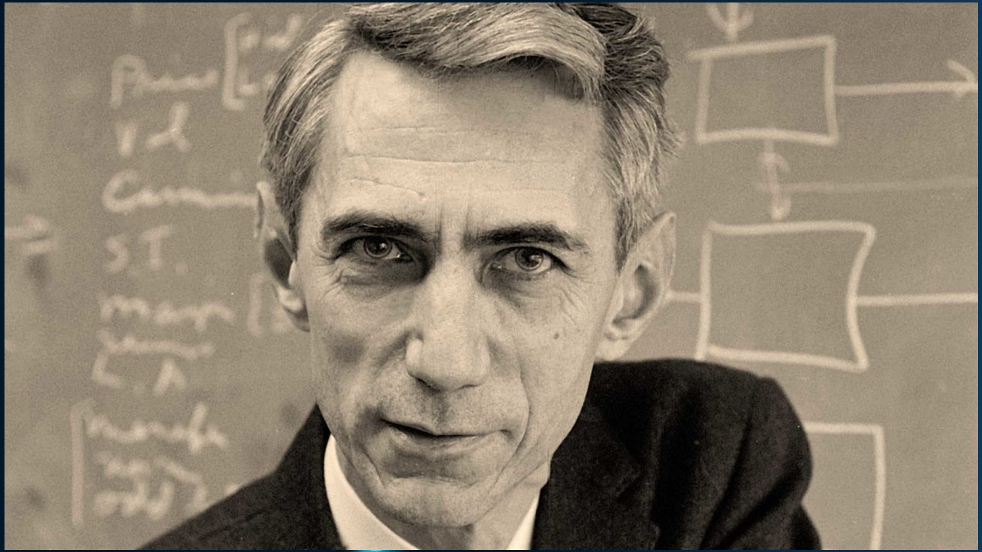
IA Géosciences - Deep learning: Classification avec des réseaux de neurones

Romain Wenger (Laboratoire Image Ville Environnement)

Table des matières

Table des matières

Claude Shannon a introduit la notion d'entropie



Définition de l'entropie (en théorie d'information)

L'entropie de Shannon, due à Claude Shannon, est une fonction mathématique qui, intuitivement, correspond à la quantité d'information contenue ou délivrée par une source d'information. Cette source peut être un texte écrit dans une langue donnée, un signal électrique ou encore un chier informatique quelconque (collection d'octets).

⇒ Permet de mesurer la quantité d'information pour un évènement donné

Exemple pratique pour l'entropie

Supposons qu'on a deux pièces de monnaie :

1. équitable : donc chaque face (A, B) a une probabilité égale à 0.5
2. truquée : par exemple la face A a une probabilité de 0.2 et B de 0.8

En jetant les deux pièces, laquelle fournie le plus d'information ?

1. équitable ⇒ on ne peut pas prédire le résultat
   - Donc on sera surpris pour la face A et la face B
2. $P(B) = 0{,}8$ ⇒ on peut prédire en avance
   - Donc on sera moins surpris pour face B et plus surpris A

Alors la première pièce nous fournit plus d'information

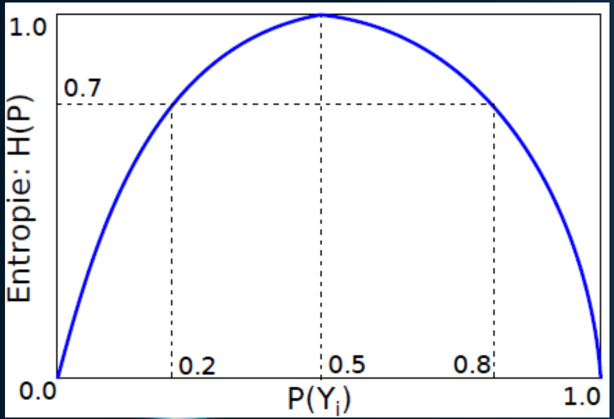
Formule mathématique pour l'Entropie

$$H(P) = -\sum_{c} P(Y_c) * \log_2(P(Y_c))$$

La première pièce juste :

$$P(Y_1) = P(A) = 0,5$$
$$P(Y_2) = P(B) = 0,5$$
$$H(P)$$
$$= -0,5\log_2(0,5)$$
$$-\ 0,5\log_2(0,5)$$

$$\Rightarrow H(P) = 1$$



La deuxième pièce truquée :

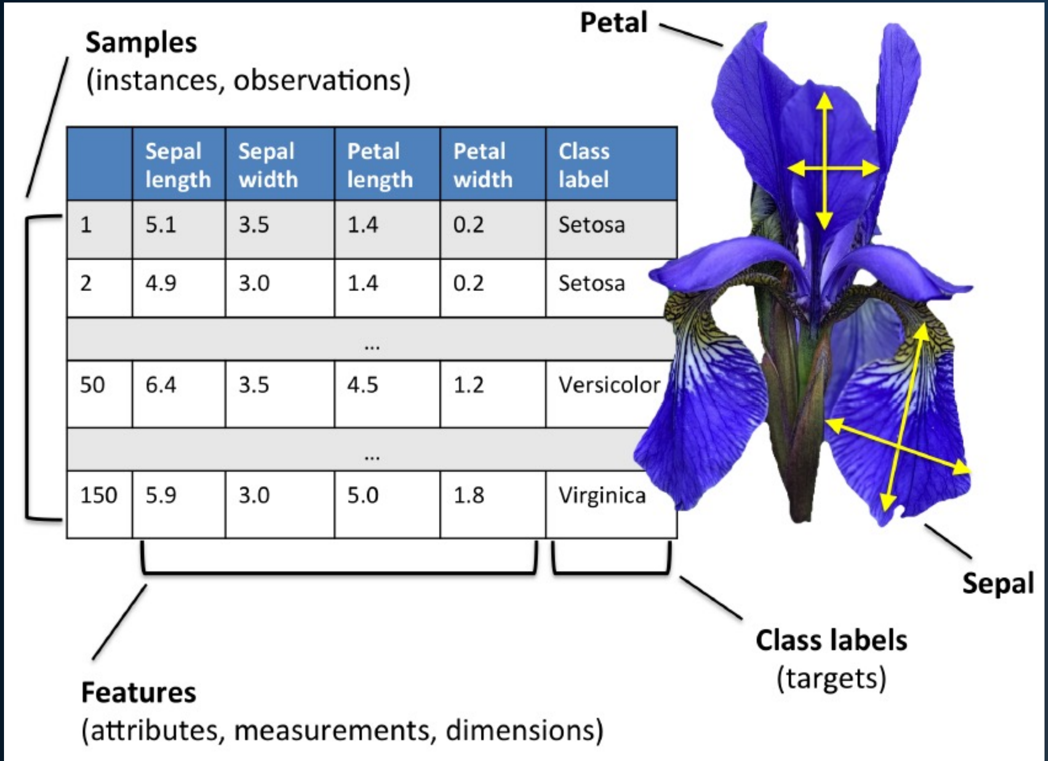
$$P(Y_1) = P(A) = 0,2$$
$$P(Y_2) = P(B) = 0,8$$
$$H(P)$$
$$= -0,2\log_2(0,2)$$
$$-\ 0,8\log_2(0,8)$$

$$\Rightarrow H(P) \approx 7$$

Apprentissage supervisé

- Deux types d'apprentissage automatique :
  - Non supervisé (on ne connaît pas à l'avance ce qu'on prédit)
  - Supervisé (on connaît à l'avance ce qu'on prédit)
- Deux tâches pour l'apprentissage supervisé :
  - Classification (catégorisation des images en chien, chat, voiture, etc.)
  - Régression (prédiction de la température, prix d'une maison, etc.)
- **Régression** : prédiction des prix des maisons
- **Classification** : catégorisation des fleurs Iris à partir de la longueur/largeur des pétales/sépales

