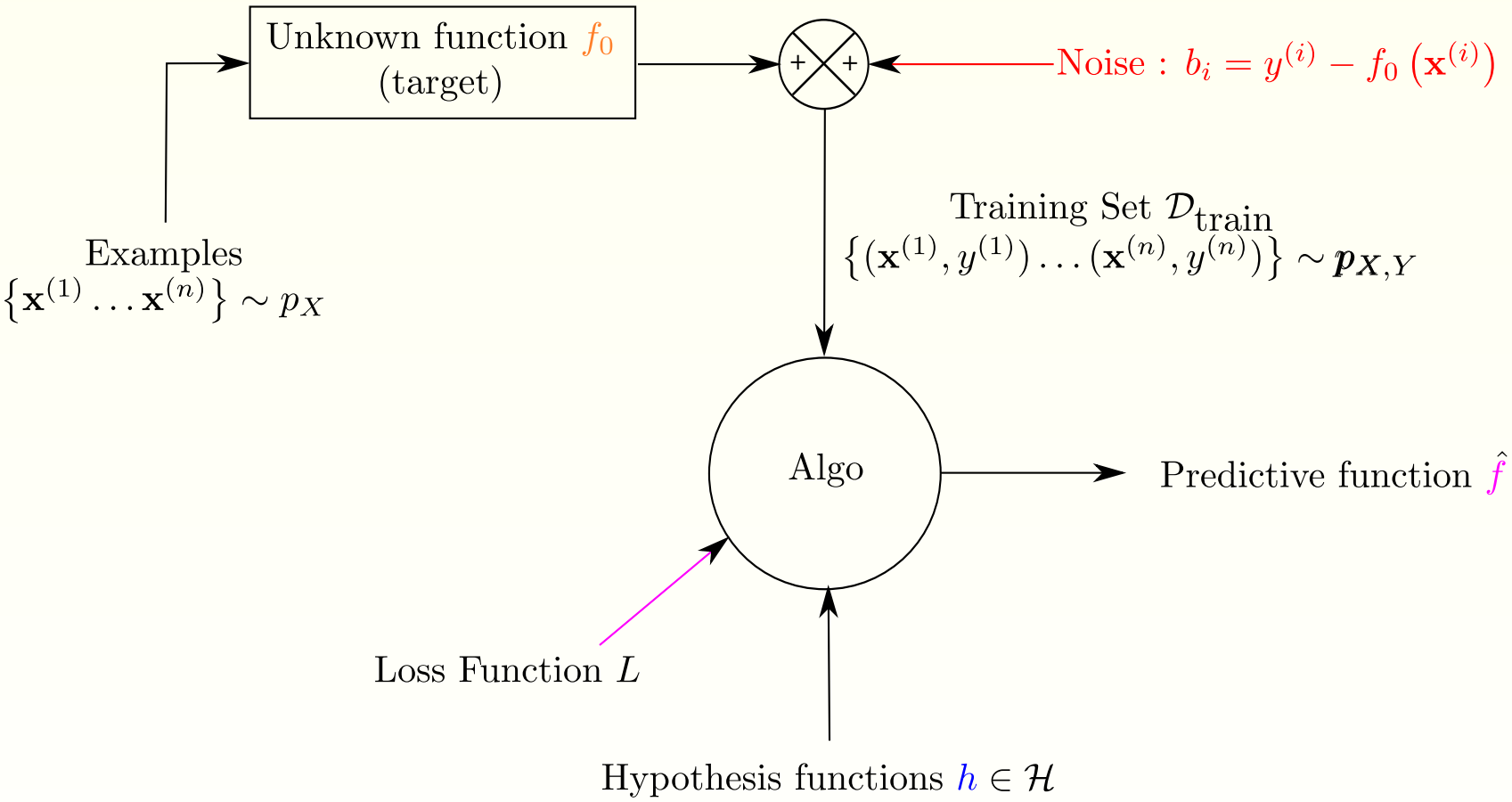
Ensemble Learning - 1
Linear Aggregation

