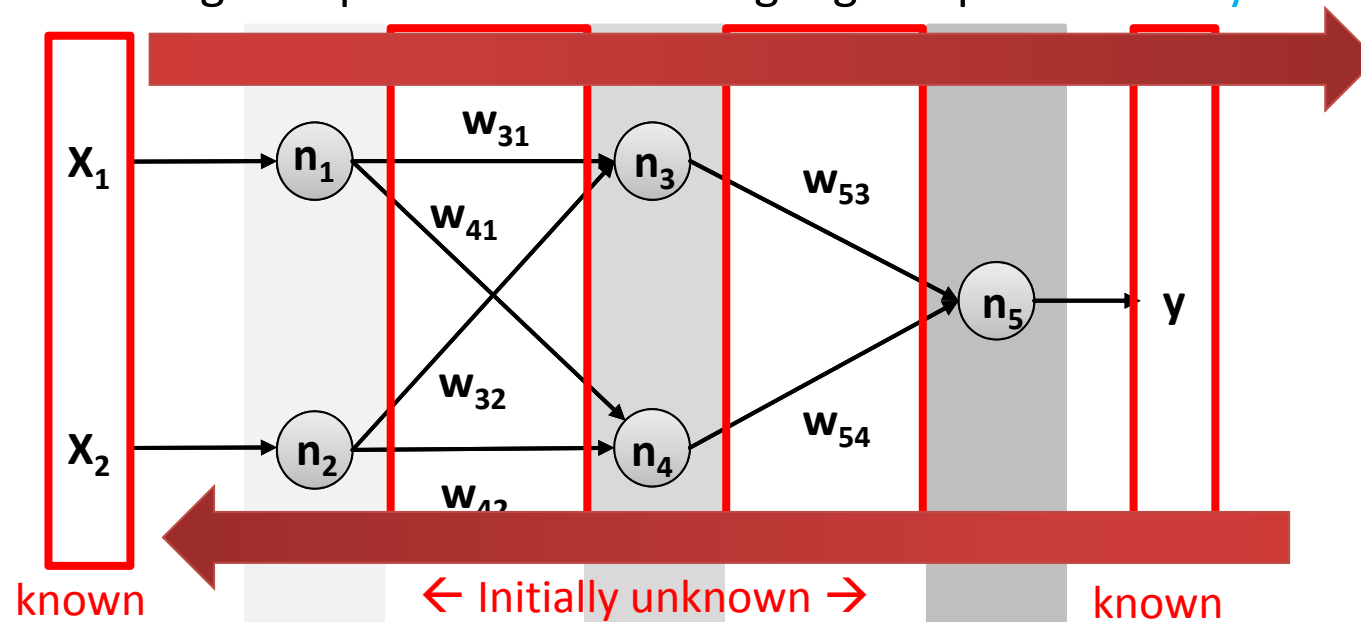
Sequence Models

- Sequence models enable various sequence predictions that are inherent different to other more traditional predictive modeling techniques or supervised learning approaches
- In contrast to mathematical sets often used, the 'sequence' model imposes an explicit order on the input/output data that needs to be preserved in training and/or inference
- Sequence models are driven by application goals and include sequence prediction, sequence classification, sequence generation, and sequence-to-sequence prediction

- Model Categorization
 - Based on different inputs/outputs to/from the sequence models
- Practical 'standard dataset' perspective
 - Often the order of samples is not important
 - Training/testing datasets and their samples have often no explicit order (i.e. 'sets')
- Practical 'sequence dataset' perspective
 - Order of samples is important
 - Sequence model learning/inference needs this order

Limitations of Feed Forward ANN

- Selected application examples revisited
 - Predicting next word in a sentence requires 'history' of previous words
 - Translating european in chinese language requires 'history' of context



- Traditional feed forward artificial neural networks show limits when a certain 'history' is required
- Each Backpropagation forward/backward pass starts a new pass independently from pass before
- The 'history' in the data is often a specific type of 'sequence' that required another approach

Recurrent Neural Network (RNN)

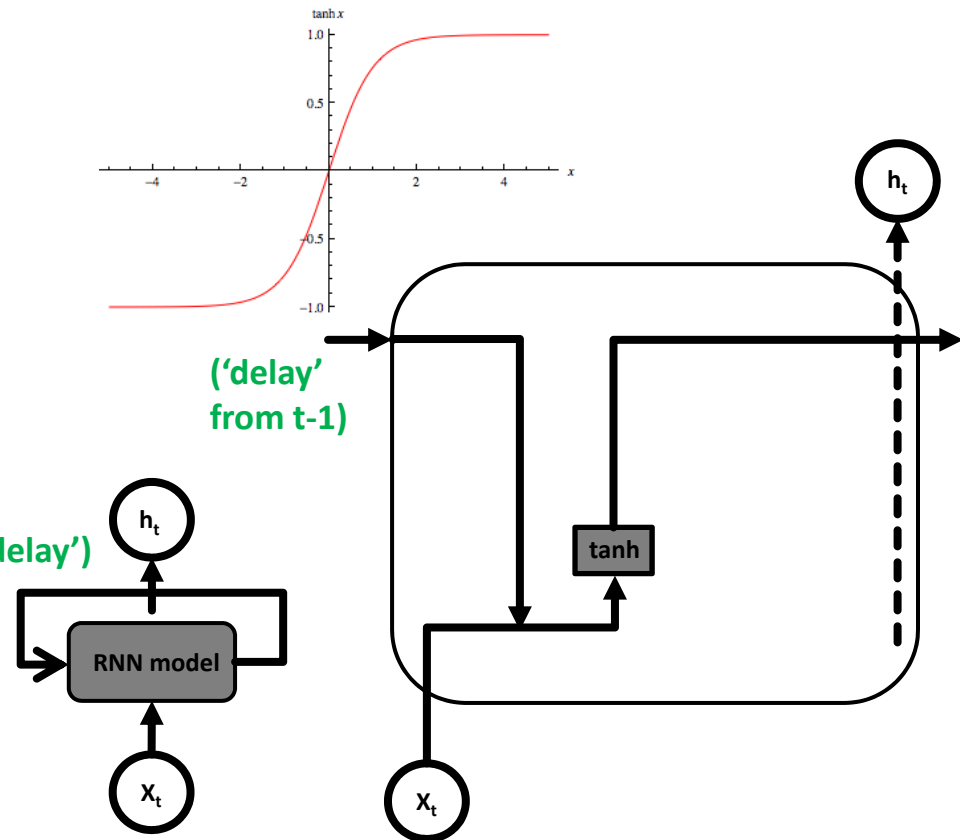
- A Recurrent Neural Network (RNN) consists of cyclic connections that enable the neural network to better model sequence data compared to a traditional feed forward artificial neural network (ANN)
- RNNs consists of 'loops' (i.e. cyclic connections) that allow for information to persist while training
- The repeating RNN model structure is very simple whereby each has only a single layer (e.g. tanh)

- Selected applications

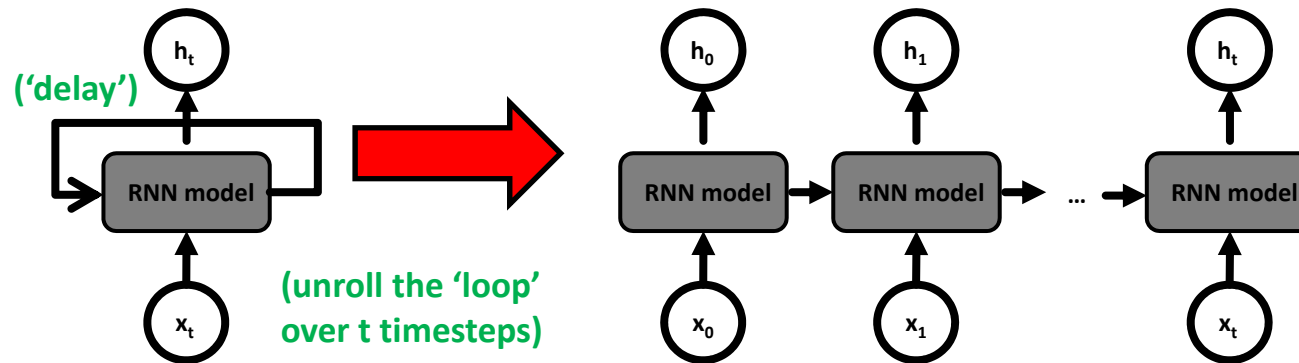
- Sequence labeling
- Sequence prediction tasks
- E.g. handwriting recognition
- E.g. language modeling

- Loops / cyclic connections

- Enable to pass information ('delay') from one step to the next iteration
- Remember 'short-term' data dependencies

