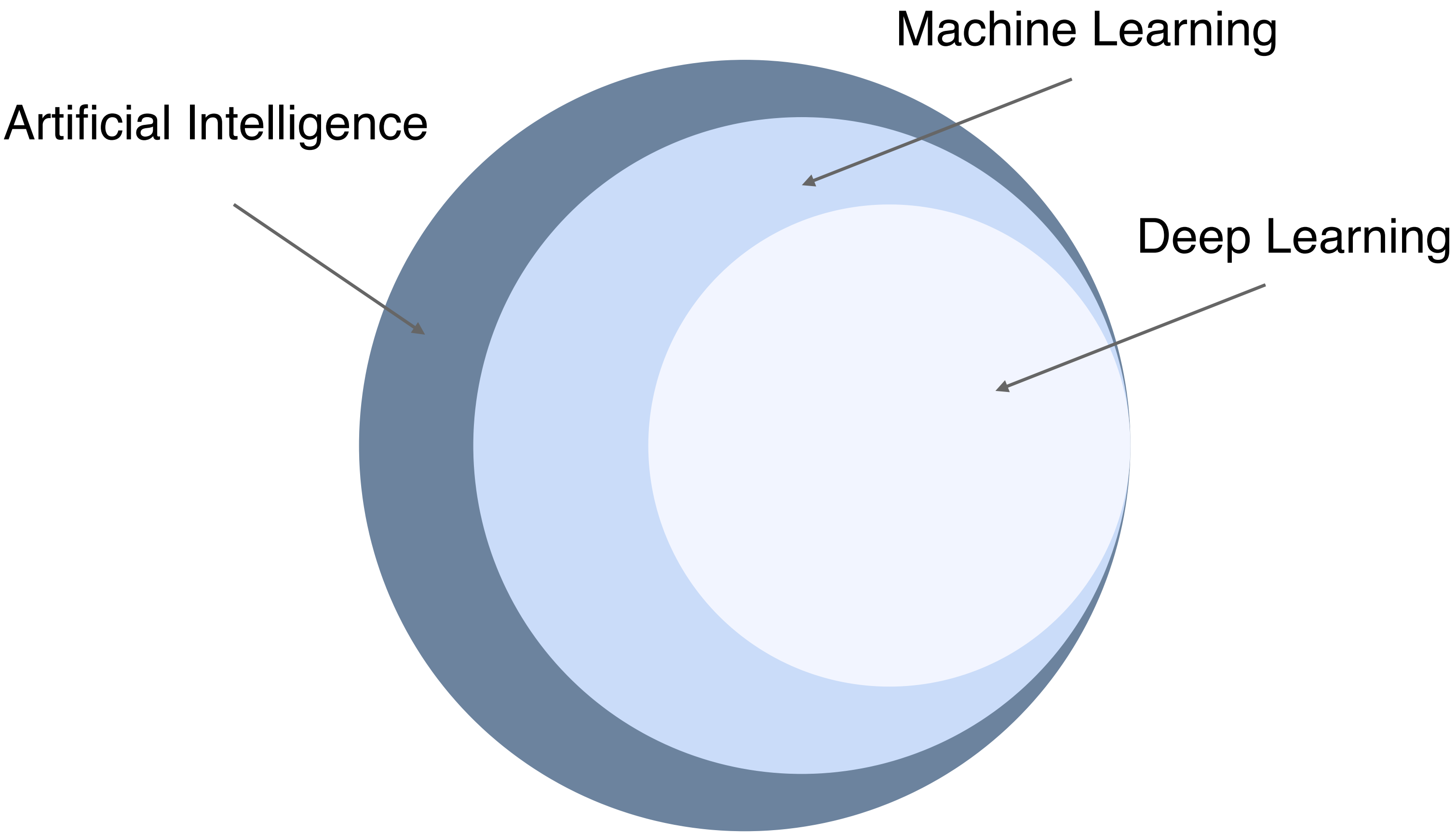
Introduction to Machine Learning

Odyssée Merveille and Emmanuel Roux
CREATIS, Lyon

Content

- A short historical background
- Supervised Learning
- Unsupervised Learning
- Conclusion

A short historical background



Machine Learning

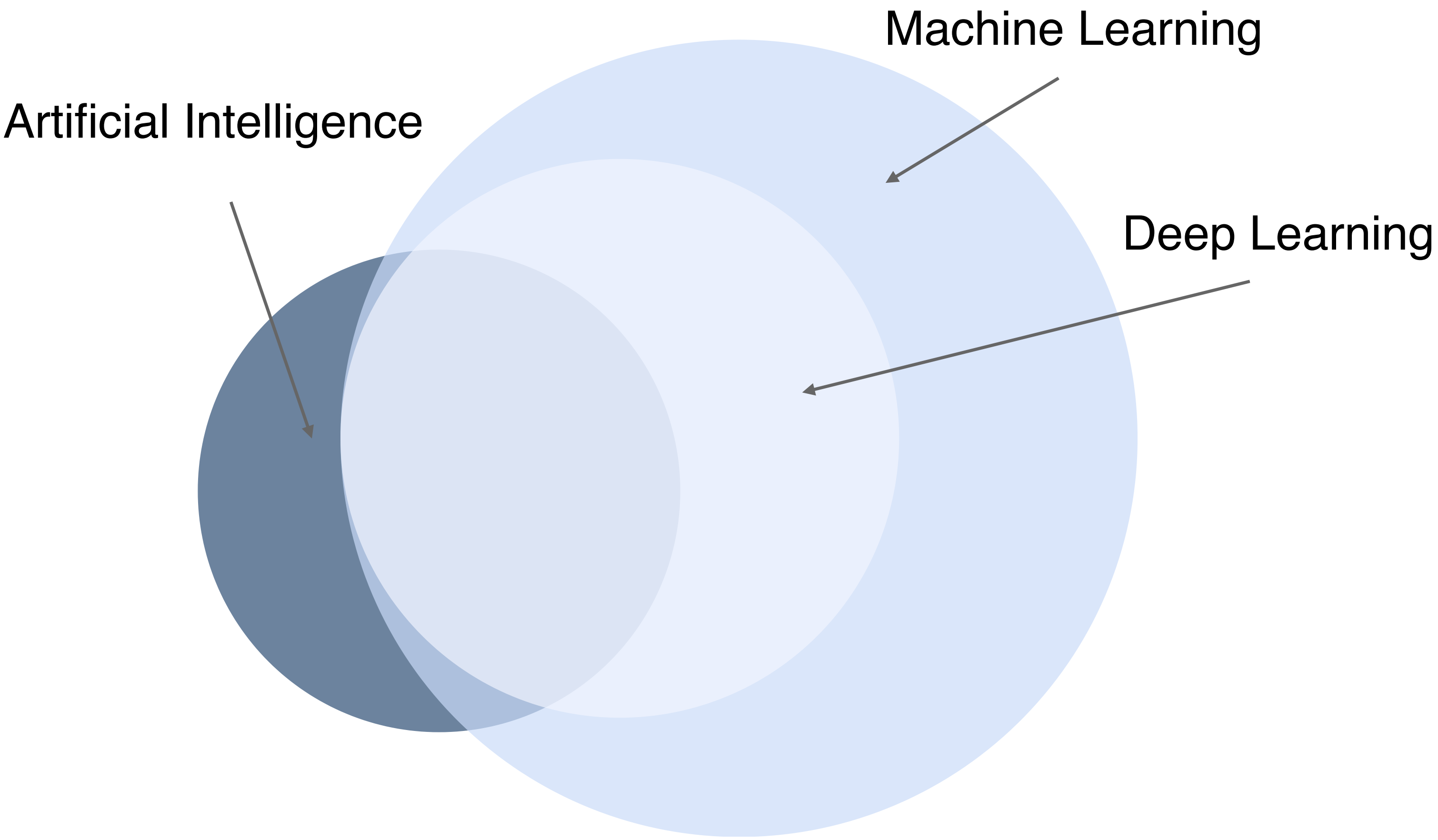
Artificial Intelligence

Deep Learning

Inspired (and simplified) from the deeplearningbook.org

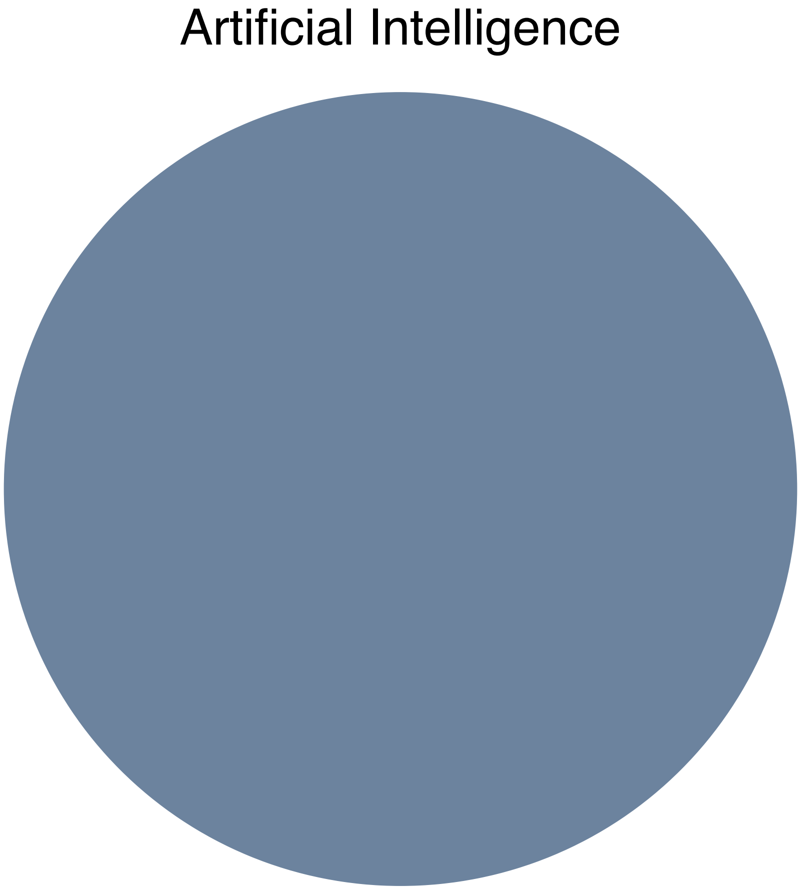
(I. Goodfellow and Y. Bengio, A. Courville, 2016)

A short historical background



Inspired from Sebastian Raschka's deep-learning course

A short historical background

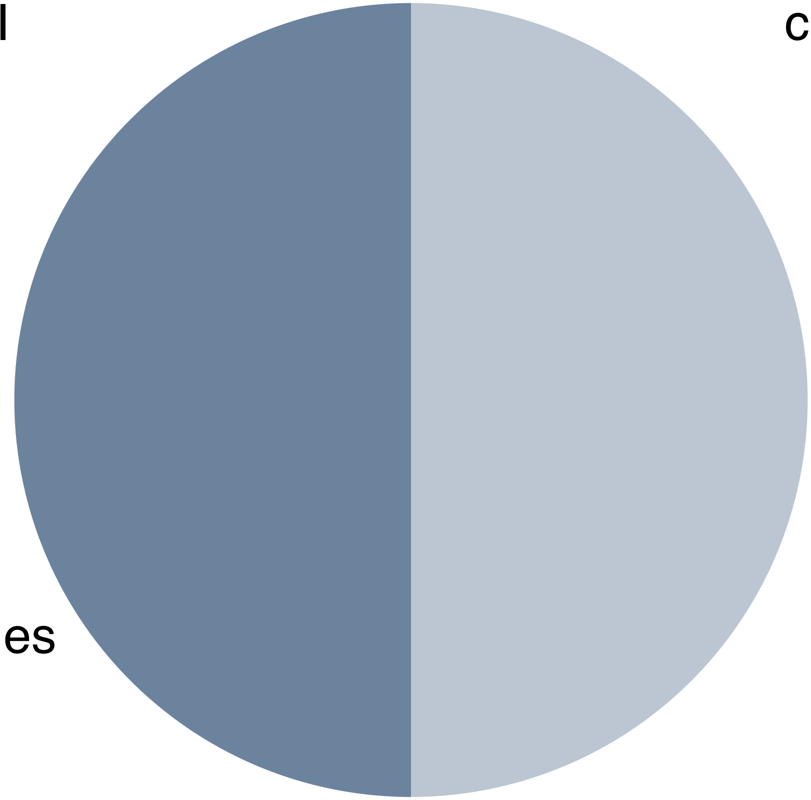


Artificial Intelligence

Inspired from (Cardon D., Cointet J.-P., Mazieres A., 2018)

A short historical background

Artificial Intelligence



Symbolic AI

DEDUCTIVE

rule-based

no need of examples

connectionism

INDUCTIVE

example based

adaptation

Inspired from (Cardon D., Cointet J.-P., Mazieres A., 2018)

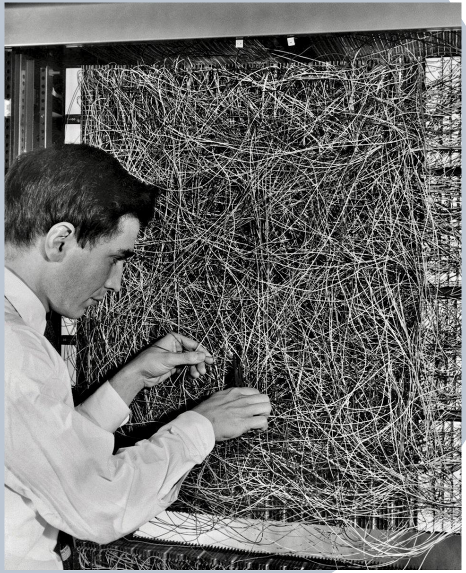
A short historical background

Cybernetics (40’s to 60’s)

Symbolic AI

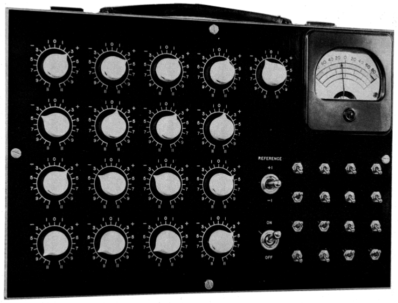
connexionism

Perceptron (Rosenblatt)

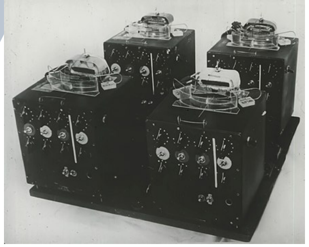


source reddit

ADALINE (Widrow & Hoff)



https://isl.stanford.edu/~widrow/papers/t1960anadaptive.pdf



Homeostat, 1948

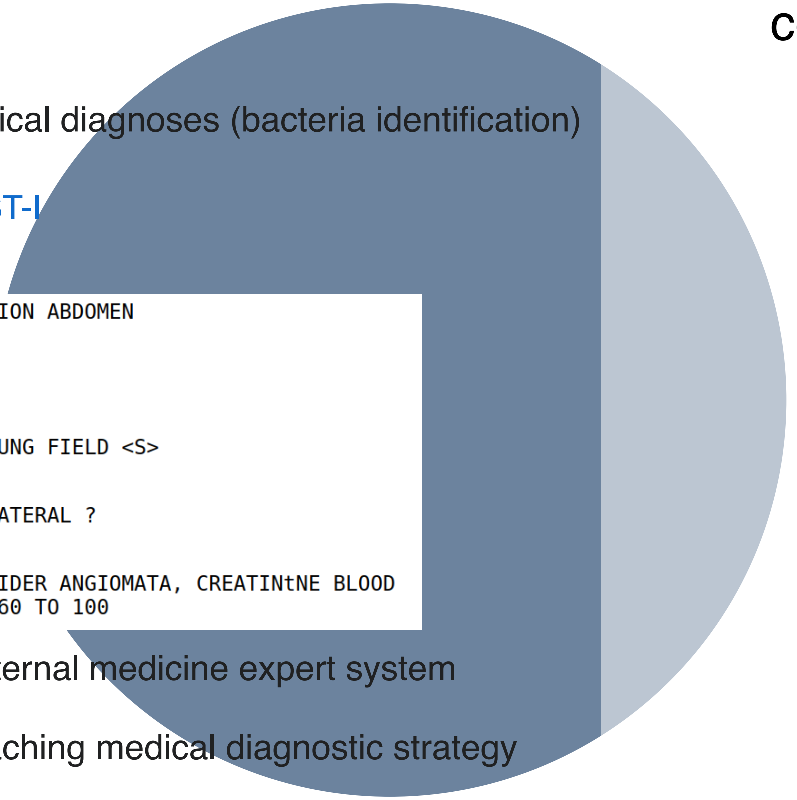
(W. Ross Ashby)

source wikipedia

Inspired from (Cardon D., Cointet J.-P., Mazieres A., 2018)

A short historical background

Symbolic Artificial Intelligence (60's to 80's)



MYCIN (Shortliffe): medical diagnoses (bacteria identification)

Transcript of an INTERNIST-I
Consultation (Myers)

```
Please Enter Findings of PALPATION ABDOMEN
*GO

SPLENOMEGALY MODERATE ?
NO

Please Enter Findings of XRAY LUNG FIELD <S>
*GO

CHEST XRAY HILAR ADENOPATHY BILATERAL ?
NO

DISREGARDING: JAUNDICE, SKIN SPIDER ANGIOMATA, CREATINtNE BLOOD
INCREASED, UREA NITROGEN BLOOD 60 TO 100
```

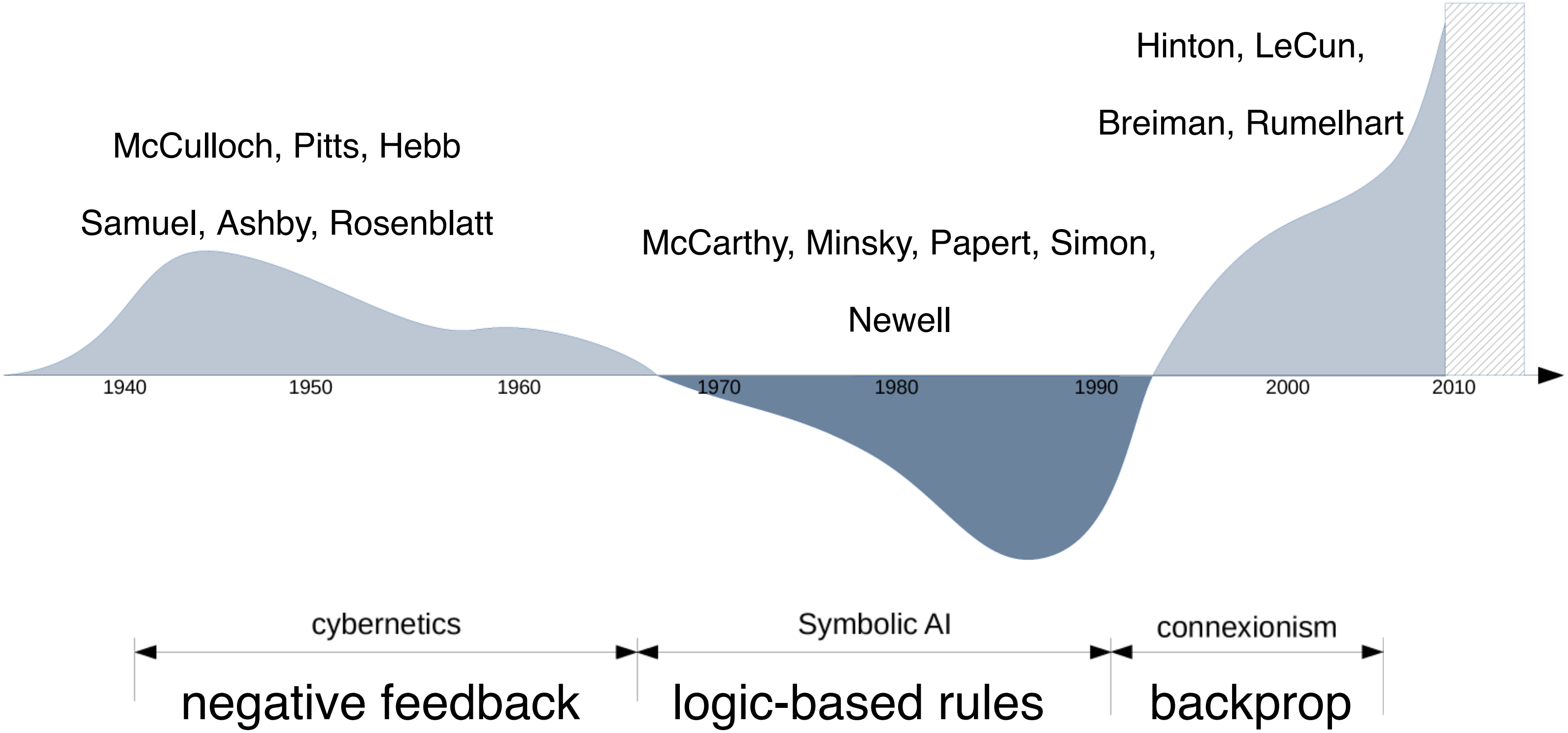
CADUCEUS (Pople): internal medicine expert system

GUIDON (Clancey): teaching medical diagnostic strategy

Inspired from (Cardon D., Cointet J.-P., Mazieres A., 2018)

A short historical background

publication trends timeline



McCulloch, Pitts, Hebb

Samuel, Ashby, Rosenblatt

McCarthy, Minsky, Papert, Simon,

Newell

Hinton, LeCun,

Breiman, Rumelhart

cybernetics — negative feedback

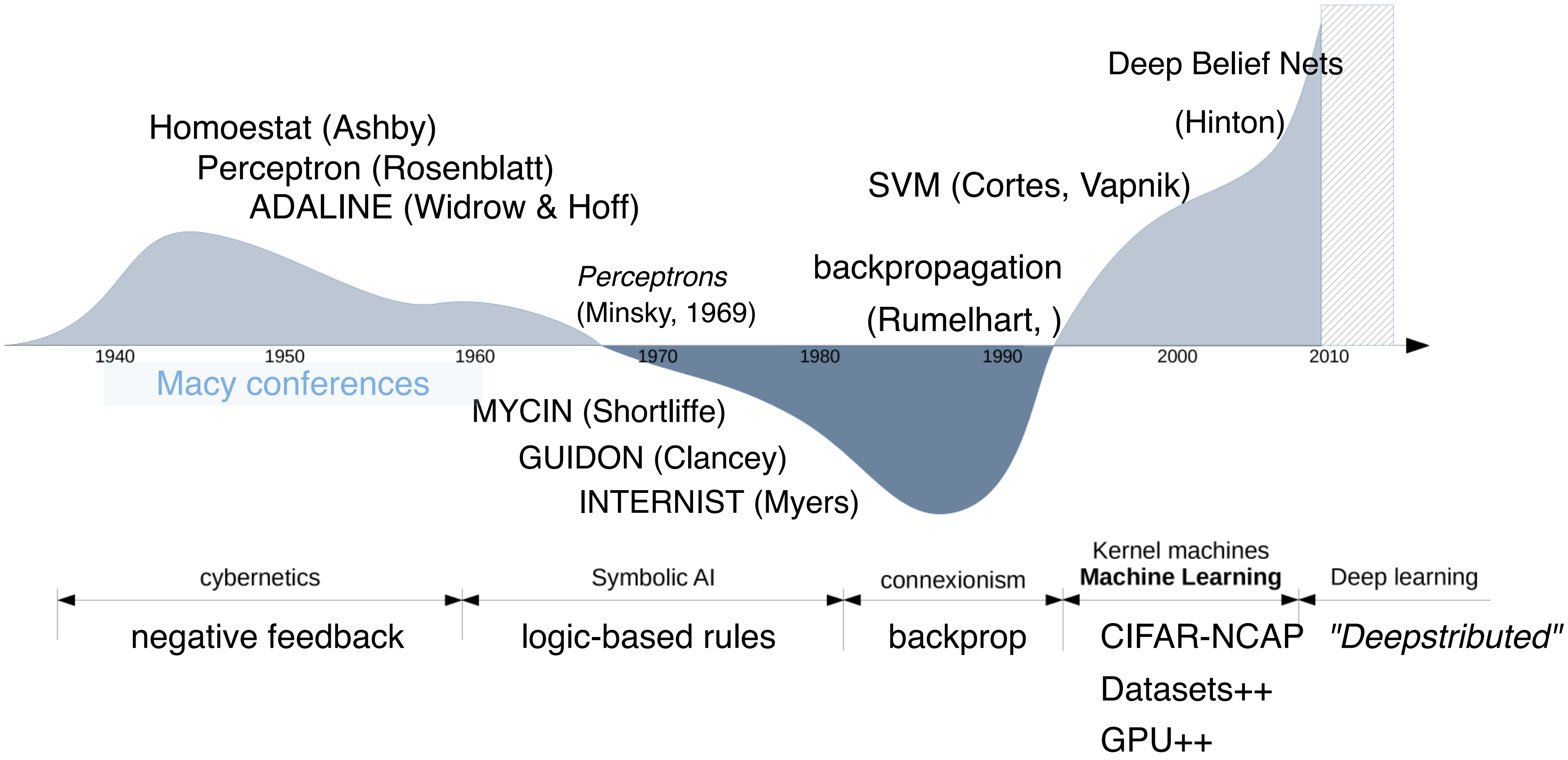
Symbolic AI — logic-based rules

connexionism — backprop

Inspired from (Cardon D., Cointet J.-P., Mazieres A., 2018)

A short historical background

ideas trends timeline



Homoestat (Ashby)
Perceptron (Rosenblatt)
ADALINE (Widrow & Hoff)

Macy conferences

Perceptrons (Minsky, 1969)

MYCIN (Shortliffe)
GUIDON (Clancey)
INTERNIST (Myers)

backpropagation (Rumelhart,)

SVM (Cortes, Vapnik)

Deep Belief Nets (Hinton)

cybernetics: negative feedback

Symbolic AI: logic-based rules

connexionism: backprop

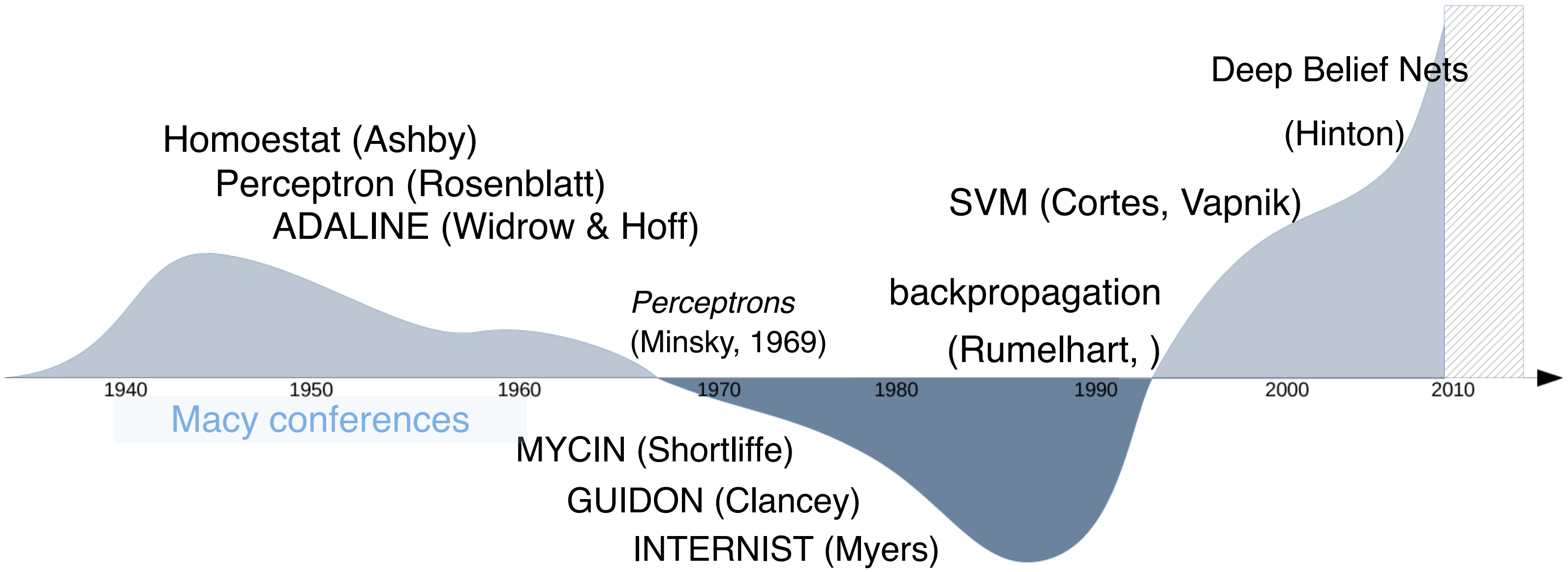
Kernel machines **Machine Learning**: CIFAR-NCAP, Datasets++, GPU++

Deep learning: *"Deepstributed"*

Inspired from (Cardon D., Cointet J.-P., Mazieres A., 2018)

A short historical background

ideas trends timeline

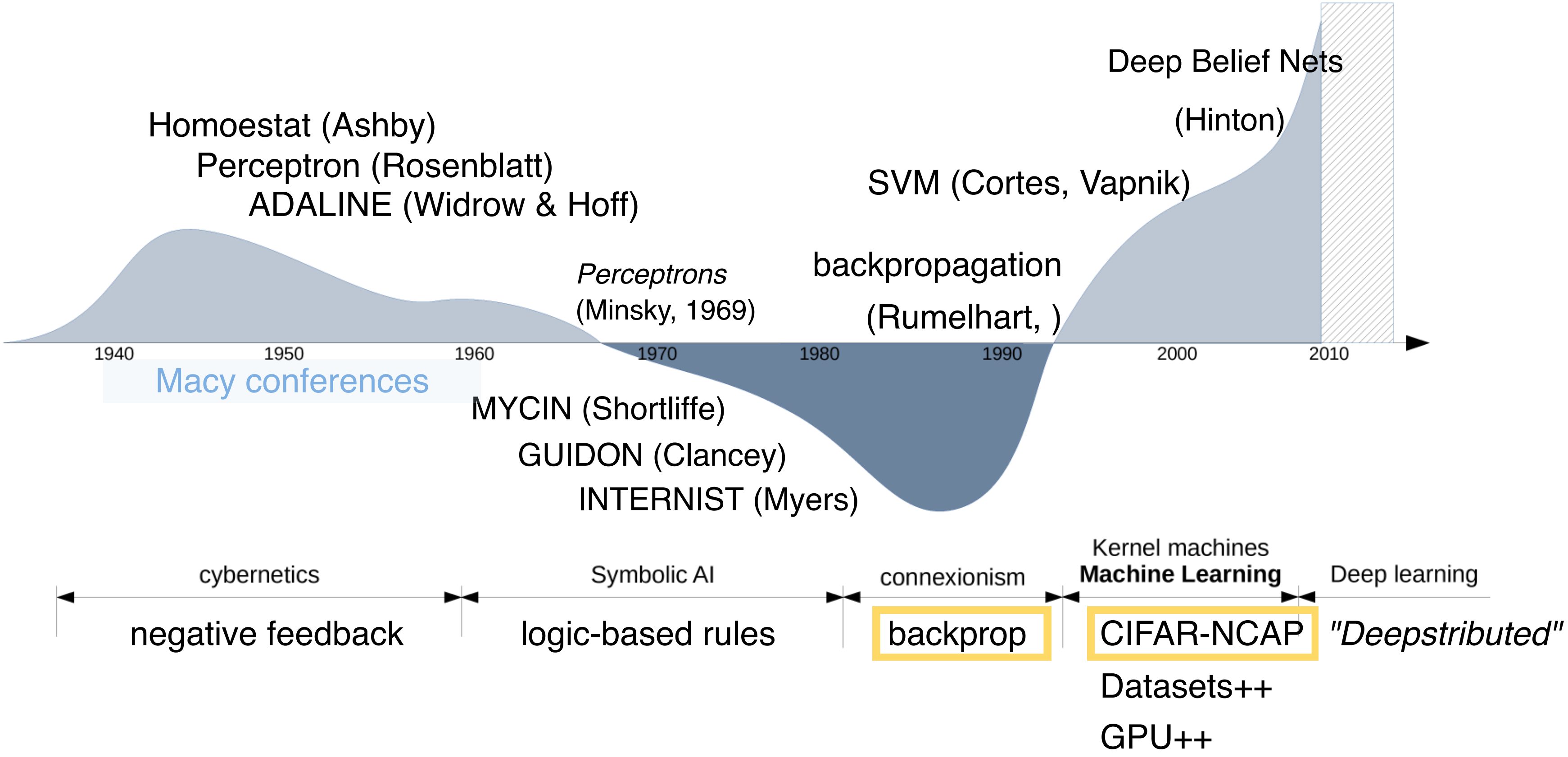


cybernetics	Symbolic AI	connexionism	Kernel machines **Machine Learning**	Deep learning
negative feedback	logic-based rules	backprop	CIFAR-NCAP Datasets++ GPU++	*"Deepstributed"*

Inspired from (Cardon D., Cointet J.-P., Mazieres A., 2018)

A short historical background

ideas trends timeline



Homoestat (Ashby)
Perceptron (Rosenblatt)
ADALINE (Widrow & Hoff)

Macy conferences

Perceptrons (Minsky, 1969)

MYCIN (Shortliffe)
GUIDON (Clancey)
INTERNIST (Myers)

backpropagation (Rumelhart,)

SVM (Cortes, Vapnik)

Deep Belief Nets (Hinton)

cybernetics: negative feedback

Symbolic AI: logic-based rules

connexionism: backprop

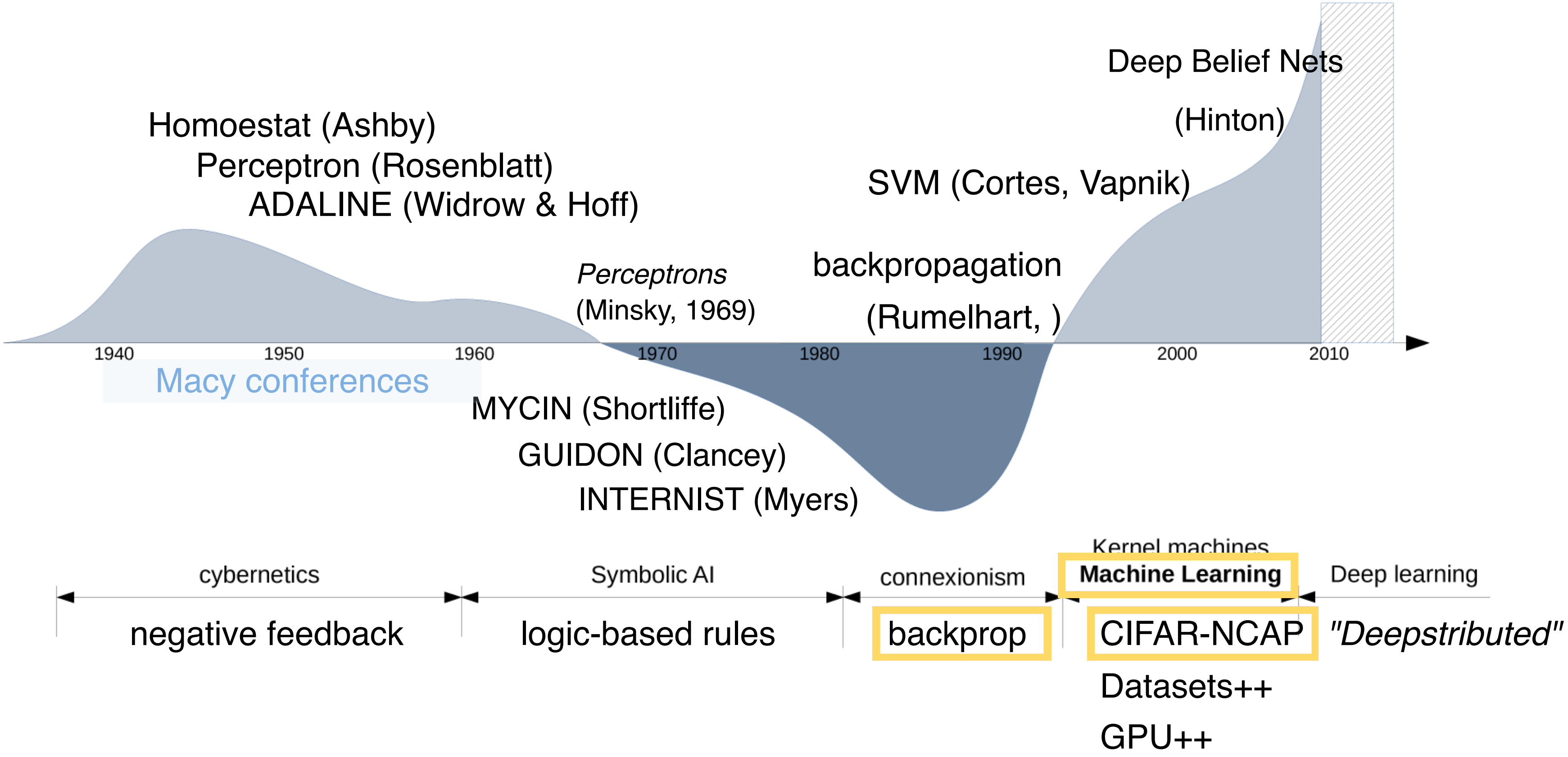
Kernel machines **Machine Learning**: CIFAR-NCAP, Datasets++, GPU++

Deep learning: *"Deepstributed"*

Inspired from (Cardon D., Cointet J.-P., Mazieres A., 2018)

A short historical background

ideas trends timeline



Homoestat (Ashby)
Perceptron (Rosenblatt)
ADALINE (Widrow & Hoff)

Perceptrons
(Minsky, 1969)

backpropagation
(Rumelhart,)

SVM (Cortes, Vapnik)

Deep Belief Nets
(Hinton)

Macy conferences

MYCIN (Shortliffe)
GUIDON (Clancey)
INTERNIST (Myers)

cybernetics — negative feedback

Symbolic AI — logic-based rules

connexionism — backprop

Kernel machines
Machine Learning — CIFAR-NCAP
Datasets++
GPU++

Deep learning — *"Deepstributed"*

Inspired from (Cardon D., Cointet J.-P., Mazieres A., 2018)

A short historical background

CREATIS

Machine Learning

“ *A computer program is said to learn [...] if*

A short historical background

> *“Machine learning is the field of study that*
> *gives computers the ability to learn*
> *without being explicitly programmed.*

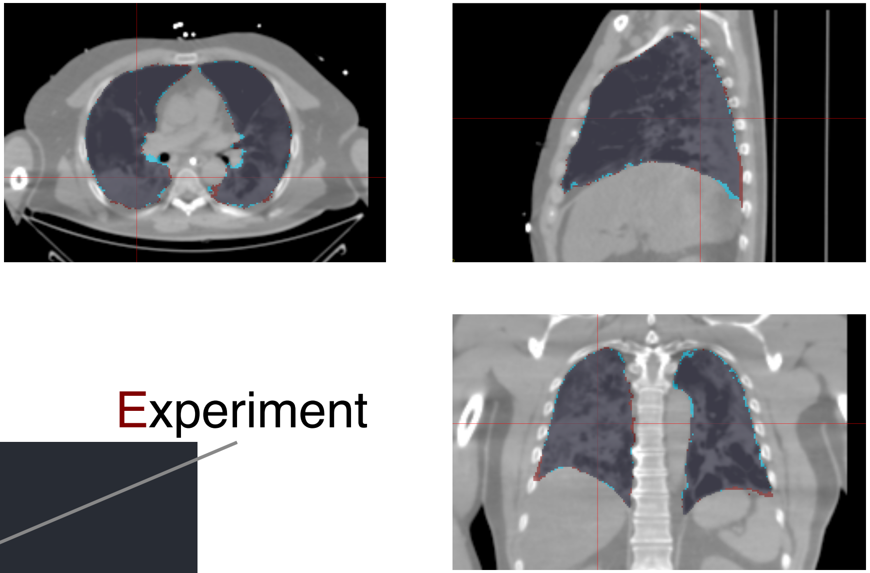
Arthur L. Samuel, *AI pionneer, 1959*

A short historical background

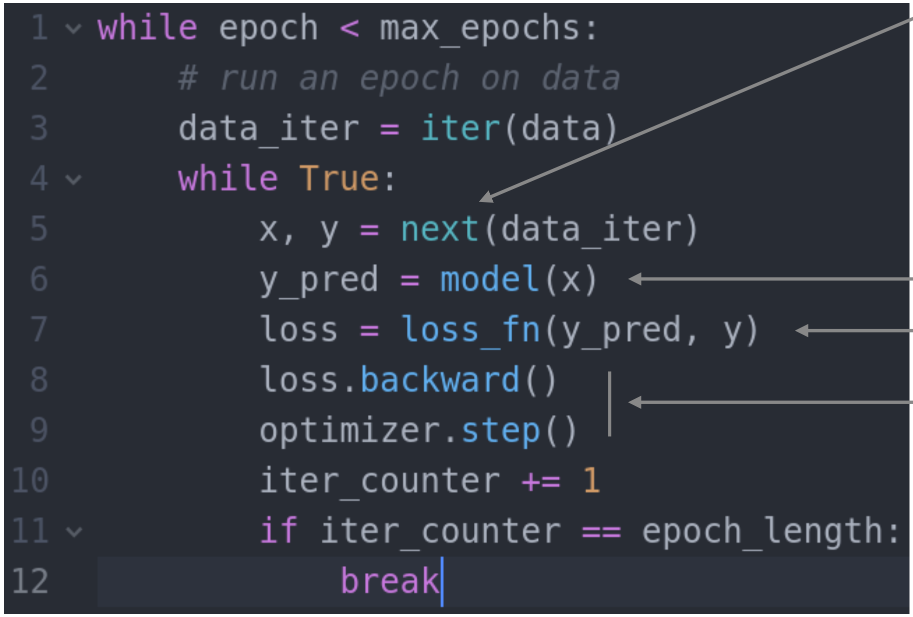
> *“ A computer program is said to learn [...] if its performance at tasks in T, as measured by a performance indicator P, improves with experience E.*

Tom Mitchel, 1978 (tweaked citation)

A short historical background



Images from PhD student Ludmilla Penarrubia



```
while epoch < max_epochs:
    # run an epoch on data
    data_iter = iter(data)
    while True:
        x, y = next(data_iter)
        y_pred = model(x)
        loss = loss_fn(y_pred, y)
        loss.backward()
        optimizer.step()
        iter_counter += 1
        if iter_counter == epoch_length:
            break
```

Experiment

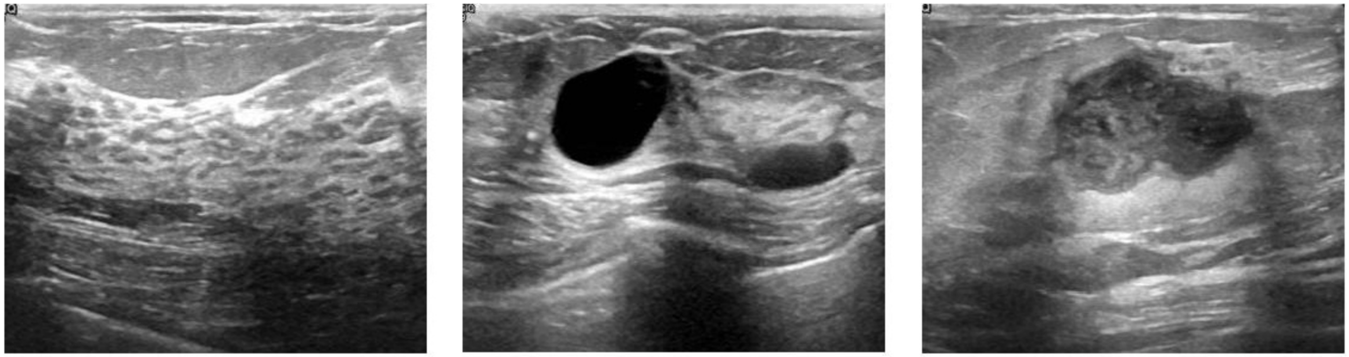
Task

Performance

Learn

A short historical background

supervised learning



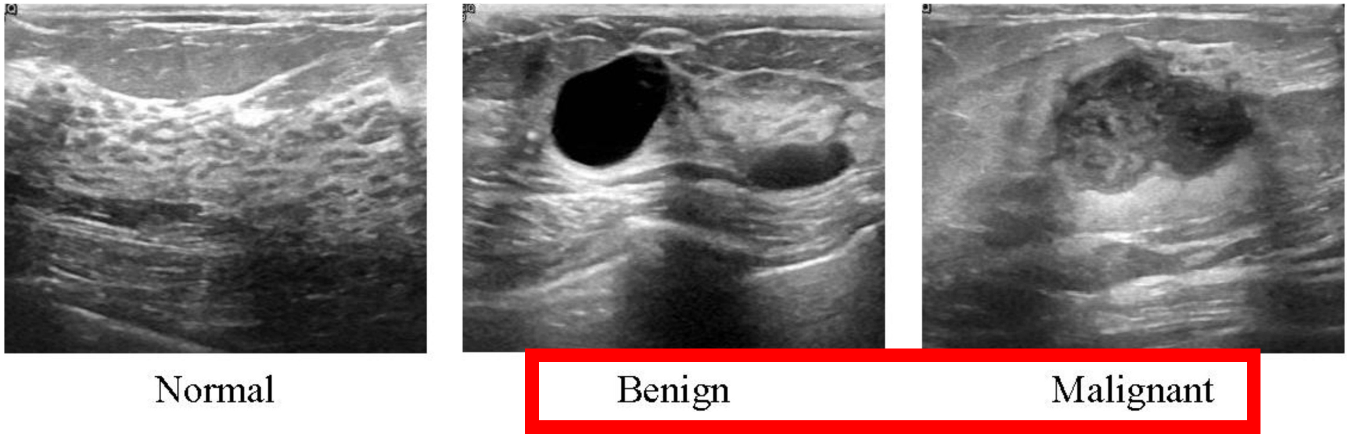
Normal Benign Malignant

Al-Dhabyani W, Gomaa M, Khaled H, Fahmy A. Dataset of breast ultrasound images. Data in Brief. 2020 Feb;28:104863. DOI: 10.1016/j.dib.2019.104863.

