Ingénierie de la fouille et de la visualisation de données massives (RCP216)

Apprentissage à large échelle

Michel Crucianu
(prenom.nom@cnam.fr)
http://cedric.cnam.fr/vertigo/Cours/RCP216/

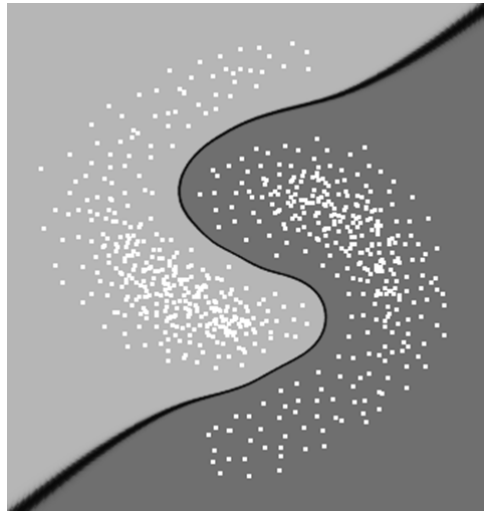
Département Informatique
Conservatoire National des Arts & Métiers, Paris, France

30 septembre 2025

Plan du cours

Apprentissage supervisé

- Construire un modèle décisionnel ou prédictif à partir de données « étiquetées » (dans l'illustration, supervision = étiquettes de classe)



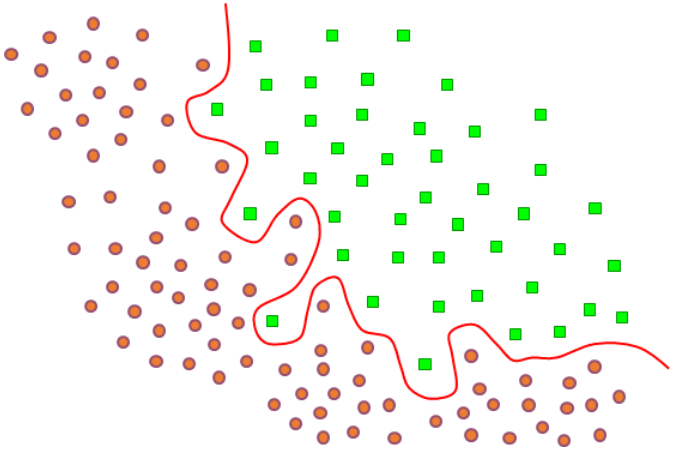
- Autres approches :
  - Apprentissage non supervisé : information de supervision indisponible (par ex. classification automatique)
  - Apprentissage semi-supervisé (voir [2]) : information de supervision disponible pour une (faible) partie des données d'apprentissage

Apprentissage supervisé, risque espéré, risque empirique

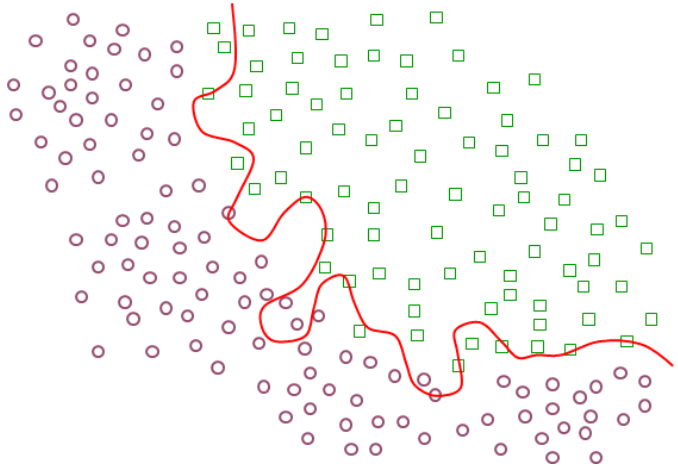
- Espace d'entrée $\mathcal{X}$ (par ex. $\mathbb{R}^p$), espace de sortie $\mathcal{Y}$ (par ex. $\{-1; 1\}$, $\mathbb{R}$)
- Variables aléatoires $(X, Y) \in \mathcal{X} \times \mathcal{Y}$ suivant la loi inconnue P
- Données $\mathcal{D}_N = \{(\mathbf{x}_i, y_i)\}_{1 \leq i \leq N}$ correspondant à des tirages indépendants et identiquement distribués, suivant P
- Supervision : $\{y_i\}_{1 \leq i \leq N}$
- Objectif : trouver, dans une famille $\mathcal{F}$, une fonction $f : \mathcal{X} \to \mathcal{Y}$ qui prédit Y à partir de $\mathbf{X}$ en minimisant le risque espéré (théorique) $R(f) = E_P[L(X, Y, f)]$
- Fonctions de « perte » L (ou d'erreur)
  - Perte quadratique pour la régression : $L(\mathbf{x}, y, f) = [f(\mathbf{x}) - y]^2$
  - Perte 0-1 pour la discrimination entre 2 classes : $L(\mathbf{x}, y, f) = \mathbf{1}_{f(\mathbf{x}) \neq y}$
- $R(f)$ ne peut pas être évalué car P est inconnue, mais on peut mesurer le risque empirique $R_{\mathcal{D}_N}(f) = \frac{1}{N} \sum_{i=1}^{N} L(\mathbf{x}_i, y_i, f)$

Minimisation du risque empirique régularisé

- Comment trouver le modèle qui présente le risque espéré le plus faible ?
  - Minimisation du risque empirique (MRE) : $\mathbf{w}^* = \arg\min_{\mathbf{w}} R_{\mathcal{D}_N}(f(\mathbf{w}))$
    $\Rightarrow$ souvent sur-apprentissage !



