Cours 3 - La régression logistique

Neila Mezghani

10 janvier 2022

COLLÈGE DE
BOIS-DE-BOULOGNE

Plan du cours

Introduction

Plan du cours

Mise en contexte (1/3)

- Certains algorithmes de régression peuvent être utilisés pour la classification.
- La régression logistique (appelée également régression logit) permet de déterminer la probabilité qu'une observation appartienne à une classe particulière.
- La régression logistique fournit la probabilité du résultat (la logistique du résultat) au lieu du résultat lui même.

Mise en contexte (2/3)

- Autrement dit, dans la régression logistique, ce n'est pas la réponse binaire (malade/pas malade) qui est directement modélisée, mais la probabilité de réalisation d'une des deux modalités (être malade par exemple).
- Si la probabilité estimée est supérieure à 50%, alors le modèle prédit que l'observation appartient à cette classe en particulier : appelée classe positive, d'étiquette « 1 »
- Sinon il prédit que l'observation appartient à l'autre classe : appelée classe négative, d'étiquette « 0 »

Mise en contexte (3/3)

- Exemples d'applications du modèle de régression logistique binomiale :
  - Achat ou non d'un produit
  - Bon ou mauvais client
  - Echec ou succès dans un examen
  - Absence ou présence d'une pathologie

Plan du cours

Retour sur le modèle de régression linéaire (1/2)

Un modèle linéaire effectue une prédiction en calculant simplement une somme pondérée des variables d'entrée tout en y ajoutant un terme constant :

$$\hat{y} = \theta_0 + \theta_1 \mathbf{x}_1 + \theta_2 \mathbf{x}_2 + ... + \theta_n \mathbf{x}_n \quad (1)$$

avec

- $\hat{y}$ est la valeur prédite
- n est le nombre de variables
- θ_j est le j ème paramètre du modèle.

Retour sur le modèle de régression linéaire (2/2)

L'équation (1) peut s'écrire de manière générale (forme vectorielle) :

$$\hat{y} = h_{\boldsymbol{\theta}}(\mathbf{x}) = \boldsymbol{\theta} \cdot \mathbf{x} \tag{2}$$

dans cette équation :

- $\boldsymbol{\theta}$ est le vecteur des paramètres du modèle. Il regroupe à la fois θ_0 et les coefficients de pondération θ_1 à θ_n des variables.
- $\mathbf{x}$ est le vecteur des valeurs d'une observation, contenant les valeurs x_0 à x_n , où x_0 est toujours égal à 1.
- $\boldsymbol{\theta} \cdot \mathbf{x}$ est le produit scalaire de $\boldsymbol{\theta}$ et $\mathbf{x}$.

Vers la régression logistique...(1/2)

- La différence entre la régression linéaire et la régression logistique est que cette dernière fournit la probabilité du résultat (la logistique du résultat) au lieu du résultat lui même.
- La probabilité estimée par le modèle de régression logistique :

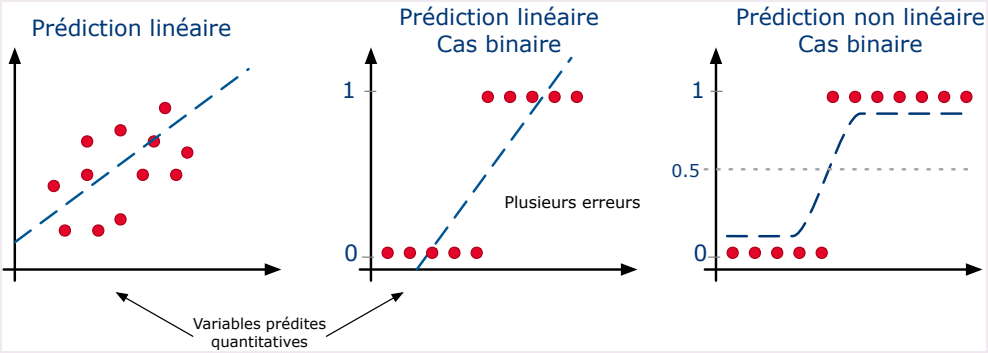
$$\hat{p} = h_{\boldsymbol{\theta}}(\mathbf{x}) = \sigma(\mathbf{x}^T \boldsymbol{\theta}) \tag{3}$$

σ est une fonction sigmoide

$\Longrightarrow$ La question qui se pose : Comment estimer ces probabilités ?

Vers la régression logistique...(2/2)

Résumé...



