

# UN ALGORITHME D'APPRENTISSAGE PARCIMONIEUX MAXIMISANT LE DÉSACCORD

---

Jean-Francis Roy

13 février 2015

Université Laval

## INTRODUCTION

---

Nous nous attaquons au problème de **classification binaire** en apprentissage automatique. Après une introduction au problème à résoudre, nous déduirons de **bornes** un **algorithme d'apprentissage**. En utilisant des techniques d'**optimisation**, nous rendrons cet algorithme **parcimonieux**.

Introduction

Motivation

Définitions de base et notation

Les méthodes d'ensemble

La C-borne

MinCq : Minimiser la C-borne

CqBoost : un algorithme de type *Boosting*

Conclusion

## MOTIVATION

---

- Prédire si un courriel est un « spam » ou non (en fonction de son contenu)
- Prédire si un client s'intéressera à un produit (en fonction de ses achats antérieurs)
- Prédire si un utilisateur cliquera sur une publicité (en fonction de son comportement)
- Prédire si une protéine et une molécule se lieront ensemble (en fonction de leur description)
- Prédire si une application Android est un « malware » (en fonction de son comportement)
- En gros : **apprendre** une fonction qui catégorise un **nouvel** élément observé

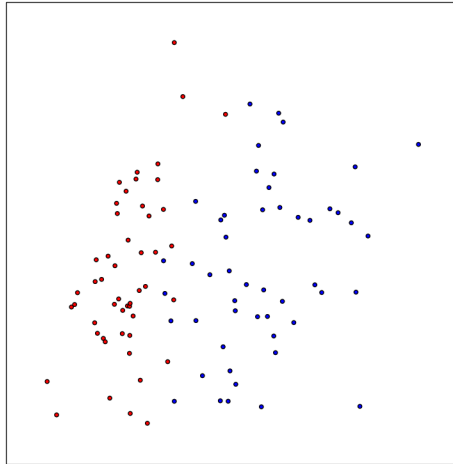
Cette fonction peut être plus ou moins **complexe**.

Lorsque le problème le permet, une solution **simple** (et **interprétable**) est préférable dans plusieurs applications.

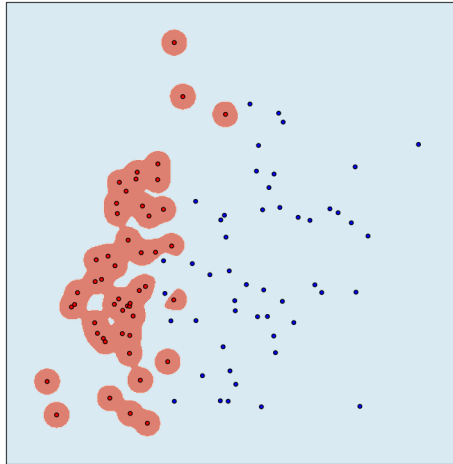
Est-ce suffisant de construire une fonction de prédiction performante sur les données que nous avons ?



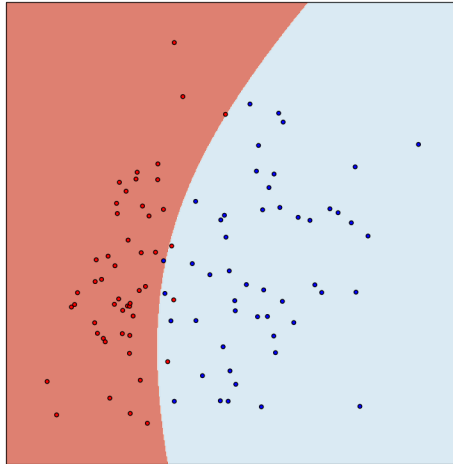
# LA TÂCHE À RÉSOUDRE



# LA TÂCHE À RÉSOUDRE



Ici, on ne fait aucune erreur.



Ici, on se permet quelques erreurs, mais on devrait mieux généraliser aux futurs exemples observés.

Nous supposons que les données auxquelles nous avons accès proviennent toutes de la même **distribution**.

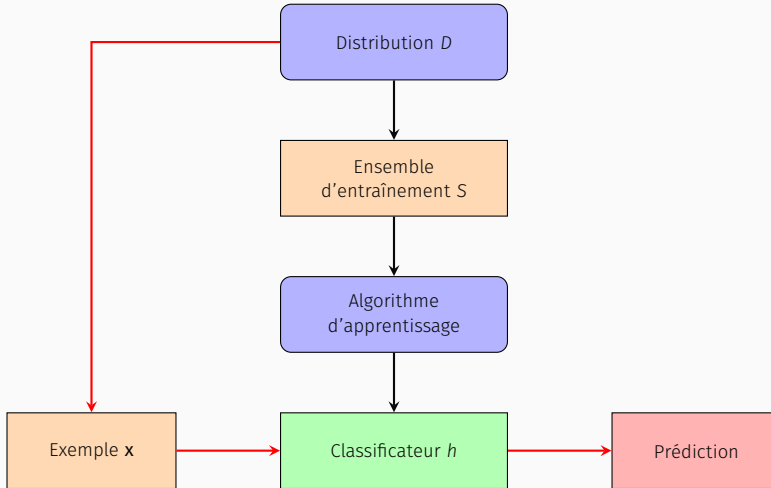
Nous recherchons une fonction de classification performante sur cette distribution !

## DÉFINITIONS DE BASE ET NOTATION

---

Nous considérons les problèmes de *classification binaire*, où

- Un **exemple** est une paire  $(\mathbf{x}, y)$ , où
  - $\mathbf{x} \in \mathcal{X}$ , où  $\mathcal{X}$  est l'espace d'entrée représentant sa description. Généralement,  $\mathcal{X} = \mathbb{R}^d$
  - $y \in \mathcal{Y}$ , où  $\mathcal{Y} = \{-1, +1\}$  est l'étiquette de l'exemple
- Chaque exemple est indépendamment et identiquement distribué (*i.i.d.*) d'une **distribution inconnue**  $D$  sur  $\mathcal{X} \times \mathcal{Y}$ .
  - On dénote cette relation par  $(\mathbf{x}, y) \sim D$
- L'*ensemble d'entraînement* est  $S = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_m, y_m)\} \sim D^m$ .
- Un algorithme d'apprentissage reçoit en entrée l'ensemble  $S$  et produit un *classificateur*  $h : \mathcal{X} \mapsto \mathcal{Y}$  en sortie.



