Plan du cours

1) Introduction au machine learning

2) Régularisation et forêts aléatoires

3) Réseau de neurones

4) Réseau de neurones convolutifs

Pour chaque séance:

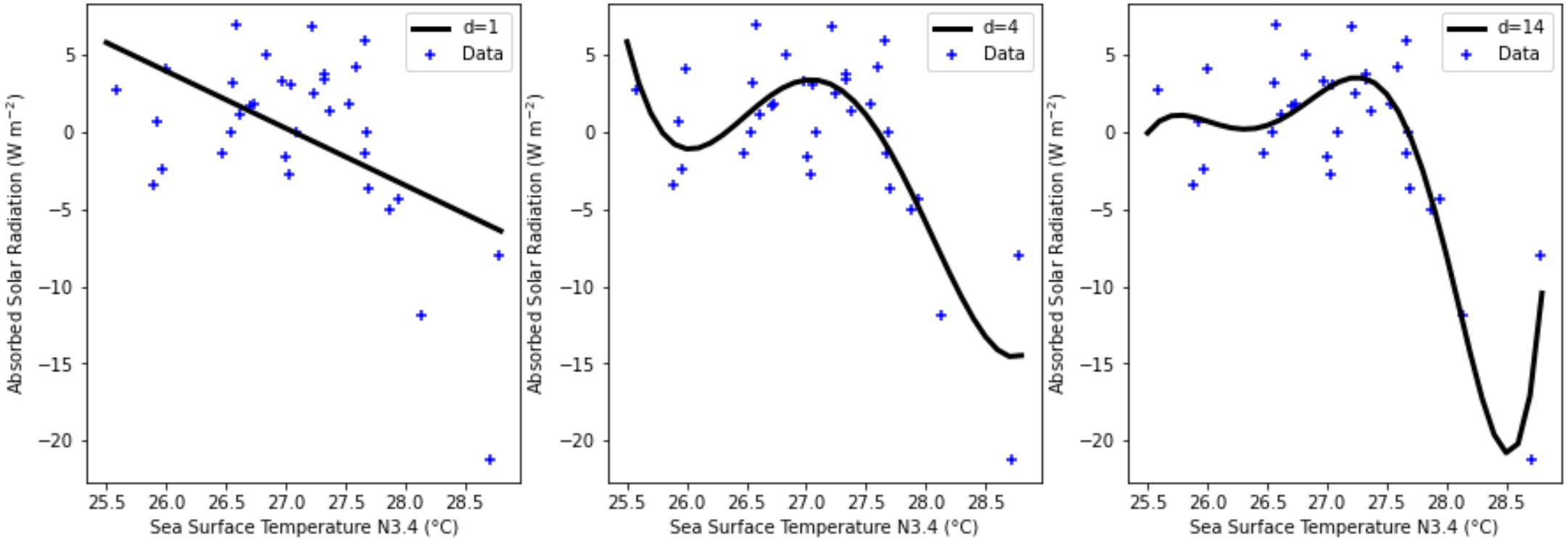
1h de cours / support transparent

2h de travaux pratiques (amener un ordinateur portable)

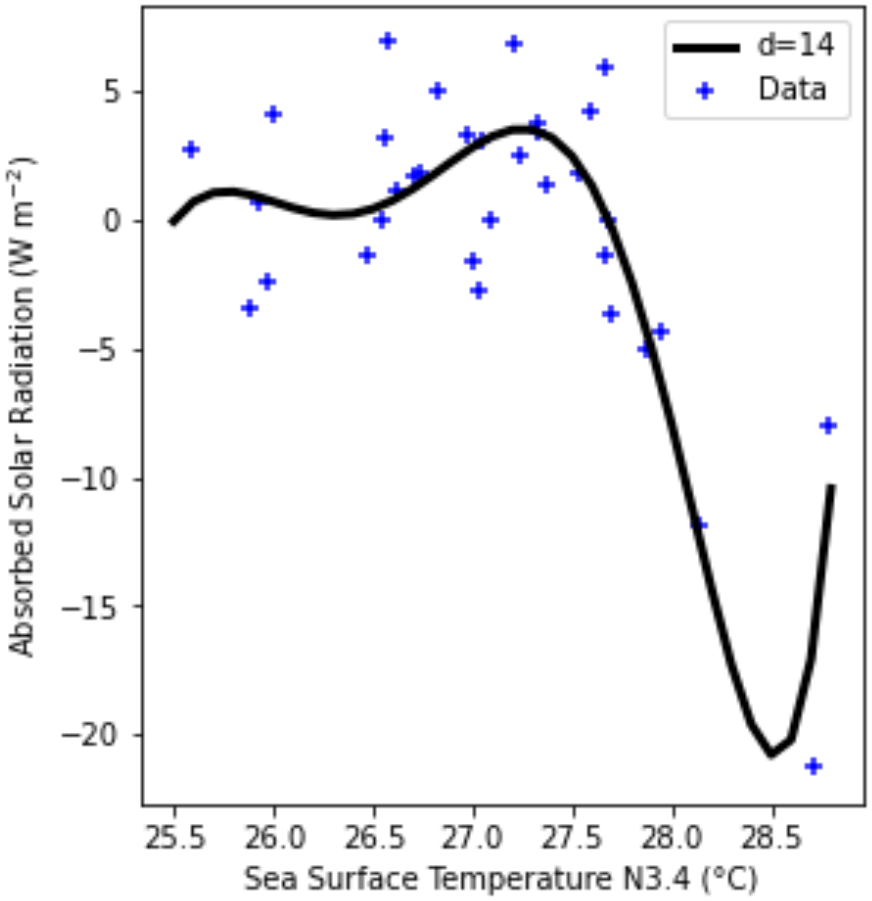
Régularisation

Reprenons l'exemple de la régression polynomiale :

$$\widehat{y_i} = \theta_0 + \theta_1 x_i + \theta_2 {x_i}^2 + \cdots + \theta_d {x_i}^d$$



Régularisation



- Dans une situation de sur-apprentissage, le modèle cherche à expliquer la partie aléatoire du signal (le bruit).
- Les paramètres ont alors tendance à augmenter.

Régularisation

- La méthode des moindres carrés sont basés sur la minimisation de: $\sum_{i=1}^{n} (\hat{y}_i - y_i)^2$
- En machine learning, on cherche ainsi à minimiser une fonction mathématique pour obtenir les paramètres $\boldsymbol{\theta}$. Celle-ci s'appelle la fonction cout et est notée $J(\boldsymbol{\theta})$.

$$\boldsymbol{\theta} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\, J(\boldsymbol{y}, \boldsymbol{X}, \boldsymbol{\theta})$$

- Pour réduire le risque de sur-apprentissage, on peut effectuer une régularisation. L'idée est de *pénaliser* les modèles avec des grands paramètres.
- Ainsi, pour une regression polynomiale au lieu de chercher à réduire:

$$J(\boldsymbol{\theta}) = \sum_{i=1}^{n} (\hat{y}_i - y_i)^2 = L_2(\hat{\boldsymbol{y}} - \boldsymbol{y})$$

On réduit:

$$J(\boldsymbol{\theta}) = \frac{1}{n} \sum_{i=1}^{n} (\hat{y}_i - y_i)^2 + \alpha\, P(\boldsymbol{\theta})$$

Régularisation

$$J(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n}(\widehat{y_i} - y_i)^2 + \ \alpha\, P(\boldsymbol{\theta})$$

Il existe plusieurs types de régularisation:

- Régularisation Ridge (L_2) avec : $P(\boldsymbol{\theta}) = \sum_{i=0}^{n} {\theta_i}^2$
- Régularisation Lasso (L_1) avec : $P(\boldsymbol{\theta}) = \sum_{i=0}^{n} |\theta_i|$
- Régularisation Elastic Net combinant Ridge et Lasso

α est un hyperparamètre.

Estimation des paramètres

En machine learning les paramètres sont le plus souvent estimés à l'aide d'une descente de gradient.

Il s'agit d'un problème d'optimisation, car on souhaite minimiser $J(\boldsymbol{\theta})$. Dans certains cas, il n'existe pas de solutions directes pour ce problème. Il faut alors faire une descente de gradient.

