

# Apprentissage Automatique

## Régularisation / SVM

**Stéphane Herbin**

`stephane.herbin@onera.fr`

# Rappel des cours précédents

## Généralités

- Programmation orientée données
- Démarche globale: base de données, analyse préliminaire, sélection de l'approche, optimisation, évaluation

## Apprentissage supervisé

- Plusieurs approches classiques: kNN, bayésien naïf, arbres de décision, méthodes ensemblistes

# Aujourd'hui

- Approfondissement:
  - Régularisation
  - Un algorithme efficace: Support Vector Machines (SVM)
  - Multiclasse
- TD:
  - SVM: étude de l'influence des paramètres
  - Validation croisée

# Apprentissage supervisé (rappel)

- On veut construire une fonction de décision  $F$  à partir d'exemples
- On dispose d'un **ensemble d'apprentissage**  $\mathcal{L}$  sous la forme de paires  $\{x_i, y_i\}$  où  $x_i$  est la donnée à classer et  $y_i$  est la classe vraie:

$$\mathbf{D} = \{(\mathbf{x}_i, y_i)\}_{i=1\dots n}$$

- L'apprentissage consiste à identifier cette fonction de classification dans un certain espace **paramétrique**  $W$  optimisant un certain **critère**  $L$ :

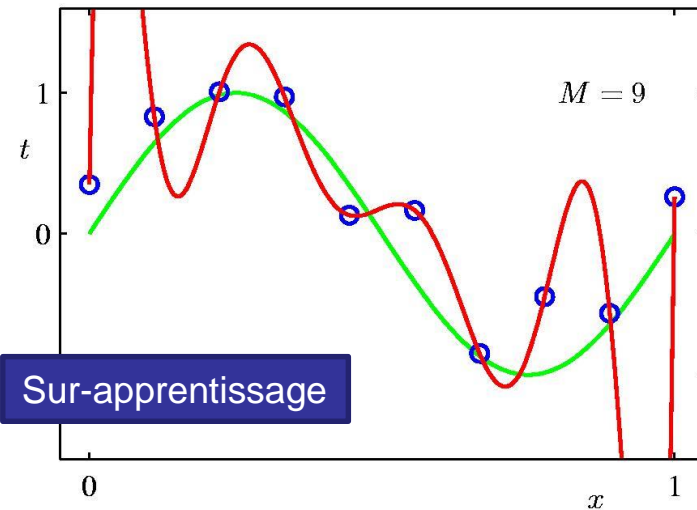
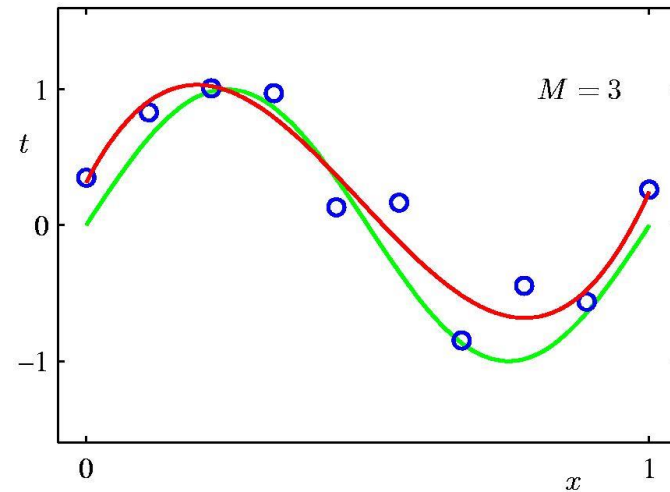
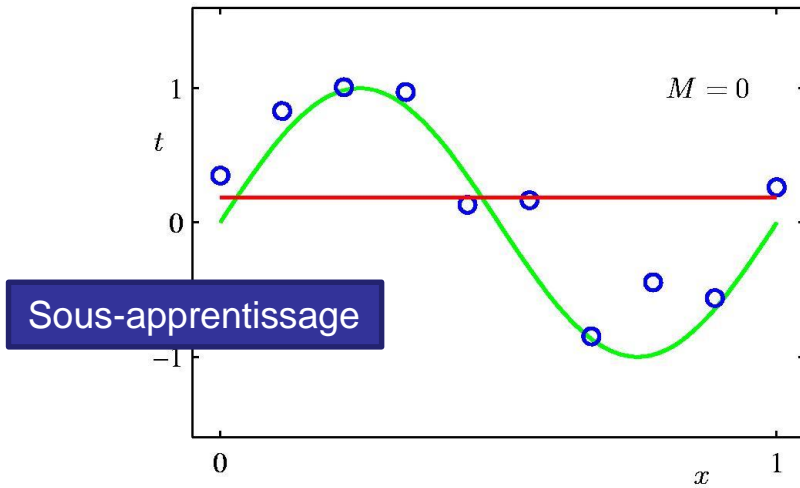
$$\mathbf{W} = \arg \min_{\mathbf{W}'} L(\mathbf{D}, \mathbf{W}')$$

- On l'applique ensuite à de nouvelles données.

$$y = F(\mathbf{x}; \mathbf{W})$$

# Régularisation

# Retour sur le sur-apprentissage



	$M = 0$	$M = 1$	$M = 3$	$M = 9$
$w_0^*$	0.19	0.82	0.31	0.35
$w_1^*$		-1.27	7.99	232.37
$w_2^*$			-25.43	-5321.83
$w_3^*$			17.37	48568.31
$w_4^*$				-231639.30
$w_5^*$				640042.26
$w_6^*$				-1061800.52
$w_7^*$				1042400.18
$w_8^*$				-557682.99
$w_9^*$				125201.43

Coefficients des polynômes

**Très grandes valeurs!**

# Moindre carrés régularisés

Idée: on rajoute une pénalisation des grandes valeurs des paramètres à la fonction de coût:

$$L(\mathbf{W}) = \sum_{i=1}^N (F(\mathbf{x}_i, \mathbf{W}) - y_i)^2 + \lambda \|\mathbf{W}\|^2$$

Coût d'attache  
aux données



Paramètre de  
régularisation

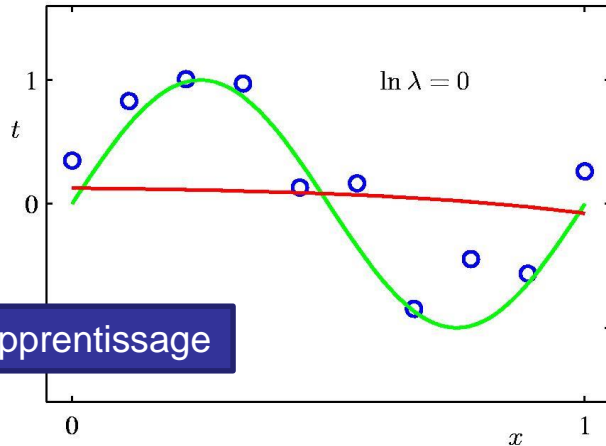


Dont l'optimum exact est alors:

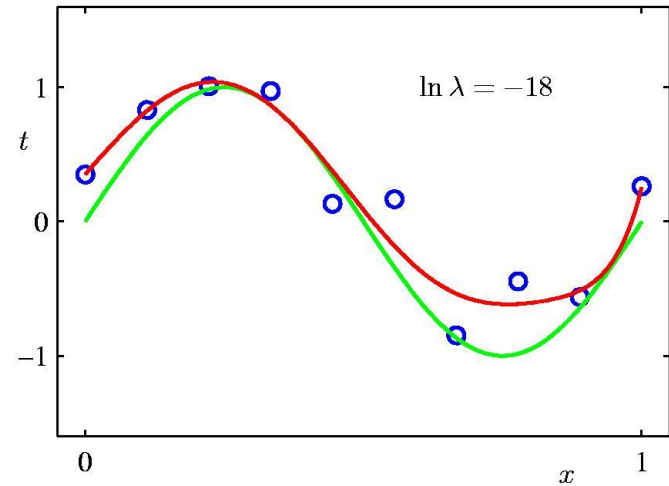
$$\mathbf{w} = \left( \lambda \mathbf{I} + \Phi^T \Phi \right)^{-1} \Phi^T \mathbf{t}.$$

Si on pénalise les grandes valeurs des coefficients du polynôme, on obtient une fonction moins « zigzagante »

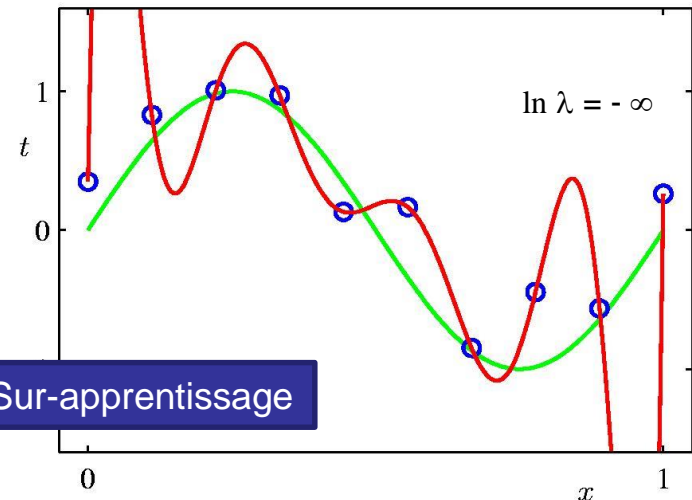
# Effet de la régularisation



Sous-apprentissage



	$\ln \lambda = -\infty$	$\ln \lambda = -18$	$\ln \lambda = 0$
$w_0^*$	0.35	0.35	0.13
$w_1^*$	232.37	4.74	-0.05
$w_2^*$	-5321.83	-0.77	-0.06
$w_3^*$	48568.31	-31.97	-0.05
$w_4^*$	-231639.30	-3.89	-0.03
$w_5^*$	640042.26	55.28	-0.02
$w_6^*$	-1061800.52	41.32	-0.01
$w_7^*$	1042400.18	-45.95	-0.00
$w_8^*$	-557682.99	-91.53	0.00
$w_9^*$	125201.43	72.68	0.01

