

Apprentissage supervisé:

Algorithmes

de « résolution »

Mourad NACHAOUI

FST Béni-Mellal

Sommaire

1. Introduction

2. Optimisation

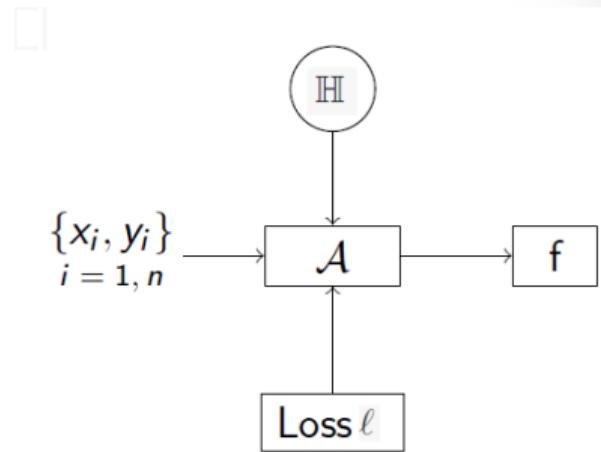
- Gradient descente
- Fonctions convexes



Apprentissage supervisé et optimisation

Composantes d'apprentissage automatique

- programme \mathcal{A} :
algorithme \mathcal{A}
- expérience E_n :
données $\{x_i, y_i\}_{i=1, n}$
- performance ℓ :
coût : ℓ
- tâche - hypothèses \mathbb{H} :
modèle $f \in \mathbb{H}$



Optimisation

$$\min_{f \in \mathbb{H}} \ell(f, \{x_i, y_i\}, i = 1, \dots, n)$$

$$\hat{f} = \arg \min_{f \in \mathbb{H}} \ell = \mathcal{A}(\mathbb{H}, \{x_i, y_i\}_{i=1, \dots, n}, \ell)$$

Minimisation du risque empirique

- On cherche : prédicteur minimisant le risque empirique

$$\hat{f}_n \in \arg \min_{f \in \mathcal{S}} \hat{R}_n(f)$$

sur une classe \mathcal{S} de fonctions ($\mathcal{S} \in \mathbb{H}$)

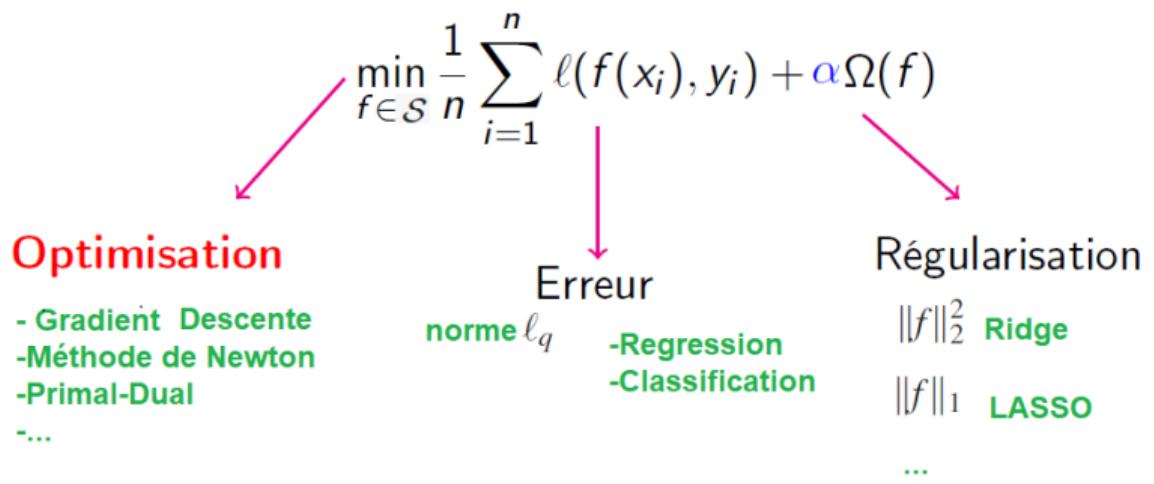
- pas trop petite pour pouvoir contenir une bonne approximation du prédicteur de Bayes
- pas trop grande pour éviter le sur-apprentissage (coller trop aux données, mal généraliser)
- et pour laquelle on sait "résoudre" le problème d'optimisation

L'apprentissage supervisé : cas général

Données d'apprentissage : $(X_i, y_i)_{i=1,\dots,n} \in (\mathcal{X} \times \mathcal{Y})^n$.

Modèle : $y = f(x)$, pour un certain $f \in \mathcal{S}$.

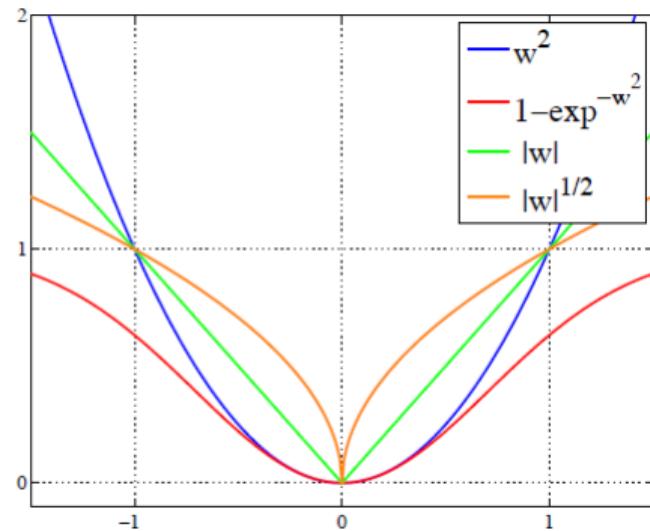
Problème à résoudre pour l'apprentissage :



Trouver un modèle qui permet d'accomplir la généralisation, contrôler la complexité, réduire l'erreur, et avec un temps raisonnable.

