Machine Learning Basics

Victor Bouvier, Sidetrade
DTY @CentraleSupelec

September 8, 2020

Who I am

Maybe you already know me

- Centrale P2016+1
- PhD student @MICS
- Cifre collaboration @Sidetrade

Who I am

Maybe you already know me

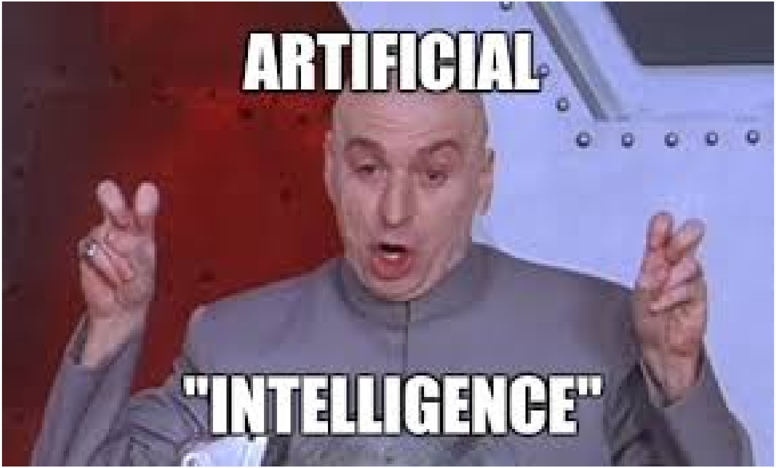
- Centrale P2016+1
- PhD student @MICS
- Cifre collaboration @Sidetrade

What is my research focus

- Deep Learning methods aim to learn powerful data representation for solving a large set of tasks
- Since DL methods don't need any pre-processing (or a few...), representations is highly sensitive to the data distribution (bias),
- **I am working on learning deep representations of data unbiased to known nuisance factors**

Content

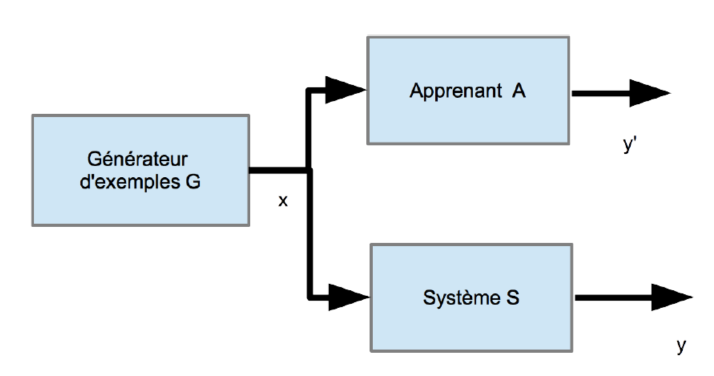
1. Tasks
    - Generalization,
    - Metrics,
    - Capacity and Regularization.
2. Supervised Learning
    - Regression VS Classification
    - Naive Bayes
    - Local-template matching
    - Linear models and kernel trick
    - Ensemble methods: Bagging VS Boosting
3. Unsupervised learning
    - K-means
    - PCA
4. Model selection


ARTIFICIAL
"INTELLIGENCE"

Task and Reflex Agent

Task & Machine Learning

Machine Learning aims to reproduce a given task T which maps an input X to an input Y by 'learning it' from the data.



Tasks and Reflex Agent

Task & Machine Learning

Machine Learning aims to reproduce a given task T which maps an input X to an input Y by 'learning it' from the data.

Tasks (non-exhaustive)

- Classification

Tasks and Reflex Agent

Task & Machine Learning

Machine Learning aims to reproduce a given task T which maps an input X to an input Y by 'learning it' from the data.

Tasks (non-exhaustive)

- Classification
- Regression

Tasks and Reflex Agent

Task & Machine Learning

Machine Learning aims to reproduce a given task T which maps an input X to an input Y by 'learning it' from the data.

Tasks (non-exhaustive)

- Classification
- Regression
- Translation

Tasks and Reflex Agent

Task & Machine Learning

Machine Learning aims to reproduce a given task T which maps an input X to an input Y by 'learning it' from the data.

Tasks (non-exhaustive)

- Classification
- Regression
- Translation
- Anomaly detection

Tasks and Reflex Agent

Task & Machine Learning

Machine Learning aims to reproduce a given task T which maps an input X to an input Y by 'learning it' from the data.

Tasks (non-exhaustive)

- Classification
- Regression
- Translation
- Anomaly detection
- Synthesis and sampling

Tasks and Reflex Agent

Task & Machine Learning

Machine Learning aims to reproduce a given task T which maps an input X to an input Y by 'learning it' from the data.

