Documentation du code Matlab sur l'algorithme ER pour l'apprentissage de réseau bayésien dynamique étiqueté

Ce document présente le code Matlab pour l'apprentissage de réseaux bayésiens dynamique étiqueté. Voici, étape par étape, la description des différents fichiers. Chaque fichier contient soit une fonction (Un moreau de programme retournant un ou plusieurs éléments), soit un script (un programme "finalisé", sans retour permettant d'enregistrer ou d'afficher des éléments de l'algorithme).

1 Éléments permettant l'apprentissage

Cette section liste l'ensemble des variables et fonctions permettant l'apprentissage d'un réseau bayésien étiqueté. Pour effectuer l'apprentissage d'un réseau bayésien dynamique étiqueté, il faut commencer par donner un certain nombre d'informations, présentés dans la permière partie de cette section. La seconde partie liste les fonctions utilisant ces variables et décrit les variables en retour de ces fonctions.

1.1 Variables initiales à définir

- n : Nombre de nœuds - entier
- T : Nombre de pas de temps - entier
- Q : Nombre de jeux de données - entier
- nIni : Nombre d'initialisation différentes (l'initialisation permet de déterminer la valeur ε d'un nœud pour le calcul de la vraisemblance) - entier
- nImp : Nombre d'impulseurs (liens positifs) de forces différentes - entier
- nInh : Nombre d'inhibiteurs (liens négatifs) de forces différentes - entier
- nAct : Nombre d'actions différentes (l'action permet de déterminer la valeur μ d'un nœud pour le calcul de la vraisemblance) - entier
- I : Vecteur de taille n décrivant le type d'initialisation de chaque nœuds (valeurs possibles : 1,...,nIni) - Vecteur d'entiers
- A : Matrice de taille $n * T * Q$ décrivant le type d'action (de 1 à nAct) de chaque nœud à chaque pas de temps pour chaque jeu de données - Matrice d'entiers
- ini : Matrice de taille 2*nIni décrivant les paramètres ε_{app} (première ligne) et ε_{sur} (seconde ligne) associés à chaque initialisation - Matrice de réels
- imp : Matrice de taille 2*nImp décrivant les paramètres ρ_{app} (première ligne) et ρ_{sur} (seconde ligne) associés à chaque impulseurs - Matrice de réels
- inh : Matrice de taille 2*nInh décrivant les paramètres τ_{app} (première ligne) et τ_{sur} (seconde ligne) associés à chaque inhibiteurs - Matrice de réels
- imp : Matrice de taille 2*nAct décrivant les paramètres μ_{app} (première ligne) et μ_{sur} (seconde ligne) associés à chaque action de pénalisation - Matrice de réels
- G : Matrice décrivant les arcs du graphe. Une ligne par arc, et 4 colonnes : [i, j, type, force] avec
  - i et j : deux nœuds du graphe pour lesquels il existe un arc de i vers j - entiers
  - type : type d'étiquettes. L'arc décrit-il un lien d'impulsion (type=1) ou d'inhibition (type=2) - entiers
  - force : force de l'étiquette, entre 1 et $nImp$ si l'arc décrit un lien d'impulsion et entre 1 et $nInh$ si l'arc décrit un lien d'initialisation - entiers
- k0 : nombre de parents max - entier
- par : nombre de processeurs en parallèle - entier
- t0 : Matrice de taille n*Q décrivant l'état de chaque nœud dans chaque jeu de données au temps $t = 0$ - Matrice d'entiers $0/1$
- D : Matrice de taille $n * T * Q$ décrivant la présence ou l'absence de chaque nœud pour chaque pas de temps et chaque jeu de données. - Matrice d'entiers $0/1$

Fonctions :

