

18/09/2019 email: jjmariage@free.fr.

Machine Learning.

- Réseaux de neurones

Fabriquer les données

Pré neuronale → Données numériques.

ex: Reconnaissance vocale.

UCI Data Rep (BD)

irisdata.

François
Skupra
V.O

Y. Ben Ami
V.F

23/01/2018

R. Lippmann \rightarrow N.N...
V.O
à écrire (Espresso)

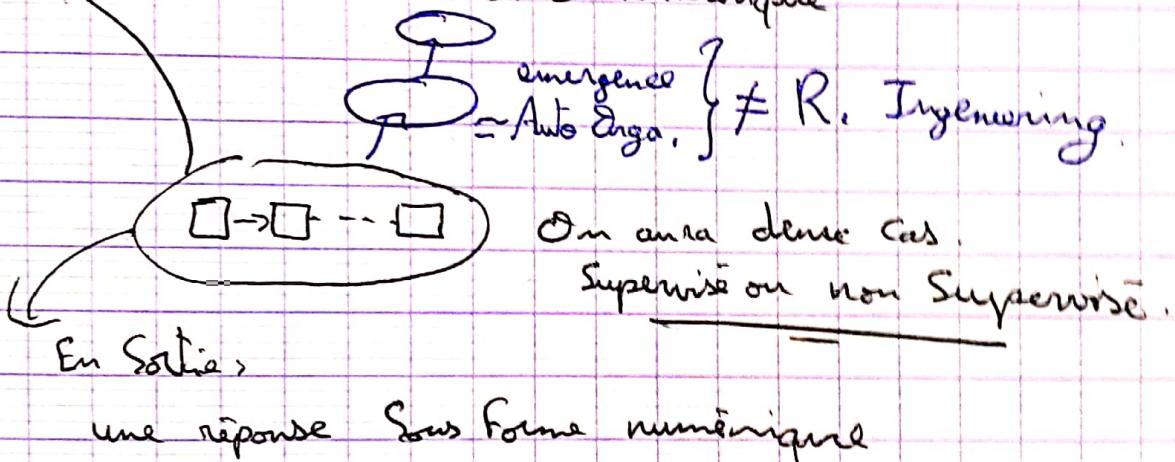
S. Papert

F. Rosen Blatt
MLP

I.A
Neurone

En Entrée :

un vecteur de donnée numérique



En Sortie :

