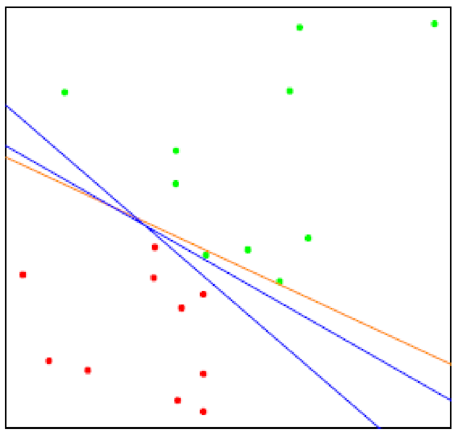


Parallel & Scalable Machine Learning

Introduction to Machine Learning Algorithms

Prof. Dr. – Ing. Morris Riedel

Adjunct Associated Professor

School of Engineering and Natural Sciences, University of Iceland

Research Group Leader, Juelich Supercomputing Centre, Germany

LECTURE 2

Introduction to Machine Learning Fundamentals

February 25th, 2019

Juelich Supercomputing Centre, Juelich, Germany

PRACE PARTNERSHIP FOR ADVANCED COMPUTING IN EUROPE TRAINING PORTAL

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

JÜLICH Forschungszentrum

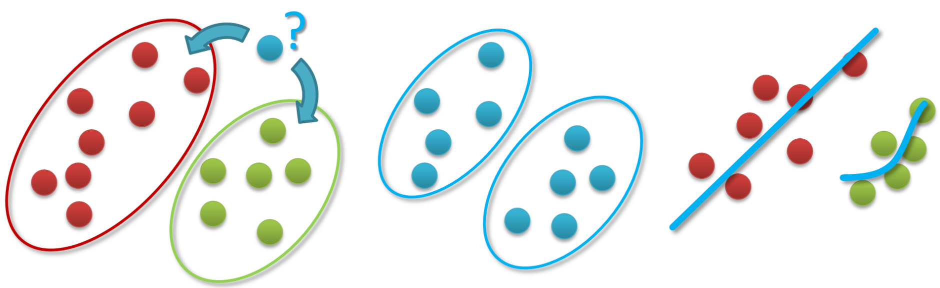
HELMHOLTZ RESEARCH FOR GRAND CHALLENGES

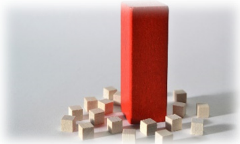
DEEP Projects

Review of Lecture 1 – Parallel & Scalable ML Driven by HPC

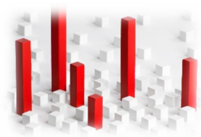
- Machine Learning Methods Overview & Momentum Today

Classification **Clustering** **Regression**

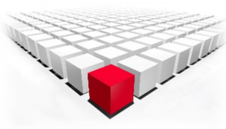




Big Data
- Larger datasets
- Easier access to collections
- Better transport to storage



Hardware
- More memory
- Graphical Processing Units (GPUs)
- Massively parallel systems



Software
- Improved scalable techniques
- New models
- Open source frameworks & toolsets

[7] NVIDIA

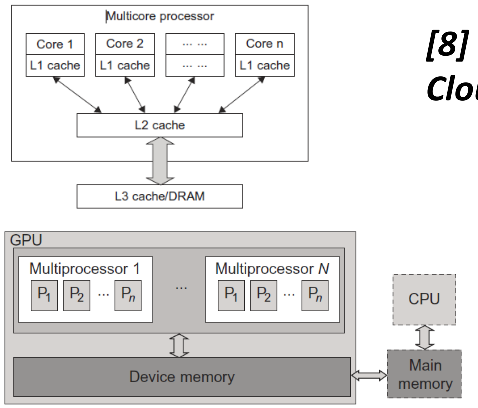
Keras

[5] Keras

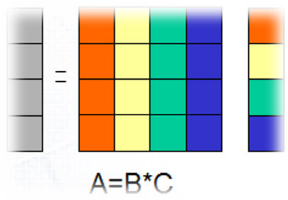
TensorFlow

[6] TensorFlow

- Parallel & Scalable Technology



[8] Distributed & Cloud Computing Book



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

P0 P1 P2 P3

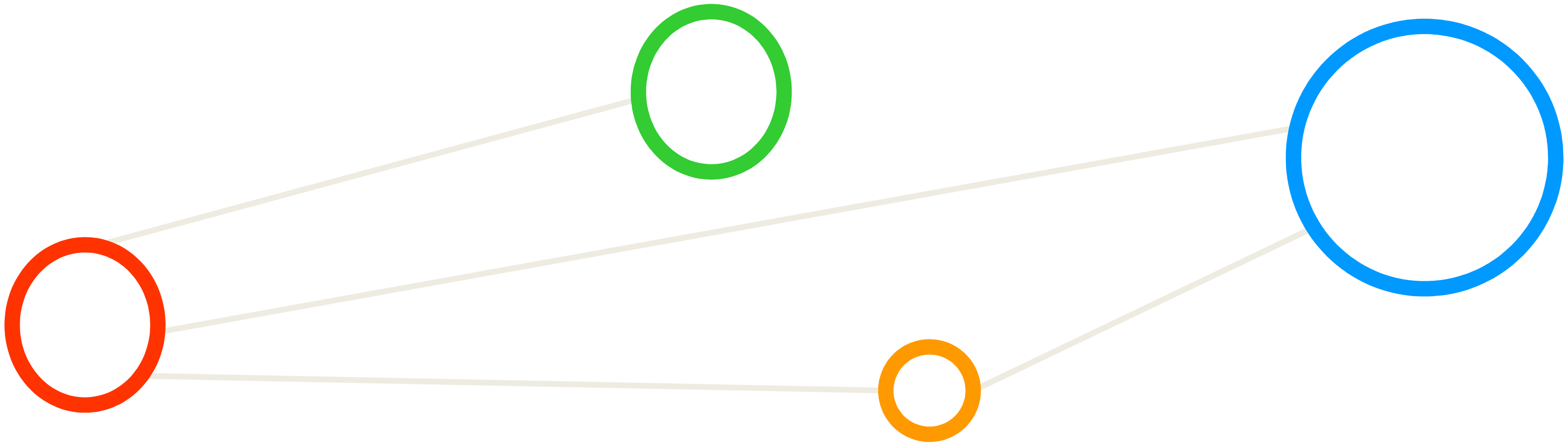
Outline of the Course

1. Parallel & Scalable Machine Learning driven by HPC
2. Introduction to Machine Learning Fundamentals
3. Introduction to Machine Learning Fundamentals
4. Feed Forward Neural Networks
5. Feed Forward Neural Networks
6. Validation and Regularization
7. Validation and Regularization
8. Data Preparation and Performance Evaluation
9. Data Preparation and Performance Evaluation
10. Theory of Generalization
11. Unsupervised Clustering and Applications
12. Unsupervised Clustering and Applications
13. Deep Learning Introduction

Theoretical Lectures

Practical Lectures

Outline

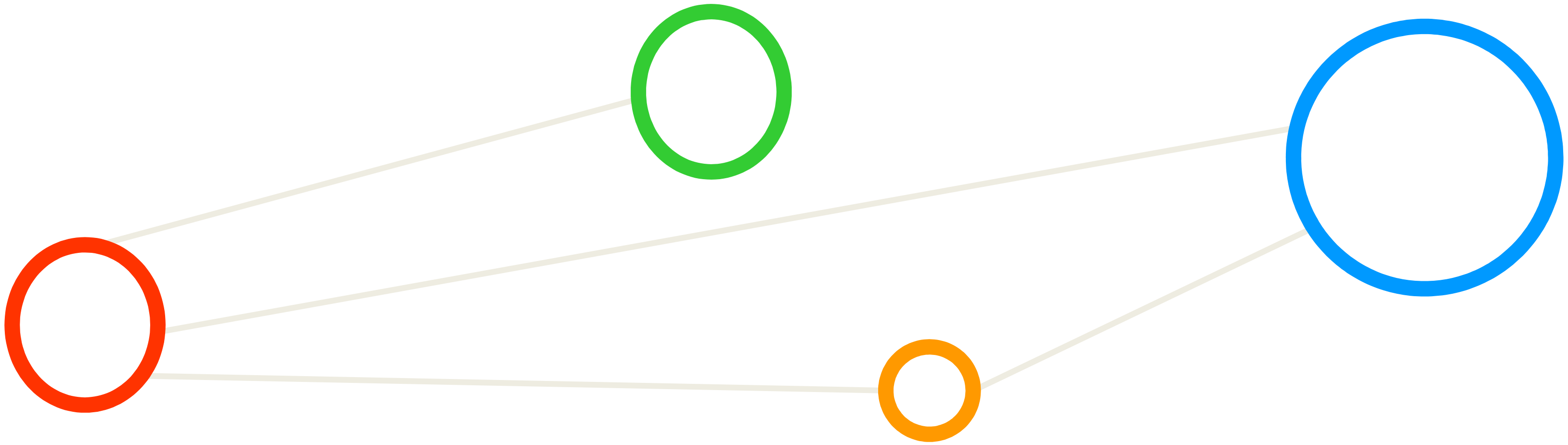


Outline

- Machine Learning Basics
  - Systematic Process to Support Learning
  - Learning Approaches
  - Simple Application Example
  - Perceptron Learning Model
  - Decision Boundary & Linear Seperability

- Learning from Data
  - Hand-written Character Recognition Problem
  - Data Exploration using Jupyter & NumPy
  - Multi-Class Classification Problem
  - Multi-Output Perceptron Model
  - Using Keras & TensorFlow in Jupyter

Machine Learning Basics

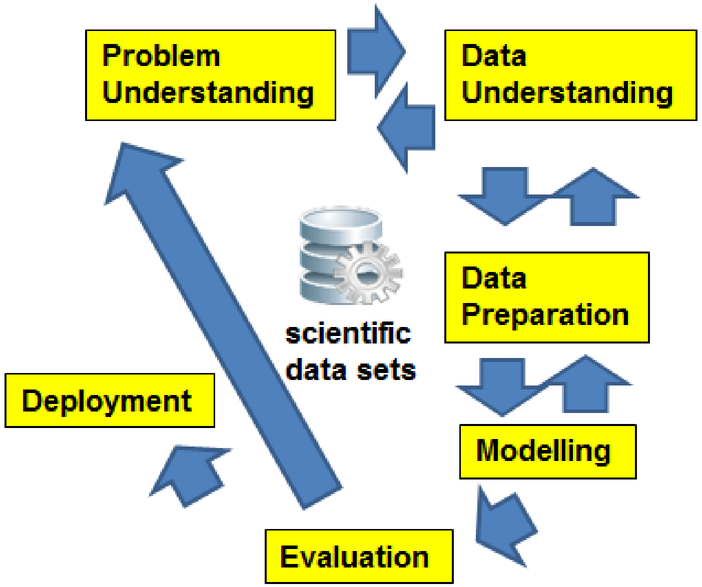


Systematic Process to Support Learning From Data

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

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

(learning takes place)



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

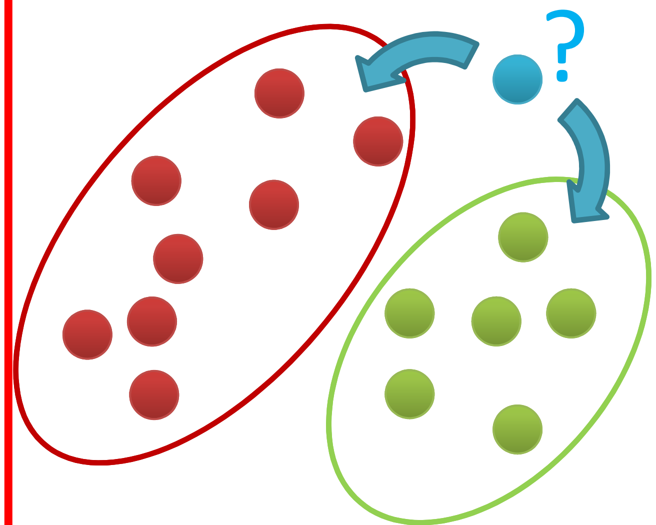
- Lessons Learned from Practice
  - Go back and forth between the different six phases

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

Machine Learning Models make use of ‘Big Data‘

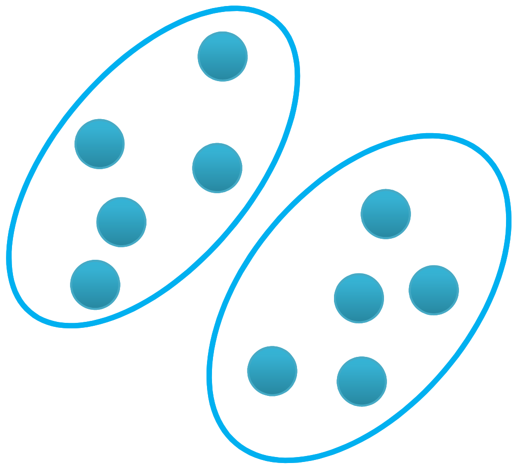
- **Machine learning methods can be roughly categorized in classification, clustering, or regression augmented with various techniques for data exploration, selection, or reduction**

Classification



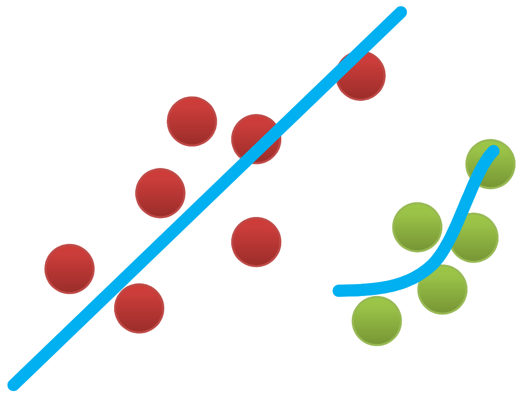
- Groups of data exist
- New data classified to existing groups

Clustering



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

Regression



- Identify a line with a certain slope describing the data

Learning Approaches – What means Learning?

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

- Supervised Learning
  - Majority of methods follow this approach in this course
  - Example: credit card approval based on previous customer applications
- Unsupervised Learning
  - Often applied before other learning → higher level data representation
  - Example: Coin recognition in vending machine based on weight and size
- Reinforcement Learning
  - Typical ‘human way‘ of learning
  - Example: Toddler tries to touch a hot cup of tea (again and again)

Learning Approaches – Supervised Learning

- Each observation of the predictor measurement(s) has an associated response measurement:
  - Input $\mathbf{x} = x_1, ..., x_d$
  - Output $y_i, i = 1, .., n$
  - Data $(\mathbf{x}_1, y_1), ..., (\mathbf{x}_N, y_N)$
- Goal: Fit a model that relates the response to the predictors
  - Prediction: Aims of accurately predicting the response for future observations
  - Inference: Aims to better understanding the relationship between the response and the predictors

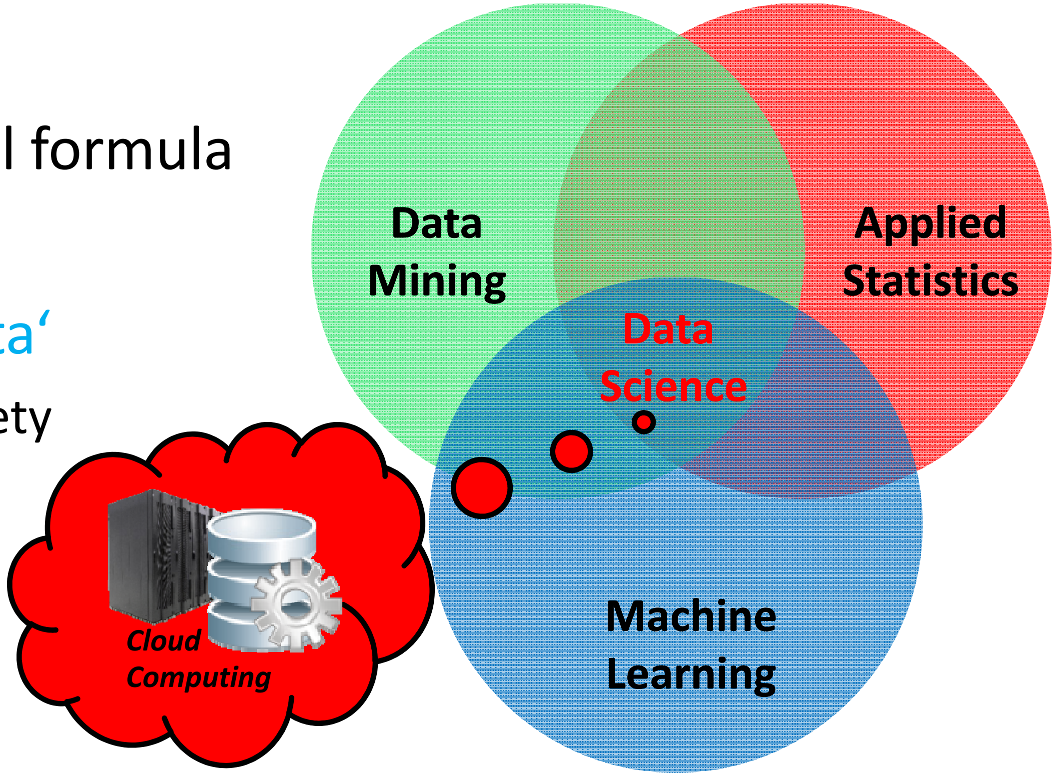
- **Supervised learning approaches fits a model that related the response to the predictors**
- **Supervised learning approaches are used in classification algorithms such as SVMs**
- **Supervised learning works with data = [input, correct output]**

