TP – METHODES NUMERIQUES ET INTELLIGENCE ARTIFICIELLE

Résoudre les problèmes en utilisant des méthodes numériques

Sujet par M. Nierenberger – Durée 2 x 2h20

Compétences associées – référentiels CPGE 2021 (PTSI-PT, PCSI-PSI, MPII-PSI, MPSI-MP) :

A - Analyser	A3.08	Analyser les principes d'intelligence artificielle.
B - Modéliser	B1.02	Identifier les grandeurs d'entrée et de sortie d'un modèle.
	B1.03	Identifier les paramètres d'un modèle.
	B3.01	Vérifier la cohérence du modèle choisi en confrontant les résultats analytiques et/ou numériques aux résultats expérimentaux.
	B3.02	Préciser les limites de validité d'un modèle.
	B3.03	Modifier les paramètres et enrichir le modèle pour minimiser l'écart entre les résultats analytiques et/ou numériques et les résultats expérimentaux.
C - Résoudre	C1.03	Choisir une démarche de résolution d'un problème d'ingénierie numérique ou d'intelligence artificielle.
	C3.01	Mener une simulation numérique.
	C3.02	Résoudre numériquement une équation ou un système d'équations.
	C3.03	Résoudre un problème en utilisant une solution d'intelligence artificielle.
D - Expérimenter	D3.03	Effectuer des traitements à partir de données.

CONTEXTE

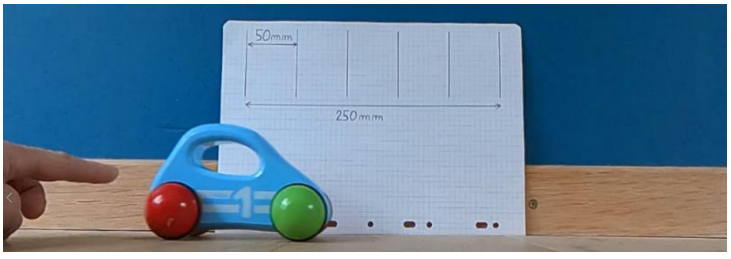
En Sciences de l'ingénieur, on est souvent confronté à l'analyse de données expérimentales issues de capteurs. On souhaite également élaborer puis valider des modélisations sur ces données.

Afin de se familiariser avec le traitement numérique de données, on propose dans ce TP d'effectuer une acquisition de données par tracking automatique sur une vidéo (méthode impliquant une intelligence artificielle – voir page suivante), puis de traiter les données extraites. On élaborera ensuite une modélisation que l'on comparera au traitement des données expérimentales.
Pour finir, on mettra en œuvre une intelligence artificielle via l'implémentation d'un réseau de neurones, afin d'aborder les notions d'apprentissage automatique sur des données expérimentales.

L'ensemble de l'analyse sera mené en mettant en œuvre des programmes python.

CAS D'ETUDE

On propose d'étudier la décélération d'une voiture miniature qui roule sur le sol dans un cadre figé, bien éclairé, avec beaucoup de contraste (conditions idéales). Elle est lâchée à partir d'une vitesse V_0 et on observe sa décélération. La vidéo a été acquise avec une GoPro Hero 9.



Caractéristiques vidéo : résolution 1920 x 1080, 240 images / s

Remarque sur l'utilisation des codes fournis : Le code utile pour ce TP est volontairement découpé en plusieurs scripts qui utilisent les variables résultant des scripts précédents. Ces variables sont sauvegardées en fin de scripts dans le répertoire de travail par une fonction $numpy$ puis ouvertes en début d'activité suivante.

Environnement de développement python conseillé : Il est conseillé d'utiliser Spyder IDE pour l'exécution / la modification des codes. Cette interface évite d'avoir à définir le chemin d'accès des fichiers et facilite le débogage en permettant la visualisation aisée des variables sauvegardées.

ACTIVITE 1 – ACQUISITION DES DONNEES PAR TRACKING

Pour l'acquisition des données, nous allons utiliser la librairie OpenCV permettant d'effectuer de nombreux traitements d'images courants en Python. Pour commencer, installons OpenCV en entrant dans la console (depuis Spyder par exemple) :