Tasks (non-exhaustive)

- Classification
- Regression
- Translation
- Anomaly detection
- Synthesis and sampling
- Imputation of missing values

Tasks and Reflex Agent

Task & Machine Learning

Machine Learning aims to reproduce a given task T which maps an input X to an input Y by 'learning it' from the data.

Tasks (non-exhaustive)

- Classification
- Regression
- Translation
- Anomaly detection
- Synthesis and sampling
- Imputation of missing values
- Denoising

Tasks and Reflex Agent

Task & Machine Learning

Machine Learning aims to reproduce a given task T which maps an input X to an input Y by 'learning it' from the data.

Tasks (non-exhaustive)

- Classification
- Regression
- Translation
- Anomaly detection
- Synthesis and sampling
- Imputation of missing values
- Denoising
- Density estimation

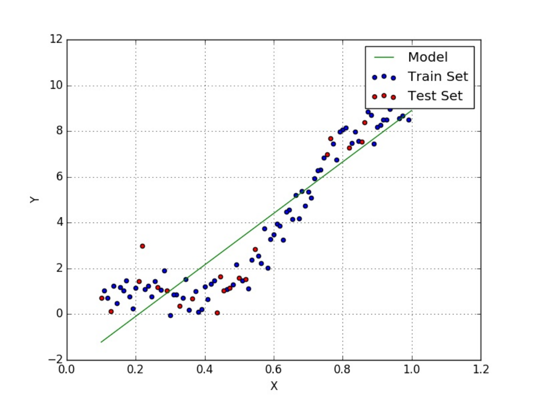
The curse of learning the task from the data

The curse of learning the task from the data

Machine Learning model

For a given x, a ML model outputs a probability distribution of y. The mapping is defined by a set of parameters θ.

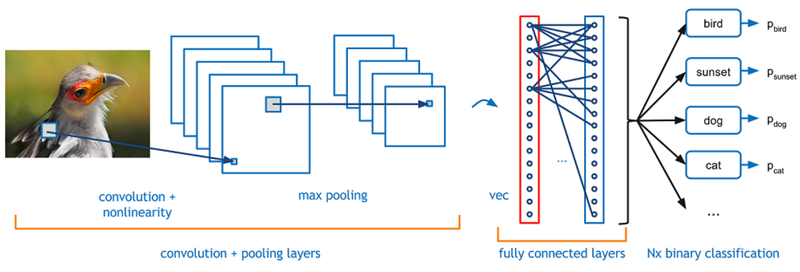
$$x \longrightarrow p_{\mathrm{model}}(y|x;\theta)$$



Machine Learning model

For a given x, a ML model outputs a probability distribution of y. The mapping is defined by a set of parameters θ.

$$x \longrightarrow p_{\mathrm{model}}(y|x;\theta)$$



The curse of learning the task from the data

Machine Learning model

For a given x, a ML model outputs a probability distribution of y. The mapping is defined by a set of parameters θ.

$$x \longrightarrow p_{\mathrm{model}}(y|x;\theta)$$

Maximum Likelihood Estimation (MLE)

When samples of data are available $(x_i)_{i=1}^{n}$ and we want to learn a model $p_{\mathrm{model}}(x;\theta)$ by finding a set of parameters θ

The curse of learning the task from the data

Machine Learning model

For a given x, a ML model outputs a probability distribution of y. The mapping is defined by a set of parameters θ.

$$x \longrightarrow p_{\mathrm{model}}(y|x;\theta)$$

Maximum Likelihood Estimation (MLE)

When samples of data are available $(x_i)_{i=1}^{n}$ and we want to learn a model $p_{\mathrm{model}}(x;\theta)$ by finding a set of parameters θ

$$\theta_{\mathrm{MLE}} = \arg\max_{\theta} \prod_{i=1}^{n} p_{\mathrm{model}}(y_i|x_i;\theta) = \arg\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} -\log p_{\mathrm{model}}(y_i|x_i;\theta)$$

The curse of learning the task from the data

Machine Learning model

For a given x, a ML model outputs a probability distribution of y. The mapping is defined by a set of parameters θ.

$$x \longrightarrow p_{\mathrm{model}}(y|x;\theta)$$

Maximum Likelihood Estimation (MLE)

When samples of data are available $(x_i)_{i=1}^{n}$ and we want to learn a model $p_{\mathrm{model}}(x;\theta)$ by finding a set of parameters θ

$$\theta_{\mathrm{MLE}} = \arg\max_{\theta} \prod_{i=1}^{n} p_{\mathrm{model}}(y_i|x_i;\theta) = \arg\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} -\log p_{\mathrm{model}}(y_i|x_i;\theta)$$

When infinite data is available:

$$\theta_{\mathrm{MLE}} = \arg\min_{\theta} E_{p_{\mathrm{data}}(x,y)}[-\log p_{\mathrm{model}}(y|x;\theta)]$$

