

Supercomputing and Big Data

Parallel and Scalable 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

Parallel and Scalable Classification using SVMs with Applications

July 26th, 2018, NextGen@Helmholtz Conference

GFZ German Research Centre for Geosciences Potsdam, Germany

HELMHOLTZ
RESEARCH FOR GRAND CHALLENGES



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



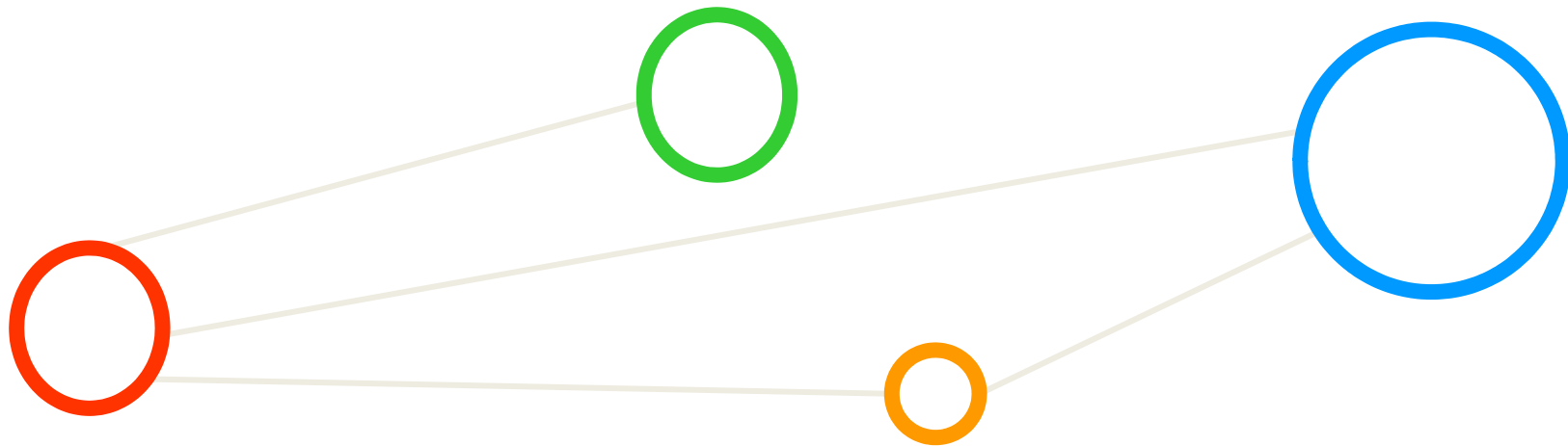
JÜLICH
Forschungszentrum

JÜLICH
SUPERCOMPUTING
CENTRE


NextGen
@HELMHOLTZ

DEEP
Projects

Outline



Outline of the Course

1. HPC Introduction & Parallel and Scalable Clustering using DBSCAN
2. Parallel and Scalable Classification using SVMs with Applications
3. Deep Learning using CNNs driven by HPC & GPUs
4. Deep Learning using LSTMs driven by HPC & GPUs

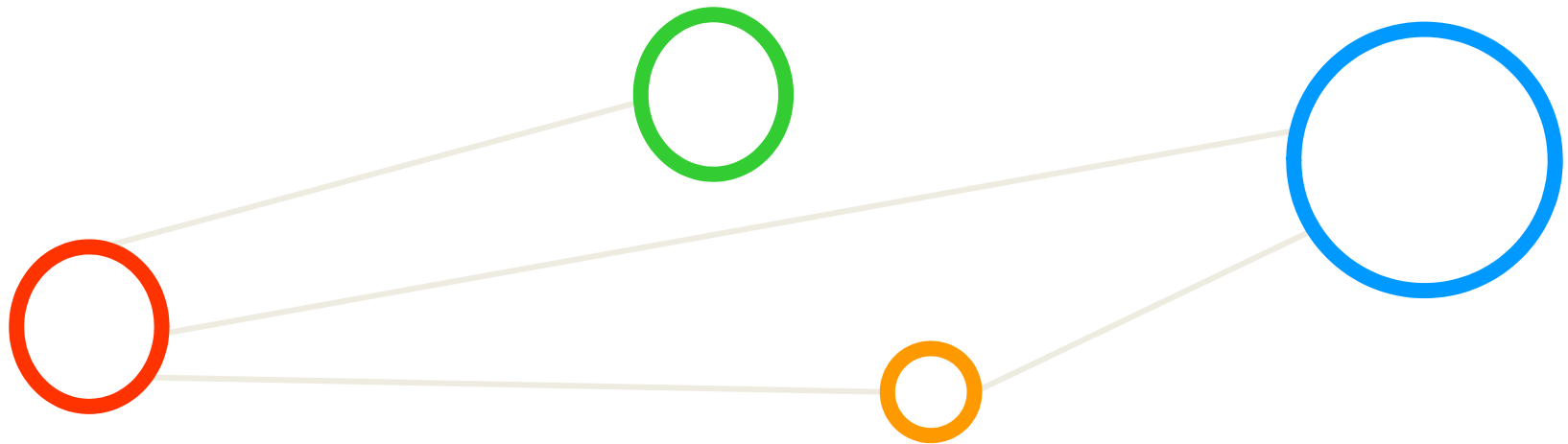


Outline

- Supervised Classification
 - Simple Example with Linear Perceptron Model
 - Data-Preprocessing
 - Learning Approaches & Mathematical Building Blocks
 - Training and Testing
 - Selected Challenges
- Application Examples
 - Remote Sensing Dataset
 - Rome and Indian Pines
 - Support Vector Machines
 - Parallel and Scalable SVM piSVM
 - Non-linear Transformation and Kernel Methods



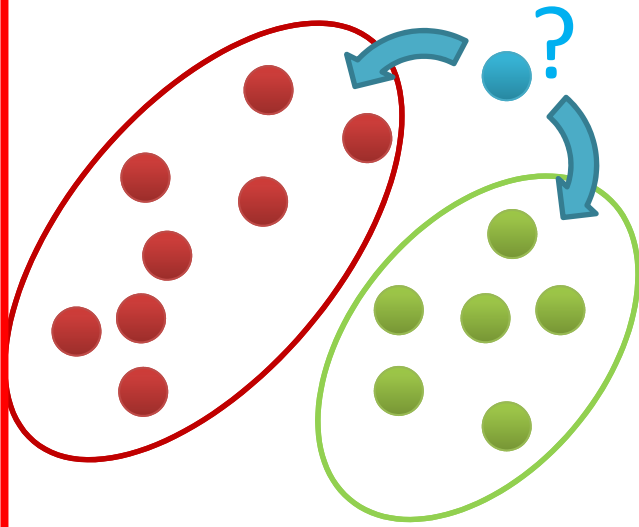
Supervised Classification



Methods Overview

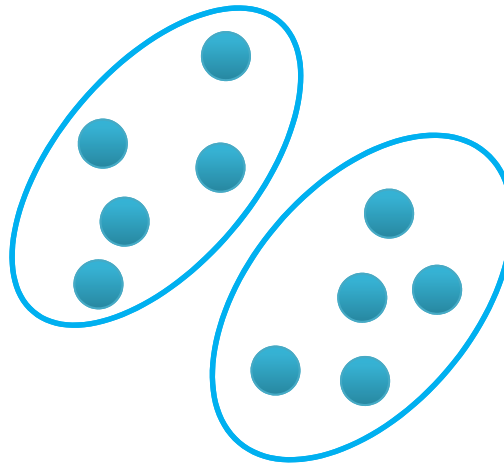
- Machine learning methods can be roughly categorized in classification, clustering, or regression augmented with various techniques for data exploration, selection, or reduction

Classification



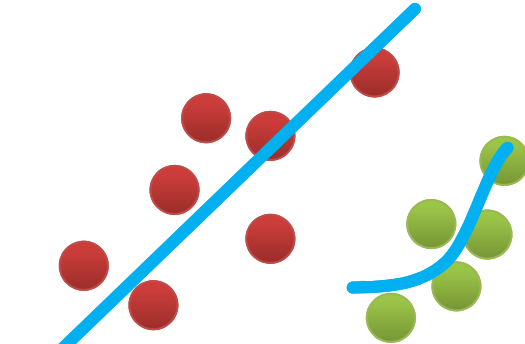
- Groups of data exist
- New data classified to existing groups

Clustering



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

Regression

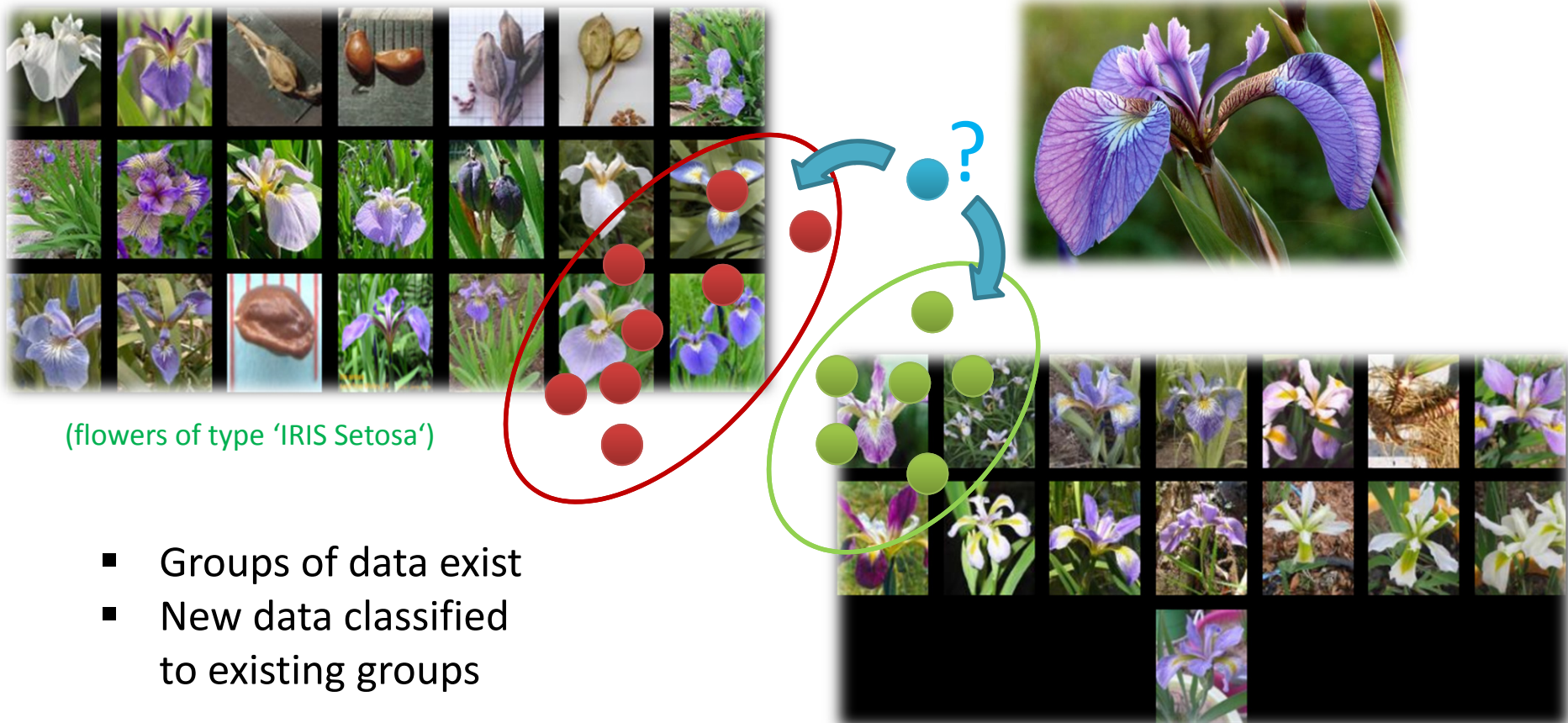


- Identify a line with a certain slope describing the data

Simple Application Example: Classification of a Flower

(1) Problem Understanding Phase

(what type of flower is this?)



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

[1] 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')



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

Learning problem: A prediction task

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



(what type of flower is this?)

Feasibility of Machine Learning in this Example

1. Some pattern exists:

- Believe in a 'pattern with 'petal length' & 'petal width' somehow influence the type

2. No exact mathematical formula

- To the best of our knowledge there is no precise formula for this problem

3. Data exists

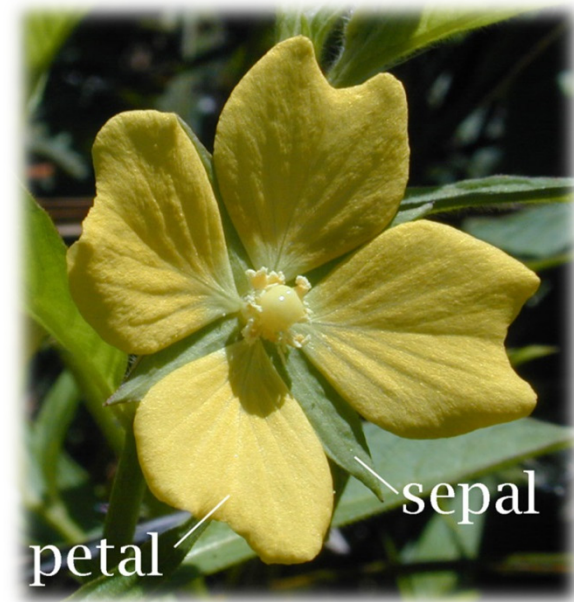
- Data collection from UCI Dataset „Iris“
- 150 labelled samples (aka 'data points')
- Balanced: 50 samples / class

(2) Data Understanding Phase

[3] UCI Machine Learning
Repository Iris Dataset

(four data attributes for each
sample in the dataset)

(one class label for each
sample in the dataset)