- dSimul(G,A,t0,I,ini,imp,inh,act). Retour : D (Matrice $n * T * Q$) - Simule des données D à l'aide d'un réseau bayésien dynamique étiqueté.
- lvTot(D,G,A,I,ini,imp,inh,act). Retour : lv (réel) - Calcul de vraisemblance de données D par rapport à un modèle de réseau bayésien dynamique étiqueté
- parEstKn(D,G,A,I,par0,parKn,parEq). Retour : [iniE (matrice 2*nIni),impE (matrice 2*nImp),inhE (matrice 2*nInh),actE (matrice 2*nAct)] - Estime les paramètres d'un réseau bayésien dynamique étiqueté par maximum de vraisemblance. Demande de définir 3 tableaux : par0, parKn et parEq. Chacun de ces tableaux contiennent 4 éléments, chacun étant des tableaux, respectivement de taille 2*nIni, 2*nInh, 2*nImp et 2*nAct permettant de définir pour tous les types d'initialisations, d'impulsion, d'inhibitions et d'actions diverses informations (un exemple est montré dans le tableau 1) :
  - Les éléments des tableau de par0 contiennent des réels correspondant à une valeur de départ pour les paramètres (par défaut : 0.1)
  - Les éléments de parKn sont binaires (entiers 0/1). Si un élément vaut 0, alors la valeur du paramètre correspondant est considéré comme inconnue, et donc à estimer. Si un élément vaut 1, alors sa valeur est considérée comme connue et égale à sa valeur de départ décrite dans par0.
  - Les éléments de parEq contiennent des entiers. Les éléments de chaque tableau de parEq contenant les mêmes entiers seront définis comme égaux. Ces entiers peuvent donc prendre des valeurs entre 1 et $nIni + nImp + nInh + nAct$. Si tous sont différents, alors aucun paramètre n'est structurellement égal à un autre.
- varILP_core(D, k, nIni, nImp, nInh). Retour : [Ale (Matrice sparse de réels), Ble (Vecteur sparse de réels), indVar (Tableau de taille 3*3 contenant des matrices d'entiers), Ri (Matrice d'entiers)] - Construit les matrices A et B du problème de programmation linéaire en nombres entiers sous contrainte que $Ax \leq B$. Elle retourne les variables suivantes :
  - Ale : Matrice sparse de grande taille dans laquelle chaque colonne désigne une variable du programme linéaire, et chaque ligne une contrainte de ce problème. Un élément de cette matrice représente le coefficient associé à une variable (en colonne) pour une contrainte (en ligne). -Matrice de réels
  - Ble : Vecteur sparse de même taille que le nombre de lignes de Ale dans lequel un élément correspond à la partie indépendante des variables des contraintes du problème de programmation linéaire. -Vecteur de réels
  - indVar est un tableau de taille 3*3. Chaque élément de ce tableau contient des matrices d'entiers correspondant à des indices de variables de la matrice Ale. Ce tableau permet de retrouver les indices de chaque variable du problème de programmation linéaire. Les éléments de ce tableau sont tous des matrices de nombres entiers :
    - indVar(1, 1) est une matrice de taille n*nImp qui contient les indices des variables $G^q_{i,j}$ (cette variable vaut 1 si il existe un lien d'impulsion de i vers j de force q, 0 sinon). Chaque ligne correspond à un nœud du graphe, chaque colonne à une étiquette d'impulsion. Un élément de cette matrice indique l'indice de cette variable dans la matrice Ale
    - indVar(1, 2) est une matrice de taille n*nInh qui contient les indices des variables $G^r_{i,j}$ (cette variable vaut 1 si il existe un lien d'inhibition de i vers j de force r, 0 sinon). Chaque ligne correspond à un nœud du graphe, chaque colonne à une étiquette d'inhibition. Un élément de cette matrice indique l'indice de cette variable dans la matrice Ale

* indVar(1, 3) est un vecteur de taille nIni qui contient les indices des variables I_i^e (cette variable vaut 1 si i a pour valeur d'initialisation e). Un élément de ce vecteur indique l'indice de cette variable dans la matrice Ale.
* Les autres éléments du tableau recensent les indices des variables intermédiaires ν, M et L, respectivement bornes inférieures, supérieures et exacte du nombre de parents de chaque étiquette pour chaque pas de temps et chaque jeu de données. Ils contiennent des matrices à 4 dimensions de taille T*Q*k+1*nImp ou T*Q*k+1*nInh contenant les indices des variables. Pour un nœud i lors d'un pas de temps t dans un jeu de données s, pour un nombre d et une étiquette l, $\underset{d,l}{\nu_i^{s,t}}$, $\underset{d,l}{M_i^{s,t}}$ et $\underset{d,l}{\Lambda_i^{s,t}}$ valent 1 si i a, au pas de temps t dans le jeu de données s, respectivement au moins, au plus et exactement d parents d'étiquette l présents, 0 sinon. indVar(2, 1), indVar(2, 2) et indVar(2, 3) indiquent respectivement les indices des variables ν, M et Λ pour les impulseurs et indVar(3, 1), indVar(3, 2) et indVar(3, 3) les indices de ces variables pour les impulseurs.