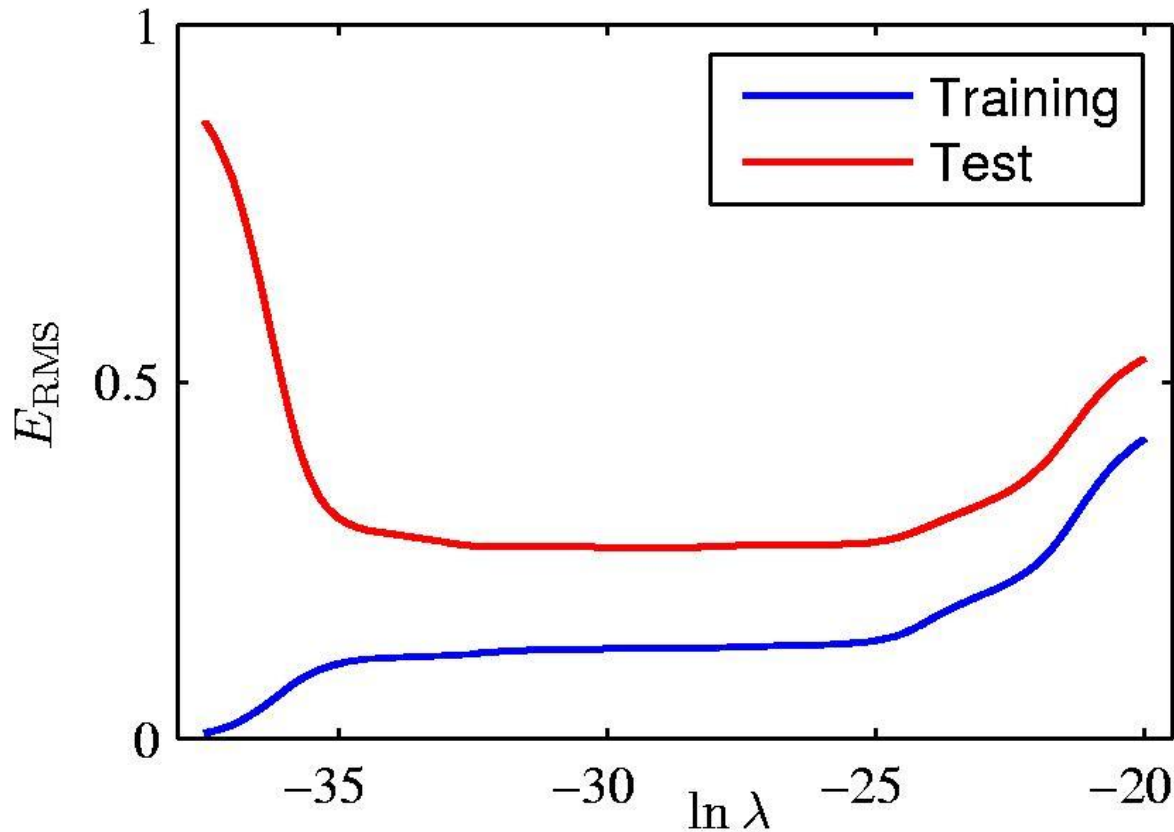


Sur-apprentissage



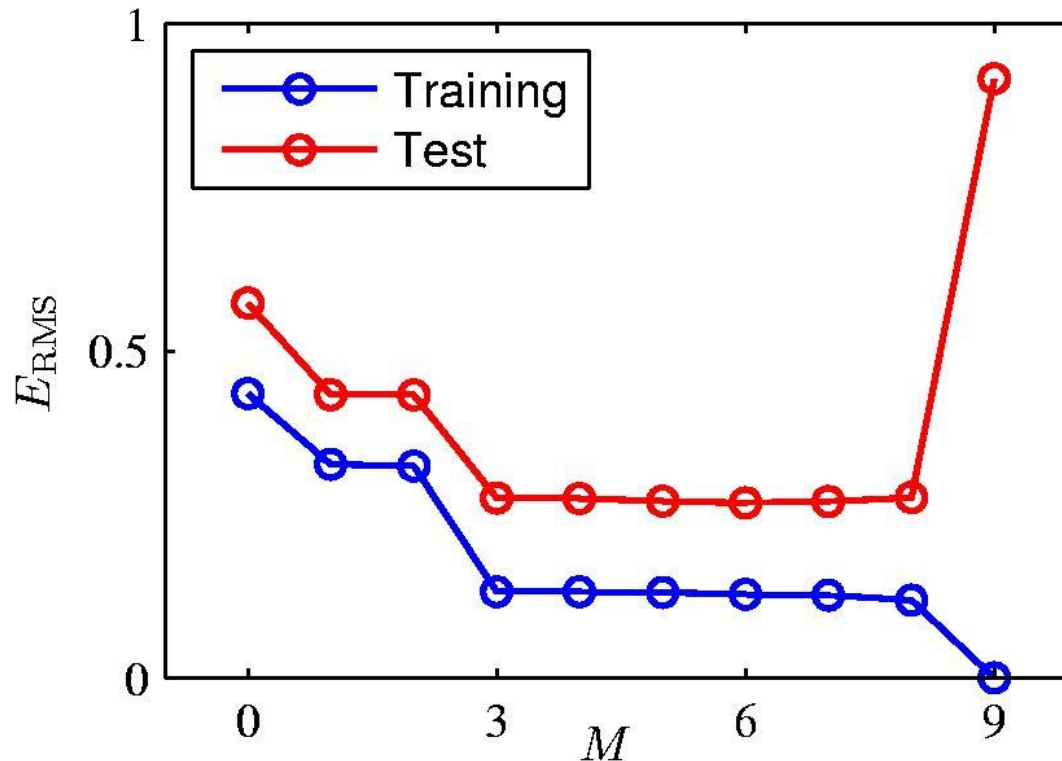
# Régularisation: $\mathcal{E}_{\text{RMS}}$ vs. $\ln(\lambda)$

$$\mathcal{E}_{\text{RMS}}(\mathbf{w}) = \frac{1}{2} \sum_{i=1}^N (F(\mathbf{x}_i, \mathbf{w}) - y_i)^2$$



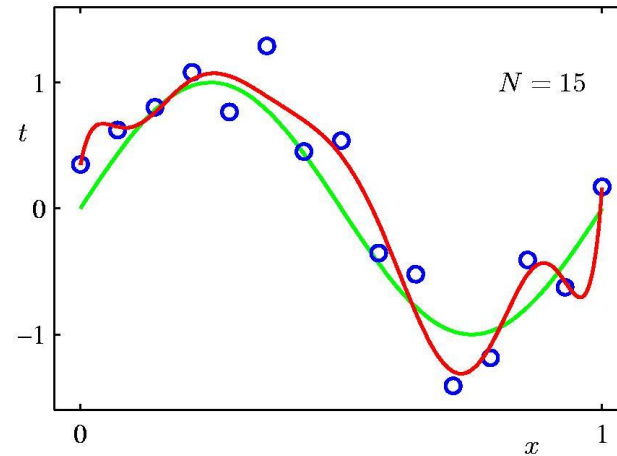
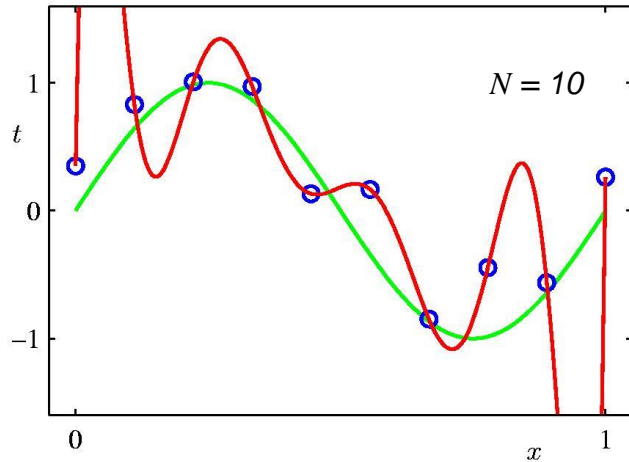
# Régularisation: $\mathcal{E}_{\text{RMS}}$ vs. $M$

$$\mathcal{E}_{\text{RMS}}(\mathbf{w}) = \frac{1}{2} \sum_{i=1}^N (F(\mathbf{x}_i, \mathbf{w}) - y_i)^2$$

