Vincent Boucheny (a.k.a. Mankalas)

Option SCIA

Réseaux de neurones 1

Promo 2007

Ce document reprend les prises de notes effectuées durant le cours de Yves-Jean Daniel et n'est en aucun cas destiné à être diffusé à l'extérieur du cadre de l'EPITA.

Table des matières

Table des figures

1 Introduction

1.1 Apprentissage numérique

Entrée : base de données.
Sortie : système d'aide à la décision.

Exemple :

Entrée Une base de données de symptômes étiquettés par des pathologies.

Sortie Une liste de symptômes renvoit un diagnostique.

1.2 Notion de transparence du système produit

1. Boîte noire : entrée → sortie sans aucune explication sur le chemin suivi.
2. Boîte transparente : entrée → sortie avec une explication totale du raisonnement suivi pour passer de l'un à l'autre
3. Boîte translucide : entrée → sortie avec un cas intermédiaire.

En général, plus une boîte est transparente, moins les sorties sont de bonne qualité.
Un apprentissage est réalisé dans un but. La qualité du système d'apprentissage est uniquement définie par sa capacité à résoudre le but. En général, la qualité doit être donnée par une fonction d'erreur. Celle-ci doit être expliquée et doit être validée par le client. Un système doit être toujours fourni avec une procédure de validation
– une preuve mathématique
– une batterie de tests

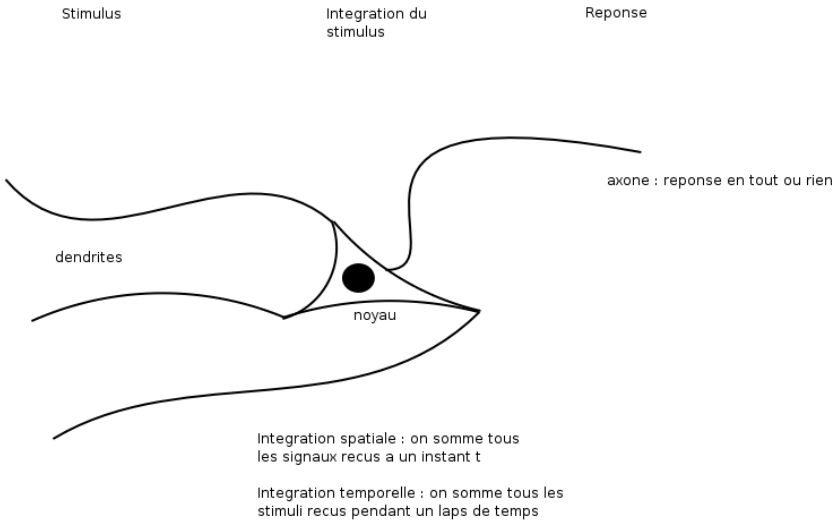

FIG. 1 – Illustration d'un neurone biologique

1.3 Apprendre à partir d'exemples

– Choix à partir d'un modèle (un modèle est une famille de fonctions).
– Apprendre dans ce modèle = recherche de la fonction de l'ensemble "passant au mieux" par les exemples. "Au mieux" est défini par une fonction d'énergie [erreur].

1. Modèles fonctionnels : entrée x associe de façon déterministe une sortie y.
2. Modèles probabilistes : entrée x associe de façon non nécessairement déterministe une sortie y.

Exemple :

1. À un poids x on associe la taille moyenne des individus de poids x.
2. À un poids x on associe une taille y telle que $P(Y = y \backslash X = x) = P$(un individu de poids x ait une taille y).

Exemple :
Dans la simulation d'un jet de dé à 6 faces, le premier modèle renvoit toujours $3,5$.

1.4 Différents types de réponse

1. Réponse par valeur : âge, couleur de pixel, etc.
2. Réponse par étiquette : booléen, étiquette de classe [classification, segmentation].

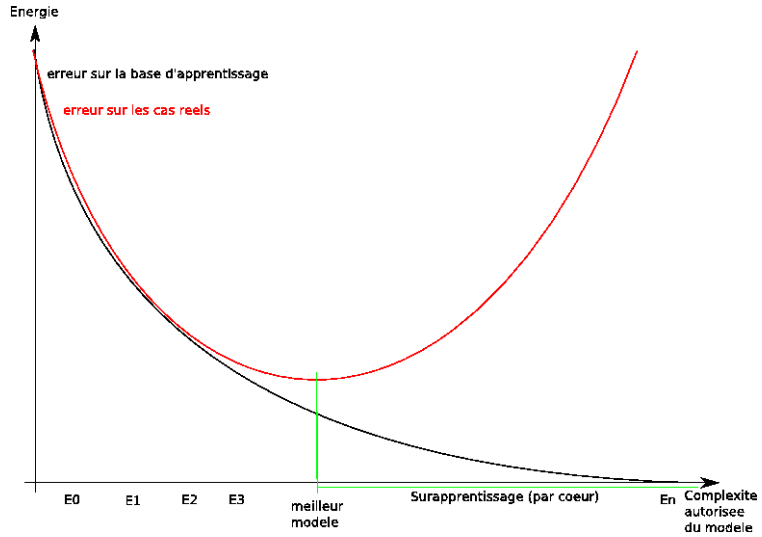

FIG. 2 – Graphe d'énergie de complexité d'un modèle

Exemple :

$$E_i = \left\{ \begin{array}{ccc} \mathbb{R} & \rightarrow & \mathbb{R} \\ x & \mapsto & \sum_{j=0}^{i} \lambda_j x^j / (\lambda_j)_{j=0}^{i} \in \mathbb{R}^{i+j} \end{array} \right\} = \mathbb{R}_i[X] \quad (1)$$

C'est ce qu'on appelle le compromis biais / variance (voir Figure 3).

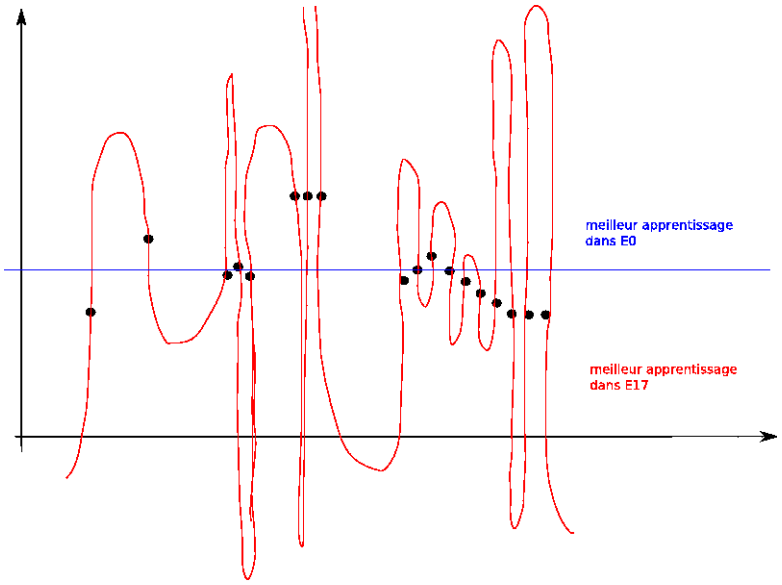

FIG. 3 – Exemple de compromis biais / variance

2 Approximation linéaire

On dispose
– d'un modèle d'apprentissage $T = \{(x(i), y(i))\}_{i=1}^{nb\_ex}$ avec $x(i) \in \mathbb{R}^n$ et $y(i) \in \mathbb{R}$.
– d'un modèle : $\left\{ \begin{array}{ccc} \mathbb{R}^n & \rightarrow & \mathbb{R} \\ x & \mapsto & \sum_{i=1}^{n} w_i x_i \end{array} \right\}$
– d'une fonction d'énergie : $E = \frac{1}{2} \sum_{i=1}^{nb\_ex} (y(i) - \widehat{y(i)})^2$ avec $\widehat{y(i)} = \sum_{i=1}^{n} w_i x_i$.
Le but est de trouver w qui minimise E.

Cas où $n = 2$

$$E = \frac{1}{2} \sum_{i=1}^{nb\_ex} (y(i) - \widehat{y(i)})^2 \quad (2)$$

$$= \frac{1}{2} \sum_{i=1}^{nb\_ex} \left(y(i) - \sum_{j=1}^{2} w_j x_j(i) \right)^2 \quad (3)$$

$$= \frac{1}{2} \sum_{i=1}^{nb\_ex} (y(i) - w_1 x_1(i) - w_2 x_2(i))^2 \quad (4)$$

On a

$$\text{grad}_w(E) = \begin{pmatrix} \frac{\partial E}{\partial w_1} \\ \frac{\partial E}{\partial w_2} \end{pmatrix} \quad (5)$$

$$= \begin{pmatrix} \sum_{i=1}^{nb\_ex} (-y(i) + w_1 x_1(i) + w_2 x_2(i)) x_1(i) \\ \sum_{i=1}^{nb\_ex} (-y(i) + w_1 x_1(i) + w_2 x_2(i)) x_2(i) \end{pmatrix} \quad (6)$$

Donc

$$\text{grad}_w(E) = 0 \iff \begin{pmatrix} w_1 \sum_{i=1}^{nb\_ex} x_1(i)^2 & + & w_2 \sum_{i=1}^{nb\_ex} x_1(i)x_2(i) & = & \sum_{i=1}^{nb\_ex} y(i)x_1(i) \\ w_1 \sum_{i=1}^{nb\_ex} x_1(i)x_2(i) & + & w_2 \sum_{i=1}^{nb\_ex} x_2(i)^2 & = & \sum_{i=1}^{nb\_ex} y(i)x_2(i) \end{pmatrix} = 0 \quad (7)$$

C'est un système linéaire de deux équations à deux inconnues : le système est résolu.

