Introduction to Machine Learning

INSA 2018

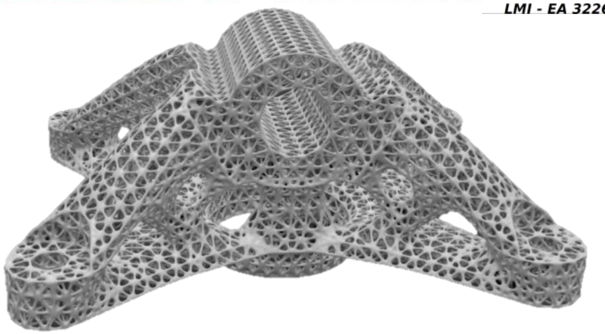
Martin-Pierre SCHMIDT
MSD3@3ds.com
martinpierre.schmidt@insa-rouen.fr

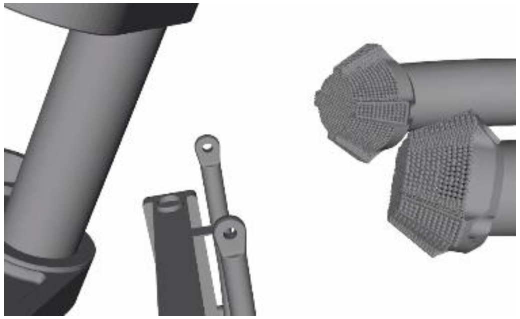
General Information

General Information

Personal presentation

- Martin-Pierre Schmidt
  - martinpierre.schmidt@insa-rouen.fr
  - MSD3@3ds.com
- Dassault Systèmes
  - Scientific Research Team
- LMI INSA Rouen
  - PhD Thesis
- Shape Synthesis through Numerical Optimization





General Information

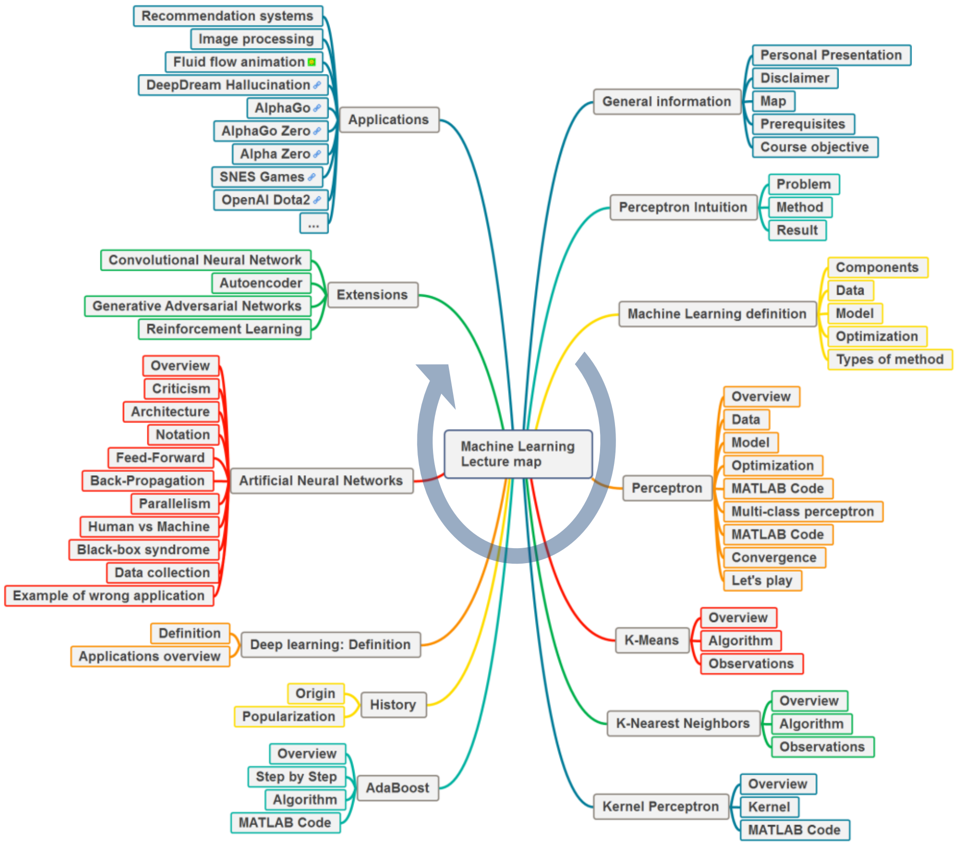
Disclaimer

- English
- Tu
- First iteration of this course
- Not from INSA

Martin-Pierre SCHMIDT
MSD3@3ds.com
martinpierre.schmidt@insa-rouen.fr

General Information

Map



Martin-Pierre SCHMIDT
MSD3@3ds.com
martinpierre.schmidt@insa-rouen.fr

General Information

Prerequisites

Poll: ANN BP

Calculus	Linear Algebra	Computer Science
Multivariable functions	Matrix-Vector arithmetic	Algorithmics
Partial derivatives	Multi-index notation	Numerical optimization
		Parallelism

Martin-Pierre SCHMIDT
MSD3@3ds.com
martinpierre.schmidt@insa-rouen.fr

General Information

Course Objectives

- Explore multiple different **Machine Learning** methods
- Work our way to **Artificial Neural Networks** (ANN)
- Understanding of underlying concepts and implementations
- Intuition on more complex systems
- **Pragmatism** and **critical thinking**

Martin-Pierre SCHMIDT
MSD3@3ds.com
martinpierre.schmidt@insa-rouen.fr

Perceptron: Intuition

Martin-Pierre SCHMIDT
MSD3@3ds.com
martinpierre.schmidt@insa-rouen.fr

Perceptron: Intuition

Problem

Poll: Cat

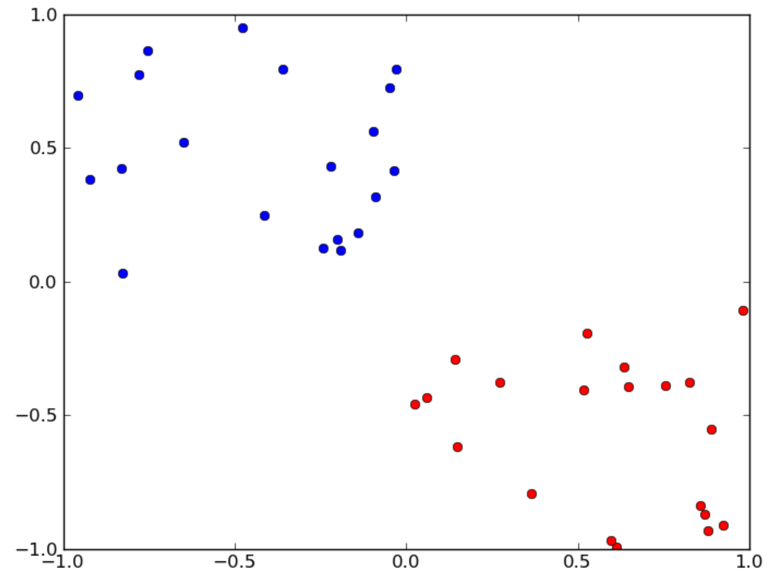
- Input
  - Feature vector
- Output
  - Discrete value
- Goal
  - Make a decision (Discrete value) based on the input data (Feature vector)

Martin-Pierre SCHMIDT
MSD3@3ds.com
martinpierre.schmidt@insa-rouen.fr

Perceptron: Intuition

Problem

- Input vector of 2 scalar value
  - Each possible input can be represented as a 2D point on a plane
- Output value
  - The output is a TRUE/FALSE answer based on the position of the 2D point

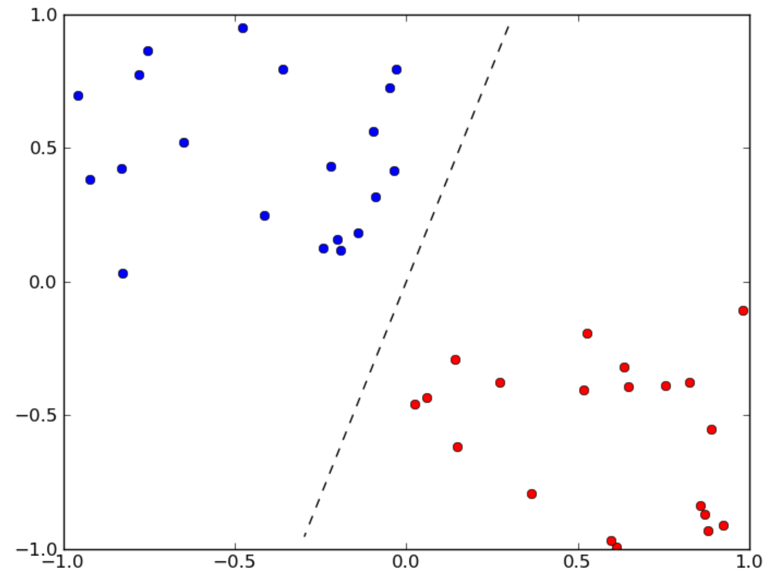


Martin-Pierre SCHMIDT
MSD3@3ds.com
martinpierre.schmidt@insa-rouen.fr

Perceptron: Intuition

Method

- Model
  - The perceptron is a **linear classifier**
- Decision function / Activation function
  - Gives the TRUE/FALSE answer based on a data point and a given line
- Training algorithm
  - Method to choose a line that gives satisfying results when coupled with the decision function

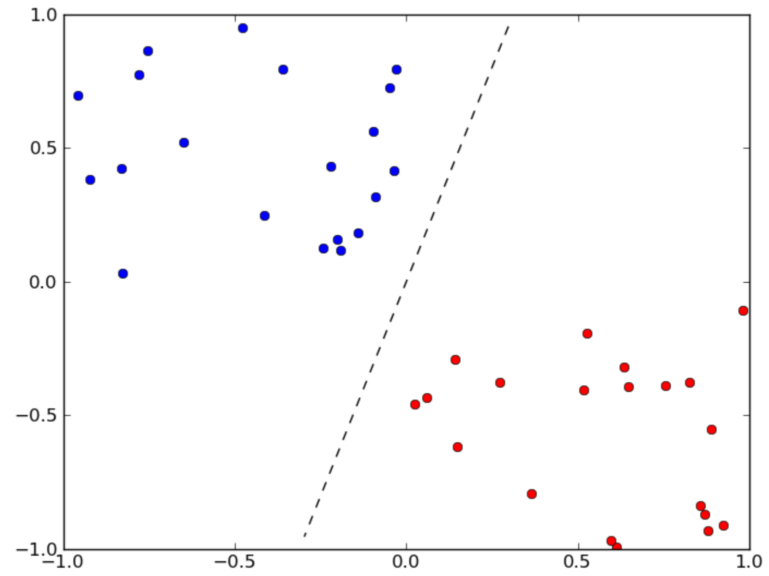


Martin-Pierre SCHMIDT
MSD3@3ds.com
martinpierre.schmidt@insa-rouen.fr

Perceptron: Intuition

Result

- Potent generalizing model
- Slow to train
- Fast to evaluate



Martin-Pierre SCHMIDT
MSD3@3ds.com
martinpierre.schmidt@insa-rouen.fr

Machine Learning : Definition

Martin-Pierre SCHMIDT
MSD3@3ds.com
martinpierre.schmidt@insa-rouen.fr

Machine Learning : Definition

Components

Data	Model	Optimization
What is provided to the machine ?	What does the machine do with it?	How does the machine learns to do it better ?

