Discrimination linéaire

R. Flamary

5 mars 2015

Sommaire

Objectifs du cours

Introduction

- Discrimination linéaire binaire.
- Problème d'optimisation.

Méthodes de discrimination linéaires

- Régression logistique.
- Perceptron de Rosenblatt.
- Séparateurs à vaste marge.

Méthodes d'optimisation

- Méthode du gradient.
- Méthode de Newton.
- Méthode du gradient stochastique.

On se concentrera sur des problèmes de classification binaire (deux classes).

Prédiction linéaire

Fonction linéaire

Fonction $f : \mathbb{R}^d \to \mathbb{R}$, de la forme

$$f(\mathbf{x}) = \sum_{i=1}^{d} w_i x_i + b = \mathbf{x}^\top \mathbf{w} + b = [\mathbf{x}^\top 1]\boldsymbol{\alpha} = \tilde{\mathbf{x}}^\top \boldsymbol{\alpha} \tag{1}$$

avec $\mathbf{w} \in \mathbb{R}^d$ un vecteur définissant un hyperplan dans $\mathbb{R}^d$ et $b \in \mathbb{R}$ un biais qui déplace la fonction perpendiculairement à l'hyperplan, et $\boldsymbol{\alpha} = \begin{bmatrix} \mathbf{w} \\ b \end{bmatrix}$ est un vecteur de $\mathbb{R}^{d+1}$ contenant la concaténation de $\mathbf{w}$ et b.

Objectifs

- Régression : $f(\cdot) \in \mathbb{R}$.
- Classification : $\text{signe}(f(\cdot)) \in \{-1, 1\}$.

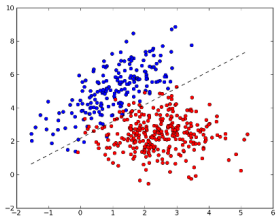
Classification linéaire

Objectif

- Apprendre une fonction linéaire $f(\cdot)$ permettant de prédire une valeur binaire $y \in \{-1, 1\}$ à partir d'une observation $\mathbf{x} \in \mathbb{R}^d$.
- En pratique on cherche à déterminer les coefficients $(\mathbf{w}, b)$ de $f(\cdot)$ à partir d'un ensemble d'apprentissage $\{\mathbf{x}_i, y_i\}_{i=1,\dots,n}$.
- La classe prédite est le signe de la fonction de prédiction $f(\cdot)$

Exemples

- Reconnaissance de caractères.
- Aide au diagnostique.
- Inspection de pièces.



Représentation des données

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^\top & 1 \\ \mathbf{x}_2^\top & 1 \\ \vdots & \vdots \\ \mathbf{x}_i^\top & 1 \\ \vdots & \vdots \\ \mathbf{x}_n^\top & 1 \end{bmatrix} = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1j} & \dots & x_{1d} & 1 \\ x_{21} & x_{22} & \dots & x_{2j} & \dots & x_{2d} & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{i1} & x_{i2} & \dots & x_{ij} & \dots & x_{id} & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{n1} & x_{n2} & \dots & x_{nj} & \dots & x_{nd} & 1 \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_i \\ \vdots \\ y_n \end{bmatrix}$$

Données d'apprentissage

Exemples :

- $\mathbf{x}_i \in \mathbb{R}^d$ observations pour $i = 1, \dots, n$.
- $\tilde{\mathbf{x}}_i \in \mathbb{R}^{d+1}$ tel que $\tilde{\mathbf{x}}_i^\top = [\mathbf{x}_i, 1]^\top$
- $y_i \in \{-1, 1\}$ valeur à prédire $i = 1, \dots, n$.

Forme matricielle :

- $\mathbf{X} \in \mathbb{R}^{n \times (d+1)}$ telle que $\mathbf{x} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n, \mathbf{e}]^\top$ avec $\mathbf{e} \in \mathbb{R}^d$ et $e_i = 1, \forall i$
- $\mathbf{y} \in \mathbb{R}^n$ telle que $\mathbf{y} = [y_1, y_2, \dots, y_n]^\top$.
- $\boldsymbol{\alpha} \in \mathbb{R}^{d+1}$est un vecteur de tel que $\boldsymbol{\alpha} = \begin{bmatrix} \mathbf{w} \\ b \end{bmatrix}$

Régression logistique

Objectifs

- Apprendre une fonction linéaire discriminante.
- Modéliser les probabilités conditionnelles (prédiction probabiliste).
- Éviter les estimations des paramètres des lois de probabilité.