[9] An Introduction to Statistical Learning

Machine Learning Prerequisites & Challenges

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

- Idea 'Learning from Data'
  - Shared with a wide variety of other disciplines
  - E.g. signal processing, data mining, etc.
- Challenges
  - Data is often complex
  - Learning from data requires processing time → Clouds

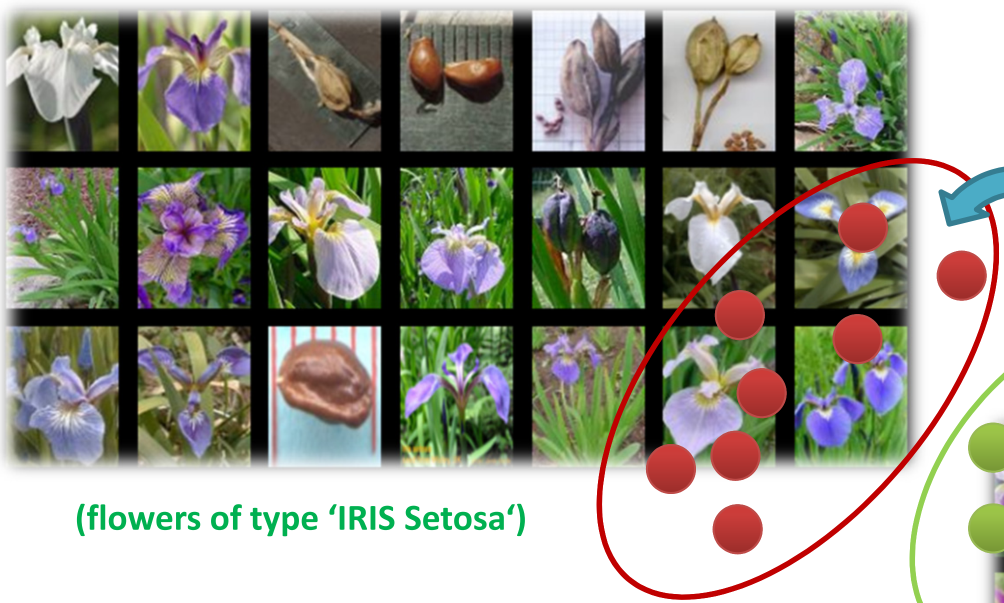


- **Machine learning is a very broad subject and goes from very abstract theory to extreme practice ('rules of thumb')**
- **Training machine learning models needs processing time**

Simple Application Example: Classification of a Flower

(1) Problem Understanding Phase

(what type of flower is this?)



(flowers of type 'IRIS Setosa')

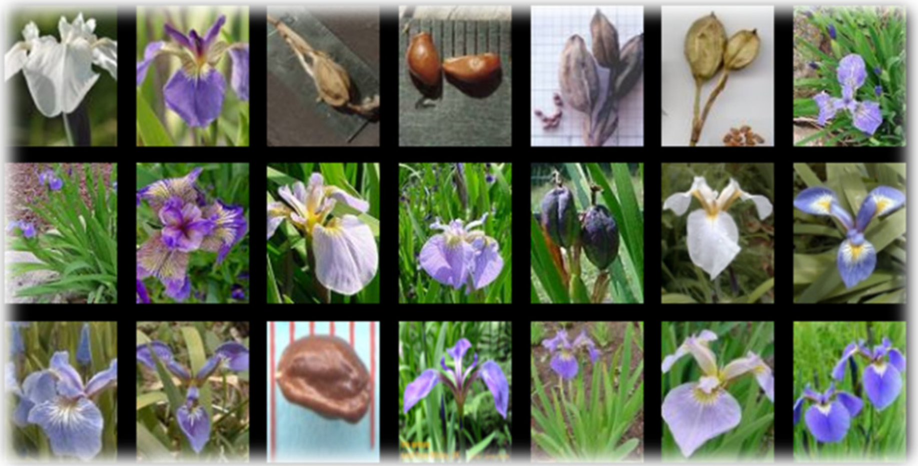
- Groups of data exist
- New data classified to existing groups

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

(flowers of type 'IRIS Virginica')

The Learning Problem in the Example

(flowers of type 'IRIS Setosa')



(flowers of type 'IRIS Virginica')

[10] 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

[12] 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

[11] UCI Machine Learning Repository Iris Dataset

(2) Data Understanding Phase

Understanding the Data – Check Metadata

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

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

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

                        ...

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

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

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

(Subject, title, or context)

(author, source, or creator)

(number of samples, instances)

(attribute information)

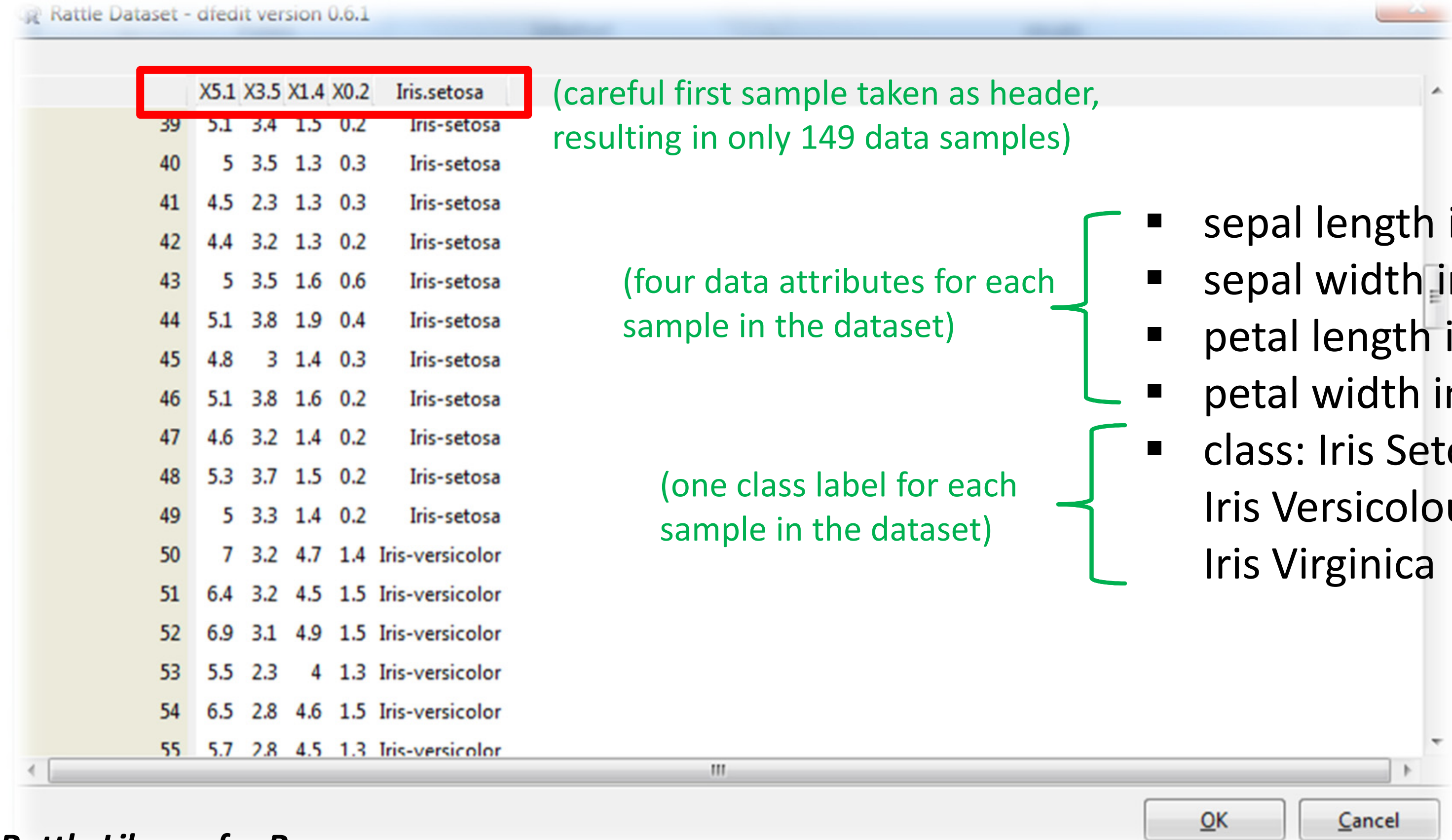
(detailed attribute information)

(detailed attribute information)

[11] UCI Machine Learning Repository Iris Dataset

Understanding the Data – Check Table View

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



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

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

(four data attributes for each sample in the dataset)

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

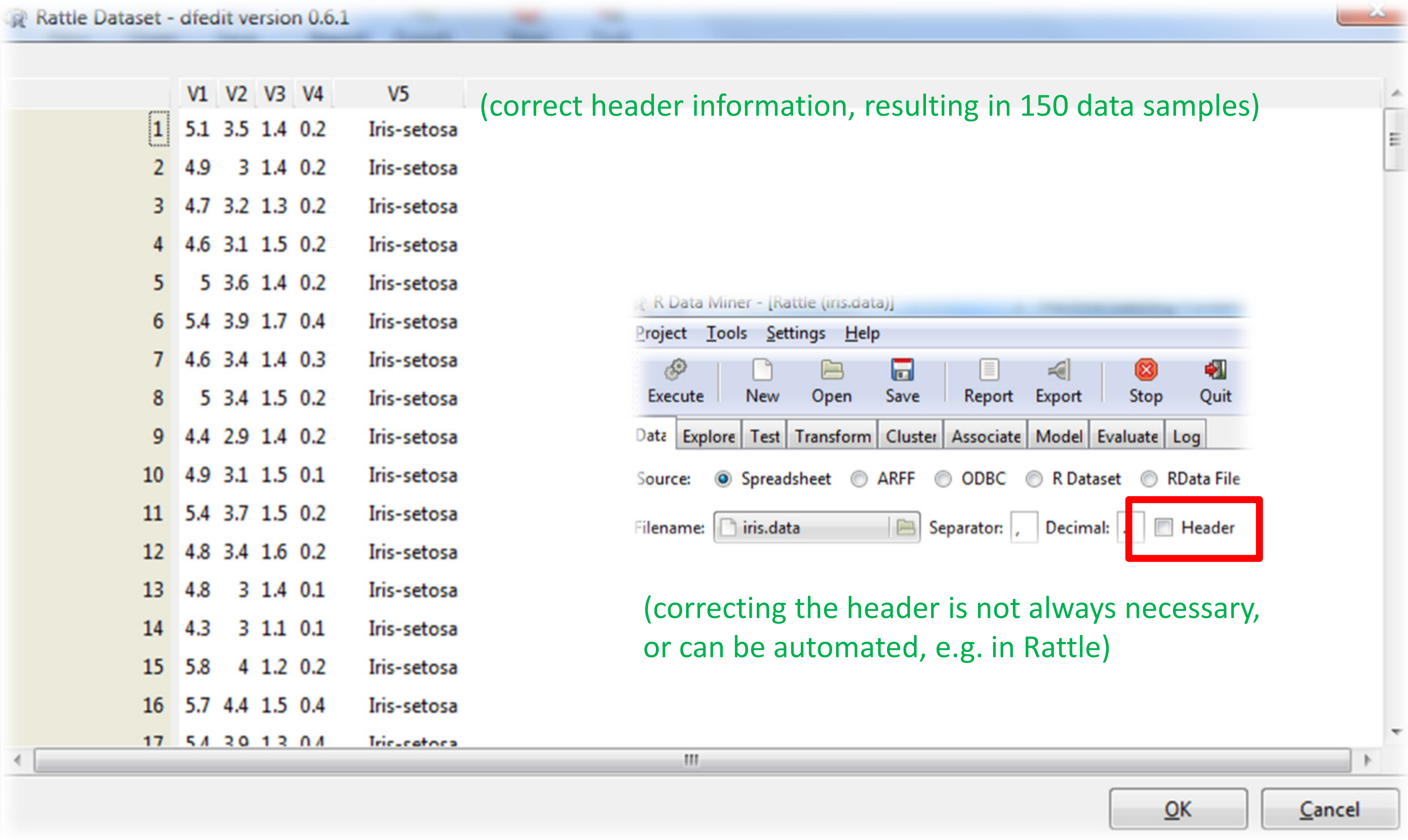
(one class label for each sample in the dataset)

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

[18] Rattle Library for R

Preparing the Data – Corrected Header

(3) Data Preparation Phase

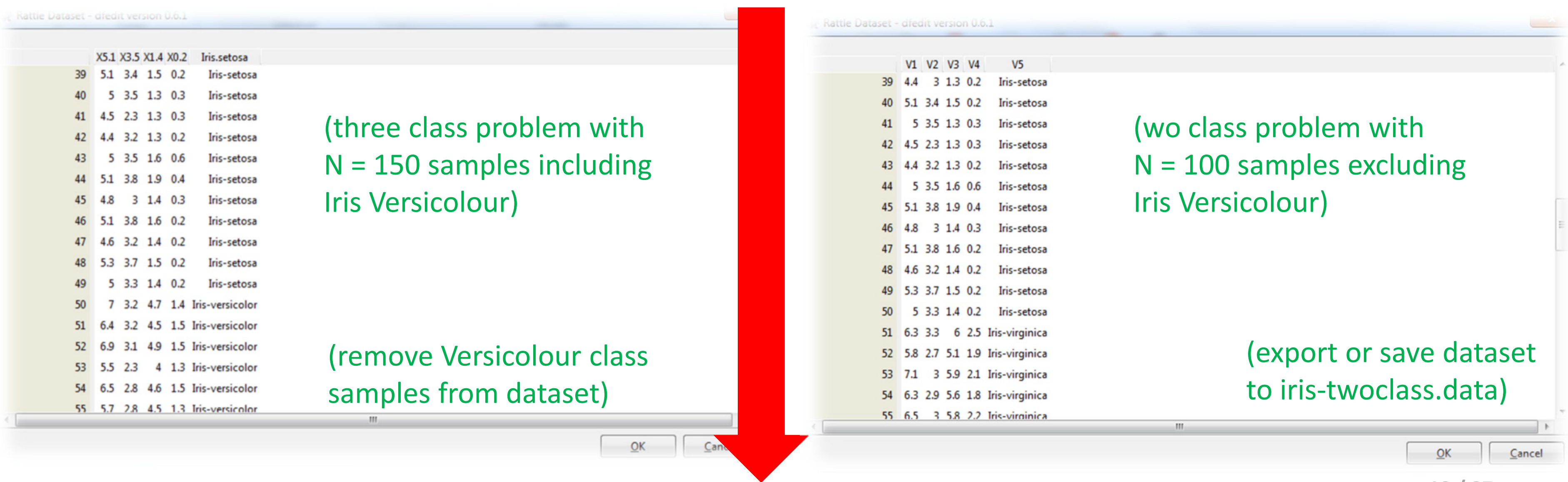


(correct header information, resulting in 150 data samples)

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

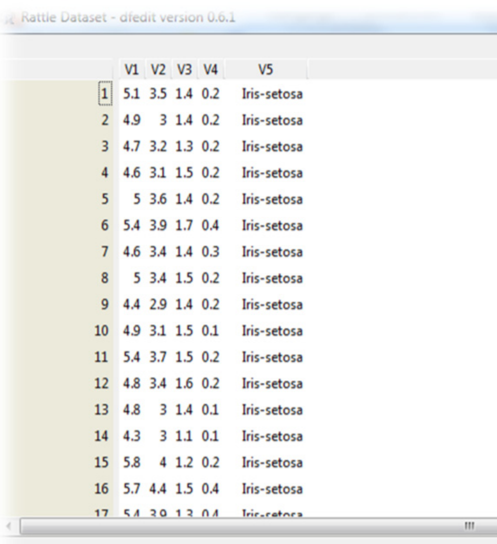
Preparing the Data – Remove Third Class Samples

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



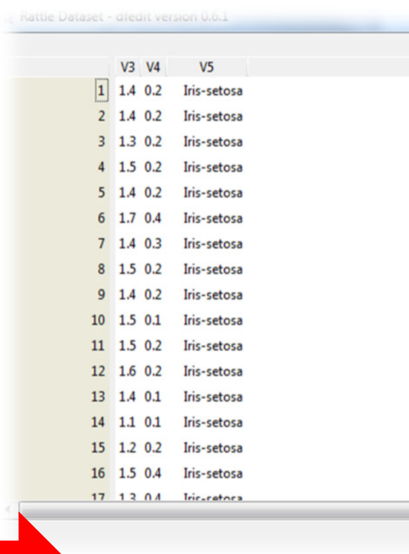
Preparing the Data – Feature Selection Process

