



Master 2 ATAL

Signal et Langue

---

# Digit Recognizer

---

*Étudiants :*

Coraline MARIE  
Romain RINCÉ

*Encadrants :*

Harold MOUCHÈRE  
Christian VIARD-GAUDIN

15 novembre 2014



UNIVERSITÉ DE NANTES

## Table des matières

|   |          |
|---|----------|
| <b>Introduction</b>                               | <b>2</b> |
| <b>1 Hidden Markov Models</b>                     | <b>3</b> |
| 1.1 Extraction de séquences . . . . .             | 3        |
| 1.2 Création et initialisation d'un HMM . . . . . | 5        |
| 1.3 Entraînement des modèles . . . . .            | 5        |
| 1.4 Performances de la reconnaissance . . . . .   | 6        |
| <b>2 Réseaux de neurones</b>                      | <b>8</b> |
| <b>Conclusion</b>                                 | <b>9</b> |

## **Introduction**

# 1 Hidden Markov Models

## 1.1 Extraction de séquences

Pour cette première partie, seul quelques extraits de codes seront présentés. L'intégralité du code source se trouve dans le fichier `src/hmmPart1.r` du projet.

### Extension de la fonction `simu_symbol`

La fonction `simu_symbol` permet de construire et d'afficher des chiffres. Pour cela, elle récupère un ensemble de 30 points différents, qu'elle relie ensuite dans un ordre précis. L'extension de cette fonction par le code ci-dessous, permet désormais de construire le chiffre 6.

```

1 simu_symbol <- function()
2 {
3   (...)
4   digit_6 <- rbind(
5     stroke(-0.5,1.0,-0.5,-0.5,10),
6     stroke(-0.4,-0.5,0.5,-0.5,7),
7     stroke(0.5,-0.4,0.5,0.4,6),
8     stroke(0.4,0.4,-0.5,0.4,7)
9   )
10  dimnames(digit_6) <- list(num=1:nrow(digit_6),point=c("x","y"))
11  plot(digit_6,type="l",col="red",xlim=c(-1,1),ylim=c(-1,1))
12  points(digit_6)
13  return(list(d1=digit_1,d2=digit_4,d3=digit_6))
14 }
```

### Analyse de la fonction `compute_symbol`

La fonction `compute_symbol` prend trois paramètres en entrée : le tracé d'un chiffre, un nombre de lignes (5) et un nombre de colonnes (3). Elle commence par construire une matrice, ayant pour dimension le nombre de lignes et de colonnes précisés précédemment. Cette matrice servira à positionner les points du tracé analysé :

$$\begin{pmatrix} 13 & 14 & 15 \\ 10 & 11 & 12 \\ 7 & 8 & 9 \\ 4 & 5 & 6 \\ 1 & 2 & 3 \end{pmatrix}$$

Elle calcule ensuite le nombre de points présents dans le tracé du chiffre, puis les positions horizontales et verticales de chacun de ces points. Pour finir, elle détermine la position de chaque point dans la matrice qu'elle a créée, et les affiche.

Voici les résultats donnés par cette fonction pour les trois tracés créés par la fonction `simu_symbol` :

```

1  pour 1 : 11 11 14 14 14 14 14 14 14 14 14 14 14 14 11 11 11 11 8
      8 8 8 5 5 5 5 2 2 2 2
2  pour 4 : 14 14 14 11 11 10 7 7 7 4 4 4 4 5 8 8 8 9 9 8 8 8 5 5
      5 2 2 2 2
3  pour 6 : 13 13 13 10 10 7 7 7 4 4 4 5 5 5 5 6 6 6 6 9 9 12 12 12
      11 11 11 11 10 10
```

$$\begin{pmatrix} 13 & 14 & 15 \\ 10 & 11 & 12 \\ 7 & 8 & 9 \\ 4 & 5 & 6 \\ 1 & 2 & 3 \end{pmatrix}$$

## Analyse de la fonction `compute_symbol_dir`

Voici les resultats donnés par cette fonction pour les trois tracés créés par la fonction `simu_symbol_dir` :

[illegible]

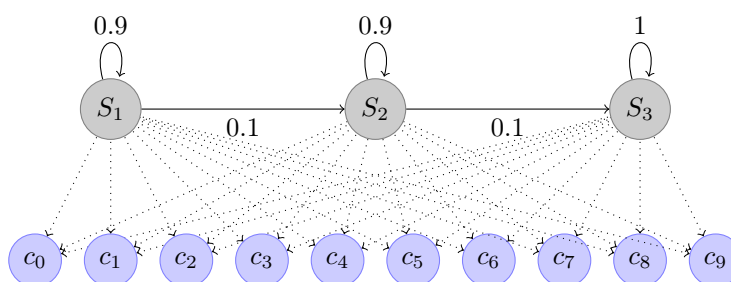
A vertical line with a downward-pointing arrow on the right side labeled with the number 7, and an upward-pointing arrow on the left side labeled with the number 2.

