IFT3395
Fondements de l’apprentissage machine

Méta-algorithmes d’apprentssage, méthodes d’ensembles (de classifieurs):

Bagging et Boosting (AdaBoost)

Professeur: Pascal Vincent

LISA Laboratoire d’Informatique des Systèmes d’Apprentissage

Rappel: cadre de l'apprentissage

- Ensemble D d'entraînement fini $\mathcal{D} = \{(x_1, y_1), \ldots, (x_m, y_m)\}$
- On suppose tirages i.i.d. d'une distribution P inconnue $(x_i, y_i) \sim P(X, Y)$
- On «entraîne» un modèle (e.g. classifieur) sur D
  Revient à chercher parmi un ensemble de fonction choisi,
  la fonction qui fait le moins d'erreurs sur D
- Mais ce qui nous intéresse vraiment c'est l'erreur de généralisation
  (espérance sur P et non pas moyenne sur D)
- Si on retirait m exemples de P on obtiendrait un D différent
  et donc un classifieur différent ⇨ variance

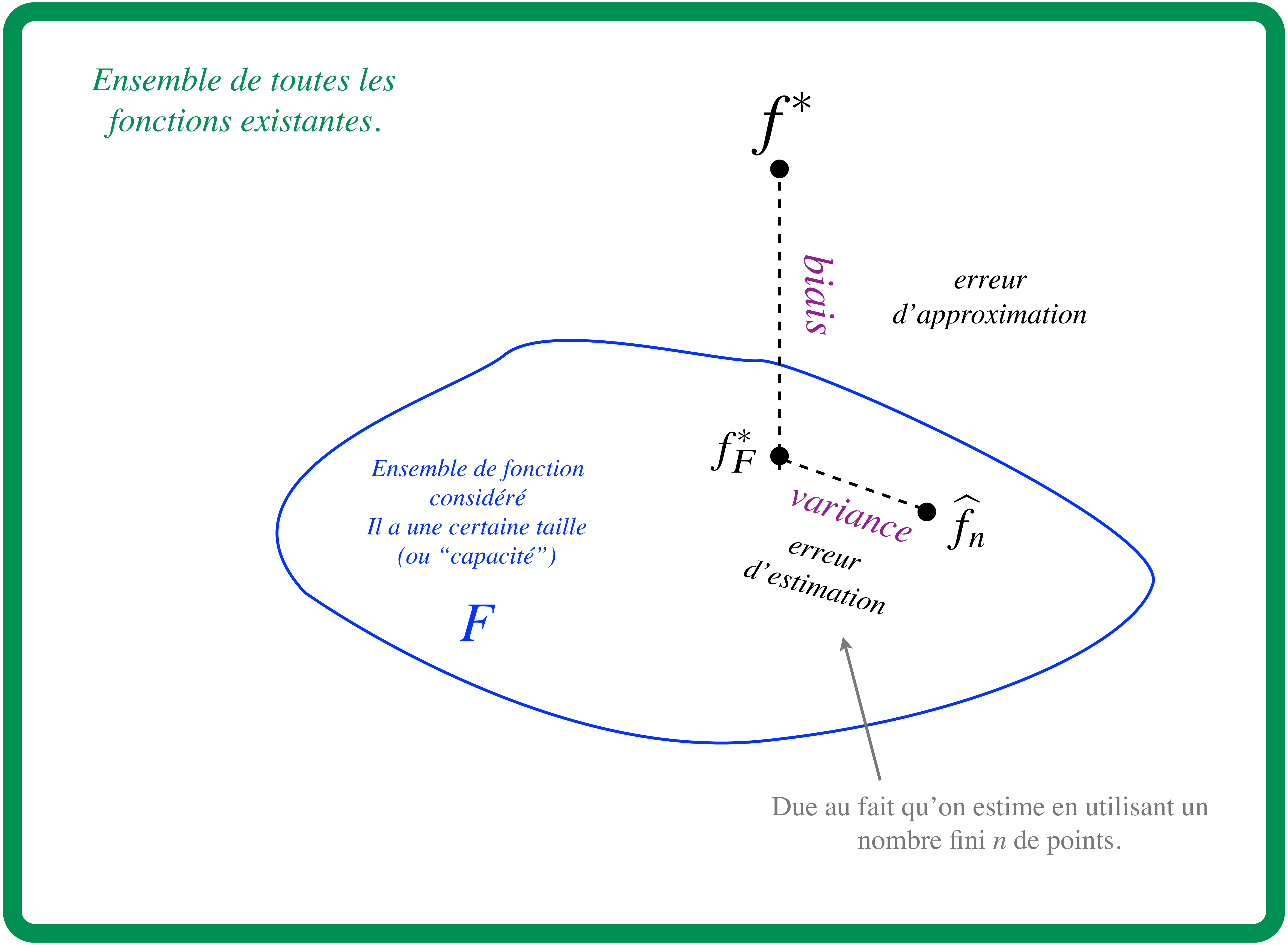
Rappel biais/variance

- L'erreur de généralisation peut se décomposer en deux termes:
- Terme de biais (erreur d'approximation):
  si la famille de fonction considérée ne contient pas la fonction idéale

  (ex: classifieur linéaire mais la vraie frontière de décision est non-linéaire).
- Terme de variance:
  variabilité dans la fonction trouvée due à la variabilité de l'ensemble de données

  (parce qu'on a un nombre fini d'exemples).
- Dilemme biais/variance: quand on enrichit l'ensemble de fonction considéré, on diminue le biais, mais on augmente la variance.

