KAI YANG

DUT INFORMATIQUE

Rapport du stage

Laboratoire des Technologies Innovantes (LTI)

Développement d'algorithmiques d'apprentissage

29 MARS 2021 ~ 4 JUIN 2021

TUTEUR ENTREPRISE : JEAN-BAPTISTE MASSON

TUTEUR IUT : BRUNO MARHIC

Sommaire

Avant-propos

Je tiens à remercier ici toutes les personnes qui m'ont soutenu personnellement ainsi que professionnellement avant et au cours de ce stage.

Je remercie tout d'abord Mademoiselle Shiqing HUANG pour sa proposition de ce stage. Elle m'a donnée l'information du laboratoire LTI pour que je puisse présenter une demande à Monsieur Masson.

Je veux remercier en suite M. Jean-Baptiste Masson et M. Laurent Delahoche, ainsi que M. Marhic, pour m'avoir accueilli au sien de leur laboratoire. Je remercie M. Delahoche pour s'occuper tous mes documents administratifs, et Je remercie M. Masson pour tous les aides et les conseils avisés qui m'ont permis de travailler au mieux à la mission.

J'adresse également mon remerciement à mon collègues stagiaire M. Avicenne LO CASCIO, qui m'accompagne dans toute la période de mon stage durant ces 10 semaines.

Je voudrais également remercier le personnel de l'IUT d'AMIENS, en particulier les nettoyeurs qui m'ont accueilli devant TP 10 chaque matin, et remercier aussi leur travail pour que l'IUT soit toujours propre.

En fin, je voudrais remercier le personnel dans restaurant universitaire, pour leur faveurs et sollicitude, ainsi leur sourire tous les midis.

Introduction

Étant actuellement un étudiant en 2ème année du DUT informatique à l'IUT d'Amiens, il me fallait effectuer un stage de 10 semaines au sein d'une entreprise. Celui-ci m'a permis de mettre en application mes connaissances théoriques apprises durant ces deux années, ainsi que d'avoir une première expérience professionnelle dans le monde du travail.

J'ai toujours passion à la recherche. Heureusement, Mademoiselle Shiqing HUANG, qui était une étudiante à l'IUT d'Amiens en 2018 ~ 2020, m'a proposé le laboratoire des technologies innovantes (LTI). J'ai pu donc demander à M. Masson, et ai eu l'opportunité de réaliser mon stage dans ce laboratoire où j'ai été accueilli par MM. Masson et Delahoche.

Avec le guide de M. Masson, je continue le travail des anciens stagiaires, pour avancer un projet s'appelé « Intuitiv », qui est la suite du projet « AliceTher ».

Contexte du stage

Le Laboratoire des Technologies Innovantes a été créé en 2004 par le regroupement de deux équipes de recherche de l'Université Picardie Jules Verne spécialisées dans l'ingénierie des matériaux et la modélisation des systèmes complexes.

Il s'est réorganisé en 2012 sous quatre thématiques organisées en deux axes principaux

Axe I : Mécanique et Couplage

- **Thème 1** : Matériaux et Efficacité Énergétique (**MEE**)
- **Thème 2** : Modélisation Mécanique et Phénomènes de Transferts (**MMPT**)

Axe II : Énergies et Systèmes

- **Thème 3** : Systèmes Intelligents (**SI**)
- **Thème 4** : Énergie Électrique et Systèmes Associés (**EESA**)

J'ai intégré l'équipe d'Amiens spécialisé dans les Systèmes Intelligents (SI) pour les aider à travailler sur le projet « AliceTHER ». Ce projet a comme but de réaliser un système intelligent permettant de réduire les consommations en énergie en prenant en compte son environnement, comme par exemple gérer l'allumage du chauffage en fonction de la température de la maison ainsi que de la présence ou non d'habitants.

M. Masson chercheur sur ce projet, travaille sur un modèle statistique qui interviendrait dans la prise de décision du système intelligent et qui serait capable de prédire les présences et les absences des habitants à l'aide de données prélevées qui peuvent être indirectement en lien avec l'habitant, tel que le bruit ou le taux de CO_2.

Ma mission est de réaliser un environnement de modélisation et calcul. L'objectif est d'avoir une interaction facile et visuelle avec le logiciel, pour montrer les méthodes théoriques aux partenaires industriels. Au cours du stage, j'ai aussi programmé plusieurs méthodes de calcul.

Cahier des charges

Analyse de l'existant

Dans le projet « Intuitiv », on mesure des grandeurs physiques (CO_2, bruit...) dans un logement et on cherche à reconstituer les présences/absences des occupants.

Les stagiaires de 2016 et de 2017 se sont engagés à utiliser le package « HMM » du langage **R** pour effectuer des analyses de données sur les données générées par le capteur, et ont effectué beaucoup de recherches sur « HSMM ».

Le stagiaire de 2018 a fait un projet WPF en utilisant C# et XAML, pour démontrer les états d'observation en graphique. Néanmoins, c'est seulement un simulateur. Ce logiciel affiche les observations en utilisant les probabilités d'entrée, et simuler les résultats. Il n'y a pas d'analyse ou traitement des données.

Le stagiaire de 2019 a adapté le code **R** dans **Matlab**. Et puis, il combine le serveur pour gérer les données dans **DB** (Database), pour lire et écrire les données en utilisant les requêtes **SQL**.