Exemples de différentes pénalités utilisées en apprentissage

Pénalité	Convexe	Diff.
$\omega(w) = w^2$ $\Omega(x) = \ x\ ^2$	✓	✓
$\omega(w) = 1 - \exp(-w^2)$	✗	✓
$\omega(w) = w $ $\Omega(x) = \ x\ _1$	✓	✗
$\omega(w) = \sqrt{ w }$ $\Omega(x) = \ x\ _{1/2}$	✗	✗



$$R(f) = \mathbb{E}(L(f(X), Y)) \quad \hat{R}_n(f) = \frac{1}{n} \sum_{i=1}^n L(f(x_i), y_i)$$

L'idéal : $f^* = \operatorname{argmin}_f R(f) \Rightarrow R(f^*) \leq R(\hat{f})$
 Ce qu'on a : $\hat{f} = \operatorname{argmin}_f \hat{R}_n(f) \Rightarrow \hat{R}_n(\hat{f}) \leq \hat{R}_n(f^*)$

$$R(\hat{f}) - R(f^*) < \underbrace{|R(\hat{f}) - \hat{R}_n(\hat{f})|}_{\leq \varepsilon} + \underbrace{|\hat{R}_n(f^*) - R(f^*)|}_{\leq \varepsilon}$$

$$\mathbb{P}\left(\sup_{f \in V(\mathcal{S})} |R(\hat{f}) - \hat{R}_n(\hat{f})| \leq \varepsilon\right) \geq \delta$$

Convergence uniforme localisée (Mendelson, Bousquet, Tsybakov, Massart)

$$\varepsilon(n, C(\mathcal{S})) \propto \mathcal{O}\left(\sqrt{\frac{C(\mathcal{S})}{n} \log\left(\frac{n}{C(\mathcal{S})}\right)}\right)$$

$C(\mathcal{S})$ = mesure de la complexité de \mathcal{S}

regression ridge

On introduit un terme de pénalisation:

$$\hat{\theta}_n^\alpha \in \operatorname{argmin}_{\theta \in \mathbb{R}^d} \left(\underbrace{\sum_{i=1}^n (Y_i - \langle X_i | \theta \rangle)^2}_{\text{terme d'attache aux données}} + \underbrace{\alpha \sum_{k=1}^d \theta_k^2}_{\|\theta\|^2} \right)$$

α est appelé paramètre de régularisation.

Interprétation: Il existe un réel $b > 0$ tel que si

$$\tilde{\mathcal{S}}_b = \{g : g(x) = \langle x | \theta \rangle, \|\theta\| \leq b\}$$

alors

$$\hat{\theta}_n^\alpha \in \operatorname{argmin}_{g \in \tilde{\mathcal{S}}_b} \frac{1}{n} \sum_{i=1}^n (Y_i - g(X_i))^2$$

Le problème peut se réécrire matriciellement:

$$\hat{\theta}_n^{\alpha\text{-ridge}} \in \operatorname{argmin}_{\theta \in \mathbb{R}^d} \left(\|Y - \mathbf{X}\theta\|^2 + \alpha \|\theta\|^2 \right)$$

→ Pénalisation par la norme euclidienne

$$\|\theta\|^2 = \sum_{k=1}^d \theta_k^2.$$

Solution exacte:

$$\hat{\theta}_n^{\alpha\text{-ridge}} = (\mathbf{X}^T \mathbf{X} + \alpha \operatorname{Id})^{-1} \mathbf{X}^T Y$$

$$\mathcal{X} = \mathbb{R}^d.$$

$$\hat{\theta}_n^{\alpha\text{-ridge}} = \left(\mathbf{X}^T \mathbf{X} + \alpha \text{Id}\right)^{-1} \mathbf{X}^T Y$$

La matrice $\mathbf{X}^T \mathbf{X} + \alpha \text{Id}$ est de taille $d \times d$

→ inversion coûteuse si d est très grand

Hypothèse de parcimonie: Il existe $d_0 \ll d$ tel que le vecteur θ n'a que d_0 composantes non nulles

Si on connaît le **support de θ** (l'emplacement des composantes non nulles), le calcul de $\hat{\theta}_n^{\alpha\text{-ridge}}$ est moins coûteux et par exemple

$$\mathbb{E} \left[R \left(\hat{\theta}_n^0 \right) - R \left(\theta \right) \right] \leq C \times \frac{\sigma^2 d_0}{n}.$$

Idée: Introduire une régularisation pour forcer la solution du problème à n'avoir que peu de composantes non nulles

- ▶ Régularisation par la norme ℓ_0

$$\|\theta\|_0 = \text{Card}\{k : \theta_k \neq 0\}$$

$$\hat{\theta}_n^\alpha \in \underset{\theta \in \mathbb{R}^d}{\operatorname{argmin}} \left(\|Y - \mathbf{X}\theta\|^2 + \alpha \|\theta\|_0 \right)$$

- **Garanties théoriques** : pour le choix $\alpha \simeq \ln(d)/n$, on a

$$\mathbb{E} \left[R \left(\hat{\theta}_n^0 \right) - R \left(\theta \right) \right] \leq C \times \frac{\sigma^2 d_0 \ln(d)}{n}.$$

- **Implémentation** : très complexe pour d grand
(énumérer tous les supports possibles...)

