Modèles supervisés pour Data-Science

Massih-Reza Amini

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

Programme

1. Régression Logistique (TP)
2. ADAptive BOOSTing (ADABOOST)

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 ...

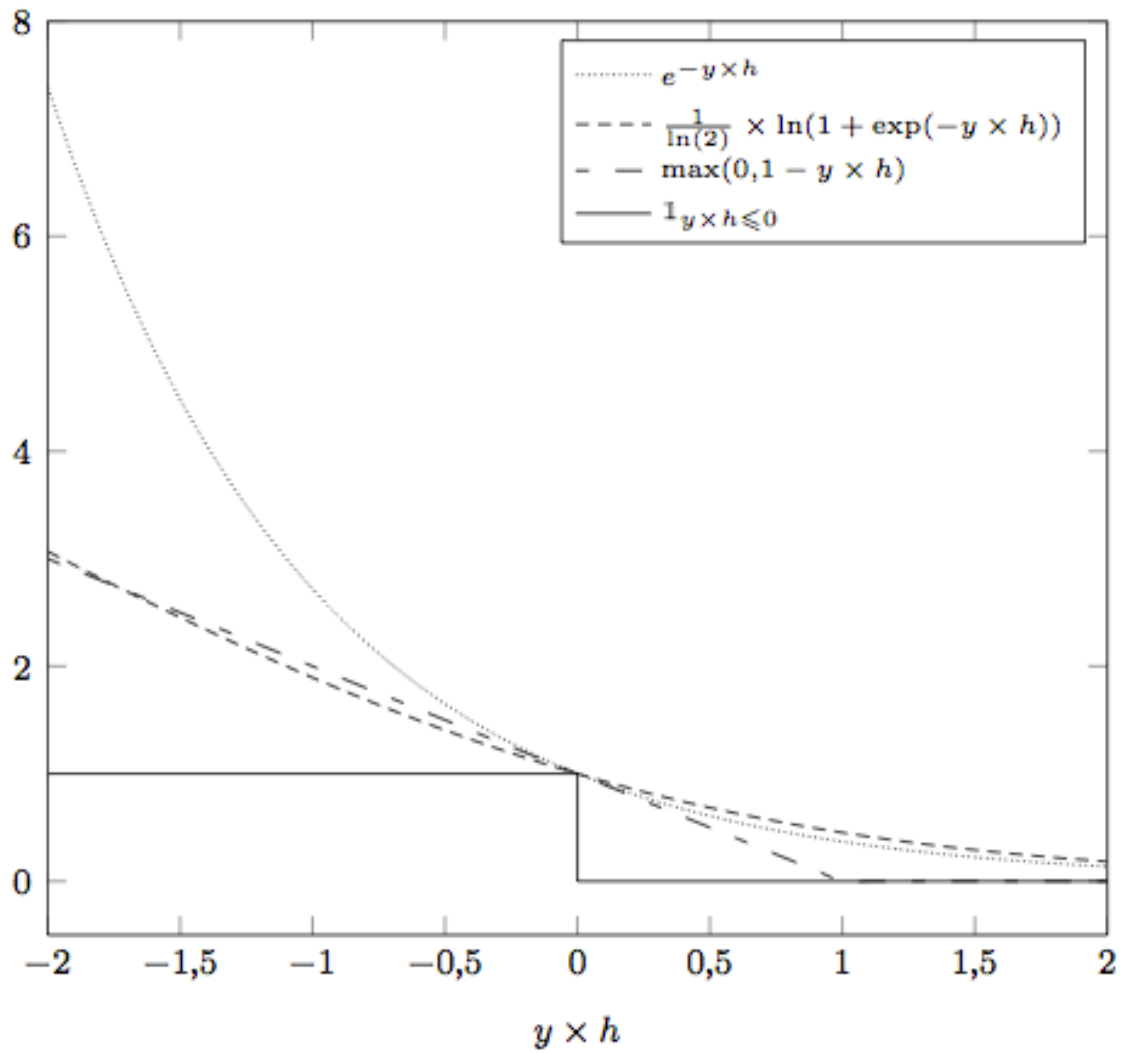
- La fonction empirique

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

  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 [**?**]

