République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Abbes LAGHROUR KHENCHELA

Faculté des Sciences et de la Technologie

Département de Mathématiques et Informatique

Mémoire

Présenté en vue de l'obtention du diplôme de Master
Spécialité : Génie Logiciel et Systèmes Distribués
Filière : Informatique
Domaine : Mathématiques et Informatique

Thème:

La combinaison des algorithmes d'optimisation par essaims (swarm) et les méthodes d'apprentissage.

Présenté par :

01) MAHTALLAH Alim

02) SOUALHI Mourad

Dirigé par :

Mr. AZIZI Nabil

Devant le jury de soutenance composé de :

Dr. HOUASSI Hichem	Président
Dr. LEDMI Makhlouf	Examinateur
Mr. AZIZI Nabil	Rapporteur

Année universitaire: 2021/2022

Remerciements

En premier lieu nous tenons à remercier dieu le tout puissant qui nous donner la volonté et la force pour accomplir ce travail.

Nous tenons à remercier M.AZIZI Nabil qui a encadré ce mémoire. Nous le remercie d'abord pour son encadrement, ses conseils avisés, tant scientifiques que pédagogiques et pour la patience, la gentillesse avec lesquelles il les a prodigués, pour son soutien, sa disponibilité et pour la collaboration étroite dans laquelle nous avons travaillé et son aide qui nous a permis de mener à bien ce mémoire.

Nous remercier également tous les enseignants du département informatique qui ont contribué à notre formation.

En fin, nous remercions nos familles, nos amis et tous ceux qui, de près ou de loin, nous ont soutenu.

ملـــــــخص:

المصنف الثنائي هو مصنف يحتاج إلى مجموعتين أثناء طور التدريب، إحداهما تحتوي على بيانات معلمة كعناصر إيجابية والأخرى تحتوي على بيانات معلمة كعناصر سلبية. الغرض من ذلك هو استنتاج نموذج تصنيف من هاتين المجموعتين. غالبًا ما يحدث أن تحتوي البيانات على جزء صغير فقط من العناصر المعلمة كعناصر إيجابية، والباقي هي عناصر غير معروفة العلامة. يسمى هذا النوع من المشكلات التعلم الإيجابي وغير المصنف. في هذا السياق، اقترحنا طريقة لاختيار أفضل العناصر الموثوقة كعناصر سلبية التصنيف بناءً على تقنيات ذكاء الأسراب، حتى نتمكن بعد ذلك من تطبيق طريقة SVM ,من أجل ذلك اقترحنا طريقة سميناها BCSA-SVM وهي عبارة عن مزيج من الترميز الثنائي لخوارزمية Crow Search Algorithm وطريقة SVM .

Abstract:

A binary classifier is a classifier that needs two sets in its training phase, one of them contains labeled positive data and the other contains labeled negative data. The purpose is to infer a classification model from these two sets. It often happens that the data contains only a small part of labeled positive elements, the rest are unlabeled elements. This type of problem is called positive and unlabeled learning. In this context, we have proposed a method for selecting the best reliable negatives based on swarm intelligence techniques, so we can then apply the SVM method. For this, we have proposed the BCSA-SVM method which is a combination of binarization of the Crow Search Algorithm and the SVM method.

Résumé:

Un classifieur binaires est un classifieur qui a besoin dans sa phase d'apprentissage de deux ensembles l'un contient des données positivement étiquetées et l'autre des données négativement étiquetées. Le but étant d'inférer un modèle de classification à partir de ces deux ensembles. Il arrive souvent que les données contiennent uniquement une petite partie des éléments positivement étiquetées, le reste ce sont des éléments non étiquetés. Ce type de problématique est appelé l'apprentissage positif et non étiqueté. Dans ce contexte, nous avons proposé une méthode de sélection des meilleurs négatifs fiables basé sur les techniques d'intelligence en essaim pour qu'on puisse par la suite appliquer la méthode SVM. Pour cela nous avons proposé la méthode BCSA-SVM qui est une combinaison de binarisation de l'algorithme Crow Search Algorithm et la méthode SVM.

Table des matières

Introduction générale :

La croissance impressionnante que les données quantitatives (categorical data) et les données qualitatives connue ces dernières décennies à besoin de renforcement des techniques permettant la classification de ces données. La classification automatique est l'une des méthodes principales.

Les méthodes d'apprentissage automatique forment une classe de techniques attrayantes pour l'accomplissement des tâches de classification des données connues et non connues ce qu'on appelle données positifs et non étiquetées. Bien choisis, ces outils peuvent être amenés à accompagner, voire à remplacer l'opérateur humain.

Les algorithmes d'apprentissage automatique ont été appliqués à divers domaines, notamment le traitement du langage naturel et de la parole, la reconnaissance de l'écriture manuscrite, la vision robotisée, la fouille de données, les moteurs de recherche sur Internet, le diagnostic médical, la bio-informatique, etc. Les techniques d'apprentissage ont ainsi joué un rôle crucial dans des applications qui vont de la mise au point de médicaments à l'analyse de grands réseaux de télécommunication.

Le problème principal de la classification est de construire un classifieur à partir d'un ensemble de données. Un grand nombre de données est généralement nécessaire pour créer un système de classification avec un taux de reconnaissance assez élevé. Pratiquement, il est souvent facile d'obtenir cette quantité de données dans certains types d'application.

Les machines à vecteurs supports sont un ensemble de techniques d'apprentissage destinées à résoudre des problèmes de discrimination, c'est à dire décider à quelle classe appartient un échantillon, ou de régression, c'est-à-dire prédire la valeur numérique d'une variable. Le succès de cette méthode est justifié par les solides bases théoriques qui la soutiennent. Il existe en effet un lien direct entre la théorie d'apprentissage statistique et l'algorithme d'apprentissage du SVM.

La méthode cherche alors l'hyperplan qui sépare les exemples positifs des exemples négatifs, en garantissant que la marge entre le plus proche des positifs et des négatifs soit maximale. Intuitivement, cela garantit un bon niveau de généralisation car de nouveaux exemples pourront ne pas être trop similaires à ceux utilisés pour trouver l'hyperplan mais être tout de même situés franchement d'un côté ou l'autre de la frontière. Un autre intérêt est la sélection de Vecteurs Supports qui représentent les vecteurs discriminants grâce auxquels est déterminé l'hyperplan. Les exemples utilisés lors de la recherche de l'hyperplan ne sont alors plus utiles et seuls ces vecteurs supports sont utilisés pour classer un nouveau cas **[01]**.

L'objet de ce mémoire est **la combinaison des algorithmes d'optimisation par essaims (swarm) et les méthodes d'apprentissage** appliqués aux données positifs et non étiquetés.

Organisation du mémoire :

Ce mémoire est organisé en **quatre chapitres Le premier chapitre** traite les notions fondamentales de l'apprentissage automatique et passer en revue les différents types d'apprentissage**. Le deuxième chapitre** décrit l'apprentissage positif et non étiqueté (PU Learning). Les machines à vecteurs supports, les algorithmes d'intelligence en essaim (swarm) et la méthode méta-heuristique Crow Search Algorithm (CSA) sont présentés dans **le troisième chapitre**. Nous allons consacrer **le quatrième chapitre** pour la description de notre contribution à la problématique de l'apprentissage positif et non étiqueté. Notre contribution s'est située principalement à la proposition d'une nouvelle méthode de binarisation de l'algorithme CSA pour que celle-ci s'adapte à notre problématique, à savoir l'apprentissage positif et non étiqueté. Cette méthode est utilisée à la sélection des meilleurs éléments négatifs parmi l'ensemble non étiqueté. Par la suite nous avons utilisé la méthode d'apprentissage SVM pour créer le meilleur modèle de classification.

Enfin **une conclusion générale plus des perspectives** de travail viennent clôturer ce mémoire.

Chapitre 1 :

L'apprentissage automatique "Machine Learning"

1- Introduction :

Actuellement l'informatique est presque présente dans tous les domaines : la santé, l'éducation, l'économie, la cosmologie etc... Cette présence se reflète dans la vie quotidienne de l'individu et elle a permis des facilités d'utilisation et de compréhension de plusieurs complexes domaines. Le développement de l'informatique et de la technologie continue à prendre une place de plus en plus importante dans tous les domaines, ce qui a permis le développement du matériel, les logiciels de surveillance, et les logiciels d'analyse qui augmentent la précision des résultats, **[04]** et avec la croissance impressionnante que le réseau Internet a connu ces deux dernières décennies pour renforcé le besoin de techniques permettant l'extraction de connaissances à partir des données. La classification automatique est l'une des méthodes principales. **[01]**

La précision des résultats sa donne une grande attention par les scientifiques, en particulier les spécialistes dans les domaines d'intelligence artificielle, les systèmes experts, l'apprentissage automatique **« Machine Learning »**, et l'apprentissage en profondeur **« Deep Learning ».** Les méthodes de l'apprentissage automatique forment une classe de techniques attrayantes pour l'accomplissement des tâches d'extraction de connaissances évoquées. Bien choisis, ces outils peuvent être amenés à accompagner, voire à remplacer l'opérateur humain.

Par exemple, aux Etats-Unis, les services postaux (USPS) ont automatisé l'aiguillage du courrier en fonction du code postal qui se trouve sur les enveloppes.

Les algorithmes d'apprentissage automatique ont été appliqués dans divers domaines, notamment le traitement du langage naturel et de la parole, la reconnaissance de l'écriture manuscrite, la vision robotisée, la fouille de données, les moteurs de recherche sur Internet, le diagnostic médical, la bio-informatique, etc. Les techniques d'apprentissage ont ainsi joué un rôle crucial dans des applications qui vont de la mise au point de médicaments à l'analyse de grands réseaux de télécommunication. **[01]**

Qu'est-ce que l'intelligence artificielle ou IA ? [02]

L'intelligence artificielle (IA) est à la fois la théorie et le développement concret de machines, systèmes et logiciels qui imitent l'intelligence humaine pour accomplir des tâches très évoluées. L'IA est basée sur une démarche d'apprentissage afin de reproduire une partie de l'intelligence humaine à travers une application, un système ou un processus. La reconnaissance de la parole, la perception visuelle et la traduction linguistique sont des exemples de systèmes d'intelligence artificielle.

Le machine Learning et le Deep Learning sont des sous-ensembles de l'intelligence artificielle.

Qu'est-ce que le machine Learning ? [02]

Le machine Learning (ML) est un sous-domaine de l'IA qui permet aux ordinateurs de développer des modèles d'apprentissage par eux-mêmes, sans aucune programmation, à partir de gros ensembles de

données (ou datasets) pour imiter la façon dont les êtres humains prennent des décisions.

On fournit donc des quantités colossales de données thématiques à un programme afin qu'il développe son apprentissage sur son sujet et qu'il puisse ensuite les traiter de manière autonome.

Le machine learning est utilisée pour identifier les tendances enfouies dans les gros ensembles de données et pour la modélisation statistique.

Qu'est-ce que le deep learning ? [02]

La couche immédiatement inférieure est occupée par le deep learning (DL), l'une des nombreuses approches de le machine learning. Le deep learning utilise des réseaux de neurones profonds pour identifier des structures dans des volumes considérables de données. Dans le contexte informatique, les « réseaux neuronaux » sont des ensembles d'algorithmes modélisés sur la structure biologique du cerveau humain.

Chaque réseau neuronal se concentre sur une couche spécifique de la tâche à apprendre. C'est la superposition de ces couches de neurones qui permet au système d'effectuer une tâche spécifique.

Citons comme exemples d'application le système de recommandation de films de Netflix qui adapte les propositions en fonction des recherches passées des utilisateurs et l'algorithme du MIT qui analyse des vidéos et permet de prédire très rapidement les futurs comportements.

Quelles différences entre intelligence artificielle, machine learning et deep learning ?

Le machine learning et le deep learning sont apparentés. Ce sont tous deux des systèmes d'apprentissage basés sur la technologie de l'intelligence artificielle (IA) mais construits sur différentes couches d'abstractions.

Un des aspects qui est commun au machine learning, au deep learning et à l'IA, c'est que toutes les applications reposent sur ces trois technologies (par exemple voitures autonomes ou logiciels qui aident les médecins à déterminer les risques d'accident cardiaque chez leurs patients) ont besoin de toujours plus de big data et de puissance de traitement pour développer leur apprentissage et faire apparaître des résultats exploitables.

Ces trois technologies nécessitent un volume d'informations pour pouvoir développer un modèle d'apprentissage **(Voir figure 1.1). [02]**

Pour mieux comprendre la différence d'approche entre ces trois technologies, voyons comment elles doivent procéder pour apprendre à un ordinateur pour reconnaître la présence de chat sur des images :

- **L'IA** exige qu'un programmeur écrit tout le code nécessaire à l'ordinateur pour reconnaître un chat présent sur une image. Le programmateur crée le modèle d'apprentissage.
- **Le machine Learning** exige que des programmeurs apprennent au système de quoi ressemble un chat en lui montrant différentes images et en corrigeant son analyse jusqu'à ce que celle-ci soit correcte (ou plus précise). On parle d'apprentissage supervisé puisque l'intervention humaine est nécessaire.
- **Le Deep Learning** divise la tâche de reconnaissance des caractéristiques du chat en plusieurs couches

une couche de l'algorithme apprend à reconnaître les yeux, une autre les oreilles, une troisième la silhouette générale, etc. Une fois connectées, ces différentes couches possèdent une certaine capacité de reconnaissance des chats afin de reconnaître l'animal sur chaque nouvelle image soumise **[02].**

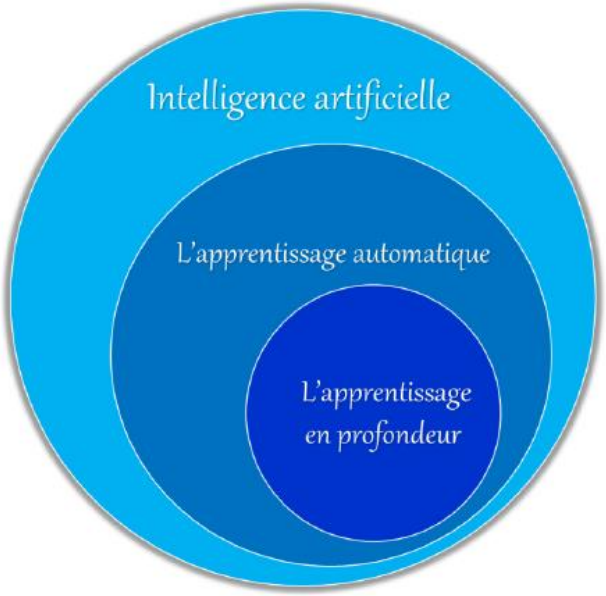


Figure 1.1 : Les trois technologies. **[04]**

2- L'apprentissage automatique « Machine Learning »

2-1- Définition : [03]

L'apprentissage automatique (en anglais : machine Learning, litt. « apprentissage machine »), apprentissage artificiel ou apprentissage statistique est un champ d'étude d'intelligence artificielle qui se fonde sur des approches mathématiques et statistiques pour donner aux ordinateurs la capacité d'« apprendre » à partir de données, c'est-à-dire d'améliorer leurs performances à résoudre des tâches sans être explicitement programmés pour chacune. Plus largement, il concerne la conception, l'analyse, l'optimisation, le développement et l'implémentation de telles méthodes.

L'apprentissage automatique comporte généralement deux phases. La première consiste à estimer un modèle à partir de données, appelées observations, qui sont disponibles et en nombre fini, lors de la phase de conception du système. L'estimation du modèle consiste à résoudre une tâche pratique, telle que traduire un discours, estimer une densité de probabilité, reconnaître la présence d'un chat dans une photographie ou participer à la conduite d'un véhicule autonome. Cette phase dite « d'apprentissage » ou « d'entraînement » est généralement réalisée préalablement à l'utilisation pratique du modèle.

La seconde phase correspond à la mise en production, le modèle étant déterminé, de nouvelles données peuvent alors être soumises afin d'obtenir le résultat correspondant à la tâche souhaitée. En pratique, certains systèmes peuvent poursuivre leur apprentissage une fois en production, pour peu qu'ils aient un moyen d'obtenir un retour sur la qualité des résultats produits.

Selon les informations disponibles durant la phase d'apprentissage, l'apprentissage est qualifié de

différentes manières. Si les données sont étiquetées (c'est-à-dire que la réponse à la tâche est connue pour ces données), il s'agit d'un apprentissage supervisé. On parle de classification ou de classement si les étiquettes sont discrètes, ou de régression si elles sont continués. Si le modèle est appris de manière incrémentale en fonction d'une récompense reçue par le programme pour chacune des actions entreprises, on parle d'apprentissage par renforcement. Dans le cas le plus général, sans étiquette, on cherche de déterminer la structure sous-jacente des données (qui peuvent être une densité de probabilité) et il s'agit alors d'apprentissage non supervisé. L'apprentissage automatique peut être appliqué à différents types de données, tels des graphes, des arbres, des courbes, ou plus simplement
des vecteurs de caractéristiques, qui peuvent être des variables qualitatives ou quantitatives continues ou discrètes**.**

2-2-Principes : [03]

L'apprentissage automatique (AA) permet à un système piloté ou assisté par ordinateur comme un programme, une IA ou un robot, d'adapter ses réponses ou comportements aux situations rencontrées, en se fondant sur l'analyse de données empiriques passées issues de bases de données, de capteurs, ou du web.

L'AA permet de surmonter la difficulté qui réside dans le fait que l'ensemble de tous les comportements possibles compte tenu de toutes les entrées possibles devient rapidement trop complexe à décrire et programmer de manière classique (on parle d'explosion combinatoire). On confie donc à des programmes d'AA le soin d'ajuster un modèle pour simplifier cette complexité et de l'utiliser de manière opérationnelle. Idéalement, l'apprentissage visera à être non supervisé, c'est-à-dire que les réponses aux données d'entraînement ne sont pas fournies au modèle.

Ces programmes, selon leur degré de perfectionnement, intègrent éventuellement des capacités de traitement probabiliste des données, d'analyse de données issues de capteurs, de reconnaissance (reconnaissance vocale, de forme, d'écriture…), de fouille de données, d'informatique théorique...**.**

2-3-Applications : [03]

L'apprentissage automatique est utilisé dans un large spectre d'applications pour doter des ordinateurs ou des machines de capacité d'analyser des données d'entrée comme : perception de leur environnement (vision, Reconnaissance de formes tels des visages, schémas, segmentation d'image, langages naturels, caractères dactylographiés ou manuscrits ; moteurs de recherche, analyse et indexation d'images et de vidéo, en particulier pour la recherche d'image par le contenu ;
aide aux diagnostics, médical notamment, bio-informatique, chémo-informatique ; interfaces cerveau-machine ; détection de fraudes à la carte de crédit, cyber-sécurité, analyse financière, dont analyse du marché boursier ; classification des séquences d'ADN ; jeu ; génie logiciel ; adaptation de sites Web ; robotique (locomotion de robots, etc.) ; analyse prédictive dans de nombreux domaines (financière, médicale, juridique, judiciaire)**.**

2-4-Exemple : [03]

- un système d'apprentissage automatique peut permettre à un robot ayant la capacité de bouger ses membres, mais sans savoir initialement rien de la coordination des mouvements qui permettent la marche, d'apprendre à marcher. Le robot commencera par effectuer des mouvements aléatoires, puis, en sélectionnant et en privilégiant les mouvements lui permettant d'avancer, mettra peu à peu en place une marche de plus en plus efficace.

Donc l'objectif visé est de rendre la machine ou l'ordinateur capable d'apporter des solutions à des problèmes compliqués, par le traitement d'une quantité astronomique d'informations. Cela offre ainsi une possibilité d'analyser et de mettre en évidence les corrélations qui existent entre deux ou plusieurs situations données, et de prédire leurs différentes implications**.**

2-5- En quoi consiste l'apprentissage automatique?

De manière générale, un programme informatique tente de résoudre un problème pour lequel nous avons la solution. Par exemple : calculer la moyenne générale des étudiants, classer les étudiants selon leur moyenne.

Pour certains problèmes, nous ne connaissons pas de solution exacte et donc nous ne pouvons pas écrire de programme informatique. Par exemple : reconnaitre automatiquement des chiffres écrits à la main à partir d'une image scannée, déterminer automatiquement une typologie des clients d'une banque, jouer automatiquement aux échecs contre un humain ou un autre programme.
En revanche, pour ces problèmes il est facile d'avoir une base de données en regroupant de nombreuses instances du problème considéré.

L'apprentissage automatique consiste alors de programmer des algorithmes qui permettent d'apprendre automatiquement de données et d'expériences passées, un algorithme cherche à résoudre au mieux un problème considéré.

2-6- Un domaine pluridisciplinaire :

L'apprentissage automatique (AA) ("Machine Learning") est la croisée de plusieurs disciplines :

1- Les statistiques : pour l'inférence de modèles à partir de données.

2- Les probabilités : pour modéliser l'aspect aléatoire inhérent aux données et au problème d'apprentissage.

3- L'intelligence artificielle : pour étudier les tâches simples de reconnaissance de formes que font les humains (comme la reconnaissance de chiffres par exemple), et parce qu'elle fonde une branche de l'AA dite symbolique qui repose sur la logique et la représentation des connaissances.

4- L'optimisation : pour optimiser un critère de performance ?, soit d'estimer des paramètres d'un modèle, soit de déterminer la meilleure décision à prendre étant donné une instance d'un problème.

5- L'informatique : puisqu'il s'agit de programmer des algorithmes et qu'en AA ceux-ci peuvent être de grande complexité et gourmands en termes de ressources de calcul et de mémoire.

2-7- Types d'apprentissage automatique : [04]

Il existe également de différents types d'apprentissage automatique. La définition du type d'apprentissage est basée sur la réponse pour deux questions suivantes :

- **Est-ce que** cet apprentissage compte sur la supervision humaine dans son entrainement et son apprentissage ?

- **Est-ce que** ce type d'apprentissage utilise une base de données fournie par l'être humain ?

1- Si la réponse est oui pour les deux questions, nous avons un apprentissage supervisé.

2- Si la réponse est non pour la première question, et oui pour la deuxième question, nous parlons d'un apprentissage non supervisé.

3- Si la réponse est non pour les deux questions, le type d'apprentissage est l'apprentissage par renforcement.

Dans ce qui suit, nous définissons chacun de ces types : apprentissage supervisé, apprentissage non supervisé, et apprentissage par renforcement..

2-7-1- Apprentissage supervisé : [04]

Dans l'apprentissage supervisé l'être humain aide l'algorithme pour apprendre, un data scientiste sert de guider et il apprend à l'algorithme les résultats qu'il doit trouver. Le même cas lorsqu'on apprend un enfant pour identifier les fruits, en les mémorisant dans sa mémoire. Dans l'apprentissage supervisé, l'algorithme apprend grâce à un jeu de données déjà étiqueté et dont le résultat est prédéfini **(Voir figure 1.2)**

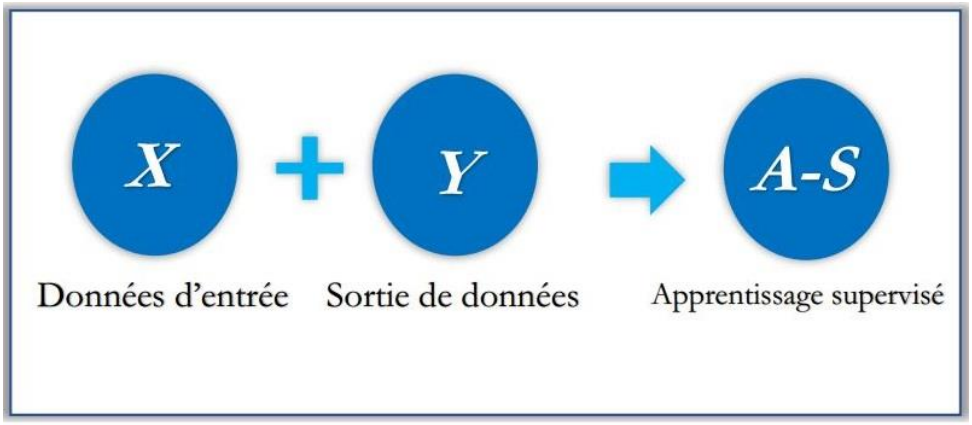


Figure 1.2 : Apprentissage supervisé **[04].**.

Les algorithmes de l'apprentissage automatique supervisé sont les plus couramment utilisés, il y a deux types d'apprentissage supervisé :

➢**La classification** : la classification consiste à trouver le lien entre une variable d'entrée(X) et une variable de sortie discrète (Y), en suivant une loi multinomiale.

➢**Régression** : la régression consiste à prédire une valeur continue pour la variable de sortie.

