Hochschule Offenburg
offenburg.university

Machine Learning V

Non-Linear Models – Part I

Machine Learning IV

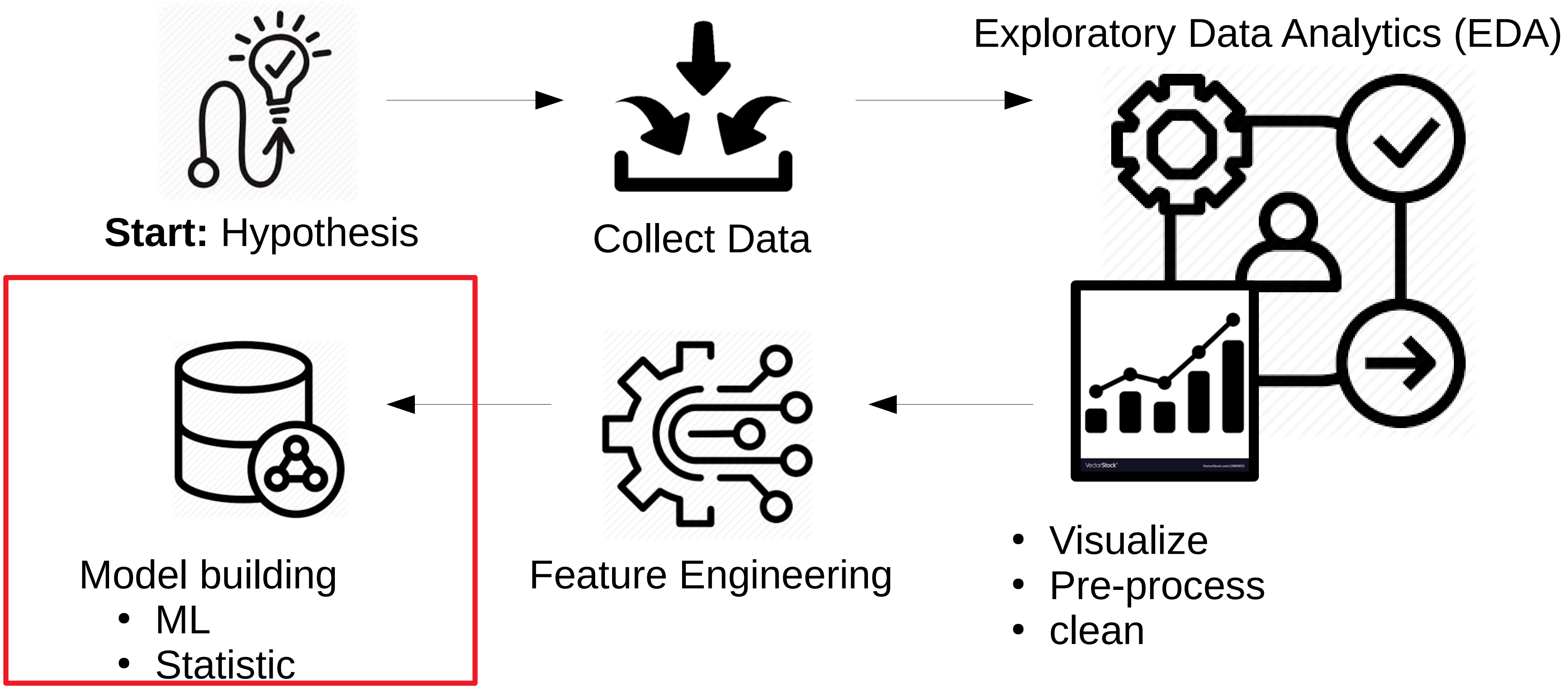
Outline

- **Part I : The Need for non-linear models**

- **Part I : Extending the simple linear classifier**
  - Adding non-linearity
  - Simple Neural Networks

- **Part II : Support Vector Machines**
  - Linear SVMs
  - The “Kernel-Trick”

Recall: Data Science Workflow



Recall: Classification

Supervised Learning: Annotated Training Data

Annotated Train Data

Vector representation
(feature extraction)

Classification
Algorithm

(X,Y)

$$\begin{pmatrix} 5 \\ 8 \\ 1 \\ 5 \\ 2 \\ 3 \\ 0 \\ 2 \\ 8 \\ \dots \\ 5 \end{pmatrix}$$

$X \in \mathbb{R}^n$

ML

Y': {-1,1}

Prediction

Recall: Classification

LEARNING: is a optimization problem → Finding the best function separating

$$min(\|f(X,Y),Y'\|)$$

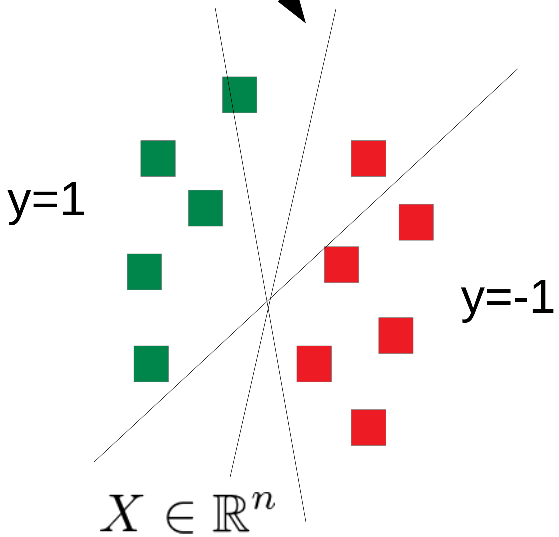
(X,Y)

$$\begin{pmatrix} 5 \\ 8 \\ 1 \\ 5 \\ 2 \\ 3 \\ 0 \\ 2 \\ 8 \\ \dots \\ 5 \end{pmatrix}$$

