# Apprentissage automatique

## Combinaison de modèles - motivation

**Sujets:** types d'apprentissage

**RAPPEL**

- Il existe différents types d'apprentissage
  - apprentissage supervisé : il y a une cible à prédire

$$\mathcal{D} = \{(\mathbf{x}_1, t_1), \ldots , (\mathbf{x}_N, t_N)\}$$

  - apprentissage non-supervisé : cible n'est pas fournie

$$\mathcal{D} = \{\mathbf{x}_1, \ldots , \mathbf{x}_N\}$$

  - apprentissage par renforcement (non couvert dans ce cours)

# APPRENTISSAGE AUTOMATIQUE

**Sujets:** types d'apprentissage

**RAPPEL**

- Il existe différents types d'apprentissage
  - apprentissage supervisé : il y a une cible à prédire

$$\mathcal{D} = \{(\mathbf{x}_1, t_1), \; \ldots \;, (\mathbf{x}_N, t_N)\}$$

  - apprentissage non-supervisé : cible n'est pas fournie

$$\mathcal{D} = \{\mathbf{x}_1, \; \ldots \;, \mathbf{x}_N\}$$

  - apprentissage par renforcement (non couvert dans ce cours)

# Combinaison de Modèles

**Sujets:** ensemble, comité

- Pourquoi utiliser un seul modèle ?
  - un système combinant une multitude de modèles différents ne serait-il pas meilleur ?

- En pratique, la réponse presque toujours oui !
  - le résultat de la combinaison de plusieurs modèles est appelée **ensemble** ou **comité**

# Combinaison de Modèles

**Sujets:** ensemble, comité

- La façon la plus simple d'obtenir $M$ modèles est d'utiliser $M$ algorithmes d'apprentissage différents
  - pour $m = 1, \ldots, M$
    - entraîner un modèle $y_m(\mathbf{x})$ à l'aide du $m^{\text{e}}$ algorithme d'apprentissage
  - retourner le modèle ensemble (comité)
    - pour la régression :

$$y_{\text{COM}}(\mathbf{x}) = \frac{1}{M}\sum_{m=1}^{M} y_m(\mathbf{x})$$

    - pour la classification : $y_{\text{COM}}(\mathbf{x})$ retourne la classe ayant le plus de votes

# Combinaison de Modèles

**Sujets:** ensemble, comité

- Les $M$ algorithmes pourraient aussi être le même algorithme, mais avec $M$ choix d'hyper-paramètres différents
  - pour $m = 1, ..., M$
    - entraîner un modèle $y_m(\mathbf{x})$ à l'aide du $m^e$ algorithme d'apprentissage
  - retourner le modèle ensemble (comité)
    - pour la régression :

$$y_{\mathrm{COM}}(\mathbf{x}) = \frac{1}{M} \sum_{m=1}^{M} y_m(\mathbf{x})$$

    - pour la classification : $y_{\mathrm{COM}}(\mathbf{x})$ retourne la classe ayant le plus de votes

# COMBINAISON DE MODÈLES

**Sujets:** *bagging, boosting*

- Même avec un seul algorithme sans hyper-paramètres, on peut améliorer sa performance à l'aide d'un ensemble

  - ***Bagging* :** particulièrement approprié lorsque chaque modèle individuel a **beaucoup de capacité**

  - ***Boosting* :** particulièrement approprié lorsque chaque modèle individuel n'a **pas beaucoup de capacité**

# Apprentissage automatique

## Combinaison de modèles - *bagging*

# COMBINAISON DE MODÈLES

**Sujets:** *bagging, boosting*

- Même avec un seul algorithme, on peut améliorer sa performance à l'aide d'un ensemble

  - ***Bagging* :** particulièrement approprié lorsque chaque modèle individuel a **beaucoup de capacité**

  - ***Boosting* :** particulièrement approprié lorsque chaque modèle individuel n'a **pas beaucoup de capacité**

# COMBINAISON DE MODÈLES

**Sujets:** *bagging*, *boosting*

- Même avec un seul algorithme, on peut améliorer sa performance à l'aide d'un ensemble

  - ***Bagging* :** particulièrement approprié lorsque chaque modèle individuel a **beaucoup de capacité**

  - ***Boosting* :** particulièrement approprié lorsque chaque modèle individuel n'a **pas beaucoup de capacité**

# DÉCOMPOSITION BIAIS-VARIANCE

**Sujets:** biais, variance, bruit

**RAPPEL**

- On peut montrer que :

$$\mathbb{E}_{\mathcal{D}}\left[\mathbb{E}_{(\mathbf{x},t)}[L(t,y(\mathbf{x};\mathcal{D}))]\right] = (\text{bias})^2 + \text{variance} + \text{noise}$$

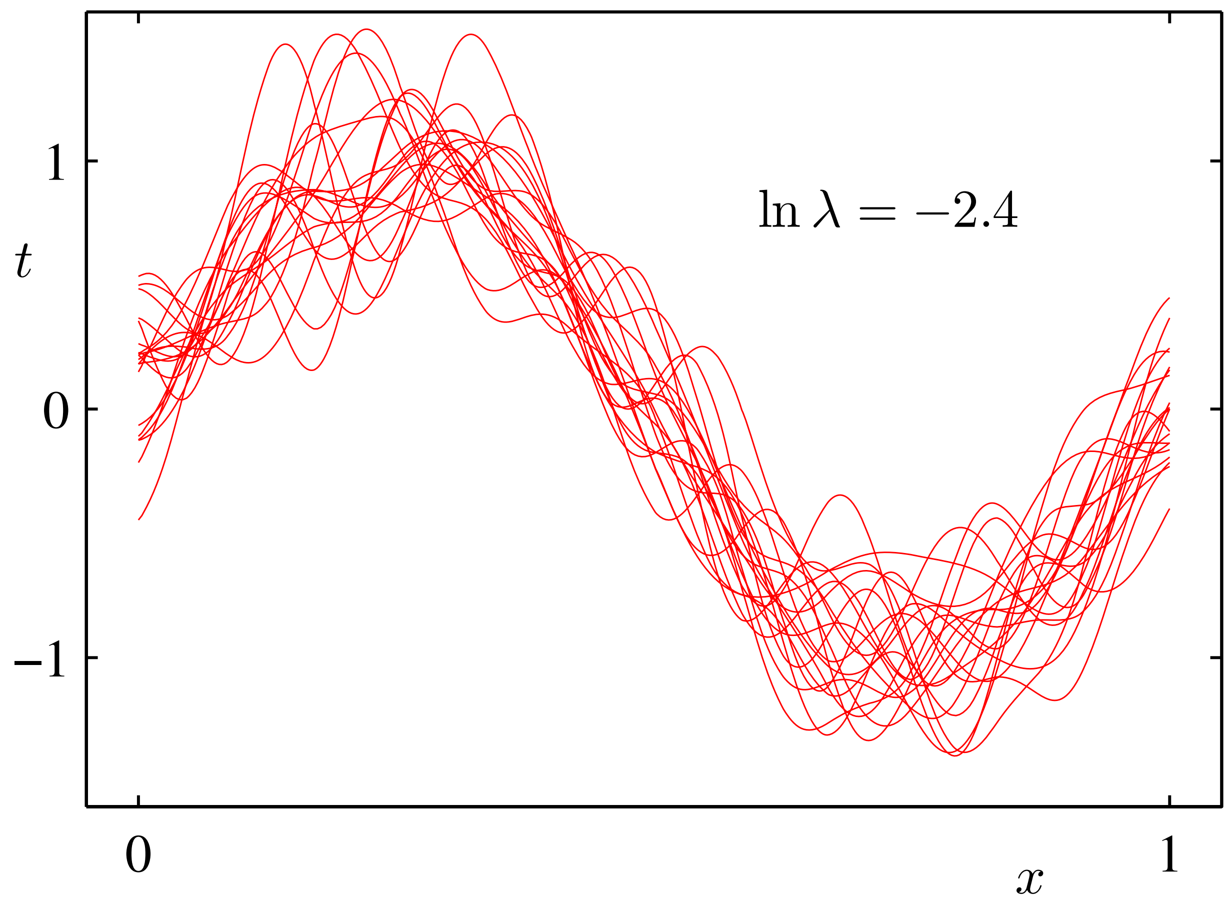
$$\begin{aligned}
(\text{bias})^2 &= \int \{\mathbb{E}_{\mathcal{D}}[y(\mathbf{x};\mathcal{D})] - h(\mathbf{x})\}^2 p(\mathbf{x})\,\mathrm{d}\mathbf{x} \\
\text{variance} &= \int \mathbb{E}_{\mathcal{D}}\left[\{y(\mathbf{x};\mathcal{D}) - \mathbb{E}_{\mathcal{D}}[y(\mathbf{x};\mathcal{D})]\}^2\right] p(\mathbf{x})\,\mathrm{d}\mathbf{x} \\
\text{noise} &= \int \{h(\mathbf{x}) - t\}^2 p(\mathbf{x},t)\,\mathrm{d}\mathbf{x}\,\mathrm{d}t
\end{aligned}$$

