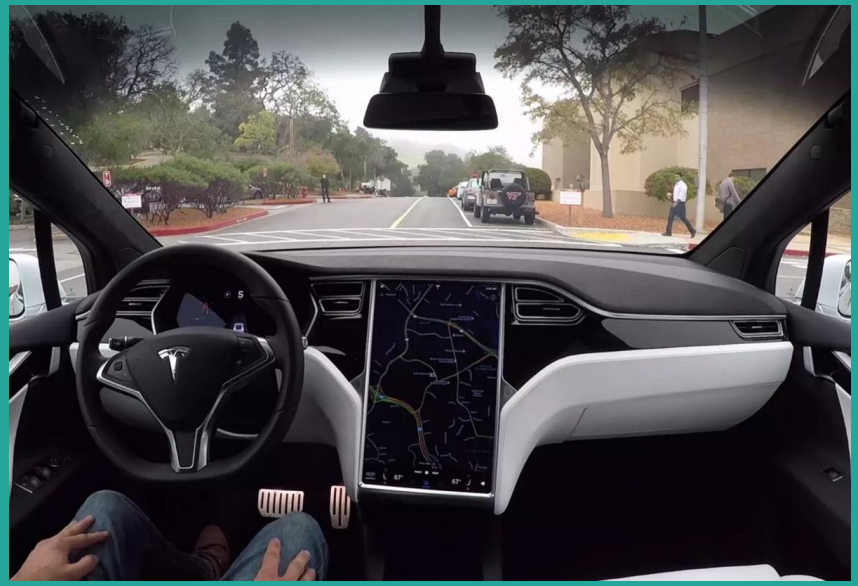
Apprentissage artificiel appliqué au contrôle d'un véhicule autonome

Par OZDEMIR Serdar

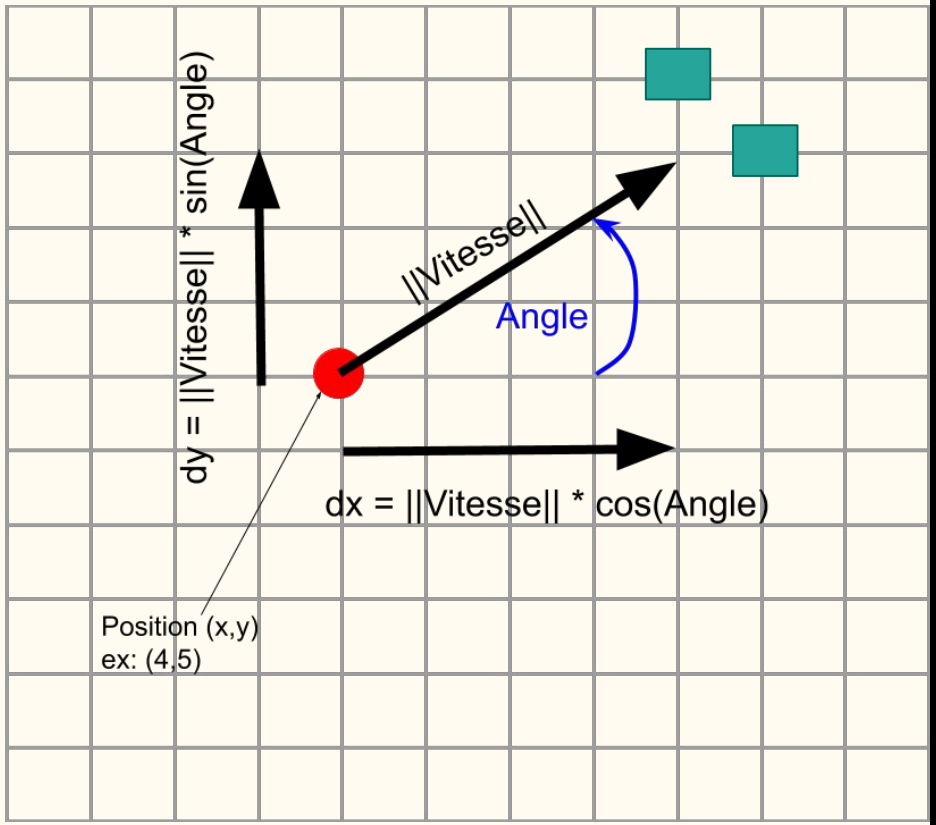
Pourquoi ce sujet?



Source : Tesla.com - Self driving demonstration

Le modèle neuro-évolutionniste

Mes véhicules



Vehicules: Classe python (voir annexe)

Propriétés :

- Position (x,y) dans le circuit
- Vitesse
- Angle
- Réseau de neurones (pour la prise de décision et la modification de l'angle et de la vitesse)

Déplacement :

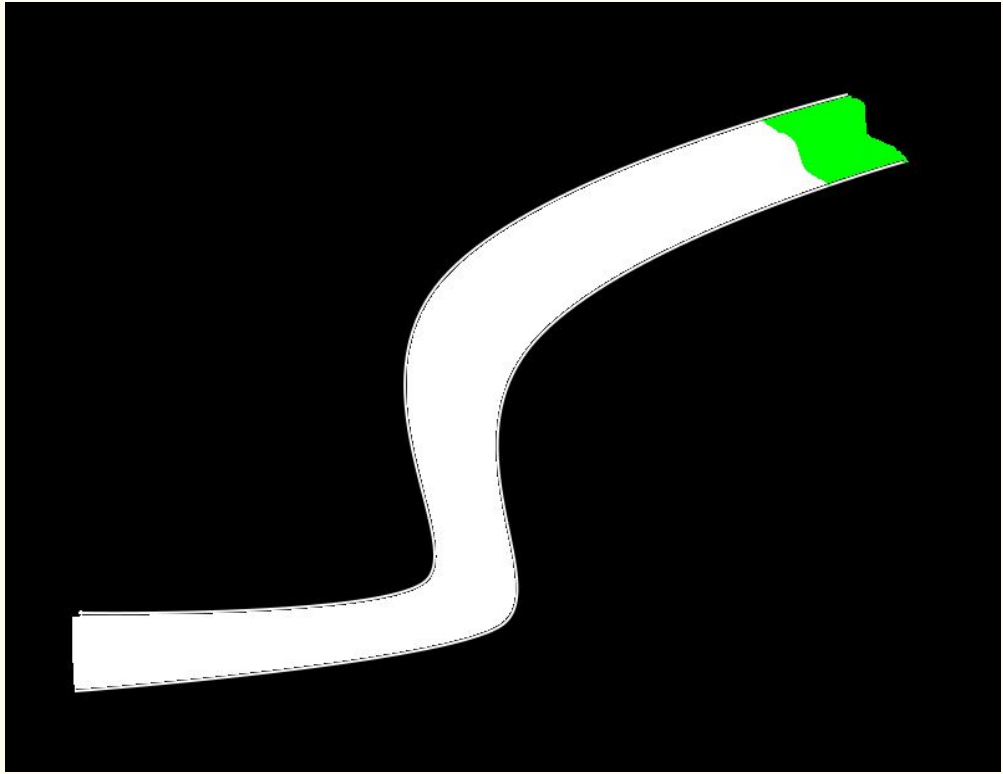
- Calcul de dx, dy
- Si aucun obstacle à la nouvelle position : x,y = x+dx , y+dy
- Sinon le véhicule est mort , ne peut plus se déplacer