Aperçu du jeu de données : Iris



	Sepal length	Sepal width	Petal length	Petal width	Class label
1	5.1	3.5	1.4	0.2	Setosa
2	4.9	3.0	1.4	0.2	Setosa
...					
50	6.4	3.5	4.5	1.2	Versicolor
...					
150	5.9	3.0	5.0	1.8	Virginica

Table des matières

Classification binaire: Iris-Setosa ou Pas

Catégorie	SL (Sepal Length)	SW (Sepal Width)	PL (Petal Length)	PW (Petal Width)
Setosa (0)	5.1	3.5	1.4	0.2
versicolor (1)	6.4	3.5	4.5	1.2
Virginica (2)	5.9	3.0	5.0	1.8
⋮	⋮	⋮	⋮	⋮

Aperçu des données Iris

Catégorie	SL (Sepal Length)	SW (Sepal Width)	PL (Petal Length)	PW (Petal Width)
Setosa (0)	5.1	3.5	1.4	0.2
Non-Setosa (1)	6.4	3.5	4.5	1.2
Non-Setosa (1)	5.9	3.0	5.0	1.8
⋮	⋮	⋮	⋮	⋮

Données Iris qui nous intéressent pour la classification binaire

Apprentissage supervisé

- Pré-traitement des données :
  - Normaliser les données pour que tout soit entre 0 et 1
  - Transformer les catégories: (chaîne de caractères) ⇒ (entiers)
    - Iris-Setosa ⇒ 0
    - Iris-versicolor (non-Setosa) ⇒ 1
    - Iris-Virginica (non-Setosa) ⇒ 1
  - Diviser en données d'entraînement (train) et de test

Etapes à suivre

- Etapes à suivre :
  - Définir notre modèle (hypothèse)
  - Définir notre objectif (fonction de coût – *cost function*)
  - Minimiser la fonction de coût avec la descente du gradient

Définir notre modèle: en fonction des données

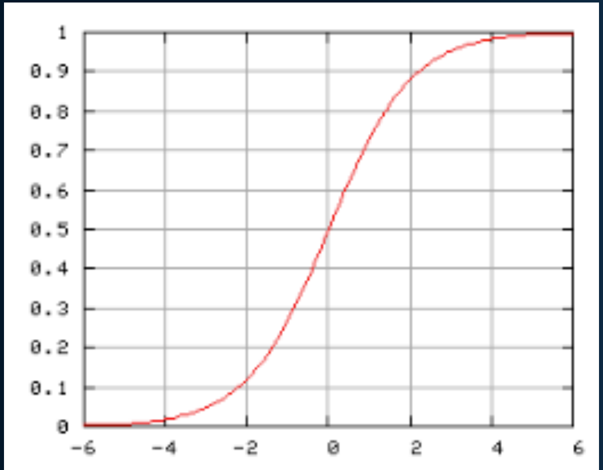
Y (Catégorie)	X_1 (Sepal Length)	X_2 (Sepal Width)	X_3 (Petal Length)	X_4 (Petal Width)
0 (Setosa)	5.1	3.5	1.4	0.2
1 (non-Setosa)	6.4	3.5	4.5	1.2
1 (non-Setosa)	5.9	3.0	5.0	1.8
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

But : Prédire la catégorie Y en fonction de $X = (X_1, X_2, X_3, X_4)$

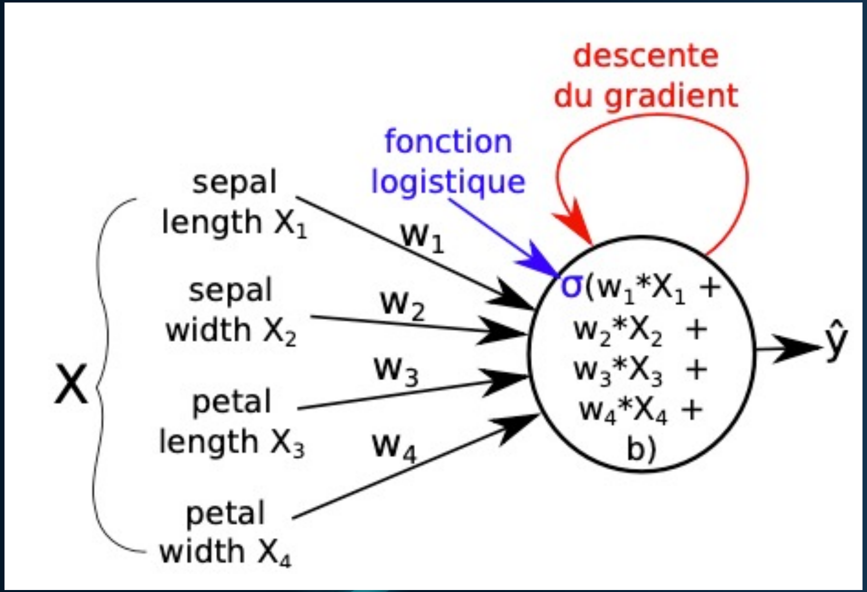
- On sait qu'on veut prédire soit 0 (Setosa) soit 1 (Non-Setosa)
- Probabiliste : prédire $\hat{Y}$ la probabilité que ça soit classe 0 ou 1
- Probabilité ⇒ valeur réelle entre 0 et 1
- Choisir une hypothèse qui satisfait la condition $0 \leq \hat{y} = h(x) \leq 1$
- Problème : la régression linéaire ne satisfait pas cette condition

Définir notre modèle : fonction logistique (sigmoid)

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



Définir notre modèle : régression logistique

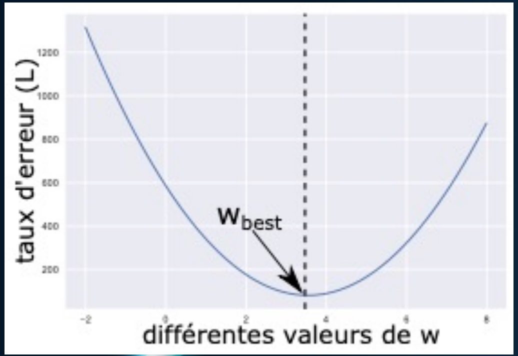


Définir notre objectif : l’erreur quadratique moyenne ?

