Techniques d'apprentissage

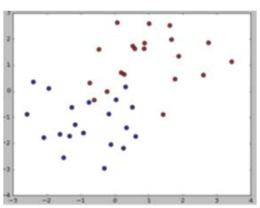
IFT 603-712

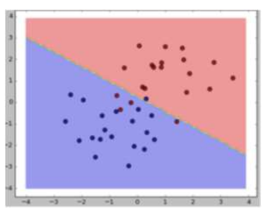
Classification linéaire

Par
Pierre-Marc Jodoin
/
Hugo Larochelle

Classification supervisée (illustrée)

Entraînement





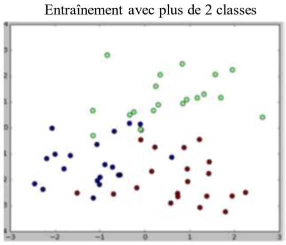
Soient des données de 2 classes ● et ●
(ici dans un espace 2D)

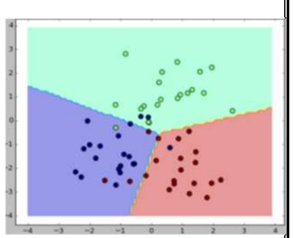
Le but est de trouver une function $y_{\vec{w}}(\vec{x})$ telle que

$y_{\vec{w}}(●) =$ classe 1
$y_{\vec{w}}(●) =$ classe 2

Classification supervisée (illustrée)

Entraînement avec plus de 2 classes





Soient des données de 3 classes ●,○et ●
(ici dans un espace 2D)

Le but est de trouver une function $y_{\vec{w}}(\vec{x})$ telle que

$y_{\vec{w}}(●) =$ classe 1
$y_{\vec{w}}(●) =$ classe 2
$y_{\vec{w}}(○) =$ classe 3

Notation

Ensemble d'entraînement: $D = \{(\vec{x}_1, t_1), (\vec{x}_2, t_2), \ldots, (\vec{x}_N, t_N)\}$

$\vec{x}_n \in \Re^d$ vecteur de données du n-ème élement
$t_n \in \{c_1, c_2, \ldots, c_K\}$ étiquette de classe du i-ème élément

Fonctions: avec *D*, on doit *apprendre* une **fonction de classification**

$$y_{\vec{w}}(\vec{x}): \Re^d \rightarrow \{c_1, c_1, \ldots, c_k\}$$

qui nous informe à quelle classe appartient le vecteur $\vec{x}$.

5

Au menu : 5 méthodes

Régression
Modèles génératifs
Discriminant de Fisher

Émettent **l'hypothèse** que les données sont **gaussiennes**
Solution de type « ***closed form*** » (inversion de matrice)

Perceptron
Régression logistique

Aucune hypothèse quant à la distribution des données
Solution obtenue grâce à une **descente de gradient**.

6

Introduction à la classification linéaire

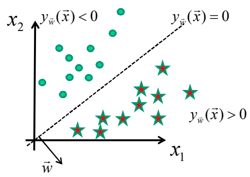
Au tableau !!!

7

Séparation linéaire

(2D et 2 classes)

$$\begin{aligned} y_{\vec w}(\vec x) &= \overset{\text{biais}}{w_0} + \overset{\text{poids}}{w_1 x_1 + w_2 x_2} \\ &= w_0 + \vec w^{\mathrm T} \vec x \\ &= \vec w'^{\mathrm T} \vec x' \end{aligned}$$



$$y_{\vec w}(\vec x) = \vec w^{\mathrm T} \vec x$$

Par simplicité

2 grands **advantages**. Une fois l'entraînement terminé,

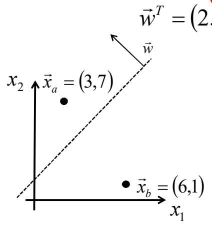
1. Plus besoin de données d'entraînement
2. Classification est très rapide (**produit scalaire** entre 2 vecteurs)

8

Séparation linéaire

Exemple jouet

$$\vec w^T = (\overset{\text{biais}}{2.2}, \overset{\text{poids}}{-5.5, 4.4})$$



$\vec x_a$ **est en FACE du plan**

$$y_{\vec w}(\vec x_a) = \vec w^T \vec x_a = (2.2, -5.5, 4.4)\begin{bmatrix}1\\3\\7\end{bmatrix} = 16.5$$

$$y_{\vec w}(\vec x_b) = \vec w^T \vec x_b = (2.2, -5.5, 4.4)\begin{bmatrix}1\\6\\1\end{bmatrix} = -26.4$$

$\vec x_b$ **est DERRIÈRE le plan**

9

10

Régression
Modèles génératifs
Discriminant de Fisher

Émettent l'hypothèque que les données sont **gaussiennes**

Solution de type « ***closed form*** » (inversion de matrice)

Perceptron
Régression logistique

Émettent **aucune hypothèse** quant à la distribution des données

Solution obtenue grâce à une **descente de gradient**.

Régression par les moindres carrés
(section 4.1.3, Bishop)

Régression par les moindres carrés
Cas 2 classes

On peut **classifier des données** en utilisant une approche de **régression** comme celle vue au chapitre précédent.

- On pourrait **prédire directement** la valeur de la cible (*t=1.0* vs *t=-1.0*)
- Si $y_w(\vec{x}) \geq 0$ on classifie dans *Classe1* sinon dans *Classe2*

Régression par moindres carrés

RAPPEL

On a vu qu'on peut utiliser une fonction d'erreur **par moindres carrés**

Maximum de vraisemblance

$$\vec{w} = \arg\min_{\vec{w}} \underbrace{\sum_{n=1}^{N} \left(t_n - y_{\vec{w}}(\vec{x}_n)\right)^2}_{E_D(\vec{w})}$$

Maximum *a posteriori*

$$\vec{w} = \arg\min_{\vec{w}} \underbrace{\sum_{n=1}^{N} \left(t_n - y_{\vec{w}}(\vec{x}_n)\right)^2 + \lambda \vec{w}^{\mathrm{T}} \vec{w}}_{E_D(\vec{w})}$$

On peut prendre la même approche pour la classification

14

Régression par les moindres carrés

Cas 2 classes

RAPPEL

Maximum de vraisemblance

$$\vec{w} = \arg\min_{\vec{w}} \sum_{n=1}^{N} \left(t_n - y_{\vec{w}}(\vec{x}_n)\right)^2$$

$$\boxed{\vec{w}_{\mathrm{MV}} = \left(X^{\mathrm{T}} X\right)^{-1} X^{\mathrm{T}} T}$$

Maximum *a posteriori*

$$\vec{w} = \arg\min_{\vec{w}} \sum_{n=1}^{N} \left(t_n - y_{\vec{w}}(\vec{x}_n)\right)^2 + \lambda \vec{w}^{\mathrm{T}} \vec{w}$$

$$\boxed{\vec{w}_{\mathrm{MAP}} = \left(X^{\mathrm{T}} X + \lambda I\right)^{-1} X^{\mathrm{T}} T}$$

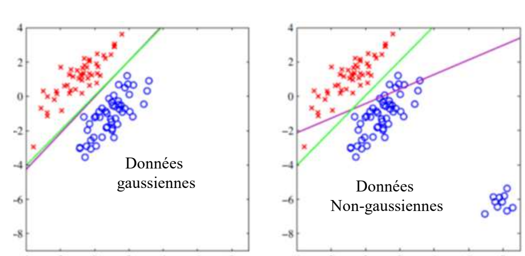
Ces fonctions de coût s'appuient sur l'hypothèse de données gaussiennes

15

Régression par moindres carrés

Régression logistique (à voir plus loin)

Moindres carrés (maximum de vraisemblance)



16

Régression par les moindres carrés

Cas K>2 classes

On va traiter le cas K classes comme une **régression multiple**

- **Cible :** vecteur à K dim. indiquant a quelle classe appartient l'entrée
- **Exemple** : Pour K=5 classes et un entrée associée à la classe 2

$$t_n = \begin{pmatrix} -1 & 1 & -1 & -1 & -1 \end{pmatrix}^{\mathrm{T}}$$

- **Classification**: On classifie dans la classe k une donnée dont la valeur de $y_{\vec{w},k}(\vec{x})$ est la plus élevée.

Régression par les moindres carrés

Cas K>2 classes

Le modèle doit maintenant **prédire un vecteur**

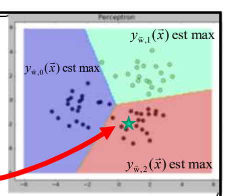
$$y_{\mathrm{W}}(\vec{x}) = \mathrm{W}^{\mathrm{T}}\vec{x}$$

où W est une matrice ***K* x *d***

Chaque ligne de W peut être vue comme un vecteur $\vec{\mathrm{w}}_k$ du modèle $y_{\vec{\mathrm{w}}_k}(\vec{x}) = \vec{\mathrm{w}}_k^T\vec{x}$ pour la k^e cible

Cas K=3 classes

Example



(1.1, -2.0)

$$y_{\mathrm{W}}(\vec{x}) = \mathrm{W}\vec{x} = \begin{bmatrix} -.6 & -.49 & .4 \\ -.4 & .41 & .24 \\ -.2 & .05 & -.36 \end{bmatrix} \begin{bmatrix} 1 \\ 1.1 \\ -2.0 \end{bmatrix} = \begin{array}{cl} \begin{bmatrix} -1.9 \\ -.43 \\ .58 \end{bmatrix} & \begin{array}{l} \text{Classe 0} \\ \text{Classe 1} \\ \text{Classe 2} \end{array} \end{array}$$

Régression
Modèles génératifs
Discriminant de Fisher

Émettent l'hypothèque que les données sont **gaussiennes**

Solution de type « ***closed form*** » (inversion de matrice)

Perceptron
Régression logistique

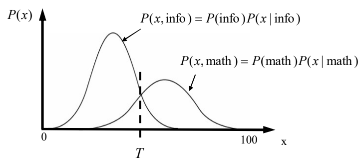
Émettent aucune hypothèse quant à la distribution des données

Solution obtenue grâce à une **descente de gradient**.