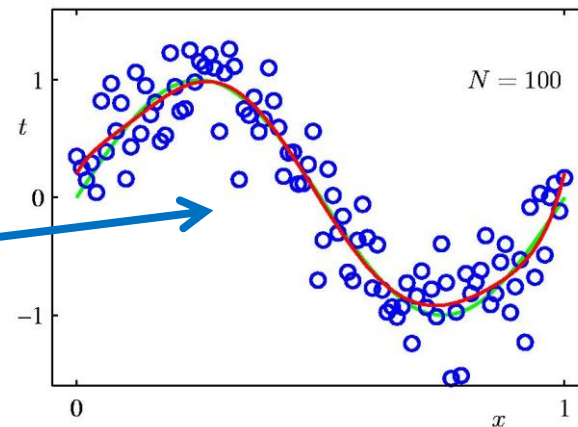


# Influence de la quantité de données

## Polynôme d'ordre 9



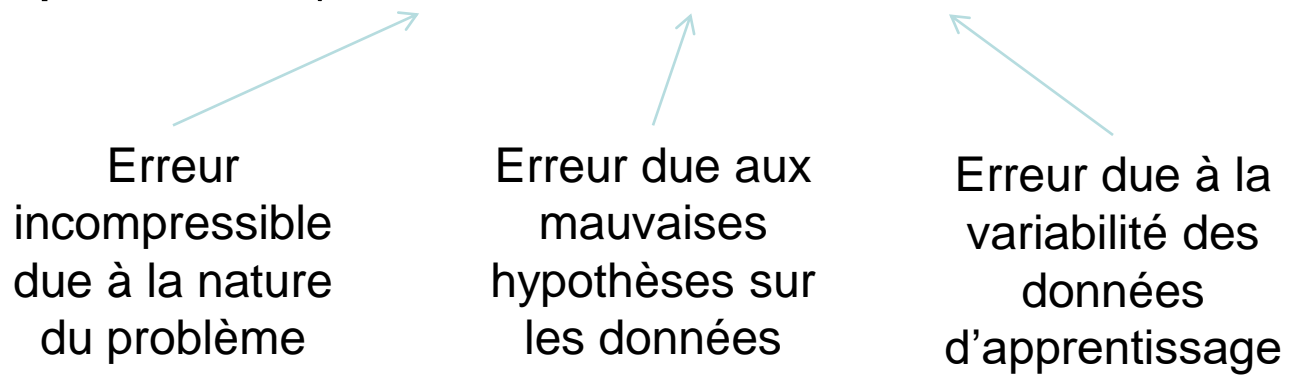
C'est aussi un moyen de  
contrôler la régression



# Compromis Biais-Variance (rappel)

On peut montrer:

$$E(\text{erreur prédiction}) = \text{bruit}^2 + \text{biais}^2 + \text{variance}$$



Erreur  
incompressible  
due à la nature  
du problème

Erreur due aux  
mauvaises  
hypothèses sur  
les données

Erreur due à la  
variabilité des  
données  
d'apprentissage

L'erreur de généralisation est un compromis entre bonnes hypothèses sur les données et qualité des données d'apprentissage

# Erreur de généralisation (rappel)

- Structure
  - **Biais:** écart entre hypothèse de modèle et « vraie » distribution des données
  - **Variance:** écarts générés par différents jeux d'apprentissage.
- Deux phénomènes à contrôler
  - **Simplisme:** modélisation trop grossière pour rendre compte de la variété des données
    - Biais++, Var –
    - Erreur d'apprentissage et de test grandes
  - **Sur-apprentissage (« Overfitting »):** modèle trop complexe se spécialisant sur les données d'apprentissage
    - Biais--, Var++
    - Ecart entre erreur d'apprentissage et erreur de test

# Trois critères à ne pas confondre

- Risque ou erreur empirique

$$\mathcal{E}_{\text{test}}(\mathbf{w}, \mathcal{D}) = \frac{1}{N} \sum_{i=1}^N \{F(\mathbf{x}_i, \mathbf{w}) \neq y_i\}$$

- Erreur de généralisation (ou idéale...)

$$\mathcal{E}(\mathbf{w}) = E_{\mathbf{x}, y} [\{F(\mathbf{x}, \mathbf{w}) \neq y\}]$$

- Critère à optimiser (forme assez générique)

$$L(\mathbf{w}, \mathcal{D}) = \frac{1}{N} \sum_{i=1}^N l(F(\mathbf{x}_i, \mathbf{w}), y_i) + r(\mathbf{w})$$

**Adéquation aux données**

**Régularisation**

# Validation croisée

# Validation croisée

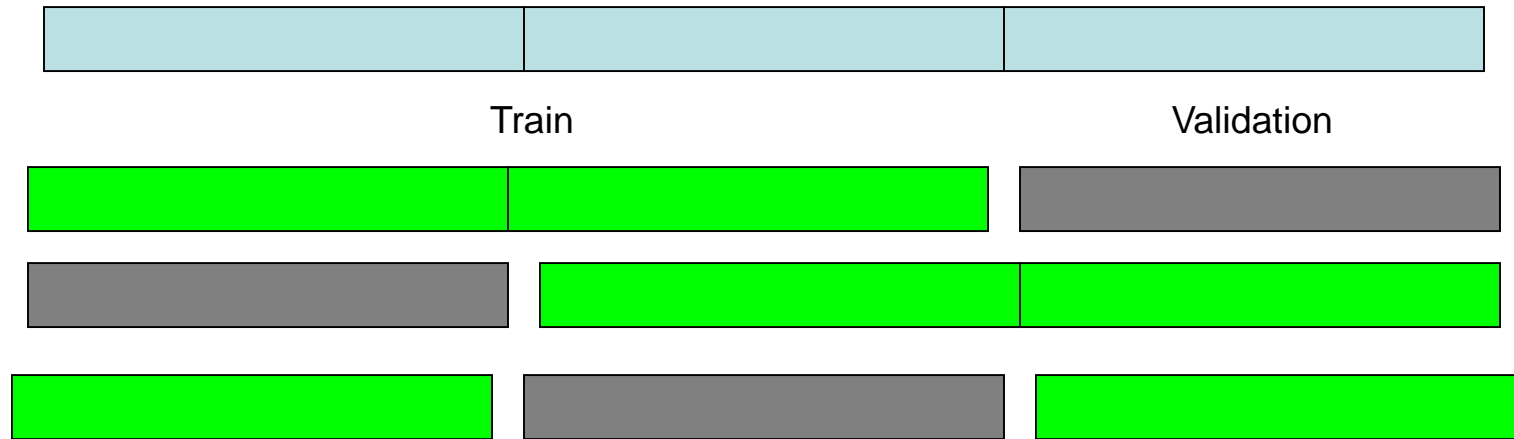
- Permet **d'estimer l'erreur de généralisation** à partir des données d'apprentissage (« astuce »)
- Principe:
  - Division des données en k sous ensembles (« fold »)
  - Choix d'une partie comme ensemble de **validation** fictif, les autres comme *train*
  - Apprentissage sur l'ensemble *train*
  - Estimation des erreurs sur *validation*
- On fait tourner l'ensemble de *validation* sur chacune des parties
- L'erreur de généralisation estimée est la **moyenne** des erreurs sur chaque ensemble de *validation*



# Stratégies de partitionnement

- k-fold

Données



- Leave-one-out



# Validation croisée: pour quoi faire?

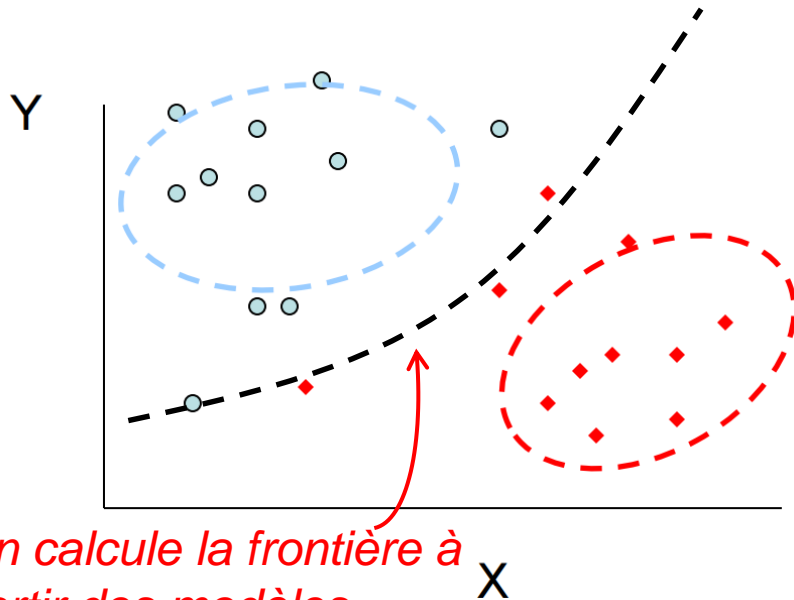
- Estimer la « vraie » erreur de prédiction (généralisation)
- Estimer la variance d'apprentissage (mais pas le biais)
- Réglage des « hyper-paramètres » (par ex. le coefficient de régularisation)
  - Recherche exhaustive ou par dichotomie (à voir en TD)
- Attention! il y a d'autres sources d'aléatoire qui ne relèvent pas de la validation croisée
  - Random forrests, Bagging
  - Initialisation et optimisation des réseaux de neurones (gradient stochastique)