Rangement des données :

$$X = \begin{pmatrix} x_1(1) & \dots & x_1(nb\_ex) \\ x_2(1) & \dots & x_2(nb\_ex) \\ \vdots & & \vdots \\ x_n(1) & \dots & x_n(nb\_ex) \end{pmatrix}; \quad Y = \begin{pmatrix} y(1) \\ \vdots \\ y(nb\_ex) \end{pmatrix}; \quad W = \begin{pmatrix} w_1 \\ \vdots \\ w_n \end{pmatrix}$$

On peut exprimer le cas $n = 2$ en notation matricielle : $XX^tW = XY$. Si XX^t est inversible, $W = [(XX^t)^{-1}X]Y$. On note $X^+ = (XX^t)^{-1}$. On l'appelle pseudo-inverse de Penrose. Donc

$$W = X^+Y \quad (8)$$

Cas général

$$E = \frac{1}{2} \sum_{i=1}^{nb\_ex} (\widehat{y(i)} - y(i))^2 \quad (9)$$

$$= \frac{1}{2} \sum_{i=1}^{nb\_ex} \left(\sum_{j=1}^{n} w_j x_j(i) - y(i) \right)^2 \quad (10)$$

$$= \frac{1}{2} \sum_{i=1}^{nb\_ex} ((X^tW)_i - y(i))^2 \quad (11)$$

Pour les dérivées usuelles, voir page 27.

$$\text{grad}_W(E) = \frac{1}{2} \sum_{i=1}^{nb\_ex} 2((X^tW)_i - y(i))\text{grad}_W((X^tW)_i - y(i)) \quad (12)$$

$$= \sum_{i=1}^{nb\_ex} ((X^tW)_i - y(i))\text{grad}_W((X^tW)_i) \quad (13)$$

$$= \sum_{i=1}^{nb\_ex} ((X^tW)_i - y(i)) \times \underbrace{X^t_{i,*}}_{i\text{ème colonne de } X} \quad (14)$$

$$= \sum_{i=1}^{nb\_ex} (X^tW)_i X^t_{i,*} - \sum_{i=1}^{nb\_ex} y(i) X^t_{i,*} \quad (15)$$

On a

$$\sum_{i=1}^{nb\_ex} y(i)X_{i,*}^t = \begin{pmatrix} X_{1,1}y(1) \\ X_{2,1}y(1) \\ \vdots \\ X_{n,1}y(1) \end{pmatrix} + \cdots + \begin{pmatrix} X_{1,nb\_ex}y(nb\_ex) \\ X_{2,nb\_ex}y(nb\_ex) \\ \vdots \\ X_{n,nb\_ex}y(nb\_ex) \end{pmatrix} \tag{16}$$

$$= \begin{pmatrix} \sum_{k=1}^{nb\_ex} X_{1,k}y(k) \\ \vdots \\ \sum_{k=1}^{nb\_ex} X_{n,k}y(k) \end{pmatrix} \tag{17}$$

De la même façon, on trouve que

$$\sum_{i=1}^{nb\_ex} (X^tW)_i X_{i,*}^t = X(X^tW) \tag{18}$$

Finalement

$$\text{grad}_W(E) = 0 \iff XX^tW = XY \tag{19}$$

Si XX^t est inversible, alors $W = (XX^t)^{-1}XY$. Mais XX^t est-elle inversible ? Si XX^t n'est pas inversible, alors il existe une infinité de points où la dérivée est nulle, il y a donc une infinité de solutions au problème, donc les points d'entrée sont tous alignés sur un sous-espace vectoriel strict. Or, si on prend assez de points ($> n$), alors si les points sont suffisamment aléatoires, cette situation est de probabilité nulle.
Conclusion pratique sur des données réelles, on peut toujours considérer que la matrice est mathématiquement inversible.
On code une inversion de Gauss et on se rend compte que dès que n est élevé, les matrices deviennent numériquement non-inversibles. Lorsqu'on prend assez de données, celles-ci deviennent de plus en plus corrélées et on se rapproche donc du cas où les exemples sont sur un sous-espace vectoriel propre : la matrice est alors numériquement non-inversible.
Pour savoir si une matrice est facile à inverser numériquement, on calcule son conditionnement. Lorsqu'elle devient numériquement non-inversible, on utilise l'algorithme de Greville pour résoudre en W l'équation $XX^tW = XY$.

2.1 Quelques applications

1. Un outil sur un tour usine des cyclindres de diamètre 10mm. Au fur et à mesure de l'usinage, il y a usure de l'outil. But : trouver la correction à apporter à la position de l'outil dans le but que la cote nominale soit respectée le plus longtemps possible entre deux changements de l'outil.

   Base de donnée :
   - Série 1 : $\emptyset$ pièce 1, $\emptyset$ pièce 2, ...
   - Série 2 : $\emptyset$ pièce 1, $\emptyset$ pièce 2, ...
   - ...

   En utilisant les informations de la première série

$$X = \begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & 2 & 3 & \dots & nb\_pieces \end{pmatrix}; \quad Y = \begin{pmatrix} \emptyset p_1, S_1 \\ \emptyset p_2, S_1 \\ \vdots \\ \emptyset p_{nb\_pieces}, S_1 \end{pmatrix}; \quad W = X^+Y = \begin{pmatrix} w_1 \\ w_2 \end{pmatrix}$$

   Donc $\emptyset(p_i, S_1) = w_1 + w_2 i$. Correction à apporter à la position de l'outil avant de présenter la pièce $i \geq 2$:

$$\Delta pos(i) = w_1 + w_2 i - 10 - (w_1 + w_2(i-1) - 10) = w2$$

   Pour $i = 1$, $\Delta pos(1) = w_1 + w_2 - 10$.

Comment utiliser les autres séries ?

$$X = \begin{pmatrix} 1 & 1 & \dots & 1 & 1 & 1 & \dots & 1 & 1 & \dots \\ 1 & 1 & \dots & 1 & 2 & 2 & \dots & 2 & 3 & \dots \end{pmatrix}; \quad Y = \begin{pmatrix} \emptyset p_1, S_1 \\ \emptyset p_2, S_1 \\ \vdots \\ \emptyset p_m, S_1 \\ \emptyset p_1, S_2 \\ \emptyset p_2, S_2 \\ \vdots \\ \emptyset p_n, S_2 \\ \emptyset p_1, S_3 \\ \vdots \\ \emptyset p_q, S_{nb\_series} \end{pmatrix}$$

2. Prédiction du poids en fonction de la taille, de l'âge, du nombre d'heures passées à réviser les partiels.

$$X = \begin{pmatrix} 1 & \dots & 1 \\ taille(e_1) & \dots & taille(e_n) \\ age(e_1) & \dots & age(e_n) \\ partiel(e_1) & \dots & partiel(e_n) \end{pmatrix}; \quad Y = \begin{pmatrix} poids(e_1) \\ \vdots \\ poids(e_n) \end{pmatrix}$$

Donc $poids = w_0 + w_1 taille + w_2 age + w_3 partiel$.

Remarque : Ce modèle peut-il nous donner une indication sur une relation de causalité ? Pour cela, un pré-traitement de normalisation des données est nécessaire. On va passer en coordonnées centrées-réduites : $taille \leftarrow \dfrac{taille - E(taille)}{\rho_{taille}}$, pareil pour les autres caractéristiques.

Remarque : On ne peut pas retoucher au biais $(1, \dots, 1)$ de X. On n'a pas besoin de centrer, réduire les composantes de Y.

On réalise la recherche de w. Plus $|w_i|$ est important, plus on peut supposer qu'il existe un lien de causalité du paramètre w_i vers le résultat.

Remarque : Théorème central limite : la somme des quantités aléatoires → loi normale (globalement).

La loi normale est symétrique, son intégrale égale 1. Si on obtient une infinité de résultats, on se rapproche de l'espérance.

Si les données sont quasi-linéaires, alors cette exploitation des résultats n'a pas de sens.