Pour cela, on calcule : $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \begin{pmatrix} \frac{\partial J}{\partial \theta_0} \\ \vdots \\ \frac{\partial J}{\partial \theta_m} \end{pmatrix}$

Estimation des paramètres

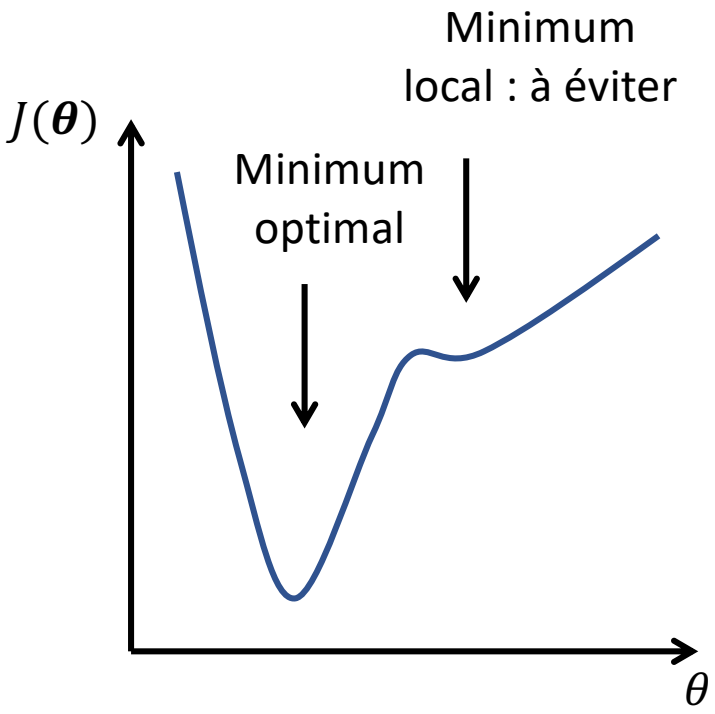
La descente de gradient est un processus itératif qui consiste à modifier successivement les paramètres $\boldsymbol{\theta}$.

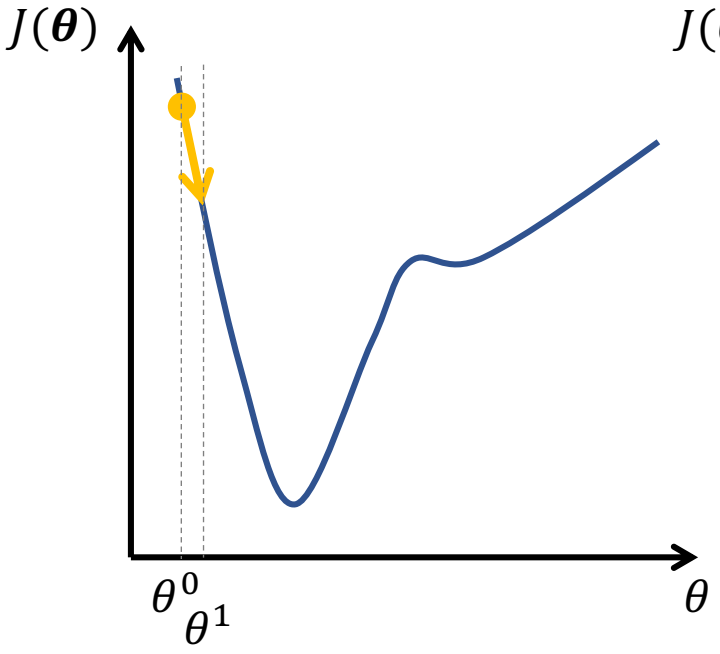
On part de valeurs aléatoires pour les θ_i, appelés θ_i^0.

On modifie alors les paramètres par : $\boldsymbol{\theta}^{k+1} = \boldsymbol{\theta}^k - \eta \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$

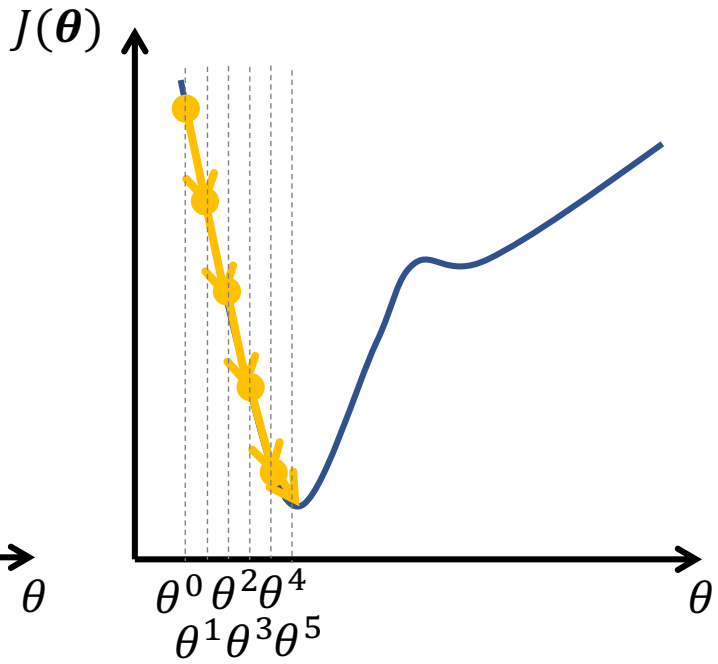
Le pas utilisé ici est noté η ($\eta > 0$) et est le *learning rate*.

Illustration : descente de gradient



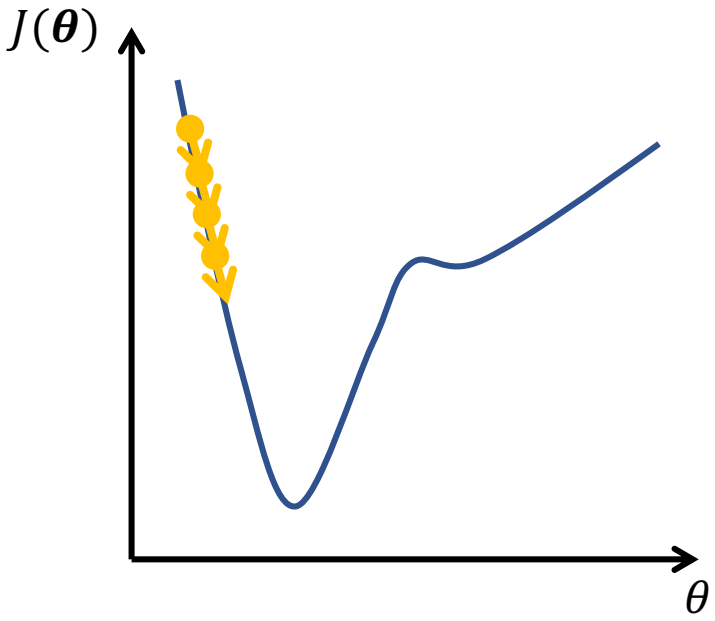


On a : $\frac{\partial J}{\partial \theta} < 0$ donc $-\eta \frac{\partial J}{\partial \theta} > 0$

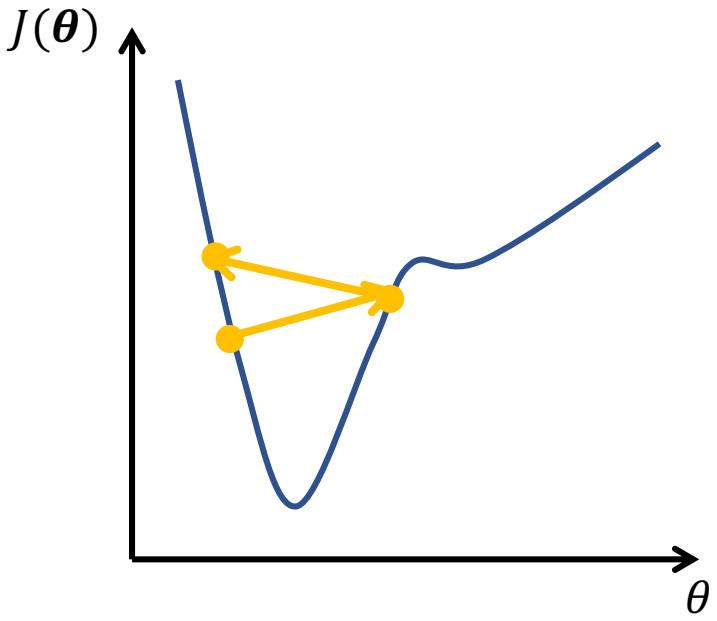


Au bout de 5 itérations…

Illustration : descente de gradient



Si learning rate trop petit : convergence plus lente et risque d'obtenir un minimum local.