une réponse sous forme numérique

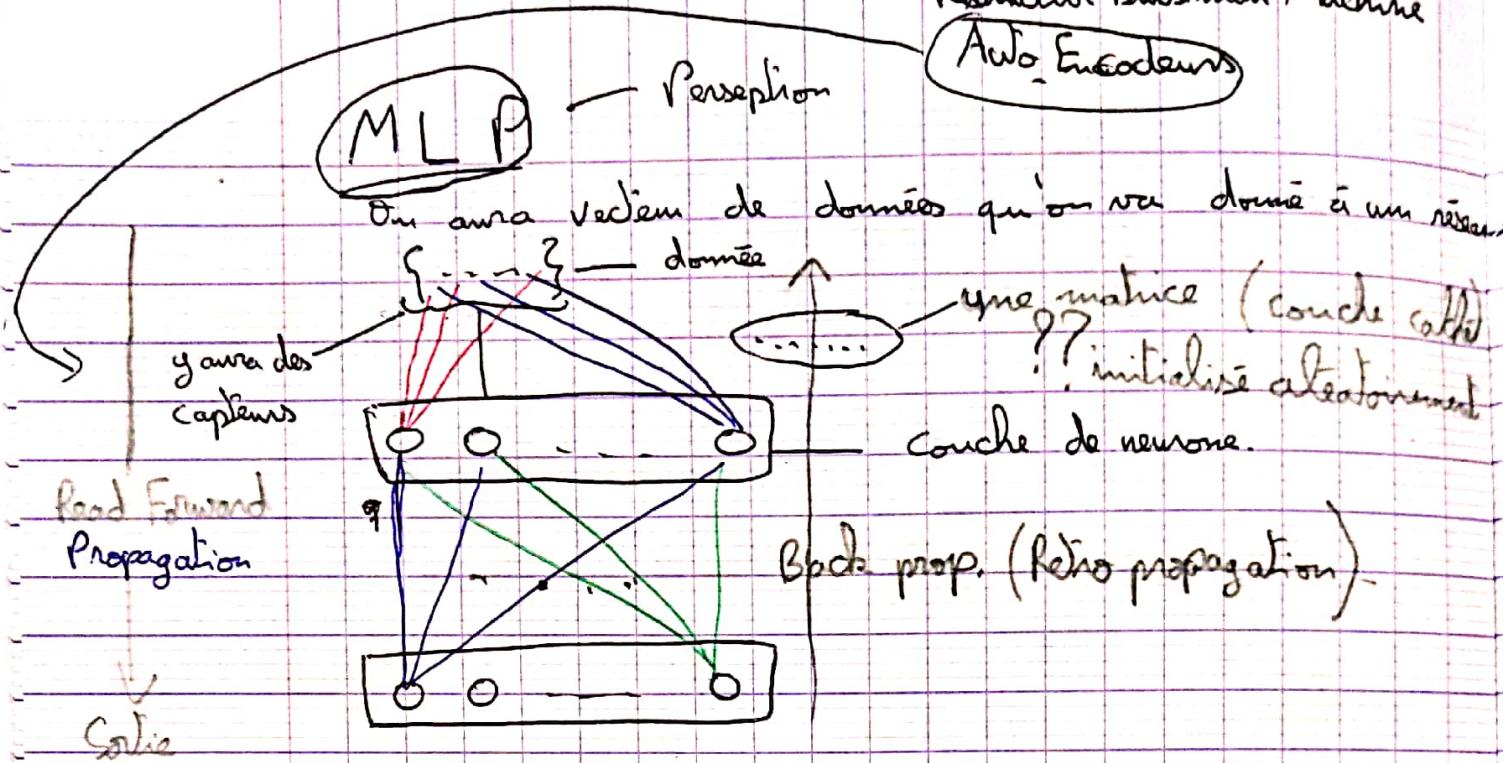
Elémentaire = Réseau de neurone + Connexion + Règle d'apprentissage.

25/01/2018

Généralisation

→ On ne lui donne pas tout, Ex: Image Sans lunette
(Il devrait reconnaître l'image avec lunette.)

Restrictive Boltzmann Machine



La mémoire de notre réseau de neurone est réduite par rapport ~~est~~ à notre espace de donnée. Sous forme d'une représentation.

Locale

- locale à chaque unité
- un vecteur de mémoire pour chaque unité.
- Ex: \rightarrow Produit Vect, Lin.
En utilisant Euclid.

Distribuée

Stocke les données de manière aléatoire dans notre réseau.
En (ça devient) difficile à retrouver les données initiales.

- Ici, on peut retrouver plus facilement les données initiales

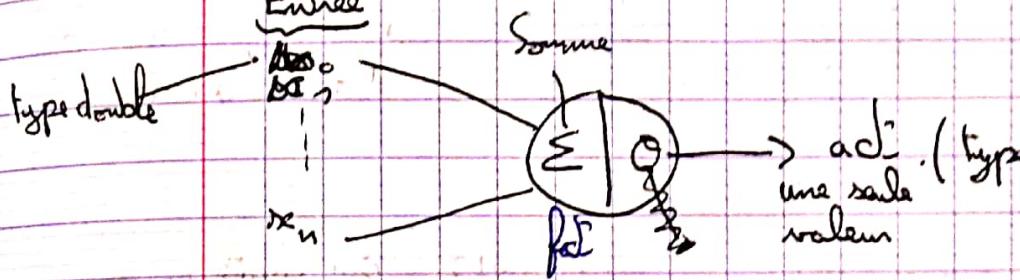
30/01/2018

Iris Data
UCI data Rep

2

~~Structure d'un~~
 de neurone.

~~Entrée~~
 type double



→ Le vecteur de mémoire est à la même taille que le vecteur de donnée.