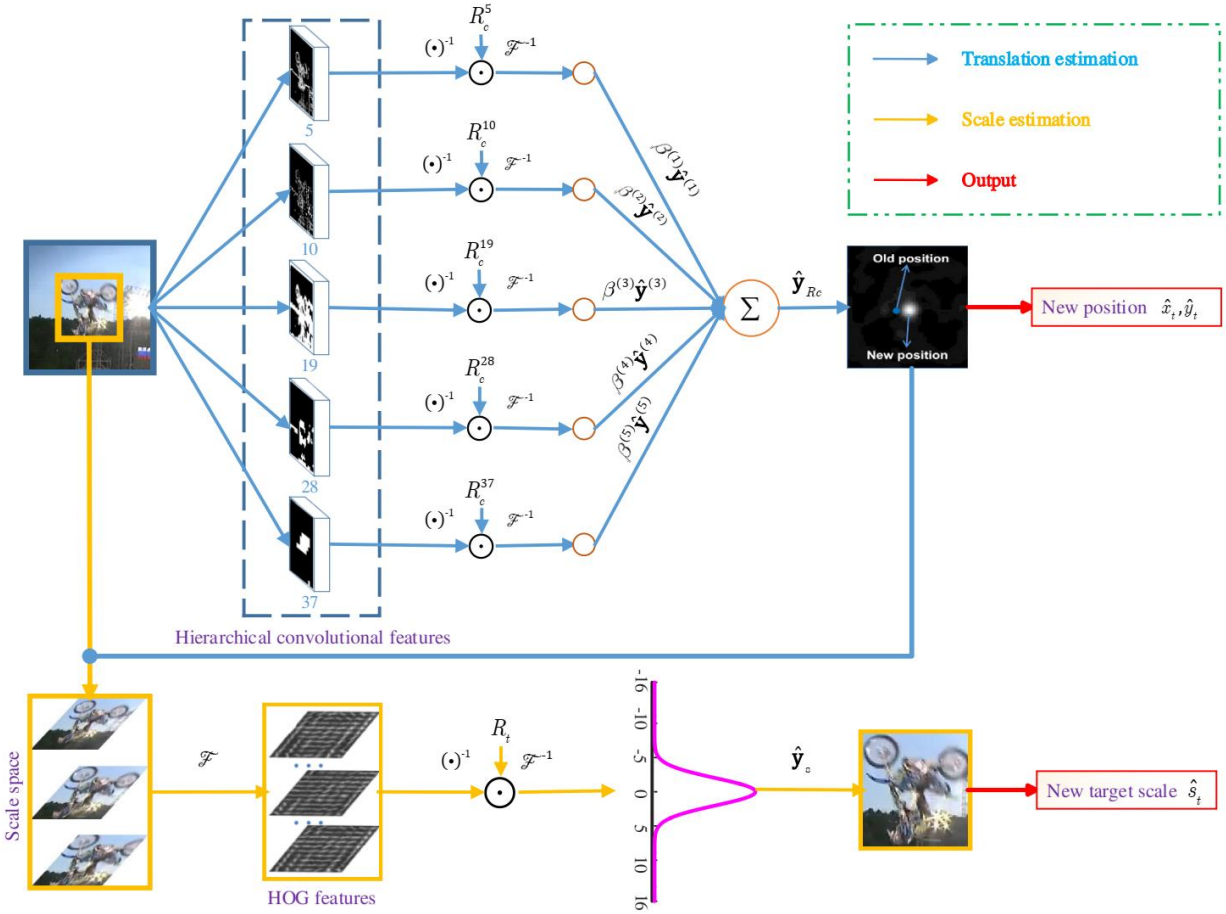
```
pip install opencv-contrib-python
```

Remarque : Si l'installation ci-dessus ne fonctionne pas, on pourra télécharger la librairie compilée (fichier .whl), la laisser dans le dossier téléchargements, puis depuis Spyder (ou autre), entrer la commande suivante en l'adaptant au nom du fichier téléchargé :

```
pip install ./downloads/opencv_contrib_python-…-win_amd64.whl
```

Nous allons utiliser les fonctions de tracking implémentées dans OpenCV, et en particulier le tracking dit KCF (Kernelized Correlation Filter). L'objet de ce TP n'est pas de chercher à comprendre comment cette technique fonctionne, mais voici schématiquement son principe de fonctionnement qui implique un réseau de neurones :

1) On identifie sur une première image une zone, dite Region Of Interest (ROI) à tracker.
2) Deux images successives sont passées dans des filtres de convolution permettant de faire ressortir des différences (Hierarchical convolutional features sur la figure ci-dessous).
3) Un réseau de neurones, visible sur la figure ci-dessous, permet de déterminer l'évolution de la position et de la taille du ROI d'une image à la suivante.



Li, Yang et al. "Robust Scale Adaptive Kernel Correlation Filter Tracker With Hierarchical Convolutional Features." IEEE Signal Processing Letters 23 (2016): 1136-1140.