La régression logistique

Plan du cours

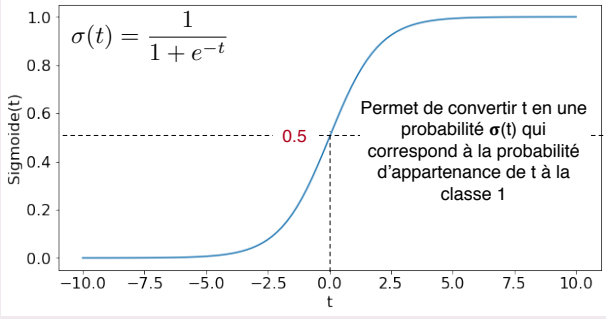
La fonction logistique (1/9)

- La fonction logistique $\sigma(t)$ est une fonction sigmoïde dont les valeurs sont comprises entre 0 et 1.

- La fonction logistique est définie par :

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

$$\sigma(t) \begin{cases} < 0.5 & \text{si } t < 0 \\ \geq 0.5 & \text{si } t \geq 0 \end{cases}$$



La fonction logistique (2/9)

- L'équation de régression linéaire est :

$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + ... + \theta_n x_n \quad (4)$$

- Si applique la fonction sigmoide, on obtient :

$$\hat{p}(x) = \frac{1}{1 + e^{-(\theta_0 + \theta_1 x_1 + ... + \theta_n x_n)}}$$

avec θ_0 le biais
et θ_1,...θ_n les coefficients de pondération des variables x

La fonction logistique (3/9)

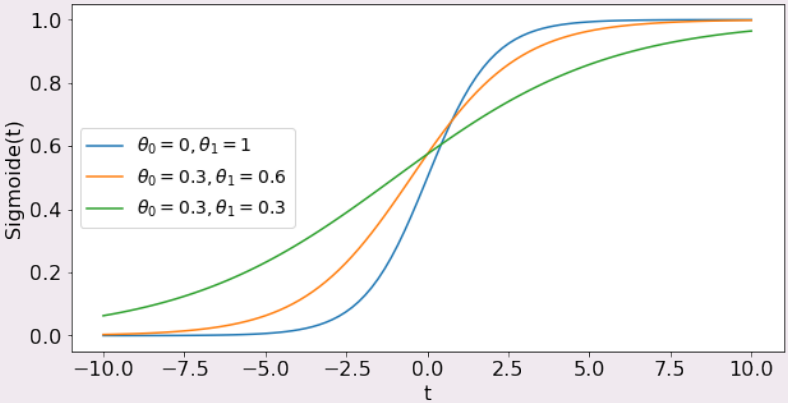
- Dans le cas d'une seule caractéristique x, l'équation de la courbe logistique est alors :

$$\hat{p}(x) = \frac{1}{1 + e^{-(\theta_0 + \theta_1 x)}}$$

avec θ_0 le biais
et θ_1 le coefficient de pondération de la variable x

La fonction logistique (4/9)

On obtient les différentes courbes logistiques pour différentes valeurs de θ_0 et θ_1



La fonction logistique (5/9)

- Le modèle de régression logistique permet d'estimer la probabilité qu'une observation x appartienne à la classe positive via la probabilité :

$$\hat{p} = h_{\boldsymbol{\theta}}(\mathbf{x}) = \sigma(\mathbf{x}^T \boldsymbol{\theta})$$

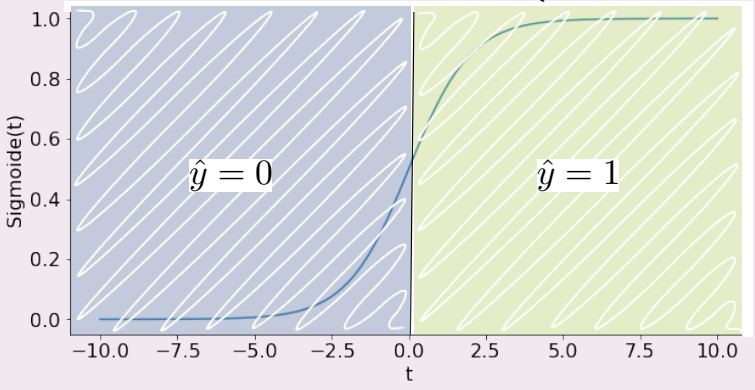
- Une fois la probabilité estimée, on peut faire une prédiction en se basant sur le modèle de régression logistique :

$$\hat{y} = \begin{cases} 0 & \text{si } \hat{p} < 0.5 \\ 1 & \text{si } \hat{p} \geq 0.5 \end{cases}$$

La fonction logistique (6/9)

