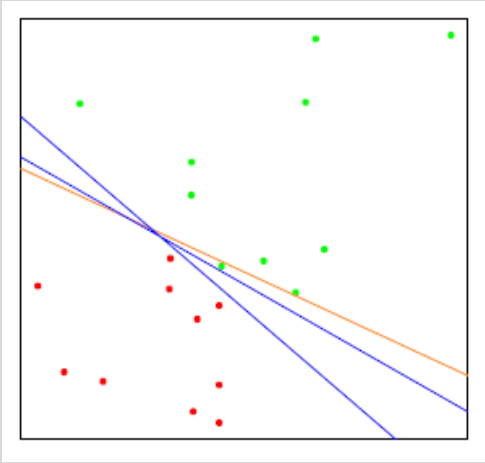


Parallel & Scalable Machine Learning

Introduction to Machine Learning Algorithms

Dr. – Ing. Morris Riedel
Adjunct Associated Professor, University of Iceland
Research Group Leader, Juelich Supercomputing Centre, Germany
Ernir Erlingsson
PhD Student, University of Iceland & DEEP-EST Application Engineer

LECTURE 11

Parallelization Benefits

January 17th, 2018
JSC, Germany

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

JÜLICH FORSCHUNGSZENTRUM

PRACE PARTNERSHIP FOR ADVANCED COMPUTING IN EUROPE TRAINING PORTAL

Review of Lecture 10

- 10-fold Cross-Validation
  - Validation estimates for out of sample performance only on training Dataset is remarkably well
  - Training Dataset is validation Dataset using 10-fold cross-validation
  - Test Dataset kept seperate
- Regularization
  - Parameter C important to steer how much error allowed

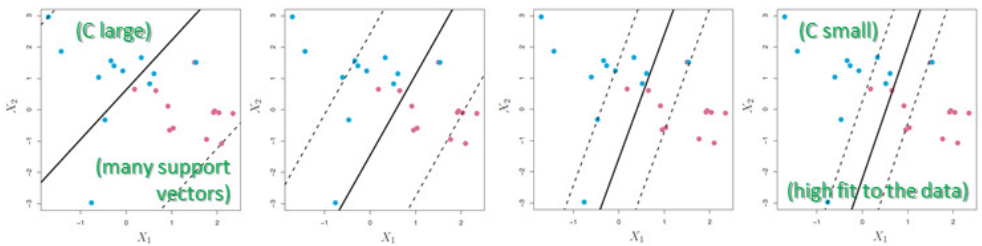
```
#SBATCH--job-name=10x-cv-indianpines-4-96-24

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

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

### submit
srun $PISVM -v 10 -D -o 1024 -q 512 -c 100 -g 8 -t 2 -m 1024 -s 0 $TRAINDATA
```

```
[train001@jrl12 pisvm-1.2.1]$ tail mpi-out.4642829
Cross Validation Accuracy = 0.0658762%
I/O time = 0.76
Cross Validation Accuracy = 0.0658762%
I/O time = 0.76
Cross Validation Accuracy = 0.0658762%
I/O time = 0.76
Cross Validation Accuracy = 0.0658762%
I/O time = 0.76
Cross Validation Accuracy = 77.0242%
I/O time = 0.77
```

(cross-validation accuracy on training set)

```
[train001@j3l02 pisvm-1.2.1]$ more mpi-out.15244572
Accuracy = 77.9678% (234336/300555) (classification)
Mean squared error = 153.787 (regression)
Squared correlation coefficient = 0.601418 (regression)
```

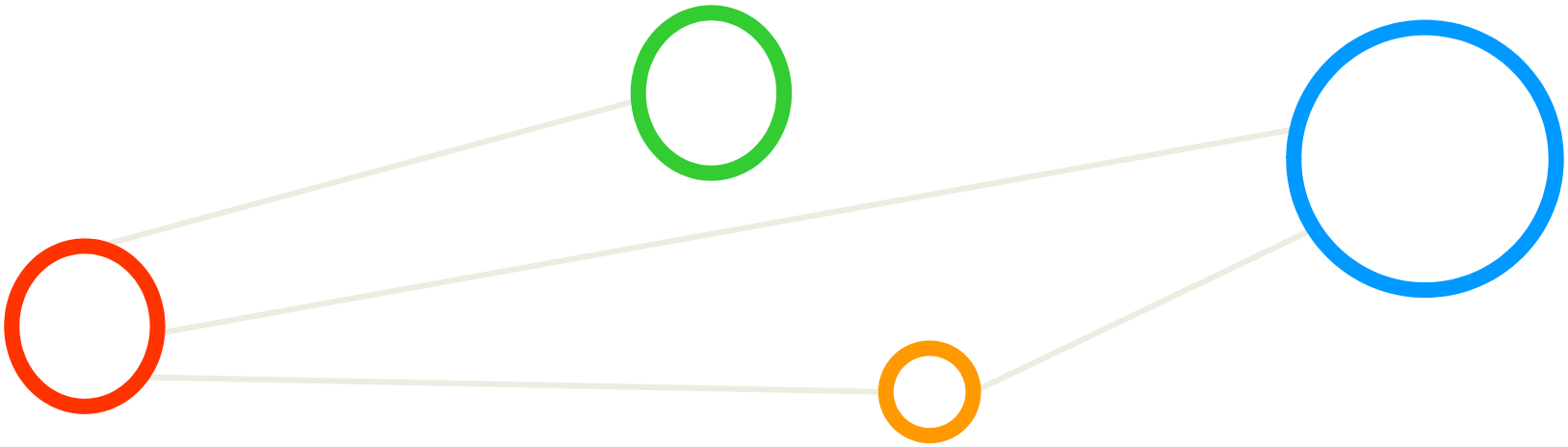
(testing on test set – 'simulating out-of-sample')



(fewer support Vectors: just #8)

[16] An Introduction to Statistical Learning

Outline



Outline of the Course

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

Day One – beginner

Day Two – moderate

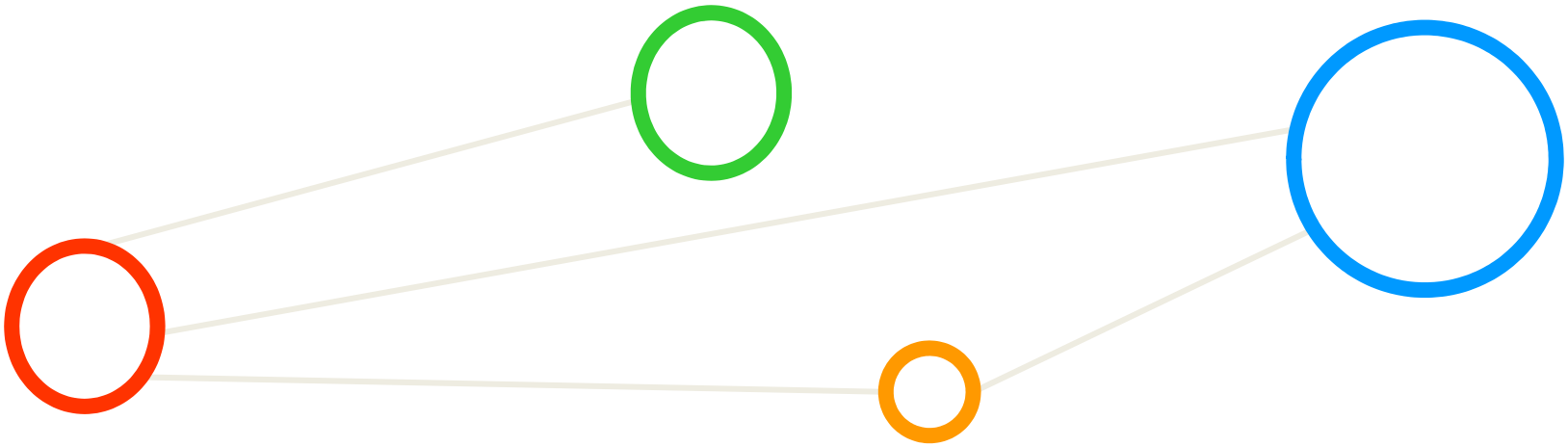
Day Three – expert

Outline

- Parallelization Benefits
  - Regularization Parameter Revisited
  - Possibility to work with large datasets
  - First Level of Parallelization in piSVM
  - Parallelization Impact in Cross-Validation
  - Second Level of Parallelization in GridSearch

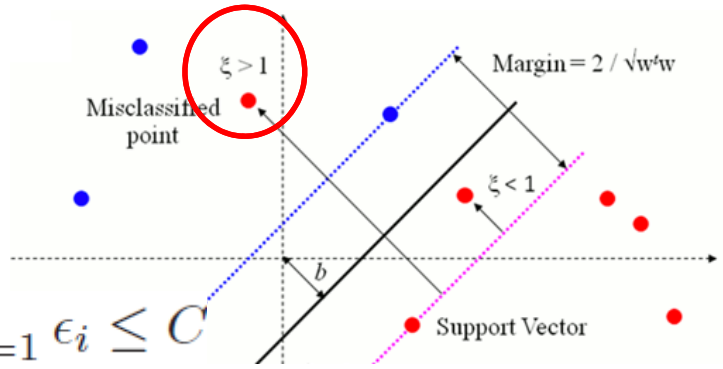
- Parallelization Benefits in Deep Learning
  - Biological Inspiration & Perceptron Limits
  - Artificial Neural Networks & Backpropagation
  - Application Examples in Science & Industry
  - Deep Learning Properties & Feature Learning
  - Parallel Computing Methods & Architectures

Parallelization Benefits



Regularization Revisited & Rules of Thumb for C

- C = 0 (too rectrictive, potentially bad for E_{out}) $\epsilon_1 = \ldots = \epsilon_n = 0$
  - No budget/costs for violations: comparable to maximal margin classifier
  - Further constraint: only works in linearly seperable cases (less in practice)
- C > 0 (flexible option, better for E_{out})
  - No more than C data points can be on the wrong side of the hyperplane ('how much misclassifications allowed')
  - Reasoning: if an observation is on the wrong side then $\epsilon_i > 1$



$\sum_{i=1}^{n} \epsilon_i \leq C$

(rule of thumb)

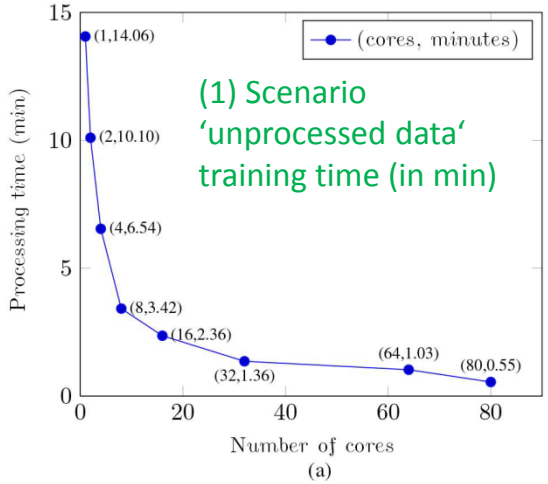
(differently handled in R library)

- **regularization parameter C (budget of errors) increase → margins will be wide and more tolerant of violations to the margin (<u>classifier fits data less</u>)**
- **regularization parameter C (budget of errors) descreases → margins will be narraw and less tolerant of violations to the margin (<u>classifier highly fit data</u>)**

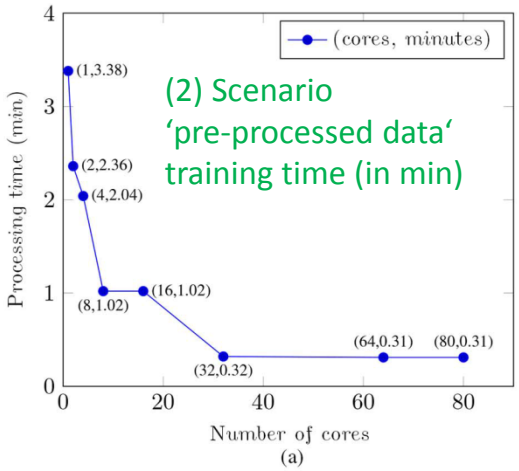
- **Determine the right C parameter for a model can be obtained using parallelization on a HPC system**

Parallelization Benefit: Lower-Time-To-Solution

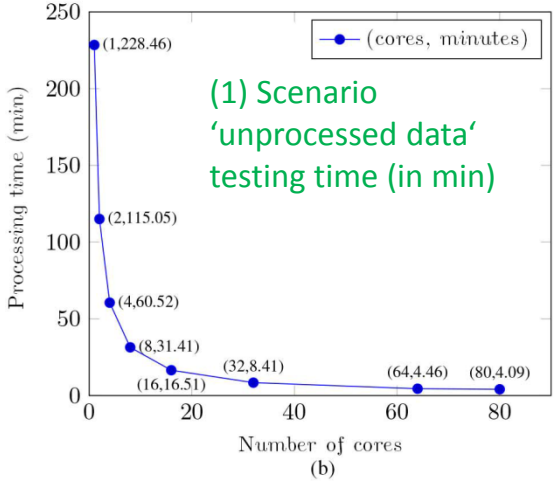
- Major speed-ups; ~interactive (<1 min); same accuracy;



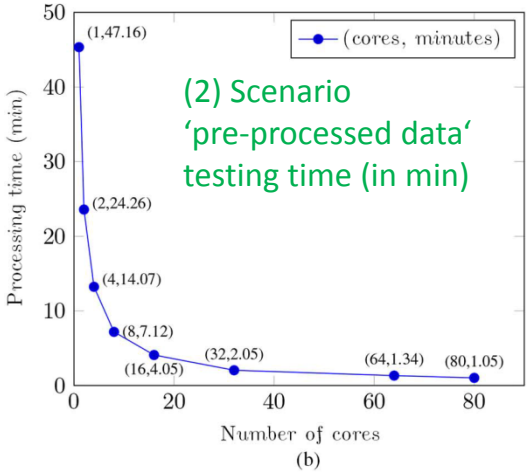
(1) Scenario
'unprocessed data'
training time (in min)



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



(1) Scenario
'unprocessed data'
testing time (in min)



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

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)

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

(aka first level of parallelism)

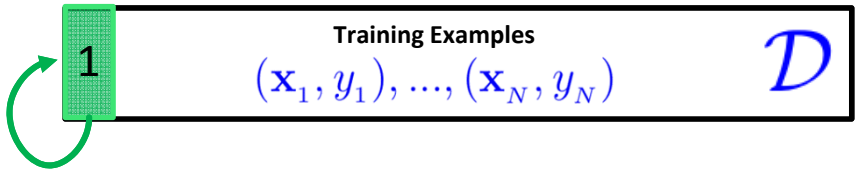
B2SHARE
Store and Share Research Data

Validation Technique – Cross-Validation – Revisited

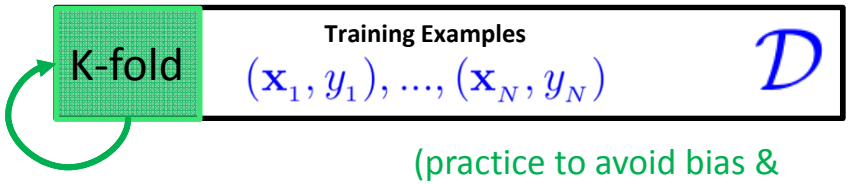
- **10-fold cross validation is mostly applied in practical problems by setting K = N/10 for real data**
- **Having N/K training sessions on N – K points each leads to long runtimes (→ use parallelization)**

