

République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Ferhat Abbas Sétif 1
Faculté des Sciences



MEMOIRE DE MASTER

Domaine: Math et Informatique

Département: Informatique

Spécialité: Informatique fondamentale et intelligence artificielle

Thème

Classification hybride (Hard et Soft) d'items éducatifs similaires

Présenté par :

Abdou Abarchi Aboubacar

Encadré par :

M^{eme} Harbouche Khadidja

Devant le jury composé de :

Président : ***M^r Abdellatif HABES***

Examinateuse : ***M_{me} Houda HAFI***

Septembre 2021

MASTER

Remerciement et dedicaces

Au terme de ce travail, je tiens à remercier Dieu le tout puissant de m'avoir donné le courage, la volonté et la patience pour achever ce travail, à mes très chers parents qui m'ont énormément soutenu durant mes études et à toutes les personnes qui ont contribué au succès de mon mémoire de fin d'étude.

Je voudrais dans un premier temps remercier, ma directrice de mémoire Madame Harbouche Khadidja, pour sa patience, sa disponibilité et surtout ses judicieux conseils, qui ont contribué à alimenter ma réflexion. Je tiens à exprimer aussi toute ma reconnaissance à Madame Ahlem Drif de m'avoir encadré, orienté, aidé et conseillé.

Je tiens à exprimer ma gratitude aux membres du jury pour avoir accepté de juger ce travail. Un énorme merci à ma famille, mes amis de prêt ou de loin pour leurs éternels soutien et leurs confiances.

Je remercie également toute l'équipe pédagogique de l'université Ferhat Abbas Sétif 1 et les intervenants professionnels responsables de ma formation, pour avoir assuré la partie théorique de celle-ci.

Enfin, je remercie mes amis Çalahadine Adamou Fodi, et Ismael Salifou qui ont toujours été là pour moi. Leur soutien inconditionnel et leurs encouragements ont été d'une grande aide et je leur souhaite bonne chance pour la rédaction de leur mémoire de master.

Je dédie ce modeste travail à :

A ma très chère mère et à mon très cher père, pour leurs amours, leurs tendresses, leurs sacrifices déployés pour m'élever dignement et leurs prières tout au long de mes études.

A mes chers membres de la famille, en l'occurrence Amina, Aichatou, Abass, et mon cousin Ousmane Salifou, pour leurs encouragement permanent. Autant de phrases et d'expressions aussi éloquente soient elles, ne sauraient exprimer l'amour et l'affection que j'éprouve pour vous. Que Dieu vous préserve de tout mal, vous comble d'une santé de fer, de bonheur et vous procure une longue vie !!!

Résumé

Les modèles de la théorie des réponses au items et les méthodes d'exploration de données éducatives sont très utilisés pour analyser les performances des apprenants sur des questionnaires et la différence qui existe entre les questionnaires (items) comme le degré de difficulté. Ces méthodes sont largement utilisées pour optimiser la réussite et améliorer l'environnement (le système) dans lequel les utilisateurs apprennent. L'inférence bayésienne appliquer aux modèles IRT permet de capter les attributs du sujet (sa compétence) et les propriétés des items (sa difficulté, son pouvoir de discrimination, la probabilité de deviner la réponse correcte) pour pouvoir prédire la probabilité de réussite et ainsi comparer les prédictions des modèles et les scores collecter. La comparaison entre les scores prédit des apprenants et ceux observés et surtout les valeurs des paramètres permettent d'évaluer les échelles des mesures qui sont utilisés par le système, la qualité des données collecter et améliorer potentiellement la notation des scores des apprenants. Une fois les critères précédents valider, les données peuvent donc être utilisé avec les méthodes d'exploration de données éducatives qui permettent de transformer des données brutes de grande quantité en informations utiles pour découvrir des modèles et établir des tendances et des relations pour résoudre des problèmes et de prédire les tendances futures. Les méthodes EDM (Educational Data Mining) comme le clustering peuvent être utilisé pour regrouper les items en composante de connaissance en utilisant des approches basées sur des modèles et des approches basées sur la similarité entre items et ainsi séquencer l'apprentissage afin que les apprenants maitrisent les compétences préalables avant de passer aux compétences qui en dépendent.

Ce projet de mémoire propose une approche pour évaluer la qualité des données éducative avec des modèles IRT (modèle de Rasch, modèle logistique a deux et trois paramètres) et un regroupement des items en composante de connaissances avec des méthodes de clustering hiérarchique, partitionnel et floue respectivement le clustering hiérarchique agglomératif, k-means clustering, c-means clustering, et le nombre de réponse correcte et incorrecte avec aide et sans aide comme critère pour la construction de la matrice de similarité.

Mots clés : Exploration de données éducatives, modèle d'apprenant, ACP, méthodes de clustering, théorie de la réponse aux items, estimations MCMC.

Abstract

Models of item response theory and educational data mining methods are widely used to analyze learner performance on questionnaires and the difference between questionnaires (items) such as degree of difficulty. These methods are widely used to optimize success and improve the environment (the system) in which users learn. Bayesian inference applied to IRT models makes it possible to capture the subject's attributes (his competence) and the properties of the items (his difficulty, his power of discrimination, the probability of guessing the correct answer) in order to be able to predict the probability of success and thus compare model predictions and scores collect. The comparison between the predicted scores of the learners and those observed and especially the values of the parameters make it possible to evaluate the scales of measures that are used by the system, the quality of the data to collect and potentially improve the scoring of the learners' scores. Once the above criteria are validated, the data can therefore be used with educational data mining methods that transform large amounts of raw data into useful information to discover patterns and establish trends and relationships to solve problems. and predict future trends. EDM (Educational Data Mining) methods such as clustering can be used to group items into knowledge components using model-based approaches and approaches based on similarity between items and thus sequence learning so that learners master the prerequisite skills before moving on to the skills that depend on them.

This thesis project proposes an approach to evaluate the quality of educational data with IRT models (Rasch model, two and three parameters logistic model) and a grouping of items into knowledge component with hierarchical, partitional and fuzzy clustering methods. respectively the hierarchical agglomerative clustering, k-means clustering, c-means clustering, and the number of correct and incorrect answers with help and without help as criteria for the construction of the similarity matrix.

Keywords: Educational Data mining, Learner model, PCA, Clustering Methods, Items response theory, MCMC estimations.

Table des matières

Table des figures	x
Liste des tableaux	xii
Liste des Acronymes	xii
Introduction Générale	1
I État de l'art	4
1 Educational data minig	5
1.1 Introduction	6
1.2 Data mining	6
1.2.1 Définition	6
1.2.2 Les méthodes du data mining	7
1.3 Intelligent Tutorial Systems (ITS)	7
1.3.1 Définition	7
1.3.2 L'architecture d'un ITS	8
1.4 Educational Data mining	9
1.4.1 Définition	9
1.4.2 Processus d'application du DM en éducation	10
1.4.2.1 Les méthodes d'analyse et d'exploration appliquer dans EDM	11
1.4.2.1.1 Modèle supervisée	12
1.4.2.1.2 Modèle non supervisée	12
1.4.2.1.3 L'estimation des paramètres	12
1.4.2.1.4 L'exploration de relations (Relationship mining)	13
1.4.2.1.5 La distillation des données pour le jugement humain (Distillation of data for human judgment)	13
1.4.2.1.6 Discovery with models	13
1.5 Conclusion	14

TABLE DES MATIÈRES

2 Modèle de l'apprenant et découverte des prérequis	15
2.1 Introduction	16
2.2 Learner Model	16
2.2.1 Définition	16
2.2.2 Caractéristiques des méthodes utilisées pour la modélisation de l'apprenant	16
2.2.3 Composantes du modèle de l'apprenant	18
2.2.3.1 Modèle cognitif	19
2.2.3.2 Modèle d'inférence	19
2.2.3.3 Modèle émotionnel	20
2.2.4 Utilitaire du modèle de l'apprenant	20
2.2.5 Types de modèles d'apprentissage	20
2.2.6 Contenu du modèle de l'apprenant	21
2.3 Modèle de compétence	21
2.3.1 Granularité	21
2.3.2 Relations pré-requises	23
2.4 Découverte des prérequis	24
2.5 Conclusion	25
3 L'approche items-to-skills mapping	26
3.1 Introduction	27
3.2 Connaissance(Knowledge)	27
3.2.1 Définition	27
3.2.2 Composante connaissance(Knowledge component)	27
3.2.3 Les types de composante de connaissance	28
3.3 Mappage des éléments aux compétences	29
3.3.1 Définition	29
3.3.2 items-to-skills mapping Structure	29
3.3.2.1 L'approche basée sur un modèle	30
3.3.2.1.1 Définition	30
3.3.2.1.2 Model Based Techniques	31
3.3.3 Similarity-based approach	31
3.3.3.1 Définition	31
3.3.3.2 Processus de calcul de la similarité des items	32
3.3.3.2.1 Données d'entrée	33
3.3.3.2.2 Données calculées	33
3.3.3.3 Mesures de similarité des items	33
3.4 Conclusion	34
4 Inférence bayésienne et la théorie de la réponse aux items	35
4.1 Introduction	36
4.2 Inférence bayésienne	37
4.2.1 Définition	37
4.2.2 probabilité conditionnelle et théorème de bayes	37
4.2.3 L'approche Bayésienne	38

TABLE DES MATIÈRES

4.2.4	Théorème de bayes dans la cadre de l'application de l'approche Bayésienne	38
4.2.4.1	Posterior distribution	39
4.2.4.2	Likelihood	39
4.2.4.3	Prior distribution	40
4.2.4.4	Chaine de Markov Monte Carlo (MCMC)	40
4.2.4.4.1	Théorème de convergence des chaînes de Markov [1] [2] [3]	42
4.2.4.4.2	Algorithme de Metropolis-Hastings	43
4.2.4.4.3	Hamiltonian Monte Carlo	43
4.2.4.4.4	No-U-Turn Sampler	44
4.2.5	Méthodes de vérifications et de comparaison de modèles	46
4.3	La théorie des réponses aux items	48
4.3.1	IRT dans la recherche organisationnelle	49
4.3.2	Les Modèles IRT	51
4.3.2.1	Le modèle Rasch	53
4.3.2.2	Modèle logistique à deux paramètres	54
4.3.2.3	Modèle logistique à trois paramètres	55
4.3.3	Estimation des paramètres	55
4.4	Un ajustement bayésien des réponses aux items avec Stan	55
4.4.1	IRT avec Stan	56
4.4.1.1	Déclaration des données	56
4.4.1.2	Les paramètres du modèle de Rasch, 2PL et 3PL	56
4.4.1.3	Le modèle de Rasch, 2PL et 3PL	57
4.4.1.4	Prédiction postérieure	58
4.5	Conclusion	58
5	Clustering Hard et Soft	60
5.1	Introduction	61
5.2	Les métriques	62
5.3	Critère de liaison	62
5.3.1	Distance inter-cluster	63
5.3.1.1	Distance de liaison unique	63
5.3.1.2	Distance de liaison complète	64
5.3.1.3	Distance de liaison moyenne	64
5.3.1.4	Centroïde Linkage Distance	64
5.3.1.5	Distance de liaison moyenne du centre de gravité	64
5.3.2	Distance intra-cluster	65
5.3.2.1	Distance de diamètre complet	65
5.3.2.2	Distance de diamètre moyen	65
5.3.2.3	Diamètre barycentre Distance	65
5.4	Le clustering hiérarchique	65
5.4.1	Introduction	65
5.4.2	Clustering hiérarchique agglomératif	66
5.4.2.1	Exemple de clustering agglomératif	66

TABLE DES MATIÈRES

5.4.3	Clustering hiérarchique divisive	67
5.4.4	Procédures de fractionnement des clusters	68
5.4.5	Evaluations des bipartitions	68
5.4.6	Déterminations de niveaux de nœuds	68
5.4.7	Conclusion	69
5.5	Le clustering partitionnel	69
5.5.1	K-means clustering	69
5.5.1.1	Le fonctionnement de l'algorithme k-means	69
5.5.1.2	Les étapes de l'algorithme	69
5.5.1.3	Les lacunes de l'algorithme k-means	71
5.5.1.4	Les solutions aux lacunes de l'algorithme k-means	71
5.6	Fuzzy clustering	72
5.6.1	Introduction	72
5.6.2	Fuzzy C-means clustering :	73
5.6.2.1	Algorithme FCM :	74
5.6.3	Conclusion	74
5.7	Les autres méthodes de clustering	75
5.8	La différence entre le clustering hiérarchique, partitionnel et Fuzzy clustering	75
5.9	Indice de validité du clustering	76
5.10	Conclusion	78
II	Contributions	79
6	Contribution and Results	80
6.1	Introduction	81
6.2	Approche proposée	81
6.3	Implémentations et résultats expérimentaux	81
6.3.1	Outils de développement	81
6.3.1.1	Materiels	81
6.3.1.2	Outils et packages	82
6.3.2	Collecte et préparation des données	84
6.3.3	Modèle IRT pour une inférence bayésienne	86
6.3.3.1	Spécification et chargement du modèle	87
6.3.3.2	Compilation du modèle et échantillonnage	88
6.3.3.3	Résultats d'échantillonnage, diagnostique du modèle, pré-diction et validation	89
6.3.4	Clustering hard et soft de la matrice de similarité	94
6.3.4.1	Création de la matrice de similarité	94
6.3.4.2	Analyse en composantes principales (ACP)	96
6.3.4.2.1	Visualisation de données	97
6.3.4.2.2	Compression de données	97
6.3.4.3	Clustering hard	97
6.3.4.3.1	K-means Clustering	97
6.3.4.3.2	Clustering hiérarchique agglomérative	98
6.3.4.4	Clustering Soft	99
6.3.4.5	Validation du clustering	102
6.4	Conclusion	103

TABLE DES MATIÈRES

Conclusion Générale	103
Annexe	106
A Les modèles IRT en code Stan.	106
B Les distributions des modèles.	112
Bibliographie	116

Table des figures

1.1	Les étapes de l'extraction de connaissances à partir de données	7
1.2	L'architecture d'un ITS	8
1.3	Principaux domaines liés à l'exploration de données éducatives	10
1.4	Data mining application process applied in education	11
1.5	Les techniques d'exploration de données éducatives.	11
2.1	Composantes du modèle de l'apprenant	18
2.2	Deux alternatives pour modéliser les relations d'agrégation.	22
2.3	Agrégation de modélisation de réseau bayésien et relations préalables simultanément	24
2.4	Modèles d'équations structurelles	25
3.1	Composante Connaissances	28
3.2	La structure du mappage des éléments aux compétences	29
3.3	L'approche basée sur un modèle	30
3.4	Q-matrix	31
3.5	Illustration de l'approche générale de l'analyse des éléments basée sur la similarité des éléments.	32
3.6	L'approche générale du calcul et de l'application de la similarité des éléments.	32
4.1	La distribution à priori, à posteriori et celle de la fonction de vraisemblance.	39
4.2	La phase « burn-in » d'une chaîne de Markov et celle de la convergence de la chaîne.	41
4.3	Les modèles IRT.	53
4.4	Probabilité de bonne réponse, conditionnelle à θ , pour les items avec $\beta = -1,0$ et 1.	54
5.1	Distances intracluster et intercluster.	63
5.2	La direction du clustering hiérarchique agglomérative et divisive.	67
5.3	Les étapes de l'algorithme k-means.	70
5.4	La méthode Elbow.	72
6.1	Le pre-processing du dataset.	85
6.2	Les étapes du workflow bayésien [4].	87
6.3	L'objet stanfit du modèle de Rasch.	89
6.4	Les distributions postérieures du modèle de Rasch.	90
6.5	Distribution postérieure de la capacité de l'apprenant[72] et de la difficulté de l'item[563] du modèle de Rasch.	90
6.6	BFMI du modèle de Rasch.	91
6.7	Le Rhat du modèle de Rasch.	91

TABLE DES FIGURES

6.8	La sortie de la divergence du modèle de Rasch.	92
6.9	Représentation graphique du taux de vrais positifs par rapport au taux de faux positifs des trois modèles utilisés.	93
6.10	Les étapes de calcul de la matrice de similarité.	94
6.11	Clusters de la méthode agglomérante	99
6.12	Visualisation des clusters de la méthode fuzzy et le degré d'appartenance des items à chaque cluster.	101
B.1	Les distributions postérieures du modèle logistique à deux paramètres. . . .	112
B.2	Distribution postérieure de la capacité de l'apprenant[72], de la difficulté de l'item[563] et de la discrimination de l'item[563] du modèle 2PL.	113
B.3	Les distributions postérieures du modèle logistique à trois paramètres. . . .	114
B.4	Distribution postérieure de la capacité de l'apprenant[72], de la difficulté de l'item[563], de la discrimination de l'item[563] et de la chance de deviner la réponse correcte de l'item[563] du modèle 2PL.	115

Liste des tableaux

2.1	Caractéristiques des méthodes utilisées pour la modélisation des apprenants partie[1]	17
2.2	Caractéristiques des méthodes utilisées pour la modélisation des apprenants partie[2]	18
3.1	La matrice d'accord pour deux items.	33
3.2	Coefficients de mesures de similarités.	34
4.1	Résumé des applications de l'IRT [5].	50
4.2	Résumé des quelques modèles IRT [5].	52
5.1	Les métriques	62
5.2	Les critères de liaison	62
5.3	Les autres méthodes de clustering	75
5.4	Indice de validité du clustering Hard	77
5.5	Indice de validité du clustering Soft	78
6.1	Caractéristiques du matériels utilisés	81
6.2	Indice de validité du clustering Hard	82
6.3	Indice de validité du clustering Hard	83
6.4	Indice de validité du clustering Hard	84
6.5	Description et statistiques du jeu de données.	84
6.6	Les différents scores du modèle de Rasch, modèle logistique à deux paramètres et le modèle à trois paramètres.	93
6.7	Rapport de classement (réponse correcte et incorrecte) entre les données observées et les prédictions du modèle de Rasch.	93
6.8	Résultats des indices de validité du clustering.	102

Liste des Acronymes

EDM	Educational Data Mining
ITS	Intelligent Tutorial Systems
IALT	Intelligent Computer Assisted Instruction
CAI	Computer Assisted Instruction
BKT	Bayesian Knowledge Tracing
AHC	Agglomerative Hierarchical Clustering
DIANA	Divisive Analysis Clustering
PDDP	Principal Directions Divisive Partitioning
FCM	Fuzzy C-means
IRT	Item Response Theory
TRI	Théorie des Réponses aux Items
MCMC	Markov Chain Monte Carlo
CTT	Classical Test Theory
DIF	Differential Item Functioning
DTF	Differential Test Functioning
CFA	Confirmatory Factor Analysis
IER	Insufficient Effort Responding
1PL	One-parameter logistic model
2PL	Two-parameter logistic model
3PL	Three-parameter logistic model
M2PL	Multidimensional two-parameter logistic model
GRM	Graded response model
GGUM	Generalized graded unfolding model
M-H,MH	Metropolis-Hastings
NUTS	No-U-Turn Sampler
HMC	Hamiltonian Monte Carlo
AIC	Akaike Information Criteria
DIC	Deviance Information Criterion
WAIC	Widely Applicable Information Criterion aussi connu sous le nom Watanabe-Akaike Information Criteria
LOO	Leave-one-out cross-validation
ADVI	Automatic Differentiation Variational Inference

Introduction générale

Les systèmes informatisés conservent généralement des données détaillées des interactions utilisateur-système, plus précisément des interactions système-apprenant dans les systèmes éducatifs et les systèmes de tutorat intelligent (ITS) en particulier. Ces données détaillées qui sont dans une grande base de données offrent des opportunités pour étudier ces données récolter. Dans les systèmes éducatifs et les systèmes de tutorat intelligent (ITS), les données sont enregistrées dans une base de données à chaque transaction qui est une interaction entre l'apprenant (élève, étudiant) et le système de tutorat. La transaction se définit comme chaque tentative correcte ou incorrect, ou encore quand l'apprenant demande d'aide ou conseil (généralement appelé « hint » en anglais) sur un « item » qui se définit comme une question ou une tache que l'apprenant doit accomplir. Les opportunités qu'offre les données collecter à partir d'un ITS sont : étudier le comportement de l'apprenant face au system, extraction des modèles supervisés et non supervisé à partir des données, estimations des paramètres Bayesian Knowledge Tracing (BKT) et de capter plus de nuances dans le comportement humain et d'estimer la capacité de l'apprenant à réussir un item et aussi d'estimer la difficulté, la discrimination des items et le paramètre de pseudo-chance. Plusieurs modèle peuvent être obtenu à partir des données des ITS : des « model-based approach » et « similarity-based approach » qui sont obtenu après avoir mapper les éléments (« items ») aux compétences (« skills »); des modèles Bayesian Knowledge Tracing (BKT); des modèles de la théorie des réponses aux items TRI 1PL, 2PL, 3PL (modèles logistique à un, deux et trois paramètres).

Problématique

L'évaluation pédagogique concerne l'inférence sur les connaissances, les compétences et les réalisations des élèves parce que les données ne sont jamais aussi complètes et sans équivoque qu'elles garantissent la certitude. La théorie des réponses aux items (Item Response Theory IRT) intervient dans cette situation pour faire un ajustement bayésien des réponses aux items en améliorant potentiellement la notation des tests et d'éclairer une décision sur

INTRODUCTION GÉNÉRALE

l'intérêt de collecter le jeu de données où IRT a été appliquer.

Aussi, dans les systèmes éducatifs, et en particulier dans les systèmes de tutorat intelligent (ITS), beaucoup d'aptitudes ont une forte relation causale dans laquelle une aptitude doit être présentée avant une autre (hiérarchie des compétences selon les pré-requis), ce qui incite à séquencer l'apprentissage afin que les apprenants maîtrisent les compétences préalables avant de passer aux compétences qui en dépendent. Dans ce cas plusieurs méthodes peuvent être utilisé comme les méthodes basées sur la similarité et ceux sur un modèle après avoir mapper les éléments aux compétences (Item-to skill mappings aussi appelé Q-matrix). Les méthodes basées sur la similarité, pour réaliser ce mappage, s'appuient sur l'hypothèse que les apprenants auront tendance à avoir des performances similaires sur des éléments qui nécessitent la même compétence. Ces méthodes cherchent d'abord à calculer une mesure ou un degré de similarité pour chaque paire d'items. Le calcul de mesure de similarité des éléments est effectuer en utilisant certains coefficients de similarité (Pearson, Kappa, Yule, Jaccard, Sokal et Fisher), tout en prenant en compte en plus des informations correctes et incorrectes (obtenu par l'apprenant sur un item) d'autres caractéristiques comportementales telles que le temps de réponse, le comportement de l'apprenant, le nombre de tentative sur un item, le hasard, etc.

Objectif et le Travail réalisé

Nous avons utilisé IRT dans l'implémentation pour un ajustement bayésien des réponses aux items avant de décider si oui ou non le jeu de données vaille le coup. L'inférence bayésienne qui est une méthode d'apprentissage des valeurs des paramètres dans les modèles statistiques à partir de données, est utilisée avec les modèles IRT pour faire l'ajustement des réponses obtenu par les apprenants.

Après la validation du jeu de données, une matrice de similarité entre les items a été créer selon le critère suivant : chaque item à un nombre de réponse correct avec aide (hints) et sans aide, et incorrect avec aide et sans aide. Ensuite un clustering hard et soft a été effectuer avec la matrice de similarité.

Plan du document :

Ce document est organisé en deux parties :

La première partie : présente l'état de l'art sur les différents domaines entrant en jeu dans le cadre de ce mémoire. Elle est composée de cinq chapitres à savoir educational data minig, modèle de l'apprenant et découverte des prérequis, l'approche items-to-skills mapping, l'inférence bayésienne et la théorie de la réponse aux items, et Clustering Hard et Soft.

Dans le premier chapitre nous présentons une définition des terminologies du data mining, du data mining éducatif et de leurs techniques, en précisant l'intégration du data mining dans l'éducation. Une présentation des concepts du modèle de l'apprenant, et une brève description des méthodes utilisées pour la modélisation de l'apprenant. Et comment les pré-requis de l'apprenant sont découverts dans le chapitre [2](#). Une définition des terminologies de la connaissance, composante de la connaissance et les approches pour le mappage des éléments aux compétences dans le troisième chapitre. Dans le quatrième chapitre nous avons défini et présenté les concepts de l'inférence bayésienne, de la théorie des réponses aux items, de l'application de l'inférence bayésienne pour faire l'estimation des paramètres des modèles IRT. Et dans le cinquième chapitre, nous présentons le clustering hard et soft plus précisément le clustering hiérarchique, partitionnel, « fuzzy clustering » (clustering flou), une brève introduction d'autre méthode de clustering et les indices de validation de clustering.

La deuxième partie : présente les contributions essentielles apportées. Elle est composée d'un seul chapitre dans lequel nous présentons les étapes parcourus jusqu'à l'obtention des résultats.

Enfin, nous clôturons notre modeste travail par une conclusion générale ainsi que les différentes perspectives à développer dans l'avenir pour toute éventuelle amélioration de ce travail.

Première partie

État de l'art

CHAPITRE 1

Educational data mining

Sommaire

1.1	Introduction	6
1.2	Data mining	6
1.3	Intelligent Tutorial Systems (ITS)	7
1.4	Educational Data mining	9
1.5	Conclusion	14

1.1 Introduction

Du fait que les systèmes informatisés conservent des journaux détaillés comme ceux des interactions utilisateur-système, ces journaux détaillés qui sont généralement dans une grande base de données ouvrent de nouvelles opportunités pour étudier ces données récolter. L'EDM est un domaine qui emprunte et étend des domaines connexes tels que l'apprentissage automatique (l'étude des programmes informatiques qui apprennent et s'améliorent avec des données empiriques), l'exploration du texte (approches pour trouver des modèles dans du texte), la psychométrie (l'étude des instruments psychologiques pour mesurer les compétences et les traits humains) et l'analyse des journaux Web (approches pour identifier les profils d'utilisateurs et la navigation modèles d'utilisateurs du site Web). Dans ce chapitre, nous présentons les techniques d'exploration de données, et un ensemble de technique d'analyse pour une variété de problèmes dans la recherche en éducation.

1.2 Data mining

1.2.1 Définition

L'exploration de données est un processus itératif et interactif visant à découvrir des modèles de données efficaces, nouveaux, utiles et compréhensibles dans de grandes bases de données. L'exploration de données, également appelée découverte de connaissances dans une base de données, fait référence à l'extraction de connaissances à partir de grandes quantités de données. Elle est utilisée pour extraire de grands volumes de données pour découvrir des modèles cachés et des relations utiles dans la prise de décision. Bien que l'exploration de données et la découverte de connaissances de base de données soient souvent traitées comme des synonymes, elles font en fait partie du processus de découverte des connaissances. La séquence des étapes identifiées lors de l'extraction des connaissances à partir des données est illustrée à la figure 1.1 [6].

L'exploration de données est un domaine interdisciplinaire utilisant à la fois des techniques d'apprentissage automatique, de reconnaissance de formes et l'utilisation de statistiques, de bases de données et de visualisation pour déterminer les façons d'utiliser l'extraction d'informations à partir de très grandes bases de données [7].

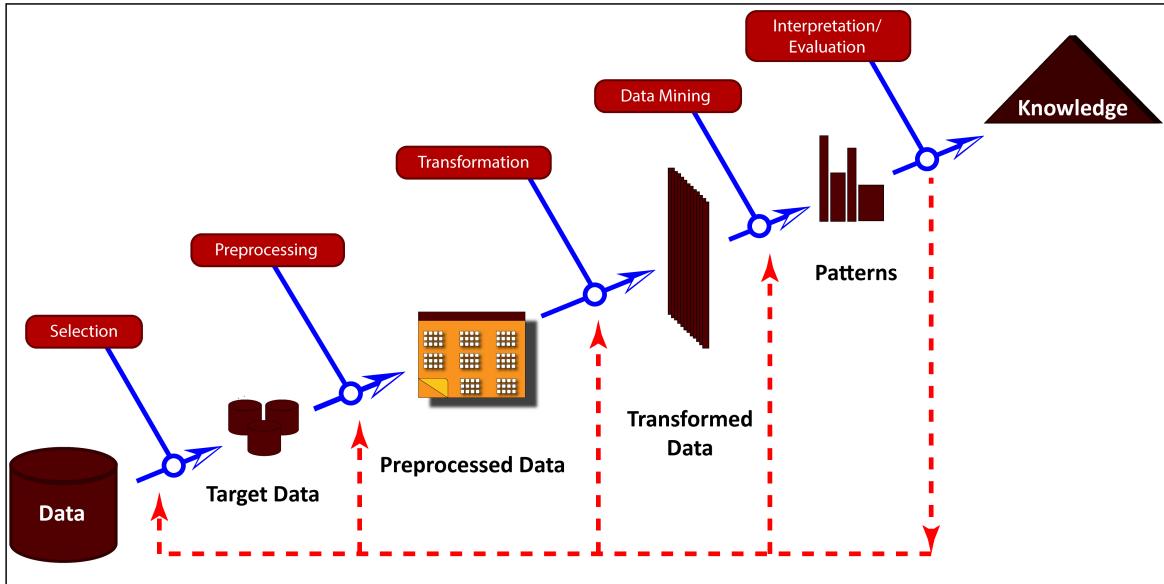


FIGURE 1.1 : Les étapes de l'extraction de connaissances à partir de données

1.2.2 Les méthodes du data mining

Le clustering et la classification sont deux tâches fondamentales dans l’exploration de données. La classification est principalement utilisée comme méthode d’apprentissage supervisé, le clustering pour l’apprentissage non supervisé (certains modèles de clustering sont pour les deux). Le but du clustering est descriptif, celui de la classification est prédictif [8]. « Comprendre notre monde nécessite de conceptualiser les similitudes et les différences entre les entités qui le composent » [9], de ce fait, certains coefficients de corrélations sont utilisés pour calculer la similarité entre deux éléments ou plusieurs et, des métriques et les critères de liaison permettent qu’en eux de déterminer le lien entre plusieurs entités afin de les groupées dans une même classe (c’est le cas du clustering). Certaines méthodes comme : « rule of association », « preaching », « neural networks », « decision trees » sont utilisé aussi pour explorer et analyser des jeux de données.

1.3 Intelligent Tutorial Systems (ITS)

1.3.1 Définition

Les systèmes de tutorat intelligents (ITS) sont des environnements d’apprentissage informatisés issus de l’IAIT (Intelligent Computer Assisted Instruction) qui visent à personnaliser l’apprentissage. En effet, ils ont été développés pour répondre aux limites du CAI (Computer Assisted Instruction) en utilisant l’intelligence artificielle pour mettre en œuvre des systèmes

plus flexibles et interactifs qui s'adaptent aux besoins spécifiques de l'apprenant en évaluant et en diagnostiquant ses problèmes afin de fournir les assistances [10]. L'objectif est de mimicer le comportement d'un tuteur humain en sa qualité de pédagogue expert et d'expert dans le domaine. Ainsi, tout comme un tuteur, un logiciel de ce type a le potentiel d'amener l'apprenant à accomplir une tâche et à fournir un retour sur leurs actions. ITS répond ainsi à la nécessité de placer l'apprenant au centre du processus d'apprentissage.

Les ITS sont essentiellement un environnement de résolution de problèmes ou d'exercice. Ils favorisent l'apprentissage dans un domaine spécifique en guidant et en aidant l'apprenant. Parfois, ils exposent d'abord le contenu de la matière à l'apprenant ; parfois ils présentent directement les exercices qui permettront à l'apprenant d'assimiler les connaissances [11].

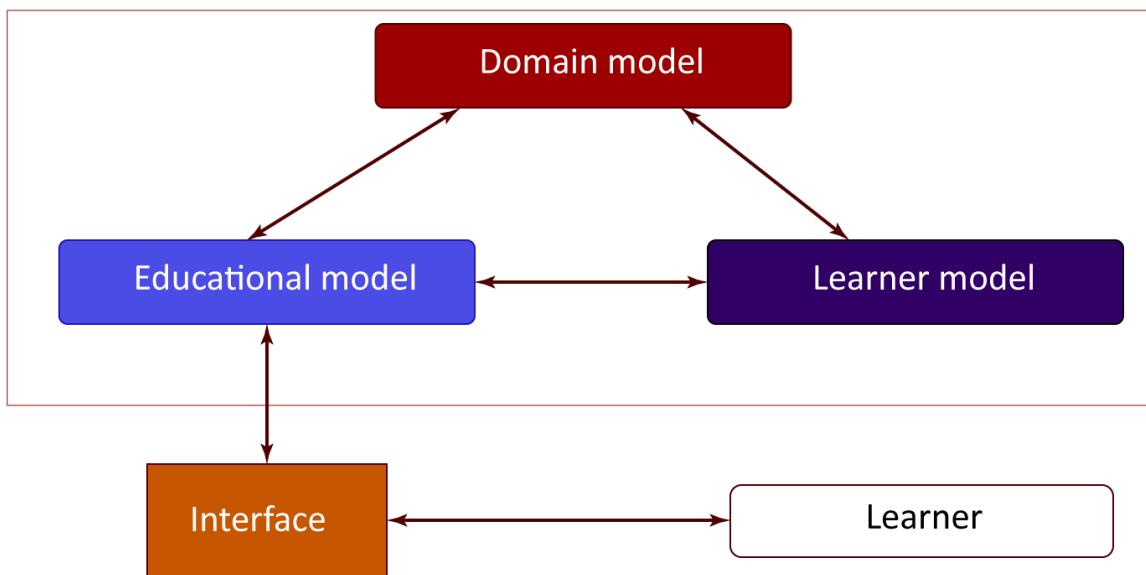


FIGURE 1.2 : L'architecture d'un ITS

1.3.2 L'architecture d'un ITS

L'architecture conceptuelle d'un ITS est illustrée à la figure 1.2 et comprend : [11]

- **Un modèle d'interface :** qui représente la couche de communication entre l'apprenant et le système, c'est-à-dire les interactions entre l'apprenant et le système. Cette interface peut faire varier le type d'environnement d'apprentissage et l'objectif étant de privilégier une approche de conception qui ne gêne pas l'apprentissage. La qualité de l'interaction peut influencer les résultats d'apprentissage. Par conséquent, les principaux problèmes avec les interfaces d'apprentissage sont les problèmes d'interaction personne / machine. Pour surmonter ces problèmes, il faut prêter attention à la convivialité pour que la charge mentale associée à l'interface soit négligeable, ainsi

qu'à l'utilité en facilitant l'accès, en permettant l'accès au domaine d'apprentissage et en soutenant la métacognition de l'apprenant. Les tendances actuelles s'orientent vers des dialogues tutoriels en langage naturel, l'intégration de la dimension affective dans l'interaction et les interfaces tangibles.

- **Modèle du domaine** : également appelé modèle expert, il représente l'expertise de l'enseignant dans le domaine, c'est-à-dire ce qui doit être enseigné. L'expertise peut être modélisée de trois manières : une approche « boîte noire » : appliquer toute méthode de raisonnement sur le domaine inexpliqué, sans aucune transparence pour l'utilisateur ; un système expert peut expliquer son raisonnement et un modèle cognitif : simuler la manière dont les humains utilisent les connaissances.
- **Un modèle d'apprentissage** : qui peut personnaliser l'apprentissage en tenant compte des spécificités de l'apprenant.
- **Un modèle éducatif** : qui sait enseigner.

1.4 Educational Data mining

1.4.1 Définition

La communauté d'exploration de données appliquée dans l'éducation a défini ce terme comme suit : L'exploration de données appliquée dans l'éducation est une discipline qui concerne le développement de méthodes permettant d'explorer les types uniques de données provenant des milieux éducatifs. Ces méthodes sont utilisées pour mieux comprendre le comportement des apprenants et environnement de leur apprentissage [12].

L'exploration de données éducatives (EDM) vise à développer, rechercher et appliquer des méthodes informatisées pour détecter des modèles dans de grandes collections de données éducatives qui seraient autrement difficiles ou impossibles à analyser en raison de l'énorme volume de données dans lesquelles elles existent [9].

Ce domaine est une forme d'intersection des trois domaines principaux tels que : l'informatique, l'éducation et les statistiques illustrés dans la figure 1.3.

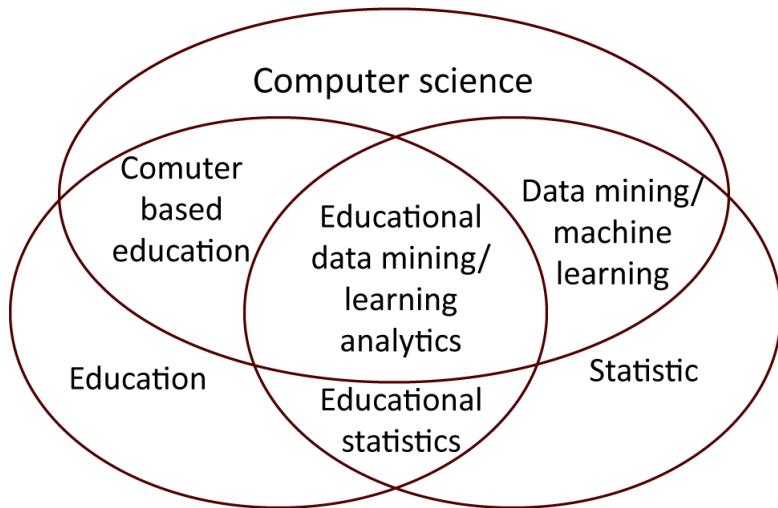


FIGURE 1.3 : Principaux domaines liés à l'exploration de données éducatives

Les techniques d'exploration de données (EDM) sont devenues plus importantes dans la recherche et le développement en raison du développement rapide de la technologie, de la croissance rapide des connaissances humaines et de l'augmentation du nombre de personnes détenant des systèmes d'enseignement informatisé.

1.4.2 Processus d'application du DM en éducation

Le processus d'application de l'exploration de données appliquée dans l'éducation peut être considéré comme un cycle itératif de formulation, de test et de raffinement d'hypothèses. Dans ce processus, l'objectif n'est pas seulement de transformer les données en connaissances, mais aussi de filtrer les connaissances extraites pour savoir comment les modifier. L'environnement éducatif pour améliorer l'apprentissage des apprenants est présenté dans la figure 1.4. Des études ont montré que l'application de l'EDM est similaire au processus Knowledge from Data (KDD). Ce processus commence par la collecte de données à utiliser à partir de l'environnement éducatif.

Les données brutes obtenues nécessitent un nettoyage et un prétraitement tels que : fusion de données hétérogènes, traitement des données manquantes, conversion de données d'une source de données à une autre, et données, etc. Cette phase nécessite souvent l'utilisation de certaines données techniques minières. Le résultat de ce dernier est un modèle capable de structurer les données stockées. Enfin, la dernière étape est l'interprétation et l'évaluation des résultats obtenus [13].