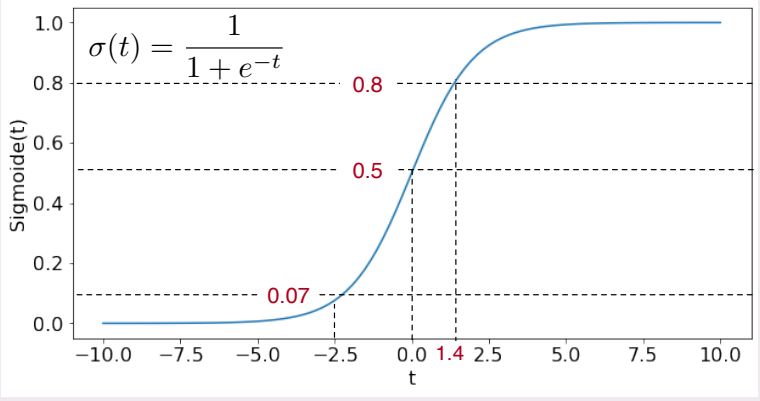
$$\sigma(t) = \frac{1}{1 + e^{-t}}$$

$$\hat{y} = \begin{cases} 0 & \text{si } \hat{p} < 0.5 \\ 1 & \text{si } \hat{p} \geq 0.5 \end{cases}$$



Exemples : Si $t = 1.4$ alors $\sigma(t) = 0.8 = 80\% \implies \hat{y} = 1$

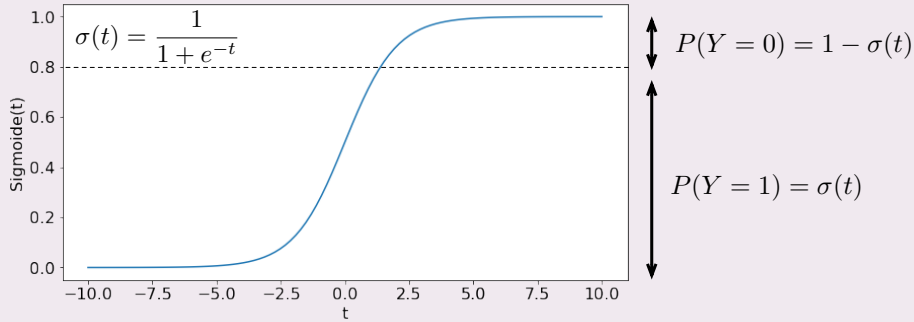
Si $t = -2.5$ alors $\sigma(t) = 0.07 = 7\% \implies \hat{y} = 0$



La fonction logistique (7/9)

La probabilité qu'une observation x appartienne à la classe positive suit une loi de Bernouilli :

$$P(Y = y) = \sigma(t)^y \times (1 - \sigma(t))^{1-y}$$

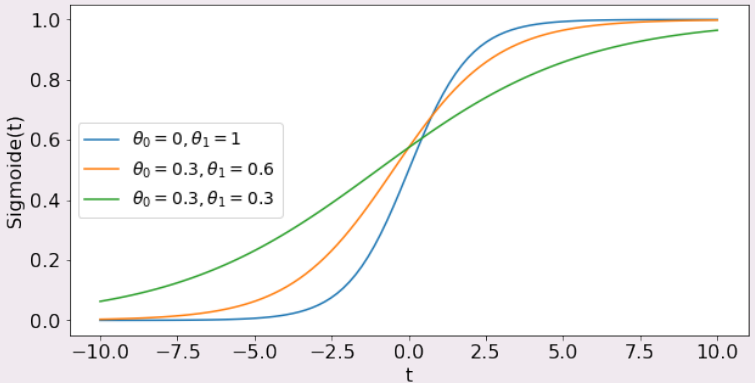


La fonction logistique (8/9)

- Le but est donc de déterminer le meilleur modèle possible c-à-d le modèle qui fait les plus petites erreurs entre les valeurs réelles et les valeurs prédites.
- Le modèle de régression logistique étant le modèle qui permet d'estimer la probabilité qu'une observation x appartienne à la classe positive via la probabilité :

$$\hat{p} = h_{\boldsymbol{\theta}}(\mathbf{x}) = \sigma(\mathbf{x}^T \boldsymbol{\theta})$$

La fonction logistique (9/9)



$\Longrightarrow$ besoin d'une fonction cout qui permet de mesurer les erreurs entre les valeurs réelles et les valeurs prédites.

Plan du cours

La fonction de coût : la fonction log loss

- La fonction coût mesurée sur l'ensemble des données d'entrainement que nous allons utilisé pour la régression logistique est la fonction log loss :

$$J(\boldsymbol{\theta}) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log(\hat{p}^{(i)}) + (1-y^{(i)})\log\left(1-\hat{p}^{(i)}\right)\right] \qquad (5)$$

avec

- $y^{(i)}$ la classe de l'observation i
- m le nombre d'observations
- $\hat{p}^{(i)}$ la probabilité que l'observation $x^{(i)}$ appartienne à la classe positive

Origine de la fonction log loss (1/5)

- En statistique, la vraisemblance (Likelihood) est un paramètre qui permet d'évaluer la performance d'un modèle par sa plausibilité vis-à-vis des données réelles

$$L = \prod_{i=1}^{m} P(Y = y_i) \tag{6}$$

- Pour m observations, en utilisant la loi de Bernouilli on obtient :

$$L = \prod_{i=1}^{m} P(Y = y_i) = \prod_{i=1}^{m} a_i^{y_i} \times (1 - a_i)^{1-y_i}$$

Origine de la fonction log loss (2/5)

- La vraisemblance étant un produit de probabilité = produit de nombres compris entre 0 et 1 $\implies$ le résultat risque de s'approcher trop de la valeur nulle.
- Pour éviter de converger vers 0, on utilise une fonction logarithmique pour transformer les produit en des sommes.