– Ri est une matrice d'entiers contenant un grand nombre de lignes et 4+nImp+nInh colonnes. Cette matrice recense les indices des variables $\underset{D_{imp},D_{inh}}{R_i^{u,s,t}}$ non indiquées dans indVar. $\underset{D_{imp},D_{inh}}{R_i^{u,s,t}}$ vaut 1 si i a exactement $D_{imp} = \{d_{imp_1}, ..., d_{imp_{nImp}}\}$ parents de chaque force d'impulsion, $D_{inh} = \{d_{inh_1}, ..., d_{inh_{nInh}}\}$ parents de chaque force d'inhibition et a pour valeur d'initialisation u (0 sinon). Chaque ligne de la variable Ri contient une variable binaire de ce type parmi toutes celles possibles (pour tous t, s, u, D) et les colonnes de cette matrice représentent, dans l'ordre :
  * L'indice de la variable R en question
  * Le pas de temps t
  * Le jeu de données s
  * Le type d'initialisation u
  * Le nombre d'impulseurs de chaque force $d_{imp_1}, ..., d_{imp_{nImp}}$
  * Le nombre d'inhibiteurs de chaque force $d_{inh_1}, ..., d_{inh_{nInh}}$

Il est à noter que ces matrices ne se construisent que pour un seul nœud i. La construction de ces matrices étant similaire pour n'importe quel nœud, i n'est pas en indice, cependant, les variables construites ne permettent de résoudre qu'un seul problème d'ILP associé à un nœud spécifique. L'apprentissage complet de structure de graphe devra se faire pour chacun des nœuds.

- La fonction varILP_core construit uniquement les matrices des contraintes de l'ILP permettant de construire les variables intermédiaires. L'objectif est d'avoir une fonction peu spécifique. Ajouter des contraintes dans l'ILP peut se faire en définissant d'autres fonctions prenant en entrée les informations. Par exemple varILP_structC(Ale,indVar, i, nImp, nInh) renvoie les matrices [A1, B1], en y ajoutant les contraintes suivantes :
  – Il n'existe aucun lien récursif (d'un nœud vers lui-même
  – Il ne peut y avoir qu'un seul lien au maximum entre deux nœuds

  De même, la fonction varILP_foodWeb permet d'ajouter des contraintes permettant de définir chaque initialisation u_i en fonction des arcs du graphe, afin de définir la règle suivante :
  – $u_i = 1$ si i est une

- lvILP(D, A,i,ini, imp, inh, act, nbV, Ri). Retour : vecteur C de même taille que le nombre de colonnes de Ale. Cette fonction permet de construire le vecteur C du programme linéaire en nombres

entiers correspondant au nœud i, c'est à dire la contribution de chaque variable du problème à la fonction objectif. Il est à noter que pour l'apprentissage de structure de réseau bayésien dynamique étiqueté, la fonction linéaire est la vraisemblance, que l'on cherche à maximiser. Or, la plupart des fonctions d'optimisation par programmation linéaire minimisent ma fonction objectif. Il vaudra alors mieux utiliser le vecteur -C pour l'optimisation.

- restoration_step_ILP(Ao, Bo, Co, indVar, i). Retour : Gi (Matrice d'entiers contenant 4 colonnes), Ini (entier). Cette fonction permet d'apprendre les parents d'un nœud i par ILP à partir des matrices et des vecteurs Ao, Bo et Co (A, B et C de l'ILP) généré par les fonctions précédentes. Gi est un tableau contenant une ligne par arc, c'est à dire par lien VERS i, et contenant les colonnes suivantes :
  - j et i : deux nœuds du graphe pour lesquels il existe un arc de j vers i - entiers
  - type : type d'étiquettes. L'arc décrit-il un lien d'impulsion (type=1) ou d'inhibition (type=2) - entiers
  - force : force de l'étiquette, entre 1 et $nImp$ si l'arc décrit un lien d'impulsion et entre 1 et $nInh$ si l'arc décrit un lien d'initialisation - entiers

  Ini est un entier correspondant au type d'initialisation de i. Le problème d'apprentissage de structure complet se fait en apprenant les parents de tous les nœuds du graphe.