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



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

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



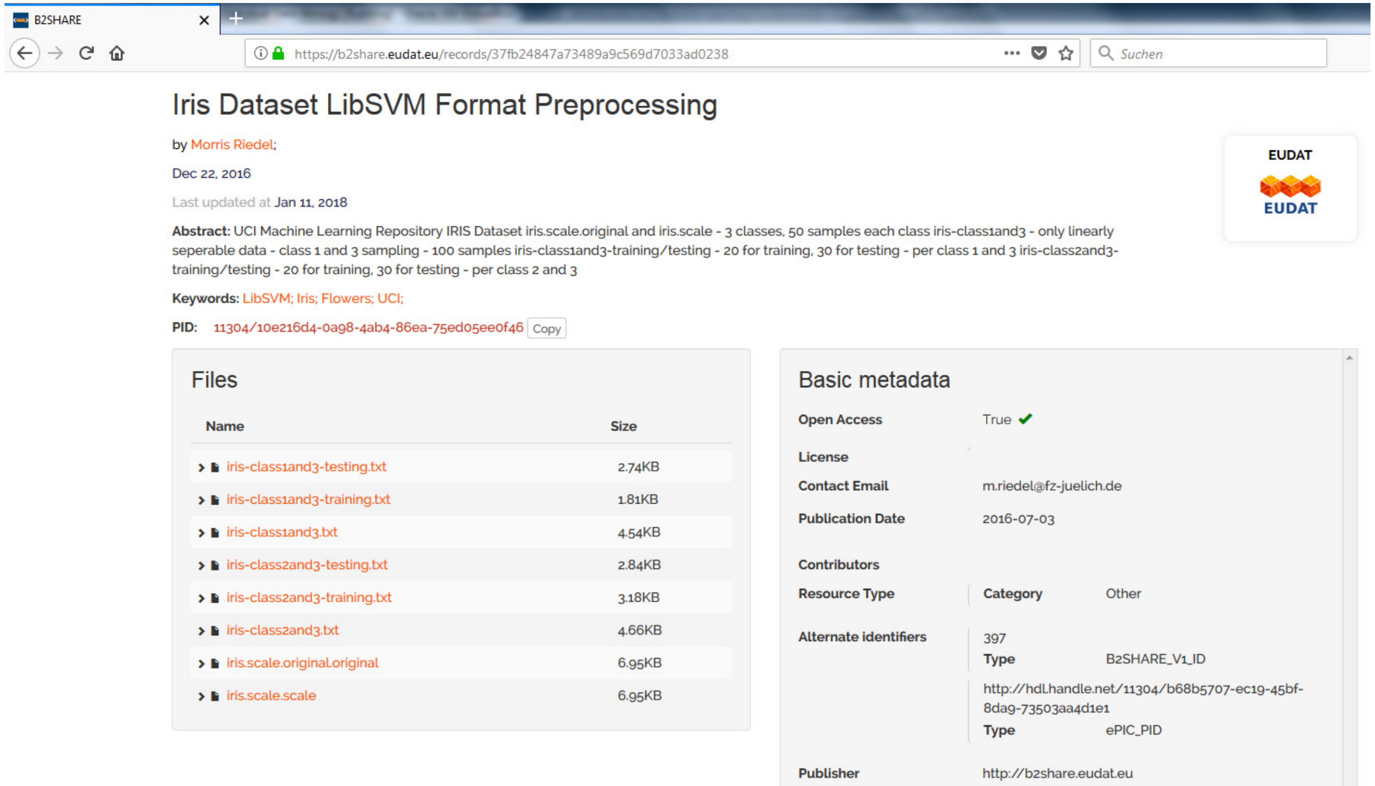
- petal length in cm
- petal width in cm
- class: Iris Setosa, or Iris Versicolour, or Iris Virginica

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

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

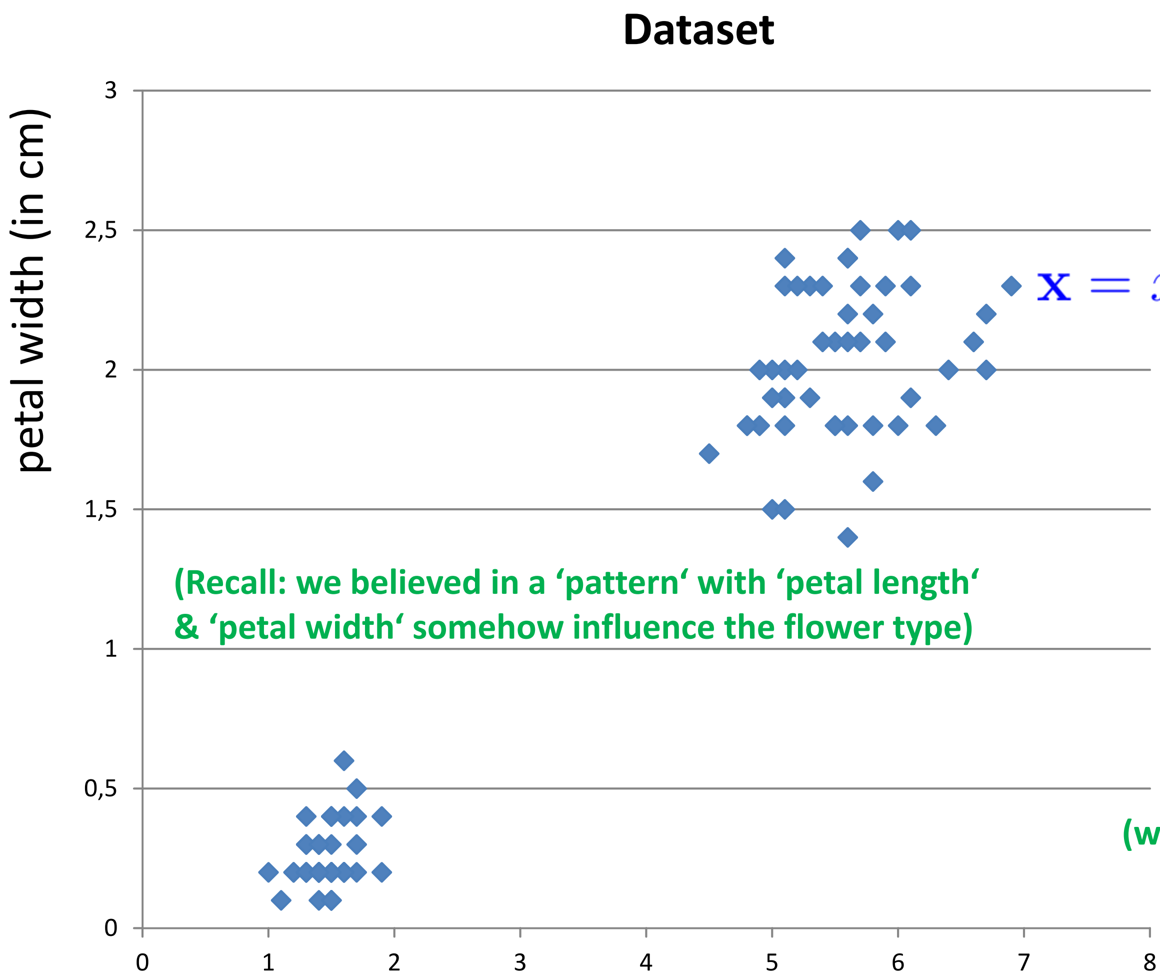
Iris Dataset – Open Data

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



[19] Iris Dataset

Check Preparation Phase: Plotting the Data (Two Classes)



(attributes with d=2)

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

(x1 is petal length,
x2 is petal width)

◆ Dataset

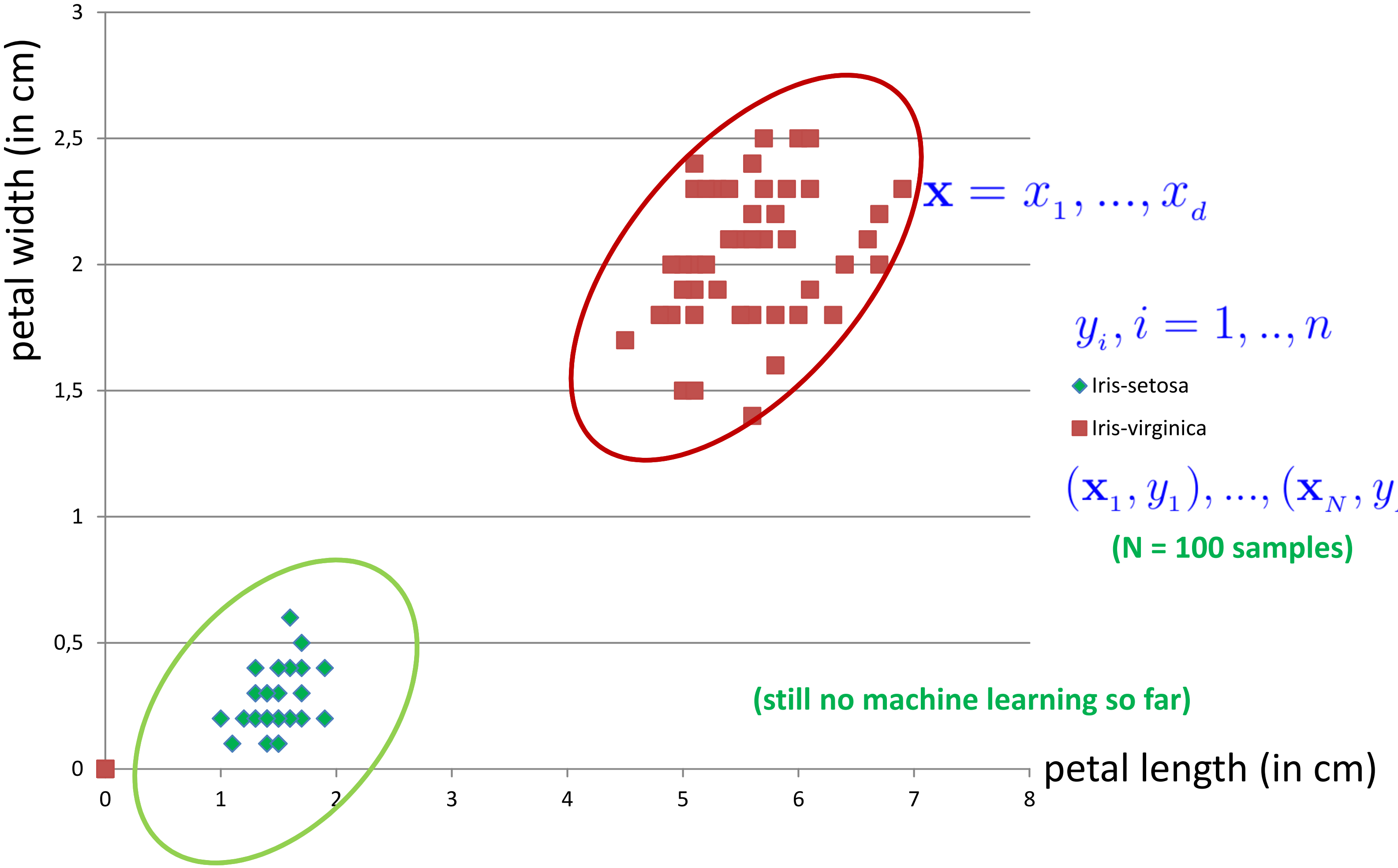
$$(\mathbf{x}_1), \ldots, (\mathbf{x}_N)$$

(N = 100 samples)

(what about the class labels?)

petal length (in cm)

Check Preparation Phase: Class Labels



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

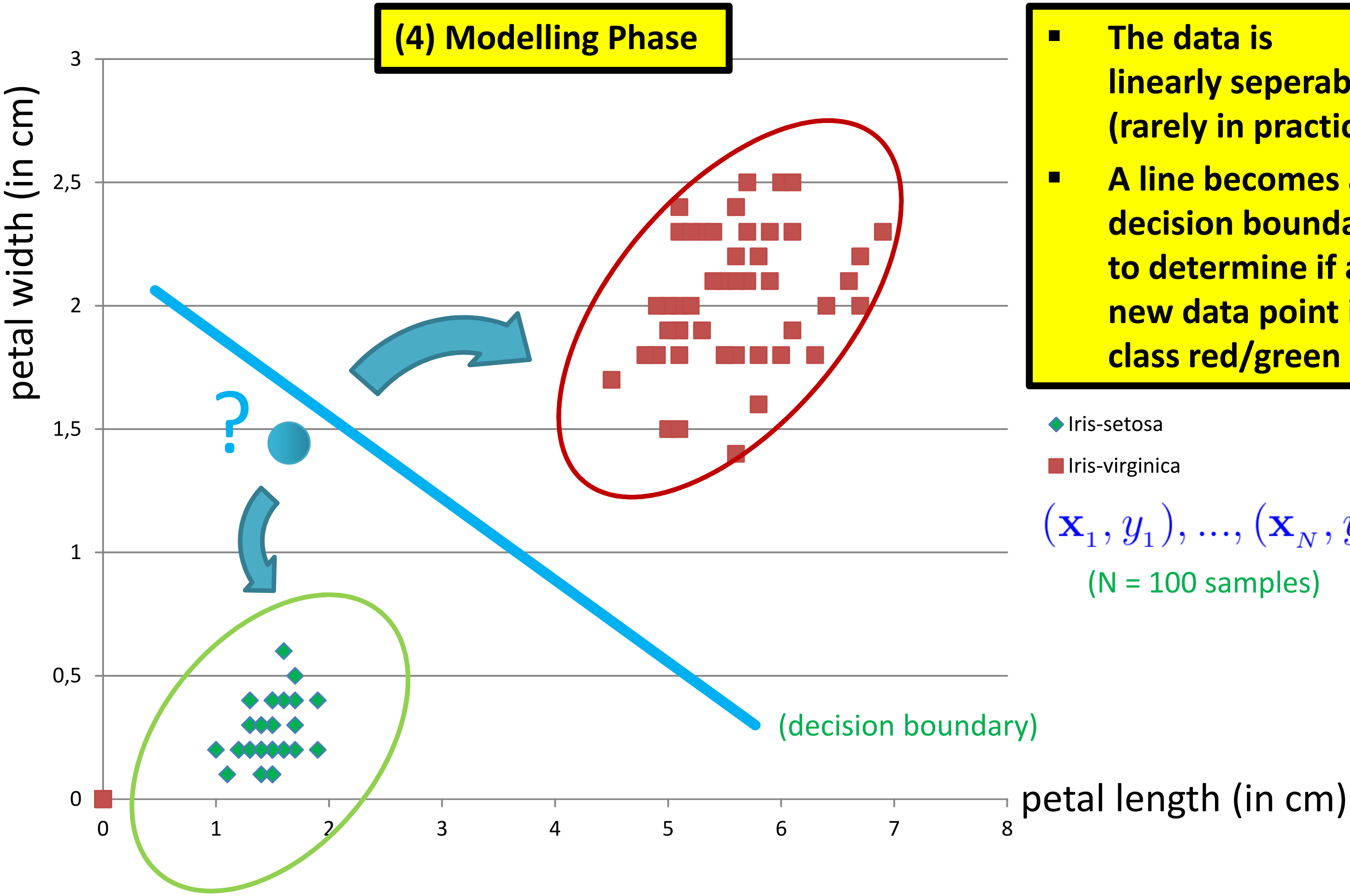
$y_i, i = 1, .., n$

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

(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

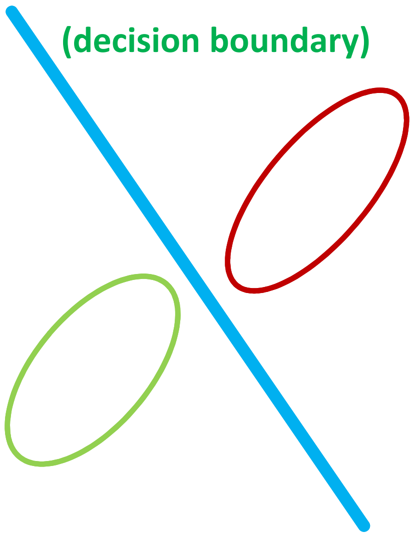
$(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)$

(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)

(decision boundary)