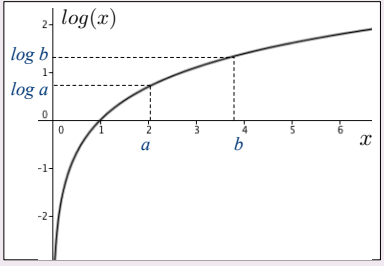
$$\log(L) = \log\left(\prod_{i=1}^{m} a_i^{y_i} \times (1 - a_i)^{1-y_i}\right) \tag{7}$$

Origine de la fonction log loss (3/5)

- La transformation en utilisant une fonction logartithmique ne pose pas de problème parce que c'est une fonction monotone croissante :

$$a < b \implies \log(a) < \log(b)$$

$\implies$ Déterminer le maximum de la vraissemblance revient à déterminer le maximum du log vraissemblance (LL).



Origine de la fonction log loss (4/5)

- On aura alors :

$$
\begin{aligned}
LL &= \log(L) \\
&= \log\left(\prod_{i=1}^{m} a_i^{y_i} \times (1 - a_i)^{1-y_i}\right) \\
&= \sum_{i=1}^{m} \log\left(a_i^{y_i} \times (1 - a_i)^{1-y_i}\right) \\
&= \sum_{i=1}^{m} y_i \log(a_i) + (1 - y_i) \log(1 - a_i)
\end{aligned}
$$

Origine de la fonction log loss (5/5)

- Rappelons qu'on cherche à maximiser la vraisemblance (Likelihood) ou bien à minimiser sa fonction négative ($-$ la fonction)
- On peut également multiplier par un facteur de normalisation (multiplier par $\frac{1}{m}$)
- On obtient alors la fonction log loss :

$$\mathcal{L} = -\frac{1}{m}\sum_{i=1}^{m} y_i \log(a_i) + (1 - y_i)\log(1 - a_i) \tag{8}$$

Comment faire pour minimiser cette fonction cout ?

Plan du cours

Entrainement (1/3)

- La fonction coût est donné par :

$$J(\boldsymbol{\theta}) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log(\hat{p}^{(i)}) + (1-y^{(i)})\log\left(1-\hat{p}^{(i)}\right)\right] \quad (9)$$

- L'objectif de l'entrainement est de déterminer la valeur de $\boldsymbol{\theta}$ qui minimise cette fonction $J(\boldsymbol{\theta})$?

$\Longrightarrow$ Comment déterminer la valeur de $\boldsymbol{\theta}$?

Entrainement (2/3)

- La fonction de coût log loss est convexe $\Longrightarrow$ l'identification d'un minimum global est assuré si on utilise l'algorithme de descente de gradient.
- Conditions :
  - Il faut choisir un taux d'apprentissage pas trop grand
  - Il faut choisir un nombre d'itération suffisamment grand

Entrainement (3/3)

- La dérivée partielle de la fonction de coût par rapport au j^{ieme} paramètre du modèle $\boldsymbol{\theta}_j$ peut être calculé selon :

$$\frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta}) = \frac{1}{m} \sum_{i=1}^{m} \underbrace{\left(\sigma(\boldsymbol{\theta}^T \mathbf{x}^{(i)}) - y^{(i)} \right)}_{\text{Erreur de prédiction}} \underbrace{x_j^{(i)}}_{\text{Valeur de la j ième variable}} \tag{10}$$

- Ensuite, on procède par une descente de gradient ordinaire ou stochastique ou mini-lots

Exemple sur la régression logistique (1/11)

