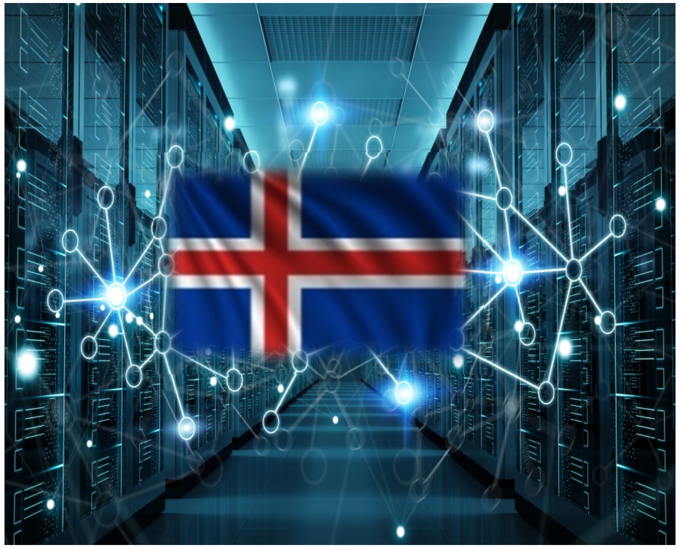



# Introduction to HPC Applications, Systems, Programming Models & Machine Learning & Data Analytics – Part Two

PROF. DR. – ING. MORRIS RIEDEL
UNIVERSITY OF ICELAND – EUROHPC JU GOVERNING BOARD MEMBER ICELAND & JUELICH SUPERCOMPUTING CENTRE (GERMANY)

29TH MAY 2022, INTERNATIONAL SUPERCOMPUTING CONFERENCE, CONGRESS CENTRE, HAMBURG

@ProfDrMorrisRiedel @Morris Riedel @MorrisRiedel @MorrisRiedel https://www.youtube.com/channel/UCWC4VKHmL4NZgFfKoHtANKg

IHPC National Competence Center for HPC & AI in Iceland

EuroHPC Joint Undertaking · EOSC NORDIC · RAISE Center of Excellence · ADMIRE · UNIVERSITY OF ICELAND · EUPEX European Pilot for Exascale · EURO · HELMHOLTZ AI ARTIFICIAL INTELLIGENCE COOPERATION UNIT · DEEP Projects · JÜLICH Forschungszentrum · JÜLICH SUPERCOMPUTING CENTRE

# Introduction to HPC Applications, Systems, Programming Models – Part One

Dr. – Ing.
Bernd Mohr

JÜLICH Forschungszentrum | JÜLICH SUPERCOMPUTING CENTRE







@ Supercomputing 2017, Denver, USA:
Bernd Mohr first EU Chair of SC series

***Importance of our Community Building:***

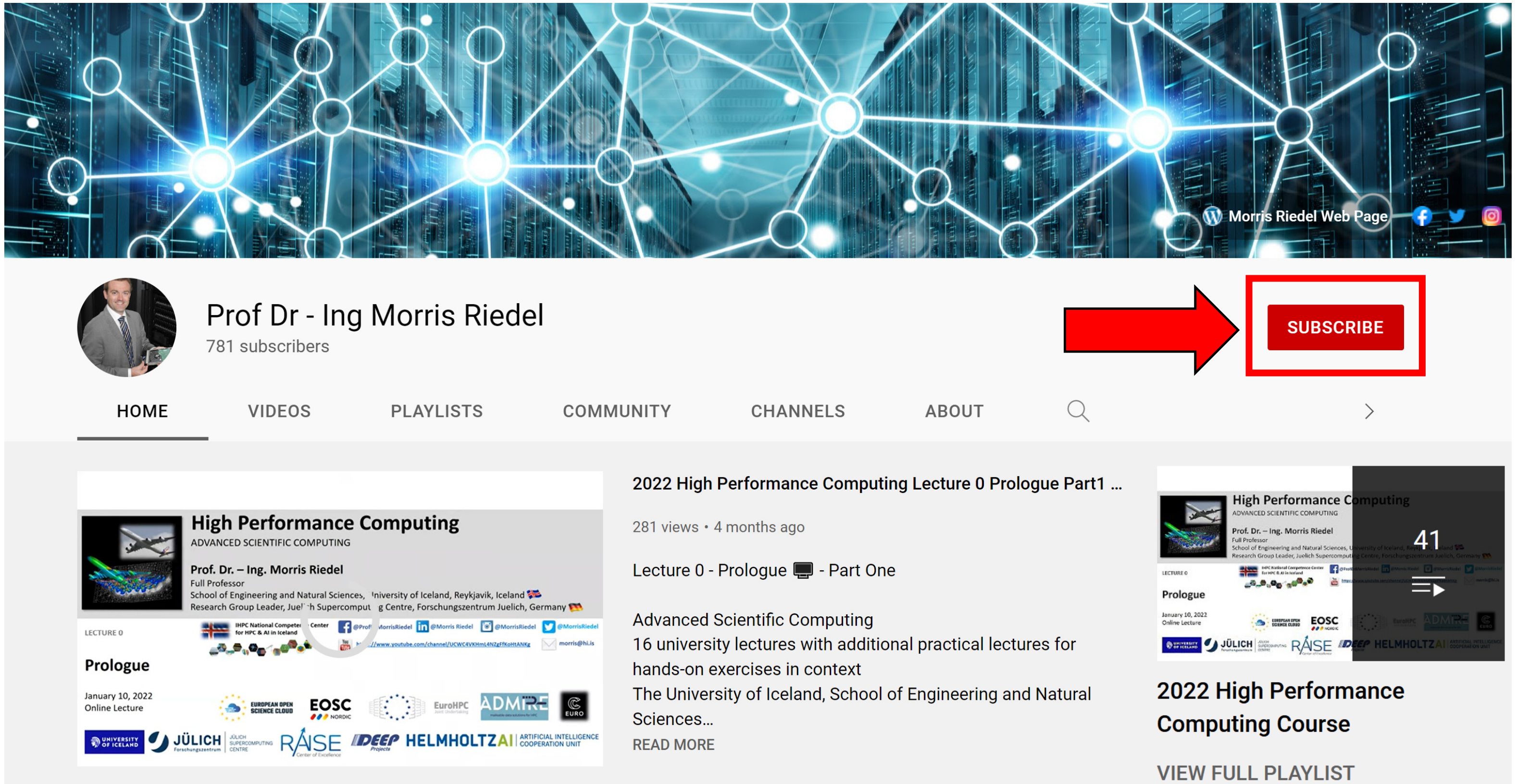
***Attending & Participating at our community conferences like SC @USA or ISC @ Europe is crucial for learning, networking, sharing and building your career over time – world-wide experts become mentors & friends!***

# Outline – Part Two

- All HPC scripts will be available with ISC Online Material after the event
- All source & data free to use

- Machine Learning Fundamentals
  - Learning Methods Overview, Prerequisites, Classification Application & Linear Perceptron Model
  - Training & Testing Process using different Datasets, Food Inspection Classification Application Example
  - Linear Regression Model & Logistic Regression Model
- Artificial Neural Network (ANN) Basics
  - Handwritten Character Recognition MNIST Dataset & Understanding Multi-Class Classification Approach
  - Limits of the Perceptron Learning Model, Multi-Output Perceptron Model & ANNs with Backpropagation
  - Observe Growth of Trainable Parameter & Understanding Overfitting
- Convolutional Neural Network (CNN) Basics
  - Moving from Shallow Learning to Deep Learning, MNIST Application Example with CNNs in Keras
  - Understanding Feature Maps in CNN Architecture, Hyperparameter Complexity & Adam Optimizer
  - Understanding Accuracy Improvements & Limits
- Selected Parallel & Scalable Machine & Deep Learning Techniques
  - piSVM MPI Implementation & Remote Sensing Applications
  - Computing Footprint in Training, Testing & Validation Methods, Distributed Training
  - Parallel & Scalable HPDBSCAN for Data Clustering & Emerging Quantum Machine Learning

# More Information: Full HPC Spring 2022 University Course




***Selected Testimonials from our community:***

*'… I have found your wonderful HPC lectures on YouTube, and I am finding them very helpful for learning the supercomputing tools necessary for my research….'*

**- student in computational nuclear physics from USA**

neeraj kumar • 1 year ago
Awesome work for HPC students....thanks

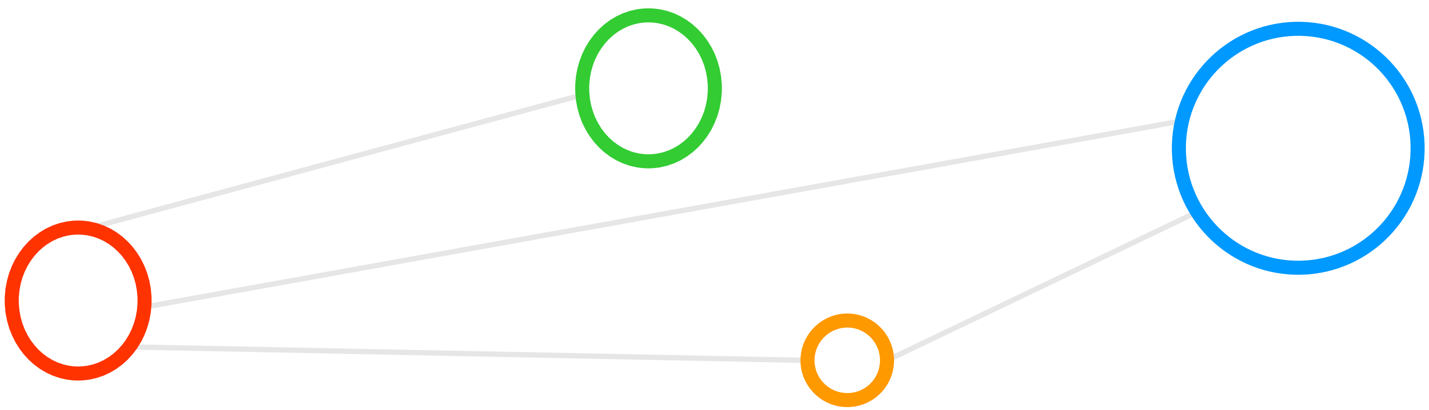
纵横10分钟 - 10 Minutes Across the World • 3 months ago 5 subscribers
Thanks so much for the lecture - One of the best courses. Very helpful!

**Andy Gill** @_andy_gill · Nov 15, 2021
Highly recommend this excellent introduction to state of the art HPC programming by @MorrisRiedel. Enough details to actually be useful, explained clearly using many examples. Thank you for posting it!

youtube.com
2021 High Performance Computing Course
High Performance Computing Course - Advanced Scientific Computing 16 university lectures with ...

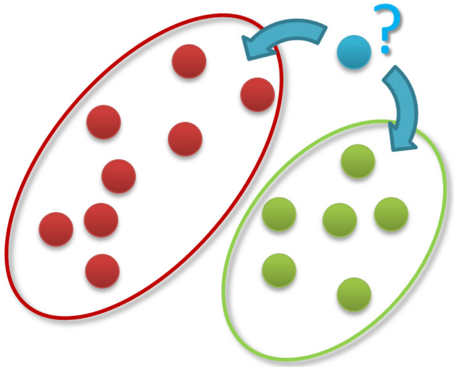
➢ [https://www.youtube.com/channel/UCWC4VKHmL4NZgFfKoHtANKg](https://www.youtube.com/channel/UCWC4VKHmL4NZgFfKoHtANKg)

# Machine Learning Fundamentals



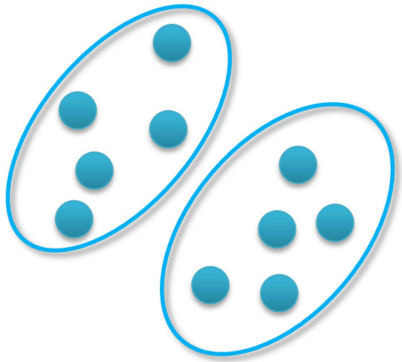
# Machine Learning Models – Short Overview & Introduction to Classification

## Classification



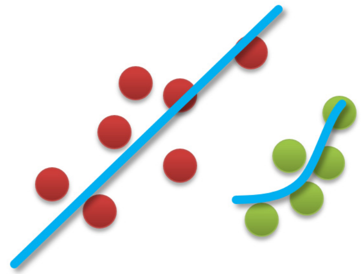

- Groups of data exist
- New data classified to existing groups

## 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

- **Machine learning methods can be roughly categorized in classification, clustering, or regression augmented with various techniques for data exploration, selection, or reduction – despite the momentum of deep learning, traditional machine learning algorithms are still widely relevant today**

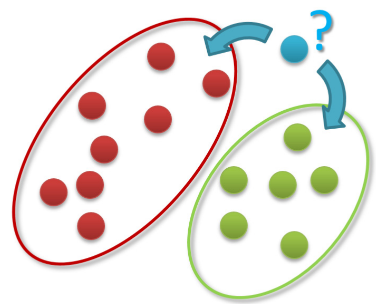
*[1] www.big-data.tips, 'Data Classification'*

# Classification Machine Learning Model – Supervised Learning Example

- Each observation of the predictor measurement(s) has an associated response measurement:
  - Input $\mathbf{x} = x_1, ..., x_d$
  - Output $y_i, i = 1, .., n$
  - Data $(\mathbf{x}_1, y_1), ..., (\mathbf{x}_N, y_N)$
  - (the output guides the learning process as a 'supervisor')

- 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

**Classification**



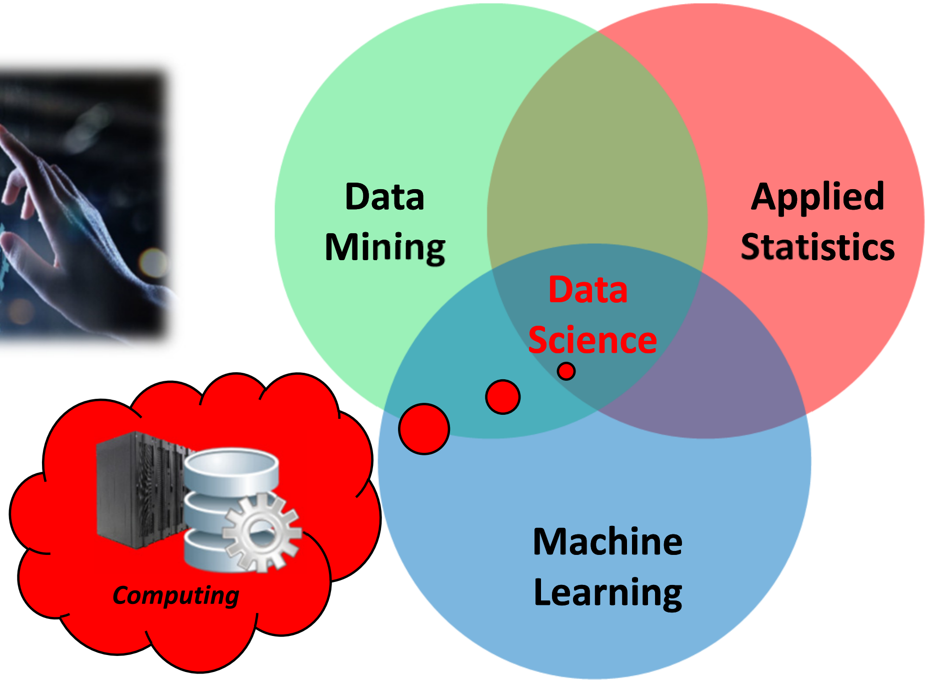

$(\mathbf{x}_1, y_1), ..., (\mathbf{x}_N, y_N)$

- **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]**

# Machine Learning Prerequisites & Computing Challenges – Revisited

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

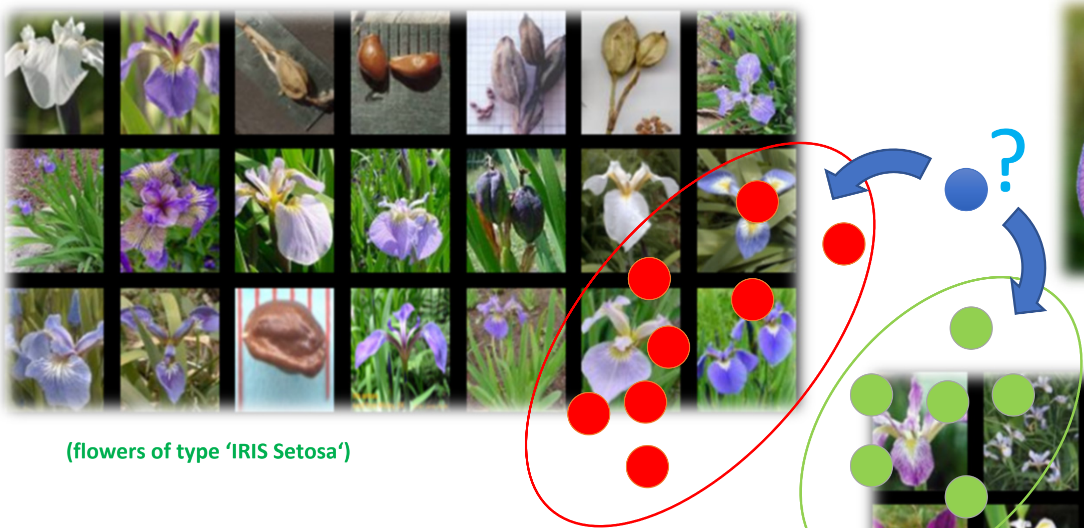
- Idea 'Learning from Big Data'
  - Shared with a wide variety of other disciplines
  - E.g. signal processing, big data data mining, etc.
- Challenges
  - Data is often complex
  - Requires 'Big Data analytics'
  - Learning from data requires processing time → Clouds or High Performance Comping


- Machine learning is a very broad subject and goes from very abstract theory to extreme practice ('rules of thumb')
- Training machine learning models needs processing time (clouds or high performance computing)
- While data analysis is more describing the process of analysin the data, the term data analytics also includes and the necessary scalable or parallel infrastructure to perform analysis of 'big data'

# Simple Application Example: Classification of a Flower

(new data: what type of flower is this?)




(flowers of type ‘IRIS Setosa‘)

(flowers of type ‘IRIS Virginica‘)

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

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

# The Learning Problem in the Example

(flowers of type 'IRIS Setosa')

(flowers of type 'IRIS Virginica')

*[2] 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

*[3] UCI Machine Learning Repository Iris Dataset*


*[4] Image source: Wikipedia, Sepal*

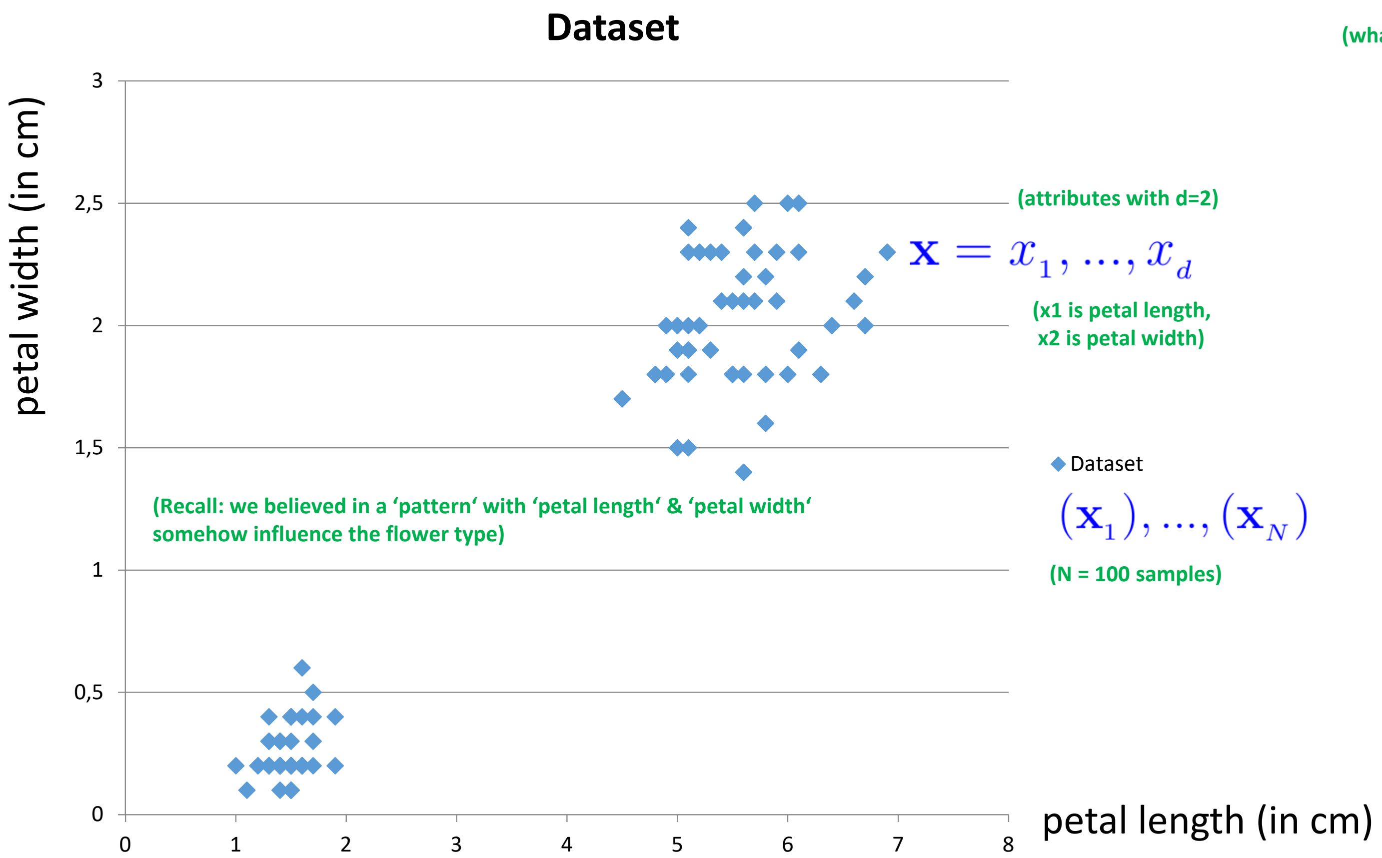
(four data attributes for each sample in the dataset)

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

(one class label for each sample in the dataset)

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

# Check Preparation Phase: Plotting the Data (Two Classes)




(what about the class labels?)

(attributes with d=2)

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

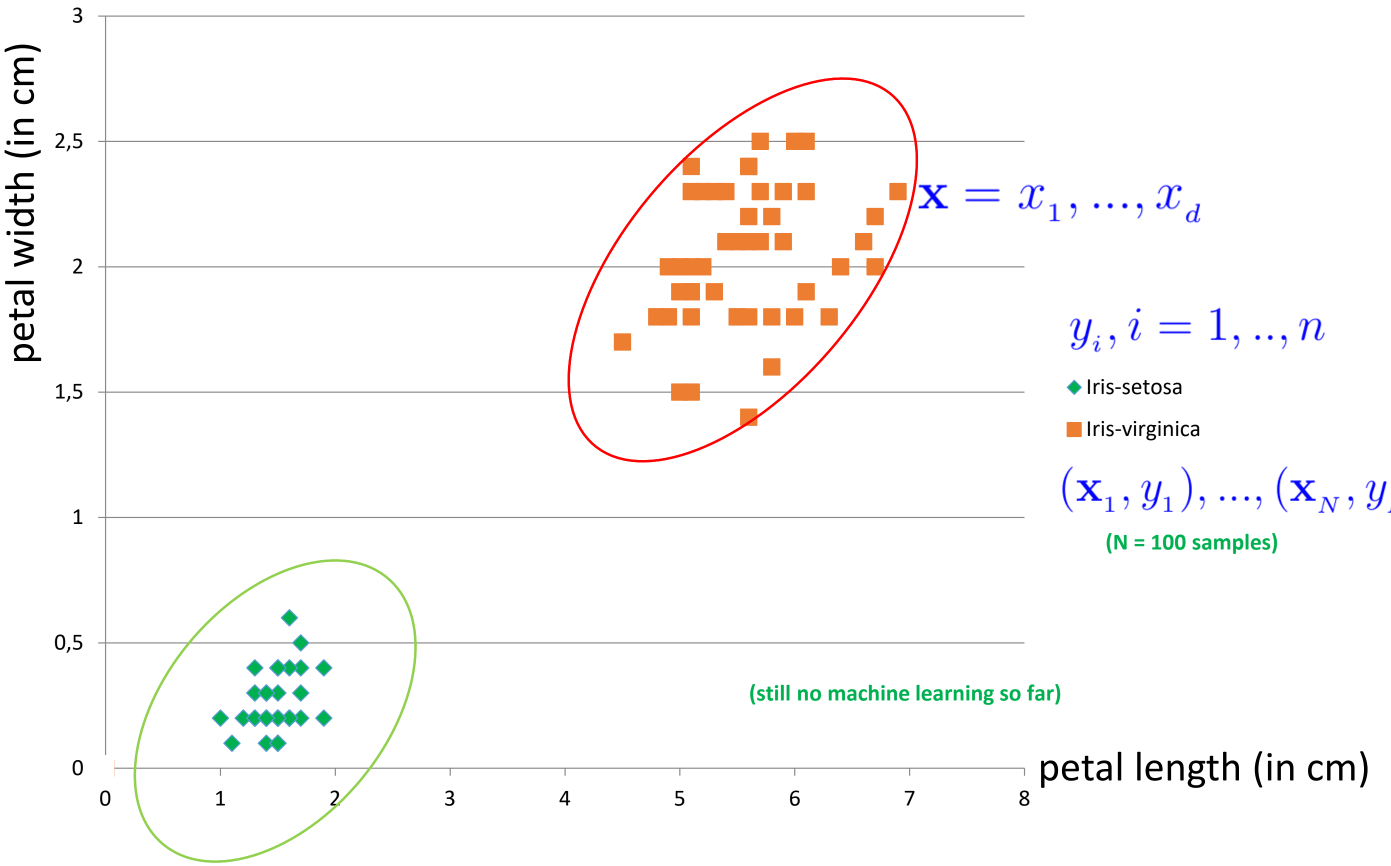
(x1 is petal length, x2 is petal width)

(Recall: we believed in a 'pattern' with 'petal length' & 'petal width' somehow influence the flower type)

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

(N = 100 samples)

# Check Preparation Phase: Class Labels




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

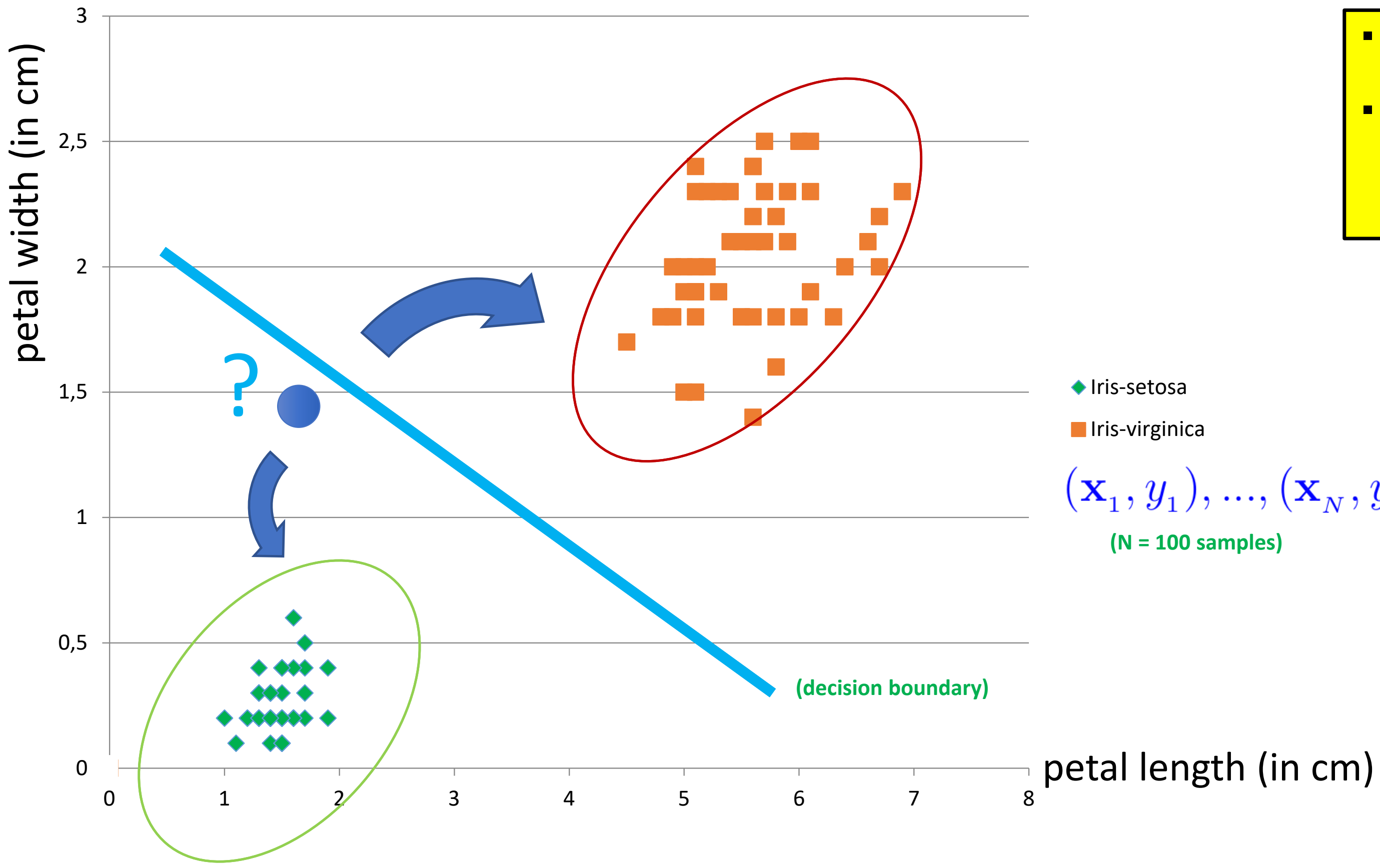
$y_i, i = 1, .., n$

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

**(N = 100 samples)**

**(still no machine learning so far)**

# Linearly Seperable Data & Linear Decision Boundary



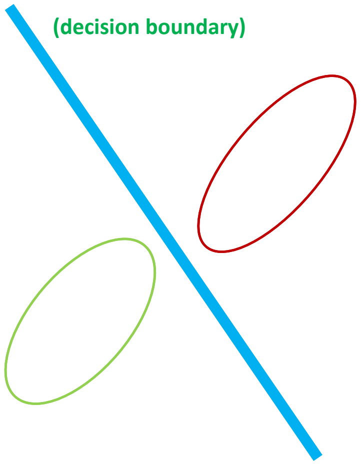

- **The data is linearly seperable (rarely in practice)**
- **A line becomes a decision boundary to determine if a new data point is class red/green**

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

**(N = 100 samples)**

# Separating Line & Mathematical Notation

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




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

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

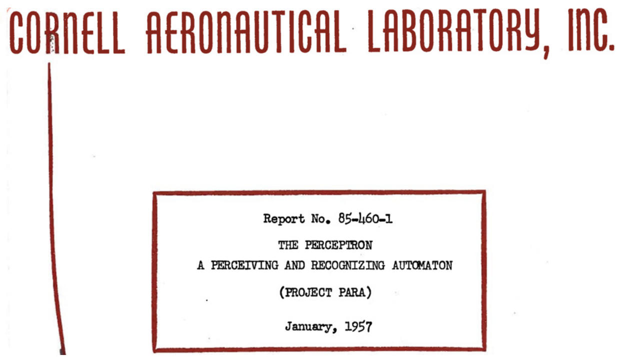
($w_i$ and threshold are still unknown to us)

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

(compact notation)

# A Simple Linear Learning Model – The Perceptron

- Human analogy in learning
  - 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'


CORNELL AERONAUTICAL LABORATORY, INC.