# « Support Vector Machines »

# Deux types d'approches: génératives vs. discriminatives

Objectif = modéliser les distributions de données puis les exploiter

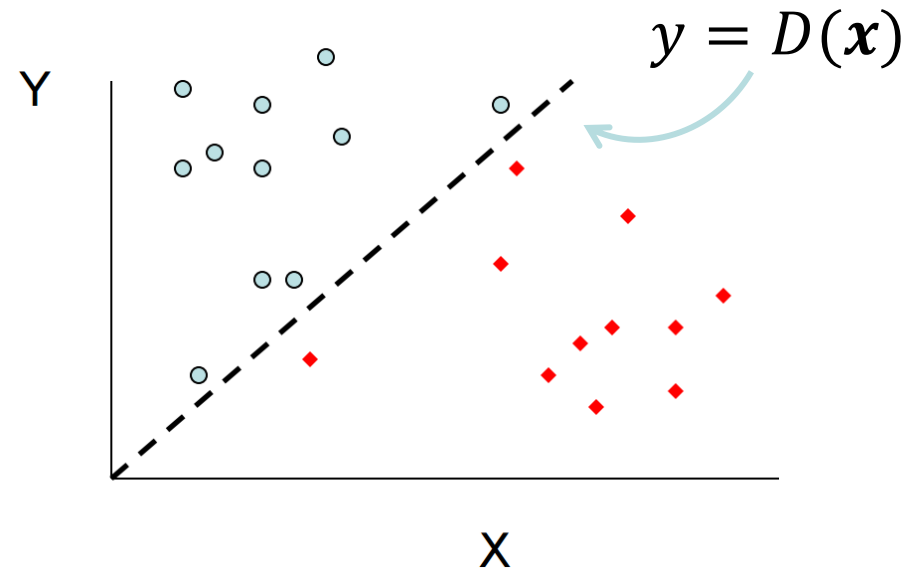
Generative model



**Le premier cours**

*On estime directement*

Discriminative model



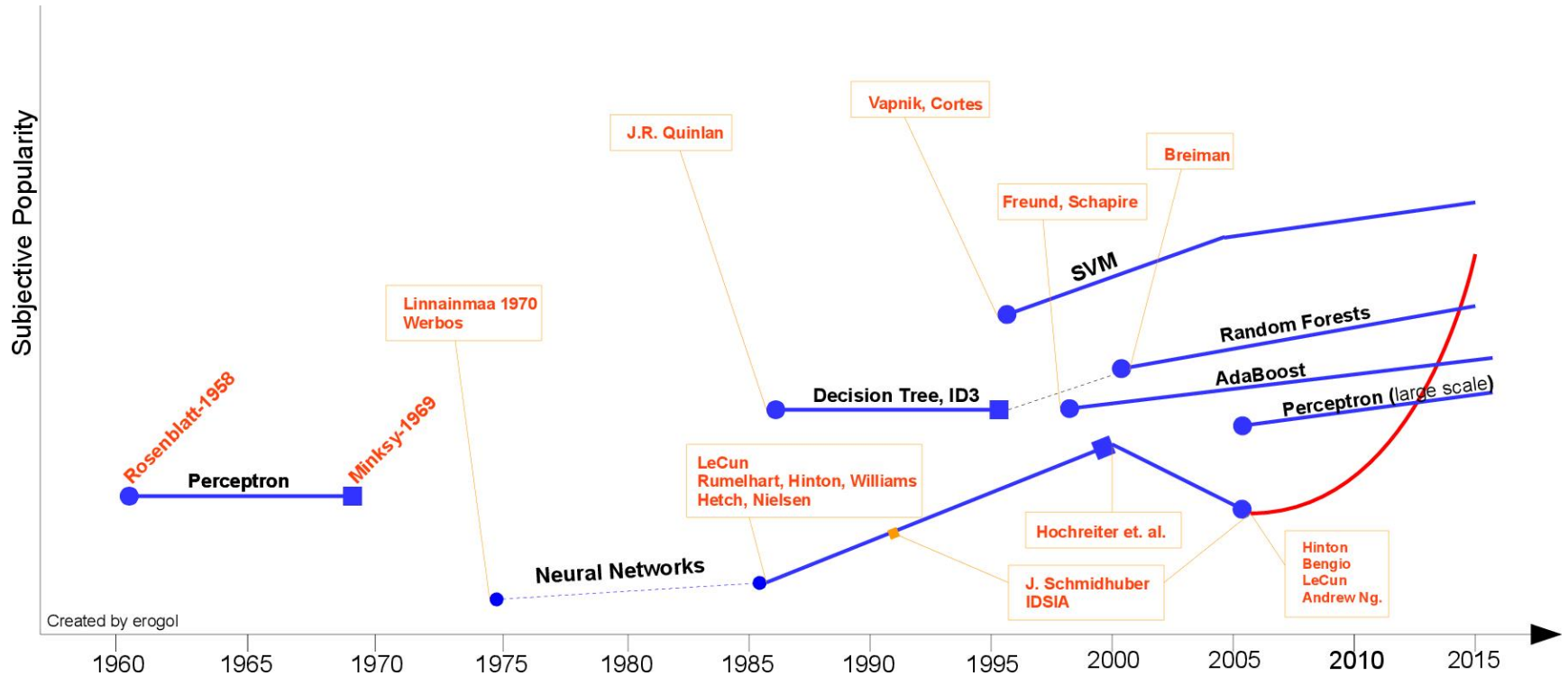
Objectif = construire les meilleures frontières

**Aujourd'hui**

# Support Vector Machines

- Historique
- Principe: maximiser la marge de séparation d'un hyperplan
- Le cas séparable
- Le cas non séparable: les fonctions de perte (« hinge loss »)
- L'extension au cas non linéaire: les noyaux
- Parcimonie
- Les paramètres de contrôle

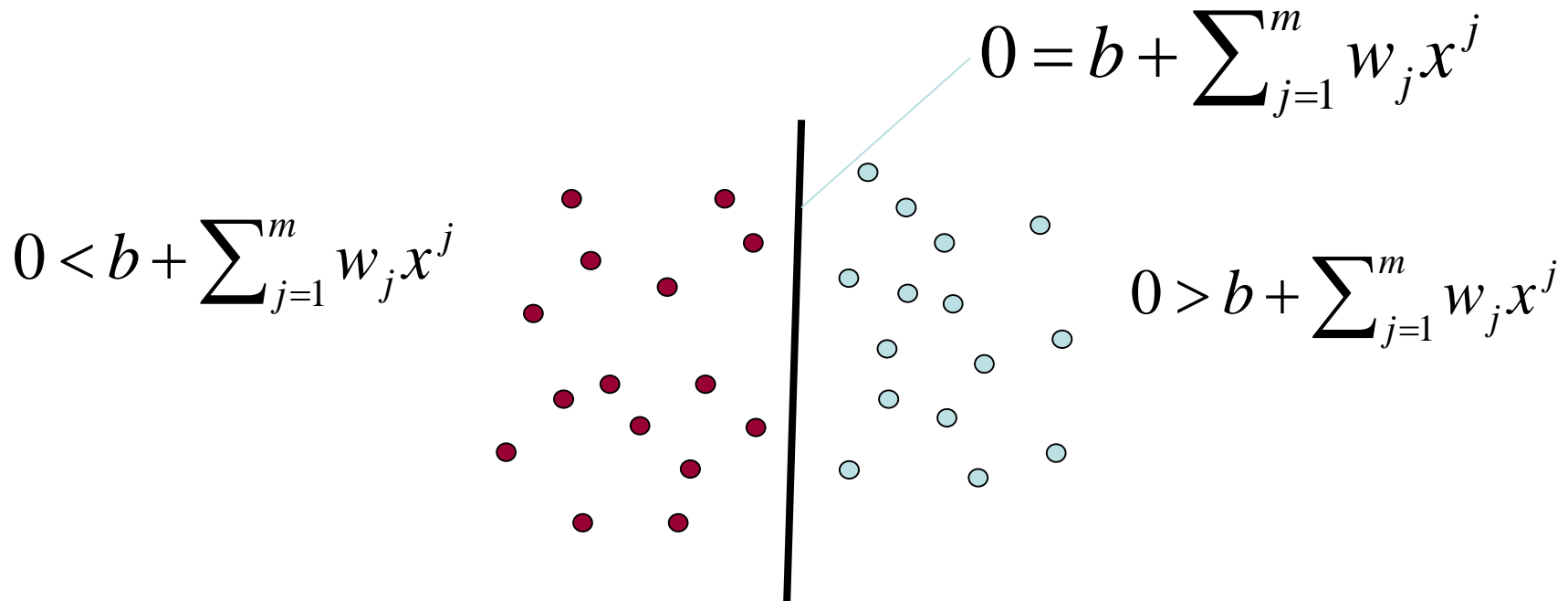
# Historique du Machine Learning



# Modèles linéaires de décision

Hypothèse = les données sont linéairement séparables.

- En 2D, par une droite
- En ND, par un hyperplan.