Martin-Pierre SCHMIDT
MSD3@3ds.com
martinpierre.schmidt@insa-rouen.fr

Machine Learning : Definition

Data

- Training set
  - List of data examples
    - "Individuals" in a population
    - Points in N-dimensional space

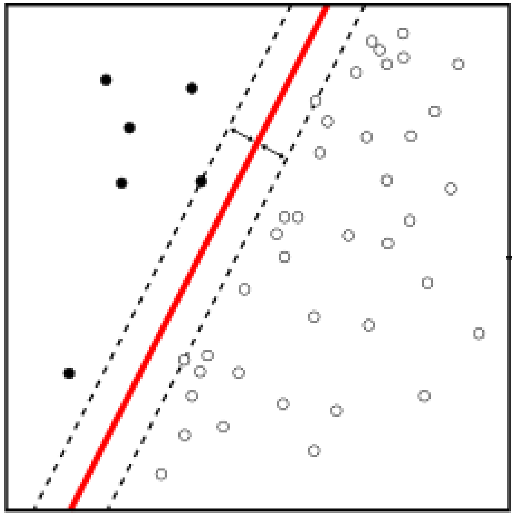
- Structured
  - Examples tend to have the same size, with parameters in the same order
- Exhibits some behavior
  - Assumption that the training set is a **representative sampling** of the N-Dimension Space **smooth underlying phenomenon**

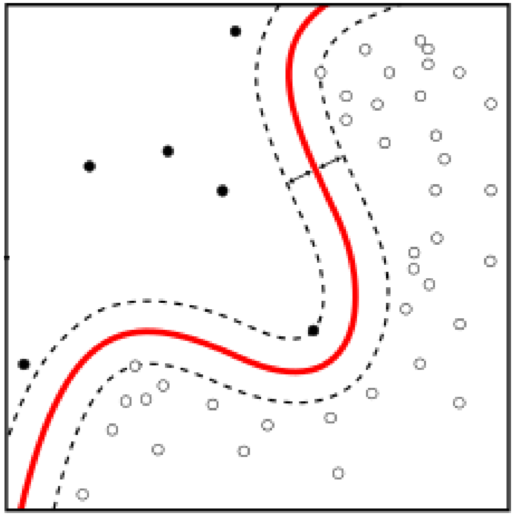
Martin-Pierre SCHMIDT
MSD3@3ds.com
martinpierre.schmidt@insa-rouen.fr

Machine Learning : Definition

Model

- Complexity
  - Order
    - Linear
    - Non-Linear
  - Input space
  - Output space
- Genericity
  - Specific
  - General





Martin-Pierre SCHMIDT
MSD3@3ds.com
martinpierre.schmidt@insa-rouen.fr

Machine Learning : Definition

Optimization

- Objective function / Cost function
  - Generally described as the **error of the model on the training data**
  - Also sometimes described by it's inverse as the "fitness" of a model (GA)

- Error minimization
  - Differentiable
    - Gradient descent
  - Non-differentiable
    - Simulated Annealing
    - Particle Swarm Optimization
    - Genetic Algorithm

Martin-Pierre SCHMIDT
MSD3@3ds.com
martinpierre.schmidt@insa-rouen.fr

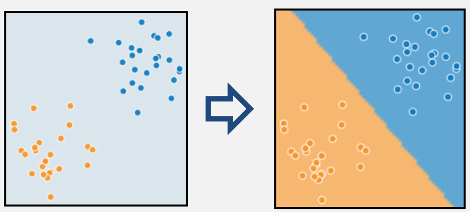
Machine Learning : Definition

Types of methods

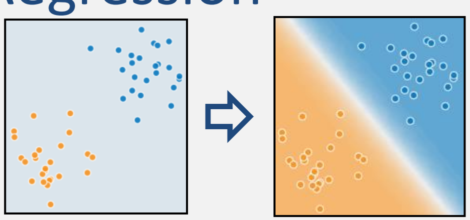
Poll: Perceptron

Supervised

Classification

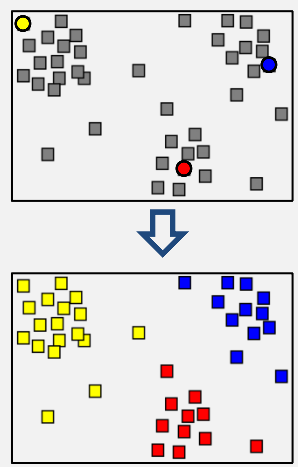


Regression



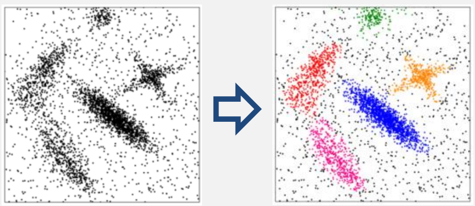
Semi-supervised

Propagation

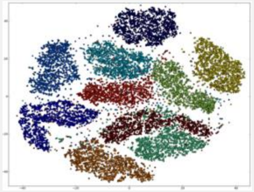


Unsupervised

Clustering



Visualization



Martin-Pierre SCHMIDT
MSD3@3ds.com
martinpierre.schmidt@insa-rouen.fr

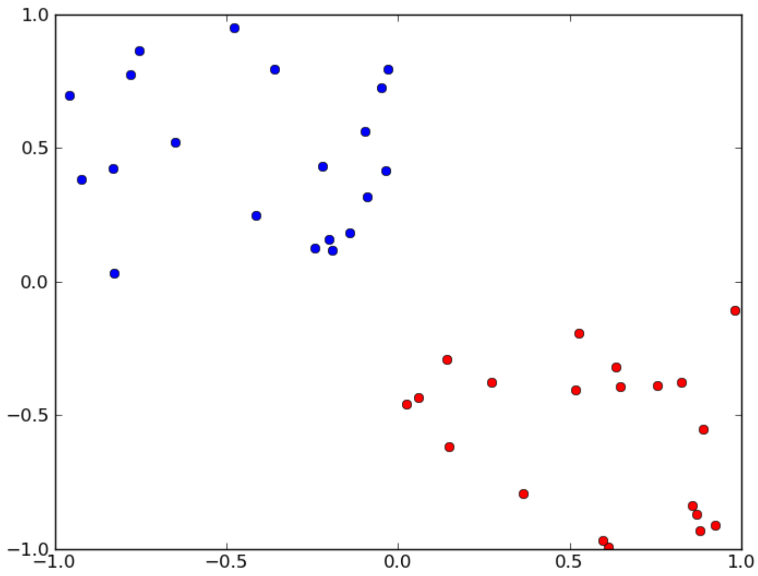
Perceptron

Martin-Pierre SCHMIDT
MSD3@3ds.com
martinpierre.schmidt@insa-rouen.fr

Perceptron

Overview

- Supervised
  - Classification
- Data
  - Feature vector
  - Label
- Model
  - Linear
- Optimization
  - Iterative method

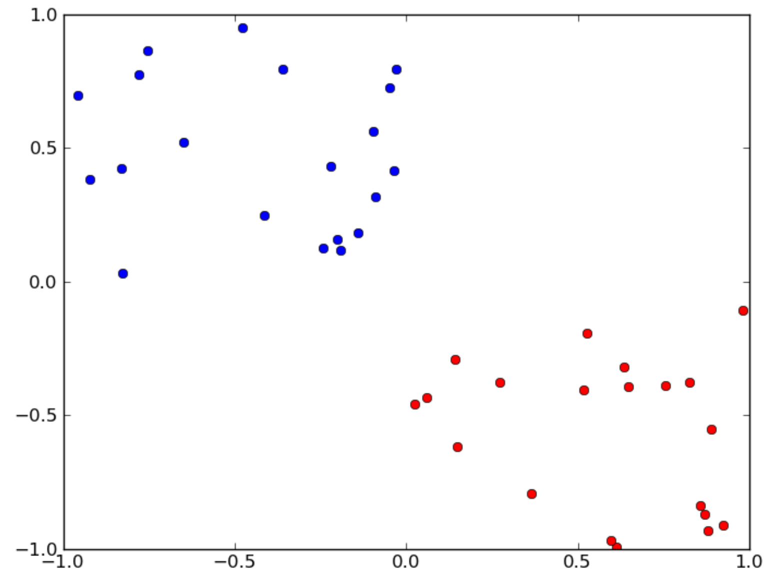


Martin-Pierre SCHMIDT
MSD3@3ds.com
martinpierre.schmidt@insa-rouen.fr

Perceptron

Data

- Input vector of size N
  - Each possible input can be represented as a point in N-dimensional space
- Output value
  - The output is a BLUE/RED answer based on the position of the point
  - The value is represented by “+1” or “-1”

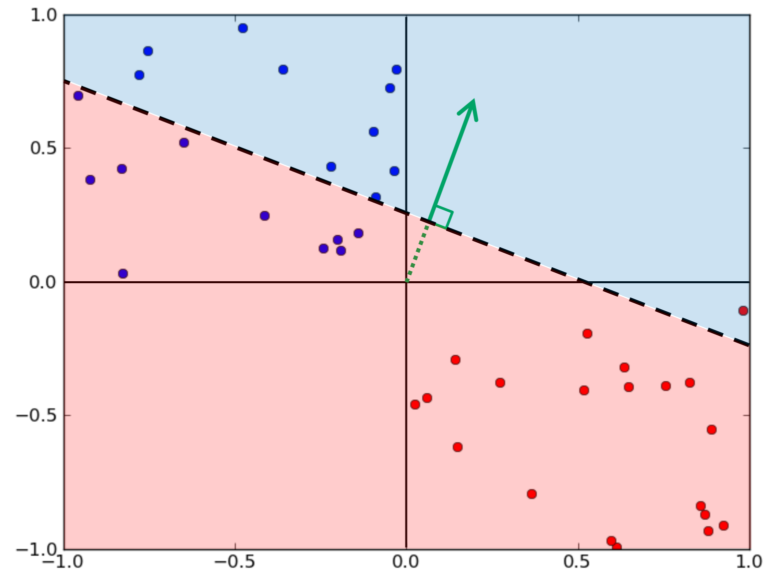


Martin-Pierre SCHMIDT
MSD3@3ds.com
martinpierre.schmidt@insa-rouen.fr

Perceptron

Model – Geometrical view

- The separating hyperplane needs N+1 parameters to be described in N-dimensional Space
  - A vector, normal to the **hyperplane**
    - Called the **Weights**
  - A translation magnitude from the origin and along that **normal vector**
    - Called the **Bias**

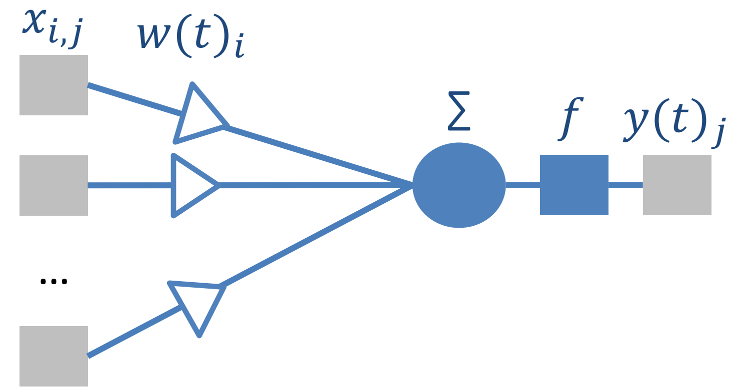


Martin-Pierre SCHMIDT
MSD3@3ds.com
martinpierre.schmidt@insa-rouen.fr

Perceptron

Model – Mathematical view

- Let $x_{i,j}$ be the i value of the j input sample point
- Let $w(\mathrm{t})_i$ be the i weight value at time t
- Let f be the activation function
- Let $y(\mathrm{t})_{j,}$ be the j output value at time
- Let d_j be the desired j output value



$$y(t)_j = f\left(\sum_{i=0}^{n} w(t)_i x_{i,j}\right)$$

Prediction function for the linear perceptron

Vector notation

Martin-Pierre SCHMIDT
MSD3@3ds.com
martinpierre.schmidt@insa-rouen.fr

Perceptron

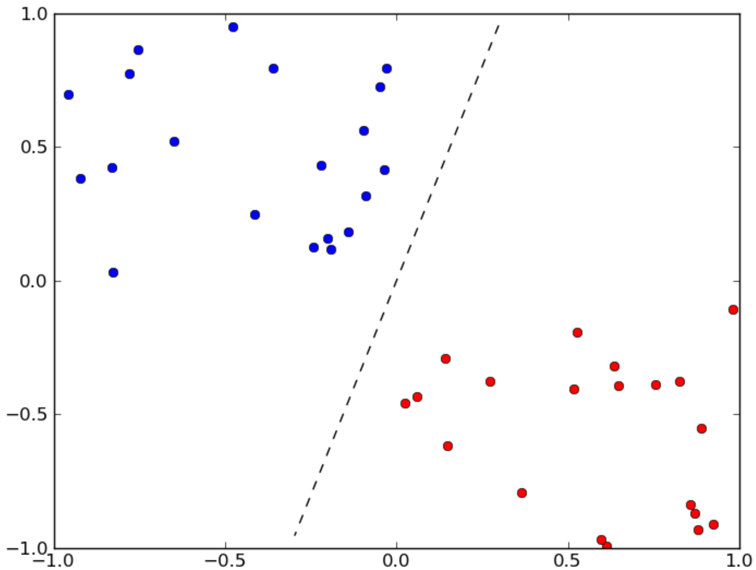
Optimization

- Let $x_{i,j}$ be the i value of the j input sample point
- Let $w(\mathrm{t})_i$ be the i weight value at time t
- Let f be the activation function
- Let $y(\mathrm{t})_{j,}$ be the j output value at time
- Let d_j be the desired j output value

