

APPRENTISSAGE MACHINE & DEEP LEARNING

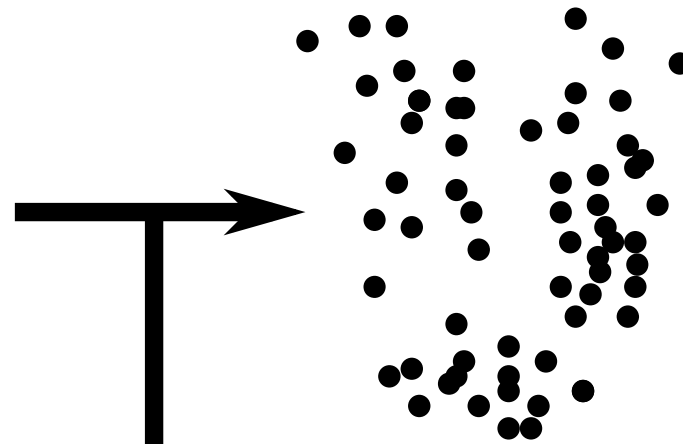
Classification supervisée L'exemple des SVMs

A. Boulch, A. Chan Hon Tong, S. Herbin, B. Le Saux

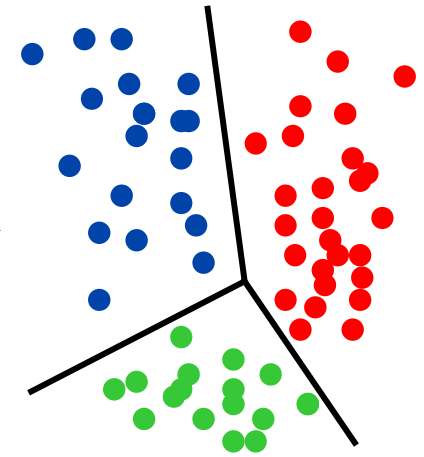
Interprétation des données



Données



Espace
adapté



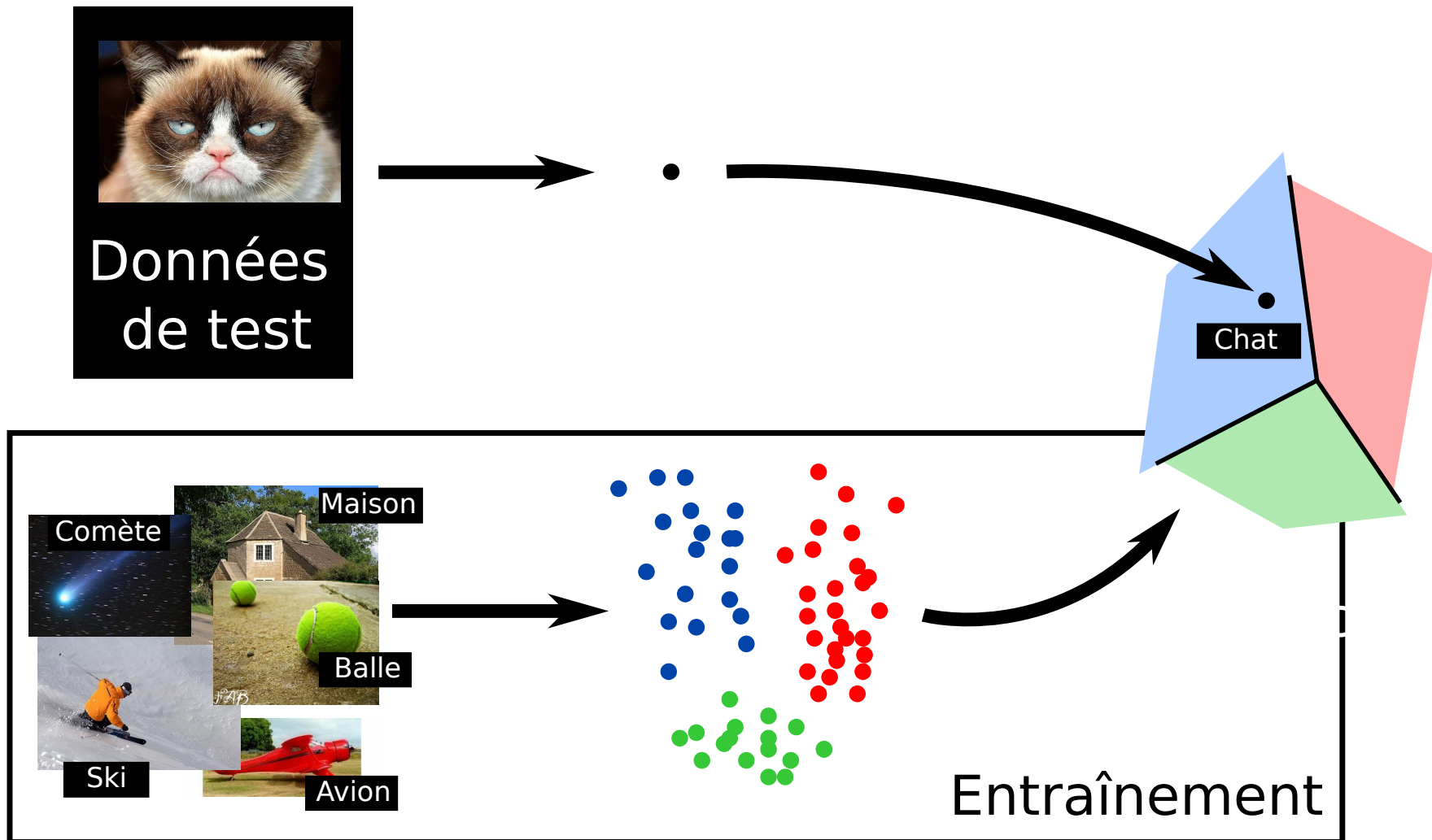
Objectif

Caractéristiques
bien pensées

Apprentissage
Expertise

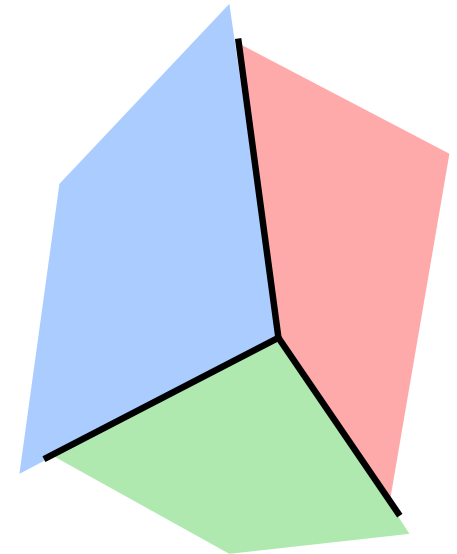
- **Apprentissage supervisé**
- Sur / Sous apprentissage
- Classification
- Régularisation
- SVM

- Prendre une décision basée sur l'expérience



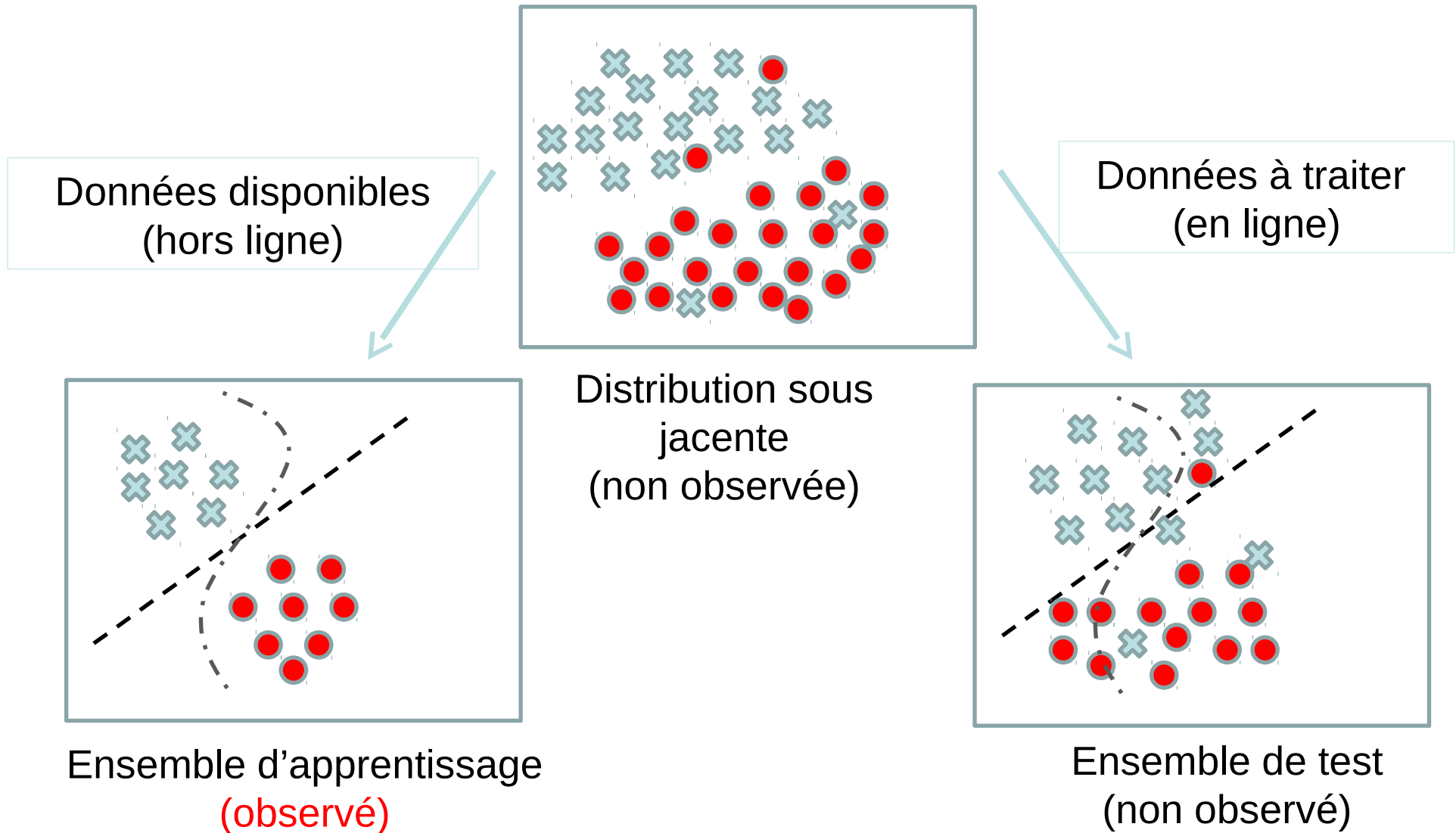
Apprentissage

- Déterminer une fonction de décision
 - Les données de **test** et de **train** sont différentes
 - Train : les données qu'on maîtrise
 - Test : les nouvelles données sur lesquelles on applique l'algorithme
- ⇒ On veut une fonction de décision la plus générique possible



