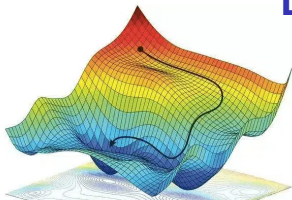


Modèles Linéaires

Descente de gradient

Perceptron



Cours 2
ML Master DAC

Nicolas Baskiotis

`nicolas.baskiotis@sorbonne-universite.fr`

équipe MLIA, Institut des Systèmes Intelligents et de Robotique (ISIR)
Sorbonne Université

S2 (2023-2024)

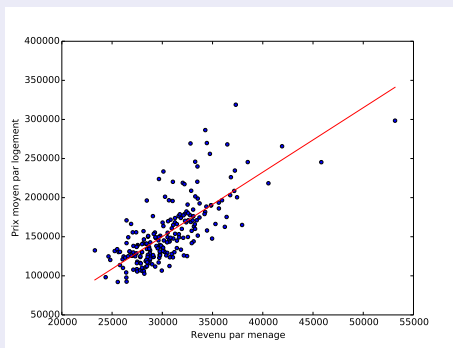
Plan

- 1 Régression linéaire
- 2 Régression logistique
- 3 Descente de gradient
- 4 Interlude
- 5 Perceptron
- 6 Sur la linéarité des modèles

Introduction

Régression linéaire

- Objectif : prédire une sortie continue réelle y à partir d'un nombre de variables d'entrée
- beaucoup d'applications, très utilisée un peu dans tous les domaines
- très flexible (transformation des entrées)



Formalisation

Objectif

Etant donné un ensemble $\{(\mathbf{x}^j, y^j) \in \mathbb{R}^d \times \mathbb{R}\}_{j=1}^N$, $\mathbf{x}^j = (x_1^j, x_2^j, \dots, x_d^j)$

- Hypothèse : variation linéaire de la sortie en fonction des entrées

$$\mathbb{E}[y|\mathbf{x}] = w_0 + \sum_{i=1}^d w_i x_i$$

⇒ On cherche :

- ▶ une fonction $f_{\mathbf{w}}(\mathbf{x}) = w_0 + \sum_{i=1}^d w_i x_i$
 - ▶ qui fait le moins d'erreurs : $f(\mathbf{x}^i)$ doit être proche de y^i
 - ▶ sous la condition que l'erreur est indépendante de \mathbf{x} , de variance σ^2 constante, suit une loi normale.
- ⇒ $y|\mathbf{x} \sim \mathcal{N}(f(\mathbf{x}), \sigma^2)$ (lien avec l'apprentissage bayésien)
- Mesure d'erreur : coût aux moindres carrés (Mean Squared Error - MSE)

$$\ell(f(\mathbf{x}), y) = (f(\mathbf{x}) - y)^2$$

Formalisation (2)

Objectif

- Minimiser :

$$\mathbb{E} [\ell(f(\mathbf{x}), y)] = \int_{\mathbf{x}, y} (y - f(\mathbf{x}))^2 p(\mathbf{x}, y) dx dy$$

- Soit trouver $\mathbf{w} \in \mathbb{R}^{d+1}$ qui minimise :

$$\frac{1}{N} \sum_{j=1}^N \ell(f_{\mathbf{w}}(\mathbf{x}^j), y^j) = \frac{1}{N} \sum_{j=1}^N (y^j - f(\mathbf{x}^j))^2 = \frac{1}{N} \sum_{j=1}^N (y^j - w_0 - \sum_{i=1}^d w_i x_i^j)^2$$

Régression : solution analytique

Formalisation

- Minimiser :

$$\mathbb{E}(\ell(f(\mathbf{x}), y)) = \int_{\mathbf{x}, y} (y - f(\mathbf{x}))^2 p(\mathbf{x}, y) dx dy$$

- Soit trouver $\mathbf{w} \in \mathbb{R}^{d+1}$ qui minimise :

$$L(\mathbf{w}) = \frac{1}{N} \sum_{j=1}^N \ell(f_{\mathbf{w}}(\mathbf{x}^j), y^j) = \frac{1}{N} \sum_{j=1}^N (y^j - f(\mathbf{x}^j))^2 = \frac{1}{N} \sum_{j=1}^N (y^j - w_0 - \sum_{i=1}^d w_i x_i^j)^2$$

- La fonction $L : \mathbb{R}^{d+1} \rightarrow \mathbb{R}$ est convexe
- ⇒ Solution analytique : annuler son gradient !
- ⇒ Trouver \mathbf{w}^* tq $\nabla_{\mathbf{w}} L(\mathbf{w}^*) = 0$

Dérivée matricielle

Ecriture pratique

- $X = \begin{pmatrix} x^1 \\ x^2 \\ \vdots \\ x^N \end{pmatrix} = \begin{pmatrix} x_1^1 & x_2^1 & \dots & x_d^1 \\ x_1^2 & x_2^2 & \dots & x_d^2 \\ & & \vdots & \\ x_1^N & x_2^N & \dots & x_d^N \end{pmatrix}$, $Y = \begin{pmatrix} y^1 \\ y^2 \\ \vdots \\ y^N \end{pmatrix}$, $W = \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_d \end{pmatrix}$
- $L(\mathbf{w}) = (XW - Y)'(XW - Y)$
- $\nabla_{\mathbf{w}} L = 2X'(XW - Y)$, \mathbf{w} optimal $\Leftrightarrow 2X'(XW - Y) = 0$
- Solution : $(X'X)^{-1}X'Y$
- Et pour w_0 ?