Weight update intuition

Weight update rule for the linear perceptron

$$w(t+1)_i = w(t)_i + \left(d_j - y(t)_j\right)x_{i,j}$$



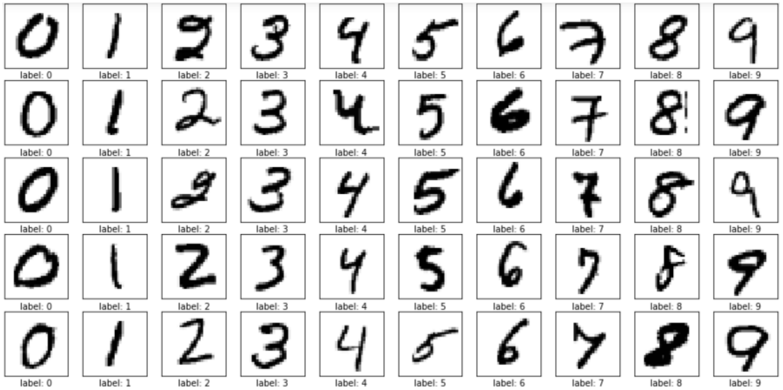
Martin-Pierre SCHMIDT
MSD3@3ds.com
martinpierre.schmidt@insa-rouen.fr

Perceptron

MATLAB Code

Poll: Success rate

- MNIST Database
  - Handwritten digits
  - 10 classes
  - 70 000 labeled images
    - 60 000 training images
    - 10 000 test images
  - High dimension feature vectors
    - 16x16 or 28x28 grayscale images



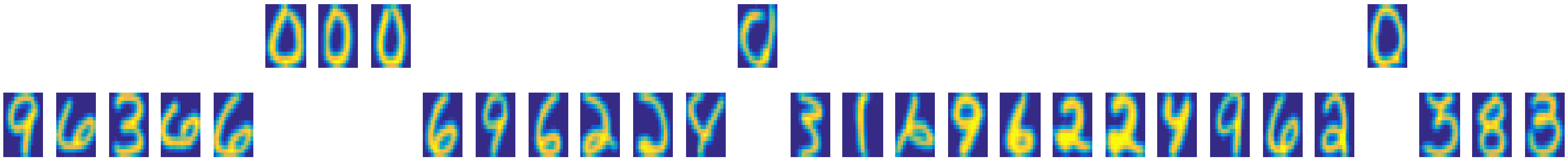
Examples of images in the MNIST database

- 2-classes
  - Detect “0” from “not 0”

Martin-Pierre SCHMIDT
MSD3@3ds.com
martinpierre.schmidt@insa-rouen.fr

Perceptron

MATLAB Code



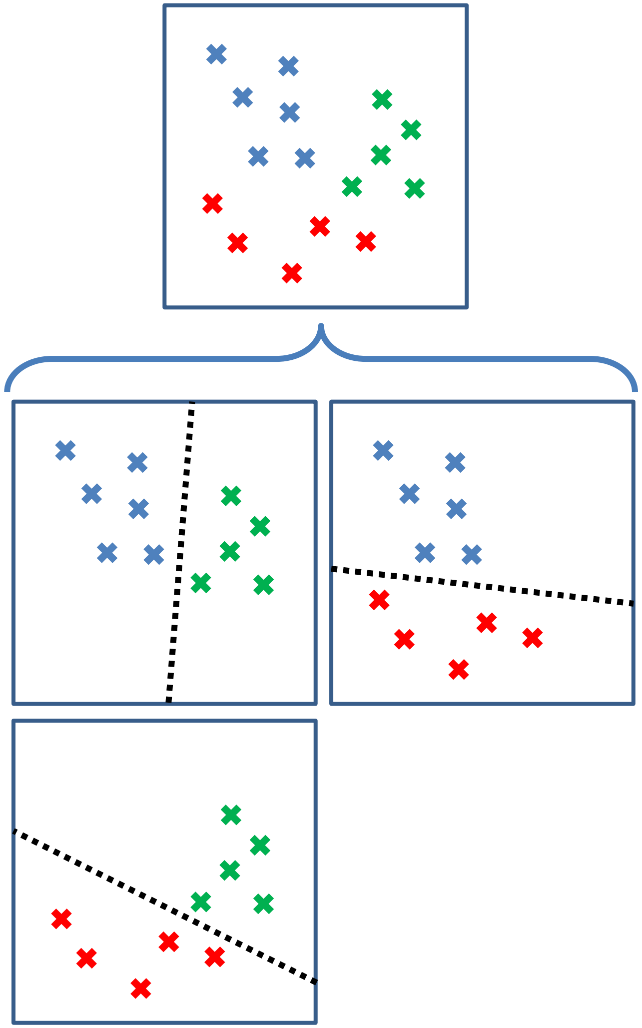
95.4% good predictions

Martin-Pierre SCHMIDT
MSD3@3ds.com
martinpierre.schmidt@insa-rouen.fr

Perceptron

Multi-class perceptron

- The perceptron is built to solve 2 classes problems
- Many different strategies can extend its scope
  - 1 vs 1
  - 1 vs All
  - Tournament

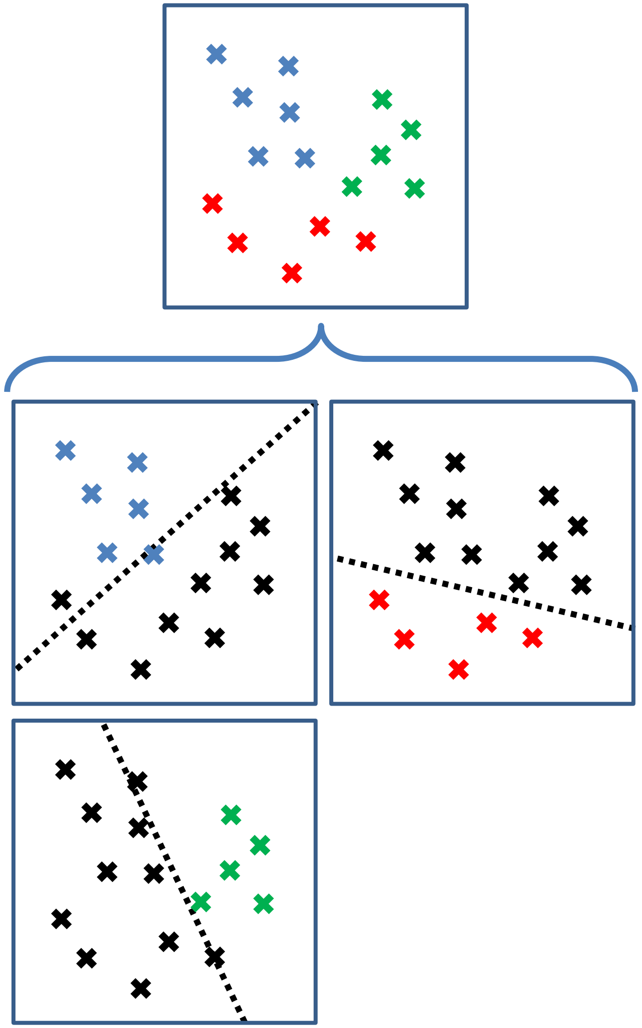


Martin-Pierre SCHMIDT
MSD3@3ds.com
martinpierre.schmidt@insa-rouen.fr

Perceptron

Multi-class perceptron

- The perceptron is built to solve 2 classes problems
- Many different strategies can extend its scope
  - 1 vs 1
  - 1 vs All
  - Tournament



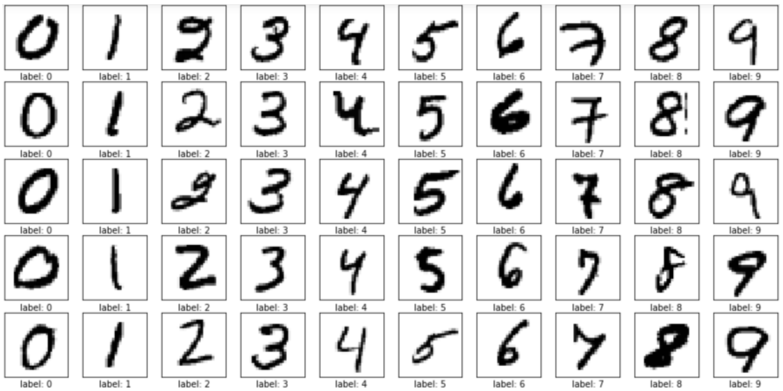
Martin-Pierre SCHMIDT
MSD3@3ds.com
martinpierre.schmidt@insa-rouen.fr

Perceptron

MATLAB Code

Poll: Success rate

- MNIST Database
  - Handwritten digits
  - 10 classes
  - 70 000 labeled images
    - 60 000 training images
    - 10 000 test images
  - High dimension feature vectors
    - 16x16 or 28x28 grayscale images



Examples of images in the MNIST database

- 10-classes
  - 1 vs all strategy

Martin-Pierre SCHMIDT
MSD3@3ds.com
martinpierre.schmidt@insa-rouen.fr

Perceptron

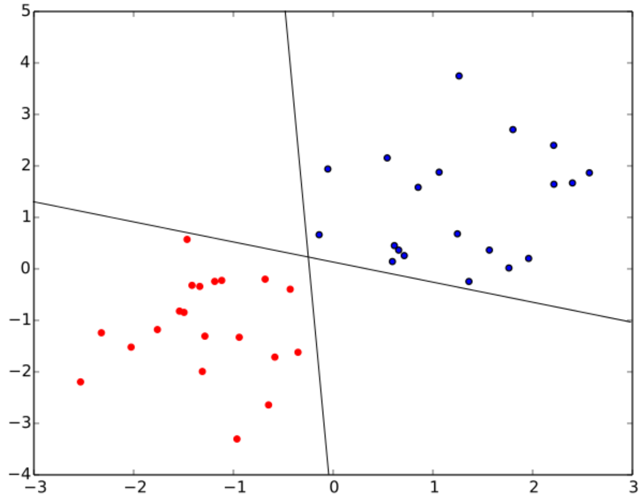
MATLAB Code

88.3% good predictions

Perceptron

Convergence

- Linearly Separable
  - Convergence in finite, upper bounded, number of steps
  - Dependency to training samples ordering
- Non-Linearly Separable
  - Generally suboptimal solution
  - Often oscillations
  - Sometimes, complete divergence

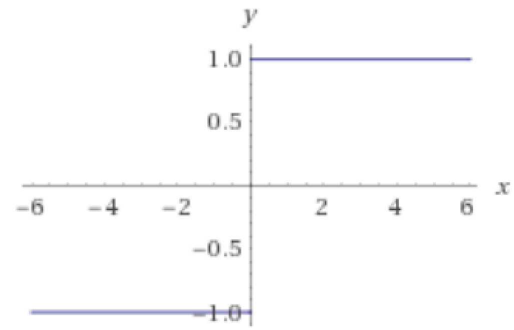


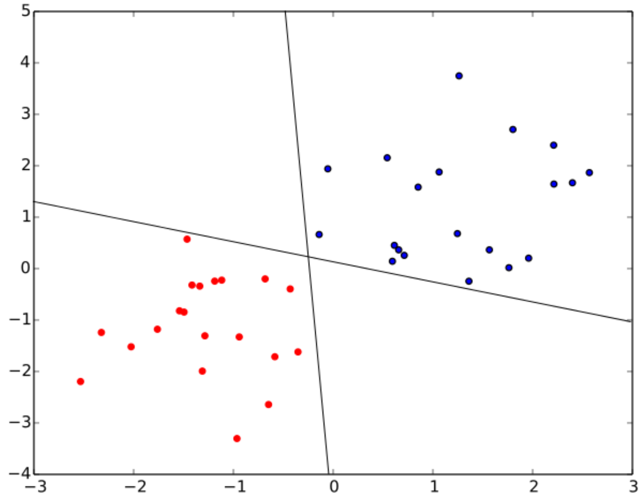
Martin-Pierre SCHMIDT
MSD3@3ds.com
martinpierre.schmidt@insa-rouen.fr

Perceptron

Activation function

- Heaviside step function as activation function
  - Multiple perfect solutions if linearly separable
  - Unstable if non linearly separable