Les algorithmes les plus célèbres utilisés dans cette approche sont les suivants:

➢**SVM (Machines à vecteurs de support)**: **[04]** est un apprentissage automatique très puissant et polyvalent modèle, capable d'effectuer la classification linéaire ou non linéaire, la régression, et même détection des valeurs aberrantes. C'est l'un des modèles les plus populaires de l'apprentissage automatique « Machine Learning », et n'importe qui intéressés par cette approche devraient l'avoir dans leur boîte à outils. Les SVM sont particulièrement bien adapté à la classification d'ensembles de données complexes mais de petite ou moyenne taille. L'algorithme SVM consiste à chercher à la fois l'hyperplan optimal ainsi que de minimiser les erreurs de classification. (**Pour plus de détail sur les machines à vecteurs de support voir chapitre 3**).

➢**La méthode de Rocchio [05] :** La méthode de Rocchio est un classifieur linéaire proposé dans **(Racchio, 1971 [06])** pour améliorer les systèmes de recherche documentaires. Il représente les catégories par des profils «prototypiques». Un profil de la classe $\mathbf{c_i}$ est une liste de termes pondérés, dont la présence et l'absence discriminent au mieux cette classe $\mathbf{c_i}$. Il se caractérise par sa simplicité et interpretabilité car pour un expert ce profil prototype est plus compréhensible qu'un réseau de neurones par exemple. L'apprentissage de ce type de classifieur est souvent précédé par une sélection et une réduction de termes. L'adaptation en catégorisation de texte de la formule bien connue de Rocchio a été proposée par **[07].**

La méthode de Rocchio calcule un classifieur $\vec{c_i} = (w_{1i}, \cdots, w_{|T|i})$ pour la catégorie $\mathbf{c_i}$ via la formule :

$$w_{ki} = \beta \cdot \sum_{d_j \in POS_i} \frac{w_{kj}}{POS_i} - \gamma \cdot \sum_{d_j \in NEG_i} \frac{w_{kj}}{NEG_i}$$

Où w_{kj} est le poids de t_k dans le document d_j, $POS_i = \{d_j \in Tr | \Phi(d_j, c_i) = vrai\}$, $NEG_i = \{d_j \in Tr | \Phi(d_j, c_i) = faux\}$, Tr est le corpus d'apprentissage.

β et **γ** sont deux paramètres choisis selon l'importance accordée aux deux ensembles POS_i et NEG_i.

Par exemple si **β** prend la valeur 1 et **γ** la valeur 0, le profil prototype $\mathbf{c_i}$ est le barycentre des exemples positifs. Le rôle des exemples négatifs est habituellement réduit par l'affectation des valeurs élevées pour **β** et des valeurs faibles pour **γ, [08]** utilisent **β** = 16 et **γ** = 4.

[09] propose une amélioration par la prise en compte seulement d'une partie ($NPOS_i$, near-positives) des exemples de l'ensemble NEG_i qui sont les plus proches aux exemples positifs, soit :

$$w_{ki} = \beta \cdot \sum_{d_j \in POS_i} \frac{w_{kj}}{POS_i} - \gamma \cdot \sum_{d_j \in NPOS_i} \frac{w_{kj}}{NPOS_i}$$

Le classement de nouveaux documents s'opère en calculant la distance euclidienne entre la représentation vectorielle de document et celle de chacune des classes ; le document est assigné à la classe la plus proche.

Cette méthode est tout à fait facile à implémenter, et est également assez efficace quand la séparation des classes soit linéaire, mais elle est peu adaptée dans les problème non linéairement séparable **(Lewis et al., 1996 [10])**.

➢ **Naïve Bayes: [04]**est un classifieur assez intuitif à comprendre. Il se base sur le théorème de Bayes des probabilités conditionnelles, et il suppose que les variables sont indépendantes entre elles. Cela permet de simplifier le calcul des probabilités.

[05] Dans un classifieur probabilistes, la CSV ($\mathbf{d_j}$,$\mathbf{c_i}$) (categorization status value) est vue comme une probabilité conditionnelle P($\mathbf{c_i} \mid \mathbf{d_j}$) d'appartenance du document $\mathbf{d_j}$ représenté par un vecteur $\vec{d_j} = (w_{1j}, \cdots, w_{|T|j})$ à la classe $\mathbf{c_i}$, et pour calculer cette probabilité le théorème de Bayes est appliqué:

$$P(c_i|\vec{d_j}) = \frac{P(c_i)P(\vec{d_j}|c_i)}{P(\vec{d_j})} \quad \text{……..(01)}$$

Dans **(01)** P($\mathbf{c_i}$) représente la probabilité qu'un document choisi aléatoirement soit appartient à la classe $\mathbf{c_i}$, et P($\vec{d}$j) représente la probabilité de choisir un document $\vec{d}$j qui est constante pour toute les classe.

Pour estimer P($\vec{d}$j $\mid$ c_i) en supposant l'indépendance des termes du document, on obtient :

$$P(\vec{d_j}|c_i) = \prod_{k=1}^{|T|} P(w_{kj}|c_i) \quad \text{…….(02)}$$

Les classifieurs résultant de cette hypothèse s'appellent les classifieurs « Bayésien Naïf », parce que l'hypothèse n'est jamais vérifiée et évidemment souvent fausse.

Lors de la phase d'apprentissage les probabilités P($\mathbf{w_{kj}} \mid \mathbf{c_i}$) sont estimées afin de les employer dans l'étape de classement d'un nouvel exemple selon la règle de bayes.

➢**Les arbres de décision [05]** : un arbre de décision sert à classer les futures observations, sachant qu'un corpus d'observations est déjà étiqueté.

Beaucoup de méthodes de catégorisation partagent quelques inconvénients, par exemple certains classifieurs ne peuvent pas être facilement compréhensibles par des humains (les classifieurs statistiques). En revanche les classifieurs symboliques, dont le classifieur d'arbre de décision est l'exemple le plus fameux, ne souffre pas de ce problème.

Un classifieur d'arbre de décision est une structure arborescente dont les nœuds internes représentent les termes, les deux branches partant de chaque nœud représentent l'existence ou l'absence du terme de ce nœud dans le document, et les feuilles représentent les catégories (voir **la figure 1.3**).

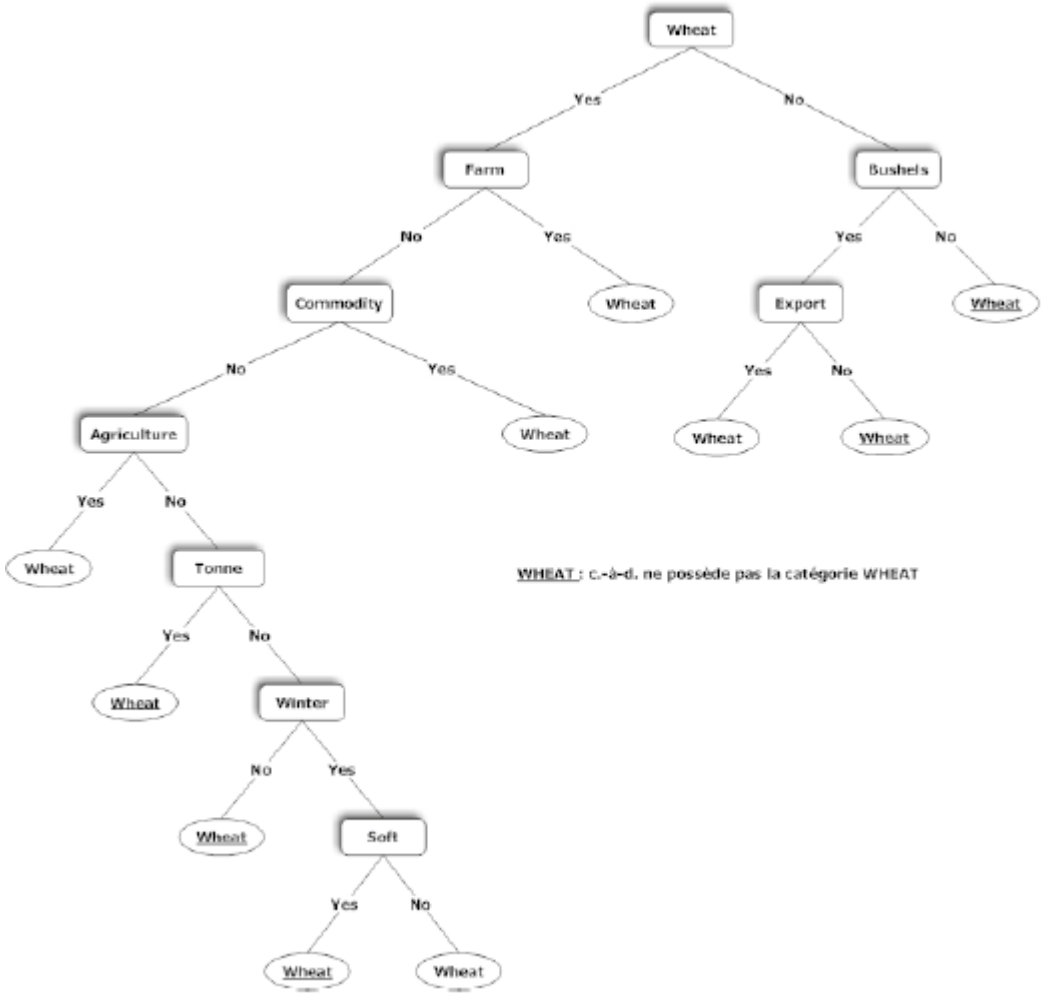


Figure 1.3 : Exemple d'arbre de décision appliquée sur le corpus Reuters **[05]** .

Pour construire un arbre de décision l'algorithme d'apprentissage [**11**] prend en entrée un échantillon Ω, comprenant N textes classés ($\mathbf{d_j}$,$\mathbf{c_i}$), et fournit en sortie un arbre de décision. L'algorithme procède de façon descendante : il part de la racine puis, récursivement, choisit l'étiquette des fils.

Il se résume dans les étapes suivantes :

– L'algorithme d'arbres de décision construit une succession de partitions sur l'échantillon de données d'apprentissage. Les partitions sont de plus en plus fines. Le premier nœud contient toutes les données de l'échantillon avec leurs classes.

– On cherche, parmi les variables prédictives, celle qui donne la meilleure partition selon un critère de sélection des variables et on répète le processus de segmentation pour chaque nœud obtenu sans se préoccuper des autres nœuds. Si les variables prédictives sont discrètes, chaque variable peut engendrer une partition dont le nombre d'éléments dépend du nombre de valeurs que la variable peut prendre.

Si les variables sont continués, elles doivent être discrétisées, soit a priori, soit sur chaque nœud.

– Un nœud est saturé s'il n'existe aucune variable (prédictive) qui permet de créer localement une partition qui améliore le critère utilisé.

– Le processus s'arrête quand tous les nœuds sont saturés.

– L'élagage qui consiste à simplifier un arbre de décision en coupant des branches. Il possède deux objectifs, l'un est de simplifier l'arbre de décision, l'autre est de diminuer le sur-apprentissage (augmenter la capacité de généralisation) et, par la même, diminuer le taux d'erreur. Il y'a deux possibilités soit élagage lors de la construction ou élagage après la construction.

Algorithme 1: *Algorithme général d'apprentissage par arbres de décision*

Données : un échantillon Ω de S textes classés (d_j, c_i)

Entrées : arbre vide ; nœud courant : racine ; échantillon courant : Ω

```
répéter
    si le nœud courant est terminal alors
        étiqueter le nœud courant par une feuille portant le nom de cette classe;
    sinon
        choisir le meilleur attribut (terme) pour créer le sous-arbre;
    fin
    // nœud courant : un nœud non encore étudié et l'échantillon courant :
       échantillon atteignant le nœud courant
jusqu'à production d'un arbre de décision;
élaguer l'arbre de décision obtenu;
```

Résultat : arbre de décision élagué

Un arbre de décision classe un document en commençant de la racine de l'arbre et en se déplaçant successivement en bas par l'intermédiaire des branches dont les conditions sont satisfaites par le document jusqu'à ce qu'un nœud de feuille soit atteinte. Le document est alors assigné à la catégorie de la feuille. Chaque branche de l'arbre constitue une règle de décision de la forme **si** condition **alors** conclusion. L'ensemble de ces règles constitue le modèle de prédiction (voir **la figure 1.4**).

si	((*wheat* et *farm*)	ou
	(*wheat* et *commodity*)	ou
	(*bushels* et *export*)	ou
	(*wheat* et *tonnes*)	ou
	(*wheat* et *winter* et $\neg soft$))	Alors *WHEAT* sinon $\neg WHEAT$

Figure 1.4 : Modèle de prédiction produit par un arbre de décision sous la forme de règles **[05]** .

Il y a un certain nombre d'algorithme d'apprentissage standard pour les arbres de décision, Parmi les plus populaires on peut citer ID3 (employés par **[12]**, C4.5 (employé par **[13]**, et C5 (employé par **[14]**).

➢**Les Forêts Aléatoires** [**04**] : cet algorithme fonde sur les arbres de décision, un modèle construit par de multiples arbres de décisions.

➢**La méthode des k plus proches voisins** [**04**]: cet algorithme consiste à essayer différentes valeurs de K pour obtenir la séparation la plus satisfaisante.

[05] Les classifieurs à base d'exemples n'établissent pas des représentations déclaratives explicites des catégories mais se fondent sur le calcul direct de la similarité entre le document à classifier et les documents d'apprentissage. Ces méthodes se sont aussi appelées lazy learners, puisque la phase d'apprentissage est réduite à un stockage des exemples d'apprentissage.

(Creecy et al.,1992 [15]) est le premier qui a appliqué les méthodes à base d'exemples à la catégorisation de textes.

(Aha et al., 1991[16]) montre qu'un algorithme d'apprentissage à base d'exemples est caractérisé par

1. une fonction de similarité,

2. une fonction de sélection des exemples typiques,

3. une fonction de classement qui détermine de quelle manière un nouvel exemple est lié aux exemples appris.

Le classifieur à base d'exemples le plus connu est les k-plus proches voisins (KNN, k-nearest neighbor). Il procède par la prédiction de classe d'un texte t en fonction des k textes les plus proches voisins déjà étiquetés en mémoire.

La méthode ne nécessite pas de phase d'apprentissage ; c'est l'échantillon d'apprentissage, associé à une fonction de distance et à une fonction de choix de la classe en fonction des classes des voisins les plus proches, qui constitue le modèle. Le modèle est constitué des trois éléments :

1) l'échantillon d'apprentissage**, 2)** la distance et **3)** la méthode de combinaison des voisins.

L'algorithme **2** montre comment classer un nouvel exemple par la méthode k-PPV

Algorithme 2: *Algorithme de classification par k-PPV*

Données : un échantillon de l textes classés en $C = c_1, c_2, \cdots, c_n$ classes

Entrées : le nombre k de voisins

1 **pour** *chaque texte* t **faire**

2 transformer le texte t en vecteur $t = (x_1, x_2, \cdots, x_m)$;

3 déterminer les k plus proches textes du texte t selon une métrique de distance;

4 combiner les classes de ces k exemples en une classe c;

5 **fin**

Résultat : le texte t associé à la classe c

La distance entre un texte et ses voisins se fait via une métrique de distance. Cette métrique peut être la métrique de Minkowski :

$$d_p(a,b) = \sqrt{\left(\sum_i |a_i - b_i|^p\right)^{\frac{1}{p}}}$$

Selon la valeur de p, on retrouve plusieurs distances connues :

– Si p = 1 cette distance est la distance de Manhattan définie par

$$d_m(a,b) = |a_1 - b_1| + |a_2 - b_2| + \cdots + |a_n - b_n|$$

– Si p = 2 c'est la distance euclidienne définie par

$$d_e(a,b) = \sqrt{\sum_i (a_i - b_i)^2}$$

– Si p = ∞ c'est la distance de Chebyshev définie par

$$d_c(a,b) = \max\{|a_1 - b_1|, |a_2 - b_2|, \cdots, |a_n - b_n|\}$$

L'emploi de k voisins, au lieu d'un seul, assure une plus grande robustesse à la prédiction. Toutefois, la valeur de k peut changer les performances du modèle, comme cela est présenté en **figure 1.5 :**

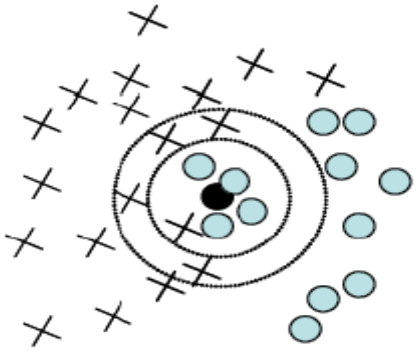


Figure 1.5 : Choix de k influences la décision : pour k = 5, la décision est de classer l'exemple « noir » dans la classe « ronds ». Pour k = 9, la décision est de le classer en tant que « croix » [**05**]

Une première façon de combiner les k classes des k plus proches voisins est le vote majoritaire. Elle consiste simplement à prendre la classe majoritaire. Une seconde façon est le vote majoritaire pondéré. Chaque vote, c'est-à-dire la classe d'un des k plus proches voisins, est pondéré: soit xi le voisin considéré, le poids de $c(x_i)$ est inversement proportionnel à la distance entre l'enregistrement à classer et xi.

➢ **Régression Logistique** [**04**]**:** l'algorithme de régression logistique consiste à trouver les meilleurs coefficients pour minimiser l'erreur entre la prédiction faite pour des destinations visitées et la vraie étiquette donnée (e.g. bon, mauvais etc.).

➢**Les réseaux de neurones** [**04**] : Les réseaux neuronaux sont un modèle informatique qui partage certaines propriétés avec le cerveau humain, dans lequel de nombreuses unités simples travaillent en parallèle sans centralisation, ils permettent de trouver des patterns complexes dans les données, il se compose de valeurs d'entrées, poids, fonction de transfert et une valeur de sortie (**Voir figure 1.6**).

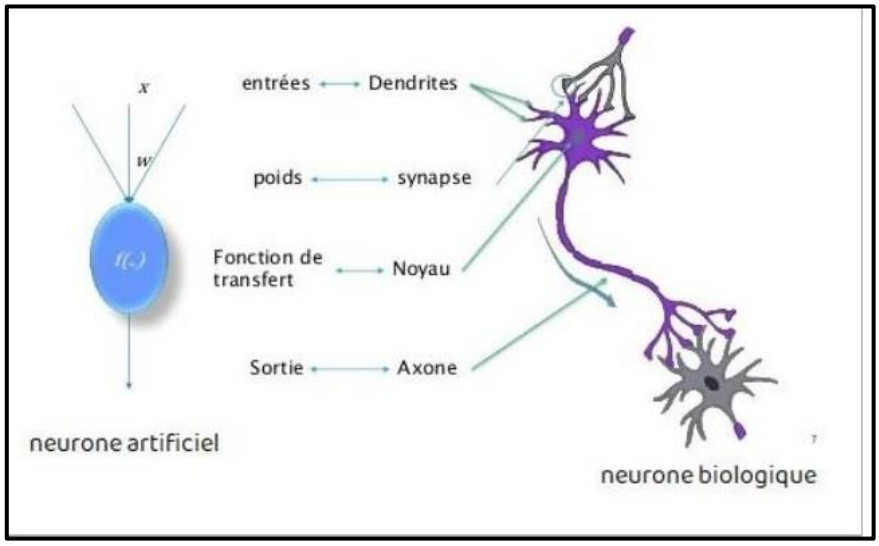


Figure 1.6 : Neurone biologique et neurone artificiel [**04**].

Il existe aussi d'autres algorithmes, tels que l'algorithme de régression linéaire, les Algorithmes génétiques. Certains algorithmes de régression peuvent également être utilisés pour la classification, et la régression, à titre d'exemple l'algorithme de la régression logistique.

2-7-1- 2- Comparaison entre classifieurs [05] :

[18] propose deux méthodes pour comparer les classifieurs et les évaluer. Cette comparaison peut être directe, ou indirecte :

1. une comparaison directe : il s'agit d'utilisation de plusieurs méthodes par le même auteur ; de cette manière, le découpage et les mesures sont identiques pour toutes les méthodes. **[18]** comparent les machines à vecteurs supports, les plus proches voisins, les réseaux de neurones, une combinaison linéaire, et des réseaux bayésiens. **[19]** proposent une série de comparaisons en mettant en compétition une variante de l'algorithme de Rocchio (appelée find similar), des arbres de décision, des réseaux bayésiens et des machines à vecteurs supports. Cette méthode de comparaison est la plus crédible de point de vue scientifique.

2. une comparaison indirecte : soient φ' et φ'' deux classifieurs. Ces deux classifieurs peuvent être comparés si deux conditions sont réunies :

(a) les deux classifieurs sont testés par différents groupes de chercheurs (même avec de conditions d'expérimentation différentes) sur deux collections Ω' et Ω'' respectivement;

(b) un ou plusieurs classifieurs de « références », $\bar{\Phi}_1, \bar{\Phi}_1, \cdots, \bar{\Phi}_m$ sont testés sur les deux collections Ω' et Ω'' par des comparaisons directes ; ceci donne une idée du niveau de difficulté d'apprentissage sur chaque collection.

Les comparaisons précédentes sont aboutées aux résultats suivants :

1. Les classifieurs par combinaison de décisions, les SVM, les méthodes à base d'exemples donnent les meilleurs résultats.

2. Les réseaux de neurones, les classifieurs « en ligne » donnent des résultats inférieurs à ceux des précédents.

3. Les classifieurs linéaires tels que la méthode de Rocchio donnent souvent de mauvais résultats.

2-7-1- 3- Évaluation des performances d'un classifieur: [20]

L'existence réelle de grands biais, dans la distribution et la répartition des classes a été observée dans différents domaines par **[21], [22], [24] et [28]**. Par exemple, dans la prise de décision médicale, les épidémies peuvent faire augmenter l'incidence d'une maladie avec le temps. Dans la détection de fraudes, la proportion des fraudes varie de manière significative de mois en mois et de zone en zone, **[23].** Dans chacun des exemples à deux classes, la prédominance d'une classe peut changer radicalement sans pour autant altérer fondamentalement les caractéristiques de la classe. Si les proportions d'éléments positifs ou/et négatifs changent dans une base de test, nous voulons que le système d'évaluation des performances ne soit pas perturbé. Pour cela, nous allons comparer deux méthodes applicables aux classifeurs à deux classes, la courbe Précision-Rappel et la courbe ROC. Les mesures que nous allons évoquer utilisent la matrice de confusion, **tableau 1.1,** qui permet la différenciation des erreurs selon chaque classe en vue d'évaluer un classifeur.

Définissons maintenant plusieurs mesures de manière formelle :

Catégorie Ci		Jugement Expert	
		Oui	Non
Jugement classifieur	Oui	TP_i	FP_i
	Non	FN_i	TN_i

Tableau 1.1 : Matrice de contingence de la classe C_i. **[20]**

Le taux de vrais positifs ("True positive rate") :

$$TPr = \frac{TP}{Pos} = \frac{TP}{TP + FN}$$

Le taux de vrais négatifs ("True négative rate") :

$$TNr = \frac{TN}{Neg} = \frac{TN}{TN + FN}$$

Le taux de faux positifs ("False positive rate") :

$$FPr = \frac{FP}{Neg} = \frac{FP}{FP + TN}$$

Le taux de faux négatifs ("False négative rate") :

$$FNr = \frac{FN}{Pos} = \frac{FN}{FN + TN}$$

Le taux de bonne classification ou l'exactitude (accuracy) :

$$acc = tbc = Pos * TPr + Neg * (1 - FPr)$$

La précision :

$$prec = \frac{TP}{PPos} = \frac{TP}{TP + FP}$$

Le rappel (recall) :

$$rec = TPr = \frac{TP}{Pos} = \frac{TP}{TP + FN}$$

Maintenant que nous avons caractérisé notre problème (estimation des taux de bonne et mauvaise classification, évaluation du type d'erreur,…..) via la matrice de confusion, nous allons représenter les performances des systèmes de classification à l'aide de **la courbe Précision Rappel** puis **de la courbe ROC.**

2-7-1- 3-1- Courbe Précision Rappel:

Nous allons maintenant traiter de la mesure "précision" et de la mesure "rappel ". Ces mesures très utilisées en recherche documentaire (information retrie val) permettent d'évaluer les performances (la pertinence) du retour d'information vis-à-vis d'une requête**, [25], [26].** La Précision et le Rappel sont généralement utilisés pour traiter des bases de documents statiques ; cependant, ils peuvent être utilisés dans des environnements dynamiques tels que la fouille de pages web, lorsque le nombre de documents non pertinents correspondant à une requête augmente à la vitesse de création de pages web sur Internet. Considérons **la figure 1.7** qui présente deux classifieurs évalués par la courbe Précision Rappel. Dans la **figure 1.7-a**, la base de test est équilibrée dans la distribution des classes (1 :1). Pour la figure **1.7-b**, les mêmes classifieurs sont utilisés, mais cette fois, le nombre d'éléments négatifs est dix fois plus important (1 :10). Nous constatons que les courbes Précision Rappel, **figures 1.7-a et 1.7-b**, diffèrent sensiblement, ce qui indique une sensibilité de la représentation par rapport à la distribution. Il s'agit donc d'un problème important mettant en évidence la faiblesse de la courbe Précision Rappel. De ce fait, nous devons faire appel à une autre représentation plus robuste vis-à-vis de la structure des bases de test. **La courbe ROC**, que nous allons maintenant présenter intervient dans ce sens.

