

|  |
| --- |
|  |
| Rapport de Data-Mining |
|  |
| Méthode de classification : k plus proches voisins |

**EISTI**

14 décembre 2015

Créé par : Hugo LEGRAND et Simon ESPIGOLÉ

Table des matières

[Introduction 2](#_Toc437726546)

[Algorithme 3](#_Toc437726547)

[Génération du jeu de données 3](#_Toc437726548)

[La méthode des k plus proches voisins 3](#_Toc437726549)

[Résultats et analyses 4](#_Toc437726550)

# Introduction

Afin de mieux appréhender les différentes méthodes de Data-Mining, nous avons réalisé une implémentation de la méthode de classification des k plus proches voisins. Cette méthode est dite supervisée : elle nécessite en effet une base d’apprentissage pour fonctionner puisqu’elle consiste à se référer à des données déjà existantes pour pouvoir classer un nouvel individu dans la même catégorie que ceux ayant des caractéristiques semblables.

Pour réaliser et tester cette méthode dans des conditions acceptables, il est nécessaire de construire des jeux de données de différentes formes (pseudo-aléatoires, fortement dépendant, peu dépendant).

Les jeux de données que nous utilisons sont constitués de valeurs non nommées que nous générons et que nous pouvons paramétrer par leur nombre d’individus, d’attributs et de classes.

# Algorithme

## Génération du jeu de données

Le jeu de donnée que nous générons est pseudo-aléatoire, nous utilisons la méthode de Box-Muller pour générer des nombres aléatoires à distribution normale centrée réduite, à partir d’une source de nombres aléatoires de loi uniforme, dont la formule mathématique est la suivante :

Avec et deux variables aléatoires indépendantes uniformément distribuées sur.

Pour maîtriser la répartition des valeurs, nous avons choisi de les borner pour les situer entre 0 et 10 environ. L’avantage d’avoir choisi un jeu de donnée non nommé nous permet d’avoir des valeurs sans grand rapport entre les différents attributs de chaque individu. Nos données sont des réels tronqués à une décimale.

## La méthode des k plus proches voisins

Nous pouvons grâce aux paramètres variables de notre programme changer le nombre d’individus et d’attributs de chaque jeu de données généré : cela nous a notamment permis de tester notre algorithme sur des jeux de test plus réduits pour étudier chaque étape de celui-ci. En effet, nous avons eu beaucoup de mal au début de l’implémentation de l’algorithme de par la difficulté de celui-ci : des valeurs totalement aberrantes ont été observées dans les résultats des classes calculées lors de l’exécution du programme, ce qui avait pour cause de pousser le pourcentage d’erreur à des valeurs maximales, dont nous n’arrivions pas à trouver les causes malgré plusieurs heures de recherche.

En outre, il est aussi possible de choisir dans notre programme la proportion du jeu de données à utiliser en tant que base d’apprentissage, la partie restante étant celle à classifier parmi les autres. La première partie de l’algorithme consiste à calculer la distance entre l’individu en cours de classification et tous les autres individus. Nous avons choisis d’utiliser la distance euclidienne représentée mathématiquement comme suit :

Les k individus ayant la distance la plus petite sont retenus comme étant les k plus proches voisins de notre individu en cours de classification. Ces individus nous permettent de déterminer la classe de notre individu : en effet la classe de l’individu sera celle qui est la plus représentée parmi les k plus proches voisins.

Cette méthode est alors itérée sur tous les individus qui n’ont pas été utilisés dans la base d’apprentissage. Nous en déduisons à la suite un taux d’erreur sur la détermination de ces classes. Celui-ci est tout simplement le nombre de classes non correctement calculées par le nombre d’individus à classifier.

# Résultats et analyses