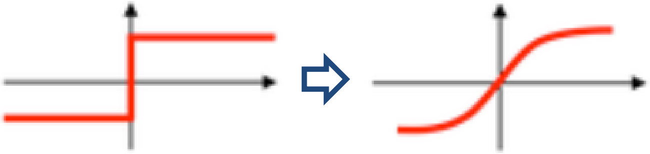


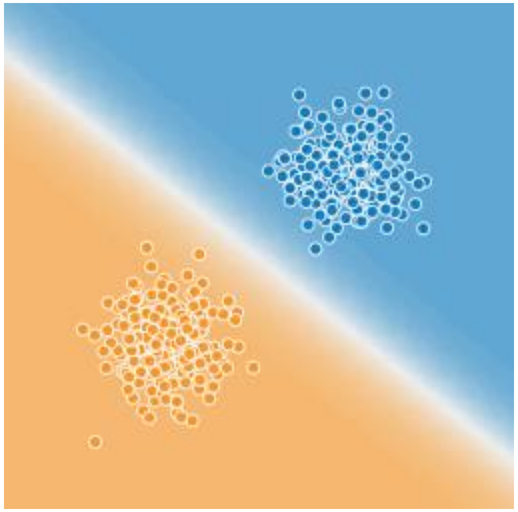
Martin-Pierre SCHMIDT
MSD3@3ds.com
martinpierre.schmidt@insa-rouen.fr

Perceptron

Activation function

- Smooth step function as activation function
  - Convergence toward a solution maximizing the **distance to misclassification**
  - Generally stable regardless of linear separability
  - The perceptron becomes analogous to **Logistic Regression**



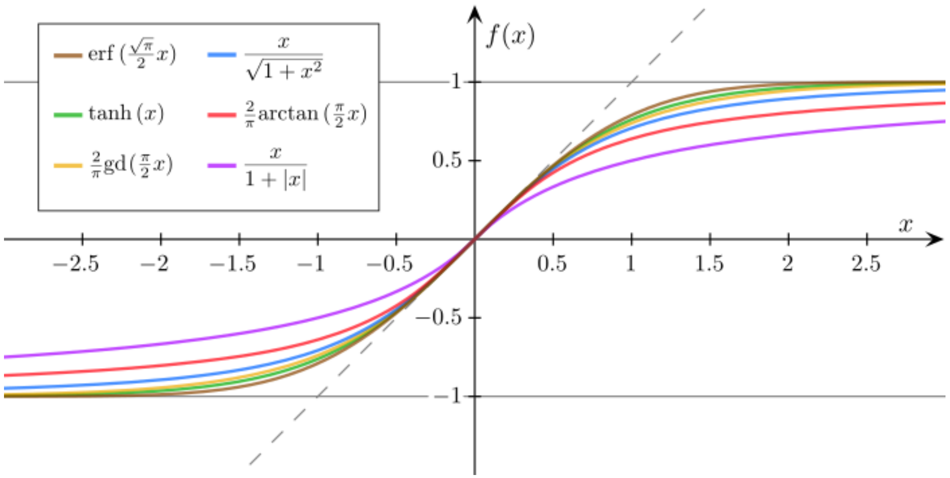


Martin-Pierre SCHMIDT
MSD3@3ds.com
martinpierre.schmidt@insa-rouen.fr

Perceptron

Activation function

- Multiple desirable properties:
  - Approximate identity near zero
  - Converges to +1 when x tends toward +∞
  - Converges to -1 when x tends toward -∞
- Many types of smooth activation functions



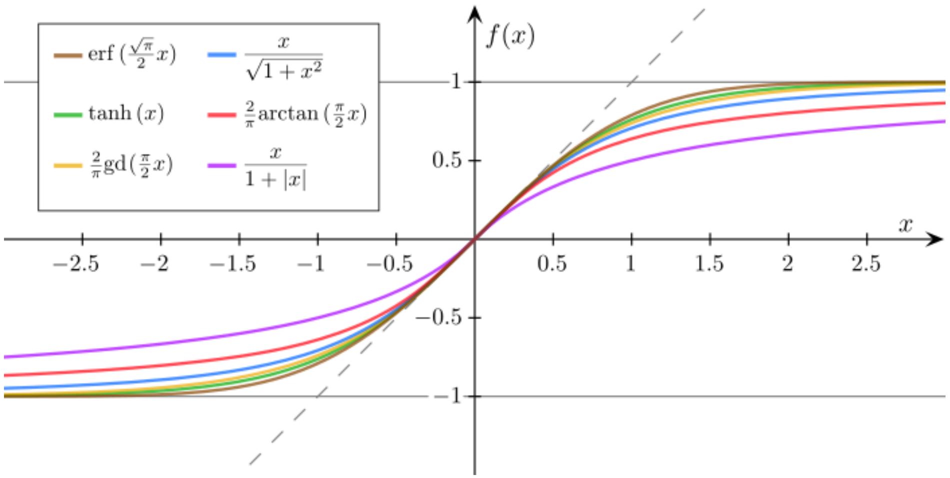
$$\tanh x = \frac{\sinh x}{\cosh x} = \frac{1 - e^{-2x}}{1 + e^{-2x}}$$

Martin-Pierre SCHMIDT
MSD3@3ds.com
martinpierre.schmidt@insa-rouen.fr

Perceptron

Activation function

- Multiple desirable properties:
  - Differentiable
  - Monotonous
  - Monotonous derivative
- Optimization now require proper partial derivatives



$$\tanh x = \frac{\sinh x}{\cosh x} = \frac{1-e^{-2x}}{1+e^{-2x}}$$

Martin-Pierre SCHMIDT
MSD3@3ds.com
martinpierre.schmidt@insa-rouen.fr

Perceptron

Let’s play

- playground.tensorflow.org
  - Tool Description
  - Simple test
  - Noise
  - Overfitting
  - Non Linearity

Martin-Pierre SCHMIDT
MSD3@3ds.com
martinpierre.schmidt@insa-rouen.fr

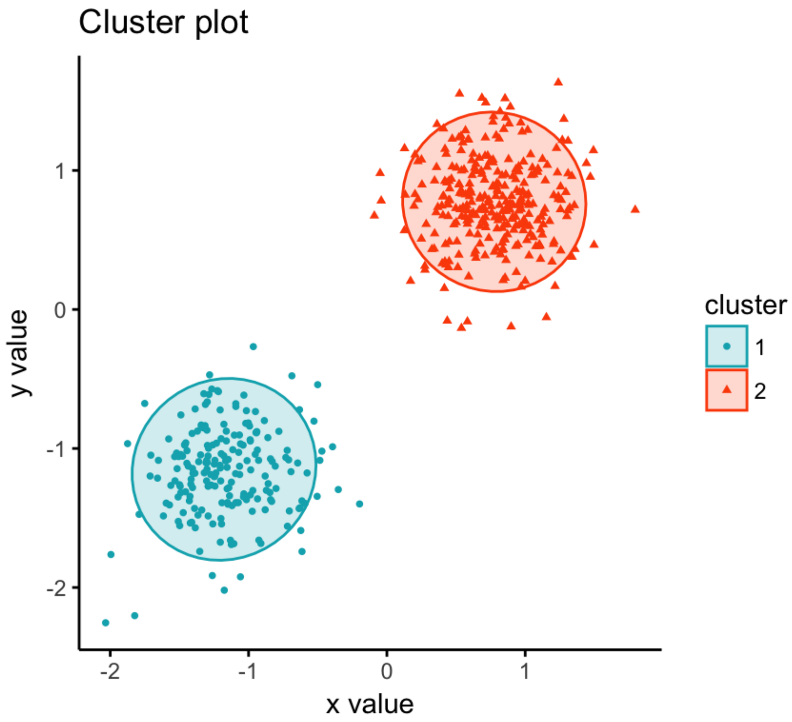
K-Means Clustering

Martin-Pierre SCHMIDT
MSD3@3ds.com
martinpierre.schmidt@insa-rouen.fr

K-Means Clustering

Overview

- Unsupervised
  - Clustering
- Data
  - Feature vectors
- Model
  - Centroid + Metric
- Optimization
  - Iterative method

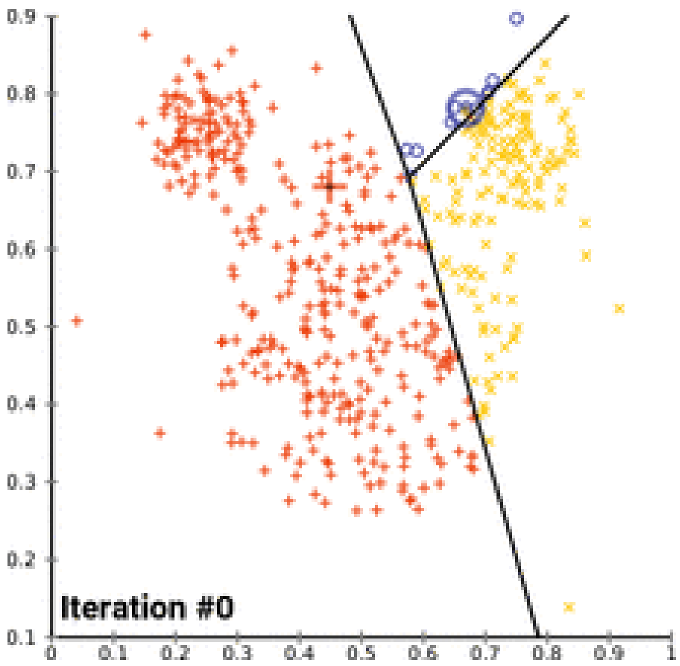


K-Means Clustering

Algorithm

Step by step algorithm

- ○ Initialization
  - Pick K random centroids
- ○ Loop
  - Label each sample with the closest centroid
  - Move each centroid to the mean of their labeled samples

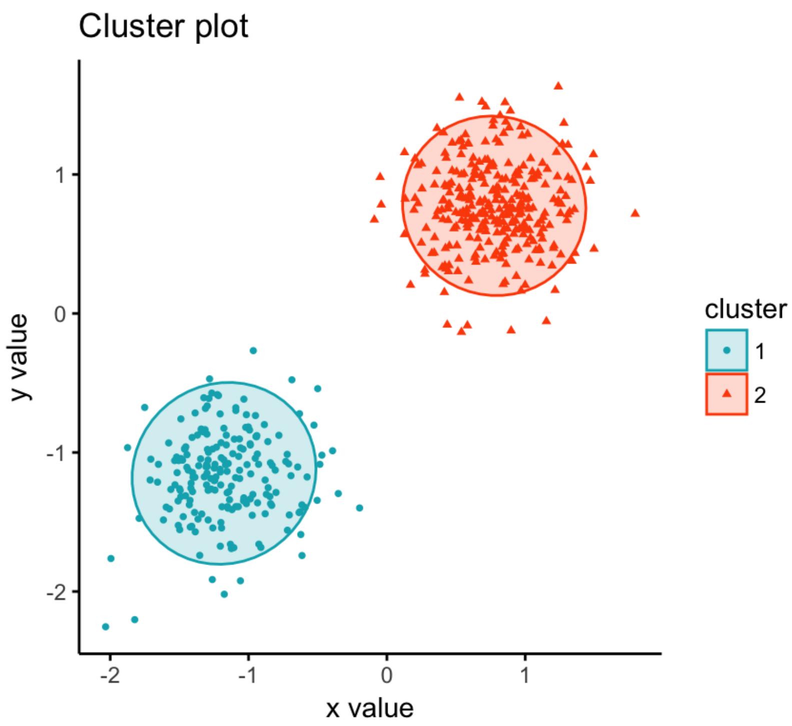


Martin-Pierre SCHMIDT
MSD3@3ds.com
martinpierre.schmidt@insa-rouen.fr

K-Means Clustering

Observations

- Minimize intra class variance
- Maximize inter class variance
- Assumes compact clusters
- Link to Voronoi diagram
- Highly dependent on the chosen metric