[2] Image source: Wikipedia, Sepal

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

Understanding the Data – Check Metadata

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

```
1. Title: Iris Plants Database
   Updated Sept 21 by C.Blake - Added discrepancy information
   (Subject, title, or context)

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

   ...

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

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

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

[3] UCI Machine Learning Repository Iris Dataset

Understanding the Data – Check Table View

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

Rattle Dataset - dcredit version 0.6.1

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

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

(four data attributes for each sample in the dataset)

(one class label for each sample in the dataset)

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

OK Cancel

[4] Rattle Library for R

Preparing the Data – Corrected Header

(3) Data Preparation Phase

Rattle Dataset - dfedit version 0.6.1

	V1	V2	V3	V4	V5
1	5.1	3.5	1.4	0.2	Iris-setosa
2	4.9	3	1.4	0.2	Iris-setosa
3	4.7	3.2	1.3	0.2	Iris-setosa
4	4.6	3.1	1.5	0.2	Iris-setosa
5	5	3.6	1.4	0.2	Iris-setosa
6	5.4	3.9	1.7	0.4	Iris-setosa
7	4.6	3.4	1.4	0.3	Iris-setosa
8	5	3.4	1.5	0.2	Iris-setosa
9	4.4	2.9	1.4	0.2	Iris-setosa
10	4.9	3.1	1.5	0.1	Iris-setosa
11	5.4	3.7	1.5	0.2	Iris-setosa
12	4.8	3.4	1.6	0.2	Iris-setosa
13	4.8	3	1.4	0.1	Iris-setosa
14	4.3	3	1.1	0.1	Iris-setosa
15	5.8	4	1.2	0.2	Iris-setosa
16	5.7	4.4	1.5	0.4	Iris-setosa
17	5.1	3.0	1.3	0.4	Iris-setosa

(correct header information, resulting in 150 data samples)

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

Project Tools Settings Help

Execute New Open Save Report Export Stop Quit

Data Explore Test Transform Cluster Associate Model Evaluate Log

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

Filename: Separator: Decimal: ☒ Header

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

OK Cancel

Preparing the Data – Remove Third Class Samples

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

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

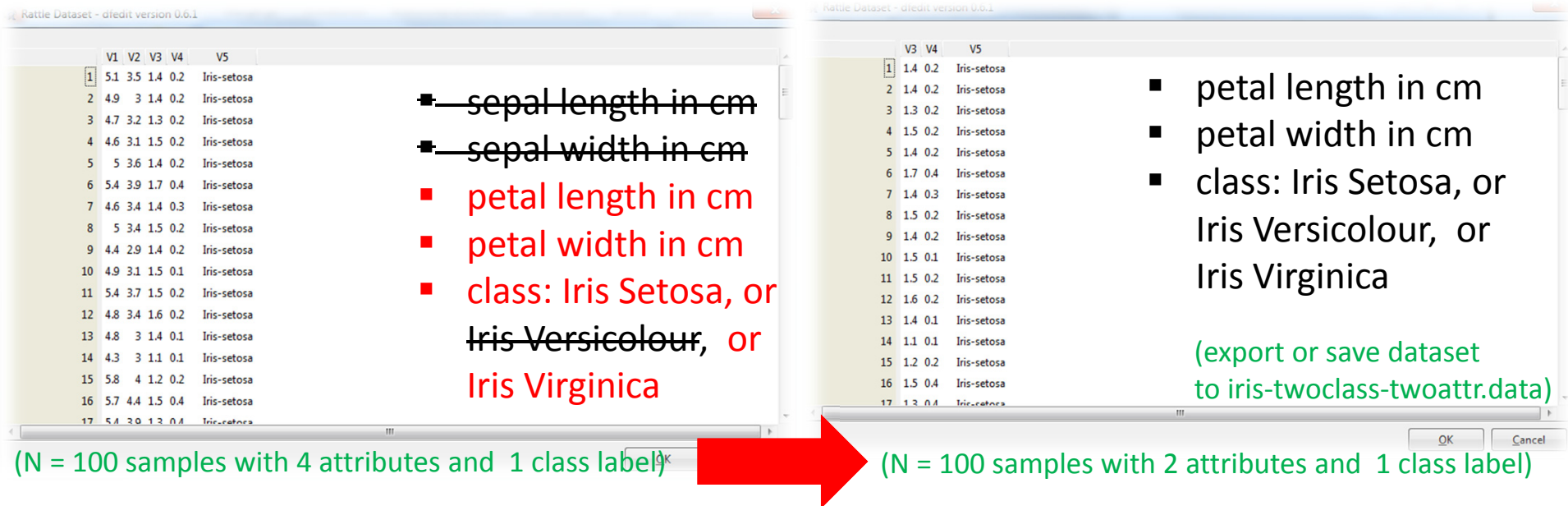
Left Window (Initial Dataset): Shows a table with columns X5.1, X3.5, X1.4, X0.2, and Iris.setosa. The data includes 150 samples, including Iris-Versicolour. A green text overlay states: "(three class problem with N = 150 samples including Iris Versicolour)". A green text overlay at the bottom states: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

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

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

Preparing the Data – Feature Selection Process

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



Left window (Full Dataset):

	V1	V2	V3	V4	V5
1	5.1	3.5	1.4	0.2	Iris-setosa
2	4.9	3	1.4	0.2	Iris-setosa
3	4.7	3.2	1.3	0.2	Iris-setosa
4	4.6	3.1	1.5	0.2	Iris-setosa
5	5	3.6	1.4	0.2	Iris-setosa
6	5.4	3.9	1.7	0.4	Iris-setosa
7	4.6	3.4	1.4	0.3	Iris-setosa
8	5	3.4	1.5	0.2	Iris-setosa
9	4.4	2.9	1.4	0.2	Iris-setosa
10	4.9	3.1	1.5	0.1	Iris-setosa
11	5.4	3.7	1.5	0.2	Iris-setosa
12	4.8	3.4	1.6	0.2	Iris-setosa
13	4.8	3	1.4	0.1	Iris-setosa
14	4.3	3	1.1	0.1	Iris-setosa
15	5.8	4	1.2	0.2	Iris-setosa
16	5.7	4.4	1.5	0.4	Iris-setosa
17	5.4	3.0	1.3	0.4	Iris-setosa

Annotations for Left window:

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

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

Right window (Selected Dataset):

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

Annotations for Right window:

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

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

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

Iris Dataset – Open Data

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

The screenshot displays the B2SHARE web interface for the 'Iris Dataset LibSVM Format Preprocessing' record. The browser address bar shows the URL: <https://b2share.eudat.eu/records/37fb24847a73489a9c569d7033ad0238>. The record is by Morris Riedel, dated Dec 22, 2016, and last updated on Jan 11, 2018. The abstract describes the UCI Machine Learning Repository IRIS Dataset, providing details on the data classes and sampling. The keywords are LibSVM, Iris, Flowers, and UCI. The PID is 11304/10e216d4-0a98-4ab4-86ea-75ed05ee0f46. The interface is divided into two main sections: 'Files' and 'Basic metadata'.

Files

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

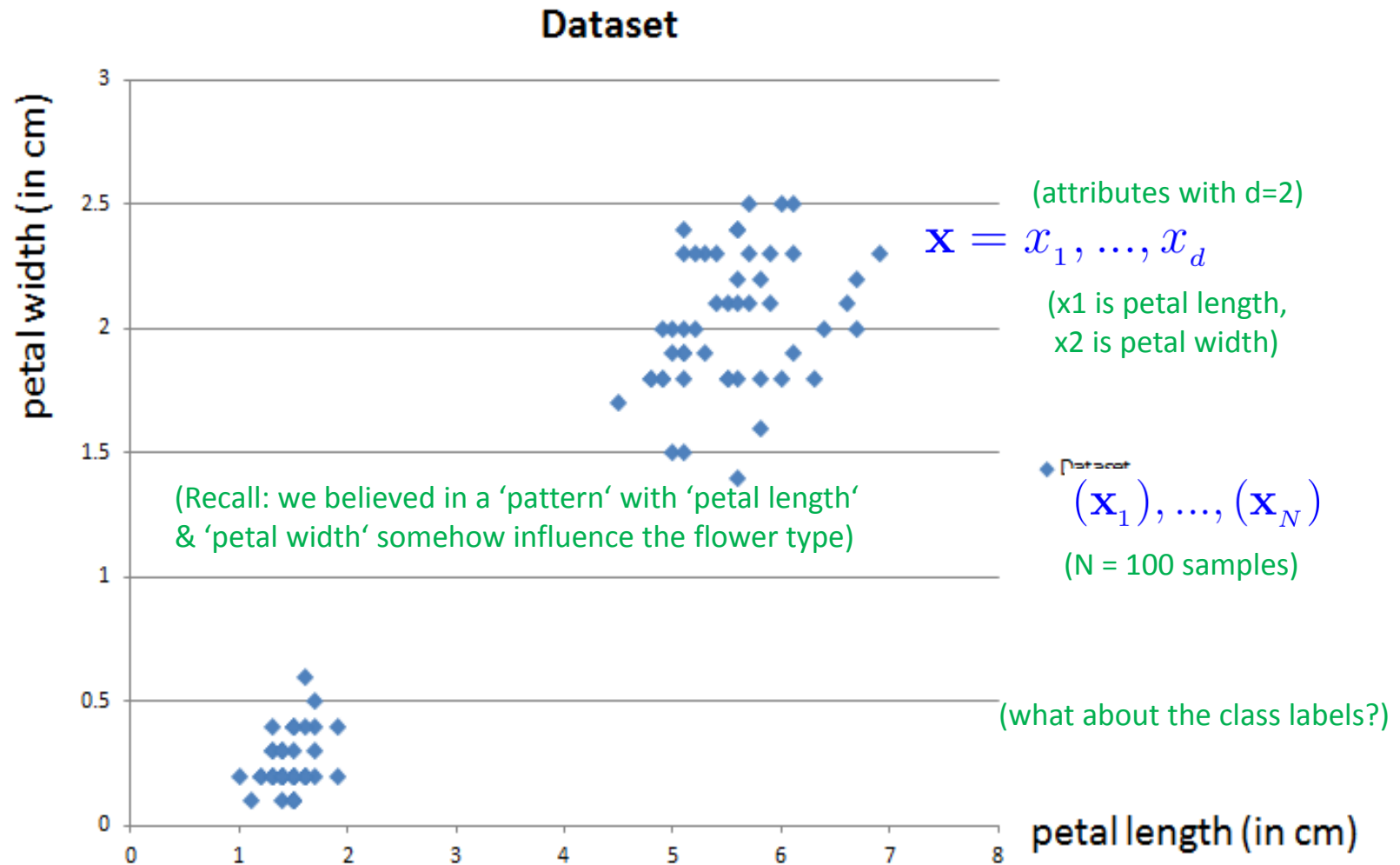
Basic metadata

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

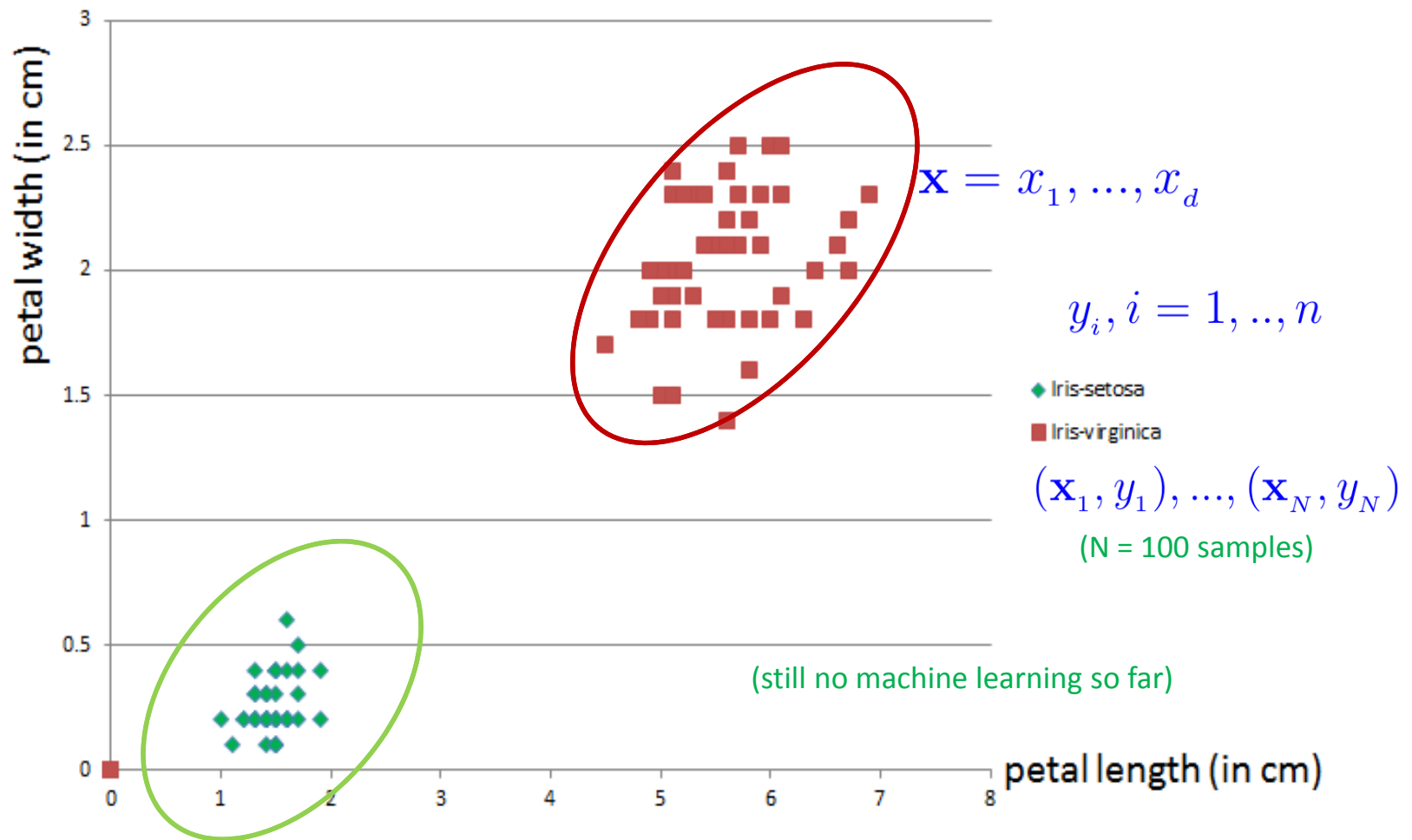
[5] Iris Dataset



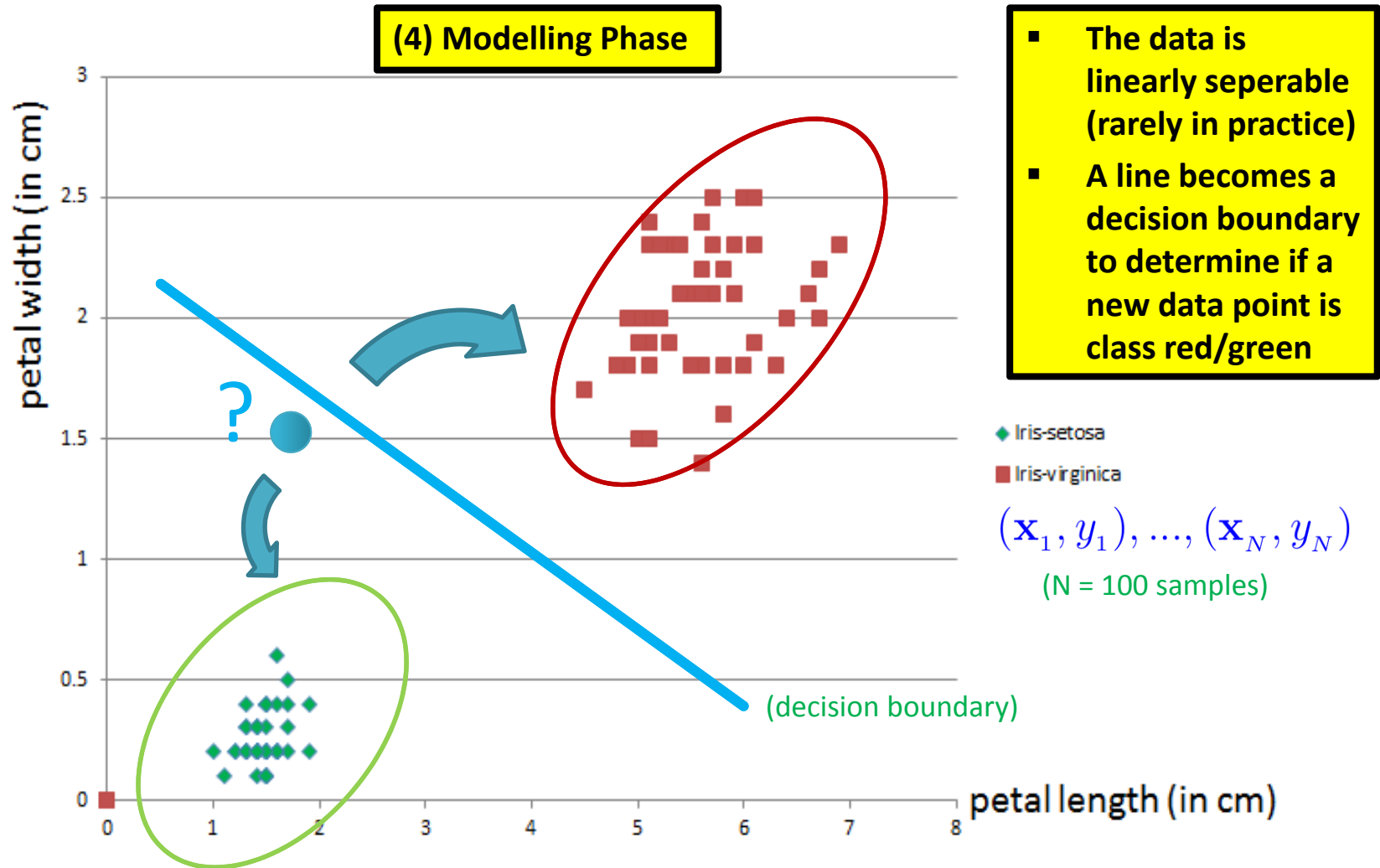
Check Preparation Phase: Plotting the Data



Check Preparation Phase: Class Labels

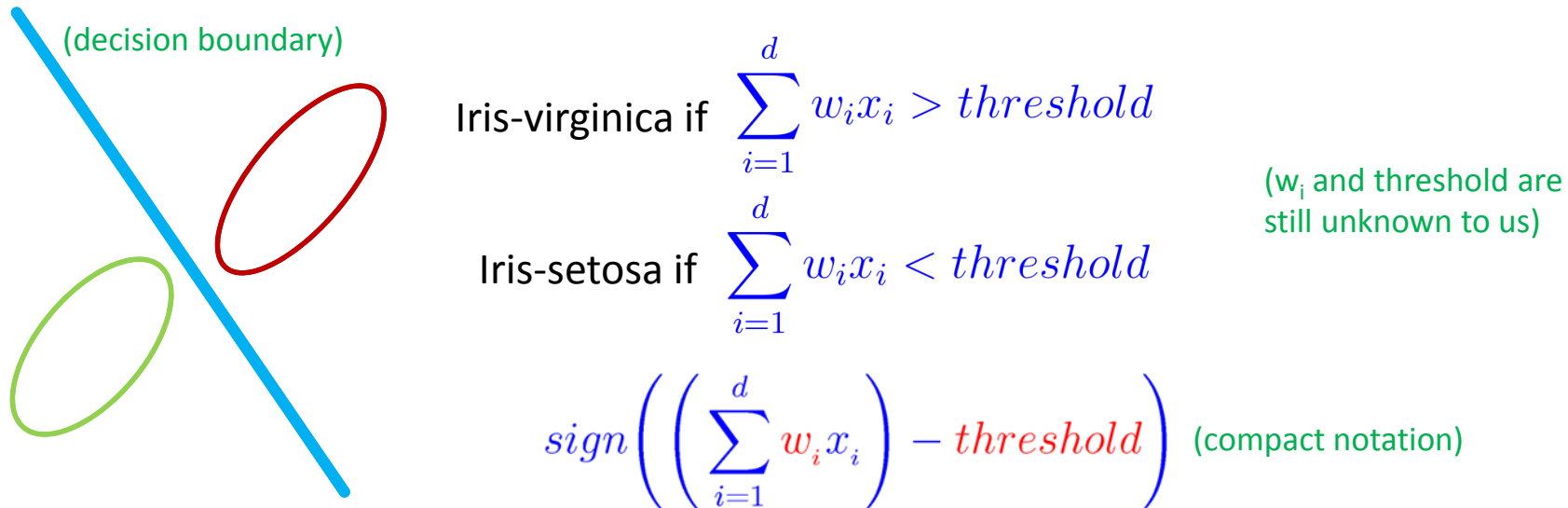


Linearly Seperable Data & Linear Decision Boundary

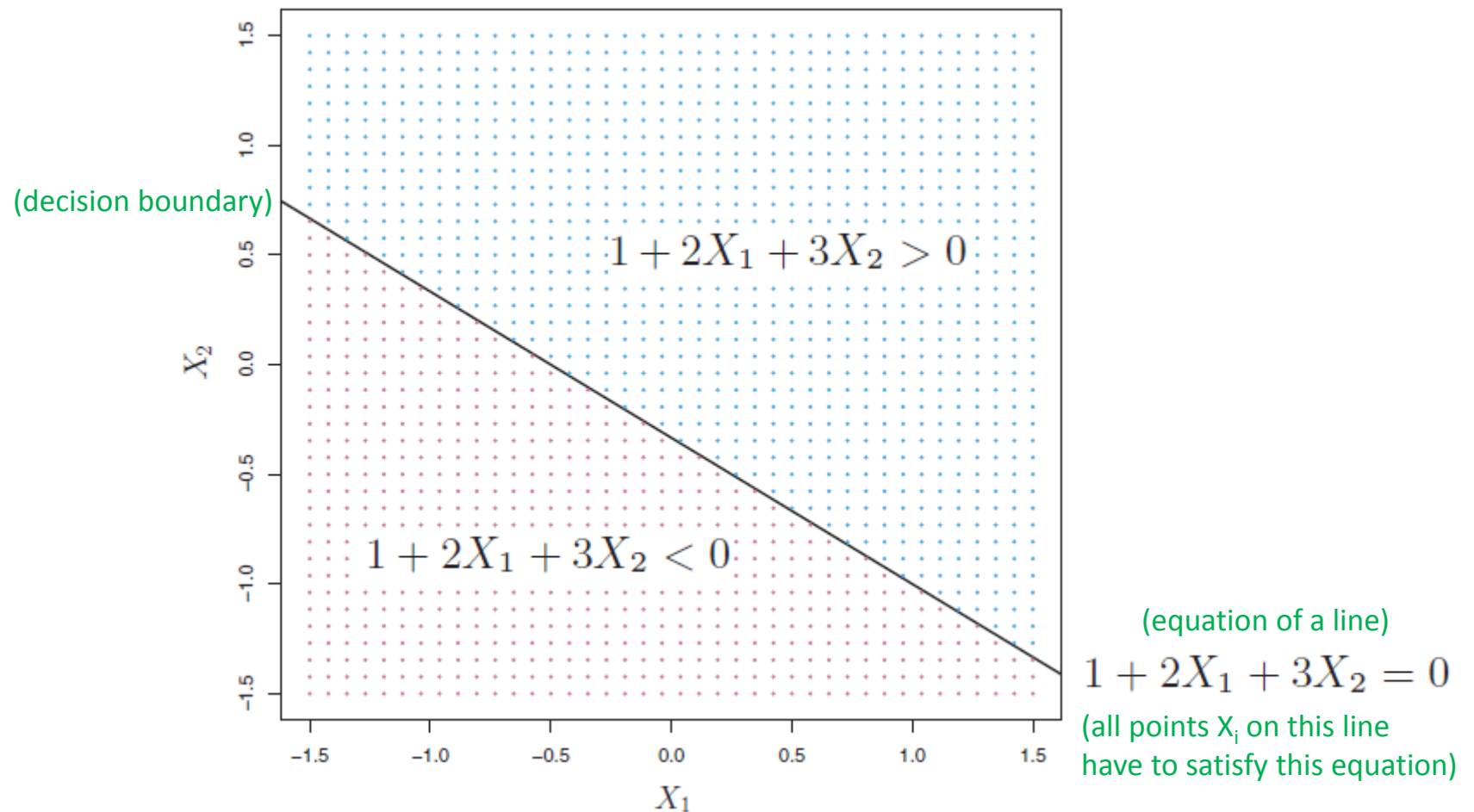


Separating Line & Mathematical Notation

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



Separating Line & 'Decision Space' Example



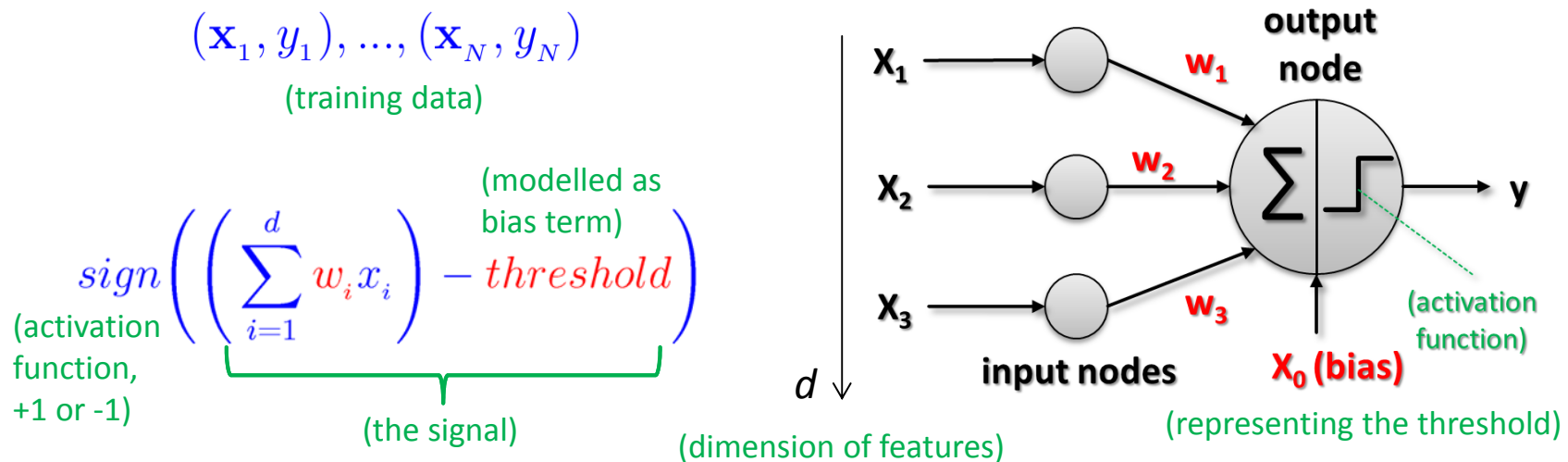
modified from [6] An Introduction to Statistical Learning

A Simple Linear Learning Model – The Perceptron

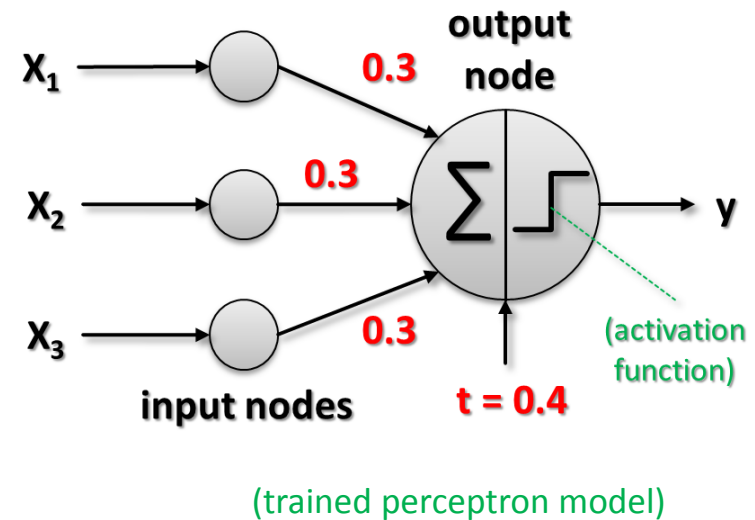
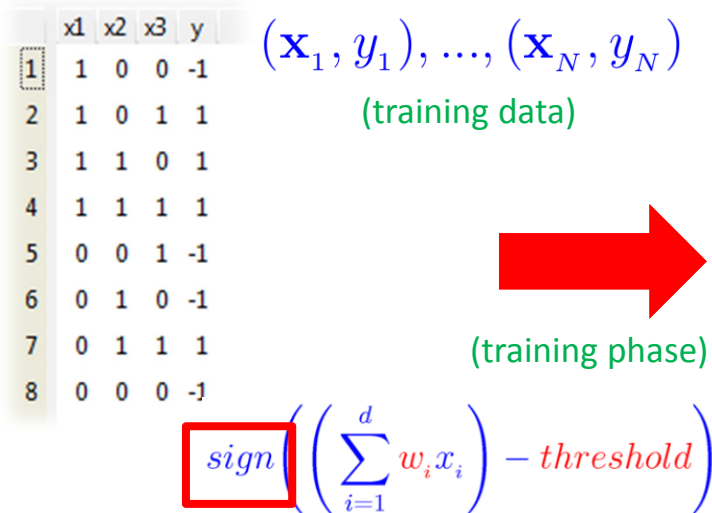
- Human analogy in learning

[7] F. Rosenblatt, 1957

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



Perceptron – Example of a Boolean Function



■ Output node interpretation

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

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

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

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

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

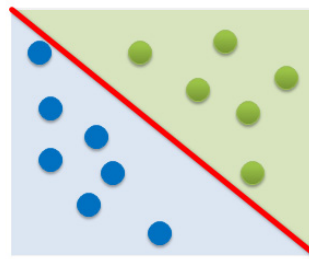
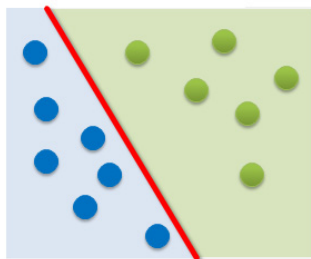
Summary Perceptron & Hypothesis Set $h(\mathbf{x})$

- When: Solving a **linear classification** problem [7] F. Rosenblatt, 1957
 - Goal: learn a simple value (+1/-1) above/below a certain threshold
 - Class label renamed: **Iris-setosa** = -1 and **Iris-virginica** = +1
- Input: $\mathbf{X} = x_1, \dots, x_d$ (attributes in one dataset)
- Linear formula (take attributes and give them different weights – think of ‘impact of the attribute’)
 - All learned formulas are **different hypothesis for the given problem**

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

(parameters that define one hypothesis vs. another)

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



(red parameters correspond to the red line in graphics)

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

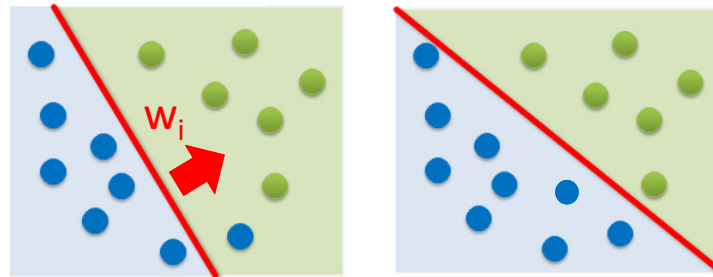
Perceptron Learning Algorithm – Understanding Vector W

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

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

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

(points on the decision boundary satisfy this equation)



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

(vector notation, using T = transpose)

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

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

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

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

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

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

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

(equivalent dotproduct notation)

[8] Rosenblatt, 1958

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

Understanding the Dot Product – Example & Interpretation

- ‘Dot product’

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

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

(our example)

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

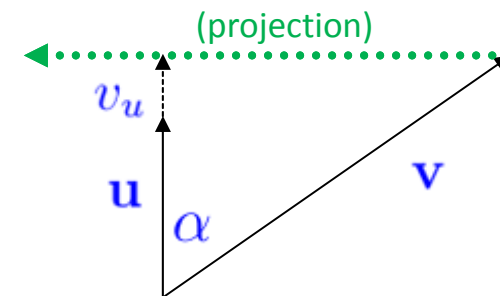
- ‘Projection capabilities of Dot product’ (simplified)

- Orthogonal projection of vector \mathbf{v} in the direction of vector \mathbf{u}

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

- Normalize using length of vector

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



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

Perceptron Learning Algorithm – Learning Step

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

(one point at a time is picked)

- Pick one misclassified training point where:

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

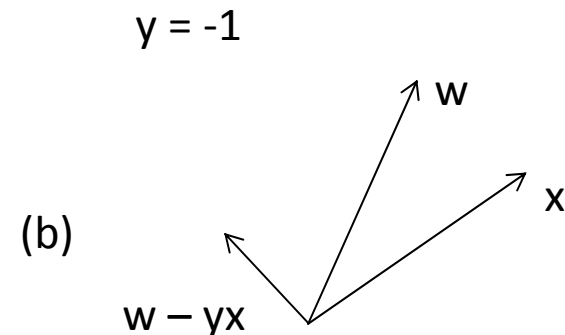
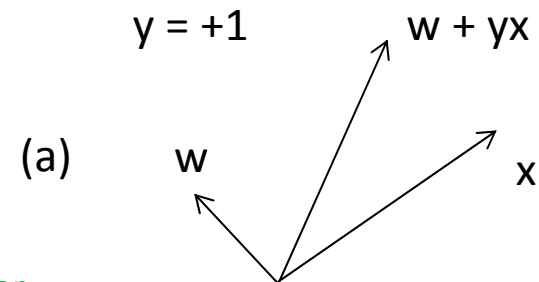
- Update the weight vector:

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

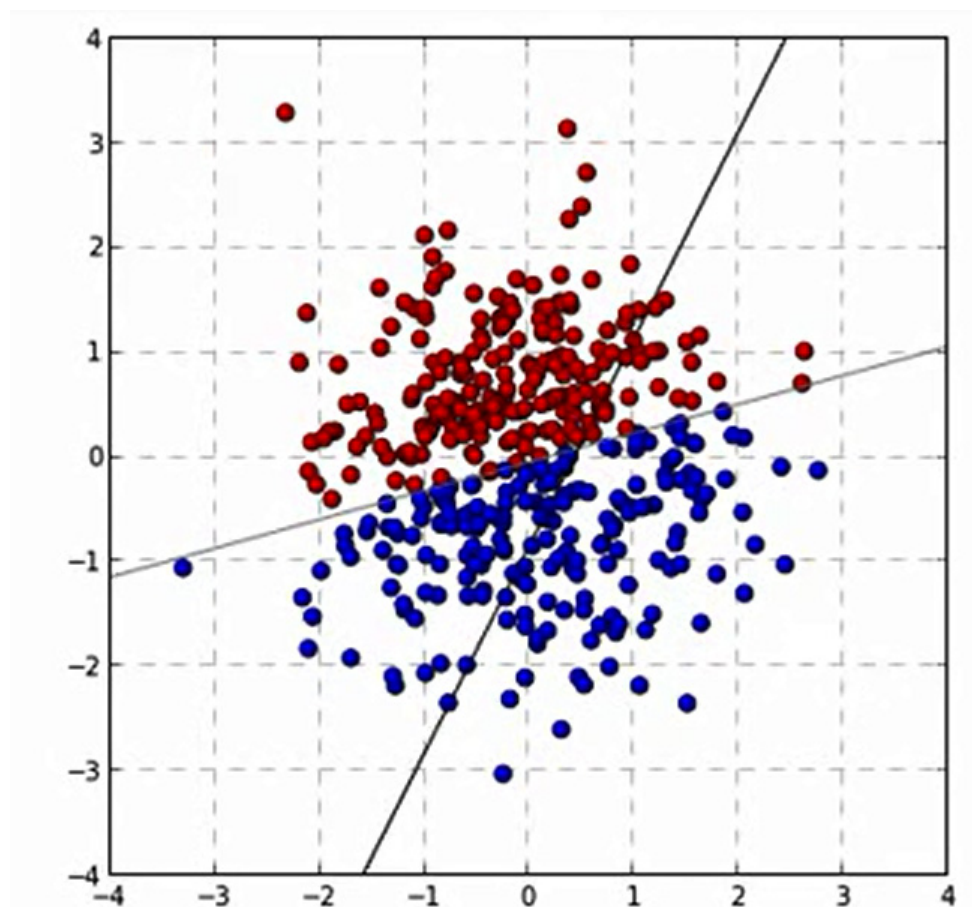
(y_n is either +1 or -1)

- Terminates when there are no misclassified points

(converges only with linearly separable data)



[Video] Perceptron Learning Algorithm



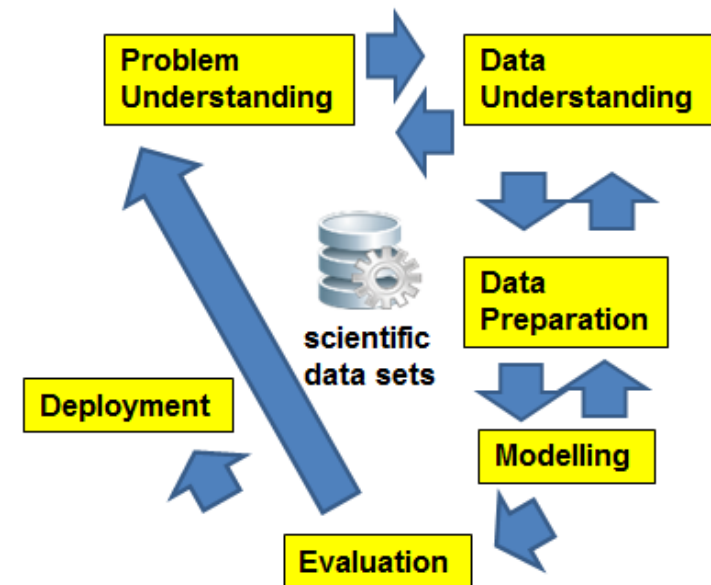
[9] PLA Video

Systematic Process to Support Learning From Data

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

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

(learning takes place)



[10] 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 & Data Mining Tasks in Applications

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

[11] Introduction to Data Mining

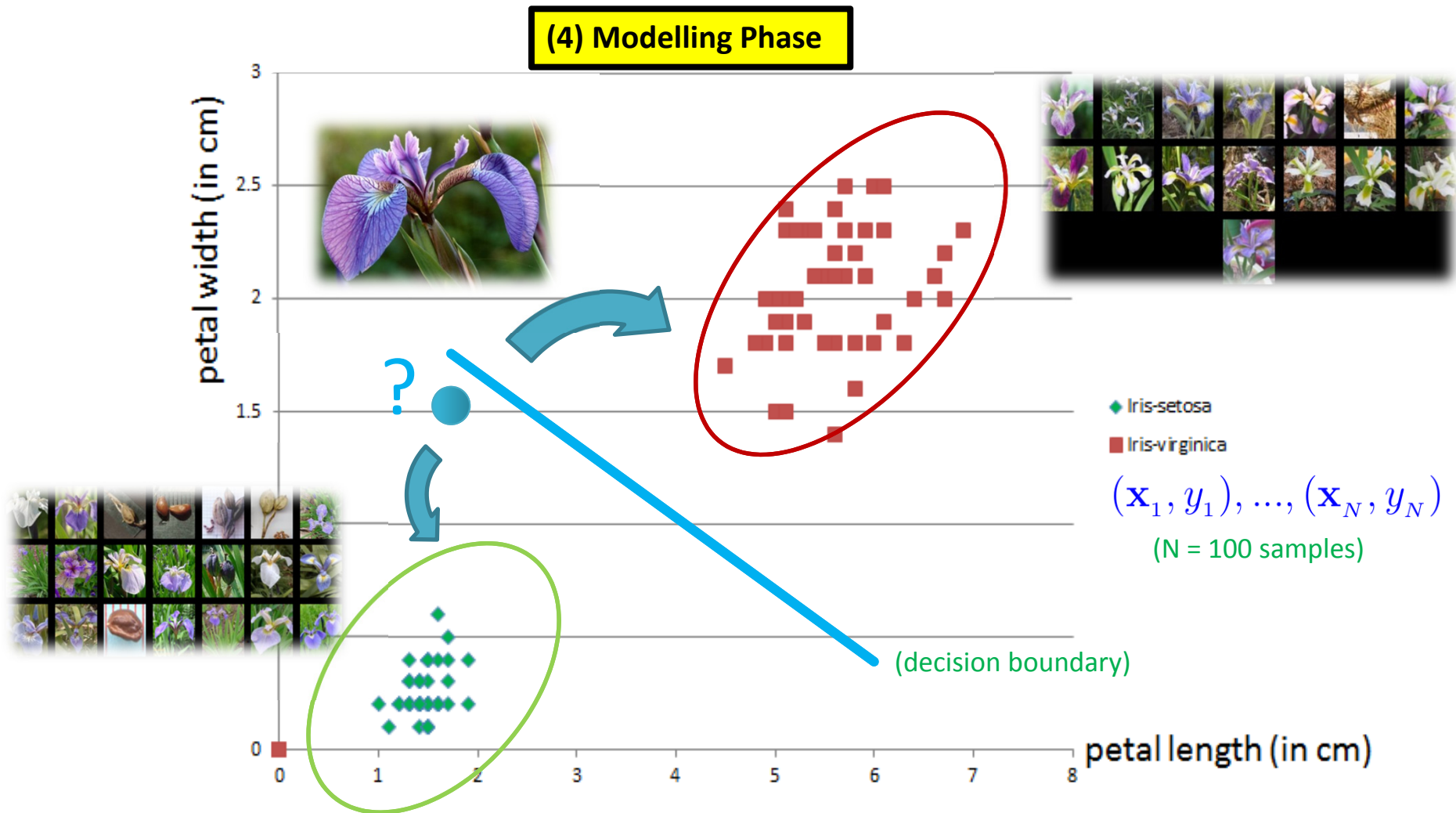
- Predictive Tasks

- Predicts the value of an attribute based on values of other attributes
- Target/dependent variable: attribute to be predicted
- Explanatory/independent variables: attributed used for making predictions
- E.g. predicting the species of a flower based on characteristics of a flower

- Descriptive Tasks

- Derive patterns that summarize the underlying relationships in the data
- Patterns here can refer to correlations, trends, trajectories, anomalies
- Often exploratory in nature and frequently require postprocessing
- E.g. credit card fraud detection with unusual transactions for owners

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



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

Summary Terminologies & Different Dataset Elements

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

(4) Modelling Phase

(5) Evaluation Phase

Model Evaluation – Training and Testing Phases

- Different Phases in Learning

- Training phase is a hypothesis search
- Testing phase checks if we are on right track (once the hypothesis clear)

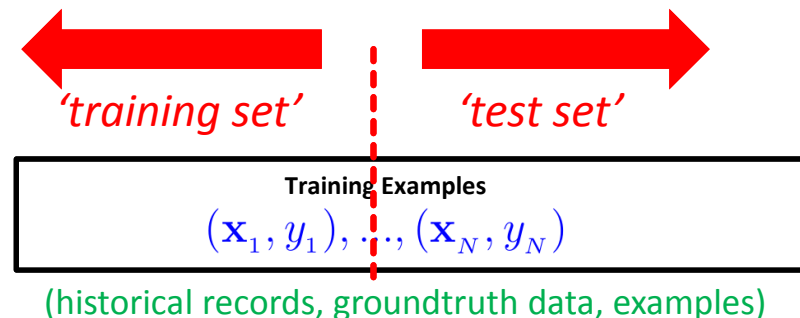
(4) Modelling Phase

(5) Evaluation Phase

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

- Work on ‘training examples’

- Create two disjoint datasets
- One used for training only (aka training set)
- Another used for testing only (aka test set)
- Exact separation is rule of thumb per use case (e.g. 10 % training, 90% test)
- Practice: If you get a dataset take immediately test data away (‘throw it into the corner and forget about it during modelling’)
- Reasoning: Once we learned from training data it has an ‘optimistic bias’



Model Evaluation – Testing Phase & Confusion Matrix

- Model is fixed
 - Model is just used with the testset
 - Parameter w_i are set and we have a linear decision function
- Evaluation of model performance
 - Counts of test records that are incorrectly predicted
 - Counts of test records that are correctly predicted
 - E.g. create confusion matrix for a two class problem

(5) Evaluation Phase

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

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

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

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

Model Evaluation – Testing Phase & Performance Metrics

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

(5) Evaluation Phase

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

- Accuracy (usually in %)

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

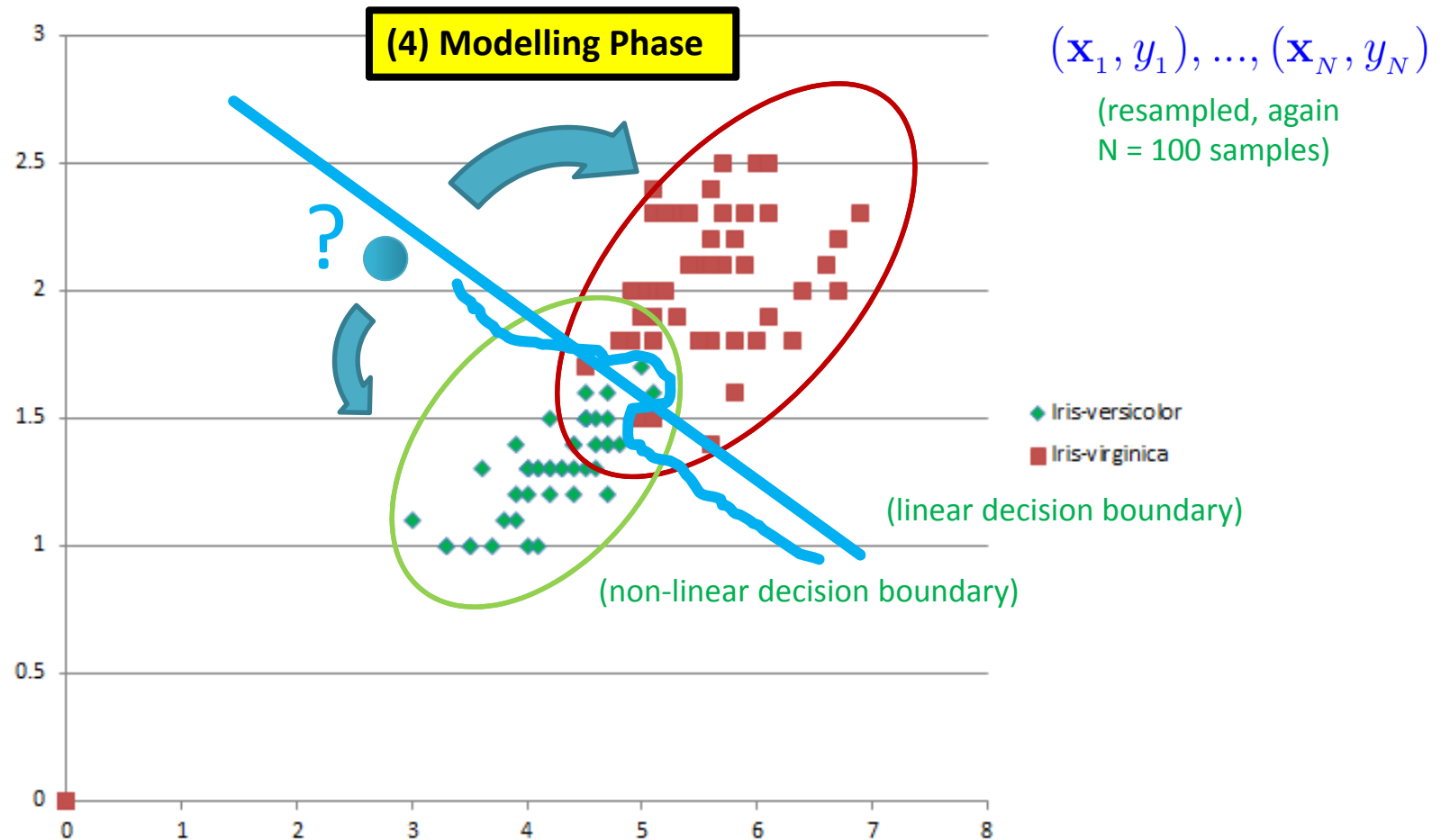
- Error rate

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

- If model evaluation is satisfactory:

(6) Deployment Phase

Non-linearly Seperable Data in Practice – Which model?



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

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

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)

➤ **Appendix B provides an introduction to statistical learning theory & feasibility of learning**

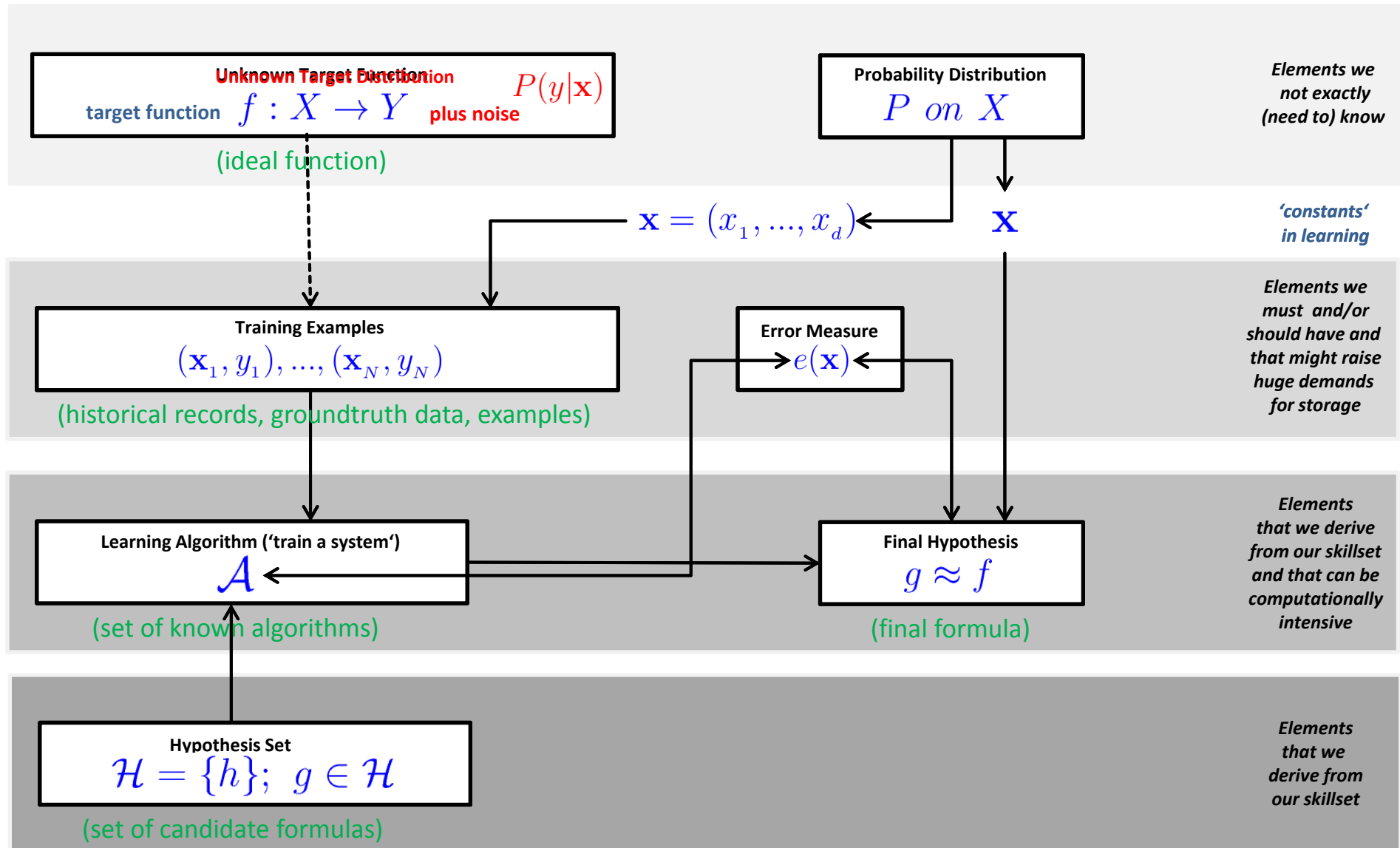
Learning Approaches – Supervised Learning

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

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

[6] An Introduction to Statistical Learning

Supervised Learning – Overview & Summary



Different Models – Understanding the Hypothesis Set

$$\text{Hypothesis Set}$$

$$\mathcal{H} = \{h\}; g \in \mathcal{H}$$

$$\mathcal{H} = \{h_1, \dots, h_m\};$$

(all candidate functions
derived from models
and their parameters)

- Already a change in model parameters of h_1, \dots, h_m means a completely different model

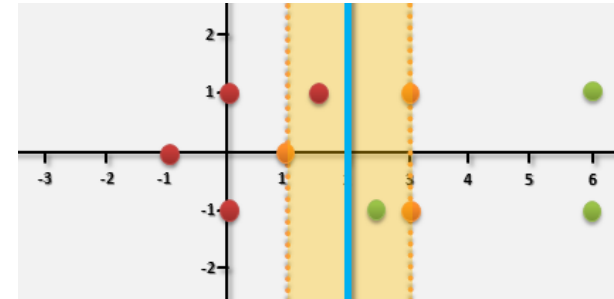
‘select one function’
that best approximates

$$\text{Final Hypothesis}$$

$$g \approx f$$

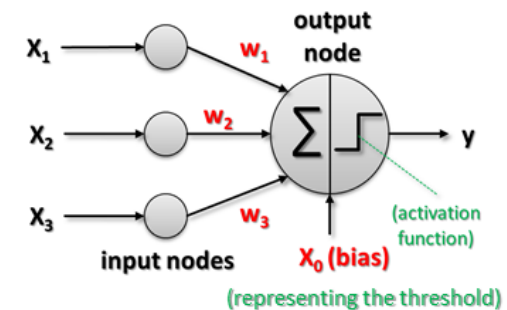


h_1



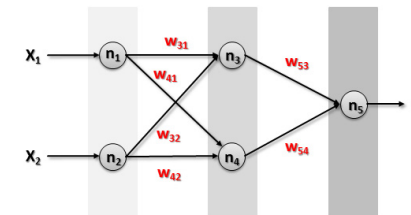
(e.g. support vector machine model)

h_2



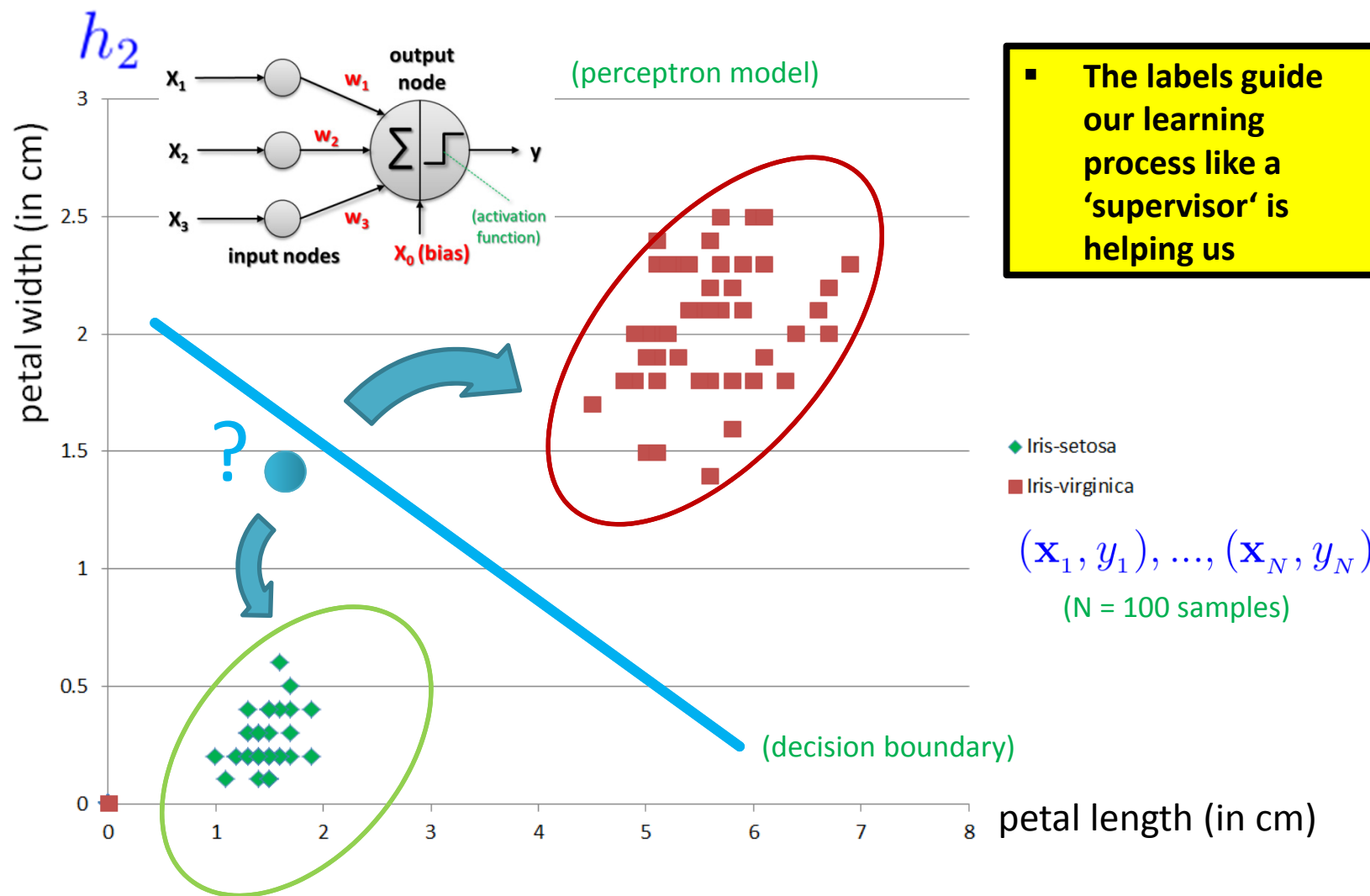
(e.g. linear perceptron model)

h_m

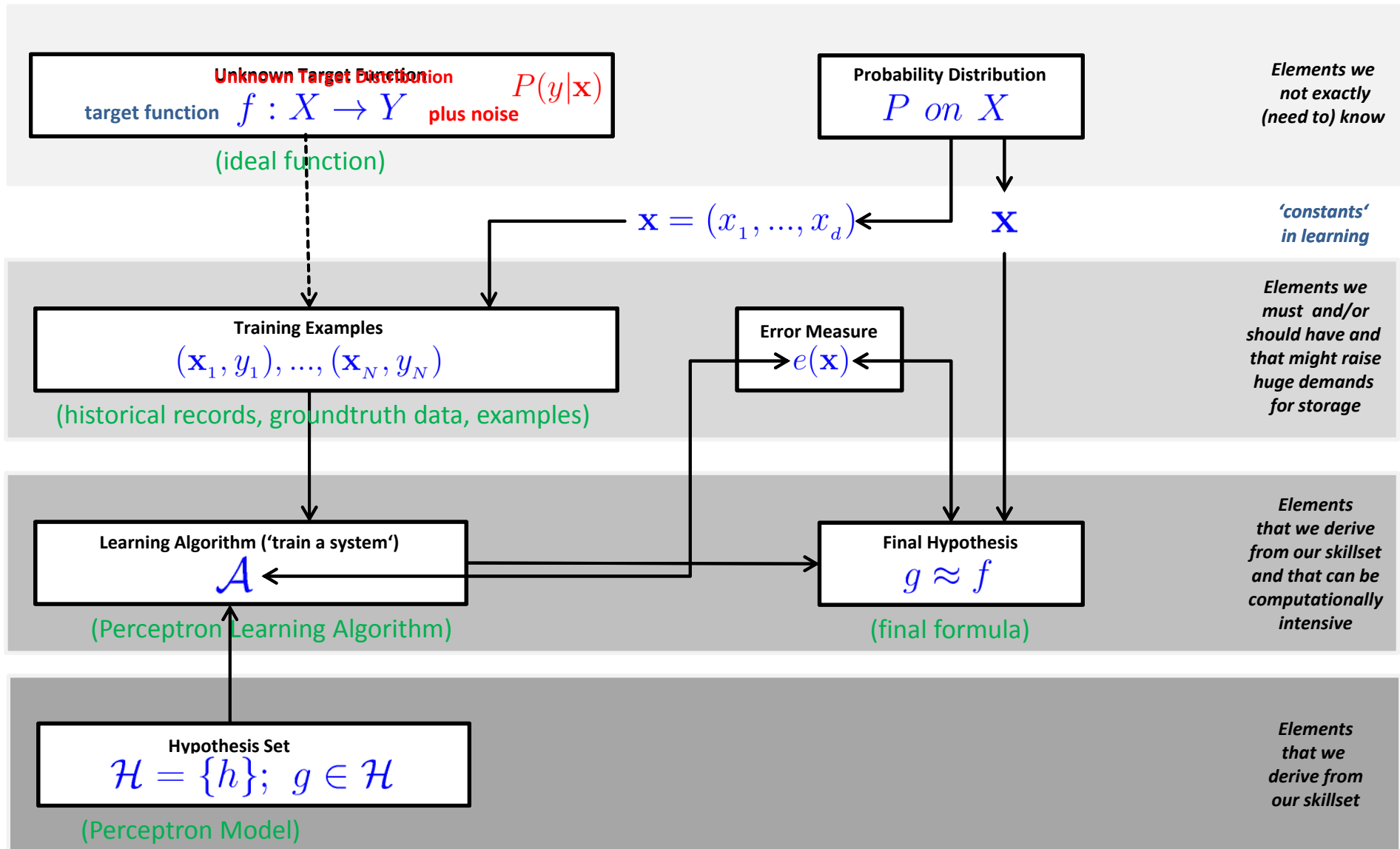


(e.g. artificial neural network model)

Learning Approaches – Supervised Learning Example



Supervised Learning – Linear Perceptron Example



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

■ Scalability

- Gigabytes, Terabytes, and Petabytes datasets that fit not into memory
- E.g. algorithms become necessary with out-of-core/CPU strategies

■ High Dimensionality

- Datasets with hundreds or thousand attributes become available
- E.g. bioinformatics with gene expression data with thousand of features

■ Heterogenous and Complex Data

- More complex data objects emerge and unstructured data sets
- E.g. Earth observation time-series data across the globe

■ Data Ownership and Distribution

- Distributed datasets are common (e.g. security and transfer challenges)

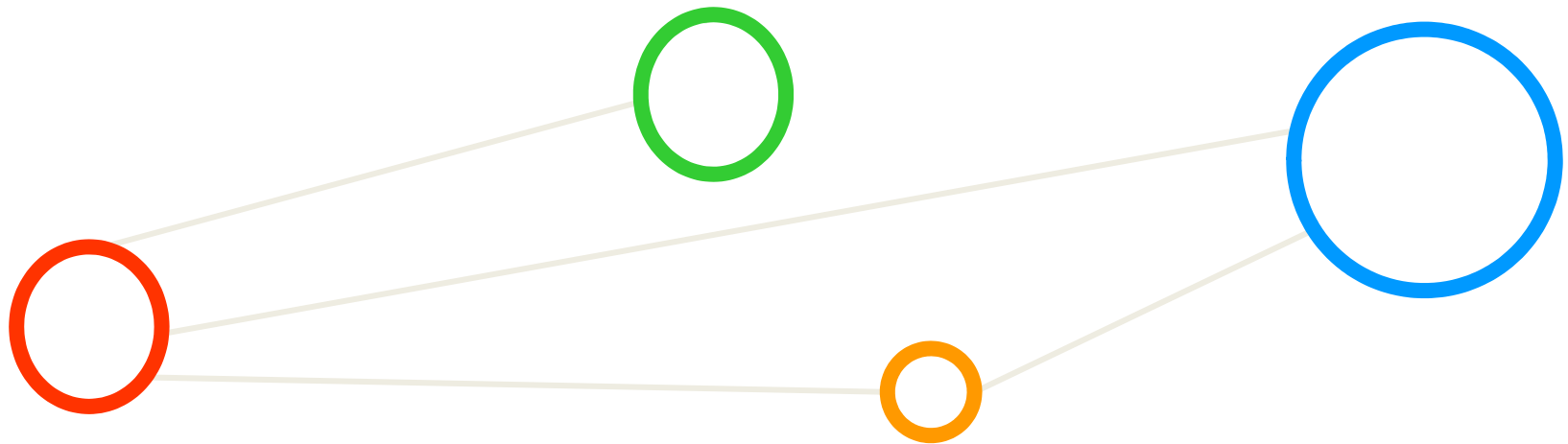
- **Key challenges faced when doing traditional data analysis and machine learning are scalability, high dimensionality of datasets, heterogenous and complex data, data ownership & distribution**
- **Combat 'overfitting' is the key challenge in machine learning using validation & regularization**

[Video] Remote Sensing



[19] YouTube Video, "What is Remote Sensing?"

Application Examples



Exercises – Explore the Rome Dataset

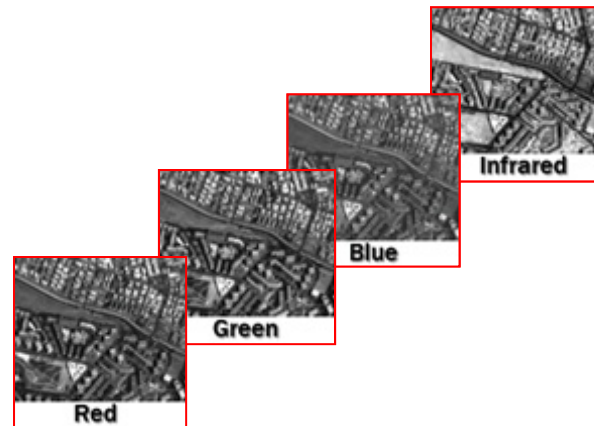


Example Rome Dataset

- Geographical location: [Image of Rome](#), Italy
 - Multispectral data obtained by [Quickbird satellite sensor](#)
- High-resolution (0.6m) panchromatic image



Low-resolution (2.4m)
multispectral images



Classes

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

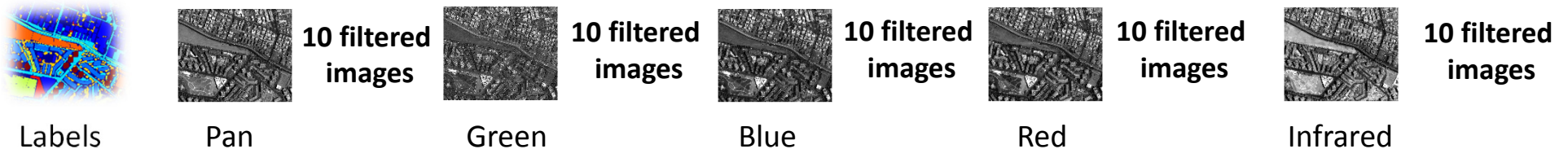
(Reasoning for picking SVM: Good classification accuracies on high dimensional datasets, even with a small ,rare' number of training samples)

[16] Rome Image dataset



Understanding the Rome Dataset & Feature Engineering

Class + Input + Features + Input + Feature + Input + Features + Input + Features + Input + Features



- Each pixel vector is stored as a line with the libSVM format
- E.g.,