The curse of learning from the data

$$\theta_{\mathrm{MLE}} = \arg\min_{\theta} \mathbb{E}_{p_{\mathrm{data}}(x,y)}[-\log p_{\mathrm{model}}(y|x;\theta)]$$

Empirical distribution

- Only a sample of data is available

The curse of learning from the data

$$\theta_{\mathrm{MLE}} = \arg\min_{\theta} \mathbb{E}_{p_{\mathrm{data}}(x,y)}[-\log p_{\mathrm{model}}(y|x;\theta)]$$

Empirical distribution

- Only a sample of data is available
- $(x_i) \sim \hat{p}_{\mathrm{data}}$ the empirical distribution:

$$\hat{p}_{\mathrm{data}}(x,y) = \frac{1}{n}\sum_{i=1}^{n} \delta(x-x_i)\delta(y-y_i) \neq p_{\mathrm{data}}(x,y)$$

The curse of learning from the data

$$\theta_{\mathrm{MLE}} = \arg\min_{\theta} \mathbb{E}_{\hat{p}_{\mathrm{data}}(x,y)}[-\log p_{\mathrm{model}}(y|x;\theta)]$$

Empirical distribution

- Only a sample of data is available
- $(x_i) \sim \hat{p}_{\mathrm{data}}$ the empirical distribution:

$$\hat{p}_{\mathrm{data}}(x,y) = \frac{1}{n}\sum_{i=1}^{n}\delta(x-x_i)\delta(y-y_i) \neq p_{\mathrm{data}}(x,y)$$

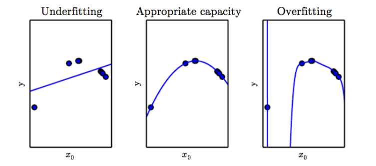
The curse of learning from the data

$$\theta_{\mathrm{MLE}} = \arg\min_{\theta} \mathbb{E}_{\hat{p}_{\mathrm{data}}(x,y)}[-\log p_{\mathrm{model}}(y|x;\theta)]$$

Empirical distribution

- Only a sample of data is available
- $(x_i) \sim \hat{p}_{\mathrm{data}}$ the empirical distribution:

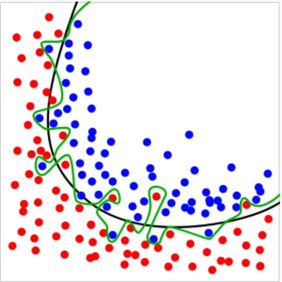
$$\hat{p}_{\mathrm{data}}(x,y) = \frac{1}{n}\sum_{i=1}^{n}\delta(x-x_i)\delta(y-y_i) \neq p_{\mathrm{data}}(x,y)$$



From Deep Learning Book

Overfitting

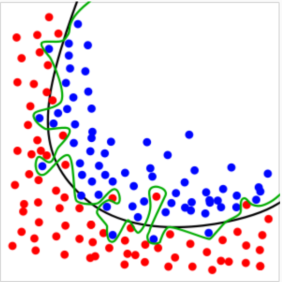
Overfitting



Overfitting

How to ensure that the model learns p_{data} and not $\hat{p}_{\text{data}}$?

Machine Learning is not optimization - Overfitting



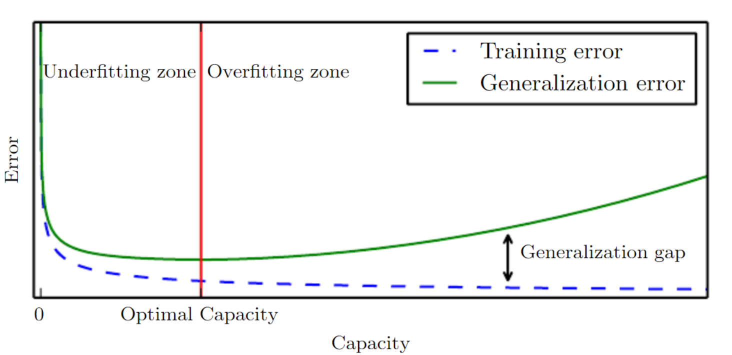
Overfitting

How to ensure that the model learns p_{data} and not $\hat{p}_{\text{data}}$?

Train - Test procedure

1. Split at random the dataset $\mathcal{D}$ into two non-overlapping subsets $\mathcal{D}_{\text{tr}}$ and $\mathcal{D}_{ts}$
2. Train the model on $\mathcal{D}_{\text{tr}}$
3. Evaluate the model on $\mathcal{D}_{\text{ts}}$

Capacity and generalization gap



From Deep Learning Book

Occam's razor

This principle states that among competing hypotheses that explain known observations equally well, we should choose the "simplest" one.

Regularization

- Regularization aims to reduce overfitting during training
- *Not discovering at test time that the model has overfitted the training data...*

What you learn is not what you want (or expected)

Accuracy is not enough

- You are hired to boot a model for anomaly detection in a insurance company
- You train a binary classifier (1 is an anomaly)
- You have 0.01% of labeled anomaly. What is the baseline model (accuracy-wise)?