En pratique, cet algorithme est déjà implémenté et devient facile à utiliser dans OpenCV. On propose de l'utiliser afin d'analyser l'évolution de la position de la voiture utilisée dans le cas d'étude 1 (vidéo $voiture\_240fps.mp4$)

- Copier l'ensemble des fichiers disponibles dans le dossier du TP dans un dossier local de l'ordinateur accessible en écriture (bureau par exemple). Ouvrir le script "$1 - Tracking.py$" dans Spyder et sélectionner le dossier contenant les vidéos comme répertoire de travail.
- Repérer dans ce script les lignes correspondant aux tâches suivantes :
  - ouverture de la vidéo
  - initialisation des variables de tracking : $xtrack, ytrack, time, currenttime$
  - sélection du ROI à tracker
  - tracking et récupération de la position du ROI tracké
  - incrémentation des variables de tracking
- Exécuter le script :
  - une fenêtre s'ouvre dans laquelle il faut sélectionner le ROI puis appuyer sur Entrée.
  - une autre fenêtre s'ouvre dans laquelle on voit l'évolution du tracking. Attendre la fin.

Remarques : - la ligne 16 sauvegarde la 1ère image dans le dossier de la vidéo.
- la variable out permet l'enregistrement d'une vidéo montrant le tracking.

Q1 : *Observer les variables de sortie $xtrack$ et $ytrack$ dans le navigateur de variables. A quelle caractéristique du ROI correspondent-elles ? Quel point sert de référence à cette mesure ? Quelles sont les unités de ces variables ?*

Q2 : *Observer l'incrémentation de la variable de sortie $time$. Quelles sont les unités de cette variable et à partir de quelle information est-elle incrémentée ?*

Q3 : *A partir de la vidéo $tracked.avi$ enregistré, commenter les performances du tracking effectué. Quelles sont d'après vous les sources d'erreur de la mesure effectuée ?*

ACTIVITE 2 – TRAITEMENT DES DONNEES DE TRACKING

On souhaite mettre en forme les données acquises pour la suite du traitement.

- Ouvrir le script "$2 - Traitement.py$".
- Ouvrir dans un éditeur d'image (Paint par ex.) le fichier $image1.jpg$ sauvegardé par le script de l'activité 1. Y effectuer une mesure puis **modifier les ligne 19, 20 et 21** du script "$2 - Traitement.py$" pour convertir les variables $xtrack$ et $ytrack$ en mètres.
- Exécuter le script. Les figures tracées sont également sauvegardées dans le répertoire de travail pour une meilleure visualisation.

Q4 : *Observer la courbe "0 – Trajectoire". L'évolution vous paraît-elle pertinente ? Quel phénomène observe-t-on sur l'axe y ? Lier la caractéristique observée à votre modification de la ligne 19 du script.*

Q5 : *Observer la courbe " 1 - Positions". L'évolution vous paraît-elle pertinente au vu du mouvement de la voiture observé sur la vidéo ? Estimez-vous que les déplacements sur y doivent être pris en compte dans la suite de l'analyse ?*

DETERMINATION DE LA VITESSE

On souhaite déterminer l'évolution de la vitesse de la voiture au cours du mouvement.

- Modifier la ligne 76 du script pour que la fonction $derivee(x, t)$ permette une dérivation numérique du vecteur x évoluant au cours du temps (vecteur t). On pourra utiliser une méthode d'Euler implicite.
- Exécuter le script.

Q6 : *Commenter et expliquer l'évolution des vitesses obtenue sur la courbe "3 – Vitesses". Pour l'explication du phénomène, vous pourrez vous appuyer sur le zoom sur la courbe de position intitulé*

"2 - Positions - zoom et filtrage". Votre réponse devra inclure les mots "résolution" et "échantillonnage".

FILTRAGE DE LA POSITION

Afin de contourner le phénomène observé, nous allons mettre en œuvre un filtrage des données de position avant la dérivation pour obtenir la vitesse.

- *Les lignes 45 à 51 du script reprennent le code d'un filtrage par moyenne glissante (voir Annexe). Modifier les lignes 54 et 55 afin de mettre en œuvre le filtrage des variables $xtrack$ et $ytrack$. En vous appuyant sur les courbes "2 - Positions - zoom et filtrage" et "3 – Vitesses", choisir une largeur adaptée de la fenêtre de filtrage.*

Q7 : *Après réglage du filtre, observer à nouveau la courbe "3 – Vitesses". Les résultats obtenus vous semblent-ils cette fois pertinents ? Noter les valeurs des temps t_d et t_f correspondant au début et à la fin de la phase de décélération.*

- Entrer les valeurs de t_d et t_f dans les lignes 98 et 99 du script puis exécuter le script afin de sauvegarder les données de vitesse associées à la décélération dans le fichier $data\_speed$.

