***Le Machine Learning : une solution numérique dans la lutte contre les pandémies mondiales***

**Table des matières**

1. **Présentationp1**
2. **Positionnement thématiquep2**
3. Bibliographie commentéep2
4. Problématique retenuep3
5. Objectif de travailp3
6. Bibliographiep3
7. Abstractp4
8. DOTp4
9. Introduction aux algorithmes SVMp5

**10) Etude mathématique des SVMp8**

**10.1) Maximisation de la marge sous la forme primale et la forme duale p8**

**10.2) Le Kernel Trickp12**

**10.3) Les cas non séparablesp14**

**10.4) Catégoriser les données en plusieurs partiesp16**

**11) Listing des algorithmesp16**

**12) Trame présentationp20**

1. ***Présentation***

**Sujet** :

La détermination de la nature d’une pandémie selon ses caractéristiques propres grâce aux algorithmes SVM.

**Motivation** :

Je constate l’apparition de plus en plus fréquente de pandémies mondiales relevant de virus/bactéries connues ou inconnues. Il est important de connaitre la nature de la pandémie pour lutter au mieux contre celle-ci. Je cherche donc à déterminer par des algorithmes SVM (Support Vector Machine) la nature d’une pandémie.

**Lien au thème** :

L’émergence de pandémies mondiales de plus en plus violentes constitue un enjeu actuel. Les algorithmes de Machine Learning sont susceptibles de trier et adapter des solutions aux problèmes engendrés, grâce à de larges plages données. On comprend l’importance de cette technologie qui apporte une réponse à un enjeu social.

1. ***Positionnement Thématique***

INFORMATIQUE (Informatique théorique), MATHEMATIQUES (Algèbre)

*Informatique :*

*Informatique :*

* Functional programming
* Artificial intelligence (Machine Learning)
* Database

*Mathématiques :*

* Pre-Hilbert space
* Mathematical optimization
* Programmation fonctionnelle
* Intelligence artificielle (Machine Learning)
* Base de données

*Mathématiques :*

* Espaces préhilbertiens
* Mathématiques de l’optimisation

1. **Bibliographie commentée**

Le Machine Learning est d’une grande utilité dans la résolution de problèmes sociaux et économiques **[8]**. Ces algorithmes, par leur capacité à traiter de larges plages de données à une rapidité fulgurante permettent des résultats très prometteurs et pourraient s’avérer être des atouts de taille pour le futur. Dans le champ des algorithmes de Machine Learning, les algorithmes SVM (Support Vector Machine) sont efficaces pour ce qui est de retourner une prédiction selon une base de données. Mais qu’est-ce qu’un algorithme SVM ?

C’est en 1995 que le concept des algorithmes SVM voit le jour dans une étude publiée et réalisée par Corinna Cortes et Vladimir Vapnik **[1]**. Derrière cet algorithme assez complexe se cachent des notions mathématiques très liées à l’algèbre et aux espaces préhilbertiens. En effet, les notions d’espaces vectoriels et d’hyperplans sont les bases des SVM **[2]**. Par ailleurs ceux-ci incluent aussi des problèmes d’optimisation quadratique, dont les multiplicateurs de Lagrange, méthode qui est expliquée très précisément par Baxter Tyson Smit **[6]**. L’objectif de ceux-ci est de répondre à un problème de classification. Prenons l’exemple suivant, on se place dans un plan divisé en deux catégories, les points et les carrés. L’objectif sera de séparer au mieux ces deux ensembles pour ensuite pouvoir prédire si un point quelconque appartient soit à la catégorie des carrés ou des ronds. C’est d’ailleurs ce que montre Carle Pearson sur son site internet **[3]**. Cependant dans la majorité des cas, les données ne sont pas linéairement séparables, la question se pose : comment rendre les données séparables ?

Certaines solutions existent pour contourner le problème dont celle du Kernel Trick qui est la plus utilisée **[4, 5]**. Cette méthode consiste à faire varier la dimension de l’espace vectoriel pour que les données de celui-ci soient à nouveau linéairement séparables. Par ailleurs, dans le cas où il y aurait beaucoup de données à traiter, les données ne seront jamais linéairement séparables, il faudra alors prévoir des marges d’erreurs en rajoutant des variables dites ressort (aussi appelées « Slack Variable »). C’est ce que montre Andrew Ng dans son étude des SVM **[7]**.

De nos jours, toute nouvelle donnée est enregistrée dans une base de données. Pour pouvoir effectuer une prédiction un algorithme SVM à besoin d’une base de données dont les données serviront de données d’entrainement. Ces données seront, comme évoqué précédemment, les données à classer avant de retourner une prédiction. Une communication entre la base de données et l’algorithme est donc nécessaire pour assurer le bon fonctionnement de l’algorithme.