Plan

- 1 Régression linéaire
- 2 Régression logistique**
- 3 Descente de gradient
- 4 Interlude
- 5 Perceptron
- 6 Sur la linéarité des modèles

Problématique

Classification binaire

- Deux classes : $Y = \{-1, +1\}$, et un ensemble d'apprentissage $\{(\mathbf{x}^i, y^i) \in \mathbb{R}^d \times Y\}$
- Peut-on utiliser un coût quadratique dans ce cas ?

Cas 2D :

- ▶ $f_{\mathbf{w}}(\mathbf{x}) = 1 \Leftrightarrow w_0 + x_1 w_1 + w_2 w_2 = 1$
- ▶ $f_{\mathbf{w}}(\mathbf{x}) = -1 \Leftrightarrow w_0 + x_1 w_1 + w_2 w_2 = -1$

Problématique

Classification binaire

- Deux classes : $Y = \{-1, +1\}$, et un ensemble d'apprentissage $\{(\mathbf{x}^i, y^i) \in \mathbb{R}^d \times Y\}$
- Peut-on utiliser un coût quadratique dans ce cas ?

Cas 2D :

- ▶ $f_{\mathbf{w}}(\mathbf{x}) = 1 \Leftrightarrow w_0 + x_1 w_1 + w_2 w_2 = 1$
- ▶ $f_{\mathbf{w}}(\mathbf{x}) = -1 \Leftrightarrow w_0 + x_1 w_1 + w_2 w_2 = -1$

- Et si $f_{\mathbf{w}}(\mathbf{x}) \gg 1$?

⇒ le coût sera très grand !

- De même si $f_{\mathbf{w}}(\mathbf{x}) \ll -1$

⇒ Le coût n'est pas adapté à la classification !

Adaptation du formalisme

Dans chaque région de l'espace, on suppose que :

- le label $+1$ suit une loi de Bernoulli paramétrée par une fonction dépendant de \mathbf{x} :

$$p(y = 1|\mathbf{x}) = \mu(\mathbf{x})$$

- le label -1 également : $p(y = -1|\mathbf{x}) = 1 - \mu(\mathbf{x})$

$$\Rightarrow p(y|\mathbf{x}) = \mu(\mathbf{x})^{\frac{y+1}{2}} (1 - \mu(\mathbf{x}))^{\frac{1-y}{2}}$$

- Comment représenter $\mu(\mathbf{x})$? Fonction linéaire ?

Adaptation du formalisme

Dans chaque région de l'espace, on suppose que :

- le label $+1$ suit une loi de Bernoulli paramétrée par une fonction dépendant de \mathbf{x} :

$$p(y = 1|\mathbf{x}) = \mu(\mathbf{x})$$

- le label -1 également : $p(y = -1|\mathbf{x}) = 1 - \mu(\mathbf{x})$

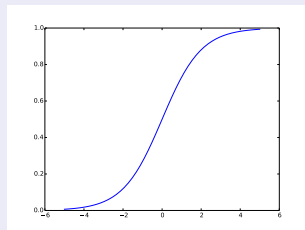
$$\Rightarrow p(y|\mathbf{x}) = \mu(\mathbf{x})^{\frac{y+1}{2}} (1 - \mu(\mathbf{x}))^{\frac{1-y}{2}}$$

- Comment représenter $\mu(\mathbf{x})$? Fonction linéaire ?

\Rightarrow **Problème ! pas entre 0 et 1 !**

- Transformation d'une fonction linéaire : la fonction sigmoïde.

$$\mu(\mathbf{x}) = \sigma(f_{\mathbf{w}}(\mathbf{x})) = \frac{1}{1 + e^{-f_{\mathbf{w}}(\mathbf{x})}}$$



Quelques remarques importantes

Que représente $f_{\mathbf{w}}(\mathbf{x})$?

- $\mu(\mathbf{x}) = \sigma(f_{\mathbf{w}}(\mathbf{x})) = \frac{1}{1+e^{-f_{\mathbf{w}}(\mathbf{x})}}$, $1 - \mu(\mathbf{x}) = \frac{e^{-f_{\mathbf{w}}(\mathbf{x})}}{1+e^{-f_{\mathbf{w}}(\mathbf{x})}} = \frac{1}{1+e^{f_{\mathbf{w}}(\mathbf{x})}} = \sigma(-f_{\mathbf{w}}(\mathbf{x}))$
- C'est le log-ratio des probabilités : $f_{\mathbf{w}}(\mathbf{x}) = \log\left(\frac{\mu(\mathbf{x})}{1-\mu(\mathbf{x})}\right) = \log\left(\frac{p(y=+1|\mathbf{x})}{p(y=-1|\mathbf{x})}\right)$
- qui est approximée par une fonction linéaire :
$$\log\frac{P(y=+1|\mathbf{x})}{P(y=-1|\mathbf{x})} = w_0 + w_1x_1 + w_2x_2\dots$$
- Quand est-ce que :
 - ▶ $p(y = +1|\mathbf{x}) = 0.5$?
 - ▶ $p(y = +1|\mathbf{x}) < 0.5$?
 - ▶ $p(y = +1|\mathbf{x}) > 0.5$?

