3iLINGENIEURS
</les sciences informatiques>

Travaux Pratiques

2eme année

Intelligence Artificielle : Modèle de Markov Caché

Module Robotique et Intelligence Artificielle

2020 - 2021

Reconnaissance de Formes par Vision Artificielle :

Application à la Reconnaissance de chiffres manuscrits

I. Application à la Reconnaissance de chiffres manuscrits isolés

I.1- Objectifs :

- Appréhender les différentes étapes qui composent un système de reconnaissance depuis les prétraitements (localisation / extraction des formes à reconnaître) jusqu'à la reconnaissance.
- Concevoir et développer un outil de reconnaissance de chiffres manuscrits basé sur le Modèle de Markov Cachée.

I.2- Travail demandé :

Développer, sous MATLAB, un système complet de reconnaissance de chiffres manuscrits dont le schéma fonctionnel sera le suivant :

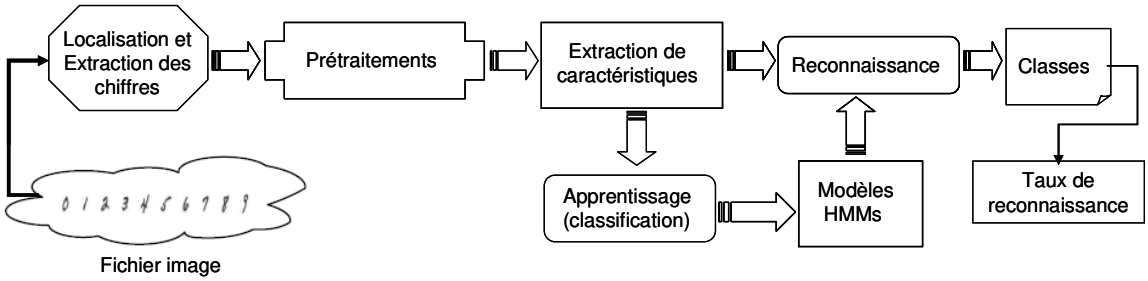


***Figure 1** : Système de reconnaissance : Procédure d'apprentissage et de reconnaissance.*

I.3- Fichiers fournis et les données mises à votre disposition

Les fichiers fournis sont à récupérer sur le réseau : `moodle.3il.fr/mod/resource/view.php?id=4039`

1. Librairie HMM utilisée (Mise en œuvre des HMM)

Pour implémenter des modèles de Markov Cachés discrets en Matlab, vous allez utiliser certaines fonctions de la ***Toolbox HMM*** mise à votre disposition. Ces fonctions sont regroupées dans le répertoire "`Library\HMMToolbox`".
La toolbox implémente les algorithmes suivant :

- *Forward* et *Forward-Backward* pour le calcul de la vraisemblance de chaque modèle à l'aide des fonctions **`forward()`**, **`forwards_backward()`** et **`log_lik_dhmm()`**.
- *Viterbi* pour le calcul du meilleur chemin, grâce à la fonction **`viterbi_path()`**.
- *Baum-Welch* pour l'apprentissage (entrainement) d'un Modèle de Markov caché discret, grâce à la fonction **`learn_dhmm()`**.

2. Base de données utilisée pour l'entrainement

Pour l'apprentissage de notre système de reconnaissance, nous avons utilisé la base de données MNIST. Cette dernière est une base standard, publique, composée de *70000 images* de chiffres manuscrits extraits de la base NIST et déjà pré-traités. La base est divisée en une base d'apprentissage (*60000 images*) et une base de test (*10000 images*).
Les spécificités de la base de données sont :

- les images sont codées sur 8 bits, et ont la même taille (28×28),
- les caractères sont normalisés et centrés
- les bases d'apprentissage et de test ont été écrites par des scripteurs distincts.

3. Interface Matlab

Une interface écrite en Matlab est mise à votre disposition (dans le répertoire `\TP-IA-HMM\Pattern Recognition`) pour tracer des séquences de points représentant des chiffres en utilisant la souris. Le chiffre tracé sera ensuite reconnue en utilisant les modèles de Markov cachés.