# DÉCOMPOSITION BIAIS-VARIANCE

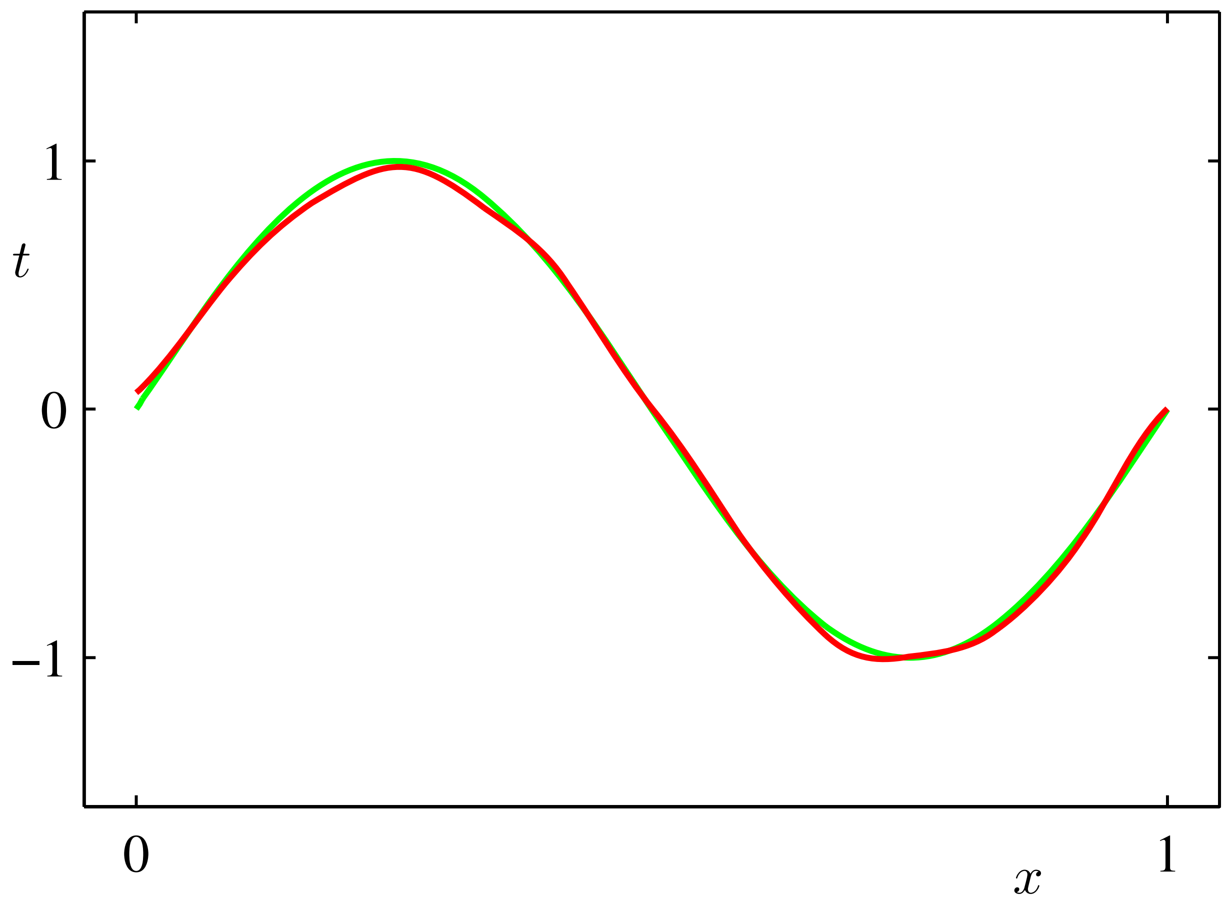
**Sujets:** réduction de variance

- Régression polynomiale de degré $25$

$100$ modèles entraînés sur
$100$ ensembles d'entraînement différents




Ensemble des $100$ modèles
vs. vrai modèle






HUGO LAROCHELLE

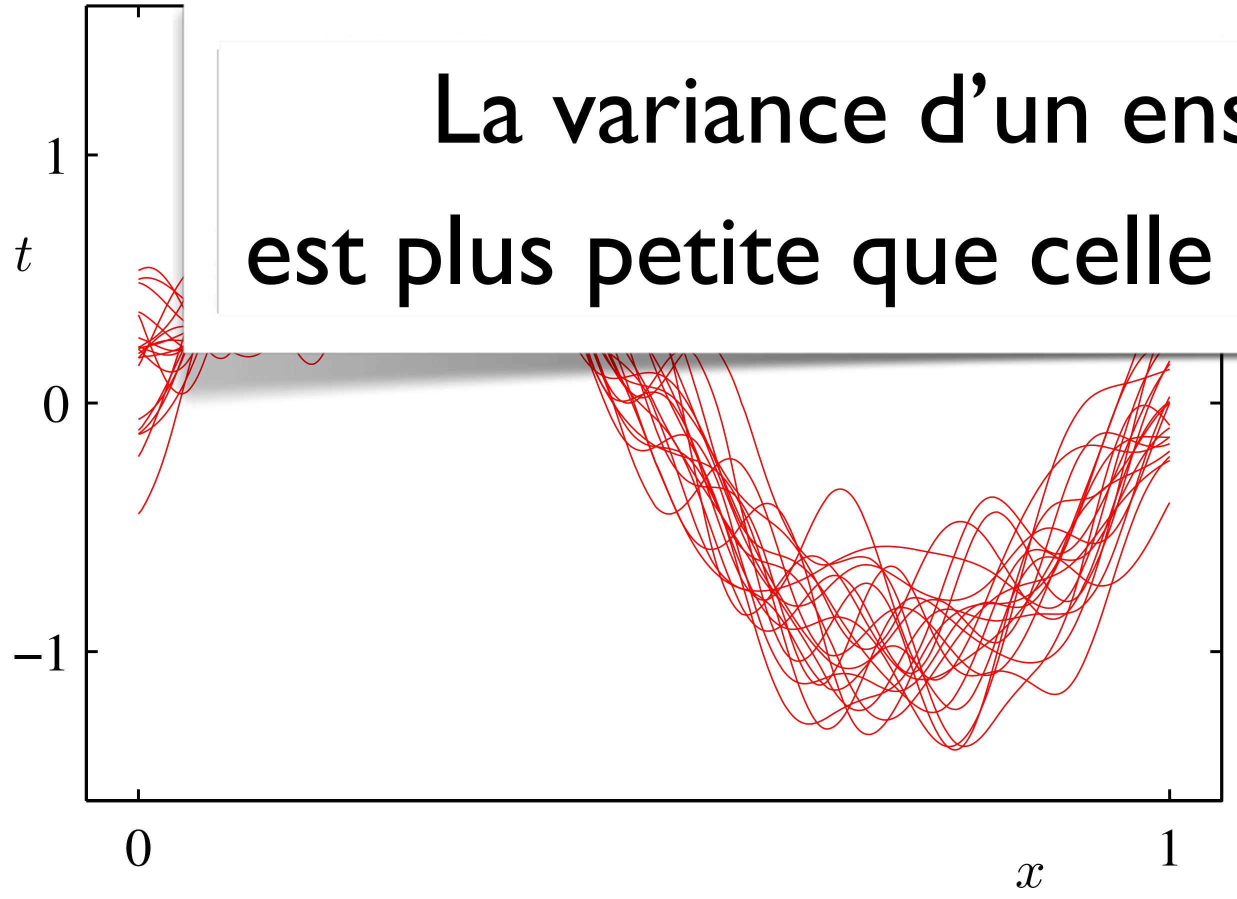
# DÉCOMPOSITION BIAIS-VARIANCE

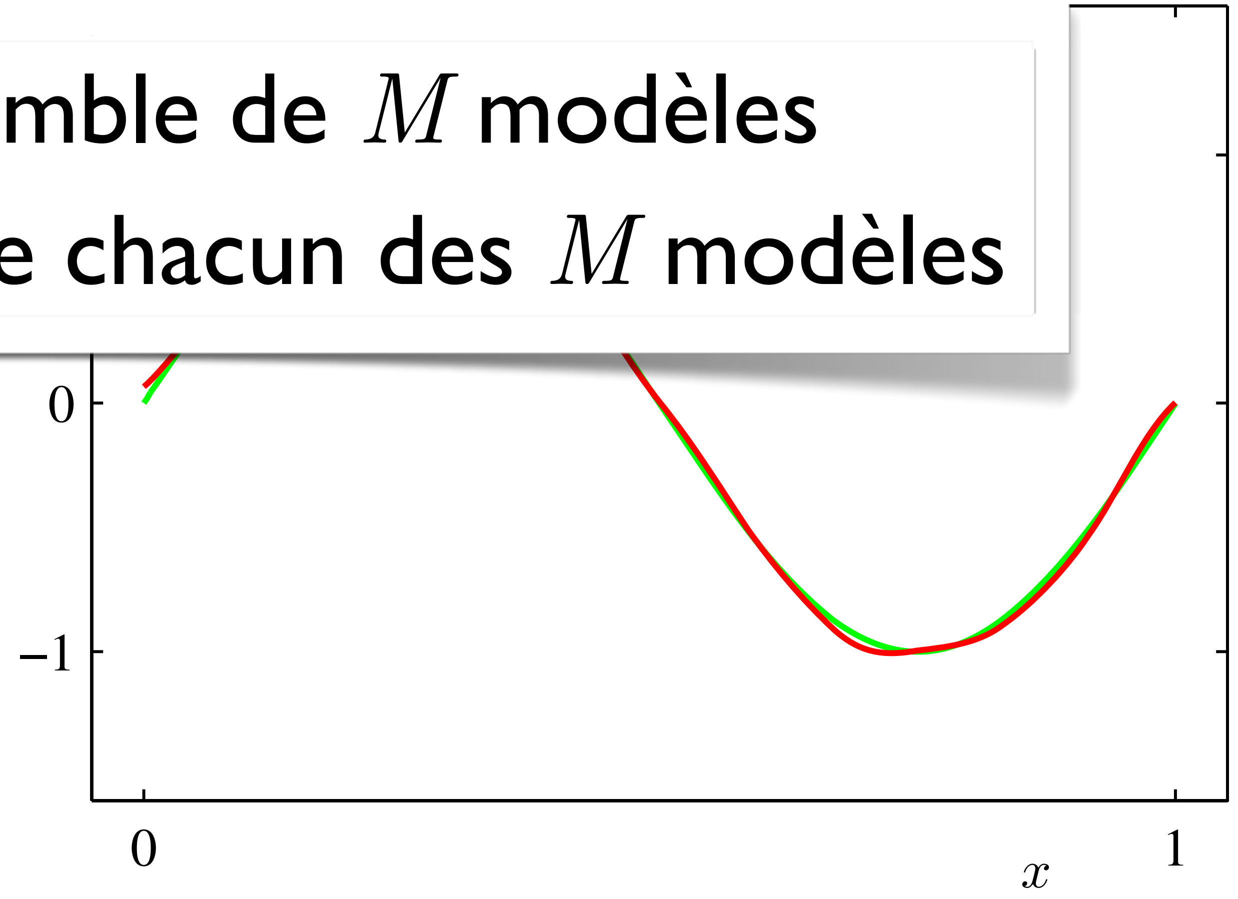
**Sujets:** réduction de variance

- Régression polynomiale de degré $25$

$100$ modèles entraînés sur
$100$ ensembles d'entraînement différents

Ensemble des $100$ modèles
vs. vrai modèle




La variance d'un ensemble de $M$ modèles
est plus petite que celle de chacun des $M$ modèles

# BOOTSTRAP

**Sujets:** *bootstrap*

- À part pour des données synthétiques, on ne peut pas générer sur demande des ensembles d'entraînement
- ***Bootstrap*** : on simule chaque collection de nouvelles données comme suit :
  - $\mathcal{D}_{\text{bootstrap}} \leftarrow \{\}$
  - pour $N$ itérations
    - choisir aléatoirement et uniformément un entier $n$ parmi $\{1,\ldots,N\}$
    - $\mathcal{D}_{\text{bootstrap}} \leftarrow \mathcal{D}_{\text{bootstrap}} \cup \{(\mathbf{x}_n, t_n)\}$
  - retourner $\mathcal{D}_{\text{bootstrap}}$

# BOOTSTRAP

**Sujets:** *bootstrap*

- À part pour des données synthétiques, on ne peut pas générer sur demande des ensembles d'entraînement
- ***Bootstrap*** : on simule chaque collection de nouvelles données comme suit :