Résolution

Maximum de vraisemblance

On cherche à maximiser :

$$P(y^1, \dots, y^N | \mathbf{x}^1, \dots, \mathbf{x}^N) = \prod_{i=1}^N P(y^i | \mathbf{x}^i)$$

$$\Leftrightarrow \text{maximiser } \log \prod_{i=1}^N P(y^i | \mathbf{x}^i)$$

$$\Leftrightarrow \text{maximiser } \sum_{i=1}^N \log P(y^i | \mathbf{x}^i)$$

$$\Leftrightarrow \text{minimiser } \sum_{i=1}^N \log \frac{1}{P(y^i | \mathbf{x}^i)}$$

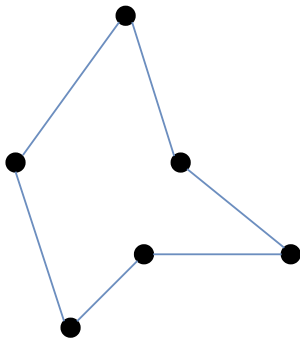
$$\Leftrightarrow \text{minimiser } \sum_{i=1}^N \log \frac{1}{\sigma(-y^i f_{\mathbf{w}}(\mathbf{x}^i))}$$

$$\Leftrightarrow \text{On cherche } \mathbf{w}^* = \operatorname{argmin}_{\mathbf{w}} \sum_{i=1}^N \log [1 + \exp(-y^i f_{\mathbf{w}}(\mathbf{x}^i))]$$

Résolution

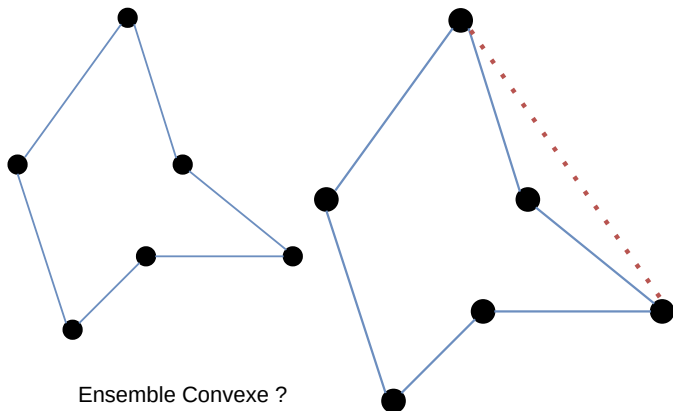
- Pas de solution analytique.
- Méthode d'optimisation numérique \rightarrow descente de gradient.

Interlude : convexité



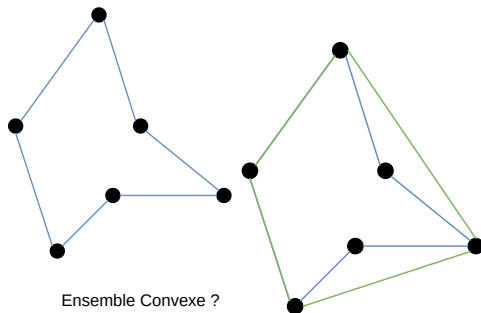
Ensemble Convexe ?

Interlude : convexité



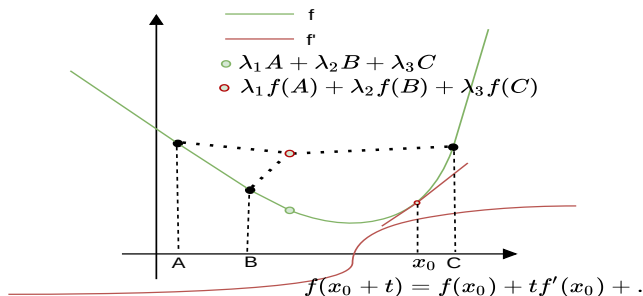
- C ensemble convexe de \mathbb{R}^n : $\forall x, y \in C, \forall \lambda \in [0, 1], \lambda x + (1 - \lambda)y \in C$
- $\sum_i \lambda_i x_i$ est une combinaison convexe ssi $\forall i, \lambda_i \geq 0$ et $\sum_i \lambda_i = 1$
- Enveloppe convexe d'un ensemble fini $\{x_i\}, i = 1 \dots n$: toutes les combinaisons convexes de l'ensemble

Interlude : convexité



- C ensemble convexe de \mathbb{R}^n : $\forall x, y \in C, \forall \lambda \in [0, 1], \lambda x + (1 - \lambda)y \in C$
- $\sum_i \lambda_i x_i$ est une combinaison convexe ssi $\forall i, \lambda_i \geq 0$ et $\sum_i \lambda_i = 1$
- Enveloppe convexe d'un ensemble fini $\{x_i\}, i = 1 \dots n$: toutes les combinaisons convexes de l'ensemble

Interlude : convexité



Notations et rappels

- Fonction $f : X \rightarrow \mathbb{R}$ convexe ssi

$$\forall x, x' \in X, \forall \lambda \in [0, 1] \text{ tq } \lambda x + (1 - \lambda)x' \in X$$

$$\text{alors } f(\lambda x + (1 - \lambda)x') \leq \lambda f(x) + (1 - \lambda)f(x')$$

- si $\lambda_i \geq 0$ et $\sum_i \lambda_i = 1$, alors $f(\sum_i \lambda_i x_i) \leq \sum_i \lambda_i f(x_i)$ (inégalité de Jensen)

Interlude : convexité

Différentiabilité

- Si $f : X \rightarrow \mathbb{R}$ est convexe ssi $\forall x, x' \in X, f(x') \geq f(x) + \langle x' - x, \nabla f(x) \rangle$
- Si f convexe, alors sa matrice hessienne est définie semi-positive : $\nabla^2 f \geq 0$.