Idée: Introduire une régularisation pour forcer la solution du problème à n'avoir que peu de composantes non nulles

- ▶ Régularisation par la norme ℓ_1

$$\|\theta\|_1 = \sum_{k=1}^d |\theta_k|$$

$$\hat{\theta}_n^{\alpha\text{-LASSO}} \in \operatorname{argmin}_{\theta \in \mathbb{R}^d} \left(\|Y - X\theta\|^2 + \alpha \|\theta\|_1 \right)$$

(**estimateur LASSO**)

Least **A**solute **S**hrinkage and **S**election **O**perator

- favorise la parcimonie
- des algorithmes efficaces pour le calculer

Idée: Introduire une régularisation pour forcer la solution du problème à n'avoir que peu de composantes non nulles

- ▶ Régularisation par la norme ℓ_1

$$\|\theta\|_1 = \sum_{k=1}^d |\theta_k|$$

$$\hat{\theta}_n^{\alpha\text{-LASSO}} \in \operatorname{argmin}_{\theta \in \mathbb{R}^d} \left(\|Y - X\theta\|^2 + \alpha \|\theta\|_1 \right)$$

(estimateur LASSO)

Least Absolute Shrinkage and Selection Operator

- **Garanties théoriques:** pour $\alpha \simeq \ln(d)/n$,

$$\mathbb{E} \left[\|\hat{\theta}_n^{\alpha\text{-LASSO}} - \theta\| \right] \leq Cd_0 \sqrt{\frac{\ln(d)}{n}}.$$

- ▶ Principe plus général: “régulariser” le risque empirique permet d'ajouter des contraintes (de diminuer la taille de \mathcal{S})
- ▶ La régularisation ℓ_1 peut aussi s'appliquer pour la régression logistique ([LogisticRegression](#)), et au-delà
- ▶ Comment résoudre les problèmes d'optimisation associés ?
On peut utiliser des variations de la [descente de gradient](#)
(voir plus tard et dans le cours de Réseaux de Neurones)

Le problème de [choix de paramètre d'algorithme](#) est un problème récurrent en apprentissage.

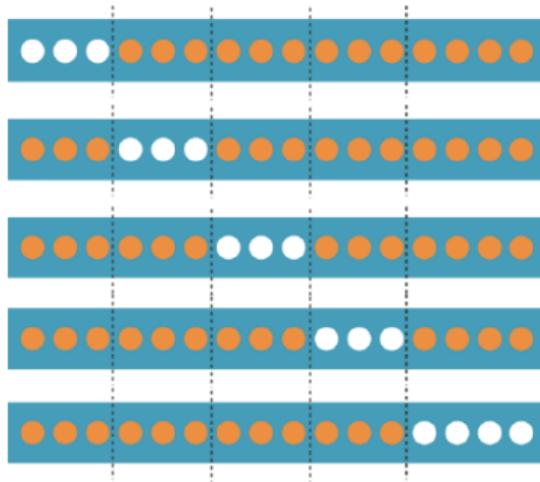
Une solution: **l'approche par validation**

- retirer une partie de la base d'apprentissage (n_v observations) pour en faire une [base de validation](#) sur laquelle tester les prédicteurs $\hat{g}_{n-n_v}^{\alpha}$ pour différents α .

Inconvénient:

- on “sacrifie” des données pour la validation, alors que la qualité du prédicteur augmente avec le nombre de données utilisées pour l’entraînement

Une alternative: la validation croisée



Choisir α minimisant le risque de validation croisée (Cross-Validation)

$$\hat{R}_n^{\text{CV}}(\alpha) = \frac{1}{B} \sum_{b=1}^B \hat{R}_{n_v}^{\text{val}}\left(\hat{g}_{n_e}^{\alpha}\left(\overline{E^{(b)}}\right)\right)$$

où $(E^{(b)})_{b=1}^B$ partition de E_n en B ensembles de taille $n_e = n - n_v$.

(entraîner sur $\overline{E^{(b)}}$, valider sur $E^{(b)}$)

- évite de “sacrifier” des données
- principe général pouvant s'utiliser dès lors qu'il s'agit de sélectionner le paramètre d'un algorithme d'apprentissage

Gradient descente

Pour une fonction $F : \mathbb{R}^d \rightarrow \mathbb{R}$, considérons le problème d'optimisation

$$\min_{\theta \in \mathbb{R}^d} F(\theta)$$

On suppose que l'on nous donne accès à certains «oracles»: l'oracle d'ordre k correspond à l'accès à : $(F(x), F'(x), \dots, F^{(k)})$

Algorithm 1: (Gradient descent (GD))

Choisir $\theta_0 \in \mathbb{R}^d$ et pour $t > 1$, soit

$$\theta_t = \theta_{t-1} - \gamma_t F'(\theta_{t-1}),$$

pour une séquence de taille de pas choisie (potentiellement adaptative)
 $(\gamma_t)_{t>1}$.

Analyse la plus simple: les moindres carrés ordinaires

Soit $\Phi \in \mathbb{R}^{n \times d}$ une matrice de conception et $y \in \mathbb{R}^n$ le vecteur des réponses. L'estimation moindres carrés consiste à trouver un minimiseur θ^* de

$$F(\theta) = \frac{1}{2n} \|\Phi\theta - y\|_2^2$$

Le gradient de F est $F'(\theta) := \nabla_\theta F(\theta) = \frac{1}{n} \Phi^T \Phi \theta - \frac{1}{n} \Phi^T y$. Notons par $H = \Phi^T \Phi \in \mathbb{R}^{d \times d}$, alors le minimiseur θ^* est caractérisé par

$$H\theta^* = \frac{1}{n} \Phi^T y$$

Utilisant le formule de Taylor et le fait que $F'(\theta) = 0$

$$F(\theta) - F(\theta^*) = F'(\theta)^T (\theta - \theta^*) + \frac{1}{2} (\theta - \theta^*)^T H (\theta - \theta^*) = \frac{1}{2} (\theta - \theta^*)^T H (\theta - \theta^*).$$