Apprentissage

Train & Test



Que veut-on apprendre ?

- Apprentissage non supervisé
 - Construire une fonction de décision à partir des caractéristiques uniquement
 - Clustering (K-means, Mean-shift ...)
- Apprentissage supervisé
 - Construire une fonction décision à partir de données pour lesquelles la décision souhaitée est connue.



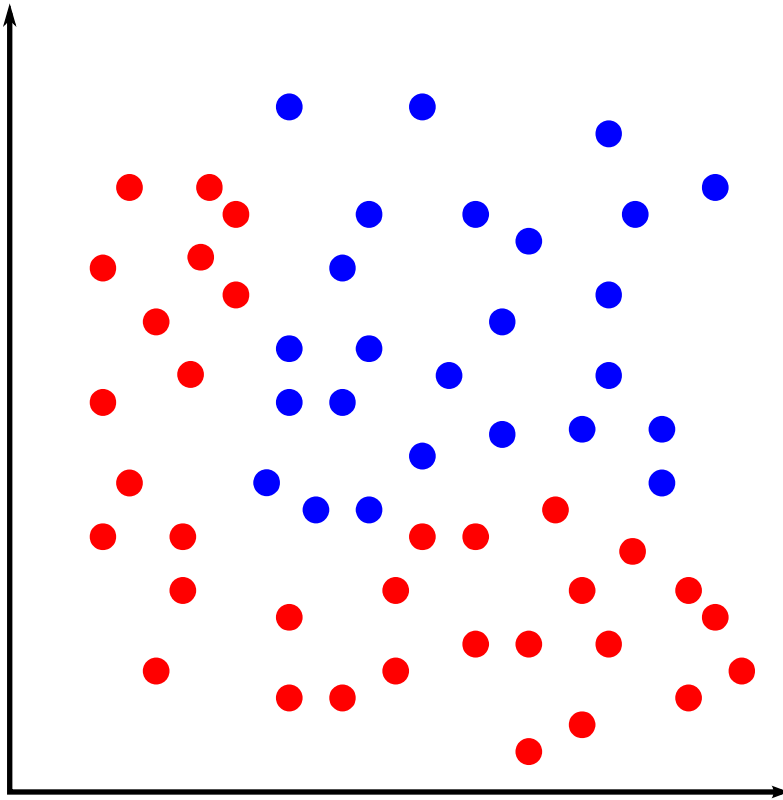
$$L = \{ (x_i, y_i) \}_i$$

La donnée

Le label associé

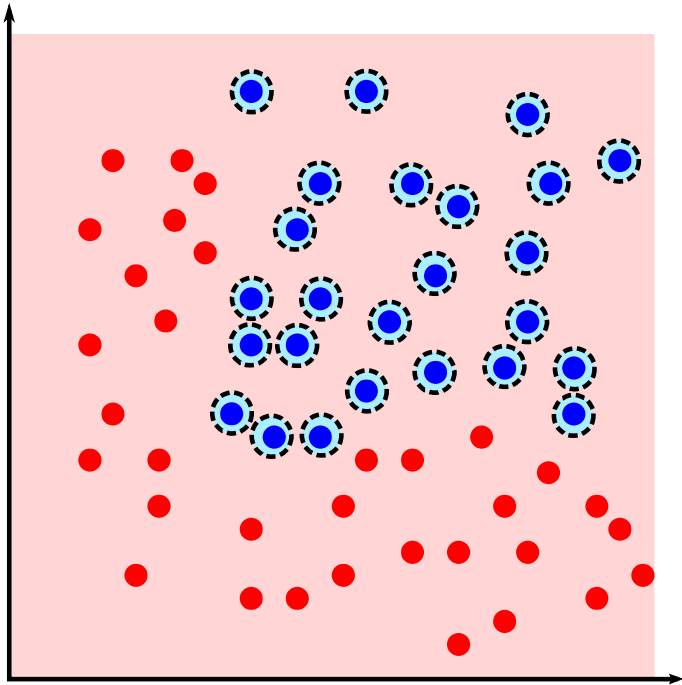
- Apprentissage supervisé
- **Sur / Sous apprentissage**
- Classification
- Régularisation
- SVM

Sur/Sous apprentissage



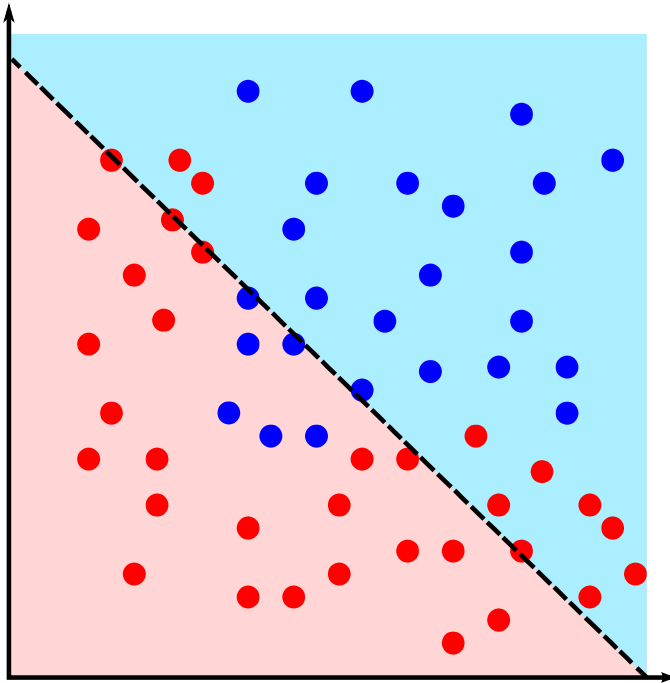
- Exemple 2D à deux classes
- Non linéairement séparable

Sur/Sous apprentissage



Sur apprentissage
(overfitting)

Apprentissage par coeur
Très bon sur les données
d'entraînement, mauvais
sur le test.



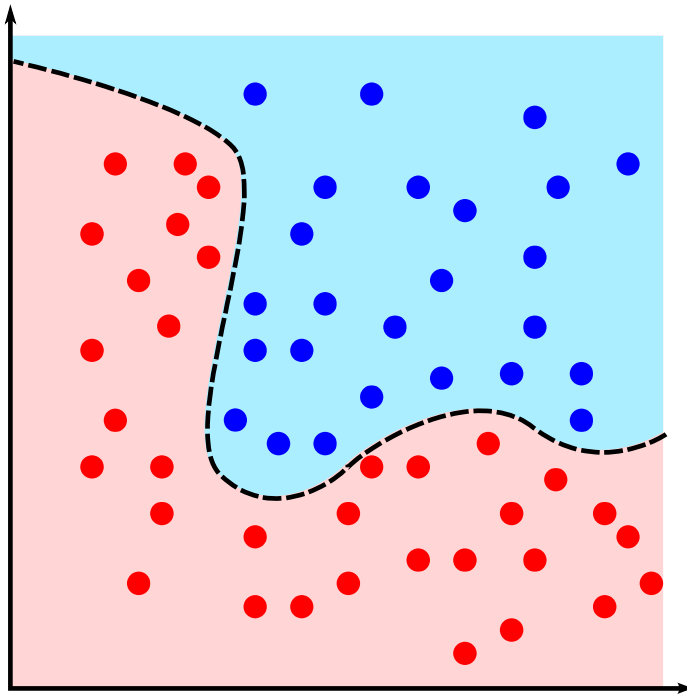
Sous apprentissage

Peu discriminant

- Type de fonction de décision non adaptées
- Problème 'trop' compliqué, mauvais espace de caractéristique