Matrice	Initialisations	Impulsions	Inhibitions	Actions
par0	0.01 0.01	0.01 0.01 0.01 0.01	0.01 0.01 0.01 0.01	0.8 1 0.8 1
parKn	0 0	0 0 0 0	0 0 0 0	1 1 1 1
parEq	1 1	3 4 5 6	7 8 9 10	11 11 13 13

Table 1: Exemple d'un contenu des tableaux par0, parKn et parEq pour l'estimation de paramètres. Cet exemple est pour un réseau bayésien dynamique étiqueté avec 1 initialisation, 2 impulseurs, 2 inhibiteurs et 2 actions différentes. La première ligne de chaque matrice concerne la valeur des paramètres pour l'apparition, la seconde ligne pour la survie. Dans cet exemple, les paramètres μ_1^{app} et μ_1^{sur} valent 0.8 et les paramètres μ_2^{app} et μ_2^{sur} valent 1. Les autres paramètres sont inconnus. On sait également que les paramètres ε^{sur} et ε^{app} sont égaux. L'estimation des paramètres se fera en respectant ces contraintes.

2 Données "Arthropodes"

Cette partie décrit la manière dont sont intégrées et utilisés les données issue de l'étude d'arthropodes piégés dans des champs expérimentaux. Les éléments permettant de les utiliser se trouvent dans le dossier

suivant :

`/home/eauclair/MIAT/Public/code_RBDE/BD_Arthropodes/`

2.1 Données brutes

Les données de Dave se présentent sous la forme de fichiers excels. Il y en a un par culture et par pas de temps. Il y a 4 cultures différentes : B, M, SR et WR ; et 2 pas de temps : *early* (avant récolte) et *late* (après récolte), plus *all*, qui combine ces pas de temps. Il y a donc 12 fichiers excel. Une ligne de l'un de ces fichiers renseigne de l'abondance d'une espèce dans une parcelle pour un piège. Ils contiennent les informations suivantes :

ARTHROPOD_BRC : Code identifiant l'espèce d'arthropode

SITE_ID : Code chiffré identifiant une parcelle (Toutes les parcelle ont un numéro différent, même si ils sont dans un fichier différent). Ce numéro va de 1 à 303

SITE_CODE : Code identifiant une parcelle et sa culture (M38 est la parcelle de maïs numéro 30. Il peut exister un SR38 par exemple. Dans mon code, j'utilise plutôt SITE_ID.)

DISTANCE : Distance du piège : 0, 2 ou 32 (0 est changé en -1 dans les fichiers "ALL")

TOTAL_T1 : Abondance de l'espèce dans la parcelle "conventionnelle" (chaque parcelle est doublée, et contient une partie conventionnelle et une partie OGM)

TOTAL_T2 : Abondance de l'espèce dans la parcelle "OGM" (chaque parcelle est doublée, et contient une partie conventionnelle et une partie OGM)

Le fichier ARTHROPOD_BRC_CODES contient les noms latins et communs de chaque espèce (désigné par l'identifiant ARTHROPOD_BRC).

2.2 Nettoyage des données

Le dossier csvData contient ces fichiers convertis au format csv. Les fichiers ALL n'y figurent pas car ils ne contiennent pas d'informations nouvelles : il s'agit simplement de la somme des abondances early et late pour chaque ligne de chaque fichier. Les scripts extraction_donnees.R et extraction_donnees_nobird.R (exclusion des espèces d'oiseaux) permettent de réorganiser ces données en créant un dossier contenant plusieurs fichiers au format csv. Il y a un fichier par parcelle, le nom du fichier indiquant la culture (B, M, SR ou WR) et l'identifiant SITE_ID de la parcelle. Chacun de ces fichiers compte une ligne par espèce, et pour chaque espèce est indiquée :

Un numéro d'espèce unique, plus simple à manipuler que le code ARTHROPOD_BRC utilisé dans la suite du code. ce numéro va de 1 à 5095.