Pour la régression linéaire

$$L(w) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i(w))^2 \mid \hat{y}_i = W * X_i + b$$

La forme de la courbe est convexe dans ce cas là



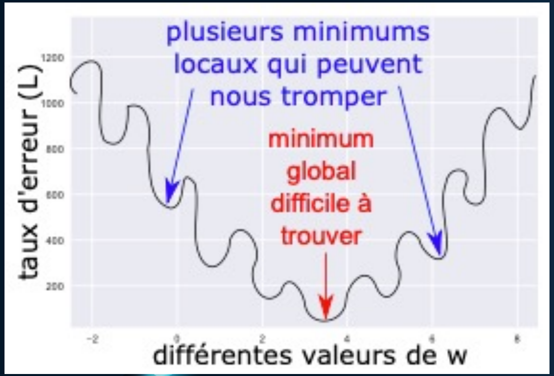
⇒ Il est facile de trouver le minimum avec la descente du gradient

Définir notre objectif : l’erreur quadratique moyenne ?

L’erreur quadratique moyenne pour la régression logistique donne :

$$L(w) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i(w))^2 \mid \hat{y}_i = \sigma(W * X_i + b)$$

σ étant non linéaire, on a une fonction de coût non convexe



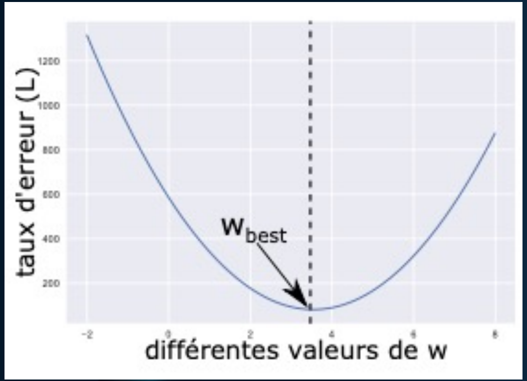
⇒ Il faut définir une nouvelle fonction de coût qui est convexe

Définir notre objectif : Entropie croisée

L'entropie croisée « Cross-Entropy » est la bonne fonction de coût

$$L(w) = \frac{1}{n}\sum_{i=1}^{n} L_i(w) \mid L_i(w) = -\sum_c y_{ic} * \log(\hat{y}_{ic})$$

Pour une classification binaire, on a $c = 2 \Rightarrow \hat{y}_{ic=0} = 1 - \hat{y}_{ic=1}$



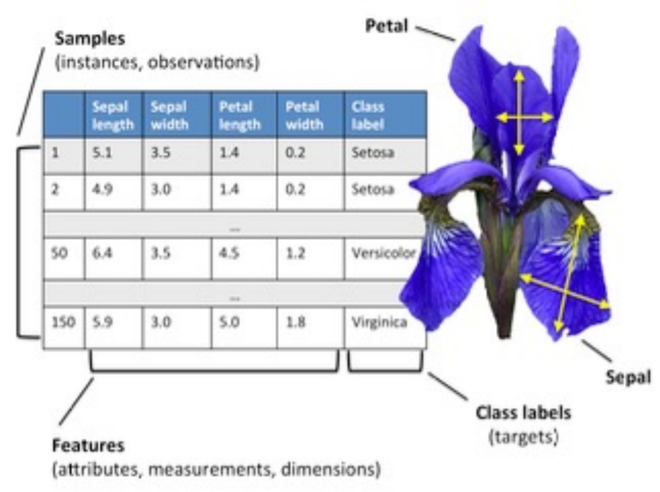
⇒ fonction convexe facile à optimiser sans minimums locaux

Définir notre objectif : pourquoi l'entropie croisée ?

$$L(w) = \frac{1}{n}\sum_{i=1}^{n} L_i(w)$$

$$L_i(w) = -y_i \log(\hat{y}_i) - (1 - y_i)\log(1 - \hat{y}_i)$$

- y_i étant la classe réelle de la fleur numéro i
- $\hat{y}_i$ étant la classe prédite de la fleur numéro i



Définir notre objectif : pourquoi l'entropie croisée ?

$$L_i(w) = -y_i \log(\hat{y}_i) - (1 - y_i) \log(1 - \hat{y}_i)$$

Notre but est de maximiser la précision

- Si c'est Setosa ($y_i = 0$) et la prédiction est Setosa ($\hat{y}_i = 0$)
  - Le coût $L(w)$ doit être nul (prédiction correcte)
  - C'est le cas : $L_i(w) = -0 \log(0) - (1 - 0) \log(1 - 0) = 0$
- Si c'est non-Setosa ($y_i = 1$) et la prédiction est non-Setosa ($\hat{y}_i = 1$)
  - Le coût $L(w)$ doit être nul (prédiction correcte)
  - C'est le cas : $L_i(w) = -1 \log(1) - (1 - 1) \log(1 - 1) = 0$
-

Définir notre objectif : pourquoi l'entropie croisée ?

$$L_i(w) = -y_i \log(\hat{y}_i) - (1 - y_i) \log(1 - \hat{y}_i)$$

Notre but est de maximiser la précision

- Si c'est Setosa ($y_i = 0$) et la prédiction est non-Setosa ($\hat{y}_i = 1$)
  - Le coût $L(w)$ doit être au maximum (prédiction incorrecte)
  - C'est le cas : $L_i(w) = -0 \log(1) - (1 - 0) \log(1 - 1) \approx \infty$
- Si c'est non-Setosa ($y_i = 1$) et la prédiction est non-Setosa ($\hat{y}_i = 0$)
  - Le coût $L(w)$ doit être au maximum (prédiction incorrecte)
  - C'est le cas : $L_i(w) = -1 \log(0) - (1 - 1) \log(1 - 0) \approx \infty$

Définir notre objectif : pourquoi l'entropie croisée ?

On veut minimiser la quantité d'information que le classifieur nous apporte étant donnée la vérité terrain (*ground truth*)

Par exemple si on sait qu'en réalité la classe d'un individu $X_i$est $y_i = A$

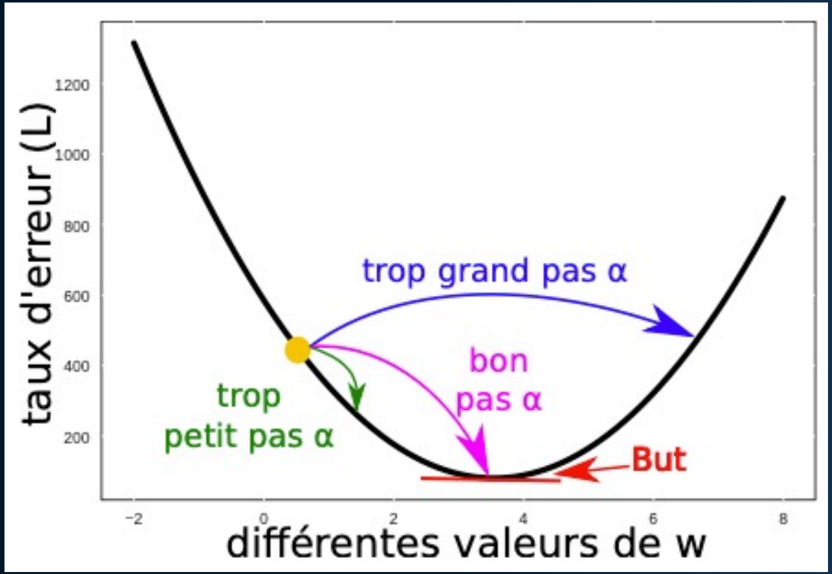
- La prédiction $\hat{y} = A$ ne nous apporte pas d'information
  - Petite entropie pour une prédiction correcte
- La prédiction $\hat{y} = B$ nous apporte une nouvelle information
  - Grande entropie pour une prédiction incorrecte

