Département Traitement du Signal et des Images

SI 221 : Reseaux de neurones

Laurence Likforman, Chloé Clavel, Giovanna Varni

3 Octobre 2017

Site pédagogique du module SI221

Site pédagogique du module SI221, Onglets “Ressources en ligne”, puis “TP Perceptron”.

1 Description des données

1.1 Données LANDSAT sur Tarascon

Ce sont les données dans BaseImages.zip du TP “Réseaux de neurones” et elles sont accessibles à partir du site pédagogique (en fichiers isolées ou en .zip). La taille du pixel est de 30m:

0.45-0.52 μ	`landsattarasconC1.ima`
0.52-0.60 μ	`landsattarasconC2.ima`
0.63-0.69 μ	`landsattarasconC3.ima`
0.75-0.90 μ (Proche InfraRouge : PIR)	`landsattarasconC4.ima`
2.08 2-35 μ (Moyen InfraRouge : MIR)	`landsattarasconC5.ima`
10.40-12.50 μ (infra rouge thermique)	`landsattarasconC6.ima`
1.55-1.75 μ (Moyen InfraRouge : MIR)	`landsattarasconC7.ima`
0.52-0.90 μ (panchromatique)	`landsattarasconC8.ima`

C’est sur ce jeu de données que vous commencerez vos travaux.

2 Quelques ordres Matlab

2.1 Lecture sous Matlab des données Landsat

Chaque canal est archivé séparément au standard de TSI (image 8 bits, extension `.ima`, associé à un fichier ASCII, extension `.dim` contenant le nombre de colonnes et le nombre de lignes).

Pour lire ce type de données sous Matlab, on dispose d’une procédure (`.m`) pour lire les images `ima2mat` dont la syntaxe, appliquée à la donnée `cam1` est la suivante :

```
im1 = ima2mat('cam1');
```

L’image est alors accessible en tant que tableau Matlab `im1`. Les dimensions de cette matrice peuvent se récupérer ainsi :

```
nlig = size(im1,1)
ncol = size(im1,2)
```

2.2 Visualisation d’une image

- Visualiser une image peut s’effectuer avec l’odre `image`

`image(im1);`

Mais ATTENTION : `im1` est une "image Matlab", c'est à dire entre 0 et 255.

- une image en niveau de gris doit utiliser la colormap `gray` :

  `colormap(gray)`

  Cette colormap par défaut est sur 100 niveaux de gris.
  Une image en 256 niveaux de gris nécessite une colormap sur 256 niveaux :

  `colormap(gray(256))`

  Si l'on veut se contenter de nn niveaux :

  `colormap(gray(nn))`

- Une image de labels (indice de classe) peut se visualiser avec la colormap `flag` (4 couleurs)

  `colormap(flag)`

  ou `prism` (8 couleurs).

  `colormap(prism)`

Attention: pour Matlab, une image est un entier compris entre 0 et 255. Si ce n'est pas le cas (image quelconque en flottant), il faudra recentrer les données (voir 2.4).

2.3 Statistiques et histogramme d'une image

- La moyenne d'une image `im1` (matrice 2-D) est donnée par `mean(im1(:))`
- la valeur minimale est donnée par : `min(im1(:))`
- la valeur maximale par `max(im1(:))`.

L'histogramme d'une image (`im1`) peut se calculer directement avec `hist` :

`hist(im1(:))`

ou, sur 256 niveaux par pas de 1 entre 0 et 255 :

`hist(im1(:),0:1:255)`

Cela donne des résultats différents : méfiez vous de `hist` utilisé sans argument complémentaire et consultez le "help" en ligne.

