Music Machine Learning

X – Data complexity

Master ATIAM - Informatique

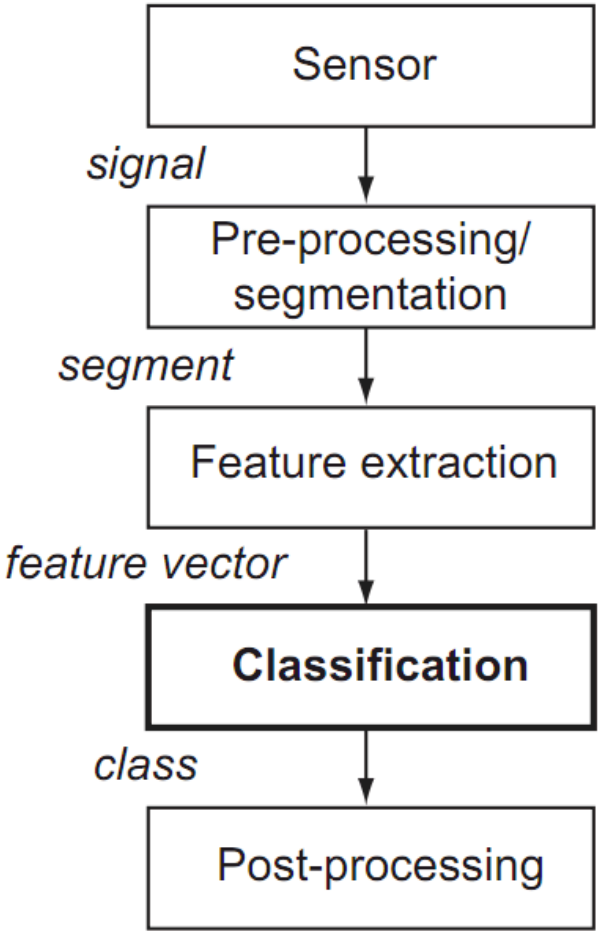
Philippe Esling (esling@ircam.fr)

Maître de conférences – UPMC

Equipe représentations musicales (IRCAM, Paris)

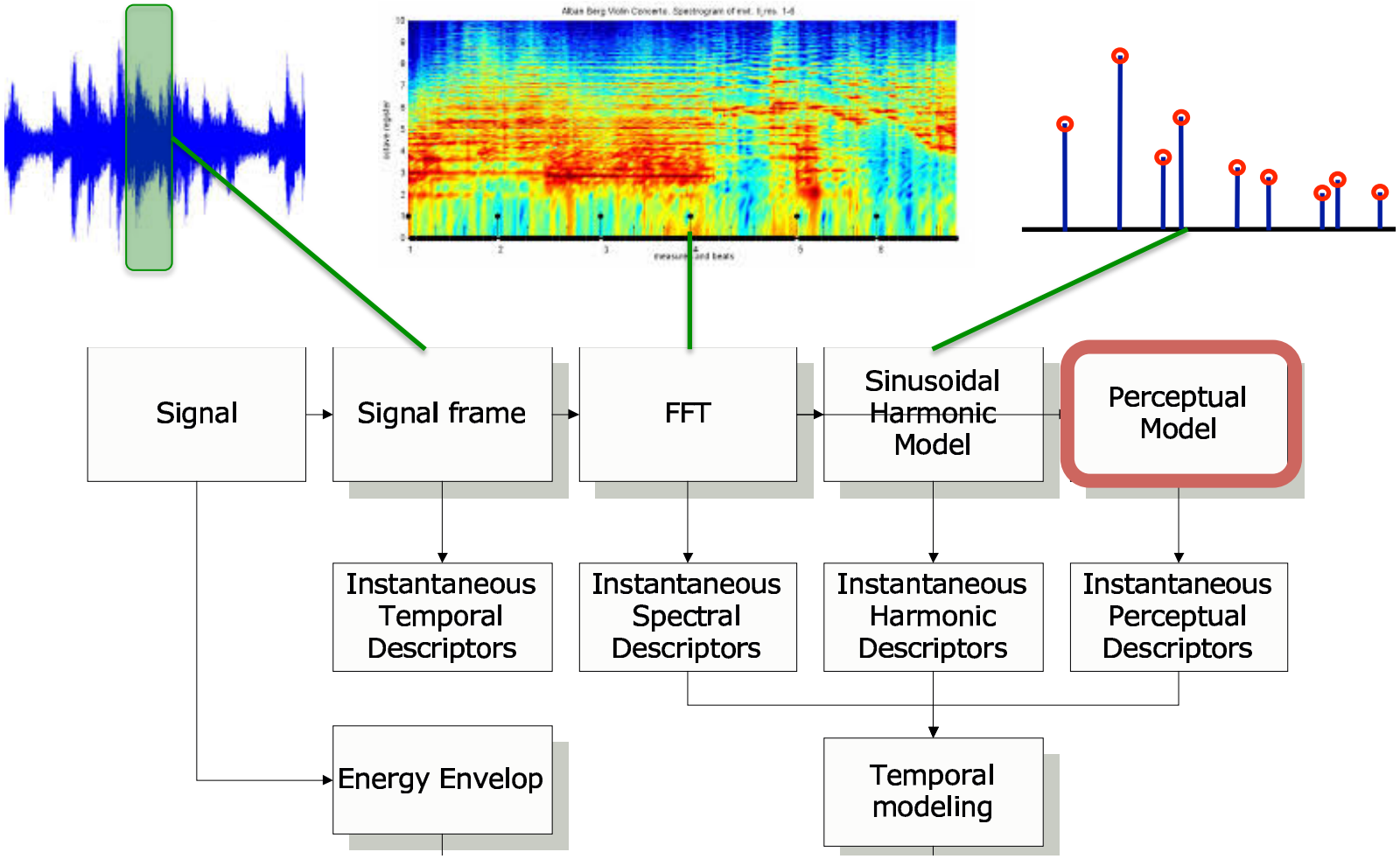
UPMC SORBONNE UNIVERSITÉS master ATIAM ircam Centre Pompidou

