IFT6390
Fondements de l'apprentissage machine

Ensemble Methods

Bagging - Boosting - AdaBoost

Professor: Ioannis Mitliagkas
Slides: Pascal Vincent

Reminder: Machine Learning Set-up

- Training set (finite sample): $\mathcal{D} = \{(x_1, y_1), \dots, (x_m, y_m)\}$
- We assume i.i.d. samples drawn from an unknow distribution P:

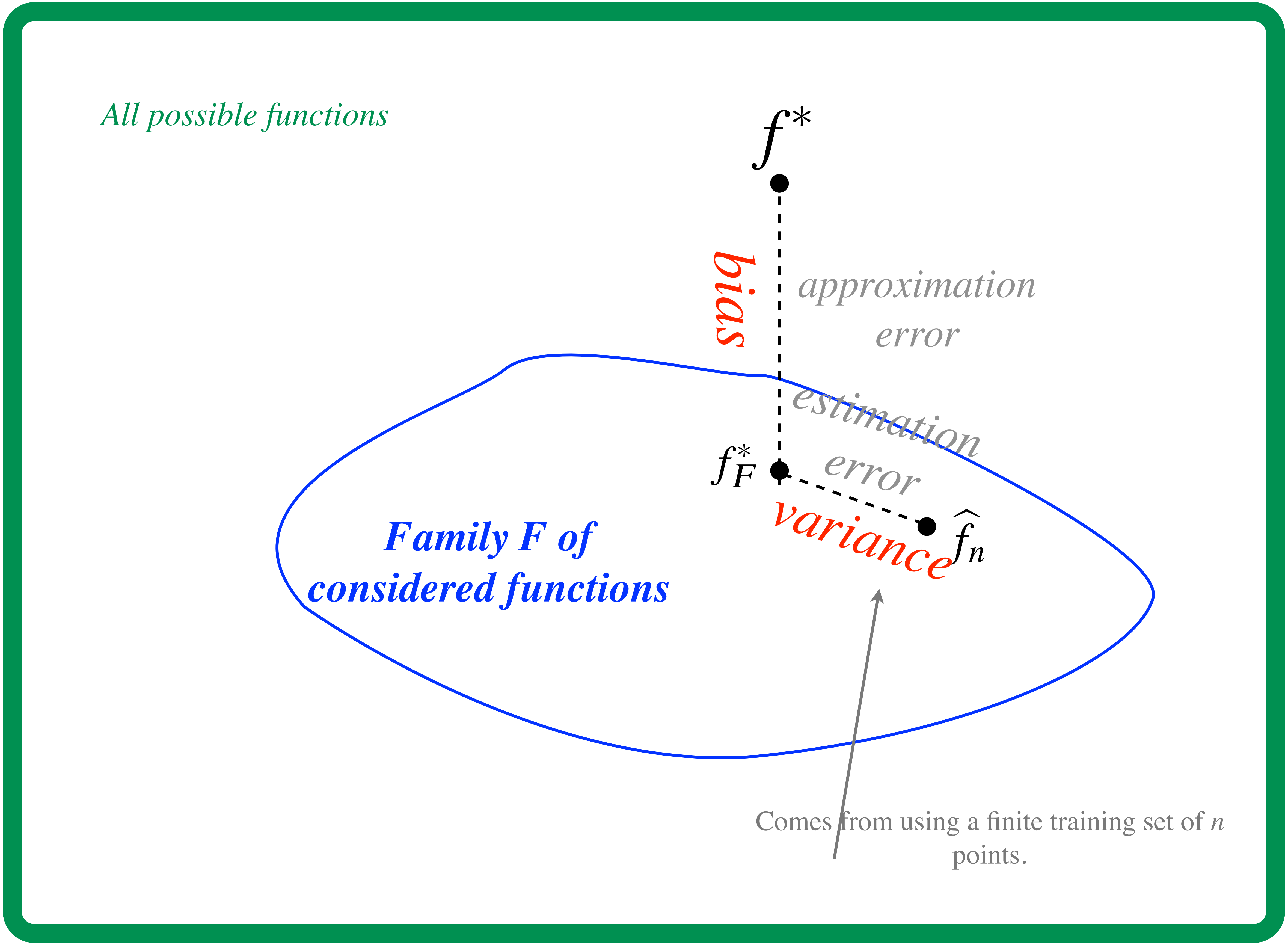
$$(x_i, y_i) \sim P(X, Y)$$

- We «train» a model (e.g. classifier) on *D*: we find the model with the lowest training error wihtin some chosen class of functions.
- What we really care about is the generalization error: expectation w.r.t. *P*, instead of (empirical) mean over *D*
- If we draw a new sample from *P*, we would get a different training set *D* and thus a different classifier ➪ variance

Reminder: bias variance trade-off

- The generalisation error can be decomposed into two terms:
- The bias term (approximation error):
  if the class of functions we consider does not contain the *true* function we're looking for.
  (ex: learning a linear classifier but the true decision boundary is non-linear)
- The variance term (estimation error):
  variance in in the function we learn due to the variance in the training data
  (because we only have a *finite* number of training examples).
