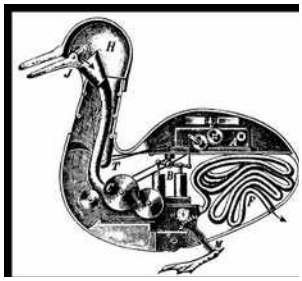
Intelligence Artificielle

Sommaire

Christophe LIGERET et Benjamin MOUSSET

I/ Culture générale ; recherche bibliographique

Activités à proposer avec les élèves :
- des exposés des élèves.
- une activité recherche sur Internet avec compte-rendu à remettreen fin de séance
- discussions / débats avec la classe : par exemple: intelligence humaine vs. Intelligence artificielle ; conscience artificielle ; IA et diagnostique médical ...



Le canard artificiel de Vaucanson (1738)

https://www.larousse.fr/encyclopedie/divers/intelligence_artificielle/187257

https://fr.wikipedia.org/wiki/Test_de_Turing

https://fr.wikipedia.org/wiki/Intelligence_artificielle

*L'**intelligence artificielle** (**IA**) est « l'ensemble des théories et des techniques mises en œuvre en vue de réaliser des machines capables de simuler l'intelligence »*

elle fait appel à la neurobiologie computationnelle (particulièrement aux réseaux neuronaux), à la logique mathématique (sous-discipline des mathématiques et de la philosophie) et à l'informatique. Elle recherche des méthodes de résolution de problèmes à forte complexité logique ou algorithmique. Par extension elle désigne, dans le langage courant, les dispositifs imitant ou remplaçant l'homme dans certaines mises en œuvre de ses fonctions cognitives.

Un des fondateurs : Alan Turing ; il lance le concept d'intelligence artificielle en 1950 : article célèbre : "jeu de l'imitation" : faculté de la machine à imiter le comportement humain.
Test : mettre un humain en face d'une machine ou d'un autre humain : conversation : si impossibilité de dire si humain / pas humain : test réussi.

Machine learning : les machines peuvent apprendre : plus seulement une liste de règles "pré apprises"
Notion d'adaptation à l'environnement de manière dynamique.

Dans les années 1980, l'apprentissage automatique (« Machine Learning ») se développe. L'ordinateur commence à déduire des « règles à suivre » rien qu'en analysant des données

Parallèlement des algorithmes « apprenants » sont créés, préfigurant les futurs réseau de neurones, l'apprentissage par renforcement, les machines à vecteurs de support, etc.). Ceci permet par exemple en mai 1997 à l'ordinateur Deep Blue de battre Garry Kasparov aux échecs

Dans les années 2000 le Web2.0, le big data et de nouvelles puissances et infrastructures de calcul, permettent

à certains ordinateurs d'explorer des masses de données sans précédent ; c'est l'apprentissage profond (« deep learning »). Dans certains domaines, les meilleurs experts humains sont dépassés par des programmes informatiques (détection, reconnaissance visuelle, analyse documentaire, traduction, jeu d'échecs en 1997, le jeu de go en 2016 et le poker en 2017.

Depuis les années 1980 la recherche se fait principalement aux États-Unis, notamment à l'université Stanford sous l'impulsion de John McCarthy, au MIT sous celle de Marvin Minsky, à l'université Carnegie-Mellon sous celle de Allen Newell et Herbert Simon et à l'université d'Édimbourg sous celle de Donald Michie, en Europe et en Chine. En France, l'un des pionniers est Jacques Pitrat.

Investissements : ils auraient été décuplés entre 2010 et 2017, dépassant de 5 Mds d'euros en 2017

Personnalités :
Plusieurs prix Turing ont été attribués à des pionniers de l'intelligence artificielle, notamment :

- Marvin Minsky (1969)
- John McCarthy (1971)
- Allen Newell et Herbert Simon (1975)
- Edward Feigenbaum et Raj Reddy (1994)
- Judea Pearl (2011)
- Yann LeCun, Geoffrey Hinton et Yoshua Bengio (2019)

En 2004, l'Institut Singularity a lancé une campagne Internet appelée « Trois lois dangereuses »

En 2005, le projet Blue Brain est lancé, il vise à simuler le cerveau des mammifères. Il s'agit d'une des méthodes envisagées pour réaliser une IA. Ils annoncent de plus comme objectif de fabriquer, dans dix ans, le premier « vrai » cerveau électronique

Le 27 janvier 2010, l'US Air Force demande l'aide de l'industrie pour développer une intelligence avancée de collecte d'information et avec la capacité de décision rapide pour aider les forces américaines pour attaquer ses ennemis rapidement à leurs points les plus vulnérables.

Entre 2014 et 2015, à la suite du développement rapide du deep learning, et à l'encontre des penseurs transhumanistes, quelques scientifiques et membres de la communauté high tech craignent que l'intelligence artificielle ne vienne à terme dépasser les performances de l'intelligence humaine. Parmi eux, l'astrophysicien britannique Stephen Hawking28, le fondateur de Microsoft Bill Gates29 et le PDG de Tesla Elon Musk

En janvier 2018, des modèles d'intelligence artificielle développés par Microsoft et Alibaba réussissent chacun de leur côté à battre les humains dans un test de lecture et de compréhension de l'Université de Stanford. Le traitement du langage naturel imite la compréhension humaine des mots et des phrases et permet maintenant aux modèles d'apprentissage automatique de traiter de grandes quantités d'informations avant de fournir des réponses précises aux questions qui leur sont posées

En février 2019, l'institut de recherche OpenAI annonce avoir créé un programme d'intelligence artificielle capable de générer des textes tellement réalistes que cette technologie pourrait être dangereuse36,37. Si le logiciel est utilisé avec une intention malveillante, il peut générer facilement des fausses nouvelles très crédibles. Inquiet par l'utilisation qui pourrait en être faite, OpenAI préfère ne pas rendre public le code source du programme

Le concept d'intelligence artificielle forte fait référence à une machine capable non seulement de produire un comportement intelligent, mais d'éprouver une impression d'une réelle conscience de soi, de « vrais sentiments » (quoi qu'on puisse mettre derrière ces mots), et « une compréhension de ses propres raisonnements »58.

L’intelligence artificielle forte a servi de moteur à la discipline, mais a également suscité de nombreux débats. En se fondant sur l'hypothèse, que tendent à confirmer les neurosciences et que des chercheurs n'hésitent pas à affirmer59, que la conscience a un support biologique et donc matériel, les scientifiques ne voient généralement pas d’obstacle de principe à créer un jour une intelligence consciente sur un support matériel autre que biologique.

Le cerveau humain, formé de 10^{11} neurones ne pouvant chacun commuter plus de 100 fois par seconde en raison de leur temps de relaxation permettait beaucoup plus de traitements logiques par unité de temps (10^{13} opérations logiques par seconde)

La notion d’intelligence artificielle faible constitue une approche pragmatique d’ingénieur : chercher à construire des systèmes de plus en plus autonomes (pour réduire le coût de leur supervision), des algorithmes capables de résoudre des problèmes d’une certaine classe, etc. Mais, cette fois, la machine simule l'intelligence, elle semble agir comme si elle était intelligente. On en voit des exemples concrets avec les programmes conversationnels qui tentent de passer le test de Turing, comme ELIZA. Ces logiciels parviennent à imiter de façon grossière le comportement d'humains face à d'autres humains lors d'un dialogue.

Apprentissage automatique / profond

Apprentissage supervisé / non supervisé

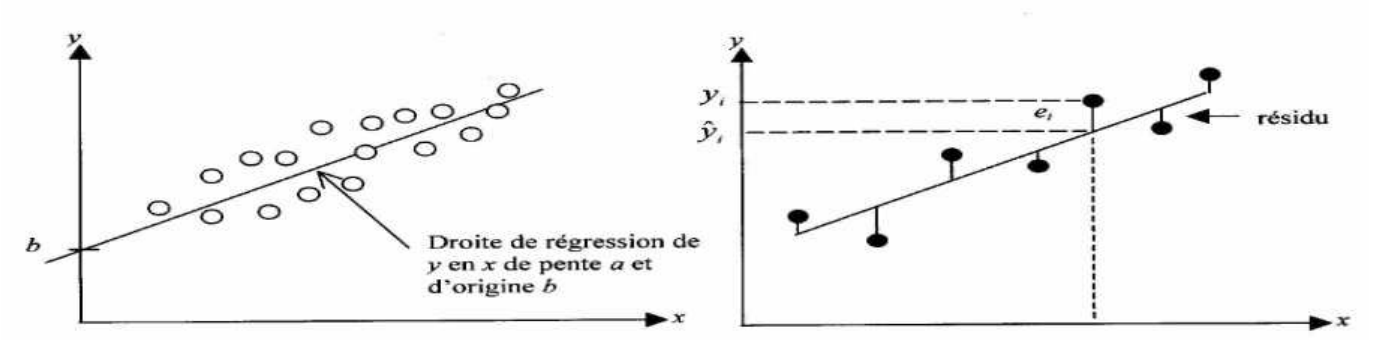
Cas les plus connus d'IA :
- Jeux d'échecs contre ordinateur
- Google Car
- Reconnaissance faciale, d'images, de voix
- Finance
- Militaire
- Medecine
- Justice / Police
- Transports
- Robotique
- Art

Portrait d'Edmond de Belamy (432 500 $)

II/ Plans d'expériences

Partie cours / exemples : donner la formule mathématique avec explications lorsqu'on est en présence d'un ensemble de points $(x_i ; y_i)$ avec $\hat{y}_i = a \times x_i + b$ où $\hat{y}_i$ est une estimation de y_i avec un modèle basé sur une fonction affine de coefficient directeur a et d'ordonnée à l'origine b .
Cet ajustement affine permet de faire automatiquement des prédictions : on cherche à minimiser un critère quadratique : E comme suit :

$$E = \sum_{i=0}^{n} \varepsilon_i^2 = \sum_{i=0}^{n} (y_i - (ax_i + b))^2.$$



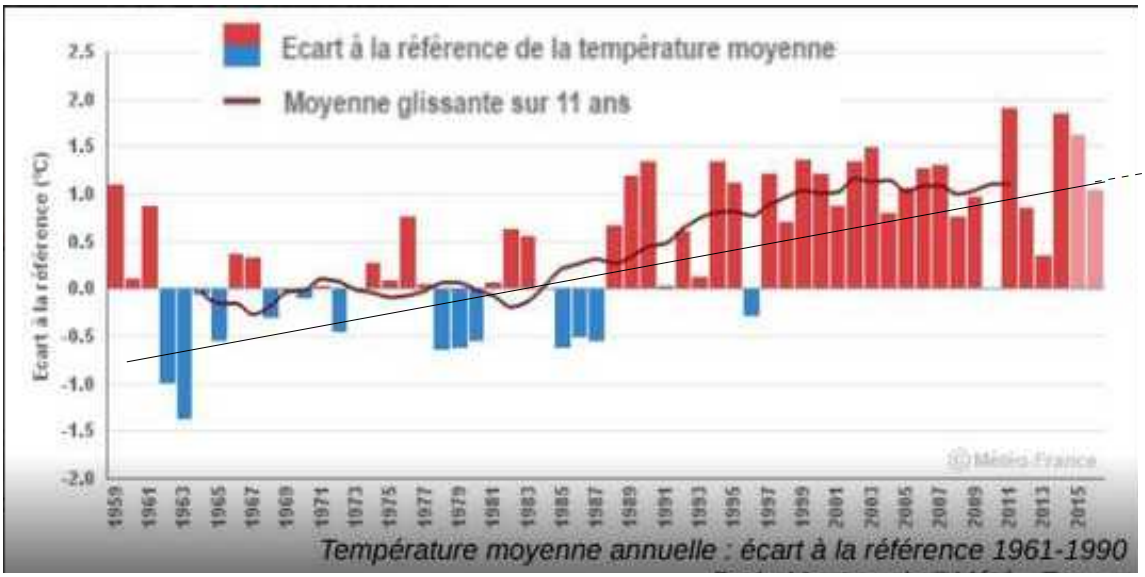
Un calcul montre que ces valeurs, notées $\hat{a}$ et $\hat{b}$, sont égales à $\hat{a} = \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\sum_{i=1}^{n}(x_i - \overline{x})^2}$ et $\hat{b} = \overline{y} - \hat{a}\overline{x}$. On exprime souvent $\hat{a}$ au moyen de la *variance* de X, s_x^2, et de la *covariance* des variables ou x , respectivement y est la moyenne des x_i , respectivement la moyenne des y_i .

Activité 1 : faire estimer par les élèves les estimations de a et de b dans un cas avec peu de points (moins de 10).

x	0	1	2	3	4	5	6	7	8	9
y	0	2	4	6	8	10	12	14	16	18

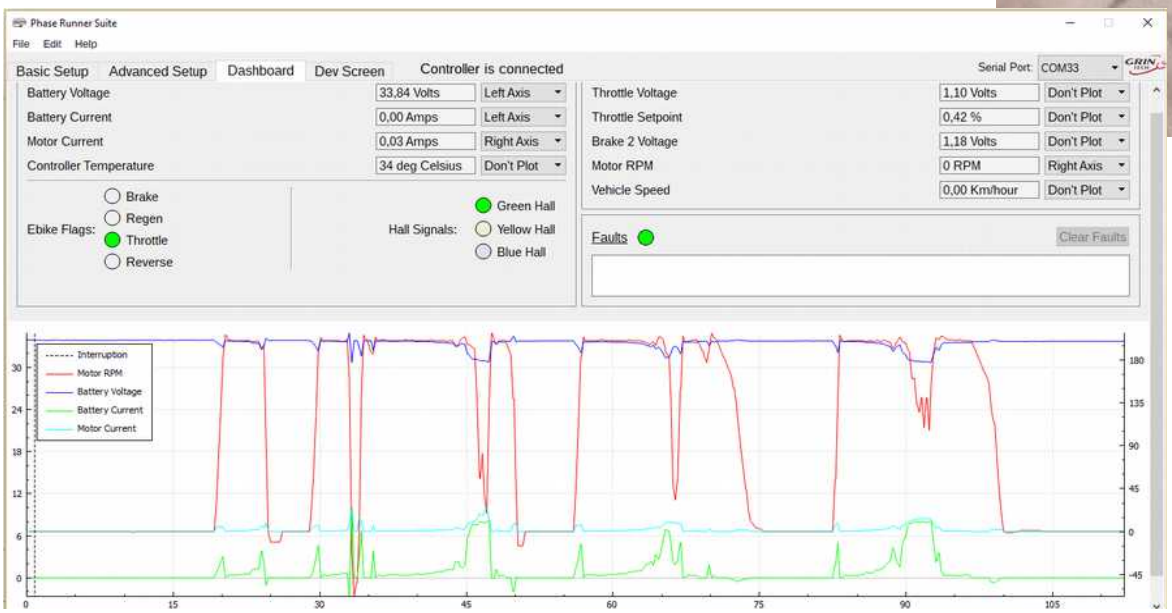
x	0	1	2	3	4	5	6	7	8	9
y	-0.1	2.2	4.1	5.8	8.2	10.5	11.3	14.2	16.1	17.8

Activité 2 : prédiction relatives au réchauffement climatique



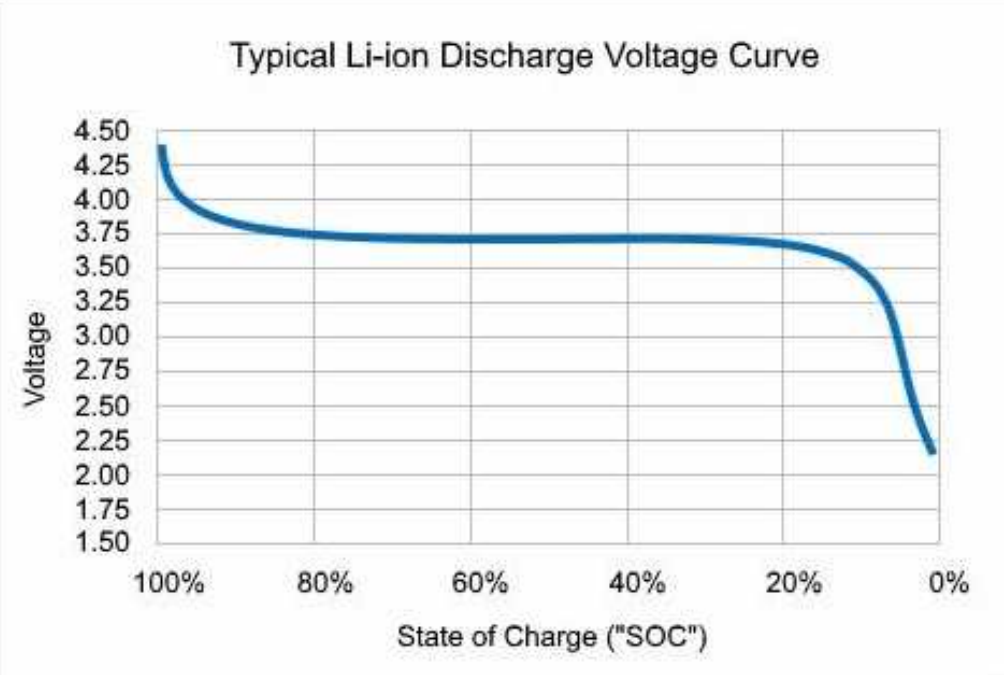
On peut ainsi prédire un réchauffement climatique de +5°C en 2100 (climat des Dinosaures) !!!