Report No. 85-460-1

THE PERCEPTRON

A PERCEIVING AND RECOGNIZING AUTOMATON

(PROJECT PARA)

January, 1957

*[5] F. Rosenblatt, 1957*

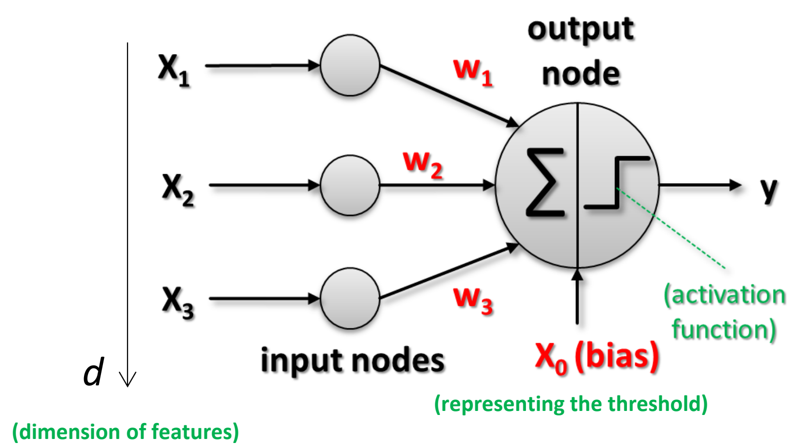
$$(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)$$

(training data)

(modelled as bias term)

$$sign\left(\underbrace{\left(\sum_{i=1}^{d} w_i x_i\right) - threshold}_{\text{(the signal)}}\right)$$

(activation function, +1 or -1)



# Perceptron – Example of a Boolean Function

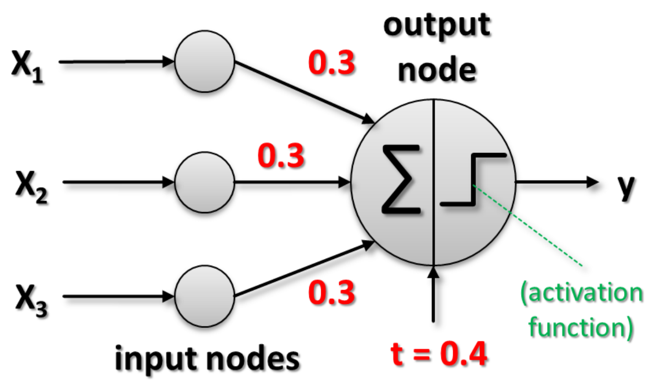
| | x1 | x2 | x3 | y |
|---|---|---|---|---|
| 1 | 1 | 0 | 0 | -1 |
| 2 | 1 | 0 | 1 | 1 |
| 3 | 1 | 1 | 0 | 1 |
| 4 | 1 | 1 | 1 | 1 |
| 5 | 0 | 0 | 1 | -1 |
| 6 | 0 | 1 | 0 | -1 |
| 7 | 0 | 1 | 1 | 1 |
| 8 | 0 | 0 | 0 | -1 |

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

(training data)

(training phase)

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




(trained perceptron model)

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

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

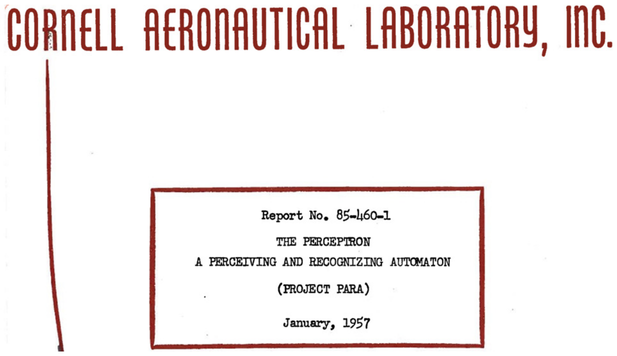
(e.g. consider sample #3, sum is positive (0.2) → +1)

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

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

# Summary Perceptron & Hypothesis Set h(x)

- When: Solving a linear classification problem
  - Goal: learn a simple value (+1/-1) above/below a certain threshold
  - Class label renamed: Iris-setosa = -1 and Iris-virginica = +1
- Input: $\mathbf{X} = x_1, \ldots, x_d$ (attributes in one dataset)



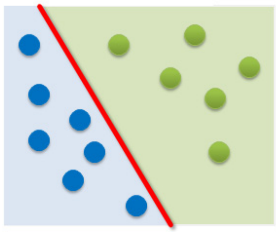

*[5] F. Rosenblatt, 1957*

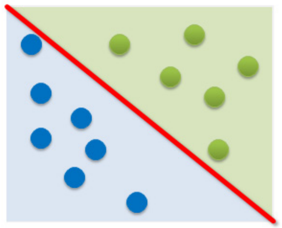
- Linear formula (take attributes and give them different weights – think of 'impact of the attribute')
  - All learned formulas are different hypothesis for the given problem

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

(parameters that define one hypothesis vs. another)

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





(red parameters correspond to the redline in graphics)

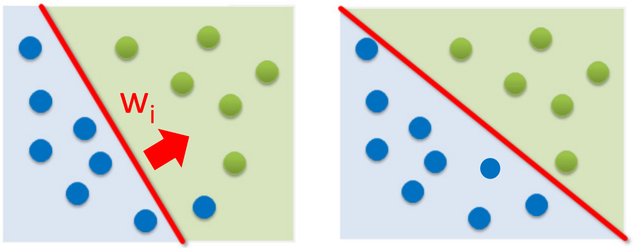
(but question remains: how do we actually learn $w_i$ and threshold?)

# Perceptron Learning Algorithm – Understanding Vector W

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

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

(points on the decision boundary satisfy this equation)




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

(vector notation, using T = transpose)

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

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

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

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

(equivalent dotproduct notation)

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

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

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

*[6] Rosenblatt, 1958*

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

# Perceptron Learning Algorithm – Learning Step

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

(one point at a time is picked)

1. Pick one misclassified training point where:

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

2. Update the weight vector:

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

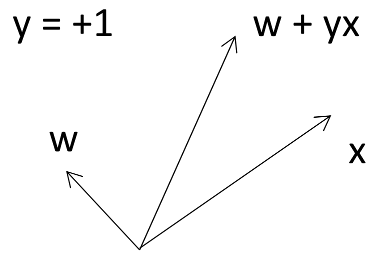
$$\mathbf{w} \leftarrow \mathbf{w} + y_n \mathbf{x}_n$$

($y_n$ is either +1 or -1)

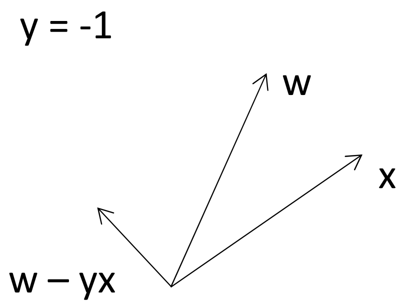
- Terminates when there are no misclassified points

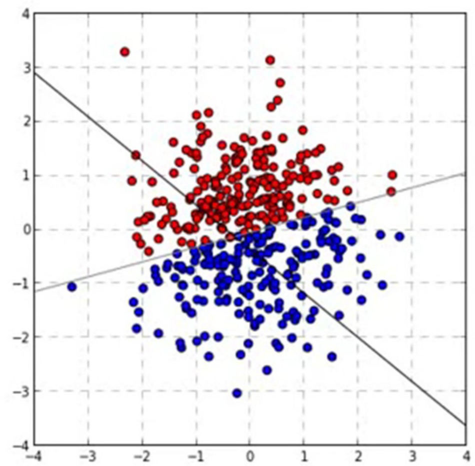
(converges only with linearly seperable data)

(a)



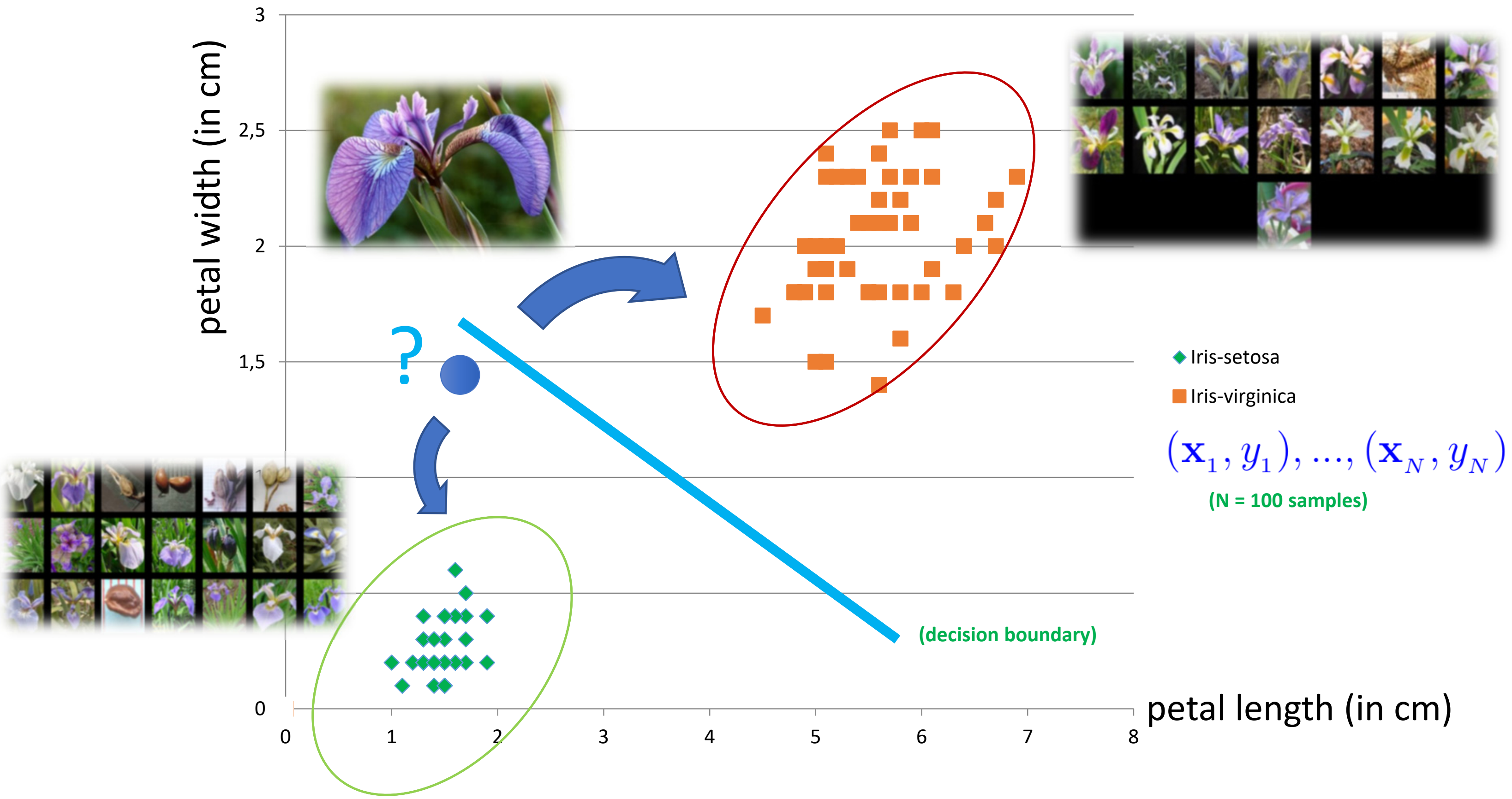

(b)



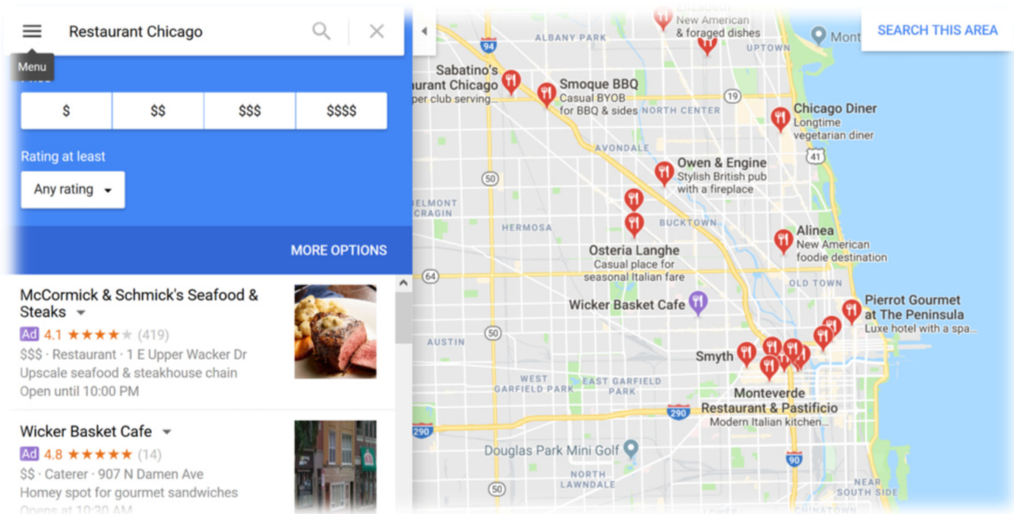

*[7] Perceptron Visualization*

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



# Food Inspection in Chicago: Advanced Application Example

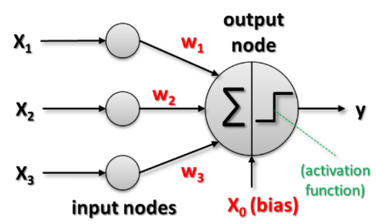
1. Some pattern exists:
   - Believe in a pattern with 'quality violations in checking restaurants' will somehow influence if food inspection pass or fail (binary classification)
2. No exact mathematical formula
   - To the best of our knowledge there is no precise formula for this problem
3. Data exists
   - Data collection from City of Chicago



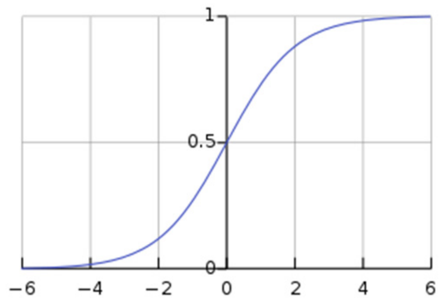

- **The goal of the advanced machine learning application with food inspection of restaurants in the City of Chicago is to predict the outcome of food inspection of new Chicago restaurants given some of existing violations of older restaurants already obtained in Chicago**
- **A key question is if the new restaurant will pass or fail the inspection just based on violations**

# Logistic Regression Using Non-Linear Activation Function

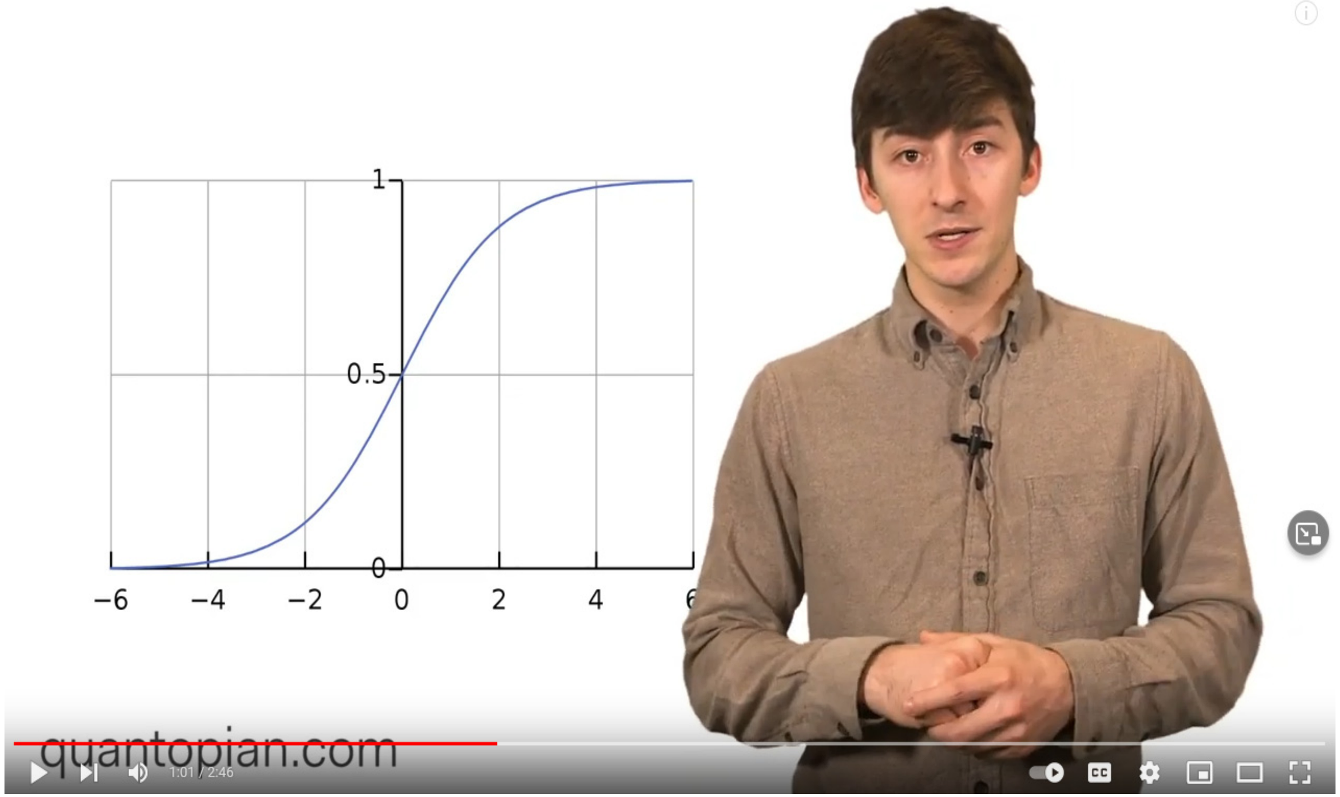
- Linear Classification
  - Simple binary classification (linearly seperable)
  - Linear combination of the inputs $x_i$ with weights $w_i$




- Linear Regression
  - Real value with the activiation being the identity function
  - E.g. how much sales given marketing money spend on TV advertising
- Logistic Regression
  - Model/error measure/learning algorithm is different
  - Captures non-linear data dependencies using the so-called Sigmoid function
  - Key idea is to bring values between 0 and 1 to estimate a probability
  - (candidate model for pass/fail in our application)

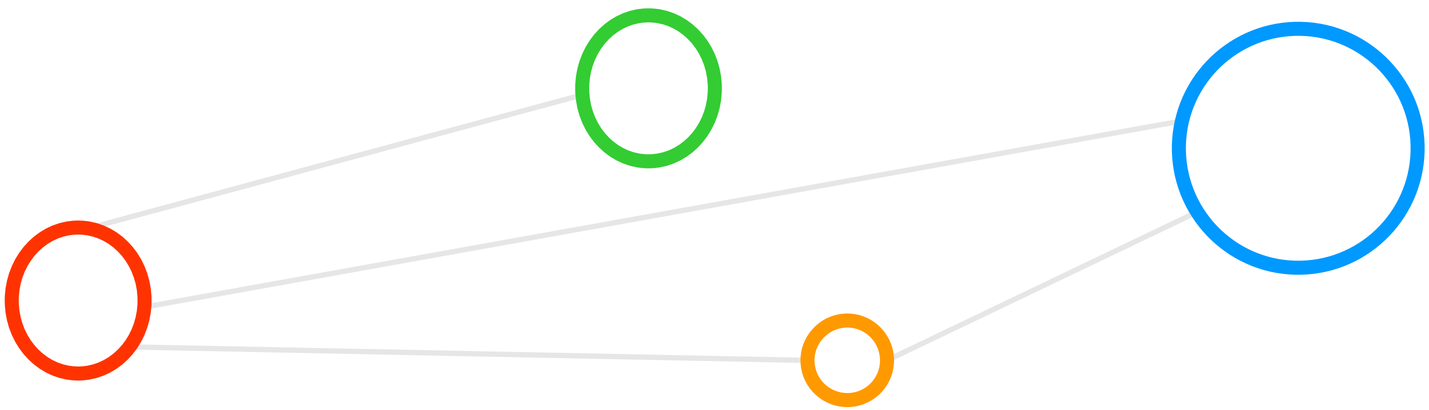


# [Video for further Studies] Logistic Regression Shortly Explained

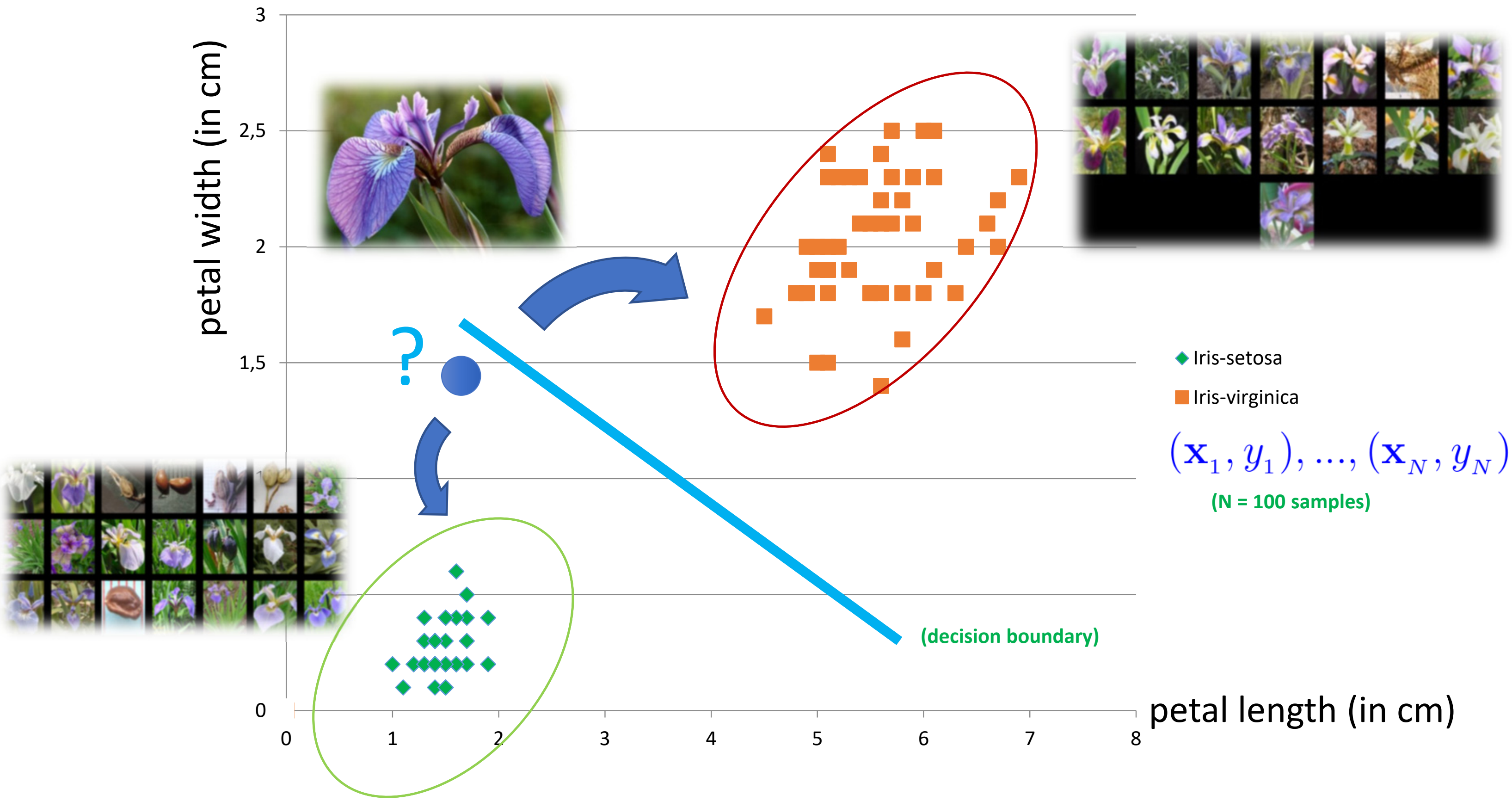



*[8] YouTube video, Logistic Regression*

# Artificial Neural Network (ANN) Basics



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




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

(N = 100 samples)

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

# Perceptron Model – General Mathematical Notation for one Neuron

non-linear activation function

linear combination of input data

$$\hat{y} = g\left(1 * {\color{red}w_0} + \sum_{i=1}^{m} x_i * {\color{red}w_i}\right) \quad \Rightarrow \quad \hat{y} = g\left({\color{red}w_0} + X^T {\color{red}\mathbf{w}}\right)$$

Output

Bias

Sum

Constants

$$X = \begin{bmatrix} x_1 \\ \vdots \\ x_m \end{bmatrix} \qquad {\color{red}\mathbf{w} = \begin{bmatrix} w_1 \\ \vdots \\ w_m \end{bmatrix}}$$

Input Data

Trainable Weights

- **Simplify the perceptron learning model formula with techniques from linear algebra for mathematical convenience**

# Multi-Class Classification: Handwritten Character Recognition MNIST Dataset

- Metadata
  - Not very challenging dataset, but good for benchmarks & tutorials
- When working with the dataset
  - Dataset is not in any standard image format like jpg, bmp, or gif (i.e. file format not known to a graphics viewer)
  - Data samples are stored in a simple file format that is designed for storing vectors and multidimensional matrices (i.e. numpy arrays)
  - 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

**▪ Handwritten Character Recognition MNIST dataset is a subset of a larger dataset from US National Institute of Standards (NIST)**

**▪ MNIST handwritten digits includes corresponding labels with values 0-9 and is therefore a labeled dataset**

**▪ MNIST digits have been size-normalized to 28 * 28 pixels & are centered in a fixed-size image for direct processing**

**▪ Two separate files for training & test: 60000 training samples (~47 MB) & 10000 test samples (~7.8 MB)**

(10 class classification problem)

```python
import numpy as np
from keras.datasets import mnist
```