- bias/varance tradeoff: if we consider a larger set of functions (hypothesis class), we get less bias but more variance...

Bias-variance tradeoff



Ensemble methods
(for classification or regression)
a simple idea

- Combine several predictors
  (classifiers or regressors)
- Each one trained on a slightly different version of the training set
- Goal: obtain more stability (i.e. reduce the variance) or capacity (i.e. reduce the bias).

Ensemble methods

What you need:

- Training data: $\mathcal{D} = \{(x_1, y_1), \ldots, (x_m, y_m)\}$
- Learning algorithm A returning a predictor (classifier or regressor) $h(x)$ from D.

$$h = A(\mathrm{D})$$

Ex: h could be a linear classifier or a decision tree, and A is the algorithm to learn its parameters.

- For binary classification, we use the representation $y \in \{-1, +1\}$
Predictor outputs +1 or -1.
- For multiclass, we use the representation $y = \mathrm{onehot}(\mathrm{classe})$
Predictor outputs a one-hot encoding of the class/label.

Bagging (Bootstrap Averaging)

[Leo Breiman 1994]

- Generate T different training data sets $\mathcal{D}_1, \ldots, \mathcal{D}_T$ of size m' from D using Bootstrapping:
  D_t is built by randomly choosing m' elements from D with replacement.
  (we can use $m'=m$ or $m'<m$)

- Learn a predictor on each of the training sets using A:

$$h_t = A(\mathcal{D}_t)$$

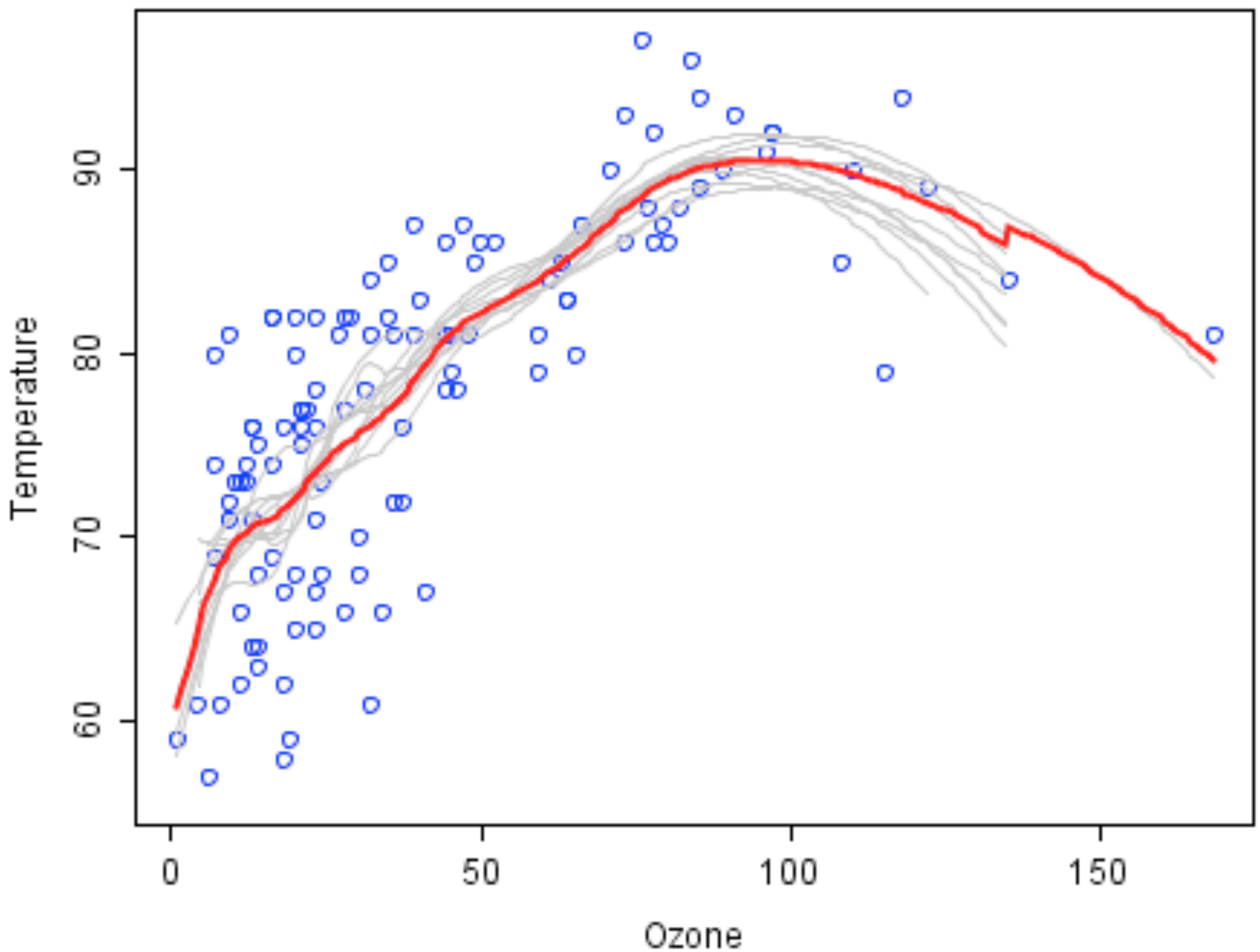
- The predictor resulting from *Bagging* simply is the mean of these *T* predictors:

$$f(x) = \frac{1}{T} \sum_{t=1}^{T} h_t(x)$$

For classifiers (using +1/-1 or one-hot encoding) this corresponds to a majority vote.

Illustration

Variance reduction with bagging on a regression problem



Random Forest

- A forest is a set of trees!
- Each decision tree is trained using a slightly different version of the training set, using both by
  - Bootstrapping (random selection of examples, with replacement)
  - Randomly selecting a subset of features (reduce the input dimension)
- The capacity of each tree (i.e. depth) is chosen using (cross-) validation.
- The final prediction is a majority vote, like in Bagging.