- 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-valdation' with K = N/10 multiple times (N/K)
    (use 1/10 for validation, use 9/10 for training, then another 1/10 … N/K times)

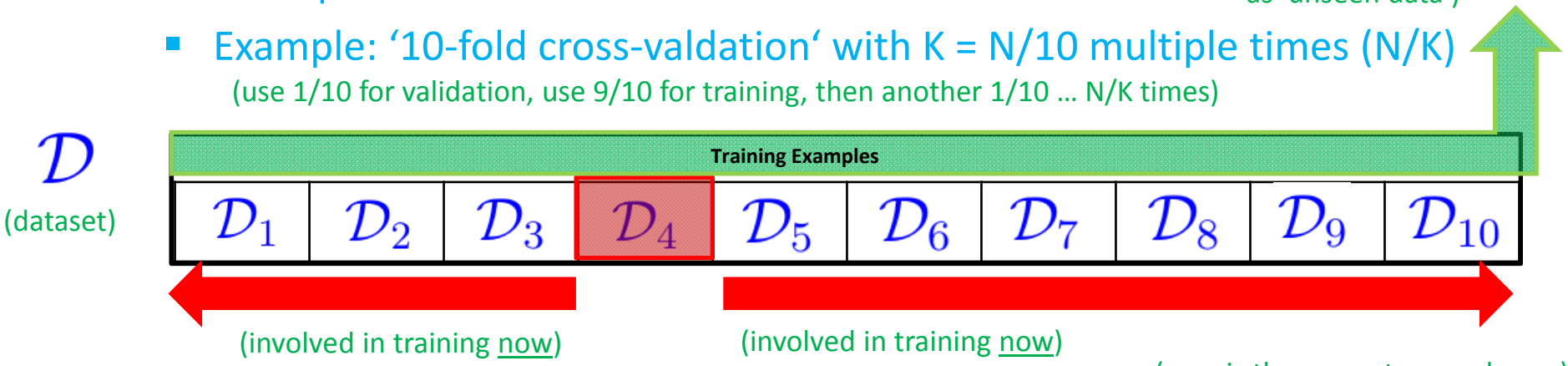
(leave 1 point out at each run → many runs)



(generalization to leave k points out at each run)



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



(now is the current example run)

Parallelization Benefit: Parallel 10-Fold Cross-Validation

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

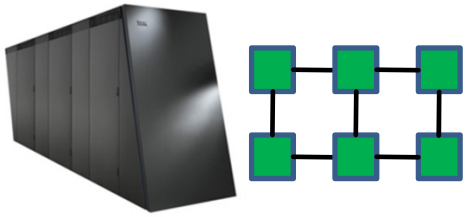
(2) Scenario ‘pre-processed data‘, 10xCV **serial**: accuracy (min)

γ/C	1	10	100	1000	10 000
2					
4					
8					
16					
32					

In Matlab one after another

(1) First Result: best parameter set from 14.41 min to 1.02 min

(2) Second Result: all parameter sets from ~9 hours to ~35 min



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

γ/C	1	10	100	1000	10 000

second level of parallelism

‘(1) each cell inherent parallel’

‘(2) all cells in parallel’

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

Parallelization Summary – aka 'GridSearch'

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

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

γ/C	1	10	100	1000	10 000
2	second level of parallelism				
4					
8					
16					
32					

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

γ/C	1	10	100	1000	10 000
2	second level of parallelism				
4					
8					
16					
32					

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

γ/C	1	10	100	1000	10 000
2	second level of parallelism				
4					
8					
16					
32					

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

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

γ/C	1	10	100	1000	10 000
2	second level of parallelism				
4					
8					
16					
32					

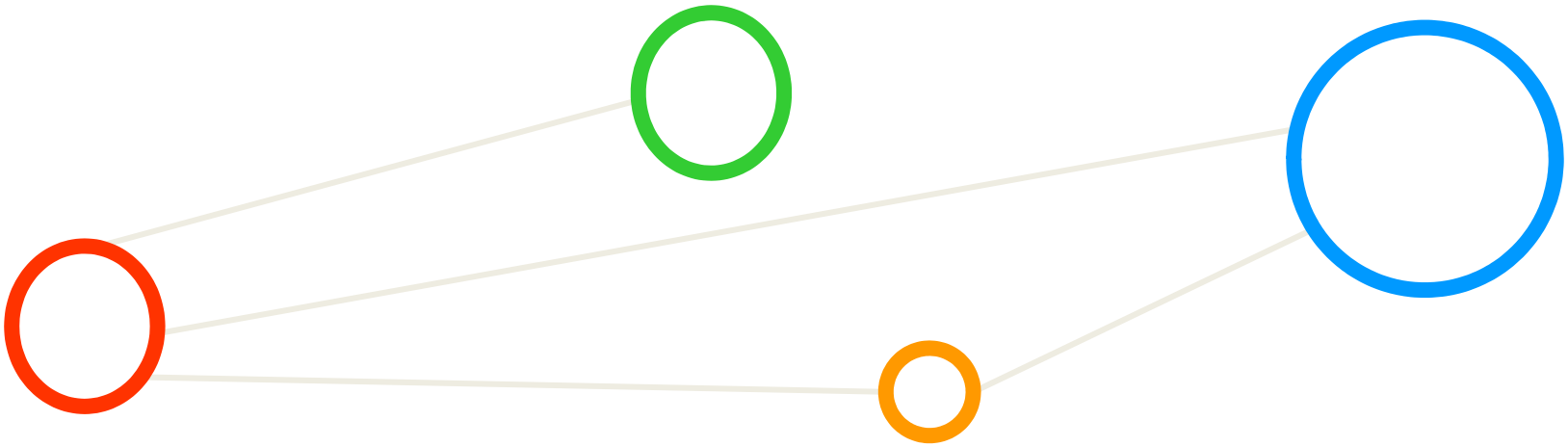
First Result: best parameter set from 14.41 min to 1.02 min
Second Result: all parameter sets from ~9 hours to ~35 min

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

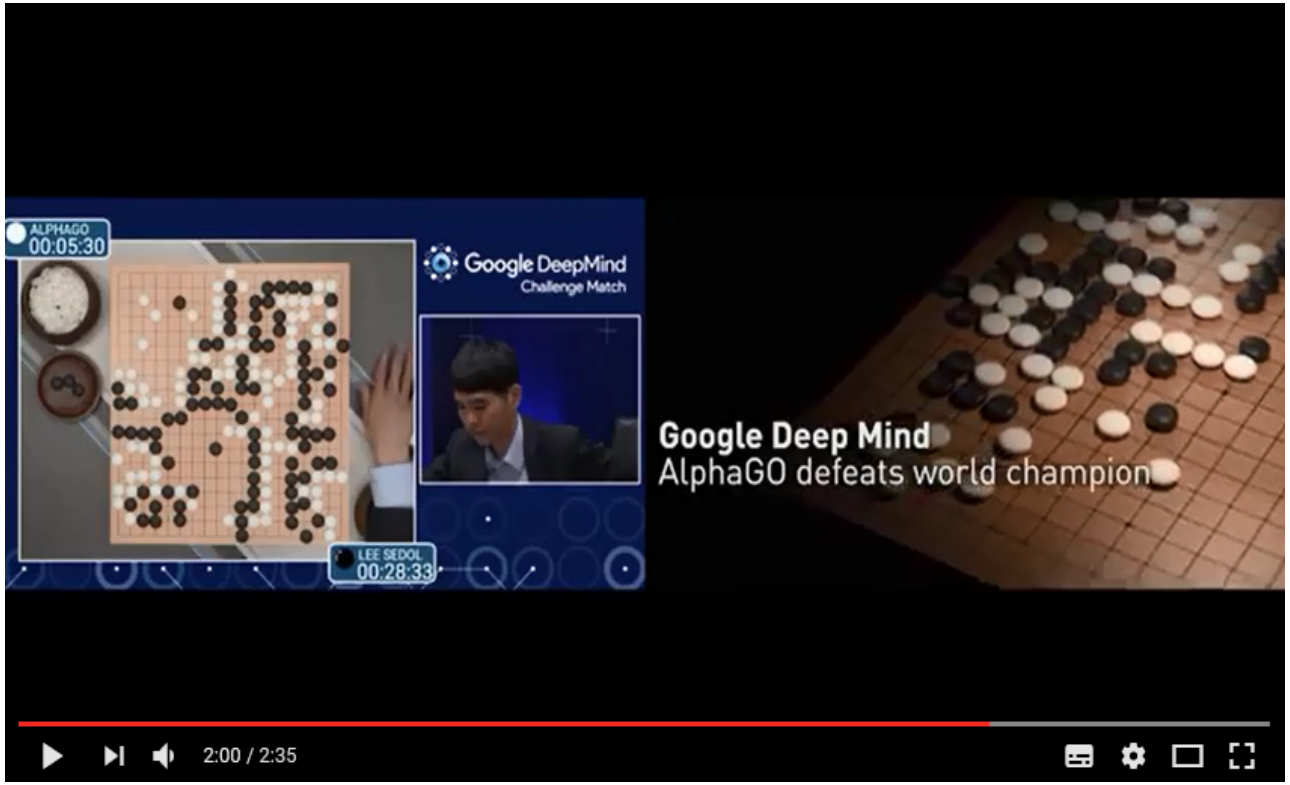
B2SHARE
Store and Share Research Data

Exercises – Check Cross-Validation Runs

Parallelization Benefits in Deep Learning

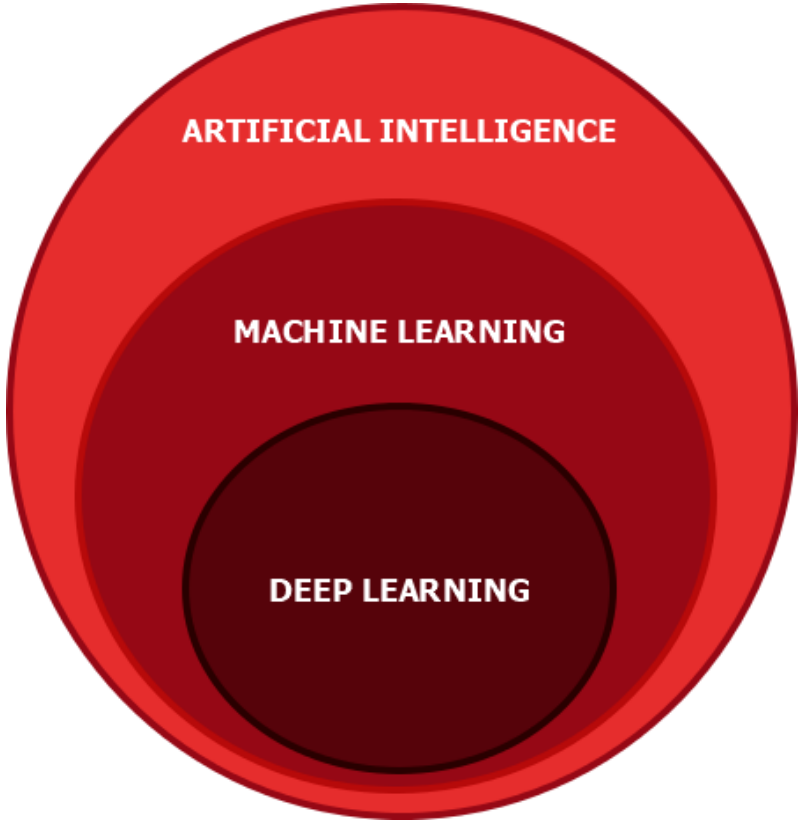


Deep Learning – What is it



[1] The Deep Learning Revolution, YouTube

Deep Learning – Definition



- **Artificial Intelligence**
  - The concept of machines being able to carry out tasks in a way we would consider intelligent
- **Machine Learning**
  - Computer systems that improve with experience and data
- **Deep Learning**
  - Is a subset of machine learning where the system is represented as a nested hierarchical features, where each feature is defined in relation to simpler features.

Deep Learning – Introduction

- Deep learning is a brand that comes in many, ever-increasing, flavours. It has had previous names, including:
  - Cybernetics (1940s)
  - Neural networks (1980s)
  - Deep learning (2006)

- Has gained traction very fast with no immediate signs of slowing down and is sometimes characterized as a buzzword

- It is used for supervised, semi-supervised and unsupervised learning.
  - *Supervised learning* uses labelled data
  - *Semi-supervised uses mostly unlabelled data but not all.*
  - *Unsupervised learning uses only unlabelled data*

- It´s currently possible effect is both *overestimated* and *underestimated*.

Deep Learning – Introduction

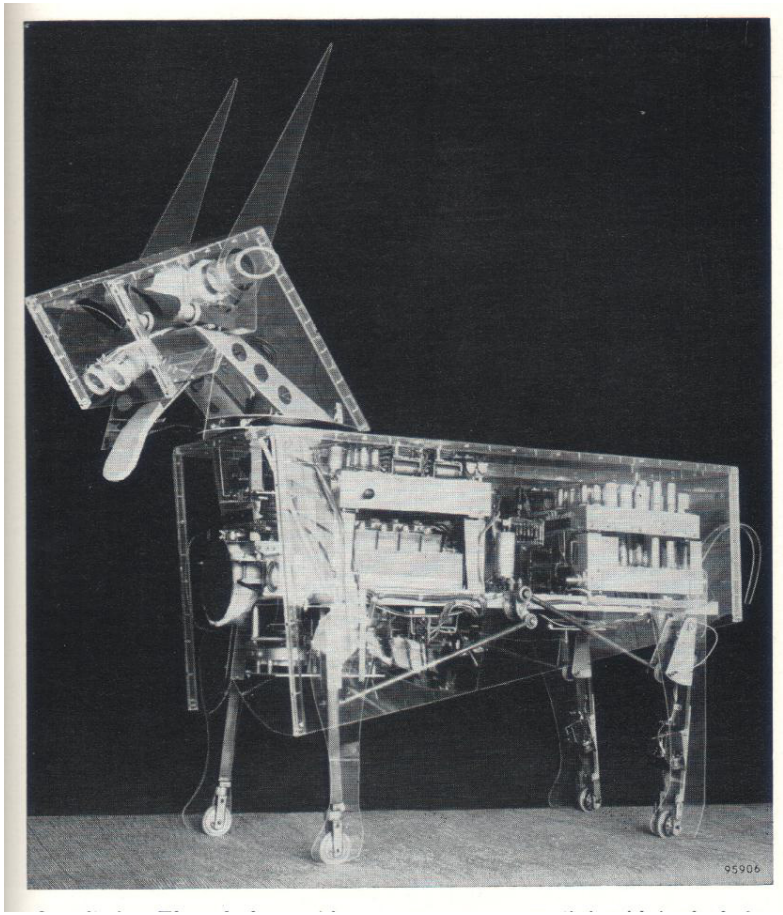
- Application areas
  - Computer vision
  - Automatic speech recognition
  - Natural language processing
  - Bioinformatics
  - And much more...

Deep Learning - Introduction

Where do human ideas and innovations come from ?