Activité 3 / idée de projets (pour les bricoleurs !) : calcul de l'état de charge (SOC state of charge) d'une batterie lithium-ion (ordinateur portable ; appareils domestiques sur batterie ; véhicule électrique...)
Objectif : donner l'état de charge (SOC) ainsi que l'état de santé (SOH) d'une batterie électrique (calcul d'autonomie, de vieillissement...)
On dispose d'une batterie dont les données sont déjà enregistrées et qui est dans différents états de charge (SOC) : 25% ; 50% ; 75% et 100%



On mesure un ensemble de points $(U\,;I)$ et on en déduit U_0 : la force électromotrice (tension à vide) et la la résistance interne r selon le modèle affine : $U=U_0-r\times I$

A partir de U_0 , on en déduit le niveau de charge (SOC) en pourcentage de la batterie

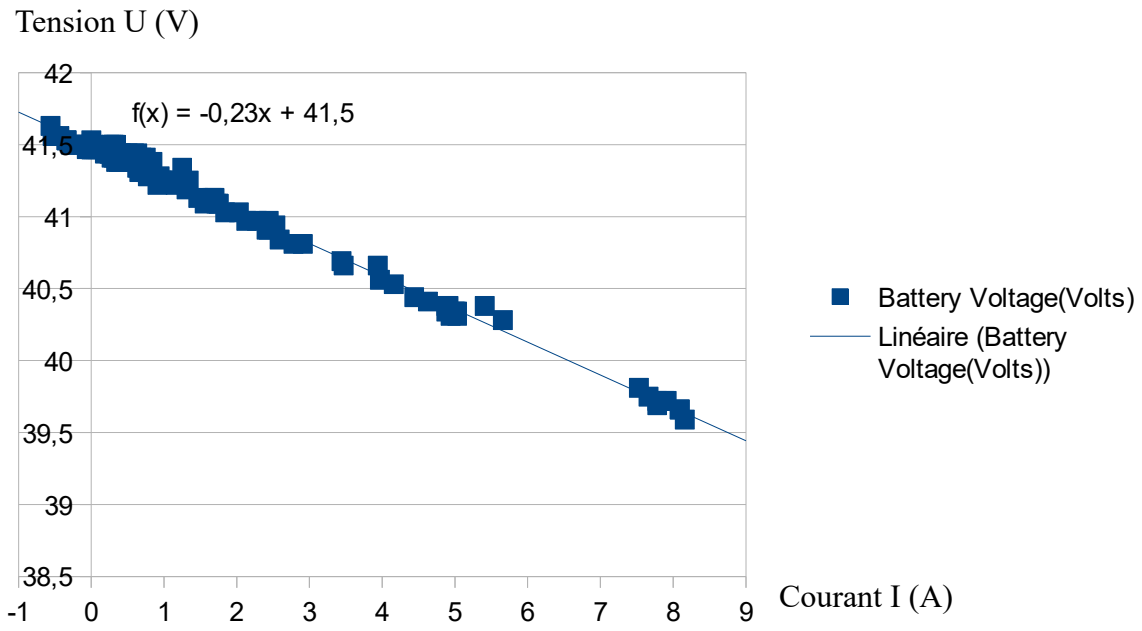


On pourra prendre pour l'estimation de l'état de charge (SOC) le polynôme de degré 4 ci-dessous :

$$\hat{SOC}=8665.3622-8168.4218\,.\,U_0+2678.5024\,.\,U_0^2-349.65261\,.\,U_0^3+14.132186\,.\,U_0^4$$

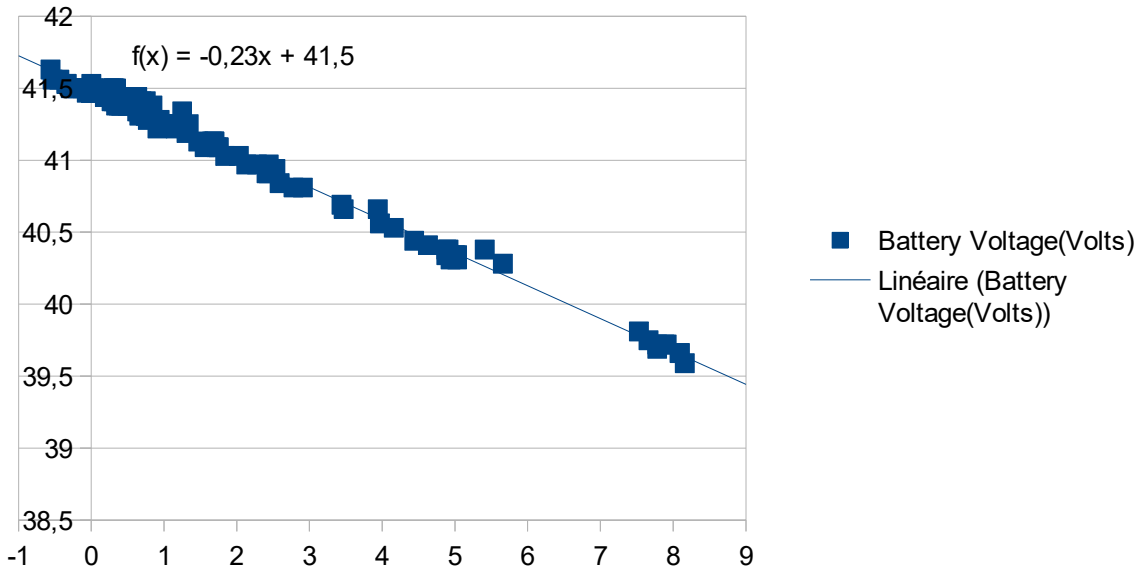
Dans cette partie : modèles électriques avec un état de charge (SOC) à 100% ; 75% ; 50% et 25%

Représentation plan (I ; U)



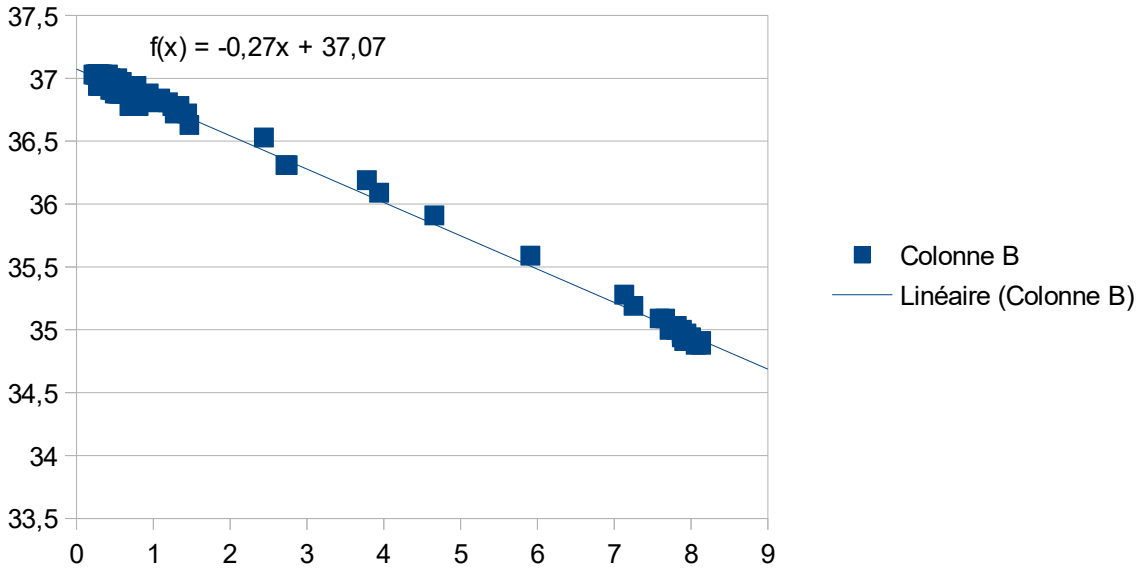
Ci-dessus : U = 41,5 – 0,23 I ; U_0 = 41,5 V ; r = 0,23 Ohms

Modèle électrique à SOC = 100%



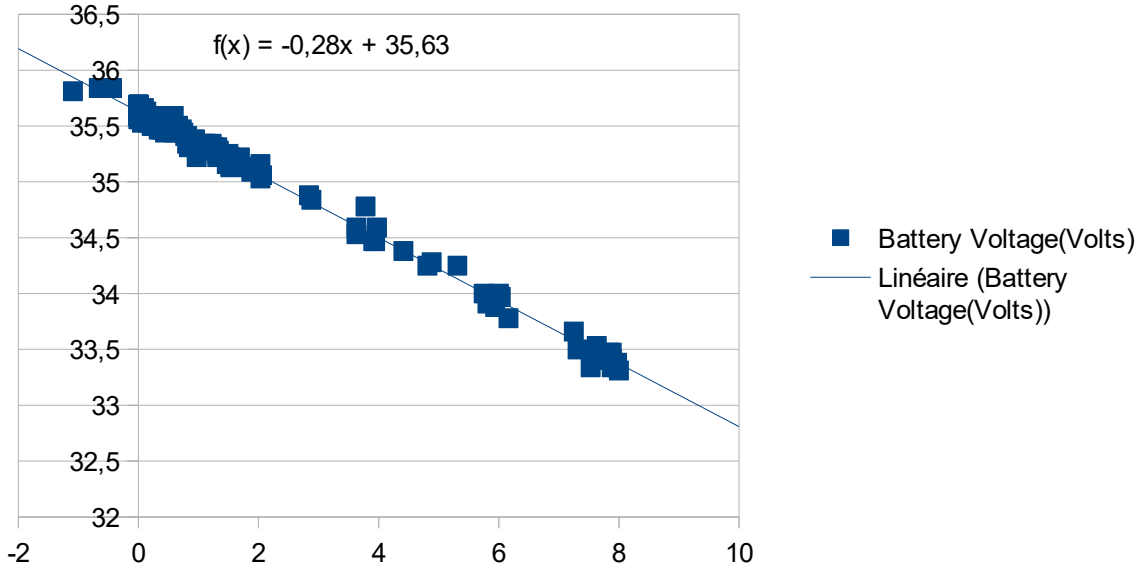
Ci-dessus : U = 41,5 – 0,23 I ; U_0 = 41,5 V ; r = 0,23 Ohms

Modèle électrique à SOC = 75%



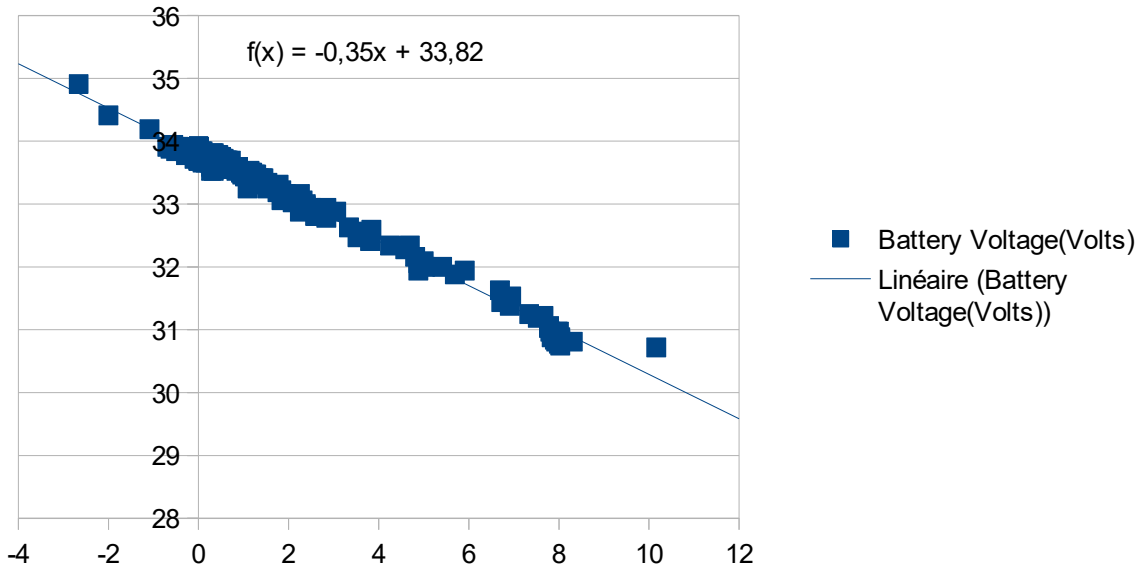
Ci-dessus : U = 37,07 – 0,27 I ; U_0 = 37,07 V ; r = 0,27 Ohms

Modèle électrique à SOC = 50%



Ci-dessus : U = 35,63 – 0,28 I ; U_0 = 35,63 V ; r = 0,28 Ohms = 33,82 V

Modèle électrique à SOC = 25%



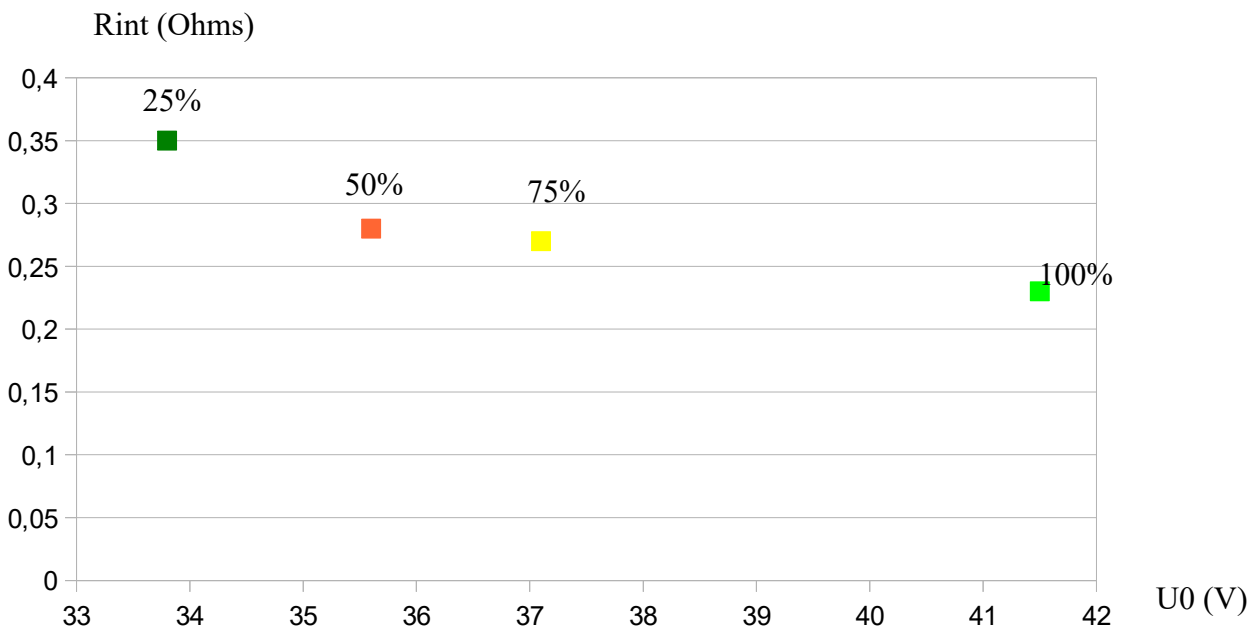
Ci-dessus : U = 33,82 – 0,35 I ; U_0 = 33,82 V ; r = 0,35 Ohms

En résumé

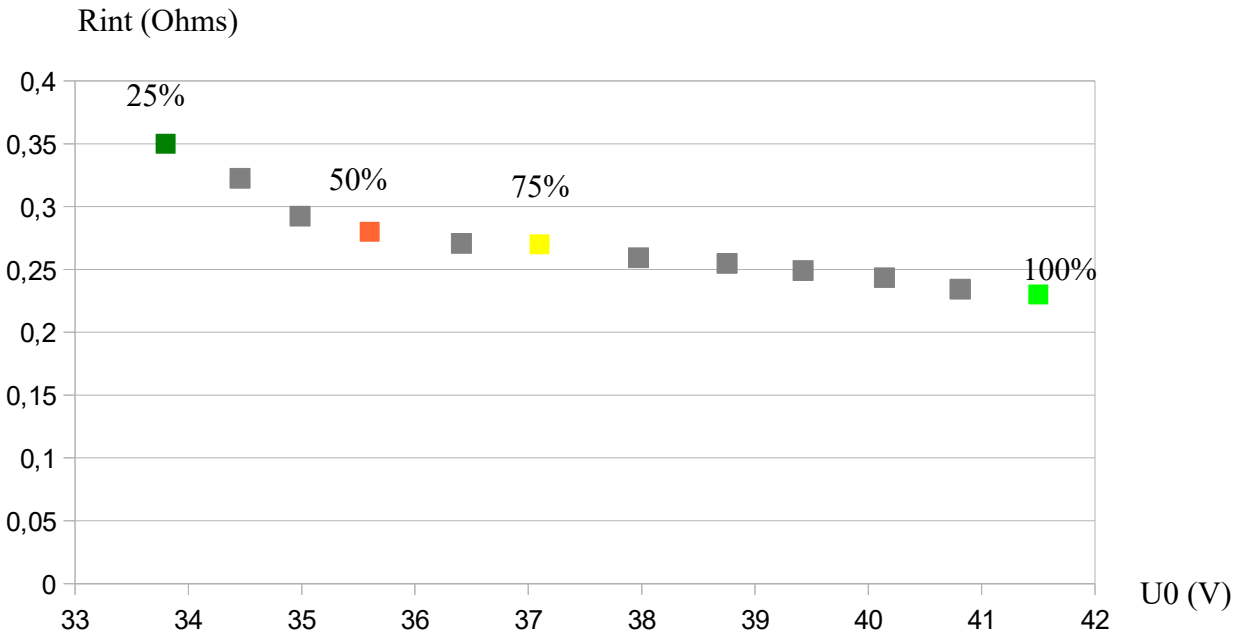
Etat de charge	100%	75%	50%	25%
U_0	41,5	37,1	35,6	33,8
r	0,23	0,27	0,28	0,35

On change de représentation : plan $(U_0 ; r)$

Etat de charge avec 4 niveaux (25% ; 50% ; 75% ; 100%)

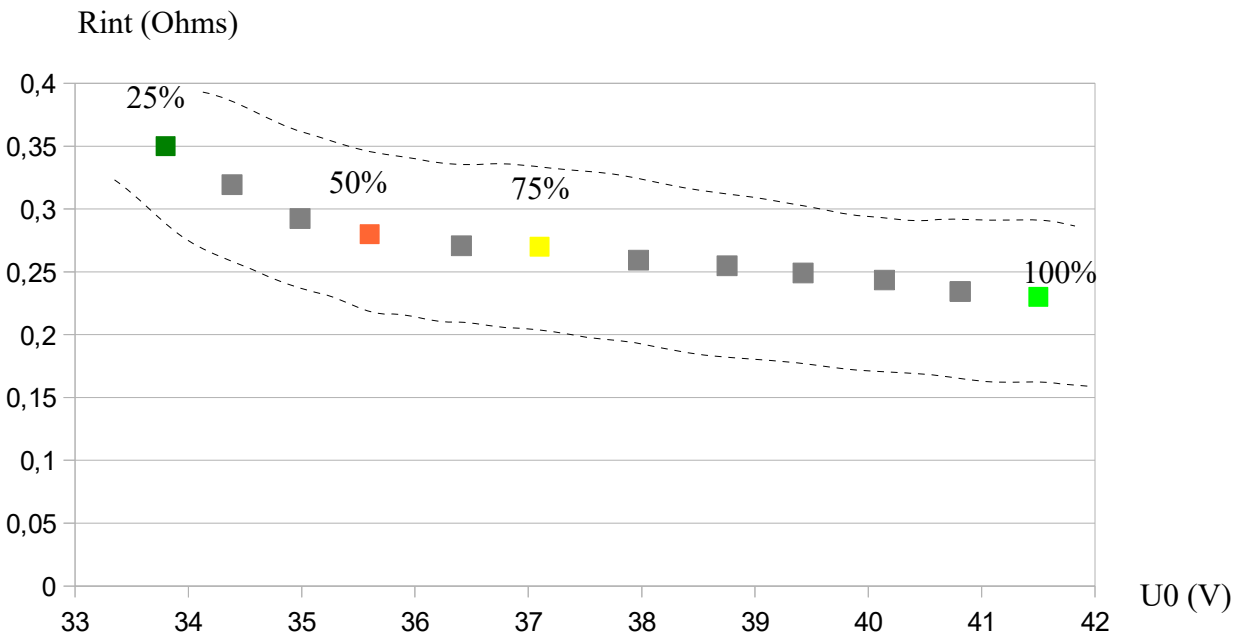


Etat de charge avec une dizaine de niveaux



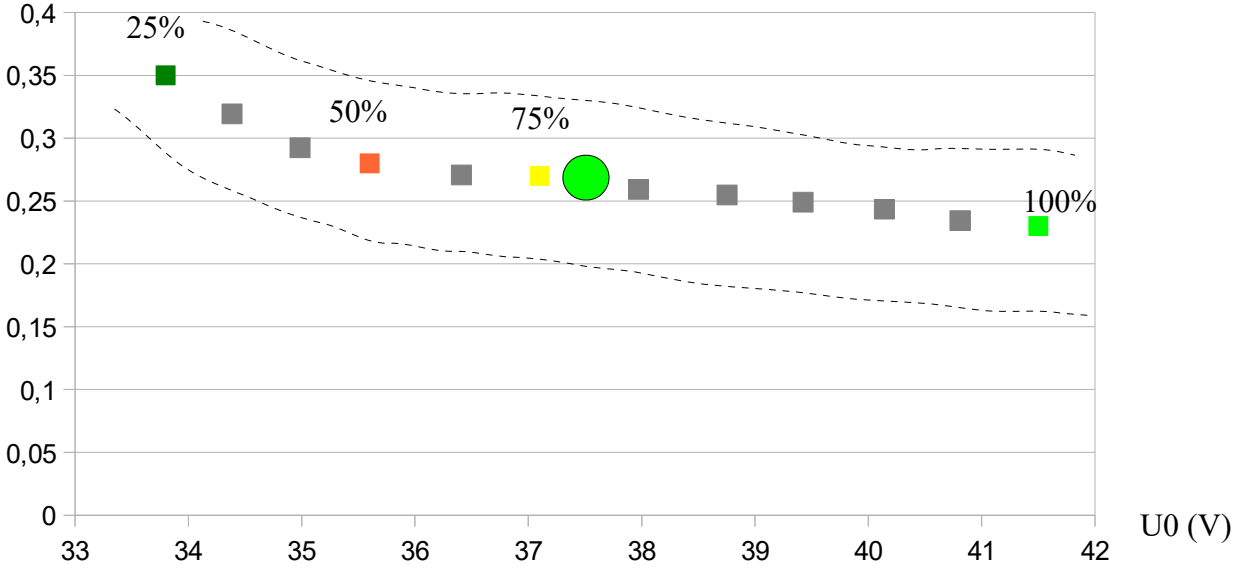
On peut utiliser un algorithme de type "plus proche voisin" pour estimer l'état de charge de la batterie.