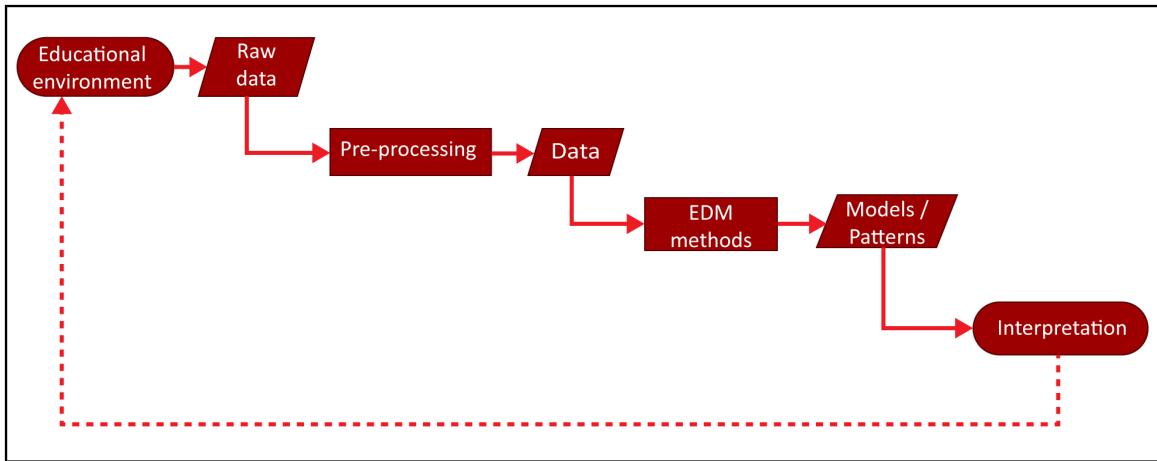


FIGURE 1.4 : Data mining application process applied in education

1.4.2.1 Les méthodes d'analyse et d'exploration appliquer dans EDM

Certaines méthodes du data mining ont été largement appliquées sur les données issues des systèmes éducatifs afin d'obtenir des connaissances cachées et des modèles. Les méthodes du data mining ont été appliquées dans plusieurs domaines de recherche tels que : le e-learning, le système de tuteur intelligent, text mining, réseaux sociaux, web mining, etc. Les méthodes d'analyse et d'exploration de données éducatives sont tirées de diverses sources littéraires, notamment l'apprentissage automatique, la psychométrie et d'autres domaines de la modélisation information, des statistiques et de la visualisation de l'information.

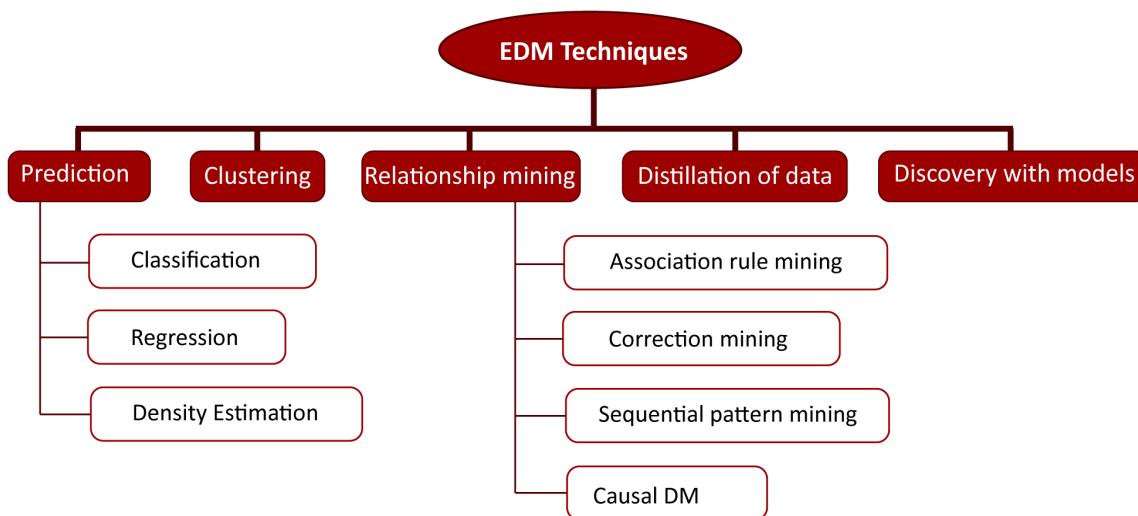


FIGURE 1.5 : Les techniques d'exploration de données éducatives.

1.4.2.1.1 Modèle supervisée

L’induction de modèle supervisée comprend des techniques d’apprentissage automatique qui déduisent des modèles de prédiction à partir de instances d’entraînement pour lesquelles les valeurs d’un attribut cible sont connues. Les modèles de prédiction acceptent des instances en entrée (généralement décrites comme un vecteur d’attribut) et en sortie une prédiction pour la variable cible. Les modèles qui prédisent des valeurs cibles catégorielles sont appelés modèles de classification ; des modèles qui prédire les valeurs cibles continues sont appelés modèles de régression. Les modèles de prédiction peuvent être basés sur différentes représentations, par exemple, les arbres de décision, les machines à vecteurs de support et modèle de régression linéaire [14].

1.4.2.1.2 Modèle non supervisée

L’induction de modèle non supervisée comprend des techniques d’apprentissage automatique qui déduisent des modèles à partir d’instances d’apprentissage pour lesquelles les valeurs d’un attribut cible ne sont pas connues. Les méthodes non supervisées utilisent une approche ascendante, c’est-à-dire que les modèles et les structures sont recherchés dans l’espace d’entrée sans catégories cibles explicitement définies ni exemples étiquetés. Une approche largement utilisée est le clustering, qui est utilisé pour identifier des groupes d’instances dans un ensemble d’apprentissage qui sont « similaires » à certains égards. En règle générale, une sorte de mesure de distance (par exemple, la distance euclidienne) est utilisée pour décider de la similitude des instances. Une fois un ensemble de clusters a été déterminé, de nouvelles instances peuvent être classées en déterminant le cluster le plus proche. Un algorithme de clustering bien connu est le clustering k-means [14].

1.4.2.1.3 L’estimation des paramètres

L’estimation des paramètres comprend des techniques statistiques pour déduire des paramètres de modèles probabilistes à partir d’un jeu de données. Ces modèles peuvent être utilisés pour prédire la probabilité d’événements d’intérêt. L’approche est basée sur l’hypothèse que le modèle a une forme paramétrique donnée (par exemple, une distribution gaussienne avec les paramètres moyenne et variance). Un exemple d’application en EDM est l’estimation des paramètres Bayesian Knowledge Tracing (BKT). Le BKT est utilisé pour déterminer la probabilité qu’un élève maîtrise une compétence sur la base de l’historique des performances passées. Un modèle BKT peut être compris comme un réseau bayésien dynamique avec quatre paramètres (priorité, estimation, glissement et taux d’apprentissage). Ces paramètres peuvent être déterminés, par exemple, avec l’algorithme Espérance-Maximisation [14]. Aussi l’utilisation de la théorie des réponses aux items (Item Response Theory IRT)

permet de capter plus de nuances dans le comportement humain et d'estimer la capacité de l'apprenant à réussir un item et aussi d'estimer la difficulté, la discrimination des items.

1.4.2.1.4 L'exploration de relations (Relationship mining)

L'exploration de relations concerne l'identification des relations entre les variables, des relations qui peuvent être de nature associative, corrélationnelle, séquentielle ou causale. Par exemple, une approche courante de l'exploration de règles d'association consiste à apprendre les règles SI-ALORS qui dépassent un seuil minimum de « support » et de « confiance ». La prise en charge indique la fréquence relative des transactions qui correspondent à la fois aux parties IF et ALORS de la règle. La confiance dénote la fréquence relative des transactions qui correspondent à la partie THEN de la règle dans l'ensemble de transactions qui correspondent à la partie IF. Apriori est, par exemple, un algorithme de règle d'association classique. Un exemple d'application en EDM est l'identification d'erreurs qui se produisent fréquemment ensemble (par exemple, les étudiants qui ont commis les erreurs A et B ont également souvent commis l'erreur C) [14].

1.4.2.1.5 La distillation des données pour le jugement humain (Distillation of data for human judgment)

La distillation des données pour le jugement humain vise à représenter les données de manière intelligible à l'aide de statistiques, méthodes de visualisation et interfaces d'information interactives. Par exemple, les performances moyennes les scores peuvent être calculés pour chaque élève et présentés à un enseignant par ordre croissant dans un graphique à barres. Un autre exemple est les courbes d'apprentissage, qui tracent les performances d'un élève (par exemple, le temps de réponse) par rapport au nombre d'occasions de pratiquer une compétence. Une courbe d'apprentissage idéale montre que la performance s'améliore en douceur et de façon monotone, suivant approximativement une loi de puissance ou une fonction exponentielle. D'un autre côté, les courbes d'apprentissage avec des pointes indiquent qu'une autre compétence pourrait interférer avec la compétence réellement modélisée, c'est-à-dire que le modèle de compétence pourrait être amélioré [14].

1.4.2.1.6 Discovery with models

Discovery with models (découverte avec des modèles) comprend des approches qui amorcent des modèles déjà existants pour faire des découvertes plutôt que de calculer de nouveaux modèles à partir de zéro. Par exemple, un modèle de prédiction pourrait être appliqué à un ensemble de données pour prédire les valeurs d'une catégorie cible d'intérêt. Les prédictions elles-mêmes pourraient être utilisées à nouveau comme données dans d'autres analyses, par

exemple, ils pourraient être corrélés avec une catégorie cible d'un autre modèle de prédiction. Un autre exemple consiste à scruter les différentes composantes d'un modèle de prédiction pour en savoir plus sur les facteurs qui influencent la prédiction [14].

1.5 Conclusion

Dans ce chapitre, nous avons passé en revue le domaine de l'exploration de données éducatives, en présentant brièvement ses applications et ses techniques. L'application des méthodes d'exploration de données dans le secteur de l'éducation est un phénomène intéressant. Les techniques d'exploration de données dans les organisations éducatives nous aident à apprendre davantage sur les performances des apprenants, le comportement des apprenants, la conception des programmes et à motiver les apprenants sur divers paramètres.

CHAPITRE 2

Modèle de l'apprenant et découverte des prérequis

Sommaire

2.1	Introduction	16
2.2	Learner Model	16
2.3	Modèle de compétence	21
2.4	Découverte des prérequis	24
2.5	Conclusion	25

2.1 Introduction

Le terme "apprentissage" fait souvent référence à l'apprentissage électronique ou e-learning. L'apprentissage en ligne, l'apprentissage à distance et l'apprentissage en ligne. L'apprentissage en ligne est connu comme des activités d'étude avec le soutien d'un ordinateur et d'un réseau.

Dans l'apprentissage en ligne, la communication entre les apprenants et les enseignants ou les apprenants et les systèmes se fait sur le réseau ou sur Internet, les supports d'apprentissage sont souvent des documents électroniques tels que des pages Web et des fichiers au lieu de livres traditionnels sur papier [15]. Dans ce chapitre, nous décrivons la modélisation de l'apprenant compte tenu de son importance dans tous les domaines de recherche. Nous commençons par une présentation des notions du modèle de l'apprenant et des caractéristiques des méthodes utilisées pour la modélisation de l'apprenant. Puis nous terminons avec la Découverte des prérequis.

2.2 Learner Model

2.2.1 Définition

Winkels décrit le modèle de l'apprenant comme une structure de données qui reflète l'état des connaissances supposées de l'apprenant sur un domaine cible (le domaine d'apprentissage) [16]. Selon Self, le modèle de l'apprenant devrait inclure les informations suivantes : les connaissances et les idées fausses de l'apprenant, ses compétences et son comportement. Pour déterminer ce que l'apprenant a fait, Self souligne qu'il est nécessaire d'avoir un historique des interactions de l'apprenant avec le système [17]. Greer considère le modèle de l'apprenant comme une représentation abstraite des croyances, des connaissances et des compétences de l'apprenant dans le système, y compris l'historique des actions de l'apprenant qui peut être analysé et interprété [18].

2.2.2 Caractéristiques des méthodes utilisées pour la modélisation de l'apprenant

Il existe plusieurs modèles existants classés selon la nature des informations extraites par le modèle et les méthodes utilisées pour les traiter 2.1 et 2.2. [19]

CHAPITRE 2. MODÈLE DE L'APPRENANT ET DÉCOUVERTE DES PRÉREQUIS

Méthodes	Applications	Avantage	Limites
Modèle de superposition	Modélisation des connaissances de l'apprenant à travers le modèle de connaissance.	-Haute expressivité des problèmes complexes et flexibilité du raisonnement humain.	-Impossible de détecter les erreurs de l'apprenant
	Identification des lacunes dans les connaissances des apprenants	- utilisation de la structure du modèle de connaissance.	-La complexité dépend de la structure du domaine. - Défaut de modéliser les erreurs de l'apprenant.
Modèle d'erreur	Identification des erreurs et fausses croyances de l'apprenant sur un domaine.	Identification des erreurs et fausses croyances de l'apprenant dans un domaine.	Difficulté à modéliser et à définir les erreurs.
Stéréotypes	Classification des apprenants selon des caractéristiques communes, les plus utilisées sont : le style d'apprentissage, le style cognitif, le niveau de connaissance et les préférences.	Initialisation de nouveau caractéristiques de l'apprenant.	
Les ontologies	-Modélisation du modèle de connaissance. -Réutilisation et partage des ressources.	-Ajout de sémantique aux relations entre concepts. -Réutilisation des ressources.	
Logique floue	-Expression linguistique du niveau de connaissance. -Évaluation des apprenants. -Identification du style de l'apprenant.	-Expression du degré d'incertitude. -Grande expressivité linguistique et logique.	-Consommation de ressources -Pas d'apprentissage.

TABLE 2.1 : Caractéristiques des méthodes utilisées pour la modélisation des apprenants partie[1]

CHAPITRE 2. MODÈLE DE L'APPRENANT ET DÉCOUVERTE DES PRÉREQUIS

Méthodes	Applications	Avantage	Limites
Réseau Bayésien	-Apprendre et prédire le comportement et le raisonnement de l'apprenant. -Classement des apprenants. -Prédiction des performances des apprenants.	-Apprendre et s'adapter à un environnement incertain. -Résoudre le problème « on-learning » -Fonction avec des données manquantes.	-Consommation de ressources -Difficulté à comprendre les inférences (interprétation) -Grand nombre de variables utilisées.
Programmation génétique	- Recommandation d'un chemin de navigation optimal. -Prédiction des performances des apprenants. -Amélioration des algorithmes de classification.	-Recherche globale -représentation flexible aux changements. -des recherches puissantes dans un espace de problèmes complexes et mal définis.	-Temps d'exécution -Convergence vers les optima locaux
Réseau flou de neurones	-Évaluation et prédiction des performances des apprenants.	-Grand Expressivement -Apprentissage -gestion des incertitudes.	La complexité augmente avec le nombre d'entrées du réseau.

TABLE 2.2 : Caractéristiques des méthodes utilisées pour la modélisation des apprenants partie[2]

2.2.3 Composantes du modèle de l'apprenant

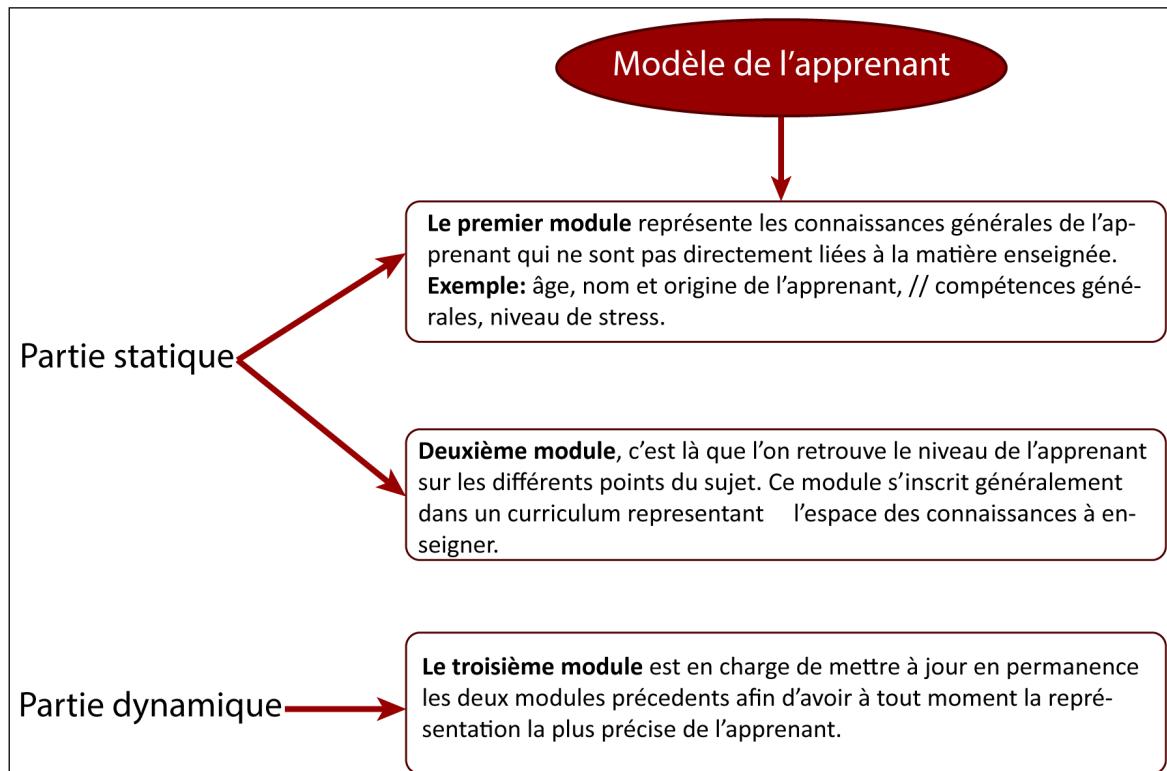


FIGURE 2.1 : Composantes du modèle de l'apprenant

Carr et Goldstein citent, par exemple, quatre informations nécessaires pour maintenir ce modèle : la difficulté du sujet, les questions directement posées par l'apprenant, la performance de l'apprenant et son expérience d'apprentissage [20]. Self définit le modèle de l'apprenant comme un 4-tuplet contenant les variables P (connaissance procédurale), C (connaissance conceptuelle), T (caractéristiques individuelles) et H (histoire) [21].

Un modèle de l'apprenant 2.1 comprend toujours des connaissances liées au domaine d'enseignement : ce que l'apprenant sait et ce que l'apprenant peut faire. Dans son état le plus complet, ce modèle comprend également des connaissances indépendantes du domaine enseigné. Certaines de ces connaissances sont liées à ses mécanismes d'apprentissage : comment fonctionne l'apprenant ? Comment découvre-t-il de nouveaux concepts ? De nouvelles techniques ? Etc. Les autres concernent les stratégies pédagogiques correspondantes : quels sont les types et modalités d'intervention les plus efficaces ? Etc. Selon Nkambou, le modèle de l'apprenant est identifié en trois parties : [22]

2.2.3.1 Modèle cognitif

Description de l'état des connaissances de l'apprenant par rapport au sujet considéré par le bailleur de fonds. Ces informations concernent :

- **Capacités** : les informations sur les capacités reflètent le niveau de connaissances de l'apprenant. Robert M. Gagne a classé les capacités de l'apprenant en cinq catégories : l'information verbale, les compétences intellectuelles, les attitudes, la motricité et les stratégies cognitives.
- **Objectifs** : les informations sur les objectifs indiquent si l'apprenant a déjà atteint ou non un objectif.
- **Ressources** : les informations sur les ressources (exercices, problèmes, tests, etc.) reflètent le fait qu'une ressource a déjà été utilisée par un apprenant et le contexte dans lequel cette ressource a été utilisée.
- **Relations éventuelles** : les informations sur les relations indiquent si l'apprenant a réussi ou échoué à établir une relation (par exemple, analogie, abstraction, cas particulier, etc.) entre deux connaissances (et par conséquent, la connaissance d'une relation entre deux connaissances est aussi connaissance (méta-connaissance)).

2.2.3.2 Modèle d'inférence

Cette partie est une sorte de moteur d'inférence qui fonctionne en permanence pour ajuster le modèle de l'apprenant. Il contient des règles qui lui permettent de raisonner sur le

modèle cognitif et sur le modèle psychologique (modèle affectif) pour inférer de nouvelles connaissances dans le modèle de l'apprenant.

2.2.3.3 Modèle émotionnel

Ce modèle est un ensemble de données qui permettent d'identifier la personnalité et les différentes facettes d'un apprenant. Il contient des connaissances sur les caractéristiques permanentes ou momentanées particulières de l'apprenant. Parmi ceux-ci, nous avons :

- Connaissance des conditions mentales PUX. Par exemple, l'apprenant est spatial ou verbal, réfléchi ou impulsif, etc.
- Connaissance des sentiments et de la personnalité. Par exemple, l'apprenant est calme ou anxieux. Il est attentif ou distractif. Etc.

2.2.4 Utilitaire du modèle de l'apprenant

Selon Ragnemalm, il y a quatre utilisations du modèle de l'apprenant : [23]

- Importance du modèle pour la planification de l'éducation : quel contenu faut-il enseigner ?
- Présentation du contenu pédagogique : quelles expériences sont appropriées pour le contenu d'apprentissage ?
- Le feedback du système doit prendre en compte les connaissances précédemment mobilisées par l'apprenant, ainsi que le contexte d'apprentissage actuel.
- Traiter les idées fausses : en les signalant à l'apprenant, en fournissant un contre-exemple ou en suscitant une discussion

2.2.5 Types de modèles d'apprentissage

- **Implicit** : lorsque des informations décrivant le comportement de l'apprenant et influençant le cours de l'interaction avec le système sont incorporées dans le système.
- **Explicit** : lorsque les informations sur l'apprenant sont intégrées et codées dans le système de manière explicite pour gérer l'interaction avec l'apprenant.
- **Static** : lorsque les connaissances de l'apprenant sont déterminées avant toute utilisation et ne peuvent pas être modifiées en cours de session.

- **dynamic** : lorsque des données peuvent être ajoutées ou modifiées pendant la session.
- **Specific** : quand il peut être adapté à une catégorie d'apprenants.
- **Surface** : lorsqu'elle contient des informations limitées qui ne peuvent expliquer l'état cognitif de l'apprenant.
- **Deep** : lorsqu'il contient des informations plus représentatives de l'état cognitif de l'apprenant. [24]

2.2.6 Contenu du modèle de l'apprenant

Le modèle de l'apprenant représente ce que le système « sait » de l'apprenant. Ces informations peuvent être de nature cognitive, comportementale ou psychologique. Les premiers modèles se sont concentrés sur les aspects cognitifs et ont mis l'accent sur les connaissances déclaratives, procédurales et heuristiques. Plus récemment, des modèles ont émergé pour représenter des aspects psychologiques : émotions et motivations. La modélisation de l'apprenant peut concerner un ou plusieurs aspects de l'apprenant : concepts, règles ou procédures de résolution maîtrisés, idées fausses, rapidité de résolution de problèmes, motivation à apprendre, capacité à réfléchir sur les connaissances apprises, aspects métacognitifs, etc. Le choix du contenu dépendra essentiellement du domaine d'enseignement, des objectifs didactiques et pédagogiques du système, des types d'interactions possibles avec l'apprenant, etc [25].

2.3 Modèle de compétence

Un modèle de compétences auquel il est fait référence ici ne concerne que les couches cachées du graphe des « couches de modélisation de l'apprenant » [26].

2.3.1 Granularité

La hiérarchie de granularité est une représentation courante d'un modèle d'apprenant [27]. Il décrit comment un domaine est décomposé en composante. Les composantes de connaissance dans un modèle de domaine sont généralement décrites à différents niveaux de granularité. Une hiérarchie de granularité capture différents niveaux de détail dans un type de réseau sémantique. Les relations d'agrégation sont utilisées pour décrire les relations entre les composants de connaissances à différents niveaux. Les relations d'agrégation peuvent être utilisées pour diviser un composant de connaissance composite en plusieurs composants

de connaissance à une granulométrie plus fine. Les observations sont généralement liés aux éléments de connaissance à un niveau plus fin. Les informations observées sont propagées par agrégation des liens vers les composants de connaissances aux niveaux plus grossiers.

Le schéma de clustering AND-OR est proposé par Collins et al pour capturer les relations d'agrégation et les groupes équivalents dans leur hiérarchie de granularité [28].

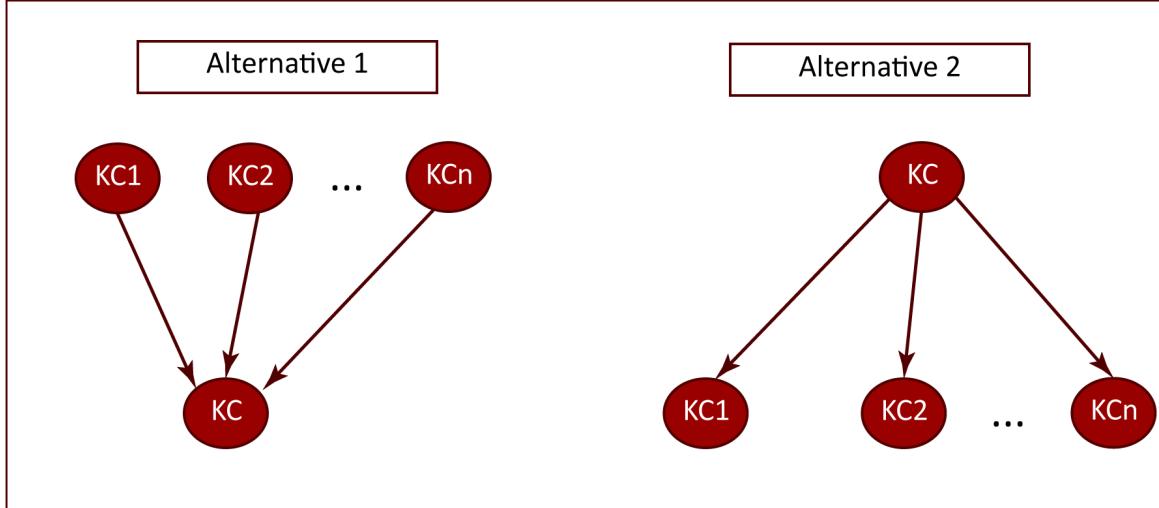


FIGURE 2.2 : Deux alternatives pour modéliser les relations d'agrégation.

Tchétagni et Nkambou ont proposé d'évaluer les connaissances des apprenants en logique propositionnelle à plusieurs niveaux de granularité. Ils ont utilisé le modèle alternatif 2 de la figure 2.2 pour représenter les relations d'agrégation dans leur hiérarchie. Ils ont souligné que dans cette architecture, il existe des restrictions sur la façon dont les preuves se propagent dans tout le réseau. Cela est dû au fait que deux nœuds enfants peuvent influencer leur parent, sans s'influencer mutuellement : ils sont séparés [29].

Carmona et Conejo ont utilisé dans la figure 2.2 le modèle alternatif 1 pour représenter les relations d'agrégation dans leur modèle d'apprenant utilisé dans MEDEA, un système ouvert pour le développement de systèmes de tuteurs intelligents. Certaines approches récentes ont abordé la granularité du modèle de compétences d'un point de vue statistique.

Dans ces approches, les modèles de compétences n'impliquent que les éléments de connaissance les plus fins, qui expliquent directement les comportements des apprenants. Un problème permanent dans un modèle d'apprenant est de savoir à quel niveau de granularité les compétences des apprenants doivent être modélisées [30]. Pardos et Heffernan ont exploré des modèles avec différents niveaux de granularité (modèles de compétences 1, 5, 39 et 106) et mesuré la précision de ces modèles en prédisant la performance des apprenants dans leur SIT, c'est-à-dire ASSISTment, ainsi que dans un test. Leurs résultats ont montré que plus la granularité du modèle de compétence est fine, meilleure est la prédiction des performances

de l'apprenant [31].

2.3.2 Relations pré-requises

Des relations préalables existent généralement entre les éléments de connaissance de certains domaines. Reye a analysé comment utiliser les réseaux bayésiens pour modéliser les relations antérieures. Ils ont déclaré que les probabilités conditionnelles dans un réseau bayésien devraient remplir certaines conditions. Par exemple, si la composante de connaissance A est une condition préalable de la composante de connaissance B, l'équation 2.1 doit être satisfaite. Cependant, ils ont également déclaré que la relation préalable n'est pas toujours stricte, de sorte qu'ils permettent une incertitude pour les probabilités conditionnelles. Les valeurs d'incertitude pour ces probabilités conditionnelles sont spécifiées par des experts dans leur méthode [32].

$$\begin{aligned} P(\text{learnerKnows}(A) \vee \text{learnerKnows}(B)) &= 1 \\ P(\text{learnerKnows}(B) \vee \text{learnerKnows}(A)) &= 0 \end{aligned} \tag{2.1}$$

Carmona et al. ont présenté les relations préalables à un modèle générique d'apprentissage du BN pour MEDEA, afin d'améliorer l'efficacité des mécanismes d'adaptation et le processus d'inférence. Ils ont utilisé une porte ET bruyante modifiée ou une porte OU bruyante modifiée pour modéliser les relations de prédicat [33].

Ferguson et al ont utilisé l'algorithme EM pour apprendre les paramètres cachés dans les BN et ont comparé le modèle de compétence plat (les compétences sont mutuellement indépendantes) avec le modèle de compétence hiérarchique (relations a priori entre des compétences données a priori) selon le critère d'information bayésien (BIC). Leurs résultats montrent qu'un modèle hiérarchique correspond mieux à leurs données que le modèle plat. [20=> a chercher]

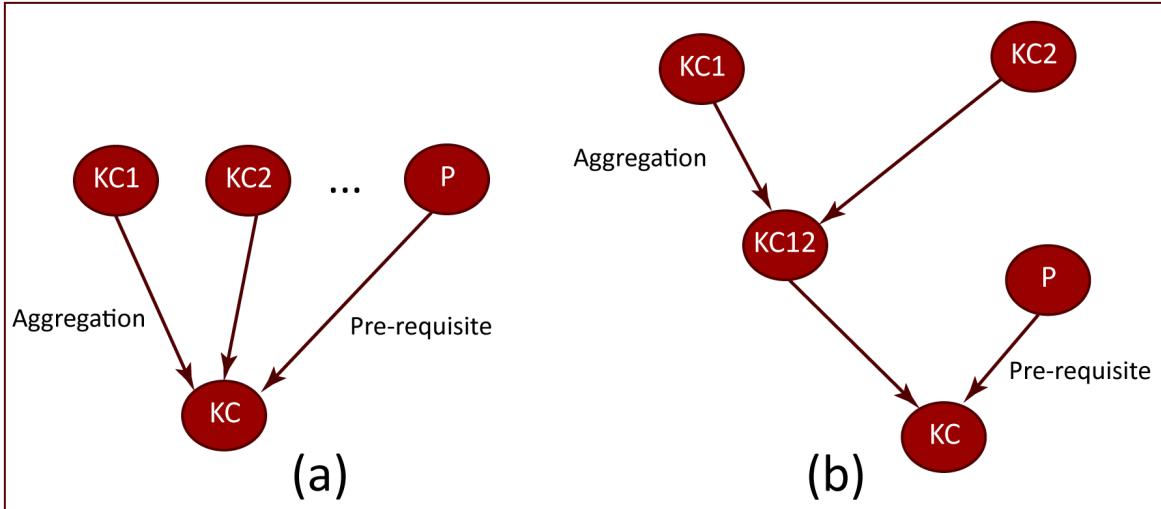


FIGURE 2.3 : Agrégation de modélisation de réseau bayésien et relations préalables simultanément

Millan et coll. discuté d'un problème courant dans la modélisation des apprenants, à savoir modéliser simultanément les relations de pré-requis et de granularité. Si les deux sont inclus dans le même modèle, les relations avec des interprétations différentes sont mélangées et il est alors difficile de construire et de comprendre le modèle. Par exemple, si une compétence KC composite est composée de deux sous-compétences, KC1 et KC2, il existe également une compétence P qui est un prérequis pour KC. Les probabilités conditionnelles de K données à ses parents sont difficiles à préciser (figure 1.3 (a)). Ils ont proposé une solution qui consiste à regrouper des variables de même type en introduisant des variables intermédiaires (figure 1.3 (b)) [27].

2.4 Découverte des prérequis

Nous modélisons les compétences comme des variables continues qui représentent le degré auquel un apprenant a maîtrisé ou est conscient d'une compétence particulière. Nous traitons les éléments comme des variables continues qui reflètent le degré auquel un apprenant a correctement terminé une tâche. En pratique, la mesure de l'achèvement des tâches est souvent une variable binaire avec des valeurs = correct / incorrect. Un élément binaire peut cependant être considéré comme une projection d'un élément continu, et les corrélations entre éléments continus idéalisés peuvent être estimées en calculant la matrice de corrélation tétrachorique parmi les éléments binaires mesurés [34].

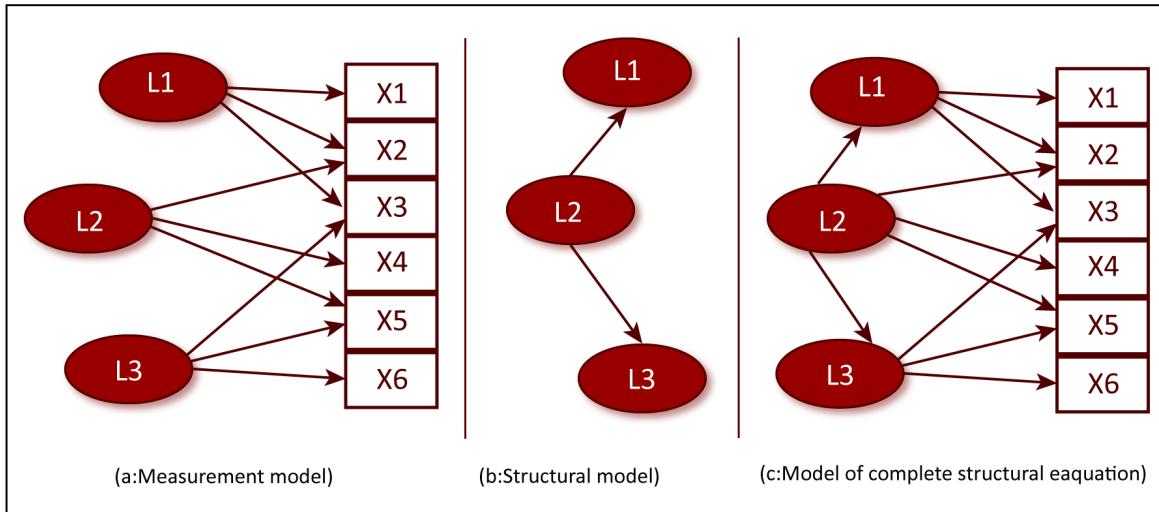


FIGURE 2.4 : Modèles d'équations structurelles

L'utilisation de la matrice Q consiste généralement à définir quels éléments « chargent » sur quelles compétences latentes. Nous pouvons définir un « modèle de mesure » qui relie les compétences latentes aux items mesurés 2.4. En modélisant les relations entre les compétences comme un modèle analytique causal du chemin entre les variables latentes 2.4, appelé « modèle structurel », nous pouvons combiner le « modèle de mesure » et le « modèle structurel » pour former un modèle d'équation structurelle linéaire complet 2.4 [35].

En supposant que le modèle de mesure est connu, nous pouvons rechercher le modèle structurel avec l'algorithme de découverte causale PC, dans lequel les entrées sont les relations d'indépendance et d'indépendance conditionnelle qui figurent parmi les variables latentes. Nous calculons ou testons les relations d'indépendance entre les latents en construisant un modèle structurel séparé et en l'ajustant aux données pour chaque test d'indépendance particulière requise. Notre méthode de construction de modèle produit un test cohérent et éprouvé de chaque relation d'indépendance conditionnelle [34].

2.5 Conclusion

Tout au long de ce chapitre, nous avons présenté les principaux concepts liés au modèle de l'apprenant dans l'environnement de l'apprentissage basé sur le Web, qui permet aux apprenants de travailler en collaboration sur un projet ou de coconstruire le sens des concepts en utilisant la technologie comme un tuyau pour le développement des connaissances partagées, aussi, la modélisation des compétences est importante pour découvrir les prérequis d'un apprenant en représentant le degré auquel il a maîtrisé.

CHAPITRE 3

L'approche items-to-skills mapping

Sommaire

3.1	Introduction	27
3.2	Connaissance(Knowledge)	27
3.3	Mappage des éléments aux compétences	29
3.4	Conclusion	34

3.1 Introduction

Les systèmes d'apprentissage personnalisé ont généralement un grand nombre d'items à résoudre par les apprenants qui sont des problèmes, questions, devoirs, etc. Avec un grand nombre d'items, il est utile de mesurer la similarité des éléments afin de les utiliser de manière efficace et de pouvoir y naviguer. Cette mesure de similarité est ensuite utilisée dans les systèmes d'apprentissage adaptatif et les systèmes de recommandations automatiques [36]. Elle permettra aussi de classer les items les plus similaires. Nous présentons donc les approches pour mapper les éléments aux compétences, les concepts de l'approche de similarité avec ses différentes catégories, et les différentes mesures de similarité utilisées.

3.2 Connaissance(Knowledge)

3.2.1 Définition

Le terme « connaissance » peut faire référence à une compréhension théorique ou pratique d'un sujet. Elle peut être implicite (comme avec une compétence ou une expertise pratique) ou explicite (comme avec la compréhension théorique d'un sujet) ; formel ou informel ; systématique ou particulier [37]. En éducation, la « connaissance » est constituée de faits de base alors que les composants de connaissance peuvent être des procédures, des schémas d'intégration, des stratégies de raisonnement complexes, des compétences métacognitives, etc. La "connaissance" en philosophie est une "vraie croyance justifiée" alors que notre utilisation des composants de la connaissance comprend à la fois une connaissance incorrecte (fausse) et une connaissance implicite (pas de croyance ou de justification explicite) [38].

3.2.2 Composante connaissance(Knowledge component)

Une composante de connaissance est une description d'une structure mentale ou d'un processus qu'un apprenant utilise, seul ou en combinaison avec d'autres éléments de connaissance, pour accomplir les étapes d'une tâche ou d'un problème [39]. Les items qui requièrent la même compétence pour différents apprenants constituent une composante de connaissance (KC) [40].