Les épidémies ont toujours existé et se multiplient avec l’augmentation de la population mondiale. La difficulté quant à ces apparitions est la détermination et/ou l’identification de la maladie pour pouvoir combattre au mieux celle-ci. **[9]** Grâce à l’expérience acquise, les bases de données concernant les pandémies mondiales sont de plus en plus complètes. Ainsi, les SVM pourraient constituer un outil de choix dans la détermination de la nature d’une pandémie.

1. ***Problématique retenue :***

La démultiplication des pandémies mondiales constitue un danger pour notre civilisation. Comment l’utilisation des algorithmes SVM pourrait-elle permettre l’aménagement de solutions pour contrer ce problème en déterminant la nature d’une pandémie en fonction de caractéristiques facilement observables ?

1. ***Objectif de travail :***

* Programmer un algorithme SVM en utilisant la librairie Sklearn avec Python.
* Savoir faire comprendre à l’algorithme si une pandémie est connue ou non d’une base de données.
* Programmer un algorithme SVM sans utiliser de boites noires avec Python.
* Créer des bases de données en accord avec les spécificités de chaque algorithme et le sujet d’étude.
* Analyser la complexité, la fiabilité, les inconvénients et avantages des algorithmes créés.

1. ***Bibliographie***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Support : | Auteur : | Titre : | Références : |
| [1] | Digital | C.Cortes et V.Vapnik | Support Vector Networks | <http://image.diku.dk/imagecanon/material/cortes_vapnik95.pdf> |
| [2] | Digital | Melep | Un peu de Machine Learning avec les SVM | : <https://zestedesavoir.com/tutoriels/1760/un-peu-de-machine-learning-avec-les-svm/> |
| [3] | Digital | **Carle Pearson** | **SVM Linéaires** | <http://cedric.cnam.fr/vertigo/Cours/ml2/coursSVMLineaires.html> |
| [4] | Digital | **Cesarsouza** | **Kernel Functions for Machine Learning Applications** | <http://crsouza.com/2010/03/17/kernel-functions-for-machine-learning-applications/> |
| [5] | Digital | (Source : l’université de Lyon) | SVM | <http://eric.univ-lyon2.fr/~ricco/cours/slides/en/svm.pdf> |
| [6] | Digital | Baxter Tyson Smit | Lagrange Multipliers Tutorial in the Context of Support Vector Machines: | <http://www.engr.mun.ca/~baxter/Publications/LagrangeForSVMs.pdf> |
| [7] | Digitale | Andrew Ng | CS229 Lecture notes | <http://cs229.stanford.edu/notes/cs229-notes3.pdf> |
| [8] | Digital | Tic Magazine | Comment le Machine Learning lutte contre le réchauffement climatique ? | <https://ticmagazine.bf/comment-le-machine-learning-lutte-contre-le-rechauffement-climatique/> |
| [9] | Digital | Johannes Kleiner et Sean Tull | The mathematical structure of integrated information theory | <https://arxiv.org/abs/2002.07655> |
| [10] | Digital | (Source : ONU Info) | Il faut réduire les risques pour éviter une « ère des pandémies », avertissent des experts | [Il faut réduire les risques pour éviter une « ère des pandémies », avertissent des experts | ONU Info](https://news.un.org/fr/story/2020/10/1081052)  https ://news.un.org/fr/story/2020/10/1081052 |

1. **Résumé en anglais : Abstract (Non demandé)**

The issue that is at stake is how the use of machine learning algorithms in the struggle against climate change could be a major asset by organizing data and finding solutions with those data. To solve this problem, we will focus on SVM algorithms. We will explain how those work and we will try to highlight the process in a simulation based on data with a SVM algorithm. Thus, we will understand how those algorithms work and how they could change our future.

1. **DOT :**

Etape 1 : (Février - Mars 2020)

Recherche d’un domaine d’étude et d’une problématique.

Etape 2 : (Mars – Juin 2020)

Réalisation des calculs autour des SVM avec pour objectif la création d’un algorithme SVM. Cette étape s’est avérée être très intéressante notamment par les nombreuses méthodes nouvellement acquises et par l’application du cours de MPSI à un problème concret.

Etape 3 : (Juin 2020)

Echange avec un ancien polytechnicien aujourd’hui expert en sécurité informatique. Cet entretien fut très enrichissant par les nombreux renseignements acquis sur la programmation d’un SVM. Programmation d’un algorithme SVM avec la librairie Sklearn, réalisée après plusieurs tentatives infructueuses.

Etape 4 : (Septembre 2020)

Application de l’algorithme SVM avec Sklearn à des bases de données diverses et variées. Création de la base de données en lien avec le sujet traité.