Etat de charge + diagnostique

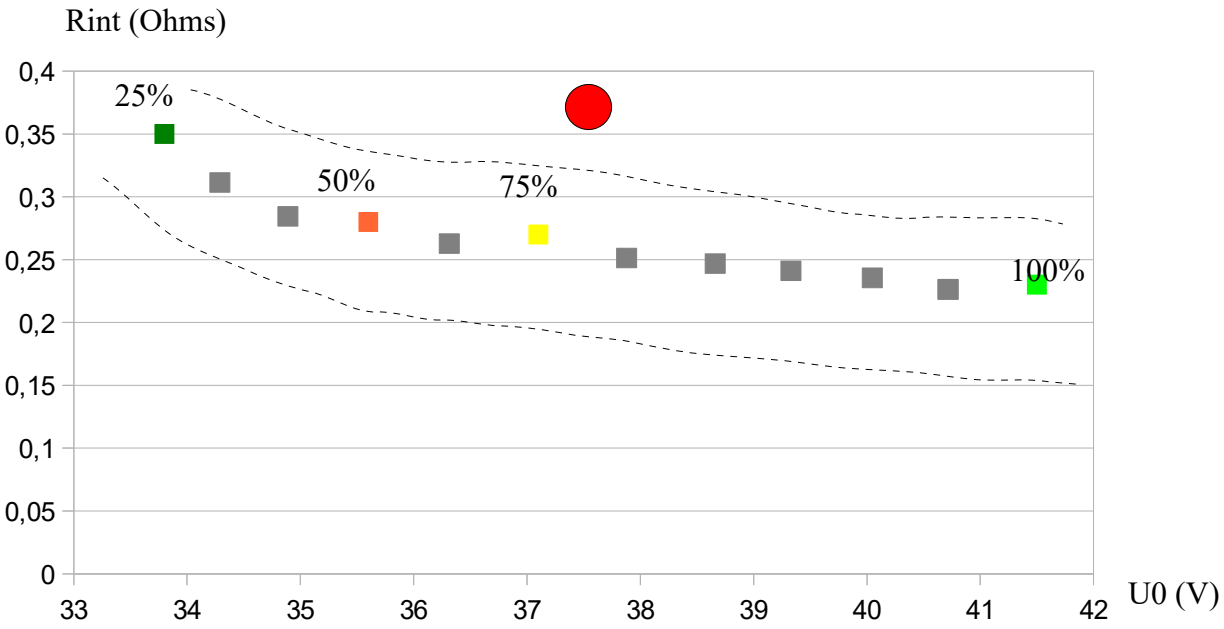


On peut définir deux courbes "enveloppes" pour réaliser du diagnostic que la batterie

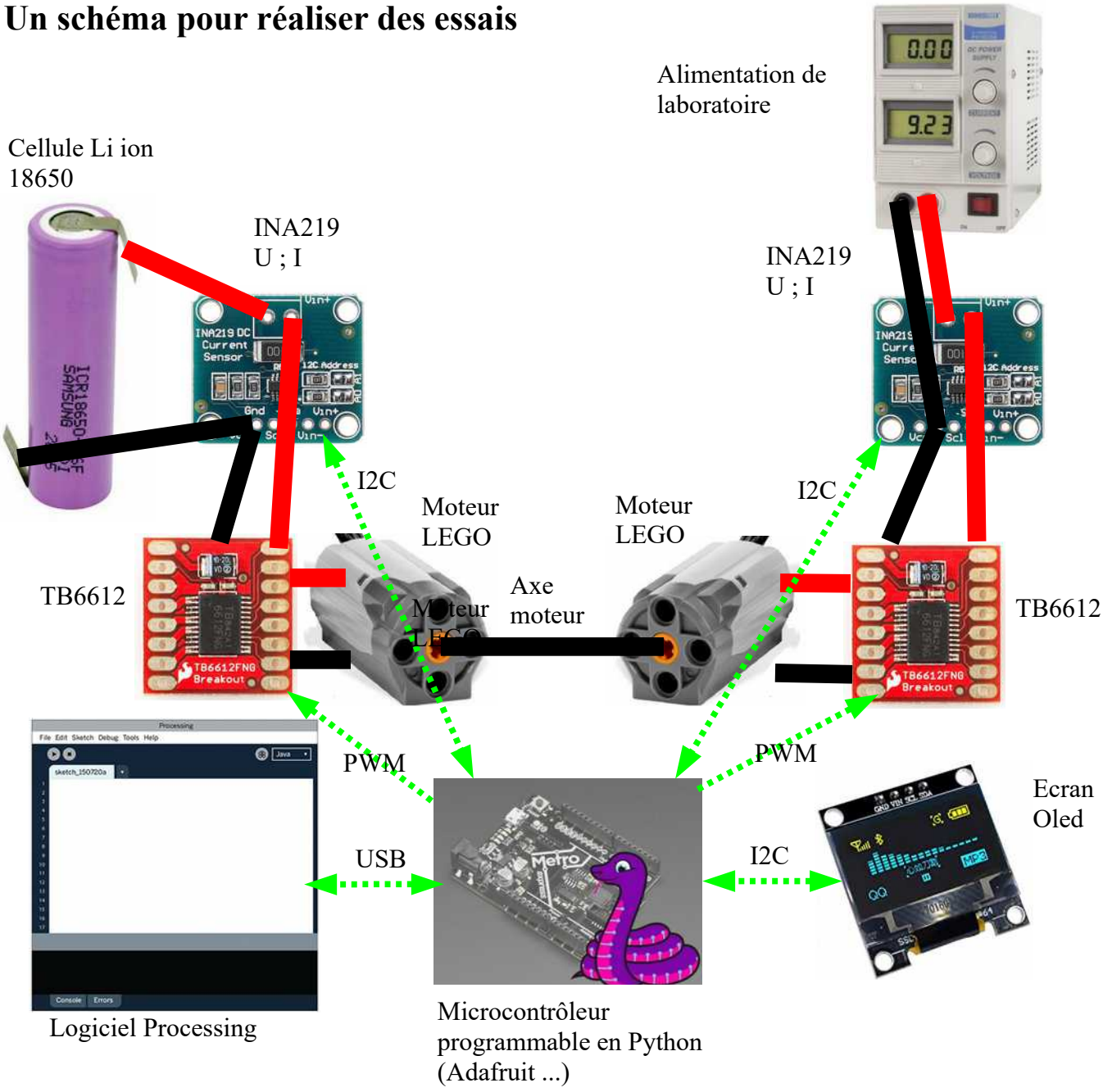
Etat de charge + diagnostique : exemple : SOC = 75% et batterie OK



Etat de charge + diagnostique : exemple : SOC = 75% et batterie DEFAUT



Un schéma pour réaliser des essais



III / Méthode des k plus proches voisins (apprentissage supervisé)

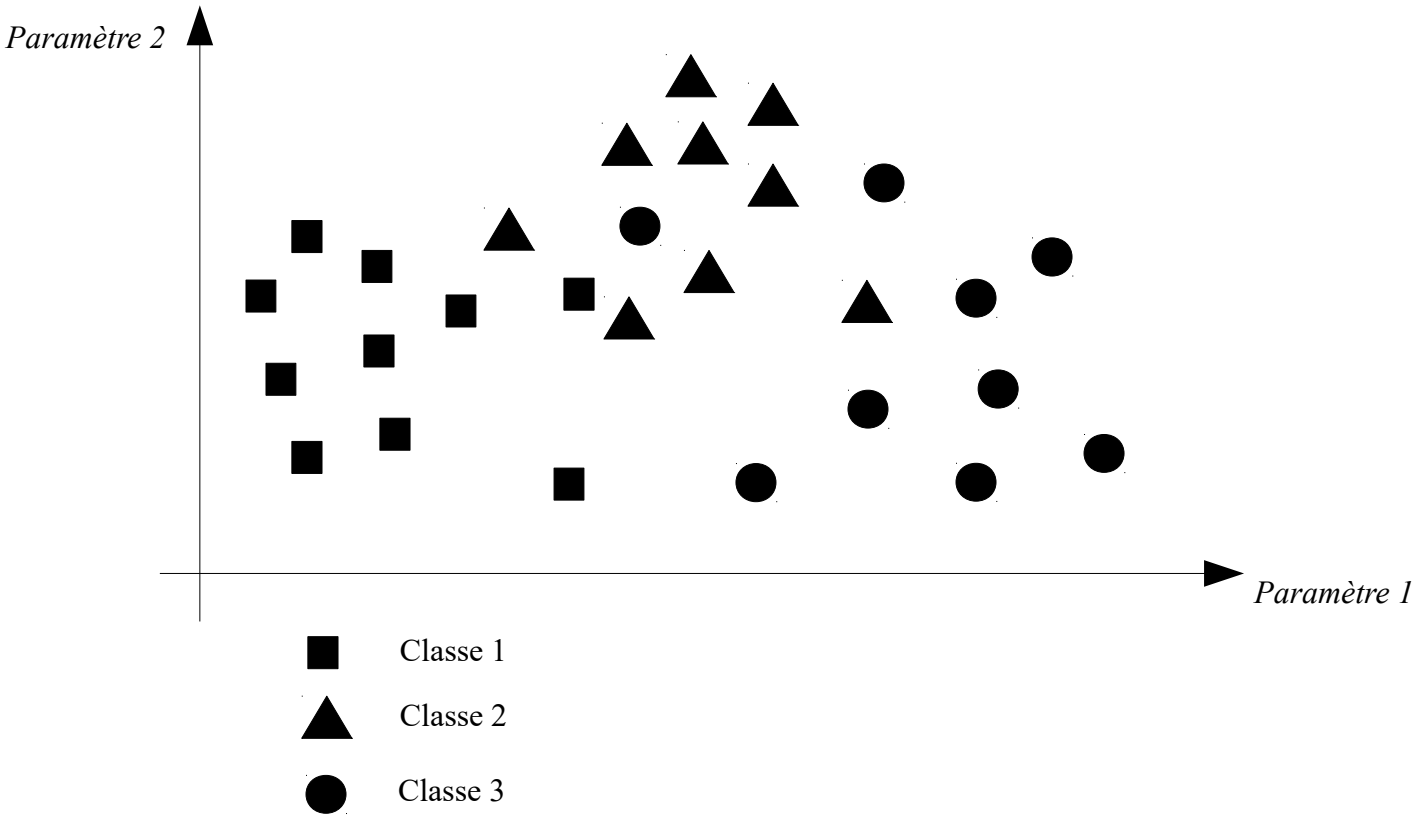
Cours et exemples

La méthode des k plus proches voisins est une méthode de classification qui permet, à partir d'une base d'exemples, d'estimer dans quelle classe se situe un nouvel individu

Cette méthode se décompose en plusieurs étapes :
Phase 1 : apprentissage : correspondance entre paramètres et classes
Phase 2 : arrivée d'un nouvel individu : à partir de ses paramètres, on va chercher dans quelle classe il appartient : on utilise la méthode des k plus proches voisins (où k est une nombre entier, donné. k est nécessairement inférieur au nombre total d'individus ayant servi à l'apprentissage) : on fait la liste de ses k plus proches voisin : on obtient $\{v_1, v_2, \ldots, v_k\}$ puis on cherche la classe modale (celle qui a le plus grand effectif) des individus $\{v_1, v_2, \ldots, v_k\}$

Phase 1 : apprentissage
Nous n'avons aucune connaissance particulière de notre problème. On va uniquement se baser sur des expériences consistant à relever des individus et à noter leur caractéristiques (paramètres : par exemple, taille, poids, couleur,...) et on déterminera ensuite dans quelle classe chacun appartient.

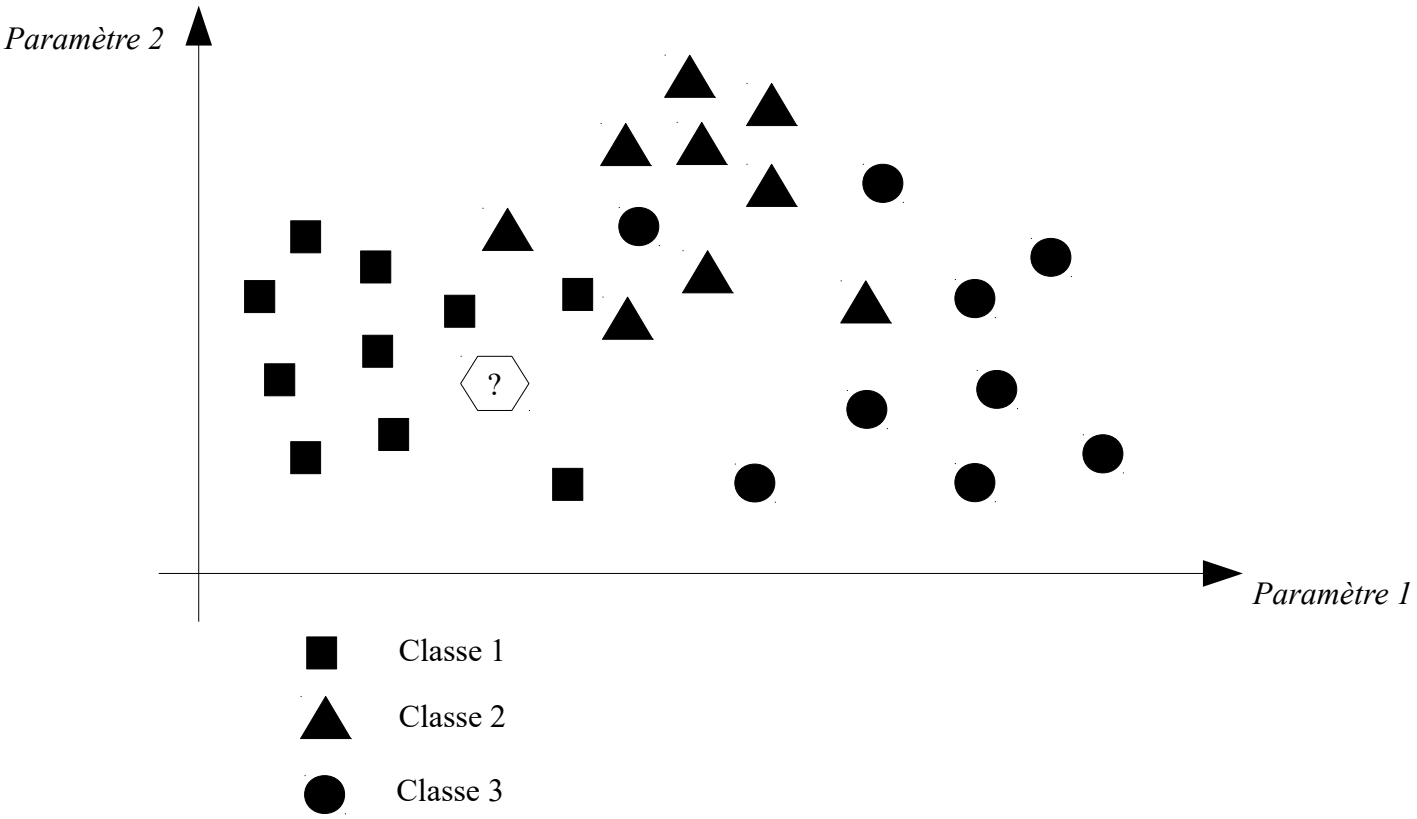
On illustration ci-dessous dans un espace à deux dimensions (2 paramètres) avec 3 classes et 28 individus : voici ce qu'on obtient une fois la phase d'apprentissage réalisée.



Les figures géométriques pleines correspondent à la base d'exemples réalisée lors de la phase d'apprentissage.

Phase 2 : affectation d'un nouvel individu dans une classe
Une fois cet apprentissage réalisé, on dispose d'un nouvel individu dont on ne connait pas la classe et, à partir de ses paramètres, on va estimer à quelle classe il appartient.

Formalisation : on va noter $[v_1, v_2, \dots, v_N]$ les N individus ayant servi à l'apprentissage.