Mauvais en train et en test

Bon apprentissage



Peu discriminant

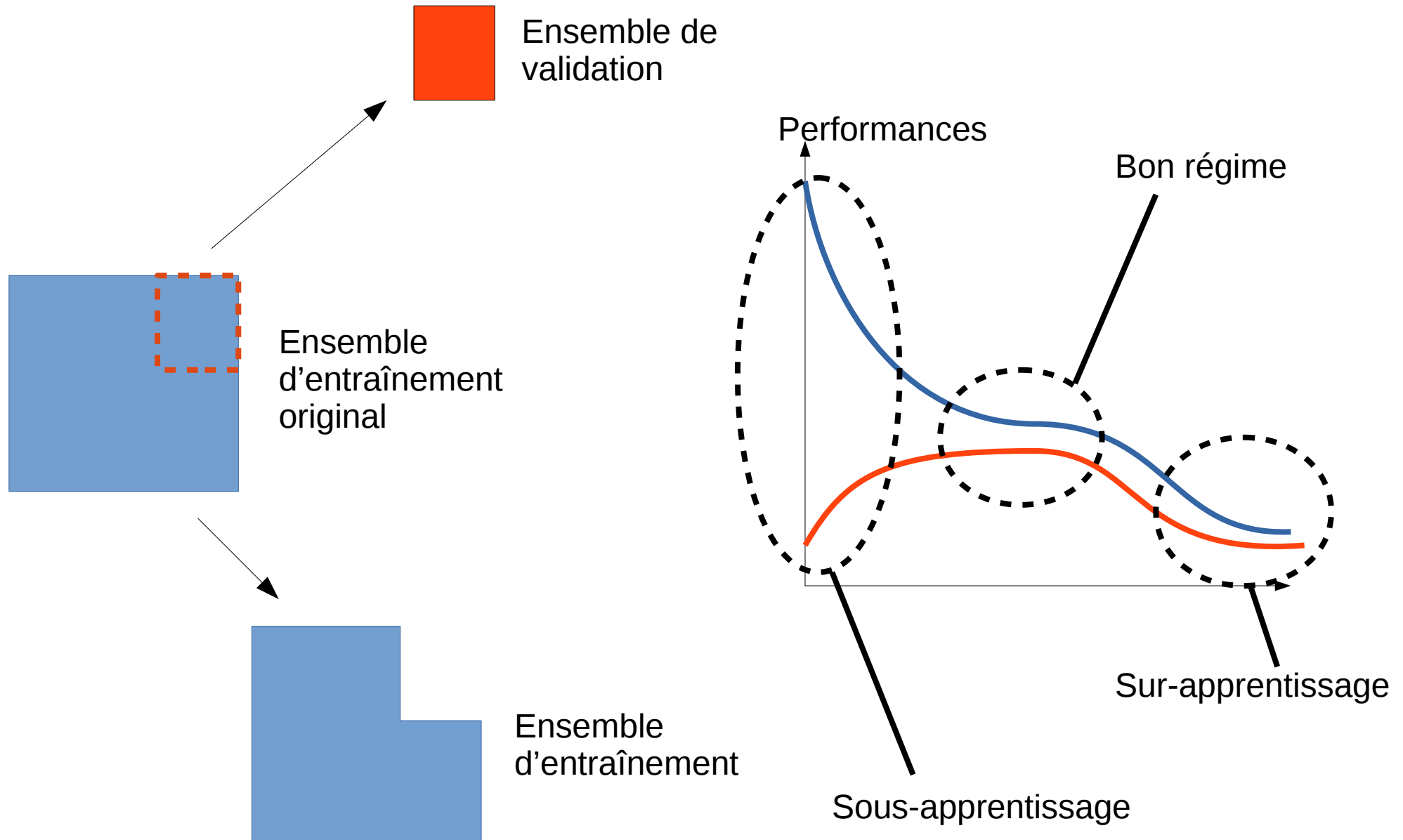
- Type de fonction de décision non adaptées
- Problème 'trop' compliqué, mauvais espace de caractéristique

Mauvais en train et en test

- Problème :
 - Comment savoir dans quel cas on se trouve ?
 - L'ensemble de test ne peut pas être utilisé (il est a priori inconnu)
 - On ne peut pas forcément visualiser la fonction de décision

⇒ Utiliser un ensemble de validation

Validation



Validation croisée

- Apprentissage supervisé
- Sur / Sous apprentissage
- **Classification**
- Régularisation
- SVM

- Régression

$$x \in \mathbb{R}^d \rightarrow y \in \mathbb{R}^k$$

- **Classification**

- Binaire $x_i \in \mathbb{R}^d \rightarrow y_i \in A = \{-1, 1\}$
- Multilabel $x_i \in \mathbb{R}^d \rightarrow y_i \in A = \{0, 1, \dots, N\}$
- Détection $x_i \in \mathbb{R}^d \rightarrow y_i \in A = \{0, 1, \dots, N\} \times \mathbb{R}^4$
- Rejet $x_i \in \mathbb{R}^d \rightarrow y_i \in A = \{0, 1, \dots, N, other\} \times \mathbb{R}^4$

Erreur de généralisation

- Erreur de généralisation (ou de test, ou idéale...)

$$E_{test}(w) = E_{X,Y}[\{D(x, w) \neq y\}]$$

- Inaccessible

Erreur empirique

- Risque ou erreur empirique

$$E_{train}(w, L) = \frac{1}{N} \sum_{i=1}^N \{ D(x_i, w) \neq y_i \}$$

- Erreur de généralisation (ou de test, ou idéale...)

$$E_{test}(w) = E_{X,Y} [\{ D(x, w) \neq y \}]$$

- Critère à optimiser (fonction objectif):

$$Loss(w, L) = \frac{1}{N} \sum_{i=1}^N l(D(x_i, w), y_i)$$

Fonction objectif

- Risque ou erreur empirique

$$E_{train}(w, L) = \frac{1}{N} \sum_{i=1}^N \{ D(x_i, w) \neq y_i \}$$

- Erreur de généralisation (ou de test, ou idéale...)

$$E_{test}(w) = E_{X,Y} [\{ D(x, w) \neq y \}]$$

- Critère à optimiser (fonction objectif):

$$Loss(w, L) = \frac{1}{N} \sum_{i=1}^N l(D(x_i, w), y_i)$$

Les fonctions de coût

- Quelques fonctions de coût classique

- Squared loss (l2)

$$l(D(x_i, w), y_i) = (D(x_i, w) - y_i)^2$$

- L1 loss

$$l(D(x_i, w), y_i) = |D(x_i, w) - y_i|$$

- 0/1 loss (2 classes -1 et 1)

$$l(D(x_i, w), y_i) = 1(D(x_i, w) y_i \leq 0)$$

- Hinge loss (2 classes -1 et 1)

$$l(D(x_i, w), y_i) = \max(0, 1 - D(x_i, w) y_i)$$

- Exponential loss (2 classes -1 et 1)

Autres fonctions de coût classiques

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

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

Exponential: $l(y, y') = \exp(-yy')$

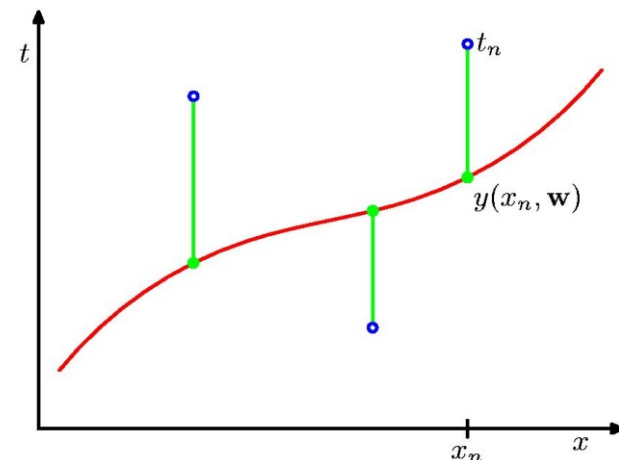
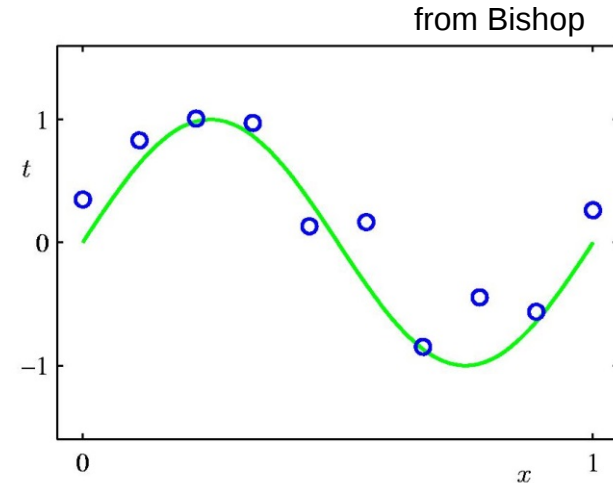
Squared loss: $l(y, y') = (y - y')^2$

Classification supervisée

- Apprentissage supervisé
- Sur / Sous apprentissage
- Classification
- **Régularisation**
- SVM