Modèles probabilistes génératifs
(section 4.2, Bishop)

Prenons le cas 1D, 2 Classes

Ex: examen de mathématique avec étudiants en math et en informatique

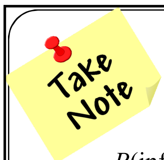


T est le seuil qui **minimise l'erreur de classification**

$$P(\text{info})P(x = T\,|\,\text{info}) = P(\text{math})P(x = T\,|\,\text{math})$$

$$\boxed{P(\text{info})P(x\,|\,\text{info}) \underset{\text{math}}{\overset{\text{info}}{\gtrless}} P(\text{math})P(x\,|\,\text{math})}$$

23



$$P(\text{info})P(x\,|\,\text{info}) \underset{\text{math}}{\overset{\text{info}}{\gtrless}} P(\text{math})P(x\,|\,\text{math})$$

est équivalent à un **maximum a posteriori**

Inconnue — Connue

$$t = \arg\max_t P\left(t\,|\,x\right) \qquad \text{où } t \in \{\text{math,info}\}$$
$$= (\cdots)$$
$$= \arg\max_t P(t)P\left(x\,|\,t\right)$$

24

Prenons le cas 1D, 2 Classes

Ex: examen de math avec étudiant en math et en informatique

$$P(\text{info})P(x\,|\,\text{info}) \underset{\text{math}}{\overset{\text{info}}{\gtrless}} P(\text{math})P(x\,|\,\text{math})$$

Si on suppose que la **vraisemblance** de chaque classe est **gaussienne**:

$$P(x\,|\,\text{info}) = \frac{1}{\sqrt{2\pi}\sigma_{\text{info}}}\exp\left(-\frac{(x-\mu_{\text{info}})^2}{2\sigma_{\text{info}}^2}\right)$$

$$P(x\,|\,\text{math}) = \frac{1}{\sqrt{2\pi}\sigma_{\text{math}}}\exp\left(-\frac{(x-\mu_{\text{math}})^2}{2\sigma_{\text{math}}^2}\right)$$

où

μ_{math} : moyenne des étudiants de math.

σ_{math} : écart-type des étudiants de math.

25

Prenons le cas 1D, 2 Classes

Ex: examen de math avec étudiant en math et en informatique

$$P(\text{info})P(x\,|\,\text{info}) \underset{\text{math}}{\overset{\text{info}}{\gtrless}} P(\text{math})P(x\,|\,\text{math})$$

Si on suppose que la **vraisemblance** de chaque classe est **gaussienne**:

$$P(x\,|\,\text{info}) = \frac{1}{\sqrt{2\pi}\sigma_{\text{info}}}\exp\left(-\frac{(x-\mu_{\text{info}})^2}{2\sigma_{\text{info}}^2}\right)$$

$$P(x\,|\,\text{math}) = \frac{1}{\sqrt{2\pi}\sigma_{\text{math}}}\exp\left(-\frac{(x-\mu_{\text{math}})^2}{2\sigma_{\text{math}}^2}\right)$$

et que

$$P(\text{info}) = \frac{\text{nb étudiants info}}{\text{nb tot étudiants}}$$ (Proportion des étudiants en info)

$$P(\text{math}) = \frac{\text{nb étudiants math}}{\text{nb tot étudiants}}$$ (Proportion des étudiants en math)

26

Modèle probabiliste génératif

Algorithme du seuil « optimal »

$$\mu_{\text{info}} = \frac{1}{N_{\text{info}}}\sum_{t_n=\text{info}} x_n,\quad \mu_{\text{math}} = \frac{1}{N_{\text{math}}}\sum_{t_n=\text{math}} x_n$$

$$\sigma_{\text{info}}^2 = \frac{1}{N_{\text{info}}}\sum_{t_n=\text{info}} (x_n-\mu_{\text{info}})^2,\quad \sigma_{\text{math}}^2 = \frac{1}{N_{\text{math}}}\sum_{t_n=\text{math}} (x_n-\mu_{\text{math}})^2$$

$$P(\text{math}) = \frac{N_{\text{math}}}{N_{\text{info}}+N_{\text{math}}},\quad P(\text{info}) = \frac{N_{\text{info}}}{N_{\text{info}}+N_{\text{math}}}$$

POUR CHAQUE note x FAIRE

$$P_i = \frac{P(\text{info})}{\sqrt{2\pi}\sigma_{\text{info}}}\exp\left(-\frac{(x-\mu_{\text{info}})^2}{2\sigma_{\text{info}}^2}\right)$$

$$P_m = \frac{P(\text{math})}{\sqrt{2\pi}\sigma_{\text{math}}}\exp\left(-\frac{(x-\mu_{\text{math}})^2}{2\sigma_{\text{math}}^2}\right)$$

SI $P_i > P_m$ ALORS

$t = 1$ /* étudiant « info » */

SINON

$t = 0$ /* étudiant « math » */

27

L'algorithme de la page précédente revient à un **classificateur quadratique**

$$y_{\vec{w}}(x) = w_2 x^2 + w_1 x + w_0 = 0$$

28

Modèle probabiliste génératif

Classificateur quadratique, cas 1D, 2 Classes

$$P(\text{info})P(x \mid \text{info}) = P(\text{math})P(x \mid \text{math})$$

$$\frac{P(\text{info})}{\sqrt{2\pi}\sigma_{\text{info}}}\exp\left(-\frac{(x-\mu_{\text{info}})^2}{2\sigma_{\text{info}}^2}\right) = \frac{P(\text{math})}{\sqrt{2\pi}\sigma_{\text{math}}}\exp\left(-\frac{(x-\mu_{\text{math}})^2}{2\sigma_{\text{math}}^2}\right)$$

On peut facilement démontrer que

$$\boxed{y_{\vec{w}}(x) = w_2 x^2 + w_1 x + w_0 = 0}$$

$$w_2 = \frac{\sigma_{\text{math}}^2 - \sigma_{\text{info}}^2}{2}$$

$$w_1 = \mu_{\text{math}}\sigma_{\text{info}}^2 - \mu_{\text{info}}\sigma_{\text{math}}^2$$

$$w_0 = \frac{\mu_{\text{info}}^2\sigma_{\text{math}}^2}{2} - \frac{\mu_{\text{math}}^2\sigma_{\text{info}}^2}{2} - \sigma_{\text{info}}^2\sigma_{\text{math}}^2 \ln\left(\frac{\sigma_{\text{math}}P(\text{info})}{\sigma_{\text{info}}P(\text{math})}\right)$$

Modèle probabiliste génératif

Classificateur linéaire, cas 1D, 2 Classes

Si on suppose que $\sigma_{\text{info}} = \sigma_{\text{math}} = \sigma$

$$P(\text{info})P(x \mid \text{info}) = P(\text{math})P(x \mid \text{math})$$

$$\frac{P(\text{info})}{\sqrt{2\pi}\sigma}\exp\left(-\frac{(x-\mu_{\text{info}})^2}{2\sigma^2}\right) = \frac{P(\text{math})}{\sqrt{2\pi}\sigma}\exp\left(-\frac{(x-\mu_{\text{math}})^2}{2\sigma^2}\right)$$

$$\boxed{y_{\vec{w}}(x) = w_1 x + w_0 = 0}$$

$$w_1 = \frac{(\mu_{\text{math}} - \mu_{\text{info}})}{\sigma^2}$$

$$w_0 = \frac{\mu_{\text{info}}^2}{2\sigma^2} - \frac{\mu_{\text{math}}^2}{2\sigma^2} - \ln\left(\frac{P(\text{info})}{P(\text{math})}\right)$$

Modèle probabiliste génératif

Classificateur linéaire, **cas d-D, 2 Classes**

$$\boxed{y_{\vec{w}}(\vec{x}) = \vec{w}^{\mathrm{T}}\vec{x} + w_0 = 0}$$

$$\vec{w} = \Sigma^{-1}(\vec{\mu}_1 - \vec{\mu}_2)$$

$$w_0 = \frac{\vec{\mu}_2^{\mathrm{T}}\Sigma^{-1}\vec{\mu}_2}{2} - \frac{\vec{\mu}_1^{\mathrm{T}}\Sigma^{-1}\vec{\mu}_1}{2} - \ln\left(\frac{P(\mathrm{C}_2)}{P(\mathrm{C}_1)}\right)$$

Tel que mentionné au chapitre 4.2.2, lorsque les 2 classes n'ont pas la même variance-covariance, on peut utiliser le modèle linéaire mais avec la matrice

$$\Sigma = P(\mathrm{C}_1)\Sigma_1 + P(\mathrm{C}_2)\Sigma_2$$

32

Modèle probabiliste génératif

Classificateur linéaire, cas 1D, 2 Classes

Maximum de vraisemblance

Si on suppose que $P(\text{info}) = P(\text{math})$

$$\cancel{P(\text{info})}P(x \mid \text{info}) = \cancel{P(\text{math})}P(x \mid \text{math})$$

$$\frac{\cancel{P(\text{info})}}{\sqrt{2\pi}\sigma}\exp\left(-\frac{(x-\mu_{\text{info}})^2}{2\sigma^2}\right) = \frac{\cancel{P(\text{math})}}{\sqrt{2\pi}\sigma}\exp\left(-\frac{(x-\mu_{\text{math}})^2}{2\sigma^2}\right)$$

$$\boxed{y_{\vec{\mathrm{w}}}(x) = w_1 x + w_0 = 0}$$

$$w_1 = \frac{(\mu_{\text{math}} - \mu_{\text{info}})}{\sigma^2}$$

$$w_0 = \frac{\mu_{\text{info}}^2}{2\sigma^2} - \frac{\mu_{\text{math}}^2}{2\sigma^2} - \ln\left(\cancel{\frac{P(\text{info})}{P(\text{math})}}\right)$$

33

Modèle probabiliste génératif

Classificateur linéaire, cas d-D, K Classes

On peut généraliser au cas à **plusieurs classes**

- Voir fin des sections 4.2 et 4.2.1

34

Régression
Modèles génératifs
Discriminant de Fisher

Émettent l'hypothèque que les données sont **gaussiennes**

Solution de type « ***closed form*** » (inversion de matrice)

Perceptron
Régression logistique

Émettent **aucune hypothèse** quant à la distribution des données

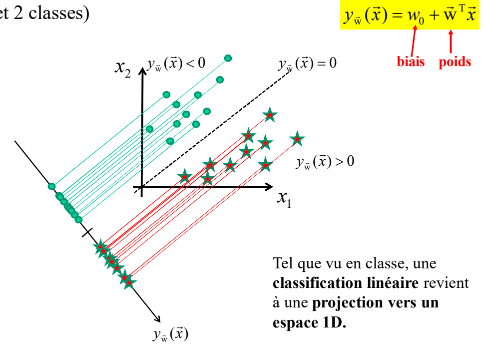
Solution obtenue grâce à une **descente de gradient**.