Approche

- On suppose que la probabilité conditionnelle $P(\omega_1|\mathbf{x})$ est de la forme :

$$P(\omega_1|\mathbf{x}) = \frac{\exp(\mathbf{w}^\top \mathbf{x} + b)}{1 + \exp(\mathbf{w}^\top \mathbf{x} + b)} = \frac{1}{1 + \exp(-\mathbf{w}^\top \mathbf{x} - b)} \tag{2}$$

et donc que

$$P(\omega_2|\mathbf{x}) = 1 - P(\omega_1|\mathbf{x}) = \frac{1}{1 + \exp(\mathbf{w}^\top \mathbf{x} + b)} \tag{3}$$

Rapport de vraisemblance

Fonction de décision

- La fonction de décision utilise le test du rapport de vraisemblance.
- Si $P(\omega_1|\mathbf{x}) > P(\omega_2|\mathbf{x})$ choisir ω_1 sinon ω_2 :

$$\text{ou} \quad P(\omega_1|\mathbf{x}) \underset{\omega_2}{\overset{\omega_1}{\gtrless}} P(\omega_2|\mathbf{x})$$

- On cherche la fonction f telle que :

$$f(\mathbf{x}) = \log\left(\frac{P(\omega_1|\mathbf{x})}{P(\omega_2|\mathbf{x})}\right) = \log(\exp(\mathbf{w}^\top \mathbf{x} + b)) = \mathbf{w}^\top \mathbf{x} + b$$

dont le signe permet d'obtenir la décision du rapport de vraisemblance.

Problème d'optimisation

Log-Vraisemblance

On cherche à maximiser la log-vraisemblance sur les données et donc à minimiser :

$$J(\mathbf{w}, b) = -\log\left(\prod_i P(y_i|\mathbf{x}_i)\right) = -\sum_{i\in\mathcal{I}_1}\log(P(\omega_1|\mathbf{x}_i)) - \sum_{i\in\mathcal{I}_2}\log(P(\omega_2|\mathbf{x}_i)) \quad (4)$$

où $\mathcal{I}_1$ et $\mathcal{I}_2$ sont les ensembles des élément de la classe ω_1 et ω_2 respectivement.

Fonction de coût

On obtient la fonction de coût suivante :

$$\begin{aligned} J(\mathbf{w}, b) &= \sum_{i\in\mathcal{I}_1}\log(1+\exp(-\mathbf{w}^\top\mathbf{x}_i - b)) + \sum_{i\in\mathcal{I}_2}\log(1+\exp(\mathbf{w}^\top\mathbf{x}_i + b)) \\ &= \sum_i \log(1+\exp(-y_i(\mathbf{w}^\top\mathbf{x}_i + b))) \end{aligned} \quad (5)$$

Calcul du gradient

- La fonction $J(\mathbf{w}, b)$ est convexe et différentiable. On va donc calculer son gradient. Après avoir reformulé le problème sous la forme :

$$J(\boldsymbol{\alpha}) = \sum_i \log(1+\exp(-y_i\boldsymbol{\alpha}^\top\tilde{\mathbf{x}}_i)) \quad (6)$$

- La dérivée partielle de $J(\boldsymbol{\alpha})$ par rapport à $\boldsymbol{\alpha}_j$ est

$$\frac{\partial J(\boldsymbol{\alpha})}{\partial \boldsymbol{\alpha}_j} = \sum_i \frac{-y_i(\tilde{\mathbf{x}}_i)_j \exp(-y_i\boldsymbol{\alpha}^\top\tilde{\mathbf{x}}_i)}{1+\exp(-y_i\boldsymbol{\alpha}^\top\tilde{\mathbf{x}}_i)} = \sum_i \frac{-y_i(\tilde{\mathbf{x}}_i)_j p_i}{1+p_i} \quad (7)$$