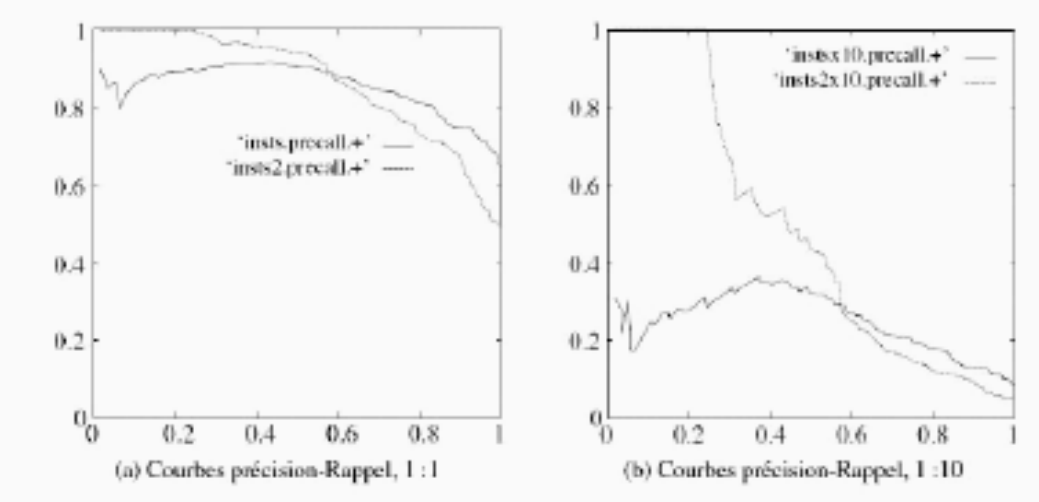
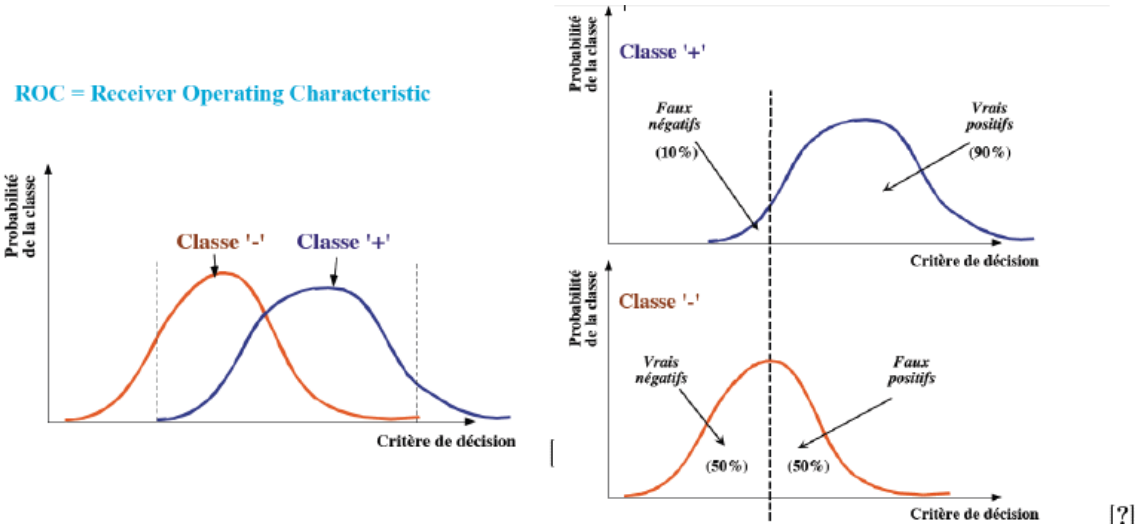


Figure 1.7 : Courbes Précision-Rappel vis-à-vis de la distribution des classes **[22].**

2-7-1- 3-2- Courbe ROC:

La courbe ROC (**Receiver Operating Characteristics**) offre à la fois une vision graphique et une mesure pertinentes de la performance d'un classifieur. Elle possède de nombreux avantages par rapport aux mesures de rappel et précision par classe : la performance est synthétisée par une unique mesure qui ne dépend pas des proportions de classe. Cet avantage se transforme néanmoins en inconvénient lorsqu'il s'agit de revenir rapidement aux mesures de rappel et de précision par classe ou d'estimer le comportement du classifieur selon les classes. Les mesures de rappel et précision sont en effet utiles car elles caractérisent précisément le comportement du classifieur sur chacune des classes. En particulier, lorsque les classes sont déséquilibrées, ces mesures fournissent des indications de performance plus représentatives et concrètes qu'un unique score pour le classifieur**. [?]** En général, les courbes ROC sont basées sur le taux de vrai positifs (tpr) et le taux de faux positifs (fpr). Il s'agit là de rapports qui ne dépendent donc pas de la distribution des classes. Cette méthode robuste permet de s'affranchir de la connaissance des coûts de classification et de la distribution des classes.



[?]

Figure 1.8 : Courbe ROC **[20].**

2-7-1- 3-3- F_mesure : [20]

La F-mesure est un indicateur de synthèse communément utilisé pour évaluer les algorithmes de classification de données textuelles, à partir de la précision et du rappel. Elle est utilisée indifféremment pour les classifications et les catégorisations.

$$\text{F-mesure} = (1+\beta^2)\cdot\frac{Precision\cdot Rappel}{(\beta^2\cdot Precision)+Rappel}$$

La F-mesure correspond à une moyenne harmonique de la précision et du rappel. Le paramètre permet de pondérer la précision ou le rappel et vaut généralement 1, **[27].** La mesure devient :

$$\text{F-mesure} = 2\cdot\frac{Precision\cdot Rappel}{Precision+Rappel}$$

L'avantage de ce choix est que lorsque la précision est égale au rappel, on obtient :

Précision = Rappel = F-mesure. Ceci facilite la lecture et on recherche à maximiser la F-mesure en maximisant simultanément la précision et le rappel. Le problème que pose cette méthode est qu'elle ne permet pas la différenciation des erreurs et reste sensible à la distribution des classes car basée sur la précision et le rappel. Nous allons maintenant nous intéresser à l'aire sous la courbe ROC afin de comparer les deux mesures. Il est donc préférable d'utiliser la seconde mesure qui est à notre disposition à savoir : l'aire sous la courbe ROC.

2-7-2- Apprentissage non supervisé [04] :

Avec l'apprentissage non supervisé la machine n'a pas besoin de l'aide pour apprendre **(Voir figure 1.9).** L'apprentissage non supervisé est une approche plus indépendante, dans laquelle un ordinateur apprend à identifier des processus et des schémas complexes sans aucun guide, il implique une formation basée sur des données sans étiquette, qui ne contiennent aucun résultat spécifique (Goodfellow, Bengio, & Courville, 2016).

Figure 1.9 : Apprentissage Non-supervisé **[04]** .

Il y a deux types d'apprentissage non supervisé :

a-Regroupement (Clustering) [04] : c'est une méthode d'analyse statistique utilisée pour organiser des données brutes en silos homogènes, à l'intérieur de chaque grappe, les données sont regroupées selon une caractéristique commune.

b- Réduction de la dimension [04] : l'objectif est de simplifier les données sans perdre trop d'informations, à titre d'exemple, fusionner plusieurs caractéristiques en un seul caractère.

Les algorithmes les plus célèbres utilisés dans cette approche sont [04] :

➢ **K-Moyenne** : est un algorithme de Regroupement (Clustering) il regroupe dans les même Cluster (Groupes) les données similaires (qui se ressemblent). Il utilise un raffinement itératif pour produire un résultat final.

➢**Analyse de classification hiérarchique (HCA)** : la mise dans un cluster hiérarchique est similaire à la mise dans un cluster normal, sauf que dans ce cas nous souhaitons mettre en place une hiérarchie des clusters. Cela peut s'avérer très important surtout quand nous désirons une flexibilité par

rapport au nombre de clusters voulu.

➢**PCA (Analyse des composants principaux)** : l'algorithme PCA consiste à transformer des variables liées entre elles, vers de nouvelles variables séparées les uns des autres. Ces nouvelles variables sont nommées les composantes principales, elles permettent au praticien de réduire le nombre de variables et de rendre l'information moins redondante.

➢**Apriori** : l'algorithme Apriori s'utilise dans une base de données transactionnelle pour extraire des ensembles d'éléments fréquents, puis générer des règles d'association.

2-7-3- L'apprentissage Semi-supervisé :

L'approche d'apprentissage semi-supervisé prend à la fois des entrées de données de formation étiquetées et non étiquetées. Ce type d'apprentissage est utile lorsqu'il est difficile d'extraire des caractéristiques utiles de données non étiquetées (approche supervisée) et que les experts en données ont du mal à étiqueter les données d'entrée (approche non supervisée).

Seule une petite quantité de données étiquetées dans ces algorithmes peut conduire à la précision du modèle.

Exemples des apprentissages semi-supervisés comprennent les tomo-densitogrammes et les IRM où un expert médical peut identifier quelques points dans les scans pour n'importe quelle maladie alors qu'il est difficile d'étiqueter tous les scans.

2-7-4- L'apprentissage par transfert : [03]

L'apprentissage par transfert peut être vu comme la capacité d'un système à reconnaître et appliquer des connaissances et des compétences, apprises à partir de tâches antérieures, sur de nouvelles tâches ou domaines partageant des similitudes. La question qui se pose est : comment identifier les similitudes entre la ou les tâche(s) cible(s) et la ou les tâche(s) source(s), puis comment transférer la connaissance de la ou des tâche(s) source(s) vers la ou les tâche(s) cible(s) ?

2-8- Le choix d'un type d'apprentissage automatique [04] :

Avec la présence de différents types de classifieurs pour l'apprentissage automatique, l'opération de choix d'un type est une question typique « Quel algorithme dois-je utiliser ? ». La réponse à cette question varie les facteurs suivants :

- ✓ **La taille, la qualité et la nature des données.**
- ✓ **Le temps de calcul disponible.**
- ✓ **L'urgence de la tâche.**
- ✓ **Le but d'utilisation de ces données.**

La figure suivante **(Voir figure 1.10),** fournir des indications sur les algorithmes à essayer en premier selon les facteurs mentionnés ci-dessus.

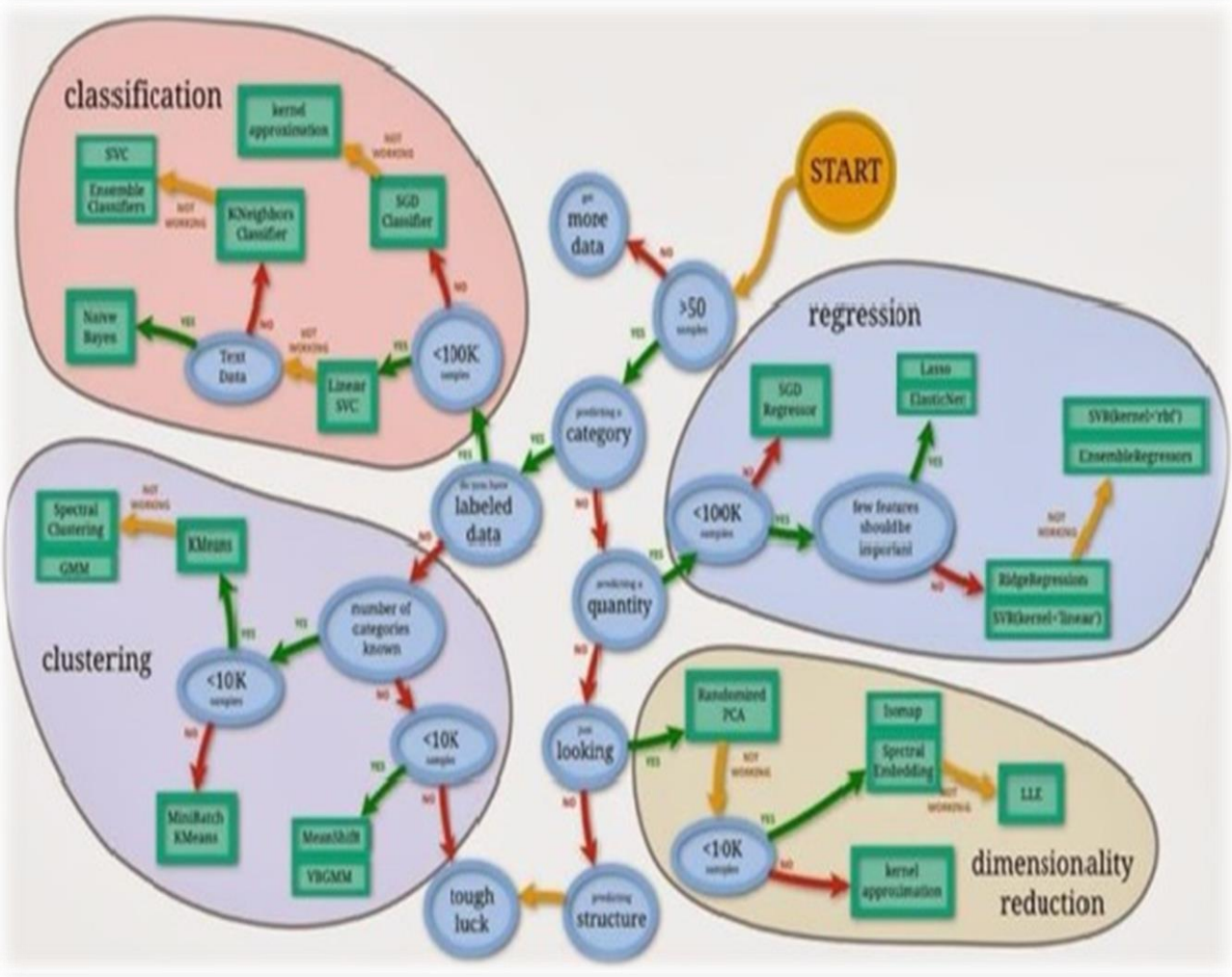


Figure 1.10 : Le choix de l'algorithme d'apprentissage selon certains facteurs (Li, 2017) **[04]** .

2-9- Étapes d'un projet d'apprentissage automatique [03] :

L'apprentissage automatique ne se résume pas à un ensemble d'algorithmes, mais suit une succession d'étapes.

1. **Définir le problème à résoudre.**
2. **Acquérir des données :** l'algorithme se nourrissant des données en entrée, c'est une étape importante. Il en va de la réussite du projet, de récolter des données pertinentes et en quantité et qualité suffisantes, et en évitant tout biais dans leur représentativité.
3. **Analyser et explorer les données :** L'exploration des données peut révéler des données d'entrée ou de sortie déséquilibrée pouvant nécessiter un rééquilibrage, le machine learning non supervisé peut révéler des clusters qu'il pourrait être utile de traiter séparément ou encore détecter des anomalies qu'il pourrait être utile de supprimer.
4. **Préparer et nettoyer les données** : les données recueillies doivent être retouchées avant utilisation. En effet, certains attributs sont inutiles, d'autre doivent être modifiés afin d'être compris par l'algorithme, et certains éléments sont inutilisables car leurs données sont

incomplètes. Plusieurs techniques telles que la visualisation de données, la transformation de données (en) ou encore la normalisation sont alors employées.

5. **Ingénierie ou extraction de caractéristiques:** les attributs peuvent être combinés entre eux pour en créer de nouveaux plus pertinents et efficaces pour l'entrainement du modèle.
6. **Choisir ou construire un modèle d'apprentissage:** un large choix d'algorithmes existe, et il faut en choisir un adapté au problème et aux données.
7. **Entrainer, évaluer et optimiser :** l'algorithme d'apprentissage automatique est entraîné et validé sur un premier jeu de données pour optimiser ses hyper paramètres.
8. **Test:** puis il est évalué sur un deuxième ensemble de données de test afin de vérifier qu'il est efficace avec un jeu de donnée indépendant des données d'entrainement, et pour vérifier qu'il ne fasse pas de sur apprentissage.
9. **Déployer :** le modèle est alors déployé en production pour faire des prédictions, et potentiellement utiliser les nouvelles données en entrée pour se ré-entraîner et être amélioré.
10. **Expliquer** : déterminer quelles sont les variables importantes et comment elles impactent les prédictions du modèle en général et au cas par cas.

3- Conclusion : La classification est une étape importante dans l'analyse de données qui consiste à regrouper les données en classes similaires. Il existe donc une quantité importante de méthodes pour la classification de données. Toutes sont issues des recherches sur l'apprentissage (" Machine Learning"). Dans ce chapitre, nous avons défini la notion de Machine Learning, en donnant les différentes notions et leurs différentes méthodes de classification. Dans le chapitre suivant, nous présentons un état de l'art des travaux qui concerne l'a**pprentissage positif et non étiqueté "le PU Learning"**.

Chapitre 2 :

Apprentissage positif et non étiqueté "PU Learning".

I-1- Introduction : [29]

A-

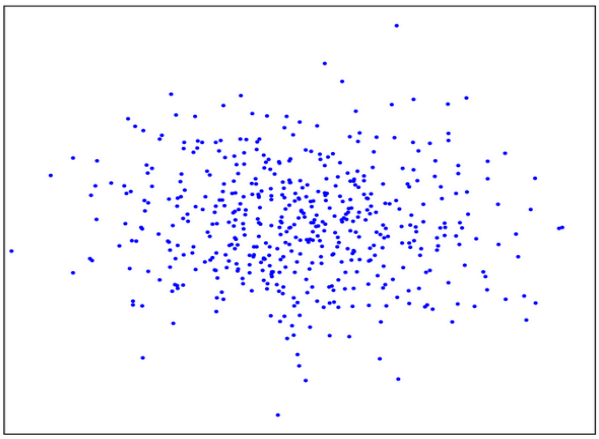


Figure 2-1 : Données positives **[29]**

La figure 2-01 montre un ensemble de points générés par une certaine distribution, que nous appellerons positifs. Tous les autres points possibles qui n'appartiennent pas à la distribution positive seront appelés négatifs. Essayez de tracer mentalement une ligne séparant les points positifs présentés de tous les points négatifs possibles. En passant, cette tâche est appelée "détection d'anomalies".

La réponse est ici :

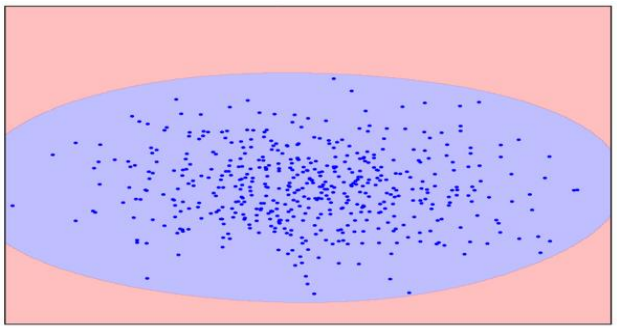
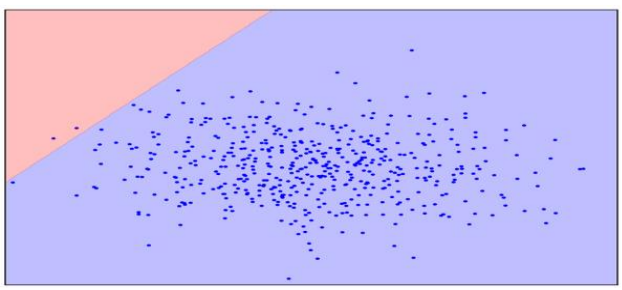


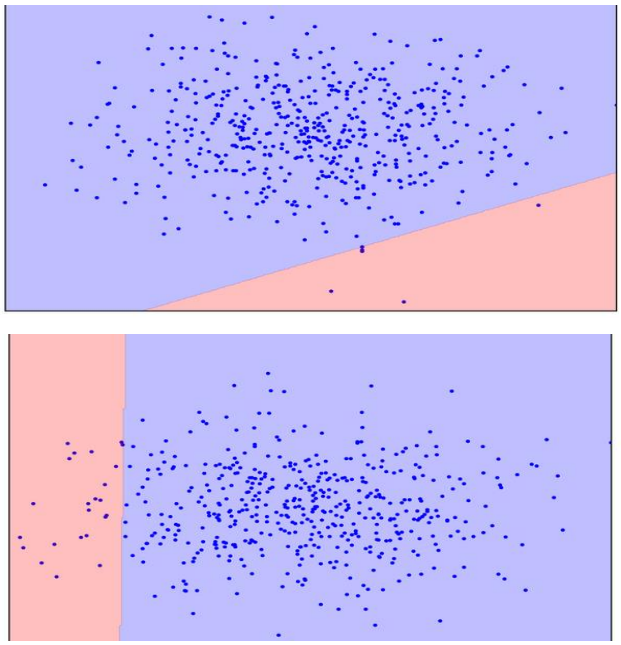
Figure 2-2 [29]

Vous avez peut-être imaginé quelque chose comme **la figure 2-02** encerclé les données dans une ellipse. En fait, c'est le nombre de méthodes de détection d'anomalies qui fonctionnent.

Maintenant, changeons un peu le problème: ayons des informations supplémentaires qu'une ligne droite devrait séparer les points positifs des points négatifs. Essayez de le dessiner dans votre esprit.

La réponse est ici :





Figures 2-3: (One-Class SVM) [29]

Dans le cas d'une ligne de démarcation droite, il n'y a pas de réponse unique. On ne sait pas dans quelle direction tracer une ligne droite.

Maintenant, on ajouter quelques points non alloués à **la figure 2-04** qui contiennent un mélange de positif et de négatif. Pour la dernière fois, on vous demander de faire un effort mental et d'imaginer une ligne séparant les points positifs et négatifs. Mais cette fois, vous pouvez utiliser des données non étiquetées!.

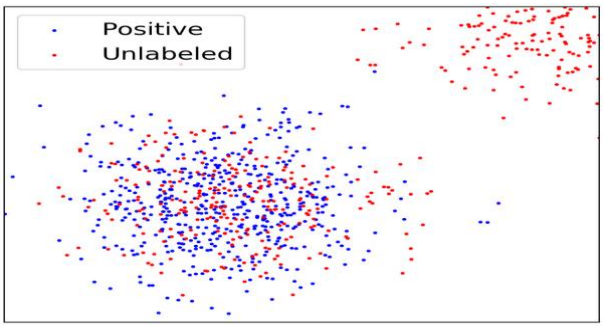


Figure 2-4 : Points positifs (bleus) et non étiquetés (rouges). Les points non marqués sont composés de points positifs et négatifs. **[29]**

La réponse est ici :

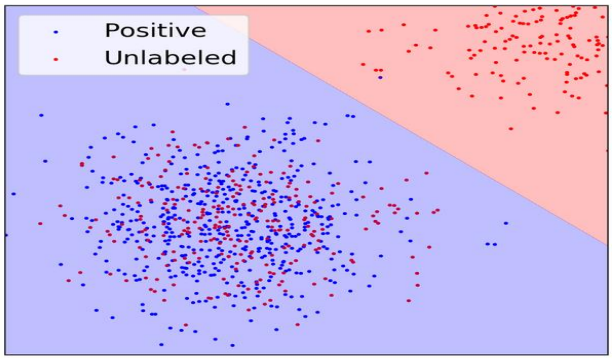


Figure 2-5 [29]

C'est devenu plus facile! Bien que nous ne sachions pas à l'avance comment chaque point non marqué est généré, nous pouvons grossièrement les marquer en les comparants avec des points positifs. Les points rouges qui ressemblent à du bleu sont probablement positifs. Au contraire, les dissemblables sont probablement négatifs. En conséquence, bien que nous ne disposions pas de données négatives «propres», des informations à leur sujet peuvent être obtenues à partir du mélange non étiqueté et utilisées pour une classification plus précise. C'est ce que font les algorithmes d'apprentissage Positive-Unlabeled, dont on va voir dans notre thème de fin d'étude master.

B- L'apprentissage positif-non étiqueté (PU) peut être traduit par «apprendre à partir de données positives et non étiquetées». En fait, l'apprentissage PU est un analogue d'une classification binaire pour les cas où il y a des données étiquetées d'une seule des classes, mais un mélange non étiqueté de données des deux classes est disponible. Dans le cas général, nous ne savons même pas combien de données dans le mélange correspondent à une classe positive et combien à une classe négative. Sur la base de tels ensembles de données, nous voulons construire un classifieur binaire: le même qu'en présence de données pures des deux classes.