# Classifieur linéaire

- Equation de l'hyperplan séparateur

$$b + \mathbf{w} \cdot \mathbf{x} = 0$$

- Expression du classifieur linéaire (pour  $y_i$  valant -1 et 1)

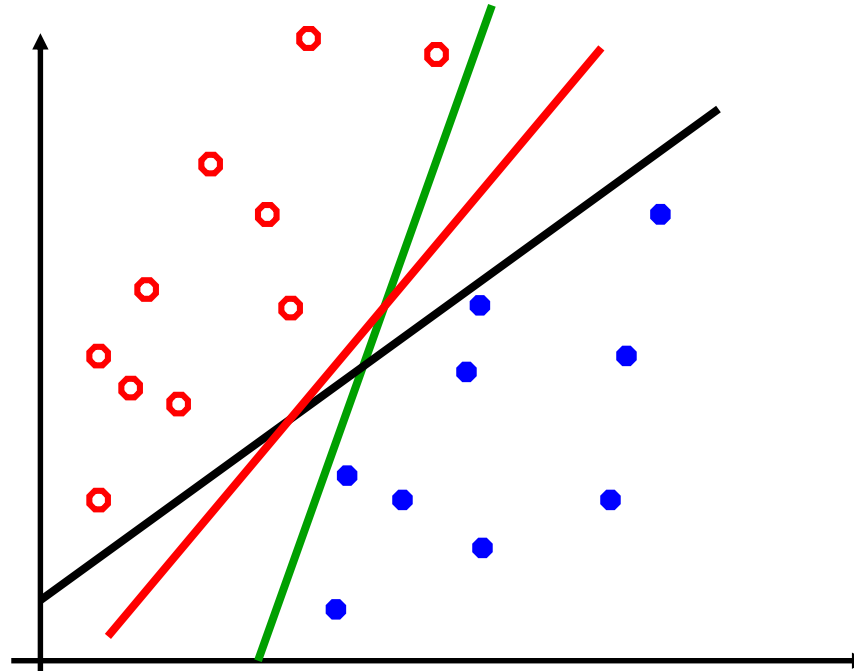
$$F(\mathbf{x}; \mathbf{w}) = \text{sign}(b + \mathbf{w} \cdot \mathbf{x})$$

- Erreur

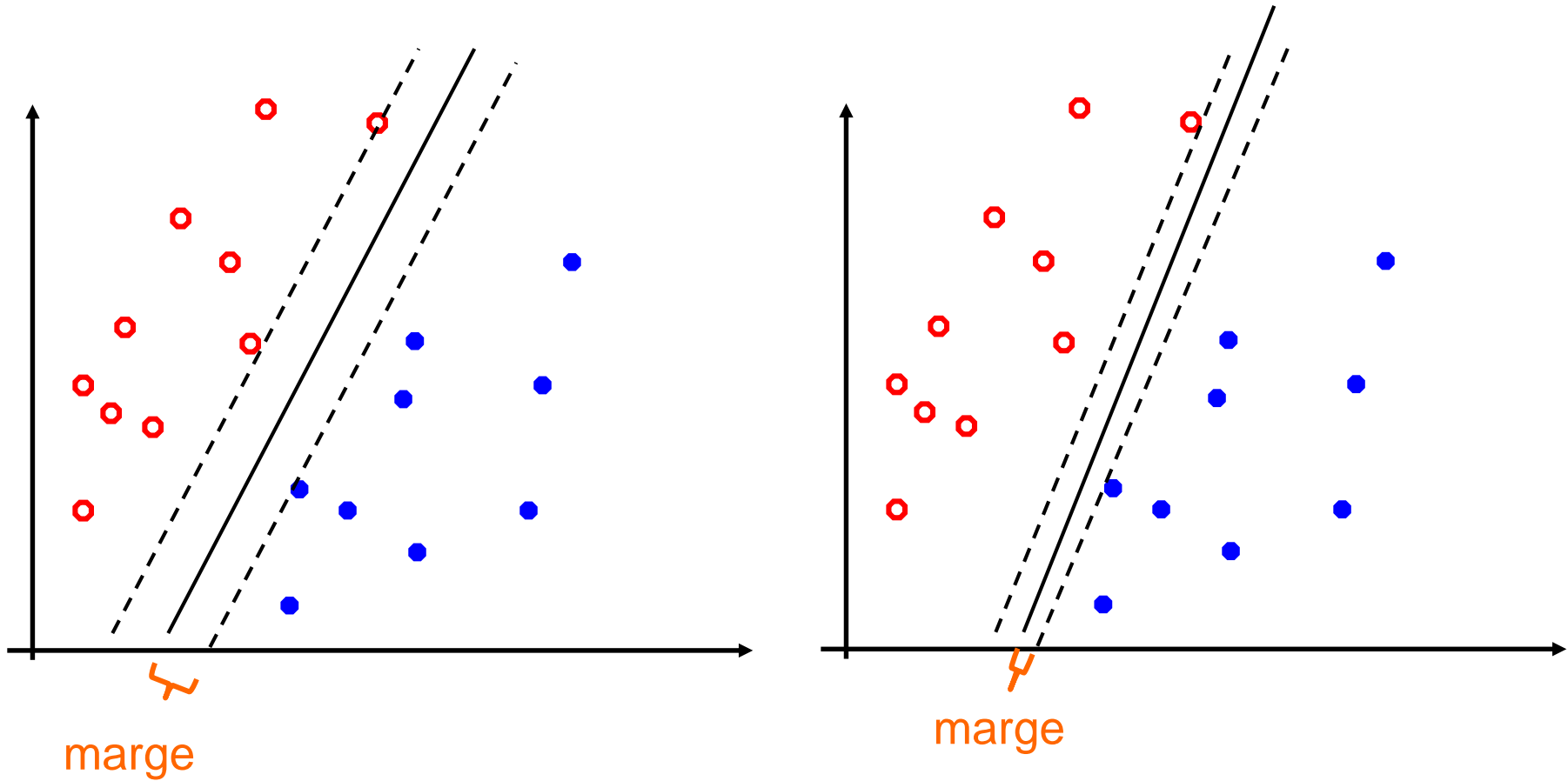
$$\mathcal{E}_{test}(\mathbf{w}, \mathcal{L}) = \frac{1}{N} \sum_{i=1}^N \{y_i \cdot \text{sign}(b + \mathbf{w} \cdot \mathbf{x}_i) < 0\}$$



# Quel hyperplan choisir?



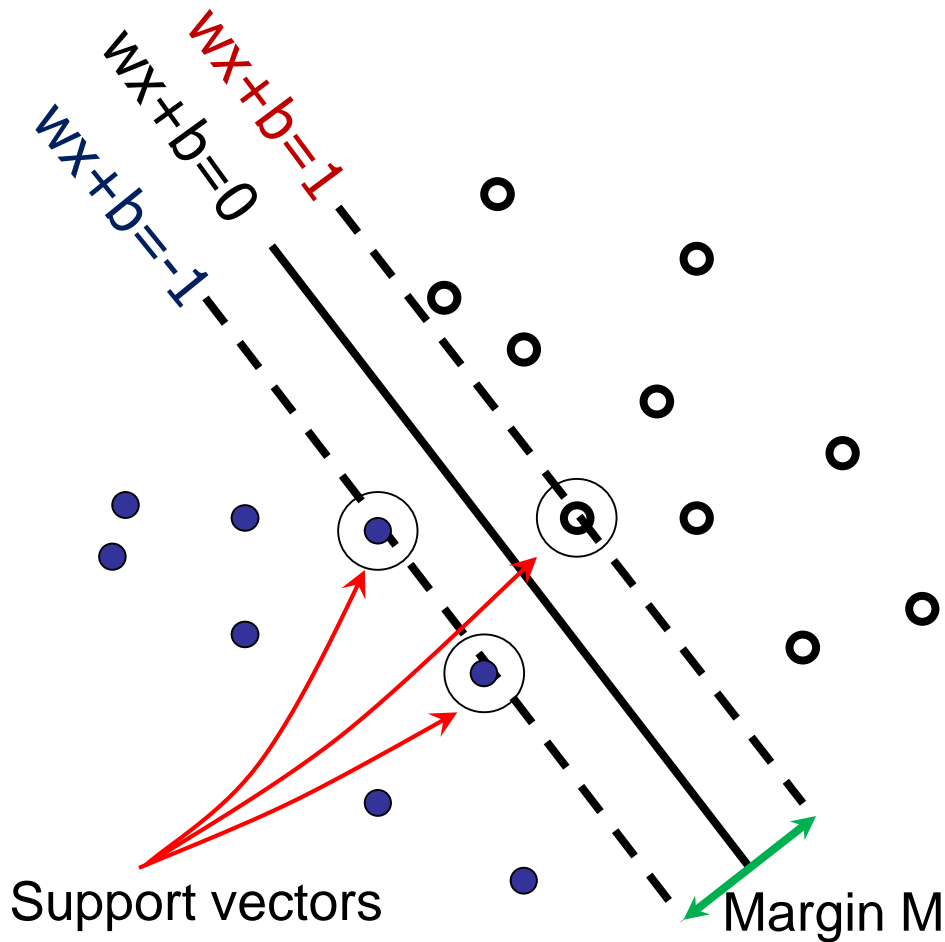
# Classifieur « Large margin »



Choisir l'hyperplan qui maximise la distance aux points les plus proches

# Support Vector Machines

- On cherche l'hyperplan qui maximise la marge.



$$\mathbf{x}_i \text{ positif } (y_i = 1): \quad \mathbf{x}_i \cdot \mathbf{w} + b \geq 1$$

$$\mathbf{x}_i \text{ négatif } (y_i = -1): \quad \mathbf{x}_i \cdot \mathbf{w} + b \leq -1$$

Pour les vecteurs de support,  $\mathbf{x}_i \cdot \mathbf{w} + b = \pm 1$

Distance entre point et hyperplan:

$$\frac{|\mathbf{x}_i \cdot \mathbf{w} + b|}{\|\mathbf{w}\|}$$

Pour les « support vectors »:

$$\frac{\mathbf{w}^T \mathbf{x} + b}{\|\mathbf{w}\|} = \frac{\pm 1}{\|\mathbf{w}\|} \quad M = \left| \frac{1}{\|\mathbf{w}\|} - \frac{-1}{\|\mathbf{w}\|} \right| = \frac{2}{\|\mathbf{w}\|}$$

# Principe du SVM (Large Margin)

- Maximiser la marge = distance des vecteurs à l'hyperplan séparateur des vecteurs de supports

$$\max \frac{1}{\|\mathbf{w}\|^2}$$

- Sous contraintes

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 \quad \forall i$$

- Les vecteurs de support vérifiant:

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) = 1$$

Le 1 est conventionnel.  
N'importe quelle  
constante >0 est valable.