		PREDICTED CLASS	
		Class=Yes	Class=No
ACTUAL CLASS	Class=Yes	a (TP)	b (FN)
	Class=No	c (FP)	d (TN)

$$\mathtt{Error\_rate} = \frac{FP + FN}{TP + TN + FP + FN}$$

Confusion matrix is all you need

		PREDICTED CLASS	
		Class=Yes	Class=No
ACTUAL CLASS	Class=Yes	a (TP)	b (FN)
	Class=No	c (FP)	d (TN)

$$\text{Sensibility} = \frac{TP}{TP + FN}$$

$$1 - \text{Specificity} = \frac{FP}{TN + FP} = 1 - \frac{TN}{TN + FP}$$

Confusion matrix is all you need

		PREDICTED CLASS	
		Class=Yes	Class=No
ACTUAL CLASS	Class=Yes	a (TP)	b (FN)
	Class=No	c (FP)	d (TN)

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Rappel} = \frac{TP}{TP + FN}$$

$$\text{F\_mesure} = 2\frac{\text{Precision} \times \text{Rappel}}{\text{Precision} + \text{Rappel}}$$

Supervised VS Unsupervised Learning

Supervised VS Unsupervised Learning

Task & Machine Learning

Machine Learning aims to reproduce a given task T which maps an input X to an input Y by 'learning it' from the data.

Supervised VS Unsupervised Learning

Task & Machine Learning

Machine Learning aims to reproduce a given task T which maps an input X to an input Y by 'learning it' from the data.

Supervision

is the case when the dataset is $\mathcal{D} = (x_i, y_i)_{i=1}^{n}$ i.e. when output samples are available.

Supervised VS Unsupervised Learning

Task & Machine Learning

Machine Learning aims to reproduce a given task T which maps an input X to an input Y by 'learning it' from the data.

Supervision

is the case when the dataset is $\mathcal{D} = (x_i, y_i)_{i=1}^{n}$ i.e. when output samples are available.

Limits of such paradigm

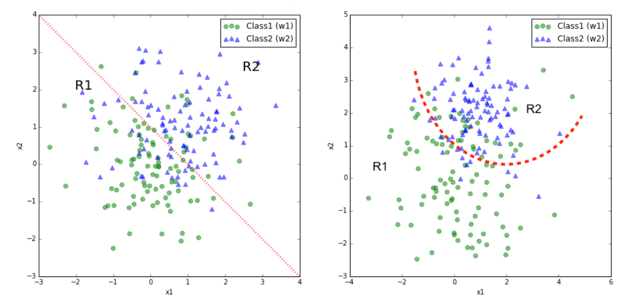
- Considering the joint distribution, a supervised learning problem is an unsupervised learning problem,
- Considering marginal distribution, an unsupervised learning problem is several supervised learning problem (self-supervised learning).

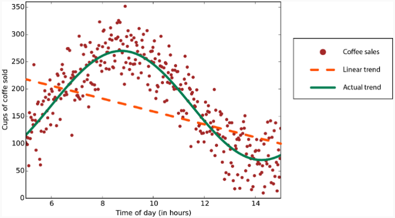
Yann Lecun’s cake

- **“Pure” Reinforcement Learning (cherry)**
  - The machine predicts a scalar reward given once in a while.
  - **A few bits for some samples**
- **Supervised Learning (icing)**
  - The machine predicts a category or a few numbers for each input
  - Predicting human-supplied data
  - **10→10,000 bits per sample**
- **Unsupervised/Predictive Learning (cake)**
  - The machine predicts any part of its input for any observed part.
  - Predicts future frames in videos
  - **Millions of bits per sample**

- **(Yes, I know, this picture is slightly offensive to RL folks. But I’ll make it up)**

Some very classical models

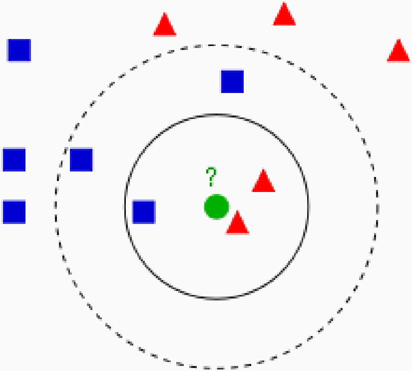

Class1 (w1)
Class2 (w2)
R1
R2
x1
x2


Coffee sales
Linear trend
Actual trend
Cups of coffe sold
Time of day (in hours)