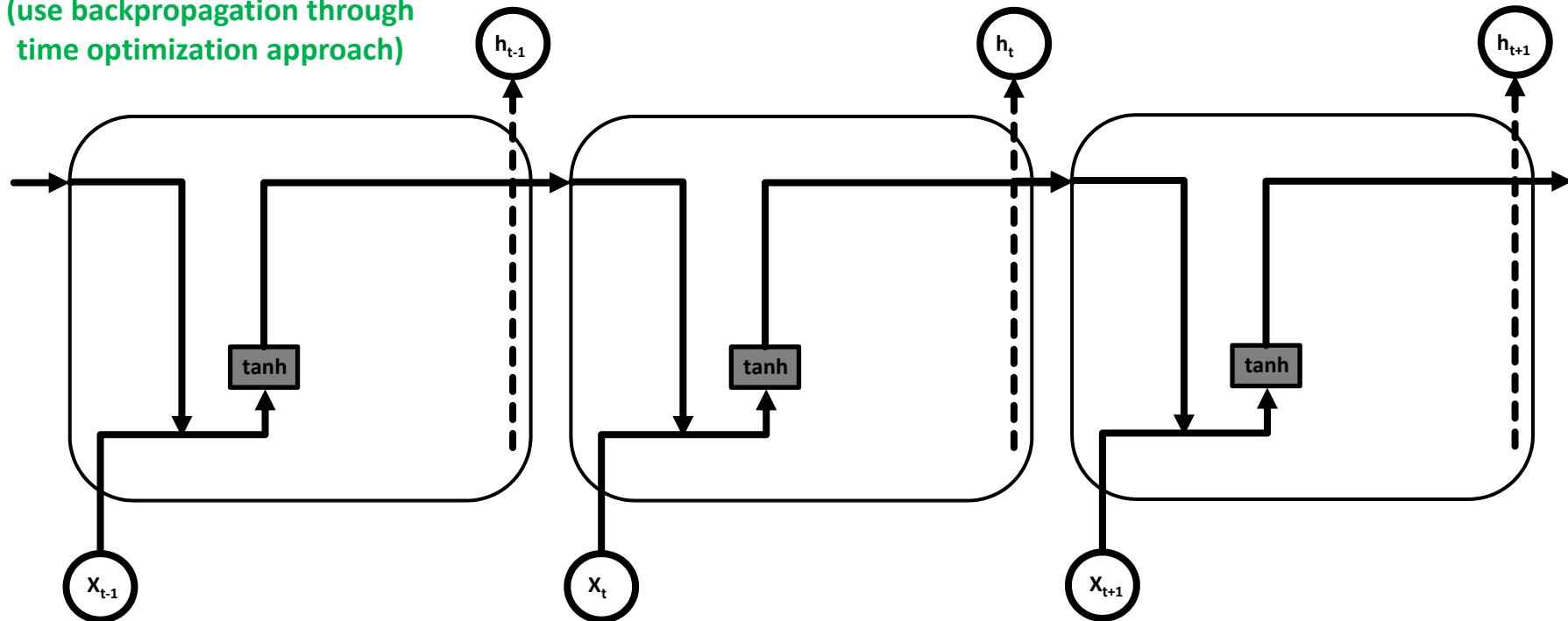


Unrolled RNN

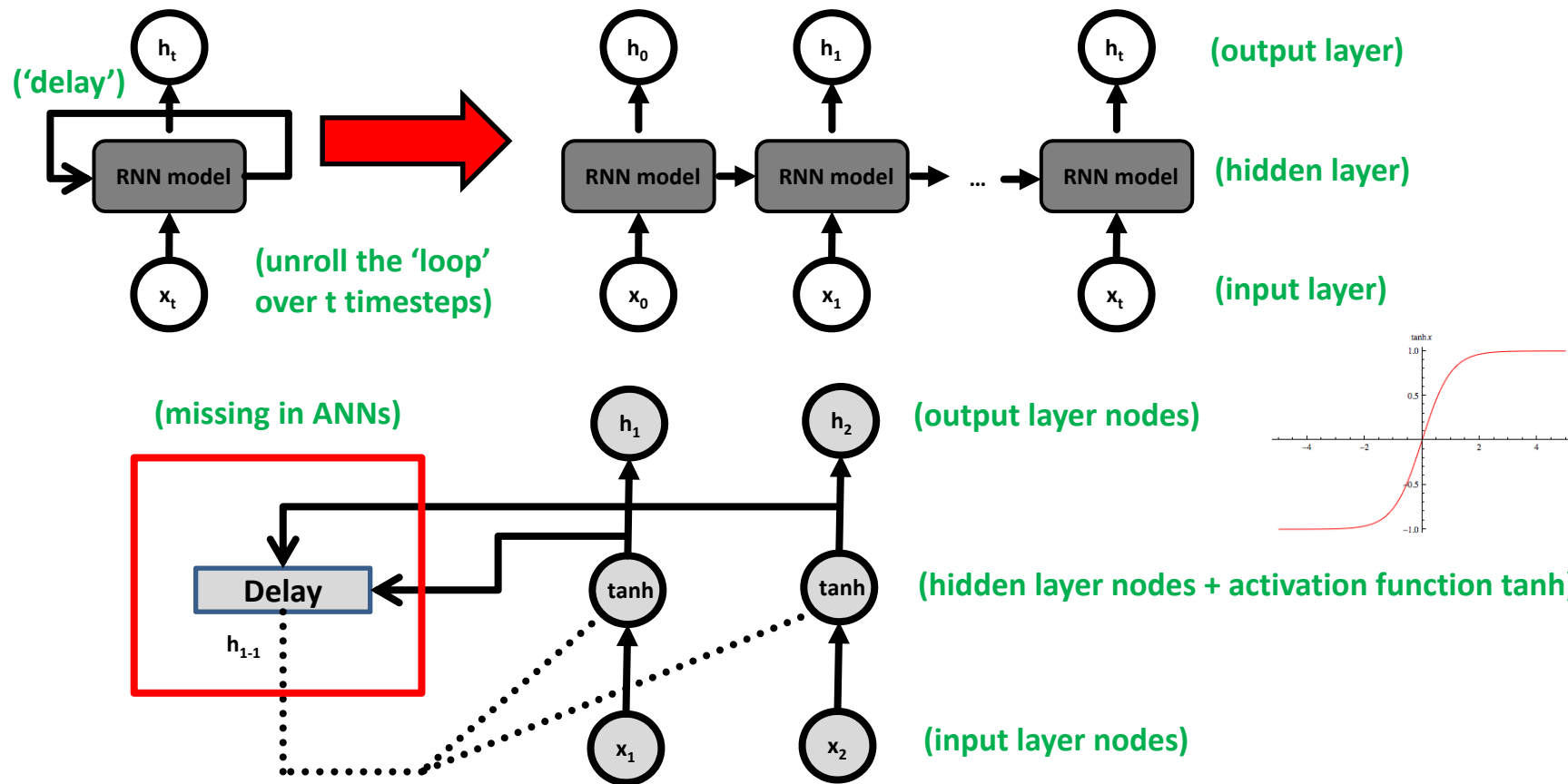


- A RNN can be viewed as multiple copies of the same network, each passing a message to a successor – this gets clear when 'unrolling the RNN loop'

(use backpropagation through time optimization approach)

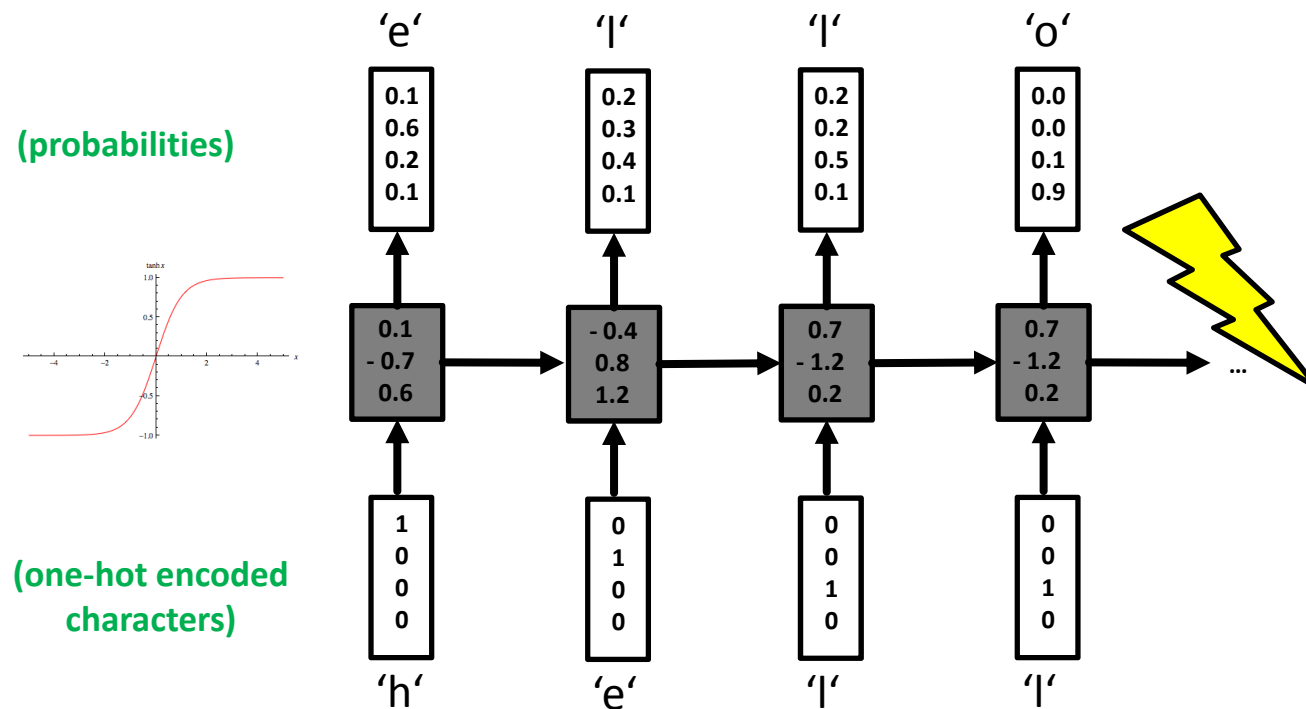
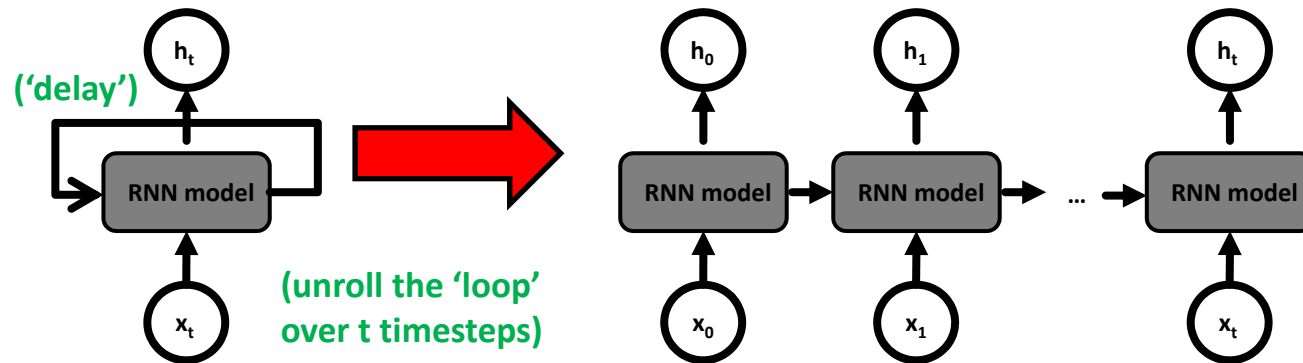


Unrolled RNN – Role of ‘Delay’ and Nodes in Layers



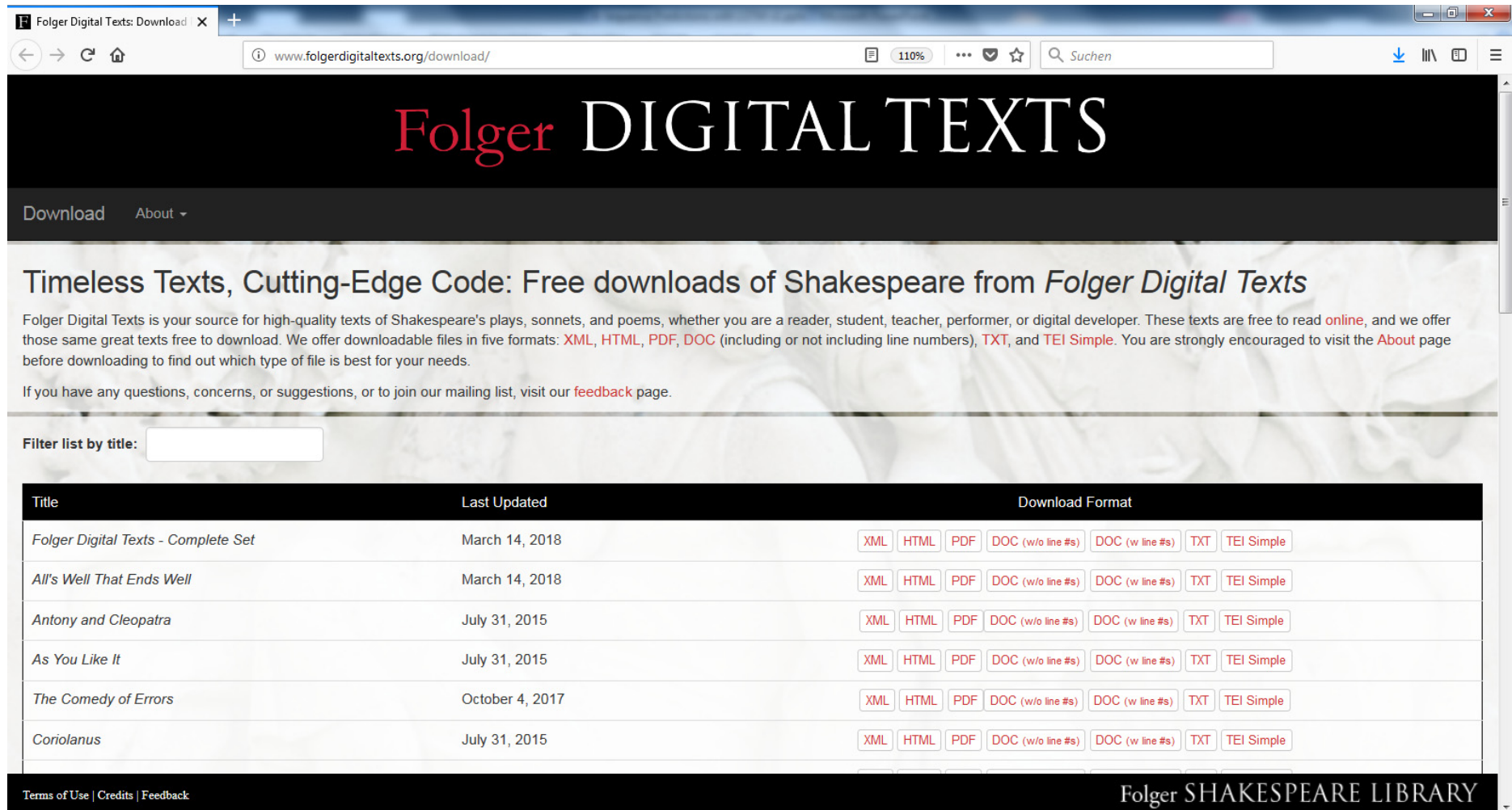
- RNNs are unrolled programmatically during the training and prediction phase
- Idea of ‘delay’ means feeding back the output of a neural network layer at a specific time t to the input of the same neural network layer at time $t+1$ → establishes something like ‘short memory’