Problématique

Ma mission est tout d'abord d'apprendre et étudier HMM, et de comprendre les algorithmes principaux concernés à HMM. Et puis, compléter le travail du stagiaire de 2018 (GUI pour HMM) pour que l'interface graphique soit utilisable en démonstration. Après complété, l'interface graphique devrait être améliorée infiniment.

Ensuite, j'ai étudié le HMM avec les observations continues et réalisé les algorithmes principaux.

Solution retenue

Le stagiaire de 2018 a choisi **C#**, mais pour ce qui est concerné les analyses ou les traitements des données, son idée est encore d'utiliser la librairie de R et appeler les fonctions de **R** (bien qu'il ne soit pas terminé). Les algorithmes de HMM ont des boucles longues, en particulier « Baum-Welch ». Le langage R est d'un langage interprété, ça prend donc longtemps pour effectuer l'analyse. En plus, Le langage R est d'un langage orienté processus, donc le nombre de paramètres des fonctions est très nombreux. Mon idée est de faire les analyses en utilisant un langage compilé et orienté objet, **C#** par exemple, pour que le HMM soit encapsulé. Dans le même temps, les méthodes correspondantes peuvent être appelées directement dans le projet **WPF**, et les données d'entrée seront plus concis.

Sur la partie de HMM avec les observations continues, grâce à la modularisation du code, seule de la petite partie du code doit être modifiée ou ajoutée.

Explication des outils théoriques

J'ai dû d'abord étudier et comprendre le modèle de Markov caché. Pendant le cours de « M3202 - Modélisations mathématiques », nous avons vu la chaîne de Markov simple. À l'aide d'exercices que M. Masson m'avez donnés, j'ai pu étudier la chaîne de Markov caché, et ses applications. Dans la suite, je vais expliquer petit-à-petit ce modèle, et présenter ses applications.

Une Chaîne de Markov

Commençons par le processus de Markov simple, comme nous voulons prédire la présence ou non d'habitants dans notre maison, si nous prenons une unité de temps, une heure par exemple, nous dessinons un processus ci-dessous.

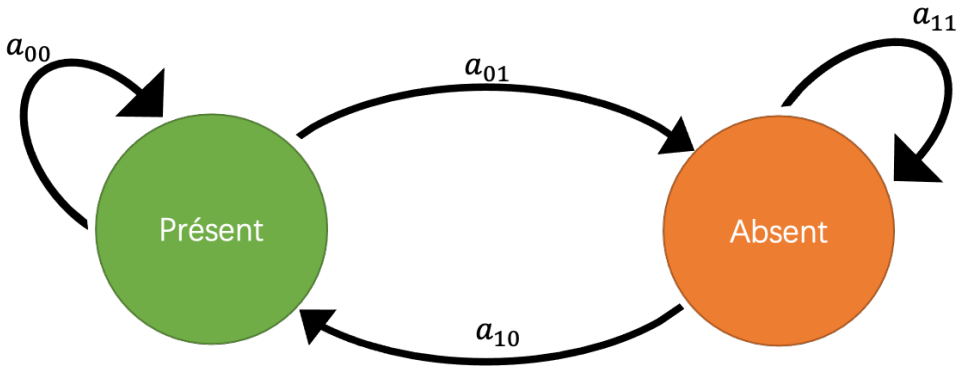


fig.1 un processus de Markov simple pour le problème Présent/Absent

Chaque état de notre habitant, présence et absence, est représenté par un état dans notre modèle. Et chaque arc représente une probabilité d'aller d'un sommet à un autre, comme indiqué, j'ai une probabilité de de rester dans l'état présent quand je le suis déjà et d'aller dans l'état absent. Donc ça correspond une matrice de transition (« Stochastic matrix » en l'anglais).

$$A = \begin{bmatrix} a_{00} & a_{01} \\ a_{10} & a_{11} \end{bmatrix} \text{ où } a_{ij} = P(X_{t+1} = q_j \,|\, X_t = q_i)$$

On définit les états par $\{q_0, q_1, \cdots, q_{N-1}\}$ où le N est le nombre d'état. Dans le cas de ce problème, N est 2, les états sont $\{q_0 = \text{"présent"}, q_1 = \text{"absent"}\}$.

On représente l'écoulement du temps de manière discrète.

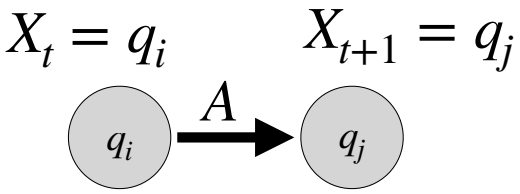


fig.2 Modélisation pour un processus de Markov

Il est évident que, l'état du $t+1$ ne dépend que l'état du t. C'est exactement la propriété de Markov. Un processus stochastique vérifie la propriété de Markov si et seulement si la distribution conditionnelle de probabilité des états futurs, étant donnés les états passés et l'état présent, ne dépend en fait que de l'état présent et non pas des états passés (limité de « mémoire »).

Et en gros, un processus de Markov est un processus stochastique possédant la propriété de Markov.

Les chaînes de Markov

Avec les unités de temps, on lie plusieurs processus de Markov. Les états sont transmis selon la matrice (la matrice de transition). On a donc obtenu une chaîne de Markov.