Etape 5 : (Octobre – Décembre 2020)

Travail sur les critères à utiliser pour déterminer les maladies inconnues. Travail sur la complexité, points forts et failles de l’algorithme.

Etape 7 : (Décembre 2020 – Mai 2021)

Création d’un algorithme SVM sans boites noires et détermination de sa complexité.

Optimisation de l’algorithme pour réduire sa complexité. Série de tests pour évaluer les performances de l’algorithme.

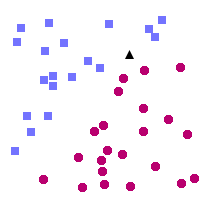
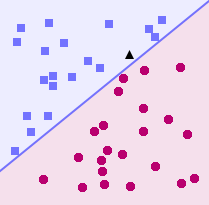
Etape 8 : (Mai – Juin 2021)

Production de graphiques pertinents pour la présentation finale.

1. **Introduction aux algorithmes SVM :**

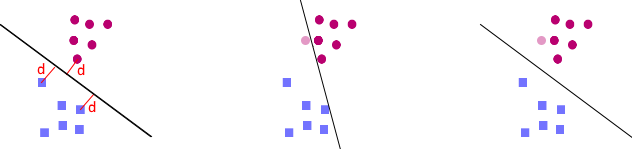
La démultiplication des pandémies mondiales constitue un danger pour notre civilisation. Comme souvent dans l’histoire de l’humanité, l’homme doit trouver des solutions pour lutter contre ce défi d’un nouveau genre. Le Machine Learning, une technologie créée en 1990, propose une réponse à cet enjeu. La question se pose à savoir comment l’utilisation des algorithmes SVM pourrait-elle permettre l’aménagement de solutions pour contrer ce problème en déterminant la nature d’une pandémie en fonction de caractéristique facilement constatables ?

Pour répondre à cette problématique, j’ai décidé de centrer mon étude sur les algorithmes SVM (Support Vector Machines) qui sont les plus pertinents au regard de la problématique des cours de Spé et les plus performants que l’on puisse trouver de nos jours. **Dans le cadre de l’initiative personnelle, je compte programmer un SVM en Python.** Les algorithmes SVM se basent sur l’utilisation des espaces vectoriels et mettent en jeu la notion d’hyperplan. L’objectif de ceux-ci est de répondre à un problème de classification. Prenons l’exemple suivant, on se place dans un plan divisé en deux catégories, les points et les carrés. L’objectif sera de séparer au mieux ces deux ensembles pour ensuite pouvoir prédire si un point quelconque appartient soit à la catégorie des carrés ou des ronds.

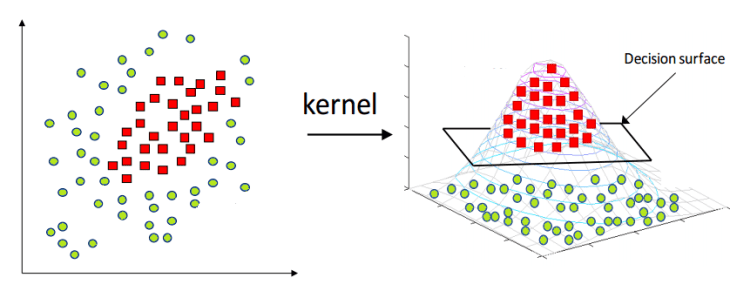
0 Exemple de classification d’un point quelconque.

On appelle ***capacité de généralisation*** : la frontière optimale la plus éloignée de tous les points d’entrainement. Par ailleurs, on appelle ***vecteur de support*** ces points d’entrainement les plus proches. Par conséquent, le but d’un SVM est de trouver cette frontière optimale, en maximisant la distance entre les points d’entraînement et la frontière, comme dans l’exemple ci-dessous :



1 Illustration de l'hyperplan optimal, c’est-à-dire à une distance d communes à tous les vecteurs de support.

Il arrive très souvent que la frontière ne soit pas linéaire, on dira alors que les points ne sont pas ***linéairement séparables***. La solution à ce problème est appelée *Kernel Trick (astuce du noyau)*. Le principe est donc de changer l’espace vectoriel initial en un espace vectoriel de plus grande dimension. Ci-dessous un exemple d’utilisation du Kernel Trick.



2 Illustration de l'action du Kernel Trick

Je baserai mon modèle d’algorithme sur différentes étapes nécessaires à la réalisation d’un SVM. Dans un premier temps je calculerai la marge nécessaire pour séparer linéairement les points à différencier. Ensuite il nous faudra maximiser cette marge, ceci permettra d’obtenir un hyperplan optimal qui générera donc une capacité de généralisation maximale. Maximiser la marge, c’est aussi maximiser la confiance et donc minimiser la probabilité d’erreur associée au classifieur. Enfin, il faudra normaliser la marge. Cette partie de la résolution est la plus compliquée. Celle-ci met notamment en jeu des problèmes d’optimisation quadratique, en particulier la méthode des multiplicateurs de Lagrange.