Naives Bayes $X = (X_1, ..., X_n) \rightarrow Y$

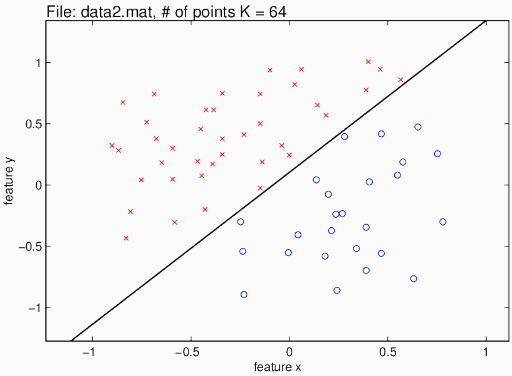
$$\mathbb{P}(Y|X) = \frac{\mathbb{P}(X|Y)\mathbb{P}(Y)}{\mathbb{P}(X)} \approx \frac{\mathbb{P}(X_1|Y)\cdots\mathbb{P}(X_n|Y)\mathbb{P}(Y)}{\mathbb{P}(X)} \propto \mathbb{P}(X_1|Y)\cdots\mathbb{P}(X_n|Y)\mathbb{P}(Y)$$

k-NNs

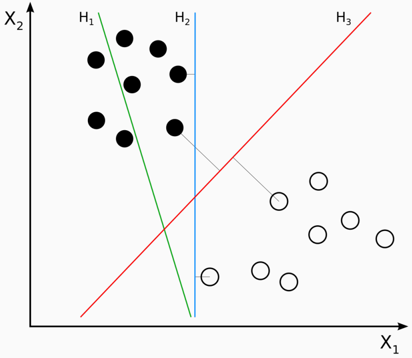


The case of linear separability

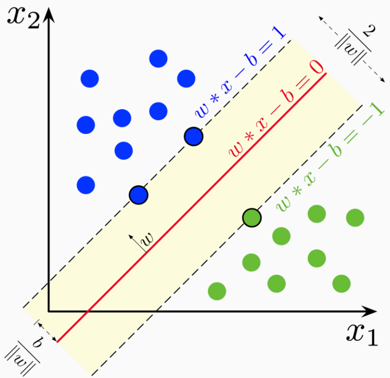
Linear separation



Linear separation

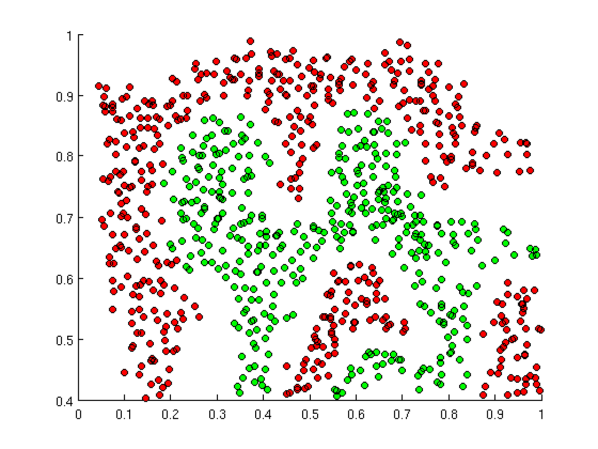


Linear separation

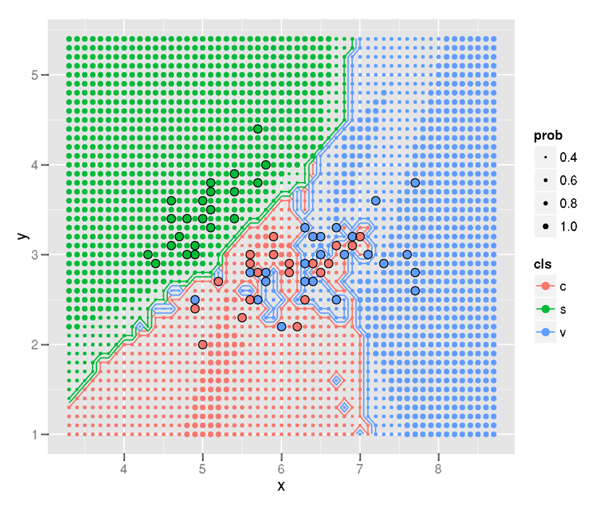


From linear to non-linear separability

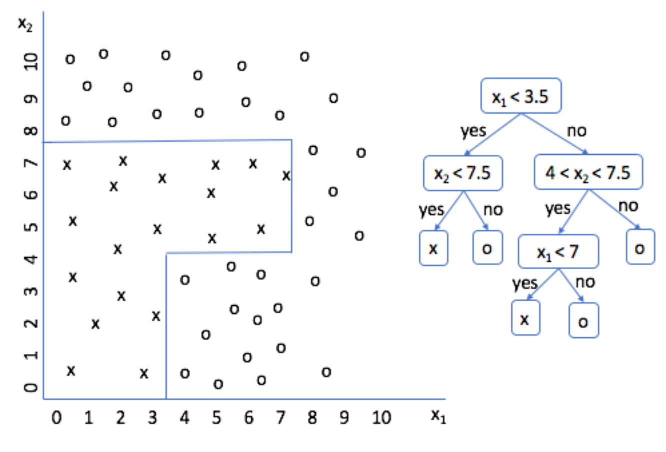
Non-linear separability



Non-linear separability



Non-linear separability