Bornes supérieures convexes classiques de l'erreur de classification



Régression Logistique

- Le modèle Logistique est défini en choisissant la classe des fonctions linéaires

$$\mathcal{H} = \{h_{\bar{\mathbf{w}}} : \mathbf{x} \mapsto w_0 + \langle \mathbf{w}, \mathbf{x} \rangle\}$$

et la fonction de coût logistique

$$\ell(\bar{\mathbf{w}}, \mathbf{x}, y) = \ln(1 + e^{-y h_{\bar{\mathbf{w}}}(\mathbf{x})})$$

- Le gradient de l'erreur empirique logistique, $\mathcal{L}(S, \mathbf{w})$, estimée sur une base d'apprentissage $S = \{(\mathbf{x}_i, y_i); i \in \{1, \ldots, m\}\}$:

$$\nabla_{\mathbf{w}} \mathcal{L}(S, \bar{\mathbf{w}}) = -\frac{1}{m} \sum_{i=1}^{m} y_i \left(1 - \frac{1}{1 + e^{-y_i h_{\bar{\mathbf{w}}}(\mathbf{x}_i)}}\right) \times \mathbf{x}_i$$

ADAptive BOOSTing [Schapire, 1999]

- L'algorithme Adaboost génère un ensemble d'apprenant faibles et les combine avec le vote majoritaire et construit un classifieur très efficace.

- Chaque apprenant faible est entraîné de façon à prendre en compte les erreurs de classification de l'apprenant précédent.
  - ☞ Ceci est fait en assignant des poids aux exemples de la base d'entraînement et en augmentant les poids des exemples pour lesquels l'apprenant se trompe sur la prédiction de leurs classes.
  - ☞ Dans ce cas, le nouveau apprenant se focalise sur les exemples *difficiles* à classer par l'apprenant précédent.

AdaBoost, algorithme

Algorithm 1 L'algorithme de Boosting

1: Base d'entraînement $S = \{(x_i, y_i) \mid i \in \{1, \ldots, m\}\}$
2: Initialisation des poids $\forall i \in \{1, \ldots, m\}, D_1(i) = \frac{1}{m}$
3: T, le nombre maximal d'itérations (ou de classifieurs à combiner)
4: **for** t=1,...,T **do**
5: Apprendre un classifieur faible $f_t : \mathcal{X} \to \{-1, +1\}$ en utilisant D_t
6: Poser $\epsilon_t = \sum_{i:f_t(x_i) \neq y_i} D_t(i)$
7: Choisir $\alpha_t = \frac{1}{2} \ln \frac{1-\epsilon_t}{\epsilon_t}$
8: Mettre à jour la distribution des poids

$$\forall i \in \{1, \ldots, m\}, D_{t+1}(i) = \frac{D_t(i)\, e^{-\alpha_t y_i f_t(x_i)}}{Z_t}$$

Où,

$$Z_t = \sum_{i=1}^{m} D_t(i)\, e^{-\alpha_t y_i f_t(x_i)}$$

9: **end for**
10: Le classifieur final: $\forall x, F(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t f_t(x)\right)$

AdaBoost, algorithme

Algorithm 2 L'algorithme de Boosting

1: Base d'entraînement $S = \{(x_i, y_i) \mid i \in \{1, \ldots, m\}\}$
2: Initialisation des poids $\forall i \in \{1, \ldots, m\}, D_1(i) = \frac{1}{m}$
3: T, le nombre maximal d'itérations (ou de classifieurs à combiner)
4: **for** t=1,...,T **do**
5: Apprendre un classifieur faible $f_t : \mathcal{X} \to \{-1, +1\}$ en utilisant D_t
6: Poser $\epsilon_t = \sum_{i:f_t(x_i) \neq y_i} D_t(i)$
7: Choisir $\alpha_t = \frac{1}{2} \ln \frac{1-\epsilon_t}{\epsilon_t}$
8: Mettre à jour la distribution des poids

