Apprentissage : Classification et Regression

A. Carlier, S. Mouysset

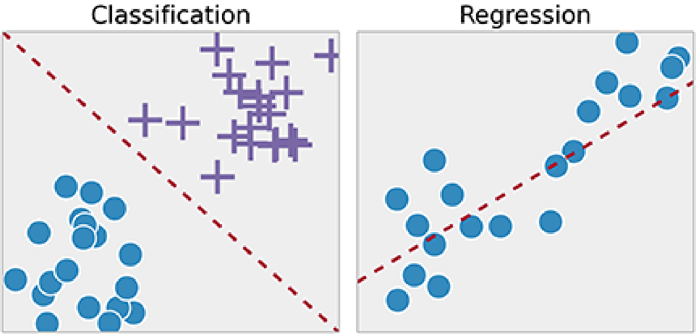
22/02/2019

Apprentissage supervisé

Il existe deux principaux types d'apprentissage supervisé :

- **Classification :** Assigner une catégorie à chaque observation :
  - Les catégories sont discrètes
  - La cible est un indice de classe : $y \in \{0, .., K-1\}$
  - Exemple : reconnaissance de chiffres manuscrits :
    - $\mathtt{x}$: vecteur ou matrice des intensités des pixels de l'image
    - y : identité du chiffre
- **Régression :** Prédire une valeur réelle à chaque observation :
  - les catégories sont continues
  - la cible est un nombre réel $y \in \mathbb{R}$
  - Exemple : prédire le cours d'une action
    - $\mathtt{x}$: vecteur contenant l'information sur l'activité économique
    - y : valeur de l'action le lendemain

Classification et Régression



$\Rightarrow$ Régression

Régression linéaire

Soit l'ensemble $\mathcal{D}$ contenant m exemples d'apprentissage.

Pour l'exemple, $\mathtt{x}^{(i)}$, le modèle linéaire s'écrit :

$$\hat{y}^{(i)} = \theta^T \mathtt{x}^{(i)}$$

et la fonction de coût quadratique s'écrit :

$$(\hat{y}^{(i)} - y^{(i)})^2$$

→ Formulation aux **Moindres Carrés** ! (cf. 1SN)

Trouver θ qui minimise la perte sur tous les exemples d'apprentissage :

$$\theta^* = \arg\min_\theta \sum_{i=1}^{m} (\theta^T x^{(i)} - y^{(i)})^2 \tag{1}$$

Régression linéaire

Soit l'ensemble $\mathcal{D}$ contenant m exemples d'apprentissage en dimension d (d variables), on définit :

- $\mathtt{X} \in \mathbb{R}^{m \times d}$ matrice des données
- $\mathrm{y} \in \mathbb{R}^{m}$: vecteur de cibles
- $\hat{\mathrm{y}} \in \mathbb{R}^{m}$: vecteur de prédictions avec $\hat{y} = \mathtt{X}\theta$
- $\theta \in \mathbb{R}^{d}$ vecteur des paramètres du modèle à estimer

Régression aux moindres carrés

Estimer le modèle linéaire θ entre les données et les cibles en minimisant la somme des résidus quadratiques :

$$\min_{\theta} \|\mathtt{X}\theta - \mathrm{y}\|^2 = J(\theta)$$

Régression linéaire

Résolution des problèmes aux moindres carrés :

Condition Nécessaire du premier ordre :

$$\frac{dJ(\theta)}{d\theta} = 0 = 2(\mathtt{X}|\mathtt{X}\theta - \mathtt{y}).$$

Si $det(\mathtt{X}^T\mathtt{X}) \neq 0$, la solution analytique est :

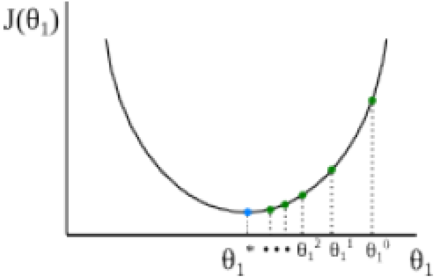
$$\hat{\theta} = (\mathtt{X}^T\mathtt{X})^{-1}\mathtt{X}^T\mathtt{y}$$

En pratique, calculs coûteux donc solution itérative : **descente de gradient** !

Remarque : les problèmes aux moindres carrés sont **convexes**
$\rightarrow$ minimum local est global !

