Rapport TME 4

Filtrage collaboratif

# Introduction

Objectifs

Présentation base de données (rapide ++)

(recopier intro du TME)

# Méthode

## Base de donnée

* Stats du site

## Découpage train test

Afin de nous permettre d’évaluer les différentes méthodes de prédiction nous avons découpé le set de données en deux : un training set et un testing set.

Pour construire le testing set nous avons dans un premier temps parcouru les utilisateurs et sélectionné un film au hasard parmi les films notés. Nous avons ensuite sélectionné pour chaque film une note donnée par un utilisateur au hasard. Toutes les informations extraites de la base de données afin de constituer le testing set ont été supprimé du training set.

Une fois constitué le testing set contenait 2472 notes ce qui correspond à 2.5% de la base de données initiale.

## Méthode d’évaluation

Pour nous permettre de mieux interpréter les évaluations des résultats obtenus nous avons réalisé des tests avec des algorithmes naïfs. Il est important de noté que les résultats de prédiction des algorithmes seront arrondit à l’entier le plus proche pour respecter le format de la base de données.

Nous avons dans un premier temps prédit les notes avec un entier aléatoire entre 1 et 5.

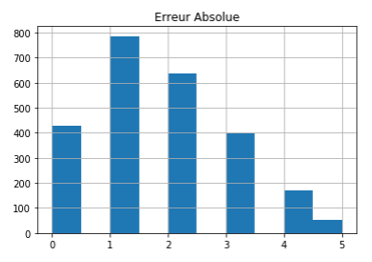
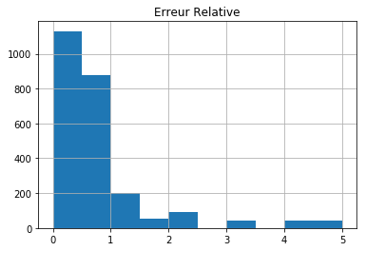


Figure 1: Répartition de l'erreur pour le tirage aléatoire

|  |  |  |
| --- | --- | --- |
|  | **Erreur Relative** | **Erreur Absolue** |
| **count** | 2472 | 2472 |
| **mean** | 0.682 | 1.695 |
| **std** | 0.892 | 1.242 |
| **min** | 0.000 | 0.000 |
| **25%** | 0.250 | 1.000 |
| **50%** | 0.500 | 2.000 |
| **75%** | 0.750 | 3.000 |
| **max** | 5.000 | 5.000 |

Figure 2: Statistiques de répartition de l'erreur en tirage aléatoire

On obtient bien une répartition normale de l’erreur absolue liée au tirage aléatoire.

Nous avons ensuite testé en remplaçant la note prédite par la moyenne de toutes les notes. Dans notre cas la moyenne était de 3.6 nous avons donc remplacé par 4.

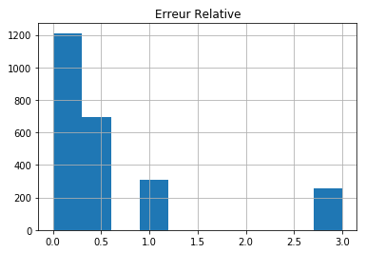
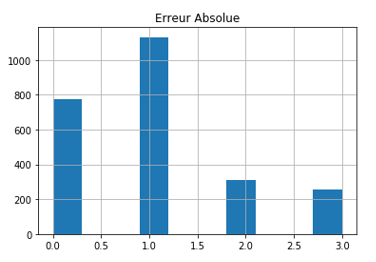


Figure 3: Répartition de l'erreur pour une prédiction constante

|  |  |  |
| --- | --- | --- |
|  | **Erreur Relative** | **Erreur Absolue** |
| **count** | 2472 | 2472 |
| **mean** | 0.566 | 1.020 |
| **std** | 0.883 | 0.924 |
| **min** | 0.000 | 0.000 |
| **25%** | 0.000 | 0.000 |
| **50%** | 0.333 | 1.000 |
| **75%** | 0.333 | 1.000 |
| **max** | 3.000 | 3.000 |

Figure 4: Statistiques de répartition de l'erreur pour une prédiction constante

On observe qu’en moyenne les résultats sont meilleurs avec une prédiction constante à la valeur moyenne qu’avec un tirage aléatoire.