  - $\mathcal{D}_{\text{bootstrap}} \leftarrow \{\}$
  - pour $N$ itérations
    - choisir aléatoirement et uniformément un entier $n$ parmi $\{1,\ldots,N\}$
    - $\mathcal{D}_{\text{bootstrap}} \leftarrow \mathcal{D}_{\text{bootstrap}} \cup \{(\mathbf{x}_n, t_n)\}$

    échantillonne $N$ exemples $\mathcal{D}$ avec remplacement
  - retourner $\mathcal{D}_{\text{bootstrap}}$

# BAGGING

**Sujets:** *bagging*

- ***Bagging* :** entraîne $M$ modèles avec un algorithme donné, sur $M$ ensembles de données *bootstrap*
  - pour $m = 1, \ldots, M$
    - génère un ensemble de données *bootstrap* $\mathcal{D}_{\mathrm{bootstrap}}$ à partir de $\mathcal{D}$
    - entraîner un modèle $y_m(\mathbf{x})$ sur $\mathcal{D}_{\mathrm{bootstrap}}$
  - retourner le modèle ensemble (comité)

$$y_{\mathrm{COM}}(\mathbf{x}) = \frac{1}{M} \sum_{m=1}^{M} y_m(\mathbf{x})$$

(régression)

# Apprentissage automatique

## Combinaison de modèles - propriétés du *bagging*

# BAGGING

**Sujets:** *bagging*

**RAPPEL**

- ***Bagging* :** entraîne $M$ modèles avec un algorithme donné, sur $M$ ensembles de données *bootstrap*
  - pour $m = 1, \ldots, M$
    - génère un ensemble de données *bootstrap* $\mathcal{D}_{\text{bootstrap}}$ à partir de $\mathcal{D}$
    - entraîner un modèle $y_m(\mathbf{x})$ sur $\mathcal{D}_{\text{bootstrap}}$
  - retourner le modèle ensemble (comité)

$$y_{\text{COM}}(\mathbf{x}) = \frac{1}{M}\sum_{m=1}^{M} y_m(\mathbf{x})$$

(régression)

