
Apprentissage et sélection de réseaux bayésiens dynamiques pour les processus online non stationnaires

Matthieu Hourbracq^{1,2}, Pierre-Henri Wuillemin¹,
Christophe Gonzales¹, Philippe Baumard²

1. Sorbonne Université, CNRS, LIP6, UMR 7606, F-75005 Paris, France

firstName.lastName@lip6.fr

2. Akheros S.A.S., France

firstName.lastName@akheros.com

RÉSUMÉ. Les réseaux bayésiens dynamiques (DBN) fournissent un formalisme graphique probabiliste décrivant, à travers des dépendances conditionnelles, des systèmes dynamiques complexes sous incertitude. Dans la plupart des cas, le processus de Markov génératif sous-jacent est supposé homogène, ce qui signifie que ni sa topologie ni ses paramètres n'évoluent au cours du temps. Par conséquent, apprendre un DBN pour modéliser un processus non stationnaire sous cette hypothèse aboutira à de pauvres capacités de prédictions. Dans cet article, nous nous plaçons dans le cas où la non stationnarité du processus s'exprime par la présence de différents modes de comportement. Dans ce contexte, nous proposons un cadre pour l'apprentissage en temps réel des différents modèles probabilistes sous-jacents à chaque mode, sans hypothèses sur leur nombre et leur évolution. Nous proposons un modèle pour la dynamique des transitions entre modes inspiré des Hidden semi-Markov Models (HsMM) et Graphical Duration Models (GDM). Nous montrons la performance de la méthode sur des données simulées.

ABSTRACT. Dynamic Bayesian Networks (DBNs) provide a principled scheme for modeling and learning conditional dependencies from complex multivariate time-series data. However, in most cases, the underlying generative Markov model is assumed to be homogeneous, meaning that neither its topology nor its parameters evolve over time. Therefore, learning a DBN to model a non-stationary process under this assumption will amount to poor predictions capabilities. Thus we build a framework to identify, in a streamed manner, transition times between underlying models and a framework to learn them in real time, without assumptions about their evolution. We propose a model for the dynamic of the transitions between modes stemming from Hidden semi-Markov Models (HsMMs) and Graphical Duration Models (GDMs). We show the method performances on simulated datasets.

MOTS-CLÉS : DBN, ns-DBN, tv-DBN, non-stationnaire, apprentissage, temps réel

KEYWORDS: DBN, ns-DBN, tv-DBN, non-stationnary, learning, real time

DOI:10.3166/RIA.32.75-109 © 2018 Lavoisier

1. Introduction

Dans beaucoup de domaines, particulièrement ceux des systèmes d'information ou de la biologie, les processus observés évoluent avec le temps sur plusieurs échelles. Leur état interne change dynamiquement, décrivant des trajectoires complexes. Certains événements ou entités peuvent en influencer d'autres à un moment donné, mais ces corrélations ne tiennent pas nécessairement en toutes circonstances ou pour toujours. Les influences entre entité peuvent ainsi changer; et tout modèle souhaitant capturer un tel processus, sans pour autant pouvoir observer le mécanisme responsable de tels changements, ne peut être stationnaire : sa structure et/ou ses paramètres doivent évoluer avec le temps aussi. En effet, dans un cadre stationnaire, seul un comportement serait décrit, moyennant toutes les observations.

1.1. Motivation

Dans l'application visée par notre étude, nous souhaitons modéliser le comportement d'un système d'information au sein d'un réseau d'ordinateurs en temps réel afin d'identifier des comportements d'intrusion. Un IDS (*Intrusion Detection Systems*) a pour but de déterminer parmi les traces des différents trafics d'information (logs, appels systèmes, paquets réseaux, etc.) des classes de comportements anormaux qui peuvent donc présenter des risques de sécurité. On distingue deux approches principales pour les IDS : détection d'anomalie et détection d'utilisation incorrecte (Debar *et al.*, 1999).

La détection d'anomalie modélise les observations sur une période de temps sans intrusion afin de pouvoir observer des déviations par rapport à cette modélisation et de les caractériser comme anomalie. Ses principaux défauts sont la nécessité de la connaissance d'une période sans intrusion et l'absence de remise en cause de cette connaissance, d'adaptation (Debar *et al.*, 1999).

La détection d'utilisation incorrecte, quant à elle, s'appuie sur une base de connaissance de comportements d'intrusion (bases de *malware*, de virus, etc.) sous forme de signatures dans les observations. Les nouvelles observations sont alors comparées à ces signatures. Ces principaux défauts sont également le manque d'adaptabilité à de nouvelles formes d'intrusion (attaque polymorphique ou vulnérabilité *zero-days*, etc.).

En général, les deux approches sont considérées comme complémentaires. Avec l'accroissement exponentiel des réseaux d'entreprises, des *data-centers* qui produisent de très importantes quantités d'observations, ces méthodes requièrent trop d'interventions humaines pour pallier à leurs faiblesses. La constante augmentation en nombre et en complexité des attaques suggèrent le besoin de nouvelles méthodes d'investigation.

1.2. Cadre général

Nous nous plaçons dans un cadre d'apprentissage non supervisé pour les IDS, ce qui invalide les deux approches précédentes. Il semble raisonnable de supposer que

les processus qui peuvent être observés dans un tel système ont un caractère non stationnaire. En effet, n'importe quel programme ou application peut accepter un large éventail d'entrées, communiquer avec d'autres programmes, et les chemins d'exécution choisis (où le processus est localisé et ce qu'il y fait) sont souvent au moins dépendant des entrées.

Nous avons ainsi besoin d'un cadre général qui permette l'apprentissage de tels processus non stationnaires en temps réel. L'objectif est d'identifier depuis un flux d'observations les différents comportements que peut suivre le processus pour ensuite les caractériser. Il est important de pouvoir déterminer, par exemple, le caractère récurrent, exceptionnel ou anormal d'un comportement identifié. Nous appellerons modes les différents comportements que peut adopter le processus. Dans cet article, nous proposons de nous focaliser sur les réseaux bayésiens dynamiques non stationnaires. Ainsi, nous souhaitons produire une collection *minimale* de modèles probabiliste représentant chacun un mode et expliquant au mieux la dynamique des processus observés.

L'hypothèse de modes de comportements semble raisonnable (tout du moins plus vraisemblable que l'existence d'un mode unique). S'il n'existe pas de modes de comportements, le système change à tout instant de dynamique et il ne serait donc pas possible de le modéliser. Dans la littérature sur les séries temporelles, notamment en économétrie (Perron, 1989; Zivot, Andrews, 2012), cela peut être dû à la présence d'une tendance (déterministe ou stochastique) avec des incidents (chocs) dont les effets sur les paramètres sont temporaires; ou à l'existence d'une racine unitaire (*unit root*) avec des chocs aux effets permanents.

Les réseaux bayésiens dynamiques (Dean, Kanazawa, 1989; Murphy, 2002a) sont un formalisme graphique probabiliste décrivant, à travers des dépendances conditionnelles, des systèmes dynamiques complexes sous incertitude. Pourtant, le terme *dynamique* dans les DBN fait référence à l'évolution du système au cours du temps et non à la dynamique de la structure du réseau ou de ses paramètres. Une fois déterminées (par exemple par apprentissage sur un ensemble d'observations), les indépendances conditionnelles et les paramètres sont censés expliquer la dynamique du comportement du système à tout moment. Toutefois, dans beaucoup d'applications (davantage lorsque les données ne sont pas produites de manière contrôlée) supposer l'homogénéité du modèle sous-jacent décrivant l'état du système semble être une hypothèse trop forte. Ce problème a reçu une attention académique particulière ces dernières années, donnant naissance, par exemple, aux réseaux bayésiens dynamiques non stationnaires (ns-DBN) (Robinson, Hartemink, 2009; 2010; Grzegorzczak, Husmeier, 2009; 2011; Gonzales *et al.*, 2015) ou réseaux bayésiens dynamiques à temps variable (TV-DBN) (Song *et al.*, 2009), avec des applications pour les systèmes biologiques (Grzegorzczak *et al.*, 2008). Puisque les processus ont beaucoup de chemins d'exécutions et un espace d'entrée important, il semble malavisé de supposer qu'un modèle homogène puisse capturer avec précision leur évolution. Deux invocations différentes peuvent donner lieu à deux traces d'exécution totalement différentes. Nous construisons par conséquent notre *framework* sur les ns-DBN.

Dans cet article, nous proposons un nouvel algorithme pour modéliser les processus non stationnaires en utilisant les réseaux bayésiens dynamiques non stationnaires. Après une introduction succincte des (d)BN et ns-DBN, nous construisons un algorithme d'apprentissage pour ns-DBN et présentons un cadre complet d'identification des différents modes d'un comportement non stationnaire, avant d'évaluer ses performances sur un des cas simulés. Enfin, nous concluons et suggérons des travaux futurs.

2. Réseaux bayésiens dynamiques (non stationnaires)

2.1. Réseaux bayésiens dynamiques

Les DBN sont des réseaux bayésiens classiques (Pearl, 1988) dans lesquels les nœuds $\{X_i(t), i = 1 \dots n\}$, représentant des variables aléatoires (discrètes), sont indexés par le temps discret t . On note $\mathbf{X}(t)$ l'ensemble des variables aléatoires au pas de temps t (qu'on appelle par convention la *time slice* t). Les DBN fournissent une représentation factorisée de la loi jointe P sur un intervalle de temps fini $[1, \tau]$, définie comme suit :

$$P(\mathbf{X}(1) \dots, \mathbf{X}(\tau)) = \prod_{t=1}^{\tau} \prod_{i=1}^n P(X_i(t) | \mathbf{U}_i(t)) \quad (1)$$

où $\mathbf{U}_i(\cdot)$ dénote les parents de $X_i(\cdot)$ et $P(X_i(t) | \mathbf{U}_i(t))$ dénote la fonction de probabilité conditionnelle associée à la variable aléatoire $X_i(t)$ sachant $\mathbf{U}_i(t)$. $\mathbf{X}(t) = \{X_1(t), \dots, X_n(t)\}$, est appelé une "slice" et représente l'ensemble des variables indexées par le même temps t . Cette probabilité jointe $P(\mathbf{X}(1), \dots, \mathbf{X}(\tau))$ représente les croyances sur les trajectoires possibles du processus dynamique $\mathbf{X}(t)$.

Les DBN suivent la propriété de Markov d'ordre un ; les parents d'une variable à la *time slice* t doivent se trouver soit dans la *time slice* $t - 1$ soit t , autrement dit seul le dernier état passé est nécessaire pour prédire le présent :

$$\mathbf{U}_i(t) \subseteq \mathbf{X}(t-1) \cup \mathbf{X}(t) \setminus X_i(t) \quad (2)$$

De plus, les probabilités conditionnelles sont temporellement invariantes (*propriété d'homogénéité temporelle*) ; ces probabilités dépendent du temps passé et non absolu :

$$P(X_i(t) | \mathbf{U}_i(t)) = P(X_i(2) | \mathbf{U}_i(2)), \forall t \in [2, \tau] \quad (3)$$

Ainsi, pour spécifier un DBN, il est seulement nécessaire de définir la topologie intra-slice (au sein d'une *time slice*), la topologie inter-slice (entre deux *time slice*), ainsi que les paramètres (*i.e* les probabilités conditionnelles, voir équation (3)), pour les deux premières *time slice*. Le résultat est un 2TBN comme montré en figure 1.

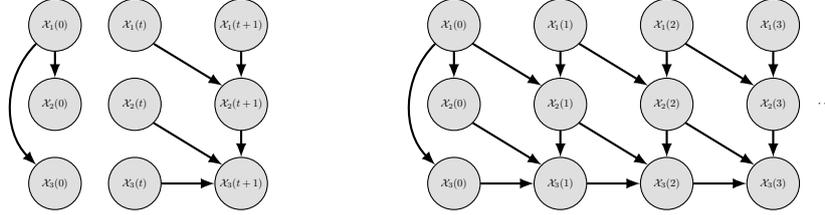


Figure 1. Un 2 Time Slice BN (2TBN) et le réseau bayésien dynamique (déroulé) associé

Dans cet article, nous considérons que les variables aléatoires $X_i(t)$ sont discrètes et P_{ij}^t est la probabilité que $X_i(t) = k$ pour l'instanciation j de ses parents, *i.e.*

$$P_{ijk}^t = P(X_i(t) = k \mid \mathbf{U}_i(t) = j), \quad \begin{array}{l} i = 1, \dots, n \\ j = 1, \dots, c_i \\ k = 1, \dots, r_i \end{array} \quad (4)$$

où r_i est le nombre de valeurs que le nœud $X_i(t)$ peut prendre et c_i le nombre d'instanciations distinctes de $\mathbf{U}_i(t)$.

L'apprentissage d'un DBN $\mathcal{B} : (\Theta^*, \mathcal{G}^*)$ de paramètres Θ^* et structure \mathcal{G}^* depuis une base de donnée \mathcal{D} revient à résoudre $\mathcal{G}^* = \arg \max_{\mathcal{G}} P(\mathcal{G} \mid \mathcal{D})$, avec $P(\mathcal{G} \mid \mathcal{D}) \propto P(\mathcal{D} \mid \mathcal{G})P(\mathcal{G})$. $P(\mathcal{G})$ permet d'imposer des contraintes sur la structure du graphe et est autrement laissé uniforme. $P(\mathcal{D} \mid \mathcal{G})$ est la vraisemblance marginale et est définie par les paramètres $\Theta_i = P_i^t = P(X_i \mid \mathbf{U}_i, \Theta_i)$:

$$P(\mathcal{D} \mid \mathcal{G}) = \int P(\mathcal{D} \mid \Theta, \mathcal{G})P(\Theta \mid \mathcal{G})d\Theta \quad (5)$$

L'équation (5) peut se résoudre de manière approchée par l'utilisation d'un score tel que BIC (Schwarz, 1978), AIC (Akaike, 1998) ou MDL (Rissanen, 1978) qui cherche à compresser les données \mathcal{D} , ou de manière exacte en utilisant des distributions a priori et posteriori conjuguées (les distributions de probabilité a priori et a posteriori sont de la même famille) tel que Dirichlet et le score *Bayesian-Dirichlet* (BDeu).