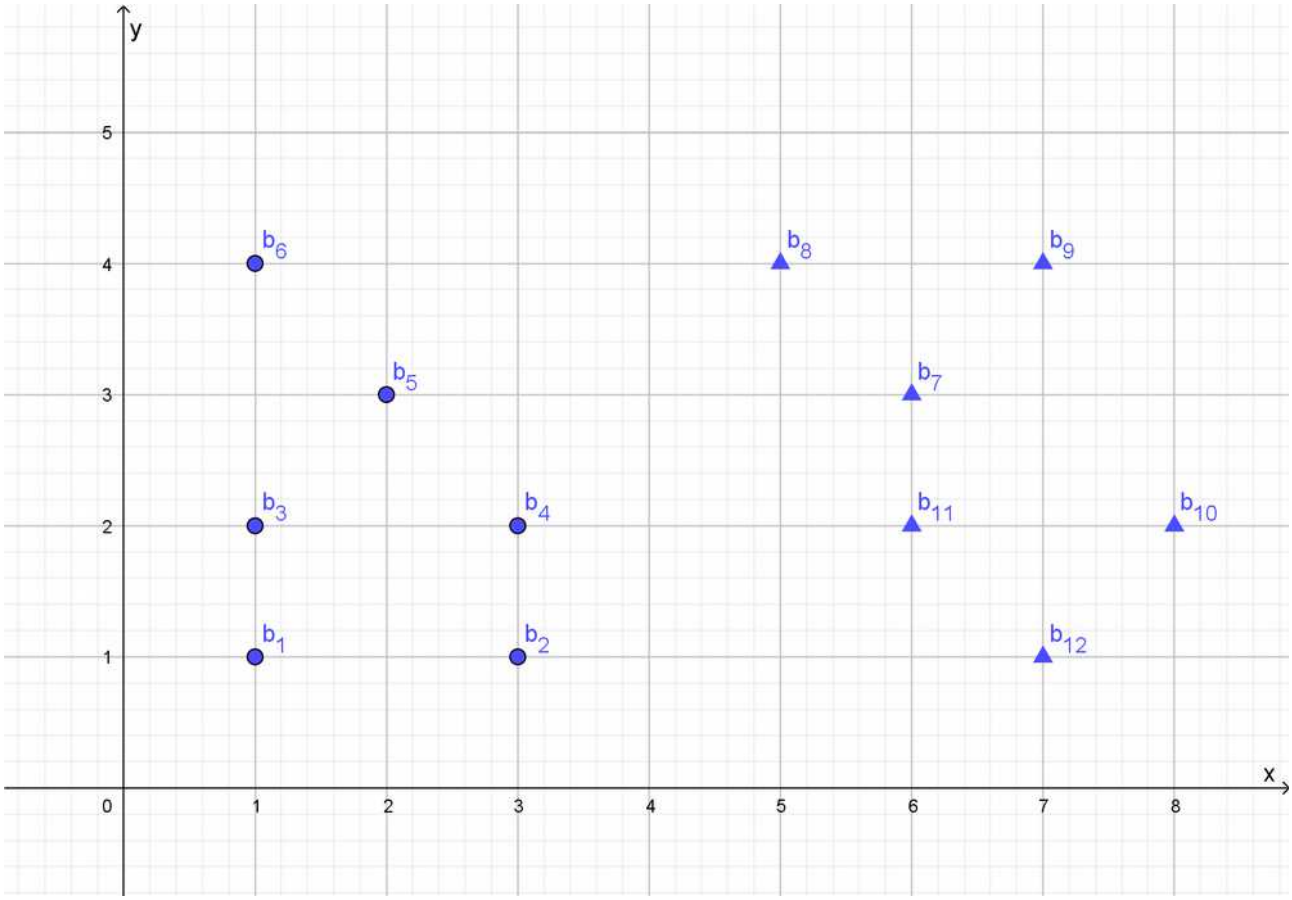
Pour cela, on va se définir une distance entre les individus (il existe plusieurs distances selon le problème rencontré : distance euclidienne : $d(M_1M_2)=\sqrt{(x_2-x_1)^2+(y_2-y_1)^2}$, norme 1 : $d(M_1M_2)=|x_2-x_1|+|y_2-y_1|$, norme infinie $d(M_1M_2)=\max(|x_2-x_1|+|y_2-y_1|)$, distance euclidienne avec pondérations : $d(M_1M_2)=\sqrt{2\times(x_2-x_1)^2+5\times(y_2-y_1)^2}$ ce qui permet de mettre plus ou moins de poids sur certains paramètres.

Puis on calcule la distance de notre nouvel individu noté c à chacun des individus de la base d'apprentissage $[v_1, v_2, \dots, v_N]$: on calcule $\{d(c, v_1), d(c, v_2), \dots, d(c, v_N)\}$: on obtient ainsi une liste de N distances.
On extrait de cette liste les k plus petites valeurs, ce qui donne les k plus proches voisin de c .
Quitte à renuméroter les individus de la base $[v_1, v_2, \dots, v_N]$, on suppose que les k plus proches voisin de c sont $[v_1, v_2, \dots, v_k]$

On regarde les classe de chacun des k plus proches voisin et on fait un décodage majoritaire : on choisit la classe qui est la plus nombreuse : ce sera une estimation de la classe de notre nouvel individu considéré.

Synthèse :
1°) réasliser la phase d'apprentissage sur N individus connus qui servent de base d'exemple et dont on connait tous les paramètres et la classe
2°) on observe un nouvel individu dont on connait les paramètres mais pas la classe
3°) On calcule la liste de toutes les ditances entre ce nouvel individu et les N de la base d'exemples.
4°) On recherche les k plus petites valeurs de la liste ci-dessus
5°) Des ces k individus issus de la base d'exemples, on déterminer la classe majoritaire : c'est le résultat

Exercice corrigé



On considère une population de 12 individus servant à l'apprentissage : ils sont dans la base d'exemples.

Chaque individu possède 2 paramètres : *x* et *y*. Chacun des individus est dans une des deux classe : classe n°1 correspondant aux ronds et classe n°2 correspondant aux triangles.

On observe un nouvel individu c dont les paramètres sont $x_c=3$ et $y_c=4$. Nous allons implémenter l'algorithmes des k plus proches voisin à ce problème avec k = 5 (fixé arbitrairement).

1°) Compléter le tableau ci-dessous

distance	*c - b1*	*c - b2*	*c - b3*	*c - b4*	*c - b5*	*c - b6*	**c - b7**	**c - b8**	**c - b9**	**c - b10**	**c - b11**	**c - b12**
valeur												

Nous avons placé en italique les individus de la base d'exemples de la classe 1 et en gras ceux de la classe 2.

Réponse :

distance	*c - b1*	*c - b2*	*c - b3*	*c - b4*	*c - b5*	*c - b6*	**c - b7**	**c - b8**	**c - b9**	**c - b10**	**c - b11**	**c - b12**
valeur	3,61	3	2,83	2	1,41	2	3,16	2	4	5,39	3,61	5

2°) Déterminer les 5 plus petites distances

Réponse :

distance	*c - b1*	*c - b2*	*c - b3*	*c - b4*	*c - b5*	*c - b6*	**c - b7**	**c - b8**	**c - b9**	**c - b10**	**c - b11**	**c - b12**
valeur	3,61	3	2,83	2	1,41	2	3,16	2	4	5,39	3,61	5

Ce qui donne

distance	*c - b3*	*c - b4*	*c - b5*	*c - b6*	**c - b8**
valeur	2,83	2	1,41	2	2

3°) En déduire la classe modale
On trouve 1 : donc on peut estimer que notre individu c(3,4) appartient à la classe numéro 1 (ce qui peut paraître cohérent par lecture graphique).

4°) Refaire tout le processus avec l'individu d(4,3)

5°) Coder et tester en langage Python un programme qui, à partir d'un entier k, d'une liste de N exemples (avec N supérieur à k) décrite par les tuples (classe , valeur_de_x , valeur_de_y) et d'un individu c(x_c , y_c) donner la classe de cet individu par la méthode des k plus proches voisins

On pourra :
- Coder une fonction informatique qui, à partir de la liste des N exemples et de l'individu c renvoie la liste des N distances

- Coder une fonction qui détermine la distance entre deux individus (on prendra pour commencer la norme euclidienne).

- Coder une fonction informatique qui, à partir de la liste des N distances renvoie la liste des indices correspondant aux k plus petites valeurs

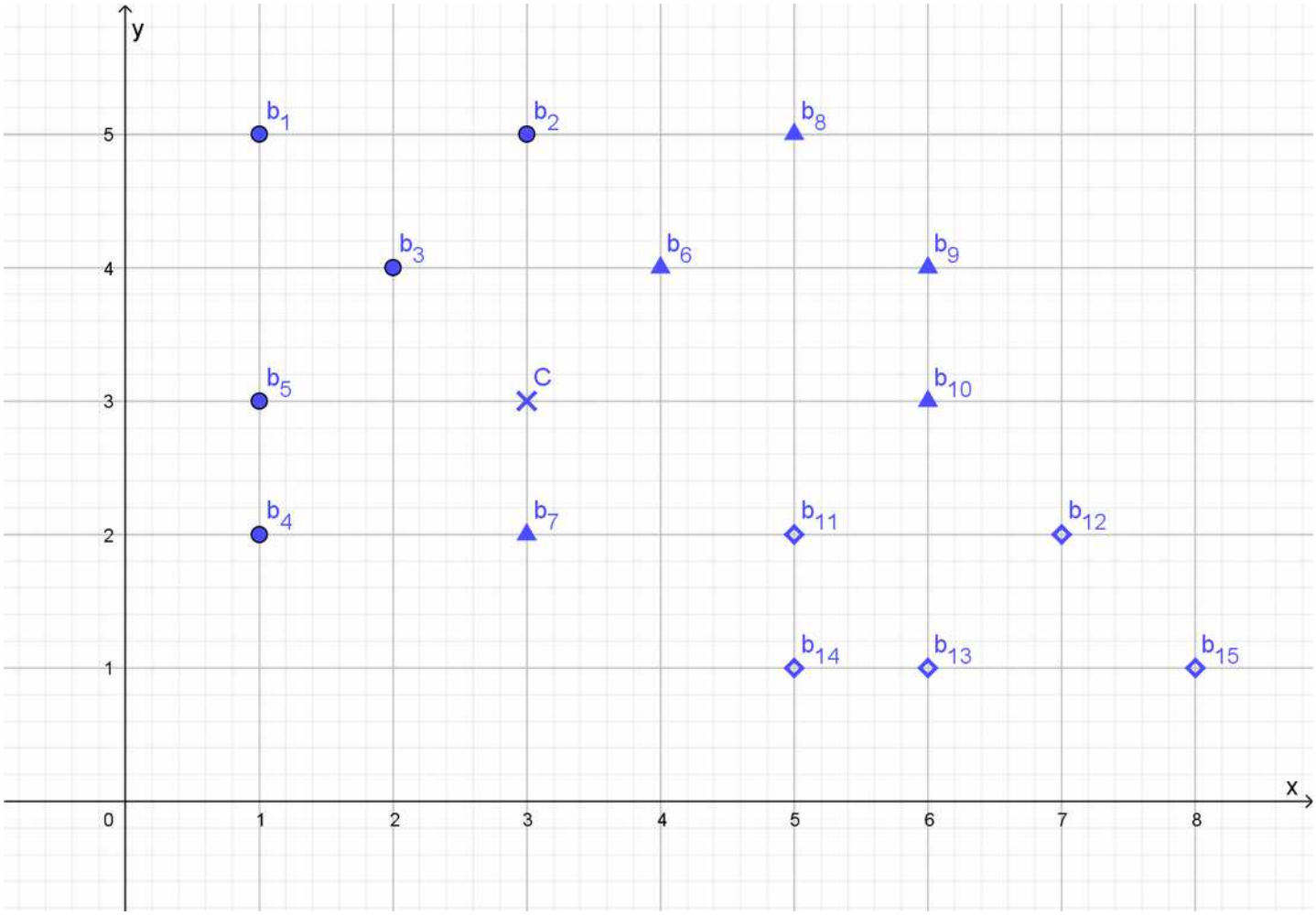
- Coder une fonction informatique qui détermine les k voisins les plus proches de l'individu

- Coder une fonction informatique qui, à partir des k plus proches voisin, détermine la classe majoritaire.

- Faire des tests pour plusieurs individus

- Refaire cdes tests et comparer les résultats lorsqu'on change la distance : norme 1, norme infinie. Norme euclidienne avec pondération de 2 sur x et 5 sur y

Exercice en autonomie
On considère les 15 exemples ci-dessous b1 à b15 répartis dans 3 classes (disques, triangles et losanges)
On observe l'individu C



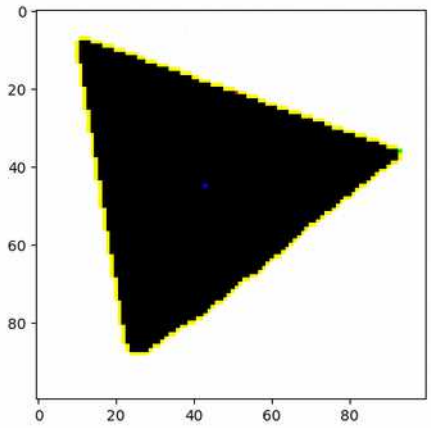
En utilisant la méthode des 4 plus proches voisins, déterminer la classe à laquelle appartient l'individu C.

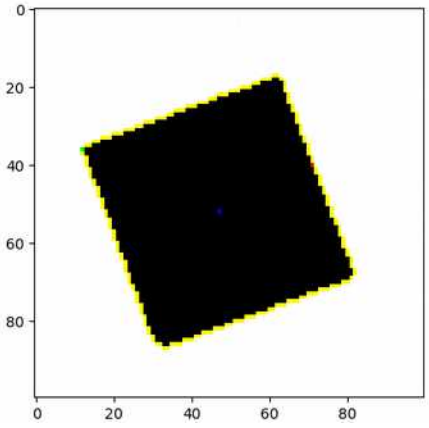
Activité : reconnaissance de formes

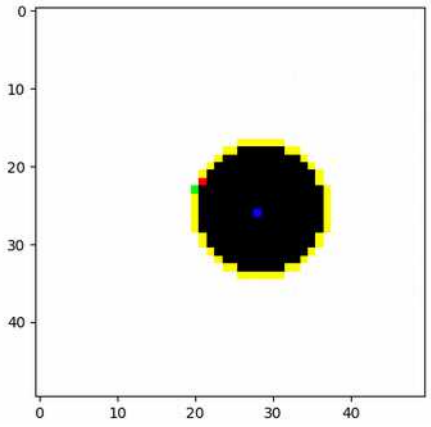
Expliquer les deux indicateurs adimentionnels

Exercice : dans le cas de triangles équilatéraux, de carrés et de disques, calculer ces indicateurs.

Activité 1 : à partir des bases d'exemples (30 au total), construire un programme basé sur la méthode des k plus proches voisins pour reconnaitre des figures géométriques). Faire l'essai avec votre téléphone portable.





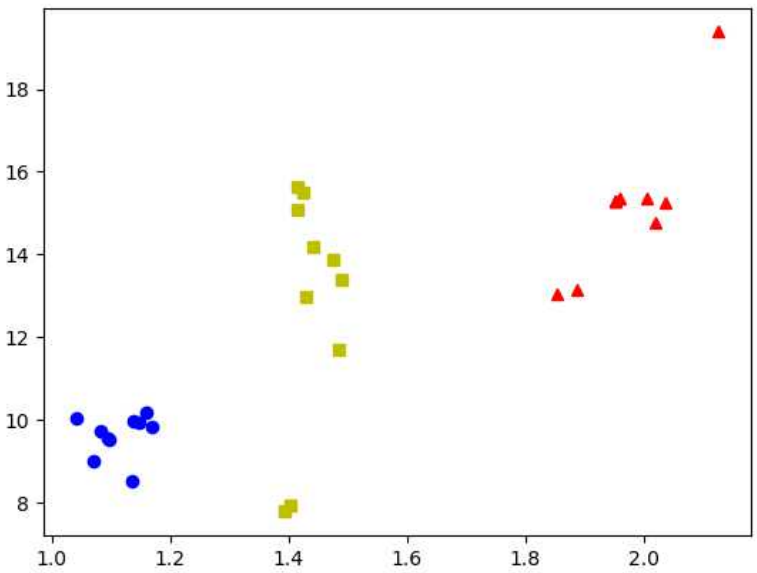


Triangle	1	2	3	4	5	6	7	8	9	10
PGD/PPD	1.886	1.959	1.952	1.952	1.952	2.126	1.852	2.005	2.020	2.037
P^2/A	13.157	15.367	15.277	15.277	15.277	19.386	13.056	15.364	14.787	15.265

Carré	1	2	3	4	5	6	7	8	9	10
PGD/PPD	1.414	1.414	1.425	1.394	1.404	1.489	1.442	1.476	1.485	1.429
P^2/A	15.638	15.073	15.501	7.780	7.919	13.380	14.168	13.886	11.710	12.978

Disque	1	2	3	4	5	6	7	8	9	10
PGD/PPD	1.137	1.041	1.084	1.147	1.160	1.094	1.097	1.170	1.139	1.071
P^2/A	8.526	10.056	9.711	9.940	10.194	9.572	9.536	9.843	9.966	9.0

P^2/A



PGD/PPD

Activité 2 : diagnostique de panne avec des moteurs lego

Faire des mesures et collecter des exemples :
- en fonctionnement normal
- avec un moteur électrique déconnecté électriquement (panne électrique)
- avec un arbre moteur cassé (panne mécanique)

Dans chaque car, on modélise le lien entre U et I par U = a + b I
On place les points (a ; b) dans le plan pour chacun de ces 3 cas.

Ensuite, le diagnostique se fait par la méthode des k plus proches voisins.

Idée : rajouter une connexion Bluetooth et envoyer des alertes sur smartphone (voire avec Arduino et Processing).

IV / Réseaux de neurones : perceptron

Cours

1. Modélisation d'un neurone:

(schéma)

- deux entrée e_i telles que $e_1 = x$ et $e_2 = y$ (où $1 \leqslant i \leqslant 2$)
- le neurone "intègre" toutes ses entrées par $\sum_{i=1}^{2} p_i \, e_i = E$ (où les p_i sont les poids des synapses)
- la réponse du neurone est régie par une fonction de seuil ϕ:

$$\phi(x) = \begin{cases} 1 & \text{si } S \geqslant \text{seuil } S_0 \\ 0 & \text{sinon} \end{cases}$$ (*graphe)*

- la sortie du neurone est donc $\phi(E) = Y = \begin{cases} 1 \\ \text{ou} \\ 0 \end{cases}$

En résumé:
À l'entrée $X \begin{pmatrix} x \\ y \end{pmatrix}$, le neurone répond par $Y = \phi\left(p_1 \, x + p_2 \, y\right)$

2. Perceptron élémentaire

(Déf. ?) Un perceptron est composé d'une couche de neurones dits "*associatifs"* et d'une couche au moins de neurone(s) dit(s) "*de décision"*.

(schéma)

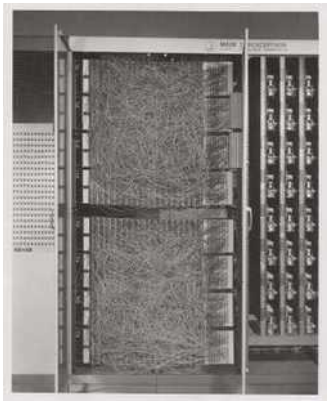
remarques:

- la propagation des signaux est unidirectionnelle: des "inputs" vers les "outputs"
  (C'est à dire des entrées vers les sorties.)
- Les neurones ne sont pas connectés entre eux au sein d'un même couche, et aucun neurone n'est connecté à lui-même.

schéma 1 *schéma 2*

- historique: Rosenblatt (1958), simulation du premier perceptron
  (de *wikipedia.fr*, juin 2019) **Frank Rosenblatt** (né à New York le 11 juillet 1928 - mort le 11 juillet 1971) était un psychologue américain qui travailla sur l'intelligence artificielle. Principal représentant du « courant neuronal », qui voulait construire celle-ci à partir de la conception du réseau neuronal humain, il fabriqua sur ce modèle le perceptron en 1958 à l'Université Cornell.