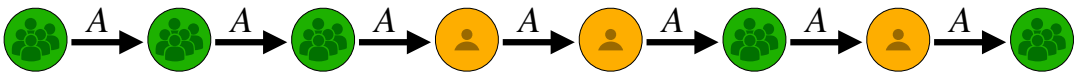


fig.3 Modélisation pour la chaîne de Markov

En gros, les chaînes de Markov est les processus de Markov à temps discret.

Une chaîne de Markov caché

Admettons à présent que pour notre modèle ne soit plus capable de connaitre directement les états de notre habitant, mais qu'à la place nous avons des données qui peuvent être indirectement liées à notre habitant comme le bruit ou le CO2. Nous allons rajouter des nouveaux sommets dans le graphe, qui représenterons les données générées par les différents états de notre habitant. Reprenons notre exemple et disons que notre utilisateur selon s'il est absent ou présent, aura des chances différentes de générer du bruit. Nous aurons donc un modèle qui ressemblera à ceci :

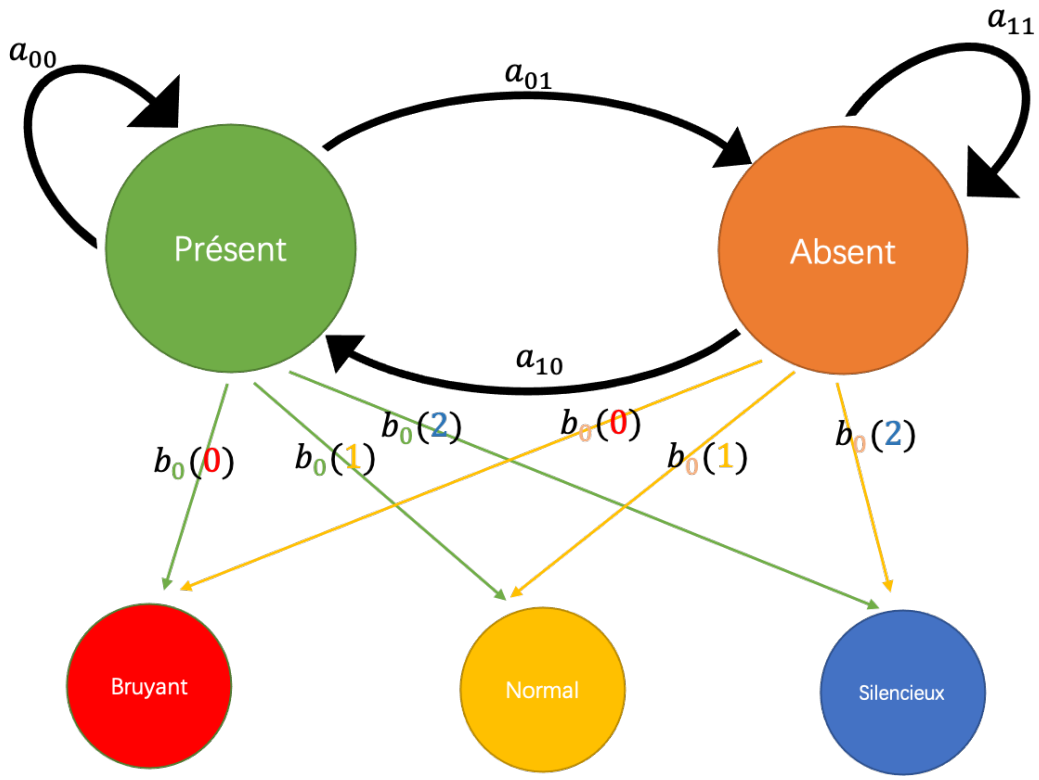


fig.4 un processus de Markov caché pour le problème Présent/Absent

L'état de présent ou absent s'appelle donc l'état caché, et l'état pour décrire le bruit s'appelle l'état d'observation. Il y a donc deux processus stochastiques. Un est le même d'avant avec la même matrice stochastique, un autre est entre l'état caché et l'état d'observation, dont la matrice stochastique s'appelle matrice d'émission.

$$B = \begin{bmatrix} b_0(0) & b_0(1) & b_0(2) \\ b_1(0) & b_1(1) & b_1(2) \end{bmatrix} \text{ où } b_i(k) = P(\mathcal{O}_t = k \,|\, X_t = q_i)$$

On définit les états d'observation par $\{v_0, v_1, \ldots, v_{M-1}\}$ où M est le nombre d'état observé. Dans le cas de ce problème, M est 3, les états d'observation sont $\{v_0 = \text{"Bruyant"}, v_1 = \text{"Normal"}, v_2 = \text{"Silencieux"}\}$.

Pour obtenir un modèle plus abstrait, on dessine le processus comme ci-dessous

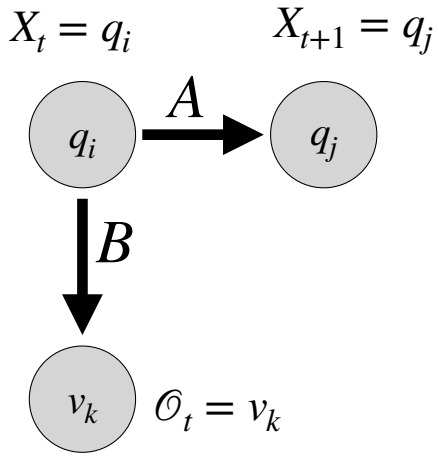


fig.5 Modélisation pour un processus de Markov caché

Les chaînes de Markov caché (Hidden Markov Model)

Item comme la chaîne de Markov simple, on lie plusieurs chaînes à temps discrète. Les états cachés sont transmis selon la matrice A (la matrice de transition), et entre chaque l'état caché et son état d'observation, ils sont transmis selon la matrice B (la matrice d'émission).