Nous voulons trouver un classificateur  $h$  qui **minimise** le *risque*, c'est à dire la **probabilité de faire une erreur** sur un **nouvel exemple** tiré selon  $D$ .

$$\begin{aligned} R_D(h) &\triangleq \Pr_{(x,y) \sim D} (h(\mathbf{x}) \neq y) \\ &= \mathbf{E}_{(x,y) \sim D} \mathbf{I}[h(\mathbf{x}) \neq y], \end{aligned}$$

où  $\mathbf{I}[a]$  est la fonction indicatrice, qui retourne 1 si le prédicat  $a$  est vrai, et 0 autrement.

- Mais nous ne connaissons pas  $D$ ...



Nous connaissons  $S$  ! Le *risque empirique* est défini comme suit :

$$\begin{aligned} R_S(h) &= \mathbf{E}_{(x,y) \sim S} \mathbf{I}[h(\mathbf{x}) \neq y] \\ &= \frac{1}{m} \sum_{k=1}^m \mathbf{I}[h(\mathbf{x}_k) \neq y_k] . \end{aligned}$$

Mais minimiser le risque empirique **ne permet pas de bien généraliser** (et c'est NP-difficile).

## LES MÉTHODES D'ENSEMBLE

---

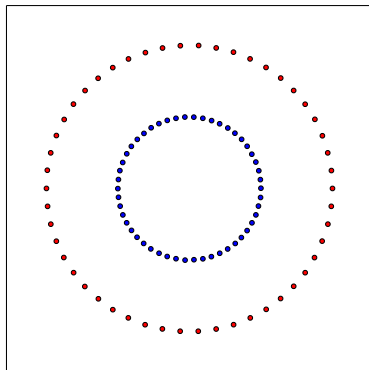
Nous considérons les classificateurs qui prennent la forme d'un *vote de majorité Q-pondéré*. Soit  $\mathcal{H}$  un ensemble de *votants* de la forme  $f: \mathcal{X} \mapsto [-1, 1]$  et soit  $Q$  une distribution sur  $\mathcal{H}$ , le vote de majorité  $B_Q$  est défini par

$$B_Q(\mathbf{x}) \triangleq \operatorname{sgn} \left[ \mathbf{E}_{f \sim Q} f(\mathbf{x}) \right],$$

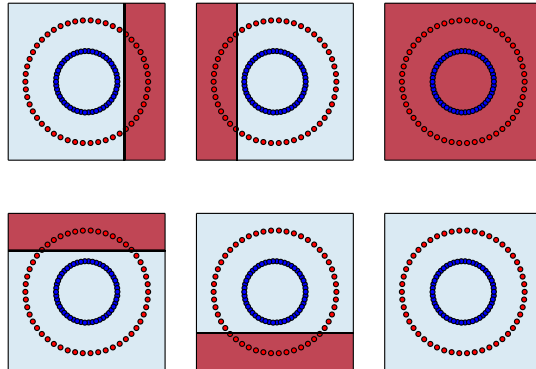
où  $\operatorname{sgn}[a] = 1$  si  $a > 0$  et  $-1$  autrement.

- Algorithmes de type *Bagging*
- Algorithmes de type *Boosting*
- *Random Forests*
- *Bayesian model averaging*
- *Support Vector Machines*

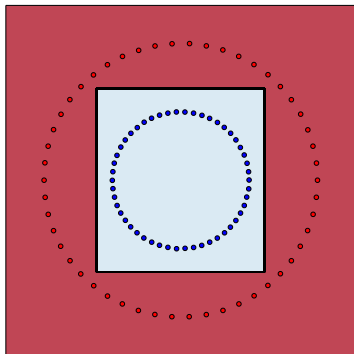
Voici un ensemble de données en deux dimensions. Chaque point rouge est un exemple négatif, et chaque point bleu un exemple positif.



Voici la frontière de décision d'un ensemble  $\mathcal{H}$  de 6 votants.



Combiner ces votants avec une distribution  $Q$  soigneusement choisie permet d'obtenir le classificateur suivant :



- Chaque votant est individuellement peu performant
- Mais la combinaison Q-pondérée peu l'être !



Étant donné une distribution  $D$  sur  $\mathcal{X} \times \mathcal{Y}$  et un ensemble de votants  $\mathcal{H}$ , nous cherchons une distribution  $Q$  sur  $\mathcal{H}$  qui minimise

$$R_D(B_Q) = \mathbf{E}_{(x,y) \sim D} \mathbf{I}[B_Q(\mathbf{x}) \neq y] .$$

Encore une fois, nous ne connaissons pas  $D$ , et minimiser le risque empirique n'est pas une bonne idée. Cherchons un **substitut** (*surrogate*)...

LA C-BORNE

---

La *marge*  $M_Q$  d'un vote de majorité sur un exemple  $(\mathbf{x}, y)$  est définie comme la proportion du vote accordée à la bonne classe, moins la proportion du vote accordée à la mauvaise classe. Dans le cas de la classification binaire, on a

$$M_Q(\mathbf{x}, y) \triangleq y \mathbf{E}_{f \sim Q} f(\mathbf{x}).$$

Lorsque la marge est **positive**, le vote de majorité ne fait **pas d'erreur**. Si elle est **négative**, celui-ci fait une **erreur**. Alors,

$$R_D(B_Q) = \Pr_{(\mathbf{x}, y) \sim D} (M_Q(\mathbf{x}, y) \leq 0) .$$

Soit  $D'$  une distribution sur  $\mathcal{X} \times \mathcal{Y}$ , posons  $M_Q^{D'}$  la variable aléatoire qui retourne la marge sur un exemple  $(\mathbf{x}, y)$  tiré selon  $D'$ .

Les deux premiers *moments statistiques* de cette variable aléatoire sont :

$$\mu_1(M_Q^{D'}) \triangleq \mathbf{E}_{(\mathbf{x}, y) \sim D'} M_Q(\mathbf{x}, y),$$

$$\mu_2(M_Q^{D'}) \triangleq \mathbf{E}_{(\mathbf{x}, y) \sim D'} M_Q(\mathbf{x}, y)^2.$$

En appliquant l'**inégalité de Markov** sur la probabilité que la variable aléatoire  $M_Q^D$  soit négative, nous obtenons l'inégalité suivante :

$$R_D(B_Q) \leq 1 - \mu_1(M_Q^D).$$

Autrement dit, si on veut un petit risque, **les votants doivent être « confiants »** en moyenne, ce qui est possible avec des votants **forts** individuellement.