Minimum

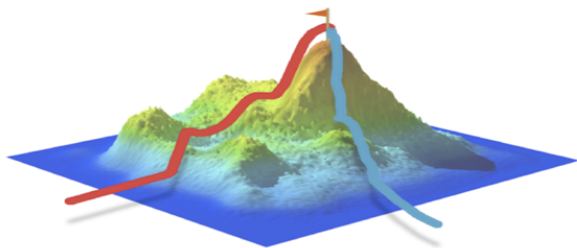
- Si f atteint son minimum, alors les minimums forment un ensemble convexe.
- Si l'ensemble est strictement convexe, le minimum est un singleton.
- Si f est strictement convexe, son gradient ne s'annule que à son minimum local.

Plan

- 1 Régression linéaire
- 2 Régression logistique
- 3 Descente de gradient**
- 4 Interlude
- 5 Perceptron
- 6 Sur la linéarité des modèles

Principe

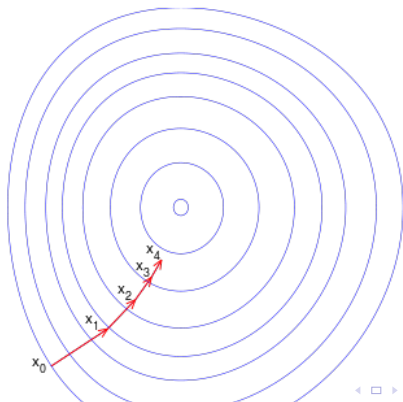
- Algorithme d'optimisation différentiable
- S'applique pour toute fonction différentiable
- Idée simple : améliorer de façon itérative la solution courante



Algorithme du gradient

Algorithme

- 1 Choisir un point x_0
- 2 Itérer :
 - ▶ Calculer $\nabla f(x_t)$
 - ▶ mettre à jour $x_{t+1} \leftarrow x_t - \alpha \nabla f(x_t)$



Pourquoi cela fonctionne ?

Développement de Taylor

- $f(\mathbf{x}) = f(\mathbf{x}_1) + \nabla f(\mathbf{x}_1) \times (\mathbf{x} - \mathbf{x}_1) + O(\|\mathbf{x} - \mathbf{x}_1\|^2)$
- On cherche à “bouger” dans une direction \mathbf{u} de façon à minimiser f :

$$f(\mathbf{x}_1 + h\mathbf{u}) - f(\mathbf{x}_1) = h\nabla f(\mathbf{x}_1)\mathbf{u} + h^2O(1)$$

- On doit donc minimiser $\nabla f(\mathbf{x}_1)\mathbf{u}$.
- On choisit le vecteur unité $\mathbf{u} = -\frac{\nabla f(\mathbf{x}_1)}{\|\nabla f(\mathbf{x}_1)\|}$

Plusieurs variantes :

- Hill climbing dans le cas discret
- Coordinate descent (line search selon les dimensions)
- Conjugate Gradient

Importance du pas de gradient

Algorithme

- 1 Choisir un point x_0
- 2 Itérer :
 - ▶ Calculer $\nabla f(x_t)$
 - ▶ mettre à jour $x_{t+1} \leftarrow x_t - \alpha \nabla f(x_t)$

Remarques

- Que se passe-t-il si :
 - ▶ α est choisi trop grand ?
 - ▶ trop petit ?
- Est-ce que l'on atteint toujours un minimum global ?
- Application à la régression logistique ?

Variantes de l'algorithme

- Cas hors-ligne (ou batch) :
Pour chaque époque (correction de w), on itère sur toute la base d'exemples
- Cas en-ligne (stochastique) :
Une correction de w est faite par rapport à un exemple tiré au hasard dans la base.
- hybride : mini-batch
Des petits sous-ensembles d'exemples sont tirés au hasard, la correction se fait selon le gradient calculé sur ces exemples.

Avantages et inconvénients ?

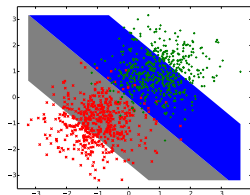
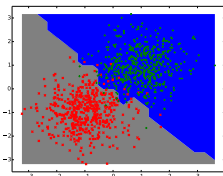
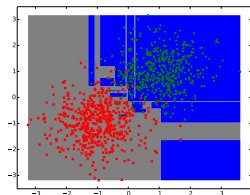
- Batch : plus stable, plus rapide
- Stochastique : bien meilleur tolérance au bruit !

Plan

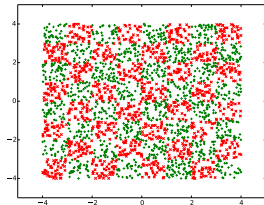
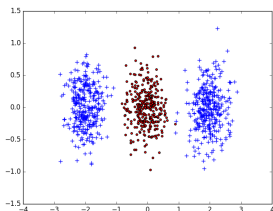
- 1 Régression linéaire
- 2 Régression logistique
- 3 Descente de gradient
- 4 Interlude**
- 5 Perceptron
- 6 Sur la linéarité des modèles

On réfléchit un peu

Quelle approche correspond à quelle frontière ?