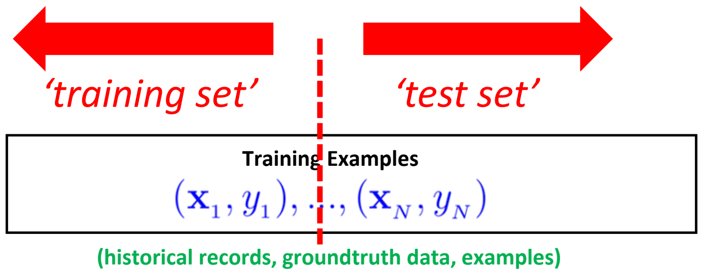
```python
# download and shuffled as training and testing set
(X_train, y_train), (X_test, y_test) = mnist.load_data()
```

(downloads data into ~home/.keras/datasets as NPZ file format of numpy that provides storage of array data using gzip compression)

# MNIST Dataset – Training/Testing Datasets & One Character Encoding

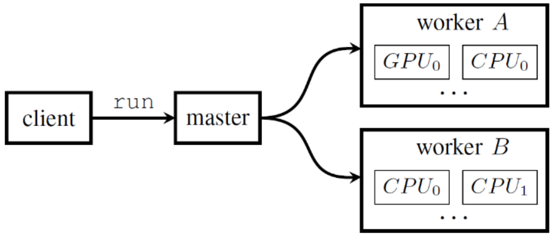
- **Different phases in machine learning**
- **Training phases is a hypothesis search**
- **Testing phase checks if we are on the right track once the hypothesis is clear**
- **Validation phane for model selection (set fixed parameters and set model types)**

- Work on two disjoint datasets
  - One for training only (i.e. training set)
  - One for testing only (i.e. test set)
  - Exact seperation is rule of thumb per use case (e.g. 10 % training, 90% test)
  - Practice: If you get a dataset take immediately test data away ('throw it into the corner and forget about it during modelling')
  - Once we learned from training data it has an 'optimistic bias'
  - Usually start by exploring the dataset and its format & labels



# Deep Learning Frameworks using GPUs also good for Artificial Neural Networks

- TensorFlow
  - One of the most popular deep learning frameworks available today
  - Execution on multi-core CPUs or many-core GPUs




*[9] Tensorflow Web page*

- **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**
- **New versions of Tensorflow have Keras shipped with it as well & many further tools**

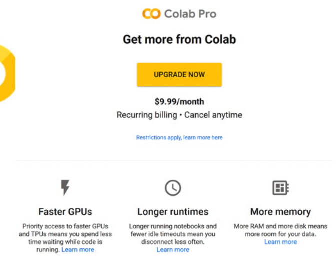
*[10] Keras Web page*

- Keras
  - Often used in combination with low-level frameworks like Tensorflow

- **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**
- **Created deep learning models with Keras run seamlessly on CPU and GPU via low-level deep learning frameworks**
- **The key idea behind the Keras tool is to enable faster experimentation with deep networks**
- **Keras is part of the more recent TensorFlow distributions**

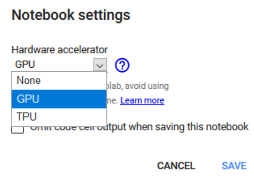
# Using Google Colaboratory Cloud Infrastructure for Deep Learning with GPUs

- Google Colaboratory (free & pro version for 9.99 $ / month)
  - 'Colab' notebooks are Jupyter notebooks that run in the Google cloud
  - Possible to run Apache Spark via PySpark Jupyter notebooks in Colab (cf. Lecture 3)
  - Possible to train Deep Learning networks via GPUs & Jupyter notebooks in Colab
  - Highly integrated with other Google services (e.g., Google Drive for data)
  - Access to vendor-specific Tensor Processing Units (TPUs)




*[11] Google Colaboratory*

(for international students: watch out – it uses the browser language automatically)




```
[ ] # initialize the optimizer and model
    model = CNN.build(input_shape=INPUT_SHAPE, classes=NB_CLASSES)
    model.compile(loss="categorical_crossentropy", optimizer=OPTIMIZER,
    metrics=["accuracy"])

    ---------------------------------------------------------------------------
    TypeError                                 Traceback (most recent call last)
    <ipython-input-11-13ca928a46c2> in <module>()
          1 # initialize the optimizer and model
    ----> 2 model = CNN.build(input_shape=INPUT_SHAPE, classes=NB_CLASSES)
          3 model.compile(loss="categorical_crossentropy", optimizer=OPTIMIZER,
          4 metrics=["accuracy"])

                              5 frames
    /usr/local/lib/python3.6/dist-packages/tensorflow/python/keras/utils/generic_utils.py in validate_kwargs(kwargs, allowed_kwargs, error_message)
        776   for kwarg in kwargs:
        777     if kwarg not in allowed_kwargs:
    --> 778       raise TypeError(error_message, kwarg)
        779
        780

    TypeError: ('Keyword argument not understood:', 'border_mode')

    SEARCH STACK OVERFLOW
```

(Clouds also face this update problem)

- **The portability of deep learning codes is hindered by the frequent updates of the different APIs of deep learning frameworks like Keras, Tensorflow, etc. (cf. different AWS EC2 AMI versions)**

(tutorials & codes need updates)

- **Update Oct/2016**: Updated for Keras 1.1.0, TensorFlow 0.10.0 and scikit-learn v0.18.
- **Update Mar/2017**: Updated for Keras 2.0.2, TensorFlow 1.0.1 and Theano 0.9.0.
- **Update Sep/2019**: Updated for Keras 2.2.5 API.

*[12] Machine Learning Mastery MNIST Tutorial*

- **Google Colaboratory offers 'Colab' notebooks that are implemented with Jupyter notebooks that in turn run in the Google cloud and are highly integrated with other Google cloud services such as Google Drive thus making 'Colab' notebooks easy to set up, access, and share with others**

# MNIST Dataset – Data Exploration Script Training Data – Revisited

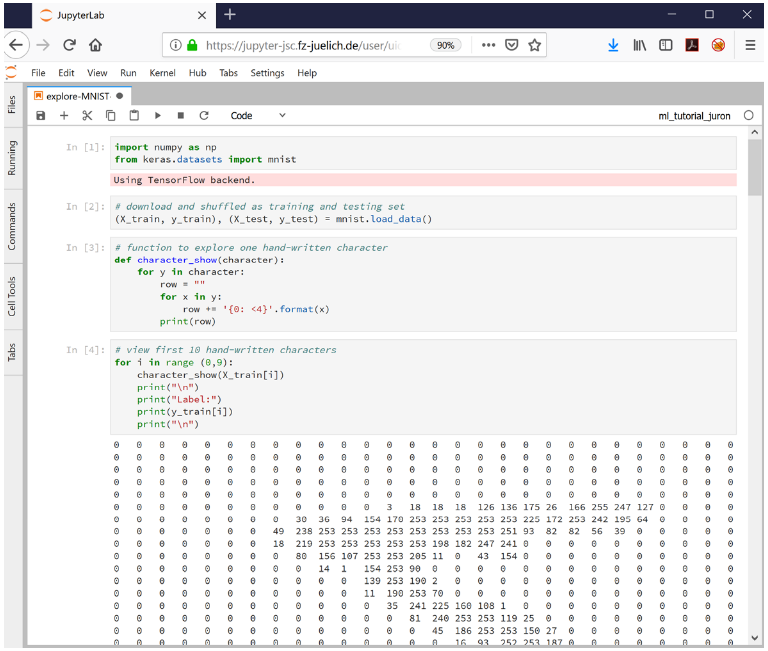
```python
import numpy as np
from keras.datasets import mnist

# download and shuffled as training and testing set
(X_train, y_train), (X_test, y_test) = mnist.load_data()
```

```python
# function to explore one hand-written character
def character_show(character):
    for y in character:
        row = ""
        for x in y:
            row += '{0: <4}'.format(x)
        print(row)
```

```python
# view first 10 hand-written 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)

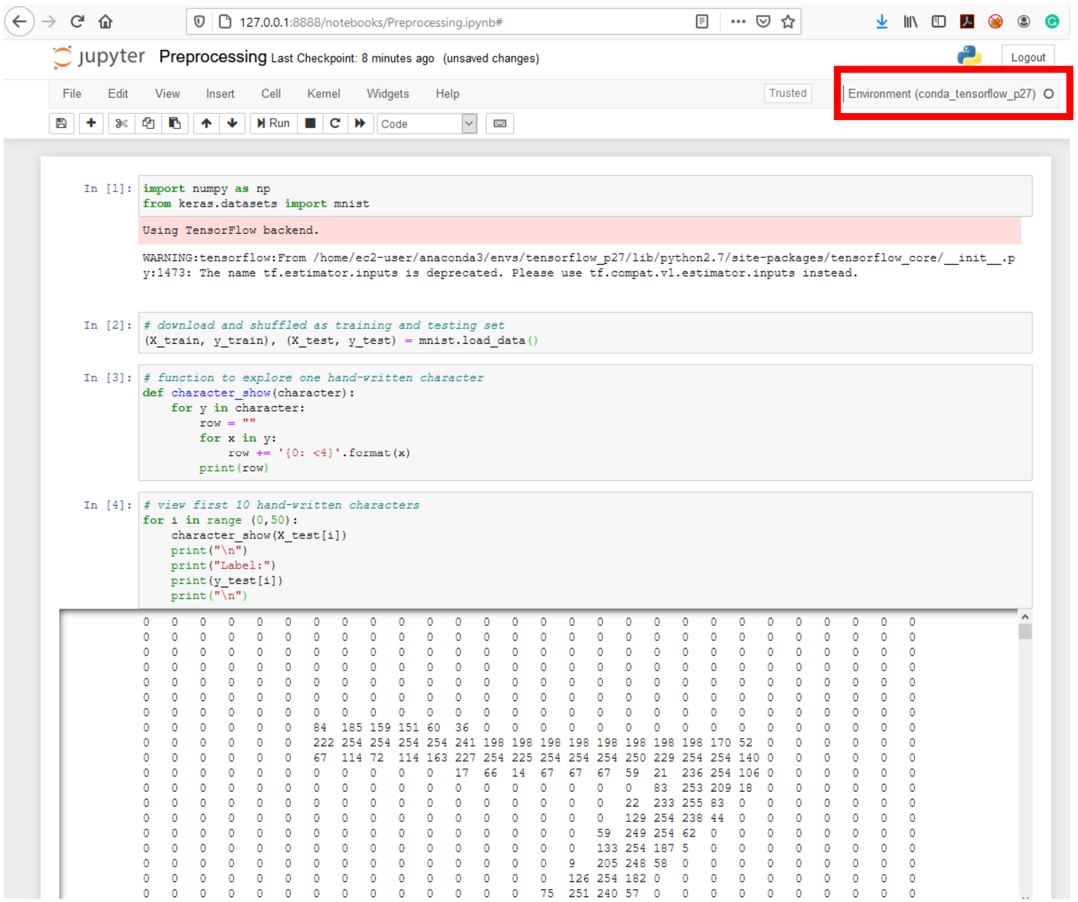
- Small helper function that prints row-wise one ‘hand-written‘ character with the grey levels stored in training dataset
- Should reveal the nature of the number (aka label)




[13] Jupyter @ JSC

- Example: loop of the training 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)

# Data Inspection using Keras Dataset MNIST with Visualization in Jupyter



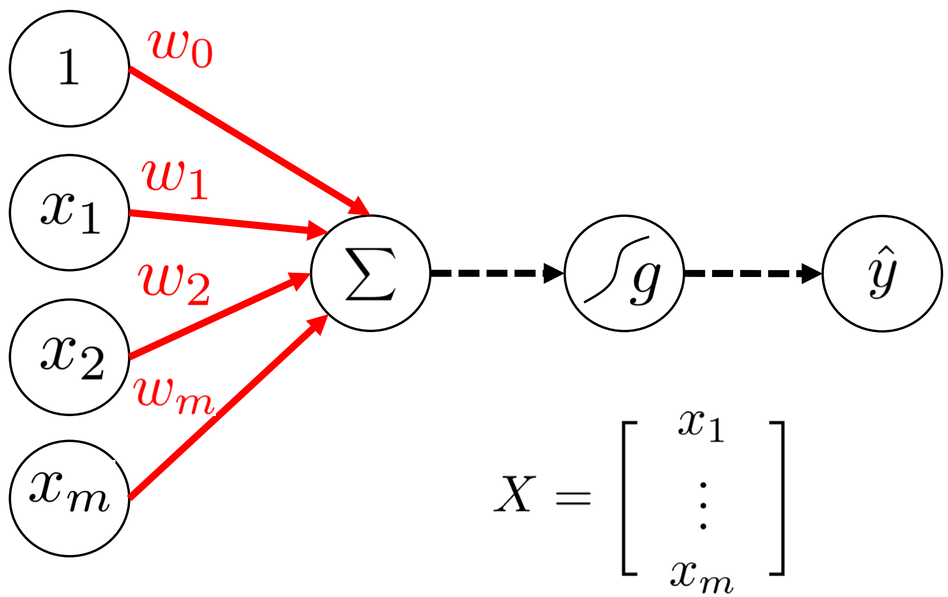
# MNIST Dataset with Perceptron Learning Model – Need for Reshape

- Two dimensional dataset (28 x 28)
  - Does not fit well with input to Perceptron Model
  - Need to prepare the data even more
  - Reshape data → we need one long vector

- **Note that the reshape from two dimensional MNIST data to one long vector means that we loose the surrounding context**
- **Loosing the surrounding context is one factor why later in this lecture deep learning networks achieving essentially better performance by, e.g., keeping the surrounding context**

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

Label:
5



# MNIST Dataset – Reshape & Normalization – Example

(one long input vector with length 784)

$$X = \begin{bmatrix} x_1 \\ \vdots \\ x_m \end{bmatrix}$$

(numbers are between 0 and 1)

```
784 input pixel values per train samples
784 input pixel values per test samples
[0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.01176471 0.07058824 0.07058824 0.07058824
 0.49411765 0.53333336 0.6862745  0.10196079 0.6509804  1.
 0.96862745 0.49803922 0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.11764706 0.14117648 0.36862746 0.6039216
 0.6666667  0.99215686 0.99215686 0.99215686 0.99215686 0.99215686
 0.88235295 0.6745098  0.99215686 0.9490196  0.7647059  0.2509804
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.19215687
 0.93333334 0.99215686 0.99215686 0.99215686 0.99215686 0.99215686
 0.99215686 0.99215686 0.99215686 0.9843137  0.3647059  0.32156864
 0.32156864 0.21960784 0.15294118 0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.07058824 0.85882354 0.99215686
 0.99215686 0.99215686 0.99215686 0.99215686 0.7764706  0.7137255
```

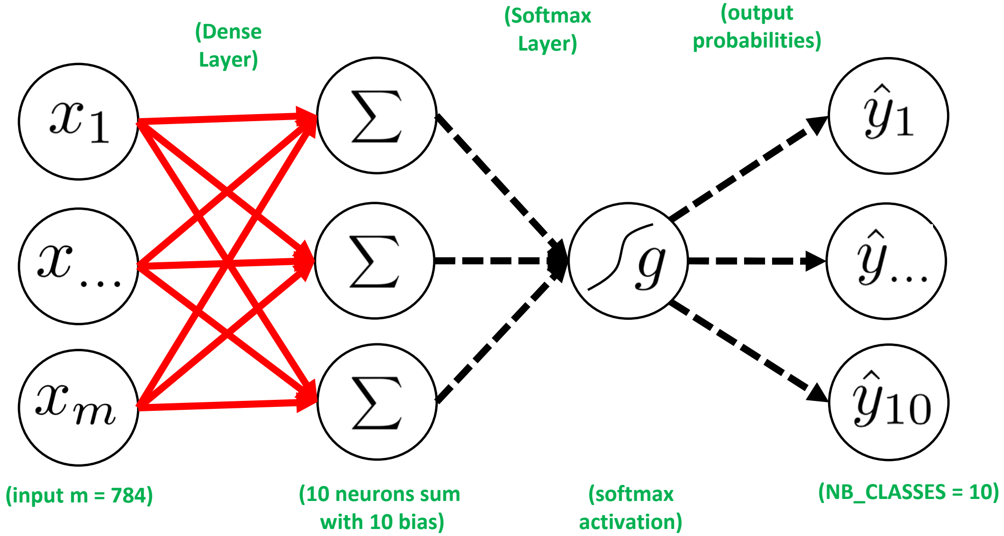
(two dimensional original input with spatial context)

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

Label:
5
```

- **While a reshape is necessary for the Perceptron Model and traditional Artificial Neural Network (ANN) it is unfortunate that we loose the spatial context on one large vectors instead of 2D**
- **That is a loss of information that hinders better learning as shown in deep learning networks that use spatial context**

# MNIST Dataset & Multi Output Perceptron Model

- 10 Class Classification Problem
  - Use 10 Perceptrons for 10 outputs with softmax activation function (enables probabilities for 10 classes)




```python
from keras.models import Sequential
from keras.layers.core import Dense, Activation
```

```python
# model Keras sequential
model = Sequential()
```

```python
# add fully connected layer - input with output
model.add(Dense(NB_CLASSES, input_shape=(RESHAPED,)))
```

```python
# add activation function layer to get class probabilities
model.add(Activation('softmax'))
```

```python
# printout a summary of the model to understand model complexity
model.summary()
```

(parameters = 784 * 10 + 10 bias = 7850)

```
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
```

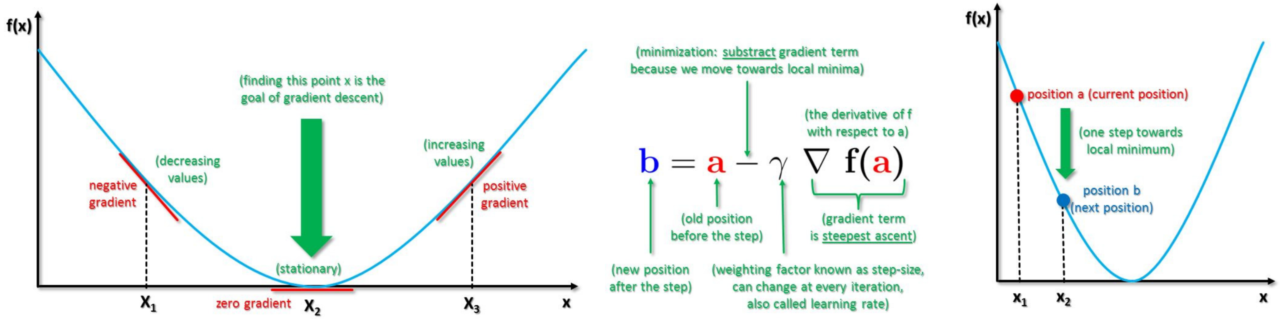
- **Note that the output units are independent among each other in contrast to neural networks with one hidden layer**
- **The output of softmax gives class probabilities**
- **The non-linear Activation function 'softmax' represents a generalization of the sigmoid function – it squashes an n-dimensional vector of arbitrary real values into a n-dimenensional vector of real values in the range of 0 and 1 – here it aggregates 10 answers provided by the Dense layer with 10 neurons**

# MNIST Dataset & Compile Multi Output Perceptron Model

- Compile the model
  - Optimizer as algorithm used to update weights while training the model
  - Specify loss function (i.e. objective function) that is used by the optimizer to navigate the space of weights
  - (note: process of optimization is also called loss minimization)
  - Indicate metric for model evaluation (e.g., accuracy)
- Specify loss function
  - Compare prediction vs. given class label
  - E.g. categorical crossentropy

```python
from keras.optimizers import SGD
```

```python
OPTIMIZER = SGD() # optimization technique
```




- **Compile the model to be executed by the Keras backend (e.g. TensorFlow)**
- **Optimizer Gradient Descent (GD) uses all the training samples available for a step within a iteration**
- **Optimizer Stochastic Gradient Descent (SGD) converges faster: only one training samples used per iteration**
- **Loss function is a multi-class logarithmic loss: target is *ti,j* and prediction is *pi,j***
- **Categorical crossentropy is suitable for multiclass label predictions (default with softmax)**

```python
# specify loss, optimizer and metric
model.compile(loss='categorical_crossentropy', optimizer=OPTIMIZER, metrics=['accuracy'])
```

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

*[14] Big Data Tips, Gradient Descent*

# Full Script: MNIST Dataset – Model 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.optimizers import SGD
from keras.utils import np_utils
```

```
# parameter setup
NB_EPOCH = 20
BATCH_SIZE = 128
NB_CLASSES = 10 # number of outputs = number of digits
OPTIMIZER = SGD() # optimization technique
VERBOSE = 1
```

```
# download and shuffled as training and testing set
(X_train, y_train), (X_test, y_test) = mnist.load_data()
```

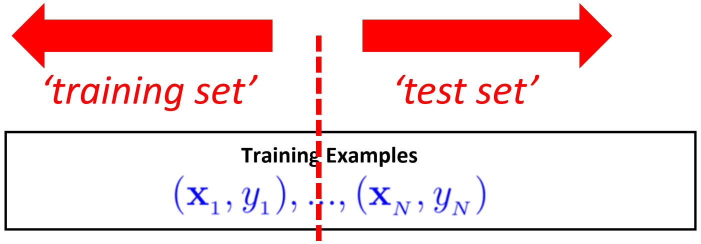
```
# X_train is 60000 rows of 28x28 values --> reshaped in 60000 x 784
RESHAPED = 784
X_train = X_train.reshape(60000, RESHAPED)
X_test = X_test.reshape(10000, RESHAPED)
X_train = X_train.astype('float32')
X_test = X_test.astype('float32')
```

```
# normalize
X_train /= 255
X_test /= 255
```

```
# output number of samples
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 overall training set – at each iteration the optimizer adjusts the weights so that the objective function is minimized – increasing leads to better accuracy, but also to overfitting**
- **BATCH_SIZE: number of training instances taken into account before the optimizer performs a weight update to the whole model**
- **OPTIMIZER: Stochastic Gradient Descent ('SGD')**

- **Data load shuffled between training and testing set in files**
- **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]; a usual technique, e.g. model training smoother & avoids data structure problems**




(historical records, groundtruth data, examples)

# Full Script: MNIST Dataset – Fitting a Multi Output Perceptron Model

(full script continued from previous slide)

```
# convert class label vectors using one hot encoding
Y_train = np_utils.to_categorical(y_train, NB_CLASSES)
Y_test = np_utils.to_categorical(y_test, NB_CLASSES)

# model Keras sequential
model = Sequential()

# add fully connected layer - input with output
model.add(Dense(NB_CLASSES, input_shape=(RESHAPED,)))

# add activation function layer to get class probabilities
model.add(Activation('softmax'))

# printout a summary of the model to understand model complexity
model.summary()

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

# model training
history = model.fit(X_train, Y_train, batch_size=BATCH_SIZE, epochs=NB_EPOCH, verbose=VERBOSE)

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

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

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

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

- **Train the model ('fit') using selected batch & epoch sizes on training & test data**

# Running a Simple ANN with no hidden layers – Multi-Output-Perceptron

127.0.0.1:8888/notebooks/ANN_0_Hidden.ipynb

jupyter ANN_0_Hidden Last Checkpoint: 27 minutes ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help | Trusted | Environment (conda_tensorflow_p27) | Logout

In [1]:
```
from __future__ import print_function
import numpy as np
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers.core import Dense, Activation
from keras.optimizers import SGD
from keras.utils import np_utils
np.random.seed(1671) # for reproducibility
```

```
Using TensorFlow backend.
```

```
WARNING:tensorflow:From /home/ec2-user/anaconda3/envs/tensorflow_p27/lib/python2.7/site-packages/tensorflow_core/__init__.p
y:1473: The name tf.estimator.inputs is deprecated. Please use tf.compat.v1.estimator.inputs instead.
```

In [2]:
```
# parameter setup
NB_EPOCH = 200
BATCH_SIZE = 128
VERBOSE = 1
NB_CLASSES = 10 # number of outputs = number of digits
OPTIMIZER = SGD() # optimization technique
N_HIDDEN = 128
VALIDATION_SPLIT=0.2 # 20% of TRAIN is reserved for VALIDATION
```

In [3]:
```
# download and shuffled as training and testing set
(X_train, y_train), (X_test, y_test) = mnist.load_data()
# X_train is 60000 rows of 28x28 values --> reshaped in 60000 x 784
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')
# normalize
X_train /= 255
X_test /= 255
print(X_train.shape[0], 'train samples')
print(X_test.shape[0], 'test samples')
# convert class vectors to binary class matrices
Y_train = np_utils.to_categorical(y_train, NB_CLASSES)
Y_test = np_utils.to_categorical(y_test, NB_CLASSES)
```

```
60000 train samples
10000 test samples
```

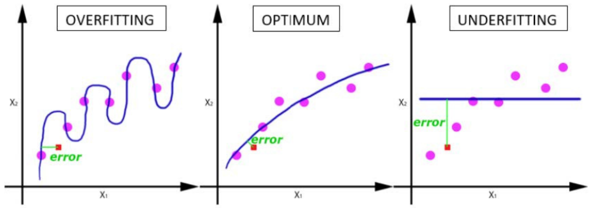
```
Epoch 199/200
48000/48000 [==============================] - 1s 14us/step - loss: 0.2762 - acc: 0.9229 - val_loss: 0.2757 - val_acc: 0.92
41
Epoch 200/200
48000/48000 [==============================] - 1s 14us/step - loss: 0.2761 - acc: 0.9230 - val_loss: 0.2756 - val_acc: 0.92
41
```

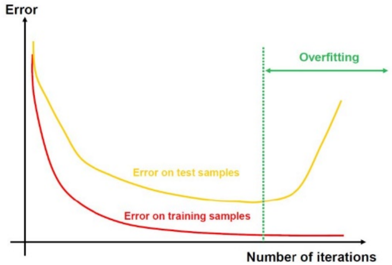
Out[6]: 'Mon, 26 Oct 2020 23:12:30 +0000'

In [7]:
```
# model evaluation
score = model.evaluate(X_test, Y_test, verbose=VERBOSE)
print("Test score:", score[0])
print('Test accuracy:', score[1])
```

```
10000/10000 [==============================] - 0s 22us/step
Test score: 0.2773858518630266
Test accuracy: 0.9227
```

- **Note that the outcome of the training process is the result of optimization techniques like SGD that tend to vary 'a bit'**
- **Note that the outcome of the training process can be dependent on the length of training increasing accuracy to a certain point when overfitting starts**
- **Overfitting can be controlled with validation and regularization techniques that belong to advanced machine learning methods to be studied in full university machine learning course in detail**



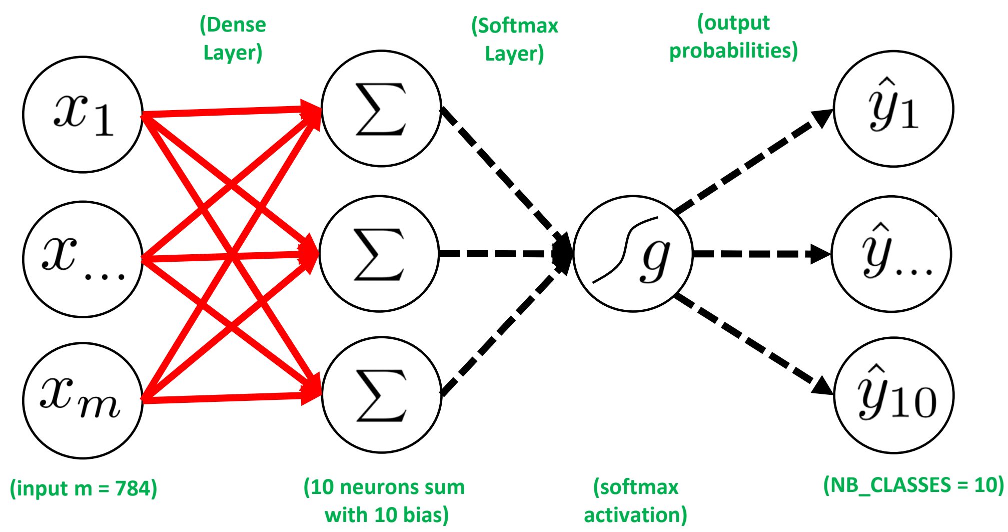



# MNIST Dataset – A Multi Output Perceptron Model – Output & Evaluation