⇒ **Minimiser l'entropie pour toutes les classes c**

$$\boldsymbol{L_i(w) = -\sum_c y_{ic} * \log(\hat{y}_{ic})}$$

Optimiser : descente du gradient

Maintenant on a une fonction de coût qu'on peut minimiser



Optimiser : descente du gradient

Le modèle : régression logistique

$$\hat{y} = \sigma(W * X + b)$$

La fonction de coût : entropie croisée

$$L(w) = -y\log(\hat{y}) - (1-y)\log(1-\hat{y})$$

Algorithme d'optimisation : descente du gradient

$$W = W - \alpha(\frac{\theta L(W)}{\theta W})$$

⇒ La méthode de descente de gradient ne change pas

Uniquement le calcul de la dérivée $\frac{\theta L(W)}{\theta W}$ change

Optimiser : la dérivée pour l'entropie croisée

$$L(w) = -y\log(\hat{y}) - (1-y)\log(1-\hat{y})$$

On veut calculer la dérivée de L

$$\frac{\theta L(w)}{\theta w} = \frac{\theta[-y\log(\hat{y}) - (1-y)\log(1-\hat{y})]}{\theta w}$$

Or y est indépendante de $w \Rightarrow \frac{\theta y}{\theta w} = 0$

$$\frac{\theta L(w)}{\theta w} = -\mathrm{y}\frac{\theta\log(\hat{y})}{\theta w} - (1-y)\frac{\theta\log(1-\hat{y})}{\theta w}$$

Or $\frac{\theta\log(\hat{y})}{\theta w} = \frac{\frac{\theta\hat{y}}{\theta w}}{\hat{y}}$

$$\Rightarrow \frac{\theta L(w)}{\theta w} = -y\frac{\frac{\theta\hat{y}}{\theta w}}{\hat{y}} + (1-y)\frac{\frac{\theta\hat{y}}{\theta w}}{1-\hat{y}}$$

Donc maintenant nous devons trouver $\frac{\theta\hat{y}}{\theta w}$

Composées de fonctions

$$(\log(u))'(x) = \frac{u'(x)}{u\ (x)}$$

$$\Leftrightarrow (\log(u))'(x) = u'(x) * \frac{1}{u(x)}$$

Optimiser : la dérivée pour l'entropie croisée

$$\hat{y} = \sigma(z) \mid z = w * X + b$$

On veut calculer la dérivée de $\hat{y}$

$$\frac{\theta \hat{y}}{\theta w} = \frac{\theta \sigma(z)}{\theta w}$$

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

$$\frac{\theta \sigma(z)}{\theta w} = (-1) \frac{\theta (1 + e^{-z})}{\theta w} (1 + e^{-z})^{-2}$$

Or $\frac{\theta (1 + e^{-z})}{\theta w} = \frac{\theta e^{-z}}{\theta w}$

Donc on veut calculer $\frac{\theta e^{-z}}{\theta w}$

Optimiser : la dérivée pour l’entropie croisée

On veut calculer la dérivée

$$\frac{\theta e^{-z}}{\theta w}$$

D’après le théorème de dérivation des fonctions composées

$$\frac{\theta e^{-z}}{\theta w} = \frac{\theta e^{-z}}{\theta z}\frac{\theta z}{\theta w}$$

$$\frac{\theta e^{-z}}{\theta z} = -e^{-z}$$

Or $z = w * x + b$

$$\Rightarrow \frac{\theta z}{\theta w} = x$$

$$\Rightarrow \frac{\theta e^{-z}}{\theta w} = -x * e^{-z}$$

Optimiser : la dérivée pour l'entropie croisée

On revient à notre objectif original

$$\frac{\theta L(w)}{\theta w} = -y\frac{\frac{\theta \hat{y}}{\theta w}}{\hat{y}} + (1-y)\frac{\frac{\theta \hat{y}}{\theta w}}{1-\hat{y}}$$

En remplaçant et en simplifiant …

$$\frac{\theta \hat{y}}{\theta w} = \frac{\theta \sigma(z)}{\theta w} = \frac{x * e^{-z}}{(1+e^{-z})^{Z}} = x * \sigma(z)\big(1-\sigma(z)\big) = x\hat{y}(1-\hat{y})$$

$$\frac{\theta L(w)}{\theta w} = (\hat{y}-y) * x$$

En réalité on fait la somme sur l'ensemble d'entrainement n

$$\frac{\theta L(w)}{\theta w} = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i) * x_i$$

Optimiser : mettre w et b à jour

Maintenant avec la descente du gradient on peut mettre w à jour

$$\boldsymbol{w} = \boldsymbol{w} - \boldsymbol{\alpha} \frac{\theta L(w)}{\theta w}$$

En remplaçant …

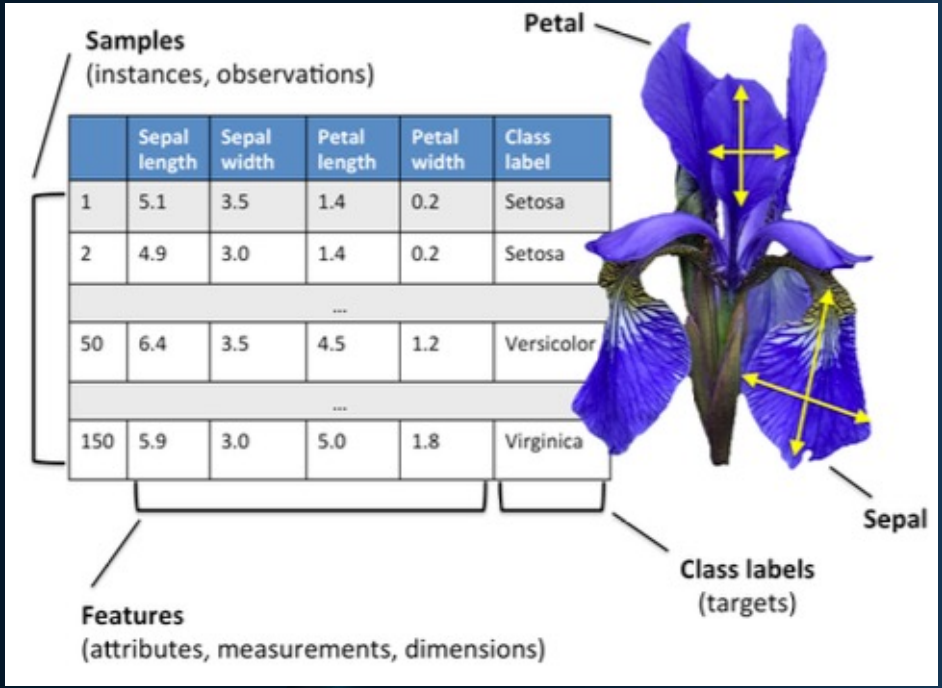
$$\boldsymbol{w} = \boldsymbol{w} - \frac{\boldsymbol{\alpha}}{n} \sum_{i=1}^{n} (\hat{y}_i - y_i) * x_i$$