Any type of audio recognition



- Up to now we have seen classification
- Each working on a specific problem
- But on which features ?
- Years of research on audio features

Audio features



Retrieving musical information (MIR)

1. **Harmonic analysis**
    1. Pitch tracking, melody estimation
    2. Multiple F0 estimation, chord estimation
2. **Rhythmic analysis**
    1. Onset detection
    2. Tempo estimation, beat tracking
3. **Content-based analysis**
    1. Instrument recognition
    2. Structure analysis
    3. Classification (genre recognition, tags, mood, …)
    4. Music similarity and cover detection
    5. Query by content (query by humming)

Known issues of machine learning

1. Overfitting / Overtraining
(**function approximation**)
1. Curse of dimensionnality
(**dimensionnality reduction**)
1. Cross-validation
(**generalization probability**)
1. Cherry picking
(**datasets discrepancies**)
1. The *No Free Lunch* theorem
(**overall accuracy**)

Defining Machine Learning

A machine learning algorithm is a combination of 3 elements:

(either explicitly specified or implicit)

✓ the choice of a specific function family: F
(often a parameterized family)

✓ a way to evaluate the quality of a function $f \in F$
(typically using a cost (or loss) function L
mesuring how wrongly f prédicts)

✓ a way to search for the «best» function $f \in F$
(typically an optimization of function parameters to minimize the overall loss over the training set).

Defining Machine Learning

Evaluating a predictor $f(x)$

The performance of a predictor is often evaluated using **several** different evaluation metrics:

- Evaluations of true quantities of interest ($ saved, #lifes saved, ...) when using predictor inside a more complicated system.
- «Standard» evaluation metrics in a specific field (e.g. BLEU (Bilingual Evaluation Understudy) scores in translation)
- Misclassification error rate for a classifier (or precision and recall, or F-score, ...).
- The loss actually being optimized by the ML algorithm (often different from all the above...)

Defining Machine Learning

Expected risk v.s. Empirical risk

Examples (x,y) are supposed drawn i.i.d. from an unknown true distribution $p(\mathrm{x},\mathrm{y})$ (from nature or industrial process)

- Generalization error = Expected risk (or just «Risk»)
  «how poorly we will do on average on the infinity of future examples from that unknown distribution»

$$R(f) = \mathbb{E}_{p(\mathbf{x},\mathbf{y})}[L(f(\mathbf{x}), \mathbf{y})]$$

- Empirical risk = average loss on a finite dataset
  «how poorly we're doing on average on this finite dataset»

$$\hat{R}(f, D) = \frac{1}{|D|} \sum_{(\mathbf{x},\mathbf{y}) \in D} L(f(\mathbf{x}), \mathbf{y})$$

where $|D|$ is the number of examples in D

Defining Machine Learning

Evaluating the generalization error

- We can't compute expected risk $R(f)$
- But $\hat{R}(f, D)$ is a good estimate of $R(f)$ provided:
  - D was not used to find/choose f
    otherwise estimate is biased ⇨ can't be the training set!
  - D is large enough (otherwise estimate is too noisy); drawn from p

➡ Must keep a separate test-set $D_{\text{test}} \neq D_{\text{train}}$ to properly estimate generalization error of $\hat{f}(D_{\text{train}})$:

$$R(\hat{f}(D_{\text{train}})) \approx \hat{R}(\hat{f}(D_{\text{train}}), D_{\text{test}})$$

generalization error — average error on **test**-set (never used for training)