L'équation (5) devient, avec un a priori de Dirichlet (Heckerman *et al.*, 1995) sur Θ :

$$P(\mathcal{D} \mid \mathcal{G}) = \prod_i^n \prod_j^{c_i} \frac{\Gamma(\alpha_{ij})}{\Gamma(N_{ij} + \alpha_{ij})} \prod_k^{r_i} \frac{\Gamma(N_{ijk} + \alpha_{ijk})}{\Gamma(\alpha_{ijk})} \quad (6)$$

avec α_{ijk} l'a priori sur N_{ijk} .

La résolution de cette équation se fait souvent par heuristiques pour diminuer l'espace de recherche, tel que l'algorithme *greedy hill climbing* (Chickering *et al.*, 1995)

où un arc est modifié et le score calculé à nouveau jusqu'à ce qu'aucun mouvement n'améliore le score.

Les DBN ont été appliqués dans des domaines variés tels la reconnaissance vocale (Mitra *et al.*, 2011), la détection de pannes (Lerner *et al.*, 2000), le diagnostic médical (Charitos *et al.*, 2009) ou la biologie (Sicard *et al.*, 2011) mais leur application dans le cadre des Systèmes de Détection d'Intrusion est rare (An *et al.*, 2006). Cependant, les modèles de Markov (cachés) ont été extensivement proposés pour modéliser des traces d'appels système et de commandes shell (Yeung, Ding, 2003 ; Zanero, Serazzi, 2008), ou du flux réseau (Ourston *et al.*, 2003). Dans ce domaine, les réseaux bayésiens sont principalement utilisés de manière statique et souvent à des fins de classification ou de mécanisme décisionnel agrégeant les résultats de modèles plus petits, qui offrent un condensé des données d'entrée (Kruegel *et al.*, 2003 ; Mutz *et al.*, 2006).

Une variante des DBN, les *Continuous Time Bayesian Networks* (CTBN), ont été utilisées pour modéliser du trafic réseau (Xu, Shelton, 2008). Les CTBN tirent parti du temps continu pour résoudre le problème de granularité temporelle des DBN qui nécessitent un intervalle de temps constant entre deux *time slice*, ce qui peut les rendre inefficaces en temps de calcul lorsque l'on considère de longues périodes "d'inactivité" ou des observations irrégulièrement espacées temporellement. Le principal désavantage de ce modèle est que deux variables ne peuvent changer d'état simultanément et qu'un paramètre doit être choisi pour définir l'échelle de temps durant laquelle des corrélations peuvent exister.

Le travail dans Xu et Shelton (2008 ; 2010) est proche du nôtre dans leur approche : l'utilisation d'une variable cachée permet de modéliser l'état inconnu de la machine - la structure du réseau est spécifiée manuellement et n'évolue pas. Après apprentissage, une fenêtre glissante est utilisée pour calculer la vraisemblance des observations sous le modèle. Sont annotées anormales toutes observations dont la vraisemblance est sous un certain seuil. Cependant, le nombre d'états de la variable cachée doit être connu à l'avance (deux dans l'article) et de nouveaux modèles ne peuvent être découverts à la volée. En outre, il n'y a aucun mécanisme de raffinement lorsque les fenêtres chevauchent des observations de différents modèles, que le seuil de vraisemblance soit dépassé ou non. Il est fort probable, lorsqu'une fenêtre chevauche une transition, qu'une partie des données à la suite de cette transition ne compense pas suffisamment la (haute) vraisemblance à gauche, ce qui donne une vraisemblance globale acceptable et introduit du bruit dans les modèles, qui à long terme rend les détections futures plus difficiles.

2.2. Réseaux bayésiens dynamiques non stationnaires

Un ns-DBN propose de considérer une homogénéité par période du modèle dynamique. Il est donc représenté par une famille de DBN $\mathcal{B} : (\Theta, \mathcal{G})$ de structure \mathcal{G} et de paramètres Θ organisés par périodes (ou époques) \mathcal{T} :

$$nsDBN = \{(\mathcal{B}_m : (\Theta_m, \mathcal{G}_m), \mathcal{T}_m)\}_{m \in \mathcal{M}}$$

L'ensemble des modèles \mathcal{M} est fini à tout temps t donné et grandit avec le temps. Pour tout $(m_i, m_j) \in \mathcal{M}^2$ on a $D(\mathcal{B}_{m_i}, \mathcal{B}_{m_j}) > \epsilon$, avec D une fonction de la famille des f -divergences (comme la distance de Hellinger (Eq. 7) (Beran, 1977) ou la divergence de Kullback-Leibler (Kullback, Leibler, 1951)). En effet il n'existe pas deux modèles dans \mathcal{M} ayant la même dynamique. La complexité des algorithmes d'apprentissage des ns-DBN dépend de la cardinalité de \mathcal{M} (d'autant plus lorsque le nombre de transitions et leur temps d'arrivé ne sont pas connus).

$$H(P, Q) = \frac{1}{\sqrt{2}} \sqrt{\sum_i (\sqrt{p_i} - \sqrt{q_i})^2} \quad (7)$$

Il n'existe pas encore de cadre pour modéliser la dynamique des transitions pour les ns-DBN. En conséquence, les ns-DBN représentent des processus non stationnaires en supposant une stationnarité par morceau (par époque), ce qui semble être une hypothèse raisonnable. On suppose aussi que la fréquence d'observations du processus est nettement supérieure à sa fréquence de changement de dynamique. Ils héritent en outre des avantages et inconvénients des DBN.

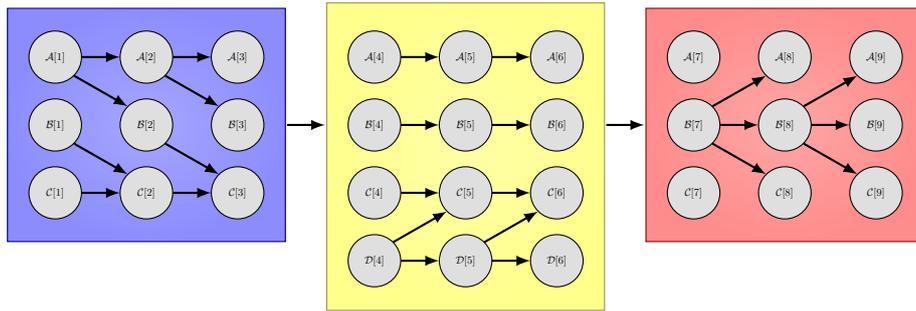


Figure 2. Réseau bayésien dynamique non stationnaire (ns-DBN) avec 3 différentes époques. Des DBN d'époques différentes peuvent avoir des paramètres, des structures et même des variables différentes. Pour des raisons de simplicité, les BN a priori comme en figure 1 ne sont pas représentés

Bien que, dans un ns-DBN, différentes époques peuvent avoir des jeux de paramètres, des structures et même des variables différentes (voir figure 2), Grzegorzcyk et Husmeier (2009) se focalise sur l'évolution des paramètres sous une structure fixe. Dans Robinson et Hartemink (2010), l'accent est mis sur l'évolution structurelle, douce, du réseau. Sous ces approches, le nombre de variables et leurs domaines restent constants avec le temps (même si certaines ne sont pas observées au cours d'époques entières). Gonzales *et al.* (2015) ont une approche similaire à la nôtre, autorisant la structure, les paramètres ainsi que les variables et leurs domaines à varier au cours du temps. Cependant, nous utilisons des critères différents ainsi qu'un mécanisme pour les fenêtres chevauchant des événements de différents modèles afin de raffiner les moments de transition détectés. En outre, nous souhaitons pouvoir identifier si un modèle

est récurrent ou exceptionnel au lieu d'en créer un nouveau après chaque transition détectée.

Les algorithmes d'apprentissage pour les ns-DBN consistent à identifier les différentes époques et les DBN associés à chacune. Les algorithmes actuels se concentrent soit sur l'évolution de la structure soit sur celle des paramètres (principalement pour réduire l'espace de recherche du problème). Malheureusement ces algorithmes nécessitent, pour ainsi dire tous, la disponibilité de toute la base de données et ne peuvent être utilisés dans le cadre de flux de données (apprentissage *online*), en temps réel, sauf peut-être pour un premier apprentissage afin de s'assurer que la première fenêtre de données ne provient pas déjà de plusieurs modèles distincts. Ces méthodes n'offrent pas non plus la possibilité de réutiliser un DBN appris précédemment. L'apprentissage *offline* des ns-DBN est habituellement réalisé en utilisant un score traditionnel (*i.e.* BDeu) modifié pour prendre en compte les statistiques suffisantes qui ont besoin d'être spécifiées par époque pour trouver les meilleures transitions ainsi qu'un ensemble de mouvement étendu pour l'apprentissage de la structure, qui doit elle aussi être spécifiée par époque.

Ainsi, Robinson et Hartemink (2010) modifient l'équation (6) pour tenir compte des différentes structures $\mathcal{G}_1, \dots, \mathcal{G}_m$ séparées par les transitions \mathcal{T} , en supposant l'indépendance mutuelle des paramètres d'une époque à une autre et des variables de domaine stable :

$$P(\mathcal{D} \mid \mathcal{G}_1, \dots, \mathcal{G}_m, \mathcal{T}) \propto \prod_h^m \prod_i^n \prod_j^{c_i} \frac{\Gamma(\alpha_{ij}(I_h))}{\Gamma(N_{ij}(I_h) + \alpha_{ij}(I_h))} \prod_k^{r_i} \frac{\Gamma(N_{ijk}(I_h) + \alpha_{ijk}(I_h))}{\Gamma(\alpha_{ijk}(I_h))} \quad (8)$$

avec I_h l'intervalle sur \mathcal{D} .

L'apprentissage est réalisé par échantillonnage (Grzegorzcyk, Husmeier, 2009; Robinson, Hartemink, 2010). En effet, lorsque les temps de transitions ne sont pas connus à priori, la distribution a posteriori sur les structures est trop complexe pour un calcul exact (Robinson, Hartemink, 2010). La stratégie d'échantillonnage est différente selon que le nombre de transitions et/ou leur temps d'arrivées sont connus ou non (plus il y a d'incertitude et plus la stratégie est complexe).

Cependant, nous avons besoin de règles plus simples pour atteindre des performances temps réel avec un flux d'observations continu. Gonzales *et al.* (2015) propose une approche intéressante : prendre en compte la force des arcs en utilisant l'information mutuelle d'un nœud et de ses parents et utiliser les paramètres précédents comme a priori pour les nœuds qui ne changent pas d'un modèle à l'autre. Davantage d'exemples sont nécessaires puisque l'information peut se propager différemment dans un réseau selon les nœuds observés.

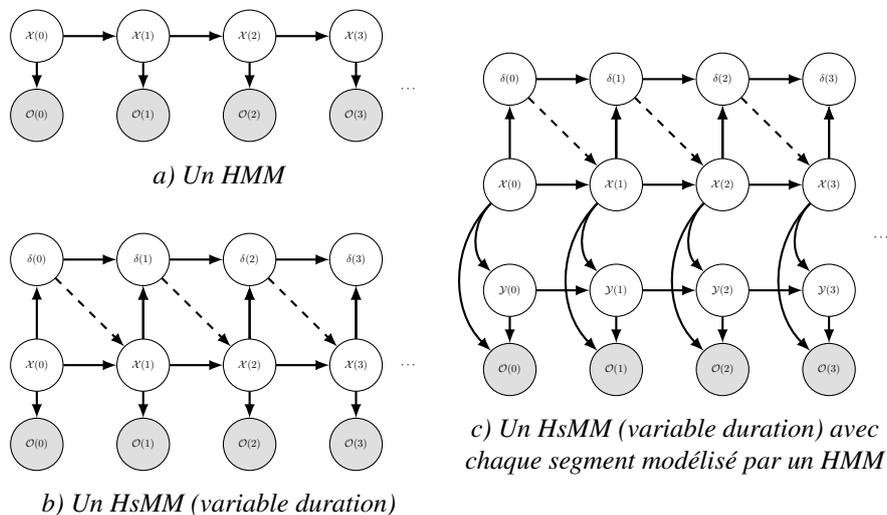


Figure 6. a) Un HMM, b) un HsMM (variable duration) et c) un HsMM-HMM (variable duration)

2.3. Granularité temporelle et problèmes de modélisation

Nous avons vu qu'un ns-DBN peut avoir des jeux de structure et paramètres différents d'une époque à l'autre. Cependant, il est aussi possible que les processus sous-jacents n'aient pas la même fréquence d'échantillonnage inter-époques et même intra-époque. Dans certaines applications, telles que la météorologie, la bio-informatique ou la physique, le temps est vraisemblablement une donnée importante et la granularité temporelle est adaptée. Dans beaucoup d'autres contextes, notamment lorsque l'échantillonnage n'est pas sous contrôle (systèmes informatiques, relevés de capteurs) et ne peut pas être synchronisé entre divers équipements, seule la séquence des événements est importante. Le temps qui sépare une *time-slice* d'une autre n'est pas constant et n'a que peu de sens ; comment prendre en compte le temps qui sépare deux instructions sur un ordinateur sachant qu'on peut parler de temps CPU (qui ne tient pas compte des opérations d'entrée-sortie ou du *multi-threading*) ou de temps humain (qui fait fi de l'ordonnanceur du système, qui change de processus à la volée). Ceci n'est pas un problème dans notre cas puisque nous sommes davantage intéressés par le déroulement du processus (inter-mode) que par sa temporalité (intra-mode). Comme mentionné dans (Xu, Shelton, 2008), il n'existe pas toujours de procédé justifiable pour intégrer le temps entre événements dans un DBN selon le domaine d'application.