3. Auto-régression :
   (a) Prédire $\sin(t)$ en fonction de $\sin(t-1)$, $\sin(t-2)$, $\sin(t-3)$ sachant que

$$\widehat{\sin(t)} = w_0 + w_1 \sin(t-1) + w_2 \sin(t-2) + w_3 \sin(t-3)$$

$$X = \begin{pmatrix} 1 & 1 & \dots & 1 \\ \sin(d+2) & \sin(d+3) & \dots & \sin(f-1) \\ \sin(d+1) & \sin(d+2) & \dots & \sin(f-2) \\ \sin(d) & \sin(d+1) & \dots & \sin(f-3) \end{pmatrix}; \quad Y = \begin{pmatrix} \sin(d+3) \\ \vdots \\ \sin(f) \end{pmatrix}$$

   (b) Test info : $\sin(n + \sin(n^2))$. On peut utiliser la prédiction de la façon suivante pour prédire $\sin(5)$.
      i. Figure 4 : utilisation correspondant naturellement à l'apprentissage.
      ii. Figure 5 : utilisation ne correspondant pas à l'apprentissage.

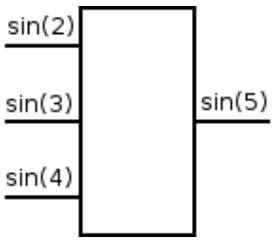

FIG. 4 – Prédiction de sin(5) - apprentissage

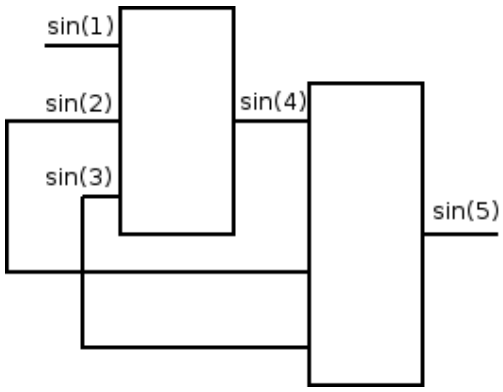

FIG. 5 – Prédiction de sin(5) - non apprentissage

Remarque : Si on veut avoir un apprentissage qui corresponde à l'utilisation, on doit, présenter le premier exemple, calculer w, présenter le deuxième exemple, calculer w. Du coup, X n'existe pas.

Remarque Autre mise en pratique : on travaille sur une matrice $X(t)$ qui correspond à une fenêtre glissante de largeur déterminée sur les données.

Remarque Hypothèse cognitive : Le comportement actuel est le même que le comportement à un passé proche.

$$X(act) = \begin{pmatrix} 1 & \dots & 1 \\ \sin(act-10) & \dots & \sin(act-1) \\ \sin(act-9) & \dots & \sin(act-2) \\ \sin(act-8) & \dots & \sin(act-3) \end{pmatrix}; \quad Y(act) = \begin{pmatrix} \sin(act-9) \\ \vdots \\ \sin(act) \end{pmatrix}$$

On obtient, après application de la méthode, un vecteur $w(actuel)$.

(c) Prédire le prix de la banane à la date t en fonction du prix de la banane aux dates $t-1$ et $t-2$ sachant que

$$\widehat{pb(t)} = w_0 + w_1 pb(t-1) + w_2 pb(t-2) + w_3 pb(t-1)pb(t-2) + w_4 \ln\left(\frac{pb(t-1)}{pb(t-2)}\right)$$

$$X = \begin{pmatrix} 1 & \dots \\ pb(d+1) & \dots \\ pb(d) & \dots \\ pb(d)pb(d+1) & \dots \\ \ln\left(\frac{pb(d+1)}{pb(d)}\right) & \dots \end{pmatrix}; \quad Y = \begin{pmatrix} pb(d+2) \\ \vdots \\ pb(f) \end{pmatrix}$$

Le modèle est linéaire par rapport à ses entrées (en w), mais le modèle n'est pas linéaire par rapport aux données.

Exemple Algorithme de visualisation d'un jeu d'échec :

On construit un arbre de positions dont la racine est la position actuelle. L'arc reliant un nœud à un fils est étiqueté par le coup qui est joué. On ajoute aux feuilles une étiquette qui est une note dans $\mathbb{R}$, une note de la position. Plus elle est élevée, plus le premier joueur à jouer est content.

On veut construire un modèle de note (position) à partir d'une base de données de parties étiquetées. On interroge des experts en échec. Question posée : que regardez-vous pour savoir qui est en train de gagner ?

On obtient une liste de critères
- le nombre de pièces de chaque joueur.
- quelles sont les pièces encore présentes.
- le nombre de pièces au centre.
- le nombre de pièces proche de la dame.
- le nombre de pièces protégées.
- ...

On en déduit le modèle suivant (par exemple) : $note = w_0 nb\_noir + w_1 nb\_blancs + w_2 nb\_noirs\_proteges + \ldots$

3 Analyse en composantes principales

3.1 Détection des linéarités

On veut trouver un sous-espace affine tel qu'en projetant notre ensemble de points sur ce sous-espace, on perde un minimum d'information.
On se donne une image en 1024×768, des 256 niveaux de gris. On représente cette image par des blocs de 16×16. On a $\frac{1024\times768}{16^2}$ blocs.
On réécrit chaque bloc dans un vecteur. Pour les représenter (graphiquement), on fait comme si chaque bloc était de dimension 2 (au lieu d'être de dimension 16×16). On note p la projection perpendiculaire sur le but. Le but est de minimiser l'espérance des éléments au carré.
On va tout d'abord chercher le meilleur axe de projection. On définit cet axe par (A, q) où A est un point de l'axe et q est un vecteur directeur unitaire.

Soit un point $M = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}$, déterminer la distance de M à l'axe.

À l'origine,

$$d(M, axe) = \sqrt{< M-A|M-A > - < M-A|q >^2} \tag{20}$$

But : trouver l'axe tel que $\sum_{i=1}^{nb\_pts} d(M(i), axe)^2$ soit minimal. On pose

$$E = \sum_{i=1}^{nb\_pts} < M(i)-A|M(i)-A > - < M(i)-A|q >^2 \tag{21}$$

Recherche de A :

$$\text{grad}_A(E) = 0 \iff \sum_{i=1}^{nb\_pts} 2(M(i)-A) - 2 < M(i)-A|q > (-q) = 0 \tag{22}$$

$$\iff \sum_{i=1}^{nb\_pts} M(i) = \sum_{i=1}^{nb\_pts} A - q < M(i)-A|q > \tag{23}$$

$$\iff \sum_{i=1}^{nb\_pts} M(i) + \sum_{i=1}^{nb\_pts} q < M(i)|q > -q < A|q > = nb\_ptsA \tag{24}$$

On note $G = E(M)$. Donc

$$G - A = -\frac{1}{nb\_pts}\left(\sum_{i=1}^{nb\_pts} q < A|q > - \sum_{i=1}^{nb\_pts} q < M(i)|q >\right) \tag{25}$$

$$= q < A|q > -q < E(M)|q > \tag{26}$$

$$= q < A-G|q > \tag{27}$$

G vérifie évidemment cette équation.
Dans toute la suite, on travaille en variables centrées : $M(i) \leftarrow M(i) - E(M)$. Sur ces variables centrées, l'axe passe par l'origine.
En variables centrées

$$E = \sum_{i=1}^{nb\_pts} < M(i)-G|M(i)-G > - < M(i)-G|q >^2 \tag{28}$$

$$= \sum_{i=1}^{nb\_pts} < M(i)|M(i) > - \sum_{i=1}^{nb\_pts} < M(i)|q >< M(i)|q > \tag{29}$$

But : minimiser E, ce qui revient à maximiser

$$\sum_{i=1}^{nb\_pts} < M|q >< M|q > = \sum_{i=1}^{nb\_pts} q^t M M^t q \quad (30)$$

$$= q^t \underbrace{\left(\sum_{i=1}^{nb\_pts} M M^t \right)}_{\text{On note } R} q \quad (31)$$

But : On recherche q qui maximise

$$\mathbb{E} = q^t R q \quad (32)$$

Losrqu'on a calculé $d(M, axe)$, on a considéré que $\|q\| = 1$.
Problème complet : Maximiser q sous la contrainte $\|q\| = 1$. On utilise la méthode du Lagrangien :

$$L = \mathbb{E} - \lambda(\|q\| - 1) \quad (33)$$

$$= \mathbb{E} - \lambda(q^t q - 1) \text{ car } q \text{ est unitaire} \quad (34)$$

$\text{grad}_q(L) = 0$ donne $2Rq = -2\lambda q = 0$ car R est carrée et symétrique. Donc $Rq = \lambda q$. q est un vecteur propre de R associé à la valeur propre λ. On recherche le meilleur vecteur propre, c'est-à-dire celui qui maximise $\mathbb{E}$.

$$\mathbb{E} = q^t R q = q^t \lambda q = \lambda q^t q \quad (35)$$

$$= \lambda \quad (36)$$

Le vecteur q recherché est le vecteur propre de R associé à la plus grande valeur propre de R.
Remarque : R positive $\Rightarrow \forall u, u^t R u \geq 0$. En effet,

$$u^t R u = u^t \sum M M^t u \quad (37)$$

$$= \sum u^t M M^t u \quad (38)$$

$$= \sum < u|M >< M|u > \quad (39)$$

$$= \sum < u|M >^2 \quad (40)$$

$$\geq 0 \quad (41)$$

Donc toutes les valeurs propres de R sont positives.
Maintenant, on recherche le deuxième meilleur axe.
On recommence le même travail sur une nouvelle base de données, non plus sur $(M(i))$ mais sur les $M(i)$ privés de leur composantes sur q_1 : $M(i) - q_1 < M|q_1 >$.