This is the test phase in ML

Defining Machine Learning

Model selection

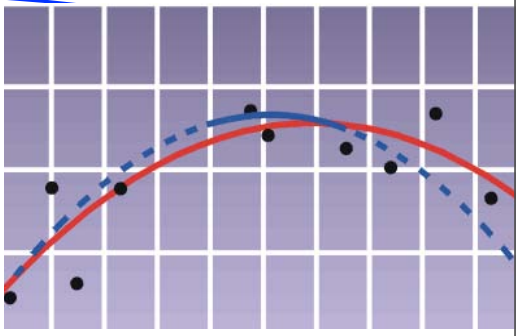
Choosing a specific function family F

Defining Machine Learning

$F_{polynomial\ p}$

Polynomial predictor (of degree p):

$$f(x) = b + a_1 x + a_2 x^2 + a_3 x^3 + \ldots + a_p x^p$$



F_{linear}

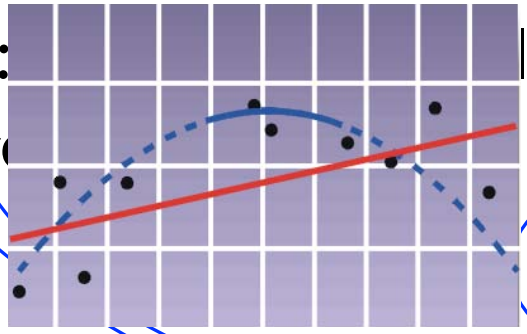
Linear (affine) predictor: $f_\theta(x) = wx + b$ (in 1 dimension)

(«linear regression») $f_\theta(x) = w^T x + b$ (in d dimensions)

$$\theta = \{w \in \mathbb{R}^d, b \in \mathbb{R}\}$$

Q: [illegible]lest

pr[illegible]



F_{const}

Constant predictor: $f_\theta(x)=b$

where $\theta=\{b\}$

(always predict the same value or class!)

Defining Machine Learning

Capacity of a learning algorithm

- Choosing a specific Machine Learning algorithm means choosing a specific function family F.
- How «big, rich, flexible, expressive, complex» that family is, defines what is informally called the «capacity» of the ML algorithm.

  Ex: capacity($F_{polynomial\ 3}$) > capacity(F_{linear})
- One can come up with several formal measures of «capacity» for a function family / learning algorithm (e.g. VC-dimension Vapnik–Chervonenkis)
- One rule-of-thumb estimate, is the number of adaptable parameters: i.e. how many *scalar* values are contained in θ.

  Notable exception: chaining many linear mappings is still a linear mapping!

Defining Machine Learning

Effective capacity, and capacity-control hyper-parameters

The «effective» capacity of a ML algo is controlled by:

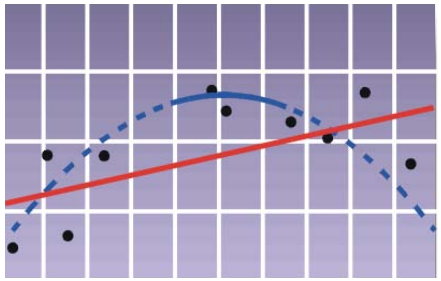
- Choice of ML algo, which determines big family F
- Hyper-parameters that further specify F
  e.g.: degree p of a polynomial predictor; Kernel choice in SVMs;
  #of layers and neurons in a neural network
- Hyper-parameters of «regularization» schemes
  e.g. constraint on the norm of the weights w
  (⇨ ridge-regression; L_2 weight decay in neural nets);
  Bayesian prior on parameters; noise injection (dropout); ...
- Hyper-parameters that control early-stopping of the iterative search/optimization procedure.
  (⇨ won't explore as far from the initial starting point)

Defining Machine Learning

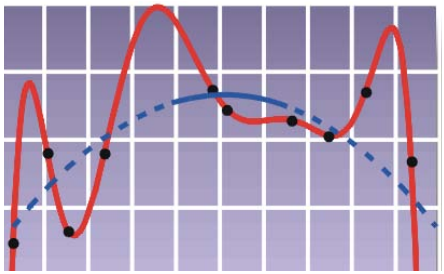
Tuning the capacity

- Capacity must be optimally tuned to ensure good generalization
- by choosing Algorithm and hyperparameters
- to avoid under-fitting and over-fitting.

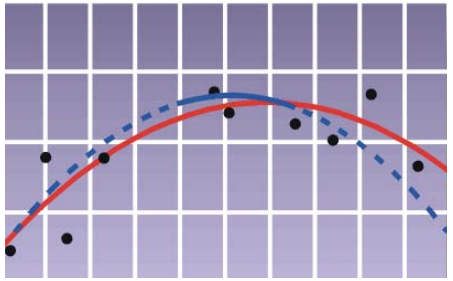
Ex: 1D regression with polynomial predictor