# Prédiction SVD

## Construction de l’algorithme

## Résultats

# Prédiction NMF

## Construction de l’algorithme

## Résultats

# Prédiction gradient

## Algorithme avec L2

### Construction de l’algorithme

L’algorithme mis en place correspond à une descente de gradient stochastique avec pénalisation L2. Pour cela nous avons bouclé sur les notes non nulles contenues dans le training set et avons cherché à minimiser le coût :

### Résultats

Nous avons dans un premier temps cherché à tester sans prendre en compte la pénalisation L2. Les paramètres de tests étaient les suivants :

* Dimension : 2
* : 0
* :
* Maximum d’itération : 5

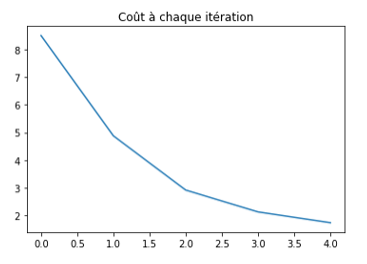
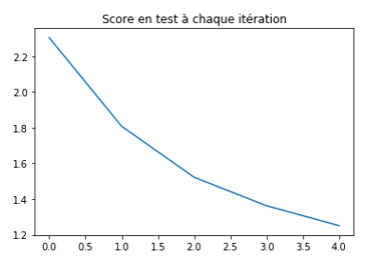


Figure 5: Validation du bon fonctionnement de l'algorithme sans pénalité L2

On obtient des courbes descendantes autant pour le coût que pour le test ce qui nous permet de vérifier que l’algorithme a été codé correctement. On peut ensuite s’intéresser au résultats.

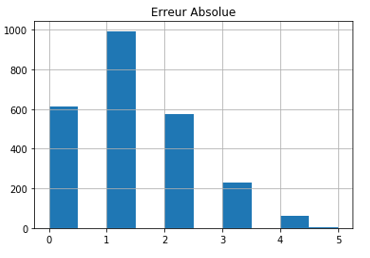
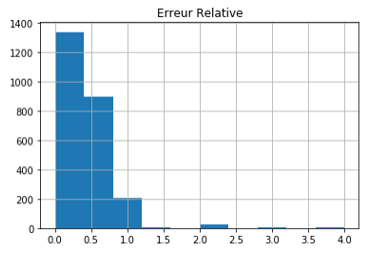


Figure 6: Répartition de l'erreur sans pénalité L2

|  |  |  |
| --- | --- | --- |
|  | **Erreur Relative** | **Erreur Absolue** |
| **count** | 2472 | 2472 |
| **mean** | 0.399 | 1.276 |
| **std** | 0.367 | 1.023 |
| **min** | 0 | 0 |
| **25%** | 0.2 | 1 |
| **50%** | 0.333 | 1 |
| **75%** | 0.6 | 2 |
| **max** | 4 | 5 |

Figure 7: Statistiques de répartition de l'erreur sans pénalité L2

On obtient en moyenne de bien meilleur résultats qu’avec les algorithmes utilisés précédemment. L’erreur absolue semble être répartie selon une loi normale autour de 1.

Dans un deuxième temps nous avons pris en compte la pénalité L2. Les paramètres de tests étaient les suivants :

* Dimension : 2
* : 0.001
* :
* Maximum d’itération : 5

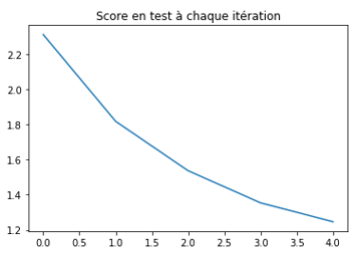
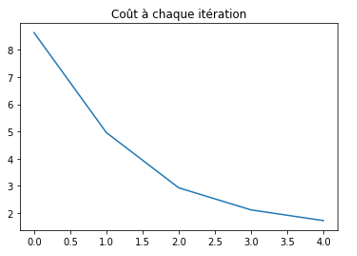
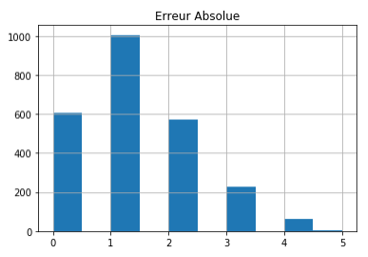
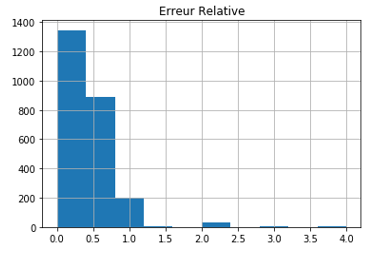


Figure 8 : Validation du bon fonctionnement de l'algorithme avec pénalité L2

Les courbes descendantes nous permettent de valider le bon fonctionnement de l’algorithme.

Figure 9: Répartition de l'erreur avec pénalité L2



|  |  |  |
| --- | --- | --- |
|  | **Erreur Relative** | **Erreur Absolue** |
| **count** | 2472 | 2472 |
| **mean** | 0.392 | 1.246 |
| **std** | 0.365 | 1.011 |
| **min** | 0 | 0 |
| **25%** | 0.2 | 1 |
| **50%** | 0.333 | 1 |
| **75%** | 0.5 | 2 |
| **max** | 4 | 5 |

Figure 10: Statistiques de répartition de l'erreur avec pénalité L2

Ajouter la pénalisation L2 permet de faire baisser l’erreur moyenne qu’elle soit absolue ou relative. La répartition de l’erreur est sensiblement la même avec et sans pénalisation L2.