Si learning rate trop grand : plus de converge vers le minimum local.

Régularisation Ridge

$$J(\boldsymbol{\theta}) = J_{sr}(\boldsymbol{\theta}) + \alpha \sum_{i=0}^{m} {\theta_i}^2$$

Fonction loss sans régularisation

La descente de gradient est alors implémentée avec :

$$\boldsymbol{\theta}^{k+1} = \boldsymbol{\theta}^{k} - \eta\left(\nabla_{\boldsymbol{\theta}} J_0(\boldsymbol{\theta}) + 2\alpha\boldsymbol{\theta}^{k}\right)$$
$$\boldsymbol{\theta}^{k+1} = \boldsymbol{\theta}_{sr} - 2\alpha\boldsymbol{\theta}^{k}$$

Régularisation Lasso

$$J(\boldsymbol{\theta}) = J_{sr}(\boldsymbol{\theta}) + \alpha \sum_{i=0}^{m} |\theta_i|$$

Fonction non-dérivable en $\theta_i = 0$

Fonction loss sans régularisation

On appelle $\boldsymbol{\theta}_{sr}$ une itération des paramètres obtenus sans régularisation:

$$\boldsymbol{\theta}_{sr} = \boldsymbol{\theta}^k - \eta \nabla_{\boldsymbol{\theta}} J_{sr}(\boldsymbol{\theta})$$

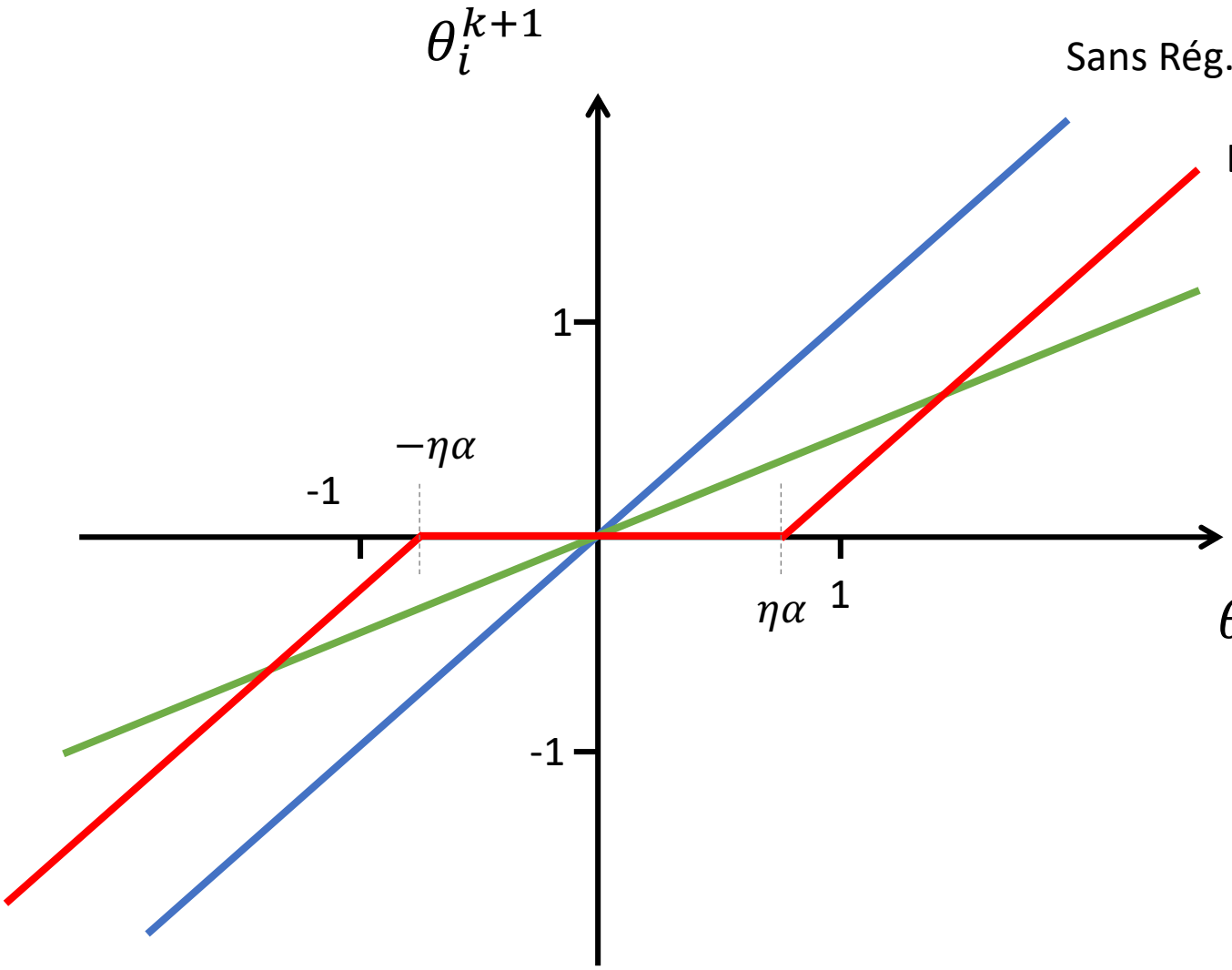
La descente de gradient est alors implémentée avec :

$$\theta_i^{k+1} = \theta_i^k - \eta \frac{\partial J_{sr}}{\partial \theta_i} - \eta\alpha = \theta_{sr} - \eta\alpha \quad \text{si } \theta_{sr} > \eta\alpha$$

$$\theta_i^{k+1} = 0 \quad \text{si } -\eta\alpha < \theta_{sr} < \eta\alpha$$

$$\theta_i^{k+1} = \theta_i^k - \eta \frac{\partial J_{sr}}{\partial \theta_i} + \eta\alpha = \theta_{sr} + \eta\alpha \quad \text{si } \theta_{sr} < -\eta\alpha$$

Illustration



Sans Rég. : $\theta_i^{k+1} = \theta_{sr}$

Lasso : $\theta_i^{k+1} = \begin{cases} \theta_{sr} - \eta\alpha \\ 0 \\ \theta_{sr} + \eta\alpha \end{cases}$

Ridge : $\theta_i^{k+1} = \theta_{sr} - 2\alpha\eta$

Régularisation

Utilisé en machine learning pour trouver les paramètres.

Ridge (L_2) :

- Evite le surapprentissage
- Utilise toutes les caractéristiques de $\boldsymbol{X}$ et peut être inutile dans le cas où $\boldsymbol{X}$ a beaucoup de caractéristiques.

Lasso (L_1) :

- Réduit le nombre de prédicteurs et donc de caractéristiques utilisés,
- Dans le cas ou plusieurs caractéristiques sont correlées, sélectionne arbitrairement une de celles-ci.