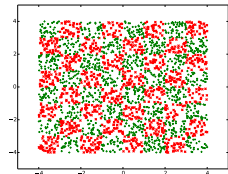
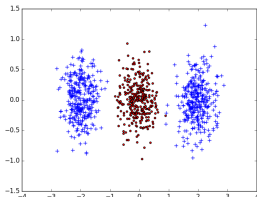
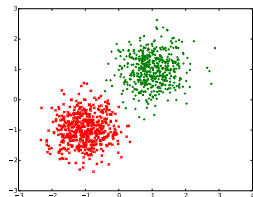
Et pour ces cas ?



Espace linéairement séparable

Linéairement séparable

Non linéairement séparable



Conclusion (temporaire)

Importance :

- de l'espace de fonctions considéré (choisi a priori)
 - du paramétrage des algorithmes
- notion d'expressivité ...

à suivre.

Plan

- 1 Régression linéaire
- 2 Régression logistique
- 3 Descente de gradient
- 4 Interlude
- 5 Perceptron**
- 6 Sur la linéarité des modèles

Historique

Prémisses

- McCulloch et Pitts (1943) : 1er modèle de neurone formel. Base de l'IA
- Règle de Hebb (1949) : apprentissage par renforcement du couplage synaptique

Premières réalisations

- Adaline (Widrow-Hoff, 1960)
- Perceptron (Rosenblatt, 1958-1962)
- Analyse de Minsky et Papert (1969)

Développement

- Réseau bouclé (Hopfield 1982)
- Réseau multi-couches (1985)

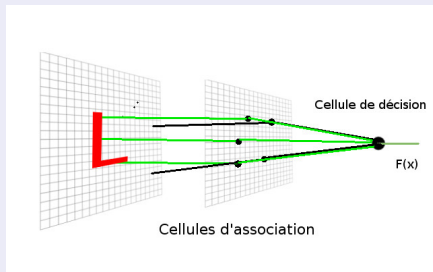
Deuxième renaissance

- Réseaux profonds (2000-)

Le perceptron de Rosenblatt (1960)

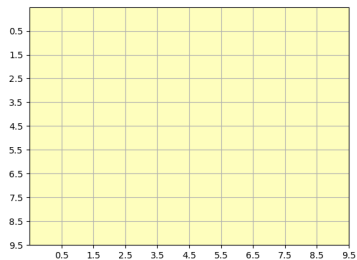
L'idée

- Reconnaissance de forme (*pattern*) entre deux classes
- Inspirée du cortex visuel



- Chaque cellule d'association produit une sortie $x_i(S)$ en fonction d'un stimulus
- La cellule de décision répond selon une fonction seuil :
$$\sum w_i x_i(S) > \theta \Rightarrow +1 \text{ sinon } -1$$

Intuition de l'apprentissage



Etat initial des poids

Intuition de l'apprentissage

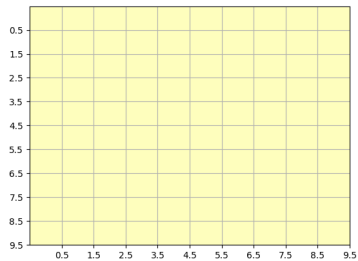
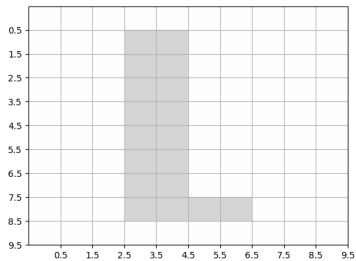


Image 1, Poids initiaux

Intuition de l'apprentissage

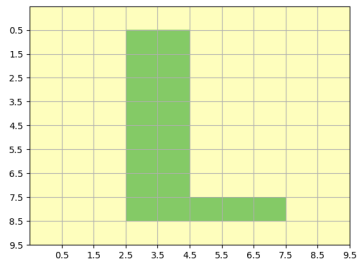
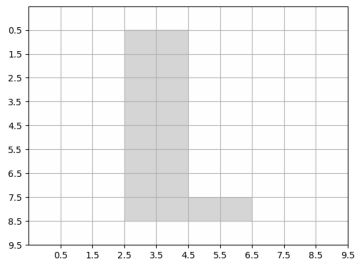


Image 1, Poids 1

Intuition de l'apprentissage

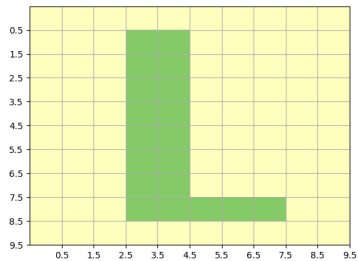
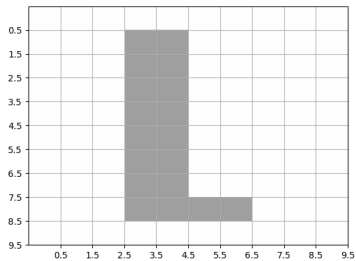


Image 2, Poids 1

Intuition de l'apprentissage

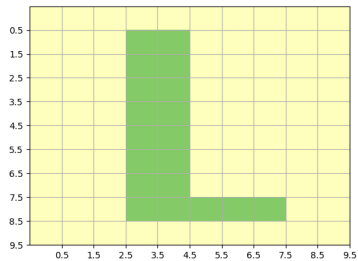
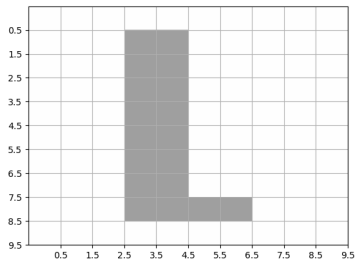