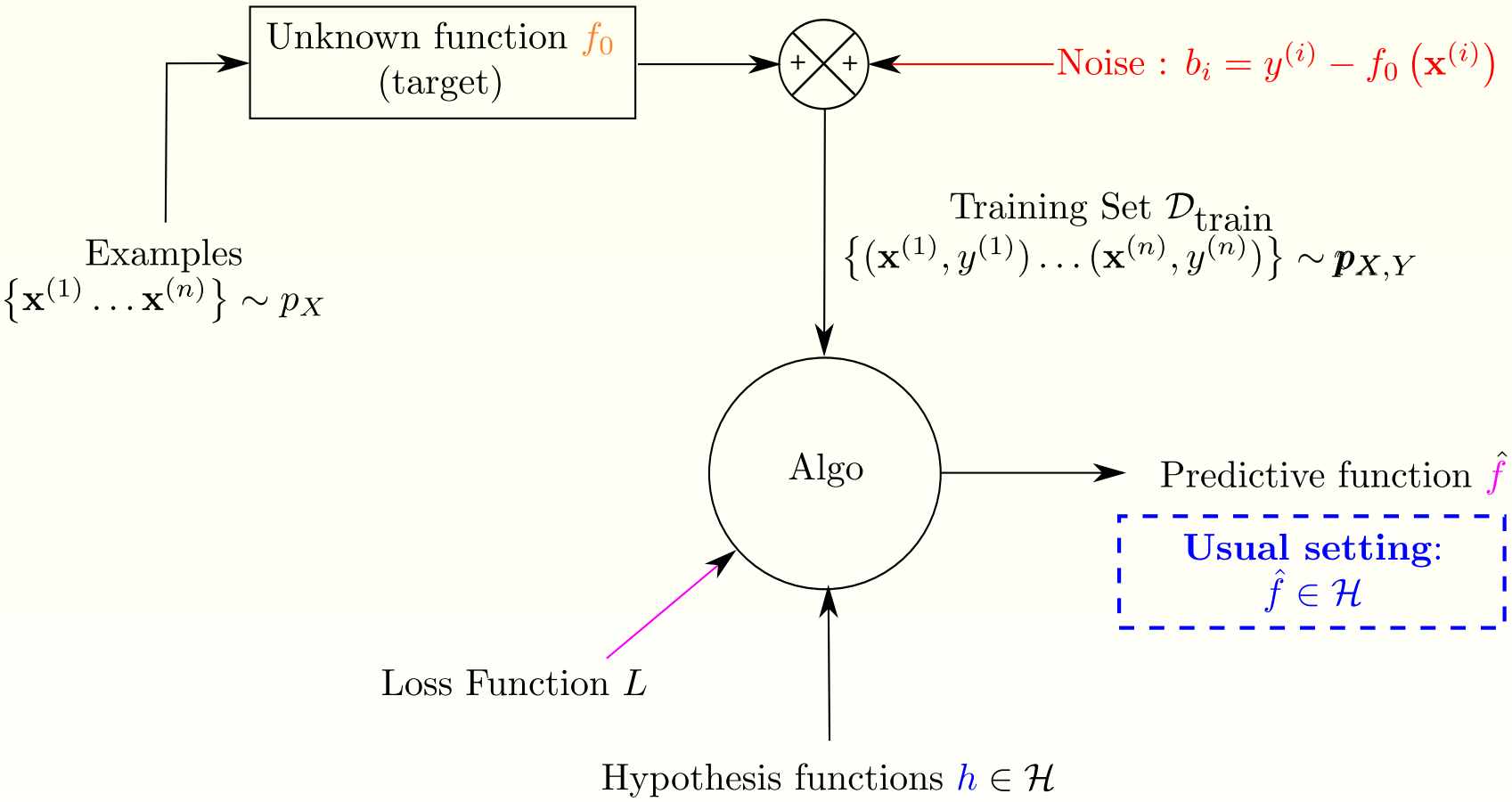
John Klein

centralelille

Supervised Learning : the learning diagram

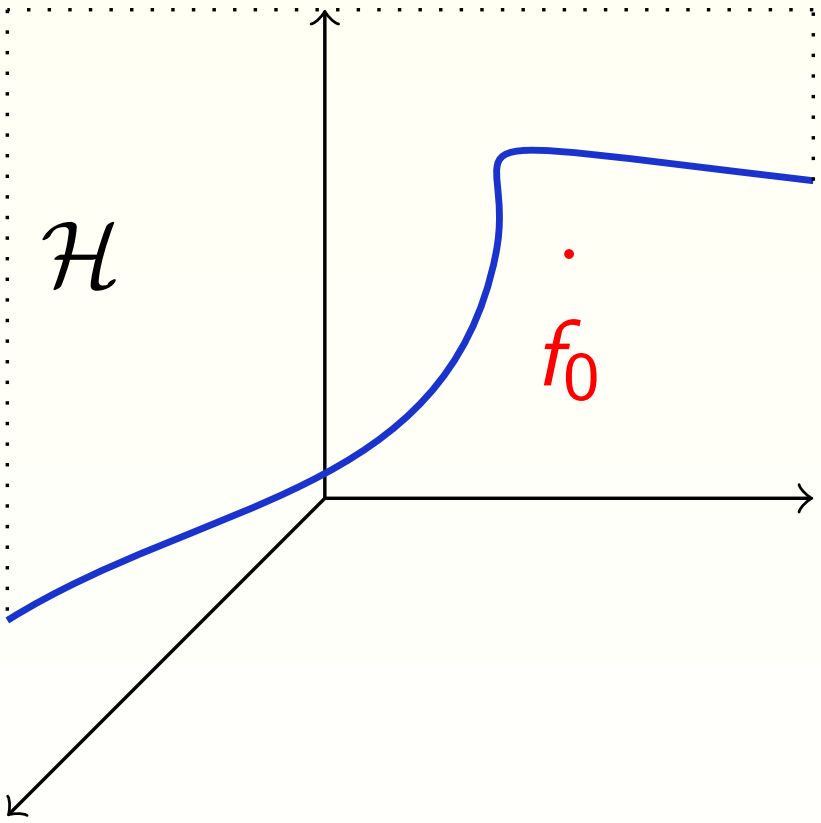


Ensemble Learning : the learning diagram

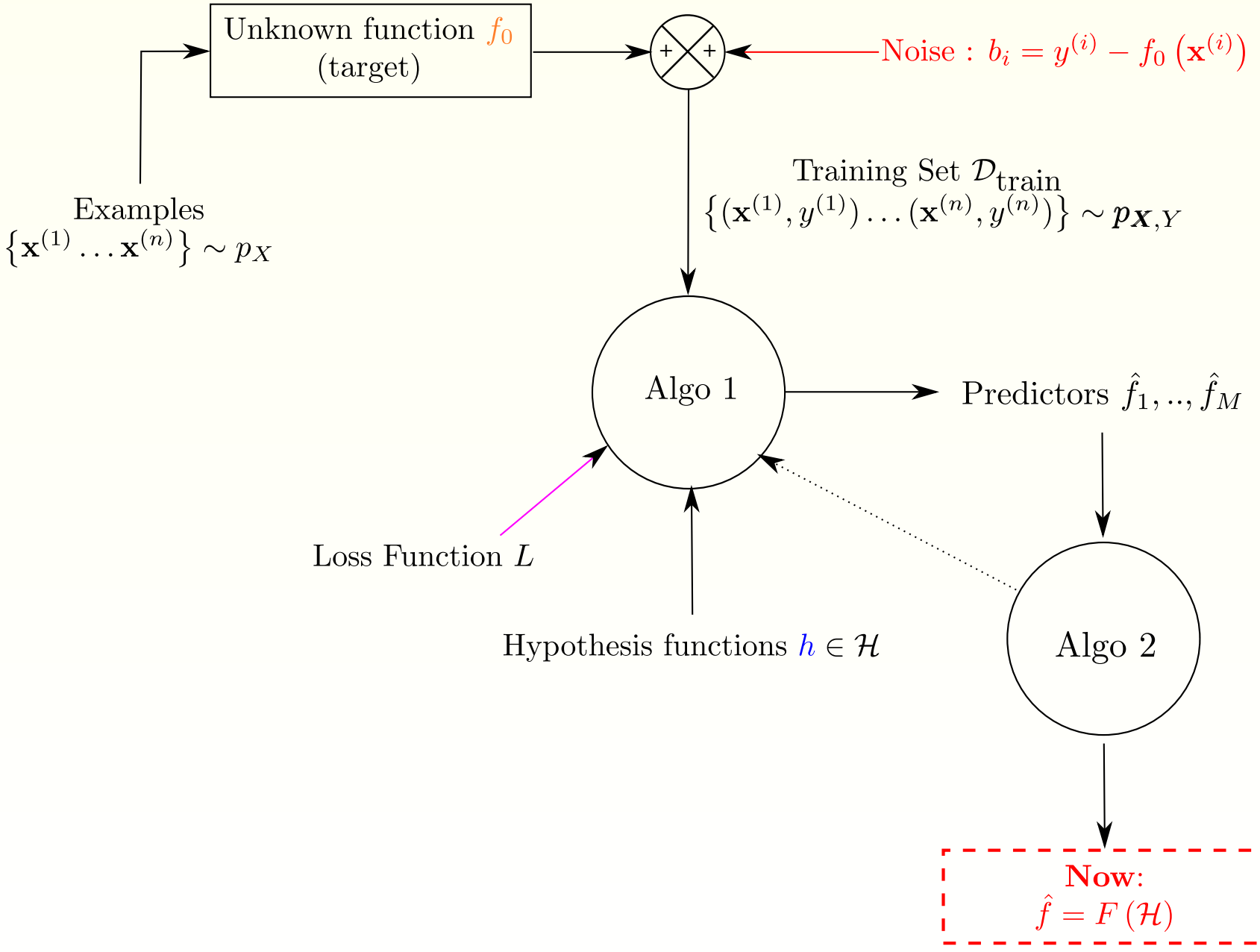


Ensemble Learning :