Frank Rosenblatt en 1958



IBM 704

3. Phase d'apprentissage et règle de Hebb

La règle de Hebb (1949):

(*wikipedia.fr*, juin 2019) La **règle de Hebb**, **théorie de Hebb**, **postulat de Hebb** ou **théorie des assemblées de neurones** a été établie par Donald Hebb en 1949. Elle est à la fois utilisée comme hypothèse en neurosciences et comme concept dans les réseaux neuronaux en mathématiques.

Cette théorie est souvent résumée par la formule : « des neurones qui s'excitent ensemble se lient entre eux. » (« *cells that fire together, wire together* »)1 C'est une règle d'apprentissage des réseaux de neurones artificiels dans le contexte de l'étude d'assemblées de neurones.

schéma 1 *schéma 2*

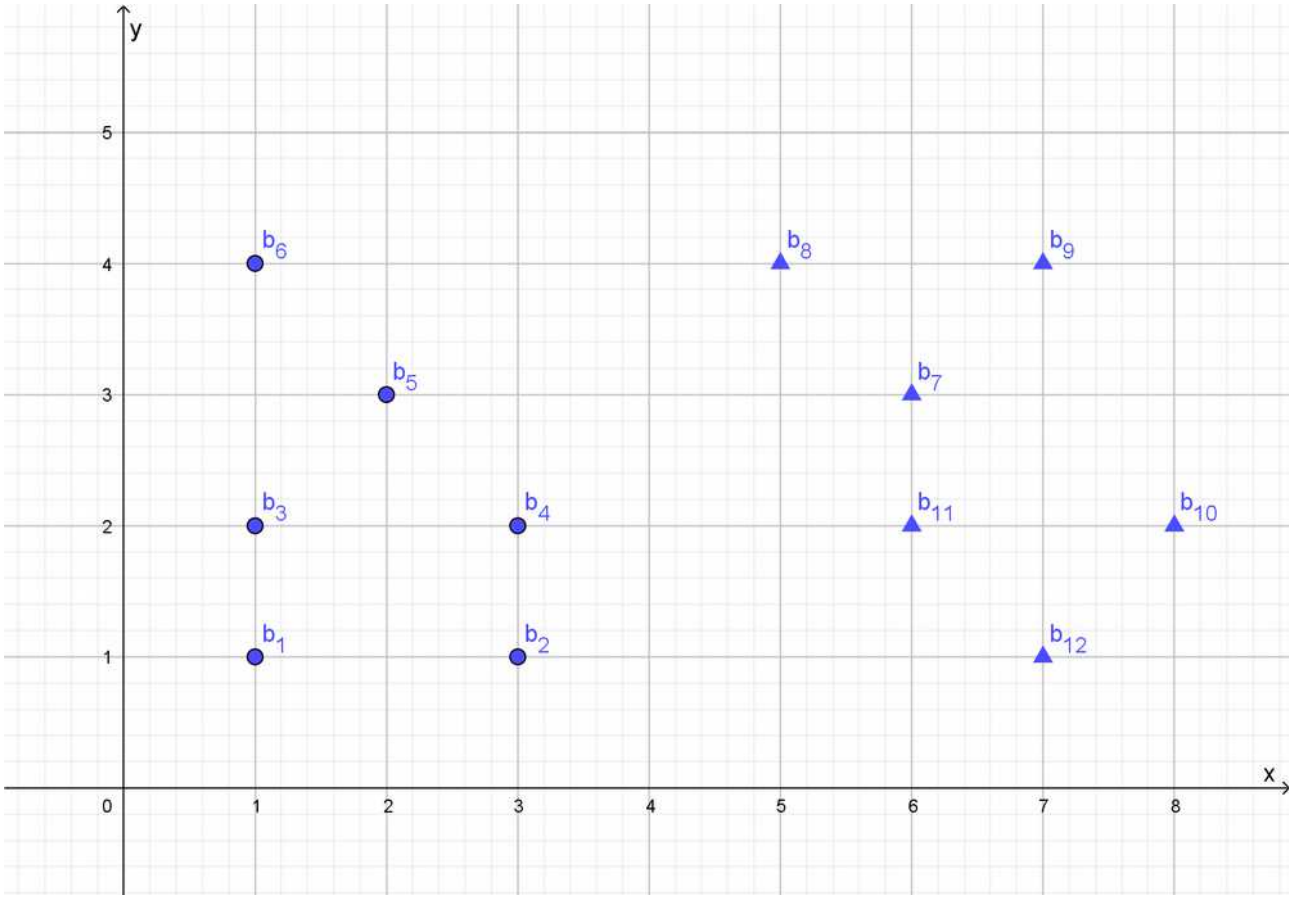
Illustrer par l'expérience de psycho-physiologie des chiens de Pavlov:
Entendre la sonnerie suffit à faire saliver les chiens au bout d'un certain temps de conditionnement (=apprentissage).
Les circuits de la salivation sont actifs en même temps que ceux de l'audition de la sonnerie.
On suppose que les connections entre ces circuits neuronaux sont renforcées.
Concrètement, pour le modèle du perceptron, cela revient à augmenter les poids p_i entre deux neurones actifs, et à les diminuer sinon.

schéma

Soit $p_i(t)$ le poids symaptique à l'instant t.
Si A est l'état du neurone pré-synaptique et si B l'état du neurone post-synaptique, alors, à l'instant $t + \Delta t$:

$$p_i(t + \Delta t) = p_i(t) + k \times A \times B$$ (où *k* est un coefficient dit d'apprentissage)

Méthode d'apprentissage du perceptron:
Un exemple:
On veut que le perceptron réponde 1 lorsqu'il "voit" des cercles et 0 lorsqu'il "voit" des triangles dans le graphique déjà vu plus haut:



Les poids synaptiques sont aléatoires au départ, par exemple $p_1 = 1$ et $p_2 = 1$.
On fixe arbitrairement le seuil $S_0 = 10$ pour le neurone de décision.

-> on "entre" $b_5 = (x, y) = (3, 4)$
-> Le neurone de décision calcule $\phi(1 \times 3 + 1 \times 4) = \phi(7) = 0$ (car $7 < 10$).
-> C'est une erreur car b_5 est un cercle. On attendait la réponse $Y = 1$.
-> on calcule l'erreur $\epsilon = Y - \phi(b_5) = 1$
-> on modifie les p_i avec la règle de Hebb:

- $p_1 \leftarrow p_1 + k \times \epsilon \times x$
- $p_2 \leftarrow p_2 + k \times \epsilon \times y$

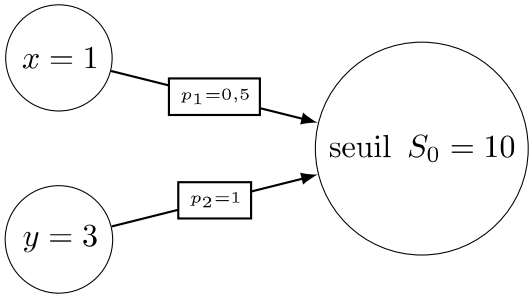
En fixant *k* (arbitrairement) à $0,5$, p_1 passe donc à $1 + 0,5 \times 1 \times 3 = 2,5$.
Et p_2 passe à $1 + 0,5 \times 1 \times 4 = 3$.

On recommence jusqu'à obtenir une erreur nulle en sortie:
-> on entre à nouveau b_5, $\phi(b_5) = \phi(2,5 \times 3 + 3 \times 4) = \phi(19,5) = 1$ (car $19,5 \geqslant S_0 = 10$)
et c'est déjà la bonne réponse, donc on passe à un autre point.

Quelques exercices sur la théorie

Exercice 1

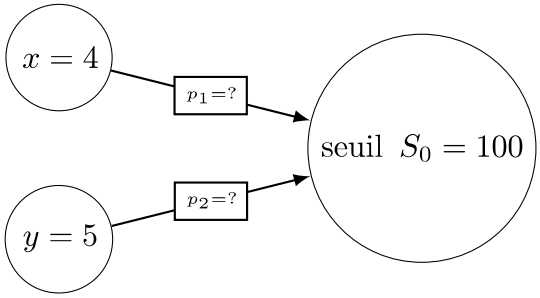
Pour les perceptrons simples suivants, dire que vaut la sortie avec les entrées proposées:



etc ...

Exercice 2

Déterminez des poids possibles p_1 et p_2 pour que le neurone de décision soit activé.



etc ...

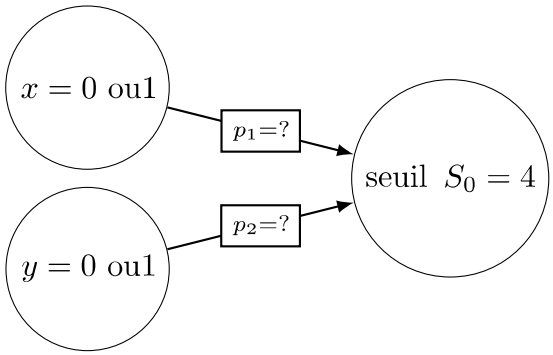
Exercice 3

Déterminez des poids possibles p_1 et p_2 pour que le neurone de décision **ne** soit **pas** activé.

Exercice 4

Les entrées x et y sont des 0 ou des 1.

Déterminez des poids possibles p_1 et p_2 pour que le neurone de décision réponde l'opération booléenne $x \vee y$



Exercice 5

Même chose pour $x \wedge y$.

Exercice 6

Montrez que la réponse x XOR y n'est pas possible.

Exercice 6

On souhaite avoir un perceptron simple à une seule cellule de décision de seuil $S_0 = 10$, qui réponde

- 0 pour l'entrée $(x, y) = (1, 2)$
- et 1 pour l'entrée $(x, y) = (5, -3)$

1. Montrer que cette situation revient à avoir:
$$\begin{cases} p_1 + 2p_2 & \leqslant & 10 \\ 5p_1 - 3p_2 & \geqslant & 10 \end{cases}$$
2. Déterminez si possible, des poids p_1 et p_2 qui conviennent.

D'autres exercices

- sur des perceptrons à trois ou plus, cellules d'entrées
- à deux , peut-être trois, cellules de décision (sortie)

Quelques exercices sur le codage

Exercice 7

Soit un perceptron simple constitué de deux neurones d'entrée et d'un neurone de décision de seuil $S_0 = 15$.

schéma

On pose:

- $X = [x, y]$ la variable codant l'entrée
- $P = [p_1, p_2]$ la variable codant les poids du neurones de décision (p_1 s'appliquant à *x* et p_2 s'appliquant à *y*)

Écrire une fonction en Python appelée "sortie(X,P)" retournant soit 0, soit 1 qui simule le fonctionnement du neurone de décision.

Exercice 7 et 8

Même exercice, en utilisant respectivement une fonction affine par morceaux et une fonction sigmoide:

(à tracer)

- $f_1(x) = \begin{cases} 0 & si \quad x \leqslant \frac{S_0}{2} \\ \frac{2}{S_0}x - 1 & si \quad \frac{S_0}{2} < x \leqslant S_0 \\ 1 & si \quad x > S_0 \end{cases} \qquad (S_0 \neq 0)$
- $f_2(x) = \dfrac{1}{1 + \exp(S_0 - x)}$

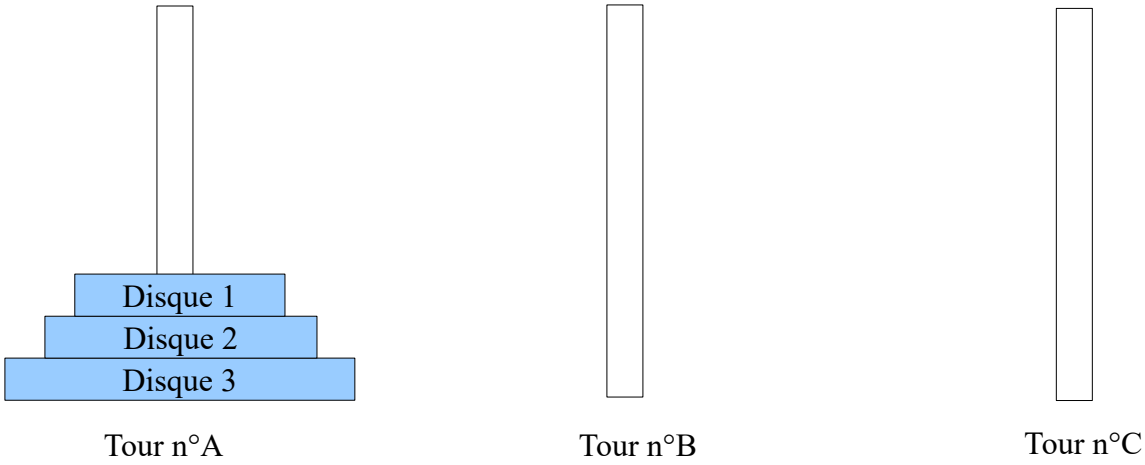
Exercice 9

Donner l'algorithme d'apprentissage en langage naturel et demander de le coder en Python.

Réalisation en groupes ?

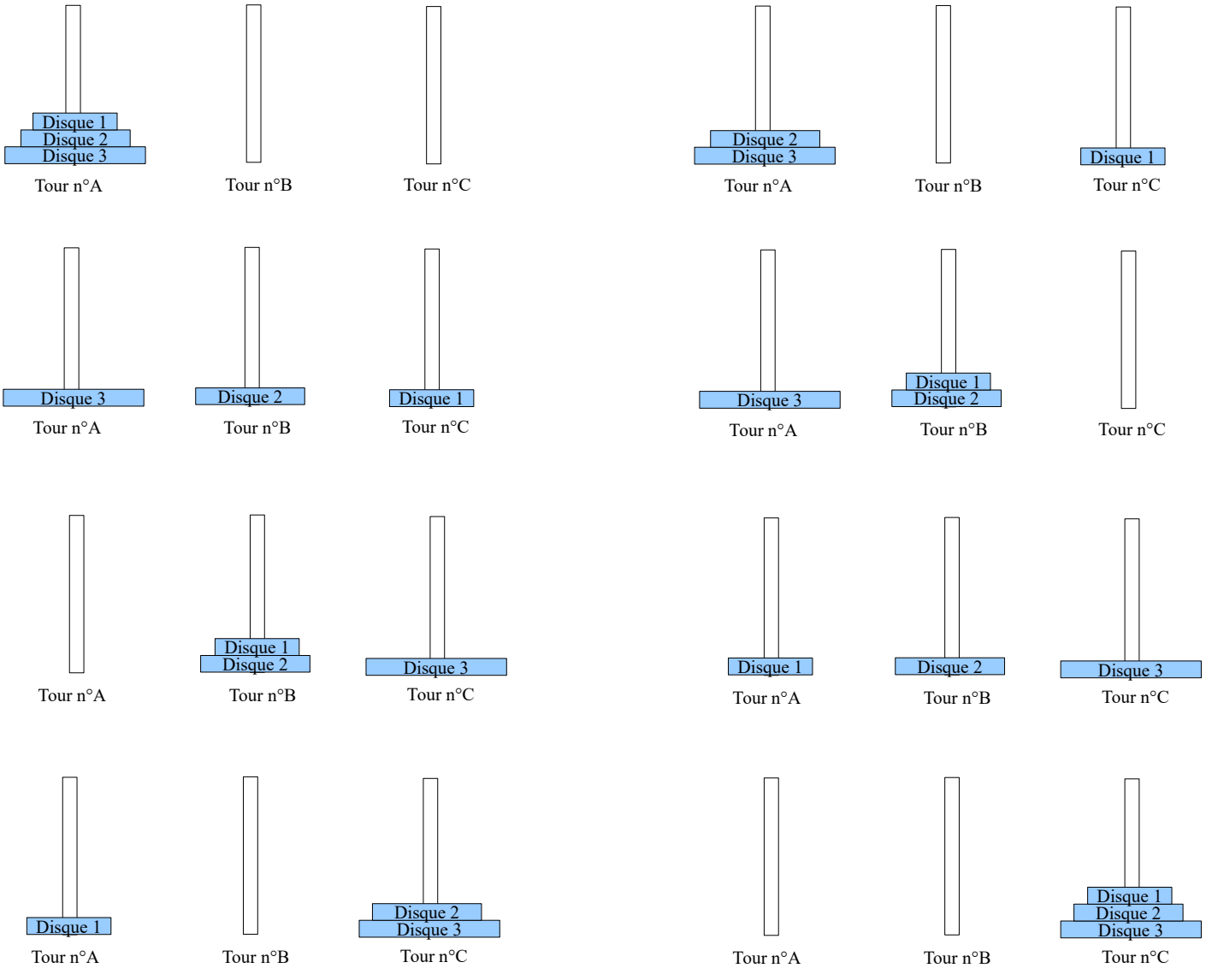
V / Jeux et IA

- Tours de Hanoï



Objectif : on souhaite déplacer les disques de la tour n°A à la tour n°C en respectant les règles suivantes :
- on ne déplace qu'un seul disque à la fois
- on ne peut pas poser un disque plus grand sur un disque plus petit