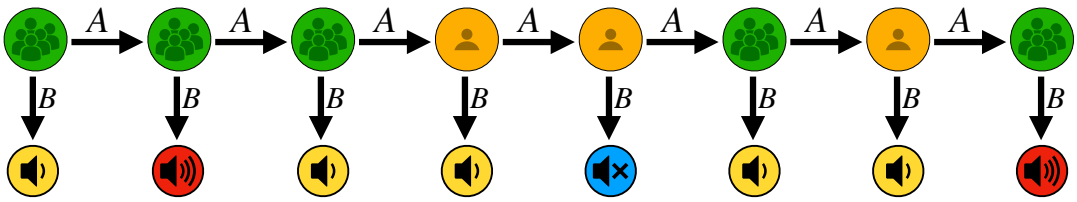


fig. 6 Modélisation pour la chaîne de Markov caché

Pour plus générale, un modèle de Markov caché est décrit par trois paramètre : l'état initial, la matrice de transition et la matrice d'émission. ($\lambda = \{\pi, A, B\}$).

Les trois problèmes

Pour un modèle de Markov caché, on a essentiellement trois problèmes à résoudre. Ceux correspondent à trois algorithmes à réaliser.

Évaluation

Si on a déjà connu tous les paramètres d'un HMM, on peut évaluer, pour une série d'observation, la probabilité. Néanmoins, si on calcule comme la chaîne de Markov simple, ça sera trop long, car il demande de faire $2TN^T$ fois de multiplications. Mais avec l'algorithme de forward-backward, on ne fait que N^2T fois de multiplications. En plus, le résultat de cet algorithme (tableau $\alpha_t(i)$ et $\beta_t(i)$) aide aussi pour les deux problèmes dans la suite.

Apprentissage

Algorithme Baum-Welch est pour trouver les paramètres d'un HMM en utilisant les observations. L'idée est d'essayer et modifier les paramètres pour que la probabilité finale soit la plus grande. Cet algorithme fait partie de l'algorithme EM, qui est plus général.

Décoder

Algorithme Viterbi est pour trouver les états cachés en utilisant les observations et les vrais paramètres. En fait, le développement d'un HMM est un automate, ça vaut dire un graphe orienté et fondé. Les poids sont les probabilités. L'idée de Algorithme Viterbi est donc de trouver le chemin de poids le plus grand.

HMM avec les observations continues

Le HMM classique traite les données avec les observations discrètes. Mais dans notre vie réelle, il existe de l'observation continue, par exemple, la concentration de CO_2. Il faut également que nous pouvons chercher la relation d'état caché avec l'observation continue.

À l'aide du document que M. Masson a résumer pour l'algorithme EM, j'ai pu étudier et réaliser notre version de HMM adapté.

Développement de la mission

Partie théorie

Une part significative de la mission a été tout d'abord de maîtriser l'utilisation des outils mathématiques présentés plus haut et de comprendre le travail effectué par mon prédécesseur afin d'être opérationnel en ce qui concerne l'utilisation des HSMM.

J'ai donc passé deux semaines dans le langage R pour découvrir HMM car le langage R est un langage mathématique, et aussi un langage script. Il est plus facile et plus rapide à faire démarrer un problème mathématique. En plus, il y a déjà un package « HMM » dans R. Le code source de ce package m'a aidé aussi beaucoup pour comprendre ce modèle. Lire et faire de la connaissance du code source me permet aussi de faire les petites modifications pour adapter notre problème dans la future.

J'ai pu ensuite étudier les formules de HMM. Cela me permet de savoir « pourquoi ». Dans les documents, il n'y a que les résultats de formule et les conseils d'usage. C'est vrai qu'il est rapide de commencer et s'intégrer, mais il n'y a peu de possibilité de progresser. J'ai donc étudié les démonstrations des formules.

Par la suite, j'ai étudié sur le HMM avec les observations continues, pour mieux adapter notre sujet.

Partie code

J'ai réalisé les trois algorithmes dans un langage compilé. Pour la raison que je dois penser à compléter le travail de GUI du stagiaire de 2018, qui est fait par C# et XAML, j'ai donc choisi C# et dotNet comme la plate-forme.

Pour les données d'entrée, j'ai choisi de passer en utilisant les fichiers *.csv, vu que c'est le fichier classique pour enregistrer les séries de donnée. Pour cette raison, j'ai dû d'abord créer une classe d'outil « CsvHelper » pour lire et écrire les fichier *.csv.

Ensuite, pour enregistrer le résultat après avoir lu, j'ai donc créé une classe qui ressemble à « DataFrame » dans R. Je l'appelle « RDataFrame ».

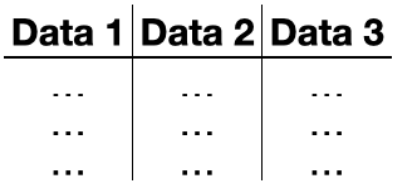


fig.7 l'illustration de la classe "RDataFrame"

Et puis, j'ai réalisé une classe pour enregistrer un tableau mais avec la possibilité d'avoir les index de type "string". Je l'appelle « RArray » car il ressemble le « Array » dans **R**.