- Notion of hypothesis set $\mathcal{H}$ of prediction functions



Ensemble Learning : the learning diagram

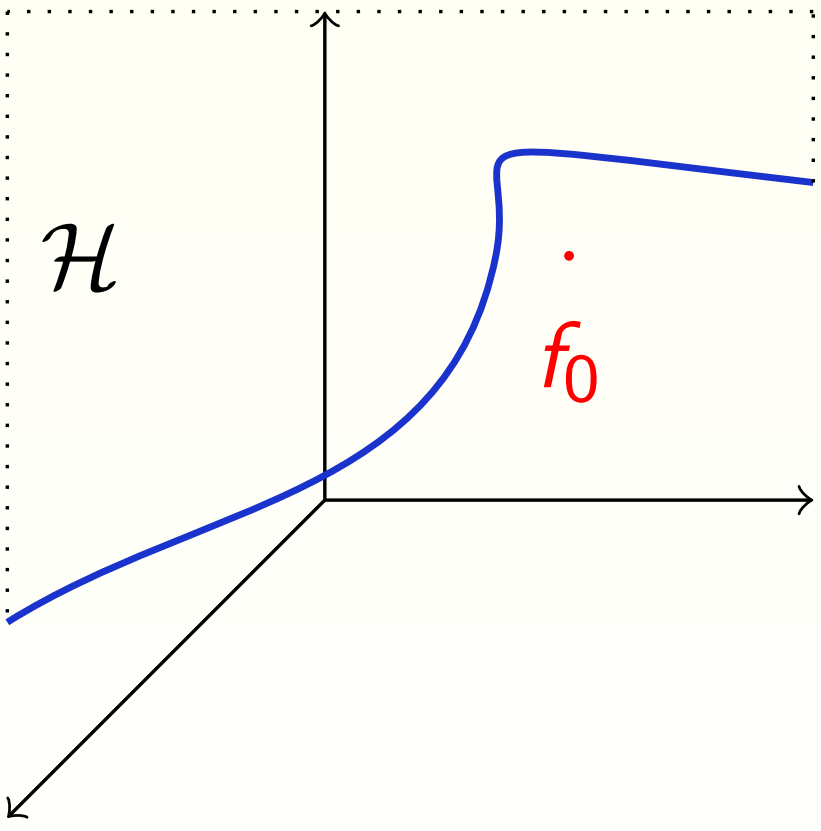


Ensemble Learning : 3 scenarios

- Aggregation : Alg. 2 after Alg. 1
- Hypothesis Blending : Alg. 1 & 2 at the same time
- Boosting : iterating Alg. 1, Alg. 2, Alg. 1, Alg. 2 ...

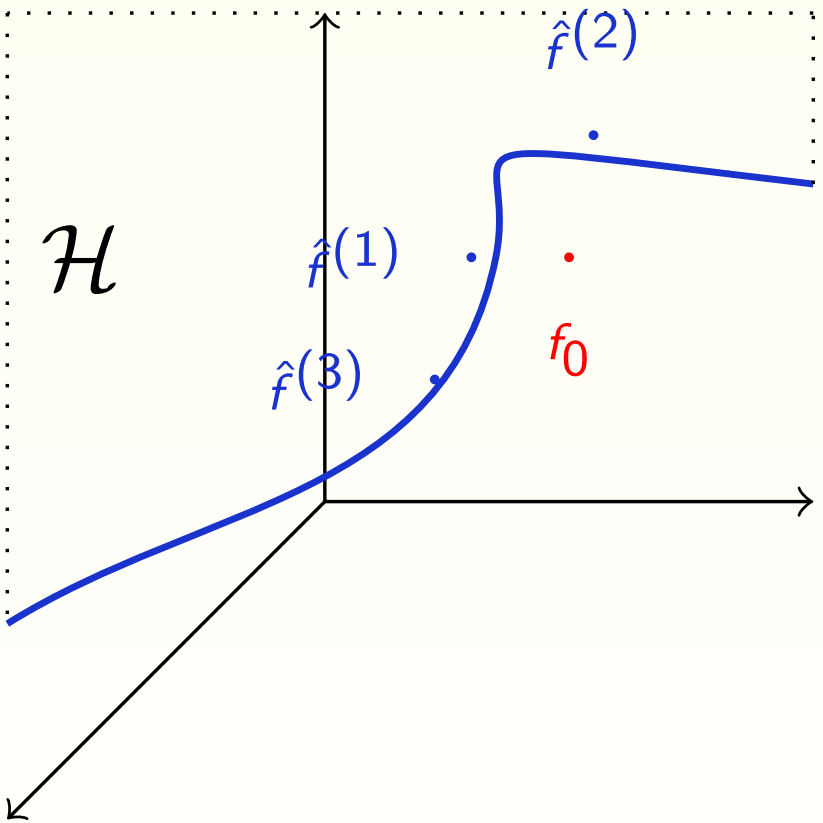
Ensemble Learning :

- No free lunch : higher learning capacity $\rightarrow$ more parameters to learn !



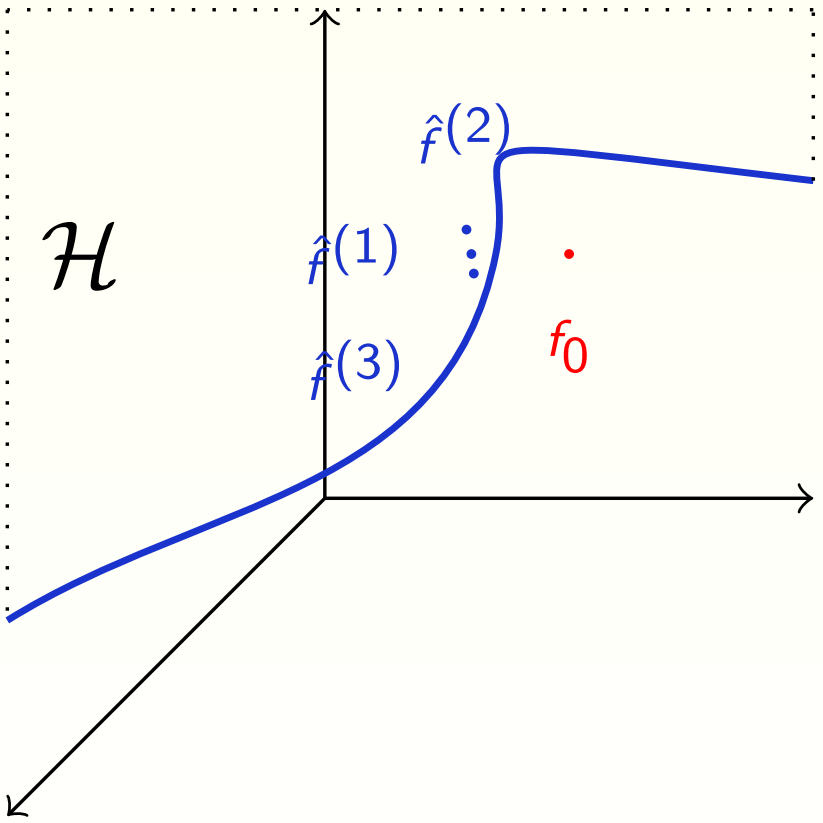
Combining predictors :

- We want diversity in the $\hat{f}^{(i)}$:



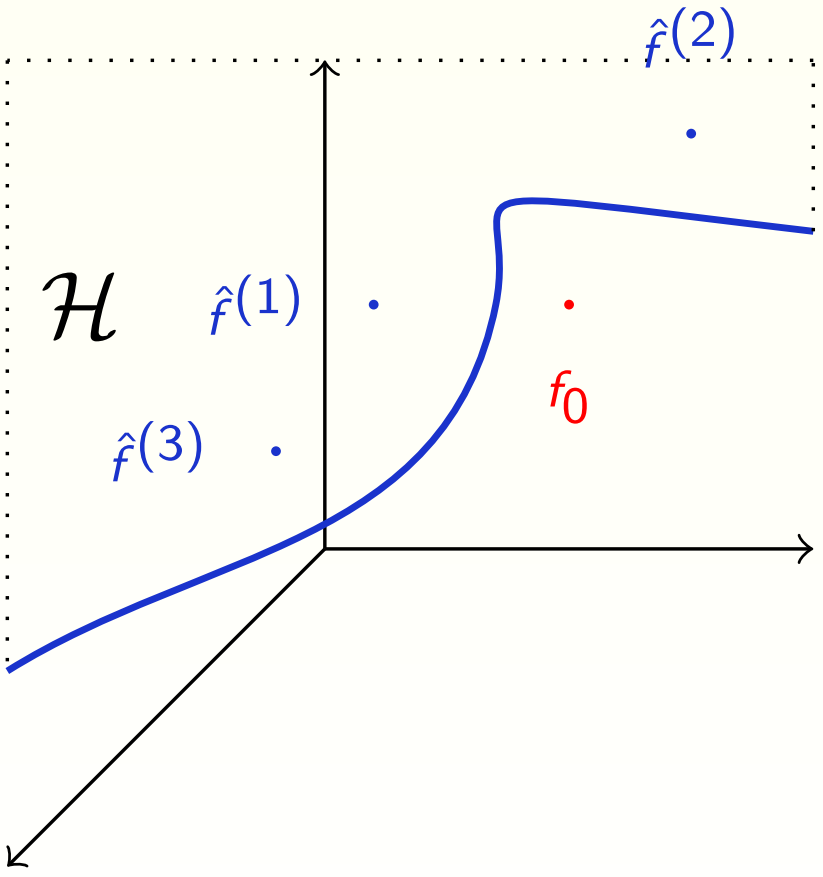
Combining predictors :

- We don't want consanguinity in the $\hat{f}^{(i)}$:



Combining predictors :

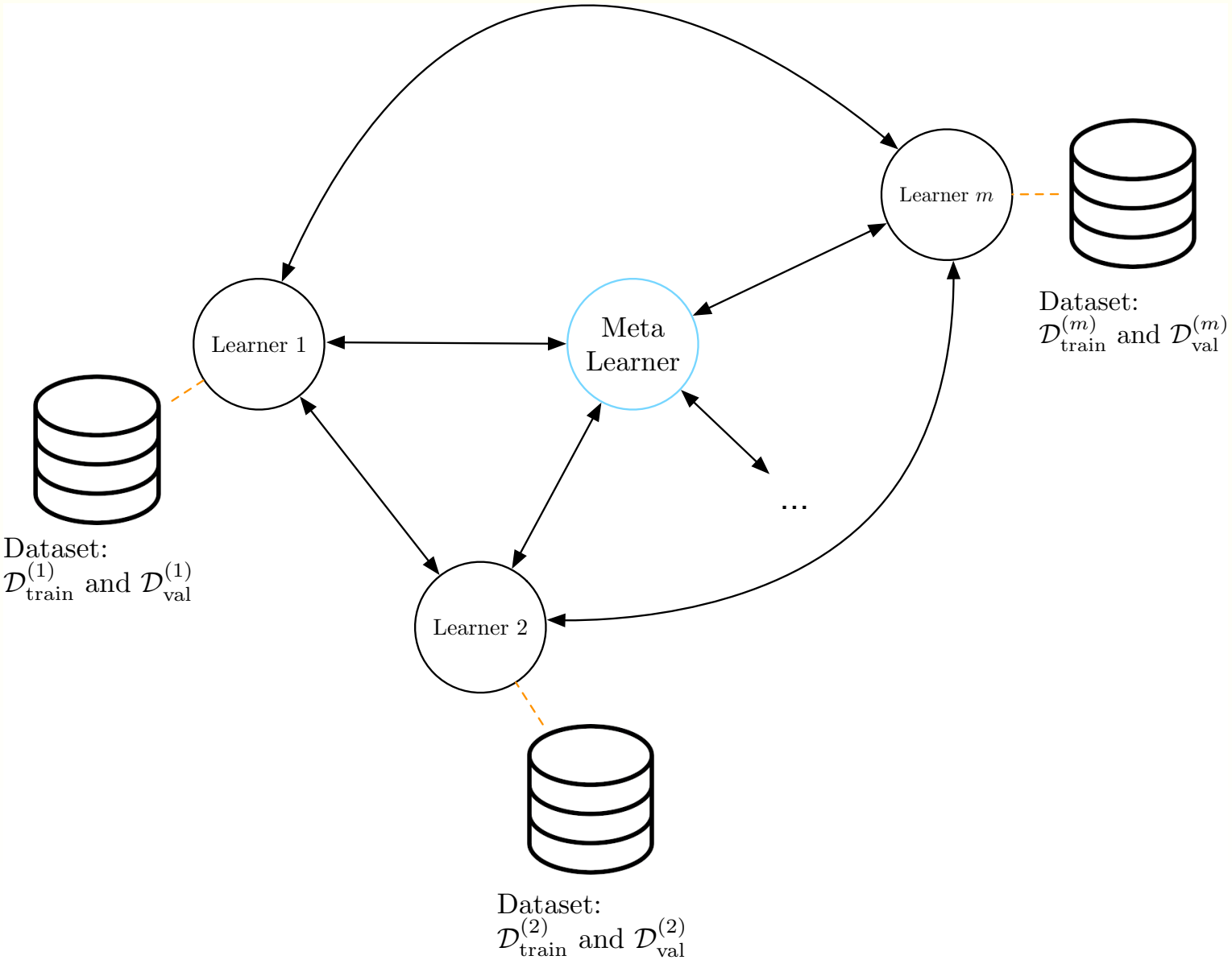
- We also don't want bad individual accuracy for the $\hat{f}^{(i)}$:



Aggregation : when we have to

Decentralized learning

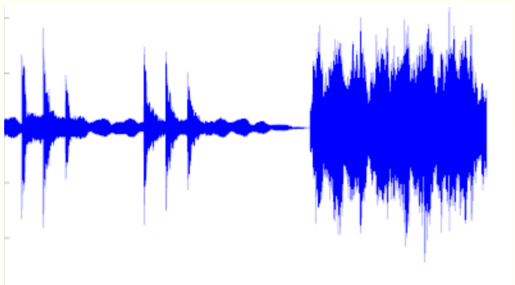
Goal : learn from several (remote and private) datasets



Aggregation : when we have to

Learning from composite inputs

Goal : learn from inputs that are tuples of signals

$\mathbf{x} =$  and 

- Cannot train a neural network on this pair of signals
- Need to (pre)train two networks for each type of signal and aggregate them.

Voting systems

Definition

A **voting system** is a fusion method applicable to any predictions.
A voting system is a made of :

- a **ballot**, which specifies the answer expected from a voter (= structure of the input space)
- a **tallying algorithm**, which specifies how votes are combined to produce election results (= aggregation rule).