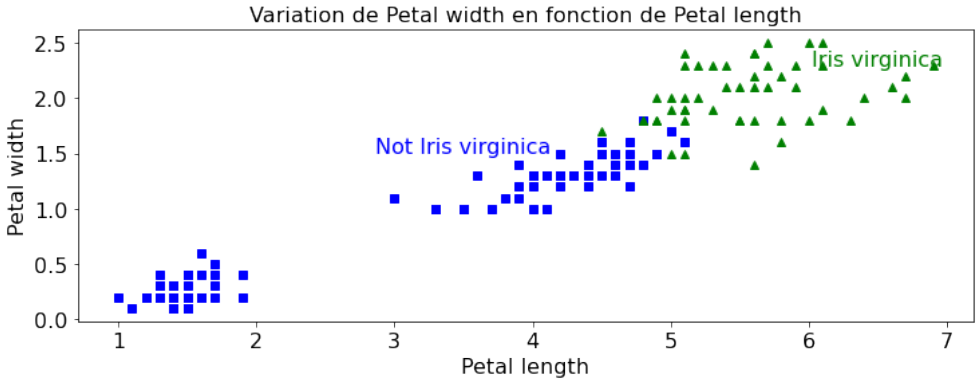
- Soit l'ensemble des données `Iris` qui comprend 150 observations de fleurs d'iris décrites par la longueur et la largeur des sépales et des pétales.
- Trois espèces différentes sont incluses Iris `setosa`, Iris `versicolor` et Iris `virginica`
- Pour illustrer la régression logistique, nous allons considérer la classe cible de l'espèce `virginica` en se basant sur la largeur du pétale (`Petal width`)

Exemple sur la régression logistique (2/11)

- La première étape sera d'affecter l'étiquette « 1 » aux observations dont la classe est `Iris virginica` et l'étiquette « 0 » aux autres classes `Not Iris virginica`.
- Ensuite nous entrainons le modèle de régression logistique en utilisant `sklearn.linear_model.LogisticRegression`

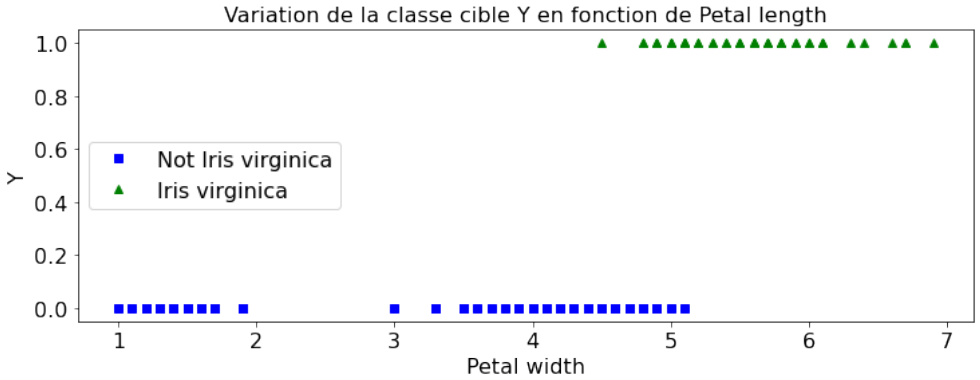
Exemple sur la régression logistique (3/11)

Variation de Petal width en fonction de Petal length



Exemple sur la régression logistique (4/11)

Variation de la classe cible Y en fonction de Petal length



Exemple sur la régression logistique (5/11)

Entrainement du modèle de régression logistique

```
from sklearn.linear_model import LogisticRegression
# instantiate the model
log_reg = LogisticRegression()
# fit the model with data
log_reg.fit(X, y)
```

Les paramètres du modèle (les coefficients) sont stockés dans `log_reg.coef_` `log_reg.intercept_` comprend la constante ajoutée à la fonction de décision.

Exemple sur la régression logistique (6/11)

Dans notre cas, on obtient : `log_reg.coef_` = 4.33
`log_reg.intercept_` = -7.19
Ce qui veut dire que la probabilité est estimé selon :

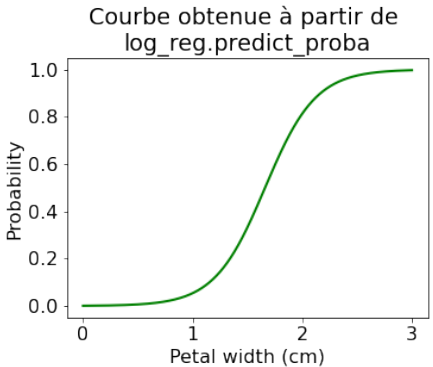
$$\hat{p} = \frac{1}{1 + e^{-(\theta_0 + \theta_1 x)}} = \frac{1}{1 + e^{-(-7.19 + 4.33x)}}$$

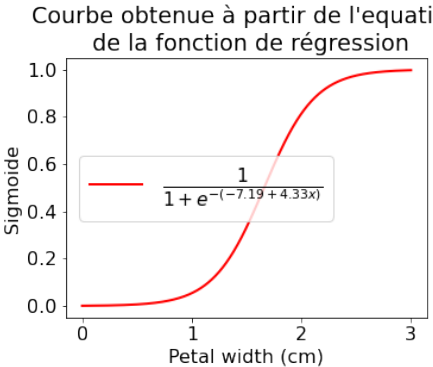
La probabilité peut être estimé également selon :

```
from sklearn.linear_model import LogisticRegression
X_new = np.linspace(0, 3, 1000).reshape(-1, 1)
y_proba = log_reg.predict_proba(X_new)
```