2.4 Bi-histogramme : scatterplot

Soient deux tableaux monodimensionnels `A` et `B`, représentant par exemple un pixel dans une image. On va représenter dans un plan chaque pixel, l'abscisse étant la valeur du tableau `A` et l'ordonnée la valeur du tableau `B`. On utilise alors la procédure `scatter`:

```
scatter( A, B);
```

Si l'on veut faire varier la taille du point, on peut écrire :

```
scatter( A, B, 5);
```

Si l'on a un troisième tableau `C` correspondant à la classe du pixel (valeur 1 ou 2), on peut changer la couleur du point :

```
scatter( A, B, 5, C);
colormap('flag');
```

2.5 Recadrage d'une image

Si on veut afficher une matrice quelconque M, il faut donc la transformer en "image matlab" I. Par exemple, on peut définir I comme suit :

```
I  = 255 * (M - min(M(:))) / (max(M(:)) - min(M(:))) ;
```

2.6 Création d'une fonction Matlab

Une fonction Matlab a la structure suivante:

```
function [out1, out2, ...] = myfonc(in1, in2, ...)

[out1, out2, ...] = %calcules

end
```

Elle va enregistrée dans un fichier `.m` ayant le meme nom de la fonction, dans ce cas le fichier s'appelera `yfonc.m`.

2.7 Retrouver les dimensiones d'une matrice

On utilise `size`:

```
a=ima2mat(‘landsat_tarascon_C1.ima’);
v=size(a);
nlig=v(1) % nombre de lignes
ncol=v(2) % nombre de colonnes
% on peut aussi opérer ainsi
nlig=size(a,1);
ncol=size(a,2);
```

2.8 Transformation d'une matrice en vecteur et vice-versa

Pour transformer une matrice `A` de taille N x M en vecteur `B` de taille NM x 1 : `B = A(:);` Cette commande place les colonnes de la matrice `A` les unes à la suite des autres dans un grand vecteur colonne.

Inversement, pour récupérer la matrice `A` à partir du vecteur `B` (il faut bien sur avoir calculé la taille de `A` auparavant !) : `A = reshape(B;N;M);`

2.9 Création d'une image de label

Pour créer une image de label, on peut utiliser les ordres logiques Matlab suivant, expliqués sur les exemples suivants. Soit un vecteur `classe`. On veut créer un vecteur de résultat `proto` ayant la valeur 1 si l' élément de classe est égal à 3, et la valeur 0 sinon :

```
proto = classe==3
```

Soit une image `ima`. On veut créer un vecteur de résultat `proto` ayant la valeur 1 si l' élément de `classe` est supérieur à 80, et la valeur 0 sinon :

```
proto = ima>80 ;
```

Soit une image `ima`. On veut créer un vecteur de résultat `proto` ayant la valeur 1 si l'élément de `classe` est compris entre 80 et 120, et la valeur 0 sinon :

```
proto = (ima>80) .* (ima<120) ;
```

Attention aux parenthèses dans cet exemple !!

2.10 Miscellanea

```
A = ones(512,1);
B=[1:512]';
C=[A B]; % matrice de 512 lignes et 2 colonnes
```

3 Réaliser un perceptron sous Matlab

Le but de ce TP est d'écrire un perceptron pour bien assimiler les caractéristiques de cette règle d'apprentissage très simple et effcace en cas de séparation linéaire.

3.1 Rappel de la règle du perceptron

Le but est de coder le perceptron, c'est à dire de trouver un vecteur $\tilde{W} = (w_1, w_2, .., w_n)^t$ tel que pour toute entrée $\tilde{X}$ on ait :

$$\tilde{W}^t \tilde{X} \ = \ s$$

et tel que cette sortie s soit supérieure à un seuil b pour la classe 1 et inférieure à ce seuil b pour la classe 2.

Le vecteur d'état $\tilde{X}$ est de dimension n. On opère alors avec le vecteur Y de dimension $n+1$ construit par la relation ;

$$Y = \begin{pmatrix} x_1 \\ x_2 \\ ... \\ x_n \\ 1 \end{pmatrix}$$

De ce fait, on utilise dans le perceptron le vecteur W de dimension $n+1$ défini par :

$$W = \begin{pmatrix} w_1 \\ w_2 \\ ... \\ w_n \\ -b \end{pmatrix}$$

La sortie du perceptron, $W^t Y$, s'analyse alors simplement par son signe : positif pour la classe 1, négatif pour la classe 2.