Pour b on ne va pas répéter tout, mais en suivant la même démarche que pour w on aura

$$\boldsymbol{b} = \boldsymbol{b} - \frac{\boldsymbol{\alpha}}{\boldsymbol{n}} \frac{\theta L(b)}{\theta b}$$

$$\boldsymbol{b} = \boldsymbol{b} - \frac{\boldsymbol{\alpha}}{n} \sum_{i=1}^{n} (\hat{y}_i - y_i)$$

Exemple de classification binaire : Iris Setosa



	Sepal length	Sepal width	Petal length	Petal width	Class label
1	5.1	3.5	1.4	0.2	Setosa
2	4.9	3.0	1.4	0.2	Setosa
...					
50	6.4	3.5	4.5	1.2	Versicolor
...					
150	5.9	3.0	5.0	1.8	Virginica

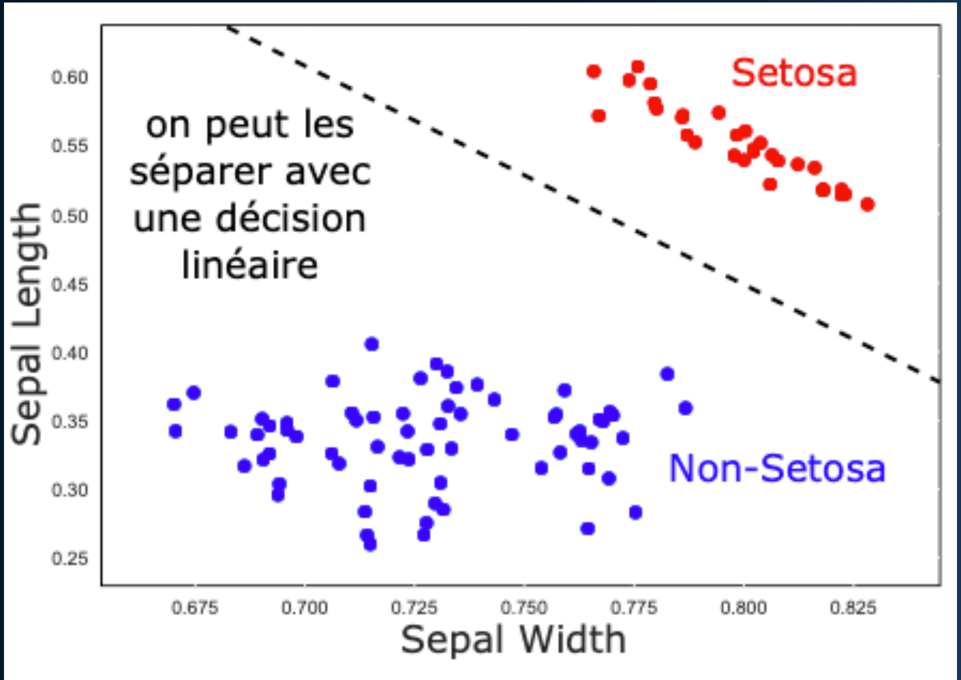
Exemple de classification binaire : Iris Setosa

Y (Catégorie)	X_1 (Sepal Length)	X_2 (Sepal Width)	X_3 (Petal Length)	X_4 (Petal Width)
0 (Setosa)	5.1	3.5	1.4	0.2
1 (non-Setosa)	6.4	3.5	4.5	1.2
1 (non-Setosa)	5.9	3.0	5.0	1.8
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

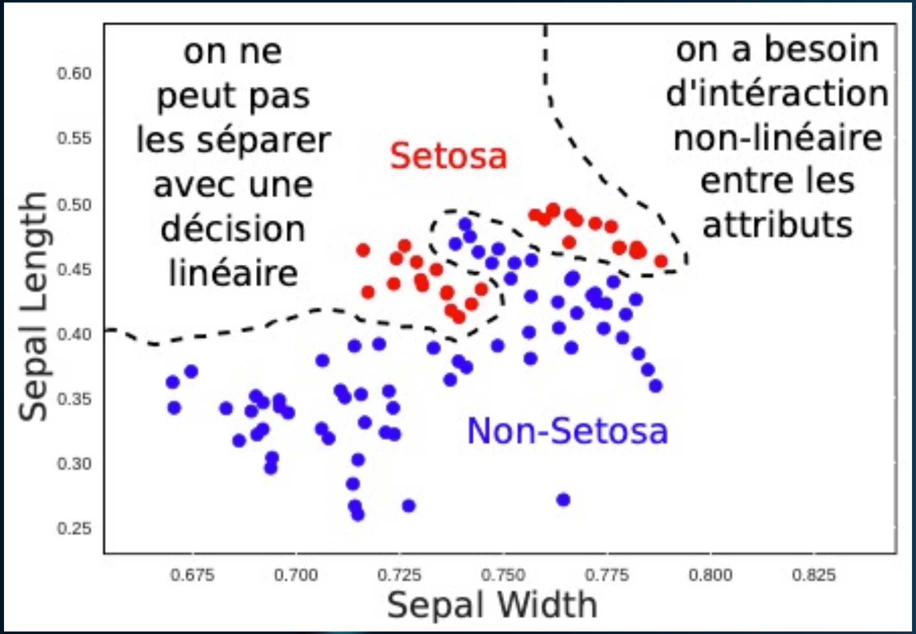
On simplifie pour utiliser uniquement deux variables (Sepal)

Y (Catégorie)	X_1 (Sepal Length)	X_2 (Sepal Width)
0 (Setosa)	5.1	3.5
1 (non-Setosa)	6.4	3.5
1 (non-Setosa)	5.9	3.0
$\vdots$	$\vdots$	$\vdots$

Validation des données 2D (décision linéaire)



Décision non linéaire (exemple)



Probabilité d’affectation d’une classe

La sortie de notre neurone est une probabilité d’appartenir à 0 ou 1

$$\hat{y}_i = \sigma(W * X_i + b)$$

Pour chercher la classe prédite il suffit de seuiller à 0.5

- Si la probabilité $\hat{y} > 0.5 \Rightarrow$ la classe prédite est 1
- Si la probabilité $\hat{y} \leq 0.5 \Rightarrow$ la classe prédite est 0