  avec $p_i = \exp(-y_i\boldsymbol{\alpha}^\top\tilde{\mathbf{x}}_i)$

- Ce qui nous permet d'obtenir le gradient :

$$\nabla_\alpha J(\boldsymbol{\alpha}) = -\mathbf{X}^\top\mathbf{P}\mathbf{y} \quad (8)$$

  où $\mathbf{P}$ est une matrice diagonale contenant les valeurs $\frac{p_i}{1+p_i}$ qui dépendent de $\boldsymbol{\alpha}$. $\nabla_\alpha J(\boldsymbol{\alpha}) = \mathbf{0}$ définit des équations non-linéaires qui ne peuvent être résolues directement $\rightarrow$ Méthode d'optimisation itérative.

Méthode du gradient

Méthodes itérative

- Basées sur la mise à jour d'une solution à chaque itération.
- A l'itération t la solution est mise à jour par :

$$\boldsymbol{\alpha}^{(t)} = \boldsymbol{\alpha}^{(t-1)} + \mu_t\mathbf{d}_t \quad (9)$$

  où $\mathbf{d}_t$ est une direction de descente c'est à dire $\mathbf{d}_t^\top\nabla_\alpha J(\boldsymbol{\alpha}^{(t-1)}) < 0$ et $\mu_t > 0$ est le pas de la descente.

Méthode du gradient

- On choisit $\mathbf{d}_t = -\nabla_\alpha J(\boldsymbol{\alpha}^{(t-1)})$, qui est une direction de descente.
- Le pas μ_t doit être choisi pour assurer une descente suffisante de la fonction.
- Méthode où chaque itération est efficace mais qui converge lentement.

Méthode du gradient (2)

Algorithme du gradient

Initialisation de $\boldsymbol{\alpha}$, μ et $\mathbf{P} = \mathbf{I}$
repeat
  for $i = 1, \ldots, n$ **do**
    $p_i \leftarrow \exp(-y_i\boldsymbol{\alpha}^\top\tilde{\mathbf{x}}_i)$
    $P_{i,i} \leftarrow \frac{p_i}{1+p_i}$
  end for
  $\mathbf{d} \leftarrow \mathbf{X}^\top\mathbf{P}\mathbf{y}$
  $\boldsymbol{\alpha} \leftarrow \boldsymbol{\alpha} + \mu\mathbf{d}$
until Convergence

Discussion

- Sensible à l'initialisation de $\boldsymbol{\alpha}$.
- On peut aussi s'assurer de la décroissance du coût en ajoutant une étape de recherche linéaire :

Méthode de backtracking
Initialisation de μ et $0 < \rho < 1$.
repeat
  $\mu \leftarrow \rho\mu$
until $J(\boldsymbol{\alpha} + \mu\mathbf{d}) < J(\boldsymbol{\alpha})$

- Les condition d'arrêt et la convergence sont discutés dans la suite.

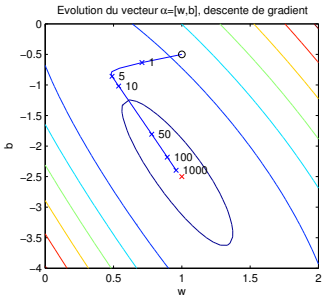
Exemple de descente de gradient

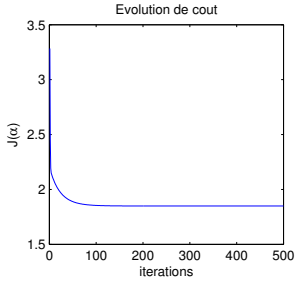
Simulation

- Régression logistique régularisée.
- Descente de gradient.
- Données (x_i, y_i) avec $d = 1$: $(1, -1), (2, -1), (3, 1), (4, 1)$
- $\mu = 0.1, \lambda = 1$
- 1000 itérations
- Initialisation $\boldsymbol{\alpha}_0 = [1, -0.5]$
- Solution du système : $\boldsymbol{\alpha}^* = [1, -2.5]$

Discussion

- Convergence lente vers la solution du système.
- Après 1000 itérations, toujours pas convergé.
- Complexité $\mathcal{O}(nd)$ par itération.