A short historical background

CREATIS

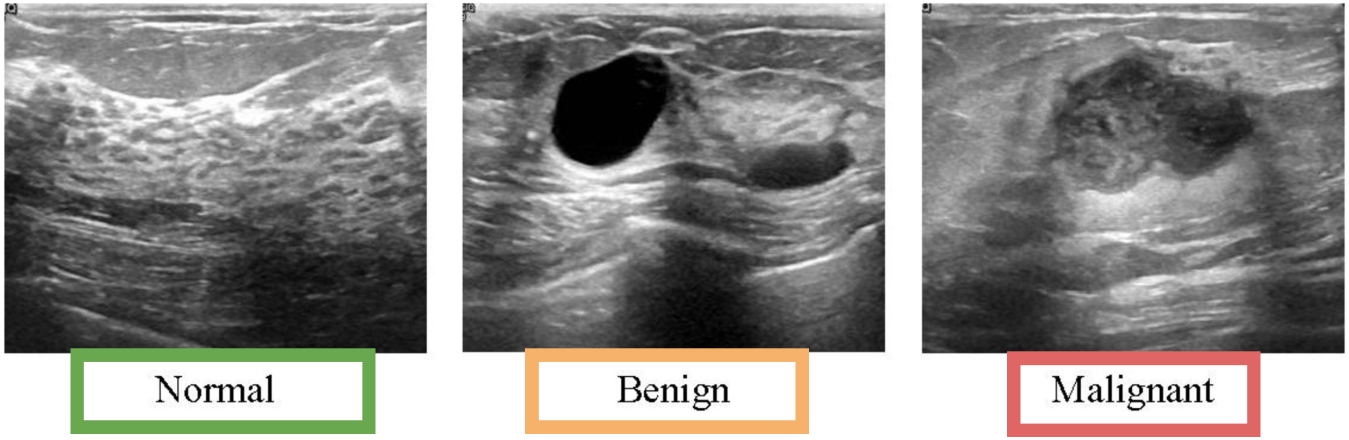
supervised learning



Al-Dhabyani W, Gomaa M, Khaled H, Fahmy A. Dataset of breast ultrasound images. Data in Brief. 2020 Feb;28:104863. DOI: 10.1016/j.dib.2019.104863.

A short historical background

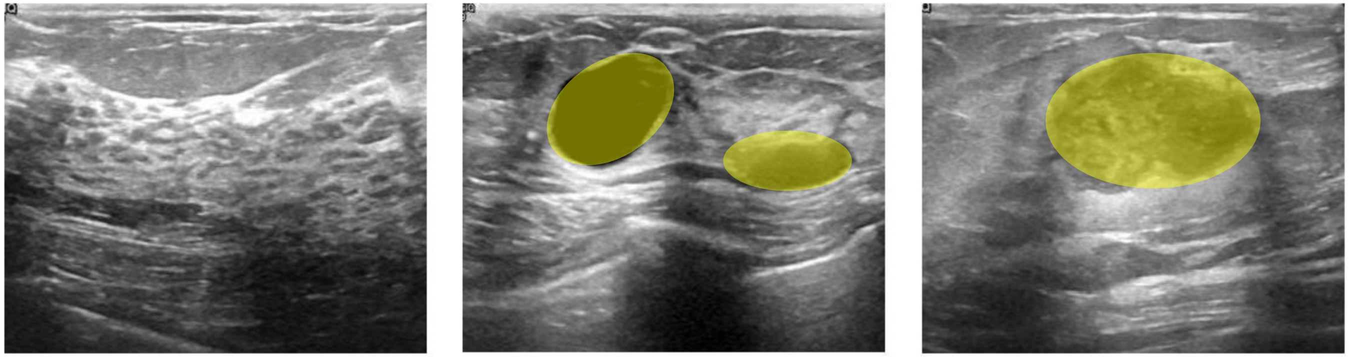
supervised learning



Al-Dhabyani W, Gomaa M, Khaled H, Fahmy A. Dataset of breast ultrasound images. Data in Brief. 2020 Feb;28:104863. DOI: 10.1016/j.dib.2019.104863.

A short historical background

supervised learning



Normal

Benign

Malignant

Al-Dhabyani W, Gomaa M, Khaled H, Fahmy A. Dataset of breast ultrasound images. Data in Brief. 2020 Feb;28:104863. DOI: 10.1016/j.dib.2019.104863.

A short historical background

supervised learning

unsupervised learning

- detection (lesions)
- classification (benign/malign)
- segmentation (organs)
- prediction (prognostic)
- ...

A short historical background

supervised learning

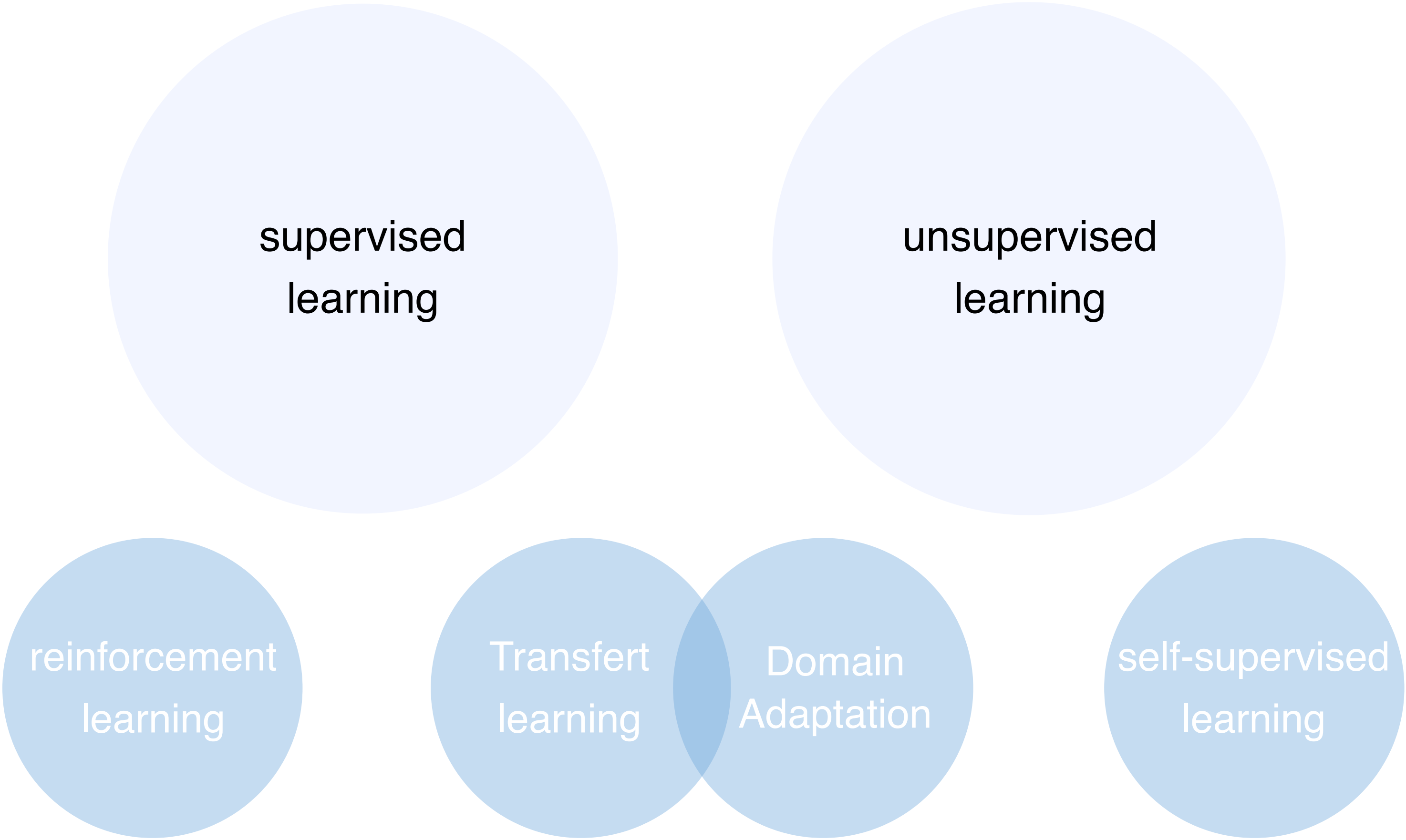
unsupervised learning

- detection (lesions)
- classification (benign/malign)
- segmentation (organs)
- prediction (prognostic)
- ...

- clustering
- dimension reduction
- representation
- density estimation
- ...

Image from PhD student Yamil Vindas

A short historical background



supervised learning

unsupervised learning

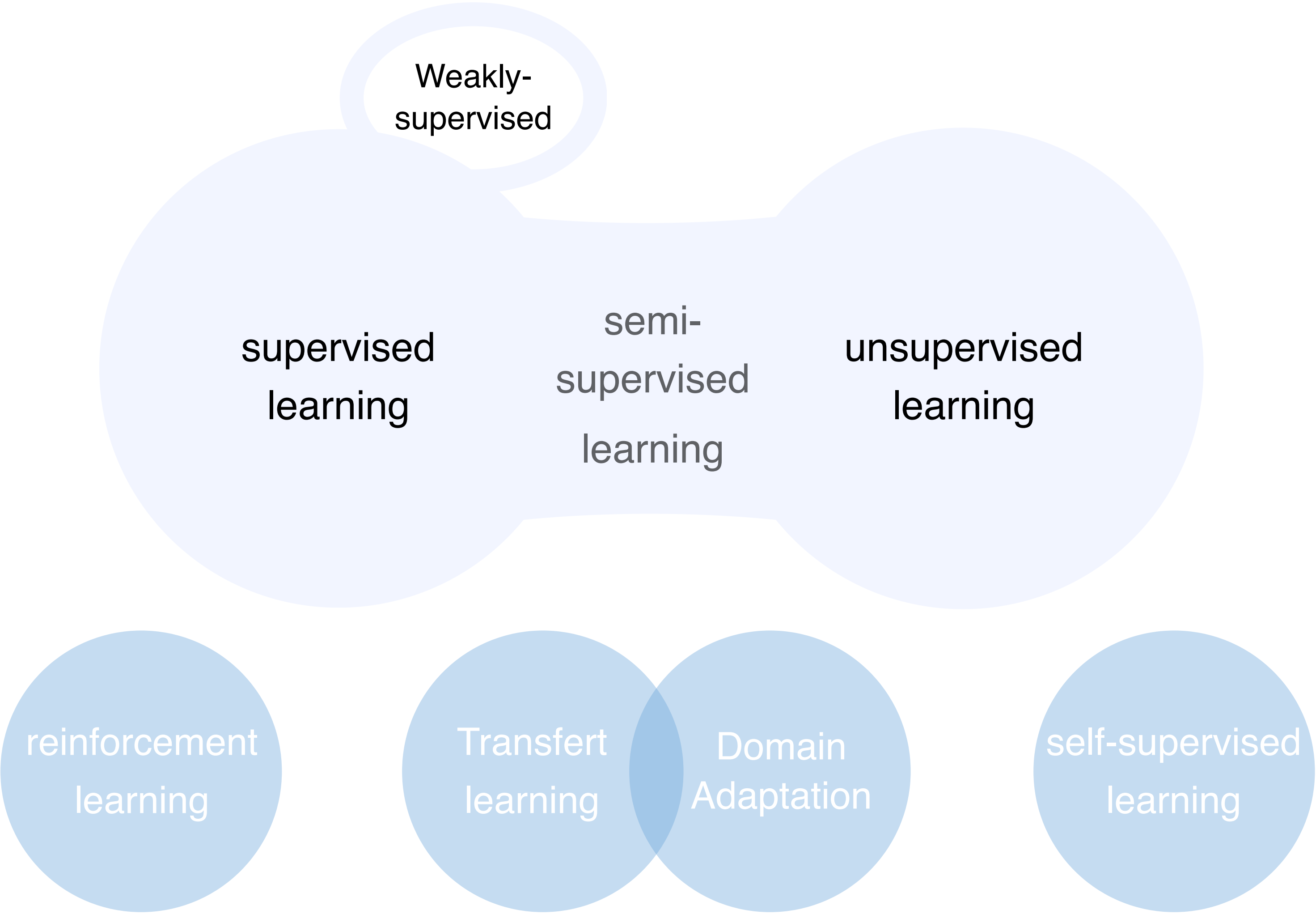
reinforcement learning

Transfert learning

Domain Adaptation

self-supervised learning

A short historical background



supervised learning

supervised
learning

Supervised Learning

Supervised machine learning

A. Introduction

B. Choice of machine learning algorithm

C. Machine learning pipeline

1. Training
2. Evaluation
3. Model selection

D. Special considerations in medical applications

Supervised machine learning

A. **Introduction**

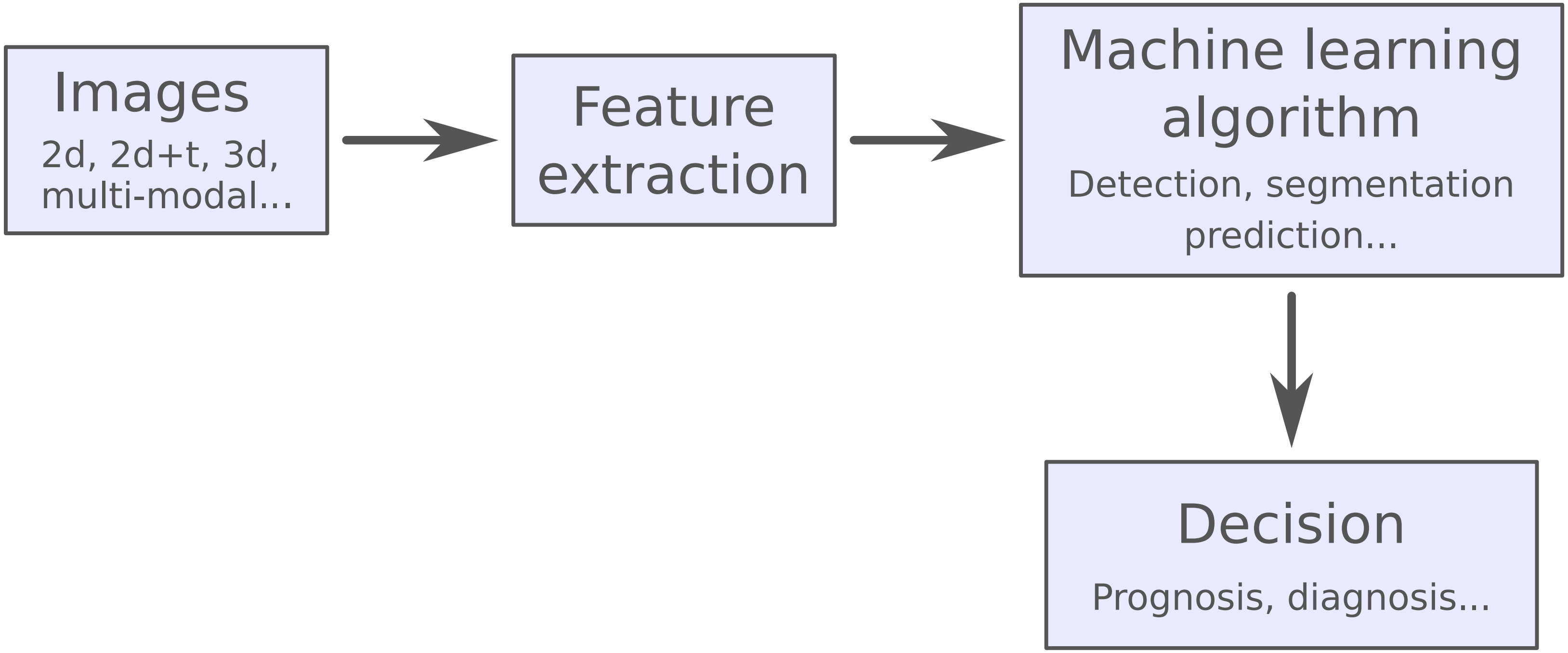
B. **Choice of machine learning algorithm**

C. **Machine learning pipeline**

1. Training
2. Evaluation
3. Model selection

D. **Special considerations in medical applications**

Introduction



Images

2d, 2d+t, 3d, multi-modal...

Feature extraction

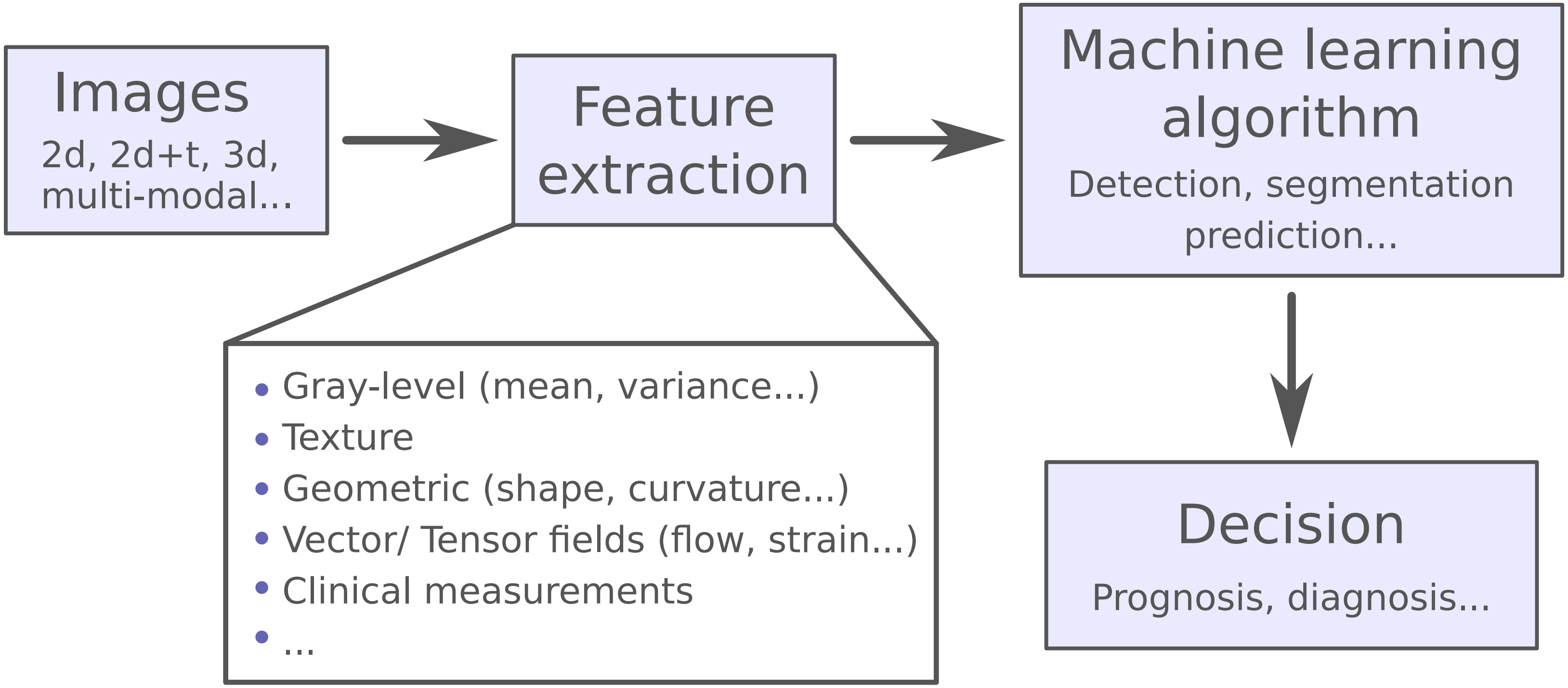
Machine learning algorithm

Detection, segmentation prediction...

Decision

Prognosis, diagnosis...

Introduction



Images

2d, 2d+t, 3d, multi-modal...

Feature extraction

- Gray-level (mean, variance...)
- Texture
- Geometric (shape, curvature...)
- Vector/ Tensor fields (flow, strain...)
- Clinical measurements
- ...

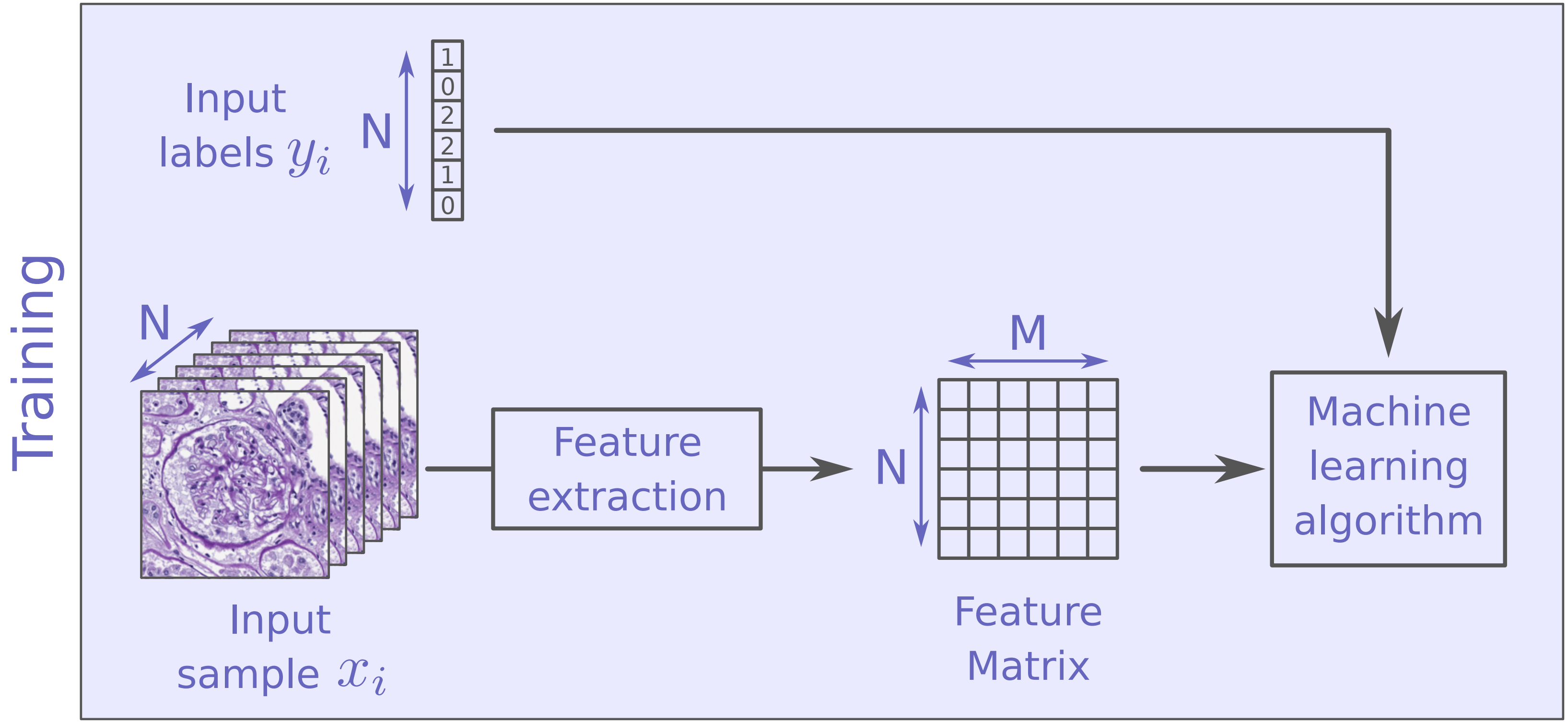
Machine learning algorithm

Detection, segmentation prediction...

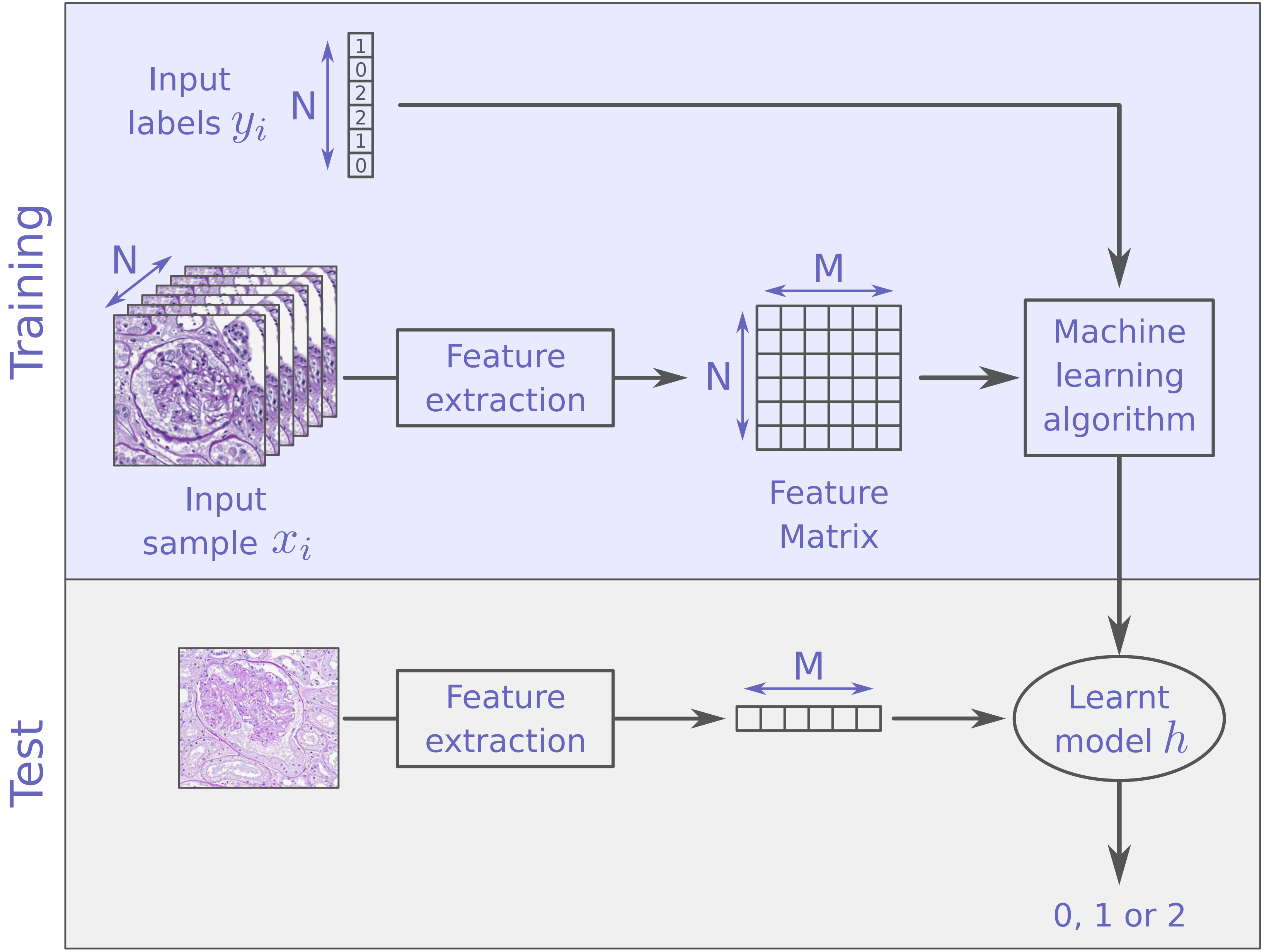
Decision

Prognosis, diagnosis...

Supervised machine learning pipeline



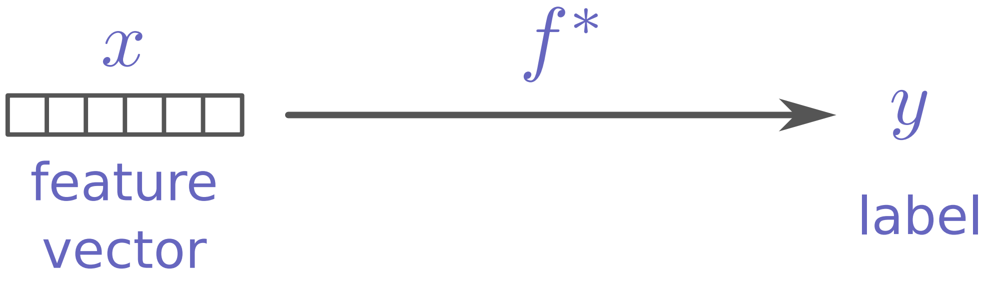
Supervised machine learning pipeline



What is supervised learning ?

Let $f^* : X \mapsto Y$ be an unknown function such as $\forall x \in X$ and $\forall y \in Y$:

$$y = f^*(x)$$



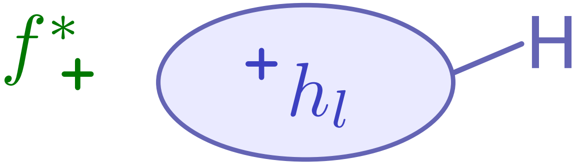
What is supervised learning ?

Let $f^* : X \mapsto Y$ be an unknown function such as $\forall x \in X$ and $\forall y \in Y$:

$$y = f^*(x)$$

- **Definition**

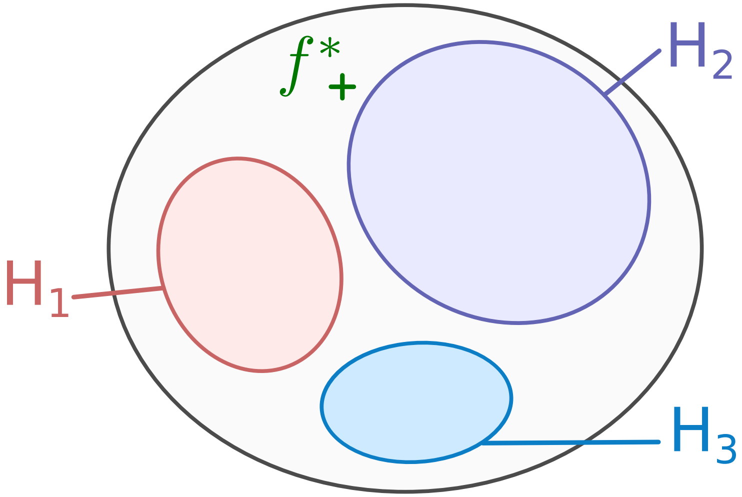
  Supervised learning is the task of learning a function $h_L \in H$ ($h_L : X \mapsto Y$), called a **hypothesis** that best approximates f^* based on a **dataset** $\mathcal{D}$ of N input/output pairs ($\mathcal{D} = \{x_i, y_i\}_{1 \leqslant i \leqslant N}$)



- H is called the **hypothesis space**
- h_l may also be called a **predictor** or a **model**

How to learn from data ?

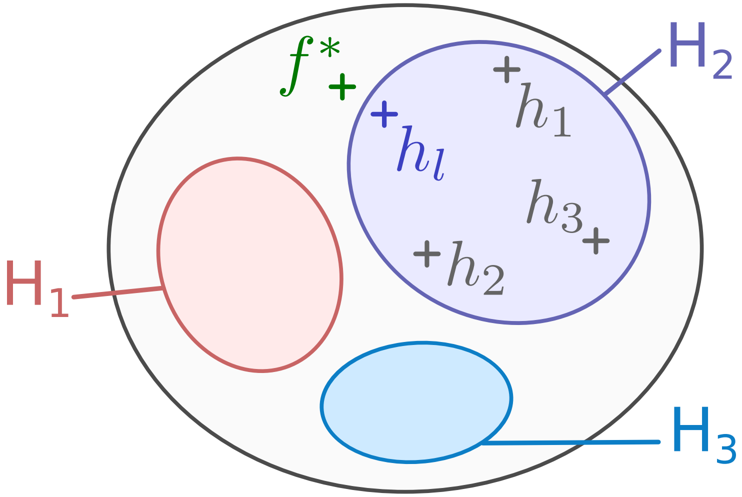
- **Choose the type of algorithm** (*i.e.* the hypothesis space H)



How to learn from data ?

- **Choose the type of algorithm** (*i.e.* the hypothesis space H)

- **Train a model** (*i.e.* find the best $h_l \in H$)
  - What is a good model ?

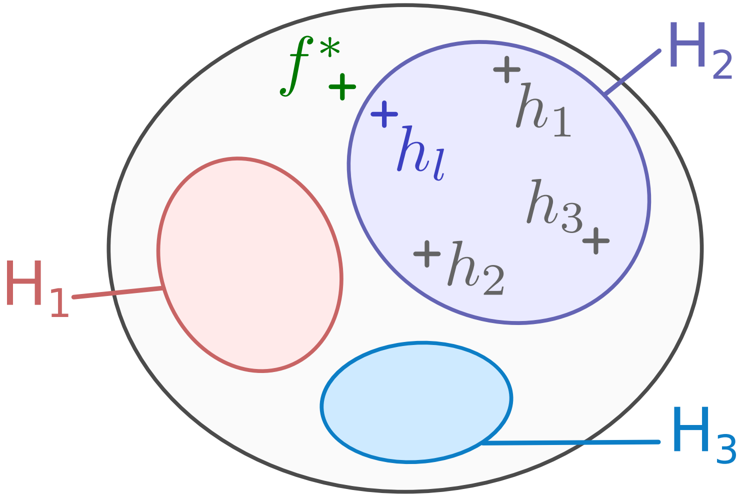


How to learn from data ?

- **Choose the type of algorithm** (*i.e.* the hypothesis space H)

- **Train a model** (*i.e.* find the best $h_l \in H$)
  - What is a good model ?

- **Evaluate the model**
  - Evaluation metrics



Supervised machine learning

A. Introduction

B. Choice of machine learning algorithm

C. Machine learning pipeline

1. Training
2. Evaluation
3. Model selection

D. Special considerations in medical applications

Classification vs Regression

Task: Learn $h : X \mapsto Y$ based on a dataset $\mathcal{D} = \{x_i, y_i\}_{1 \leqslant i \leqslant N}$

Two different tasks depending on the type of label y_i:

Classification vs Regression

Task: Learn $h : X \mapsto Y$ based on a dataset $\mathcal{D} = \{x_i, y_i\}_{1 \leqslant i \leqslant N}$

Two different tasks depending on the type of label y_i:

- **Classification**: $y_i \in \mathbb{N}$

  Example: Does the image contain a malignant melanoma ?

$\xrightarrow{h}$ 0 or 1

(no) (yes)

Classification vs Regression

Task: Learn $h : X \mapsto Y$ based on a dataset $\mathcal{D} = \{x_i, y_i\}_{1 \leqslant i \leqslant N}$

Two different tasks depending on the type of label y_i:

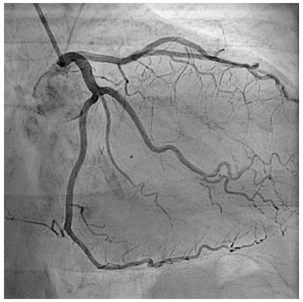
- **Classification**: $y_i \in \mathbb{N}$

  Example: Does the image contain a malignant melanoma ?

$\xrightarrow{h}$ 0 (no) or 1 (yes)

- **Regression**: $y_i \in \mathbb{R}$

  Example: FFR (Fractional Flow Reserve) prediction from a coronary angiography.



$\xrightarrow{h}$ [0, 1]

Various types of models

Choose a type of model (*i.e.* the hypothesis space H):

- **Linear models**
  - Naive Bayes
  - Logistic regression
  - Perceptron
  - Linear Discriminant Analysis (LDA)
- **Support Vector Machine (SVM)**
- **K Nearest Neighbors**
- **Decision Tree**
- **Neural networks**

How to make the choice ?

- There is no “best” algorithm that will work on any dataset
  ⟶ **”No free lunch” theorems** [1]

[1] Wolpert, D. H., ”The lack of a priori distinctions between learning algorithms”, Neural computation, 1996

How to make the choice ?

- There is no “best” algorithm that will work on any dataset
  $\longrightarrow$ **”No free lunch” theorems** [1]

The choice of a “good” machine learning algorithm depends on:

- The complexity of the unknown targeted function f^*
- The amount of labeled data
- The dimension of the input space X
- The amount of noise in the data and labels
- ...

[1] Wolpert, D. H., ”The lack of a priori distinctions between learning algorithms”, Neural computation, 1996

Supervised machine learning

A. Introduction

B. Choice of machine learning algorithm

C. **Machine learning pipeline**

1. Training
2. Evaluation
3. Model selection

D. Special considerations in medical applications

Parameters vs hyperparameters

Once H is chosen, learning a model h consists in finding the best $h \in H$ given a dataset. A model h is defined by:

Parameters vs hyperparameters

Once H is chosen, learning a model h consists in finding the best $h \in H$ given a dataset. A model h is defined by:

- **A set of parameters** Θ_1

  The parameters of a model are learnt from the data.

  Examples:

  - The weight values in neural networks
  - The support vectors in SVM
  - The split values in decision trees

Parameters vs hyperparameters

Once H is chosen, learning a model h consists in finding the best $h \in H$ given a dataset. A model h is defined by:

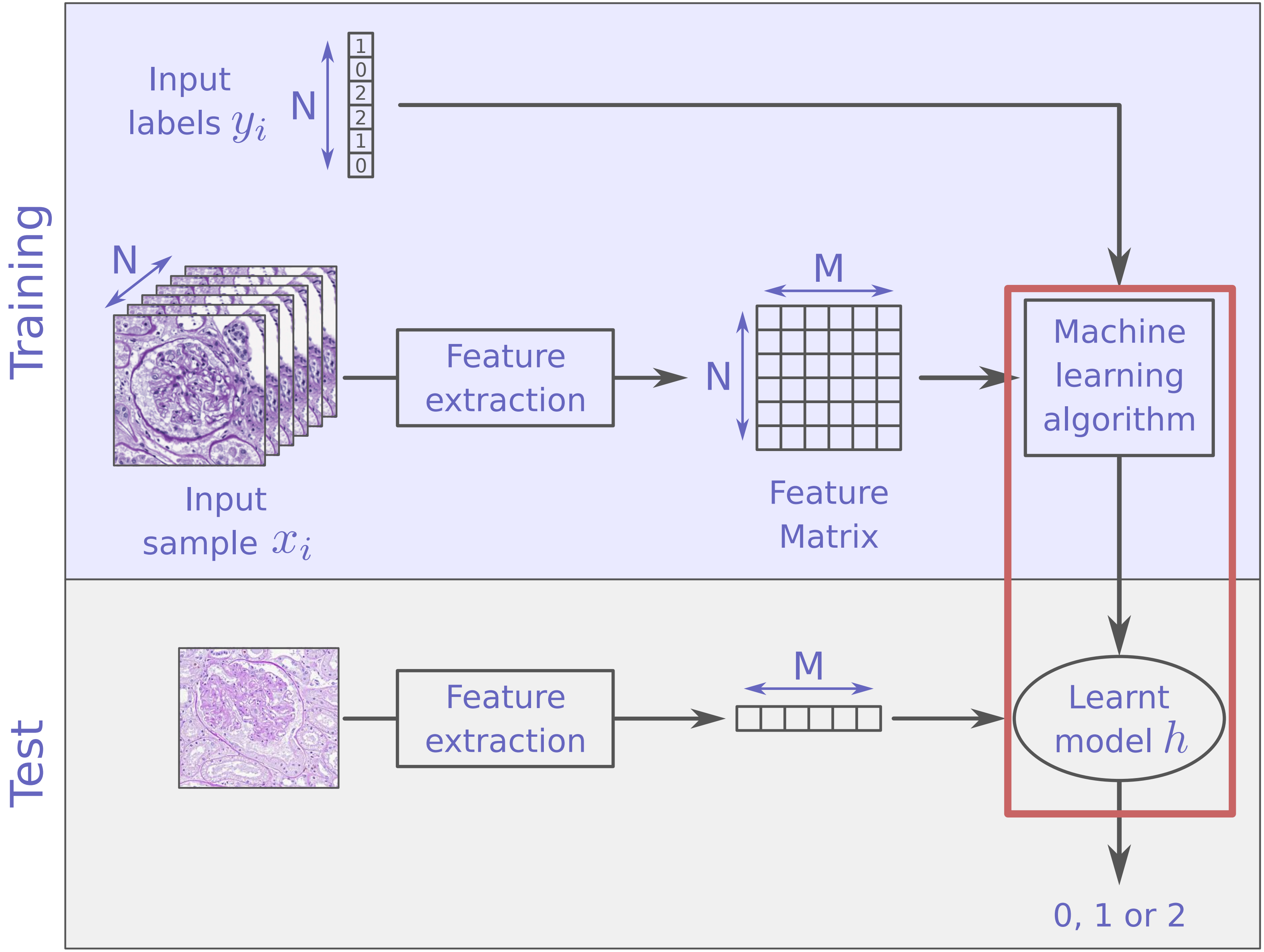
- **A set of parameters Θ_1**
  The parameters of a model are learnt from the data.

  Examples:
  - The weight values in neural networks
  - The support vectors in SVM
  - The split values in decision trees

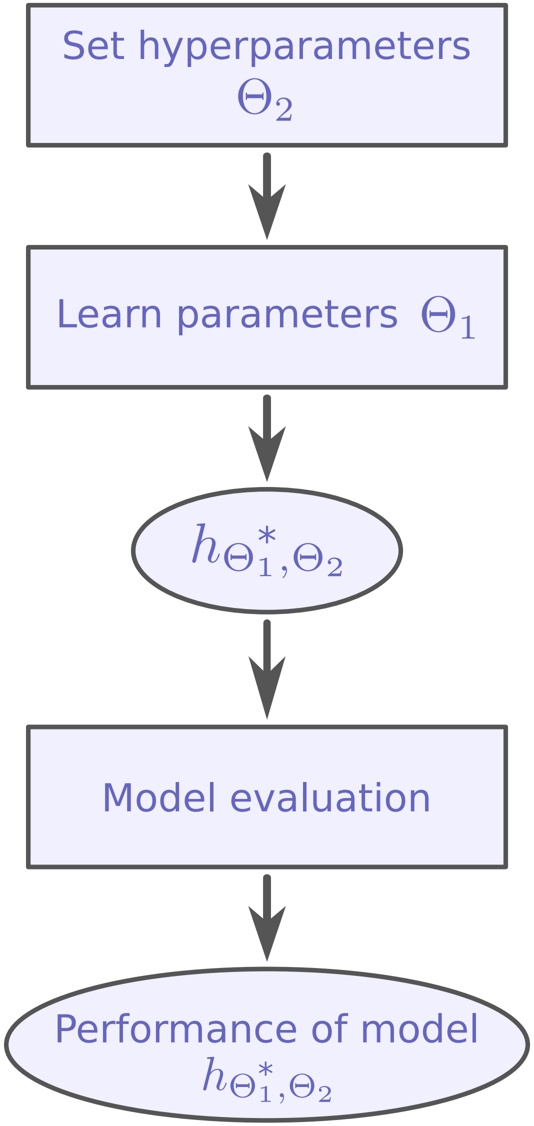
- **A set of hyperparameters Θ_2**
  The hyperparameters cannot be learnt from the data. They have to be set before training the model.

  Examples:
  - The number of trees in a random forest
  - The learning rate in neural networks
  - The number of neighbors ”k” in KNN

Machine Learning pipeline



Machine Learning pipeline



Set hyperparameters Θ_2

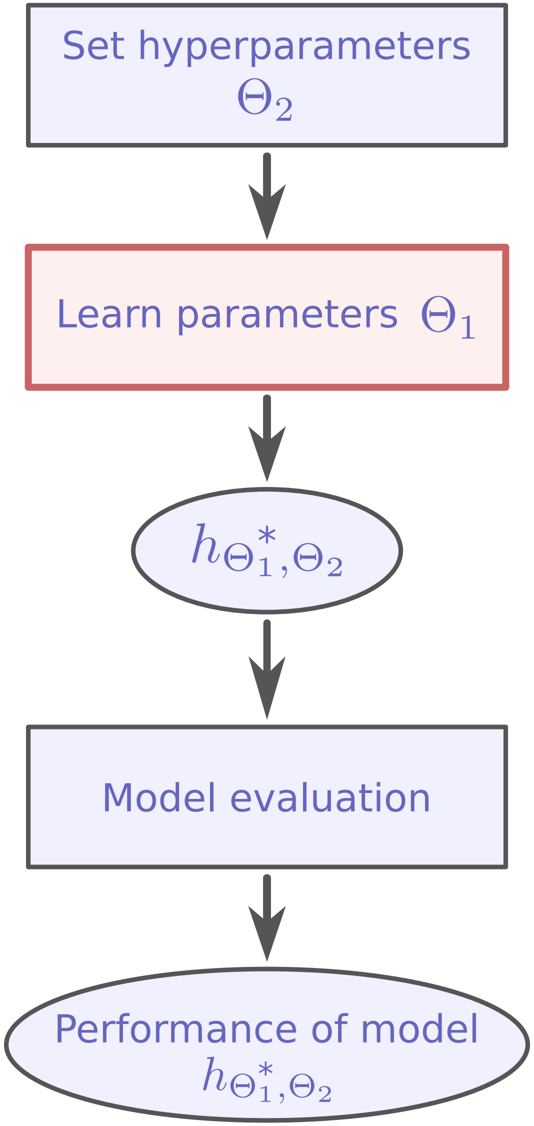
Learn parameters Θ_1

$h_{\Theta_1^*,\Theta_2}$

Model evaluation

Performance of model $h_{\Theta_1^*,\Theta_2}$

Machine Learning pipeline



Set hyperparameters Θ_2

Learn parameters Θ_1

$h_{\Theta_1^*,\Theta_2}$

Model evaluation

Performance of model $h_{\Theta_1^*,\Theta_2}$

Supervised machine learning

A. Introduction

B. Choice of machine learning algorithm

C. Machine learning pipeline

1. Training
2. Evaluation
3. Model selection

D. Special considerations in medical applications

Loss function

Loss function

The loss function L ($L : Y \times Y \mapsto \mathbb{R}^+$) associates a cost to the prediction $\tilde{y}_i = h(x_i)$ of a model h compared to its true label $y_i = f^*(x_i)$.

Loss function

The loss function L ($L : Y \times Y \mapsto \mathbb{R}^+$) associates a cost to the prediction $\tilde{y}_i = h(x_i)$ of a model h compared to its true label $y_i = f^*(x_i)$.