Comme exemple de jouet, considérons un classifieur pour les photos de chats et de chiens. Nous avons quelques photos de chats et bien d'autres photos de chats et de chiens. En sortie, nous voulons obtenir un classifieur: une fonction qui prédit pour chaque image la probabilité qu'un chat soit représenté. En même temps, le classifieur peut marquer notre mélange existant d'images pour chats et chiens. Dans le même temps, il peut être difficile/coûteux/long/impossible de marquer manuellement des images pour entraîner le classifieur, vous voulez le faire sans lui.

L'apprentissage PU est une tâche assez naturelle. Les données trouvées dans le monde réel sont souvent sales. La capacité d'apprendre à partir de données sales semble être un hack de vie utile de qualité relativement élevée.

I-2- Application de l'apprentissage PU : [29]

On divisera de manière informelle les cas dans lesquels l'apprentissage PU peut être utile en trois catégories.

La première catégorie est probablement la plus évidente: l'apprentissage PU peut être utilisé à la place de la classification binaire habituelle. Dans certaines tâches, les données sont intrinsèquement légèrement sales. En principe, cette contamination peut être ignorée et un classifieur conventionnel peut être utilisé. Mais il est possible de modifier très peu la fonction de perte lors de l'entraînement du classifieur ou même les prédictions elles-mêmes après l'entraînement, et la classification devient plus précise.

À titre d'exemple, considérons l'identification de nouveaux gènes responsables du développement des gènes de la maladie. L'approche standard consiste à traiter les gènes de la maladie déjà trouvés

comme positifs et tous les autres gènes comme négatifs. Il est clair que des gènes de maladie non encore trouvés peuvent être présents dans ces données négatives. De plus, la tâche elle-même est de trouver ces gènes de maladies parmi les données «négatives». Cette contradiction interne est remarquée ici. Les chercheurs se sont écartés de l'approche standard et ont considéré les gènes non liés à des gènes de maladies déjà découverts comme un mélange non marqué, puis ont appliqué des méthodes d'apprentissage PU.

Dans la deuxième catégorie, l'apprentissage PU peut être utilisé pour la détection d'anomalies comme un analogue de la classification à une classe (OCC). Nous avons déjà vu dans le préambule comment des données non étiquetées exactement peuvent aider. Toutes les méthodes, sans exception, sont obligées de faire une hypothèse sur la distribution des données négatives. Par exemple, il est sage d'encercler les données positives dans une ellipse (une hypersphère dans le cas multidimensionnel), en dehors de laquelle tous les points sont négatifs. Dans ce cas, nous supposons que les données négatives sont uniformément réparties ... Au lieu de faire de telles hypothèses, les méthodes d'apprentissage PU peuvent estimer la distribution des données négatives sur la base des données non étiquetées. Ceci est particulièrement important si les distributions des deux classes se chevauchent fortement.

La troisième catégorie d'apprentissage PU peut être attribuée à des problèmes dans lesquels ni la classification binaire ni la classification à classe unique ne peuvent être utilisées.

Selon la législation, des données complètes sur toutes les enchères de marchés publics sont accessibles au public pour tous ceux qui souhaitent passer un mois à les analyser. Ils ont collecté des données de plus d'un million d'enchères organisées de 2014 à 2018. Qui a mis, quand et combien, qui a gagné, pendant quelle période l'enchère a eu lieu, qui a tenu, ce qui a acheté - tout cela est dans les données. Mais, bien sûr, il n'y a pas d'étiquette «la corruption est ici», donc vous ne pouvez pas construire un classifieur ordinaire. Au lieu de cela, nous avons appliqué l'apprentissage PU. Hypothèse de base: les participants bénéficiant d'un avantage injuste gagneront toujours. En utilisant cette hypothèse, les perdants des enchères peuvent être considérés comme équitables (positifs) et les gagnants comme potentiellement malhonnêtes (non marqués). Lorsqu'elles sont configurées de cette manière, les méthodes d'apprentissage PU peuvent trouver des modèles suspects dans les données en fonction de différences subtiles entre les gagnants et les perdants.

II- Approche basée sur la similarité pour un apprentissage positif et non étiqueté [30] :

II -1- Introduction [30] :

Les méthodes traditionnelles d'apprentissage supervisé exigent que les deux exemples positifs et négatifs soient disponibles pour la formation.

Cependant, dans de nombreuses applications du monde réel [**31**], il n'est pas facile d'obtenir des exemples négatifs. Par exemple, dans la classification des pages Web, les utilisateurs peuvent marquer

leur favori Pages Web, mais ils ne veulent pas marquer les pages ennuyeuses qu'ils ne montrent aucune préférence. Par conséquent, l'apprentissage positif et non étiqueté (apprentissage PU) est étudié pour traiter la situation des exemples positifs et les apprentissages non étiquetés disponibles dans la phase de formation [**32**].

Dans l'apprentissage PU, les exemples négatifs n'étant pas disponibles, la plupart des travaux existants se concentrent sur l'identification de quelques exemples négatifs fiables à partir des exemples non étiquetés, afin que les méthodes d'apprentissage supervisé puissent être appliquées pour construire le classifieur. Cependant, il peut exister un certain nombre d'exemples non étiquetés qui peuvent ne pas être explicitement identifié comme positif ou négatif fiable (ils appellent-les exemples ambigus ici). Par rapport aux exemples qui peuvent être clairement classés comme positifs ou négatifs, les exemples ambigus sont plus susceptibles de se situer près de la limite de décision et de jouer un rôle essentiel dans la construction du classifieur. Par conséquent, il est essentiel de traiter de manière appropriée les exemples ambigus afin d'apprendre un classifieur précis.

Compte tenu des travaux d'apprentissage PU existants, différents stratégies ont été proposées pour faire face à l'ambiguïté des exemples. Les étiquettes des exemples ambigus étant difficiles à déterminer, un groupe d'ouvrages [**32**]**;** [**33**] **;** [**34**]**;** [**35**] exclut les exemples ambigus dans la phase d'apprentissage, et le classifieur est formé en utilisant uniquement les exemples positifs et quelques exemples négatifs. Par exemple, Spy-EM (Spy Expectation Maximization) [**32**] utilise une technique d'espionnage pour identifier certains exemples négatifs fiables à partir des exemples non étiquetés, puis EM est exécuté pour construire le classifieur en utilisant les exemples positifs et les exemples négatifs extraits.
Cependant, la capacité de classification de ces méthodes peut être limitée, puisque les exemples ambigus, qui peuvent contribuer à la construction du classifieur, sont exclus de l'apprentissage processus.

Un autre groupe d'œuvres comprend les exemples ambigus à apprendre le classifieur en leur affectant directement à la classe positive ou à la classe négative. Par exemple, dans LELC **[36]**, les exemples ambigus sont regroupés en micro-clusters. Pour chaque micro-cluster, les distances de ses exemples aux prototypes positifs et prototypes négatifs identifiés sont calculées. Sur la base d'une stratégie de vote, le micro-cluster (y compris tous ses exemples) est affecté à la classe dont le micro-cluster est le plus proche. Par compte tenu des exemples ambigus, LELC est plus performant que d'autres méthodes d'apprentissage PU **[36]**. Cependant, dans le LELC, il peut exister des micro-clusters, dans lesquels certains exemples sont biaisés vers la classe positive, tandis que les autres sont plus proches de la classe négative. Dans ce cas, en appliquant simplement l'ensemble d'exemples du micro-cluster à l'un des deux classes, une erreur de classification peut être encourue.

Une nouvelle approche, appelée l'apprentissage PU basé sur la similarité (SPUL), en utilisant les exemples ambigus comme un moyen efficace d'améliorer l'apprentissage PU. Au lieu d'éliminer les

exemples ambigus dans la phase d'apprentissage, l'idée est de traiter explicitement les exemples ambigus en considérant leur similitude à la fois avec la classe positive et la classe négative. Plus précisément, leur approche proposée fonctionne en trois étapes. Dans un premier temps, ils extraient les exemples négatifs fiables à partir des données non étiquetées et construire les prototypes positifs et négatifs représentatifs. Dans la deuxième étape, ils regroupent les exemples non étiquetés restants en micro-clusters et attribuent à chaque exemple deux poids de similarité, qui indiquer la similitude d'un exemple ambigu avec le respectivement la classe positive et la classe négative. Pour faire ça, les mécanismes basés sur la similarité locale et globale sont proposés pour générer les poids de similarité. Dans la troisième étape, ils étendent la machine à vecteurs de support standard (SVM) pour incorporer les exemples ambigus avec leurs poids de similarité dans une phase d'apprentissage, de sorte que les exemples ambigus peuvent contribuer différemment à la construction du classifieur en fonction de leurs poids de similarité. Extensif des expériences ont été menées pour étudier les performances de SPUL et les résultats statistiques montrent que SPUL surpasse les méthodes d'apprentissage PU de pointe.

II -2- travaux connexes [30]:

II -2.1- Apprentissage positif et non étiquette :

Ces dernières années, l'apprentissage PU a trouvé diverses applications dans l'exploration de texte en raison du fait que la collecte d'un grand ensemble de documents négatifs est toujours coûteuse et difficile. En passant brièvement en revue les travaux existants sur l'apprentissage PU dans ce qui suit.

Le premier groupe de travaux [**32 ; 33 ;34 ; 35**] adopte un cadre itératif pour extraire les exemples négatifs des exemples non étiquetés, et entraîne alternativement le classifieur. Par exemple, Spy-EM (Spy Expectation Maximization) [**32]** utilise une technique Spy pour extraire les exemples négatifs, et L'algorithme EM est utilisé pour former le classifieur de manière itérative. RocSVM (Rocchio-Support Vector Machine) [**34**] extrait les exemples négatifs fiables en utilisant la technique de recherche d'information Rocchio [**37**]. Dans cette catégorie, à l'exception des exemples positifs et des exemples extraits, les autres exemples ambigus sont exclus dans le processus de formation. Par conséquent, les performances peuvent être limitées.

Le deuxième groupe de travail n'inclut pas l'itératif cadre. Par exemple, méthode de classification à une classe [**38**] propose de construire une classe classifieur en utilisant uniquement les exemples positifs. Étant donné que les informations de données non étiquetées ne sont pas prises en compte, le classifieur à une classe est toujours inférieur aux méthodes basées sur la classification binaire [**34**]. Un autre exemple est LELC [**36**]. LELC regroupe les exemples ambigus en micro clusters, puis attribue tout un micro-cluster d'exemples à la classe dont le micro-cluster est le plus proche. Cependant, il peut y avoir des micro-clusters dans lesquels quelques exemples sont biaisés vers la classe positive et les autres exemples sont plus proches de la classe négative. Dans ce cas, l'application du micro cluster à l'une des deux classes peut entraîner une mauvaise classification.

II -2.2- Machine à vecteurs de support :

SVM [**39**] s'est avéré être un puissant outil de classification.

Brièvement en revue SVM comme suit.

Soit $S = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_{|S|}, y_{|S|})\}$ un ensemble de formation, où $x^i \in R^d$ et $y^i \in \{+1, -1\}$. SVM vise à rechercher un hyperplan séparateur optimal $w \cdot \varphi(x) + b = 0$, où $\varphi(x)$ est l'image de l'exemple x dans l'espace des caractéristiques. L'hyperplan de séparation optimal peut être obtenu en résolvant la fonction d'optimisation suivante :

$$\begin{aligned} \min \quad & F(\mathbf{w}, b, \xi_i) = \tfrac{1}{2}\mathbf{w} \cdot \mathbf{w} + C\tfrac{1}{2}\sum_{i=1}^{|S|} \xi_i \\ st. \quad & y_i(\mathbf{w} \cdot \phi(\mathbf{x}_i) + b) \geq 1 - \xi_i, \\ & \xi_i > 0, \quad i = 1, \ldots, |S|. \end{aligned} \qquad (1)$$

Où ξi sont des variables pour assouplir les contraintes de marge, et C est un paramètre pour équilibrer les erreurs de classification. Par en introduisant la fonction de Lagrange [**39**], le classifieur de décision peut être obtenu. Pour un exemple de test x, si $w \cdot \varphi(x) + b > 0$, il est classé dans la classe positive ; sinon, il est négatif.

Dans ce qui suit, le SVM a pour incorporer les exemples avec des poids de similarité dans une phase d'apprentissage, tels que les exemples ambigus peuvent contribuer différemment à la construction du classifieur.

II -3- Préliminaire [30] :

Soit S un ensemble d'apprentissage d'un problème d'apprentissage PU. Suppose que PS et US stockent les exemples positifs et les exemples non étiquette, respectivement. On a donc S = PS ∪ US. Pour les exemples ambigus, on save pas quel classe à laquelle il devrait appartenir, en représentant un exemple ambigu x en utilisant un modèle de données basé sur la similarité :

$$\{\mathbf{x}, (m^+(\mathbf{x}), m^-(\mathbf{x}))\}, \qquad (2)$$

Où m+(x) et m−(x) sont des poids de similarité qui représentent la similarité de x vers la classe positive et la classe négative, respectivement. On a $0 \leq m+(x) \leq 1$ et $0 \leq m-(x) \leq 1$. {x, (1, 0)} signifie que x est positif, tandis que {x, (0, 1)} indique que x est identifié comme étant négatif.

Pour {x, (m+(x), m−(x))}, où $0 < m+(x) < 1$ et $0 < m-(x) < 1$, cela implique que la similitude de x vers la classe positive et la classe négative sont toutes deux considérées.

En utilisant le modèle de données basé sur la similarité, ils peuvent générer les poids de similarité pour les exemples ambigus basés sur les exemples positifs et extraits négatifs. Ces ambiguïtés Les exemples et leurs poids de similarité sont ensuite incorporés dans un modèle d'apprentissage basé sur SVM.

II -4- Approche d'apprentissage PU basée sur la similarité [30] :

L'apprentissage PU viser à construire un classifieur en utilisant les exemples positifs et les exemples non étiquetés. Il a été trouvé diverses applications dans le domaine de l'exploration de texte. L'approche SPUL proposée fonctionne en trois étapes.

1-Dans un premier temps, extraction les exemples négatifs fiables et construction les exemples représentatifs positifs et négatifs prototypes.

2-Dans la deuxième étape, regroupement les non-étiquetés restants données (exemples ambigus) en micro-grappes et attribution des poids de similarité aux exemples ambigus. Des mécanismes basés sur la similarité locale et basés sur la similarité globale sont proposés pour générer les poids de similarité.

3-Dans la troisième étape, ils étendent le SVM standard pour incorporer les exemples ambigus et leur similarité poids dans la phase d'apprentissage pour construire un classifieur.

Dans ce qui suit, la présentation des informations détaillées de la au-dessus de trois étapes.

II -4.1- Étape 1 : Extraction des exemples négatifs :

Dans un premier temps, l'extraction des exemples négatifs fiables et les placés dans le sous-ensemble **NS**. En plus les exemples positifs et négatifs sont utilisés pour établir le représentant des prototypes positifs et prototypes négatifs.

Tout d'abord, l'extraction les exemples négatifs fiables des données non étiquetées. Comme LELC, il intègre la technique de Spy [**32**] et la technique de Rocchio [**34**] pour extraire les exemples négatifs les plus fiables. Soit les sous-ensembles $\mathbf{S_1}$ et $\mathbf{S_2}$ contiennent les exemples négatifs fiables extraits par la technique de Spy et la technique de Rocchio. Les exemples sont classés comme négatifs fiables seulement si les deux techniques conviennent qu'elles sont négatives. $\mathbf{NS = S_1 \cap S_2}$, où le sous-ensemble **NS** contient les exemples négatifs fiables. Une fois les exemples négatifs fiables déterminés, la suppression du sous-ensemble de données non étiqueté, c'est-à-dire, **US = US – NS**.

De plus, les prototypes positifs représentatifs et négatifs sont ensuite mis en place en regroupant les exemples négatifs fiables en micro-clusters. Plus précisément, le clustering K-mean est utilisé pour regrouper les exemples de **NS** en m micro-clusters, désignés par $\mathbf{NS_1, NS_2, \ldots, NS}_m$ où **m = t * |NS|/(|US| + |NS|)** et t est fixé à 30 dans les expériences, comme recommandé dans [**40; 36**]. Ensuite, le $K^{ème}$ prototype représentatif positif, noté p_k, et le $K^{ème}$ prototype représentatif négatif, noté n_k, sont construits comme suit :

$$\mathbf{p}_k = \alpha \frac{1}{|PS|} \sum_{\mathbf{x} \in PS} \frac{\mathbf{x}}{\|\mathbf{x}\|} - \beta \frac{1}{|NS_k|} \sum_{\mathbf{x} \in NS_k} \frac{\mathbf{x}}{\|\mathbf{x}\|}, \quad (3)$$

$$\mathbf{n}_k = \alpha \frac{1}{|NS_k|} \sum_{\mathbf{x} \in NS_k} \frac{\mathbf{x}}{\|\mathbf{x}\|} - \beta \frac{1}{|PS|} \sum_{\mathbf{x} \in PS} \frac{\mathbf{x}}{\|\mathbf{x}\|}, \quad (4)$$

$$k = 1, \ldots, m.$$

où $\| \mathbf{x} \|$ représente la norme de l'exemple $\boldsymbol{x}$; paramètres $\boldsymbol{\alpha}$ et $\boldsymbol{\beta}$ sont fixés à 16 et 4, respectivement, comme recommandé dans [**40 ; 36**].

Après **l'étape 1**, l'obtention des exemples négatifs fiables dans **NS**, **m** prototypes positifs représentatifs $\boldsymbol{p_k}$ et **m** prototypes négatifs représentatifs $\boldsymbol{n_k}$.

II -4.2- Étape 2 : Génération du poids de similarité

Dans cette étape, la création des deux poids de similarité **m+(x)** et **m−(x)** pour les exemples du sous-ensemble **US**, de sorte que les exemples ambigus puissent être incorporés dans la phase d'apprentissage en considère leur ressemblance avec la classe positive et les classe négative. Plus précisément, le regroupement d'abord les exemples dans sous-ensemble US en r micro-clusters, c'est-à-dire **US1, US2, . . . ,USr**, où **r** est défini comme **r = t * |US|/(|US| + |NS|)** et encore ils avez fixé **t à 30** dans les expériences. Ensuite, la similitude les poids **m+(x_i)** et **m−(x_i)** sont générés pour chaque exemple dans les sous-ensembles **US_i (i = 1, . . . , r)**. Pour générer la similitude pondérations, mettant en avant les similarités locales et globales schémas basés sur la similarité dans ce qui suit.

Schéma de génération de poids de similarité locale :

Dans ce schéma, la génération des poids de similarité en capturant les informations de données locales. Pour chaque micro-cluster **USj (j = 1, 2, . . . , r)**, supposant qu'il existe $\boldsymbol{l_p^j}$ exemples similaires au prototype positif le plus proche $\boldsymbol{p_k}$, et $\boldsymbol{l_n^j}$ exemples similaires au prototype négatif le plus proche $\boldsymbol{n_k}$.

Autrement dit, pour les $\boldsymbol{l_p^j}$ exemples, ils ont : $\max_{k=1}^{m} Sim(\mathbf{x}, \mathbf{p}_k) > \max_{k=1}^{m} Sim(\mathbf{x}, \mathbf{n}_k)$ (5)

où **Sim(., .)** est calculé comme suit : $Sim(\mathbf{x}, \mathbf{p}_k) = \frac{\mathbf{x} \cdot \mathbf{p}_k}{\|\mathbf{x}\| \cdot \|\mathbf{p}_k\|}$.

De même, pour les exemples, $\boldsymbol{l_n^j}$ on a $\max_{k=1}^{m} Sim(\mathbf{x}, \mathbf{p}_k) < \max_{k=1}^{m} Sim(\mathbf{x}, \mathbf{n}_k)$. (6)

Sur la base des fonctions ci-dessus, les poids de similarité pour les données ambiguës en $\boldsymbol{US_j}$ sont calculés comme suit :

$$m^+(\mathbf{x}_i) = \frac{l_p^j}{l_p^j + l_n^j}, \mathbf{x}_i \in US_j \quad (7)$$

$$m^-(\mathbf{x}_i) = \frac{l_n^j}{l_p^j + l_n^j}, \mathbf{x}_i \in US_j \quad (8)$$

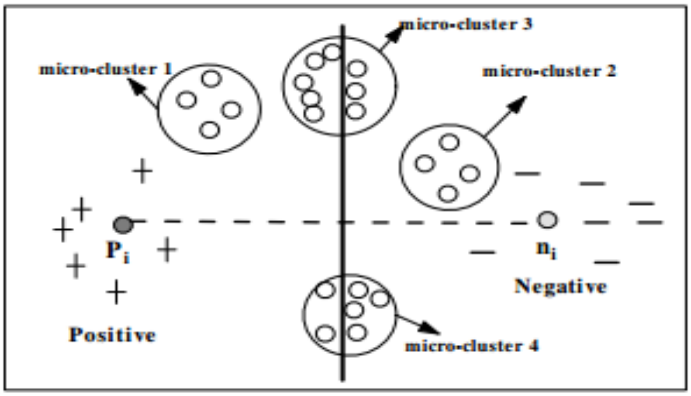


Figure 2.6 : Illustration de l'attribution du poids de similarité au micro clusters dans le schéma de production local. "+" représente l'exemple positif. "-" désigne l'exemple négatif fiable. "o" représente l'exemple ambigu.) **[30]**

La figure 2.6 présente un exemple d'attribution de la similarité poids aux données ambiguës. Sur la base des équations **(7)** et **(8),** les exemples des micro-grappes 1, 2, 3 et 4 se voient attribuer poids (1, 0), (0, 1), $(\frac{5}{8},\frac{3}{8})$ et $(\frac{1}{3},\frac{2}{3})$, respectivement. Distingué de LELC, qui attribue directement tout un micro-cluster d'exemples à une classe, SPUL permet l'ambiguïté exemples ayant des poids différents associés à la classe positive et à la classe négative, de sorte que la similitude des exemples ambigus envers les deux classes peut être envisagée. L'avantage du système de production locale est qu'il est simple à mettre en œuvre. Cependant, il ne peut pas distinguer la différence d'exemples dans le même micro-cluster. Les exemples du même micro-cluster ont exactement les mêmes poids envers les deux classes. En effet, les poids de similarité des exemples d'un même micro-cluster peuvent être différents, puisque ils sont situés physiquement différents.

Schéma global de génération de poids de similarité

Pour considérer l'emplacement des exemples ambigus, La proposition d'un schéma de génération global pour attribuer des poids aux exemples ambigus.

Pour l'exemple ambigu $\mathbf{x_i}$ dans le sous-ensemble **US**, en calculant d'abord sa similarité avec chacun des exemples représentatifs prototypes positifs et négatifs.

$$Sim(\mathbf{x}_i, \mathbf{p}_k) = \frac{\mathbf{x}_i \cdot \mathbf{p}_k}{\|\mathbf{x}_i\| \cdot \|\mathbf{p}_k\|}, \quad k = 1, 2, \ldots, m \qquad (9)$$

$$Sim(\mathbf{x}_i, \mathbf{n}_k) = \frac{\mathbf{x}_i \cdot \mathbf{n}_k}{\|\mathbf{x}_i\| \cdot \|\mathbf{n}_k\|}, \quad k = 1, 2, \ldots, m. \qquad (10)$$

Pour $\mathbf{x_i} \in$ **US**, les poids correspondants vers la classe positive et la classe négative sont calculés comme suit :

$$m^+(\mathbf{x}_i) = \frac{\sum_{k=1}^{m} Sim(\mathbf{x}_i, \mathbf{p}_k)}{\sum_{k=1}^{m} (Sim(\mathbf{x}_i, \mathbf{p}_k) + Sim(\mathbf{x}_i, \mathbf{n}_k))}, \qquad (11)$$

$$m^-(\mathbf{x}_i) = \frac{\sum_{k=1}^{m} Sim(\mathbf{x}_i, \mathbf{p}_k)}{\sum_{k=1}^{m} (Sim(\mathbf{x}_i, \mathbf{p}_k) + Sim(\mathbf{x}_i, \mathbf{n}_k))}. \qquad (12)$$

Le schéma de génération global traite chaque exemple ambigu dans le sous-ensemble **US** différemment et les poids sont calculés basé sur les emplacements des exemples vers le représentant prototypes positifs et négatifs, respectivement.

II -4.3- Étape 3 : Construction d'un classifieur basé sur SVM

Après avoir effectué les deux étapes ci-dessus, chaque exemple ambigu se voit attribuer deux poids de similarité : $\mathbf{m^+(x_i)}$ **et** $\mathbf{m^-(x_i)}$.

Dans ce qui suit, on donneront une nouvelle formulation de SVM par incorporant les données dans l'ensemble positif **PS**, l'ensemble négatif **NS**, exemple ambigu **US** et les poids de similarité dans un modèle d'apprentissage basé sur SVM.