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

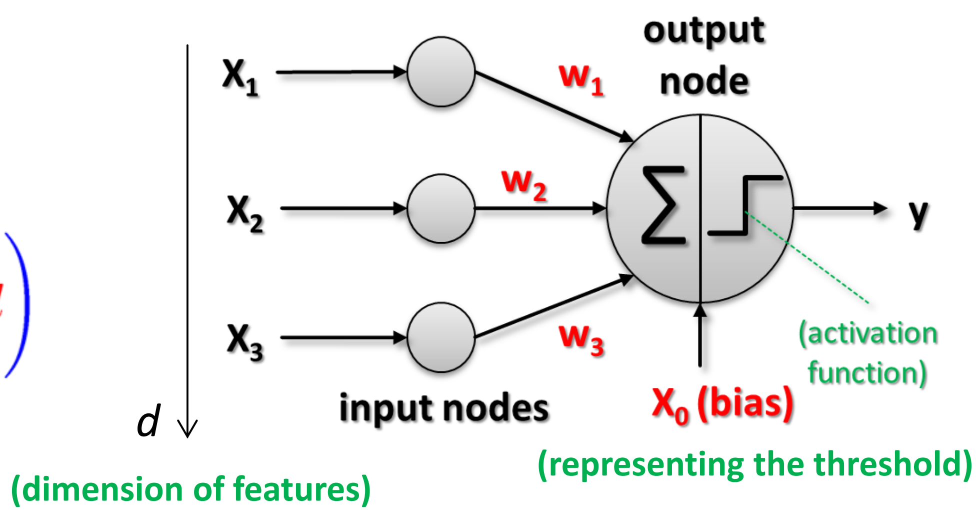
$$(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)$$

(training data)

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

(activation function, +1 or -1)

(modelled as bias term)



Perceptron – Example of a Boolean Function

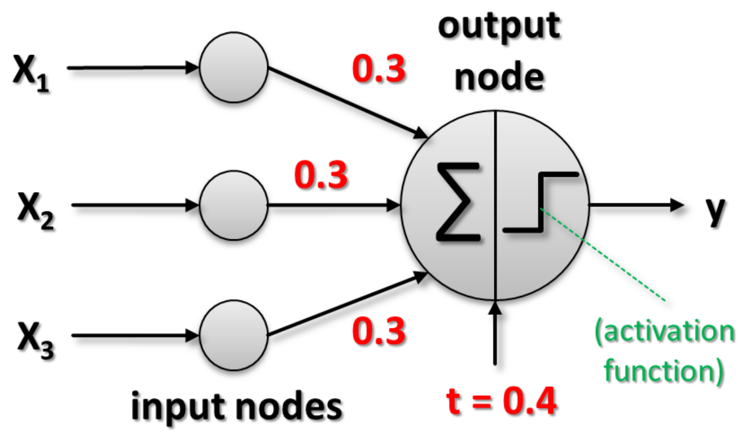
	x1	x2	x3	y
1	1	0	0	-1
2	1	0	1	1
3	1	1	0	1
4	1	1	1	1
5	0	0	1	-1
6	0	1	0	-1
7	0	1	1	1
8	0	0	0	-1

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

(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)

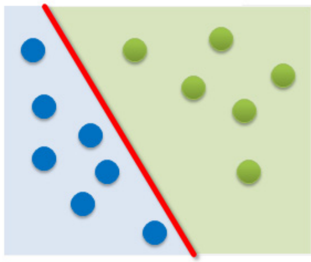
Summary Perceptron & Hypothesis Set h(x)

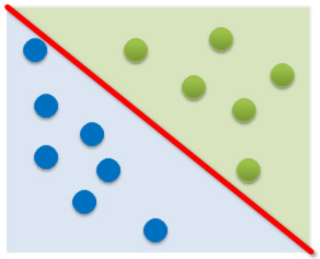
- When: Solving a linear classification problem *[13] F. Rosenblatt, 1957*
  - Goal: learn a simple value (+1/-1) above/below a certain threshold
  - Class label renamed: Iris-setosa = -1 and Iris-virginica = +1
- Input: $\mathbf{x} = x_1, \ldots, x_d$ **(attributes in one dataset)**
- Linear formula **(take attributes and give them different weights – think of 'impact of the attribute')**
  - All learned formulas are different hypothesis for the given problem

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

(parameters that define one hypothesis vs. another)

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





(red parameters correspond to the redline in graphics)

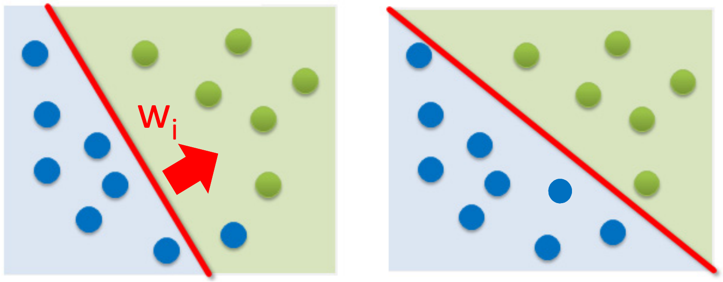
(but question remains: how do we actually learn w_i and threshold?)

Perceptron Learning Algorithm – Understanding Vector W

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

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

(points on the decision boundary satisfy this equation)



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

(vector notation, using T = transpose)

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

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

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

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

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

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

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

(equivalent dotproduct notation)

[14] Rosenblatt, 1958

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

Understanding the Dot Product – Example & Interpretation

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

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

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

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

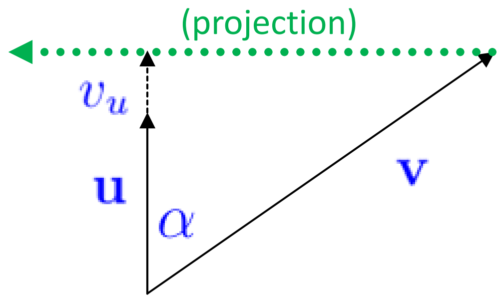
(our example)

- 'Projection capabilities of Dot product' (simplified)
  - Orthogonal projection of vector $\mathbf{v}$ in the direction of vector $\mathbf{u}$

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

- Normalize using length of vector

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



Perceptron Learning Algorithm – Learning Step

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

(one point at a time is picked)

1. Pick one misclassified training point where:

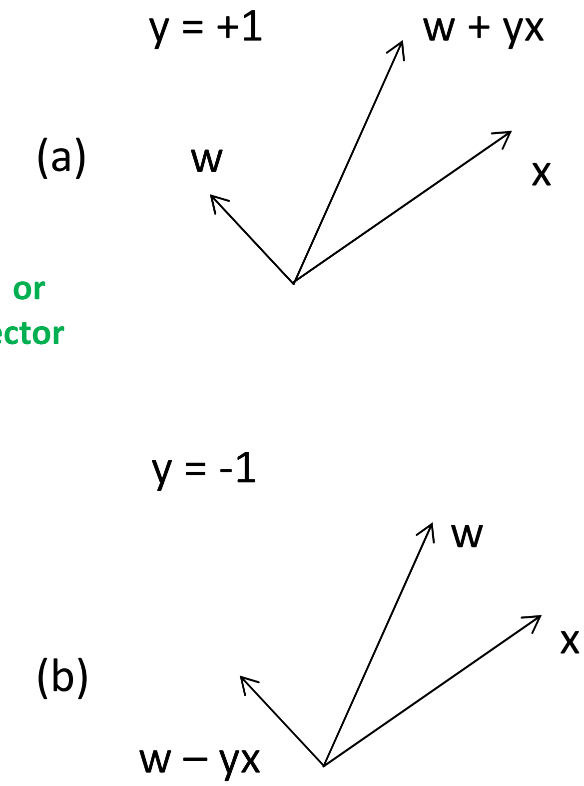
   $$sign(\mathbf{w}^T \mathbf{x}_n) \neq y_n$$

2. Update the weight vector:

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

   (y_n is either +1 or -1)

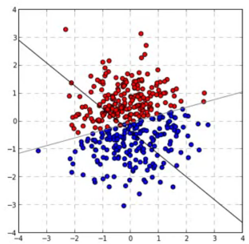
(a) adding a vector or
(b) subtracting a vector



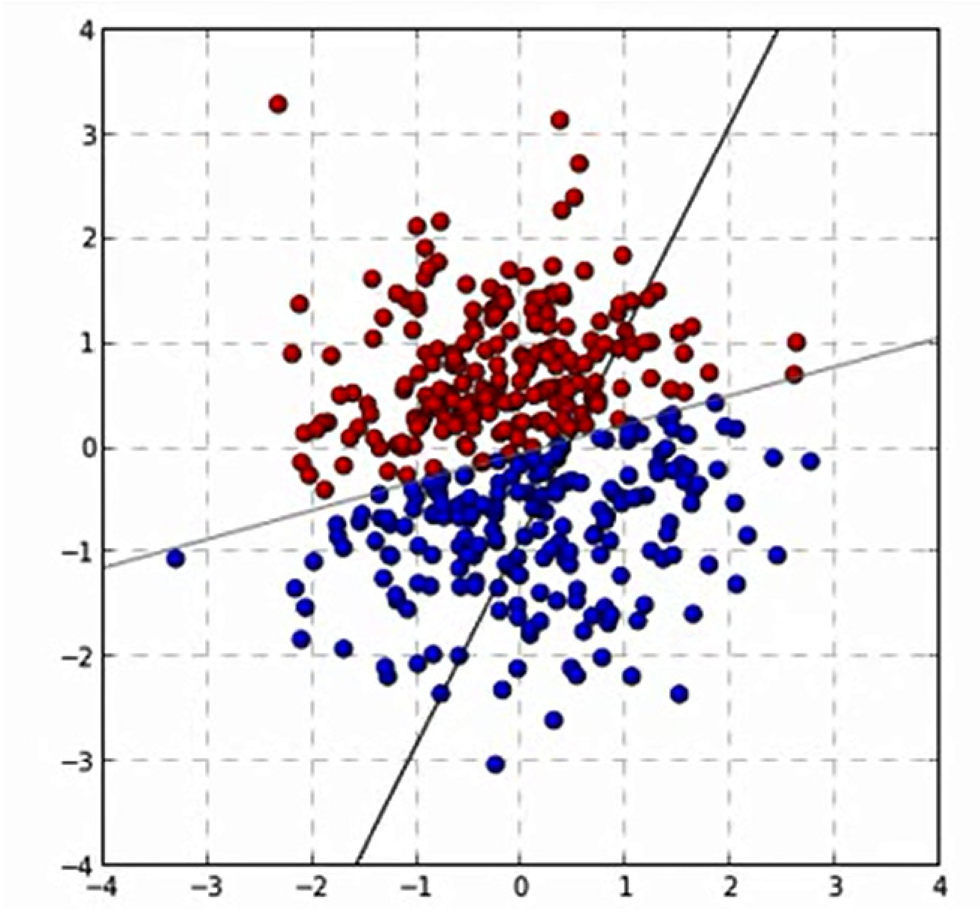
- Terminates when there are no misclassified points

(converges only with linearly seperable data)

[15] Perceptron Visualization

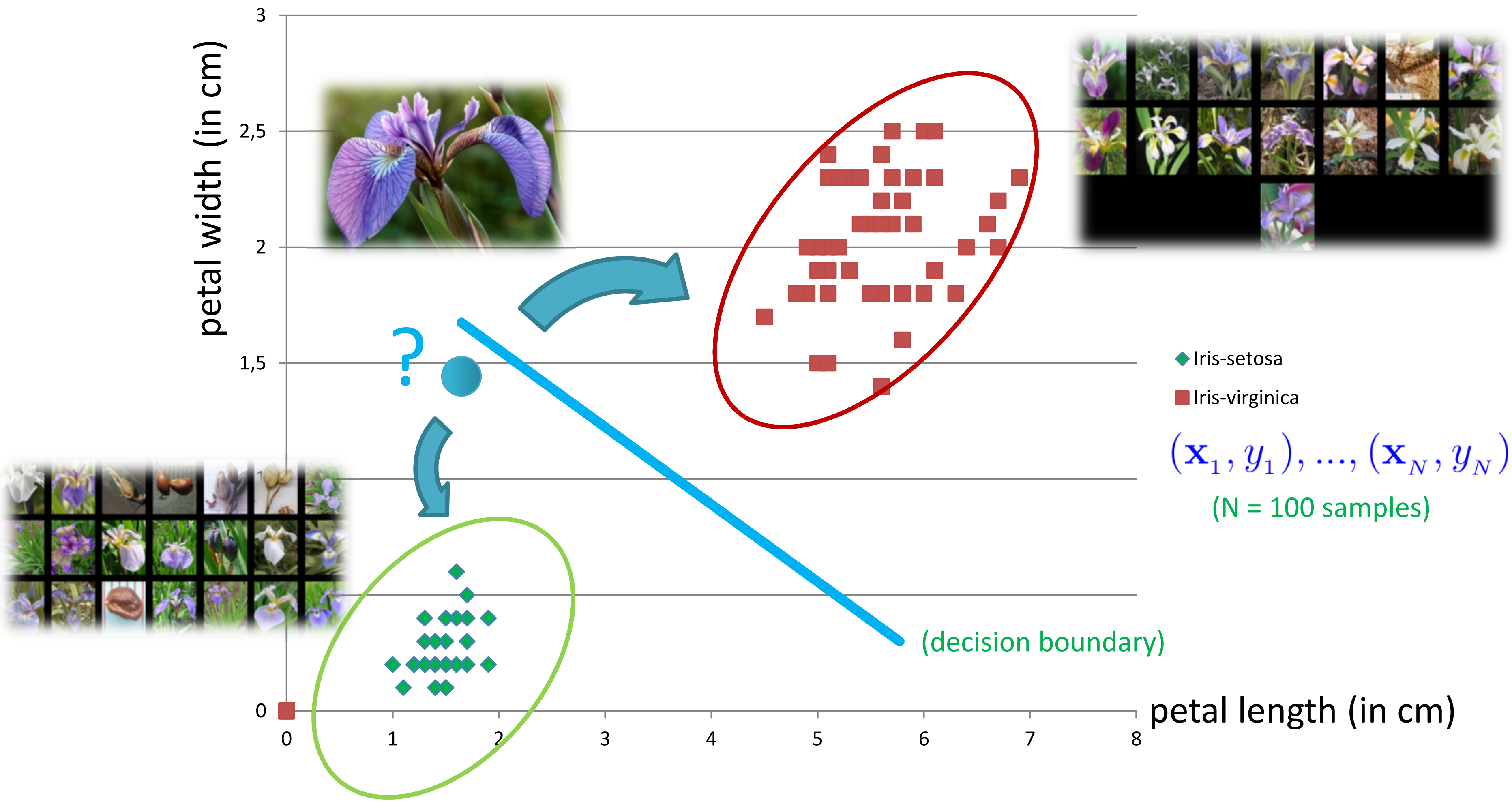


[Video] Perceptron Learning Algorithm



[15] PLA Video

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



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

(N = 100 samples)

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

Summary Terminologies & Different Dataset Elements

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

Summary: Simple Linear Learning Model – Perceptron

[2] F. Rosenblatt, 1957

non-linear activation function

linear combination of input data

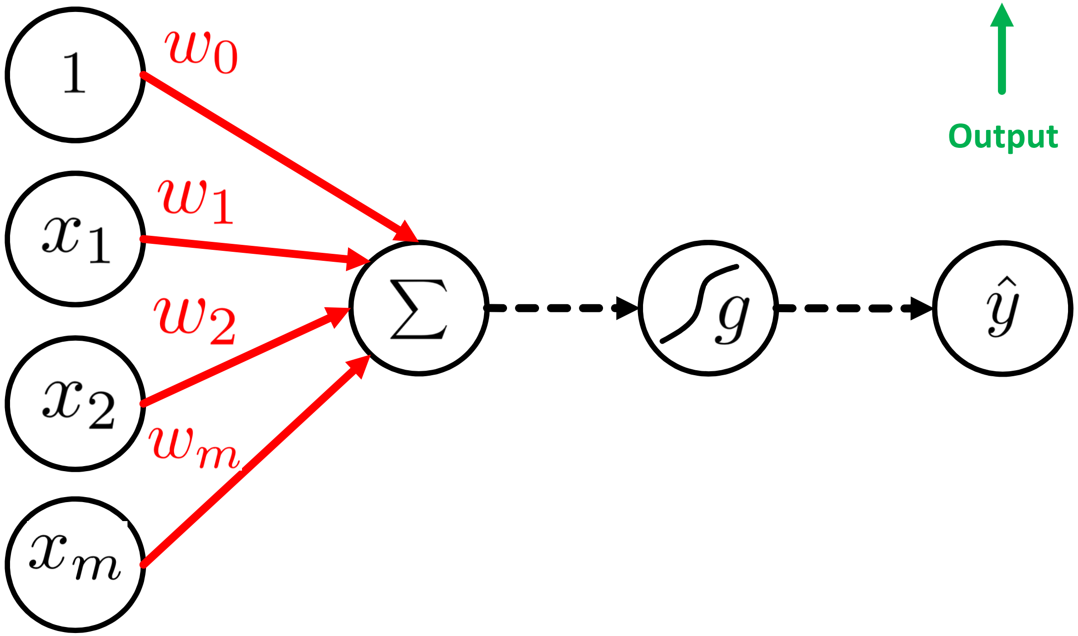
$$\hat{y} = g\left(1 * w_0 + \sum_{i=1}^{m} x_i * w_i\right)$$