Examples:

- **Binary loss** for classification

$$L(y_i, \tilde{y}_i) = \begin{cases} 1 & \text{if } y_i \neq \tilde{y}_i \\ 0 & \text{otherwise} \end{cases}$$

- **Quadratic loss** for regression

$$L(y_i, \tilde{y}_i) = (y_i - \tilde{y}_i)^2$$

Real risk and model error

- **Real Risk**

  Let assume that $\{x_i, y_i\}_{1 \leqslant i \leqslant N}$ is drawn from a joint probability distribution $P(x, y)$ over X and Y.

  The **Real risk** $R(h)$ of a hypothesis h is:

  $$R(h) = \mathbb{E}[L(h(x), y)] = \int_{X \times Y} L(h(x), y) \, \mathrm{d}P(x, y)$$

Real risk and model error

Real Risk

Let assume that $\{x_i, y_i\}_{1 \leqslant i \leqslant N}$ is drawn from a joint probability distribution $P(x, y)$ over X and Y.

The **Real risk** $R(h)$ of a hypothesis h is:

$$R(h) = \mathbb{E}[L(h(x), y)] = \int_{X \times Y} L(h(x), y) \, \mathrm{d}P(x, y)$$

Model error

- f^*: the unknown function we want to learn
- h_l: the model we learn from dataset $\mathcal{D}$

The **model error** is defined as:

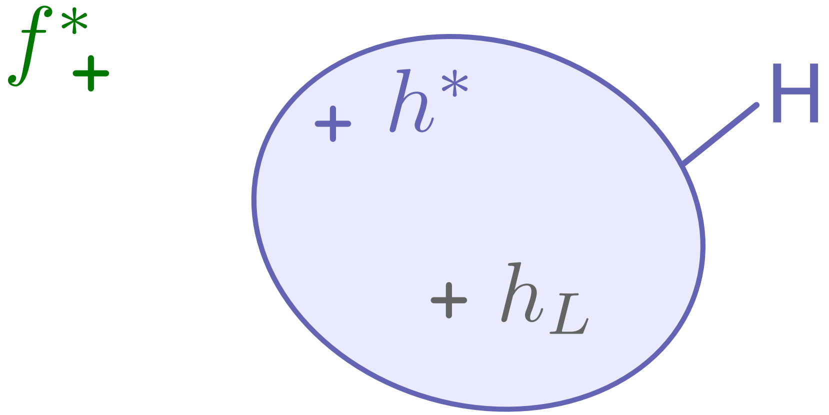
$$\boxed{\text{Error} = R(h_l) - R(f^*)}$$

Remark: We usually assume that $R(f^*) = 0$ (deterministic model)

Error decomposition

- f^*: the unknown function we want to learn
- h^*: the optimal model in H
- h_l: the model we learn from dataset $\mathcal{D}$

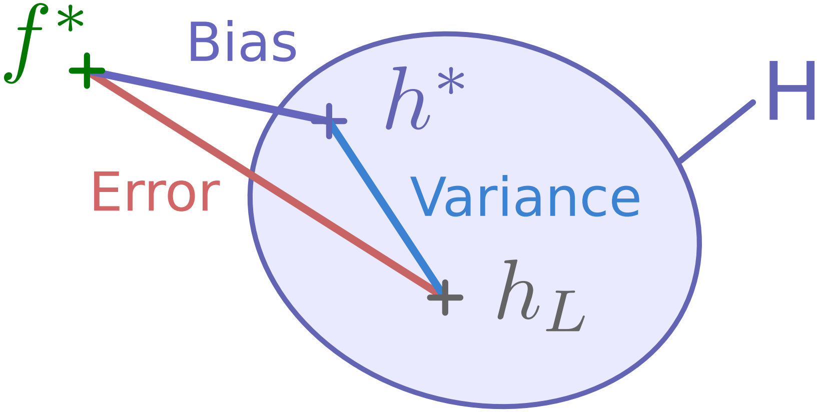
$$\underbrace{R(h_l) - R(f^*)}_{\text{Error}}$$



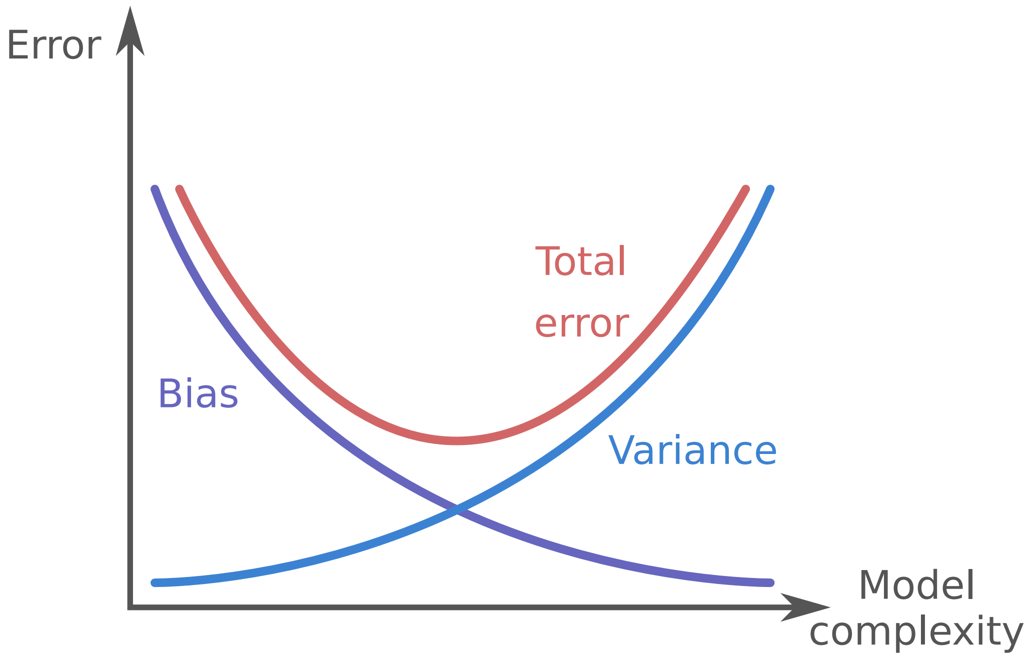
Error decomposition

- f^*: the unknown function we want to learn
- h^*: the optimal model in H
- h_l: the model we learn from dataset $\mathcal{D}$

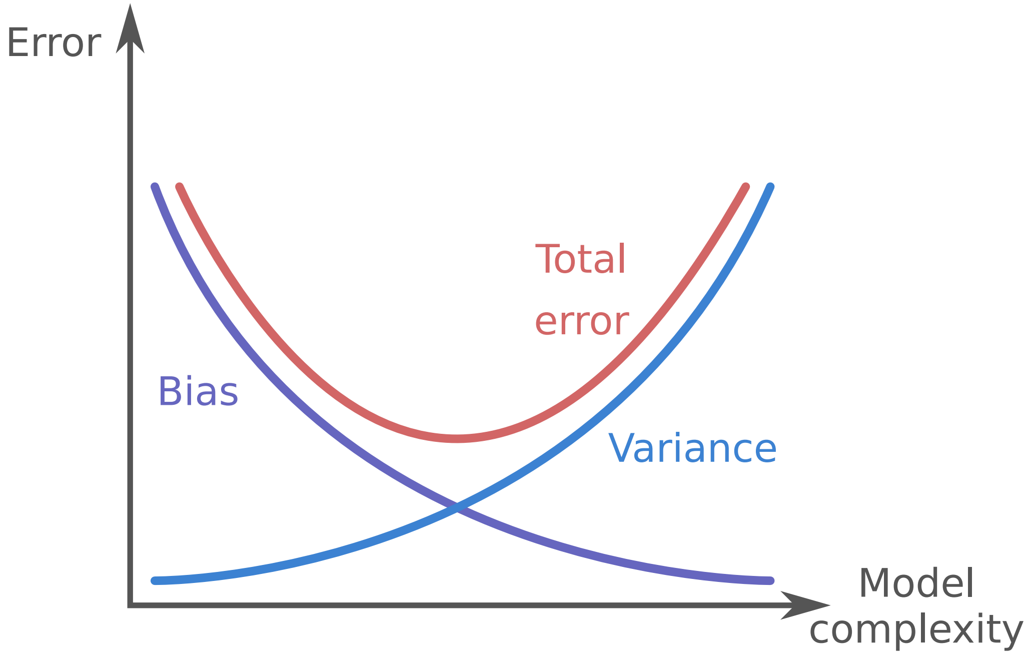
$$\underbrace{R(h_l) - R(f^*)}_{\text{Error}} = \underbrace{R(h_l) - R(h^*)}_{\text{Variance}} + \underbrace{R(h^*) - R(f^*)}_{\text{Bias}}$$

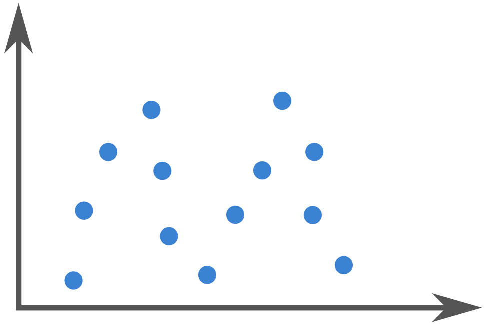


Bias / Variance trade off

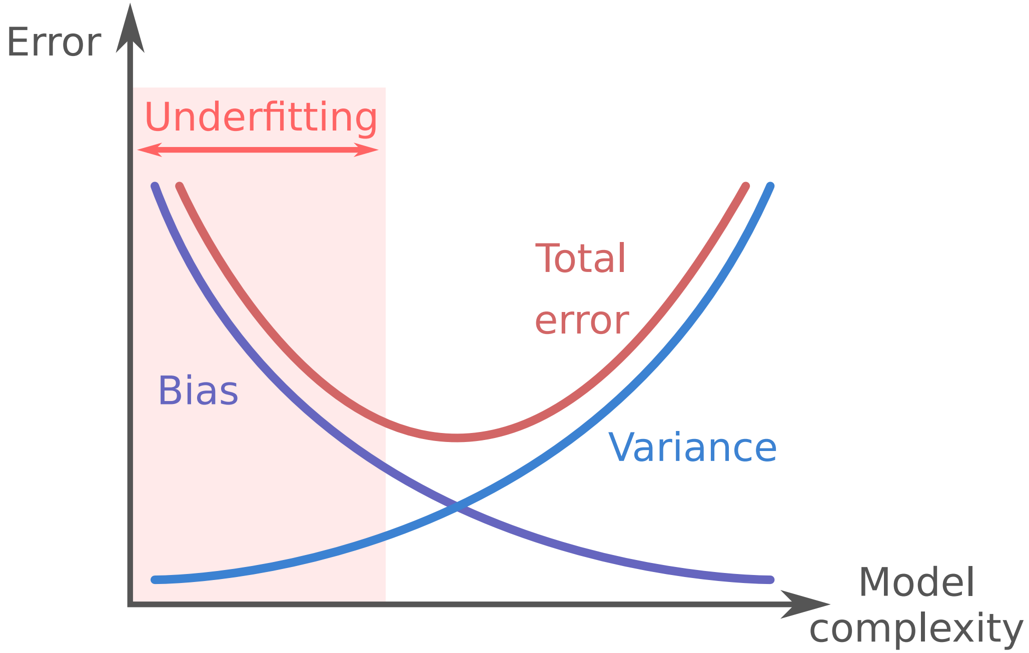


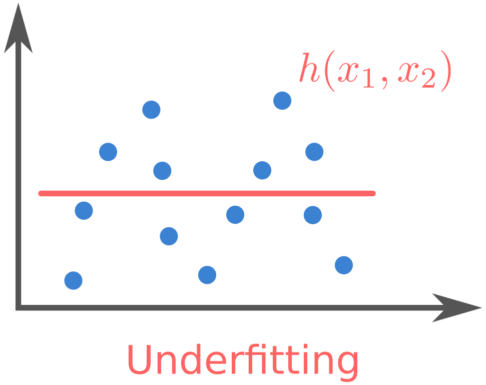
Bias / Variance trade off



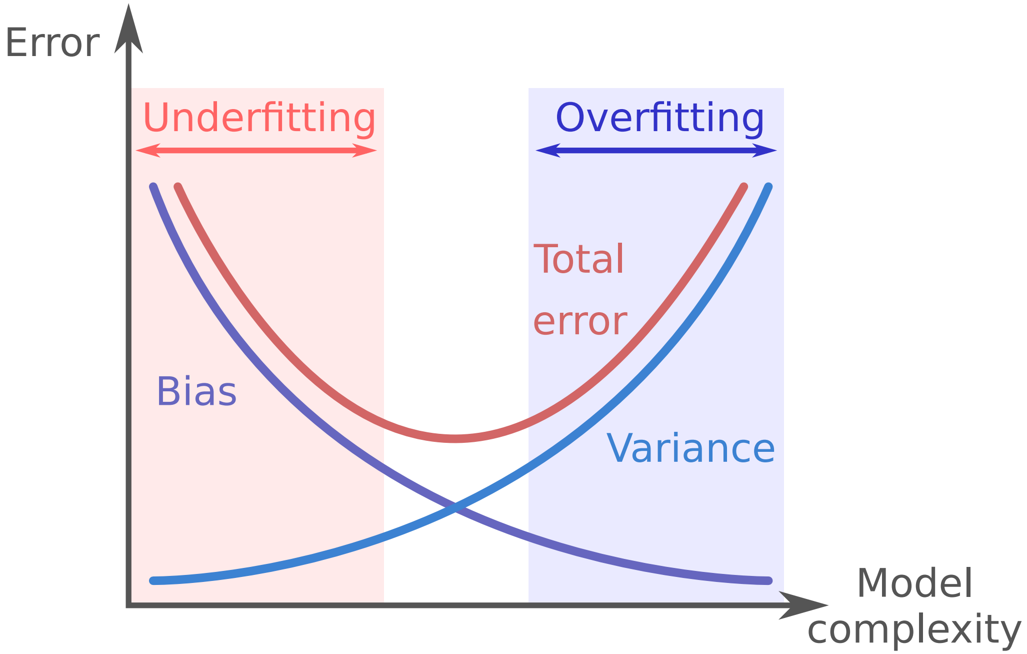


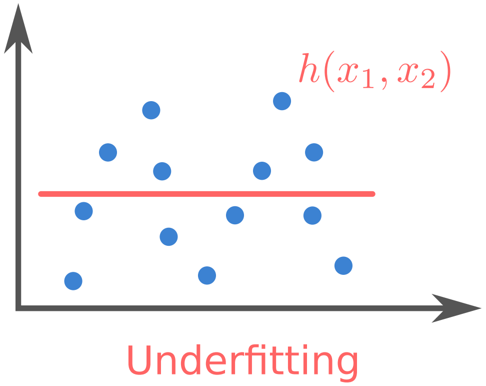
Bias / Variance trade off

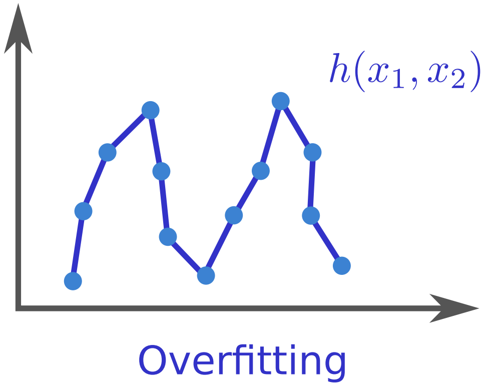




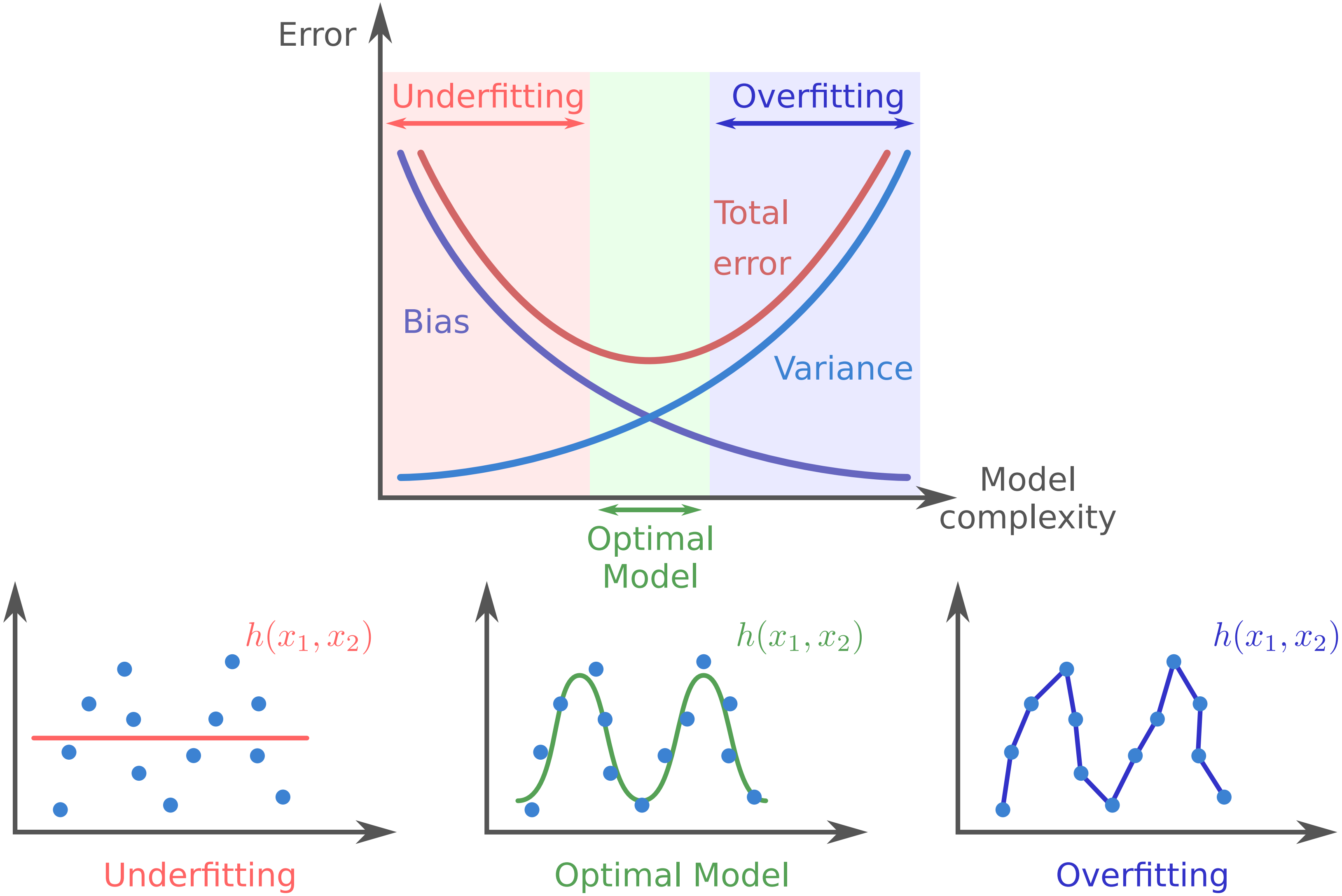
Bias / Variance trade off







Bias / Variance trade off



Real risk *vs.* Empirical risk

Learning a model is finding its best set of parameters Θ_1, which is done by minimizing the model error (= Real Risk)

- **Real Risk**

$$R(h) = \int_{X \times Y} L(h(x), y) \, \mathrm{d}P(x, y)$$

Real risk *vs.* Empirical risk

Learning a model is finding its best set of parameters Θ_1, which is done by minimizing the model error ($=$ Real Risk)

- **Real Risk**

$$R(h) = \int_{X \times Y} L(h(x), y)\, \mathrm{d}P(x, y)$$

$\longrightarrow$ **In practice $P(x, y)$ is not known.**

Real risk *vs.* Empirical risk

Learning a model is finding its best set of parameters Θ_1, which is done by minimizing the model error ($=$ Real Risk)

- **Real Risk**

$$R(h) = \int_{X \times Y} L(h(x), y) \, \mathrm{d}P(x, y)$$

$\longrightarrow$ **In practice $P(x, y)$ is not known.**

- **Empirical Risk**
  Approximation of the real risk over a dataset $\mathcal{D} = \{x_i, y_i\}_{1 \leqslant i \leqslant N}$

$$\boxed{R_{\text{emp}}(h) = \frac{1}{|N|} \sum_{x,y \in \mathcal{D}} L(h(x), y)}$$

Real risk *vs.* Empirical risk

Learning a model is finding its best set of parameters Θ_1, which is done by minimizing the model error (= Real Risk)

- **Real Risk**

$$R(h) = \int_{X \times Y} L(h(x), y) \, \mathrm{d}P(x, y)$$

$\longrightarrow$ **In practice $P(x, y)$ is not known.**

- **Empirical Risk**

  Approximation of the real risk over a dataset $\mathcal{D} = \{x_i, y_i\}_{1 \leqslant i \leqslant N}$

$$\boxed{R_{\text{emp}}(h) = \frac{1}{|N|} \sum_{x,y \in \mathcal{D}} L(h(x), y)}$$

$$R_{\text{emp}}(h_l) \underset{N \to +\infty}{\longrightarrow} R(h_l)$$

Empirical Risk Minimization

- In theory, learning a model is minimizing the error $R(h_l)$
- In practice, we cannot compute $R(h_l)$ so we minimize $R_{\mathrm{emp}}(h_l)$

  $\longrightarrow$This is called **Empirical Risk Minimization**

- **Empirical Risk Minimization (ERM)**

$$\boxed{h_l = \underset{h \in H}{\arg\min} \frac{1}{|N|} \sum_{x,y \in \mathcal{D}} L(h(x), y)}$$

where $\mathcal{D} = \{x_i, y_i\}_{1 \leqslant i \leqslant N}$ is the **training dataset**

Supervised machine learning

A. Introduction

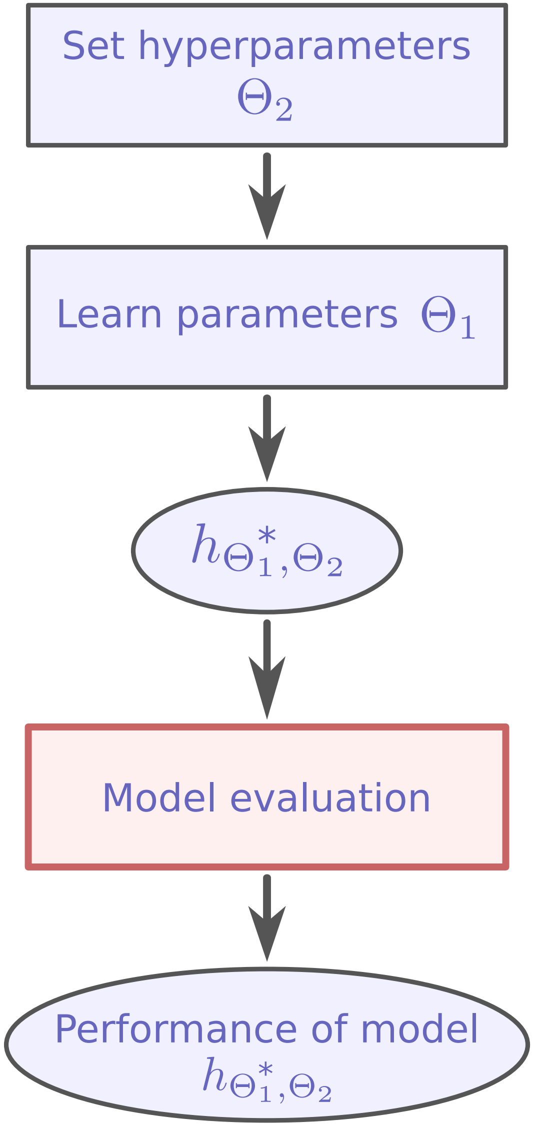
B. Choice of machine learning algorithm

C. Machine learning pipeline

1. Training
2. Evaluation
3. Model selection

D. Special considerations in medical applications

Machine Learning pipeline


Set hyperparameters Θ_2
Learn parameters Θ_1
$h_{\Theta_1^*,\Theta_2}$
Model evaluation
Performance of model $h_{\Theta_1^*,\Theta_2}$

Model evaluation

A good model is a model exhibiting:

- **High performance**
- A good **generalization** power when seeing new data
- **Stable** performance for small dataset variations

To select a good model, we need to validate its performance according to these 3 criteria

⟶ Choose a **validation strategy**

Validation strategies

Several validation strategies were developed:

- Hold out
- K-fold cross validation
- Leave-one-out cross validation
- Bootstrapping

Validation strategies

Several validation strategies were developed:

- Hold out
- K-fold cross validation
- Leave-one-out cross validation
- Bootstrapping

$\longrightarrow$ **They all require to split the dataset**

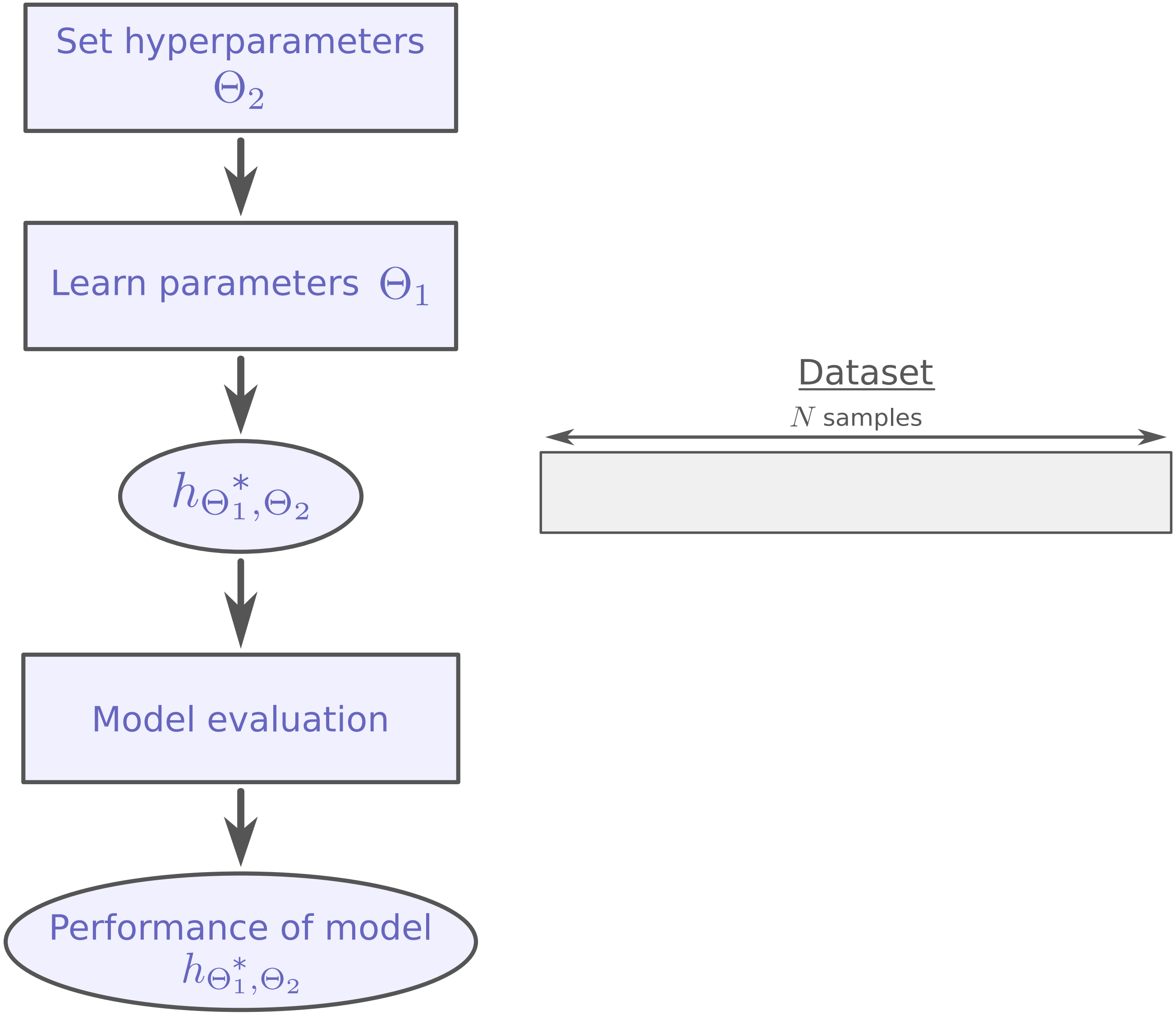
Validation strategies

Several validation strategies were developed:

- Hold out
- **K-fold cross validation**
- Leave-one-out cross validation
- Bootstrapping

⟶ **They all require to split the dataset**

Dataset splitting



Set hyperparameters Θ_2

Learn parameters Θ_1

$h_{\Theta_1^*,\Theta_2}$

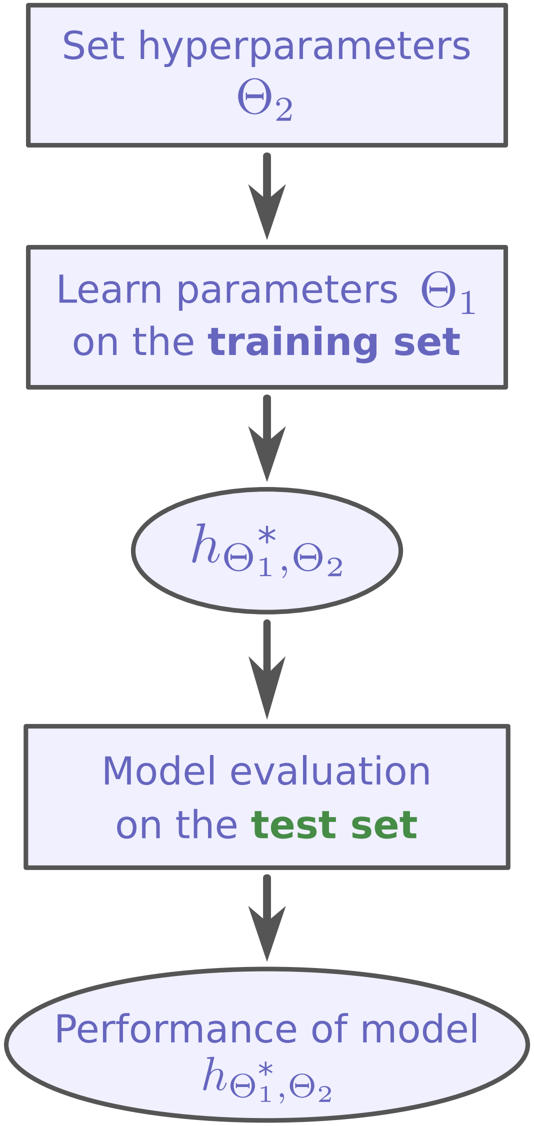
Model evaluation

Performance of model $h_{\Theta_1^*,\Theta_2}$

Dataset

N samples

Dataset splitting



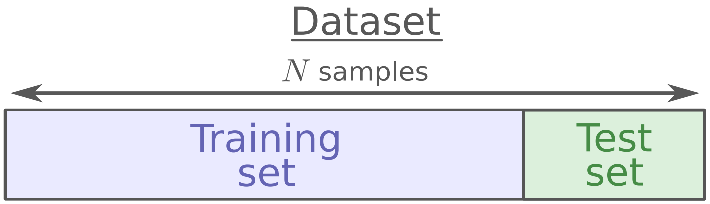
Set hyperparameters Θ_2

Learn parameters Θ_1 on the **training set**

$h_{\Theta_1^*,\Theta_2}$

Model evaluation on the **test set**

Performance of model $h_{\Theta_1^*,\Theta_2}$



Dataset

N samples

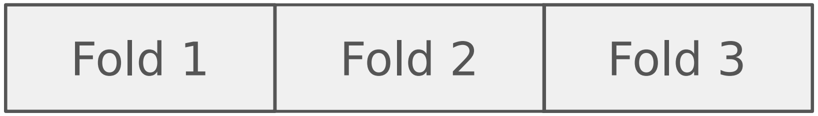
Training set

Test set

K-fold cross validation for model evaluation

<u>Goal</u>: Evaluation of the model **mean performance**, **generalization** and **stability**.

- Split the dataset in k folds

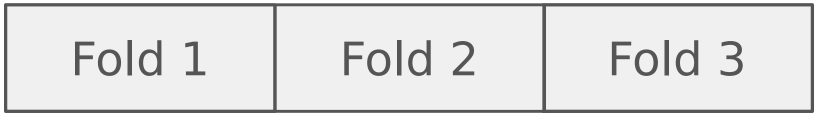


Fold 1 Fold 2 Fold 3

K-fold cross validation for model evaluation

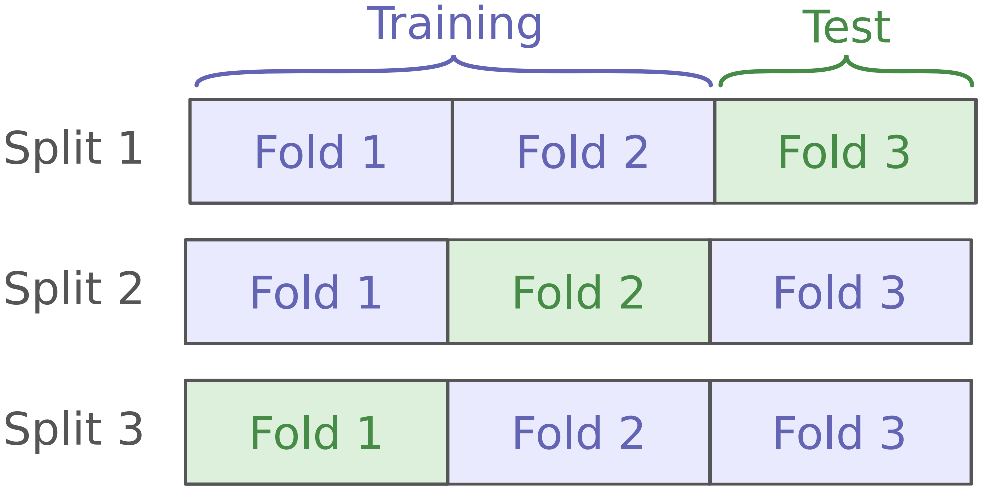
Objective: Evaluation of the model **mean performance**, **generalization** and **stability**.

- Split the dataset in k folds



Fold 1 | Fold 2 | Fold 3

- Generate the k combinations of 1 test fold and the remaining $k-1$ training folds



Training | Test

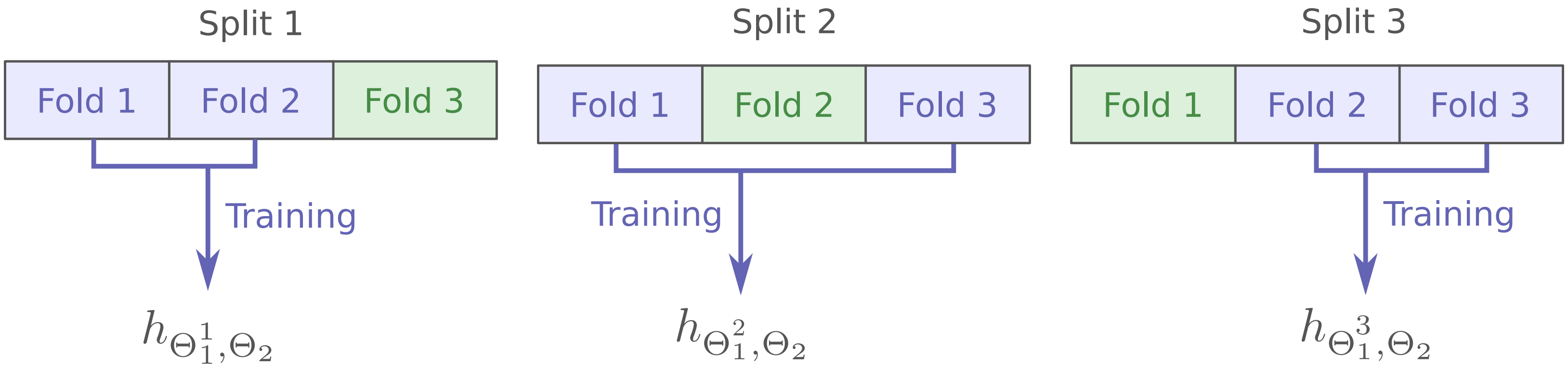
Split 1: Fold 1 | Fold 2 | Fold 3

Split 2: Fold 1 | Fold 2 | Fold 3

Split 3: Fold 1 | Fold 2 | Fold 3

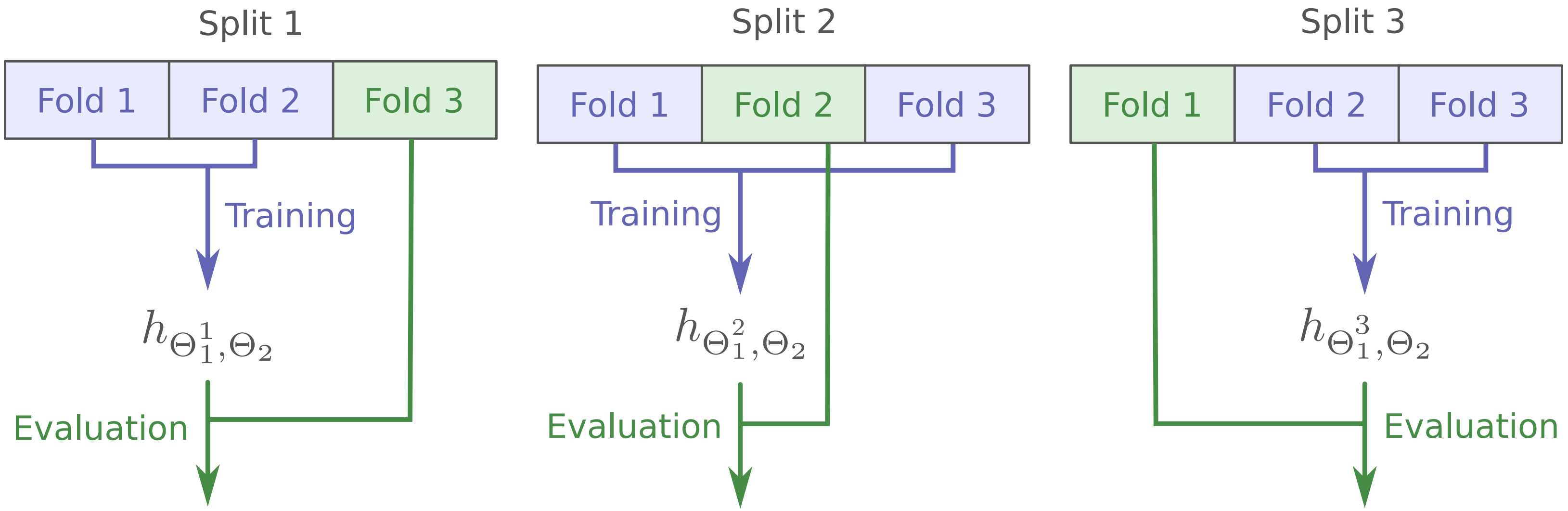
K-fold cross validation for model evaluation

- For each split, use the training folds to learn a model $h_{\Theta_1^i,\Theta_2}$ and evaluate the model on the remaining test fold.



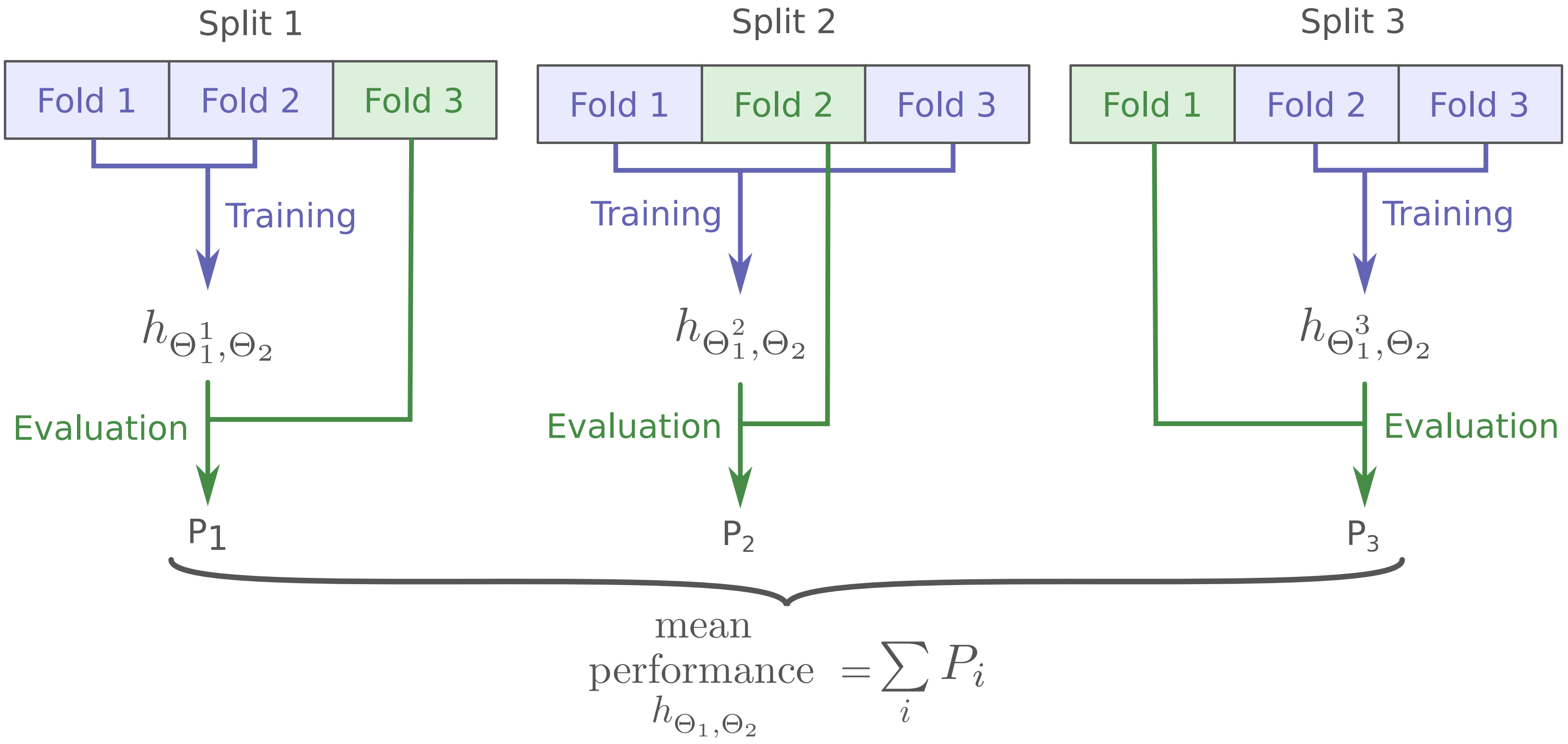
K-fold cross validation for model evaluation

- For each split, use the training folds to learn a model $h_{\Theta_1^i,\Theta_2}$ and evaluate the model on the remaining test fold.



K-fold cross validation for model evaluation

- For each split, use the training folds to learn a model $h_{\Theta_1^i,\Theta_2}$ and evaluate the model on the remaining test fold.



- Compute the mean and standard deviation of the performance of the model.

What is the performance of a model ?

- The performance of a model is assessed based on one or several **metrics**

What is the performance of a model ?

- The performance of a model is assessed based on one or several **metrics**

Examples of popular metrics:

- **Regression metrics**
  - Mean Square Error (MSE)
  - Root Mean Square Error (RMSE)
  - Peak Signal-to-Noise Ratio (PSNR)
  - Structural Similarity (SSIM)

What is the performance of a model ?

- The performance of a model is assessed based on one or several **metrics**

Examples of classic metrics:

- **Regression metrics**
  - Mean Square Error (MSE)
  - Root Mean Square Error (RMSE)
  - Peak Signal-to-Noise Ratio (PSNR)
  - Structural Similarity (SSIM)
- **Classification metrics**
  - Accuracy
  - Dice / F1
  - Intersection over union (IoU)
  - Sensitivity
  - Specificity
  - Precision

Based on the **Confusion Matrix**

Confusion Matrix

		Estimated class	
		Positive	Negative
True class	Negative	FP	TN
	Positive	TP	FN

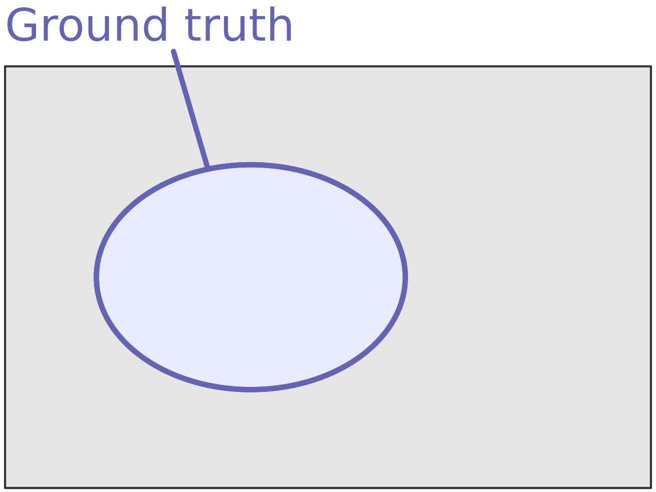
- **FP**: false positive
- **TN**: true negative
- **TP**: true positive
- **FN**: false negative

Confusion Matrix

		Estimated class	
		Positive	Negative
True class	Negative	FP	TN
	Positive	TP	FN

- **FP**: false positive
- **TN**: true negative
- **TP**: true positive
- **FN**: false negative

- Example for segmentation:



$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

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

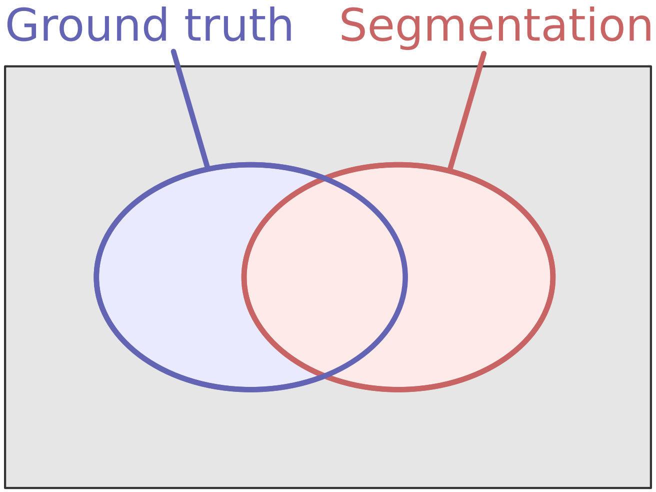
$$\text{Specificity} = \frac{TN}{TN + FP}$$

Confusion Matrix

		Estimated class	
		Positive	Negative
True class	Negative	FP	TN
	Positive	TP	FN

- **FP**: false positive
- **TN**: true negative
- **TP**: true positive
- **FN**: false negative

- Example for segmentation:



$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

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

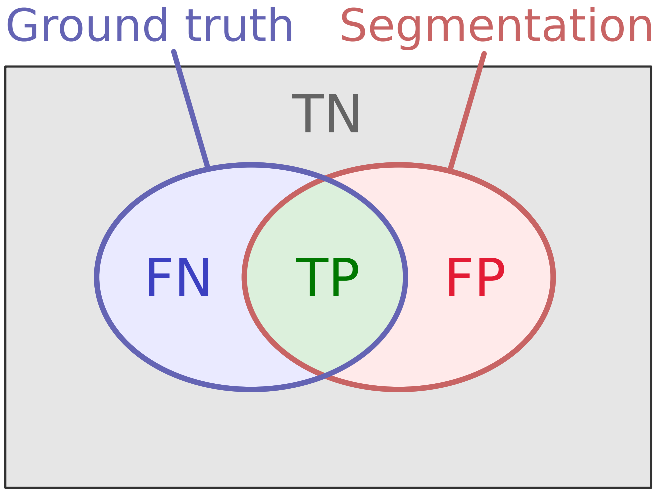
$$\text{Specificity} = \frac{TN}{TN + FP}$$

Confusion Matrix

		Estimated class	
		Positive	Negative
True class	Negative	FP	TN
	Positive	TP	FN

- **FP**: false positive
- **TN**: true negative
- **TP**: true positive
- **FN**: false negative

- Example for segmentation:



$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

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

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

Choice of metric

- The metrics should be chosen depending on the application.
  $\longrightarrow$ For the same task, the notion of performance may be very different depending on the application.