Vecteur de mémoire

$$W \left\{ \begin{matrix} w_0, w_1, w_2, w_3, w_4, w_5 \\ \dots \dots \dots \dots \end{matrix} \right\}$$

$$\begin{array}{|c|} \hline \begin{matrix} w_0 = D_0 \times V_0 \\ w_1 = D_1 \times V_0 \\ w_2 = D_2 \times V_0 \\ \vdots \\ w_n = D_n \times V_0 \end{matrix} & \begin{matrix} v_0 \\ v_1 \\ \dots \\ v_s \end{matrix} \cdot \quad \quad \quad 0 \quad \rightarrow \text{produit des vecteurs par les matrices} \\ \hline \end{array}$$

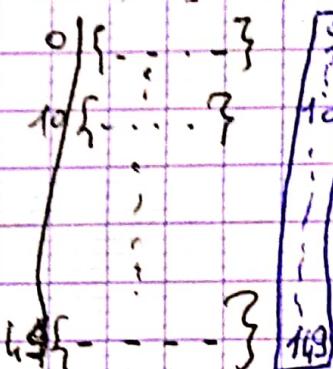
Initialisation de notre réseau:

Calculation du vecteur W
(vecteur de mémoire).

- Calcul de la moyenne des vecteurs.
- On fera une intégration entre des valeurs qui s'approchent.
- On tire une valeur aléatoire qui s'approche vraiment.

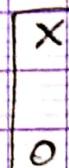
Alg. du Shuffle.

Iris Data



On initialise un tableau avec allocation dynamique qui contient des indices par rapport à la taille de notre "Iris Data".

Shuffle quand y a pas une de succession de données d'évenements



Si la valeur générée est >, on permute entre la première valeur et X. On décale de une case pour ne plus toucher à X.

Struct → renvoie une chaîne de char
char fin Struct

elle signale dès qu'il y a le "char fin de ligne"
ou "char fin de fichier".

160

Struct

~~char~~ NAN

string tabtable

Note

Number

Iris Data

UCI data Rep

→ On initialise notre tableau après avoir parcouru tout le fichier.

Struct Vec { → Vecteur de données

double * V;

char * Name;

double norme; // La somme carré de tout les des de

}

① On normalise les vecteurs avant de les utiliser.

② // Normalisation:

→ diviser chaque valeur du vecteur par la norme.

(utilisé =

→ permet

Avec les femmes ^{l'âge} la vie fait terrible mais sans les femmes la vie fait horribles.

→ Structure du Neurone :

Struct node {

 double * w; // Vecteur de mémoire

 char * etiq;

 double act; // état d'activation du neurone.
 : (distance Euclidienne)

entre V et w.

}

Comment remplir notre w ??

→ Regarder (*) (cours 30/01/2018)

Best Match Unit

Lee BMU, pour sélectionner le neurone le plus proche de notre vecteur de donnée piégé.

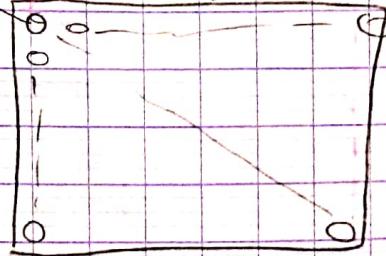
Le calcul d'état d'activation :

→ On calcule la distance euclidienne entre le vecteur de donnée piégé et les vecteurs de mémoire (w).

Algo :

→ on considère w_0 la plus petite distance et on calcule pour w_1 . Si c'est plus petit on permute(w_1) il devient le vecteur piégé ainsi de suite

→ f --- } w.



{ - - }

Free

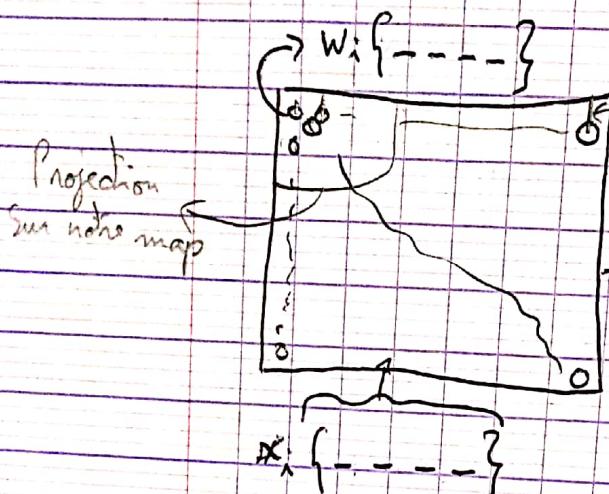
WST M \Rightarrow Winners Takes Most

SOM Self Organising Map

T. Kohonen

Réduction de la taille de données initial (\leftarrow Cartes projection corticales \Rightarrow un espace bi-dimensionnel \hookrightarrow Cartes topologique. \Rightarrow \hookrightarrow sensiblement à la densité des données)