ACTIVITE 3 : MODELISATION ET IDENTIFICATION D'UN MODELE

MODELISATION

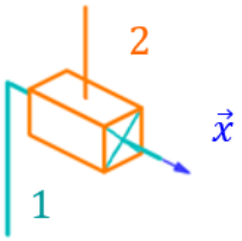
Assimilons l'ensemble en mouvement à un solide 2 de masse m en translation suivant un axe $(O, \vec{x})$ par rapport à un référentiel Galiléen lié au solide 1. Avec ν un coefficient de frottement visqueux (en N.s/m) et F_f une force de frottement sec supposée constante, l'équation en résultante du PFD appliqué au solide 2, projetée sur $\vec{x}$ donne :

$$m\frac{d^2x(t)}{dt^2} = -\nu\frac{dx(t)}{dt} - F_f$$

Soit, si l'on introduit $V(t)$ tel que $\frac{dx(t)}{dt} = \dot{x}(t) = V(t)$:

$$\dot{V}(t) + \frac{\nu}{m}V(t) = -\frac{F_f}{m}$$

On introduit également la condition initiale : $V(t=0) = V_0$



Q8 : *Ecrire puis résoudre l'équation précédente dans le cas où les frottements visqueux sont négligeables ($\nu \approx 0$). Esquisser l'allure de $V(t)$ dans ce cas.*

En vous appuyant sur l'équation différentielle ci-dessus, donner l'allure de $V(t)$ si les frottements visqueux ne sont pas négligeables.

Q9 : *En vous basant sur les résultats des questions 7 et 8, estimez-vous que les frottements visqueux doivent être pris en compte pour l'étude de la décélération de la voiture ?*

Quels termes du modèle pourront être identifiés à partir des données expérimentales de la quest. 7 ?

IDENTIFICATION DU MODELE AVEC SGDREGRESSOR

Nous allons utiliser la fonction $SGDRegressor$ (Stochastic Gradient Descent Regressor) de $Scikit-learn$ (voir Annexe). Elle nous permettra d'identifier les paramètres du modèle.

- Ouvrir le script "$3-Identification\ modele.py$".
- Repérer dans ce script les lignes correspondant aux tâches suivantes :
  - mise en forme des données
  - création et entrainement du modèle
  - utilisation du modèle pour effectuer des prédictions qui servent au tracé de la courbe modèle.

Q10 : *Exécuter le script et visualiser la courbe intitulée "4 - Vitesses décélération - régression linéaire" enregistrée automatiquement dans le répertoire de travail. L'identification par la fonction $SGDRegressor$ n'étant pas satisfaisante, expliquer pourquoi. Modifier un paramètre de la fonction (ligne 20) pour rendre l'identification satisfaisante.*

Q11 : *Relever les valeurs des coefficients identifiés par le modèle. Sachant que la voiture a une masse $m = 281\ gr$, évaluer les frottements secs F_f et la vitesse initiale V_0.*

VALIDATION EXPERIMENTALE DU PARAMETRE IDENTIFIE

Q12 : *Proposer un protocole expérimental simple permettant d'évaluer les frottements secs F_f.*

Q13 : *Ce protocole, une fois mis en œuvre sur la voiture, a permis d'évaluer $F_f \approx 0{,}2\ N$ (précision de la mesure évaluée à $\pm 0{,}03\ N$). Comparer cette valeur à celle obtenue à la question 11.*

Q14 : *Commenter la modélisation. Voyez-vous des éléments qui ont été négligés ?*

IDENTIFICATION D'UN MODELE DU TYPE "RESEAU DE NEURONES"

Nous allons dans cette partie mettre en œuvre un réseau de neurones pour retrouver l'identification précédente. On utilise pour cela la fonction $MLPRegressor$ (Multi-layer Perceptron Regressor) de $Scikit-learn$. La base du fonctionnement des réseaux de neurones est rappelée en Annexe.

Remarque : Un réseau de neurones est plutôt destiné à traiter / modéliser des situations complexes, comme celle traitée en partie 1 (tracking sur une image). On l'utilise ici dans une situation simpliste afin de pouvoir comprendre manuellement le fonctionnement du réseau.

- Ouvrir le script "$3 - Reseau\ de\ neurones.py$".
- Repérer dans ce script les lignes correspondant aux tâches suivantes :
  - séparation du dataset en un dataset d'entrainement et un dataset de test (33% conservés pour test)
  - création et entrainement du réseau de neurones. Parmi ses propriétés, on note principalement :
    - le nombre de couches intermédiaires (cachées) : ici 1 couche de 2 neurones
    - la fonction d'activation des perceptrons : ici la fonction sigmoïde ($logistic$) $\phi_s(z) = \frac{1}{1+e^{-z}}$
  - le test du modèle sur le dataset de test
- Exécuter le script. Observer sur la courbe " 5 - Réseau de neurones" le dataset d'entrainement, celui de test et les prédictions effectuées par le modèle. On notera que les prédictions sont ici très proches de celles faites avec le modèle linéaire identifié précédemment.