2.4. *Hidden Markov Models (HMM), Hidden Semi-Markov Models (HsMM) et ns-DBN*

Les HsMM sont une extension des HMM (Baum, Petrie, 1966 ; Rabiner, 1989) avec des durées explicitées par la présence d'une variable supplémentaire. Chaque état a une durée variable et produit une séquence d'observations dont le nombre est déterminé par la durée passée dans l'état, contrairement à une observation unique dans le cadre des HMM. La probabilité de changer d'état peut maintenant aussi dépendre du temps passé depuis l'arrivée dans cet état (si l'arc est présent, pour les HsMM *variable duration*). Ceci permet aux HsMM de représenter des structures et dépendances temporelles complexes et/ou de longues durées, décrites par des distributions arbitraires et non limitée à des exponentielles, ainsi que des processus non homogènes. Murphy (2002b) et Yu (2010) réalisent une vue d'ensemble sur les HsMM, leur apprentissage et leurs applications. La figure 6a montre un HMM, avec \mathcal{X} l'état caché et \mathcal{O} l'observation, la figure 6b un HsMM avec durée explicite δ et la figure 6c un HsMM avec des observations qui ne sont plus conditionnellement indépendantes ; chaque segment est modélisé par un HMM comme en figure 6a (Murphy, 2002b). Il est possible de ne pas avoir de dépendance entre les observables de deux segments.

Dans le cadre des DBN la séquence d'observations est modélisée par un DBN pour donner un modèle montré en figure 7. Il est possible d'ajouter un arc pour faire dépendre la durée δ d'un mode du mode précédent, pour permettre de capturer des relations plus complexes entre modes. Ce modèle n'est pas un DBN valide dans le cas général des ns-DBN - avec évolution de la structure et des paramètres - puisque la structure du DBN (en gris) n'est pas fixe. En revanche, nous pourrions utiliser un graphe complet et introduire une variable de sélection des parents afin de garder une structure fixe d'un mode à l'autre, comme dans le cadre des Probabilistic Relational Models (PRM) sous incertitude structurelle (Getoor *et al.*, 2001). Nous verrons en section 4 qu'il est possible de représenter les transitions entre modes en utilisant une variable de durée explicite et en découplant les modèles (des observations) de la dynamique des modes, c'est-à-dire d'apprendre les différents modèles d'observations (apprentissage de ns-DBN) puis d'apprendre la dynamique des transitions. L'apprentissage des ns-DBN prend en compte la complexité des modèles appris pour pénaliser la vraisemblance globale et prend donc en compte indirectement leur durée. Le modèle de transitions entre modes est d'avantage un outil de modélisation dans le contexte des ns-DBN.

3. Apprentissage de processus non stationnaires

Nous présentons dans cette section un nouvel algorithme pour l'apprentissage en ligne de réseaux bayésiens dynamiques non stationnaires. Dans la littérature (Grzegorzcyk, Husmeier, 2009 ; Robinson, Hartemink, 2010 ; Grzegorzcyk, Husmeier, 2011), l'hypothèse que deux modèles temporellement adjacents sont gouvernés par des distributions et/ou une structure similaires est souvent faite. Cependant, nous ne fe-

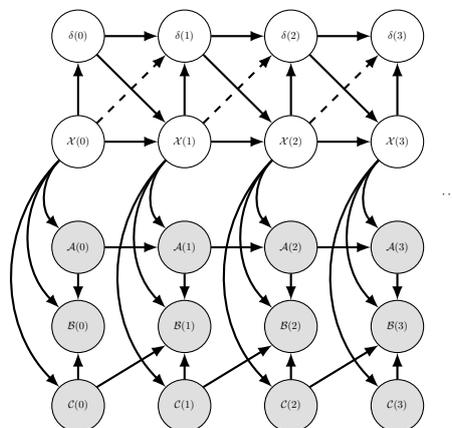


Figure 7. Un HsMM avec des observations modélisées par un DBN

rons pas ici cette hypothèse et ne nous restreindrons donc pas à des évolutions douces d'un modèle à l'autre.

Les données sont reçues de manière continue et remplissent une fenêtre w de taille r avant d'être traitées. Pour chaque nouvelle fenêtre d'observations, notre algorithme doit choisir entre utiliser un modèle déjà connu ou en créer un nouveau. Ce choix est basé sur la vraisemblance des données fenêtrées. En effet, nous nous attendons à ce que la vraisemblance d'un modèle décroît si le comportement sous-jacent change (voir figure 8). L'algorithme commence par une phase d'initialisation qui fournit un premier réseau servant de point de départ.

Plus formellement, à tout instant τ , l'algorithme confronte une collection de M modèles (DBN) $\{\mathcal{B}_m : (\Theta, \mathcal{G})_m\}_M$ à la fenêtre de données $w[\tau, \tau + r]$. Nous sommes intéressés par le caractère récurrent ou exceptionnel des modèles et ne pouvons simplement en créer de nouveaux après chaque déviation rencontrée, c'est pourquoi nous avons besoin de savoir s'il existe déjà un modèle m acceptable pour w , potentiellement différent du modèle courant. Par ailleurs, bien évidemment, notre algorithme doit être capable de créer de nouveaux modèles lorsque ceux existants ne décrivent pas correctement les trajectoires observées dans w .

Ainsi, à partir d'un flux d'observation, nous attendons de notre algorithme de fournir : i) une ensemble de dBN, ii) un ensemble de fenêtres temporelles découpant le flux et iii) pour chaque fenêtre temporelle, une sélection du dBN qui décrit au mieux le comportement du signal dans la fenêtre. On remarque qu'il est alors facile de discriminer entre des dBN récurrents (caractérisant beaucoup de fenêtres) et des dBN exceptionnels (caractérisant peu de fenêtres, voire une seule fenêtre).

3.1. Apprentissage avec variables fixes et fenêtre de taille fixe

Soit un ensemble M de DBN, il s'agit, pour chaque fenêtre, soit de trouver $m^* \in M$ le modèle décrivant au mieux les observations de la fenêtre, soit d'apprendre un nouveau modèle m' qu'on ajoutera alors à la liste des modèles possibles. Dans un premier temps, nous supposons que les variables suivies dans le flux ne changent pas au cours du temps. Nous supposons également que la taille de la fenêtre temporelle est fixe.

Pour évaluer à quel point un modèle m est capable d'expliquer les données $\mathbf{w}[\tau, \tau + r]$, nous utilisons un critère simple basé sur la vraisemblance des données \mathbf{w} . Dans un DBN stationnaire, la log-vraisemblance des données vis-à-vis d'un réseau de structure \mathcal{G} et de paramètres Θ est :

$$LL(\mathbf{w} : \Theta, \mathcal{G}) \propto \sum_{t=\tau}^{\tau+r} \sum_{i,j,k} N_{ijk} \log(\theta_{ijk}) \quad (9)$$

où $\theta_{ijk} = P(X_i(t) = k \mid \mathbf{U}_i(t) = j)$ et N_{ijk} est le nombre de cas où $X_i(t) = k$ et $\mathbf{U}_i(t) = j$ dans \mathbf{w} .

Pour chaque DBN $\mathcal{B}_m : (\Theta_m, \mathcal{G}_m)$, nous calculons donc $LL(\mathbf{w} : \Theta_m, \mathcal{G}_m)$. Le meilleur modèle \mathcal{B}_{m^*} pour w ne peut cependant pas encore être sélectionné en maximisant LL dans l'équation (9) puisque l'algorithme peut/doit aussi créer des modèles à la volée.

À cet effet, on peut remarquer que la distribution de la log-vraisemblance d'une fenêtre \mathbf{w} est approximativement normalement distribuée (Duan *et al.*, 2009; Liu, Leimeire, 2017), centrée sur l'entropie du modèle (comme la somme de $r + 1$ variables aléatoires indépendamment et identiquement distribuées, en utilisant le théorème central limite). Nous proposons donc un test d'hypothèses statistique dans le but de trouver la log-vraisemblance p -value LL_{tr} tel que 99 % des appariements se font avec une log-vraisemblance égale ou supérieure (voir figure 8). Contrairement aux travaux de Johansson et Olofsson (2007) ou Duan *et al.* (2009), nous sommes d'avantage intéressés par la borne inférieure que supérieure (des observations plus vraisemblables que de *moyenne* ne posent pas de problème pour notre sélection). Pour calculer une première estimation de LL_{tr} après avoir découvert un nouveau modèle, nous utilisons l'échantillonnage de Gibbs (Casella, George, 1992) : des trajectoires sont échantillonnées depuis le DBN, autant que nécessaire pour remplir plusieurs fenêtres, avant de calculer leurs vraisemblances en faisant glisser la fenêtre. Pour chaque modèle \mathcal{B}_m , nous calculons donc $\frac{LL_{tr}(\Theta_m, \mathcal{G}_m)}{LL(\mathbf{w} : \Theta, \mathcal{G})}$. Notre règle de sélection devient :

$$m^* = \arg \max_{m \in \mathcal{M}_\alpha} LL(\mathbf{w} : \Theta_m, \mathcal{G}_m)$$

$$\text{où } \mathcal{M}_\alpha = \left\{ m \text{ t.q. } \frac{LL_{tr}(\Theta_m, \mathcal{G}_m)}{LL(\mathbf{w} : \Theta_m, \mathcal{G}_m)} \geq \alpha \right\} \quad (10)$$

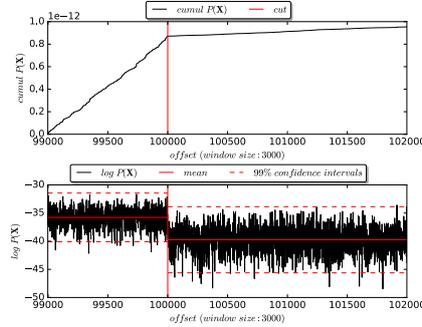


Figure 8. Cumulé de $P(\mathbf{X})$ et $\log P(\mathbf{X})$ pour une fenêtre chevauchant deux comportements distincts. La ligne rouge verticale représente une transition entre deux modèles et les lignes horizontales les gaussiennes avec des intervalles de confiance à 99 %

Avec α le paramètre de seuil. Nous utilisons un seuil à 0,97 plutôt que 1 sur le ratio de vraisemblance puisque le premier apprentissage est, selon la taille initiale de la fenêtre, imprécis (peu d'événements observés pour potentiellement beaucoup de paramètres à apprendre) et nous autorisons une certaine divergence de cet ordre de grandeur. Ce seuil pourrait être dynamique et autoriser de moins en moins de divergence entre vraisemblance observée et vraisemblance attendue. Plus α est grand plus les modèles découverts sont distants.

Si l'ensemble des modèles \mathcal{M}_α dans l'équation (10) est vide, aucun modèle actuellement appris ne représente correctement les observations de la fenêtre courante. L'algorithme apprend alors un nouveau DBN et le sélectionnera comme modèle courant. Par contre, si un modèle \mathcal{B}_m déjà présent est sélectionné, il y a quand même une phase d'apprentissage afin de mettre à jour les paramètres (et éventuellement la structure) au regard des nouvelles données. Nous n'utilisons pas $\Theta_{m_{t-1}^*}$ ni $\mathcal{G}_{m_{t-1}^*}$ comme a priori sur $P(\mathcal{G}_{m_t})$ laissant la structure, le domaine des variables et les paramètres évoluer sans contraintes. En effet, au fur et à mesure que les observations augmentent en quantité pour les modèles, leur structure doit être ré-évaluée : nous choisissons de ré-estimer la structure (et les paramètres) dès que la quantité d'observations a doublé depuis le dernier apprentissage de structure (voir ligne 8 Algorithm 1).

Durant les expériences, la figure 16 montre comment une fenêtre de mauvaise taille peut perturber l'apprentissage. En effet, si quelques points en fin de fenêtre appartiennent à un autre modèle, leur nombre et/ou effet peut ne pas être suffisamment important par rapport à la taille totale de la fenêtre pour rejeter l'hypothèse nulle. Nous proposons donc, dans un second temps, d'ajouter un mécanisme d'adaptation de la taille de la fenêtre au cours de l'apprentissage.

3.2. Fenêtre dynamique adaptative

Lorsque l'algorithme prédit, pour la fenêtre courante, un modèle $\mathcal{B}_{m_i^*}$ différent du modèle $\mathcal{B}_{m_{i-1}^*}$ de la fenêtre précédente (*i.e.* $\mathcal{B}_{m_i^*}$ est soit un DBN nouvellement créé ou un DBN déjà connu), la prédiction porte non seulement sur le changement de dynamique du processus mais aussi sur le temps τ auquel ce changement survient (le point de changement). Dans cette section, nous proposons d'investiguer davantage la valeur de ce point en regardant la distribution de la vraisemblance dans $\mathbf{w}[\tau - r, \tau + r]$.