### Conclusion

Ajouter la pénalisation L2 à l’algorithme de descente de gradient stochastique permet d’améliorer sensiblement les résultats mais ne change pas la répartition de l’erreur absolue. Une recherche du meilleur paramètre pourrait être conduite afin de minimiser l’erreur moyenne pour cette méthode.

## Algorithme L2 avec biais

### Construction de l’algorithme

Afin de tenir compte du biais dans l’algorithme de descente de gradient stochastique avec pénalité L2, nous avons modifié la fonction coût vue précédemment :

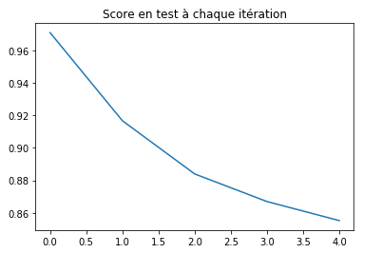
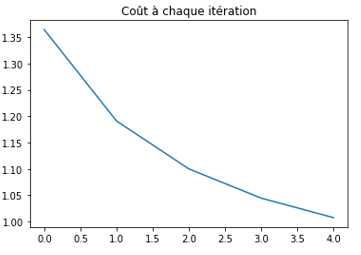
### Résultats

Nous avons testé l’algorithme avec biais en utilisant aussi la pénalisation L2. Les paramètres de tests étaient les suivants :

* Dimension : 2
* : 0
* :
* Maximum d’itération : 5

Les résultats obtenus :

Figure 11: Validation du bon fonctionnement de l'algorithme avec biais et pénalisation L2



Les courbes descendantes nous permettent de valider le bon fonctionnement de l’algorithme.

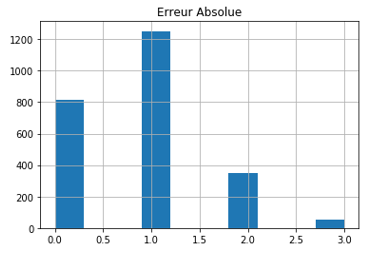
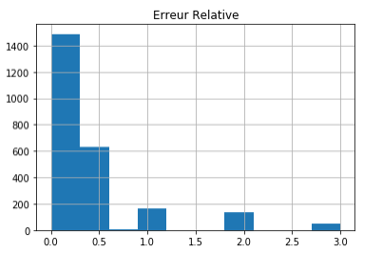


Figure 12: Répartition de l'erreur pour une prédiction avec biais et pénalité L2

|  |  |  |
| --- | --- | --- |
|  | **Erreur Relative** | **Erreur Absolue** |
| **count** | 2472 | 2472 |
| **mean** | 0.398 | 0.855 |
| **std** | 0.599 | 0.734 |
| **min** | 0 | 0 |
| **25%** | 0 | 0 |
| **50%** | 0.25 | 1 |
| **75%** | 0.4 | 1 |
| **max** | 3 | 3 |

Figure 13: Statistiques de répartition de l'erreur pour une prédiction avec biais et pénalité L2

La valeur moyenne de l’erreur relative est légèrement plus élevée pour l’algorithme avec biais par rapport à celui sans biais. Cependant l’erreur absolue passe de 1.246 à 0.885 avec la prise en compte des biais. De plus lorsque l’on analyse la répartition de l’erreur relative on observe que pour la méthode avec biais le nombre d’erreurs comprises entre 0 et 0.25 est plus élevé que pour la méthode sans biais. On pourrait mener des analyses en augmentant le nombre d’itération afin de voir si l’erreur relative de la méthode avec biais devient inférieure à celle de la méthode sans biais. Là encore la répartition de l’erreur n’est pas modifiée.

## Conclusion

On a pu observer que l’ajout de pénalité et/ou de biais permet d’améliorer les scores de la méthode de descente de gradient stochastique, que ce soit en erreur relative ou en erreur absolue. Quelle que soit la méthode utilisée la répartition de l’erreur reste sensiblement la même est suit une loi normal centrée en 1.

Pour ce qui est de l’ajout de biais dans la méthode avec pénalité L2, on obtient une erreur absolue plus faible 0.885 contre 1.246. Cependant l’erreur relative augmente sensiblement et passe de 0.398 à 0.392. Cette différence s’explique par la répartition de l’erreur relative. Bien qu’en nombre plus élevée entre 0 et 0.25, la quantité d’erreur pour des valeurs supérieures à 2 est plus importante.

# Conclusion

* Comparaison de toutes les méthodes