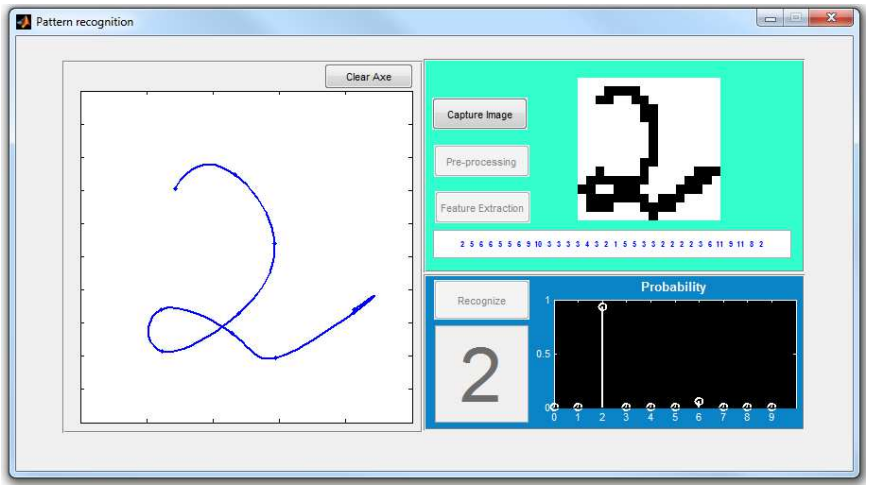


***Figure 2**: Système de reconnaissance*

Cette interface comprend les fichiers « **`OptDigit.fig`** » et « **`OptDigit.m`** » pour définir la fenêtre de l'application.

- Le bouton « ***Capture Image*** » fait appel à la fonction « **`imageAcquisition.m`** » pour tracer (à l'aide de la souris) et afficher l'image du chiffre à reconnaître.
- Le bouton « ***Pre-processing*** » fait appel à la fonction «`[imBW, imData] =` **`imagePreprocessing`**`(imdata, mrows, ncols)`» pour effectuer le prétraitement nécessaire (*binarisation*, *recadrage*, *redimensionnement* et *normalisation*).
- Le bouton « ***Feature Extraction*** » réalise l'extraction des caractéristiques ou des primitives de l'image du chiffre en faisant appel à la fonction « `features = imageFeaturesExtraction(imBW)` ».
- Le bouton « ***Recognize*** » utilise la fonction «`probabilities = OptDigitsRecognition(imData, features, HMMmodel)` » pour reconnaître le chiffre écrit dans la zone de dessin à l'aide les modèles de Markov cachés. Le fichier `HMMmodel.mat` (dans le répertoire `\MatFile`) contient les Modèles de Markov entrainés pour les 10 classes de chiffres.

I.4- Mise en pratique

A. Configuration du chemin d'accès

Etape 1 : Avant toute chose, il faut préciser à Matlab la liste des répertoires dans lesquels se trouvent les différentes fonctions (fichiers *m-files*) que vous allez utiliser. Pour cela :

- allez dans le répertoire `\TP-IA-HMM`
- exécuter la fonction **`init.m`** en tapant la commande «**`init`** »

NB : Il faudra refaire les mêmes procédures à chaque début de session Matlab.

B. Implémentation d'outil de reconnaissance

D'une façon générale, l'apprentissage et la reconnaissance d'un caractère manuscrit suivent les étapes illustrées sur la figure 1 :

- *Localisation et Extraction des chiffres ;*
- *Prétraitements ;*
- *Extraction des caractéristiques.*