$$R_2 = \sum (M(i) - q_1 < M(i)|q_1 >)(M(i) - q_1 < M(i)|q_1 >)^t \quad (42)$$

$$= \sum (M M^t - < M|q_1 > M q_1^t - < M|q_1 > q_1 M^t + < M|q_1 >^2 q_1 q_1^t) \quad (43)$$

$$= R + \sum (-M^t q_1 M q_1^t - M^t q_1 q_1 M^t + (M^t q_1)^2 q_1 q_1^t) \quad (44)$$

$$= R + \sum (-M^t q_1 M q_1^t - M^t q_1 q_1 M^t) + \sum q_1^t M M^t q_1 q_1 q_1^t \quad (45)$$

$$= R + \sum (-M^t q_1 M q_1^t - M^t q_1 q_1 M^t) + q_1^t R q_1 q_1 q_1^t \quad (46)$$

$$= R + \sum (-M^t q_1 M q_1^t - M^t q_1 q_1 M^t) + q_1^t \lambda_1 q_1 q_1 q_1^t \quad (47)$$

$$= R + \sum (-M^t q_1 M q_1^t - M^t q_1 q_1 M^t) + \lambda_1 q_1 q_1^t \quad (48)$$

$$= R + \sum (-q_1^t M M q_1^t - q_1 M^t M^t q_1) + \lambda_1 q_1 q_1^t \quad (49)$$

Comme R est symétrique positive, ses vecteurs propres sont orthogonaux 2 à 2.
Remarque : Si de plus, on fixe des vecteur propres unitaires, si on choisit

$$P = \mathrm{Mat}_{(e_i),(q_i)} Id \text{ (matrice de passage)}$$

alors

$$p^{-1} = p^t$$

On a

$$\begin{aligned}
R_2 q_1 &= Rq_1 + \sum(-q_1^t MM - q_1 M^t M^t q_1 q_1) + \lambda_1 q_1 && (50)\\
&= 2\lambda_1 q_1 - \sum q_1^t MM - \sum q_1 M^t q_1 M^t q_1 && (51)\\
&= 2\lambda_1 q_1 - \sum q_1^t MM - q_1 q_1 \sum MM^t q_1 && (52)\\
&= 2\lambda_1 q_1 - \sum q_1^t MM - \lambda_1 q_1 && (53)\\
&= \lambda_1 q_1 - \sum q_1^t MM && (54)\\
&= \lambda_1 q_1 - \sum MM^t q_1 && (55)\\
&= \lambda_1 q_1 - \lambda_1 q_1 && (56)\\
&= 0 && (57)
\end{aligned}$$

Pour $i \neq 1$,

$$\begin{aligned}
R_2 q_i &= Rq_i - \sum M^t q_1 q_1 M^t q_i && (58)\\
&= \lambda_i q_i - \sum q_i^t MM^t q_1 q_1 && (59)\\
&= \lambda_i q_i - q_i^t Rq_1 q_1 && (60)\\
&= \lambda_i q_i - q_i^t \lambda_1 q_1 q_1 && (61)\\
&= 0 && (62)
\end{aligned}$$

On a montré que les vecteurs propres de R sont vecteurs propres de R_2. On montre de même que les vecteurs propres de R_2 sont des vecteurs propres de R, et ceci associés aux mêmes vecteurs propres (sauf pour q_1).
Donc, le $2^{\text{ème}}$ meilleur axe est celui engendré par q_2. Par récurrence, le $k^{\text{ième}}$ meilleur axe est celui associé à la $k^{\text{ième}}$ plus grande valeur propre de R.
On estime tout point x par sa projection sur les k premiers axes

$$\hat{x} = \sum_{i=1}^{k} < x|q_i > q_i$$

La quantité perdue sur x est

$$\begin{aligned}
\|x - \hat{x}\|^2 &= \left\| \sum_{i=k+1}^{n} < x|q_i > q_i \right\|^2 && (63)\\
&= \sum_{i=k+1}^{n} < x|q_i >^2 && (64)
\end{aligned}$$

L'espérance de la quantité perdue sur la base de données est

$$\frac{1}{nb\_pts}\sum_{i=1}^{nb\_pts}\sum_{j=k+1}^{n} < x|q_j >^2 = \frac{1}{nb\_pts}\sum_{j=k+1}^{n}\sum_{i=1}^{nb\_pts} < x|q_j >^2 \quad (65)$$

$$= \frac{1}{nb\_pts}\sum_{j=k+1}^{n}\sum_{i=1}^{nb\_pts} q_j^t x x^t q_j \quad (66)$$

$$= \frac{1}{nb\_pts}\sum_{j=k+1}^{n} q_j^t \left(\sum x x^t\right) q_j \quad (67)$$

$$= \frac{1}{nb\_pts}\sum_{j=k+1}^{n} q_j^t R q_j \quad (68)$$

$$= \frac{1}{nb\_pts}\sum_{j=k+1}^{n} q_j^t \lambda_j q_j \quad (69)$$

$$= \frac{1}{nb\_pts}\sum_{j=k+1}^{n} \lambda_j \quad (70)$$

Pourcentage de perte :

$$\frac{qte\ perdue}{qte\ a\ perdre} = \frac{\frac{1}{nb\_pts}\sum_{j=k+1}^{n}\lambda_j}{\frac{1}{nb\_pts}\sum_{j=1}^{n}\lambda_j} \quad (71)$$

$$= \frac{\sum_{j=k+1}^{n}\lambda_j}{\sum_{j=1}^{n}\lambda_j} \quad (72)$$

$$(73)$$

Supposons que l'on dispose d'un algorithme itératif capable de calculer successivement $(\lambda_1, q_1)(\lambda_2, q_2)\ldots$. La condition d'arrêt de cet algo est : % perte > tolérance. Réexprimons le pourcentage de perte sans faire appel au futur (mauvaise condition d'arrêt). Rappel : la trace est invariante par changement de base.
Pourcentage de perte :

$$\frac{\sum_{j=k+1}^{n}\lambda_j}{\sum_{j=1}^{n}\lambda_j} = \frac{\sum_{j=k+1}^{n}\lambda_j}{\mathrm{Tr}(R)} \quad (74)$$

$$= \frac{\mathrm{Tr}(R) - \sum_{j=1}^{k}\lambda_j}{\mathrm{Tr}(R)} \quad (75)$$

$$= 1 - \frac{\sum_{j=1}^{k}\lambda_j}{\mathrm{Tr}(R)} \quad (76)$$

C'est une bonne condition d'arrêt car on ne fait appel qu'à des quantités déjà calculées.

Résumé implémentation

1. Calculer l'espérance de M : $E(M)$.
2. Centrer les données : $\forall M, M \leftarrow M - E(M)$.

3. Réduction des données. On veut égaliser les différents champs. Pour ceci, on norme les composantes :

$$\forall M \in [\![1..nb\_pts]\!], \forall i \in [\![1..n]\!], M_i \leftarrow \frac{M_i}{\rho_i} \text{ avec } \rho_i = \sqrt{\frac{1}{nb\_pts} \sum_{k=1}^{nb\_pts} (M(k)_i)^2} \tag{77}$$

4. Calculer la matrice

$$R = \sum_{i=1}^{nb\_pts} M(i)M(i)^t \tag{78}$$

5. Chercher les vecteurs propres et les valeurs propres de R.

Recherche des vecteurs propres et les valeurs propres de R

1. Recherche globale (tous les vecteurs, toutes les valeurs) : cf. cours de maths.
   - Avantage : méthode la plus rapide si on désire tous les vecteurs propres.
   - Défaut :
     - En général instable numériquement.
     - Très lourde si on désire quelques vecteurs propres seulement.
     - Il faut avoir la matrice R.
2. Recherche incrémentale
   (a) En utilisant R. Idée : calcul de $R^p u$ où u est un vecteur propre de $\mathbb{R}^n$. u se décompose dans la base des vecteurs propres :

$$\vec{u} = \sum_{i=1}^{n} \alpha_i q_i \Rightarrow Ru = \sum_{i=1}^{n} \alpha_i R q_i = \sum_{i=1}^{n} \alpha_i \lambda_i q_i \tag{79}$$

On itère :

$$R^2 u = \sum_{i=1}^{n} \alpha_i \lambda_i R q_i = \sum_{i=1}^{n} \alpha_i \lambda_i^2 q_i \tag{80}$$

Par récurrence immédiate :

$$\begin{aligned} R^p u &= \sum_{i=1}^{n} \alpha_i \lambda_i^p q_i && (81)\\ &= \lambda_1^p \sum_{i=1}^{n} \alpha_i \left(\frac{\lambda_i}{\lambda_1}\right)^p q_i && (82)\\ &\underset{\substack{p\to+\infty \\ proba=1}}{\sim} \lambda_1^p \alpha_1 q_1 && (83) \end{aligned}$$

α_1 non nul avec une probabilité égale à 1. Donc

$$\frac{R^p u}{\|R^p u\|} \underset{proba=1}{\sim} \pm q_1 \tag{84}$$

À renommage près, on peut dire que :

$$\frac{R^p u}{\|R^p u\|} \to q_1 \tag{85}$$

/* Changement de notation du à une rupture de cours : R est renommé en A */

Cette méthode ne fonctionne pas dans la pratique par manque de précision de la part des ordinateurs.