capacity too low
⇨under-fitting



capacity too high
⇨over-fitting



optimal capacity
⇨good generalisation

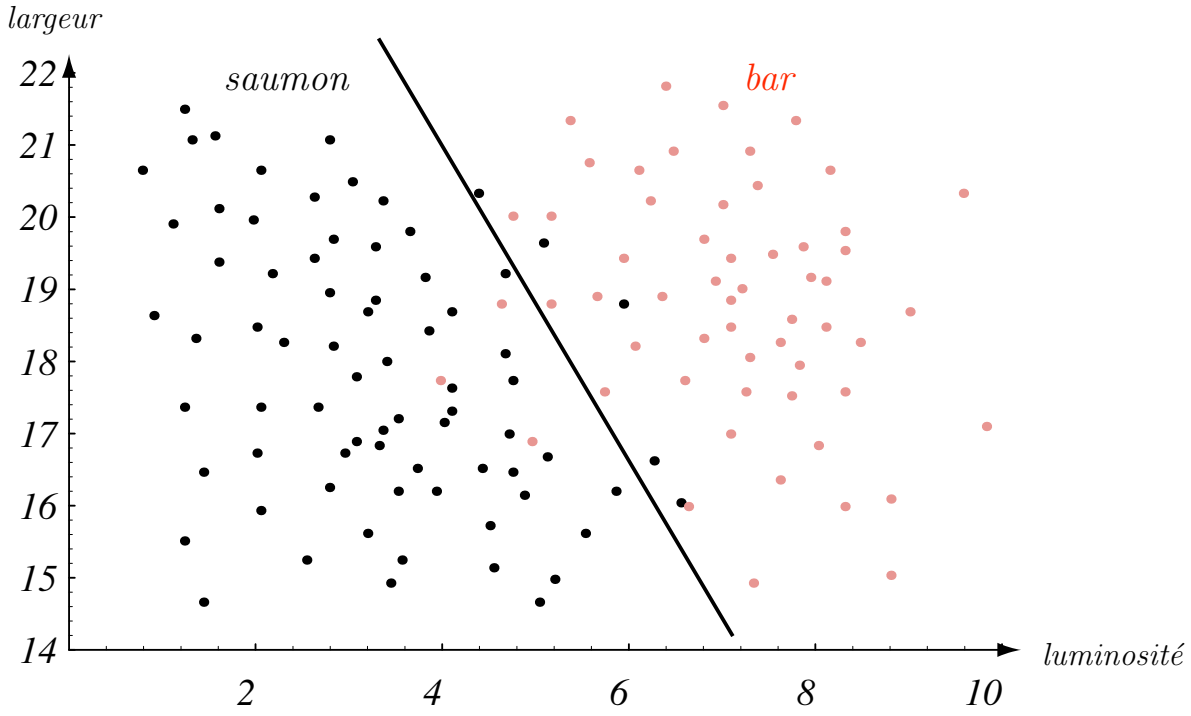
performance on training set is not a good estimate of generalization

Defining Machine Learning

Ex: 2D classification

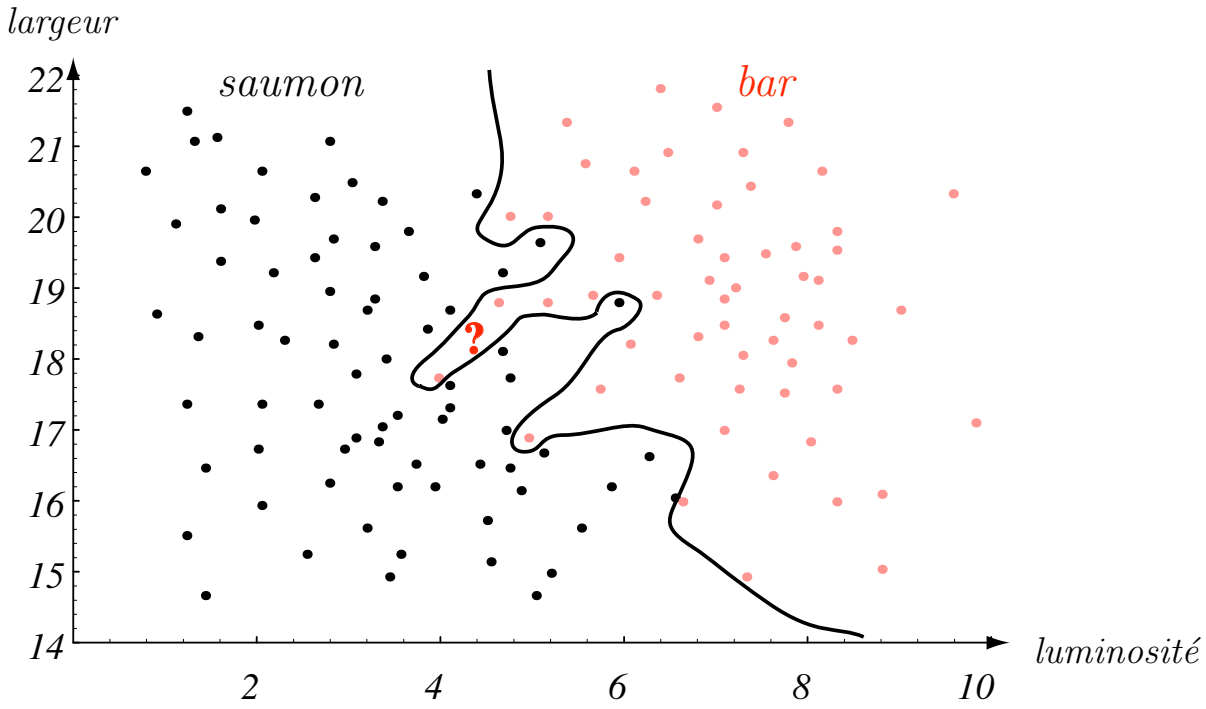
Linear classifier

- Function family too poor (too inflexible)
- = **Capacity too low** for this problem (relative to number of examples)
- => Under-fitting