En appliquant l'**inégalité de Cantelli-Chebyshev** sur la probabilité qu'elle soit négative, nous obtenons l'inégalité suivante.

**Théorème (La C-borne (LACASSE et al., 2006 ; LAVIOLETTE, MARCHAND et ROY, 2011))**

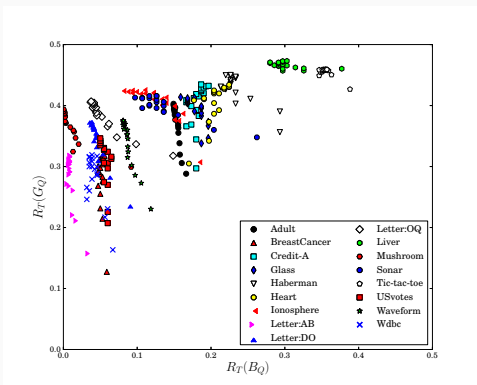
$$R_D(B_Q) \leq C_Q^D \triangleq 1 - \frac{\mu_1(M_Q^D)^2}{\mu_2(M_Q^D)}.$$

En d'autres mots, si on veut un petit risque, les votants doivent être **confiants** en moyenne **et/ou** il doivent avoir un **grand désaccord** (ils doivent faire leurs erreurs à des endroits différents). Également possible avec des **votants faibles** !

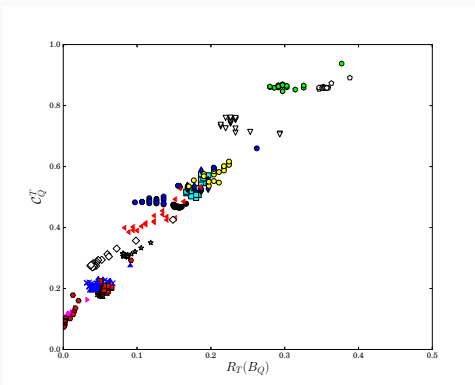
Nous avons généré des votes de majorité en utilisant des ensembles de données, en les séparant en un ensemble d'entraînement  $S$  et un ensemble de test  $T$ .

Nous avons exécuté l'algorithme AdaBoost sur  $S$  sur des votants faibles (des *souches de décision*) et avons récupéré des votes de majorité pendant et après l'exécution de l'algorithme.

Pour chaque vote de majorité, nous avons calculé le premier moment de la marge via la métrique  $R_T(G_Q) \triangleq \frac{1}{2}(1 - \mu_1(M_Q^T))$ , la valeur de  $\mathcal{C}_Q^T$ , et le risque  $R_T(B_Q)$ .



(a) Relation entre  $\frac{1}{2} (1 - \mu_1(M_Q^T))$  et  $R_T(B_Q)$ .



(b) Relation entre  $C_Q^T$  et  $R_T(B_Q)$ .



La  $\mathcal{C}$ -borne dépend des deux premiers moments de la marge du *vrai* risque du vote de majorité. Hors, nous ne pouvons pas calculer ces valeurs, nous devons les estimer empiriquement.

Nous présentons comment **borner** ces valeurs en fonction de leur estimation empirique, **uniformément pour toute distribution  $Q$** .

Les bornes PAC-bayésiennes permettent de **borner le « vrai » risque** du classificateur par vote de majorité à partir d'une **estimation empirique**.

Traditionnellement, le « vrai » premier moment de la marge est borné par son estimation empirique, puis le résultat est obtenu par l'inégalité  $R_D(B_Q) \leq 1 - \mu_1(M_Q^D)$ .

En présence de votants forts, ces bornes peuvent être très serrées.

En présence de **votants faibles**, nous devons également considérer le **second moment de la marge**. Le théorème suivant borne le risque du classificateur par vote de majorité en utilisant les estimations empiriques des deux premiers moments de la marge.

## Théorème

Pour toute distribution  $D$  sur  $\mathcal{X} \times \mathcal{Y}$ , pour tout ensemble  $\mathcal{H}$  de votants  $f: \mathcal{X} \mapsto [-1, 1]$ , pour toute distribution a priori  $P$  sur  $\mathcal{H}$ , et pour tout  $\delta \in (0, 1]$ , nous avons

$$\Pr_{S \sim D^m} \left( \begin{array}{l} \text{Pour toute distribution } Q \text{ sur } \mathcal{H}, \\ R_D(B_Q) \leq 1 - \frac{\left( \mu_1(M_Q^S) - \sqrt{\frac{2}{m} \left[ \text{KL}(Q \| P) + \ln \frac{\sqrt{m}}{\delta/2} \right]} \right)^2}{\mu_2(M_Q^S) + \sqrt{\frac{2}{m} \left[ 2 \text{KL}(Q \| P) + \ln \frac{\sqrt{m}}{\delta/2} \right]}} \geq 1 - \delta, \end{array} \right)$$

où  $\text{KL}(Q \| P)$  est la divergence Kullback-Leibler entre les distributions  $Q$  et  $P$ .

Ce résultat nous suggère que de trouver une distribution  $Q$  l'estimation empirique  $\mathcal{C}_Q^S$  de la  $\mathcal{C}$ -borne devrait donner un vote de majorité ayant une bonne erreur de généralisation.<sup>1</sup>

---

<sup>1</sup>Le terme  $\text{KL}(Q\|P)$  est ignoré dans cette présentation. LAVIOLETTE, MARCHAND et ROY (2011) présentent une version de cette borne qui, au prix d'une restriction supplémentaire sur les distributions  $Q$  considérées, ne contiennent pas de terme  $\text{KL}(Q\|P)$ .

**MINCQ : MINIMISER LA C-BORNE**

---

Minimiser la  $\mathcal{C}$ -borne empirique pose problème : comme nous sommes en présence d'une faible marge, sa forme est très près d'une forme indéterminée 0/0.

$$\mathcal{C}_Q^S = 1 - \frac{\mu_1(M_Q^S)^2}{\mu_2 M_Q^S}.$$

LAVIOLETTE, MARCHAND et ROY (2011) ont montré que de **forcer** le premier moment  $\mu_1(M_Q^S)$  à être **égal** à une petite valeur  $\mu > 0$  **ne restreint pas l'ensemble des votes de majorité possibles**.