Choice of metric

- The metrics should be chosen depending on the application.
  $\longrightarrow$ For the same task, the notion of performance may be very different depending on the application.

Example: Detection of a rare disease.

- Test used to select people that should be immediately hospitalized or may die
  $\longrightarrow$ Missing a case is very bad (false negative)
  $\longrightarrow$ We want a test with a high **sensitivity**

- Test used to select people that will receive a very effective treatment. However giving the treatment to someone who is not sick is deadly.
  $\longrightarrow$ Detecting a person that is not sick is very bad (false positive)
  $\longrightarrow$ We want a test with a high **specificity**

Supervised machine learning

A. Introduction

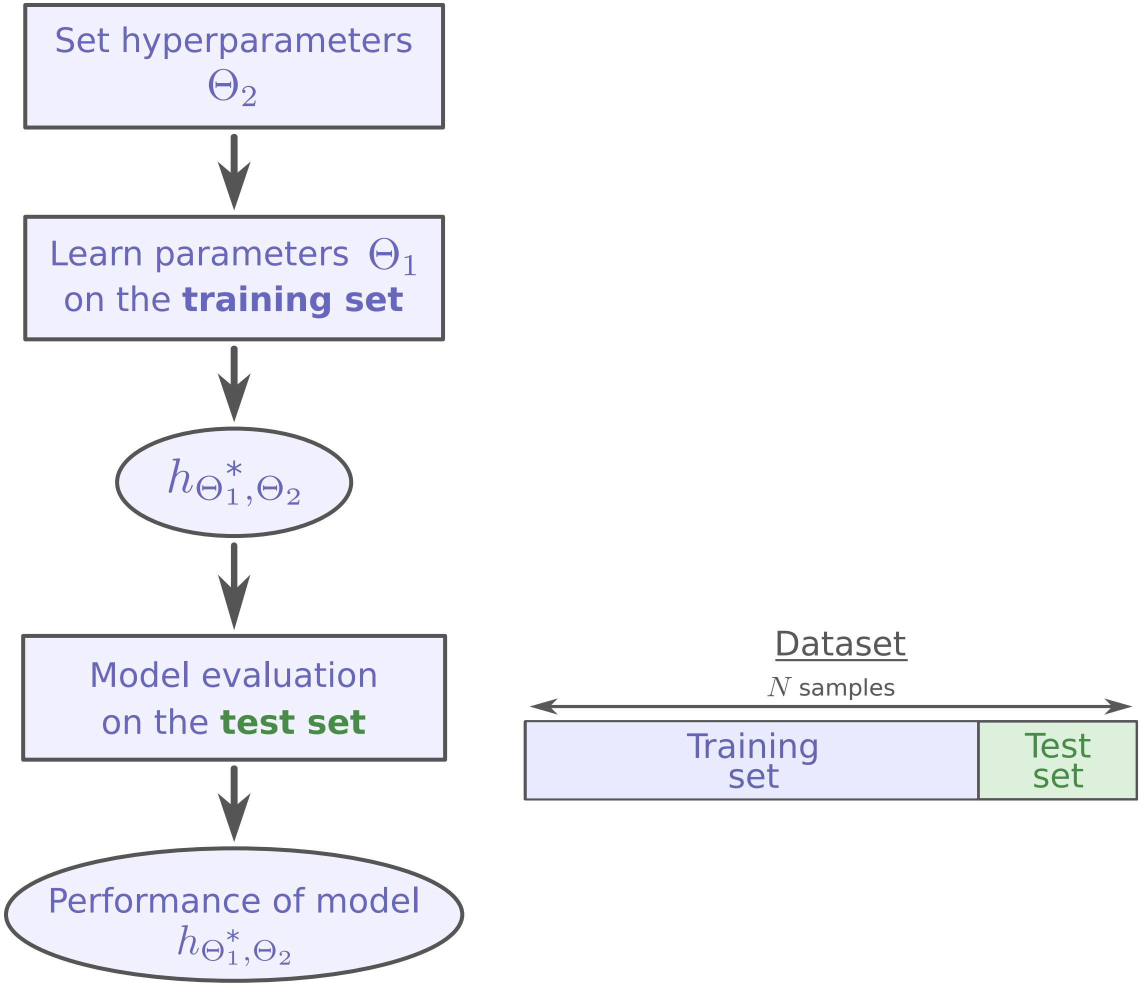
B. Choice of machine learning algorithm

C. Machine learning pipeline

1. Training
2. Evaluation
3. **Model selection**

D. Special considerations in medical applications

How to choose the model hyperparameters ?



Set hyperparameters Θ_2

Learn parameters Θ_1 on the **training set**

$h_{\Theta_1^*,\Theta_2}$

Model evaluation on the **test set**

Performance of model $h_{\Theta_1^*,\Theta_2}$

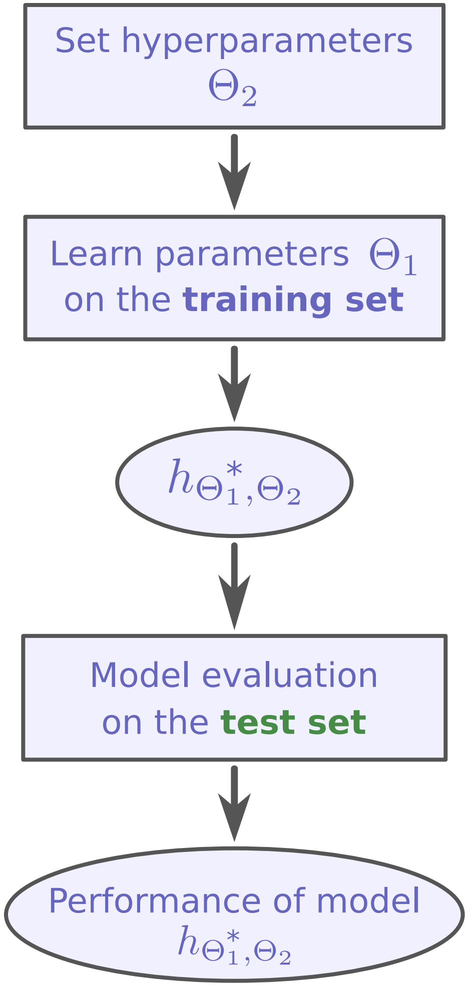
Dataset

N samples

Training set

Test set

How to choose the model hyperparameters ?



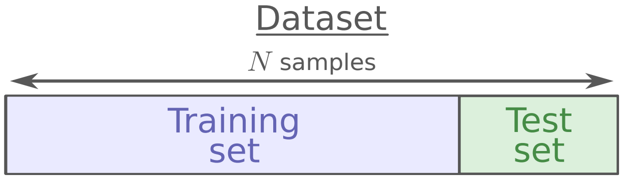
Set hyperparameters Θ_2

Learn parameters Θ_1 on the **training set**

$h_{\Theta_1^*,\Theta_2}$

Model evaluation on the **test set**

Performance of model $h_{\Theta_1^*,\Theta_2}$



Dataset

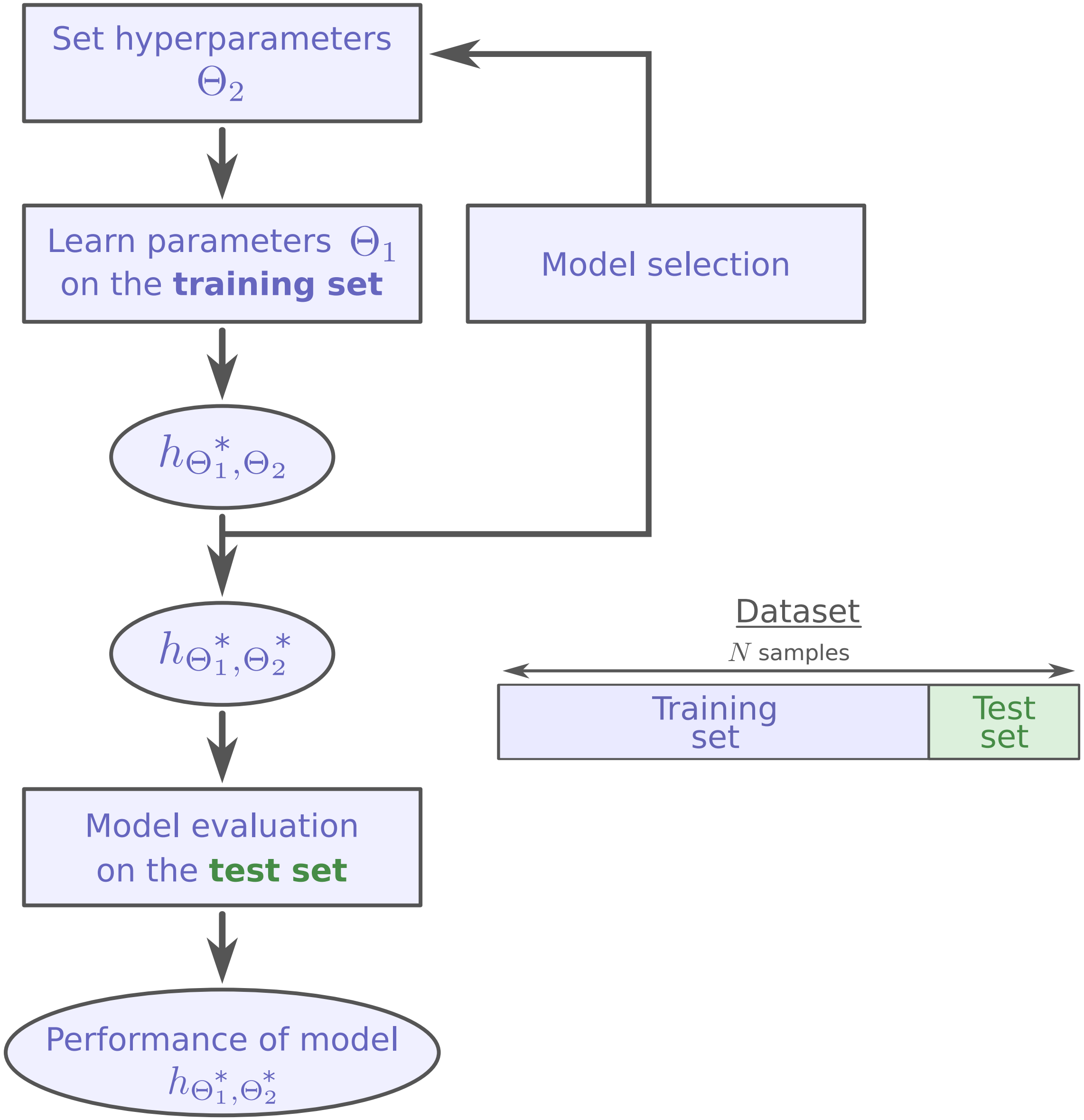
N samples

Training set

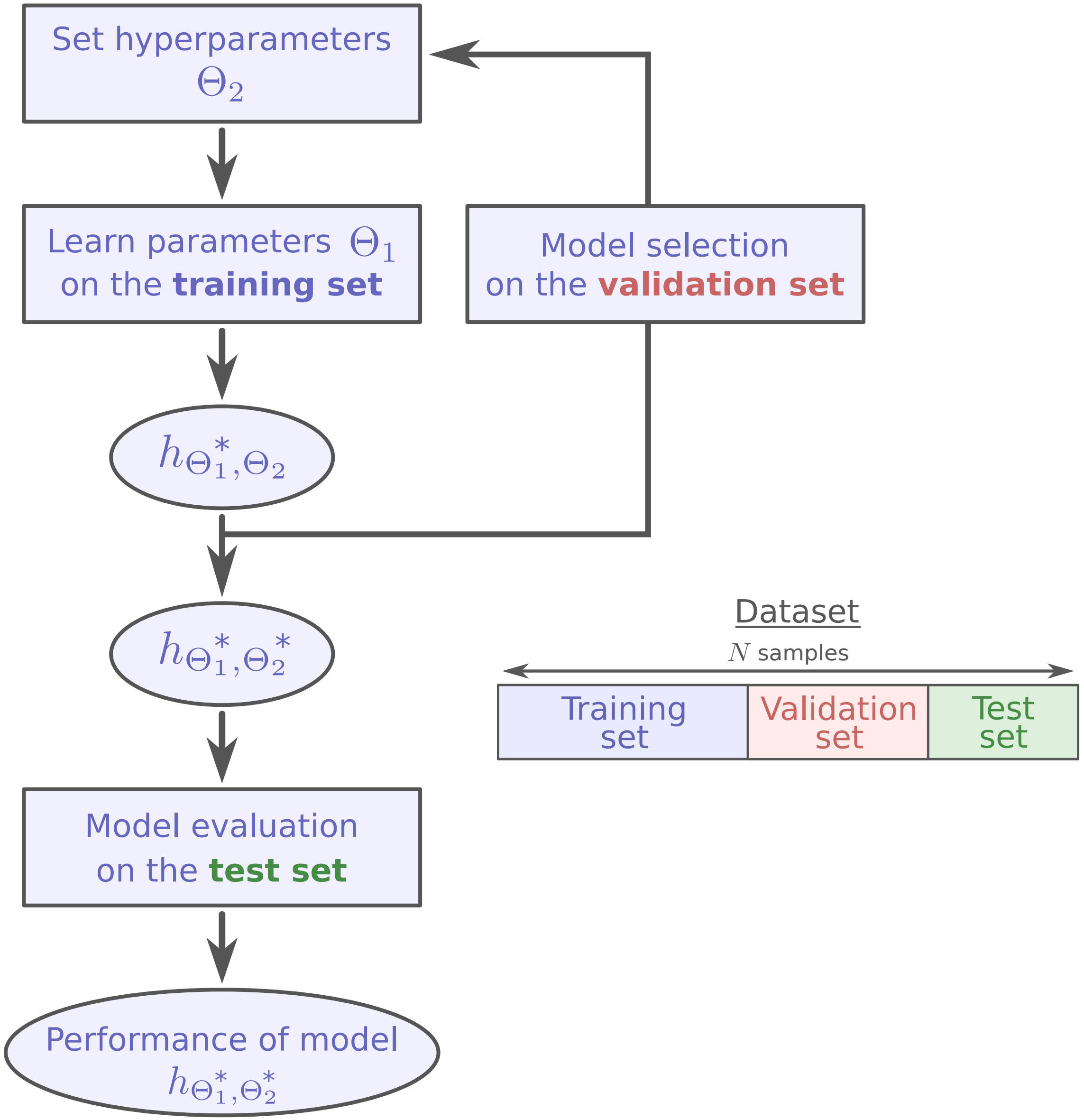
Test set

$\longrightarrow$ **Model selection**

Model selection


Set hyperparameters Θ_2
Learn parameters Θ_1 on the **training set**
Model selection
$h_{\Theta_1^*,\Theta_2}$
$h_{\Theta_1^*,\Theta_2^*}$
Model evaluation on the **test set**
Performance of model $h_{\Theta_1^*,\Theta_2^*}$
Dataset
N samples
Training set
Test set

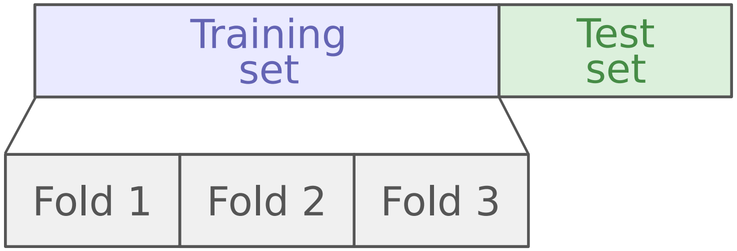
Model selection


Set hyperparameters Θ_2
Learn parameters Θ_1 on the **training set**
Model selection on the **validation set**
$h_{\Theta_1^*,\Theta_2}$
$h_{\Theta_1^*,\Theta_2^*}$
Model evaluation on the **test set**
Performance of model $h_{\Theta_1^*,\Theta_2^*}$
Dataset
N samples
Training set
Validation set
Test set

k-fold cross validation for model selection

Objective: Selection of the best set of hyperparameters Θ_2^*.

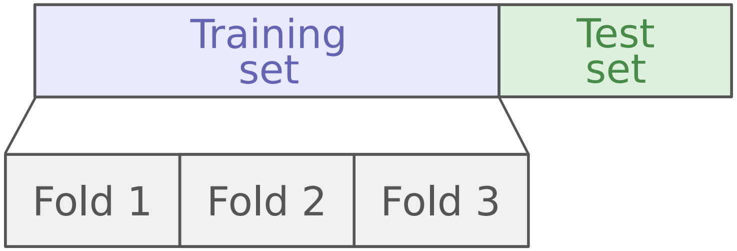
- Split the dataset in two: a training and a test set.
  Keep the test set aside and split the training set in k folds



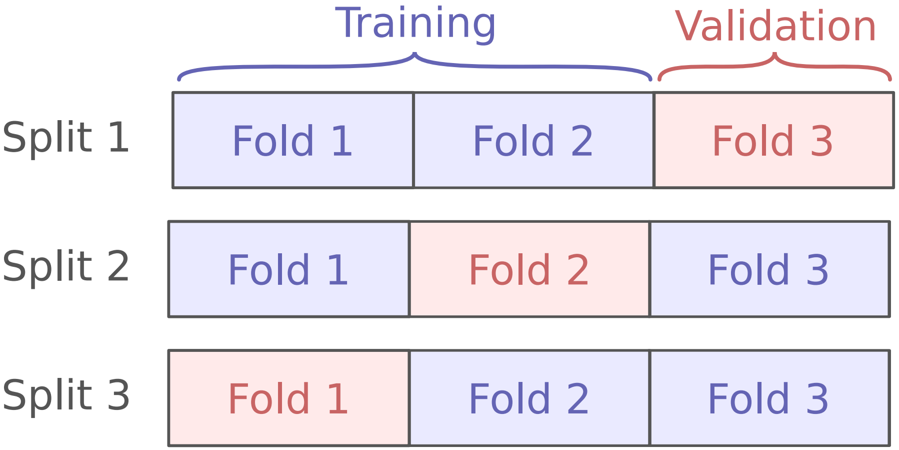
k-fold cross validation for model selection

Objective: Selection of the best set of hyperparameters Θ_2^*.

- Split the dataset in two: a training and a test set.
  Keep the test set aside and split the training set in k folds

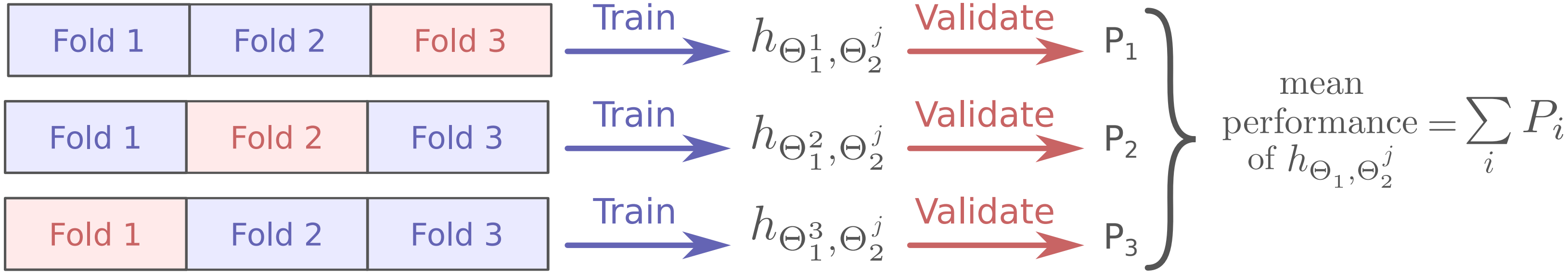


- Generate the k combinations of 1 validation fold and the remaining $k - 1$ training folds from the training set



k-fold cross validation for model selection

- For each set of hyperparameters Θ_2^j, perform a k-fold cross validation evaluation.
  Compute the mean performance of the model for fixed Θ_2^j.



Fold 1 | Fold 2 | Fold 3 Train $\xrightarrow{}$ $h_{\Theta_1^1,\Theta_2^j}$ Validate $\xrightarrow{}$ P_1

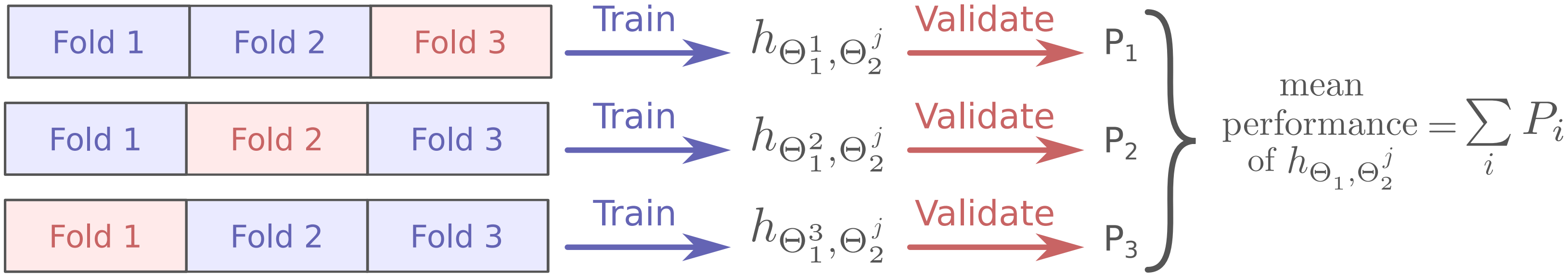
Fold 1 | Fold 2 | Fold 3 Train $\xrightarrow{}$ $h_{\Theta_1^2,\Theta_2^j}$ Validate $\xrightarrow{}$ P_2

Fold 1 | Fold 2 | Fold 3 Train $\xrightarrow{}$ $h_{\Theta_1^3,\Theta_2^j}$ Validate $\xrightarrow{}$ P_3

mean performance of $h_{\Theta_1,\Theta_2^j} = \sum_i P_i$

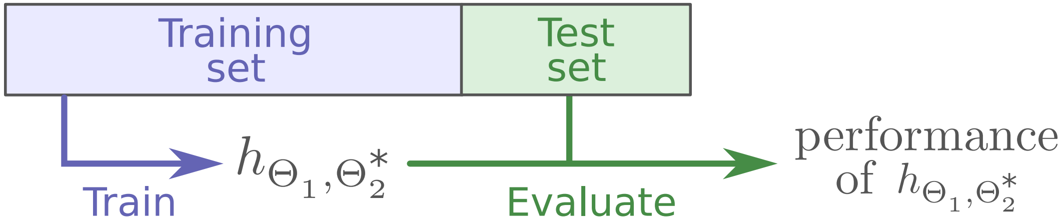
k-fold cross validation for model selection

- For each set of hyperparameters Θ_2^j, perform a k-fold cross validation evaluation.
  Compute the mean performance of the model for fixed Θ_2^j.



Fold 1
Fold 2
Fold 3
Train
$h_{\Theta_1^1,\Theta_2^j}$
Validate
P_1
Fold 1
Fold 2
Fold 3
Train
$h_{\Theta_1^2,\Theta_2^j}$
Validate
P_2
Fold 1
Fold 2
Fold 3
Train
$h_{\Theta_1^3,\Theta_2^j}$
Validate
P_3
mean performance of h_{Θ_1,Θ_2^j} $= \sum_i P_i$

- Choose the set of hyperparameters Θ_2^* providing the best mean performance and train a new model on the full training set.
- Evaluate the performance of this model on the test set.



Training set
Test set
Train
h_{Θ_1,Θ_2^*}
Evaluate
performance of h_{Θ_1,Θ_2^*}

Problems with k-fold cross validation for model selection

- The choice of the best model is done based on the average performance on training set and not on an independent dataset.

  $\longrightarrow$ Introduction of a **model selection bias**

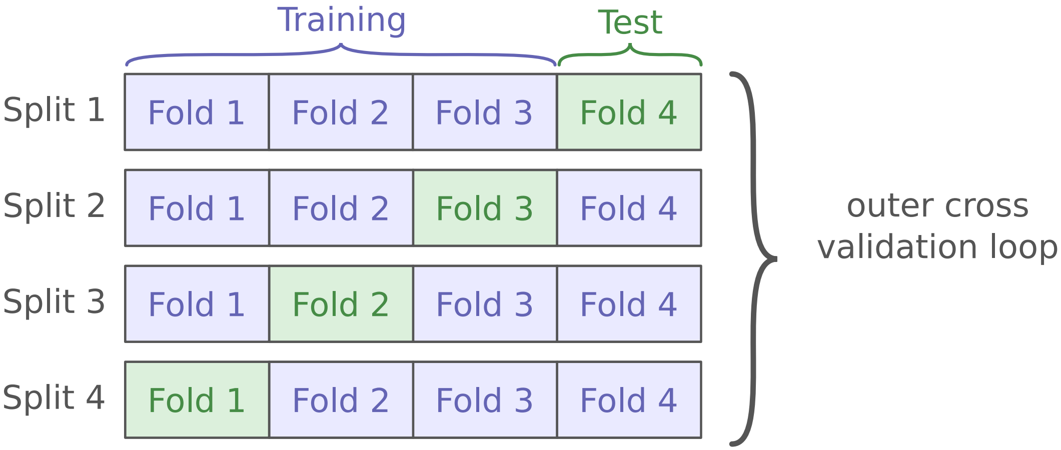
- The performance of the selected model is evaluated on a single test set

  $\longrightarrow$ No estimation of the **variance due to the test set choice**.

$\longrightarrow$ Use **nested k-fold cross validation**

Nested k-fold cross validation

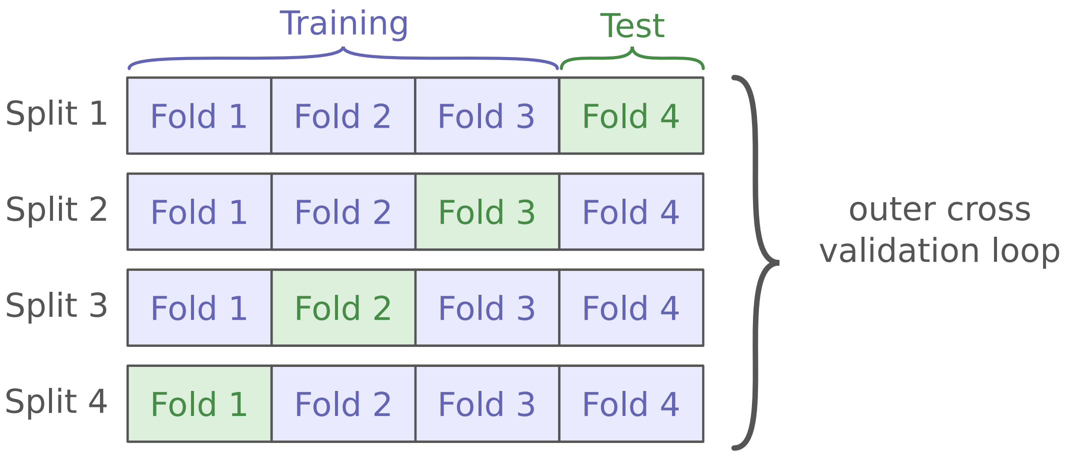
- Split the dataset in k-folds and generate the classic k combinations.



Split 1	Fold 1	Fold 2	Fold 3	Fold 4
Split 2	Fold 1	Fold 2	Fold 3	Fold 4
Split 3	Fold 1	Fold 2	Fold 3	Fold 4
Split 4	Fold 1	Fold 2	Fold 3	Fold 4

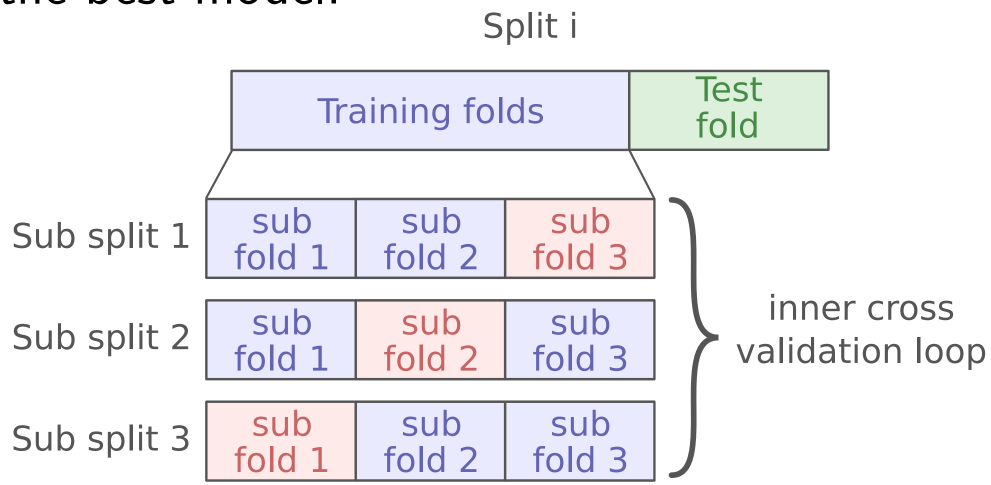
Nested k-fold cross validation

- Split the dataset in k-folds and generate the classic k combinations.



Split 1	Fold 1	Fold 2	Fold 3	Fold 4
Split 2	Fold 1	Fold 2	Fold 3	Fold 4
Split 3	Fold 1	Fold 2	Fold 3	Fold 4
Split 4	Fold 1	Fold 2	Fold 3	Fold 4

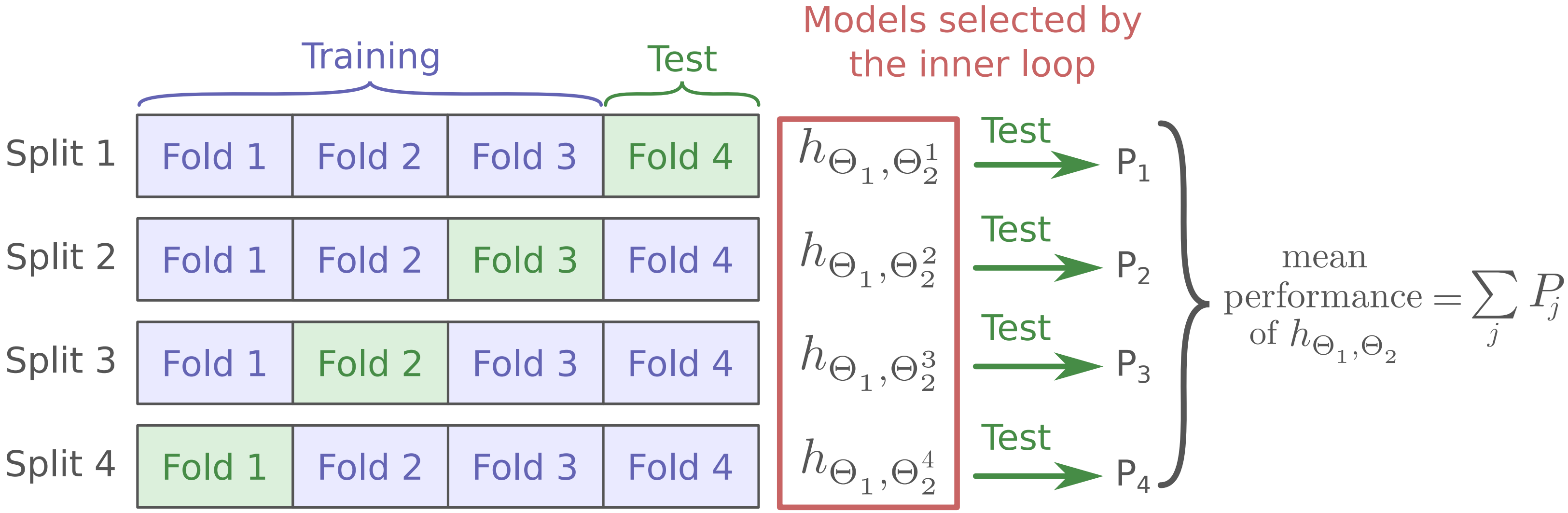
- For each split, perform a k-fold cross validation on the training folds to select the best model.



Sub split 1	sub fold 1	sub fold 2	sub fold 3
Sub split 2	sub fold 1	sub fold 2	sub fold 3
Sub split 3	sub fold 1	sub fold 2	sub fold 3

Nested k-fold cross validation

- For each split *j*, test the best inner loop model on the test fold
- Compute the mean and standard deviation of the performance of the models
  ⟶ Provides an estimate of the generalization and stability of the learnt models



Training

Test

Models selected by the inner loop

Split 1	Fold 1	Fold 2	Fold 3	Fold 4	h_{Θ_1,Θ_2^1}	Test	P_1
Split 2	Fold 1	Fold 2	Fold 3	Fold 4	h_{Θ_1,Θ_2^2}	Test	P_2
Split 3	Fold 1	Fold 2	Fold 3	Fold 4	h_{Θ_1,Θ_2^3}	Test	P_3
Split 4	Fold 1	Fold 2	Fold 3	Fold 4	h_{Θ_1,Θ_2^4}	Test	P_4

$$\text{mean performance of } h_{\Theta_1,\Theta_2} = \sum_j P_j$$

Nested k-fold cross validation

Remarks:

- The inner loop does the model selection and the outer loop does the evaluation of the selected model

- The model selection is included in the learning where the hyperparameters are learnt from the data.

- Two common strategies to obtain the final model:
    - Run the inner loop one more time on the complete dataset and choose the hyperparameters yielding the best mean performance
    - Use the k models selected by the inner loops to do **ensembling**.

How to learn a “good” model

- Keep in mind the bias/variance tradeoff when learning a model.
  - How to reduce the bias (avoid underfitting)
    - Increase the complexity of your model
    - Add more features
  - How to reduce the variance (avoid overfitting)
    - Use a validation strategy
    - Reduce the complexity of the model
    - Add more training data
    - Reduce the number of features (dimensionality reduction)
    - Use regularization
    - Perform ensembling

How to learn a “good” model

- Keep in mind the bias/variance tradeoff when learning a model.
  - How to reduce the bias (avoid underfitting)
    - Increase the complexity of your model
    - Add more features
  - How to reduce the variance (avoid overfitting)
    - Use a validation strategy
    - Reduce the complexity of the model
    - Add more training data
    - Reduce the number of features (dimensionality reduction)
    - Use regularization
    - Perform ensembling
- Carefully chose your metrics and evaluation strategy

Supervised machine learning

A. Introduction

B. Choice of machine learning algorithm

C. Machine learning pipeline

1. Training
2. Evaluation
3. Model selection

D. Special considerations in medical applications

Imbalanced classification

- In medical applications the datasets are often imbalanced (number of healthy cases $\gg$ number of pathological cases)

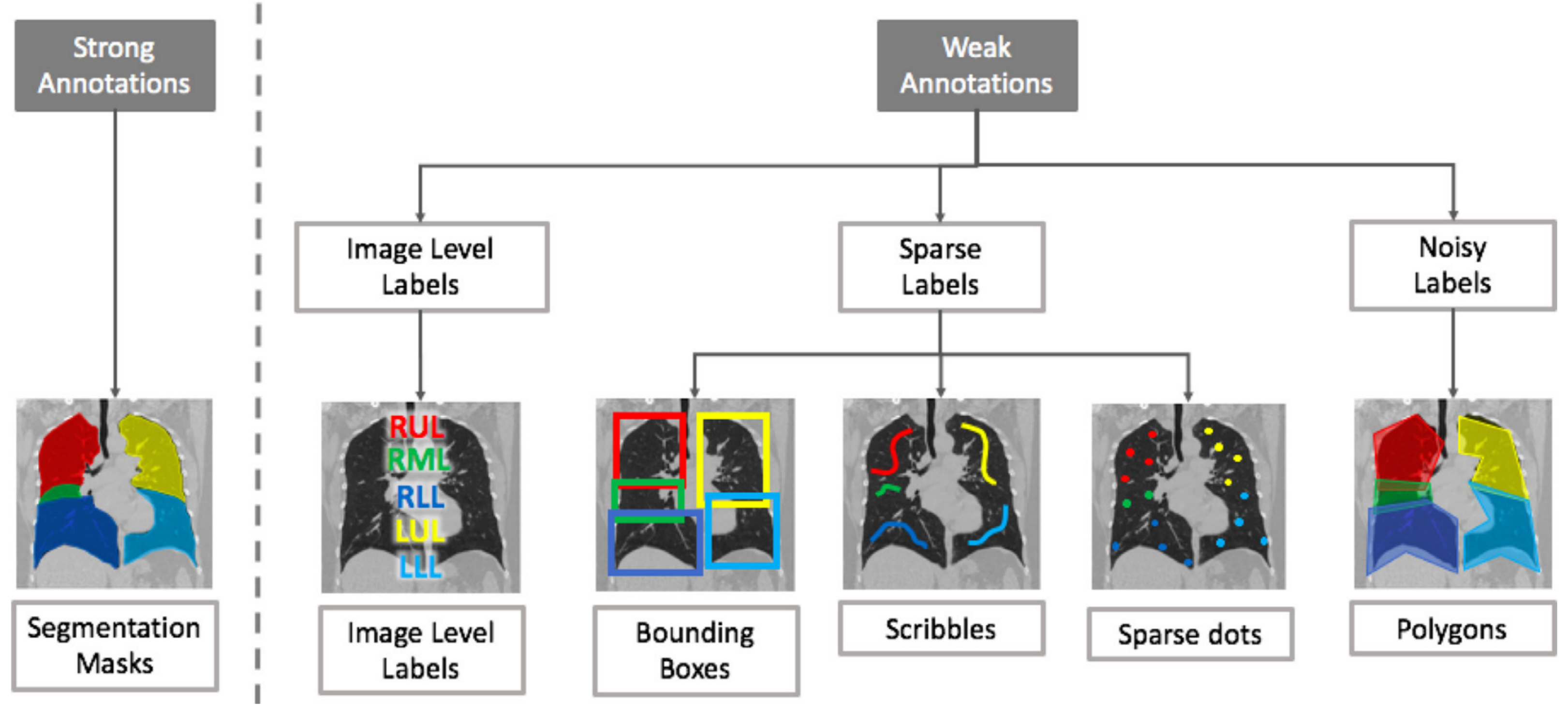
Imbalanced classification

- In medical applications the datasets are often imbalanced (number of healthy cases $\gg$ number of pathological cases)

- Specific strategies should be used:
  - **Resampling methods**
    oversampling of the rare class, downsampling of the majority class, data augmentation...
  - **Cost-sensitive training**
    Add weight in the loss to penalize misclassifications of the rare class more.
  - **Adapt the metrics**
    Dice or MCC over Accuracy, Precision/Recall over Sensitivity/Specificity...