Matrice Hessienne

- La matrice Hessienne est la matrice symétrique $\mathbf{H} \in \mathbb{R}^{d+1 \times d+1}$ telle que

$$H_{u,v} = \frac{\partial^2 J(\boldsymbol{\alpha})}{\partial \boldsymbol{\alpha}_u \partial \boldsymbol{\alpha}_v} \tag{10}$$

  Elle contient les dérivées secondes d'une fonction à plusieurs variables.
- Pour la régression logistique on a

$$\frac{\partial^2 J(\boldsymbol{\alpha})}{\partial \boldsymbol{\alpha}_u \partial \boldsymbol{\alpha}_v} = \sum_i \frac{(\tilde{\mathbf{x}}_i)_u (\tilde{\mathbf{x}}_i)_v \exp(-y_i \boldsymbol{\alpha}^\top \tilde{\mathbf{x}}_i)}{(1 + \exp(-y_i \boldsymbol{\alpha}^\top \tilde{\mathbf{x}}_i))^2} = \sum_i \frac{(\tilde{\mathbf{x}}_i)_u (\tilde{\mathbf{x}}_i)_v p_i}{(1 + p_i)^2} \tag{11}$$

- Ce qui nous donne en matriciel :

$$\mathbf{H} = \mathbf{X}^\top \tilde{\mathbf{P}} \mathbf{X} \tag{12}$$

  avec $\tilde{\mathbf{P}}$ une matrice diagonale contenant $\frac{p_i}{(1+p_i)^2}$ sur chaque élément de sa diagonale.

Méthode de Newton

Principe

- Approximation quadratique $\tilde{J}(\boldsymbol{\alpha})$ de la fonction $J(\boldsymbol{\alpha})$ à chaque itération.
- La minimisation revient à prendre $\mathbf{d} = -\mathbf{H}^{-1} \nabla_\alpha J(\boldsymbol{\alpha}^{(t-1)})$
- Si la fonction à optimiser est convexe, $\mathbf{H}$ est définie positive et $\mathbf{d}$ est donc une direction de descente.

Algorithme de Newton

Initialisation de $\boldsymbol{\alpha}$, μ et $\mathbf{P} = \mathbf{I}$ et $\tilde{\mathbf{P}} = \mathbf{I}$
repeat
  Mise à jour de $\mathbf{P}$ et $\tilde{\mathbf{P}}$
  $\mathbf{d} \leftarrow (\mathbf{X}^\top \tilde{\mathbf{P}} \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{P} \mathbf{y}$
  $\boldsymbol{\alpha} \leftarrow \boldsymbol{\alpha} + \mu \mathbf{d}$
until Convergence

Discussion

- Meilleure vitesse de convergence.
- Nécessite le calcul et l'inversion de la Hessienne.
- Itération plus complexe que la méthode du gradient.
- Si le problème est quadratique, l'algorithme converge en une seule itération.

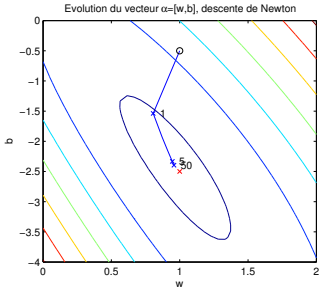
Exemple de descente de Newton

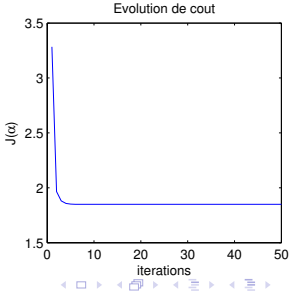
Simulation

- Régression logistique régularisée.
- Descente de gradient.
- Données (x_i, y_i) avec $d = 1$: $(1, -1), (2, -1), (3, 1), (4, 1)$
- $\mu = 0.1, \lambda = 1$
- 1000 itérations
- Initialisation $\boldsymbol{\alpha}_0 = [1, -0.5]$
- Solution du système : $\boldsymbol{\alpha}^* = [1, -2.5]$

Discussion

- Convergence Rapide vers la solution du système.
- Après 5 itérations, même position que 100 en descente de gradient.
- Complexité $\mathcal{O}(nd^2 + d^3)$ par itération.