Output

Bias

Sum

Constants



Input Data

Trainable Weights

Sum

Non-Linearity

Output

Keras

[1] Lego Bricks

Summary: Perceptron Model – Mathematical Convenience

non-linear activation function

linear combination of input data

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

Output

Bias

Sum

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

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

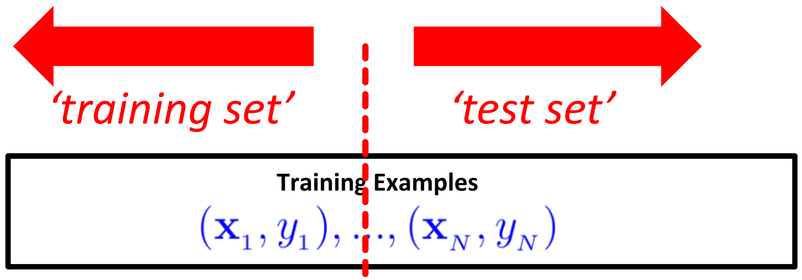
Trainable Weights

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

Model Evaluation – Training and Testing Phases

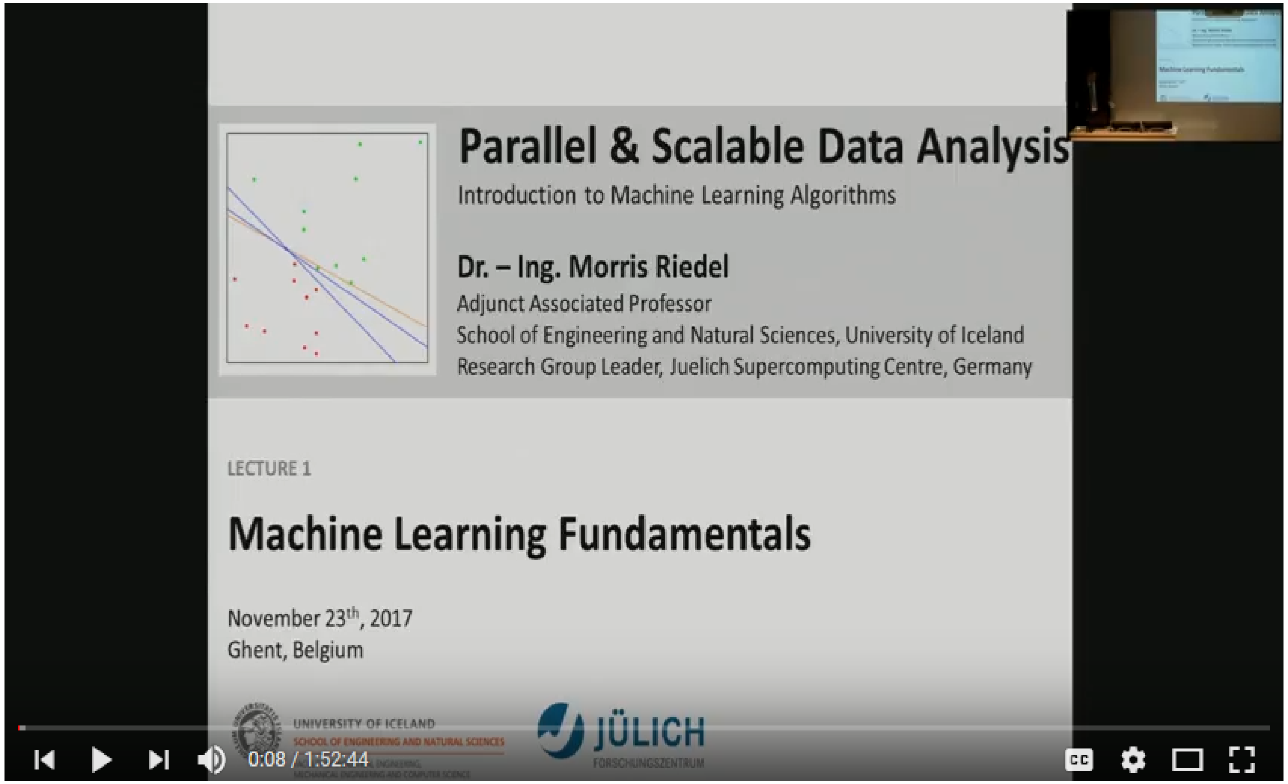
- Different Phases in Learning
  - Training phase is a hypothesis search
  - Testing phase checks if we are on right track (once the hypothesis clear)
- Work on 'training examples'
  - Create two disjoint datasets
  - One used for training only (aka training set)
  - Another used for testing only (aka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

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



(historical records, groundtruth data, examples)

[YouTube Lectures] More Machine Learning Fundamentals

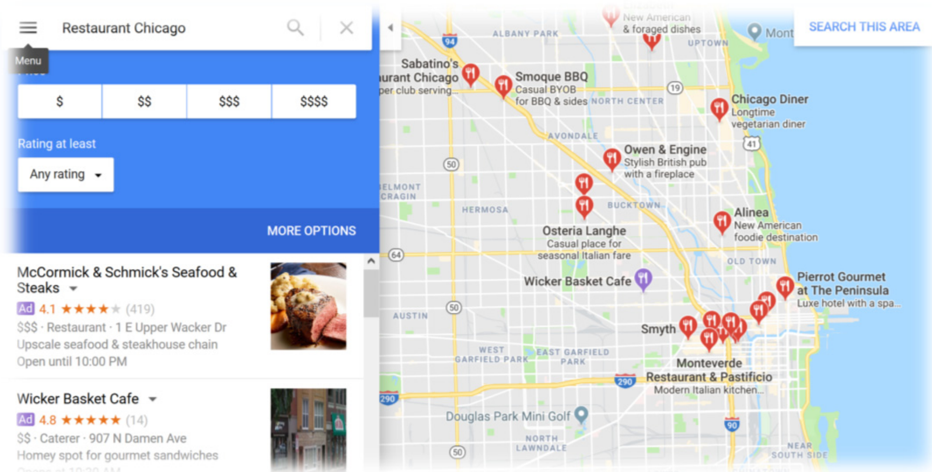


[16] Morris Riedel, 'Introduction to Machine Learning Algorithms', Invited YouTube Lecture, six lectures, University of Ghent, 2017

- Note that this course is not a full machine learning course but rather focusses on applications

Food Inspection in Chicago: Advanced Application Example

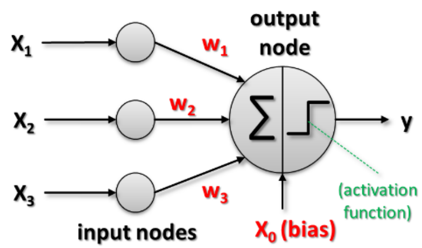
1. Some pattern exists:
   - Believe in a pattern with ‘quality violations in checking restaurants‘ will somehow influence if food inspection pass or fail (binary classification)
2. No exact mathematical formula
   - To the best of our knowledge there is no precise formula for this problem
3. Data exists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



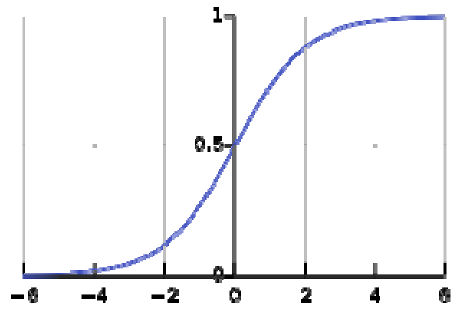
- **The goal of the advanced machine learning application with food inspection of restaurants in the City of Chicago is to predict the outcome of food inspection of new Chicago restaurants given some of existing violations of other restaurants already obtained in Chicago**

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
  - Different from above: 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



Big Data: Python/NumPy & Vectorization Not Enough

- 'Small-scale example' of the power of 'parallelization'
  - Enables element-wise computations at the same time (aka in parallel)
  - 'small-scale' since we are still within one computer – but perform operations in parallel on different data

$h(\mathbf{x}) = s\ (\mathbf{w}^T\mathbf{x})$ (logistic regression)

(vector notation, using T = transpose)

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

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

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

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

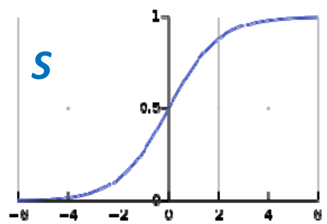
(equivalent dotproduct notation)

```
In [ ]: import numpy as np

        ...

        print(np.dot(w.T,x))
```

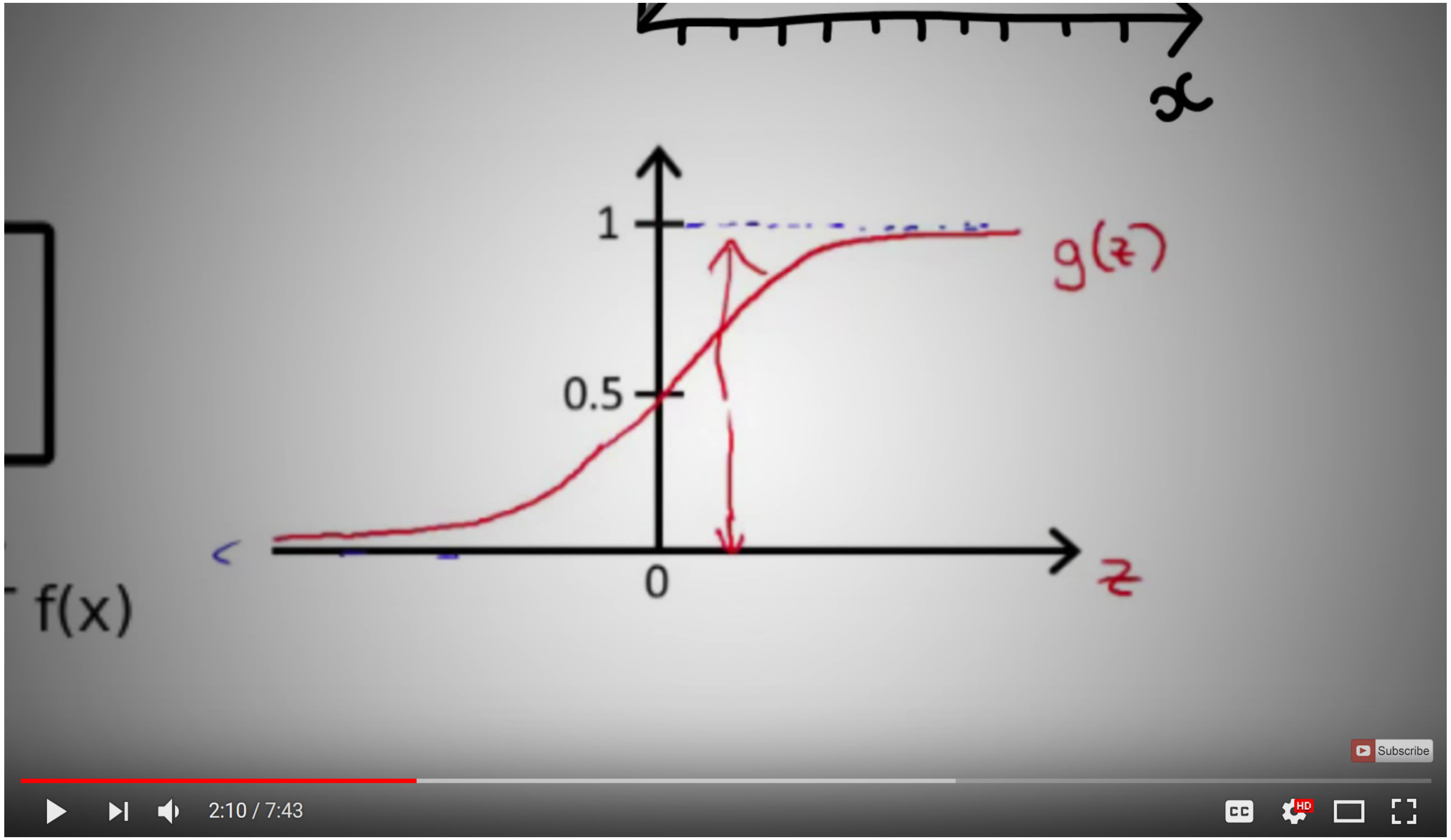
(np.dot() is a vectorized function that is fast, but still not fast enough when facing big data sets)



(sigmoid function to bring all values towards probability between 0 and 1)

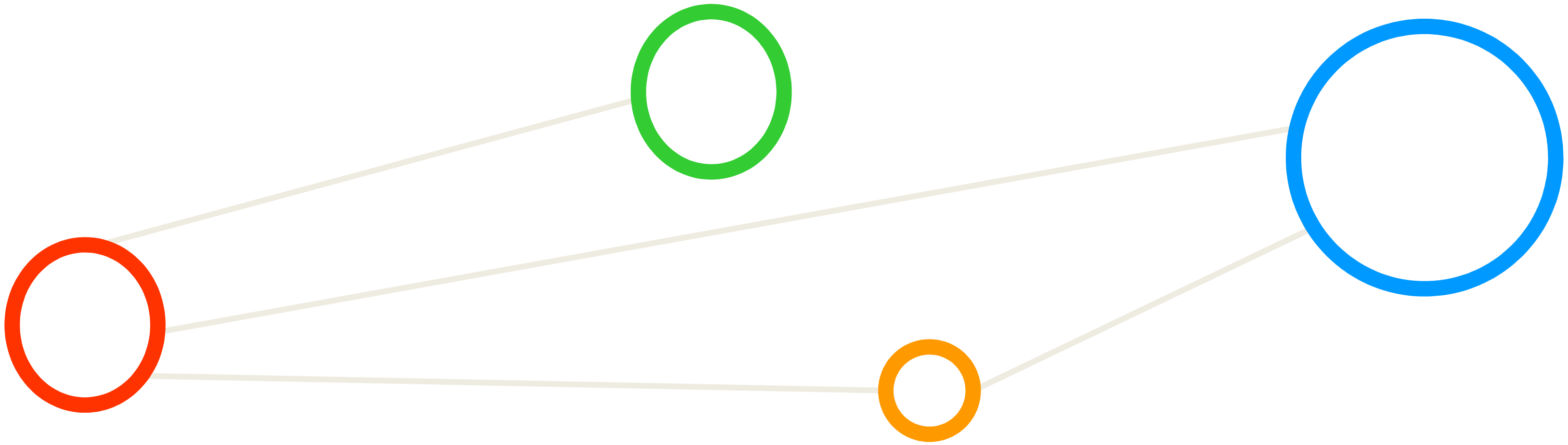
- **Challenges for Big Data in real life scenarios require a large-scale & elastic Cloud infrastructure**
- **Vectorization matter in small-scale per CPU/node, but large-scale parallelization is also important**