1. Localisation et Extraction des chiffres (segmentation).

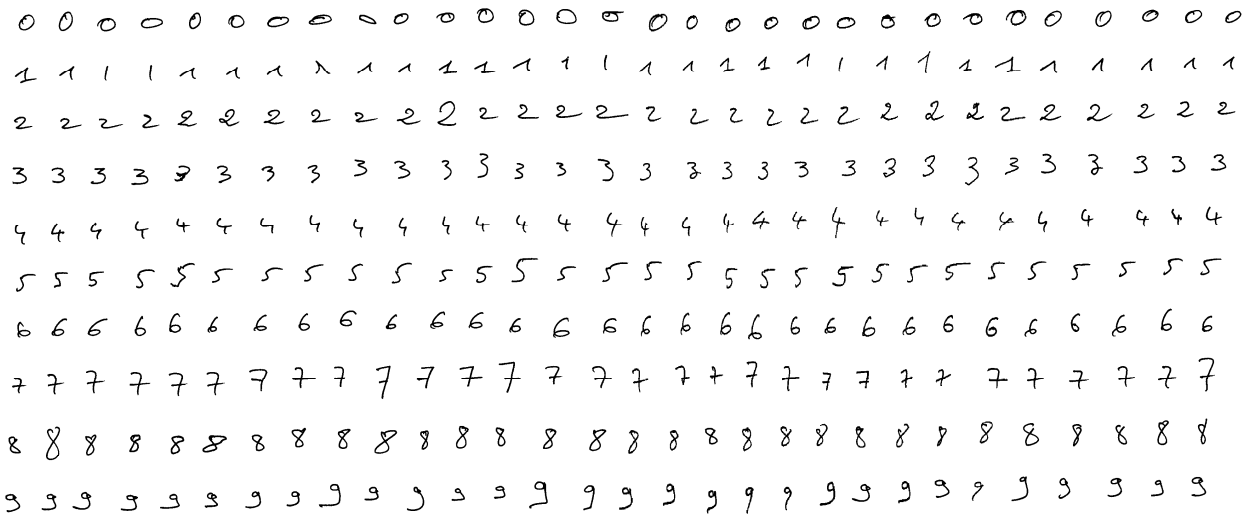
Pour cette étape, une base de données de chiffres manuscrits est mise à votre disposition dans le répertoire « `TP-IA-HMM\HMM_Modelisation\Fichier_BDD` ».

a) Base de données (BDD)

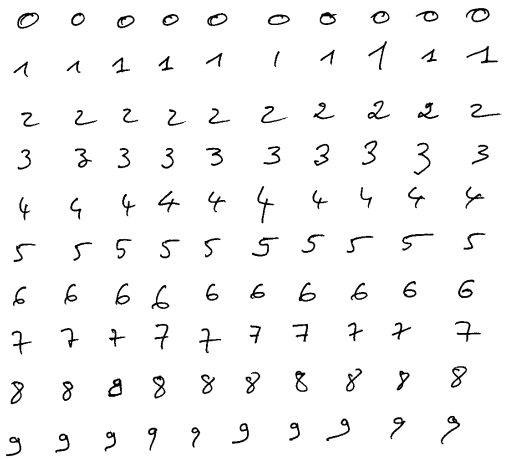
Cette base est composée de deux fichiers images « *train.tif* » et « *test.tif* ». La base est donc découpée en un ensemble d'***apprentissage*** et un ensemble de ***test***.

Les chiffres contenus dans le fichier « *train.tif* », une fois extraits serviront de base d'apprentissage et les chiffres contenus dans le fichier « *test.tif* » serviront à tester les performances du système de reconnaissance que vous allez développer.

- Le fichier « *train.tif* » contient 400 chiffres manuscrits. Pour chaque type de chiffre, vous avez 40 exemplaires de référence :



- Le fichier « *test.tif* » contient un ensemble de 100 chiffres manuscrits (10 exemples par classe) :



b) *Segmentation et extraction des chiffres de la base*

Le répertoire «`HMM_Modelisation`» sera votre répertoire de travail pour la mise en place de l'outil de reconnaissance de chiffre manuscrit.

Etape 2 : Exécuter le script Matlab « `Main.m` » pour lire et afficher le contenu des fichiers images « *train.tif* » et « *test.tif* ».

➔ Etudier le code et commenter chaque ligne. A quoi correspondent les deux figures ?