Conditions d'arrêt des algorithmes

Convergence

- L'utilisation de méthodes itérative soulève le problème de convergence.
- La convergence est démontrée dans certains cas (Nocedal, Convex Optimization)
- Un point stationnaire est atteint par l'algorithme si :

$$\nabla_\alpha J(\boldsymbol{\alpha}) = \mathbf{0} \tag{13}$$

Conditions d'arrêt

En pratique on arrête les itérations selon une des conditions suivantes :

- Norme du gradient inférieure à un seuil : $\|\nabla_\alpha J(\boldsymbol{\alpha})\| < \epsilon$
- Variation relative du coût entre deux itérations inférieure à un seuil : $\frac{|J(\boldsymbol{\alpha}^t) - J(\boldsymbol{\alpha}^{t-1})|}{J(\boldsymbol{\alpha}^{t-1})|} < \epsilon$
- Nombre maximum d'itérations t_{max} atteint.

Régularisation

A priori sur w

- Un *a priori* $P(\mathbf{w})$ sur la loi de probabilité de $\mathbf{w}$ peut être facilement ajouté à la log-vraisemblance.
- Si on suppose que $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \sigma\mathbf{I})$ alors on a

$$P(\mathbf{w}) = e^{-\frac{\|\mathbf{w}\|^2}{2\sigma}} \tag{14}$$

- Et minimiser la log-vraisemblance revient donc à minimiser :

$$\begin{aligned} J(\mathbf{w}, b) &= -\log\left(P(\mathbf{w})\prod_i P(y_i|\mathbf{x}_i)\right) = -\log\left(P(\mathbf{w})\right) - \log\left(\prod_i P(y_i|\mathbf{x}_i)\right) \\ &= \sum_i \log(1+\exp(-y_i\boldsymbol{\alpha}^\top\tilde{\mathbf{x}}_i)) + \frac{1}{2\sigma}\|\mathbf{w}\|^2 \end{aligned} \tag{15}$$

  On reconnaît un régularisation de type ridge avec $\lambda = \frac{1}{\sigma}$
- Il est possible d'avoir d'autres *a priori* qui mèneront à des régularisations différentes.

Conclusions sur la régression logistique

Avantages

- Modélisation probabiliste. La probabilité d'appartenir à chaque classe peut être obtenue.
- Problème convexe, il existe une solution unique.
- Possibilité de régulariser pour éviter le sur-apprentissage.
- Beaucoup moins de paramètres à estimer que la discrimination bayesienne

$$d+1 < \underbrace{2d+2}_{\text{(Cas 1)}} \ll \underbrace{3d+2}_{\text{(Cas 2)}} \ll \underbrace{d^2+2d+2}_{\text{(Cas 3)}}$$

Inconvénient

- Problème difficile à optimiser (non linéaire).
- Méthodes itératives nécessaires (gradient ou Newton).

Méthode du perceptron

Historique

- Le perceptron a été proposé en 1957 par Frank Rosenblatt.
- Tout premier réseau de neurone (linéaire).
- Capable d'apprendre uniquement sur des données séparables.

Principe

- On cherche un hyperplan définit par $f(\mathbf{x}) = \mathbf{w}^\top\mathbf{x} + b = 0$ séparant les données d'apprentissage.
- Méthode itérative.
- Mise à jour $(\mathbf{w}, b)$ par rapport aux exemples mal classés.
- Arrêt des itérations quand tous les exemples sont bien classifiés.

Problème d'optimisation

Fonction de Coût

La méthode du perceptron consiste à minimiser :

$$J(\boldsymbol{\alpha}) = J(\mathbf{w}, b) = -\sum_{i \in \mathcal{M}} y_i(\mathbf{x}_i^\top \mathbf{w} + b) = \sum_i \max(0, -y_i(\mathbf{x}_i^\top \mathbf{w} + b)) \qquad (16)$$

où $\mathcal{M}$ est l'ensemble des exemples mal classés.