# Formulation du SVM

$$\min_{w,b} \|w\|^2$$

Tel que:

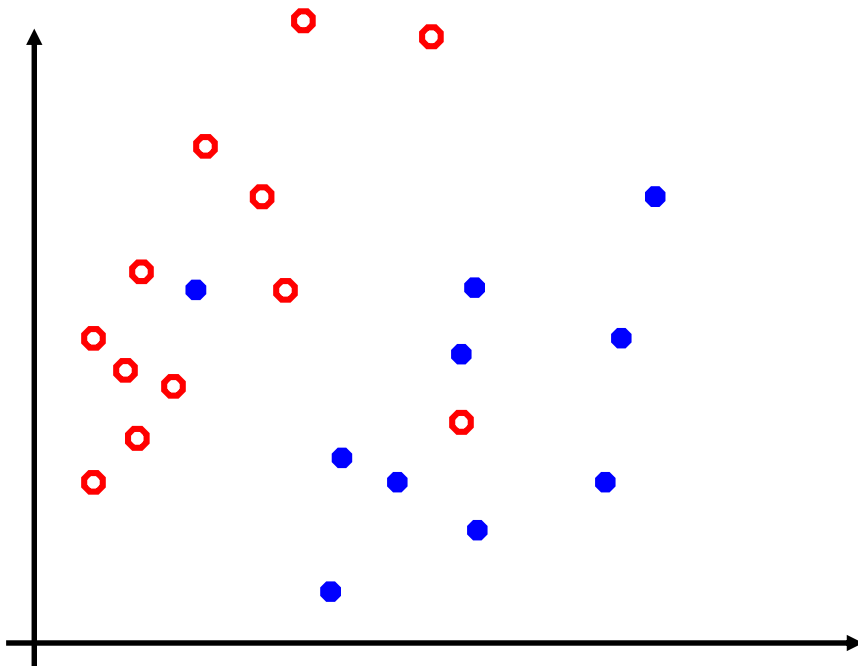
$$y_i (w \cdot x_i + b) \geq 1 \quad \forall i$$

Si les données sont séparables  
Problème d'optimisation quadratique  
Avec contraintes linéaires

**Problème d'optimisation quadratique classique**

**Mais avec beaucoup de contraintes! (autant que d'exemples d'apprentissage)**

# Classification « Soft Margin »



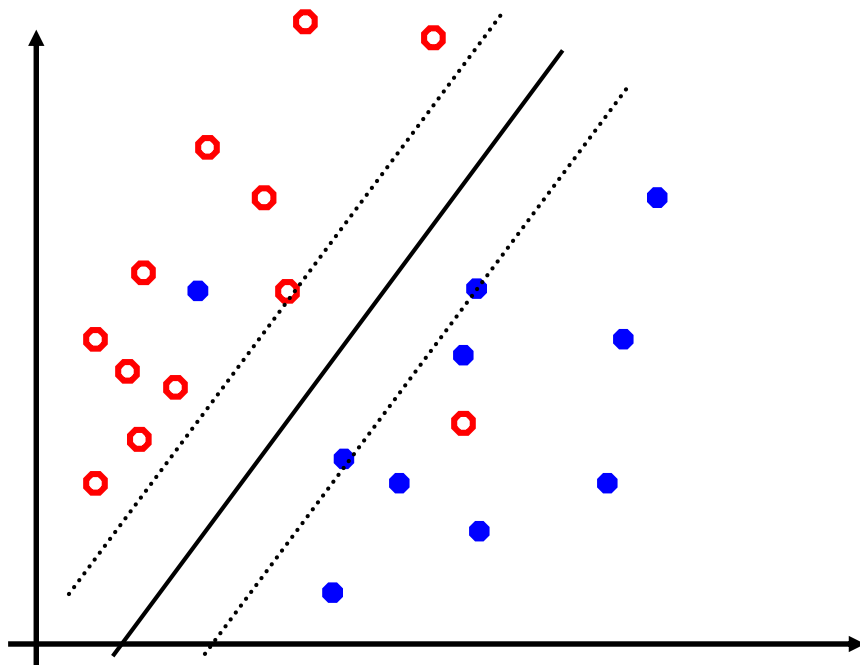
$$\min_{w,b} \|w\|^2$$

Tel que:

$$y_i (w \cdot x_i + b) \geq 1 \quad \forall i$$

**Comment traiter le cas non linéairement séparable?**

# Classification « Soft Margin »



$$\min_{w,b} \|w\|^2$$

Tel que:

$$y_i(w \cdot x_i + b) \geq 1 \quad \forall i$$

**On aimerait obtenir une séparation robuste à quelques données non séparées**

# Idée: « Slack variables »

$$\min_{w,b} \|w\|^2$$

tq:

$$y_i(w \cdot x_i + b) \geq 1 \quad \forall i$$



$$\min_{w,b} \|w\|^2 + C \sum_i \zeta_i$$

tq:

$$y_i(w \cdot x_i + b) \geq 1 - \zeta_i \quad \forall i$$

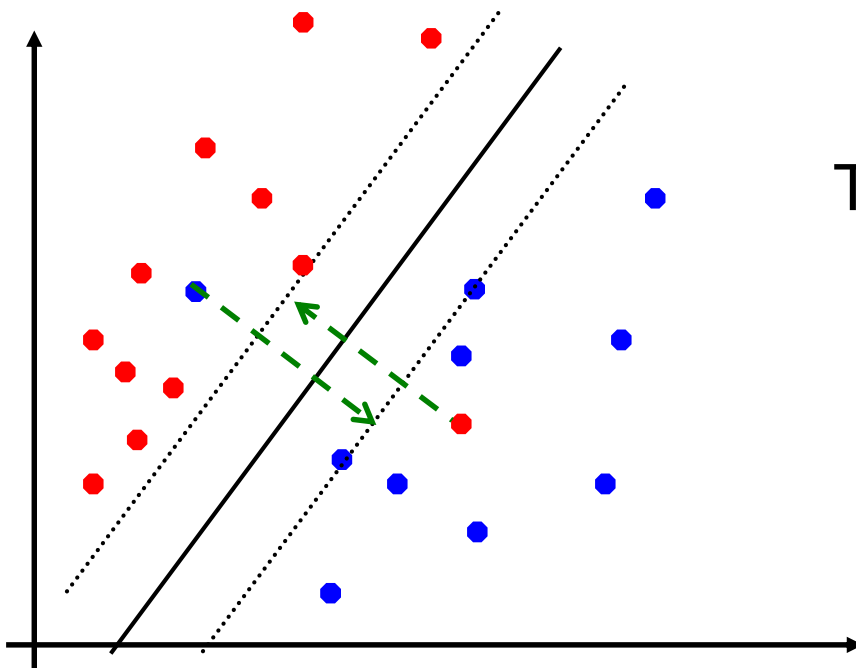
$$\zeta_i \geq 0$$

**Permet de relacher la contrainte de séparabilité pour chaque exemple.**

slack variables  
(une par exemple)



# « Slack variables »



$$\min_{w,b} \|w\|^2 + C \sum_i \zeta_i$$

Tel que:

$$y_i(w \cdot x_i + b) + \zeta_i \geq 1 \quad \forall i$$

$$\zeta_i \geq 0$$

## Relâchement de la contrainte

# Utilisation des « Slack variables »

marge

Compromis entre marge et  
pénalisation de la contrainte

$$\min_{w,b} \|w\|^2 + C \sum_i \varsigma_i$$

Valeur du  
relâchement de la  
contrainte

tq

$$y_i (w \cdot x_i + b) \geq 1 - \varsigma_i \quad \forall i$$
$$\varsigma_i \geq 0$$

Contrainte autorisée  
à être relâchée

# Soft margin SVM

$$\min_{w,b} \|w\|^2 + C \sum_i \zeta_i$$

Tel que

$$y_i (w \cdot x_i + b) \geq 1 - \zeta_i \quad \forall i$$

$$\zeta_i \geq 0$$

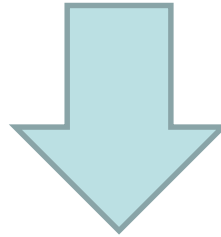
**On garde un problème quadratique!**

**Mais avec un très grand nombre de variables+contraintes**

# Autre formulation

tq:

$$\min_{w,b} \|w\|^2 + C \sum_i \varsigma_i$$
$$y_i(w \cdot x_i + b) \geq 1 - \varsigma_i \quad \forall i$$
$$\varsigma_i \geq 0$$
$$\varsigma_i = \max(0, 1 - y_i(w \cdot x_i + b))$$



$$\min_{w,b} \|w\|^2 + C \sum_i \max(0, 1 - y_i(w \cdot x_i + b))$$

**Problème d'optimisation non contraint**

**→ Autres méthodes d'optimisation (descente de gradient)**

# Interprétation du « Soft Margin SVM »

$$\min_{w,b} \|w\|^2 + C \sum_i \max(0, 1 - y_i (w \cdot x_i + b))$$

On retrouve la formulation:

$$\text{Loss}(\mathbf{w}, \mathcal{D}) = \frac{1}{N} \sum_{i=1}^N l(F(\mathbf{x}_i, \mathbf{w}), y_i) + r(\mathbf{w})$$