RNN Model – Simple Example – Predict Next Character



- Sequence values that are separated by a significant number of words (i.e. deep RNN) leads to the vanishing gradient problem
- Reasoning is that small gradients or weights with values than 1 are multiplied many times through the multiple time steps, i.e. gradients shrink asymptotically to zero
- Effect is that weights of those earlier layers are not changed significantly and the network will not learn long-term dependencies

RNN Example – Data Repository



The screenshot shows the 'Folger Digital Texts' website. The header features the site's name in a large, serif font. Below the header, there is a navigation bar with 'Download' and 'About' links. The main content area has a heading 'Timeless Texts, Cutting-Edge Code: Free downloads of Shakespeare from *Folger Digital Texts*' followed by a paragraph describing the site's mission and the formats available for download: XML, HTML, PDF, DOC, TXT, and TEI Simple. A search bar is present, and a table lists several plays with their last update dates and available download formats. The footer includes links for 'Terms of Use', 'Credits', and 'Feedback', along with the 'Folger SHAKESPEARE LIBRARY' logo.

Folger DIGITAL TEXTS

Download About ▾

Timeless Texts, Cutting-Edge Code: Free downloads of Shakespeare from *Folger Digital Texts*

Folger Digital Texts is your source for high-quality texts of Shakespeare's plays, sonnets, and poems, whether you are a reader, student, teacher, performer, or digital developer. These texts are free to read [online](#), and we offer those same great texts free to download. We offer downloadable files in five formats: [XML](#), [HTML](#), [PDF](#), [DOC](#) (including or not including line numbers), [TXT](#), and [TEI Simple](#). You are strongly encouraged to visit the [About](#) page before downloading to find out which type of file is best for your needs.

If you have any questions, concerns, or suggestions, or to join our mailing list, visit our [feedback](#) page.

Filter list by title:

Title	Last Updated	Download Format
<i>Folger Digital Texts - Complete Set</i>	March 14, 2018	XML HTML PDF DOC (w/o line #s) DOC (w line #s) TXT TEI Simple
<i>All's Well That Ends Well</i>	March 14, 2018	XML HTML PDF DOC (w/o line #s) DOC (w line #s) TXT TEI Simple
<i>Antony and Cleopatra</i>	July 31, 2015	XML HTML PDF DOC (w/o line #s) DOC (w line #s) TXT TEI Simple
<i>As You Like It</i>	July 31, 2015	XML HTML PDF DOC (w/o line #s) DOC (w line #s) TXT TEI Simple
<i>The Comedy of Errors</i>	October 4, 2017	XML HTML PDF DOC (w/o line #s) DOC (w line #s) TXT TEI Simple
<i>Coriolanus</i>	July 31, 2015	XML HTML PDF DOC (w/o line #s) DOC (w line #s) TXT TEI Simple

[Terms of Use](#) | [Credits](#) | [Feedback](#)

Folger SHAKESPEARE LIBRARY

[42] Folger Digital Texts

RNN Example – Language Model Setup

- Typical approach
 - Create ‘generative model’ to predict the next word given previous words
 - Enables to generate text by sampling from the output probabilities
 - Build a ‘word-based language model’ → can be computational complex
- Simplified model for tutorial
 - Reasoning: simpler model and quicker training
 - Train a ‘character based language model’ on one text of Shakespeare
 - Take advantage of standard RNN cells
 - Predict (only) the next character given 10 previous characters
 - Use the trained language model to generate some text in the same style

(10 characters → prediction)

```
it turned -> i
t turned i -> n
turned in -> t
turned int -> o
urned into ->
rned into -> a
ned into a ->
ed into a -> p
d into a p -> i
into a pi -> g
```

[37] *Deep Learning with Keras*

RNN Example – Keras Python Script – Preprocessing

```
from __future__ import print_function
from keras.layers import Dense, Activation
from keras.layers.recurrent import SimpleRNN
from keras.models import Sequential
import numpy as np
```

- Import necessary modules, e.g. SimpleRNN for a simple RNN cell, or Dense for a fully connected layer

```
# preprocessing of input text data
fin = open("/homea/hpclab/train001/data/macbeth/Mac.txt", 'rb')
lines = []
for line in fin:
    line = line.strip().lower()
    line = line.decode("ascii", "ignore")
    if len(line) == 0:
        continue
    lines.append(line)
fin.close()
text = " ".join(lines)
```

- Preprocessing of original files that e.g. contain line breaks, non-ASCII characters, capital characters; Result is variable text with 'cleaned text'

```
# lookup tables char vs index of chars
chars = set([c for c in text])
nb_chars = len(chars)
char2index = dict((c, i) for i, c in enumerate(chars))
index2char = dict((i, c) for i, c in enumerate(chars))
```

- Create lookup tables for characters per index & vice versa

- Character-level RNN: vocabulary is the set of characters that occur in the text → use index of character instead of a character itself

[37] *Deep Learning with Keras*

RNN Example – Keras Python Script – Input & Label Texts

```
# go through dataset, create input and label text
SEQLEN = 10
STEP = 1
input_chars = []
label_chars = []
for i in range(0, len(text) - SEQLEN, STEP):
    input_chars.append(text[i:i + SEQLEN])
    label_chars.append(text[i + SEQLEN])
```

```
# input and label text as vectors
X = np.zeros((len(input_chars), SEQLEN, nb_chars), dtype=np.bool)
y = np.zeros((len(input_chars), nb_chars), dtype=np.bool)
for i, input_char in enumerate(input_chars):
    for j, ch in enumerate(input_char):
        X[i, j, char2index[ch]] = 1
    y[i, char2index[label_chars[i]]] = 1
```

- Each row of input to the RNN corresponds to one of the input texts
- SEQLEN characters input; vocabulary size = nb_chars (set of different characters in text) → one-hot encoded vector of size (nb_chars)

- Task: Predict (only) the next character given 10 previous characters → SEQLEN = 10, STEP=1

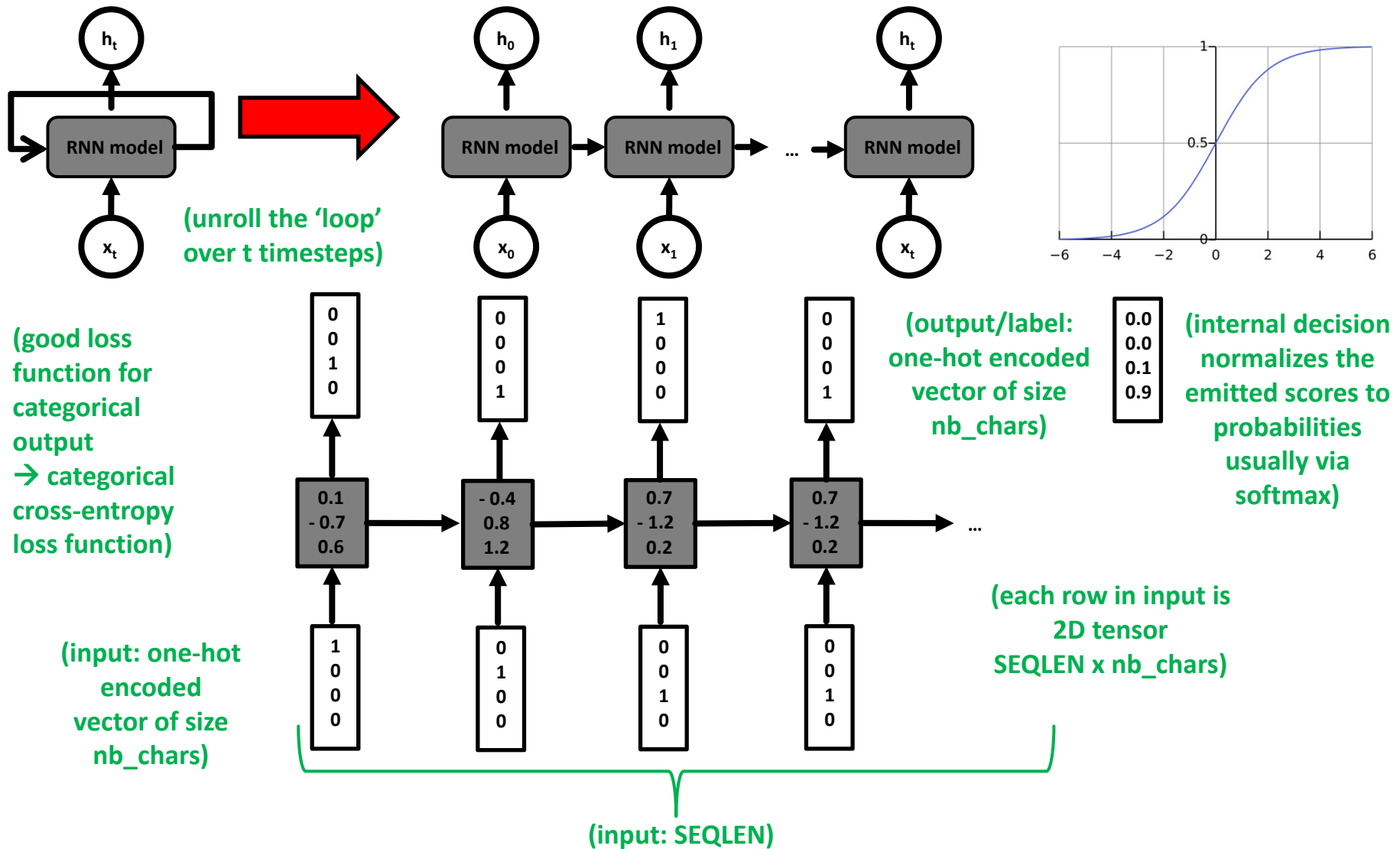
- Moving step-wise through text by STEP=1 number of characters & extract span of text with size SEQLEN=10

(input_chars
→
label_chars)

it turned -> i
t turned i -> n
turned in -> t
turned int -> o
urned into -> a
rned into -> a
ned into a -> p
ed into a -> p
d into a p -> i
into a pi -> g

[37] Deep Learning with Keras

RNN Example – Modelling & Decisions



RNN Example – Keras Python Script – Model & Parameter

```
# hyperparameter
HIDDEN_SIZE = 128
BATCH_SIZE = 128
NUM_ITERATIONS = 25
NUM_EPOCHS_PER_ITERATION = 1
NUM_PREDS_PER_EPOCH = 100
```

```
# RNN model
model = Sequential()
model.add(SimpleRNN(HIDDEN_SIZE, return_sequences=False,
                    input_shape=(SEQLEN, nb_chars),
                    unroll=True))
model.add(Dense(nb_chars))
model.add(Activation("softmax"))
model.compile(loss="categorical_crossentropy", optimizer="rmsprop")
```