$X \in \mathbb{R}^n$

ML

Y': {-1,1}



Recall: Linear Classifier

A Simple Linear Model: binary classification

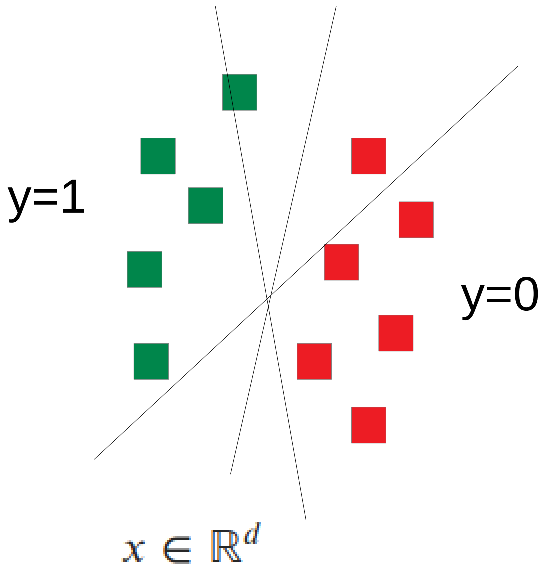
Parameterization of prediction function f with d-dimensional data as:

$$f(x) = y' = sgn(w^T x + b) = sgn(\sum_{j=0}^{d} x_j w_j + b_j)$$

With data samples $x \in \mathbb{R}^d$

Model parameters $w \in \mathbb{R}^d$

Model: hyper plane



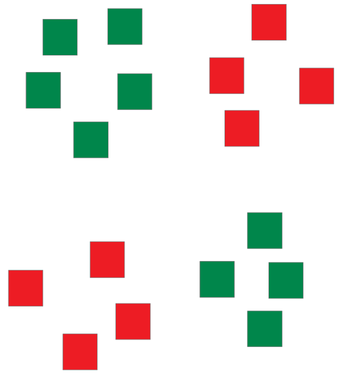
Limitations of Linear Models

(Obviously), linear models have limitations

Consider this very simple binary classification Example:

How to separate "green" from "red" with a linear Model (= hyper plane)?

Simple counter example



$x \in \mathbb{R}^d$

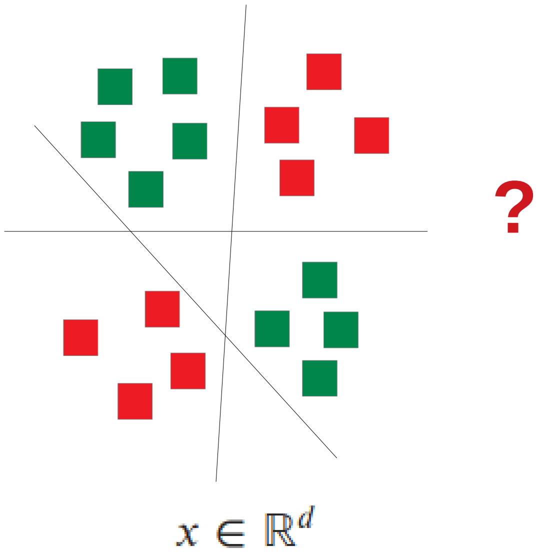
Limitations of Linear Models

(Obviously), linear models have limitations

Consider this very simple binary classification Example:

How to separate "green" from "red" with a linear Model (= hyper plane)?

Simple counter example



Limitations of Linear Models

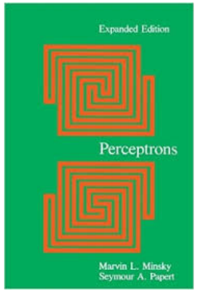
(Obviously), linear models have limitations

Consider this very simple binary classification Example:

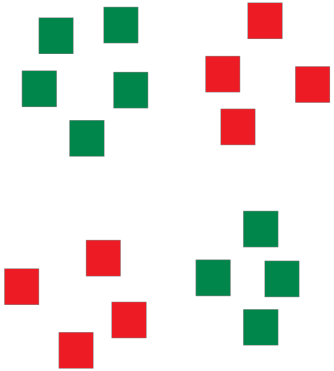
Simple counter example

→ **known as "X-Or" Problem**

→ **one reason for the so-called "AI Winter"**



Caused by the Minsky book
On the shortcomings of the
First neural networks...

