Maximum Margin Interval Trees

Alexandre Drouin[1], Toby Dylan Hocking[2], François Laviolette[1]

[1]Département d'informatique et de génie logiciel, Université Laval
[2]Centre d'innovation Génome Québec et Université McGill, Université McGill

Université Laval
Sainte-Foy, Québec
24 mai 2017

McGill

Plan

1. Introduction
2. L'algorithme Maximum Margin Interval Tree
3. Résultats
4. Conclusion

Introduction

Un problème d'apprentissage supervisé

Ensemble de données

$$\mathcal{S} \stackrel{\text{def}}{=} \{(\mathbf{x}_1, \mathbf{y}_1), (\mathbf{x}_2, \mathbf{y}_2), ..., (\mathbf{x}_n, \mathbf{y}_n))\} \sim D^n$$

$\mathbf{x} \in \mathbb{R}^p$ est un vecteur de caractéristiques

$\mathbf{y}_i \stackrel{\text{def}}{=} \left[\underline{y_i}, \overline{y_i}\right]$, avec $\underline{y_i}, \overline{y_i} \in \overline{\mathbb{R}}$ et $\underline{y_i} < \overline{y_i}$, est un intervalle

D est une distribution inconnue qui génère les données

Données censurées

- Censure à droite : $\mathbf{y} = [\underline{y_i}, +\infty)$
- Censure à gauche : $\mathbf{y} = (-\infty, \overline{y_i}]$
- Censure par intervalle : $\mathbf{y} = [\underline{y_i}, \overline{y_i}]$

Un problème d'apprentissage supervisé

Ensemble de données

$$\mathcal{S} \stackrel{\text{def}}{=} \{(\mathbf{x}_1, \mathbf{y}_1), (\mathbf{x}_2, \mathbf{y}_2), ..., (\mathbf{x}_n, \mathbf{y}_n))\} \sim D^n$$

$\mathbf{x} \in \mathbb{R}^p$ est un vecteur de caractéristiques

$\mathbf{y}_i \stackrel{\text{def}}{=} \left[\underline{y_i}, \overline{y_i}\right]$, avec $\underline{y_i}, \overline{y_i} \in \overline{\mathbb{R}}$ et $\underline{y_i} < \overline{y_i}$, est un intervalle

D est une distribution inconnue qui génère les données

Données censurées

- Censure à droite : $\mathbf{y} = [\underline{y_i}, +\infty)$
- Censure à gauche : $\mathbf{y} = (-\infty, \overline{y_i}]$
- Censure par intervalle : $\mathbf{y} = [\underline{y_i}, \overline{y_i}]$

Un problème d'apprentissage supervisé

Ensemble de données

$$\mathcal{S} \stackrel{\text{def}}{=} \{(\mathbf{x}_1, \mathbf{y}_1), (\mathbf{x}_2, \mathbf{y}_2), ..., (\mathbf{x}_n, \mathbf{y}_n))\} \sim D^n$$

$\mathbf{x} \in \mathbb{R}^p$ est un vecteur de caractéristiques

$\mathbf{y}_i \stackrel{\text{def}}{=} \left[\underline{y_i}, \overline{y_i}\right]$, avec $\underline{y_i}, \overline{y_i} \in \overline{\mathbb{R}}$ et $\underline{y_i} < \overline{y_i}$, est un intervalle

D est une distribution inconnue qui génère les données

Données censurées

- Censure à droite : $\mathbf{y} = [\underline{y_i}, +\infty)$
- Censure à gauche : $\mathbf{y} = (-\infty, \overline{y_i}]$
- Censure par intervalle : $\mathbf{y} = [\underline{y_i}, \overline{y_i}]$

Un problème d'apprentissage supervisé

Ensemble de données

$$\mathcal{S} \stackrel{\text{def}}{=} \{(\mathbf{x}_1, \mathbf{y}_1), (\mathbf{x}_2, \mathbf{y}_2), ..., (\mathbf{x}_n, \mathbf{y}_n))\} \sim D^n$$

$\mathbf{x} \in \mathbb{R}^p$ est un vecteur de caractéristiques

$\mathbf{y}_i \stackrel{\text{def}}{=} \left[\underline{y_i}, \overline{y_i}\right]$, avec $\underline{y_i}, \overline{y_i} \in \overline{\mathbb{R}}$ et $\underline{y_i} < \overline{y_i}$, est un intervalle

D est une distribution inconnue qui génère les données

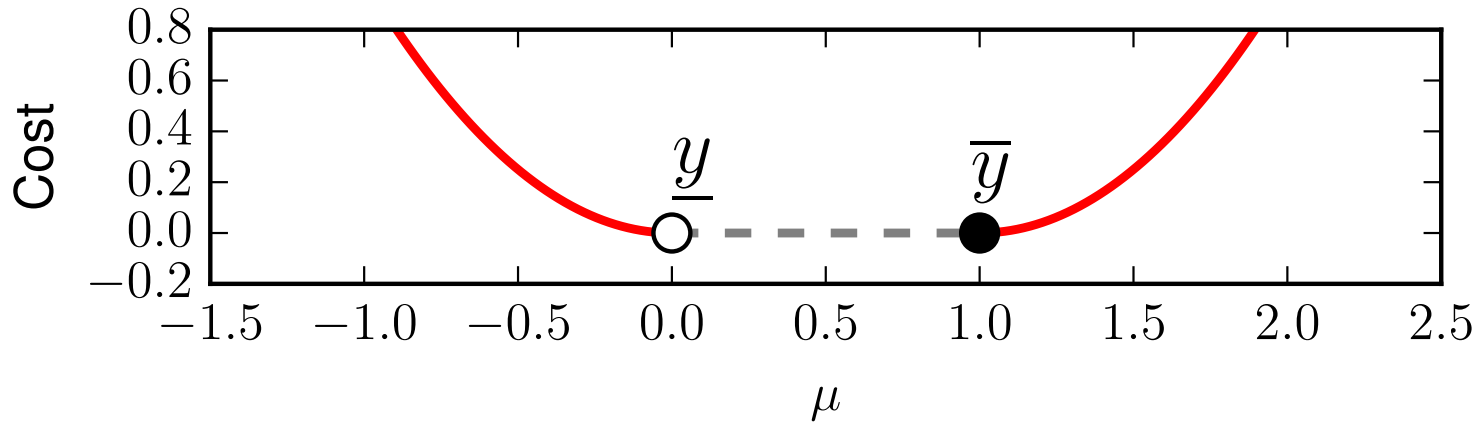
Données censurées

- Censure à droite : $\mathbf{y} = [\underline{y_i}, +\infty)$
- Censure à gauche : $\mathbf{y} = (-\infty, \overline{y_i}]$
- Censure par intervalle : $\mathbf{y} = [\underline{y_i}, \overline{y_i}]$

Erreur par rapport à un intervalle

Comment mesure-t-on l'erreur ?

Soit $\mu \in \mathbb{R}$ la valeur prédite par un prédicteur quelconque. L'erreur par rapport à l'intervalle $[\underline{y}, \overline{y}]$ est :



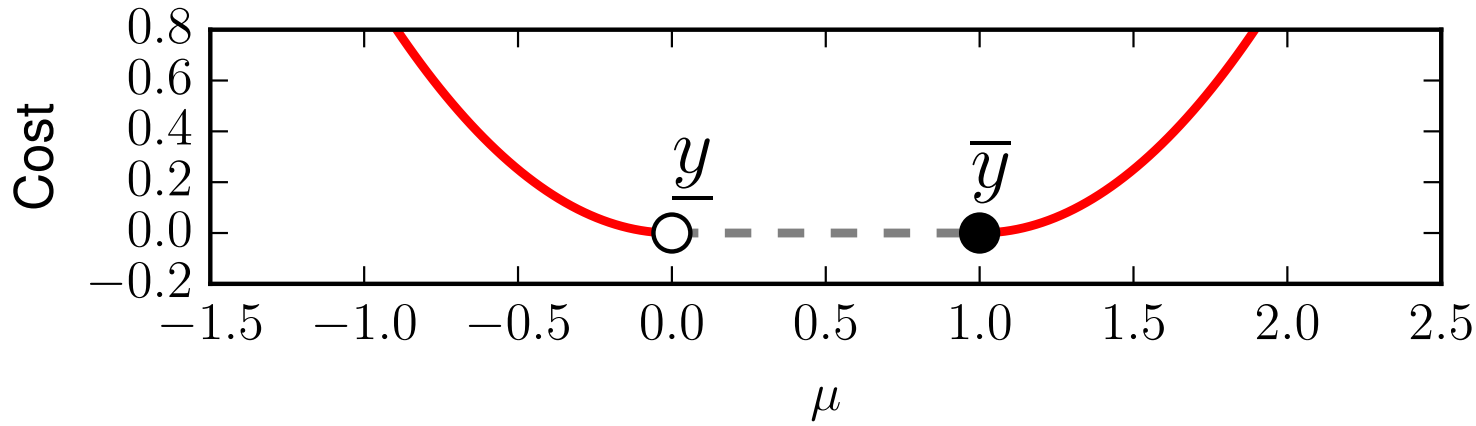
Hinge loss

1. Soit $\ell : \mathbb{R} \to \mathbb{R}$, une fonction croissante
2. Soit $[x]_+$, la fonction partie positive, i.e. $[x]_+ \neq 0 \Leftrightarrow x > 0$
3. Le *hinge loss* associé à ℓ est $\phi_\ell(x) = \ell([x]_+)$

Erreur par rapport à un intervalle

Comment mesure-t-on l'erreur ?