Composante de connaissance est un terme qui est utilisée comme concept, principe, fait ou compétence, et des termes de sciences cognitives comme schéma, règle de production, idée fausse ou facette [41]. La description de la composante de connaissance est présentée dans la figure 3.1.

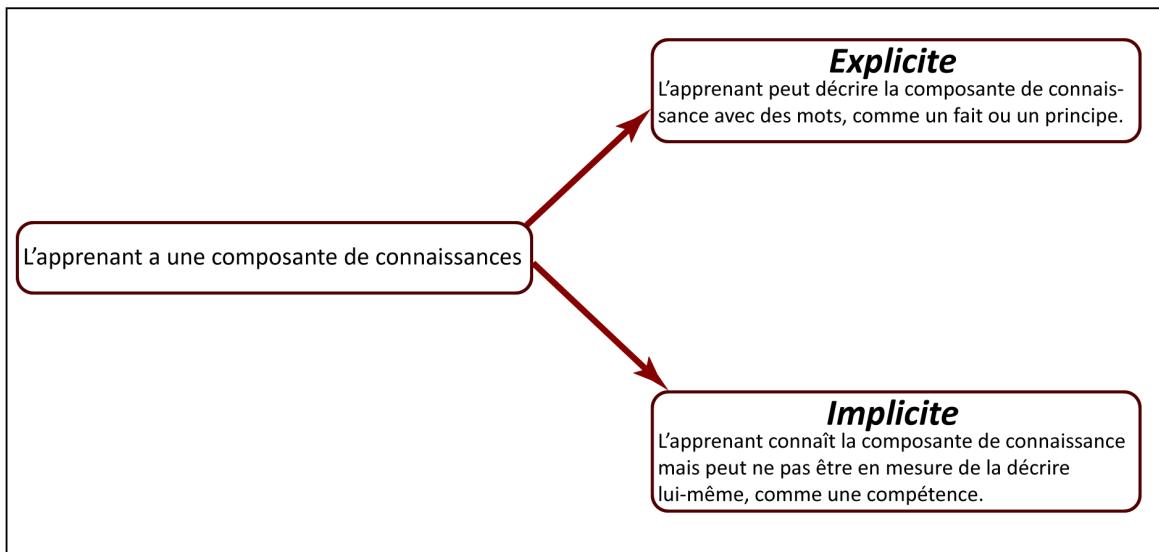


FIGURE 3.1 : Composante Connaissances

Exemple : "Si l'angle A = 60 et (A = B, C = 60) Alors triangle équilatéral."

L'élève peut dessiner un triangle équilatéral sans être capable de décrire la règle. Une grande partie de ce que les apprenants de première langue savent de leur langue maternelle implique des éléments de connaissances implicites. Pertinence : la plupart des composants de connaissance explicites impliquent de nombreux composants de connaissance tacites (compétences innées ou acquises, le savoir-faire et l'expérience). La réponse et la fonctionnalité sont liées par la composante de connaissance, où les deux peuvent être externes, dans le monde, comme des signaux dans un stimulus et une réponse motrice ou interne, dans l'esprit, comme des fonctionnalités inférées et un nouvel objectif[41].

3.2.3 Les types de composante de connaissance

La composante de connaissance est une représentation mentale de : [42]

- **Connaissance du domaine** : faits, concepts, principes, règles, procédures, stratégies.
- **Connaissances préalables** : connaissance de l'encodage des fonctionnalités.
- **Connaissance intégrative** : schémas ou procédures qui connectent d'autres KC.
- **Connaissances métacognitives** : sur les connaissances, le contrôle de l'utilisation ou l'acquisition des connaissances.
- **Croyances et intérêts** : ce que l'on aime, croit.

- **Toute représentation externe des connaissances :** (comme les descriptions de manuels ou un exemple) ou les structures cognitives génériques (mémoire de travail), soit, les paramètres continus sur les représentations des connaissances (force, niveau d'engagement, valeur implicite d'un objectif, affect) ne sont pas des composants de connaissances.

3.3 Mappage des éléments aux compétences

3.3.1 Définition

Les compétences (skills) déterminent le résultat d'un individu sur une tâche dans une période donnée, à savoir si le résultat obtenu par l'individu est un succès ou un échec. La tâche à laquelle un individu est évalué peut-être une question, un exercice, ou tout autre défi qui nécessitera certaines compétences. Les systèmes de tutorat intelligent cherchent donc à mapper ces compétences aux questionnaires ou (items) afin d'amener l'apprenant à obtenir un niveau de maîtrise sur un ensemble de compétences et aussi recommander les tâches ou « items » (ou exercices) les plus appropriés compte tenu de la dernière tâche effectuer par l'apprenant [43]. Ce « mapping » est aussi appelée Q-matrix en EDM [44].

3.3.2 items-to-skills mapping Structure

Il existe deux approches pour mapper les éléments aux compétences présentées dans la figure 3.2 :

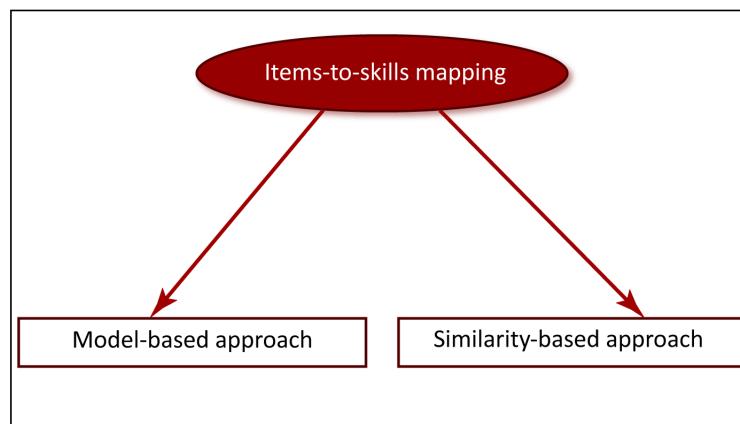


FIGURE 3.2 : La structure du mappage des éléments aux compétences

3.3.2.1 L'approche basée sur un modèle

3.3.2.1.1 Définition

Cette approche est un modèle simplifié qui explique les données observées. Basé sur une matrice de réponses des apprenants aux éléments (ou items), le modèle prédit les réponses de l'apprenant. Le modèle attribue plusieurs compétences latentes aux apprenants et utilise le mappage des éléments aux facteurs latents correspondants. Ce type de modèle peut souvent être exprimé naturellement à l'aide de la multiplication matricielle, c'est-à-dire que l'ajustement d'un modèle conduit à une factorisation matricielle. Une fois que nous obtenons le résultat après avoir ajusté le modèle aux données, nous avons désigné les items avec la même valeur d'un facteur latent comme « similaires ». Cette approche conduit naturellement à plusieurs composantes de connaissance par compétence [36]. L'approche basé sur le modèle présenté dans la figure 3.3 développe un modèle de notation des utilisateurs, les algorithmes de cette catégorie adoptent une approche probabiliste en calculant la valeur attendue de la prédiction de l'utilisateur et en tenant compte des notes de l'utilisateur sur d'autres éléments [45].

Cette approche peut donc être produite par différents algorithmes d'apprentissage automatique tels que le réseau bayésien, le clustering et l'approche basée sur des règles [46].

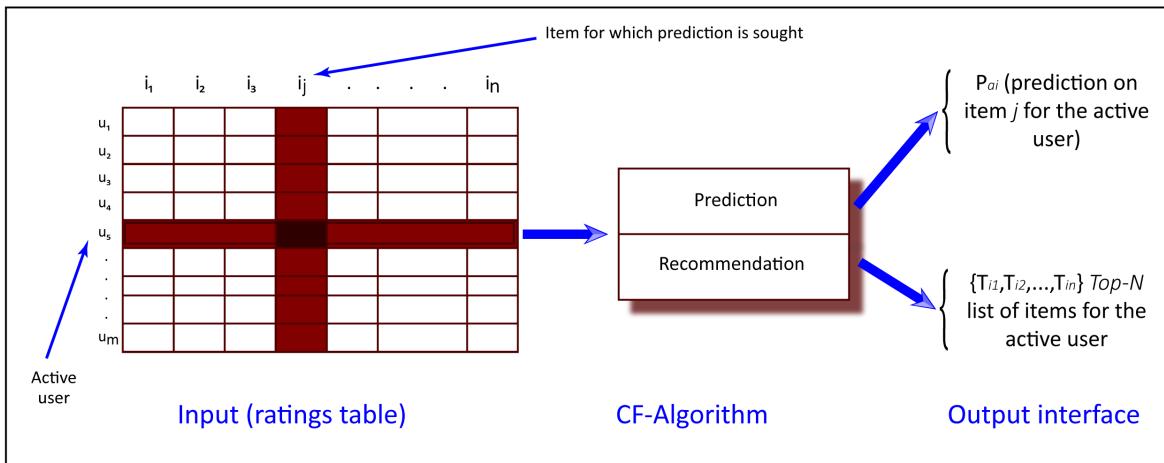


FIGURE 3.3 : L'approche basée sur un modèle

Cette approche produit des composantes de connaissances différents et multiples par compétence. Le modèle est généralement calculé à l'aide d'une technique d'optimisation qui ne mène qu'à des optima locaux (par exemple, une descente de gradient).

Dans les systèmes de recommandation, cette approche est utilisée pour la mise en œuvre du filtrage collaboratif ; elle est souvent appelée "Singular Value Decomposition" (SVD) [47].

3.3.2.1.2 Model Based Techniques

Dans le contexte pédagogique, de nombreuses variantes de l'approche basée sur les modèles ont été proposées :

- **Q-matrix** : Est une matrice binaire illustrée à la figure 3.4 montrant la relation entre les éléments de test et les attributs ou concepts latents ou sous-jacents. Les états de connaissance ont été attribués aux apprenants en fonction de leurs réponses aux tests et de la matrice q construite [48].

	Questions				
	Q1	Q2	Q3	Q4	
Concepts	Con 1	1	0	0	1
	Con 2	1	1	0	1
	Con 3	1	0	1	0
	Con 4	1	1	0	1
	Con 5	1	1	1	0

1 If a question Q_i is related to the concept con_j
0 If not.

FIGURE 3.4 : Q-matrix

3.3.3 Similarity-based approach

3.3.3.1 Définition

En statistiques et en mathématiques, la mesure de similarité ou la fonction de similarité est une fonction qui mesure la similarité entre deux éléments. Les valeurs de similarité entre les éléments sont quantifiées par l'observation de la performance des utilisateurs sur les éléments, qu'il s'agisse de négatif ou non [49]. La mesure de similarité est une étape très importante dans une analyse plus approfondie telle que le clustering des items [36].

Dans cette approche, on calcule directement une mesure de similarité pour chaque paire d'éléments. Ces mesures de similarités sont ensuite utilisées pour faire le clustering, pour faire la visualisation en projetant ces éléments dans un plan à 2 ou 3 dimensions, ou pour faire d'autre analyse en essayant de récupérer les éléments les plus similaires. Cette approche est illustrée à la figure 3.5.

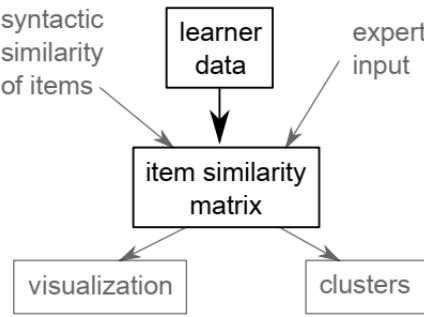


FIGURE 3.5 : Illustration de l'approche générale de l'analyse des éléments basée sur la similarité des éléments.

Dans l'apprentissage éducatif, la définition de la similarité des éléments a été analysée en utilisant la corrélation des réponses des apprenants et les temps de résolution des problèmes, ainsi qu'en utilisant les mauvaises réponses des apprenants. [36]

3.3.3.2 Processus de calcul de la similarité des items

Une étape importante de l'approche basée sur les items consiste à calculer le degré de similarité entre chaque paire d'items, puis à sélectionner les éléments les plus similaires. Il peut être décrit comme l'opération de calcul de similarité entre deux éléments i et j, où il s'agit d'abord de définir les utilisateurs qui ont évalué ces deux éléments, puis d'appliquer une technique de calcul de similarité pour déterminer le score de similarité entre i et j. [45] La figure 3.6 ci-dessous montre l'approche générale du calcul et de l'application de la similarité des items.

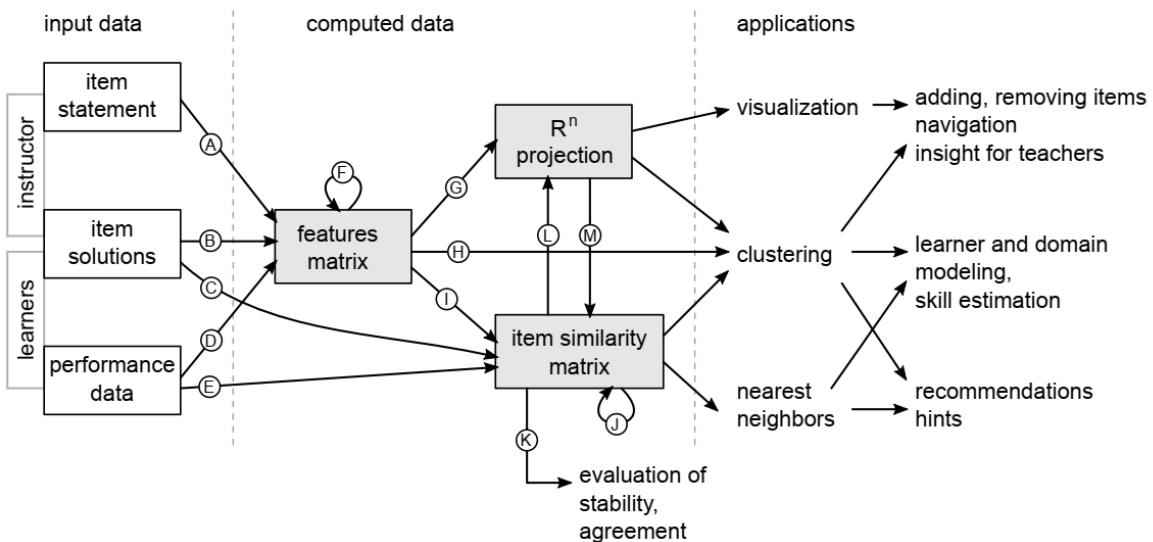


FIGURE 3.6 : L'approche générale du calcul et de l'application de la similarité des éléments.

Nous allons brièvement parcourir les trois étapes données d'entrée, données calculées, applications illustrées dans la figure 3.6.

3.3.3.2.1 Données d'entrée

- Item statement (Flèche A) : Un item est la spécification de la tâche que l'apprenant doit résoudre donnée en langage naturel, ou des exemples d'entrée-sortie, et la construction de la « features matrix » nécessite d'abord un prétraitement à des items.
- Item solution : est simplement la solution donnée par l'apprenant ou la solution commune des apprenants.
- Performance data : informations sur la performance des apprenants lors de la résolution des items par exemple : exactitude de la réponse, temps de réponse, nombre d'indices utilisés et nombre de tentative faite sur un seul item.

3.3.3.2.2 Données calculées

- Features matrix : c'est la matrice qui contient les éléments et leurs caractéristiques qui peuvent être par exemple une solution d'élément ou des mots-clés apparaissant dans une déclaration d'élément.
- Item similarity matrix : est une matrice bidimensionnelle m , où $m[i,j]$ désigne la similarité des items i et j . Le calcul de la matrice de similarité basée sur la matrice de caractéristiques (flèche I) ou sa projection dans R_n (flèche M) est une opération courante en apprentissage automatique, avec de nombreux choix disponibles, par exemple la similarité cosinus, le coefficient de corrélation de Pearson et la distance euclidienne (transformé en mesure de similarité par soustraction).

3.3.3.3 Mesures de similarité des items

Pour calculer le score de similarité entre deux items, il faut tout d'abord créer la matrice d'accord entre item i et item j . Cette matrice d'accord appeler aussi matrice de confusion ou (confusion matrix en anglais) est illustrer par le tableau 3.1.

	Item i (Correct)	Item i (Incorrect)
Item j (Correct)	a	b
Item j (Incorrect)	c	d

TABLE 3.1 : La matrice d'accord pour deux items.

Où,

- a : nombre d'apprenants qui ont répondu correctement à la fois à l'item i et j.
- b : nombre d'apprenants ayant répondu incorrectement à l'item i et l'item j correctement.
- c : nombre d'apprenants ayant répondu incorrectement à l'item j et l'item i correctement.
- d : nombre d'apprenants ayant répondu à la fois à l'item i et j de manière incorrecte.

Ensuite cette matrice d'accord est utilisée pour calculer la similarité entre l'item i et j. Les coefficients de calcule de similarité qui peuvent être utiliser sont les suivants 3.2 :

Mesures	Equation
Yule	$S_y = (ad - bc)/(ad + bc)$
Pearson	$Sp = (ad - bc)/\sqrt{(a + b)(a + c)(b + d)(c + d)}$
Cohen Kappa	$Sc = (Po - Pe)/(1 - Pe)$ $Po = (a + d)/n$ $Pe = ((a + b)(a + c) + (b + d)(c + d))/n^2$
Sokal	$Ss = (a + b)/(a + b + c + d)$
Jaccard	$Sj = a/(a + b + c)$
Ochiai	$So = a/\sqrt{(a + b)(a + c)}$

TABLE 3.2 : Coefficients de mesures de similarités.

3.4 Conclusion

Dans ce chapitre nous avons a introduit les concepts de base de la Composante connaissance, Mappage des éléments aux compétences, en spécifiant différents coefficients de similarité qui peuvent ensuite être utilisées dans une analyse plus approfondie des relations entre les items, telles que le clustering des items ou la visualisation de ces derniers.

CHAPITRE 4

Inférence bayésienne et la théorie de la réponse aux items

Sommaire

4.1	Introduction	36
4.2	Inférence bayésienne	37
4.3	La théorie des réponses aux items	48
4.4	Un ajustement bayésien des réponses aux items avec Stan	55
4.5	Conclusion	58

4.1 Introduction

Lors du 25e anniversaire de la Psychometric Society en 1961, Harold Gulliksen a décrit « le problème de la théorie des tests » comme « la relation entre la capacité de l’individu et son score au test ». Vingt-cinq ans plus tard, Charles Lewis a observé qu’ « une grande partie des progrès récents de la théorie des tests a été réalisée en traitant l’étude de la relation entre les réponses à un ensemble d’items de test et un trait hypothétique (ou traits latents) d’un individu comme un problème d’inférence statistique », apportant ainsi des solutions à des problèmes autrefois insolubles tels que l’adaptation des tests aux candidats individuels, le tri des relations dans les modèles de réussite dans les systèmes scolaires hiérarchiques [50] et l’estimation des paramètre des modèles IRT. Et les concepts et outils d’inférence statistique aider à expliquer les relations entre les preuves et l’inférence sur les connaissances, l’apprentissage et les réalisations des élèves que ce qui est traditionnellement associé à la théorie des tests standard et aux résultats standardisés [51].

L’inférence peut être défini comme un raisonnement à partir de ce que nous savons et de ce que nous observons jusqu’aux explications, conclusions ou prédictions parce que nous raisonnons toujours en présence d’incertitude et les informations avec lesquelles nous travaillons sont généralement incomplètes, peu concluantes, se prêtent à plus d’une explication. L’utilisation de l’inférence bayésienne permet donc d’estimer à partir d’observations les paramètres d’écrivant la distribution de probabilité en tenant compte des informations supplémentaire appelé aussi des informations apriori. Par exemple dans le cas où on cherche à estimer les paramètre θ_1 et θ_2 venant du problème où 30% des machines à sous ont une probabilité θ_1 de donner 1 euro, le reste a une probabilité θ_2 .

L’évaluation pédagogique concerne l’inférence sur les connaissances, les compétences et les réalisations des élèves, c’est-à-dire qu’il existe un intérêt pour l’étude des variables latentes [52], utilisé surtout dans La théorie de la réponse aux items (TRI). Parce que les données ne sont jamais aussi complètes et sans équivoque qu’elles garantissent la certitude, la théorie des tests a évolué en partie pour répondre aux questions de poids, de couverture et d’importance des données. Les concepts et techniques qui en résultent peuvent être considérés comme des applications de principes plus généraux d’inférence en présence d’incertitude. Comme le dit Glen Shafer « La probabilité n’est pas vraiment une question de nombres ; il s’agit de la structure du raisonnement. ».

Dans cette partie nous verrons l’inférence bayésienne appliquer à la théorie des réponses aux items et la plateforme Stan qui permet d’appliquer cette inférence à partir des données.

4.2 Inférence bayésienne

4.2.1 Définition

L'inférence bayésienne est une méthode d'apprentissage des valeurs des paramètres dans les modèles statistiques à partir de données. L'inférence bayésienne est une approche entièrement probabiliste, dont les résultats sont des distributions de probabilité. Une autre caractéristique distinctive de l'inférence bayésienne est l'utilisation d'informations a priori dans les analyses. L'inférence bayésienne ne concerne pas un modèle statistique particulier, en fait, une multitude de modèles statistiques et de machine learning (que ce soit : la régression linéaire, la régression logistique, les réseaux de neurones artificiels, ...) peuvent être « entraînés » de manière bayésienne ou non bayésienne. L'inférence bayésienne peut être résumée comme étant une méthode d'apprentissage à partir de données.

4.2.2 probabilité conditionnelle et théorème de bayes

Les probabilités conditionnelles jouent un rôle central dans le processus d'inférence sur le monde incertain. La probabilité conditionnelle est définie comme la probabilité qu'un événement ou un résultat se produise, sur la base de l'occurrence d'un événement ou d'un résultat précédent. La probabilité conditionnelle fait référence aux chances qu'un résultat se produise étant donné qu'un autre événement s'est également produit. Il est souvent indiqué comme la probabilité de B étant donné A et s'écrit $P(B|A)$, où la probabilité de B dépend de celle que A se produise. La formule de la probabilité conditionnelle est :

$$Pr(B|A) = \frac{Pr(A|B)}{Pr(A)} \quad (4.1)$$

Le théorème de Bayes , du nom du mathématicien britannique du XVIIIe siècle Thomas Bayes, est une formule mathématique permettant de déterminer la probabilité conditionnelle. Le théorème fournit un moyen de réviser les prédictions ou théories existantes (probabilités de mise à jour) compte tenu de preuves nouvelles ou supplémentaires. Dans de nombreux problèmes d'analyse de données, le théorème de bayes fourni la possibilité de déduire la valeur des paramètres d'un modèle à partir de données. Le théorème de bayes est défini par la formule suivante :

$$Pr(B|A) = \frac{Pr(A|B) * Pr(B)}{Pr(A)} \quad (4.2)$$

Ou $Pr(B|A)$ est la probabilité de l'évènement B sachant A. L'équation 4.2 peut être interpréter comme suit :

$$Pr(hypothesis|data) = \frac{Pr(data|hypothesis) * Pr(hypothesis)}{Pr(data)} \quad (4.3)$$

$Pr(hypothesis|data)$ est la probabilité d'une hypothèse qu'on cherche sachant les données. En appliquant le théorème de Bayes, $Pr(hypothesis|data)$ est le produit de la probabilité des données sachant l'hypothèse $Pr(data|hypothesis)$ qui peut être aussi un modèle et la probabilité de l'hypothèse $Pr(hypothesis)$ ensuite divisé par la probabilité des données $Pr(data)$. Cette hypothèse est généralement quelque chose d'inobservé ou inconnu qu'on cherche à obtenir à partir des données. Par exemple pour les modèles de régression linéaire, c'est les paramètres du modèle qu'on cherche à estimer. Le théorème de Bayes permet donc d'obtenir la probabilité de l'hypothèse sachant les données récolter.

4.2.3 L'approche Bayésienne

L'approche Bayésienne se base sur l'idée d'avoir certaines croyances préalables sur le système et puis mettre à jour ces croyances sur la base des données observées. C'est-à-dire avoir des connaissances a priori sur les paramètres inconnus. Ces connaissances prennent la forme d'une loi sur l'espace des paramètres qui s'appelle la loi a priori. L'approche bayésienne cherche donc à combiner d'une part l'information empirique obtenu sur les paramètres et l'interprétation des valeurs plus subjective (il y'a certaines idées sur les valeurs que les paramètres pourrait avoir). Cette procédure de mise à jour est basée sur le théorème de Bayes.

4.2.4 Théorème de Bayes dans la cadre de l'application de l'approche Bayésienne

En remplaçant hypothèse (hypothesis) par θ de l'équation 4.3 comme étant le paramètre à estimer on obtient donc :

$$Pr(\theta|data) = \frac{Pr(data|\theta) * Pr(\theta)}{Pr(data)} \quad (4.4)$$

Où,

- $Pr(\theta|data)$ (Posterior distribution) : est la distribution à posteriori du paramètre θ . C'est l'estimation de θ a posteriori (après avoir vu les données).
- $Pr(data|\theta)$ (Likelihood) : c'est la vraisemblance qui est capturée à partir des données sachant le modèle utiliser pour faire l'inférence.
- $Pr(\theta)$ (prior distribution) : c'est la distribution a priori sur le paramètre θ .

- $Pr(data)$: qui est la probabilité des données.

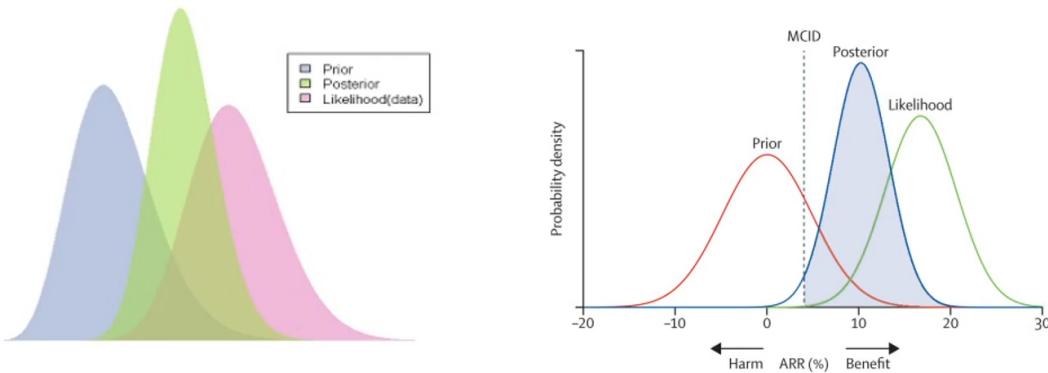


FIGURE 4.1 : La distribution à priori, à posteriori et celle de la fonction de vraisemblance.

4.2.4.1 Posterior distribution

La probabilité postérieure est la probabilité qu'un événement se produise après que toutes les informations sur le système étudier aient été prises en compte. Elle est étroitement liée à la probabilité antérieure, qui est la probabilité qu'un événement se produise avant la pris en compte de nouvelles preuves. Elle est comme un ajustement sur la probabilité antérieure.

$$\text{Posterior probability} = \text{prior probability} + \text{new evidence.} \quad (4.5)$$

$$\text{Posterior Distribution} = \text{Prior Distribution} + \text{Likelihood Function.}$$

Par exemple, les données historiques suggèrent qu'environ 60% des étudiants qui commencent l'université obtiendront leur diplôme dans les 6 ans. C'est la probabilité a priori. Cependant, ce chiffre est en réalité bien inférieur. Alors les preuves collecter peuvent suggérer que le chiffre réel est en fait plus proche de 50% ; c'est la probabilité postérieure.

La distribution à posteriori est un moyen de résumer les connaissances ou les informations sur les quantités incertaines en analyse bayésienne. C'est une combinaison de la distribution a priori et de la fonction de vraisemblance, qui indique les informations contenues dans les données observées (les « nouvelles preuves »).

4.2.4.2 Likelihood

Habituellement lorsque l'on parle de distribution de probabilité, on suppose que nous connaissons les valeurs des paramètres. Aussi appelé fonction de vraisemblance (ou plus simplement vraisemblance), c'est une fonction des paramètres d'un modèle statistique calculée à partir de données observées [53]. C'est-à-dire la probabilité d'observer les données sous un certain modèle. Par exemple, la probabilité des données sachant θ en utilisant l'équation 4.4. En

inférence bayésienne, cette vraisemblance peut être interprétée comme la densité de probabilité des données conditionnellement à une valeur des paramètres et comme une mesure de l'information apportée par les données sur la valeur des paramètres.

4.2.4.3 Prior distribution

La distribution à priori capture la connaissance ou l'information apriori. Elle permet de mettre en forme les connaissances ou les informations avant de faire l'expérimentation. Ces sont des distribution subjective sans tenir compte des données.

Ensuite Il y'a la probabilité des données, qui est défini par :

$$Pr(data) = \int L(data|\theta)Pr(\theta)d\theta. \quad (4.6)$$

Le calcule d'une intégrale avec un seul paramètres (intégrale a une dimension) est possible mais lorsqu'il y'a plus d'un paramètre, cette dernière devient une intégrale multiple qui est difficile voire impossible à calculer.

L'approche Bayésienne d'aujourd'hui utilise des méthodes de simulation MCMC (chaine de Markov Monte Carlo) pour passer outre ce problème de calcule d'intégrale multiple. Nous verrons dans la section suivantes les chaines de Markov de Monte Carlo et leurs méthodes.

4.2.4.4 Chaine de Markov Monte Carlo (MCMC)

Malheureusement, dans la plupart des cas pratiques, il est impossible d'obtenir une solution analytique pour la distribution a posteriori. Le dénominateur sous la forme continue du théorème de Bayes consiste en une intégrale sur un espace potentiellement de grande dimension. L'un des moyens de résoudre ce problème est d'obtenir des échantillons de la distribution a posteriori. Parmi les approches les plus utilisées pour surmonter ces difficultés, on trouve les méthodes du Monte Carlo par chaînes de Markov et l'inférence variationnelle (dont nous n'évoquerons pas). Les méthodes de Monte Carlo par chaîne de Markov (MCMC) proposent des schémas pour dessiner,tirer (draw en anglais qui est présent dans les objets fit des résultats d'échantillonages) une série d'échantillons corrélés qui convergent vers la distribution cible [54].

Les algorithmes MCMC consistent à simuler séquentiellement une seule chaîne de Markov dont la distribution limite est celle choisie (par exemple, le maximum de la fonction de vraisemblance fois une densité de probabilité a priori des paramètres). Plus précisément, une chaîne de Markov est une séquence de variables aléatoires telle que la valeur suivante ou état de la séquence dépend uniquement de l'état présent et non des états passés [54]. Alors,

une séquence de variables aléatoires est générée $\hat{q}_0, \hat{q}_1, \dots$ telle que l'état suivant \hat{q}_{t+1} avec $t \geq 0$ est distribué selon la probabilité de transition $T(\hat{q}_t \rightarrow \hat{q}_{t+1}) \stackrel{\text{def}}{=} P(\hat{q}_{t+1} | \hat{q}_t)$ [55] tout en visant la convergence de la chaîne, c'est-à-dire que la chaîne est ergodique (stationnaire, irréductible et non périodique).

Une chaîne de Markov est définie par les valeurs initiales des distributions marginales des paramètres et la probabilité de transition entre deux états de l'espace des paramètres : $T(\hat{q} \rightarrow \hat{q}')$, pour aller d'un état \hat{q} à l'autre état \hat{q}' . Les algorithmes MCMC sont généralement influencés lors de l'implémentation par le point de départ de la chaîne (ce qui conduit à une phase de « burn-in », comme l'illustre la figure 4.2), par le choix de la probabilité de transition, le taux de convergence, l'acceptance de l'algorithme, etc.

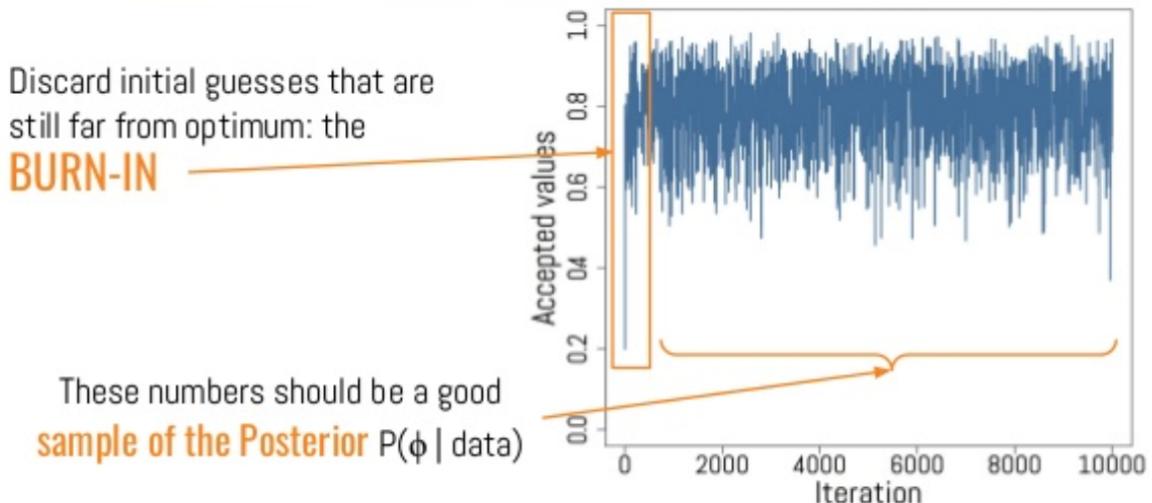


FIGURE 4.2 : La phase « burn-in » d'une chaîne de Markov et celle de la convergence de la chaîne.

Les algorithmes de Monte Carlo à chaînes de Markov (MCMC) sont donc des outils informatiques standard pour analyser les modèles complexes bayésiens [56] même si des fois des problèmes peuvent surgir dans leurs implémentations [57]. Parmi les méthodes MCMC, il y'a l'algorithme de Metropolis , l'algorithme de Metropolis–Hastings, échantillonnage de Gibbs, échantillonnage par tranches, Metropolis à tentatives multiples, Méthode de sur-relaxation l'hamiltonien Monte Carlo (HMC), No-U-Turn Sampler (NUTS), la méthode MCMC de Langevin et les méthodes basées sur le changement de dimension. Bien que notre travail ne soit pas centré sur les algorithmes MCMC, nous passerons en revue quelques définitions et théories qui justifient le fonctionnement des méthodes MCMC, et un bref résumé de quelques algorithmes MCMC.

4.2.4.4.1 Théorème de convergence des chaînes de Markov [1] [2] [3]

Une chaîne de Markov à temps discret est définie par une séquence de variables aléatoires X_0, X_1, X_2, \dots qui peut prendre des valeurs possibles dans l'espace d'état X avec une distribution initiale définie pour X_0 et des probabilités de transition définies comme :

$$p(x, A) = P(X_{n+1} \in A | X_n = x), \quad \forall A \subseteq X \quad (4.7)$$

Où la propriété de Markov est :

$$P(X_{n+1} \in A | X_0, X_1, \dots, X_n) = P(X_{n+1} \in A | X_n) \quad \forall A \subseteq X \quad (4.8)$$

Comme nous l'avons évoqué précédemment la probabilité que la chaîne passe à l'état suivant ne dépend que de l'état actuel et non des états précédents.

Définition 1 : Considérons une chaîne de Markov $\{X_i\}$ sur l'espace d'états X avec une probabilité de transition $P(x, .)$. Soit $\pi(.)$ une distribution de probabilité définie sur X . Alors π est une distribution stationnaire pour la chaîne de Markov $x, y \in X$:

$$\int_{x \in X} \pi(dx) P(x, dy) = \pi(dy) \quad (4.9)$$

Définition 2 : Une chaîne de Markov est ϕ -irréductible, s'il existe une mesure σ -finie non nulle sur X telle que :

$$\begin{aligned} \forall A : A \subseteq X \quad \text{avec} \quad \phi(A) > 0 \quad \& \quad \forall x \in X \\ \implies \exists n \in \mathbb{N} : P_n(x, A) > 0 \end{aligned} \quad (4.10)$$

Définition 3 : Une chaîne de Markov est apériodique, s'il n'y a pas de sous-ensembles disjoints non vides $X_1, \dots, X_d \subseteq X$ pour $d \geq 2$ tel que $P(x, X_{i+1}) = 1$ pour tout $x \in X_i$ ($1 \leq i \leq d-1$) et $P(x, X_1) = 1$ pour tout $x \in X_d$.

Théorème 1 : Considérons une chaîne de Markov apériodique et ϕ -irréductible définie sur un espace d'états X de distribution stationnaire π . Pour $\pi.a.e. x \in X$:

$$\lim_{x \rightarrow \infty} \|P^n(x, .) - \pi(.)\| = 0 \quad (4.11)$$

Soit $\lim_{x \rightarrow \infty} P^n(x, A) = \pi(A) \quad \forall A \subseteq X$.

Dans la section suivante, nous décrirons quelques algorithmes MCMC les plus courants qui respectent l'aspect stationnaire, irréductible et non périodique des chaînes de Markov.

4.2.4.4.2 Algorithme de Metropolis-Hastings

L'algorithme de Metropolis-Hastings, proposé par METROPOLIS [58] et généralisé par HASTINGS [59] est principalement utilisé comme moyen de simuler des observations à partir de distributions cibles. L'algorithme produit une chaîne de Markov dont la distribution limite des membres est la densité cible $\pi(x)$ et qui s'itère en « sautant » de l'état actuel x de l'espace des paramètres à l'état suivant y avec une probabilité d'acceptation suivante :

$$\alpha(x, y) = \min\left(1, \frac{\pi(y)q(y, x)}{\pi(x)q(x, y)}\right) \quad (4.12)$$

Où $q(x, y)$ est la probabilité conditionnelle de proposer l'état y étant donné l'état x . L'algorithme de Metropolis-Hastings est décrit dans l'algorithme 1.