```
Epoch 7/20
60000/60000 [==============================] - 2s 26us/step - loss: 0.4419 - acc: 0.8838
Epoch 8/20
60000/60000 [==============================] - 2s 26us/step - loss: 0.4271 - acc: 0.8866
Epoch 9/20
60000/60000 [==============================] - 2s 25us/step - loss: 0.4151 - acc: 0.8888
Epoch 10/20
60000/60000 [==============================] - 2s 26us/step - loss: 0.4052 - acc: 0.8910
Epoch 11/20
60000/60000 [==============================] - 2s 26us/step - loss: 0.3968 - acc: 0.8924
Epoch 12/20
60000/60000 [==============================] - 2s 25us/step - loss: 0.3896 - acc: 0.8944
Epoch 13/20
60000/60000 [==============================] - 2s 26us/step - loss: 0.3832 - acc: 0.8956
Epoch 14/20
60000/60000 [==============================] - 2s 25us/step - loss: 0.3777 - acc: 0.8969
Epoch 15/20
60000/60000 [==============================] - 2s 25us/step - loss: 0.3727 - acc: 0.8982
Epoch 16/20
60000/60000 [==============================] - 1s 24us/step - loss: 0.3682 - acc: 0.8989
Epoch 17/20
60000/60000 [==============================] - 1s 25us/step - loss: 0.3641 - acc: 0.9001
Epoch 18/20
60000/60000 [==============================] - 1s 25us/step - loss: 0.3604 - acc: 0.9007
Epoch 19/20
60000/60000 [==============================] - 2s 25us/step - loss: 0.3570 - acc: 0.9016
Epoch 20/20
60000/60000 [==============================] - 1s 24us/step - loss: 0.3538 - acc: 0.9023
```

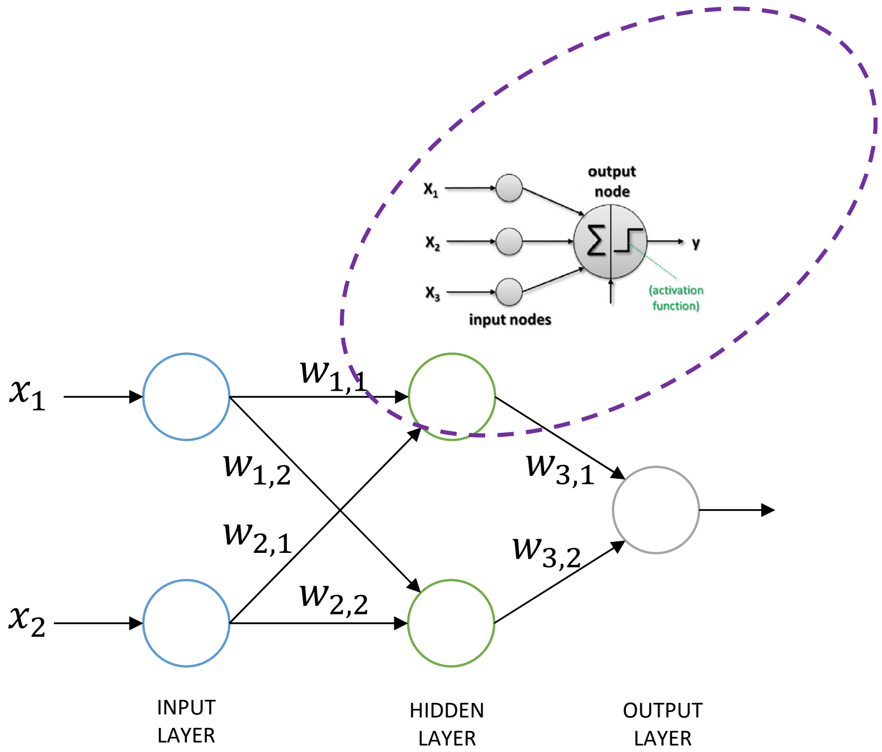
```python
# model evaluation
score = model.evaluate(X_test, Y_test, verbose=VERBOSE)
print("Test score:", score[0])
print('Test accuracy:', score[1])
```

```
10000/10000 [==============================] - 0s 41us/step
Test score: 0.33423959468007086
Test accuracy: 0.9101
```

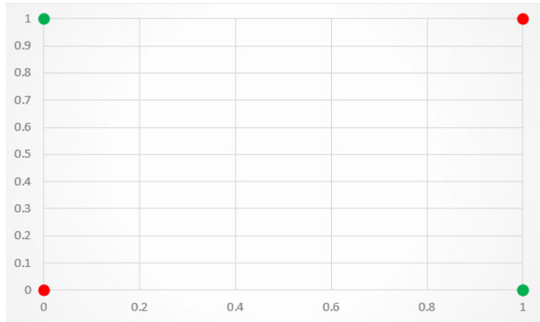



- **How to improve the model design by extending the neural network topology?**
- **Which layers are required?**
- **Think about input layer need to match the data – what data we had?**
- **Maybe hidden layers?**
- **How many hidden layers?**
- **What activation function for which layer (e.g. maybe ReLU)?**
- **Think Dense layer – Keras?**
- **Think about final Activation as Softmax → output probability**

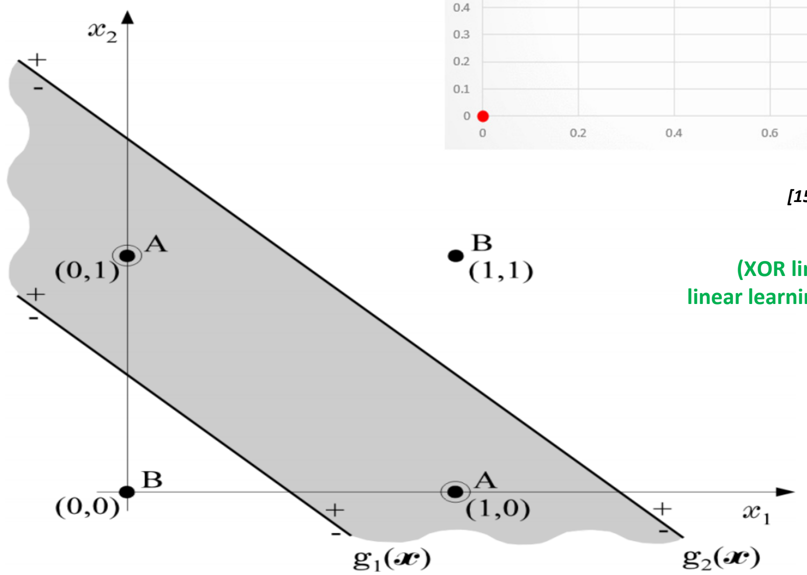
# From Limits of Linear Perceptron Model to Multi Layer Perceptrons (MLP)




| Input 1 | Input 2 | Output |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 1 | 0 |
| 1 | 0 | 1 |



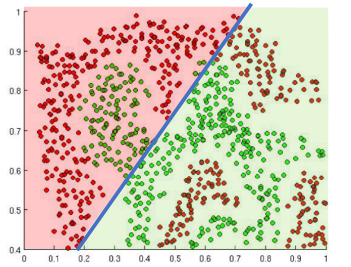

*[15] The XOR Problem*



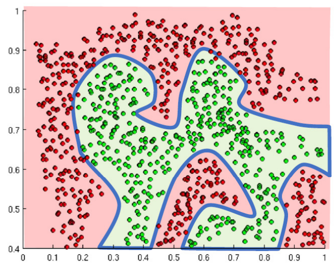

(XOR limit of
linear learning models)

*[16] Multilayer Perceptron*

# Introduce Non-Linearities – The Role of Activation Functions in ANNs



Linear Activation functions produce linear decisions no matter the network size



Non-linearities allow us to approximate arbitrarily complex functions

non-linear activation function

linear combination of input data

$$\hat{y} = g\left(1 * w_0 + \sum_{i=1}^{m} x_i * w_i\right)$$

Output Bias Sum

$$\hat{y} = g\left(w_0 + X^T \mathbf{w}\right)$$

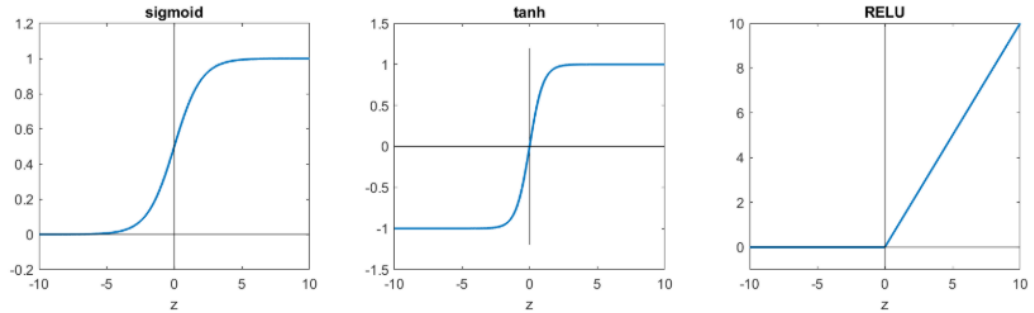
Constants

$$X = \begin{bmatrix} x_1 \\ \vdots \\ x_m \end{bmatrix} \quad \mathbf{w} = \begin{bmatrix} w_1 \\ \vdots \\ w_m \end{bmatrix}$$

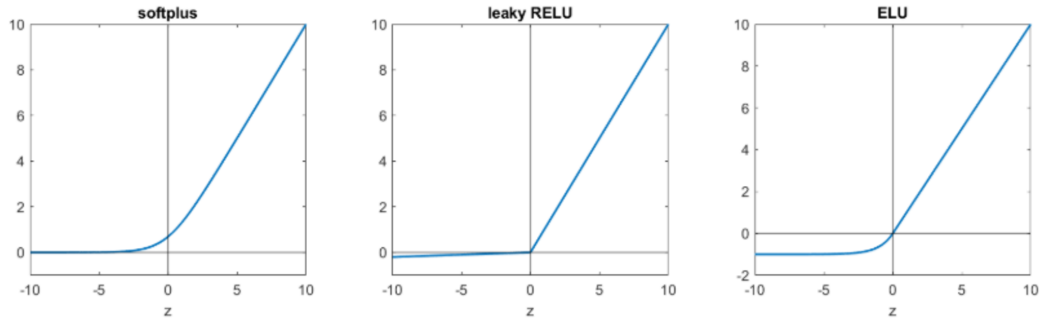
Input Data

Trainable Weights

- The choice of the architecture and the activation function plays a key role in the definition of the network
- Each activation function takes a single number and performs a certain fixed mathematical operation on it




$$g(z) = \frac{1}{1 + e^{-z}} \qquad g(z) = \tanh z \qquad g(z) = \max(z, 0)$$



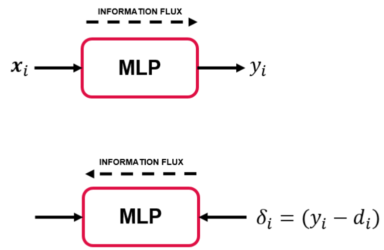

$$g(z) = \log(1 + e^{z}) \qquad g(z) = \max(z, z\alpha),\ 0 < \alpha < 1$$

*[17] Introduction to Deep Learning* *[18] Understanding the Neural Network*

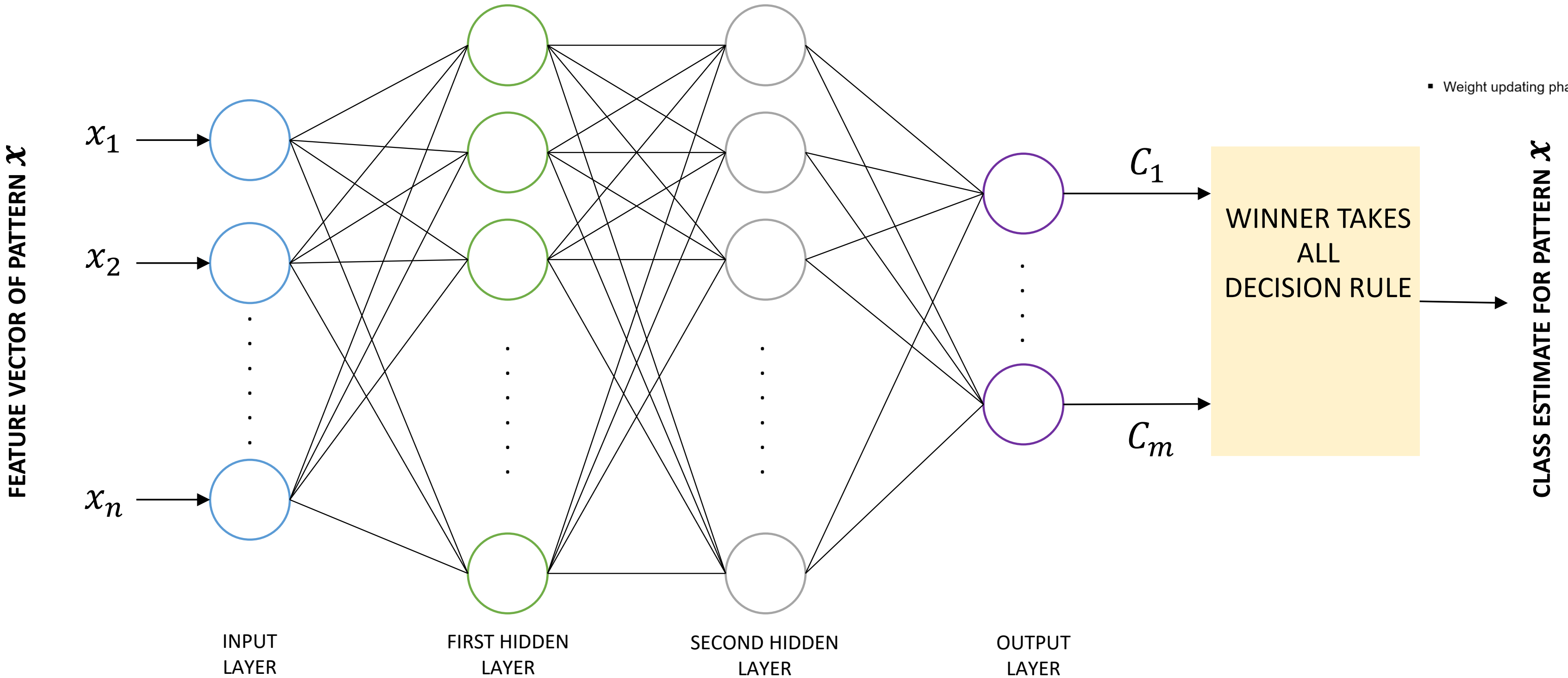
# Artificial Neural Network (ANN) Basic Network Topology & Learning Algorithm

- **Forward interconnection of several layers of perceptrons create an Artificial Neural Network (ANNs)**
- **Multi Layer Perceptrons (MLPs) can be used as universal approximators**
- **In classification problems, they allow modeling nonlinear discriminant functions**
- **Interconnecting neurons aims at increasing the capability of modeling complex input-output relationships**

- Forward propagation phase
- Backward propagation phase
- Weight updating phase




(backpropagation of error)



# Training an ANN with Backpropogation performing Weight Updates

1. Initialize weights randomly $\sim\mathcal{N}(0,\sigma^2)$
2. Loop until convergence
3. Compute gradient $\frac{\partial\mathcal{L}(\boldsymbol{W})}{\partial\boldsymbol{W}}$ (it explains how the loss changes with respect to each of the weights)
4. Update weights $\boldsymbol{W} \coloneqq \boldsymbol{W} - \eta\,\frac{\partial\mathcal{L}(\boldsymbol{W})}{\partial\boldsymbol{W}}$ (in the opposite direction of the gradient )
5. Return weights

(compute-intensive)

(Learning rate)

(goal: minimize the 'Loss' using an optimization problem)

- **The Learning Rate determines the adjustment magnitude & how much do you trust the computed gradient**
- **Computing the gradients using optimization is the most computational part when there is a high number of weights**

$$\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N}(y_i - d_i)^2$$

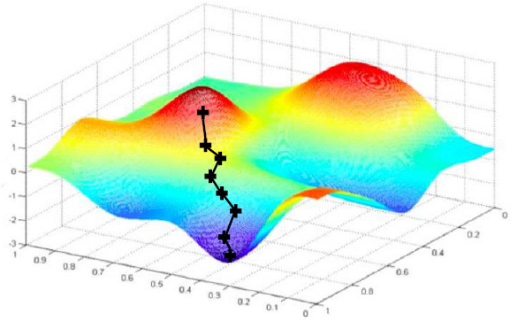
TOTAL NUMBER OF TRAINING SAMPLES

OUTPUT VALUE OBTAINED BY THE MLP FOR THE i-th SAMPLE

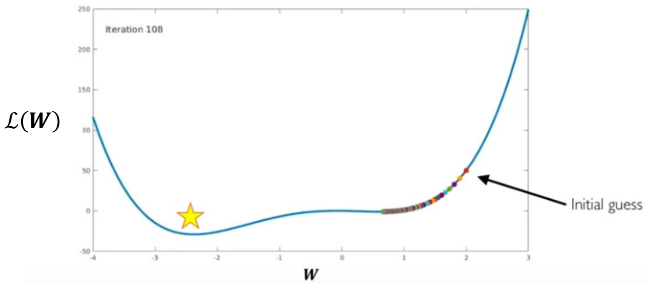
DESIRED OUTPUT (TARGET) VALUE FOR THE i-th SAMPLE

$$\boldsymbol{W}^* = \arg\min \sum_{i=1}^{N} \mathcal{L}(f(x_i;\boldsymbol{W}), d_i)$$

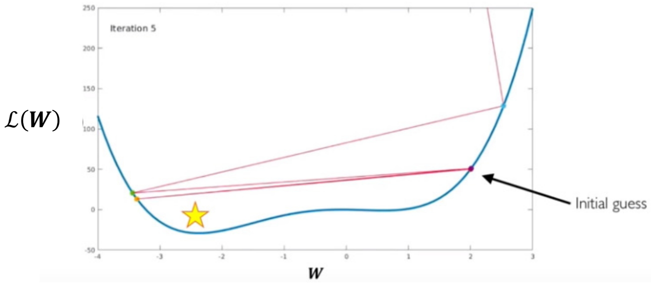
$$\boldsymbol{W}^* = \arg\min_{\boldsymbol{W}} \mathcal{L}(\mathbf{W})$$



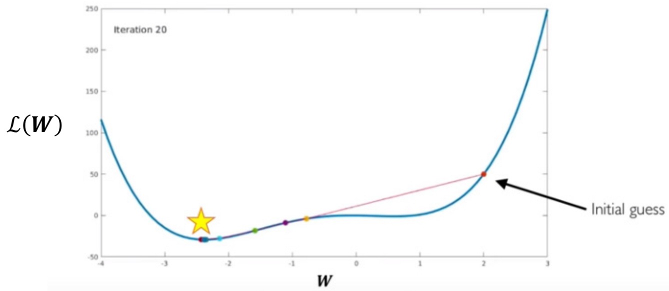
[17] Introduction to Deep Learning




**Small learning rate** converges slowly and gets stuck in false local minima




**Large learning rates** overshoot, become unstable and diverge




**Stable learning rates** converge smoothly and avoid local minima

# MNIST Dataset – Add Two Hidden Layers for Artificial Neural Network (ANN)

- **All parameter value remain the same as before**
- **We add N_HIDDEN as parameter in order to set 128 neurons in one hidden layer – this number is a hyperparameter that is not directly defined and needs to be find with parameter search**

```
# parameter setup
NB_EPOCH = 20
BATCH_SIZE = 128
NB_CLASSES = 10 # number of outputs = number of digits
OPTIMIZER = SGD() # optimization technique
VERBOSE = 1
N_HIDDEN = 128 # number of neurons in one hidden layer
```

```
# model Keras sequential
model = Sequential()
```

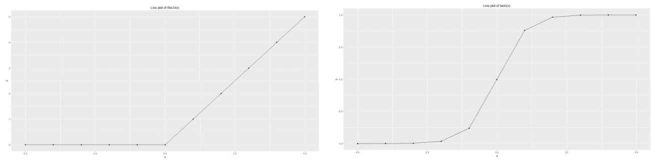
```
# modeling step
# 2 hidden layers each N_HIDDEN neurons
model.add(Dense(N_HIDDEN, input_shape=(RESHAPED,)))
model.add(Activation('relu'))
model.add(Dense(N_HIDDEN))
model.add(Activation('relu'))
model.add(Dense(NB_CLASSES))
```

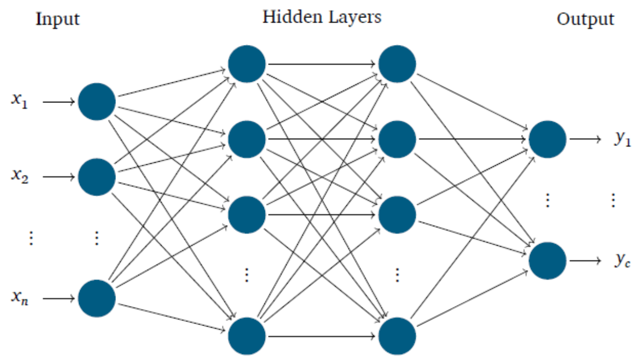
```
# add activation function layer to get class probabilities
model.add(Activation('softmax'))
```

*[19] big-data.tips, 'Relu Neural Network'*

*[20] big-data.tips, 'tanh'*

(activation functions ReLU & Tanh)



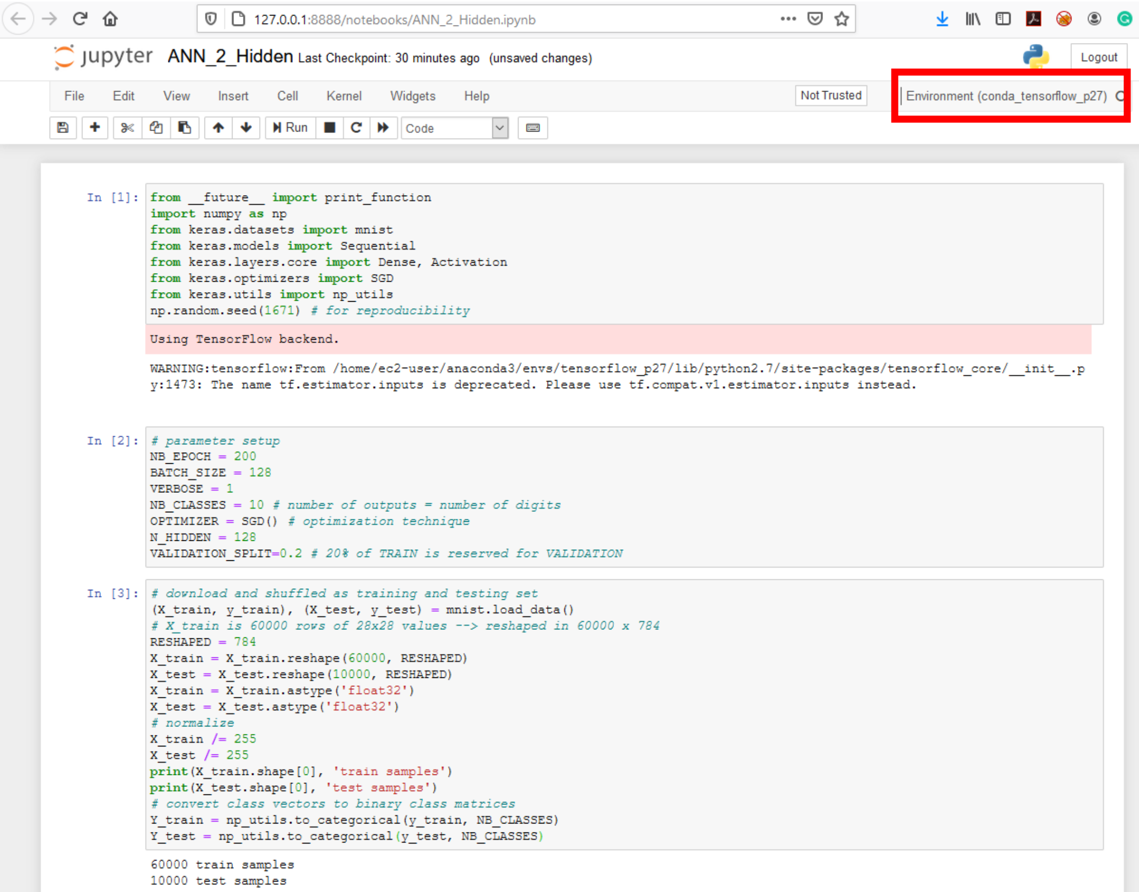


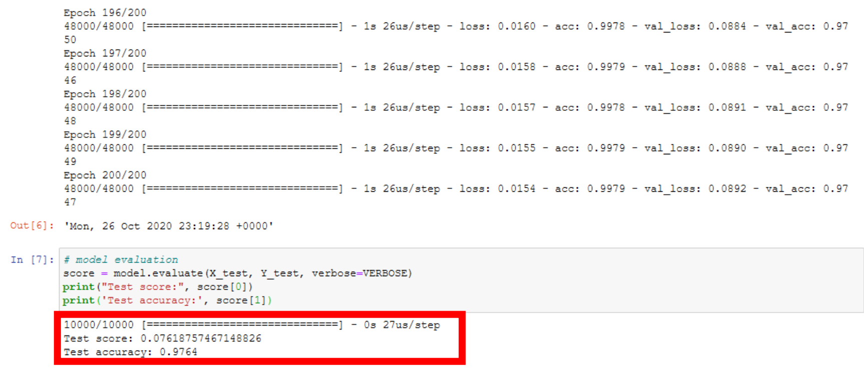

```
model.add(Dense(N_HIDDEN))
model.add(Activation('relu'))
```

```
model.add(Dense(N_HIDDEN))
model.add(Activation('tanh'))
```

- **The non-linear Activation function 'relu' represents a so-called Rectified Linear Unit (ReLU) that only recently became very popular because it generates good experimental results in ANNs and more recent deep learning models – it just returns 0 for negative values and grows linearly for only positive values**
- **A hidden layer in an ANN can be represented by a fully connected Dense layer in Keras by just specifying the number of hidden neurons in the hidden layer**

# Running a Simple ANN with two hidden layers (200 Epochs: very long learning)



# MNIST Dataset – ANN Model Parameters & Output Evaluation (20 Epochs)

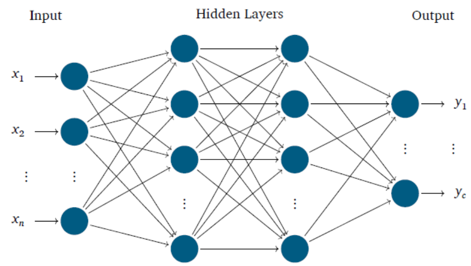
```
Epoch 7/20
60000/60000 [==============================] - 1s 18us/step - loss: 0.2743 - acc: 0.9223
Epoch 8/20
60000/60000 [==============================] - 1s 18us/step - loss: 0.2601 - acc: 0.9266
Epoch 9/20
60000/60000 [==============================] - 1s 18us/step - loss: 0.2477 - acc: 0.9301
Epoch 10/20
60000/60000 [==============================] - 1s 18us/step - loss: 0.2365 - acc: 0.9329
Epoch 11/20
60000/60000 [==============================] - 1s 18us/step - loss: 0.2264 - acc: 0.9356
Epoch 12/20
60000/60000 [==============================] - 1s 18us/step - loss: 0.2175 - acc: 0.9386
Epoch 13/20
60000/60000 [==============================] - 1s 18us/step - loss: 0.2092 - acc: 0.9412
Epoch 14/20
60000/60000 [==============================] - 1s 18us/step - loss: 0.2013 - acc: 0.9432
Epoch 15/20
60000/60000 [==============================] - 1s 18us/step - loss: 0.1942 - acc: 0.9454
Epoch 16/20
60000/60000 [==============================] - 1s 18us/step - loss: 0.1876 - acc: 0.9472
Epoch 17/20
60000/60000 [==============================] - 1s 18us/step - loss: 0.1813 - acc: 0.9487
Epoch 18/20
60000/60000 [==============================] - 1s 18us/step - loss: 0.1754 - acc: 0.9502
Epoch 19/20
60000/60000 [==============================] - 1s 18us/step - loss: 0.1700 - acc: 0.9522
Epoch 20/20
60000/60000 [==============================] - 1s 18us/step - loss: 0.1647 - acc: 0.9536
```

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