Pour le calcul de act, on parcourt notre map. Séquence



état activation

(Value act) \Rightarrow celle-ci permet de déterminer le BMU,

Map (topologie)

(ceci puisque c'est un modèle WTA)

$(0,0)$

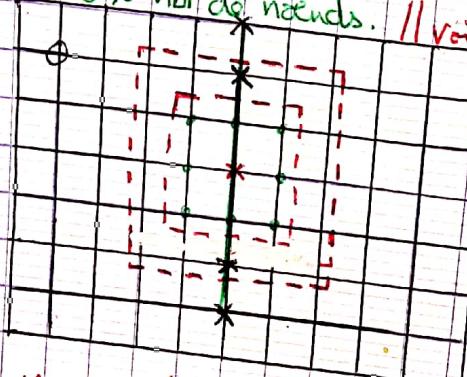
Les neurones servent de ~~outil~~ de reconnaissance notre vecteur de donnée.

Comment gérer notre topologie ??

① phase de compétition \Rightarrow permet de déterminer BMU, grâce à l'activation.

② phase de diffusion \Rightarrow grâce au voisinage

Chaque neurone doit avoir le même nombre de voisin. \hookrightarrow 50% nbr de voisins. // voisinage initial.



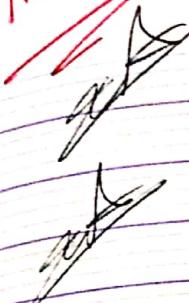
$j \neq \text{BMU}$

~~Si la coordonnée~~

Si le résultat de sa ~~est~~ (Coordonnée BMU) - (Coordonnée de Voisinage) est: \rightarrow négatif \Rightarrow on reste à 0 (le premier élément de notre matrice) \rightarrow ça dépasse le nombre de colonnes, on reste à 0.

Vecteur Moyen \Rightarrow $(-0.8, +0.1)$

Neurobiologie



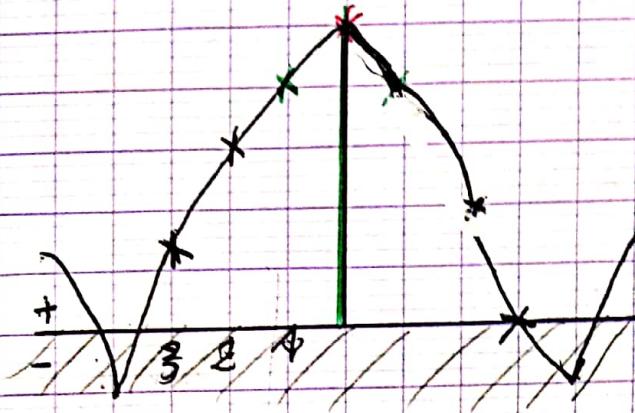
On donne : α

Mexican Hat Function

Écoutez pour un domaine d'application des réseaux de neurones.

Sequentiellement

$$(\ell, \sigma) = (\ell_{n-1}, \sigma_{n-1})$$



On retient que sur le positif.

Règle d'apprentissage

$$w_{ij}^{(t+1)} = w_{ij}^{(t)} + \alpha Vng(x) (x - w_{ij}^{(t)})$$
$$\forall j \in Vng(\cdot)$$
$$w_{ij}^{(t+1)} = w_{ij}^{(t)}, \forall i \notin Vng(\cdot)$$

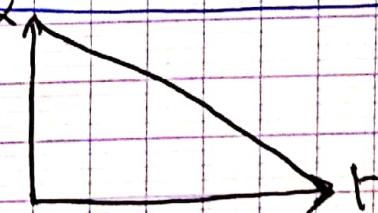
α : coef. d'apprentissage

$$\alpha_{\text{init}}: 0,7 \text{ à } 0,9$$

croissance linéaire

$$\alpha^{(t)} = \alpha_{\text{init}} \left(1 - \frac{t}{t_{\text{final}}} \right)$$

t : nbr iteration



→ La règle d'apprentissage sert à modifier le vecteur de mémoire

à devoisage) est :
(premier élément de notre map)

Analogie avec cellule souche.

Résumé :

① Phase Initialization:

→ charger Data.