```
2 1:0.364706 2:0.360784 3:0.356863 4:0.356863 5:0.349206 6:0.306878 7:0.419355
8:0.453608 9:0.368421 10:1 11:1 12:0.423529 13:0.403922 14:0.403922 15:0.369919
16:0.320833 17:0.302564 18:0.481481 19:0.483516 20:0.32 21:0.625 22:0.833333
23:0.376471 24:0.376471 25:0.372549 26:0.358566 27:0.318367 28:0.243386 29:0.455446
30:0.4 31:0.319149 32:0.368421 33:0.4 34:0.556863 35:0.54902 36:0.436 37:0.322176
38:0.215962 39:0.151079 40:0.257576 41:0.267857 42:0.266667 43:0.277778 44:0.4375
45:0.360784 46:0.360784 47:0.368627 48:0.368627 49:0.363636 50:0.353846 51:0.347826
52:0.335294 53:0.333333 54:0.978723 55:1
```

[17] G. Cavallaro & M. Riedel et al., 2014

Inspecting and Understanding the Rome Dataset

- Data is publicly available in EUDAT B2SHARE tool



[18] Rome Image dataset

Rome data set OK

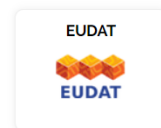
by [Unknown]

Dec 22, 2016

Last updated at Jan 11, 2018

Abstract: Attribute area

PID: [11304/4615928c-e1a5-11e3-8cd7-14feb57d12b9](https://hdl.handle.net/11304/4615928c-e1a5-11e3-8cd7-14feb57d12b9) [Copy](#)



Files

Name	Size
> sdap_area_all_test.el	419.97MB
> sdap_area_all_training.el	46.65MB
> sdap_area_panch_test.el	114.76MB
> sdap_area_panch_training.el	12.75MB

Basic metadata

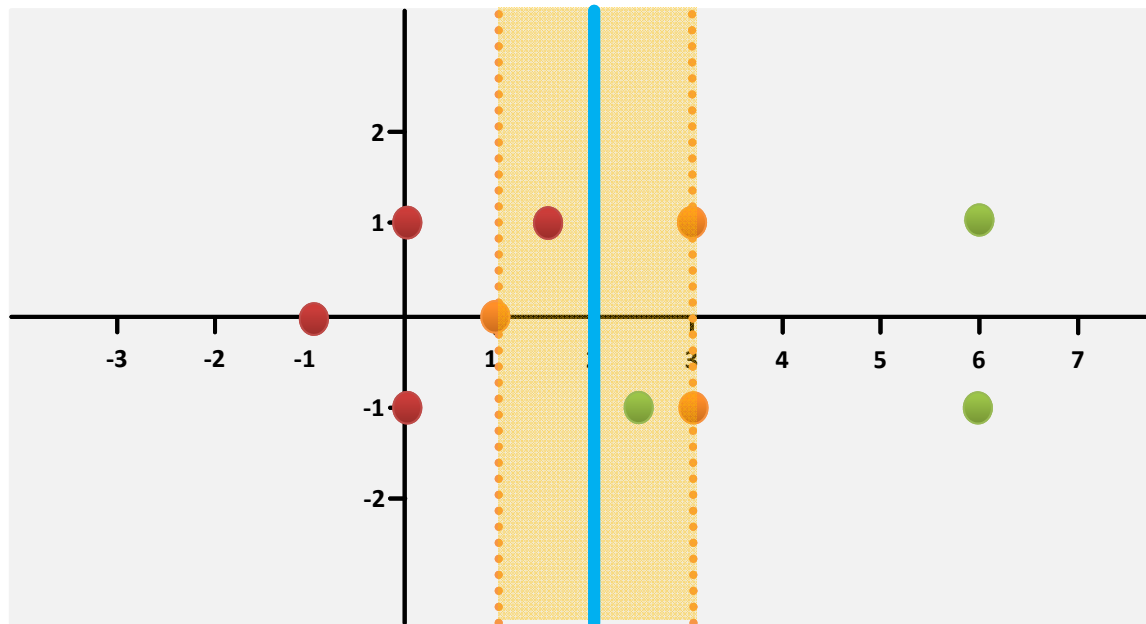
Open Access	True ✓
License	
Contact Email	
Publication Date	2014-05-22
Contributors	
Resource Type	Category Other
Alternate identifiers	86
Type	B2SHARE_V1_ID
	http://hdl.handle.net/11304/a2892f1c-7e13-42e9-be24-88aegda847b1
Type	ePIC_PID
	http://b2share.eudat.eu
	en

```
[train001@jrl12 rome]$ pwd
/homea/hpclab/train001/data/rome
[train001@jrl12 rome]$ ls -al
total 580256
drwxr-xr-x 2 train001 hpclab      512 Jan 14 21:52 .
drwxr-xr-x 6 train001 hpclab      512 Jan 14 21:47 ..
-rw-r--r-- 1 train001 hpclab 419974873 Dec 22 2016 sdap_area_all_test.el
-rw-r--r-- 1 train001 hpclab 46652874 Dec 22 2016 sdap_area_all_training.el
-rw-r--r-- 1 train001 hpclab 114763982 Dec 22 2016 sdap_area_panch_test.el
-rw-r--r-- 1 train001 hpclab 12745692 Dec 22 2016 sdap_area_panch_training.el
```

(persistent handle link for publication into papers)

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

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



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

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

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

➤ Appendix C shows how Support Vector Machines (SVMs) are mathematically established

Term Support Vector Machines Refined

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

[6] *An Introduction to Statistical Learning*

- Term detailed refinement into **'three separate techniques'**
 - Practice: applications mostly use the SVMs with kernel methods
- **'Maximal margin classifier'**
 - A simple and intuitive classifier with a 'best' linear class boundary
 - Requires that data is **'linearly separable'**
- **'Support Vector Classifier'**
 - Extension to the maximal margin classifier for non-linearly separable data
 - Applied to a broader range of cases, idea of **'allowing some error'**
- **'Support Vector Machines' → Using Non-Linear Kernel Methods**
 - Extension of the support vector classifier
 - Enables non-linear class boundaries & via **kernels**;

Exercises – Submit piSVM & Rome (linear)



JURECA System – SSH Login

- Use your account train004 - train050
- Windows: use putty / MobaXterm
- UNIX: `ssh trainXYZ@jureca.fz-juelich.de`
- Example

```
adminuser@linux-8djg:~> ssh train001@jureca.fz-juelich.de
Warning: the ECDSA host key for 'jureca.fz-juelich.de' differs from the key for the IP address '134.94.33.9'
Offending key for IP in /home/adminuser/.ssh/known_hosts:12
Matching host key in /home/adminuser/.ssh/known_hosts:19
Are you sure you want to continue connecting (yes/no)? yes
]Last login: Mon Aug 21 14:29:03 2017 from zam2036.zam.kfa-juelich.de
*****
*                               Welcome to JURECA                               *
*                                                                           *
* Information about the system, latest changes, user documentation and FAQs: *
*                               http://www.fz-juelich.de/ias/jsc/jureca         *
*****
*                               ### Known Issues ###                        *
*                                                                           *
* An up-to-date list of known issues on the system is maintained at         *
*                               http://www.fz-juelich.de/ias/jsc/jureca-known-issues *
* Open issues:                                                             *
*   - Intel compiler error with std::valarray and                         *
*     optimized headers, added 2016-03-20                                   *
```

➤ Remember to use your own trainXYZ account in order to login to the JURECA system

Rome Remote Sensing Dataset

- Data is already available in the tutorial directory

Rome data set OK

22 May 2014
<http://bzshare.eudat.eu>

Abstract: Attribute area

The record appears in these collections:
Generic

Name	Date	Size	
sdap_area_panch_training.el	22 May 2014	12.7 MB	Download
sdap_area_all_training.el	22 May 2014	46.7 MB	Download
sdap_area_panch_test.el	22 May 2014	114.8 MB	Download
sdap_area_all_test.el	22 May 2014	420.0 MB	Download

Export
Export as [BibTeX](#), [MARC](#), [MARCXML](#), [DC](#), [EndNote](#), [NLM](#), [RefWorks](#)

Metadata
PID: <http://hdl.handle.net/11304/4615928c-e1a5-11e3-8cd7-14feb57d12b9>
Publication: <http://bzshare.eudat.eu>
Publication Date: 2014-05-22

(persistent handle link for publication into papers)



[18] Rome Image dataset

```
[train001@jrl12 rome]$ pwd
/homea/hpclab/train001/data/rome
[train001@jrl12 rome]$ ls -al
total 580256
drwxr-xr-x 2 train001 hpclab      512 Jan 14 21:52 .
drwxr-xr-x 6 train001 hpclab      512 Jan 14 21:47 ..
-rw-r--r-- 1 train001 hpclab 419974873 Dec 22 2016 sdap_area_all_test.el
-rw-r--r-- 1 train001 hpclab 46652874 Dec 22 2016 sdap_area_all_training.el
-rw-r--r-- 1 train001 hpclab 114763982 Dec 22 2016 sdap_area_panch_test.el
-rw-r--r-- 1 train001 hpclab 12745692 Dec 22 2016 sdap_area_panch_training.el
```

HPC Environment – Modules Revisited

- **Module** environment tool
 - Avoids to manually setup environment information for every application
 - Simplifies shell initialization and lets users easily modify their environment
- **Module avail**
 - Lists all available modules on the HPC system (e.g. compilers, MPI, etc.)
- **Module spider**
 - Find modules in the installed set of modules and more information
- **Module load → needed before piSVM run**
 - Loads particular modules into the current work environment, E.g.:
 - Module load Intel
 - Module load IntelMPI

Parallel & Scalable PiSVM – Parameters

```
[train001@j3l02 pisvm-1.2.1]$ ./pisvm-train
Usage: svm-train [options] training_set_file [model_file]
options:
```

```
-s svm_type : set type of SVM (default 0)
    0 -- C-SVC
    1 -- nu-SVC
    2 -- one-class SVM
    3 -- epsilon-SVR
    4 -- nu-SVR
-t kernel_type : set type of kernel function (default 2)
    0 -- linear: u'*v
```

```
    1 -- polynomial: (gamma*u'*v + coef0)^degree
    2 -- radial basis function: exp(-gamma*|u-v|^2)
    3 -- sigmoid: tanh(gamma*u'*v + coef0)
-d degree : set degree in kernel function (default 3)
-g gamma : set gamma in kernel function (default 1/k)
-r coef0 : set coef0 in kernel function (default 0)
-c cost : set the parameter C of C-SVC, epsilon-SVR, and nu-SVR (default 1)
-n nu : set the parameter nu of nu-SVC, one-class SVM, and nu-SVR (default 0.5)
-p epsilon : set the epsilon in loss function of epsilon-SVR (default 0.1)
-m cachesize : set cache memory size in MB (default 40)
-e epsilon : set tolerance of termination criterion (default 0.001)
-h shrinking: whether to use the shrinking heuristics, 0 or 1 (default 1)
-b probability_estimates: whether to train a SVC or SVR model for probability estimates, 0 or 1 (default 0)
-wi weight: set the parameter C of class i to weight*C, for C-SVC (default 1)
-v n: n-fold cross validation mode
-o n: max. size of working set
-q n: max. number of new variables entering working set
flags:
-D: Assume the feature vectors are dense (default: sparse)
```

- **C-SVC:** The cost (C) in this case refers to a soft-margin specifying how much error is allowed and thus represents a regularization parameter that prevents overfitting → more details tomorrow
- **nu-SVC:** nu in this case refers to values between 0 and 1 and thus represents a lower and upper bound on the number of examples that are support vectors and that lie on the wrong side of the hyperplane

Training Rome on JURECA – Job Script (linear)

- Use Rome Dataset with parallel & scalable piSVM tool
 - Parameters are equal to the serial libsvm and some additional parameters for parallelization

```
#!/bin/bash -x
#SBATCH --nodes=2
#SBATCH --ntasks=48
#SBATCH --ntasks-per-node=24
#SBATCH --output=mpi-out.%j
#SBATCH --error=mpi-err.%j
#SBATCH --time=01:00:00
#SBATCH --partition=batch
#SBATCH --mail-user=m.riedel@fz-juelich.de
#SBATCH --mail-type=ALL
#SBATCH --job-name=train-rome-lin-2-48-24

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

### location data
TRAINDATA=/homea/hpclab/train001/data/rome/sdap_area_all_training.el

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

- **Note the tutorial reservation with `--reservation=bigdata-cpu` just valid for today on JURECA**

Testing Rome on JURECA – Job Script (linear)

- Use Rome Dataset with parallel & scalable piSVM tool
 - Parameters are equal to the serial libsvm and some additional parameters for parallelization

```
#SBATCH --ntasks-per-node=24
#SBATCH --output=mpi-out.%j
#SBATCH --error=mpi-err.%j
#SBATCH --time=01:00:00
#SBATCH --partition=batch
#SBATCH --mail-user=m.riedel@fz-juelich.de
#SBATCH --mail-type=ALL
#SBATCH --job-name=pred-rome-2-48-24

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

### location data
TESTDATA=/homea/hpclab/train001/data/rome/sdap_area_all_test.el

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

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

■ Note the tutorial reservation with `--reservation=bigdata-cpu` just valid for today on JURECA

Testing Rome on JURECA – Check Outcome

- The output of the training run is a model file

- Used for input for the testing/predicting phase
- In-sample property → Support vectors of the model

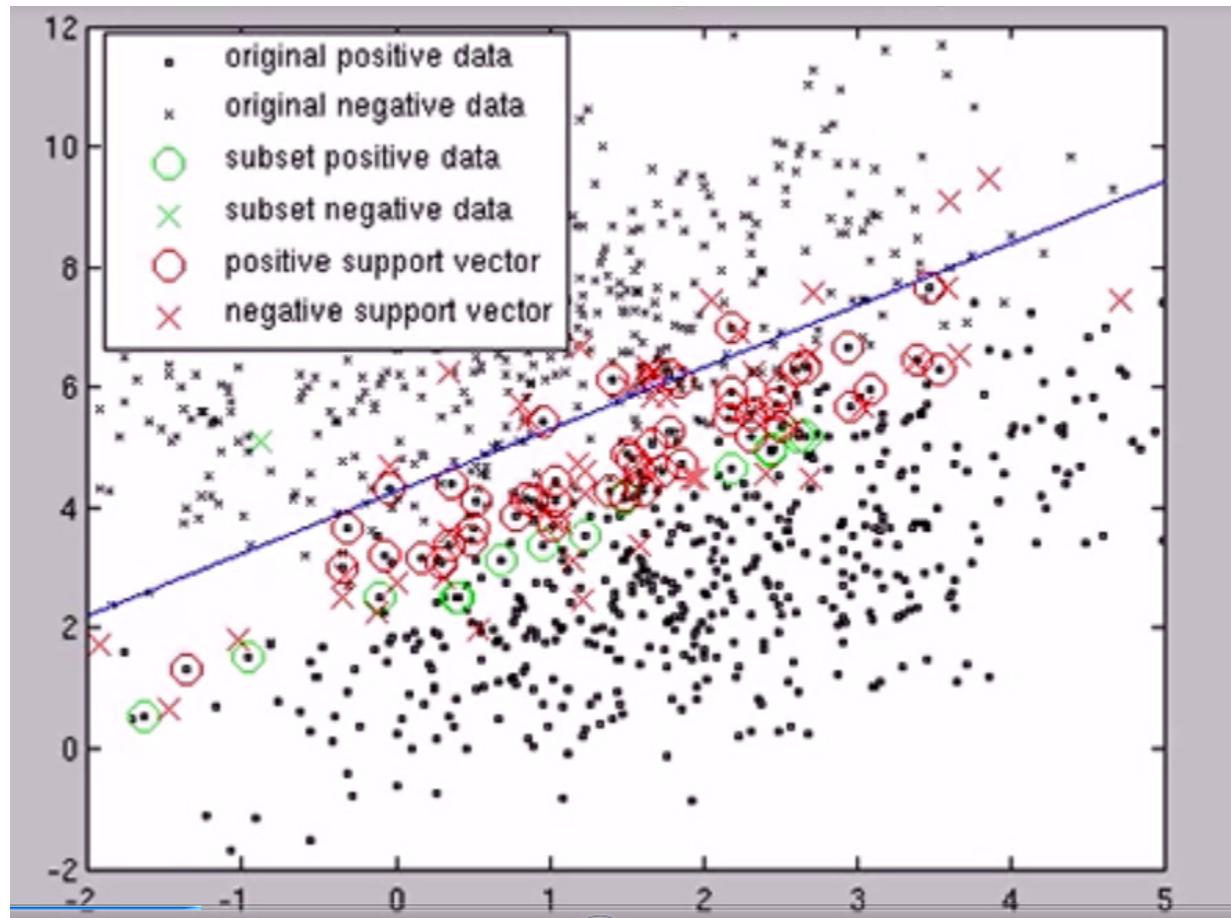
```
### trained model data  
MODELDATA=/homea/hpclab/train001/tools/pisvm-1.2.1/sdap_area_all_training.el.model
```

- The output of the testing/predicting phase is accuracy

- Accuracy measurement of model performance (cf. Lecture 1)
- The job output file consists of a couple of lines:

```
[train001@j3l02 pisvm-1.2.1]$ more mpi-out.15244542  
Accuracy = 91.5994% (639235/697859) (classification)  
Mean squared error = 1.04794 (regression)  
Squared correlation coefficient = 0.835385 (regression)
```

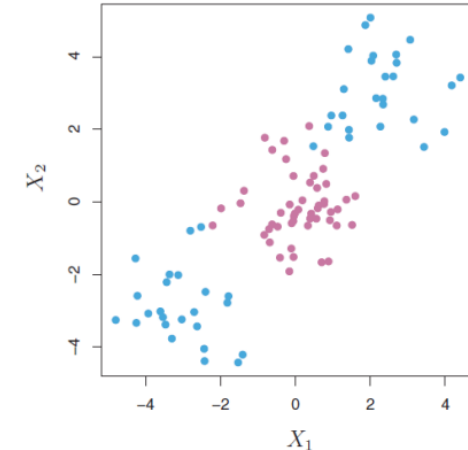
[Video] Training Process of Support Vector Machines



[5] YouTube Video, 'Cascade SVM'

Need for Non-linear Decision Boundaries

- Lessons learned from practice
 - Scientists and engineers are often faced with **non-linear class boundaries**
- Non-linear transformations approach
 - **Enlarge feature space (computationally intensive)**
 - Use **quadratic, cubic, or higher-order polynomial** functions of the predictors
- Example with Support Vector Classifier



(time invest: mapping done by explicitly carrying out the map into the feature space)

X_1, X_2, \dots, X_p (previously used p features)

$X_1, X_1^2, X_2, X_2^2, \dots, X_p, X_p^2$ (new $2p$ features)

(decision boundary is linear in the enlarged feature space)

(decision boundary is non-linear in the original feature space with $q(x) = 0$ where q is a quadratic polynomial)

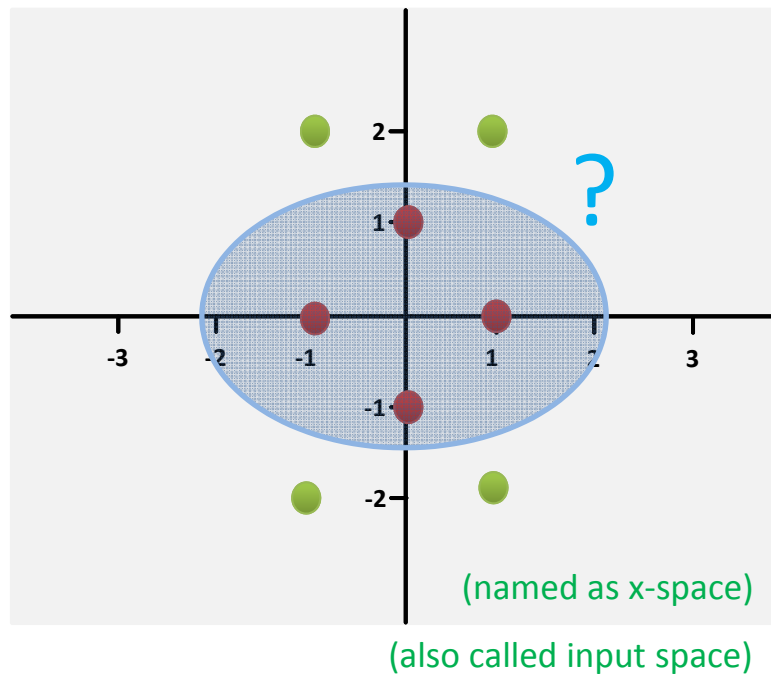
$$\begin{aligned}
 & \underset{\beta_0, \beta_{11}, \beta_{12}, \dots, \beta_{p1}, \beta_{p2}, \epsilon_1, \dots, \epsilon_n}{\text{maximize}} && M \\
 & \text{subject to } y_i \left(\beta_0 + \sum_{j=1}^p \beta_{j1} x_{ij} + \sum_{j=1}^p \beta_{j2} x_{ij}^2 \right) \geq M(1 - \epsilon_i) \\
 & \sum_{i=1}^n \epsilon_i \leq C, \quad \epsilon_i \geq 0, \quad \sum_{j=1}^p \sum_{k=1}^2 \beta_{jk}^2 = 1
 \end{aligned}$$

[6] An Introduction to Statistical Learning

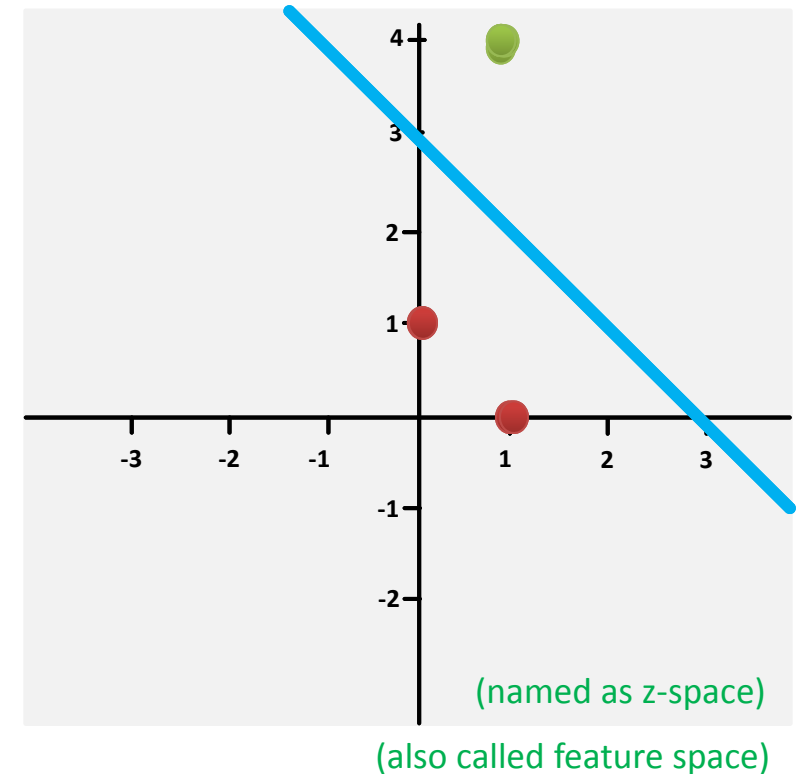
Understanding Non-Linear Transformations (1)

- Example: 'Use measure of distances from the origin/centre'
- Classification
 - (1) new point; (2) transform to z-space; (3) classify it with e.g. perceptron

(still linear models applicable)

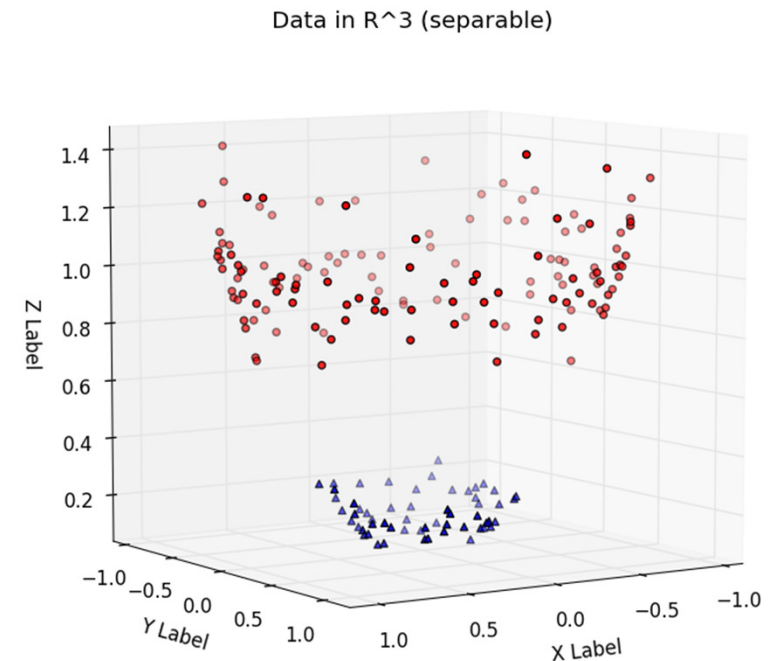
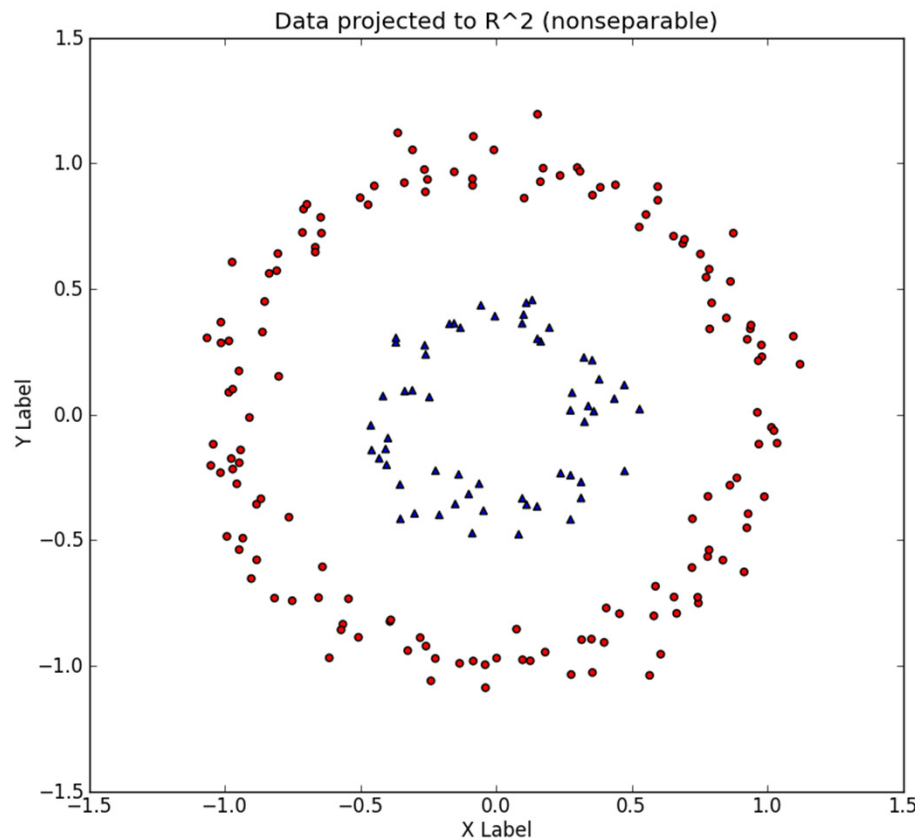


Φ
→
(‘changing constants’)



Understanding Non-Linear Transformations (2)

- Example: From 2 dimensional to 3 dimensional: $[x_1, x_2] = [x_1, x_2, x_1^2 + x_2^2]$
 - Much higher dimensional can cause memory and computing problems

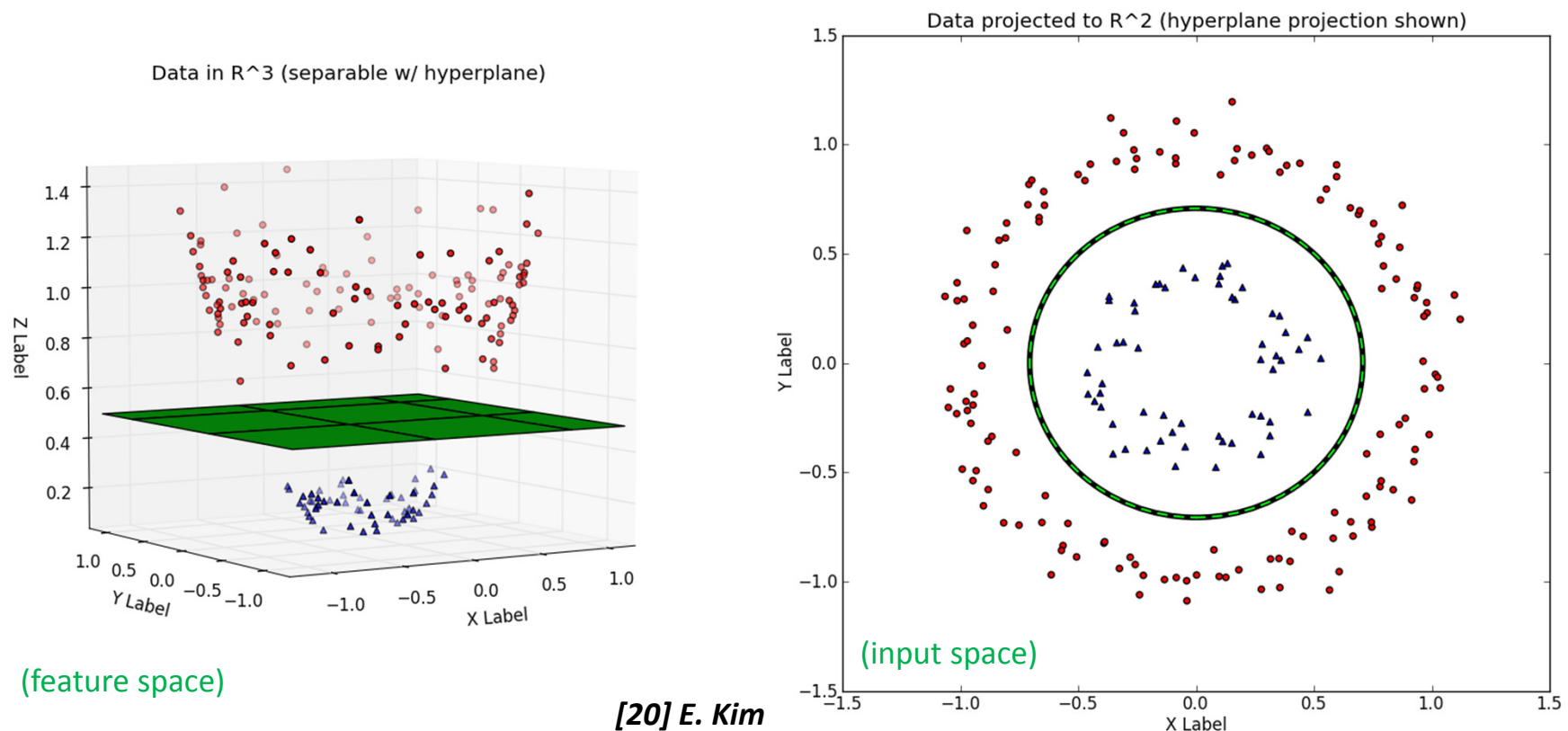


[20] E. Kim

- **Problems: Not clear which type of mapping (search); optimization is computationally expensive task**

Understanding Non-linear Transformations (3)

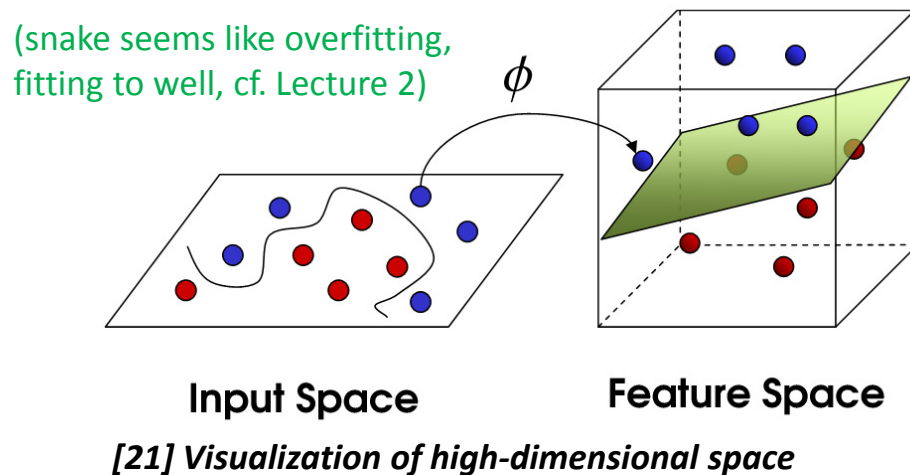
- Example: From 2 dimensional to 3 dimensional: $[x_1, x_2] = [x_1, x_2, x_1^2 + x_2^2]$
 - Separating hyperplane can be found and 'mapped back' to input space



➤ Appendix D shows how the 'Kernel Trick' uses non-linear transformations with SVMs

Visualization of SVs

- Problem: z-Space is infinite (unknown)
 - How can the Support Vectors (from existing points) be visualized?
 - Solution: non-zero alphas have been the identified support vectors
(solution of quadratic programming optimization will be a set of alphas we can visualize)
 - Support vectors exist in Z – space (just transformed original data points)
 - Example: million-D means a million-D vector for \mathbf{W}
 - But number of support vector is very low, expected E_{out} is related to #SVs
(generalization behaviour despite million-D & snake-like overfitting)



- Counting the number of support vectors remains to be a good indicator for generalization behaviour even when performing non-linear transforms and kernel methods that can lead to infinite-D spaces

(rule of thumb)

Parallel & Scalable PiSVM - Parameters