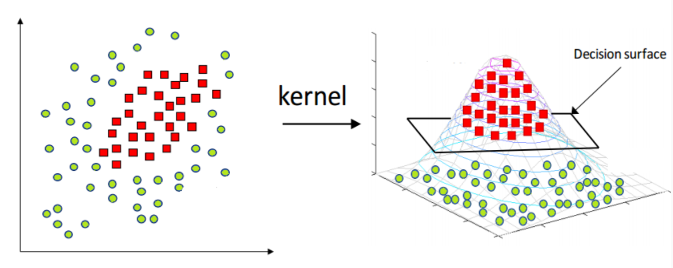


Mercer's theorem

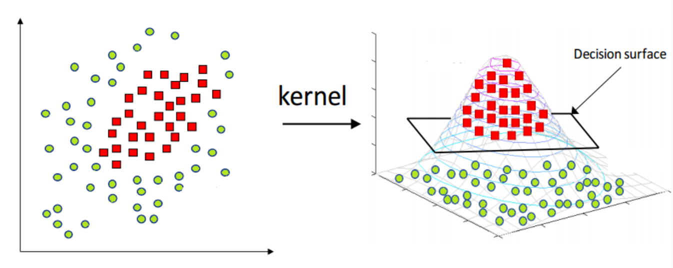
Let $K : \mathcal{R}^d \times \mathcal{R}^d$ continuous, symetric and semi-definite positive, then it exists $\varphi : \mathcal{R}^d \to \mathcal{R}^{d'}$ such that:

$$\forall (x, y) \in \mathcal{R}^d, K(x, y) = \varphi(x)\varphi(y)$$

Kernel trick



Kernel trick

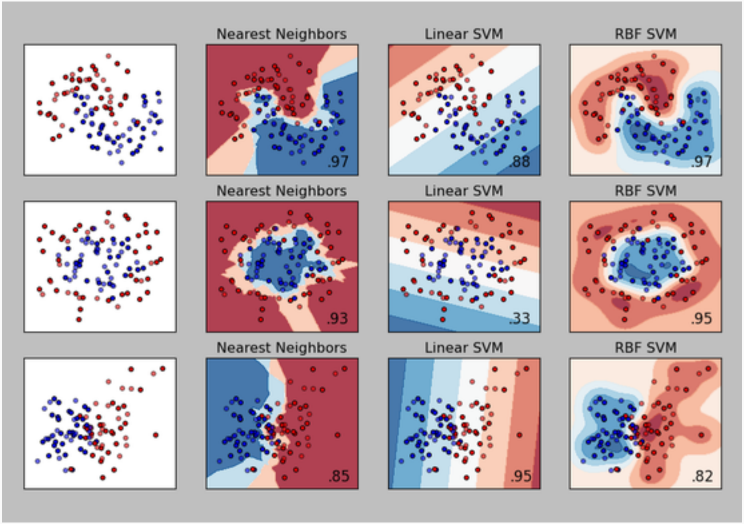


Limitations

- Linear kernel $K(x, y) = x \cdot y$
- Polynomial kernel $K(x, y) = (1 + x \cdot y)^d$
- Gaussian kernel $K(x, y) = \exp(-\gamma ||x - y||^2)$
- Quadratic kernel $K(x, y) = (1 + \gamma ||x - y||^2)^{-\alpha}$

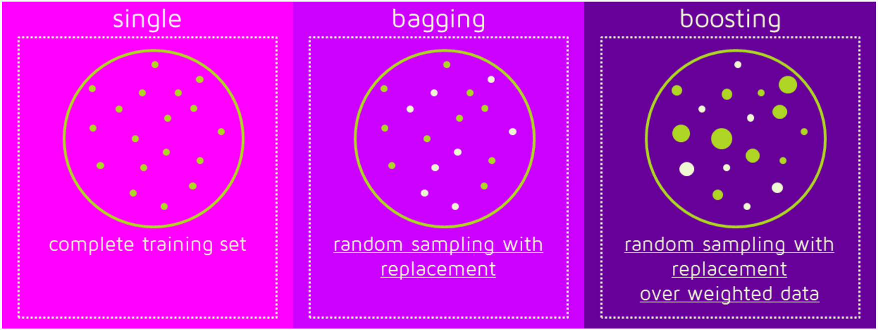
There is no reason for this kernel to work...