→ Normaliser.

→ $w_i \leftarrow \text{random}(\text{centres})$ après avoir calculé le Vecteur Moyen

→ Shuffle (init Tab).

→ param Cfg(α , nbr(t), Mailage, ...)

$\text{nbr_it}_1, \text{nbr_it}_2, \dots$ type Carré...

② Phase d'apprentissage :

Pour nbr_IT total (phase 1 et 2)

$\forall x \in \text{Data-B}$

$\forall \text{node}_j \in \text{Map}$

$\exists \text{dist}_E(x, w_i) \rightarrow \text{BMC}_i$.

$\forall \text{node}_i \in \text{Nrd}(i)$

\exists appliquer learning Rule {

Vecteur Moyen
w.

$x \in \text{Data}$

①

trouver le BMC \rightarrow étiquetage Map

taille des vecteurs de données

Struct N_cfg {

int n_in; //nbr ap

int n_out; //nbr boutons

int n_c_out; 6 to

int n_l_out; 6

} N;

Struct neuronne {

double * w;

double act; // distance Euclidienne V, W

char * ep;

type def Struct neuronne t_node;

$$n_{out} = 5 * \sqrt{\text{nbr VecEnv}} \approx 60$$

Struct Net

double & caps

t_node & map;

{ double alpha; //coeff Apprentissage.

int r_vng; //rayon Voisinage / il est égal au nbr de fois
char t_eliq; // qui on agrandit notre voisinage

}

{ type def struct Net t_net;
t_net * Net = NULL;

Struct: ~~bmu~~ bmu

double act;

int bmu_l;

int bmu_c;

}

typedef struct bmu t_bmu;
t_bmu * bmu = Null;

Coeff d'apprentissage

$\alpha_{init}: 0.7 \text{ à } 0.9$ ~ ① Equilibrage : $\frac{1}{a} \text{ à } \frac{1}{5}$ nbr total Iteration

② Affinage :

$\frac{3}{4} \text{ à } \frac{4}{5}$ nb_total_IT

$\rightarrow \alpha = \frac{1}{t_0} + \alpha_{init}$.

For nbr iter
Scanner
vect
Vect

linearisat^(t) = $\alpha_{init} - \left(1 - \left(\frac{t}{t_{total}} \right) \right)$. || pas la phase
au début, il décend. (Voir photo) \rightarrow plus d'explications.

Minimum

Inverse du temps (inverse time):

$$\alpha^{(t)} = \alpha_{init} \left(\frac{C}{C+t} \right)$$

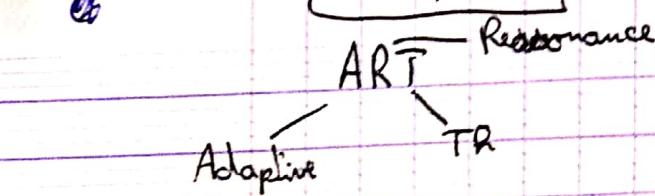
$$C = \text{Constante Fixée à } C = t_{total}/100$$

Triangle de Kanizsa.