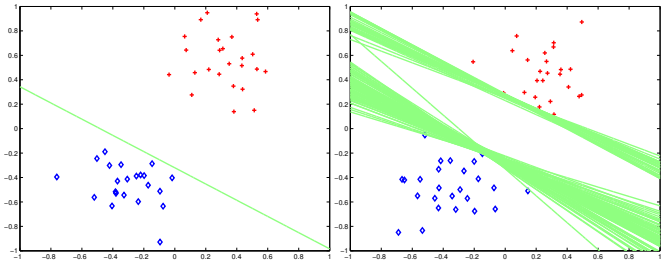
Algorithme du perceptron

Initialisation de $\boldsymbol{\alpha}$ et $\mu > 0$
repeat
  for $i \in \mathcal{I}$ **do**
    if $y_i \tilde{\mathbf{x}}_i^\top \boldsymbol{\alpha} < 0$ **then**
      $\boldsymbol{\alpha} \leftarrow \boldsymbol{\alpha} + \mu y_i \tilde{\mathbf{x}}_i$
    end if
  end for
until $y_i \tilde{\mathbf{x}}_i^\top \boldsymbol{\alpha} \geq 0, \quad \forall i$

Discussion

- $\mathcal{I}$ définit l'ordre dans lequel les exemples sont parcourus.
- Chaque itération fait appel au gradient d'un exemple et pas de l'ensemble.
- Optimisation de type "gradient stochastique".
- Converge en un nombre fini d'itérations si données séparables.

Exemple de perceptron



Discussion

- Chaque droite verte correspond à une solution obtenue avec
- La solution finale dépend de l'initialisation et de l'ordre dans lequel les exemples ont été présentés.
- On remarque que les hyperplans sont souvent proche d'une des deux classes.
- Généralisation des solutions ?

Conclusion sur le perceptron

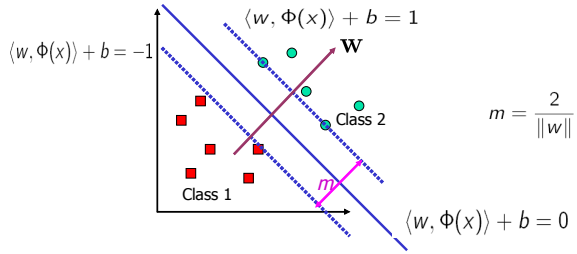
Avantage

- Méthode historique.
- Trouve une solution en un nombre fini d'itérations si données séparables.
- Itérations très peu coûteuses en temps de calcul.

Inconvénients

- Pas de solution unique.
- Pas de convergence si les données sont non séparables.
- Risque de sur-apprentissage car pas de régularisation.
- Mauvaises performances dans le cas multiclasse.

Séparateur à Vaste Marge



Principe

- Trouver l'hyperplan qui maximise la marge entre les classes.
- On veut promouvoir une marge entre les classes on a donc les contraintes suivantes :

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

Problème d'optimisation

- La distance d'un exemple à l'hyperplan est

$$d(\mathbf{x}) = \frac{|\mathbf{w}^\top \mathbf{x} + b|}{\|\mathbf{w}\|}$$

- Les contraintes $y_i(\mathbf{w}^\top \mathbf{x} + b) \geq 1$ nous assurent que la distance minimale des exemples à l'hyperplan est $\frac{1}{\|\mathbf{w}\|}$. La marge est donc égale à

$$m = \frac{2}{\|\mathbf{w}\|} \tag{17}$$

- Maximiser la marge revient donc à minimiser $\|\mathbf{w}\|$ et donc $\|\mathbf{w}\|^2$.
- Le problème d'optimisation des séparateurs à vaste marge sur des données séparables est donc :

$$\begin{aligned} \min_{\mathbf{w},b} \quad & \|\mathbf{w}\|^2 \\ & y_i(\mathbf{w}^\top \mathbf{x} + b) \geq 1 \quad \forall i \end{aligned} \tag{18}$$

Méthodes d'optimisation

Reformulation pour des données non séparables

Le problème d'optimisation devient :

$$\min_{\mathbf{w},b} \quad \sum_{i=1}^{n} \max(0, 1 - y_i(\mathbf{w}^\top \mathbf{x}_i + b)) + \frac{\lambda}{2}\|\mathbf{w}\|^2 \tag{19}$$

Optimisation

Plusieurs approches d'optimisation possibles.