Q15 : *Représenter le réseau de neurones créé par le script (agencement de plusieurs perceptrons). En utilisant les attributs $coefs\_$ et $intercepts\_$ de la fonction $MLPRegressor$, extraire les poids et les biais des différents perceptrons du réseau et les inscrire sur votre figure. Pour cela, on commencera par écrire dans la console : $regr.coefs\_$ puis $regr.intercepts\_$*

On se basera ensuite sur l'extrait de documentation ci-dessous :

coefs_ **list of shape (n_layers - 1,) :**

The i[th] element in the list represents the weight matrix corresponding to layer i.

intercepts_ **list of shape (n_layers - 1,) :**

The i[th] element in the list represents the bias vector corresponding to layer i + 1.

Q16 : *Vérifier le bon fonctionnement du réseau de neurones précédemment représenté en calculant manuellement la réponse du réseau à un échantillon du dataset de test. Par exemple $t \approx 0{,}9s$ ➔ $V(t)_{predit}$? On rappelle que la fonction d'activation n'est pas appliquée dans la couche de sortie pour un problème de régression et on pourra utiliser les rappels sur le fonctionnement des neurones artificiels donné en Annexe.*

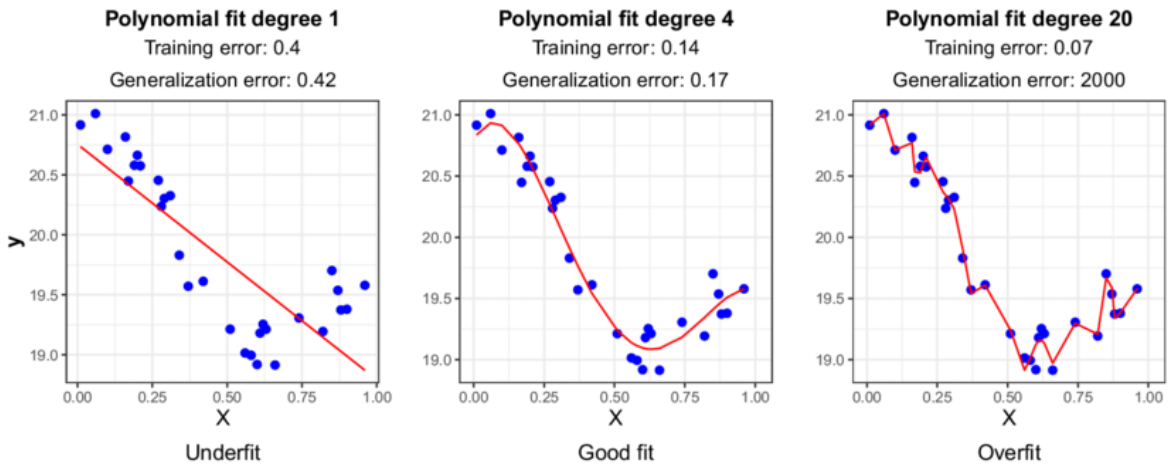
Comparer le résultat obtenu à la valeur associée relevée dans la variable vxd_test.

Nous allons maintenant faire varier des paramètres propres au réseau de neurones :

- Essayer d'augmenter le nombre de couches et de neurones. On pourra par exemple essayer 2 couches de 4 neurones cachés en écrivant $hidden\_layer\_sizes = (4,4)$.
  ➔ On obtient un résultat très légèrement meilleur qu'avec 2 neurones cachés.
- En choisissant à nouveau une couche de 2 neurones cachés, essayer de modifier la tolérance du solveur, à la hausse ($1e-7$ par ex.) ou à la baisse ($1e-12$ par ex.).
  ➔ La modification de la tolérance a une influence très importante sur l'identification du modèle.

Et le surapprentissage ?

On parle de surapprentissage du réseau lorsqu'on observe une amélioration des performances sur la base de données d'entraînement en même temps qu'une baisse / une stagnation des performances sur celle de test (validation). Concrètement, cela veut dire que le réseau est en train de "trop se spécialiser" sur la base de données d'entrainement comme le montre la figure ci-dessous.