Voting systems

- Ballot forms :
  - Plurality ballot : only one choice : $f_m(\mathbf{x}) \in \mathcal{C}$.
  - Approval ballot, several choices : $f_m(\mathbf{x}) \subset \mathcal{C}$.
  - Cumulative ballot, several points to distribute : $f_m(\mathbf{x})$ is a histogram on $\mathcal{C}$.
  - Ranked ballot, a score is assigned to each candidate label : $f_m(\mathbf{x}) \in \mathbb{R}^M$.

Majority Vote ensemble

- Let f_{ens} denote the aggregated predictor.
- The aggregation rule reads

$$f_{\text{ens}}(\mathbf{x}) = \arg\max_{y \in \mathcal{C}} \sum_{m=1}^{M} \mathbb{1}_y \left(f_m(\mathbf{x}) \right)$$

Super-Majority Vote ensemble

- Compute $s = \arg\max_{y \in \mathcal{C}} \sum_{m=1}^{M} \mathbb{1}_y \left(f_m \left(\mathbf{x} \right) \right)$
- If $\sum_{m=1}^{M} \mathbb{1}_s \left(f_m \left(\mathbf{x} \right) \right) >$ threshold then return $f_{\text{ens}} \left(\mathbf{x} \right) = s$
- Else ... do sth about it

Vote based rules : good way to aggregate ?

- Suppose $\mathcal{C} = \{0; 1\}$.

Theorem

Condorcet jury's theorem. *Suppose M is odd : $M = 2R + 1$.*
Suppose $\theta = \mathbb{P}(f_m(\mathbf{x}) = y)$ for any m and any pair $(\mathbf{x}, y)$.
Let $p_M = \mathbb{P}(f_{ens}(\mathbf{x}) = y)$ where f_{ens} is the majority vote ensemble.

The following properties hold :

(i) *If $\theta > \frac{1}{2}$ then $p_M \xrightarrow[M \to +\infty]{} 1$.*

(ii) *If $\theta < \frac{1}{2}$ then $p_M \xrightarrow[M \to +\infty]{} 0$.*

(iii) *If $\theta = \frac{1}{2}$ then $p_M = \frac{1}{2}$.*

Vote based rules : good way to aggregate ?

- Suppose $\{\mathbb{1}_y(f_m(\mathbf{x}))\}_{m=1}^M$ is an i.i.d. sample drawn from a Bernoulli random variable Z on the probability space made of two events : $\{y=f_m(\mathbf{x})\}$ and $\{y\neq f_m(\mathbf{x})\}$.
- $\mathbb{E}[Z] = \theta$.
- The law of large numbers implies that the proportion of classifiers choosing the correct label y tends to θ as M increases.
- Now if $\theta > \frac{1}{2}$, then y has a majority !
- But wait .. does this apply in practice ?

Weighted Vote ensemble

- Ranked ballot : each classifier is assigned a weight w_m
- The aggregation rule reads

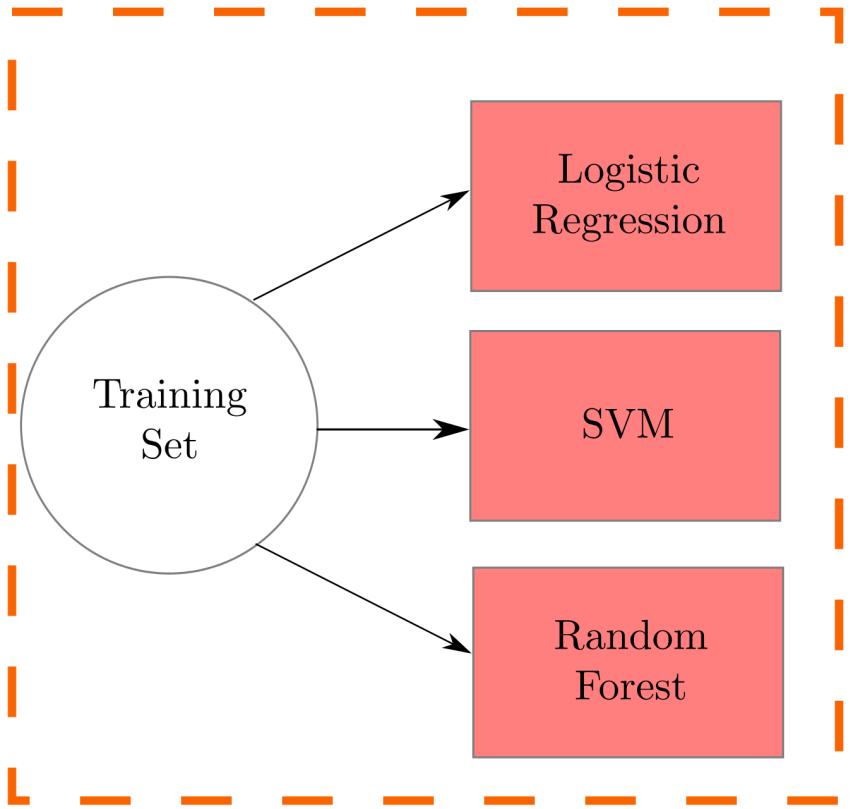
$$f_{\text{ens}}(\mathbf{x}) = \arg\max_{y \in \mathcal{C}} \sum_{m=1}^{M} w_m \mathbb{1}_y (f_m(\mathbf{x})).$$

- Principled way to set the $(w_m)_{m=1}^M$?