- Passage dans le dual en calculant le Lagrangien.
  $\rightarrow$ Problème quadratique sous contrainte de taille $n + 1$
- Résolution directe avec une approche de type descente de gradient.
  $\rightarrow$ Problème non-différentiable de taille $d + 1$.

Conclusion sur les SVM

Avantages

- Problème strictement convexe, une seule solution.
- Méthode consistante qui converge vers la décision de Bayes quand le nombre d'exemples tend vers l'infini.
- Possibilité d'extension à des problèmes non-linéaire en passant dans le dual.
- Très bonnes performances en pratique.

Inconvénients

- Validation du paramètre λ.
- Fonction de coût non différentiable utilisation de méthodes de sous-gradient.

Prédiction linéaire

Problème d'apprentissage général :

$$\min_{\mathbf{w},b} \quad \sum_{i=1}^{n} L(y_i, \mathbf{w}^\top \mathbf{x}_i + b) + \lambda \Omega(\mathbf{w}) \tag{20}$$

Avec

- $L(\cdots)$ la fonction de coût d'attache aux données.
- $\Omega(\cdot)$ le terme de régularisation.

Exemples :

Fonctions de coût $L(y, \hat{y})$

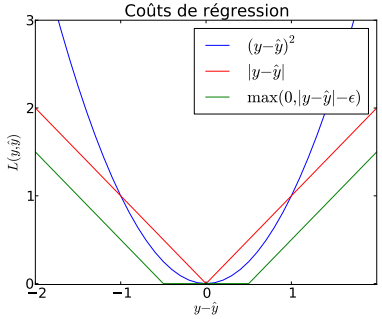
- $(y - \hat{y})^2$, quadratique.
- $|y - \hat{y}|$, valeur absolue.
- $\max(0, 1 - y\hat{y})$, charnière.
- $\log(1 + e^{-y\hat{y}})$, logistique.

Régularisations $\Omega(\mathbf{w})$

- $\|\mathbf{w}\|_2^2$, quadratique.
- $\|\mathbf{w}\|_1$, norme ℓ_1.
- $\mathbf{w}^\top \mathbf{\Sigma} \mathbf{w}$, Mahalanobis.

Attache aux données de Régression

Coût	$L(y, \hat{y})$	Rég.	Cvx.
Carré	$(y - \hat{y})^2$	✓	✓
Valeur absolue	$\lvert y - \hat{y} \rvert$	-	✓
ϵ insensible	$\max(0, \lvert y - \hat{y} \rvert - \epsilon)$	-	✓

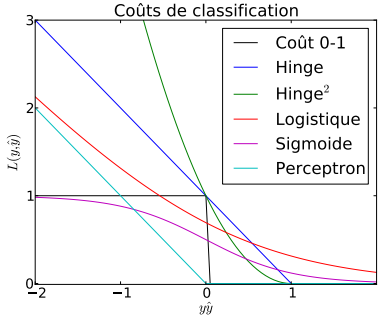


Problème de régression

- **Objectif** : prédire une valeur réelle.
- Erreur si $y \neq \hat{y}$.
- **Mesure d'erreur** : $|y - \hat{y}|$

Attache aux données de Classification

Coût	$L(y, \hat{y})$	Rég.	Cvx.
Coût 0-1	$(1 - \text{sgn}(y\hat{y}))/2$	-	-
Hinge	$\max(0, 1 - y\hat{y})$	-	✓
Hinge au carré	$\max(0, 1 - y\hat{y})^2$	✓	✓
Logistique	$\log(1 + \exp(-y\hat{y}))$	✓	✓
Sigmoïde	$(1 - \tanh(y\hat{y}))/2$	✓	-
Perceptron	$\max(0, -y\hat{y})$	-	✓



Problème de régression

- **Objectif** : prédire une valeur binaire.
- Erreur si $y \neq \text{signe}(\hat{y})$ ou si y et $\hat{y}$ sont de signe différents.
- **Mesure d'erreur** : $y\hat{y}$