Un point de changement représente une transition d'un état à un autre du processus qui génère ces données. La figure 8 montre la valeur cumulée de $P(\mathbf{X})$ et $\log P(\mathbf{X})$ pour une telle fenêtre avec un point de changement ($c = 100\,000$). En supposant les données i.i.d. la série temporelle $\log P(\mathbf{X}(t))$ est stationnaire et les paramètres du modèles sont donc constants. Si la série n'est pas stationnaire alors elle provient de deux modèles distincts. Pour trouver une valeur correcte de c , on pourrait caractériser le changement de pente dans le cumul de $P(\mathbf{X})$. Cependant, pour être plus précis, nous maximisons sur c la vraisemblance d'un modèle où le point de changement c sépare deux processus stationnaires distincts (ici gaussiens) avec μ_t constant qui ne dépend pas de t , et la fonction de covariance qui ne dépend que de la différence de temps entre deux mesures.

L'algorithme est itératif et garde en mémoire les k dernières fenêtres passées de la distribution de la log-vraisemblance pour un modèle donné (l'effet et le choix de k sont discutés en section 5). Pour trouver un premier point de changement dans $\mathbf{w}[\tau - r, \tau + r]$, l'algorithme construit une fenêtre D de taille au plus $(k + 2)r$ (certaines fenêtres passées peuvent être de taille inférieure à r si un point de changement avait été trouvé). L'algorithme cherche un premier point en maximisant sur $c \in [\tau - r, \tau + r]$ la vraisemblance de deux modèles gaussiens, l'un à gauche de c sur $[[0, \dots, kr, \dots, c[[$ et l'autre à droite sur $[[c, (k + 2)r]$.

La log-vraisemblance des données D sous une loi normale est, lorsque μ et σ sont estimés depuis D :

$$LL(D, \mu, \sigma) \propto -\frac{n}{2} \cdot (\log(\sigma^2) + \log(2\pi) + 1)$$

c sépare donc deux processus gaussiens tel que la log-vraisemblance sous ces deux modèles est meilleure que la log-vraisemblance sous un seul modèle :

$$LL(D[0, c, \mu_1, \sigma_1]) + LL(D[c, (k + 2)r, \mu_2, \sigma_2]) \geq \alpha \cdot LL(D, \mu, \sigma)$$

avec α un paramètre de coût / pénalité pour la création d'un point de changement (l'effet et le choix de ce paramètre sont discutés en section 5). Un point n'est valide que si la log-vraisemblance moyenne à gauche n'est pas plus basse que la log-vraisemblance moyenne à droite. En effet, nous ne souhaitons pas avoir de point de changement lorsque les données observées sont plus vraisemblables que par le passé.

Soit $f(c) = LL(D[0, c], \mu_1, \sigma_1) + LL(D[c, (k+2)r], \mu_2, \sigma_2) - \alpha \cdot LL(D, \mu, \sigma)$.
À chaque itération, l'algorithme cherche donc c^* tel que :

$$c^* = \arg \max_{c \in \mathcal{C}} f(c)$$

$$\text{avec } \mathcal{C} = \left\{ c \text{ t.q. } \begin{array}{l} f(c) \geq 0 \\ \mu_1 \geq \mu_2 \end{array} \right\} \quad (11)$$

Après avoir trouvé un point de changement, l'algorithme recommence à gauche de ce point (nous avons toujours les fenêtres passées). Nous ne cherchons qu'à gauche puisque nous sommes intéressés par le premier point à partir duquel le signal de log-vraisemblance ne correspond plus au modèle courant. Lorsqu'aucun point n'est trouvé, l'algorithme se termine.

Nous mettons ensuite à jour $\mathcal{B}_{m_{t-1}^*}$ sur $w[\tau-r, \tau-r+c^*]$ avec le point de changement optimisé c^* le plus à gauche. Si $\mathcal{B}_{m_t^*}$ est un nouveau modèle. Nous utilisons un a priori de Dirichlet non informatif et nous supposons que les paramètres et la structure évoluent sans corrélations d'un modèle à un autre.

Dans la section 5 nous montrons qu'il est possible de se passer de l'équation (10) et d'utiliser directement l'équation (11). La règle de sélection se base maintenant sur la log-vraisemblance moyenne avant c^* et devient alors :

$$m^* = \arg \max_{m \in M_c} \frac{LL(w[0, c_m^*] : \Theta_m, \mathcal{G}_m)}{c_m^*}$$

$$\text{avec } M_c = \left\{ m \text{ t.q. } \begin{array}{l} \mathcal{B}_{m_t} = \mathcal{B}_{m_{t-1}^*} \Rightarrow c_m^* \geq 0 \\ \mathcal{B}_{m_t} \neq \mathcal{B}_{m_{t-1}^*} \Rightarrow c_m^* \geq c_{min} \end{array} \right\} \quad (12)$$

Si $\mathcal{B}_{m_t} = \mathcal{B}_{m_{t-1}^*}$ il nous faut alors $c_m^* \geq 0$ ce qui signifie qu'on peut avoir ou ne pas avoir un point de changement pour le modèle courant. Si $\mathcal{B}_{m_t} \neq \mathcal{B}_{m_{t-1}^*}$ alors $c_m^* \geq c_{min}$; il ne peut y avoir de point de changement pour un autre modèle \mathcal{B}_{m_t} que le courant ($\mathcal{B}_{m_{t-1}^*}$) tant que \mathcal{B}_{m_t} n'a pas été élu modèle courant ou que $c_{m_{t-1}^*}^* = 0$ pour $\mathcal{B}_{m_{t-1}^*}$. Il ne pourra être élu modèle courant que si on trouve un point de changement pour $\mathcal{B}_{m_{t-1}^*}$ et que la prochaine fenêtre (ou courante si $c_{m_{t-1}^*}^* = 0$) est fidèle à \mathcal{B}_{m_t} sur les c_{min} premières observations. c_{min} contrôle donc la durée minimale d'un modèle en taille de fenêtre. c_{min} évite de sélectionner, après un point de changement, un autre modèle pour seulement quelques observations (sans signification statistique donc) avant de changer à nouveau de modèle (celui qui aurait dû être sélectionné). Cet effet est montré dans la section 5, Expériences et résultats, lorsqu'aucune contrainte n'est mise sur c_m^* avec $\mathcal{B}_{m_t} \neq \mathcal{B}_{m_{t-1}^*}$. Dans un premier temps, $c_{min} = r$ et donc $c_m^* = r$ si $\mathcal{B}_{m_t} \neq \mathcal{B}_{m_{t-1}^*}$.

Si tous les modèles candidats ont un point de changement immédiat ou invalide ($c_{m_i}^* = 0, c_{m_j}^* < c_{min}$) alors un nouveau modèle est appris.

Autrement dit, nous pouvons continuer d'apprendre $\mathcal{B}_{m_{t-1}^*}$ jusqu'à trouver un point de changement, après quoi nous cherchons si un modèle (peut-être le même

après mise à jour des paramètres et de la structure) le remplace ou si nous avons besoin d'en créer un nouveau, et ainsi de suite. C'est donc seulement à la suite d'un point de changement que nous cherchons parmi tous les modèles déjà connus.

Algorithme 1 : Boucle principale

Données : modèle précédent m_{t-1}^* , observations $\mathbf{w}[\tau - r, \tau]$, $\mathbf{w}[\tau, \tau + r]$

Données : $\mathbf{D} = \{\mathcal{B}_m\}$, $\mathcal{B}_{m_{t-1}^*}$, α (seuil de vraisemblance)

```

1 début
2    $\mathcal{M}_\alpha \leftarrow \text{compare modèles}(\mathbf{D}, \mathbf{w})$ 
3   si  $\mathcal{M}_\alpha \neq \{\}$  alors
4      $m_t^* \leftarrow \arg \max_m \{LL(\mathbf{w} : \Theta, \mathcal{G}) : (LL, LL_{tr}, m) \in \mathcal{M}_\alpha\}$  (Eq. 10)
5     si  $m_t^* \neq m_{t-1}^*$  alors
6       trouver la coupure  $c$  sur  $\mathbf{w}[\tau - r, \tau + r]$  (Eq. 11)
7       mettre à jour le modèle précédent  $\mathcal{B}_{m_{t-1}^*}$  sur  $\mathbf{w}[\tau - r, \tau - r + c]$ 
8     si nouvelles observations  $\geq 2 * \text{observations précédentes}$  alors
9       mettre à jour structure et paramètres de  $\mathcal{B}_{m_t^*}$  sur
10       $\mathbf{w}[\tau - r + c, \tau + r]$ 
11      observations précédentes  $\leftarrow$ 
12      observations précédentes + nouvelles observations
13     sinon
14      mettre à jour paramètres de  $\mathcal{B}_{m_t^*}$  sur  $\mathbf{w}[\tau - r + c, \tau + r]$ 
15     retourner
16   trouver la coupure  $c$  sur  $\mathbf{w}[\tau - r, \tau + r]$  (Eq. 11)
17   mettre à jour le modèle précédent  $\mathcal{B}_{m_{t-1}^*}$  sur  $\mathbf{w}[\tau - r, \tau - r + c]$ 
18   apprendre nouveau modèle sur  $\mathbf{w}[\tau - r + c, \tau + r]$ 

```

3.3. Apprentissage avec variables de domaines incompatibles

Comme le montre la figure 2, le nombre de variables actives peut changer au cours du processus. Dans ce cas, on peut être amené à confronter un modèle et une base de données avec un nombre différent de variables. Si les variables de la base de données forment un sous-ensemble des variables du modèle, avec X_e dans \mathcal{G}_m mais pas dans la base, nous réalisons des inférences pour estimer $P(X_i | \mathbf{U}_i \setminus X_e)$ et calculons ensuite les vraisemblances.

Si les variables du modèle forment un sous-ensemble des variables de la base, ces informations ne sont pas exploitables par le modèle et sont simplement mises de côté. Un tel modèle ne sera pas sélectionné pour la fenêtre courante. Si le domaine des variables Ω_{X_i} diffère, nous ajoutons les états manquants en utilisant les paramètres a priori (non informatifs) de Dirichlet α_{ijk} et calculons ensuite les vraisemblances.

Les algorithmes 1, 2 et l'équation 12 décrivent notre cadre général pour l'apprentissage de processus non stationnaires en temps réel par ns-DBN. Bien que la pro-

chaîne section détaille nos expériences, il est intéressant de noter que la complexité de notre algorithme ne dépend pas de la taille de la base de données mais seulement de la taille de la fenêtre et du nombre de modèles connus, ce qui est une qualité importante pour l'apprentissage en ligne.

Algorithme 2 : Compare modèles

Données : observations $\mathbf{w}[\tau, \tau + r]$

Données : $\mathbf{D} = \{\mathcal{B}_m\}$, $Dir(\{\alpha_{ijk}\})$ (paramètres de Dirichlet), α
(seuil de vraisemblance)

1 **début**

2 $\mathcal{M}_\alpha \leftarrow \{\}$

3 **pour** $\mathcal{B}_m = (\Theta, \mathcal{G})_m \in \mathbf{D}$ **faire**

4 **tant que** $\exists X_e \in \mathcal{B}_m, X_e \notin \mathbf{w}$ **faire**

5 $\forall j \in \llbracket 1, c_e \rrbracket$, éliminer X_e par inférence :
6 $P(X_i | (\mathbf{U}_i \setminus X_e) = j)$

7 **tant que** $\exists X_e \in \mathbf{w}, X_e \notin \mathcal{B}_m$, **faire**

8 \mid retirer X_e

9 **tant que** $\exists X_i \in \mathbf{w}, X_i = k$ **and** $X_i \in \mathcal{B}_m, k \notin \Omega_{X_i}$ **faire**

10 $\Omega_{X_i} \leftarrow \Omega_{X_i} \cup k$

11 $\theta_{ijk} \leftarrow \frac{\alpha_{ijk}}{N_{ij} + \alpha_{ij}}$

12 $\theta_{ij\{o \neq k\}} \leftarrow \frac{N_{ijo} + \alpha_{ijo}}{N_{ij} + \alpha_{ij}}$

13 **pour** $X_l \in \mathcal{B}_m : X_l \in \mathbf{U}_l$ **faire**

14 $\mid \forall j \in \llbracket 1, c_i \rrbracket$, calculer par inférence :

15 $\mid P(X_l | (\mathbf{U}_l \setminus X_l) = j, X_l = k) \leftarrow P(X_l | (\mathbf{U}_l \setminus X_l) = j)$

16 compute $LL_{tr}(\Theta, \mathcal{G})$

17 compute $LL(\mathbf{w} : \Theta, \mathcal{G})$ from Eq. 9

18 **si** $\frac{LL_{tr}(\Theta, \mathcal{G})}{LL(\mathbf{w} : \Theta, \mathcal{G})} \geq \alpha$ **alors**

19 $\mid \mathcal{M}_\alpha \leftarrow \mathcal{M}_\alpha \cup (LL, LL_{tr}, m)$

20 *Return* \mathcal{M}_α

4. Graphical duration models

Pour représenter les durées de séjours et les transitions entre modes, nous utilisons un Graphical Duration Model (GDM), introduits par Donat *et al.* (2008 ; 2010), avec quelques modifications. Donat *et al.* (2008) utilisent les GDM afin de modéliser un processus stochastique de dégradation des voies de chemin de fer. La figure 13a représente un GDM sans variable de contexte pour piloter la dynamique $(\delta_t, \mathcal{M}_t)$.

Le processus $(\delta_t, \mathcal{M}_t)$ est Markovien et généralise les processus semi-Markoviens (Barbu *et al.*, 2004) en autorisant des distributions sur les temps de séjours arbitraires et non géométriques (non exponentiellement distribuées).