Discriminant linéaire de Fisher
(section 4.1.4, Bishop)

Classification linéaire = Projection 1D

(2D et 2 classes)

$y_{\vec{w}}(\vec{x}) = w_0 + \vec{w}^{\mathrm{T}}\vec{x}$

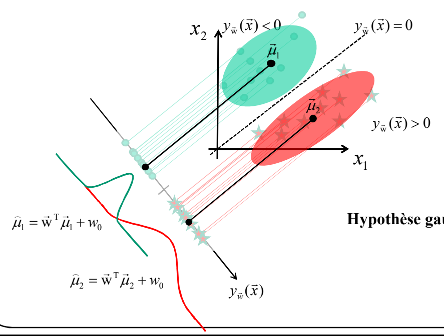


Tel que vu en classe, une **classification linéaire** revient à une **projection vers un espace 1D.**

Classification linéaire = Projection 1D

(2D et 2 classes)

$y_{\vec{w}}(\vec{x}) = w_0 + \vec{w}^{\mathrm{T}}\vec{x}$

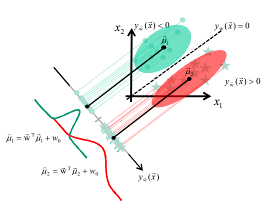


Hypothèse gaussienne…

Classification linéaire = Projection 1D

(2D et 2 classes)

$y_{\vec{w}}(\vec{x}) = w_0 + \vec{w}^{\mathrm{T}}\vec{x}$



Intuitivement, une bonne solution $y_{\vec{w}}(\vec{x})$ en est une pour laquelle la **distance entre les moyennes projetées est grande.**

$$\begin{aligned}\vec{w} &= \arg\max_{\vec{w}} \left|\tilde{\mu}_1 - \tilde{\mu}_2\right| \\ &= \arg\max_{\vec{w}} \left|\vec{w}^{\mathrm{T}}\vec{\mu}_1 + w_0 - \vec{w}^{\mathrm{T}}\vec{\mu}_2 - w_0\right| \\ &= \arg\max_{\vec{w}} \left|\vec{w}^{\mathrm{T}}\left(\vec{\mu}_1 - \vec{\mu}_2\right)\right|\end{aligned}$$

Classification linéaire = Projection 1D

(2D et 2 classes)

$y_{\vec{w}}(\vec{x}) = w_0 + \vec{w}^{\mathrm{T}}\vec{x}$

$$\vec{w} = \arg\max_{\vec{w}} \left| \vec{w}^{\mathrm{T}} (\vec{\mu}_1 - \vec{\mu}_2) \right|$$

Ce **problème est mal posé** car il suffit d'augmenter **W** infiniment pour maximiser cette fonction.

41

Classification linéaire = Projection 1D

(2D et 2 classes)

$y_{\vec{w}}(\vec{x}) = w_0 + \vec{w}^{\mathrm{T}}\vec{x}$

$$\vec{w} = \arg\max_{\vec{w}} \left| \vec{w}^{\mathrm{T}} (\vec{\mu}_1 - \vec{\mu}_2) \right|$$

Par contre si on impose que la **norme de *w* = 1**

on obtient que

$$\vec{w} \propto (\vec{\mu}_1 - \vec{\mu}_2)$$

(preuve au tableau)

42

Discriminant linéaire

Une fois **w** calculé, il faut trouver le biais w_0

- Un choix fréquent lorsque les classes sont balancées

$$w_0 = -\frac{\vec{w}^{\mathrm{T}}\vec{\mu}_1 + \vec{w}^{\mathrm{T}}\vec{\mu}_2}{2}$$

- Sinon

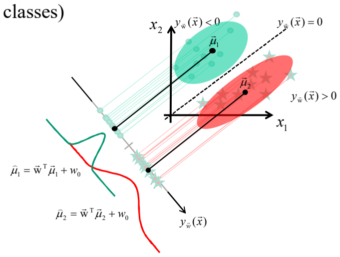
$$w_0 = -\vec{w}^{\mathrm{T}} \left(\frac{N_1}{N_1 + N_2} \vec{\mu}_1 + \frac{N_2}{N_1 + N_2} \vec{\mu}_2 \right)$$

où N1 et N2 sont le nombre d'éléments dans chaque classe.

43

Discriminant linéaire

(2D et 2 classes)



Les 2 gaussiennes projetées:

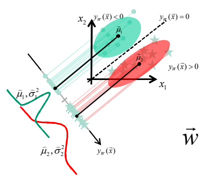
$$\hat{\mu}_1 = \vec{w}^{\mathrm{T}}\vec{\mu}_1 + w_0 \qquad \hat{\mu}_2 = \vec{w}^{\mathrm{T}}\vec{\mu}_2 + w_0$$

$$\hat{\sigma}_1^2 = \vec{w}^{\mathrm{T}}\Sigma_1\vec{w} \qquad \hat{\sigma}_2^2 = \vec{w}^{\mathrm{T}}\Sigma_2\vec{w}$$

44

Discriminant linéaire **de Fisher**

(2D et 2 classes)



Critère de Fisher

$$\vec{w} = \arg\max_{\vec{w}} \frac{(\hat{\mu}_1 - \hat{\mu}_2)^2}{\hat{\sigma}_1^2 + \hat{\sigma}_2^2}$$

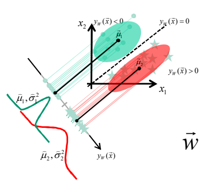
On obtient le meilleur W en forçant à 0

$$\nabla_{\vec{w}} \frac{(\hat{\mu}_1 - \hat{\mu}_2)^2}{\hat{\sigma}_1^2 + \hat{\sigma}_2^2} = 0$$

45

Discriminant linéaire **de Fisher**

(2D et 2 classes)



Critère de Fisher

$$\vec{w} = \arg\max_{\vec{w}} \frac{(\hat{\mu}_1 - \hat{\mu}_2)^2}{\hat{\sigma}_1^2 + \hat{\sigma}_2^2}$$

Preuve en classe ou en devoir

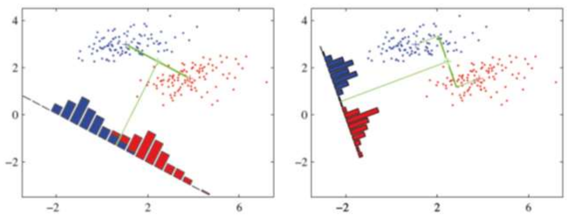
$$\Rightarrow \vec{w} \propto \Sigma_{\vec{w}}^{-1}(\vec{\mu}_1 - \vec{\mu}_2)$$

$$\Rightarrow \Sigma_{\vec{w}} = \sum_{t_n=C_1}(\vec{x}_n - \vec{\mu}_1)(\vec{x}_n - \vec{\mu}_1)^{\mathrm{T}} + \sum_{t_n=C_2}(\vec{x}_n - \vec{\mu}_2)(\vec{x}_n - \vec{\mu}_2)^{\mathrm{T}}$$

46

Discriminant linéaire de Fisher

(2D et 2 classes)



Sans minimisation intra-classe

$$\vec{w} \propto (\vec{\mu}_1 - \vec{\mu}_2)$$

Avec minimisation intra-classe

$$\vec{w} \propto \Sigma_{\vec{w}}^{-1}(\vec{\mu}_1 - \vec{\mu}_2)$$

47

Discriminant linéaire de Fisher

Algorithme 2-Classes, **entraînement**

Calculer $\vec{\mu}_1, \vec{\mu}_2$

$$\Sigma_{\vec{w}} = \sum_{t_n = C_1} (\vec{x}_n - \vec{\mu}_1)(\vec{x}_n - \vec{\mu}_1)^{\mathrm{T}} + \sum_{t_n = C_2} (\vec{x}_n - \vec{\mu}_2)(\vec{x}_n - \vec{\mu}_2)^{\mathrm{T}}$$

$$\vec{w} = \Sigma_{\vec{w}}^{-1}(\vec{\mu}_1 - \vec{\mu}_2)$$

$$w_0 = -\frac{\vec{w}^{\mathrm{T}}\vec{\mu}_1 + \vec{w}^{\mathrm{T}}\vec{\mu}_2}{2} \quad \left(\text{ou } w_0 = -\vec{w}^{\mathrm{T}}\left(\frac{N_1}{N_1+N_2}\vec{\mu}_1 + \frac{N_2}{N_1+N_2}\vec{\mu}_2\right)\right)$$

Algorithme 2-Classes, **généralisation**

POUR CHAQUE donnée test $\vec{x}$ FAIRE

$t = y_{\vec{w}}(\vec{x}) = \vec{w}^{\mathrm{T}}\vec{x} + w_0$

SI $t < 0$ ALORS

t=1

SINON

t=2

48

Discriminant linéaire de Fisher

- On peut voir l'analyse discriminante linéaire comme un cas particulier des **moindres carrés**
  - ➢ voir section 4.1.5
- Il est possible de généraliser au cas à **plus de 2 classes**
  - ➢ voir section 4.1.6

49

50

Régression
Modèles génératifs
Discriminant de Fisher

Émettent l'hypothèque que les données sont **gaussiennes**

Solution de type « ***closed form*** » (inversion de matrice)

Perceptron
Régression logistique

Émettent **aucune hypothèse** quant à la distribution des données

Solution obtenue grâce à une **descente de gradient**.