**Pour minimiser la  $\mathcal{C}$ -borne**, nous pouvons donc fixer la marge moyenne à être égale à une certaine valeur, puis **se concentrer sur le deuxième moment (le désaccord)**.



## MinCq<sup>2</sup>

Étant donné un ensemble  $\mathcal{H}$  de votants, un ensemble d'entraînement  $S$  et une valeur  $\mu > 0$ , MinCq consiste à trouver la distribution  $Q$  sur  $\mathcal{H}$  dont  $\mu_1(M_Q^S) = \mu$  qui **minimise**  $\mu_2(M_Q^S)$ .

---

<sup>2</sup>La version de MinCq minimisant directement une borne PAC-bayésienne doit également considérer une restriction supplémentaire sur  $Q$ , et nécessite également que  $\mathcal{H}$  soit symétrique.

Soit  $\mathcal{H}$  un ensemble de votants  $f: \mathcal{X} \mapsto [-1, 1]$ ,  $\mathbf{H}$  une matrice de  $m \times n$  éléments telle que  $H_{ki} = f_i(x_k)$ ,  $\mathbf{q}$  un vecteur de  $n$  éléments représentant les poids  $Q$  sur les votants et  $\mathbf{y}$  un vecteur de  $m$  éléments contenant les étiquettes sur chaque exemple de l'ensemble  $S$ . Nous avons

$$\begin{aligned}\mu_1(M_Q^S) &= \frac{1}{m} \sum_{k=1}^m M_Q(\mathbf{x}_k, y_k) \\ &= \frac{1}{m} \sum_{k=1}^m \left( y_k \sum_{i=1}^n q_i f_i(\mathbf{x}_k) \right) \\ &= \frac{1}{m} \sum_{k=1}^m \sum_{i=1}^n y_k q_i H_{ki} \\ &= \frac{1}{m} \mathbf{y}^\top \mathbf{H} \mathbf{q},\end{aligned}$$

et

$$\begin{aligned}\mu_2(M_Q^S) &= \frac{1}{m} \sum_{k=1}^m M_Q^2(\mathbf{x}_k, y_k) \\ &= \frac{1}{m} \sum_{k=1}^m \left( y_k \sum_{i=1}^n q_i f_i(\mathbf{x}_k) \right)^2 \\ &= \frac{1}{m} \sum_{k=1}^m \sum_{i=1}^n \sum_{j=1}^n y_k^2 q_i q_j H_{ki} H_{kj} \\ &= \frac{1}{m} \mathbf{q}^\top \mathbf{H}^\top \mathbf{H} \mathbf{q}.\end{aligned}$$

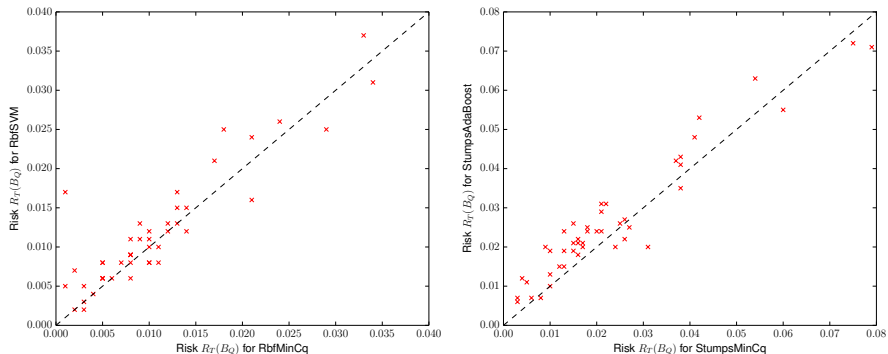
$$\begin{aligned} \text{Résoudre : } & \operatorname{argmin}_{\mathbf{q}} \frac{1}{m} \mathbf{q}^{\top} \mathbf{H}^{\top} \mathbf{H} \mathbf{q} \\ \text{sous contraintes : } & \frac{1}{m} \mathbf{y}^{\top} \mathbf{H} \mathbf{q} = \mu, \\ & \mathbf{q} \succeq \mathbf{0}, \end{aligned}$$

où  $\mathbf{0}$  est un vecteur de  $m$  zéros.

Nous considérons 45 **ensembles de données** de classification binaire tirés de la base de données *MNIST* (LECUN et CORTES).

Nous comparons **MinCq** en utilisant comme votants de base des *noyaux Radial Basis Function (RBF)* à l'algorithme **Support Vector Machine (SVM)** (CORTES et VAPNIK, 1995), un *algorithme à noyaux* dont la performance est état de l'art.

Nous comparons ensuite **MinCq**, cette fois-ci en utilisant des souches de décisions comme votants de base, avec **AdaBoost** (E. SCHAPIRE et SINGER, 1999).



**FIG. 2 :** Comparaison de MinCq avec SVM (à gauche) et AdaBoost (à droite). Chaque point représente une paire de risques sur l'ensemble de test. Un point au dessus de la diagonale indique une meilleure performance de MinCq.

MinCq est performant, mais retourne un vote de majorité **dense** (tous les votants de base sont considérés, et ont pratiquement tous un poids non nul).

Nous désirons créer une version de MinCq qui **choisit itérativement les votants à ajouter au vote de majorité**, et qui s'arrête avec une solution **parcimonieuse**.

## CQBOOST : UN ALGORITHME DE TYPE *BOOSTING*

---



Nous désirons créer une version de MinCq qui **choisit itérativement les votants à ajouter au vote de majorité**, et qui s'arrête avec une solution **parcimonieuse**.

Pour ce faire, nous avons besoin d'une manière d'évaluer la **qualité** des votants (du point de vue de MinCq), pour décider d'un **ordre** dans lequel les considérer.

Nous avons également besoin d'une méthode d'optimisation correspondant à cette idée : ajouter itérativement les votants dans le problème d'optimisation.

La *génération de colonnes* (*column generation*) est une technique d'optimisation qui a été utilisée avec succès pour rendre des algorithmes de type *Boosting* plus « *tractables* ».

Elle a notamment été utilisée pour développer LPBoost (DEMIRIZ, Kristin P BENNETT et SHAWE-TAYLOR, 2002) dans le cadre de la programmation linéaire, et CG-Boost (BI, ZHANG et Kristin P. BENNETT, 2004) pour la programmation quadratique.