Formulation primordiale

Comme les poids de similarité $\mathbf{m^+(x_i)}$et $\mathbf{m^-(x_i)}$ indiquent la différents degrés de similitude pour un exemple ambigu vers la classe positive et la classe négative, respectivement, la fonction d'optimisation peut être formulée comme suit :

$$
\begin{aligned}
\min \quad & F(\mathbf{w}, b, \xi) \\
& = \tfrac{1}{2}\mathbf{w}\cdot\mathbf{w} + C_1 \textstyle\sum_{PS} \xi_i + C_2 \sum_{US} m^+(\mathbf{x}_j)\xi_j \\
& + C_3 \textstyle\sum_{US} m^-(\mathbf{x}_k)\xi_k + C_4 \sum_{NS} \xi_g \\
s.t. \quad & \mathbf{w}\cdot\phi(\mathbf{x}_i) + b \geq 1 - \xi_i, \qquad \mathbf{x}_i \in PS \\
& \mathbf{w}\cdot\phi(\mathbf{x}_j) + b \geq 1 - \xi_j, \qquad \mathbf{x}_j \in US \\
& \mathbf{w}\cdot\phi(\mathbf{x}_k) + b \leq -1 + \xi_k, \quad \mathbf{x}_k \in US \\
& \mathbf{w}\cdot\phi(\mathbf{x}_g) + b \leq -1 + \xi_g, \quad \mathbf{x}_g \in NS \\
& \xi_i \geq 0,\ \xi_j \geq 0,\ \xi_k \geq 0,\ \xi_g \geq 0, \qquad (13)
\end{aligned}
$$

où **C1**, **C2**, **C3** et **C4** sont des facteurs de pénalité contrôlant la compromis entre la marge de l'hyperplan et les erreurs. **ξi, ξj**, **ξk** et **ξg** sont les termes d'erreur. $\mathbf{m^+(x_j)\xi_j}$ et $\mathbf{m^-(x_k)\xi_k}$ peuvent être considérées comme des erreurs avec des poids différents. Notez que, une plus petite valeur de $\mathbf{m^+(x_i)}$peut réduire l'effet du paramètre **ξi**, de sorte que l'exemple correspondant ***xi*** devient moins significatif vers la classe positive.

Double problème

Supposons que $\boldsymbol{\alpha_i}$, $\boldsymbol{\alpha_j}$, $\boldsymbol{\alpha_k}$ et $\boldsymbol{\alpha_g}$ soient des multiplicateurs de lagrange. Pour simplifier la présentation, la définition de certaines notations est la suivante :

$$
\alpha_i^+ = \begin{cases} \alpha_i, & \mathbf{x}_i \in PS \\ \alpha_j, & \mathbf{x}_j \in US \end{cases} \quad C_i^+ = \begin{cases} C_1, & \mathbf{x}_i \in PS \\ C_2 m^+(\mathbf{x}_j), & \mathbf{x}_j \in US \end{cases}
$$

$$
\alpha_j^- = \begin{cases} \alpha_k, & \mathbf{x}_k \in US \\ \alpha_g, & \mathbf{x}_g \in NS \end{cases} \quad C_j^- = \begin{cases} C_2 m^-(\mathbf{x}_k), & \mathbf{x}_k \in US \\ C_3, & \mathbf{x}_g \in NS \end{cases}
$$

Sur la base des définitions ci-dessus, en mettant $\mathbf{S_+}$ = **PS** ∪ **US**, $\mathbf{S_-}$ = **US** ∪ **NS** et $\mathbf{S_*}$ = $\mathbf{S_+}$ ∪ $\mathbf{S_-}$.

Le formulaire (13) peut être obtenu comme suit :

$$
\begin{aligned}
\max \ & F(\alpha) = \textstyle\sum_{\mathbf{x}_i \in S_*} \alpha_i - \frac{1}{2} \sum_{\mathbf{x}_i, \mathbf{x}_j \in S_*} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \\
s.t. \quad & 0 \leq \alpha_i \leq C_i^+, \quad \mathbf{x}_i \in S_+ \\
& 0 \leq \alpha_j \leq C_j^-, \quad \mathbf{x}_j \in S_- \\
& \textstyle\sum_{\mathbf{x}_i \in S_+} \alpha_i - \sum_{\mathbf{x}_j \in S_-} \alpha_j = 0, \qquad (14)
\end{aligned}
$$

où $\boldsymbol{K(x_i, x_j)}$ est le produit scalaire de ***φ(xi)*** et ***φ(xj)***. Après avoir résolu le problème en (14), w peut être obtenu comme le suivant :

$$
\mathbf{w} = \textstyle\sum_{\mathbf{x}_i \in S_+} \alpha_i^+ \phi(\mathbf{x}_i) - \sum_{\mathbf{x}_j \in S_-} \alpha_j^- \phi(\mathbf{x}_j). \qquad (15)
$$

En utilisant les conditions de Karush-Kuhn-Tucker (KKT) [**39**], ***b*** peut être obtenu. Pour un exemple de test, $\boldsymbol{si\ f(x) = w \cdot \varphi(x) + b > 0}$ est vrai, il appartient à la classe positive. Si non c'est négatif.

Chapitre 3 :

Les Machines à Vecteurs Supports, Essaim (swarm) & Algorithme de Recherche Crow (CSA).

I- Les Machines à Vecteurs Supports [05] :

I-1- généralité autour SVM [05] :

Les Machines à Vecteurs de Support (SVM - Support Vector Machine) ont été introduites par **(Vapnik, 1995)** [**41**]**,,** et en catégorisation de texte par **(Joachims, 1998)** [**42**]**.** Il s'agit donc d'une classe récente de méthodes d'apprentissage automatique.

Le SVM revient à chercher un hyperplan dont la distance entre les exemples d'apprentissage positifs et négatifs est maximale. L'hyperplan optimal séparant les points de deux classes est celui qui passe « au milieu » de ces classes, c'est-à-dire dont la distance aux points les plus proches est maximale. Ces exemples les plus proches qui suffisent à déterminer cet hyperplan sont appelés ***vecteurs de support***, ou encore exemples critiques. La distance séparant l'hyperplan de ces points est appelée « marge ».

I-1-1- Cas des classes linéairement séparables : Soit **S** un ensemble de points linéairement séparables, $\mathbf{S = \{x_i \in R^n \mid i=1,\ldots, L\}}$. Chaque point x_i appartient à une classe $y_i \in \{-1,+1\}$. Un hyperplan sépare l'ensemble **S** selon les deux classes. Cet hyperplan séparateur est défini en fonction du vecteur de poids w (vecteur normal à l'hyperplan), b et $\|w\|$ (*la norme euclidienne de* w) ; l'hyperplan vérifie l'équation **(Jalam, 2003)** [**11**],

$$w \cdot x + b = 0$$

$$\text{tel que} \begin{cases} w \cdot x_i + b \geq +1 & \text{si } y_i = +1 \\ w \cdot x_i + b \leq +1 & \text{si } y_i = -1 \end{cases} \quad \text{.....(01)}$$

pour **i=1,….., L** ; où le produit scalaire w. x_i est calculé comme

$$w \cdot x = \sum_i w_j x_j \text{pour } j = 1, \cdots, n \quad \text{.....(02)}$$

Pour cet hyperplan, la marge vaut $1/\|w\|$, et donc la recherche de l'hyperplan optimal revient à minimiser$\|w\|$, soit à résoudre le problème suivant :

$$\begin{cases} \text{minimiser} & \frac{1}{2}\|w\|^2 \\ \text{sous les contraintes} & \begin{cases} w \cdot x_i + b \geq +1 & \text{si } y_i = +1 \\ w \cdot x_i + b \leq +1 & \text{si } y_i = -1 \end{cases} \text{ pour } i = 1, 2, \cdots, l \end{cases} \quad \text{.....(03)}$$

En écrivant le lagrangien, on montre que la solution f s'écrit sous la forme

$$h(x) = w_0 \cdot x + b = \sum_i \alpha_i x_i x + b \quad \text{.....(04)}$$

où α_i est le multiplicateur de Lagrange associé à l'exemple **i**. Les x_i qui interviennent dans la solution

sont nommés vecteurs de support. On note que l'ensemble de ces points **SV =** x_i pour **i =1,….,m** avec **m≤1**. Ce sont les points de **S** les plus proches de l'hyperplan qui sont suffisent à déterminer cet hyperplan optimal (voir **la figure 3.1**)

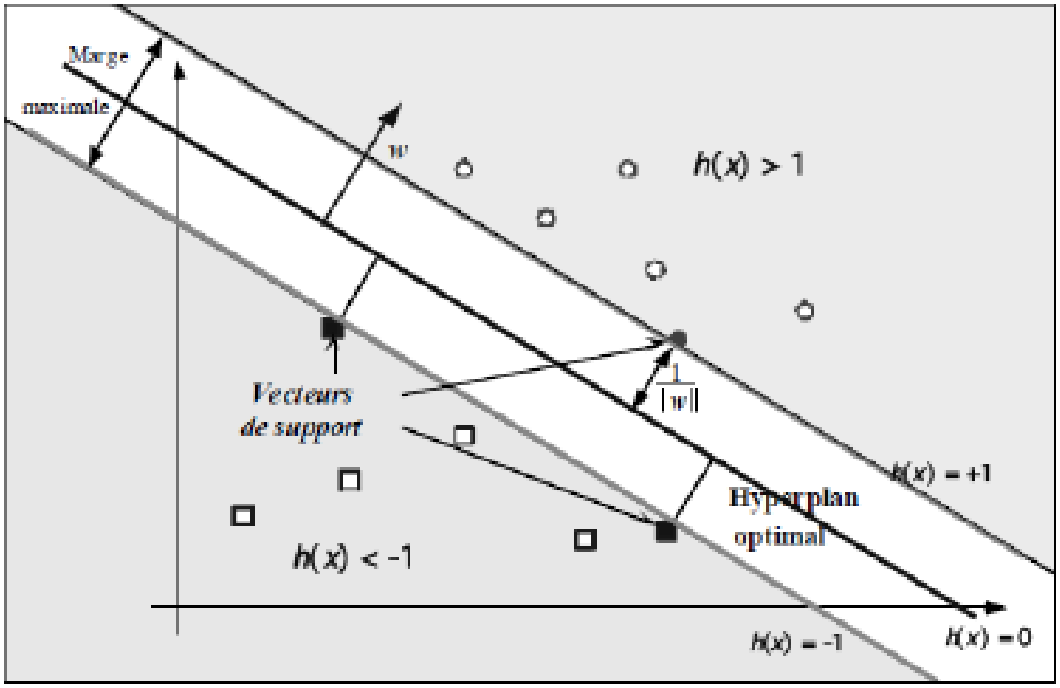


Figure 3.1: L'hyperplan optimal de la méthode SVM [**05**].

I-1-2- Cas des classes non séparables : cette approche s'étend au cas où les données ne sont pas séparables grâce à des variables d'écart qui permettent à certains points de se situer du mauvais côté de la frontière. De plus, les SVM s'étendent très élégamment pour construire des modèles non linéaires en enrichissant l'espace de représentation. Pour cela, on peut substituer le produit scalaire dans **la formule 04** par une fonction noyau symétrique ***K(x, y)*** et obtenir un comportement non linéaire,
la table 3.1 montre quelques exemples de noyaux.

La solution s'exprime sous la forme : $h(x) = \sum_i \alpha_i x_i x + b$		
Fonction noyau	**Forme fonctionnelle**	**Commentaire**
- polynomiale	$K(x,y) = (x.y + c)^n$	La puissance n est déterminée *a priori* par l'utilisateur.
- fonctions à base radiale	$K(x,y) = \exp\left(-\frac{\|x-y\|^2}{2\sigma^2}\right)$	L'écart type σ^2, commun à tous les noyaux, est spécifié *a priori* par l'utilisateur.
- fonctions sigmoïdes	$K(x,y) = \tanh((a(x \cdot y) - b))$	Le théorème de Mercer n'est vérifié que pour certaines valeurs de a et b.

La Table 3.1: Les fonctions noyau les plus courantes avec leurs paramètres [**05**].

La méthode de SVM est facile à l'emploi, et a montré une très grande performance par rapport aux autres méthodes indépendamment de la dimension de l'espace des termes. Cette méthode est applicable pour des tâches de classification à deux classes, mais il existe des extensions pour la classification multi classe.

II- Les algorithmes d'intelligence en essaim (swarm) [43]:

II -1- Introduction [43]:

Swarm (essaim) **I**ntelligence **(SI)** est une technique de **(IA)** qui s'interagir d'une manière simple et qui suit quelques règles simples aussi **[44]**. Quelques algorithmes SI les plus populaires sont **Particle Swarm Optimization (PSO)**, **Artificial Bee Optimization** des **Colonies (ABC), Optimisation des Colonies de fourmis(Ant) (ACO),** Algorithme **Firefly (FFA)** et **recherche de coucou (CS) (Cuckoo Search).** Les méthodes SI ont beaucoup de succès dans le domaine de l'optimisation.

Les algorithmes SI peuvent être utilisés dans de nombreux problèmes du monde réel tels que planification, optimisation, clustering et routage. Algorithmes SI ont plus d'avantages comme l'évolutivité, l'adaptabilité, la collective robustesse et simplicité individuelle. SI est utile pour résoudre les problèmes complexes tels que les méthodes traditionnelles. De plus, SI a peu limitations dans les applications urgentes, réglage des paramètres et stagnation **[45]**. Il est donc nécessaire d'étudier les algorithmes SI pour comprendre et l'améliorer le système existant. Dans cette étude, l'analyse de quelques algorithmes SI populaires tels que (PSO, ABC, ACO, FFA et CS) est résumer avec les paramètres de contrôle critiques et la distribution aléatoire qui est utilisée dans les algorithmes SI.

II -2-Travaux connexes [43]:

Les algorithmes basés sur SI sont un sous-ensemble **d'algorithmes bio-inspirés (BAs),** qui lui-même est un sous-ensemble **d'algorithmes inspirés de la nature (NAs)** comme illustré à **la figure 3.2.** Tous les NAs généralement stochastiques et développés on inspirant des éléments naturels tels que les animaux, les insectes, et plantes. Les NAs comprennent des sous-classes comme les BA et la physique ou algorithmes basés sur la chimie. Le sous-ensemble le plus populaire de NA est BA. Les applications basées sur la chimie ou la physique comprennent les algorithmes comme l'algorithme Harmony Search (HS), Simulé Recuit (SA = Simulated Annealing) et trou noir (BH = Black Hole) **[46]**.

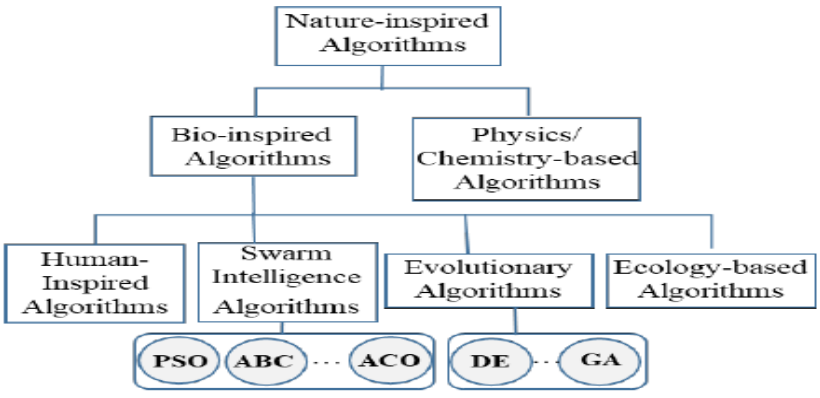


Figure 3.2: Structure hiérarchique des algorithmes SI **[43]**.

De plus, les BA sont classés en algorithmes SI, comme Evolutionary Algorithmes (EA), algorithmes inspirés par l'homme basés sur l'écologie. EAs inclut des algorithmes comme l'évolution différentielle (DE) et algorithme génétique (GA). SI est l'intelligence collective de groupes d'agents simples tels que les insectes, les poissons, les oiseaux, les bactéries, vers et autres animaux en fonction de leur comportement dans la vie réelle **[47]**. SI comprend des algorithmes tels que PSO, ABC, ACO, BA, CSS, FFA.

II -3- Domaines d'application et performances des algorithmes SI[43]:

Les résultats comparatifs des algorithmes SI sont résumés dans **Tableau 3.2** et le classement des applications ci-dessous.

II -3- 1 Problème du voyageur de commerce (TSP= Travelling Salesman Problem) :

Le TSP est le problème rencontré par un vendeur qui, partant d'une ville particulière, a pour mission de trouver le plus court possible (aller-retour) à travers un ensemble donné de villes clientes.

Le vendeur a pour mandat de visiter chaque ville une fois avant retournant finalement à town/city de départ. Le TSP peut être représenté par un graphe pondéré complet G = (V, E) avec V étant l'ensemble des nœuds (villes) et E étant l'ensemble des arrêtes pleinement reliant les nœuds dans le graphe **G [47, 48].**

Dans le **tableau 3.2**, la mise en œuvre de TSP avec des algorithmes SI (GA, PSO, ACO et ABC) est comparée et l'étude indique qu'ABC obtenu de meilleures performances que les autres. ACO est recommandé pour moins de 80 villes **[49]**.

II -3- 2- Sélections de fonctionnalités (FS) :

Pour apprendre des données, la dimensionnalité des données doit être réduite en premier. FS se concentre sur la suppression des éléments non pertinents et sur l'ensemble des fonctionnalités redondantes pour réduire la quantité de données **[50]**. Comme la montre le **tableau 3.2**, deux études suggèrent ce champ de sélection de fonctionnalités doit être étudié davantage. Car les résultats expérimentaux des algorithmes FS en SI obtenus sur des différents problèmes du monde réel sont plus difficiles à comparer et ils ont presque les mêmes performances **[51]**.

II -3- 3- Apprentissage de l'essaim de robots :

Le tableau 3.2 montre l'étude d'apprentissage de l'essaim de robots qui compare les algorithmes SI incluent : l'algorithme de chauve-souris (BA = bat algorithm), le PSO, l'optimiseur de loup gris (GWO = grey wolf optimizer) en fonction de leurs stratégies d'apprentissage. Ce comparatif est une étude qui montre que PSO surpasse BA et que BA surpasse GWO en général. GWO est plus performant que PSO et BA sous un grand nombre de robots (NR= number of robots) et longue portée de communication (CR= communication range). L'augmentation du NR et du CR peut améliorer considérablement les performances de GWO **[52]**.

Application	Algorithmes comparé	Remarque	Meilleure performance Algorithme	Ref.
TSP	GA, PSO, ACO et ABC	ABC a réalisé de meilleures performances que les autres. ACO est recommandé pour moins de 80 villes.	ABC	**[49]**
FS	SI le plus populaire algorithmes	Les résultats expérimentaux obtenus sur différents problèmes du monde réel sont plus difficile à comparer.	Besoin à étudier plus.	**[50]**
	PSO, ACO, AFSA, ABC, FFA, et BA	Plus d'études sont fortement nécessaires dans le domaine de la caractéristique sélections/réductions à l'aide d'algorithmes méta-heuristiques à l'avenir.	Besoin à étudier plus.	**[51]**
Robot Essaim Apprentissage	BA, PSO, et GWO	Les stratégies d'apprentissage de l'algorithme SI sont comparées. Résultats : OSP surpasse BA et BA surpasse GWO en général. GWO fonctionne mieux que PSO et BA sous un grand nombre de robots(NR) et longue portée de communication (CR). Augmenter NR et CR peut améliorer considérablement les performances de GWO.	GWO effectue mieux sous grand NR et CR	**[52]**
Regroupement	PSO, FFA, CS, et BA	Analyse des données médicales : le clustering CS est le plus lent. Luciole est lent lorsque le nombre d'agents est important. OSP et les BA sont relativement plus rapides que les deux autres approches. Le temps d'exécution de ces quatre approches existantes n'est toujours pas acceptable.	PSO et BA sont mieux que les autres. Besoin à étudier plus.	**[54]**
	SI le plus populaire algorithmes	Data mining : Utilisation de l'algorithme PSO, dans un cadre autonome ou y compris les variations et les améliorations. C'est le regroupement primaire technique dans 66,4% des articles sélectionnés. D'autres plus performants les algorithmes sont Ant Colony Optimization (ACO), Artificial Bee Colony (ABC) et algorithme d'essaim de poissons artificiels (AFSA).	Meilleure performance Algorithme : PSO Deuxième niveau interprètes : ACO, ABC et AFSA	**[53]**
Planification	ABC, PSO, et ACO	Planification dynamique des tâches dans le cloud computing : l'algorithme ABC est le supérieur et surpasse les autres algorithmes. Le PSO et l'ACO peuvent être placés respectivement au deuxième et au troisième niveau.	ABC	**[55]**
	PSO et ACO	Cloud Scheduling : l'algorithme PSO dans la planification du cloud est un meilleur option par rapport à ACO.	PSO	**[56]**

La Table 3.2: Domaine d'application appliqué avec les algorithmes SI et les performances **[43]**.

II -3- 4- Regroupement:

Le processus de regroupement de données est au cœur de l'exploration de données sur de vrais problèmes. Le processus vise à diviser les données en groupes selon leurs similitudes et leurs dissemblances **[53]**. Il y a plusieurs algorithmes sont disponibles pour le clustering. Ici, ils étudient sur le clustering mis en œuvre à l'aide d'algorithmes SI.

Des algorithmes SI tels que PSO, FFA, CS et BA sont utilisés et comparés pour le clustering dans l'exploration de données médicales **[54]**. Les résultats de cette étude montrent que le clustering CS est

le plus lent. Le regroupement Luciole est lent lorsque le nombre d'agents est important. PSO et Bat sont relativement plus rapides que les deux autres approches. Le temps d'exécution de ces quatre approches existantes n'est toujours pas acceptable.

Figueiredo et al **[53]** comparent également l'essaim algorithmes d'intelligence les plus populaires pour le clustering dans l'exploration de données. Cette étude montre que l'algorithme PSO est la principale technique de clustering dans 66,4% des papiers sélectionnés et performe mieux que les autres. Les algorithmes performants de deuxième niveau sont Ant Colony Optimisation (ACO), colonie d'abeilles artificielles (ABC) et artificielle Algorithme d'essaim de poissons (AFSA).

II -3- 5- Planification:

La planification dynamique des tâches dans le cloud est considérée comme un NP-difficile, et de nombreux problèmes méta-heuristiques sont aptes à résoudre ce problème **[55]**.

GF Elhady et al **[55]**, ont étudié comparativement les algorithmes SI (ABC, PSO et ACO) dans la planification dynamique des tâches dans le cloud. Leur étude montre que l'algorithme ABC est supérieur et surpasse les autres algorithmes. Le PSO et l'ACO peuvent être mis au deuxième niveau et troisième niveau respectivement.

SJ Mohana et al **[56]**, effectue une analyse comparative de l'essaim techniques d'optimisation du renseignement (PSO et ACO) pour le cloud planification et résume leur observation sous forme d'algorithme PSO dans la planification cloud est une meilleure option par rapport à ACO.

II -4- Conclusion [43]:

Pour résumer, les algorithmes SI ont plus d'avantages tels que évolutivité, adaptabilité, robustesse collective et individuelle simplicité, SI ont la capacité de résoudre des problèmes complexes tels que la recherche du chemin le plus court, sélection des fonctionnalités, planification des tâches et regroupement.

III- Une nouvelle méthode métaheuristique pour résoudre les problèmes d'optimisation : Algorithme de Recherche Crow (CSA) [57].

III -1- Introduction [57]:

L'ingénierie de conception est définie comme un processus de prise de décision pour construire des produits qui répondent à des besoins spécifiques. Le plus souvent, les problèmes de l'ingénierie de conception incluent des fonctions objectives complexes avec un grand nombre de variables de décision. Les solutions réalisables sont l'ensemble de tous les plans caractérisés par toutes les valeurs possibles des paramètres de conception (variables de décision). Une technique d'optimisation essaie de trouver la solution optimale parmi toutes les possibilités solutions.

Les algorithmes métaheuristiques ont montré des performances prometteuses pour résoudre la plupart des problèmes d'optimisation du monde réel qui sont extrêmement non linéaire et multimodal.

Tous les algorithmes métaheuristiques utilisent un certain compromis entre la randomisation et la recherche locale **[58].** Ces algorithmes peuvent trouver de bonnes solutions à des problèmes d'optimisation difficiles, mais il n'y a aucune garantie que les solutions optimales puissent être atteintes. On espère que ces algorithmes fonctionnent la plupart du temps, mais pas tous le temps. Les algorithmes métaheuristiques pourraient convenir optimisation **[59].** Basé sur la convention de **Glover, tous les méthodes inspirées de la nature sont appelées métaheuristiques [60].**

La tendance actuelle est d'utiliser des algorithmes métaheuristiques inspirés de la nature pour résoudre des problèmes difficiles, et il a été démontré que les métaheuristiques sont étonnamment très efficaces **[58,59].** Pour cette raison, la littérature sur les métaheuristiques s'est considérablement développé deux dernières décennies **[61,62].** Certains des algorithmes métaheuristiques bien connus sont les suivants : algorithme génétique (AG) basé sur sélection **[63],** optimisation par essaim de particules (PSO) basée sur comportement des troupeaux d'oiseaux et des bancs de poissons **[64],** recherche d'harmonie (HS) basé sur le processus d'improvisation musicale [**65]**, algorithme de recherche de coucou basé sur le parasitisme du couvain de certaines espèces de coucous **[66],** algorithme de chauve-souris (BA) basé sur le comportement d'écholocation des microbats **[67],** optimiseur de recherche de groupe (GSO) basé sur la recherche du comportement d'animaux **[68],** algorithme de luciole (FA) basé sur les motifs de lumière clignotante des lucioles tropicales **[69],** etc.