Non-linear separation



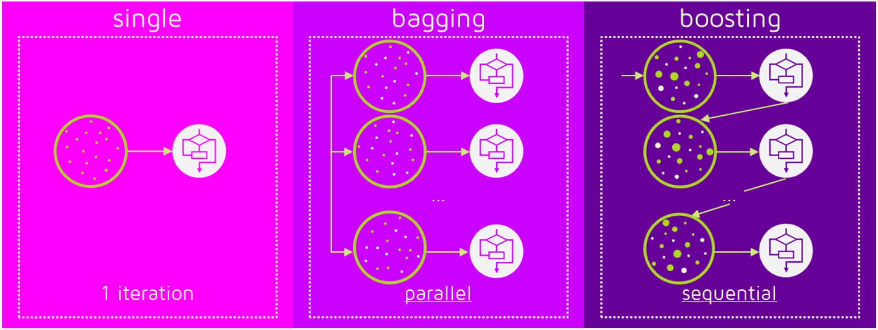
A lot of weak learners is better than a single strong learner

Boosting VS Bagging



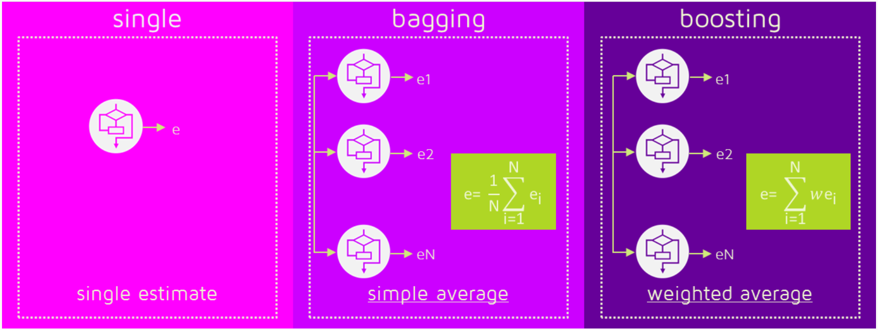
`quantdare.com/what-is-the-difference-between-bagging-and-boosting/`

Boosting VS Bagging



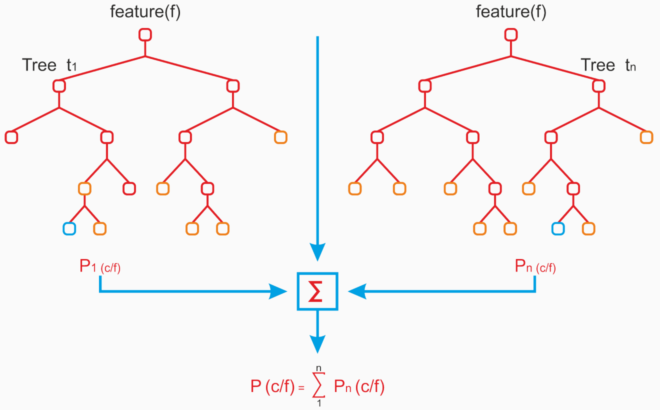
quantdare.com/
what-is-the-difference-between-bagging-and-boosting/

Boosting VS Bagging



`quantdare.com/what-is-the-difference-between-bagging-and-boosting/`

Bagging decision trees: Random Forest



From O'Reilly media

Those algorithms do local template matching

Those algorithms do local template matching

$$\hat{y}(x) = \arg\max_{y \in \mathcal{Y}} \sum_{i=1}^{n} \mathbb{1}(y = y_i) w(x, x_i)$$

- $\mathbb{1}(y = y_i)$: is the vote of the training point x_i
- $w(x, x_i)$ is its contribution to the vote:
  - $w(x, x_i) = 1$
  - $\lim_{||x - x_i|| \to \infty} w(x, x_i) = 0$

Those algorithms do local template matching

$$\hat{y}(x) = \arg\max_{y \in \mathcal{Y}} \sum_{i=1}^{n} \mathbf{1}(y = y_i) w(x, x_i)$$

- $\mathbf{1}(y = y_i)$: is the vote of the training point x_i
- $w(x, x_i)$ is its contribution to the vote:
    - $w(x, x_i) = 1$
    - $\lim_{||x - x_i|| \to \infty} w(x, x_i) = 0$

Local template matching

- $k - NN$: it's definition...
- SVM:

$$\sum_{i=1}^{n} \alpha_i y_i \varphi(x_i) \cdot \varphi(x) = \sum_{i=1}^{n} \alpha_i y_i K(x, x_i), \quad \alpha_i \in \{0, 1\}$$