Exercice 1 : résolution manuelle avec 3 disques

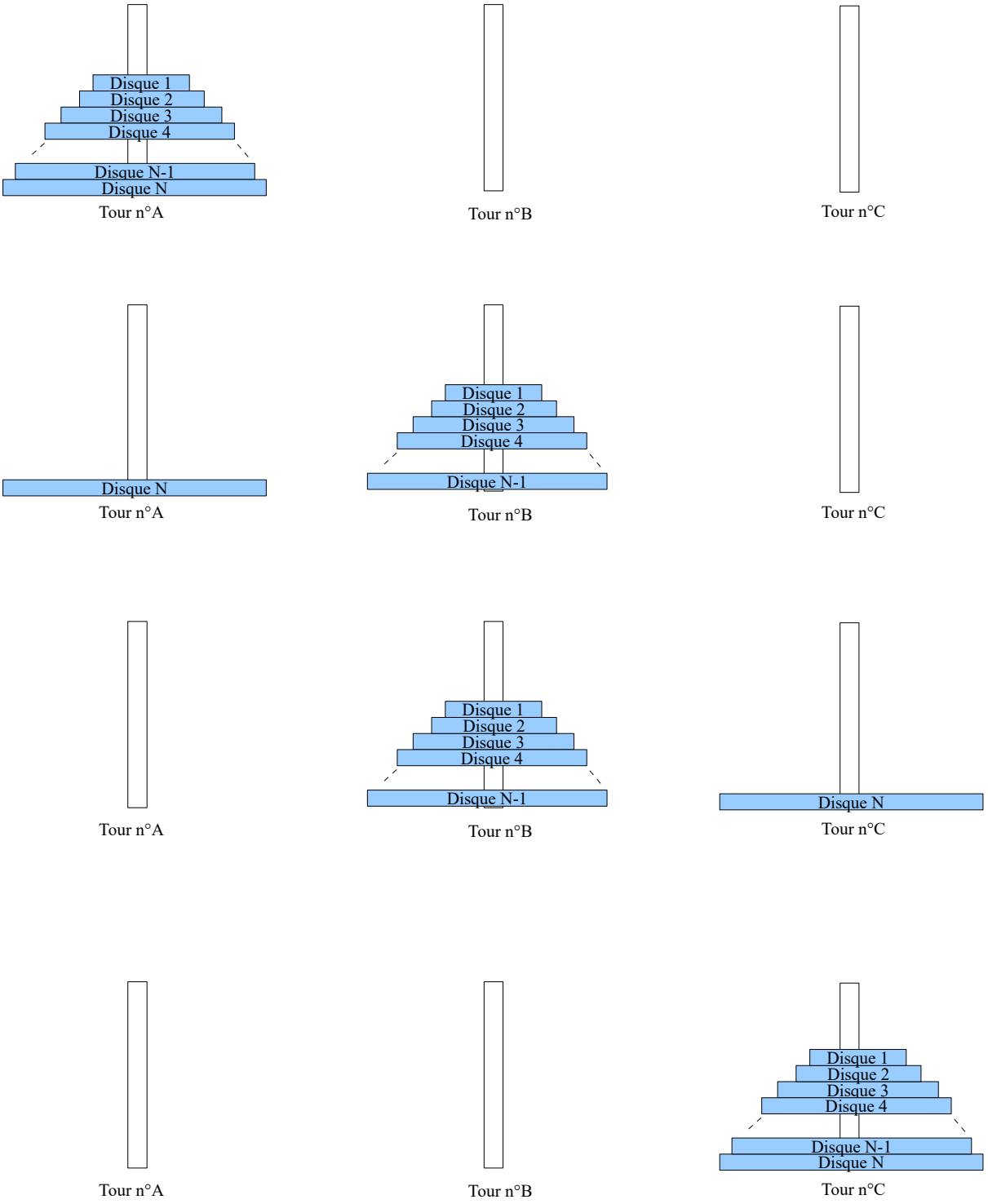


Exercice 2 : résolution dans la cas général avec N disques

Exemple de résolution manuelle avec :

- la possibilité de ne déplacer qu'un seul disaque à la fois

- la possibilité de déplacer une tour complète de taille "N-1"



Exercice 3 : *Réflexion autour de la récursivité et calcul de la complexité.*

On note d_N *le nombre de déplacements nécessaires lorsqu'on a une tour de taille* N

On remarque que lorsque $N=1$ *, alors* $d_1=1$ *: on déplace l'unique disque de la position A à la position C*

Dans le cas général : pour déplacer une tour de taille N *de la position A à la position C, il faut :*

- déplacer la tour de taille $N-1$ *"du dessus" de la position A à la position B, ce qui nécessite* d_{N-1} *déplacements.*

- déplacer le disque numéro N *de la position A à la position C, ce qui nécessite 1 déplacement*

- déplacer la tour de taille $N-1$ *de la position B à la position C, ce qui nécessite* d_{N-1} *déplacements.*

Ainsi, on remarque que $d_N = d_{N-1} + 1 + d_{n-1}$ *:* $d_1 = 1$ *et* $d_N = 2 \times d_{N-1} + 1$
On peut alors calculer le nombre d'opérations en fonction de N *de "proche en proche".*

N	1	2	3	4	5	6	7	8	9	10
d_N	1	3	7	15	31	63	127	255	511	1023

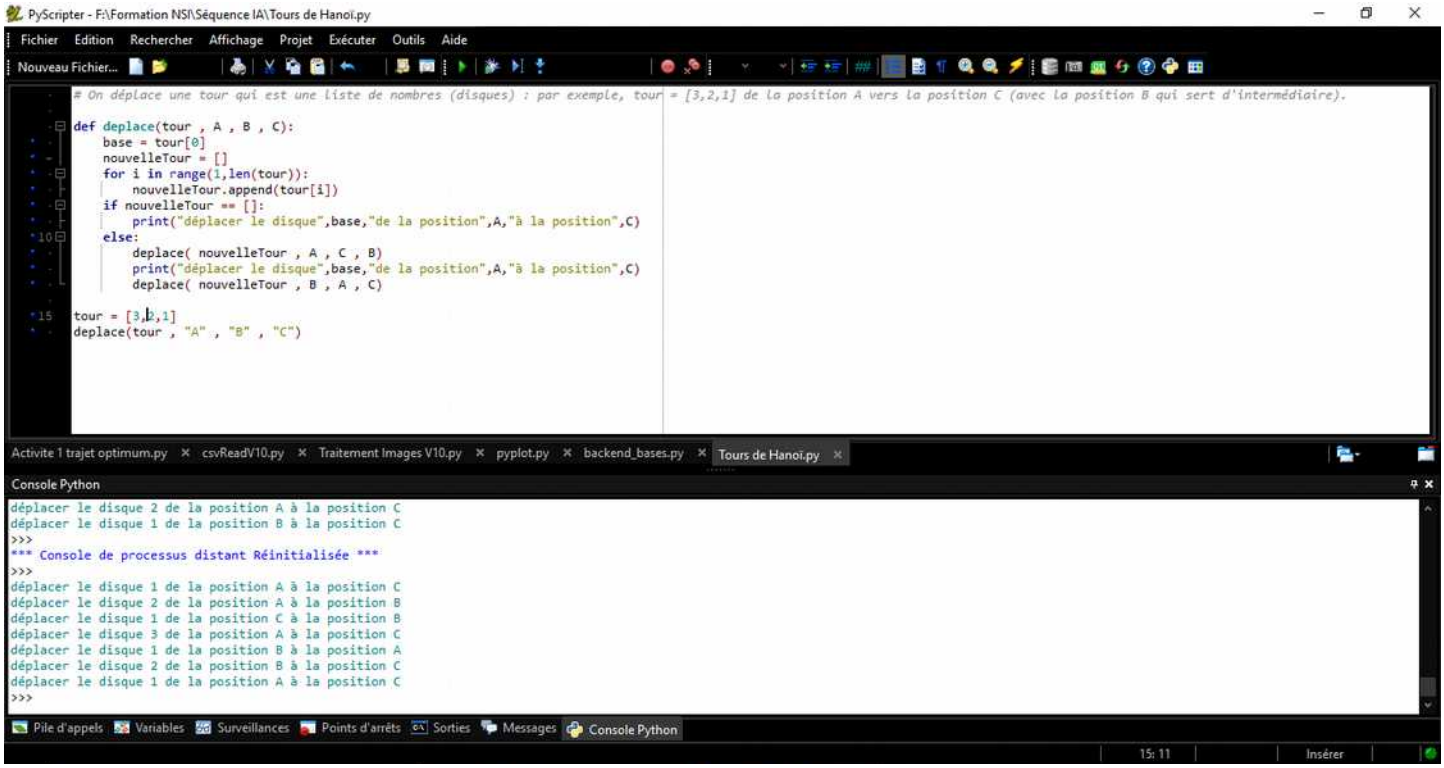
On peut faire la conjecture / démonstration que $d_N = 2^N - 1$

Exercice 4 / activité :*Programme en Python :*

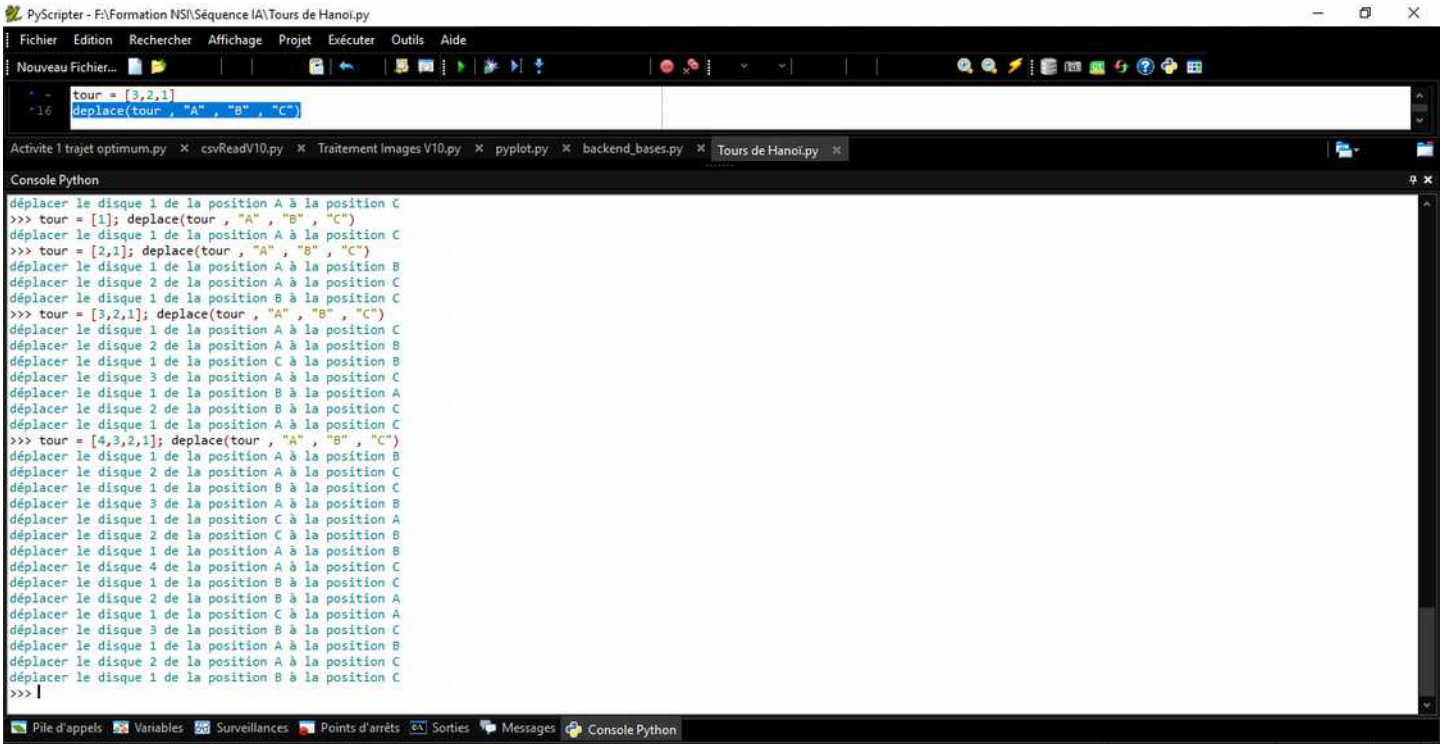
 # On déplace une tour qui est une liste de nombres (disques) : par exemple, tour = [3,2,1] de la position A vers la position C (avec la position B qui sert d'intermédiaire).

```
def deplace(tour , A , B , C):
    base = tour[0]
    nouvelleTour = []
    for i in range(1,len(tour)):
        nouvelleTour.append(tour[i])
        if nouvelleTour == []:
            print("déplacer le disque",base,"de la position",A,"à la position",C)
        else:
            deplace( nouvelleTour , A , C , B)
            print("déplacer le disque",base,"de la position",A,"à la position",C)
            deplace( nouvelleTour , B , A , C)

tour = [3,2,1]
deplace(tour , "A" , "B" , "C")
```



Quelques résultats



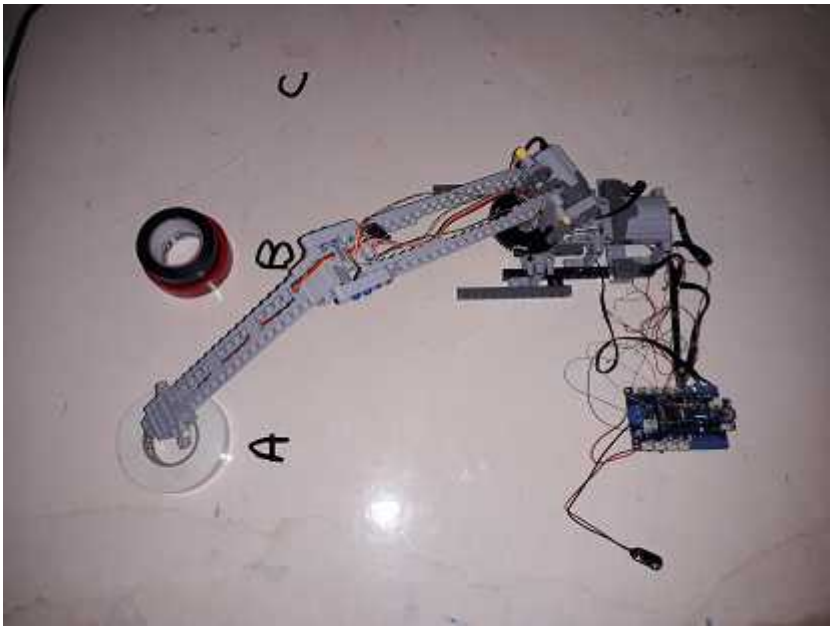
Quelques modifications pour compter le nombre de déplacements

```
global d # On définit une variable globale pour compter le nombre de déplacements
d = 0

# On déplace une tour qui est une liste de nombres (disques) : par exemple, tour
# = [3,2,1] de la position A vers la position C (avec la position B qui sert
# d'intermédiaire).

def deplace(tour , A , B , C):
    global d # On informe le programme que cette variable est globale
    base = tour[0]
    nouvelleTour = []
    for i in range(1,len(tour)):
        nouvelleTour.append(tour[i])
    if nouvelleTour == []:
        print("déplacer le disque",base,"de la position",A,"à la position",C)
        d = d+1 # On a fait un déplacement de plus
    else:
        deplace( nouvelleTour , A , C , B)
        print("déplacer le disque",base,"de la position",A,"à la position",C)
        d = d+1 # On a fait un déplacement de plus
        deplace( nouvelleTour , B , A , C)
```

Activité / idée de projet : pour aller plus loin (projet avec les élèves : tours de Hanoï et robotique)

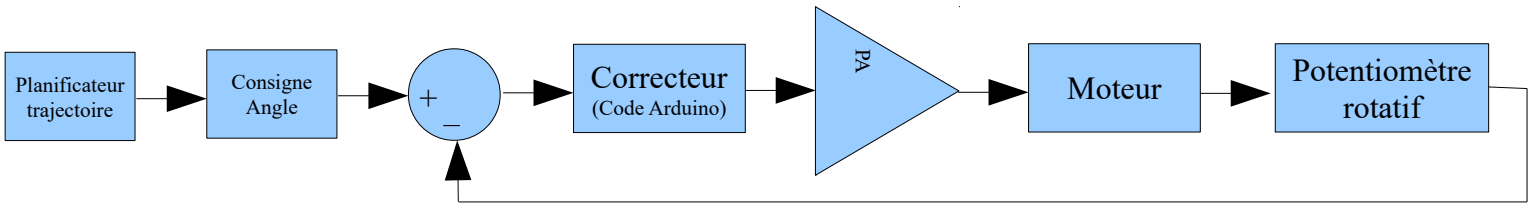


Carte Arduino MKR ZERO
https://store.arduino.cc/arduino-mkrzero

Carte Arduino MKR Motor Carrier
https://store.arduino.cc/mkr-motor-carrier

Moteurs lego power functions
https://shop.lego.com/fr-FR/category/power-functions

Potentiomètre pour mesure d'angle
www.conrad.fr



Servo Geek KittenBot® ; 270 ° Gris avec Fil pour Lego

https://fr.banggood.com/KittenBot-4Pcs-270-Gray-Geek-Servo-with-Wire-for-LegoMi
1325386.html

Algorithme du min – max

Cours /explications
Principe : chaque coup joué possède un coût (une valeur) : par exemple

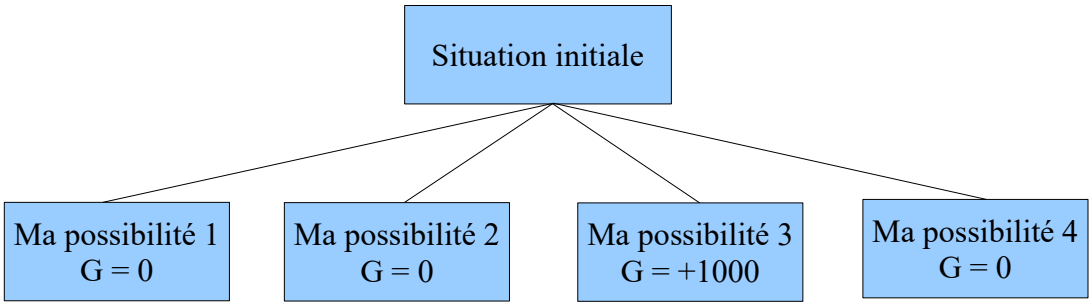
- Morpion : on gagne la partie : +1000 ; on perd la partie : - 1000

- Jeu d'échec : on pert un pion : -10 ; on pert un fou : - 50 ; on pert une tour : -80 ; on pert un cheval : -100 ; on pert une reine : -500 ; on pert le roi (échet et mat) : -1000
A l'inverse : on mange un pion : +10 ; on mange un fou : +50 etc...

A chaque situation, on va explorer toutes les situations possibles et on va affecter à chaque situation un coût. On construit ainsi un arbre de jeu. On va calculer les coût de chaque possibilités selon l'algorithme du min / max : lorsque c'est moi qui joue, je prends le gain maximum et lorsque c'est l'IA qui joue, elle prend le gain minimum (de mon point de vue).