Aggregation

Classification : combining predictions that are class labels

Weighted Vote ensemble

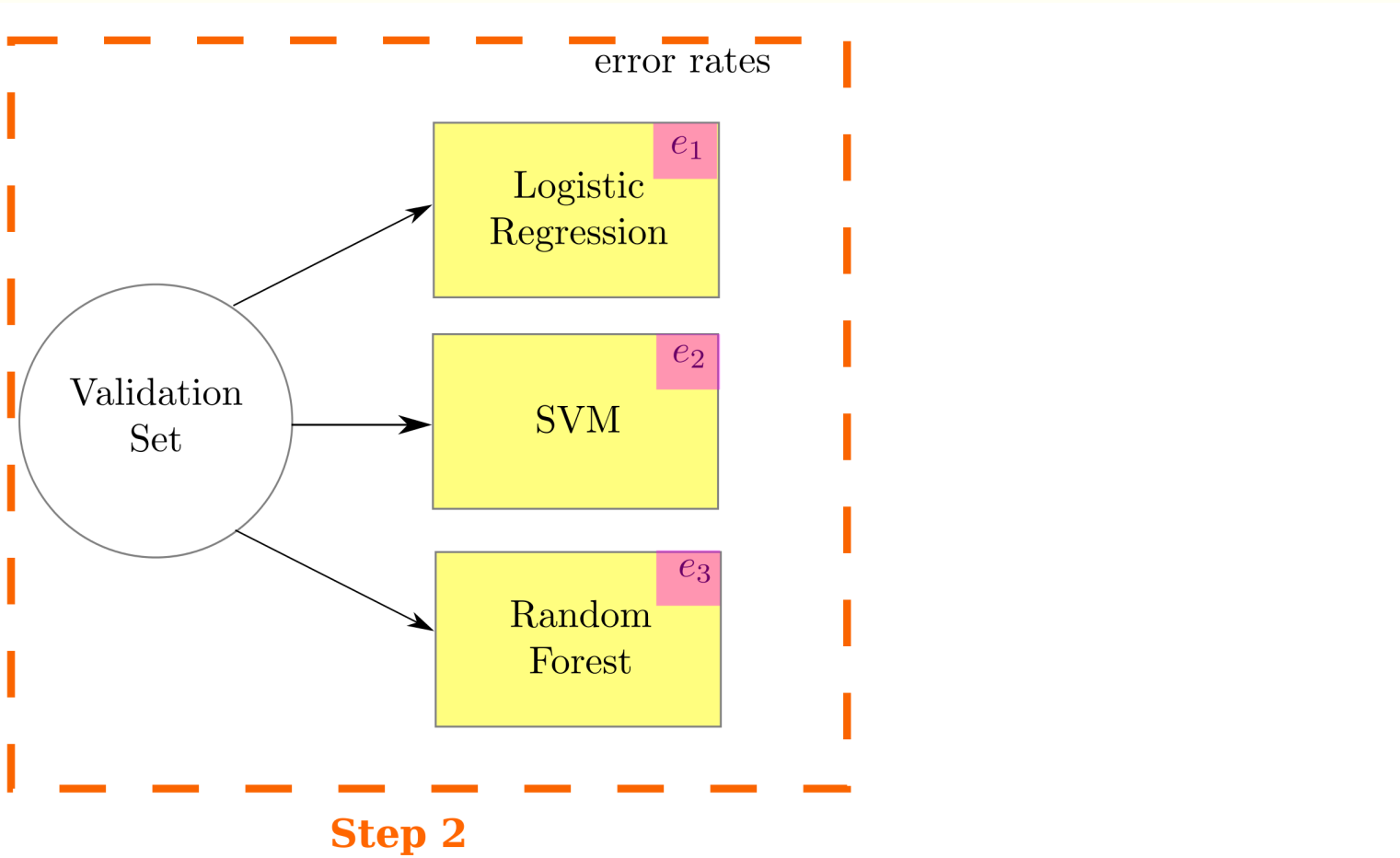


Step 1

Aggregation

Classification : combining predictions that are class labels

Weighted Vote ensemble



Weighted Vote ensemble

- The error rate (risk) is $e_m = \mathbb{E}_{\mathbf{x},y\sim p_{X,Y}}\left[L_{0-1}\left(y, f_m\left(\mathbf{x}\right)\right)\right]$.
- We can set $w_m = 1 - e_m$.
- **Step 3** : return prediction $\arg\max_{y\in\mathcal{C}} \sum_{m=1}^{M} w_m \mathbb{1}_y\left(f_m\left(\mathbf{x}\right)\right)$ for any unseen example $\mathbf{x}$.

Weighted Vote ensemble is intuitive but is it optimal ?

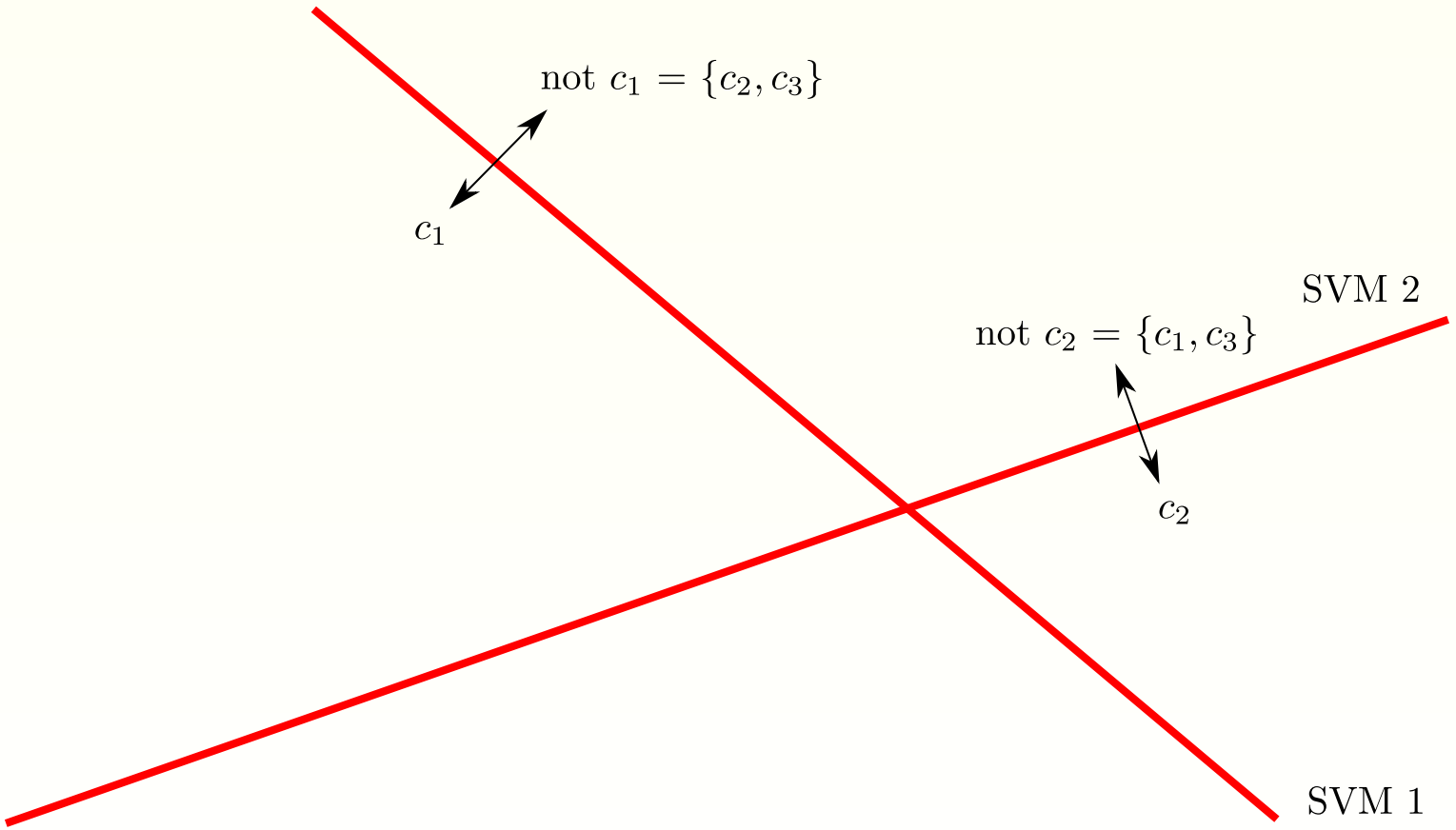
Exponentially Weights achieves a form of optimality.

- Empirical risk $\hat{e}_m = \frac{1}{n_{\text{val}}} \sum_{(\mathbf{x},y)\in\mathcal{D}_{\text{val}}} L_{0-1}\left(y, f_m\left(\mathbf{x}\right)\right)$.
- In this setting, we set $w_m = \frac{\exp(-\eta\hat{e}_m)}{\sum_{m'}\exp(-\eta\hat{e}_{m'})}$.
- $\hat{e}_{\text{ens}}$ is the ensemble empirical risk.
- $\mathbf{w} = [w_1..w_M]$ denotes the vector of convex weights.
- EWV solves $\arg\min_{\mathbf{w}} \sum_{m=1}^{M} w_m\hat{e}_m + \text{pen}_{\text{KL}}\left(\mathbf{w}\right)$
- For some convex loss, $\sum_{m=1}^{M} w_m\hat{e}_m \geq \hat{e}_{\text{ens}}\left(\mathbf{w}\right)$.