- Hyperparameter HIDDEN_SIZE=128 means output dimension of size 128 for ok text; parameter by experimentation

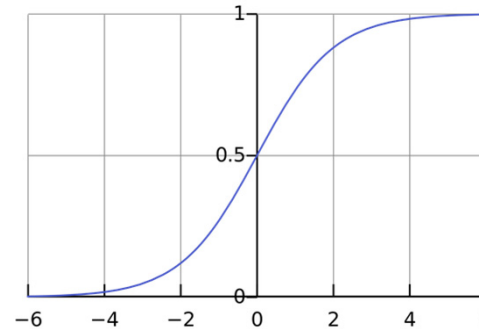
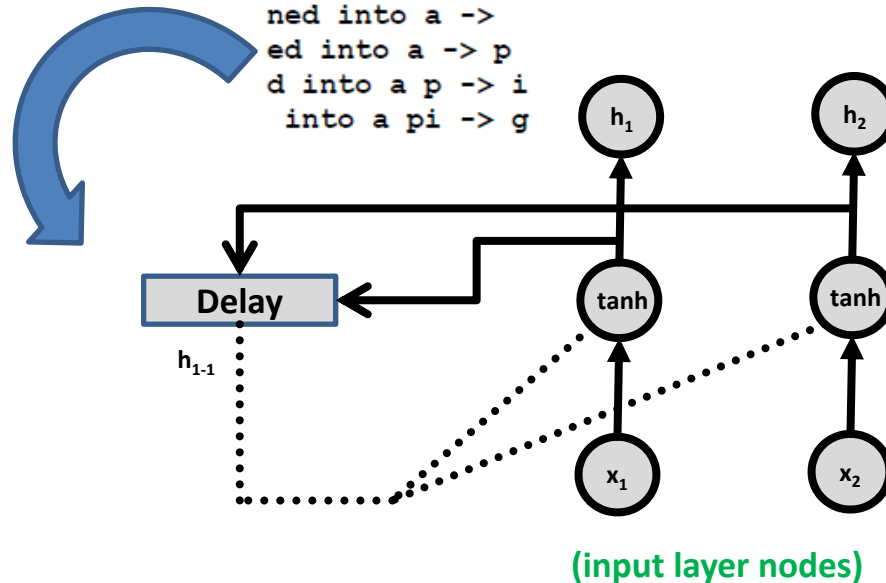
- Sequential model adding first a SimpleRNN layer of size 128, return_sequences = False means single character as output/label not a 'sequence of characters', input tensor is SEQLEN x nb_chars; unroll = True - performance

- Adding a Dense layer of size nb_chars & activation function 'softmax' (emits scores for each of the characters in vocabulary → probabilities)
- Use optimizer 'rmsprop' with 'categorical_crossentropy' loss function

RNN Example – Keras Model & Activation Functions

(iterations)

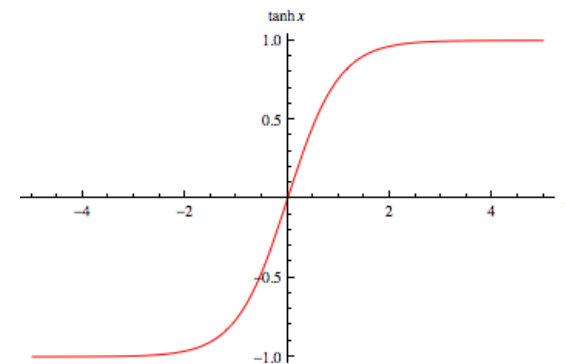
```
it turned -> i
t turned i -> n
turned in -> t
turned int -> o
urned into ->
rned into -> a
ned into a ->
ed into a -> p
d into a p -> i
into a pi -> g
```



(internal decision
normalizes the
emitted scores to
probabilities
usually via
softmax)

(Dense layer with 'number of characters' as nodes +
'softmax' activation function as output layer nodes)

(SimpleRNN layer with 128 hidden nodes with
default hyperbolic tangent as activation function,
i.e. values squashed between 1 and -1)



RNN Example – Keras Python Script – Training Process

```
# training process
for iteration in range(NUM_ITERATIONS):
    print("=" * 50)
    print("Iteration #: %d" % (iteration))
    model.fit(X, y, batch_size=BATCH_SIZE, epochs=NUM_EPOCHS_PER_ITERATION)
    test_idx = np.random.randint(len(input_chars))
    test_chars = input_chars[test_idx]
    print("Generating from seed: %s" % (test_chars))
    print(test_chars, end="")
    for i in range(NUM_PREDICTIONS_PER_EPOCH):
        Xtest = np.zeros((1, SEQUENCE_LENGTH, nb_chars))
        for i, ch in enumerate(test_chars):
            Xtest[0, i, char2index[ch]] = 1
        pred = model.predict(Xtest, verbose=0)[0]
        ypred = index2char[np.argmax(pred)]
        print(ypred, end="")
        # move forward with test_chars + ypred
        test_chars = test_chars[1:] + ypred
    print()
```

- Cf. supervised learning process (day one)
 - Labels existing (not in this unsupervised example)
 - Train model for fixed number of epochs
 - Evaluate model against test dataset

[37] Deep Learning with Keras

- Train model for epochs = 1 since no labelled dataset and then testing; training for 25 iterations → NUM_ITERATIONS; aka training for 25 epochs/iterations

- Test: generate a character from model given a random input; dropping the first character from the input & append the predicted character from our previous run & generate another character (100 x)

RNN Example – Submit Script

- Job submit script
 - Specify good name for the job
 - Allocate GPUs for deep learning job
 - Specify job queue
 - Restore module environment with all dependencies
 - Use python with rnn-example.py script
- Use sbatch
 - Use jobscript

```
#!/bin/bash -x
#SBATCH--nodes=1
#SBATCH--ntasks=1
#SBATCH--output=rnn_out.%j
#SBATCH--error=rnn_err.%j
#SBATCH--time=01:00:00
#SBATCH--mail-user=m.riedel@fz-juelich.de
#SBATCH--mail-type=ALL
#SBATCH--job-name=simple-RNN

#SBATCH--partition=gpus
#SBATCH--gres=gpu:1

#SBATCH--reservation=deep_learning

### location executable
KERASSCRIPT=/homea/hpclab/train001/tools/rnn/rnn-example.py

module restore dl_tutorial

### submit
python $KERASSCRIPT
```

RNN Example – Output Interpretation

- Challenge: unsupervised learning problem
 - Check output with 'more out.txt'
 - Idea: string gives us an indication of the quality of the model
 - More epochs/iterations → better quality of the model

[illegible]

(learned well to spell compared to first iteration but no coherent thoughts → still interesting since no word concept)

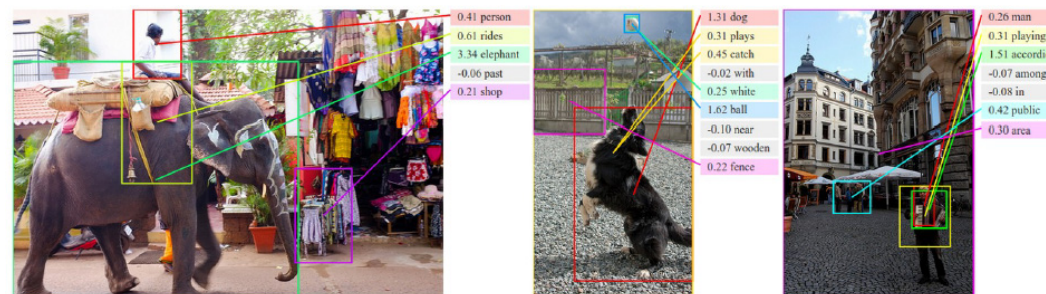
```
99968/101872 [=====>.] - ETA: 0s - loss: 1.5870
101632/101872 [=====>.] - ETA: 0s - loss: 1.5873
101872/101872 [=====] - 3s 30us/step - loss: 1.5871
Generating from seed: eeks when
eeks when the did the pronor me the cantant in the with the dines and the servant he childred macbeth
=====
Iteration #: 23
Epoch 1/1
```

Different Useful LSTM Models

- Standard LSTM
 - Memory cells with single LSTM layer; used in simple network structures
- Stacked LSTM
 - LSTM layers are stacked one on top of another; creating deep networks
- CNN LSTM
 - CNNs to learn features (e.g. images); LSTM for image sequences
- Encoder-Decoder LSTM
 - One LSTM network → encode input; one LSTM network → decode output
- Bidirectional LSTM
 - Input sequences are presented and learned both forward & backwards
- Generative LSTM
 - LSTMs learn the inherent structure relationship in input sequences; then generate new plausible sequences

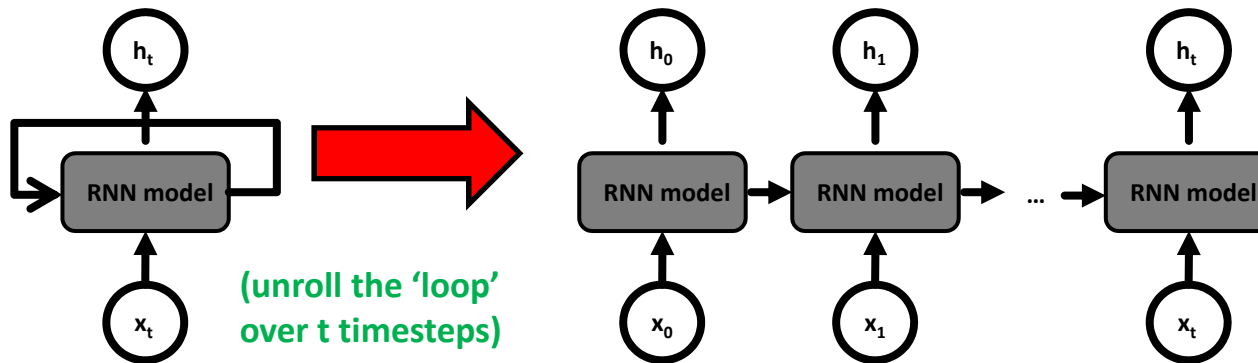
Long Short-Term Memory (LSTM) Models

- Specific type of Recurrent Neural Network (RNN)
 - Different to techniques like standard Artificial Neural Networks (ANNs) or Convolutional Neural Networks (CNNs)
 - Solving certain limits of ANNs through RNNs design
 - RNNs offer short-term memory – LSTMs add ‘long-term’ capabilities
 - Idea: improved performance through ‘more memory’ (cp. HPC?!)
- Designed specifically for sequence prediction problems
 - World-class results in complex problem domains & applications
 - E.g. language translation, automatic image captioning, text generation