Algorithm 1 Algorithme de Metropolis-Hastings [1]

```

1: Initial : value  $X_0$ , burn-in iterations  $B$ , number of samples  $M$ 
2: Output :  $S_1, \dots, S_M$  samples
3: for  $n$  in  $\{1, \dots, (B + M)\}$  do
4:   Draw  $Y_n \sim Q(X_{n-1},)$ , where  $Q$  is the proposal distribution with probability
   density function  $q$ .
5:   Calculate  $A_n = \frac{\pi(Y_n)q(Y_n, Y_{n-1})}{\pi(X_{n-1})q(X_{n-1}, Y_n)}$ 
6:   Draw  $U_n \sim \text{Uniform}[0,1]$ 
7:   if  $U_n < A_n$  then
8:      $X_n = Y_n$  (« accept »)
9:   else
10:     $X_n = X_{n-1}$  (« reject »)
11:   end if
12: end for
13: return  $S_1 = X_{B+1}, \dots, S_M = X_{B+M}$ .
```

4.2.4.4.3 Hamiltonian Monte Carlo

L'hamiltonien Monte Carlo (HMC) est une méthode de Monte Carlo à chaîne de Markov (MCMC) capable de traiter des distributions cibles de grande dimension et qui utilise les dérivées de la fonction de densité échantillonnée pour générer des transitions efficaces couvrant la partie postérieure [60] [61]. Il utilise une simulation de dynamique hamiltonienne approximative basée sur une intégration numérique qui est ensuite corrigée en effectuant une étape d'acceptation Metropolis. Bien qu'il s'agisse d'un cas particulier d'échantilleurs en temps continu, il peut être mis en œuvre en temps discret et est en fait à l'origine du succès du package Stan qui est utilisé dans la section implémentation 6.3.3. L'hamiltonien Monte Carlo améliore l'efficacité en utilisant le gradient du log postérieur pour diriger la chaîne de Markov vers les régions de densité postérieure plus élevée, où la plupart des échantillons

sont prélevés, par conséquent, une chaîne HMC bien réglée acceptera les propositions à un taux beaucoup plus élevé que l'algorithme MH traditionnel [62].

Le processus repose sur une variable auxiliaire $v \in \mathbb{R}^d$ qui augmente la distribution cible tout en échantillonnant à partir de π (une mesure cible de probabilité π sur $\Theta \subset \mathbb{R}_d$ par rapport à la mesure de Lebesgue, notée $\pi(\theta) \propto \exp\{-U(\theta)\}$, où U est une fonction continûment dérivable appelée fonction de potentiel [63]. L'hamiltonien est défini par :

$$H(\theta, v) = U(\theta) + \frac{1}{2}v^T M^{-1}v. \quad (4.13)$$

Et HMC se base sur la dynamique hamiltonienne pour générer des propositions pour θ :

$$\frac{d\theta}{dt} = \frac{\partial H}{\partial v} = M^{-1}v \text{ et } \frac{dv}{dt} = -\frac{\partial H}{\partial \theta} = -\nabla U(\theta) \quad (4.14)$$

L'algorithme HMC est :

Algorithm 2 Algorithme de Monte Carlo hamiltonien [63]

```

1: Initial : Starting point  $\theta_0$  , step size  $\epsilon$ , number of leapfrog steps  $L$ , covariance matrix
    $M$ , number of iterations  $N$ .
2: Output : sample  $\{\theta_0, \dots, \theta_N\}$  drawn from  $\pi(\theta)$ 
3: for  $n \leftarrow 1$  to  $N$  do
4:   draw  $v$  from  $\mathcal{N}(\cdot | 0, M)$ 
5:   compute  $\theta^*, v^* = \text{Leapfrog}(\theta_{n-1}, v, \epsilon, L)$  compute

$$\rho = 1 \wedge \exp\{H(\theta_{n-1}, v) - H(\theta^*, -v^*)\} \quad (4.15)$$

6:   set

$$(\theta_n, v_n) = \begin{cases} (\theta^*, -v^*), \text{with probability } \rho \\ (\theta_{n-1}, v), \text{otherwise.} \end{cases} \quad (4.16)$$

7: end for
8: return  $(\theta_0, \dots, \theta_N)$ 
9: function LEAPFROG( $\theta, v, \epsilon, L$ )
10:   compute  $(\theta_0, v_0) = (\theta, v, -\epsilon/2\nabla U(\theta))$ 
11:   for  $l \leftarrow 1$  to  $L$  do
12:     compute  $\theta_l = \theta_{l-1} + \epsilon M^{-1} v_{l-1}$ 
13:     compute  $v_l = v_{l-1} - \epsilon \nabla U(\theta_l)$ 
14:   end for
15:   return  $(\theta_l, v_l, +\epsilon/2\nabla U(\theta_l))$ 
16: end function

```

4.2.4.4.4 No-U-Turn Sampler

L'Hamiltonien Monte Carlo est un algorithme puissant, mais qui souffre de la nécessité d'ajuster le nombre d'étapes \mathcal{L} par exemple en utilisant des heuristiques basées sur des sta-

tistiques d'autocorrélation à partir d'essais préliminaires coûteux [60]. Ce besoin de fixer le paramètre \mathcal{L} est éliminé par l'algorithme No-U-Turn Sampler (NUTS).

Un échantillonneur MCMC qui utilise l'algorithme NUTS conserve la capacité de HMC à supprimer le comportement de marche aléatoire sans avoir besoin de définir le nombre \mathcal{L} d'étapes de saut que l'algorithme prend pour générer une proposition. Le critère utilisé se base sur le produit scalaire entre \tilde{r} (la quantité de mouvement actuelle) et $\tilde{\theta} - \theta$ (le vecteur de la position initiale à la position actuelle), qui est la dérivée par rapport au temps (dans le système hamiltonien) de la moitié de la distance au carré entre la position initiale θ et la position actuelle $\tilde{\theta}$ [64] :

$$\frac{d}{dt} \frac{(\tilde{\theta} - \theta)^T (\tilde{\theta} - \theta)}{2} = (\tilde{\theta} - \theta)^T \frac{d}{dt} (\tilde{\theta} - \theta) = (\tilde{\theta} - \theta)^T \tilde{r}. \quad (4.17)$$

Le pseudocode implémentant NUTS est fourni dans l'algorithme 3. L'échantillonneur par défaut (NUTS) de Stan est plus efficace pour explorer le postérieur [65] [64] et adapte automatiquement le nombre de pas de saut, éliminant le besoin de spécifier les paramètres de réglage par l'utilisateur.

Algorithm 3 No-U-Turn Sampler [65]

```

1: Resample  $r \sim \mathcal{N}(0, I)$ . ( $I$  denote the identity matrix.)
2: Resample  $u \sim \text{Uniform}(0, \exp\{\mathcal{L}(\theta^t) - \frac{1}{2}r^T r\})$ .
3: Initialize  $\theta^- = \theta^t, \theta^+ = \theta^t, r^- = r, r^+ = r, j = 0, \theta^{t+1} = \theta^t, n = 1, s = 1$ .
4: while  $s = 1$  do
5:   Chose a direction  $v_j \sim \text{Uniform}(\{-1, 1\})$ .
6:   if  $v_j = -1$  then
7:      $\{\theta^-, r^-, -, -, \theta', n', s'\} \leftarrow \text{Recurse}(\theta^-, r^-, u, v_j, j, \epsilon)$ .
8:   else
9:      $\{-, -, \theta^+, r^+, \theta', n', s'\} \leftarrow \text{Recurse}(\theta^+, r^+, u, v_j, j, \epsilon)$ .
10:  end if
11:  if  $s' = 1$  then
12:    With probability  $\min\{1, \frac{n'}{n}\}$ , set  $\theta^{t+1} \leftarrow \theta'$ .
13:  end if
14:   $n \leftarrow n + n'$ .
15:   $s \leftarrow s' \mathbb{I}[(\theta^+ - \theta^-)^T r^- \geq 0] \mathbb{I}[(\theta^+ - \theta^-)^T r^+ \geq 0]$ .
16:   $j \leftarrow j + 1$ .
17: end while
18: function RECURSE( $\theta, r, u, v, j, \epsilon$ )
19:   if  $j = 0$  then
20:     Base case take one leapfrog step in the direction v.
21:      $r' \leftarrow \frac{v\epsilon}{2} \nabla_{\theta} \mathcal{L}(\theta)$ .
22:      $\theta' \leftarrow v\epsilon r'$ .
23:      $r' \leftarrow \frac{v\epsilon}{2} \nabla_{\theta} \mathcal{L}(\theta')$ .
24:      $n' \leftarrow \mathbb{I}[u \leq \exp\{\mathcal{L}(\theta') - \frac{1}{2}r'^T r'\}]$ .
25:      $s' \leftarrow \mathbb{I}[\mathcal{L}(\theta') - \frac{1}{2}r'^T r' > u - \Delta_{\max}]$ 
26:     return  $\{\theta', r', \theta', r', \theta', n', s'\}$ .
27:   else
28:     Recursion implicitly build the left and right subtrees.
29:      $\{\theta^-, r^-, \theta^+, r^+, \theta', n', s'\} \leftarrow \text{Recurse}(\theta, r, u, v, j - 1, \epsilon)$ .
30:     if  $s' = 1$  then
31:       if  $v = -1$  then
32:          $\{\theta^-, r^-, -, -, \theta'', n'', s''\} \leftarrow \text{Recurse}(\theta^-, r^-, u, v, j - 1, \epsilon)$ .
33:       else
34:          $\{-, -, \theta^+, r^+, \theta'', n'', s''\} \leftarrow \text{Recurse}(\theta^+, r^+, u, v, j - 1, \epsilon)$ .
35:       end if
36:       With probability  $\frac{n''}{n'+n''}$ , set  $\theta' \leftarrow \theta''$ .
37:        $s \leftarrow s'' \mathbb{I}[(\theta^+ - \theta^-)^T r^- \geq 0] \mathbb{I}[(\theta^+ - \theta^-)^T r^+ \geq 0]$ .
38:     end if
39:     return  $\{\theta^-, r^-, \theta^+, r^+, \theta', n' + n'', s'\}$ .
40:   end if
41: end function

```

4.2.5 Méthodes de vérifications et de comparaison de modèles

Les méthodes de vérifications de modèles sont différentes des méthodes de comparaisons du fait qu'elles permettent d'examiner si un modèle capture les caractéristiques importantes

d'un ensemble de données, tandis que les méthodes de comparaisons répondent à la question de savoir quel modèle parmi un groupe de modèles candidats est le meilleur.

La théorie des réponses aux items (TRI) dans le cadre bayésien comme tout autres méthodes permettent une analyse plus approfondie en utilisant des distributions postérieures du modèle. En général, certaines techniques populaires de vérification de modèle bayésien qui ont été appliquées à l'IRT comprennent l'analyse résiduelle bayésienne, les vérifications prédictives préalables et les vérifications prédictives postérieures, et les méthodes de comparaison de modèles bayésiens incluent le critère d'information sur la déviance DIC [66], le facteur de Bayes [67] et la probabilité de validation croisée et le facteur de Bayes partiel [68].

Pour faire de la sélection de modèle (en comparant plusieurs modèles) il faut vérifier s'il y'a un effet des valeurs des paramètres sur la variable cible (par exemple sur la réussite ou pas à un item).

Dans le cadre fréquentiste, l'idée est de pénalisé les modèles qui ont trop de paramètres. Pour le faire il y'a AIC :

$$AIC = -2 \log(L(\widehat{\theta}_1, \dots, \widehat{\theta}_k)) + 2K. \quad (4.18)$$

Avec L la vraisemblance et k le nombre de paramètres θ , et $-2 \log(L(\widehat{\theta}_1, \dots, \widehat{\theta}_k))$ est la déviance qui mesure la qualité de l'ajustement du modèle de donnée. Plus il y'a de paramètres plus cette quantité est petite et mieux le modèle est ajuster aux données.

$2k$: c'est la pénalité, plus il y'a des paramètres plus cette quantité augmente. Cette pénalité est un équilibre entre l'ajustement du modèle des données et la complexité du modèle qui est capturer par le nombre de paramètres.

En bayésien, le critère d'information WAIC (Watanabe Akaike Information Criteria) et la validation croisée LOO (leave-one-out cross-validation) sont considéré comme des méthodes de sélection de modèles entièrement bayésiennes en raison de leur utilisation de l'ensemble de la distribution postérieure autre que les estimations ponctuelles.

WAIC [69] est une méthode de sélection de modèle entièrement bayésienne basé sur un critère d'information et est considéré comme une version améliorée de DIC, définit comme suit :

$$WAIC = -2 \sum_{i=1}^n \log E [p(y_i|\theta)] + 2p_{WAIC} \quad (4.19)$$

Où $E [p(y_i|\theta)]$ est la moyenne a posteriori de la vraisemblance de la i ème observation. P_{waic} est le nombre effectif de paramètres calculés en utilisant la variance postérieure de la vraisemblance. La valeur de $waic$ est la somme entre la deviance + le nombre de paramètre p_{waic} .

Une autre méthode de comparaison de modèles entièrement bayésienne est la validation croisée LOO [70] définit :

$$LOO = -2 \sum_{i=1}^n \log \int p(y_i|\theta)p_{post(-i)}(\theta)d\theta. \quad (4.20)$$

Où $p_{post(-i)}(\theta)$ est la distribution postérieure basée sur les données moins le point de données i .

4.3 La théorie des réponses aux items

La théorie des réponses aux items (TRI) aussi appelé théorie des traits latents, théorie du vrai score fort ou théorie moderne des tests mentaux est un paradigme pour la conception, l'analyse et la notation des test et des questionnaires [53]. TRI intervient dans la mesure ou la théorie classique n'apporte pas toujours des réponses satisfaisantes. Par exemple un item jugé facile ou difficile peut ne plus l'être dans un échantillon différent qu'il appartenait.

Dans cette situation, la théorie des réponses aux items tente de produire des propriétés de l'item qui soit indépendante d'un groupe particulier d'individus. En d'autres termes, il s'agit de parvenir à l'élaboration d'instruments de mesure dont les caractéristiques ne soient pas excessivement influencées par tel ou tel autre groupe de référence : ce qui, d'une certaine manière, conduit à définir des échelles qualifiées parfois d'"absolues" [71]. La première tentative de développement de ce type d'échelle remonte au début des années 1950 (échelle de Guttman). Initialement, ils reposaient sur un modèle totalement déterministe qui a ensuite été remplacé par un modèle plus réaliste de type probabiliste (modèle de Rasch). Ces modèles reposent sur l'hypothèse selon laquelle la réponse d'un individu à un item est déterminée ou peut être expliquer par deux facteurs qui sont :

- D'une part, certains attributs du sujet (sa compétence par exemple), qui, n'étant pas directement accessibles à l'observation et à la mesure, sont généralement qualifiés de traits latents [71] ;
- D'autre part, les propriétés de l'item lui-même, notamment, sa difficulté, son pouvoir

de discrimination, sans oublier le rôle que la "chance" (réponses "au hasard") peut jouer dans certains cas [71].

Sur le plan technique et mathématique, la théorie des réponses aux items utilise des modèles à un, deux ou trois paramètre(s), qui établissent la relation fondamentale entre le trait latent de l'individu (son niveau de compétence par exemple) et la probabilité pour cet individu de réussir un item donné. Cette relation est formalisée par une fonction (appelée fonction caractéristique de l'item), et peut être représentée géométriquement par une courbe (la courbe caractéristique de l'item). La forme la plus simple de cette fonction est celle qui repose sur le modèle de Rasch [71].

L'objectif de la méthode est double, il s'agit, d'une part, d'estimer les propriétés métriques des items (calcul des paramètres dits de difficulté, de discrimination et, éventuellement, de pseudo-chance) et, d'autre part, d'estimer le niveau de l'individu par rapport au trait latent considéré. Par ailleurs, ces estimations sont supposées indépendantes des échantillons particuliers (d'individus d'une part et d'items de l'autre) à partir desquels l'étude est réalisée [71].

4.3.1 IRT dans la recherche organisationnelle

Les modèles de la théorie des réponses aux items (TRI) présentent plusieurs avantages pour la recherche organisationnelle. L'un des avantages méthodologiques de l'IRT est la détection des réponses aberrante parmi d'autre avantage comme la construction et l'évaluation des échelles et l'examen des biais de mesure. Le tableau 4.1 présente un résumé de ces applications ainsi que quelques avantages par rapport aux approches alternatives.

**CHAPITRE 4. INFÉRENCE BAYÉSIENNE ET LA THÉORIE DE LA RÉPONSE
AUX ITEMS**

Problèmes méthodologiques	Applications de l'IRT	Avantages de l'IRT par rapport aux méthodes alternatives
Construction et évaluation de l'échelle	<ul style="list-style-type: none"> • Peut être utilisé pour raccourcir une échelle ou examiner la qualité des échelles existantes • Peut être utilisé pour développer d'autres types d'évaluations. 	<ul style="list-style-type: none"> • Contrairement au CTT, les paramètres des éléments IRT sont invariants d'un échantillon à l'autre. • IRT peut être utilisé pour créer des tests adaptatifs, mais pas CTT. • Bien que les méthodes CTT soient cohérentes avec les modèles de dominance, l'IRT peut être utilisé pour créer des mesures ponctuelles idéales.
Identifier l'élément (l'item) et tester le biais	<ul style="list-style-type: none"> • Peut différencier les différences moyennes observées du biais • Peut détecter les DIF et DTF compensatoires • Peut examiner les différences au niveau des options 	<ul style="list-style-type: none"> • La comparaison des différences moyennes avec le CTT confond le biais avec de vraies différences dans le trait latent. • Les modalités d'examen du DTF sont plus développées qu'en CFA.
Détection des réponses aberrantes	<ul style="list-style-type: none"> • Peut être utilisé pour détecter différents types de réponses aberrantes telles que IER, falsification et réponse faussement faible. 	<ul style="list-style-type: none"> • Les méthodes IRT peuvent détecter plusieurs types de réponses aberrantes tandis que d'autres méthodes ne peuvent détecter que des réponses imprudentes. • Les méthodes IRT sont souvent plus efficaces pour détecter les réponses aberrantes que les méthodes traditionnelles de réponse à l'effort insuffisant (IER).

TABLE 4.1 : Résumé des applications de l'IRT [5].

4.3.2 Les Modèles IRT

On distingue deux familles de modèle IRT : les modèles unidimensionnels qui nécessitent une seule dimension de trait et les modèles multidimensionnels qui sont supposées provenir de plusieurs traits. Cependant, en raison de la complexité considérablement accrue, la majorité des recherches et des applications IRT utilisent un modèle unidimensionnel.

Les modèles IRT se distingue par le nombre de paramètre qu'il utilise. Le modèle à paramètre unique (1PL) suppose tous les éléments qui correspondent au modèle ont la même difficulté, de sorte que ces éléments sont décrits par un seul paramètre. C'est le cas du modèle de Rasch. Le modèle dit à deux paramètre (2PL) fait appel au paramètre de difficulté et de discrimination. Par contre le modèle à trois paramètres (3PL) est celui qui cherche à estimer en plus le paramètre de difficulté et de discrimination, la pseudo chance.

Il en résulte des modèles à un paramètre ayant la propriété d'objectivité spécifique, ce qui signifie que le rang de la difficulté de l'item est le même pour tous les répondants indépendamment de la capacité, et que le rang de la capacité de la personne est le même pour les items indépendamment de la difficulté. Ainsi, les modèles à 1 paramètre (comme ceux de Rasch) sont indépendants de l'échantillon, une propriété qui n'est pas valable pour les modèles à deux et trois paramètres [53].

Le modèle général de Birnbaum, auquel le modèle de Rasch se rattache est plus complexe car il admet des courbes d'allures différentes pour chacun des items et suppose donc d'estimer plus d'un paramètre par item puisqu'il tient également compte du pouvoir discriminant de chacun d'eux. Le modèle proposé par Lazarsfeld est en quelque sorte une simplification du modèle général de Birnbaum puisqu'il postule des distributions linéaires de pentes variables [72].

Il existe aussi d'autre modèles IRT qui sont résumé dans le tableau 4.2.

**CHAPITRE 4. INFÉRENCE BAYÉSIENNE ET LA THÉORIE DE LA RÉPONSE
AUX ITEMS**

Modèle IRT	Description	Application
Modèle logistique multidimensionnel à deux paramètres (Multidimensional two-parameter logistic model M2PL)	Suit un processus de réponse de dominance et suppose que tous les items varient sur un vecteur de paramètres de discrimination ($a_i = a_1, \dots, a_m$) et de difficulté (b_i).	Utilisé pour les éléments dichotomiques où il est peu probable de deviner. Peut être utilisé lorsque les données sont multidimensionnelles.
Modèle de réponse graduée (Graded response model GRM)	Suit un processus de réponse de dominance et suppose que tous les items varient en fonction de la discrimination (a_i) et que chaque option de réponse varie en fonction de la difficulté (b_{ik}). En d'autres termes, chaque élément aura un paramètre a et $C - 1$ paramètres b , où C est le nombre total d'options de réponse.	Utilisé pour les éléments polytomiques avec trois options de réponse ou plus.
Modèle de réponse graduée multidimensionnelle (Multidimensional graded response model MGRM)	Suit un processus de réponse de dominance et suppose que tous les items varient sur un vecteur de paramètres de discrimination ($a_i = a_1, \dots, a_m$) et chaque option de réponse varie en fonction de la difficulté (b_{ik}). Chaque élément aura un vecteur d'un paramètre pour chaque facteur latent et des paramètres $C - 1$ b , où C est le nombre total d'options de réponse.	Utilisé pour les éléments polytomiques avec trois options de réponse ou plus. Peut être utilisé lorsque les données sont multidimensionnelles.
Modèle de dépliage gradué généralisé (Generalized graded unfolding model GGUM)	Suit un processus de réponse ponctuelle idéal et suppose que tous les éléments varient en fonction de la discrimination (a_i), de l'emplacement de l'élément (δ_i) et des seuils (τ_{i1}).	Peut être utilisé pour les éléments ponctuels idéaux dichotomiques ou polytomiques.

TABLE 4.2 : Résumé des quelques modèles IRT [5].

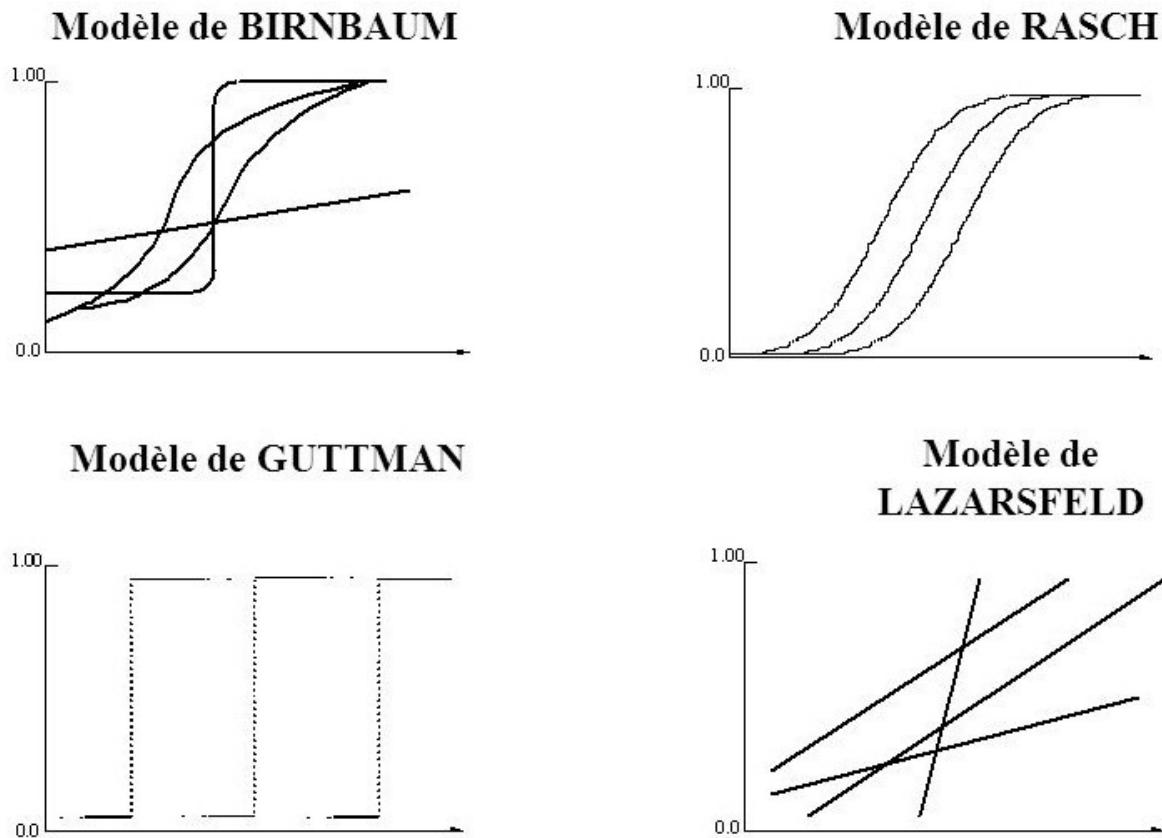


FIGURE 4.3 : Les modèles IRT.

4.3.2.1 Le modèle Rasch

La structure cumulative ou d'emboîtement du scalogramme de Guttman rend les analyses exploratoires traditionnelles mal appropriées (AFC par exemple). Dans cette situation Rasch a proposé en 1960 une représentation numérique des niveaux de compétences de sujets et de difficulté d'items. Le modèle de Rasch qui est le modèle à un paramètre (1PL), est considéré comme l'approche la plus simple pour modéliser la relation entre le trait latent et la probabilité de réussir correctement un item. Dans le modèle de Rasch chaque sujet est caractérisé par un niveau θ_i d'aptitude ($i = 1, \dots, N$) sur un continuum numérique latent. Le modèle de Rasch cherche à donner du sens non pas seulement au classement des sujets par compétences, non pas seulement au classement des items par difficultés mais le modèle cherche à donner du sens dans le fait que les coordonnées d'un sujet peuvent être directement et numériquement comparer à la difficulté d'un item pour qu'on puisse dire par exemple que ce sujet à la compétence nécessaire pour pouvoir résoudre l'item de tel difficulté. Donc ça devient possible de les comparer directement à partir du moment où on les projette sur une seule et même dimension (cet θ_i de sujet et β_j d'item). La probabilité de réussir l'item est une fonction croissante de la différence $\theta_i - \beta_j$ [72]. Rasch propose la fonction logistique

suivant [51] :

$$P(x_1, \dots, x_n | \theta, \beta_1, \dots, \beta_n) = \prod_{j=1}^n P(x_j | \theta, \beta_j), \quad \text{avec,} \quad (4.21)$$

$$P(x_j | \theta, \beta_j) = \frac{\exp [\theta - \beta_j]}{1 + \exp [\theta - \beta_j]}$$

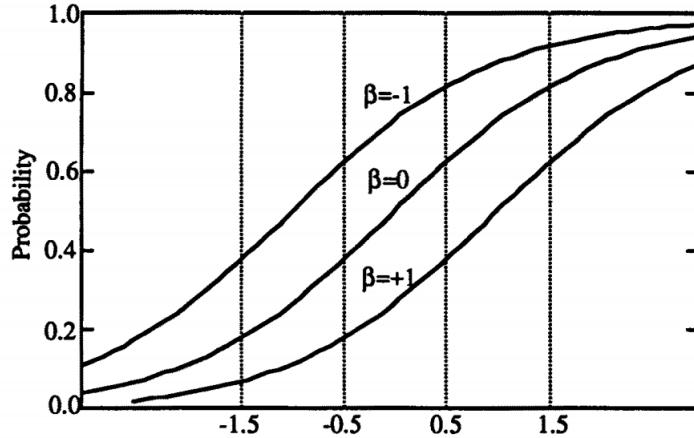


FIGURE 4.4 : Probabilité de bonne réponse, conditionnelle à θ , pour les items avec $\beta = -1, 0$ et 1 .

Où x_j est la réponse à l'élément j (1 pour correcte, 0 pour incorrecte). La figure 4.4 montre les probabilités de réponse correcte à trois éléments, avec les paramètres de difficulté $-1, 0$ et $+1$, en fonction de θ . Des valeurs faibles de θ indiquent des chances plus faibles de réponse correcte et des valeurs élevées indiquent des chances plus élevées [51].

4.3.2.2 Modèle logistique à deux paramètres

Le modèle de Rasch suppose que tous les items ont la même forme, mais cette hypothèse pourrait ne pas être raisonnable. Pour éviter cette hypothèse, le modèle logistique à deux paramètres inclut le paramètre de discrimination des items. Le paramètre de discrimination est une mesure de la capacité différentielle d'un élément. Une valeur élevée du paramètre de discrimination suggère un élément qui a une grande capacité à différencier les sujets. En pratique, une valeur élevée du paramètre de discrimination signifie que la probabilité d'une réponse correcte augmente plus rapidement à mesure que la capacité (trait latent) augmente. La probabilité d'une réponse correcte est donnée par la formule suivante.

$$P_i(\theta_j | X = 1) = \frac{\exp [\alpha_i(\theta_j - \beta_i)]}{1 + \exp [\alpha_i(\theta_j - \beta_i)]} \quad (4.22)$$

4.3.2.3 Modèle logistique à trois paramètres

Le modèle logistique à trois paramètres (3PL) a été proposé par Birnbaum en 1968 incluant le paramètre de pseudo-chance dans le modèle. Le modèle 3PL spécifie la probabilité de réponse correcte sur le i ème élément à l'aide de l'équation suivante.

$$P_i(\theta_j) = c_i + \frac{1 - c_i}{1 + \exp[-\alpha_i(\theta_j - \beta_i)]} \quad (4.23)$$

4.3.3 Estimation des paramètres

L'estimation des paramètres surtout dans un cadre bayésien est effectuée à l'aide des méthodes Markov Chain Monte Carlo implémentée dans plusieurs logiciels comme OpenBUGS [73], JAGS [74], Stan [75] etc. Les priors les plus utilisés dans les modèles pour l'estimation MCMC des paramètres du modèle IRT sont :

$$\begin{aligned} \theta_j &\sim Normal(0, 1), \quad j = 1, \dots, N \quad \text{ou} \\ \theta_j &\sim Uniform(-4, 4), \quad j = 1, \dots, N \end{aligned} \quad (4.24)$$

Et,

$$\begin{aligned} \beta_i &\sim Normal(0, 1), \quad i = 1, \dots, n \\ \alpha_i &\sim Normal(0, 1), \quad \text{et } \alpha_i > 0, \quad i = 1, \dots, n \\ c_i &\sim Beta(5, 17) \quad \text{et } 0 < c_i < 0.3, \quad i = 1, \dots, n \end{aligned} \quad (4.25)$$

4.4 Un ajustement bayésien des réponses aux items avec Stan

Stan est une plate-forme pour la modélisation statistique, l'analyse de données et la prédiction dans les sciences sociales, biologiques et physiques, l'ingénierie et les affaires. Il permet de spécifier les fonctions de densité de log afin d'obtenir :

- Inférence statistique bayésienne complète avec échantillonnage MCMC (NUTS, HMC)
- Inférence bayésienne approximative avec inférence variationnelle (ADVI)
- Estimation du maximum de vraisemblance pénalisée avec optimisation (L-BFGS)

La bibliothèque mathématique de Stan fournit des fonctions de probabilité différentiables et une algèbre linéaire (C++ autodiff). Stan peut être utilisé avec les langages d'analyse de données les plus populaires (R, Python, shell, MATLAB, Julia, Stata) et fonctionne sur toutes les principales plates-formes (Linux, Mac, Windows) [75].

4.4.1 IRT avec Stan

La théorie de l'item-réponse (TRI) modélise la situation dans laquelle un certain nombre d'étudiants répondent chacun à une ou plusieurs questions d'un groupe de questions de test. Le modèle est basé sur des paramètres pour la capacité des étudiants, la difficulté des questions, et dans des modèles plus articulés, le caractère discriminant des questions et la probabilité de deviner correctement [76].

4.4.1.1 Déclaration des données

Les données fournies pour un modèle IRT peuvent être déclarées comme suit pour tenir compte du fait que tous les étudiants ne sont pas tenus de répondre à toutes les questions. Ces données sont dans le bloc « data » où les informations pertinentes sur les données et les données elles-mêmes sont spécifier.

```

1  data {
2      // numbers of things
3      int<lower=1> N;    // number of observations
4      int<lower=1> I;    // items ,   number of questions
5      int<lower=1> S;    // subjects ,  number of users
6      // data
7      int<lower=1,upper=I> item[N];
8      int<lower=1,upper=S> subject[N];
9      int<lower=0,upper=1> grade[N];
10     // data for posterior prediction using new data also used for Cross-validation
11     int<lower=1> N_new;
12     int<lower=1,upper=I> item_new[N_new];
13     int<lower=1,upper=S> subject_new[N_new];
14 }
```

Dans cette déclaration il y'a N un entier positif qui est le nombre d'observation, où pour toute valeurs n de 1 à N $grade[n]$ est la réponse obtenue par l'étudiant $subject[n]$ à la question $item[n]$.

4.4.1.2 Les paramètres du modèle de Rasch, 2PL et 3PL

Les paramètres des modèles IRT sont :

```

1  parameters {
2      // parameters
```

```

3   real ability[S];           // alpha: ability of student
4   real difficulty[I];      // beta: difficulty of question
5   real delta;              // mean student ability
6   // end for Rasch model
7   vector<lower=0>[I] discrimination; // discrimination of question
8   // end for 2PL
9   vector<lower=0,upper=1>[I] guessing; // 
10  // end for 3PL
11 }
```

Le paramètre $ability_i$ est le coefficient de capacité de l’élève S_i , $difficulty_j$ le coefficient de difficulté de la question I_j , $discrimination_i$ qui modélise à quel point l’item k est discriminant et $guessing_i$ le paramètre de pseudo hasard. La paramétrisation non standard utilisée comprend également un terme d’interception delta, qui représente la réponse moyenne de l’étudiant à la question moyenne [76].

4.4.1.3 Le modèle de Rasch, 2PL et 3PL

Le modèle de Rasch :

```

1 model {
2   ability ~ normal(0,1);
3   difficulty ~ normal(0,1);
4   delta ~ normal(0.75,1);
5   for(n in 1:N)
6     grade[n] ~ bernoulli_logit(ability[subject[n]] - difficulty[item[n]] + delta);
7 }
```

Modèle logistique à deux paramètres :

```

1 model {
2   ability ~ normal(0,1);
3   difficulty ~ normal(0,1);
4   discrimination ~ lognormal(0,1);
5   delta ~ normal(0.75,1);
6   grade ~ bernoulli_logit(discrimination[item] .* (ability[subject] - (difficulty[item] +
7     delta)));
}
```

Modèle logistique à trois paramètres :

```

1 model {
2   ability ~ normal(0,1);
3   difficulty ~ normal(0,1);
4   discrimination ~ lognormal(0,1);
5   guessing ~ beta(5,17);
6   delta ~ normal(0.75,1);
7   grade ~ bernoulli(guessing[item] + ((1-guessing[item]).*(inv_logit(discrimination[item]
8     .* (ability[subject] - (difficulty[item] + delta))))));
}
```

Ces modèles utilisent la distribution de *Bernoulli* paramétrée en *logit*.

Si $\alpha \in \mathbb{R}$, alors pour $y \in \{0, 1\}$,

$$BernoulliLogit(y|\alpha) = BernoulliLogit(y|logit^{-1}(\alpha)) = \begin{cases} logit^{-1}(\alpha) & \text{si } y = 1, \text{ et} \\ 1 - logit^{-1}(\alpha) & \text{si } y = 0. \end{cases} \quad (4.26)$$

Et aussi la distribution de *Bernoulli* avec la fonction $inv_logit(x) = \frac{\exp(x)}{1 + \exp(x)}$ utilisée dans le modèle à trois paramètres.

4.4.1.4 Prédiction postérieure

Le bloc « generated quantities » est utilisé pour calculer de nouvelles variables et d'obtenir leur distributions postérieures correspondantes comme prédire des nouvelles valeurs où récupérer le log-vraisemblance, qui est utilisée pour calculer l'indice de l'ajustement du modèle à des fins de comparaison et de sélection de modèles. Pour le modèle de Rasch ce bloc est défini comme suit :

```

1 generated quantities {
2     int<lower=0,upper=1> y_pred[N];
3     int<lower=0,upper=1> yNew_pred[N_new];
4     vector[N] log_liks;
5
6     for(n in 1:N){
7         y_pred[n] = bernoulli_logit_rng(ability[subject[n]] - difficulty[item[n]] + delta);
8         log_liks[n] = bernoulli_logit_lpmf(grade[n] | ability[subject[n]] - difficulty[item[n]]
9             + delta);
10    }
11    for (n in 1:N_new) {
12        yNew_pred[n] = bernoulli_logit_rng(ability[subject_new[n]] - difficulty[item_new[n]] +
13            delta);
14    }
}

```

Dans l'implémentation en utilisant le workflow bayésien figure 6.2, les distributions des priors peuvent changer dans le but d'améliorer la précision du modèle.

4.5 Conclusion

L'inférence bayésienne est une méthode probabiliste qui prend en compte les informations apriori et cherche des résultats qui sont des distributions de probabilité. Sans oublier Les probabilités conditionnelles et le théorème de bayes qui jouent un rôle central dans le processus d'inférence, les méthodes de Monte Carlo par Chaine de Markov résolve une difficulté de calcule d'intégrale sur un espace de grande dimension causée par le dénominateur du théorème de bayes. En pratique les résultats des modèles utilisé dans la théorie des réponses au

CHAPITRE 4. INFÉRENCE BAYÉSIENNE ET LA THÉORIE DE LA RÉPONSE AUX ITEMS

items offre des possibilités d’analyser les modèles utilisés et de sélectionner le meilleur modèle. Après toute ces notions vues dans ce chapitre, le chapitre suivant évoquera le clustering hard et soft.

CHAPITRE 5

Clustering Hard et Soft

Sommaire

5.1	Introduction	61
5.2	Les métriques	62
5.3	Critère de liaison	62
5.4	Le clustering hiérarchique	65
5.5	Le clustering partitionnel	69
5.6	Fuzzy clustering	72
5.7	Les autres méthodes de clustering	75
5.8	La différence entre le clustering hiérarchique, partitionnel et Fuzzy clustering	75
5.9	Indice de validité du clustering	76
5.10	Conclusion	78

5.1 Introduction

Le clustering est un moyen de classer les données brutes de manière raisonnable et de rechercher les modèles cachés qui peuvent exister dans les ensembles de données [77]. Le clustering est une méthode de l'apprentissage automatique non-supervisé et un outil mathématique qui tente de découvrir des structures ou certains modèles dans un jeu de données, où les objets à l'intérieur de chaque cluster montrent un certain degré de similitude [78]. Le but de l'analyse de cluster consiste à partitionner un ensemble de N objets en clusters de sorte que les objets dans le cluster doivent être similaires les uns aux autres et les objets dans différents groupes doivent être différents avec les uns des autres [79]. Il peut être réalisé par divers algorithmes qui diffèrent considérablement dans leur notion de ce qui constitue un cluster et comment les trouvés efficacement. L'analyse de cluster est un processus itératif de découverte de connaissances ou d'optimisation multi-objectifs interactive [78]. Les chercheurs ont mis au point de nombreux algorithmes de clustering, qui ont été largement appliqués. Ils peuvent être globalement classés en deux groupes : partitionnelle et hiérarchique. Les algorithmes de partition traitent les données d'entrée et créent une partition qui regroupe les données en clusters. En revanche, les algorithmes hiérarchiques créent un ensemble de partitions imbriquées appelé hiérarchie de cluster [80]. En général, un algorithme de clustering hiérarchique partitionne un ensemble de données en différents clusters via une agglomération ou une approche divisive basée sur un dendrogramme [81]. Les concepts du clustering incluent des groupes avec de petites distances entre les membres du cluster, des zones denses, des intervalles ou des distributions statiques particulières.