51

Perceptron
(section 4.1.7, Bishop)

52

Perceptron (2 classes)

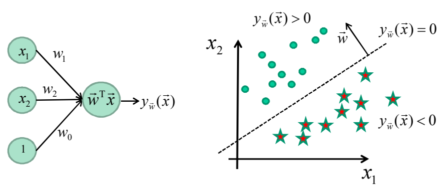
Contrairement aux approches précédentes, le perceptron **n'émet pas** l'hypothèse que les données sont **gaussiennes**

Le perceptron part de la definition brute de la classification binaire par **hyperplan**

$$
\begin{aligned}
y_{\vec{w}}(\vec{x}) &= sign\left(\vec{w}^T\vec{x}\right) \\
&= sign\left(\underbrace{w_0}_{\text{biais}} + \underbrace{w_1x_1 + w_2x_2 + \ldots + w_dx_d}_{\text{poids}}\right)
\end{aligned}
$$

53

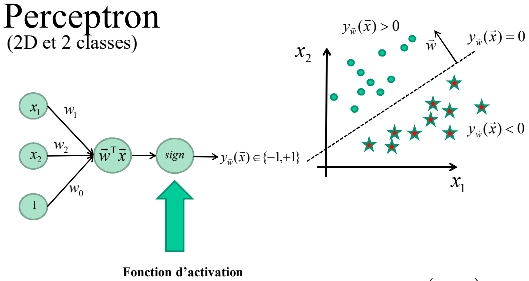
Perceptron
(2D et 2 classes)



$$
\begin{aligned}
y_{\vec{w}}(\vec{x}) &= w_0 + w_1x_1 + w_2x_1 \\
&= \vec{w}^{\mathrm{T}}\vec{x} \\
&= \vec{w}'^{\mathrm{T}}\vec{x}' \\
&\Rightarrow \vec{w}^{\mathrm{T}}\vec{x}
\end{aligned}
$$

54

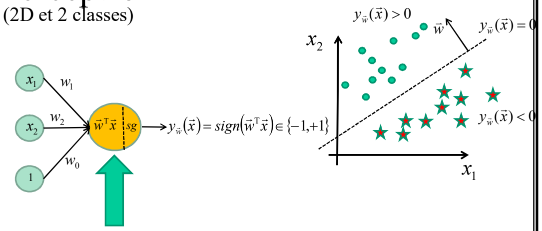
Perceptron
(2D et 2 classes)



$$y_{\vec{w}}(\vec{x}) = sign\left(\vec{w}^{\mathrm{T}}\vec{x}\right)$$

55

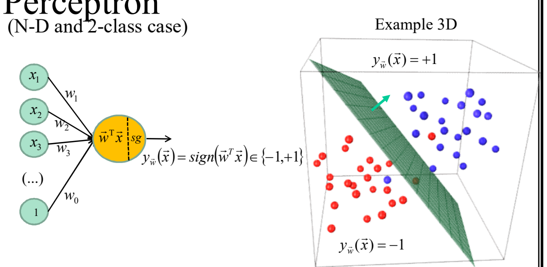
Perceptron
(2D et 2 classes)



Neurone
Produit scalaire + fonction d'activation

56

Perceptron
(N-D and 2-class case)



57

Nouvelle fonction de coût pour <u>**apprendre *W***</u>

<u>**Le but**</u>: avec des données d'entraînement $D = \{(\vec{x}_1, t_1), (\vec{x}_2, t_2), \ldots, (\vec{x}_N, t_N)\}$, estimer ***w*** afin que:

$$y_{\vec{w}}(\vec{x}_n) = t_n \qquad \forall n$$

En d'autres mots, minimiser **l'erreur d'entraînement**

$$E_D(\vec{w}) = \frac{1}{N}\sum_{n=1}^{N} l(y_{\vec{w}}(\vec{x}_n), t_n)$$

où *l(.,.)* est une **fonction de perte** (*loss function* en anglais).

Trouver la bonne fonction de perte et le bon algorithme **d'optimisation** est un sujet central en **apprentissage machine.**

58

Régression et classification

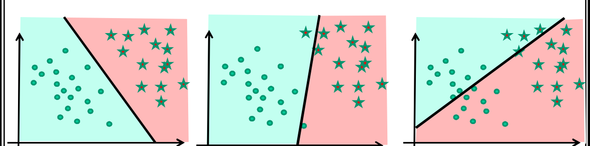
Vous vous souvenez de la **régression**?

Maximum de vraisemblance $$\vec{w} = \arg\min_{\vec{w}} \underbrace{\sum_{n=1}^{N} \frac{\left(t_n - y_{\vec{w}}(\vec{x}_n)\right)^2}{2}}_{E_D(\vec{w})}$$

Maximum *a posteriori* $$\vec{w} = \arg\min_{\vec{w}} \underbrace{\sum_{n=1}^{N} \left(t_n - y_{\vec{w}}(\vec{x}_n)\right)^2 + \lambda \vec{w}^{\mathrm{T}} \vec{w}}_{E_D(\vec{w})}$$

C'est un peu la même idée pour le Perceptron mais avec une <u>nouvelle fonction de coût</u>.

Fonction de perte

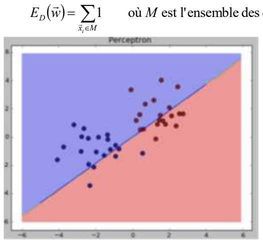


$E_D(y_{\vec{w}}(\vec{x})) \approx 0$ $E_D(y_{\vec{w}}(\vec{x})) > 0$ $E_D(y_{\vec{w}}(\vec{x})) >> 0$

Nouvelle fonction de coût pour **<u>apprendre *W*</u>**

Une function simple et indépendante de la distribution des données serait de compter 1 pour chaque donnée mal classée et 0 sinon

$$E_D(\vec{w}) = \sum_{\vec{x}_i \in M} 1 \qquad \text{où } M \text{ est l'ensemble des données mal classées}$$

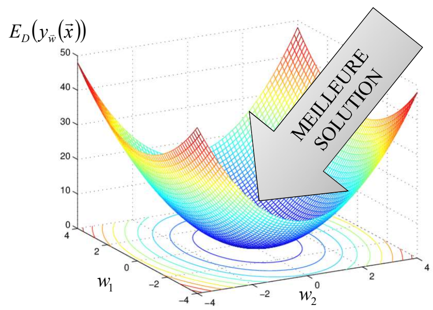


Exemple:

$$E_D(\vec{w}) = 15$$

Ainsi, la meilleure solution serait celle pour laquelle on **aurait aucune donnée mal classée**.

Malheureusement, cette function n'est **pas dérivable** partout et $\nabla_{\vec{w}} E_D(\vec{w}) = 0$ pour des **solutions non-optimales**



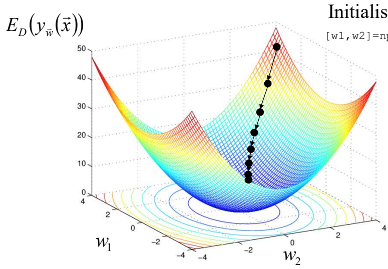
62

Perceptron

Question: comment trouver la meilleure solution? $\nabla_{\vec{w}} E_D(y_{\vec{w}}(\vec{x})) = 0$

Initialisation aléatoire

```
[w1,w2]=np.random.randn(2)
```



63

Gradient descent

Question: how to find the best solution? $\nabla E_D(y_{\vec{w}}(\vec{x})) = 0$

$$\vec{w}^{[k+1]} = \vec{w}^{[k]} - \eta \nabla E_D\left(y_{\vec{w}^{[k]}}(\vec{x})\right)$$

→ Gradient de la perte

→ Taux d'apprentissage *(Learning rate)*

65

Critère du perceptron (perte)

Observation

Une **donnée mal classée** survient lorsque

$$\vec{w}^{\mathrm{T}}\vec{x}_n > 0 \text{ et } t_n = -1$$

ou

$$\vec{w}^{\mathrm{T}}\vec{x}_n < 0 \text{ et } t_n = +1.$$

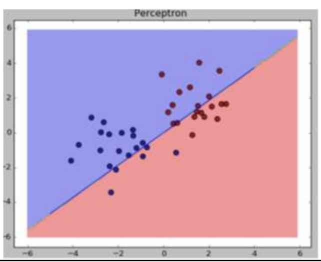
DONC $-\vec{w}^{\mathrm{T}}\vec{x}_n t_n$ est **TOUJOURS positif pour des données mal classés**

68

Critère du perceptron

Le **critère du perception** est une function qui pénalise les données mal classées

$$E_D(\vec{w}) = \sum_{\vec{x}_n \in M} -\vec{w}^{\mathrm{T}}\vec{x}_n t_n \qquad \text{où } M \text{ est l'ensemble des données mal classées}$$



$$E_D(\vec{w}) = 464.15$$

71

Perceptron

Question: comment trouver la meilleure solution $\vec{w}$ avec cette function de perte?

Réponse: une solution frequente est la **<u>descente de gradient</u>**.

$$\vec{w}^{[k+1]} = \vec{w}^{[k]} - \eta \nabla E_D\left(\vec{w}^{[k]}\right)$$

Gradient de la function de coût

Taux d'apprentissage (*learning rate*).

Descente de gradient de base

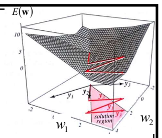
Initialiser $\vec{w}$
k=0
FAIRE k=k+1
$\vec{w} = \vec{w} - \eta \nabla E_D(\vec{w})$
JUSQU'À ce que toutes les données soient bien classées

72

Perceptron

Pour le critère du Perceptron

$$\nabla E_D(\vec{w}) = \sum_{\vec{x}_n \in M} -t_n \vec{x}_n$$



[Duda-Hart-Stork]

Batch optimization

Initialiser $\vec{w}$
k=0
DO k=k+1

$$\vec{w} = \vec{w} - \eta \left(\sum_{\vec{x}_n \in M} -t_n \vec{x}_n \right)$$

UNTIL toutes les données sont bien classées

NOTE importante sur le **taux d'apprentissage** η:

- **Trop faible** => convergence lente
- **Trop grand** => peut ne pas converger (et même diverger)
- Peut **décroître** à chaque itération (e.g. $\eta^{[k]} = cst / k$)

Perceptron

Une autre version de l'algorithme consiste à analyser **une donnée par itération**.

Descente de gradient stochastique

Initialiser $\vec{w}$
k=0
DO k=k+1
FOR n = 1 to N
IF $\vec{w}^{\mathrm{T}} \vec{x}_n t_n < 0$ THEN /* donnée mal classée */
$\vec{w} = \vec{w} + \eta t_n \vec{x}_n$
UNTIL toutes les données sont bien classées.