Régression linéaire

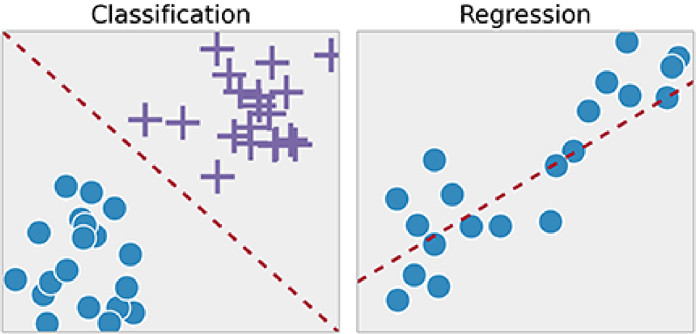
Commencer à un point aléatoire θ_1^0
Répéter :
Déterminer une direction
Choisir un pas d'apprentissage
Actualiser
jusqu'à convergence



Algorithme Descente de gradient $(\mathcal{D}, \alpha)$

1: Initialiser $\overrightarrow{\theta} \leftarrow \overrightarrow{0}$
2: TANT QUE pas convergence FAIRE
3: POUR k de 1 à d FAIRE
4: $\theta_k \leftarrow \theta_k - \alpha \dfrac{\partial J(\theta)}{\partial \theta_k}$

Classification et Régression



$\Rightarrow$ Classification

Régression logistique

Il s'agit d'un **algorithme de classification** malgré son nom.
Il est très populaire et très utilisé car il est simple et efficace en général.

Exemples d'utilisation :

- Emails : spam/non spam
- Achats sur Internet : frauduleux/non-frauduleux
- tumeur cancéreuse : maligne/non maligne
- reconnaissance de chiffres manuscrits

Régression logistique

En entrée, les données peuvent être :

- continues : âge, poids, intensités de pixels
- catégorielles : groupe sanguin

En sortie, la sortie du modèle peut être :

- **binaire** : échec/succès, 0/1, -/+
  $\Rightarrow$ fonction **sigmoïde ou logistique**
- **multinomiale** (multi-classes) : chiffres
  $\Rightarrow$ fonction **softmax**

Régression logistique : cas binaire

Comme avec la régression linéaire, on prend un modèle linéaire type :

$$z = \theta_0 + \theta_1 x_1 + ... + \theta_n x_n = \theta^T \mathtt{x}$$

Ce modèle linéaire agit comme **séparateur** des 2 classes.
Puis on veut une probabilité : $0 \leq h_{\theta(x)} = g(z) \leq 1$ telle que :

- si $h_\theta(x) \leq 0.5$, alors la classe est 0 ($y = 0$)
- si $h_\theta(x) > 0.5$, alors la classe est 1 ($y = 1$)

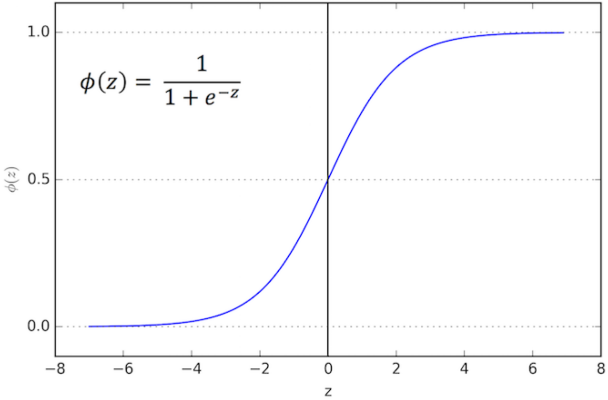
Quelle est cette fonction g telle que :

$$h_\theta(x) = g(z) = g(\theta^T \mathtt{x}) = P(y = 1|x; \theta) \text{ ?}$$

Régression logistique : cas binaire

La **sigmoïde** ou **fonction logistique** définie par :

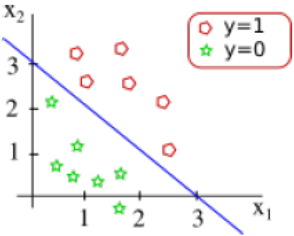
$$g(z) = \frac{1}{1 + exp(-z)}$$



Fonction sigmoïde ou Logistique $(z = \theta^T \mathrm{x})$

Régression logistique : cas binaire

Cas 2D où les données sont linéairement séparables

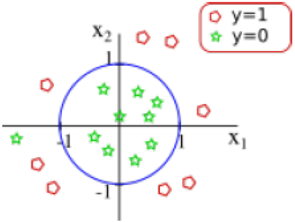


$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$$

- on prédit $y = 1$ si $-3 + x_1 + x_2 > 0$
- on prédit $y = 0$ si $x_1 + x_2 \leq 3$

Régression logistique : cas binaire

Cas 2D où les données sont linéairement séparables



$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2)$$

- on prédit $y = 1$ si $x_1^2 + x_2^2 > 1$
- on prédit $y = 0$ si $x_1^2 + x_2^2 \leq 1$

Régression logistique : cas binaire

Comment estimer automatiquement θ ?

On a un **corpus d'apprentissage** $\mathcal{D}$, contenant m exemples avec :

$$x^{(j)} = \begin{bmatrix} x_0^{(j)} \\ x_1^{(j)} \\ \vdots \\ x_d^{(j)} \end{bmatrix} \in \mathbb{R}^{d+1} \text{ et } y^{(j)} \in \{0, 1\}, \forall j \in \{1, .., m\}$$

et le **modèle** est défini par la fonction sigmoïde :

$$P(y = 1|x; \theta) = \frac{1}{1 + e^{-\theta^T x}}$$

Régression logistique : cas binaire

Il faut minimiser une **fonction de perte** à l'aide d'une technique d'optimisation (descente de gradient).