Pour créer un SVM, il faut donc générer le problème sous forme d’équations, d’où cette brève introduction mathématique au SVM. L’approche d’un SVM consiste à créer une fonction de classement linéaire de la forme suivante :

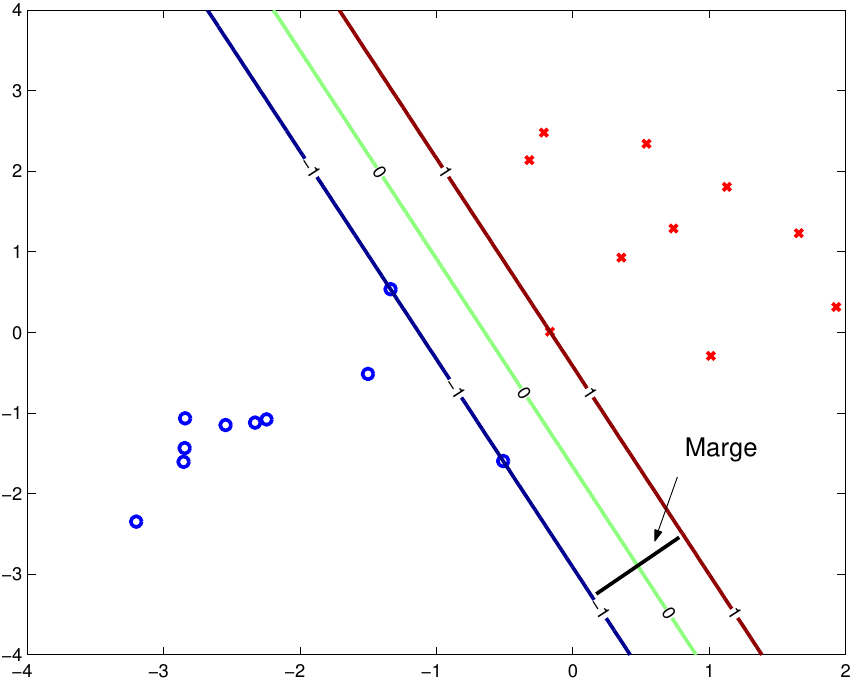
Soit E un espace vectoriel finie de dimension n, et un hyperplan associé de dimension n. L’équation caractéristique de h est donc, où sont des scalaires et une famille de vecteurs. Par ailleurs, il faut que l’hyperplan soit affine, ainsi je rajouterai à le scalaire βo, ce scalaire est nommé *biais*. Une fois l’entrainement terminé, pour classer une nouvelle entrée , le SVM aura juste à déterminer le signe de , tel que :

Aussi, j’assigne à tout point un label qui vaut 1 ou -1 selon la catégorie ou se trouve . Ainsi, je trouve

En résumé, je souhaite savoir, pour un point , s’il se trouve d’un côté ou de l’autre de l’hyperplan. La fonction me permet de répondre à cette question, grâce à la classification suivante :

1. **Etude mathématique des SVM :**

La première étape consiste à maximiser la marge, c’est-à-dire, l’hyperplan optimal doit maximiser la distance entre la frontière de séparation et les points de chaque classe qui lui sont le plus proche. Je note la distance entre un point quelconque et l’hyperplan Cette distance d’un point quelconque x avec la frontière est donnée par :



3 Illustration de la marge par rapport à l’hyperplan (ligne verte).

Soit la fonction définie par : . On appelle cette fonction la ***marge.*** Une fois la marge trouvée, il faut la maximiser. Pour cela, il existe deux formes, lesquels sont l’expression primale et l’expression duale.

* 1. **Maximisation de la marge sous la forme primale et la forme duale :**

J’ai donc :

J’impose alors la condition de normalisation pour les vecteurs de support , c’est-à-dire un point x placé sur la frontière, ainsi :

Or maximiser la marge revient à minimiser .

Je viens de trouver, la forme primale de la marge. Tout ce qui suit fait donc parti de la forme duale de la marge.

Je dois résoudre le problème suivant :

Minimiser sous la contrainte .

Un problème d’optimisation possède une forme duale si on évolue dans un espace convexe, ce qui est le cas. Ainsi, j’utilise alors la méthode des multiplicateurs de Lagrange (problème d’optimisation quadratique) pour résoudre ce problème.

**DEFINITION** ***espace convexe :*** *Une partie P d'un espace affine E est dite convexe si, pour toute paire de points A et B de P, le segment [AB] est inclus dans P.*

Soit , je maximise sous la contrainte suivante :

.