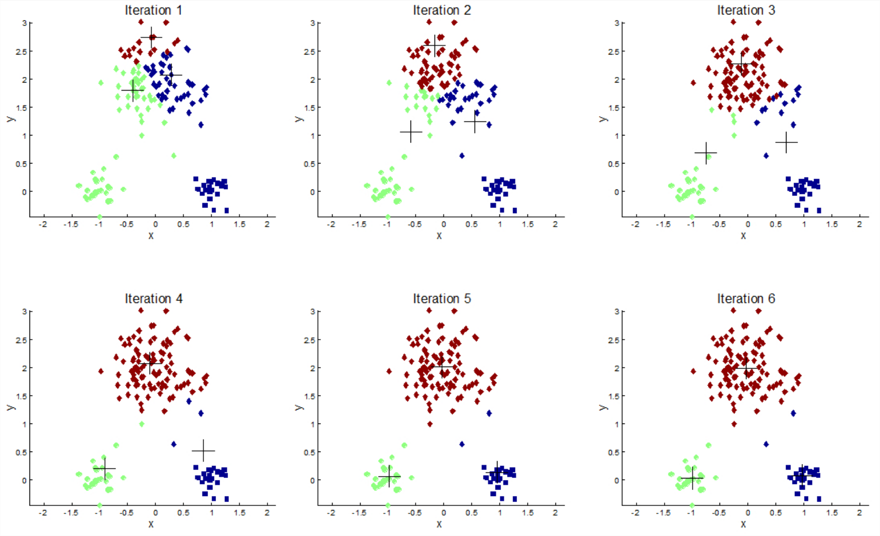
- Random Forest divides around training points with hard separation

Some very classical unsupervised models

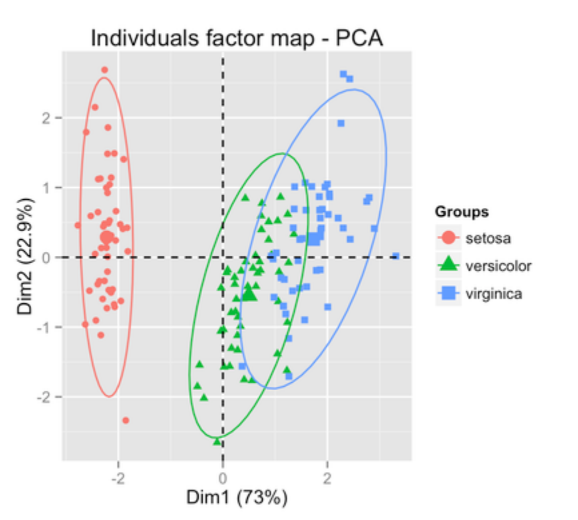
- **"Pure" Reinforcement Learning (cherry)**
  - The machine predicts a scalar reward given once in a while.
  - **A few bits for some samples**
- **Supervised Learning (icing)**
  - The machine predicts a category or a few numbers for each input
  - Predicting human-supplied data
  - **10→10,000 bits per sample**
- **Unsupervised/Predictive Learning (cake)**
  - The machine predicts any part of its input for any observed part.
  - Predicts future frames in videos
  - **Millions of bits per sample**

- **(Yes, I know, this picture is slightly offensive to RL folks. But I'll make it up)**

K-Means


Iteration 1
Iteration 2
Iteration 3
Iteration 4
Iteration 5
Iteration 6
x
y

Principal Component Analysis



Model selection

Hyperparamters and validation

Hyperparameters

- Train / test split,

Hyperparamters and validation

Hyperparameters

- Train / test split,
- Model,

Hyperparamters and validation

Hyperparameters

- Train / test split,
- Model,
- Initializers,
- Number of parameters (Capacity),

Hyperparamters and validation

Hyperparameters

- Train / test split,
- Model,
- Initializers,
- Number of parameters (Capacity),
- Regularization coefficient
- ...

Hyperparamters and validation

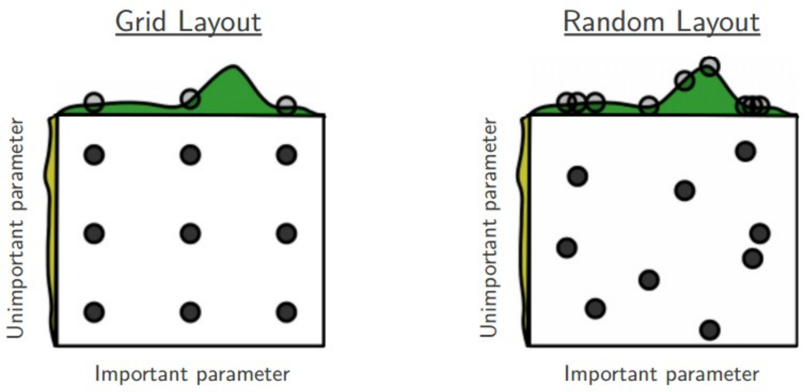
Hyperparameters

- Train / test split,
- Model,
- Initializers,
- Number of parameters (Capacity),
- Regularization coefficient
- ...

Train - Validation - Test sets

- For each combination of parameters train a model on a train set
- Select the best model on the validation set
- Evaluate the model generalize well on the test set

Grid search


Grid Layout
Unimportant parameter
Important parameter
Random Layout
Unimportant parameter
Important parameter