Le dilemme biais-variance



Les méthodes d'ensemble
(de classifieurs ou régresseurs)
une idée simple

- Combiner plusieurs prédicteurs
  (classifieurs ou régresseurs)
- Chacun entraîné sur une version légèrement différente de l'ensemble d'apprentissage.
- But: améliorer la stabilité (réduction de la variance) ou la capacité (réduction du biais).

Méthodes d'ensemble

Nécessitent:

- Ensemble de données d'entraînement $\mathcal{D} = \{(x_1, y_1), \ldots, (x_m, y_m)\}$
- Un algorithme A qui apprend un prédicteur (classifieur ou régresseur) $h(x)$ à partir de D.

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

  Ex: h pourrait être un classifieur linéaire, ou un arbre de décision et A est l'algorithme d'apprentissage utilisé pour en apprendre les paramètres.

- Pour la classification binaire, représentation $y \in \{-1, +1\}$
  Prédicteur retourne +1 ou -1.
- Pour la classification multiple, représentation $y = \mathrm{onehot}(\mathrm{classe})$
  Prédicteur retourne une représentation onehot de la classe gagnante.

Bagging (Bootstrap Averaging)

[Leo Breiman 1994]

- On génère T ensembles de données différents $\mathcal{D}_1, \ldots, \mathcal{D}_T$ de taille m' à partir de D par Bootstrap:
  D$_t$ est constitué en choisissant au hazard m' éléments de D avec remise.
- Sur chacun on entraîne un prédicteur avec l'algorithme A:

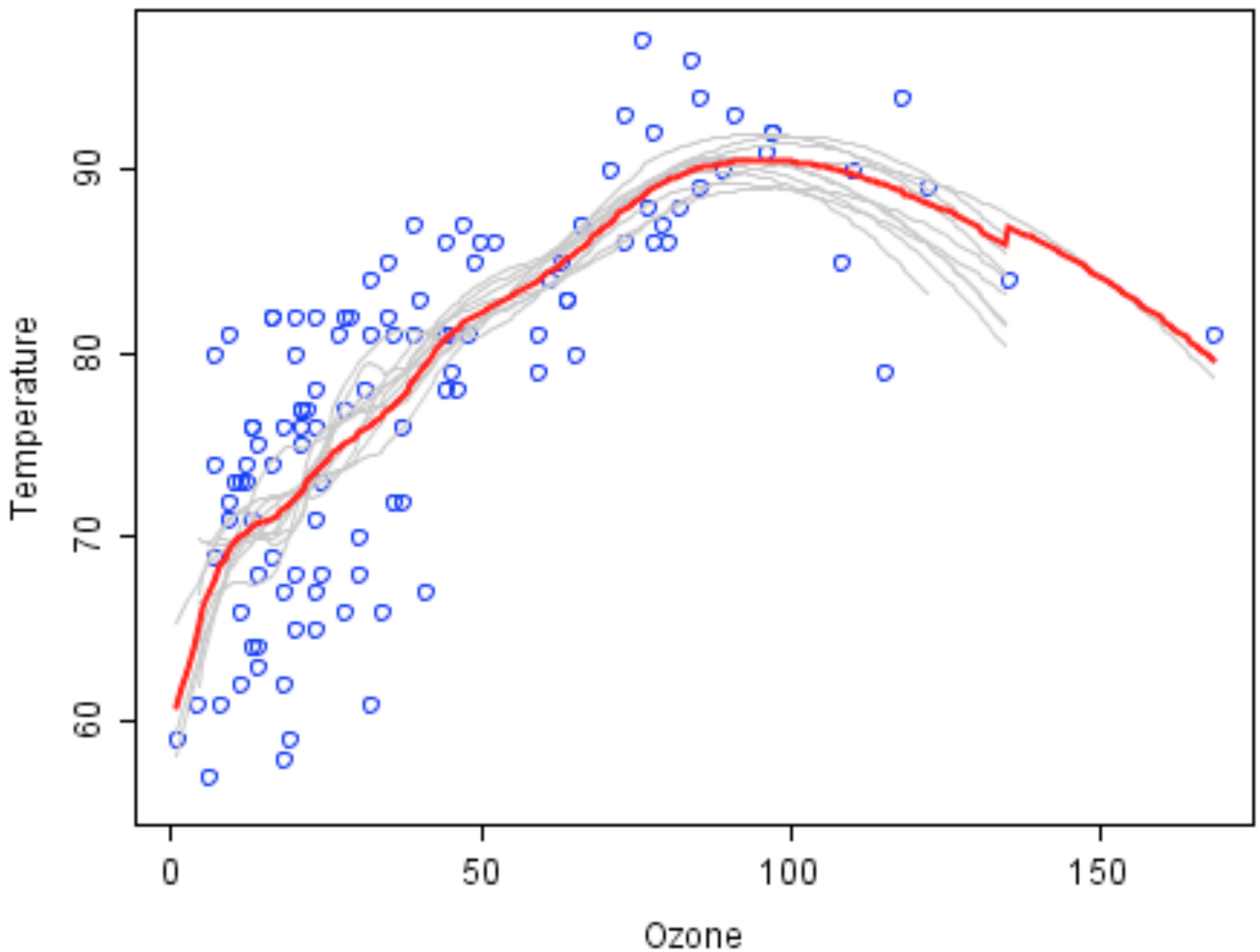
$$h_t = A(\mathcal{D}_t)$$

- Le prédicteur résultant du Bagging fait simplement la moyenne des prédictions de ces T prédicteurs:

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

Pour des classifieurs retournant la classe gagnante (+1 ou -1 ou représentation onehot) cela correspond à un vote majoritaire.

Illustration
Réduction de variance avec Bagging pour un pb de régression



Forêts de décision aléatoire (random decision forest)

- Une forêt est un ensemble d’arbres
- Chacun est entraîné sur une version un peu différente de l’ensemble de donnée d’entrainement, obtenue à la fois
  - par Bootstrap (choix aléatoire d’exemples avec remise)
  - par une sélection aléatoire d’un sous-ensemble des traits caractéristiques (dimensions d’entrée)
- Chaque arbre est entraîné en choisissant sa capacité optimale (profondeur…) par validation simple ou validation croisée.
- La prédiction finale est un vote majoritaire comme dans le Bagging.