[Video] Logistic Regression in Short

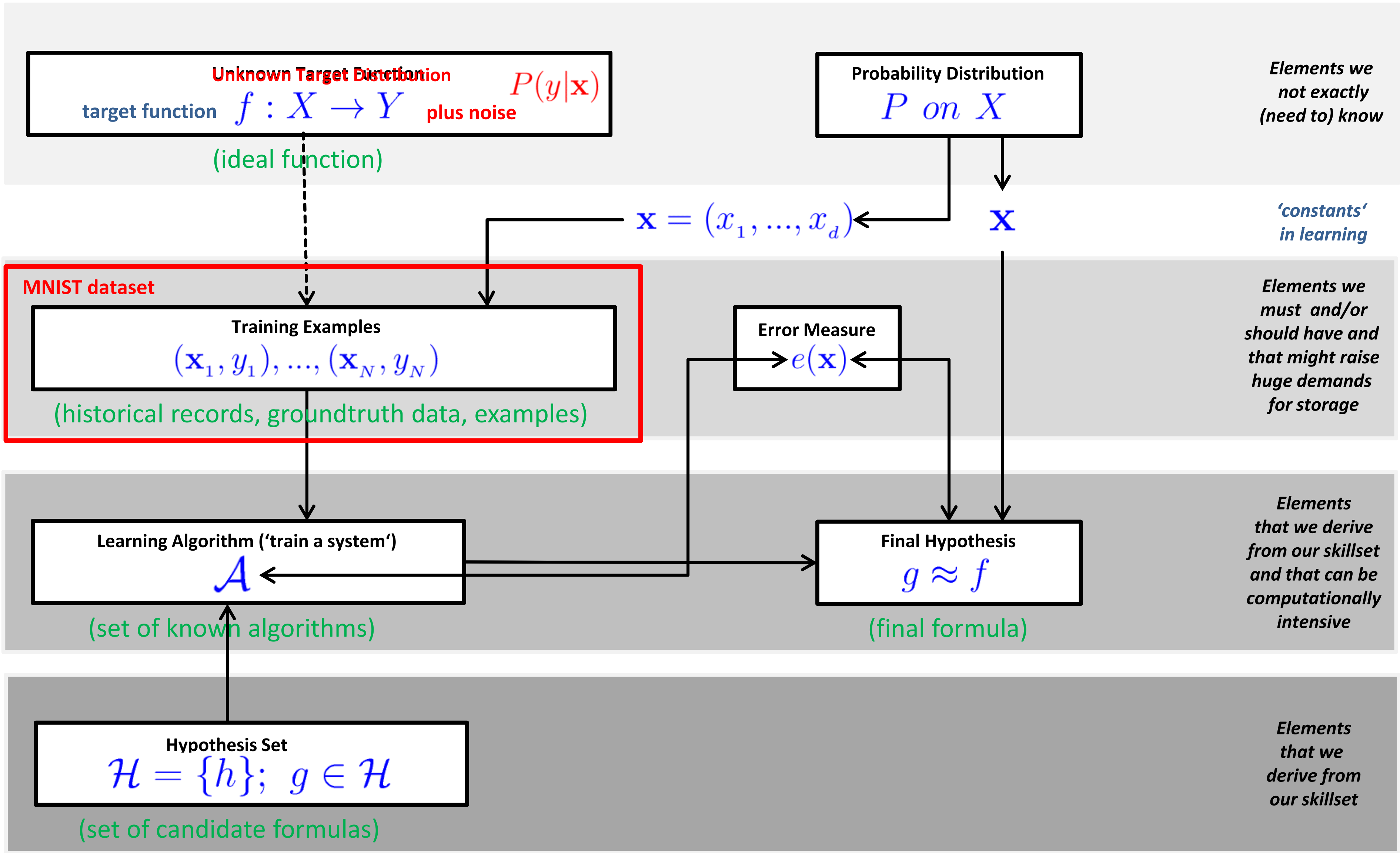


[17] YouTube, Logistic Regression

Learning from Data



Supervised Learning – MNIST Example Overview



ANN – Handwritten Character Recognition MNIST Dataset

(1) Problem Understanding Phase

- Metadata
  - Subset of a larger dataset from US National Institute of Standards (NIST)
  - Handwritten digits including corresponding labels with values 0 to 9
  - All digits have been size-normalized to 28 * 28 pixels
    and are centered in a fixed-size image for direct processing
  - Not very challenging dataset, but good for experiments / tutorials
- Dataset Samples
  - Labelled data (10 classes)
  - Two separate files
    for training and test
  - 60000 training samples (~47 MB)
  - 10000 test samples (~7.8 MB)

(10 class classification problem)

(2) Data Understanding Phase

MNIST Dataset – Data Access

(2) Data Understanding Phase

- 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
- Available for the tutorial
  - Easy download via Keras from an Amazon Web Services (AWS) cloud

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

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

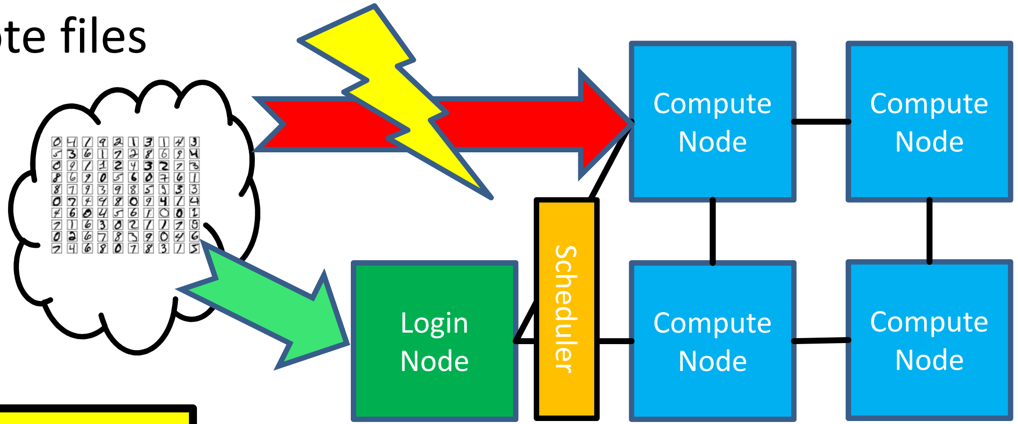
```python
# download and shuffled as training and testing set
(X_train, y_train), (X_test, y_test) = mnist.load_data()
```

```
[riedell@juronl-adm datasets]$ pwd
/p/home/jusers/riedell/juron/.keras/datasets
[riedell@juronl-adm datasets]$ ls -al
total 11234
drwxr-xr-x 2 riedell jusers     4096 Jan 20 22:05 .
drwxr-xr-x 3 riedell jusers     4096 Jan 20 22:03 ..
-rw-r--r-- 1 riedell jusers 11490434 Jan 20 22:05 mnist.npz
```

MNIST Dataset – Data Access & HPC Challenges

(2) Data Understanding Phase

- Warning for HPC environments
  - Note that HPC batch nodes often do not allow for download of remote files




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

```
Downloading data from https://s3.amazonaws.com/img-datasets/mnist.npz
11493376/11490434 [==============================] - 6s 1us/step
```

- **A useful workaround for download remotely stored datasets and files is to start the Keras script on the login node and after data download stop the script for a proper execution on batch nodes for training & inference**

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

```
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()
```

```
[riedell@juronl-adm datasets]$ pwd
/p/home/jusers/riedell/juron/.keras/datasets
[riedell@juronl-adm datasets]$ ls -al
total 11234
drwxr-xr-x 2 riedell jusers     4096 Jan 20 22:05 .
drwxr-xr-x 3 riedell jusers     4096 Jan 20 22:03 ..
-rw-r--r-- 1 riedell jusers 11490434 Jan 20 22:05 mnist.npz
```

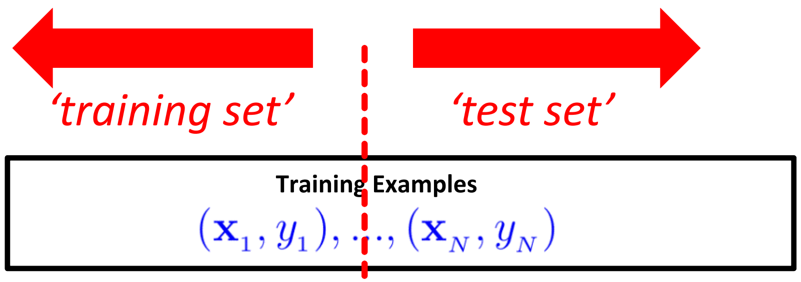
MNIST Dataset – Training and Testing Datasets

(3) Data Preparation Phase

- Different Phases in Learning
  - Training phase is a hypothesis search
  - Testing phase checks if we are on right track (once the hypothesis clear)
  - Validation phase for model selection (more details later in tutorial)
- Start 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'

```
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()
```



(historical records, groundtruth data, examples)

MNIST Dataset – Exploration – One Character Encoding

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	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 – Data Exploration Script Training Data

```
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()
```

```
# 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)
```

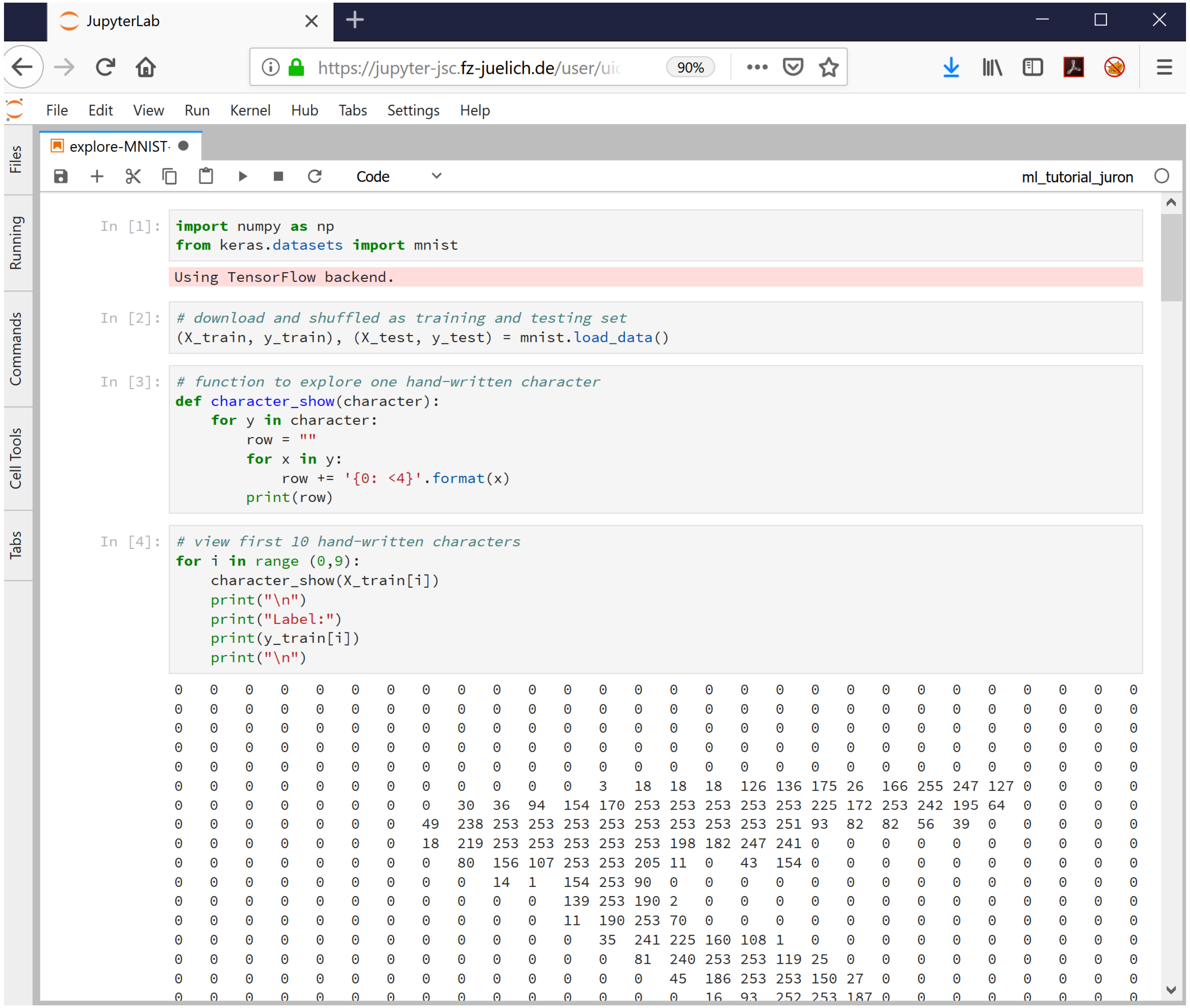
```
# view first 10 hand-written characters
for i in range (0,9):
    character_show(X_train[i])
    print("\n")
    print("Label:")
    print(y_train[i])
    print("\n")
```

- 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)

- 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)

MNIST Dataset – Data Exploration via Jupyter



MNIST Dataset – Exploration – Selected Training Samples

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	3	18	18	18	126	136	175	26	166	255	247	127	0	0	0	0
0	0	0	0	0	0	0	0	30	36	94	154	170	253	253	253	253	253	225	172	253	242	195	64	0	0	0	0
0	0	0	0	0	0	0	49	238	253	253	253	253	253	253	253	253	251	93	82	82	56	39	0	0	0	0	0
0	0	0	0	0	0	0	18	219	253	253	253	253	253	198	182	247	241	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	80	156	107	253	253	205	11	0	43	154	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	14	1	154	253	90	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	139	253	190	2	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	11	190	253	70	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	35	241	225	160	108	1	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	81	240	253	253	119	25	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	45	186	253	253	150	27	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	16	93	252	253	187	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	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

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

Label:
0

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

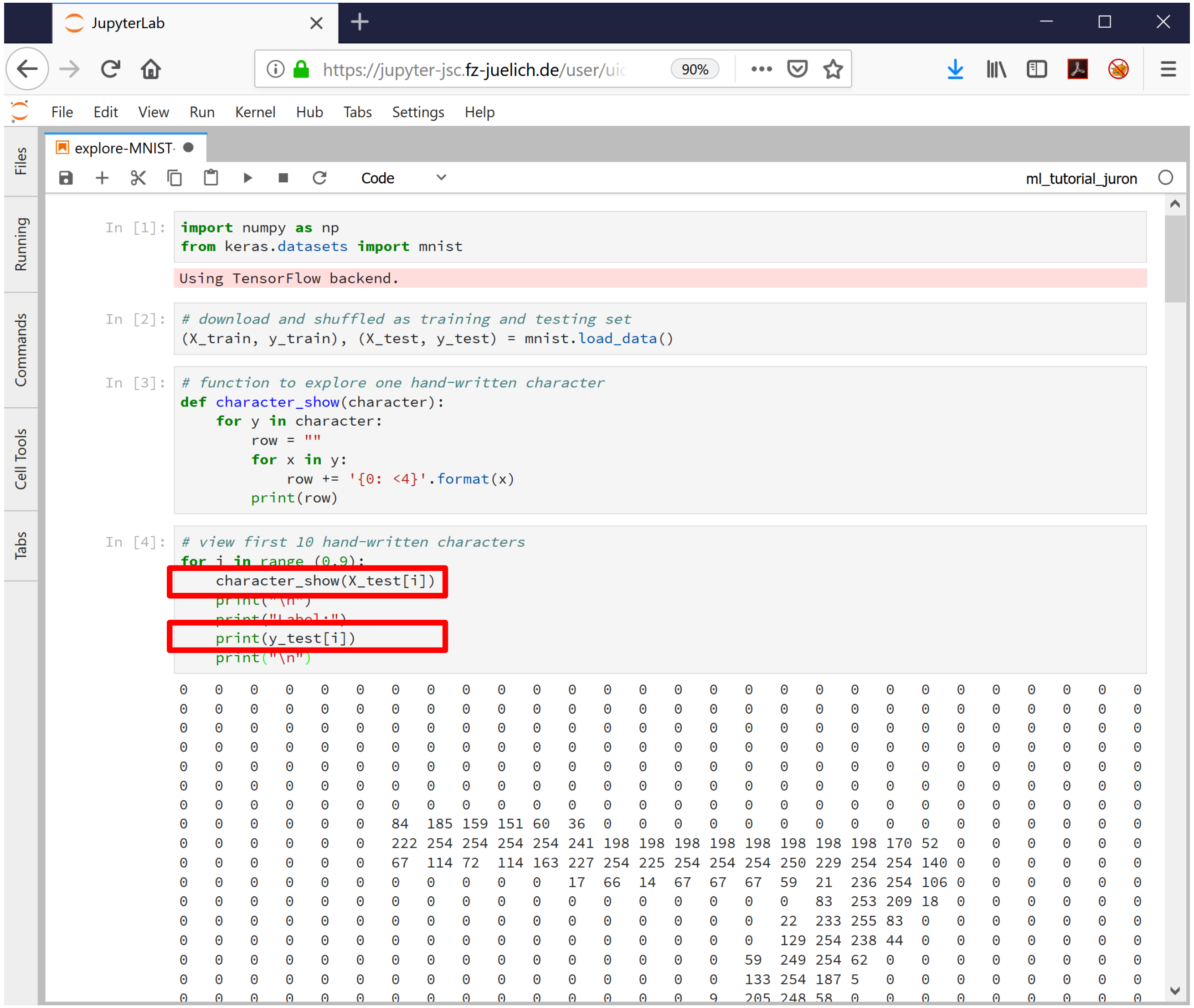
Label:
4

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

Label:
2

Exercises – Explore Testing Data

Exercises – Explore Testing Data – Solution



MNIST Dataset & Perceptron Model

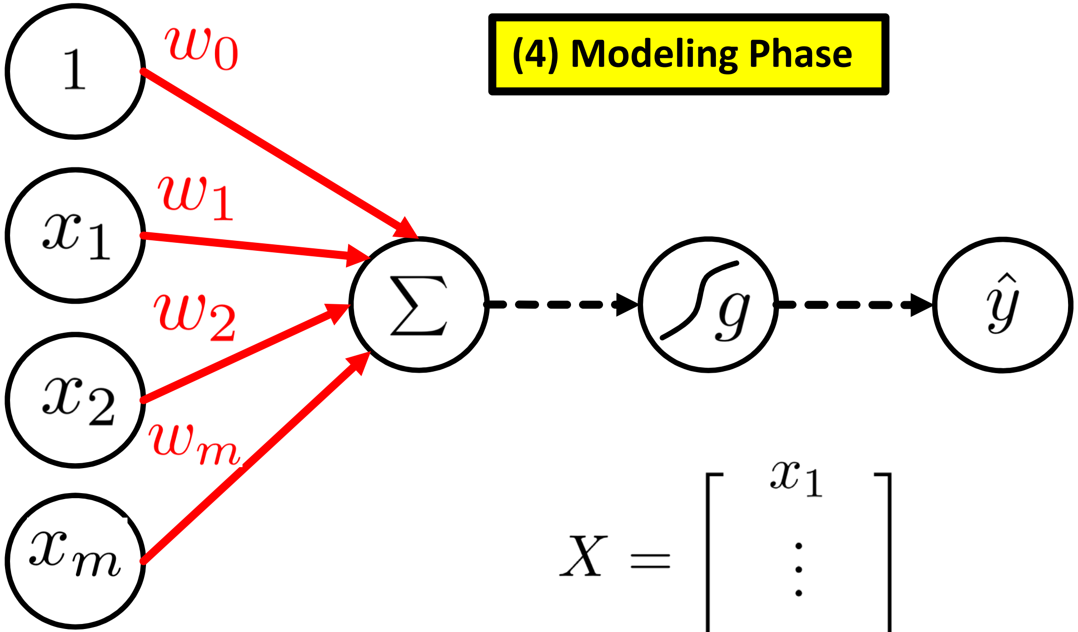
- Two dimensional dataset (28 x 28)
  - Does not fit well with input to Perceptron Model