Critère du perceptron

Fonctions d'énergie similaires au critère du Perceptron dont le **gradient est le même**

$$E_D(\vec{w}) = \sum_{\vec{x}_n \in M} -\vec{w}^{\mathrm{T}} \vec{x}_n t_n \quad \text{où } M \text{ est l'ensemble des données mal classées}$$

$$E_D(\vec{w}) = \sum_{n=1}^{N} \max\left(0, -t_n \vec{w}^T \vec{x}_n\right)$$

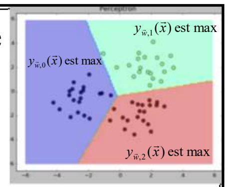
$$E_D(\vec{w}) = \sum_{n=1}^{N} \max\left(0, 1 - t_n \vec{w}^T \vec{x}_n\right)$$

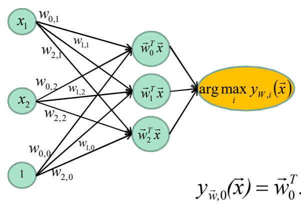
"Hinge Loss" or "SVM" Loss

Chapitre 6

Perceptron Multiclasse

(2D et 3 classes)



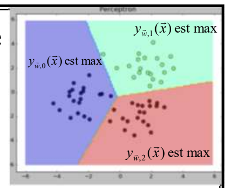


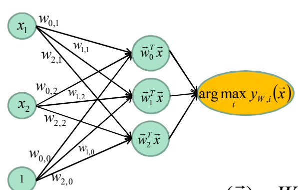
$$y_{\vec{w},0}(\vec{x}) = \vec{w}_0^T\vec{x} = w_{0,0} + w_{0,1}x_1 + w_{0,2}x_2$$
$$y_{\vec{w},1}(\vec{x}) = \vec{w}_1^T\vec{x} = w_{1,0} + w_{1,1}x_1 + w_{1,2}x_2$$
$$y_{\vec{w},2}(\vec{x}) = \vec{w}_2^T\vec{x} = w_{2,0} + w_{2,1}x_1 + w_{2,2}x_2$$

76

Perceptron Multiclasse

(2D et 3 classes)





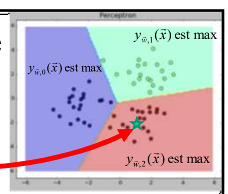
$$y_W(\vec{x}) = W^{\mathrm{T}}\vec{x}$$

$$y_W(\vec{x}) = \begin{bmatrix} w_{0,0} & w_{0,1} & w_{0,2} \\ w_{1,0} & w_{1,1} & w_{1,2} \\ w_{2,0} & w_{2,1} & w_{2,2} \end{bmatrix} \begin{bmatrix} 1 \\ x_1 \\ x_2 \end{bmatrix}$$

Perceptron Multiclasse

Exemple

(1.1, -2.0)

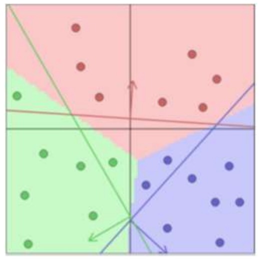


$$y_W(\vec{x}) = \begin{bmatrix} -2 & -3.6 & 0.5 \\ -4 & 2.4 & 4.1 \\ -6 & 4 & -4.9 \end{bmatrix} \begin{bmatrix} 1 \\ 1.1 \\ -2 \end{bmatrix} = \begin{array}{l} \begin{bmatrix} -6.9 \\ -9.6 \\ 8.2 \end{bmatrix} \end{array} \begin{array}{l} \text{Classe 0} \\ \text{Classe 1} \\ \text{Classe 2} \end{array}$$

78

$$y_W(\vec{x}) = \begin{bmatrix} w_{0,0} & w_{0,1} & w_{0,2} \\ w_{1,0} & w_{1,1} & w_{1,2} \\ w_{2,0} & w_{2,1} & w_{2,2} \end{bmatrix} \begin{bmatrix} 1 \\ x_1 \\ x_2 \end{bmatrix}$$

Tel qu'illustré ici, chaque **ligne de la matrice W** contient les paramètres (**normale + biais**) du **plan de séparation** linéaire de chaque classe.

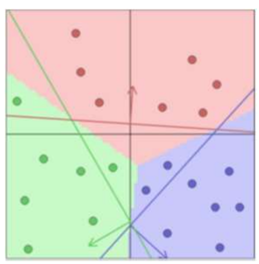


http://vision.stanford.edu/teaching/cs231n-demos/linear-classify/

79

$$y_W(\vec{x}) = \begin{bmatrix} w_{0,0} & w_{0,1} & w_{0,2} \\ w_{1,0} & w_{1,1} & w_{1,2} \\ w_{2,0} & w_{2,1} & w_{2,2} \end{bmatrix} \begin{bmatrix} 1 \\ x_1 \\ x_2 \end{bmatrix}$$

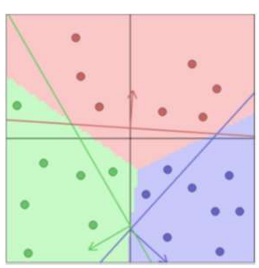
La **zone bleue** est la zone pour laquelle le score de la **classe 0** est le plus élevé.



80

$$y_W(\vec{x}) = \begin{bmatrix} w_{0,0} & w_{0,1} & w_{0,2} \\ w_{1,0} & w_{1,1} & w_{1,2} \\ w_{2,0} & w_{2,1} & w_{2,2} \end{bmatrix} \begin{bmatrix} 1 \\ x_1 \\ x_2 \end{bmatrix}$$

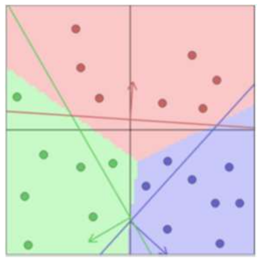
La **zone verte** est la zone pour laquelle le score de la **classe 1** est le plus élevé.



81

$$y_W(\vec{x}) = \begin{bmatrix} w_{0,0} & w_{0,1} & w_{0,2} \\ w_{1,0} & w_{1,1} & w_{1,2} \\ w_{2,0} & w_{2,1} & w_{2,2} \end{bmatrix} \begin{bmatrix} 1 \\ x_1 \\ x_2 \end{bmatrix}$$

La **zone rouge** est la zone pour laquelle le score de la **classe 2** est le plus élevé.



82

Perceptron Multiclasse

Fonction de coût

$$E_D(W) = \sum_{\vec{x}_n \in M} \left(\vec{w}_j^T \vec{x}_n - \vec{w}_{t_n}^T \vec{x}_n \right)$$

Somme sur l'ensemble des données mal classées

Score de la mauvaise classe

Score de la bonne classe

$$\nabla E_D(\mathrm{W}) = \sum_{\vec{x}_n \in M} \vec{x}_n$$

83

Perceptron Multiclasse

Descente de gradient stochastique

Initialiser **W**
k=0, i=0
DO k=k+1
  FOR n = 1 to N
    $j = \arg\max \mathrm{W}^T \vec{x}_n$
    IF $j \neq t_i$ THEN /* donnée mal classée*/
      $\vec{w}_j = \vec{w}_j - \eta \vec{x}_n$
      $\vec{w}_{t_n} = \vec{w}_{t_n} + \eta \vec{x}_n$
UNTIL toutes les données sont bien classées.

84

Perceptron Multiclasse

Exemple d'entraînement ($\eta=1$)

$$\vec{x}_n = (0.4, -1), t_n = 0$$

$$y_W(\vec{x}) = \begin{bmatrix} -2 & 3.6 & 0.5 \\ -4 & 2.4 & 4.1 \\ -6 & 4 & -4.9 \end{bmatrix} \begin{bmatrix} 1 \\ 0.4 \\ -1 \end{bmatrix} = \begin{array}{c} \begin{bmatrix} -1.6 \\ -7.1 \\ 0.5 \end{bmatrix} \end{array} \begin{array}{l} \text{Classe 0} \\ \text{Classe 1} \\ \text{Classe 2} \end{array}$$

FAUX!

85

Perceptron Multiclasse

Exemple d'entraînement ($\eta=1$)

$$\vec{x}_n = (0.4, -1.0), t_n = 0$$

$$\vec{w}_0 \leftarrow \vec{w}_0 + \vec{x}_n \qquad \begin{bmatrix} -2.0 \\ 3.6 \\ 0.5 \end{bmatrix} + \begin{bmatrix} 1 \\ 0.4 \\ -1 \end{bmatrix} = \begin{bmatrix} -1.0 \\ 4.0 \\ -0.5 \end{bmatrix}$$

$$\vec{w}_2 \leftarrow \vec{w}_2 - \vec{x}_n \qquad \begin{bmatrix} -6.0 \\ 4.0 \\ -4.9 \end{bmatrix} - \begin{bmatrix} 1 \\ 0.4 \\ -1 \end{bmatrix} = \begin{bmatrix} -7.0 \\ 3.6 \\ -3.9 \end{bmatrix}$$

86

En résumé

2 classes

$$E_D(\vec{w}) = \sum_{\vec{x}_n \in M} -t_n \vec{w}^T \vec{x}_n$$ où M est l'ensemble des données mal classées

$$E_D(\vec{w}) = \sum_{n=1}^{N} \max(0, -t_n \vec{w}^T \vec{x}_n)$$

$$E_D(\vec{w}) = \sum_{n=1}^{N} \max(0, 1 - t_n \vec{w}^T \vec{x}_n)$$ "Hinge Loss" or "SVM" Loss

K classes

$$E_D(W) = \sum_{\vec{x}_n \in M} \left(\vec{w}_j^T \vec{x}_n - \vec{w}_{t_n}^T \vec{x}_n\right)$$ où M est l'ensemble des données mal classées

$$E_D(W) = \sum_{n=1}^{N} \sum_j \max\left(0, \vec{w}_j^T \vec{x}_n - \vec{w}_{t_n}^T \vec{x}_n\right)$$

$$E_D(W) = \sum_{n=1}^{N} \sum_j \max\left(0, 1 + \vec{w}_j^T \vec{x}_n - \vec{w}_{t_n}^T \vec{x}_n\right)$$ "Hinge Loss" or "SVM" Loss

87

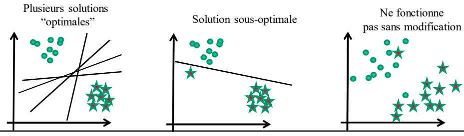
88

Perceptron

Advantages:
- Très simple
- Ne suppose pas que les données sont **gaussiennes**.
- Si les données sont linéairement séparables, le Perceptron est **<u>garanti</u>(!) de converger** en un nombre fini d'itérations (voir Duda-Hart-Stork pour la preuve)

Limitations:
- Gradient nul pour plusieurs solutions => plusieurs solutions "parfaites"
- Les données doivent être **linéairement séparables**

Plusieurs solutions "optimales" | Solution sous-optimale | Ne fonctionne pas sans modification



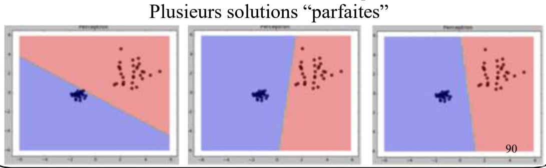
89

Perceptron

Advantages:
- Très simple
- Ne suppose pas que les données sont **gaussiennes**.
- Si les données sont linéairement séparables, le Perceptron est **<u>garanti</u> (!) de converger** en un nombre fini d'iterations (see Duda-Hart-Stork for proof)

Limitations:
- Gradient nul pour plusieurs solutions => plusieurs solutions "parfaites"
- Les données doivent être **linéairement séparables**

Plusieurs solutions "parfaites"



90

Comment améliorer le Perceptron?