Mon circuit

Circuit = Matrice (n * m) depuis une image

Voir generer_circuit (Annexe ligne 13)

- Pixel blanc en (x,y) : Circuit[x][y]=0
- Pixel noir en (x,y): Circuit[x][y]=1
- Pixel vert en (x,y) : Circuit [x][y]=2



Détection d'obstacles

Vision du véhicule: on veut des nombres entre 0 et 1 dans chaque direction

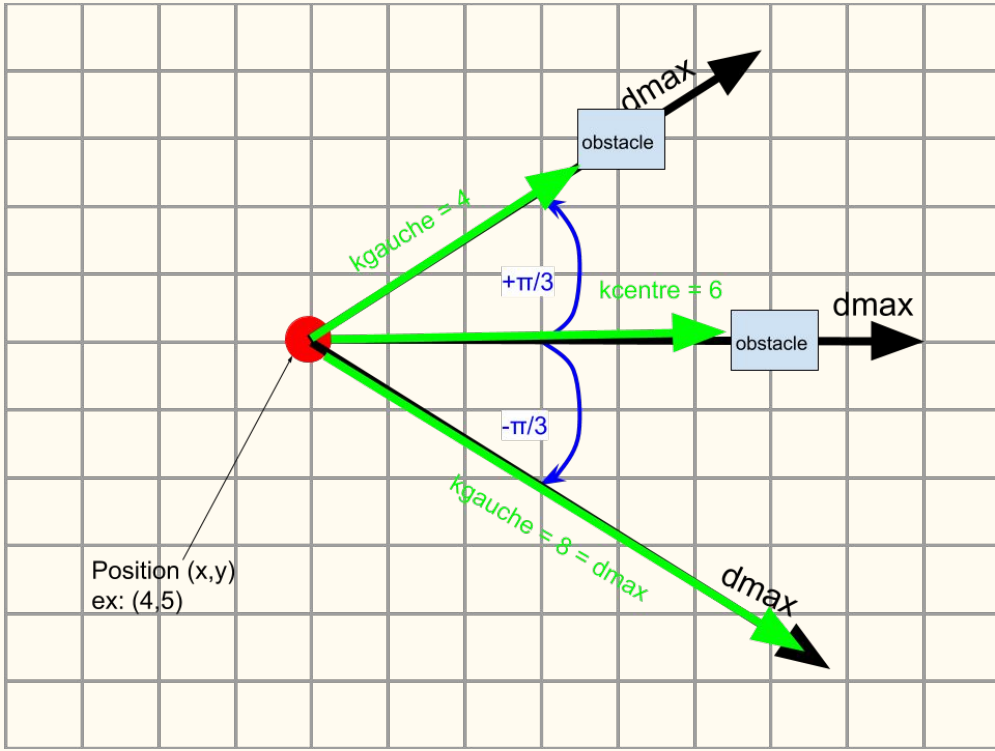
Voir annexe detect_entree() ligne 170

Dans chaque direction , le véhicule incrémente un rayon k jusqu'à Dmax : si il y a un obstacle à la position observée , on retient ce k et on le divise par Dmax

Exemple ici : Obstacle à 4 blocs à gauche , 6 blocs devant , aucun obstacle à droite (donc à dmax=8)

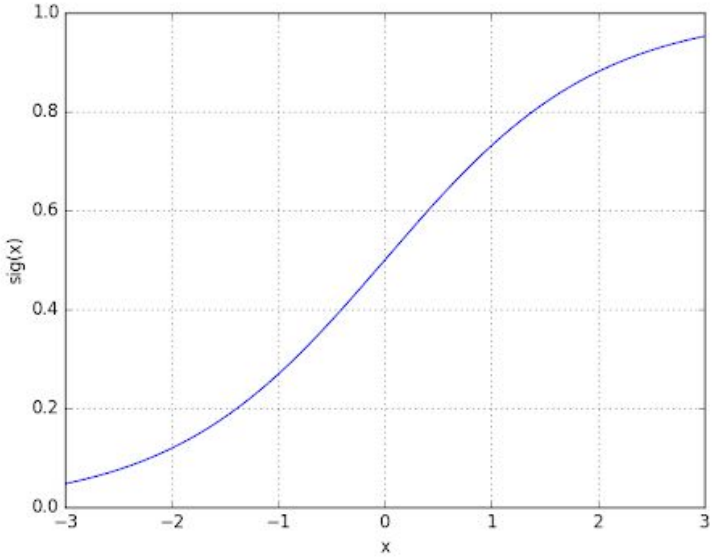
detect_entrée() = [4/8 ; 6/8 ; 8/8]