Defining Machine Learning

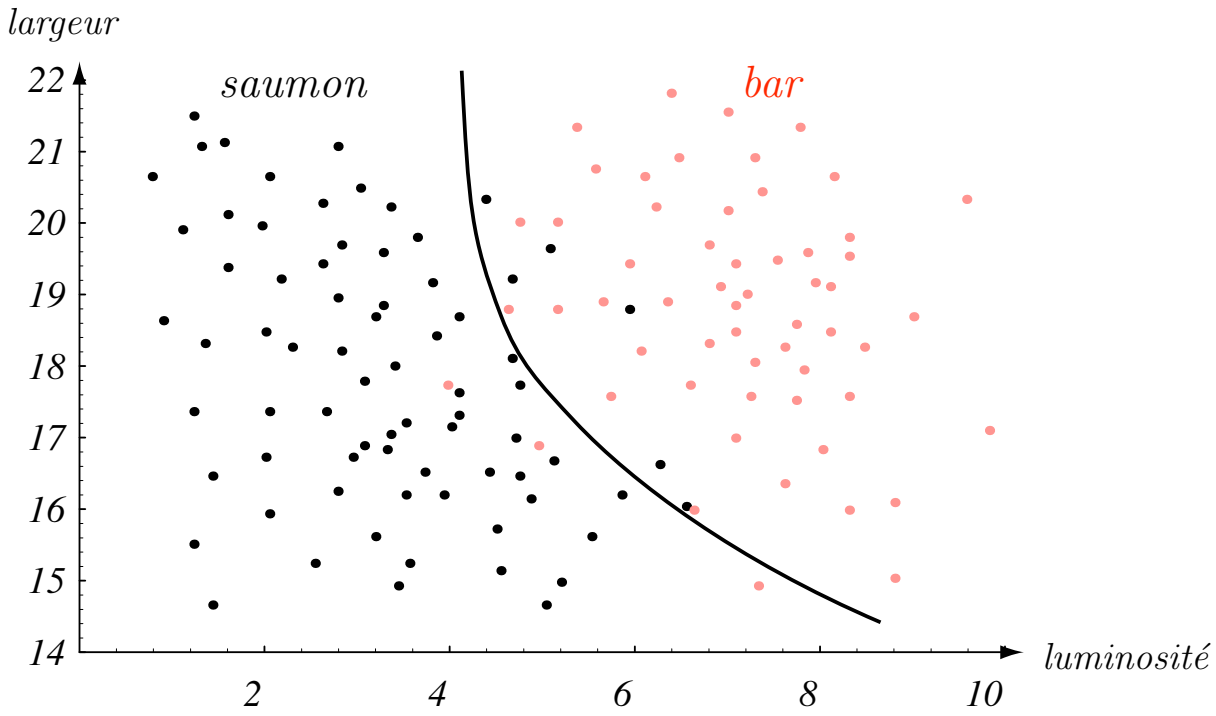
- Function family too rich
  (too flexible)
- = **Capacity too high** for this problem
  (relative to the number of examples)
- => Over-fitting