Trois façons d'améliorer le Perceptron

1. Nouvelle **fonction d'activation** + nouvelle **fonction de coût**
   → **Régression logistique**
2. Utiliser des **fonctions de base**
   → **Méthodes à noyau (chap. 5-6)**
3. **Nouveau réseau**
   → **Réseaux de neurones multicouches (chap. 7)**

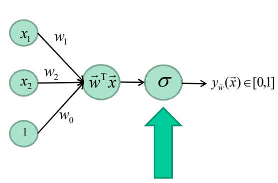
Régression logistique

(Sections 4.2.0, 4.3.2, 5.2.0 –Bishop)

Amélioration du Perceptron

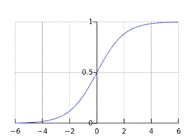
(2D, 2 classes)

Nouvelle fonction d'activation : **sigmoïde logistique**



Fonction d'activation sigmoïdale

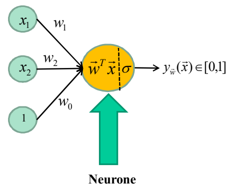
$$\sigma(v) = \frac{1}{1+e^{-v}}$$



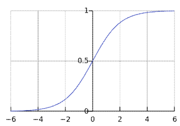
$$\boxed{y_{\vec{w}}(\vec{x}) = \sigma\left(\vec{w}^T\vec{x}\right) = \frac{1}{1+e^{-\vec{w}^T\vec{x}}}}$$

Amélioration du Perceptron
(2D, 2 classes)

Nouvelle fonction d'activation : **sigmoïde logistique**



$$\sigma(t) = \frac{1}{1+e^{-t}}$$

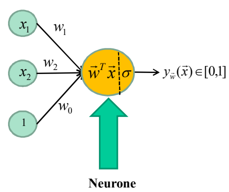


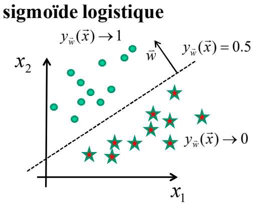
$$\boxed{y_{\vec{w}}(\vec{x}) = \sigma\left(\vec{w}^T\vec{x}\right) = \frac{1}{1+e^{-\vec{w}^T\vec{x}}}}$$

94

Amélioration du Perceptron
(2D, 2 classes)

Nouvelle fonction d'activation : **sigmoïde logistique**



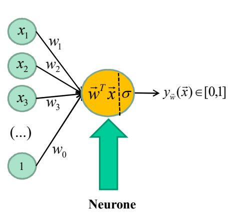


$$y_{\vec{w}}(\vec{x}) = \sigma\left(\vec{w}^T\vec{x}\right)$$

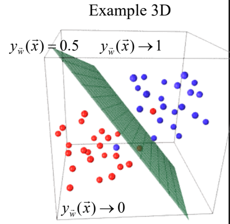
95

Amélioration du Perceptron
(N-D, 2 classes)

Nouvelle fonction d'activation : **sigmoïde logistique**



Example 3D

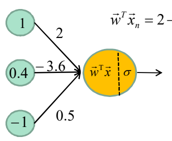


96

Amélioration du Perceptron
(N-D, 2 classes)

Exemple

$\vec{x}_n = (0.4, -1.0), \vec{w} = [2.0, -3.6, 0.5]$



$\vec{w}^T \vec{x}_n = 2 - 3.6 * 0.4 - 0.5 = -1.94$

$y_{\vec{w}}(\vec{x}) = \sigma(-1.94) = \frac{1}{1+e^{1.94}} = 0.125$

Puisque 0.125 est inférieur à 0.5, $\vec{x}_n$ est **<u>derrière</u>** le plan.

97

Amélioration du Perceptron
(N-D, 2 classes)

Avec une sigmoïde, on peut **simuler une probabilité conditionnelle** sur c_1 étant donné $\vec{x}$

$$y_{\vec{w}}(\vec{x}) = \sigma(\vec{w}^T \vec{x}) \Rightarrow P(c_1 \mid \vec{x})$$

Preuve:

$$P(c_1 \mid \vec{x}) = \frac{P(\vec{x} \mid c_1)P(c_1)}{P(\vec{x} \mid c_0)P(c_0) + P(\vec{x} \mid c_1)P(c_1)} \quad \text{(Bayes)}$$

$$= \frac{1}{1 + \frac{P(\vec{x} \mid c_0)P(c_0)}{P(\vec{x} \mid c_1)P(c_1)}}$$

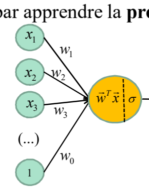
$$= \frac{1}{1+e^{-a}} \qquad \text{où } a = \ln\left[\frac{P(\vec{x} \mid c_0)P(c_0)}{P(\vec{x} \mid c_1)P(c_1)}\right]$$

$$= \sigma(a)$$

98

Amélioration du Perceptron
(N-D, 2 classes)

En d'autres mots, si on entraîne correctement un réseau logistique, on fini par apprendre la **probabilité conditionnelle de la classe c_1.**



$$y_{\vec{w}}(\vec{x}) = \sigma(\vec{w}^T \vec{x}) = P(c_1 \mid \vec{w}, \vec{x})$$

$$\Rightarrow P(c_0 \mid \vec{w}, \vec{x}) = 1 - P(c_1 \mid \vec{w}, \vec{x}) = 1 - y_{\vec{w}}(\vec{x})$$

Quelle est la function de coût d'un réseau logistique?

99

Fonction de coût d'un réseau logistique?
(2 classes)

Dans le cas d'un réseau logistique nous avons

Ensemble d'entraînement : $D=\{(\vec{x}_1,t_1),(\vec{x}_2,t_2),\ldots,(\vec{x}_n,t_n)\}$

Sortie du réseau: $y_{\vec{w}}(\vec{x})=\sigma(\vec{w}^T\vec{x})=P(c_1\mid\vec{w},\vec{x})$

$$P(D\mid\vec{w})=\prod_{n=1}^{N}P(c_1\mid\vec{w},\vec{x}_n)^{t_n}\left(1-P(c_1\mid\vec{w},\vec{x}_n)\right)^{1-t_n}$$
$$=\prod_{n=1}^{N}y_{\vec{w}}(\vec{x}_n)^{t_n}\left(1-y_{\vec{w}}(\vec{x}_n)\right)^{1-t_n}$$

105

____

Fonction de coût d'un réseau logistique?
(2 classes)

$$P(D\mid\vec{w})=\prod_{n=1}^{N}y_{\vec{w}}(\vec{x}_n)^{t_n}\left(1-y_{\vec{w}}(\vec{x}_n)\right)^{1-t_n}$$

Solution : Maximum de vraisemblance

$$W=\arg\max_W P(D\mid W)$$
$$=\arg\max_W \prod_{n=1}^{N}y_W(\vec{x}_n)^{t_n}\left(1-y_W(\vec{x}_n)\right)^{1-t_n}$$
$$=\arg\min_W \sum_{n=1}^{N}-\ln\left[y_W(\vec{x}_n)^{t_n}\left(1-y_W(\vec{x}_n)\right)^{1-t_n}\right]$$
$$=\arg\min_W -\underbrace{\sum_{n=1}^{N}t_n\ln\left(y_W(\vec{x}_n)\right)+(1-t_n)\ln\left(1-y_W(\vec{x}_n)\right)}_{E_D(\vec{w})}$$

106

____

Fonction de coût d'un réseau logistique?
(2 classes)

$$P(D\mid\vec{w})=-\prod_{n=1}^{N}y_{\vec{w}}(\vec{x}_n)^{t_n}\left(1-y_{\vec{w}}(\vec{x}_n)\right)^{1-t_n}$$

La fonction de coût est **–ln de la vraisemblance**

$$E_D(\vec{w})=-\sum_{n=1}^{N}t_n\ln(y_{\vec{w}}(\vec{x}_n))+(1-t_n)\ln(1-y_{\vec{w}}(\vec{x}_n))$$

Entropie croisée (*Cross entropy*)

On peut également démontrer que

$$\nabla_{\vec{w}}E_D(\vec{w})=\sum_{n=1}^{N}(y_{\vec{w}}(\vec{x}_n)-t_n)\vec{x}_n$$

Contrairement au Perceptron le gradient ne depend pas seulement des données mal classées

Preuve en classe ou en devoir

107

____

Optimisation d’un réseau logistique

Optimisation par batch

Initialiser $\vec{w}$
k=0, i=0
DO k=k+1

$$\frac{dE_D(\vec{w})}{d\vec{w}} = \sum_{n=1}^{N} \left(y_{\vec{w}}(\vec{x}_n) - t_n\right)\vec{x}_n$$

$$\vec{w} = \vec{w} - \eta \frac{dE_D(\vec{w})}{d\vec{w}}$$

UNTIL K==K_MAX.

Descente de gradient stochastique

Initialiser $\vec{w}$
k=0, i=0
DO k=k+1
FOR n = 1 to N

$$\vec{w} = \vec{w} - \eta\left(y_{\vec{w}}(\vec{x}_n) - t_n\right)\vec{x}_n$$

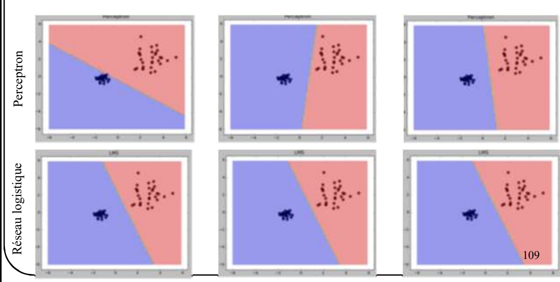
UNTIL K==K_MAX.