Boosting (AdaBoost)

[Y. Freund and R.E. Schapire 1995]

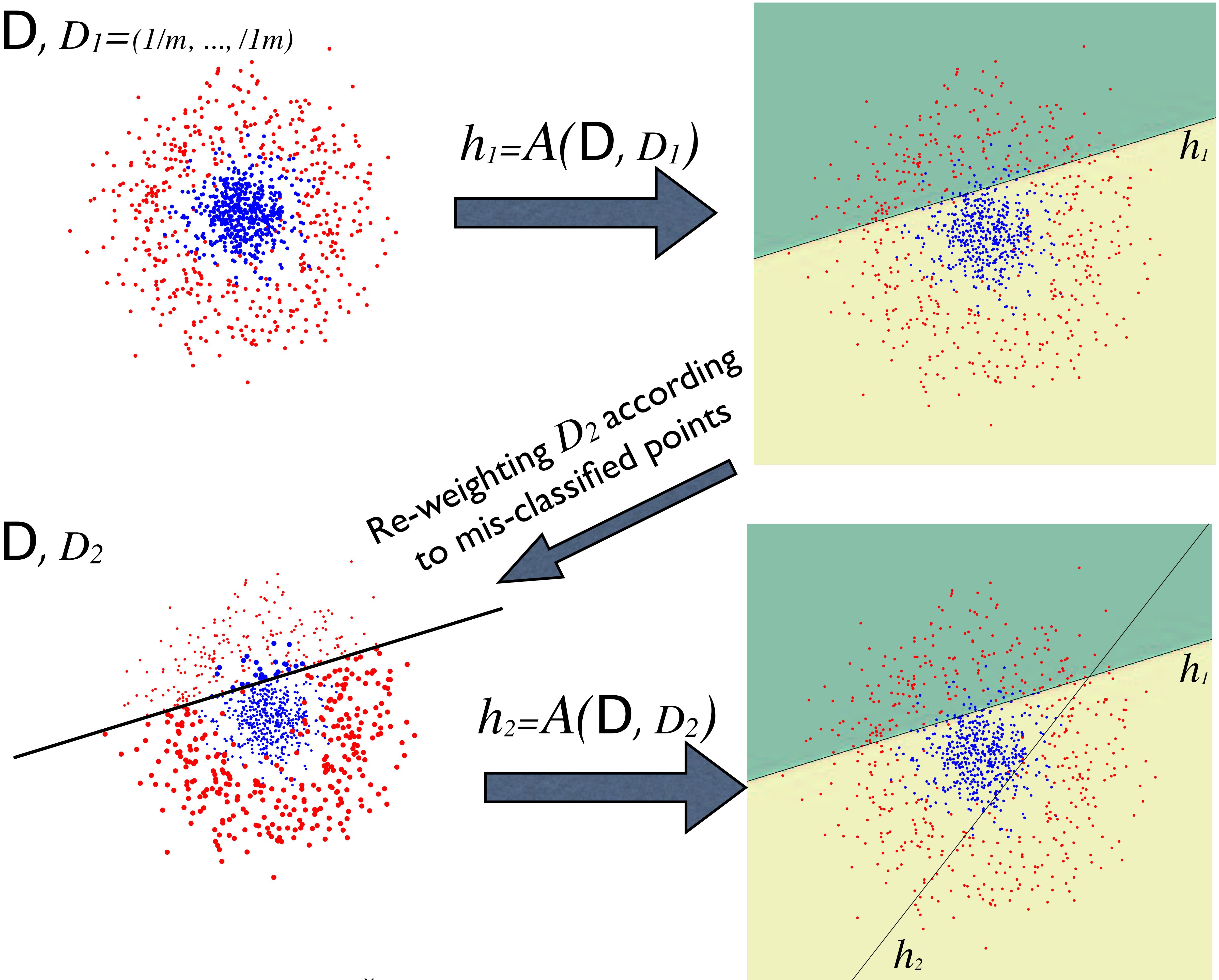
- We will build a strong classifier H whose discriminant function f is obtained by training several weak classifiers $h_1, \dots, h_T$ with a learning algorithm «weak classifier» A.
- f will be a linear combinaison of the weak classifiers h_t:

$$f(x) = \sum_{t=1}^{T} \alpha_t h_t(x) \qquad\qquad \alpha_t \geq 0$$

- For binary classification with outputs +1/-1: $H(x)=\mathrm{sign}(f(x))$
- AdaBoost learns both the weak classifiers h_t and the weights α_t.