En fin, et le plus important, c'est la classe qui encapsule toutes les données et les méthodes pour un HMM.

Ci-dessous est les quatre classes pour la partie « Model ».

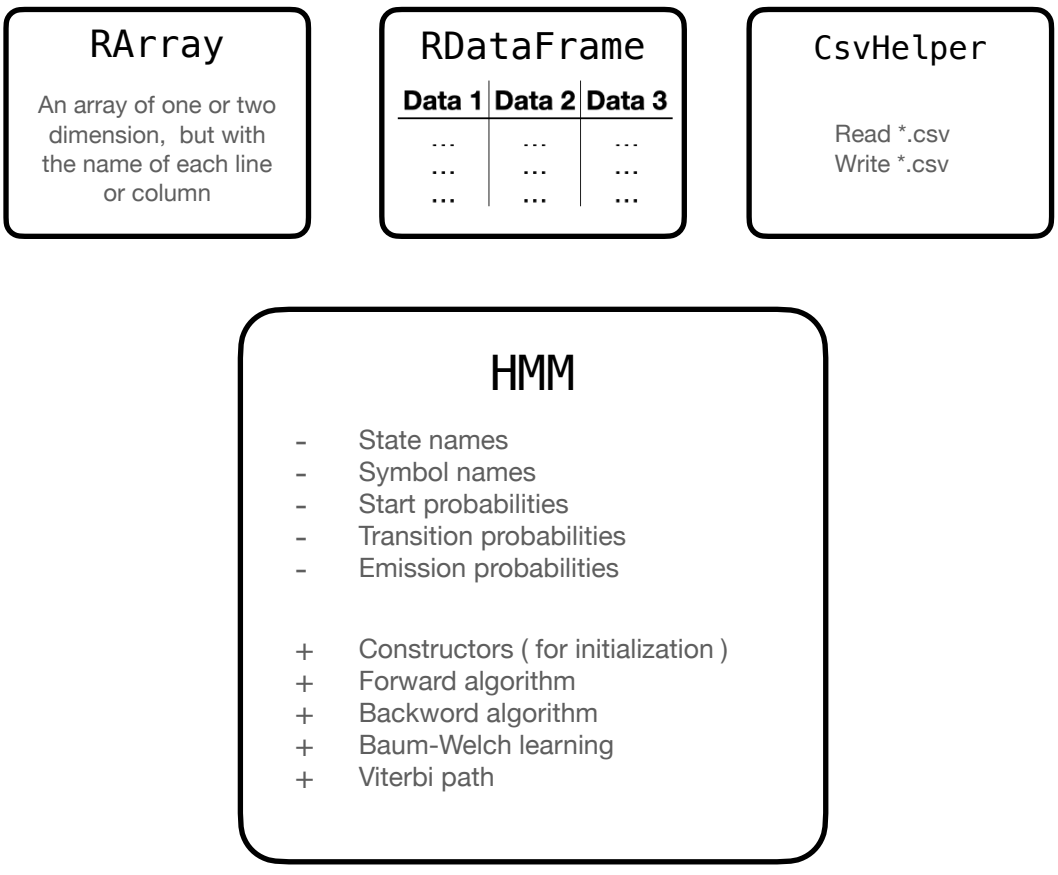


fig.8 Le projet "Model"

Après avoir fini cette partie, j'ai premièrement fait un projet Console, pour tester tous les fonctionnements du projet « Model ». Ça me permet aussi d'éviter de "debug" dans un autre projet complexe (un projet WPF par exemple) dans le future.

Une fois cela fait, j'ai donc commencé à construire l'interface graphique pour qu'il soit plus facile d'entrer les données et que l'affichage soit plus lisible et joli.

Résultats obtenus

Pour la théorie, il y a donc plusieurs documents pour démonter les formules.

Pour le code **R**, il y a code source du package « HMM », et le code pour présenter l'utilisation des fonctions. Ainsi que deux fichiers pour lancer le simulateur pour créer les données de tester pour « HMM » des observations discrètes et continues.

Pour le code **C#**, il y a trois projets. Un projet « Model » concerne toutes les réalisations de l'algorithme, les données et les outils pour HMM. Un projet « View » pour les interfaces graphiques, et un projet « MyConsole » pour tester ou debug dans la ligne de commande.

Voici est la fenêtre principale de cette interface graphique:

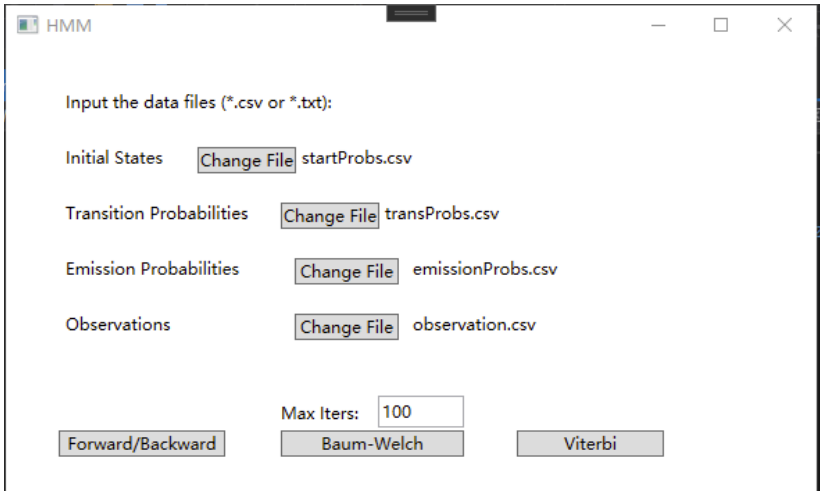
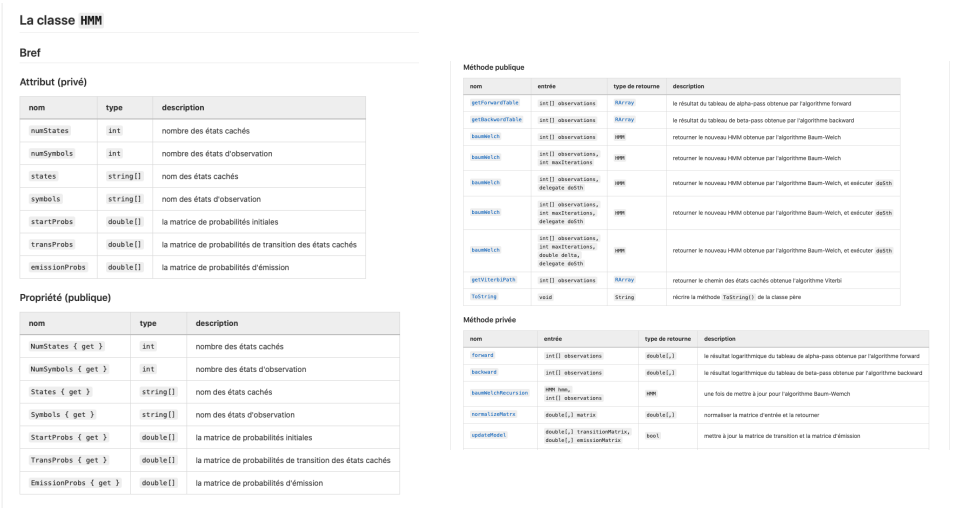


fig.9 la fenêtre principale

Dans cette fenêtre, nous pouvons charger les données en fournissant les fichiers "*.csv". Nous pouvons double-clic le nom du fichier pour la Prévisualisation, ou appuyer un des trois boutons en bas, pour lancer le calcule de l'analyse.

En fin, pour expliquer toutes les méthodes et son utilisation, pour que les personnes à la suite peuvent continuer à faire, j'ai fait les documentations pour toutes les classes en utilisant « Markdown ». Ça permet d'afficher dans le site « GitHub » ou « GitLab » avec le mis en page (https://gitlab.com/ltiut/2021/documentation/-/tree/master/fr_FR/Model).

Voici est un exemple pour la classe « HMM ». Tous d'abord, j'ai un résumé pour tous les attributs, propriété, surcharges de l'opérateur et méthodes.



La classe `HMM`

Bref

Attribut (privé)

nom	type	description
`numStates`	`int`	nombre des états cachés
`numSymbols`	`int`	nombre des états d'observation
`states`	`string[]`	nom des états cachés
`symbols`	`string[]`	nom des états d'observation
`startProbs`	`double[]`	la matrice de probabilités initiales
`transProbs`	`double[]`	la matrice de probabilités de transition des états cachés
`emissionProbs`	`double[]`	la matrice de probabilités d'émission

Propriété (publique)

nom	type	description
`NumStates { get }`	`int`	nombre des états cachés
`NumSymbols { get }`	`int`	nombre des états d'observation
`States { get }`	`string[]`	nom des états cachés
`Symbols { get }`	`string[]`	nom des états d'observation
`StartProbs { get }`	`double[]`	la matrice de probabilités initiales
`TransProbs { get }`	`double[]`	la matrice de probabilités de transition des états cachés
`EmissionProbs { get }`	`double[]`	la matrice de probabilités d'émission