- Inspired by nature
  - First we **observe** and then we try to **replicate**

Deep Learning - Introduction

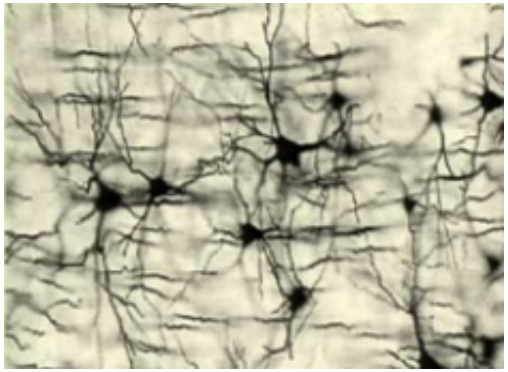


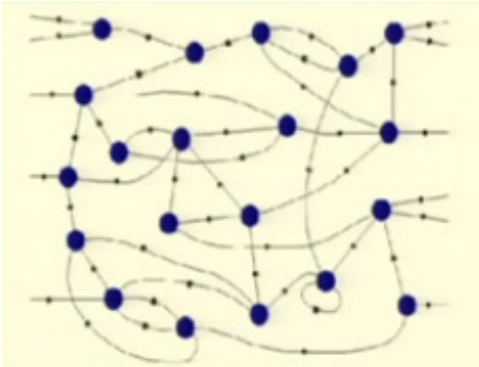
1950s Cybernetics: Cyber the dog

[2] 'Cybernetic Zoo' web page

Deep Learning - Introduction

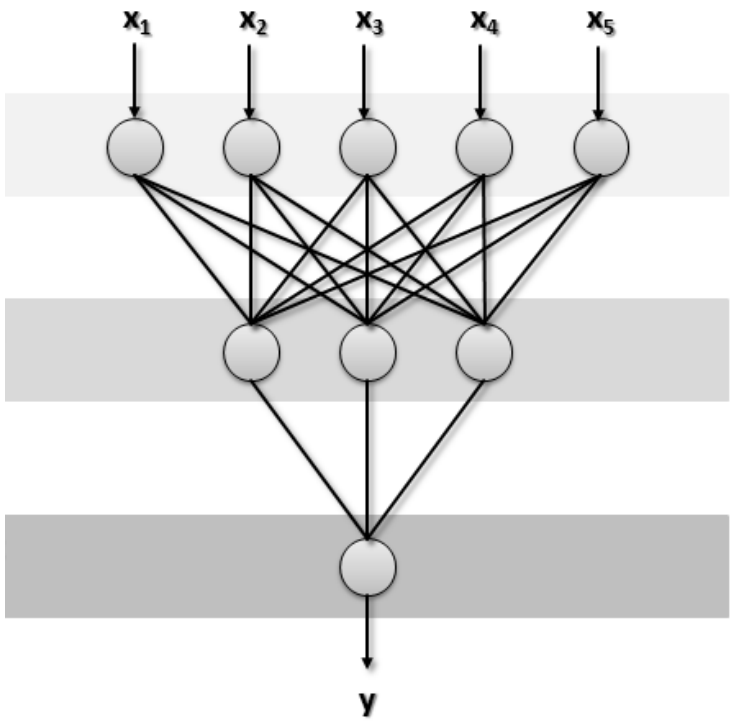
Neural Networks (NNs) is an attempt at replicating neural functions of the brain to solve problems.





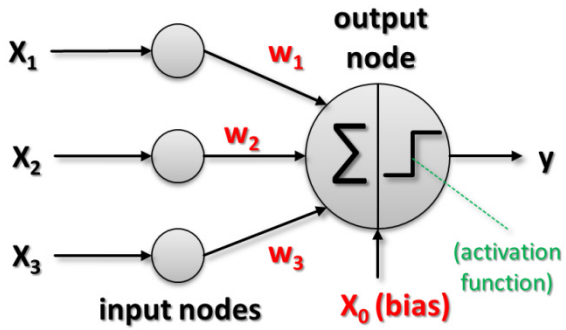
Caveat: Not considered to be an accurate model but rather based on how neurons interconnect. Furthermore, we don´t know well enough how the brain operates to properly replicate it, e.g. what is a consciousness ?

Artificial Neural Network (ANN)

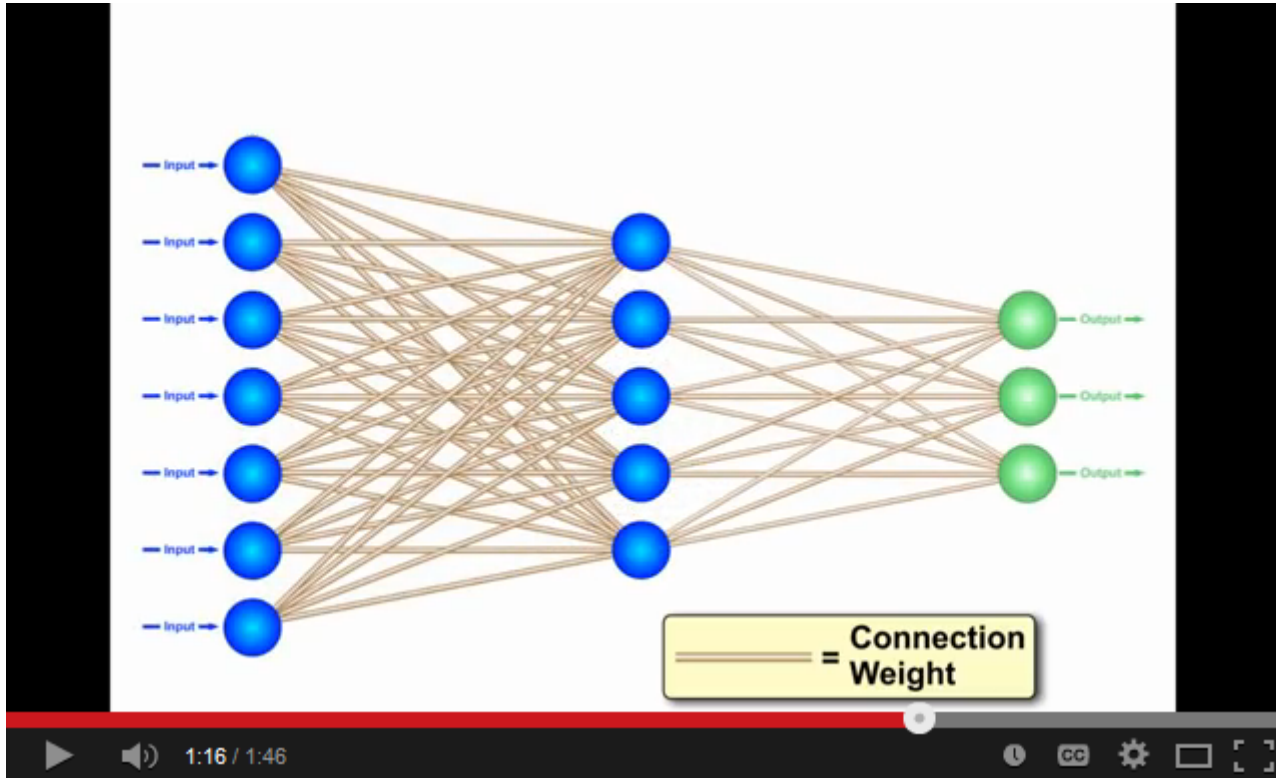


- A computational model of biological learning
- synomous with deep learning
- The nodes simulate neurons and the edges simulate synapses with weight values.

- Neurons modelled as perceptrons that "fire" their activation function when the sum of weights crosses a certain threshold.



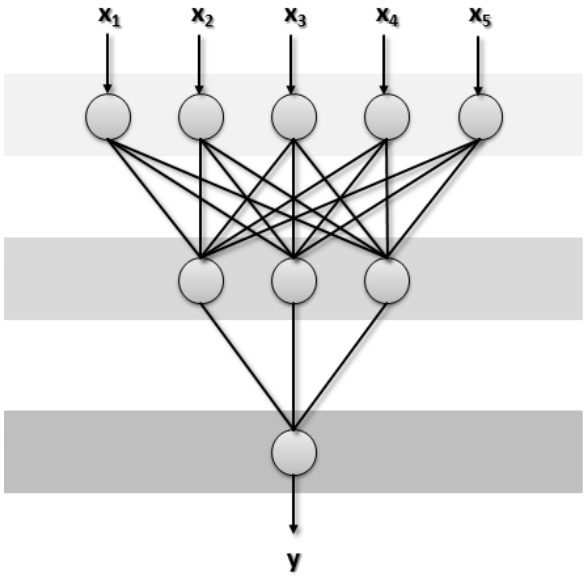
[Video] Towards Multi-Layer Perceptrons

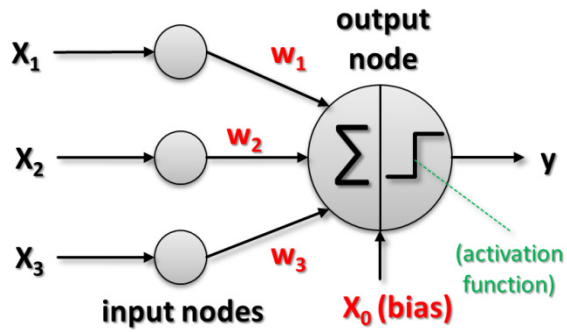


[3] YouTube Video, Neural Networks – A Simple Explanation

Neural Networks - Timeline

- 1943: The first mathematical model of the human brain
- 1957: Perceptron
- 1965: The first multi-layered network
- 1987: Multi-layered Perceptron (backpropagation)
- 1995: Support Vector Machines (SVMs)
- 1998: Gradient based learning
- 2006: Deep Neural Network
- 2011: AlexNet (CNNs)
- 2014: Generative Adversarial Networks (GAN)



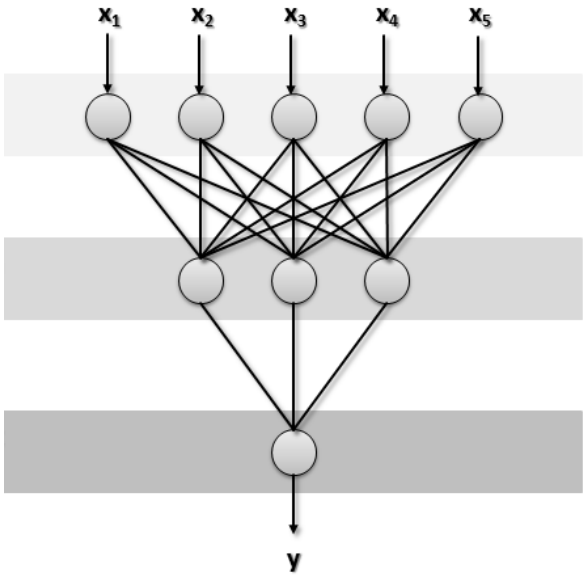


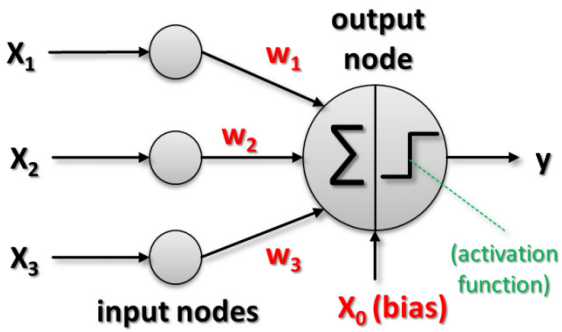
Neural Networks - Resurgence

The renaissance of neural networks via deep learning, accelerated by:

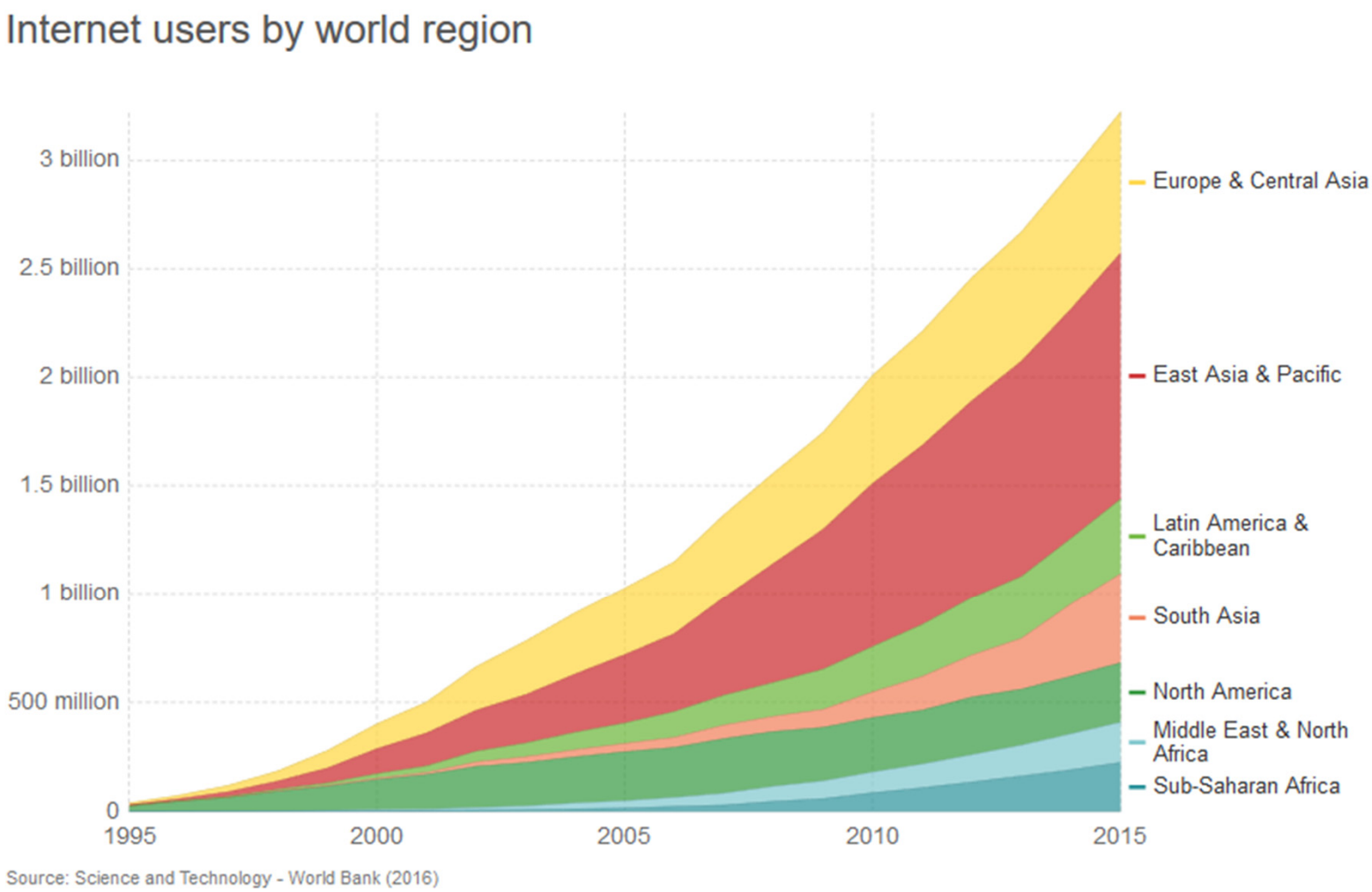
- Big Data
  - The first web page 1992
  - 163 zettabytes (1 million petabytes) forecasted for 2025 by the International Data Corporation (IDC)