108

Réseau logistique

Advantages:

- **Plus stable que le Perceptron**
- Fonctionne mieux avec des **données non séparables**

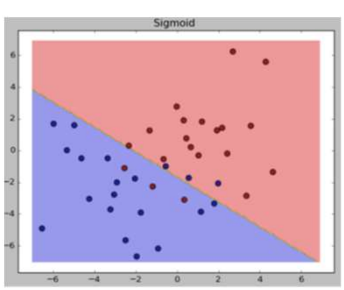


109

Réseau logistique

Advantages:

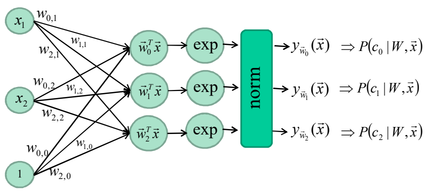
- **Plus stable que le Perceptron**
- Fonctionne mieux avec des **données non séparables**



110

Et pour K>2 classes?

Nouvelle fonction d'activation : **Softmax**

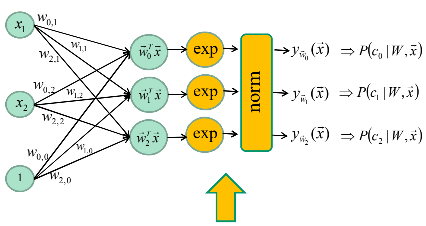


$$y_{\vec{w}_i}(\vec{x}) = \frac{e^{\vec{w}_i^T\vec{x}}}{\sum_c e^{\vec{w}_c^T\vec{x}}}$$

111

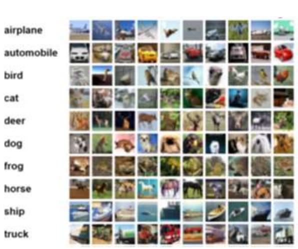
Et pour K>2 classes?

Nouvelle fonction d'activation : **Softmax**



112

Et pour K>2 classes?



'airplane' $\Rightarrow$ t = [1000000000]
'automobile' $\Rightarrow$ t = [0100000000]
'bird' $\Rightarrow$ t = [0010000000]
'cat' $\Rightarrow$ t = [0001000000]
'deer' $\Rightarrow$ t = [0000100000]
'dog' $\Rightarrow$ t = [0000010000]
'frog' $\Rightarrow$ t = [0000001000]
'horse' $\Rightarrow$ t = [0000000100]
'ship' $\Rightarrow$ t = [0000000010]
'truck' $\Rightarrow$ t = [0000000001]

Étiquettes de classe : **one-hot vector** 113

Et pour K>2 classes?

$$P(D|W)=\prod_{n=1}^{N}\prod_{k=1}^{K}\left(P(t_n|W,\vec{x}_n)\right)^{t_{nk}}$$

Entropie croisée (*cross entropy*)

$$E_D(\mathrm{W})=-\ln(P(D|W))=-\sum_{n=1}^{N}\sum_{k=1}^{K}t_{kn}\ln P(t_n|W,\vec{x}_n)$$

Puisqu'on veut que la sortie du réseau $y_W(\vec{x}_n)$ soit égale à $P(t_n|W,\vec{x}_n)$

$$\boxed{E_D(\mathrm{W})=-\sum_{n=1}^{N}\sum_{k=1}^{K}t_{kn}\ln y_{W_k}(\vec{x}_n)}$$

114

Et pour K>2 classes?

En general, on ajoute 1/N pour normaliser le calcul de la loss

$$E_D(\mathrm{W})=-\frac{1}{N}\sum_{n=1}^{N}\sum_{k=1}^{K}t_{kn}\ln y_W(\vec{x}_n)$$

On peut montrer que

$$\boxed{\nabla_W E_D(\mathrm{W})=\frac{1}{N}\sum_{n=1}^{N}\vec{x}_n\left(y_W(\vec{x}_n)-t_{kn}\right)}$$

115

Optimisation d'un réseau logistique multiclasse

Optimisation par batch

Initialiser **W**
k=0, i=0
DO k=k+1

$$\nabla_W E(\mathrm{W})=\frac{1}{N}\sum_{n=1}^{N}(y_W(\vec{x}_n)-t_n)\vec{x}_n$$

$$\mathrm{W}=\mathrm{W}-\eta\nabla_W E(\mathrm{W})$$

UNTIL K==K_MAX.

Descente de gradient stochastique

Initialiser **W**
k=0, i=0
DO k=k+1
FOR n = 1 to N

$$\mathrm{W}=\mathrm{W}-\eta\left(y_W(\vec{x}_n)-t_n\right)\vec{x}_n$$

UNTIL K==K_MAX.

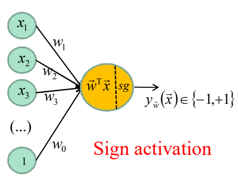
116

Wow! Beaucoup d'information…

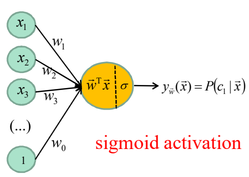
Résumons…

Réseaux de neurones

2 classes



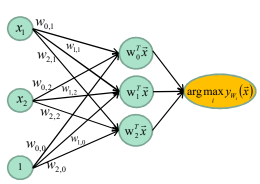
Sign activation

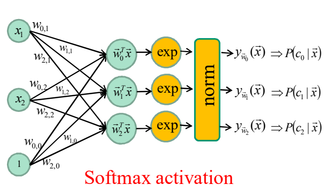


sigmoid activation

Réseaux de neurones

K classes





Softmax activation

Fonctions de coûts

2 classes

$$E_D(\vec{w}) = \sum_{\vec{x}_n \in M} -t_n \vec{w}^{\mathrm{T}} \vec{x}_n \qquad \text{où } M \text{ est l'ensemble des données mal classées}$$

$$E_D(\vec{w}) = \sum_{n=1}^{N} \max\left(0, -t_n \vec{w}^{\mathrm{T}} \vec{x}_n\right)$$

$$E_D(\vec{w}) = \sum_{n=1}^{N} \max\left(0, 1 - t_n \vec{w}^{\mathrm{T}} \vec{x}_n\right)$$

"Hinge Loss" ou "SVM" Loss

$$E_D(\vec{w}) = -\sum_{n=1}^{N} t_n \ln\left(y_{\vec{w}}(\vec{x}_n)\right) + (1-t_n)\ln\left(1 - y_{\vec{w}}(\vec{x}_n)\right)$$

Entropie croisée
(ou *cross entropy*)

120

Fonctions de coûts

K classes

$$E_D(\mathrm{W}) = \sum_{\vec{x}_n \in M} \left(\mathrm{W}_j^{\mathrm{T}} \vec{x}_n - \mathrm{W}_{t_n}^{\mathrm{T}} \vec{x}_n\right) \qquad \text{où } M \text{ est l'ensemble des données mal classées}$$

$$E_D(\mathrm{W}) = \sum_{n=1}^{M} \sum_{j} \max\left(0, \mathrm{W}_j^{\mathrm{T}} \vec{x}_n - \mathrm{W}_{t_n}^{\mathrm{T}} \vec{x}_n\right)$$

$$E_D(\mathrm{W}) = \sum_{n=1}^{N} \sum_{j} \max\left(0, 1 + \mathrm{W}_j^{\mathrm{T}} \vec{x}_n - \mathrm{W}_{t_n}^{\mathrm{T}} \vec{x}_n\right)$$

"Hinge Loss" ou "SVM" Loss

$$E_D(\mathrm{W}) = -\sum_{n=1}^{N} \sum_{k=1}^{K} t_{kn} \ln y_{W,k}(\vec{x}_n)$$

Entropie croisée avec « *one hot vector* »
(ou *cross entropy*)

121

Optimisation

Descente de gradient

$$\vec{\mathrm{w}}^{[k+1]} = \vec{\mathrm{w}}^{[k]} - \eta^{[k]} \nabla E$$

→ Gradient de la function de coût

→ Taux d'apprentissage ou "learning rate".

Optimisation par Batch

Initialiser $\vec{\mathrm{w}}$
k=0
FAIRE k=k+1

$$\vec{w} = \vec{w} - \eta^{[k]} \sum_i \nabla E(\vec{x}_i)$$

JUSQU'À ce que toutes les données sont bien classées ou k==MAX_ITER

Descente de gradient stochastique

Initialiser $\vec{\mathrm{w}}$
k=0
FAIRE k=k+1
FOR n = 1 to N

$$\vec{w} = \vec{w} - \eta^{[k]} \nabla E(\vec{x}_n)$$

JUSQU'À ce que toutes les données sont bien classées ou k== MAX_ITER

Parfois $\eta^{[k]} = cst / k$

122

Régularisation

Régularisation

Différents poids peuvent donner le même score

$$\vec{x} = (1.0, 1.0, 1.0)$$
$$\vec{w}_1 = [1, 0, 0]$$
$$\vec{w}_2 = [1/3, 1/3, 1/3]$$

Quels poids sont les meilleurs?