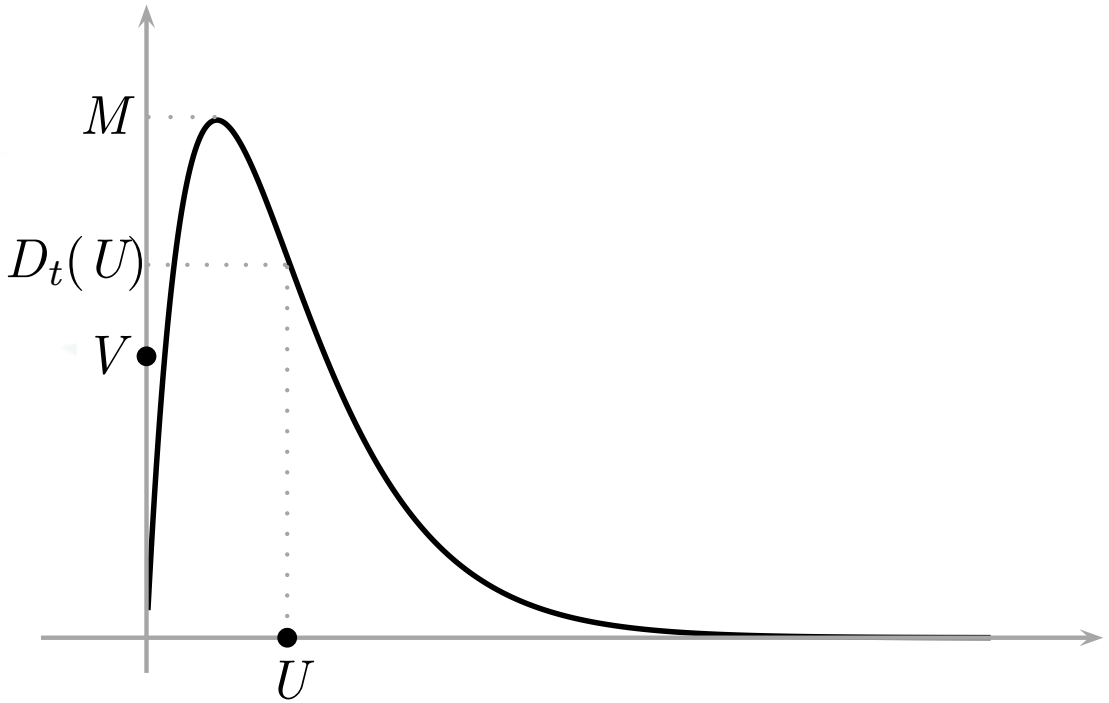
$$\forall i \in \{1, \ldots, m\}, D_{t+1}(i) = \frac{D_t(i) e^{-\alpha_t y_i f_t(x_i)}}{Z_t}$$

Où,

$$Z_t = \sum_{i=1}^{m} D_t(i) e^{-\alpha_t y_i f_t(x_i)}$$

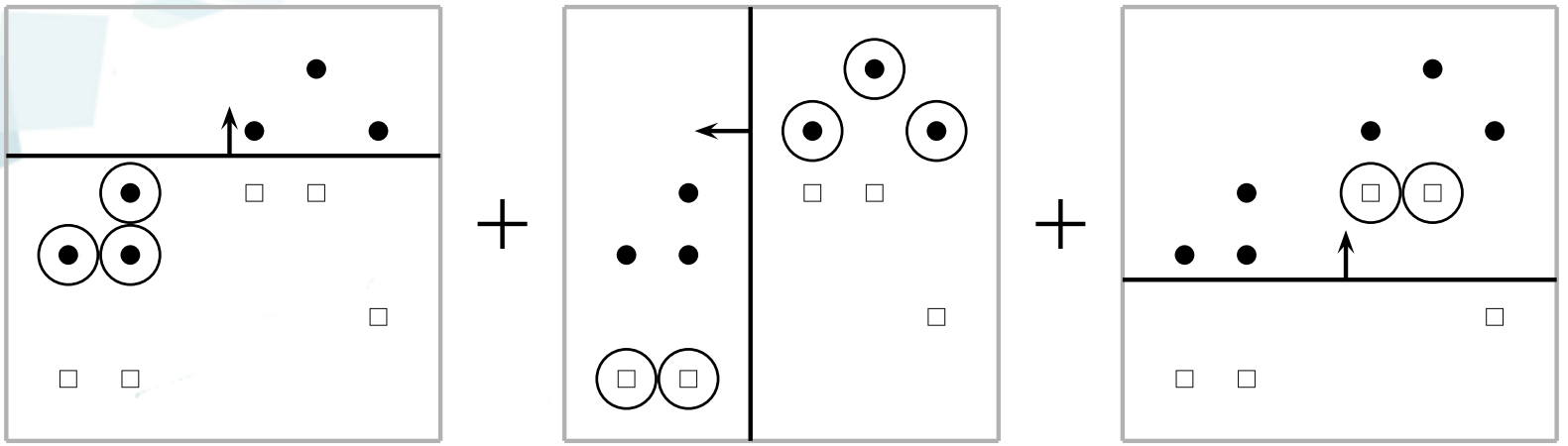
9: **end for**
10: Le classifieur final: $\forall x, F(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t f_t(x)\right)$