# BAGGING

**Sujets:** analyse théorique de l'erreur

- On va analyser théoriquement l'erreur de l'ensemble
  - on va considérer le cas de la régression

- Soit $h(\mathbf{x})$ le vrai modèle à apprendre

- Alors, on peut noter chaque modèle $y_m(\mathbf{x})$ de l'ensemble

$$y_m(\mathbf{x}) = h(\mathbf{x}) + \epsilon_m(\mathbf{x})$$

# Bagging

**Sujets:** analyse théorique de l'erreur

- On va analyser théoriquement l'erreur de l'ensemble
  - on va considérer le cas de la régression

- Soit $h(\mathbf{x})$ le vrai modèle à apprendre

- Alors, on peut noter chaque modèle $y_m(\mathbf{x})$ de l'ensemble

$$y_m(\mathbf{x}) = h(\mathbf{x}) + \underbrace{\epsilon_m(\mathbf{x})}_{\text{erreur (signée) de } y_m(\mathbf{x})}$$

# Bagging

**Sujets:** analyse théorique de l'erreur

- L'erreur de généralisation de chaque $y_m(\mathbf{x})$ est donc

$$\mathbb{E}_{\mathbf{x}}\left[\{y_m(\mathbf{x}) - h(\mathbf{x})\}^2\right] = \mathbb{E}_{\mathbf{x}}\left[\epsilon_m(\mathbf{x})^2\right]$$

- L'erreur de l'ensemble des $y_m(\mathbf{x})$ est alors

$$\begin{aligned} E_{\mathrm{COM}} &= \mathbb{E}_{\mathbf{x}}\left[\left\{\frac{1}{M}\sum_{m=1}^{M} y_m(\mathbf{x}) - h(\mathbf{x})\right\}^2\right] \\ &= \mathbb{E}_{\mathbf{x}}\left[\left\{\frac{1}{M}\sum_{m=1}^{M} \epsilon_m(\mathbf{x})\right\}^2\right] \end{aligned}$$

# BAGGING

**Sujets:** analyse théorique de l'erreur

- On peut montrer que

$$E_{\text{COM}} = \frac{1}{M} E_{\text{AV}} \qquad \textbf{où} \qquad E_{\text{AV}} = \frac{1}{M} \sum_{m=1}^{M} \mathbb{E}_{\mathbf{x}} \left[ \epsilon_m(\mathbf{x})^2 \right]$$

si on suppose que les erreurs des $y_m(\mathbf{x})$ sont d'espérance $0$ et ne sont pas corrélées (sont «différentes») :

$$\mathbb{E}_{\mathbf{x}} \left[ \epsilon_m(\mathbf{x}) \epsilon_l(\mathbf{x}) \right] = 0, \qquad m \neq l$$

# BAGGING

**Sujets:** analyse théorique de l'erreur

- On peut montrer que

$$E_{\text{COM}} = \frac{1}{M} E_{\text{AV}} \qquad \text{où} \qquad E_{\text{AV}} = \overbrace{\frac{1}{M} \sum_{m=1}^{M} \mathbb{E}_{\mathbf{x}} \left[ \epsilon_m(\mathbf{x})^2 \right]}^{\text{moyenne des erreurs des } y_m(\mathbf{x})}$$

si on suppose que les erreurs des $y_m(\mathbf{x})$ sont d'espérance $0$ et ne sont pas corrélées (sont «différentes») :

$$\mathbb{E}_{\mathbf{x}} \left[ \epsilon_m(\mathbf{x}) \epsilon_l(\mathbf{x}) \right] = 0, \qquad m \neq l$$

# BAGGING

**Sujets:** analyse théorique de l'erreur

- On pourrait réduire l'erreur d'un facteur $1/M$ !
- Par contre, la supposition d'erreurs non-corrélées n'est pas vraie en pratique
  - les $y_m(\mathbf{x})$ font souvent le «même genre» d'erreurs
- On peut quand même démontrer que l'erreur $E_{\mathrm{COM}}$ de l'ensemble ne va pas dépasser l'erreur moyenne $E_{\mathrm{AV}}$ des $y_m(\mathbf{x})$

# Apprentissage automatique

## Combinaison de modèles - *boosting*

# COMBINAISON DE MODÈLES

**Sujets:** *bagging, boosting*

- Même avec un seul algorithme, on peut améliorer sa performance à l'aide d'un ensemble
  - ***Bagging* :** particulièrement approprié lorsque chaque modèle individuel a **beaucoup de capacité**
  - ***Boosting* :** particulièrement approprié lorsque chaque modèle individuel n'a **pas beaucoup de capacité**

# COMBINAISON DE MODÈLES

**Sujets:** *bagging, boosting*

- Même avec un seul algorithme, on peut améliorer sa performance à l'aide d'un ensemble

  - ***Bagging* :** particulièrement approprié lorsque chaque modèle individuel a **beaucoup de capacité**

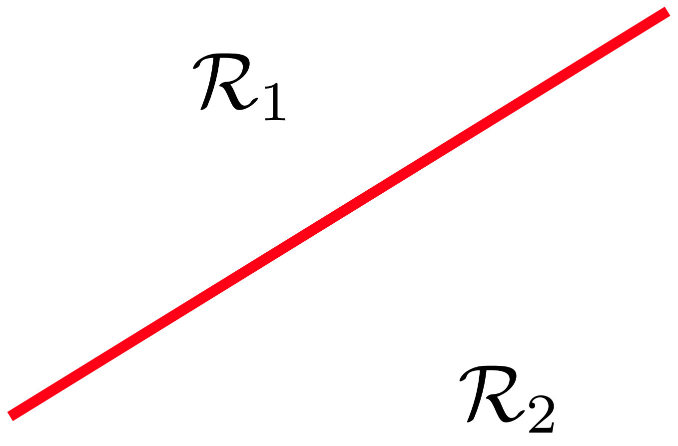
  - ***Boosting* :** particulièrement approprié lorsque chaque modèle individuel n'a **pas beaucoup de capacité**

# CLASSIFICATION

**Sujets:** classification binaire, séparabilité linéaire

RAPPEL

- Cas spécial : classification binaire
  - classe $\mathcal{C}_1$ correspond à $t = 1$
  - classe $\mathcal{C}_2$ correspond à $t = 0$ (ou $t =$ -1)



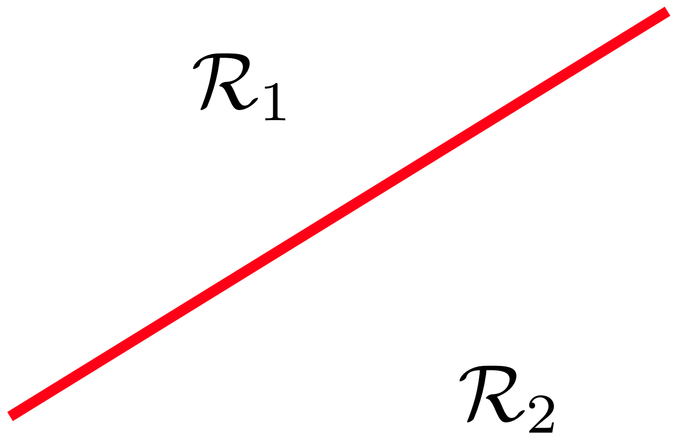

- Cas spécial : classification linéaire
  - la surface de décision entre chaque paire de régions de décision est linéaire, i.e. un hyperplan (droite pour $D$=2)
  - on dit qu'un problème est **linéairement séparable** si une surface linéaire permet de classifier parfaitement

# CLASSIFICATION

**Sujets:** classification binaire, séparabilité linéaire

RAPPEL

- Cas spécial : classification binaire
  - classe $\mathcal{C}_1$ correspond à $t = 1$
  - classe $\mathcal{C}_2$ correspond à $t = 0$ (ou $t =$ -1)