Je pose

On appelle *multiplicateur de Lagrange.*

Par projection, j’obtiens :

Or par la condition de Karush-Kuhn-Tucker, j’ai . C’est-à-dire, que je me place dans la situation où un point se situe sur la frontière.

Or donc

J’ai finalement déterminé les valeurs de et . Je reprends maintenant la fonction L établie précédemment.

Je sais que :

Je réinjecte (1) et (2) dans (3) pour obtenir une expression de L qui ne dépend que du multiplicateur de Lagrange . Une fois trouvée, cette expression sera la forme duale de la marge.

Je pose le produit scalaire suivant :, et montre que c’est bien un produit scalaire de .

***Montrons que est symétrique :***

Le produit est commutatif dans donc :

***Montrons que est bilinéaire :***

**Enfin, montrons que *:***

J’ai montré que est un produit scalaire sur .

En utilisant ce produit scalaire je trouve donc :

J’ai trouvé la forme duale de la marge. La contrainte devient donc

,

Pour résumer, j’ai réussi à obtenir l’expression de et , ainsi, j’obtiens

De plus je connais désormais la forme primale et la forme duale de la marge, et leur contrainte associée.

Forme duale :

,

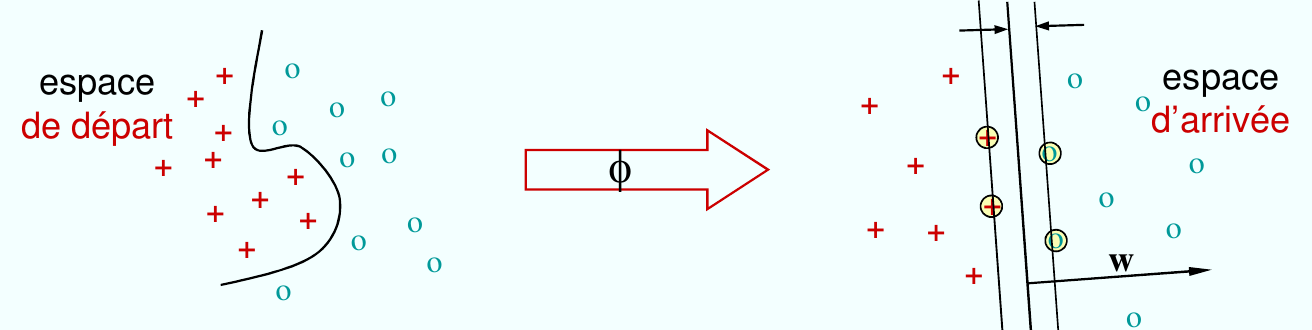
Forme primale :

La forme duale me permettra un travail plus approfondi des SVM notamment par l’étude du Kernel Trick. Cette astuce permet de séparer des données qui dans une dimension donnée ne sont pas linéairement séparables.

* 1. ***Le Kernel Trick***

Dans la plupart des cas les données ne sont pas linéairement séparables. Pour les rendre à nouveau linéairement séparables on utilise le Kernel Trick, un outil puissant et efficace. L’objectif et de changer la dimension de l’espace de travail pour pouvoir rendre les données linéairement séparables. C’est là qu’intervient la fonction L établie précédemment.

C’est en changeant le produit scalaire que la dimension de l’espace de travail change. En effet, je peux remplacer le produit scalaire ci-dessus par un autre. Soit une fonction de transformation. Soit un ensemble borné, fini, et , tels que :



4 Représentation graphique du Kernel Trick.

On appelle le *noyau positif défini* ou *kernel*. De cette manière, il suffit juste de connaitre pour pouvoir calculer le noyau. Calculer le noyau en utilisant serait trop couteux en calculs. Prenons l’exemple suivant, soit E l’espace de description de dimension n et E’ l’espace de re-description de dimension . Le calcul de se fait en tandis que le calcul de se fait . Pour un gain de temps, il faudra trouver directement la fonction , il n’est donc pas nécessaire de connaitre . Un noyau doit pouvoir satisfaire deux conditions, la condition de Mercer ou celle de la matrice de Gram pour prouver sa validité.

***Condition de Mercer :***  pour tout x, y et pour toutes fonctions de :

D’une manière plus concrète, doit être continue, symétrique et définie semi-positive.

***Matrice de Gram :***  , ,

doit être doit être définie positive. (

Je sais à présent comment vérifier si le produit scalaire est un noyau. Trois approches sont possibles pour obtenir une fonction noyau.