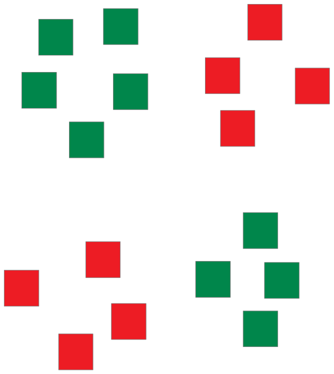


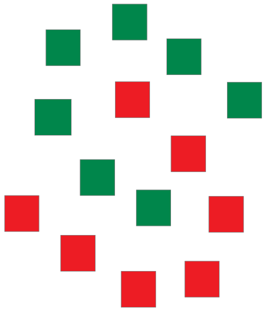
$x \in \mathbb{R}^d$

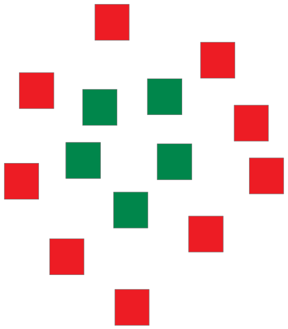
Limitations of Linear Models

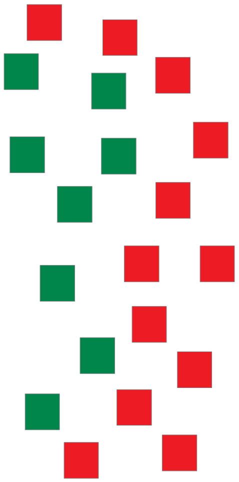
(Obviously), linear models have limitations

More simple (binary 2D) examples:









Limitations of Linear Models

Why are linear models working at all?

Limitations of Linear Models

Why are linear models working at all?

→ very high dimensional feature spaces often time allow linear models to Separate the data

→ very simple (linear) model even can be of advantage in theses settings:

- **“curse of dimensionality”**
- **Avoid overfitting**

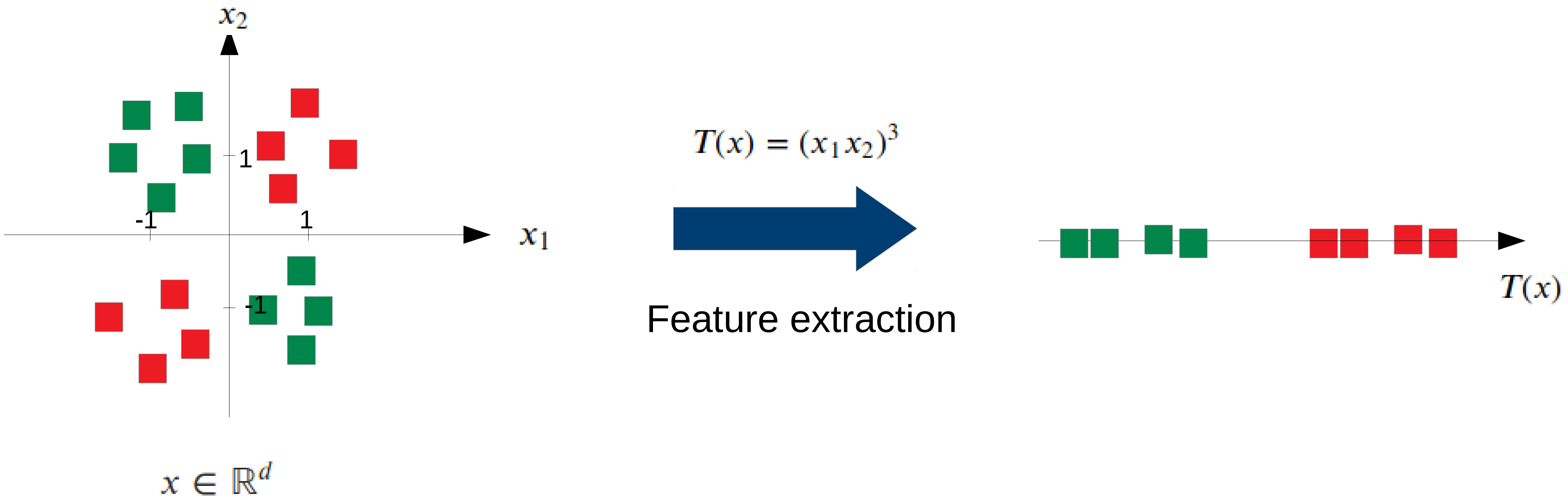
Adding non-linearity

Three canonical ways (we already saw two of them):

Adding non-linearity

Three canonical ways:

1. extracting non-linear features:

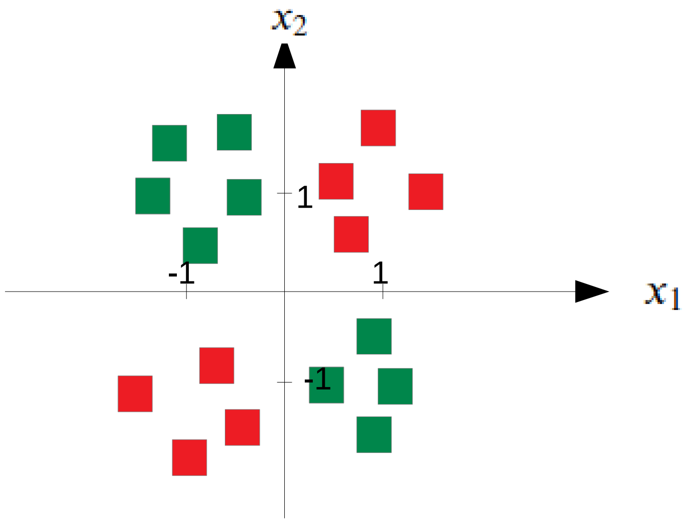


Adding non-linearity

Three canonical ways:

1. extracting non-linear features:

NOTE: just an example, usually not
That simple for real data.
→ see ML Part VII



$x \in \mathbb{R}^d$

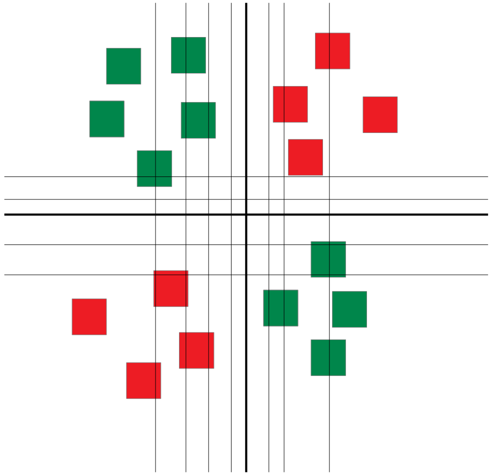
$$T(x) = (x_1 x_2)^3$$

Feature extraction

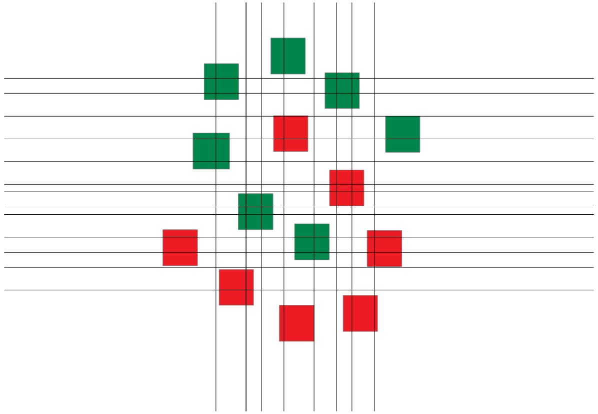
Adding non-linearity

Three canonical ways:

2. use ensembles of linear models (like Random Forrest)



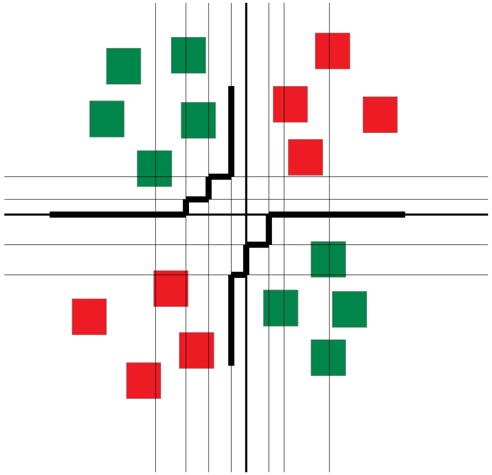
$x \in \mathbb{R}^d$



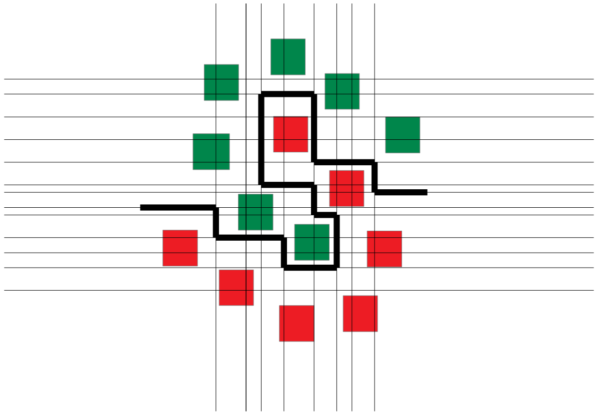
Adding non-linearity

Three canonical ways:

2. use ensembles of linear models → approximation of non-linear models by piece-wise linear models



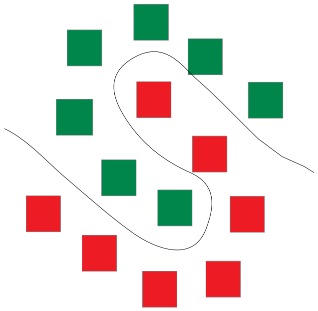
$x \in \mathbb{R}^d$



Adding non-linearity

Three canonical ways:

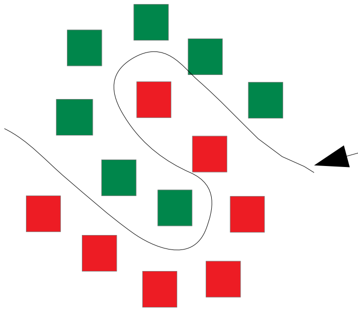
3. use non-linear functions



Adding non-linearity

Three canonical ways:

3. use non-linear functions



How to parameterize the non-linear model?