Etape 3 : La fonction suivante :

`Images = Localisation_extraction(filename)`

Permet de localiser et d'extraire tous les chiffres manuscrits contenus dans un fichier image passé en paramètre **`"filename"`**. La fonction renvoie un tableau de cellule (*cell array*) **`Images`** (de dimension *NxM*, où *N* représente le nombre de classe et *M* le nombre d'exemple par classe) qui contient les exemplaires de chaque classe de chiffres.

NB : Cette fonction utilise la fonction **`[ind1,ind2,l] = imageLocalisation(imdata,dim)`** qui recherche à partir de l'histogramme de projections horizontales (**`dim = 'h'`**) ou verticales (**`dim = 'v'`**) de pixels noir, les plages

correspondant à un nombre de pixels noirs non nul. Le début et la fin de chaque plage détectée sur l'histogramme sont respectivement **`ind1`** et **`ind2`**.

- l'histogramme de projections horizontales permettent d'isoler les lignes de texte les unes des autres.
- l'histogramme de projections verticales calculées sur une ligne de texte isolée et extraite, permettent la séparation des chiffres qui compose la ligne.

➔ Compléter la fonction :

```
l = histogramme_projection(imdata,dim)
```

Permettant de rechercher à partir de l'histogramme de projections horizontales (**`dim='h'`**) ou verticales (**`dim='v'`**) des pixels noirs dans une image en noir et blanc **`imdata`**. Cette fonction renvoie un vecteur **`l`** qui correspond à l'histogramme de projection de cette image **`imdata`**.

➔ Compléter le script Matlab « `Main.m` » afin :

- d'afficher l'histogramme de projections verticales de la BDD d'apprentissage « *train.tif* ». A quoi correspondent les pics ?
- d'afficher l'histogramme de projections horizontales de la BDD d'apprentissage « *train.tif* ». A quoi correspondent les pics ? Expliquer comment allez-vous procéder pour isoler chaque ligne de la BDD.
- d'isoler la première ligne du fichier « *train.tif* », afficher son l'histogramme de projections verticales et réaliser l'extraction du premier chiffre de la ligne.

➔ Appliquer la fonction **`Localisation_extraction`** à la localisation des chiffres dans les deux images binaires fournies. Sauvegarder les résultats dans des fichiers « *.mat* », dans le répertoire « `MatFile` » sous le nom « *train.mat* » pour la BDD d'apprentissage et sous le nom « *test.mat* » pour la BDD de test

➔ Utiliser la fonction **`imageShow(Images)`**, pour vérifier que les chiffres ont été bien localisés et récupérés.

Par exemple si on veut afficher le premier exemplaire du chiffre 0, il faut taper : **`imageShow(Images{1,1})`**, pour le chiffre 1, il faut taper : **`imageShow(Images{2,1})`**, etc ...

Etape 4 : Quels sont les avantages et les inconvénients de cette méthode de localisation par projections ?

2. Prétraitements :

a) Binarisation

La binarisation permet de passer d'une image de couleur ou de niveaux de gris à une image binaire composée de 2 valeurs 0 et 1 (noir et blanc), plus simple à traiter.

Chaque image extraite de la BDD, doit être binarisée à l'aide d'une opération de seuillage. Chaque pixel de l'image est comparé à ce seuil et prend la valeur 0 ou 1 selon qu'il est supérieur ou inférieur.

Etape 5 : Compléter la fonction :

```
imgBW = imageBinarisation(imdata, seuil)
```

Pour convertir une image de niveaux de gris **`imdata`** à une image binaire **`imgBW`**. La valeur du fond est égale à 0 et la valeur des pixels décrivant le chiffre est égale 1 en fonction de la valeur **`seuil`**.

➔ Compléter le script Matlab « `Main.m` » afin :

- Affichez l'image *rice.png* avant et après binarisation à l'aide de la fonction **`imshow()`**.
- Affichez l'image *printedtext.png* avant et après binarisation à l'aide de la fonction **`imshow()`**. Quel est le seuil optimal ?

➔ Quels sont les avantages et les inconvénients de cette méthode ?