Mesure d'accuracy

L'entropie est un peu difficile à interpréter lors de l'évaluation

$$L(w) = \frac{1}{n}\sum_{i=1}^{n} -\sum_{c} y_{ic} * \log(\hat{y}_{ic})$$

Une autre mesure de performance pour un modèle est l'accuracy

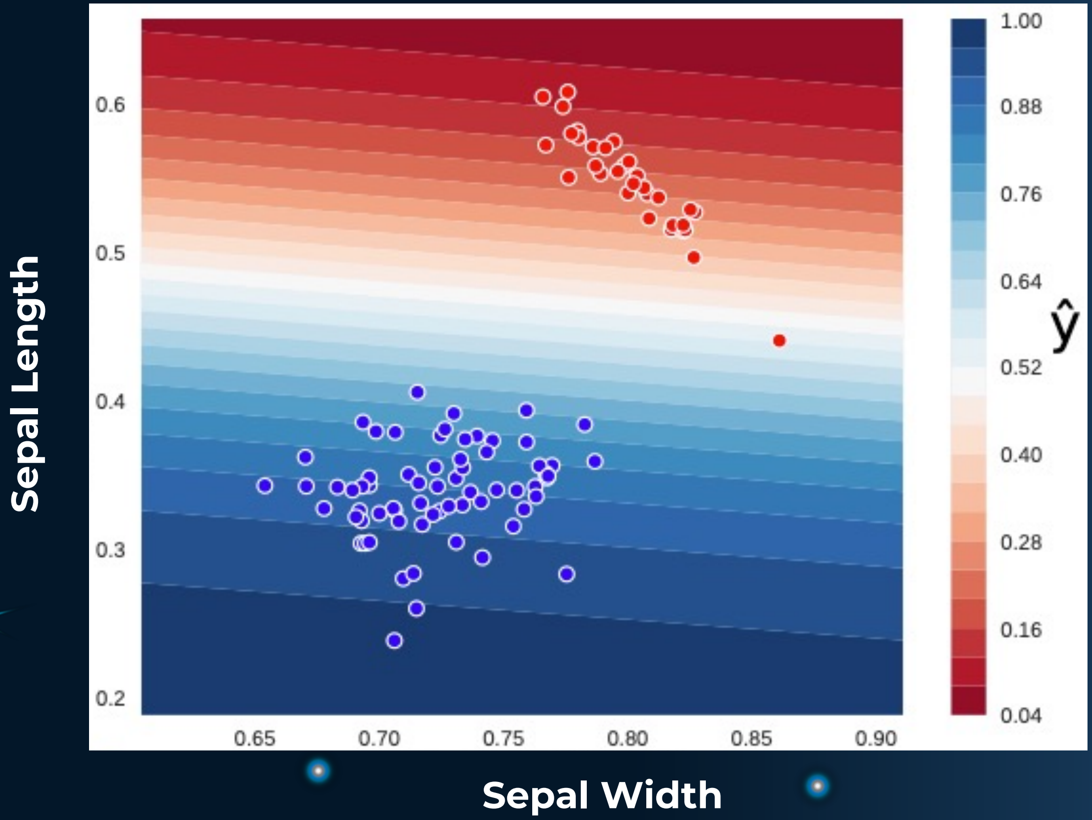
$$accuracy = \frac{nombre\ d'instances\ prédites\ correctement}{nombre\ total\ d'istances}$$

Par exemple pour 100 fleurs, si on prédit le type correct pour 76 fleurs alors

$$\Rightarrow accuracy = \frac{76}{100} = 76\%$$

Cette mesure est toujours entre 0 et 1

Décision binaire probabiliste



Décision binaire non probabiliste (Affectation des fleurs)

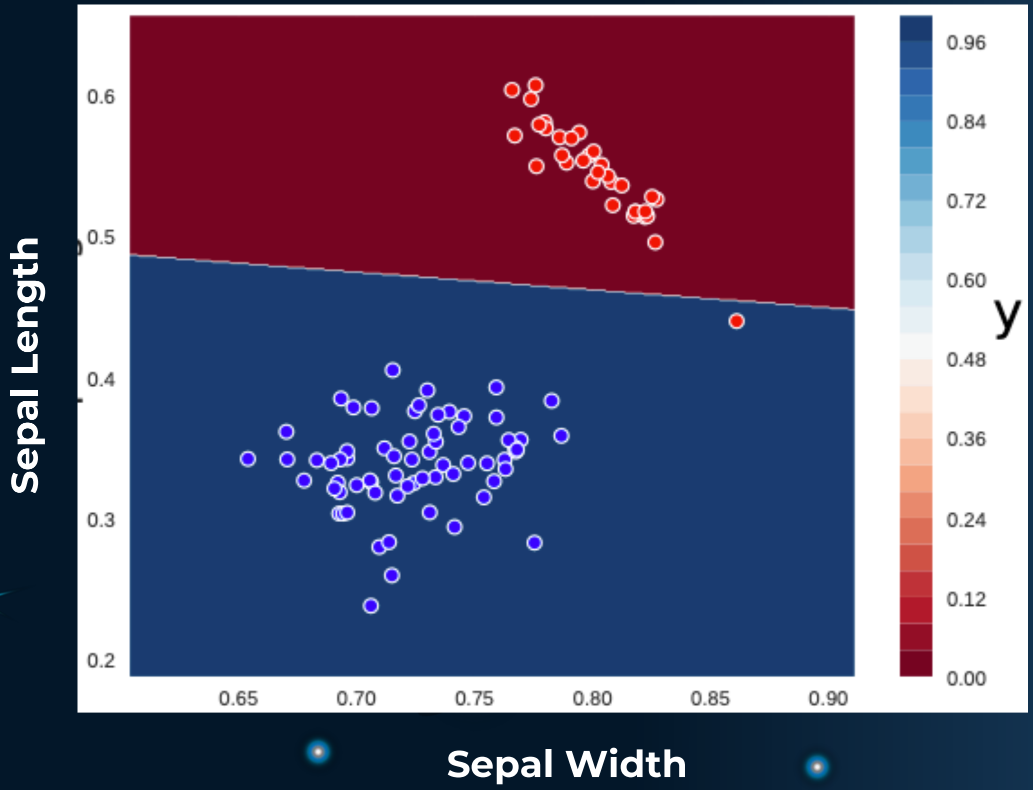
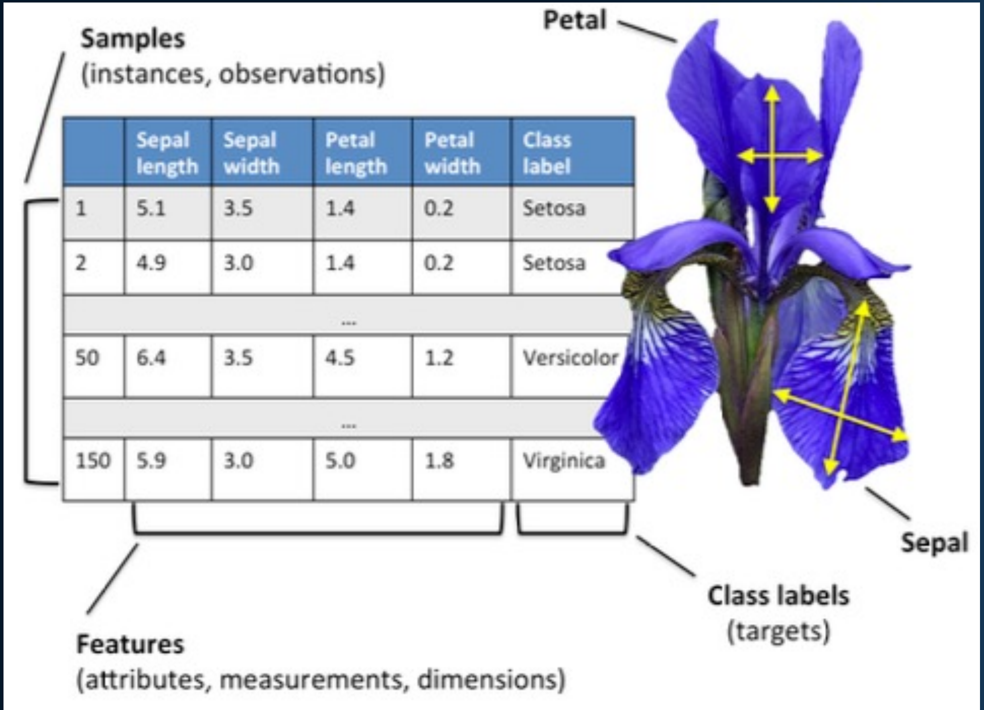