Solveig Badillo et al. "An Introduction to Machine Learning" March 2020, Clinical Pharmacology & Therapeutics 107(4)

On ne parvient pas à mettre clairement en évidence le surapprentissage sur les données de ce TP de par leur linéarité. On constate plus facilement ce surapprentissage en cas d'un faible nombre de données comportant des non-linéarités importantes. Il est obtenu en réduisant excessivement la tolérance et le nombre d'itérations maximal du modèle.
On cherche bien sûr à éviter le surapprentissage.

Pour illustrer les éléments précédents relatifs au choix du nombre de neurones et de couches cachées, on peut effectuer différents essais en utilisant une version simplifiée du playground tensorflow disponible au lien suivant. On peut y choisir le nombre de neurones et de couches cachées, la fonction d'activation, et le type de données (plus ou moins complexes) sur lesquelles on souhaite travailler.

Conclusion de cette analyse :

Lors d'une modélisation par un réseau de neurones, il faudra faire attention, lors de l'apprentissage, à utiliser un nombre de couches de neurones et de neurones par couches adaptés. De plus, le choix du solveur servant à l'apprentissage et de ses paramètres (tolérance, nombre maximal d'itérations) doit être adapté au problème.

POUR ALLER PLUS LOIN :

A vous de jouer : Effectuer une prise de vue d'un mouvement de décélération / d'accélération. Attention à bien maintenir la caméra fixe lors de l'acquisition. Utiliser la démarche précédente pour caractériser l'évolution des paramètres du mouvement au cours du temps.

Remarque : Un exemple faisant apparaître un phénomène de frottements visqueux serait intéressant à étudier.

ANNEXE – FICHE METHODE

Cette fiche reprend succinctement les prérequis vus en cours et nécessaires à ce TP.

DERIVEE NUMERIQUE – SCHEMA D'EULER

Rappelons la définition mathématique d'une dérivée (taux d'accroissement) : $f'(x) = \frac{df(x)}{dx} = \frac{f(x+dx)-f(x)}{dx}$

En appliquant cette définition entre deux points discrétisés, on peut exprimer la dérivée de deux manières :

Dérivée en x_i dépendant des valeurs au point i et au point **précédent** - **Schéma d'Euler implicite** (utilisé ici) :

$$f'(x_i) = \frac{f(x_i) - f(x_{i-1})}{x_i - x_{i-1}}$$

Dérivée en x_i dépendant des valeurs au point i et au point **suivant** - **Schéma d'Euler explicite** :

$$f'(x_i) = \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i}$$

FILTRAGE PAR MOYENNE GLISSANTE

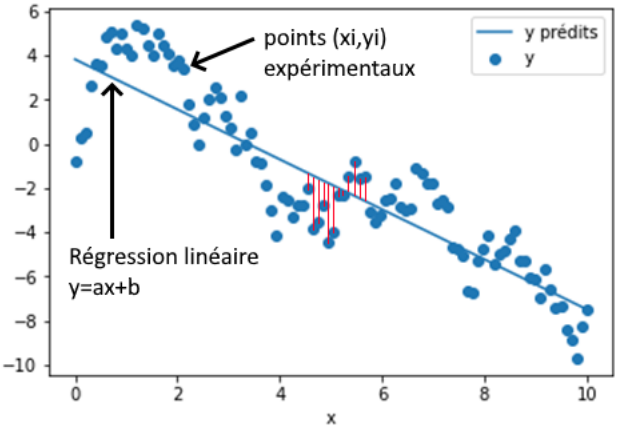
Un filtre par moyenne glissante consiste à remplacer une valeur de mesure par la moyenne des n valeurs entourant cette mesure. Soit un vecteur U ci-dessous, qui pourrait représenter l'évolution temporelle d'une grandeur physique. On souhaite filtrer le vecteur U ci-dessous avec un filtre de type moyenne glissante sur un intervalle de $n = 3$ valeurs (supposons n toujours impair).

$$U = \begin{pmatrix} 1 \\ 3 \\ 2 \\ 4 \\ 3 \\ 2 \\ 1 \\ 6 \\ 8 \end{pmatrix} \rightarrow U_{moyenne\ glissante,n=3} = \begin{pmatrix} ? \\ 2 \\ 3 \\ 3 \\ 3 \\ 2 \\ 3 \\ 5 \\ ? \end{pmatrix}$$

Chaque valeur est remplacée par la moyenne des n (3 ici) valeurs l'entourant. Les valeurs aux extrémités ne sont pas définies avec cette méthode (? Sur l'illustration ci-dessus). Habituellement, on peut pour ces valeurs conserver les valeurs du vecteur U initial.