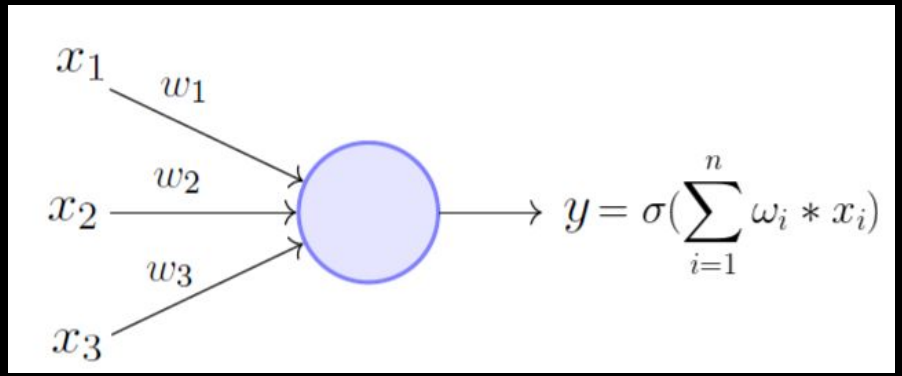
= [0.5 ; 0.75 ; 1.0]



Le réseau de neurones

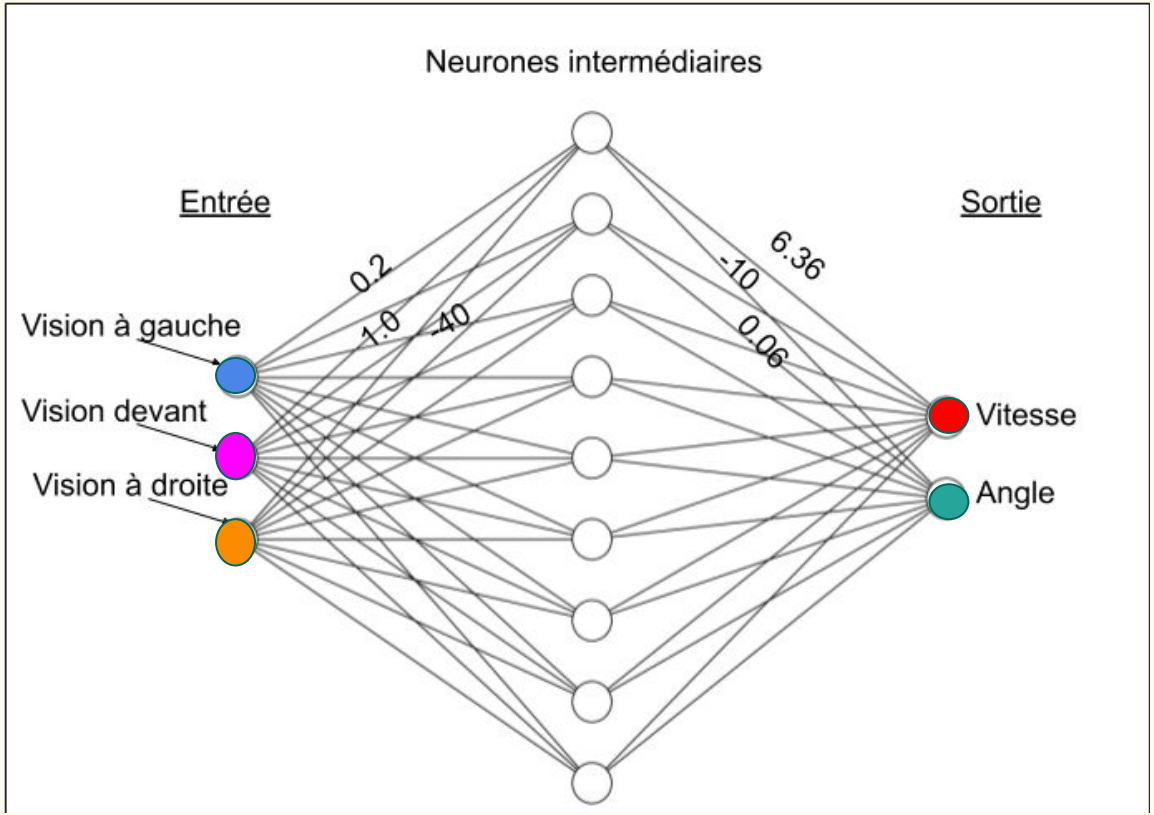
$$\sigma : \mathbb{R} \to [0;1]$$
$$x \mapsto \frac{1}{1+e^{-x}}$$





Un neurone : Une combinaison linéaire pondérée d'entrées dans [0;1]

Mon réseau de neurones



Implémenté comme liste de flottants sous python

Exemple : couche d'entrée:

entrée= [0.5;0.75;1.0]

Sortie: idem entre 0 et 1

Sortie = [0.002;0.4]

Vitesse/vmax Angle/Pi

Et en python?

Classe Neurones ***Voir annexe ligne 48***

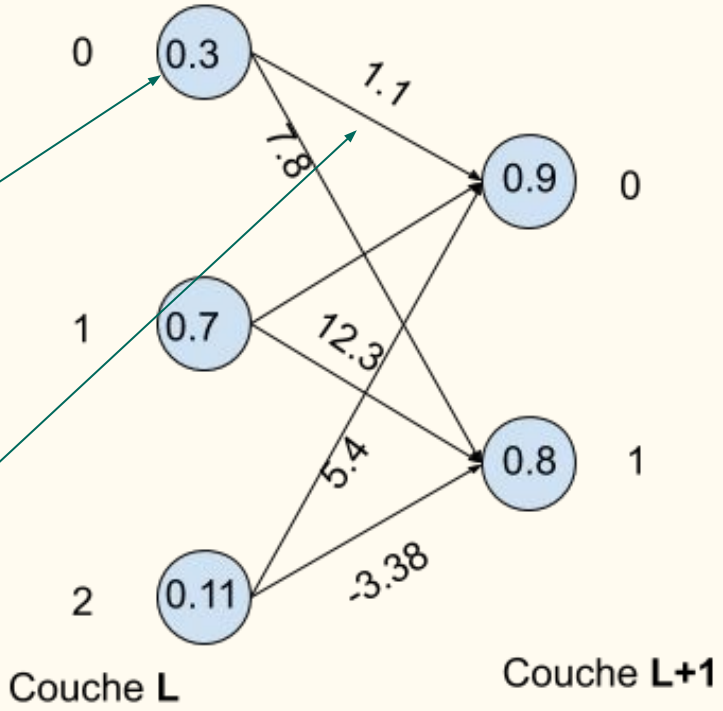
Deux parties :

- Matrice des activations :
  A[L][i]=valeur du ie neurone de la couche L

  Ex: A[L][0]=0.3

- Matrice des poids connexionnels:
  W[L][i][j] = poids de la connexion entre le ième neurone de la couche L et le jème de la couche L+1

  Ex: W[L][0][0]=1.1



Comment calculer une sortie en fonction de l'entrée?