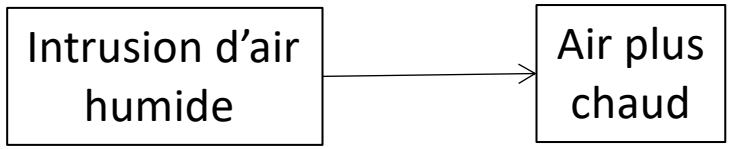
Foret aléatoire

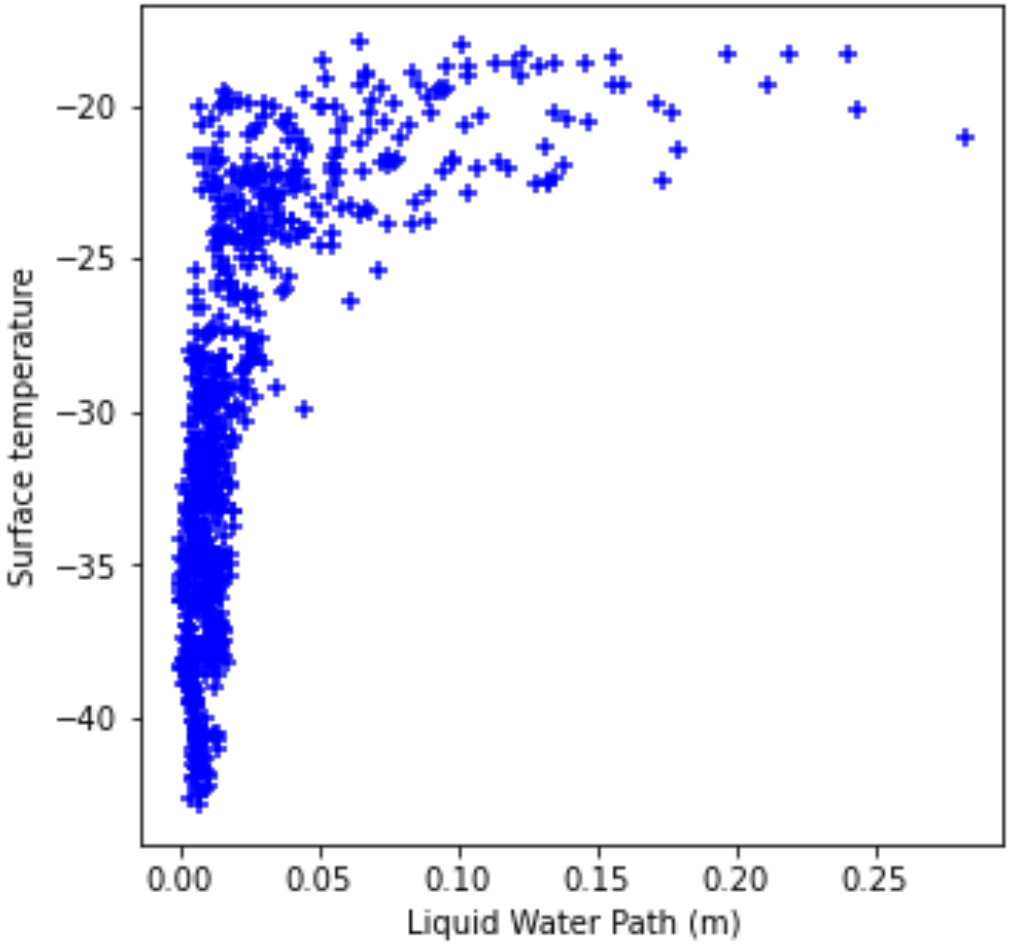
- Méthode pouvant utiliser des variables qualitative et quantitatives en tant que features. Peut être utiliser pour de la classification ou de la regression.
- Méthode basée sur des arbres de décision.

Illustration

Données: campagne SHEBA en Arctique.

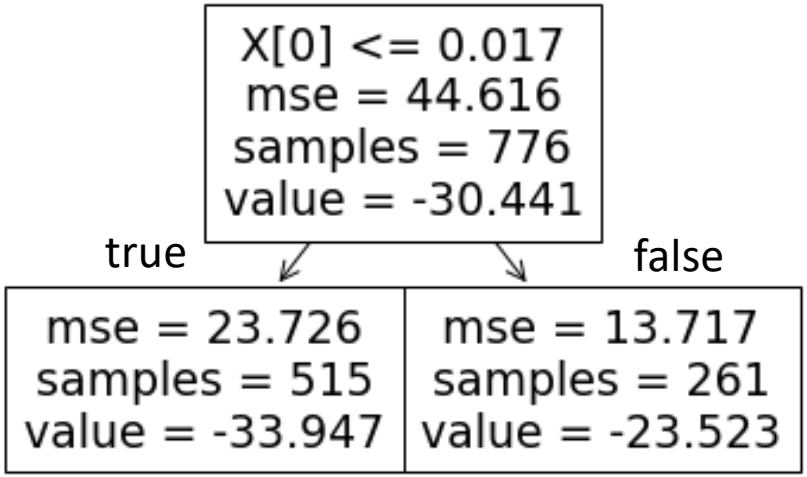
- X = Contenue en vapeur d'eau de l'atmosphère.
- y = Température de surface observée

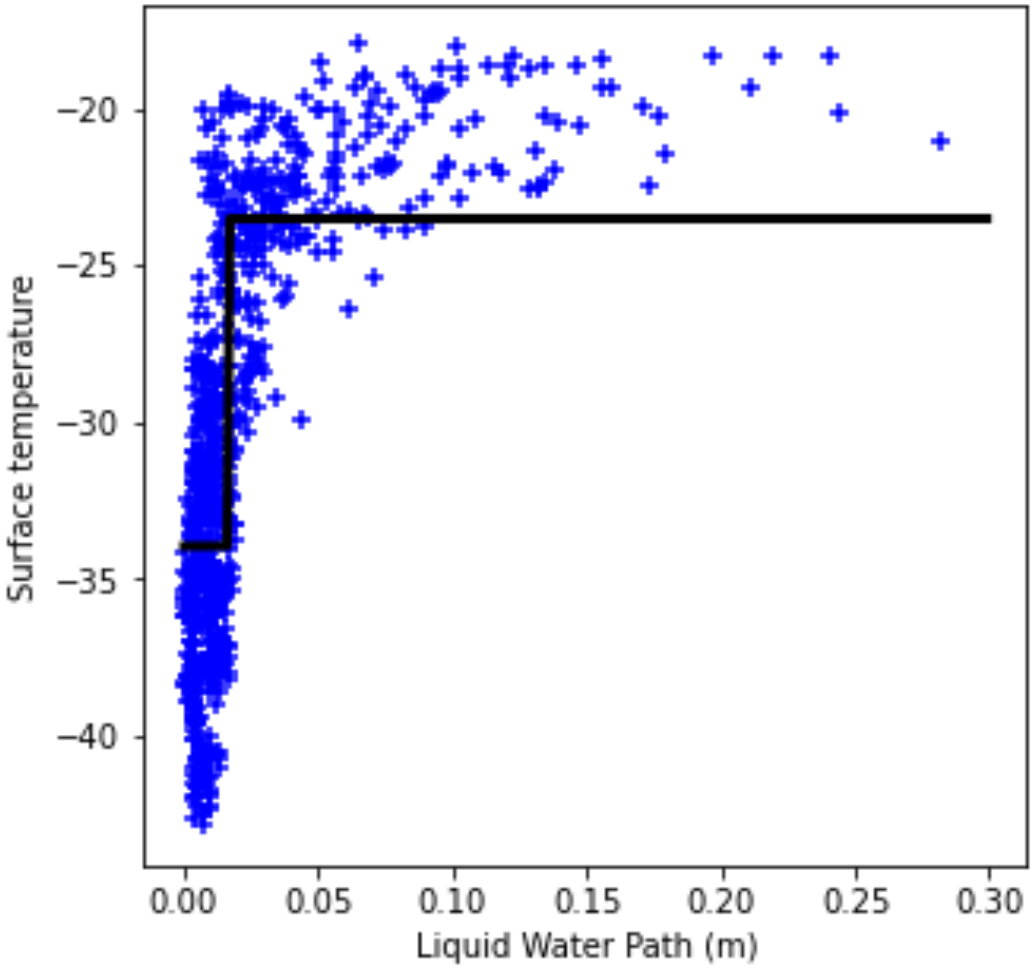




Illustration

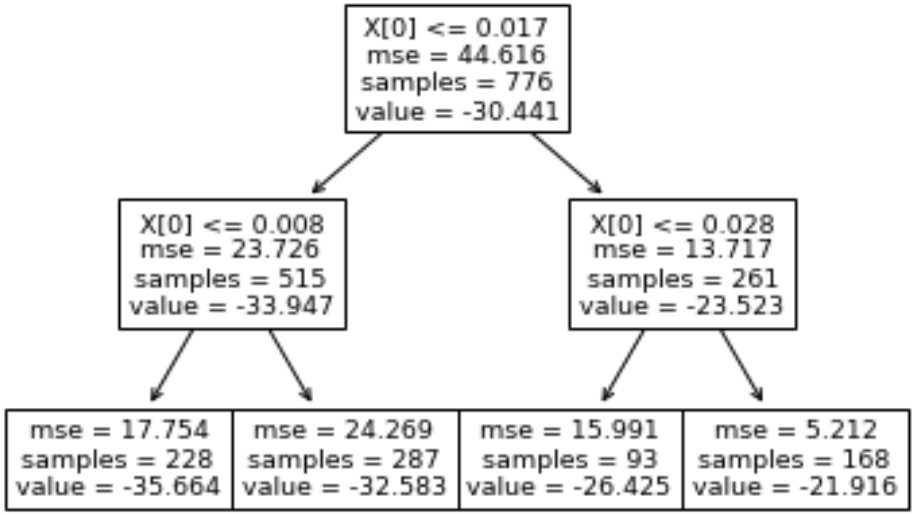
On construit un arbre de décision à partir des valeurs de X.
-> on les construit à partir de l'Entropie et de la réduction d'information effectuée.