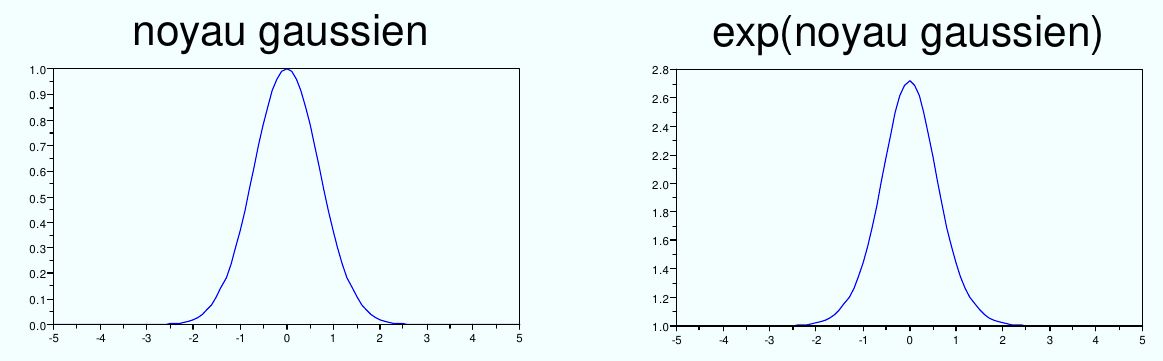
La première se fait par construction directe en utilisant . Je calcule d’abord et (pour ensuite faire le scalaire des deux fonctions.

La deuxième, consiste à transformer un noyau déjà existant, de sorte que celui-ci soit plus adapté au problème. Certains noyaux sont très utilisés dans les problèmes de classification, ci-joint se trouve les noyaux les plus utilisés.

|  |  |
| --- | --- |
| Noyau polynomial |  |
| Noyau gaussien |  |
| Noyau laplacien |  |
| Noyau rationnel |  |
| Noyau RBF |  |

Sur ces noyaux il est possible d’appliquer une fonction qui permettent un changement de dimension plus efficace. La seule condition à cela est que ce nouveau noyau soit toujours : symétrique, continu et défini semi-positif. Prenons l’exemple suivant :

Si est définie semi-positive, continue et symétrique, alors l’est aussi.



5 Exemple de transformation de noyau par l'exponentielle.

Enfin, la dernière approche consiste à effectuer des combinaisons linéaires de noyau défini semi-positifs, symétriques et continus.

Pour résumer, Je veux faire varier la dimension d’un espace pour que les données deviennent linéairement séparables. L’objectif est donc de trouver un produit scalaire qui puisse être en accord avec le théorème de Mercer. Aussi, je peux modifier des noyaux existants par combinaisons linéaires où en leur appliquant des fonctions qui ne changent pas leurs caractéristiques. Par ces méthodes il devient donc possible de changer n’importe quel espace de dimension finie en un autre espace dans lequel les données seront linéairement séparables.

* 1. ***Le cas non séparable***

Fort de ce qui a été dit précédemment, j’utilise le Kernel Trick pour rendre des données linéairement séparables, mais est-il toujours possible d’utiliser cette technique ? La réponse est non, il existe des ensembles qui ne seront jamais linéairement séparables. D’où l’introduction de variables qui autorisent les erreurs de classification. Ces variables sont appelées *slack variable*. La gestion d’une marge d’erreur est absolument nécessaire dans le traitement de larges bases de données, auquel cas le SVM ne renverrait rien faute de ne pas avoir trouvé une marge convenable. En outre, il est plus aisé de tolérer des erreurs que de chercher de nouveau noyaux pour séparer les données. La complexité deviendrait moindre, donc l’algorithme sera plus efficace. Soit une slack variable, celle-ci répond aux caractéristiques suivantes :

* est un vecteur de taille n
* S’il n’y a pas d’erreur :
* S’il y a une erreur :

L’enjeu est de minimiser pour avoir une marge d’erreur la plus infime qui soit et donc avoir une prédiction plus juste. On associe à un coût qui représentera la tolérance à l’erreur. Ce coût doit être intelligemment choisi. Soit le coût associé , si est trop grands, il y a risque de sur-apprentissage. Dans le cas inverse, il y a un risque de sous-apprentissage. J’obtiens en considérant la slack variable, les formes duales et primales suivantes :

Forme duale :

, ,

Forme primale :

Comme il faut minimiser à la fois et , devient donc :

Ainsi, la forme primale devient :

* 1. ***Catégoriser les données en plusieurs parties.***

Je sais comment séparer des données en deux catégories, mais est-il possible de séparer des données en plus de deux catégories ? Un hyperplan permet de séparer des données en deux catégories, ceci et donc un problème si je veux séparer les données plusieurs catégories. Pour cela, des méthodes existent, à savoir les méthodes du one-VS-all et du one-VS-one. Je ne traiterai pas cette partie de l’algorithme dans ce sujet.