Aggregation : when we have to

Large scale problems : large number of class labels

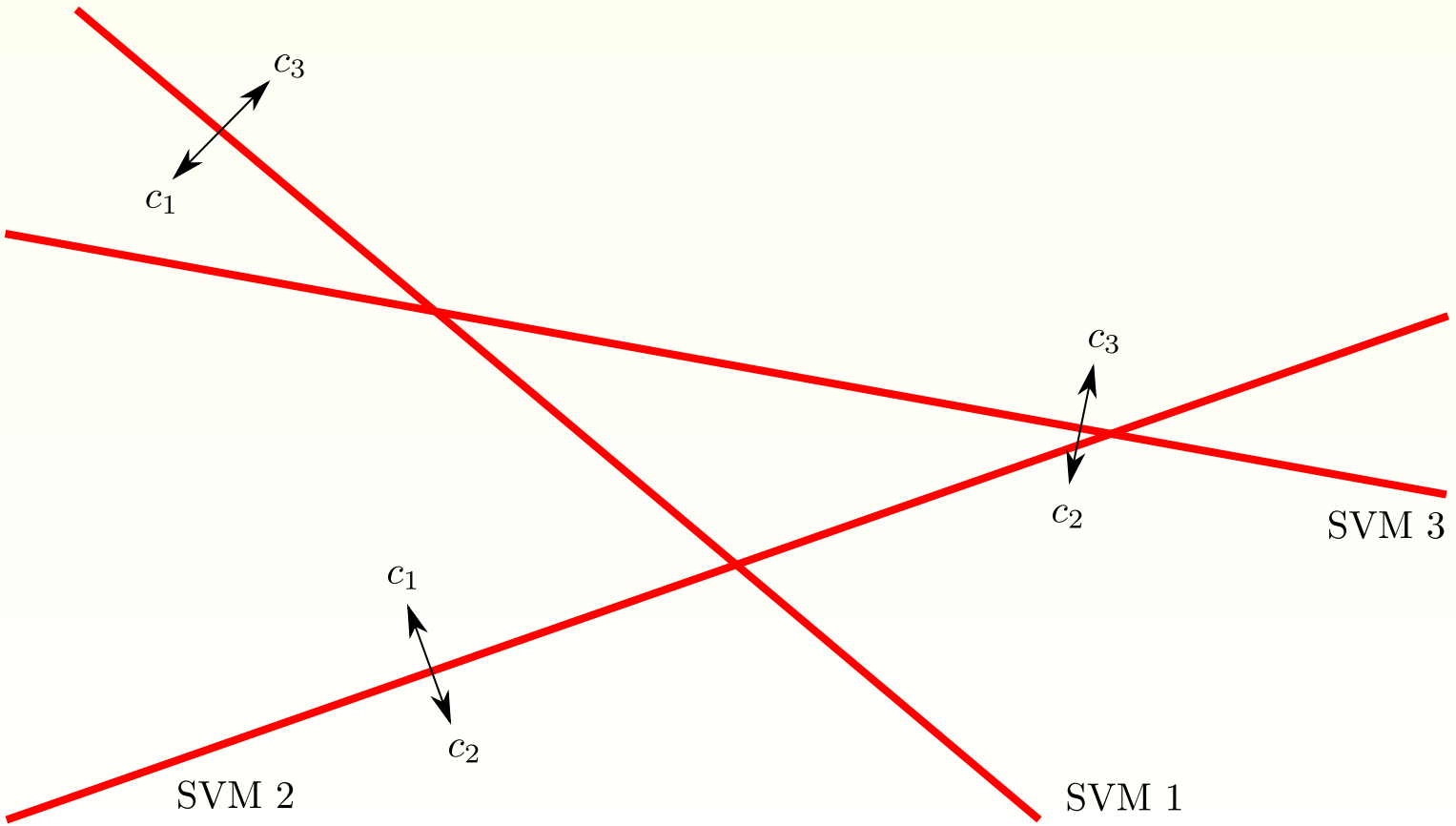
Goal : use SVMs in multi-class problems



Aggregation : when we have to

Large scale problems : large number of class labels

Goal : use SVMs in multi-class problems



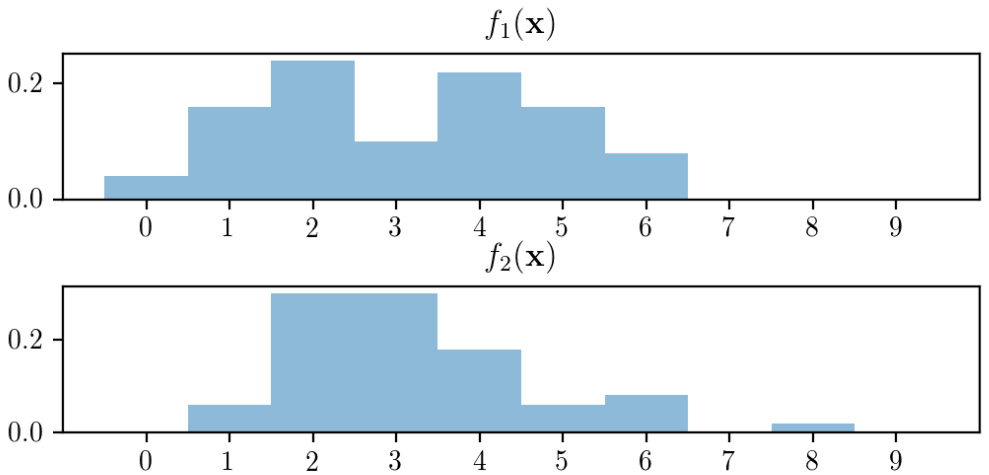
Approval Majority Vote ensemble

- So here $A_m = f_m(\mathbf{x})$ is a **subset** of $\mathcal{C}$.
- The aggregation rule reads

$$f_{\text{ens}}(\mathbf{x}) = \arg\max_{y \in \mathcal{C}} \sum_{m=1}^{M} \mathbb{1}_{A_m}(y)$$

- Weighted versions also welcome !
- Remark : for SVMs, weights can be drawn from margins.

- Many classifiers provide more information than their predictions : a vector of class label scores.
- When scores are unnormalized, we do not have supervision.
- When scores are probability distributions, we know that the correct prediction is δ_y !



Soft Majority Vote ensemble

- So here $p_m = f_m(\mathbf{x})$ is a **probability** conditional distribution : $p_m(y) = \mathbb{P}(y|\mathbf{x}, h_m)$.
- The aggregation rule reads

$$f_{\text{ens}}(\mathbf{x}) = \arg\max_{y \in \mathcal{C}} \sum_{m=1}^{M} p_m(y)$$

- Linear Opinion Pools = weighted version with convex coefficients : $\sum_m w_m = 1$ and $w_m \geq 0, \forall m$
  $\Rightarrow$ the convex combination of probability distributions remains a probability distribution.