La règle du perceptron est de modifier, à l'étape t, ce vecteur $W(t)$ chaque fois que la classification est erronée. Ce qui donne :

$$W(t+1) \ = \ W(t) \pm \eta Y$$

le signe dépendant de la classe. Le paramètre η, théoriquement égal à 1, permet d'"adoucir" le processus d'apprentissage comme tout paramètre utilisé par exemple dans une descente de gradient.

On appellera `époque` le fait d'utiliser une fois la totalité des pixels de l'image.

3.1.1 Etude d'un seuil sur image

Pour bien comprendre ces choix (signe de η, ...), vous allez ici utiliser une image monocanal et voir comment un perceptron peut apprendre à classer chaque pixel selon deux classes (eau et terre).

Mise en place du corpus d'apprentissage – attribution des labels à l'image landsattarasconC4.ima

En vous aidant des outils Matlab présentés précédemment, faites un programme qui attribue une classe de *référence* à chaque pixel de l'image `landsattarasconC4.ima` selon le

critère suivant : toute valeur de pixel inférieure à 30 est dans la classe 1, les autres pixels sont dans la classe 2. *Nous nous plaçons donc dans le cas linéairement séparable et votre objectif sera d'apprendre à déterminer ce seuil grâce au perceptron.*

Apprentissage du perceptron – la théorie

Après avoir relu attentivement la Section 3.1, étant donné que vous connaissez le seuil que vous cherchez,

- faire un schéma en dimension 2 pour comprendre quel vecteur W il faudra trouver;
- comprendre le sens dans lequel on doit opérer sur le vecteur W quand la variable est mal classée.

Apprentissage du perceptron – implémentation

A partir de l'image labellisée que vous avez construit précédemment et qui vous sert ici de corpus d'apprentissage :

- écrire un algorithme simplifié balayant l'image classiquement (balayage video) avec $\eta = 0.01$;
- voir combien d'époques il lui faut pour converger;
- donner le taux de bonne classification et de mauvaise classification sur ce corpus d'apprentissage;
- vérifier que le vecteur est bien celui recherché : on fera le raisonnement à partir du schéma graphique initial;
- changer la valeur de η et observer ce que cela implique.

Il faut alors se poser les questions :

- est-il normal que ce classifieur donne manifestement toujours le bon résultat ?

3.1.2 Effet d'une erreur sur une valeur

On va modifier le fichier des labels de l'image utilisée en apprentissage en attribuant le label des pixels ayant la valeur 110 comme étant celui de la classe 1 : cela reviendrait à avoir un capteur "aveuglé" pour la valeur 110.

- Combien de pixels sont concernés par ce changement ?
- Faire tourner l'algorithme en mettant (si ce n'était pas le cas) un seuil au nombre d'époque. A-t-on le même résultat ?
- Donner le taux de bonne classification et de mauvaise classification;
- essayez de comprendre ce qui se passe dans les différentes étapes de l'apprentissage.

3.1.3 Effet d'une erreur "fatale" de saturation

On va modifier le fichier des labels de l'image utilisée en apprentissage en attribuant le label des pixels ayant une valeur supérieure à 140 comme étant celui de la classe 1 : cela reviendrait à avoir un capteur "aveuglé" pour toutes les valeurs entre 141 et 255.

- Combien de pixels sont concernés par ce changement ?
- Faire tourner l'algorithme en mettant (si ce n'était pas le cas) un seuil au nombre d'époque. A-t-on le même résultat ?
- Donner le taux de bonne classification et de mauvaise classification;
- essayez de comprendre ce qui se passe dans les différentes étapes de l'apprentissage.

3.1.4 Recherche d'une solution de type perceptron à un problème donné

Vous avez sur le site un masque à deux classes `tarascon_2Classes.ima`. Ce masque est en fait une sorte de vérité terrain de classification non garantie (obtenue avec un k-moyennes). Est-il possible d'apprendre à partir de ce masque un perceptron donnant une solution acceptable ?