Boosting
Intuition

- h_{t+1} is trained with the goal of correcting the mistakes made by the previous classifiers: the training is *focused* on some of the examples.
- First learn h_1 on D with algorithm A, by giving the same importance to each of the m examples.
- Check which examples in D are mis-classified by h_1
- Learn h_2 using algorithm A, but try to reduce the error by focussing on the examples that are mis-classified by h_1
- Combine h_1 and h_2 and check which examples are still mis-classified
- Learn h_3 trying to reduce the error...
- Add h_3 to the combination, etc...



Graphiques de Jan Šochman, Jiří Matas

Weighted data sets

how to focus on particular points

- We consider a training set $\mathcal{D} = \{(x_1, y_1), \ldots, (x_m, y_m)\}$ weighted by some vector $D = (D(1), \ldots, D(m))$ with $D(i) \geq 0$.

- The weight $D(i)$ tells which relative importance should be given by the classifier to the example (x_i, y_i).

- Ideally, the classifier then tries to learn a function that minimizaes the weighted mis-classification error:

$$A(\mathcal{D}, D) = \arg\min_{h \in \mathcal{H}} \sum_{i=1}^{m} D(i) I_{\{h(x_i) \neq y_i\}}$$

- Usually, we can easily modify a learning algorithm to incorporate weights (weighted empirical risk minimization).

- If not, we can always generate a new (non-weighted) training data set D' by randomly drawing examples from D with probabilities D(i)

AdaBoost

(for binary classification)

[Y. Freund and R.E. Schapire 1995]

Given: $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in X$, $y_i \in Y = \{-1, +1\}$
Initialize $D_1(i) = 1/m$.
For $t = 1, \ldots, T$:

- Train weak learner using distribution D_t. h_t=A(D, D_t) (weak classifier)
- Get weak hypothesis $h_t : X \to \{-1, +1\}$ with error

$$\epsilon_t = \Pr_{i \sim D_t}[h_t(x_i) \neq y_i] = \sum_{i=1}^{m} D_t(i) I_{\{h_t(x_i) \neq y_i\}}$$

- Choose $\alpha_t = \frac{1}{2} \ln\left(\frac{1 - \epsilon_t}{\epsilon_t}\right)$.
- Update:

$$\begin{aligned} D_{t+1}(i) &= \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t} & \text{if } h_t(x_i) = y_i \\ e^{\alpha_t} & \text{if } h_t(x_i) \neq y_i \end{cases} \\ &= \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t} \end{aligned}$$

where Z_t is a normalization factor (chosen so that D_{t+1} will be a distribution).

Output the final hypothesis:

strong classifier: $H(x) = \operatorname{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$.

AdaBoost

- For a detailed presentation, see the slides od
  Jan Šochman, Jiří Matas
- The are a lot of Boosting extensions and variants
  (LogitBoost, AnyBoost, ...)
- AdaBoost can be seen as gradient descent in some function space
  => See Pascal Vincent's boosting_gradient document on StudiUM
- AdaBoost with decision trees is often achieves very good performance in practice
- Used in a very popular face detection algorithm
  [Viola & Jones 2001]

Summary

Bagging:

- Technique for variance reduction
- Mean of predictors trained on variants of D obtained by bootstrapping (re-sampling).
- Very useful for classifiers that are very sensitive to training data (e.g. decision trees with big depth).

Random Foest:

bagging of trees + random feature selection

Boosting:

- Bias reduction
- Combination of weak classifiers (low capacity, high bias)
  ➪ strong classifier (more capacity, less bias)
- Incremental addition of weak learners, trained on re-weighted data (weighted according to previous errors)
- Capacity (and variance) can be controled with early stopping: we stop to add new weak learners
- Works very well with decision trees with very small depth (even with only one node: «stumps»)