Mesures de performance

L'itération de gradient avec pas fixe $\gamma_t = \gamma$ est

$$\theta_t = \theta_{t-1} - \gamma F'(\theta_{t-1}) = \theta_{t-1} - \frac{\gamma}{n} \Phi^T (\Phi \theta_{t-1} - y) = \theta_{t-1} - \gamma H(\theta_{t-1} - \theta^*),$$

donc

$$\theta_t - \theta^* = \theta_{t-1} - \theta^* - \gamma H(\theta_{t-1} - \theta^*) = (I - \gamma H)(\theta_{t-1} - \theta^*),$$

par récursivité $\theta_t - \theta^* = (I - \gamma H)(\theta_0 - \theta^*)$,

Deux mesures de performance d'optimisation

$$\|\theta_t - \theta^*\|_2^2 = (\theta_0 - \theta^*)^T (I - \gamma H)^{2t} (\theta_0 - \theta^*)$$

$$F(\theta_t) - F(\theta^*) = (\theta_0 - \theta^*)^T (I - \gamma H)^{2t} H (\theta_0 - \theta^*).$$

convergence de $\|\theta_t - \theta^*\|_2$

Tenant compte la forme $\|\theta_t - \theta^*\|_2$, il suffit de contrôler les valeurs propres de $(I - \gamma H)^{2t}$.

Les valeurs propres de $(I - \gamma H)^{2t}$ sont $(1 - \gamma\lambda)^{2t}$ pour $\lambda \in [\underline{\lambda}, \bar{\lambda}]$ une valeur propre de H .

Donc les valeurs propres de $(I - \gamma H)^{2t}$ ont une magnitude inférieure à

$$\max_{\lambda \in [\underline{\lambda}, \bar{\lambda}]} |1 - \gamma\lambda|$$

Choix optimal

on peut vérifier que minimiser $\max_{\lambda \in [\underline{\lambda}, \bar{\lambda}]} |1 - \gamma\lambda|$ se fait en mettant

$$\gamma = 2/(\underline{\lambda} + \bar{\lambda}),$$

avec une valeur optimale de $\frac{1-\kappa}{1+\kappa} \in (0, 1)$, telle que $\kappa = \frac{\bar{\lambda}}{\underline{\lambda}}$.

Convergence de $F(\theta_t) - F(\theta^*)$

- Afin d'obtenir un taux de convergence, nous devrons limiter les valeurs propres de $(I - \gamma H)^{2t}H$ au lieu de $(I - \gamma H)^{2t}$.
- La principale différence est que pour les valeurs propres λ de H qui sont proches de zéro $(1 - \gamma\lambda)^{2t}$ ne fait pas un fort effet de contraction, mais ils comptent moins car ils sont multipliés par λ dans la borne.
- Nous pouvons maintenant préciser ce compromis, pour $\gamma \leq 1/\bar{\lambda}$, comme

$$\begin{aligned} |\lambda(1 - \gamma\lambda)^{2t}| &\leq \lambda \exp(-\gamma\lambda)^{2t} = \lambda \exp(-2\gamma\lambda t) \\ &= \frac{1}{2t\gamma} 2t\gamma\lambda \exp(-\gamma\lambda) \leq \frac{1}{2t\gamma} \sup_{\alpha \geq 0} \alpha \exp(-\alpha) = \frac{1}{2te\gamma} \leq \frac{1}{4t\gamma} \end{aligned}$$

Ceci mène à

$$F(\theta_t) - F(\theta^*) \leq \frac{1}{4t\gamma} \|\theta_0 - \theta^*\|_2^2.$$

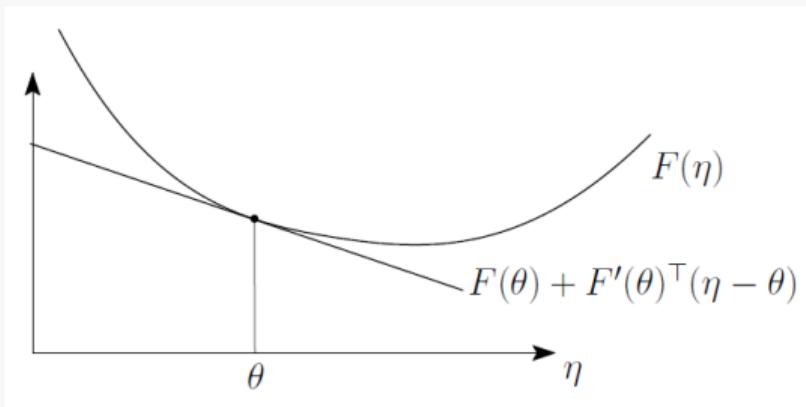
(fonction Convexe)

$$\forall \eta, \theta \in \mathbb{R}^d \text{ et } \alpha \in [0, 1], \quad F(\alpha\eta + (1 - \alpha)\theta) \leq \alpha F(\eta) + (1 - \alpha)F(\theta).$$

et si F est différentiable alors

$$F(\eta) \geq F(\theta) + F'(\theta)^T(\eta - \theta), \forall \eta, \theta \in \mathbb{R}^d$$

Ceci correspond à la fonction F étant au-dessus de sa tangente en θ ,

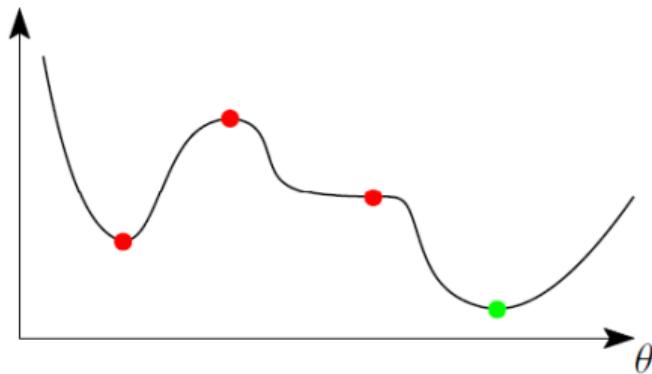


S

Supposons que $F : \mathbb{R}^d \rightarrow \mathbb{R}$ est convexe et différentiable. Alors $\theta^* \in \mathbb{R}^d$ est un minimiseur global de F si et seulement si

$$F'(\theta^*) = 0.$$

Ceci implique que pour la fonction convexe, nous devons seulement rechercher des points stationnaires. Ce n'est pas le cas pour les fonctions non convexes.