Comment échantillonner les exemples suivant la distribution D_t

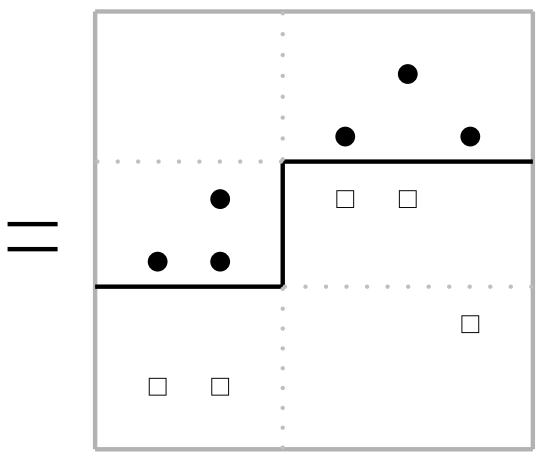


- Choisir aléatoirement un indice $U \in \{1, \ldots, m\}$ et une valeur réelle $V \in [0, \max_{i \in \{1,\ldots,m\}} D_t(i)]$, si $D_t(U) > V$ alors accepter l'exemple $(\mathbf{x}_U, y_U)$.

AdaBoost, interprétation géométrique



$\alpha_1 = 0.5$ $\alpha_2 = 0.1$ $\alpha_3 = 0.75$



AdaBoost : lien avec le principe MRE

- Si on pose $\forall x, H(x) = \sum_{t=1}^{T} \alpha_t f_t(x)$ et $F(x) = \text{sign}(H(x))$ alors

$$\frac{1}{m}\sum_{i=1}^{m}[\![y_i \neq F(x_i)]\!] \leq \frac{1}{m}\sum_{i=1}^{m} e^{-y_i H(x_i)}$$

- D'autre part

$$\frac{1}{m}\sum_{i=1}^{m} e^{-y_i H(x_i)} = \sum_{i=1}^{m} Z_1 D_2(i) \prod_{t>1} e^{-y_i \alpha_t f_t(x_i)}$$

alors

$$\frac{1}{m}\sum_{i=1}^{m} e^{-y_i H(x_i)} = \prod_{t=1}^{T} Z_t$$

AdaBoost : lien avec le principe MRE

- Le minimum du terme de normalisation, par rapport aux poids de combinaison, α_t

$$\forall t, Z_t = \epsilon_t e^{\alpha_t} + (1 - \epsilon_t) e^{-\alpha_t}$$

est atteint pour $\alpha_t = \frac{1}{2} \ln \frac{1-\epsilon_t}{\epsilon_t}$

- En posant $\gamma_t = \frac{1}{2} - \epsilon_t$, et quand $\epsilon_t < \frac{1}{2}$

$$\forall t, Z_t = \sqrt{1 - 4\gamma_t^2} \leq e^{-2\gamma_t^2}$$

- L'erreur de classification décroît de façon exponentielle vers 0

$$\frac{1}{m} \sum_{i=1}^{m} [\![y_i \neq F(x_i)]\!] \leq \prod_{t=1}^{T} Z_t \leq e^{-2 \sum_{t=1}^{T} \gamma_t^2}$$

References

L. Bottou

Une Approche théorique de l'Apprentissage Connexionniste: Applications à la Reconnaissance de la Parole.
Thèse de doctorat, Université de Paris XI, Orsay, France.
1999

R.E. Schapire

Theoretical views of boosting and applications.
In Proceedings of the 10th International Conference on Algorithmic Learning Theory, pages 13–25.
1999

D.E. Rumelhart, G.E. Hinton, R. Williams

Learning internal representations by error propagation.
Parallel Distributed Processing: Explorations in the Microstructure of Cognition,
1986

V.N. Vapnik

The nature of statistical learning theory (second edition).
Springer-Verlag.

1999