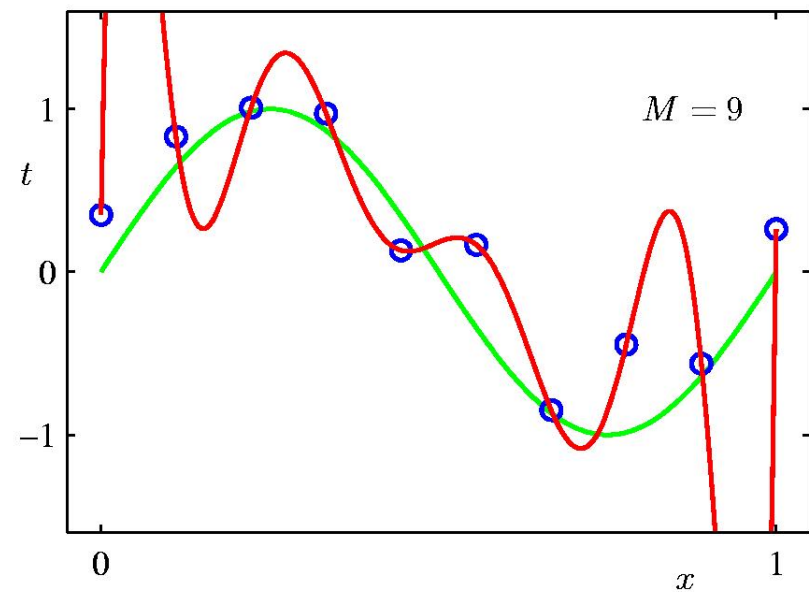
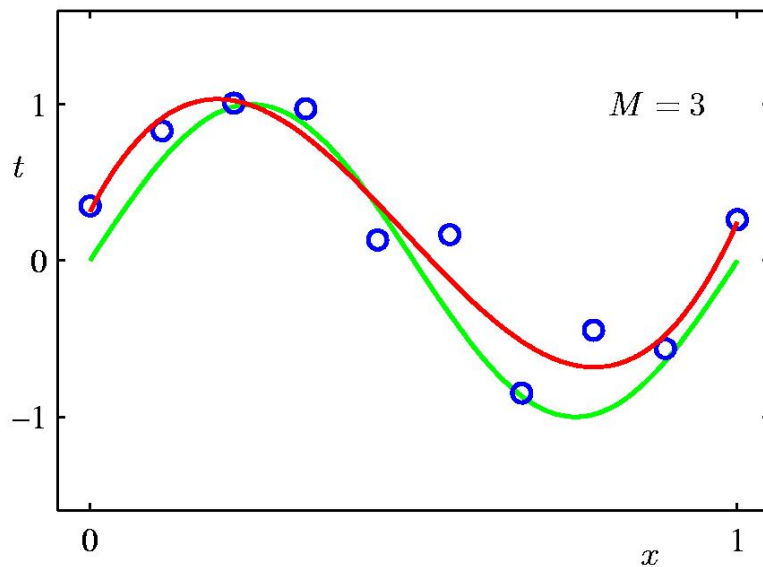
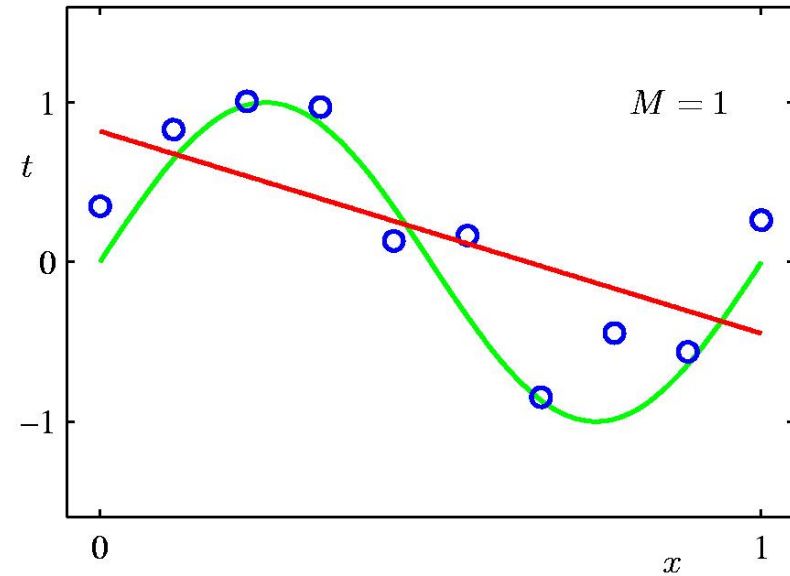
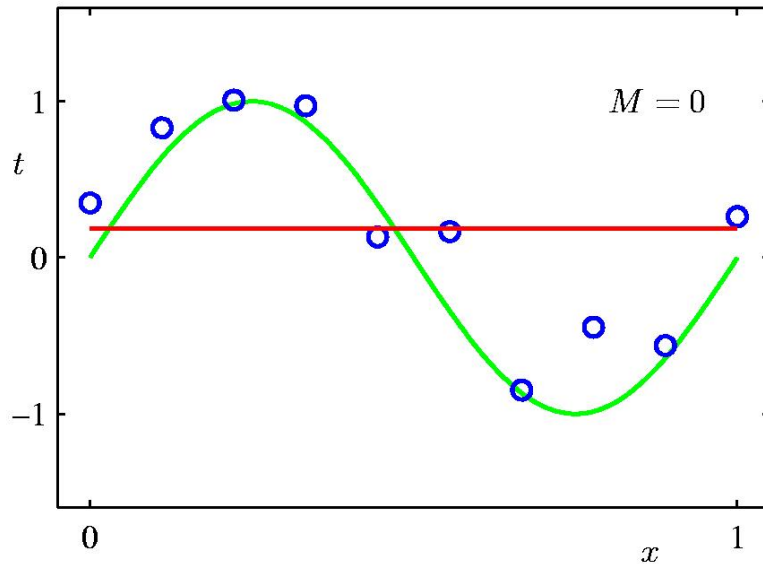
Un exemple « paradigmatique » : la régression polynomiale

- La courbe verte est la véritable fonction à estimer (non polynomiale)
- Les données sont uniformément échantillonnées en x mais bruitées en y .
- L'erreur de régression est mesurée par la distance au carré entre les points vrais et le polynôme estimé.



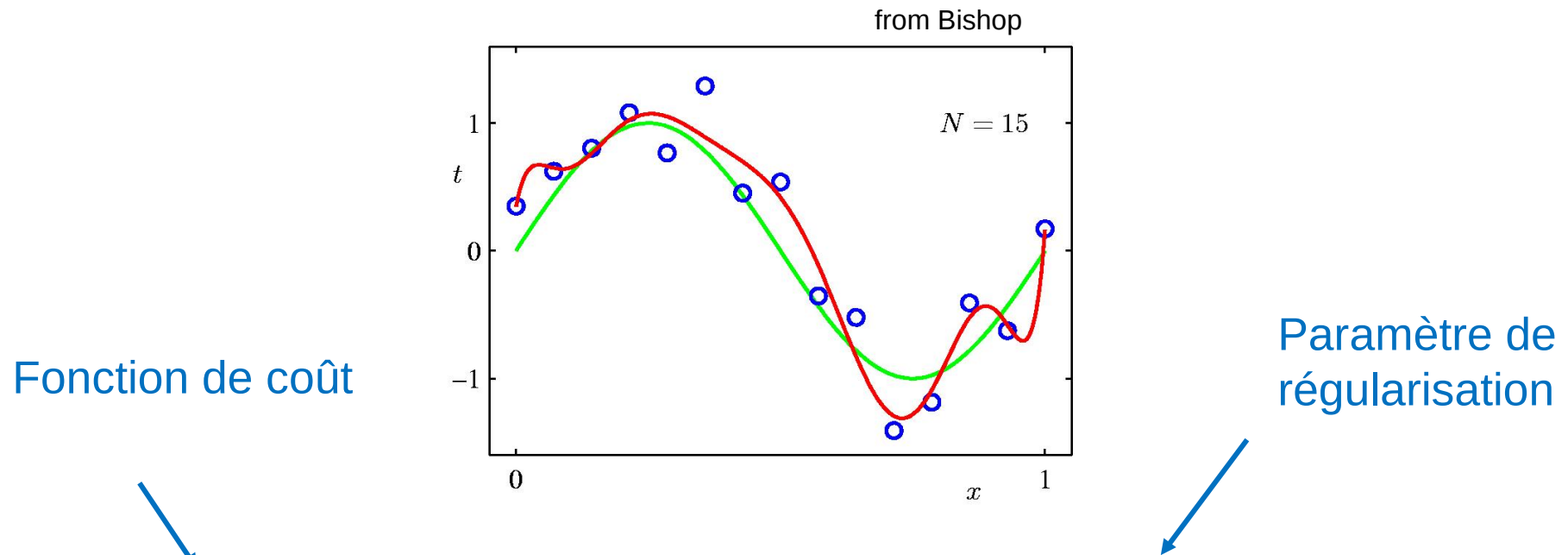
Quelles sont les meilleures régressions?

from Bishop



Une approche simple pour contrôler la complexité

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



$$E(\mathbf{w}) = \frac{1}{2} \sum_{i=1}^N \{D(\mathbf{x}_i, \mathbf{w}) - t_i\}^2 + \frac{\lambda}{2} \|\mathbf{w}\|^2$$

↑
Valeur vraie au
point \mathbf{x}_i

Minimiser l'erreur

- Risque ou erreur empirique

$$E_{\text{train}}(\mathbf{w}, L) = \frac{1}{N} \sum_{i=1}^N \{D(\mathbf{x}_i, \mathbf{w}) \neq y_i\}$$

- Erreur de généralisation (ou de test, ou idéale...)

$$E_{\text{test}}(\mathbf{w}) = E_{\mathbf{x}, Y} \left[\{D(\mathbf{x}, \mathbf{w}) \neq y\} \right]$$

- Critère à optimiser (fonction objectif):