```
[train001@j3l02 pisvm-1.2.1]$ ./pisvm-train
Usage: svm-train [options] training_set_file [model_file]
options:
-s svm_type : set type of SVM (default 0)
    0 -- C-SVC
    1 -- nu-SVC
    2 -- one-class SVM
    3 -- epsilon-SVR
    4 -- nu-SVR
-t kernel_type : set type of kernel function (default 2)
    0 -- linear:  $u \cdot v$ 
    1 -- polynomial:  $(\gamma u \cdot v + \text{coef0})^{\text{degree}}$ 
    2 -- radial basis function:  $\exp(-\gamma |u - v|^2)$ 
    3 -- sigmoid:  $\tanh(\gamma u \cdot v + \text{coef0})$ 
-d degree : set degree in kernel function (default 3)
-g gamma : set gamma in kernel function (default 1/k)
-r coef0 : set coef0 in kernel function (default 0)
-c cost : set the parameter C of C-SVC, epsilon-SVR, and nu-SVR (default 1)
-n nu : set the parameter nu of nu-SVC, one-class SVM, and nu-SVR (default 0.5)
-p epsilon : set the epsilon in loss function of epsilon-SVR (default 0.1)
-m cachesize : set cache memory size in MB (default 40)
-e epsilon : set tolerance of termination criterion (default 0.001)
-h shrinking: whether to use the shrinking heuristics, 0 or 1 (default 1)
-b probability_estimates: whether to train a SVC or SVR model for probability estimates, 0 or 1 (default 0)
-wi weight: set the parameter C of class i to weight*C, for C-SVC (default 1)
-v n: n-fold cross validation mode
-o n: max. size of working set
-q n: max. number of new variables entering working set
flags:
-D: Assume the feature vectors are dense (default: sparse)
```

Training Rome on JURECA – Job Script (RBF)

- Use Rome Dataset with parallel & scalable piSVM tool
 - Parameters are equal to the serial libsvm and some additional

```
#!/bin/bash -x
#SBATCH--nodes=2
#SBATCH--ntasks=48
#SBATCH--ntasks-per-node=24
#SBATCH--output=mpi-out.%j
#SBATCH--error=mpi-err.%j
#SBATCH--time=01:00:00
#SBATCH--partition=batch
#SBATCH--mail-user=m.riedel@fz-juelich.de
#SBATCH--mail-type=ALL
#SBATCH--job-name=train-rome-rbf-2-48-24

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

### location data
TRAINDATA=/homea/hpclab/train001/data/rome/sdap_area_all_training.el

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

- **Note the tutorial reservation with `--reservation=bigdata-cpu` just valid for today on JURECA**

Testing Rome on JURECA – Job Script (RBF)

- Use Rome Dataset with parallel & scalable piSVM tool
 - Parameters are equal to the serial libsvm and some additional parameters for parallelization

```
#SBATCH --ntasks-per-node=24
#SBATCH --output=mpi-out.%j
#SBATCH --error=mpi-err.%j
#SBATCH --time=01:00:00
#SBATCH --partition=batch
#SBATCH --mail-user=m.riedel@fz-juelich.de
#SBATCH --mail-type=ALL
#SBATCH --job-name=pred-rome-2-48-24

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

### location data
TESTDATA=/homea/hpclab/train001/data/rome/sdap_area_all_test.el

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

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

■ **Note the tutorial reservation with `--reservation=bigdata-cpu` just valid for today on JURECA**

Testing Rome on JURECA – Check Outcome

- The output of the training run is a model file

- Used for input for the testing/predicting phase
- In-sample property → Support vectors of the model

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

- The output of the testing/predicting phase is accuracy

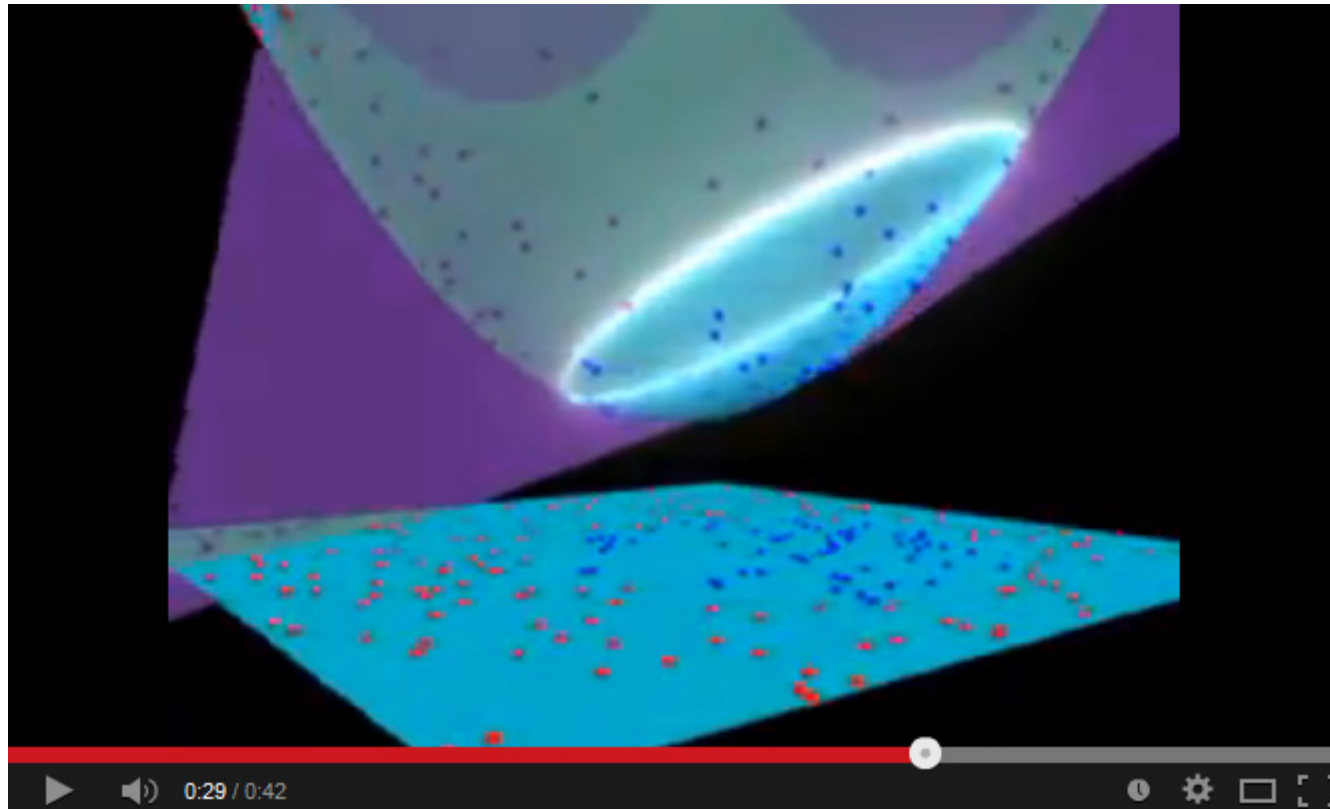
- Accuracy measurement of model performance

```
[train001@j3l02 pisvm-1.2.1]$ more mpi-out.15244544  
Accuracy = 98.0558% (684291/697859) (classification)  
Mean squared error = 0.195456 (regression)  
Squared correlation coefficient = 0.968742 (regression)
```

- Output of linear SVM was as follows:

```
[train001@j3l02 pisvm-1.2.1]$ more mpi-out.15244542  
Accuracy = 91.5994% (639235/697859) (classification)  
Mean squared error = 1.04794 (regression)  
Squared correlation coefficient = 0.835385 (regression)
```


[Video] SVM with Polynomial Kernel Example



[22] YouTube, SVM with Polynomial Kernel

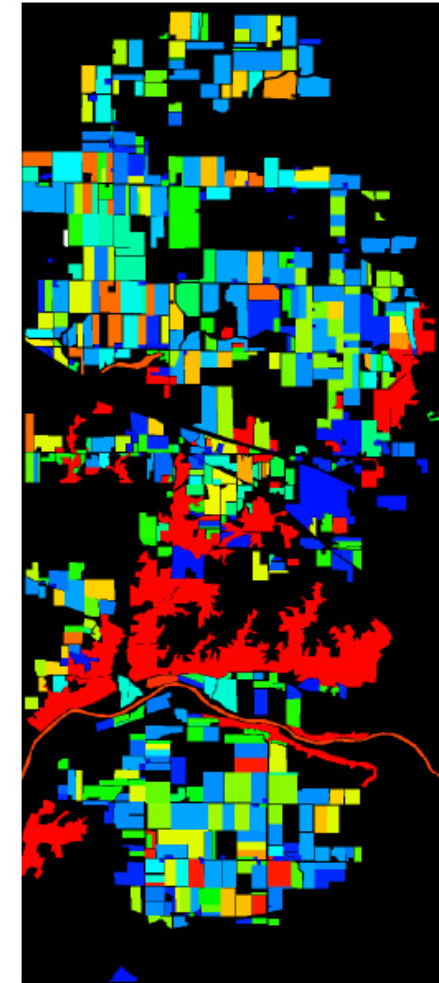
Indian Pines Dataset – Preprocessing

Corrected by JPL

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

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

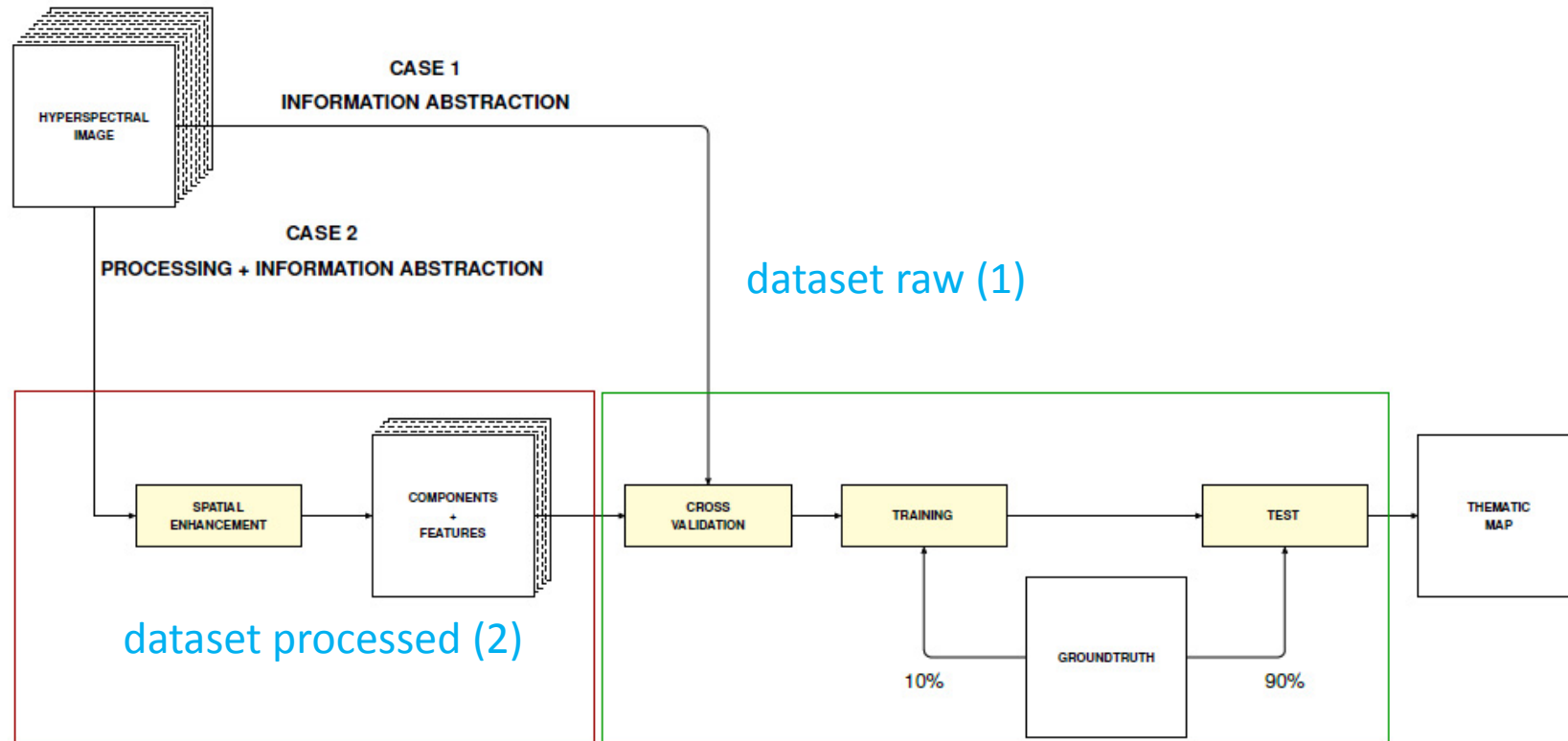
[23] G. Cavallaro and M. Riedel, et al. , 2015



(non-linearly separable) dataset

Indian Pines – Experimental Setup

Two Cases



Feature Enhancement & Selection

Kernel Principle Component Analysis (**KPCA**)

Extended Self-Dual Attribute Profile (**ESDAP**)

Nonparametric weighted feature extraction (**NWFE**)

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

Publicly Available Datasets – Location

■ *Indian Pines Dataset Raw and Processed*



[24] *Indian Pines Raw and Processed*

Indian pines: raw and processed

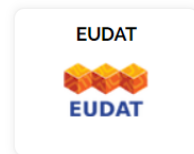
by [Unknown]

Dec 22, 2016

Last updated at Jan 11, 2018

Abstract: 1) Indian raw: 1417x614x200 (training 10% and test) 2) Indian processed: 1417x614x30 (training 10% and test)

PID: [11304/7e8eec8e-ad61-11e4-ac7e-860aa0063d1f](#) [Copy](#)



Files

Name	Size
> indian_processed_test.el	105.59MB
> indian_processed_training.el	11.73MB
> indian_raw_test.el	747.13MB
> indian_raw_training.el	83.01MB

Basic metadata

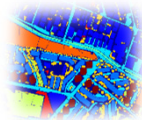
Open Access	True ✓
License	
Contact Email	
Publication Date	2015-02-04
Contributors	
Resource Type	Category Other
Alternate identifiers	172 Type B2SHARE_V1_ID http://hdl.handle.net/11304/9ec5eac8-61b4-4617-ae1c-1f8c8cd3cd74 Type ePIC_PID
Publisher	https://b2share.eudat.eu
Language	en

```
[train001@jrl07 indianpines]$ pwd
/homea/hpclab/train001/data/indianpines
[train001@jrl07 indianpines]$ ls -al
total 925344
drwxr-xr-x 2 train001 hpclab      512 Jul  7  2016 .
drwxr-xr-x 5 train001 hpclab      512 Jan 14 13:10 ..
-rw-r--r-- 1 train001 hpclab       36 Jul  7  2016 b2share.txt
-rw-r--r-- 1 train001 hpclab 105594346 Feb  5  2015 indian_processed_test.el
-rw-r--r-- 1 train001 hpclab 11732509 Feb  5  2015 indian_processed_training.el
-rw-r--r-- 1 train001 hpclab 747125597 Feb  5  2015 indian_raw_test.el
-rw-r--r-- 1 train001 hpclab 83014311 Feb  5  2015 indian_raw_training.el
```

Inspecting and Understanding the Indian Pines Dataset

- Dataset raw (1)

Class + *Original Spectral Bands*



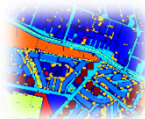
200 spectral bands

48 1:0.365 2:0.361 3:0.356 209:0.333 220:0.978

libSVM

- dataset processed (2)

Class + *Transformed Features*



30 image features

48 1:0.245 2:0.360 3:0.326 29:0.241 30:0.878

libSVM

Exercises – Indian Pines (Raw) piSVM Runs



HPC Environment – Modules Revisited

- **Module** environment tool
 - Avoids to manually setup environment information for every application
 - Simplifies shell initialization and lets users easily modify their environment
- **Module avail**
 - Lists all available modules on the HPC system (e.g. compilers, MPI, etc.)
- **Module spider**
 - Find modules in the installed set of modules and more information
- **Module load → needed before piSVM run**
 - Loads particular modules into the current work environment, E.g.:

```
[train001@jrl05 jsc_mpi]$ module load Intel/2018.1.163-GCC-5.4.0  
[train001@jrl05 jsc_mpi]$ module load IntelMPI/2018.1.163
```


Parallel & Scalable PiSVM – Parameters

```
[train001@j3l02 pisvm-1.2.1]$ ./pisvm-train
Usage: svm-train [options] training_set_file [model_file]
options:
```

```
-s svm_type : set type of SVM (default 0)
    0 -- C-SVC
    1 -- nu-SVC
    2 -- one-class SVM
    3 -- epsilon-SVR
    4 -- nu-SVR
-t kernel_type : set type of kernel function (default 2)
    0 -- linear:  $u \cdot v$ 
    1 -- polynomial:  $(\gamma u \cdot v + \text{coef0})^{\text{degree}}$ 
    2 -- radial basis function:  $\exp(-\gamma |u - v|^2)$ 
    3 -- sigmoid:  $\tanh(\gamma u \cdot v + \text{coef0})$ 
-d degree : set degree in kernel function (default 3)
-g gamma : set gamma in kernel function (default 1/k)
```

```
-r coef0 : set coef0 in kernel function (default 0)
-c cost : set the parameter C of C-SVC, epsilon-SVR, and nu-SVR (default 1)
-n nu : set the parameter nu of nu-SVC, one-class SVM, and nu-SVR (default 0.5)
-p epsilon : set the epsilon in loss function of epsilon-SVR (default 0.1)
-m cachesize : set cache memory size in MB (default 40)
-e epsilon : set tolerance of termination criterion (default 0.001)
-h shrinking: whether to use the shrinking heuristics, 0 or 1 (default 1)
-b probability_estimates: whether to train a SVC or SVR model for probability estimates, 0 or 1 (default 0)
-wi weight: set the parameter C of class i to weight*C, for C-SVC (default 1)
-v n: n-fold cross validation mode
-o n: max. size of working set
-q n: max. number of new variables entering working set
flags:
-D: Assume the feature vectors are dense (default: sparse)
```

- **C-SVC:** The cost (C) in this case refers to a soft-margin specifying how much error is allowed and thus represents a regularization parameter that prevents overfitting → more details tomorrow
- **nu-SVC:** nu in this case refers to values between 0 and 1 and thus represents a lower and upper bound on the number of examples that are support vectors and that lie on the wrong side of the hyperplane

Training Indian Pines (Raw) on JURECA – Job Script (RBF)

- Use Indian Pines Dataset with parallel & scalable piSVM tool
 - Parameters are equal to the serial libsvm and some additional parameters for parallelization

```
#!/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
```

- **Note the tutorial reservation with `--reservation=ml-hpc-2` just valid for today on JURECA**

Testing Indian Pines (Raw) on JURECA – Job Script

- Use Rome Dataset with parallel & scalable piSVM tool
 - Parameters are equal to the serial libsvm and some additional parameters for parallelization

```
#!/bin/bash -x
#SBATCH -n=4
#SBATCH -t=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
```

- **Note the tutorial reservation with `--reservation=ml-hpc-2` just valid for today on JURECA**

Testing/Predicting Rome on JURECA – Check Outcome

- The output of the training run is a model file

- Used for input for the testing/predicting phase
- In-sample property → Support vectors of the model

```
[train001@j3l02 pisvm-1.2.1]$ head indian_raw_training.el.model
svm_type c_svc
kernel_type rbf
gamma 8
nr_class 52
total_sv 32911
rho 1.66407 -0.0618225 -0.411599 1.29602 -0.178103 0.0331523 1.02995 0.157358
00999 0.502322 -0.201472 -1.37352 0.905974 -0.360055 -0.470548 -0.857891 -0.0
-0.914877 -0.733324 -0.536122 -0.689851 -0.873 -0.420949 -0.577946 -0.112022
45 -1.67725 -1.09567 -0.9257 -1.43485 -1.67098 -0.777169 -1.16524 -1.03982 -1
9 -1.56535 -2.12569 -1.63647 -2.30329 -1.53157 -1.39873 -2.59068 -2.30643 -3.
1 52381 -1 90054 -3 14899 -2 12109 -2 0321 -2 74675 -2 77067 -1 6752 -1 3931
```

- The output of the testing/predicting phase is accuracy

- Accuracy measurement of model performance (cf. Lecture 1)
- The job output file consists of a couple of lines

```
[train001@j3l02 pisvm-1.2.1]$ more mpi-out.15244571
Accuracy = 40.0828% (120471/300555) (classification)
Mean squared error = 453.392 (regression)
Squared correlation coefficient = 0.137346 (regression)
```

SVM Multi-class Classification - One vs. One

- Multi-class classification common in science & engineering
 - Requires different approach as previous 'binary classification' (2 classes)
 - Cf. associated remote sensing SVM application (e.g. 52 land cover classes)
 - Reduce the problem of multiclass to multiple binary classification problems
(advanced topic – required much more study – here just the two most popular approaches)
- One vs. One (all pairs) classification
 - Given $K > 2$ classes, this approach creates $\binom{K}{2}$ different SVMs ($K(K-1)/2$)
 - Each of the different SVMs compares a pair of classes (i.e. binary classifier)
 - Classification is done by using test data points with each of the classifiers
 - Count number of times that each point is assigned to each of the k classes
 - Class is which it was most frequently assigned in $\binom{K}{2}$ pairwise classification

(the more classes – the more SVMs are created to perform pairwise classification – the more computational complexity)

[6] *An Introduction to Statistical Learning*

- One vs. one multi-class classification creates different SVMs that compare each a pair of k classes

SVM Multi-class Classification – One vs. All (aka Rest)

- One vs. All (aka Rest) classification

- Given $K > 2$ classes, this approach fits only K SVMs
- Each time one of the K classes is compared to the remaining $K-1$ classes
- Coefficients that result from fitting an SVM comparing the k th class (coded as +1) to all others (coded as -1) are $\beta_{0k}, \beta_{1k}, \dots, \beta_{pk}$
- Classification with testset data x^* and compute confidence score
- Assign the testset data to the class for which the following is largest:

$$\beta_{0k} + \beta_{1k}x_1^* + \beta_{2k}x_2^* + \dots + \beta_{pk}x_p^*$$

- Reasoning: high level of confidence that the test data points belong to the k th class rather than to any of the other classes

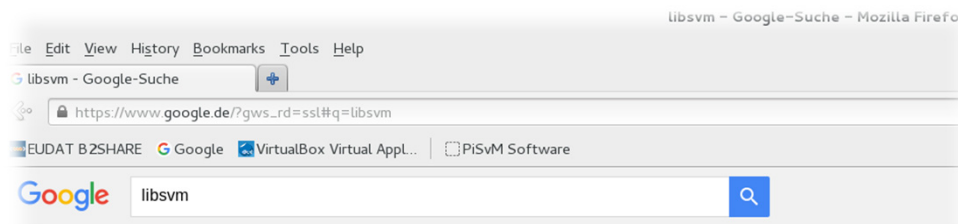
(less SVMs are created – but more comparisons are done while creating the classifiers – can be computationally intensive)

[6] *An Introduction to Statistical Learning*

- One vs. all multi-class classification creates K SVMs comparing it with to the remaining $K-1$ classes

LibSVM – Defacto Standard SVM Implementation

- Free available tool
 - Includes Sequential Minimal Optimization (SMO) implementation



Ungefähr 343.000 Ergebnisse (0,47 Sekunden)

LIBSVM -- A Library for Support Vector Machines
<https://www.csie.ntu.edu.tw/~cjlin/libsvm/> ▾ Diese Seite übersetzen
LIBSVM -- A Library for Support Vector Machines. Chih-Chung Chang and Chih-Jen Lin. V...
released on December 14, 2015. It conducts some minor ...

Libsvm faq
I would like to use libsvm in my
See the previous FAQ.

LIBSVM Tools
LIBSVM Tools. Last modified:
01/26/2016 23:20:07. This page ...

Weitere Ergebnisse von ntu.edu.tw »

Download LIBSVM
LIBSVM. Chih-Chung Chang and
Chih-Jen Lin. Most available ...

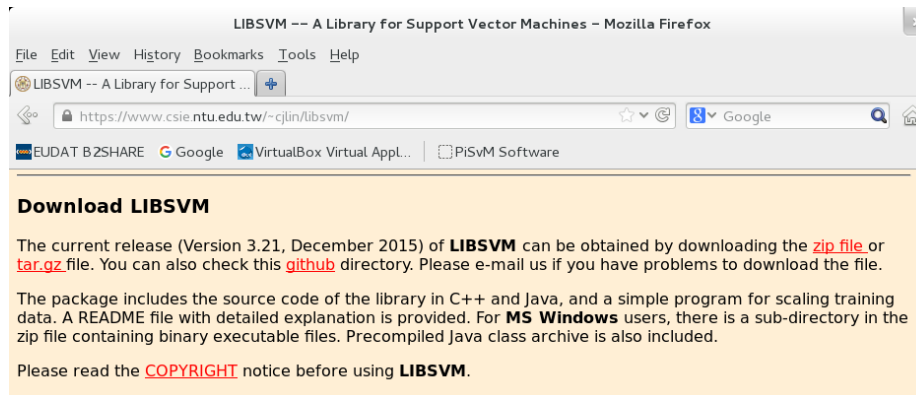
LIBSVM Data: Classific
LIBSVM Data: Classification ...
data sets (references given ...



[25] *LibSVM Webpage*

LibSVM – Download

- Download tar.gz (or in Windows zip bundle)



[25] LibSVM Webpage

- Put package in a folder of your choice
 - Alternatively copy file to your usual working environment

```
adminuser@linux-8djg:~/tools> scp libsvm-3.21.tar.gz mriedel@jureca.fz-juelich.de:/homeb/zam/mriedel
libsvm-3.21.tar.gz                                     100% 827KB 827.4KB/s  00:00
```

```
-bash-4.2$ ls -al
total 64
drwxr-xr-x  2 mriedel zam   512 Jul  6 20:00 .
drwxr-xr-x 29 mriedel zam 32768 Jul  6 19:58 ..
-rw-r--r--  1 mriedel zam 847291 Jul  6 20:00 libsvm-3.21.tar.gz
-bash-4.2$ pwd
/homeb/zam/mriedel/serialtools
```

LibSVM – Make (only in UNIX)

- Use make to generate executables (needs g++ compiler)

```
-bash-4.2$ pwd
/homeb/zam/mriedel/serialtools/libsvm-3.21
-bash-4.2$ make
g++ -Wall -Wconversion -O3 -fPIC -c svm.cpp
g++ -Wall -Wconversion -O3 -fPIC svm-train.c svm.o -o svm-train -lm
g++ -Wall -Wconversion -O3 -fPIC svm-predict.c svm.o -o svm-predict -lm
g++ -Wall -Wconversion -O3 -fPIC svm-scale.c -o svm-scale
```

- Check executables important for us

```
-bash-4.2$ pwd
/homeb/zam/mriedel/serialtools/libsvm-3.21
-bash-4.2$ ls -al
total 896
drwxr-xr-x 8 mriedel zam 32768 Jul  6 20:05 .
drwxr-xr-x 3 mriedel zam  512 Jul  6 20:03 ..
-rw-r--r-- 1 mriedel zam  1497 Dec 14 2015 COPYRIGHT
-rw-r--r-- 1 mriedel zam 83089 Dec 14 2015 FAQ.html
-rw-r--r-- 1 mriedel zam 27670 Dec 14 2015 heart_scale
drwxr-xr-x 3 mriedel zam  512 Dec 14 2015 java
-rw-r--r-- 1 mriedel zam  732 Dec 14 2015 Makefile
-rw-r--r-- 1 mriedel zam 1136 Dec 14 2015 Makefile.win
drwxr-xr-x 2 mriedel zam  512 Dec 14 2015 matlab
drwxr-xr-x 2 mriedel zam  512 Dec 14 2015 python
-rw-r--r-- 1 mriedel zam 28679 Dec 14 2015 README
-rw-r--r-- 1 mriedel zam 64836 Dec 14 2015 svm.cpp
-rw-r--r-- 1 mriedel zam  477 Dec 14 2015 svm.def
-rw-r--r-- 1 mriedel zam  3382 Dec 14 2015 svm.h
-rw-r--r-- 1 mriedel zam 100224 Jul  6 20:05 svm.o
-rwxr-xr-x 1 mriedel zam 78270 Jul  6 20:05 svm-predict
-rw-r--r-- 1 mriedel zam  5536 Dec 14 2015 svm-predict.c
-rwxr-xr-x 1 mriedel zam 18587 Jul  6 20:05 svm-scale
-rw-r--r-- 1 mriedel zam  8539 Dec 14 2015 svm-scale.c
drwxr-xr-x 5 mriedel zam  512 Dec 14 2015 svm-try
-rwxr-xr-x 1 mriedel zam 78509 Jul  6 20:05 svm-train
-rw-r--r-- 1 mriedel zam  8588 Dec 14 2015 svm-train.c
drwxr-xr-x 2 mriedel zam  512 Dec 14 2015 tools
drwxr-xr-x 2 mriedel zam  512 Dec 14 2015 windows
```

(use in testing phase)

(use in training phase)

[25] LibSVM Webpage

LibSVM – svm-train Parameters

■ Important parameters (training phase)

```
-bash-4.2$ ./svm-train
```

```
Usage: svm-train [options] training_set_file [model_file]
```

(we need a training set file)

```
-s svm_type : set type of SVM (default 0)
```

(take default here = C-SVC)

```
  0 -- C-SVC                (multi-class classification)
```

```
  1 -- nu-SVC                (multi-class classification)
```

```
  2 -- one-class SVM
```

```
  3 -- epsilon-SVR           (regression)
```

```
  4 -- nu-SVR                (regression)
```

```
-t kernel_type : set type of kernel function (default 2)
```

(in this lecture we have just 'linear kernels')

```
  0 -- linear: u'*v
```

```
  1 -- polynomial: (gamma*u'*v + coef0)^degree
```

```
  2 -- radial basis function: exp(-gamma*|u-v|^2)
```

```
  3 -- sigmoid: tanh(gamma*u'*v + coef0)
```

```
  4 -- precomputed kernel (kernel values in training_set_file)
```

```
-d degree : set degree in kernel function (default 3)
```

```
-g gamma : set gamma in kernel function (default 1/num_features)
```

```
-r coef0 : set coef0 in kernel function (default 0)
```

```
-c cost : set the parameter C of C-SVC, epsilon-SVR, and nu-SVR (default 1)
```

(Regularization Parameter)

```
-n nu : set the parameter nu of nu-SVC, one-class SVM, and nu-SVR (default 0.5)
```

```
-p epsilon : set the epsilon in loss function of epsilon-SVR (default 0.1)
```

```
-m cachesize : set cache memory size in MB (default 100)
```

```
-e epsilon : set tolerance of termination criterion (default 0.001)
```

```
-h shrinking : whether to use the shrinking heuristics, 0 or 1 (default 1)
```

```
-b probability_estimates : whether to train a SVC or SVR model for probability estimates, 0 or 1 (default 0)
```

```
-wi weight : set the parameter C of class i to weight*C, for C-SVC (default 1)
```

```
-v n: n-fold cross validation mode
```

```
-q : quiet mode (no outputs)
```

Training Examples

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

[25] LibSVM Webpage

LibSVM – svm-predict Parameters

- Important parameters (testing phase)

```
-bash-4.2$ ./svm-predict
```

```
Usage: svm-predict [options] test_file model_file output_file
```

```
options:
```

```
-b probability_estimates: whether to predict probability estimates, 0 or 1 (default 0); for one-class SVM only 0 is supported
```

```
-q : quiet mode (no outputs)
```

(the model file is generated in the training phase → the support vectors found in optimization)

(test file is a testing dataset set aside to be used once training is finished)

(output file gives us indications how each sample was classified)

Testing Examples

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

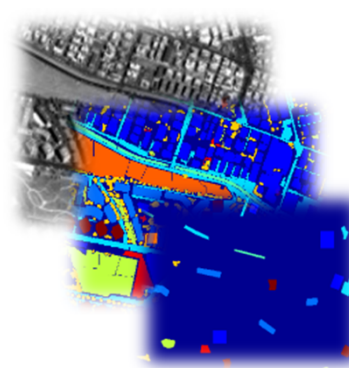
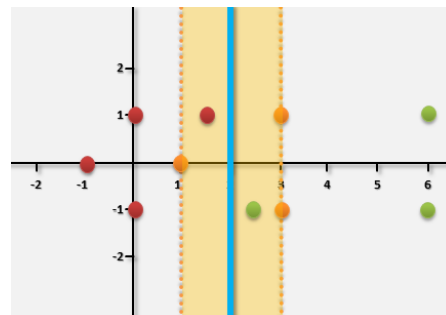
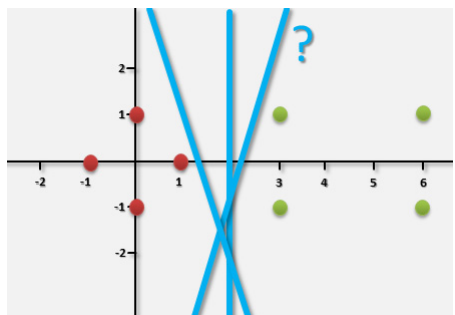
Review of Parallel SVM Implementations

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

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

Parallel and Scalable Machine Learning – piSVM

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



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



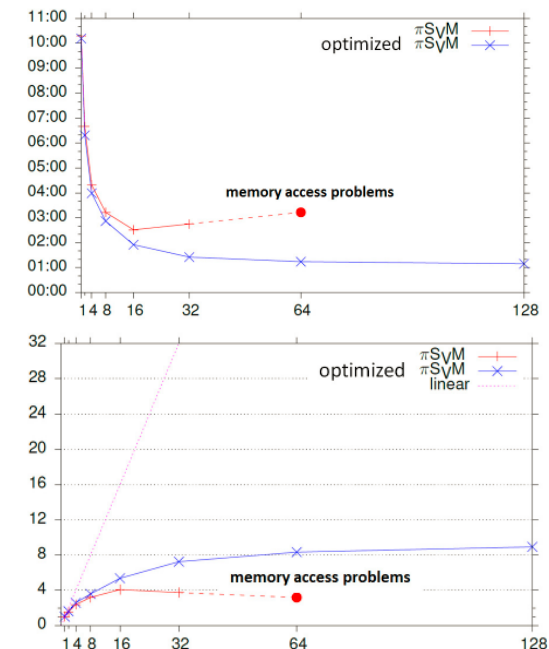
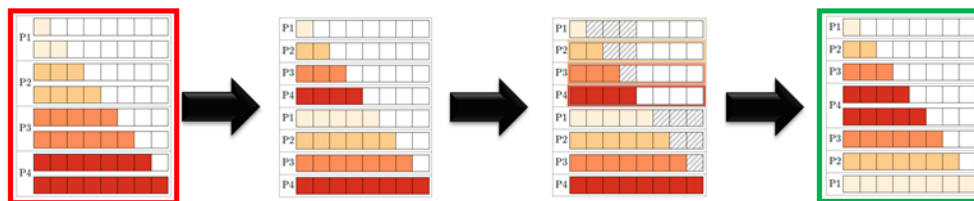
[23] G. Cavallaro & M. Riedel et al., ‘On Understanding Big Data Impacts in Remotely Sensed Image Classification Using Support Vector Machine Methods’, *Journal of Applied Earth Observations and Remote Sensing*

Parallel SVM with MPI Technique – piSVM Implementation



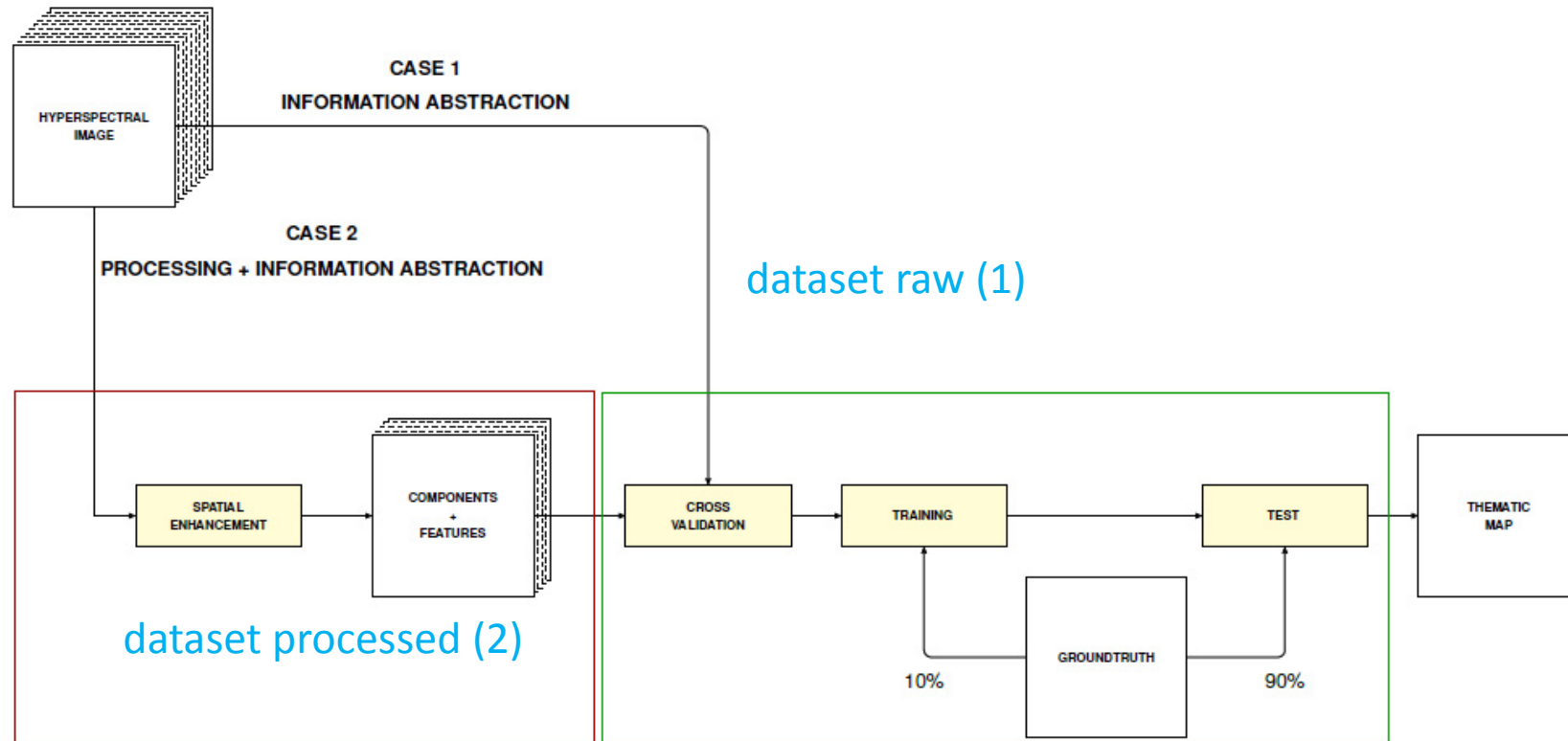
[27] piSVM on SourceForge, 2008

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



Indian Pines – Experimental Setup

Two Cases



Feature Enhancement & Selection

Kernel Principle Component Analysis (**KPCA**)

Extended Self-Dual Attribute Profile (**ESDAP**)

Nonparametric weighted feature extraction (**NWFE**)

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

Exercises – Indian Pines (Processed) piSVM Runs



Training Indian Pines (Proc) on JURECA – Job Script (RBF)

- Use Indian Pines Dataset with parallel & scalable piSVM tool
 - Parameters are equal to the serial libsvm and some additional parameters for parallelization