Risque empirique nul



Erreurs sur données de test

  - Minimisation du risque empirique régularisé (MRER) :
    $\mathbf{w}^* = \arg\min_{\mathbf{w}} \left[R_{\mathcal{D}_N}(f(\mathbf{w})) + \mathrm{Reg}(f(\mathbf{w}))\right]$
    où $\mathrm{Reg}(f(\mathbf{w}))$ pénalise la complexité du modèle

La validation croisée

- Estimation du risque espéré :
  - A partir du risque empirique et en tenant compte de bornes de généralisation : $R(f^*_{\mathcal{D}_N}) \leq R_{\mathcal{D}_N}(f^*_{\mathcal{D}_N}) + B(N, \mathcal{F})$
    - $\rightarrow$ Difficile d'obtenir des bornes utiles en pratique
  - Partitionnement de $\mathcal{D}_N$ en 2 ensembles mutuellement exclusifs et issus de la même distribution : données d'apprentissage et données de validation ; apprentissage du modèle sur les données d'apprentissage ; estimation du risque espéré par l'erreur du modèle sur les données de validation
    - $\rightarrow$ Pour réduire la variance de cet estimateur on moyenne les résultats obtenus sur plusieurs découpages différents
- Validation croisée - méthodes exhaustives :
  - *Leave p out* : $N - p$ données pour l'apprentissage et p pour la validation, l'estimation du risque théorique est la moyenne sur tous les C_N^p découpages possibles $\Rightarrow$ coût excessif
  - *Leave one out* : $N - 1$ données pour l'apprentissage et 1 pour la validation, moyenne sur tous les $C_N^1 = N$ découpages possibles $\Rightarrow$ coût élevé

La validation croisée (2)

- Validation croisée - méthodes non exhaustives :
  - *k-fold* : découpage fixé des N données en k parties, apprentissage sur $k-1$ parties et validation sur la k-ême ; la plus fréquemment utilisée (souvent $k = 10$)
  - Échantillonnage répété : échantillon aléatoire de n données pour la validation (les autres $N - n$ pour l'apprentissage), on répète cela k fois (sans la contrainte $k \cdot n = N$) ; désavantage : certaines données ne sont dans aucun échantillon, d'autres sont dans plusieurs échantillons
- Tous les partitionnements peuvent être explorés en parallèle !
- Propriétés statistiques :
  - L'estimateur obtenu a un biais (sur-estime le risque espéré car l'apprentissage se fait à chaque fois sur moins de N données) mais est asymptotiquement ($N \to \infty$) sans biais
  - La variance de l'estimateur reste élevée

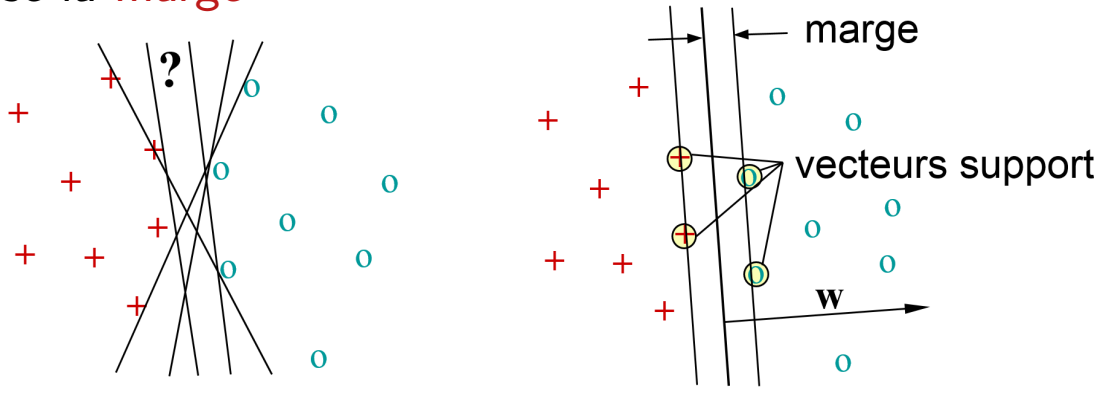
Apprentissage supervisé : problématique de l'échelle

- Déterminer le modèle décisionnel par apprentissage supervisé :
  - $\mathcal{D}_N$: N données de dimension p (par ex. vecteurs de $\mathbb{R}^p$) avec information de supervision
  - On fixe la famille (paramétrée) $\mathcal{F}$ de fonctions de décision et on cherche les paramètres optimaux (la fonction optimale) sur les données d'apprentissage par MRER à l'aide d'une méthode d'optimisation