Average

- So here $f_m(\mathbf{x}) \in \mathcal{Y} = \mathbb{R}$.
- The output space is a vector space.
- Linear aggregation is basically an average prediction :

$$f_{\text{ens}}(\mathbf{x}) = \frac{1}{M} \sum_{m=1}^{M} f_m(\mathbf{x})$$

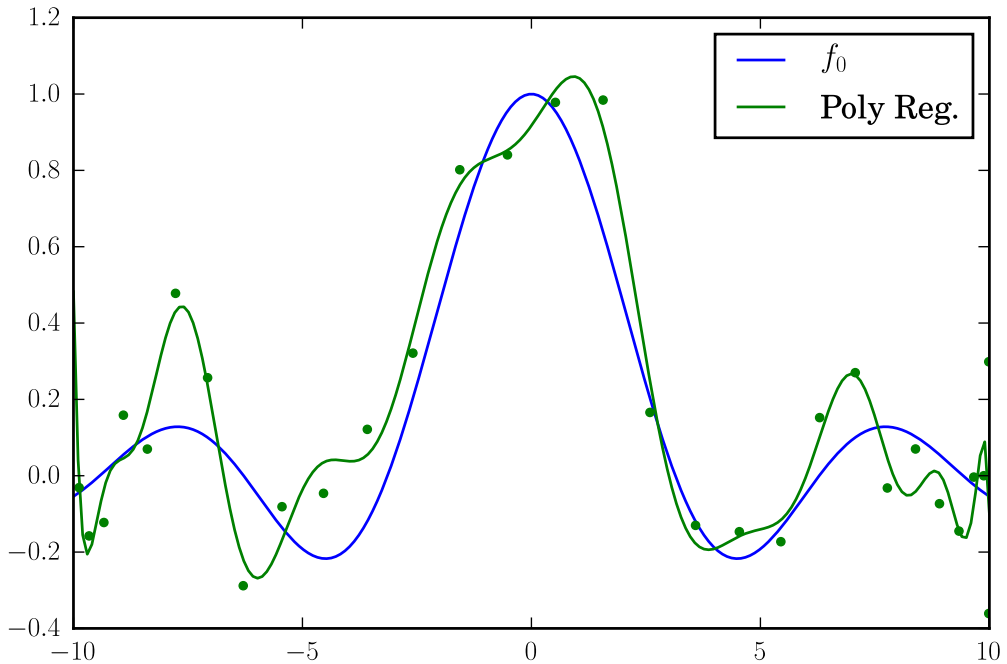
- When $\mathcal{C}$ contains quantized numbers from $\mathcal{Y}$, averages may be used also in classification with the help of a rounding function.
- Score based aggregation for classification amounts to aggregation for regression.

Bagging

1. Generate a dataset $\mathcal{D}_{\text{boot}}$ from $\mathcal{D}_{\text{train}}$ (unif. sampling with replacement).
2. Run your training algorithm on $\mathcal{D}_{\text{boot}}$.
3. After M such trainings, combine using majority voting (classifiers) or average (regressors).

Bagging

- Suppose one fits a 20 degree polynomial on this noisy data :

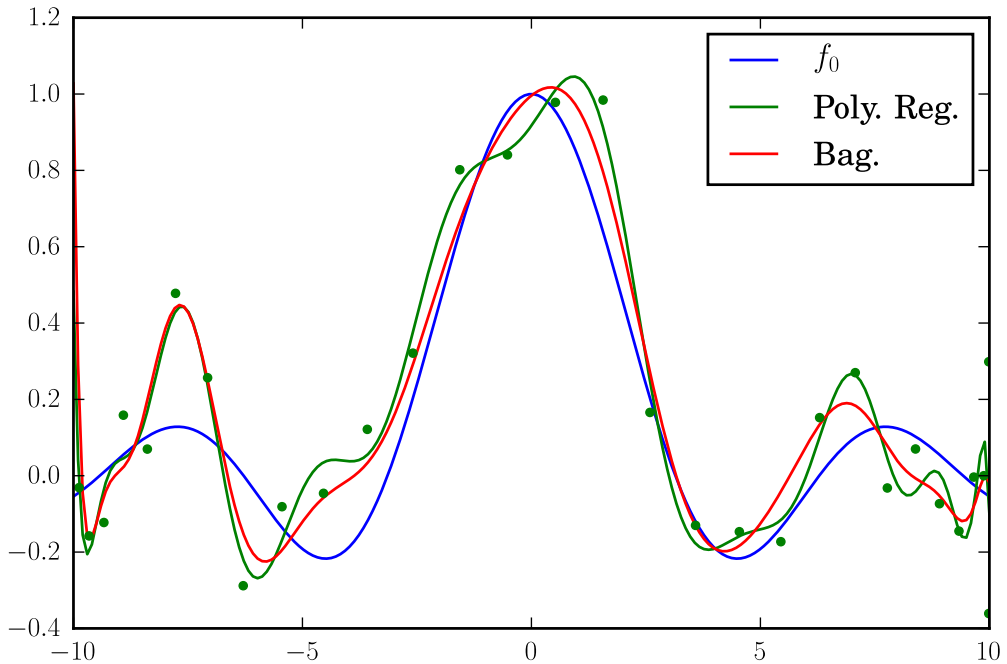


Aggregation : when we want to

Regression : combining predictions in $\mathcal{Y} = \mathbb{R}$

Bagging

- Bagging will help a bit to mitigate overfitting



Bagging

- Let ϵ_m denote the (signed) error function between the true function f_0 and one of my predictors f_m that was trained on a bootstrap sample (no noise) :

$$\epsilon_m(\mathbf{x}) = f_m(\mathbf{x}) - f_0(\mathbf{x}).$$

- For regression problems, the usual loss function is the **quadratic** one.

- The expected loss for the predictor $f_m(\mathbf{x})$ is thus :

$$e_m = \mathbb{E}_X\left[(f_m(\mathbf{x}) - f_0(\mathbf{x}))^2\right] = \mathbb{E}_X\left[\epsilon_m(\mathbf{x})^2\right]$$

Bagging

Now, the expected loss for the bagging ensemble writes :

$$e_{\text{ens}} = \mathbb{E}_X \left[\left(\frac{1}{M} \sum_{m=1}^{M} f_m(\mathbf{x}) - f_0(\mathbf{x}) \right)^2 \right] = \mathbb{E}_X \left[\left(\frac{1}{M} \sum_{m=1}^{M} \epsilon_m(\mathbf{x}) \right)^2 \right]$$

If $\mathbb{E}_X \left[\epsilon_m^2 \right] \leq C \ (\forall m)$ and $\mathbb{E}_X \left[\epsilon_m \epsilon_{m'} \right] = 0 \ (\forall m \neq m')$, then

$$e_{\text{ens}} = \frac{1}{M} \times e_{\text{ave}} \quad \text{with} \quad e_{\text{ave}} = \frac{1}{M} \sum_{m=1}^{M} \mathbb{E}_X \left[\epsilon_m^2 \right] = \frac{1}{M} \sum_{m=1}^{m} e_m$$

$\rightarrow$ reduced error !

Random Subspaces

- Same idea as bagging, but instead of choosing at random examples, one choses at random features !
- Random draws are also with replacement, so some base learners will "focus" on some features.
- The random subspace method is instrumental for **fat** data ($\text{len}(\mathbf{x}) > n$).
- Random forest = Decisions Tree + Bagging + Random Subspace.