Martin-Pierre SCHMIDT
MSD3@3ds.com
martinpierre.schmidt@insa-rouen.fr

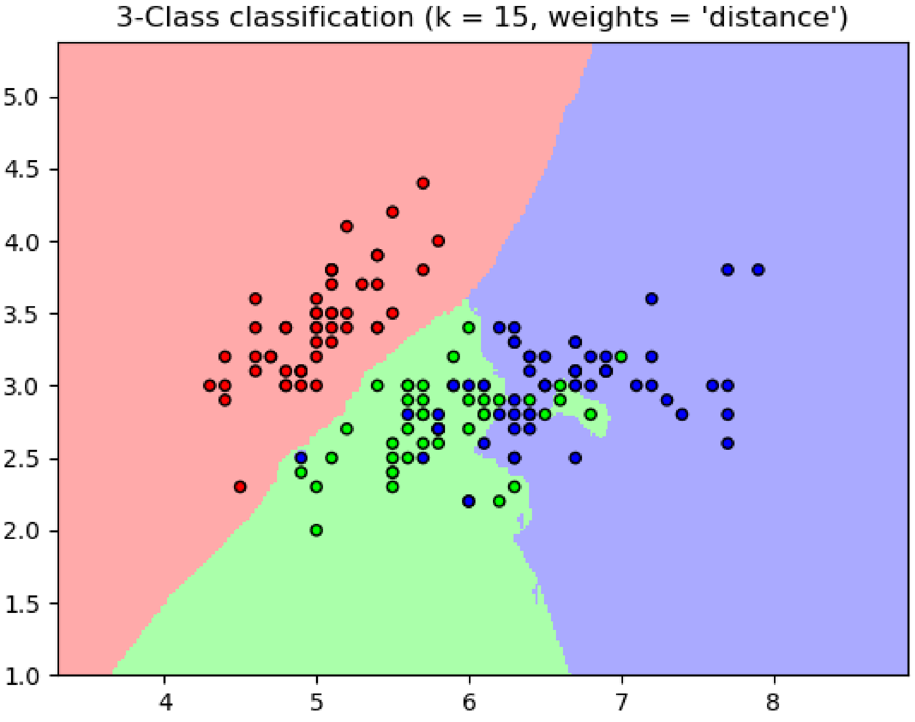
K-Nearest Neighbors

Martin-Pierre SCHMIDT
MSD3@3ds.com
martinpierre.schmidt@insa-rouen.fr

K-Nearest Neighbors

Overview

- Supervised
  - Classification
- Data
  - Feature vectors
  - Labels
- Model
  - Kernel based
- Optimization
  - None or Iterative method

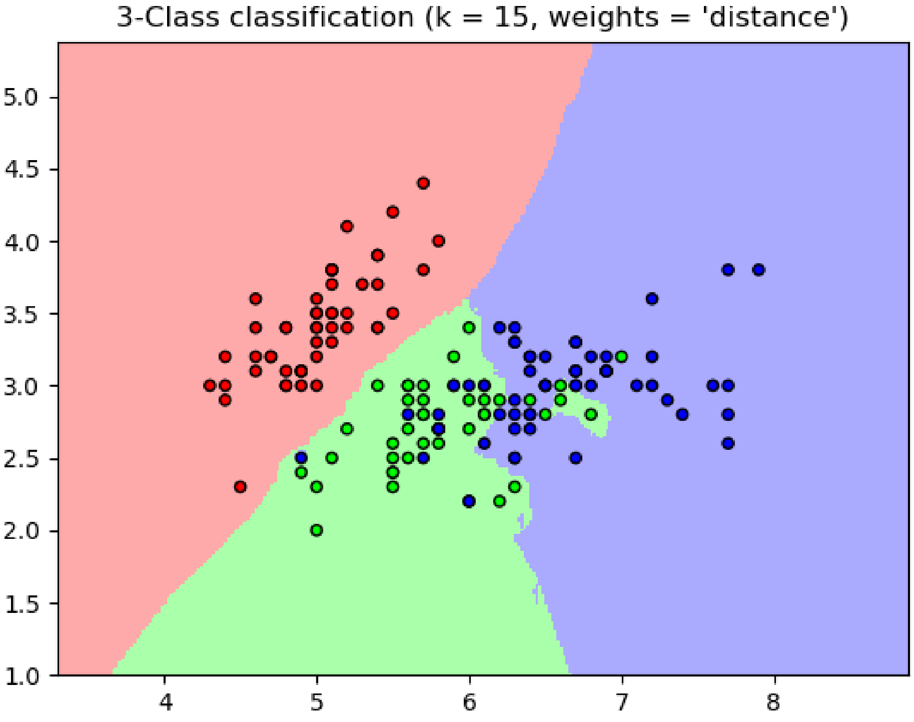


Martin-Pierre SCHMIDT
MSD3@3ds.com
martinpierre.schmidt@insa-rouen.fr

K-Nearest Neighbors

Algorithm

- For a given test feature vector
  - Find the k closest training feature vectors
  - Take the class most present in k neighbors
  - Optional
    - Weight based on neighbors distances



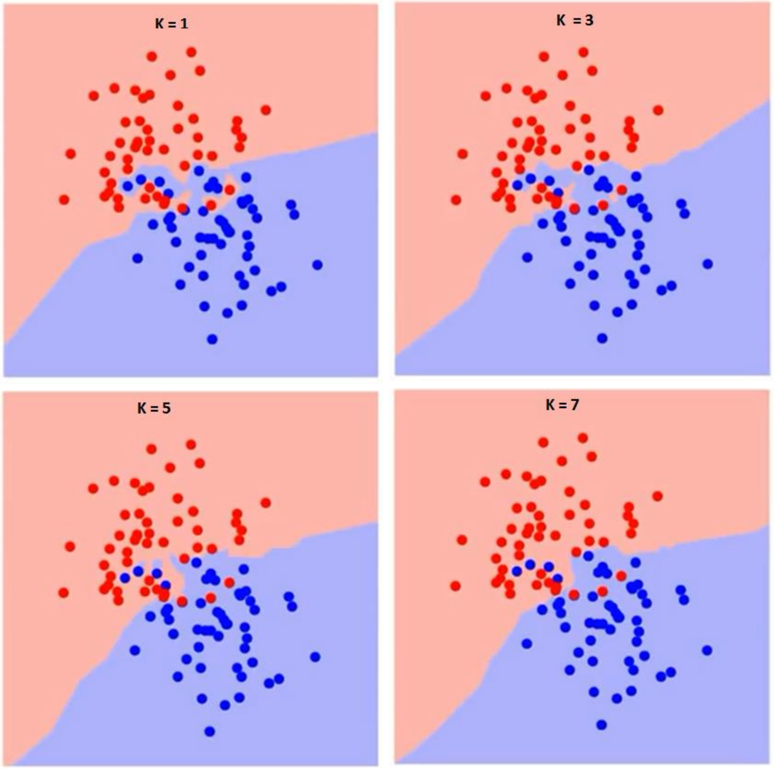
Martin-Pierre SCHMIDT
MSD3@3ds.com
martinpierre.schmidt@insa-rouen.fr

K-Nearest Neighbors

Observations

Poll: Success rate

- Can easily handle any number of classes
- Smoothness controlled by "k"
- Highly dependent on the chosen metric
- Costly evaluation
  - Can be optimized by making the dataset sparse



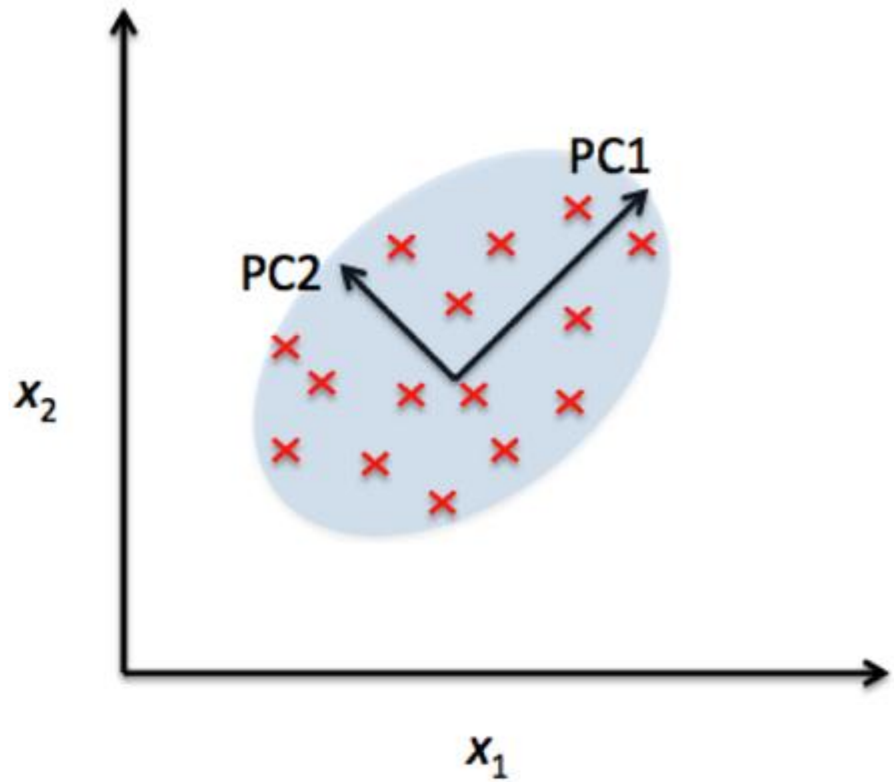
Martin-Pierre SCHMIDT
MSD3@3ds.com
martinpierre.schmidt@insa-rouen.fr

Principal Component Analysis

Principal Component Analysis

Overview

- Unsupervised
  - Visualization
- Data
  - Feature vectors
- Model
  - Orthogonal linear transformation
- Optimization
  - Analytical or Iterative method



Martin-Pierre SCHMIDT
MSD3@3ds.com
martinpierre.schmidt@insa-rouen.fr

Principal Component Analysis

Algorithm

- Let ***p*** be the dimension of the data
- Let ***n*** be the number of data points
- Let ***X*** be the ***p*** by ***n*** data matrix
  - ***X*** must have zero mean
- Let ***r*** be the resulting first principal component of ***X***

$\mathbf{r}$ = a random vector of length p
do c times:
    $\mathbf{s} = 0$ (a vector of length p)
    for each row $\mathbf{x} \in \mathbf{X}$
        $\mathbf{s} = \mathbf{s} + (\mathbf{x} \cdot \mathbf{r})\mathbf{x}$
    $\mathbf{r} = \dfrac{\mathbf{s}}{|\mathbf{s}|}$
return $\mathbf{r}$

Martin-Pierre SCHMIDT
MSD3@3ds.com
martinpierre.schmidt@insa-rouen.fr

Principal Component Analysis

Algorithm

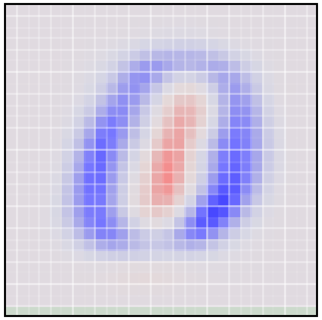
- Iterative algorithm is very efficient for high dimension data (when ***p*** is high)
  - Avoids computing the full covariance matrix
- Iterative algorithm is numerically instable after the first few principal components
  - Numerical error of each principal component computation is carried to the next
- This method is particularly well suited for **visualization** thought **dimensionality reduction**

Martin-Pierre SCHMIDT
MSD3@3ds.com
martinpierre.schmidt@insa-rouen.fr

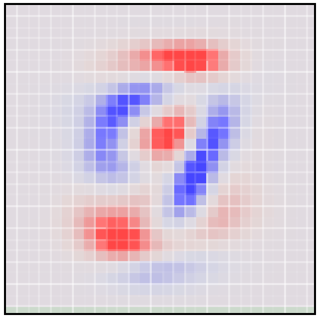
Principal Component Analysis

Example

- Each principal component is a vector of size equal to the feature vector dimension
- If the input feature vectors are images, each principal component can also be seen as an image