1. Complexité de l'apprentissage :
   - Méthodes linéaires (SVM linéaires, régression linéaire, régression logistique) : $O(N \times p)$
   - Perceptrons multi-couches (non linéaires) : $O(N \times p^2)$
   - Méthodes à noyaux non linéaires : souvent $O(N^3 \times p)$
   - De plus, certaines méthodes d'optimisation se parallélisent mieux que d'autres !
2. Complexité de la décision (pour 1 nouvelle donnée) :
   - Méthodes linéaires : $O(p)$
   - Perceptrons multi-couches : $O(p^2)$
   - Méthodes à noyaux non linéaires : $O(v \times p)$, v étant par ex. le nombre de vecteurs support (pire des cas : $v \sim N$)

Machines à vecteurs de support

- *Support Vector Machines* (voir [3]) : « séparateurs à vastes marges » (pas les seuls « séparateurs » qui maximisent une marge !), « machines à vecteurs (de) support »
- Idée : lorsque plusieurs séparations (et même une infinité) sont possibles, préférer celle qui maximise la marge



- Pourquoi s'intéresser à la marge :
  - Maximiser la marge : pour des données à l'intérieur d'une hypersphère de rayon r, la VC-dimension h de la famille de hyperplans de marge $\frac{1}{\|\mathbf{w}\|} \geq m$ est bornée, $h \leq \left(\frac{r}{m}\right)^2 + 1 \Rightarrow$ augmenter la marge améliore les bornes de généralisation
  - Estimation du risque espéré par *leave one out cross-validation* : $R(f) \leq \frac{E[v(N)]}{N}$, v étant le nombre de vecteurs support ($\rightarrow$ intéressant si $v \ll N$)

SVM : problème d'optimisation *primal* pour séparation linéaire

- Soit $\mathbf{w}$ le vecteur normal à l'hyperplan de séparation et supposons que l'équation de cet hyperplan est $\mathbf{w}^T\mathbf{x} + b = 0$
- La distance entre un vecteur support $\mathbf{x}_{SV}$ et l'hyperplan est $\frac{\left|\mathbf{w}^T\mathbf{x}_{SV}+b\right|}{\|\mathbf{w}\|}$
- Avec la condition de normalisation $\left|\mathbf{w}^T\mathbf{x}_{SV} + b\right| = 1$ pour tous les vecteurs support on déduit que la marge est $\frac{1}{\|\mathbf{w}\|}$
- Maximiser la marge revient donc à :

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2$$

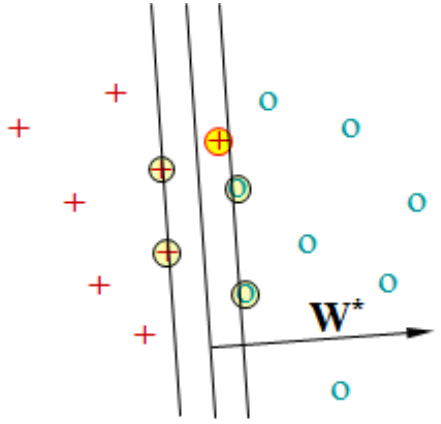
sous les contraintes d'inégalité (qui définissent la notion de marge) :

$$y_i\left(\mathbf{w}^T\mathbf{x}_i + b\right) \geq 1, \quad y_i \in \{-1; 1\}, \quad 1 \leq i \leq N$$

SVM : problème d'optimisation *primal* pour séparation linéaire (2)

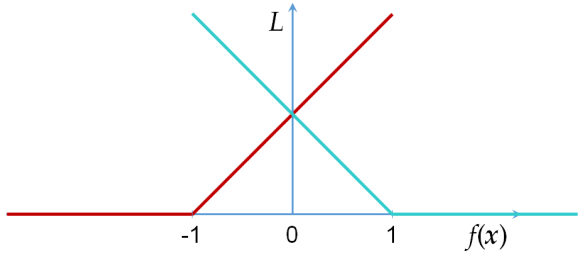
- Si on accepte que certaines données d'apprentissage soient dans la marge, voire même mal classées, ces contraintes deviennent : $y_i \left(\mathbf{w}^T \mathbf{x}_i + b\right) \geq 1 - \xi_i$

  avec $\xi_i \geq 0$, $1 \leq i \leq N$ (ξ_i étant l'« erreur » faite pour la donnée i)



- Remarque : on parle d'« erreur » dès qu'une donnée se trouve au-delà du « bord » de la classe, même si elle est encore du bon côté de la frontière de décision :

Hinge loss :



SVM : problème d'optimisation *primal* pour séparation linéaire (3)

- Le critère à optimiser est un compromis entre maximisation de la marge et minimisation de l'erreur d'apprentissage :

$$\min_{\mathbf{w},b} \left(\frac{1}{2}\|\mathbf{w}\|^2 + C \sum_{i=1}^{N} \xi_i \right) \tag{1}$$