4

Cette fonction arrive à déterminer précisément l'angle de la direction du tracé. Cependant, si l'image n'est pas dans le bon sens ou si l'écriture est inclinée ou en italique, la reconnaissance de l'angle sera faussée. De plus, le chiffre "1" et le chiffre "7" peuvent être facilement confondu à cause de certaines polices d'écriture. L'ordre de chaque points est également essentiel pour la construction de la séquence.

## 1.2 Création et initialisation d'un HMM

Pour cette nouvelle partie, nous considérerons un HMM à 3 états ( $S_x$ ), observant 10 chiffres ( $c_x$ ) et une topologie gauche-droite :



Pour ce HMM observant une topologie gauche-droite, il est nécessaire que l'état  $S_1$  soit l'état initial. Pour cela, les probabilités initiales attribuées aux états sont de 1 pour  $S_1$ , et 0 pour  $S_2$  et  $S_3$ .

En ce qui concerne la matrice de transition, nous savons que la séquence  $T=30$ , et qu'il y a 3 états. Cela signifie qu'il n'y a qu'une chance sur dix de passer à l'état suivant et neuf chances sur dix de rester dans cet état. La matrice de transition est donc :

$$\begin{pmatrix} 0.9 & 0.1 & 0 \\ 0 & 0.9 & 0.1 \\ 0 & 0 & 1 \end{pmatrix}$$

Sans plus d'informations sur les chiffres, la matrice d'observation sera initialisée avec une distribution uniforme.

## 1.3 Entraînement des modèles

Pour cette troisième partie, seul quelques extraits de codes seront présentés. L'intégralité du code source se trouve dans le fichier `src/hmmDigitR.r` du projet.

### Création de la fonction `initHMMDigit`

La fonction `initHMMDigit` prend en entrée le nombre d'états et le nombre d'observations pour le HMM. Elle initialise ensuite les probabilités initiales (1, 0, 0, ...), ainsi que les matrices de transitions et d'émissions.

```

1  initHMMDigit <- function(nbrEtats , nbrObsv){
2    (...)
3
4    # Mise en place des proba de transition pour le HMM optimal
5    (...)
6    matriceInit <- matrix(transProbs , nrow = nbrEtats , ncol =
      nbrEtats , byrow = T)
7
8    # Mise en place des proba d'émission
9    for (i in 1:nbrEtats*nbrObsv){
10     emissionProbs <- c(emissionProbs , 1/nbrObsv)
11    }
12    matriceEmission <- matrix(emissionProbs , nrow = nbrEtats , ncol =
      nbrObsv , byrow = T)
13
14    # Appel de initHMM
15    resultat <- initHMM(etats , obsv , probInit , matriceInit ,
      matriceEmission)
16    return (resultat)
17  }

```

Une fois initialisés, les HMMs sont ensuite entraînés. Pour cela nous avons utilisé différents paramètres :

- Train\_compute\_symbol (5x3, 5x4, dir\_8 et dir\_16)
- Le nombre d'états (3, 4 et 7)

Pour plus de simplicité, les models entraînés ont été sauvegardés dans le dossier **Results**. Ainsi, il est plus facile de les recharger. Les Hmms pour 7 états n'ont pas pu être entraînés pour tous les Train\_compute\_symbol, car cela était trop coûteux pour la machine.

## 1.4 Performances de la reconnaissance

Après la phase d'entraînement des HMMs, et avant de tester le système, il faut au préalable définir des models en réunissant les HMMs entraînés de même catégories. Chaque model est composé de 10 HMMs, un pour chaque chiffre ayant été entraîné par le même type de données (5x3, 5x4, ...).

Pour plus de facilité, et pour améliorer le temps de chargement, l'intégralité de ces modèles a été au préalable sauvegardé dans le dossier **Results**. Ainsi il est plus facile de les charger et de lancer les tests.

La phase de test permet de mesurer, grace aux fonctions **classify** et **recognition**, le taux de reconnaissance de chaque HMM. De plus, toujours dans l'optique d'améliorer la lecture des résultats

### Résultats

Voici les résultats obtenus pour 3 états, avec les quatre types de données suivants :

*Pour 5x3 : 81.5%*

|    |   |     |     |     |     |     |    |     |     |     |     |
|----|---|-----|-----|-----|-----|-----|----|-----|-----|-----|-----|
| 1  |   | 0   | 1   | 2   | 3   | 4   | 5  | 6   | 7   | 8   | 9   |
| 2  | 0 | 121 | 1   | 0   | 0   | 0   | 2  | 1   | 0   | 4   | 0   |
| 3  | 1 | 1   | 104 | 23  | 0   | 1   | 0  | 0   | 3   | 1   | 3   |
| 4  | 2 | 4   | 1   | 102 | 1   | 8   | 0  | 4   | 2   | 1   | 15  |
| 5  | 3 | 0   | 0   | 2   | 128 | 1   | 0  | 0   | 3   | 0   | 10  |
| 6  | 4 | 1   | 3   | 13  | 11  | 104 | 1  | 9   | 4   | 0   | 5   |
| 7  | 5 | 1   | 3   | 1   | 33  | 0   | 72 | 8   | 1   | 14  | 4   |
| 8  | 6 | 2   | 0   | 0   | 1   | 3   | 0  | 135 | 1   | 1   | 0   |
| 9  | 7 | 0   | 0   | 0   | 0   | 11  | 0  | 2   | 111 | 3   | 0   |
| 10 | 8 | 2   | 0   | 0   | 0   | 1   | 0  | 0   | 2   | 123 | 0   |
| 11 | 9 | 0   | 0   | 8   | 8   | 0   | 0  | 0   | 1   | 2   | 110 |