```
#!/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_processed_training.el

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

- **Note the tutorial reservation with `--reservation=bigdata-cpu` just valid for today on JURECA**

Testing Indian Pines (Proc) on JURECA – Job Script

- Use Rome Dataset with parallel & scalable piSVM tool
 - Parameters are equal to the serial libsvm and some additional parameters for parallelization

```
#!/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_processed_test.el

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

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

- **Note the tutorial reservation with `--reservation=bigdata-cpu` just valid for today on JURECA**

Testing/Predicting Rome on JURECA – Check Outcome

- The output of the training run is a model file

- Used for input for the testing/predicting phase
- In-sample property → Support vectors of the model

```
[train001@j3l02 pisvm-1.2.1]$ head indian_processed_training.el.model
svm_type c_svc
kernel_type rbf
gamma 8
nr_class 52
total_sv 26062
rho 1.07276 -0.440311 -0.640771 0.548192 -0.722158 -0.168075 0.551069 -
6 -0.000349507 0.314899 -0.270607 -0.941572 0.328627 -0.241276 -0.6386
0.818067 -1.85287 -0.793607 -0.513811 -0.754098 -0.870286 -0.489813 -0.
041 -0.138668 -1.62263 -0.856795 -0.43055 -0.751147 -1.68925 -0.704193
47 -0.995375 -0.973193 -1.13304 -1.48614 -0.925452 -1.1306 -3.10747 -0.
```

- The output of the testing/predicting phase is accuracy

- Accuracy measurement of model performance (cf. Lecture 1)
- The job output file consists of a couple of lines

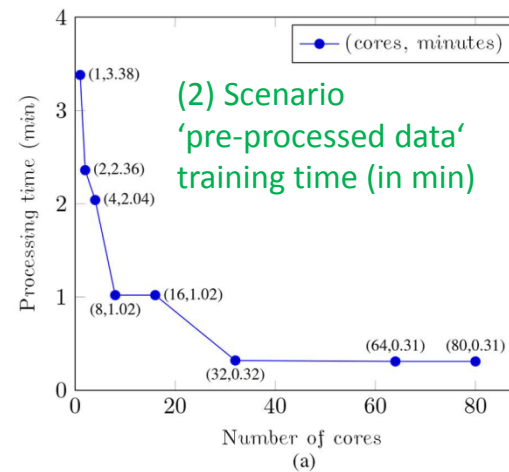
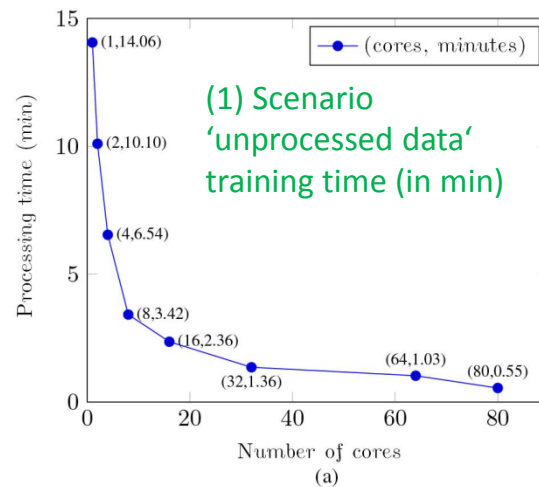
```
[train001@j3l02 pisvm-1.2.1]$ more mpi-out.15244572
Accuracy = 77.9678% (234336/300555) (classification)
Mean squared error = 153.787 (regression)
Squared correlation coefficient = 0.601418 (regression)
```

Exercises – Indian Pines – Change Number of Nodes



Parallelization Benefit: Lower-Time-To-Solution

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



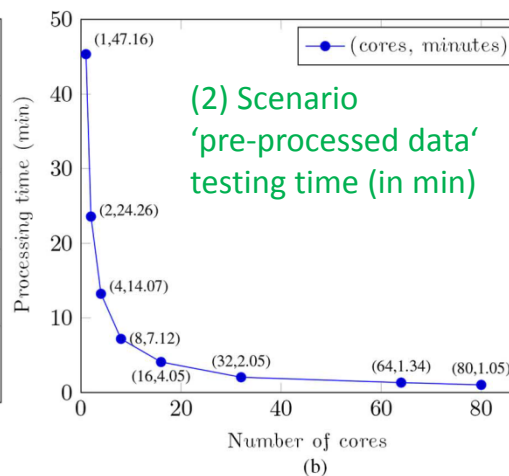
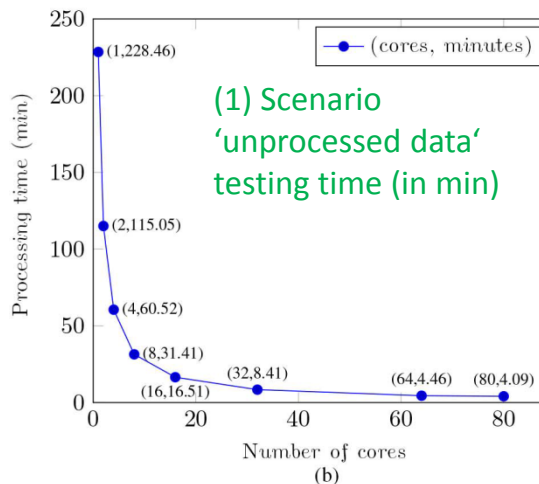
manual & serial activities (in min)

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

'big data' is not always better data

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

(cf. Importance of feature engineering above)



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

Exercises – Indian Pines – Perform n-fold Cross-Validation



Parallelization Benefit – 10-fold Cross-Validation

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

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

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

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

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

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

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

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

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

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

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

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

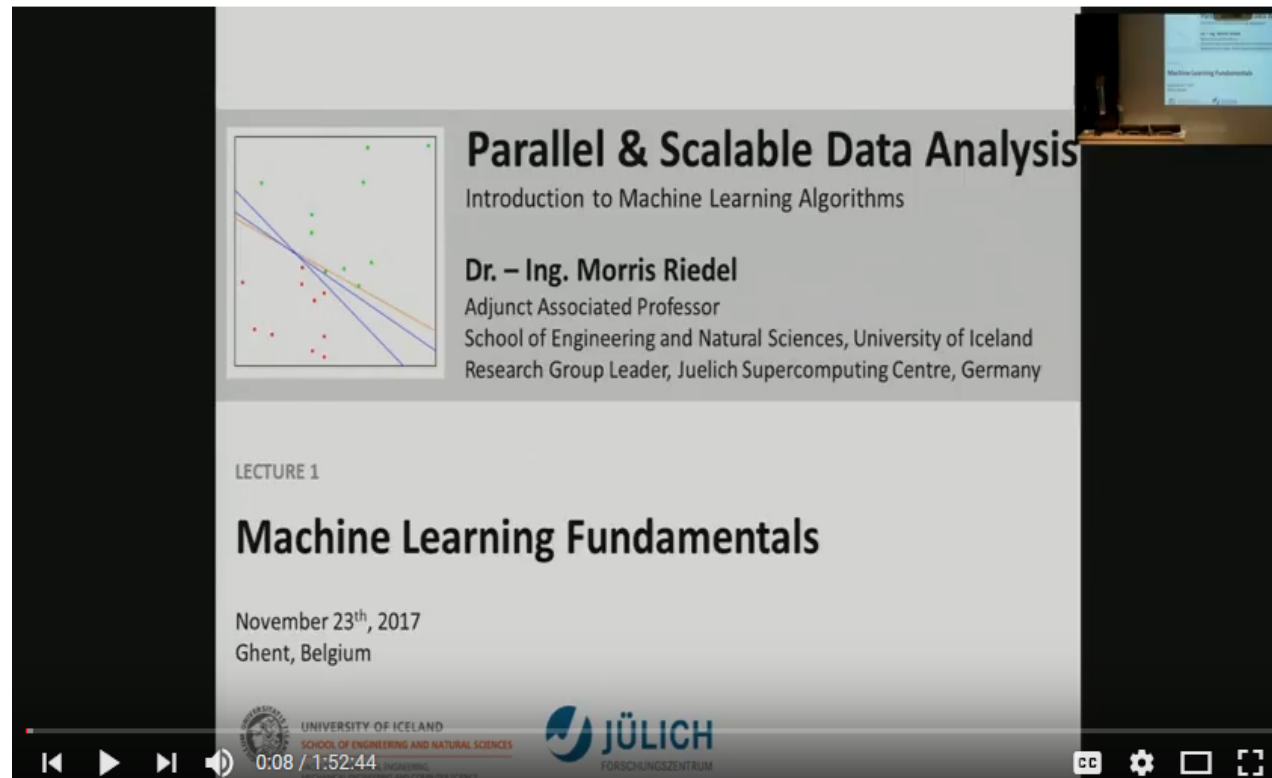


Prevent Overfitting for better 'out-of-sample' generalization



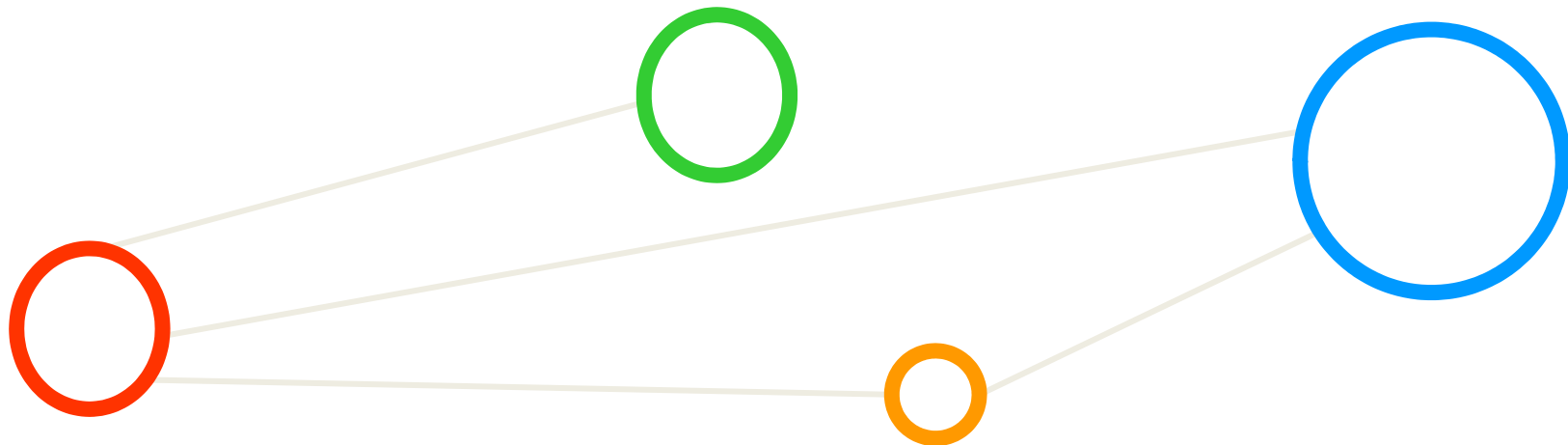
[12] Stop Overfitting, YouTube

[YouTube Lectures] More about parallel piSVM & HPC



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

Appendix A: CRISP-DM Process

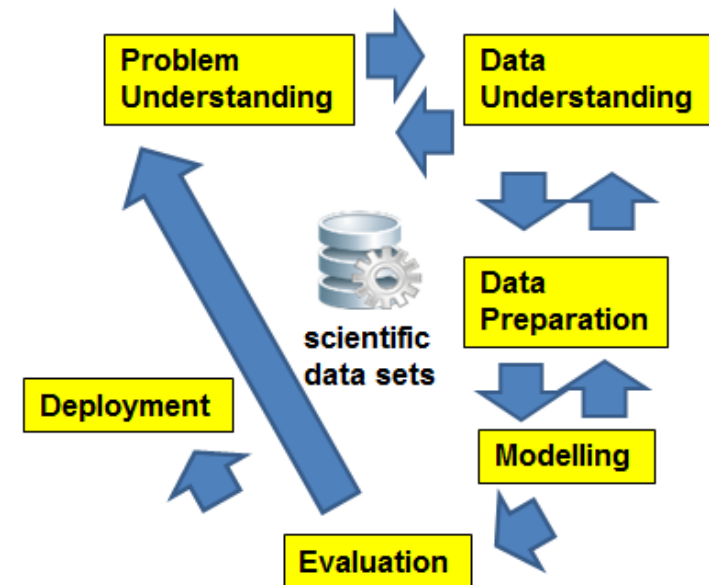


Summary: Systematic Process

- 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

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



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

1 – Problem (Business) Understanding

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

- Task A – Determine Business Objectives

[11] CRISP-DM User Guide

- Background, Business Objectives, Business Success Criteria

- Task B – Situation Assessment

- Inventory of Resources, Requirements, Assumptions, and Constraints
 - Risks and Contingencies, Terminology, Costs & Benefits

- Task C – Determine Data Mining Goal

- Data Mining Goals and Success Criteria

- Task D – Produce Project Plan

- Project Plan
 - Initial Assessment of Tools & Techniques

2 – Data Understanding

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

[11] 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

[11] 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;

[11] 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

[11] 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

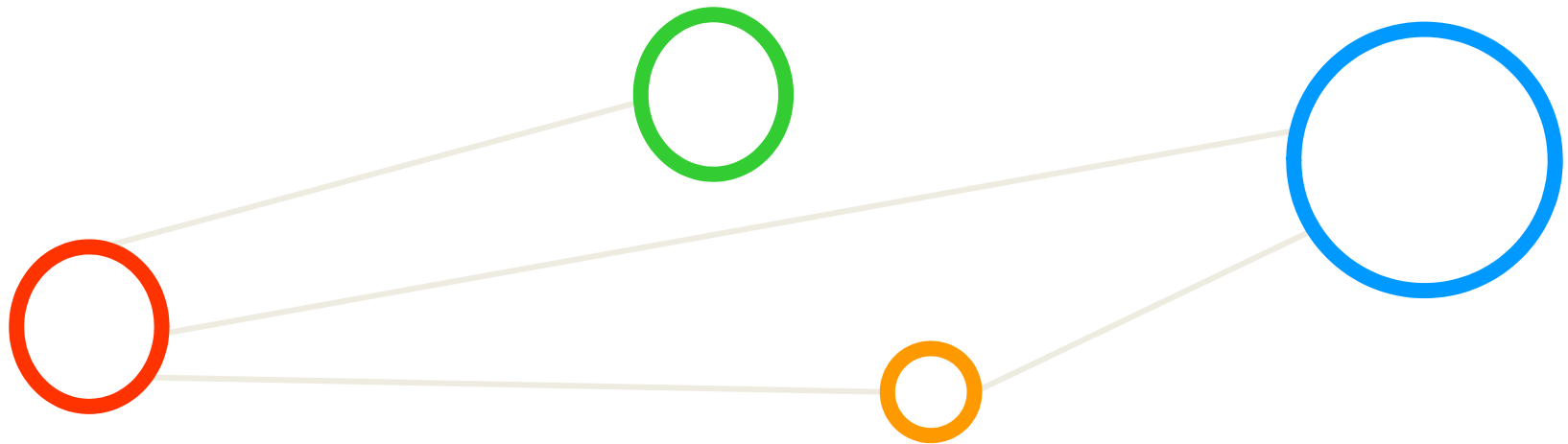
6 – Deployment

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

[11] CRISP-DM User Guide

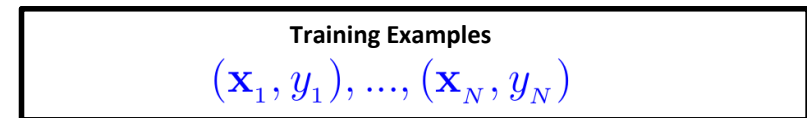
- Task A – Plan Deployment
 - Establish a deployment plan
- Task B – Plan Monitoring and Maintenance
 - Create a monitoring and maintenance plan
- Task C – Product Final Report
 - Create final report and provide final presentation
- Task D – Review Project
 - Document experience, provide documentation

Appendix B: Learning Theory Basics



Learning Approaches – Supervised Learning – Formalization

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



(historical records, groundtruth data, examples)

- 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]

[6] *An Introduction to Statistical Learning*

Feasibility of Learning

- Statistical Learning Theory deals with the problem of finding a predictive function based on data

[13] Wikipedia on 'statistical learning theory'

- Theoretical framework underlying practical learning algorithms
 - E.g. Support Vector Machines (SVMs)
 - Best understood for 'Supervised Learning'
- Theoretical background used to solve 'A learning problem'
 - Inferring one 'target function' that maps between input and output
 - Learned function can be used to predict output from future input (fitting existing data is not enough)

Unknown Target Function

$$f : X \rightarrow Y$$

(ideal function)

Mathematical Building Blocks (1)

Unknown Target Function

$$f : X \rightarrow Y$$

(ideal function)

*Elements we
not exactly
(need to) know*



Training Examples

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

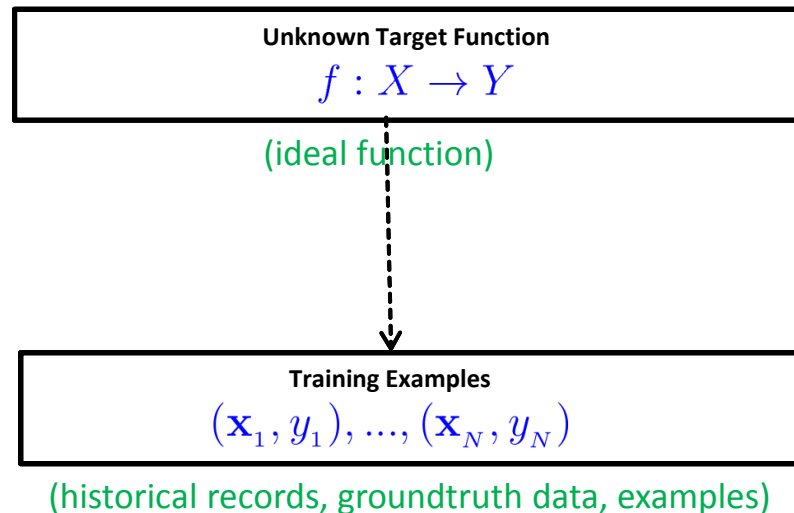
(historical records, groundtruth data, examples)

*Elements we
must and/or
should have and
that might raise
huge demands
for storage*

*Elements
that we derive
from our skillset
and that can be
computationally
intensive*

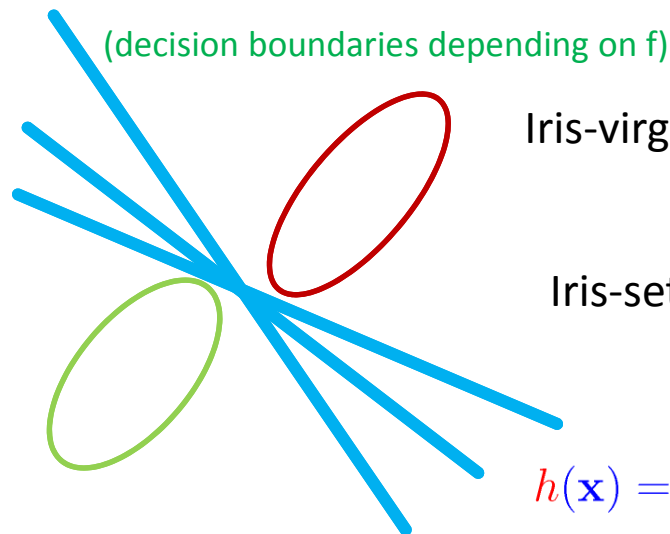
*Elements
that we
derive from
our skillset*

Mathematical Building Blocks (1) – Our Linear Example



1. Some pattern exists
2. No exact mathematical formula (i.e. target function)
3. Data exists

(if we would know the exact target function we dont need machine learning, it would not make sense)



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

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

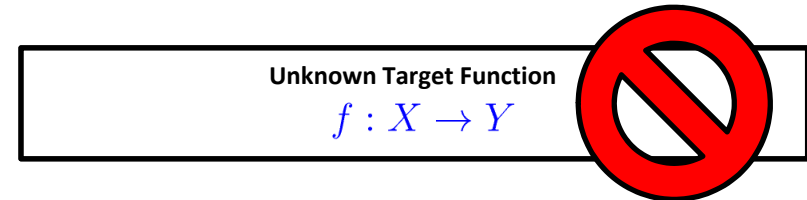
(w_i and threshold are still unknown to us)

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

(we search a function similiar like a target function)

Feasibility of Learning – Hypothesis Set & Final Hypothesis

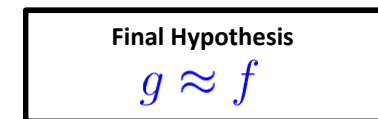
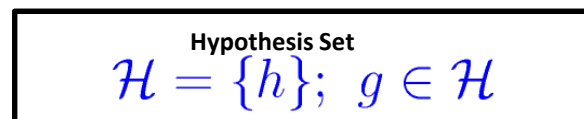
- The ‘ideal function’ will remain unknown in learning
 - Impossible to know and learn from data
 - If known a straightforward implementation would be better than learning
 - E.g. hidden features/attributes of data not known or not part of data
- But ‘(function) approximation’ of the target function is possible
 - Use training examples to learn and approximate it
 - Hypothesis set \mathcal{H} consists of m different hypothesis (candidate functions)



$$\mathcal{H} = \{h_1, \dots, h_m\};$$

‘select one function’
that best approximates

$$g : X \rightarrow Y$$



Feasibility of Learning – Understanding the Hypothesis Set

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

$$\mathcal{H} = \{h_1, \dots, h_m\};$$

(all candidate functions
derived from models
and their parameters)

- Already a change in model parameters of h_1, \dots, h_m means a completely different model

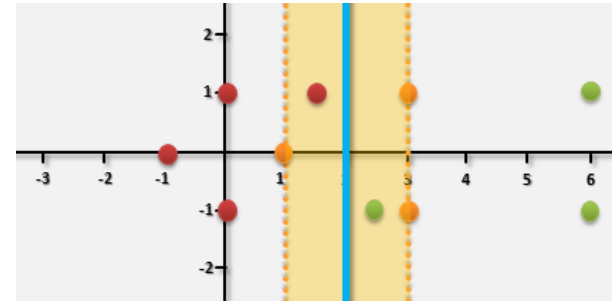
‘select one function’
that best approximates

Final Hypothesis

$$g \approx f$$

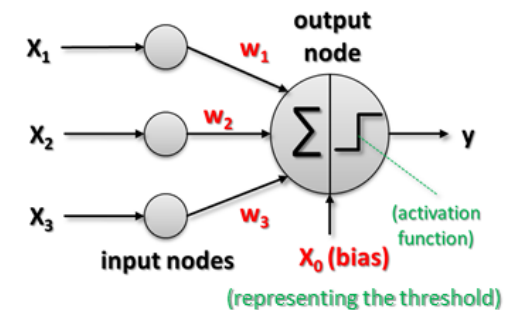


h_1



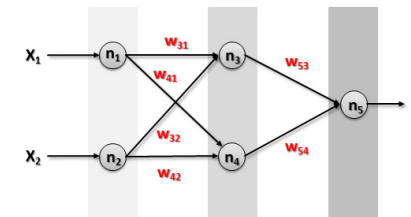
(e.g. support vector machine model)

h_2



(e.g. linear perceptron model)

h_m



(e.g. artificial neural network model)

Mathematical Building Blocks (2)

Unknown Target Function

$$f : X \rightarrow Y$$

(ideal function)

*Elements we
not exactly
(need to) know*



Training Examples

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

(historical records, groundtruth data, examples)

*Elements we
must and/or
should have and
that might raise
huge demands
for storage*

Final Hypothesis

$$g \approx f$$

*Elements
that we derive
from our skillset
and that can be
computationally
intensive*

Hypothesis Set

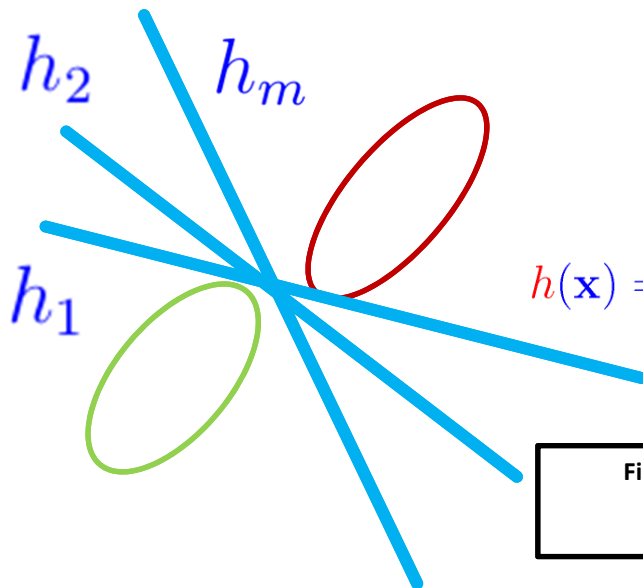
$$\mathcal{H} = \{h\}; g \in \mathcal{H}$$

(set of candidate formulas)

*Elements
that we
derive from
our skillset*

Mathematical Building Blocks (2) – Our Linear Example

(decision boundaries depending on f)



▪ Already a change in model parameters of h_1, \dots, h_m means a completely different model

$$\mathcal{H} = \{h_1, \dots, h_m\};$$

(we search a function similar like a target function)

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

Final Hypothesis

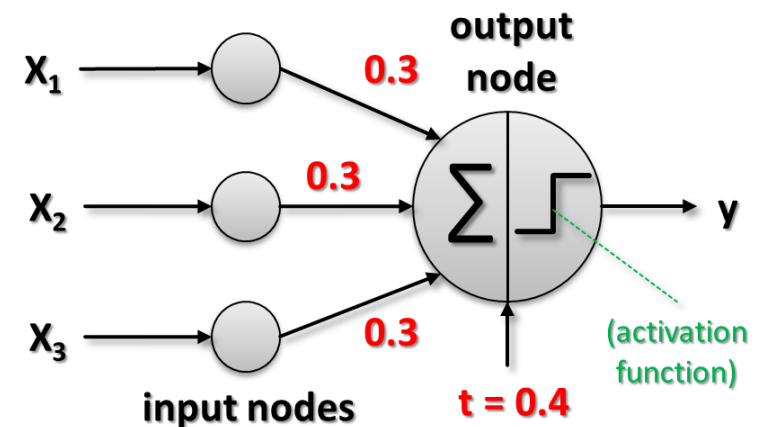
$$g \approx f$$

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

Hypothesis Set

$$\mathcal{H} = \{h\}; g \in \mathcal{H}$$

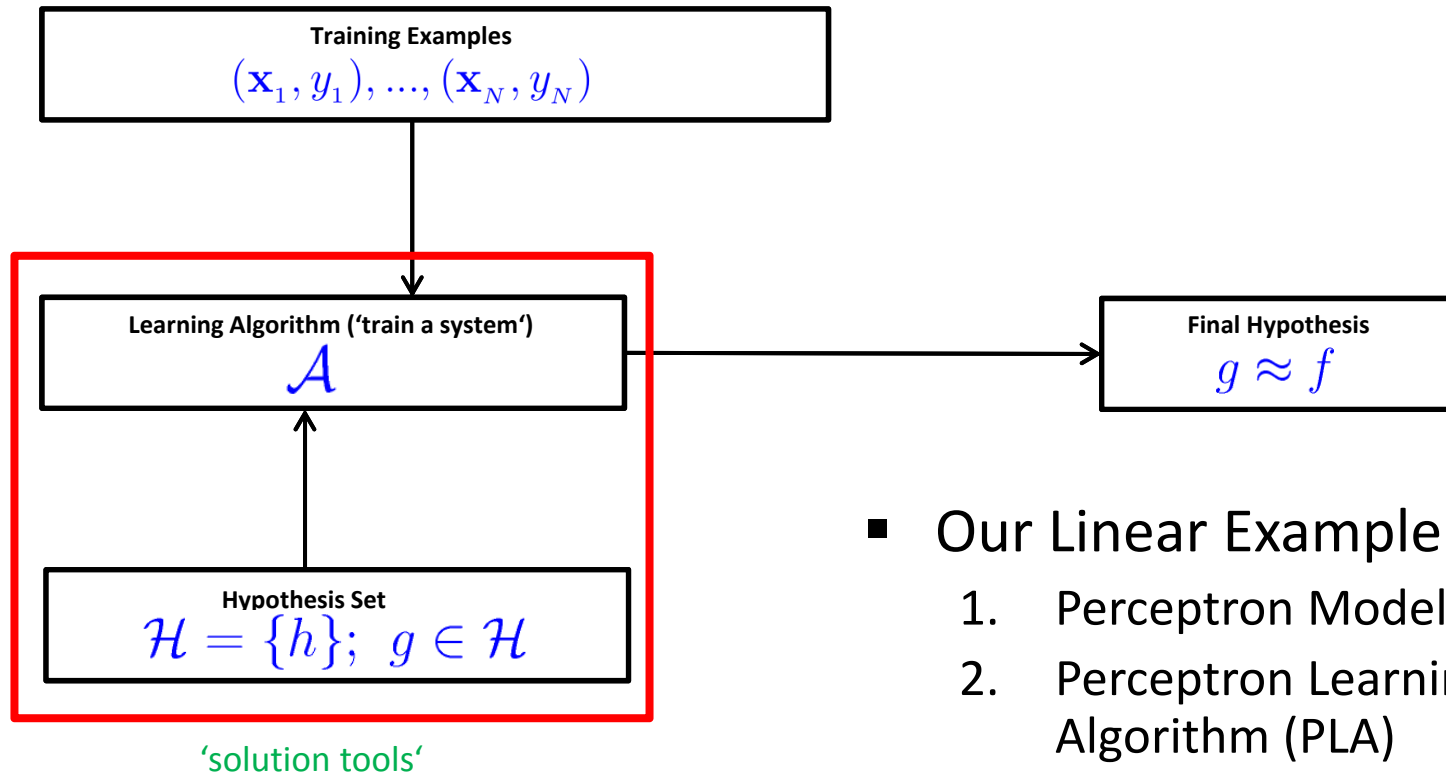
(Perceptron model – linear model)



(trained perceptron model and our selected final hypothesis)

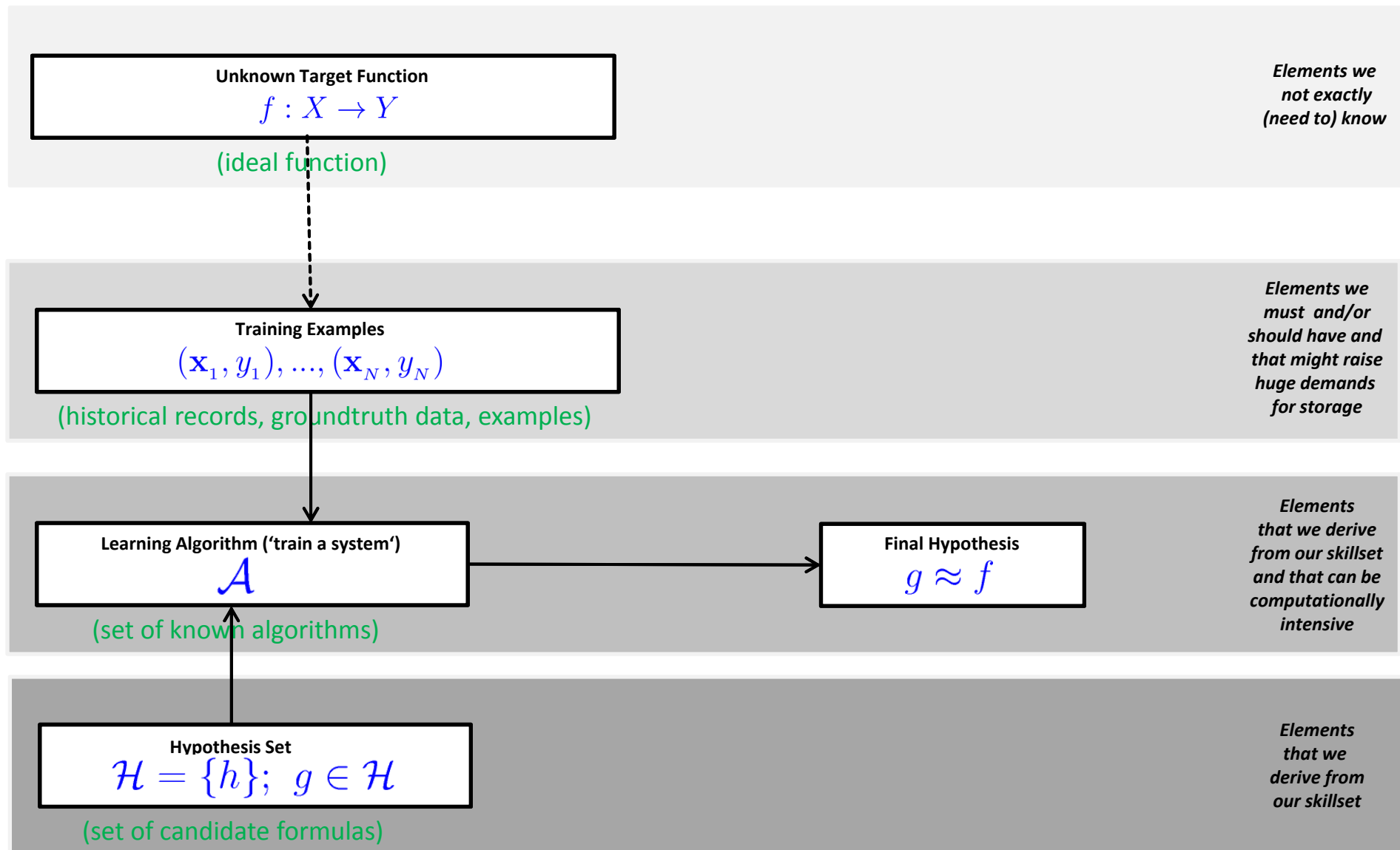
The Learning Model: Hypothesis Set & Learning Algorithm

- The solution tools – the **learning model**:
 1. **Hypothesis set \mathcal{H}** - a set of candidate formulas /models
 2. **Learning Algorithm \mathcal{A}** - ‘train a system’ with known algorithms



- Our Linear Example
 1. Perceptron Model
 2. Perceptron Learning Algorithm (PLA)

Mathematical Building Blocks (3)



Mathematical Building Blocks (3) – Our Linear Example

Unknown Target Function
 $f : X \rightarrow Y$

(ideal function)

Training Examples
 $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)$

(historical records, groundtruth data, examples)

Learning Algorithm ("train a system")
 \mathcal{A}

(Perceptron Learning Algorithm)

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

(Perceptron model – linear model)

Final Hypothesis
 $g \approx f$

	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

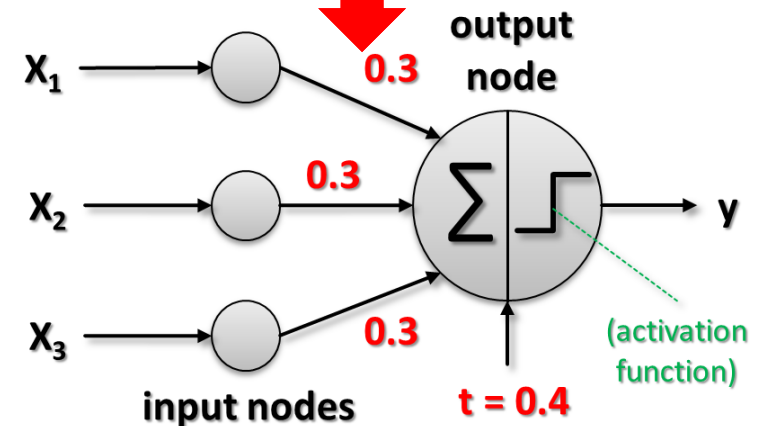
$(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)$

(training data)

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

(training phase;
Find w_i and threshold
that fit the data)

(algorithm uses
training dataset)



(trained perceptron model
and our selected final hypothesis)

Feasibility of Learning – Probability Distribution

- Predict output from future input
(fitting existing data is not enough)

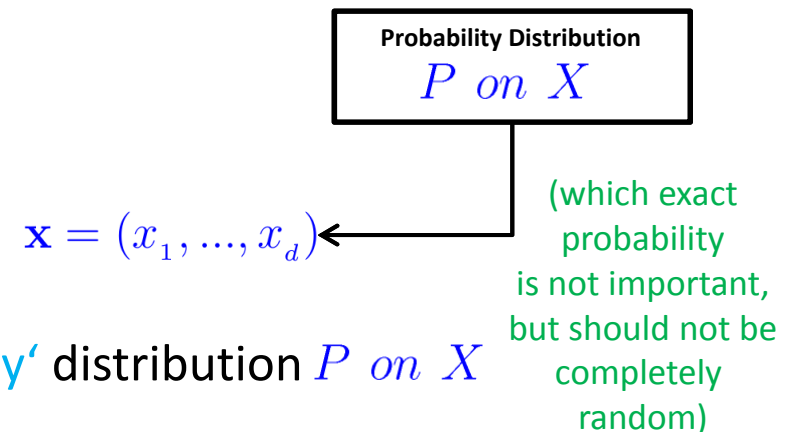
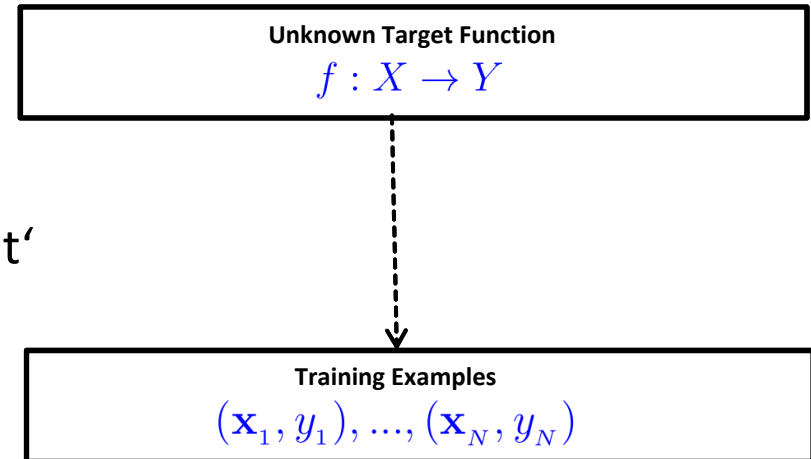
- In-sample '1000 points' fit well
- Possible: Out-of-sample \geq '1001 point' doesn't fit very well

- Learning 'any target function' is not feasible (can be anything)

- Assumptions about 'future input'

- Statement is possible to define about the data outside the in-sample data $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)$

- All samples (also future ones) are derived from same 'unknown probability' distribution P on X



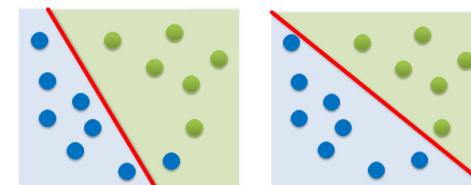
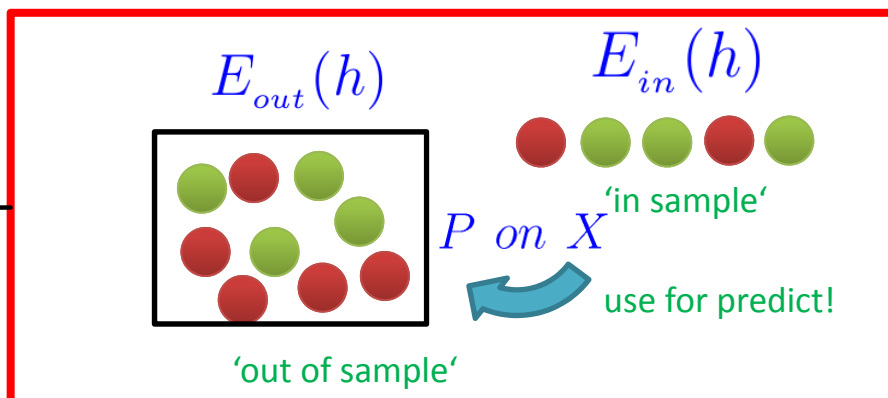
▪ Statistical Learning Theory assumes an unknown probability distribution over the input space X

Feasibility of Learning – In Sample vs. Out of Sample

- Given ‘unknown’ probability P on X
 - Given large sample N for $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)$
 - There is a probability of ‘picking one point or another’ (i.e. from statistics)
 - ‘Error on in sample’ is known quantity (using labelled data): $E_{in}(h)$
 - ‘Error on out of sample’ is unknown quantity: $E_{out}(h)$
 - In-sample frequency is likely close to out-of-sample frequency E_{in} tracks E_{out}

depend on
which
hypothesis h
out of m
different ones

$$\mathcal{H} = \{h_1, \dots, h_m\};$$



$$E_{in}(h) \approx E_{out}(h)$$

use $E_{in}(h)$ as a proxy – thus the other way around in learning

$$E_{out}(h) \approx E_{in}(h)$$

- Statistical Learning Theory part that enables that learning is feasible in a probabilistic sense (P on X)**