Soit $\mu \in \mathbb{R}$ la valeur prédite par un prédicteur quelconque. L'erreur par rapport à l'intervalle $[\underline{y}, \overline{y}]$ est :



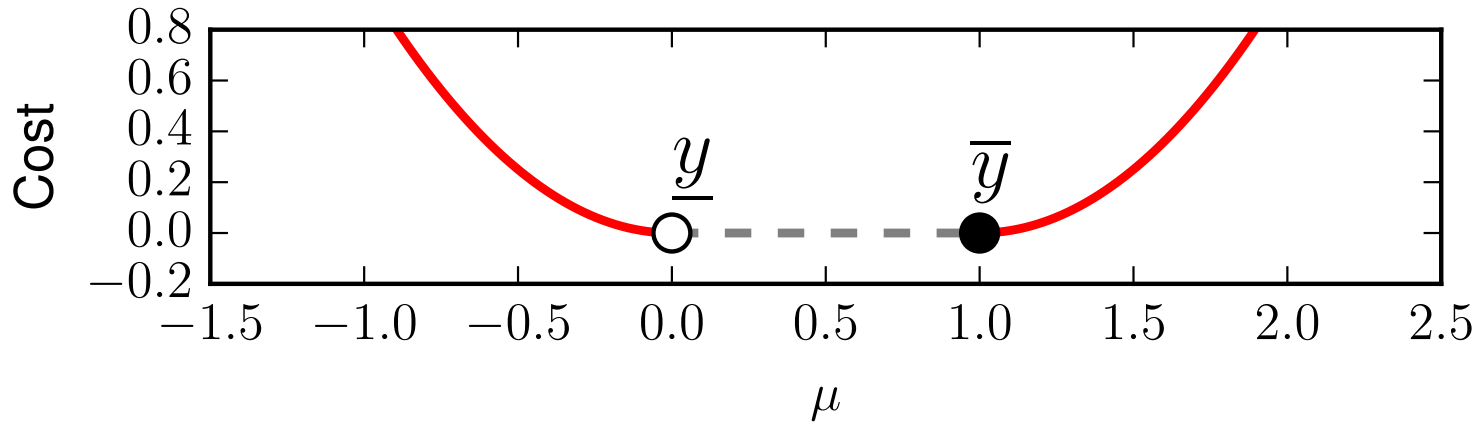
Hinge loss

1. Soit $\ell : \mathbb{R} \to \mathbb{R}$, une fonction croissante
2. Soit $[x]_+$, la fonction partie positive, i.e. $[x]_+ \neq 0 \Leftrightarrow x > 0$
3. Le *hinge loss* associé à ℓ est $\phi_\ell(x) = \ell([x]_+)$

Erreur par rapport à un intervalle

Comment mesure-t-on l'erreur ?

Soit $\mu \in \mathbb{R}$ la valeur prédite par un prédicteur quelconque. L'erreur par rapport à l'intervalle $[\underline{y}, \overline{y}]$ est :



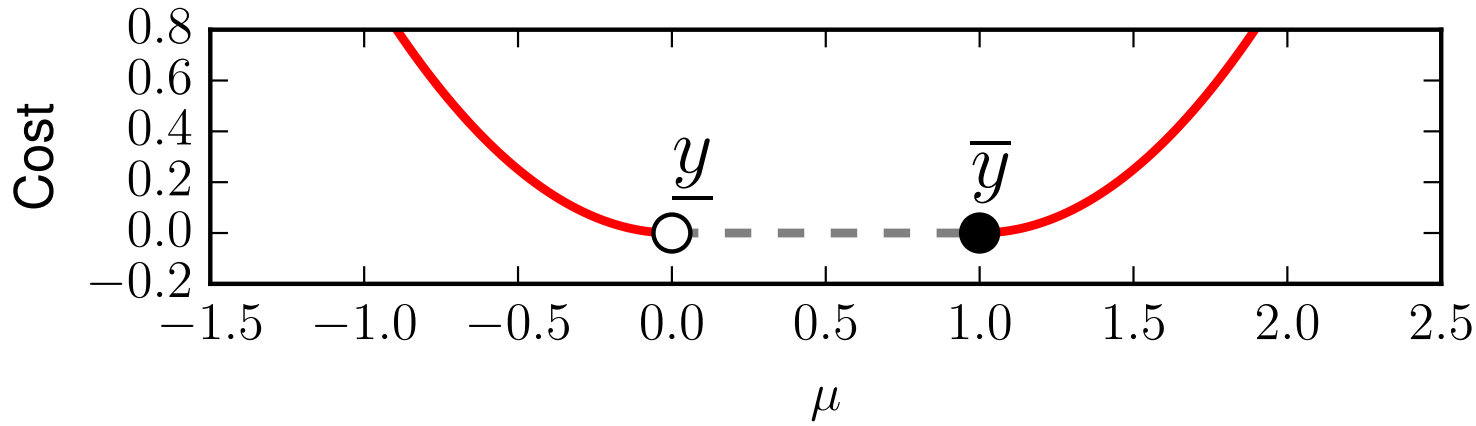
Hinge loss

1. Soit $\ell : \mathbb{R} \to \mathbb{R}$, une fonction croissante
2. Soit $[x]_+$, la fonction partie positive, i.e. $[x]_+ \neq 0 \Leftrightarrow x > 0$
3. Le *hinge loss* associé à ℓ est $\phi_\ell(x) = \ell([x]_+)$

Erreur par rapport à un intervalle

Comment mesure-t-on l'erreur ?

Soit $\mu \in \mathbb{R}$ la valeur prédite par un prédicteur quelconque. L'erreur par rapport à l'intervalle $[\underline{y}, \overline{y}]$ est :



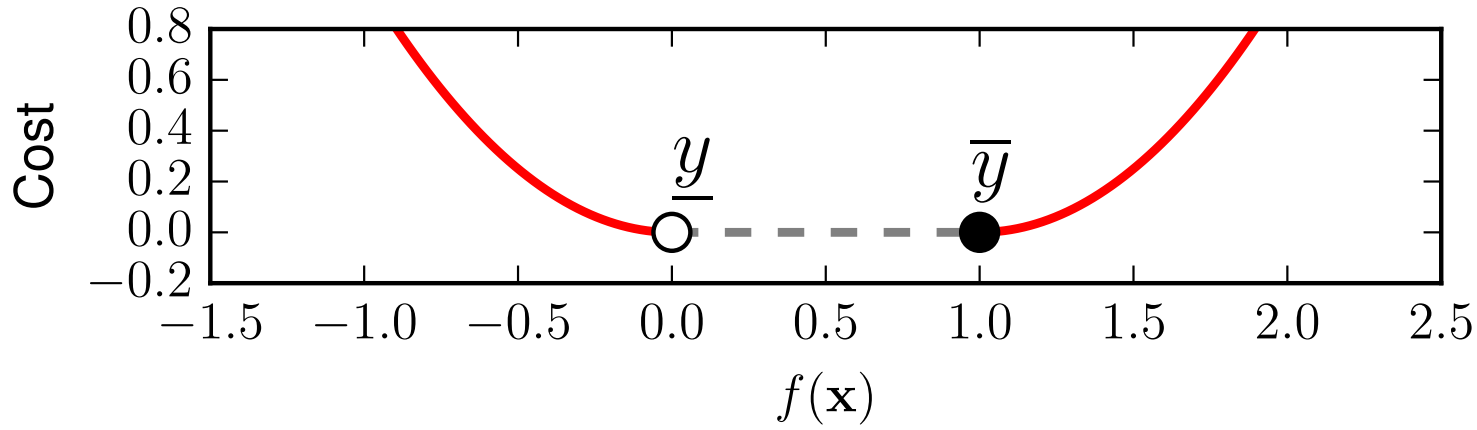
Hinge loss

1. Soit $\ell : \mathbb{R} \to \mathbb{R}$, une fonction croissante
2. Soit $[x]_+$, la fonction partie positive, i.e. $[x]_+ \neq 0 \Leftrightarrow x > 0$
3. Le *hinge loss* associé à ℓ est $\phi_\ell(x) = \ell([x]_+)$

Objectif

Notre but est de trouver une fonction $f : \mathbb{R}^p \to \mathbb{R}$ qui minimise l'erreur pour tout exemple tiré de D :

$$\underset{f}{\text{minimize}} \underset{(\mathbf{x}_i, \mathbf{y}_i) \sim D}{\mathbf{E}} \phi_\ell(\underline{y_i} - f(\mathbf{x}_i)) + \phi_\ell(f(\mathbf{x}_i) - \overline{y_i}),$$



L'algorithme Maximum Margin Interval Tree

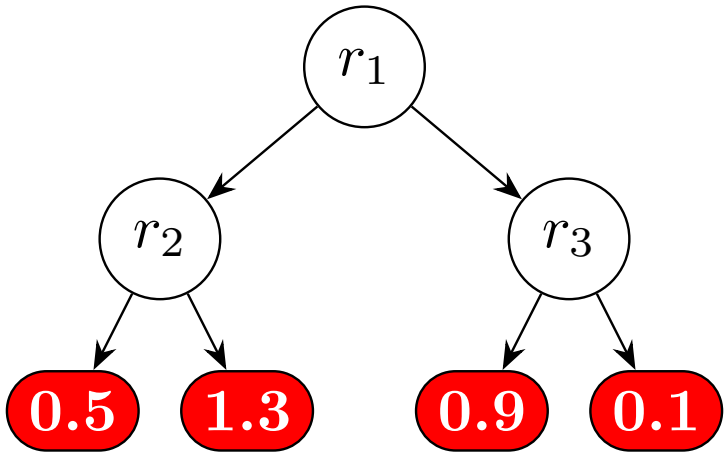
Maximum Margin Interval Tree

- Nous cherchons un arbre de régression $T : \mathbb{R}^p \to \mathbb{R}$ qui minimise :

$$C(T) \stackrel{\text{def}}{=} \sum_{(\mathbf{x}_i, \mathbf{y}_i) \in S} \left[\phi_\ell \left(-T(\mathbf{x}_i) + \underline{y_i} + \epsilon \right) + \phi_\ell \left(T(\mathbf{x}_i) - \overline{y_i} + \epsilon \right) \right]$$

où ϵ est un hyperparamètre de marge servant à la régularisation.

- Nous considérons les cas $\ell(x) = x$ et $\ell(x) = x^2$.