Adding non-linearity

Adding non-linearity to our simple linear classifier

$$f(x) = y' = w^T x = \sum_{j=0}^{d} w_j x_j \quad \xrightarrow{\text{Train by solving optimization}} \quad \arg\min_{w} \sum_{i=0}^{N} L(y_i, w^T x_i)$$

$$f(x) = \phi(w^T x)$$

Step I: add a very simple element-wise non-linear mapping.

(like in the previous feature extraction example)

Note: for simplicity we neglect b (assuming b=0), for real problems: Analog Optimization with b...

Adding non-linearity

Adding non-linearity to our simple linear classifier

$$f(x) = y' = w^T x = \sum_{j=0}^{d} w_j x_j$$

$$f(x) = \phi(w^T x)$$

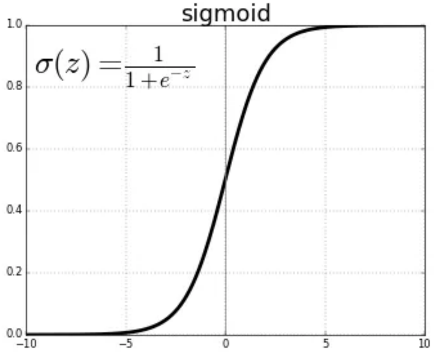
What are good choices for these functions?

Adding non-linearity

Adding non-linearity to our simple linear classifier

$$f(x) = y' = w^T x = \sum_{j=0}^{d} w_j x_j$$

$$f(x) = \phi(w^T x)$$



What are good choices for these functions?

Properties:

- Between 0 and 1 → pseudo probability interpretation
- Stable range of output → gradient optimization

Common choices:

- **Sigmoid function**
- Tanh
- ...

Adding non-linearity

Adding non-linearity to our simple linear classifier

$$f(x) = y' = w^T x = \sum_{j=0}^{d} w_j x_j$$

$$f(x) = \phi_3(w_3\phi_2(W_2\phi_1(W_1 x)))$$

W are now Matrices to produce vector outputs
(recall multi class formulation for linear models)

Step II: concatenate several of these operations

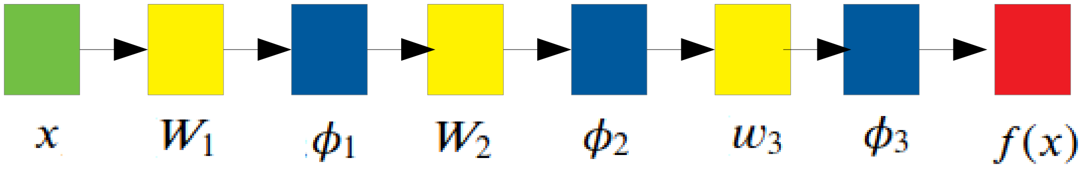
(like we do in the ensemble approach)

Adding non-linearity

Let’s display this in a slightly different way (no change in math formulation!)

$$f(x) = y' = w^T x = \sum_{j=0}^{d} w_j x_j$$

$$f(x) = \phi_3(w_3\phi_2(W_2\phi_1(W_1 x)))$$



Matrix/Vector Mult

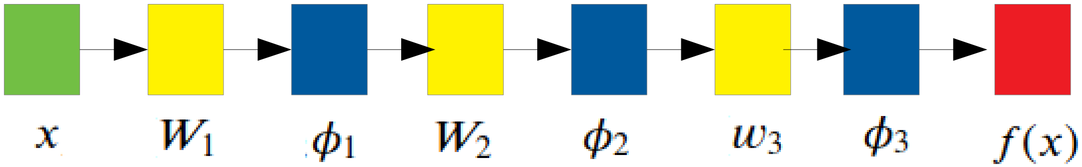
Element wise non-linear

Adding non-linearity

Let's display this in a slightly different way (no change in math formulation!)

$$f(x) = y' = w^T x = \sum_{j=0}^{d} w_j x_j$$

$$f(x) = \phi_3(w_3 \phi_2(W_2 \phi_1(W_1 x)))$$



Note: there is a theoretical prove that we need only two concatenations to approximate any smooth function if the W are large enough!