Le clustering est donc une méthode qui cherche à minimiser l'inertie intra-cluster et maximiser l'inertie inter-cluster. Pour minimiser l'inertie intra-cluster et maximiser l'inertie inter-cluster, les algorithmes de clustering utilisent des métriques pour mesurer la distance entre les clusters et les critères de liaison qui sont la façon dont la distance entre les clusters est calculée.

Avant de rentrer directement sur le fonctionnement des algorithmes de clustering, nous verrons d'abord les métriques et les critères de liaison qui sont utilisés dans le processus du clustering ensuite le clustering hiérarchique, partitionnel et l'indice de validité du clustering.

5.2 Les métriques

Le choix d'une métrique appropriée influencera la forme des grappes, car certains éléments peuvent être relativement plus proches les uns des autres sous une métrique que dans une autre. Par exemple, en deux dimensions, sous la métrique de distance Manhattan, la distance entre l'origine (0,0) et (.5, .5) est la même que la distance entre l'origine et (0, 1), tandis que sous le distance euclidienne métrique, cette dernière est strictement supérieure.

Certaines métriques couramment utilisées pour le clustering hiérarchique sont :

Names	Formula
Euclidean	$\ a - b\ _2 = \sqrt{\sum_i (a_i - b_i)^2}$
Squared Euclidean distance	$\ a - b\ _2^2 = \sum_i (a_i - b_i)^2$
Manhattan distance	$\ a - b\ _1 = \sum_i a_i - b_i $
Maximum distance	$\ a - b\ _\infty = \max_i a_i - b_i $
Mahalanobis distance	$\sqrt{(a - b)^T S^{-1} (a - b)}$ where S is the Covariance matrix

TABLE 5.1 : Les métriques

5.3 Critère de liaison

Le critère de liaison détermine la distance entre les ensembles d'observations en fonction des distances par paires entre observations. Certains critères de liaison couramment utilisés entre deux ensembles d'observations A et B sont : [5.2](#)

Names	Formula
Maximum or complete-linkage clustering	$\max\{d(a, b) : a \in A, b \in B\}$
Minimum or single-linkage clustering	$\min\{d(a, b) : a \in A, b \in B\}$
Unweighted average linkage clustering (or UPGMA)	$\frac{1}{ A \cdot B } \sum_{a \in A} \sum_{b \in B} d(a, b)$
Weighted average linkage clustering (or WPGMA)	$d(i \cup j, k) = \frac{d(i, k) + d(j, k)}{2}$
Centroid linkage clustering, or UPGMC	$\ c_s - c_t\ $ where c_s and c_t are the centroids of clusters s and t , respectively.
Minimum energy clustering	$\frac{2}{nm} \sum_{i,j=1}^{n,m} \ a_i - b_j\ ^2 - \frac{1}{n^2} \sum_{i,j=1}^n \ a_i - a_j\ ^2 - \frac{1}{m^2} \sum_{i,j=1}^m \ b_i - b_j\ ^2$

TABLE 5.2 : Les critères de liaison

Puisque le clustering est une méthode qui cherche à minimiser l'inertie au sein des clusters et à maximiser l'inertie entre les clusters, il existe donc deux types de distance, celle entre les objets des différents clusters qui est la distance inter-cluster et celle entre les objets du même cluster qui est la distance intra-cluster. Comme le montre la figure 5.1.

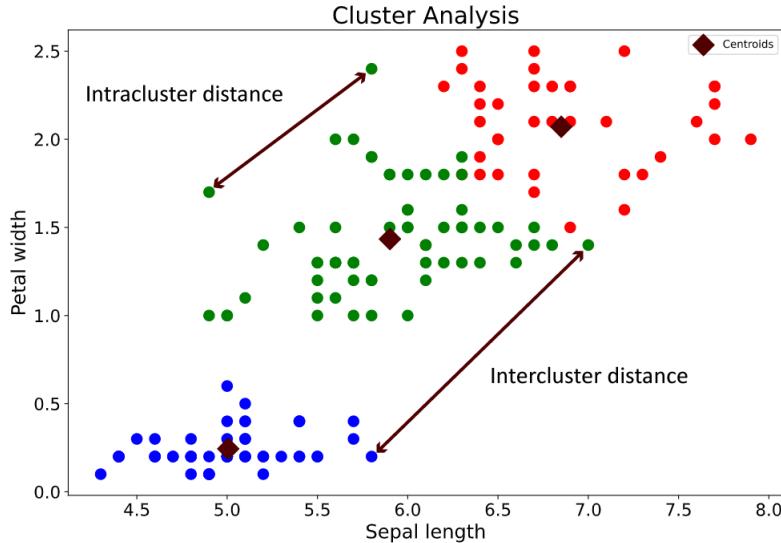


FIGURE 5.1 : Distances intracluster et intercluster.

5.3.1 Distance inter-cluster

La distance inter-cluster est la distance entre deux objets appartenant à deux clusters différents. Il y'a 5 types de distance inter-cluster :

5.3.1.1 Distance de liaison unique

La distance de liaison unique est la distance la plus proche entre deux objets appartenant à deux clusters différents. Le clustering de liaison unique est basé sur le critère de connectivité.

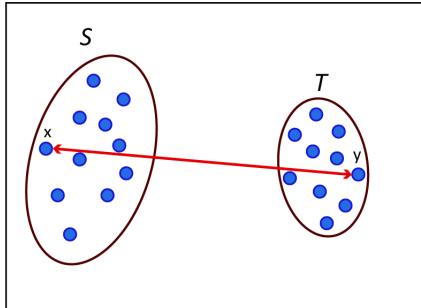
La distances entre deux clusters S et T est : [82]

$$\delta(S, T) = \min \left\{ d(x, y) \mid x \in S, y \in T \right\} \quad (5.1)$$

Le principal avantage de la liaison simple est qu'elle peut gérer des formes non elliptiques [83]. Cependant, il est sensible au bruit et aux valeurs aberrantes [84] [85].

5.3.1.2 Distance de liaison complète

La distance de liaison complète est la distance entre deux objets les plus éloignés appartenant à deux clusters différents. La liaison complète est la plus grande distance entre les points de données dans deux clusters S et T définie par la formule suivante :



$$\delta(S, T) = \max \left\{ d(x, y) \mid x \in S, y \in T \right\} \quad (5.2)$$

Cette distance ne tient pas compte de la structure du cluster et ne peut pas détecter les amas non sphériques [83].

5.3.1.3 Distance de liaison moyenne

La distance de liaison moyenne est la distance moyenne entre tous les objets appartenant à deux clusters différents. Le clustering de liaison moyenne a une procédure similaire que celle à liaison unique sauf le calcul de la distance entre deux clusters. Il utilise la moyenne entre les points de deux clusters S et T comme suit [83] :

$$\delta(S, T) = \frac{1}{|S||T|} \sum_{x \in S, y \in T} d(x, y) \quad (5.3)$$

Le clustering utilisant la liaison moyenne est moins sensible au bruit et aux valeurs aberrantes. Le seul inconvénient est sa polarisation vers les amas globulaires [85].

5.3.1.4 Centroïde Linkage Distance

La distance de liaison centroïde est la distance entre les centres v_s et v_t de deux clusters S et T respectivement, définie comme :

$$\begin{aligned} \delta(S, T) &= d(v_s, v_t) \text{ où} \\ v_s &= \frac{1}{|S|} \sum_{x \in S} x, v_t = \frac{1}{|T|} \sum_{y \in T} y \end{aligned} \quad (5.4)$$

5.3.1.5 Distance de liaison moyenne du centre de gravité

La distance de liaison moyenne du centre de gravité est la distance entre le centre d'un cluster et tous les objets appartenant à un autre cluster, définie comme :

$$\delta(S, T) = \frac{1}{|S| + |T|} \left\{ \sum_{x \in S} d(x, vt) + \sum_{y \in T} d(y, vs) \right\} \quad (5.5)$$

5.3.2 Distance intra-cluster

La distance intra-cluster est la distance entre deux objets appartenant au même cluster. Il y'en a trois types :

5.3.2.1 Distance de diamètre complet

La distance de diamètre complet est la distance entre deux objets les plus éloignés appartenant au même cluster défini comme :

$$\Delta(S) = \max_{x,y \in S} \{d(x, y)\} \quad (5.6)$$

5.3.2.2 Distance de diamètre moyen

La distance de diamètre moyen est la distance moyenne entre tous les objets appartenant au même cluster défini comme :

$$\Delta(S) = \frac{1}{|S| \cdot (|S| - 1)} \sum_{x,y \in S; x \neq y} \{d(x, y)\} \quad (5.7)$$

5.3.2.3 Diamètre barycentre Distance

La distance de diamètre barycentre est à double distance moyenne entre tous les objets et le centre de groupe de s défini comme :

$$\Delta(S) = 2 \left\{ \frac{\sum_{x \in S} d(x, \bar{v})}{|S|} \right\} \quad \text{où } \bar{v} = \frac{1}{|S|} \sum_{x \in S} x \quad (5.8)$$

5.4 Le clustering hiérarchique

5.4.1 Introduction

Dès les années 1970, on a estimé qu'environ 75% de tous les travaux publiés sur le clustering utilisaient des algorithmes hiérarchiques [86]. Le clustering hiérarchique est une méthode de clustering qui permet d'identifier des clusters dans un ensemble de données. Contrairement aux autres méthodes de clustering comme le k-means, le clustering hiérarchique n'oblige pas à spécifier à l'avance le nombre de clusters à générer. Et le résultat est une représentation arborescente des observations, appelée dendrogramme. L'interprétation des informa-

tions contenues dans un dendrogramme est souvent d'un ou plusieurs types : relations d'inclusion d'ensemble, partition des ensembles d'objets et clusters significatifs [86]. C'est une méthode de clustering d'apprentissage automatique non supervisé qui procède à une décomposition de données en cherchant à construire une hiérarchie de clusters. Cette méthode suit deux approches basées sur la direction du progrès, c'est-à-dire s'il s'agit du flux descendant ou ascendant de création de clusters. Il s'agit respectivement de l'Approche Divisive et de l'Approche Agglomérative.

Le clustering agglomératif commence à partir des clusters singleton et obtient une hiérarchie en fusionnant successivement les clusters, tandis que le clustering divisif commence par un cluster unique contenant tous les points et se poursuit en fractionnant les clusters de manière itérative [87]. Le clustering hiérarchique crée donc une décomposition hiérarchique de l'ensemble de données.

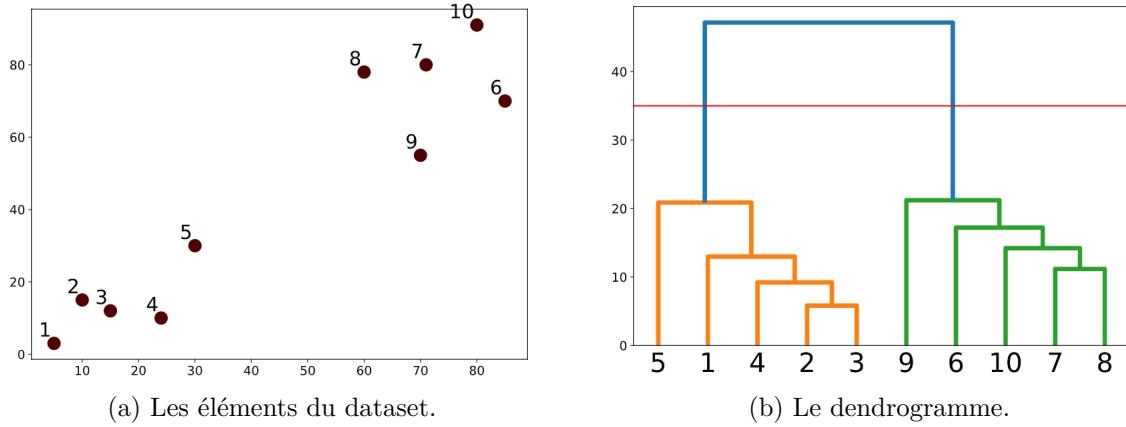
5.4.2 Clustering hiérarchique agglomératif

En clustering, l'un des algorithmes les plus largement utilisés est les algorithmes agglomératifs. Il s'agit d'une approche « ascendante » : chaque observation commence dans son propre cluster, et les paires de clusters sont fusionnées au fur et à mesure que l'on monte dans la hiérarchie. Les résultats de la classification hiérarchique sont généralement présentés dans un dendrogramme. Cependant, pour certains cas particuliers, des méthodes agglomératives efficaces optimales (de complexité) sont connues : SLINK pour une liaison simple et CLINK pour un clustering à liaison complète [78]. L'algorithme de clustering AHC a trois mesures de distance principales : liaison unique, liaison complète et liaison moyenne. Il est également appelé l'algorithme de clustering du plus proche voisin lorsque le critère single-linkage (liaison unique) est utilisée pour mesurer la distance entre les clusters [80]. Dans le clustering hiérarchique agglomératif classique, une paire de clusters à fusionner à la distance minimale inter-cluster [88].

5.4.2.1 Exemple de clustering agglomératif

Par exemple, supposons que les données de la figure 5.2a doivent être regroupées et que la métrique de distance est la distance euclidienne. Le dendrogramme obtenu après le clustering serait comme celui de la figure 5.2b

Et pour obtenir le nombre de clusters optimal avec le dendrogramme, la méthode la plus connue est celle qui cherche la distance verticale la plus élevée qui ne croise aucun cluster et les lignes verticales franchissent le seuil représente le nombre optimal de clusters. En utilisant le dendrogramme précédent on obtient 2 clusters comme le montre la figure 5.2b.



5.4.3 Clustering hiérarchique divisive

Le principe de base du clustering divisif a été publié sous le nom d’algorithme DIANA (Divisive Analysis Clustering) [89]. Contrairement au clustering hiérarchique agglomératif, le clustering hiérarchique divisif suit une approche descendante dans laquelle tous les points de données appartiennent à un seul grand cluster et celui-ci est divisé en groupes plus petits en fonction d’une logique de terminaison ou d’un point au-delà duquel il n’y aura plus de division de points de données. Le principe du clustering divisif est illustrer par la figure ci-dessous.

Le fractionnement est effectué de manière récursive au fur et à mesure que l’on descend dans la hiérarchie et il existe $2^{n-1} - 1$ [90] façons de diviser un ensemble de n objets en deux sous-ensembles [91]. Par conséquent le fractionnement prend top du temps sur la recherche de toutes les bipartitions possibles. Le clustering hiérarchique divisif procède par le fractionnement des clusters en deux clusters les moins similaires en utilisant des critères de type distance ou rapport et ensuite détermine les niveaux des noeuds (qui représente les groupes obtenus dans le dendrogramme).

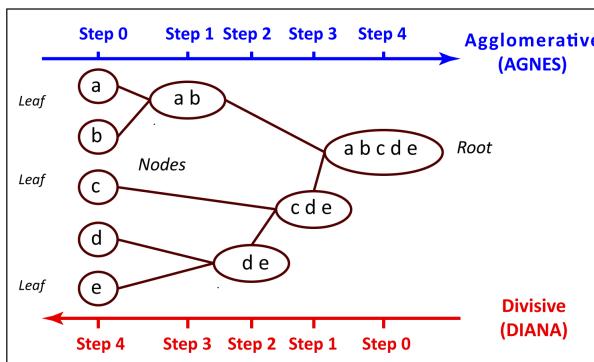


FIGURE 5.2 : La direction du clustering hiérarchique agglomérative et divisive.

5.4.4 Procédures de fractionnement des clusters

Un certain nombre de procédures de fractionnement ont été conçus dans le passé. La procédure de Williams Et Lambert (1959) [92] est dite monothéique dans le sens où les ensembles d'objets sont divisé en fonction des valeurs d'une seule variable. Cette idée a été mise à jour en utilisant une composante principale au lieu d'une seule variable (algorithme Principal Directions Divisive Partitioning ou PDDP par Boley 1997) [91].

Une autre approche pour contourner la complexité du fractionnement consiste à extraire un ou plusieurs objets, de l'ensemble à fractionner. Macnaughton-Smith et al. (1964) [93] ont proposé de sélectionner l'objet le plus distant comme center d'un nouveau cluster éloigné, puis les objets (les éléments du data set) les plus proche s'agrègent vers ce nouveau cluster [91].

Une idée similaire a été développée par Hubert (1973) [94], qui suggéré d'utiliser une paire d'objets comme point d'origine de la nouvelle bipartition. Son choix a été de sélectionner les deux objets qui sont les plus dissemblables, puis de construire les deux sous-clusters en fonction des distances à ces points d'origines [91].

Exploitant cette idée, Roux [95] [96] considérait les bipartitions générées par toutes les paires d'objets, en conservant la bipartition avec la meilleure évaluation de certains critères a priori [91].

5.4.5 Evaluations des bipartitions

Quel que soit le critère retenu, une série de très bonnes bipartitions ne se traduit pas automatiquement par une bonne hiérarchie. Les critères qui permettent d'évaluer les bipartitions sont de deux types : critère de type distance et les critères de type rapport. L'idée est donc de prendre en compte non seulement les dissemblances entre les clusters, mais aussi les dissemblances avec les éléments de deux clusters. Ces critères qui sont appelé aussi indice de validité sont énoncé dans le tableau 5.4.

5.4.6 Déterminations de niveaux de nœuds

Pour l'algorithme agglomératif, la valeur de l'indice de validité obtenu pour l'estimation du nombre optimale de clusters devient le niveau du nœud correspondant, et le dendrogramme ne montre aucun croisement (ou inversion) des branches. Malheureusement, les procédures qui divisent, en général, ne bénéficient pas de cette propriété, en raison du non-optimalité des fractionnements successifs. Une règle est alors nécessaire pour obtenir des niveaux cohérents

et une véritable représentation arborescente [91].

5.4.7 Conclusion

Le clustering hiérarchique dans la recherche d'informations identifie donc des groupements ou regroupements des « objets » étudiés qui représentent le mieux certaines relations de similitude. Il permet ainsi d'obtenir des résultats dans une hiérarchie de cluster appelé dendrogramme. Et si le dendrogramme ne présente aucun croisement des branches, la visualisation de ce dernier permet de voir comment les différents sous-clusters sont liés les uns aux autres et à quelle distance sont les points de données.

5.5 Le clustering partitionnel

5.5.1 K-means clustering

K-means est un algorithme de clustering qui est largement utilisé pour analyser les clusters des grands ensembles de données. Il a été proposé par MacQueen en 1967 et c'était l'un des algorithmes d'apprentissage non supervisé le plus simple, qui a été appliqué pour résoudre les problèmes du clustering [97]. Il s'agit d'un algorithme de partitionnement des clusters qui consiste à classer les objets d'un jeu de données en k clusters différents, de telle sorte que les clusters générés sont compacts et indépendants [98].

5.5.1.1 Le fonctionnement de l'algorithme k-means

K-Means clustering fonctionne de cette façon : par exemple pour regrouper les données en trois clusters on commence par placer trois points appelés centroïde au hasard dans le jeu de données ensuite on affecte chaque point de data set au centroïde le plus proche ce qui nous donne trois clusters puis on déplace chaque centroïde au milieu de son cluster, on recommence jusqu'à ce que les centroïdes convergent vers une position d'équilibre. L'algorithme requiert donc à l'initialisation le nombre k de cluster à générer et k centroïdes (les centres des clusters) sont initialisés avec des coordonnées aléatoires.

5.5.1.2 Les étapes de l'algorithme

L'algorithme de K-means clustering est un algorithme itératif qui procède comme suit :

- Sélection de k clusters à générer, où la valeur k est fixée à l'avance.
- Initialisation des centroïdes avec des coordonnées aléatoires.

- Calcul de la distance entre les objets et le centre de gravité du cluster.
- Affectation des points au centroïde le plus proche.
- Déplacement du centroïde à la moyenne du cluster.
- Répétition des étapes précédentes jusqu'à ce qu'il n'y ait pas de changement au centre des clusters.

Les étapes sont illustrées par la figure ci-dessous.

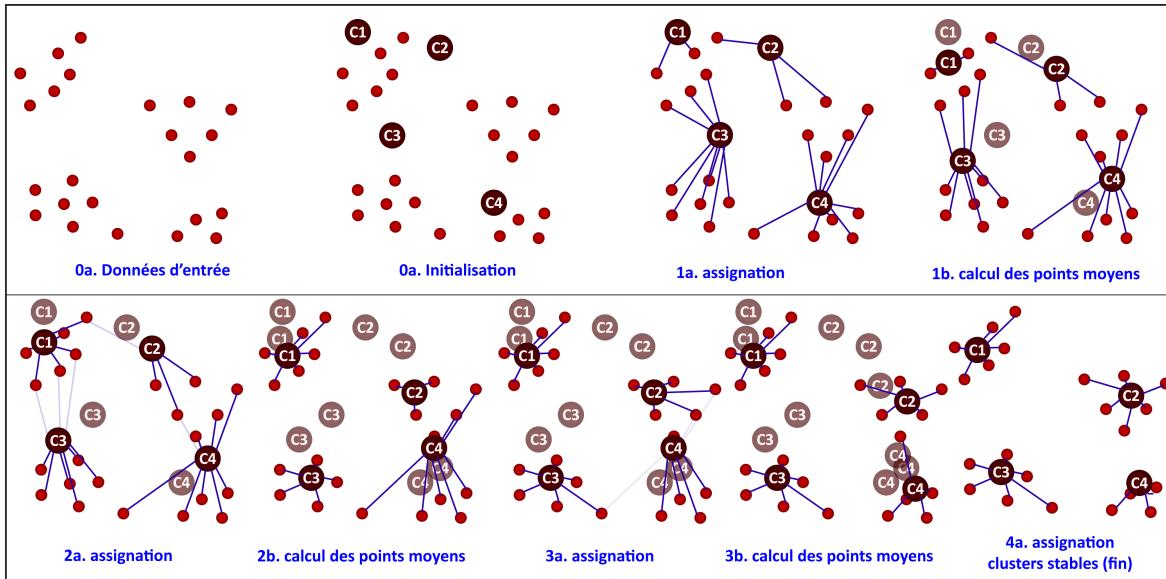


FIGURE 5.3 : Les étapes de l'algorithme k-means.

Après l'initialisation du nombre k des clusters trouvés dans le jeu de données et k centroïdes, la distance euclidienne est généralement considérée pour déterminer la distance entre chaque objet de données et les centres du cluster. Cette distance est utilisée pour regrouper chaque objet de données au centre le plus proche [99]. La distance euclidienne entre un vecteur $X = (x_1, x_2, x_3, \dots, x_n)$ et un autre vecteur $Y = (y_1, y_2, y_3, \dots, y_n)$, La distance euclidienne $d(x_i, y_i)$ peut être obtenue comme suit :

$$d(x_i, y_i) = \left[\sum_{i=1}^n (x_i - y_i)^2 \right]^{\frac{1}{2}} \quad (5.9)$$

Lorsque tous les objets de données sont inclus dans certains clusters, la première étape est terminée et un regroupement précoce est effectué. Ce processus itératif continue jusqu'à ce que la fonction de critère soit minimisée. [98].

En supposant que l'objet cible est x, x_i indique la moyenne du cluster C_i , la fonction critère est définie comme suit :

$$E = \sum_{i=1}^k \sum_{x \in C_i} |x - x_i|^2 \quad (5.10)$$

5.5.1.3 Les lacunes de l'algorithme k-means

L'algorithme de clustering k-means converge toujours vers le minimum local. Avant qu'il ne converge, l'algorithme doit calculer la distance entre chaque objet de données et chaque centre de cluster dans chaque itération. Du au choix aléatoire des centres de cluster initiaux, la valeur précise t connu sous le nom de nombre d'itérations k-means varie en fonction des centres de cluster de départ [100]. Aussi le fait de devoir choisir a priori le paramètre K est un inconvénient, car l'algorithme peut donner des fausses informations sur les clusters générés et il est influencé par des valeurs aberrantes appelé aussi en anglais outliers.

5.5.1.4 Les solutions aux lacunes de l'algorithme k-means

Il n'est pas nécessaire de calculer la distance entre chaque objet de données et chaque centre de cluster dans chaque itération. En supposant que le cluster C s'est formé après l'itération j , l'objet de données x est affecté au cluster C , mais dans quelques itérations, l'objet de données x est toujours affecté au cluster C . Donc si on constate que la distance de l'objet de données x aux autre cluster est petite que au cluster C , il est possible d'arrêter le calcul de la distance lié l'objet x , car ceci provoque un long temps d'exécution affectant ainsi l'efficacité du clustering [98].

Selon la position initiale des centroïdes, K-means peut donner de mauvais clusters. La configuration des clusters trouver par K-means peut ne pas être la plus optimale. On parle d'optimum local. La solution est d'exécuter K-means avec différentes positions de d'épart des centroïdes. La solution retenue est celle qui minimise la somme des distances entre les points (x) d'un cluster et son centre (u). Cela équivaut à minimiser la variance des clusters. K-means cherche la position des centroïdes qui minimise la distance entre les points d'un cluster (X_i) et le centre (U_i) de ce dernier. L'algorithme de K-means cherche donc à minimiser une fonction coût appeler inertie et qui représente la distance entre le point d'un cluster et le centre de ce dernier. Les paramètres de la méthode `kmeans()` à optimiser pour le langage R :

- **Centers :** les clusters obtenus par l'algorithme sont plus cohérent lorsque le nombre de centers demander par l'algorithme est le même que celui des clusters naturels du data set. Dans un data set avec de nombreuses dimensions il est difficile de voir un nombre de clusters à l'œil nu. Pour choisir le bon nombre de cluster il faut utiliser la méthode Elbow qui consiste à tracer l'évolution du cout du model en fonction du nombre de

cluster et de détecter une zone de coude, cette zone nous indique le nombre de cluster optimale. C'est-à-dire celui qui nous permet de réduire au maximum le cout du model tout en conservant un nombre raisonnable de cluster.

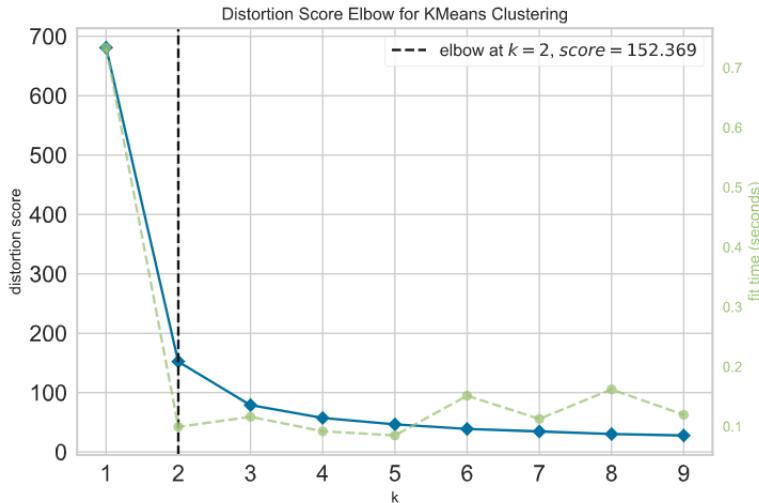


FIGURE 5.4 : La méthode Elbow.

- **nstart** : avec le nombre donner au paramètre nstart l'algorithme tente plusieurs configuration initial de la position des centroïdes et retient la meilleures configuration qui minimise la variance des cluster. La valeur par défaut est 1 mais la valeur recommander est 25 par contre cette valeur conduit ‘a un gaspillage de ressource avec un grand data set.
- **Iter.max** : le nombre de fois que l'algorithme doit s'exécuter. La valeur par défaut est 10 ce qui permet au centroïdes de converger ver position d'équilibre.
- **algorithm** : l'algorithme choisis est celui qui donne une plus forte cohérence des clusters. Avec python l'algorithme kmeans++ donne des meilleurs résultats.

5.6 Fuzzy clustering

5.6.1 Introduction

Les méthodes de clustering traditionnelles (Hard) se limitent du fait que chaque point de l'ensemble de données appartient à un seul cluster. La théorie des ensembles flous proposée par Zadeh [101] en 1965 a fait jaillir le concept d'incertitude d'appartenance, qui est décrit par la fonction d'appartenance [79]. L'utilisation d'ensembles flous fournit des informations

d'appartenance de classe imprécises. L'application de la théorie des ensembles flous dans l'analyse de cluster a été proposée pour la première fois dans les travaux de Bellman, Kalaba et Zadeh [101] et Ruspini [102].

Ruspini a proposé l'utilisation d'ensembles flous dans le clustering sous la forme d'une fonction objective qui obtient une meilleure partition floue des données en utilisant des méthodes Fuzzy (flou). Et cette méthode devient un problème d'optimisation de fonction objective qui appartient à la catégorie du clustering flou basé sur les fonctions objectives. Il existe aussi le clustering flou basé sur une relation floue et le clustering « fuzzy generalized k-nearest neighbor rule ».

De nombreuses méthodes de regroupement flou ont été proposées en raison de modifications des fonctions objectives, qui visent à améliorer le résultat en ce qui concerne le bruit, les valeurs aberrantes, etc [80]. La méthode la plus utilisée est c-means (FCM) qui est une technique non supervisée qui classe les points de données similaires en clusters [87].

Dans de nombreuses situations, le clustering Soft est plus naturel que le clustering Hard parce que les objets situés aux limites entre plusieurs classes ne sont pas forcés d'appartenir entièrement à l'un des clusters, mais se voient plutôt attribuer des degrés d'appartenance compris entre 0 et 1 indiquant leur appartenance partielle. Au contraire, dans les techniques de clustering hard, les données sont groupées de manière exclusive, de sorte que si une certaine donnée appartient à un cluster défini, elle ne peut pas être incluse dans un autre cluster [103]. Nous verrons dans cette section le Fuzzy c-means qui est une technique de clustering flou très populaire.

5.6.2 Fuzzy C-means clustering :

Fuzzy C-means (FCM) est une technique de regroupement de données dans laquelle chaque point de données appartient à un cluster dans une certaine mesure spécifiée par un degré d'appartenance. Cette technique a été initialement développé par Dunn 1973 et améliorer par Jim Bezdek en 1981 par rapport aux méthodes de clustering antérieures [104]. Il fournit une méthode de regroupement des points de données qui peuplent un espace multidimensionnel dans un nombre spécifique de clusters différents. Le principal avantage de cette méthode est qu'elle fournit des appartenances graduelles de points de données à des clusters mesurés en degrés dans un intervalle [0,1]. Cela donne la flexibilité d'exprimer que les points de données peuvent appartenir à plus d'un cluster.

5.6.2.1 Algorithme FCM :

Le but est de minimiser la fonction objective définie comme suit :

$$J_m = \sum_{i=1}^N \sum_{j=1}^C U_{ij}^m \|x_i - c_j\|^2 \quad \text{avec } 1 \leq m < \infty \quad (5.11)$$

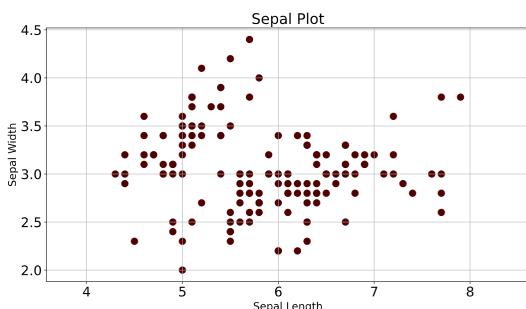
Où U_{ij} est le degré auquel une observation x_i appartient à un cluster j , c_j est le centre du cluster j et m le fuzzifier.

On remarque que FCM diffère de k-means du fait que FCM utilise les valeurs d'appartenance U_{ij} et le fuzzifier m . La variable U_{ij}^m est inversement lié à la distance entre x et le centre de cluster et est défini par :

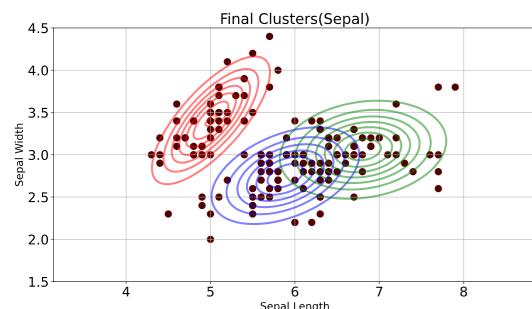
$$U_{ij}^m = \frac{1}{\sum_{k=1}^C \left(\frac{\|x_i - c_k\|}{\|x_i - c_j\|} \right)^{\frac{2}{m-1}}} \quad (5.12)$$

Le paramètre m est un nombre réel supérieur à 1 ($1 \leq m < \infty$) et il définit le niveau de flou du cluster. Une valeur de m proche de 1 donne une solution de cluster qui devient de plus en plus similaire à la solution de clustering **Hard** tel que k-means ; alors qu'une valeur de m proche de l'infini conduit à un flou complet. Le bon choix est d'utiliser $m = 2.0$ [105]. Enfin le centroïde d'un cluster est la moyenne de tous les points, pondérée par leur degré d'appartenance au cluster.

$$C_j = \frac{\sum_{i=1}^N U_{ij}^m \cdot x_i}{\sum_{i=1}^N U_{ij}^m} \quad (5.13)$$



(a) Les éléments du dataset Iris [106].



(b) Illustration du degré d'appartenance des éléments au cluster.

5.6.3 Conclusion

Le clustering flou c-means peut être considéré comme un meilleur algorithme par rapport à l'algorithme k-Means. Contrairement à l'algorithme k-Means où les points de données appartiennent exclusivement à un cluster, dans le cas de l'algorithme flou c-means, le point de

données peut appartenir à plus d'un cluster avec une vraisemblance ou une probabilité. Le regroupement flou c-means donne des résultats comparativement meilleurs pour les ensembles de données qui se chevauchent.

5.7 Les autres méthodes de clustering

En dehors des méthodes de clustering hiérarchique, partitionnel et floue (fuzzy) vu précédemment, les autres méthodes de clustering sont énoncées de façon brève dans le tableau 5.3

Clustering Method	Description	Avantages	Disadvantages	Algorithms
Distribution based Clustering	Basé sur la distribution de probabilité des données, les clusters sont dérivés de diverses métriques telles que la moyenne, la variance, etc.	Le nombre de clusters n'a pas besoin d'être spécifié a priori, fonctionne sur des données en temps réel, les métriques sont faciles à comprendre et à régler.	Algorithme complexe et lent, ne peut pas être adapté à des données plus volumineuses	Gaussian Mixed Models, DB-CLASD
Density based Clustering (Model-based methods)	Basé sur la densité des points de données, également connu sous le nom de clustering basé sur un modèle.	Peut gérer le bruit et les valeurs aberrantes, n'a pas besoin de spécifier le nombre de clusters au début, les clusters créés sont très homogènes, aucune restriction sur les formes de cluster.	Algorithme complexe et lent, ne peut pas être adapté à des données plus volumineuses	DENCAST, DBSCAN
Constraint based (Supervised Clustering)	Le clustering est dirigé et contrôlé par les contraintes	Crée une limite de décision parfaite, peut déterminer automatiquement les classes de résultats en fonction des contraintes, les données futures peuvent être classées en fonction des limites de formation.	Surapprentissage, niveau élevé d'erreurs de classification erronée, ne peut pas être entraîné sur des ensembles de données plus volumineux	Decision Trees, Random Forest, Gradient Boosting.

TABLE 5.3 : Les autres méthodes de clustering

5.8 La différence entre le clustering hiérarchique, partitionnel et Fuzzy clustering

Le clustering hiérarchique et partitionnel présentent des différences essentielles en termes de temps d'exécution, d'hypothèses, de paramètres d'entrée et de clusters résultants. En

règle générale, le clustering partitionnel est plus rapide que le clustering hiérarchique. La classification hiérarchique nécessite uniquement une mesure de similarité, tandis que la classification partitionnelle requiert des hypothèses plus strictes telles que le nombre de classes et les centres initiaux. La mise en cluster hiérarchique ne nécessite aucun paramètre d'entrée, tandis que les algorithmes de mise en cluster partiels nécessitent le nombre de clusters à exécuter. La classification hiérarchique renvoie une division beaucoup plus significative et subjective des grappes, mais la classification partitionnelle produit exactement k clusters. Les algorithmes de classification hiérarchique conviennent mieux aux données catégoriques, à condition qu'une mesure de similarité puisse être définie en conséquence. En terme générale ils sont considérés comme des méthodes de clustering Hard, où chaque élément est affecté à un seul cluster. Par contre les méthodes de clustering Soft comme le Fuzzy clustering regroupent les éléments de données de telle sorte qu'un élément puisse exister dans plusieurs clusters.