REGRESSION LINEAIRE – METHODE DES MOINDRES CARRES

Soient x_i et y_i respectivement des abscisses et ordonnées discrètes, issues de mesures par exemple, auxquelles on souhaite associer un modèle linéaire du type $y = f(x) = ax + b$. **Trouver la meilleure fonction f revient à trouver le couple (a, b) (pente, ordonnée à l'origine) qui minimise les écarts entre la courbe modèle et les points discrets. Pour cela, on minimise la fonction coût J définie telle que :** $J(a,b) = \sum_i (y_i - (ax_i + b))^2$



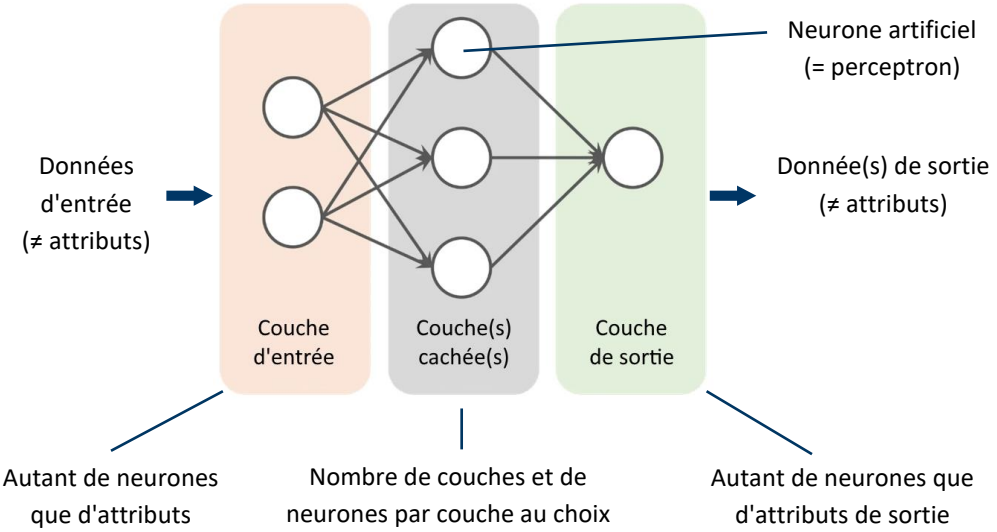
Concrètement, cela revient à trouver (a, b) permettant de minimiser la somme des écarts $y_i - (ax_i + b)$ (en rouge sur le schéma ci-dessus) au carré.

La fonction SGDRegressor de la librairie Scikit Learn permet d'effectuer cette optimisation et de déterminer les coefficients (a, b) optimaux (vis-à-vis d'un critère de tolérance imposé).

RESEAU DE NEURONES

STRUCTURE

Un réseau de neurones artificiels est un ensemble de perceptrons (voir ci-dessous) organisés en couches. Chaque neurone (perceptron) est lié à tous les neurones de la couche précédente et à tous ceux de la couche suivante.



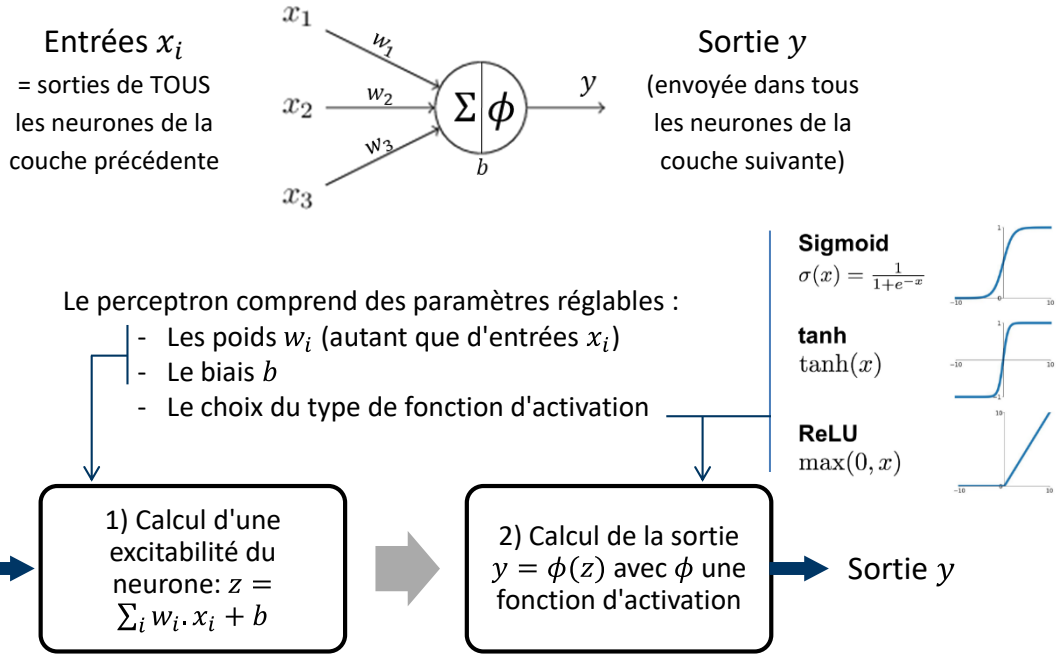
Le but du réseau de neurones est d'associer des attributs de sortie aux attributs d'entrée. Par attribut, on entend une caractéristique (classe ou valeur) associée à un échantillon.