Par récurrence : la combinaison linéaire devient un produit matriciel et on calcule chaque couche d'activations jusqu'à la dernière

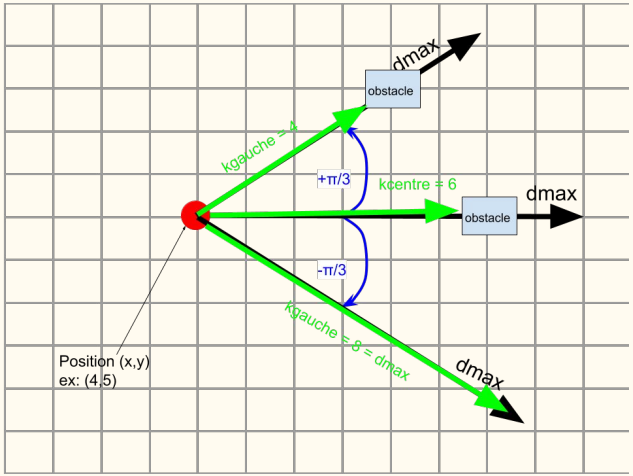
Voir annexe fonction propagation() ligne 67

$$a[i+1] = \sigma\left(\begin{bmatrix} a[i][0] \\ \dots \\ a[i][n] \end{bmatrix} \begin{bmatrix} w[i][0][0] & \dots & w[i][0][m] \\ \dots & \dots & \dots \\ w[i][n][0] & \dots & w[i][n][m] \end{bmatrix}\right)$$

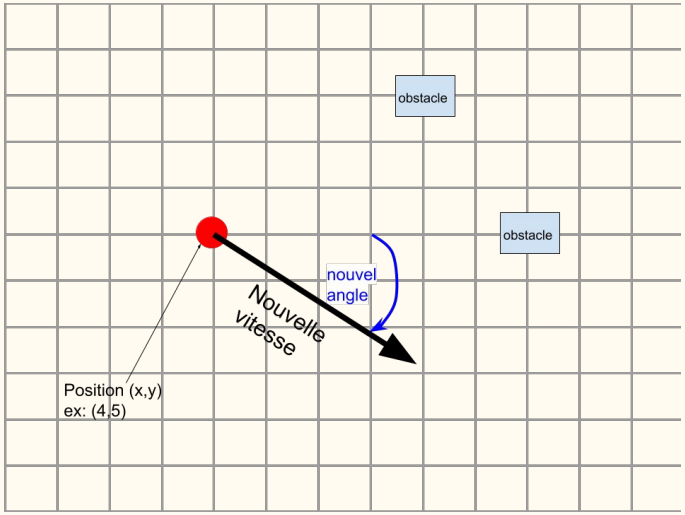
$$\Leftrightarrow$$

$$\mathrm{a[i+1]} = \sigma(a[i] \cdot w[i])$$

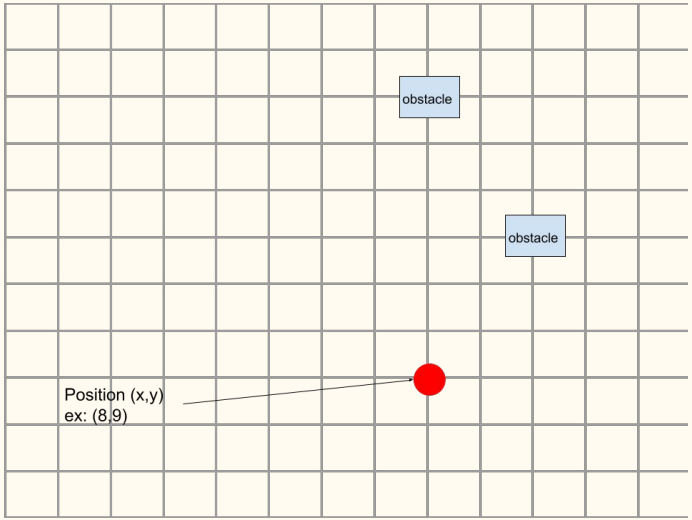
Déroulement de la vie d’un véhicule



Calcul par propagation



Pas d'obstacle
Actualiser la position



Observer les alentours

Neuro-evolution : régler un réseau de neurones par algorithme génétique

L’évolution

Idée :

- **générer un grand nombre de véhicules , les laisser se déplacer jusqu’à leur mort**
- **sélectionner les véhicules ayant parcouru le plus de distance comme parents pour la prochaine génération.**
- **Faire muter quelques parents au hasard**
- **Créer les individus de la génération suivante par reproduction entre ces parents**

Et recommencer à chaque génération

Voir annexe evolution() l.117

Taux de mutation: % de chance qu’un parent au hasard subisse une mutation
ex:0.2 (20%)