Les corbeaux sont un genre d'oiseaux largement répandu qui sont maintenant considérés faire partie des animaux les plus intelligents du monde **[70,71].** En tant que groupe, les corbeaux montrent des exemples remarquables d'intelligence et obtiennent souvent de très bons résultats aux tests d'intelligence. Ils peuvent mémoriser visages, utiliser des outils, communiquer de manière sophistiquée et cacher et récupérer de la nourriture au fil des saisons **[70,72].**

Dans une volée de corbeaux, il y a un comportement qui a beaucoup de similitudes avec un processus d'optimisation. Selon ce comportement, les corbeaux cachent leur excès de nourriture dans certaines positions (cachettes) du l'environnement et récupèrent les aliments stockés quand c'est nécessaire. Les corbeaux sont des oiseaux gourmands puisqu'ils se succèdent pour obtenir une meilleure source nourriture. Trouver une source de nourriture cachée par un corbeau n'est pas une tâche facile car si un corbeau vis qu'un autre qui le suit, le corbeau essaie de tromper ce corbeau en allant à une autre position de l'environnement. Du point de vue de l'optimisation, les corbeaux sont des chercheurs, l'environnement est l'espace de recherche, chaque position de l'environnement correspond à une solution faisable, la qualité de la source de nourriture est fonction objective (forme physique) et la meilleure source de nourriture de l'environnement est la solution globale du problème. Sur la base de ces similitudes, **CSA tente de simuler le comportement intelligent du corbeau pour trouver la solution des problèmes d'optimisation.**

III -2- Algorithme de recherche des corbeaux (CSA) [57] :

Le CSA standard est un algorithme méta-heuristique inspiré des essaims, initialement proposé par **[57].**Cet algorithme a été principalement inspiré par le mécanisme de recherche des corbeaux dans la nourriture cachée des autres corbeaux.

Les corbeaux sont des oiseaux sociaux, connus pour être les plus intelligents, avec les plus grands rapports taille-cerveau corporelle de tous les autres oiseaux.

Les corbeaux font partie de la famille des corvidés, des oiseaux connus pour leur intelligence exceptionnelle, les corbeaux avaient un comportement social similaire à celui des humains. Il peut stocker de la nourriture en prévision de futures périodes de pénurie, au lieu de manger immédiatement. Les corbeaux sont appelés voleurs intelligents parce qu'ils prennent des mesures de sécurité supplémentaires pour cacher leur nourriture et voler la nourriture des autres.

Basés sur le comportement du corbeau, les quatre principes de CSA sont:

• Ils vivent dans le troupeau de plusieurs corbeaux,

• Ils stockent leur surplus de nourriture dans des cachettes,

• Ils se suivent pour voler la nourriture des autres,

• Ils protègent la nourriture de leur cache contre le pillage stochastique.

Le processus évolutif CSA émule le mécanisme mené par le troupeau de ***N*** corbeaux, lorsque chaque corbeau cache sa nourriture supplémentaire et recherche la nourriture cachée des autres corbeaux. L'algorithme CSA est calculé en plusieurs itérations $(it_{\max})$, A chaque itération la position d'un corbeau ***i*** représente une solution possible du problème d'optimisation.

Pour un problème de d- dimension, le vecteur de position i^{it} corbeaux est présenté par **l'éq. (1):**

$$X_i^{it} = \left[x_1^{it}, x_2^{it}, \cdots, x_d^{it} \right] \qquad (1)$$

Ou $it = 1,2,\cdots,it_{\max}$, $i = 1,2,\cdots,N$ et d est le nombre de dimensions.

Pour chaque corbeau, il existe une mémoire, dans laquelle il mémorise sa meilleure position visitée, celle-ci est présentée par **l'éq. (2):**

$$M_i^{it} = \left[m_1^{it}, m_2^{it}, \cdots, m_d^{it} \right] \qquad (2)$$

Les corbeaux se déplacent dans l'espace de recherche pour rechercher les meilleures sources de nourriture (cachettes = endroit a cachées). A chaque itération ***it***, le corbeau ***i*** décide de suivre le corbeau ***j*** (tiré au hasard dans le troupeau) pour accéder à sa meilleure cachette M_j^{it}. La position courante du corbeau ***i*** est mise à jour selon deux comportements : poursuite et évasion.

Dans le cas d'une poursuite, le corbeau ***j*** n'identifie pas que le corbeau i le suit. Ainsi, le corbeau ***i*** s'approchera de la cachette du corbeau ***j***. Pour le cas d'évasion, le corbeau ***j*** sait qu'il est suivi par le corbeau ***i***. Ainsi, le corbeau ***j*** trompera le corbeau **i** en se déplaçant vers une nouvelle position aléatoire

dans l'espace de recherche.

En utilisant ces cas, la position du corbeau ***i*** est mise à jour comme dans **l'éq. (3).**

Dans ce cas, la nouvelle position du corbeau ***i*** est obtenue comme suit :

$$X_i^{it+1} = \begin{cases} X_i^{it} + r_i \times fl_i^{it} \times (M_j^{it} - X_i^{it}) & r_j \geq AP^{it} \\ \text{a random position} & \text{otherwise} \end{cases} \quad (3)$$

Où r_i et r_j sont des nombres aléatoires avec une distribution uniforme entre 0 et 1, fl_j^{it} est la longueur de vol du corbeau i à l'itération ***it***, il indique l'ampleur du mouvement depuis la position actuelle du corbeau ***j*** vers la meilleure position mémoire du corbeau ***j***, en sélectionnant une petite valeur de fl_j^{it}, l'algorithme effectue une recherche locale autour de leur position courante. D'autre part, en sélectionnant de grandes valeurs, l'algorithme effectue une recherche globale (loin de leur position actuelle). Le paramètre AP_j^{it} dénote la probabilité de prise de conscience du corbeau ***j*** à l'itération *it*, ce paramètre joue un rôle important dans le mécanisme de recherche. De petites valeurs de ***AP*** augmentent l'intensification tandis que de grandes valeurs de ***AP*** augmentent la diversification.

Dans CSA, les positions de tous les corbeaux sont initialisées de manière aléatoire et une fois la position du corbeau est met à jour, la nouvelle position est évaluée à l'aide d'une fonction objective et le vecteur mémoire est mis à jour comme présenté dans **l'éq. (4).**

$$M_i^{it+1} = \begin{cases} X_i^{it+1} & \text{if } F(X_i^{it+1}) \text{ is better than } F(M_i^{it}) \\ M_i^{it} & \text{otherwise} \end{cases} \quad (4)$$

Où ***F(.)*** représente la fonction objectif à optimiser.

En raison de sa simplicité et de son efficacité, CSA a été utilisé pour résoudre différents problèmes tels que le problème de sélection de caractéristiques **[51],** la segmentation d'image **[54],** le diagnostic de maladies **[51],** l'optimisation électromagnétique **[50],** le problème de répartition de la charge économique **[53]** et les problèmes d'ordonnancement **[52].**

III -3- Mise en œuvre de CSA en général pour l'optimisation [57] :

Le pseudocode de CSA est illustré **à la Figure 3.4** La procédure par étapes pour la mise en œuvre de la CSA est donnée comme suit :

Étape 1 : Initialiser le problème et les paramètres ajustables.

Le problème d'optimisation, les variables de décision et les contraintes sont définis. Ensuite, les paramètres ajustables de CSA (taille du troupeau (***N***), nombre maximal d'itérations $(it_{\max})$, longueur de vol (***fl***) et la probabilité de prise de conscience (***AP***)) sont évalués.

Étape 2 : Initialiser la position et la mémoire des corbeaux.

N corbeaux sont positionnés au hasard dans un espace de recherche de **D** dimensions comme les membres du troupeau. Chaque corbeau indique une solution réalisable du problème et ***d*** est le nombre de variables de décision.

$$\text{Crows} = \begin{bmatrix} x_1^1 & x_2^1 & \dots & x_d^1 \\ x_1^2 & x_2^2 & \dots & x_d^2 \\ \vdots & \vdots & \vdots & \vdots \\ x_1^N & x_2^N & \dots & x_d^N \end{bmatrix} \quad (5)$$

La mémoire de chaque corbeau est initialisée. Puisque à l'itération initiale, les corbeaux n'ont aucune expérience, on suppose qu'ils ont caché leurs aliments à leurs positions initiales.

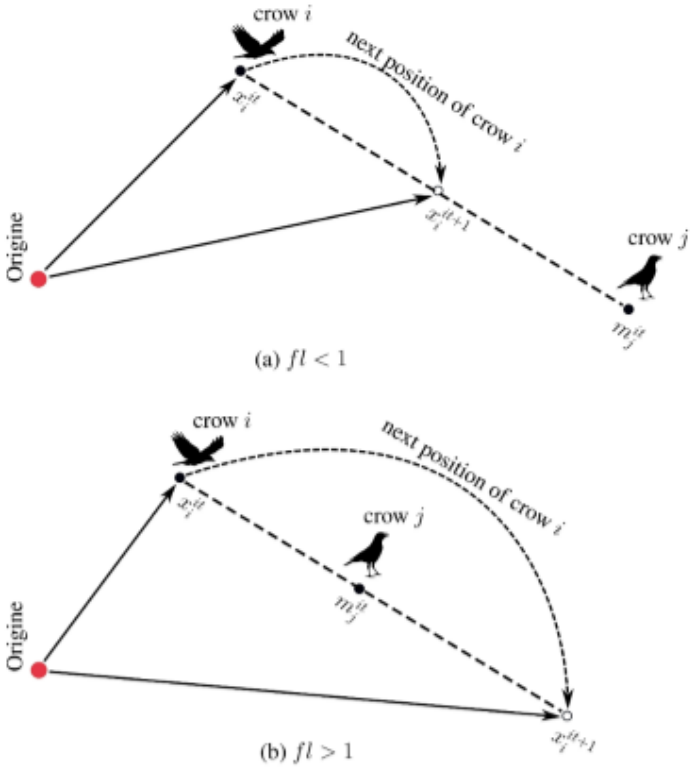


Figure 3.3: organigramme de l'état 1 dans CSA *(a)* $\boldsymbol{fl} < 1$ *et (b)* $\boldsymbol{fl} > 1$. Corbeau ***i*** peut aller à n'importe quelle position sur la ligne pointillée **[57].**

Crow search algorithm

Randomly initialize the position of a flock of N crows in the search space
Evaluate the position of the crows
Initialize the memory of each crow
while $iter < iter_{max}$
  for $i = 1 : N$ (*all N crows of the flock*)
    Randomly choose one of the crows to follow (*for example j*)
    Define an awareness probability
      if $r_j \geq AP^{j,iter}$
        $x^{i,iter+1} = x^{i,iter} + r_i \times fl^{i,iter} \times \left(m^{j,iter} - x^{j,iter}\right)$
      else
        $x^{i,iter+1} = a\ random\ position\ of\ search\ space$
      end if
  end for
Check the feasibility of new positions
Evaluate the new position of the crows
Update the memory of crows
end while

Figure 3.4: Pseudo code du CSA proposé **[57].**

$$\text{Memory} = \begin{bmatrix} m_1^1 & m_2^1 & \dots & m_d^1 \\ m_1^2 & m_2^2 & \dots & m_d^2 \\ \vdots & \vdots & \vdots & \vdots \\ m_1^N & m_2^N & \dots & m_d^N \end{bmatrix} \quad (6)$$

Étape 3 : Évaluer la fonction de remise en forme (objectif).

Pour chaque corbeau, la qualité de sa position est calculée en insérant les valeurs des variables de décision dans la fonction objective.

Étape 4 : Générer une nouvelle position.

Les corbeaux génèrent une nouvelle position dans l'espace de recherche comme suit : supposons que le corbeau ***i*** veuille générer une nouvelle position. Pour ce but, ce corbeau sélectionne au hasard l'un des corbeaux du troupeau (par exemple corbeau ***j***) et le suit pour découvrir la position des aliments cachés par ce corbeau (m^j). La nouvelle position du corbeau ***i*** est obtenue par **Eq. (3).** Cette processus est répété pour tous les corbeaux.

Étape 5 : Vérifier la faisabilité des nouveaux postes.

La faisabilité de la nouvelle position de chaque corbeau est vérifiée. Si la nouvelle position d'un corbeau est possible, le corbeau met à jour sa position. Sinon, le corbeau reste dans la position actuelle et ne déplace pas vers la nouvelle position générée.

Étape 6 : Évaluer la fonction de fitness des nouveaux postes.

La valeur de la fonction de fitness pour la nouvelle position de chaque corbeau est calculée.

Étape 7 : Mettre à jour la mémoire.

Les corbeaux mettent à jour leur mémoire comme suit :

$$M_i^{it+1} = \begin{cases} X_i^{it+1} & \text{if } F(X_i^{it+1}) \text{ is better than } F(M_i^{it}) \\ M_i^{it} & \text{otherwise} \end{cases} \quad (4)$$

Où **f(.)** désigne la valeur de la fonction objectif.

On voit que si la valeur de la fonction de fitness de la nouvelle position du corbeau est meilleure que la valeur de la fonction de fitness de la mémoire position, le corbeau met à jour sa mémoire par la nouvelle position.

Étape 8 : Vérifier le critère de terminaison.

Les étapes 4 à 7 sont répétées jusqu'à ce que $iter^{max}$ soit atteint. Lorsque le critère de terminaison est rempli, la meilleure position de la mémoire en termes de la valeur de la fonction objectif est rapportée comme la solution du problème d'optimisation.

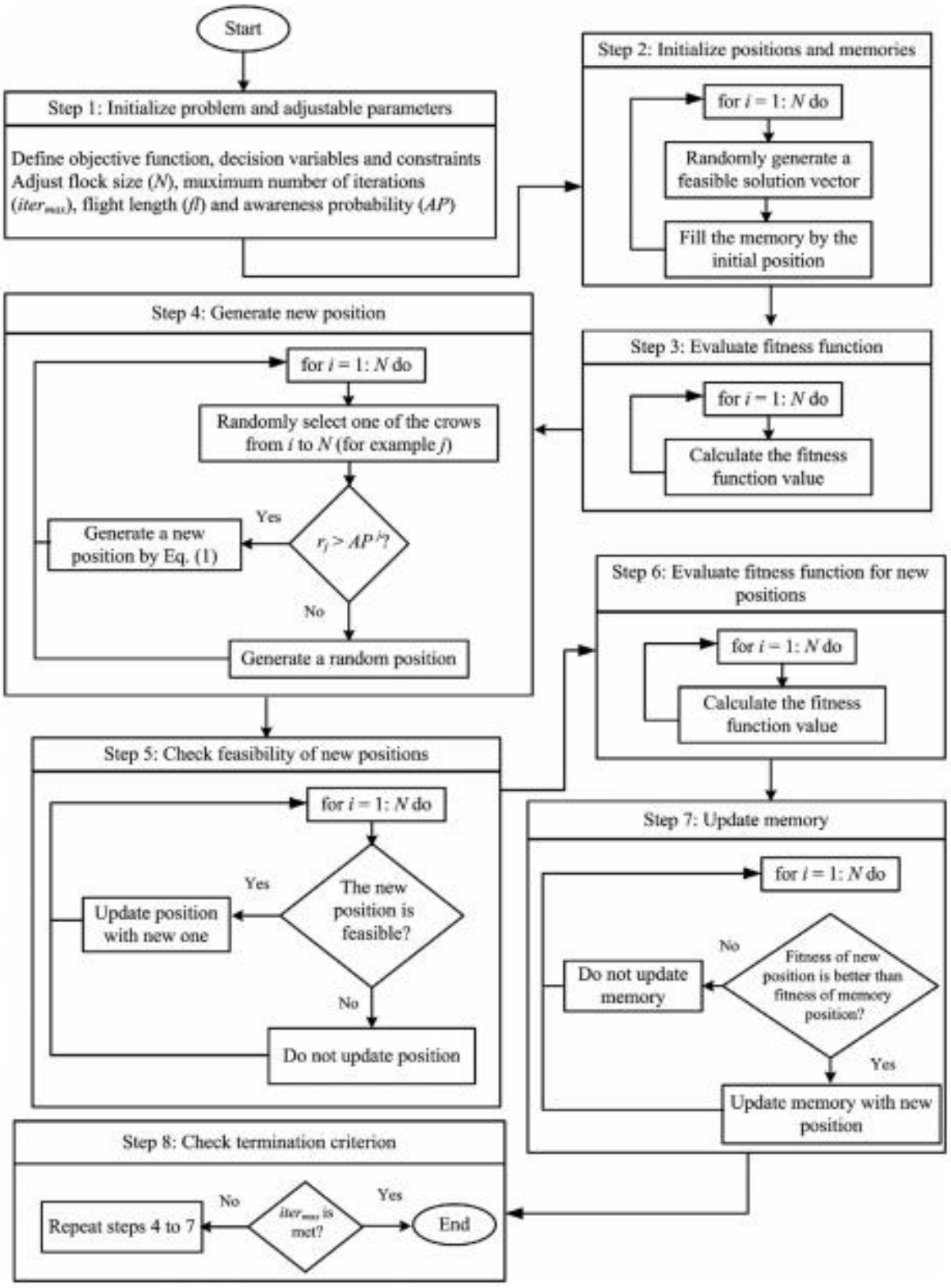


Figure 3.5: Organigramme de CSA pour faire de l'optimisation **[57].**

III -4- Conclusion [57] :

EN Basant sur le comportement d'intelligent des corbeaux, un nouvel algorithme métaheuristique, appelé CSA, est proposé. CSA est un algorithme d'optimisation basé sur la population qui est assez simple avec deux paramètres réglables (longueur de vol et probabilité de prise de conscience) uniquement, ce qui le rend très attractif pour les applications dans différents domaines d'ingénierie. En CSA, le paramètre de sensibilisation de la probabilité est directement utilisé pour contrôler la diversité de l'algorithme. L'utilité de La CSA est évaluée en résolvant les différents problèmes de conception technique qui ont des différentes natures en fonctions d'objectifs, les contraintes et les variables de décision. Les résultats de la simulation montrent que les performances du nouvel algorithme proposé sont prometteuses puisqu'il a produit des résultats compétitifs par rapport aux autres algorithmes. D'après les résultats, on voit que le taux de convergence de CSA est bon et cet algorithme trouve la solution des problèmes étudiés en environ 1 s.

De plus, d'autres techniques de binarisation peuvent être incorporées dans l'algorithme de recherche de corbeau pour travailler dans l'espace binaire.

Chapitre 4 : La réalisation :

Introduction :

Afin de tirer profit de l'étude bibliographique présenté dans les chapitres précédents, poussée sur les méthodes d'apprentissage, l'apprentissage positif et non étiqueté et la méthode méta-heuristique **Crow Search Algorithm (CSA).**

Nous allons consacrer ce dernier chapitre à la description de notre contribution à la problématique de l'apprentissage positif et non étiqueté.

Notre contribution s'est située principalement à la proposition d'une nouvelle méthode de binarisation de l'algorithme CSA pour que celle-ci s'adapte à notre problématique, à savoir l'apprentissage positif et non étiqueté. Cette méthode est utilisée à la sélection des meilleurs éléments négatifs parmi l'ensemble non étiqueté. Par la suite nous avons utilisé la méthode d'apprentissage SVM pour créer le meilleur modèle de classification.

I- BCSA-SVM un algorithme de recherche binaire crow pour l'apprentissage positif et non étiqueté :

I- 1- Introduction :

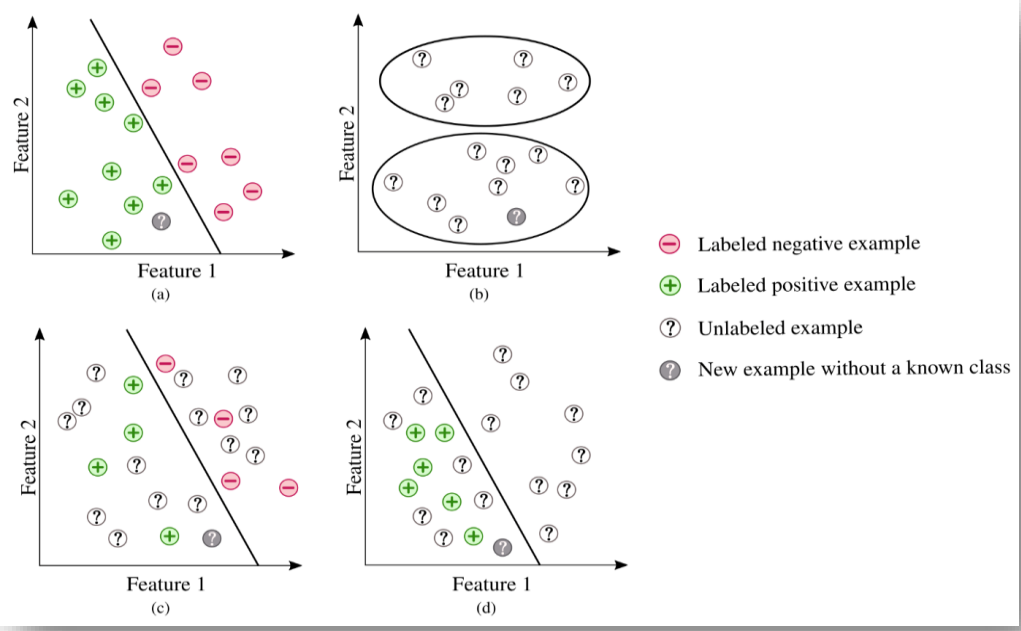


Figure 4-1:Étiquetage des exemples pour les méthodes de prédiction supervisée (a), non supervisée (b), semi-supervisée (c) et positives et non étiquetées (d)

Dans **l'apprentissage supervisé (Figure 4-1 (a)),** tous les exemples d'ensembles d'apprentissage sont étiquetés. Le processus d'apprentissage est effectué sur des exemples d'apprentissage positifs et négatifs et le nouvel élément sans étiquette de classe connue est prédit comme positif selon le modèle appris (ligne noire).

Dans l'apprentissage non supervisé (Figure 4-1 (b)), aucune étiquette n'est donnée, donc tous les exemples ne sont pas étiquetés et ils sont regroupés en fonction de leurs similitudes.

Dans l'apprentissage semi-supervisé (Figure 4-1(c)), les informations d'un petit ensemble d'exemples étiquetés, comprenant des échantillons positifs et négatifs, et un grand ensemble d'exemples non étiquetés sont utilisées pour apprendre les paramètres du modèle.

Dans l'apprentissage positif et non étiqueté (Figure 4-1(d)), il n'y a pas d'exemples négatifs étiquetés pour la formation, seulement un petit ensemble d'exemples positifs étiquetés et un grand ensemble d'exemples non étiquetés.

I- 2- Description de la méthode :

Comme mentionné dans le chapitre précédent, l'algorithme original de recherche de corbeaux (CSA) a été appliqué principalement à des problèmes d'optimisation continue. Notre problème est vu comme un problème d'optimisation binaire et combinatoire. Par conséquent, certaines modifications du CSA original sont nécessaires afin de le préparer à traiter l'apprentissage positif non étiqueté.

Dans cet algorithme, chaque corbeau de la volée représente une solution possible c.à.d. un positif fiable parmi l'ensemble sans étiquette, comme cela sera détaillé dans la section suivante.

I- 3- Encodage de la position du corbeau :

Crow Search Algorithm (**CSA**) est une technique de recherche évolutive basée sur la population, qui a été introduite pour la première fois en **[57].** Dans cet article, afin de détecter les meilleurs exemples négatifs fiables dans les problèmes d'apprentissage PU, une nouvelle version binaire de CSA est proposée appelée **BCSA-SVM** en tant qu'algorithme de couplage de recherche avec un classifieur basé sur SVM.

BCSA-SVM utilise le codage binaire pour représenter l'ensemble sans étiquette.

Avec le schéma suivant, la position Xi de chaque corbeau sera simplement un vecteur qui fournit des valeurs binaires décrivant la sélection comme des négatifs fiables parmi les exemples non étiquetés.