1ère idée normalisation à chaque itération. On obtient une suite

$$\left\{ \begin{array}{rcl} u_0 & = & u \\ \forall n \in \mathbb{N}, u_{n+1} & = & \frac{Au_n}{\|Au_n\|} \end{array} \right. \tag{86}$$

Exprimons u_n en fonction de n :

$$u_1 = \frac{Au}{\|Au\|} \tag{87}$$

$$u_2 = \frac{A^2u}{\|Au_n\|} \div \frac{\|A^2u\|}{\|Au\|} = \frac{A^2u}{\|A^2u\|} \tag{88}$$

$$\vdots \tag{89}$$

$$u_p = \frac{A^pu}{\|A^pu\|} \tag{90}$$

Donc

$$u_p \underset{proba=1}{\rightarrow} \pm q_1 \tag{91}$$

Mais en machine, des erreurs de calcul se produisent et cette méthode devient assez instable.

2ème idée changer la normalisation.

$$\left\{ \begin{array}{rcl} u_0 & = & u \\ \forall n \in \mathbb{N}, u_{n+1} & = & \frac{Au_n}{(Au_n)_1} \end{array} \right. \tag{92}$$

Problème si la première composante est "ennuyeuse" (trop petite ou trop grande valeur). On change donc la méthode en

$$\left\{ \begin{array}{rcl} u_0 & = & u \\ \forall n \in \mathbb{N}, \alpha_{n+1} \leftarrow Alea[\![1..n]\!], u_{n+1} & = & \frac{Au_n}{(Au_n)_\alpha} \end{array} \right. \tag{93}$$

3ième idée pourquoi ne pas étendre ce principe à n'importe quel axe ?

$$\left\{ \begin{array}{rcl} u_0 & = & u \\ \forall n \in \mathbb{N}, v_{n+1} \leftarrow \text{vecteur aléatoire de dim } n, u_{n+1} & = & \frac{Au_n}{<Au_n|v_{n+1}>} \end{array} \right. \tag{94}$$

Un problème qui se pose est le coût de calcul de v_n. La solution choisie est : $v_n \leftarrow v_0$ et $v_0 \leftarrow$ vecteur aléatoire de dim n. On peut exprimer u_n en fonction de n :

$$u_1 = \frac{Au}{<Au|v_0>} \tag{95}$$

$$u_2 = \frac{A^2u}{<Au|v_0>} \div \frac{1}{<Au|v_0>} < A^2u|v_0> \tag{96}$$

$$\vdots \tag{97}$$

$$u_p = \frac{A^pu}{<A^pu|v_0>} \tag{98}$$

On note $v_0 = \sum \beta_i q_i$. Donc

$$u_p = \frac{\lambda_1^p \sum_{i=1}^{n} \left(\frac{\lambda_i}{\lambda_1}\right)^p \alpha_i q_i}{\lambda_1^p \sum_{i=1}^{n} \left(\frac{\lambda_i}{\lambda_1}\right)^p \alpha_i \beta_i} \tag{99}$$

$$\underset{\substack{proba=1 \\ <u|q_1>\neq 0 \wedge \\ <v|q_1>\neq 0}}{\sim} \frac{\lambda_1^p \alpha_1 q_1}{\lambda_1^p \alpha_1 \beta_1} \tag{100}$$

Ceci est la méthode de la déflation :

$$q_1 \leftarrow \frac{u_p}{\|u_p\|}; \quad Aq_1 = \lambda q_1 \Rightarrow \lambda_1 \leftarrow \|Aq_1\| \tag{101}$$

Recherche de q_2. Principe :
– $u \leftarrow$ vecteur aléatoire de dim n.
– $u \leftarrow u - < u|q_1 > q_1$.
On a $A^p u = \lambda_2^p \sum_{i=2}^{n} \left(\frac{\lambda_i}{\lambda_2}\right)^p \alpha_i q_i$. Si on applique la méthode de la déflation avec le vecteur u ainsi construit : $u_p \underset{proba=1}{\sim} \frac{\lambda_2^p \alpha_2 q_2}{\lambda_2^p \alpha_2 \beta_2}$, on code, et on converge vers q_1 à cause des erreurs de calcul.
Donc, au lieu de retirer une fois pour toutes la composante q_1, on la retire à chaque itération. On a

$$\begin{cases} u_0 & \leftarrow & \text{vecteur aléatoire de dim } n \\ v_0 & \leftarrow & \text{vecteur aléatoire de dim } n \\ u_n & = & u_n - < u_n|q_1 > q_1 \\ u_{n+1} & = & \frac{Au_n}{<Au_n|v_0>} \end{cases} \tag{102}$$

Compactons ces deux lignes :

$$u_{n+1} = \frac{A(u_n - < u_n|q_1 > q_1)}{< A(u_n - < u_n|q_1 > q_1)|v_0 >} \tag{103}$$

$$= \frac{Au_n - < u_n|q_1 > \lambda_1 q_1}{< Au_n - < u_n|q_1 > \lambda_1 q_1|v_0 >} \tag{104}$$

$$= \frac{Au_n - \lambda_1 q_1 q_1^t u_n}{< Au_n - \lambda_1 q_1 q_1^t u_n|v_0 >} \tag{105}$$

$$= \frac{(A - \lambda_1 q_1 q_1^t)u_n}{< (A - \lambda_1 q_1 q_1^t)u_n|v_0 >} \tag{106}$$

On note $B = A - \lambda_1 q_1 q_1^t$, on a donc

$$\forall n \in \mathbb{N}, u_{n+1} = \frac{Bu_n}{< Bu_n|v_0 >} \tag{107}$$

Il s'agit exactement du même algorithme que précédemment en remplaçant A par $B - A - \lambda_1 q_1 q_1^t$.

3.2 Algo de recherche des vecteurs et valeurs propres par ordre décroissant des λ_i

Algorithm 1 Recherche des vecteurs et valeurs propres

Require: Une matrice A symétrique positive à valeurs propres distinctes — le nombre de couples $(\vec{vp}, vp)$ cherchés (noté nbv)
for $i \leftarrow 1$ à nbv **do**
$u \leftarrow$ vecteur aléatoire de dim n
$v \leftarrow$ vecteur aléatoire de dim n
repeat
$u \leftarrow \frac{Au}{< Au|v >}$
until convergence
$(q, \lambda) \leftarrow \left(\frac{u}{\|u\|}, \frac{Au}{\|u\|}\right)$
Sortir (q, λ)
$A \leftarrow A - \lambda q q^t$
end for
end

3.3 Méthode générique de recherche de vecteurs et valeurs propres

Algorithmes HA et GHA (Hebbian Algorithm, Generic HA). On revient au problème de départ sous sa forme énergétique :

$$E = \frac{1}{2}\|x - < x|q > q\|^2 \tag{108}$$

On recherche un algorithme stochastique : on présente et on met à jour exemple par exemple.

Principe de l'algorithme

Algorithm 2 algorithmic il y a des exemples Présenter x $q \leftarrow q - \varepsilon \text{grad}_q(E)$

end

Cet algorithme est un outil de base en minimisation.
Dans le calcul de E, q est supposé de norme 1. Or cette condition est altérée par $q \leftarrow q - \varepsilon \text{grad}_q(E)$. On va forcer la norme de q à valoir 1 en renormant q à chaque itération. Attention, ce n'est plus une descente de gradient en énergie. Donc, si on veut garantir la convergence de l'algorithme vers q_1, il faudrait prouver que

1. L'algorithme converge.
2. Il converge vers q_1.

On a

$$\begin{aligned}
E &= \frac{1}{2}(x^t - < x|q > q^t)(x - < x|q > q) && (109)\\
&= \frac{1}{2}(x^t x - 2 < x|q >^2 + \underbrace{< x|q >^2}_{u}\underbrace{q^t q}_{v}) && (110)\\
\text{grad}_q(E) &= \frac{1}{2}(-4 < x|q > x + \underbrace{< x|q >^2}_{u}\underbrace{2q}_{v'} + \underbrace{2 < x|q > x}_{u'}\underbrace{q^t q}_{v;=1}) && (111)\\
&= - < x|q > x + 2 < x|q >^2 q && (112)
\end{aligned}$$