Boosting (AdaBoost)

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

- On va construire un classifieur fort H dont la fonction discriminante f est obtenue en entraînant une série de T classifieurs faibles $h_1, \ldots, h_T$ avec un algorithme «classifieur faible» A.

- f est une combinaison des classifieurs faibles h_t :

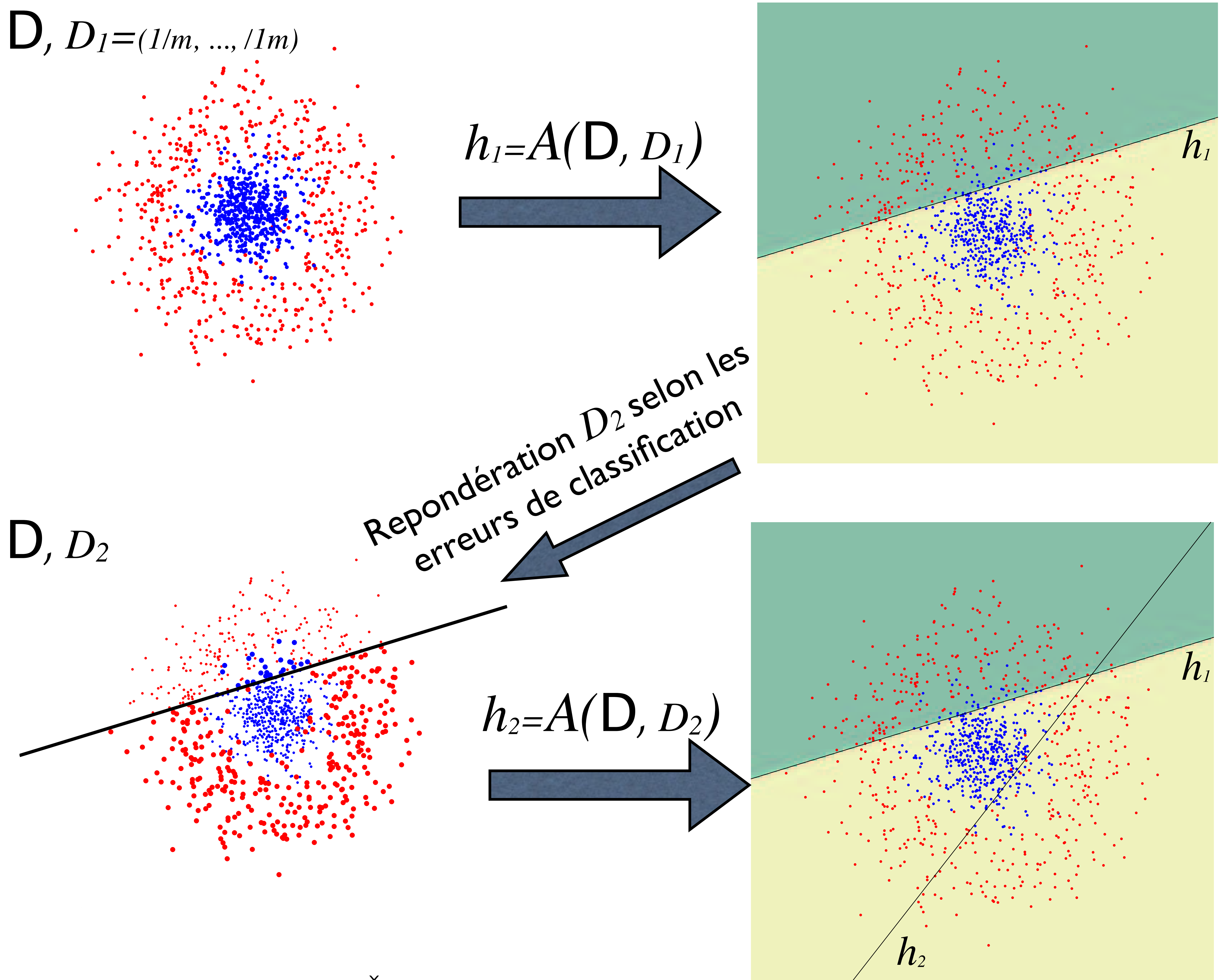
$$f(x) = \sum_{t=1}^{T} \alpha_t h_t(x) \qquad\qquad \alpha_t \geq 0$$