➔ Proposer une autre méthode de binarisation.

b) Recadrage, Redimensionnement ou Normalisation

❖ ***Recadrage***

Etape 6 : Utiliser la fonction :

```
bw_crop = imageCropping(imgBW)
```

pour effectuer le recadrage de l'image binaire **`imgBW`** passée en paramètre.

Figure 3 *: Image avant et après recadrage*.

➔ Expliquer comment fonctionne cette fonction ?
➔ Affichez l'image du chiffre 1 avant et après recadrage à l'aide de la fonction **`imageShow()`**.
➔ Proposer une autre méthode (solution) permettant de recadrer une image binaire.

❖ La ***normalisation ou le redimensionnement*** permet de standardiser les dimensions des données écrites en procédant à la normalisation de l'image en une taille fixée.

Etape 7 : Utiliser la fonction :

```
imBW_Resize = imageResize(bw_crop,mrows,ncols)
```

pour redimensionner l'image du chiffre recadrée **`bw_crop`**. On choisira une taille fixe (16 × 16) pour toutes les images des chiffres.

➔ Comment fonctionne-t-elle ?
➔ Affichez l'image du chiffre 3 avant et après normalisation à l'aide de la fonction **`imageShow()`**. Quelle est la taille de l'image avant et après normalisation ?

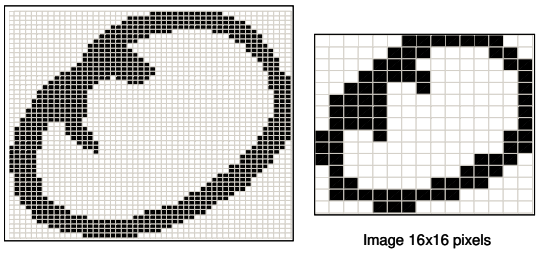


Figure 4*: Image avant et après normalisation*

Etape 8 : Que fait la fonction :

```
[imBW, imData] = imagePreprocessing(imdata, mrows, ncols)
```

3. Extraction des caractéristiques ou des primitives

L'extraction des primitives a pour rôle de convertir une image de chiffre en une suite de symboles issus d'un alphabet fini de symboles.

Pour chacun des chiffres prétraités, nous allons extraire les caractéristiques. Pour faire simple, nous choisissons d'extraire leurs histogrammes de projections horizontales et verticales.

Etape 9 : Compléter la fonction :

```
observation = imageFeaturesExtraction(imBW)
```

Etape 10 : Compléter le script Matlab (**`FeatureExtraction`**) pour extraire les caractéristiques de l'ensemble des chiffres prétraité de la base d'apprentissage « *train.tif* ».
Le résultat sera sauvegardé dans un fichier « *.mat* », dans le répertoire «`MatFile`» sous le nom « *features.mat* ». La variable contenant les caractéristiques est un tableau de cellule (*cell array*) de dimension *MxN*, où *N = 10,* représentent le nombre de classe et *M* = 40, le nombre d'exemple par classe (par exemple : `features{M}{N}`).

➔ Décrire le fonctionnement de ce script. Il y a deux boucles dans ce dernier, à quoi servent-elles ?

I.5- Apprentissage et Reconnaissance

La phase d'apprentissage tentera de tirer de cette analyse une généralisation des caractéristiques extraites de la base d'apprentissage que le processus de reconnaissance utilisera pour prendre une décision.

Pour modéliser un problème de reconnaissance de chiffres manuscrits à l'aide des HMMs :

- Il faut d'abord définir une topologie de HMM
- Chacune des 10 classes des chiffres sera caractérisée par un modèle de HMM représenté par son *vecteur de probabilité initiale* (π), sa *matrice de transition* (A) et sa *matrice d'observation* (B).
- La phase d'apprentissage permet d'associer un modèle à chaque classe et l'entraîner avec les échantillons d'apprentissage.
- L'apprentissage agira sur les paramètres du modèle (A, B, et π), de façon à maximiser la vraisemblance de l'ensemble des observations générées pour une même classe. Permettant ainsi d'obtenir le meilleur modèle HMM pour chacune des 10 classes.
- Une fois l'apprentissage réalisé, on devrait disposer donc de 10 modèles de chiffres.
- La phase de reconnaissance : étant donné une observation, évaluer la probabilité qu'elle soit engendrée par chacun des 10 modèles et sélectionner celui qui est le plus probable.