(Forte convexité)

Une fonction différentiable F est dite μ -fortement convexe, avec $\mu > 0$, si et seulement si

$$F(\eta) \geq F(\theta) + F'(\theta)^T(\eta - \theta) + \frac{\mu}{2}\|\eta - \theta\|_2^2, \quad \forall \eta, \theta \in \mathbb{R}^d.$$

Pour des fonctions deux fois différentiables, cela équivaut à $F'' < \mu I$

(Lissage)

Une fonction différentiable F est dite L -lisse si et seulement si

$$|F'(\eta) - F(\theta) - F'(\theta)^T(\eta - \theta)| \leq \frac{L}{2}\|\eta - \theta\|_2^2, \quad \forall \theta, \eta \in \mathbb{R}^d$$

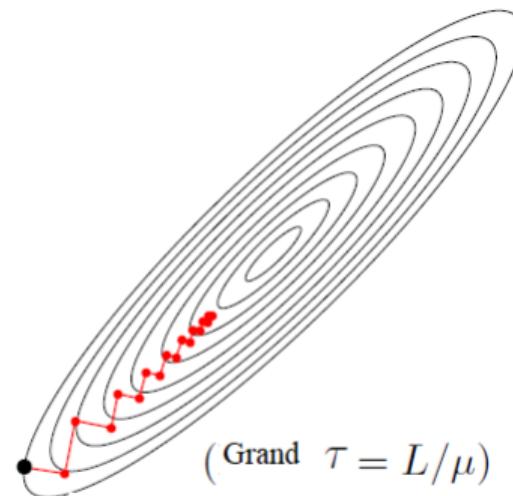
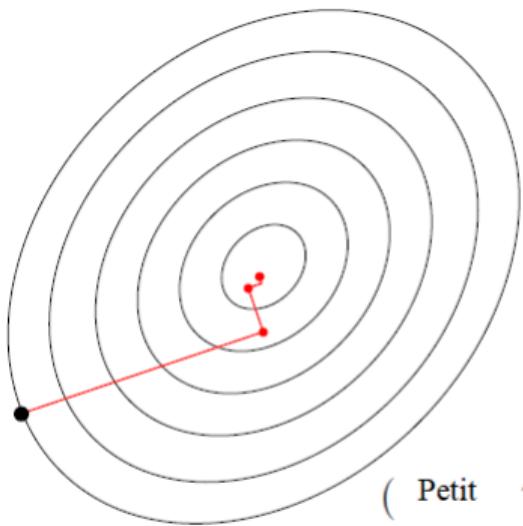
Ceci est équivalent à F ayant un gradient L -Lipschitz, c'est-à-dire

$$\|F'(\eta) - F'(\theta)\|_2^2 \leq L\|\eta - \theta\|_2^2, \quad \forall \theta, \eta \in \mathbb{R}^d.$$

Pour les fonctions deux fois différentiables, cela équivaut à

$$-LI \leq F''(\theta) \leq LI$$

Lorsqu'une fonction est à la fois lisse et fortement convexe, on note $\tau = L/\mu > 1$ son nombre de condition. La performance de la descente de gradient dépendra de ce nombre de condition (voir la descente la plus raide ci-dessous, c'est-à-dire la descente de gradient avec recherche de ligne exacte): avec un petit nombre de condition (à gauche), nous obtenons une convergence rapide, tandis que pour un grand nombre de condition (à droite), nous obtenons des oscillations.



Pour les problèmes d'apprentissage automatique, pour les prédictions linéaires et les pertes lisses (carrées ou logistiques), alors nous avons des problèmes lisses. Si nous utilisons un régulariseur carré ℓ_2 -régulariseur $\frac{\mu}{2}$, nous obtenons un problème μ -fortement convexe. Notez que lors de l'utilisation de la régularisation, la valeur de μ décroît avec la taille de échantillon n , typiquement entre $1/n$ et $1/\sqrt{n}$, conduisant à des nombres de condition entre \sqrt{n} et n . Dans ce contexte, la descente de gradient sur le risque empirique, est souvent appelée technique «batch».

Théorème: (Convergence de GD pour les fonctions fortement convexes)

Supposons que F est L -lisse et μ -fortement convexe. En choisissant $\gamma_t = 1/L$, les itérations $(\theta_t)_{t \geq 0}$ de GD sur F satisfont

$$F(\theta_t) - F(\theta^*) \leq \exp(-t\mu/L)(F(\theta_0) - F(\theta^*)).$$

- On a forcément $\mu \leq L$. Le rapport $\tau := L/\mu$ est appelé nombre de condition.
- Si nous supposons seulement que la fonction est lisse et convexe (pas fortement convexe), alors GD avec un pas constant $\gamma = 1/L$ converge également quand un minimiseur existe, mais à un rythme plus lent en $O(1/t)$.
- Le choix de la taille de pas ne nécessite qu'une borne supérieure L sur la constante de lissage (en cas de surestimation, le taux de convergence ne se dégrade que légèrement).
- Notez que la descente de gradient est adaptative à une forte convexité: le même algorithme s'applique aux cas fortement convexes et convexes, et les deux limites s'appliquent. Cette adaptivité est importante en pratique, car souvent, localement autour de l'optimum global, la constante de forte convexité converge vers la valeur propre minimale de la Hessien à θ^* , qui peut être très significativement supérieure à μ (la constante globale).