- Cas spécial : classification linéaire
  - la surface de décision entre chaque paire de régions de décision est linéaire, i.e. un hyperplan (droite pour $D$=2)
  - on dit qu'un problème est **linéairement séparable** si une surface linéaire permet de classifier parfaitement

# BOOSTING

**Sujets:** *weak learner, boosting*

- Si les modèles n'ont pas beaucoup de capacité, leur variance sera déjà petite
  - on appelle de tels modèles des *weak learners*
- Peut-être devraient-ils plutôt se diviser le travail
- C'est le principe derrière le ***boosting***
  - on entraîne les modèles en séquence
  - chaque modèle se concentre sur les exemples mal modélisés par les modèles précédents
    - pour le $m^e$ modèle, chaque exemple $(\mathbf{x}_n, t_n)$ aura un poids $w_n^{(m)}$

# BOOSTING

**Sujets:** *boosting*

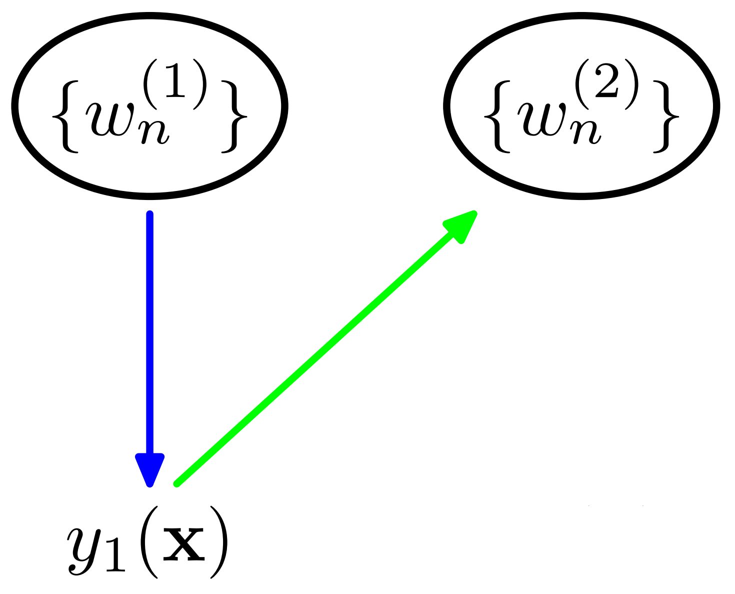
- Illustration du *boosting*

$$\begin{tikzcd} \{w_n^{(1)}\} \arrow[d] \\ y_1(\mathbf{x}) \end{tikzcd}$$

# BOOSTING

**Sujets:** *boosting*

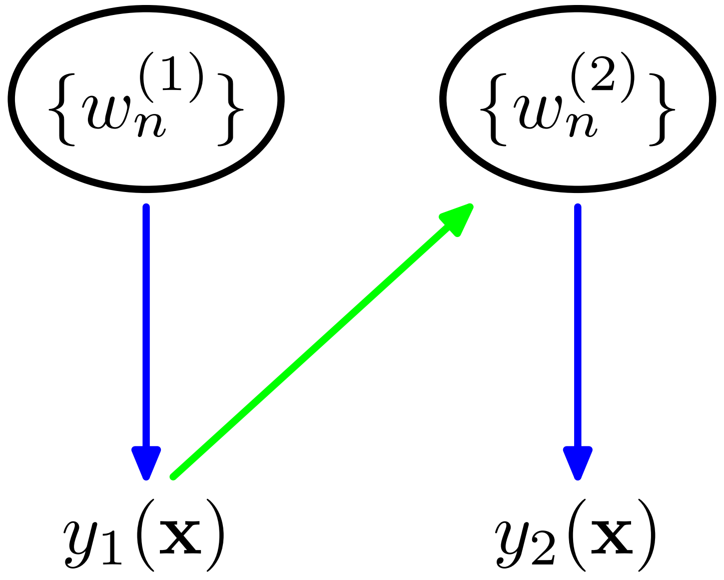
- Illustration du *boosting*



# Boosting

**Sujets:** *boosting*

- Illustration du *boosting*



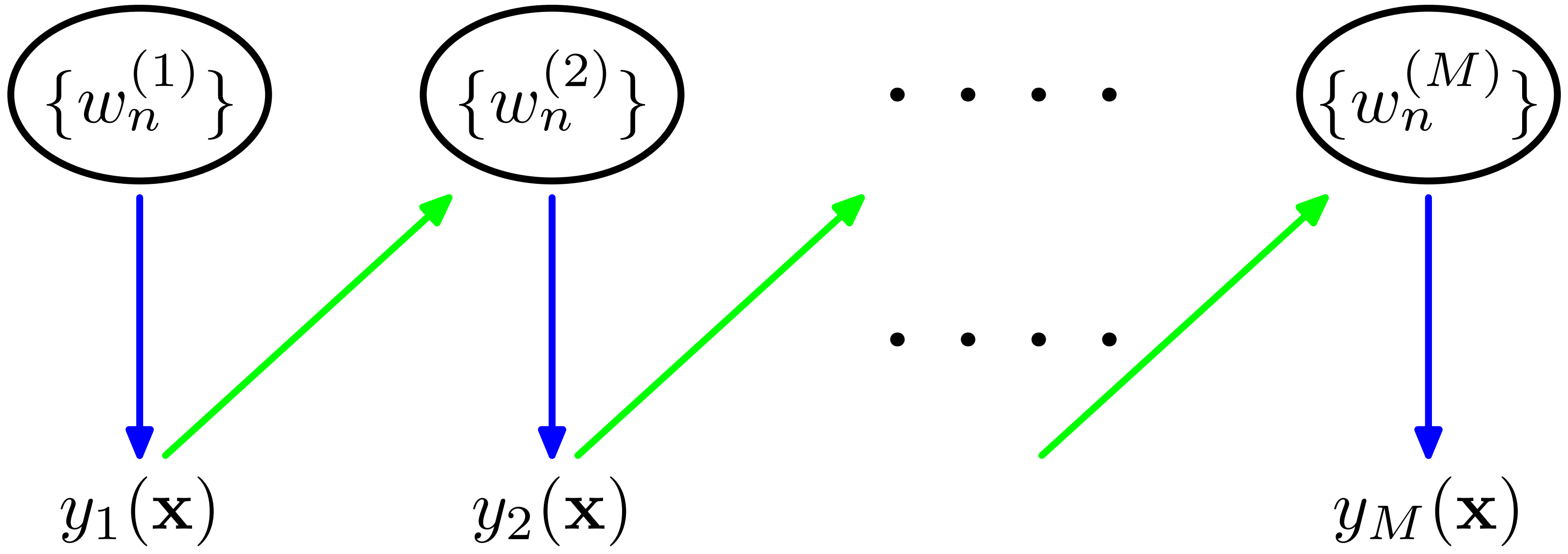



Hugo Larochelle

# BOOSTING

**Sujets:** *boosting*

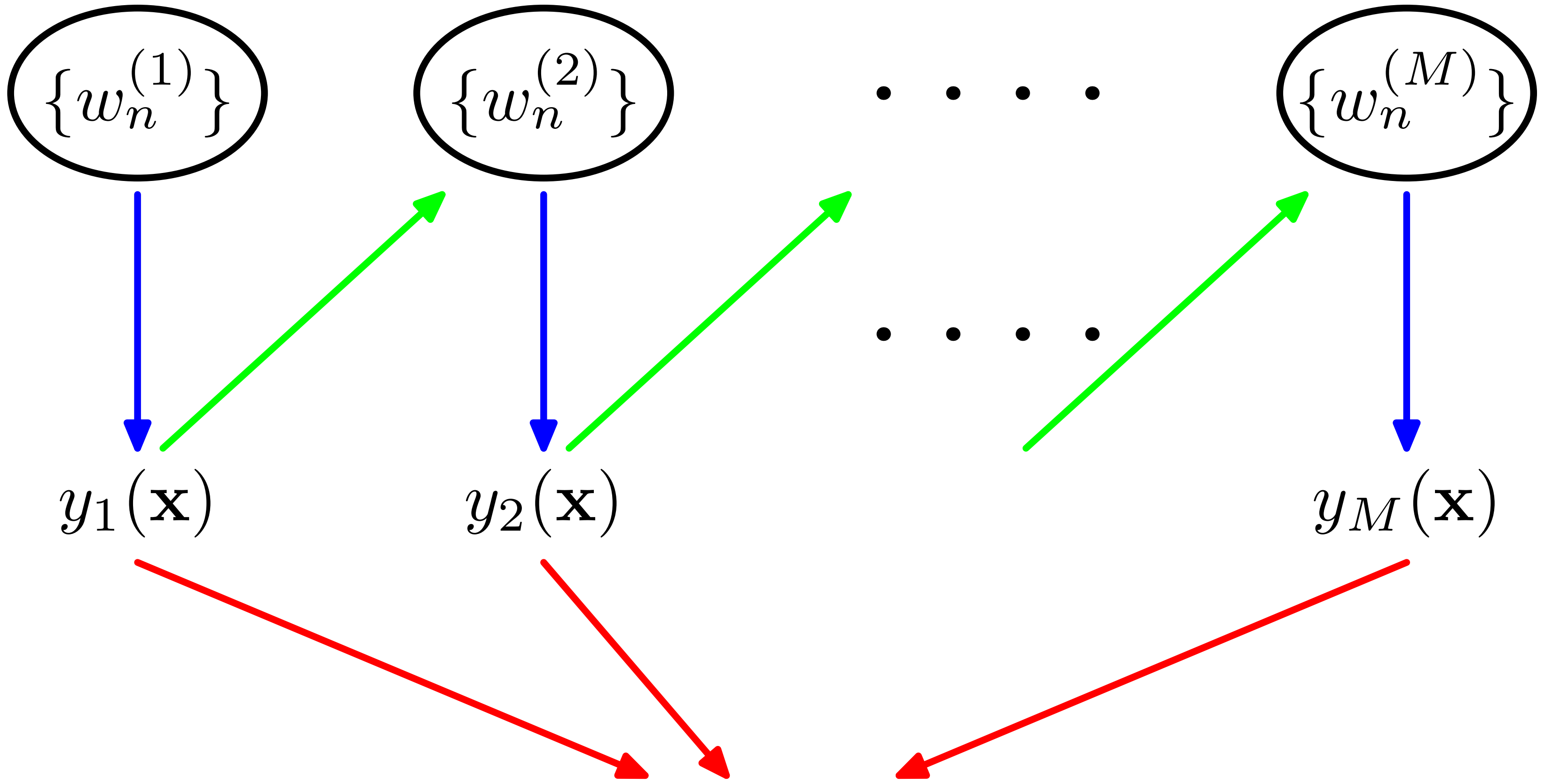
- Illustration du *boosting*



# BOOSTING

**Sujets:** *boosting*

- Illustration du *boosting*




$$Y_M(\mathbf{x}) = \operatorname{sign}\left(\sum_m^M \alpha_m y_m(\mathbf{x})\right) \qquad \text{(classification)}$$

# AdaBoost

**Sujets:** AdaBoost

- Pseudocode de AdaBoost (*boosting* pour la classification)

1. Initialize the data weighting coefficients $\{w_n\}$ by setting $w_n^{(1)} = 1/N$ for $n = 1, \ldots, N$.