(2) Data Understanding Phase

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	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

(4) Modeling Phase



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

- Consequence
  - We need to prepare the data even more → we need one long vector

MNIST Dataset – Reshape & Normalization

```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()

# reshape for input to perceptron 28 x 28 = 784
RESHAPED = 784
X_train = X_train.reshape(60000, RESHAPED)
X_test = X_test.reshape(10000, RESHAPED)
# float32 for GPU execution
X_train = X_train.astype('float32')
X_test = X_test.astype('float32')

# normalization
X_train /= 255
X_test /= 255

# data exploration: number of samples
print(X_train.shape[0], 'train samples')
print(X_test.shape[0], 'test samples')

# data exploration: number of values / samples
print(X_train.shape[1], 'input pixel values per train samples')
print(X_test.shape[1], 'input pixel values per test samples')

# data output: vectorized character
print(X_train[0])
```

(3) Data Preparation Phase

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

- **Reshape from 28 x 28 matrix of pixels to 784 pixel values considered to be the input for the neural networks later**
- **Normalization is added for mathematical convenience since computing with numbers get easier (not too large)**

MNIST Dataset – Reshape & Normalization – Example

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

```
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
```

(one long input vector)

(3) Data Preparation Phase

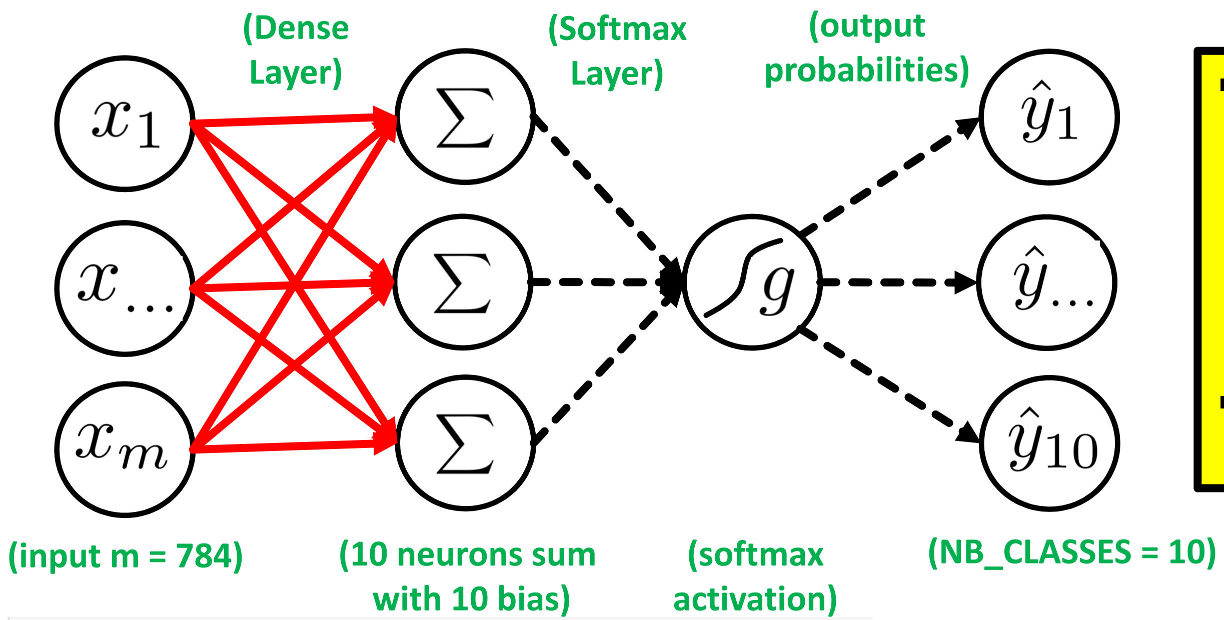
(numbers are between 0 and 1)

Exercises – Perform Data Reshaping & Normalization

MNIST Dataset & Multi Output Perceptron Model

- 10 Class Classification Problem
  - Use 10 Perceptrons for 10 outputs with softmax activation function

(4) Modeling Phase



- **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**

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

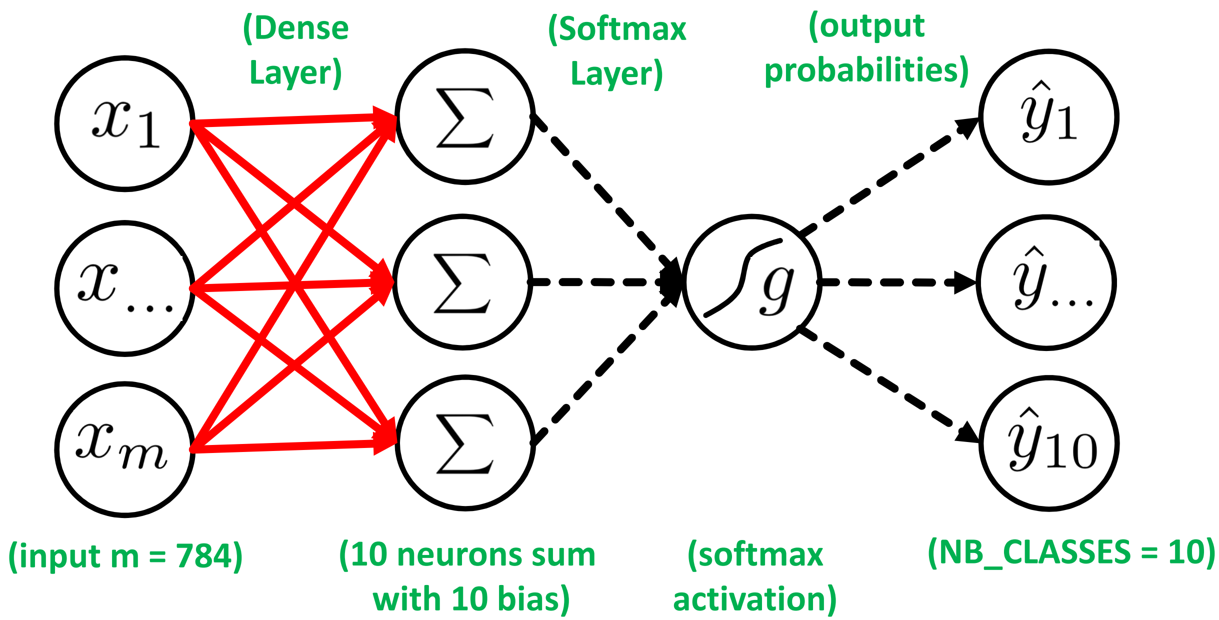
```python
# model Keras sequential
model = Sequential()
```

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

MNIST Dataset & Activation Function Softmax

(4) Modeling Phase

- Activation Function Softmax
  - Softmax enables probabilities for 10 classes




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

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

$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}$$

- **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**

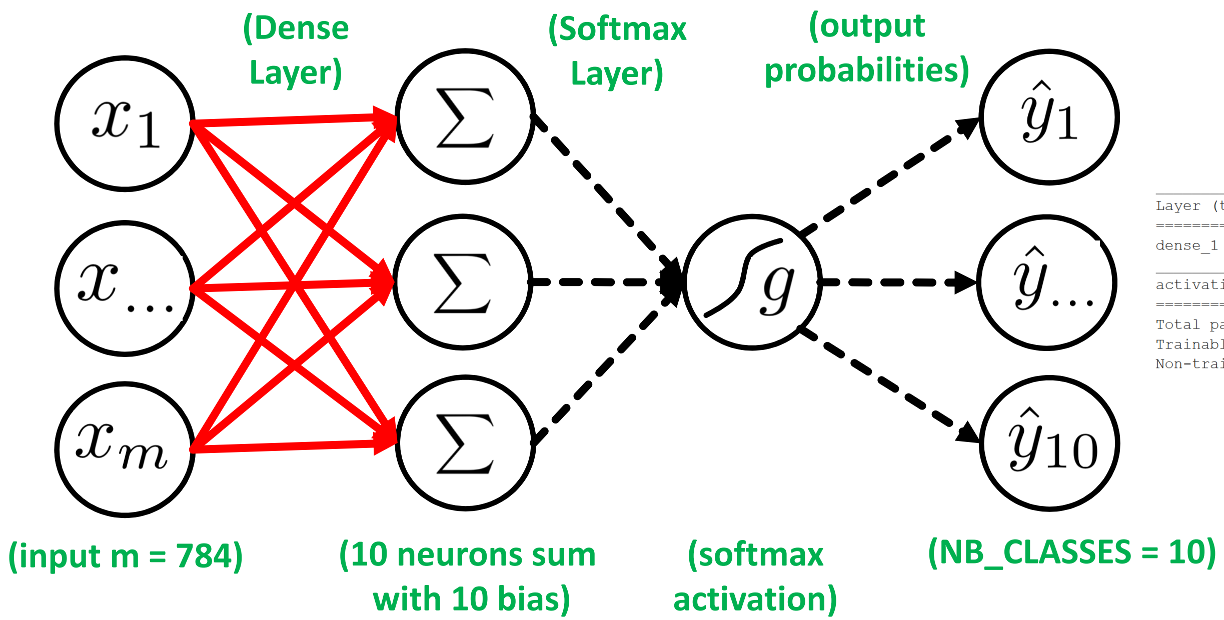
AUDIENCE QUESTION

How many parameters we have to learn and why?

MNIST Dataset & Model Summary & Parameters

- Activation Function Softmax
  - Softmax enables probabilities for 10 classes

(4) Modeling Phase



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

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

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

MNIST Dataset & Compile the Model

- Compile the model
  - Choose 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
- Specify loss function
  - Compare prediction vs. Given class label
  - E.g. categorical crossentropy

(4) Modeling Phase

- **Compile the model to be executed by the Keras backend (e.g. TensorFlow)**

```
In [1]: 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

Using TensorFlow backend.
```

- **Loss function is a multi-class logarithmic loss: target is *ti,j* and prediction is *pi,j***
- **Categorical crossentropy is very suitable for multiclass label predictions (default with softmax)**

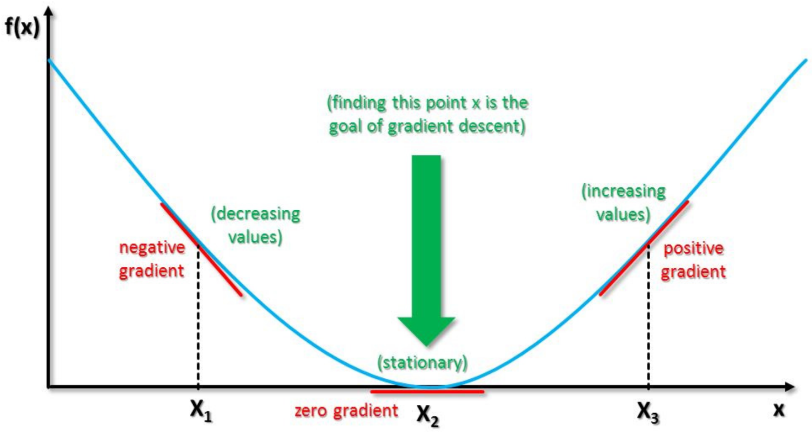
```
# specify loss, optimizer and metric
model.compile(loss='categorical_crossentropy', optimizer=OPTIMIZER, metrics=['accuracy'])
```

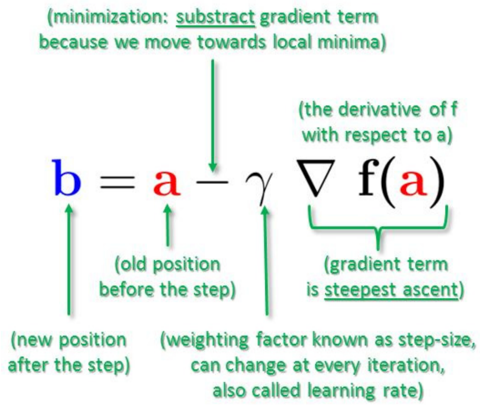
$$L_i = -\Sigma_j t_{i,j} \log(p_{i,j})$$

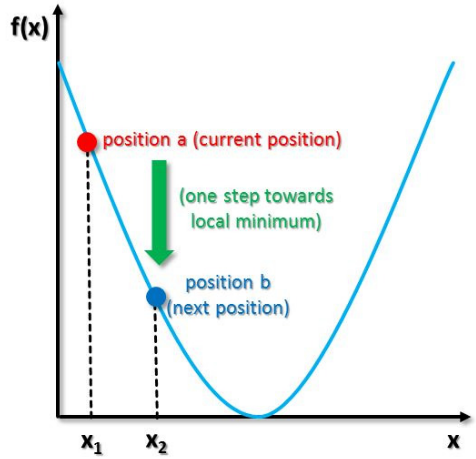
MNIST Dataset & Optimization / Learning Approach

- Choosing an Optimizer
  - Example: Stochastic Gradient Descent (SGD)

(4) Modeling Phase







[4] Big Data Tips, Gradient Descent

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

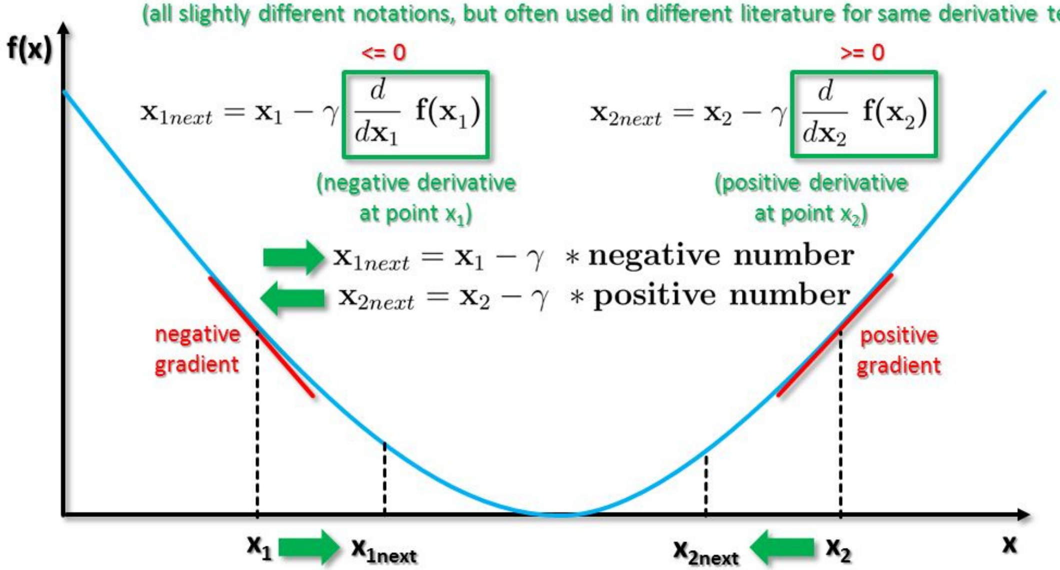
```python
OPTIMIZER = SGD() # optimization technique
```