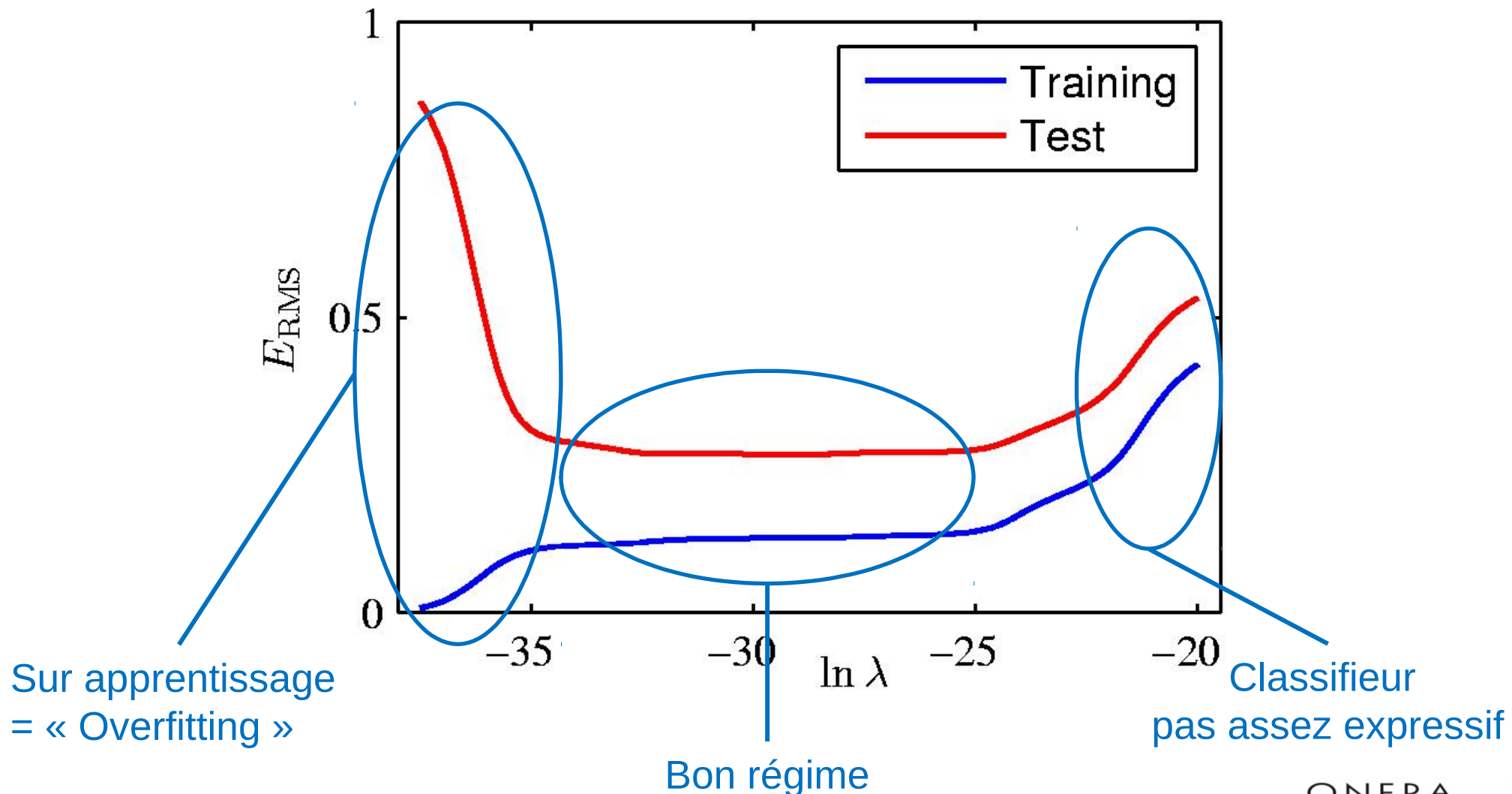
$$\text{Loss}(\mathbf{w}, L) = \frac{1}{N} \sum_{i=1}^N l(D(\mathbf{x}_i, \mathbf{w}), y_i) + r(\mathbf{w})$$

Adéquation aux données

Régularisation

Regularisation: E_{RMS} vs. $\ln(\lambda)$

$$E_{\text{RMS}}(\mathbf{w}) = \frac{1}{2} \sum_{i=1}^N \{D(\mathbf{x}_i, \mathbf{w}) - t_i\}^2$$

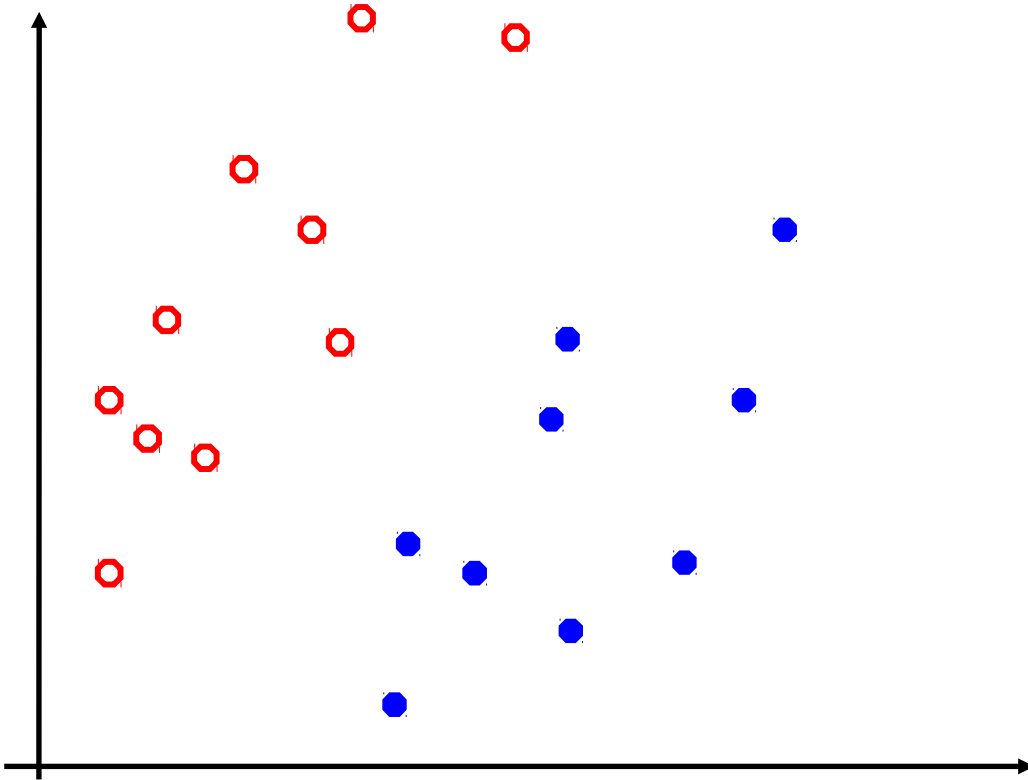


Classification supervisée

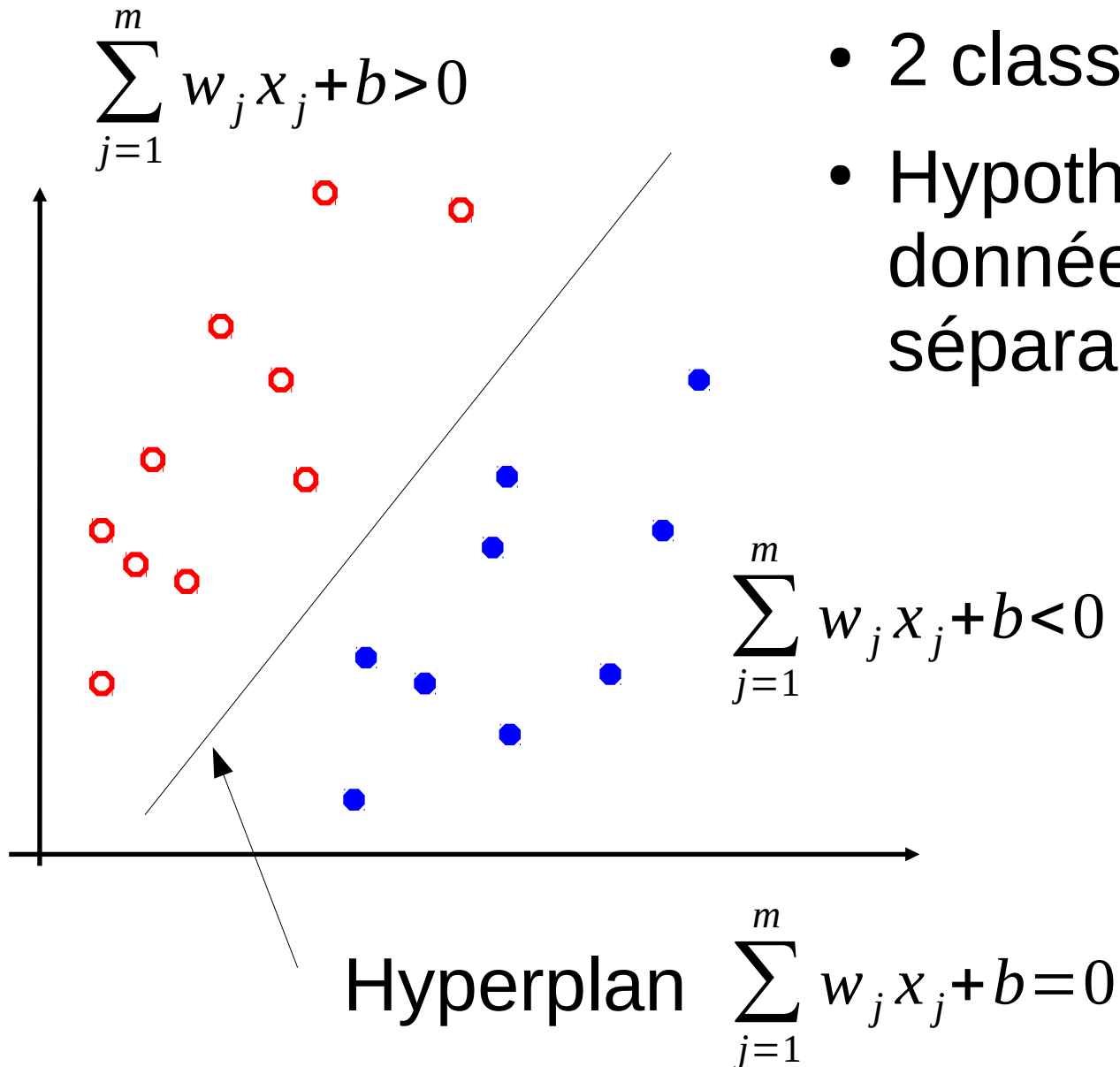
- Apprentissage supervisé
- Sur / Sous apprentissage
- Classification
- Régularisation
- **SVM**
 - Principe
 - Classifieur linéaire
 - Noyaux
 - Multiclasse