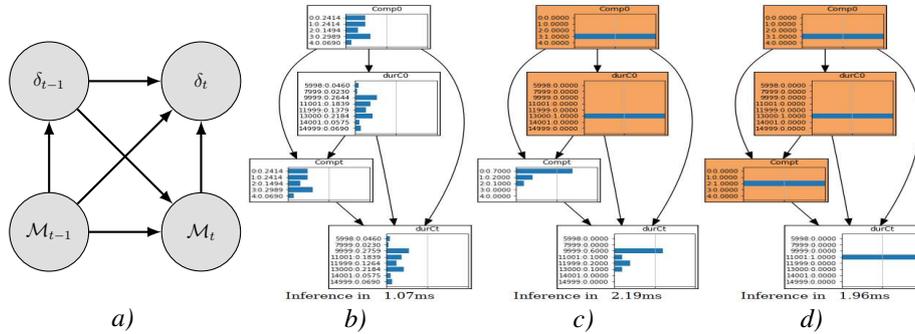


Figure 13. a) Un GDM, b) inférence sans évidences, c) avec évidences à $t - 1$ et d) avec évidences à t

Dans le cadre des ns-DBN la variable d'état du système \mathcal{M}_t correspond à un DBN ; il s'agit du mode courant du processus non stationnaire. Nous quantifions différemment la durée δ_t ; à l'origine, $\delta_t \in \mathbb{N}^+$ décroît avec le temps et une transition arrive lorsque $\delta_t = 1$, ce qui nécessite de connaître à l'avance la durée du mode ou de travailler hors ligne avec un historique et introduit beaucoup de paramètres inutiles. Ici, δ_t prend pour valeur la durée complète d'observation d'un mode ; l'apprentissage n'est effectué que lorsqu'une transition est observée et par conséquent $P(\mathcal{M}_t = \mathcal{M}_{t-1}) = 0$ et il ne peut pas y avoir d'état absorbant.

Quel que soit l'algorithme utilisé, les transitions détectées sont approximatives et continues et il est nécessaire de réaliser un *clustering* afin de regrouper les périodes identiques à quelques événements près. Pour ce faire, nous utilisons l'algorithme k-means - variante de Lloyd (Lloyd, 1982) - avec une fonction de score pour trouver le k idéal : $k^* = \arg \max_k -n - \bar{\sigma}^2$ avec n le nombre de paramètres du GDM et $\bar{\sigma}^2$ la variance moyenne des k clusters découverts.

Ce modèle nous permet donc à tout instant de faire des prédictions soit sur le prochain mode et sa durée (figure 13c) soit sur la durée du mode courant. Pour n'importe quelle fenêtre, nous connaissons \mathcal{M}_{t-1} , δ_{t-1} et \mathcal{M}_t et pouvons ainsi faire une prédiction sur le temps restant du mode \mathcal{M}_t par inférence (figure 13d).

Bien que nous utilisons ce modèle pour exhiber des sous-comportements surprenants (en mode ou en durée), il pourrait y avoir un *feedback* de ce modèle vers la sélection et découverte de DBN précédemment établies dans un cadre hors ligne. Cependant, dans notre cadre en ligne, cela aurait pour effet de *forcer* la sélection - et la durée - d'une mode plutôt qu'un autre à cause de la dynamique *temporellement locale* connue seulement jusqu'à ce temps. Nous n'avons donc pas implémenté ce *feedback*.

5. Expériences et résultats

Nos expériences consistent à modéliser des processus non-stationnaires simulés. En utilisant la librairie aGrUM (<http://agrums.lip6.fr>), nous avons généré un DBN de 10 nœuds par *time slice* de cardinalité moyenne 7 ($\llbracket 3, 10 \rrbracket$) et de degré moyen 3. Nous avons ensuite perturbé la structure et les paramètres du modèle en utilisant la distance de Hellinger (Beran, 1977) entre deux modèles comme critère d'arrêt. Plusieurs seuils ont été choisis pour observer à quel point deux réseaux doivent être distants pour être reconnus comme deux modèles distincts. Des distances de Hellinger supérieures à 0,8 impliquent toujours des changements de paramètres pour tous les nœuds et parfois des changements structuraux pour un petit sous-ensemble de nœuds. Des distances sous ce seuil impliquent des changements de paramètres pour quelques nœuds, de faible degré, et un arc est parfois créé, ajoutant peu d'informations.

Les bases de données sont ensuite échantillonnées pour chaque modèle (par *forward sampling*, c'est-à-dire tirage selon l'ordre topologique) avant d'être combinées pour former une base de données unique de 600 000 événements. Différentes taille d'époques ont été utilisées pour étudier l'impact de la taille des échantillons vis-à-vis de la distance de Hellinger entre réseaux ainsi que différentes résolutions de la fenêtre glissante pour étudier le comportement de l'algorithme lorsque la fenêtre chevauche des données de deux modèles distincts (*i.e.* l'époque n'est pas un multiple de la résolution de la fenêtre). Nous avons lancé chaque jeu de test avec et sans fenêtre adaptative. Il est important de remarquer que notre algorithme n'a aucune information a priori quant au nombre de modèles, leurs variables et les domaines de ces dernières, le nombre de transitions ou leur périodicité.

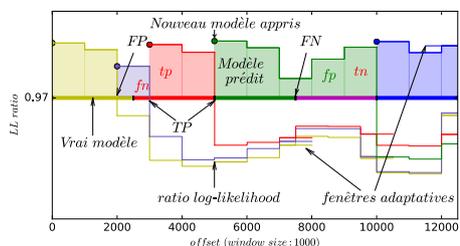


Figure 14. Lecture des figures

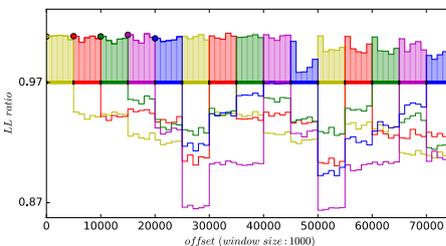


Figure 15. Résultats pour des époques de 5K observations, Hellinger $\approx 0,6$, fenêtre fixe

La figure 14 fictive explique comment lire les courbes et tableaux associés aux expériences, où *FN* signifie un faux négatif pour une transition (transitions manquées), *FP* signifie un faux positif pour une transition (transitions erronées) et *TP* signifie un vrai positif pour une transition (transitions correctes). Aussi, *tp* est le nombre de (vrai) événements appris par les modèles corrects, *fp* est le nombre de (faux) événements appris par de mauvais réseaux et *fn* est le nombre de (vrai) événements ratés par des modèles mais appris par d'autres. Les fenêtres adaptatives peuvent être vues lorsque

les courbes sont étendues sur la gauche (pour le modèle courant qui déplace la fenêtre) ou sur la droite (modèles non sélectionnés qui ne déplacent pas la fenêtre). Dans les tableaux, l'erreur moyenne, minimale et maximale des points de changement est montrée, avec l'écart type, en termes de nombre d'événements. Finalement, la *précision* $tp/(tp+fp)$ et le *rappel* $tp/(tp+fn)$ sur les événements est calculée (voir tableaux), c'est-à-dire la *précision* moyenne et le *rappel* moyen sur les modèles découverts. Le *rappel* est le pourcentage d'événements corrects trouvés pour tous les événements corrects qui auraient dû être trouvés. La *précision* est inversement proportionnelle au bruit (les événements générés depuis un autre modèle qui sont utilisés pour mettre à jour le modèle courant). Par souci d'espace, les résultats sont moyennés pour tous les seuils de distance Hellinger. Nous nous focalisons sur les cas où l'époque n'est pas un multiple de la résolution de la fenêtre et montrons un meilleur cas (figure 15) et pire cas (figure 16, 17, 18, 19) avec et sans fenêtre adaptative. Les résultats pour les fenêtres statiques et dynamiques sont présentés dans les tableaux 1 et 2, respectivement.

5.1. Fenêtres statiques

Tableau 1. Résultats fenêtre fixe. Pour les coupures, erreur minimale, moyenne et maximale en nombre d'événements, avec écart type

époque	taille fenêtre	FN	FP	TP	erreur moy.	écart type	min	max	précision	rappel
2500	1000	1,000	0,000	0,000	NA	NA	NA	NA	0,200	1,000
2500	1000	0,000	0,000	1,000	251,046	249,998	0,000	500,000	0,829	0,875
2500	2000	0,602	0,000	1,000	521,052	361,644	0,000	1000,000	0,500	0,952
5000	1500	0,101	0,035	0,965	351,216	258,546	0,000	1000,000	0,833	0,935
5000	3000	0,000	0,060	0,940	752,101	579,236	0,000	2000,000	0,856	0,854
10000	1500	0,000	0,131	0,869	381,356	295,694	0,000	1000,000	0,961	0,961
10000	3000	0,102	0,008	0,992	772,810	601,843	0,000	2000,000	0,833	0,929
15000	2000	0,000	0,204	0,795	512,820	499,835	0,000	1000,000	0,963	0,958

La figure 15 est un exemple d'une exécution réussie : l'époque est un multiple de la résolution de la fenêtre et par conséquent, la fenêtre glissante contient, à tout instant, uniquement des observations d'un modèle à la fois. Dans ces cas, les modèles et moments de transition sont correctement identifiés, avec ou sans fenêtre adaptative.

Cependant, des erreurs surviennent lorsque l'on utilise des résolutions arbitraires pour la fenêtre statique, comme le montrent les figures 16 et 17.

Dans le cas statique, comme montré en tableau 1, deux phénomènes expliquent les faibles taux de *précision* et *rappel* obtenus pour certaines expériences. Le premier phénomène survient lorsque nous avons découvert moins de modèles que nous aurions dû, principalement avec des seuils sur la distance de Hellinger bas, auquel cas des transitions ont été ratées et certains modèles moyennent plusieurs modèles sous-jacents réels, augmentant leur bruit et rendant les transitions futures plus dures à détecter. Par conséquent, les *FN* de transitions augmentent alors que la *précision* et le *rappel* sur les événements diminuent. Un tel cas est exposé par la figure 16 et les

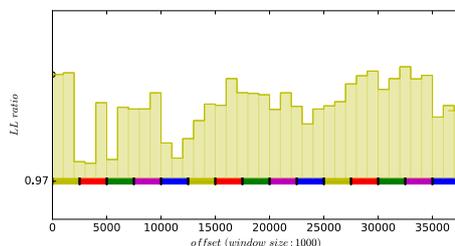


Figure 16. Résultats pour des époques de 2K5 observations, Hellinger $\approx 0,6$, fenêtre fixe

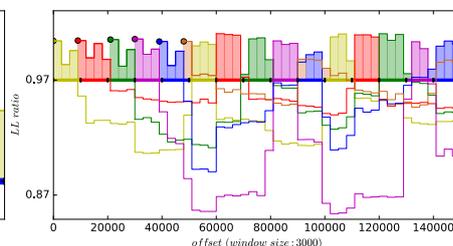


Figure 17. Résultats pour des époques de 10K observations, Hellinger $\approx 0,6$, fenêtre fixe

deux premières lignes du tableau 1, où la première ligne et la figure 16 montrent les résultats pour des modèles d'origine proches et la seconde ligne les résultats pour des modèles d'origine distants. Le second phénomène survient lorsque l'on a découvert davantage de modèles que nous aurions dû, principalement avec de grandes distances de Hellinger. Dans ce cas, la plupart des modèles en excès ont été créés lorsque la fenêtre chevauche des observations de deux modèles d'origine distincts, modélisant ainsi la transition elle-même (la prochaine fenêtre crée ou reconnaît le bon modèle), comme en figure 17 (le modèle représenté en marron). Ainsi, nous avons deux transitions au lieu d'une, augmentant FP pour les transitions. La *précision* et le *rappel* sont moins affectés par ces FP puisque seules quelques transitions provoquent la création de modèles spécifiques, réduisant faiblement le *rappel* d'autres réseaux découverts (à juste titre) mais augmentant sensiblement leur *précision* (réduction du bruit).

Les résultats pour les fenêtres statiques pourraient être pires : dans nos expériences, une époque n'est pas un multiple de la résolution de la fenêtre mais un multiple de l'époque peut être un multiple de la résolution de la fenêtre, dans quel cas la fenêtre commence ou termine au niveau d'une vraie transition, ce qui a pour effet "d'augmenter" potentiellement la probabilité de correctement identifier une transition et/ou un modèle. Ainsi, FN et l'erreur sur les points de changement pourraient être plus grande alors que la *précision* et le *rappel* pourraient être plus faibles.

5.2. Fenêtres adaptatives

Les résultats pour les fenêtres adaptatives, montrés en tableau 2, révèlent que la taille de la fenêtre a peu d'impact sur l'identification correcte des transitions et des modèles, et ceci devrait être vrai tant que la résolution de la fenêtre est plus petite que celle d'une époque. On peut supposer que la taille raisonnable des modèles d'entrées explique le fait que les résultats ne soient pas plus mauvais pour de petites époques. Avec les fenêtres adaptatives, les deux phénomènes précédents sont résolus en cherchant le point de changement optimal, comme en figures 18 et 19 : dans le premier cas, nous n'apprenons pas depuis des fenêtres qui chevauchent deux modèles ce qui

Tableau 2. Résultats fenêtre adaptative, avec les colonnes comme en tableau 1

époque	taille fenêtre	FN	FP	TP	erreur moy,	écart type	min	max	précision	rappel
2500	1000	0,000	0,000	1,000	4,399	18,045	0,000	254,000	0,998	0,998
2500	2000	0,000	0,000	1,000	2,435	4,683	0,000	38,501	0,999	0,999
5000	1500	0,000	0,000	1,000	3,097	7,784	0,000	62,504	0,999	0,999
5000	3000	0,000	0,000	1,000	8,702	43,383	0,000	390,511	0,998	0,998
10000	1500	0,000	0,000	1,000	32,923	80,526	0,000	311,522	0,997	0,996
10000	3000	0,000	0,000	1,000	19,559	103,199	0,000	778,000	0,998	0,998
15000	2000	0,000	0,000	1,000	8,551	32,423	0,000	202,534	0,999	0,999

réduit le bruit et évite d'accroître la difficulté d'identification des transitions futures. Dans le second cas, chercher un point de changement (Eq. 11) évite en soi-même la création d'un modèle représentant la transition seule.

