Modèles supervisés pour Data-Science

Massih-Reza Amini

Université Grenoble Alpes
Laboratoire d'Informatique de Grenoble
Massih-Reza.Amini@imag.fr

Programme

1. Principe de la minimisation du risque empirique
2. Optimisation convexe non-contrainte
3. Algorithme de descente de Gradient
4. ADAptive LInear NEuron (ADALINE)

Rappel

- Comme la distribution de probabilité $\mathcal{D}$ est inconnue, la forme analytique du vrai risque $R(f)$ ne peut pas être estimée, et donc la fonction de prédiction ne peut pas être calculée directement.

- Principe de la Minimisation du Risque Empirique: Trouver $h : \mathcal{X} \to \mathbb{R}$, à partir d'une classe de fonctions $\mathcal{H}$ en minimisant l'estimateur non-biaisé de R sur une base d'apprentissage $S = (x_i, y_i)_{i=1}^m$:

$$\hat{R}_m(f, S) = \frac{1}{m} \sum_{i=1}^{m} [\![y_i \times h(x_i) \leq 0]\!]$$

Définir alors la fonction de classification f comme $\forall x, f(x) = \text{sgn}(h(x))$

Or la minimisation de l'erreur de classification empirique est impossible ...

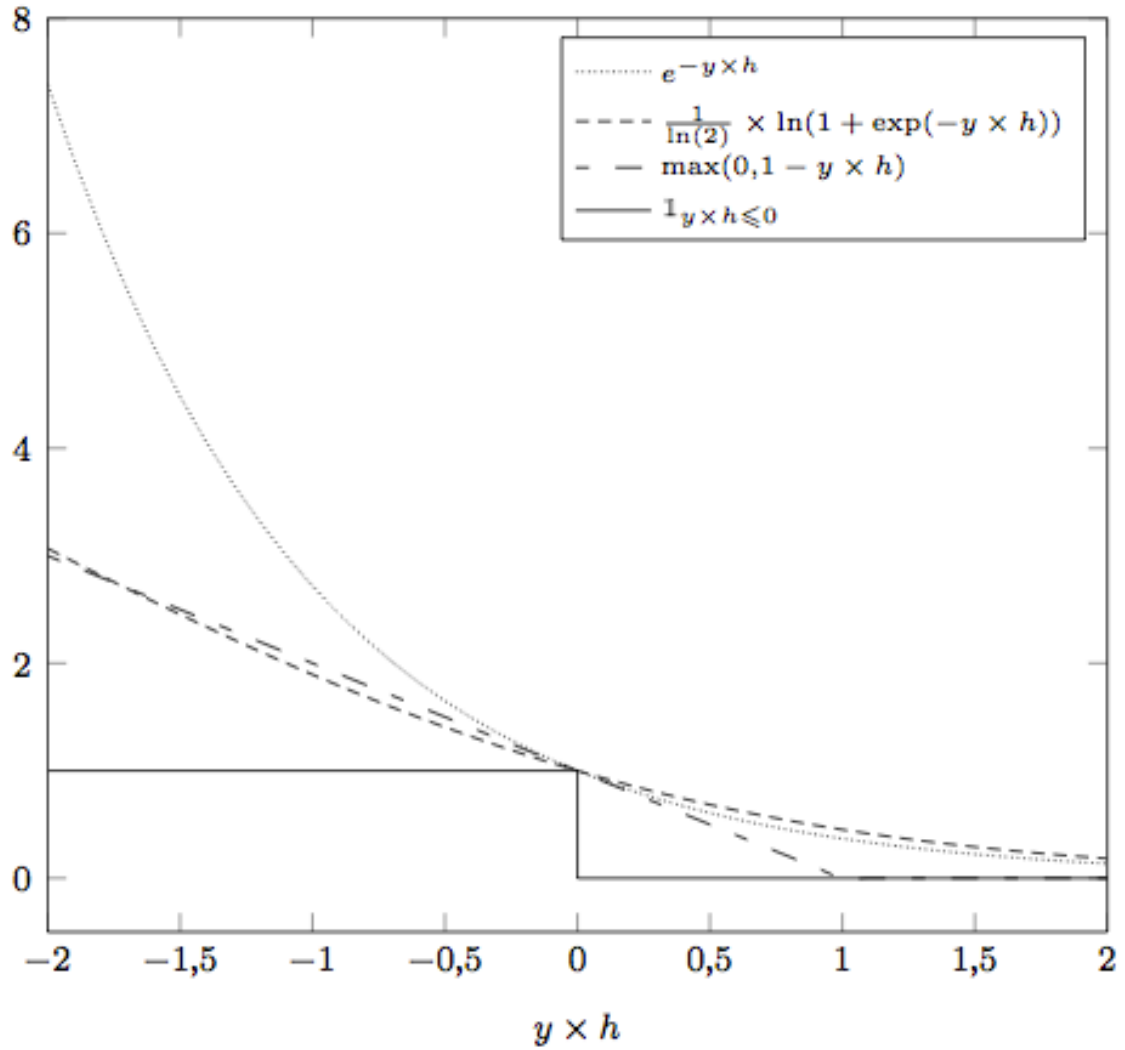
- La fonction empirique

$$R_m : \boldsymbol{w} \mapsto \frac{1}{m} \sum_{i=1}^{m} [\![y_i \times h_{\boldsymbol{w}}(x_i) \leq 0]\!]$$

  n'est ni continue, ni dérivable.

- La minimisation d'une borne supérieure convexe $\hat{\mathcal{L}}(\boldsymbol{w})$ de R_m prenant la valeur 1 en 0 conduira à la même minimiseur que R_m [Bartlett et al.]