Annotation scarcity and weak supervision

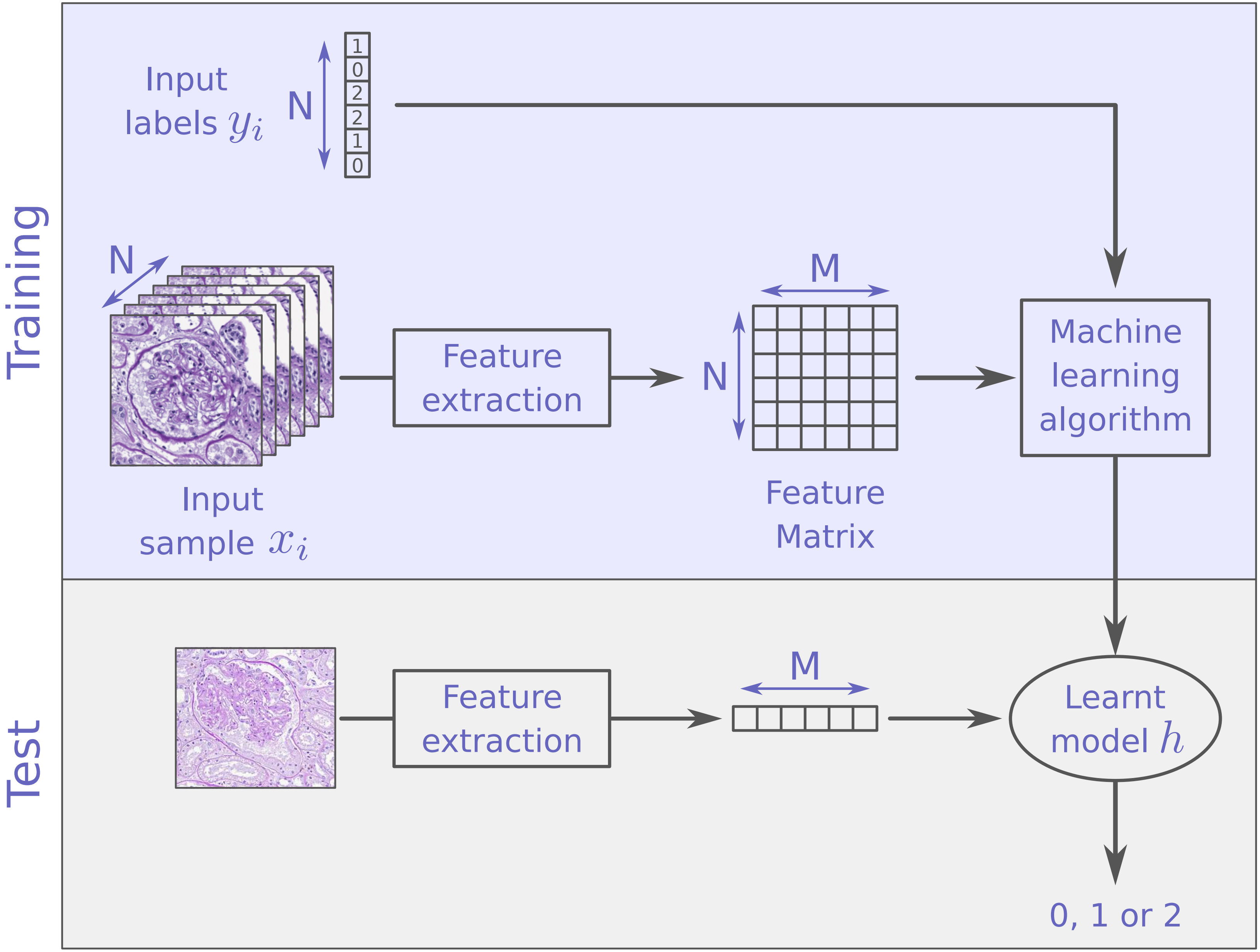
- Annotations are very expensive in medical applications.
  ⟶ Weak annotations, semi-supervision



Tajbakhsh *et al.*, "Embracing imperfect datasets: A review of deep learning solutions for medical image segmentation", MedIA, 2020

Karimi *et al.*, "Deep learning with noisy labels: Exploring techniques and remedies in medical image analysis", MedIA, 2020

Classical supervised machine learning pipeline



Training

Input labels y_i

Input sample x_i

Feature extraction

Feature Matrix

Machine learning algorithm

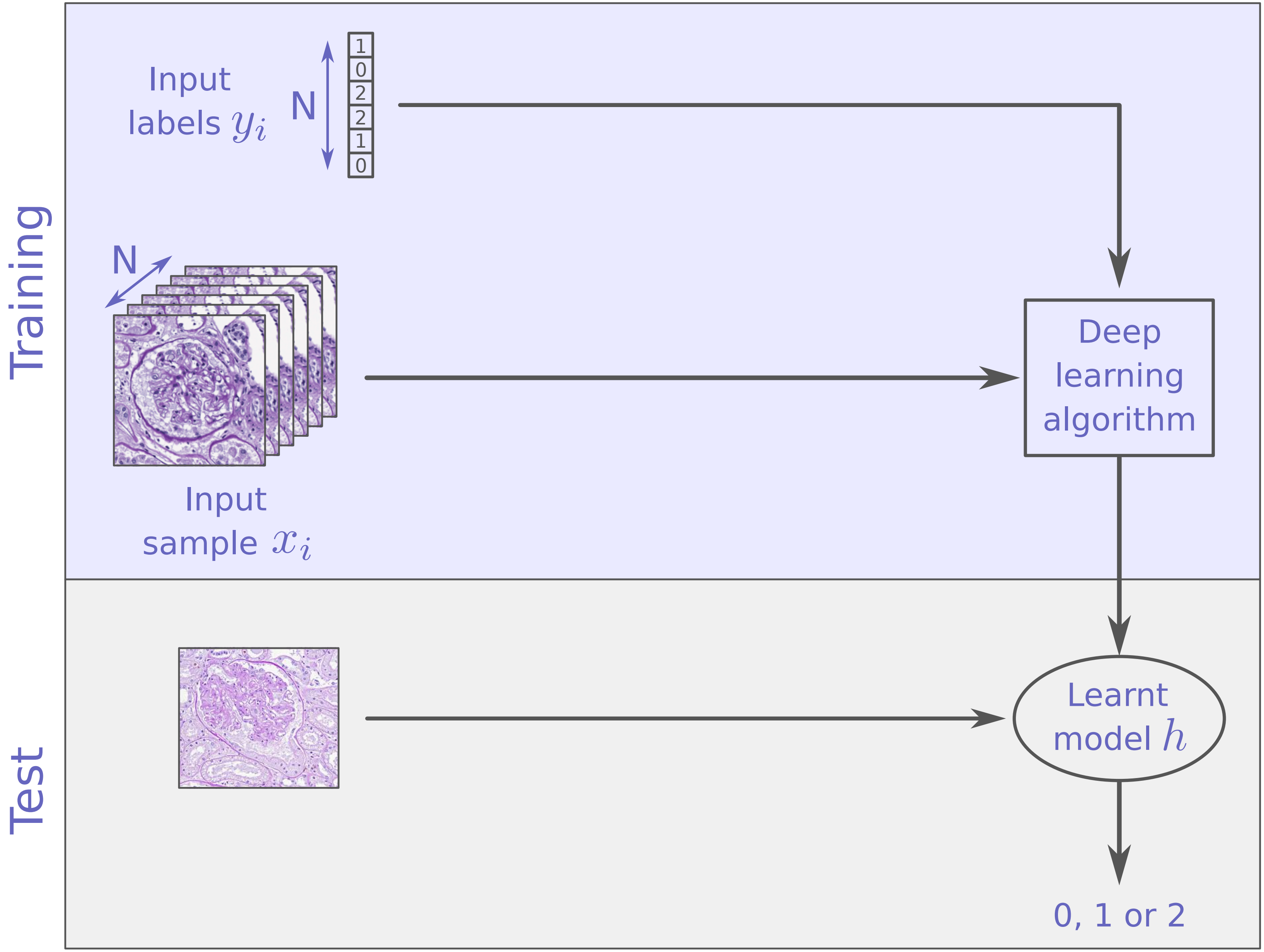
Test

Feature extraction

Learnt model h

0, 1 or 2

Supervised deep learning pipeline



Training

Input labels y_i

N

1
0
2
2
1
0

N

Input sample x_i

Deep learning algorithm

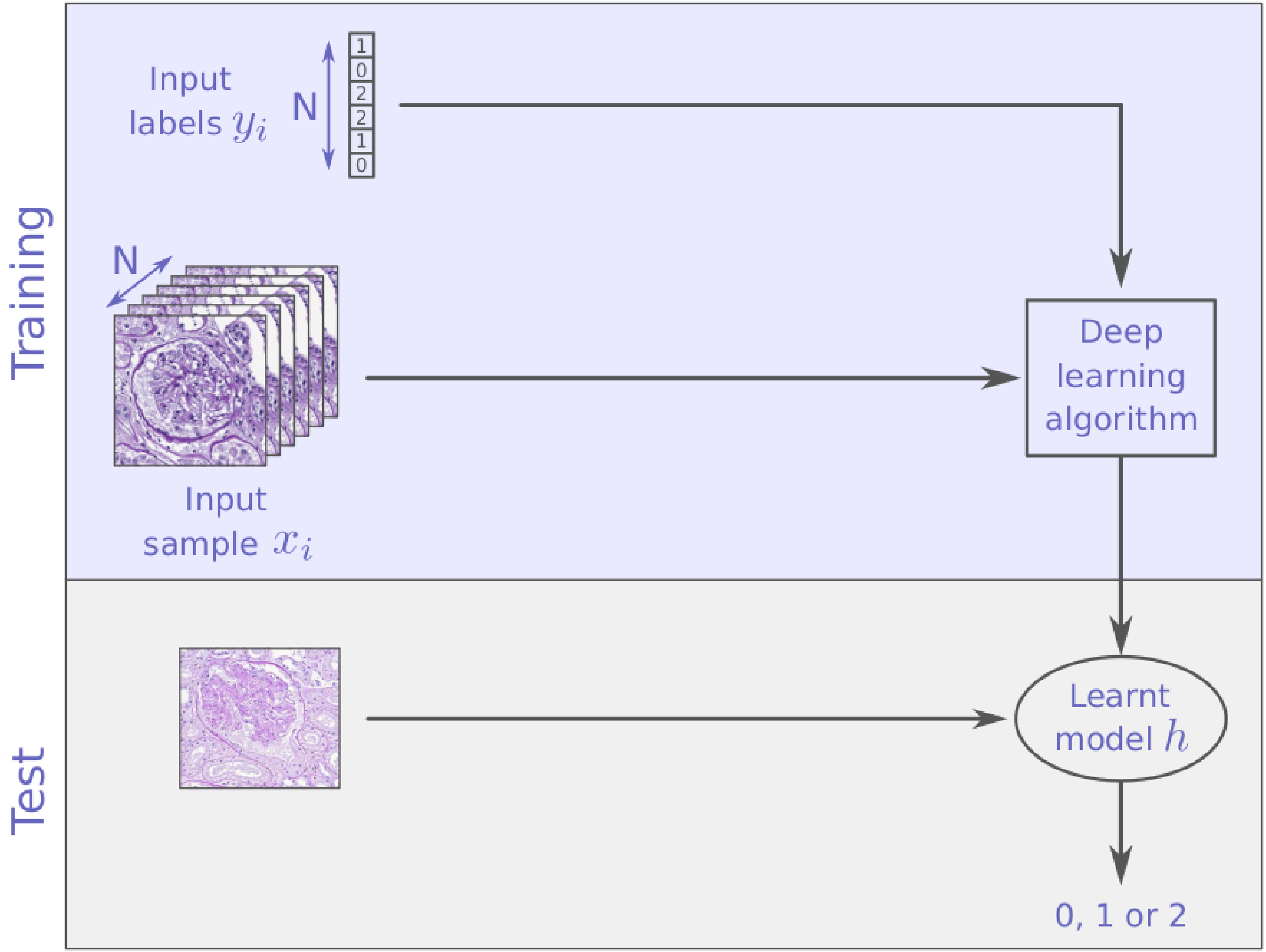
Test

Learnt model h

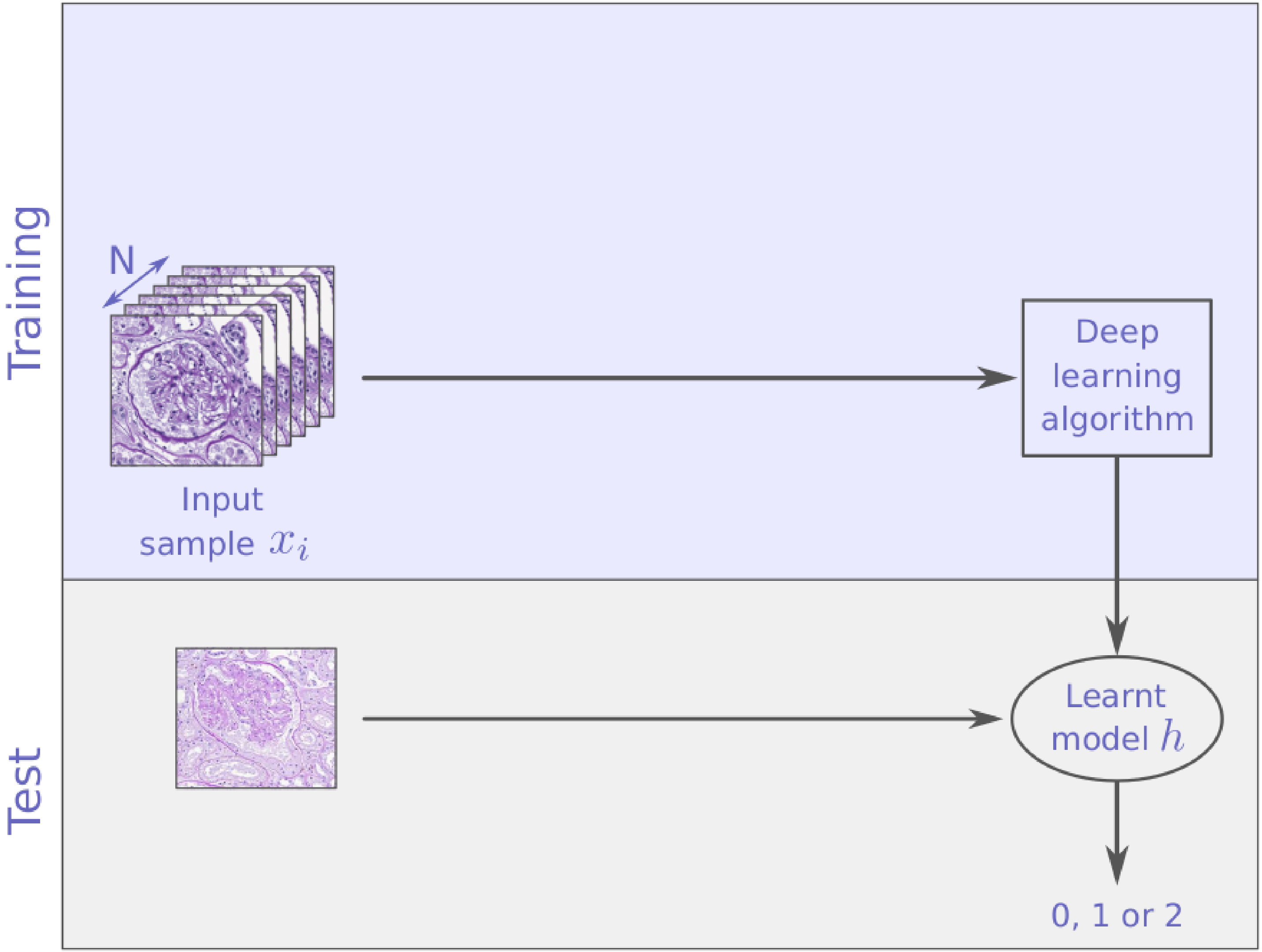
0, 1 or 2

Unsupervised Learning

from supervised learning ...



to unsupervised learning



unsupervised learning

supervised learning

$$\mathcal{D} = \{x_i, y_i\}_{1 \leq i \leq N}$$

unsupervised learning

CREATIS

unsupervised learning

$$\mathcal{D} = \{x_i\}_{1 \leq i \leq N}$$

unsupervised learning

unsupervised learning

$$\mathcal{D} = \{x_i\}_{1 \leq i \leq N}$$

unsupervised learning

with unsupervised learning you can find

... efficient representations (embedding, interpret)

... estimations of your data distribution (generate new samples)

... groups of similar samples (free labels, fill blanks)

... outliers (anomaly detection, de-noising)

unsupervised learning

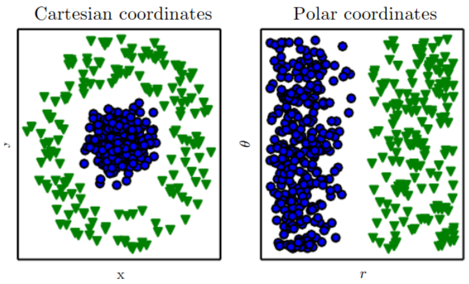
with unsupervised learning you can find

... efficient representations (embedding, interpret)

... estimations of your data distribution (generate new samples)

... groups of similar samples (free labels, fill blanks)

... outliers (anomaly detection, de-noising)



unsupervised learning

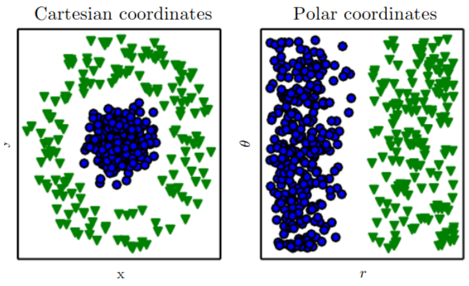
with unsupervised learning you can find

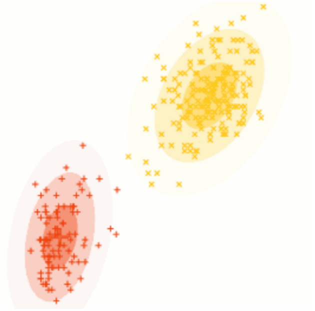
... efficient representations (embedding, interpret)

... estimations of your data distribution (generate new samples)

... groups of similar samples (free labels, fill blanks)

... outliers (anomaly detection, de-noising)





unsupervised learning

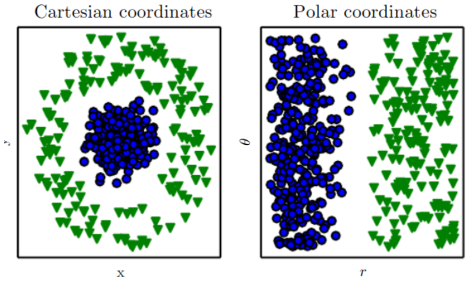
with unsupervised learning you can find

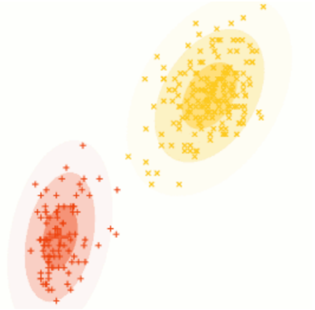
... efficient representations (embedding, interpret)

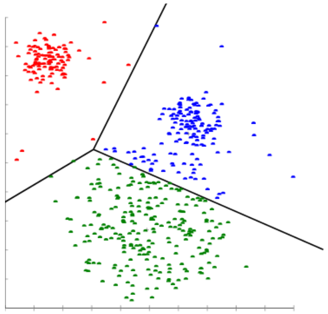
... estimations of your data distribution (generate new samples)

... groups of similar samples (free labels, fill blanks)

... outliers (anomaly detection, de-noising)







unsupervised learning

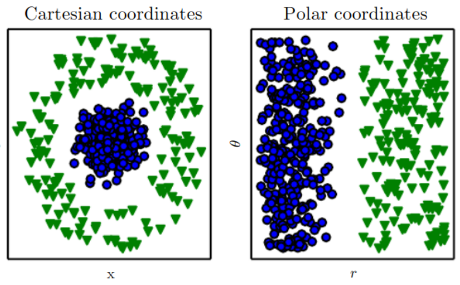
with unsupervised learning you can find

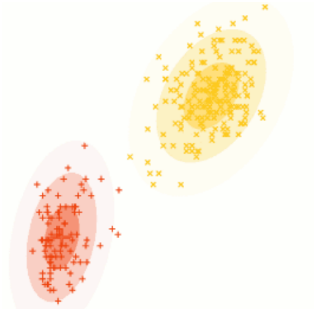
... efficient representations (embedding, interpret)

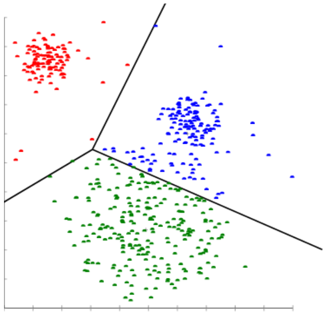
... estimations of your data distribution (generate new samples)

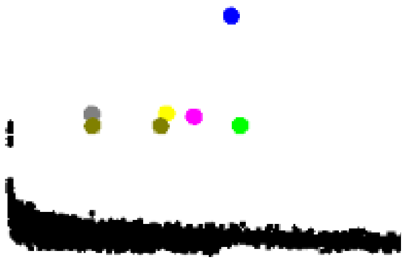
... groups of similar samples (free labels, fill blanks)

... outliers (anomaly detection, de-noising)









unsupervised learning

How to ?

Dimension reduction

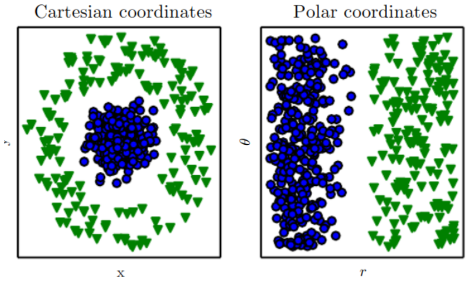
Clustering

unsupervised learning

How to ?

Dimension reduction

Clustering

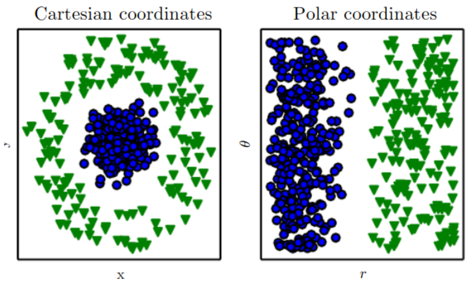


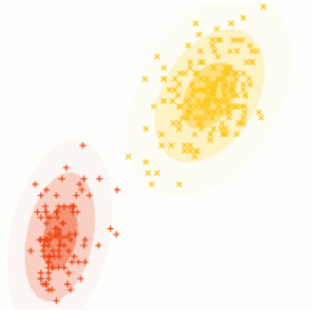
unsupervised learning

How to ?

Dimension reduction

Clustering



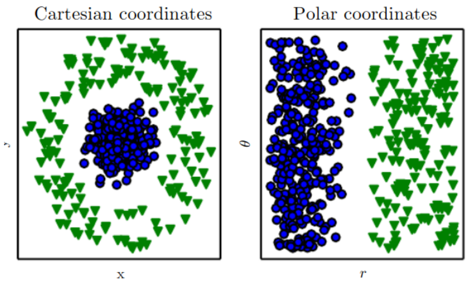


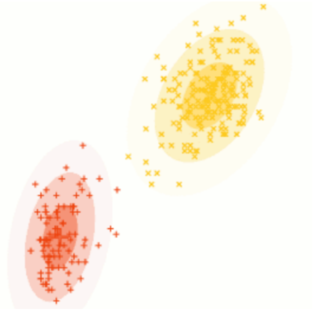
unsupervised learning

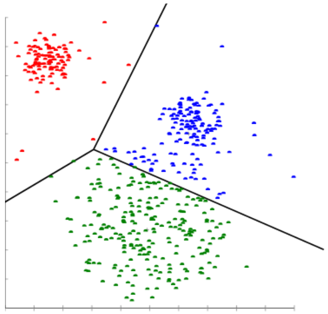
How to ?

Dimension reduction

Clustering





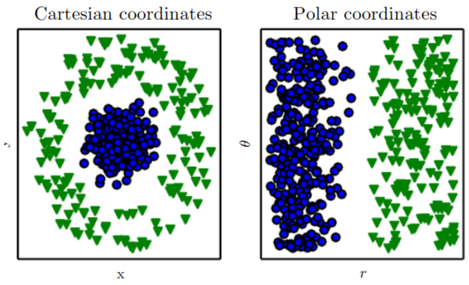


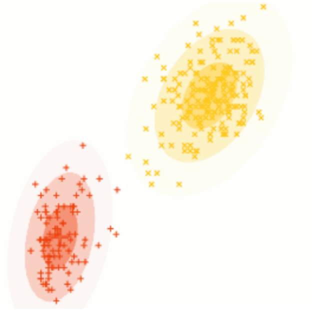
unsupervised learning

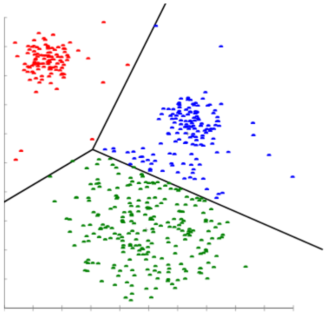
How to ?

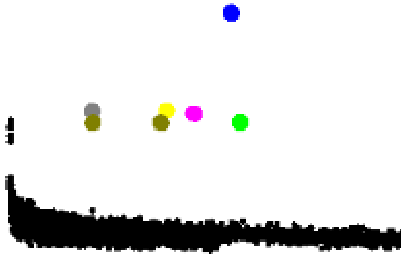
Dimension reduction

Clustering



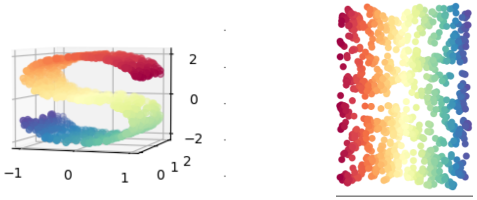






Dimension reduction

"the curse of dimensionality"



To avoid redundancy and unnecessary computational load

To visualize the data

To improve data representation

(supervised task pre-processing: semi-supervised learning)

Dimension reduction

Feature selection

Feature extraction

unsupervised learning

CREATIS

Feature selection

$$\mathcal{D} = \{x_i\}_{1 \leq i \leq N}$$

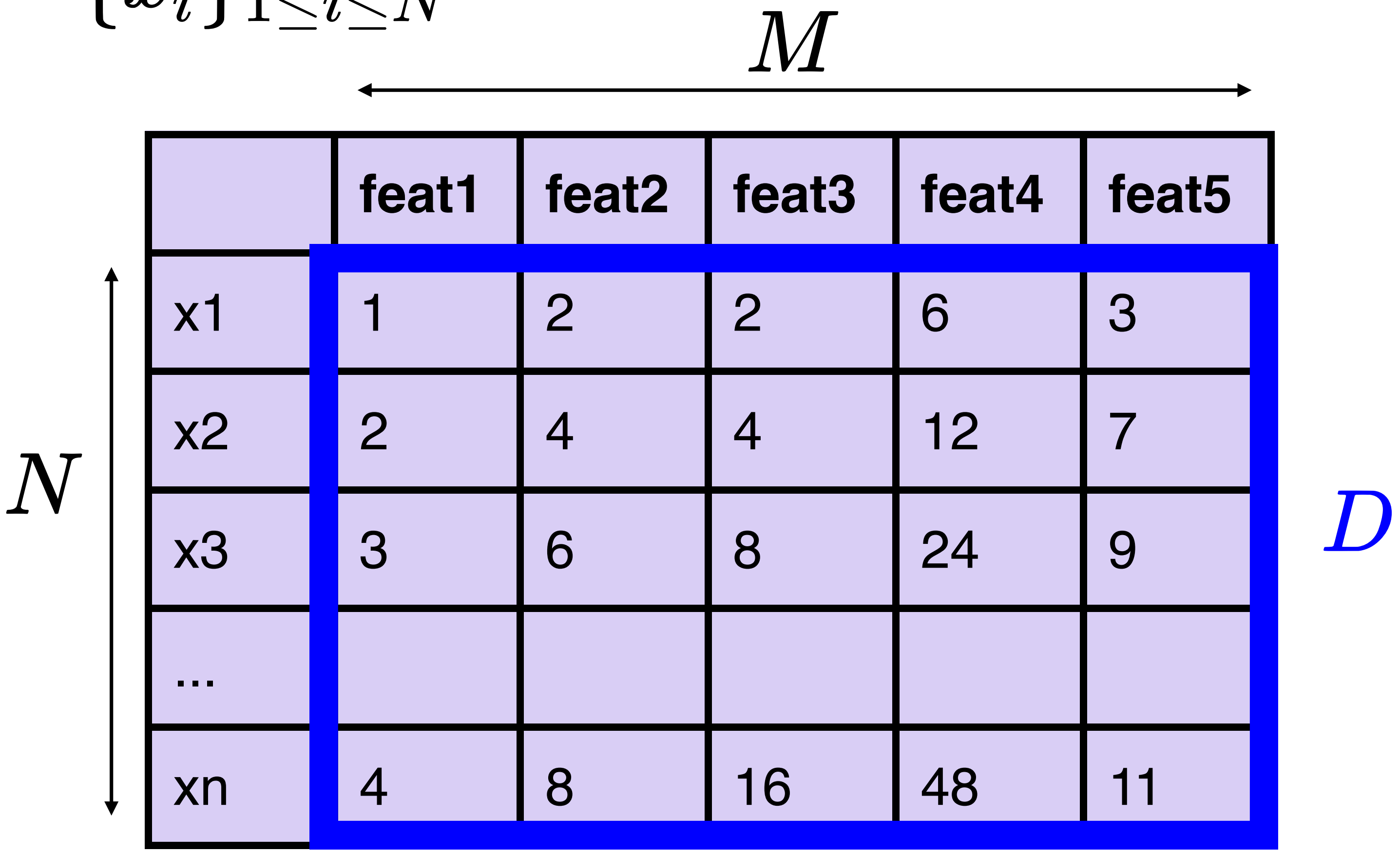
	feat1	feat2	feat3	feat4	feat5
x1	1	2	2	6	3
x2	2	4	4	12	7
x3	3	6	8	24	9
...					
xn	4	8	16	48	11

unsupervised learning

CREATIS

Feature selection

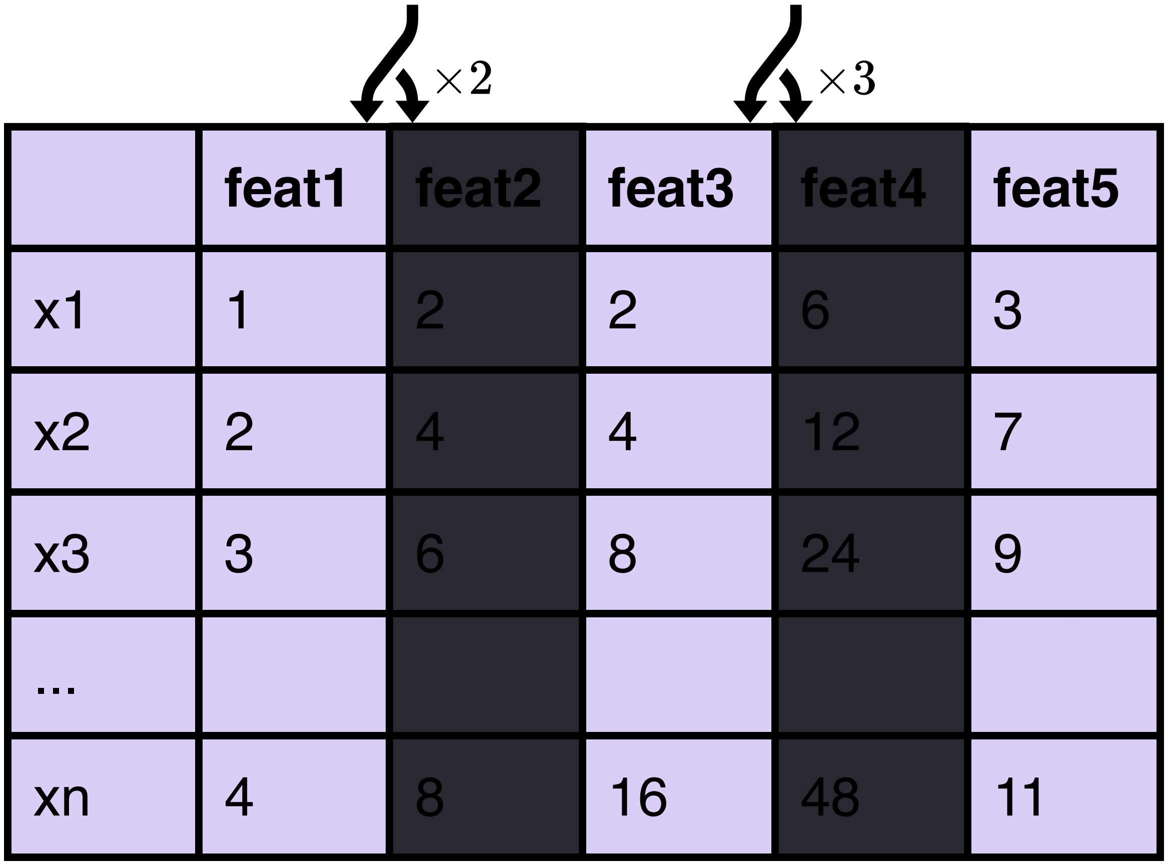
$$\mathcal{D} = \{x_i\}_{1 \leq i \leq N}$$



	feat1	feat2	feat3	feat4	feat5
x1	1	2	2	6	3
x2	2	4	4	12	7
x3	3	6	8	24	9
...					
xn	4	8	16	48	11

unsupervised learning

Feature selection



	feat1	feat2	feat3	feat4	feat5
x1	1	2	2	6	3
x2	2	4	4	12	7
x3	3	6	8	24	9
...					
xn	4	8	16	48	11

Feature selection

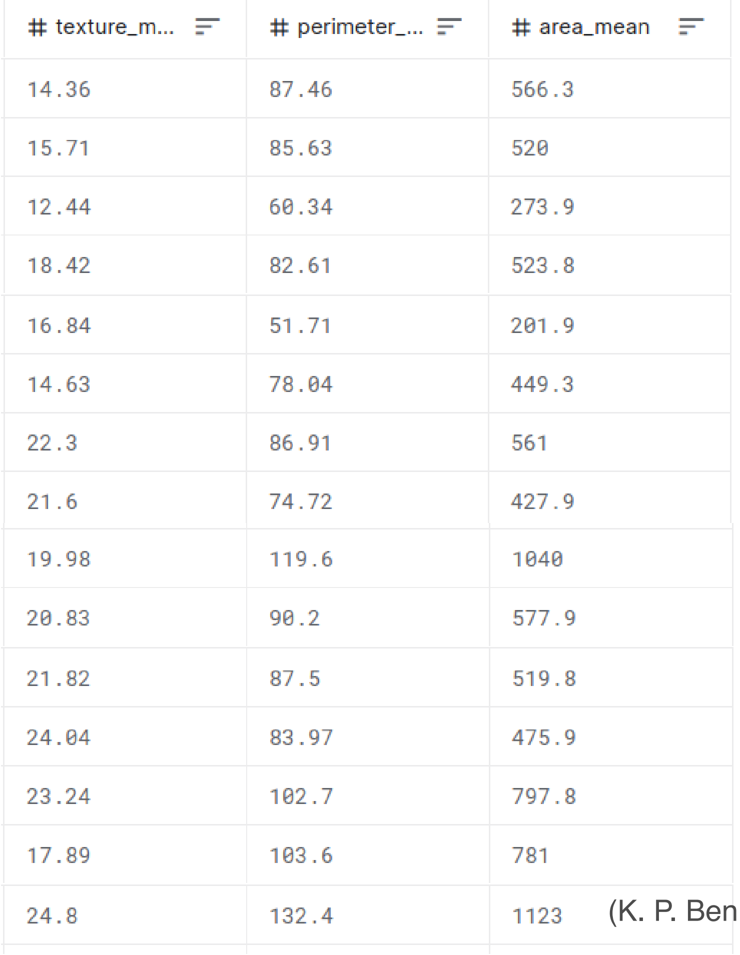
	feat1	feat3	feat5
x1	1	2	3
x2	2	4	7
x3	3	8	9
...			
xn	4	16	11

unsupervised learning

Feature selection example with the Breast Cancer dataset

$M = 10$ (only 3 here)

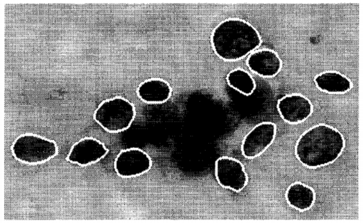
$N = 569$



# texture_m...	# perimeter_...	# area_mean
14.36	87.46	566.3
15.71	85.63	520
12.44	60.34	273.9
18.42	82.61	523.8
16.84	51.71	201.9
14.63	78.04	449.3
22.3	86.91	561
21.6	74.72	427.9
19.98	119.6	1040
20.83	90.2	577.9
21.82	87.5	519.8
24.04	83.97	475.9
23.24	102.7	797.8
17.89	103.6	781
24.8	132.4	1123

D

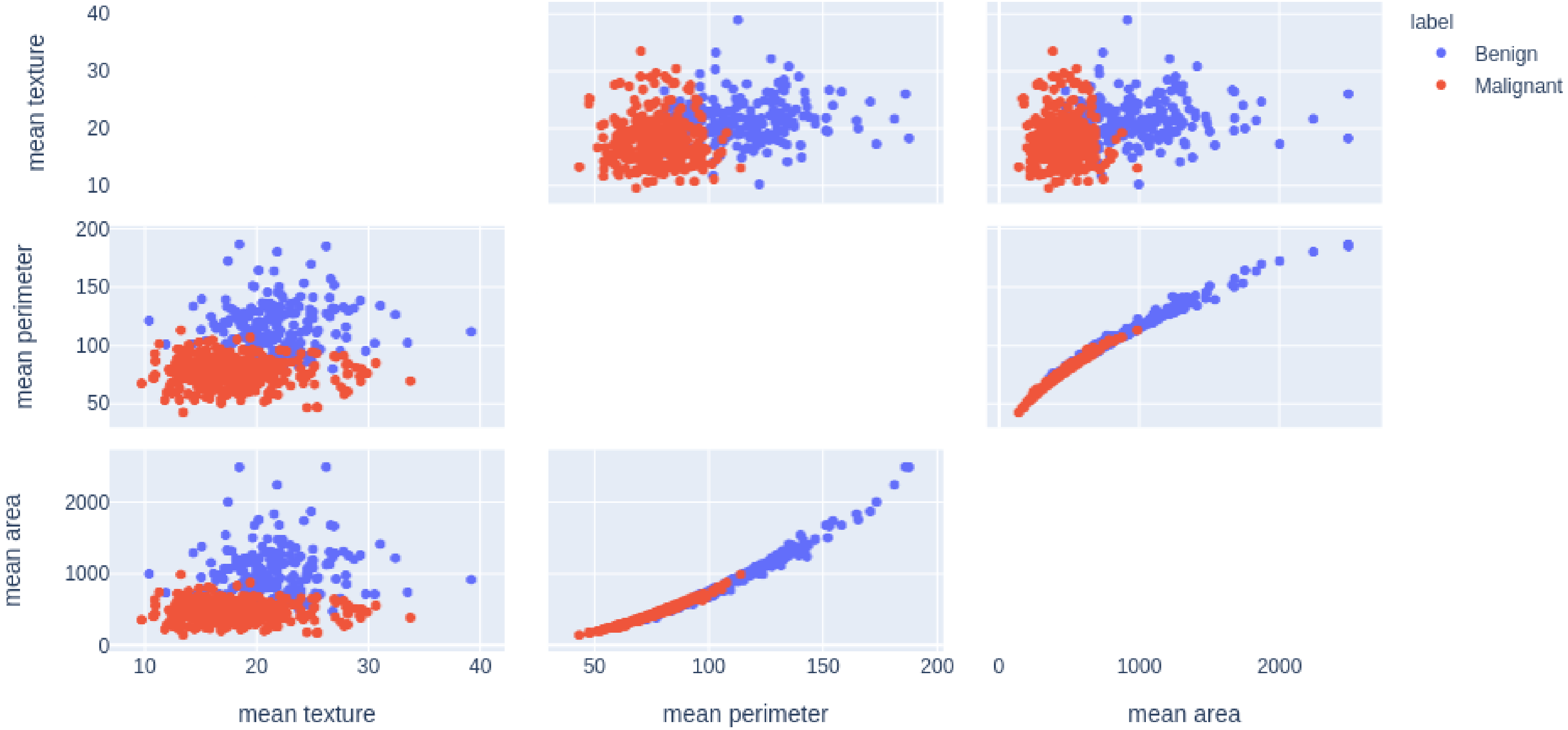
malignant breast fine needle aspirates



(K. P. Bennett and O. L. Mangasarian, 1994)

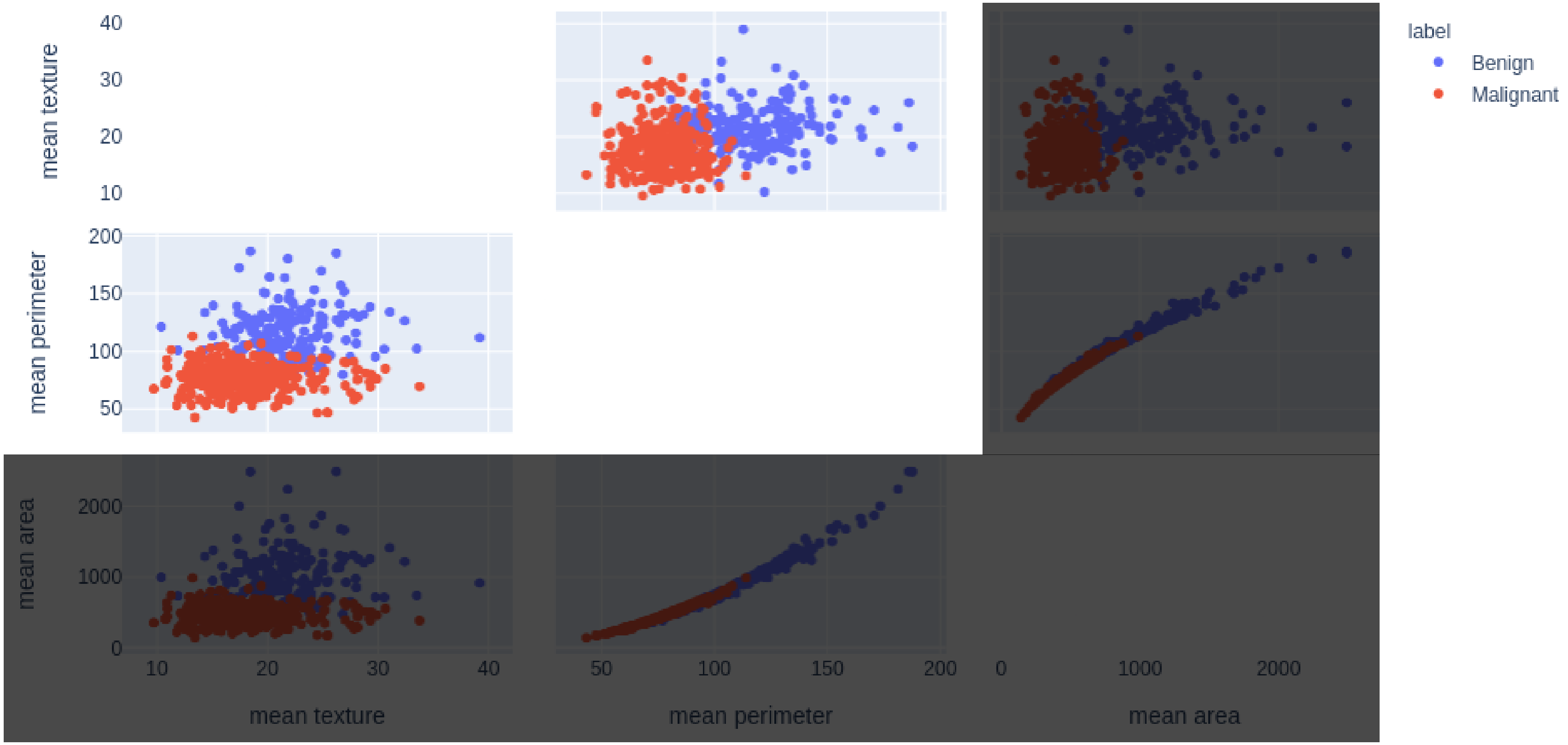
unsupervised learning

Feature selection example with the Breast Cancer dataset



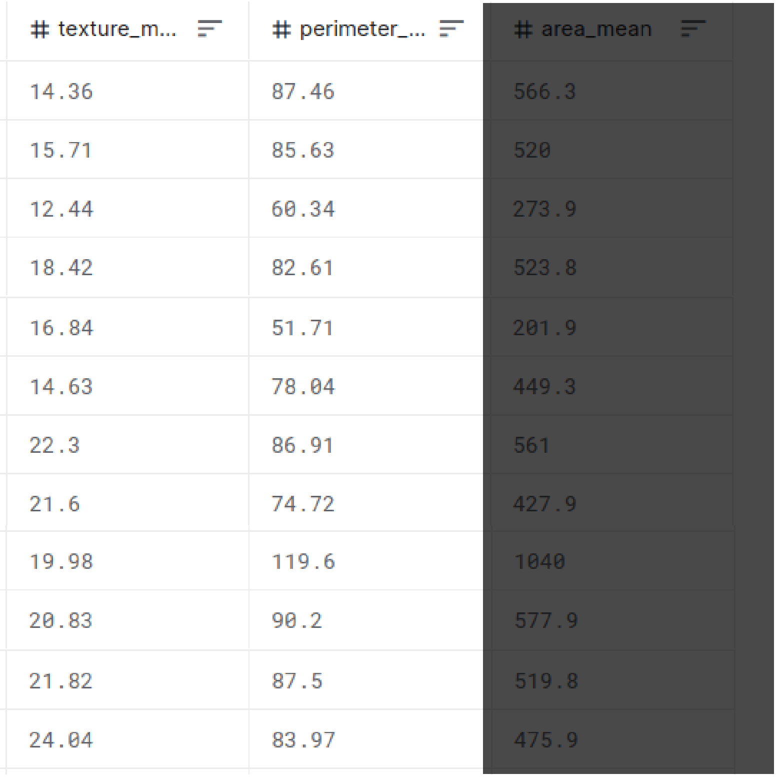
unsupervised learning

Feature selection example with the Breast Cancer dataset



Feature extraction

e.g. Principal Component Analysis (PCA)



# texture_m...	# perimeter_...	# area_mean
14.36	87.46	566.3
15.71	85.63	520
12.44	60.34	273.9
18.42	82.61	523.8
16.84	51.71	201.9
14.63	78.04	449.3
22.3	86.91	561
21.6	74.72	427.9
19.98	119.6	1040
20.83	90.2	577.9
21.82	87.5	519.8
24.04	83.97	475.9

unsupervised learning

Feature extraction
e.g. Principal Component Analysis (PCA)

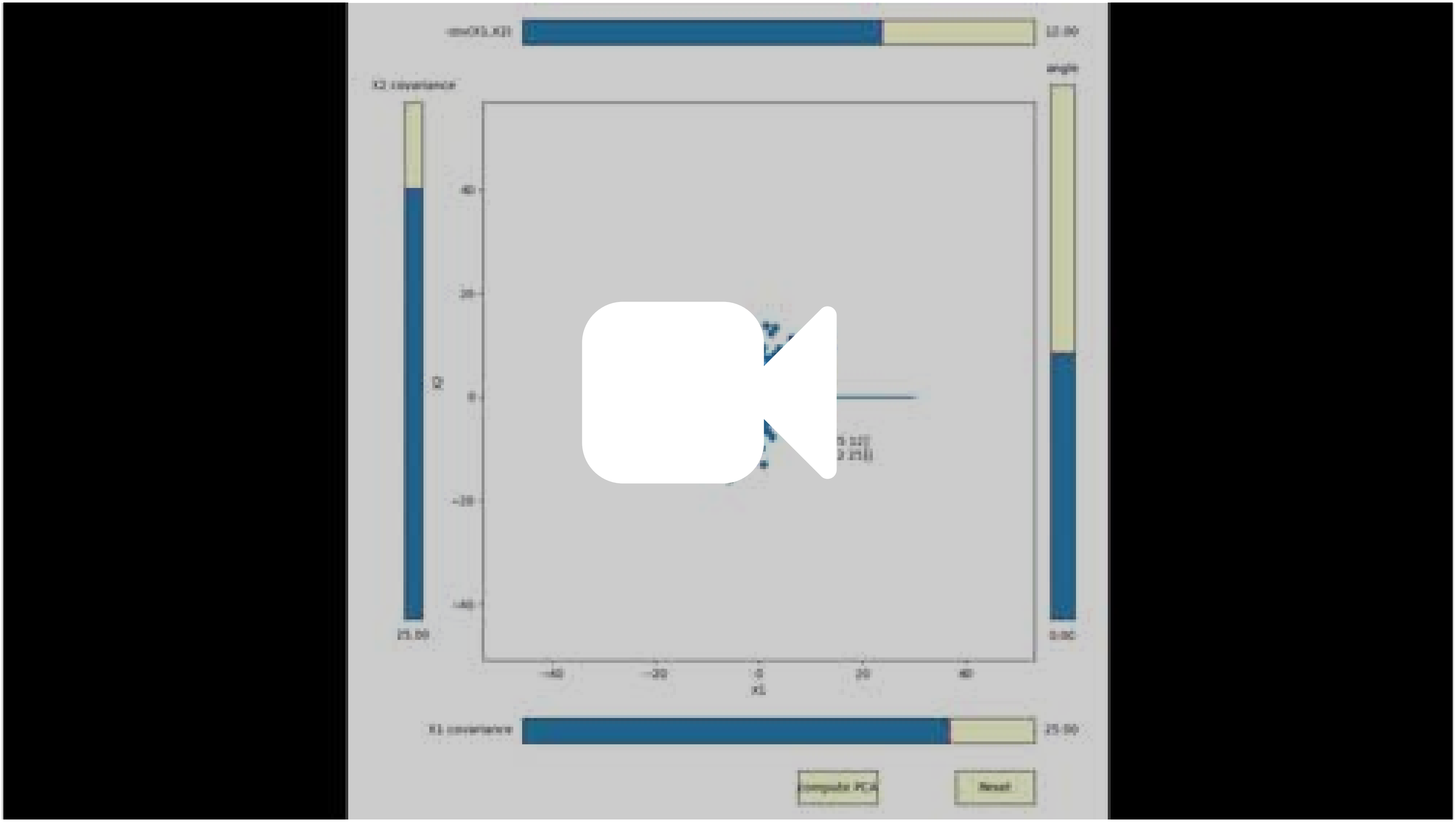
linearly combine features to find mutually orthogonal components

the (principal) components are ranked from
the most "significant" to least "significant"