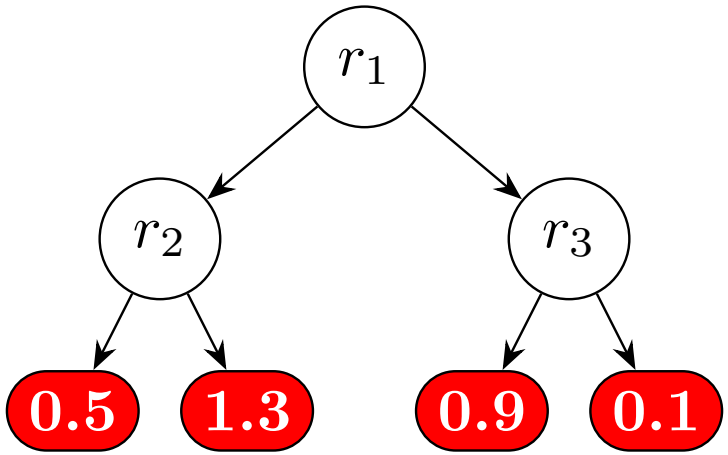
Maximum Margin Interval Tree

- Nous cherchons un arbre de régression $T : \mathbb{R}^p \to \mathbb{R}$ qui minimise :

$$C(T) \stackrel{\text{def}}{=} \sum_{(\mathbf{x}_i, \mathbf{y}_i) \in S} \left[\phi_\ell \left(-T(\mathbf{x}_i) + \underline{y_i} + \epsilon \right) + \phi_\ell \left(T(\mathbf{x}_i) - \overline{y_i} + \epsilon \right) \right]$$

où ϵ est un hyperparamètre de marge servant à la régularisation.

- Nous considérons les cas $\ell(x) = x$ et $\ell(x) = x^2$.



Les feuilles d'un MMIT

- Soit $\widetilde{T}$, l'ensemble des feuilles d'un arbre T
- Une feuille $\tau \in \widetilde{T}$ est associée à un ensemble d'exemples $S_\tau \subseteq S$ t.q.
  - $S = \bigcup_{\tau \in \widetilde{T}} S_\tau$
  - $S_\tau \cap S_{\tau'} \neq \emptyset \Leftrightarrow \tau = \tau'$
- Chaque feuille prédit une constante pour tous les exemples dans S_τ
- La contribution d'une feuille au coût total de l'arbre, étant donné qu'elle prédit $\mu \in \mathbb{R}$, est donnée par

$$C_\tau(\mu) \stackrel{\text{def}}{=} \sum_{(\mathbf{x}_i, \mathbf{y}_i) \in S_\tau} \left[\phi_\ell(-\mu + \underline{y_i} + \epsilon) + \phi_\ell(\mu - \overline{y_i} + \epsilon)\right]$$

Les feuilles d'un MMIT

- Soit $\widetilde{T}$, l'ensemble des feuilles d'un arbre T
- Une feuille $\tau \in \widetilde{T}$ est associée à un ensemble d'exemples $S_\tau \subseteq S$ t.q.
  - $S = \bigcup_{\tau \in \widetilde{T}} S_\tau$
  - $S_\tau \cap S_{\tau'} \neq \emptyset \Leftrightarrow \tau = \tau'$
- Chaque feuille prédit une constante pour tous les exemples dans S_τ
- La contribution d'une feuille au coût total de l'arbre, étant donné qu'elle prédit $\mu \in \mathbb{R}$, est donnée par

$$C_\tau(\mu) \stackrel{\text{def}}{=} \sum_{(\mathbf{x}_i, \mathbf{y}_i) \in S_\tau} \left[\phi_\ell(-\mu + \underline{y_i} + \epsilon) + \phi_\ell(\mu - \overline{y_i} + \epsilon) \right]$$

Les feuilles d'un MMIT

- Soit $\widetilde{T}$, l'ensemble des feuilles d'un arbre T
- Une feuille $\tau \in \widetilde{T}$ est associée à un ensemble d'exemples $S_\tau \subseteq S$ t.q.
  - $S = \bigcup_{\tau \in \widetilde{T}} S_\tau$
  - $S_\tau \cap S_{\tau'} \neq \emptyset \Leftrightarrow \tau = \tau'$
- Chaque feuille prédit une constante pour tous les exemples dans S_τ
- La contribution d'une feuille au coût total de l'arbre, étant donné qu'elle prédit $\mu \in \mathbb{R}$, est donnée par

$$C_\tau(\mu) \stackrel{\text{def}}{=} \sum_{(\mathbf{x}_i, \mathbf{y}_i) \in S_\tau} \left[\phi_\ell(-\mu + \underline{y_i} + \epsilon) + \phi_\ell(\mu - \overline{y_i} + \epsilon) \right]$$

Les feuilles d'un MMIT

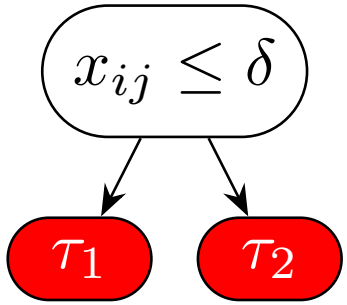
- Soit $\widetilde{T}$, l'ensemble des feuilles d'un arbre T
- Une feuille $\tau \in \widetilde{T}$ est associée à un ensemble d'exemples $S_\tau \subseteq S$ t.q.
  - $S = \bigcup_{\tau \in \widetilde{T}} S_\tau$
  - $S_\tau \cap S_{\tau'} \neq \emptyset \Leftrightarrow \tau = \tau'$
- Chaque feuille prédit une constante pour tous les exemples dans S_τ
- La contribution d'une feuille au coût total de l'arbre, étant donné qu'elle prédit $\mu \in \mathbb{R}$, est donnée par

$$C_\tau(\mu) \stackrel{\text{def}}{=} \sum_{(\mathbf{x}_i, \mathbf{y}_i) \in S_\tau} \left[\phi_\ell(-\mu + \underline{y_i} + \epsilon) + \phi_\ell(\mu - \overline{y_i} + \epsilon) \right]$$

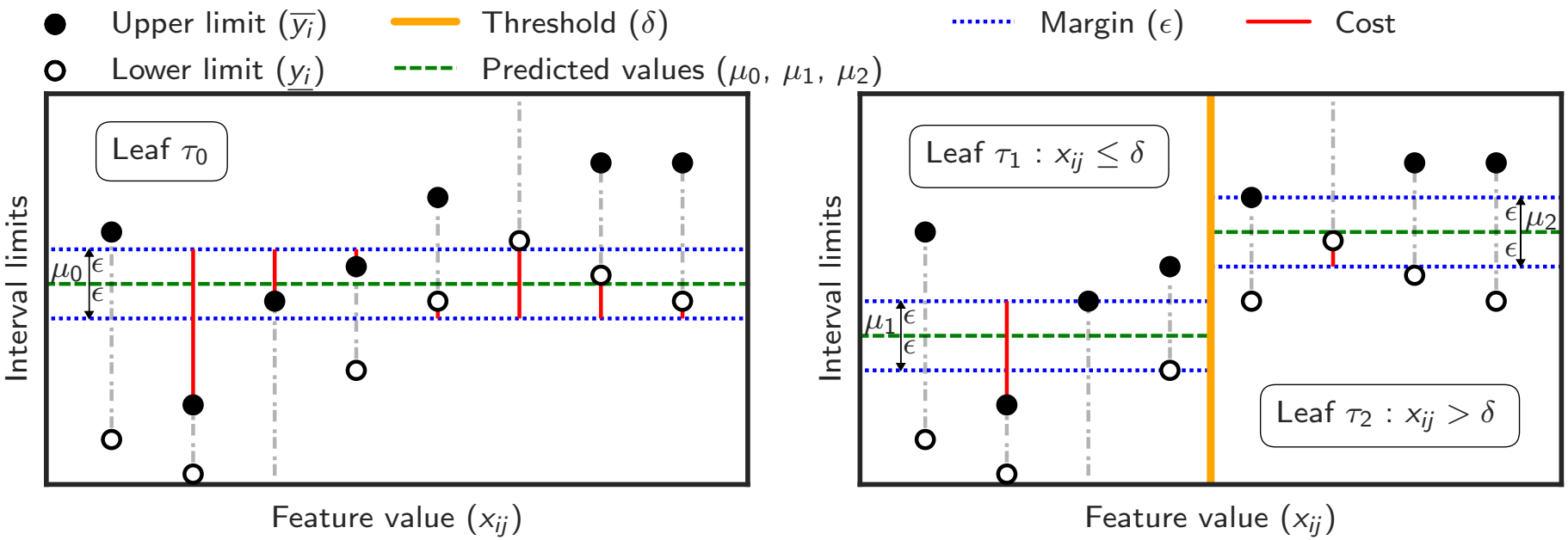
L'arbre est construit par partitionnement récursif

- Soit une feuille $\tau_0 \in \widetilde{T}$:

- Nous remplaçons τ_0 par deux nouvelles feuilles τ_1 et τ_2 selon une règle à valeur booléenne $r : \mathbb{R}^p \to \{\text{Vrai}, \text{Faux}\}$:



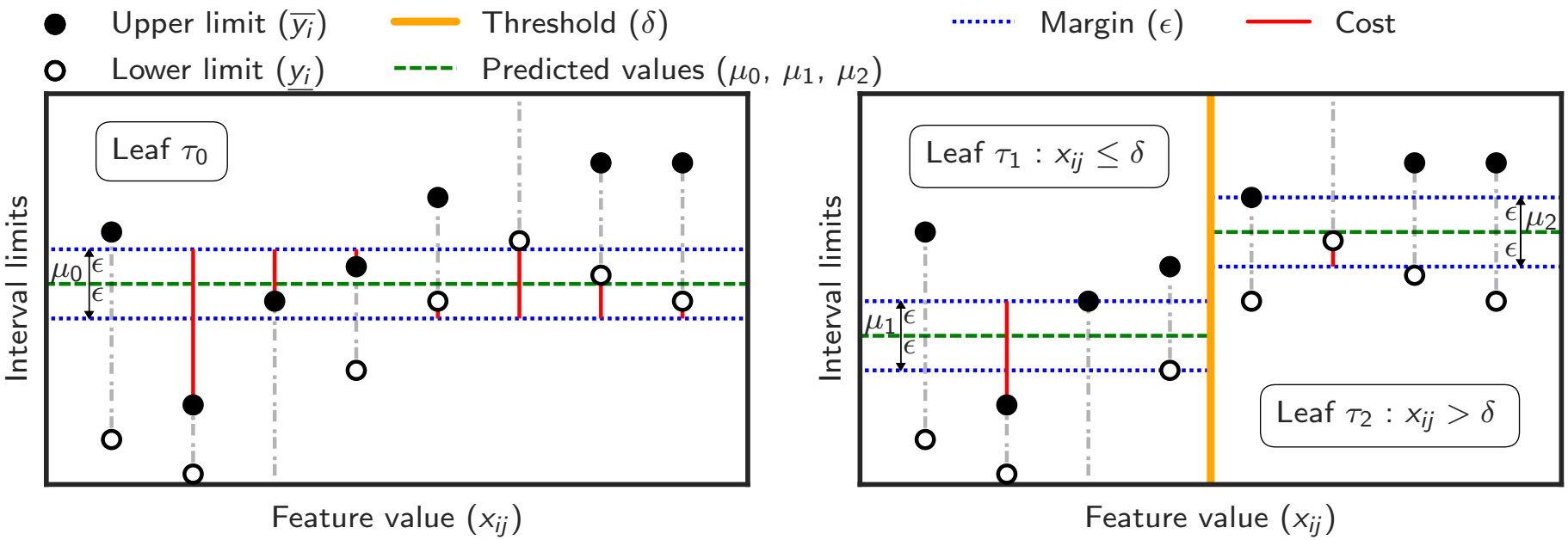
Partitionnement d'une feuille



$$\overleftarrow{C}_{\tau_0}(\mu_1 \,|\, j, \delta) \stackrel{\text{def}}{=} \sum_{(\mathbf{x}_i, \mathbf{y}_i) \in S_{\tau_0} \,:\, x_{ij} \leq \delta} \left[\phi_\ell(-\mu_1 + \underline{y_i} + \epsilon) + \phi_\ell(\mu_1 - \overline{y_i} + \epsilon)\right]$$

$$\overrightarrow{C}_{\tau_0}(\mu_2 \,|\, j, \delta) \stackrel{\text{def}}{=} \sum_{(\mathbf{x}_i, \mathbf{y}_i) \in S_{\tau_0} \,:\, x_{ij} > \delta} \left[\phi_\ell(-\mu_2 + \underline{y_i} + \epsilon) + \phi_\ell(\mu_2 - \overline{y_i} + \epsilon)\right]$$

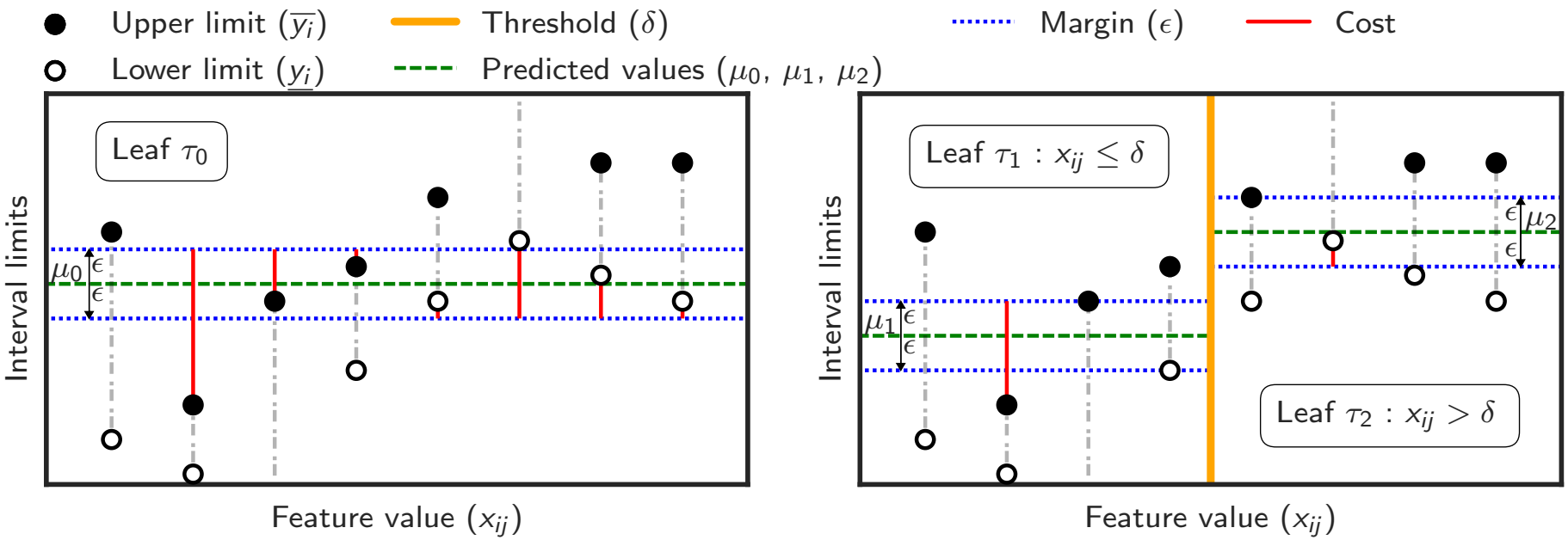
Partitionnement d'une feuille



$$\overleftarrow{C}_{\tau_0}(\mu_1 \,|\, j, \delta) \stackrel{\text{def}}{=} \sum_{(\mathbf{x}_i, \mathbf{y}_i) \in S_{\tau_0} \,:\, x_{ij} \leq \delta} \left[\phi_\ell(-\mu_1 + \underline{y_i} + \epsilon) + \phi_\ell(\mu_1 - \overline{y_i} + \epsilon)\right]$$

$$\overrightarrow{C}_{\tau_0}(\mu_2 \,|\, j, \delta) \stackrel{\text{def}}{=} \sum_{(\mathbf{x}_i, \mathbf{y}_i) \in S_{\tau_0} \,:\, x_{ij} > \delta} \left[\phi_\ell(-\mu_2 + \underline{y_i} + \epsilon) + \phi_\ell(\mu_2 - \overline{y_i} + \epsilon)\right]$$

Partitionnement d'une feuille



$$\overleftarrow{C}_{\tau_0}(\mu_1 \,|\, j, \delta) \stackrel{\text{def}}{=} \sum_{(\mathbf{x}_i, \mathbf{y}_i) \in S_{\tau_0} \,:\, x_{ij} \leq \delta} \left[\phi_\ell(-\mu_1 + \underline{y_i} + \epsilon) + \phi_\ell(\mu_1 - \overline{y_i} + \epsilon)\right]$$

$$\overrightarrow{C}_{\tau_0}(\mu_2 \,|\, j, \delta) \stackrel{\text{def}}{=} \sum_{(\mathbf{x}_i, \mathbf{y}_i) \in S_{\tau_0} \,:\, x_{ij} > \delta} \left[\phi_\ell(-\mu_2 + \underline{y_i} + \epsilon) + \phi_\ell(\mu_2 - \overline{y_i} + \epsilon)\right]$$

Problème d'optimisation convexe

Nous trouvons la règle à utiliser et les valeurs à prédire dans les feuilles en solutionnant le problème d'optimisation suivant :

$$\min_{j,\delta,\mu_1,\mu_2} \left[\overleftarrow{C_\tau}(\mu_1 \,|\, j, \delta) + \overrightarrow{C_\tau}(\mu_2 \,|\, j, \delta) \right]$$

Pour une règle donnée $(x_{ij} \leq \delta)$, ceci correspond à deux problèmes d'optimisation convexe :

$$\min_{j,\delta} \left[\min_{\mu_1} \overleftarrow{C_\tau}(\mu_1 \,|\, j, \delta) + \min_{\mu_2} \overrightarrow{C_\tau}(\mu_2 \,|\, j, \delta) \right]$$

Problème d'optimisation convexe

Nous trouvons la règle à utiliser et les valeurs à prédire dans les feuilles en solutionnant le problème d'optimisation suivant :

$$\min_{j,\delta,\mu_1,\mu_2} \left[\overleftarrow{C_\tau}(\mu_1 \,|\, j, \delta) + \overrightarrow{C_\tau}(\mu_2 \,|\, j, \delta) \right]$$

Pour une règle donnée $(x_{ij} \leq \delta)$, ceci correspond à deux problèmes d'optimisation convexe :

$$\min_{j,\delta} \left[\min_{\mu_1} \overleftarrow{C_\tau}(\mu_1 \,|\, j, \delta) + \min_{\mu_2} \overrightarrow{C_\tau}(\mu_2 \,|\, j, \delta) \right]$$

Vers une solution par programmation dynamique

Observation

$\overleftarrow{C_\tau}(\mu_1 \,|\, j, \delta)$ et $\overrightarrow{C_\tau}(\mu_2 \,|\, j, \delta)$ calculent la somme d'un ensemble de *hinge loss* de type :

- $\mu \mapsto \phi_\ell(-\mu + \underline{y_i} + \epsilon)$
- $\mu \mapsto \phi_\ell(\mu - \overline{y_i} + \epsilon)$

- Supposons qu'il existe un algorithme Ω :
  - **Entrée :** un ensemble quelconque de tels *hinge loss* définis sur μ
  - **Tâche :** calcule la somme de tous les hinge loss
  - **Sortie :** la valeur d'un minimum global et la valeur de μ correspondante
- Nous montrerons que nous pouvons utiliser un tel Ω pour résoudre le problème d'optimisation précédent de façon efficace

Vers une solution par programmation dynamique

Observation

$\overleftarrow{C_\tau}(\mu_1 \,|\, j, \delta)$ et $\overrightarrow{C_\tau}(\mu_2 \,|\, j, \delta)$ calculent la somme d'un ensemble de *hinge loss* de type :

- $\mu \mapsto \phi_\ell(-\mu + \underline{y_i} + \epsilon)$
- $\mu \mapsto \phi_\ell(\mu - \overline{y_i} + \epsilon)$