Quelques variantes de gradient descente

Pour l'algorithme de gradient descente classique à chaque itération, nécessite de calculer un gradient «complet» $F'(\theta_t)$ qui pourrait être coûteux. Une alternative consiste à ne calculer que des estimations stochastiques non biaisées du gradient $g_t(\theta_t)$, c'est-à-dire telles que $\mathbb{E}[g_t(\theta_t)|\theta_t] = F'(\theta_t)$, ce qui pourrait être beaucoup plus rapide à calculer. Cela conduit à l'algorithme suivant.

Algorithme (Descente de gradient stochastique (SDG))

Choisir la séquence de taille de pas $(\gamma_t)_{t \geq 0}$, choisir $\theta_0 \in \mathbb{R}^d$ et pour $t \geq 0$, soit

$$\theta_{t+1} = \theta_t - \gamma_t g_t(\theta_t).$$

SGD dans l'apprentissage automatique. Où l'itération t on peut choisir uniformément au hasard $i_t \in \{1, \dots, n\}$ et définissez $g_t(\theta) = F'[\ell(y_{i_t}, f_\theta(x_{i_t}))]$. Il existe un «mini-batch» variantes où à chaque itération, le gradient est moyenné sur un sous-ensemble aléatoire d'indices.

Accélération Nesterov:

Pour les fonctions convexes, une simple modification de la descente de gradient permet de obtenir de meilleurs taux de convergence. L'algorithme est le suivant et est basé sur la mise à jour des itérations:

$$\begin{aligned}\theta_t &= \eta_{t-1} - \frac{1}{L} F'(\eta_{t-1}) \\ \eta_t &= \theta_t + \frac{t-1}{t+2} (\theta_t - \theta_{t-1}).\end{aligned}$$

Cette simple modification remonte à Nesterov en 1983, et conduit au taux de convergence suivant

$$F(\theta_t) - F(\theta^*) \leq \frac{2L}{(t+1)^2} \|\theta_0 - \theta^*\|^2$$

Pour les fonctions fortement convexes, l'algorithme a une forme similaire à celle des fonctions convexes, mais avec tous les coefficients indépendants de t :

$$\begin{aligned}\theta_t &= \eta_{t-1} - \frac{1}{L} F'(\eta_{t-1}) \\ \eta_t &= \theta_t + \frac{1 - \sqrt{\mu/L}}{1 + \sqrt{\mu/L}} (\theta_t - \theta_{t-1}).\end{aligned}$$

et le taux de convergence est

$$F(\theta_t) - F(\theta^*) \leq L(1 - \sqrt{\mu/L})^t \|\theta_0 - \theta^*\|^2,$$

c'est-à-dire que le temps caractéristique de convergence va de τ à τ . Si τ est grand (typiquement d'ordre \sqrt{n} ou n pour l'apprentissage automatique), les gains sont substantiels. En pratique, cela conduit à des améliorations significatives.

Méthodes de Newton:

Étant donné θ_{t-1} , la méthode de Newton minimise l'expansion de Taylor du second ordre autour de θ_{t-1} ,

$$F(\theta_{t-1}) + F'(\theta_{t-1})^T(\theta - \theta_{t-1}) + \frac{1}{2}(\theta - \theta_{t-1})^T F''(\theta_{t-1})^T(\theta - \theta_{t-1}),$$

ce qui conduit à

$$\theta_t = \theta_{t-1} - F''(\theta_{t-1})^{-1} F'(\theta_{t-1}),$$

qui est une itération expansive, car la complexité en temps d'exécution est $O(d^3)$ en général pour résoudre le système linéaire . Cela conduit à une convergence quadratique locale: Si $\|\theta_{t-1} - \theta^*\|$ assez petit, pour une certaine constante C , on a $(C\|\theta_t - \theta^*\|) = (C\|\theta_{t-1} - \theta^*\|)^2$.

Descente de proximal gradient:

De nombreux problèmes d'optimisation sont dits «composites», c'est-à-dire que la fonction objectif F est la somme d'une fonction lisse G et d'une fonction non lisse H (telle qu'une norme). Il s'avère qu'une simple modification de la descente de gradient permet de bénéficier des taux de convergence rapides de l'optimisation en douceur.

Pour cela, nous devons d'abord voir la descente de gradient comme une méthode proximale. En effet, on peut voir l'itération $\theta_t = \theta_{t-1} - \frac{1}{L}G'(\theta_{t-1})$, comme

$$\theta_t = \arg \min_{\theta \in \mathbb{R}^d} G(\theta_{t-1}) + (\theta - \theta_{t-1})^T G'(\theta_{t-1}) + \frac{L}{2} \|\theta - \theta_{t-1}\|^2,$$

où, pour une fonction L -lisse G , la fonction objectif ci-dessus est une borne supérieure de $G(\theta)$ qui est serrée à θ_{t-1} .

$$\theta_t = \arg \min_{\theta \in \mathbb{R}^d} G(\theta_{t-1}) + (\theta - \theta_{t-1})^T G'(\theta_{t-1}) + \frac{L}{2} \|\theta - \theta_{t-1}\|^2 + H(\theta),$$

où H est laissé tel quel. Il s'avère que les taux de convergence pour $G + H$ sont les mêmes que l'optimisation douce, avec une accélération potentielle.

Autres algorithmes

- Stochastic gradient descent
- Coordinate gradient descent
- Accelerated gradient descent
- Averaged gradient descent
- Subgradient descent
- Proximal gradient descent
- Conjugate gradient descent
- Conditional gradient descent
- Quasi Newton methods
- Alternative direction method of multipliers
- Douglas-Rachford
- ...