# AdaBoost

**Sujets:** AdaBoost

- Pseudocode de AdaBoost (*boosting* pour la classification)

2. For $m = 1, \ldots, M$:
   (a) Fit a classifier $y_m(\mathbf{x})$ to the training data by minimizing the weighted error function

$$J_m = \sum_{n=1}^{N} w_n^{(m)} I(y_m(\mathbf{x}_n) \neq t_n)$$

   where $I(y_m(\mathbf{x}_n) \neq t_n)$ is the indicator function and equals 1 when $y_m(\mathbf{x}_n) \neq t_n$ and 0 otherwise.

   (b) ...

   (c) ...

# AdaBoost

**Sujets:** AdaBoost

- Pseudocode de AdaBoost (*boosting* pour la classification)

2. For $m = 1, \ldots, M$:

   (a) ...

   (b) Evaluate the quantities

$$\epsilon_m = \frac{\displaystyle\sum_{n=1}^{N} w_n^{(m)} I(y_m(\mathbf{x}_n) \neq t_n)}{\displaystyle\sum_{n=1}^{N} w_n^{(m)}}$$

   and then use these to evaluate

$$\alpha_m = \ln\left\{\frac{1-\epsilon_m}{\epsilon_m}\right\}.$$

   (c) ...

# AdaBoost

**Sujets:** AdaBoost

- Pseudocode de AdaBoost (*boosting* pour la classification)

2. For $m = 1, \ldots, M$:
   (a) …
   (b) …
   (c) Update the data weighting coefficients

$$w_n^{(m+1)} = w_n^{(m)} \exp\left\{\alpha_m I(y_m(\mathbf{x}_n) \neq t_n)\right\}$$

# ADABOOST

**Sujets:** AdaBoost

- Pseudocode de AdaBoost (*boosting* pour la classification)

3. Make predictions using the final model, which is given by

$$Y_M(\mathbf{x}) = \operatorname{sign}\left(\sum_{m=1}^{M} \alpha_m y_m(\mathbf{x})\right).$$

# ADABOOST

**Sujets:** AdaBoost

- Pseudocode de AdaBoost (*boosting* pour la classification)

3. Make predictions using the final model, which is given by

$$Y_M(\mathbf{x}) = \operatorname{sign}\left( \sum_{m=1}^{M} \alpha_m \underbrace{y_m(\mathbf{x})}_{} \right).$$