- Supposons qu'il existe un algorithme Ω :
  - **Entrée :** un ensemble quelconque de tels *hinge loss* définis sur μ
  - **Tâche :** calcule la somme de tous les hinge loss
  - **Sortie :** la valeur d'un minimum global et la valeur de μ correspondante

- Nous montrerons que nous pouvons utiliser un tel Ω pour résoudre le problème d'optimisation précédent de façon efficace

Vers une solution par programmation dynamique

Observation

$\overleftarrow{C_\tau}(\mu_1 \,|\, j, \delta)$ et $\overrightarrow{C_\tau}(\mu_2 \,|\, j, \delta)$ calculent la somme d'un ensemble de *hinge loss* de type :

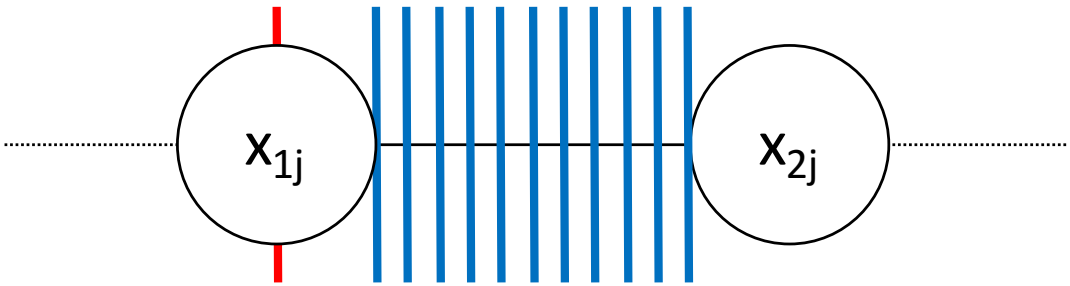
- $\mu \mapsto \phi_\ell(-\mu + \underline{y_i} + \epsilon)$
- $\mu \mapsto \phi_\ell(\mu - \overline{y_i} + \epsilon)$

- Supposons qu'il existe un algorithme Ω :
  - **Entrée :** un ensemble quelconque de tels *hinge loss* définis sur μ
  - **Tâche :** calcule la somme de tous les hinge loss
  - **Sortie :** la valeur d'un minimum global et la valeur de μ correspondante

- Nous montrerons que nous pouvons utiliser un tel Ω pour résoudre le problème d'optimisation précédent de façon efficace

Espace de recherche pour δ

- Étant donné un attribut j, nous cherchons le meilleur seuil $\delta \in \mathbb{R}$ pour une règle de type $x_{ij} \leq \delta$
- Il existe une infinité de valeurs possibles
- Seul les valeurs $\delta \in \{x_{1j}, \ldots, x_{nj}\}$ doivent être considérées, car les autres valeurs ne changent pas les valeurs de S_{τ_1} et S_{τ_2}.



- Soit $n_j \leq n$, le nombre de valeurs uniques que prend l'attribut j, nous considérons les seuils : $\delta_{j,1} < \cdots < \delta_{j,n_j}$

Vers une solution par programmation dynamique

- Pour chaque valeur de seuil $\delta_{j,k}$, nous définissons $\Phi_{j,k}$, l'ensemble des *hinge loss* de type $\mu \mapsto \phi_\ell(-\mu + \underline{y_i} + \epsilon)$ et $\mu \mapsto \phi_\ell(\mu - \overline{y_i} + \epsilon)$ pour les $(\mathbf{x}_i, \mathbf{y}_i) \in S_\tau$ tels que $x_{ij} = \delta_{j,k}$.

- Puisque $\delta_{j,i} < \delta_{j,i+1}$, nous avons :

$$\min_\mu \overleftarrow{C_\tau}(\mu \,|\, j, \delta_{j,k}) = \Omega(\Phi_{j,1} \cup \cdots \cup \Phi_{j,k})$$

$$\min_\mu \overrightarrow{C_\tau}(\mu \,|\, j, \delta_{j,k}) = \Omega(\Phi_{j,k+1} \cup \cdots \cup \Phi_{j,n_j})$$

- De plus, nous pouvons obtenir $\overleftarrow{C_\tau}(\mu \,|\, j, \delta_{j,k})$ à partir de $\overleftarrow{C_\tau}(\mu \,|\, j, \delta_{j,k-1})$ en ajoutant les *hinge loss* dans $\Phi_{j,k}$.

- De façon similaire, nous pouvons obtenir $\overrightarrow{C_\tau}(\mu \,|\, j, \delta_{j,k})$ à partir de $\overrightarrow{C_\tau}(\mu \,|\, j, \delta_{j,k-1})$ en enlevant les *hinge loss* dans $\Phi_{j,k}$.

Vers une solution par programmation dynamique

- Pour chaque valeur de seuil $\delta_{j,k}$, nous définissons $\Phi_{j,k}$, l'ensemble des *hinge loss* de type $\mu \mapsto \phi_\ell(-\mu + \underline{y_i} + \epsilon)$ et $\mu \mapsto \phi_\ell(\mu - \overline{y_i} + \epsilon)$ pour les $(\mathbf{x}_i, \mathbf{y}_i) \in S_\tau$ tels que $x_{ij} = \delta_{j,k}$.

- Puisque $\delta_{j,i} < \delta_{j,i+1}$, nous avons :

$$\min_\mu \overleftarrow{C_\tau}(\mu \,|\, j, \delta_{j,k}) = \Omega(\Phi_{j,1} \cup \cdots \cup \Phi_{j,k})$$

$$\min_\mu \overrightarrow{C_\tau}(\mu \,|\, j, \delta_{j,k}) = \Omega(\Phi_{j,k+1} \cup \cdots \cup \Phi_{j,n_j})$$

- De plus, nous pouvons obtenir $\overleftarrow{C_\tau}(\mu \,|\, j, \delta_{j,k})$ à partir de $\overleftarrow{C_\tau}(\mu \,|\, j, \delta_{j,k-1})$ en ajoutant les *hinge loss* dans $\Phi_{j,k}$.

- De façon similaire, nous pouvons obtenir $\overrightarrow{C_\tau}(\mu \,|\, j, \delta_{j,k})$ à partir de $\overrightarrow{C_\tau}(\mu \,|\, j, \delta_{j,k-1})$ en enlevant les *hinge loss* dans $\Phi_{j,k}$.

Vers une solution par programmation dynamique

- Pour chaque valeur de seuil $\delta_{j,k}$, nous définissons $\Phi_{j,k}$, l'ensemble des *hinge loss* de type $\mu \mapsto \phi_\ell(-\mu + \underline{y_i} + \epsilon)$ et $\mu \mapsto \phi_\ell(\mu - \overline{y_i} + \epsilon)$ pour les $(\mathbf{x}_i, \mathbf{y}_i) \in S_\tau$ tels que $x_{ij} = \delta_{j,k}$.

- Puisque $\delta_{j,i} < \delta_{j,i+1}$, nous avons :

$$\min_\mu \overleftarrow{C_\tau}(\mu \,|\, j, \delta_{j,k}) = \Omega(\Phi_{j,1} \cup \cdots \cup \Phi_{j,k})$$

$$\min_\mu \overrightarrow{C_\tau}(\mu \,|\, j, \delta_{j,k}) = \Omega(\Phi_{j,k+1} \cup \cdots \cup \Phi_{j,n_j})$$

- De plus, nous pouvons obtenir $\overleftarrow{C_\tau}(\mu \,|\, j, \delta_{j,k})$ à partir de $\overleftarrow{C_\tau}(\mu \,|\, j, \delta_{j,k-1})$ en ajoutant les *hinge loss* dans $\Phi_{j,k}$.

- De façon similaire, nous pouvons obtenir $\overrightarrow{C_\tau}(\mu \,|\, j, \delta_{j,k})$ à partir de $\overrightarrow{C_\tau}(\mu \,|\, j, \delta_{j,k-1})$ en enlevant les *hinge loss* dans $\Phi_{j,k}$.

Vers une solution par programmation dynamique

- Pour chaque valeur de seuil $\delta_{j,k}$, nous définissons $\Phi_{j,k}$, l'ensemble des *hinge loss* de type $\mu \mapsto \phi_\ell(-\mu + \underline{y_i} + \epsilon)$ et $\mu \mapsto \phi_\ell(\mu - \overline{y_i} + \epsilon)$ pour les $(\mathbf{x}_i, \mathbf{y}_i) \in S_\tau$ tels que $x_{ij} = \delta_{j,k}$.

- Puisque $\delta_{j,i} < \delta_{j,i+1}$, nous avons :

$$\min_\mu \overleftarrow{C_\tau}(\mu \,|\, j, \delta_{j,k}) = \Omega(\Phi_{j,1} \cup \cdots \cup \Phi_{j,k})$$

$$\min_\mu \overrightarrow{C_\tau}(\mu \,|\, j, \delta_{j,k}) = \Omega(\Phi_{j,k+1} \cup \cdots \cup \Phi_{j,n_j})$$

- De plus, nous pouvons obtenir $\overleftarrow{C_\tau}(\mu \,|\, j, \delta_{j,k})$ à partir de $\overleftarrow{C_\tau}(\mu \,|\, j, \delta_{j,k-1})$ en ajoutant les *hinge loss* dans $\Phi_{j,k}$.

- De façon similaire, nous pouvons obtenir $\overrightarrow{C_\tau}(\mu \,|\, j, \delta_{j,k})$ à partir de $\overrightarrow{C_\tau}(\mu \,|\, j, \delta_{j,k-1})$ en enlevant les *hinge loss* dans $\Phi_{j,k}$.