```
10000/10000 [==============================] - 0s 33us/step
Test score: 0.16286438911408185
Test accuracy: 0.9514
```

- ✓ **Multi Output Perceptron: ~91,01% (20 Epochs)**
- ✓ **ANN 2 Hidden Layers: ~95,14 % (20 Epochs)**

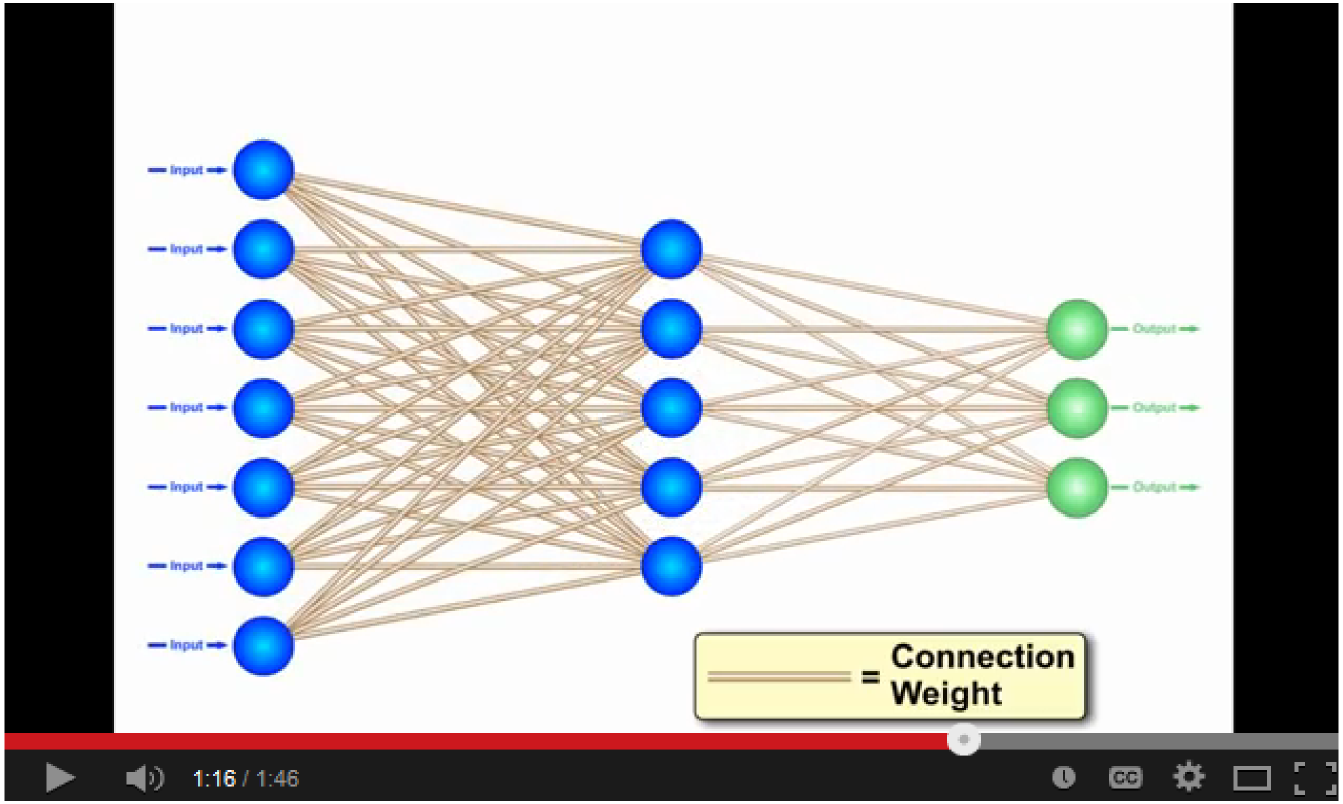
```
# printout a summary of the model to understand model complexity
model.summary()
```




```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 128)               100480
_________________________________________________________________
activation_1 (Activation)    (None, 128)               0
_________________________________________________________________
dense_2 (Dense)              (None, 128)               16512
_________________________________________________________________
activation_2 (Activation)    (None, 128)               0
_________________________________________________________________
dense_3 (Dense)              (None, 10)                1290
_________________________________________________________________
activation_3 (Activation)    (None, 10)                0
=================================================================
Total params: 118,282
Trainable params: 118,282
Non-trainable params: 0
_________________________________________________________________
```

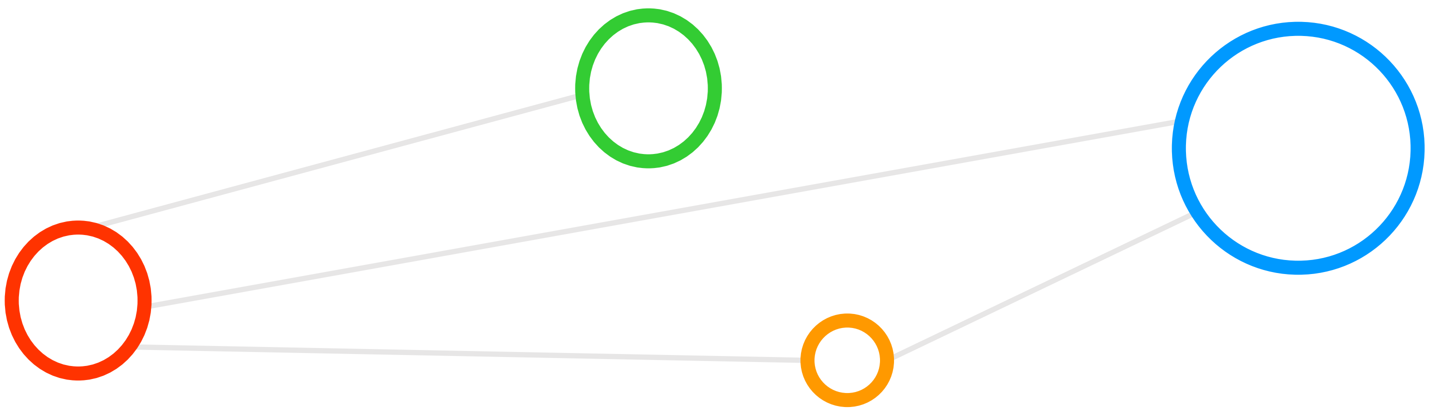
- **Dense Layer connects every neuron in this dense layer to the next dense layer with each of its neuron also called a fully connected network element with weights as trainiable parameters**
- **Choosing a model with different layers is a model selection that directly also influences the number of parameters (e.g. add Dense layer from Keras means new weights)**
- **Adding a layer with these new weights means much more computational complexity since each of the weights must be trained in each epoch (depending on #neurons in layer)**

# [Video for further Studies] Neural Networks Summary



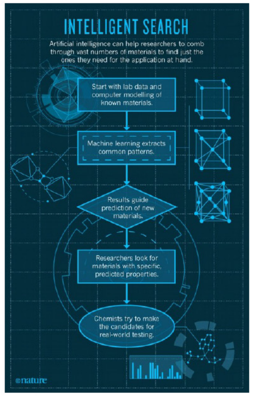

*[21] YouTube Video, Neural Networks – A Simple Explanation*

# Convolutional Neural Network (CNN) Basics

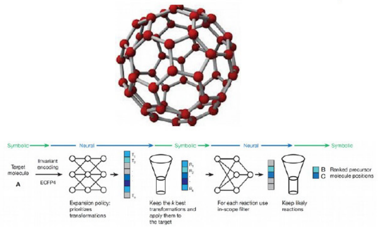


# Impact of Deep Learning in Various Application Domains

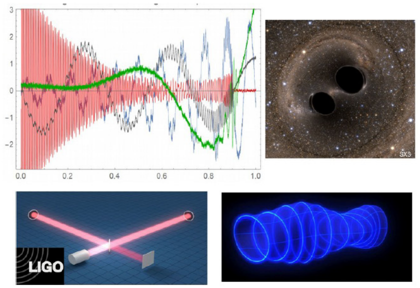
## Vision, Natural Speech

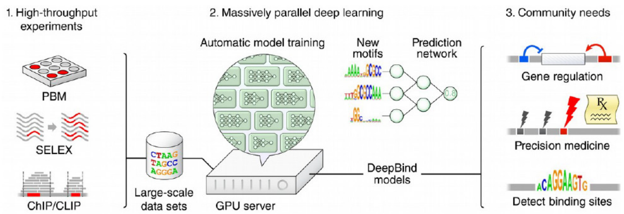

## Material Science, Chemistry



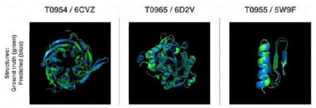
## Gravitational Waves Detection



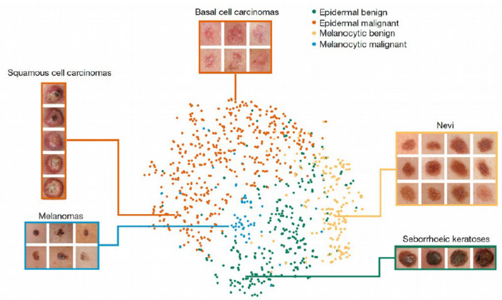

## Omics (Genomics, Proteomics, Metabolomics)



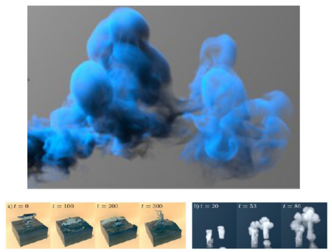

## Protein Folding



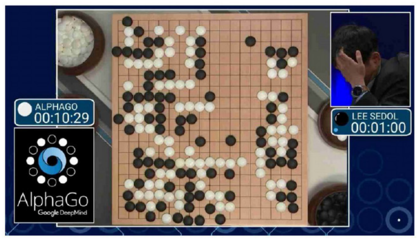

## Medical Imaging and Diagnostics



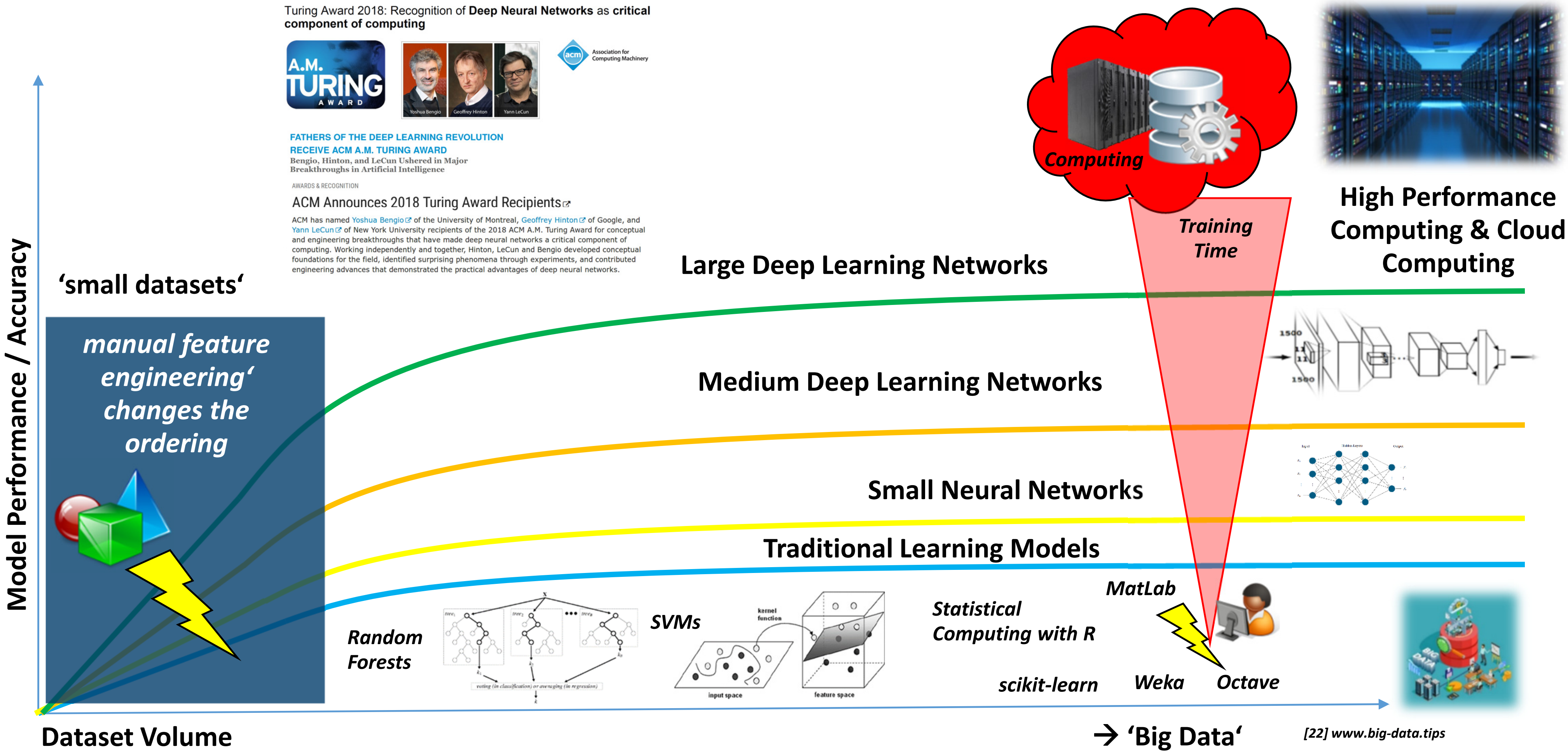

## Fluid, Gas Dynamics



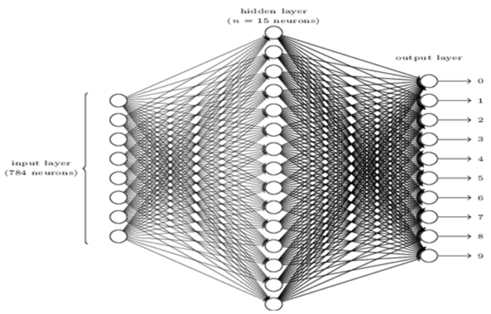
## Games, Control Optimization

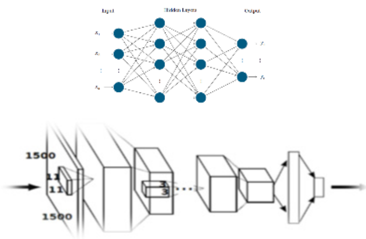


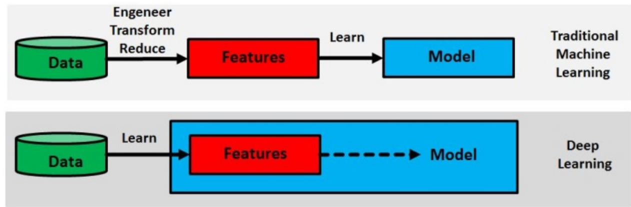
# Complex Relationships: ML & DL vs. HPC/Clouds & Big Data



# Innovative Deep Learning Techniques – Revisited

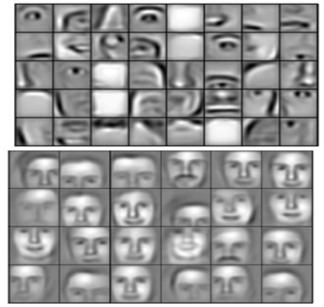





[23] M. Riedel, 'Deep Learning - Using a Convolutional Neural Network', Invited YouTube Lecture, six lectures, University of Ghent, 2017

[24] M. Riedel et al., 'Introduction to Deep Learning Models', JSC Tutorial, three days, JSC, 2019

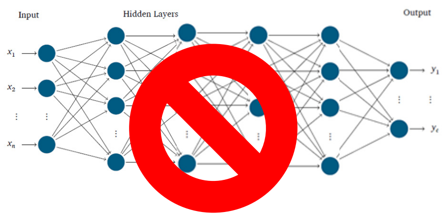


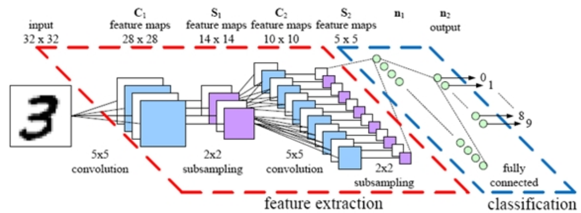

- **Innovation via specific layers and architecture types is the success of Convolutional Neural Networks (CNNs)**

[27] A. Rosebrock

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

[26] Neural Network 3D Simulation

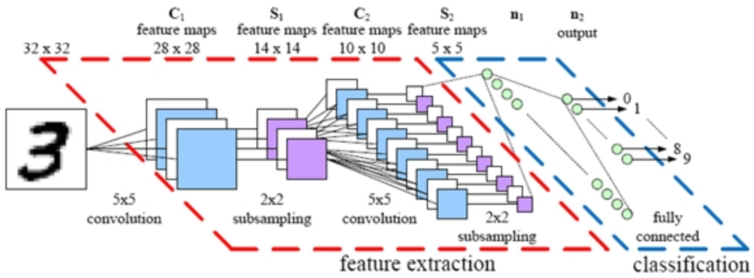


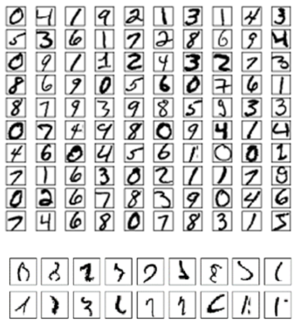
# CNNs – Basic Principles & Local Receptive Fields

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

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

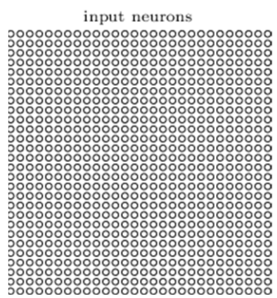



*[27] A. Rosebrock*

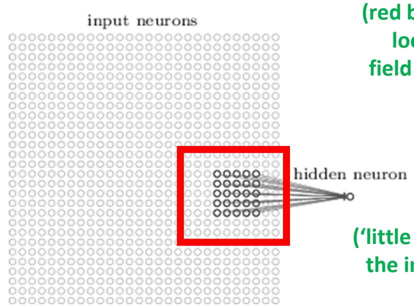


*[28] M. Nielsen*

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




(28 * 28 pixel image)



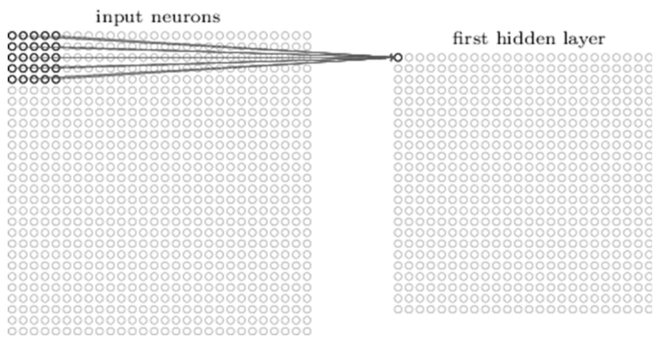

(5 * 5 local connectivity)

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

('little window' on the input pixels)

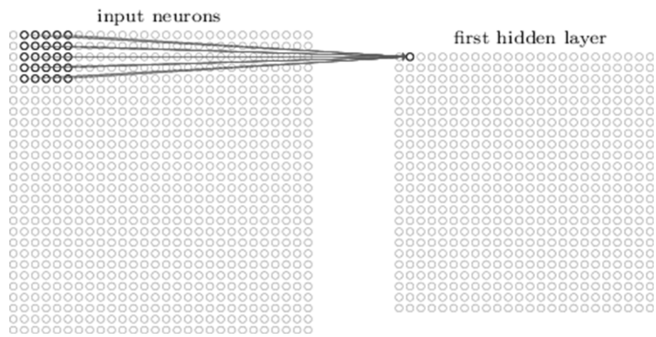
# CNNs – Principle Sliding with Convolutions & Feature Maps

- MNIST database example
  - Apply stride length = 1
  - Creates ‘feature map‘ of 24 * 24 neurons (hidden layer)
- Role of Convolutions & Filter
  - Valid convolution does not exceed the input‘s boundary
  - Same convolution adds a so called ‘padding‘ to maintain the input‘s dimension for each convolutional layer
  - Feature maps reflect where in the input a part of local features were activated by the applied filter



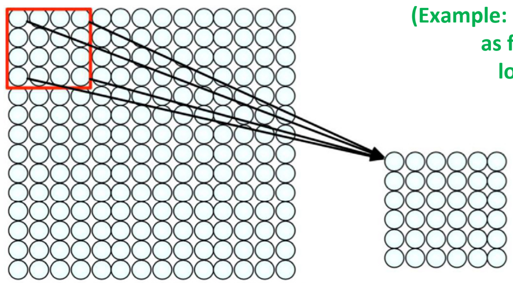

(28 * 28 pixel image)

(24 * 24 feature map)




(28 * 28 pixel image)

(24 * 24 feature map)



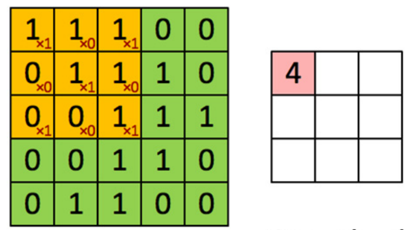
4x4 filter: matrix of weights $w_{ij}$

$$\sum_{i=1}^{4}\sum_{j=1}^{4} w_{ij}\, x_{i+p,j+q} + b$$

for neuron (p,q) in hidden layer

(Example: apply a set of weights as filter to extract local features)

[17] Introduction to Deep Learning




Image

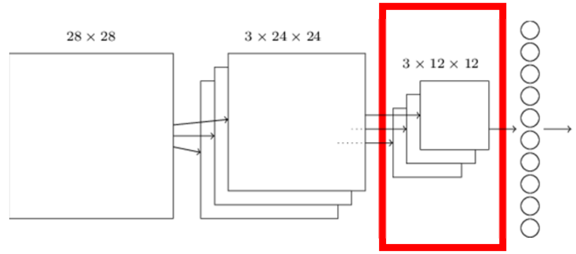
Convolved Feature

# CNNs – Understanding Application Example MNIST - Summary

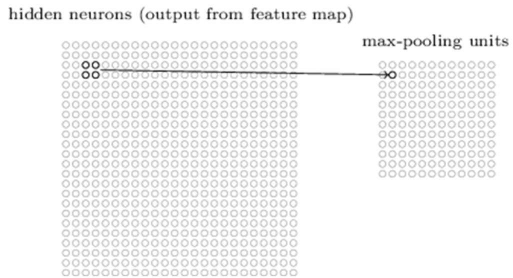
- MNIST database example
  - Pooling Layer & Apply ‘fully connected layer (flatten)‘: layer connects every neuron from the max-pooling outcome layer to every neuron of the 10 out neurons

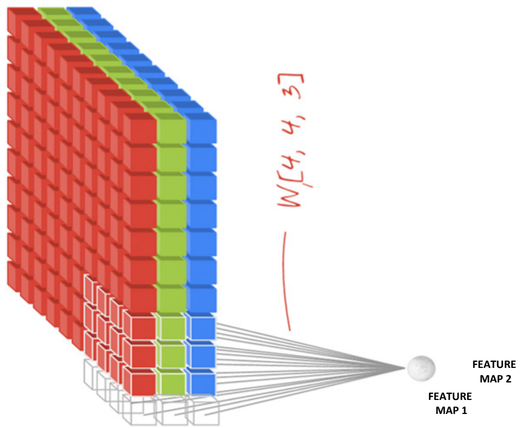
(another indicator that even with cutting edge technology machine learning never achieves 100% performance)

(Pooling layers: simplify the information in the output from the Convolution, e.g. max pooling )



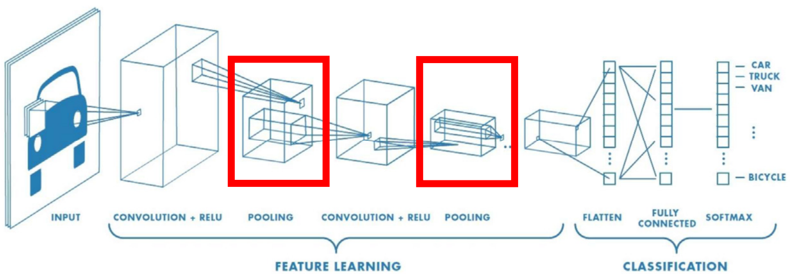

[28] M. Nielsen



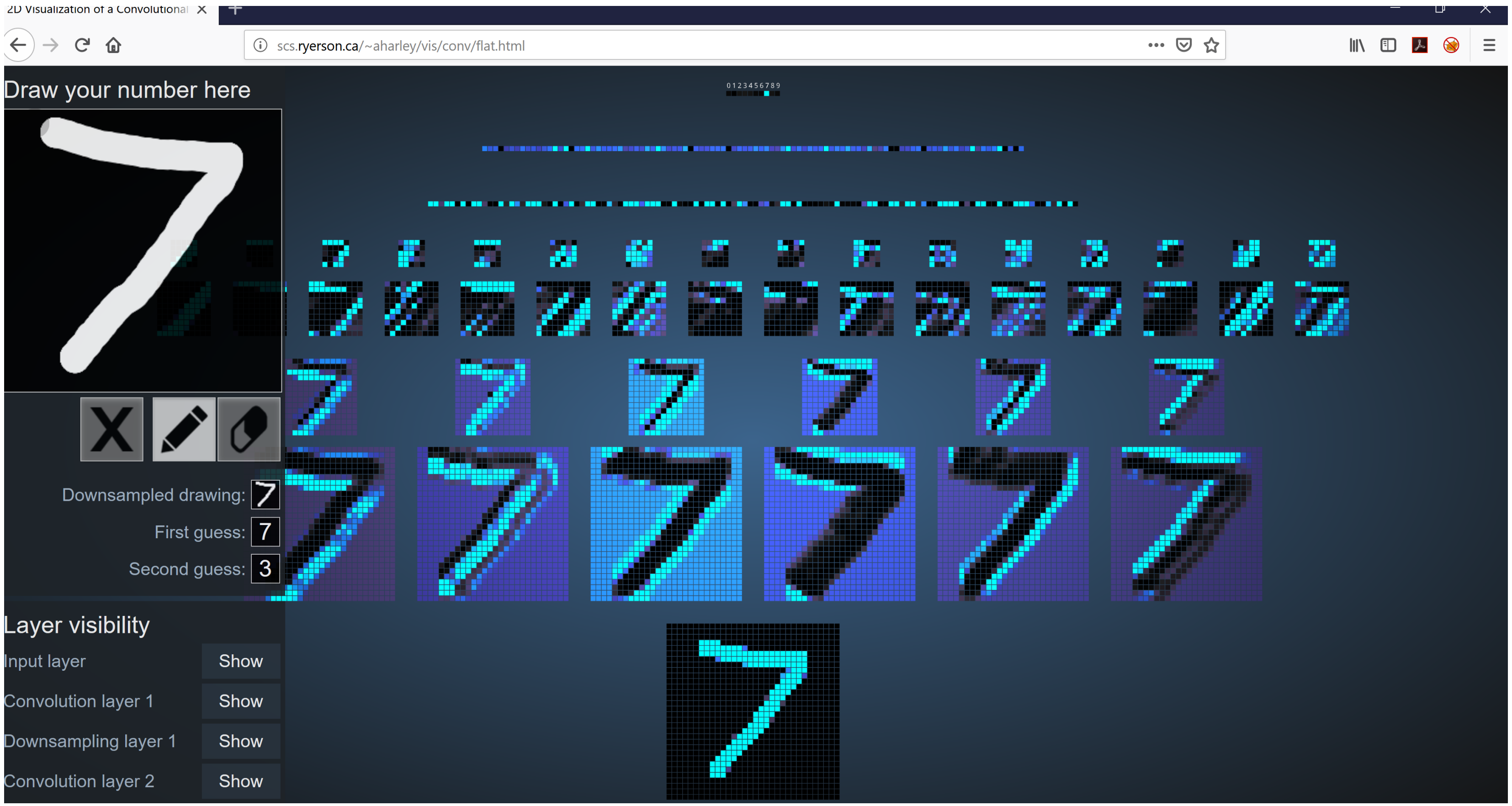




[29] M. Görner et al.

(Convolution Example)



# Understanding Feature Maps & Convolutions Summary – Online Web Tool




[30] Harley, A.W., *An Interactive Node-Link Visualization of Convolutional Neural Networks*

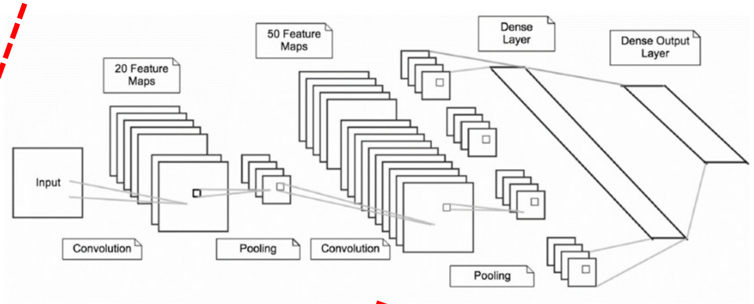
# MNIST Dataset – Convolutional Neural Network (CNN) Model

```
from keras import backend as K
from keras.models import Sequential
from keras.layers.convolutional import Conv2D
from keras.layers.convolutional import MaxPooling2D
from keras.layers.core import Activation
from keras.layers.core import Flatten
from keras.layers.core import Dense
from keras.datasets import mnist
from keras.utils import np_utils
from keras.optimizers import SGD, RMSprop, Adam
import numpy as np
import matplotlib.pyplot as plt
```

```
#define the CNN model
class CNN:
  @staticmethod
  def build(input_shape, classes):
    model = Sequential()
    # CONV => RELU => POOL
    model.add(Conv2D(20, kernel_size=5, padding="same",
    input_shape=input_shape))
    model.add(Activation("relu"))
    model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
    # CONV => RELU => POOL
    model.add(Conv2D(50, kernel_size=5, border_mode="same"))
    model.add(Activation("relu"))
    model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
    # Flatten => RELU layers
    model.add(Flatten())
    model.add(Dense(500))
    model.add(Activation("relu"))
    # a softmax classifier
    model.add(Dense(classes))
    model.add(Activation("softmax"))
    return model
```

- **Increasing the number of filters learned to 50 in the next layer from 20 in the first layer**
- **Increasing the number of filters in deeper layers is a common technique in deep learning architecture modeling**
- **Flattening the output as input for a Dense layer (fully connected layer)**
- **Fully connected / Dense layer responsible with softmax activation for classification based on learned filters and features**

*[31] A. Gulli et al.*