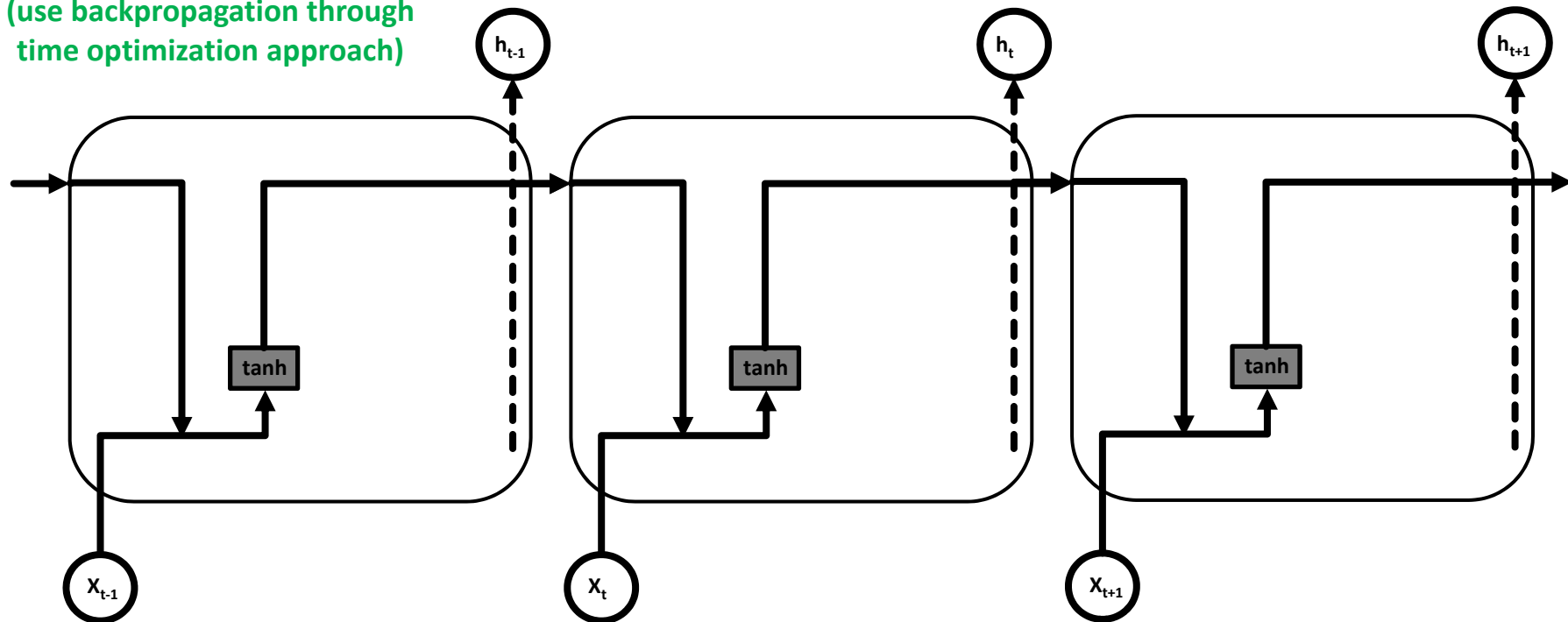
[43] A. Karpathy & F. Li, ‘Deep Visual-Semantic Alignments for Generating Image Descriptions’

Unrolled RNN – Revisited

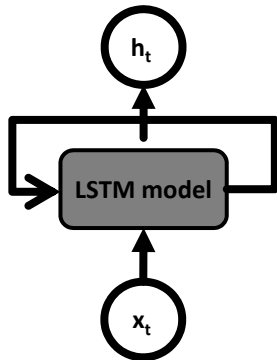


- A RNN can be viewed as multiple copies of the same network, each passing a message to a successor – this gets clear when 'unrolling the RNN loop'

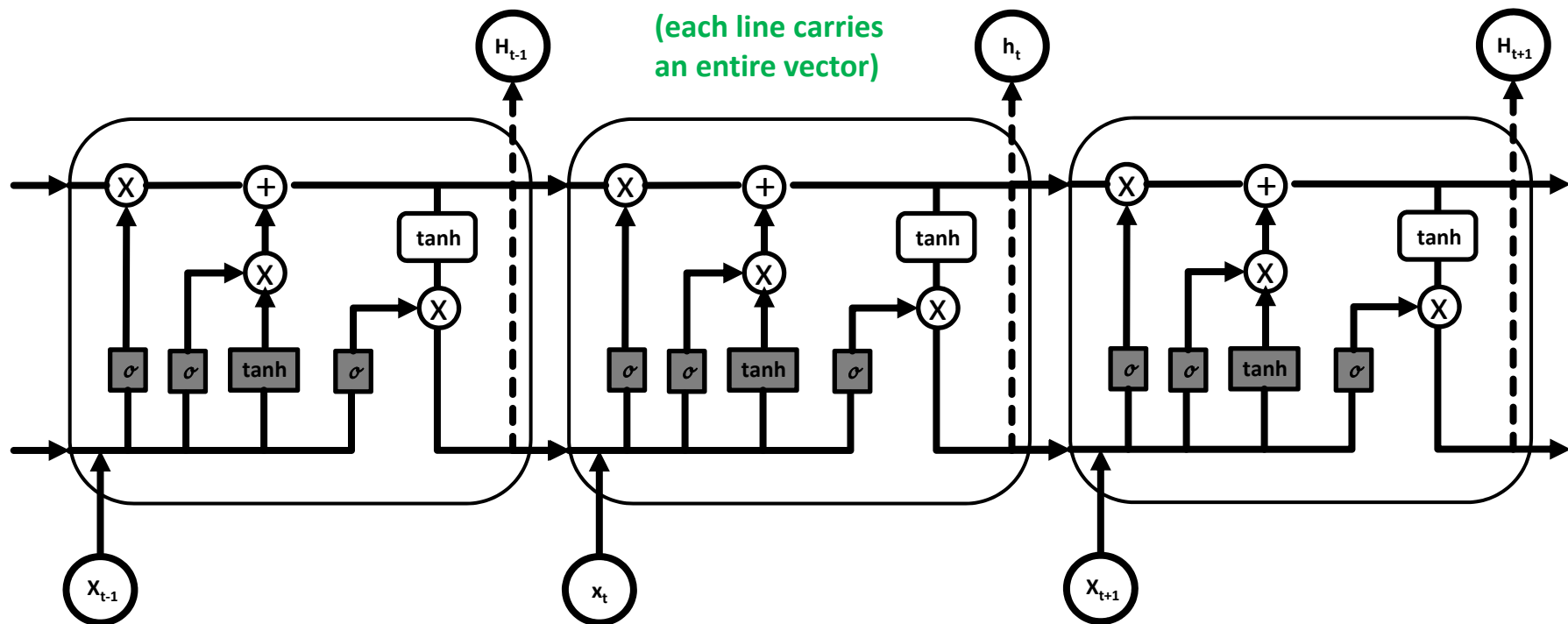
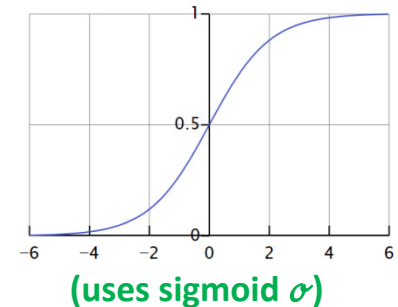
(use backpropagation through time optimization approach)



Long Short Term Memory (LSTM) Model

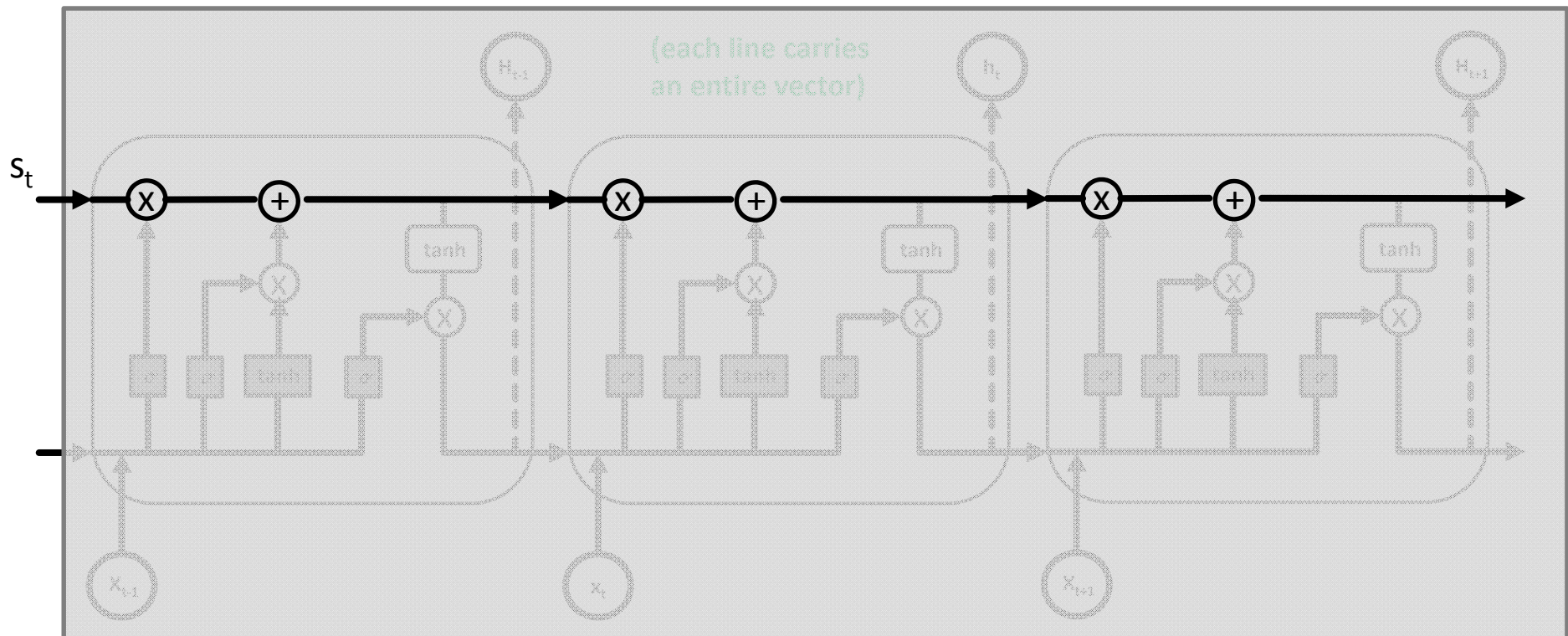


- Long Short Term Memory (LSTM) networks are a special kind of Recurrent Neural Networks (RNNs)
- LSTMs learn long-term dependencies in data by remembering information for long periods of time
- The LSTM chain structure consists of four neural network layers interacting in a specific way



LSTM Model – Memory Cell & Cell State

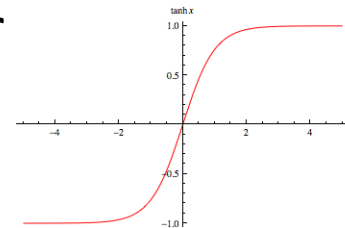
- LSTM introduce a 'memory cell' structure into the underlying basic RNN architecture using four key elements: an input gate, a neuron with self-current connection, a forget gate, and an output gate
- The data in the LSTM memory cell flows straight down the chain with some linear interactions (x,+)
- The cell state s_t can be different at each of the LSTM model steps & modified with gate structures
- Linear interactions of the cell state are pointwise multiplication (x) and pointwise addition (+)
- In order to protect and control the cell state s_t three different types of gates exist in the structure