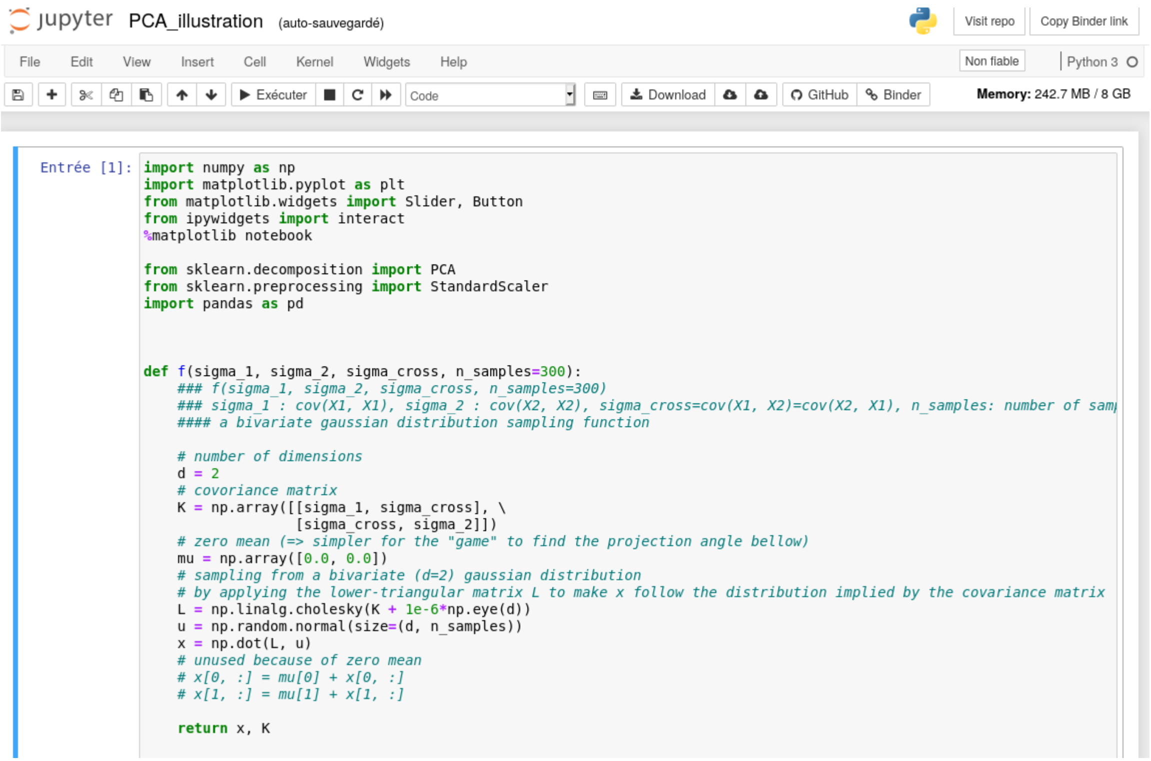
projecting the data on the first components maximize its spread (variance)

dimension reduction: select the *d* first components

Feature extraction
demo with PCA



Feature extraction
demo with PCA



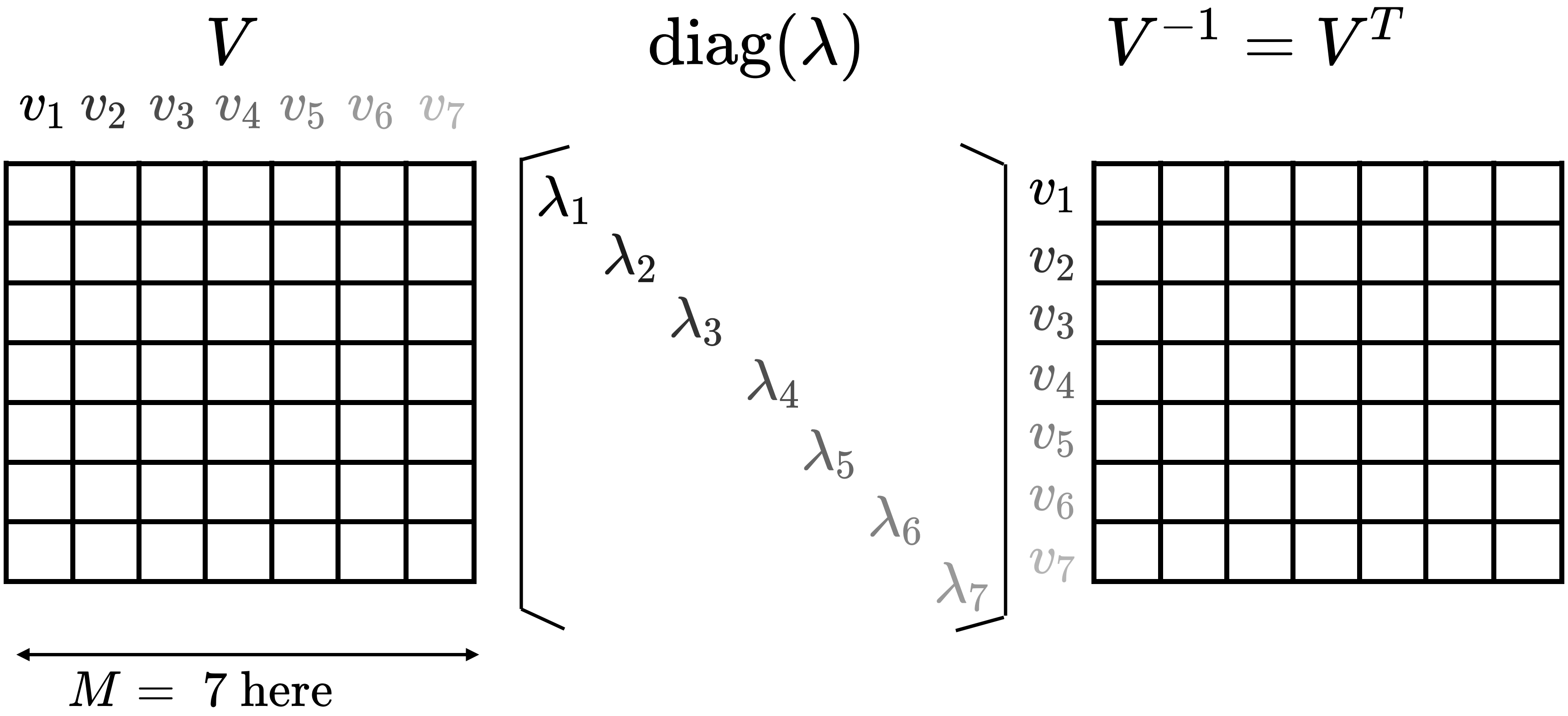
```python
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.widgets import Slider, Button
from ipywidgets import interact
%matplotlib notebook

from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
import pandas as pd



def f(sigma_1, sigma_2, sigma_cross, n_samples=300):
    ### f(sigma_1, sigma_2, sigma_cross, n_samples=300)
    ### sigma_1 : cov(X1, X1), sigma_2 : cov(X2, X2), sigma_cross=cov(X1, X2)=cov(X2, X1), n_samples: number of samp
    #### a bivariate gaussian distribution sampling function

    # number of dimensions
    d = 2
    # covoriance matrix
    K = np.array([[sigma_1, sigma_cross], \
                  [sigma_cross, sigma_2]])
    # zero mean (=> simpler for the "game" to find the projection angle bellow)
    mu = np.array([0.0, 0.0])
    # sampling from a bivariate (d=2) gaussian distribution
    # by applying the lower-triangular matrix L to make x follow the distribution implied by the covariance matrix
    L = np.linalg.cholesky(K + 1e-6*np.eye(d))
    u = np.random.normal(size=(d, n_samples))
    x = np.dot(L, u)
    # unused because of zero mean
    # x[0, :] = mu[0] + x[0, :]
    # x[1, :] = mu[1] + x[1, :]

    return x, K
```

https://github.com/emmanuelrouxfr/PCA_illustration

Feature extraction

PCA

$$D = V \mathrm{diag}(\lambda) V^{-1}$$


V
$\mathrm{diag}(\lambda)$
$V^{-1} = V^T$
v_1 v_2 v_3 v_4 v_5 v_6 v_7
λ_1 λ_2 λ_3 λ_4 λ_5 λ_6 λ_7
v_1 v_2 v_3 v_4 v_5 v_6 v_7
$M =$ 7 here

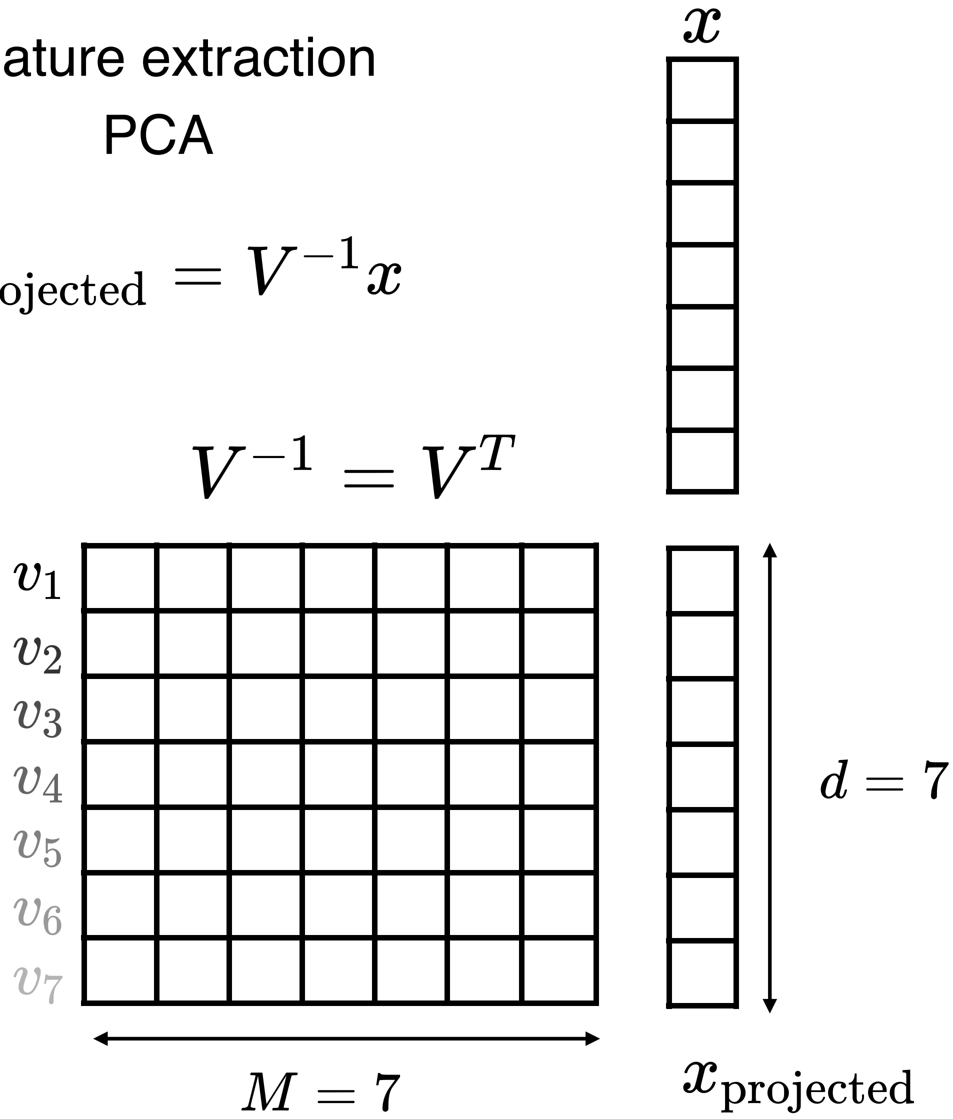
unsupervised learning

Feature extraction

PCA

$$x_{\text{projected}} = V^{-1}x$$

$$V^{-1} = V^T$$



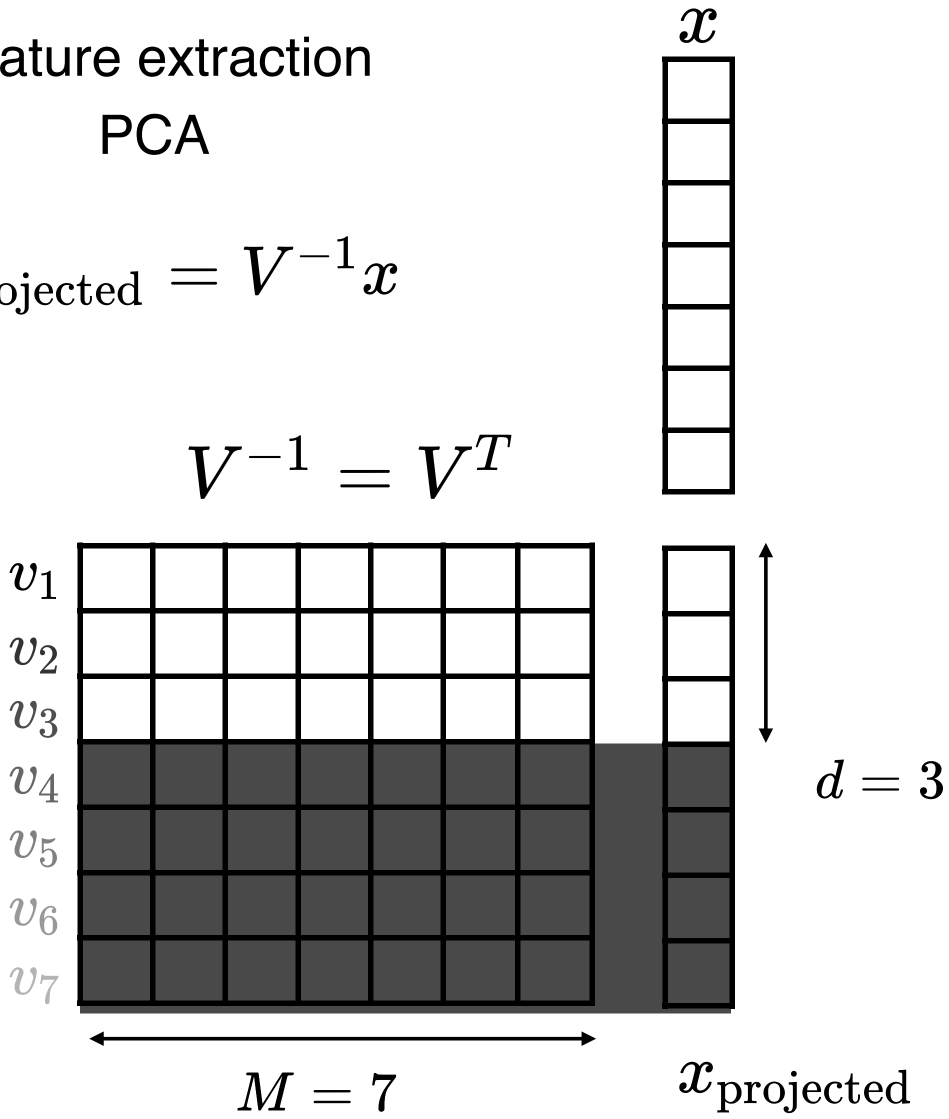
unsupervised learning

Feature extraction

PCA

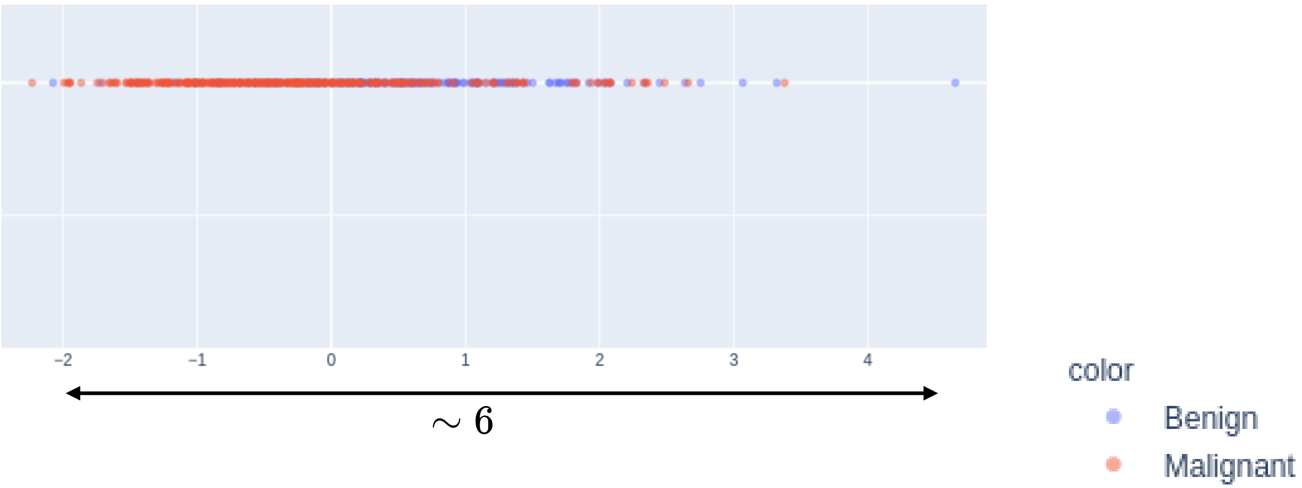
$$x_{\text{projected}} = V^{-1}x$$

$$V^{-1} = V^T$$



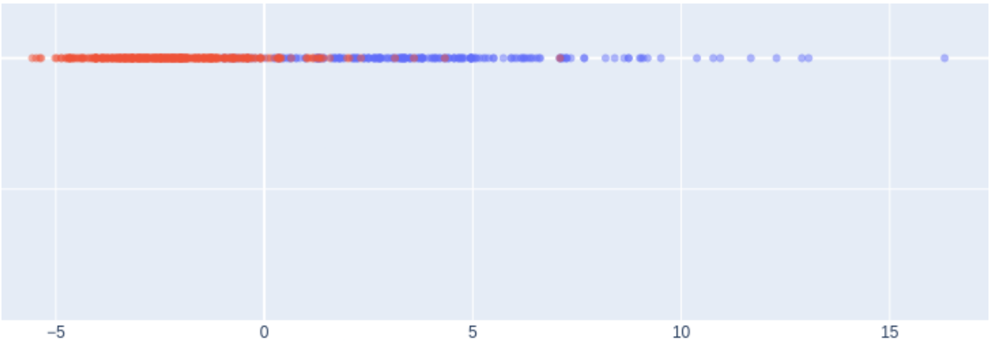
unsupervised learning

selection of the "mean texture" feature (normalized)



~ 6

1st dimension of PCA

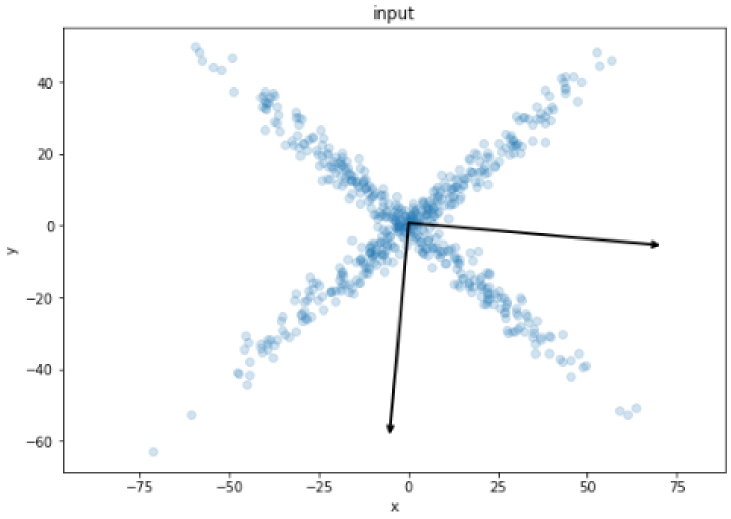


~ 20

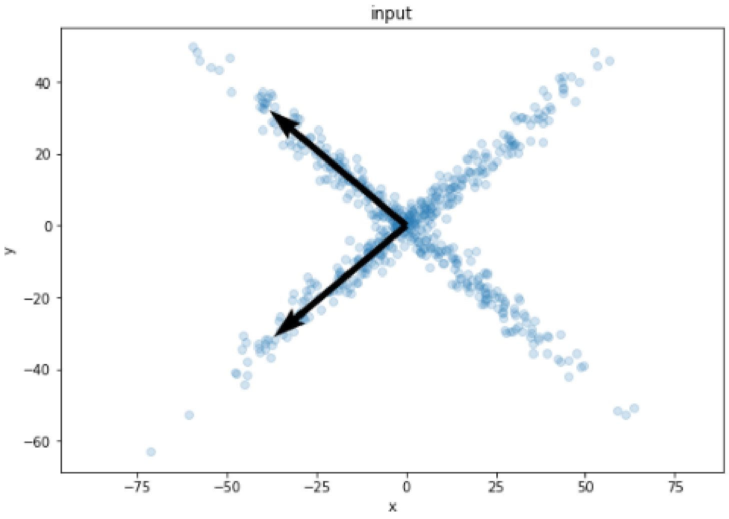
unsupervised learning

Dimension reduction (linear)

ICA



PCA



ICA

non-linear
dimension reduction

ISOMAP

Locally Linear Embedding

Hessian Eigenmapping

Local Tangent Space Alignment

t-distributed Stochastic Neighbor Embedding (t-SNE)

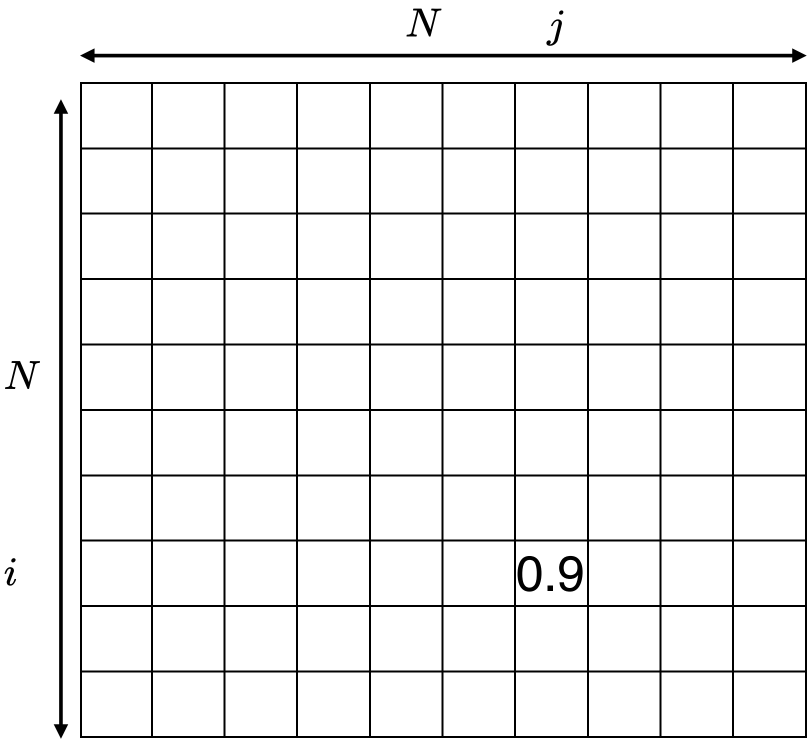
UMAP

(deep) auto-encoders...

unsupervised learning

Dimension reduction (non-linear)

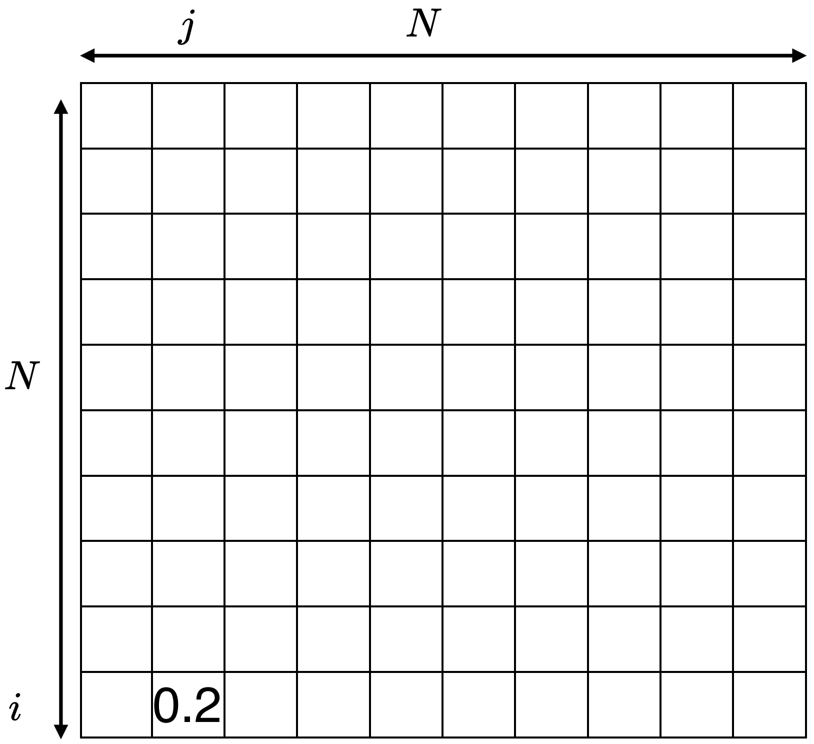
t-distributed Stochastic Neighbor Embedding (t-SNE)



similarity matrix in input space

Dimension reduction (non-linear)

t-distributed Stochastic Neighbor Embedding (t-SNE)



similarity matrix in input space

unsupervised learning

CREATIS

Dimension reduction (non-linear)

t-distributed Stochastic Neighbor Embedding (t-SNE)

N

N

						0.9			
	0.2								

similarity matrix in input space

N

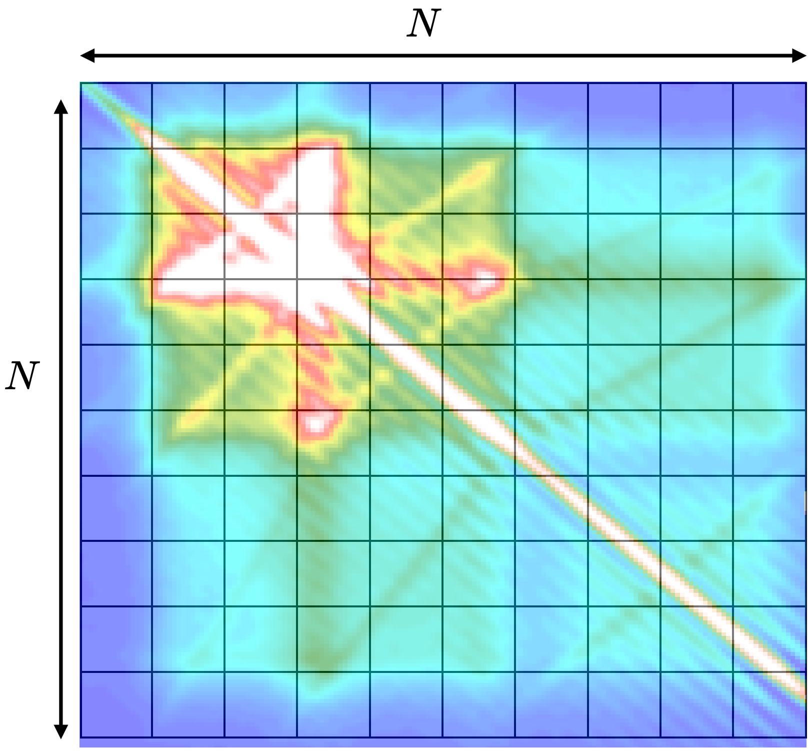
N

						0.8			
	0.3								

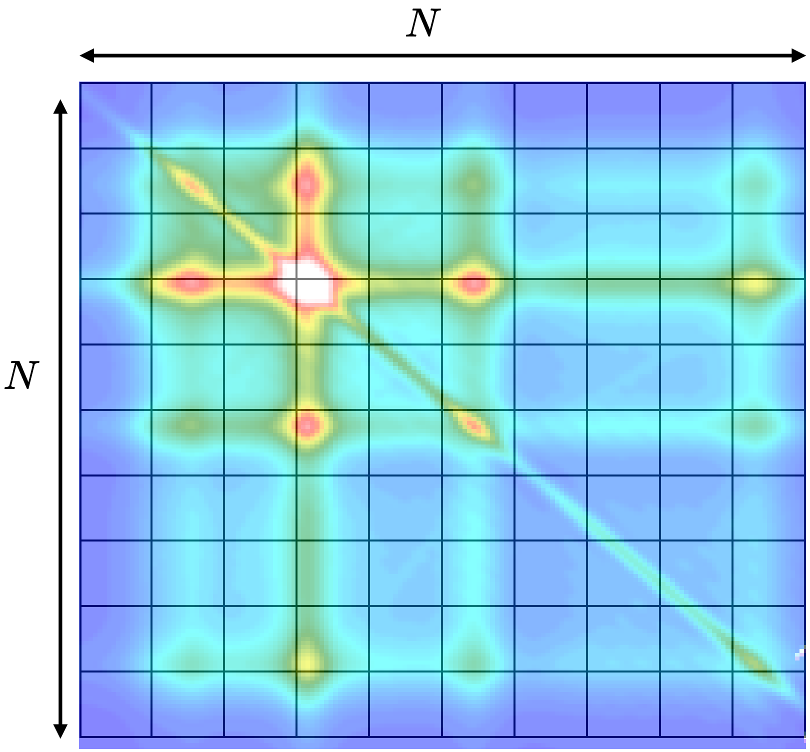
similarity matrix in lower space

Dimension reduction (non-linear)

t-distributed Stochastic Neighbor Embedding (t-SNE)



similarity matrix in input space



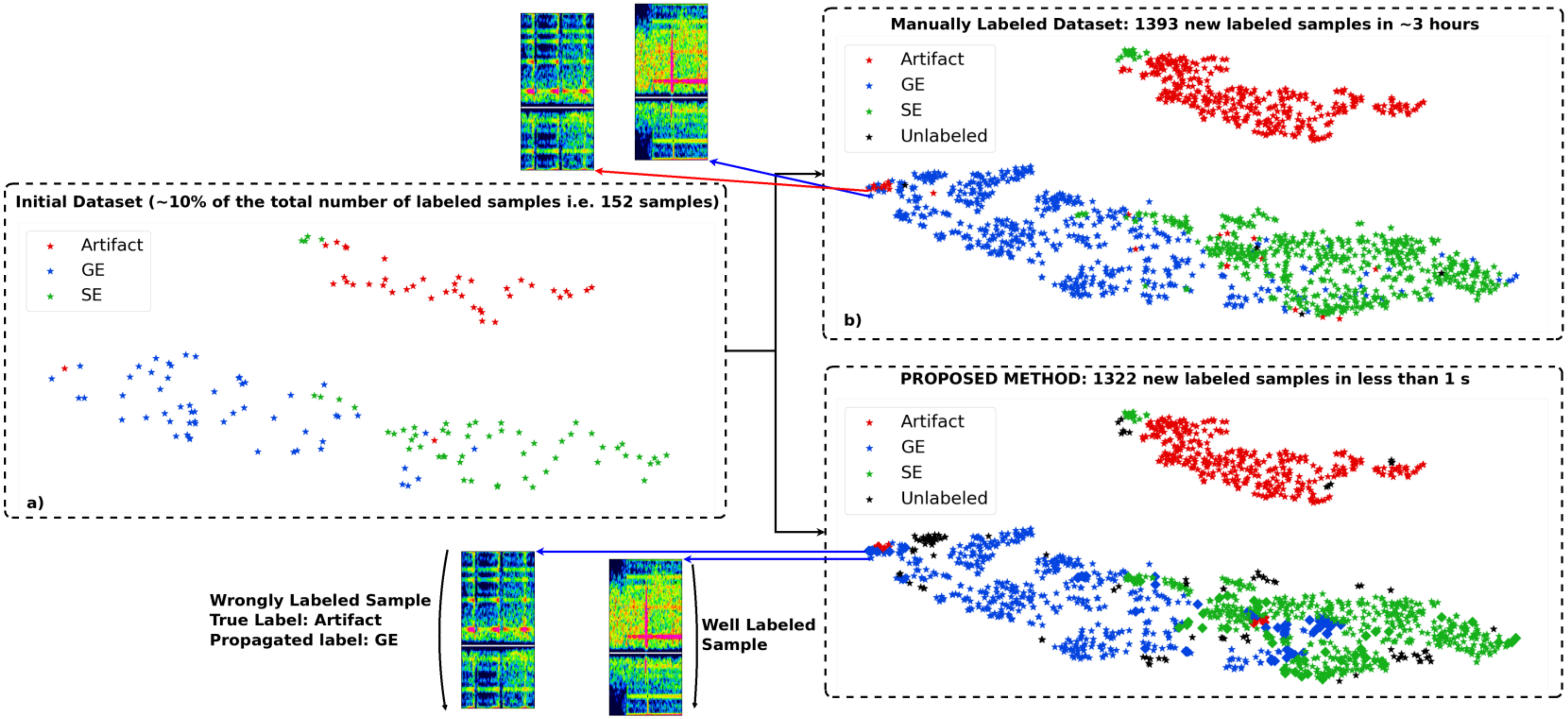
similarity matrix in lower space

L J Frasinski

unsupervised learning

example of t-SNE application

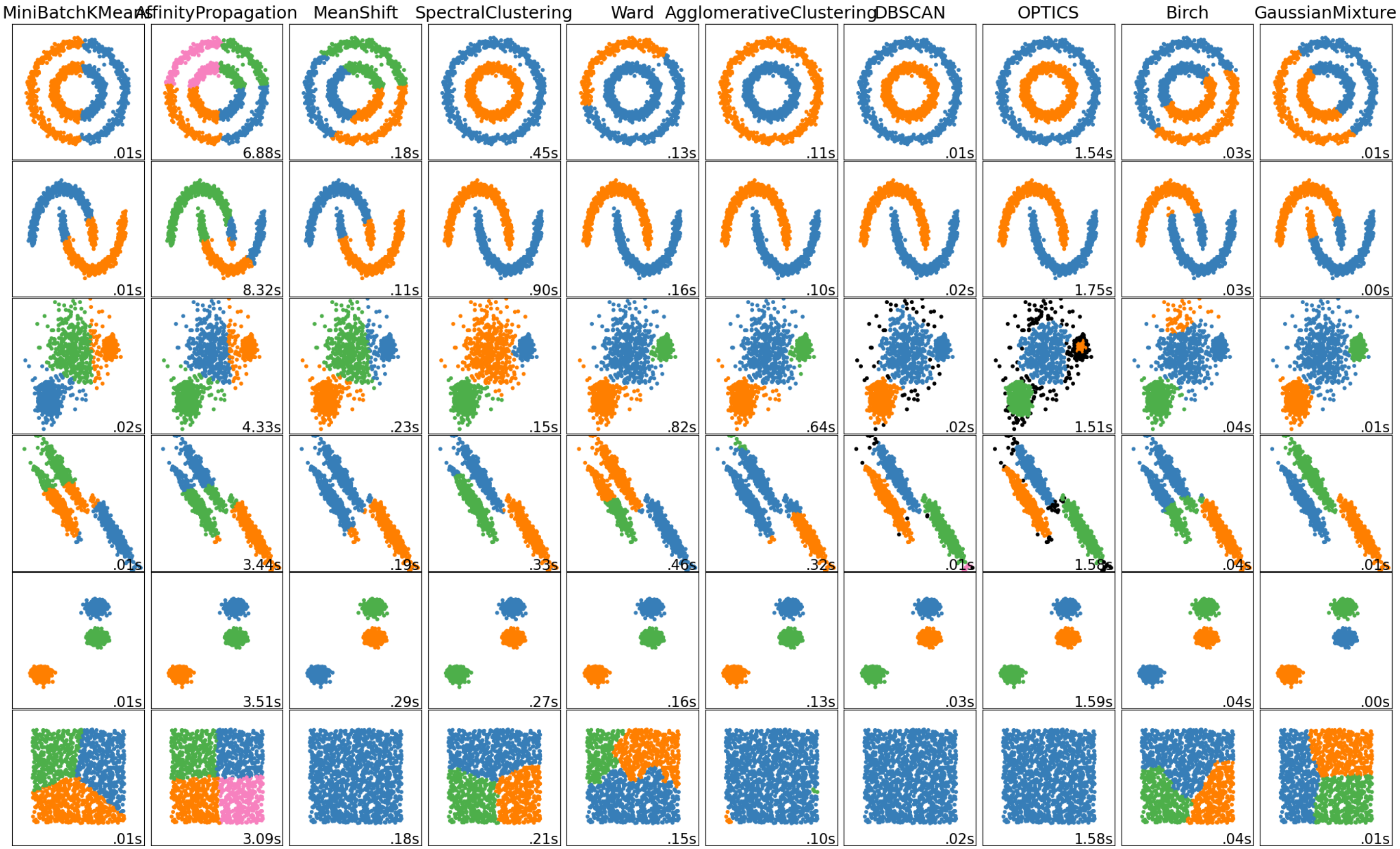
accelerating the annotation of a
Transcranial Doppler ultrasound micro-embolic dataset



(Vindas et al. 2021, IEEE IUS 2021 *submitted*)

clustering

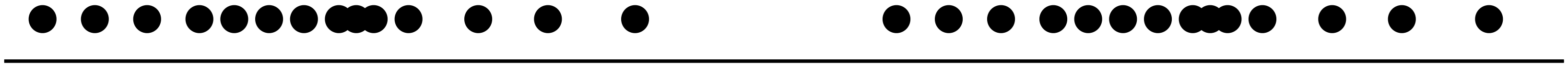
Find groups of similar examples (clusters)



unsupervised learning

clustering

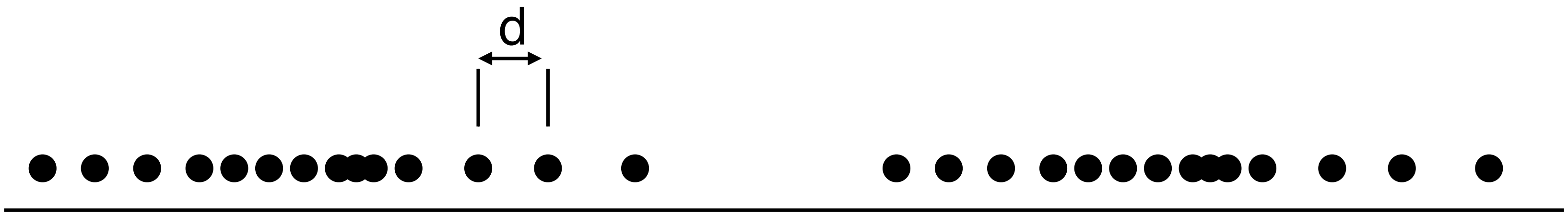
what is a cluster ?



clustering

what is a cluster ?

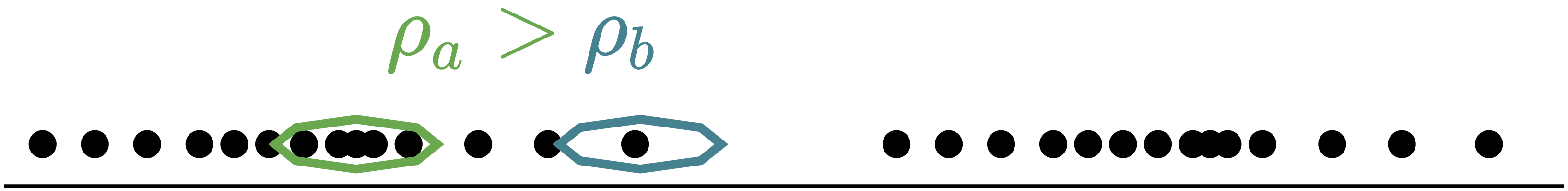
- distance-based definition



clustering

what is a cluster ?

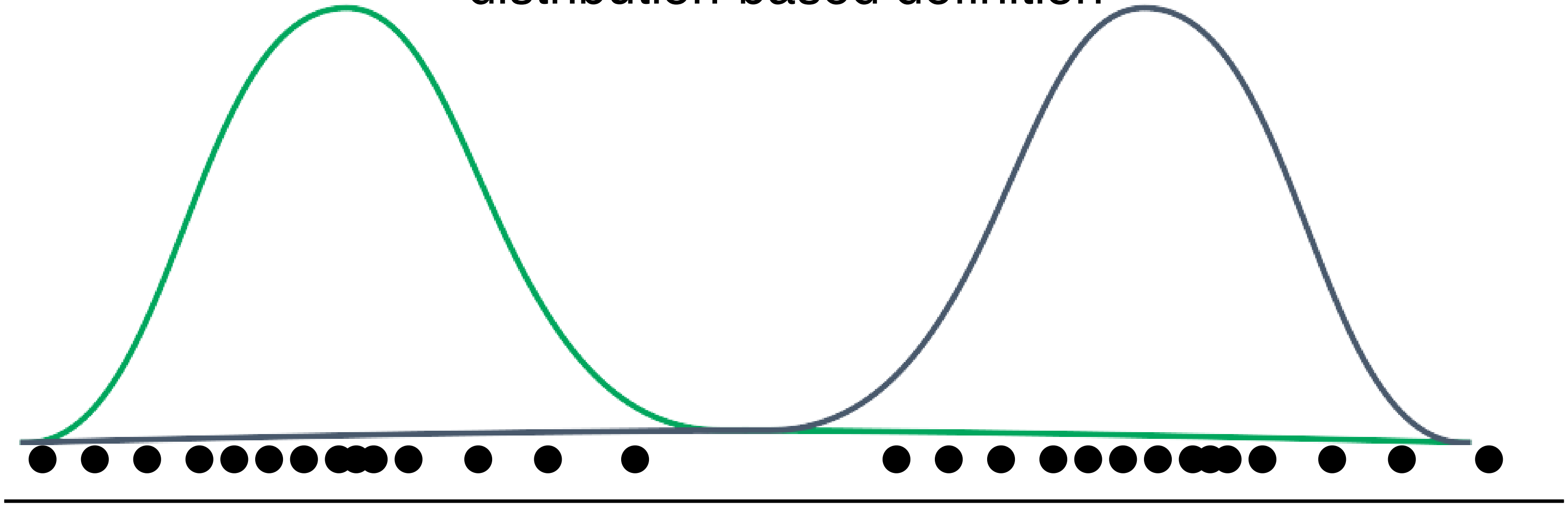
- distance-based definition
- density-based definition



clustering

what is a cluster ?

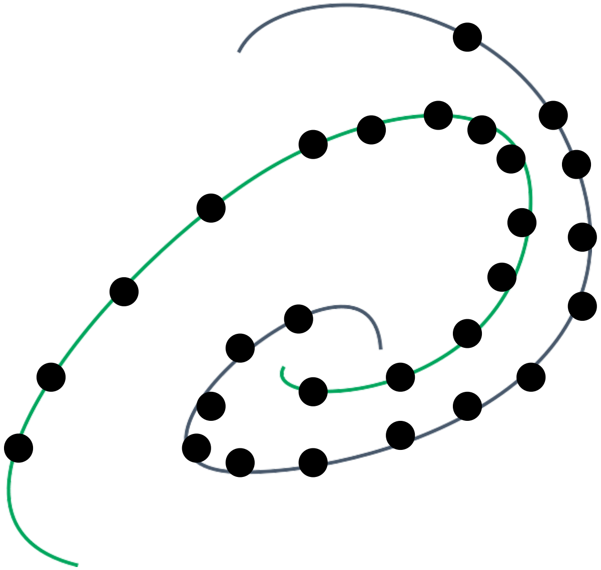
- distance-based definition
- density-based definition
- distribution-based definition



clustering

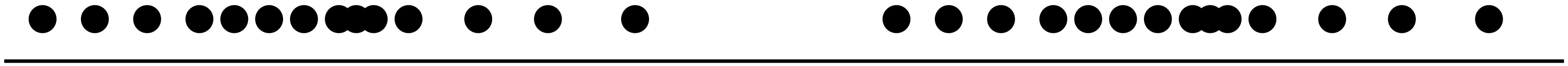
what is a cluster ?