1. ***Listing des algorithmes.***
2. import numpy as np
3. import sqlite3
4. from sklearn import svm, datasets
5. from sklearn.model\_selection import GridSearchCV, KFold
6. import time
8. def BDD():
10. bdd=sqlite3.connect('C:\Users\Stanislas Brusselle\Desktop\TIPE\BDD\BDD\_virus.db')
12. requete1 = """SELECT R0, Virulence, Incubation FROM Classification\_virus"""
13. X = np.array([ e for e in bdd.execute(requete1).fetchall()])
15. requete2 = """SELECT Virus FROM Classification\_virus"""
16. Y = [i for i in bdd.execute(requete2).fetchall()]
18. bdd.close
19. return([X, Y])


23. def ClassificateurNumerique(prediction):
25. clf = svm.SVC(kernel='rbf', gamma= ,C= , probability=True)
26. clf.fit(BBD()[0], BDD()[1])
27. predic=clf.predict([prediction])
29. #Ici commence la partie pour prédire si le virus est connu de la base de données ou non
31. #Renvoie les proba de d'incertitude liées à chaque catégories
32. W=clf.predict\_proba([prediction])
33. print(W)
34. #Séletionne la proba la plus élevée et sa place dans la liste.
35. a=W[0][0]
36. rang=0
37. for i in range(0, len(W[0])):
38. if a>=W[0][i]:
39. a=W[0][i]
40. rang=i
41. #Si la proba est plus élevée que 0.09 alors la prédiction est considérée comme nouveau virus.
42. if W[0][rang]>=0.09:
43. print('Virus inconnu a la base de donnees')
44. #Ajoute le virus à la base de données.
45. c=bdd.cursor()
46. c.execute('INSERT INTO Classification\_virus VALUES (?,?,?,?)', (raw\_input("Nom: "), prediction[0], prediction[1], prediction[2]))
47. bdd.commit()
48. bdd.close()
49. print("Le virus a bien ete enregistre")
50. else:
51. bdd.close()
52. return(predic, W[0][rang])
54. def Prediction():
56. #caractéristique de la pandémie à déterminer.
57. R0 = input("R0 : ")
58. Virulence = input("Virulence : ")
59. Incubation = input("temps d'incubation : ")
61. #retourne la nature de la pandémie.
62. return(ClassificateurNumerique([R0,Virulence, Incubation]))
64. #################FIN DU CODE##########################################
66. from math import\*
67. import numpy as np
68. import sqlite3
70. def Categories(X, Y):
71. #ATTENTION : Y ne doit contenir que deux catégories distinctes.
72. CD1=[]
73. CD2=[]
75. for i in range(0, len(Y)):
76. if Y[i]!=Y[0]:
77. CD1.append(X[i])
78. c=Y[i]
79. else:
80. CD2.append(X[i])
81. #Renvoie les valeurs de X triées selon leurs catégories.
82. return [[CD1, CD2],[c, Y[0]]]
84. def distanceH(x, n):
85. return((sqrt(produit\_scalaire(X,n)\*\*2))/(sqrt(produit\_scalaire(n,n))))
87. def distanceP(x, y):
88. return (sqrt(produit\_scalaire(x, y)))
90. def produit\_scalaire(X, Y):
91. c=0
92. for i in range(0, len(X)):
93. c+=X[i]\*Y[i]
94. return c
96. def norme(x):
97. return sqrt(produit\_scalaire(x, x))
99. def equahyperplan(barycentre, M, n):
100. #créer un vecteur à partir du barycentre et de M
101. coord=np.array(barycentre)-np.array(M)
102. #faire le produit scalaire de n et du vecteur coord
103. return produit\_scalaire(coord, np.array(n))
105. def verifopt(x, y, barycentre, n, Xinit):
106. l1=0
107. l2=0
108. if equahyperplan(barycentre, Xinit, n)>0:
109. for i in x :
110. if equahyperplan(barycentre, i, n)>0:
111. l1=1
112. else :
113. return False
114. for j in y:
115. if equahyperplan(barycentre, j, n)<0:
116. l2=1
117. else :
118. return False
119. if l2==1 and l1==1:
120. return True
122. elif equahyperplan(barycentre, Xinit, n)<0:
123. for i in x:
124. if equahyperplan(barycentre, i, n)<0:
125. l1=1
126. else :
127. return False
128. for j in y:
129. if equahyperplan(barycentre, j, n)>0:
130. l2=1
131. else :
132. return False
133. if l2==1 and l1==1:
134. return True
136. def BDD():
138. bdd=sqlite3.connect('C:\Users\Stanislas Brusselle\Desktop\TIPE\BDD\BDD\_virus\_algo2.db')
139. requete1 = """SELECT R0, Virulence, Incubation FROM Pandemie"""
140. X = [ e for e in bdd.execute(requete1).fetchall()]
142. requete2 = """SELECT Virus FROM Pandemie"""
143. Y = [i for i in bdd.execute(requete2).fetchall()]
144. bdd.close()
146. return([X, Y])