Quel algorithme choisir lorsque n et p sont grands

L'erreur statistique est de l'ordre de $n \propto \frac{1}{\varepsilon^2}$

Principe : l'erreur statistique \approx l'erreur algorithmique

méthodes	Gradient stochastique	1er ordre Gradient	2nd ordre Q Newton	QP dual
pour une itération	p	np	$p(n + p)$	
nombre d'itération	$\frac{\kappa\gamma}{\varepsilon}$	$\kappa \log \frac{1}{\varepsilon}$	$\log \log \frac{1}{\varepsilon}$	
temps pour ε	$\frac{\kappa\gamma p}{\varepsilon}$	$\kappa np \log \frac{1}{\varepsilon}$	$pn \log \log \frac{1}{\varepsilon}$	n^2
temps pour ε	$\frac{\kappa\gamma p}{\varepsilon}$	$\kappa \frac{p}{\varepsilon^2} \log \frac{1}{\varepsilon}$	$\frac{p}{\varepsilon^2} \log \log \frac{1}{\varepsilon}$	$\frac{1}{\varepsilon^4}$

L'algorithme le plus rapide est
le gradient stochastique

Les différents types de modèles universels

- modèles basées sur les variables
 - la notion de dictionnaire (base, polynômes, ondelettes. . .)

$$\hat{f}(x) = \sum_{k=1}^n \alpha_k \phi_k(x)$$

- modèles basées sur les exemples
 - plus proches voisins
 - noyaux (SVM)
- combinaison d'éléments simples (partitions)
 - les forêts aléatoires
 - le boosting
- modèles profonds (deep learning)
 - linéaires vs non linéaires : → C'est un problème de représentation (extraction de caractéristiques)

La régression logistique

Définition (Modèle probabiliste)

Soit $\theta \subset \mathbb{R}^p$ un ensemble de paramètres. On appelle modèle P un ensemble de lois de probabilités à valeur dans Y , possédant une densité par rapport à la mesure de référence sur Y et indexés par $\Theta : P = \{p_\theta d\mu | \theta \in \Theta\}$

Exemple. Modèles Binomial, Multinomial, Gaussien univarié et multivarié.

Définition. (Vraisemblance)

Soit une donnée $y \in \mathcal{Y}$. On appelle vraisemblance la fonction $\theta \rightarrow p_\theta(x)$

On considère un ensemble d'entraînement i.i.d. y_1, \dots, y_n (dans ce contexte aussi échantillon). La vraisemblance de l'ensemble d'entraînement est

$$L(\theta) := \prod_{i=1}^n p_\theta(y_i)$$

Principe du maximum de vraisemblance

Principe : un bon choix de paramètre est un choix de paramètre qui maximise la probabilité des données observées, i.e. qui maximise la vraisemblance.

- principe du à Sir Ronald Fisher
- validé a posteriori par les bonnes propriétés du maximum de vraisemblance

Reformulation en terme de risque

On définit comme fonction de perte la log-vraisemblance $\ell(\theta, y) = -\log(p_\theta(y))$. Le risque associé est

$$R(\theta) = -E[\log(p_\theta(Y))]$$

En particulier si $Y \sim p_{\theta_0} d\mu$ pour $\theta_0 \in \Theta$ alors le paramètre cible est $\theta^* = \theta_0$. Le risque empirique est alors par définition

$$\hat{R}_n(\theta) = -\frac{1}{n} \sum_{i=1}^n \log(p_\theta(y_i))$$

La régression logistique (Cox, 1958)

Le modèle logistique : y_i réalisation d'une variable aléatoire $Y \sim \text{Bernoulli}$ de paramètre

$$p(x_i) = \frac{\exp(\varphi(x_i))^T w}{1 + \exp((\varphi(x_i))^T w)}$$

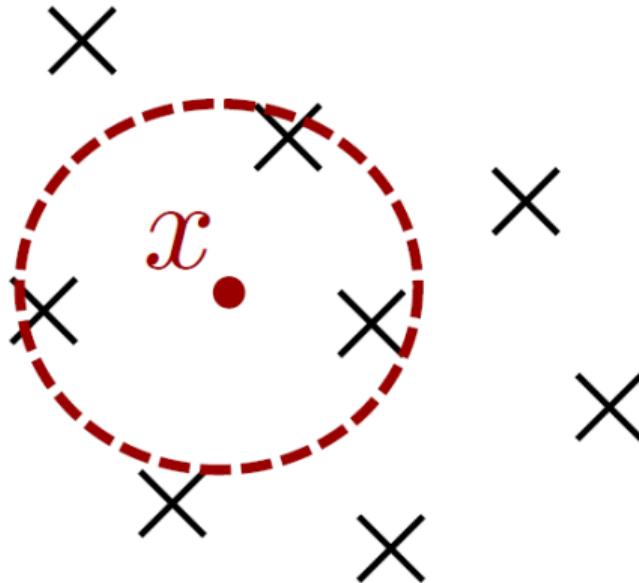
Le cout logistique : le max de vraisemblance

$$\hat{R}_n(w) = -\log \mathcal{L}(w) = \sum_{i=1}^n -y_i (\phi(x_i)^T w) + \log(1 + \exp(\varphi(x_i))^T w)$$

Cette fonction cout est non linéaire mais

- différentiable
- convexe
- consistante

Une approche par “moyennage local”



k plus proches voisins (k -nearest neighbours, k -nn):

Pour prédire l'étiquette de x , “moyenner” les étiquettes Y_i des points de la base d'apprentissage les plus proches de x

Définition formelle

Soit d une distance sur X . Pour chaque x , on peut ordonner les points de la base d'apprentissage par leur distance à x :

$$d(x, X_{\pi_1^{(n)}(x)}) \leq d(x, X_{\pi_2^{(n)}(x)}) \leq \dots d(x, X_{\pi_n^{(n)}(x)}).$$

$\pi_i^{(n)}(x)$: indice du i-ème point de E_n le plus proche de x

Prédicteur des k -plus proches voisins

Pour $k \in \{1, \dots, n\}$, le prédicteur k -ppv est tel que

$$\hat{f}_n^{k-ppv}(x) = \text{"moyenne" des } \left\{ Y_{\pi_i^{(n)}(x)}, 1 \leq i \leq n \right\}$$