5.9 Indice de validité du clustering

Les algorithmes supervisés ont beaucoup de métriques pour vérifier leur qualité d'ajustement comme la précision, la valeur R^2 , la sensibilité, la spécificité, etc. Concernant les algorithmes non supervisés et plus précisément les algorithmes de clustering l'indice de validité de cluster est utilisé pour mesurer la qualité de la partition trouvée et aussi les partitions optimales. L'indice de validité de cluster est utilisé donc pour mesurer la qualité des clusters et pour rechercher le nombre optimal de clusters lorsque le nombre de clusters dans les données l'ensemble n'est pas connu à l'avance [107]. Il utilise les propriétés des clusters telles que la compacité (ou la variation) et la séparation (ou l'isolement) qui sont souvent considérées comme des caractéristiques majeures permettant de valider les clusters. La compacité indique la variation ou la dispersion des données au sein d'un cluster, et la séparation indique l'isolement des clusters les uns des autres [108] . Plusieurs indices de validité courants sont énoncés dans le tableau suivant :

Indice	Description	Formule
Les indices de validité adaptés aux clustering partitionnelle		
Indice de Dunn	Il est l'un des indices les plus anciens et les plus cités est proposé par [109]. L'index de Dunn (DU) identifie les clusters qui sont bien séparés et compacts. Le but est donc de maximiser la distance inter-clusters tout en minimisant la distance intra-cluster.	$DU_k = \min_{i=1,\dots,k} \left\{ \min_{j=1+1,\dots,k} \left(\frac{diss(c_i, c_j)}{\max_{m=1,\dots,k} diam(c_m)} \right) \right\}$ <p>où $diss(c_i, c_j) = \min_{x \in c_i, y \in c_j} d(x, y)$ est la dissemblance entre les clusters c_i et c_j, et $diam(C) = \max_{x,y \in C} d(x, y)$ est la fonction intra-cluster (ou diamètre) du cluster. Si l'index de Dunn est grand, cela signifie qu'il existe des clusters compacts et bien séparés [110].</p>
Indice Calinski-Harabasz	Cet indice [111] est basé sur un rapport entre la matrice de dispersion des clusters (BCSM) et la matrice de dispersion des clusters (WCSM).	$CH_k = \frac{BCSM}{k-1} \cdot \frac{n-k}{WCSM}$ <p>où n est le nombre total de points et k le nombre de clusters. Le BCSM est basé sur la distance entre les clusters et est défini par : $BCSM = \sum_{i=1}^k n_i \cdot d(z_i, z_{tot})^2$, où z_i est le centre du cluster c_i et n_i le nombre de points dans c_i. Le WCSM est : $WCSM = \sum_{i=1}^k \sum_{x \in c_i} d(x, z_i)^2$ où x est un point de données appartenant au cluster c_i. Pour obtenir des clusters bien séparés et compacts, BCSM est maximisé et WCSM minimisé [110].</p>
Indice Davies-Bouldin	Semblable à l'indice de Dunn, l'indice Davies-Bouldin [112] identifie des clusters éloignés les uns des autres et compacte.	$DB_k = \frac{1}{k} \sum_{i=1}^k \max_{j=1,\dots,k; i \neq j} \left\{ \frac{diam(c_i) + diam(c_j)}{d(z_i, z_j)} \right\}$ <p>où le diamètre d'un cluster est : $diam(c_i) = \sqrt{\frac{1}{n_i} \sum_{x \in c_i} d(x, z_i)^2}$, avec n_i le nombre de points et z_i le centre de gravité du cluster c_i. Puisque l'objectif est d'obtenir des clusters avec des distances intra-clusters minimales, les petites valeurs de DB sont intéressantes. Par conséquent, cet index est minimisé lors de la recherche du meilleur nombre de clusters [110].</p>
Indice de silhouette	La statistique de silhouette [113] est une autre façon bien connue d'estimer le nombre de groupes dans une donnée ensemble. L'indice Silhouette (SI) calcule pour chaque point une largeur en fonction de son appartenance à groupe. Cette largeur de silhouette est alors une moyenne sur l'ensemble des observations.	$SI_k = \frac{1}{k} \sum_{i=1}^n \frac{(b_i - a_i)}{\max(a_i, b_i)},$ <p>où n est le nombre total de points, a_i est la distance moyenne entre le point i et tous les autres points de son propre cluster et b_i est le minimum des dissemblances moyennes entre i et les points d'autres clusters. Enfin, la partition avec le SI le plus élevé est considérée comme optimale [110].</p>

TABLE 5.4 : Indice de validité du clustering Hard

Indice	Formule
	Il y'a aussi d'autre indice de validité de clustering partitionnel comme l'indice de Maulik-Bandyopadhyay et Geometric index. Les indices de validité utilisé pour le fuzzy clustering sont :
Le coefficient de partition PC [114]	$V_{PC} = \frac{1}{n} \sum_{i=1}^c \sum_{j=1}^n u_{ij}^2$
L'entropie de partition PE [115]	$V_{PE} = -\frac{1}{n} \sum_{i=1}^c \sum_{j=1}^n u_{ij} \log u_{ij}$
Fonction de validité proposée par Wu et Yang [116]	$V_{PCAES} = \sum_{i=1}^c PCAES_i = \sum_{i=1}^c \sum_{j=1}^n u_{ij}^2 / u_M - \sum_{i=1}^c \exp(-\min_{k \neq i} \{ \ v_i - v_k\ ^2 / \beta_T \})$ où $u_M = \min_{1 \leq i \leq c} \left\{ \sum_{j=1}^n u_{ij}^2 \right\}$, $\beta_T = \frac{\sum_{l=1}^c \ v_l - \bar{v}\ ^2}{c}$ et $\bar{v} = \sum_{j=1}^n x_j / n$ La grande valeur VPCAES signifie que les c clusters sont compact et séparé les uns des autres, et la petite valeur VPCAES implique qu'ils ne sont pas compacts ou séparés.
Fonction de validité proposée par Fukuyama et Sugeno [117]	$V_{FS} = J_m(U, V) - K_m(U, V) = \sum_{i=1}^c \sum_{j=1}^n u_{ij}^m \ x_j - v_i\ ^2 - \sum_{i=1}^c \sum_{j=1}^n u_{ij}^m \ v_i - \bar{v}\ ^2,$ où $\bar{v} = \sum_{j=1}^n x_j / n$.
Fonction de validité proposée par Xie et Beni [118]	$V_{XB} = \frac{J_m(U, V)/n}{Sep(V)} = \frac{\sum_{i=1}^c \sum_{j=1}^n u_{ij}^m \ x_j - v_i\ ^2}{n \min_{i \neq j} \ v_i - v_j\ ^2}$

TABLE 5.5 : Indice de validité du clustering Soft

5.10 Conclusion

Nous avons passé en revue dans cette partie le clustering qui est une méthode de l'apprentissage automatique non-supervisé et les techniques qu'il utilise pour l'extraction des clusters. Plusieurs objectif pousse à utiliser certaines techniques de clustering comme le clustering partitionnel et hiérarchique qui forme des clusters où chaque élément appartient à un seul cluster et le clustering flou (fuzzy) où chaque élément peut appartenir à plusieurs clusters avec un degré d'appartenance. Ensuite après l'extraction des clusters, les indices de validité de clusters peuvent être utilisé pour évaluer la qualité des clusters obtenue où détecter à l'avance le nombre de clusters optimale avant d'utiliser les algorithmes.

Deuxième partie

Contributions

CHAPITRE 6

Contribution and Results

Sommaire

6.1	Introduction	81
6.2	Approche proposée	81
6.3	Implémentations et résultats expérimentaux	81
6.4	Conclusion	103

6.1 Introduction

Nous présentons dans ce chapitre les notions vues dans les chapitres précédents appliquer à un dataset. Partant de la collecte et la préparation du dataset, de l’application de la théorie des réponses aux items pour un ajustement bayésiens des réponses obtenues par les apprenants, jusqu’aux clustering hard et soft des items basés sur la similarité.

6.2 Approche proposée

6.3 Implémentations et résultats expérimentaux

6.3.1 Outils de développement

les outils matériels et logiciels utilisés pour le développement sont :

6.3.1.1 Materiels

Marque	CPU	RAM	OS
Lenovo Y40-80	AMD Intel Core i5 2.20 GHz	16Go	Windows10 64bits
Et Linux Ubuntu 20.04 LTS installé sur WSL2			

TABLE 6.1 : Caractéristiques du matériels utilisés

WSL2 : Windows Subsystem for Linux (WSL) est une couche de compatibilité permettant d’exécuter des exécutables binaires Linux de manière native sur Windows 10 et Windows Server 2019. WSL2 est une version améliorer de WSL1 qui introduit d’importants changements, notamment la présence d’un véritable noyau Linux [119] via un sous-ensemble de fonctionnalités Hyper-V.

6.3.1.2 Outils et packages

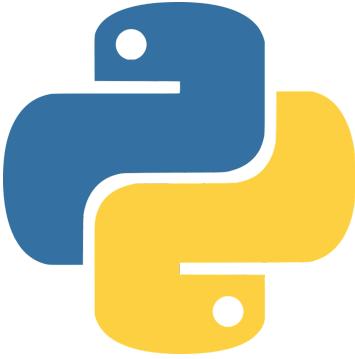
Outils Packages	Description
Les outils et packages utilisés sur windows 10 :	
 Python [120]	<p>Créé par le développeur Guido Van Rossum et publié pour la première fois en 1991. Il permet aux programmeurs d'utiliser différents styles de programmation pour créer des programmes simples ou complexes. Python est l'un des langages de programmation les plus populaires pour la science des données. C'est un langage de programmation de haut niveau, structuré, open source, interprété et dynamique. Il est multiparadigme, multi-plateforme et multi-usages. La syntaxe de Python aide les programmeurs à coder en moins d'étapes par rapport à d'autres langages comme JAVA ou C++, ce qui facilite le travail et le rend plus rapide, facile et amusant à faire. Il possède une bibliothèque standard complète et volumineuse dotée d'une gestion automatique de la mémoire et de fonctionnalités dynamiques [121]. Dans notre travail, nous avons utilisé Python 3.5 intégré à Anaconda.</p>
 Anaconda	<p>Anaconda [122] est une distribution libre et open source [123] des langages de programmation Python et R appliquée au développement d'applications dédiées à la science des données et à l'apprentissage automatique (traitement de données à grande échelle, analyse prédictive, calcul scientifique), qui vise à simplifier la gestion des paquets et de déploiement. Les versions de paquetages sont gérées par le système de gestion de paquets conda [124] et est adapté pour Windows, Linux et MacOS. Anaconda fournit interface graphique qui permet de lancer des applications, de gérer les librairies avec gestionnaire de paquets conda, et les environnements de développement. Plusieurs applications sont disponibles par défaut comme JupyterLab, Jupyter Notebook, QtConsole5, Spyder, Glue, Orange, RStudio, Visual Studio Code.</p>
 Jupyter Notebook [125]	<p>Jupyter Notebook est un environnement de programmation interactif basé sur le web utilisé pour programmer en python, R, Ruby, Julia ou encore Scala qui permet de réaliser des notebooks contenant à la fois du texte en markdown et du code en Julia, Python, R etc.</p>
 Scikit-learn [126]	<p>Scikit-learn est une bibliothèque libre Python destinée à l'apprentissage automatique. Elle propose un ensemble d'outils efficace clé en main pour l'analyse et l'exploration de données. Cette bibliothèque prend en charge les algorithmes de classifications, de régression, du clustering, la réduction de dimensionnalité et de prétraitement des données pour l'extraction et la normalisation des caractéristiques.</p>

TABLE 6.2 : Indice de validité du clustering Hard

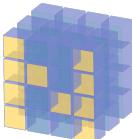
Outils Packages	Description
 Pandas [127]	Pandas est une bibliothèque écrite en Python qui permet la manipulation et l'analyse des données. Elle propose en particulier des structures de données et des opérations de manipulation de tableaux numériques et de séries temporelles. Elle propose principalement comme structures de données les Series, DataFrames, Panels, Panels4D. Et aussi des fonctionnalités comme la manipuler des données aisément et efficacement avec des index pouvant être des chaînes de caractères, des outils pour lire et écrire des données structurées, gestion des données manquantes, le tri, le redimensionnement et , fusion et jointure de large volume de données et analyse de séries temporelles.
 NumPy	est une bibliothèque python pour le calcul scientifique qui fournit un objet tableau multidimensionnel, divers objets dérivés (tels que des tableaux et des matrices masqués) et un assortiment de routines pour des opérations rapides sur des tableaux, notamment mathématiques, logiques, manipulation de forme, tri, sélection, transformées de Fourier discrètes, algèbre linéaire de base, opérations statistiques de base, simulation aléatoire etc.
 Matplotlib [129]	Matplotlib est une bibliothèque du langage de programmation Python destinée à tracer et visualiser des données sous formes de graphiques en 2 ou 3 dimensions [130]. Cette bibliothèque permet de produire divers tracés par exemple des histogrammes, des fonctions de Rosenbrock, spirale logarithmique, graphiques d'erreurs, nuages de points etc.

TABLE 6.3 : Indice de validité du clustering Hard

Outils Packages	Description
Les outils et packages supplémentaires utilisés sur Linux Ubuntu 20.04 LTS installé sur WSL2 :	
 Stan	Stan [75] est une plate-forme pour la modélisation statistique, l'analyse de données et la prédition dans les sciences sociales, biologiques et physiques, l'ingénierie et les affaires. Il permet de spécifier les fonctions de densité de log afin d'obtenir une inférence statistique bayésienne complète avec échantillonnage MCMC (NUTS, HMC), une inférence bayésienne approximative avec inférence variationnelle (ADVI), une estimation du maximum de vraisemblance pénalisée avec optimisation (L-BFGS). La bibliothèque mathématique de Stan fournit des fonctions de probabilité différentiables et une algèbre linéaire (C++ autodiff). Stan peut être utilisé avec les langages d'analyse de données les plus populaires (R, Python, shell, MATLAB, Julia, Stata) et fonctionne sur toutes les principales plates-formes (Linux, Mac, Windows) sauf la version 3 qui fonctionne sur Linux et Mac, c'est ce qui nous a poussée à utiliser Linux Ubuntu 20.04 LTS sur WSL2. Dans l'implémentation nous avons utilisé Pystan 3.2.0 [131] qui est une interface Python pour Stan, un package pour l'inférence bayésienne.
 ArviZ	ArviZ [132] est un package Python pour l'analyse exploratoire des modèles bayésiens. Comprend des fonctions d'analyse postérieure, de stockage de données, de diagnostic d'échantillons, de vérification de modèle et de comparaison. Ce package fournit des outils backend indépendants pour les diagnostics et les visualisations de l'inférence bayésienne en Python, en convertissant d'abord les données d'inférence en objets xarray [133].

TABLE 6.4 : Indice de validité du clustering Hard

6.3.2 Collecte et préparation des données

KDD Cup 2010 est une compétition d'exploration de données éducatives dans laquelle les participants sont chargés de prédire les performances des élèves aux problèmes algébriques en fonction des informations concernant les performances passées.

Algèbre I 2005-2006 [134] est un jeu de données qui est fourni pour débuter et se familiariser avec le format et le développement d'un modèle d'apprentissage. Une brève description du dataset initial avant l'étape du pre-processing est dans le tableau 6.5.

Caractéristique	Description	Nombre Total
le nombre total de transactions = 809694.		
Anon Student Id	identifiant unique et anonyme d'un étudiant	574
Problem Name	identifiant unique d'un problème	1084
Correct First Attempt	l'évaluation par le tuteur de la première tentative de l'étudiant 1 si correcte, 0 sinon.	
Et 11 autres caractéristiques du dataset : Row, Problem View, Step Start Time, etc.		

TABLE 6.5 : Description et statistiques du jeu de données.

Les étapes suivies pour la collecte et la préparation du dataset sont illustré par la figure.

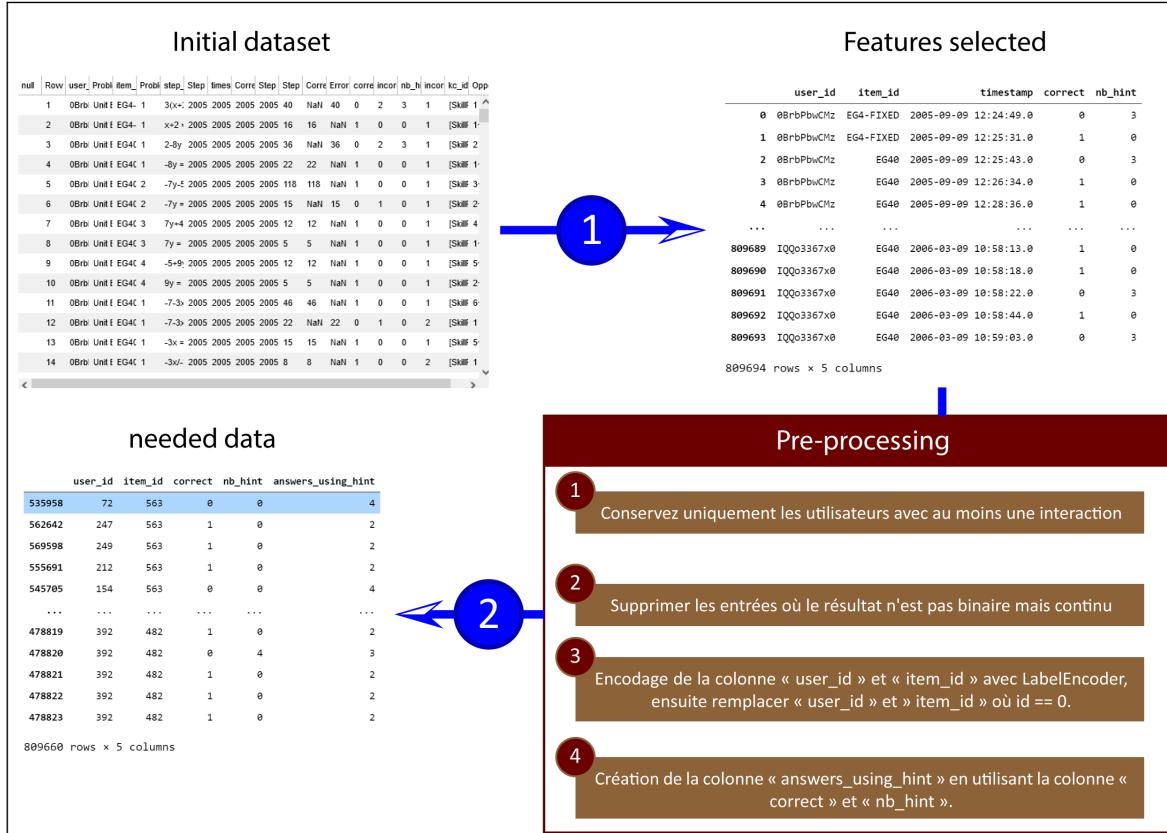


FIGURE 6.1 : Le pre-processing du dataset.

Première étape : dans notre travail, nous nous concentrons sur la performance de l'apprenant qui est liée à l'item, au résultat obtenu par l'apprenant : (correct/incorrect soit 0 ou 1), ou en prenant un indice (hint).

Deuxième étape : seuls les apprenants ayant au moins une interactions avec les items ont été sélectionner. Les valeurs qui ne sont pas binaire (0 ou 1) dans la colonne « correct » sont éliminées. Les valeurs de la colonne « user_id » et « item_id » sont encoder et où « id = 0 » est remplacer par le nombre total des apprenants et des items respectivement. L'encodage permet d'encoder les identifiants en valeur numérique, parce que le modèle IRT prend seulement des valeurs numériques à partir de 1. La colonne « answers_using_hint » ajouter au dataset sera utilisé pour calculer la matrice de similarité entre items selon notre critère : réponse correcte avec aide (hints) et sans aide, et incorrecte avec aide et sans aide. Le script python avec lequel la colonne « answers_using_hint » est dans [6.3.2](#).

```
1 conditions = [
2     (needed[ 'correct' ] == 1) & (needed[ 'nb_hint' ] > 0),
3     (needed[ 'correct' ] == 1) & (needed[ 'nb_hint' ] == 0),
4     (needed[ 'correct' ] == 0) & (needed[ 'nb_hint' ] > 0),
5     (needed[ 'correct' ] == 0) & (needed[ 'nb_hint' ] == 0)
6 ]
7 # 1 ==> Correct avec hint
8 # 2 ==> Correct sans hint
9 # 3 ==> Incorrect avec hint
10 # 4 ==> Incorrect sans hint
11 values = [1, 2, 3, 4]
12
13 # create a new column and use np.select to assign values to it using our lists as
14 # arguments
15 needed[ 'answers_using_hint' ] = np.select(conditions , values)
```

6.3.3 Modèle IRT pour une inférence bayésienne

Dans l’implémentations nous utiliseront le workflow bayésien complet pour faire l’ajustement des réponses aux items plutôt qu’une simple inférence bayésienne. La différence est que l’inférence bayésienne n’est que la formulation et le calcul de probabilités conditionnelles ou de densités de probabilité, $p(\theta|y)\propto p(\theta)p(y|\theta)$ et le workflow bayésien comprend les trois étapes : création de modèle, inférence et vérification/amélioration de modèle, ainsi que la comparaison de différents modèles, non seulement à des fins de choix de modèle ou de moyenne de modèle, mais plus important encore pour mieux comprendre ces modèles [4]. Les étapes du workflow (flux) sont illustrées par la figure 6.2.

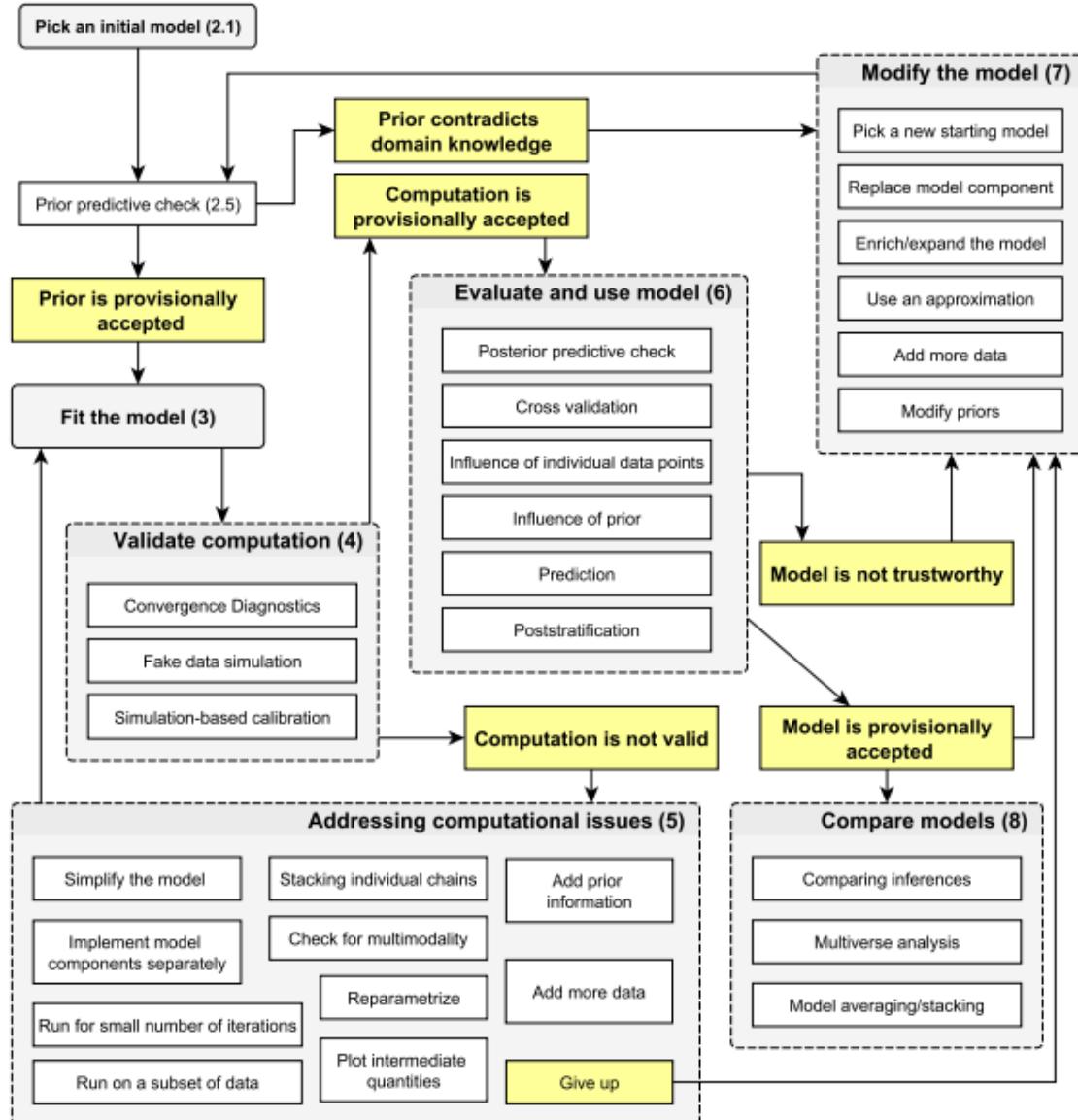


FIGURE 6.2 : Les étapes du workflow bayésien [4].

6.3.3.1 Spécification et chargement du modèle

Nous avons utilisé le modèle de Rasch, 2PL et 3PL pour faire l'estimation des paramètres des modèles avec le logiciel Stan en utilisant le code Stan définit dans le chapitre 4, sauf le bloc « generated quantities » parce qu'en l'ajoutant, la taille de modèle ajusté devient déraisonnablement grande du fait qu'on a un grand nombre d'item, de sujet et d'observation.

Le code stan pour le modèle de Rasch est :

```

1 _1pl_model = """
2 data {
3     // numbers of things
4     int<lower=1> N;    // number of observations
5     int<lower=1> I;    // items , number of questions

```

```
6 int<lower=1> S; // subjects , number of learners
7 // data
8 int<lower=1,upper=I> item[N];
9 int<lower=1,upper=S> subject[N];
10 int<lower=0,upper=1> grade[N];
11 }
12 parameters {
13 // parameters
14 real ability[S]; // alpha: ability of student
15 real difficulty[I]; // beta: difficulty of question
16 real delta; // mean student ability
17 }
18 model {
19 ability ~ normal(0,1);
20 difficulty ~ normal(0,1);
21 delta ~ normal(0.75,1);
22 for(n in 1:N)
23 grade[n] ~ bernoulli_logit(ability[subject[n]] - difficulty[item[n]] + delta);
24 }
"""

```

Avant de charger et compiler le modèle, Stan prend les données dans un dictionnaire, où les noms des clés doivent être identiques aux noms spécifiés dans la section data du modèle. Les données de notre dataset dans un format dictionnaire est :

```
1 { 'I': 1084,
2   'N': 809694,
3   'S': 574,
4   'grade': array([0, 1, 1, ..., 1, 1, 1]),
5   'item': array([563, 563, 563, ..., 482, 482, 482]),
6   'subject': array([-72, 249, 251, ..., 395, 395])}
```

6.3.3.2 Compilation du modèle et échantillonnage

Après avoir spécifier le modèle, la compilation (en code c++) est faite en utilisant la fonction `stan.build()` qui prend en paramètre le modèle , les données et `random_seed` qui contrôle l'effet aléatoire. Le script est :

```
1 posteriori = stan.build(_1pl_model, data=train_data, random_seed
2 =2021)
```

Une fois le modèle compiler avec les données, la méthode `sample()` fait des échantillonnages dans les distributions spécifier dans le modèle. La méthode `sample()` prend en paramètre `num_chains` le nombre de chaine qui sont exécuter en parallèle, `num_samples` le nombre d'échantillon à générer, `num_warmup` le nombre d'échantillon initial à rejeter qui sont généralement loin d'être optimales et `num_thin` spécifie un intervalle d'échantillonnage auquel les échantillons sont conservés.

```
1 fit = posteriori.sample(num_chains=4, num_samples=2000,num_warmup  
=1000,num_thin=1)
```

6.3.3.3 Résultats d'échantillonnage, diagnostique du modèle, prédiction et validation

Une fois l'échantillonnage terminé, l'objet `stanfit` renvoyé par la méthode `sample()` contient la sortie dérivée de l'ajustement du modèle c'est-à dire les résultats de l'inférence. L'objet `fit` du modèle de Rasch est afficher dans le tableau de la figure 6.3. La figure 6.4 affiche les distributions postérieures des paramètres du modèle de Rasch pour toute les itérations et la figure 6.5 les distributions postérieures pour le paramètre `ability` du premier sujet (apprenant) et `difficulty` du premier item du modèle de Rasch.

parameters	lp__	accept_stat__	stepsize__	treedepth__	n_leapfrog__	divergent__	energy__	ability.1	ability.2	ability.3	...	difficulty.1076
draws												
0	-416448.288708	0.998050	0.052932	6.0	63.0	0.0	417288.019118	-0.022487	-0.107462	-0.140198	...	0.353572
1	-416443.188906	0.890472	0.074084	6.0	63.0	0.0	417253.103468	-0.066812	-0.437116	0.102424	...	0.379315
2	-416504.473024	0.780922	0.073781	6.0	63.0	0.0	417357.224283	-0.103783	-0.383655	-0.059003	...	0.393575
3	-416466.823558	0.984599	0.053784	6.0	63.0	0.0	417260.273036	0.041638	-0.185406	-0.060335	...	0.361062
4	-416478.655666	0.863225	0.052932	6.0	63.0	0.0	417292.610382	-0.025012	-0.299755	-0.194773	...	0.348540
...
7995	-416387.973215	0.939589	0.053784	6.0	63.0	0.0	417224.794425	-0.038801	-0.165676	0.025864	...	0.426115
7996	-416412.235114	0.984944	0.052932	6.0	63.0	0.0	417274.728681	0.047786	-0.234386	0.085228	...	0.459138
7997	-416451.136363	0.997193	0.074084	6.0	63.0	0.0	417293.316726	0.013349	-0.333091	0.040941	...	0.282024
7998	-416460.505170	0.991663	0.073781	6.0	63.0	0.0	417272.823532	-0.010581	-0.264042	-0.095163	...	0.405627
7999	-416400.448150	0.991106	0.053784	6.0	63.0	0.0	417216.633927	0.000125	-0.289614	0.079655	...	0.449201

8000 rows × 1666 columns

FIGURE 6.3 : L'objet stanfit du modèle de Rasch.

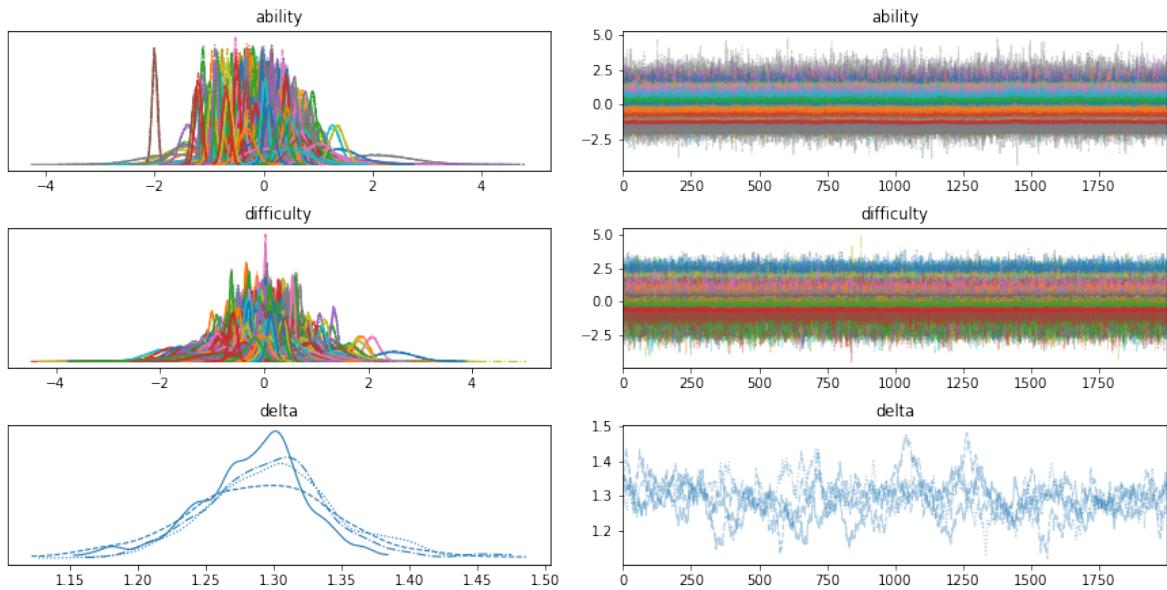


FIGURE 6.4 : Les distributions postérieures du modèle de Rasch.

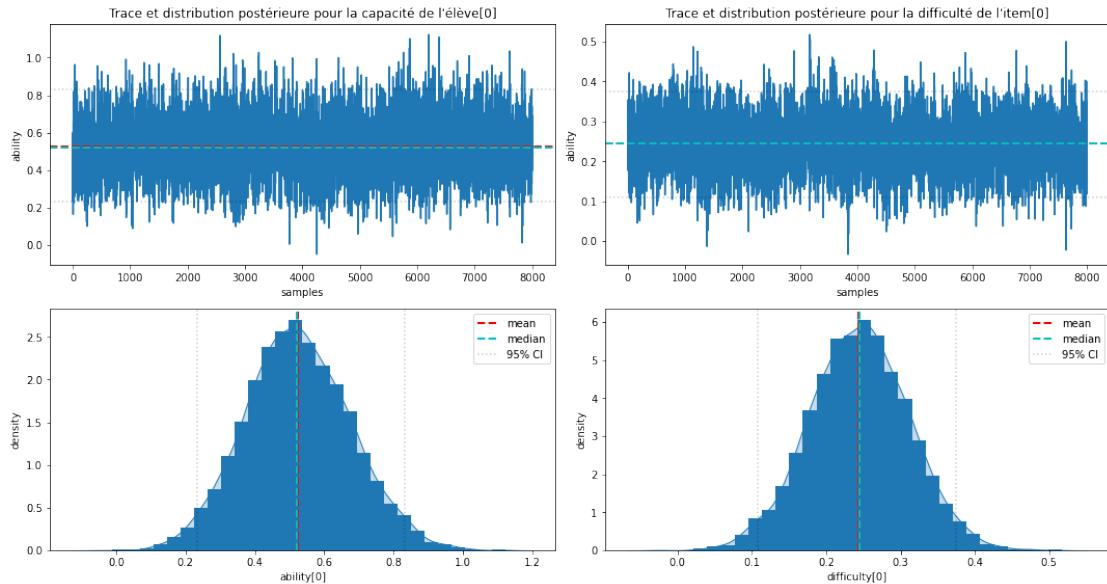


FIGURE 6.5 : Distribution postérieure de la capacité de l'apprenant[72] et de la difficulté de l'item[563] du modèle de Rasch.

En plus des valeurs des paramètres du modèle, le tableau de la figure contient également d'autre valeurs des paramètres utilisés par l'échantillonneur qui vont servir de diagnostique. L'étiquette `lp_` est pour les densités logarithmiques (jusqu'à une constante additive), `accept_stat_` est pour les probabilités d'acceptation, `stepsize_` est pour la taille de pas de l'intégrateur leapfrog pour simuler l'hamiltonien, `treedepth_` est la profondeur de l'arbre exploré par l'échantillonneur sans demi-tour (Journal base 2 du nombre d'évaluations de densité

et de gradient log), `n_leapfrog` est le nombre d'évaluations de densité et de gradient, `divergent` est un indicateur indiquant une instabilité numérique lors de l'intégration numérique entraînant la non conservation de l'hamiltonien, et `energy` est la valeur hamiltonienne.

Le BFMI : Bayesian Fraction of Missing Information, `bfmi` inférieur à 0,2 indique qu'il faut peut-être reparamétriser votre modèle.

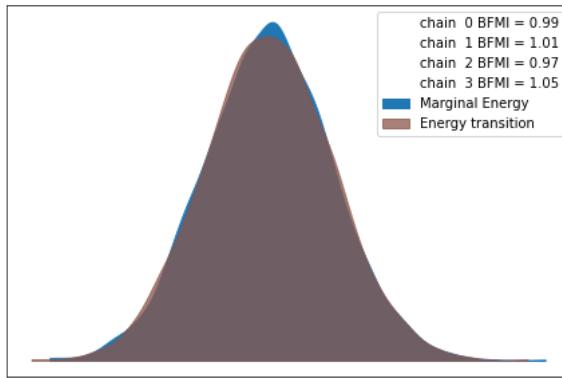


FIGURE 6.6 : BFMI du modèle de Rasch.

Le Rhat : un $Rhat > 1.1$ indique généralement des problèmes d'ajustement et si $Rhat$ est d'environ 1, alors il n'y aura aucune diminution de la variance d'échantillonnage, quelle que soit la durée d'itération, et donc la chaîne de Markov est susceptible (mais pas garantie) d'avoir convergé. Celui ne notre modèle est d'environ 1 pour tous les paramètres.

```
<xarray.Dataset>
Dimensions:          (ability_dim_0: 574, difficulty_dim_0: 1084)
Coordinates:
  * ability_dim_0    (ability_dim_0) int64 0 1 2 3 4 5 ... 569 570 571 572 573
  * difficulty_dim_0 (difficulty_dim_0) int64 0 1 2 3 4 ... 1080 1081 1082 1083
Data variables:
  ability            (ability_dim_0) float64 1.037 1.004 1.006 ... 1.007 1.017
  difficulty         (difficulty_dim_0) float64 1.004 1.002 ... 1.001 1.005
  delta              float64 1.055
```

FIGURE 6.7 : Le Rhat du modèle de Rasch.

Vérification des divergences : aucune divergence dans le modèle comme le montre la figure 6.8.

xarray.Dataset	
► Dimensions:	(chain: 4, draw: 2000)
► Coordinates:	(2)
▼ Data variables:	
accept_stat	(chain, draw) float64 0.9981 0.8632 ... 0.9396 0.9911
stepsize	(chain, draw) float64 0.05293 0.05293 ... 0.05378 0.05378
treedepth	(chain, draw) int64 6 6 6 6 6 6 6 ... 6 6 6 6 6 6 6
n_leapfrog	(chain, draw) int64 63 63 63 63 63 ... 63 63 63 63 63
diverging	(chain, draw) bool False False False ... False False
energy	(chain, draw) float64 4.173e+05 4.173e+05 ... 4.172e+05
lp	(chain, draw) float64 -4.164e+05 ... -4.164e+05
► Attributes:	(9)

FIGURE 6.8 : La sortie de la divergence du modèle de Rasch.

Après diagnostique du modèle, les valeurs des paramètres obtenu peuvent être utilisé pour faire des prédictions postérieures en utilisant la fonction logistique de Rasch 4.21 ou bien faire des prédictions directement en utilisant le bloc « generated quantities » 4.4.1.4 et aussi récupérer le log de vraisemblance (log likelihood) qui est utilisé pour comparer plusieurs modèles (sélection de modèle). Mais l'utilisation de ce bloc entraîne un coût de calcul et d'utilisation de la mémoire surtout avec un jeu de données de grande dimension. Le script 6.3.3.3 est utilisé pour prédire la probabilité de réussite à un item pour le modèle de Rasch.

```

1 ability = np.mean(fit['ability'], axis=1)
2 difficulty = np.mean(fit['difficulty'], axis=1)
3 y_pred = []
4 for i in range(0, len(needed)):
5     diff = train_data['item'][i] # Item index
6     abilt = train_data['subject'][i] # Subject index
7     p = np.exp(ability[abilt - 1] - difficulty[diff - 1])/(1+np.exp(ability[abilt - 1]
8         - difficulty[diff - 1]))
9     y_pred.append(p)
10    y_pred = np.round(y_pred).astype(int)

```

Nous terminons donc cette étape avec la validation des prédictions faite par les modèles IRT utilisés en comparant les scores observés et les scores prédit, c'est-à-dire chercher à comprendre à quel point les modèles représentent les données. Plusieurs métriques peuvent être utilisé pour faire les contrôles prédictif postérieur.