On a $q \leftarrow q - \varepsilon \text{grad}_q(E)$. Donc

$$\begin{aligned}
q_{new} &= q + \varepsilon < x|q > x - \varepsilon < x|q >^2 q && (113)\\
\|q_{new}\| &= \sqrt{\sum_{i=1}^{n}(q_i + \varepsilon < x|q > x_i - \varepsilon < x|q >^2 q_i)^2} && (114)\\
&= \sqrt{\sum_{i=1}^{n} q_i^2 + 2\varepsilon < x|q > x_i q_i - 2\varepsilon < x|q >^2 q_i^2 + \circ(\varepsilon^2)} && (115)\\
&= \sqrt{\left(\sum_{i=1}^{n} q_i^2\right) + 2\varepsilon < x|q > \sum_{i=1}^{n} x_i q_i - 2\varepsilon < x|q >^2 \sum_{i=1}^{n} q_i^2 + \circ(\varepsilon^2)} && (116)\\
&= \sqrt{1 + 2\varepsilon < x|q >^2 - 2\varepsilon < x|q >^2 + \circ(\varepsilon^2)} && (117)\\
&= \sqrt{1 + \circ(\varepsilon^2)} && (118)\\
&= 1 + \circ(\varepsilon^2) && (119)
\end{aligned}$$

Donc

$$\frac{q_{new}}{\|q_{new}\|} = q + \varepsilon < x|q > (x - < x|q > q) \tag{120}$$

On note $y =< x|q >$, on a alors

$$\frac{q_{new}}{\|q_{new}\|} = q + \varepsilon y(x - yq) \tag{121}$$

Ce résultat est valide uniquement si ε est très petit. La normalisation par $\|q_{new}\|$ n'a rien changé dans la formule. Dans la situation présente, la descente de gradient réalise automatiquement (par effet de bord) la normalisation pour ε assez petit.

Algorithm 3 Apprentissage

repeat
  Mélanger la base d'apprentissage $x(1), x(2), \ldots, x(nb\_ex)$
  for $i \leftarrow 1$ à nb_ex **do**
    Présenter x
    Calculer $y = w^t x$
    $w \leftarrow w + \varepsilon y(x - yw)$
  end for
  Diminuer la valeur de ε
until convergence

end

Remarque fondamentale : On travaille avec une base de données centrée. Si la base de donnée n'est pas centrée, on ajoute une entrée virtuelle x_0 valant 1 qui sert de biais, mais ce n'est pas économique.

Conclusion,

$$w \to q_1 \tag{122}$$

GHA Calcul des autres vecteurs propres. On va appliquer la même méthode mais sur les exemples privés de leur composante sur q_1. On renomme les w_i précédents en $w_{1,i}$ et y en y_1. Modification des $w_{2,*}$.

$$\begin{aligned}
w_{2,*} &\leftarrow w_{2,*} + \varepsilon w_{2,*}^t(x - w_{1,*}^t x w_{1,*})(x - w_{1,*}^t x w_{1,*} - w_{2,*}^t(x - w_{1,*}^t x w_{1,*})w_{2,*}) && (123)\\
&= w_{2,*} + \varepsilon(y_2 - y_1 \underbrace{w_{2,*}^t w_{1,*}}_{\to 0 \text{ (abus)}})(x - y_1 w_{1,*} - y_2 w_{2,*} + y_1 \underbrace{w_{2,*}^t w_{1,*}}_{\to 0 \text{ (abus)}} w_{2,*}) && (124)\\
&\approx w_{2,*} + \varepsilon y_2(x - y_1 w_{1,*} - y_2 w_{2,*}) && (125)\\
&\approx w_{2,*} + \varepsilon y_2 \left(x - \sum_{i=1}^{2} y_i w_{i,*}\right) && (126)
\end{aligned}$$

Par récurrence,

$$w_{i,*} \leftarrow w_{i,*} + \varepsilon y_i \left(x - \sum_{j=1}^{i} y_j w_{j,*}\right) \tag{127}$$

Le principe de calcul de $w_{2,*}, w_{3,*}, \ldots$ est que les $w_{i,*}$ sont justes. En réalité, la convergence n'est pas successive mais simultanée, c'est-à-dire $w_{2,*}, w_{3,*}, \ldots$ commencent à converger vers $q_1, q_2, \ldots$ avant que les $w_{i,*}$ précédents n'aient fini de converger. Une parallélisation est donc possible.

3.4 Applicatif

Représentation des données Diminuer le nombre de champs pour la représentation des données en limitant la perte d'information due à la projection.

Compression linéaire On décompose un signal dans une base adaptée et on supprime la partie des données portée sur les axes les moins significatifs.

Construction de base de données simplifiées à axes orthogonaux Recherche des axes qui rendent les nouveaux champs les plus indépendants possibles ← pré-traitement des données.

Classification On dispose de vignettes de visage. On assimile chaque vignette à un ensemble de vecteurs. On calcule les vecteurs et valeurs propres. On classe les vignettes en fonction des axes et valeurs propres.

Détection d'indépendance des variables Exemple dans une base de données : `Individu (age, poids, taille)`.

Si la relation suivante existe

$$\lambda age + \mu poids + \gamma taille = 0$$

alors $\lambda_3 \approx 0$.

Si on obtient $\lambda_2 \approx 0$, alors

$$(age = \mu poids \vee \lambda age = poids) \wedge (age = \gamma taille \vee taille = \lambda age) \wedge \ldots$$

Si uniquement $\lambda + 3 = 0$, en oubliant le cas où $\lambda = 0$, $age = \mu poids + \gamma taille$.

Pour rechercher la donnée la plus fortement corrélée, on projette les données sur (q_1, q_2) et on cherche le champ qui a le moins perdu

4 Perceptron

Classification d'un problème binaire. On dispose d'une base d'apprentissage $T = \{(x(i), d(i))\}_{i=1}^{nb\_ex}$ avec $x(i) \in \mathbb{R}^n$ et $d(i) \in \{-1, 1\}$. On cherche un hyperplan H, $H_w = \{x \in \mathbb{R}^n / x^t w = 0\}$ tel que d'un côté de l'hyperplan, il y ait tous les exemples à $d = 1$ et de l'autre côté, tous les exemples à $d = -1$.
Si le problème est solvable, il admet une infinité de solutions. Le fait de définir les côtés comme étant larges ou stricts ne change pas ce qui est classifiable.
L'exemple $x(i)$ est bien placé ssi Si $d(i) = 1$ alors $x(i)^t w => 0$ ou Si $d(i) = -1$, alors $x(i)^t w < 0$, ce qui revient à dire que l'exemple $x(i)$ est bien placé ssi $d(i) w^t x(i) > 0$.

4.1 On suppose le problème linéairement séparable

Algorithm 4 Placement des exemples

repeat
  Prendre un exemple $(x(i), d(i))$
  if $d(i) w^t x(i) > 0$ **then**
    Ne rien faire
  else
    $w \leftarrow w + \varepsilon d(i) x(i)$
  end if
until tous les exemples soient bien placés

end

Preuve de la convergence On renomme $x(i)d(i)$ en $y(i)$. On numérote le temps en nombre d'appels au "sinon" du code. $y(t)$ est l'exemple $x(i)d(i)$ qui a été utilisé lors du $t^{\text{ième}}$ appel au "sinon". On suppose que l'algorithme ne termine pas.
On a

$$\begin{aligned} w(t) &= w_0 + \varepsilon y(1) + \varepsilon y(2) + \cdots + \varepsilon y(t) && (128) \\ &= w_0 + \sum_{i=1}^{t} y(i) && (129) \end{aligned}$$

Comme le problème est linéairement séparable, $\exists u, \forall i, u^t d(i) x(i) > 0$. Comme la base d'exemples est finie, $\exists \alpha > 0 / \forall i, u^t d(i) x(i) > \alpha$. On a

$$\begin{aligned} w_{tps}^t u &= u^t w_0 + \varepsilon \sum_{i=1}^{tps} u^t y(i) && (130) \\ &> u^t w_0 + \varepsilon \alpha tps && (131) \end{aligned}$$

Rappel : inégalité de Cauchy-Schwartz

$$\|a\|^2 \|b\|^2 \geq < a|b >^2 \quad (132)$$

Preuve :

$$\begin{aligned} < \lambda a + b | \lambda a + b > \geq 0 &\Rightarrow \lambda^2 \|a\|^2 + 2\lambda < a|b > + \|b\|^2 \geq 0 \\ &\Rightarrow \Delta \leq 0 \\ &\Rightarrow 4 < a|b >^2 - 4\|a\|^2 \|b\|^2 \leq 0 \end{aligned}$$