First principal component in the MNIST Database



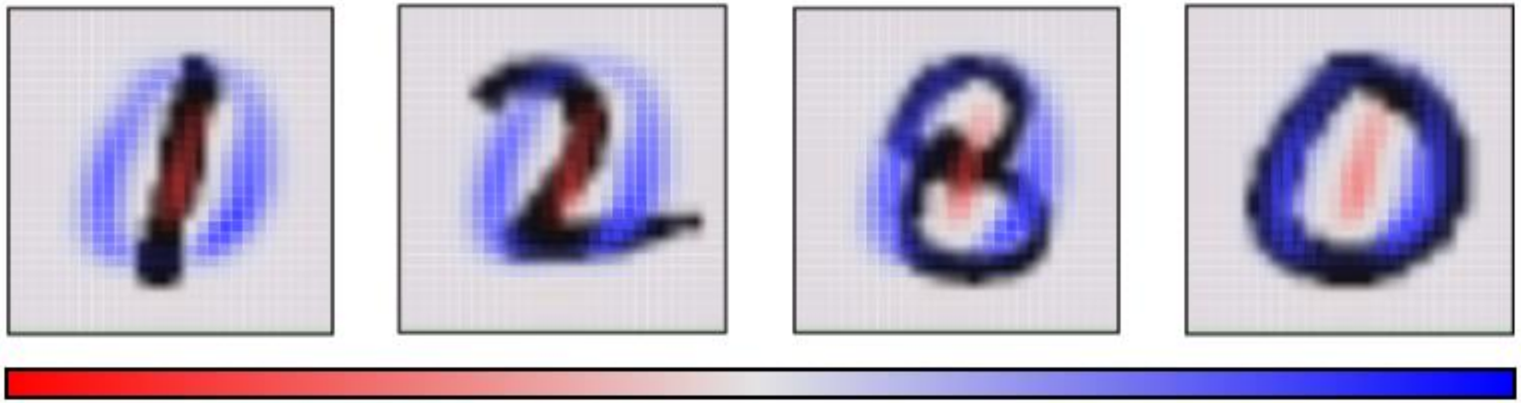
Second principal component in the MNIST Database

2D example and color coding

Martin-Pierre SCHMIDT
MSD3@3ds.com
martinpierre.schmidt@insa-rouen.fr

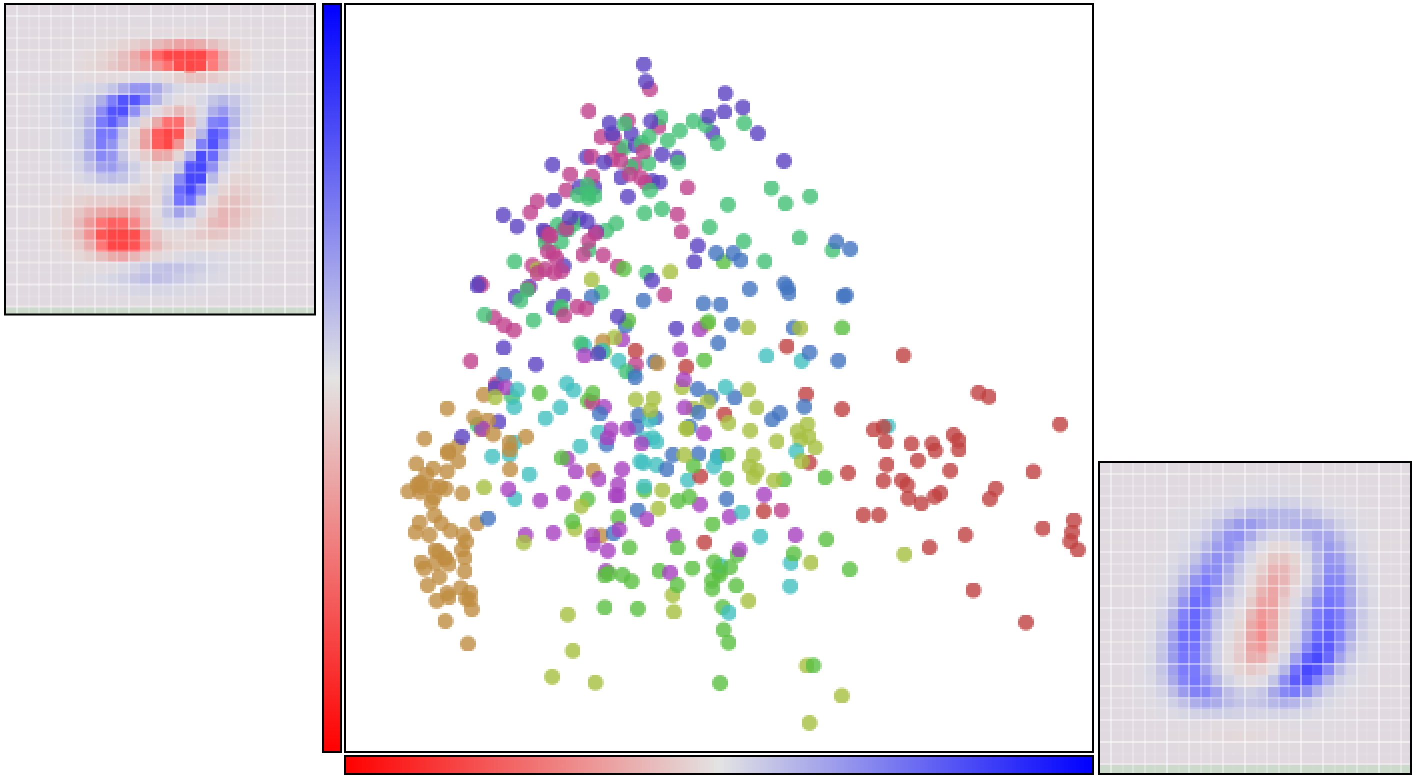
Principal Component Analysis

Example



Principal Component Analysis

Example



Principal Component Analysis

Example

- http://colah.github.io/posts/2014-10-Visualizing-MNIST/

Martin-Pierre SCHMIDT
MSD3@3ds.com
martinpierre.schmidt@insa-rouen.fr

Principal Component Analysis

Implementation for images

- Store all the images in a single matrix ***X***
  - Each row corresponds to the pixel values of an image
  - The labels are NOT used
- Transform ***X*** to give it zero mean
- Repeat 2 or 3 times :
  - Run the simple iterative algorithm to extract ***PC*** the first principal component of ***X***
  - Update ***X*** by projecting each data point onto de hyperplane defined by ***PC***
  - Store the value of ***PC*** before computing the next principal component

Projection ?

- Visualize the resulting point cloud in 2D or 3D
  - Color code the points based on their label to see if some structure emerges

Martin-Pierre SCHMIDT
MSD3@3ds.com
martinpierre.schmidt@insa-rouen.fr

Kernel Perceptron

Martin-Pierre SCHMIDT
MSD3@3ds.com
martinpierre.schmidt@insa-rouen.fr

Kernel Perceptron

Overview

- Variant of traditional perceptron
  - Described in 1960
  - Capable of learning (almost arbitrary) nonlinear phenomena
- Type of instance-based learner
  - No explicit generalization, no fixed set of parameters to optimize
  - Hypothesis built directly by **combining training instances**
  - Can easily be extended with additional data
  - Example
    - K-nearest neighbors
- First of the **Kernel Machines**

Martin-Pierre SCHMIDT
MSD3@3ds.com
martinpierre.schmidt@insa-rouen.fr

Kernel Perceptron

Kernel

- The model is made of input instances
  - Good adaptability to different problems
  - Often sparse
- The prediction emerges from the **weighted sum of similarities** with instances from the training set
  - The similarity function is called **kernel** and implies a **metric to compare instances**
  - A vector often called **alpha** associates a “relevance score” to each instance of the training set
    - This defines the subset of training example later used to classify new data
- Other examples of kernel machines
  - Support Vector Machines (SVM)
  - Principal Component Analysis (PCA)
  - Gaussian Processes

Martin-Pierre SCHMIDT
MSD3@3ds.com
martinpierre.schmidt@insa-rouen.fr

Kernel Perceptron

Algorithm derivation

To derive a kernelized version of the perceptron algorithm, we must first formulate it in dual form, starting from the observation that the weight vector $\mathbf{w}$ can be expressed as a linear combination of the n training samples. The equation for the weight vector is

$$\mathbf{w} = \sum_i^n \alpha_i y_i \mathbf{x}_i$$

where α_i is the number of times $\mathbf{x}_i$ was misclassified, forcing an update $\mathbf{w} \leftarrow \mathbf{w} + y_i \, \mathbf{x}_i$. Using this result, we can formulate the dual perceptron algorithm, which loops through the samples as before, making predictions, but instead of storing and updating a weight vector $\mathbf{w}$, it updates a "mistake counter" vector $\boldsymbol{\alpha}$. We must also rewrite the prediction formula to get rid of $\mathbf{w}$:

$$\begin{aligned}\hat{y} &= \operatorname{sgn}(\mathbf{w}^\top \mathbf{x}) \\ &= \operatorname{sgn}\left(\sum_i^n \alpha_i y_i \mathbf{x}_i\right)^\top \mathbf{x} \\ &= \operatorname{sgn}\sum_i^n \alpha_i y_i (\mathbf{x}_i \cdot \mathbf{x})\end{aligned}$$

Martin-Pierre SCHMIDT
MSD3@3ds.com
martinpierre.schmidt@insa-rouen.fr

Kernel Perceptron

Algorithm derivation

Plugging these two equations into the training loop turn it into the *dual perceptron* algorithm.

Finally, we can replace the dot product in the dual perceptron by an arbitrary kernel function, to get the effect of a feature map Φ without computing $\Phi(\mathbf{x})$ explicitly for any samples. Doing this yields the kernel perceptron algorithm:[4]

Initialize $\boldsymbol{\alpha}$ to an all-zeros vector of length n, the number of training samples.

For some fixed number of iterations, or until some stopping criterion is met:

For each training example $\mathbf{x}_j, y_j$:

Let $\hat{y} = \operatorname{sgn} \sum_i^n \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}_j)$

If $\hat{y} \neq y_j$, perform an update by incrementing the mistake counter:

$\alpha_j \leftarrow \alpha_j + 1$

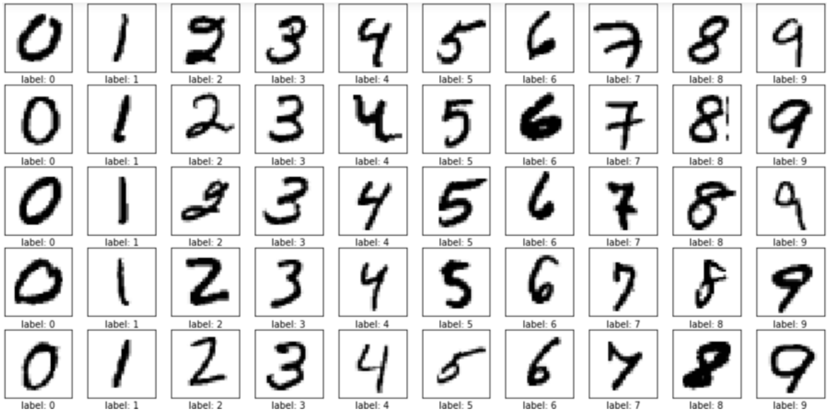
Martin-Pierre SCHMIDT
MSD3@3ds.com
martinpierre.schmidt@insa-rouen.fr

Kernel Perceptron

MATLAB Code

Poll: Success rate

- MNIST Database
  - Handwritten digits
  - 10 classes
  - 70 000 labeled images
    - 60 000 training images
    - 10 000 test images
  - High dimension feature vectors
    - 16x16 or 28x28 grayscale images
- 2-classes
  - Detect “0” from “not 0”
- 10-classes
  - 1 vs all strategy