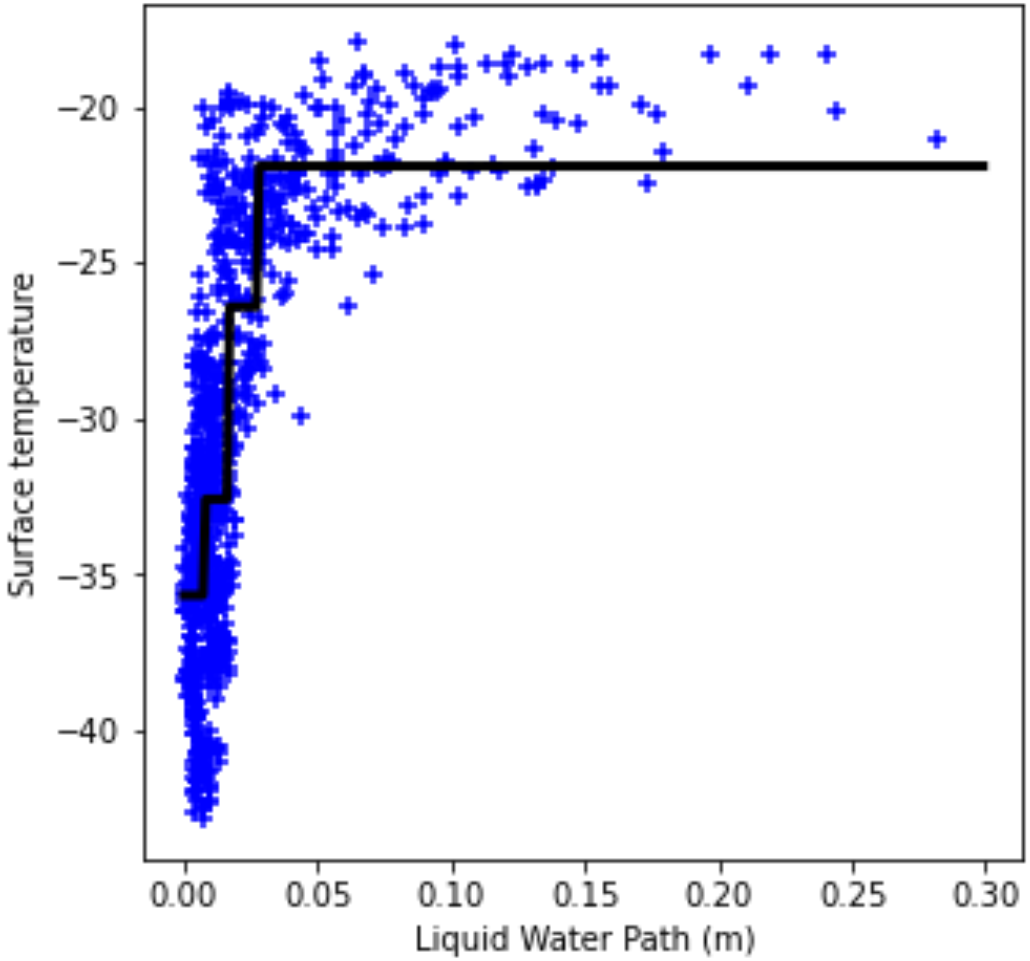




Illustration

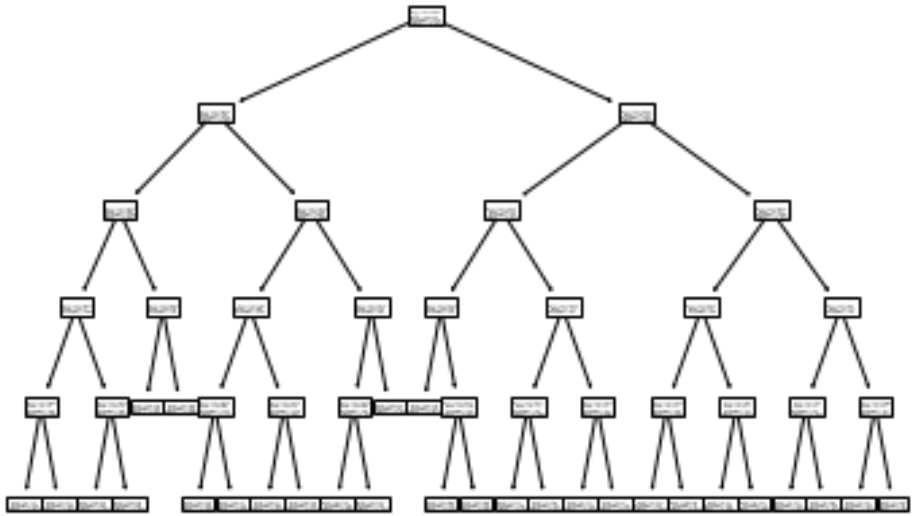
Augmentons la profondeur de l'arbre à 2.