Bornes supérieures convexes classiques de l'erreur de classification



Optimisation convexe non-contrainte

- Le problème d'apprentissage se définit comme un problème d'optimisation non-contraintes plus facile.
- Considérer l'expansion de Taylor de la fonction objectif autour de son minimiseur

$$\hat{\mathcal{L}}(\boldsymbol{w}) = \hat{\mathcal{L}}(\boldsymbol{w}^*) + (w - w^*)^\top \underbrace{\nabla \hat{\mathcal{L}}(\boldsymbol{w}^*)}_{=0} + \frac{1}{2}(\boldsymbol{w} - \boldsymbol{w}^*)^\top \mathbf{H}(\boldsymbol{w} - \boldsymbol{w}^*) + o(\| w - w^* \|^2)$$

- La matrice Hessienne est symétrique et d'après le théorème de Schwarz ses vecteurs propres $(\boldsymbol{v}_i)_{i=1}^d$ forment une base orthonormée.

$$\forall (i,j) \in \{1,\ldots,d\}^2, \mathbf{H}\boldsymbol{v}_i = \lambda_i \boldsymbol{v}_i, \text{ et } \boldsymbol{v}_i^\top \boldsymbol{v}_j = \begin{cases} +1 : \text{si } i = j, \\ 0 : \text{otherwise.} \end{cases}$$

Optimisation convexe non-contrainte (2)

- Chaque vecteur de poids $\boldsymbol{w} - \boldsymbol{w}^*$ peut être décomposée de façon unique sur cette base

$$\boldsymbol{w} - \boldsymbol{w}^* = \sum_{i=1}^{d} q_i v_i$$

- C'est à dire

$$\hat{\mathcal{L}}(\boldsymbol{w}) = \hat{\mathcal{L}}(\boldsymbol{w}^*) + \frac{1}{2} \sum_{i=1}^{d} \lambda_i q_i^2$$