Matrix/Vector Mult

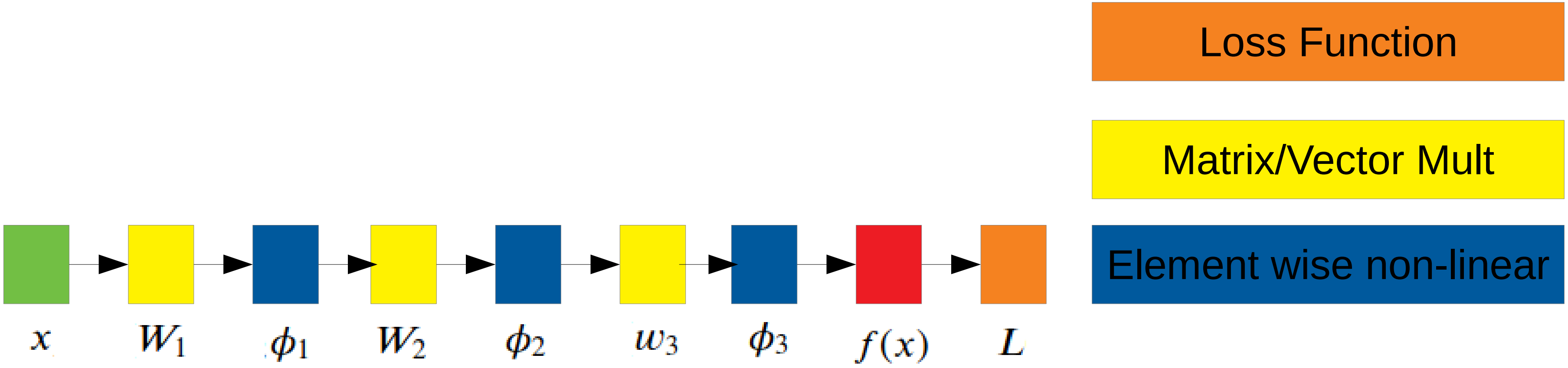
Element wise non-linear

Adding non-linearity

For training (optimization), we need to add loss function

→ **same approach as in the linear case:**

$$\arg\min_{w} \sum_{i=0}^{N} L(y_i, f(x))$$



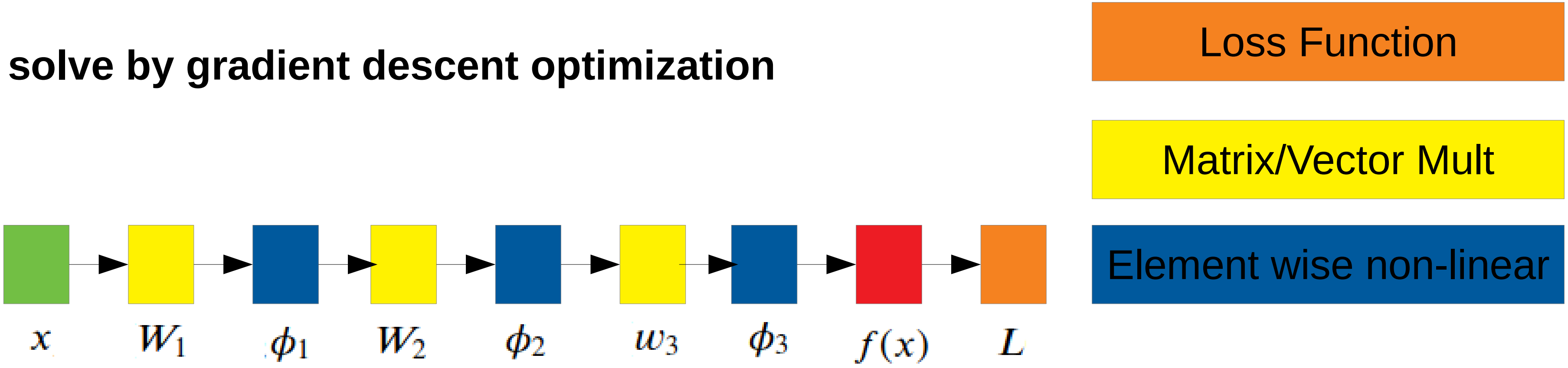
Adding non-linearity

For training (optimization), we need to add loss function

→ **same approach as in the linear case:**

$$\arg\min_{w} \sum_{i=0}^{N} L(y_i, f(x))$$

→ **solve by gradient descent optimization**



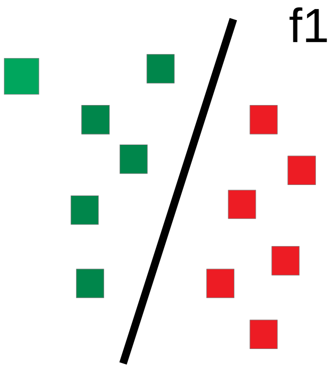
Adding non-linearity

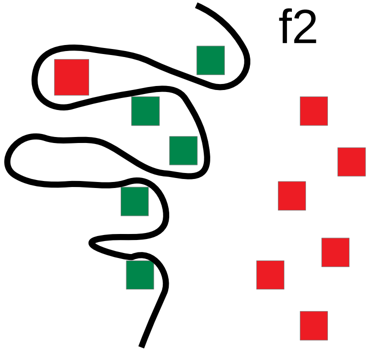
Recall **OVERFITTING**

Model “to close” to train data

With non-linear model much more likely to happen in practice.

→ **we need to work against this...**





Adding non-linearity

Adding regularization term to the Loss function

→ **here L2 regularization:**

$$\arg\min_{w} \sum_{i=0}^{N} L(y_i, f(x)) + \lambda \sum_j w_j^2$$

All parameters to be learned

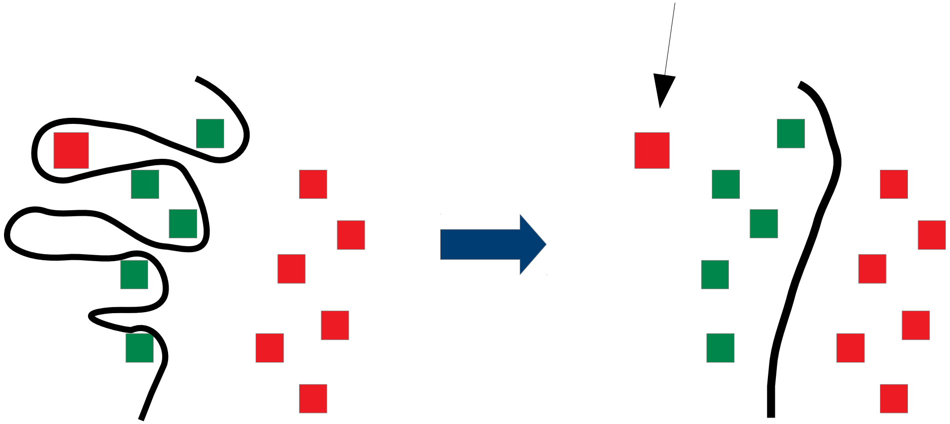
Scalar hyper parameter: impact of regularization