- distance-based definition
- density-based definition
- distribution-based definition
- path-based distribution (graphs)



clustering

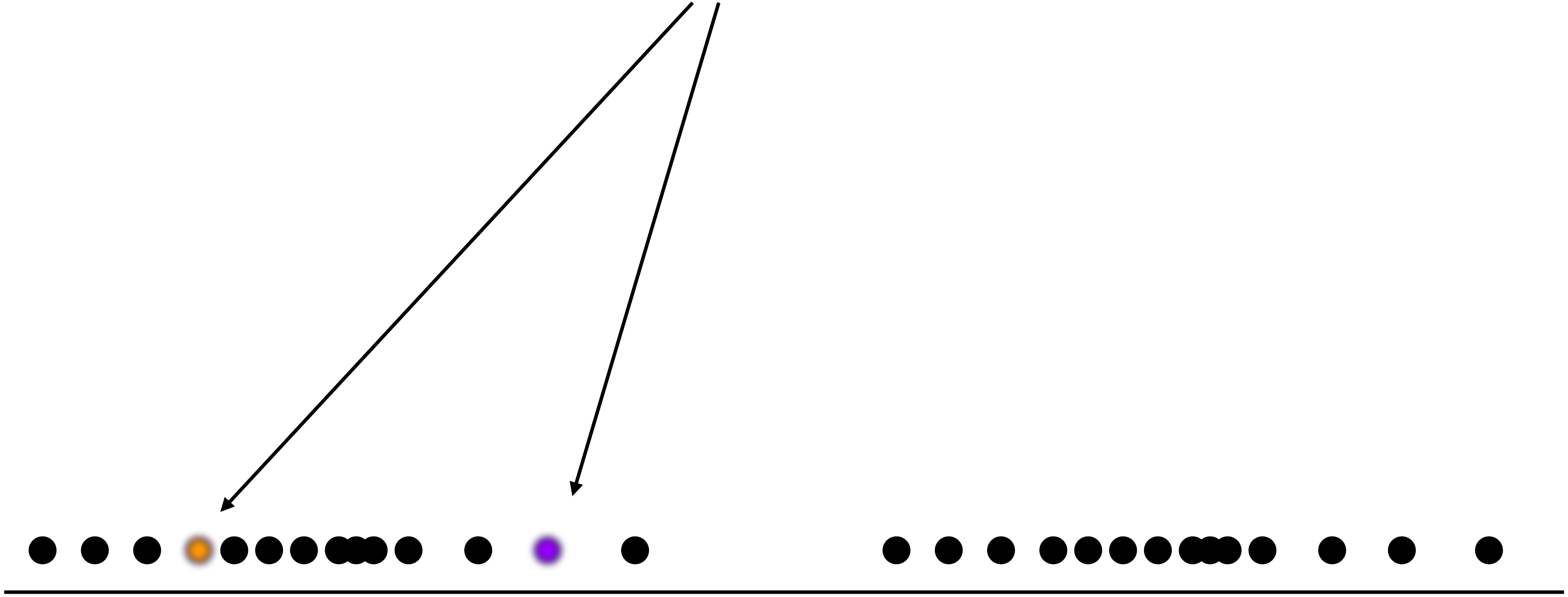
K-means (distance-based method)



clustering

K-means (distance-based method)

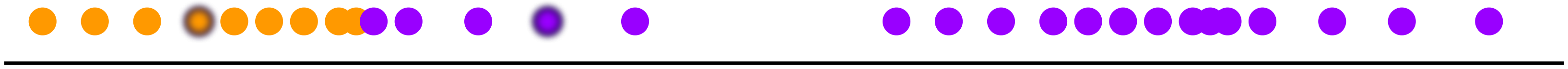
1. initialize k samples as **centers** *



clustering

K-means (distance-based method)

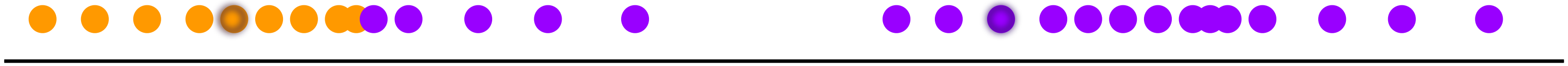
1. initialize k samples as centers
2. for each sample associate the label of its **closest center**



clustering

K-means (distance-based method)

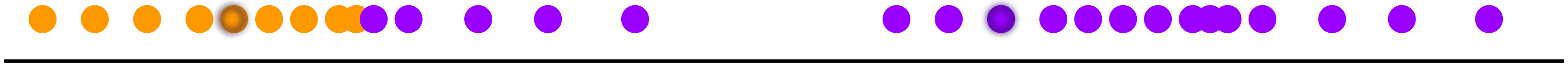
1. initialize k samples as **centers** *
2. for each sample associate the label of its **closest center**
3. update the centers (mean position of its group)



clustering

K-means (distance-based method)

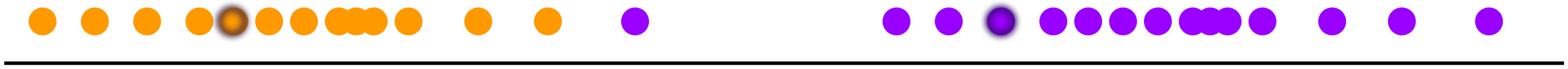
1. initialize k samples as **centers** *
2. for each sample associate the label of its **closest center**
3. update the centers (mean position of its group)
4. repeat steps 2. and 3. until no update in the clusters



clustering

K-means (distance-based method)

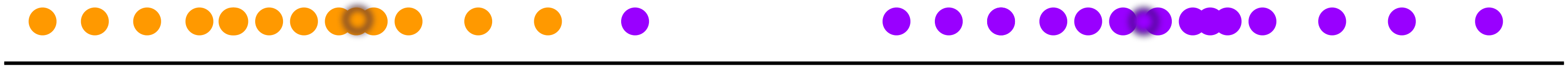
1. initialize k samples as **centers** *
2. for each sample associate the label of its **closest center**
3. update the centers (mean position of its group)
4. repeat steps 2. and 3. until no update in the clusters



clustering

K-means (distance-based method)

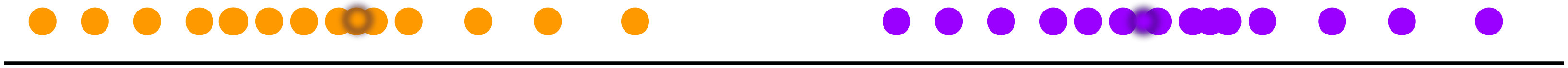
1. initialize k samples as **centers** *
2. for each sample associate the label of its **closest center**
3. update the centers (mean position of its group)
4. repeat steps 2. and 3. until no update in the clusters



clustering

K-means (distance-based method)

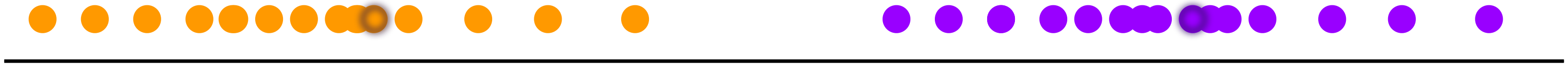
1. initialize k samples as **centers** *
2. for each sample associate the label of its **closest center**
3. update the centers (mean position of its group)
4. repeat steps 2. and 3. until no update in the clusters



clustering

K-means (distance-based method)

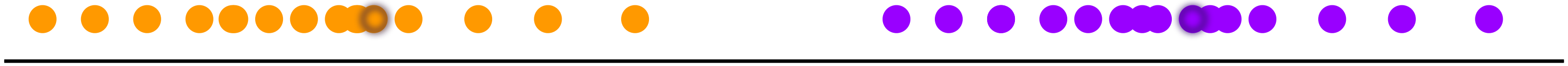
1. initialize k samples as **centers** *
2. for each sample associate the label of its **closest center**
3. update the centers (mean position of its group)
4. repeat steps 2. and 3. until no update in the clusters



clustering

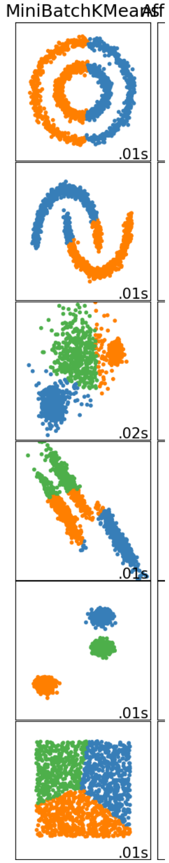
K-means (distance-based method)

1. initialize k samples as **centers** *
2. for each sample associate the label of its **closest center**
3. update the centers (mean position of its group)
4. repeat steps 2. and 3. until no update in the clusters



clustering

K-means (distance-based method)



+ fast (O(n))

- need to know / find *k* (number of clusters)
- can detect only circular clusters

alt. k-median (more computation because need to sort...)

clustering

hierarchical clustering (distance-based method)

agglomerative (bottom up) or divisive (top-down)

use of an appropriate metric *d* (between samples *a* and *b*)

and

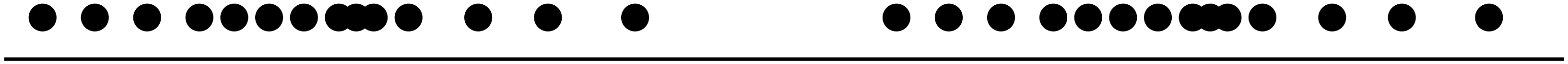
a linkage criterion (dissimilarity between sets)

example: single-linkage clustering

$$\min\{d(a,b) : a \in A, b \in B\}$$

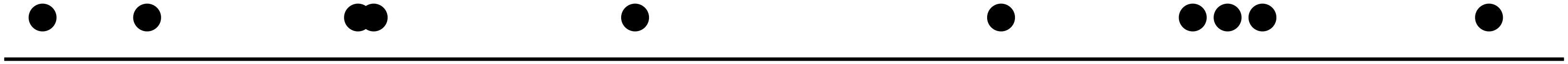
clustering

hierarchical clustering (distance-based method)



clustering

hierarchical clustering (distance-based method)

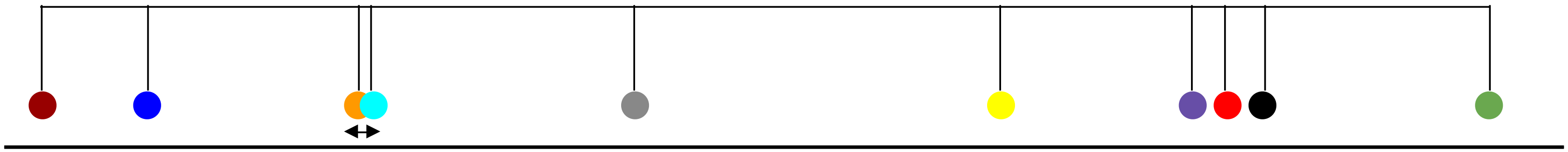


clustering

hierarchical clustering (distance-based method)

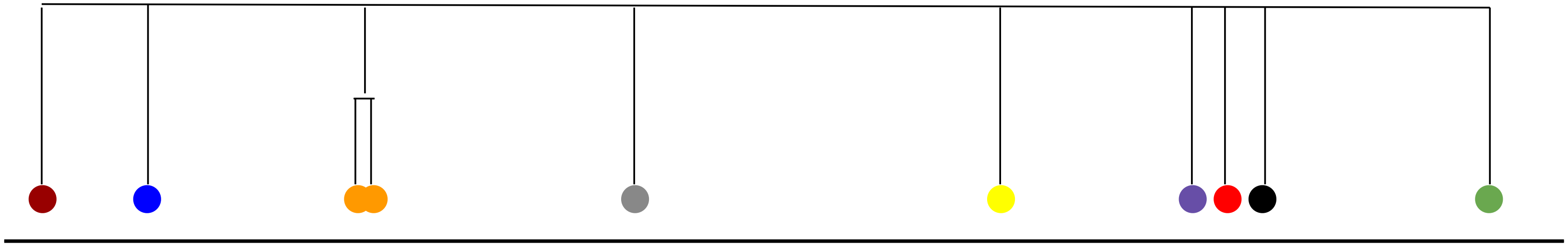
clustering

hierarchical clustering (distance-based method)



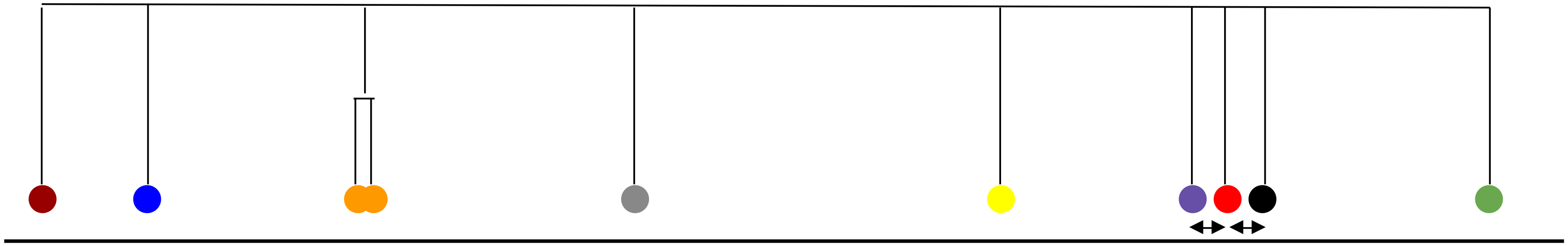
clustering

hierarchical clustering (distance-based method)



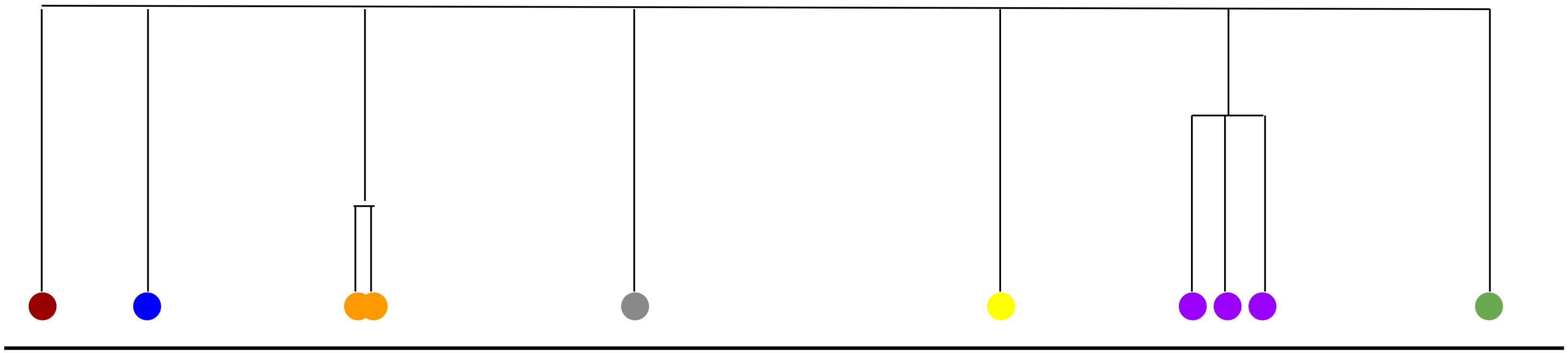
clustering

hierarchical clustering (distance-based method)



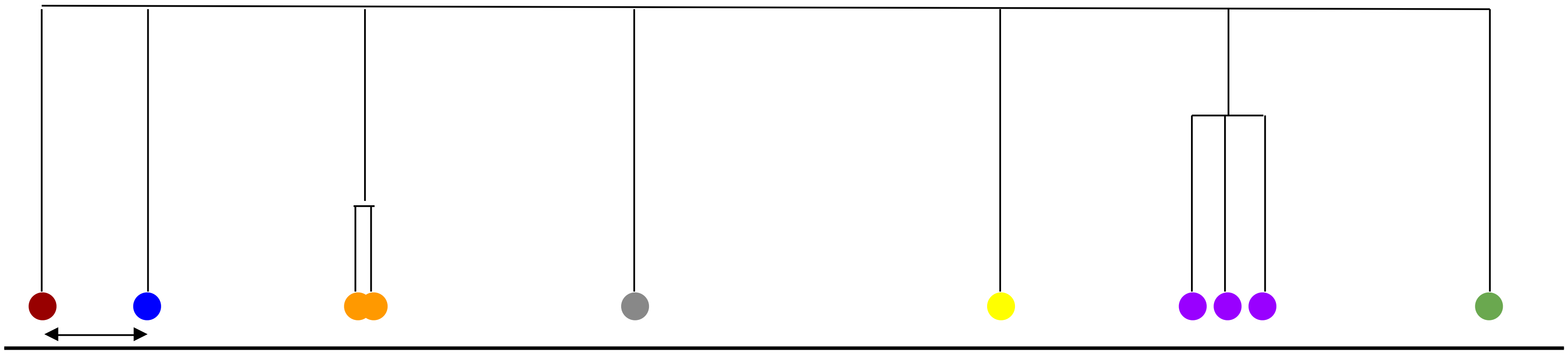
clustering

hierarchical clustering (distance-based method)



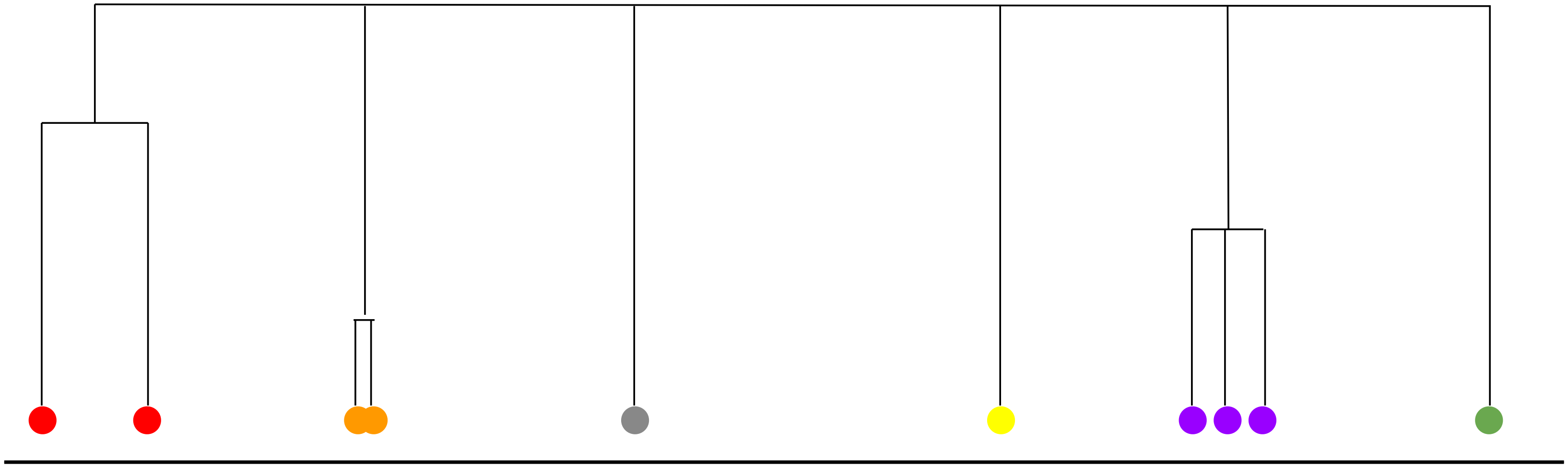
clustering

hierarchical clustering (distance-based method)



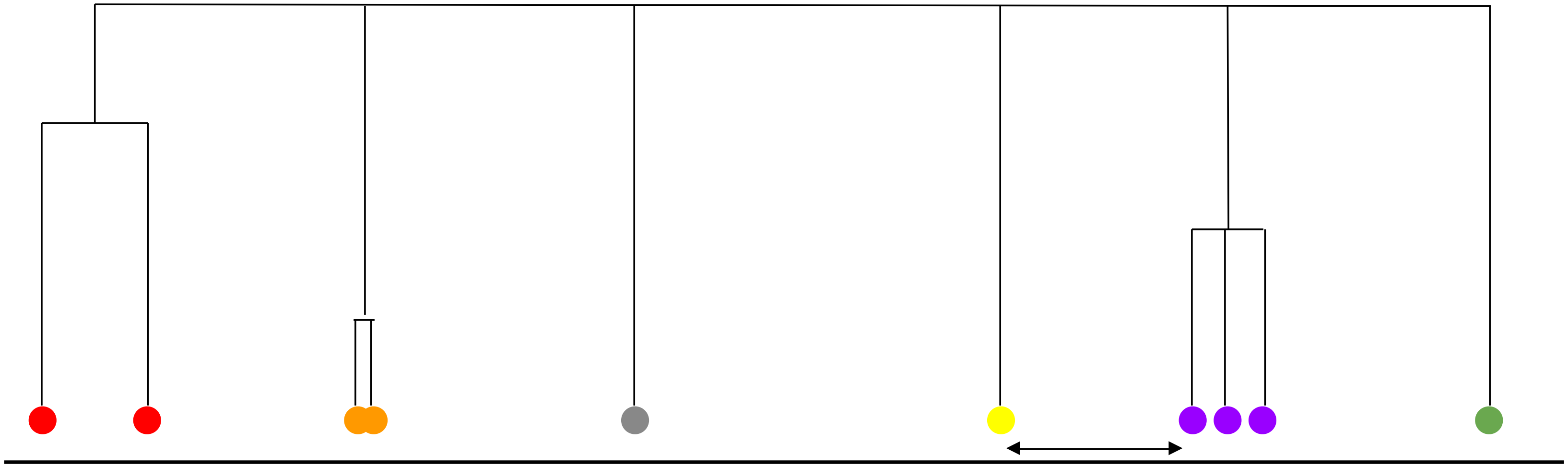
clustering

hierarchical clustering (distance-based method)



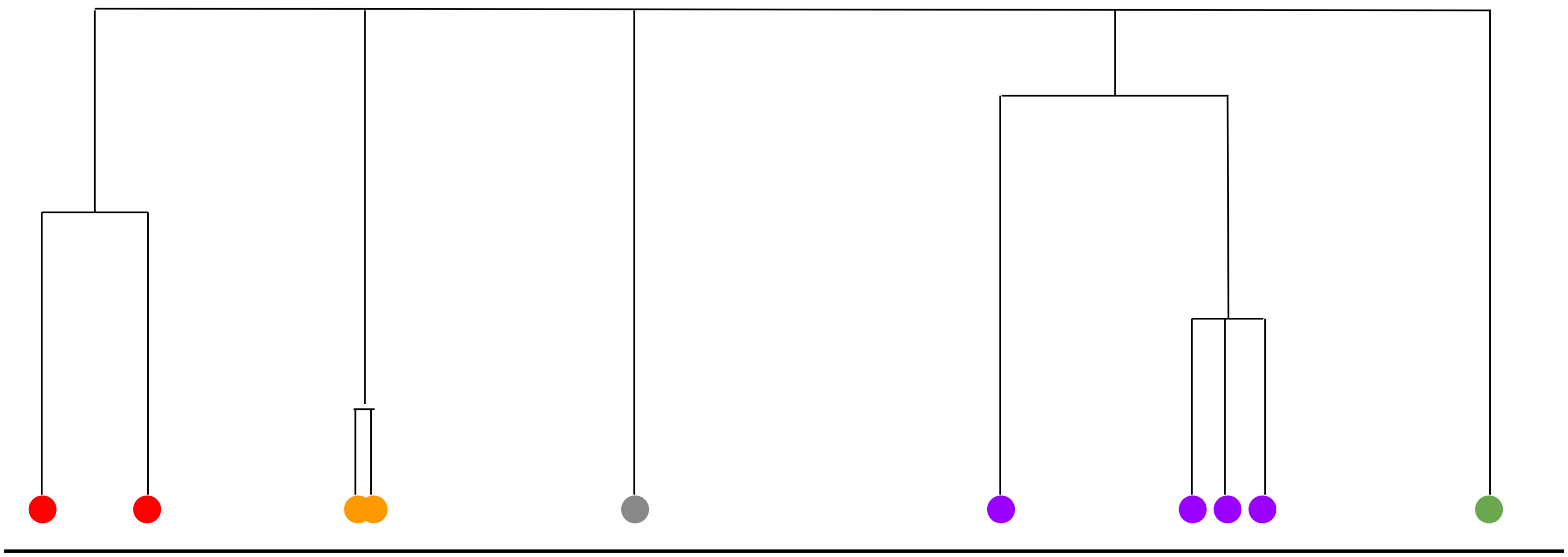
clustering

hierarchical clustering (distance-based method)



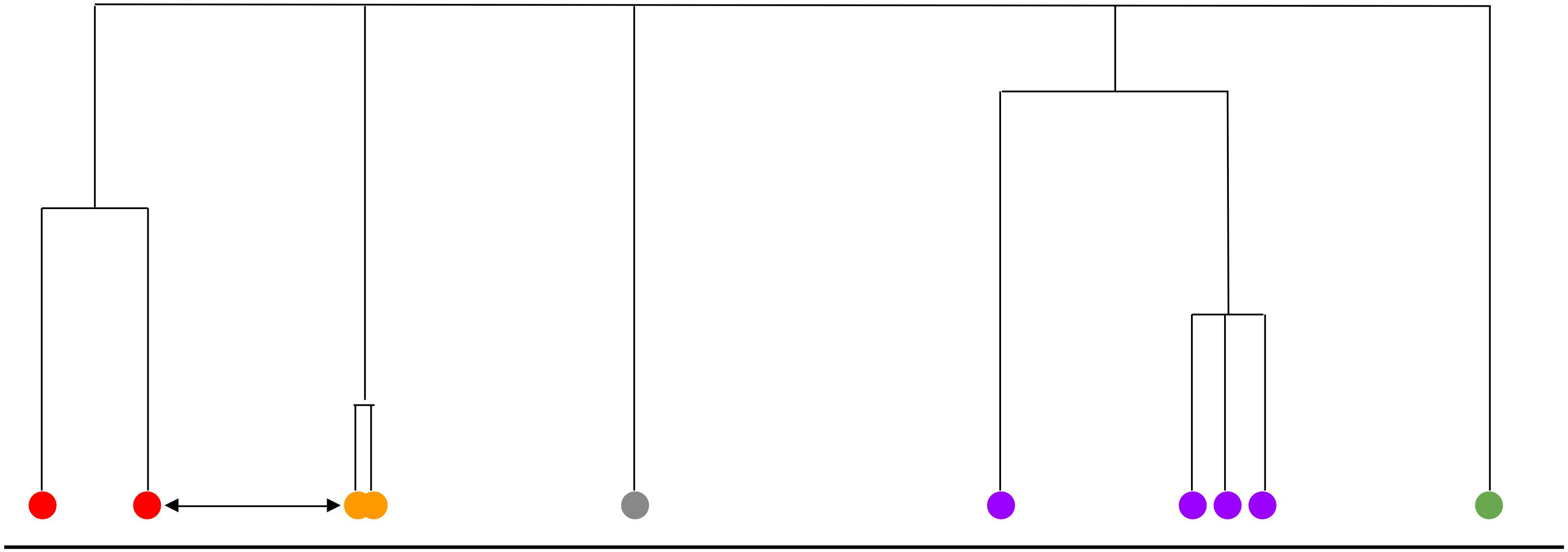
clustering

hierarchical clustering (distance-based method)



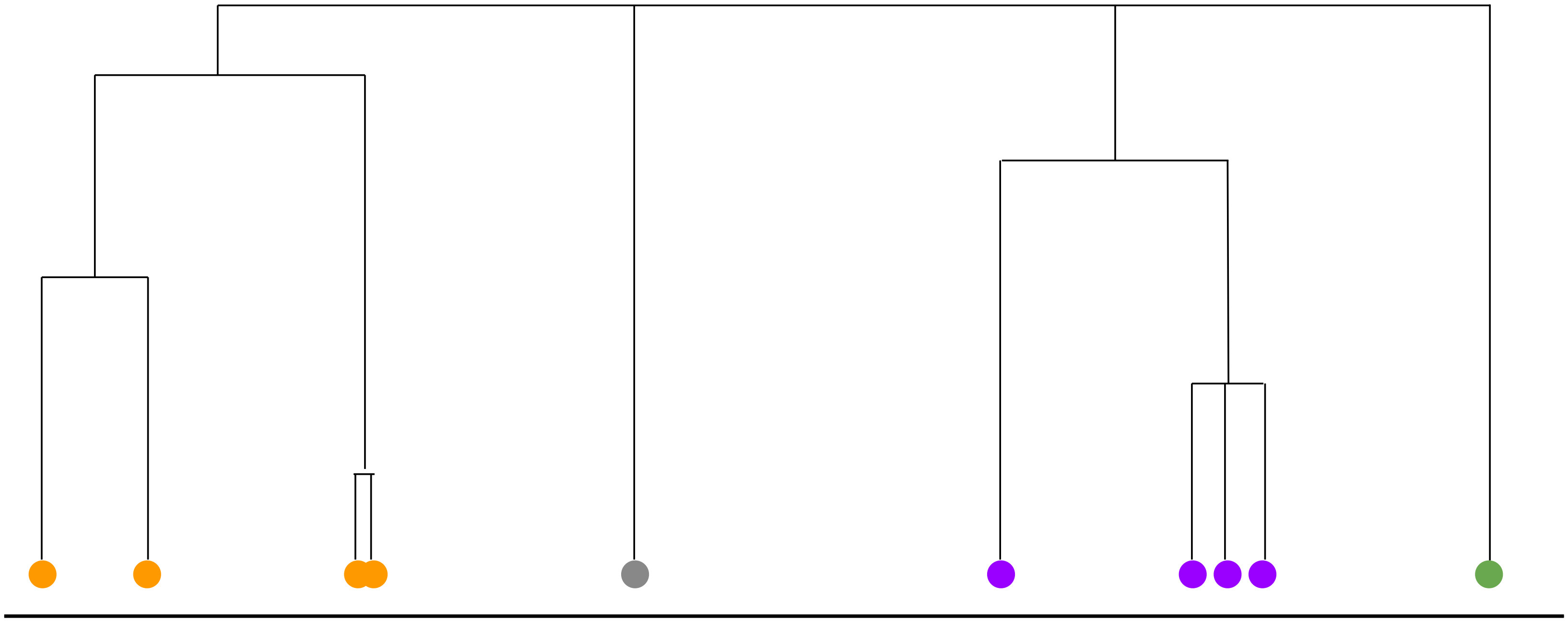
clustering

hierarchical clustering (distance-based method)



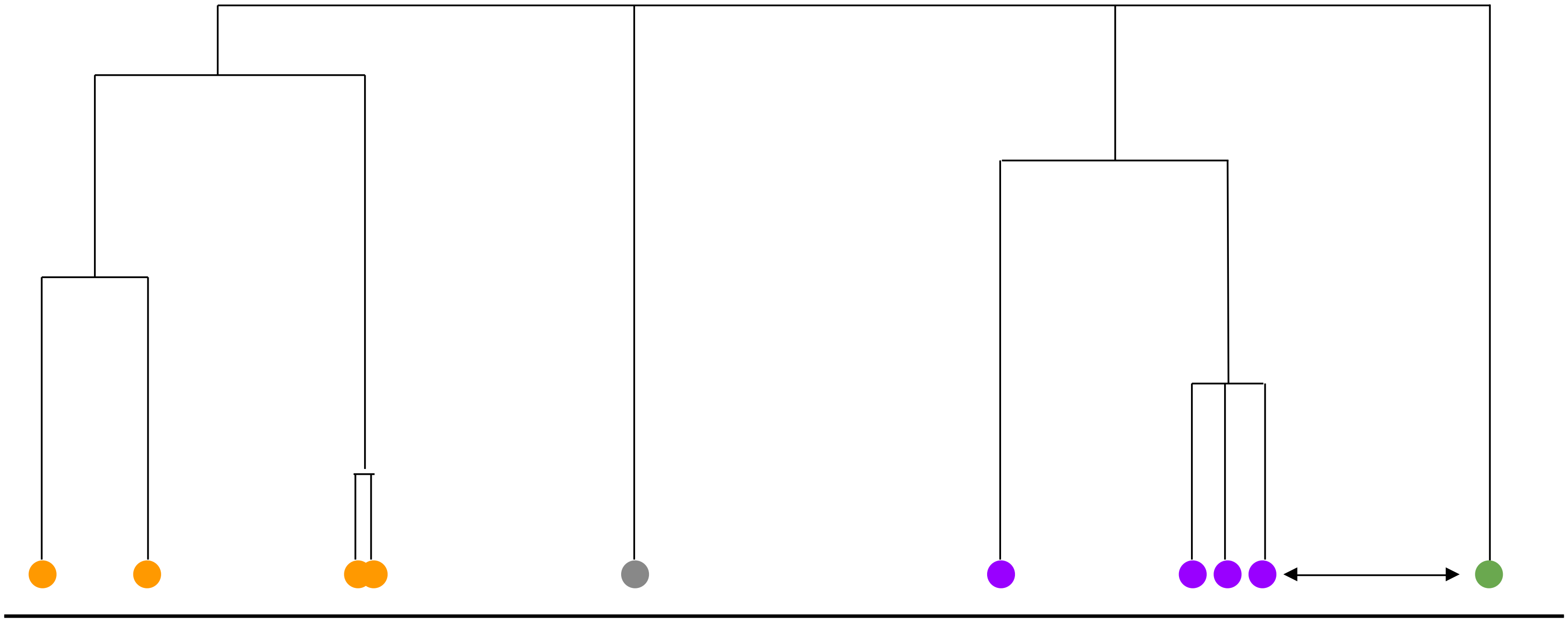
clustering

hierarchical clustering (distance-based method)



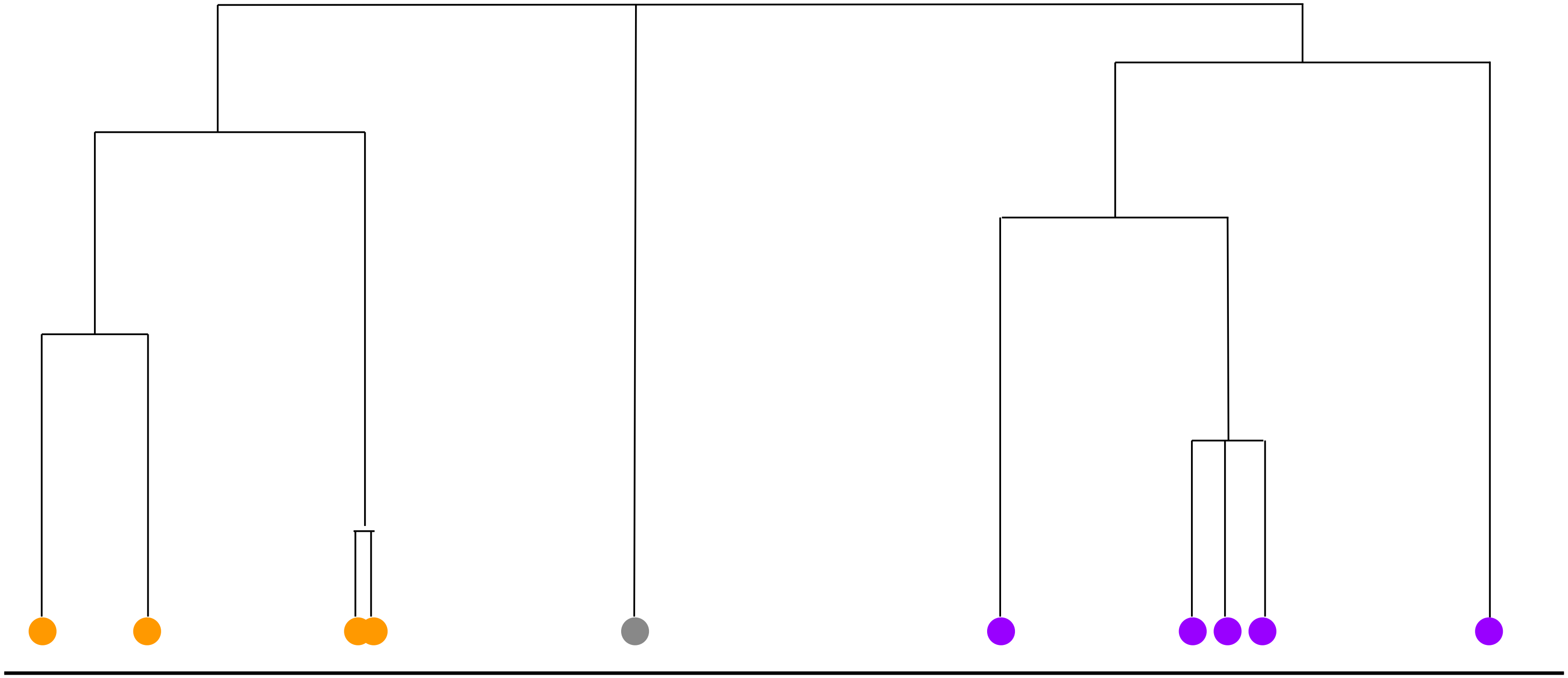
clustering

hierarchical clustering (distance-based method)

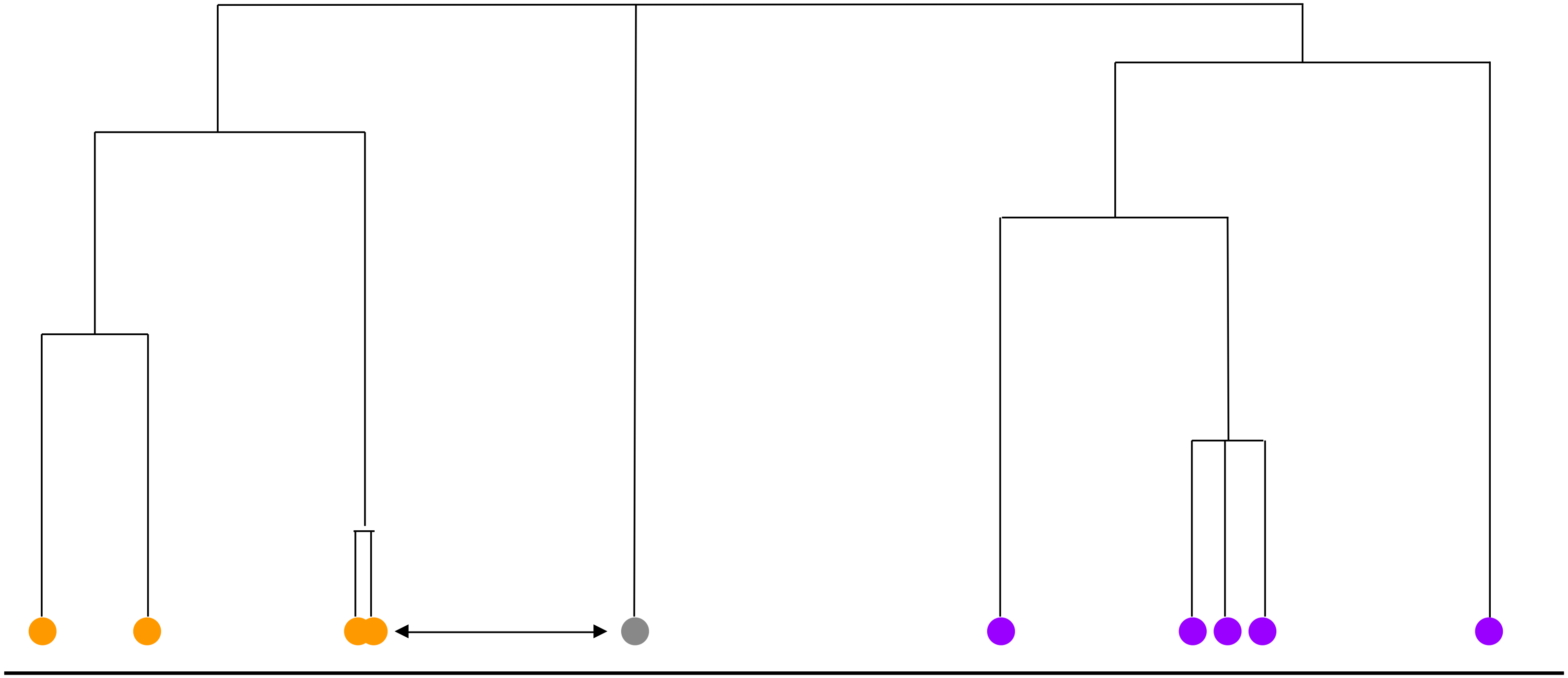


clustering

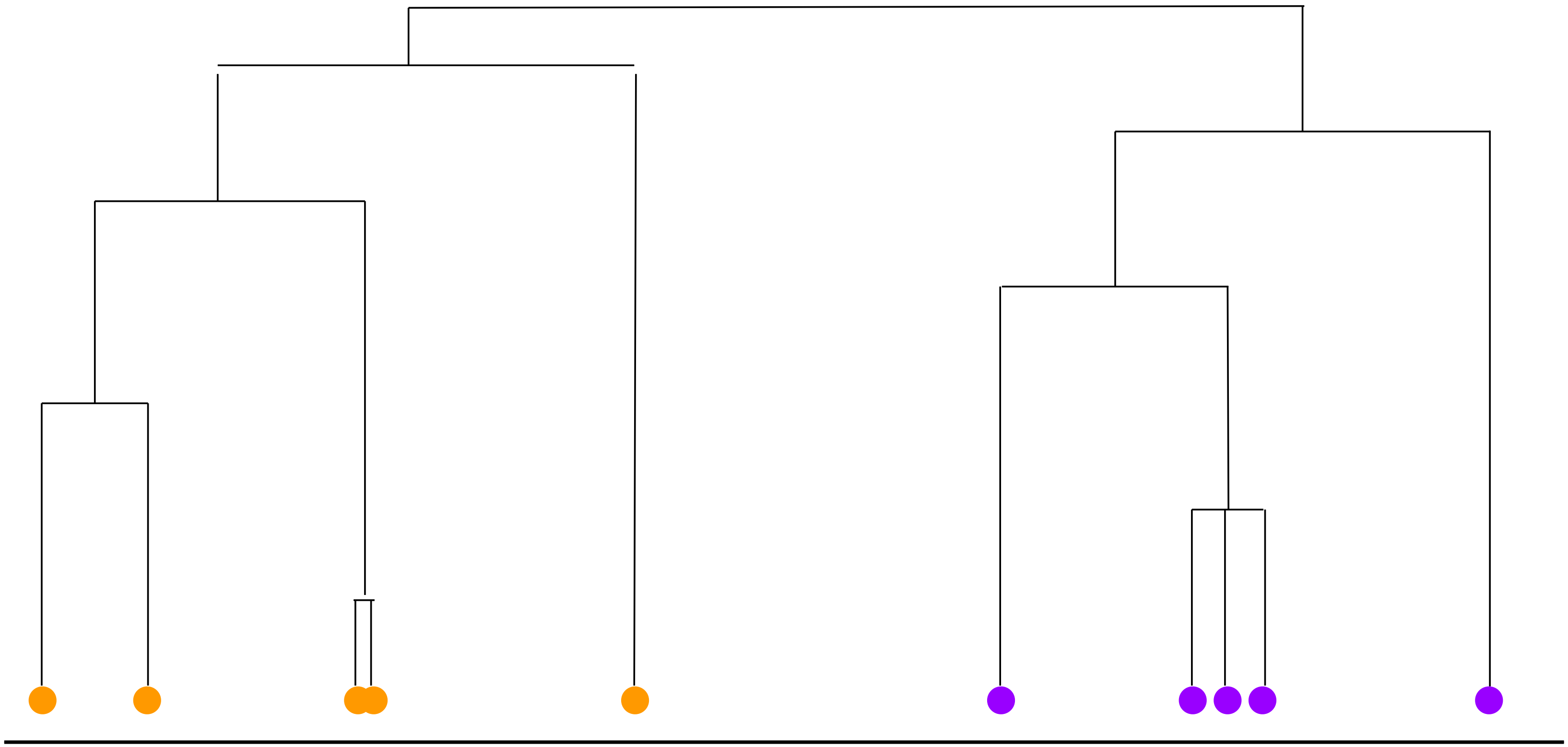
hierarchical clustering (distance-based method)



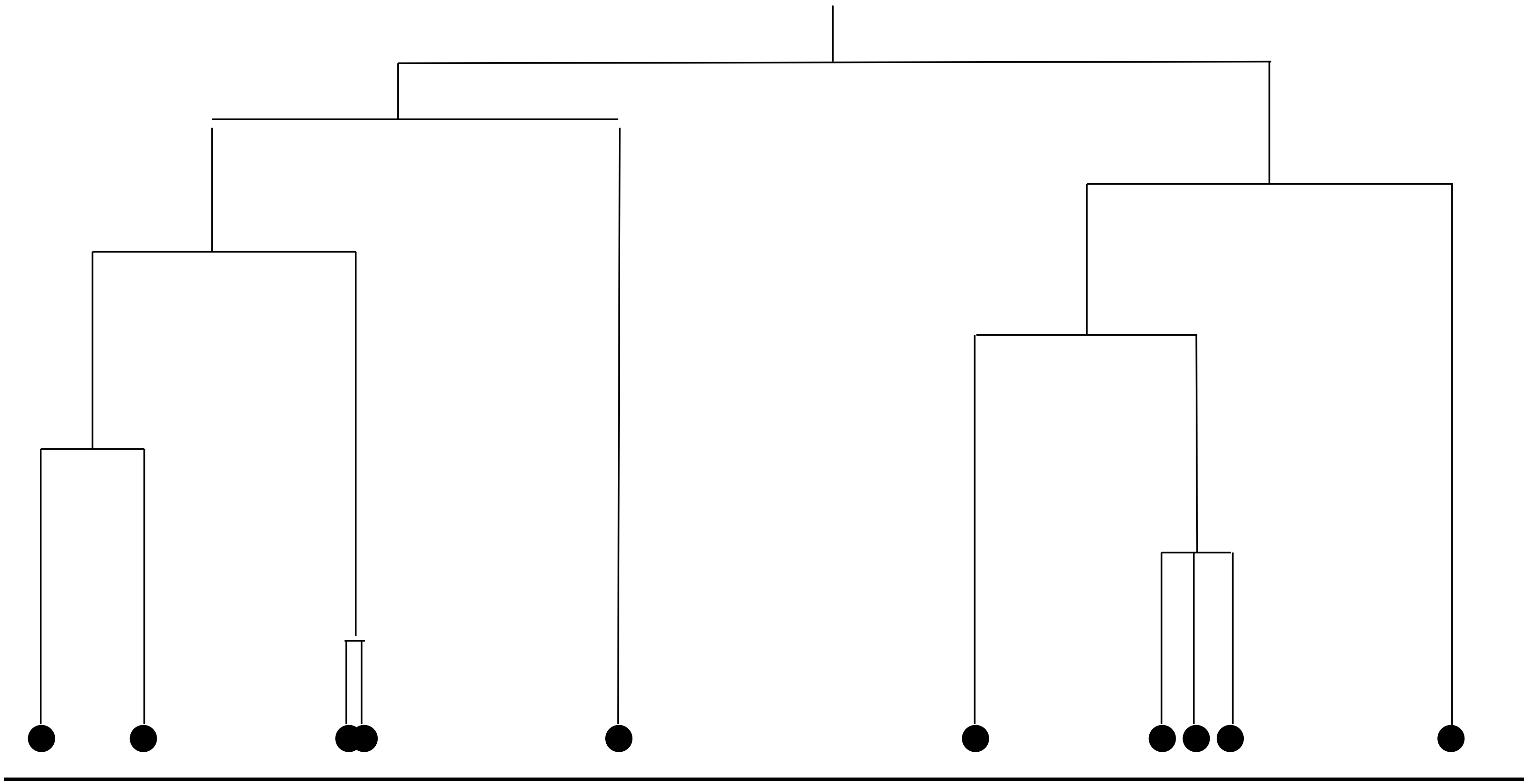
clustering



clustering

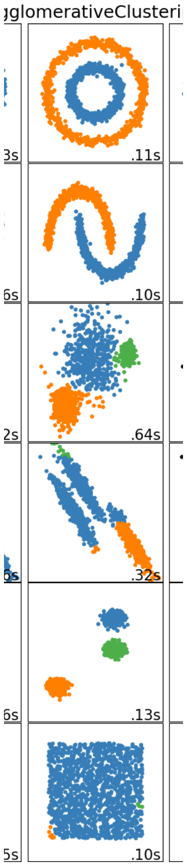


clustering



clustering

hierarchical clustering (distance-based method)