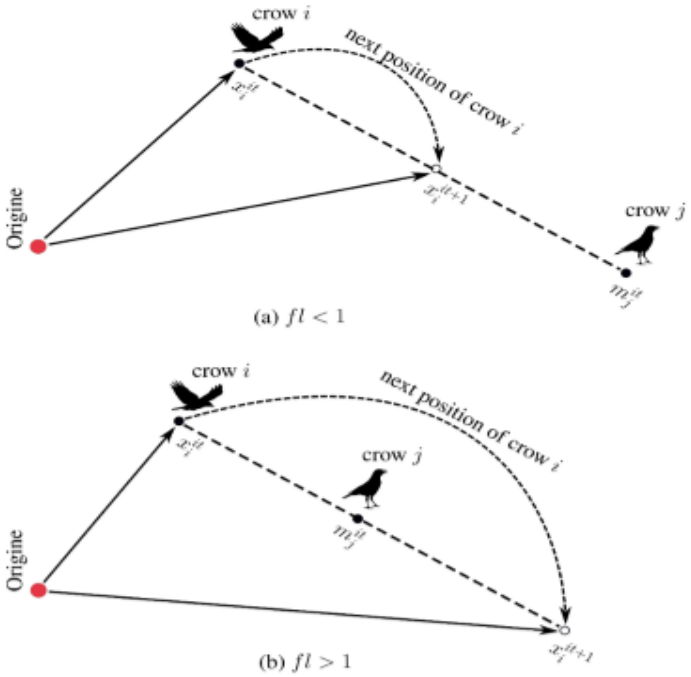


Figure 4-2: organigramme de l'état 1 dans CSA *(a)* $\boldsymbol{fl} < 1$ *et (b)* $\boldsymbol{fl} > 1$. Corbeau ***i*** peut aller à n'importe quelle position sur la ligne pointillée **[57].**

Nous représentons la volée comme un ensemble de ***P*** corbeaux. Chaque corbeau est déterminé avec ses positions actuelles et mémoire. Chaque position de corbeau $\boldsymbol{X_i}$ et mémoire de corbeau $\boldsymbol{M_i}$, $i = 1,\cdots,P$, caractérisée par ***N*** éléments binaires, $x_i^1,\cdots,x_i^N$ et $M_i^1,\cdots,M_i^N$, où ***N*** est la taille de l'ensemble de données non étiqueté. $x_i^j \in \{0,1\}$, définir si oui ou non le $\boldsymbol{j^{ème}}$ exemple sans étiquette dans le $\boldsymbol{i^{ème}}$ corbeau a été sélectionné comme exemple négatif, et $M_i^j \in \{0,1\}$, représente la cachette de la $\boldsymbol{j^{ème}}$ règle dans le $\boldsymbol{i^{ème}}$ corbeau. Une représentation de la position du corbeau et de la mémoire est montrée dans **la figure 4-2**.

I- 4- Initialisation de la volée :

Pour initialiser la population, une initialisation uniforme aléatoire est utilisée, dans laquelle les corbeaux de la volée initiale sont dispersés de manière aléatoire dans l'espace de recherche binaire. Après avoir généré les corbeaux initiaux de la volée, la position de mémoire initiale de chaque corbeau a été attribuée à sa position actuelle.

I- 5- Évaluation de la fonction de fitness :

La valeur de la fonction de fitness pour la nouvelle position de chaque corbeau est calculée.

Pour chaque corbeau, la qualité de sa position est calculée en insérant les valeurs de la variable de décision dans la fonction objective. Nous avons choisi la mesure AUC ROC qui permet d'évaluer les performances d'un système en combinant deux indicateurs sensibility (le taux des vrais positifs) et spécifity (le taux des faux positifs).

I- 6- A la recherche d'un nouveau poste :

Dans CS Algorithme, c'est le vecteur de mise à jour de la position qui pilote le processus d'optimisation. A l'itération, chaque corbeau a une mémoire où la position de la cachette est mémorisée M_j^{it} . A cette itération, pour mettre à jour sa position, le corbeau ***i*** choisit un corbeau aléatoire ***j*** dans la volée et décide de suivre sa cachette, deux états se produisent. Le premier est que le corbeau choisi ne sait pas qu'il est suivi, et le second est lorsque le corbeau sait qu'il est suivi.

• **État 1** : Le corbeau ***i*** s'approchera de la cachette du corbeau ***j***. Ensuite, la nouvelle position du corbeau ***i*** est obtenue comme suit :

• **Etat 2** : Crow ***j*** changera de position dans l'espace de recherche pour protéger sa cache.

Dans l'algorithme BCSA-SVM, l'équation de mise à jour de la position de chaque corbeau est modifiée par **l'éq. (1)** suivante :

$$X_i^{it+1} = \begin{cases} X_i^{it} \oplus R_i \otimes fl_i^{it} \otimes (M_j^{it} \; ! \; X_i^{it}) & r_j \geq AP_i^{it} \\ \text{Choose a random position} & \text{otherwise} \end{cases} \quad \textbf{(1)}$$

Où, X_j^{it+1} indique la position mise à jour du corbeau ***i*** à l'itération ***it+1***, M_j^t désigne la position mémorisée du corbeau ***j*** à l'itération ***it***, ***ri*** et ***rj*** sont des vecteurs générés aléatoirement comprenant 0 ou 1 éléments, $\boldsymbol{fl_j^{it}}$ désigne la longueur de vol du corbeau ***i*** à l'itération ***it*** et $\boldsymbol{AP_j^{it}}$ est la probabilité de prise de conscience du corbeau ***i*** à l'itération ***it***.

Dans **éq. (1)** nous avons introduit les trois nouveaux opérateurs Différence, Multiplication et Fusion, ils sont définis comme suit :

<u>Opérateur de différence</u> ! :

Dans la version continue de CSA, cet opérateur calcule la différence entre la position actuelle du i^{ème} corbeau et la position mémorisée. Dans cette nouvelle version binaire de CSA nous proposons le mécanisme qui suit pour implémenter cet opérateur.

Nous avons utilisé la différence entre deux vecteurs binaires pour générer une solution intermédiaire. Cet opérateur est défini dans **Eq.2** et **Eq.3.** Dans lesquelles, nous utilisons DX pour montrer la différence entre la position du corbeau *i* (X_i^{it}) et la position mémorisée du corbeau *z* choisi au hasard (M_z^{it}).

$$DX_i^{it} = M_z^{it} \ominus X_i^{it} \; \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (2)$$

$$\mathrm{DX_{ij}^t} = \begin{cases} 0 \text{ or } 1 \text{ randomly} & \text{if } \mathrm{M_{iz}^{it}} = 0 \text{ and } \mathrm{X_{ij}^{it}} = 1 \\ \mathrm{M_{iz}^{it}} - \mathrm{X_{ij}^{it}} & \text{otherwise} \end{cases} \quad \ldots\ldots\ldots\ldots\ldots (3)$$

Ci-dessous est l'implémentation de cet opérateur en python :

```
def Bin_Sub_op(M, X):
    # binary substraction
    DX = np.zeros(M.size, dtype=int)
    for i in range(M.size):
        if M[i] == 0 and X[i] == 1:
            DX[i] = np.random.randint(2, size=1)[0]
        else:
            DX[i] = M[i] - X[i]
    return DX
```

Opérateur de fusion ⊕:

Cet opérateur fusionne deux vecteurs X_j^{it} and Q_{ij}^{it}. L'opérateur fusion est défini par **Eq.4.**

$$X_{ij}^{t+1} = \begin{cases} 0 \text{ or } 1 \text{ randomly} & if\ X_{ij}^t = Q_{ij}^t = 1 \\ X_{ij}^t + Q_{ij}^t & otherwise \end{cases} \quad \dots\dots\dots\dots (4)$$

La fonction en python est la suivante :

```
def Bin_Add_op(X, P):
    # binary addition
    M = np.zeros(P.size, dtype=int)
    for i in range(P.size):
        if X[i] == P[i] and P[i] == 1:
            M[i] = np.random.randint(2, size=1)[0]
        else:
            M[i] = X[i] + P[i]
    return M
```

Opérateur de multiplication ⊗:

Pour binarisé cet opérateur nous avons introduit un nouveau paramètre *fl_max* qui désigne le vol maximal d'un corbeau. L'opérateur de multiplication permet de calculer le produit d'un vecteur binaire par un scalaire *fl*. Pour garder la même philosophie de la méthode originale, donc selon la valeur et le signe de *fl* qui représente la distance du vol d'un corbeau, un nombre aléatoire de 0 ou de 1 est généré qui remplacent respectivement les 1 ou les 0 du vecteur X_{ij}^t.

$$X_{ij}^{t+1} = \begin{cases} X_{ij}^t & if\ fl = 1 \\ vecteur\ null & if\ fl = 0 \\ remplacer\ aleatoirement\ \ nb\ 1\ de\ \ de\ X_{ij}^t\ par\ des\ 0 & if\ fl < 0 \\ remplacer\ aleatoirement\ \ nb\ 0\ de\ \ de\ X_{ij}^t\ par\ des\ 1 & if\ fl > 0 \end{cases} \dots\dots\dots\dots\dots (5)$$

La fonction correspondante en python est la suivante :

```
def mult_scalar(fl, V, fl_Max):
    P = V
    nb1 = V[V == 1].size.   #calculer
    nb0 = V[V == 0].size
    if nb0 != 0:
        stp1 = fl_Max / nb0
    if nb1 != 0:
        stp0 = 1 / nb1
    if fl == 1:
        P = V
    elif fl == 0:
        P = 0 * V
    elif fl > 1:
        if nb0 != 0:
            add1 = round(fl / stp1)  # nbr de 1 à ajouter aléatoirement
            indx0 = np.argwhere(V == 0)
            randindx0 = np.random.choice(
                np.random.permutation(indx0.size), add1)
            V[randindx0] = 1
            P = V
    else:
        if nb1 != 0:
            add0 = round(fl / stp0)  # nbr de 0 à ajouter aléatoirement
            indx1 = np.argwhere(V == 1)
            randindx1 = np.random.choice(
                np.random.permutation(indx1.size), add0)
            V[randindx1] = 0
            P = V
    return P
```

Finalement, le corbeau met à jour sa mémoire si la valeur de la fonction de fitness de la nouvelle position est mieux que sa valeur de la position mémorisée. Le corbeau met à jour sa mémoire par la nouvelle position comme dans **Eq 6**

$$M_i^{it+1} = \begin{cases} X_i^{it+1}, & if\ Fn\left(X_i^{it+1}\right) is\ better\ than\ Fn(M_i^{it}) \\ M_i^{it} & otherwise \end{cases} \dots\dots\dots\dots (6)$$

Où ***Fn()*** est défini comme la fonction objectif.

II- Les données quantitatives (categorical data) [72] :

Les données sont importantes pour tout. La plupart des organisations commerciales collectent des données sur les ventes, les revenus, les cours des actions et les bénéfices. Le gouvernement recueille des données sur les taux d'alphabétisation, les taux de criminalité, le taux de natalité, etc. Les données sont importantes pour la recherche scientifique et universitaire, comme l'apprentissage automatique et les réseaux de neurones. La collecte et l'analyse de données aident à prendre de meilleures décisions. L'une des technologies émergentes dans le monde moderne est l'Internet of Things (IoT). Cette technologie collecte des données à l'aide de capteurs et d'actionneurs et les traites pour une prise de décision intelligente.

I I-1 Qu'est-ce qu'une donnée quantitative?

Les données quantitatives sont des données qui peuvent être mesurées (taille, poids…) ou repérées (température…)

Exemples de propriétés physiques quantitatives : Le point de fusion, (par exemple, le fer fond à une température de 1 535 °C), le point d'ébullition, la masse volumique, la viscosité, la solubilité, la conductivité électrique, la conductivité thermique…

Les données quantitatives sont ensuite divisées en **données discrètes et continues.** **Les données discrètes** ont certaines valeurs telles que des nombres entiers. Quelques exemples sont le nombre d'étudiants, le nombre de machines, etc. ils sont dénombrables.

D'autre part, **les données continues** peuvent avoir n'importe quelle valeur dans une plage. Quelques exemples sont la taille et le poids; ils sont mesurables).

Certains caractères ne peuvent prendre que des valeurs entières, par exemple le nombre des enfants d'une famille, le nombre de pièces d'un logement. Un tel caractère est qualifié de **«discret».**

La valeur d'autres caractères peut varier d'aussi peu que l'on voudra dans un intervalle fini ou infini, par exemple la taille d'une personne, le poids d'un enfant. Un tel caractère est dit **«continu».**

De plus, l'application d'opérations arithmétiques telles que l'addition et la division à des données quantitatives leur donne plus de sens. Par exemple, en ajoutant toutes les hauteurs et en divisant la réponse par le nombre total d'observations, on obtient la moyenne. C'est une mesure importante pour analyser.

I I-2 Différence entre les données qualitatives et quantitatives :

La principale différence entre les données qualitatives et quantitatives est que les données qualitatives sont descriptives tandis que les données quantitatives sont numériques.

Par exemple, les données qualitatives sont le genre, le pays, la ville, la nationalité, etc., tandis que les données quantitatives sont la longueur, la largeur, la taille, le poids, la superficie, la vitesse, l'âge, etc. nombre de véhicules, nombre d'élèves, etc.) ou données continues (données mesurables telles que la

taille, le poids, etc.).

I I-3 Bases de données utilisées et résultats [73]:

Base de données (Dataset)	Les attributs (Attributes)	% de Positive % of Pos.	Nbr. de Positive Nbr. of Pos.	Nbr. de Sans étiquette N. of Unlabeled	Tester Inst. Test Inst.
audiology	69	30% 40% 50%	15 20 26	79 74 68	11 11 11
breast-cancer	9	30% 40% 50%	54 72 91	203 185 166	29 29 29
chess	36	30% 40% 50%	451 601 751	2425 2275 2125	320 320 320
dermatology	34	30% 40% 50%	30 40 50	136 126 116	19 19 19
hepatitis	19	30% 40% 50%	9 12 15	130 127 124	16 16 16
lymph	18	30% 40% 50%	17 22 28	111 106 100	14 14 14
nursery	8	30% 40% 50%	1166 1555 1944	6562 6173 5784	859 859 859
pima	8	30% 40% 50%	135 180 225	556 511 466	77 77 77
soybean	35	30% 40% 50%	25 33 42	140 132 123	18 18 18
vote	16	30% 40% 50%	72 96 120	319 295 271	44 44 44

La Table 4-1 : Caractéristiques des Base de données **[73].**

Dans cette section, nous fournissons un ensemble exhaustif d'expériences pour montrer l'efficacité de notre approche PU Learning en données catégorielles. Les expériences sont réalisées sur 30 échantillons dérivés de 10 ensembles de données, accessibles au public sur le référentiel d'apprentissage automatique de **l'UCI (the UCI machine Learning repository http://archive.ics.uci.edu/ml/).**

Pour chaque ensemble de données, nous produisons trois échantillons différents qui diffèrent les uns des autres par le nombre d'exemples étiquetés comme positifs, respectivement 30 %, 40 % et 50 % de la classe positive. Les exemples positifs restants plus tous les exemples négatifs sont considérés comme non étiquetés. Nous supposons la classe majoritaire comme positive, l'autre comme négative. Si l'ensemble de données ne décrit pas un problème de classification binaire nous sélectionnons les deux plus grandes classes (en nombre d'instances) pour réduire le problème à une tâche de classification binaire. Pour terminer, en tant que prétraitement supplémentaire, tous les attributs numériques sont discrétisés en 10 bacs avec une valeur égale largeur.

Les détails sur les 30 échantillons sont présentés dans **le tableau 4-1.**

Pour évaluer les résultats des différents PU classifieurs, nous utilisons la F-Measure comme indicateur de performance.

Étant donné la matrice de confusion suivante :

		Predicted Class		
		Positive	**Negative**	
Actual Class	**Positive**	True Positive (TP)	False Negative (FN) **Type II Error**	**Sensitivity** $\frac{TP}{(TP+FN)}$
	Negative	False Positive (FP) **Type I Error**	True Negative (TN)	**Specificity** $\frac{TN}{(TN+FP)}$
		Precision $\frac{TP}{(TP+FP)}$	**Negative Predictive Value** $\frac{TN}{(TN+FN)}$	**Accuracy** $\frac{TP+TN}{(TP+TN+FP+FN)}$

La F-Measure est définie comme :

$$F\text{-}Measure = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

Tandis que la précision est égale à :

$$precision = \frac{TP}{TP + FP}$$

Et recall (le rappel) est égal à :

$$recall = \frac{TP}{TP + FN}$$

Où TP est le nombre d'exemples positifs classés comme positifs, FP est le nombre d'exemples négatifs classés comme positifs et FN est le nombre d'exemples positifs classés comme négatif. La F-measure permet une meilleure évaluation dans des ensembles de données déséquilibrés ; pour ça raison pour laquelle nous préférons utiliser ces mesures plutôt que la précision. Les valeurs de F-measure sont calculées et moyenné sur un schéma de validation croisée de 10 fois. Notez que, contrairement à d'autres partiellement paramètres supervisés, tels que la détection d'anomalies, le but de l'apprentissage PU est de construire un classifieur binaire qui devrait bien discriminer les exemples positifs et négatifs. **[74]**, nous menons nos expériences comme dans un classifieur binaire classique, en utilisant F-Measure comme indicateur de performance, plutôt que l'AUC ou d'autres métriques couramment adoptées dans les tâches de détection d'anomalies **[73]**.

Pour évaluer notre méthode nous avons utilisé les mêmes données et résultats mentionné dans le papier [**Basile et al., "Density Estimators for Positive-Unlabeled Learning." [75]**].

Chaque base de données est divisée en 10 parties (folds) sur lesquelles un processus d'apprentissage et de test sont appliqués 10 fois. Chaque fois deux parties différentes sont utilisées pour l'apprentissage et le test. Le résultat final sera la moyenne des 10 parties.

Les résultats obtenus dans le tableau suivant montrent les meilleures performances en rouge, notre méthode surpasse les autres méthodes 14 fois sur 30 avec **la meilleure moyenne F-mesure de 0.802**

Dataset	**%**	**BCSA-SVM**	**GPU^{BN}**	**GPU^{MT}**	**Pulce**	**PNB**	**APNB**	**PTAN**	**APTAN**
Audiology	30	**1**	0.839	0.902	0.745	0.68	0.7	0.66	0.66
Audiology	40	**1**	0.879	0.804	0.846	0.75	0.74	0.71	0.66
Audiology	50	**1**	0.98	0.991	0.899	0.8	0.8	0.78	0.71
Breast-cancer	30	0.437	0.475	0.45	**0.534**	0.4	0.39	0.43	0.43
Breast-cancer	40	0.441	0.483	**0.513**	0.438	0.42	0.4	0.43	0.45
Breast-cancer	50	0.433	0.517	**0.535**	0.443	0.42	0.41	0.44	0.44
Chess	30	**0.742**	0.689	0.663	0.696	0.58	0.64	0.59	0.64
Chess	40	**0.769**	0.691	0.665	0.688	0.58	0.64	0.6	0.64
Chess	50	**0.787**	0.773	0.65	0.655	0.58	0.64	0.6	0.64
Dermatology	30	0.986	**1**	0.834	0.992	0.57	0.57	0.57	0.56
Dermatology	40	0.992	**1**	0.836	0.992	0.57	0.58	0.57	0.57
Dermatology	50	**0.992**	**0.992**	0.951	**0.992**	0.59	0.6	0.57	0.58
Hepatitis	30	0.808	0.822	0.665	0.843	**0.87**	**0.87**	0.85	0.86
Hepatitis	40	0.808	0.778	0.654	0.873	**0.88**	**0.88**	0.85	0.85

Hepatitis	50	0.837	0.764	0.742	0.855	**0.88**	**0.88**	0.86	0.85
Lymph	30	0.839	0.827	0.782	**0.851**	0.84	0.85	0.79	0.84
Lymph	40	**0.844**	0.825	0.795	0.827	0.84	0.83	0.79	0.81
Lymph	50	0.826	0.824	0.835	0.814	0.86	**0.87**	0.81	0.82
Nursery	30	0.767	**0.809**	0.761	0.739	0.65	0.65	0.56	0.5
Nursery	40	0.78	**1**	0.762	0.773	0.69	0.69	0.61	0.56
Nursery	50	0.771	**0.96**	0.779	0.807	0.69	0.7	0.74	0.44
Pima	30	**0.6**	0.588	0.576	0.532	0.49	0.5	0.5	0.5
Pima	40	0.573	0.568	**0.593**	0.547	0.49	0.5	0.5	0.51
Pima	50	0.592	**0.609**	0.605	0.528	0.49	0.51	0.5	0.52
Soybean	30	**0.945**	0.893	0.766	0.738	0.81	0.86	0.8	0.81
Soybean	40	**0.933**	0.883	0.852	0.767	0.86	0.86	0.84	0.83
Soybean	50	**0.947**	0.89	0.923	0.823	0.92	0.92	0.88	0.86
Vote	30	0.771	**0.826**	0.799	0.679	0.62	0.62	0.56	0.55
Vote	40	**0.909**	0.85	0.79	0.8	0.71	0.71	0.58	0.54
Vote	50	**0.942**	0.844	0.829	0.829	0.77	0.77	0.61	0.56
#wins		**14**	8	3	3	3	4	0	0
Avg. F1-score		**0.802**	0.796	0.743	0.751	0.677	0.686	0.653	0.643
30%		**0.789**	0.777	0.72	0.735	0.651	0.665	0.631	0.635
40%		**0.805**	0.796	0.727	0.755	0.679	0.683	0.648	0.642
50%		0.813	**0.815**	0.784	0.764	0.7	0.71	0.679	0.652
Avg. ranking		**2.31**	2.66	4.03	3.71	5.14	4.49	6.34	6.23

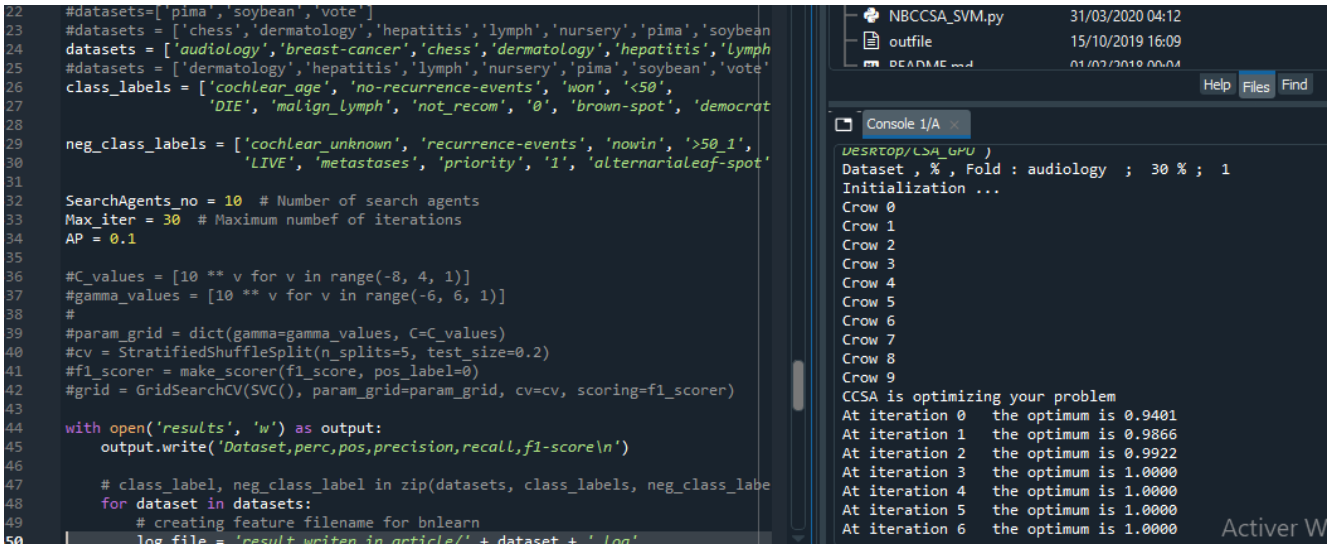
La Table 4-2 : Résultats obtenus et comparaison par rapport à d'autres **[75]** approches.