- Chaque **colonne** de la matrice de classification **H** correspond à un votant de base.
- Un **sous-ensemble des colonnes** de cette matrice pourrait être **suffisant** pour obtenir l'**optimalité** du problème d'optimisation sous-jacent.
- La pondération  $Q$  obtenue est alors plus **parcimonieuse** (« *sparse* »)
  - Plus **rapide d'exécution**
  - Plus facilement **interprétable**

- Étant donné un problème d'optimisation **primal**, seulement un **sous-ensemble des classificateurs de base** est considéré. On appelle ce problème le « *restricted master problem* »
- Résoudre le problème primal restreint correspond à résoudre une **relaxation du problème dual**, où les contraintes reliées aux classificateurs de base non considérés sont absentes
- En ajoutant itérativement des colonnes à considérer, l'algorithme **converge** vers l'optimum du problème d'optimisation original, **possiblement avant d'avoir généré toutes les colonnes**
- Si aucune colonne ne viole la contrainte du problème dual, l'algorithme s'arrête car on a optimalité

Plusieurs techniques d'optimisation existent pour obtenir une formulation *duale* d'un problème d'optimisation.

Généralement, les **variables** du problème d'optimisation deviennent des **contraintes** dans le problème dual, et vice-versa.

Une formulation duale intéressante peut nous aider à voir le problème d'un différent point de vue.

Construire une formulation duale d'un problème ne donne pas toujours un problème d'optimisation **intéressant**. Il faut parfois transformer le problème primal pour obtenir un dual intéressant.

Introduisons des variables  $\gamma_k$  égales aux **marges sur chaque exemple**  $k$ .

$$\begin{aligned}\gamma_k &= M_Q(\mathbf{x}_k, y_k) \\ &= y_k \mathbf{E}_{f \sim Q} f(\mathbf{x}) \\ &= y_k \sum_{i=1}^n q_i H_{ki}, \quad \forall k \in \{1, \dots, m\}.\end{aligned}$$

On a donc

$$\boldsymbol{\gamma} = \text{diag}(\mathbf{y}) \mathbf{H} \mathbf{q}.$$

$$\begin{aligned} \text{Résoudre : } & \operatorname{argmin}_{\mathbf{q}, \boldsymbol{\gamma}} \frac{1}{m} \boldsymbol{\gamma}^\top \boldsymbol{\gamma} \\ \text{sous contraintes : } & \frac{1}{m} \mathbf{1}^\top \boldsymbol{\gamma} = \mu, \\ & \boldsymbol{\gamma} = \operatorname{diag}(\mathbf{y}) \mathbf{H} \mathbf{q} \\ & \mathbf{q} \succeq \mathbf{0}, \end{aligned}$$

où  $\mathbf{1}$  est un vecteur de  $m$  uns, et où  $\mathbf{0}$  est un vecteur de  $m$  zéros.

Utilisons la méthode de *Lagrange* pour construire notre formulation duale.

En additionnant une somme pondérée des contraintes à la fonction objectif, nous obtenons le *Lagrangien* du problème d'optimisation.

$$\Lambda(\mathbf{q}, \boldsymbol{\gamma}, \boldsymbol{\alpha}, \beta, \boldsymbol{\xi}) \triangleq \frac{1}{m} \boldsymbol{\gamma}^\top \boldsymbol{\gamma} + \boldsymbol{\alpha}^\top (\boldsymbol{\gamma} - \text{diag}(\mathbf{y}) \mathbf{H} \mathbf{q}) + \beta \left( \frac{1}{m} \mathbf{1}^\top \boldsymbol{\gamma} - \mu \right) - \boldsymbol{\xi}^\top \mathbf{q}.$$

Les variables  $\boldsymbol{\alpha}$ ,  $\beta$  et  $\boldsymbol{\xi}$  sont nommés les *multiplicateurs de Lagrange, variables duales*.



Le *Lagrangien dual* est le minimum du Lagrangien, en fonction des variables primales  $\mathbf{q}$  et  $\gamma$ .

$$\Lambda^D(\boldsymbol{\alpha}, \beta, \boldsymbol{\xi}) \triangleq \inf_{\mathbf{q}, \gamma} \Lambda(\mathbf{q}, \gamma, \boldsymbol{\alpha}, \beta, \boldsymbol{\xi}).$$

Le problème d'optimisation dual est obtenu en trouvant le **maximum** du Lagrangien dual, sous contrainte que les multiplicateurs de Lagrange associés aux contraintes d'inégalité du problème primal soient non négatives.

$$\text{Résoudre : } \underset{\alpha, \beta, \xi}{\operatorname{argmax}} \Lambda^D(\alpha, \beta, \xi)$$

$$\text{sous contraintes : } \xi \succeq \mathbf{0}.$$

Retournons à la définition du Lagrangien dual et évaluons sa valeur. Nous avons

$$\begin{aligned}\Lambda^D(\boldsymbol{\alpha}, \beta, \boldsymbol{\xi}) &\triangleq \inf_{\mathbf{q}, \boldsymbol{\gamma}} \Lambda(\mathbf{q}, \boldsymbol{\gamma}, \boldsymbol{\alpha}, \beta, \boldsymbol{\xi}) \\ &= \inf_{\mathbf{q}, \boldsymbol{\gamma}} \left[ \frac{1}{m} \boldsymbol{\gamma}^\top \boldsymbol{\gamma} + \boldsymbol{\alpha}^\top (\boldsymbol{\gamma} - \text{diag}(\mathbf{y}) \mathbf{H} \mathbf{q}) + \beta \left( \frac{1}{m} \mathbf{1}^\top \boldsymbol{\gamma} - \mu \right) - \boldsymbol{\xi}^\top \mathbf{q} \right].\end{aligned}$$

Le minimum en fonction de  $\mathbf{q}$  et  $\boldsymbol{\gamma}$  est caractérisé par les points où les gradients de  $\Lambda(\mathbf{q}, \boldsymbol{\gamma}, \boldsymbol{\alpha}, \beta, \boldsymbol{\xi})$  par rapport à  $\mathbf{q}$  et  $\boldsymbol{\gamma}$  sont nuls.