Exemple : jeu de Morpion :

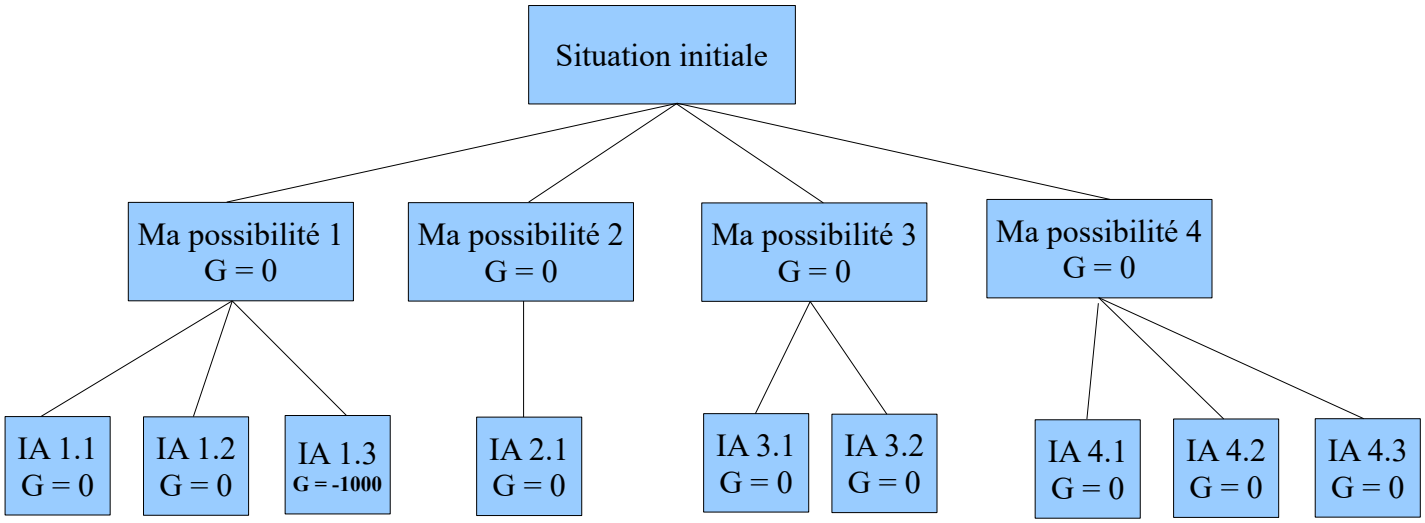
On regarde uniquement à un horizon de 1 coup :

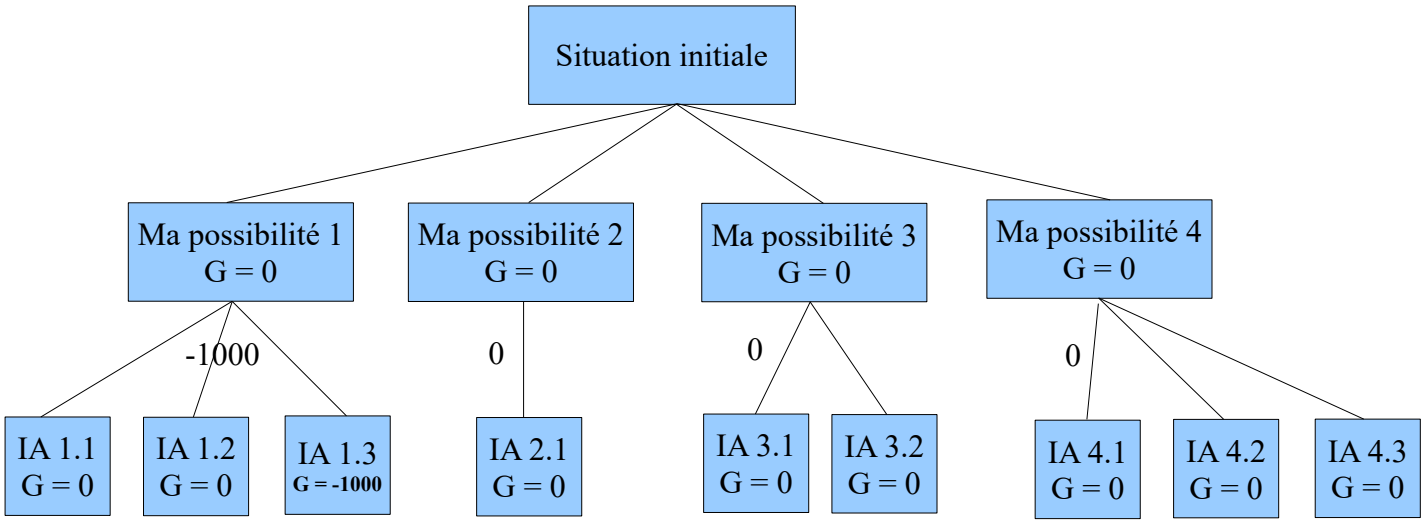


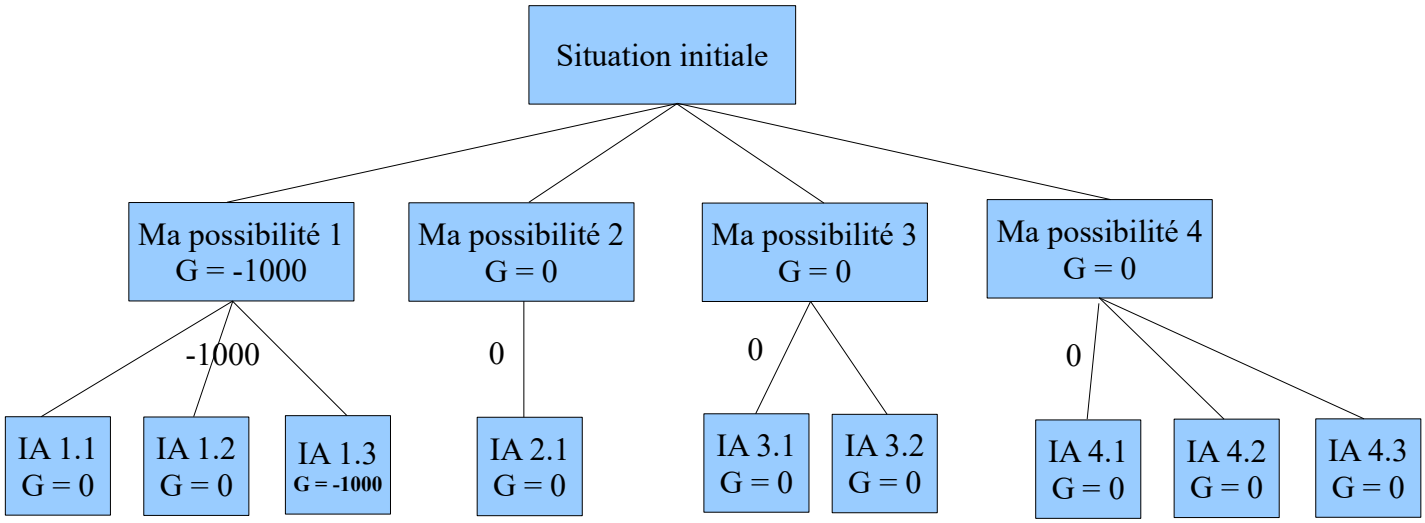
On prend le maximum : on jouera la possibilité numéro 3 car max{0 ; 0 ; 1000 ; 0} = 1000

Exercice 1 : horizon de 2 coups

On regarde à un horizon de 2 coups :

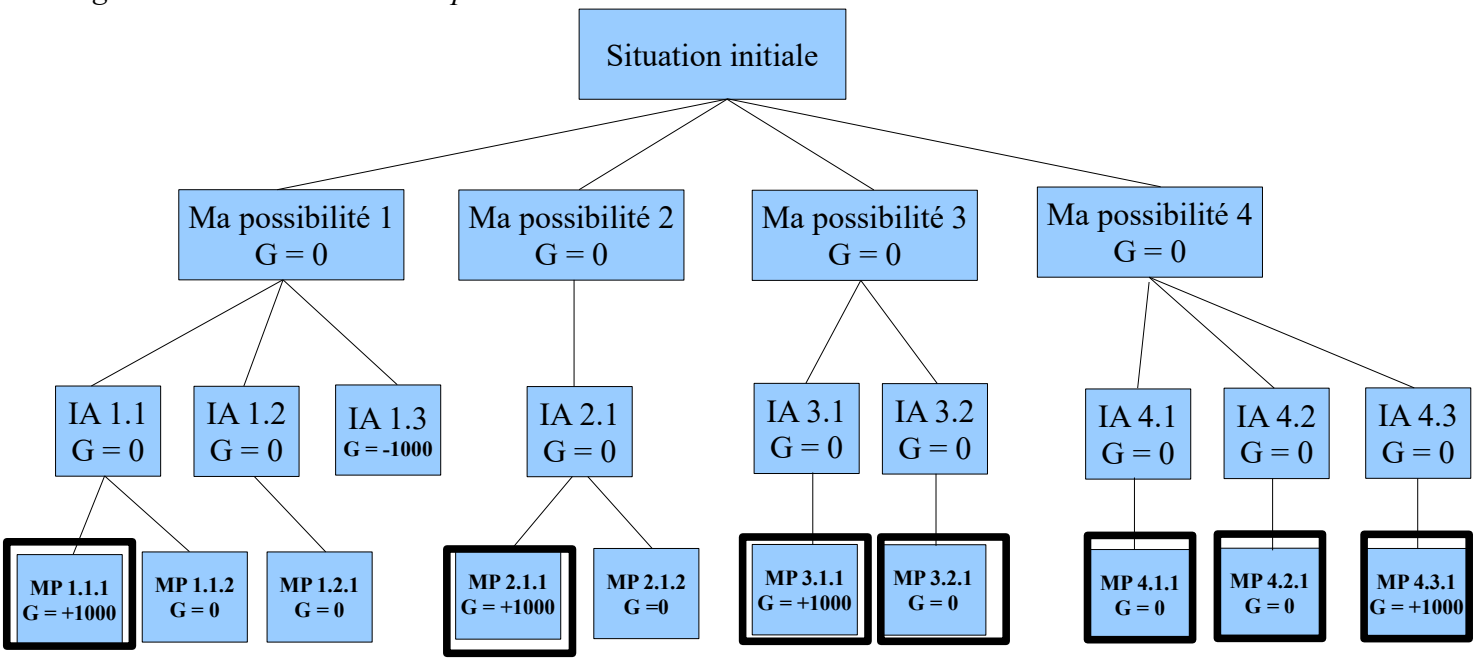


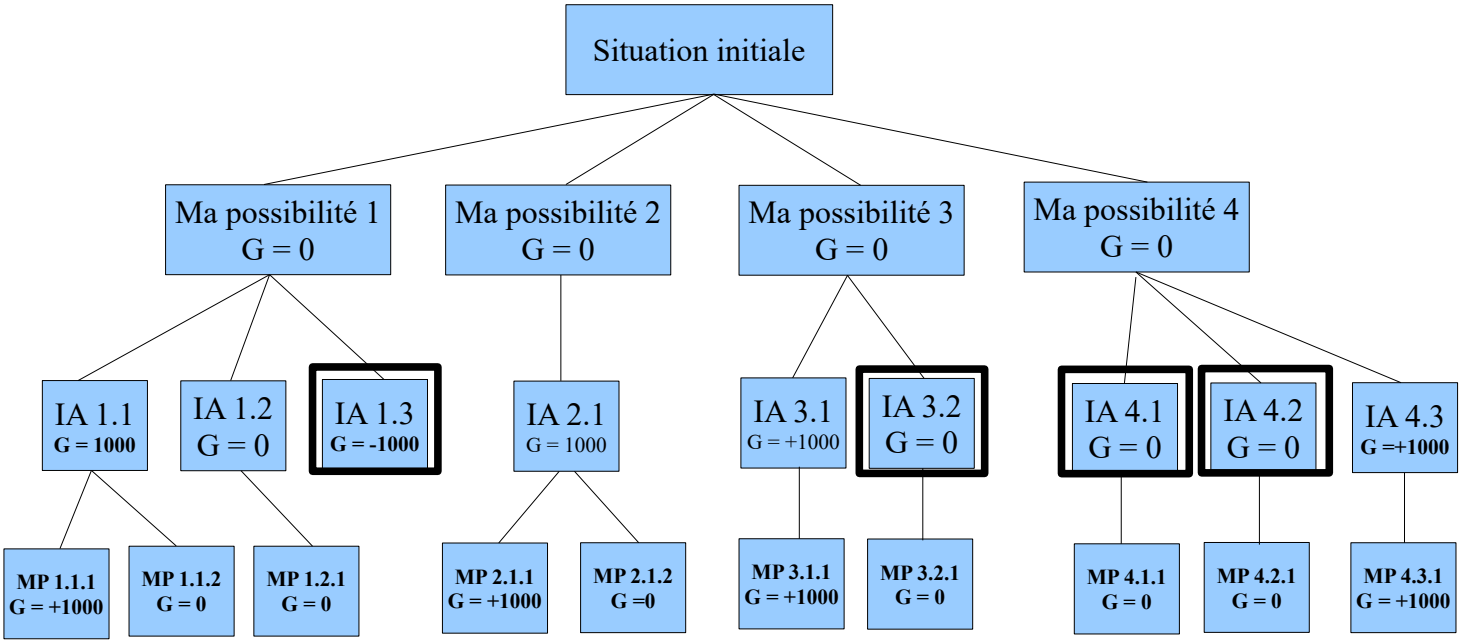


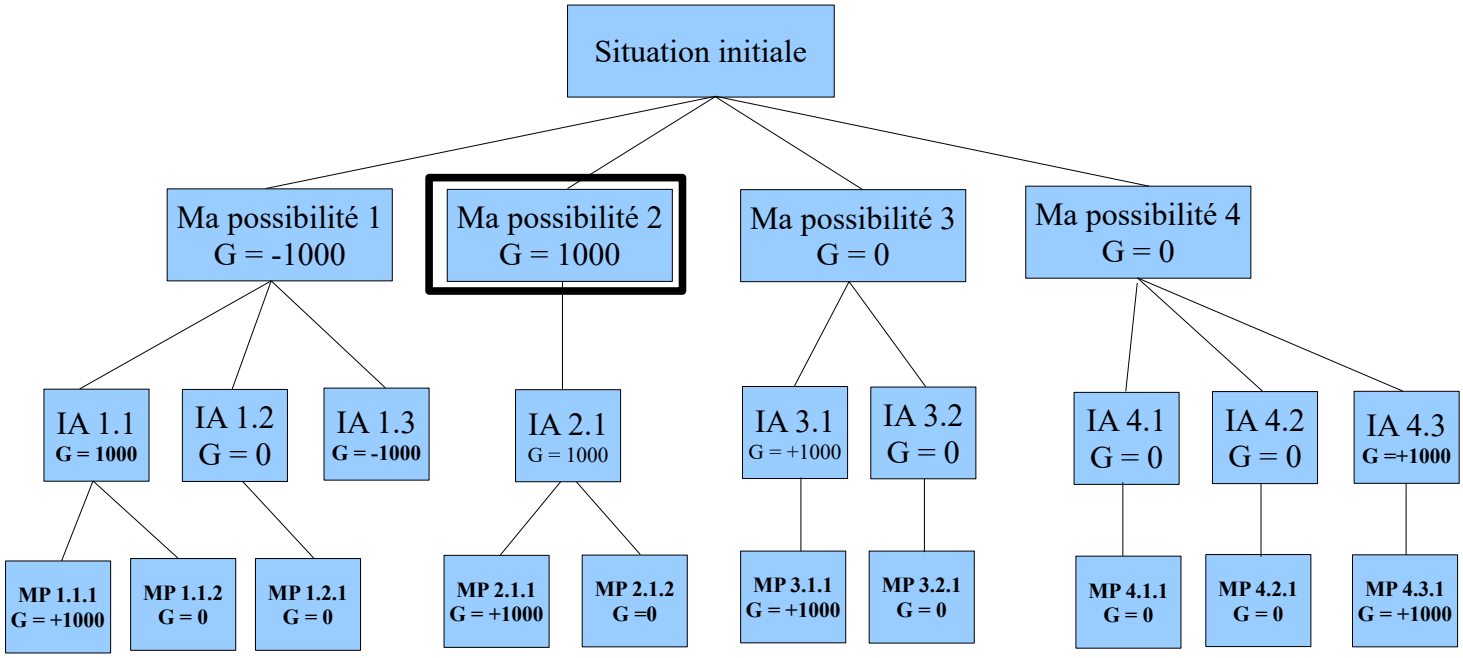


Exercice 3 : horizon de 3 coups

On regarde à un horizon de 3 coups :







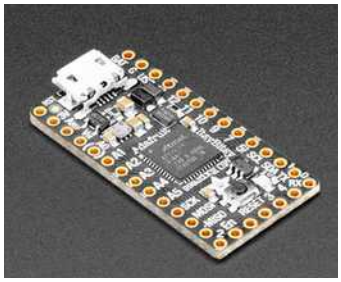
Activité / idée de projet : coder en langage Python un algorithme de jeu automatique du Morpion avec possibilité de régler le niveau dedifficulté (réglage de l'horizon).

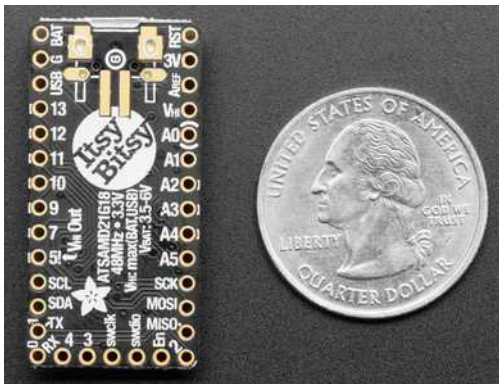
Opérateurs automatiques, bots sur Internet
- jeu de questions / réponses ; arbres et structure alternative
Assistants personnels intelligents ("dit Google"...)