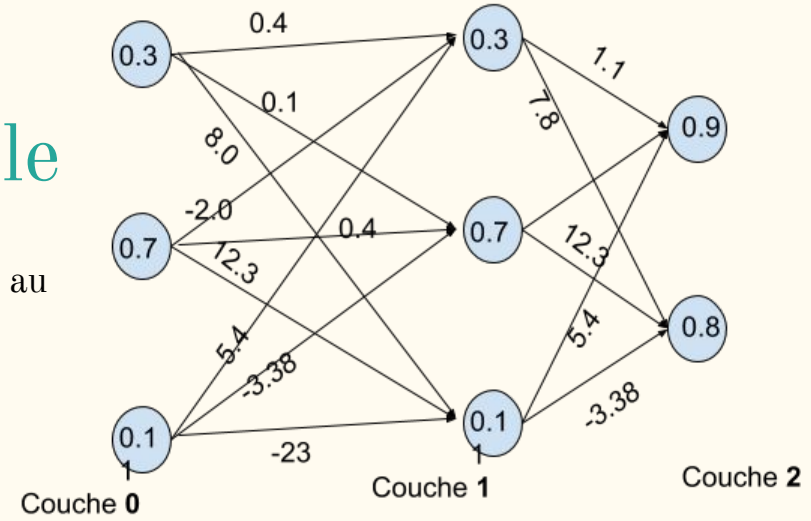
Taux de rétention: % des meilleurs individus conservés comme parents
ex: 0.6 (60%)

Taux de sauvetage: % de chances qu’un individu (même mauvais) soit tout de meme retenu pour etre parent
ex: 0.05 (5%)

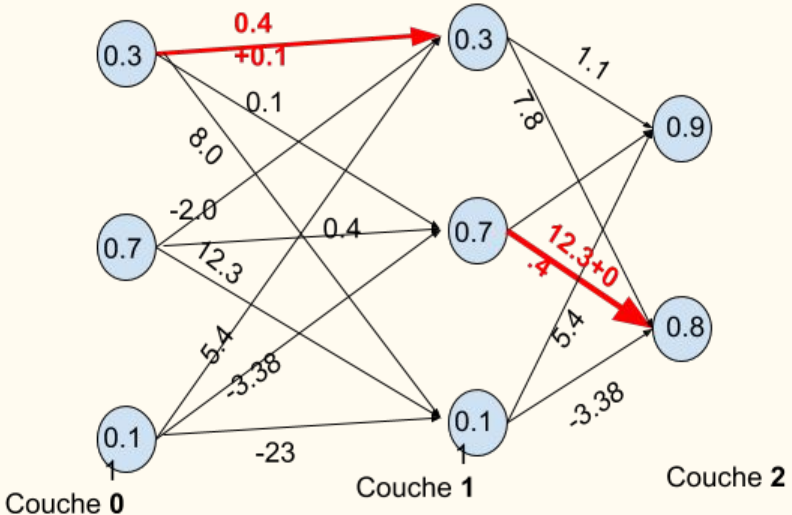
Mutation d'un véhicule

Principe simple : ajout d'un nombre aléatoire au poids d'une connexion au hasard entre deux neurones