Adding non-linearity

Adding regularization term to the Loss function

→ **here L2 regularization:**

$$\arg\min_{w} \sum_{i=0}^{N} L(y_i, f(x)) + \lambda \sum_j w_j^2$$



→ **L1 regularization:**

$$\arg\min_{w} \sum_{i=0}^{N} L(y_i, f(x)) + \lambda \sum_j |w_j|$$

Regularization will punish high parameter values
→ smoother model
→ training errors allowed !

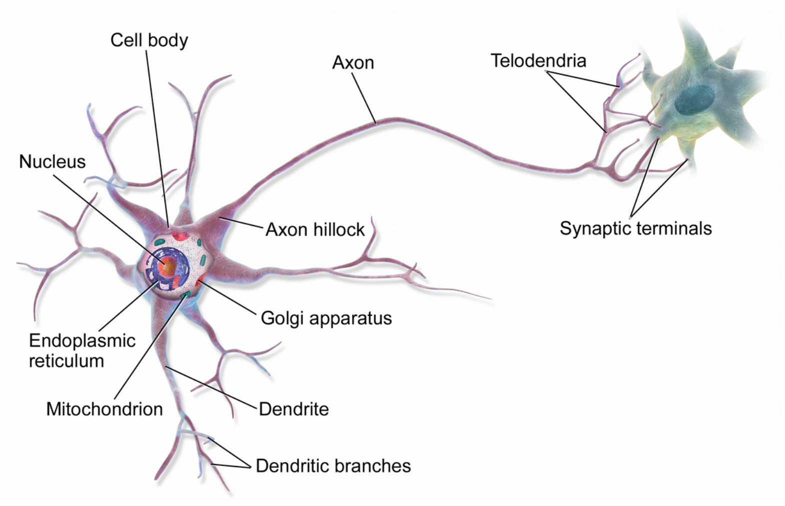
A simple Neural Network

You probably have not noticed, but we just build a simple neural network!

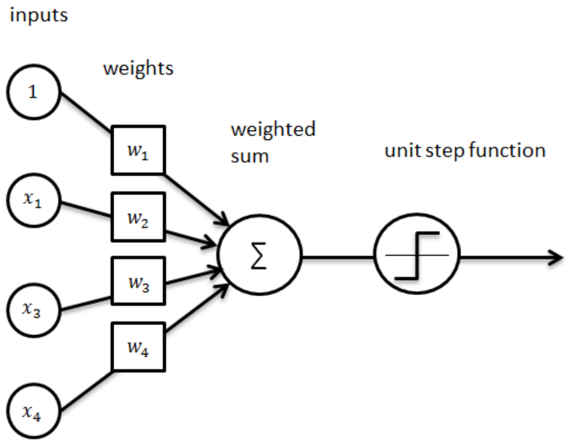
A simple Neural Network

Reinterpretation – no change in mathematical formulation!

A “neuron” (after Rosenblatt’s Perceptron)



Biological model (from wikipedia)



Rosenblatt’s perceptron model (from wikipedia)

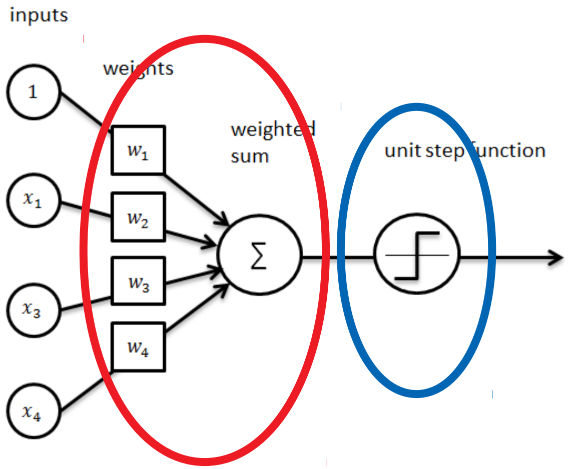
A simple Neural Network

Reinterpretation – no change in mathematical formulation!