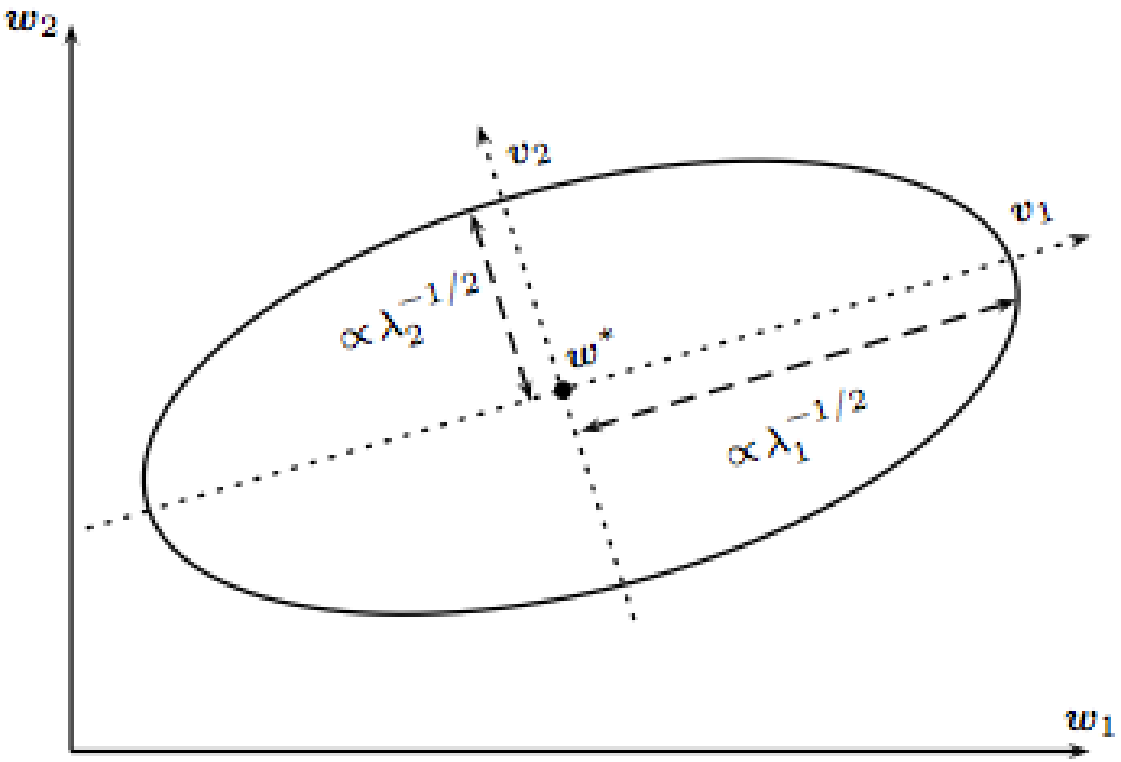
- De plus, comme la fonction objectif admet un minimum global, la matrice Hessienne est définie positive

$$(\boldsymbol{w} - \boldsymbol{w}^*)^\top \mathbf{H} (\boldsymbol{w} - \boldsymbol{w}^*) = \sum_{i=1}^{d} \lambda_i q_i^2 = 2(\hat{\mathcal{L}}(\boldsymbol{w}) - \hat{\mathcal{L}}(\boldsymbol{w}^*)) \geq 0$$

Toutes les valeurs propres de $\mathbf{H}$ sont positives.

Optimisation convexe non-contrainte (3)

- Ceci implique que les lignes de niveau de $\hat{\mathcal{L}}$, définies par des lignes à valeur constante, sont des ellipses