Image 2, Poids 2

Intuition de l'apprentissage

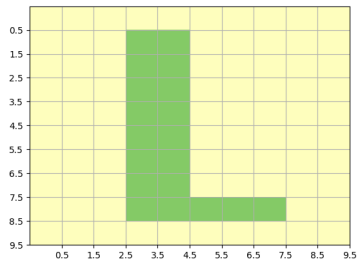
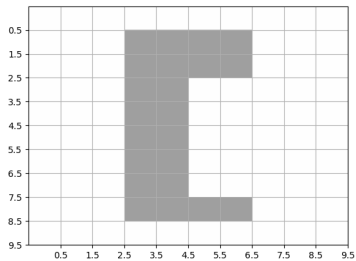


Image 3, Poids 2

Intuition de l'apprentissage

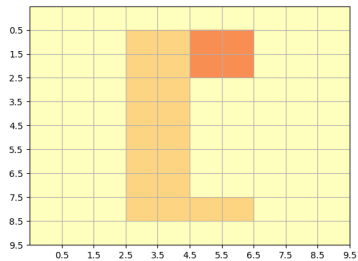
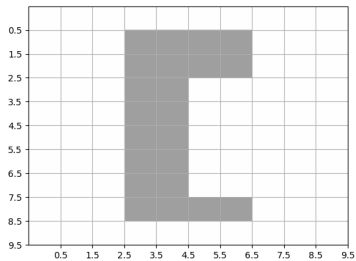


Image 3, Poids 3

Intuition de l'apprentissage

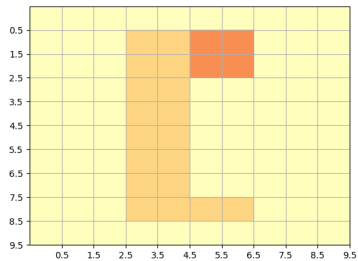
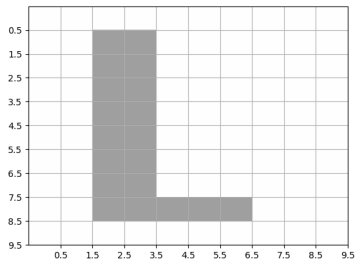


Image 4, Poids 3

Intuition de l'apprentissage

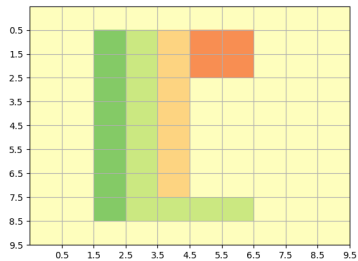
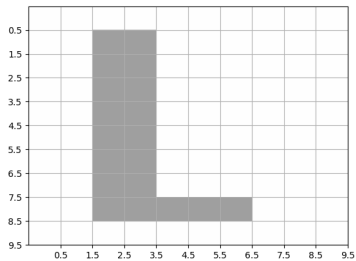
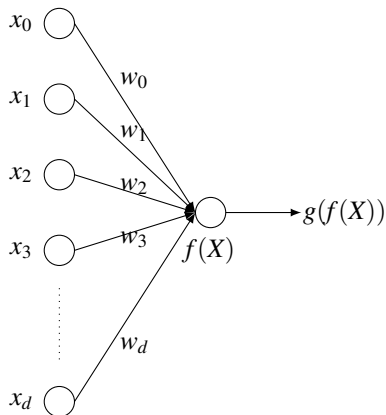


Image 4, Poids 4

Formalisation



Le perceptron considère

- $f(\mathbf{x}) = \sum_{i=1}^d x_i w_i = \langle \mathbf{x}, \mathbf{w} \rangle$
- Fonction de décision :
 $g(x) = \text{sign}(x)$
- Sortie : $g(f(\mathbf{x})) = \text{sign}(\langle \mathbf{x}, \mathbf{w} \rangle)$

Algorithme d'apprentissage

Algorithme du perceptron

- Initialiser au hasard \mathbf{w}
- Tant qu'il n'y a pas convergence :
 - ▶ pour tous les exemples (x^i, y^i) :
 - ★ si $(y^i \times \langle \mathbf{w}, \mathbf{x}^i \rangle) < 0$ alors $\mathbf{w} = \mathbf{w} + \epsilon y^i \mathbf{x}^i$
- Décision : $f(x) = \text{sign}(\langle \mathbf{w}, \mathbf{x} \rangle)$

Autre formulation

A quoi correspond la règle de mise à jour :

- Si $(y < \mathbf{w} \cdot \mathbf{x} >) > 0$ ne rien faire
- Si $(y < \mathbf{w} \cdot \mathbf{x} >) < 0$ corriger $\mathbf{w} = \mathbf{w} + y\mathbf{x}$?

Autre formulation

A quoi correspond la règle de mise à jour :

- Si $(y < \mathbf{w} \cdot \mathbf{x} >) > 0$ ne rien faire
- Si $(y < \mathbf{w} \cdot \mathbf{x} >) < 0$ corriger $\mathbf{w} = \mathbf{w} + y\mathbf{x}$?