Utilisation de l'algorithme Ω

Le coût associé à chacune des n_j valeurs de seuil est donnée par :

$$\begin{array}{rcc}
\delta_{j,1}: & \Omega(\Phi_{j,1}) & + \quad \Omega(\Phi_{j,2} \cup \cdots \cup \Phi_{j,n_j}) \\
\vdots & \vdots & \vdots \\
\delta_{j,i}: & \Omega(\Phi_{j,1} \cup \cdots \cup \Phi_{j,i}) & + \quad \Omega(\Phi_{j,i+1} \cup \cdots \cup \Phi_{j,n_j}) \\
\vdots & \vdots & \vdots \\
\delta_{j,n_j-1}: & \Omega(\Phi_{j,1} \cup \cdots \cup \Phi_{j,n_j-1}) & + \quad \Omega(\Phi_{j,n_j})
\end{array}$$

Si Ω est un programme dynamique, nous pouvons calculer chaque solution efficacement

Utilisation de l'algorithme Ω

Le coût associé à chacune des n_j valeurs de seuil est donnée par :

$$\begin{array}{rcc}
\delta_{j,1}: & \Omega(\Phi_{j,1}) & + \quad \Omega(\Phi_{j,2} \cup \cdots \cup \Phi_{j,n_j}) \\
\vdots & \vdots & \vdots \\
\delta_{j,i}: & \Omega(\Phi_{j,1} \cup \cdots \cup \Phi_{j,i}) & + \quad \Omega(\Phi_{j,i+1} \cup \cdots \cup \Phi_{j,n_j}) \\
\vdots & \vdots & \vdots \\
\delta_{j,n_j-1}: & \Omega(\Phi_{j,1} \cup \cdots \cup \Phi_{j,n_j-1}) & + \quad \Omega(\Phi_{j,n_j})
\end{array}$$

De haut en bas pour la première colonne

Utilisation de l'algorithme Ω

Le coût associé à chacune des n_j valeurs de seuil est donnée par :

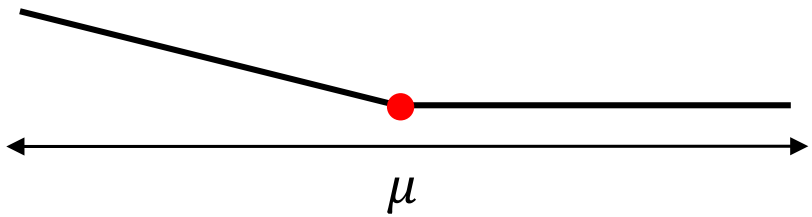
$$\begin{array}{lcl}
\delta_{j,1}: & \Omega(\Phi_{j,1}) & + \quad \Omega(\Phi_{j,2} \cup \cdots \cup \Phi_{j,n_j}) \\
\vdots & \vdots & \vdots \\
\delta_{j,i}: & \Omega(\Phi_{j,1} \cup \cdots \cup \Phi_{j,i}) & + \quad \Omega(\Phi_{j,i+1} \cup \cdots \cup \Phi_{j,n_j}) \\
\vdots & \vdots & \vdots \\
\delta_{j,n_j-1}: & \Omega(\Phi_{j,1} \cup \cdots \cup \Phi_{j,n_j-1}) & + \quad \Omega(\Phi_{j,n_j})
\end{array}$$

De bas en haut pour la deuxième colonne

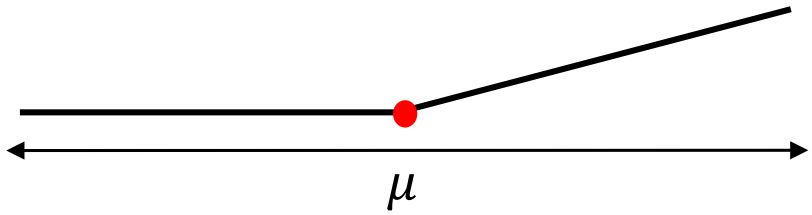
Intuition du fonctionnement de l'algorithme Ω

Les *hinge loss* que nous considérons ont la forme suivante (cas $\ell(x) = x$) :

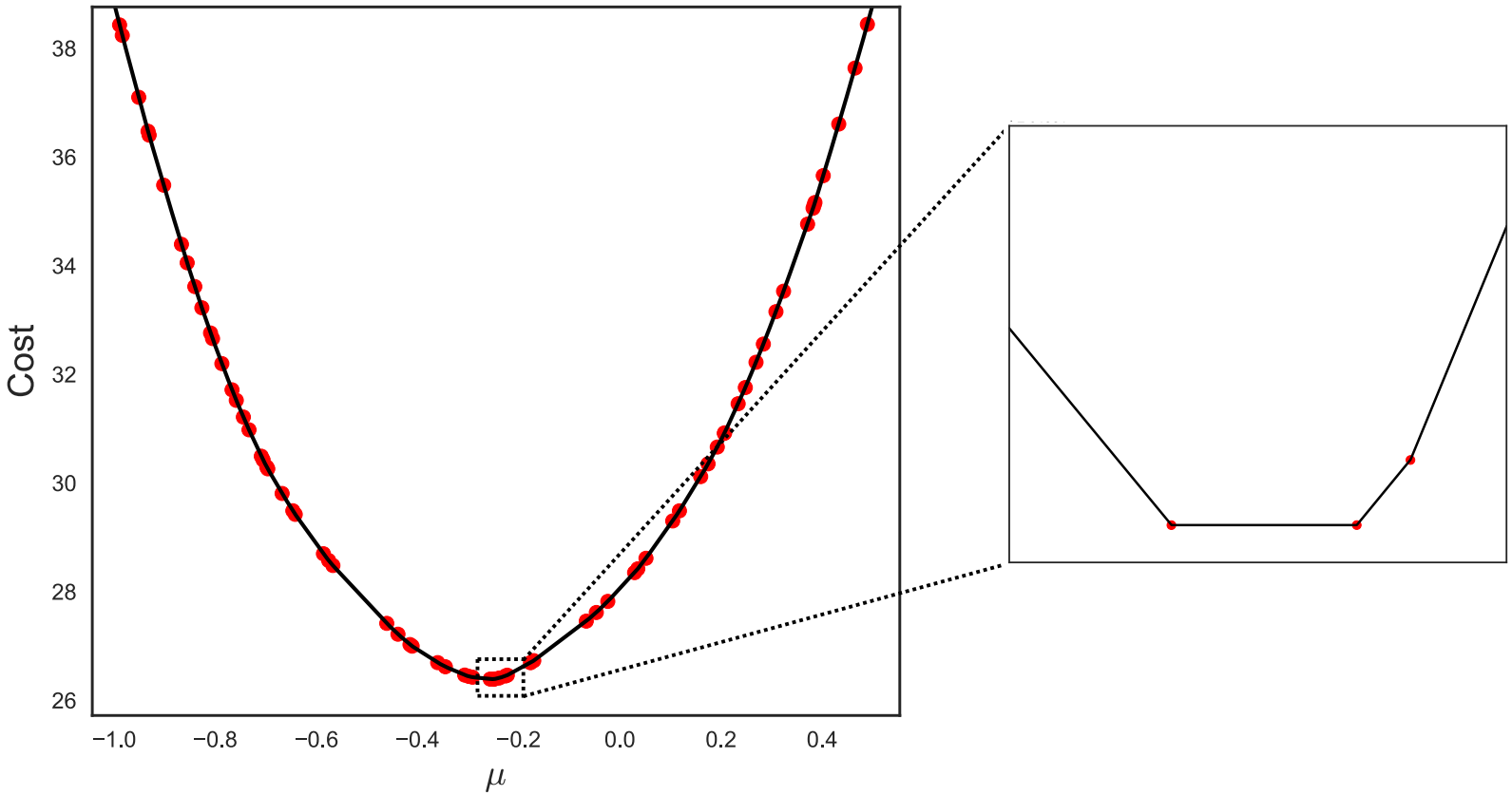
- Cas $\mu \mapsto \phi_\ell(-\mu + \underline{y_i} + \epsilon)$:



- Cas $\mu \mapsto \phi_\ell(\mu - \overline{y_i} + \epsilon)$:

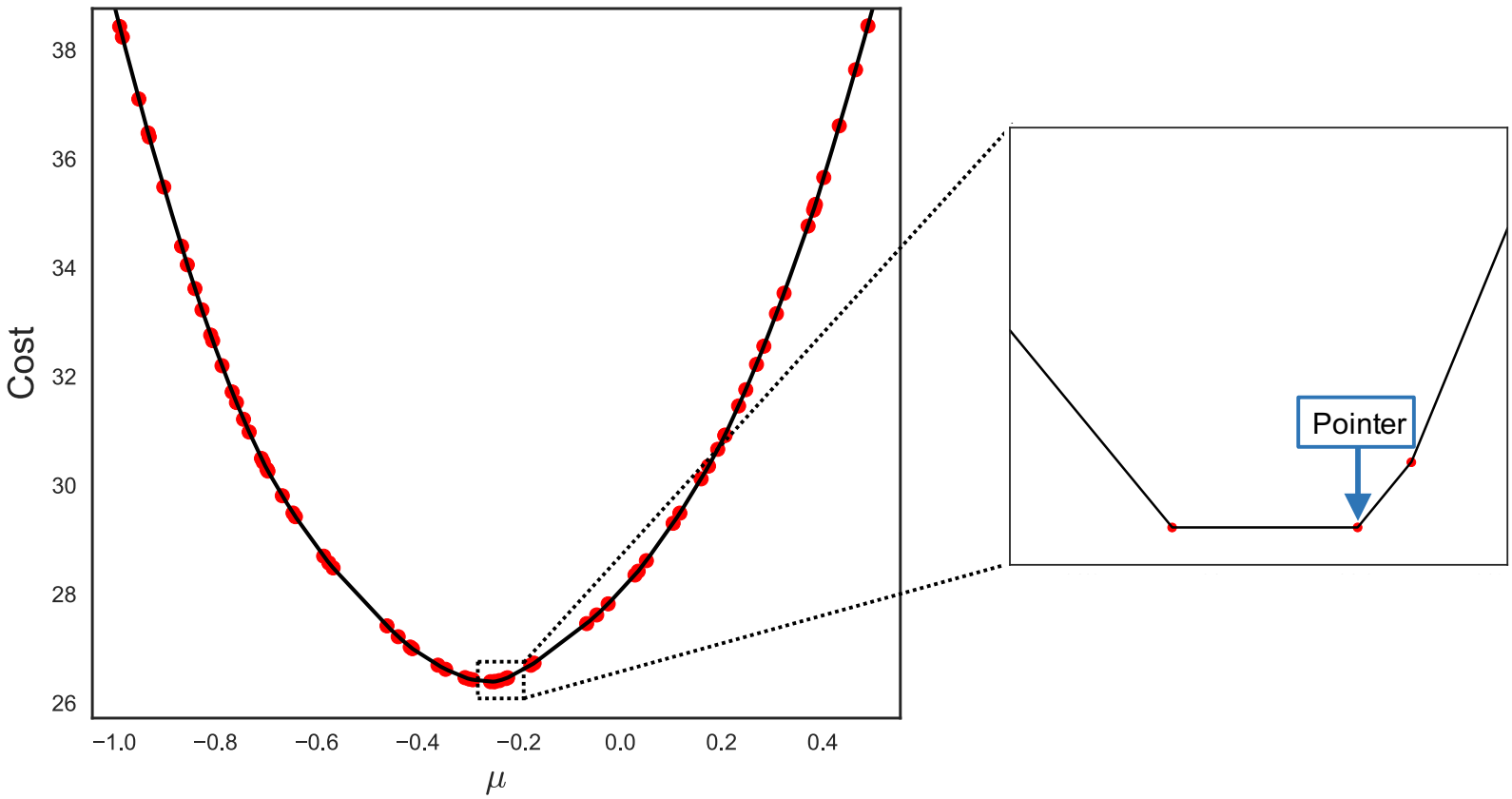


Intuition du fonctionnement de l'algorithme Ω



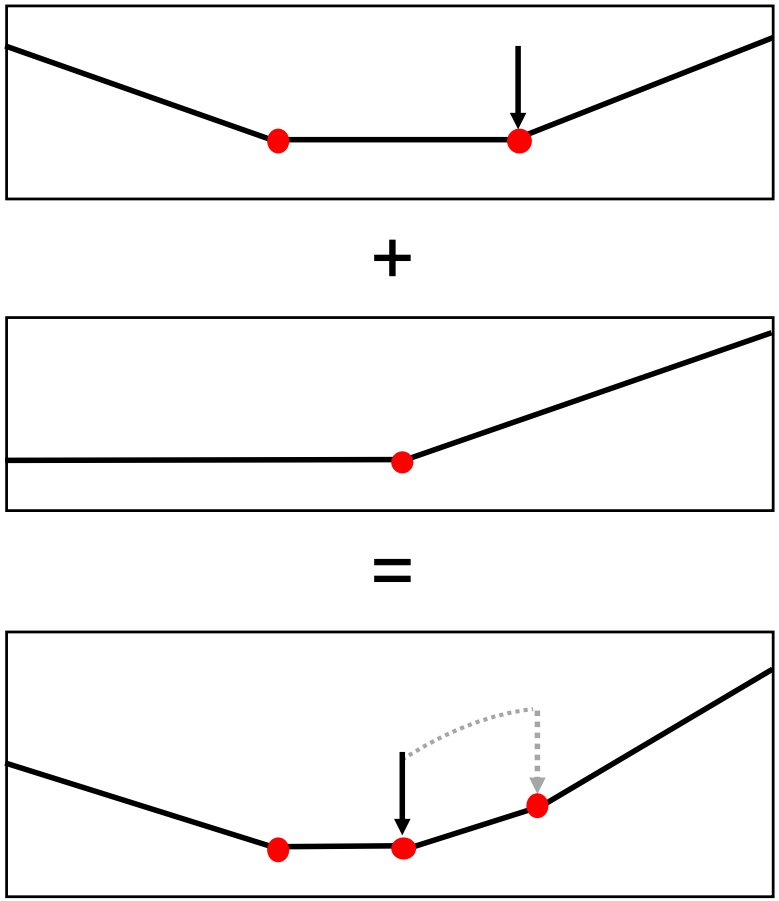
Une somme de *hinge loss* est une fonction convexe qui est linéaire (ou quadratique) par parties

Intuition du fonctionnement de l'algorithme Ω



Nous maintenons toujours un pointeur sur le premier point de changement le plus à droite de tout minimum global

Intuition du fonctionnement de l'algorithme Ω



On ajoute les *hinge loss* un par un. Le pointeur doit parfois être déplacé.

Complexité de calcul de l'algorithme Ω

- Supposons que nous avons n *hinge loss*
- Nous ajoutons les *hinge loss* à la somme un par un (n fois)
- L'insertion d'un nouveau loss se fait en $O(\max(\log n, k))$ où k est le nombre de déplacements de pointeur requis
- Cas $\ell(x) = x$: nous avons démontré que $k \in \{0, 1\}$, donc $O(n \log n)$
- Cas $\ell(x) = x^2$: nous n'avons pas de telle garantie (démonstration empirique)

Complexité de calcul de l'algorithme Ω

- Supposons que nous avons *n hinge loss*
- Nous ajoutons les *hinge loss* à la somme un par un (*n* fois)
- L'insertion d'un nouveau loss se fait en $O(\max(\log n, k))$ où k est le nombre de déplacements de pointeur requis
- Cas $\ell(x) = x$: nous avons démontré que $k \in \{0, 1\}$, donc $O(n \log n)$
- Cas $\ell(x) = x^2$: nous n'avons pas de telle garantie (démonstration empirique)

Complexité de calcul de l'algorithme Ω

- Supposons que nous avons n *hinge loss*
- Nous ajoutons les *hinge loss* à la somme un par un (n fois)
- L'insertion d'un nouveau loss se fait en $O(\max(\log n, k))$ où k est le nombre de déplacements de pointeur requis
- Cas $\ell(x) = x$: nous avons démontré que $k \in \{0, 1\}$, donc $O(n \log n)$
- Cas $\ell(x) = x^2$: nous n'avons pas de telle garantie (démonstration empirique)

Complexité de calcul de l'algorithme Ω

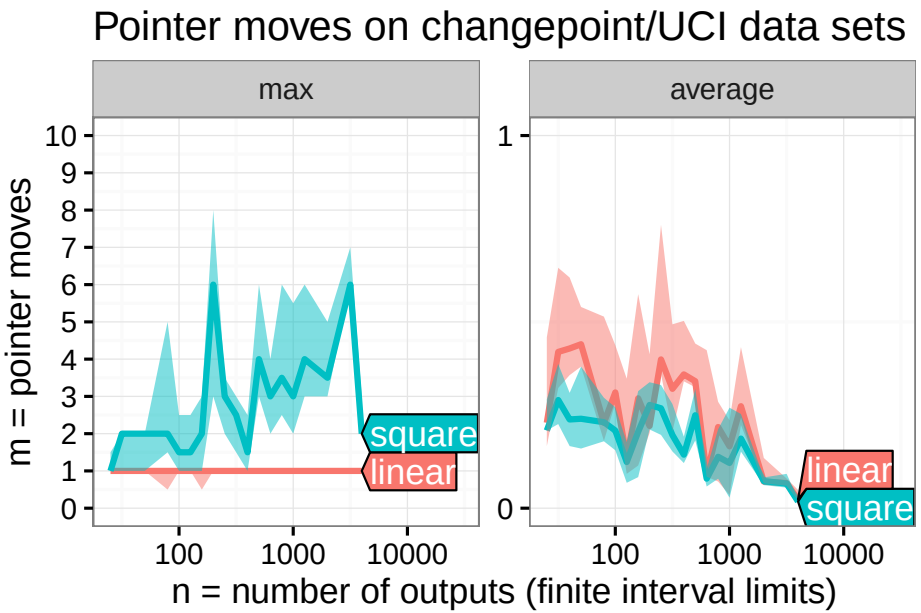
- Supposons que nous avons n *hinge loss*
- Nous ajoutons les *hinge loss* à la somme un par un (n fois)
- L'insertion d'un nouveau loss se fait en $O(\max(\log n, k))$ où k est le nombre de déplacements de pointeur requis
- Cas $\ell(x) = x$: nous avons démontré que $k \in \{0, 1\}$, donc $O(n \log n)$
- Cas $\ell(x) = x^2$: nous n'avons pas de telle garantie (démonstration empirique)

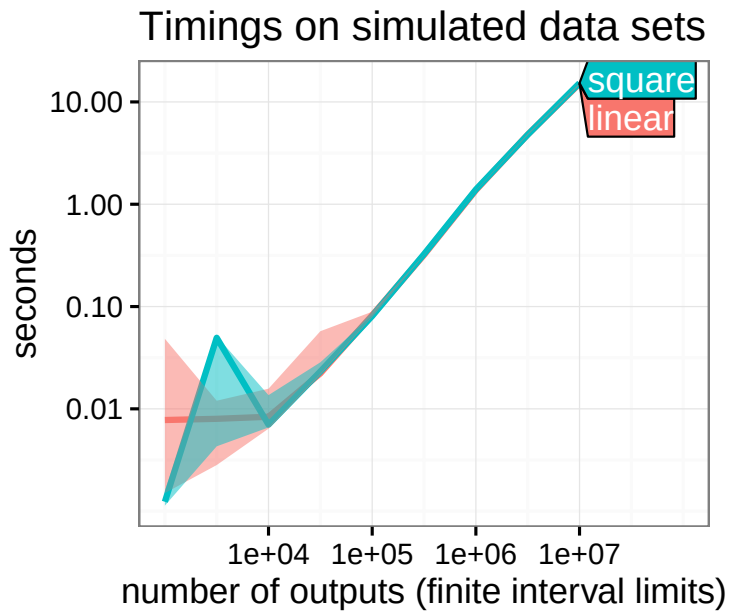
Complexité de calcul de l'algorithme Ω

- Supposons que nous avons n *hinge loss*
- Nous ajoutons les *hinge loss* à la somme un par un (n fois)
- L'insertion d'un nouveau loss se fait en $O(\max(\log n, k))$ où k est le nombre de déplacements de pointeur requis
- Cas $\ell(x) = x$: nous avons démontré que $k \in \{0, 1\}$, donc $O(n \log n)$
- Cas $\ell(x) = x^2$: nous n'avons pas de telle garantie (démonstration empirique)