A “neuron” (after Rosenblatt’s Perceptron)

Scalar Product

Non-linear element-wise function

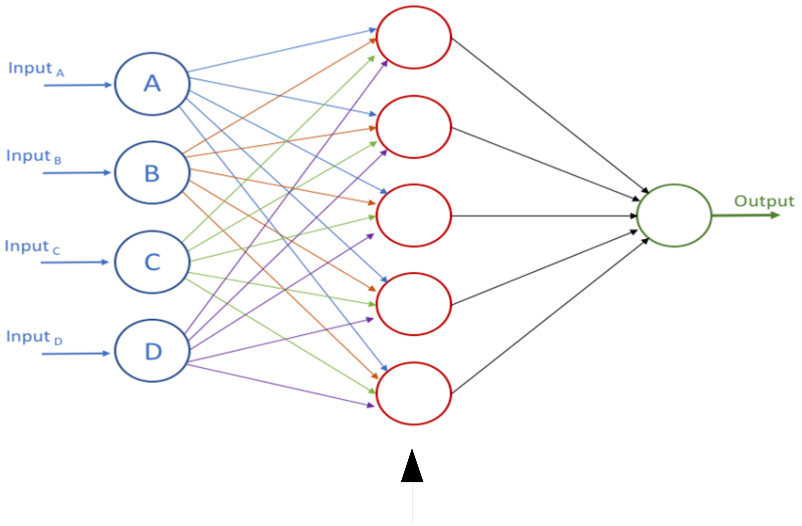


Rosenblatt’s perceptron model (from wikipedia)

A simple Neural Network

Reinterpretation – no change in mathematical formulation!

A simple (“fully connected”) Neural Network:



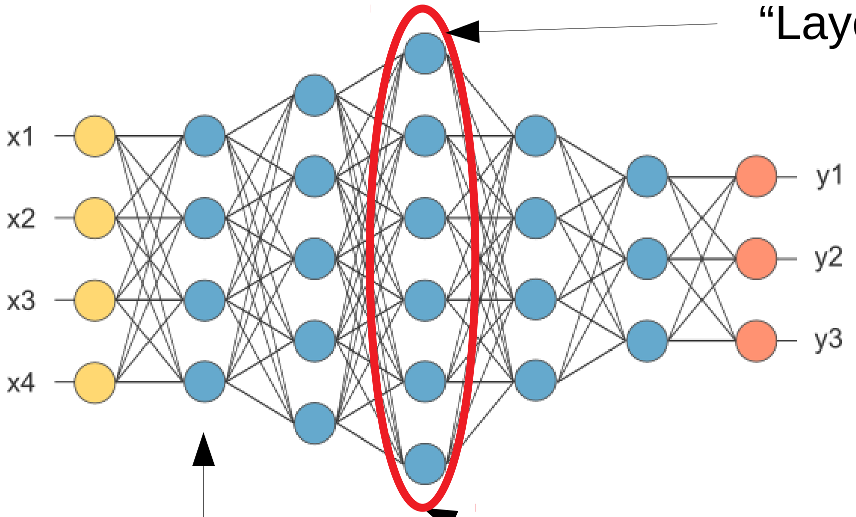
“Neurons” → Scalar Product turns into Matrix Mult

A simple Neural Network

Reinterpretation – no change in mathematical formulation!

A deeper (“fully connected”) Neural Network:

“Layer of the Network



$\phi_1(W_1)$

$\phi_3(W_3)$

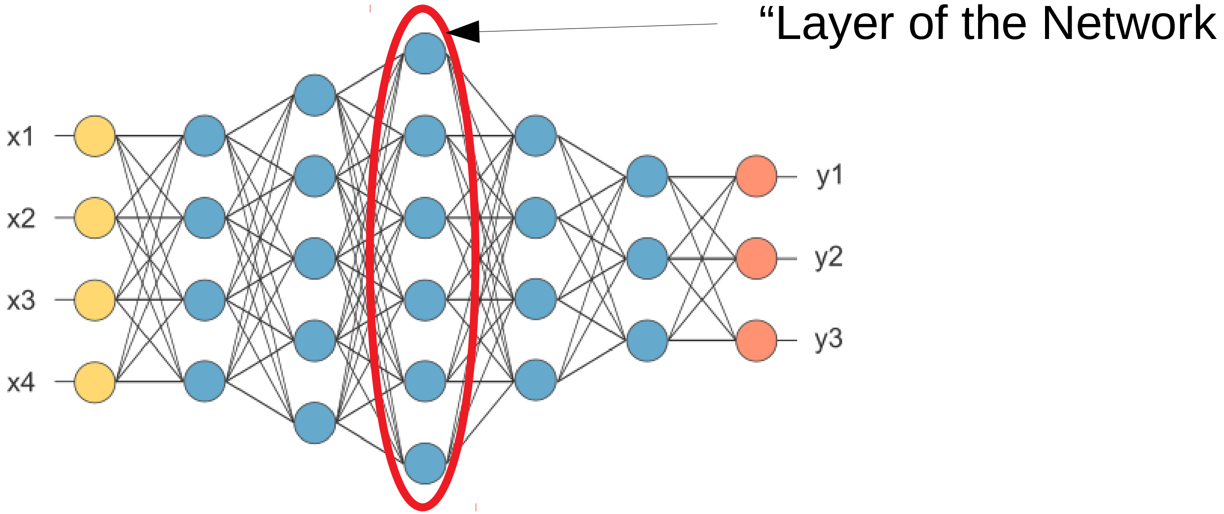
Training still via optimization of

$$\arg\min_{w} \sum_{i=0}^{N} L(y_i, f(x)) + \lambda \sum_j w_j^2$$

A simple Neural Network

A Note on “DEEP LEARNING”

This is more than just many layers! → different network topology and operators



Discussion

Part II coming up ...