- Plus la valeur de C est élevée, plus la minimisation de l'erreur d'apprentissage est importante par rapport à la maximisation de la marge
- La fonction de décision appliquée à une donnée $\mathbf{x}$

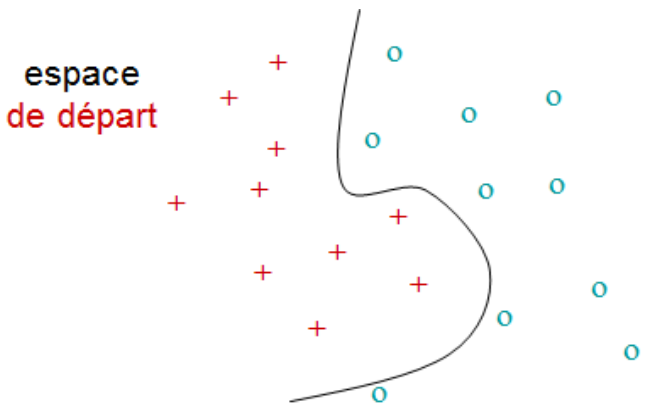
$$f^*(\mathbf{x}) = \mathbf{w}^{*T}\mathbf{x} + b^* \tag{2}$$

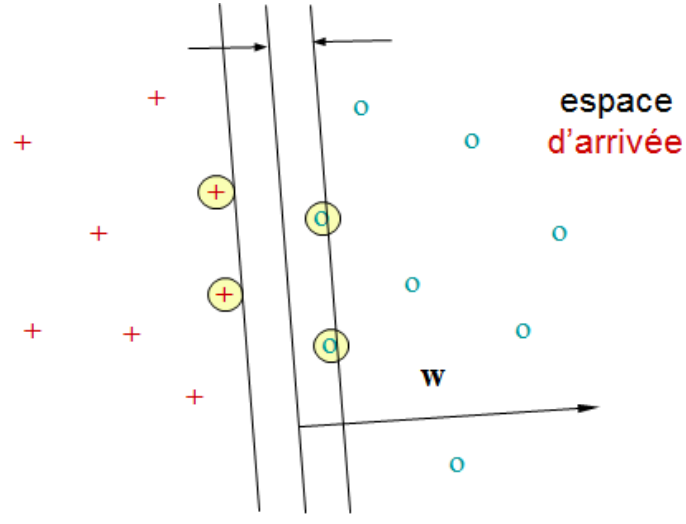
  s'annule sur la frontière de discrimination ; * désigne les paramètres optimaux (et la fonction optimale) trouvés par résolution du problème d'optimisation
- Obtenir des étiquettes de classe : appliquer fonction signe ou fonction de Heaviside

SVM : séparation non linéaire

- Que faire si les données ne sont (vraiment) pas linéairement séparables ?

→ Transposer les données dans un autre espace (en général de dimension supérieure) dans lequel elles deviennent (presque) linéairement séparables : $\phi : \mathbb{R}^p \to \mathcal{H}$, $\langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle = K(\mathbf{x}_i, \mathbf{x}_j)$





- Grâce à la fonction noyau K, tout calcul impliquant des produits scalaires dans l'espace d'arrivée peut être réalisé en utilisant K dans l'espace de départ

SVM : séparation non linéaire

- La résolution du problème d'optimisation *primal* dans l'espace d'arrivée peut devenir impossible ou peu pratique $\Rightarrow$ résoudre un problème quadratique *dual* dans l'espace de départ, grâce à l'emploi de la fonction noyau :

$$\begin{cases} \max_\alpha \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i,j=1}^N \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \\ 0 \leq \alpha_i \leq C, 1 \leq i \leq N \\ \sum_{i=1}^N \alpha_i y_i = 0 \end{cases} \tag{3}$$

- Seuls les vecteurs de support ont des coefficients α_i non nuls
- La fonction de décision appliquée à une donnée $\mathbf{x}$ écrite en utilisant le noyau est :

$$f^*(\mathbf{x}) = \sum_{i=1}^{v} \alpha_i^* y_i K(\mathbf{x}, \mathbf{x}_i) + b^* \tag{4}$$

où $\mathbf{x}_i$ sont les vecteurs de support et v leur nombre ; la fonction de décision dépend donc exclusivement des vecteurs de support

SVM : fonctions noyau

- Noyau valide = satisfait la condition de Mercer : $\mathcal{X}$ compact dans $\mathbb{R}^p$ et $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, symétrique, t.q. $\forall f \in L_2(\mathcal{X})$, $\int_{\mathcal{X}^2} K(\mathbf{x}, \mathbf{y}) f(\mathbf{x}) f(\mathbf{y}) d\mathbf{x} d\mathbf{y} \geq 0$
- Exemples de fonctions noyau pour des données de nature très diverse :
  - Pour données vectorielles : noyau linéaire, noyau gaussien ou RBF, noyau angulaire...
  - Pour ensembles : à partir de l'indice de Jaccard
  - Pour arbres : par ex. comptage de sous-arbres isomorphes
- Pour certains noyaux, la fonction ϕ peut être explicitée et les calculs peuvent être facilement réalisés dans l'espace d'arrivée, il est donc possible de résoudre directement le problème *primal*
- Possible de construire des noyaux hybrides pour données hybrides (décrites par plusieurs variables de nature différente)
- Quelques détails supplémentaires dans `https://cedric.cnam.fr/vertigo/Cours/ml2/Bloc_svm/session5.html` et suite (SVM pour discrimination, régression, estimation du support d'une distribution ; analyse factorielle à noyaux ; ingénierie des noyaux)