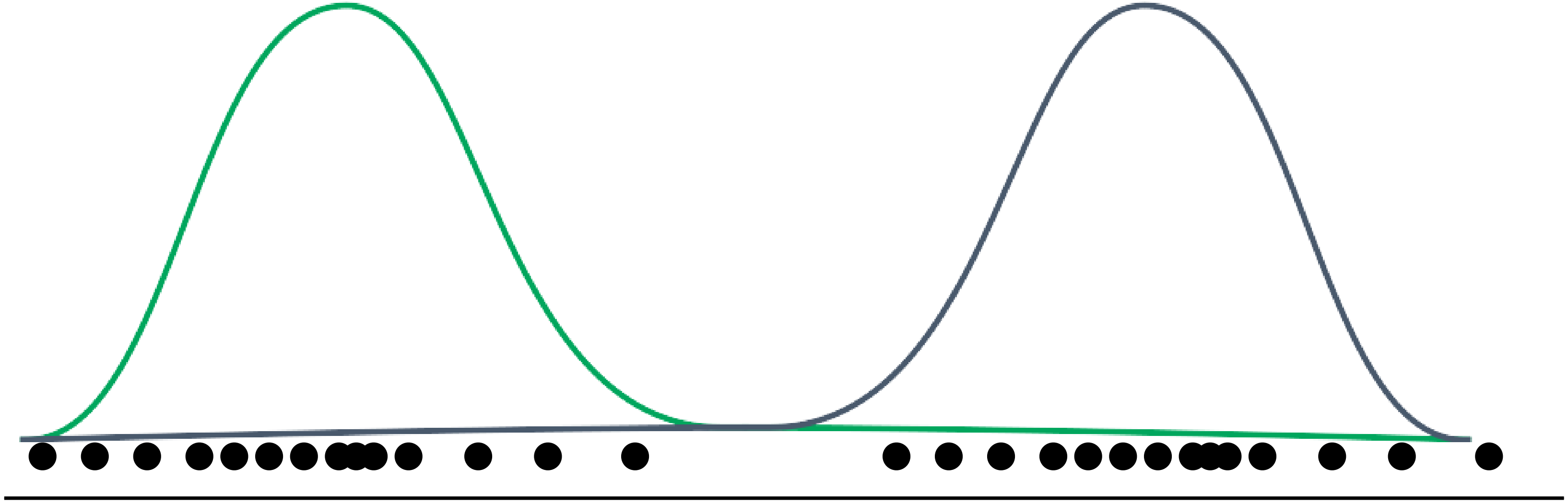


+ does not need to know the number of clusters before.
+ does not depend on the chosen distance metric (source?)
+ sub-groups discovery

- lower efficiency, O(n^3)

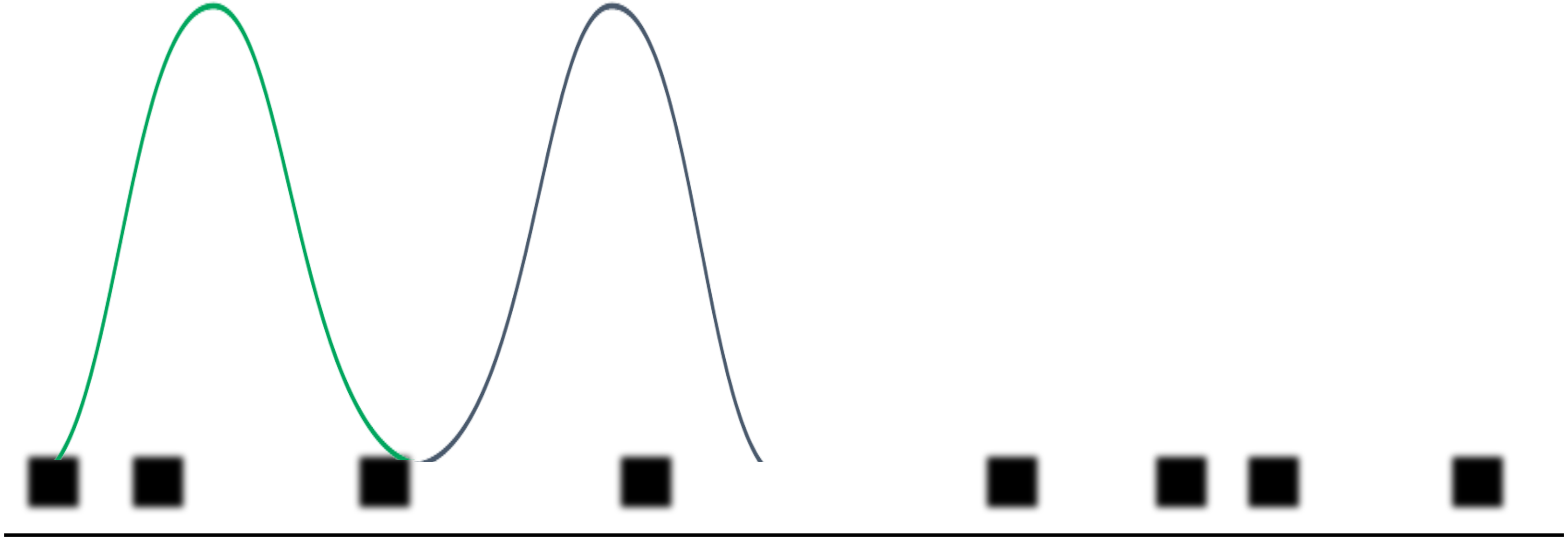
Gaussian Mixture Model with Expected-Maximization
(distribution-based method)

k-means with probability of assignment
(instead of closest point assignment)



Gaussian Mixture Model with Expected-Maximization (distribution-based method)

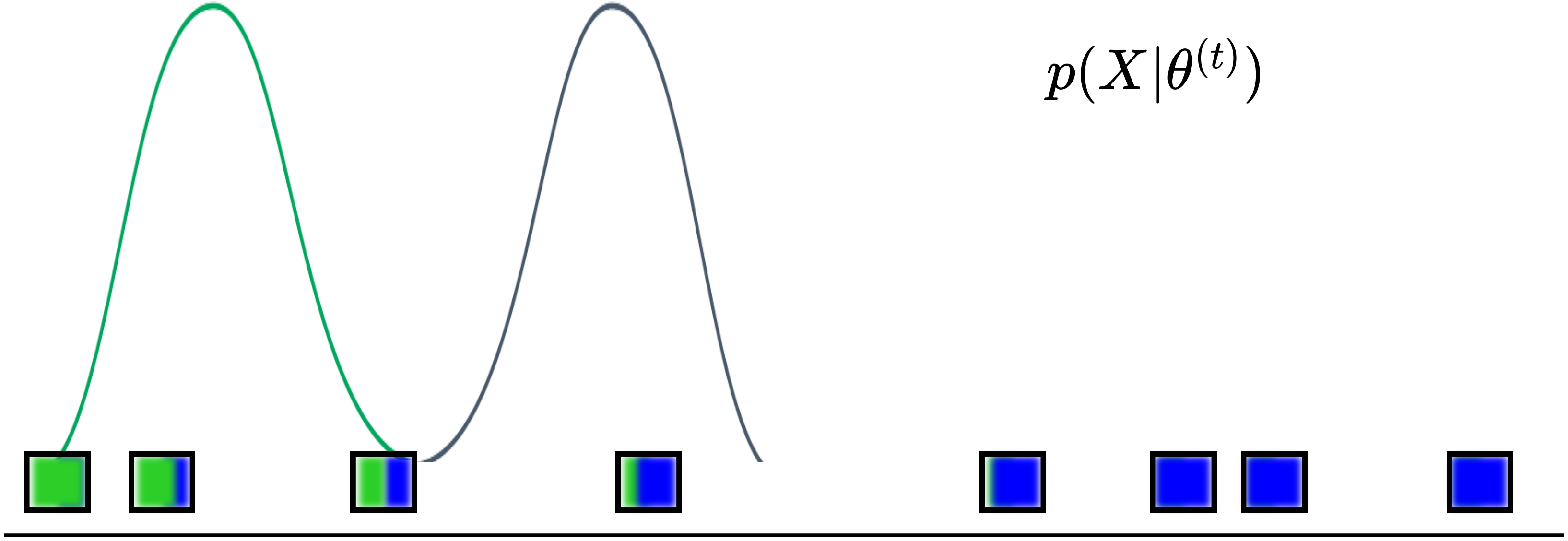
initialize the k = 2 distribution (*several strategies)



Gaussian Mixture Model with Expected-Maximization (distribution-based method)

Expectation (E) step

find the probability for each point to be generated by each mixture



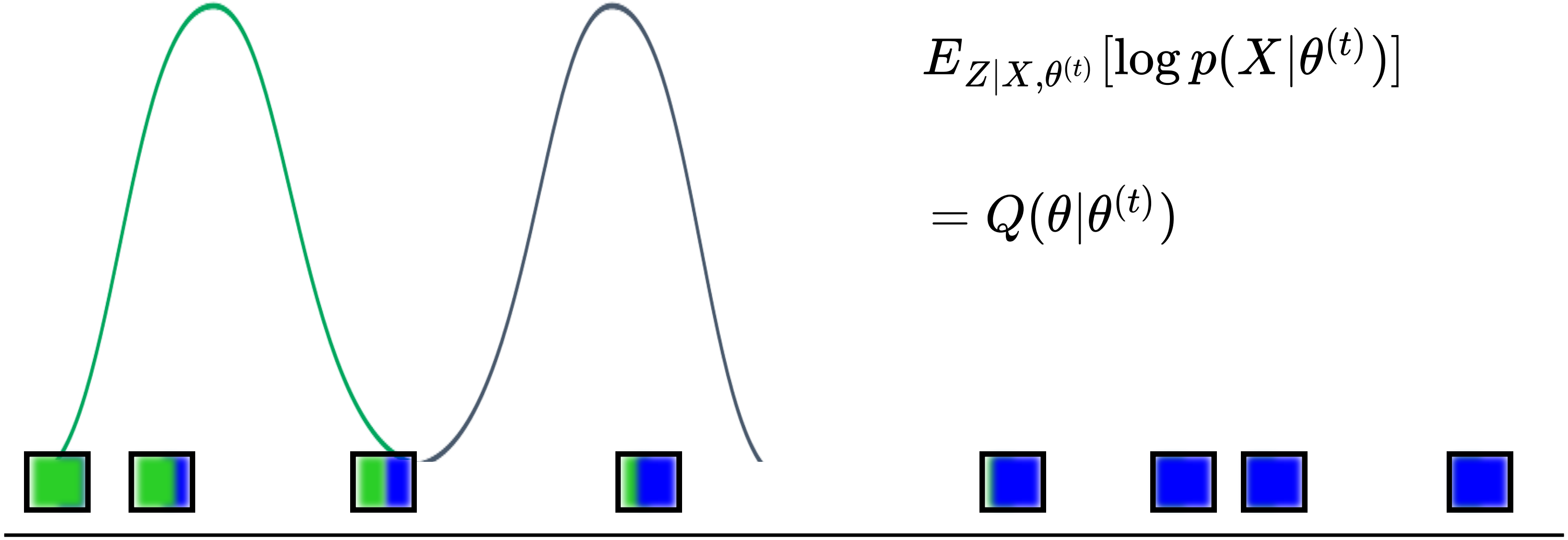
$$p(X|\theta^{(t)})$$

unsupervised learning

Gaussian Mixture Model with Expected-Maximization (distribution-based method)

Expectation (E) step

find the probability for each point to be generated by each mixture



$$E_{Z|X,\theta^{(t)}}[\log p(X|\theta^{(t)})]$$

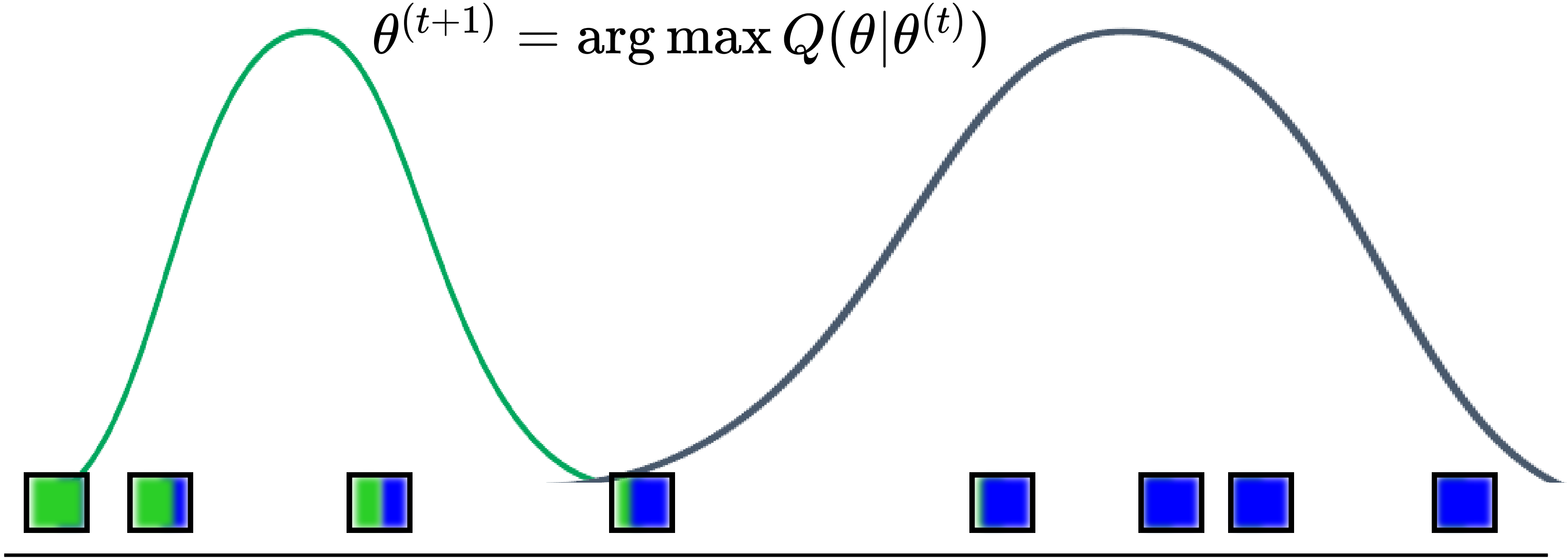
$$= Q(\theta|\theta^{(t)})$$

Gaussian Mixture Model with Expected-Maximization (distribution-based method)

maximization (M) step:

fit the mixture to the samples

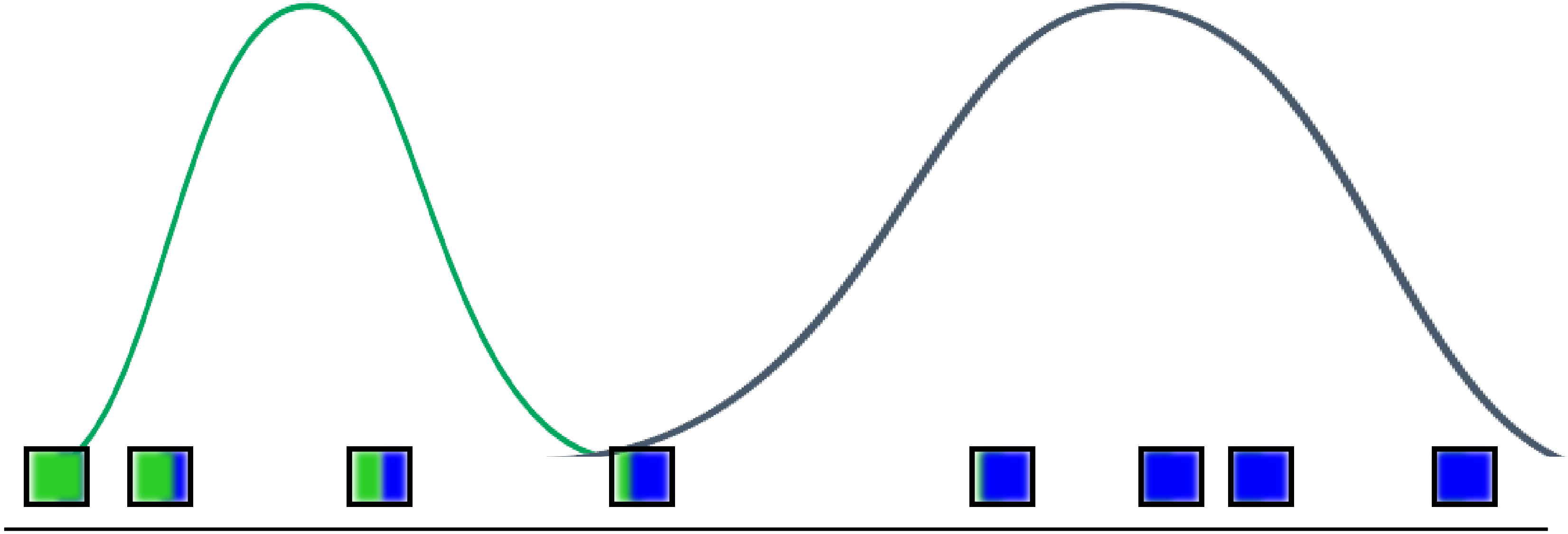
$$\theta^{(t+1)} = \arg\max Q(\theta|\theta^{(t)})$$



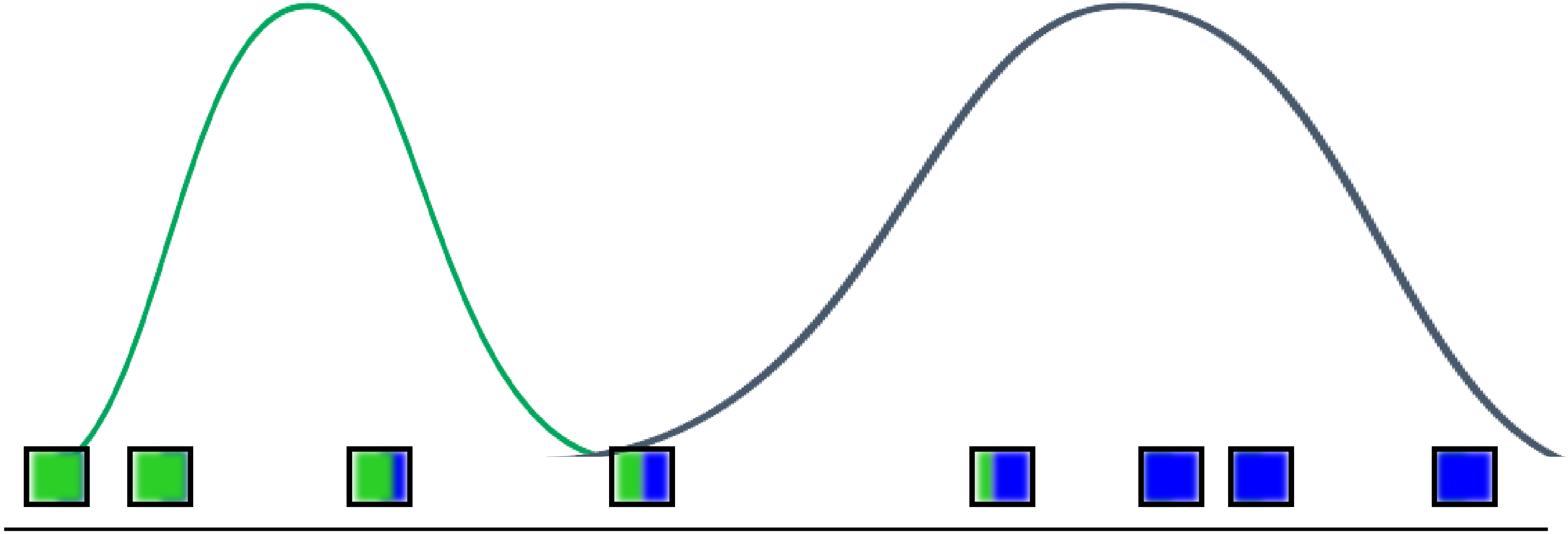
Gaussian Mixture Model with Expected-Maximization (distribution-based method)

ready for a new E step ?

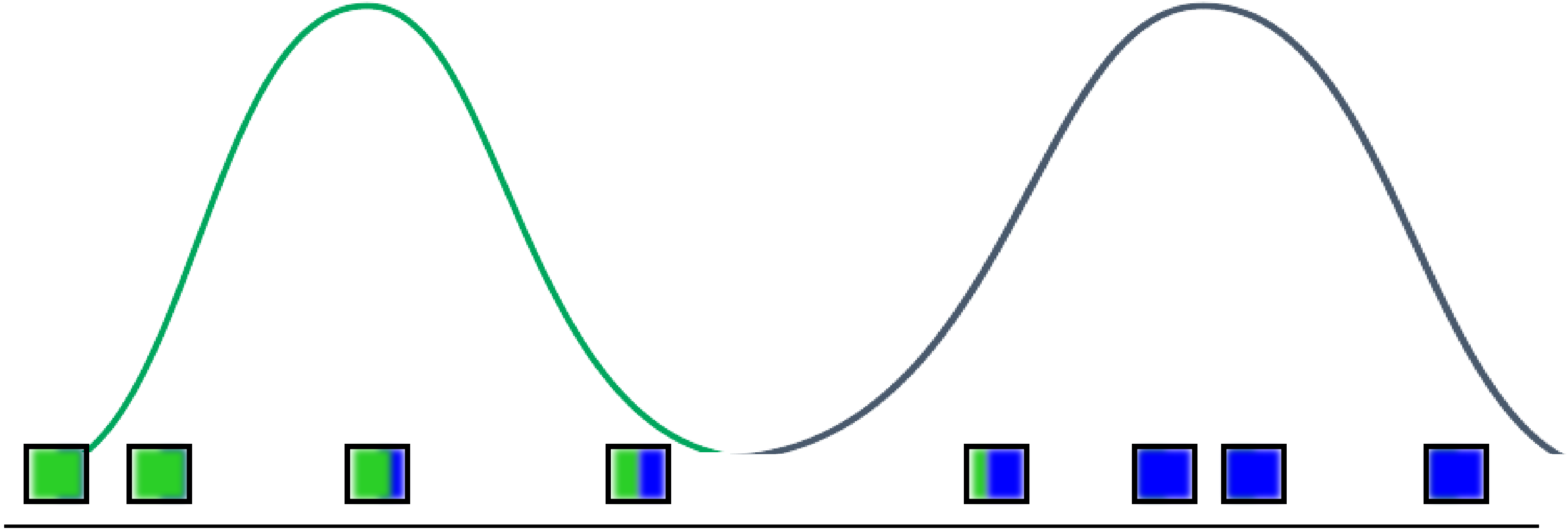
check the colors in the squares...



Gaussian Mixture Model with Expected-Maximization
(distribution-based method)

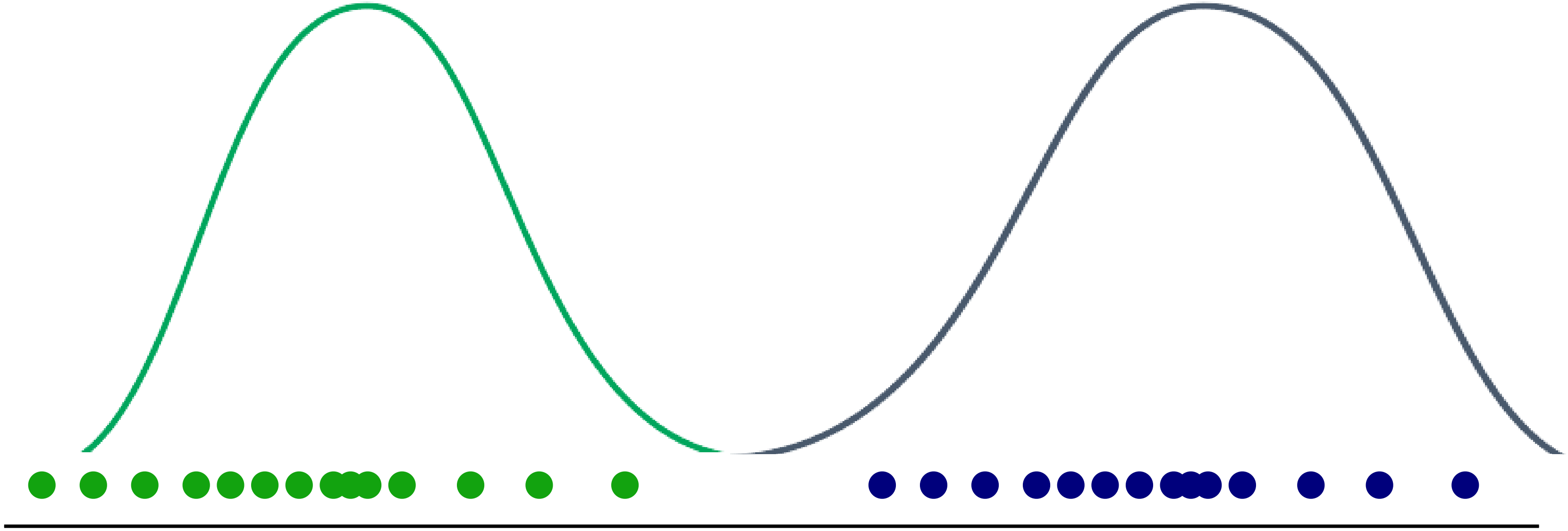


Gaussian Mixture Model with Expected-Maximization
(distribution-based method)



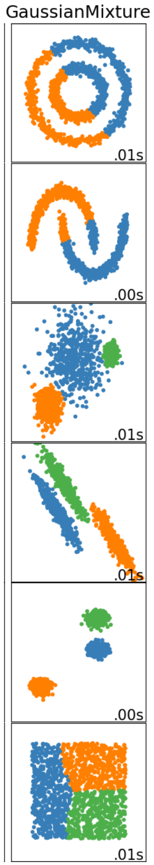
Gaussian Mixture Model with Expected-Maximization (distribution-based method)

no more move ? assign the labels => clusters
or keep the multiple labels ...



clustering

Gaussian Mixture Model with Expected-Maximization
(distribution-based method)



+ not restricted to circular clusters... possibly ellipses !
+ support mixed membership labeling
+ you can generate new samples (probabilistic model)

- need to fix the number of Gaussians (expected number of clusters) as in k-means

clustering

DBSCAN (density-based method)

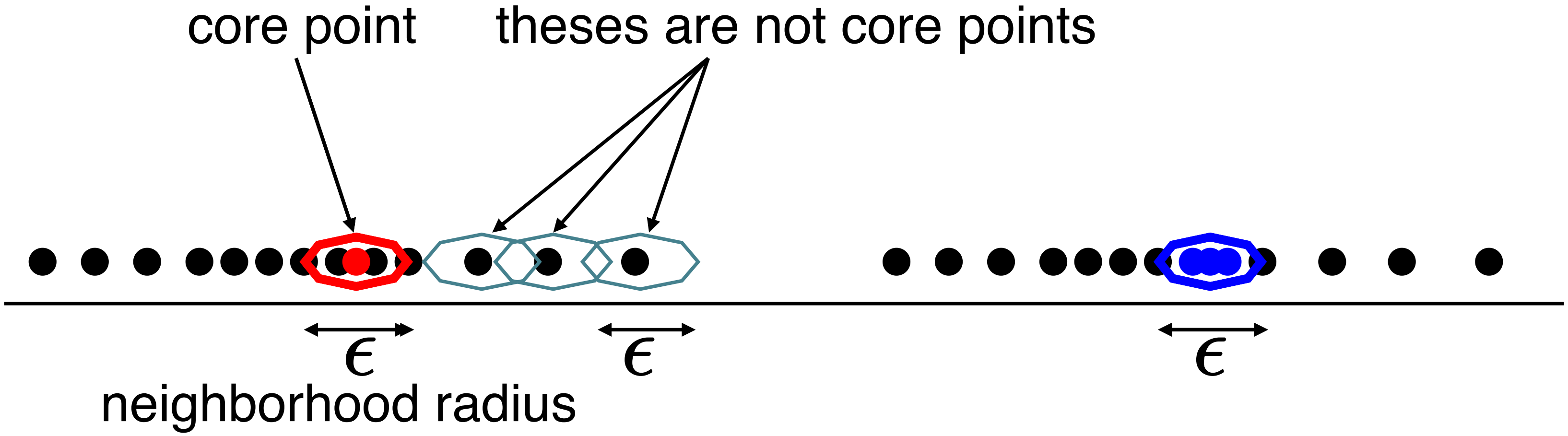
All points within the cluster are mutually density-connected

If a point is "density-reachable" from some point of the cluster, it is also part of the cluster

ϵ : neighborhood radius
minPts: minimum number of neighbors to be a **core point**

DBSCAN (density-based method)

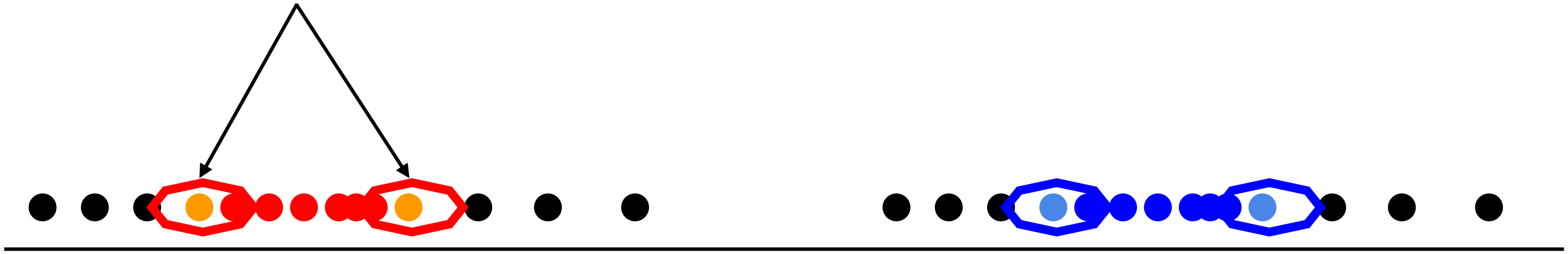
minPts = 2



DBSCAN (density-based method)

minPts = 2

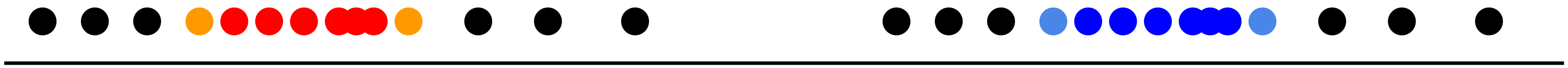
not core points but reachable!



DBSCAN (density-based method)

minPts = 2

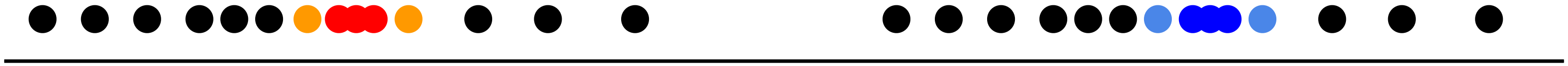
the rest is "noise"



DBSCAN (density-based method)

minPts = 2

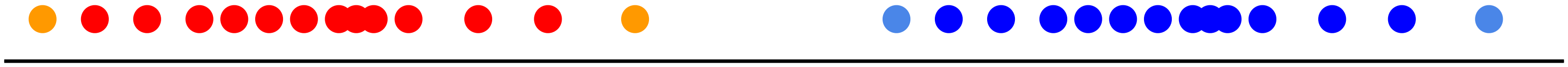
different results with **smaller** epsilon ...



DBSCAN (density-based method)

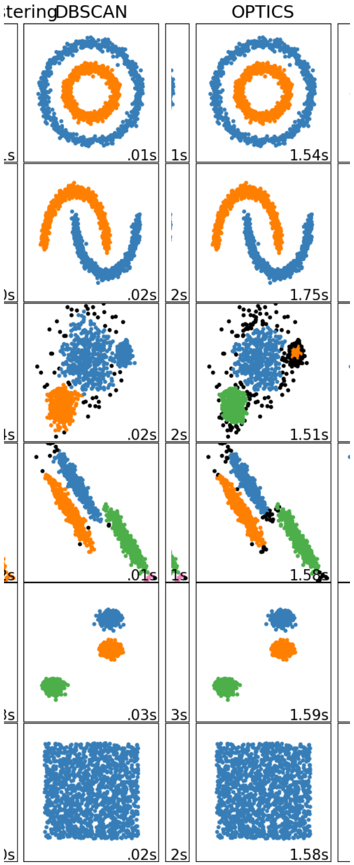
minPts = 2

different results with **greater** epsilon ...



clustering

DBSCAN (density-based method)



+ Does not assume any predefined shape on data clusters

- data defined by set of coordinates (not capable of handling arbitrary feature spaces)
- computationally costly... (...)
- not robust to clusters of varying density

=> OPTICS (density-based method)

clustering

Performance Metrics ?

Silhouette coefficient

Calinski-Harabaz index

Davies-Bouldin Index

Rand index

Mutual Information based scores

Homogeneity, completeness and V-measure

Fowlkes-Mallows scores

Contingency Matrix

Pair Confusion Matrix

clustering

Silhouette coefficient (between -1 and 1)

for each sample

the higher its value, the more similar the sample is within its cluster (and not to neighboring clusters).

If most samples have a low or negative value, then the clustering configuration is not appropriate.

$$\frac{b-a}{\max(a,b)}$$

with a the mean distance between a sample and all other points in the same cluster

with b the mean distance between a sample and all other points in the next nearest cluster

clustering

Performance Metrics ?

Calinski-Harabaz index

The higher the Calinski-Harabaz index $s(k)$ the more dense and well separated the $k\text{-}th$cluster is.

$$\frac{\mathrm{Tr}(B_k)}{\mathrm{Tr}(W_k)}\frac{N-k}{k-1}$$

with $B_k$ the inter-cluster dispersion matrix

and W_k the intra-cluster dispersion matrix

unsupervised learning

unsupervised learning

Dimension reduction

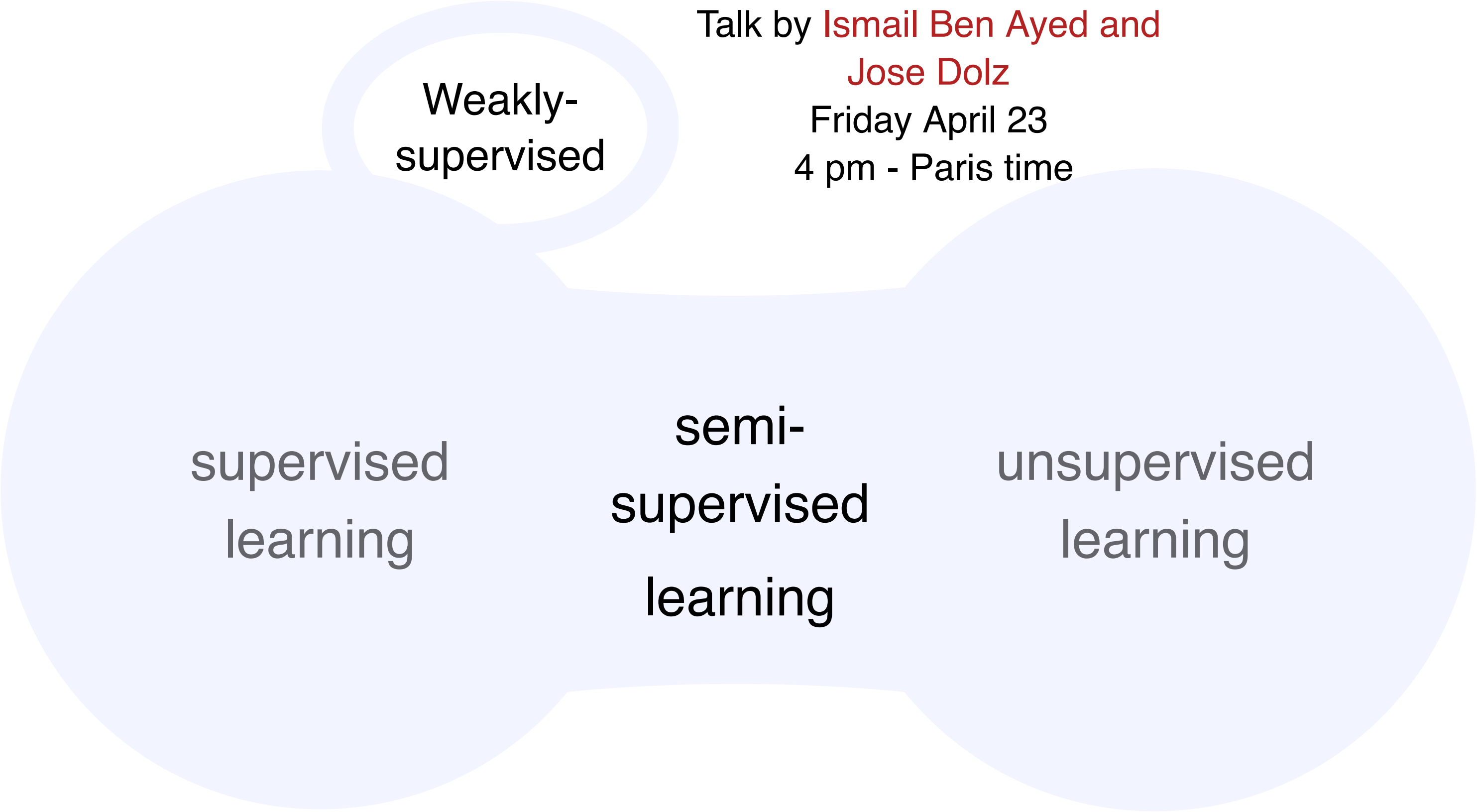
Clustering

Conclusion

Conclusion

- A machine learning expert must make assumptions on the data distribution and the task
- Metrics should be chosen in relation with the application
- Issues specific to medical imaging should be addressed
  - Imbalanced dataset
  - Annotation scarcity
  - High dimensionality

Conclusion



Weakly-supervised

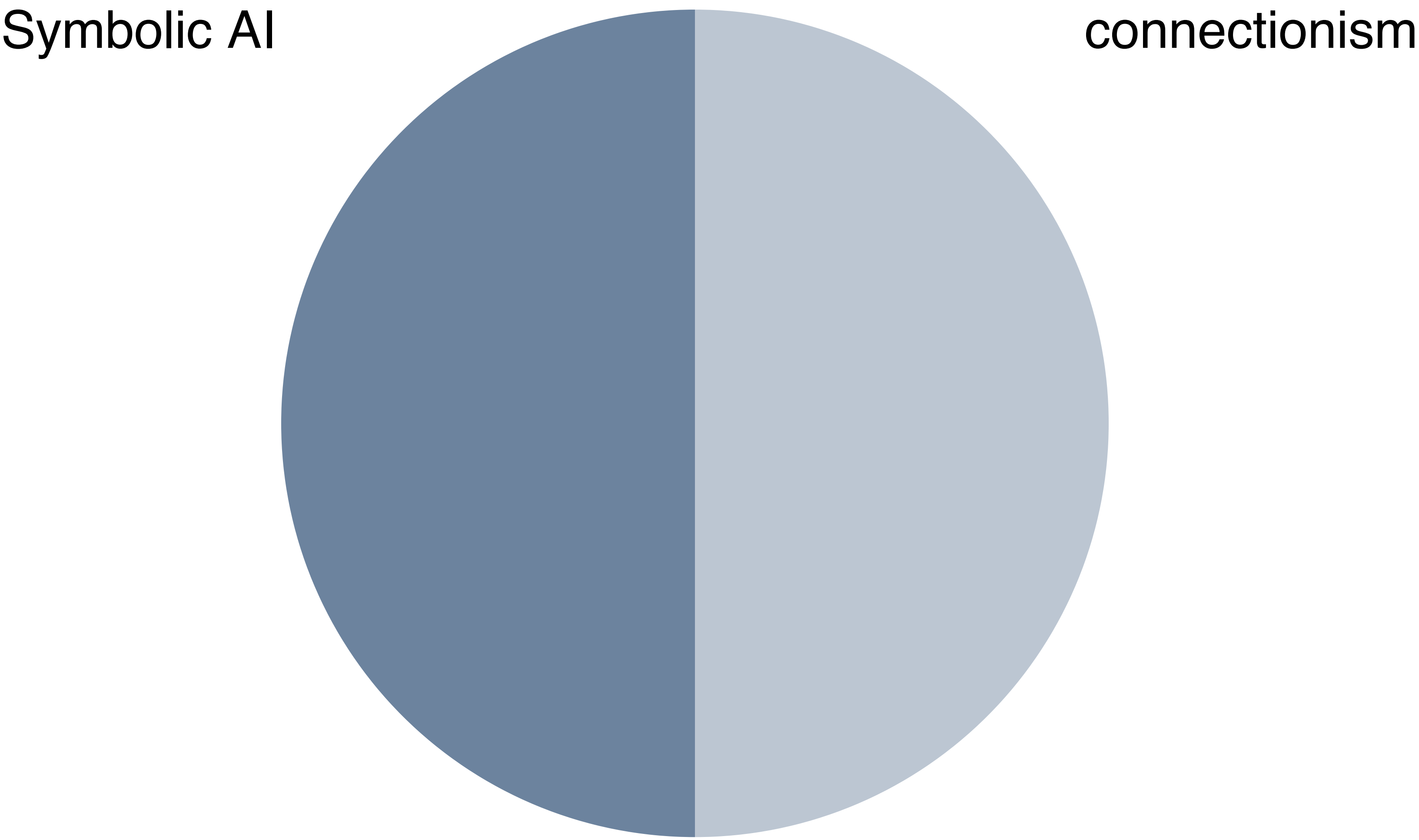
Talk by Ismail Ben Ayed and Jose Dolz
Friday April 23
4 pm - Paris time

supervised learning

semi-supervised learning

unsupervised learning

Conclusion



Conclusion

#responsibleAI (biases, ethics)

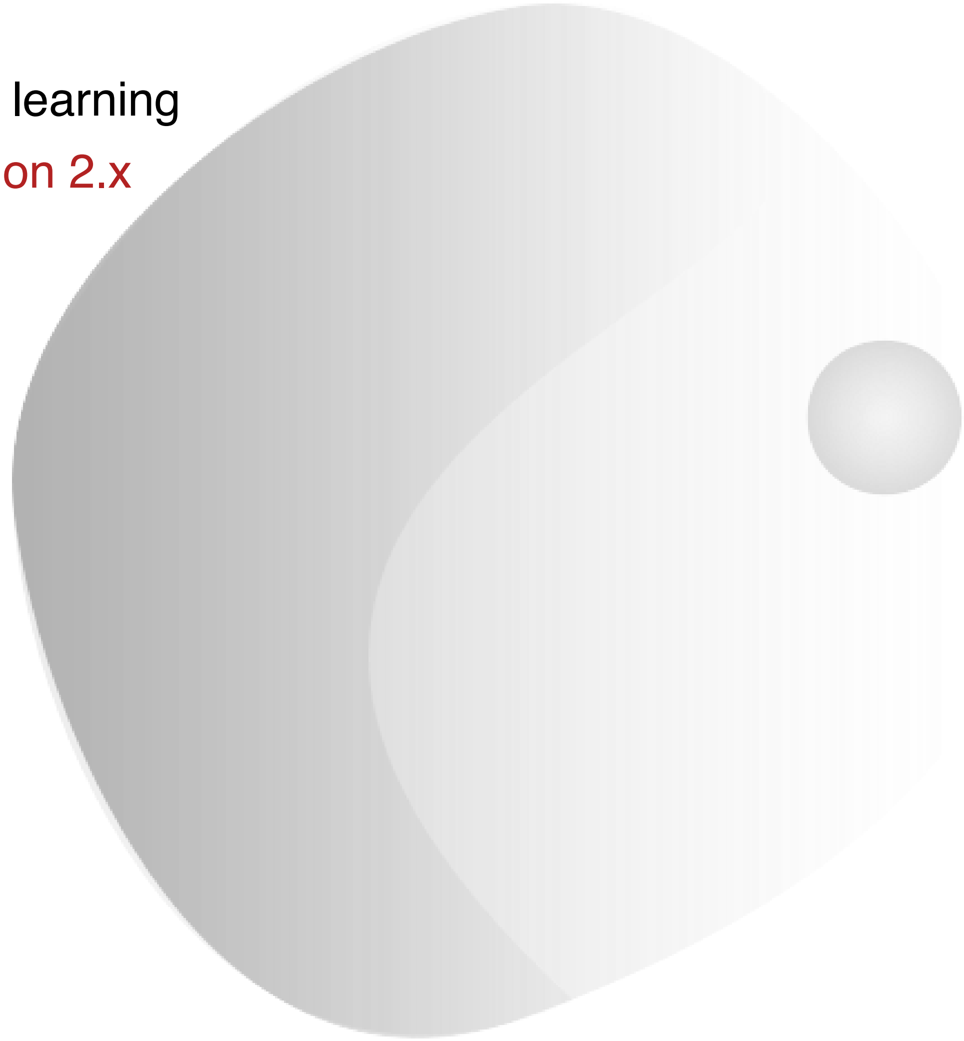
« a priori » within learning

Hands-on session 2.x

explanable AI (xAI)

Talk by Narine Kokhlikyan

Tuesday April 20
4.20 pm - Paris time



Biblio

- Book “Artificial Intelligence: A Modern approach” Russell Norvig
- Book “Understanding Machine Learning: From Theory to Algorithms” by Shai Shalev-Shwartz and Shai Ben-David
- Lecture notes ”Machine Learning” Central Supélec by Jérémy Fix, Hervé Frezza-Buet, Matthieu Geist, Frédéric Pennerath
- Lectures “Machine Learning for Intelligent Systems”, Cornell University by Kilian Weinberger Youtube link
- Model evaluation and selection https://arxiv.org/abs/1811.12808

Biblio

deeplearningbook.org `Goodfellow-et-al-2016`

Cardon, D., Cointet, J. P., & Mazières, A. (2018). La revanche des neurones: L'invention des machines inductives et la controverse de l'intelligence artificielle. *Réseaux*, *211*(5), 173-220.

Shervine Amidi (lecture notes)
https://stanford.edu/~shervine/teaching/cs-229

Sebastian Raschka (lecture notes)
https://github.com/rasbt/stat453-deep-learning-ss20/blob/master/L01-intro/L01-intro_slides.pdf

Nando de Freitas (lecture notes)
https://www.cs.ubc.ca/~nando/540-2013/lectures/l1.pdf

Stephane Canu (lecture notes)
http://asi.insa-rouen.fr/enseignants/~scanu/

https://scikit-learn.org/

https://en.wikipedia.org/