Esp : Le code ARTHROPOD_BRC identifiant l'espèce

E0 : L'abondance de l'espèce en *early* sur le piège 0

E2 : L'abondance de l'espèce en *early* sur le piège 2

E32 : L'abondance de l'espèce en *early* sur le piège 32

L0 : L'abondance de l'espèce en *late* sur le piège 0

L2 : L'abondance de l'espèce en *late* sur le piège 2

L32 : L'abondance de l'espèce en *late* sur le piège 32

Pour n espèces d'arthropodes au total, il y a n ligne pour chacun de ces fichiers. Les abondances de toutes les espèces de la base de données sont présentes dans ce jeu de données, qu'elles aient été observées dans la parcelle ou non. Il y a donc beaucoup de données manquantes dans ces fichier indiquées par *NA*. Les scripts matlab utilisent ces données. Les dossiers cleanData, cleanData_OGM, nobird et nobird_OGM contiennent ces fichiers pour les différents types de cultures (conventionnelle ou OGM) selon si les espèces d'oiseaux sont intégrés ou pas.

2.3 Ajout de connaissances

Le fichier learning_data.xlsx indique, pour quelques espèces, des informations relatives à sa taille, sa masse et son groupe fonctionnel. À partir de ce fichier, j'ai généré le fichier size.csv, qui contient, pour chaque espèce présente dans le fichier learning_data.xlsx :

id : Le code ARTHROPOD_BRC identifiant l'espèce

nb : Le numéro d'espèce correspondant à celui généré lors de l'étape précédente

size : La classe de taille de l'espèce

2.4 Intégration sur matlab

Le script matlab script_data_arthro.m utilise les fichiers csv générés précédemment et les convertis en données de présence absence par culture, étape par étapes. À chaque espèce correspond un numéro (de 1 à 5095) et à chaque parcelle un autre numéro (de 1 à 303). Ces numéros vont constituer l'indice de ces espèces et de ces parcelles dans les matrices de données. Il en est de même pour les pas de temps : 1 correspond à *early*, 2 à *late*. Certains numéros ne correspondent à aucune parcelle. Le numéro de parcelle va donc de 1 à 303, mais il n'y a que 257 parcelles différentes.

1. Création d'une matrice 3D (dim1 : espèce ; dim2 : temps/piège ; dim3 : parcelle) la dimension 2 donne à la fois le pas de temps et le piège. Il y a donc 6 éléments dans cette dimension.

2. Réduction de la matrice en ne prenant en compte que les espèces observées (non manquante) au moins une fois dans l'une des parcelles. Il y a 251 espèces observées au moins une fois. Une espèce "observée" est une espèce qui a été renseignée dans le jeu de données initial. Une espèce peut n'avoir que des valeurs observées à 0 et être inclue dans la matrice réduite. Le vecteur *nz* généré lors de cette étape permet de renseigner du numéro des espèce retenues afin de ne pas perdre leur identification.

3. Transformation en présence/absence : une espèce est présente dans une parcelle à un pas de temps dans un piège si elle son abondance est supérieure à 0

4. Séparation par pièges et par culture. On distinguera les données "intérieures" constitués des pièges 2 et 32 et les données "extérieures", c'est à dire le piège 0. Après cette étape, on peut distinguer 8 matrices différentes : une par culture (B, M, SR, WR) et par type de piège (intérieur, extérieur) de données binaires à 3 dimensions : 251(espèces)*2(temps)*257(parcelles).

5. Sélection d'un jeu de données particulier (dans le script : piège intérieur, betterave) et suppression des espèces avec au moins une observation "présente" (1). Matrice de données binaires à 3 dimensions (comme précédemment) avec seulement n espèces (celles avec au moins une observation "présente"). Cette étape permet également d'obtenir un vecteur *block* de taille n contenant la taille de chaque espèce. Le vecteur *lNz* généré lors de cette étape permet de renseigner du numéro des espèce retenues afin de ne pas perdre leur identification.

6. Sélection d'un jeu de données pour un piège (intérieur ou extérieur) pour l'ensemble des parcelles, sans distinction de culture. Comme précédemment, on ne retient que les espèces avec au moins une observation "présente" dans l'une des parcelles et on génère le vecteur *block*.