SVM et apprentissage à large échelle

- Problèmes linéaires : SVM linéaires, résolution du problème d'optimisation *primal* avec par ex. la descente de (sous-)gradient stochastique, facilement parallélisable
- Problèmes non linéaires (classes vraiment non linéairement séparables dans l'espace de départ) : SVM à noyaux non linéaires
  - Noyaux permettant de déterminer une représentation vectorielle explicite et de dimension finie dans l'espace d'arrivée : résolution directe du problème d'optimisation primal
  - Autres noyaux (par ex. RBF) : solveurs quadratiques peu parallélisables $\rightarrow$ pour passer à l'échelle, différentes pistes (voir aussi [1]) :
    1. Approximation du noyau ($\rightarrow$ résultats différents de ceux obtenus avec le noyau initial)
    2. Méthodes de réduction de la complexité pour noyaux à décroissance rapide (par ex. RBF, voir cours sur la réduction de l'ordre de complexité)

SVM et apprentissage à large échelle (2)

- Autres solutions pour réduire le coût de l'apprentissage :
  1. Apprentissage incrémental, par ex. :
     a. apprentissage initial sur un échantillon,
     b. conservation des vecteurs de support,
     c. ajout d'un nouvel échantillon,
     d. réapprentissage sur $\{$vecteurs support$\} \cup \{$nouvel échantillon$\}$, reprise à partir de b.
  2. Apprentissage actif : apprentissage initial sur un échantillon, ajout progressif des données qui peuvent améliorer le plus le modèle (par ex. les plus proches de la frontière)

- Réduire le coût de la décision avec SVM (si nombre élevé de vecteurs support) :
  1. Approximation de la solution avec (nettement) moins de vecteurs support
  2. Méthodes de réduction de la complexité pour noyaux à décroissance rapide (voir cours sur la réduction de l'ordre de complexité)

SVM linéaires et descente de gradient stochastique

- SVM linéaires, N très élevé $\rightarrow$ résoudre le problème d'optimisation *primal* :

$$\min_{\mathbf{w},b} \left(\frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N} L_H(\mathbf{x}_i, y_i; \mathbf{w}, b) \right) \tag{5}$$

  où $L_H(\mathbf{x}_i, y_i; \mathbf{w}, b) = \max\{0, 1 - y_i\left(\mathbf{w}^T\mathbf{x}_i + b\right)\}$ est la *hinge loss*
- Inclure le seuil b comme composante supplémentaire de $\mathbf{w}$: ajouter une composante supplémentaire correspondante, toujours égale à 1, à l'entrée $\mathbf{x}$
- Le problème d'optimisation devient $\min_{\mathbf{w}} g(\mathbf{w})$, où

$$g(\mathbf{w}) = \left(\frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N} L_H(\mathbf{x}_i, y_i; \mathbf{w}) \right) \tag{6}$$

  avec $L_H(\mathbf{x}_i, y_i; \mathbf{w}) = \max\{0, 1 - y_i(\mathbf{w}^T\mathbf{x}_i)\}$
- Pour minimiser $I(\mathbf{w})$ différentiable, une solution simple est la descente de gradient :

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta\frac{\partial}{\partial\mathbf{w}} I, \text{ avec } \eta > 0 \tag{7}$$

SVM linéaires et descente de gradient stochastique (2)

- *Hinge loss* dérivable presque partout $\Rightarrow$ descente de sous-gradient pour g
- Calcul du (sous-)gradient :
  - Minimisation de l'inverse de la marge :

$$\frac{\partial}{\partial \mathbf{w}} \frac{1}{2} \|\mathbf{w}\|^2 = \mathbf{w} \tag{8}$$

  - Minimisation de l'erreur d'apprentissage :

$$\frac{\partial}{\partial \mathbf{w}} L_H(\mathbf{x}_i, y_i; \mathbf{w}) \leftarrow \begin{cases} -y_i \mathbf{x}_i & \text{si } y_i \mathbf{w}^T \mathbf{x}_i < 1 \\ 0 & \text{sinon} \end{cases} \tag{9}$$

- La composante $\sum_{i=1}^{N} L_H(\mathbf{x}_i, y_i; \mathbf{w})$ étant additive par rapport aux données $(\mathbf{x}_i, y_i)$, une version stochastique peut être définie :
  - A chaque itération t un i est choisi, $\mathbf{w}_{t+1}$ est mis à jour sur la base de la seule contribution de $\frac{\partial}{\partial \mathbf{w}} L_H(\mathbf{x}_i, y_i; \mathbf{w}_t)$
  - *Mini-batch* : un échantillon (taux $0 < s \ll 1$) choisi à chaque itération plutôt qu'une seule donnée
  - Avantage de la version stochastique : coût de calcul plus faible car dans chaque itération une faible partie des données est utilisée