Principe

- 2 classes



- 2 classes
- Hypothèse : données linéairement séparables



- Apprentissage supervisé
- Sur / Sous apprentissage
- Classification
- Régularisation
- **SVM**
 - Principe
 - **Classifieur linéaire**
 - Noyaux
 - Multiclasse

- Équation de l'hyperplan séparateur

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

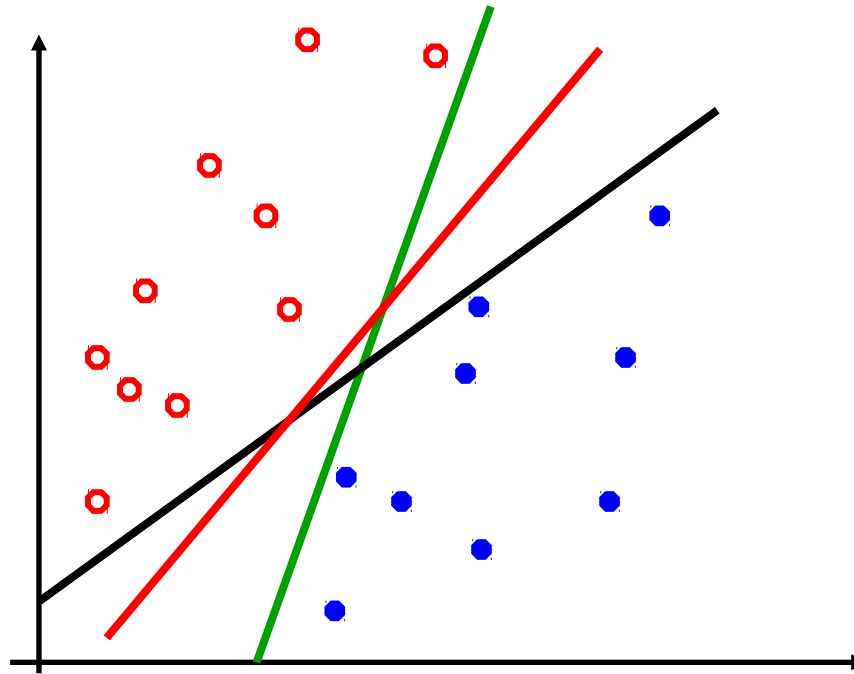
- Équation du classifieur linéaire ($y=1$ ou -1)

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

- Erreur

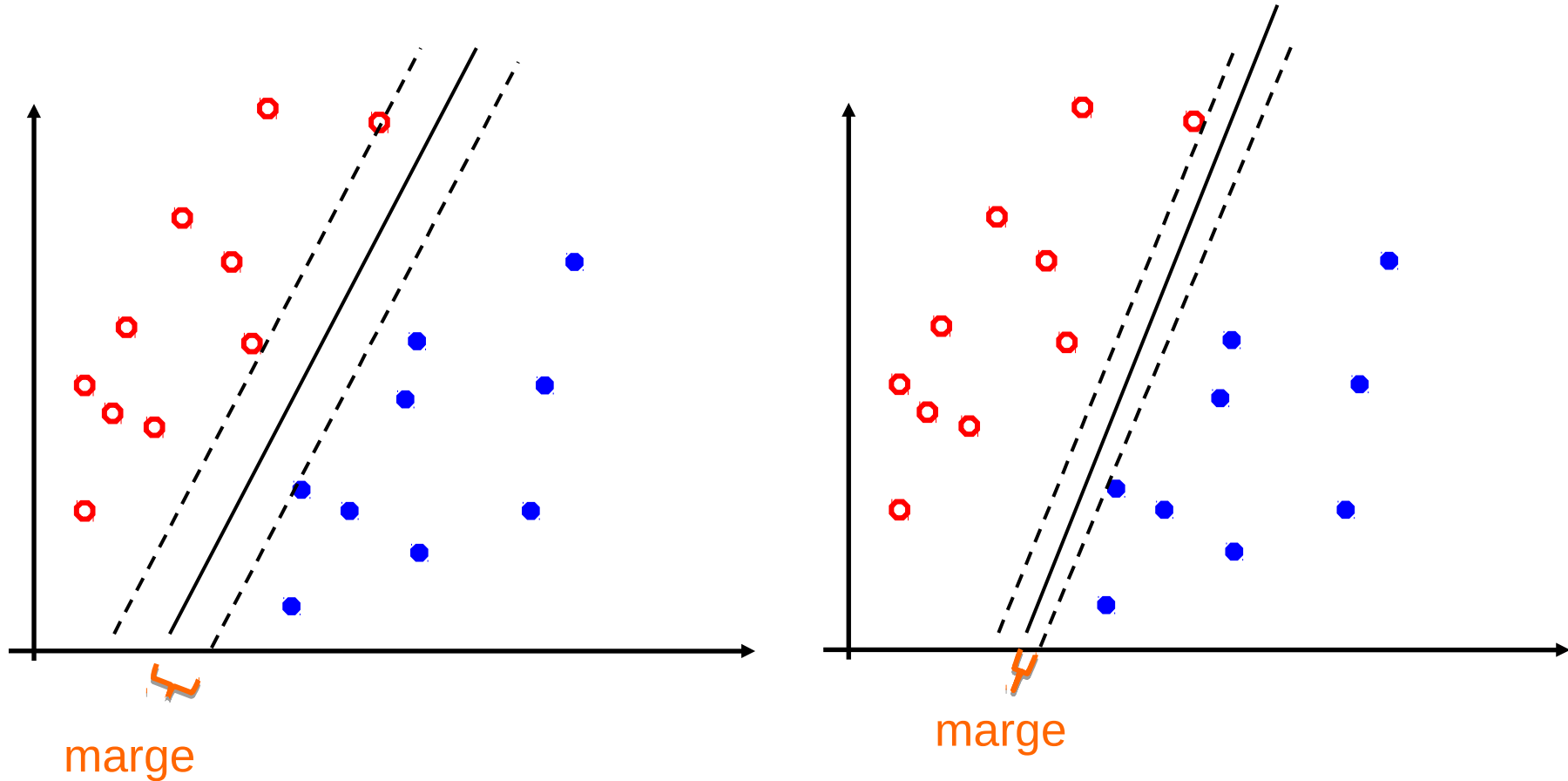
$$E_{\text{test}}(\mathbf{w}, L) = \frac{1}{N} \sum_{i=1}^N \{ y_i \cdot \text{sign}(b + \mathbf{w} \cdot \mathbf{x}_i) < 0 \}$$

Classifieur linéaire



Quel plan choisir ?

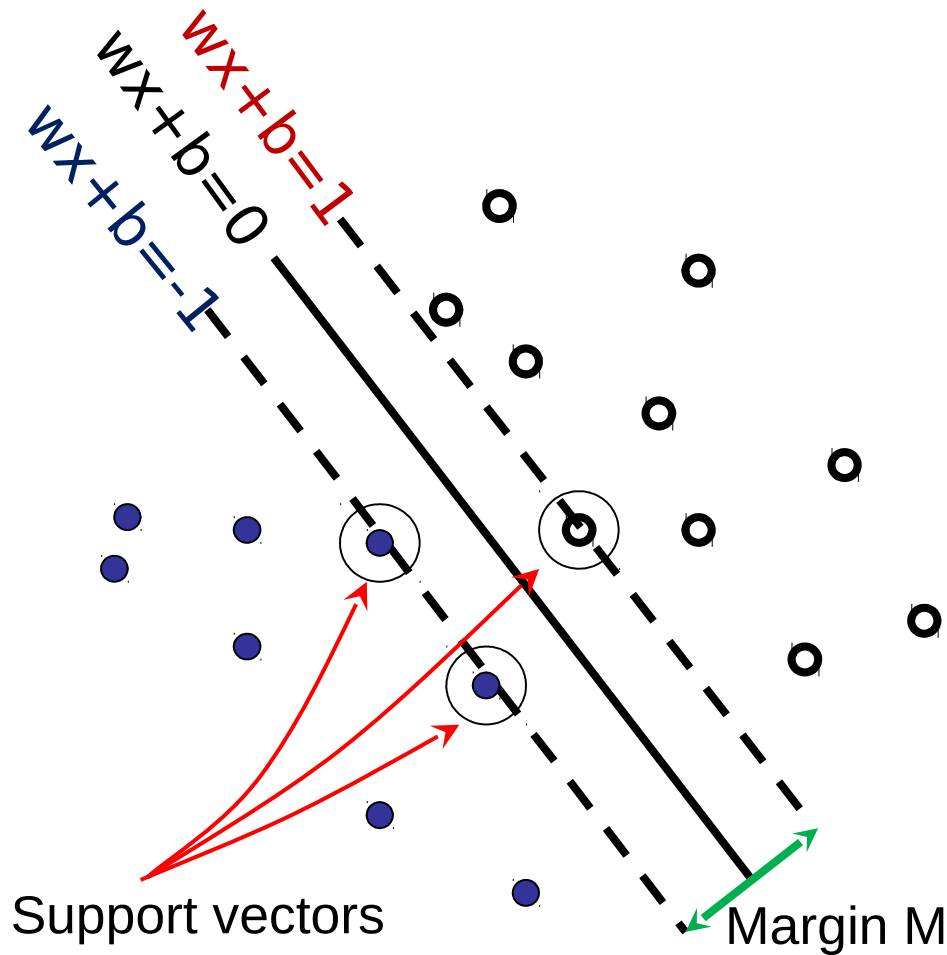
Classifieur Large Marge



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

Support Vector Machine

- On cherche l'hyperplan qui maximise la marge.