Computing of LSTM Cell – Step 1-2

1. New x_t input together with the output from cell h_{ht-1} are squashed via a tanh layer

- Outputs between -1 and 1

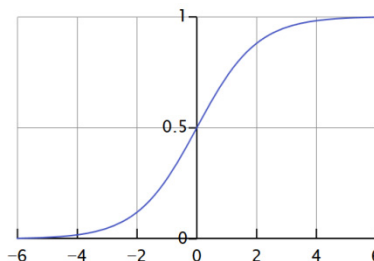


[44] *Adventures in Machine Learning*

2. New x_t input together with the output from cell h_{ht-1} is passed through the 'input gate'

- Layer of sigmoid activated nodes whose output is multiplied by squashed input

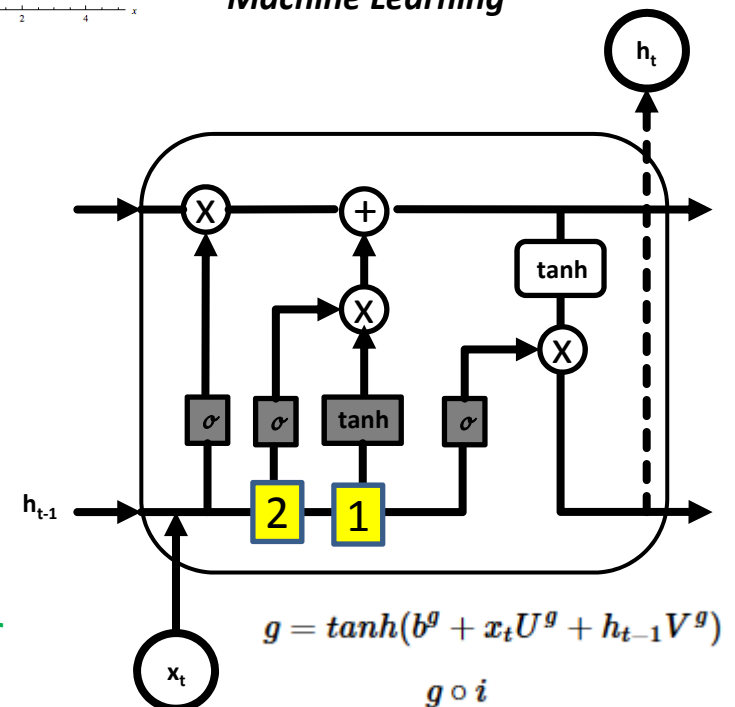
$$i = \sigma(b^i + x_t U^i + h_{t-1} V^i)$$



(uses sigmoid σ)

(gate sigmoid σ can act to 'switch off' any elements of the input vector that are not required)

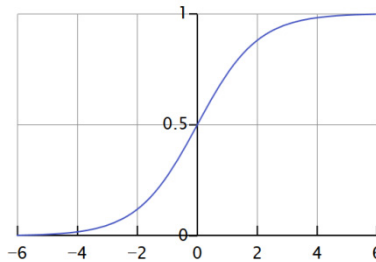
(sigmoid function outputs values between 0 and 1, weights connecting the input to these nodes can be trained to output values close to zero to 'switch off' certain input values – or outputs close to 1 to 'pass through')



Computing of LSTM Cell – Step 3

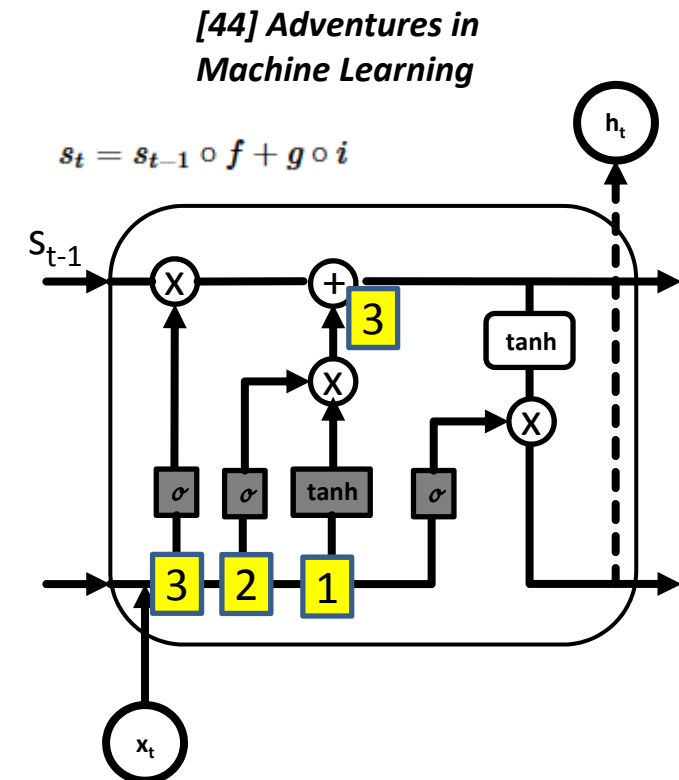
3. Internal state / forget gate $f = \sigma(b^f + x_t U^f + h_{t-1} V^f)$

- LSTM cells have internal cell state s_t
- ‘Delay’ – lagged one time step: s_{t-1}
- Added to the input data to create an effective ‘layer of recurrence’
- Addition instead of ‘usual’ multiplication reduces risk of vanishing gradients
- The connection to cell state is carefully controlled by a forget gate with sigmoid (works like the input gate)



(uses sigmoid σ)

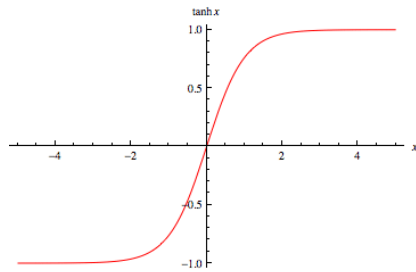
(gate sigmoid σ can act to ‘switch off’ any elements of the cell state to steer what variables should be remembered or forgotten)



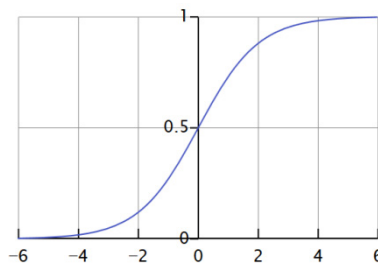
Computing of LSTM Cell – Step 4

4. Output layer & output gate $o = \sigma(b^o + x_t U^o + h_{t-1} V^o)$

- Output layer with tanh squashing function



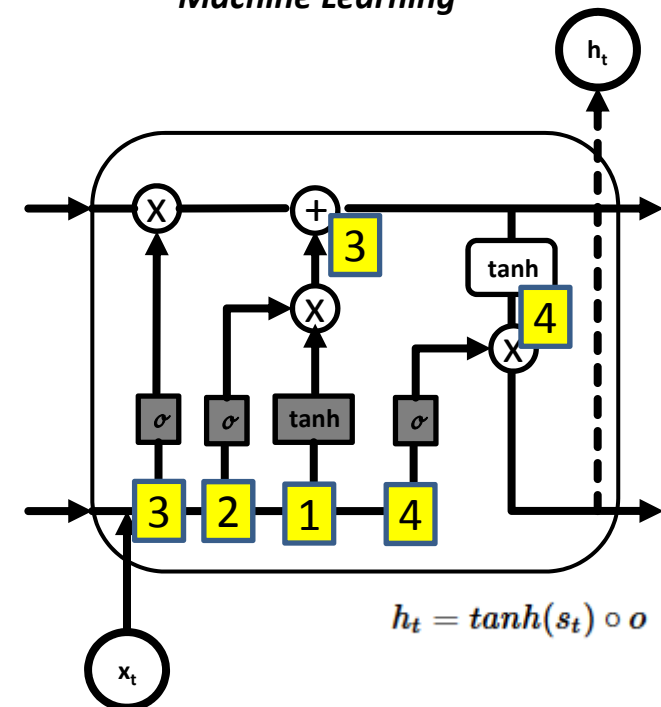
- Output is controlled via output gate with sigmoid activation function



(uses sigmoid σ)

(gate sigmoid σ can learn to determine which values are allowed as an output from the cell)

[44] *Adventures in Machine Learning*



Low-Level Tools – Tensorflow

- 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
- LSTM models are created using tensors & graphs and there are LSTM package contributions

[31] Tensorflow Deep Learning Framework

```
...
lstm = rnn_cell.BasicLSTMCell(lstm_size, state_is_tuple=False)
...
stacked_lstm = rnn_cell.MultiRNNCell([lstm] * number_of_layers,
    state_is_tuple=False)
...
initial_state = state = stacked_lstm.zero_state(batch_size, tf.float32)

for i in range(num_steps):
    # The value of state is updated
    # after processing each batch of words.
    output, state = stacked_lstm(words[:, i], state)

    # The rest of the code.
    # ...

final_state = s
```

- The class `BasicLSTMCell()` offers a simple LSTM Cell implementation in Tensorflow

High-level Tools – Keras

- 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

```
keras.layers.LSTM( units,  
                  activation='tanh',  
                  recurrent_activation='hard_sigmoid',  
                  use_bias=True,  
                  kernel_initializer='glorot_uniform',  
                  recurrent_initializer='orthogonal',  
                  bias_initializer='zeros',  
                  unit_forget_bias=True,  
                  kernel_regularizer=None,  
                  recurrent_regularizer=None,  
                  bias_regularizer=None,  
                  activity_regularizer=None,  
                  kernel_constraint=None,  
                  recurrent_constraint=None,  
                  bias_constraint=None,  
                  dropout=0.0, ...)
```





- Tool Keras supports the LSTM model via `keras.layers.LSTM()` that offers a wide variety of configuration options




Keras

[30] Keras Python Deep Learning Library

LSTM Example – Data Repository

 Search kaggle  Competitions Datasets Kernels Discussion Learn ...  



UMICH SI650 - Sentiment Classification

This is an in-class contest hosted by University of Michigan SI650 (Information Retrieval)
28 teams · 7 years ago

[Overview](#) [Data](#) [Leaderboard](#) [Rules](#) [Team](#) [My Submissions](#) [Late Submission](#)

Overview

Description	<p>This is a text classification task - sentiment classification. Every document (a line in the data file) is a sentence extracted from social media (blogs). Your goal is to classify the sentiment of each sentence into "positive" or "negative".</p>
Rules	<p>The training data contains 7086 sentences, already labeled with 1 (positive sentiment) or 0 (negative sentiment). The test data contains 33052 sentences that are unlabeled. The submission should be a .txt file with 33052 lines. In each line, there should be exactly one integer, 0 or 1, according to your classification results.</p> <p>You can make 5 submissions per day. Once you submit your results, you will get an accuracy score computed based on 20% of the test data. This score will position you somewhere on the leaderboard. Once the competition ends, you will see the final accuracy computed based on 100% of the test data. The evaluation metric is the inverse of the the mis-classification error - so the higher the better.</p> <p>You can use any classifiers, any features, and either supervised or semi-supervised methods. Be creative in both the methods and the usernames you select!</p>

[45] Kaggle, UMICH SI650 – Sentiment Classification Data

LSTM Example – Dataset & Application

- Sentiment analysis (many-to-one RNN topology)
 - Input: sentence as sequence of words (i.e. movie ratings texts)
 - Output: Sentiment value (positive/negative movie rating)
 - Application was a former competition (i.e. Kaggle platform overall idea)
 - Goal: Create LSTM network that will learn to predict a correct sentiment
- Small dataset example for tutorial: training & test data available
 - Training samples: 7086 short sentences (labelled) [~440 KB]
 - Test samples: 33052 short sentences[~1.94 MB]
 - Format: label & tab separated sentence
 - <https://www.kaggle.com/c/si650winter11/data>

Data Description

Training data: 7086 lines.
Format: 1|0 (tab) sentence

Test data: 33052 lines, each contains one sentence.