- Advances in Computation
  - Multi-core CPUs
  - Many-core GPUs
  - Parallelism the key





Big Data – Internet Users



[4] Our world in data, Online

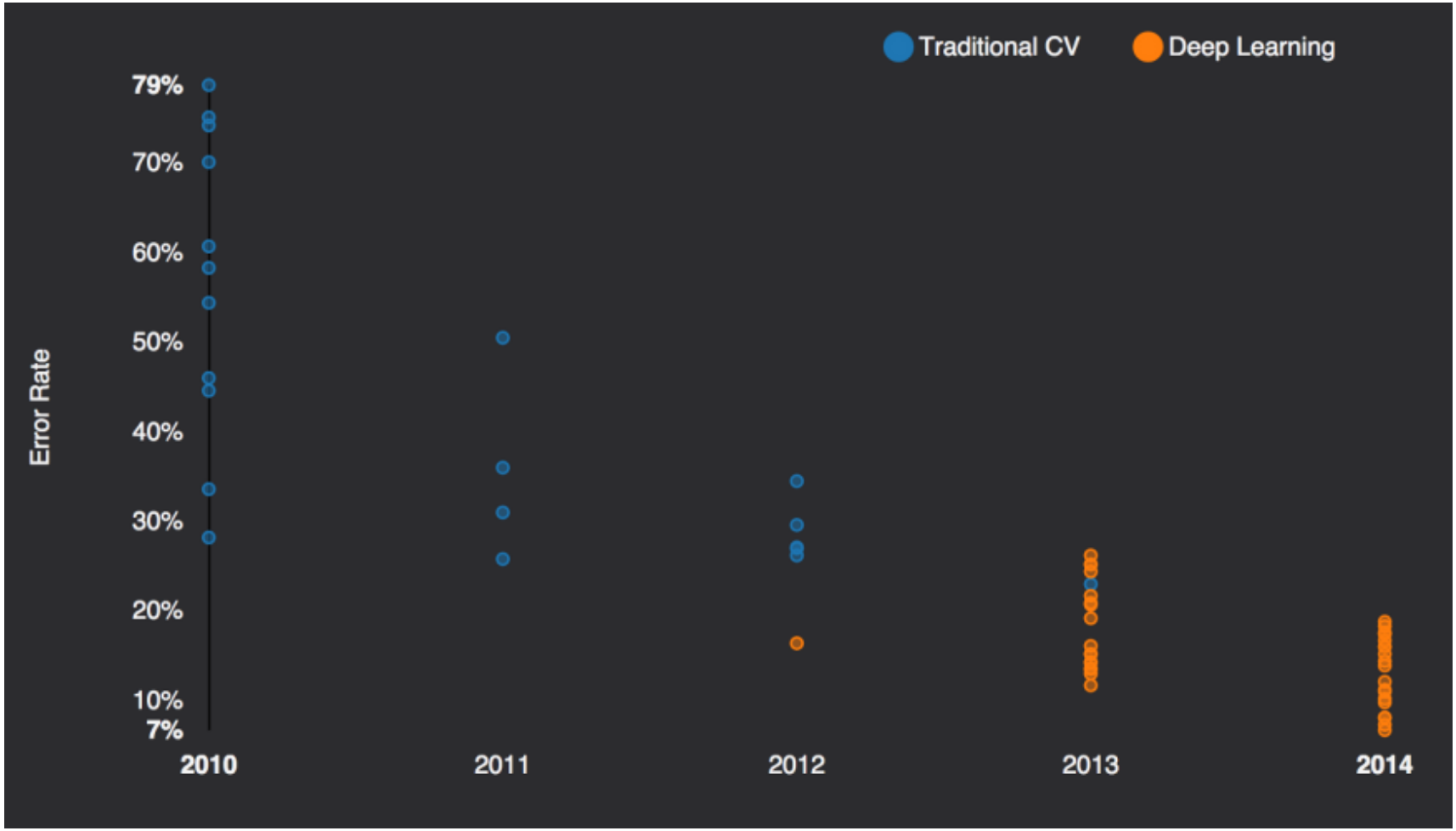
Big Data - ImageNet Dataset

- Dataset: **ImageNet**
  - Total number of images: *14.197.122*

High level category	# synset (subcategories)	Avg # images per synset	Total # images
amphibian	94	591	56K
animal	3822	732	2799K
appliance	51	1164	59K
bird	856	949	812K
covering	946	819	774K
device	2385	675	1610K
fabric	262	690	181K

[5] ImageNet Web page

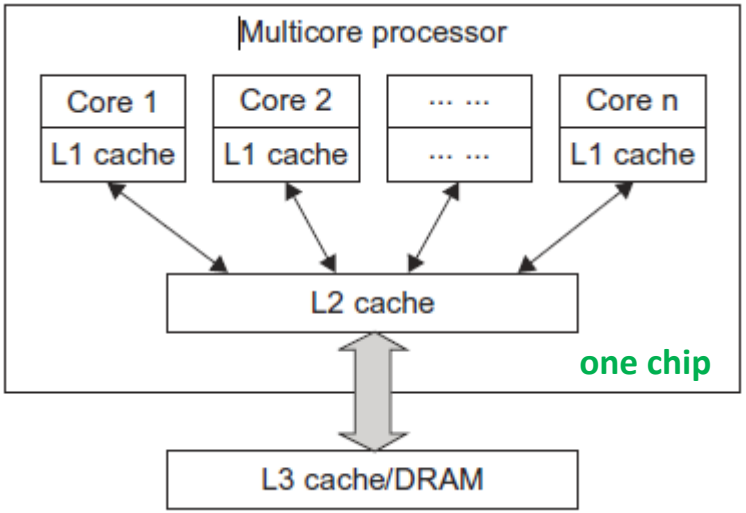
Big Data - ImageNet Dataset



[6] J. Dean et al., 'Large-Scale Deep Learning'

Advances in Computation - Multi-core CPUs

- Significant advances in CPU (or microprocessor chips)
  - **Multi-core architecture** with dual, quad, six, or n processing cores
  - Processing cores are all on one chip
- Multi-core CPU chip architecture
  - Hierarchy of caches (on/off chip)
  - L1 cache is private to each core; on-chip
  - L2 cache is shared; on-chip
  - L3 cache or Dynamic random access memory (DRAM); off-chip

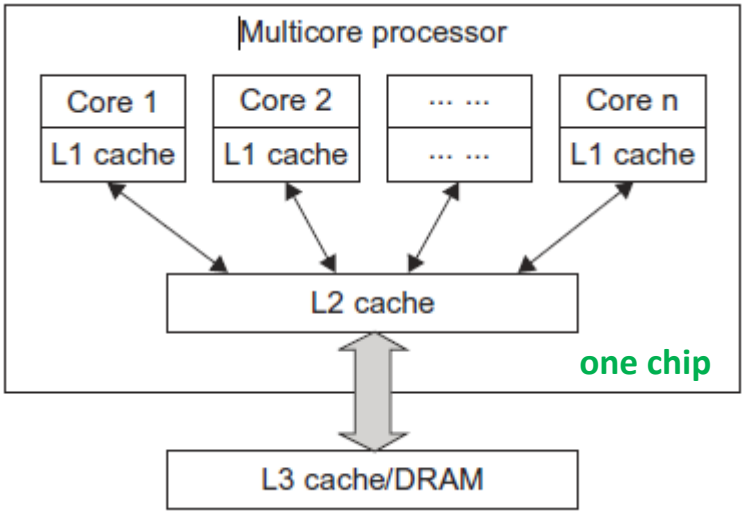


[7] Distributed & Cloud Computing Book

Advances in Computation - Multi-core CPUs

- Clock-rate for single processors increased from 10 MHz (Intel 286) years to 4 GHz (Pentium 4) in 30 years

- Clock rate increase with higher 5 GHz reached a limit due to power limitations / heat

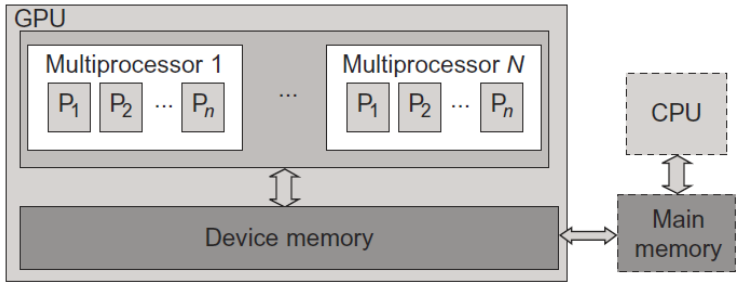


[7] Distributed & Cloud Computing Book

- Multi-core CPU chips have quad, six, or n processing cores on one chip and use cache hierarchies

Advances in Computation - GPGPUs

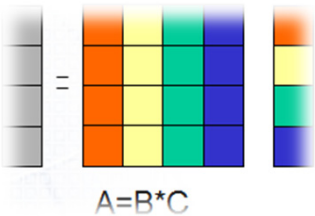
- The graphics Processing Unit (GPU) is repurposed as General-Purpose GPUs (GPGPUs) and used for computing.
- Slower than CPUs but more than makes up for it with sheer volume, i.e. consists of *very many* simple cores
  - High throughput computing-oriented architecture
  - Use massive parallelism by executing a lot of concurrent threads
  - Handle an ever increasing amount of multiple instruction threads
  - CPUs instead typically execute a single long thread as fast as possible
- Simplicity leads to leass power consumption
- Many-core GPUs are already used in large clusters and within massively parallel supercomputers

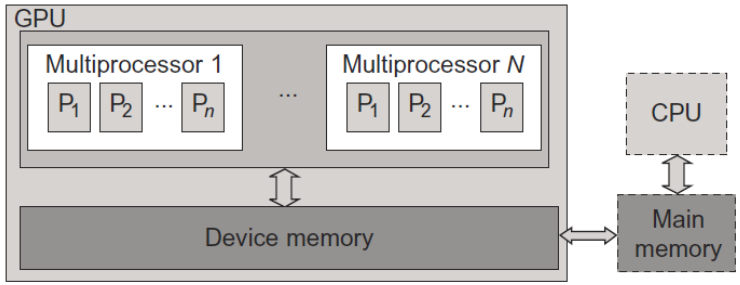


[7] Distributed & Cloud Computing Book

Advances in Computation - GPGPUs

- GPUs accelerate computing thru massive parallelism, with thousands of threads.
- GPUs are designed to compute a large number of floating point operations in parallel
- GPU accelerator architecture example - NVIDIA card
  - GPUs can have **256 cores** on one single GPU chip (NVIDIA TEGRA X1)
  - Each core can work with **eight threads** of instructions
  - GPU is able to concurrently execute 256 * 8 = **2048 threads**
  - Interaction and thus major (bandwidth) bottleneck between CPU and GPU is via memory interactions
  - E.g. applications that use matrix – vector multiplication

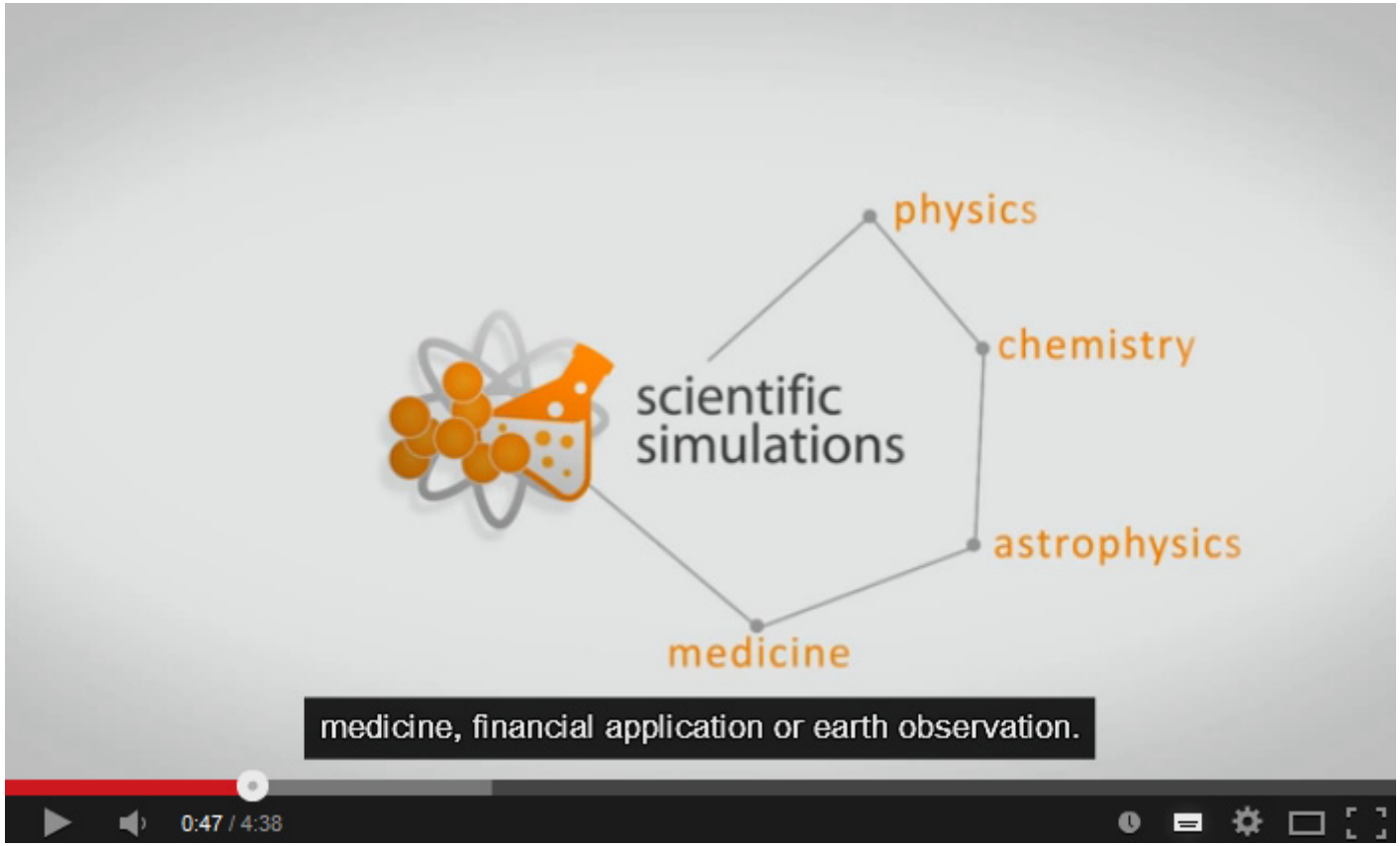




[7] Distributed & Cloud Computing Book

(other well known accelerators & many-core processors are e.g. Intel Xeon Phi → run 'CPU' applications easier)