Résultats

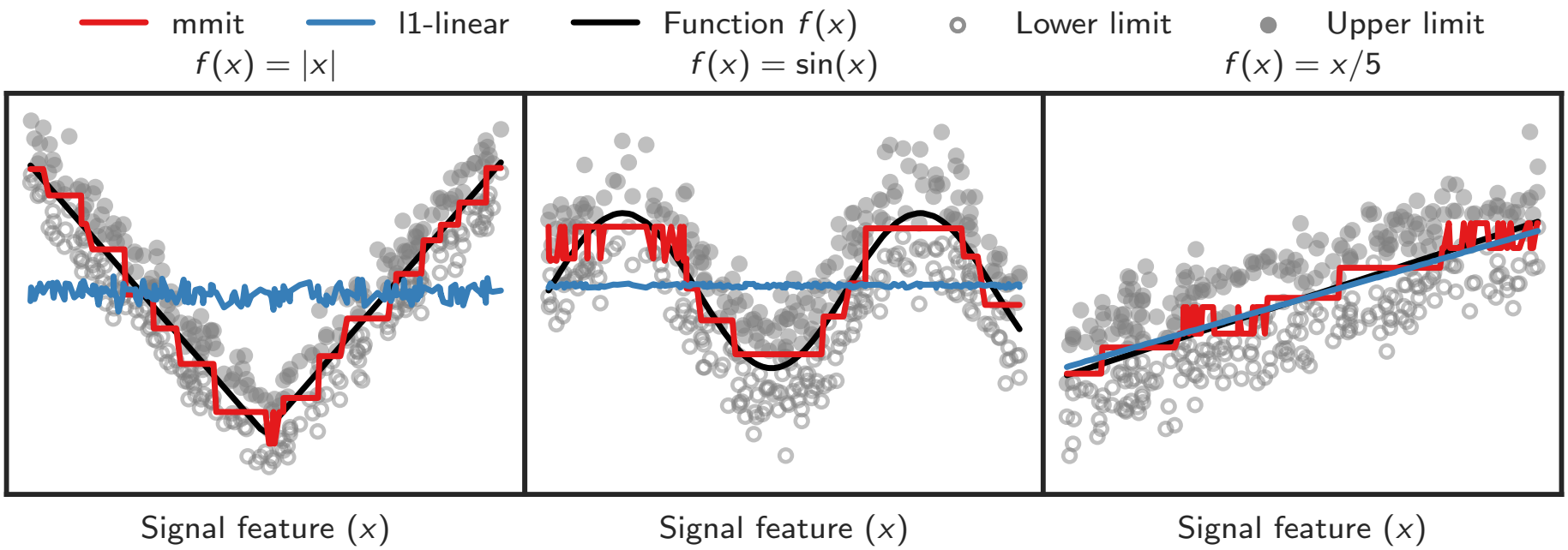
Mesure empirique de la complexité de calcul





En moyenne, nous avons observé un temps de calcul proportionnel à $n \log n$ pour le cas linéaire et le cas quadratique

MMIT peut modéliser des fonctions non-linéaires



MMIT arrive à modéliser des fonctions non-linéaires, alors que le modèle linéaire de Hocking et al. (2013) n'y arrive pas

Comparaison à l'état de l'art

Nous avons comparé notre méthode aux algorithmes suivants :

- **Constant :** utilise l'ensemble d'entraînement pour trouver la fonction constante qui traverse le plus d'intervalles possible
- **Interval-CART :** un arbre de régression (Breiman et al., 1984) où chaque exemple $(\mathbf{x}_i, \mathbf{y}_i) \in S$ a été remplacé par deux exemples de régression standard : $(\mathbf{x}_i, \underline{y_i} + \epsilon)$ et $(\mathbf{x}_i, \overline{y_i} - \epsilon)$.
- **L1-Linear :** régression linéaire par intervalle à base de marge (Hocking et al., 2013)
- **TransfoTree :** un algorithme d'arbre de décision pour données censurées n'utilisant pas de marge (Hothorn et Zeileis, 2017)

Comparaison à l'état de l'art

Nous avons comparé notre méthode aux algorithmes suivants :

- **Constant :** utilise l'ensemble d'entraînement pour trouver la fonction constante qui traverse le plus d'intervalles possible
- **Interval-CART :** un arbre de régression (Breiman et al., 1984) où chaque exemple $(\mathbf{x}_i, \mathbf{y}_i) \in S$ a été remplacé par deux exemples de régression standard : $(\mathbf{x}_i, \underline{y_i} + \epsilon)$ et $(\mathbf{x}_i, \overline{y_i} - \epsilon)$.
- **L1-Linear :** régression linéaire par intervalle à base de marge (Hocking et al., 2013)
- **TransfoTree :** un algorithme d'arbre de décision pour données censurées n'utilisant pas de marge (Hothorn et Zeileis, 2017)

Comparaison à l'état de l'art

Nous avons comparé notre méthode aux algorithmes suivants :

- **Constant :** utilise l'ensemble d'entraînement pour trouver la fonction constante qui traverse le plus d'intervalles possible
- **Interval-CART :** un arbre de régression (Breiman et al., 1984) où chaque exemple $(\mathbf{x}_i, \mathbf{y}_i) \in S$ a été remplacé par deux exemples de régression standard : $(\mathbf{x}_i, \underline{y_i} + \epsilon)$ et $(\mathbf{x}_i, \overline{y_i} - \epsilon)$.
- **L1-Linear :** régression linéaire par intervalle à base de marge (Hocking et al., 2013)
- **TransfoTree :** un algorithme d'arbre de décision pour données censurées n'utilisant pas de marge (Hothorn et Zeileis, 2017)

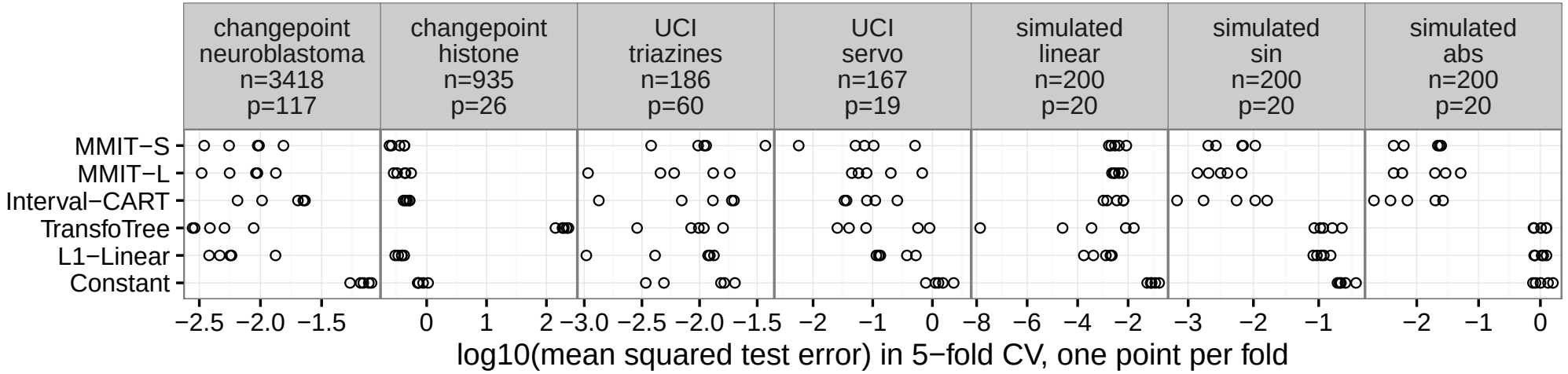
Comparaison à l'état de l'art

Nous avons comparé notre méthode aux algorithmes suivants :

- **Constant :** utilise l'ensemble d'entraînement pour trouver la fonction constante qui traverse le plus d'intervalles possible
- **Interval-CART :** un arbre de régression (Breiman et al., 1984) où chaque exemple $(\mathbf{x}_i, \mathbf{y}_i) \in S$ a été remplacé par deux exemples de régression standard : $(\mathbf{x}_i, \underline{y_i} + \epsilon)$ et $(\mathbf{x}_i, \overline{y_i} - \epsilon)$.
- **L1-Linear :** régression linéaire par intervalle à base de marge (Hocking et al., 2013)
- **TransfoTree :** un algorithme d'arbre de décision pour données censurées n'utilisant pas de marge (Hothorn et Zeileis, 2017)

Comparaison à l'état de l'art

Résultats pour 5 ensembles de test par ensemble de données :



Métrique utilisée pour la comparaison :

$$\mathrm{MSE}(h,S) = \frac{1}{n}\sum_{i=1}^{n}\left([h(\mathbf{x}_i)-\underline{y_i}]I[f(\mathbf{x}_i)<\underline{y_i}]+[h(\mathbf{x}_i)-\overline{y_i}]I[f(\mathbf{x}_i)>\overline{y_i}]\right)^2$$

Conclusion

Conclusion

- Nous avons proposé un nouvel algorithme d'arbre de décision pour le paradigme de la régression par intervalles
- Nous avons montré que les arbres pouvaient être entraînés en solutionnant une série de problèmes convexes
- Nous avons proposé un algorithme de programmation dynamique efficace pour cette tâche
- Notre algorithme se compare favorablement à l'état de l'art

Conclusion

- Nous avons proposé un nouvel algorithme d'arbre de décision pour le paradigme de la régression par intervalles
- Nous avons montré que les arbres pouvaient être entraînés en solutionnant une série de problèmes convexes
- Nous avons proposé un algorithme de programmation dynamique efficace pour cette tâche
- Notre algorithme se compare favorablement à l'état de l'art

Conclusion

- Nous avons proposé un nouvel algorithme d'arbre de décision pour le paradigme de la régression par intervalles
- Nous avons montré que les arbres pouvaient être entraînés en solutionnant une série de problèmes convexes
- Nous avons proposé un algorithme de programmation dynamique efficace pour cette tâche
- Notre algorithme se compare favorablement à l'état de l'art

Conclusion

- Nous avons proposé un nouvel algorithme d'arbre de décision pour le paradigme de la régression par intervalles
- Nous avons montré que les arbres pouvaient être entraînés en solutionnant une série de problèmes convexes
- Nous avons proposé un algorithme de programmation dynamique efficace pour cette tâche
- Notre algorithme se compare favorablement à l'état de l'art

Merci !

Questions ?