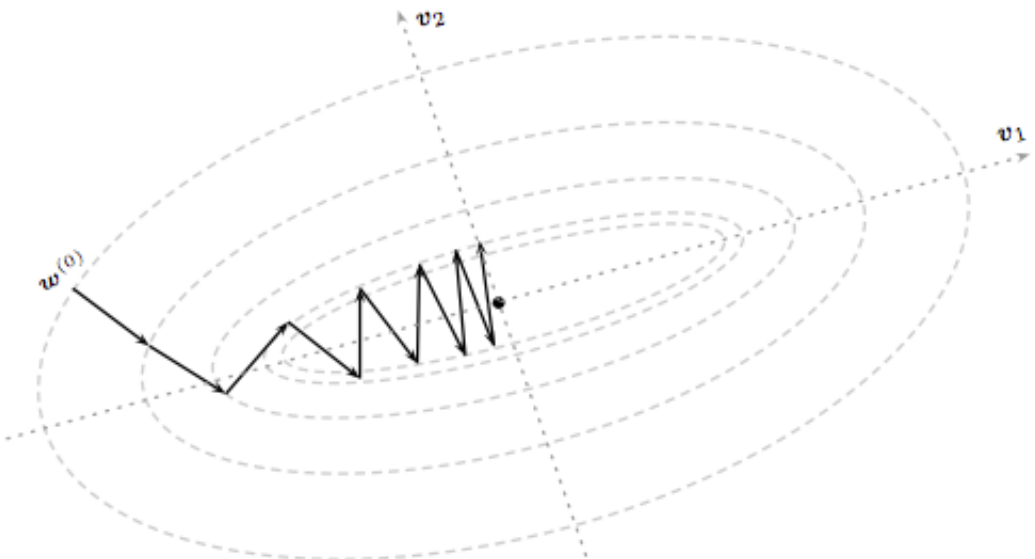


Algorithme de descente de Gradient [Rumelhart et al., 1986]

- L'algorithme de descente de gradient est un procédé itératif de la mise à jour des poids :

$$\forall t \in \mathbb{N}, \boldsymbol{w}^{(t+1)} = \boldsymbol{w}^{(t)} - \eta \nabla \hat{\mathcal{L}}(\boldsymbol{w}^{(t)})$$

Où $\eta > 0$ est le pas d'apprentissage



Convergence

- Prendre la décomposition dy vecteur $\boldsymbol{w} - \boldsymbol{w}^*$ dans la base orthonormée $(\boldsymbol{v}_i)_{i=1}^d$ formée par les $v.p.$ de la matrice Hessienne

$$\nabla \hat{\mathcal{L}}(\boldsymbol{w}) = \sum_{i=1}^{d} q_i \lambda_i \boldsymbol{v}_i$$

- Soit $\boldsymbol{w}^{(t)}$ le vecteur poids obtenu à partir de $\boldsymbol{w}^{(t-1)}$ en appliquant la mise à jour de l'algorithme de descente de Gradient

$$\boldsymbol{w}^{(t)} - \boldsymbol{w}^{(t-1)} = \sum_{i=1}^{d} \left(q_i^{(t)} - q_i^{(t-1)} \right) \boldsymbol{v}_i = -\eta \nabla \hat{\mathcal{L}}(\boldsymbol{w}^{(t-1)}) = -\eta \sum_{i=1}^{d} q_i^{(t-1)} \lambda_i \boldsymbol{v}_i$$

- D'où

$$\forall i \in \{1, \ldots, d\}, q_i^{(t)} = (1 - \eta \lambda_i)^t q_i^{(0)}$$

Et l'algorithme converge ssi

$$\eta < \frac{1}{2\lambda_{max}}$$

ADAptive LInear NEuron [Widrow & Hoff, 1960]

- ADAptive LInear NEuron
- Fonction de prédiction linéaire:

$$\begin{aligned} h_{\boldsymbol{w}} : \mathcal{X} &\rightarrow \mathbb{R} \\ x &\mapsto \langle \bar{w}, x \rangle + w_0 \end{aligned}$$

- Trouve les paramètres qui minimise la borne supérieure convexe de l'erreur 0/1

$$\hat{\mathcal{L}}(\boldsymbol{w}) = \frac{1}{m} \sum_{i=1}^{m} (y_i - h_{\boldsymbol{w}}(\mathbf{x}_i))^2$$