*Pour 5x4 : 82.6%*

|    |   |     |     |     |     |     |    |     |     |     |     |
|----|---|-----|-----|-----|-----|-----|----|-----|-----|-----|-----|
| 1  |   | 0   | 1   | 2   | 3   | 4   | 5  | 6   | 7   | 8   | 9   |
| 2  | 0 | 117 | 3   | 0   | 0   | 0   | 2  | 0   | 0   | 7   | 0   |
| 3  | 1 | 3   | 105 | 9   | 3   | 7   | 0  | 1   | 0   | 0   | 8   |
| 4  | 2 | 3   | 1   | 109 | 3   | 4   | 0  | 1   | 4   | 1   | 12  |
| 5  | 3 | 0   | 0   | 1   | 130 | 1   | 0  | 0   | 2   | 0   | 10  |
| 6  | 4 | 0   | 2   | 2   | 9   | 122 | 0  | 7   | 3   | 0   | 6   |
| 7  | 5 | 3   | 5   | 0   | 35  | 0   | 59 | 9   | 1   | 23  | 2   |
| 8  | 6 | 5   | 0   | 0   | 0   | 1   | 0  | 135 | 0   | 2   | 0   |
| 9  | 7 | 0   | 0   | 1   | 0   | 14  | 0  | 0   | 110 | 1   | 1   |
| 10 | 8 | 2   | 1   | 0   | 0   | 1   | 1  | 0   | 2   | 121 | 0   |
| 11 | 9 | 0   | 0   | 2   | 8   | 0   | 1  | 0   | 1   | 0   | 117 |

*Pour dir\_8 : 81.4%*

|    |   |     |     |     |     |     |    |     |     |    |     |
|----|---|-----|-----|-----|-----|-----|----|-----|-----|----|-----|
| 1  |   | 0   | 1   | 2   | 3   | 4   | 5  | 6   | 7   | 8  | 9   |
| 2  | 0 | 116 | 0   | 1   | 0   | 1   | 2  | 6   | 0   | 1  | 2   |
| 3  | 1 | 0   | 100 | 10  | 0   | 7   | 0  | 4   | 15  | 0  | 0   |
| 4  | 2 | 3   | 0   | 123 | 1   | 1   | 0  | 0   | 3   | 5  | 2   |
| 5  | 3 | 0   | 0   | 0   | 119 | 0   | 17 | 0   | 0   | 7  | 1   |
| 6  | 4 | 2   | 0   | 0   | 0   | 142 | 0  | 2   | 0   | 0  | 5   |
| 7  | 5 | 1   | 0   | 1   | 26  | 0   | 79 | 0   | 0   | 12 | 18  |
| 8  | 6 | 24  | 0   | 0   | 0   | 0   | 1  | 115 | 0   | 0  | 3   |
| 9  | 7 | 0   | 0   | 12  | 1   | 0   | 0  | 0   | 111 | 2  | 1   |
| 10 | 8 | 13  | 0   | 3   | 2   | 0   | 2  | 5   | 13  | 87 | 3   |
| 11 | 9 | 0   | 2   | 2   | 1   | 2   | 3  | 0   | 0   | 3  | 116 |

*Pour dir\_16 : 82.8%*

|    |   |     |     |     |     |     |    |     |     |    |     |
|----|---|-----|-----|-----|-----|-----|----|-----|-----|----|-----|
| 1  |   | 0   | 1   | 2   | 3   | 4   | 5  | 6   | 7   | 8  | 9   |
| 2  | 0 | 115 | 1   | 1   | 1   | 1   | 1  | 6   | 0   | 1  | 2   |
| 3  | 1 | 0   | 107 | 13  | 0   | 5   | 0  | 1   | 7   | 2  | 1   |
| 4  | 2 | 1   | 1   | 123 | 0   | 0   | 0  | 0   | 3   | 4  | 6   |
| 5  | 3 | 0   | 0   | 0   | 125 | 0   | 9  | 0   | 0   | 8  | 2   |
| 6  | 4 | 0   | 0   | 0   | 0   | 144 | 1  | 1   | 0   | 1  | 4   |
| 7  | 5 | 1   | 0   | 0   | 30  | 0   | 89 | 4   | 4   | 6  | 3   |
| 8  | 6 | 15  | 0   | 0   | 0   | 0   | 1  | 122 | 0   | 1  | 4   |
| 9  | 7 | 0   | 0   | 5   | 1   | 0   | 1  | 1   | 116 | 3  | 0   |
| 10 | 8 | 6   | 1   | 4   | 1   | 0   | 2  | 7   | 16  | 82 | 9   |
| 11 | 9 | 2   | 1   | 3   | 2   | 6   | 4  | 0   | 0   | 6  | 105 |



## 2 Réseaux de neurones

## Conclusion