Nombre d'erreurs d'entrainement: 0

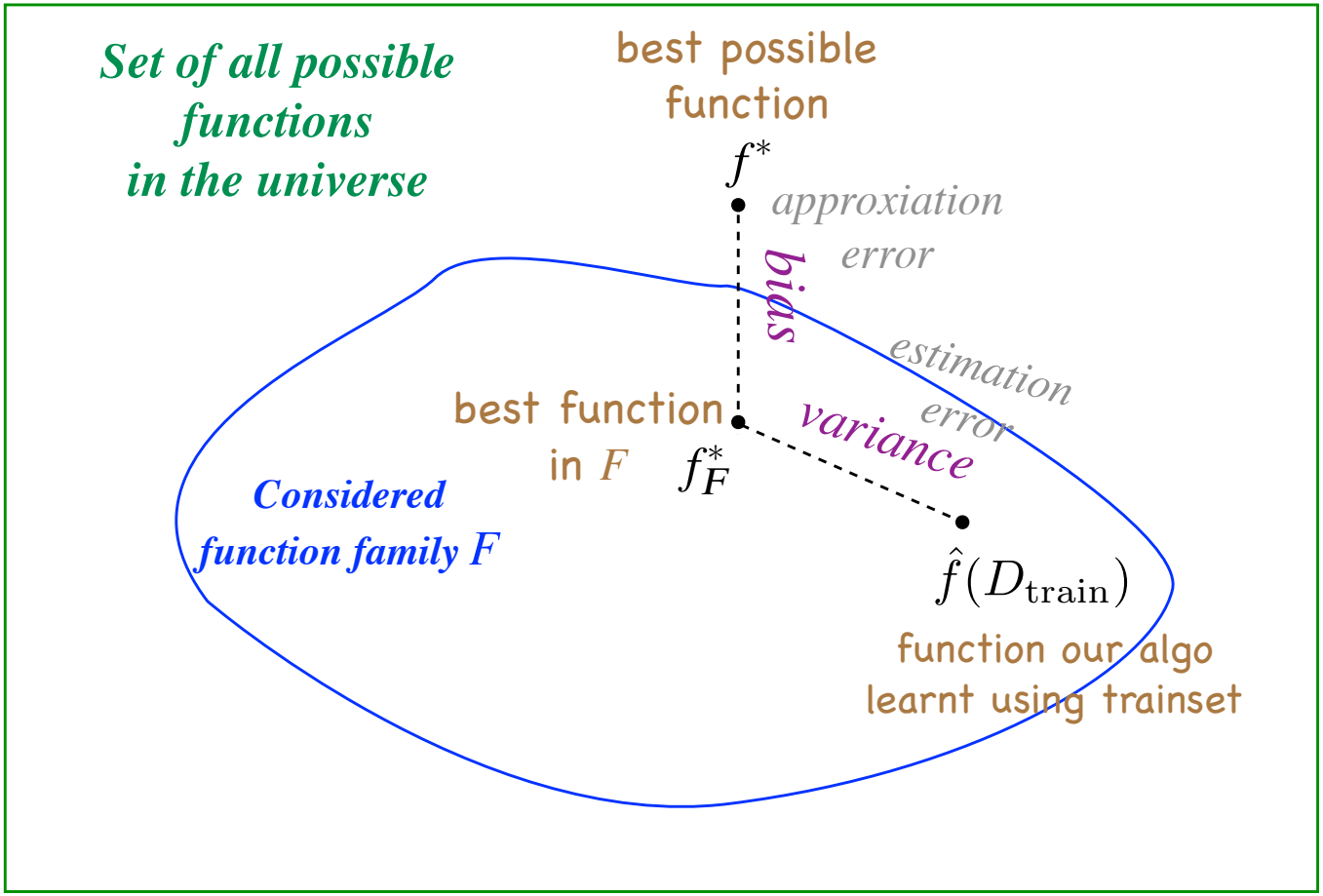
Defining Machine Learning

- **Optimal capacity** for this problem
  (par rapport à la quantité de données)
- => Best generalization
  (on future test points)