- Dans le cas de classification binaire avec sortie +1 -1, $H(x)=\text{signe}(f(x))$

- AdaBoost apprend et les h_t et les α_t correspondant.

Boosting
Compréhension intuitive

- h_{t+1} est entraîné à corriger les erreurs restantes des classifieurs précédents de la série: son entraînement se concentre davantage sur certains points.
- On apprend h_1 sur D à l'aide de l'algorithme A en accordant autant d'importance à chacun de ses m exemples.
- On regarde quels exemples de D demeurent mal classifiés par h_1
- On apprend h_2 à l'aide de l'algorithme A mais en essayant davantage de réduire l'erreur sur les exemples mal classifiés
- On combine h_1 et h_2 et on regarde quels exemples demeurent mal classfiés.
- On apprend h_3 en essayant de réduire l'erreur sur ces exemples
- on ajoute h_3 à la combinaison etc...



Graphiques de Jan Šochman, Jiří Matas

Ensemble de donnée pondéré

comment se concentrer sur certains points

- On considère un ensemble de données $\mathcal{D} = \{(x_1, y_1), \ldots, (x_m, y_m)\}$ pondéré par un vecteur de poids $D = (D(1), \ldots, D(m))$ avec $D(i) \geq 0$.

- Le poids $D(i)$ indique l'importance relative que le classifieur devrait accorder à l'exemple (x_i, y_i).

- Le classifieur essaye alors (idéalement) d'apprendre une fonction qui minimise l'erreur de classification pondérée:

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

- On peut généralement modifier un algorithme d'apprentissage pour qu'il tienne compte de tels poids (minimisation d'un risque empirique pondéré).

- Sinon on peut toujours générer un nouvel ensemble d'entraînement non pondéré D' en tirant des exemples de D selon la probabilité $D(i)$

AdaBoost

(pour classification binaire)

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

avec mes annotations en bleu

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. $h_t = A(\mathrm{D}, D_t)$ (classifieur faible)
- Get weak hypothesis $h_t : X \to \{-1, +1\}$ with error

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

- 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:

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

AdaBoost

- Pour une présentation plus détaillée voir acétates de Jan Šochman, Jiří Matas

- Il existe de nombreuses extensions et autres variantes de Boosting (LogitBoost, AnyBoost, ...)

- Peut être vues comme une descente de gradient dans un espace de fonctions (AnyBoost)
  => Voir mon document boosting_gradient sur le site du cours.

- AdaBoost avec des arbres de déci très bons résultats en pratique.

- Utilisé dans un algo populaire de détection de visages
  [Viola & Jones 2001]

En résumé

Bagging:

- Technique de réduction de variance
- Moyenne des prédicteurs entraînés sur plusieurs variantes de D obtenues par rééchantillonnage Bootstrap.
- Très utile pour les classifieurs particulièrement sensibles aux données (e.g. arbres de décision profonds).

Forêts= bagging d'arbres + sélection aléatoire de dimensions d'entrée (features)

Boosting:

- Réduction du biais
- Combinaison de classifieurs faibles (faible capacité, biais élevé)
  ➪ classifieur fort (capacité plus élevée, biais réduit)
- Classifieurs faibles ajoutés un à un à une série, entraînés sur données repondérées en fonction des erreurs commises.
- Contrôle de capacité (donc de la variance) par arrêt prématuré: on arrête d'ajouter des classifieurs faibles
- Fonctionne très bien avec des arbres de décision peu profonds (voire un seul noeud: des «stumps»)