- Règle de mise à jour : L'algorithme de la descente de gradient stochastique avec un pas d'apprentissage $\eta > 0$

$$\forall (\mathbf{x}, y), \begin{pmatrix} w_0 \\ \bar{\boldsymbol{w}} \end{pmatrix} \leftarrow \begin{pmatrix} w_0 \\ \bar{\boldsymbol{w}} \end{pmatrix} + \eta (y - h_{\boldsymbol{w}}(\mathbf{x})) \begin{pmatrix} 1 \\ \mathbf{x} \end{pmatrix} \tag{1}$$

Adaline

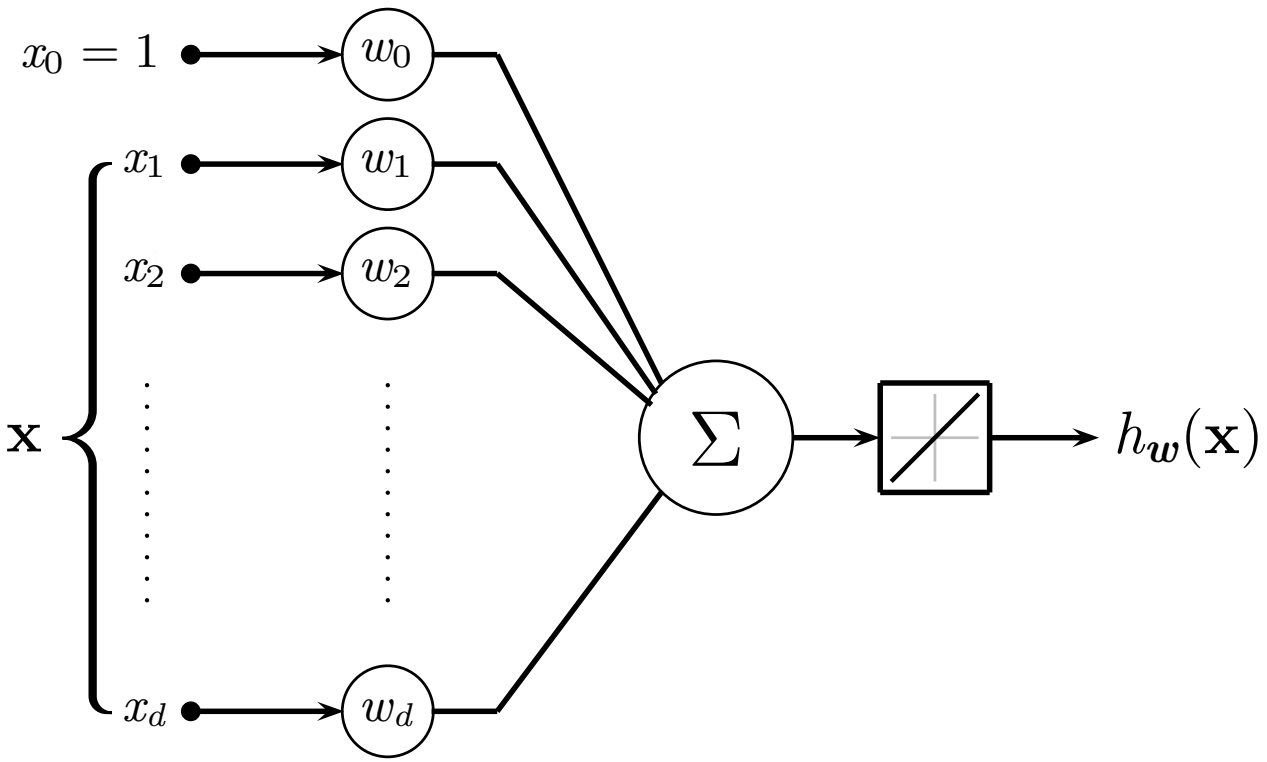
- ADAptive LInear NEuron
- Fonction de prédiction linéaire:

$$\begin{aligned} h_{\boldsymbol{w}} : \mathcal{X} &\rightarrow \mathbb{R} \\ x &\mapsto \langle \bar{w}, x \rangle + w_0 \end{aligned}$$