Feasibility of Learning – Union Bound & Factor **M**

- The union bound means that (for any countable set of m ‘events’) the probability that at least one of the events happens is not greater than the sum of the probabilities of the m individual ‘events’

- Assuming no overlaps in hypothesis set
 - Apply mathematical rule ‘union bound’ (i.e. poor bound)
 - Characterizes the number of data samples N needed

Final Hypothesis

$$g \approx f$$

Think if E_{in} deviates from E_{out} with more than tolerance ϵ it is a ‘bad event’ in order to apply union bound

$$\Pr [| E_{in}(g) - E_{out}(g) | > \epsilon] \leq \Pr [| E_{in}(h_1) - E_{out}(h_1) | > \epsilon$$

$$\text{or } | E_{in}(h_2) - E_{out}(h_2) | > \epsilon \dots$$

$$\text{or } | E_{in}(h_M) - E_{out}(h_M) | > \epsilon]$$

‘visiting **M**
different
hypothesis’

$$\Pr [| E_{in}(g) - E_{out}(g) | > \epsilon] \leq \sum_{m=1}^M \Pr [| E_{in}(h_m) - E_{out}(h_m) | > \epsilon]$$

sum of \Pr
is ‘worst case’
bound

$$\Pr [| E_{in}(g) - E_{out}(g) | > \epsilon] \leq \sum_{m=1}^M 2e^{-2\epsilon^2 N}$$

fixed quantity for each hypothesis
obtained from Hoeffdings Inequality

$$\Pr [| E_{in}(g) - E_{out}(g) | > \epsilon] \leq 2Me^{-2\epsilon^2 N}$$

problematic: if **M** is too big we loose the link
between the in-sample and out-of-sample

Feasibility of Learning – Modified Hoeffding's Inequality

- Errors in-sample $E_{in}(g)$ track errors out-of-sample $E_{out}(g)$
 - Statement is made being 'Probably Approximately Correct (PAC)'
 - Given M as number of hypothesis of hypothesis set \mathcal{H} [14] Valiant, 'A Theory of the Learnable', 1984
 - 'Tolerance parameter' in learning ϵ
 - Mathematically established via 'modified Hoeffdings Inequality':
(original Hoeffdings Inequality doesn't apply to multiple hypothesis)

$$\Pr \left[\left| E_{in}(g) - E_{out}(g) \right| > \epsilon \right] \leq 2Me^{-2\epsilon^2 N}$$

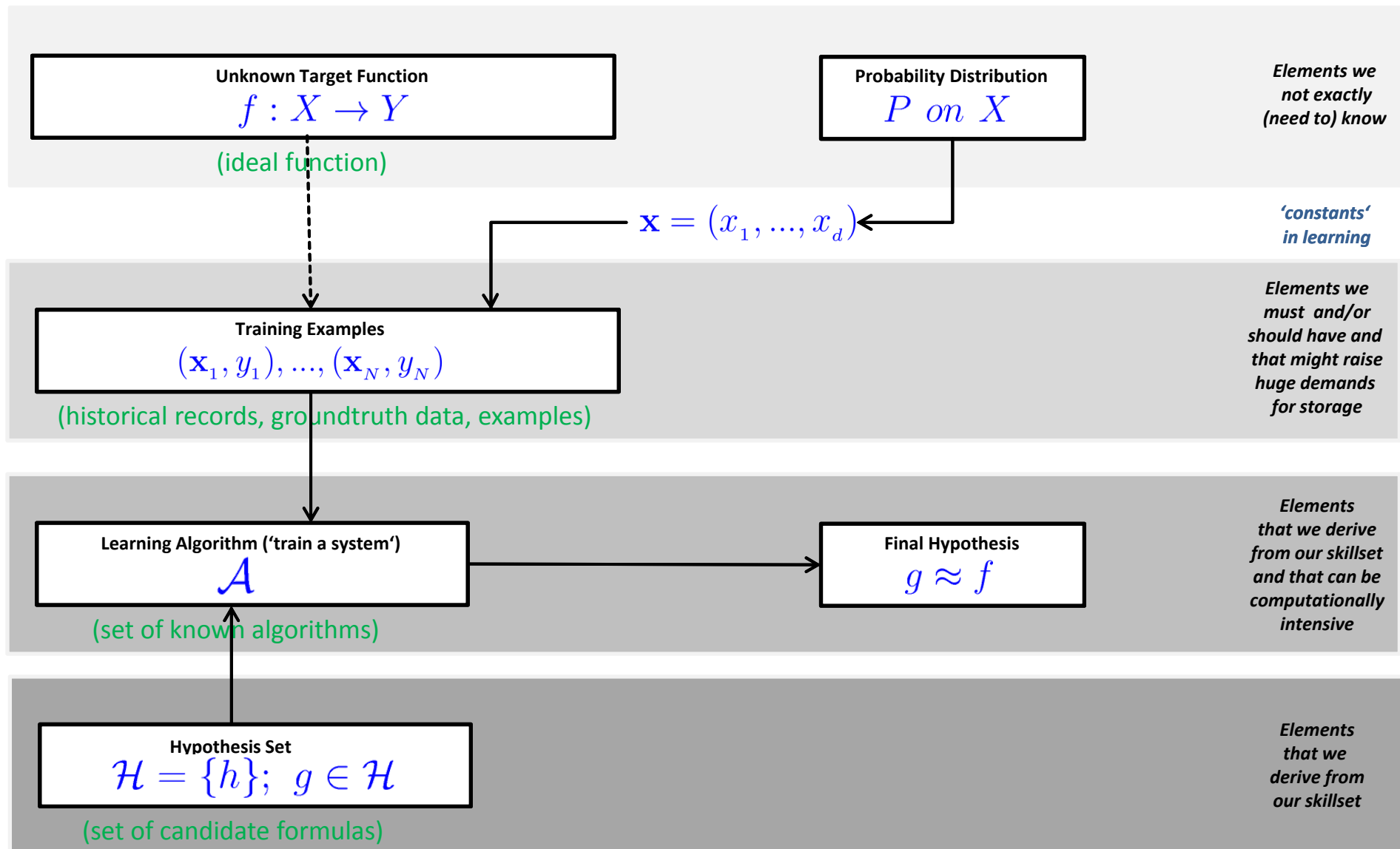
'Approximately'
'Probably'

'Probability that E_{in} deviates from E_{out} by more than the tolerance ϵ is a small quantity depending on M and N '

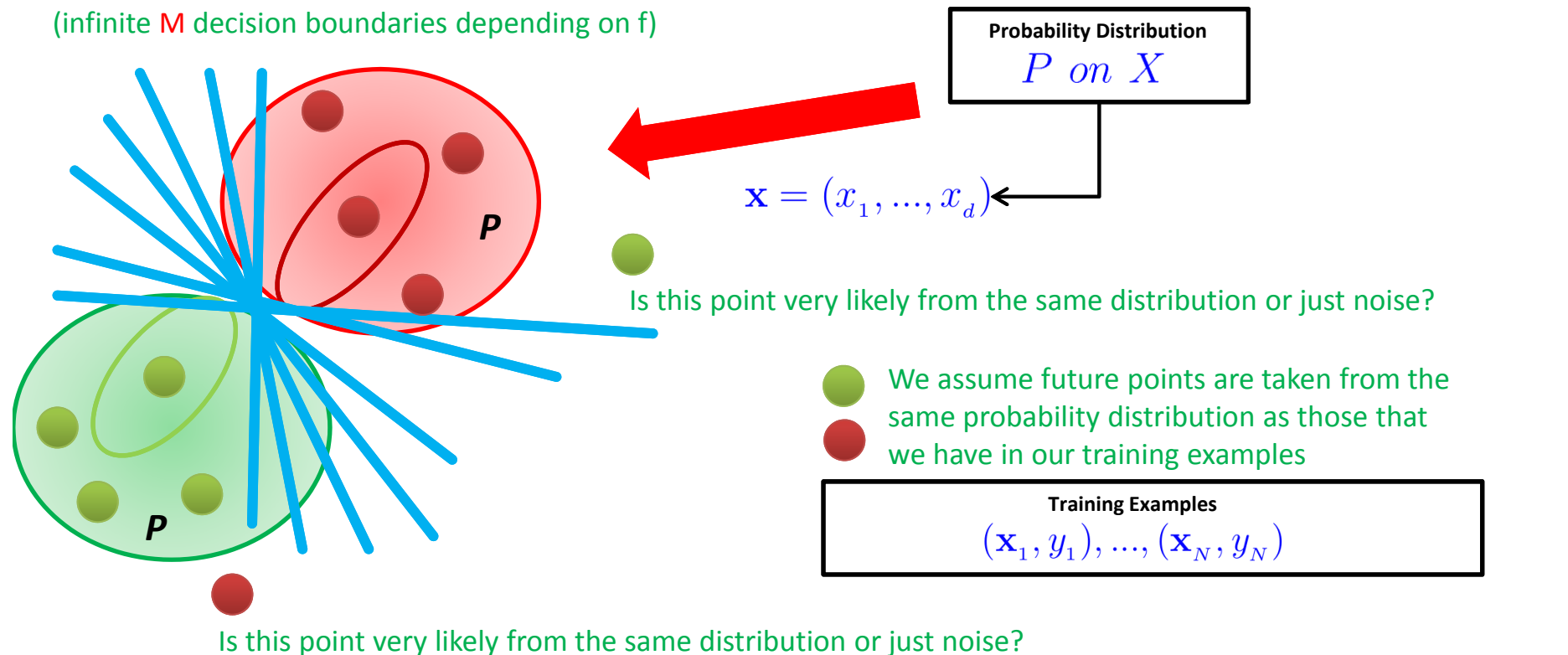
- Theoretical 'Big Data' Impact \rightarrow more $N \rightarrow$ better learning
 - The more samples N the more reliable will track $E_{in}(g) E_{out}(g)$ well
 - (But: the 'quality of samples' also matter, not only the number of samples)

Statistical Learning Theory part describing the Probably Approximately Correct (PAC) learning

Mathematical Building Blocks (4)



Mathematical Building Blocks (4) – Our Linear Example



(we help here with the assumption for the samples)

(we do not solve the M problem here)

$$\Pr \left[\left| E_{in}(g) - E_{out}(g) \right| > \epsilon \right] \leq 2Me^{-2\epsilon^2 N}$$

(counter example would be for instance a random number generator, impossible to learn this!)

Statistical Learning Theory – Error Measure & Noisy Targets

- Question: How can we learn a function from (noisy) data?
- ‘Error measures’ to quantify our progress, the goal is: $h \approx f$
 - Often user-defined, if not often ‘squared error’:

$$e(h(\mathbf{x}), f(\mathbf{x})) = (h(\mathbf{x}) - f(\mathbf{x}))^2$$

Error Measure

α

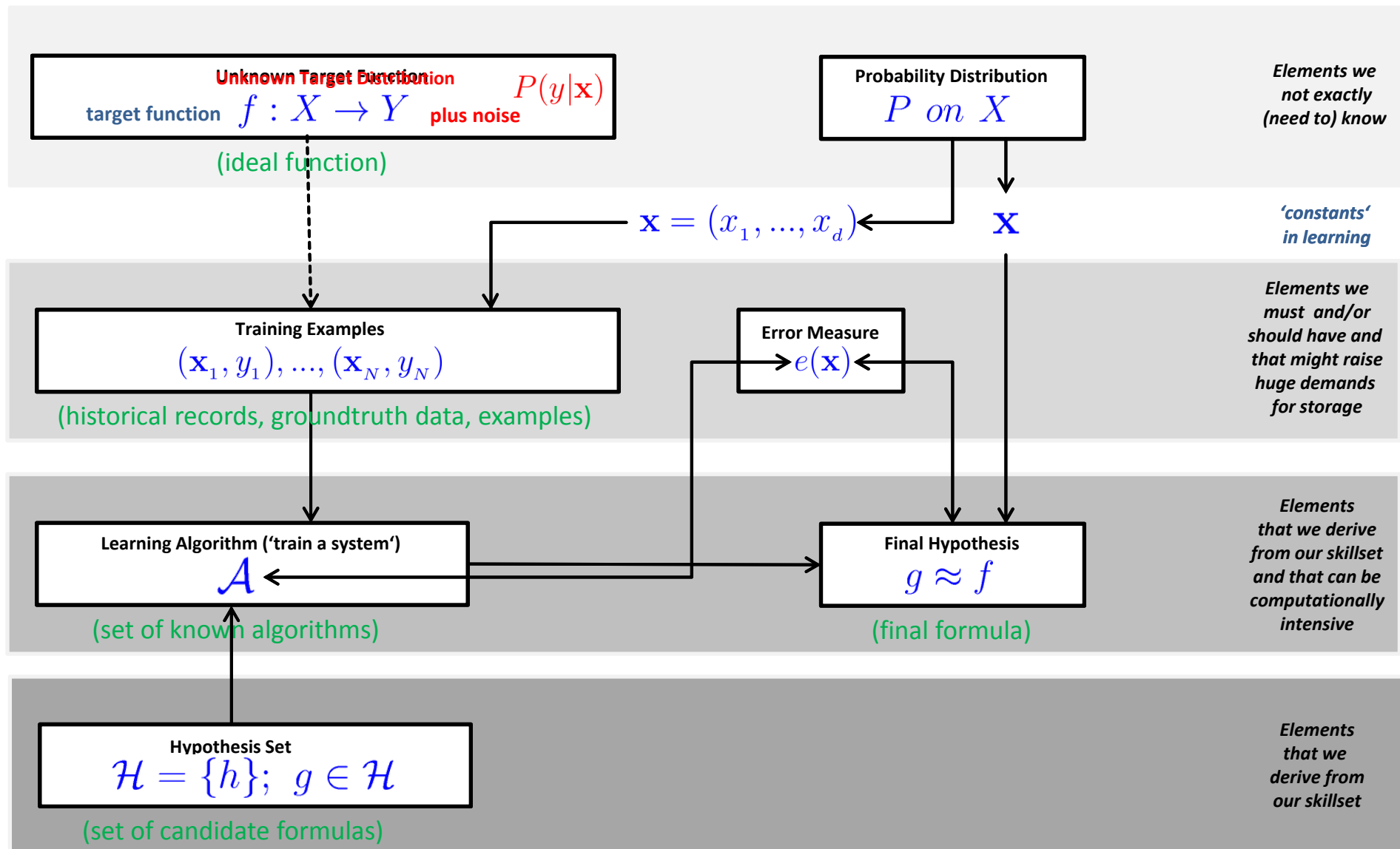
- E.g. ‘point-wise error measure’
(e.g. think movie rated now and in 10 years from now)
- ‘(Noisy) Target function’ is not a (deterministic) function
 - Getting with ‘same x in’ the ‘same y out’ is not always given in practice
 - Problem: ‘Noise’ in the data that hinders us from learning
 - Idea: Use a ‘target distribution’ instead of ‘target function’
 - E.g. credit approval (yes/no)

Unknown Target Distribution $P(y|\mathbf{x})$
target function $f : X \rightarrow Y$ plus noise

(ideal function)

- Statistical Learning Theory refines the learning problem of learning an unknown target distribution

Mathematical Building Blocks (5)



Mathematical Building Blocks (5) – Our Linear Example

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

(one point at a time is picked)

- Pick one misclassified training point where:

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

Error Measure α

- Update the weight vector:

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

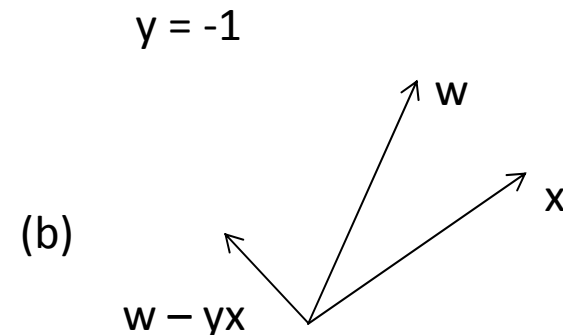
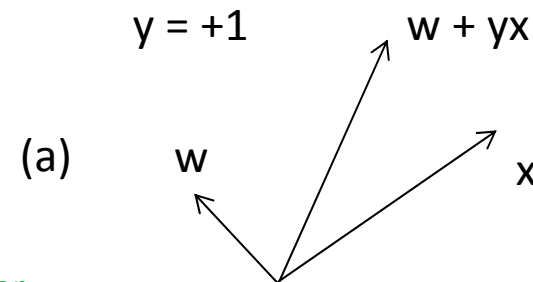
(y_n is either +1 or -1)

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

- Terminates when there are no misclassified points

(converges only with linearly separable data)

Error Measure α



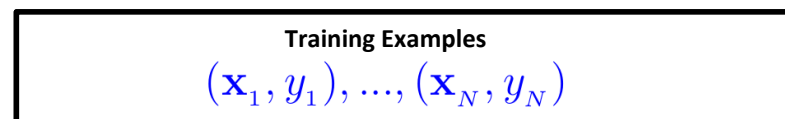
Training and Testing – Influence on Learning

- Mathematical notations

- Testing follows: $\Pr [| E_{in}(g) - E_{out}(g) | > \epsilon] \leq 2 e^{-2\epsilon^2 N}$
(hypothesis clear)
- Training follows: $\Pr [| E_{in}(g) - E_{out}(g) | > \epsilon] \leq 2Me^{-2\epsilon^2 N}$
(hypothesis search) (e.g. student exam training on examples to get E_{in} , 'down', then test via exam)

- Practice on 'training examples'

- Create two disjoint datasets
- One used for training only
(aka training set)
- Another used for testing only
(aka test set)



(historical records, groundtruth data, examples)

- Training & Testing are different phases in the learning process
 - Concrete number of samples in each set often influences learning

Theory of Generalization – Initial Generalization & Limits

- Learning is feasible in a probabilistic sense
 - Reported final hypothesis – using a ‘generalization window’ on $E_{out}(g)$
 - Expecting ‘out of sample performance’ tracks ‘in sample performance’
 - Approach: $E_{in}(g)$ acts as a ‘proxy’ for $E_{out}(g)$

$$E_{out}(g) \approx E_{in}(g)$$

This is not full learning – rather ‘good generalization’ since the quantity $E_{out}(g)$ is an unknown quantity

- Reasoning
 - Above condition is not the final hypothesis condition:
 - More similar like $E_{out}(g)$ approximates 0 (out of sample error is close to 0 if approximating f)
 - $E_{out}(g)$ measures how far away the value is from the ‘target function’
 - Problematic because $E_{out}(g)$ is an unknown quantity (cannot be used...)
 - The learning process thus requires ‘two general core building blocks’

Final Hypothesis

$$g \approx f$$

Theory of Generalization – Learning Process Reviewed

- ‘Learning Well’

- Two core building blocks that achieve $E_{out}(g)$ approximates 0

- First core building block

- Theoretical result using Hoeffdings Inequality $E_{out}(g) \approx E_{in}(g)$
- Using $E_{out}(g)$ directly is not possible – it is an unknown quantity

- Second core building block

(try to get the ‘in-sample’ error lower)

- Practical result using tools & techniques to get $E_{in}(g) \approx 0$
- e.g. linear models with the Perceptron Learning Algorithm (PLA)
- Using $E_{in}(g)$ is possible – it is a known quantity – ‘so lets get it small’
- Lessons learned from practice: in many situations ‘close to 0’ impossible
- E.g. remote sensing images use case of land cover classification

- Full learning means that we can make sure that $E_{out}(g)$ is close enough to $E_{in}(g)$ [from theory]
- Full learning means that we can make sure that $E_{in}(g)$ is small enough [from practical techniques]

Complexity of the Hypothesis Set – Infinite Spaces Problem

$$\Pr [| E_{in}(g) - E_{out}(g) | > \epsilon] \leq 2Me^{-2\epsilon^2 N}$$

theory helps to find a way to deal
with infinite M hypothesis spaces

- Tradeoff & Review
 - Tradeoff between ϵ , M , and the ‘complexity of the hypothesis space H ’
 - Contribution of detailed learning theory is to ‘understand factor M ’
- M Elements of the hypothesis set \mathcal{H} M elements in H here
 - Ok if N gets big, but problematic if M gets big \rightarrow bound gets meaningless
 - E.g. classification models like perceptron, support vector machines, etc.
 - **Challenge:** those classification models have continuous parameters
 - **Consequence:** those classification models have infinite hypothesis spaces
 - **Approach:** despite their size, the models still have limited expressive power

■ Many elements of the hypothesis set H have continuous parameter with infinite M hypothesis spaces

Factor **M** from the Union Bound & Hypothesis Overlaps

$$\Pr [| E_{in}(g) - E_{out}(g) | > \epsilon] \leq \Pr [| E_{in}(h_1) - E_{out}(h_1) | > \epsilon$$

assumes no overlaps, all probabilities happen disjointly

$$\text{or } | E_{in}(h_2) - E_{out}(h_2) | > \epsilon \dots$$

$$\text{or } | E_{in}(h_M) - E_{out}(h_M) | > \epsilon]$$

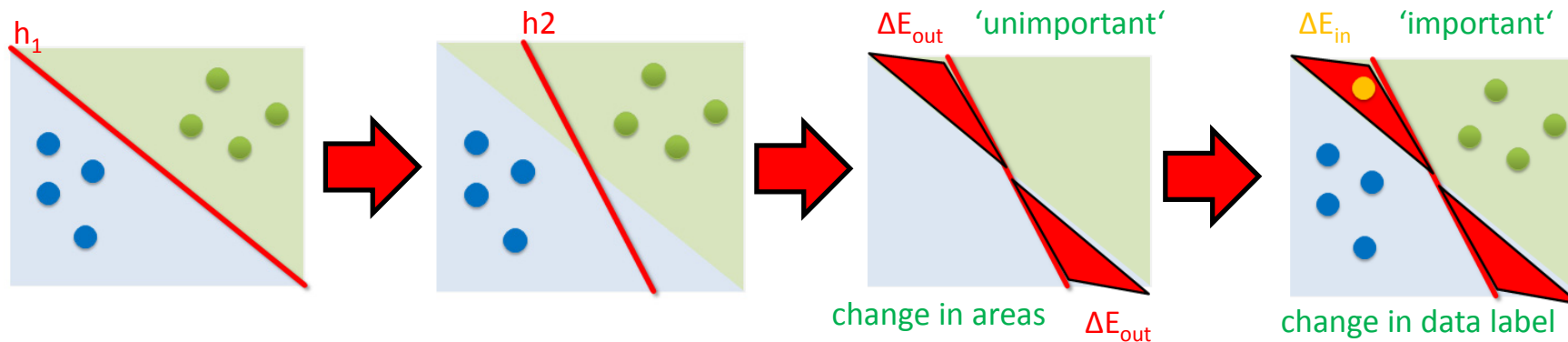
$$\Pr [| E_{in}(g) - E_{out}(g) | > \epsilon] \leq 2Me^{-2\epsilon^2 N}$$

takes no overlaps of **M** hypothesis into account

- Union bound is a ‘poor bound’, ignores correlation between h
 - Overlaps are common: the interest is shifted to data points changing label

$$| E_{in}(h_1) - E_{out}(h_1) | \approx | E_{in}(h_2) - E_{out}(h_2) |$$

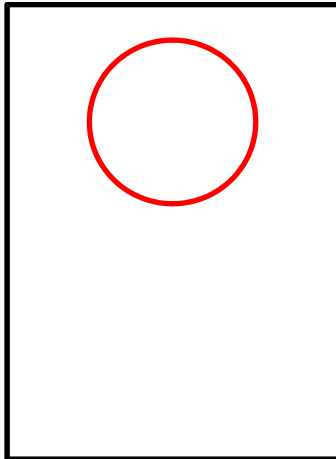
(at least very often, indicator to reduce **M**)



■ Statistical Learning Theory provides a quantity able to characterize the overlaps for a better bound

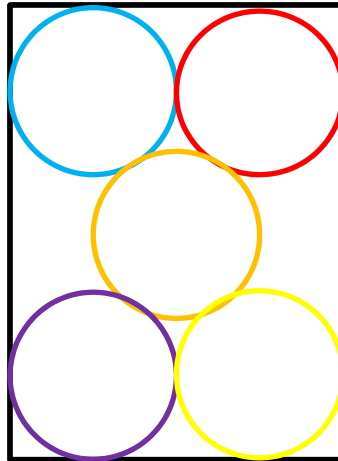
Replacing M & Large Overlaps

(Hoeffding Inequality)



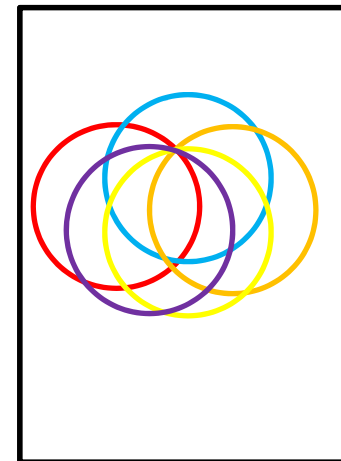
(valid for 1 hypothesis)

(Union Bound)



(valid for M hypothesis, worst case)

(towards Vapnik Chervonenkis Bound)



(valid for m(N) as growth function)

- Characterizing the overlaps is the idea of a ‘growth function’
 - Number of dichotomies: $m_{\mathcal{H}}(N) = \max_{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N} |\mathcal{H}(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N)|$
Number of hypothesis but on finite number N of points
 - Much redundancy: Many hypothesis will reports the same dichotomies

■ The mathematical proofs that $m_{\mathcal{H}}(N)$ can replace M is a key part of the theory of generalization

Complexity of the Hypothesis Set – VC Inequality

$$\Pr [| E_{in}(g) - E_{out}(g) | > \epsilon] \leq 2Me^{-2\epsilon^2 N}$$

$$m_{\mathcal{H}}(N) = \max_{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N} |\mathcal{H}(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N)|$$

■ Vapnik-Chervonenkis (VC) Inequality

- Result of mathematical proof when replacing M with growth function m
- $2N$ of growth function to have another sample ($2 \times E_{in}(h)$, no $E_{out}(h)$)

$$\Pr [| E_{in}(g) - E_{out}(g) | > \epsilon] \leq 4m_{\mathcal{H}}(2N)e^{-1/8\epsilon^2 N}$$

Important for bound:
 $m_h(N)$ is polynomial in N

(characterization of generalization)

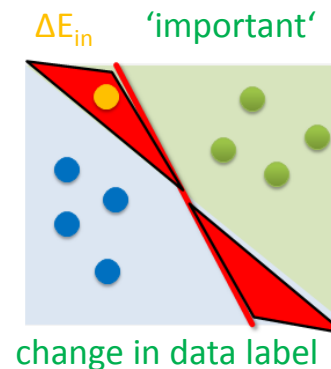
- In Short – finally : We are able to learn and can generalize ‘out-of-sample’

- The Vapnik-Chervonenkis Inequality is the most important result in machine learning theory
- The mathematical proof brings us that M can be replaced by growth function (no infinity anymore)
- The bound changes thus from ‘infinity with M ’ to a realistic bound that we can work with: $\max 2^N$

'Growth Function' – Perceptron Example

■ Dichotomies

- Hypothesis set separates data, but **only change important**
- Set of 'mini-hypothesis' is restricted to finite data points N
- Number of **mini-hypothesis = number of dichotomies**

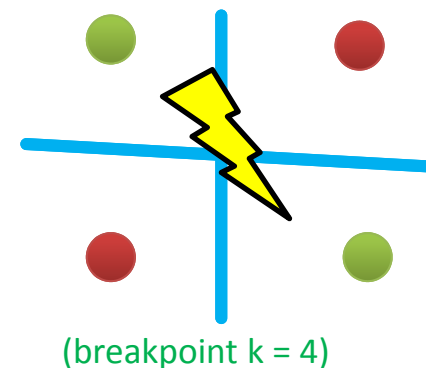


■ 'Growth Function'

- Based on the number of dichotomies (cardinality) $m_{\mathcal{H}}(N) = \max_{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N} |\mathcal{H}(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N)|$
- Pick $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N$ wisely to maximise the dichotomies (# at most 2^N)

■ 2D Perceptron

- Practice: **restriction on dichotomies means less mini-hypothesis possible** (less than 2^N)
- E.g. for $N = 4$ points, there is always a pattern that can not be realized

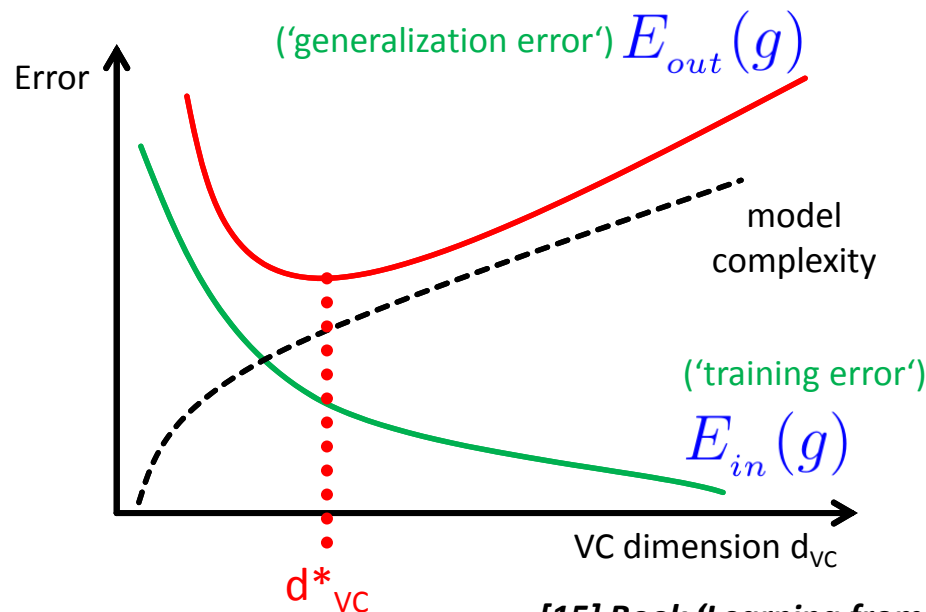


[15] Book 'Learning from Data'

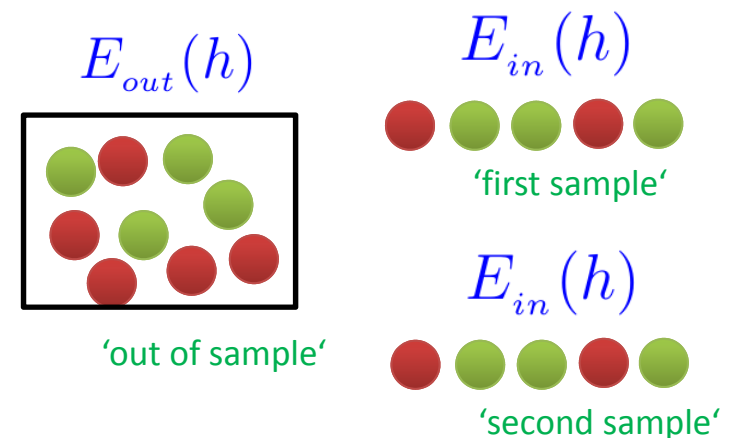
Towards Complexity of the Hypothesis Set – VC Dimension

- Vapnik-Chervonenkis (VC) Dimension over instance space X
 - VC dimension gets a ‘generalization bound’ on all possible target functions
 - Practice: think how much model parameters (‘degrees of freedom’)

Issue: unknown to ‘compute’ – VC solved this using the growth function on different samples



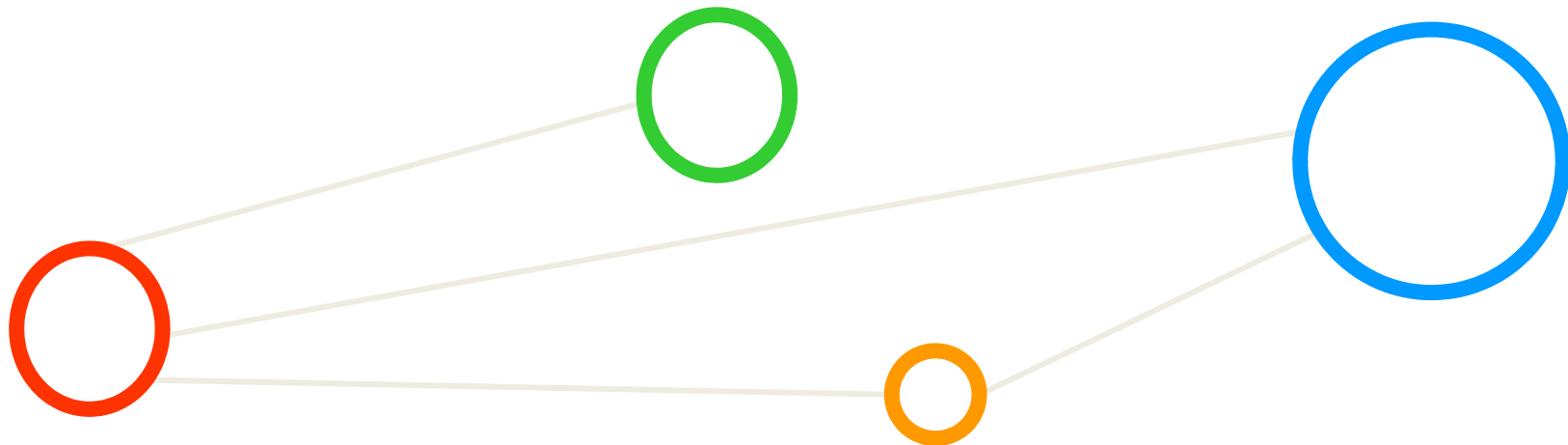
[15] Book ‘Learning from Data’



idea: ‘first sample’ frequency close to ‘second sample’ frequency

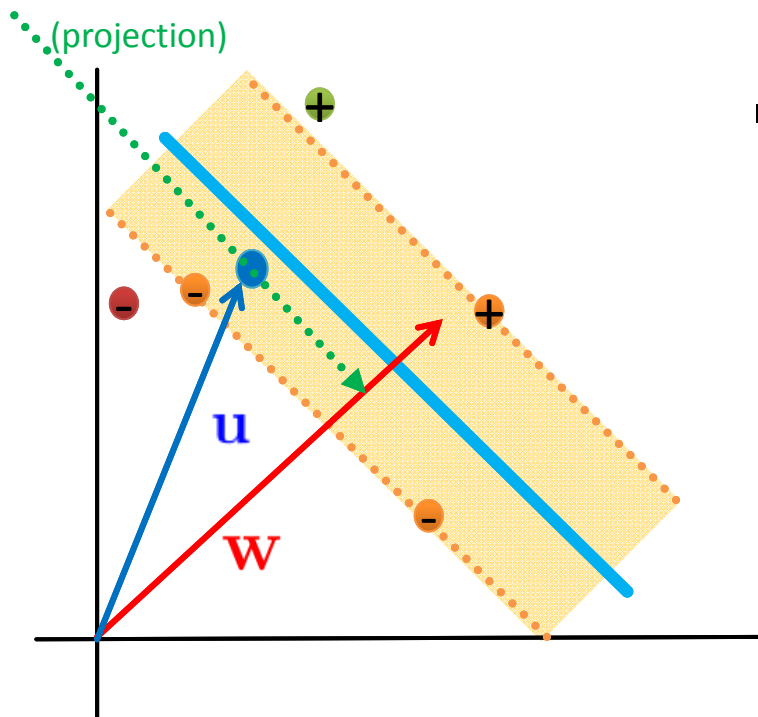
- Complexity of Hypothesis set H can be measured by the Vapnik-Chervonenkis (VC) Dimension d_{VC}
- Ignoring the model complexity d_{VC} leads to situations where $E_{in}(g)$ gets down and $E_{out}(g)$ gets up

Appendix C: Geometric Interpretation of SVMs & Kernels



Geometric SVM Interpretation and Setup (1)

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

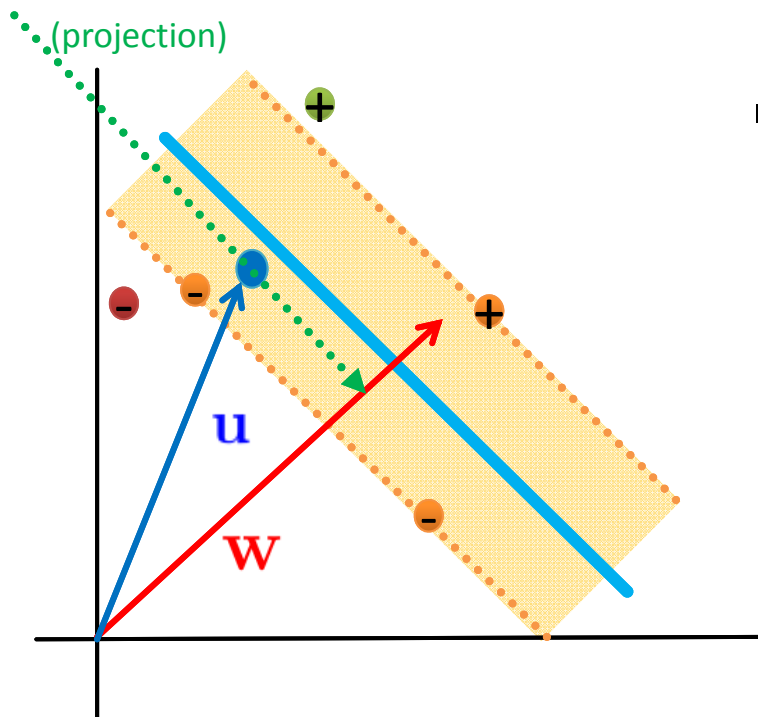


- Dot product $\mathbf{w} \cdot \mathbf{u} \geq C; C = -b$