Table des matières

Classification des fleurs Iris



	Sepal length	Sepal width	Petal length	Petal width	Class label
1	5.1	3.5	1.4	0.2	Setosa
2	4.9	3.0	1.4	0.2	Setosa
...					
50	6.4	3.5	4.5	1.2	Versicolor
...					
150	5.9	3.0	5.0	1.8	Virginica

Multi-classes

Y (Catégorie)	X_1 (Sepal Length)	X_2 (Sepal Width)	X_3 (Petal Length)	X_4 (Petal Width)
0 (Setosa)	5.1	3.5	1.4	0.2
1 (versicolor)	6.4	3.5	4.5	1.2
2 (Virginica)	5.9	3.0	5.0	1.8
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

On simplifie pour utiliser uniquement deux variables (Sepal)

Y (Catégorie)	X_1 (Sepal Length)	X_2 (Sepal Width)
0 (Setosa)	5.1	3.5
1 (versicolor)	6.4	3.5
2 (Virginica)	5.9	3.0
$\vdots$	$\vdots$	$\vdots$

Transformer les classes à des labels entiers continues

Par exemple si on a des labels en chaîne de caractères

$$Y = [Setosa, versicolor, Virginica, Setosa, \dots] \Rightarrow Y = [0, 1, 2, 1, \dots]$$

Si on a des labels entiers mais qui ne sont pas continus

$$Y = [1,2,4,1, \dots] \Rightarrow Y = [0,1,2,0, \dots]$$

One-hot encoding (représentation binaire des labels)

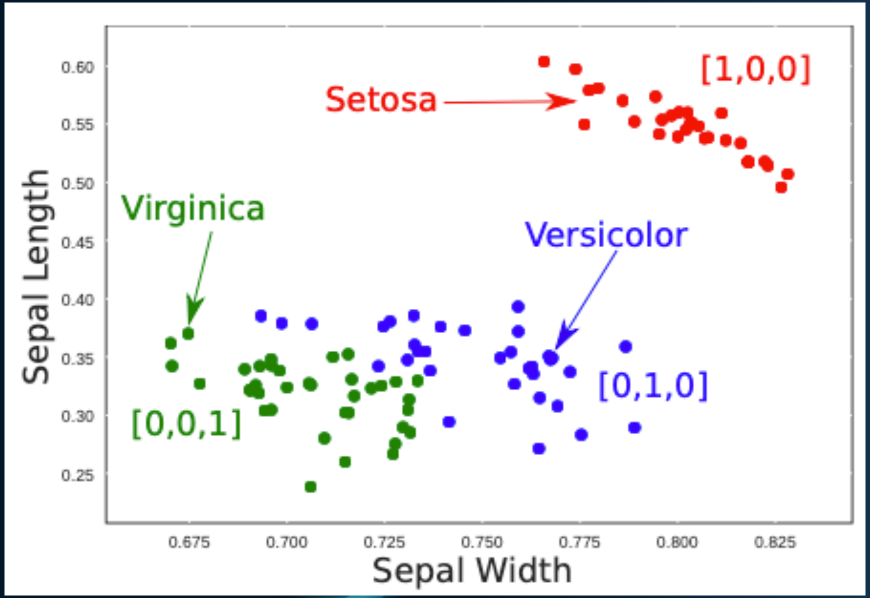
Tout d'abord, on les transforme en entiers continus

Y (Catégorie)	X_1 (Sepal Length)	X_2 (Sepal Width)
0	5.1	3.5
1	6.4	3.5
2	5.9	3.0
1	6.9	2.7

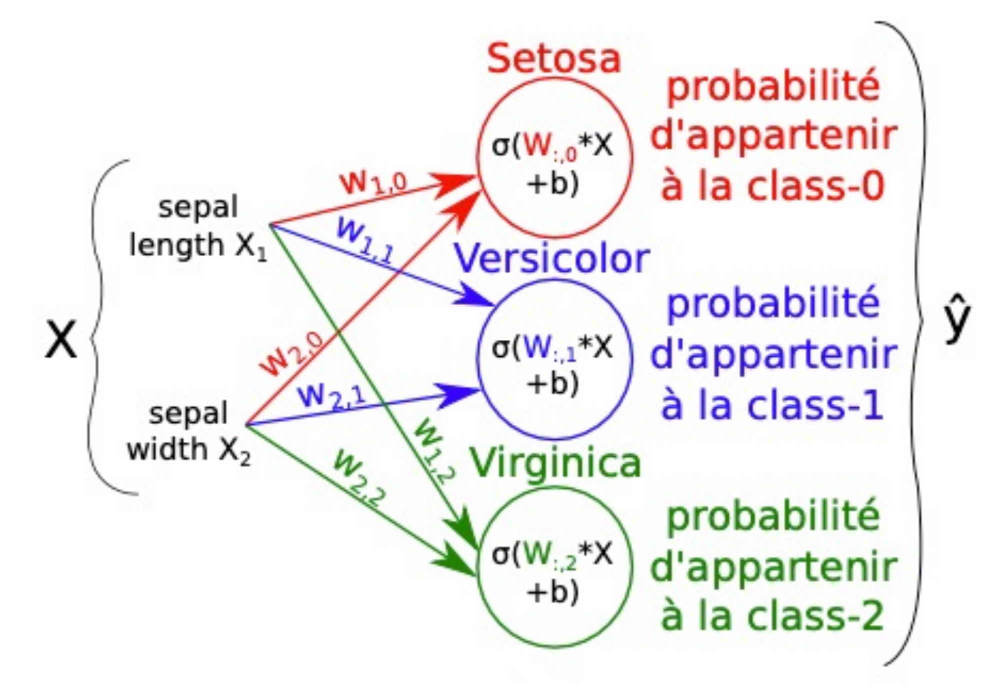
Ensuite en binaire

Y (Catégorie)	X_1 (Sepal Length)	X_2 (Sepal Width)
(1,0,0)	5.1	3.5
(0,1,0)	6.4	3.5
(0,0,1)	5.9	3.0
(0,1,0)	6.9	2.7