Descente de gradient stochastique : une implémentation MapReduce

Data : N données $(\mathbf{x}_i, y_i) \in \mathbb{R}^p \times \{-1; 1\}$, taux d'échantillonnage s, pas initial η
Result : Vecteur $\mathbf{w}$ et seuil b

1 L'ensemble $\mathcal{E}$ de N données est découpé en fragments (partitions) distribués aux nœuds de calcul ; un fragment doit tenir dans la mémoire d'un nœud ;
2 Choisir une solution initiale pour $\mathbf{w}_0$ et b_0 ;
3 **while** *nombre maximal d'itérations non atteint* **do**
4 Transmettre la solution courante $\mathbf{w}_t$ et b_t à tous les nœuds de calcul ;
5 Chaque tâche *Map* choisit un échantillon de données $(\mathbf{x}_i, y_i)$ de son fragment, en respectant le taux d'échantillonnage s ;
6 Chaque tâche *Map* calcule la contribution de cet échantillon à $\mathbf{w}_{t+1}$ et b_{t+1} ;
7 Les résultats des tâches *Map* sont transmis aux tâches *Reduce* qui calculent $\mathbf{w}_{t+1}$ et b_{t+1} en ajoutant également la contribution du terme de régularisation ;
8 **end**

Descente de gradient stochastique : une implémentation MapReduce (2)

- Dans l'étape 7 (*Reduce*), le partitionnement peut être fait, par exemple, par groupe de dimensions de $\mathbb{R}^p$
- Échantillonnage : le nombre d'opérations est proportionnel au nombre de données de la partition affectée à un nœud de calcul
- D'autres variantes d'exécution distribuée existent

Évaluation pour problèmes à 2 classes : courbes ROC

- Discrimination entre deux classes
  - $\Rightarrow$ une classe peut être considérée la « classe d'intérêt »
  - $\Rightarrow$ le modèle appris est vu comme le « détecteur » de la classe d'intérêt
    - Un modèle est en général décrit par un vecteur de paramètres (par ex. $\mathbf{w}$ pour SVM linéaire) et un seuil de détection (b pour SVM linéaire)
- Pour un détecteur appris, les cas suivants peuvent être constatés :

	Positifs ($\in$ classe intérêt)	**Négatifs ($\notin$ classe intérêt)**
Détectés positifs	Vrais Positifs	Faux Positifs
Détectés négatifs	Faux Négatifs	Vrais Négatifs

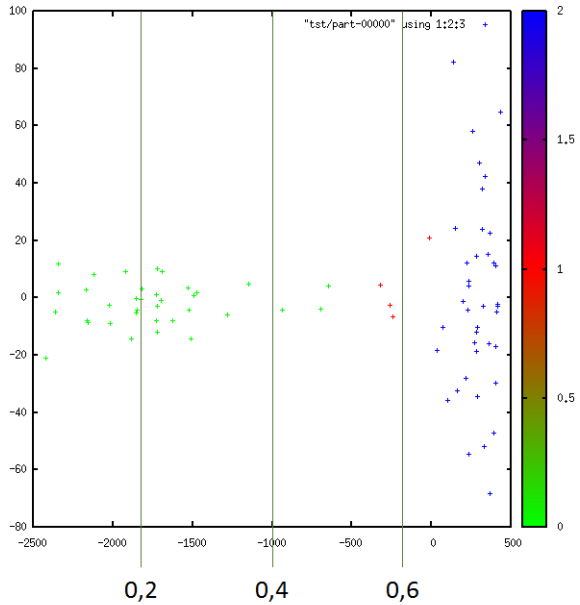
- On définit les mesures suivantes :

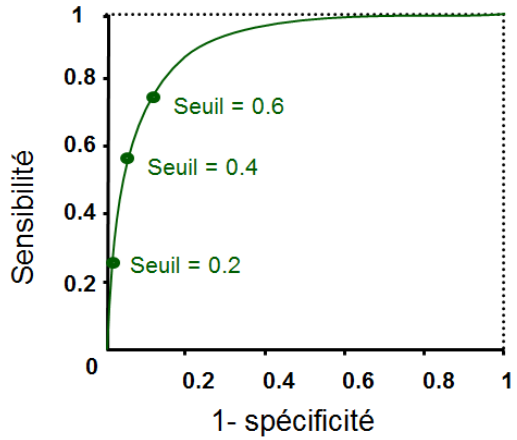
  Sensibilité (ou taux de vrais positifs) $= \frac{\text{Vrais Positifs}}{\text{Total Positifs}} = \frac{\text{VP}}{\text{VP + FN}}$