3 Apprentissage complet

Cette section décrit les différents scripts et fonctions permettant de faire tourner un algorithme E-R complet à partir des fonctions décrites dans les sections précédentes.
Le script script_example.m montre, sur un exemple de modèle type "réseau écologique" comment, pas à pas :

- Donner les valeurs des variables d'entrée
- Donner les valeurs des matrices de valeurs initiale, valeurs connues d'égalité des paramètres
- Simuler un graphe (par SBM) et des données (par RBDE)
- Effectuer l'étape d'estimation des paramètres
- Effectuer l'étape de restauration, ou d'apprentissage de structure

La plupart de ces étapes peuvent également être appellées par des fonctions :

- dataGen_param(nIni, nImp, nInh, nAct, parExist). Retour : [ini, imp, inh, act, par0, parKn, parEq] - Construit les matrices des valeurs de départ, valeurs connues et valeurs égales des différents paramètres, et donne des valeurs aléatoires pour chaque paramètre inconnu (ces valeurs sont intégrées dans les matrices ini, imp, inh et act). La variable parExist est une matrice de taille 4*2. Chaque colonne représente un type de paramètres (respectivement ε, ρ, τ et μ), la première ligne représente l'apparition, la seconde ligne la survie. La valeur d'un élément de cette matrice est 0 ou 1. Cela permet de dire à la fonction les paramètres qui ont un effet sur l'apparition ou la survie (les valeurs 1 dans la matrice) et ceux qui n'en ont pas (les valeurs de 0 dans la matrice).
- ERlearning(D,A,I,k0,par0,parKn,parEq,Gsel). Retour : [graphFinTot (Matrice d'entiers), iniFinTot (Vecteur d'entiers), timeERFinTot (réel), lvFinTot (réel), paramFinTot (matrice de réels), paramSBMFinTot (vecteur de réels)] - Effectue l'apprentissage complet de structure d'un graphe par algorithme E-R sans prior SBM ni structure particulière. Elle retourne les variables suivantes :
  - graphFinTot : Matrice décrivant les arcs du graphe appris. Une ligne par arc, et 4 colonnes : [i, j, type, force] avec
    - i et j : deux nœuds du graphe pour lesquels il existe un arc de i vers j - entiers
    - type : type d'étiquettes. L'arc décrit-il un lien d'impulsion (type=1) ou d'inhibition (type=2) - entiers
    - force : force de l'étiquette, entre 1 et $nImp$ si l'arc décrit un lien d'impulsion et entre 1 et $nInh$ si l'arc décrit un lien d'initialisation - entiers
  - iniFinTot : Vecteur de taille n donnant pour chaque nœud sa valeur d'initialisation (ou de comportement, permettant de choisir une valeur de ε)
  - timeERFinTot : Le temps, en secondes, d'exécution de l'algorithme E-R complet
  - lvFinTot : La log-vraisemblance finale du graphe appris par rapport aux données et aux paramètres appris

  - paramFinTot : Matrice de taille *Nombre d'étiquettes* * 2, décrivant, pour chaque paramètre (dans l'ordre : ε, ρ, τ, μ), sa valeur apprise lors de la dernière étape E de l'algorithme pour l'apparition (première ligne) et la survie (seconde ligne)

- ERlearningSBM(D,A,I,k0,par0,parKn,parEq,Gsel,SBMblock). Retour : [graphFinTot (Matrice d'entiers), iniFinTot (Vecteur d'entiers), timeERFinTot (réel), lvFinTot (réel), paramFinTot (vecteur de réels), paramSBMFinTot (vecteur de réels)] - Effectue l'apprentissage complet de structure d'un graphe par algorithme E-R avec prior SBM. Le retour est le même que pour la fonction précédente, avec en plus le vecteur de réels paramFinTot de taille 5, donnant les valeurs apprises lors de la dernière étape E des paramètres relatifs au SBM : [α, β_1, β_2, α_{FT}, β_{FT}]

- ERlearningSBMFoodWeb(D,A,I,k0,par0,parKn,parEq,Gsel,SBMblock). Retour : [graphFinTot (Matrice d'entiers), iniFinTot (Vecteur d'entiers), timeERFinTot (réel), lvFinTot (réel), paramFinTot (vecteur de réels), paramSBMFinTot (vecteur de réels)] - Effectue l'apprentissage complet de structure d'un graphe par algorithme E-R avec prior SBM pour le cas particulier de l'aprpentissage de réseaux écologiques. Les retours sont les mêmes que pour la fonction précédente. La seule différence est que cette fonction attribue lors de l'algorithme les différents comportement I en fonction du graphe appris, tel que $I(espcebasale) = 1$ et $I(espcenonbasale) = 2$.