```
# initialize the optimizer and model
model = CNN.build(input_shape=INPUT_SHAPE, classes=NB_CLASSES)
model.compile(loss="categorical_crossentropy", optimizer=OPTIMIZER,
metrics=["accuracy"])
```

```
# printout a summary of the model to understand model complexity
model.summary()
```

| Layer (type) | Output Shape | Param # |
|---|---|---|
| conv2d_1 (Conv2D) | (None, 20, 28, 28) | 520 |
| activation_1 (Activation) | (None, 20, 28, 28) | 0 |
| max_pooling2d_1 (MaxPooling2 | (None, 20, 14, 14) | 0 |
| conv2d_2 (Conv2D) | (None, 50, 14, 14) | 25050 |
| activation_2 (Activation) | (None, 50, 14, 14) | 0 |
| max_pooling2d_2 (MaxPooling2 | (None, 50, 7, 7) | 0 |
| flatten_1 (Flatten) | (None, 2450) | 0 |
| dense_1 (Dense) | (None, 500) | 1225500 |
| activation_3 (Activation) | (None, 500) | 0 |
| dense_2 (Dense) | (None, 10) | 5010 |
| activation_4 (Activation) | (None, 10) | 0 |

```
Total params: 1,256,080
Trainable params: 1,256,080
Non-trainable params: 0
```

# MNIST Dataset – Model Parameters & 2D Input Data

```python
# parameter setup
NB_EPOCH = 20
BATCH_SIZE = 128
VERBOSE = 1
OPTIMIZER = Adam()
VALIDATION_SPLIT=0.2
IMG_ROWS, IMG_COLS = 28, 28 # input image dimensions
NB_CLASSES = 10 # number of outputs = number of digits
INPUT_SHAPE = (1, IMG_ROWS, IMG_COLS)
```

```python
# data: shuffled and split between train and test sets
(X_train, y_train), (X_test, y_test) = mnist.load_data()
K.set_image_dim_ordering("th")
# consider them as float and normalize
X_train = X_train.astype('float32')
X_test = X_test.astype('float32')
X_train /= 255
X_test /= 255
# we need a 60K x [1 x 28 x 28] shape as input to the CONVNET
X_train = X_train[:, np.newaxis, :, :]
X_test = X_test[:, np.newaxis, :, :]
print(X_train.shape[0], 'train samples')
print(X_test.shape[0], 'test samples')
# convert class vectors to binary class matrices
y_train = np_utils.to_categorical(y_train, NB_CLASSES)
y_test = np_utils.to_categorical(y_test, NB_CLASSES)
```

- **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)**

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

- **Compared to the Multi-Output Perceptron and Artificial Neural Networks (ANN) model, the input dataset remains as 2d matrice with 1 x 28 x 28 per image, including also the class vectors that are converted to binary class matrices**

# MNIST Dataset – CNN Model Output & Evaluation

```
Epoch 14/20
48000/48000 [==============================] - 4s 88us/step - loss: 0.0065 - acc: 0.9980 - val_loss: 0.0346 - val_acc: 0.9921
Epoch 15/20
48000/48000 [==============================] - 4s 89us/step - loss: 0.0030 - acc: 0.9990 - val_loss: 0.0418 - val_acc: 0.9903
Epoch 16/20
48000/48000 [==============================] - 4s 88us/step - loss: 0.0057 - acc: 0.9980 - val_loss: 0.0470 - val_acc: 0.9910
Epoch 17/20
48000/48000 [==============================] - 4s 88us/step - loss: 0.0043 - acc: 0.9985 - val_loss: 0.0440 - val_acc: 0.9906
Epoch 18/20
48000/48000 [==============================] - 4s 88us/step - loss: 0.0046 - acc: 0.9985 - val_loss: 0.0474 - val_acc: 0.9891
Epoch 19/20
48000/48000 [==============================] - 4s 88us/step - loss: 0.0047 - acc: 0.9986 - val_loss: 0.0353 - val_acc: 0.9928
Epoch 20/20
48000/48000 [==============================] - 4s 88us/step - loss: 3.4055e-04 - acc: 1.0000 - val_loss: 0.0374 - val_acc: 0.9927
```

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

```
10000/10000 [==============================] - 1s 70us/step
Test score: 0.0303058747581508
Test accuracy: 0.9936
```

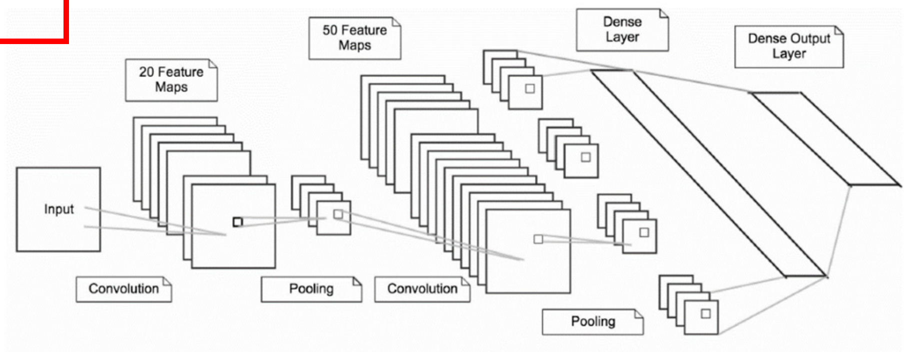
Why not 100%

?

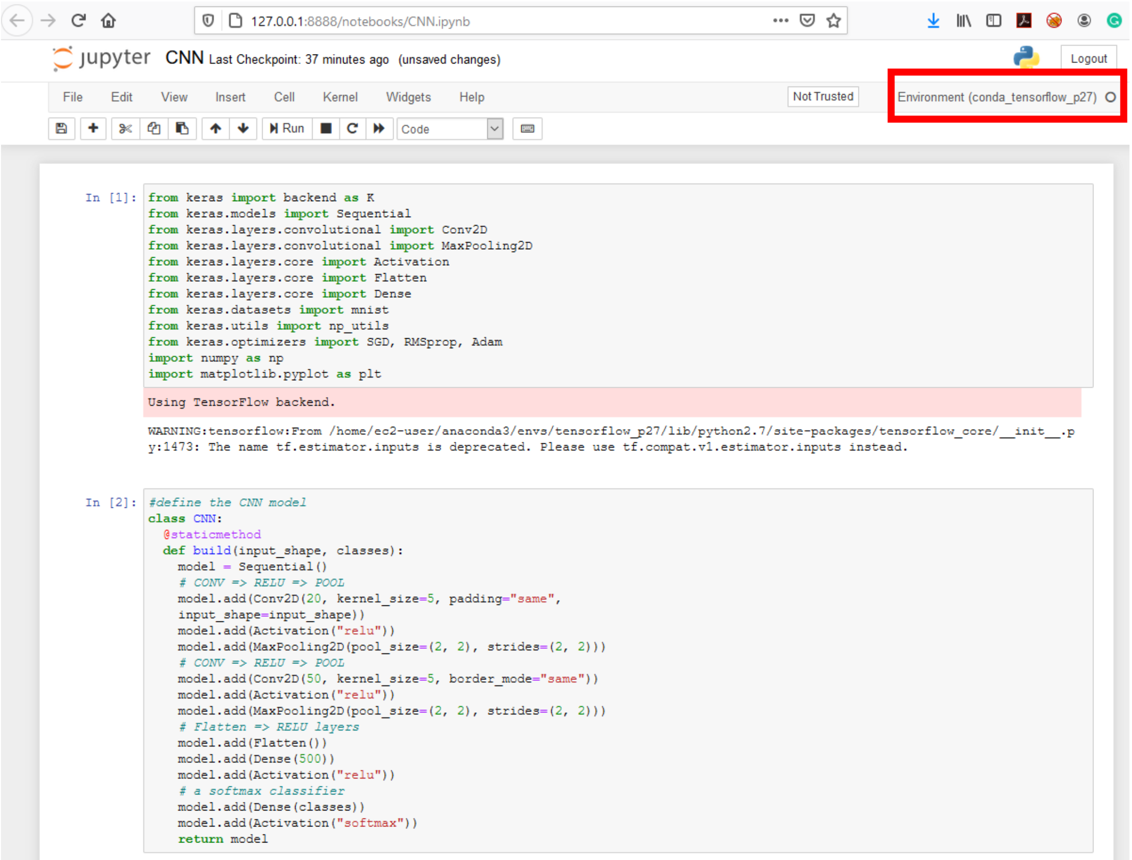
some samples even for a human unrecognizable

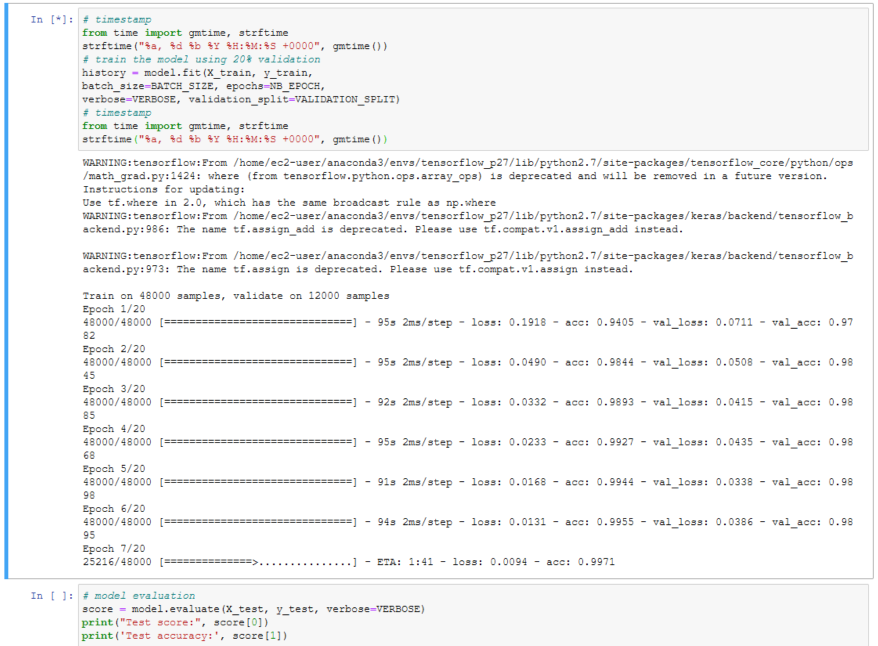
- ✓ **Multi Output Perceptron: ~91,01% (20 Epochs)**
- ✓ **ANN 2 Hidden Layers: ~95,14 % (20 Epochs)**
- ✓ **CNN Deep Learning Model: ~99,36 % (20 Epochs)**

*[31] A. Gulli et al.*



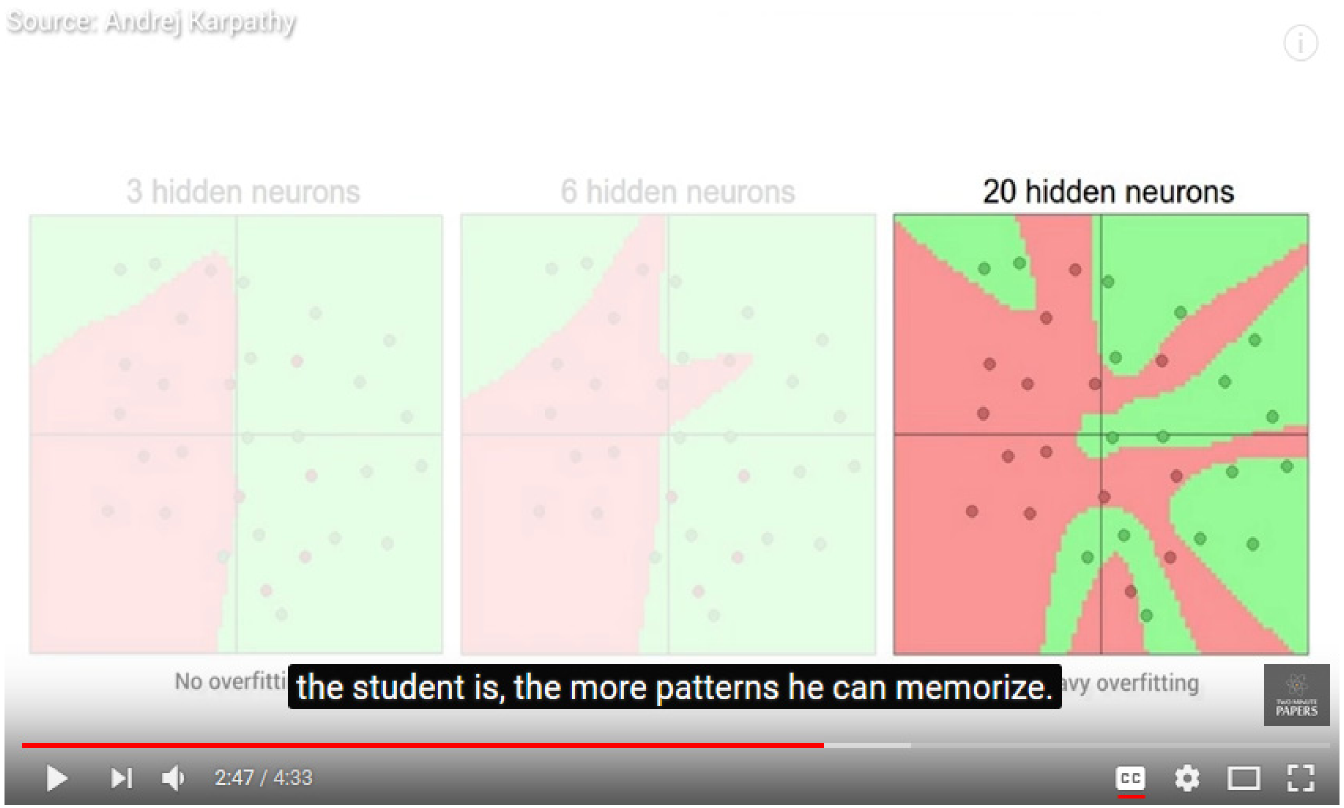
# Running a Deep Learning Model with Convolutional Neural Network (CNN)



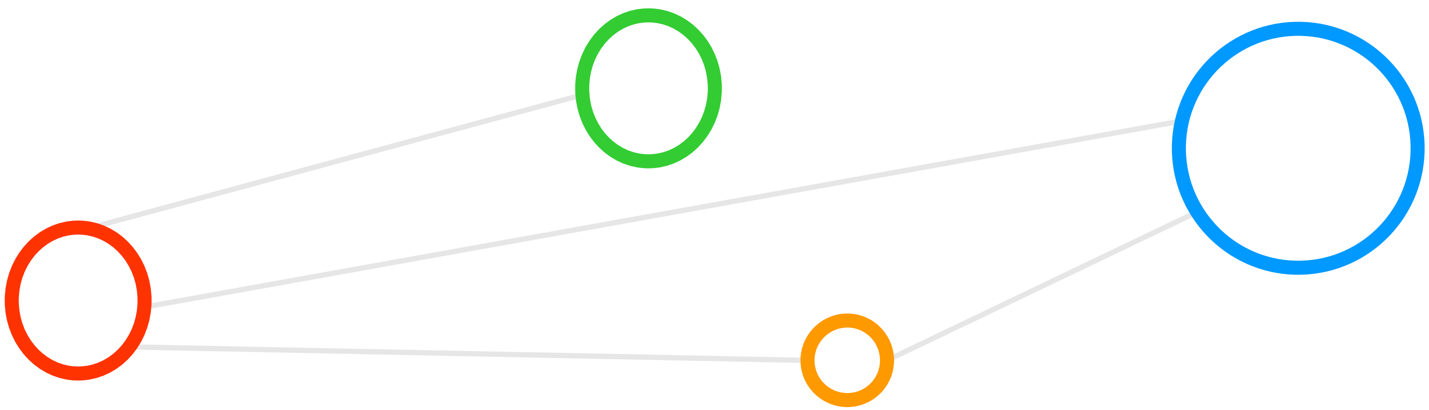

- **Using Deep Learning Techniques such as Convolutional Neural Networks (CNNs) in clouds can lead to significant improvements in accuracy, but also to significant longer run-times than traditional Artificial Neural Networks (ANNs) and are thus much more costly in clouds**
- **Using CPU resources for deep learning techniques is usually not recommended**

# [Video for further Studies] Overfitting in Deep Neural Networks



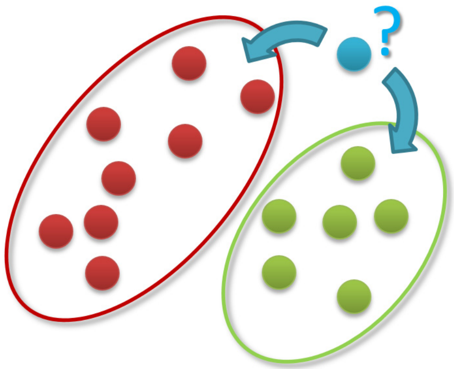

*[33] YouTube Video, Overfitting and Regularization For Deep Learning*

# Selected Parallel & Scalable Machine & Deep Learning Techniques



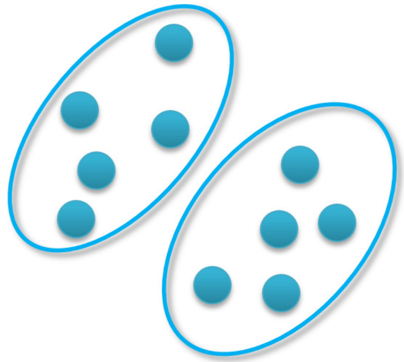
# Machine Learning Models – Understanding Parallel Benefits

## Classification



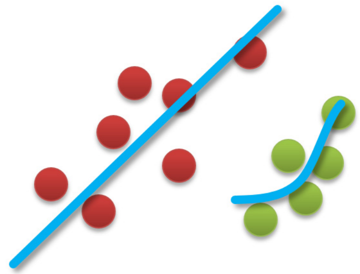
- Groups of data exist
- New data classified to existing groups

## 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

- **Machine learning methods can be roughly categorized in classification, clustering, or regression augmented with various techniques for data exploration, selection, or reduction – despite the momentum of deep learning, traditional machine learning algorithms are still widely relevant today**

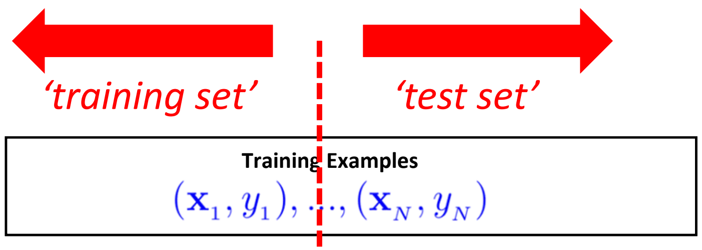
*[1] www.big-data.tips, 'Data Classification'*

# Terminologies & Different Dataset Elements & Processes – Overview

- Machine Learning Models
  - Based on various algorithms that learn from existing data sets
- Labelled Dataset (samples)
  - 'in-sample' data given to us: $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)$
- Learning vs. Memorizing
  - The goal is to create a system that works well 'out of sample'
  - In other words we want to classify 'future data' (ouf of sample) correct
- Dataset Part One: Training set
  - Used for training a machine learning algorithms
  - Result after using a training set: a trained system
- Dataset Part Two: Test set
  - Used for testing whether the trained system might work well
  - Result after using a test set: accuracy of the trained model

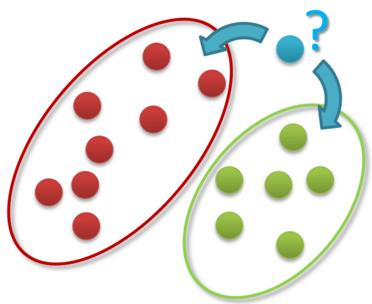
(exact separation is rule of thumb, but different in each data analysis case: e.g., 10% training data, 90% test data)

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




(historical records, groundtruth data, examples)

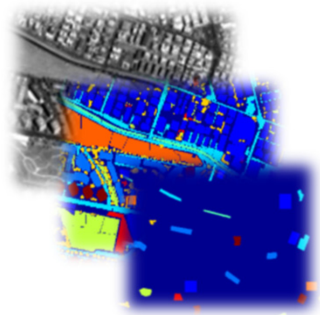
## Classification



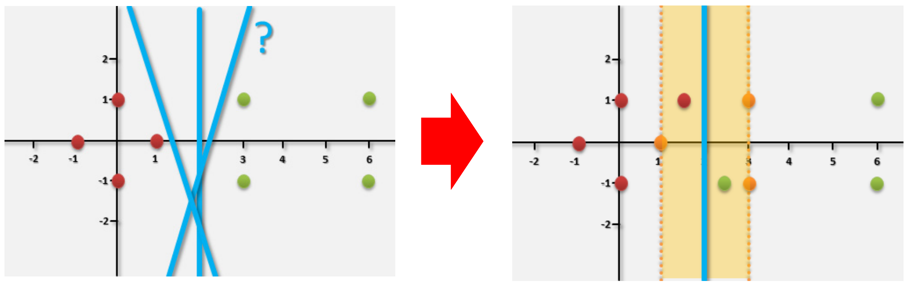

$(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)$

# Parallel and Scalable Machine Learning – Parallel Support Vector Machine (SVM)

- ‘Different kind‘ of parallel algorithms
  - ‘learn from data‘ instead of modelling/approximate reality with physics
  - 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 |




[35] C. Cortes & V. Vapnik, ‘Support Vector Networks‘, Machine Learning, 1995

```
#!/bin/bash -x
#SBATCH--nodes=4
#SBATCH--ntasks=96
#SBATCH--ntasks-per-node=24
#SBATCH--output=mpi-out.%j
#SBATCH--error=mpi-err.%j
#SBATCH--time=04:00:00
#SBATCH--partition=batch
#SBATCH--mail-user=m.riedel@fz-juelich.de
#SBATCH--mail-type=ALL
#SBATCH--job-name=train-indianpines-4-96-24
#SBATCH--reservation=ml-hpc-2

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

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

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

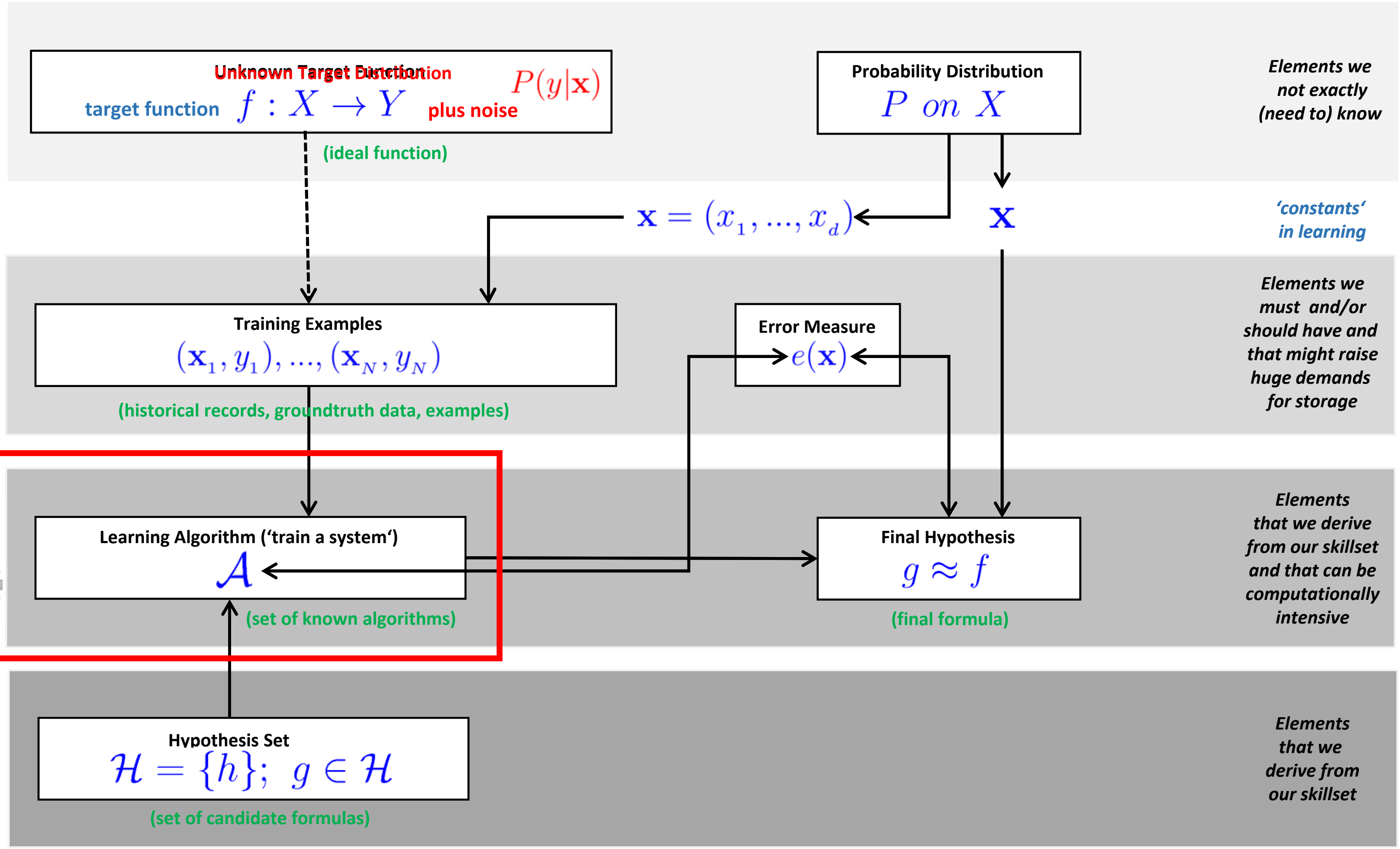
```
#!/bin/bash -x
#SBATCH--nodes=4
#SBATCH--ntasks=96
#SBATCH--ntasks-per-node=24
#SBATCH--output=mpi-out.%j
#SBATCH--error=mpi-err.%j
#SBATCH--time=04:00:00
#SBATCH--partition=batch
#SBATCH--mail-user=m.riedel@fz-juelich.de
#SBATCH--mail-type=ALL
#SBATCH--job-name=pred-indianpines-4-96-24
#SBATCH--reservation=ml-hpc-2

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

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

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

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

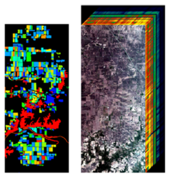
[34] G. Cavallaro & M. Riedel & J.A. Benediktsson et al., ‘On Understanding Big Data Impacts in Remotely Sensed Image Classification Using Support Vector Machine Methods‘, Journal of Applied Earth Observations and Remote Sensing, 2015

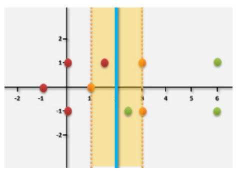
[36] www.big-data.tips, ‘SVM Train‘

# Overview Machine Learning Terminologies & Computing-intensive Processes

?

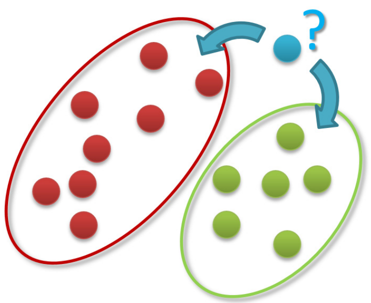


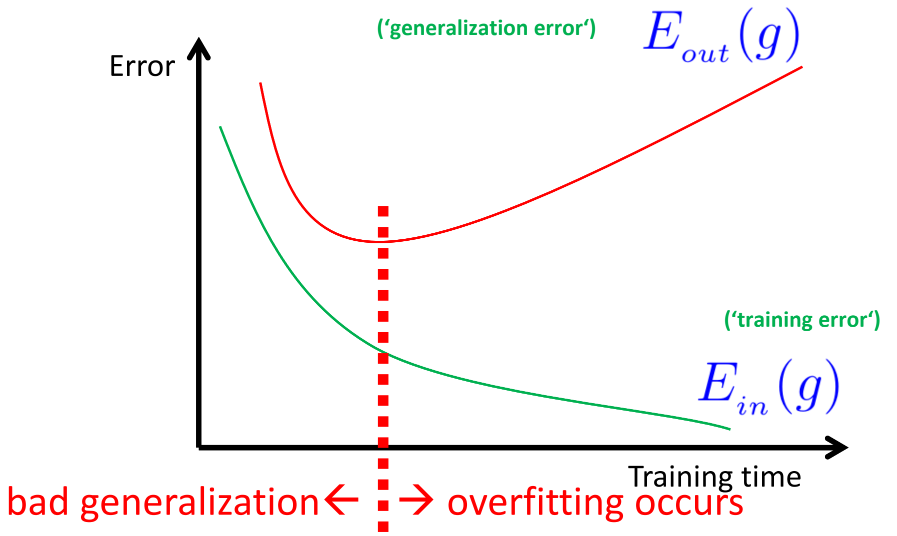


# Problem of Overfitting – Clarifying Terms

## Classification



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




- A good model must have low training error ($E_{in}$) and low generalization error ($E_{out}$)
- Model overfitting is if a model fits the data too well ($E_{in}$) with a poorer generalization error ($E_{out}$) than another model with a higher training error ($E_{in}$)
- The two general approaches to prevent overfitting are (1) validation and (2) regularization

*[37] www.big-data.tips, 'Generalization in Machine Learning'*

# Validation Technique – Proper Model Selection Process is Compute-Intensive

**Hypothesis Set**
$\mathcal{H} = \{h\};\ g \in \mathcal{H}$

(set of candidate formulas across models)

- Many different models
  Use validation error to perform select decisions
- Careful consideration:
  - 'Picked means decided' hypothesis has already bias (→ contamination)
  - Using $\mathcal{D}_{Val}$ M times

**Final Hypothesis**
$g_{m*} \approx f$

(test this on unseen data good, but depends on availability in practice)

(training not on full data set)

$\mathcal{D}_{Train}$

$\mathcal{H}_1$ $\mathcal{H}_2$ $\mathcal{H}_M$

(training)

$g_1$ $g_2$ $g_M$

(out-of-sample w.r.t. $D_{Train}$)

$\mathcal{D}_{Val}$

(validate)

(unbiased estimates)

$E_{val_1}$ $E_{val_2}$ $E_{val_M}$

(pick 'best' → bias)

(decides model selection)

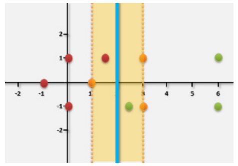
$\mathcal{H}_{m*}\ E_{val_{m*}}$

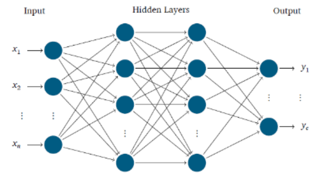
$\mathcal{D}$

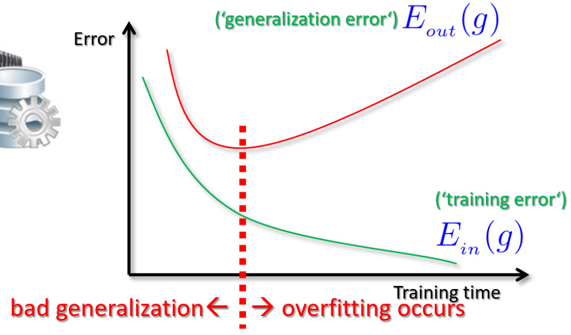
(final training on full set, use the validation samples too)

(final real training to get even better out-of-sample)

$g_{m*}$






- **Model selection is choosing (a) different types of models or (b) parameter values inside models**
- **Model selection takes advantage of the validation error in order to decide → 'pick the best'**

# Validation Technique – Cross-Validation – Leave-more-out

- 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
  - 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)

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

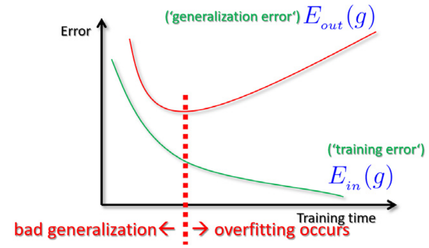
(generalization to leave k points out at each run)

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

(fewer training sessions than above)

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

- 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)




$\mathcal{D}$ (dataset)

Training Examples

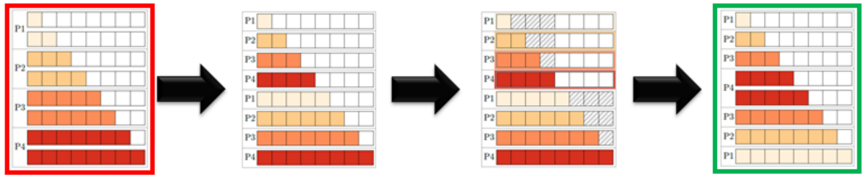
| $\mathcal{D}_1$ | $\mathcal{D}_2$ | $\mathcal{D}_3$ | $\mathcal{D}_4$ | $\mathcal{D}_5$ | $\mathcal{D}_6$ | $\mathcal{D}_7$ | $\mathcal{D}_8$ | $\mathcal{D}_9$ | $\mathcal{D}_{10}$ |
|---|---|---|---|---|---|---|---|---|---|

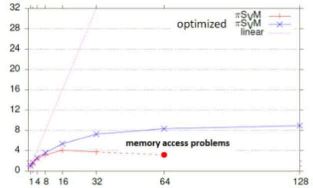
(involved in training now) (involved in training now)

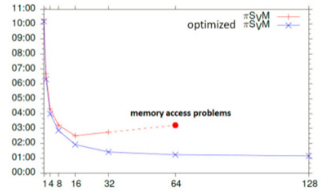
*[38] www.big-data.tips, ‘Cross Validation‘*

# Parallel Support Vector Machine (SVM) – piSVM MPI Implementation & Impact

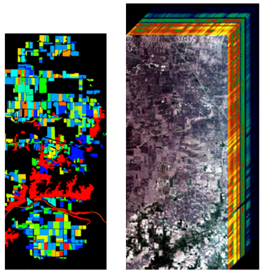
- Original piSVM 1.2 version (2011)
  - Open-source and based on libSVM library, C
  - Message Passing Interface (MPI)
  - New version appeared 2014-10 v. 1.3 (no major improvements)
  - Lack of ‘big data‘ support (e.g. memory)
- Tuned scalable parallel piSVM tool 1.2.1
  - Highly scalable version maintained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




πSVM