Avec

$$r(\mathbf{w}) = \frac{1}{C} \|\mathbf{w}\|^2$$

$$l(F(\mathbf{x}_i, \mathbf{w}), y_i) = \max(0, 1 - y_i (\mathbf{w} \cdot \mathbf{x}_i + b))$$

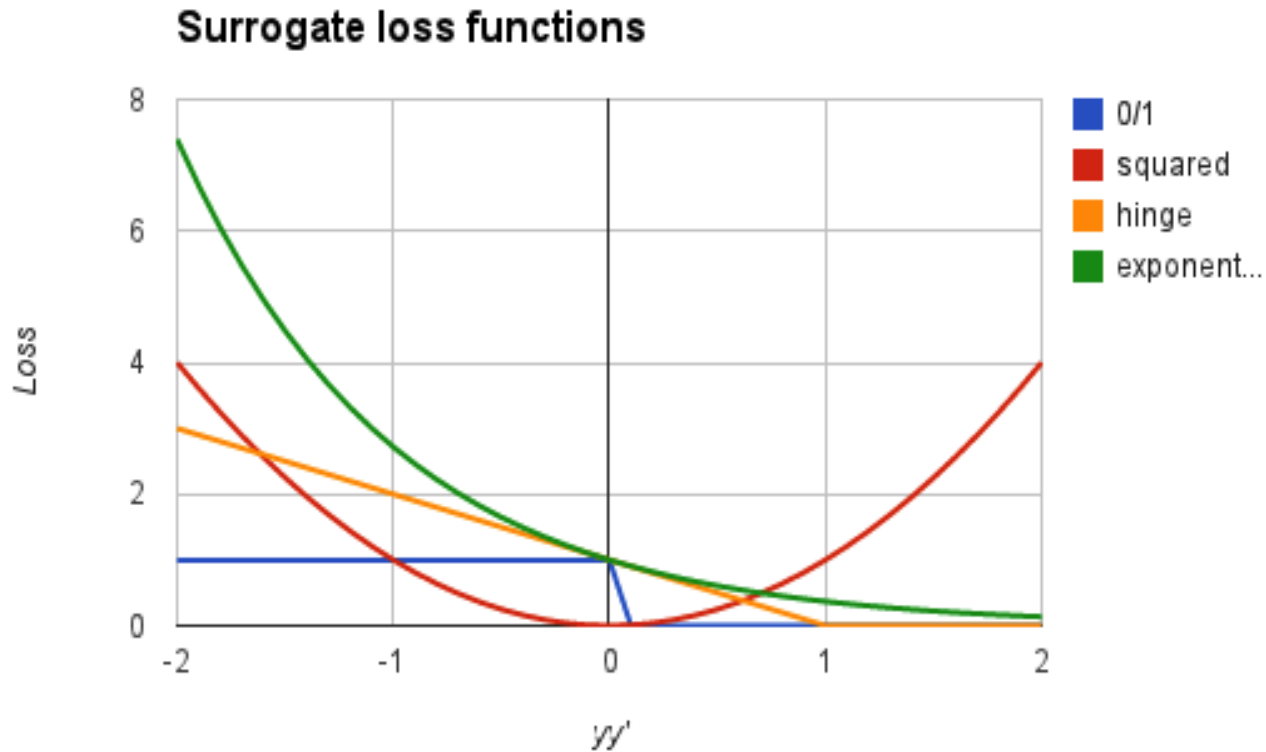
**Le SVM est un cas particulier du formalisme:  
« erreur empirique + régularisation »**

# Autres Fonctions de coût

0/1 loss:  $l(y, y') = 1[y y' \leq 0]$

Hinge:  $l(y, y') = \max(0, 1 - y y')$

Squared loss:  $l(y, y') = (y - y')^2$  Exponential:  $l(y, y') = \exp(-y y')$



# Forme duale du SVM

- Problème d'optimisation sous contrainte

*Pour simplifier l'expression des calculs*

Primal

$$\operatorname{argmin}_{\mathbf{w}} \frac{\|\mathbf{w}\|^2}{2} + C \sum_i \xi_i \quad \text{Multiplicateurs de Lagrange}$$

$$s. t. \quad \forall i, y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 - \xi_i \quad \alpha_i$$

$$\xi_i \geq 0 \quad \beta_i$$

Dual (Lagrangien)

$$L(\mathbf{w}, \xi, \alpha, \beta)$$

$$= \frac{\|\mathbf{w}\|^2}{2} + \sum_i (C\xi_i - \alpha_i(y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1 + \xi_i) - \beta_i\xi_i)$$

$$s. t. \quad \forall i, \alpha_i \geq 0, \beta_i \geq 0$$

# Forme duale du SVM

- Lagrangien

$$L(\alpha) = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_j \cdot \mathbf{x}_j$$

Maximisation dans le dual!

$$s. t. \forall i, 0 \leq \alpha_i \leq C$$

On garde un  
pb. quadratique

Dual des contraintes « slack »

Solution optimale (conditions de Kuhn-Tucker):  $\alpha_i (y_i w^T x_i - 1 + \xi_i) = 0$

Interprétation:  $\alpha_i = 0$  si la contrainte est satisfaite (bonne classification)

$\alpha_i > 0$  si la contrainte n'est pas satisfaite (mauvaise classification)

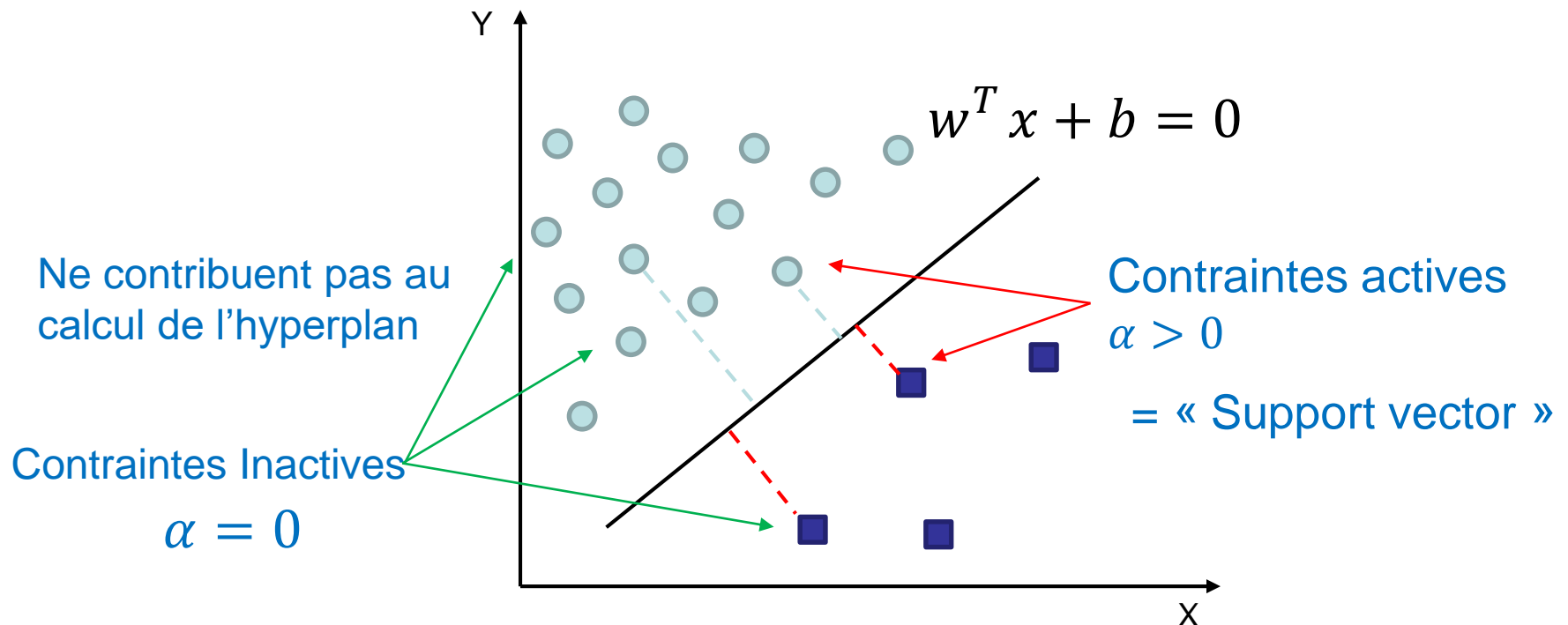


# Parcimonie du SVM

- Seuls certains  $\alpha$  sont non nuls = autre manière de définir les vecteurs de support.