Exemple sur la régression logistique (7/11)

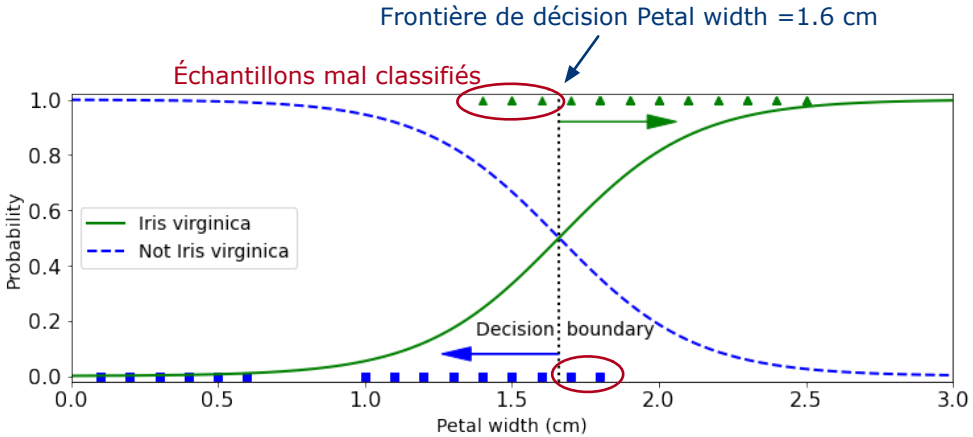
Génération d'un ensemble de valeurs aléatoires dans l'intervalle [0, 3] cm et calcul la valeur prédite de la probabilité pour représenter la courbe de régression



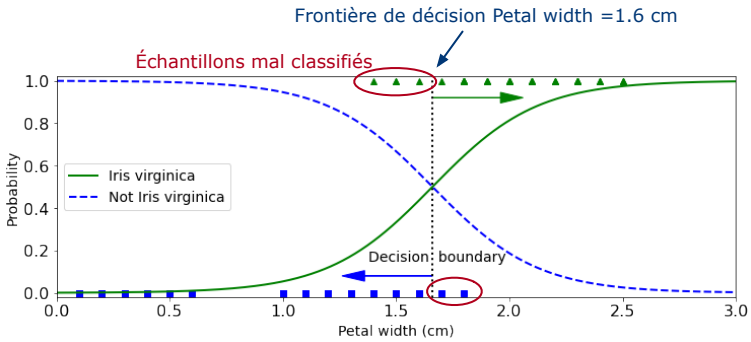


Exemple sur la régression logistique (8/11)

Probabilités estimées et frontière de décision avec la variable Petal width :



Exemple sur la régression logistique (9/11)

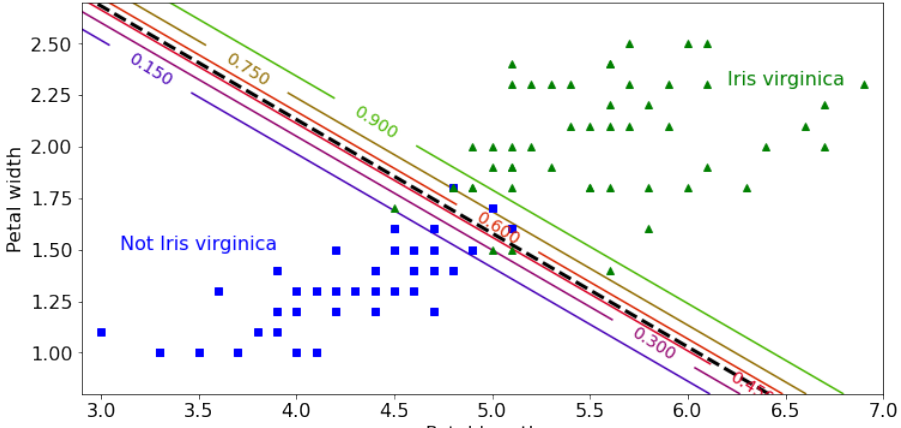


La fonction `log_reg.predict` permet de prédire la variable cible :

```
log_reg.predict([[1.7], [1.5], [2], [3]])
array([1, 0, 1, 1])
```

Exemple sur la régression logistique (10/11)

Frontière de décision linéaire avec les deux variables Petal width et Petal length