Méthode publique

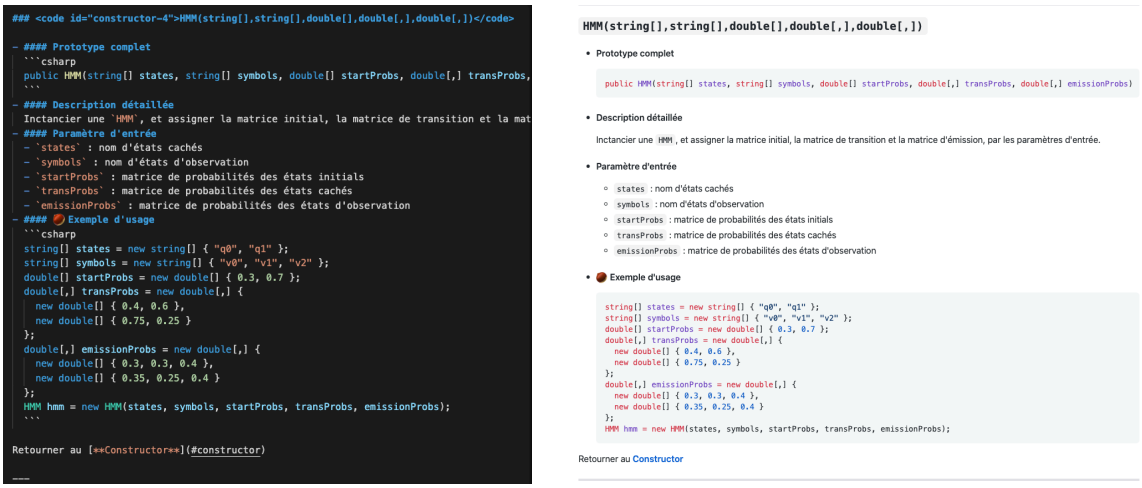
nom	entrée	type de retourne	description
`getForwardTable`	`int[] observations`	`RArray`	le résultat du tableau de alpha-pass obtenue par l'algorithme forward
`getBackwardTable`	`int[] observations`	`RArray`	le résultat du tableau de beta-pass obtenue par l'algorithme backward
`baumWelch`	`int[] observations`	`HMM`	retourner le nouveau HMM obtenue par l'algorithme Baum-Welch
`baumWelch`	`int[] observations, int maxIterations`	`HMM`	retourner le nouveau HMM obtenue par l'algorithme Baum-Welch
`baumWelch`	`int[] observations, delegate doSth`	`HMM`	retourner le nouveau HMM obtenue par l'algorithme Baum-Welch, et exécuter `doSth`
`baumWelch`	`int[] observations, int maxIterations, delegate doSth`	`HMM`	retourner le nouveau HMM obtenue par l'algorithme Baum-Welch, et exécuter `doSth`
`baumWelch`	`int[] observations, int maxIterations, double delta, delegate doSth`	`HMM`	retourner le nouveau HMM obtenue par l'algorithme Baum-Welch, et exécuter `doSth`
`getViterbiPath`	`int[] observations`	`RArray`	retourner le chemin des états cachés obtenue l'algorithme Viterbi
`ToString`	`void`	`String`	récrire la méthode `ToString()` de la classe père

Méthode privée

nom	entrée	type de retourne	description
`forward`	`int[] observations`	`double[,]`	le résultat logarithmique du tableau de alpha-pass obtenue par l'algorithme forward
`backward`	`int[] observations`	`double[,]`	le résultat logarithmique du tableau de beta-pass obtenue par l'algorithme backward
`baumWelchRecursion`	`HMM hmm, int[] observations`	`HMM`	une fois de mettre à jour pour l'algorithme Baum-Wemch
`normalizeMatrx`	`double[,] matrix`	`double[,]`	normaliser la matrice d'entrée et la retourner
`updateModel`	`double[,] transitionMatrix, double[,] emissionMatrix`	`bool`	mettre à jour la matrice de transition et la matrice d'émission

fig.10 Le résumé de la classe "HMM"

Par la suite, ou en cliquant le terme (les méthodes ou les surcharges de l'opérateur), nous pouvons voir tous les détails. Il y a aussi un exemple pour présenter comment utiliser pour les constructeurs ou les méthodes publiques.



```
### <code id="constructor-4">HMM(string[],string[],double[],double[,],double[,])</code>

- #### Prototype complet
  ```csharp
  public HMM(string[] states, string[] symbols, double[] startProbs, double[,] transProbs,
  ```
- #### Description détaillée
  Inctancier une `HMM`, et assigner la matrice initial, la matrice de transition et la mat
- #### Paramètre d'entrée
  - `states` : nom d'états cachés
  - `symbols` : nom d'états d'observation
  - `startProbs` : matrice de probabilités des états initials
  - `transProbs` : matrice de probabilités des états cachés
  - `emissionProbs` : matrice de probabilités des états d'observation
- #### 🍎 Exemple d'usage
  ```csharp
  string[] states = new string[] { "q0", "q1" };
  string[] symbols = new string[] { "v0", "v1", "v2" };
  double[] startProbs = new double[] { 0.3, 0.7 };
  double[,] transProbs = new double[,] {
    new double[] { 0.4, 0.6 },
    new double[] { 0.75, 0.25 }
  };
  double[,] emissionProbs = new double[,] {
    new double[] { 0.3, 0.3, 0.4 },
    new double[] { 0.35, 0.25, 0.4 }
  };
  HMM hmm = new HMM(states, symbols, startProbs, transProbs, emissionProbs);
  ```

Retourner au [**Constructor**](#constructor)

___
```

`HMM(string[],string[],double[],double[,],double[,])`

- Prototype complet

```
public HMM(string[] states, string[] symbols, double[] startProbs, double[,] transProbs, double[,] emissionProbs)
```

- Description détaillée

  Inctancier une `HMM`, et assigner la matrice initial, la matrice de transition et la matrice d'émission, par les paramètres d'entrée.

- Paramètre d'entrée
  - `states` : nom d'états cachés
  - `symbols` : nom d'états d'observation
  - `startProbs` : matrice de probabilités des états initials
  - `transProbs` : matrice de probabilités des états cachés
  - `emissionProbs` : matrice de probabilités des états d'observation
- 🍎 Exemple d'usage

```
string[] states = new string[] { "q0", "q1" };
string[] symbols = new string[] { "v0", "v1", "v2" };
double[] startProbs = new double[] { 0.3, 0.7 };
double[,] transProbs = new double[,] {
  new double[] { 0.4, 0.6 },
  new double[] { 0.75, 0.25 }
};
double[,] emissionProbs = new double[,] {
  new double[] { 0.3, 0.3, 0.4 },
  new double[] { 0.35, 0.25, 0.4 }
};
HMM hmm = new HMM(states, symbols, startProbs, transProbs, emissionProbs);
```

Retourner au Constructor

fig.11 Le code de documentation pour un constructeur et son affichage dans GitHub

Un exemple de difficulté que j'ai rencontré

En fait, pendant la durée de travail, j'ai rencontré beaucoup de difficulté. Elles concernent à la théorie, à l'utilisation de "API", et aussi à l'environnement de système. Je vais ci-dessous en présenter un, qui m'a pris beaucoup de temps à résoudre.