Hinge loss

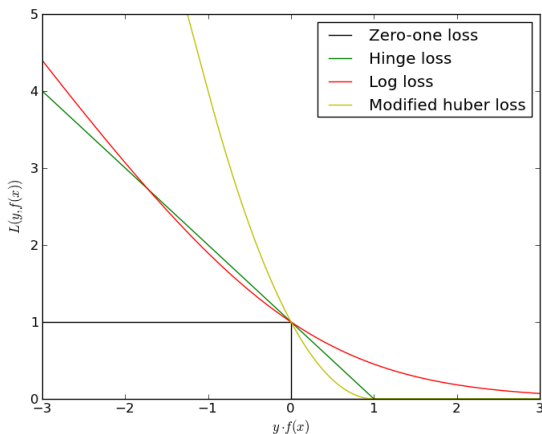
$l(f(x), y) = \max(0, \alpha - yf(x))$ avec $\alpha = 0$

Descente de gradient !

- $l(f_{\mathbf{w}}(x), y) = \max(0, \alpha - y < \mathbf{w} \cdot \mathbf{x} >)$
- $\nabla_{\mathbf{w}} l(f_{\mathbf{w}}(x), y) = \begin{cases} 0 & \text{si } (y < \mathbf{w} \cdot \mathbf{x} >) > \alpha \\ -yx_i & \text{sinon} \end{cases}$

À quoi sert ce α ?

Et pourquoi pas d'autres erreurs ?



Théorème de convergence (Novikov, 1962)

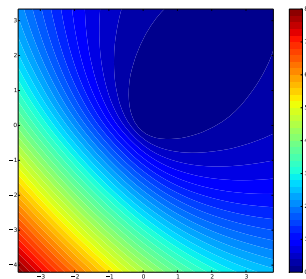
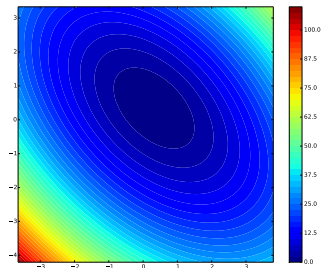
- Si
 - ▶ $\exists R, \forall x : \|x\| \leq R$
 - ▶ les données peuvent être séparées avec une marge ρ
 - ▶ l'ensemble d'apprentissage est présenté au perceptron un nombre suffisant de fois
- alors après au plus R^2/ρ^2 corrections, l'algorithme converge.

Exploration de l'espace des solutions

Vision duale de l'espace des exemples

- $R_{hinge}(f_{\mathbf{w}}) = \mathbb{E}(l_{hinge}(f_{\mathbf{w}}(x), y)) = \mathbb{E}(\max(0, -f_{\mathbf{w}}(x)y))$
- $R_{mse}(f_{\mathbf{w}}) = \mathbb{E}(l_{mse}(f_{\mathbf{w}}(x), y)) = \mathbb{E}((f_{\mathbf{w}}(x) - y)^2)$

Pour un ensemble fixé de données, le risque est vu comme une fonction de \mathbf{w} .



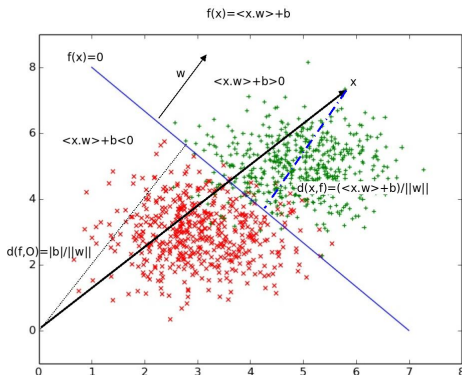
Plan

- 1 Régression linéaire
- 2 Régression logistique
- 3 Descente de gradient
- 4 Interlude
- 5 Perceptron
- 6 Sur la linéarité des modèles**

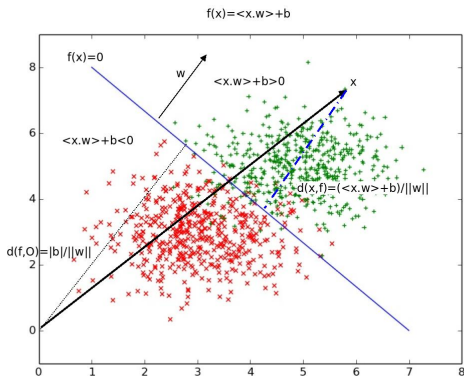
Considérations géométriques

Soit y la sortie attendue :

- Que représente w par rapport à la séparatrice ?
- Que représente $\langle w, x \rangle$?
- Que représente $y \langle w, x \rangle$?
- A quoi correspond la règle de mise à jour :
 - ▶ Si $(y \langle w, x \rangle) > 0$ ne rien faire
 - ▶ Si $(y \langle w, x \rangle) < 0$ corriger $w = w + yx$?



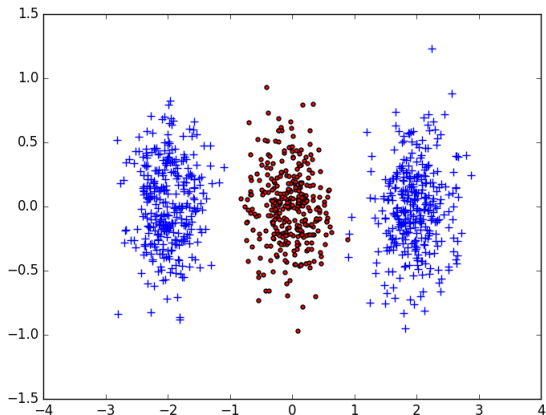
Considérations géométriques



Questions

- Solution unique ?
- Certaines solutions meilleures que d'autres ?