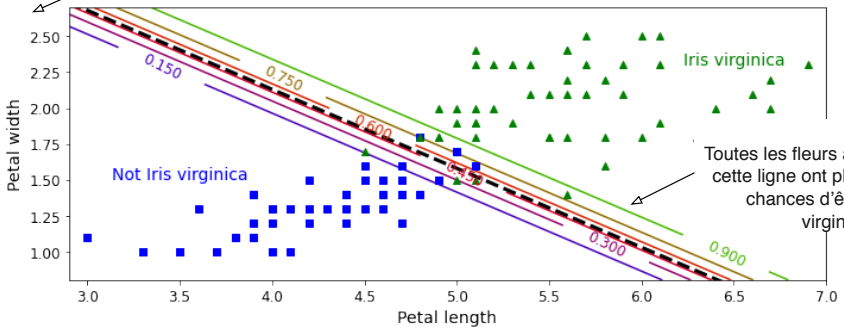


Exemple sur la régression logistique (11/11)

Frontière de décision avec les deux variables Petal width et Petal length

Chacune de ces lignes parallèles indique les observations où le modèle estime une probabilité donnée. Cette probabilité va de 15 % à 90 %.

La classe virginica étant la classe positive (Label =1)



Toutes les fleurs au-dessus de la cette ligne ont plus de 90 % de chances d'être des Iris virginica.

Régression softmax

Plan du cours

Principe de la régression softmax (1/4)

- La régression **softmax**, ou régression logistique multinomiale est considérée comme étant un modèle de régression logistique généralisé = Il permet de traiter plusieurs classes directement sans avoir à entraîner plusieurs classificateurs binaires puis à les combiner
- Étant donné une observation x, le modèle de régression softmax calcule d'abord un score $s_k(x)$ pour chaque classe k, puis estime la probabilité de chaque classe en appliquant aux scores la fonction softmax

$$s_k(x) = (\boldsymbol{\theta}^{(k)})^T \mathbf{x} \tag{11}$$

Principe de la régression softmax (2/4)

$$s_k(\mathbf{x}) = (\boldsymbol{\theta}^{(k)})^T \mathbf{x}$$

- Cette formule ressemble à celle qui permet de calculer la prédiction en régression linéaire
- La différence importante est que chaque classe possède un vecteur de paramètres $\boldsymbol{\theta}^{(k)}$ qui lui est spécifique.
- Tous ces vecteurs constituent les lignes de la matrice de paramètres Θ.

Principe de la régression softmax (3/4)

- Une fois le score de chaque classe calculé, la probabilité est estimé en appliquant aux scores la fonction softmax.

$$\hat{p}_k = \sigma(\boldsymbol{s}(\boldsymbol{x}))_k = \frac{\exp(\boldsymbol{s}_k(\boldsymbol{x}))}{\sum_{j=1}^{K} \exp(\boldsymbol{s}_j(\boldsymbol{x}))} \tag{12}$$

avec :

- K le nombre de classes. $j \in [1, K]$
- $s(x)$ un vecteur contenant les scores de chaque classe pour l'observation x.
- $\sigma(s(x))_k$ est la probabilité estimée que l'observation x appartienne à la classe k compte tenu des scores de chaque classe pour cette observation.

Principe de la régression softmax (4/4)

- À partir des probabilités d'appartenance à chaque classe, le classificateur de régression softmax prédit la classe ayant la plus forte probabilité estimée :

$$\hat{y} = \arg\max_k \sigma(s(\mathbf{x}))_k = \arg\max_k s_k(\mathbf{x}) = \arg\max_k \left((\boldsymbol{\theta}^{(k)})^T \mathbf{x} \right) \quad (13)$$

$\arg\max$ renvoie la valeur de k qui maximise la probabilité estimée $\sigma(s(\mathbf{x}))_k$

Plan du cours

La fonction cout : L'entropie croisée (1/2)

- L'objectif est d'avoir un modèle qui estime une forte probabilité pour la classe ciblée et par conséquent de faibles probabilités pour les autres classes.
- En statistique, on utilise fréquemment l'entropie croisée (cross entropy) pour mesurer l'adéquation entre un ensemble de probabilités estimées d'appartenance à des classes et les classes ciblées.

La fonction cout : L'entropie croisée (2/2)

- La fonction de coût d'entropie croisée est donné par :

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} \log\left(\hat{p}_k^{(i)}\right) \tag{14}$$

avec

- $y_k^{(i)}$ la probabilité cible que la i ème observation appartienne à la classe k

Plan du cours

Entrainement

- L'objectif de l'entrainement est de déterminer la matrice Θ qui minimise la fonction cout.
- Le vecteur gradient par rapport à $\boldsymbol{\theta}(k)$ de la fonction de coût d'entropie croisée est donné par :

$$\nabla_{\boldsymbol{\theta}^{(k)}} = \frac{1}{m} \sum_{i=1}^{m} \left(\hat{p}_k^{(i)} - y_k^{(i)} \right) \mathbf{x}^{(i)} \tag{15}$$

- Une fois le vecteur gradient de chaque classe déterminé, on peut utiliser la descente de gradient pour déterminer la matrice Θ

Exemple sur la régression logistique softmax (1/6)