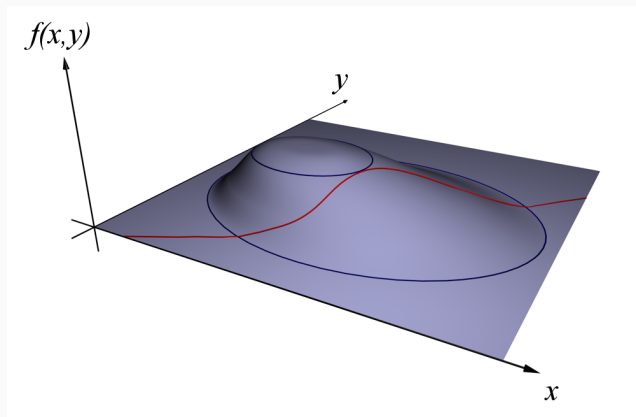


FIG. 3 : Graphique d'une fonction à maximiser et d'une contrainte d'égalité (en rouge). Source : Wikipédia.

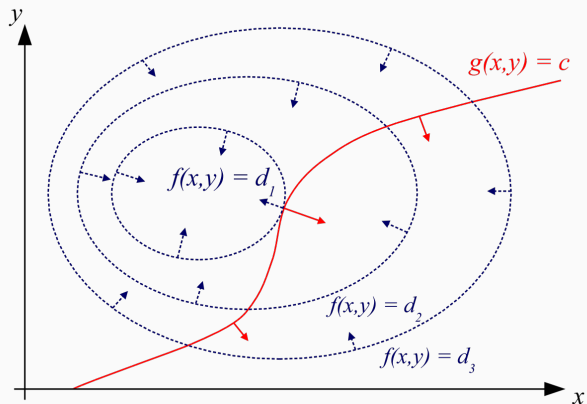


FIG. 4 : Courbes de niveaux (en bleu) d'une fonction à maximiser et d'une contrainte d'égalité (en rouge). Le point tangent où les courbes se recoupent est la solution optimale. Source : Wikipédia.

Le point optimal du Lagrangien en fonction de  $\mathbf{q}$  et  $\boldsymbol{\gamma}$  est donc caractérisé par :

$$\mathbf{H}^T \text{diag}(\mathbf{y}) \boldsymbol{\alpha} = -\boldsymbol{\xi}$$

et

$$\boldsymbol{\gamma} = -\frac{m}{2} \boldsymbol{\alpha} - \frac{\beta}{2} \mathbf{1}.$$

Nous connaissons donc maintenant une **contrainte d'égalité** qui doit être respectée à l'optimum, et la **valeur optimale** de  $\boldsymbol{\gamma}$  et que nous pouvons remplacer dans l'expression du Lagrangien.

De laborieux (mais simples) calculs nous donne donc, sous contrainte  $\mathbf{H}^\top \text{diag}(\mathbf{y}) \boldsymbol{\alpha} = -\boldsymbol{\xi}$ , que :

$$\begin{aligned}\Lambda^D(\boldsymbol{\alpha}, \beta, \boldsymbol{\xi}) &= \inf_{\mathbf{q}, \gamma} \Lambda(\mathbf{q}, \gamma, \boldsymbol{\alpha}, \beta, \boldsymbol{\xi}) \\ &= -\frac{m}{4} \boldsymbol{\alpha}^\top \boldsymbol{\alpha} - \frac{\beta}{2} \mathbf{1}^\top \boldsymbol{\alpha} - \frac{\beta^2}{4} - \beta \boldsymbol{\mu}.\end{aligned}$$

$$\begin{aligned} \text{Résoudre : } & \operatorname{argmax}_{\boldsymbol{\alpha}, \beta, \boldsymbol{\xi}} -\frac{m}{4} \boldsymbol{\alpha}^{\top} \boldsymbol{\alpha} - \frac{\beta}{2} \mathbf{1}^{\top} \boldsymbol{\alpha} - \frac{\beta^2}{4} - \beta \mu \\ \text{sous contraintes : } & \mathbf{H}^{\top} \operatorname{diag}(\mathbf{y}) \boldsymbol{\alpha} = -\boldsymbol{\xi}, \\ & \boldsymbol{\xi} \succeq \mathbf{0}. \end{aligned}$$

Les variables  $\boldsymbol{\xi}$  ne faisant pas partie de la fonction objectif, celles-ci peuvent être retirées.



$$\text{Résoudre : } \operatorname{argmax}_{\boldsymbol{\alpha}, \beta} -\frac{m}{4}\boldsymbol{\alpha}^\top \boldsymbol{\alpha} - \frac{\beta}{2}\mathbf{1}^\top \boldsymbol{\alpha} - \frac{\beta^2}{4} - \beta\mu$$

$$\text{sous contraintes : } \mathbf{H}^\top \operatorname{diag}(\mathbf{y})\boldsymbol{\alpha} \preceq 0.$$

Nous remarquons que le problème d'optimisation retourne maintenant un **vecteur de poids sur les exemples** plutôt que sur les votants. Une variable  $\beta$  contrôle un compromis entre la partie quadratique et la partie linéaire de la fonction objectif.

Le terme de gauche  $\mathbf{H}^\top \text{diag}(\mathbf{y}) \boldsymbol{\alpha}$  de la contrainte d'inégalité peut être vue comme un **« score » donné à chaque votant  $f$** . Cette mesure est appelée le « *edge* » d'un votant.

Pour chaque votant  $i \in \{0, \dots, n\}$ , nous avons :

$$(\mathbf{H}^\top \text{diag}(\mathbf{y}) \boldsymbol{\alpha})_i = \sum_{k=1}^m \alpha_k y_k H_{ki} = \sum_{k=1}^m \alpha_k y_k f_i(\mathbf{x}_k).$$

Cette valeur sera utilisée pour **guider le choix du prochain votant** à ajouter dans le vote de majorité. Celle-ci apparaît également dans d'autres algorithmes de génération de colonne, tels que LPBoost (DEMIRIZ, Kristin P BENNETT et SHAWE-TAYLOR, 2002) et CG-Boost (BI, ZHANG et Kristin P. BENNETT, 2004).

---

**Algorithm 1** CqBoost

---

- Posons  $t = 0$ ,  $\mathbf{q}$  un vecteur de  $n$  zéros et  $\boldsymbol{\alpha}$  un vecteur de  $m$  éléments égaux à  $\frac{-1}{m}$ .
- Soit  $\hat{\mathbf{H}}$  une matrice vide de  $m$  lignes.