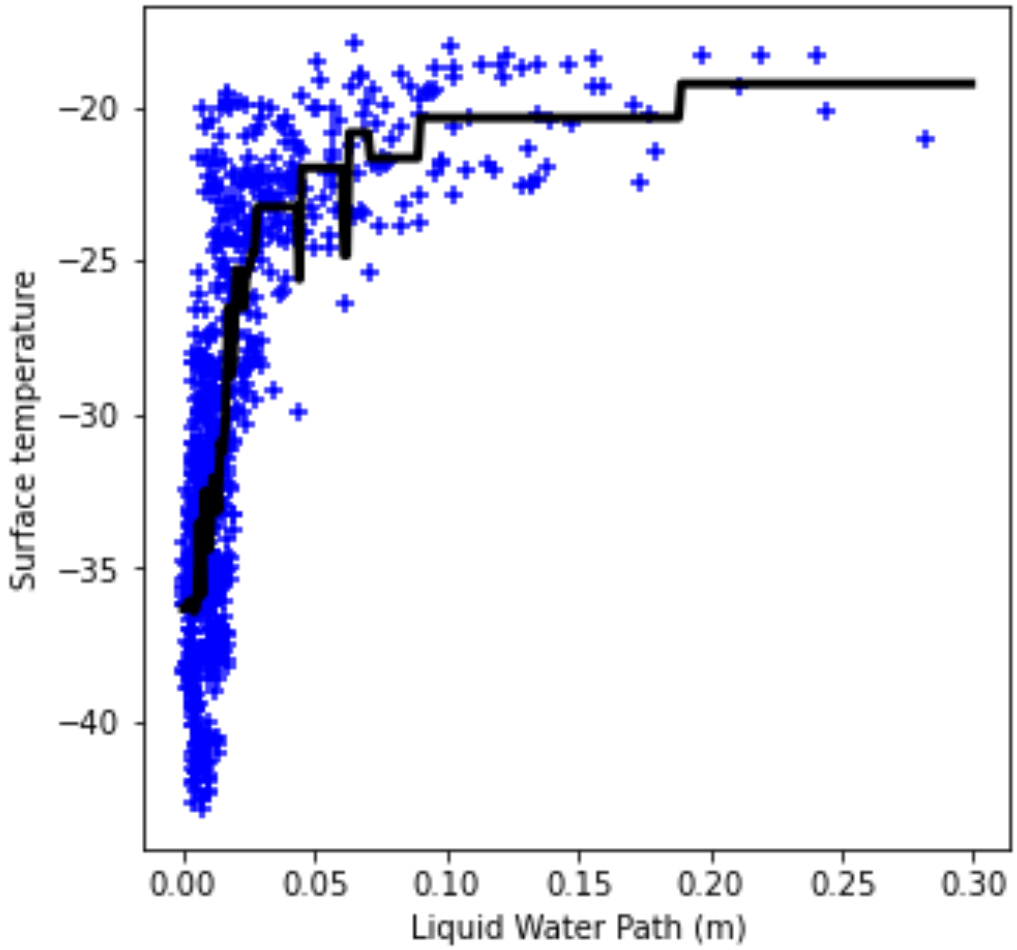




Illustration

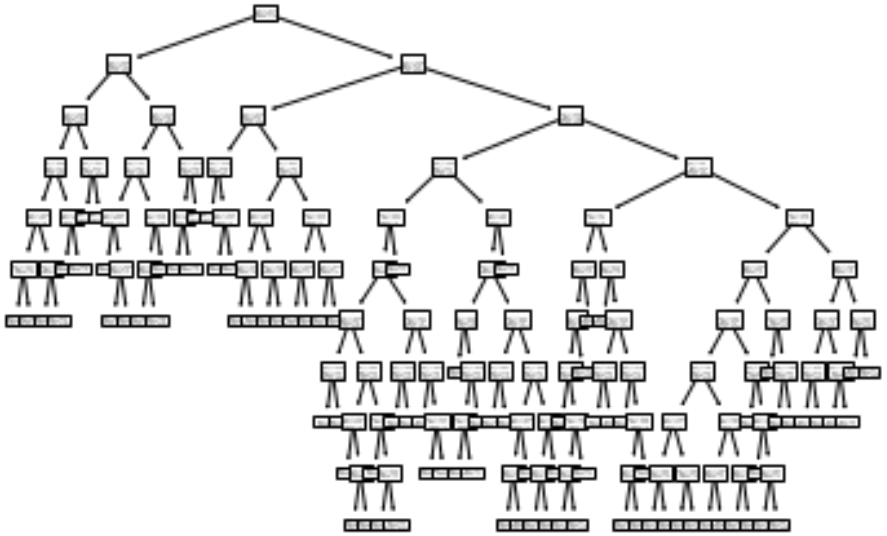
Profondeur = 5

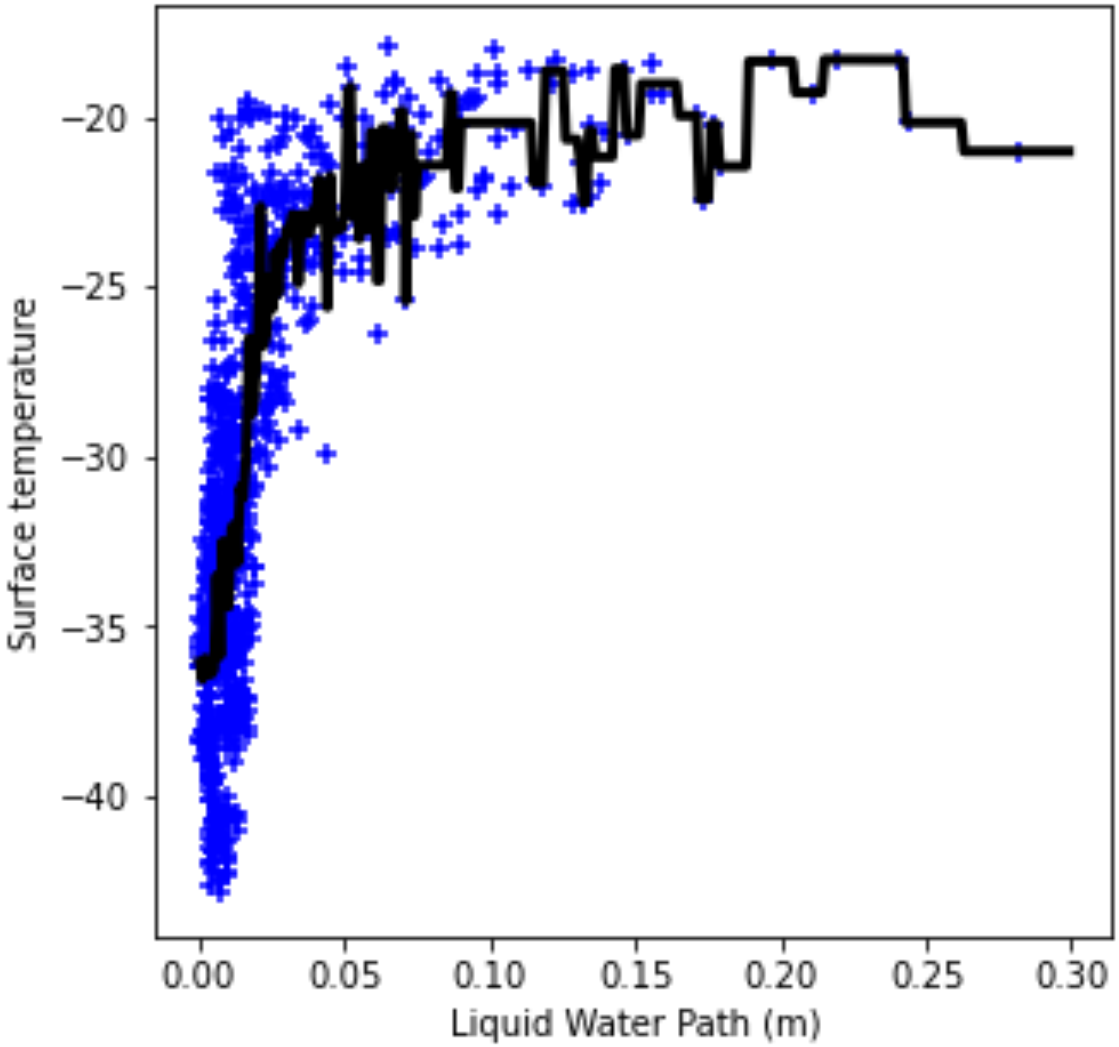




Illustration

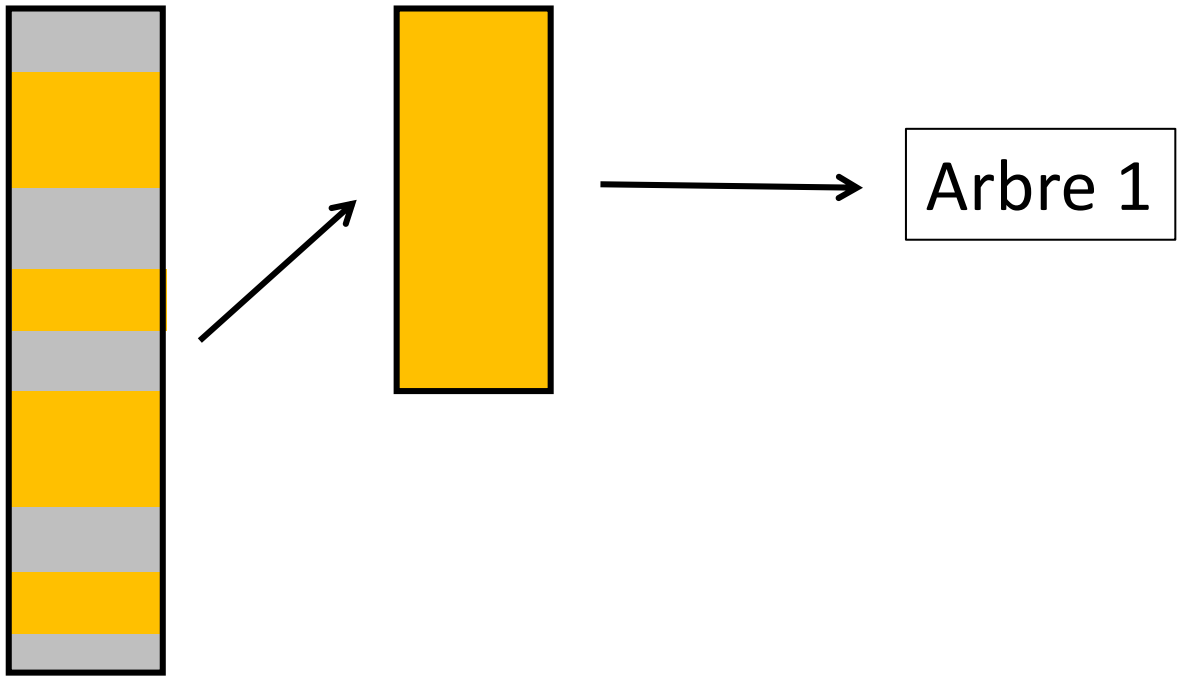
Profondeur = 10





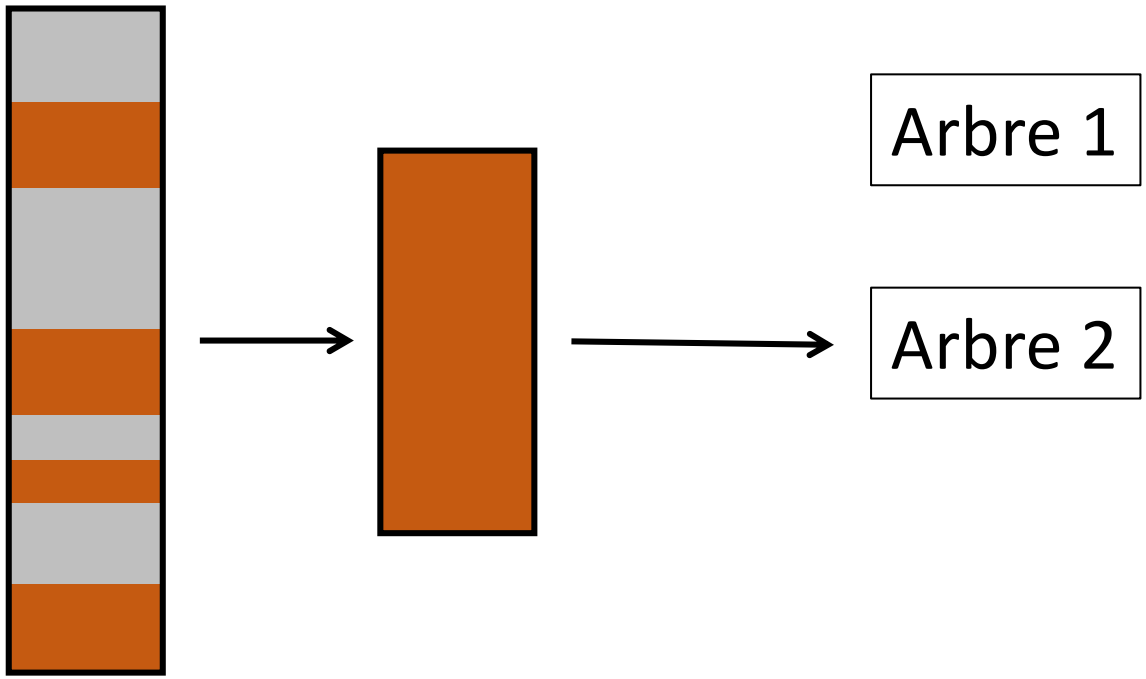
Arbre de décision et foret aléatoire

- Les arbres décisionnels ont l'inconvénient de surapprendre très vite.