VI / Projets autour du véhicule autonome

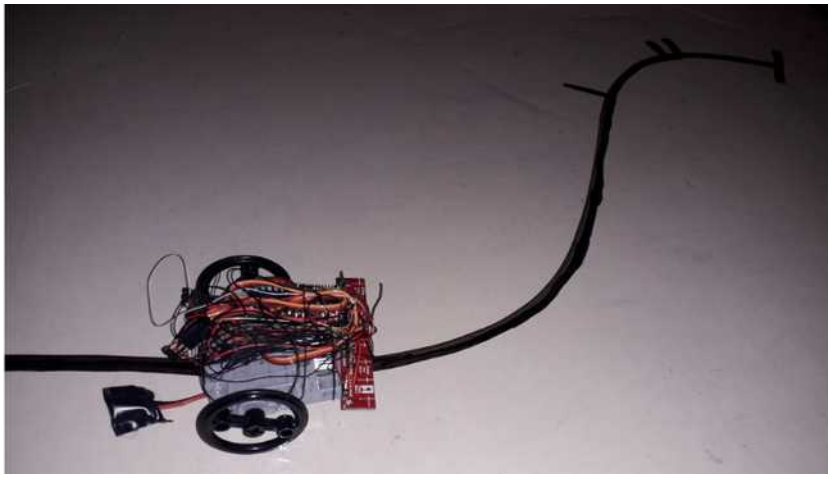
Activités : recherches documentaires de la part des élèves : voiture autonome ; Google Car ; problèmes juridiques

Thèmes abordés
- *suivi de ligne*
- *détection d'obstables*
- *planification de trajectoires*
- *système ACARS*

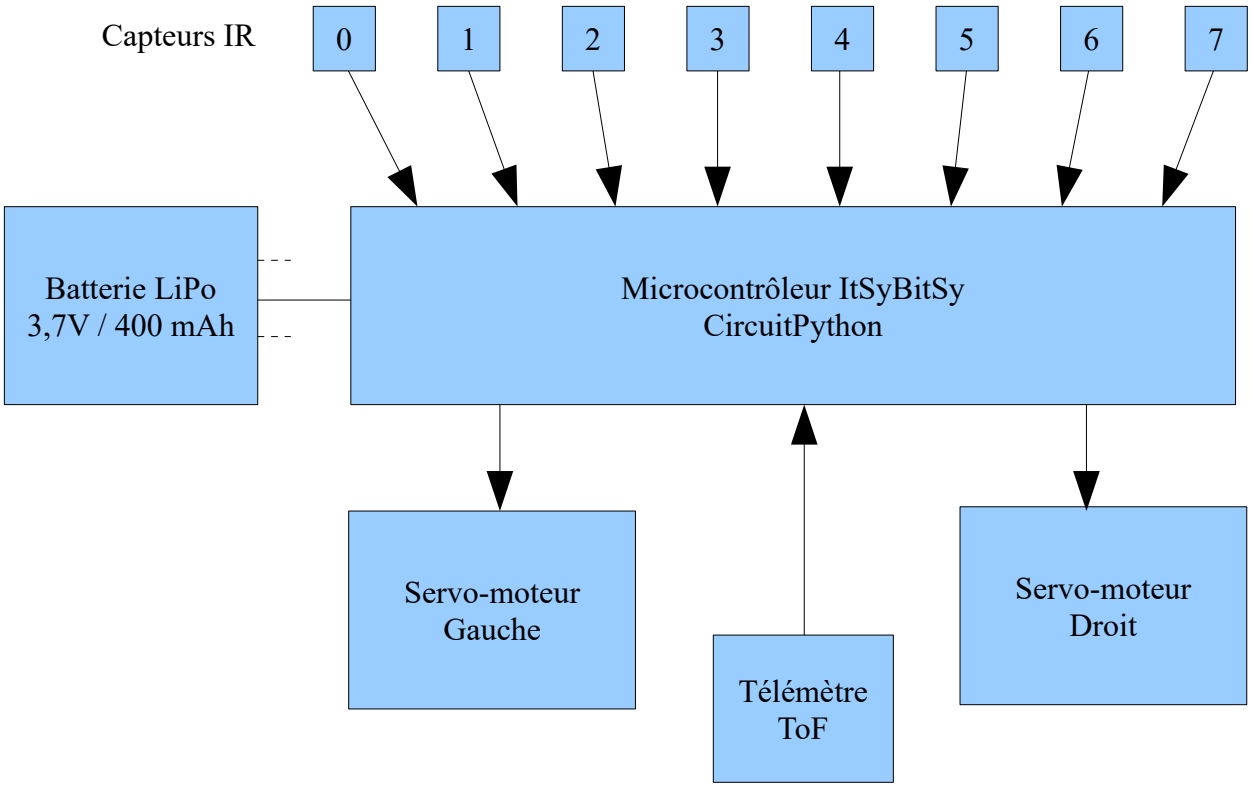




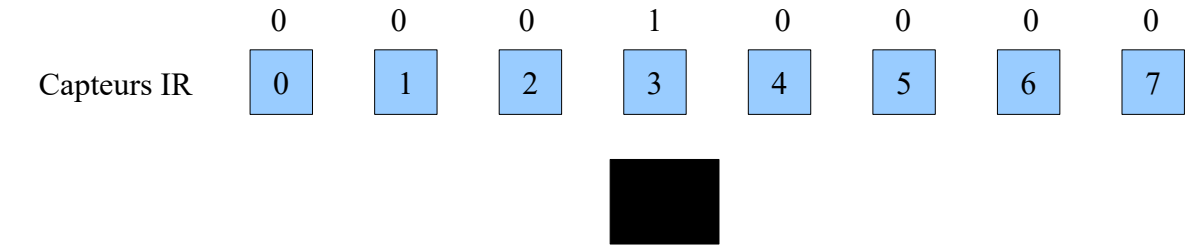
Capteur de distance IR GP2Y0E02B

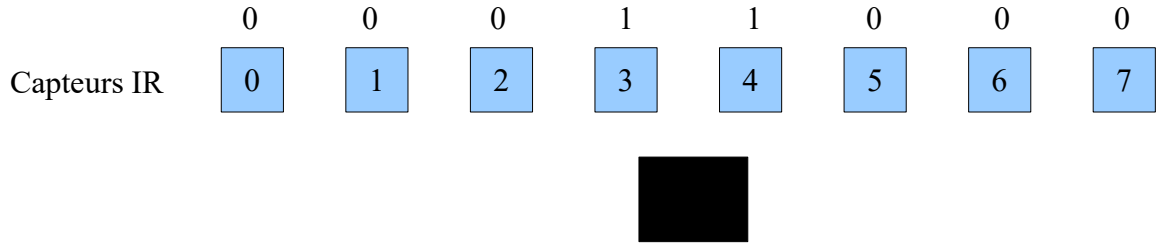


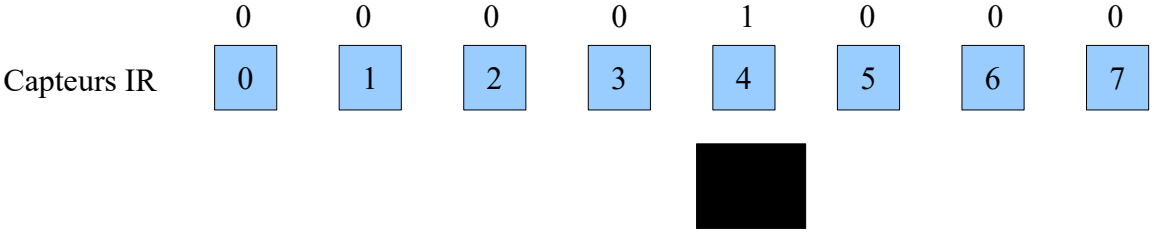
Capteurs IR	0	1	2	3	4	5	6	7



Capteurs utilisés en mode tout-ou-rien





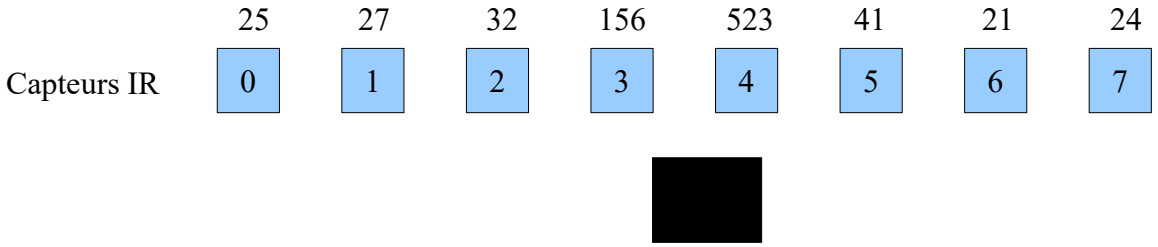



```
if [c0 , c1 , c2 , c3 , c4 , c5 , c6 , c7] == [0 , 0 , 0 , 0 , 1 , 0 , 0 , 0]:
        commandeMoteurGauche(120)
        commandeMoteurDroit(80)
```

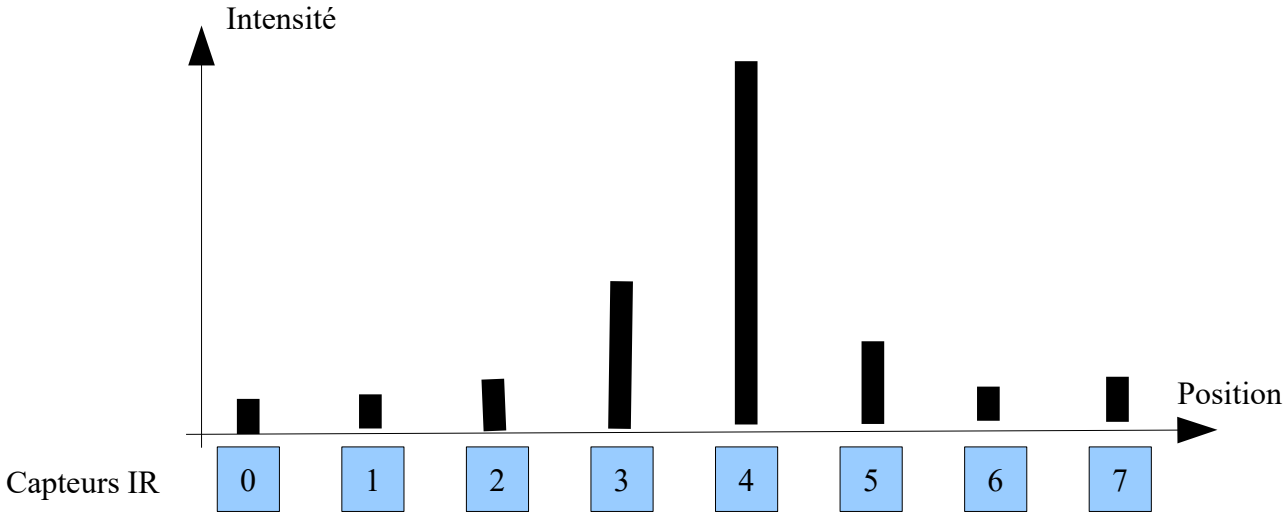
 # 256 cas possibles !!!

Capteurs utilisés en mode analogique (0 – 1023 : 1024 niveaux) : l'objectif est d'estimer la position de la ligne

en utilisant des notions de statistiques et en considérant un indicateur de position (ici la médiane).
Un indicateur de dispersion (écart-type) pourrait permettre d'estimer la largeur de la ligne...

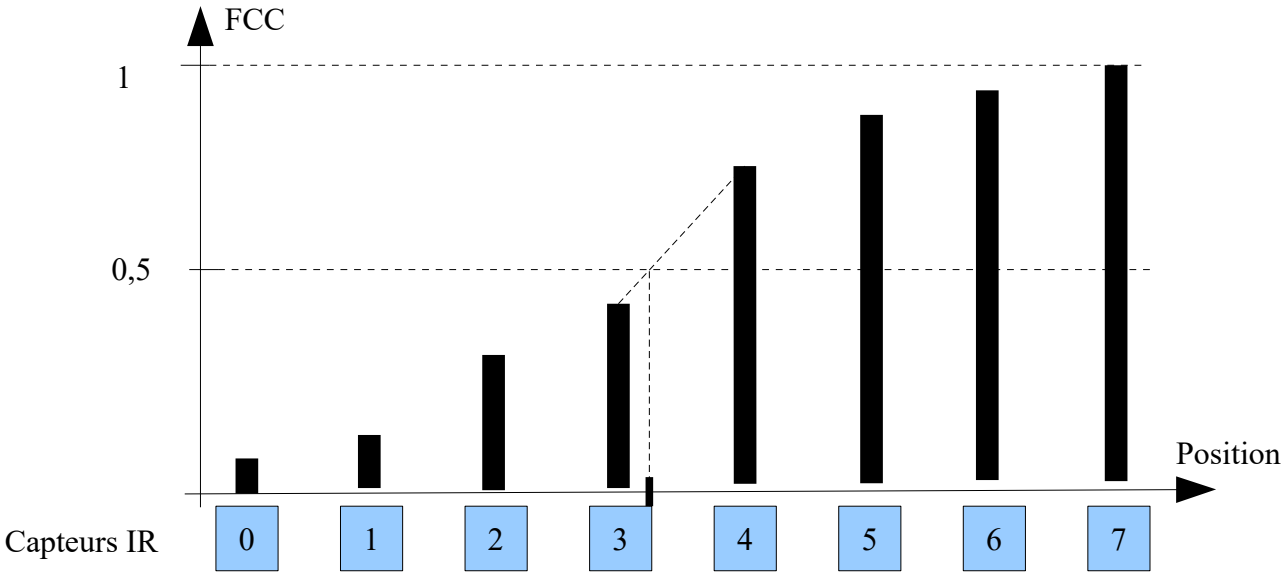


On considère une série statistique



On a une liste de 8 valeurs : [25 , 27 , 32 , 156 , 523 , 41 , 21 , 24]

1°) On calcule les effectifs cumulés croissants
2°) On calcule les fréquences cumulées croissantes
3°) On cherche 2 valeurs de positions qui "encadrent 0,5"
4°) On considère le polygone des FCC et on détermine l'antécédent de 0,5



La

médiane est in indicateur continu.

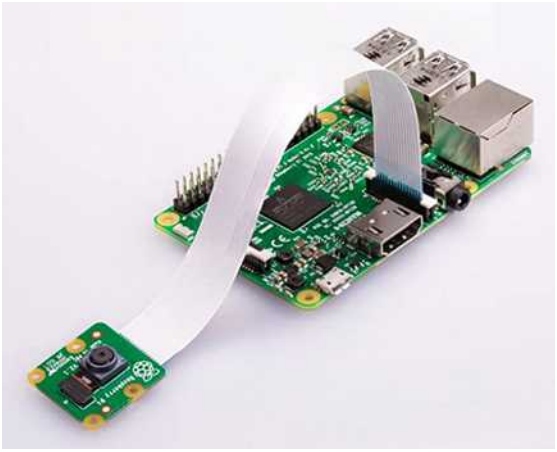
Pas besoin d'étudier les 256 cas.

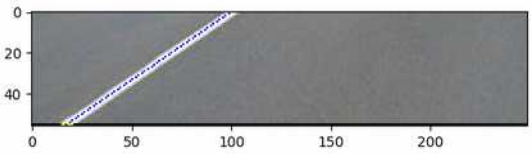
GAIN = 1.5 # dépend des caractéristiques électro-macaniques du robot : à tester expérimentalement

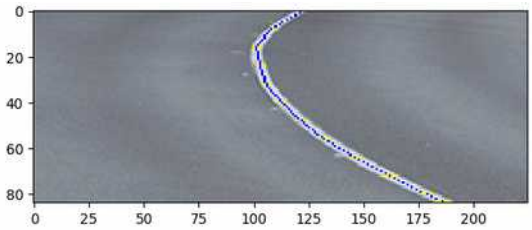
ErreurPosition = mediane – 3.5
commandeDifférentielle = GAIN * ErreurPosition

commandeMoteurGauche(100 + commandeDifférentielle)
commandeMoteurDroit(100 - commandeDifférentielle)

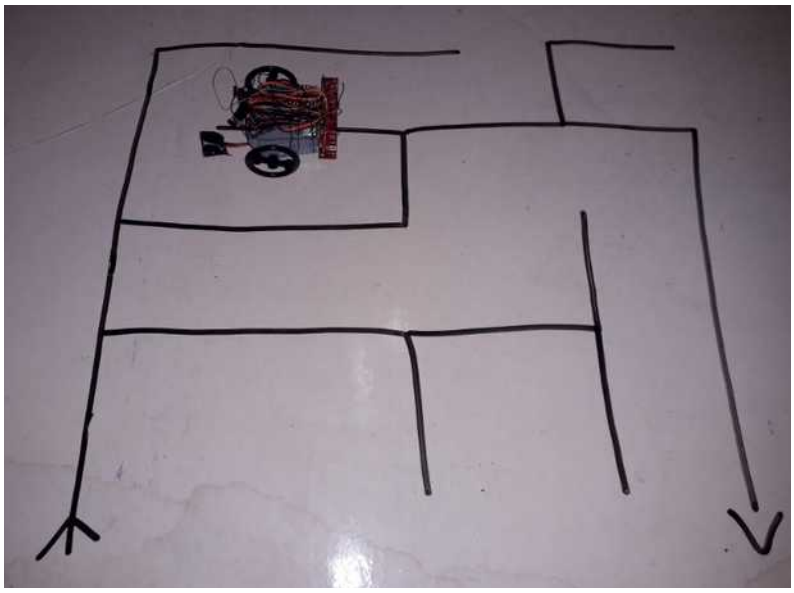
Utilisation d'une caméra pour "lire la route"



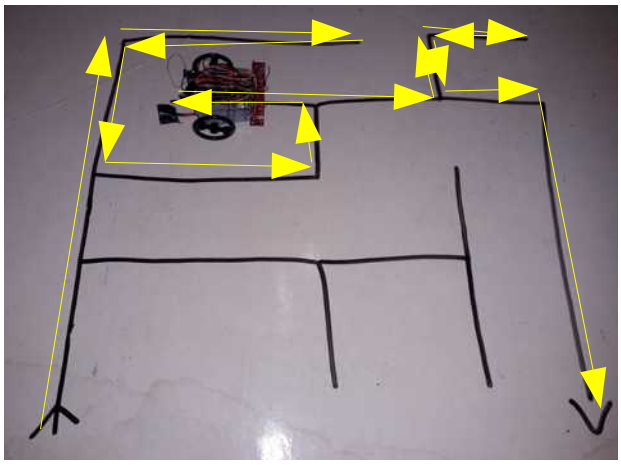


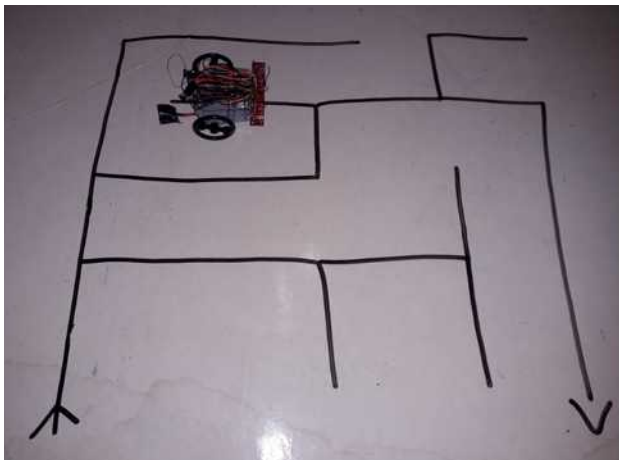


Pour aller plus loin : explorer / sortir d'un labyrinthe

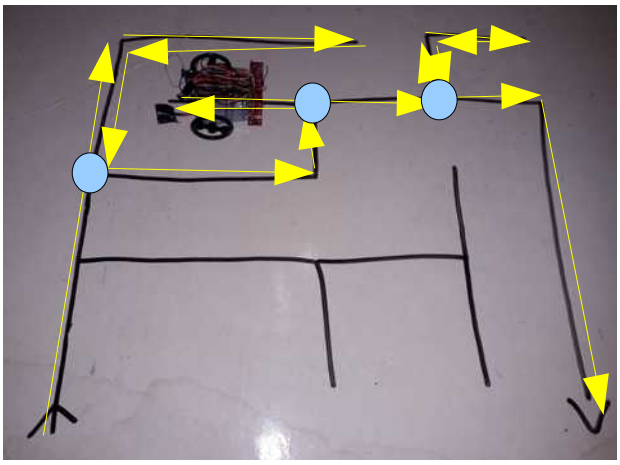


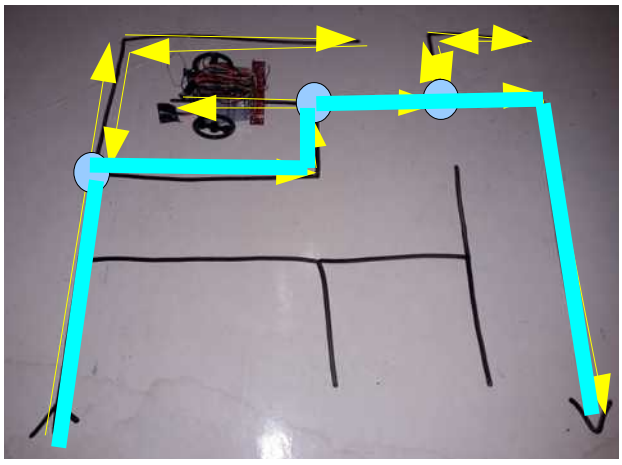
Technique de la main droite ou de la main gauche





Simplifier le chemin : détecter les points de passages multiples.





Problématique : comment détecter les points de passage multiples ?
Repérage du robot dans un plan : odométrie (=> capteurs incrémentaux sur chaque roue), caméra et traitement image ...