Peut on utiliser la même fonction de perte que pour la régression linéaire ?

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} perte\left(h_\theta(x^{(i)}), y^{(i)}\right)$$

avec

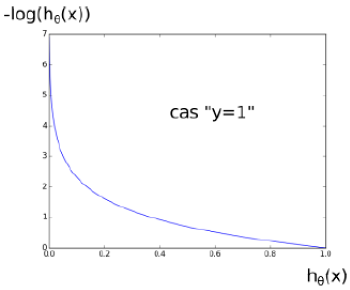
$$perte(h_\theta(x^{(i)}), y^{(i)}) = \frac{1}{2}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2 = \frac{1}{2}\left(\frac{1}{1+e^{-\theta^T x^{(i)}}} - y^{(i)}\right)^2$$

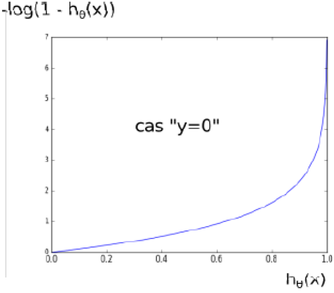
Or cette fonction n'est **pas convexe** donc pas utilisable avec la descente de gradient !

Régression logistique : cas binaire

On introduit donc la **fonction de perte logistique ou entropie croisée (cross-entropy)** définie par :

$$
\begin{aligned}
perte(h_{\theta(x^{(i)})}, y^{(i)})) &= \begin{cases} -log(h_\theta(x)) \text{ si } y = 1 \\ -log(1 - h_\theta(x)) \text{ si } y = 0 \end{cases} \\
&= -ylog(h_\theta(x)) - (1-y)log(1-h_\theta(x))
\end{aligned}
$$





Régression logistique : cas binaire

Sur les m exemples, la fonction de perte devient :

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} -y^{(i)} log(h_\theta(x^{(i)})) - (1 - y^{(i)}) log(1 - h_\theta(x^{(i)}))$$

Pour minimiser cette fonction, on applique la descente de gradient.

$$\begin{aligned} \frac{\partial J(\theta_j)}{\partial \theta_j} &= \frac{\partial}{\partial \theta_j} [-y log(h_\theta(x)) - (1 - y) log(1 - h_\theta(x))] \\ &= -y \frac{\partial}{\partial \theta_j} (log(h_\theta(x))) - (1 - y) \frac{\partial}{\partial \theta_j} log(1 - h_\theta(x)) \\ &= \left(y_i - h_\theta(x^{(i)}) \right) x_j^{(i)} \\ &= \text{erreur } \times x_j^{(i)} \end{aligned} \tag{2}$$

On obtient donc la même formule que pour la régression linéaire mais avec une fonction h différente.

Régression logistique : cas multiclasse

Comment faire quand on a k **classes avec** $k > 2$?

On utilise la **fonction softmax** :

$$P(y = i|x, \theta) = \frac{e^{\theta_i^T x}}{\sum_{j=1}^{k} e^{\theta_j^T x}}$$

avec $y^{(i)} = [0 \ \ldots \ 0 \ 1 \ 0 \ldots \ 0]^T$ avec 1 à la $i$ème coordonnée.

Régression logistique : cas multiclasse

Pour chaque vecteur de données de test $x^{(i)}$, on calcule un vecteur de probabilités d'obtenir l'une des k classes.

$$h_\theta(x^{(i)}) = \begin{bmatrix} p(y^{(i)} = 1 | x^{(i)}; \theta) \\ p(y^{(i)} = 2 | x^{(i)}; \theta) \\ \vdots \\ p(y^{(i)} = k | x^{(i)}; \theta) \end{bmatrix} = \frac{1}{\sum_{j=1}^{k} e^{\theta_j^T x^{(i)}}} \begin{bmatrix} e^{\theta_1^T x^{(i)}} \\ e^{\theta_2^T x^{(i)}} \\ \vdots \\ e^{\theta_k^T x^{(i)}} \end{bmatrix}$$

Régression logistique : cas multiclasse

La fonction de coût devient :

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\sum_{j=1}^{k}\mathbb{I}(y^{(i)} = j) log \left(\frac{e^{\theta_j^T x^{(i)}}}{\sum_{p=1}^{k} e^{\theta_p^T x^{(i)}}} \right) \tag{3}$$

Le gradient s'écrit :

$$\nabla_{\theta_j} J(\theta) = -\frac{1}{m}\sum_{i=1}^{m} x^{(i)} \left(\mathbb{I}(y^{(i)} = j) - P(y^{(i)} = j | x^{(i)}; \theta) \right)$$