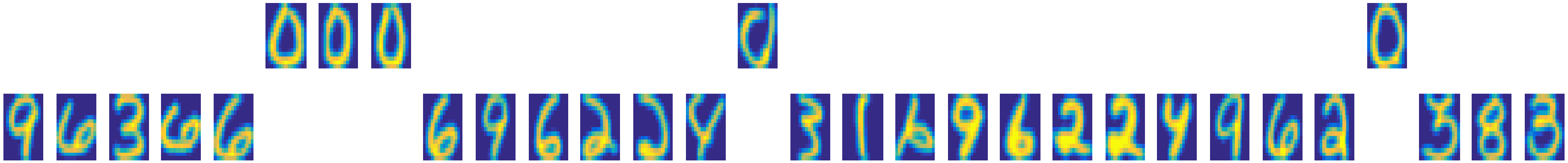


Examples of images in the MNIST database

Martin-Pierre SCHMIDT
MSD3@3ds.com
martinpierre.schmidt@insa-rouen.fr

Kernel Perceptron

MATLAB Code



98.7% good predictions

Martin-Pierre SCHMIDT
MSD3@3ds.com
martinpierre.schmidt@insa-rouen.fr

Kernel Perceptron

MATLAB Code

93.4% good predictions

Martin-Pierre SCHMIDT
MSD3@3ds.com
martinpierre.schmidt@insa-rouen.fr

AdaBoost

Martin-Pierre SCHMIDT
MSD3@3ds.com
martinpierre.schmidt@insa-rouen.fr

AdaBoost

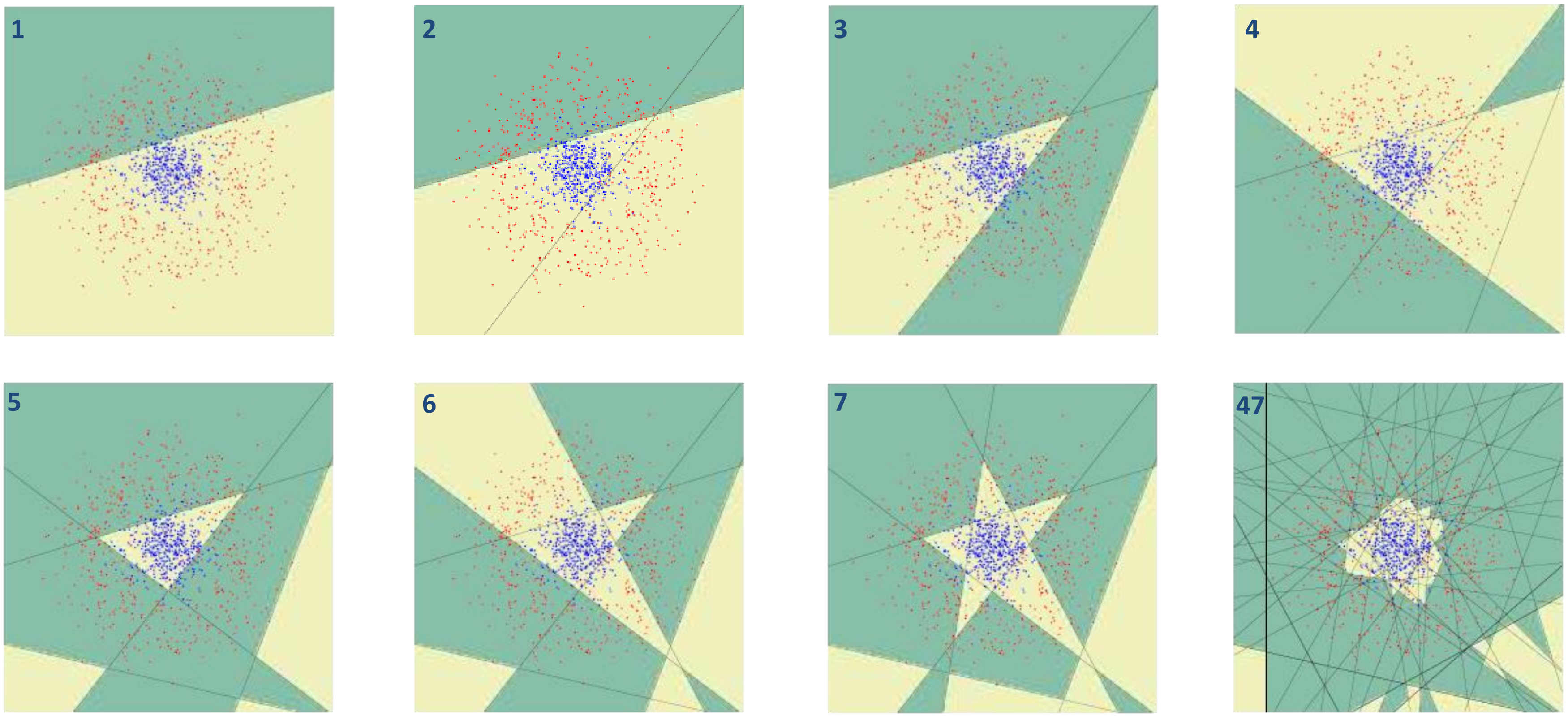
Overview

- Meta-Algorithm: 2003 Gödel Prize for Schapire and Freund (1996)
- **Ada**ptive **Boost**ing
  - Boosting
    - Combination of multiple **weak classifiers** to form one **strong classifier**
  - Adaptive
    - Favor misclassified instances in subsequent weak learners
    - Focus on “hard to classify” cases
- Many advantages
  - Fairly robust to overfitting
  - Low parameter count
  - High versatility
  - Fast and lightweight

Martin-Pierre SCHMIDT
MSD3@3ds.com
martinpierre.schmidt@insa-rouen.fr

AdaBoost

Step by Step



Martin-Pierre SCHMIDT
MSD3@3ds.com
martinpierre.schmidt@insa-rouen.fr

AdaBoost

Algorithm

For t in $1 \ldots T$:

- Choose $h_t(x)$:
  - Find weak learner $h_t(x)$ that minimizes ϵ_t, the weighted sum error for misclassified points $\epsilon_t = \sum_{\substack{i=1 \\ h_t(x_i) \neq y_i}}^{n} w_{i,t}$
  - Choose $\alpha_t = \frac{1}{2} \ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$
- Add to ensemble:
  - $F_t(x) = F_{t-1}(x) + \alpha_t h_t(x)$
- Update weights:
  - $w_{i,t+1} = w_{i,t} e^{-y_i \alpha_t h_t(x_i)}$ for all i
  - Renormalize $w_{i,t+1}$ such that $\sum_i w_{i,t+1} = 1$

With:

- Samples $x_1 \ldots x_n$
- Desired outputs $y_1 \ldots y_n, y \in \{-1, 1\}$
- Initial weights $w_{1,1} \ldots w_{n,1}$ set to $\frac{1}{n}$
- Error function $E(f(x), y, i) = e^{-y_i f(x_i)}$
- Weak learners $h: x \to [-1, 1]$

Martin-Pierre SCHMIDT
MSD3@3ds.com
martinpierre.schmidt@insa-rouen.fr

AdaBoost

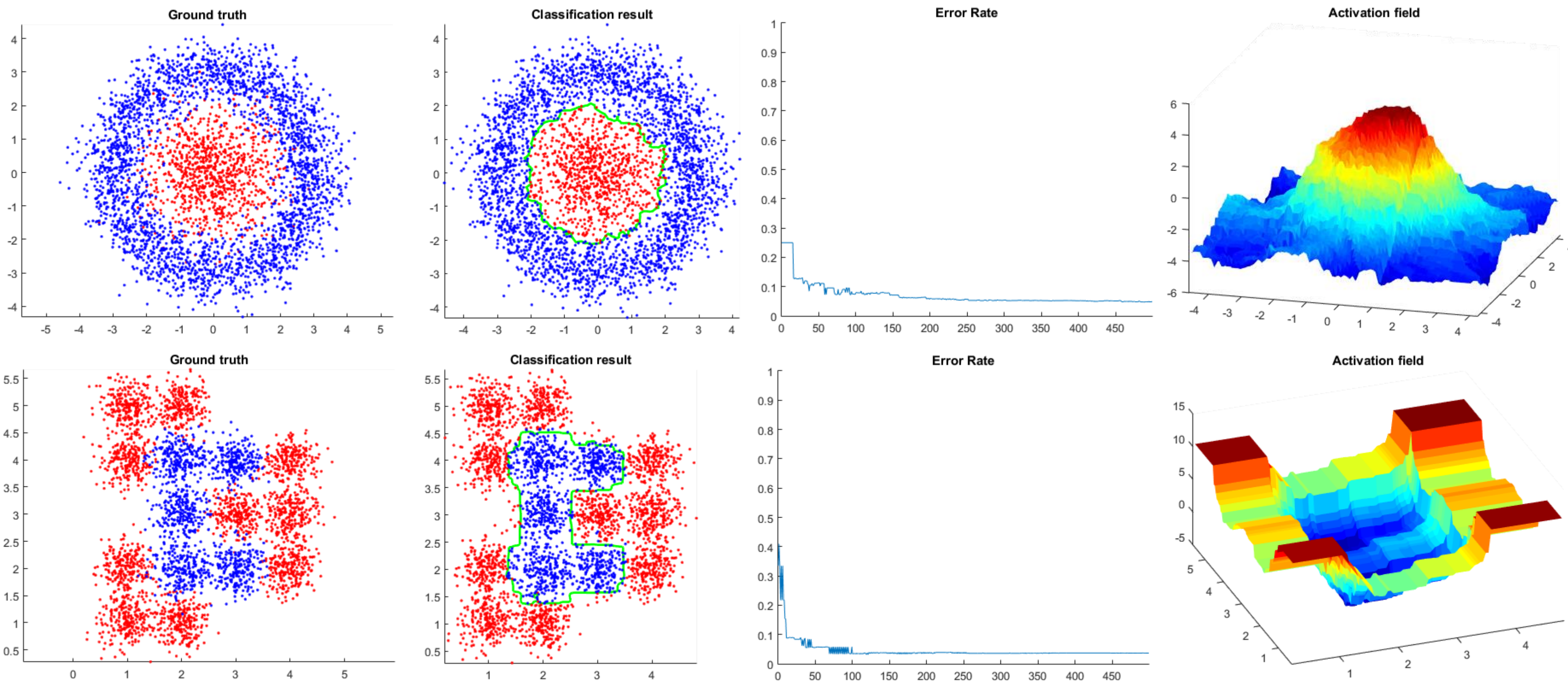
MATLAB Code

- Problem generation
  - 2D point distribution
  - Various patterns
- Weak classifier selection
  - Parallel separator
  - Linear perceptron
- Strong classifier construction
  - AdaBoost implementation
  - Weak learners stack + weight vector

Martin-Pierre SCHMIDT
MSD3@3ds.com
martinpierre.schmidt@insa-rouen.fr

AdaBoost

MATLAB Code



Martin-Pierre SCHMIDT
MSD3@3ds.com
martinpierre.schmidt@insa-rouen.fr

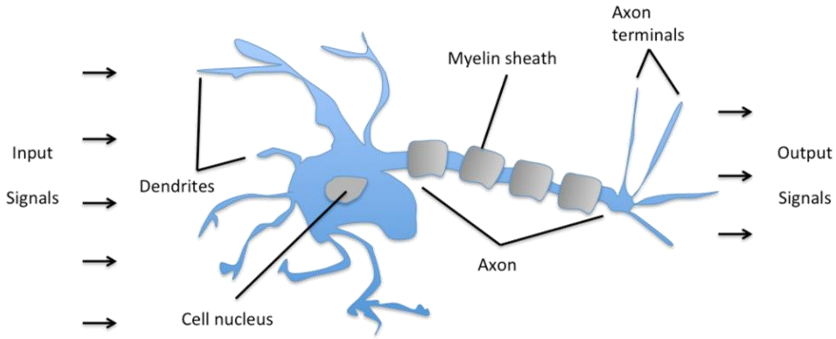
History of ANNs

Martin-Pierre SCHMIDT
MSD3@3ds.com
martinpierre.schmidt@insa-rouen.fr

History of ANNs

Origin

- 1943 – First artificial neuron
  - Neurophysiologist Warren McCulloch
  - Mathematician Walter Pitts
  - Paper on the way neurons might work
  - Modeled by electrical circuit



Martin-Pierre SCHMIDT
MSD3@3ds.com
martinpierre.schmidt@insa-rouen.fr

History of ANNs

Origin

- 1949 – First insight into neuron learning
  - Donald Hebb - The Organization of Behavior
  - Neurons have different signal “strengths”
  - Connections increase with repeated use
  - Neurons which fire together strengthen together

Neuron groups

Martin-Pierre SCHMIDT
MSD3@3ds.com
martinpierre.schmidt@insa-rouen.fr

History of ANNs

Origin

- 1950 – First Artificial Neural Network ?
  - IBM Research Lab
  - Nathanial Rochester
  - Computer simulation
  - Fail

Martin-Pierre SCHMIDT
MSD3@3ds.com
martinpierre.schmidt@insa-rouen.fr

History of ANNs

Origin

- 1959 – First Artificial Neural Network !
  - Stanford
  - Bernard Widrow and Marcian Hoff
  - MADALINE (**M**ultiple **ADA**ptive **LIN**ear **E**lements)
  - Adaptive filter that reduces echo on phone line

Martin-Pierre SCHMIDT
MSD3@3ds.com
martinpierre.schmidt@insa-rouen.fr

History of ANNs

Origin

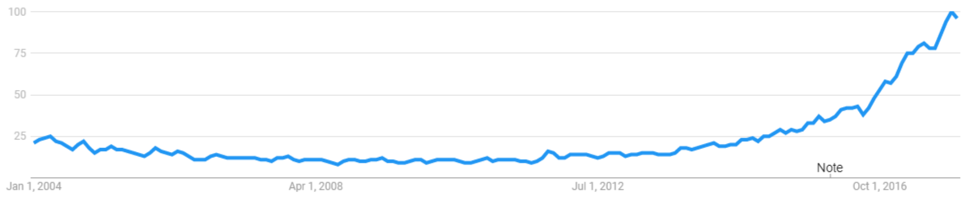
- 1986 – **Back-Propagation** networks
  - 3 independent research groups
  - Distribution of error throughout the network
  - Multi layer networks
    - Slower than previous implementations
    - Better generalization capabilities