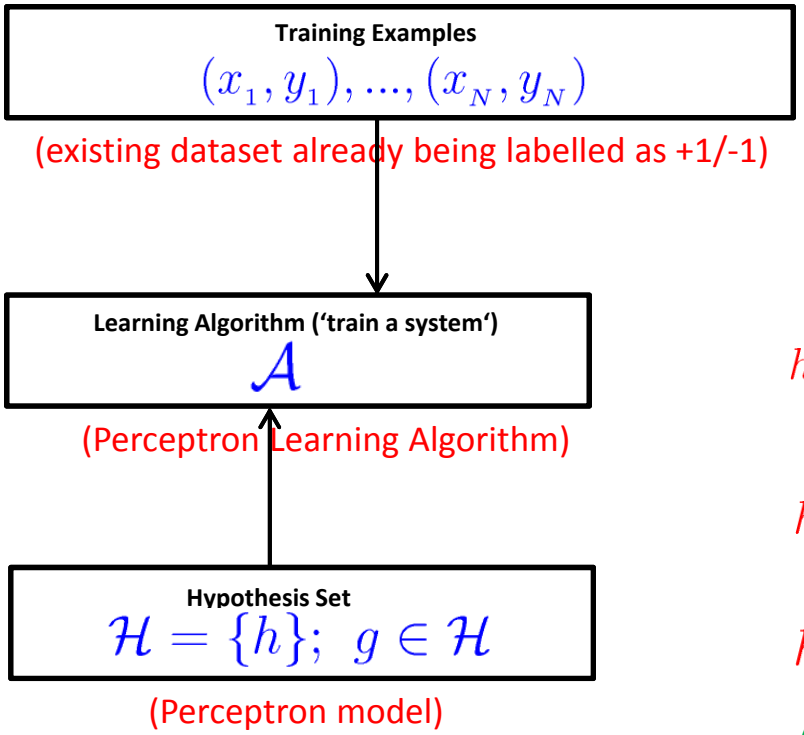
[Video] GPGPUs & Applications

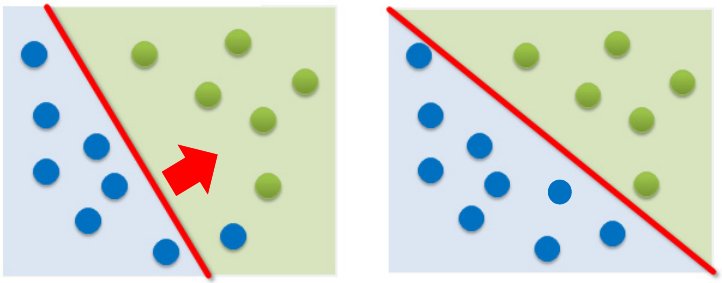


[8] 'HPC – GPGPUs', YouTube Video

Perceptron Learning Algorithm – Revisited

- When: If we believe there is a linear pattern to be detected
  - Assumption: Linearly seperable data (lets the algorithm converge)





[9] Rosenblatt, 1958

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

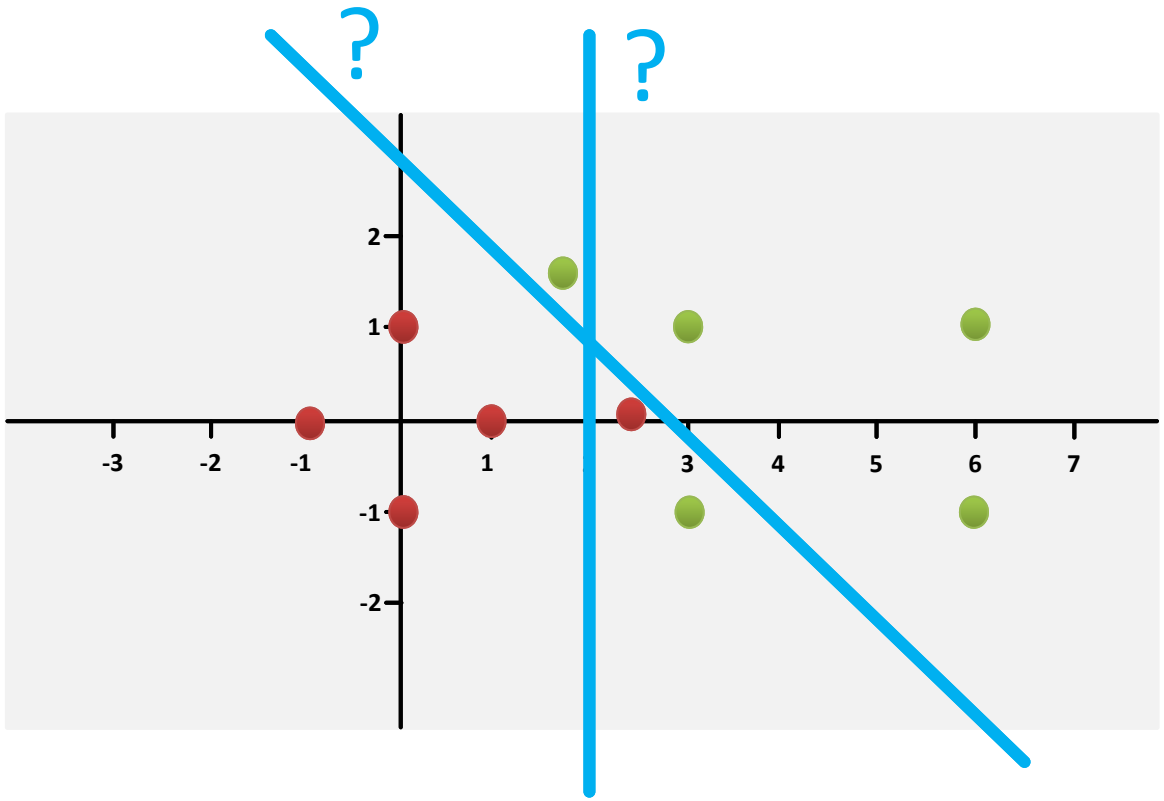
$$h(\mathbf{x}) = sign\left(\left(\sum_{i=1}^{d} w_i x_i\right) + w_0\right); w_0 = -threshold$$

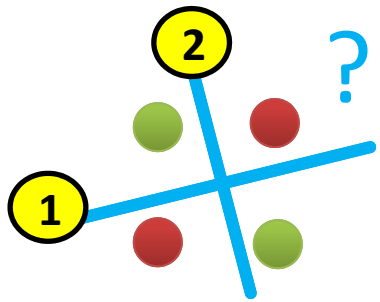
$$h(\mathbf{x}) = sign(\mathbf{w}^T\mathbf{x})$$ **(vector notation, using transpose)**

(transpose = reflecting elements along main diagonal)

Practice: Non-linearly Seperable Data

- More often in practice, requires a ‘soft threshold‘
  - ‘soft-threshold‘ means allowing ‘some errors‘ being ‘overall‘ better

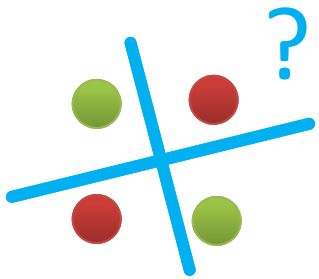




(known also as XOR problem)

Simple Application Example: Limitations of Perceptrons

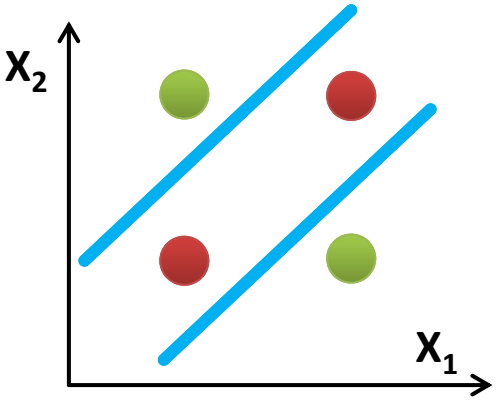
- Simple perceptrons fail: ‘not linearly seperable’



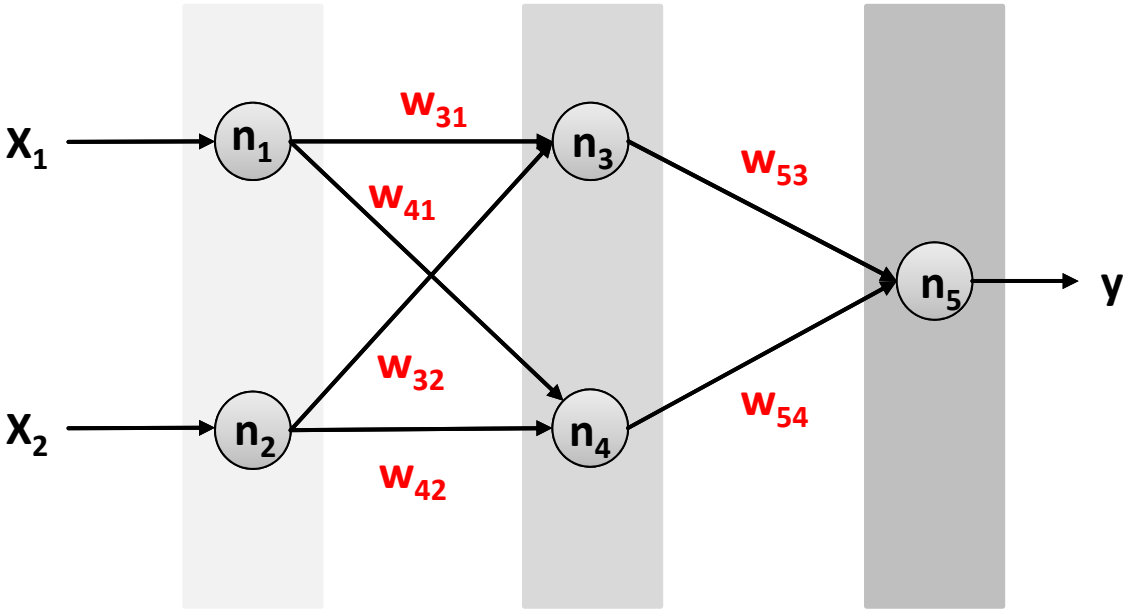
X_1	X_2	Y
0	0	-1
1	0	1
0	1	1
1	1	-1

Labelled Data Table

(Idea: instances can be classified using two lines at once to model XOR)



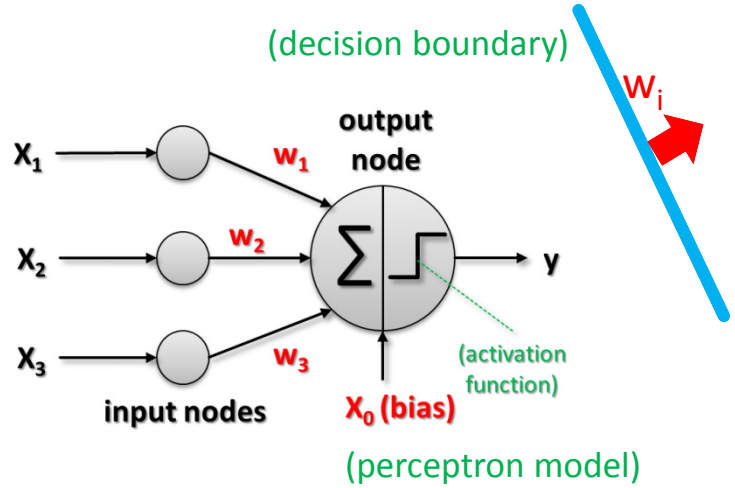
Decision Boundary

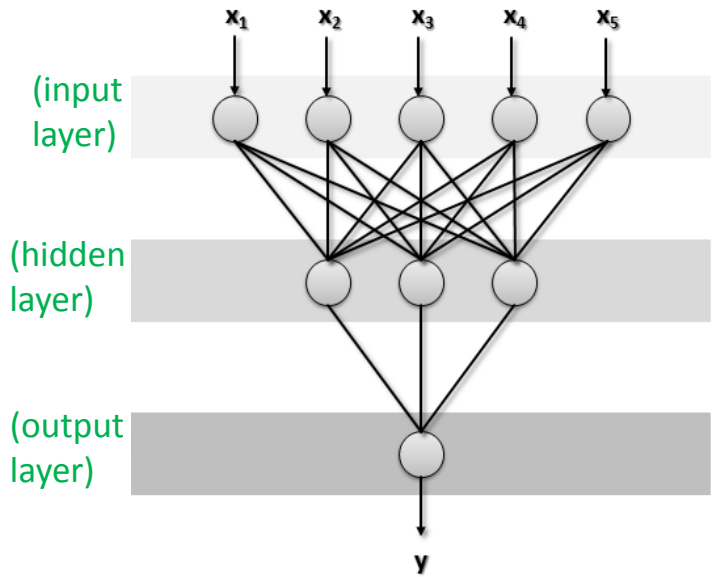


Two-Layer, feed-forward Artificial Neural Network topology

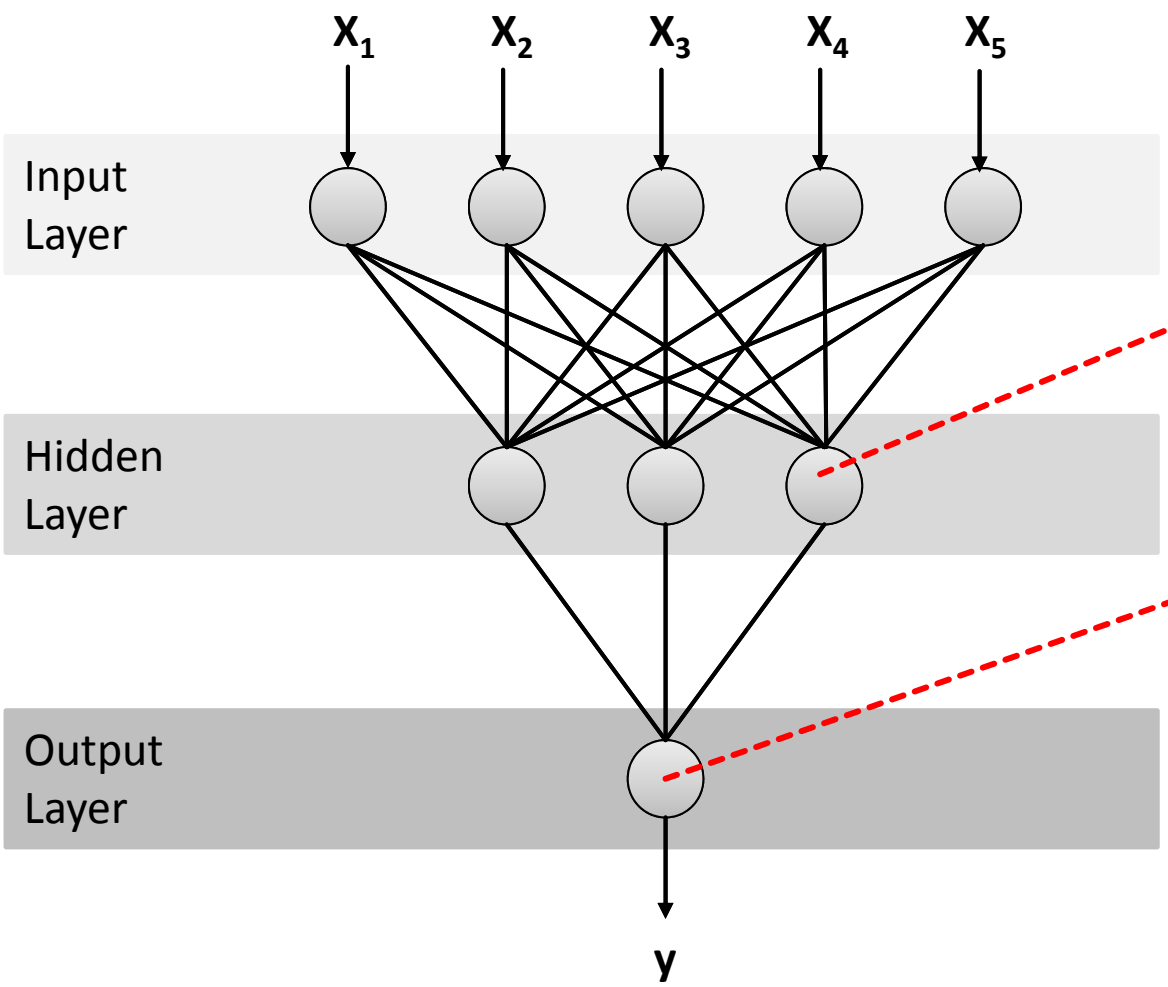
Multi Layer Perceptrons – Artificial Neural Networks