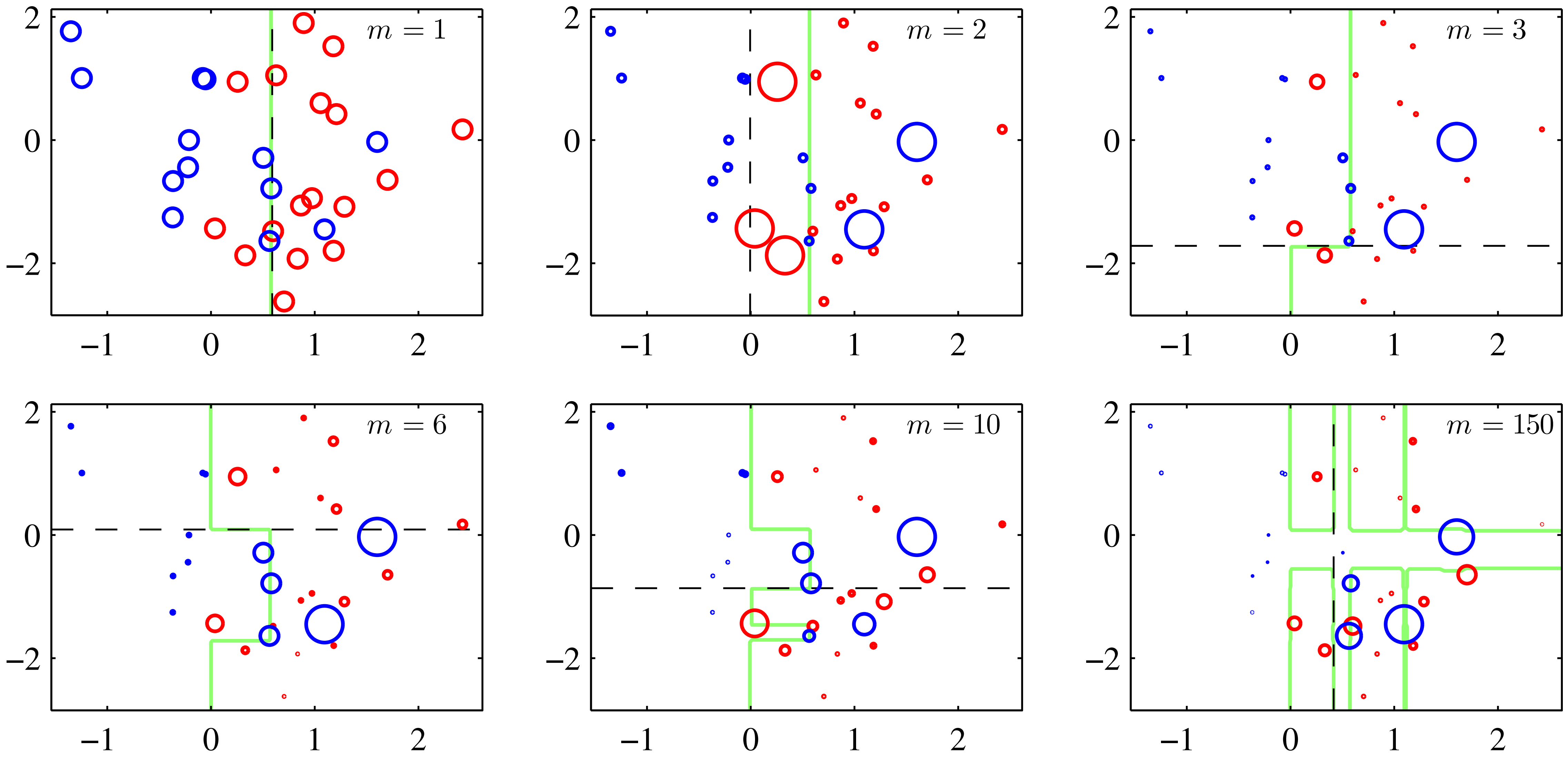
$$y_m(\mathbf{x}) \in \{\text{-}1,1\}$$

( $y_m(\mathbf{x})$ est la fonction discriminante )

# ADABOOST

**Sujets:** AdaBoost

- Exemple : *weak learners* classifient sur une seule dimension



# AdaBoost

**Sujets:** erreur de classification pondérée

- Chaque modèle optimise l'erreur de classification pondérée

$$J_m = \sum_{n=1}^{N} w_n^{(m)} I(y_m(\mathbf{x}_n) \neq t_n)$$

  - la plupart des algorithmes peuvent être adaptés au cas pondéré
  - sinon, une approche simple est d'entraîner sur une version *bootstrap* de l'ensemble d'entraînement, mais où la probabilité d'insérer $(\mathbf{x}_n, t_n)$ est $w_n^{(m)}$

# BOOTSTRAP

**Sujets:** *bootstrap*

**RAPPEL**

- À part pour des données synthétiques, on ne peut pas générer sur demande des ensembles d'entraînement
- ***Bootstrap*** : on simule chaque collection de nouvelles données comme suit :
  - $\mathcal{D}_{\text{bootstrap}} \leftarrow \{\}$
  - pour $N$ itérations
    - choisir aléatoirement et uniformément un entier $n$ parmi $\{1,...,N\}$
    - $\mathcal{D}_{\text{bootstrap}} \leftarrow \mathcal{D}_{\text{bootstrap}} \cup \{(\mathbf{x}_n, t_n)\}$
  - retourner $\mathcal{D}_{\text{bootstrap}}$

# BOOTSTRAP

**Sujets:** *bootstrap*

RAPPEL

- À part pour des données synthétiques, on ne peut pas générer sur demande des ensembles d'entraînement
- ***Bootstrap*** : on simule chaque collection de nouvelles données comme suit :
  - $\mathcal{D}_{\text{bootstrap}} \leftarrow \{\}$
  - pour $N$ itérations
    - choisir aléatoirement et ~~uniformément~~ avec probablités proportionnelles à $\{w_n{}^{(m)}\}$ un entier $n$ parmi $\{1,\ldots,N\}$
    - $\mathcal{D}_{\text{bootstrap}} \leftarrow \mathcal{D}_{\text{bootstrap}} \cup \{(\mathbf{x}_n, t_n)\}$
  - retourner $\mathcal{D}_{\text{bootstrap}}$

# Apprentissage automatique

## Combinaison de modèles - propriétés du *boosting*

# Boosting

**Sujets:** *weak learner, boosting*

**RAPPEL**

- Si les modèles n'ont pas beaucoup de capacité, leur variance sera déjà petite
  - on appelle de tels modèles des *weak learners*
- Peut-être devraient-ils se plutôt se diviser le travail
- C'est le principe derrière le ***boosting***
  - on entraîne les modèles en séquence
  - chaque modèle se concentre sur les exemples mal modélisés par les modèles précédents
    - pour le $m$e modèle, chaque exemple $(\mathbf{x}_n, t_n)$ aura un poids $w_n^{(m)}$

# AdaBoost

**Sujets:** dérivation théorique d'Adaboost

- On peut voir AdaBoost comme optimisant la somme d'une perte exponentielle

$$E = \sum_{n=1}^{N} \exp\{-t_n f_m(\mathbf{x}_n)\}$$

où $f_m(\mathbf{x}_n)$ est le score pour la classe $t_n{=}1$ :

$$f_m(\mathbf{x}) = \frac{1}{2} \sum_{l=1}^{m} \alpha_l y_l(\mathbf{x})$$

# ADABOOST

**Sujets:** dérivation théorique d'Adaboost

- AdaBoost optimise ces pertes séquentiellement
  - à partir de l'ensemble $f_{m\text{-}1}(\mathbf{x})$ de l'itération précédente, on veut y ajouter un nouveau modèle $y_m(\mathbf{x})$

$$
\begin{aligned}
E &= \sum_{n=1}^{N} \exp\left\{-t_n f_{m-1}(\mathbf{x}_n) - \frac{1}{2} t_n \alpha_m y_m(\mathbf{x}_n)\right\} \\
&= \sum_{n=1}^{N} w_n^{(m)} \exp\left\{-\frac{1}{2} t_n \alpha_m y_m(\mathbf{x}_n)\right\}
\end{aligned}
$$

# AdaBoost

**Sujets:** dérivation théorique d'Adaboost

- AdaBoost optimise ces pertes séquentiellement
  - le meilleur $y_m(\mathbf{x})$ à ajouter sera celui qui optimise $E$, qui équivaut à optimiser $J_m$

| | |
|---|---|
| $\mathcal{T}_m$ | exemples **bien** classifiés par $y_m(\mathbf{x})$ |
| $\mathcal{M}_m$ | exemples **mal** classifiés par $y_m(\mathbf{x})$ |

$$
\begin{aligned}
E &= e^{-\alpha_m/2} \sum_{n \in \mathcal{T}_m} w_n^{(m)} + e^{\alpha_m/2} \sum_{n \in \mathcal{M}_m} w_n^{(m)} \\
&= (e^{\alpha_m/2} - e^{-\alpha_m/2}) \sum_{n=1}^{N} w_n^{(m)} I(y_m(\mathbf{x}_n) \neq t_n) + e^{-\alpha_m/2} \sum_{n=1}^{N} w_n^{(m)}
\end{aligned}
$$

# AdaBoost

**Sujets:** dérivation théorique d'Adaboost

- AdaBoost optimise ces pertes séquentiellement
  - le meilleur $y_m(\mathbf{x})$ à ajouter sera celui qui optimise $E$, qui équivaut à optimiser $J_m$

| | |
|---|---|
| $\mathcal{T}_m$ | exemples **bien** classifiés par $y_m(\mathbf{x})$ |
| $\mathcal{M}_m$ | exemples **mal** classifiés par $y_m(\mathbf{x})$ |

$$\begin{aligned} E &= e^{-\alpha_m/2} \sum_{n \in \mathcal{T}_m} w_n^{(m)} + e^{\alpha_m/2} \sum_{n \in \mathcal{M}_m} w_n^{(m)} \\ &= (e^{\alpha_m/2} - e^{-\alpha_m/2}) \boxed{\sum_{n=1}^{N} w_n^{(m)} I(y_m(\mathbf{x}_n) \neq t_n)} + e^{-\alpha_m/2} \sum_{n=1}^{N} w_n^{(m)} \end{aligned}$$