Visualisation des données multi-classes



C neurones pour C classes : Softmax Classifier



Probabilité d'appartenir aux trois classes

Etant données une fleur avec Sepal width & length :

$$X_i = (X_{i,1}, X_{i,2})$$

Il faut calculer le vecteur de probabilité

$$\hat{Y}_i = (\hat{y}_{i,0}, \hat{y}_{i,1}, \hat{y}_{i,2})$$

Avec

$\hat{y}_{i,0}$: la probabilité d'appartenir à la classe 0 (Setosa)

$\hat{y}_{i,1}$: la probabilité d'appartenir à la classe 1 (Versicolor)

$\hat{y}_{i,2}$: la probabilité d'appartenir à la classe 2 (Virginica)

Fonction Softmax : une généralisation de la fonction sigmoid

$\widehat{Y}_i$ est un vecteur dont les éléments représentent une distribution de probabilité sur les C classes

$$\hat{Y}_i = (\hat{y}_{i,0}, \hat{y}_{i,1}, \hat{y}_{i,2})$$

1) Chaque élément de $\widehat{Y}_i$ doit satisfaire

$$0 \leq \hat{y}_{i,j} \leq 1 \mid \forall j \in \{0, 1, 2\}$$

2) La somme des éléments de $\widehat{Y}_i$ doit satisfaire

$$\sum_{j=0}^{c} \hat{y}_{i,j} = 1$$

D'où la fonction softmax avec $z_j = W_{:,j} * X + b$

$$\hat{y}_{i,j} = \frac{e^{z_j}}{\sum_{k=0}^{C-1} e^{z_k}}$$

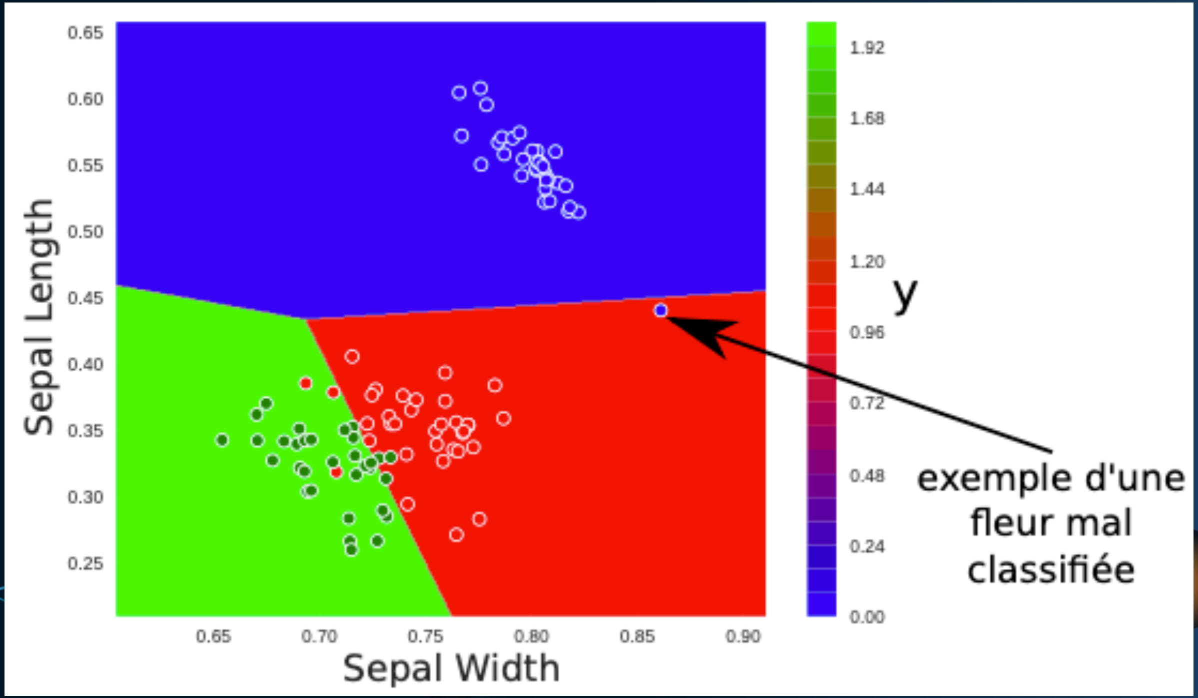
Affectation lors de la régression logistique multi-classes

On prédit la classe ayant la probabilité maximale :

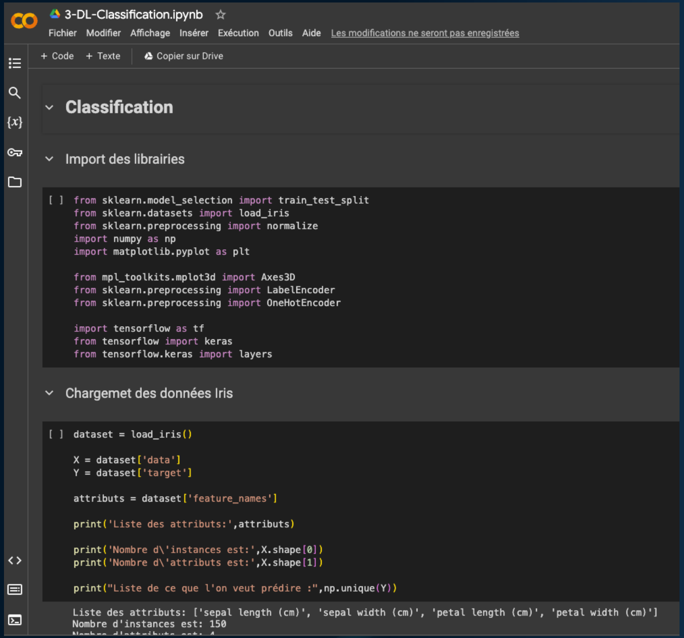
$$argmax_j \, \hat{y}_j$$

La mesure de l'accuracy ne change pas par rapport à la classification binaire

Décision dans le cas de trois classes sur Iris



De la théorie vers la pratique



Télécharger le fichier (https://github.com/r-wenger/cours_m1-m2-OTG/blob/main/IA_geosciences_M2/CM/3_DL_Classification.ipynb) et l'importer sur Google Colab