 - With \mathbf{u} takes the projection on the \mathbf{w}
 - Depending on where projection is it is left or right from the decision boundary
 - Simple transformation brings decision rule:
- ① $\mathbf{w} \cdot \mathbf{u} + b \geq 0 \rightarrow$ means $+$
- (given that b and \mathbf{w} are unknown to us)

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

Geometric SVM Interpretation and Setup (2)

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

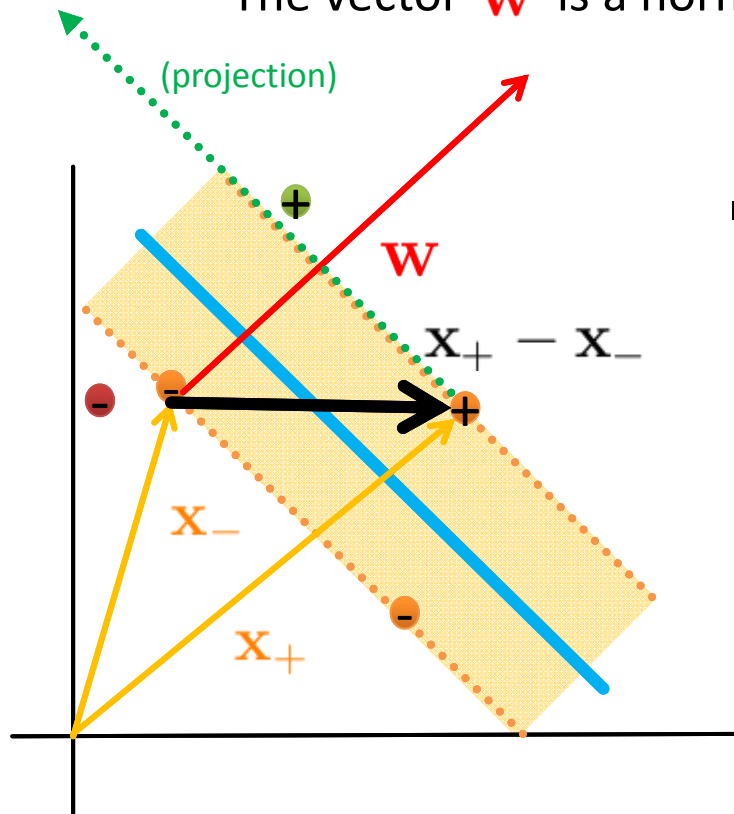


- Multiply equations by y_i
 - Positive samples: $y_i(\mathbf{x}_i \cdot \mathbf{w} + b) \geq 1$
 - Negative samples: $y_i(\mathbf{x}_i \cdot \mathbf{w} + b) \leq -1$
 - Both **same** due to $y_i = +$ and $y_i = -$
(brings us mathematical convenience often quoted)
$$y_i(\mathbf{x}_i \cdot \mathbf{w} + b) - 1 \geq 0$$

(additional constraints just for support vectors itself helps)
- ② $y_i(\mathbf{x}_i \cdot \mathbf{w} + b) - 1 = 0$

Geometric SVM Interpretation and Setup (3)

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



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

- Unit vector is helpful for ‘margin width’

- Projection (dot product) for margin width:

$$\mathbf{x}_+ - \mathbf{x}_- \cdot \frac{\mathbf{w}}{\|\mathbf{w}\|} \quad (\text{unit vector})$$

$\downarrow \quad \downarrow$

$$1 - b \quad 1 + b \quad \rightarrow \quad \frac{2}{\|\mathbf{w}\|} \quad \textcircled{3}$$

- When enforce constraint: $y_i = + \text{ (green dot) }$

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

Constrained Optimization Steps SVM (1)

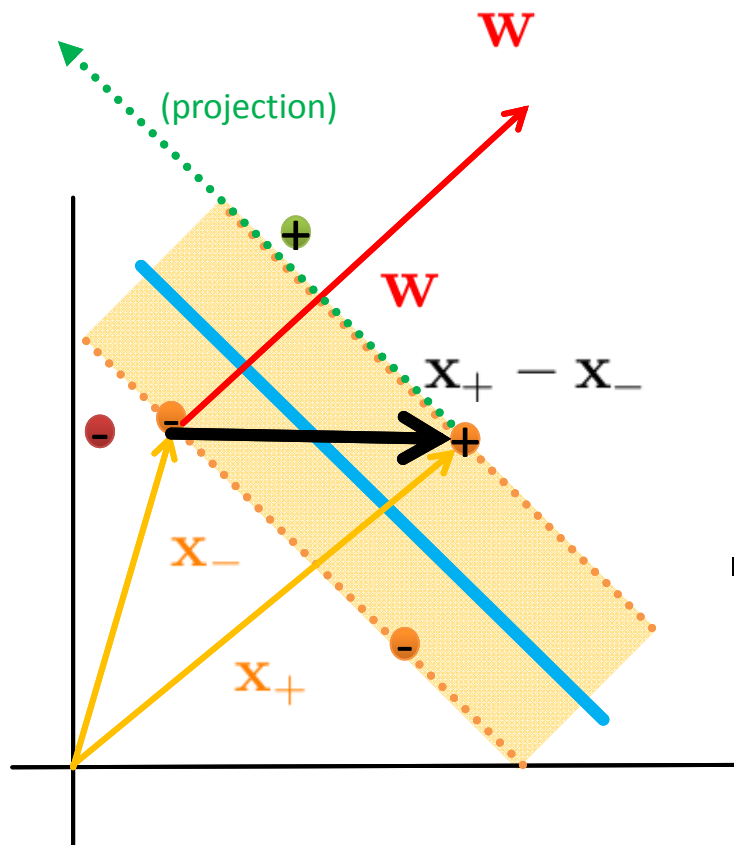
- Use 'constraint optimization' of mathematical toolkit

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

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

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

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



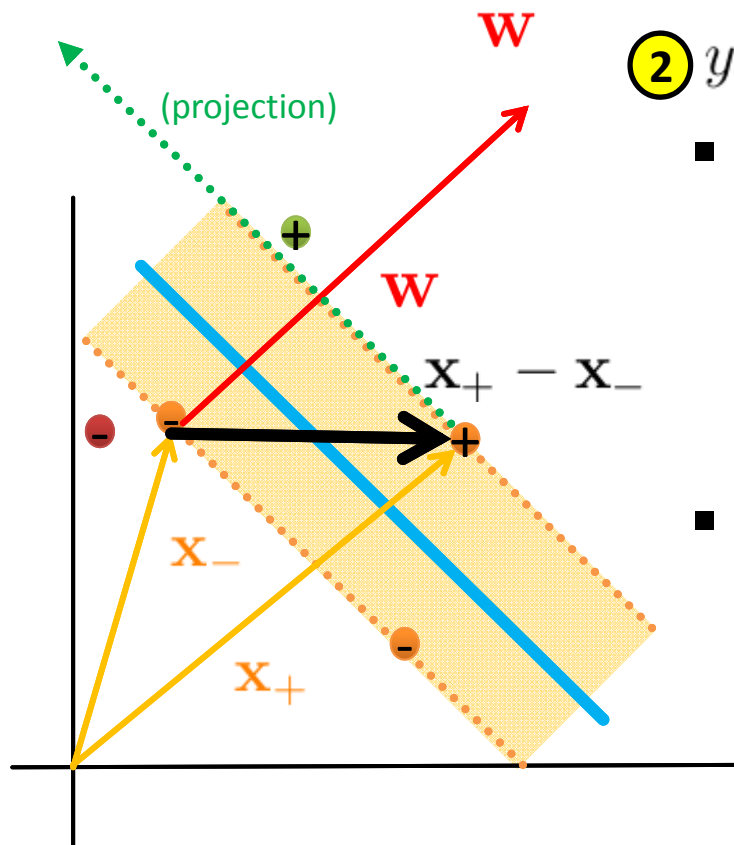
- Next: Find the extreme values

- Subject to constraints

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

Constrained Optimization Steps SVM (2)

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



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

- Introduce a multiplier for each constraint

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



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

- Find derivatives for extremum & set 0

- But two unknowns that might vary
- First differentiate w.r.t. \mathbf{w}
- Second differentiate w.r.t. b

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

Constrained Optimization Steps SVM (3)

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

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

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

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

- Simple transformation brings:

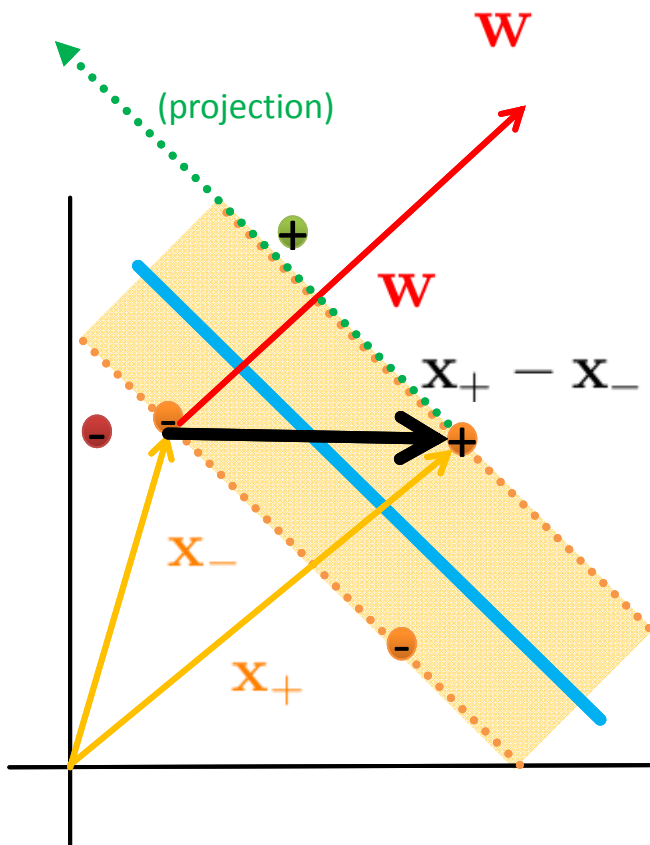
$$\textcircled{5} \mathbf{w} = \sum \alpha_i y_i \mathbf{x}_i$$

(i.e. vector is linear sum of samples)

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

- Second differentiate w.r.t. b

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



Constrained Optimization Steps SVM (4)

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

- Find minimum

- Quadratic optimization problem

- Take advantage of **5** $\mathbf{w} = \sum \alpha_i y_i \mathbf{x}_i$

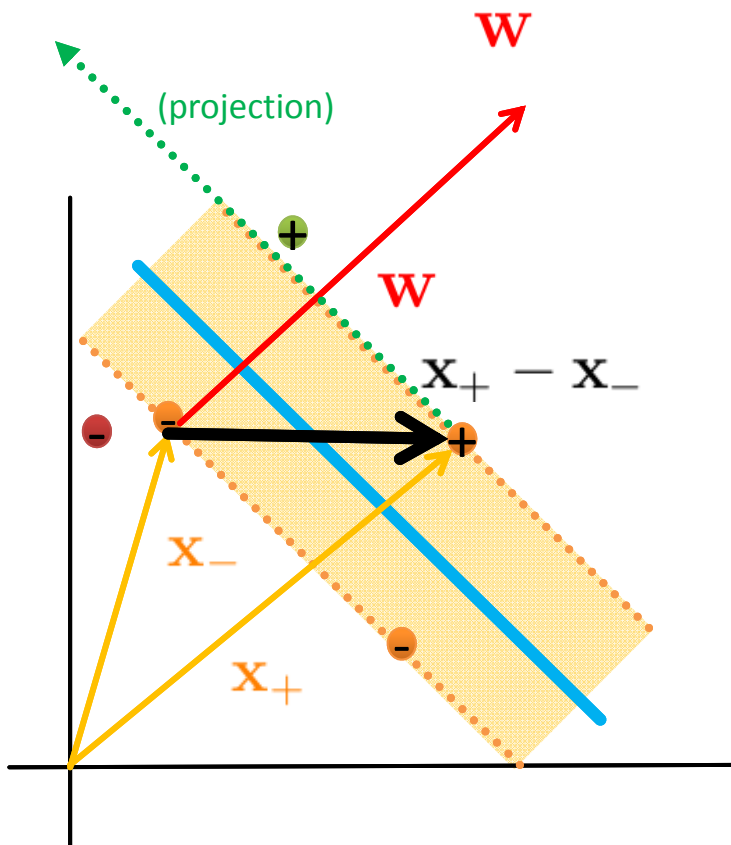
$$\mathcal{L} = \frac{1}{2} \left(\sum \alpha_i y_i \mathbf{x}_i \right) \cdot \left(\sum \alpha_j y_j \mathbf{x}_j \right)$$

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

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

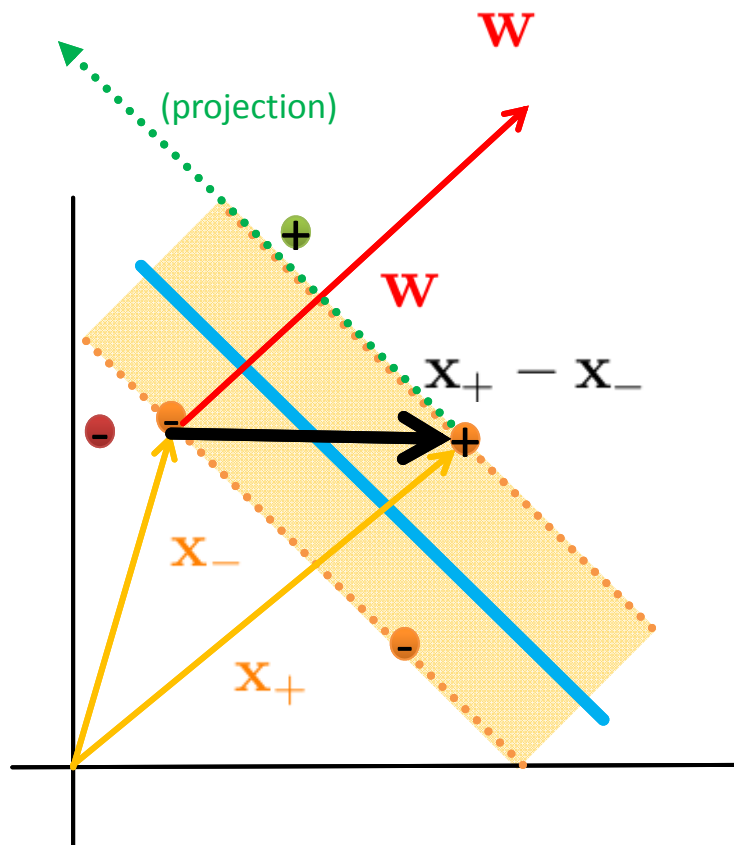
(b constant in front sum)

$$\mathbf{5} \sum \alpha_i y_i = 0$$



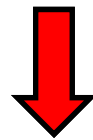
Constrained Optimization Steps SVM (5)

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



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

(was 0)



(results in)

(optimization depends only on dot product of samples)

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

- Equation to be solved by some quadratic programming package

Use of SVM Classifier to Perform Classification

- Use findings for decision rule

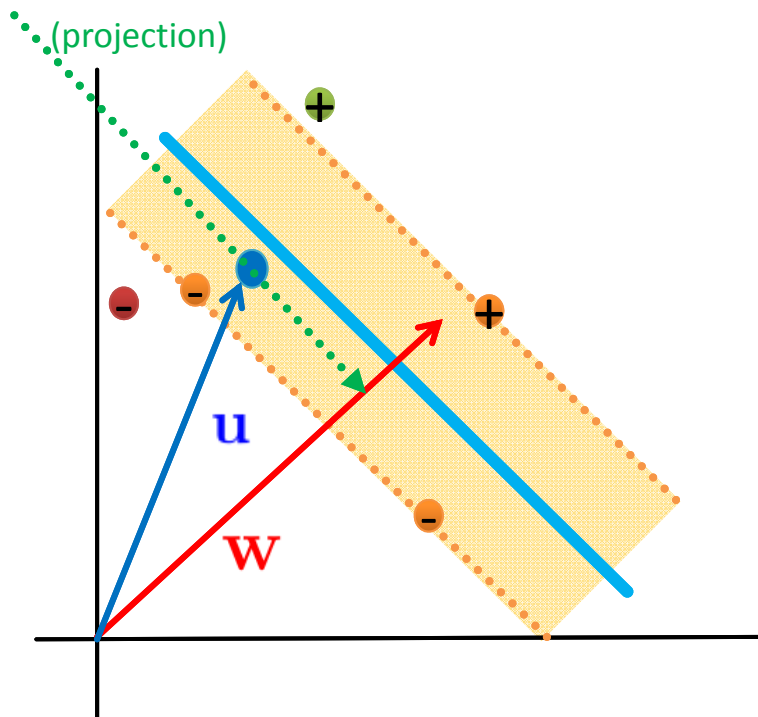
$$\textcircled{5} \mathbf{w} = \sum \alpha_i y_i \mathbf{x}_i$$

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



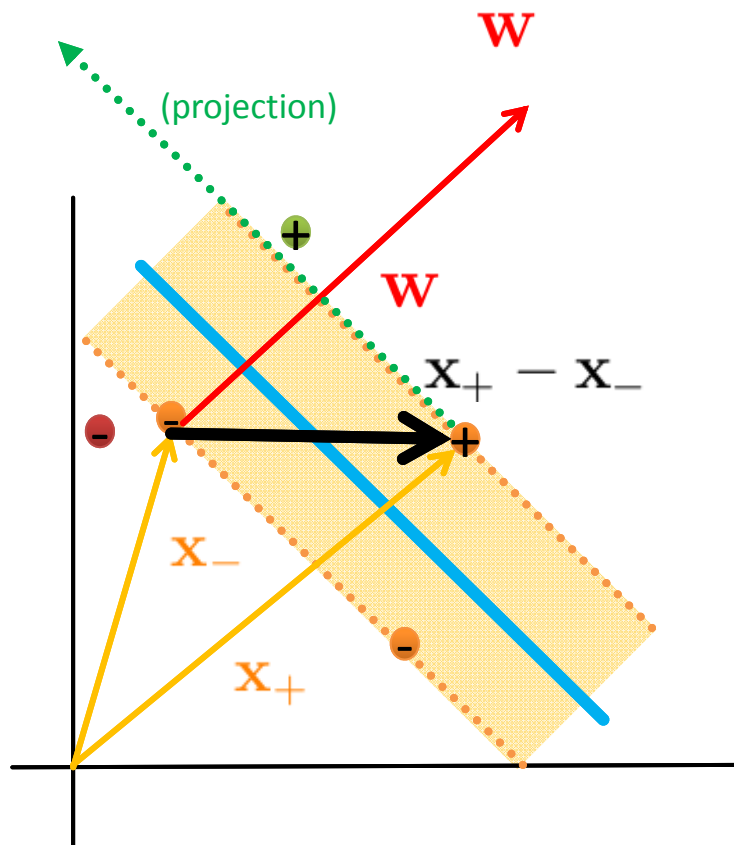
$$\sum \alpha_i y_i \mathbf{x}_i \cdot \mathbf{u}_i + b \geq 0 \quad +$$

(decision rule also depends on dotproduct)



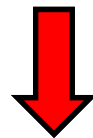
Constrained Optimization Steps SVM & Dot Product

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



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

(was 0)



(results in)

(optimization depends only on dot product of samples)

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

- Equation to be solved by some quadratic programming package

Kernel Methods & Dot Product Dependency

- Use findings for **decision rule**

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



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



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

(decision rule also depends on dotproduct)

- Dotproduct** enables nice more elements

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

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

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

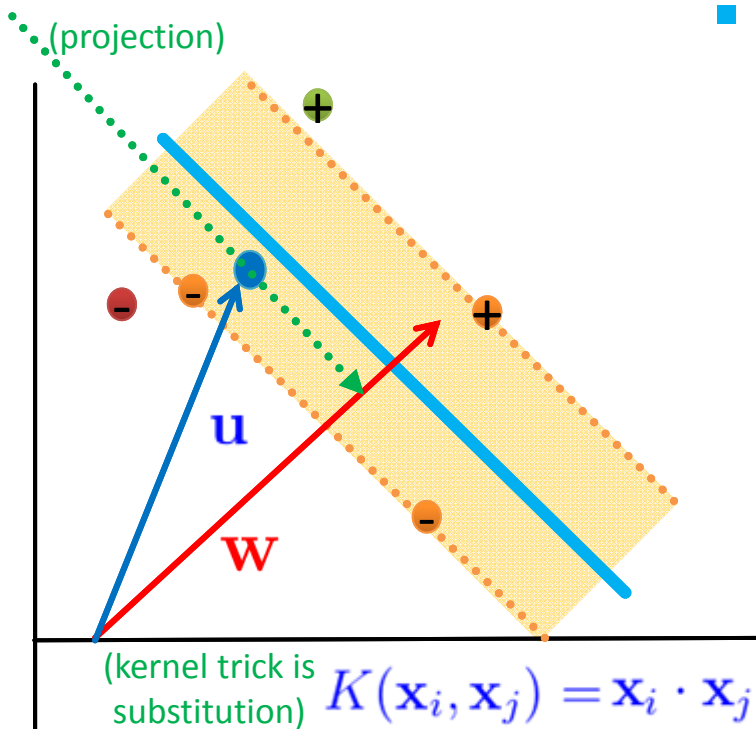
(optimization depends only on dot product of samples)

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

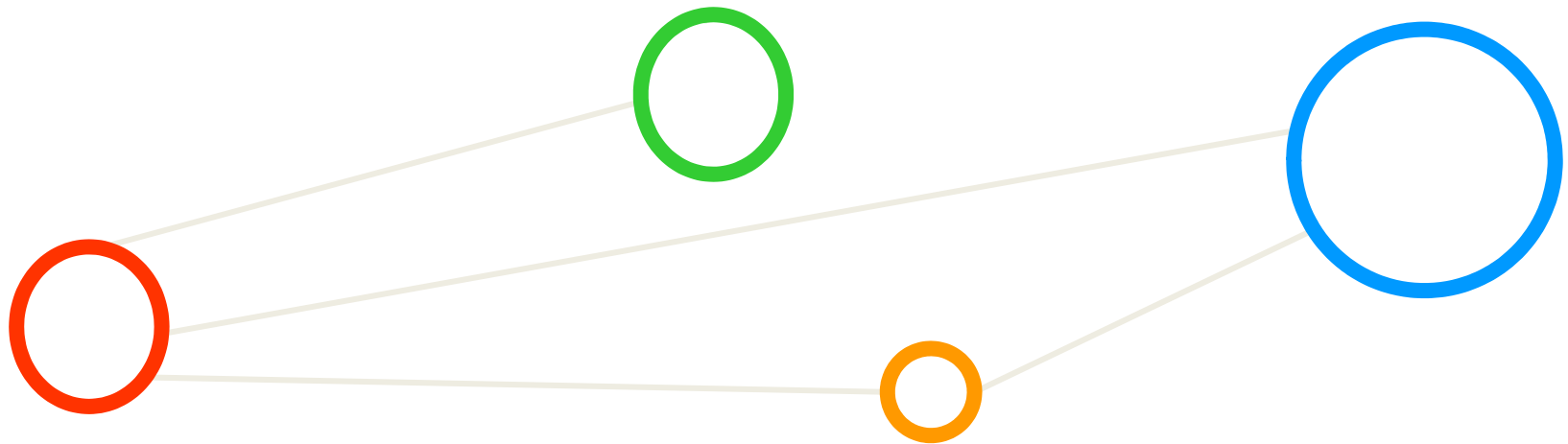
(kernel trick is substitution)

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

(trusted Kernel avoids to know Phi)

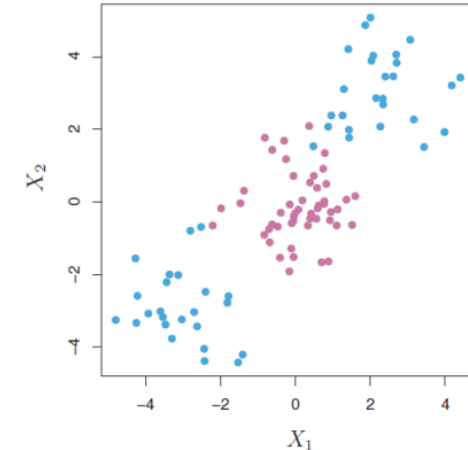


Appendix D: Kernel Methods



Need for Non-linear Decision Boundaries

- Lessons learned from practice
 - Scientists and engineers are often faced with **non-linear class boundaries**
- Non-linear transformations approach
 - **Enlarge feature space (computationally intensive)**
 - Use **quadratic, cubic, or higher-order polynomial** functions of the predictors
- Example with Support Vector Classifier



(time invest: mapping done by explicitly carrying out the map into the feature space)

X_1, X_2, \dots, X_p (previously used p features)

$X_1, X_1^2, X_2, X_2^2, \dots, X_p, X_p^2$ (new $2p$ features)

(decision boundary is linear in the enlarged feature space)

(decision boundary is non-linear in the original feature space with $q(x) = 0$ where q is a quadratic polynomial)

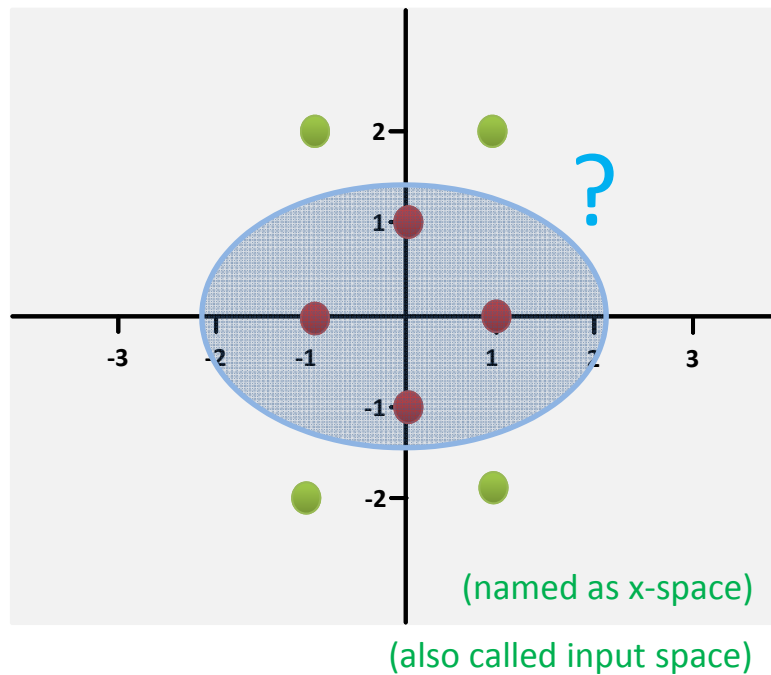
$$\begin{aligned}
 & \underset{\beta_0, \beta_{11}, \beta_{12}, \dots, \beta_{p1}, \beta_{p2}, \epsilon_1, \dots, \epsilon_n}{\text{maximize}} && M \\
 & \text{subject to } y_i \left(\beta_0 + \sum_{j=1}^p \beta_{j1} x_{ij} + \sum_{j=1}^p \beta_{j2} x_{ij}^2 \right) \geq M(1 - \epsilon_i) \\
 & \sum_{i=1}^n \epsilon_i \leq C, \quad \epsilon_i \geq 0, \quad \sum_{j=1}^p \sum_{k=1}^2 \beta_{jk}^2 = 1
 \end{aligned}$$

[6] An Introduction to Statistical Learning

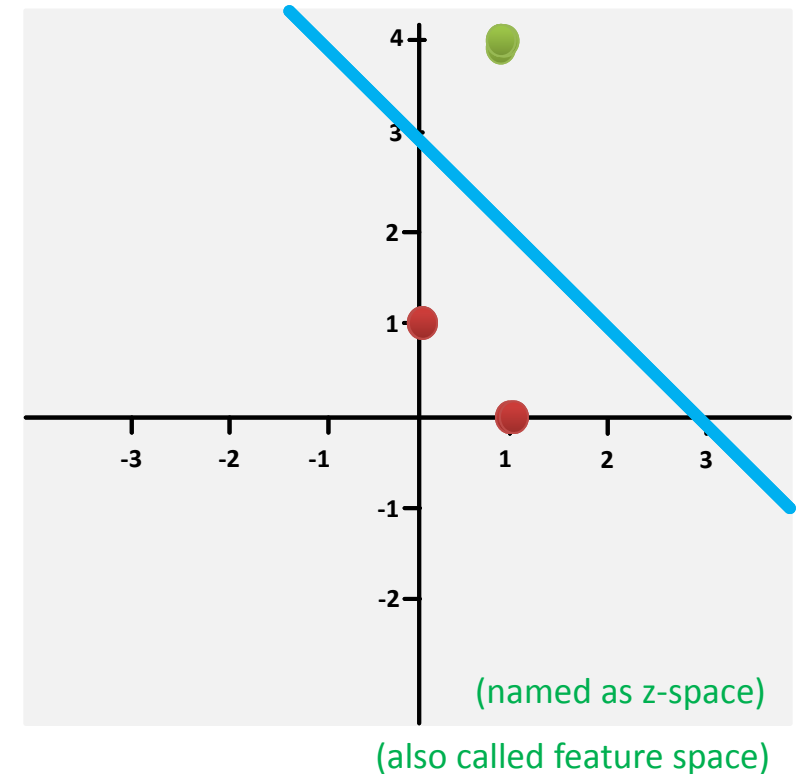
Understanding Non-Linear Transformations (1)

- Example: 'Use measure of distances from the origin/centre'
- Classification
 - (1) new point; (2) transform to z-space; (3) classify it with e.g. perceptron

(still linear models applicable)

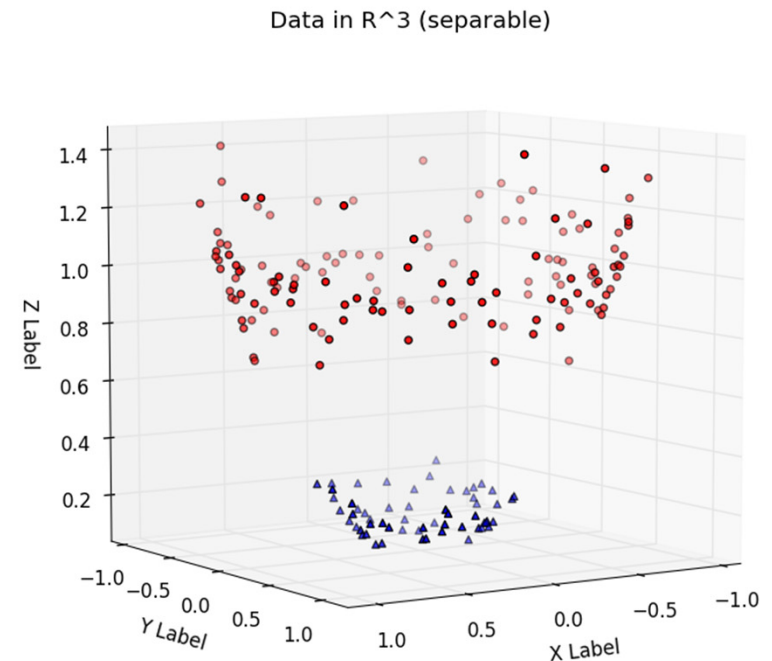
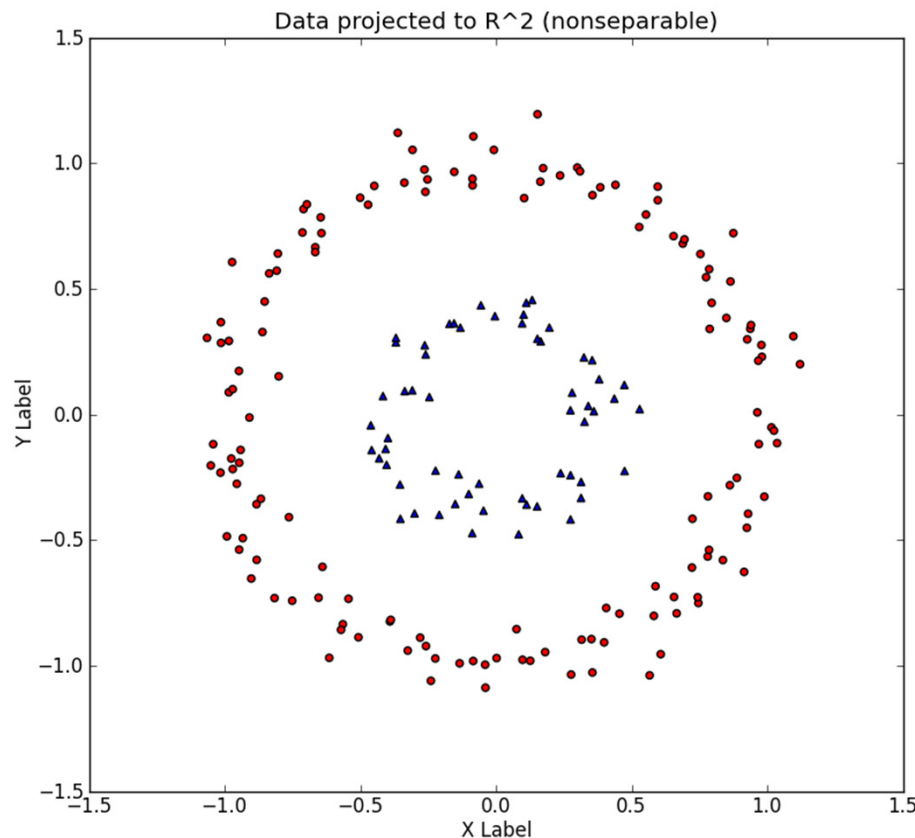


Φ
→
(‘changing constants’)



Understanding Non-Linear Transformations (2)

- Example: From 2 dimensional to 3 dimensional: $[x_1, x_2] = [x_1, x_2, x_1^2 + x_2^2]$
 - Much higher dimensional can cause memory and computing problems

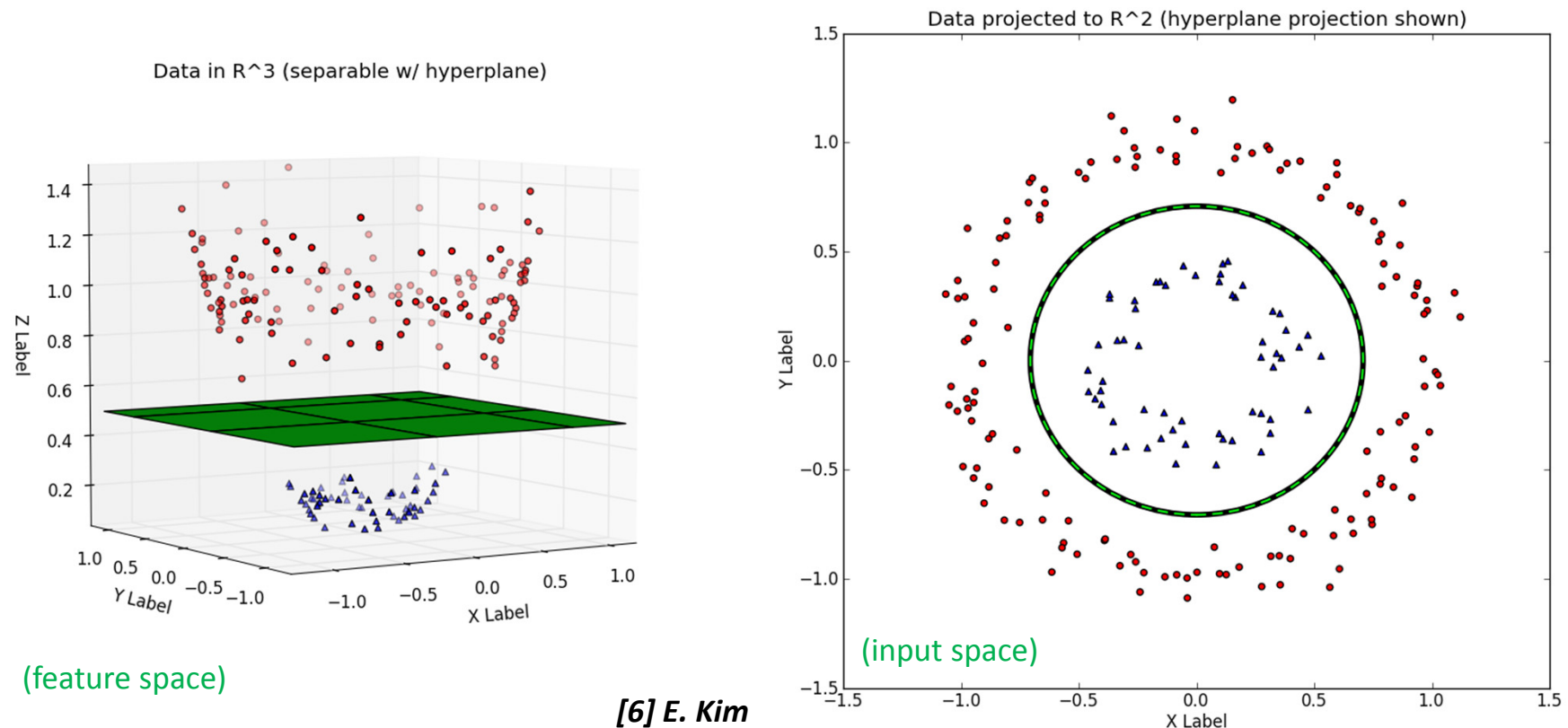


[20] E. Kim

- **Problems: Not clear which type of mapping (search); optimization is computationally expensive task**

Understanding Non-linear Transformations (3)

- Example: From 2 dimensional to 3 dimensional: $[x_1, x_2] = [x_1, x_2, x_1^2 + x_2^2]$
 - Separating hyperplane can be found and ‘mapped back’ to input space



- **Problem: ‘curse of dimensionality’ – As dimensionality increases & volume of space too: sparse data!**

Term Support Vector Machines – Revisited

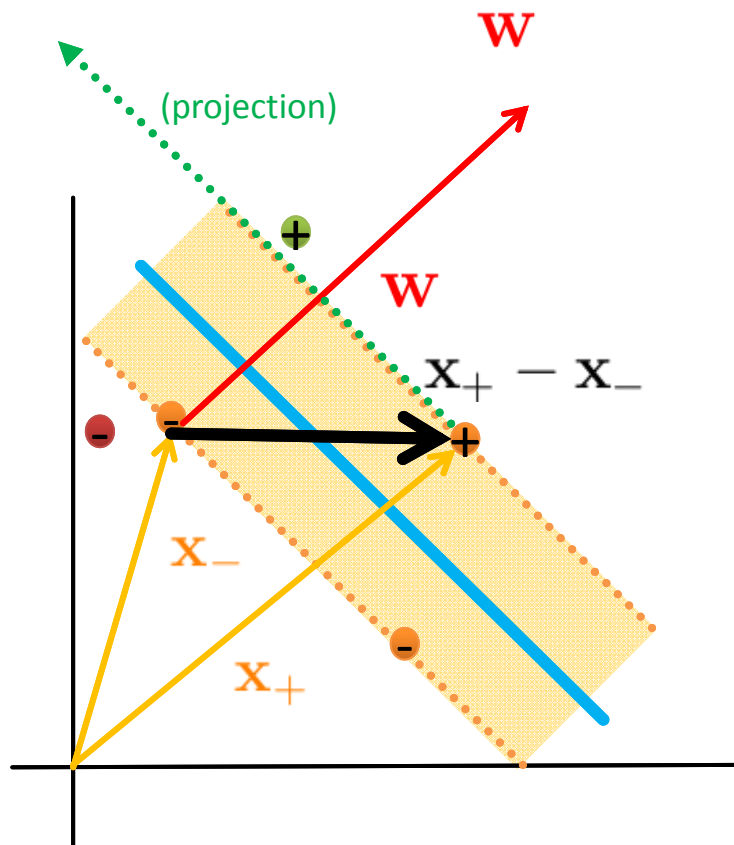
- Support Vector Machines (SVMs) are a classification technique developed ~1990
- SVMs perform well in many settings & are considered as one of the best 'out of the box classifiers'

[1] *An Introduction to Statistical Learning*

- Term detailed refinement into **'three separate techniques'**
 - Practice: applications mostly use the SVMs with kernel methods
- **'Maximal margin classifier'**
 - A simple and intuitive classifier with a 'best' linear class boundary
 - Requires that data is **'linearly separable'**
- **'Support Vector Classifier'**
 - Extension to the maximal margin classifier for non-linearly separable data
 - Applied to a broader range of cases, idea of **'allowing some error'**
- **'Support Vector Machines' → Using Non-Linear Kernel Methods**
 - Extension of the support vector classifier
 - Enables non-linear class boundaries & via **kernels**;

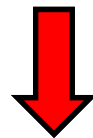
Constrained Optimization Steps SVM & Dot Product

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



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

(was 0)



(results in)

(optimization depends only on dot product of samples)

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

- Equation to be solved by some quadratic programming package

Kernel Methods & Dot Product Dependency

- Use findings for **decision rule**

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



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



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

(decision rule also depends on dotproduct)

- Dotproduct** enables nice more elements

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

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

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

(optimization depends only on dot product of samples)

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

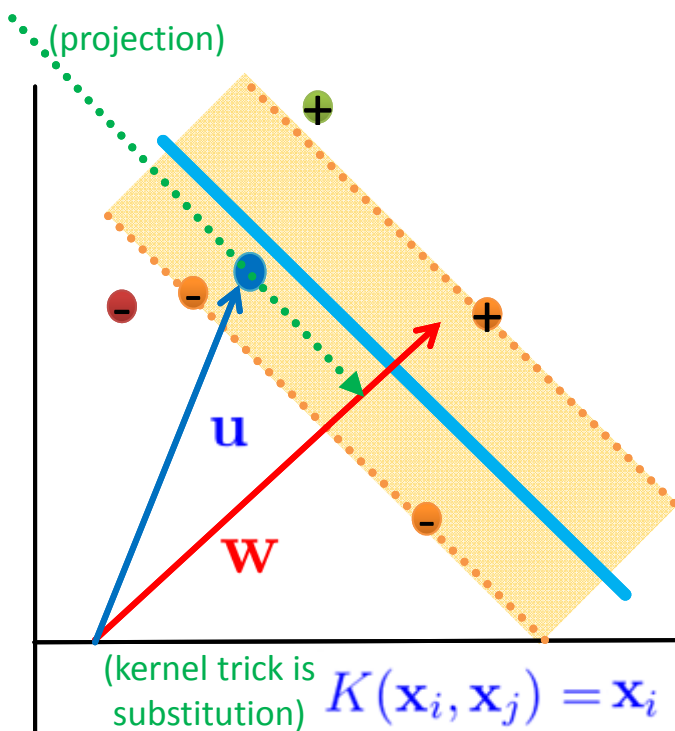
(kernel trick is substitution)

$$K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i \cdot \mathbf{x}_j$$

$\textcircled{7}$

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

(trusted Kernel avoids to know Phi)



Support Vector Machines & Kernel Methods

- Support Vector Machines are extensions of the support vector classifier using kernel methods
- Support Vector Machines enable non-linear decision boundaries that can be efficiently computed
- Support Vector Machines avoids 'curse of dimensionality' and mapping search using a 'kernel trick'

- Non-linear transformations

[1] *An Introduction to Statistical Learning*

- Lead to high number of features → Computations become unmanageable
(including the danger to run into 'curse of dimensionality')

- Benefits with SVMs

- Enlarge feature space using 'kernel trick' → ensures efficient computations
 - Map training data into a higher-dimensional feature space using Φ
 - Create a separating 'hyperplane' with maximum margin in feature space

- Solve constraint optimization problem

- Using Lagrange multipliers & quadratic programming (cf. earlier classifiers)
 - Solution involves the inner products of the data points (dot products)

- Inner product of two r-vectors a and b is defined as $\langle a, b \rangle = \sum_{i=1}^r a_i b_i$

- Inner product of two data points: $\langle x_i, x_{i'} \rangle = \sum_{j=1} x_{ij} x_{i'j}$

Linear SV Classifier Refined & Role of SVs

- Linear support vector classifier

- Details w.r.t. inner products

- With n parameters $\alpha_i, i = 1, \dots, n$ $f(x) = \beta_0 + \sum_{i=1}^n \alpha_i \langle x, x_i \rangle$
(Lagrange multipliers)

- Use training set to estimate parameters (between all pairs of training data points)

- Estimate $\alpha_1, \dots, \alpha_n$ and β_0 using $\binom{n}{2}$ inner products $\langle x_i, x_{i'} \rangle$
 $n(n-1)/2$ number of pairs

- Evaluate $f(x)$ with a new point

- Compute the inner product between new point x and each of the training points x_i

- Identify support vectors \rightarrow Quadratic programming

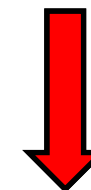
- α_i is zero most of the times (identified as not support vectors)
 - α_i is nonzero several times (identified as the support vectors)

$$\langle x_i, x_{i'} \rangle = \sum_{j=1}^p x_{ij} x_{i'j}$$



$$f(x) = \beta_0 + \sum_{i=1}^n \alpha_i \langle x, x_i \rangle$$

$$f(x) = \beta_0 + \sum_{i=1}^n \alpha_i \langle x, x_i \rangle$$



'big data' reduction & less computing

$$f(x) = \beta_0 + \sum_{i \in S} \alpha_i \langle x, x_i \rangle$$

(S with indices of support vectors)

[6] An Introduction to Statistical Learning