```
//Jeder Prozess hat l Werte an die Stelle Rang * l in p_cache_status geschrieben
for(int k = 0; k < p; ++k)
{
    //Jeder Prozess broadcastet sein Ergebnis zu allen anderen Prozessen
    MPI_Bcast(&p_cache_status[k * l], l, MPI_CHAR, k, comm);
}
```

Using MPI_Allgather() instead

```
for(int i = 0; i < p; ++i) {
    if(rank == i) { //Der sendende Prozess kopiert in den Sendebereich
        for(int j = 0; j < lmn; ++j)
            G_buf[j] = G_n[j];
    }
    //Alle anderen Prozesse erhalten die Daten
    MPI_Bcast(G_buf, lmn, MPI_DOUBLE, i, comm);
    //Und addieren sie auf
    for(int j = 0; j < lmn; ++j)
        G[not_work_set[j]] += G_buf[j];
}
```

Using MPI_Allreduce() instead

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

| $\gamma$/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) |

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

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

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


*[39] piSVM on SourceForge, 2008*

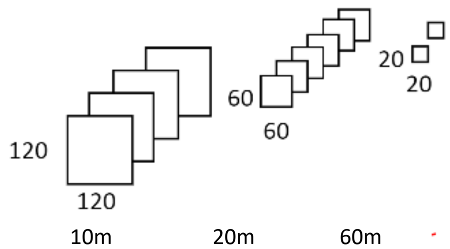
*[34] G. Cavallaro & M. Riedel & J.A. Benediktsson et al., ‘On Understanding Big Data Impacts in Remotely Sensed Image Classification Using Support Vector Machine Methods’, Journal of Applied Earth Observations and Remote Sensing*

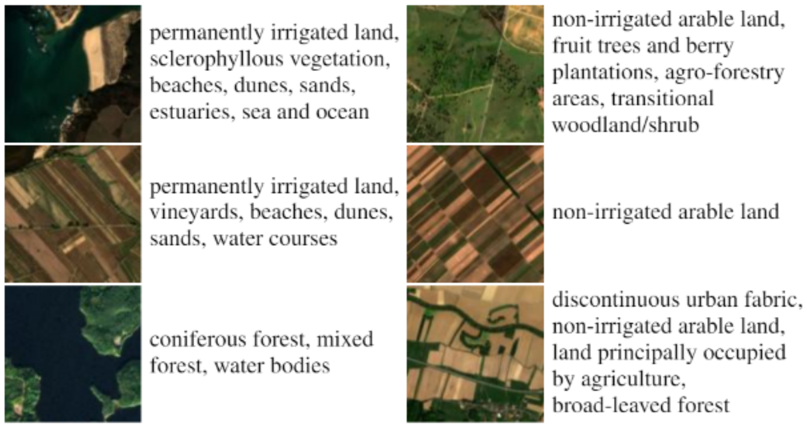
# Multispectral Remote Sensing Dataset Example used in Deep Learning

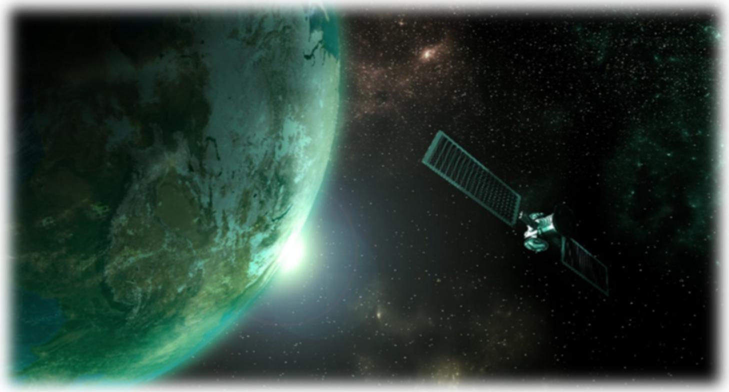
| Datasets | Image type | Image per class | Scene classes | Annotation type | Total images | Spatial resolution (m) | Image sizes | Year |
|---|---|---|---|---|---|---|---|---|
| BigEarthNet | Satellite MS | 328 to 217119 | 43 | Multi label | 590,326 | 10<br>20<br>60 | 120x120<br>60x60<br>20x20 | 2018 |

*[40] G. Sumbul et al.*

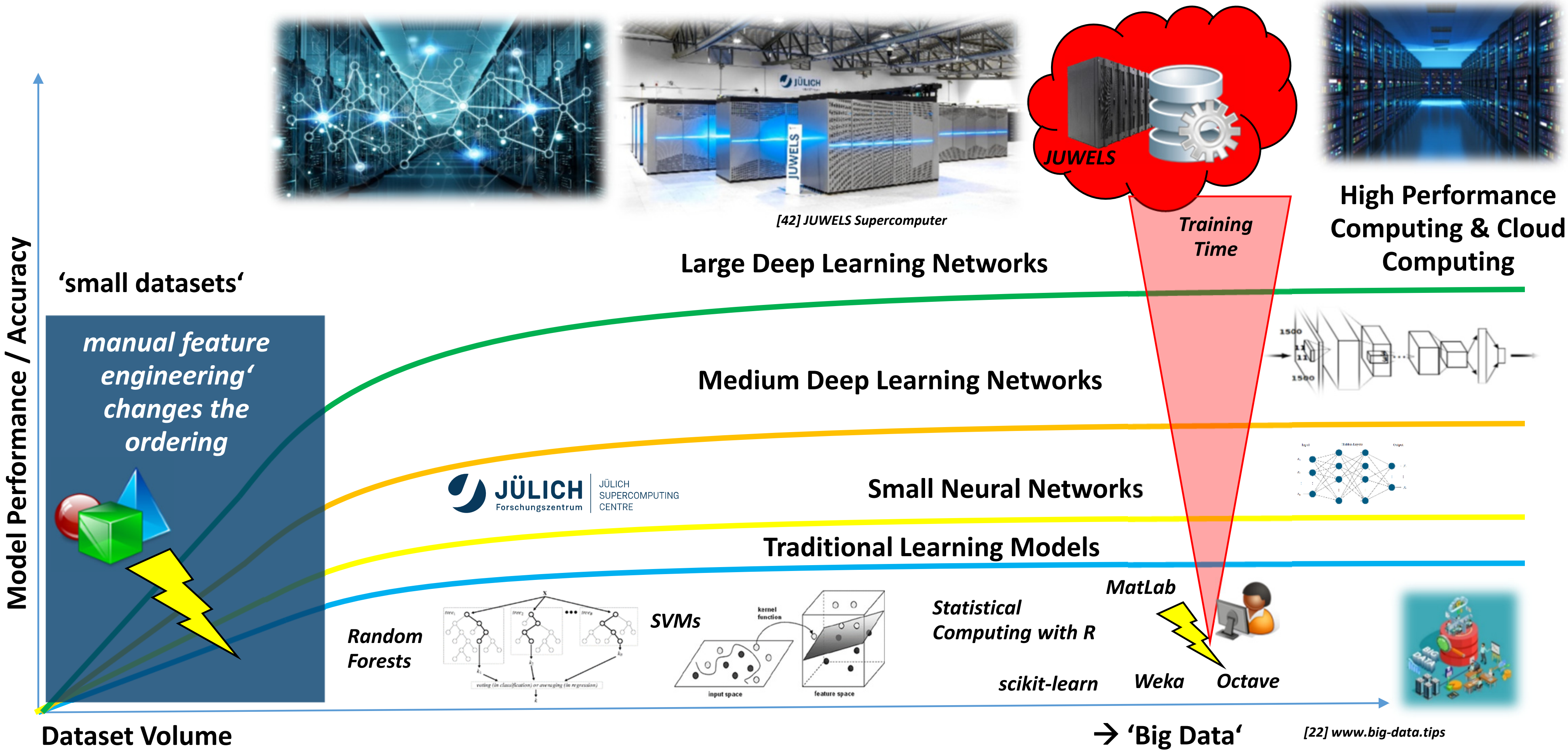
*[41] Big Earth Net Dataset*



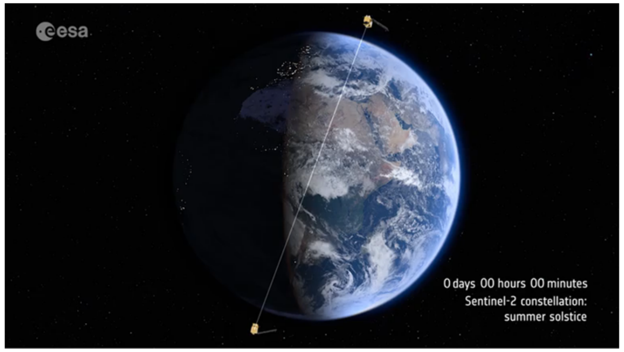



# HPC Relationship to 'Big Data' in Machine & Deep Learning – Scalability

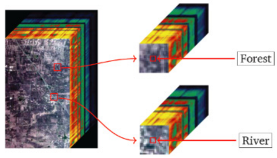


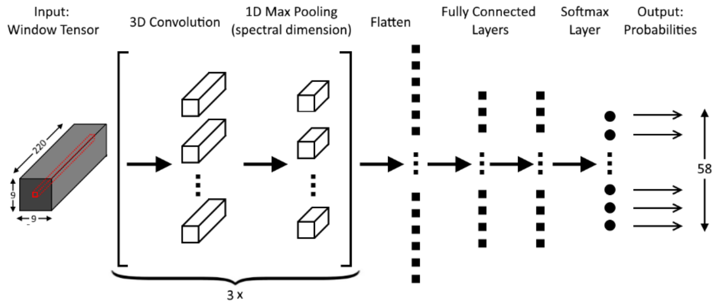
# Deep Learning Application Example – Using High Performance Computing




- Using Convolutional Neural Networks (CNNs) with hyperspectral remote sensing image data

*[43] J. Lange and M. Riedel et al., IGARSS Conference, 2018*

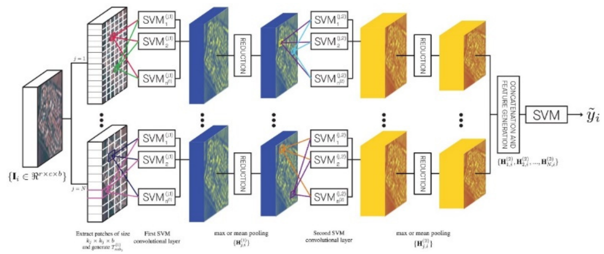



| Feature | Representation / Value |
|---|---|
| Conv. Layer Filters | 48, 32, 32 |
| Conv. Layer Filter size | (3,3,5), (3,3,5), (3,3,5) |
| Dense Layer Neurons | 128, 128 |
| Optimizer | SGD |
| Loss Function | mean squared error |
| Activation Functions | ReLU |
| Training Epochs | 600 |
| Batch Size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

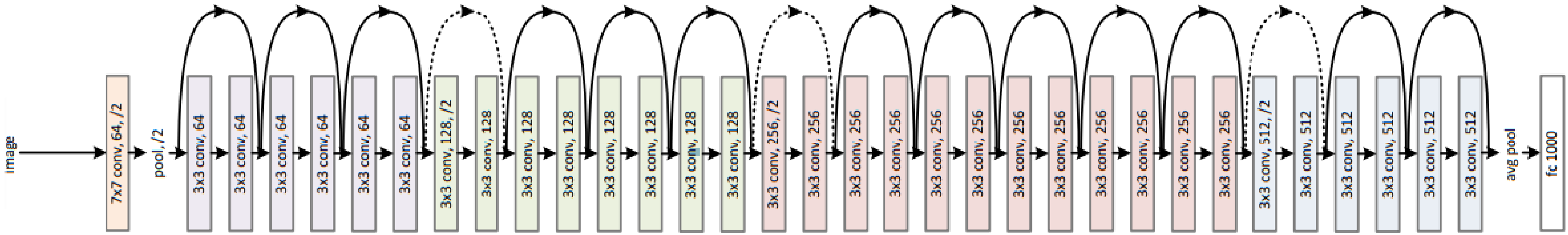
*[44] G. Cavallaro, M. Riedel et al., IGARSS 2019*

- Find Hyperparameters & joint 'new-old' modeling & transfer learning given rare labeled/annotated data in science (e.g. 36,000 vs. 14,197,122 images ImageNet)



# Deep Learning via RESNET-50 Architecture – A Case for interconnecting GPUs

- Classification of land cover in scenes in Remote Sensing
  - Very suitable for parallelization via distributed training on multi GPUs

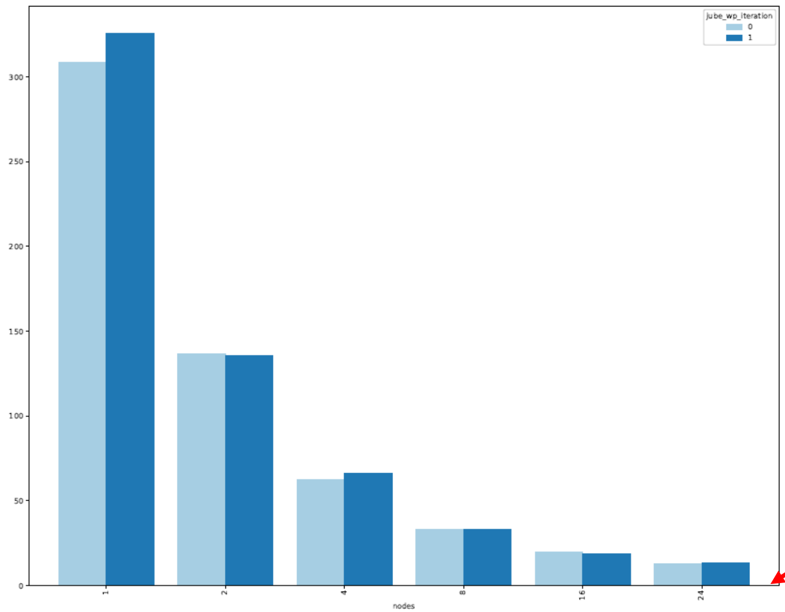



*[45] RESNET*

- **RESNET-50 is a known neural network architecture that has established a strong baseline in terms of accuracy**
- **The computational complexity of training the RESNET-50 architecture relies in the fact that is has ~ 25.6 millions of trainable parameters**
- **RESNET-50 still represents a good trade-off between accuracy, depth and number of parameters**
- **The setups of RESNET-50 makes it very suitable for parallelization via distributed training on multi GPUs**

# Distributed Training with Multi GPU Usage using Horovod

Time per epoch [sec]





A partition of the JUWELS system
has 56 compute nodes,
each with 4 NVIDIA V100 GPUs
(equipped with 16 GB of memory)

24 nodes x 4 GPUs = 96 GPUs

- Horovod is a distributed training framework used in combination with low-level deep learning frameworks like Tensorflow
- Horovod uses MPI for inter-process communication, e.g., MPI_Allreduce()
- Distributed training using data parallelism approach means: (1) Gradients for different batches of data are calculated separately on each node; (2) But averaged across nodes to apply consistent updated to the deep learning model in each node

Other distributed training
approaches possible with DeepSpeed

*[46] DeepSpeed*

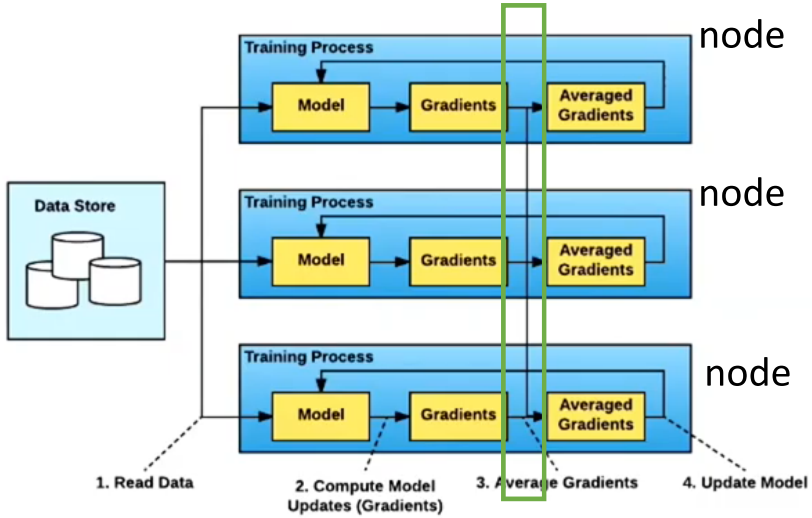
DeepSpeed is a deep learning optimization library that makes distributed training easy, efficient, and effective.

*10x Larger Models*

*10x Faster Training*

*Minimal Code Change*

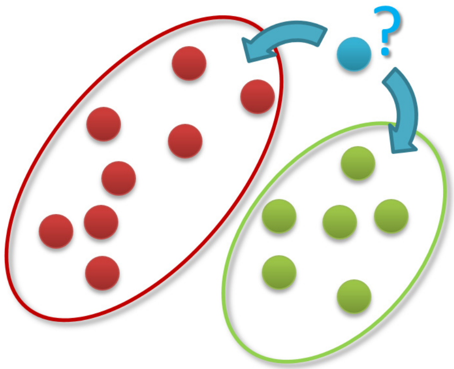
Horovod distributed training via MPI_Allreduce()




*[47] Horovod*

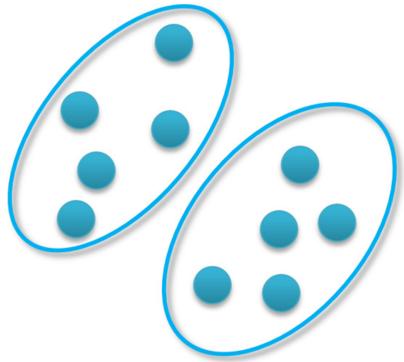
# Machine Learning Models – Short Overview & Introduction to Classification

## Classification



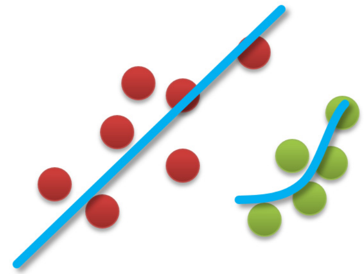

- Groups of data exist
- New data classified to existing groups

## Clustering



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

## Regression

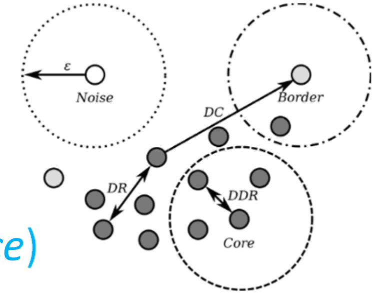


- Identify a line with a certain slope describing the data

- **Machine learning methods can be roughly categorized in classification, clustering, or regression augmented with various techniques for data exploration, selection, or reduction – despite the momentum of deep learning, traditional machine learning algorithms are still widely relevant today**

# Data Science Example: DBSCAN Clustering Algorithm

- DBSCAN Algorithm
  - 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 (cf. K-Means Clustering with K clusters)
  - Forms arbitrarily shaped clusters (no ‘bow ties‘)
  - Identifies inherently also outliers/noise




(MinPoints = 4)

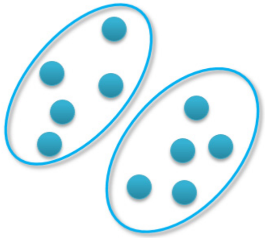
(DR = Density Reachable)

(DDR = Directly Density Reachable)

(DC = Density Connected)

*[49] Ester et al.*

**Clustering**

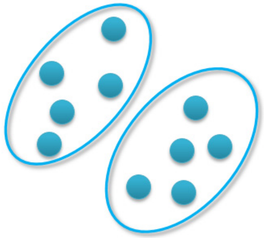


```
#!/bin/bash
#SBATCH --job-name=HPDBSCAN
#SBATCH -o HPDBSCAN-%j.out
#SBATCH -e HPDBSCAN-%j.err
#SBATCH --nodes=2
#SBATCH --ntasks=4
#SBATCH --ntasks-per-node=4
#SBATCH --time=00:20:00
#SBATCH --cpus-per-task=4
#SBATCH --reservation=ml-hpc-1

export OMP_NUM_THREADS=4

# location executable
HPDBSCAN=/homea/hpclab/train001/tools/hpdbscan/dbscan

# your own copy of bremen small
BREMENSMALLDATA=/homea/hpclab/train001/bremenSmall.h5

# your own copy of bremen big
BREMENBIGDATA=/homea/hpclab/train001/bremen.h5

srun $HPDBSCAN -m 100 -e 300 -t 12 $BREMENSMALLDATA
```

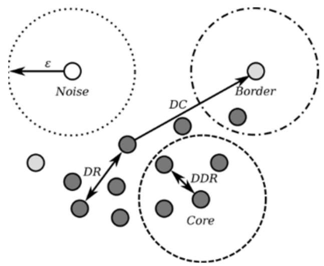
- **Density-based spatial clustering of applications with noise (DBSCAN) is a data clustering algorithm that requires only two parameters and has no requirement to specify number of clusters**
- **Parameter Epsilon: Algorithm looks for a similar point within a given search radius Epsilon**
- **Parameter minPoints: Algorithm checks that cluster consist of a given minimum number of points**

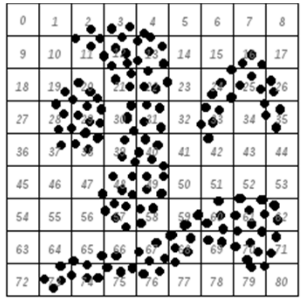
*[48] M. Goetz and M. Riedel et al,*
*Proceedings IEEE Supercomputing Conference, 2015*

# Data Parallelism Example: Smart Domain Decomposition in Data Sciences

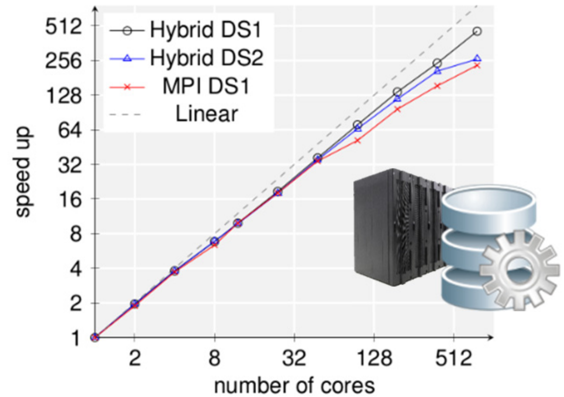


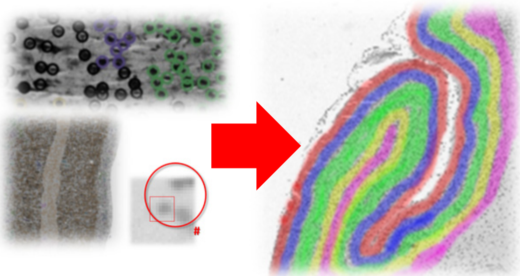
**Clustering**



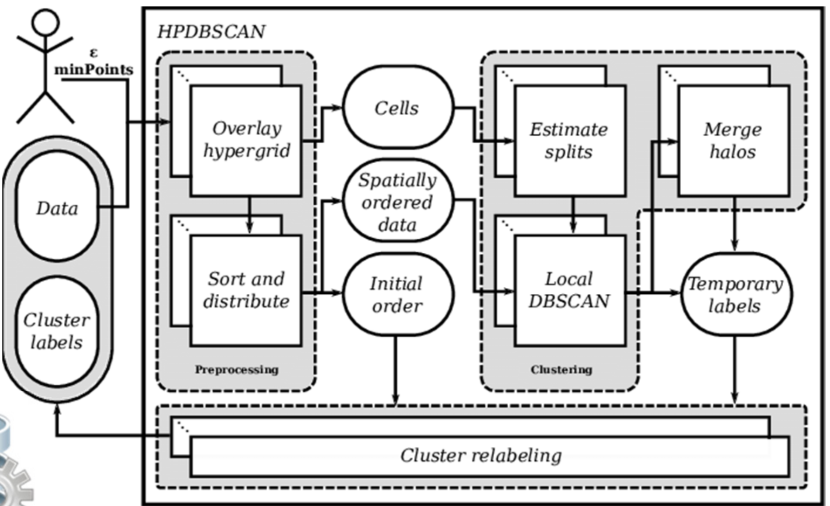
**Overlayed spatial grid**



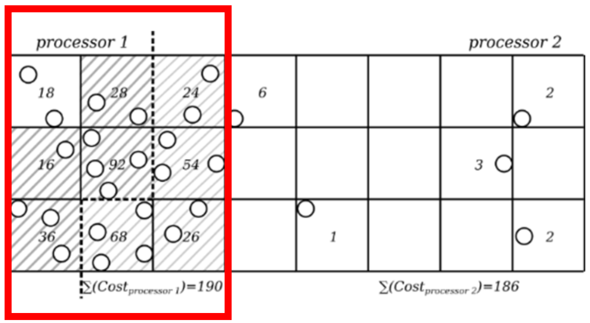



[48] M. Goetz and M. Riedel et al, Proceedings IEEE Supercomputing Conference, 2015

Algorithm can be used in many application domains




cluster merge across halo regions/layers

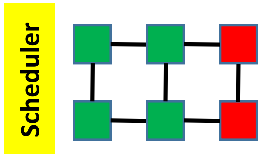


# HPDBSCAN Clustering – Using Parallel File Formats & File Systems