150. def Predict(X, Y, M, accuracy):
152. #Objectifs :
153. #Je vais calculer les distances à chaque point des deux catégories puis déterminer les valeurs minimales de distance.
154. #Enfin je vais faire passer un hyperplan séparant en deux ces distances trouvées.
155. #Il faudra ensuite effectuer une rotation de l'hyperplan autour d'un barycentre pour que la séparation soit optimale.
157. #Hypothèse simplificatrice :
158. #Je supposerai dans un premier temps les données sont linéairement séparables.
159. #Détermination du vecteur normal et du barycentre de rotation.
160. W=Categories(X, Y)
161. #ATTENTION : je définis x et y comme étant x les valeurs qui correspondent à la première classe et y celles de la deuxième.
162. x=W[0][0]
163. y=W[0][1]
164. label=W[1]
166. c=distanceP(x[0], y[0])
168. for i in x:
169. for j in y:
170. if distanceP(i, j)<=c:
171. c=distanceP(i, j)
172. L=[i, j]
173. Xinit=L[0]
174. #Calcul de la distance minimale entre A et B, ci-dessus.
175. #Détermination du barycentre de rotation de l'hyperplan
177. barycentre= np.array(L[1])-(c/2)
178. n=np.array(L[1])-barycentre
179. npremier=n
181. #Détermination de l'équation de l'hyperplan
183. result = equahyperplan(barycentre, M, n)
184. #optimisation de l'hyperplan
185. #J'optimise n dans le cas où l'hyperplan n'est pas optimal.
186. compteur=0
187. while verifopt(x, y, barycentre, n, Xinit)==False:
188. #J'utilise les matrices de rotations pour optimiser la position de l'hyperplan.
189. n =np.array(n).dot(np.array([[cos(pi/accuracy), -sin(pi/accuracy)], [sin(pi/accuracy), cos(pi/accuracy)]]))
190. print(n)
192. #Si après 100 calculs l'hyperplan optimal n'est pas trouvé, on quitte la fonction while et retourne une
193. #prédiction sans que l'hyperplan ne soit optimisé.
194. compteur+=1
195. if compteur>=100:
196. print('nbr itérations dépassé')
197. n=npremier
198. break
200. if equahyperplan(barycentre, Xinit, n)>0:
202. if result>0:
203. return(label[0])
204. else :
205. return(label[1])
207. if equahyperplan(barycentre, Xinit, n)<0:
209. if result<0:
210. return(label[0])
211. else :
212. return(label[1])
214. #################FIN DU CODE##########################################
215. X=BDD()[0]
216. Y=BDD()[1]
217. Predict(X, Y, [1.05, 1, 2], 5)
219. ***Trame présentation :***

Introduction :

La démultiplication des pandémies mondiales constitue un danger pour notre civilisation. Comme souvent dans l’histoire de l’humanité, l’homme doit trouver des solutions pour lutter contre ce défi d’un nouveau genre. Le Machine Learning, une technologie créée en 1990, propose une réponse à cet enjeu. La question se pose à savoir comment l’utilisation des algorithmes SVM pourrait-elle permettre l’aménagement de solutions pour contrer ce problème en déterminant la nature d’une pandémie en fonction de caractéristiques facilement observables ?

* **Diapo 3 :** annoncer les deux algo + pk 2 algo ?
* **Diapo 6 :** pk ces librairies ?

Initialisation SVM (l.19, l.23)

* **Diapo 7 :** initialisation BDD (l.8-16)

Réel travail de recherche derrière -> données réelles (sauf SARS-COV2)

Commenter performances SVM, idée épée à double tranchant avec SARS-COV2

Transition nouveau virus

* **Diapo 8 :** graphe descriptif de l.27-45

Pk 9%

*PredictProba* l.27

* **Diapo 9 :** Pk RBF ? -> connu comme fiable et tolérant
* **Diapo 10 :** commenter complexité
* **Diapo 11 :** paramètres optimaux pour 10000 données donc plus on tend vers 10000 plus le % d’erreur sera faible.
* **Diapo 12 :** Pk ces hypothèses ?

+ commenter idée (segment, médiatrice pour raisonner en 2D, mais s’étend à nD)

* **Diapo 13 :** matrice de rotations pour optimale (choix de la précision)

*Break* si >100 par soucie de complexité. (l.122-133)

* **Diapo 14 :** OPT NON à partir certain moment mais nbr data augmente.
* **Diapo 15 :** méthode pour avoir ces graphs -> créer deux ranges de données parfaitement linéairement séparables à partir de la fonction *random*. Prédire résultat connu et voir si la prédiction est « *correcte* » ou non. Cette méthode pour fiabilité à chaque fois.