Définition formelle

Régression :

$$\hat{f}_n^{k-ppv}(x) = \frac{1}{k} \sum_{i=1}^k Y_{\pi_i^{(n)}(x)}$$

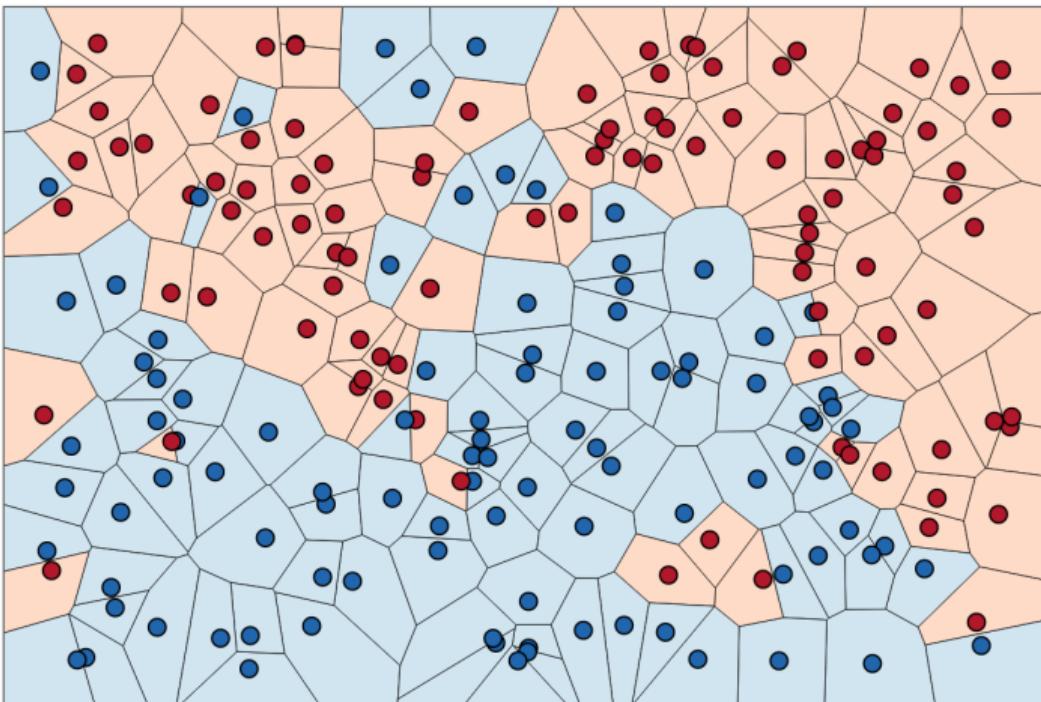
(moyenne arithmétique)

Classification :

$$\hat{f}_n^{k-ppv}(x) = \arg \max_{y \in \mathcal{Y}} \sum_{i=1}^k \mathbb{1}_{Y_{\pi_i^{(n)}(x)}}$$

(classe majoritaire)

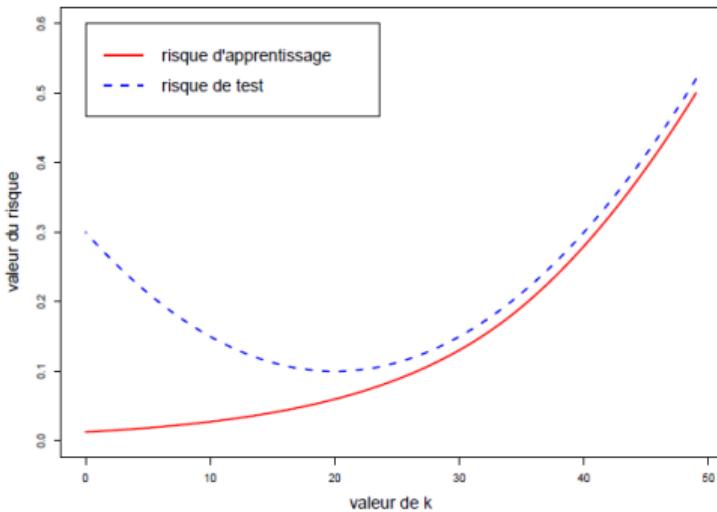
Visualisation (classification binaire)



Classifieur du plus-proche voisin

Comment choisir k ?

Evaluons le risque d'apprentissage et le risque de test pour différentes valeurs de k .



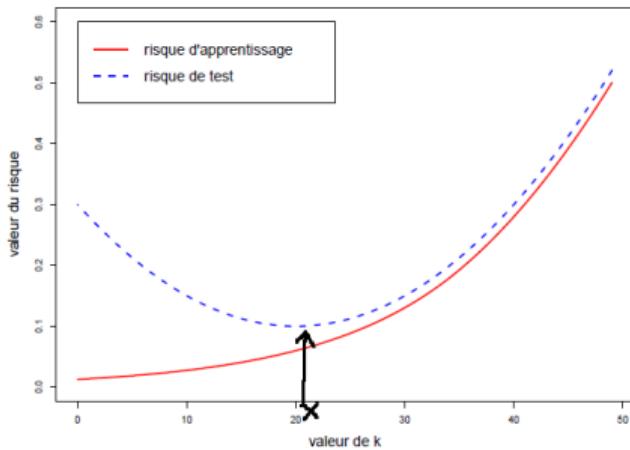
- petites valeurs de k : **forte variance** (sur-apprentissage)
- grandes valeurs de k : **fort biais** (le classifieur appris est trop simple)

Comment choisir k ?

Peut-on sélectionner la valeur \hat{k} ci-dessous ?

$$\hat{k} = \hat{k}(E_n, E_m^{(t)})$$

ajusté pour minimiser le risque sur cette base de test particulière...



(⚠️ sur-apprentissage)

Comment choisir k ?

La méthode la plus courante pour choisir K est la validation croisée.