73.6175

es

01/02/2018



Rep. local
Non Sup

Propagating

C-PN

Outer Network

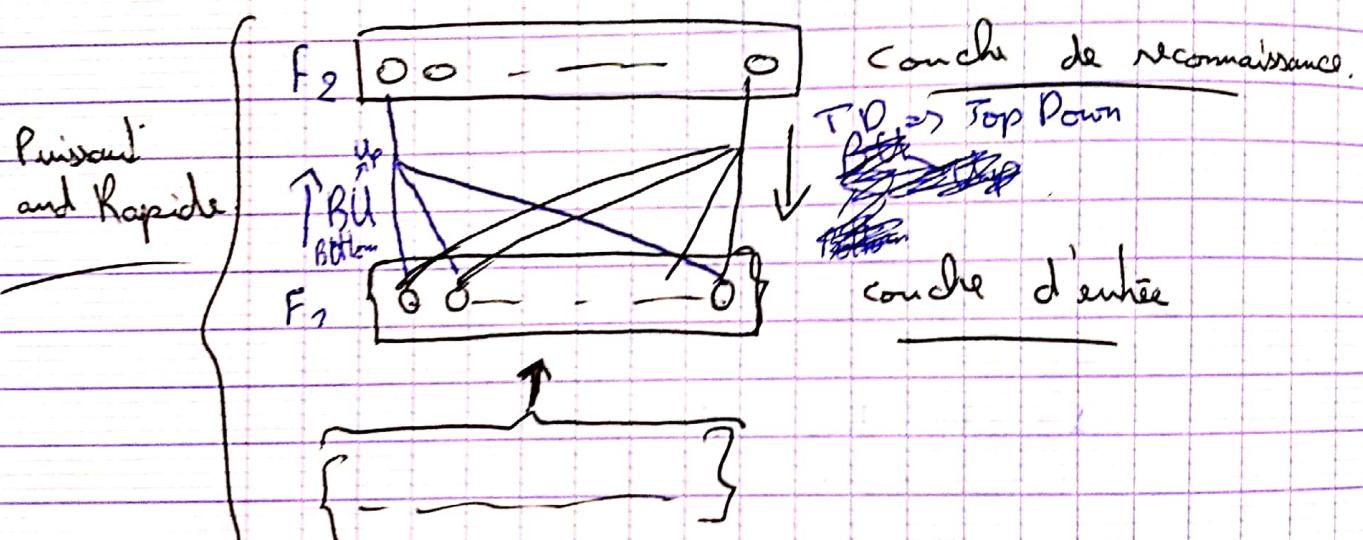
S. Grossberg } Boston University.
G. Carpenter }

→ Fonctionne en temps réel (avec une base de données qui change)

→ Apprentissage lent

→ Apprentissage rapide

Documentation
on the website
of the university



Modèle WTA



Chaque couche, contient des sous couche ...

Neo Cognition K. Fukushima \Rightarrow Colonnes Corticales...
 (Hubel & Wiesel)

Fonction
de Voisinage $\rightarrow n \rho_{(j, bmu)}^{(t)}$ = $n \rho_0(||j - bmu||, t)$

H. Ritter, Martínez
 , Stiller 83

$||j - bmu|| =$ dist. entre ~~Eucleenne~~ nodes dans la carte
 (coord l & C)

$t \downarrow n \rho_0(\cdot) \rightarrow 0$ quand $t \rightarrow \infty$

$$n \rho_0^{(t)}(j, bmu) = \Sigma^{(t)} \exp\left(-\frac{d_{node}(j, bmu)^2}{\epsilon^2(t)}\right)$$

(Rayon de Voisinage)

Σ = amplitude, α = longueur; $Vals \ init \neq$, eval
 identiques

$$\alpha^{(t)} = \alpha_{init} \left(\frac{\alpha_{final}}{\alpha_i} \right) \frac{t}{t_{max}}$$

, $\alpha_{init} \rightarrow \alpha_{final}$

$$\Sigma^{(t)} = \Sigma_{init} \left(\frac{\Sigma_{final}}{\Sigma_{init}} \right) \frac{t}{t_{max}}$$

$\Sigma_{init} \rightarrow \Sigma_{final}$.

$$n \rho_{(j, bmu)}^{(t)}$$

$$\begin{cases} 1, \text{ si } ||j - bmu|| < R_{size}^{(t)} \\ 0, \text{ sinon} \end{cases}$$

Rayon de
Voisinage

α : coeff d'apprentissage

$$0 < \alpha^{(t)} < 1$$

$$\alpha_{\text{init}} > \alpha_{\text{final}} \quad (\alpha_{\text{init}} = 0,5, \alpha_{\text{final}} = 0,005)$$

① Evolution Linéaire :

$$\alpha^{(t)} = \alpha_{\text{init}} \left(1 - \frac{t}{t_{\text{max}}} \right)$$

② Power Series :

$$\alpha^{(t)} = \alpha_{\text{init}} \left(\left(\frac{\alpha_{\text{final}}}{\alpha_{\text{init}}} \right)^{\frac{t}{t_{\text{max}}}} \right)^K$$

③ Inv Time :

$$\rightarrow \alpha^{(t)} = \alpha_{\text{init}} / \left(1 + 100 \left(\frac{t}{t_{\text{max}}} \right) \right)$$

$$\rightarrow \alpha^{(t)} \geq \alpha_{\text{init}} \left(\frac{C}{C + t} \right)$$

$$C = \text{const} = \frac{t_{\text{max}}}{100}$$