  $1-$ Spécificité (ou taux de faux positifs) $= \frac{\text{Faux Positifs}}{\text{Total Négatifs}} = \frac{\text{FP}}{\text{VN + FP}} = 1 - \frac{\text{VN}}{\text{VN + FP}}$
- Idéalement
  - Toutes les détections positives devraient correspondre à de vrais positifs : pas de faux négatifs (FN $= 0$), ou sensibilité $= 1$
  - Ce qui n'est pas détecté devrait correspondre aux seuls vrais négatifs : pas de faux positifs (FP $= 0$), ou $1-$ spécificité $= 0$

Courbes ROC : définition

- Pour un même vecteur de paramètres (par ex. $\mathbf{w}$), peut-on réduire en même temps FN et FP en faisant varier le seuil de détection ?
- Dans la figure de gauche (données de $\mathbb{R}^2$, classe d'intérêt en vert), considérons $\mathbf{w}$ fixé tel que la frontière de détection soit une droite verticale





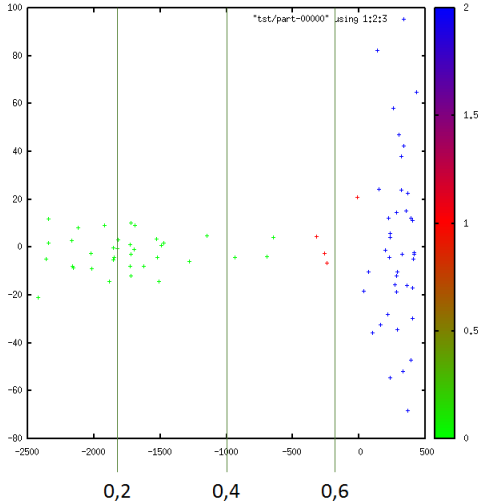
Courbe ROC

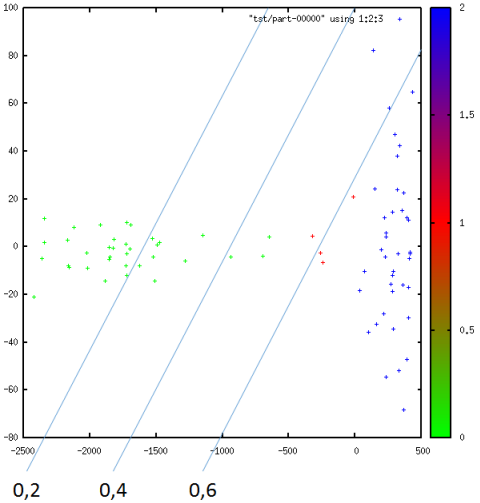
- Évolution des deux mesures lorsqu'on déplace cette frontière à gauche ou à droite ?

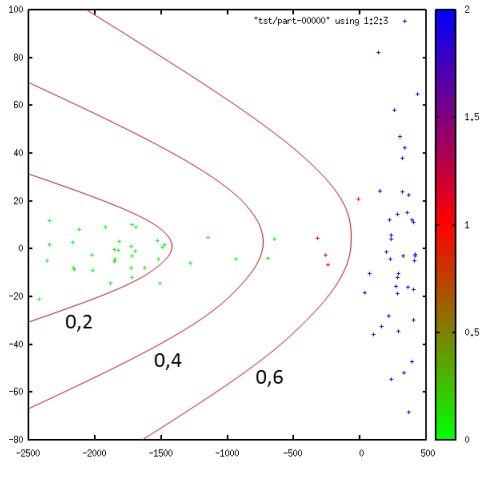
$\Rightarrow$ lorsqu'on augmente la sensibilité, (1 - spécificité) augmente également

Courbes ROC : comparaison entre modèles

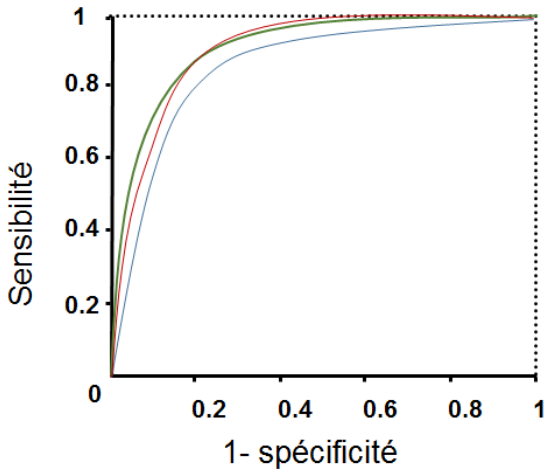
- Par exemple, frontières verticales, ou inclinées, ou non linéaires :







  - En général : plus l'aire sous la courbe (AUC) ROC est élevée, meilleur est le modèle
  - Intersection entre courbes ROC : comparaison des sensibilités à spécificité donnée, etc.