Données non séparables linéairement

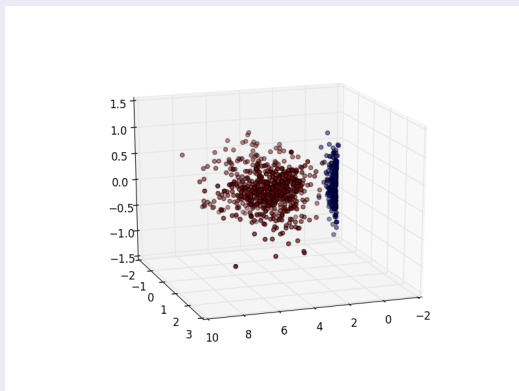


Que faire ?

Données non séparables linéairement

Transformation de la représentation

- On augmente d'une dimension : $(x_1, x_2) \rightarrow (x_1^2, x_1x_2, x_2)$



- Le problème est de nouveau séparable linéairement !
- Autre solution ?

Projection des données

Exprimer les données dans un espace de plus grande dimension

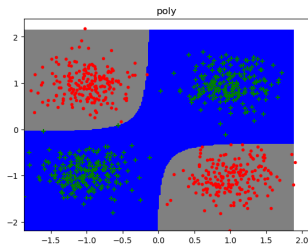
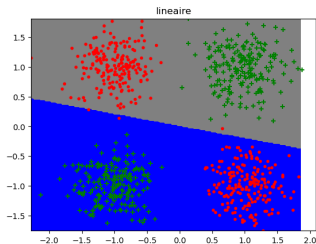
- On considère une fonction $\phi(\mathbf{x}) : \mathbb{R}^d \rightarrow \mathbb{R}^{d'}$, généralement $d' \gg d$.
- Nouvel ensemble d'apprentissage : $\{(\mathbf{z}^k, y^k)\}_{k=1}^N$ avec $\mathbf{z}^i = \phi(\mathbf{x}^i)$
- Modèle linéaire : $\sum_{i=1}^{d'} w_i z_i = \sum_{i=1}^{d'} w_i \phi(x_i)$
- Même algo d'optimisation : descente de gradient avec le coût perceptron

Exemple : Projection polynomiale

On considère $\mathbf{x} \in \mathbb{R}^3$, pour une projection de degré 2 :

- $\phi(\mathbf{x}) = (x_1, x_2, x_3, x_1x_2, x_1x_3, x_2x_3, x_1^2, x_2^2, x_3^2) \in \mathbb{R}^9$
- Nouvelle forme de la frontière de décision :
 $w_1x_1 + w_2x_2 + w_3x_3 + w_4x_1x_2 + w_5x_1x_3 + w_6x_2x_3 + w_7x_1^2 + w_8x_2^2 + w_9x_3^2 = 0$

Projection des données



Exemple : Projection polynomiale

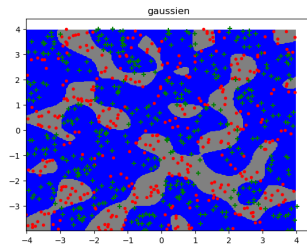
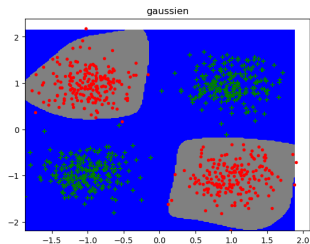
On considère $\mathbf{x} \in \mathbb{R}^3$, pour une projection de degré 2 :

- $\phi(\mathbf{x}) = (x_1, x_2, x_3, x_1x_2, x_1x_3, x_2x_3, x_1^2, x_2^2, x_3^2) \in \mathbb{R}^9$

- Nouvelle forme de la frontière de décision :

$$w_1x_1 + w_2x_2 + w_3x_3 + w_4x_1x_2 + w_5x_1x_3 + w_6x_2x_3 + w_7x_1^2 + w_8x_2^2 + w_9x_3^2 = 0$$

Projection des données



Exemple : Projection Gaussienne sur une base

On considère une base $\{\mathbf{x}^{ij}\}_{j=1}^M$ de point dans \mathbb{R}^d , la projection d'un exemple \mathbf{x} est constitué des M distances gaussiennes aux points \mathbf{x}^{ij} de la base :

- $\phi(\mathbf{x}) = (e^{-\|\mathbf{x}-\mathbf{x}'^1\|^2/2\sigma}, e^{-\|\mathbf{x}-\mathbf{x}'^2\|^2/2\sigma}, \dots, e^{-\|\mathbf{x}-\mathbf{x}'^M\|^2/2\sigma})$
- Nouvelle forme de la frontière de décision :
 $w_1 e^{-\|\mathbf{x}-\mathbf{x}'^1\|^2/2\sigma} + w_2 e^{-\|\mathbf{x}-\mathbf{x}'^2\|^2/2\sigma} + \dots + w_M e^{-\|\mathbf{x}-\mathbf{x}'^M\|^2/2\sigma} = 0$
- Seuls les points \mathbf{x}' proche de \mathbf{x} jouent un rôle (et donc seuls les poids associés).
- Très proche d'un k -NN avec prise en compte de la distance, **mais** les poids sont appris automatiquement !