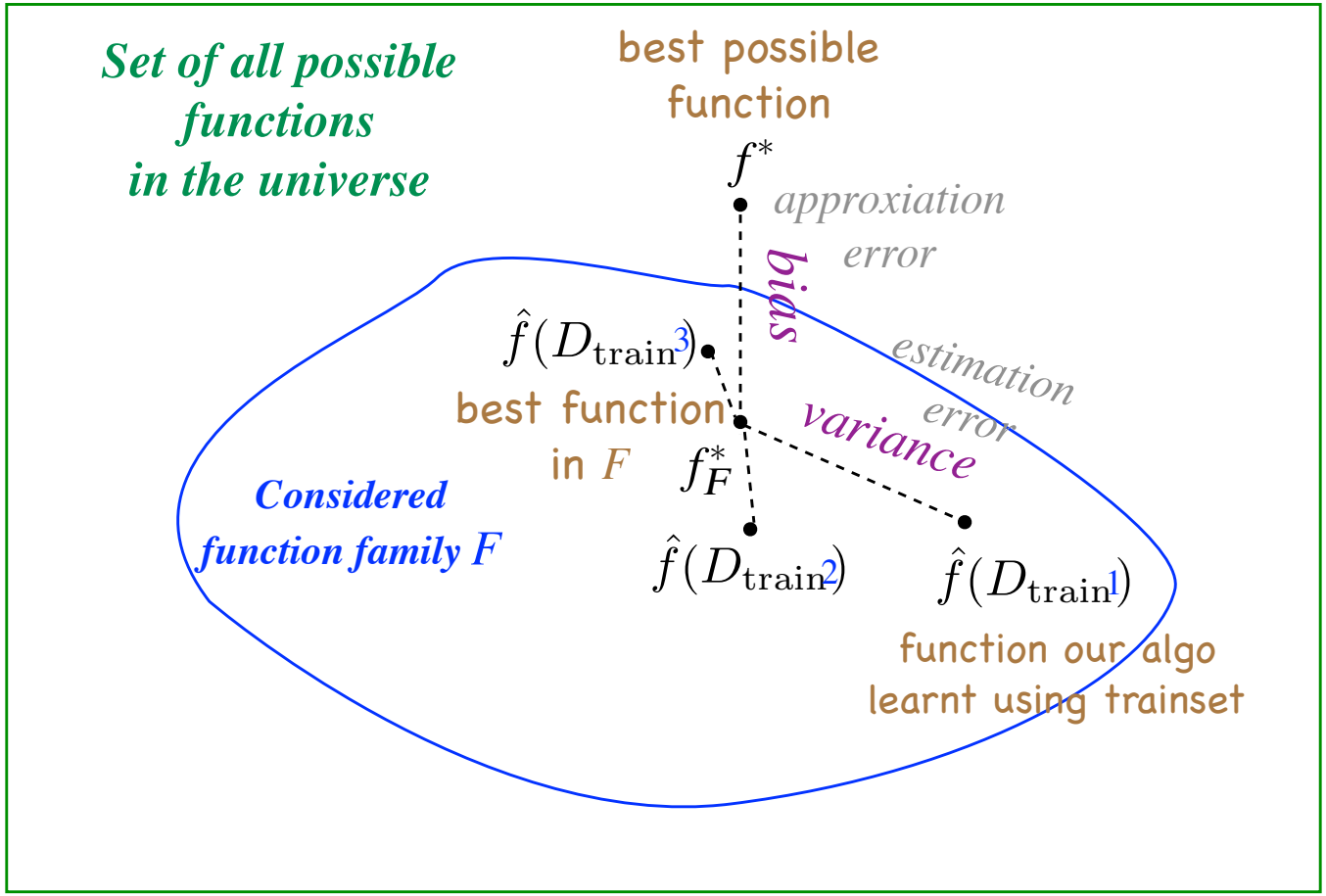
Defining Machine Learning

Decomposing the generalization error



Markov Chains

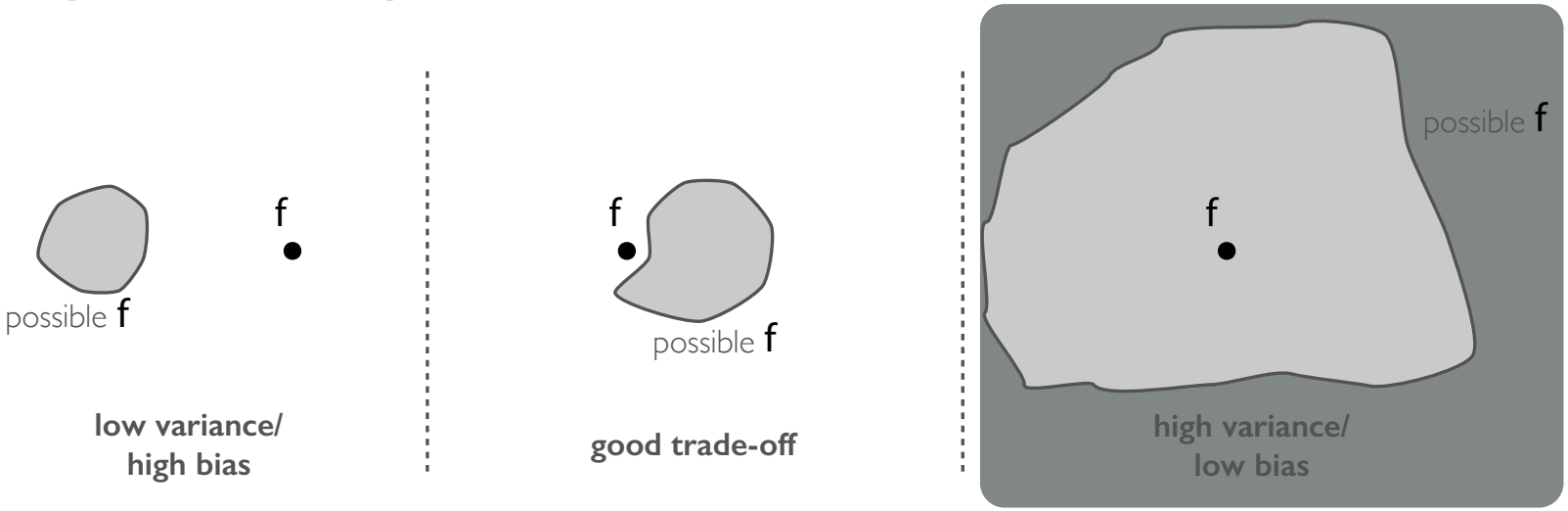
What is responsibe for the variance?