**loop**

- Choisir la colonne  $i$  qui maximise  $\sum_{k=1}^m \alpha_k y_k H_{ki}$ .
- **If** la colonne  $i$  ne viole pas la contrainte dual  $\sum_{k=1}^m \alpha_k y_k H_{ki} \leq 0 + \epsilon$  **then break**.
- Ajouter la colonne  $i$  à la matrice  $\hat{\mathbf{H}}$ .
- Mettre à jour  $\mathbf{q}$  et  $\boldsymbol{\alpha}$  avec la solution primale et duale de MinCq.

**end loop**

**return**  $\mathbf{q}$

---

Sur plusieurs ensembles de données provenant du *UCI Machine Learning Repository* (BLAKE et MERZ, 1998), nous comparons CqBoost avec l'algorithme original MinCq, mais également CqBoost avec LPBoost et CG-Boost.

Nous utilisons dans un premier temps des souches de décision comme votants de base, puis ensuite des noyaux RBF.

# RÉSULTATS EMPIRIQUES : SOUCHES DE DÉCISIONS

Ens. de données	CqBoost		MinCq	
	$R_T(B_Q)$	Colonnes	$R_T(B_Q)$	Colonnes
Letter :AB	<b>0.0039</b>	<b>160</b>	0.0051	320
australian	0.1855	<b>70</b>	0.1855	280
balance	0.0256	<b>17</b>	0.0256	80
breast	0.0458	<b>83</b>	0.0458	180
car	0.1516	<b>16</b>	0.1516	120
cmc	0.2785	<b>36</b>	0.2785	180
credit	0.1391	<b>84</b>	0.1391	300
cylinder	0.3852	<b>224</b>	<b>0.3741</b>	700
ecoli	0.0714	<b>54</b>	0.0714	140
glass	<b>0.2430</b>	<b>80</b>	0.2617	180
heart	0.2593	<b>65</b>	0.2593	260
hepatitis	0.2468	<b>78</b>	<b>0.2078</b>	380
horse	0.3696	<b>152</b>	<b>0.3533</b>	520
ionosphere	0.1943	<b>179</b>	<b>0.1029</b>	680
monks	0.2361	<b>12</b>	0.2361	120
pima	<b>0.2578</b>	<b>74</b>	0.2656	160
tictactoe	0.3215	<b>20</b>	0.3215	180
titanic	0.2282	<b>6</b>	0.2282	60
wine	0.1685	<b>95</b>	<b>0.0449</b>	260
yeast	0.3032	<b>67</b>	<b>0.2965</b>	160
zoo	0.0200	<b>20</b>	0.0200	320

# RÉSULTATS EMPIRIQUES : SOUCHES DE DÉCISIONS

Ens. de données	CqBoost		CG-Boost		LPBoost	
	$R_T(B_Q)$	Colonnes	$R_T(B_Q)$	Colonnes	$R_T(B_Q)$	Colonnes
Letter :AB	0.0039	160	<b>0.0026</b>	160	0.0129	<b>51</b>
australian	0.1855	70	0.2348	147	<b>0.1449</b>	<b>1</b>
balance	<b>0.0256</b>	17	0.0385	40	0.0288	<b>16</b>
breast	0.0458	83	<b>0.0372</b>	90	0.0430	<b>24</b>
car	<b>0.1516</b>	16	0.2569	60	0.1678	<b>14</b>
cmc	<b>0.2785</b>	36	0.3084	113	0.3084	<b>35</b>
credit	0.1391	84	0.1768	161	<b>0.1333</b>	<b>1</b>
cylinder	0.3852	224	<b>0.3630</b>	330	<b>0.3630</b>	<b>1</b>
ecoli	<b>0.0714</b>	54	0.0833	70	0.0893	<b>25</b>
glass	0.2430	80	<b>0.2243</b>	98	0.3645	<b>1</b>
heart	0.2593	65	<b>0.2296</b>	130	0.2741	<b>38</b>
hepatitis	0.2468	78	0.2078	190	<b>0.1818</b>	<b>1</b>
horse	0.3696	152	0.3696	256	<b>0.2065</b>	<b>1</b>
ionosphere	0.1943	179	0.1543	330	<b>0.1086</b>	<b>63</b>
monks	0.2361	<b>12</b>	<b>0.2315</b>	114	<b>0.2315</b>	21
pima	0.2578	74	0.3594	80	<b>0.2474</b>	<b>40</b>
tictactoe	<b>0.3215</b>	20	0.3779	90	0.3549	<b>1</b>
titanic	<b>0.2282</b>	6	0.2309	18	0.2309	<b>1</b>
wine	0.1685	95	<b>0.0562</b>	130	0.0899	<b>1</b>
yeast	0.3032	67	<b>0.2992</b>	81	0.3747	<b>1</b>
zoo	<b>0.0200</b>	20	0.1000	152	0.0400	<b>10</b>

# RÉSULTATS EMPIRIQUES : NOYAUX RBF

Ens. de données	CqBoost		MinCq	
	$R_T(B_Q)$	Colonnes	$R_T(B_Q)$	Colonnes
Letter :AB	<b>0.0103</b>	<b>20</b>	0.0180	778
australian	0.1449	<b>14</b>	<b>0.1333</b>	345
balance	0.0737	<b>10</b>	<b>0.0577</b>	313
breast	0.0458	<b>38</b>	<b>0.0372</b>	350
car	<b>0.0718</b>	<b>34</b>	0.2836	864
cmc	0.3043	<b>9</b>	<b>0.2908</b>	737
credit	<b>0.1275</b>	<b>72</b>	0.1391	345
cylinder	<b>0.3037</b>	<b>59</b>	0.3148	270
ecoli	<b>0.0714</b>	<b>40</b>	0.0893	168
glass	<b>0.1963</b>	<b>21</b>	0.2617	107
heart	0.1630	<b>3</b>	<b>0.1481</b>	135
hepatitis	0.2078	<b>39</b>	<b>0.1948</b>	78
horse	0.1848	<b>15</b>	<b>0.1793</b>	184
ionosphere	0.1486	<b>14</b>	<b>0.1257</b>	176
monks	<b>0.2037</b>	<b>43</b>	0.3426	216
pima	<b>0.2474</b>	<b>5</b>	0.2630	384
tictactoe	<b>0.2129</b>	<b>19</b>	0.3403	479
titanic	<b>0.2164</b>	<b>15</b>	0.2309	1101
wine	0.0337	<b>9</b>	<b>0.0225</b>	89
yeast	<b>0.2736</b>	<b>23</b>	0.3369	742
zoo	0.0400	<b>18</b>	0.0400	51