Voir annexe mutation() l.81



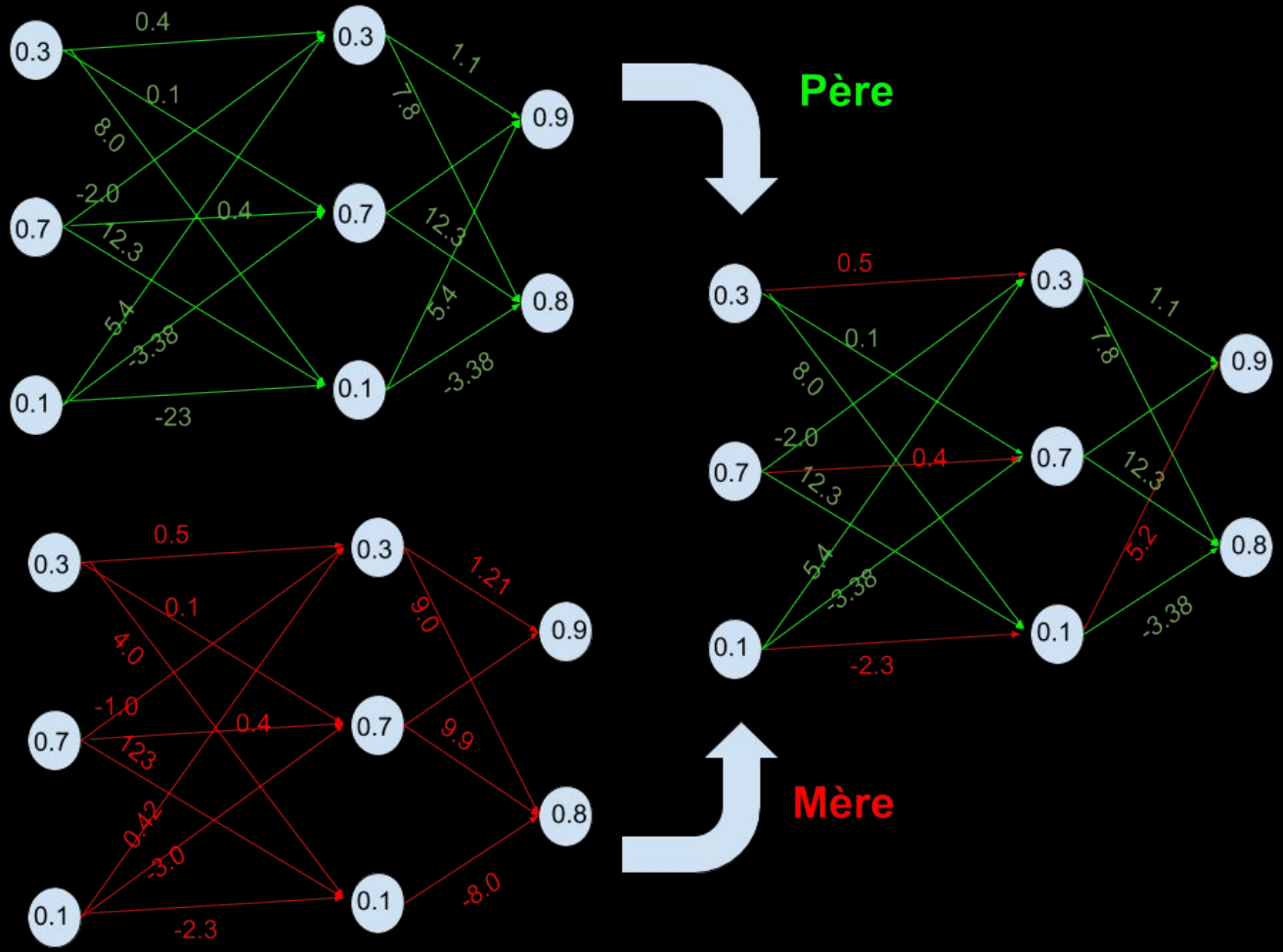
Mutation



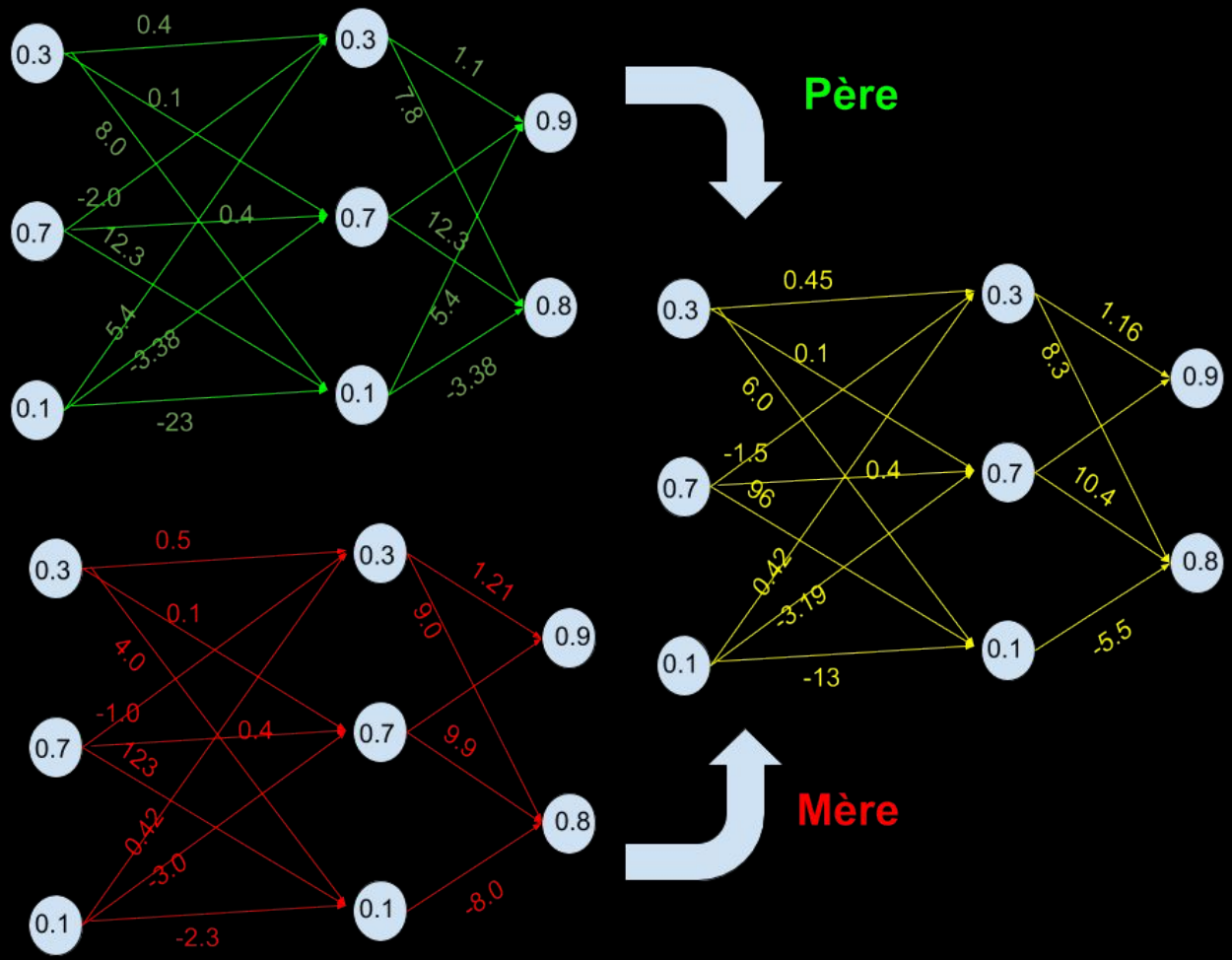
Reproduction des véhicules

Deux opérateurs de reproduction
Le premier : **Reproduction par copie**

Voir annexe reproduction() l.98



Deux opérateurs de reproduction
Le deuxieme :
Reproduction Barycentrique
Les poids de l'enfant sont la moyenne des poids des deux parents
Voir annexe reproduction() l.98