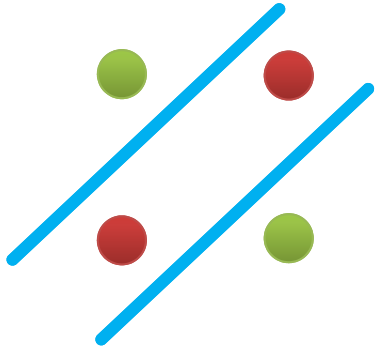
- Key Building Block
  - Perceptron learning model
  - Simplest linear learning model
  - Linearity in learned weights w_i
  - One decision boundary
- Artificial Neural Networks (ANNs)
  - Creating more complex structures
  - Enable the modelling of more complex relationships in the datasets
  - May contain several intermediary layers
  - E.g. 2-4 hidden layers with hidden nodes
  - Use of activation function that can produce output values that are nonlinear in their input parameters





Artificial Neural Networks (ANN) – Layers & Nodes





- **Think each hidden node as a 'simple perceptron' that each creates one hyperplane**

- **Think the output node simply combines the results of all the perceptrons to yield the 'decision boundary' above**

- **Feed-forward neural network: nodes in one layer are connected only to the nodes in the next layer ('a constraint of network construction')**

ANN - Learning Algorithm & Optimization

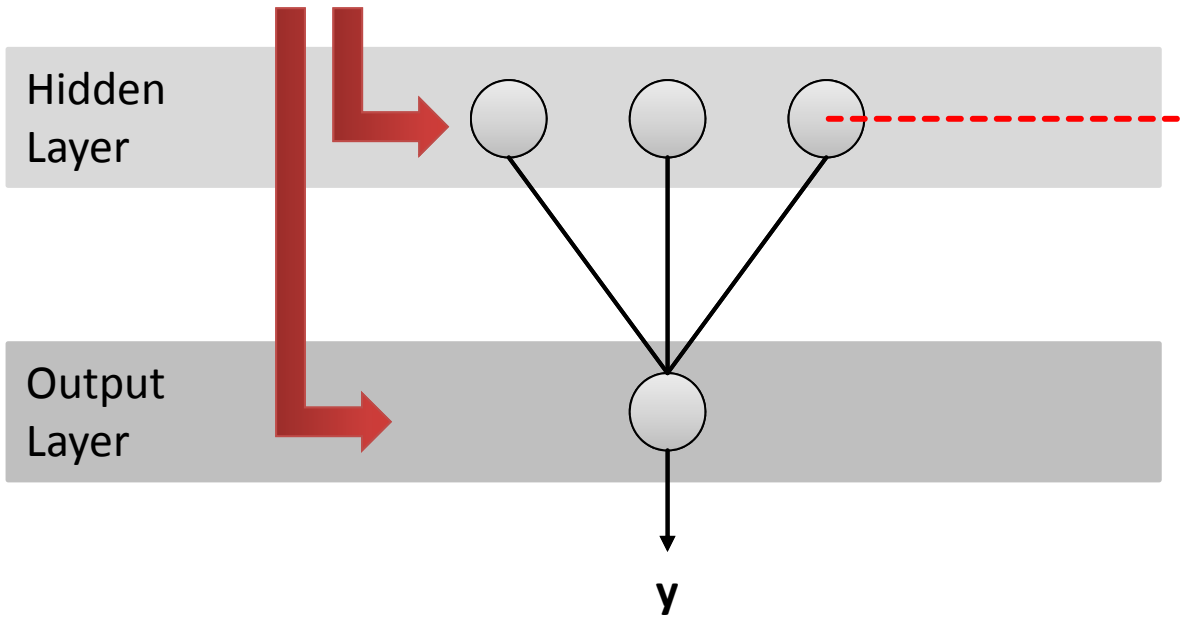
- Determine a set of weights **w** that 'minimize the total sum of squared errors':

y = sign (w · x)

Linear perceptron

Sum of squared errors depend on w, because predicted class y is a 'function of the weights' assigned to the hidden and output nodes

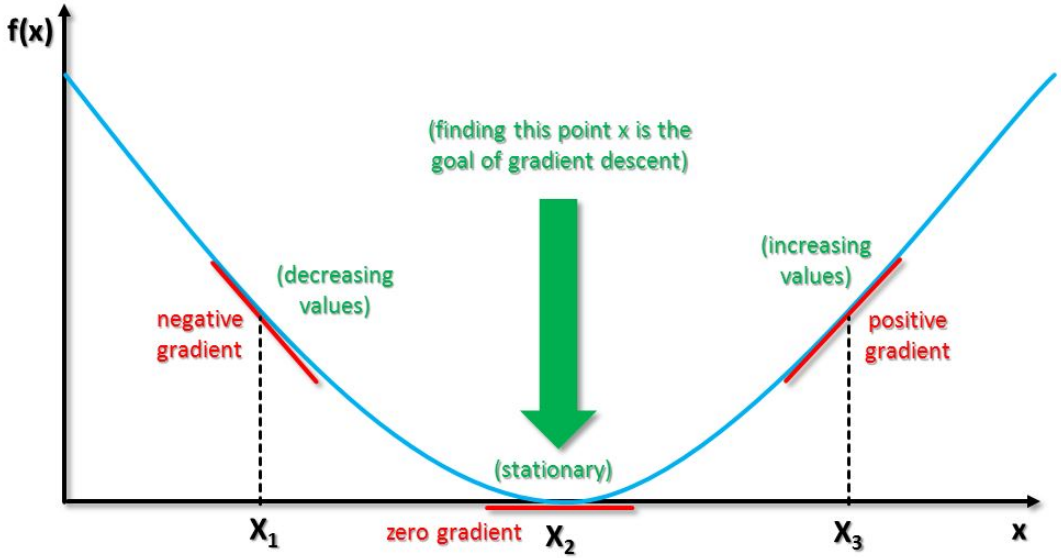
$$E(w) = \frac{1}{2}\sum_{i=1}^{N}(y_i - \acute{y}_i)^2$$



- **Error term, associated with each hidden node**

- **Error function is quadratic in its parameters and a global minimum can be easily found**
- **Other objective / loss functions possible, e.g. categorical cross-entropy**

Gradient Descent Method (1)



(minimization: substract gradient term because we move towards local minima)

(the derivative of f with respect to a)

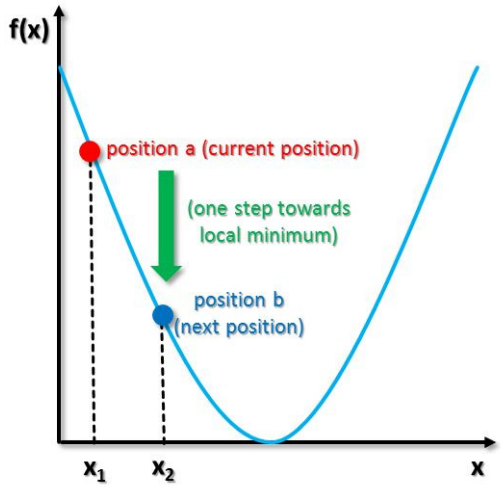
$$\mathbf{b} = \mathbf{a} - \gamma \nabla \mathbf{f}(\mathbf{a})$$

(old position before the step)

(gradient term is steepest ascent)

(new position after the step)

(weighting factor known as step-size, can change at every iteration, also called learning rate)



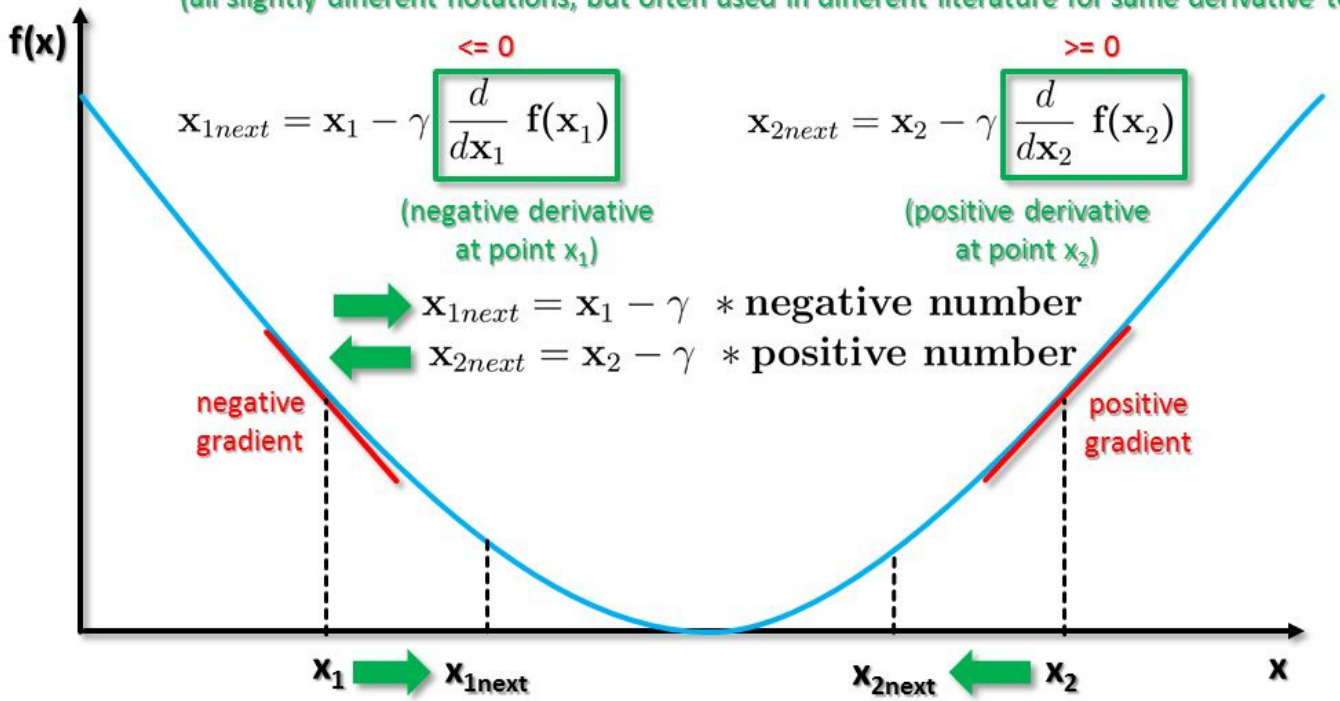
[10] Big Data Tips, Gradient Descent

Gradient Descent Method (2)

- **Gradient Descent (GD) uses all the training samples available for a step within a iteration**
- **Stochastic Gradient Descent (SGD) converges faster: only one training samples used per iteration**

$$\mathbf{b} = \mathbf{a} - \gamma \nabla \mathbf{f}(\mathbf{a}) \quad \mathbf{b} = \mathbf{a} - \gamma \frac{\partial}{\partial \mathbf{a}} \mathbf{f}(\mathbf{a}) \quad \mathbf{b} = \mathbf{a} - \gamma \frac{d}{d\mathbf{a}} \mathbf{f}(\mathbf{a})$$

(all slightly different notations, but often used in different literature for same derivative term)



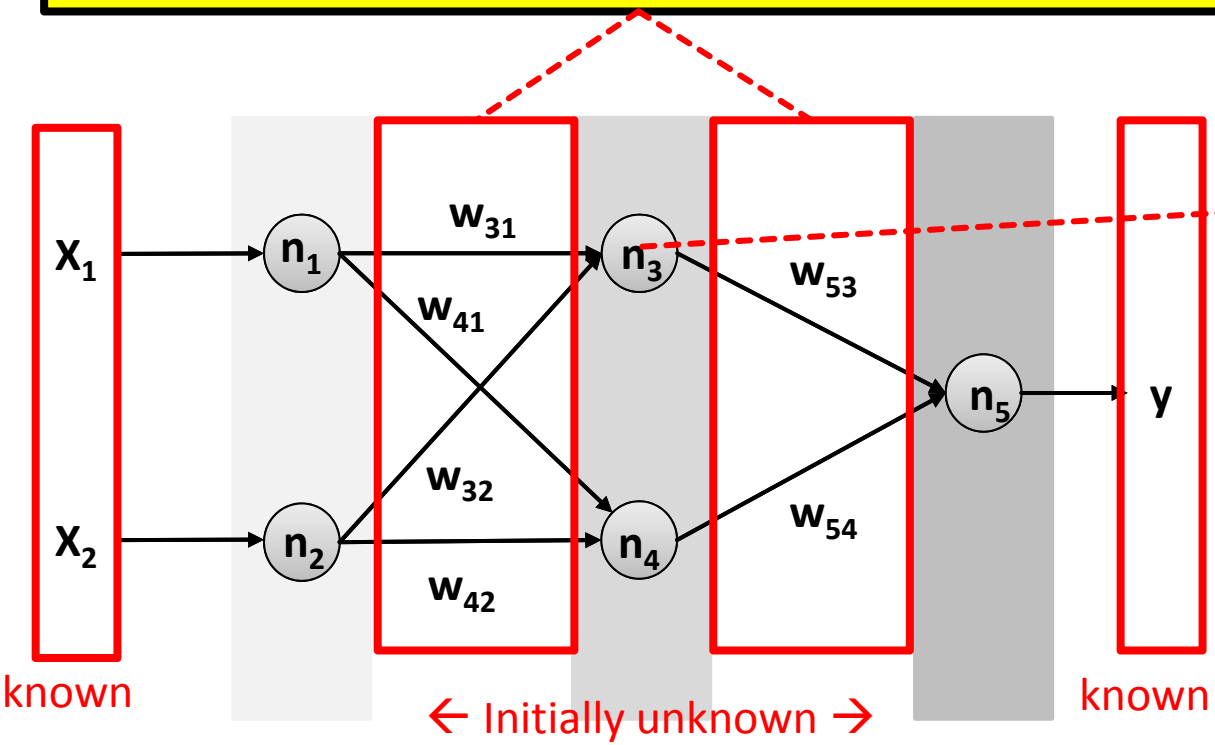
[10] Big Data Tips, Gradient Descent

ANN – Backpropagation Algorithm (BP) Basics

- One of the most widely used algorithms for supervised learning
  - Applicable in multi-layered feed-forward neural networks

- **'Gradient descent method' can be used to learn the weights of the output and hidden nodes of a artificial neural network**