# AdaBoost

**Sujets:** dérivation théorique d'Adaboost

- AdaBoost optimise ces pertes séquentiellement
  - on trouve le poids $\alpha_m$ de $y_m(\mathbf{x})$ en optimisant aussi $E$, mais cette fois par rapport à $\alpha_m$

| | |
|---|---|
| $\mathcal{T}_m$ | exemples **bien** classifiés par $y_m(\mathbf{x})$ |
| $\mathcal{M}_m$ | exemples **mal** classifiés par $y_m(\mathbf{x})$ |

$$\begin{aligned} E &= e^{-\alpha_m/2} \sum_{n \in \mathcal{T}_m} w_n^{(m)} + e^{\alpha_m/2} \sum_{n \in \mathcal{M}_m} w_n^{(m)} \\ &= (e^{\alpha_m/2} - e^{-\alpha_m/2}) \boxed{\sum_{n=1}^{N} w_n^{(m)} I(y_m(\mathbf{x}_n) \neq t_n)} + e^{-\alpha_m/2} \sum_{n=1}^{N} w_n^{(m)} \end{aligned}$$

# AdaBoost

**Sujets:** dérivation théorique d'Adaboost

- AdaBoost optimise ces pertes séquentiellement
  - on trouve le poids $\alpha_m$ de $y_m(\mathbf{x})$ en optimisant aussi $E$, mais cette fois par rapport à $\alpha_m$

| | |
|---|---|
| $\mathcal{T}_m$ | exemples **bien** classifiés par $y_m(\mathbf{x})$ |
| $\mathcal{M}_m$ | exemples **mal** classifiés par $y_m(\mathbf{x})$ |

$$\begin{aligned}E &= e^{-\alpha_m/2} \sum_{n \in \mathcal{T}_m} w_n^{(m)} + e^{\alpha_m/2} \sum_{n \in \mathcal{M}_m} w_n^{(m)} \\ &= (e^{\alpha_m/2} - e^{-\alpha_m/2}) \sum_{n=1}^{N} w_n^{(m)} I(y_m(\mathbf{x}_n) \neq t_n) + e^{-\alpha_m/2} \sum_{n=1}^{N} w_n^{(m)}\end{aligned}$$

# ADABOOST

**Sujets:** dérivation théorique d'Adaboost

- AdaBoost optimise ces pertes séquentiellement
  - on peut montrer que la solution est $\alpha_m = \ln \left\{ \frac{1-\epsilon_m}{\epsilon_m} \right\}$

| | |
|---|---|
| $\mathcal{T}_m$ | exemples **bien** classifiés par $y_m(\mathbf{x})$ |
| $\mathcal{M}_m$ | exemples **mal** classifiés par $y_m(\mathbf{x})$ |

$$
\begin{aligned}
E &= e^{-\alpha_m/2} \sum_{n \in \mathcal{T}_m} w_n^{(m)} + e^{\alpha_m/2} \sum_{n \in \mathcal{M}_m} w_n^{(m)} \\
&= (e^{\alpha_m/2} - e^{-\alpha_m/2}) \sum_{n=1}^{N} w_n^{(m)} I(y_m(\mathbf{x}_n) \neq t_n) + e^{-\alpha_m/2} \sum_{n=1}^{N} w_n^{(m)}
\end{aligned}
$$

# AdaBoost

**Sujets:** dérivation théorique d'Adaboost

- AdaBoost optimise ces pertes séquentiellement
  - à la prochaine itération, le poids de chaque exemple devra être

$$w_n^{(m+1)} = w_n^{(m)} \exp\left\{-\frac{1}{2} t_n \alpha_m y_m(\mathbf{x}_n)\right\}$$

  - étant donné que $t_n y_m(\mathbf{x}_n) = 1 - 2I(y_m(\mathbf{x}_n) \neq t_n)$, alors

$$w_n^{(m+1)} = w_n^{(m)} \exp(-\alpha_m/2) \exp\{\alpha_m I(y_m(\mathbf{x}_n) \neq t_n)\}$$

où on peut ignorer $\exp(-\alpha_m/2)$ puisqu'il ne dépend pas de $n$

# AdaBoost

**Sujets:** dérivation théorique d'Adaboost

- AdaBoost optimise ces pertes séquentiellement
  - à la prochaine itération, le poids de chaque exemple devra être

$$w_n^{(m+1)} = w_n^{(m)} \exp\left\{ -\frac{1}{2} t_n \alpha_m y_m(\mathbf{x}_n) \right\}$$

  - étant donné que $t_n y_m(\mathbf{x}_n) = 1 - 2I(y_m(\mathbf{x}_n) \neq t_n)$, alors

$$w_n^{(m+1)} = w_n^{(m)} \cancel{\exp(-\alpha_m/2)} \exp\left\{ \alpha_m I(y_m(\mathbf{x}_n) \neq t_n) \right\}$$

où on peut ignorer $\exp(-\alpha_m/2)$ puisqu'il ne dépend pas de $n$

# ADABOOST

**Sujets:** généralisation d'AdaBoost

- Ainsi, on peut généraliser AdaBoost à d'autres problèmes que la classification binaire en

1. utilisant une perte adaptée à notre problème, autre que la perte exponentielle (p. ex. la différence au carré en régression)

2. suivant les mêmes étapes de dérivation d'Adaboost pour trouver comment entraîner chaque $y_m(\mathbf{x}_n)$ à partir de $f_{m\text{-}1}(\mathbf{x}_n)$ et comment obtenir $\alpha_m$

# Apprentissage automatique

## Combinaison de modèles - résumé

# Ensemble

**Sujets:** résumé de la création d'ensemble

- Modèle
  - régression : la cible est mieux prédite par une moyenne de plusieurs modèles
  - classification : la cible est mieux prédite par le vote majoritaire de plusieurs modèles
- Entraînement :
  - exécute $M$ algorithmes d'apprentissage différents
- Prédiction : $y_{\mathrm{COM}}(\mathbf{x}) = \frac{1}{M}\sum_{m=1}^{M} y_m(\mathbf{x})$ (régression)

# Bagging

**Sujets:** résumé du *bagging*

- Modèle :
  - on suppose qu'on a le bon modèle (algorithme d'apprentissage) mais pas assez de données pour bien l'entraîner sans sur-apprentissage (trop de variance)
- Entraînement : réduction de variance
  - génère $M$ ensembles de données *bootstrap* et entraîne un modèle sur chacun, avec le même algorithme d'apprentissage
- Prédiction : $y_{\mathrm{COM}}(\mathbf{x}) = \frac{1}{M}\sum_{m=1}^{M} y_m(\mathbf{x})$ (régression)

# ADABOOST

**Sujets:** résumé de AdaBoost

- Modèle :
  - un seul modèle est trop faible et il vaut mieux en entraîner plusieurs, de façon synchronisée et où chaque modèle a un poids différent
- Entraînement : minimise une perte exponentielle
  - on entraîne $M$ modèles en séquence, où chaque modèle se concentre sur les exemples mal classifiés par les modèles précédents
- Prédiction : $Y_M(\mathbf{x}) = \operatorname{sign}\left(\sum_{m=1}^{M} \alpha_m y_m(\mathbf{x})\right)$

# COMBINAISON DE MODÈLES

**Sujets:** arbre de décisions

- Il existe plusieurs autres types de combinaisons
    - **arbre de décision** (voir section 14.4)
        - on utilise un seul modèle à la fois pour prédire la cible à partir de $\mathbf{x}$
        - le choix du modèle change en fonction de $\mathbf{x}$ et est déterminé par une série de questions structurées en arbre
    - **mélange d'experts** (voir section 14.5.3)
        - on utilise une moyenne pondérée de modèles (appelés «experts»)
        - le poids de chaque expert varie en fonction de $\mathbf{x}$