Dans la partie de l'algorithme **Baum-Welch**, je voulais afficher la progression du calcul car les itérations prennent beaucoup de temps. Pendant le calcul, il n'est pas interactif s'il n'y a rien sur l'écran. Mon idée est d'ajouter un « progress bar » et afficher quelque chose après chaque fin d'itération (comme afficher ci-dessous).

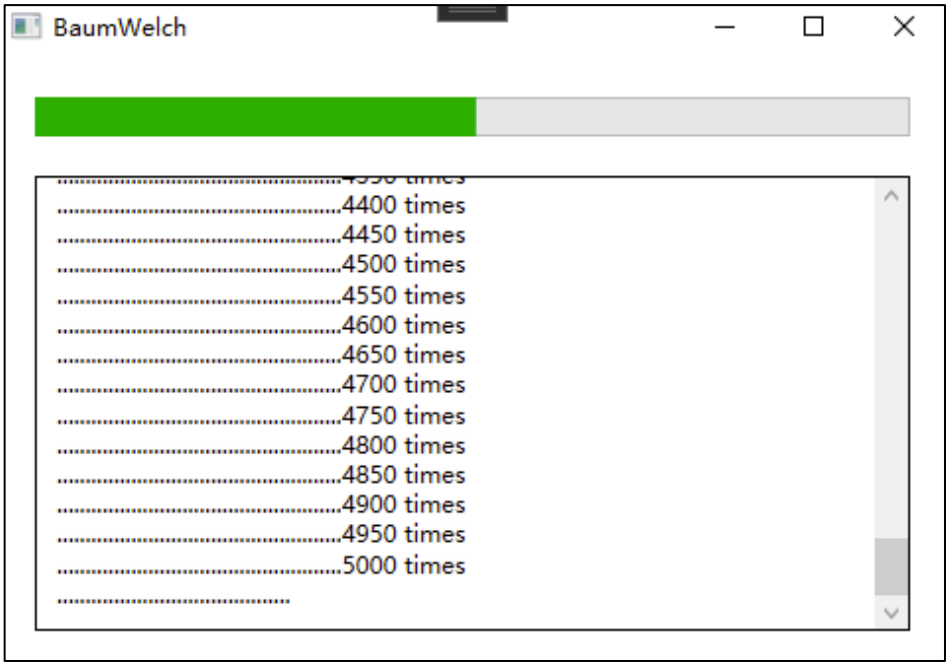


fig.12 Le résultat d'attente

Néanmoins, le « progress bar » ne bouge pas du tout pendant le calcule, et il bouge à 100% directement à la fin de calcule. L'affichage de nombre de fois s'affiche normalement dans ligne de commande, mais dans « TextBloc » de GUI, ça ne s'affiche pas.

J'ai passé beaucoup de temps pour vérifier les grammaires du code et la logique, j'ai également passé beaucoup de temps pour faire les test codes pour afficher les valeur intermédiaire dans la ligne de commande. Mais tout passe bien.

C’était un jour, je regardais un projet de Android avec Java, il y a une notion pour la classe « Handler » pour gérer les threads. Si suite un mis à jour d’interface est une opération longue, l’interface va geler ou échoué.

J’étais éclairé, j’ai donc cherché comment gérer le thread de l’interface sous C#. J’ai pu donc trouver une solution avec la méthode « BeginInvoke » dans la classe « Dispatcher ». Après la petite modification de priorité de l’opération appelée, et avec l’aide de documentation de C# dans le site de Microsoft, j’ai pu obtenir ce que je voulais.

Conclusion

Grâce à ce stage, j'ai pu découvrir le monde de la recherche dans le milieu de l'informatique. Celui-ci m'a permis de découvrir les projets que l'on pouvait traiter en l'analyse de données, les méthodes et le rythme de travail au sein d'un laboratoire. C'est une expérience spéciale pour moi en France. Et j'ai également expérimenté ce que c'est que de travailler en France et en quoi c'est différent d'étudier à l'école.

Au cours de ce stage, j'ai une compréhension très approfondie de HMM, et j'ai également une certaine compréhension des connaissances correspondantes, telles que l'algorithme EM, Multiplicateur de Lagrange et le modèle de mélange Gaussien. En outre, J'ai obtenu de la connaissance de l'apprentissage automatique et le traitement des données.

De plus, j'ai fait des progrès dans la gestion de la structure du code et projet.

Si je devais recommencer mon stage, je commencerais à écrire de la documentation dans le même temps que j'écris du code. Parce que l'écriture de la documentation aidera également la construction de la structure du code. Pendant la rédaction de la documentation, je peux facilement trouver les structures déraisonnables. Par exemple, quelles méthodes sont inutile, quels paramètres d'entrée sont déraisonnables. Mais si le code est terminé, il y aura trop de changement pour une petite modification.

Malgré tout, ce stage a vraiment été positif pour moi. Ça va être une expérience majeure sur mon CV (Curriculum Vitae).