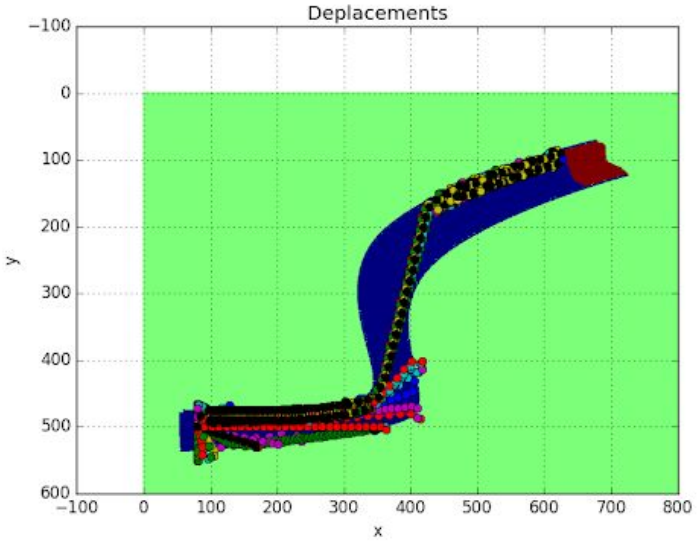


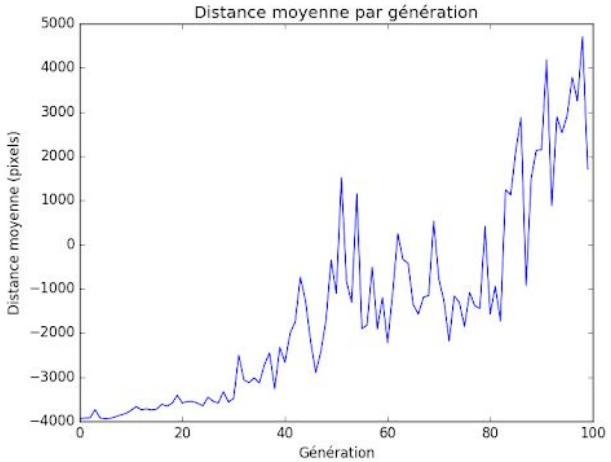
Les résultats

Taux de mutation: 0.2
Taux de rétention: 0.6
Taux de sauvetage: 0.05

Population : 70 Véhicules
100 générations

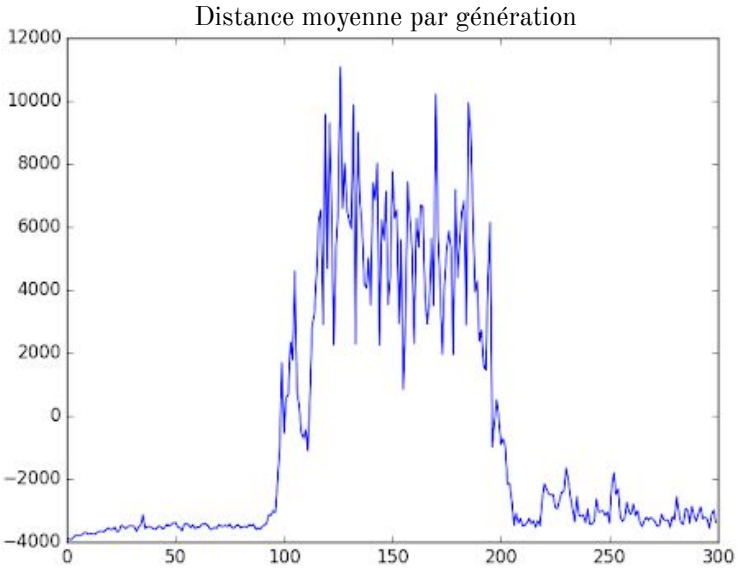
Constante de normalisation à 10
(aspect important)

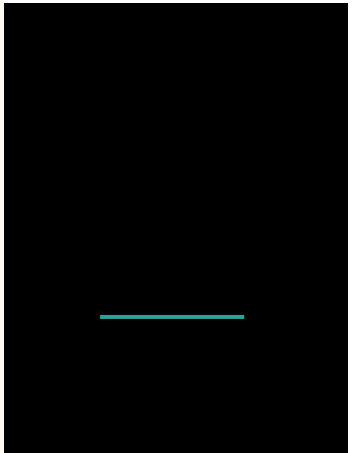


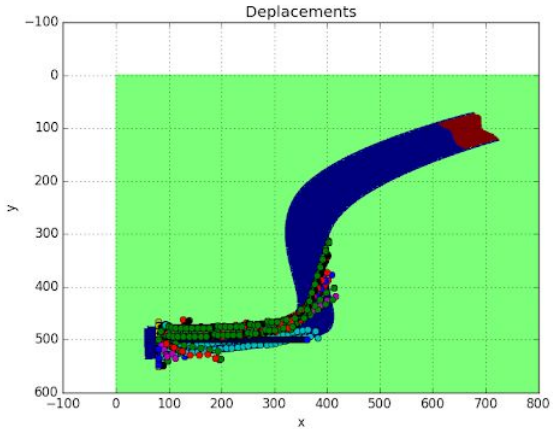


Discussion sur les mutations

Important de faire décroitre le nombre aléatoire ajouté aux poids pendant la mutation: sinon perte de solution