III- Présentation des résultats :

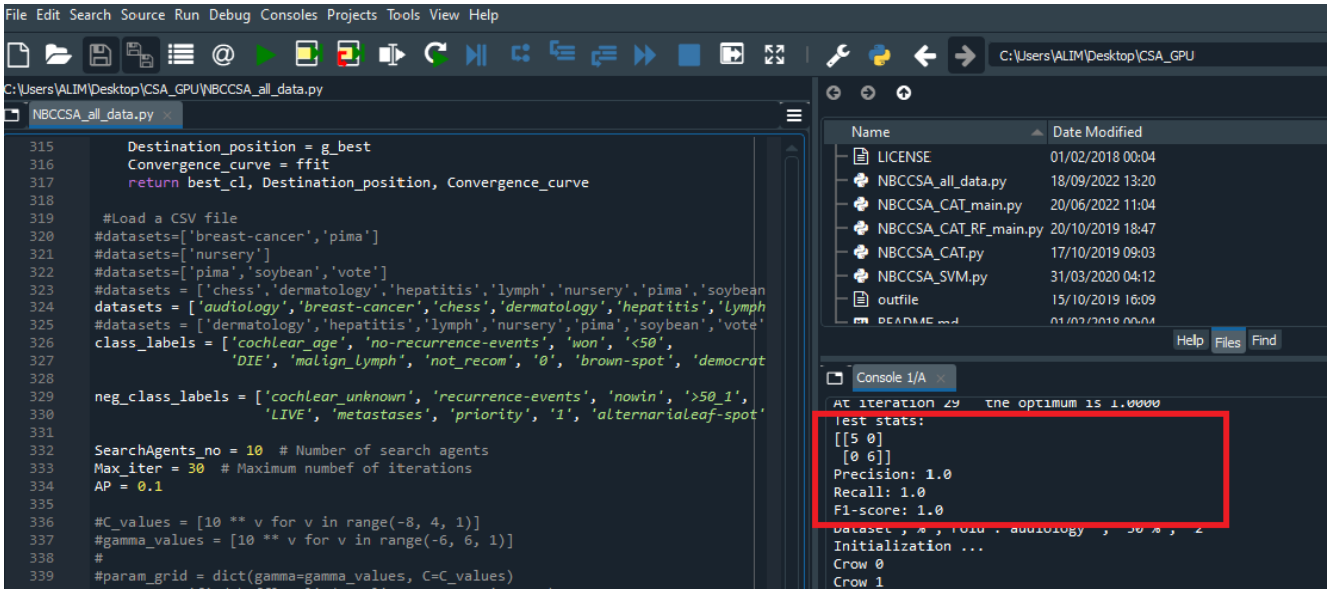
Ci-dessous les résultats après l'importation du package arff **(**pour la lecture des 10 bases de données **['audiology', 'breast- cancer', 'chess' ,'dermatology' ,'hepatitis' ,'lymph', 'nursery', 'pima', 'soybean', 'vote'])**

Les expériences sont réalisées sur 30 échantillons dérivés de 10 ensembles de données, qui nous permet de connaitre les mesures de performances tel que **la meilleure moyenne F-mesure de 0.802**

1- Initialisation :



2- Les mesures precision, recall,f1-mesure de la base de données audiology :



3- Les mesures precision, recall,f1-mesure de la base de données audiology sur l'état de sortie sont :

audiology - Bloc-notes

Fichier Edition Format Affichage Aide

```
perc,fold,precision,recall,f1-scor
30,1,1.0,1.0,1.0
30,2,1.0,1.0,1.0
30,3,1.0,1.0,1.0
30,4,1.0,1.0,1.0
30,5,1.0,1.0,1.0
30,6,1.0,1.0,1.0
30,7,1.0,1.0,1.0
30,8,1.0,1.0,1.0
30,9,1.0,1.0,1.0
30,10,1.0,1.0,1.0
40,1,1.0,1.0,1.0
40,2,1.0,1.0,1.0
40,3,1.0,1.0,1.0
40,4,1.0,1.0,1.0
40,5,1.0,1.0,1.0
40,6,1.0,1.0,1.0
40,7,1.0,1.0,1.0
40,8,1.0,1.0,1.0
40,9,1.0,1.0,1.0
40,10,1.0,1.0,1.0
50,1,1.0,1.0,1.0
50,2,1.0,1.0,1.0
50,3,1.0,1.0,1.0
50,4,1.0,1.0,1.0
50,5,1.0,1.0,1.0
50,6,1.0,1.0,1.0
50,7,1.0,1.0,1.0
50,8,1.0,1.0,1.0
50,9,1.0,1.0,1.0
50,10,1.0,1.0,1.0
```

4- Les mesures f1-mesure de toutes les bases de données sur l'état de sortie sont :

	Audiology	f-mesure	
fold	30%	40%	50%
1	1	1	1
2	1	1	1
3	1	1	1
4	1	1	1
5	1	1	1
6	1	1	1
7	1	1	1
8	1	1	1
9	1	1	1
10	1	1	1
la moyenne	1	1	1

	breast-cancer	f-mesure	
fold	30%	40%	50%
1	0,555555556	0,57143	0,57143
2	0,454545455	0,52632	0,4
3	0,476190476	0,36364	0,38095
4	0,56	0,6	0,4
5	0,307692308	0,25	0,47619
6	0,466666667	0,38462	0,52632
7	0,363636364	0,38095	0,18182
8	0,4	0,44444	0,53333
9	0,4	0,52174	0,33333
10	0,384615385	0,36364	0,52174
la moyenne	0,436890221	0,44068	0,43251

	chess	f-mesure	
fold	30%	40%	50%
1	0,774193548	0,6791	0,71918
2	0,826568266	0,65532	0,75093
3	0,871060172	0,92025	0,93464
4	0,700460829	0,95	0,90123
5	0,655246253	0,765	0,85714
6	0,8	0,7581	0,73253
7	0,941538462	0,89736	0,85955
8	0,56612529	0,77714	0,78571
9	0,638297872	0,64119	0,64819
10	0,645435244	0,64544	0,68217
la moyenne	0,741892594	0,76889	0,78713

	dermatology	f-mesure	
fold	30%	40%	50%
1	0,923076923	0,92308	0,92308
2	1	1	1
3	1	1	1
4	0,933333333	1	1
5	1	1	1
6	1	1	1
7	1	1	1
8	1	1	1
9	1	1	1
10	1	1	1
la moyenne	0,985641026	0,99231	0,99231

Audiology	f-mesure		
fold	30%	40%	50%
1	1	1	1
2	1	1	1
3	1	1	1
4	1	1	1
5	1	1	1
6	1	1	1
7	1	1	1
8	1	1	1
9	1	1	1
10	1	1	1
la moyenne	1	1	1

breast-cancer	f-mesure		
fold	30%	40%	50%
1	0,555555556	0,57143	0,57143
2	0,454545455	0,52632	0,4
3	0,476190476	0,36364	0,38095
4	0,56	0,6	0,4
5	0,307692308	0,25	0,47619
6	0,466666667	0,38462	0,52632
7	0,363636364	0,38095	0,18182
8	0,4	0,44444	0,53333
9	0,4	0,52174	0,33333
10	0,384615385	0,36364	0,52174
la moyenne	0,436890221	0,44068	0,43251

chess	f-mesure		
fold	30%	40%	50%
1	0,774193548	0,6791	0,71918
2	0,826568266	0,65532	0,75093
3	0,871060172	0,92025	0,93464
4	0,700460829	0,95	0,90123
5	0,655246253	0,765	0,85714
6	0,8	0,7581	0,73253
7	0,941538462	0,89736	0,85955
8	0,56612529	0,77714	0,78571
9	0,638297872	0,64119	0,64819
10	0,645435244	0,64544	0,68217
la moyenne	0,741892594	0,76889	0,78713

dermatology	f-mesure		
fold	30%	40%	50%
1	0,923076923	0,92308	0,92308
2	1	1	1
3	1	1	1
4	0,933333333	1	1
5	1	1	1
6	1	1	1
7	1	1	1
8	1	1	1
9	1	1	1
10	1	1	1
la moyenne	0,985641026	0,99231	0,99231

soybean	f-mesure		
fold	30%	40%	50%
1	0,888888889	0,94737	0,94118
2	0,947368421	1	1
3	1	1	0,94737
4	1	0,94118	1
5	1	0,84211	0,94737
6	0,9	0,9	0,9
7	0,818181818	0,75	0,78261
8	1	1	1
9	1	1	1
10	0,9	0,94737	0,94737
la moyenne	0,945443913	0,9328	0,94659

vote	f-mesure		
fold	30%	40%	50%
1	0,971428571	0,97143	0,97143
2	0,914285714	0,91429	0,94444
3	0,711111111	0,9697	0,9697
4	0,68	0,73913	0,94444
5	0,666666667	1	0,94118
6	0,727272727	0,875	0,90323
7	0,780487805	0,94118	0,9697
8	0,755555556	1	1
9	0,755555556	0,78947	0,83333
10	0,744186047	0,88889	0,94118
la moyenne	0,770654975	0,90891	0,94186

IV-Conclusion :

Dans ce chapitre nous avons introduit une nouvelle méthode appelée BCSA-SVM qui est un couplage d'un algorithme d'intelligence en essaim CSA (Crow Search Algorithm) basé sur l'imitation du comportement des corbeaux et la méthode SVM. La nature binaire de notre problématique nécessite la transformation de l'algorithme.

Nous avons illustré notre procédé par une application sur des bases de données réelles (30 bases de données).

Nous avons décrit chaque étape de notre processus et présenté les résultats de nos expériences, et pour valider cette méthode nous avons fait une comparaison par rapport à d'autres méthodes d'apprentissage. Les résultats obtenus sont très satisfaisants et montrent l'efficacité de notre approche.

Conclusion générale et Perspectives:

Le travail présenté dans ce mémoire se situe dans le contexte de l'apprentissage automatique (machine learning), et plus particulièrement dans le cadre de l'apprentissage positif et non étiqueté (PU learning). Généralement, les classifieurs binaires prennent en entrée deux ensembles de données, l'un contenant des données positivement étiquetées et l'autre des données négativement étiquetées. Le but étant d'apprendre un séparateur de ces deux ensembles de données. Malheureusement, il arrive souvent que ces ensembles contiennent uniquement des données positivement étiquetées ainsi que des données non étiquetées.

Notre objectif était de tirer profit des techniques d'intelligence en essaim (Swarm Intelligence) qui permettent d'améliorer les performances des méthodes d'apprentissage positif et non étiqueté (Positive and unlabeled learning).

Pour mener à bien ce mémoire nous avons commencé par une étude bibliographique pour cerner notre problématique. Nous avons commencé par un premier chapitre sur les notions de machine learning et ses méthodes. Dans le deuxième chapitre nous avons présenté un état de l'art de l'apprentissage positif et non étiqueté en résumant quelques papiers. Le troisième chapitre est consacré pour les méthodes utilisées dans notre contribution à savoir l'algorithme Crow Search Algorithm dans sa version continue et la méthode SVM.

Le dernier chapitre a été consacré à notre contribution. En premier temps, nous avons proposé une nouvelle méthode pour la sélection des négatifs fiable à partir des données non étiquetées, cette méthode est basée sur l'algorithme Crow Search Algorithm et pour s'adapter à la nature binaire de notre problématique, nous avons proposé une version binaire de cet algorithme. Par la suite et pour induire notre modèle de classification, nous avons utilisé la méthode SVM. Les expérimentations et les résultats obtenus montrent clairement que notre méthode surpasse la plupart du temps les autres méthodes utilisées pour la comparaison.

Perspectives

- Envisager la réutilisation de cette méthode qui pourrait être appliquée sur d'autres types de données pour la classification dite apprentissage positif et non étiqueté (PU learning) tel que la classification de textes, web, ...etc.
- Envisager l'optimisation de l'étape de sélection par la réduction de la dimension, on pourrait par exemple utiliser le clustering et choisir un représentant pour chaque cluster au lieu de toute la population.
- Envisager à accélérer le déplacement des corbeaux en pensant à un processus de parallélisme.
- Penser à faire une étude comparative pour choisir la meilleure fonction objective.

Référence :

[01] : Marref Nadia (18-12-2013), « Apprentissage Incrémental & Machines à Vecteurs Supports », Thèse de Magister Université HADJ LAKHDAR – BATNA.

[02] : https://www.talend.com/fr/resources/ai-vs-machine-learning-vs-deep-learning/.

[03] : https://fr.wikipedia.org/wiki/Apprentissage_automatique.

[04] : BENSIAH Oussama Akram (2020), « La proposition d'une nouvelle approche basée Deep Learning pour la prédiction du cancer du sein », Thèse de Master en Informatique Université L'Arbi Ben M'hidi Oum El Bouaghi. .

[05]: AZIZI Nabil (2009/2010), «Apprentissage automatique et fusion d'informations : Application à l'extraction des connaissances des documents web » , mémoire de Magister en Informatique Université Abbes Laghrour Khenchela

[06] Rocchio, J. (1971). Relevance feedback in information retrieval. In Salton, G., éditeur : The SMART Retrieval System - Experiments in Automatic Document Processing, page 313–323, Englewood, Cliffs, New Jersey. Prentice Hall.

[07] Hull, D. (1994). Improving text retrieval for the routing problem using latent semantic indexing. In SIGIR '94 : Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval, pages 282–291, New York, NY, USA. Springer-Verlag New York, Inc.

[08] Cohen, W. W. et Singer, Y. (1996). Context-sensitive learning methods for text categorization. In SIGIR '96 : Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval, pages 307–315,New York, NY, USA. ACM. http://doi.acm.org/10.1145/243199.243278.

[09] Singhal, A., Mitra, M. et Buckley, C. (1997). Learning routing queries in a query zone. SIGIR Forum, 31(SI):25–32. http://doi.acm.org/10.1145/278459.258530.

[10] Lewis, D. D., Schapire, R. E., Callan, J. P. et Papka, R. (1996). Training algorithms for linear text classifiers. In SIGIR '96 : Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval, pages 298–306, New York, NY, USA. ACM. http://doi.acm.org/10.1145/243199.243277

[11] Jalam, R. (2003). Apprentissage automatique et catégorisation de textes multilingues.
Thèse de doctorat, Université LUMIERE LYON2.

[12] Fuhr, N., Hartmann, S., Lustig, G., Schwantner, M., Tzeras, K., Darmstadt, T. H., Informatique, F. et Knorz, G. (1991). Air/x - a rule-based multistage indexing system for large subject fields. In Proceedings of RIAO'91, pages 606–623.

[13] Cohen, W. W. et Hirsh, H. (1998). Joins that generalize: Text classification using whirl. In In Proc. of the Fourth Int'l Conference on Knowledge Discovery and Data Mining, pages 169–173.

[14] Li, Y. et Jain, A. K. (1998). Classification of text documents. Pattern Recognition, International Conférence on, 2:1295. http://doi.ieeecomputersociety.org/10.1109/ ICPR.1998.711938

[15] Creecy, R. H., Masand, B. M., Smith, S. J. et Waltz, D. L. (1992). Trading mips and memory for knowledge engineering. Commun. ACM, 35(8):48–64. http://doi.acm.org/10.1145/135226.135228.

[16] Aha, D. W., Kibler, D. et Albert, M. K. (1991). Instance-based learning algorithms. Mach. Learn., 6(1):37–66. http://dx.doi.org/10.1023/A:1022689900470

[17] Breiman, L., Friedman, J. H., Olshen, R. A. et Stone, C. J. (1984). Classification and Regression Trees. Chapman & Hall, New York, NY.

[18] Yang, Y. (1999). An evaluation of statistical approaches to text categorization. Journal of Information Retrieval, 1:67–88.

[19] Dumais, S., Platt, J., Heckerman, D. et Sahami, M. (1998). Inductive learning algorithms and representations for text categorization. In CIKM '98 : Proceedings of the seventh international conference on Information and knowledge management, pages 148–155, New York, NY, USA. ACM.

[20] : Laghmassi Leila & Sekkiou Amel (2016/2017), «Application des méthodes d'apprentissage automatique pour l'inférence des réseaux de régulation génétique» , mémoire de master en Informatique Université Abbes Laghrour Khenchela (p 40 à p 43).

[21] S.Stern Clearwater. A rulelearning program in high energy physics event classi_cation_1991

[22] Povost.F Fawcett.T. Adaptative fraud detection, volume 1 of data mining and knowledge discovery, 1997.

[23] Povost.F Fawcett.T. Robust classi_cation systems for imprecise environments, 2001.

[24] Matwin.S Kubat.M, Holte.R. Machine learnong for the detection of oils spills in satellite radar images.

[25] Lewis.D. Evaluating text categorization. in proc. workshop on speech and natural language.

[26] Lewis.D. Representation quality in text classi_cation :an introduction and experiment. In proc. workshop on speech and natural language, 1990.

[27] Métais E Nakache D. Evaluation : nouvelle approche avec juges, 2005.

[28] Neri.F Saitta.L. Learning in the "real world".

[29] https://tech-fr.netlify.app/articles/fr512032/index.html.

[30] Similarity-Based Approach for Positive and Unlabelled Learning , Yanshan Xiao , Bo Liu , Jie Yin, Longbing Cao, Chengqi Zhang, Zhifeng Hao.

[31] Fung et al., 2006: G. P. C. Fung, J. X. Yu, H. Lu, and P. S. Yu. Text classification without negative examples revisit. TKDE, 18:6–20,2006.

[32] Liu et al., 2002: B. Liu, W. S. Lee, P. S. Yu, and X. Li. Partially supervised classification of text documents. ICML, 2002.

[33] Yu et al., 2004 :H. Yu, J. Han, and K. C. C. Chang. Pebl: web page classification without negative examples. TKDE, 16(1): 70–81, 2004.

[34] Li and Liu, 2003 : X. Li and B. Liu. Learning to classify texts using positive and unlabeled data. IJCAI, pages 587–592, 2003.
[35] Liu et al., 2003 : B. Liu, Y. Dai, X. Li, W. S. Lee, and P. S. Yu. Building text classifiers using positive and unlabeled examples. ICDM, pages 179–186, 2003.
[36] Li et al., 2009 : X. L. Li, P. S. Yu, B. Liu, and S. K. NG. Positive unlabeled learning for data stream classification. SDM, Pages 257–268, 2009.
[37] Rocchio, 1971: J. Rocchio. Relevance feedback in information retrieval. In G. Salton (ed), The SMART retrieval system: Experiments in automatic document processing. Prentice Hall, Englewood Cliffs, NJ, 1971.
[38] Scholkopf et al., 2001 : B. Scholkopf, J. C. Platt, J. C. Shawe-Taylor, A. J. Smola, and R. C. Williamson. Estimating the support of a high-dimensional distribution. Neural Computation, 13:1443–1471, 2001.
[39] Vapnik, 1998: V: Vapnik. Statistical learning theory. Springer, 1998.
[40] Buckley et al., 1994: C. Buckley, G. Salton, and J. Allan. The effect of adding relevance information in a relevance feedback environment. ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR), pages 292–300, 1994
[41] Vapnik, 1995 **:**Vapnik, V. N. (1995). The nature of statistical learning theory. Springer-Verlag New York, Inc., New York, NY, USA.
[42] Joachims, 1998 : Joachims, T. (1998). Text categorization with suport vector machines: Learning with many relevant features. In ECML '98 : Proceedings of the 10th European Conference on Machine Learning, pages 137–142, London, UK. Springer-Verlag.
[43] Survey of Swarm Intelligence Algorithms: Suganya Selvaraj Department of Financial Information Security . **And** Eunmi Choi Department of Financial Information Security / Department of Software,College of Computer Science,
[44] Hassanien A.E., Emary E., "Swarm Intelligence: Principles, Advances, and Applications", CRC Press, Taylor & Francis Group, 2015.
[45] H. Ahmed, J. Glasgow, Swarm intelligence: concepts, models and applications. Technical Report. Queen's University, Canada, School of Computing (2012).
[46] S. Binitha and S. S. Sathya, "A survey of bio inspired optimization algorithms," International Journal of Soft Computing and Engineering, vol. 2, pp. 137–151, 2012.
[47] Odili JB, Kahar MNM (2016) Solving the traveling salesman's problem using the African buffalo optimization. Comput Intell Neurosci 2016:3.
[48] G. Dong et al., "Solving Traveling Salesman Problems with Ant Colony Optimization Algorithms in Sequential and Parallel Computing Environments: A Normalized Comparison." Int. J. Mach. Learn. Comput., 8(2), 98-103, 2018.
[49] Shima Sabet, Mohammad Shokouhifar, and Fardad Farokhi; "A Comparison Between Swarm Intelligence Algorithms For Routing Problems"; Electrical & Computer Engineering: An International Journal (ECIJ) Volume 5, Number 1, March 2016.
[50] Brezočnik, Lucija, et al., "Swarm Intelligence Algorithms for Feature Selection: A Review." Applied Sciences 8.9 (2018):1521.
[51] Basir, M.A.; Ahmad, F. Comparison on Swarm Algorithms for Feature Selections Reductions. Int. J. Sci. Eng. Res. 2014, 5, 479–486.
[52] Fan, J.; Hu, M.; Chu, X.; Yang, D. A comparison analysis of swarm intelligence algorithms for robot swarm learning. In Proceedings of the 2017 Winter Simulation Conference (WSC), Las Vegas, NV, USA, 3–6 December 2017; pp. 1–6.
[53] Figueiredo et al., "Swarm intelligence for clustering—A systematic review with new perspectives on data mining" Engineering Applications of Artificial Intelligence, 82 (2019), pp. 313-329.
[54] X. Gong, L. Liu, S. Fong, Q. Xu, T. Wen and Z. Liu, "Comparative Research of Swam Intelligence Clustering Algorithms for Analyzing Medical Data," in IEEE Access, vol. 7, pp. 137560-137569, 2019. doi:10.1109/ACCESS.2018.2881020
[55] GF Elhady, M A. Tawfeek, "A comparative study into swarm intelligence algorithms for dynamic tasks scheduling in cloud computing", Intelligent Computing and Information Systems (ICICIS) 2015 IEEE Seventh International Conference on, pp. 362-369, 2015.
[56] S.J. Mohana, M. Dr, Dr Saroja, M. Venkatachalam Comparative analysis of swarm intelligence optimization techniques for cloud scheduling, IJISET, 1 (10) (2014), pp. 15-19.
[57] Askarzadeh, A. (2016). A novel metaheuristic method for solving constrained engineering optimization problems: Crow search algorithm. Computers & Structures, 169, 1–12. https://doi.org/10.1016/j.compstruc.2016.03.001
[58] Blum C, Roli A. Metaheuristics in combinatorial optimization: overview and conceptual comparison. ACM Comput Surv 2003;35:268–308.
[59] Yang XS. Metaheuristic optimization. Scholarpedia 2011;6. 11472.
[60] Glover F. Future paths for integer programming and links to artificial intelligence. Comput Oper Res 1986;13:533–49.
[61] Yang XS. Nature-inspired metaheuristic algorithms. Luniver Press; 2008.
[62] Yang XS. Engineering optimization: an introduction with metaheuristic applications. Wiley; 2010.
[63] Holland J. Adaptation in natural and artificial systems. Ann Anbor: University of Michigan Press; 1975.
[64] Kennedy J, Eberhart RC. Particle swarm optimization. In: Proc of IEEE international conference on neural networks, Piscataway, NJ; 1995. p. 1942–48.
[65] Geem ZW, Kim JH, Loganathan GV. A new heuristic optimization algorithm: harmony search. Simulation 2001;76:60–8.
[66] Yang X-S, Deb S. Cuckoo search via L_evy flights. In: Proceedings of world congress on nature & biologically inspired computing (NaBIC), Coimbatore, India; 2009. p. 210e4.

[67] Yang XS. A new metaheuristic bat-inspired algorithm. In: Gonzalez JR et al., editors. Nature-inspired cooperative strategies for optimization (NICSO 2010). Springer, SCI 284; 2010. p. 65–74.
[68] He S, Wu QH, Saunders JR. Group search optimizer: an optimization algorithm inspired by animal searching behavior. IEEE Trans Evol Comput 2009;13:973–90.
[69] Yang XS. Firefly algorithm, stochastic test functions and design optimisation. Int J Bio-Inspired Comput 2010;2(2):78–84.
[70] Rincon, Paul, Science/nature|crows and jays top bird IQ scale, BBC News.
[71]Prior H, Schwarz A, Güntürkün O. Mirror-induced behavior in the magpie (picapica): evidence of self-recognition. PLoS Biol 2008;6(8):e202.
[72] : https://fr.sawakinome.com/articles/science/difference-between-qualitative-and-quantitative-data.html #Quantitative%20Data.
[73] Dino Ienco, Ruggero G. Pensa. Positive and unlabeled learning in categorical data. Neurocomputing, (pages 11 et 12): Elsevier, 2016, 196 (july), pp.113 - 124. 10.1016/j.neucom.2016.01.089. hal-01374450 (HAL Id: hal-01374450 https://hal.archives-ouvertes.fr/hal-01374450)
[74] Y. Xiao, B. Liu, J. Yin, L. Cao, C. Zhang, Z. Hao, Similarity-based approach for positive and unlabeled learning, in: Proceedings of IJCAI 2011, AAAI, Palo Alto, CA, USA, Barcelona, Spain, 2011, pp. 1577-1582.
Kookmin University, Seoul, South Korea.
[75] Density Estimators for Positive-Unlabeled Learning : Teresa M. A. Basile2,3, Nicola Di Mauro1(B) , Floriana Esposito1 ,Stefano Ferilli1, and Antonio Vergari1 ** 1 Department of Computer Science, University of Bari “Aldo Moro”, Bari, Italy , nicola.dimauro@uniba.it , 2 Department of Physics, University of Bari “Aldo Moro”, Bari, Italy , 3 National Institute for Nuclear Physics (INFN), Bari Division, Bari, Italy**