Martin-Pierre SCHMIDT
MSD3@3ds.com
martinpierre.schmidt@insa-rouen.fr

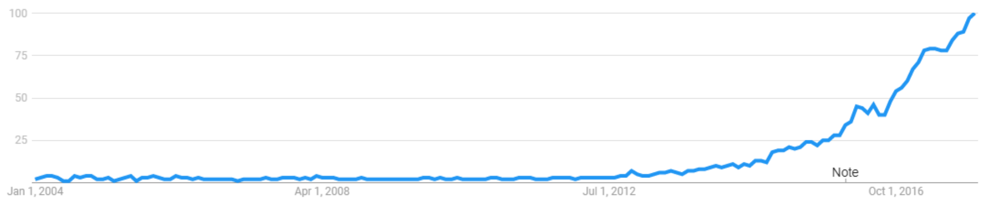
History of ANNs

Popularization

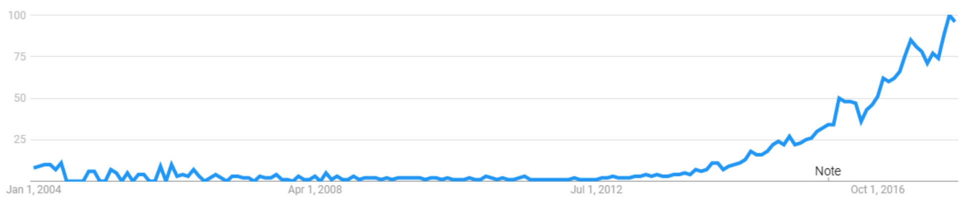
- “Machine Learning”



- “Convolutional neural network”



- “Deep Learning”



Martin-Pierre SCHMIDT
MSD3@3ds.com
martinpierre.schmidt@insa-rouen.fr

History of ANNs

Popularization

- Factors
  - Availability of Data
    - Active and passive data gathering
  - Computing Power
    - CPU, GPU, TPU
    - Parallel Computation libraries
  - Wide applicability

Martin-Pierre SCHMIDT
MSD3@3ds.com
martinpierre.schmidt@insa-rouen.fr

History of ANNs

Popularization

- Actors
  - Research groups
    - Universities
    - Research Institutions
  - Large Companies
    - Software
    - Hardware
  - Start-Ups

Martin-Pierre SCHMIDT
MSD3@3ds.com
martinpierre.schmidt@insa-rouen.fr

Deep Learning : Definition

Martin-Pierre SCHMIDT
MSD3@3ds.com
martinpierre.schmidt@insa-rouen.fr

Deep Learning : Definition

Definition

- Data
  - Supervised or Unsupervised

- Model
  - Cascade layers of nonlinear processing units for feature extraction
  - Multiple levels of representation and abstractions
  - Hierarchy of concepts

- Optimization
  - Backpropagation
  - Gradient descent

Martin-Pierre SCHMIDT
MSD3@3ds.com
martinpierre.schmidt@insa-rouen.fr

Deep Learning : Definition

Applications overview

- Automatic speech recognition
- Image recognition
- Visual art processing
- Natural language processing
- Drug discovery and toxicology
- Customer relationship management
- Recommendation systems
- Bioinformatics
- Mobile advertising
- Image restoration

Martin-Pierre SCHMIDT
MSD3@3ds.com
martinpierre.schmidt@insa-rouen.fr

Artificial Neural Network

Martin-Pierre SCHMIDT
MSD3@3ds.com
martinpierre.schmidt@insa-rouen.fr

Artificial Neural Network

Overview

- Many variants of the same basic concept
  - Multilayer perceptron
  - Convolutional neural network
  - Memory networks
  - Cycle networks
  - Adversarial networks
  - Auto-encoders
  - …
- Simple architecture
  - Allows for efficient parallelization
- Implemented in libraries for most programming languages
- High abstraction power
  - Universal function approximator

Martin-Pierre SCHMIDT
MSD3@3ds.com
martinpierre.schmidt@insa-rouen.fr

Artificial Neural Network

Criticism

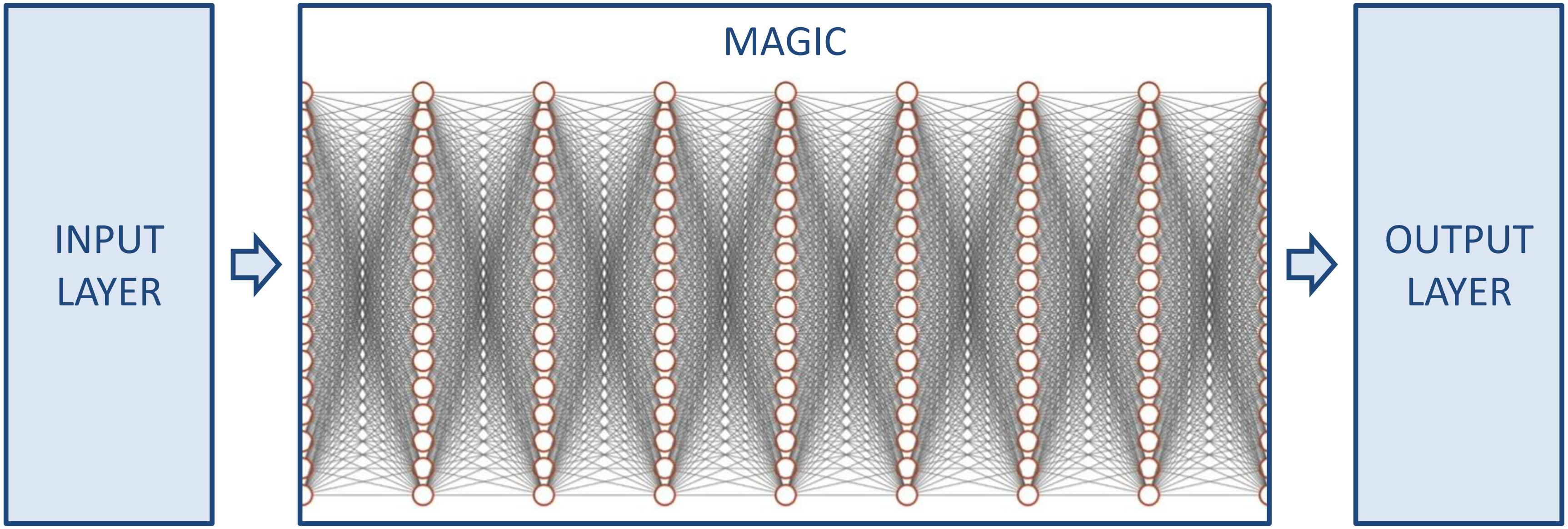
- Lack of theoretical understanding
  - Capacity
  - Convergence
  - Architecture
  - Overfitting
  - Training
- Un-rigorous scientific community
  - Trend effect
  - Anthropomorphism
  - Easy publications
- Hardware
  - Specific needs
  - Power consumption

Martin-Pierre SCHMIDT
MSD3@3ds.com
martinpierre.schmidt@insa-rouen.fr

Artificial Neural Network

Architecture

Recap architecture
Perceptron & Adaboost

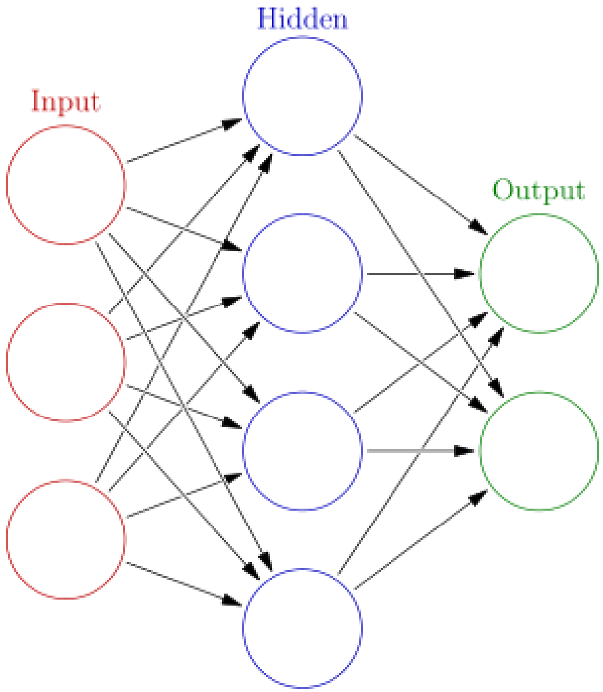


Martin-Pierre SCHMIDT
MSD3@3ds.com
martinpierre.schmidt@insa-rouen.fr

Artificial Neural Network

Notation

- Value of neuron k in layer L : $\mathbf{a}_{\mathbf{k}}^{(L)}$
- Value of bias for neuron k in layer L : $\mathbf{b}_{\mathbf{k}}^{(L)}$
- Value of weight going from neuron k in layer (L-1) to neuron j in layer L : $\mathbf{w}_{\mathbf{j,k}}^{(L)}$
- Weighted sum with bias for neuron k : $\boldsymbol{z}_{\boldsymbol{k}}^{(L)}$
- Activation function (sigmoid, logistic, tanh, ReLU, …) : $\boldsymbol{\sigma}$
- Expected output value for neuron n : $\mathbf{y}_{\boldsymbol{k}}$
- Cost function : $\mathbf{C_0}$



Martin-Pierre SCHMIDT
MSD3@3ds.com
martinpierre.schmidt@insa-rouen.fr

Artificial Neural Network

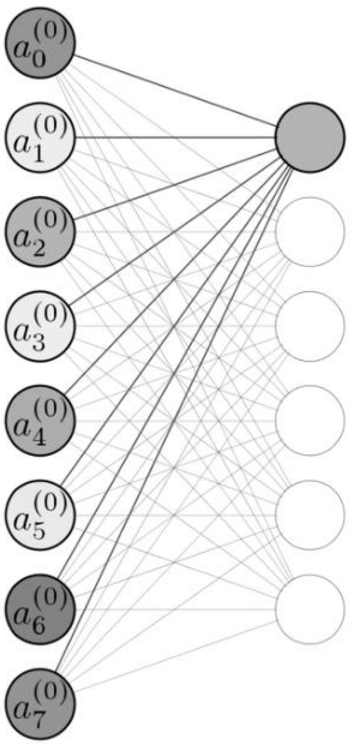
Feed-Forward

- The **Feed-Forward** algorithm is the process of computing the values of the **output layer** given :
  - The values of the **input layer**
    - Represented as a single vector
  - The values of the **weights**
    - Represented as a matrix between each layer
- Each neuron of the network simply contain a scalar value
  - The value of a neuron depends solely on the values of the previous layer and the weight in between
  - By feeding the values of each layer into the next one, we propagate the information forward though the network

Martin-Pierre SCHMIDT
MSD3@3ds.com
martinpierre.schmidt@insa-rouen.fr

Artificial Neural Network

Feed-Forward



$$a_0^{(1)} = \overset{\text{Sigmoid}}{\overset{\downarrow}{\sigma}}\left(w_{0,0}\ a_0^{(0)} + w_{0,1}\ a_1^{(0)} + \cdots + w_{0,n}\ a_n^{(0)} + \underset{\text{Bias}}{\underset{\uparrow}{b_0}} \right)$$

$$\sigma\left(\begin{bmatrix} w_{0,0} & w_{0,1} & \cdots & w_{0,n} \\ w_{1,0} & w_{1,1} & \cdots & w_{1,n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{k,0} & w_{k,1} & \cdots & w_{k,n} \end{bmatrix} \begin{bmatrix} a_0^{(0)} \\ a_1^{(0)} \\ \vdots \\ a_n^{(0)} \end{bmatrix} + \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_n \end{bmatrix} \right)$$

$$\sigma\left(\mathbf{W}\mathbf{a}^{(0)} + \mathbf{b}\right)$$

Martin-Pierre SCHMIDT
MSD3@3ds.com
martinpierre.schmidt@insa-rouen.fr

Artificial Neural Network

Feed-Forward

- playground.tensorflow.org
  - Solve the concentric distribution problem
  - Solve the checker pattern problem
  - Don’t solve the spiral problem

Martin-Pierre SCHMIDT
MSD3@3ds.com
martinpierre.schmidt@insa-rouen.fr