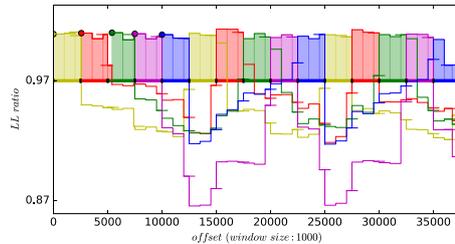


Figure 18. Résultats pour des époques de 2K5 observations, Hellinger $\approx 0,6$, fenêtre adaptative

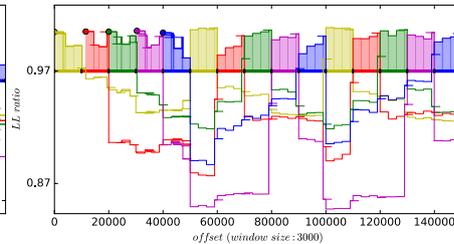


Figure 19. Résultats pour des époques de 10K observations, Hellinger $\approx 0,6$, fenêtre adaptative

La capacité de l'algorithme à ajouter des modalités à des variables connues évite la création de modèles redondants dans les deux cas, ce qui réduit *FP* pour les transitions et est d'une importance cruciale pour les *outliers* qui surviennent de temps à autre et doivent être incorporés aux modèles déjà existants.

5.3. Fenêtres adaptatives depuis équation (12)

Dans cette section nous présentons les résultats pour le même jeu d'expériences que précédemment mais en utilisant directement l'équation (12) sans passer par (10). Seuls les résultats pour les fenêtres adaptatives sont donc présentés étant donné que l'équation (12) est l'adaptation de la fenêtre.

Les résultats montrés en tableau 3 sont aussi bons ou meilleurs que les résultats en tableau 2, notamment sur la précision des points de changement. En effet, lorsque l'on utilise l'équation (10), il se peut qu'une partie de la fenêtre (en fin de fenêtre) n'ait pas une vraisemblance suffisamment basse pour faire échouer le test statistique.

Tableau 3. Résultats fenêtre adaptative, toutes distances de hellinger confondues

époque	taille fenêtre	FN	FP	TP	erreur moy,	écart type	min	max	précision	rappel
2500	1000	0,000	0,000	1,000	2,580	9,690	0,000	103,500	0,999	0,999
2500	2000	0,000	0,000	1,000	1,651	2,779	0,000	25,000	0,999	0,999
5000	1500	0,000	0,000	1,000	1,968	3,469	0,000	23,500	0,999	0,999
5000	3000	0,000	0,000	1,000	75,231	155,166	0,000	406,501	0,987	0,985
10000	1500	0,000	0,000	1,000	6,059	19,158	0,000	102,000	0,999	0,999
10000	3000	0,000	0,000	1,000	2,100	3,211	0,000	13,496	0,999	0,999
15000	2000	0,000	0,000	1,000	24,381	43,969	0,000	163,000	0,998	0,998

La fenêtre est alors apprise, introduisant du bruit et augmentant le risque de ne pas identifier la transition courante ou de l'identifier plus tardivement (ou d'avoir un point de changement plus lointain qu'il ne faut, à cause du changement de paramètre et/ou structure induit par la fenêtre précédente).

Utiliser directement l'équation (12) sans passer par (10) permet aussi de s'affranchir du sampling et offre donc un gain de temps significatif (pour chaque création de nouveau modèle et ré-estimation de la structure).

Tableau 4. Résultats fenêtre adaptative, Hellinger $\approx 0,6$

époque	taille fenêtre	FN	FP	TP	erreur moy,	écart type	min	max	précision	rappel
2500	1000	0,000	0,000	1,000	4,834	18,511	0,000	200,000	0,998	0,998
2500	2000	0,000	0,000	1,000	2,975	4,73	0,000	3,000	0,999	0,999
5000	1500	0,000	0,000	1,000	3,634	6,179	0,000	43,000	0,999	0,999
5000	3000	0,000	0,000	1,000	150,202	309,607	0,000	809,000	0,974	0,970
10000	1500	0,000	0,000	1,000	11,644	37,318	0,000	200,000	0,999	0,999
10000	3000	0,000	0,000	1,000	3,777	5,465	0,000	23,000	0,999	0,999
15000	2000	0,000	0,000	1,000	48,513	87,309	0,000	323,000	0,997	0,997

Tableau 5. Résultats fenêtre adaptative, Hellinger $\geq 0,8$

époque	taille fenêtre	FN	FP	TP	erreur moy,	écart type	min	max	précision	rappel
2500	1000	0,000	0,000	1,000	0,339	0,867	0,000	7,000	0,999	0,999
2500	2000	0,000	0,000	1,000	0,370	0,830	0,000	7,000	0,999	0,999
5000	1500	0,000	0,000	1,000	0,300	0,759	0,000	4,000	0,999	0,999
5000	3000	0,000	0,000	1,000	0,269	0,721	0,000	4,000	0,999	0,999
10000	1500	0,000	0,000	1,000	0,472	0,997	0,000	4,000	0,999	0,999
10000	3000	0,000	0,000	1,000	0,422	0,941	0,000	4,000	0,999	0,999
15000	2000	0,000	0,000	1,000	0,256	0,629	0,000	3,000	0,999	0,999

Les tableaux 4, 5 présentent les résultats du tableau 3 selon les distances de Hellinger qui séparent les modèles d'origine. On remarque que les résultats du tableau 3

sont principalement guidés par ceux du tableau 4, c'est-à-dire par les expériences avec des modèles plus facilement confondables. Lorsque les modèles sont distants, la plus grande imprécision sur le temps d'un point de changement est 7. Ces résultats sont attendus puisqu'il existe un seuil en dessous duquel il n'est plus possible de différencier deux modèles. Plus les modèles sont confondables, plus l'imprécision des points de changement augmente. Nous verrons dans la prochaine section que la modification d'un paramètre lors de la recherche de point de changement permet de contrôler la finesse de la détection de ces points. On peut donc trouver la valeur du paramètre qui nous permet de descendre jusqu'à la distance de Hellinger souhaitée.

Il est intéressant de remarquer que l'absence de faux positifs pour les transitions (FP) implique que si un seul modèle (stationnaire) avait généré la base de donnée nous aurions retrouvé ce modèle. En revanche la présence de faux positifs implique qu'un modèle est découpé en sous-modèles plus spécifiques et/ou moins complexes, ce qui peut être bénéfique pour certaines applications tant que le taux de faux négatifs n'augmente pas. Par exemple, la chaîne de Markov qui modélise un pile ou face fortement biaisé peut être représentée de manière équivalente par un ns-DBN avec deux modes, chacun ayant une variable aléatoire dégénérée (qui ne prend qu'une valeur) si le biais est très fort (une ou plusieurs fenêtres contiennent majoritairement une seule valeur). La prochaine section revient sur cette notion.

5.4. Cas extrêmes

Dans cette section nous montrons le comportement de l'algorithme lorsque les époques sont très (trop) petites pour permettre d'apprendre de manière efficace différents modes. Les expériences sont réalisées avec des réseaux ayant une distance de Hellinger supérieure ou égale à 0,8. Les figures 20 et 21 montrent l'évolution du rappel, de la précision et de la sur-segmentation (découverte de davantage de modes) en fonction de la taille des époques, pour un α de 0,02 et 0,05 respectivement. La taille de la fenêtre est fixée à une époque et demie et $c_{min} = 10$ pour toutes les tailles d'époque.

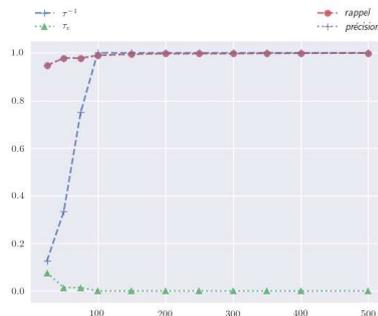


Figure 20. Rappel, précision et τ^{-1} , τ_e pour $\alpha = 0,02$

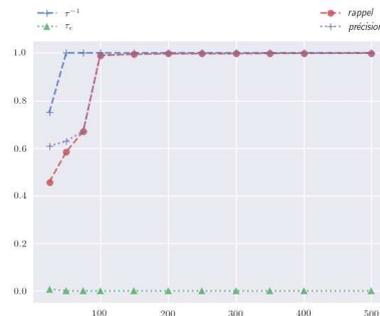


Figure 21. Rappel, précision et τ^{-1} , τ_e pour $\alpha = 0,05$

On constate en figure 20 un maintien du rappel et de la précision au dessus de 90% lorsque les époques sont plus petites (jusque 25 points de données). En notant $\tau = \frac{\#\text{modèles découverts}}{\#\text{modèles réels}}$ le taux de sur-segmentation des modèles, on constate qu'une sur-segmentation apparaît dès que les époques sont de taille 75 ou moins, avec $\tau > 1$, comme l'indique la courbe marquée par des + (on visualise τ^{-1} pour des raisons d'échelle). Il s'agit de faux positifs sur les transitions mais les modèles précédant et courant peuvent appartenir au même mode sous-jacent et non à deux modes sous-jacent distincts. Ces modèles superflus absorbent une faible part des événements, notée τ_e , car ils n'apparaissent que très rarement comme montré par la courbe marquée par des \blacktriangle ; moins de 10% des événements appartiennent à ces modèles pour des époques de taille 25. Ils sont peu bruités comme le montre le maintien du rappel et de la précision. Le taux de sur-segmentation τ des modèles est tout de même de presque 10. L'augmentation de α permet de combattre cette sur-segmentation pour les cas les plus extrêmes pour un coût en rappel et précision, comme montré en figure 21. Le taux de faux négatif sur les transitions augmente. La sur-segmentation sur les modèles comme sur les événements est beaucoup plus faible puisque moins de points de changements sont découverts. Par conséquent, les frontières entre modes sont plus floues et les modèles sont difficilement qualifiables, ce qu'indique la perte de rappel et de précision.

Ainsi, lorsque les époques sont trop petites et/ou la fréquence d'échantillonnage est trop faible, il est nécessaire de réaliser un compromis entre la sur-segmentation et les performances de classification de l'algorithme. La prochaine section montre, entre autre, qu'un α trop grand entraîne une perte de précision et de rappel (et donc de qualification des modèles et modes) systématique, et qu'il vaut mieux choisir un α plus petit si les époques le permettent, c'est-à-dire qu'elles ne provoquent pas de sur-segmentation. Il est important de noter que ces résultats dépendent de la complexité structurelle et paramétrique des modèles de chaque mode sous-jacent.

5.5. Influence des paramètres

Dans cette section nous étudions l'impact des paramètres de l'algorithme sur la qualité des modèles découverts. Les trois paramètres d'intérêt sont :

- k , équation (11), le nombre de fenêtre passées gardées en mémoire pour l'algorithme de point de changement.
- α , équation (11), le facteur de vraisemblance qui sert de pénalité dans $f(c)$ et contrôle quand un point est un potentiel point de changement.
- c_{min} , équation (12), qui correspond à la durée minimale d'un mode et donc au nombre de points minimum (de la nouvelle fenêtre) pour le test statistique de présence de point de changement. Il permet de ne pas choisir de modèles candidats fallacieux.

Les résultats sont séparés selon la distance de hellinger inter-modèles (ceux utilisés pour générer les base de données). Pour chaque point nous avons moyenné les résultats de 7 expériences (celles présentées dans les tableaux précédents) donc avec des époques et fenêtres de tailles différentes les unes des autres.

Un seul paramètre varie à la fois, les autres sont maintenus fixes. k , α et c_{min} ont comme valeur par défaut (valeur utilisée pour toutes les expériences précédentes), 4, 0,005 et 100, respectivement.

5.5.1. Étude de k

Nous faisons varier k , le nombre de fenêtres passées retenues pour détecter un prochain point de changement, avec $c_{min} = 100$ et $\alpha = 0,005$. Le point le plus à droite ($k = 40$) sur les figures 22, 23, 24, 25, 26 et 27 correspond à une asymptote (i.e. toutes les fenêtres précédentes sont retenues) et à été fixé à 40 pour ne pas écraser les courbes sur la gauche et parce-qu'il y a au moins 40 fenêtres de données pour chacun des modèles d'origine (120 000 évènements par modèle et des fenêtres de taille maximale 3 000).

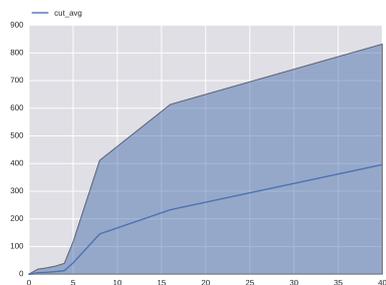


Figure 22. Erreur moyenne en fonction de k , $Hellinger \approx 0,6$

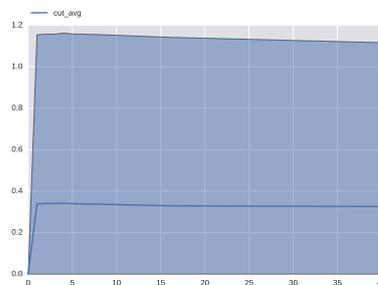


Figure 23. Erreur moyenne en fonction de k , $Hellinger \geq 0,8$

Le point $(0, 0)$ a été fixé par convention puisqu'il est nécessaire d'avoir des données précédant les nouvelles observations pour trouver un point de changement (vis-à-vis du modèle) dans ces dernières. Avec $k = 0$, aucun point de changement n'est trouvé et nous n'avons par conséquent pas de valeur pour l'erreur sur les points de changement.