MNIST Dataset & Stochastic Gradient Descent Method

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

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

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

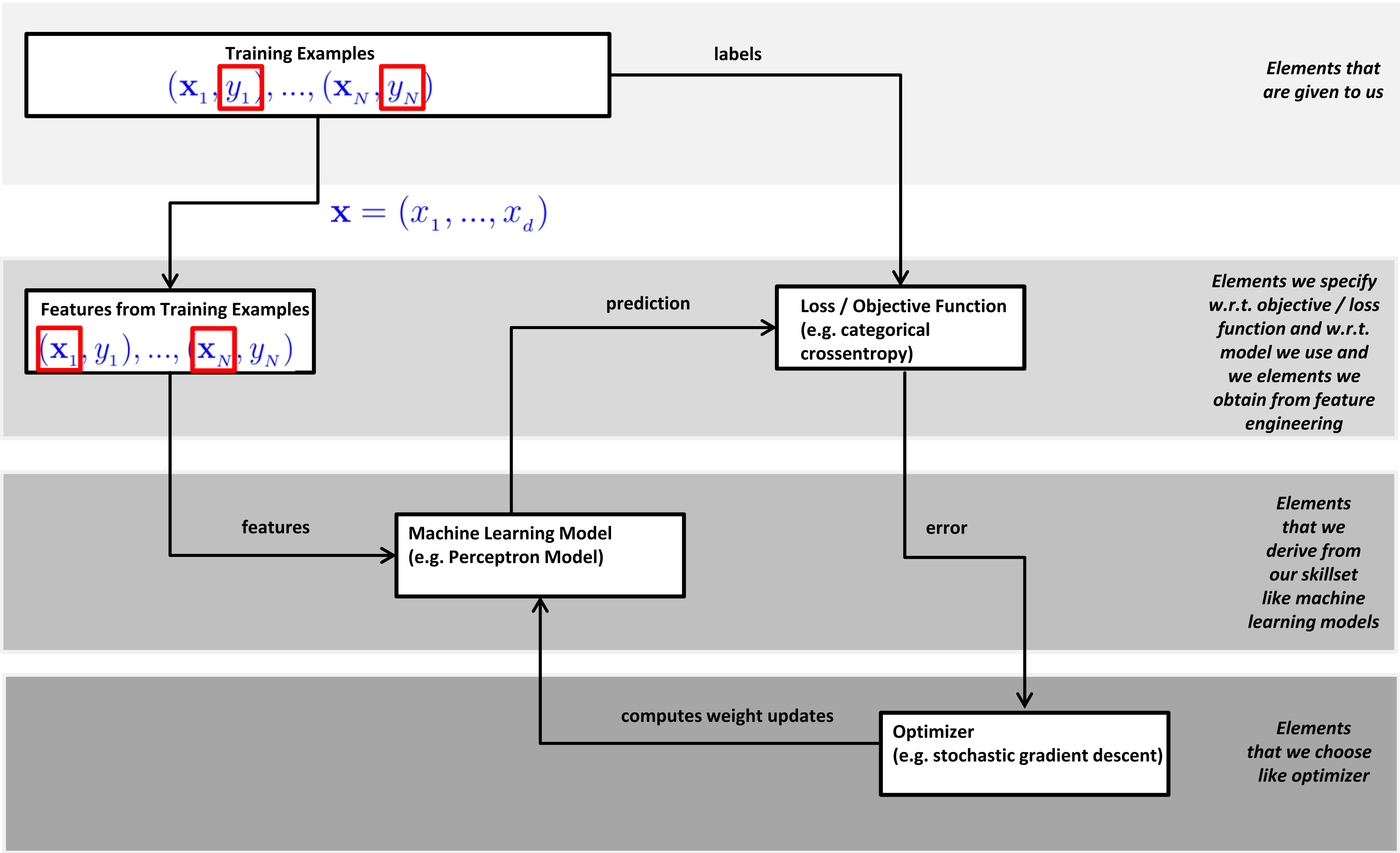



```
from keras.optimizers import SGD
```

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

[4] Big Data Tips, Gradient Descent

Optimizer – Effect to the Model – Overview & SGD Example



MNIST Dataset – Model Parameters & Data Normalization

```python
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
```

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

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

```python
# 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')
```

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

```python
# 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**
- **BATCH_SIZE: number of training instances taken into account before the optimizer performs a weight update to the model**
- **OPTIMIZER: Stochastic Gradient Descent ('SGD') – only one training sample/iteration**

- **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]**

MNIST Dataset – A Multi Output Perceptron Model

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

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

- The non-linear activation function 'softmax' 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

```
# 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])
```

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

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

- Train the model ('fit')

Exercises – Multi Output Perceptron Model & 20 Epochs

Model Evaluation – Testing Phase & Confusion Matrix

- Model is fixed
  - Model is just used with the testset
  - Parameters are set
- Evaluation of model performance
  - Counts of test records that are incorrectly predicted
  - Counts of test records that are correctly predicted
  - E.g. create confusion matrix for a two class problem

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

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

Model Evaluation – Testing Phase & Performance Metrics

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

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

- Accuracy (usually in %)

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

- Error rate

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

Exercises – Evaluate Multi Output Perceptron Model

MNIST Dataset – A Multi Output Perceptron Model – Output

```
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])
```

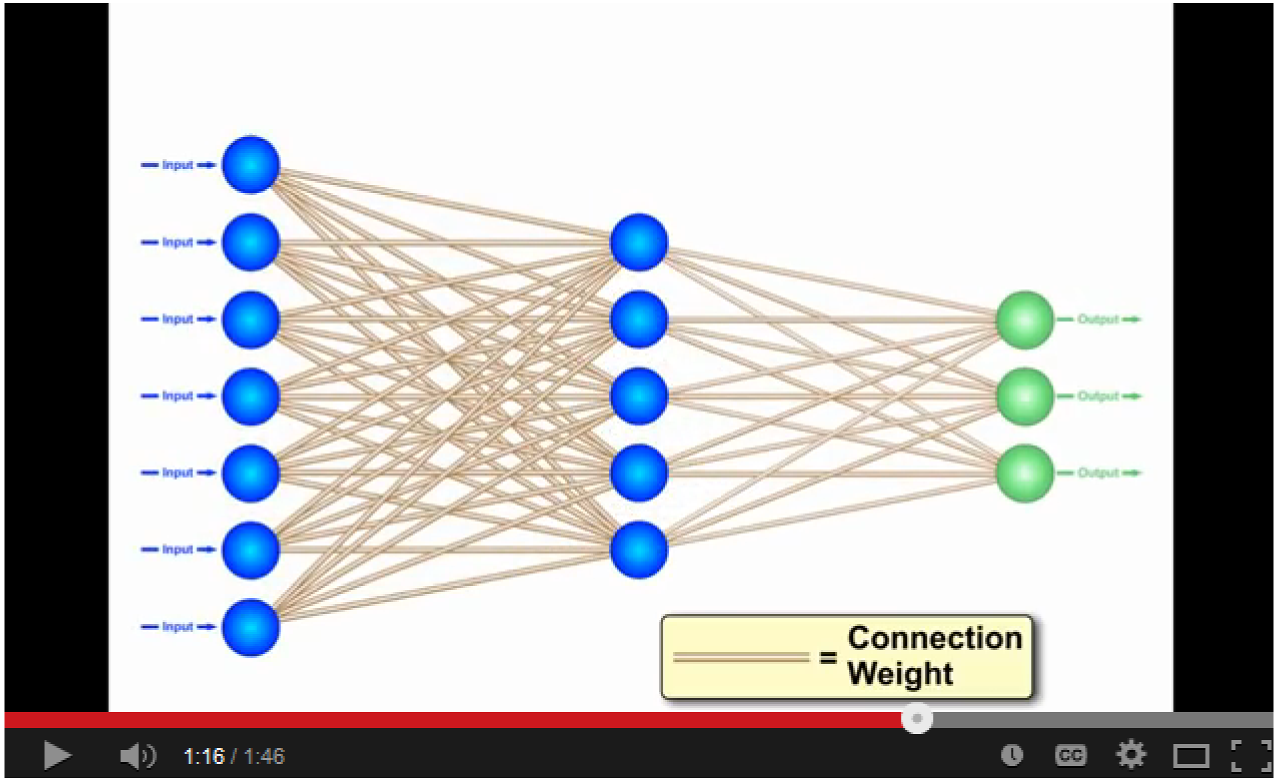
```
10000/10000 [==============================] - 0s 41us/step
Test score: 0.33423959468007086
Test accuracy: 0.9101
```

AUDIENCE QUESTION

What would you change to get better accuracy?

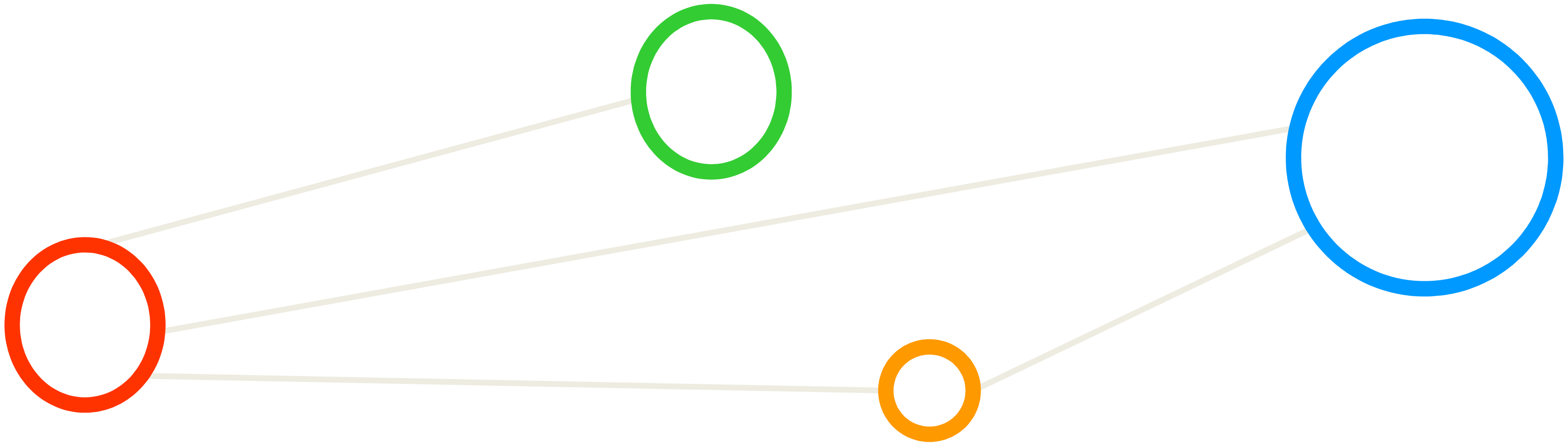
Exercises – Multi Output Perceptron Model & 50 Epochs

[Video] Towards Multi-Layer Perceptrons



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

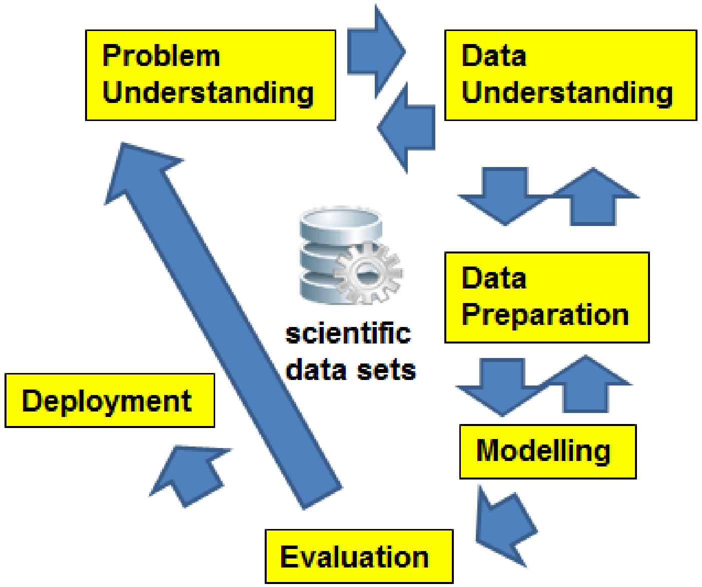
Appendix A: CRISP-DM Process



Summary: Systematic Process

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

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



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

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

1 – Problem (Business) Understanding

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

[21] CRISP-DM User Guide

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

2 – Data Understanding

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

[21] CRISP-DM User Guide

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

3 – Data Preparation

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

[21] CRISP-DM User Guide

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

4 – Modeling

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

[21] CRISP-DM User Guide

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

5 – Evaluation

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

[21] CRISP-DM User Guide

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

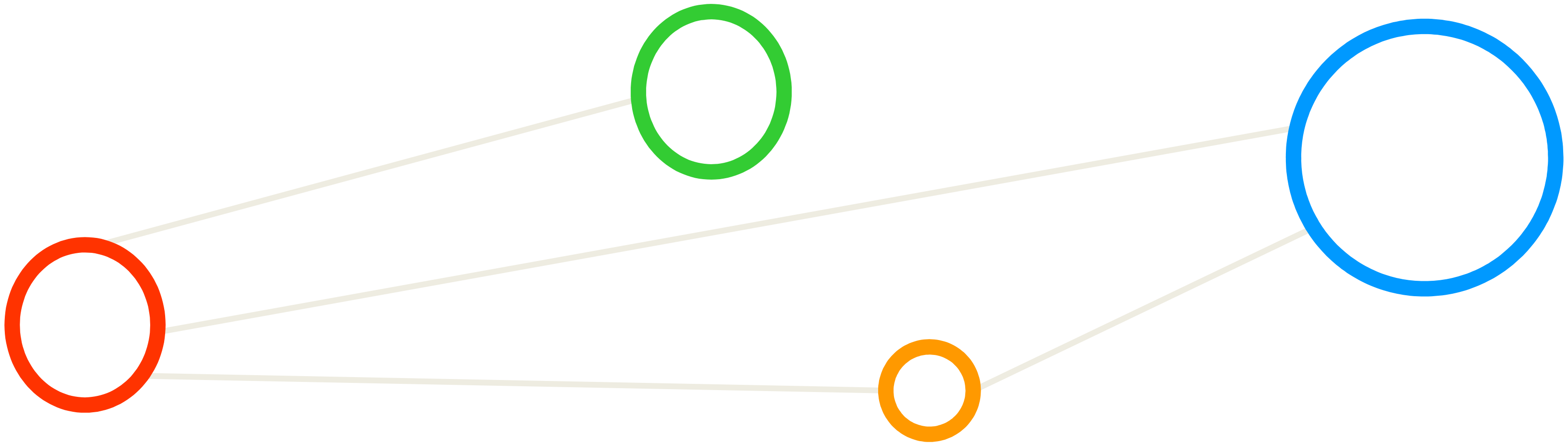
6 – Deployment

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

[21] CRISP-DM User Guide

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

Lecture Bibliography



Lecture Bibliography (1)

- [1] Lego Bricks Images,
  Online: https://tucsonbotanical.org/wp-content/uploads/2015/08/You-build-it_image.jpg
- [2] F. Rosenblatt, 'The Perceptron--a perceiving and recognizing automaton',
  Report 85-460-1, Cornell Aeronautical Laboratory, 1957
- [3] Keras Python Deep Learning Library,
  Online: https://keras.io/
- [4] Big Data Tips, 'Gradient Descent',
  Online: http://www.big-data.tips/gradient-descent
- [5] Keras Python High-Level Deep Learning Library,
  Online: https://keras.io/
- [6] TensorFlow Python Low-Level Deep learning Library,
  Online: https://www.tensorflow.org/
- [7] NVIDIA Web Page,
  Online: https://www.nvidia.com/en-us/
- [8] K. Hwang, G. C. Fox, J. J. Dongarra, 'Distributed and Cloud Computing', Book,
  Online: http://store.elsevier.com/product.jsp?locale=en_EU&isbn=9780128002049
- [9] An Introduction to Statistical Learning with Applications in R,
  Online: http://www-bcf.usc.edu/~gareth/ISL/index.html
- [10] Species Iris Group of North America Database,
  Online: http://www.signa.org

Lecture Bibliography (2)

- [11] UCI Machine Learning Repository Iris Dataset,
  Online: https://archive.ics.uci.edu/ml/datasets/Iris
- [12] Wikipedia 'Sepal',
  Online: https://en.wikipedia.org/wiki/Sepal
- [13] F. Rosenblatt, 'The Perceptron--a perceiving and recognizing automaton',
  Report 85-460-1, Cornell Aeronautical Laboratory, 1957
- [14] Rosenblatt,'The Perceptron: A probabilistic model for information storage and orgainzation in the brain',
  Psychological Review 65(6), pp. 386-408, 1958
- [15] YouTube Video, 'Perceptron, Linear'
  Online: https://www.youtube.com/watch?v=FLPvNdwC6Qo
- [16] Morris Riedel, 'Introduction to Machine Learning Algorithms', Invited YouTube Lecture, six lectures
  University of Ghent, 2017,
  Online: https://www.youtube.com/watch?v=KgiuUZ3WeP8&list=PLrmNhuZo9sgbcWtMGN0i6G9HEvh08JG0J
- [17] YouTube Video, 'Logistic Regression - Fun and Easy Machine Learning',
  Online: https://www.youtube.com/watch?v=7qJ7GksOXoA
- [18] Rattle Library for R,
  Online: http://rattle.togaware.com/
- [19] B2Share, 'Iris Dataset LibSVM Format Preprocessing',
  Online: https://b2share.eudat.eu/records/37fb24847a73489a9c569d7033ad0238
- [10] C. Shearer, CRISP-DM model, Journal Data Warehousing, 5:13

Lecture Bibliography (3)

- [21] Pete Chapman, 'CRISP-DM User Guide', 1999,
  Online: http://lyle.smu.edu/~mhd/8331f03/crisp.pdf
- [22] YouTube Video, 'Neural Networks, A Simple Explanation',
  Online: http://www.youtube.com/watch?v=gcK_5x2KsLA

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