Plusieurs scripts utilisent ces fonctions pour apprendre des graphes par l'algorithme ER :

- Le script script_expe.m permet d'effectuer l'apprentissage des différentes expérimentations décrites dans le plan d'expérience, et d'en enregistrer les résultats comme données au format .mat

- Le script script_expe_arthro.m permet d'effectuer l'apprentissage de structure de réseau écologique sur les données arthropodes décrites plus haut.

- Le script script_precRec.m permet, à partir d'un graphe réel Gr et d'un graphe appris Gapp, de calculer les précisions et rappels de chaque étiquette.

- Le script script_result_expe.m permet, à partir des données résultats d'expérimentation, de générer des fichier .txt décrivant, pour chaque expérimentation :

  - Une vue générale (les fichier se terminant en OV.txt) donnant, pour chaque expérimentation :

    Exp : Son numéro - Identifiant entre 1 et 15

    app : Une valeur décrivant si, lors de cette expérimentation, les interactions avaient un impact sur l'apparition (1) ou non (0) - variable binaire

    sur : Une valeur décrivant si, lors de cette expérimentation, les interactions avaient un impact sur la survie (1) ou non (0) - variable binaire

    lv : La log-vraisemblance du graphe appris - variable continue

    time : Le temps d'apprentissage en seconde - variable continue

  - Les données relatives à l'estimation des paramètres (les fichier se terminant en Par.txt) donnant, pour chaque expérimentation :

    Exp : Son numéro - Identifiant entre 1 et 15

    RepsApp : La valeur du paramètre "réel" (utilisé pour la simulation) ε^{app} - variable continue (comprise entre 0 et 1)

    RrhoApp : La valeur du paramètre "réel" (utilisé pour la simulation) ρ^{app} - variable continue (comprise entre 0 et 1)

    RtauApp : La valeur du paramètre "réel" (utilisé pour la simulation) τ^{app} - variable continue (comprise entre 0 et 1)

EepsApp : La valeur du paramètre ε^{app} estimé lors de la dernière étape E - variable continue (comprise entre 0 et 1)

ErhoApp : La valeur du paramètre ρ^{app} estimé lors de la dernière étape E - variable continue (comprise entre 0 et 1)

EtauApp : La valeur du paramètre τ^{app} estimé lors de la dernière étape E - variable continue (comprise entre 0 et 1)

RepsSur : La valeur du paramètre "réel" (utilisé pour la simulation) ε^{sur} - variable continue (comprise entre 0 et 1)

RrhoSur : La valeur du paramètre "réel" (utilisé pour la simulation) ρ^{sur} - variable continue (comprise entre 0 et 1)

RtauSur : La valeur du paramètre "réel" (utilisé pour la simulation) τ^{sur} - variable continue (comprise entre 0 et 1)

EepsSur : La valeur du paramètre ε^{sur} estimé lors de la dernière étape E - variable continue (comprise entre 0 et 1)

ErhoSur : La valeur du paramètre ρ^{sur} estimé lors de la dernière étape E - variable continue (comprise entre 0 et 1)

EtauSur : La valeur du paramètre τ^{sur} estimé lors de la dernière étape E - variable continue (comprise entre 0 et 1)

– Les données relatives à l'apprentissage du graphe (les fichier se terminant en App.txt) donnant, pour chaque expérimentation :

Exp : Son numéro - Identifiant entre 1 et 15

preImp : La précision relative aux impulseurs - variable continue (pourcentage)

preInh : La précision relative aux inhibiteurs - variable continue (pourcentage)

recImp : Le rappel relatif aux impulseurs - variable continue (pourcentage)

recInh : Le rappel relatif aux inhibiteurs - variable continue (pourcentage)