- Pour illustrer la régression logistique multinomiale, nous allons considérer les trois espèces différentes d'Iris `setosa`, Iris `versicolor` et Iris `virginica`
- On considère les deux variables Petal width et Petal length

Exemple sur la régression logistique softmax (2/6)

L'entrainement du modèle de régression logistique softmax se fait avec `LogisticRegression` en fixant l'hyperparamètre `multi_class` à la valeur `multinomial`

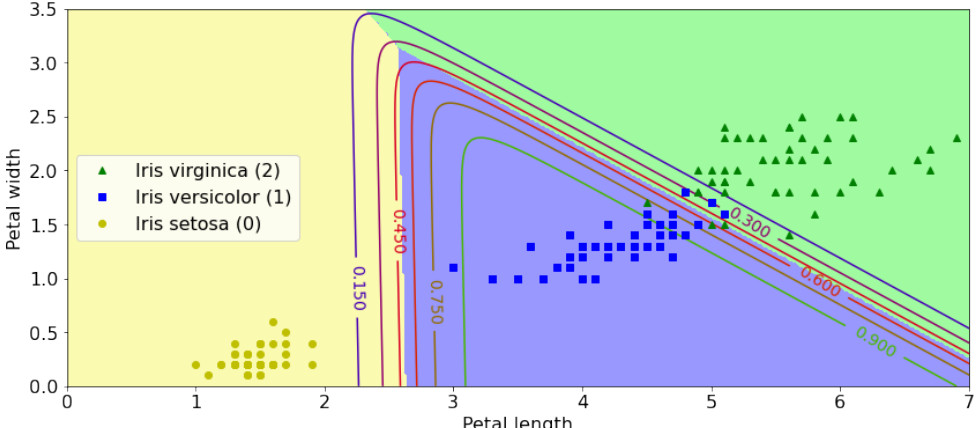
```
from sklearn.linear_model import LogisticRegression
X = iris["data"][:, (2, 3)]
y = iris["target"]
softmax_reg = LogisticRegression(multi_class="multinomial",
solver="lbfgs", C=10, random_state=42)
softmax_reg.fit(X, y)
```

Exemple sur la régression logistique softmax (3/6)

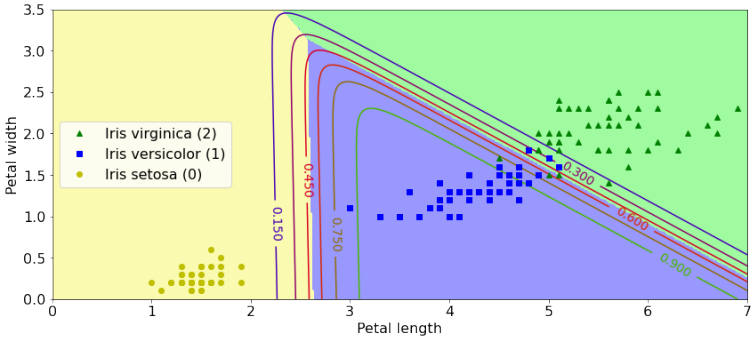
- Pour illustrer la régression logistique multinomiale, nous allons considérer les trois espèces différentes d'Iris `setosa`, Iris `versicolor` et Iris `virginica`
- On considère les deux variables Petal width et Petal length

Exemple sur la régression logistique softmax (4/6)

Frontières de décision avec les deux variables Petal width et Petal length (différentes zones)



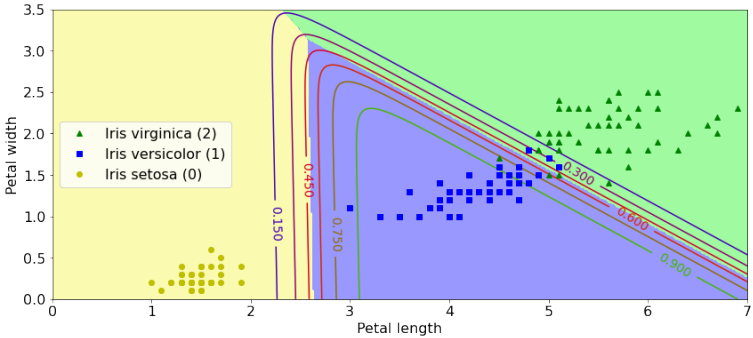
Exemple sur la régression logistique softmax (5/6)



La fonction `log_reg.predict` permet de prédire la variable cible :

```
softmax_reg.predict([[5, 2],[4, 1], [1, 2], [7, 3], [7, 0.5]])
array([2, 1, 0, 2, 2])
```

Exemple sur la régression logistique softmax (6/6)



Les probabilités pour la classe Iris versicolor est représentées par des courbes. Exemple : la ligne étiquetée 0.450 représente la frontière des 45 % de probabilité.

A l'intersection des 3 zones, la probabilité est 1/3=33%