Choosing the right function

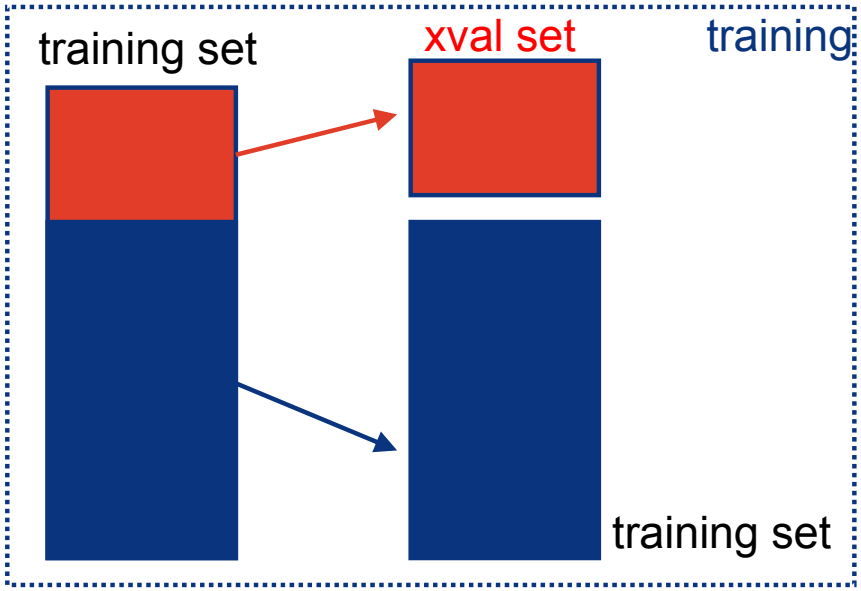
Topics: why training is hard

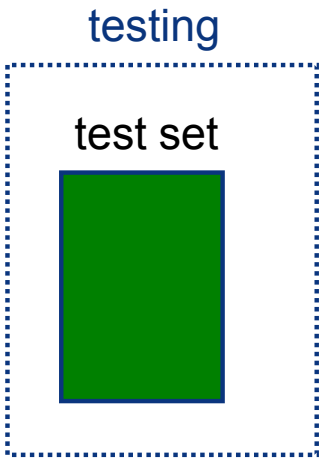
- Second hypothesis: overfitting
  - we are exploring a space of complex functions
  - deep nets usually have lots of parameters
- Might be in a high variance / low bias situation



Cross-validation

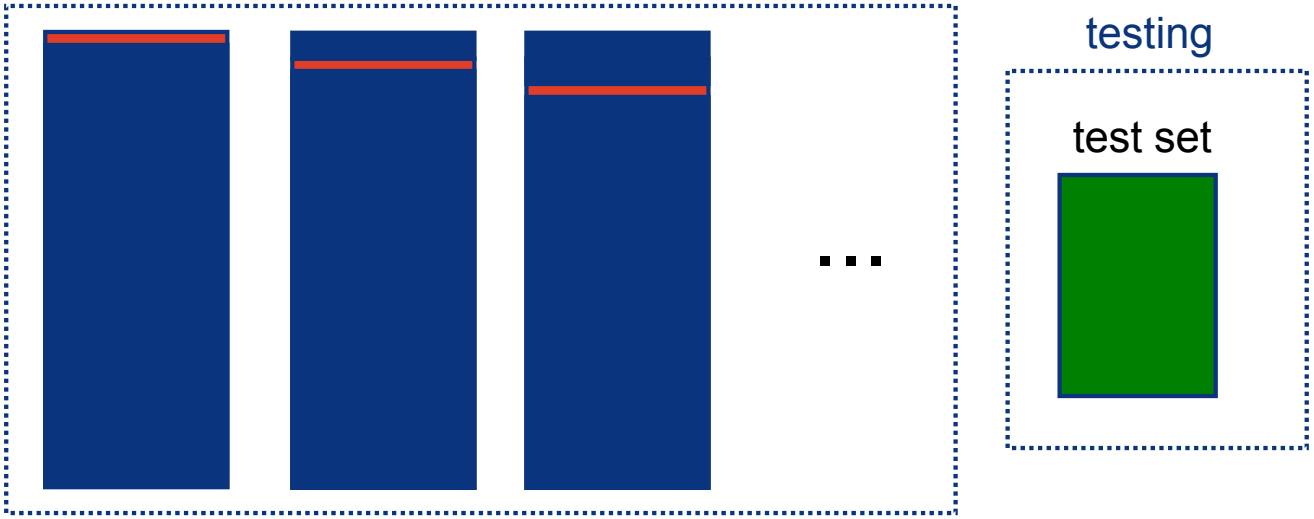
- basic idea:
  - leave some data out of your training set (cross validation set)
  - train with different parameters
  - evaluate performance on cross validation set
  - pick best parameter configuration





Cross-validation

- many variations
- leave-one-out CV:
  - compute n estimators of $P_X(x)$ by leaving one X_i out at a time
  - for each $P_X(x)$ evaluate $P_X(X_i)$ on the point that was left out
  - pick $P_X(x)$ that maximizes this likelihood




Training
Validation
0.5
0.4
0.3
0.2
0.1
0.0
underfitting
overfitting
number of epochs