The ('Trusted') Kernel Trick

- Summary for computation

- All that is needed to compute coefficients are inner products

(compute the hyperplane without explicitly carrying out the map into the feature space)

$$f(x) = \beta_0 + \sum_{i=1}^n \alpha_i \langle x, x_i \rangle$$

- Kernel Trick

- Replace the inner product with a generalization of the inner product

(inner product used before)

$$\langle x_i, x_{i'} \rangle = \sum_{j=1}^p x_{ij} x_{i'j}$$

- K is some kernel function

$$K(x_i, x_{i'})$$

(kernel ~ distance measure)

- Kernel types

- Linear kernel

$$K(x_i, x_{i'}) = \sum_{j=1}^p x_{ij} x_{i'j} \quad (\text{linear in features})$$

- Polynomial kernel

$$K(x_i, x_{i'}) = \left(1 + \sum_{j=1}^p x_{ij} x_{i'j}\right)^d \quad (\text{polynomial of degree } d)$$

[6] *An Introduction to Statistical Learning*

- Kernel trick refers to a mechanism of using different kernel functions (e.g. polynomial)
- A kernel is a function that quantifies the similarity of two data points (e.g. close to each other)

Kernel Trick – Example

- Consider again a simple two dimensional dataset

- We found an ideal mapping Φ after long search
- Then we need to transform the whole dataset according to Φ

$$\Phi : (x_1, x_2) \mapsto (x_1^2, x_2^2, \sqrt{2}x_1, \sqrt{2}x_2, 1)$$

- Instead, with the ‘kernel trick’ we ‘wait’ and ‘let the kernel do the job’:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j) \xrightarrow{\text{red arrow}} \min_{\alpha} \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) - \sum \alpha_i$$

(no need to compute the mapping already)

$$\Phi(\mathbf{u}) \cdot \Phi(\mathbf{v}) = (u_1^2, u_2^2, \sqrt{2}u_1, \sqrt{2}u_2, 1) \cdot (v_1^2, v_2^2, \sqrt{2}v_1, \sqrt{2}v_2, 1)$$

$$\Phi(\mathbf{u}) \cdot \Phi(\mathbf{v}) = u_1^2 v_1^2 + u_2^2 v_2^2 + 2u_1 v_1 + 2u_2 v_2 + 1$$

$$\Phi(\mathbf{u}) \cdot \Phi(\mathbf{v}) = (\mathbf{u} \cdot \mathbf{v} + 1)^2 = K(\mathbf{u}, \mathbf{v})$$

(in transformed space still a dot product
in the original space \rightarrow no mapping needed)
(we can save computing time by do not perform the mapping)

- Example shows that the dot product in the transformed space can be expressed in terms of a similarity function in the original space (here dot product is a similarity between two vectors)

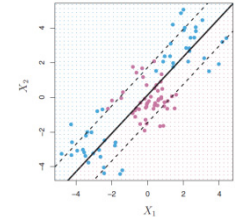
Linear vs. Polynomial Kernel Example

■ Linear kernel

- Enables linear decision boundaries (i.e. like linear support vector classifier)

$$K(x_i, x_{i'}) = \sum_{j=1}^p x_{ij} x_{i'j} \quad (\text{linear in features})$$

(observed useless for non-linear data)



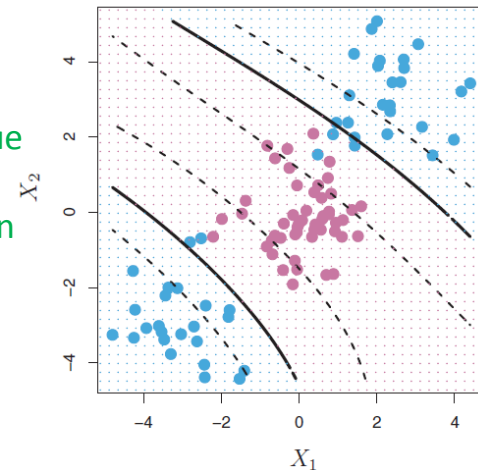
■ Polynomial kernel

- Satisfy Mercer's theorem = trusted kernel
- Enables non-linear decision boundaries (when choosing degree $d > 1$)
- Amounts to fit a support vector classifier in a higher-dimensional space
- Using polynomials of degree d ($d=1$ linear support vector classifier)

$$K(x_i, x_{i'}) = \left(1 + \sum_{j=1}^p x_{ij} x_{i'j}\right)^d \quad (\text{polynomial of degree } d)$$

(SVM with polynomial kernel of degree 3)

(significantly improved decision rule due to much more flexible decision boundary)

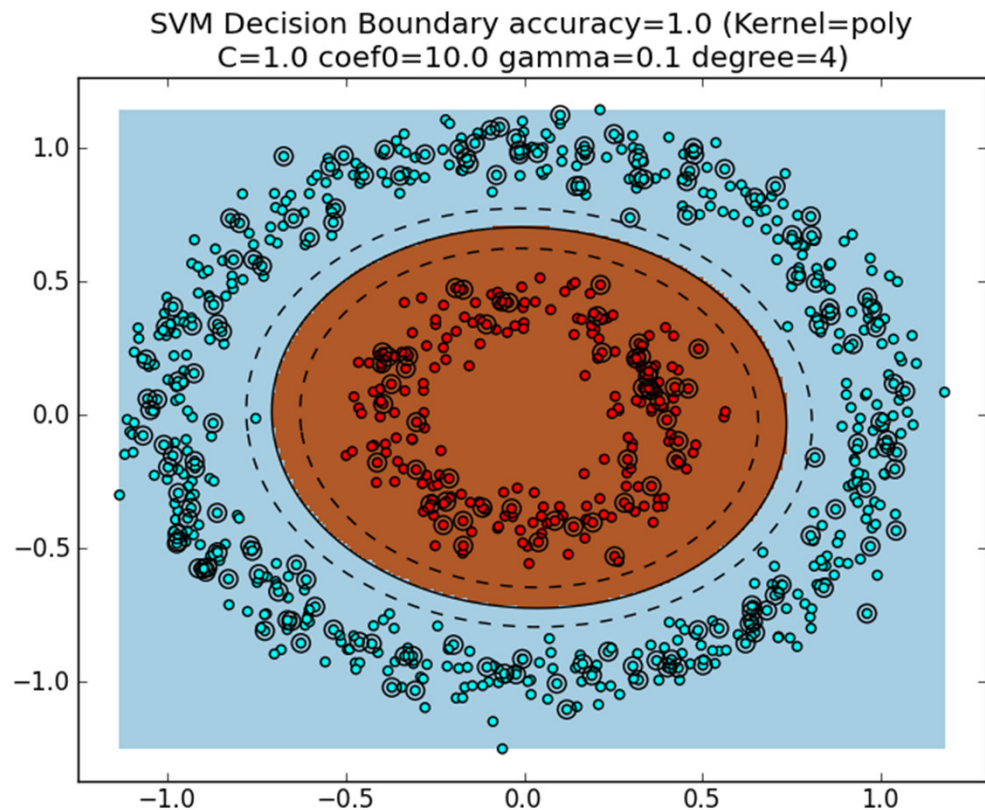
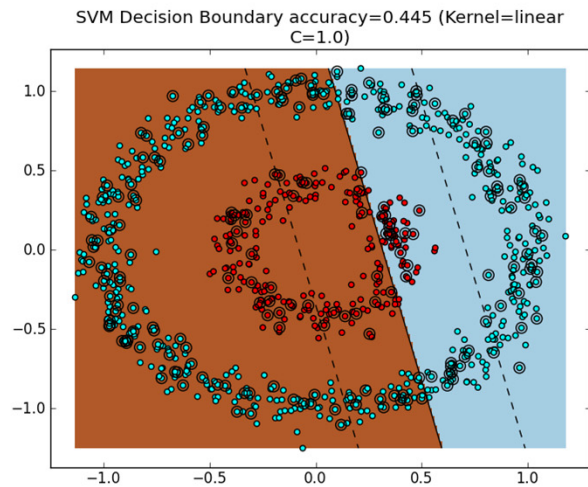


[6] *An Introduction to Statistical Learning*

■ Polynomial kernel applied to non-linear data is an improvement over linear support vector classifiers

Polynomial Kernel Example

- Circled data points are from the test set



$$K(x_i, x_{i'}) = (1 + \sum_{j=1}^p x_{ij} x_{i'j})^d \quad \rightarrow$$

[20] E. Kim

RBF Kernel

- Radial Basis Function (RBF) kernel

(also known as radial kernel)

- One of the mostly used kernel function
- Uses parameter γ as positive constant

$$K(x_i, x_{i'}) = \exp(-\gamma \sum_{j=1}^p (x_{ij} - x_{i'j})^2)$$

- 'Local Behaviour functionality'

- Related to Euclidean distance measure

(ruler distance)

$$d([x_1, x_2, \dots, x_n], [y_1, y_2, \dots, y_n]) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2}$$

- Example

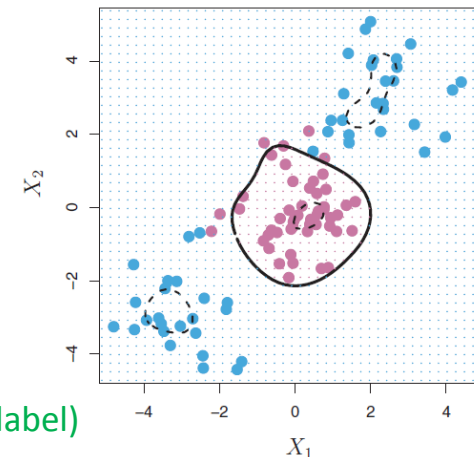
- Use test data $x^* = (x_1^* \dots x_p^*)^T$
- Euclidean distance gives x^* far from x_i

$$\sum_{j=1}^p (x_j^* - x_{ij})^2 \text{ (large value with large distance)}$$

➔ $K(x^*, x_i) = \exp(-\gamma \sum_{j=1}^p (x_j^* - x_{ij})^2)$ (tiny value)

➔ $f(x) = \beta_0 + \sum_{i \in S} \alpha_i K(x, x_i)$ (training data x_i plays no role for x^* & its class label)

(SVM with radial kernel)



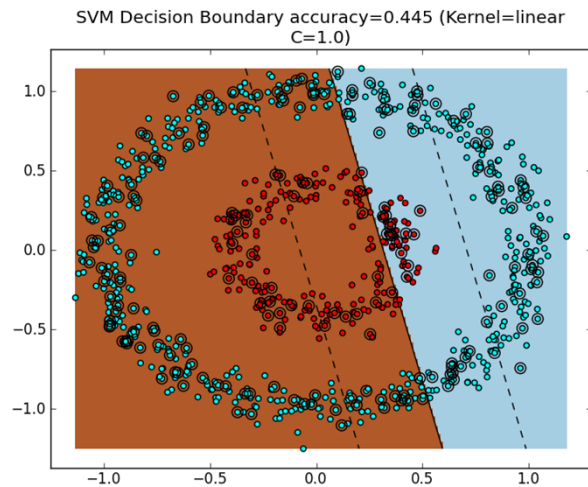
[1] An Introduction to Statistical Learning

- RBF kernel have local behaviour (only nearby training data points have an effect on the class label)

RBF Kernel Example

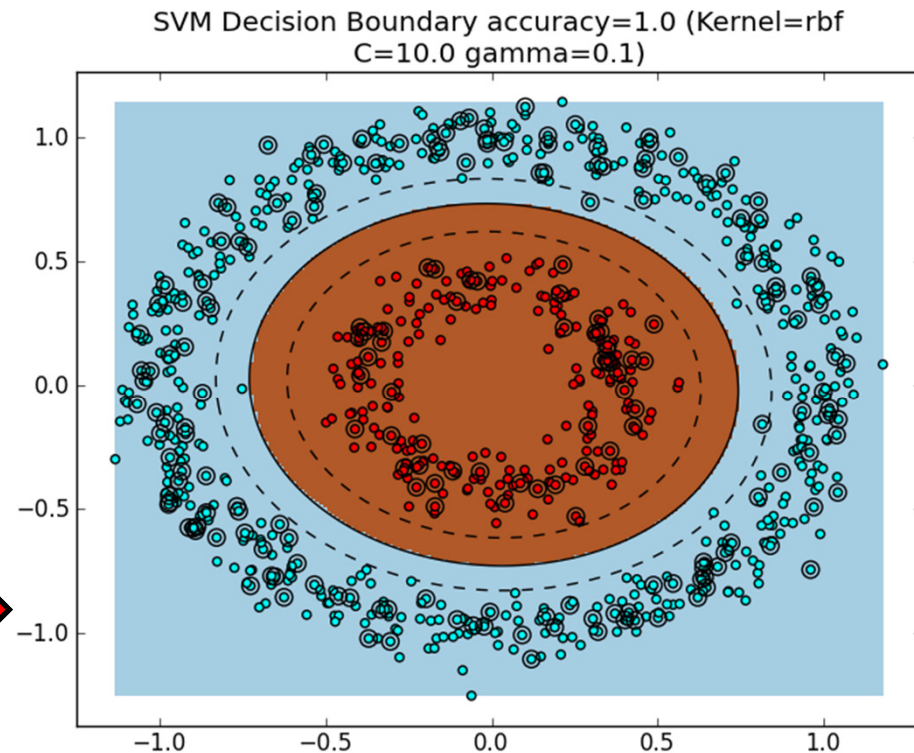
- Circled data points are from the test set

(similar decision boundary as polynomial kernel)



$$K(x^*, x_i) =$$

$$\exp(-\gamma \sum_{j=1}^p (x_j^* - x_{ij})^2)$$



[20] E. Kim

Exact SVM Definition using non-linear Kernels

- True Support Vector Machines are Support Vector Classifiers combined with a non-linear kernel
- There are many non-linear kernels, but mostly known are polynomial and RBF kernels

[6] *An Introduction to Statistical Learning*

- General form of SVM classifier

- Assuming non-linear kernel function K
- Based on 'smaller' collection S of SVs

$$f(x) = \beta_0 + \sum_{i \in S} \alpha_i K(x, x_i)$$

- Major benefit of Kernels: Computing done in original space

(independent from transformed space)

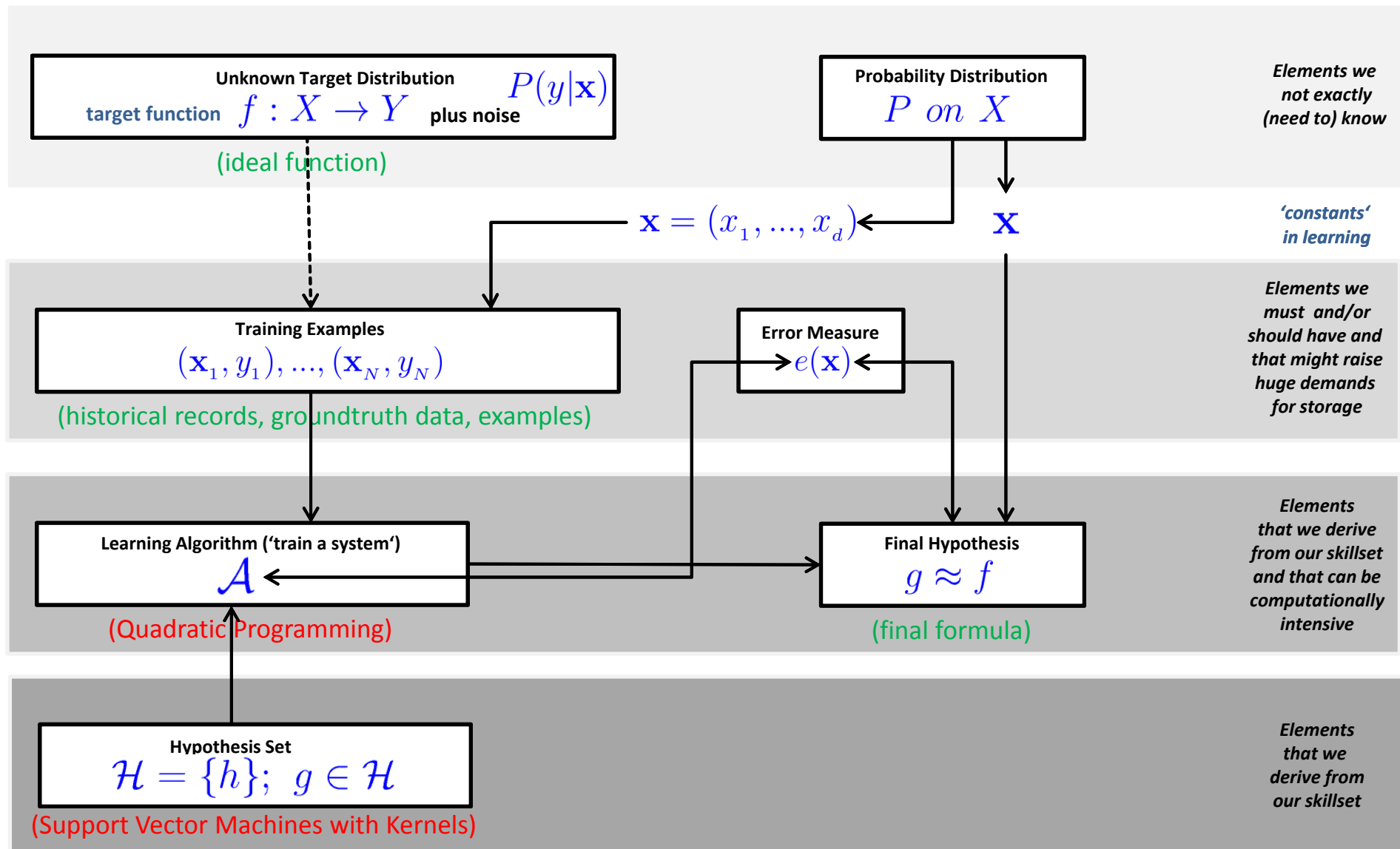
- Linear Kernel $K(x_i, x_{i'}) = \sum_{j=1}^p x_{ij} x_{i'j}$ (linear in features)

- Polynomial Kernel $K(x_i, x_{i'}) = (1 + \sum_{j=1}^p x_{ij} x_{i'j})^d$ (polynomial of degree d)

- RBF Kernel $K(x_i, x_{i'}) = \exp(-\gamma \sum_{j=1}^p (x_{ij} - x_{i'j})^2)$ (large distance, small impact)

(the win: kernel can compute this without ever computing the coordinates of the data in that space, next slides)

Solution Tools: Support Vector Classifier & QP Algorithm



Non-Linear Transformations with Support Vector Machines

- Same idea: work in z space instead of x space with SVMs

- Understanding effect on solving (labels remain same)
- SVMs will give the 'best separator'

$$\mathcal{L}(\alpha) = \sum_{n=1}^N \alpha_n - \frac{1}{2} \sum_{n=1}^N \sum_{m=1}^N y_n y_m \alpha_n \alpha_m \mathbf{x}_n^T \mathbf{x}_m \quad (\text{replace this simply with } z\text{'s obtained by } \Phi)$$

(result from this new inner product is given to quadratic programming optimization as input as before)

- Value: inner product is done with z instead of x – the only change
- Result after quadratic programming is hyperplane in z space using the value
- Impacts of Φ to optimization
 - From linear to 2D → probably no drastic change
 - From 2D to million-D → sounds like a drastic change but just inner product
 - Input for $K(x_i, x'_i)$ remains the number of data points (nothing to do with million-D)
 - Computing longer million-D vectors is 'easy' – optimization steps 'difficult'

■ Infinite-D Z spaces are possible since the non-linear transformation does not affect the optimization

Kernels & Infinite Z spaces

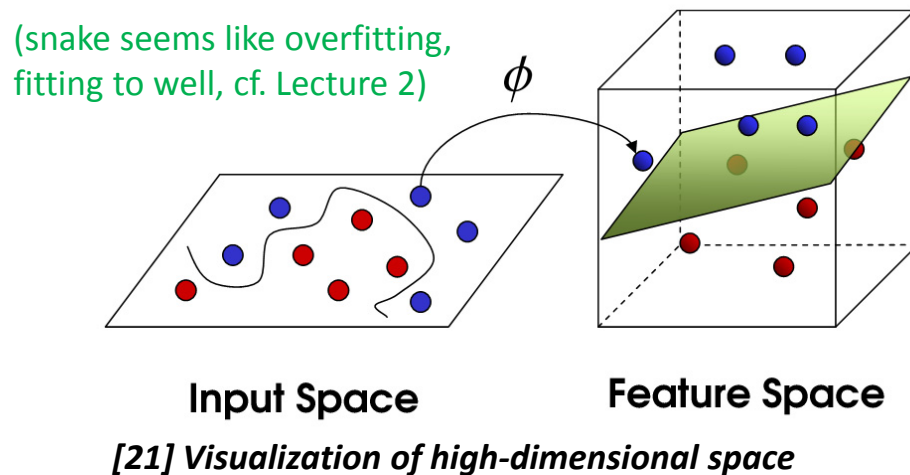
- Understanding **advantage of using a kernel**
 - Better than simply enlarging the feature space
 - E.g. using functions of the original features like $X_1, X_1^2, X_2, X_2^2, \dots, X_p, X_p^2$.
- **Computational advantages**
 - By using kernels only compute $K(x_i, x_{i'})$
 - Limited to just all $\binom{n}{2}$ distinct pairs i, i'
(number of 2 element sets from n element set)
 - Computing without explicitly working in the enlarged feature space
 - Important because in many applications the enlarged feature space is large
(computing would be infeasible then w/o kernels)
- **Infinite-D Z spaces** $K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$
 - Possible since all that is needed to compute coefficients are inner products

[1] *An Introduction to Statistical Learning*

■ **Kernel methods like RBF have an implicit and infinite-dimensional features space that is not 'visited'**

Visualization of SVs

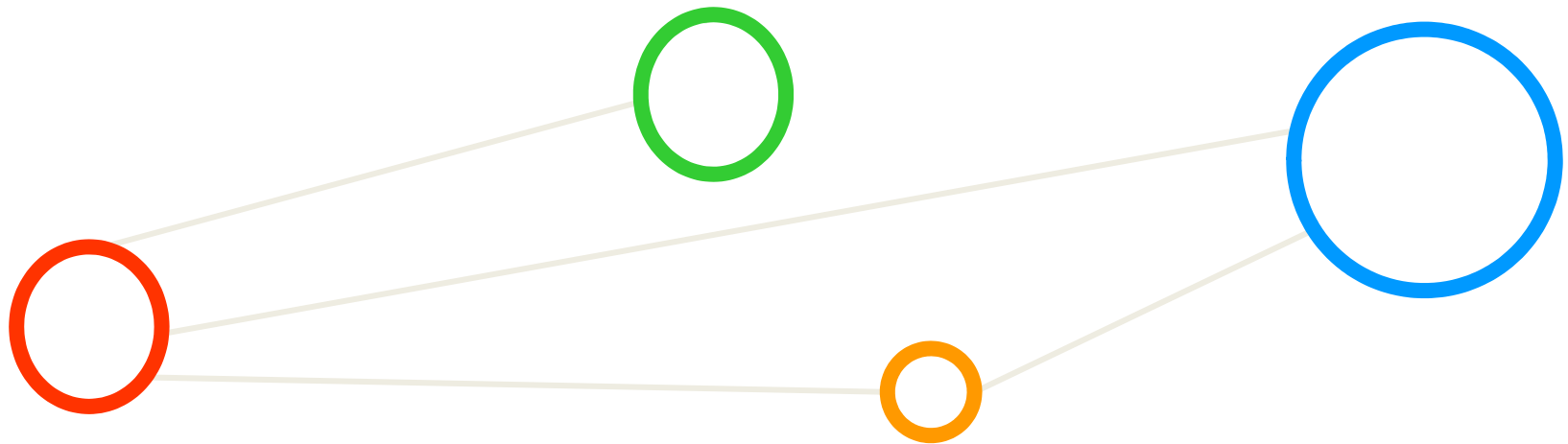
- Problem: z-Space is infinite (unknown)
 - How can the Support Vectors (from existing points) be visualized?
 - Solution: non-zero alphas have been the identified support vectors
(solution of quadratic programming optimization will be a set of alphas we can visualize)
 - Support vectors exist in Z – space (just transformed original data points)
 - Example: million-D means a million-D vector for \mathbf{W}
 - But number of support vector is very low, expected E_{out} is related to #SVs
(generalization behaviour despite million-D & snake-like overfitting)



- Counting the number of support vectors remains to be a good indicator for generalization behaviour even when performing non-linear transforms and kernel methods that can lead to infinite-D spaces

(rule of thumb)

Lecture Bibliography



Lecture Bibliography (1)

- [1] Species Iris Group of North America Database,
Online: <http://www.signa.org>
- [2] UCI Machine Learning Repository Iris Dataset,
Online: <https://archive.ics.uci.edu/ml/datasets/Iris>
- [3] Wikipedia 'Sepal',
Online: <https://en.wikipedia.org/wiki/Sepal>
- [4] Rattle Library for R,
Online: <http://rattle.togaware.com/>
- [5] B2Share, 'Iris Dataset LibSVM Format Preprocessing',
Online: <https://b2share.eudat.eu/records/37fb24847a73489a9c569d7033ad0238>
- [6] An Introduction to Statistical Learning with Applications in R,
Online: <http://www-bcf.usc.edu/~gareth/ISL/index.html>
- [7] F. Rosenblatt, 'The Perceptron--a perceiving and recognizing automaton',
Report 85-460-1, Cornell Aeronautical Laboratory, 1957
- [8] Rosenblatt, 'The Perceptron: A probabilistic model for information storage and organization in the brain',
Psychological Review 65(6), pp. 386-408, 1958
- [9] PLA Algorithm, YouTube Video,
Online:
- [10] C. Shearer, CRISP-DM model, Journal Data Warehousing, 5:13
- [11] Pete Chapman, 'CRISP-DM User Guide', 1999,
Online: <http://lyle.smu.edu/~mhd/8331f03/crisp.pdf>

Lecture Bibliography (2)

- [11] Introduction to Data Mining, Pang-Ning Tan, Michael Steinbach, Vipin Kumar, Addison Wesley, ISBN 0321321367, English, ~769 pages, 2005
- [12] Udacity, 'Overfitting',
Online: <https://www.youtube.com/watch?v=CxAxRCv9WoA>
- [13] Wikipedia on 'Statistical Learning Theory',
Online: http://en.wikipedia.org/wiki/Statistical_learning_theory
- [14] Leslie G. Valiant, 'A Theory of the Learnable', Communications of the ACM 27(11):1134–1142, 1984,
Online: <https://people.mpi-inf.mpg.de/~mehlhorn/SeminarEvolvability/ValiantLearnable.pdf>
- [15] Y.S. Abu-Mostafa, M. Magdon-Ismail, Hsuan-Tien Lin, 'Learning from Data – A short course', AML Book, ISBN 978-1-60049-006-4
- [16] Rome Image dataset
Online: <https://b2share.eudat.eu/records/daf6c389e54340b4b1416cf874251e77>
- [17] G. Cavallaro, M. Riedel, et al., "Smart data analytics methods for remote sensing applications," in the Proceedings of the IEEE Geoscience and Remote Sensing Symposium, Quebec City, QC, 2014, pp. 1405-1408.
- [18] Rome Image dataset
Online: <https://b2share.eudat.eu/records/daf6c389e54340b4b1416cf874251e77>
- [19] YouTube Video, "What is Remote Sensing?"
Online: <https://www.youtube.com/watch?v=nU-CjAKry5c>
- [20] E. Kim, 'Everything You Wanted to Know about the Kernel Trick',
Online: http://www.eric-kim.net/eric-kim-net/posts/1/kernel_trick.html

Lecture Bibliography (3)

- [21] Visualization of high-dimensional space,
Online: <http://i.imgur.com/WuxyO.png>
- [22] YouTube, 'SVM with Polynomial Kernel',
Online: <https://www.youtube.com/watch?v=3liCbRZPrZA>
- [23] 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.
- [24] Indian Pines Raw and Processed
Online: <http://hdl.handle.net/11304/9ec5eac8-61b4-4617-ae1c-1f8c8cd3cd74>
- [25] LibSVM Webpage,
Online: <https://www.csie.ntu.edu.tw/~cjlin/libsvm/>
- [26] M. Goetz, M. Riedel et al., 'On Parallel and Scalable Classification and Clustering Techniques for Earth Science Datasets' 6th Workshop on Data Mining in Earth System Science, Proceedings of the International Conference of Computational Science (ICCS), Reykjavik,
Online: <http://www.proceedings.com/26605.html>
- [27] Original piSVM tool,
Online: <http://pisvm.sourceforge.net/>
- [28] 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>

Acknowledgements – Membership of my HPDP Research Group



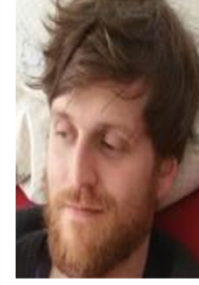
PD Dr.
G. Cavallaro



Senior PhD
Student A.S. Memon



Senior PhD
Student M.S. Memon



PhD Student
E. Erlingsson



PhD Student
C. Bakarat



Dr. M. Goetz
(now KIT)



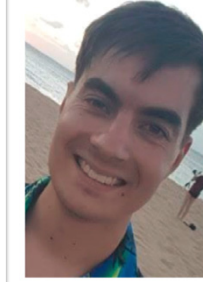
MSc M.
Richerzhagen,
now other group



MSc
P. Glock
(now INM-1)



MSc
C. Bodenstein
(now Soccerwatch.tv)



MSc Student
G.S. Guðmundsson
(Landsverkjun)

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