**[45] Kaggle, UMICH SI650 –
Sentiment Classification Data**

LSTM Example – Dataset Exploration

```
/homea/hpclab/train001/data/sentiments
[train001@jrl04 sentiments]$ ls -al
total 2912
drwxr-xr-x  2 train001 hpclab    512 Jun  7 06:33 .
drwxr-xr-x 12 train001 hpclab    512 Jun  7 05:55 ..
-rw-r--r--  1 train001 hpclab 2033345 Jun  7 06:44 testdata.txt
-rw-r--r--  1 train001 hpclab 447540 Jun  7 06:16 training-original.txt
-rw-r--r--  1 train001 hpclab 447540 Jun  7 06:10 training.txt
```

```
-bash-4.2$ head training.txt
1      The Da Vinci Code book is just awesome.
1      this was the first clive cussler i've ever read, but even books like Relic, and Da Vinci code were more plausible than this.
1      i liked the Da Vinci Code a lot.
1      i liked the Da Vinci Code a lot.
1      I liked the Da Vinci Code but it ultimatly didn't seem to hold it's own.
1      that's not even an exaggeration ) and at midnight we went to Wal-Mart to buy the Da Vinci Code, which is amazing of course.
1      I loved the Da Vinci Code, but now I want something better and different!..
1      i thought da vinci code was great, same with kite runner.
1      The Da Vinci Code is actually a good movie...
1      I thought the Da Vinci Code was a pretty good book.
```

(labelled training dataset)

```
-bash-4.2$ head testdata.txt
" I don't care what anyone says, I like Hillary Clinton.
have an awesome time at purdue!..
Yep, I'm still in London, which is pretty awesome: P Remind me to post the million and one pictures that I took when I get back to Markham!...
Have to say, I hate Paris Hilton's behavior but I do think she's kinda cute..
i will love the lakers.
I'm so glad I love Paris Hilton, too, or this would be excruciating.
considering most Geico commercials are stupid...
i liked MIT though, esp their little info book(
Before I left Missouri, I thought London was going to be so good and cool and fun and a really great experience and I was really excited.
I still like Tom Cruise.
```

(testing dataset)

LSTM Example – Keras Python Script – Preprocessing

```
from keras.layers.core import Activation, Dense, Dropout, SpatialDropout1D
from keras.layers.embeddings import Embedding
from keras.layers.recurrent import LSTM
from keras.models import Sequential
from keras.preprocessing import sequence
from sklearn.model_selection import train_test_split
import collections
import matplotlib.pyplot as plt
import nltk
import numpy as np
import os
```

```
# obtain punkt if not there already
nltk.download('punkt')
```

```
# define a data directory
DATA_DIR = "/homea/hpclab/train001/data/sentiments"
```

- Location for labeled training data and testset data

```
/homea/hpclab/train001/data/sentiments
[train001@jrl04 sentiments]$ ls -al
total 2912
drwxr-xr-x  2 train001 hpclab   512 Jun  7 06:33 .
drwxr-xr-x 12 train001 hpclab   512 Jun  7 05:55 ..
-rw-r--r--  1 train001 hpclab 2033345 Jun  7 06:44 testdata.txt
-rw-r--r--  1 train001 hpclab 447540 Jun  7 06:16 training-original.txt
-rw-r--r--  1 train001 hpclab 447540 Jun  7 06:10 training.txt
```

- Import necessary modules, e.g. LSTM for a simple LSTM cell, or Dense for a fully connected layer
- Import good sklearn model selection tools
- Import numpy for as helper tool

- Natural Language Toolkit (NLTK) is for building Python programs working on human language datasets (punkt is tokenizer)

[37] Deep Learning with Keras

LSTM Example – Keras Python Script – Vocabulary Setup

```
# exploratory analysis
maxlen = 0
word_freqs = collections.Counter()
num_recs = 0
ftrain = open(os.path.join(DATA_DIR, "training.txt"), 'rb')
for line in ftrain:
    label, sentence = line.strip().split('\t')
    words = nltk.word_tokenize(sentence.decode("ascii", "ignore").lower())
    if len(words) > maxlen:
        maxlen = len(words)
    for word in words:
        word_freqs[word] += 1
    num_recs += 1
ftrain.close()
```

- Perform exploratory analysis in order to find out the number of unique words in the whole corpus & how many words are roughly in each sentence

```
# vocabulary setup
MAX_FEATURES = 2000
MAX_SENTENCE_LENGTH = 40

# vocabulary and indices
vocab_size = min(MAX_FEATURES, len(word_freqs)) + 2
word2index = {x[0]: i+2 for i, x in
    enumerate(word_freqs.most_common(MAX_FEATURES))}
word2index["PAD"] = 0
word2index["UNK"] = 1
index2word = {v:k for k, v in word2index.items()}
```

- Exploration reveals maxlen: 42 & len(word_freqs): 2313
- Number of words in sentence (maxlen) enables a fixed sequence length & PAD = 0; truncate long ones

[37] Deep Learning with Keras

- Creating indices index2word and vice versa
- Out of vocabulary means UNK (unknown)

LSTM Example – Keras Python Script – Indices & Padding

```
# input sentence -> word index sequences
X = np.empty((num_recs, ), dtype=list)
y = np.zeros((num_recs, ))
i = 0
ftrain = open(os.path.join(DATA_DIR, "training.txt"), 'rb')
for line in ftrain:
    label, sentence = line.strip().split('\t')
    words = nltk.word_tokenize(sentence.decode("ascii", "ignore").lower())
    seqs = []
    for word in words:
        if word2index.has_key(word):
            seqs.append(word2index[word])
        else:
            seqs.append(word2index["UNK"])
    X[i] = seqs
    y[i] = int(label)
    i += 1
ftrain.close()
```

```
# perform padding if needed
X = sequence.pad_sequences(X, maxlen=MAX_SENTENCE_LENGTH)
```

```
# split into training and testing
Xtrain, Xtest, ytrain, ytest = train_test_split(X, y,
        test_size=0.2, random_state=42)
```

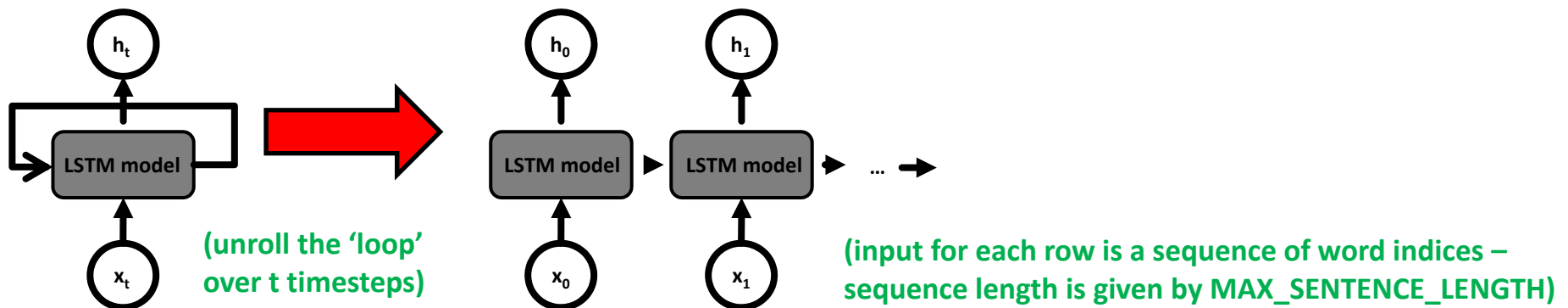
- Convert input sentences from the training data to word index sequences and add unknown ones as UNK in index

- Perform padding to the maximum sentence length (40)
- Labels are binary (positive/negative sentiment) and do not need padding

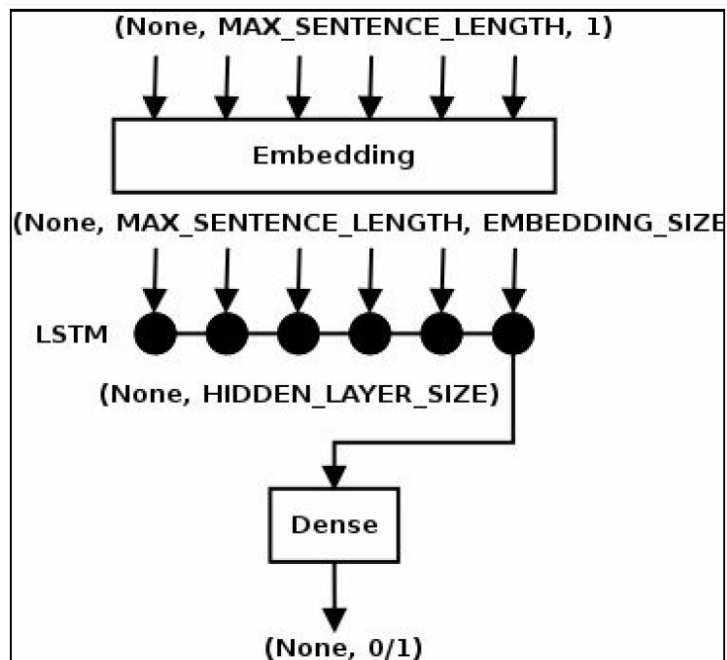
[37] *Deep Learning with Keras*

- Split between training & testing set (ratio rule of thumb 80:20)
- There is another test set put aside for nicely checking out-of-sample

LSTM Example – Modelling & Decisions



(tensor layout: `None X MAX_SENTENCE_LENGTH X 1`)



(tensor dimensions: first is `None` →

indicate that the batch size is currently unknown, i.e. number of records fed to the network → defined in runtime using `BATCH_SIZE` parameter)

(tensor fed to embedding layer →

weights are initialized with small random values & learned i.e. layer transforms the tensor to a shape of `None X MAX_SENTENCE_LENGTH X EMBEDDING_SIZE`)

(output of LSTM is the tensor

`None X HIDDEN_LAYER_SIZE`, because last tensor can be defined as `return_sequences = False` → we just need 0/1 output)

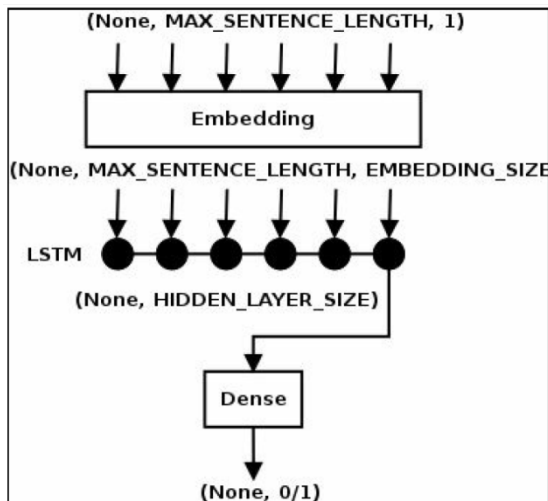
(Dense layer with Sigmoid activation function → 0 – negative review / 1 positive review)

modified from [37] Deep Learning with Keras

LSTM Example – Keras Python Script – Model & Parameter

```
#hyperparameter
EMBEDDING_SIZE = 128
HIDDEN_LAYER_SIZE = 64
BATCH_SIZE = 32
NUM_EPOCHS = 10
```

```
# LSTM model
model = Sequential()
model.add(Embedding(vocab_size, EMBEDDING_SIZE,
input_length=MAX_SENTENCE_LENGTH))
model.add(SpatialDropout1D(Dropout(0.2)))
model.add(LSTM(HIDDEN_LAYER_SIZE, dropout=0.2, recurrent_dropout=0.2))
model.add(Dense(1))
model.add(Activation("sigmoid"))
model.compile(loss="binary_crossentropy", optimizer="adam",
metrics=["accuracy"])
```



- Hyperparameters embedding=128; hidden layers=64; parameter by experimentation

- Create first embedding layer with input tensor None X maximum sentence length X 1
- Add regularizer SpatialDropout1D
- Add LSTM cell with hidden layer size 64 with regularizers dropout and recurrent_dropout
- Add Dense layer and Sigmoid activation