On a donc

$$\|w_{tps}\|^2 \|u\|^2 \geq (w_{tps}^t u)^2 \quad (133)$$

Si $u = 0$, alors $\alpha = 0$, ce qui est IMPOSSIBLE. Donc $u \neq 0$. Donc

$$\|w_{tps}\|^2 \geq \frac{w_{tps}^t u}{\|u\|^2} \tag{134}$$

D'autre part,

$$\begin{aligned}
\|w_{tps}\|^2 &= \|w_{tps-1} + \varepsilon y_{tps}\|^2 && (135)\\
&\leq \|w_{tps-1}\|^2 + \varepsilon^2 \|y_{tps}\|^2 && (136)\\
\|w_{tps-1}\|^2 &\leq \|w_{tps-2}\|^2 + \varepsilon^2 \|y_{tps-1}\|^2 && (137)\\
\|w_{tps-2}\|^2 &\leq \|w_{tps-3}\|^2 + \varepsilon^2 \|y_{tps-2}\|^2 && (138)\\
&\vdots && (139)
\end{aligned}$$

En sommant ces termes, on obtient

$$\|w_{tps}\|^2 \leq \|w_0\|^2 + \varepsilon^2 \sum_{i=1}^{tps} \|y(i)\|^2 \tag{140}$$

Comme on a un nombre fini d'exemples, $\exists \beta > 0, \forall i, \|d(i)x(i)\|^2 < \beta$. Donc

$$\|w_{tps}\|^2 \leq \|w_0\|^2 + \varepsilon^2 \beta tps \tag{141}$$

Avec (133)

$$(w_{tps}^t u)^2 \leq (\|w_0\|^2 + \varepsilon^2 \beta tps)\|u\|^2 \tag{142}$$

Avec (131)

$$w_{tps}^t u > u^t w_0 + \varepsilon \alpha tps \tag{143}$$

En combinant ces deux lignes

$$\begin{aligned}
& u^t w_0 + \varepsilon \alpha tps < \sqrt{\|w_o\|^2 + \varepsilon \beta tps}\|u\| && (144)\\
\iff \quad & \frac{u^t w_0 + \varepsilon \alpha tps}{\sqrt{tps}} < \sqrt{\frac{\|w_0\|^2}{tps} + \varepsilon^2 \beta}\|u\| && (145)
\end{aligned}$$

Or, l'algorithme ne termine pas (hypothèse absurde), donc (145) est valide quelque soit le temps. Or le côté gauche tend vers $+\infty$ quand $tps \to +\infty$ et le côté droit a une limite finie quand $tps \to +\infty$. C'est IMPOSSIBLE, donc l'algorithme termine.

Remarque : La valeur de ε et de w_0 n'interviennent pas dans le fait que l'algorithme converge. Par contre, elles peuvent influencer le temps de convergence.

Remarque : Dans les applications pratiques, on n'oubliera pas de rajouter à tous les exemples une composante virtuelle à 1. Sinon, on cherche un séparateur qui passe par l'origine, et c'est rarement ce que l'on désire.

4.2 On suppose le cas non-linéairement séparable

Remarque Définition : Une époque est la présentation de tous les exemples de la base de données.
On va présenter plusieurs fois la base d'apprentissage et à chaque époque, on va diminuer la valeur de ε.
Règles classiques :

- $\sum_{epoque=1}^{\infty} \varepsilon_{epoque}\ DV.$

– $\sum_{epoque=1}^{\infty} \varepsilon_{epoque}^2$ CV.

Exemple :

$$\varepsilon_{epoque} = \frac{k_1}{k_2 epoque + k_3}$$

4.3 Perceptron multi-couches

4.3.1 Présentation

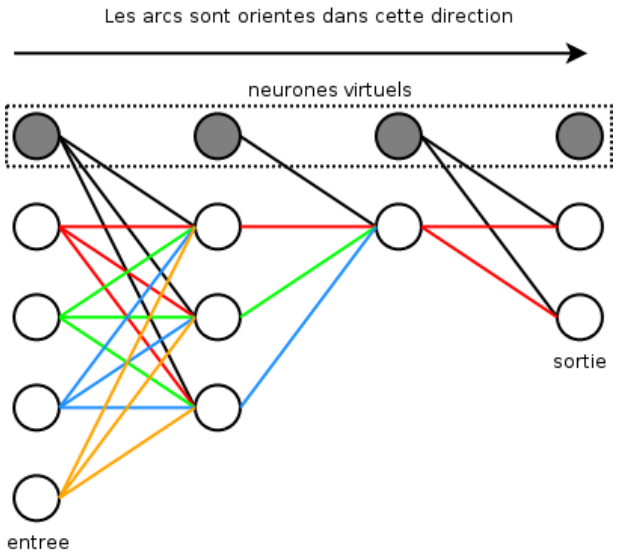

FIG. 6 – Représentation d'un perceptron multi-couches

Soit n le nombre de couches. Les couches sont numérotées de 0 à $n-1$. nb_c est le nombre de neurones sur la couche c. $N_{c,i}$ est le neurone i de la couche c. Les neurones sont numérotés dans une couche de 1 à nb_c (sans compter le neurone virtuel). On ajoute à chaque couche un neurone virtuel $N_{c,0}$ (il n'a que des sorties, pas d'entrée). On note $w_{c,i,j}$ le poids de la connexion du neurone $N_{c-1,j}$ vers le neurone $N_{c,i}$.
On va noter $A_{c,i}$ l'activation du neurone $N_{c,i}$. On note $S_{c,i}$ la réponse du neurone $N_{c,i}$. On note $\varphi_{c,i}$ la fonction de tranfert du neurone $N_{c,i}$.

4.3.2 Fonctionnement

$\forall c \in [\![0..n-1]\!], S_{c,0} = 1.$

$\forall c \in [\![1..n-1]\!], \forall i \in [\![1..nb_c]\!], A_{c,i} = \sum_{j=0}^{nb_{c-1}} w_{c,i,j} S_{c-1,j}$ et $S_{c,i} = \varphi_{c,i}(A_{c,i})$.

$\forall i \in [\![1..nb_0]\!], S_{0,i} = i^{\text{ème}}$ composante de l'exemple courant.

4.3.3 Fonctionnement du réseau

On note X la situation courante.

Algorithm 5 Fonctionnement

for $c \leftarrow 0$ à $n-2$ **do**
  $S_{c,0} \leftarrow 1$ {Initialisation obligatoire des neurones virtuels}
end for
{Fonctionnement}
for $i \leftarrow 1$ à nb_0 **do**
  $S_{0,i} \leftarrow X_i$.
end for
for $c \leftarrow 1$ à $n-1$ **do**
  for $i \leftarrow 1$ à nb_c **do**
    $A_{c,i} \leftarrow \sum_{j=0}^{nb_{c-1}} w_{c,i,j} S_{c-1,j}$
    $S_{c,i} \leftarrow \varphi_{c,i}(A_{c,i})$

end for
end for
end

La sortie est $S_{n-1,[\![1..nb_{n-1}]\!]}$
On dispose d'une base d'apprentissage $T = \{(x(i), y(i))\}_{i=1}^{nb\_ex}$ avec $x(i) \in \mathbb{R}^{nb_0}$ et $y(i) \in \mathbb{R}^{nb_{n-1}}$.
Recherche des $w_{c,i,j}$ permettant de s'adapter au mieux à la base d'apprentissage T.

Algorithme de la rétro-propagation du gradient (back-prop) Soit

$$\mathbb{E} = \sum_{i=1}^{nb\_ex} E(x(i)) \text{ avec } E(X) = \sum_{i=1}^{nb_{n-1}} (Y_i - S_{n-1,i})^2$$

On effectue $\forall c, i, j, w_{c,i,j} \leftarrow w_{c,i,j} - \varepsilon \frac{\partial \mathbb{E}}{\partial w_{c,i,j}}$

1. Calcul de $\frac{\partial E}{\partial w_{n-1,i,j}}$:

$$\frac{\partial E}{\partial w_{n-1,i,j}} = \frac{\partial E}{\partial S_{n-1,i}} \frac{\partial S_{n-1,i}}{\partial A_{n-1,i}} \frac{\partial A_{n-1,i}}{\partial w_{n-1,i,j}} \tag{146}$$

On a

$$\frac{\partial E}{\partial S_{n-1,i}} = -2(Y_i - S_{n-1,i}) = 2(S_{n-1,i} - Y_i) \tag{147}$$
$$\frac{\partial S_{n-1,i}}{\partial A_{n-1,i}} = \varphi'_{n-1,i}(A_{n-1,i}) \tag{148}$$
$$\frac{\partial A_{n-1,i}}{\partial w_{n-1,i,j}} = S_{n-2,j} \tag{149}$$

2. Calcul de $\frac{\partial E}{\partial w_{c,i,j}}$ sous l'hypothèse que l'on a déjà calculé les $\frac{\partial E}{\partial w_{c',i,j}}$ avec $c' > c$.
On admet que

$$\frac{d}{dx} f(g_1(x), g_2(x), \ldots, g_n(x)) = \left\langle (\text{grad}_{g_1,\ldots,g_n}(f))(g_1(x), \ldots, g_n(x)) \middle| \begin{pmatrix} g'_1(x) \\ \vdots \\ g'_n(x) \end{pmatrix} \right\rangle \tag{150}$$