- La solution est la foret aléatoire.



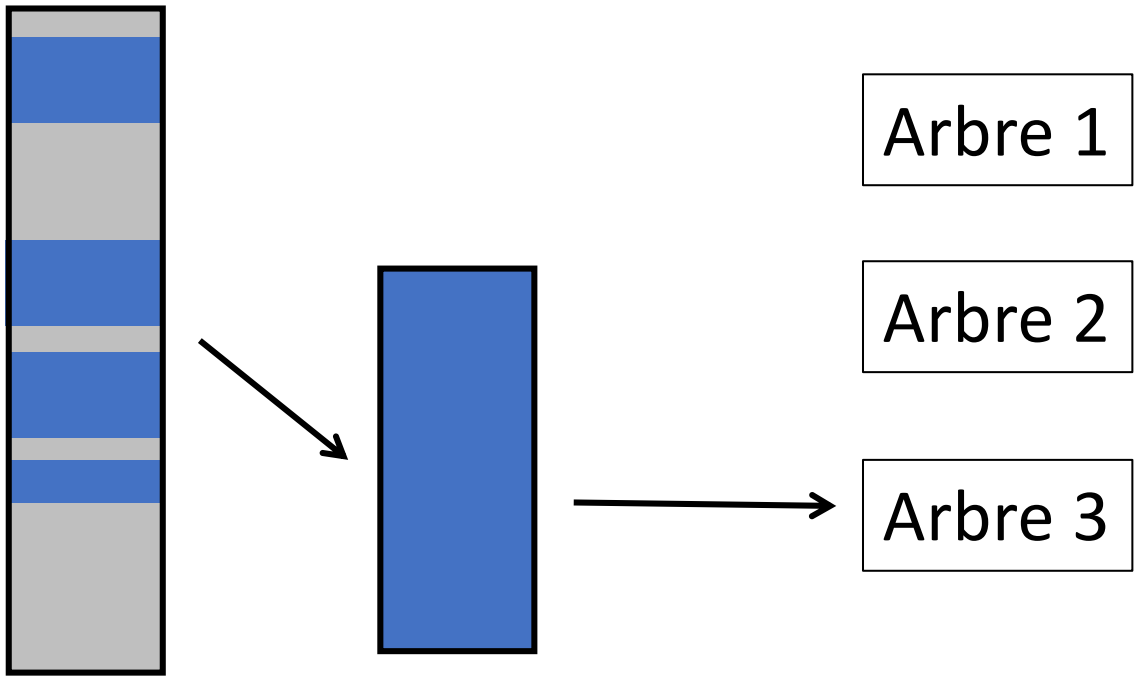
Arbre de décision et foret aléatoire

- Les arbres décisionnels ont l'inconvénient de surapprendre très vite.
- La solution est la foret aléatoire.



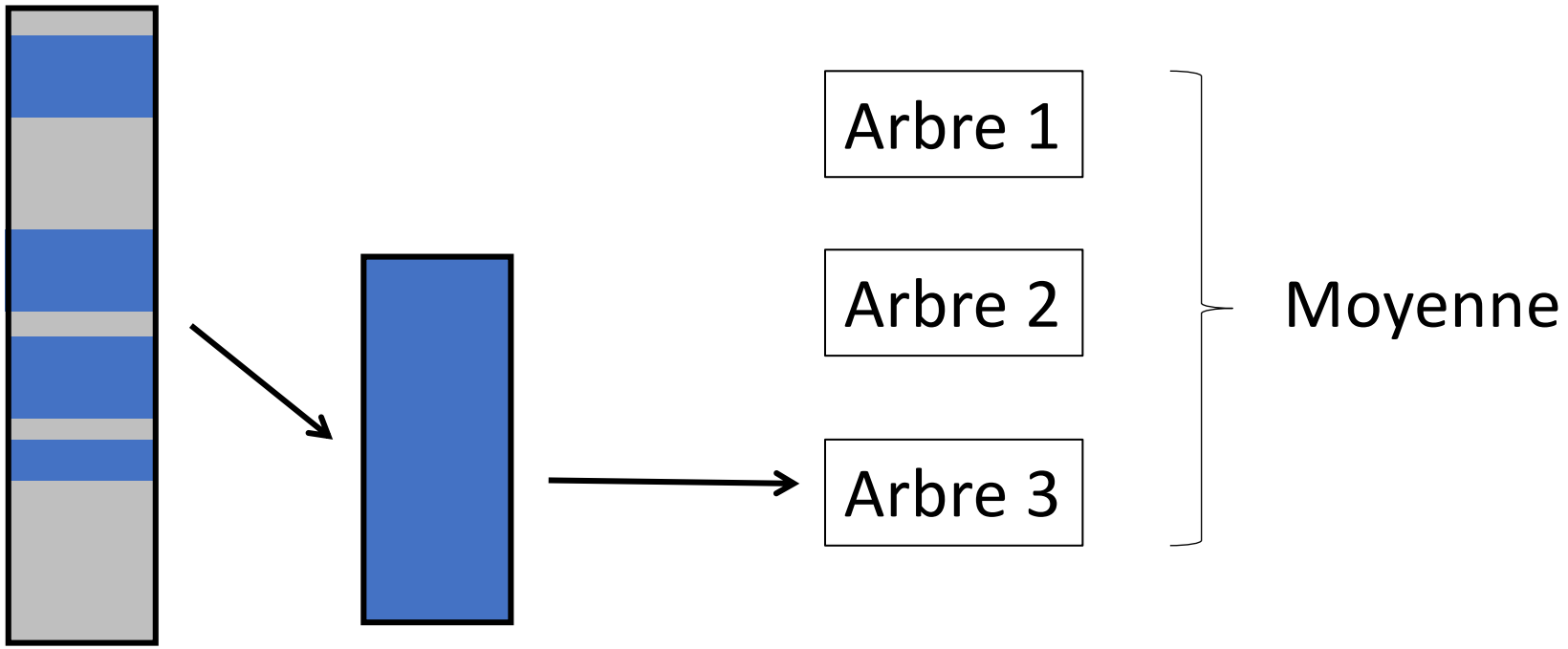
Arbre de décision et foret aléatoire

- Les arbres décisionnels ont l'inconvénient de surapprendre très vite.
- La solution est la foret aléatoire.