Intuitivement, plus le nombre de fenêtres retenues augmente plus l'imprécision sur les points de changements augmente (figure 22). En effet, plus l'apprentissage d'un modèle se renforce avec le temps pour converger plus la vraisemblance de données issues de son modèle d'origine augmente. Ainsi, retenir des fenêtres de données trop lointaines augmente l'écart type de la loi à gauche de tout point de changement potentiel ce qui a pour effet de rendre les détections de changements doux difficiles. Les changements plus brusques, figure 23, sont plus facilement détectables.

L'augmentation de k introduit donc d'abord une augmentation de l'imprécision sur les points de changements (figure 22) ce qui se traduit par une augmentation du bruit dans les réseaux appris et donc une diminution du rappel sur les évènements (figure 24). Comme l'augmentation du bruit pour un réseau donné signifie une diminution de la précision d'un autre réseau (son taux de faux négatifs sur les évènements augmente puisque ces derniers ont été absorbés par un autre modèle), la précision moyenne

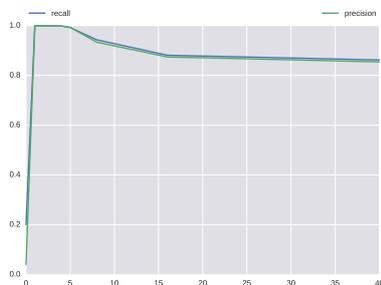


Figure 24. Rappel et précision en fonction de k , $Hellinger \approx 0,6$

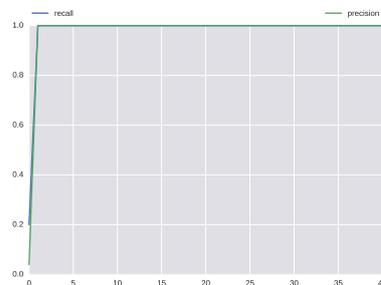


Figure 25. Rappel et précision en fonction de k , $Hellinger \geq 0,8$

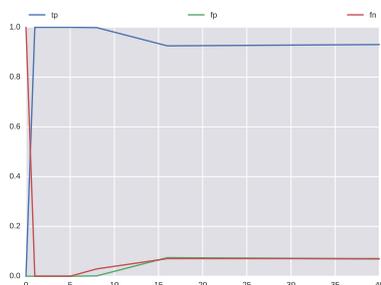


Figure 26. TP, FP, FN en fonction de k , $Hellinger \approx 0,6$

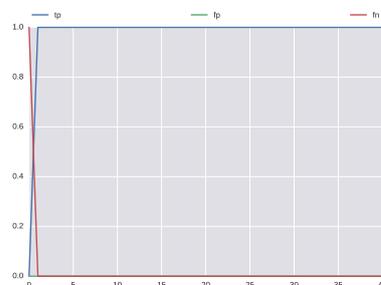


Figure 27. TP, FP, FN en fonction de k , $Hellinger \geq 0,8$

diminue de concert. Le rappel et la précision restent cependant corrects, malgré l'augmentation avec $k \geq 6$ du taux de faux négatifs et faux positifs pour les transitions (figure 26). Le rappel et la précision chutent alors davantage puisque toute transition ratée ou erronée introduit du bruit pour le modèle courant et une perte de précision pour le modèle qui aurait dû être sélectionné.

Le cas avec des modèles originaux très distants est insensible à l'augmentation de k (figures 23, 25, 27). L'erreur sur les points de changement a une moyenne inférieure à 1 événement et l'écart type est très faible (figure 23).

Globalement k n'est pas le paramètre qui a le plus d'influence sur notre algorithme et les performances se maintiennent relativement bien ; le rappel et la précision restent élevés, les capacités de prédiction des modèles appris restent correctes. Les meilleurs résultats sont atteints avec $k \in \{1, 2, 3\}$.

5.5.2. Étude de α

Nous faisons varier α , le facteur de vraisemblance qui sert de pénalité dans $f(c)$ et contrôle quand un point est un potentiel point de changement, en maintenant $c_{min} = 100$ et $k = 4$.

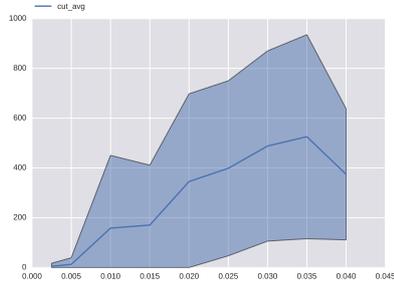


Figure 28. Erreur moyenne en fonction de α , Hellinger $\approx 0,6$

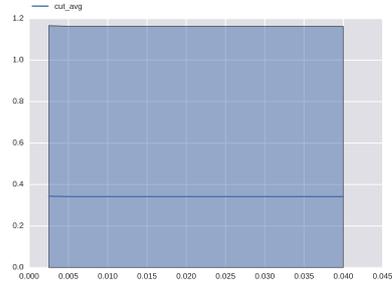


Figure 29. Erreur moyenne en fonction de α , Hellinger $\geq 0,8$

Plus intuitivement que précédemment, puisque α contrôle directement de combien un point de changement doit améliorer la vraisemblance de la fenêtre pour un modèle considéré. L'augmentation de α diminue le facteur d'amélioration attendu (puisque la log-vraisemblance est négative). α contrôle donc indirectement la distance minimale détectable et par conséquent entre modèles appris. Un α trop grand pour un processus qui évolue doucement d'un mode à un autre entraîne une augmentation de l'erreur sur les points de coupures (figure 28).

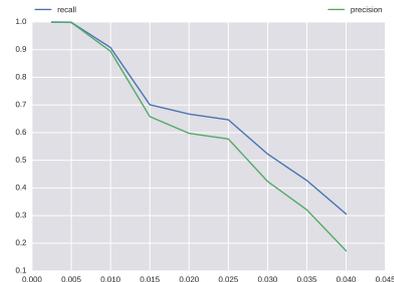


Figure 30. Rappel et précision en fonction de α , Hellinger $\approx 0,6$

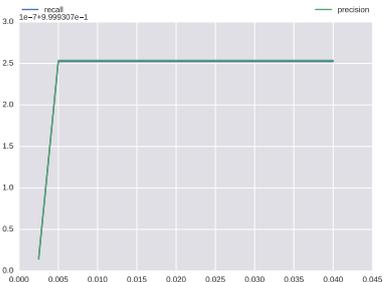


Figure 31. Rappel et précision en fonction de α , Hellinger $\geq 0,8$

Entre $\alpha = 0,02$ et $\alpha = 0,025$, pour des distances de Hellinger d'environ 0,6, certains jeux d'expériences commencent à échouer complètement (aucune transition n'est détectée). Sur la figure 32 cela se traduit par $TP + FP < 1$ (l'ensemble des transitions découvertes doit sommer à 1 pour chaque expérience autrement nous avons manqué toutes les transitions ($FN = 1$) pour cette expérience). À partir de cette valeur de α , la plupart des jeux d'expériences perdent la capacité de détecter des points de coupures petit-à-petit pour certains modèles ; nous avons une augmentation du taux de faux négatifs (figure 32) pour des modèles entiers qui sont confondus avec un autre. D'autres modèles au sein de la même expérience sont correctement détectés mais présentent beaucoup de bruit à cause de l'erreur élevée des points de changements. Toutes expériences confondues le rappel et la précision diminuent donc drastiquement avec un

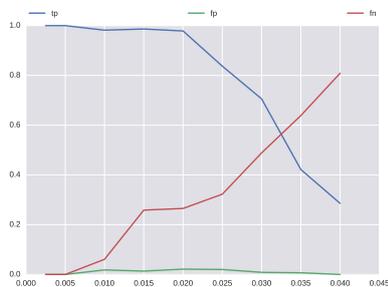


Figure 32. TP, FP, FN en fonction de α , $Hellinger \approx 0,6$

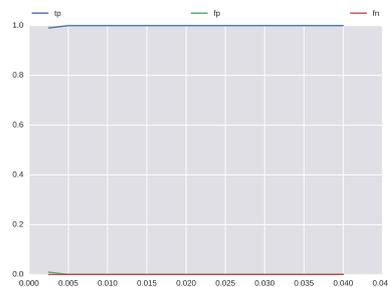


Figure 33. TP, FP, FN en fonction de α , $Hellinger \geq 0,8$

α destiné à détecter des changements trop brusques (figure 30) pour un processus qui évolue doucement d'un mode à un autre.

Il est intéressant de noter que α ne fait pas monter le taux de faux positifs pour les transitions mais augmente les faux négatifs (figure 32). Dans certaines applications (notamment pour les IDS), il est préférable d'avoir moins d'alertes, mais pertinentes, que beaucoup d'alertes dont la plupart sont de fausses alarmes.

Encore une fois, les processus aux changements brusques sont peu sensibles à la valeur de α (figures 29, 31, 33).

α est le paramètre qui influence le plus notre algorithme en imposant une limite sur la distance qui sépare les modèles découverts. Pour des modèles très proches, une valeur de α légèrement supérieure au "paramètre idéal" (qui permettrait de détecter exactement les modèles d'origine) pourrait éviter la création de modèles redondants car trop proches (pour des distances de Hellinger plus petites que 0,1 par exemple).

Pour détecter des modèles séparés par une distance de Hellinger plus grande que 0,6, $\alpha = 0,005$ semble idéal.

5.5.3. Étude de c_{min}

Finalement, nous faisons varier c_{min} , la durée minimale d'un mode (ou le nombre minimal de points autorisé d'une nouvelle fenêtre après un point de coupure pour un modèle $m_t^* \neq m_{t-1}^*$).

c_{min} fait office de taille d'échantillon pour le test en équation 11, 12. Intuitivement, que dire d'une transition d'un modèle vers un autre pour une durée de 2 points avant de transiter à nouveau vers un autre modèle ? Si la vraisemblance moyenne d'un modèle m_i est meilleure que la vraisemblance moyenne d'un modèle m_j avec $c_i^* \lll c_j^*$, on peut se poser la question du meilleur modèle à choisir. Il est en effet possible d'avoir plusieurs *matches* sur une même fenêtre adaptée différemment par chaque modèle candidat.

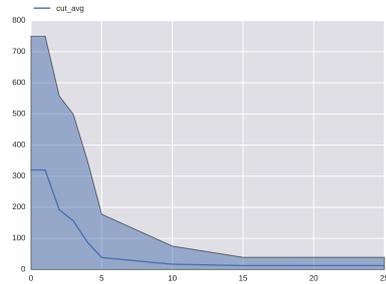


Figure 34. Erreur moyenne en fonction de c_{min} , Hellinger $\approx 0,6$

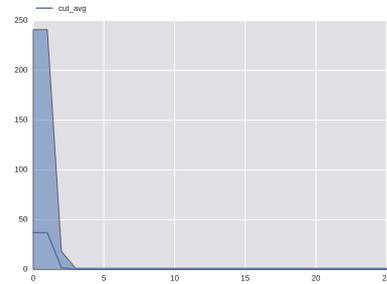


Figure 35. Erreur moyenne en fonction de c_{min} , Hellinger $\geq 0,8$

En règle générale, plus la taille d'échantillon est importante plus les tests statistiques sont fiables. L'augmentation de c_{min} diminue donc l'erreur sur les points de changement que ce soit pour les expériences avec des modèles relativement proches (figure 34) ou distants (figure 35).

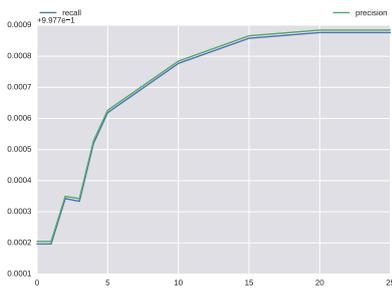


Figure 36. Rappel et précision en fonction de c_{min} , Hellinger $\approx 0,6$

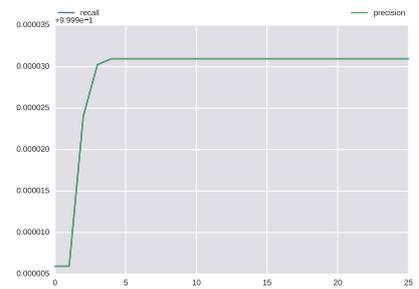


Figure 37. Rappel et précision en fonction de c_{min} , Hellinger $\geq 0,8$

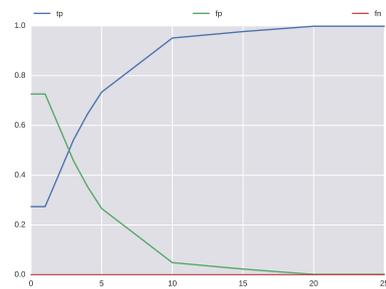


Figure 38. TP, FP, FN en fonction de c_{min} , Hellinger $\approx 0,6$

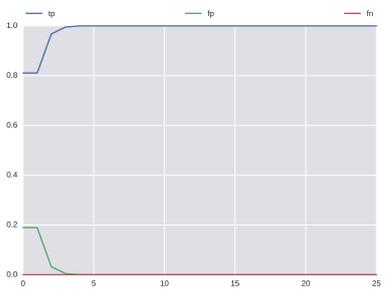


Figure 39. TP, FP, FN en fonction de c_{min} , Hellinger $\geq 0,8$

L'augmentation de c_{min} permet d'éviter les transitions archaïques soit le taux de faux positifs (figures 38, 39). Conjointement avec une meilleure précision des points de changements, le rappel et la précision augmentent (figures 36, 37).