Exemple :
$f(x, y) \mapsto 2(x^2 + y)$ avec $g_1(x) : x \mapsto \ln(x)$ et $g_2(x) : x \mapsto \cos(x)$

$$\frac{df(\ln(x), \cos(x))}{dx} = \left\langle \begin{pmatrix} 4\ln(x) \\ 2 \end{pmatrix} \middle| \begin{pmatrix} \frac{1}{x} \\ -\sin(x) \end{pmatrix} \right\rangle = \frac{2}{x} - 2\sin(x)$$

Version 1

$$\frac{\partial E}{\partial w_{c,i,j}} = \frac{\partial E}{\partial S_{c,i}} \frac{\partial S_{c,i}}{\partial A_{c,i}} \frac{\partial A_{c,i}}{\partial w_{c,i,j}}$$

On a

$$\frac{\partial S_{c,i}}{\partial A_{c,i}} = \varphi'_{c,i}(A_{c,i})$$

et

$$\frac{\partial A_{c,i}}{\partial w_{c,i,j}} = S_{c-1,j}$$

et

$$\frac{\partial E}{\partial S_{c,i}} = \sum_{k=1}^{nb_{c+1}} \frac{\partial E}{\partial S_{c+1,k}} \frac{\partial S_{c+1,k}}{\partial S_{c,i}}$$

On calcule

$$\frac{\partial S_{c+1,k}}{\partial S_{c,i}} = \frac{\partial S_{c+1,k}}{\partial A_{c+1,k}} \frac{\partial A_{c+1,k}}{\partial S_{c,i}}$$

et

$$\frac{\partial S_{c+1,k}}{\partial A_{c+1,k}} = \varphi'_{c+1,k}(A_{c+1,k})$$

et

$$\frac{\partial A_{c+1,k}}{\partial S_{c,i}} = w_{c+1,k,i}$$

Version 2 On recommence cette opération en prenant :

$$\frac{\partial E}{\partial w_{c,i,j}} = \frac{\partial E}{\partial A_{c,i}} \frac{\partial A_{c,i}}{\partial w_{c,i,j}}$$

On a

$$\frac{\partial A_{c,i}}{\partial w_{c,i,j}} = S_{c-1,j}$$

et

$$\frac{\partial E}{\partial A_{c,i}} = \sum_{k=1}^{nb_{c+1}} \frac{\partial E}{\partial A_{c+1,k}} \frac{\partial A_{c+1,k}}{\partial A_{c,i}}$$

On a

$$\frac{\partial A_{c+1,k}}{\partial A_{c,i}} = \frac{\partial A_{c+1,k}}{\partial S_{c,i}} \frac{\partial S_{c,i}}{\partial A_{c,i}} = w_{c+1,k,i}\varphi'_{c,i}(A_{c,i})$$

3. Algorithme de calcul de $\frac{\partial E}{\partial w_{c,i,j}}$

Algorithm 6 Version 1

for $i \leftarrow 1$ à nb_{n-1} **do**
  $\frac{\partial E}{\partial S_{n-1,i}} \leftarrow 2(S_{n-1,i} - Y_i)$
end for
for $c \leftarrow n-2$ à 1 par pas de -1 **do**
  for $i \leftarrow 1$ à nb_c **do**
    $\frac{\partial E}{\partial S_{c,i}} \leftarrow \sum_{k=1}^{nb_{c+1}} \frac{\partial E}{\partial S_{c+1,k}} \varphi'_{c+1,k}(A_{c+1,k}) w_{c+1,k,i}$
  end for
end for
for $c \leftarrow 1$ à $n-1$ **do**
  for $i \leftarrow 1$ à nb_c **do**
    for $j \leftarrow 0$ à nb_{c-1} **do**
      $\frac{\partial E}{\partial w_{c,i,j}} \leftarrow \frac{\partial E}{\partial S_{c,i}} \varphi'_{c,i}(A_{c,i}) S_{c-1,j}$

end for
end for
end for

end

h] Version 2

for $i \leftarrow 1$ à nb_{n-1} **do**
$\frac{\partial E}{\partial A_{n-1,i}} \leftarrow 2(S_{n-1,i} - Y_i)\varphi'_{n-1,i}(A_{n-1,i})$
end for
for $c \leftarrow n-2$ à 1 par pas de -1 **do**
for $i \leftarrow 1$ à nb_c **do**
$\frac{\partial E}{\partial A_{c,i}} \leftarrow \varphi'_{c,i}(A_{c,i}) \sum_{k=1}^{nb_{c+1}} \frac{\partial E}{\partial A_{c+1,k}} w_{c+1,k,i}$
end for
end for
for $c \leftarrow 1$ à $n-1$ **do**
for $i \leftarrow 1$ à nb_c **do**
for $j \leftarrow 0$ à nb_{c-1} **do**
$\frac{\partial E}{\partial w_{c,i,j}} \leftarrow \frac{\partial E}{\partial A_{c,i}} S_{c-1,j}$
end for
end for
end for

end

Conclusion La version 2 est beaucoup plus rapide que la première.

4. h] Calcul de $\frac{\partial \mathbb{E}}{\partial w_{c,i,j}}$

$\frac{\partial \mathbb{E}}{\partial w} \leftarrow 0$
for $num \leftarrow 1$ à nb_ex **do**
Faire tourner le réseau avec comme entrée $x(num)$.
Par V2, calculer $\frac{\partial E}{\partial w}$ sachant que la sortie désirée est Y_i.
$\frac{\partial \mathbb{E}}{\partial w} + \leftarrow \frac{\partial E}{\partial w}$.
end for

end

5. Mise à jour des poids : $w \leftarrow w - \varepsilon \frac{\partial \mathbb{E}}{\partial w}$
6. Fonctionnement global.

Algorithm 9 Fonctionnement global
Créer la machine et l'initialiser
repeat
Calcul de $\frac{\partial \mathbb{E}}{\partial w}$
Mise à jour de w
until l'apprentissage soit suffisant

end

7. Choix de $\varphi_{c,i}$. On choisit des fonctions de type sigmoïdes (elle doit être strictement croissante, bornée, C^∞).
Exemple fonctions sigmoïdes :

- $\varphi(x) = \frac{1}{1+e^{-x}}$ (mauvaise).
- $\varphi(x) = \frac{2}{1+e^{-x}} - 1$ (bonne).
- $\varphi(x) = \frac{2}{\pi}\arctan(x)$ (longue).

8. Initialisation.

   (a) Aléatoire :
   - w initialisé en aléatoire uniforme dans $[-1, 1]$.
   - w initialisé selon une distribution gaussienne centrée en 0 de variance 1.

A Dérivations usuelles

–

$$\text{grad}_u(\lambda u) = \begin{pmatrix} \lambda \\ \lambda \\ \vdots \\ \lambda \end{pmatrix}$$

–

$$\text{grad}_u(u^t v) = \text{grad}_u\left(\sum_{i=1}^{n} u_i v_i\right) = \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{pmatrix} = v$$

– Soit M une matrice symétrique,

$$\text{grad}_u(u^t M u) = ???$$

On a

$$Mu = \left(\sum_{k=1}^{n} M_{i,k} u_k\right)_{i \in [\![1..n]\!]} \Rightarrow u^t M u = \sum_{i=1}^{n} u_i \sum_{k=1}^{n} M_{i,k} u_k = \sum_{(i,k) \in [\![1..n]\!]^2} u_i M_{i,k} u_k$$

On a donc

$$\begin{aligned}
\frac{\partial u^t M u}{\partial u_l} &= \frac{\partial}{\partial u_l}\left[\sum_{\substack{i \neq l \\ k \neq l}} u_i M_{i,k} u_k + \sum_{\substack{i = l \\ k \neq l}} u_i M_{i,k} u_k + \sum_{\substack{i \neq l \\ k = l}} u_i M_{i,k} u_k + \sum_{\substack{i = l \\ k = l}} u_i M_{i,k} u_k\right] \\
&= \frac{\partial}{\partial u_l}\left[u_l \sum_{\substack{k=1 \\ k \neq l}} M_{l,k} u_k + u_l \sum_{i \neq l} u_i M_{i,l} + u_l M_{l,l} u_l\right] \\
&= \sum_{k \neq l} M_{l,k} u_k + \sum_{i \neq l} M_{i,l} u_i + 2 M_{l,l} u_l \\
&= \sum_{k \neq l} M_{l,k} u_k + \sum_{i \neq l} M_{l,i} u_i + 2 M_{l,l} u_l \quad (M \text{ est symétrique}) \\
&= 2 \sum_{k \neq l} M_{l,k} u_k + 2 M_{l,l} u_l \\
&= 2 \sum_{k} M_{l,k} u_k
\end{aligned}$$

Finalement

$$\text{grad}_u(u^t M u) = 2 \begin{pmatrix} \sum\limits_k M_{1,k} u_k \\ \sum\limits_k M_{2,k} u_k \\ \vdots \\ \sum\limits_k M_{n,k} u_k \end{pmatrix} = 2Mu$$

–

$$\text{grad}_u(u^t u) = \text{grad}_u(u^t I u) = 2Iu = 2u$$