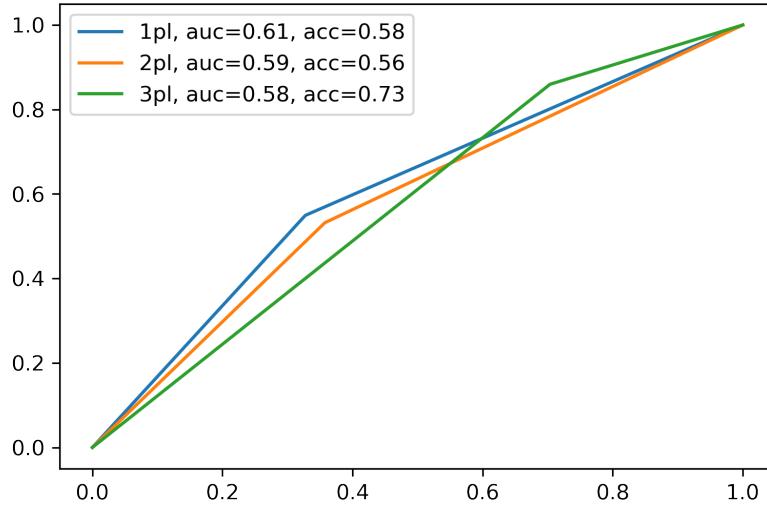


FIGURE 6.9 : Représentation graphique du taux de vrais positifs par rapport au taux de faux positifs des trois modèles utilisés.

Métriques	Rasch Model	2PL	3PL
roc_auc_score	0.611	0.58	
accuracy	0.58	0.56	0.73
recall_score	0.55	0.53	0.86
Average precision-recall score	0.81	0.80	0.80

TABLE 6.6 : Les différents scores du modèle de Rasch, modèle logistique à deux paramètres et le modèle à trois paramètres.

	precision	recall	f1-score	support
0(incorrect)	0.31	0.67	0.43	189052
1(correct)	0.85	0.55	0.67	620642
accuracy			0.58	809694
macro avg	0.58	0.61	0.55	809694
weighted avg	0.72	0.58	0.61	809694

TABLE 6.7 : Rapport de classement (réponse correcte et incorrecte) entre les données observées et les prédictions du modèle de Rasch.

Les évaluations des performances précédentes des modèles tournent autour de 55% et 60% pour toutes les métriques utilisées. Dans cette étude, en respectant le workflow bayésien, les calculs et les modèles sont provisoirement acceptés. Cependant, plusieurs problèmes peuvent être à l'origine des scores pas assez bons obtenus comme le manque de données (les étudiants

n'ont pas répondu à toutes les questions). Il y'a donc un intérêt de collecter le jeu de données conte tenu des données manquantes. La question qui se pose ici est :

- Est-ce-que les prédictions des modèles peuvent être considérer comme un ajustement bayésien des réponses aux items ?
- Est-ce-que les modèles améliorent la notation des tests ?

Dans la section suivante, les données sont utilisées pour faire le clustering des items basés sur la similarité.

6.3.4 Clustering hard et soft de la matrice de similarité

6.3.4.1 Création de la matrice de similarité

Les données collecter dans la partie [6.3.2] et ceux de l'ajustement bayésien avec le modèle de Rasch (c'est-à dire les prédictions faites faire le modèle d'inférence bayésien) sont utilisée pour créer la matrice de réponse selon quatre catégories : correct avec aide (CH) et sans aide (CS), et incorrect avec aide (IH) et sans aide (IS). La matrice de réponses est ensuite utilisée pour créer la matrice de similarité entre items avec le coefficient de kappa. La figure 6.10 illustre toutes les étapes.

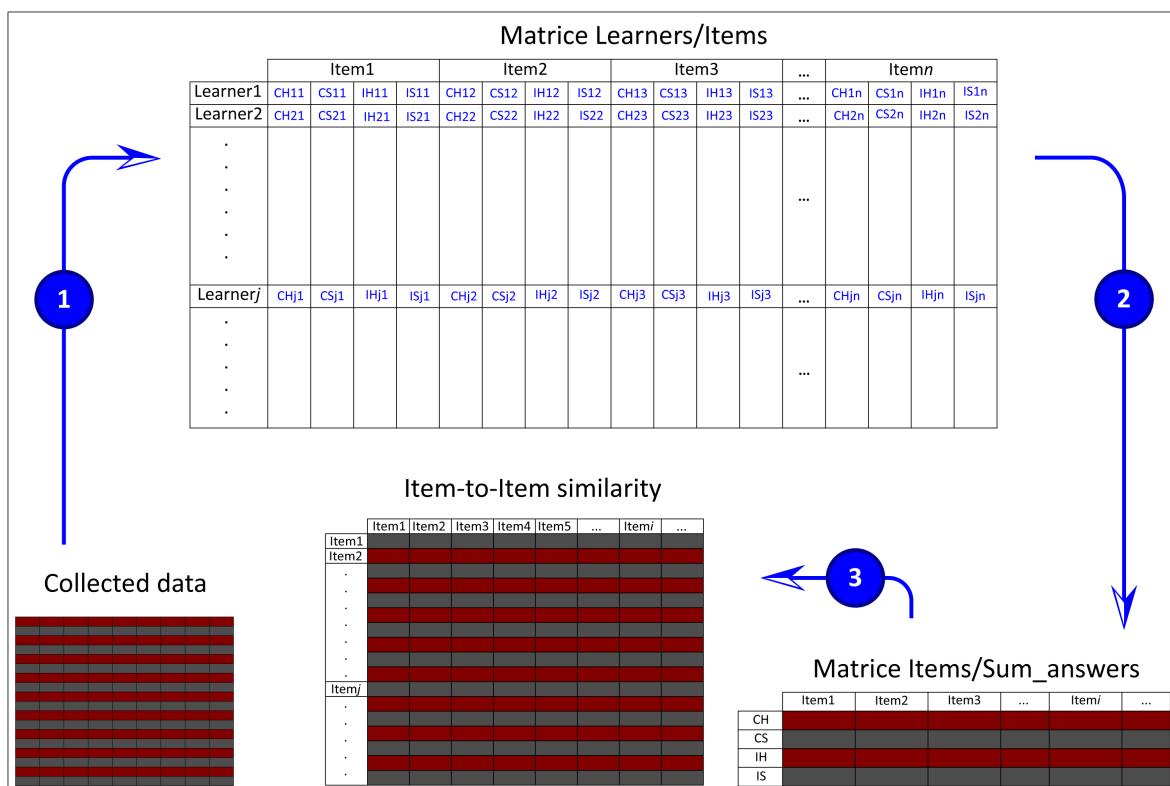


FIGURE 6.10 : Les étapes de calcule de la matrice de similarité.

Le script utiliser pour calculer la matrice des sommes des réponses est :

```

1 def createSumMatrix(data):
2     column_names = data['item_id'].unique()
3     rows_name = ["CH", "CS", "IH", "IS"]
4     columns = pd.DataFrame(columns = column_names)
5     index = pd.DataFrame(rows_name, columns = ['catégories'])
6     sumOfAnswers_df = pd.concat([index,columns], axis=1)
7     sumOfAnswers_df = sumOfAnswers_df.set_index("catégories")
8     sumOfAnswers_df = sumOfAnswers_df.replace(np.nan,0)
9     column_name = data['item_id'].unique()
10    rows_name = data['user_id'].unique()
11    for i in tqdm(rows_name):
12        for j in column_name:
13            dictVal = data[(data["user_id"] == i) & (data["item_id"] == j)][
14                'answers_using_hint'].value_counts()
15            chVal = sumOfAnswers_df.loc["CH",j]
16            csVal = sumOfAnswers_df.loc["CS",j]
17            ihVal = sumOfAnswers_df.loc["IH",j]
18            isVal = sumOfAnswers_df.loc["IS",j]
19            if len(dictVal) != 0:
20                for key, value in dictVal.items():
21                    key = int(key)
22                    if key == 1:
23                        chVal = int(chVal) + value
24                    elif key == 2:
25                        csVal = int(csVal) + value
26                    elif key == 3:
27                        ihVal = int(ihVal) + value
28                    elif key == 4:
29                        isVal = int(isVal) + value
30            sumOfAnswers_df.loc["CH",j] = chVal
31            sumOfAnswers_df.loc["CS",j] = csVal
32            sumOfAnswers_df.loc["IH",j] = ihVal
33            sumOfAnswers_df.loc["IS",j] = isVal
34
35    return sumOfAnswers_df

```

La matrice de similarité entre items est calculée avec le code [6.3.4.1](#).

```

1 def item_to_item_similarity1(data):
2     column_names = data.columns
3     columns = pd.DataFrame(columns = column_names)
4     items = pd.DataFrame(column_names, columns = ['item_id'])
5     items_similarity = pd.concat([items,columns], axis=1)
6     items_similarity = items_similarity.set_index("item_id")
7     index = data.columns
8     for i in tqdm(index):
9         for j in index:
10             val = cohen_kappa_score(data[i],data[j])
11             items_similarity.loc[i,j] = val
12
13    return items_similarity

```

6.3.4.2 Analyse en composantes principales (ACP)

Les grands ensembles de données incluent souvent des mesures sur de nombreuses variables. Il peut être possible de réduire considérablement le nombre de variables tout en conservant une grande partie des informations dans l'ensemble de données d'origine. Un certain nombre de techniques de réduction de dimension existent pour ce faire, et l'analyse en composantes principales est probablement la plus largement utilisée d'entre elles. L'analyse en composantes principales réduira les dimensions et abstraire la signification des variables caractéristique et permet ainsi de saisir les facteurs qui influencent la variance des échantillons.

L'analyse en composantes principales (ACP ou PCA en anglais pour principal component analysis), analyse un tableau de données représentant des observations décrites par plusieurs variables dépendantes, qui sont, en général, corrélées entre elles. Son objectif est d'extraire les informations importantes du tableau de données et d'exprimer ces informations sous la forme d'un ensemble de nouvelles variables orthogonales appelées composantes principales [135].

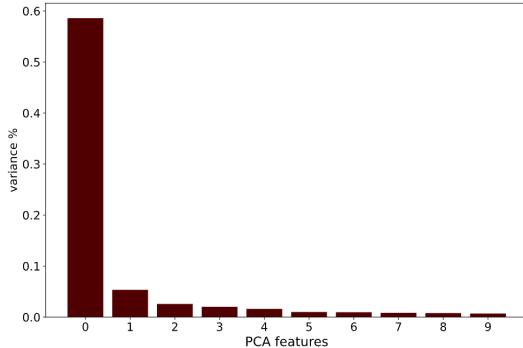
Les objectifs de l'ACP sont :

- Extraire les informations les plus importantes du tableau de données ;
- Compresser la taille de l'ensemble de données en ne gardant que ces informations importantes ;
- Simplifier la description de l'ensemble de données ; et
- Analyser la structure des observations et les variables.

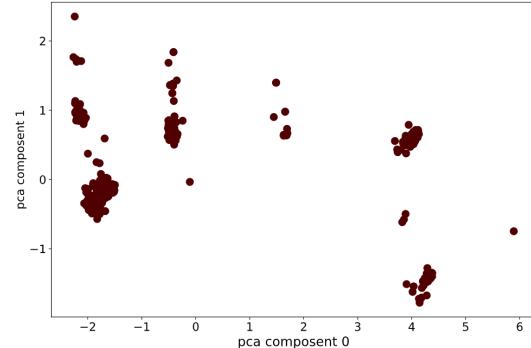
Afin d'atteindre ces objectifs, l'ACP calcule de nouvelles variables appelées composantes principales qui sont obtenues sous forme de combinaisons linéaires des variables d'origine. La première composante principale doit avoir la plus grande variance possible (c'est-à-dire l'inertie et donc cette composante « expliquera » ou « extraira » la plus grande partie de l'inertie du tableau de données). La deuxième composante est calculée sous la contrainte d'être orthogonale à la première composante et d'avoir la plus grande inertie possible. Les autres composantes sont calculées de la même manière. Les valeurs de ces nouvelles variables pour les observations sont appelées scores de facteurs, et ces scores de facteurs peuvent être interprétés géométriquement comme les projections des observations sur les composantes principales [135].

Nous allons appliquer l'analyse en composantes principales avant le clustering, d'abord la visualisation des données ensuite la compression des données après quoi le clustering deviendra précis et efficace.

6.3.4.2.1 Visualisation de données On projette notre dataset dans un espace 2D.

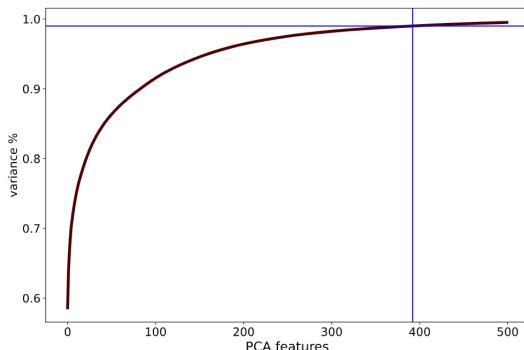


(a) Le pourcentage de la variance préserver par les 10 premières composantes.

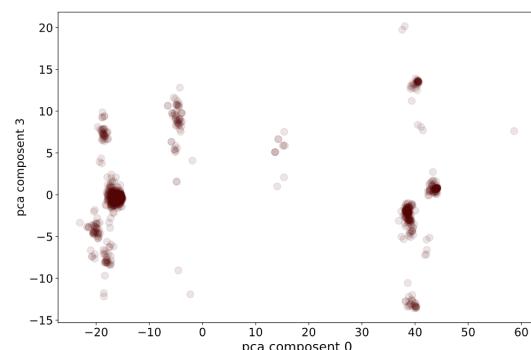


(b) Visualisation en 2D des deux composantes.

6.3.4.2.2 Compression de données La compression avec l'ACP réduit la taille du dataset tout en conservant 99% de la variance des données.



(c) Le pourcentage de la variance préserver par chacune des composantes.



(d) Visualisation de la composante 0 et 3 parmi les 392.

Visualisation de la composante 0 et 3 parmi les **392** et l'objet `features` ci-dessous sera utiliser pour faire le clustering.

```

1 pca = decomposition.PCA(n_components=392)
2 pca.fit(items_similarity)
3 features = pca.transform(items_similarity)

```

6.3.4.3 Clustering hard

6.3.4.3.1 K-means Clustering K-means est une méthode de clustering qui partitionne un ensemble de données en K clusters distincts et non chevauchants. Avant d'utilise cette méthode, on cherche d'abord le nombre optimal de cluster avec la méthode Elbow. Le nombre de clusters utilisé pour le clustering partitionnel avec l'algorithme k-means **4** est choisi en se basant sur la figure.

Algorithm 4 K-means Clustering

```

1: procedure KMEANS( $X, k$ )
2:   Initialize  $k$  centroids  $c_1, \dots, c_k$ 
3:   while centroids not converged do
4:     for all  $x_i \in X$  do                                 $\triangleright$  Expectation
       $r_{ik} = \begin{cases} 1 & \text{if } k = \operatorname{argmin}_k \|x_i - c_k\|^2 \\ 0 & \text{else} \end{cases}$           (6.1)
    5:   end for
    6:   for all  $centroids c_j$  do                          $\triangleright$  Maximization
       $c_j = \frac{\sum_{i=1}^n r_{ij} x_i}{\sum_{i=1}^n r_{ij}}$           (6.2)
    7:   end for
    8:   end while
    9:   return For each  $x_i$ , return last  $k$  where  $r_k == 1$  and  $c_1, \dots, c_k$ .
10: end procedure

```

6.3.4.3.2 Clustering hiérarchique agglomérative Le clustering hiérarchique agglomérative est une approche « ascendante » qui utilise trois distance mesures principales : liaison unique, liaison complète et liaison moyenne, dont le but est de minimiser l'inertie intra-cluster et maximiser l'inertie inter-cluster. L'algorithme de la méthode agglomérative est :

Algorithm 5 Agglomerative Clustering.

```

1: procedure HAC( $matrix D$ , method  $d(A, B)$ )
2:   Initialize a cluster per sample  $\forall i \in x_i : C_i = i$ 
3:   Initialize available cluster  $S = \{C_1, \dots, C_n\}$ 
4:   while  $S$  not empty do
5:      $C_j, C_k = \operatorname{argmin}_{C_j, C_k \in S} d(C_j, C_k)$            $\triangleright$  Closest two clusters
6:      $C_z = C_j \cup C_k, S = S \setminus \{C_j, C_k\}$            $\triangleright$  Merge and mark unavailable
7:     if  $C_z$  not all samples then
8:        $S = S \cup C_z$ 
9:     end if
10:     $\forall C_i : \text{update } D(C_i, C_z)$            $\triangleright$  Update distances
11:   end while
12: end procedure

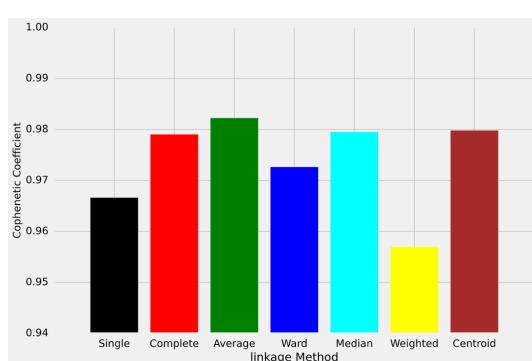
```

Cependant le critère de liaison utilisé influence les résultats du clustering. Pour remédier à cela, On peut utiliser le coefficient de Cophénétique pour trouver la méthode de liaison optimale. Ensuite le script est utilisé pour faire le clustering hiérarchique agglomérative.

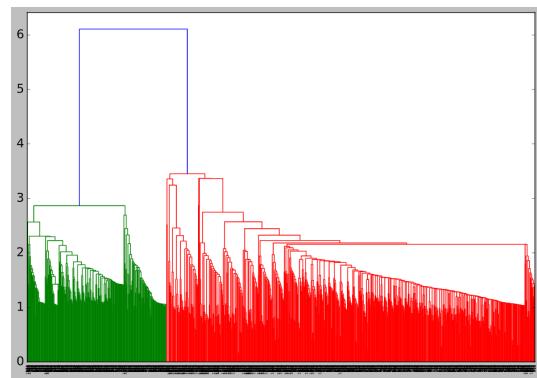
```

1  dendrogram = sch.dendrogram(sch.linkage(features, method="average"))
2  hc = AgglomerativeClustering(n_clusters=2, affinity="euclidean", linkage="average")
3  y_hc = hc.fit_predict(features)

```



(e) Les scores des méthodes de liaison.



(f) Le dendrogramme du dataset utilisant la méthode de liaison « average ».

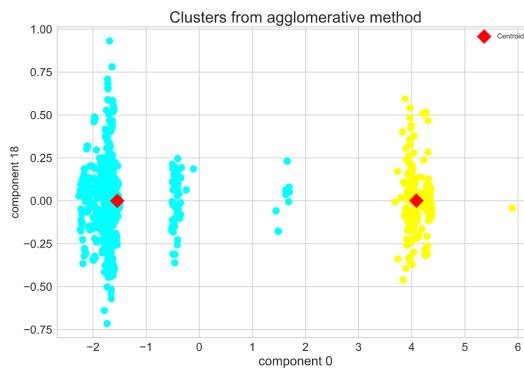


FIGURE 6.11 : Clusters de la méthode agglomérante

6.3.4.4 Clustering Soft

Le clustering flou C-Means est une approche de clustering soft, où chaque point de données se voit attribuer une probabilité ou un score de probabilité d'appartenir à un cluster. C'est un algorithme de clustering non supervisé qui dépend d'un paramètre m qui correspond au degré de flou des clusters. Le bon choix est d'utiliser $m = 2.0$ (Hathaway et Bezdek 2001). L'algorithme du clustering fuzzy c-means est :

La méthode `cmeans` de Scikit-Fuzzy [136] (qui est une collection d'algorithmes de logique floue écrits en python) permet d'implémenter l'algorithme FCM sur un jeu de données tout en ajustant certains paramètres. Cette méthode a été utiliser pour l'implémentation comme suit :

```
1  cntr, u, u0, d, jm, p, fpc = fuzz.cluster.cmeans(data=features.T,c=2,m=2, error=0.005,
maxiter=1000, init=None, seed=2021)
```

Où, les paramètres et les objets retournées par la méthode `fuzz.cluster.cmeans()` sont :

Paramètres :

Algorithm 6 Fuzzy c-means algorithm.

- 1: Initialize $U = [u_{ij}]$ matrix, $U^{(0)}$
- 2: At k-step : calculate the centers vectors $C^{(k)} = [c_j]$ with $U^{(k)}$

$$c_j = \frac{\sum_{i=1}^N u_{ij}^m \cdot x_i}{\sum_{i=1}^N u_{ij}^m} \quad (6.3)$$

- 3: Update $U^{(k)}$, $U^{(k+1)}$

$$u_{ij} = \frac{1}{\sum_{k=1}^C \left(\frac{\|x_i - c_j\|}{\|x_i - c_k\|} \right)^{\frac{2}{m-1}}} \quad (6.4)$$

- 4: If $\|U^{(k+1)} - U^{(k)}\| < \epsilon$ the STOP ; otherwise return to step 2.

- data : (tableau 2d, taille (S,N)), les données à regrouper. N est le nombre d'ensembles de données ; S est le nombre d'entités dans chaque vecteur d'échantillon.
- c : (int), Nombre de clusters souhaité.
- m : (float), Exponentiation du tableau appliquée à la fonction d'appartenance u_{old} à chaque itération, où $U_{new} = u_{old} * *m$.
- error : (float), critère d'arrêt ; arrêter tôt si la norme de $(u[p] - u[p - 1]) < erreur$.
- maxiter : (int), Nombre maximal d'itérations autorisées.
- init : (tableau 2d, taille (S, N)), Matrice c-partitionnée floue initiale. Si aucun n'est fourni, l'algorithme est initialisé de manière aléatoire.
- seed : (int), Si fourni, définit une valeur de départ aléatoire d' `init`. Aucun effet si `init` est fourni. Principalement à des fins de débogage/test.

Retour :

- cntr : (tableau 2d, taille (S, c)), les centres de cluster. Données pour chaque centre le long de chaque caractéristique fournie pour chaque cluster (des `c` clusters demandés).
- u : (tableau 2d, (S, N)), Matrice c-partitionnée floue finale.
- u0 : (tableau 2d, (S, N)), Estimation initiale à la matrice c-partitionnée floue.
- d : (tableau 2d, (S, N)), Matrice de distance euclidienne finale.
- jm : (tableau 1d, longueur P), Historique de la fonction objective.

- p : (int), Nombre d'itérations exécutées.
- fpc : (float), le coefficient de partition floue final.

La figure 6.12 affiche les clusters obtenus par la méthode FCM et le degré d'appartenance des points de données aux clusters.

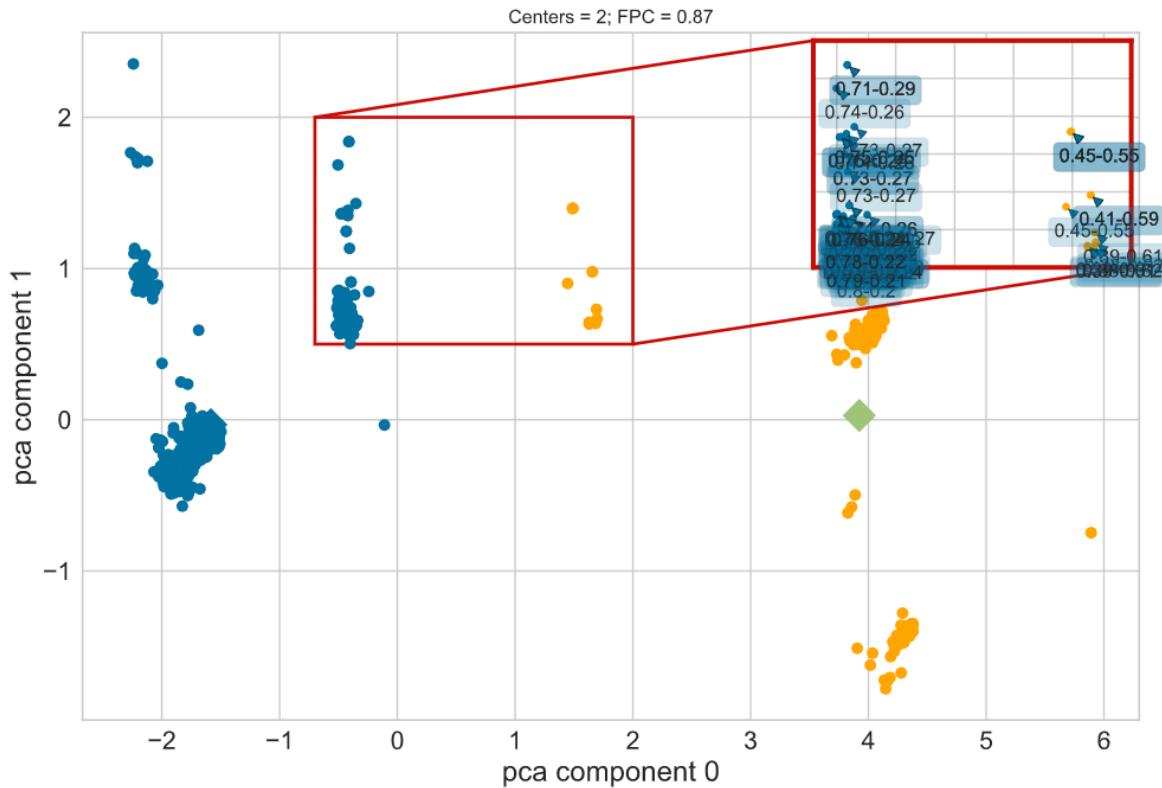


FIGURE 6.12 : Visualisation des clusters de la méthode fuzzy et le degré d'appartenance des items à chaque cluster.

6.3.4.5 Validation du clustering

Kmeans clustering					
Nombre de clusters	Indice de validité				Nombre d'items dans un cluster
	Indice de Dunn	Indice Calinski Harabasz	Indice Davies-Bouldin	Silhouette score	
2	0.43	2414	0.58	0.62	308 ~ 776
3	0.51	1452	1.05	0.44	68 ~ 719
4	0.50	1126	1.21	0.38	68 ~ 719
5	0.47	937	1.27	0.306	52 ~ 667
6	0.46	810	1.30	0.295	43 ~ 624
7	0.45	711	1.33	0.285	33 ~ 591
8	0.45	626	1.29	0.289	7 ~ 591
9	0.38	566	1.33	0.26	7 ~ 564
10	0.37	527	1.47	0.24	7 ~ 564
Agglomérative clustering					
Nombre de clusters	Indice de validité				Nombre d'items dans un cluster
	Indice de Dunn	Indice Calinski Harabasz	Indice Davies-Bouldin	Silhouette score	
2	0.57	2368	0.58	0.62	297 ~ 787
3	0.51	1452	1.05	0.44	68 ~ 719
4	0.48	974	1.054	0.43	2 ~ 717
5	0.48	733	0.937	0.41	1 ~ 716
6	0.48	594	0.933	0.38	1 ~ 716
7	0.48	506	0.9366	0.354	1 ~ 716
8	0.48	435	0.80	0.352	1 ~ 716
9	0.56	444	0.89	0.34	1 ~ 716
10	0.52	438	0.95	0.30	1 ~ 666
Fuzzy clustering					
Nombre de clusters	Indice de validité				Nombre d'items dans un cluster
	FPC	Indice Calinski Harabasz	Indice Davies-Bouldin	Silhouette score	
2	0.865	2413.6	0.59	0.62	308 ~ 776
3	0.54	1378	1.27	0.40	71 ~ 705
4	0.45	1394	1.02	0.53	1 ~ 776
5	0.35	708.51	2.30	-0.008	3 ~ 757
6	0.30	697	3.06	0.024	1 ~ 766
7	0.26	486.75	4.2	0.011	6 ~ 735
8	0.23	481.55	4.29	-0.022	1 ~ 685
9	0.20	666	2.57	0.21	1 ~ 680
10	0.18	408	3.72	-0.0094	1 ~ 726

TABLE 6.8 : Résultats des indices de validité du clustering.

6.4 Conclusion

Ce chapitre se résume comme un pipeline pour travailler avec les données éducatives. D'abord une analyse des réponses des apprenants en utilisant le workflow bayésien avec des modèles de la théorie des réponses aux items et une classification hard et soft des items basé sur la similarité. La théorie des réponses aux items a permis d'analyse les scores des apprenants afin d'estimer les propriétés métriques des items et le niveau des apprenants par rapport au trait latent considéré, ensuite les valeurs des paramètres des modèles ont été utilisé pour prédire la probabilité de réussite à un item. Les prédictions faites par les modèles permettent d'examiner la qualité de l'échelle de mesure qui a été utilisé par le système où les données ont été collecter et montrent la qualité du jeu de données, c'est-à-dire si le jeu de données peut être utilisé pour une étude. La classification hard et soft des items basé sur la similarité est l'étude qui a été réalisé avec le jeu de données.

Conclusion Générale et Perspectives

Conclusion Générale

Dans ce mémoire, nous avons travaillé sur l'intégration des méthodes du data mining dans les systèmes d'apprentissage qui ont pour but d'optimiser les méthodes d'apprentissage, de construire et d'évaluer des échelles de mesure, de gérer un grand nombre d'item, ainsi que les techniques de regroupement des items en composants de connaissances.

Les paramètres estimés à l'aide de l'inférence bayésienne pour les modèles de la théorie de la réponse aux items peuvent être utilisé pour évaluer la qualité des échelles de mesures et de prédire les scores des apprenants avant d'effectuer la collecte de données pour faire un regroupement des items selon leurs similitudes.

Dans ce travail, nous avons eu deux contributions

Dans la première contribution, nous avons proposé une étape d'analyse et de validation des données des systèmes éducatif à l'aide de l'inférence bayésienne et des modèles de la théorie de la réponse aux items. Cette étape a été utilisé comme un workflow bayésien, c'est-à-dire un processus itératif jusqu'à l'obtention d'un modèle acceptable ou provisoirement acceptable. Les paramètres des modèles IRT ainsi obtenu décrivent les compétences des apprenants sur un continuum de trait latent d'une part et d'autre part les propriétés de l'item notamment, sa difficulté, son pouvoir de discrimination, le rôle que la "chance" (réponses "au hasard") peut jouer dans certains cas. Une fois l'obtention d'un modèle acceptable ou provisoirement acceptable, les valeurs de ces paramètres peuvent être utilisées pour prédire la probabilité qu'un apprenant obtient une réponse correcte. Enfin les valeurs des métriques entre les données observées et ceux prédicts par les modèles peuvent être utilisées pour comparer les modèles IRT avec des spécifications alternatives, ou pour éclairer une décision sur l'intérêt de collecter ce type de données.

Dans la deuxième contribution, un regroupement des items en composante de connaissance a été appliquer avec une méthode de clustering partitionnel (à l'aide de l'algorithme K-means), hiérarchique (à l'aide de l'algorithme hiérarchique agglomérative) et floue (à l'aide de l'algorithme c-means). Ces méthodes de clustering ont utilisé la matrice de similarité Item-Item

CONCLUSION GÉNÉRALE ET PERSPECTIVES

calculer à partir du nombre correct et incorrect des réponses avec aide et sans aide, et le coefficient de kappa pour le calcul de degré de similarité.

Perspectives

A la fin de ce projet de fin d'études, nous avons dégagé les perspectives suivantes à développer dans l'avenir pour une amélioration de ce travail :

- Extension des modèles IRT mentionnés dans ce mémoire aux modèles à plusieurs niveaux de sorte qu'ils prennent en compte les variations de capacités entre les unités de groupe telles que les écoles, ainsi qu'au sein des unités de groupe. Ces modèles à plusieurs niveaux distingueront ainsi les capacités au niveau individuel(étudiants), au niveau du groupe (classes), ou les deux.
- Estimations des probabilités de réussite à un item avec l'algorithme Bayesian Knowledge Tracing (BKT).
- Appliquer d'autre coefficient de similarité comme Yule, Fisher, Sokal et aussi le critère de calcule de similitude entre items.

ANNEXE A

Les modèles IRT en code Stan.

Listing A.1 : Code Stan pour le modèle de Rasch

```
1  _1pl_model = """
2  data {
3      // numbers of things
4      int<lower=1> N;    // number of observations
5      int<lower=1> I;    // items ,   number of questions
6      int<lower=1> S;    // subjects ,   number of users
7      // data
8      int<lower=1,upper=I> item[N];
9      int<lower=1,upper=S> subject[N];
10     int<lower=0,upper=1> grade[N];
11     // data for posterior prediction using new data also used for Cross-validation
12     int<lower=1> N_new;
13     int<lower=1,upper=I> item_new[N_new];
14     int<lower=1,upper=S> subject_new[N_new];
15 }
16 parameters {
17     // parameters
18     real ability[S];           // alpha: ability of student
19     real difficulty[I];        // beta: difficulty of question
20     real delta;                // mean student ability
21 }
22 model {
23     ability ~ normal(0,1);
24     difficulty ~ normal(0,1);
25     delta ~ normal(0.75,1);
26     for(n in 1:N)
27         grade[n] ~ bernoulli_logit(ability[subject[n]] - difficulty[item[n]] + delta);
28 }
29 generated quantities {
30     int<lower=0,upper=1> y_pred[N];
31     int<lower=0,upper=1> yNew_pred[N_new];
32     vector[N] log_lik;
33
34     for(n in 1:N){
```

ANNEXE A. LES MODÈLES IRT EN CODE STAN.

```
35     y_pred[n] = bernoulli_logit_rng(ability[subject[n]] - difficulty[item[n]] +
36                                         delta);
37     log_lik[n] = bernoulli_logit_lpmf( grade[n] | ability[subject[n]] - difficulty[
38                                         item[n]]
39                                         + delta);
40     }
41   }
42   """
43 
```

ANNEXE A. LES MODÈLES IRT EN CODE STAN.

Listing A.2 : Code Stan pour 2PL

```
1      _2pl_model = """
2      data {
3          // numbers of things
4          int<lower=1> N;    // number of observations
5          int<lower=1> I;    // items , number of questions
6          int<lower=1> S;    // subjects , number of users
7          // data
8          int<lower=1,upper=I> item[N];
9          int<lower=1,upper=S> subject[N];
10         int<lower=0,upper=1> grade[N];
11         // data for posterior prediction using new data also used for Cross-validation
12         int<lower=1> N_new;
13         int<lower=1,upper=I> item_new[N_new];
14         int<lower=1,upper=S> subject_new[N_new];
15     }
16     parameters {
17         // parameters
18         real ability[S];           // alpha: ability of student
19         real difficulty[I];       // beta: difficulty of question
20         vector<lower=0>[I] discrimination;   // discrimination of question
21         real delta;                // mean student ability
22     }
23     model {
24         ability ~ normal(0,1);
25         difficulty ~ normal(0,1);
26         discrimination ~ lognormal(0,1);
27         delta ~ normal(0.75,1);
28         grade ~ bernoulli_logit(discrimination[item] .* (ability[subject] - (difficulty[item] + delta)));
29     }
30     generated quantities {
31         int<lower=0,upper=1> y_pred[N];
32         int<lower=0,upper=1> yNew_pred[N_new];
33         vector[N] log_liks;
34
35         for (n in 1:N) {
36             y_pred[n] = bernoulli_logit_rng(discrimination[item[n]] * (ability[subject[n]] - (difficulty[item[n]] + delta)));
37             log_liks[n] = bernoulli_logit_lpmf(grade[n] | discrimination[item[n]] * (ability[subject[n]] - (difficulty[item[n]] + delta)));
38         }
39         for (n in 1:N_new) {
40             yNew_pred[n] = bernoulli_logit_rng(discrimination[item[n]] * (ability[subject_new[n]] - (difficulty[item_new[n]] + delta)));
41         }
42     }
43     """

```

ANNEXE A. LES MODÈLES IRT EN CODE STAN.

Listing A.3 : Code Stan pour 3PL

```

1      _3pl_model = """
2      data {
3          // numbers of things
4          int<lower=1> N;    // number of observations
5          int<lower=1> I;    // items , number of questions
6          int<lower=1> S;    // subjects , number of users
7          // data
8          int<lower=1,upper=I> item[N];
9          int<lower=1,upper=S> subject[N];
10         int<lower=0,upper=1> grade[N];
11         // data for posterior prediction using new data also used for Cross-validation
12         int<lower=1> N_new;
13         int<lower=1,upper=I> item_new[N_new];
14         int<lower=1,upper=S> subject_new[N_new];
15     }
16     parameters {
17         // parameters
18         real ability[S];           // alpha: ability of student
19         real difficulty[I];       // beta: difficulty of question
20         vector<lower=0>[I] discrimination;   // discrimination of question
21         vector<lower=0,upper=1>[I] guessing;
22         real delta;                // mean student ability
23     }
24     model {
25         ability ~ normal(0,1);
26         difficulty ~ normal(0,1);
27         discrimination ~ lognormal(0,1);
28         guessing ~ beta(5,17);
29         delta ~ normal(0.75,1);
30         grade ~ bernoulli(guessing[item] + ((1-guessing[item]).*(inv_logit(discrimination[
31             item] .* (ability[subject] - (difficulty[item] + delta)))));
32     }
33     generated quantities {
34         int<lower=0,upper=1> y_pred[N];
35         int<lower=0,upper=1> yNew_pred[N_new];
36         vector[N] log_lik;
37
38         for (n in 1:N) {
39             y_pred[n] = bernoulli_rng(guessing[item[n]] + ((1-guessing[item[n]]).*
40                 inv_logit(discrimination[item[n]].* (ability[subject[n]] - (difficulty[
41                     item[n]] + delta))));}
42             log_lik[n] = bernoulli_lpmf( grade[n] | guessing[item[n]] + ((1-guessing[item[n]]).*
43                 inv_logit(discrimination[item[n]].* (ability[subject[n]] - (difficulty[
44                     item[n]] + delta))));}
45         }
46     }
47 }
```

"""

Listing A.4 : Code Stan pour le modèle de Rasch avec cross-validation

```

1      _1pl_cross_val_model = """
2      functions {
3          int[] permutation_rng(int N) {
4              int y[N];
5              for (n in 1:N)
6                  y[n] = n;
7              vector[N] theta = rep_vector(1.0 / N, N);
8              for (n in 1:N){
9                  int i = categorical_rng(theta);
10                 int temp = y[n];
11                 y[n] = y[i];
12                 y[i] = temp;
13             }
14             return y;
15         }
16     }
17
18     data {
19         // numbers of things
20
21         int<lower=1> N;    // number of observations
22         int<lower = 0, upper = N> N_test;
23         int<lower=1> I;    // items ,   number of questions
24         int<lower=1> S;    // subjects ,   number of users
25
26         // data
27
28         int<lower=1,upper=I> item[N];
29         int<lower=1,upper=S> subject[N];
30         int<lower=0,upper=1> grade[N];
31     }
32     transformed data {
33         int N_train = N - N_test;
34         int permutation[N] = permutation_rng(N);
35         // train
36         int item_train[N_train] = item[permutation[1 : N_train]];
37         int subject_train[N_train] = subject[permutation[1 : N_train]];
38         int grade_train[N_train] = grade[permutation[1 : N_train]];
39         int s_train = size(subject_train);
40         int i_train = size(item_train);
41
42
43         // test
44         int item_test[N_test] = item[permutation[N_train + 1 : N]];
45         int subject_test[N_test] = subject[permutation[N_train + 1 : N]];
46         int grade_test[N_test] = grade[permutation[N_train + 1 : N]];
47     }
48     parameters {

```

ANNEXE A. LES MODÈLES IRT EN CODE STAN.

```
49      // parameters
50      real ability[s_train];           // alpha: ability od student
51      real difficulty[i_train];        // beta: difficulty of question
52      real delta;                   // man student ability
53
54  }
55  model {
56
57      ability ~ normal(0,1);
58      difficulty ~ normal(0,1);
59      delta ~ normal(0.75,1);
60
61      for (n in 1:N_train) {
62          grade_train[n] ~ beroulli_logit(ability[subject_train[n]] - difficulty[
63              item_train[n]] + delta);
64      }
65  }
66  generated quantities {
67      int<lower=0,upper=1> grade_pred[N_test];
68      int<lower=0,upper=1> grade_observed[N_test];
69      grade_observed = grade_test;
70
71      for (n in 1:N_test) {
72          grade_pred[n] = beroulli_logit_rng(ability[subject_test[n]] - difficulty[item_test
73              [n]] + delta);
74      }
75  """
```

ANNEXE B

Les distributions des modèles.