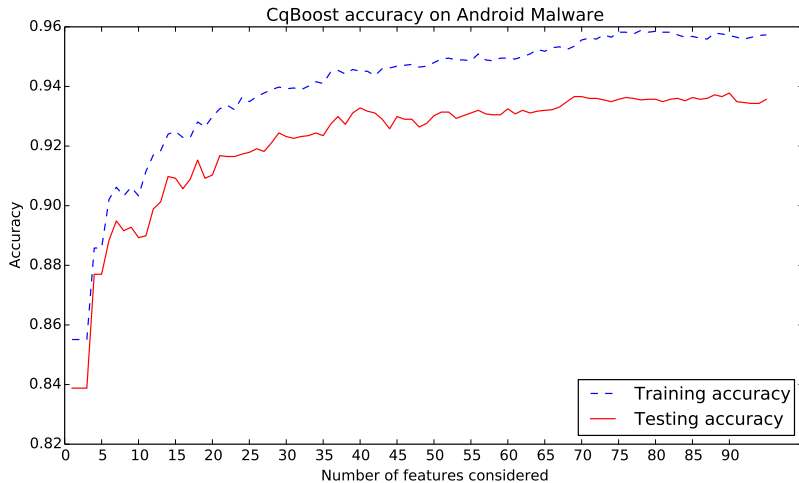
# RÉSULTATS EMPIRIQUES : NOYAUX RBF

Ens. de données	CqBoost		CG-Boost		LPBoost	
	$R_T(B_Q)$	Colonnes	$R_T(B_Q)$	Colonnes	$R_T(B_Q)$	Colonnes
Letter :AB	<b>0.0103</b>	<b>20</b>	0.0116	359	0.0142	26
australian	0.1449	<b>14</b>	<b>0.1333</b>	295	0.1449	33
balance	0.0737	10	<b>0.0288</b>	98	0.4455	<b>1</b>
breast	0.0458	38	<b>0.0430</b>	141	<b>0.0430</b>	<b>9</b>
car	0.0718	<b>34</b>	0.0729	567	<b>0.0243</b>	90
cmc	0.3043	<b>9</b>	<b>0.2948</b>	447	0.3003	98
credit	<b>0.1275</b>	72	0.1333	263	0.1333	<b>62</b>
cylinder	0.3037	<b>59</b>	<b>0.2741</b>	194	<b>0.2741</b>	116
ecoli	<b>0.0714</b>	40	0.1012	139	0.5060	<b>1</b>
glass	<b>0.1963</b>	<b>21</b>	0.2804	82	0.2150	44
heart	<b>0.1630</b>	3	0.4815	114	0.4815	<b>1</b>
hepatitis	0.2078	39	0.2078	66	<b>0.1948</b>	<b>12</b>
horse	<b>0.1848</b>	15	0.2011	151	0.3641	<b>8</b>
ionosphere	0.1486	<b>14</b>	<b>0.1200</b>	142	0.1429	41
monks	<b>0.2037</b>	43	0.2546	163	0.5093	<b>1</b>
pima	<b>0.2474</b>	<b>5</b>	0.2552	202	0.2578	69
tictactoe	0.2129	<b>19</b>	0.3883	340	<b>0.0480</b>	170
titanic	<b>0.2164</b>	15	0.2309	769	0.3200	<b>1</b>
wine	<b>0.0337</b>	<b>9</b>	0.4157	85	0.0449	20
yeast	0.2736	<b>23</b>	0.2749	422	<b>0.2709</b>	66
zoo	<b>0.0400</b>	18	0.1000	46	0.1000	<b>12</b>

Sur un ensemble de données de logiciels Android dont l'étiquette est *malicieux* ou *non-malicieux*, nous exécutons CqBoost sur des votants représentant une permission ou un comportement de l'application.

La figure suivante présente le risque sur l'ensemble d'entraînement ainsi que le risque sur l'ensemble de test, en fonction du nombre de colonnes (votants) ajoutées dans le vote de majorité.

# RÉSULTATS EMPIRIQUES : ANDROID MALWARE











## CONCLUSION

---

- Nous avons introduit le problème de la classification binaire
- Nous avons formalisé le problème à résoudre
- Nous avons introduit les votes de majorité (les méthodes d'ensemble)
- Nous avons présenté la  $\mathcal{C}$ -borne, une borne supérieure du risque
- Nous avons présenté MinCq, un algorithme minimisant cette borne
- Nous avons présenté CqBoost, une méthode d'ensemble rendant les solutions plus parcimonieuses

QUESTIONS?

-  BI, Jinbo, Tong ZHANG et Kristin P. BENNETT (2004). “Column-generation boosting methods for mixture of kernels”. In : *Proceedings of the 2004 ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '04*, p. 521.
-  BLAKE, C.L. et C.J. MERZ (1998). *UCI Repository of machine learning databases*.  
<http://www.ics.uci.edu/~mlern/MLRepository.html> : Department of Information et Computer Science, Irvine, CA : University of California.
-  CORTES, Corinna et Vladimir VAPNIK (1995). “Support-Vector Networks”. In : *Machine Learning* 20.3, p. 273–297.
-  DEMIRIZ, Ayhan, Kristin P BENNETT et John SHAWE-TAYLOR (2002). “Linear programming boosting via column generation”. In : *Machine Learning* 46.1-3, p. 225–254.
-  E. SCHAPIRE, Robert et Yoram SINGER (1999). “Improved Boosting Using Confidence-rated Predictions”. In : *Machine Learning* 37.3, p. 297–336.
-  LACASSE, Alexandre et al. (2006). “PAC-Bayes Bounds for the Risk of the Majority Vote and the Variance of the Gibbs Classifier”. In : *NIPS*, p. 769–776.
-  LAVIOLETTE, François, Mario MARCHAND et Jean-François ROY (2011). “From PAC-Bayes Bounds to Quadratic Programs for Majority Votes”. In : *ICML*, p. 649–656.
-  LECUN, Yann et Corinna CORTES. “The MNIST database of handwritten digits”. In : URL :  
<http://yann.lecun.com/exdb/mnist/>.