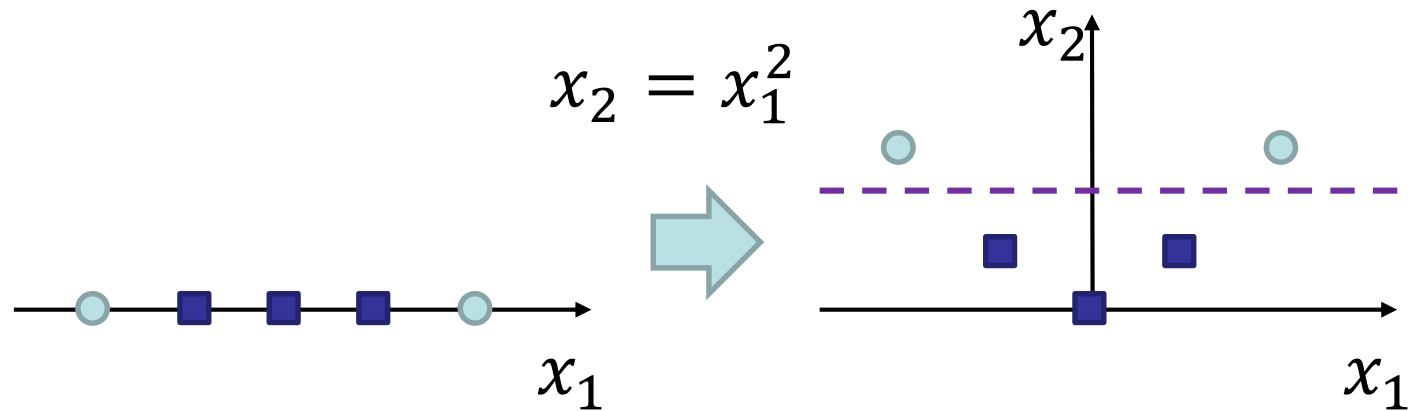
$$\text{Optimalité} = \alpha_i (y_i w^T x_i - 1 + \xi_i) = 0$$

Direction de l'hyperplan séparateur  $w = \sum_i \alpha_i y_i x_i$



# Données non linéairement séparables

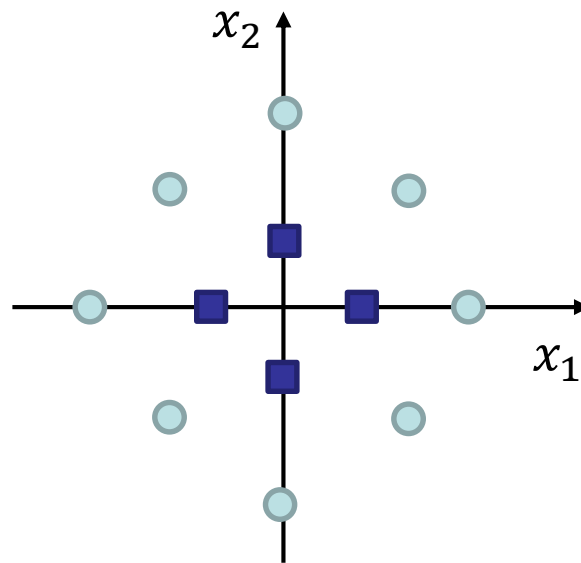
- Transformation non linéaire  $\phi(x)$  pour séparer linéairement les données d'origine



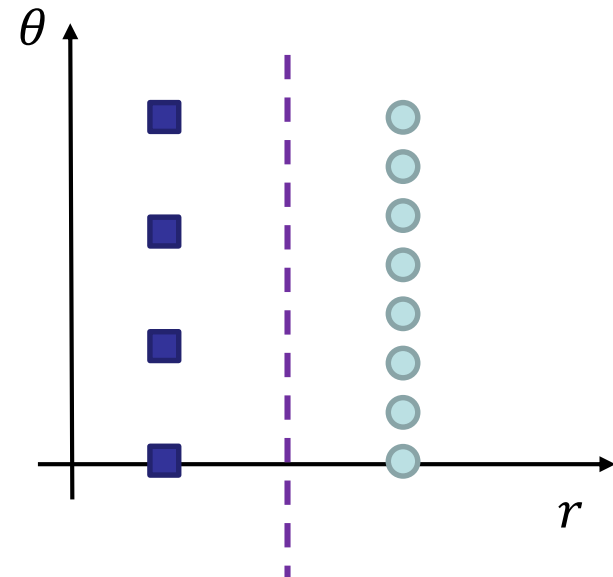
$\phi(x)$  = Transformation polynomiale

# Données non linéairement séparables

- Transformation non linéaire  $\phi(x)$  pour séparer linéairement les données d'origine



Coordonnées polaires



$\phi(x)$  = Transformation polaire

# Retour sur la formulation duale du SVM

Lagrangien

$$\max_{\alpha} \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \boxed{x_i x_j}$$

$$\text{tq } \forall i, 0 \leq \alpha_i \leq C$$

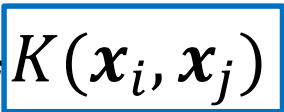
**Produit scalaire  
uniquement**



# « Kernel trick »

$$\max_{\alpha} \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$$

tq  $\forall i, 0 \leq \alpha_i \leq C$



Noyau

Le noyau  $K$  est un produit scalaire dans l'espace transformé:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$$

Il est uniquement nécessaire de connaître la similarité entre données pour introduire la non linéarité dans le problème (avec des conditions...)

# Utilisation de noyaux dans les SVM

- Permet d'introduire des mesures de similarités propres au domaine étudié et sans avoir à gérer la complexité de la transformation
- Permet de séparer modélisation = noyau de la classification et SVM (optimisation)
- Définit la fonction de classification à partir de noyaux « centrés » sur les vecteurs de support

$$F(\mathbf{x}, \mathbf{w}) = b + \sum_i \alpha_i y_i \mathbf{K}(\mathbf{x}_i, \mathbf{x})$$

# Noyaux courants

- Polynômes de degrés supérieurs à  $d$

$$K(\mathbf{x}, \mathbf{y}) = (\mathbf{x} \cdot \mathbf{y} + 1)^d$$

- Noyau gaussien

$$K(\mathbf{x}, \mathbf{y}) = \exp\left(-\frac{(\mathbf{x} - \mathbf{y})^T (\mathbf{x} - \mathbf{y})}{2\sigma^2}\right)$$

Paramètres à définir  
= degré de liberté  
supplémentaire

- Intersection d'histogrammes

$$K(\mathbf{x}, \mathbf{y}) = \sum_i \min(x^i, y^i)$$

# Résumé sur SVM

- Une formulation optimale quadratique du problème de classification binaire:
  - Primal: optimisation d'un critère empirique + régularisation
  - Dual: permet d'introduire parcimonie et « kernel trick »
  - ➔ plusieurs manières d'optimiser
- Les solutions s'expriment comme des combinaisons linéaires éparées de noyaux:

$$F(\mathbf{x}) = \text{sign}(b + \sum_i \alpha_i y_i \mathbf{K}(\mathbf{x}_i, \mathbf{x}))$$

où  $\alpha_i > 0$  seulement pour les vecteurs de support, 0 sinon.

- En pratique, ce qu'il faut régler:
  - Le coefficient de régularisation: C
  - Le type de noyau et ses caractéristiques
  - Les paramètres de l'optimiseur



# Multiclasse

# Différents types de classification

- Binaire

$$\mathcal{A} = \{-1, 1\}$$

- Multi classe

$$\mathcal{A} = \{1, 2 \dots L\}$$

- Détection (quoi et où)

$$\mathcal{A} = \{1, 2 \dots L\} \times R^4$$

- Caractérisation des données:

- Rejet
- Anomalie

$$\mathcal{A} = \{1, 2 \dots L, \text{ambigu}, \text{inconnu}\}$$

# Hypothèses multiples

- Toutes les classes/hypothèses ne se valent pas
  - Classes plus rares que d'autres (non équilibrées)
  - Coût d'une erreur de classification dépend des classes (Zèbre vs. Gazelle vs. Lion)
- Deux stratégies:
  - Optimiser un critère multi-hypothèse dans l'apprentissage
    - Par exemple entropie dans arbre de décision, softmax dans réseaux de neurones...
  - Utiliser un ensemble de classifieurs binaires
    - SVM, adaboost, perceptron...

# Multiclasse à partir de classifieurs

- Comment passer d'une classification binaire à  $N$  classes?
- Plusieurs techniques:
  - One vs Rest
  - One vs One (ou All vs All)
- OVO:
  - On apprend autant de classifieurs que de **paires de classes** ( $N(N-1)/2$ )
  - Classification = choix de la classe ayant le plus de **votes**
  - **Pb**: peut être indécidable dans certains cas
- OVR:
  - On apprend **un classifieur par classe**
  - Classification = choix de la classe ayant **le meilleur score**
  - **Pb**: déséquilibre des données entre classe cible et « reste »

# Evaluation du multi-classe

- Erreur globale:

$$Err = \frac{\text{nombre d'échantillons mal classés}}{\text{nombre d'échantillons testés}}$$

- Matrice de confusion:

$\text{conf}(i, j)$  = probabilité de classer comme  $i$  | vraie classe est  $j$   
estimée sur données de test

- Risque ou coût moyen

$$R = \sum_j \sum_i \lambda(i, j) \text{conf}(i, j) p(j)$$

où  $\lambda(i, j)$  est le coût de décider  $i$  lorsque  $j$  est vrai

# A retenir

- Régularisation
  - Un moyen de contrôler le compromis biais-variance
- SVM
  - Un algorithme optimal et flexible qui permet de traiter un grand nombre de configurations de données (en dimension raisonnable)
- Validation croisée
  - Un moyen empirique d'estimer l'erreur de généralisation
  - Une technique pour optimiser les hyper-paramètres (par ex. ceux du SVM)
- Multi-classe
  - Un problème qui peut s'exprimer et se résoudre de différentes manières

# Le TD

- Partie 1: Paramétrage du SVM
  - 4 activités sur données 2D
  - Tester et fournir des éléments de codes, illustrations et commentaires
  - Utilisation de la bibliothèque scikit-learn
- Partie 2: Classification de chiffres manuscrits
  - Passage au multi-classe
  - Optimisation globale (caractéristique, noyau, régularisation...)