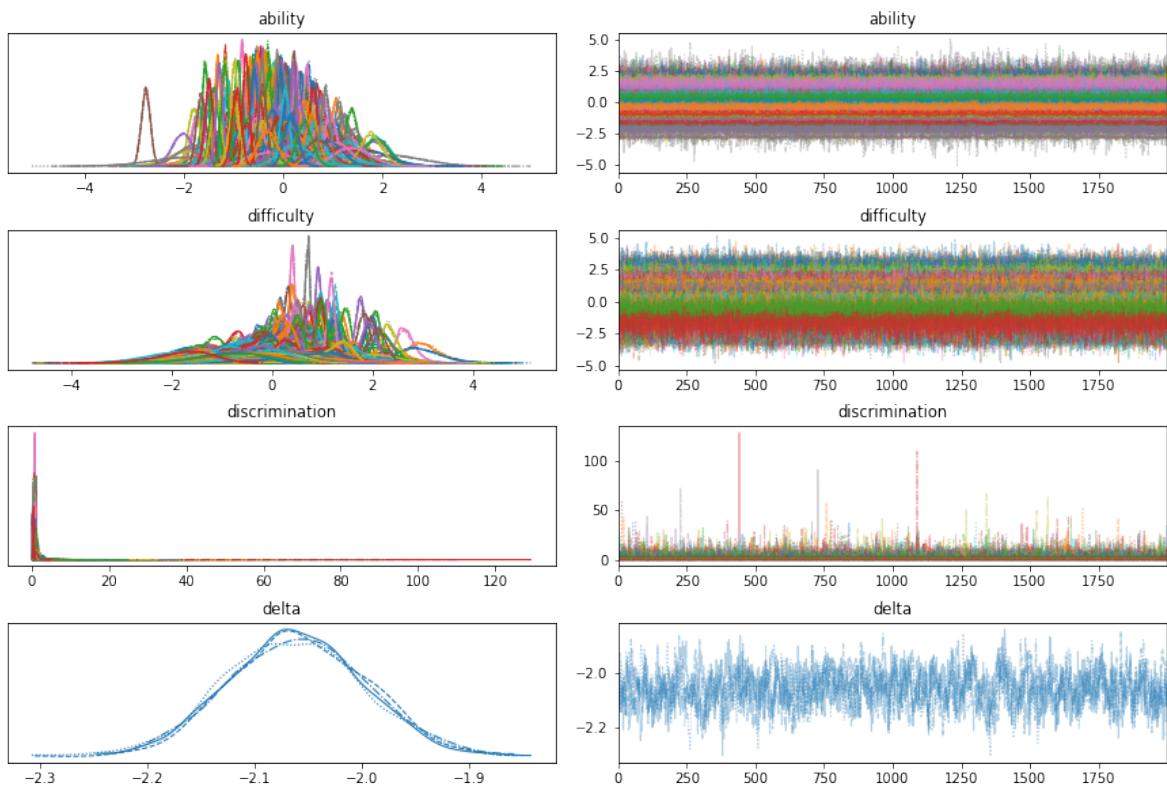


FIGURE B.1 : Les distributions postérieures du modèle logistique à deux paramètres.

ANNEXE B. LES DISTRIBUTIONS DES MODÈLES.

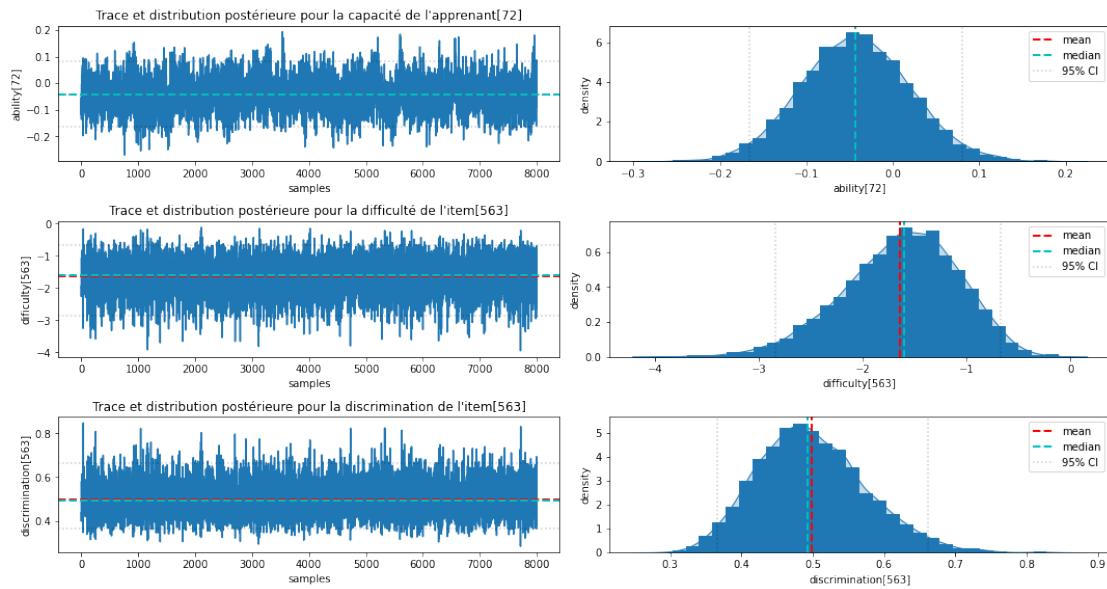


FIGURE B.2 : Distribution postérieure de la capacité de l'apprenant[72], de la difficulté de l'item[563] et de la discrimination de l'item[563] du modèle 2PL.

ANNEXE B. LES DISTRIBUTIONS DES MODÈLES.

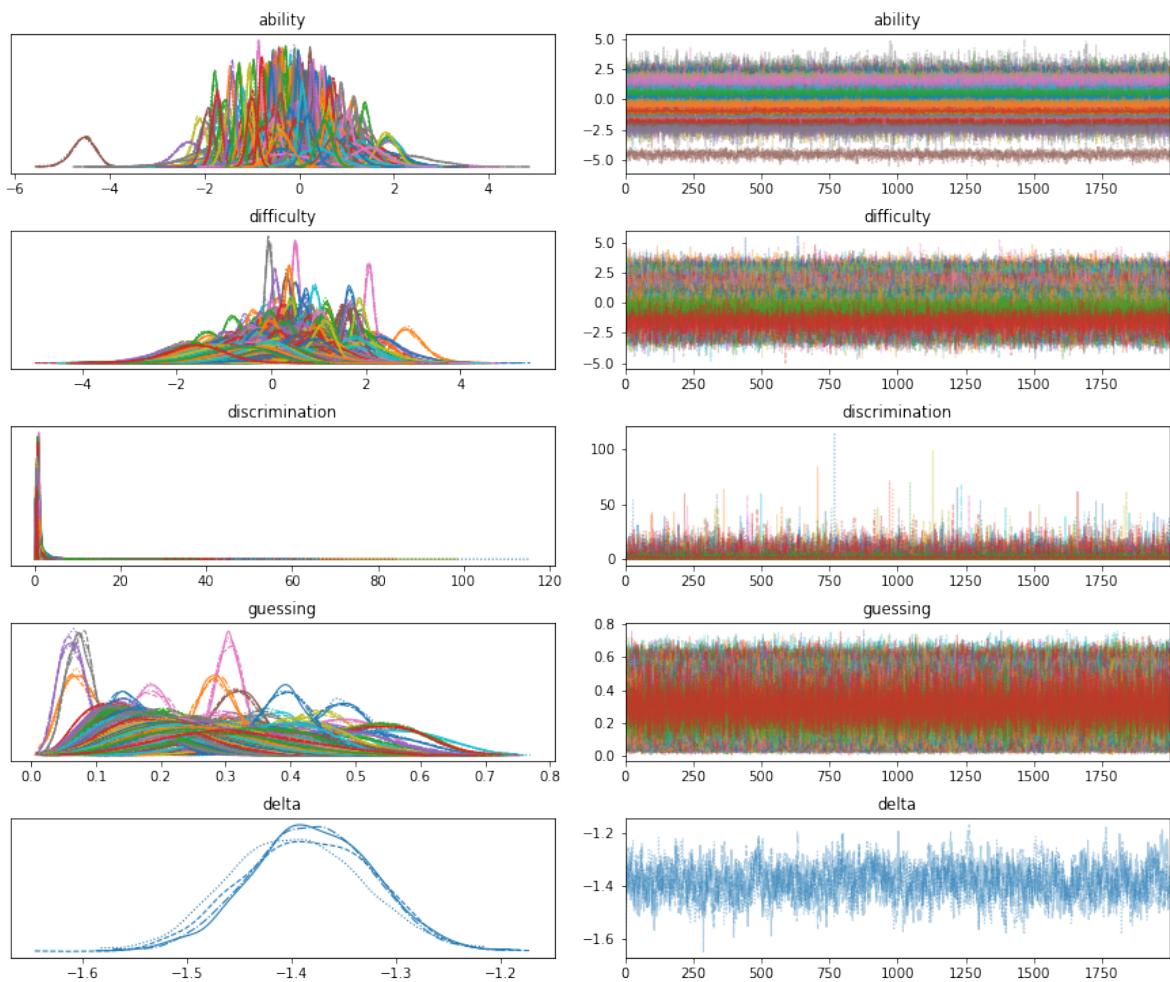


FIGURE B.3 : Les distributions postérieures du modèle logistique à trois paramètres.

ANNEXE B. LES DISTRIBUTIONS DES MODÈLES.

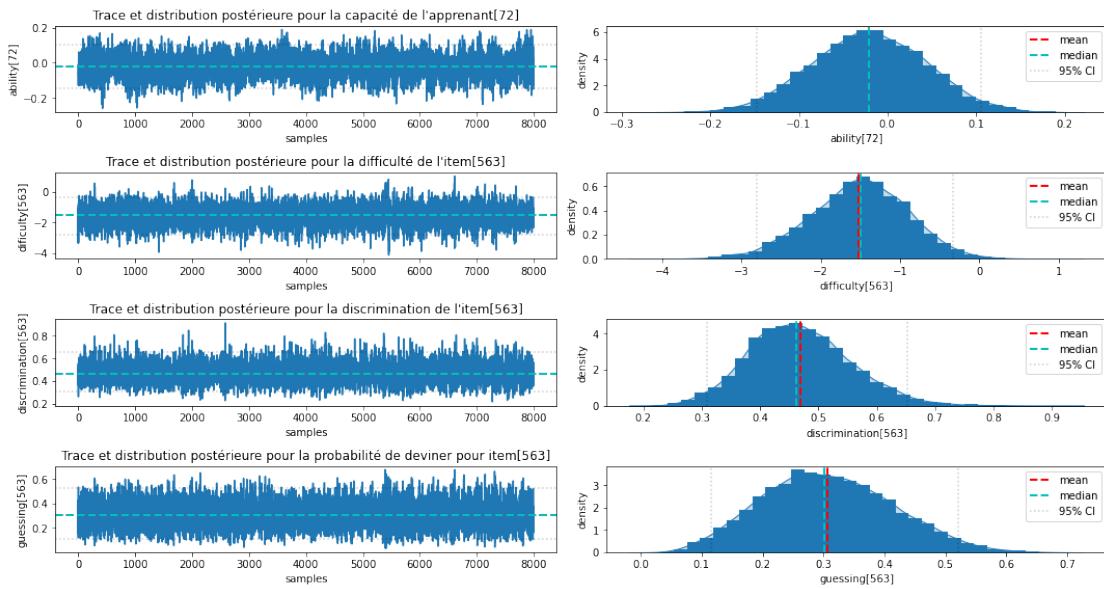


FIGURE B.4 : Distribution postérieure de la capacité de l'apprenant[72], de la difficulté de l'item[563], de la discrimination de l'item[563] et de la chance de deviner la réponse correcte de l'item[563] du modèle 2PL.

Bibliographie

- [1] R. Entezari, *Bayesian Computations via MCMC, with Applications to Big Data and Spatial Data*. PhD thesis, University of Toronto (Canada), 2018.
- [2] S. P. Meyn and R. L. Tweedie, *Markov chains and stochastic stability*. Springer Science & Business Media, 2012.
- [3] J. S. Rosenthal, *First Look At Rigorous Probability Theory*, A. World Scientific Publishing Company, 2006.
- [4] A. Gelman, A. Vehtari, D. Simpson, C. C. Margossian, B. Carpenter, Y. Yao, L. Kennedy, J. Gabry, P.-C. Bürkner, and M. Modrák, “Bayesian workflow,” *arXiv preprint arXiv :2011.01808*, 2020.
- [5] C. D. Nye, S.-H. Joo, B. Zhang, and S. Stark, “Advancing and evaluating irt model data fit indices in organizational research,” *Organizational Research Methods*, vol. 23, no. 3, pp. 457–486, 2020.
- [6] J. Han and M. Kamber, *Data Mining : Concepts and Techniques*. 01 2000.
- [7] P. Cabena, P. Hadjinian, R. Stadler, J. Verhees, and A. Zanasi, *Discovering data mining : from concept to implementation*. Prentice-Hall, Inc., 1998.
- [8] M. Veyssières and R. E. Plant, “Identification of vegetation state and transition domains in california’s hardwood rangelands,” *University of California*, vol. 101, 1998.
- [9] C. Romero and S. Ventura, “Educational data mining : A survey from 1995 to 2005,” *Expert Systems with Applications*, vol. 33, pp. 135–146, 07 2007.
- [10] C. Buche, *Un système tutoriel intelligent et adaptatif pour l’apprentissage de compétences en environnement virtuel de formation*. Theses, Université de Bretagne occidentale - Brest, Nov. 2005.
- [11] C. Romero, S. Ventura, M. Pechenizkiy, and R. Baker, *Handbook of Educational Data Mining*. 10 2010.
- [12] R. Baker and K. Yacef, “The state of educational data mining in 2009 : A review and future visions,” *Journal of Educational Data Mining*, vol. 1, pp. 3–17, 01 2009.
- [13] S. Bendjebbar, *Utilisation des Techniques de Data Mining pour la Modélisation des Tuteurs*. Theses, Université 8 mai 1945 Guelma, 2016.
- [14] O. Scheuer and B. M. McLaren, *Educational Data Mining*, pp. 1075–1079. Boston, MA : Springer US, 2012.

- [15] L. Nguyen and P. Do, "Learner model in adaptive learning," *Proceedings of World Academy of Science, Engineering and Technology*, vol. 45, pp. 396–401, 01 2008.
- [16] R. Winkels, "User modelling in help systems.," vol. 438, pp. 184–193, 06 1990.
- [17] A. Paiva and J. Self, "Tagus — a user and learner modeling workbench," *User Modeling and User-Adapted Interaction*, vol. 4, pp. 197–226, 09 1994.
- [18] S. Greer, "Psycho-oncology : Its aims, achievements and future tasks," *Psycho-Oncology*, vol. 3, pp. 87 – 101, 07 1994.
- [19] P. Brusilovsky, "Student model centered architecture for intelligent learning environments," 01 1994.
- [20] I. Goldstein and B. Roberts, "Nudge, a knowledge-based scheduling program," *Proceedings of IJCAI-77*, 01 1977.
- [21] J. Elster, *The Multiple Self*. Cambridge University Press, 1987.
- [22] R. Nkambou, "Modélisation des connaissances de la matière dans un système tutoriel intelligent : modèles, outils et applications /," 06 2021.
- [23] E. Ragnemalm, "Collaborative dialogue with a learning companion as a source of information on student reasoning.," pp. 650–658, 01 1996.
- [24] P. Holt, S. Dubs, M. Jones, and J. Greer, "The state of student modelling" in student modelling : The key to individualized knowledge-based inst," 06 2021.
- [25] D. Fragne, "Proposition de l'architecture de l'agent gestionnaire du modèle de l'apprenant dans un système tuteur multi-agents en apprentissage de la lecture : contribution au projet amical," 12 2009.
- [26] J. Gobert, M. Pedro, J. Raziuddin, and R. Baker, "From log files to assessment metrics : Measuring students' science inquiry skills using educational data mining," *Journal of the Learning Sciences*, vol. 22, pp. 521–563, 10 2013.
- [27] E. Millán, T. Loboda, and J. L. P. de-la Cruz, "Bayesian networks for student model engineering," vol. 55, no. 4, pp. 1663–1683, 2010.
- [28] J. Collins, J. Greer, and S. Huang, "Adaptive assessment using granularity hierarchies and bayesian nets," 02 1970.
- [29] J. M. P. Tchétagni and R. Nkambou, "Hierarchical representation and evaluation of the student in an intelligent tutoring system," in *Intelligent Tutoring Systems* (S. A. Cerri, G. Gouardères, and F. Paraguaçu, eds.), (Berlin, Heidelberg), pp. 708–717, Springer Berlin Heidelberg, 2002.
- [30] C. Carmona and R. Conejo, "A learner model in a distributed environment," in *Adaptive Hypermedia and Adaptive Web-Based Systems* (P. M. E. De Bra and W. Nejdl, eds.), (Berlin, Heidelberg), pp. 353–359, Springer Berlin Heidelberg, 2004.
- [31] Z. A. Pardos, N. T. Heffernan, B. Anderson, and C. L. Heffernan, "The effect of model granularity on student performance prediction using bayesian networks," in *User Modeling 2007* (C. Conati, K. McCoy, and G. Paliouras, eds.), (Berlin, Heidelberg), pp. 435–439, Springer Berlin Heidelberg, 2007.

- [32] J. Reye, “Student modelling based on belief networks,” *International Journal of Artificial Intelligence in Education*, vol. 14, pp. 63–96, 2004.
- [33] C. Carmona, E. Millán, J. L. Pérez-de-la Cruz, M. Trella, and R. Conejo, “Introducing prerequisite relations in a multi-layered bayesian student model,” in *User Modeling 2005* (L. Ardissono, P. Brna, and A. Mitrovic, eds.), (Berlin, Heidelberg), pp. 347–356, Springer Berlin Heidelberg, 2005.
- [34] I. Goldin, r. scheines, and e. silver, “Discovering prerequisite relationships among knowledge components,” 01 2014.
- [35] K. Bollen, “Structural equations with latent variables, new york,” *Wiley Interscience*, 01 1989.
- [36] R. Pelánek, T. Effenberger, M. Vaněk, V. Sassmann, and D. Gmíterko, “Measuring item similarity in introductory programming,” in *Proceedings of the Fifth Annual ACM Conference on Learning at Scale*, pp. 1–4, 2018.
- [37] Wikipedia contributors, “knowledge : definition of knowledge in oxford dictionary (american english) (us),” 2004. Archived from the original on 14 July 2010.
- [38] B. S. Bloom *et al.*, “Taxonomy of educational objectives. vol. 1 : Cognitive domain,” *New York : McKay*, vol. 20, no. 24, p. 1, 1956.
- [39] K. R. Koedinger, A. T. Corbett, and C. Perfetti, “The knowledge-learning-instruction framework : Bridging the science-practice chasm to enhance robust student learning,” *Cognitive science*, vol. 36, no. 5, pp. 757–798, 2012.
- [40] T. Nazaretsky, S. Hershkovitz, and G. Alexandron, “Kappa learning : A new method for measuring similarity between educational items using performance data,” *arXiv preprint arXiv :1812.08390*, 2018.
- [41] K. VanLehn, “The behavior of tutoring systems,” *International journal of artificial intelligence in education*, vol. 16, no. 3, pp. 227–265, 2006.
- [42] N. M. Chang, *Learning to discriminate and generalize through problem comparisons*. PhD thesis, Carnegie Mellon University, 2006.
- [43] M. C. Desmarais, “Mapping question items to skills with non-negative matrix factorization,” *ACM SIGKDD Explorations Newsletter*, vol. 13, no. 2, pp. 30–36, 2012.
- [44] R. S. Baker, K. Yacef, *et al.*, “The state of educational data mining in 2009 : A review and future visions,” *Journal of educational data mining*, vol. 1, no. 1, pp. 3–17, 2009.
- [45] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, “Item-based collaborative filtering recommendation algorithms,” *Proceedings of ACM World Wide Web Conference*, vol. 1, 08 2001.
- [46] J. S. Breese, D. Heckerman, and C. Kadie, “Empirical analysis of predictive algorithms for collaborative filtering,” *arXiv preprint arXiv :1301.7363*, 2013.
- [47] Y. Koren and R. Bell, “Advances in collaborative filtering. recommender systems handbook, francesco ricci, lior rokach, bracha shapira, paul b. kantor editors, chapter 5,” 2011.

- [48] T. Barnes, “The q-matrix method : Mining student response data for knowledge,” in *American Association for Artificial Intelligence 2005 Educational Data Mining Workshop*, pp. 1–8, Pittsburgh, PA : AAAI Press, 2005.
- [49] S. Nithya, A. Srinivasan, M. Senthilkumar, *et al.*, “Calculating the user-item similarity using pearson’s and cosine correlation,” in *2017 International Conference on Trends in Electronics and Informatics (ICEI)*, pp. 1000–1004, IEEE, 2017.
- [50] M. Aitkin and N. Longford, “Statistical modelling issues in school effectiveness studies,” *Journal of the Royal Statistical Society : Series A (General)*, vol. 149, no. 1, pp. 1–26, 1986.
- [51] R. J. Mislevy, “Evidence and inference in educational assessment,” *Psychometrika*, vol. 59, no. 4, pp. 439–483, 1994.
- [52] C. L. Azevedo, H. Bolfarine, and D. F. Andrade, “Bayesian inference for a skew-normal irt model under the centred parameterization,” *Computational Statistics & Data Analysis*, vol. 55, no. 1, pp. 353–365, 2011.
- [53] R. A. Fisher, “On the mathematical foundations of theoretical statistics,” *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, vol. 222, no. 594-604, pp. 309–368, 1922.
- [54] R. M. Neal, *Probabilistic inference using Markov chain Monte Carlo methods*. Department of Computer Science, University of Toronto Toronto, ON, Canada, 1993.
- [55] Y. G. Gbedo, *Les techniques Monte Carlo par chaînes de Markov appliquées à la détermination des distributions de partons*. PhD thesis, Université Grenoble Alpes, 2017.
- [56] A. E. Gelfand and A. F. Smith, “Sampling-based approaches to calculating marginal densities,” *Journal of the American statistical association*, vol. 85, no. 410, pp. 398–409, 1990.
- [57] C. P. Robert and W. Changye, “Markov chain monte carlo methods, a survey with some frequent misunderstandings,” *arXiv preprint arXiv:2001.06249*, 2020.
- [58] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, “Equation of state calculations by fast computing machines,” *The journal of chemical physics*, vol. 21, no. 6, pp. 1087–1092, 1953.
- [59] W. K. Hastings, “Monte carlo sampling methods using markov chains and their applications,” 1970.
- [60] R. M. Neal *et al.*, “Mcmc using hamiltonian dynamics,” *Handbook of markov chain monte carlo*, vol. 2, no. 11, p. 2, 2011.
- [61] M. Betancourt and M. Girolami, “Hamiltonian monte carlo for hierarchical models,” *Current trends in Bayesian methodology with applications*, vol. 79, no. 30, pp. 2–4, 2015.
- [62] A. Gelman, W. R. Gilks, and G. O. Roberts, “Weak convergence and optimal scaling of random walk metropolis algorithms,” *The annals of applied probability*, vol. 7, no. 1, pp. 110–120, 1997.

- [63] C. Wu, J. Stoehr, and C. P. Robert, “Faster hamiltonian monte carlo by learning leap-frog scale,” *arXiv preprint arXiv :1810.04449*, 2018.
- [64] M. D. Hoffman, A. Gelman, *et al.*, “The no-u-turn sampler : adaptively setting path lengths in hamiltonian monte carlo.,” *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1593–1623, 2014.
- [65] M. D. Hoffman and A. Gelman, “The no-u-turn sampler : Adaptively setting path lengths in hamiltonian monte carlo. arxiv preprint arxiv : 1111.4246,” 2011.
- [66] D. Spiegelhalter, N. G. Best, B. P. Carlin, and A. van der Linde, “Bayesian measures of model complexity and fit,” *Quality control and applied statistics*, vol. 48, no. 4, pp. 431–432, 2003.
- [67] R. E. Kass and A. E. Raftery, “Bayes factors,” *Journal of the american statistical association*, vol. 90, no. 430, pp. 773–795, 1995.
- [68] A. O’Hagan, “Fractional bayes factors for model comparison,” *Journal of the Royal Statistical Society : Series B (Methodological)*, vol. 57, no. 1, pp. 99–118, 1995.
- [69] S. Watanabe and M. Opper, “Asymptotic equivalence of bayes cross validation and widely applicable information criterion in singular learning theory.,” *Journal of machine learning research*, vol. 11, no. 12, 2010.
- [70] S. Geisser and W. F. Eddy, “A predictive approach to model selection,” *Journal of the American Statistical Association*, vol. 74, no. 365, pp. 153–160, 1979.
- [71] “Item response theory - wikipedia.”
- [72] Y. Noël, “Introduction aux modèles de réponse à l’item (mri),” Jan 2019.
- [73] D. Lunn, D. Spiegelhalter, A. Thomas, and N. Best, “The bugs project : Evolution, critique and future directions,” *Statistics in medicine*, vol. 28, no. 25, pp. 3049–3067, 2009.
- [74] M. Plummer *et al.*, “Jags : A program for analysis of bayesian graphical models using gibbs sampling,” in *Proceedings of the 3rd international workshop on distributed statistical computing*, vol. 124, pp. 1–10, Vienna, Austria., 2003.
- [75] “Stan documentation.” <https://mc-stan.org>. accessed : 09-July-2021.
- [76] A. Gelman and J. Hill, *Data Analysis Using Regression And Multilevel/Hierarchical Models*, vol. 3. 11 2006.
- [77] Z. Huang, “Extensions to the k-means algorithm for clustering large data sets with categorical values,” *Data mining and knowledge discovery*, vol. 2, no. 3, pp. 283–304, 1998.
- [78] K. Sasirekha and P. Baby, “Agglomerative hierarchical clustering algorithm-a,” *International Journal of Scientific and Research Publications*, vol. 83, p. 83, 2013.
- [79] M. Yang, “A survey of hierarchical clustering,” *Mathl. Comput. Modelling*, vol. 18, no. 11, pp. 1–16, 1993.

- [80] S. Zhou, Z. Xu, and F. Liu, “Method for determining the optimal number of clusters based on agglomerative hierarchical clustering,” *IEEE transactions on neural networks and learning systems*, vol. 28, no. 12, pp. 3007–3017, 2016.
- [81] C. Zhong, D. Miao, and P. Fränti, “Minimum spanning tree based split-and-merge : A hierarchical clustering method,” *Information Sciences*, vol. 181, no. 16, pp. 3397–3410, 2011.
- [82] U. Maulik and S. Bandyopadhyay, “Performance evaluation of some clustering algorithms and validity indices,” *IEEE Transactions on pattern analysis and machine intelligence*, vol. 24, no. 12, pp. 1650–1654, 2002.
- [83] V. Kumar, J. K. Chhabra, and D. Kumar, “Performance evaluation of distance metrics in the clustering algorithms,” *INFOCOMP Journal of Computer Science*, vol. 13, no. 1, pp. 38–52, 2014.
- [84] A. K. Jain and R. C. Dubes, *Algorithms for clustering data*. Prentice-Hall, Inc., 1988.
- [85] T. Pang-Ning, M. Steinbach, and V. Kumar, “Introduction to data mining addison-wesley,” 2005.
- [86] F. Murtagh and P. Contreras, “Algorithms for hierarchical clustering : an overview,” *Wiley Interdisciplinary Reviews : Data Mining and Knowledge Discovery*, vol. 2, no. 1, pp. 86–97, 2012.
- [87] I. Gurrutxaga, I. Albisu, O. Arbelaitz, J. I. Martín, J. Muguerza, J. M. Pérez, and I. Perona, “Sep/cop : An efficient method to find the best partition in hierarchical clustering based on a new cluster validity index,” *Pattern Recognition*, vol. 43, no. 10, pp. 3364–3373, 2010.
- [88] S. Wu and T. W. Chow, “Clustering of the self-organizing map using a clustering validity index based on inter-cluster and intra-cluster density,” *Pattern Recognition*, vol. 37, no. 2, pp. 175–188, 2004.
- [89] L. Kaufman and P. Rousseeuw, *Finding Groups in Data : An Introduction To Cluster Analysis*. 01 1990.
- [90] A. W. Edwards and L. L. Cavalli-Sforza, “A method for cluster analysis,” *Biometrics*, pp. 362–375, 1965.
- [91] M. Roux, “A comparative study of divisive and agglomerative hierarchical clustering algorithms,” *Journal of Classification*, vol. 35, no. 2, pp. 345–366, 2018.
- [92] W. T. Williams and J. M. Lambert, “Multivariate methods in plant ecology : I. association-analysis in plant communities,” *The Journal of Ecology*, pp. 83–101, 1959.
- [93] P. Macnaughton-Smith, W. Williams, M. Dale, and L. Mockett, “Dissimilarity analysis : a new technique of hierarchical sub-division,” *Nature*, vol. 202, no. 4936, pp. 1034–1035, 1964.
- [94] L. Hubert, “Monotone invariant clustering procedures,” *Psychometrika*, vol. 38, no. 1, pp. 47–62, 1973.

- [95] M. Roux, “Basic procedures in hierarchical cluster analysis,” in *Applied multivariate analysis in SAR and environmental studies*, pp. 115–135, Springer, 1991.
- [96] M. Roux, “About divisive methods in hierarchical clustering,” *Data Science and Its Applications*, pp. 101–106, 1995.
- [97] J.-G. Sun, J. Liu, and L.-Y. Zhao, “Clustering algorithms research,” *Journal of software*, vol. 19, no. 1, pp. 48–61, 2008.
- [98] S. Na, L. Xumin, and G. Yong, “Research on k-means clustering algorithm : An improved k-means clustering algorithm,” in *2010 Third International Symposium on intelligent information technology and security informatics*, pp. 63–67, Ieee, 2010.
- [99] A. Fahim, A. Salem, F. A. Torkey, and M. Ramadan, “An efficient enhanced k-means clustering algorithm,” *Journal of Zhejiang University-Science A*, vol. 7, no. 10, pp. 1626–1633, 2006.
- [100] K. A. Nazeer and M. Sebastian, “Improving the accuracy and efficiency of the k-means clustering algorithm,” in *Proceedings of the world congress on engineering*, vol. 1, pp. 1–3, Citeseer, 2009.
- [101] L. A. Zadeh, “Information and control,” *Fuzzy sets*, vol. 8, no. 3, pp. 338–353, 1965.
- [102] E. H. Ruspini, “A new approach to clustering,” *Information and control*, vol. 15, no. 1, pp. 22–32, 1969.
- [103] D. J. Bora, D. Gupta, and A. Kumar, “A comparative study between fuzzy clustering algorithm and hard clustering algorithm,” *arXiv preprint arXiv :1404.6059*, 2014.
- [104] A. Taherpour, A. Cheshmeh Sefidi, A. Bemani, and T. Hamule, “Application of fuzzy c-means algorithm for the estimation of asphaltene precipitation,” *Petroleum Science and Technology*, vol. 36, no. 3, pp. 239–243, 2018.
- [105] R. J. Hathaway and J. C. Bezdek, “Fuzzy c-means clustering of incomplete data,” *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 31, no. 5, pp. 735–744, 2001.
- [106] R.A.Fisher, “iris,” 2018.
- [107] M. R. Rezaee, B. P. Lelieveldt, and J. H. Reiber, “A new cluster validity index for the fuzzy c-mean,” *Pattern recognition letters*, vol. 19, no. 3-4, pp. 237–246, 1998.
- [108] Y. Zhang, W. Wang, X. Zhang, and Y. Li, “A cluster validity index for fuzzy clustering,” *Information Sciences*, vol. 178, no. 4, pp. 1205–1218, 2008.
- [109] J. C. Dunn, “Well-separated clusters and optimal fuzzy partitions,” *Journal of cybernetics*, vol. 4, no. 1, pp. 95–104, 1974.
- [110] S. Saitta, B. Raphael, and I. F. C. Smith, “A comprehensive validity index for clustering,” *Intelligent Data Analysis*, vol. 12, pp. 529–548, 2008. 6.
- [111] T. Caliński and J. Harabasz, “A dendrite method for cluster analysis,” *Communications in Statistics-theory and Methods*, vol. 3, no. 1, pp. 1–27, 1974.

- [112] D. L. Davies and D. W. Bouldin, “A cluster separation measure,” *IEEE transactions on pattern analysis and machine intelligence*, no. 2, pp. 224–227, 1979.
- [113] L. Kaufman and P. J. Rousseeuw, *Finding groups in data : an introduction to cluster analysis*, vol. 344. John Wiley & Sons, 2009.
- [114] J. C. Bezdek, “Numerical taxonomy with fuzzy sets,” *Journal of mathematical biology*, vol. 1, no. 1, pp. 57–71, 1974.
- [115] J. C. Bezdek, “Cluster validity with fuzzy sets,” 1973.
- [116] K.-L. Wu and M.-S. Yang, “A cluster validity index for fuzzy clustering,” *pattern recognition letters*, vol. 26, no. 9, pp. 1275–1291, 2005.
- [117] Y. Fukuyama, “A new method of choosing the number of clusters for the fuzzy c-mean method,” in *Proc. 5th Fuzzy Syst. Symp.*, 1989, pp. 247–250, 1989.
- [118] X. L. Xie and G. Beni, “A validity measure for fuzzy clustering,” *IEEE Transactions on pattern analysis and machine intelligence*, vol. 13, no. 8, pp. 841–847, 1991.
- [119] Craigloewen-Msft, “Comparing wsl 1 and wsl 2.”
- [120] G. Van Rossum and F. L. Drake, *Python 3 Reference Manual*. Scotts Valley, CA : CreateSpace, 2009.
- [121] “Python.” <https://pythonprogramminglanguage.com>. accessed : 09-July-2021.
- [122] “Anaconda software distribution,” 2020.
- [123] “Anaconda.” <https://www.anaconda.com/>. accessed : 09-July-2021.
- [124] “Conda.” <https://conda.io/en/latest/>. accessed : 09-July-2021.
- [125] T. Kluyver, B. Ragan-Kelley, F. Pérez, B. Granger, M. Bussonnier, J. Frederic, K. Kelley, J. Hamrick, J. Grout, S. Corlay, P. Ivanov, D. Avila, S. Abdalla, and C. Willing, “Ju-
pyter notebooks – a publishing format for reproducible computational workflows,” in *Positioning and Power in Academic Publishing : Players, Agents and Agendas* (F. Loizides and B. Schmidt, eds.), pp. 87 – 90, IOS Press, 2016.
- [126] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blon-
del, P. Prettenhofer, R. Weiss, V. Dubourg, *et al.*, “Scikit-learn : Machine learning in
python,” *Journal of machine learning research*, vol. 12, no. Oct, pp. 2825–2830, 2011.
- [127] W. McKinney *et al.*, “Data structures for statistical computing in python,” in *Proce-
edings of the 9th Python in Science Conference*, vol. 445, pp. 51–56, Austin, TX, 2010.
- [128] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. Fernández del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, “Array programming with NumPy,” *Nature*, vol. 585, p. 357–362, 2020.
- [129] J. D. Hunter, “Matplotlib : A 2d graphics environment,” *Computing in science & en-
gineering*, vol. 9, no. 3, pp. 90–95, 2007.

- [130] S. Tosi, *Matplotlib for Python Developers : Build Remarkable Publication Quality Plots the Easy Way*. From technologies to solutions, Packt Publishing, 2009.
- [131] A. Riddell, A. Hartikainen, and M. Carter, “pystan (3.2.0).” PyPI, Mar. 2021.
- [132] R. Kumar, C. Carroll, A. Hartikainen, and O. Martin, “Arviz a unified library for exploratory analysis of bayesian models in python,” *Journal of Open Source Software*, vol. 4, no. 33, p. 1143, 2019.
- [133] “Exploratory analysis of bayesian models.” <https://arviz-devs.github.io/arviz/>. accessed : 09-July-2021.
- [134] S. Blog, “Kdd cup 2010 : Student performance evaluation.” <https://kdd.org/kdd-cup/view/kdd-cup-2010-student-performance-evaluation/Data>.
- [135] H. Abdi and L. J. Williams, “Principal component analysis,” *Wiley interdisciplinary reviews : computational statistics*, vol. 2, no. 4, pp. 433–459, 2010.
- [136] J. Warner, J. Sexauer, Scikit-Fuzzy, Twmeggs, Alexsavio, A. Unnikrishnan, G. Castelão, F. A. Pontes, T. Uelwer, pd2f, and et al., “Jdwarner/scikit-fuzzy : Scikit-fuzzy version 0.4.2.” <https://zenodo.org/record/3541386>, Nov 2019.