[11] Introduction to Data Mining

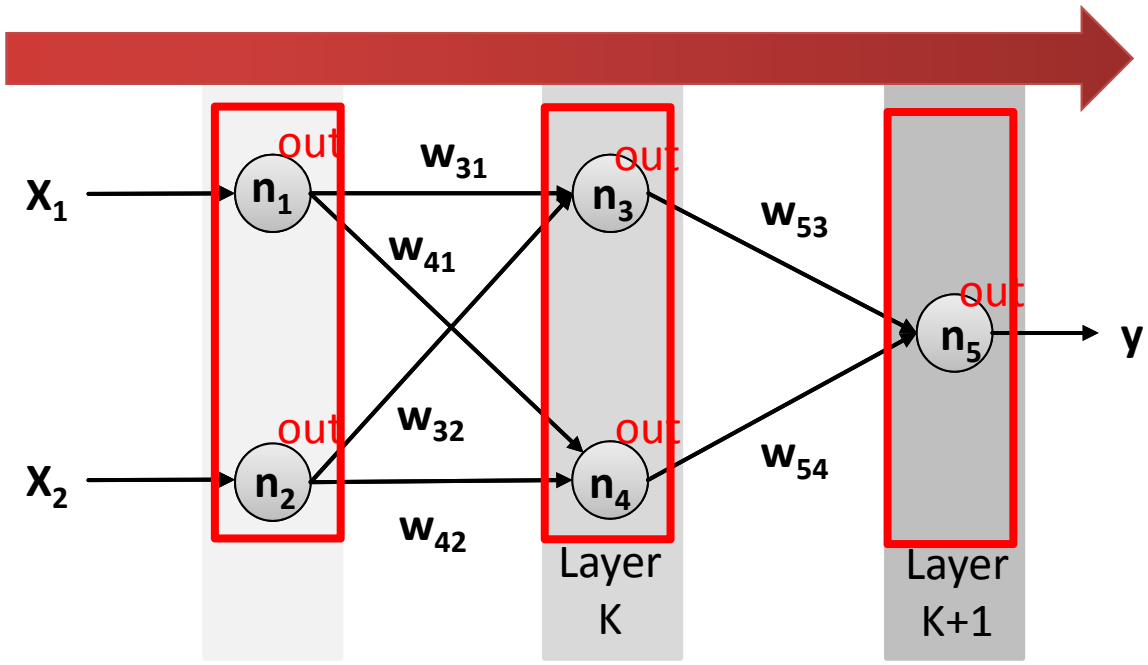


- **Hidden nodes problem: computing error term hard: $\partial E / \partial wj$**
- **Their Output values are unknown to us (here)…**

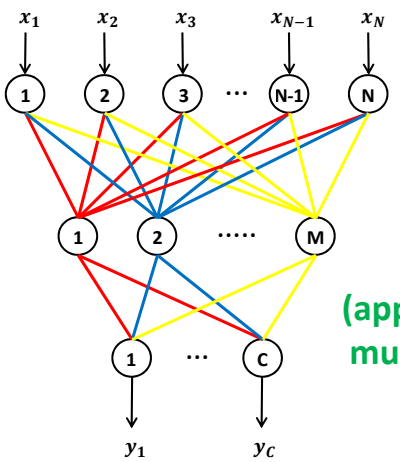
- **The backpropagation algorithm solves exactly this problem with two phases per iteration(!)**

ANN – Backpropagation Algorithm Forward Phase

1. ‘Forward phase (does not change weights, re-use old weights)’:
   - Weights obtained from the previous iteration are used to compute the output value of each neuron in the network (‘initialize weights randomly’)
   - Computation progresses in the ‘forward direction’, i.e. outputs ‘out’ of the neurons at level k are computed prior to level k+1



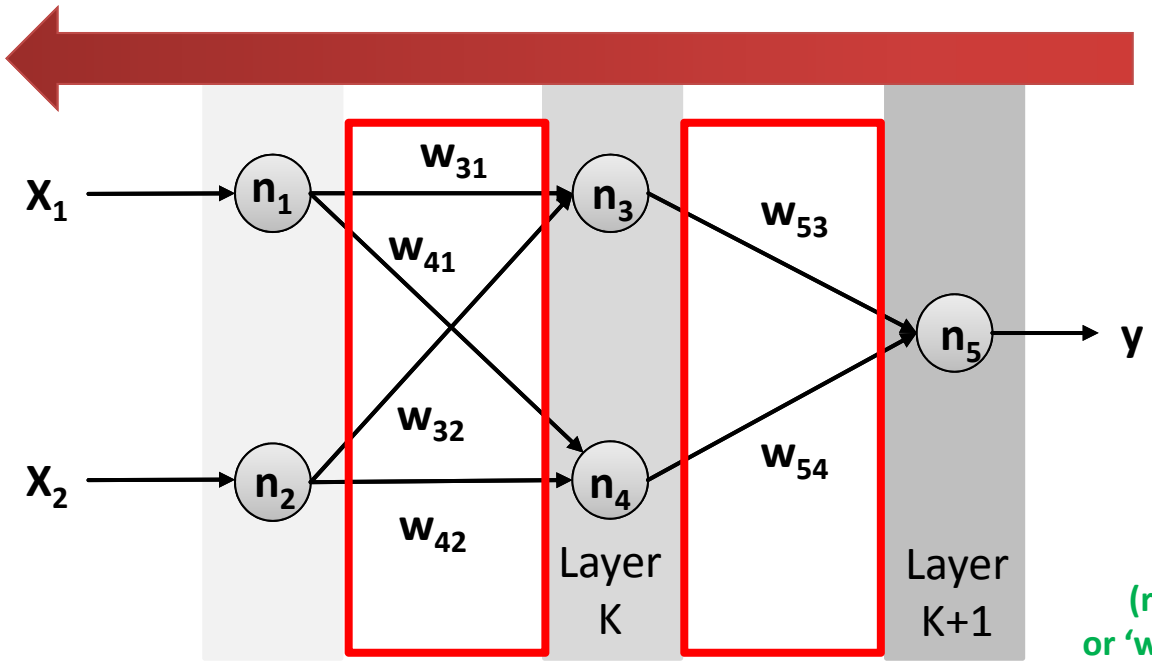
- **Use corresponding ‘activation function’ but with ‘old weights’**



(applicable for multiclass to)

ANN – Backpropagation Algorithm Backward Phase

2. ‘Backward phase (‘learning’ → change the weights in the ANN)’:
   - Weight update formula is applied in the ‘reverse direction’
   - Weights at level K + 1 are updated before the weights at level k
   - Idea: use the errors for neurons at layer k + 1 to estimate errors for neurons at layer k



$$w_j <- wj - \lambda \frac{\partial E(w)}{\partial w_j}$$

weight update formula
of the ‘gradient descent method’

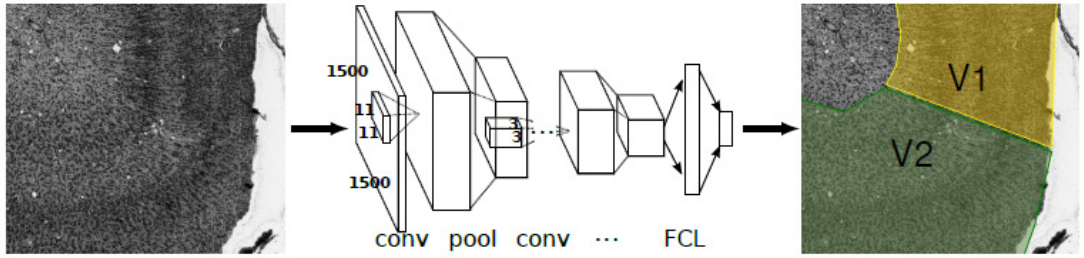
Now that can compute
the error one-by-one

$$E_{in}(\mathbf{w}) + \frac{\lambda}{N}\mathbf{w}^T\mathbf{w}$$

(regularization method ‘weight decay‘
or ‘weight drop‘ is used in neural networks‘)

Deep Learning Architectures

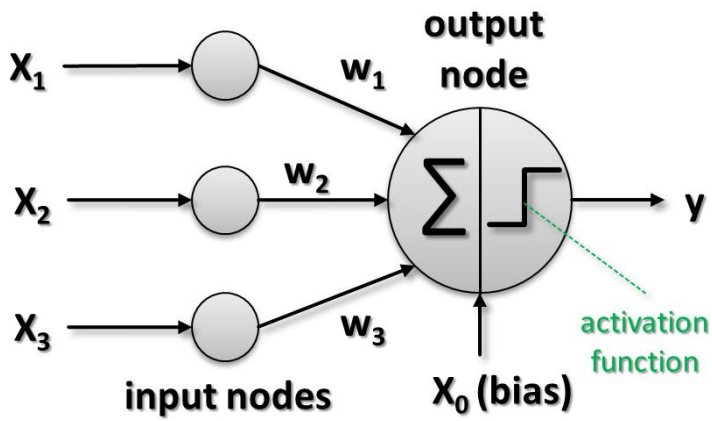
- Deep Neural Network (DNN)
  - ‘Shallow ANN‘ approach with many hidden layers between input/output
- Convolutional Neural Network (CNN, sometimes ConvNet)
  - Connectivity pattern between neurons replicating the visual cortex



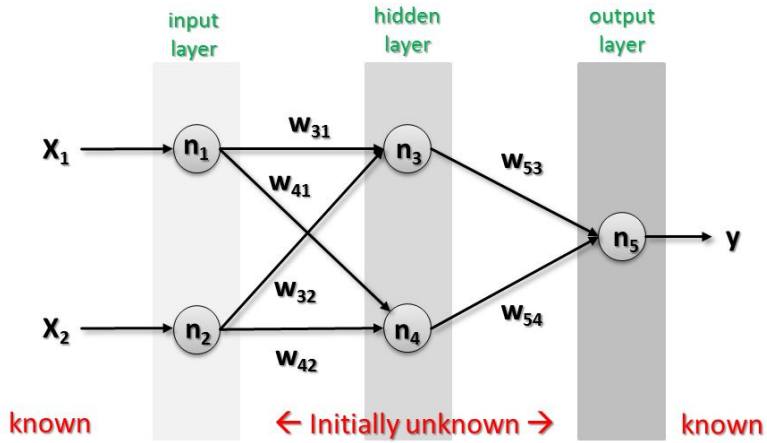
- Deep Belief Network (DBN)
  - Composed of multiple layers of variables; only connections between layers
- Recurrent Neural Network (RNN)
  - ‘ANN‘ but connections form a directed cycle; state and temporal behaviour

Artificial Neural Network – Feature Engineering & Layers

- Approach: Prepare data before
  - Classical Machine Learning
  - Feature engineering (e.g. SDAP)
  - Dimensionality reduction techniques (e.g. DAFE: smaller, better data)
  - Low number of layers (many layers computationally infeasible in the past)
  - Very succesful for speech recognitition ('state-of-the-art in your phone')



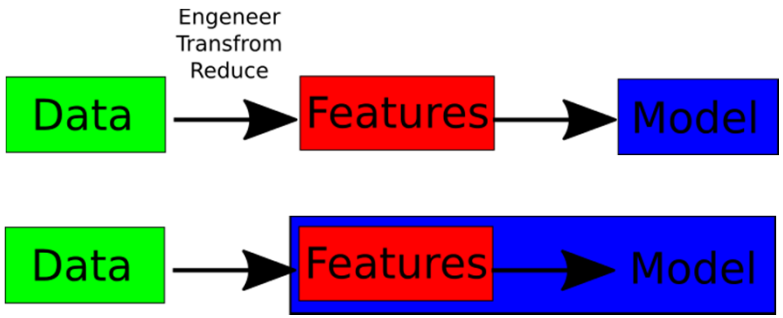
(Perceptron model: designed after human brain neuron)



(Artificial neural network two layer feed – forward)

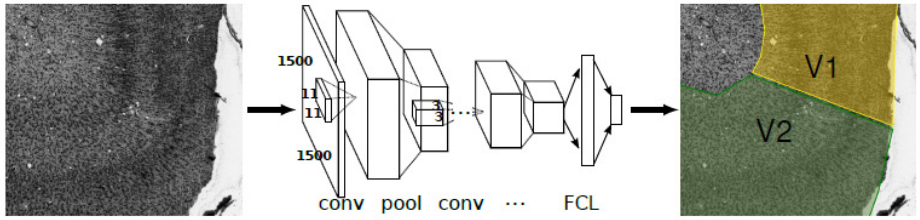
Deep Learning – Feature Learning & More Smart Layers

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

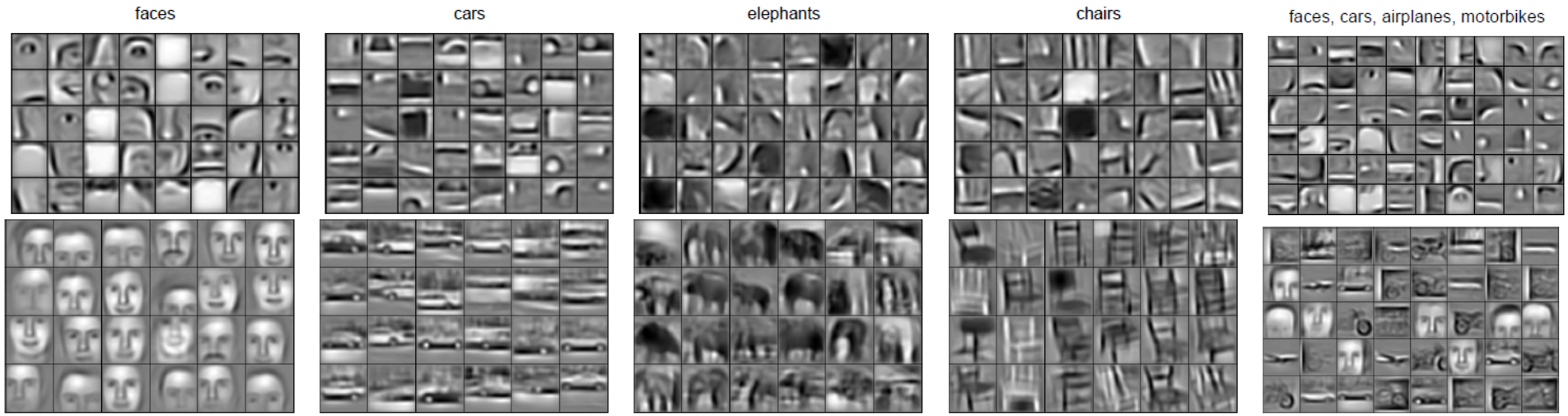


  - Very succesful for image recognition and other emerging areas
  - Assumption: data was generated by the interactions of many different factors on different levels (i.e. form a hierarchical representation)
  - Organize factors into multiple levels, corresponding to different levels of abstraction or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