1. Topologie de réseau des états

Deux topologies des HMMs peuvent être utilisées pour modéliser le problème reconnaissance des formes: le ***modèle ergodique*** et le ***modèle gauche-droite***.

Etape 11 : Rappeler la définition du ***modèle ergodique*** et du ***modèle gauche-droite*** (séquentiel et parallèle).

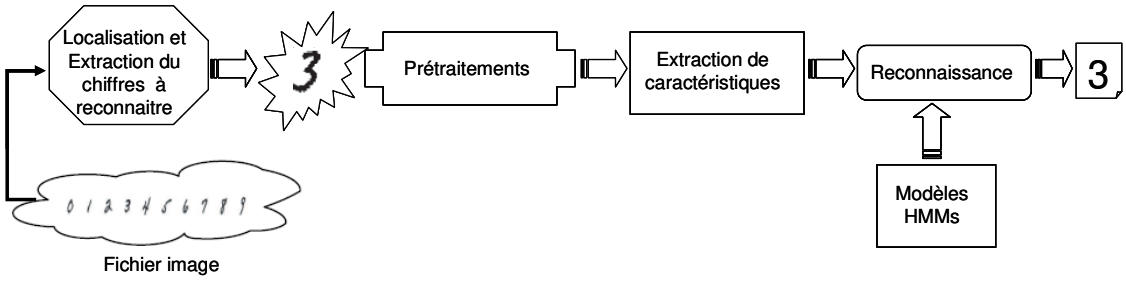
2. Apprentissage (classification)

Etape 12 : Utiliser le script Matlab (**`mainHMM`**) pour réaliser l'apprentissage de chacune des 10 classes et les sauvegarder dans un fichier d'apprentissage (***`hmmModel.mat`***) dans le répertoire « `MatFile`».

Ce script **`mainHMM`** utilise les fonctions **`train_HMM()`** et **`learn_dhmm()`** pour entrainer les Modèle de Markov caché, grâce à la l'algorithme de *Baum-Welch* de la *toolbox HMM*.

➔ Décrire le fonctionnement de ce script. A quoi correspondent les paramètres d'entrée de la fonction **`train_HMM`** ?

3. Reconnaissance



En phase de reconnaissance, on confronte l'image du chiffre à identifier à tous les modèles de chiffres :

- On commence par mener sur cette image la même extraction de caractéristiques que celle utilisée durant l'apprentissage.
- On évalue ensuite la *vraisemblance de l'observation* ainsi obtenue pour chacun des 10 modèles.
- Le modèle ayant retourné la plus grande valeur de vraisemblance fournira l'étiquette du chiffre à reconnaître

La vraisemblance de chaque modèle sera calculée à l'aide de *l'algorithme forward* grâce à la fonction **`forward()`** de la *toolbox HMM*.

Etape 13 : Que fait la fonction :

```
probabilities = imageRecognition(obs, hmmModel)
```

Etape 14 : Compléter le script Matlab **`HMM_recognize`** et tester les performances des modèles ainsi appris pour effectuer la reconnaissance de chiffres manuscrits (utiliser la ***BDD de test*** et le script Matlab **`HMM_recognize`**).

I.6- Performance

Etape 15 : Compléter le script **`Performance`** et évaluer les performances du système de reconnaissance en terme de :

- *la matrice de confusion*
- *taux de reconnaissance*
- *taux d'erreur*

Etape 16 : En déduire le nombre d'images bien classées et le nombre d'images mal classées.

Etape 17 : Discuter sur les chiffres bien et mal classés.

Etape 18 : Evaluer l'influence du nombre d'états cachés des chaînes de Markov sur les performances en reconnaissance. Conclure.

Etape 19 : Evaluer l'influence du type de HMM sur les performances en reconnaissance. Conclure.

Etape 20 : Discuter du choix des caractéristiques et notamment de l'influence sur les performances du classifieur.