Pour les distances utilisées une valeur de $c_{min} \geq 20$ semble suffisante. Cependant, il est intéressant de remarquer que plus les modèles sont proches, plus c_{min} a besoin d'être augmenté avant d'atteindre son asymptote en terme de rappel et précision (figures 36, 37). Avec des modèles séparés par une distance de Hellinger plus petite que 0,6, il faudrait sans doute fixer c_{min} aux alentours de 100. En effet, pour distinguer deux distributions proches, il faut beaucoup plus de données.

5.6. Première application sur un cas réel

L'application visée est la modélisation et la prédiction d'incongruités pour un Système de Détection d'Intrusion (IDS) en temps réel. Nous présentons les premiers résultats obtenus sur des logs apache (une ligne est présentée ci-dessous).

```
173.36.240.171 - - [01/Jun/2015:00:01:19 +0000] "GET
/projects/scapy/ HTTP/1.1" 200 6205
"http://r.search.yahoo.com/_ylt=
AwrTccnLoGtVzh8A_NUnnI1Q;_ylu=
X3oDMTByYnR1Zmd1BGNvbG8DZ3ExBHBvcwMyBHZ0aWQDBHN1YwNzcg
--/RV=2/RE=1433145676/RO=10/RU=http%3a%2f%2fsecdev.org
%2fprojects%2fscapy%2f/RK=0/RS=DV3KSkU3cpgfOsdls5DOVKn
1zrs-" "Mozilla/5.0 (Windows NT 6.1; WOW64; rv:38.0)
Gecko/20100101 Firefox/38.0"
```

Le serveur a enregistré un mois de logs, présentant 29 341 adresses ip et quelques anomalies / patterns d'attaque (5) on été identifiés au préalable par des experts (manuellement et par signature).

Sur ce mois, 33 modèles ont été découverts. Pour chaque ip, nous avons un ensemble de modèles ; certains sont partagés par plusieurs adresses ip et d'autres sont uniques. Un modèle apparaît seulement pendant les attaques et inclut toutes les occurrences d'attaques connues détectables.

Les figures 40a, 40b et 40c montrent trois DBN identifiés qui exhibent des changements de relations entre variables d'un mode à l'autre. Nous avons donc identifié un modèle qui a capturé les dynamiques anormales qui surviennent lors des attaques. Cependant, toutes les attaques n'ont pas été détectées car toutes ne sont pas détectables telles quelles. En effet, plusieurs adresses ip n'apparaissent qu'une seule fois et ne réalisent qu'une action ; nous n'avons donc pas de trajectoire pour ces dernières et il y a peu de choses à faire de ces données trop parcellaires.

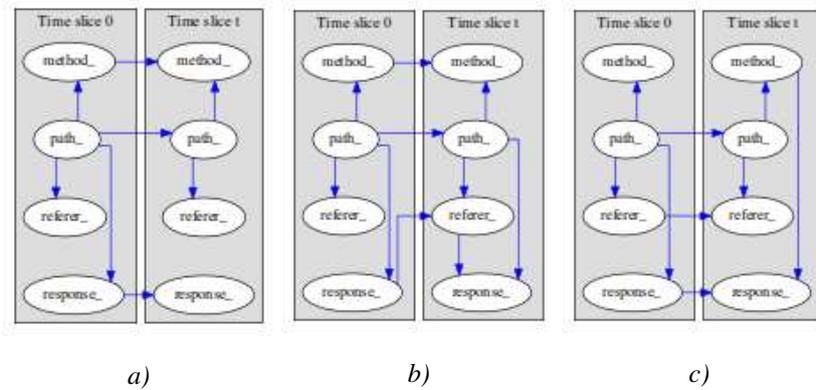


Figure 40. Trois DBN exhibant des relations différentes entre variables et/ou des paramètres différents

6. Conclusions et travaux futurs

Nous avons construit un cadre général utilisant les réseaux bayésiens dynamiques pour apprendre des processus non stationnaires de manière continue, dirigé comme un compromis entre rapidité et précision. Les expériences montrent de très bons résultats sur les premiers jeux de données simulées et réelles. Nous avons proposé un modèle pour la dynamique des transitions entre modes qui nous permet de déterminer le temps restant d'un mode ou de prédire le prochain mode et sa durée. Il serait intéressant (et nécessaire) d'expérimenter davantage avec α pour descendre à de plus basses distances entre comportements. Plusieurs améliorations viennent à l'esprit : nous avons mentionné un seuil dynamique sur les ratios de log-vraisemblance pour prendre en compte la convergence d'un modèle, ainsi que le besoin d'une procédure de fusion et suppression entre modèles, puisque nous nous attendons à ce que les résultats soient de plus en plus mauvais au fur et à mesure que la distance entre réseaux d'origine diminue. Bien qu'une procédure de suppression naïve pourrait consister à utiliser un paramètre pour chaque modèle, décroissant avec le temps lorsque le modèle n'est pas rencontré, la fusion de modèles nécessite de comparer leurs probabilités jointes, ce qui implique des calculs lourds. Le problème du point de changement optimal peut aussi être poussé plus loin. Finalement, nous appliquerons cet algorithme dans le cadre de la détection d'anomalies dans un système de détection d'intrusion (hôte et réseau) suite aux premiers résultats encourageants obtenus sur des logs apache.

Remerciements

Ce travail est supporté par Akheros S.A.S./bourse ANRT CIFRE #2014/0268 et le projet européen SCISSOR H2020-ICT-2014-1 #644425.

Bibliographie

- Akaike H. (1998). Information theory and an extension of the maximum likelihood principle. In *Selected papers of hirotugu akaike*, p. 199–213. Springer.
- An X., Jutla D., Cercone N. (2006). Privacy intrusion detection using dynamic bayesian networks. In *Acm international conference proceeding series*, vol. 156, p. 208–215.
- Barbu V., Boussemart M., Limnios N. (2004). Discrete-time semi-markov model for reliability and survival analysis. *Communications in Statistics-Theory and Methods*, vol. 33, n° 11, p. 2833–2868.
- Baum L. E., Petrie T. (1966). Statistical inference for probabilistic functions of finite state markov chains. *The annals of mathematical statistics*, vol. 37, n° 6, p. 1554–1563.
- Beran R. (1977). Minimum hellinger distance estimates for parametric models. *The Annals of Statistics*, p. 445–463.
- Casella G., George E. I. (1992). Explaining the gibbs sampler. *The American Statistician*, vol. 46, n° 3, p. 167–174.
- Charitos T., Van Der Gaag L. C., Visscher S., Schurink K. A., Lucas P. J. (2009). A dynamic bayesian network for diagnosing ventilator-associated pneumonia in icu patients. *Expert Systems with Applications*, vol. 36, n° 2, p. 1249–1258.
- Chickering D., Geiger D., Heckerman D. (1995). Learning bayesian networks: Search methods and experimental results. In *proceedings of fifth conference on artificial intelligence and statistics*, p. 112–128.
- Dean T., Kanazawa K. (1989). A model for reasoning about persistence and causation. *Computational intelligence*, vol. 5, n° 2, p. 142–150.
- Debar H., Dacier M., Wespi A. (1999). Towards a taxonomy of intrusion-detection systems. *Computer Networks*, vol. 31, n° 8, p. 805–822.
- Donat R., Bouillaut L., Aknin P., Leray P. (2008). Reliability analysis using graphical duration models. In *Availability, reliability and security, 2008. ares 08. third international conference on*, p. 795–800.
- Donat R., Leray P., Bouillaut L., Aknin P. (2010). A dynamic bayesian network to represent discrete duration models. *Neurocomputing*, vol. 73, n° 4, p. 570–577.
- Duan J., Zeng J., Zhang D. (2009). A method for determination on hmm distance threshold. In *Fuzzy systems and knowledge discovery, 2009. fskd'09. sixth international conference on*, vol. 1, p. 387–391.
- Getoor L., Friedman N., Koller D., Taskar B. (2001). Learning probabilistic models of relational structure. In *Icml*, vol. 1, p. 170–177.
- Gonzales C., Dubuisson S., Manfredotti C. (2015). A new algorithm for learning non-stationary dynamic bayesian networks with application to event detection. In *The twenty-eighth international affairs conference*.
- Grzegorzcyk M., Husmeier D. (2009). Non-stationary continuous dynamic bayesian networks. In *Advances in neural information processing systems*, p. 682–690.
- Grzegorzcyk M., Husmeier D. (2011). Non-homogeneous dynamic bayesian networks for continuous data. *Machine Learning*, vol. 83, n° 3, p. 355–419.

- Grzegorzczuk M., Husmeier D., Edwards K. D., Ghazal P., Millar A. J. (2008). Modelling non-stationary gene regulatory processes with a non-homogeneous bayesian network and the allocation sampler. *Bioinformatics*, vol. 24, n° 18, p. 2071–2078.
- Heckerman D., Geiger D., Chickering D. M. (1995). Learning bayesian networks: The combination of knowledge and statistical data. *Machine learning*, vol. 20, n° 3, p. 197–243.
- Johansson M., Olofsson T. (2007). Bayesian model selection for markov, hidden markov, and multinomial models. *IEEE signal processing letters*, vol. 14, n° 2, p. 129–132.
- Kruegel C., Mutz D., Robertson W., Valeur F. (2003). Bayesian event classification for intrusion detection. In *Computer security applications conference, 2003. proceedings. 19th annual*, p. 14–23.
- Kullback S., Leibler R. A. (1951). On information and sufficiency. *The annals of mathematical statistics*, vol. 22, n° 1, p. 79–86.
- Lerner U., Parr R., Koller D., Biswas G. *et al.* (2000). Bayesian fault detection and diagnosis in dynamic systems. In *Aaai/iaai*, p. 531–537.
- Liu T., Lemeire J. (2017). Efficient and effective learning of hmms based on identification of hidden states. *Mathematical Problems in Engineering*, vol. 2017.
- Lloyd S. (1982). Least squares quantization in pcm. *IEEE transactions on information theory*, vol. 28, n° 2, p. 129–137.
- Mitra V., Nam H., Espy-Wilson C. Y., Saltzman E., Goldstein L. (2011). Gesture-based dynamic bayesian network for noise robust speech recognition. In *Acoustics, speech and signal processing (icassp), 2011 ieee international conference on*, p. 5172–5175.
- Murphy K. P. (2002a). *Dynamic bayesian networks: representation, inference and learning*. Thèse de doctorat non publiée, University of California, Berkeley.
- Murphy K. P. (2002b). Hidden semi-markov models (hsmms). *unpublished notes*, vol. 2.
- Mutz D., Valeur F., Vigna G., Kruegel C. (2006). Anomalous system call detection. *ACM Transactions on Information and System Security (TISSEC)*, vol. 9, n° 1, p. 61–93.
- Ourston D., Matzner S., Stump W., Hopkins B. (2003). Applications of hidden markov models to detecting multi-stage network attacks. In *System sciences, 2003. proceedings of the 36th annual hawaii international conference on*, p. 10–pp.
- Pearl J. (1988). *Probabilistic reasoning in intelligent systems: Networks of plausible inference*. San Francisco, CA, USA, Morgan Kaufmann Publishers Inc.
- Perron P. (1989). The great crash, the oil price shock, and the unit root hypothesis. *Econometrica: Journal of the Econometric Society*, p. 1361–1401.
- Rabiner L. R. (1989). A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, vol. 77, n° 2, p. 257–286.
- Rissanen J. (1978). Modeling by shortest data description. *Automatica*, vol. 14, n° 5, p. 465–471.
- Robinson J. W., Hartemink A. J. (2009). Non-stationary dynamic bayesian networks. In *Advances in neural information processing systems*, p. 1369–1376.
- Robinson J. W., Hartemink A. J. (2010). Learning non-stationary dynamic bayesian networks. *The Journal of Machine Learning Research*, vol. 11, p. 3647–3680.

- Schwarz G. (1978, 03). Estimating the dimension of a model. *Ann. Statist.*, vol. 6, n° 2, p. 461–464. Consulté sur <http://dx.doi.org/10.1214/aos/1176344136>
- Sicard M., Baudrit C., Leclerc-Perlat M., Wuillemin P.-H., Perrot N. (2011). Expert knowledge integration to model complex food processes. application on the camembert cheese ripening process. *Expert Systems with Applications*, vol. 38, n° 9, p. 11804–11812.
- Song L., Kolar M., Xing E. P. (2009). Time-varying dynamic bayesian networks. In *Advances in neural information processing systems*, p. 1732–1740.
- Xu J., Shelton C. R. (2008). Continuous time bayesian networks for host level network intrusion detection. In *Machine learning and knowledge discovery in databases*, p. 613–627. Springer.
- Xu J., Shelton C. R. (2010). Intrusion detection using continuous time bayesian networks. *Journal of Artificial Intelligence Research*, p. 745–774.
- Yeung D.-Y., Ding Y. (2003). Host-based intrusion detection using dynamic and static behavioral models. *Pattern recognition*, vol. 36, n° 1, p. 229–243.
- Yu S.-Z. (2010). Hidden semi-markov models. *Artificial intelligence*, vol. 174, n° 2, p. 215–243.
- Zanero S., Serazzi G. (2008). Unsupervised learning algorithms for intrusion detection. In *Network operations and management symposium, 2008. noms 2008. ieee*, p. 1043–1048.
- Zivot E., Andrews D. W. (2012). Further evidence on the great crash, the oil-price shock, and the unit-root hypothesis. *Journal of Business & Economic Statistics*.