Solution: Maximum a posteriori

$$\vec{w}_1 \vec{x} = \vec{w}_2 \vec{x} = 1$$

Maximum *a posteriori*

Régularisation

Maximum de vraisemblance

$$W = \underset{W}{\operatorname{argmax}} P(D \mid W)$$
$$\ldots$$
$$= \underset{W}{\operatorname{argmin}} \underbrace{\sum_{n=1}^{N} -\ln P(t_n \mid \vec{x}_n, W)}_{E_D(W)}$$

Maximum *a posteriori*

$$W = \underset{W}{\operatorname{argmax}} P(W \mid D)$$
$$\ldots$$
$$= \underset{W}{\operatorname{argmin}} \underbrace{\sum_{n=1}^{N} -\ln P(t_n \mid \vec{x}_n, W)}_{E_D(W)} + \lambda R(W)$$

Note :
il est fréquent de combiner différentes **fonctions de coût** avec différentes **fonctions de régularisation**

127

Maximum *a posteriori*

Constante

$$E(\mathrm{W}) = \sum_{n=1}^{N} l\left(y_W(\vec{x}_n), t_n\right) + \lambda R(\mathrm{W})$$

Fonction de perte

Regularisation

$$R(\theta) = \|W\|_1 \ ou \ \|W\|_2$$

128

Maximum *a posteriori*

Exemple : ***Hinge loss*** + **régularisation L2**

$$E(\mathrm{w}) = \sum_{n=1}^{N} \max\left(0, 1 - t_n \mathrm{w}^T \vec{x}_n\right) + \lambda \|\mathrm{w}\|^2$$

$$\nabla_{\mathrm{w}} E(\mathrm{w}) = \sum_{\vec{x}_n \in M} -t_n \vec{x}_n + 2\lambda \sum_{d=0}^{D} w_d$$

129

Maximum *a posteriori*

Exemple : **entropie croisée** + **régularisation L2**

$$\arg\min_W - \ln(P(D|W)) + \lambda \|W\|^2$$

$$\arg\min_W - \sum_{n=0}^{N} t_n \ln(y_W(\vec{x}_n)) + (1-t_n)\ln(1-y_W(\vec{x}_n)) + \lambda \sum_{i=1}^{d} (w_i)^2$$

$$\nabla_W E(\mathrm{w}) = \sum_{n=1}^{N} (y_W(\vec{x}_n) - t_n)\vec{x}_n + 2\lambda \sum_{d=0}^{D} w_d$$

130

Exemples

```
from sklearn.linear_model import SGDClassifier

C = SGDClassifier(loss='perceptron',learning_rate='constant',
                  eta0=1, n_iter=1000)


C = SGDClassifier(loss='log', learning_rate ='constant',
                  eta0=1, n_iter=1000)


C = SGDClassifier(loss='hinge',penalty='l2',alpha=0.01,
                  learning_rate ='invscaling', eta0=1, n_iter=1000)
```

131

132

Mieux comprendre

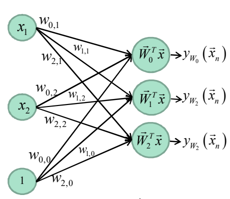
Entropie croisée vs *Hinge loss*

133

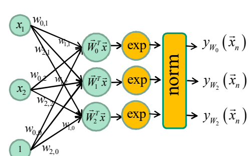
Entropie croisée vs *Hinge Loss*

Dépendamment de la *loss* utilisée, la **sortie du réseau** sera différente

- *Hinge loss* : sortie multiplication matrice-vecteur
- Entropie croisée : sortie softmax



$$y_{W_i}(\vec{x}_n) = \vec{W}_i^{\mathrm{T}}\vec{x}$$



$$y_{W_i}(\vec{x}_n) = \frac{e^{\vec{W}_i^{\mathrm{T}}\vec{x}}}{\sum_j e^{\vec{W}_j^{\mathrm{T}}\vec{x}}}$$

134

$$\vec{x} = \begin{bmatrix} -15 \\ 22 \\ -44 \\ 56 \end{bmatrix}, t = 2$$

$$\underbrace{\begin{bmatrix} 0.0 & 0.01 & -0.05 & 0.1 & 0.05 \\ 0.2 & 0.7 & 0.2 & 0.05 & 0.16 \\ -0.3 & 0.0 & -0.45 & -0.2 & 0.03 \end{bmatrix}}_{W} \underbrace{\begin{bmatrix} 1 \\ -15 \\ 22 \\ -44 \\ 56 \end{bmatrix}}_{\vec{x}}$$

Hinge loss

Score

$$\begin{bmatrix} -2.85 \\ 0.86 \\ 0.28 \end{bmatrix} \qquad \begin{gathered} \max(0, -2.85 - 0.28 + 1) + \\ \max(0, 0.86 - 0.28 + 1) \\ = \\ \max(0, -2.13) + \max(0, 1.58) \\ = \\ \boldsymbol{1.58} \end{gathered}$$

Entropie croisée

Score

$$\begin{bmatrix} -2.85 \\ 0.86 \\ 0.28 \end{bmatrix} \xrightarrow{\text{exp}} \begin{bmatrix} 0.06 \\ 2.36 \\ 1.32 \end{bmatrix} \xrightarrow{\text{norm}} \begin{bmatrix} 0.02 \\ 0.63 \\ 0.35 \end{bmatrix} \qquad -\ln(0.35) = \boldsymbol{0.452}$$

(Softmax)

135

$$\vec{x} = \begin{bmatrix}-15\\22\\-44\\56\end{bmatrix}, t = 2$$

$$\begin{bmatrix}0.0 & 0.01 & -0.05 & 0.1 & 0.05\\0.2 & 0.7 & 0.2 & 0.05 & 0.16\\-0.3 & 0.0 & -0.45 & -0.2 & 0.03\end{bmatrix}\begin{bmatrix}1\\-15\\22\\-44\\56\end{bmatrix}$$

Hinge loss

Score

$$\begin{bmatrix}-2.85\\0.86\\0.28\end{bmatrix}$$

$$\max(0, -2.85 - 0.28 + 1) + \max(0, 0.86 - 0.28 + 1) = \max(0, -2.13) + \max(0, 1.58) = \mathbf{1.58}$$

Entropie croisée

Score

$$\begin{bmatrix}-2.85\\0.86\\0.28\end{bmatrix} \xrightarrow{\exp} \begin{bmatrix}0.06\\2.36\\1.32\end{bmatrix} \xrightarrow{\text{norm}} \begin{bmatrix}0.02\\0.63\\0.35\end{bmatrix} \quad -\ln(0.35) = \mathbf{0.452}$$

(Softmax)

Q1: Qu'arrive-t-il si le score de la classe 0 (donc -2.85) change un peu?

136

$$\vec{x} = \begin{bmatrix}-15\\22\\-44\\56\end{bmatrix}, t = 2$$

$$\begin{bmatrix}0.0 & 0.01 & -0.05 & 0.1 & 0.05\\0.2 & 0.7 & 0.2 & 0.05 & 0.16\\-0.3 & 0.0 & -0.45 & -0.2 & 0.03\end{bmatrix}\begin{bmatrix}1\\-15\\22\\-44\\56\end{bmatrix}$$

Hinge loss

Score

$$\begin{bmatrix}-2.85\\0.86\\0.28\end{bmatrix}$$

$$\max(0, -2.85 - 0.28 + 1) + \max(0, 0.86 - 0.28 + 1) = \max(0, -2.13) + \max(0, 1.58) = \mathbf{1.58}$$

Entropie croisée

Score

$$\begin{bmatrix}-2.85\\0.86\\0.28\end{bmatrix} \xrightarrow{\exp} \begin{bmatrix}0.06\\2.36\\1.32\end{bmatrix} \xrightarrow{\text{norm}} \begin{bmatrix}0.02\\0.63\\0.35\end{bmatrix} \quad -\ln(0.35) = \mathbf{0.452}$$

(Softmax)

Q2: Qu'arrive-t-il si le score de la classe 1 (donc 0.86) augmente?

137

$$\vec{x} = \begin{bmatrix}-15\\22\\-44\\56\end{bmatrix}, t = 2$$

$$\begin{bmatrix}0.0 & 0.01 & -0.05 & 0.1 & 0.05\\0.2 & 0.7 & 0.2 & 0.05 & 0.16\\-0.3 & 0.0 & -0.45 & -0.2 & 0.03\end{bmatrix}\begin{bmatrix}1\\-15\\22\\-44\\56\end{bmatrix}$$

Hinge loss

Score

$$\begin{bmatrix}-2.85\\0.86\\0.28\end{bmatrix}$$

$$\max(0, -2.85 - 0.28 + 1) + \max(0, 0.86 - 0.28 + 1) = \max(0, -2.13) + \max(0, 1.58) = \mathbf{1.58}$$

Entropie croisée

Score

$$\begin{bmatrix}-2.85\\0.86\\0.28\end{bmatrix} \xrightarrow{\exp} \begin{bmatrix}0.06\\2.36\\1.32\end{bmatrix} \xrightarrow{\text{norm}} \begin{bmatrix}0.02\\0.63\\0.35\end{bmatrix} \quad -\ln(0.35) = \mathbf{0.452}$$

(Softmax)

Q3: quelles sont les valeurs MIN/MAX de ces deux loss?

138

$$\vec{x} = \begin{bmatrix}-15\\22\\-44\\56\end{bmatrix}, t = 2$$

$$\begin{bmatrix} 0.0 & 0.01 & -0.05 & 0.1 & 0.05 \\ 0.2 & 0.7 & 0.2 & 0.05 & 0.16 \\ -0.3 & 0.0 & -0.45 & -0.2 & 0.03 \end{bmatrix} \begin{bmatrix} 1 \\ -15 \\ 22 \\ -44 \\ 56 \end{bmatrix}$$

Hinge loss

Score

$$\begin{bmatrix}-2.85\\0.86\\0.28\end{bmatrix}$$

$$\max(0, -2.85 - 0.28 + 1) + \max(0, 0.86 - 0.28 + 1) = \max(0, -2.13) + \max(0, 1.58) = \mathbf{1.58}$$

Entropie croisée

Score

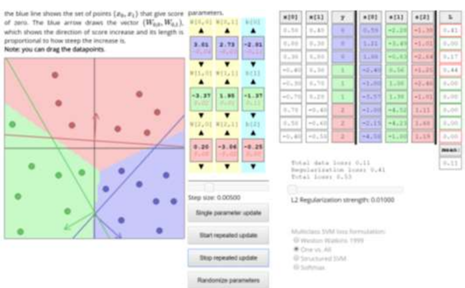
$$\begin{bmatrix}-2.85\\0.86\\0.28\end{bmatrix} \xrightarrow{\text{exp}} \begin{bmatrix}0.06\\2.36\\1.32\end{bmatrix} \xrightarrow{\text{norm}} \begin{bmatrix}0.02\\0.63\\0.35\end{bmatrix} \quad -\ln(0.35) = \mathbf{0.452}$$

(Softmax)

Q4: quelle serait la loss totale si on devait ajouter un terme de régularisation L2?

139

Superbe système en ligne de Stanford pour illustrer la classification multiclasses



http://vision.stanford.edu/teaching/cs231n-demos/linear-classify/