- All hyperparameters are tuned experimentally over many runs
- Compile model using binary cross-entropy loss function good for a binary model used here
- Use of Adam optimizer as good general purpose optimizer

[37] Deep Learning with Keras

LSTM Example – Keras Python Script – Train & Evaluate

```
# training
story = model.fit(Xtrain, ytrain, batch_size=BATCH_SIZE, epochs=NUM_EPOCHS,
validation_data=(Xtest, ytest))

# evaluation
score, acc = model.evaluate(Xtest, ytest, batch_size=BATCH_SIZE)
print("Test score: %.3f, accuracy: %.3f" % (score, acc))
for i in range(5):
    idx = np.random.randint(len(Xtest))
    xtest = Xtest[idx].reshape(1,40)
    ylabel = ytest[idx]
    ypred = model.predict(xtest)[0][0]
    sent = " ".join([index2word[x] for x in xtest[0].tolist() if x != 0])
    print("%.0ft %dt %s" % (ypred, ylabel, sent))
```

- Supervised learning process
 - Labels existing (not in this unsupervised example)
 - Train model for fixed number of epochs
 - Evaluate model against test dataset (splitted training)

- Train the LSTM network for 10 epochs (NUM_EPOCHS) & with batch size 32
- Perform validation at each epoch using test data

- Evaluate model against the full test set showing score and accuracy
- Show the LSTM prediction with pick of a few random sentences from the test set (predicted label, label & actual sentence)

LSTM Example – Submit Script (JURECA)

- Job submit script
 - Specify good name for the job
 - Allocate GPUs for deep learning job
 - Specify job queue
 - Restore module environment with all dependencies
 - Use python with lstm-example.py script
- Use sbatch
 - Use job script

```
#!/bin/bash -x
#SBATCH--nodes=1
#SBATCH--ntasks=1
#SBATCH--output=lstm_out.%j
#SBATCH--error=lstm_err.%j
#SBATCH--time=01:00:00
#SBATCH--mail-user=m.riedel@fz-juelich.de
#SBATCH--mail-type=ALL
#SBATCH--job-name=simple-LSTM

#SBATCH--partition=gpus
#SBATCH--gres=gpu:1

#SBATCH--reservation=deep_learning

### location executable
KERASSCRIPT=/homea/hpclab/train001/tools/lstm/lstm-example.py

module restore dl_tutorial

### submit
python $KERASSCRIPT
```

LSTM Example – Output Interpretation

- Supervised learning problem
 - Check output with 'more out.txt'
 - Idea: predicted sentiment should be closed to sentiment labels
 - More epochs/iterations → better quality of the model

(learned well
compared to
first iteration →
one can observe
loss decrease and
increase in
accuracy over
multiple epochs)

```
Train on 5668 samples, validate on 1418 samples
Epoch 1/10

 32/5668 [.....] - ETA: 35:08 - loss: 0.6938 - acc: 0.4688
 64/5668 [.....] - ETA: 17:36 - loss: 0.6927 - acc: 0.5312
 96/5668 [.....] - ETA: 11:45 - loss: 0.6911 - acc: 0.5625
```

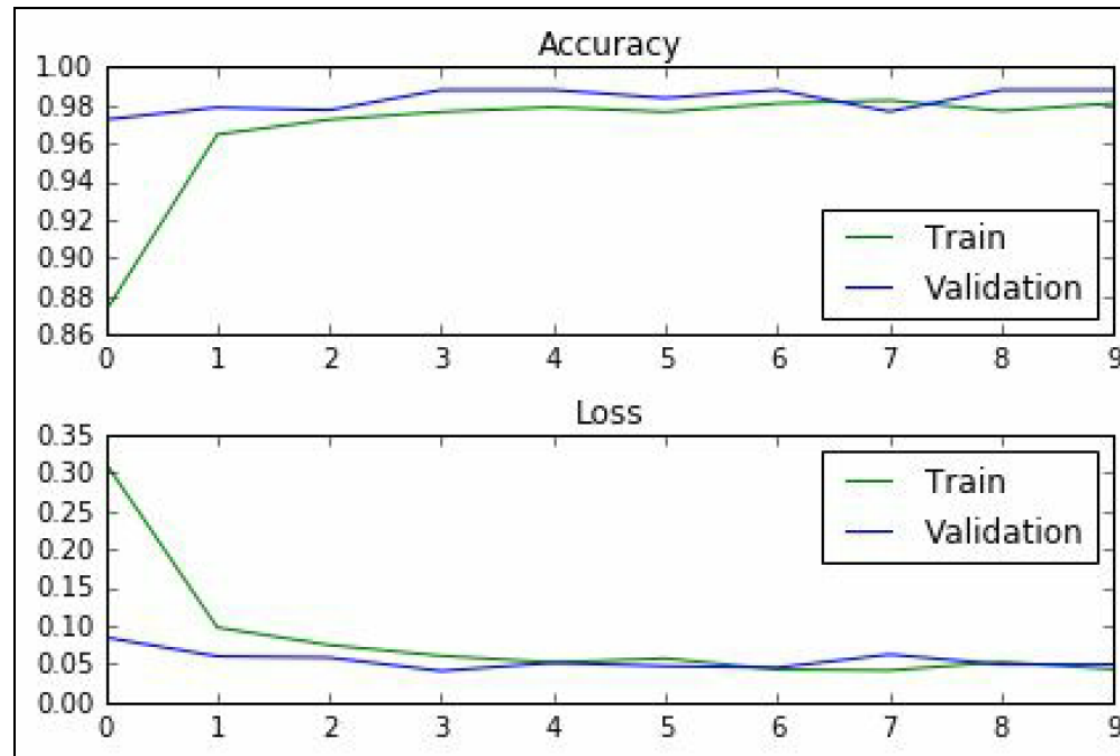
```
5664/5668 [=====>.] - ETA: 0s - loss: 0.0015 - acc: 0.9995
5668/5668 [=====] - 15s 3ms/step - loss: 0.0015 - acc: 0.9995 - val_loss: 0.0845 - val_acc: 0.9718
Epoch 10/10

 32/5668 [.....] - ETA: 13s - loss: 0.0697 - acc: 0.9688
 64/5668 [.....] - ETA: 13s - loss: 0.0353 - acc: 0.9844
 96/5668 [.....] - ETA: 13s - loss: 0.0240 - acc: 0.9896
```

```
Test score: 0.072, accuracy: 0.980
1t 1t the people who are worth it know how much i love the da vinci code .
1t 1t anyway , thats why i love `` brokeback mountain .
0t 0t the da vinci code sucked .
0t 0t this quiz sucks and harry potter sucks ok bye..
1t 1t because i would like to make friends who like the same things i like ,
```


LSTM Example – Model Evaluation

- Selected plots (e.g. for papers)
 - E.g. [matplotlib](#) & [pyplot](#) can be used to create simple graphs



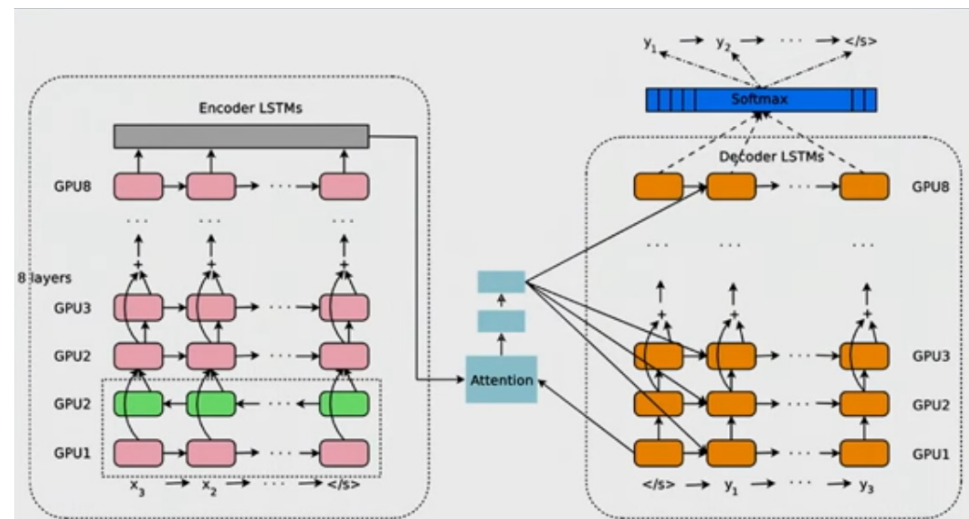
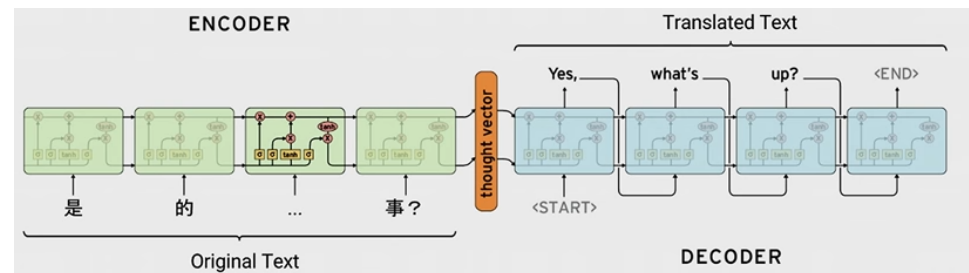
[37] Deep Learning with Keras

Different Useful LSTM Models – Many other Applications

- **Standard LSTM**
 - Memory cells with **single LSTM layer**; used in simple network structures
- **Stacked LSTM**
 - **LSTM layers are stacked** one on top of another; creating deep networks
- **CNN LSTM**
 - CNNs to learn features (e.g. images); **LSTM for image sequences**
- **Encoder-Decoder LSTM**
 - One LSTM network → **encode input**; one LSTM network → **decode output**
- **Bidirectional LSTM**
 - Input sequences are presented and **learned both forward & backwards**
- **Generative LSTM**
 - LSTMs **learn the inherent structure relationship** in input sequences; then **generate new plausible sequences**

Tensorflow – LSTM Google Translate Example & GPUs

- Use of 2 LSTM networks in a stacked manner
 - Called 'sequence-2-sequence' model
 - Encoder network
 - Decoder network
- Needs context of sentence (memory) for translation



[47] Sequence Models

[Video] RNN & LSTM

SOLUTION

Gating units - LSTM, GRU

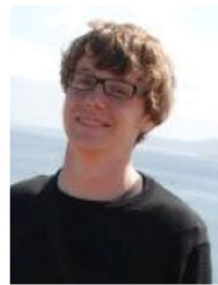
information

it for future time steps. The most popular gating types today are GRU and LSTM. Besides



[46] Recurrent Neural Networks, YouTube

Acknowledgements – Membership of my HPDP Research Group



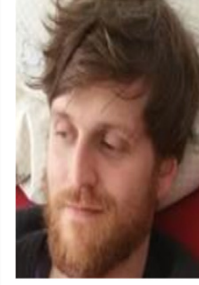
PD Dr.
G. Cavallaro



Senior PhD
Student A.S. Memon



Senior PhD
Student M.S. Memon



PhD Student
E. Erlingsson



PhD Student
C. Bakarar



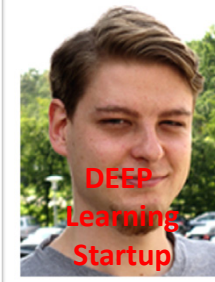
Dr. M. Goetz
(now KIT)



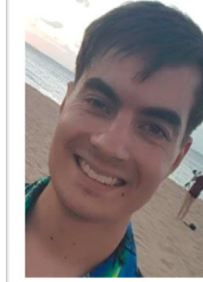
MSc M.
Richerzhagen,
now other group



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>