L'ensemble des échantillons disponible (dataset) est généralement séparé en 2 :

- Un dataset d'entrainement (environ 70% des échantillons), servant à la détermination des coefficients propres aux perceptrons.
- Un dataset de test (environ 30% des échantillons) servant à l'évaluation des performances du modèle.

PERCEPTRON = NEURONE ARTIFICIEL

Le fonctionnement du modèle du perceptron (Frank Rosenblatt, 1957) est présenté ci-dessous :



Entrainer un réseau de neurones revient à déterminer tous les poids w_i et biais b de chacun des neurones du réseau pour que les attributs d'entrée du réseau engendrent ceux de sortie pour tous les échantillons du dataset d'entrainement. La détermination de ces coefficients repose sur la minimisation d'une fonction coût.

Nom :

TP METHODES NUMERIQUES ET IA - DOCUMENT REPONSE

ACTIVITE 1 – ACQUISITION DES DONNEES PAR TRACKING

Q1 : *A quelle caractéristique du ROI correspondent $xtrack$ et $ytrack$?*

Quel point sert de référence à cette mesure ?

Quelles sont les unités de ces variables ?

Q2 : *Quelles sont les unités de la variable $time$?*

A partir de quelle information est-elle incrémentée ?

Q3 : *Commenter les performances du tracking effectué :*

Sources d'erreur de la mesure effectuée ?

ACTIVITE 2 – TRAITEMENT DES DONNEES DE TRACKING

Q4 : *Courbe "0 – Trajectoire" - évolution pertinente ?*

Quel phénomène observe-t-on sur l'axe y ?

Lien avec la caractéristique de la ligne 19 du script :

Q5 : *Courbe " 1 - Positions" - évolution pertinente au vu du mouvement de la voiture ?*

Estimez-vous que les déplacements sur y doivent être pris en compte dans la suite de l'analyse ?

Q6 : *Commenter <u>et expliquer</u> l'évolution des vitesses obtenue sur la courbe "3 – Vitesses". Inclure les mots "résolution" et "échantillonnage" !*

Q7 : *Caractéristique retenue pour le filtre :*

Courbe "3 – Vitesses" - résultats pertinents ?

Début et fin de la décélération :

$t_d =$ $t_f =$

ACTIVITE 3 : MODELISATION ET IDENTIFICATION D'UN MODELE

MODELISATION

Q8 : *Ecrire puis résoudre l'équation précédente dans le cas où* $v \approx 0$.

Allure de $V(t)$ *dans ce cas :*

Allure de $V(t)$ *si les frottements visqueux ne sont pas négligeables :*

Q9 : *Les frottements visqueux doivent-ils être pris en compte pour l'étude de la décélération de la voiture ?*

Quels termes du modèle pourront être identifiés à partir des données expérimentales de la question 7 ?

IDENTIFICATION DU MODELE AVEC SGDREGRESSOR

Q10 : *Pourquoi l'identification avec $SGDRegressor$ n'est-elle pas satisfaisante ?*

Modification proposée :

Q11 : *Valeurs des coefficients identifiés par le modèle :*

Evaluation des frottements secs F_f et la vitesse initiale V_0 :

VALIDATION EXPERIMENTALE DU PARAMETRE IDENTIFIE

Q12 : *Protocole expérimental simple permettant d'évaluer les frottements secs F_f :*

Q13 : Expérimentalement, $F_f \approx 0{,}2\ N$ *(précision de la mesure évaluée à $\pm 0{,}03\ N$). Comparer cette valeur à celle obtenue à la question 11 :*

Q14 : *Commenter la modélisation. Voyez-vous des éléments qui ont été négligés ?*

IDENTIFICATION D'UN MODELE DU TYPE "RESEAU DE NEURONES"

Q15 : *Représenter le réseau de neurones créé par le script (agencement de plusieurs perceptrons) et y annoter les différents poids et biais extraits du modèle :*

Q16 : *Calculant manuellement la réponse du réseau à un échantillon du dataset de test :*

Valeur retenue : $t =$ ➔ $V(t)_{réelle} =$

Valeur prédite : ➔ $V(t)_{predite} =$