```
[morris@jotunn hpdbscan]$ more HPDBSCAN-199947.out
Calculating Cell Space...
        Computing Dimensions... [OK] in 0.054227
        Computing Cells...      [OK] in 0.022971
        Sorting Points...       [OK] in 0.133213
        Distributing Points...  [OK] in 0.111046
DBSCAN...
        Local Scan...           [OK] in 139.057375
        Merging Neighbors...    [OK] in 0.010089
        Adjust Labels ...       [OK] in 0.010476
        Rec. Init. Order ...    [OK] in 0.556270
        Writing File ...        [OK] in 0.170748
Result...
        21      Clusters
        2974394 Cluster Points
        25606   Noise Points
        2949094 Core Points
Took: 140.453795s
```

```
[morris@jotunn hpdbscan]$ h5dump -d /Clusters bremenSmall.h5
HDF5 "bremenSmall.h5" {
DATASET "/Clusters" {
   DATATYPE  H5T_STD_I64LE
   DATASPACE  SIMPLE { ( 3000000 ) / ( 3000000 ) }
   DATA {
   (0): 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
   (23): 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
   (46): 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
   (69): 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
   (92): 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
   (115): 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
```

- **The input data for the parallel & scalable HPDBSCAN clustering algorithm is a HDF5 file and all the processors read in parallel chunks of the data**
- **The HDF5 file before the execution of HPDBSCAN has 0 as Cluster Ids for its specific initialization**




Jötunn compute nodes

```
[morris@jotunn hpdbscan]$ h5dump -d /Clusters bremenSmall.h5
HDF5 "bremenSmall.h5" {
DATASET "/Clusters" {
   DATATYPE  H5T_STD_I64LE
   DATASPACE  SIMPLE { ( 3000000 ) / ( 3000000 ) }
   DATA {
   (0): -45205, -45205, -45205, -45205, -45205, -45205, -45205, -45205,
   (8): -45205, -45205, -45205, -45205, -45205, 4825, -45205, -45205, -45205,
   (17): -45205, -4108, -45205, -45205, -45205, -45205, -45205, -45205,
   (25): -45205, -45205, -45205, -45205, -45205, -45205, -45205, -45205,
   (33): -45205, -45205, -45205, -45205, -45205, -45205, -45205, -45205,
   (41): -45205, -45205, -45205, -45205, -45205, -45205, -45205, -45205,
   (49): -45205, -45205, -45205, -45205, -45205, -45205, -45205, -45205,
   (57): -45205, -45205, -45205, -45205, -45205, -45205, -45205, -45205,
   (65): -45205, -45205, -45205, -45205, -45205, -45205, -45205, -45205,
   (73): -45205, -45205, -45205, -45205, -45205, -45205, -45205, -45205,
   (81): -45205, -45205, -45205, -490063, -45205, -45205, -45205, -45205,
   (89): -45205, -45205, -45205, -45205, -45205, -45205, -45205, 45205,
   (97): -45205, -45205, -45205, -45205, -45205, -45205, -45205, -45205,
   (105): -45205, -45205, -45205, -45205, -45205, -45205, -45205, -45205,
   (113): -45205, -45205, -45205, -45205, -45205, -45205, -490063, -45205,
```

- **The standard out of the HPDBSCAN parallel & scalable DBSCAN clustering algorithm is not the result of the DBSCAN clustering algorithm and only shows meta information such as the numbers of clusters found, noise, and running time**

- **The real outcome of the parallel & scalable HPDBSCAN algorithm is directly written into the HDF5 file assigning for each point cloud data element a specific cluster ID, or using minus numbers to indicate noise points (no real clusters)**

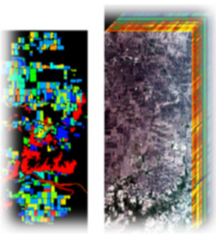

*[48] M. Goetz and M. Riedel et al,*
***Proceedings IEEE Supercomputing Conference, 2015***

# [Video for further Studies] DBSCAN Algorithm Steps & Visualization Example

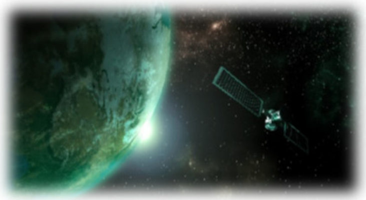

*[50] YouTube video, DBSCAN Explanation and Visualization*

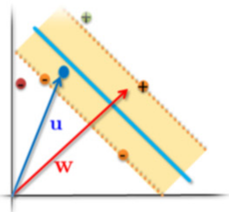
# Emerging Quantum Machine Learning – Initial Results on Quantum Annealing

- Disruptive Technology in the HPC Ecosystems
  - Different concept than traditional (super)computers
  - Information as ‘0’ or ‘1’ or both simultaneously

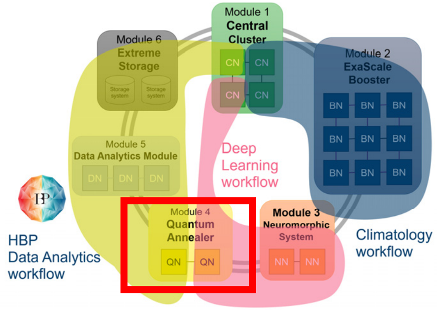



[51] D-Wave Systems YouTube Channel

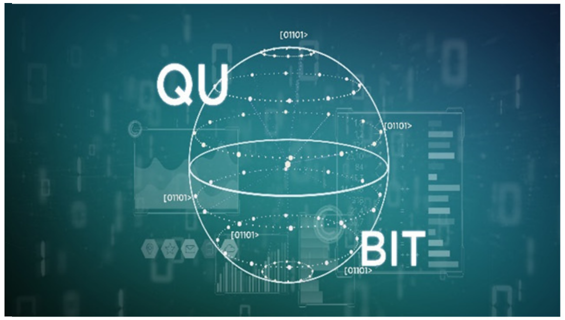


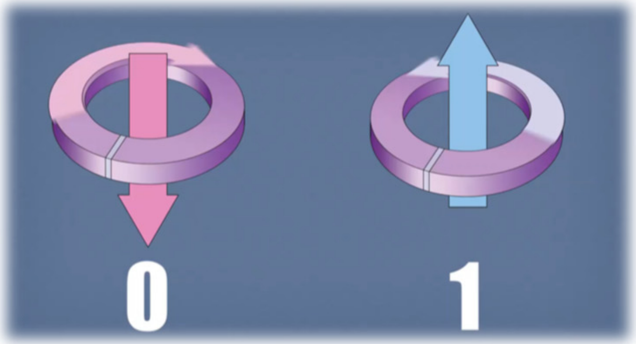



```
In [ ]: from quantum_SVM import *

# Hyperparameters
B=[2,3,5,10]
K=[2,3]
xi=[0,1,5]
gamma=[-1,0.125,0.25,0.5,1,2,4,8]
n_experiments=len(B)*len(K)*len(xi)*len(gamma)

hyperparameters=np.zeros([n_experiments,4], dtype=float)

path_data_key='input_datasets/calibration/'+id_dataset+'/'
data_key = id_dataset+'calibtrain'
path_out='outputs/calibration/'+id_dataset+'/'

trainacc=np.zeros([fold], dtype=float)
trainauroc=np.zeros([fold], dtype=float)
trainauprc=np.zeros([fold], dtype=float)
```




[52] DEEP Projects Web Page





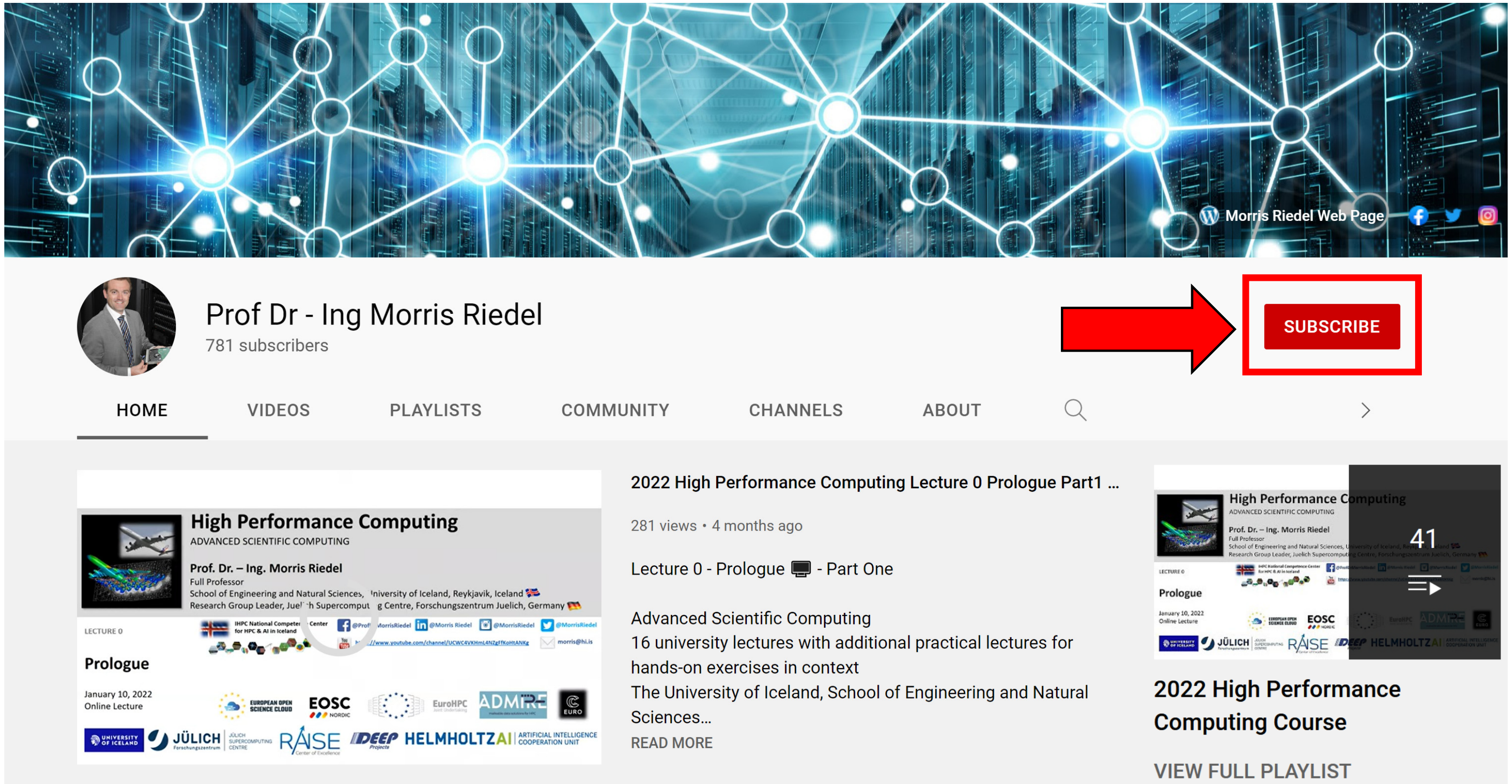
[53] M. Riedel, UTMessan 2020 YouTube Video

*[54] G. Cavallaro & M. Riedel et al., ‘Approaching Remote Sensing Image Classification with ensembles of support vector machines on the D-Wave Quantum Annealer‘*

*[55] M. Riedel, G. Cavallaro, J.A. Bendiktsson, ‘Practice and Experience in Using Parallel and Scalable Machine Learning in Remote Sensing from HPC Over Cloud to Quantum Computing‘*

*[56] Delilbasic, A., Cavallaro, G., Willsch, M., Melgani, F., Riedel, M., Michielsen, K., ‘Quantum Support Vector Machine Algorithms for Remote Sensing Data Classification‘*

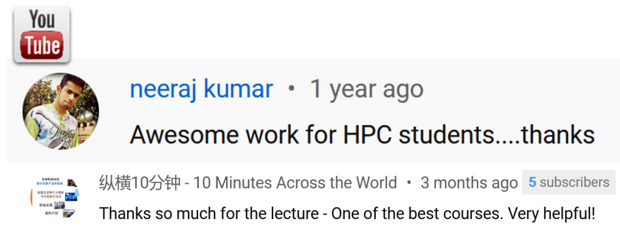
# More Information: Full HPC Spring 2022 University Course

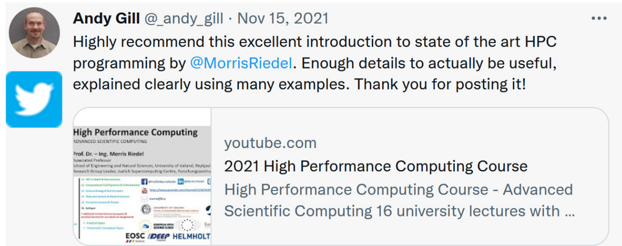



***Selected Testimonials from our community:***

*'… I have found your wonderful HPC lectures on YouTube, and I am finding them very helpful for learning the supercomputing tools necessary for my research….'*

**- student in computational nuclear physics from USA**




➢ https://www.youtube.com/channel/UCWC4VKHmL4NZgFfKoHtANKg

# Lecture Bibliography

# Lecture Bibliography (1)


- [1] www.big-data.tips, 'Data Classification', Online:
  http://www.big-data.tips/data-classification
- [2] Species Iris Group of North America Database, Online:
  http://www.signa.org
- [3] UCI Machine Learning Repository Iris Dataset, Online:
  https://archive.ics.uci.edu/ml/datasets/Iris
- [4] Wikipedia 'Sepal', Online:
  https://en.wikipedia.org/wiki/Sepal
- [5] F. Rosenblatt, 'The Perceptron--a perceiving and recognizing automaton',
  Report 85-460-1, Cornell Aeronautical Laboratory, 1957, Online:
  https://blogs.umass.edu/brain-wars/files/2016/03/rosenblatt-1957.pdf
- [6] Rosenblatt,'The Perceptron: A probabilistic model for information storage and orgainzation in the brain',
  Psychological Review 65(6), pp. 386-408, 1958, Online:
  https://psycnet.apa.org/doi/10.1037/h0042519
- [7] YouTube Video, 'Perceptron, Linear', Online:
  https://www.youtube.com/watch?v=FLPvNdwC6Qo
- [8] YouTube Video, 'Logistic Regression', Online:
  https://www.youtube.com/watch?v=1-0zMWp5w8U
- [9] Tensorflow, Online:
  https://www.tensorflow.org/
- [10] Keras Python Deep Learning Library, Online:
  https://keras.io/

# Lecture Bibliography (2)


- [11] Google Colaboratory, Online:
  https://colab.research.google.com
- [12] Machine Learning Mastery MNIST Tutorial, Online:
  https://machinelearningmastery.com/handwritten-digit-recognition-using-convolutional-neural-networks-python-keras/
- [13] Jupyter @JSC Web page, Online:
  https://jupyter-jsc.fz-juelich.de/hub/home
- [14] Big Data Tips, ‘Gradient Descent‘, Online:
  http://www.big-data.tips/gradient-descent
- [15] The XOR Problem in Neural Networks, Online:
  https://medium.com/@jayeshbahire/the-xor-problem-in-neural-networks-50006411840b
- [16] Multilayer Perceptron,Online:
  https://www.eecs.yorku.ca/course_archive/2012-13/F/4404-5327/lectures/10%20Multilayer%20Perceptrons.pdf
- [17] MIT 6.S191: Introduction to Deep Learning, Online:
  http://www.introtodeeplearning.com
- [18] Understanding the Neural Network, Online:
  http://www.cs.cmu.edu/~bhiksha/courses/deeplearning/Fall.2019/www/hwnotes/HW1p1.html
- [19] www.big-data.tips, ‘Relu Neural Network‘, Online:
  http://www.big-data.tips/relu-neural-network
- [20] www.big-data.tips, ‘tanh‘, Online:
  http://www.big-data.tips/tanh
- [21] YouTube Video, ’Neural Networks, A Simple Explanation’, Online:
  http://www.youtube.com/watch?v=gcK_5x2KsLA

# Lecture Bibliography (3)


- [22] Big Data Tips – Big Data Mining & Machine Learning, Online:
  http://www.big-data.tips/
- [23] Morris Riedel, 'Deep Learning - Using a Convolutional Neural Network', Invited YouTube Lecture, University of Ghent, 2017, Online:
  https://www.youtube.com/watch?v=gOL1_YIosYk&list=PLrmNhuZo9sgZUdaZ-f6OHK2yFW1kTS2qF
- [24] M. Riedel et al., 'Introduction to Deep Learning Models', JSC Tutorial, three days, JSC, 2019, Online:
  http://www.morrisriedel.de/introduction-to-deep-learning-models
- [25] H. Lee et al., 'Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representations', Online:
  http://doi.acm.org/10.1145/1553374.1553453
- [26] YouTube Video, 'Neural Network 3D Simulation', Online:
  https://www.youtube.com/watch?v=3JQ3hYko51Y
- [27] 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/
- [28] M. Nielsen, 'Neural Networks and Deep Learning', Online:
  http://neuralnetworksanddeeplearning.com/
- [29] M. Görner, Online:
  https://sites.google.com/site/nttrungmtwiki/home/it/data-science---python/tensorflow/tensorflow-and-deep-learning-part-3?tmpl=%2Fsystem%2Fapp%2Ftemplates%2Fprint%2F&showPrintDialog=1
- [30] Harley, A.W., An Interactive Node-Link Visualization of Convolutional Neural Networks, Online:
  https://www.cs.cmu.edu/~aharley/vis/conv/flat.html
- [31] 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
- [32] D. Kingma and Jimmy Ba, 'Adam: A Method for Stochastic Optimization', Online:
  https://arxiv.org/abs/1412.6980

# Lecture Bibliography (4)


- [33] YouTube Video, ‘Overfitting and Regularization For Deep Learning | Two Minute Papers #56’, Online: https://www.youtube.com/watch?v=6aF9sJrzxaM
- [34] 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, Online: https://www.researchgate.net/publication/282524415_On_Understanding_Big_Data_Impacts_in_Remotely_Sensed_Image_Classification_Using_Support_Vector_Machine_Methods
- [35] C. Cortes & V. Vapnik (1995). Support-vector networks. Machine learning, 20(3), 273-297, Online: https://doi.org/10.1007/BF00994018
- [36] www.big-data.tips, ‘SVM Train’, Online: http://www.big-data.tips/svm-train
- [37] www.big-data.tips, ‘Generalization in Machine Learning’, Online: http://www.big-data.tips/generalization-in-machine-learning
- [38] www.big-data.tips, ‘Cross Validation’, Online: http://www.big-data.tips/cross-validation
- [39] Original piSVM tool, Online: http://pisvm.sourceforge.net/
- [40] G. Sumbul, M. Charfuelan, B. Demir, V. Markl, BigEarthNet: A Large-Scale Benchmark Archive for Remote Sensing Image Understanding, IEEE International Conference on Geoscience and Remote Sensing Symposium, Yokohama, Japan, 2019.
- [41] Tensorflow Dataset ‘Big Earth Net’, Online: https://www.tensorflow.org/datasets/datasets#bigearthnet
- [42] JUWELS Supercomputer, Online: https://www.fz-juelich.de/ias/jsc/EN/Expertise/Supercomputers/JUWELS/JUWELS_node.html

# Lecture Bibliography (5)


- [43] 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, Online: https://www.researchgate.net/publication/328991957_The_Influence_of_Sampling_Methods_on_Pixel-Wise_Hyperspectral_Image_Classification_with_3D_Convolutional_Neural_Networks
- [44] G. Cavallaro, Y. Bazi, F. Melgani, M. Riedel, 'Multi-Scale Convolutional SVM Networks for Multi-Class Classification Problems of Remote Sensing Images', Proceedings of the IGARSS 2019 Conference, Online: https://www.researchgate.net/publication/337439088_Multi-Scale_Convolutional_SVM_Networks_for_Multi-Class_Classification_Problems_of_Remote_Sensing_Images
- [45] Kaiming He et al., 'Deep Residual Learning for Image Recognition', Online: https://arxiv.org/pdf/1512.03385.pdf
- [46] DeepSpeed, Online: https://www.deepspeed.ai/
- [47] Horovod: Uber's Open Source Distributed Deep Learning Framework for TensorFlow, Online: https://www.slideshare.net/databricks/horovod-ubers-open-source-distributed-deep-learning-framework-for-tensorflow
- [48] M. Goetz, C. Bodenstein, M. Riedel, 'HPDBSCAN – Highly Parallel DBSCAN', in proceedings of the ACM/IEEE International Conference for High Performance Computing, Networking, Storage, and Analysis (SC2015), Machine Learning in HPC Environments (MLHPC) Workshop, 2015, Online: https://www.researchgate.net/publication/301463871_HPDBSCAN_highly_parallel_DBSCAN
- [49] Ester, Martin, et al. "A density-based algorithm for discovering clusters in large spatial databases with noise." Kdd. Vol. 96. 1996, Online: https://dl.acm.org/citation.cfm?id=3001507
- [50] YouTube Video, 'DBSCAN Visualization and Explanation', Online: https://www.youtube.com/watch?v=_A9Tq6mGtLI
- [51] D-Wave Systems YouTube Channel, Online: https://www.youtube.com/user/dwavesystems
- [52] DEEP Projects Web page, Online: http://www.deep-projects.eu/

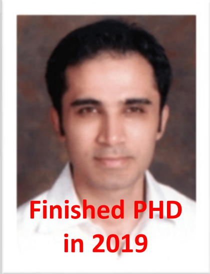
# Lecture Bibliography (6)


- [53] YouTube, Morris Riedel, UTmessan 2020 - Demystifying Quantum Computing, Online: https://www.youtube.com/watch?v=EQGshhspn9A
- [54] Cavallaro, G. & Riedel, M. et al.: APPROACHING REMOTE SENSING IMAGE CLASSIFICATION WITH ENSEMBLES OF SUPPORT VECTOR MACHINES ON THE D-WAVE QUANTUM ANNEALER, in conference proceedings of the (IGARSS 2020), Virtual Conference, Hawai, USA, Online: https://www.researchgate.net/publication/349431346_Approaching_Remote_Sensing_Image_Classification_with_Ensembles_of_Support_Vector_Machines_on_the_D-Wave_Quantum_Annealer
- [55] Riedel, M., Cavallaro, G., Benediktsson, J. A.: Practice and Experience in Using Parallel and Scalable Machine Learning in Remote Sensing from HPC Over Cloud to Quantum Computing, in conference proceedings of the IEEE International Geoscience and Remote Sensing Symposium (IGARSS 2021), July 12 – 16, 2021, Virtual Conference, Brussels, Belgium, Online: https://www.researchgate.net/publication/353342722_Practice_and_Experience_in_Using_Parallel_and_Scalable_Machine_Learning_in_Remote_Sensing_from_HPC_Over_Cloud_to_Quantum_Computing
- [56] Delilbasic, A., Cavallaro, G., Willsch, M., Melgani, F., Riedel, M., Michielsen, K.: Quantum Support Vector Machine Algorithms for Remote Sensing Data Classification, in conference proceedings of the IEEE International Geoscience and Remote Sensing Symposium (IGARSS 2021), July 12 – 16, 2021, Virtual Conference, Brussels, Belgium, Online: https://www.researchgate.net/publication/353296104_Quantum_Support_Vector_Machine_Algorithms_for_Remote_Sensing_Data_Classification

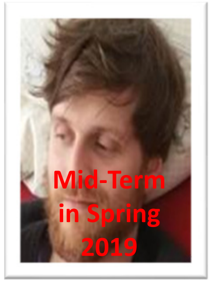
# Acknowledgements – High Productivity Data Processing Research Group


PD Dr.
G. Cavallaro


PD Dr.
A.S. Memon



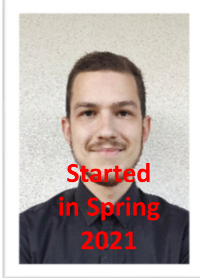

PD Dr.
M.S. Memon




PhD Student
E. Erlingsson

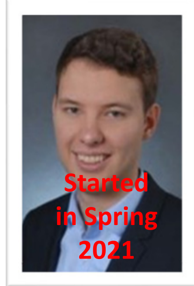

PhD Student
S. Bakarat


PhD Student
R. Sedona

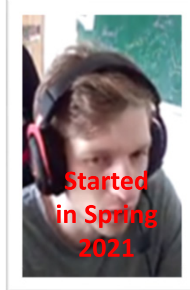



PhD Student
A. Delilbasic


PhD Student
S. Sharma

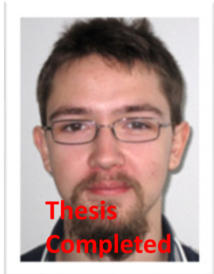



PhD Student
M. Aach




PhD Student
D. Helmrich


Dr. M. Goetz
(now KIT)

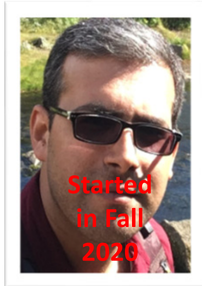



MSc M.
Richerzhagen
(now other division)

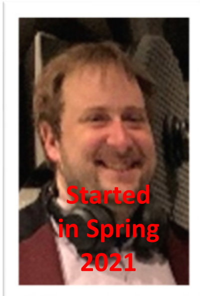

MSc
P. Glock
(now INM-1)


MSc
C. Bodenstein
(now
Soccerwatch.tv)


MSc G.S.
Guðmundsson
(Landsverkjun)




PhD Student
Reza




PhD Student
E. Sumner


This research group has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 763558 (DEEP-EST EU Project) and grant agreement No 951740 (EuroCC EU Project) & 951733 (RAISE EU Project) & 956748 (ADMIRE EU Project)


Data
Science
Big
Research
science
analysis
methods
big
simulations
processing
computational
Computational
TB
European
technologies
computing
Scientific
Supercomputing
performance
datasets
research
Climate
HPC
Juelich
Centre
climate
Computing
nodes
centers
storage
ENES
EUDAT
databases
modelling
structures
access
Hardware
management
data-intensive
distributed
large-scale
Service
community
Environment
scientific
important
Infrastructure
Enable
Modelling
images
often
services
measurement
new
funding
concepts
Policy-based
resources
fusion
within
device
Analysis
Computer
expertise
disciplines
results
Approaches
construction
Fusion
different
files
forms
cross-disciplinary
year
like
Cross-Disciplinary
EU
E.g
Test
High
PB
Key
basis
DKRZ
Use
etc
increasing
via
physical
IPCC
Common
Methods
Using
Work
brain
formats
Project
MapReduce
create
GB
codes
SMAQ
international
model
engineering
References
DLCL
Level
stored
Resources
analyze
Large
Summary
Matters
Concrete
external
levels
JSC
use
e.g
acquisition
System
DLCLs
Understanding
Germany
Provide
NASA
Energy
systems
Simulation
AR5
Technologies
Earth
directory
day
Processing
Health
compute
cores
project
Structure
General
current
system
Landscape
hundreds
Nutshell
Services
network