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

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

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 (Large Margin)

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

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

- Sous contraintes

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

- Les vecteurs supports :

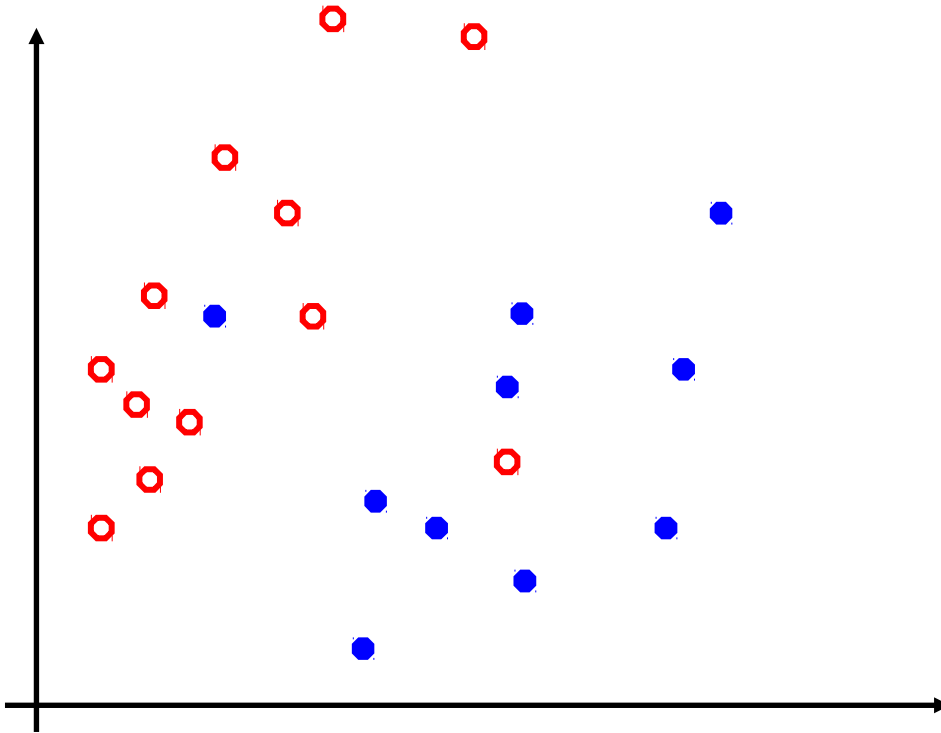
$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) = 1$$

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
 - Grand nombre de manières de l'optimiser

Classification Soft Margin



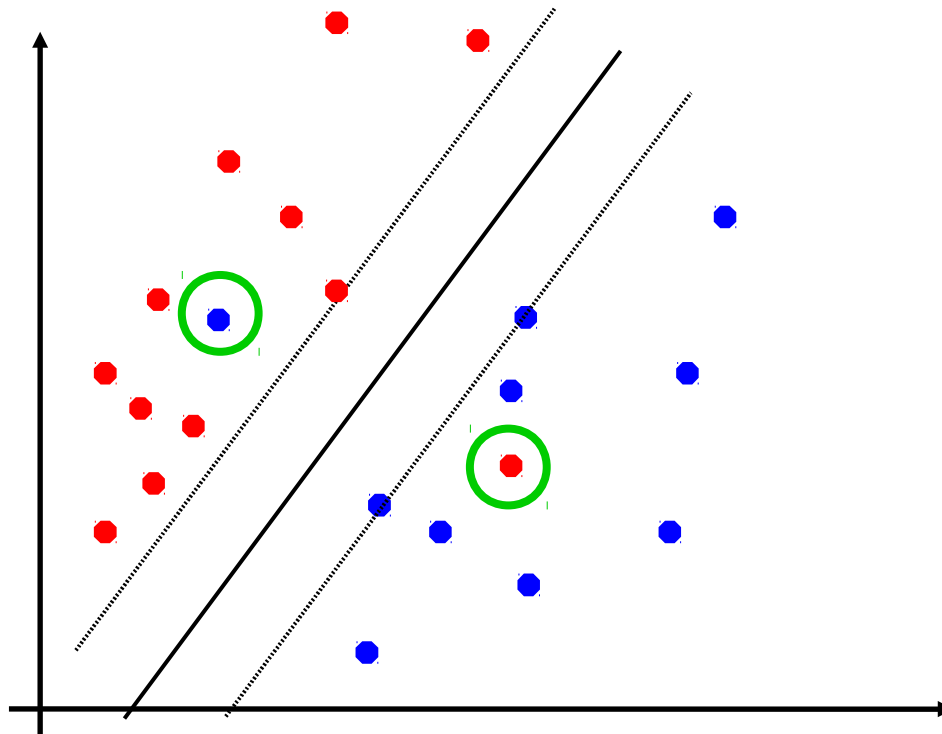
$$\min_{w,b} \|w\|^2$$

Tel que:

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

Comment traiter le cas non linéairement séparable?

- Tolérer certaines violations de contraintes



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:

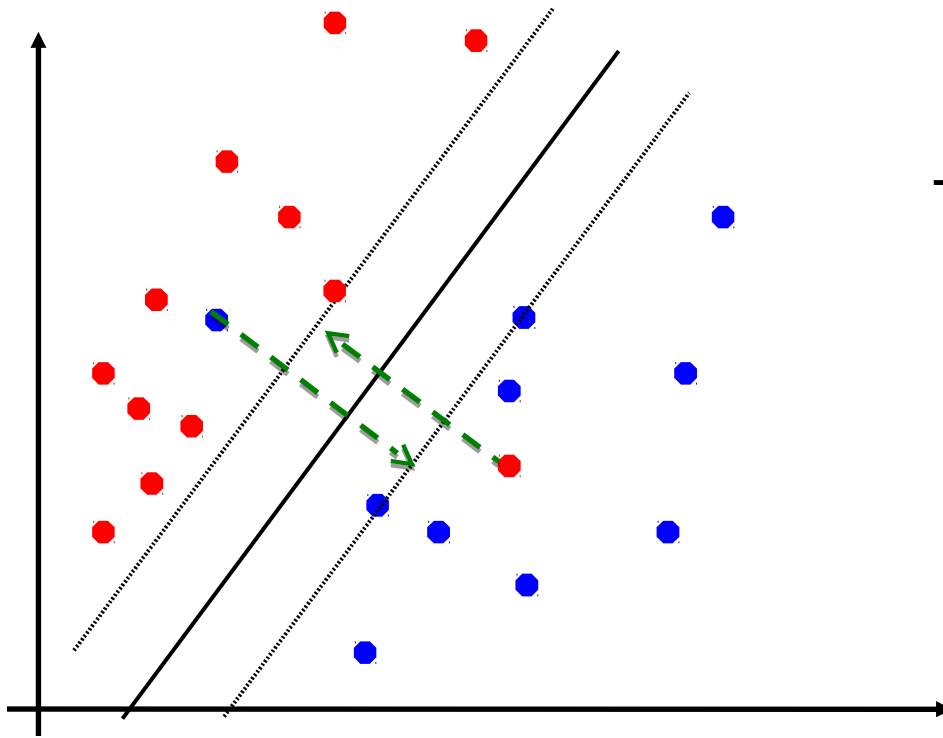
$$y_i(w \cdot x_i + b) \geq 1 - \zeta_i \quad \forall i$$

$$\zeta_i \geq 0$$

Permet de relacher la contrainte de séparabilité pour chaque exemple.

slack variables
(une par exemple)

Slack Variables



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

Tel que:

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

Relâchement de la contrainte

Slack Variables

marge

Compromis entre marge et pénalisation de la contrainte

Valeur du relâchement de la contrainte

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

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!