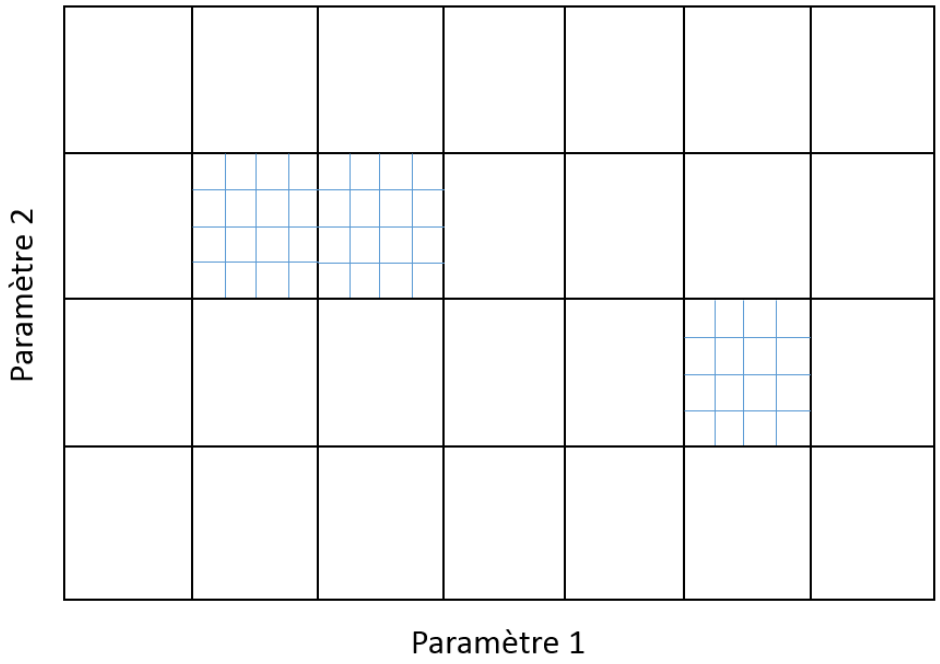


Grid search pour le choix des hyperparamètres

- La recherche du modèle qui minimise le risque empirique régularisé dépend de « hyperparamètres » qui délimitent la famille de fonctions candidates $\mathcal{F}$, ainsi que le degré de régularisation (et éventuellement le type, par ex. L_1 ou L_2) ; exemples :
  - SVM à noyau RBF : variance du noyau RBF, respectivement valeur de C (et éventuellement type de régularisation)
  - SVM linéaires : valeur de C (et éventuellement type de régularisation)
- L'algorithme d'optimisation des paramètres (par ex. descente de gradient stochastique) opère une fois fixées les valeurs des hyperparamètres
- Pour trouver les meilleures valeurs des hyperparamètres
  1. Définition d'intervalles de variation pour les hyperparamètres et d'une procédure d'exploration de cet espace
  2. Exploration de l'espace des hyperparamètres
  3. Choix des valeurs pour lesquelles le modèle obtenu (après application de l'algorithme d'optimisation des paramètres) est meilleur (tel qu'indiqué, par ex., par les résultats de la validation croisée)
- Estimation du risque espéré (erreur de généralisation) du modèle obtenu : sur d'autres données que celles employées pour le choix des hyperparamètres !

Grid search pour le choix des hyperparamètres (2)

- Une solution simple est la recherche dans une grille : ensemble ($\leftarrow$ intervalle et pas de variation) de valeurs à tester pour chacun des m paramètres $\Rightarrow$ grille de dimension m



- Tous les points de la grille peuvent être explorés en parallèle !
- La grille peut avoir plusieurs niveaux de « finesse » et la recherche sera dans ce cas hiérarchique : recherche exhaustive suivant la grille grossière, et là où les résultats sont meilleurs on affine suivant le(s) niveau(x) plus fin(s)

Spark : validation croisée et *grid search*

- *Spark ML(lib) DataFrame* : interface de programmation (API) de niveau supérieur à MLlib RDD, afin de permettre de définir et évaluer des *pipelines* (chaînes de traitement) de plusieurs outils de modélisation et apprentissage statistique
- Notions importantes :
  - *DataFrame* : structure de données distribuée organisée en colonnes nommées, optimisée par rapport aux `RDD` (format binaire compact pour les données, emploi d'optimiseur)
  - *Transformer* : méthode qui transforme un `DataFrame` en un autre `DataFrame` (par ex., un modèle qui fait des prédictions à partir de données d'entrée)
  - *Estimator* : algorithme qui produit un *Transformer* à partir d'un `DataFrame` (par ex., un modèle qui est obtenu par apprentissage sur la `DataFrame`)
  - *Pipeline* : chaîne de traitement composée d'une succession de *Estimator* / *Transformer*
  - *Param* : outil de spécification de (recherche de) paramètres pour *Estimator* et *Transformer*
- Utilisation de *grid search* (voir `ParamGridBuilder`) et de sélection de modèle par validation croisée (*k-fold*, voir `CrossValidator`)

Références I


L. Bottou, O. Chapelle, D. DeCoste, and J. Weston. *Large-Scale Kernel Machines (Neural Information Processing)*. The MIT Press, 2007.


O. Chapelle, B. Schölkopf, and A. Zien, editors. *Semi-Supervised Learning*. MIT Press, Cambridge, MA, 2006.


B. Schölkopf and A. Smola. *Learning with Kernels*. MIT Press, 2002.