Arbre de décision et foret aléatoire

- Les arbres décisionnels ont l'inconvénient de surapprendre très vite.
- La solution est la foret aléatoire.

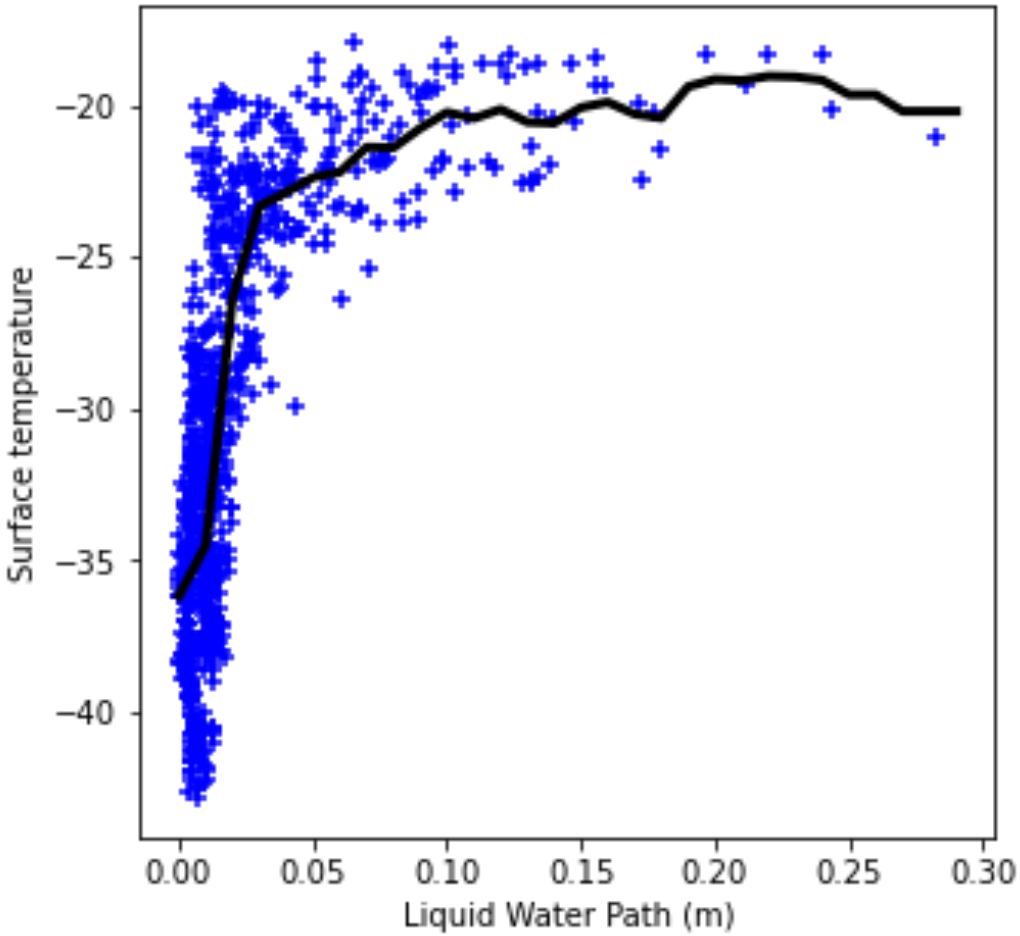


Illustration

Le résultat de la forêt aléatoire dépend de :

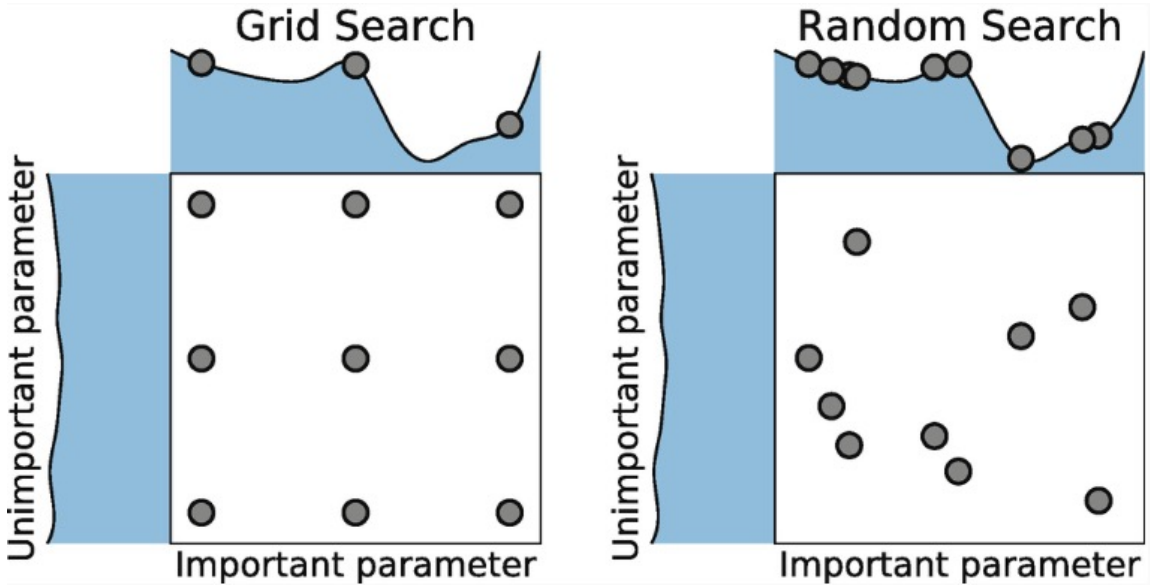
- du nombre d'arbres utilisé (`n_estimator`),
- du nombre de features utilisé pour chaque étape (`max_features`). Une valeur grande réduit l'erreur, mais augmente le risque de surapprentissage.
- du nombre de donnée minimal à utiliser pour réaliser une étape (`min_samples`). Un nombre de 2 réalise des arbres pleinement développés. Un petit nombre augmente le risque de surapprentissage.

Ce sont les hyperparamètres.



Choisir les hyperparamètres

- Il faut les choisir sur une base de donnée de validation ou en utilisant une validation croisée.
- On peut spécifier une liste d'hyperparamètre à utiliser, et garder le meilleur modèle avec grid ou random Search.



Grid Search

- Entrainement d'un modèle pour chaque combinaison d'hyperparamètre.
- La meilleur combinaison évaluée avec validation croisée est retenue

- ➢ Recherche exhaustive, peut être mené en parallèle
- ➢ Peu adapté pour certain hyperparamètres quantitatifs,
- ➢ Couteux (par exemple pour un choix de 8 hyperparamètres parmis 8 : 8^8 = 16 777 216 modèles à entrainer.

Random Search

- on définit un intervalle pour chaque hyperparamètres,
- Pour chaque hyperparamètre, on définit une liste de valeurs à tester ou une loi de probabilité pour générer des valeurs aléatoires.
- On entraîne alors n modèle à partir d'une liste de n combinaisons d'hyperparamètre générée aléatoirement.

➢ Recherche moins exhaustive, peut être mené en parallèle

➢ Moins couteux (n modèles à entrainer).

Optimisation Bayesienne

- On calcule une fonction d'acquisition, estimant une probabilité d'amélioration du modèle.
- On entraîne un modèle succesivement pour les maximum de la fonction d'acquisition.