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



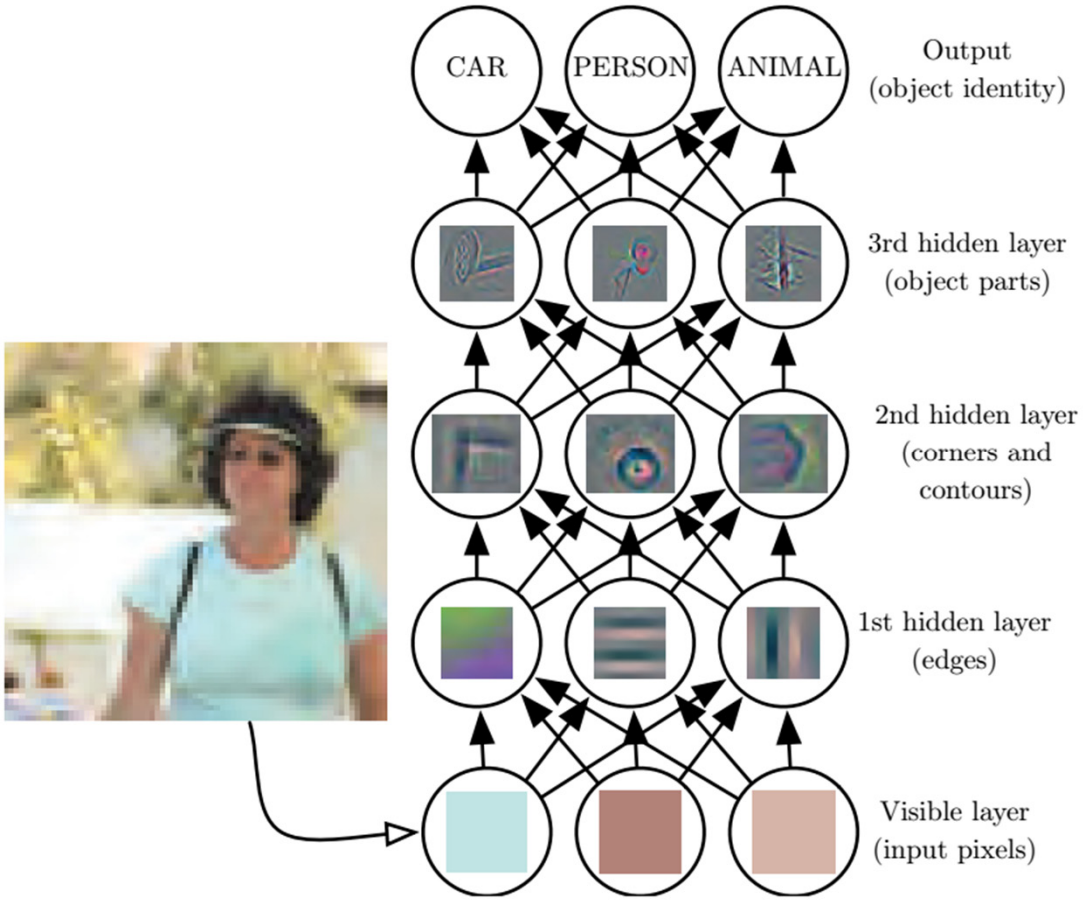
Deep Learning – Feature Learning Benefits



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

- **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**

Deep Learning – Feature Learning Benefits



[12] Ian Goodfellow, Yoshua Bengio, and Aaron Courville 'Deep Learning'

Deep Learning – Tools of the Trade

Caffe

TensorFlow

Keras

theano

Caffe2

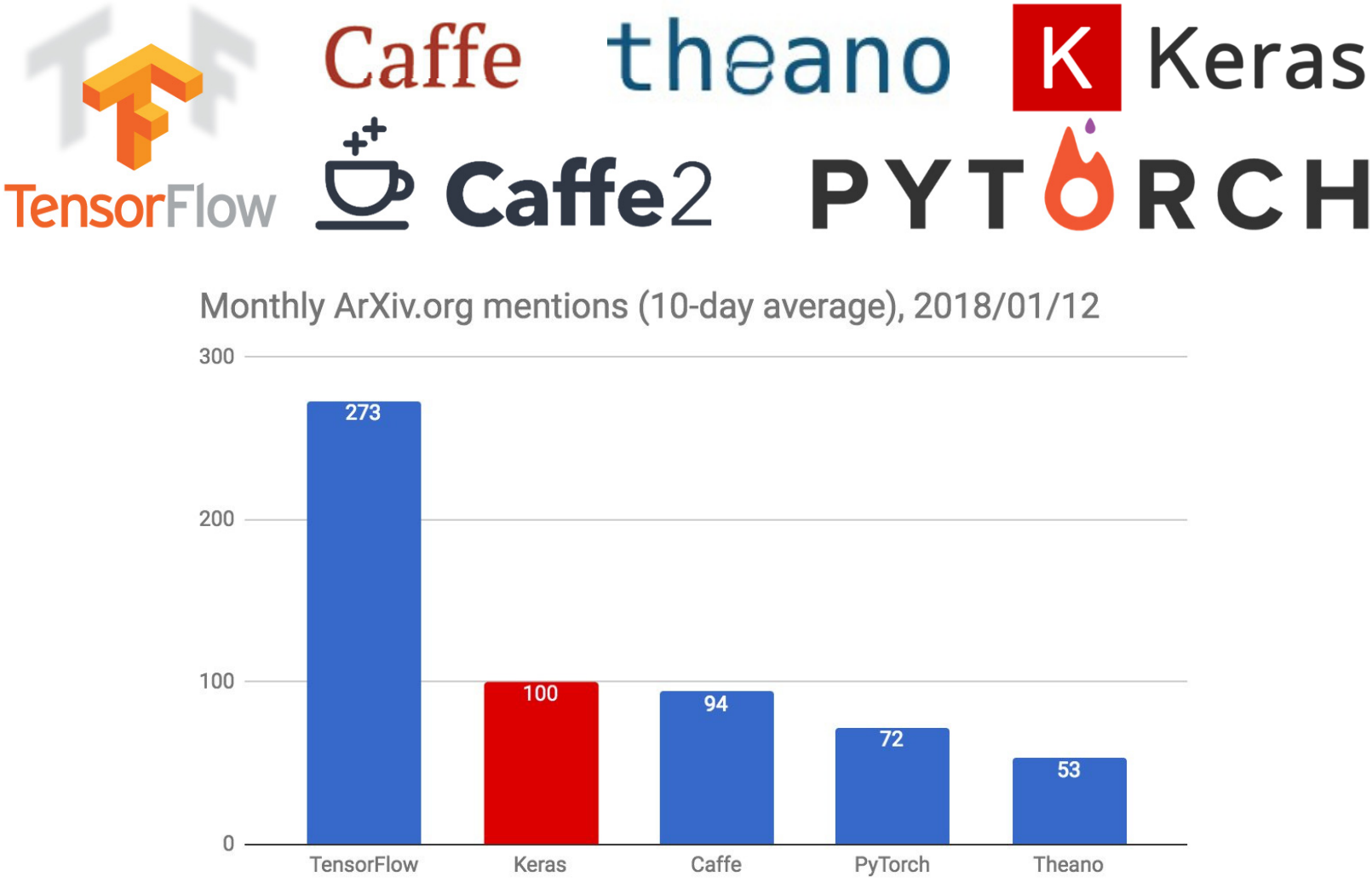
PYTORCH

Deep Learning – Tools of the Trade

TensorFlow Caffe theano Keras Caffe2 PYTORCH

- The top 5 mentions on arXiv.org, many more exist
- Free and Open Source (FOSS) frameworks, libraries and extensions
- Mostly used with Python, a major contributer to its growing popularity.
- Initiated and maintained by a mixture of both academia and industry

Deep Learning – Tools of the Trade



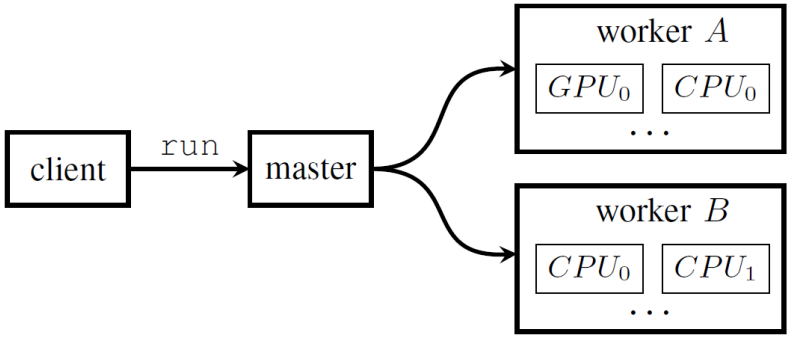
Keras with Tensorflow Backend – GPU Support

- 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 *[13] 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

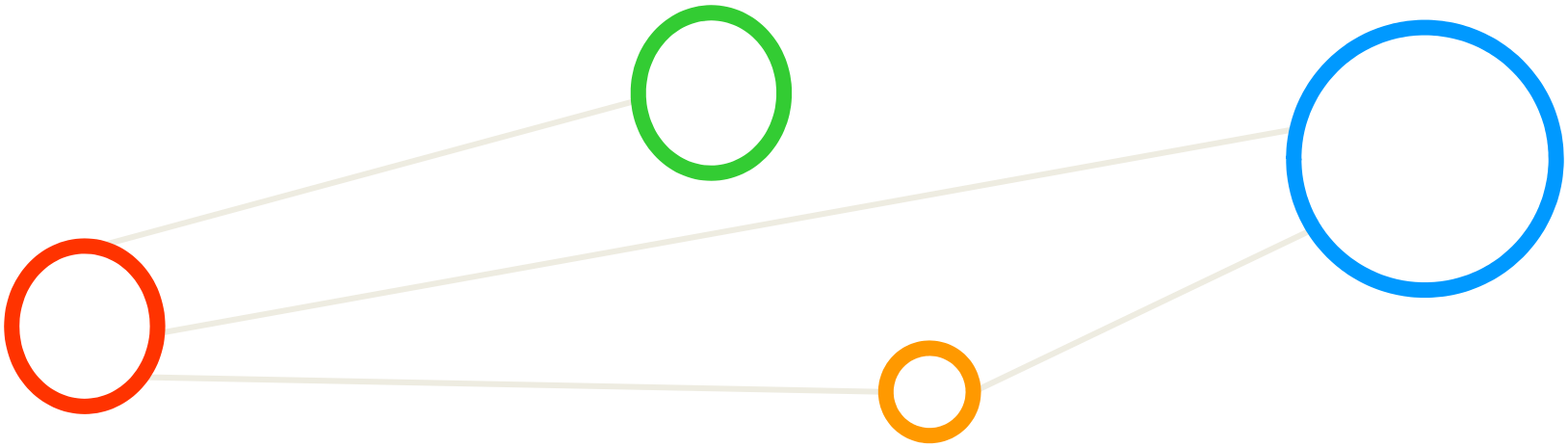
[14] Tensorflow Deep Learning Framework



[15] A Tour of Tensorflow

TensorFlow

Opportunities in Machine Learning



GPU Hackathon

About

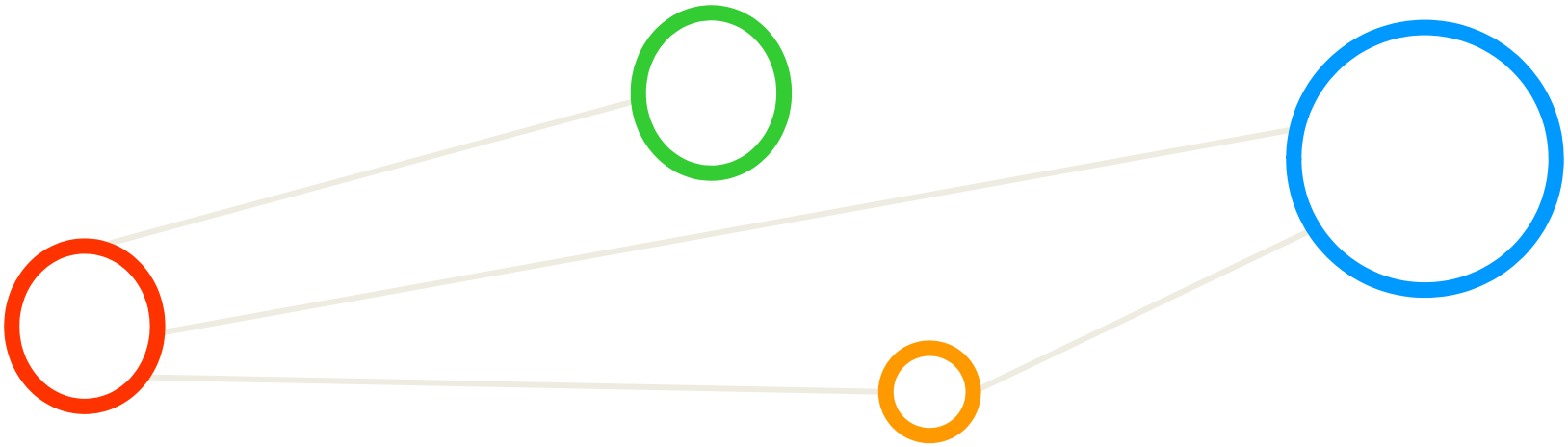
- Bring your scientific application (team of 3–6 people)
- 2 mentors per team from academia, NVIDIA, PGI, IBM, …
- Work 5 days intensively
- Optimize, port application
  - Port: Take first steps to GPU-acceleration
  - Optimize: Further accelerate GPU-enabled application
  - New: **Deep Learning**
- CUDA, OpenACC; C/C++, Fortran, Python, …
- **No GPU experience required!**
- **Free!**

GPU Hackathon

Dresden

- 5–9 Mar 2018 @ TU Dresden
- Application deadline: 19 Jan++
- → https://www.olcf.ornl.gov/training-event/2018-gpu-hackathons/
- European hackathon 2017 in Jülich
  - Great atmosphere, many lines coded, lots of jobs submitted
  - All applications accelerated or plan formulated

Lecture Bibliography



Lecture Bibliography (1)

- [1] YouTube Video, 'The Deep Learning Revolution',
  Online: https://www.youtube.com/watch?v=Dy0hJWltsyE
- [2] 'Cybernetic Zoo' web page,
  Online: http://cyberneticzoo.com/cyberneticanimals/1958-cyber-the-dog-a-h-bruinsma-netherlands/
- [3] YouTube Video, 'Neural Networks, A Simple Explanation',
  Online: http://www.youtube.com/watch?v=gcK_5x2KsLA
- [4] 'Our World in Data' web page,
  Online: https://ourworldindata.org/internet
- [5] ImageNet Web page,
  Online: http://image-net.org
- [6] J. Dean et al., 'Large scale deep learning', Keynote GPU Technical Conference, 2015
- [7] 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
- [8] 'HPC – Get a low-cost supercomputer by unleashing the power of GPUs',
  Online: https://www.youtube.com/watch?v=HYlWxPeL9-k
- [9] Rosenblatt,'The Perceptron: A probabilistic model for information storage and orgainzation in the brain', Psychological Review 65(6), pp. 386-408, 1958
- [10] Big Data Tips, 'Gradient Descent',
  Online: http://www.big-data.tips/gradient-descent
- [11] 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

Lecture Bibliography (2)

- [12] Ian Goodfellow, Yoshua Bengio, and Aaron Courville, 'Deep learning', The MIT Press, 2016
- [13] Keras Python Deep Learning Library,
  Online: https://keras.io/
- [14] Tensorflow Deep Learning Framework,
  Online: https://www.tensorflow.org/
- [15] A Tour of Tensorflow,
  Online: https://arxiv.org/pdf/1610.01178.pdf
- [16] An Introduction to Statistical Learning with Applications in R,
  Online: http://www-bcf.usc.edu/~gareth/ISL/index.html
- [17] 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.

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