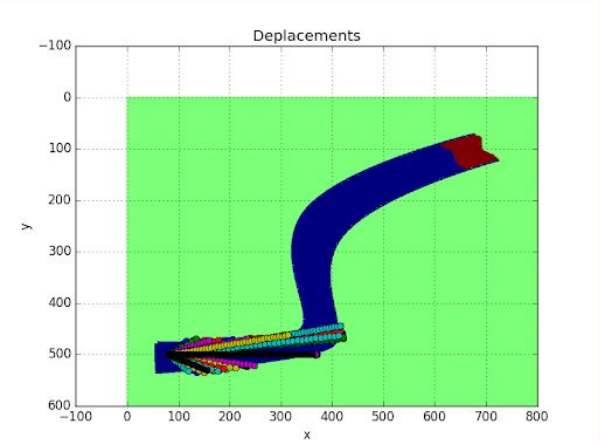


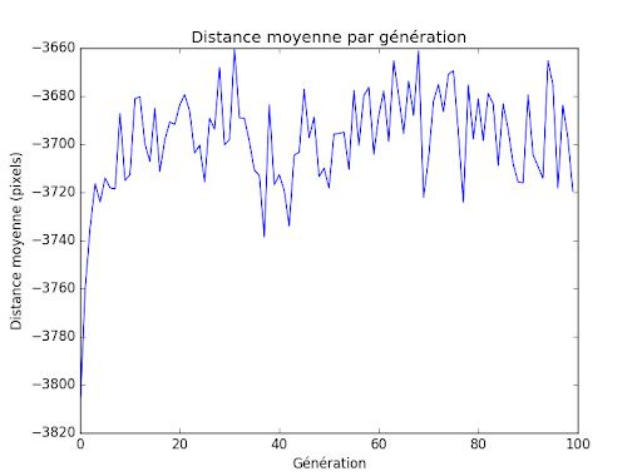




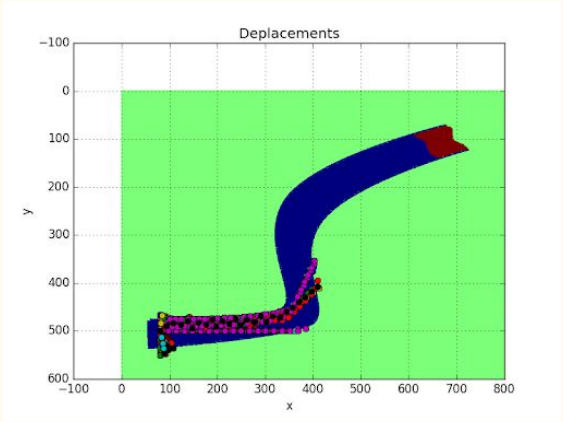
Importance du facteur de normalisation

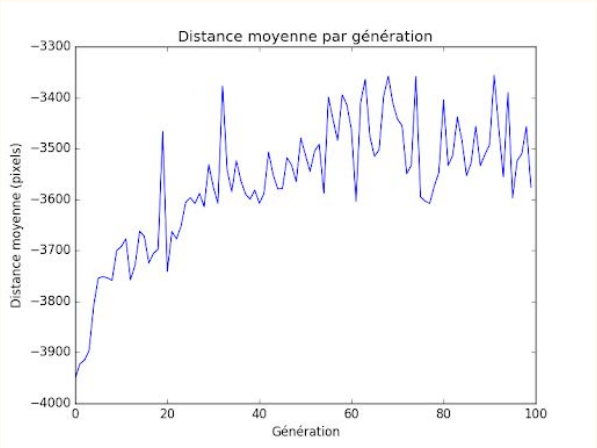
1-Normalisation



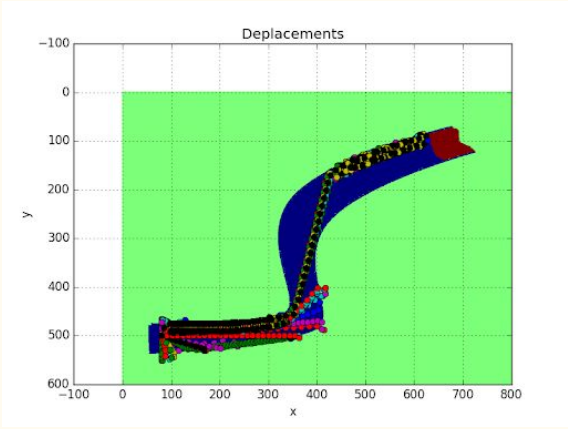


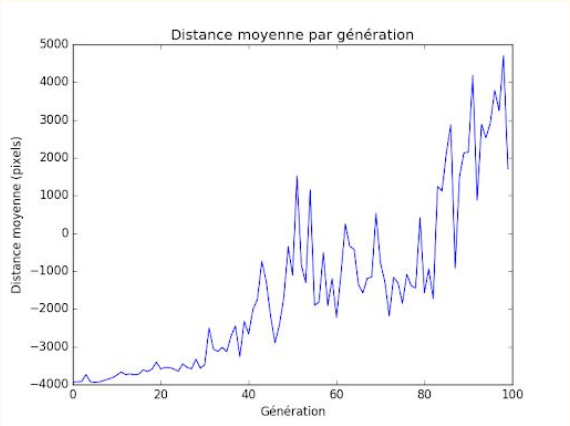
7-Normalisation





10-Normalisation

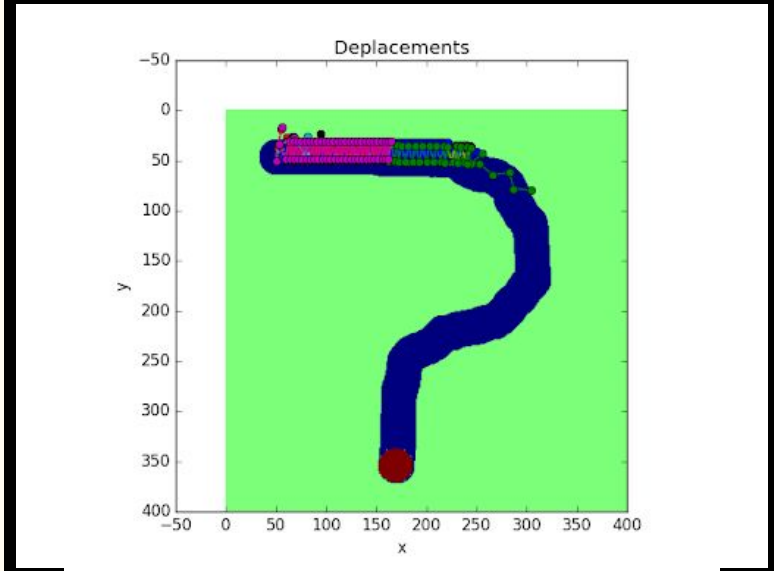


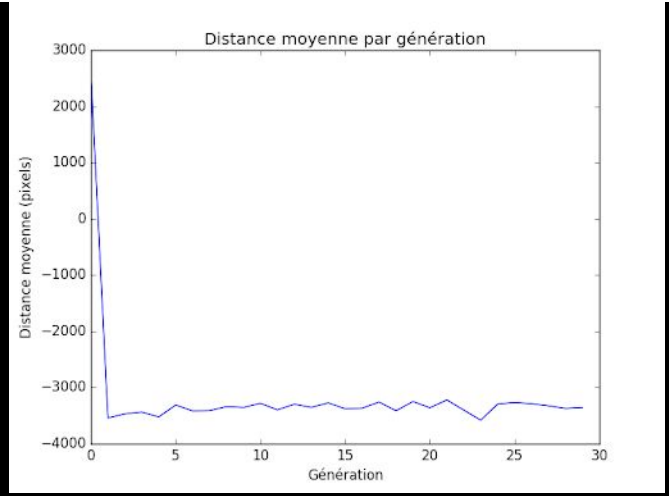


Problème: aspect apprentissage

Adaptation à un nouveau milieu peu fructueuse

Voir annexe sauvegarde_generation() & lecture_generation() l.317





Conclusion