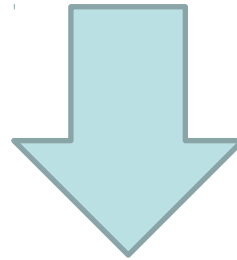
Soft Margin SVM

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

tq:

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

$$\zeta_i = \max(0, 1 - y_i(w \cdot x_i + b))$$



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

Regularisation

Hinge Loss

Problème d'optimisation non contraint

Autres méthodes d'optimisation (descente de gradient)

Code (Python)

```
>>> # data

>>> import numpy as np

>>> X = np.array([[-1, -1], [-2, -1], [1, 1], [2, 1]])

>>> y = np.array([1, 1, 2, 2])


>>> # create and train the classifier

>>> from sklearn.svm import SVC

>>> clf = SVC(C=1.0)

>>> clf.fit(X, y)


>>> # predict

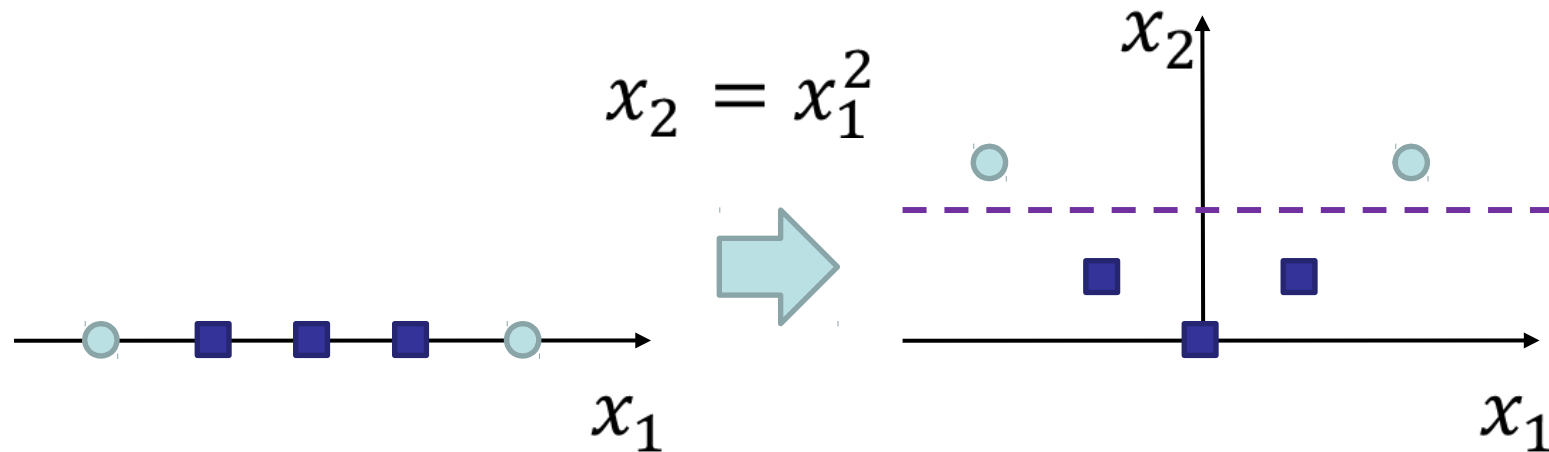
>>> print(clf.predict([[-0.8, -1]]))

[1]
```

- Apprentissage supervisé
- Sur / Sous apprentissage
- Classification
- Régularisation
- **SVM**
 - Principe
 - Classifieur linéaire
 - **Noyaux**
 - Multiclasse

Données non linéairement séparables

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



$\phi(x)$ = Transformation polynomiale

SVM : forme duale

- Problème d'optimisation sous contrainte

Pour simplifier l'expression des calculs

Primal

$$\begin{aligned} \operatorname{argmin}_{\mathbf{w}} \frac{\|\mathbf{w}\|^2}{2} + C \sum_i \xi_i & \quad \text{Multiplicateurs de Lagrange} \\ \text{s. t. } \forall i, y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 - \xi_i & \quad \alpha_i \\ \xi_i \geq 0 & \quad \beta_i \end{aligned}$$

Dual (Lagrangien)

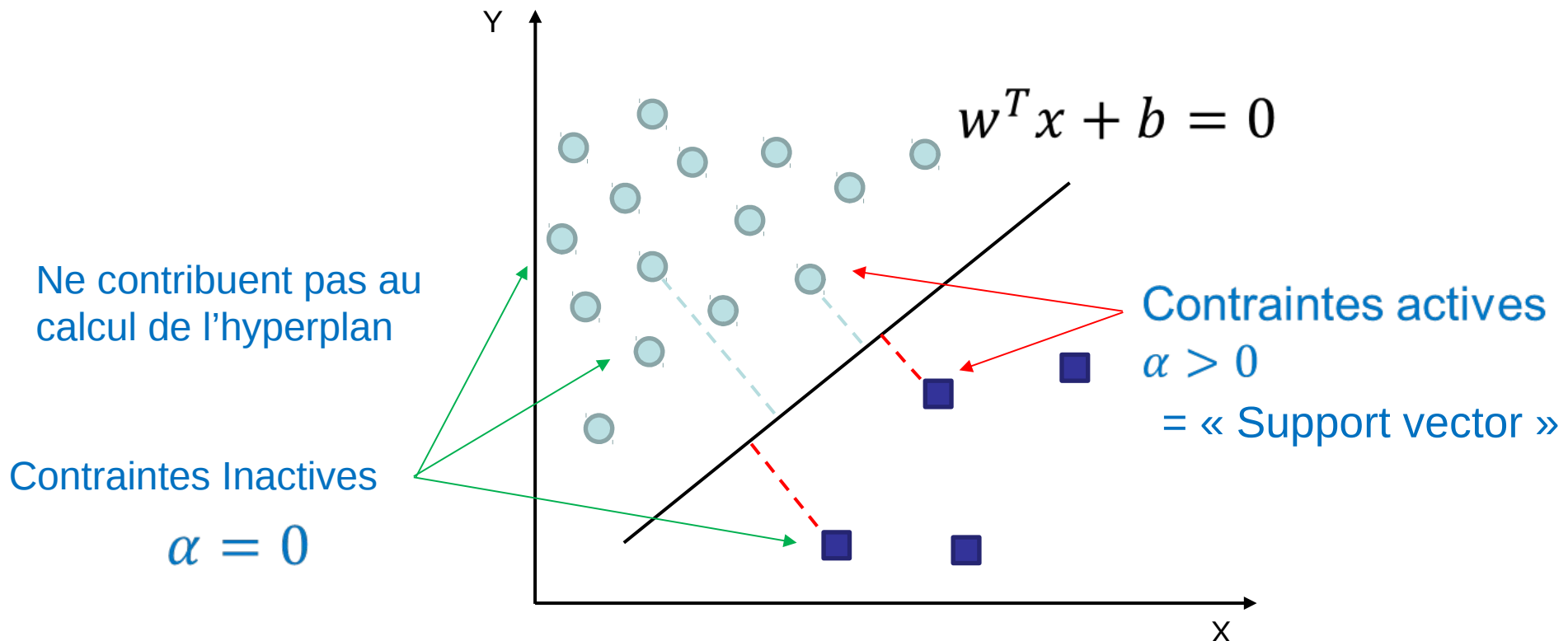
$$\begin{aligned} L(\mathbf{w}, \xi, \alpha, \beta) \\ = \frac{\|\mathbf{w}\|^2}{2} + \sum_i (C \xi_i - \alpha_i (y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1 + \xi_i) - \beta_i \xi_i) \\ \text{s. t. } \forall i, \alpha_i \geq 0, \beta_i \geq 0 \end{aligned}$$

Sparsité du SVM

- Seuls certains α 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 $w = \sum_i \alpha_i y_i x_i$



Formulation duale

Lagrangien

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

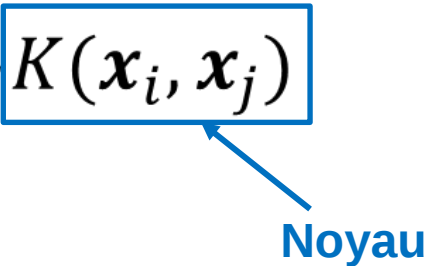
$$\text{tq } \forall i, 0 \leq \alpha_i \leq C$$

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...)

- 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

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

- 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

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

Noyaux - Intérêt

- Noyaux = partie « métier »
 - Connaissance du sens à donner à la similarité
 - Choix expert des espaces de représentation et des caractéristiques informatives
- Classification/Optimisation = partie « ML »
 - Utilisation d'optimiseurs génériques
 - Certaine optimalité à représentation du problème donnée
- Utilisation des SVM à noyaux.
 - Représentations multi dimensionnelles (mais pas trop...)
 - Données de volume raisonnable
 - mais passage à l'échelle plus délicat (on se restreint alors en général aux noyaux linéaires)
- « Deep Learning » montrera que ce peut être sous-optimal...

- Apprentissage supervisé
- Sur / Sous apprentissage
- Classification
- Régularisation
- **SVM**
 - Principe
 - Classifieur linéaire
 - Noyaux
 - **Multiclasse**

- Comment passer d'une classification binaire à multiple?
- Plusieurs techniques:
 - One vs All
 - One vs One (ou All vs All)
- OVO:
 - on apprend autant de classifieurs que de paires de classes
 - Classification = choix de la classe ayant le plus de votes
- OVA:
 - on apprend un classifieur par classe
 - Classification = choix de la classe ayant le meilleur score

Multiclasse – One vs One

apple vs orange



apple



orange



apple



banana



banana



+1



+1



-1

orange vs banana



+1



-1



-1

apple vs banana



+1



+1



-1



-1

Multiclasse – One vs One

apple vs orange



+1



+1



-1

Vote

orange vs banana



+1



-1



-1

apple vs banana



+1



+1



-1



-1



Quelle classe?

Multiclasse – One vs One

apple vs orange



+1



+1

orange



-1

apple vs banana



+1



+1

apple



-1



-1

orange vs banana



+1



-1



-1

orange



orange

SVM - conclusion

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

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

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

- Apprentissage supervisé
 - Problème connu
 - En pratique de nombreuses librairies
- SVM
 - Démarche générique et solide théoriquement
 - Bonne performances (jeu de données de taille modeste, faibles dimensions)
- Il existe de nombreuses méthodes de classification :
 - Arbres de décision, boosting, réseaux de neurones...

Codes

- LibSVM + LibLinear (nombreuses interfaces)
<https://www.csie.ntu.edu.tw/~cjlin/libsvm/>
<https://www.csie.ntu.edu.tw/~cjlin/liblinear/>
- Scikit-Learn (Python)
<https://www.csie.ntu.edu.tw/~cjlin/liblinear/>
- Matlab