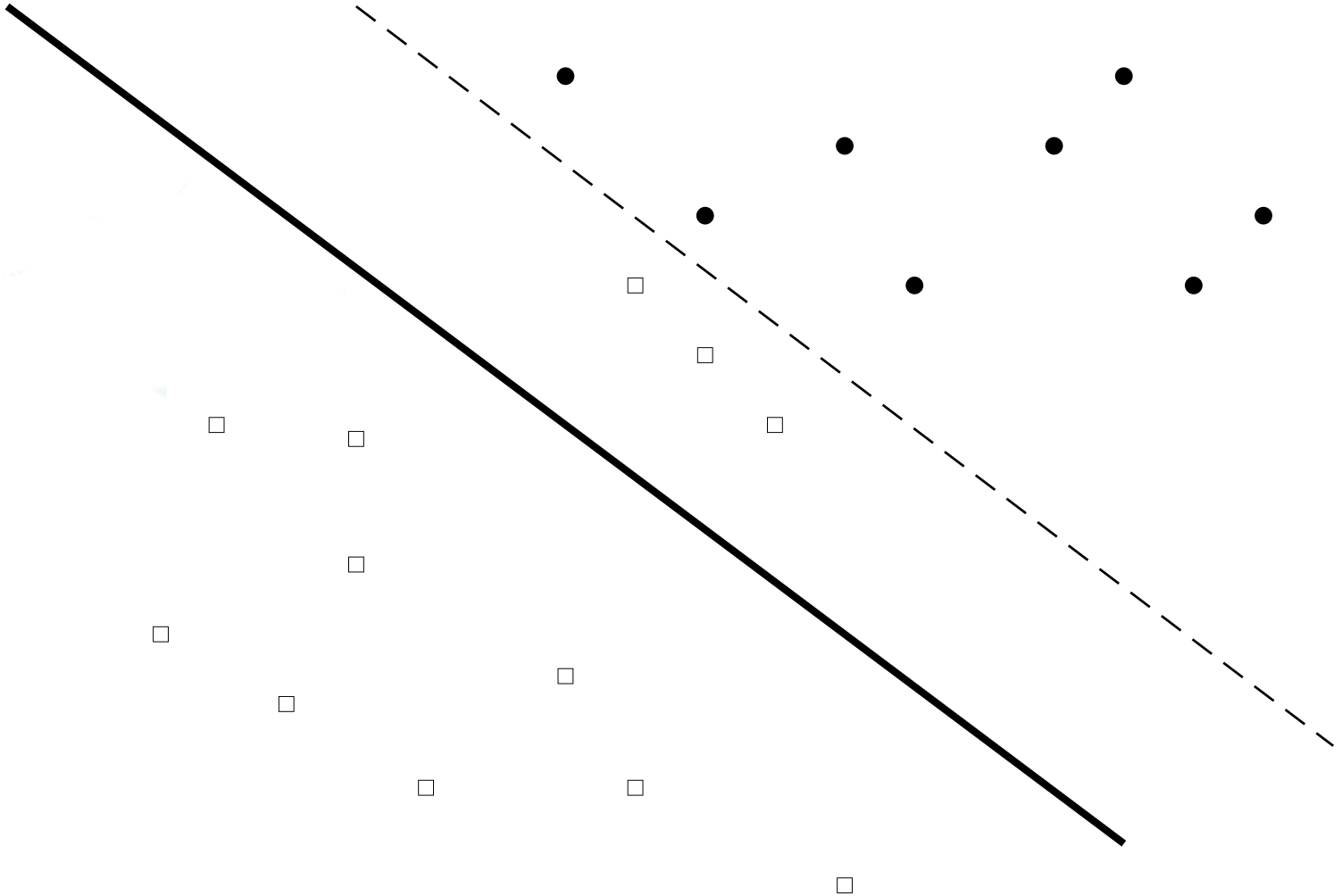
Algorithm 1 L'algorithme de Adaline

Base d'apprentissage $S = \{(x_i, y_i) \mid i \in \{1, \ldots, m\}\}$
Initialiser les poids $w^{(0)} \leftarrow 0$
$t \leftarrow 0$
Pas d'apprentissage $\eta > 0$
repeat
    Choisir un exemple aléatoire $(x^{(t)}, y^{(t)}) \in S$
    $w_0^{(t+1)} \leftarrow w_0^{(t)} + \eta \times (y^{(t)} - h_w(x^{(t)}))$
    $\bar{w}^{(t+1)} \leftarrow \bar{w}^{(t)} + \eta \times (y^{(t)} - h_w(x^{(t)})) \times x^{(t)}$
    $t \leftarrow t + 1$
until $t > T$

Formal models



Perceptron vs Adaline



Régularisation

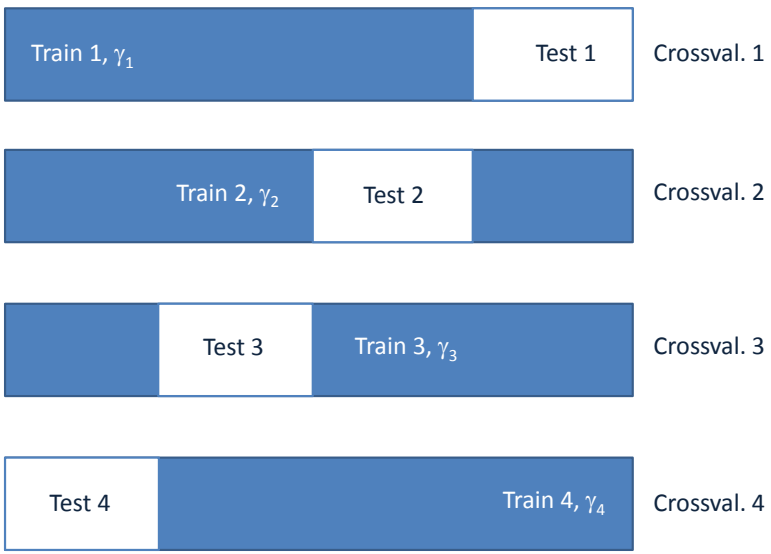
- Trouver la fonction de prédiction en minimisant le risque empirique en ajoutant une pénalité pour la taille du modèle,
- Une approche simple consiste à choisir une grande classe de fonctions $\mathcal{F}$ et de définir un *régulariseur*, typiquement la norme $|| g ||$, et de minimiser le risque empirique réguliariseur

$$\hat{f} = \underset{f \in \mathcal{F}}{\operatorname{argmin}} \hat{R}_m(f, S) + \underbrace{\gamma}_{hyperparameter} \times || f ||^2$$

- L'hyperparamètre, ou le *paramètre de régularisation* permet de choisir le bon compromis entre la complexité et l'ajustement.

La validation croisée

- Créer une partition de K sous-ensembles de la base d'apprentissage
  - Pour chacune des K expériences, utiliser $K-1$ sous-ensembles pour apprendre et le dernier sous-ensemble pour la validation. Pour $K=4$:



- La valeur de l'hyper-paramètre correspond à la valeur de γ_k pour laquelle la performance de test est la meilleure sur un de ces sous-ensembles.

References

P. L. Bartlett, M. I. Jordan and J.D. McAuliffe
Convexity, classification, and risk bounds.
Journal of the American Statistical Association, 101(473): 138–156, 2006.

F. Rosenblatt
The perceptron: A probabilistic model for information storage and organization in the brain.
Psychological Review, 65: 386–408.
1958

D. E. Rumelhart, G. E. Hinton and R. Williams
Learning internal representations by error propagation.
Parallel Distributed Processing: Explorations in the Microstructure of Cognition,
1986

G. Widrow and M. Hoff
Adaptive switching circuits.
Institute of Radio Engineers, Western Electronic Show and Convention, Convention Record, 4: 96–104.
1960.