Projet de Data Mining :

Support Vector Machine for Functional Data Classification

Nous avons décidé de réaliser notre projet de Data Mining sur un article scientifique qui applique une méthode de classification supervisée (SVM) sur des données fonctionnelles.

❖ Introduction

Dans un premier temps, il est important de définir ce que représente des données fonctionnelles. Ces données correspondent à la réalisation de variables aléatoires à valeur dans un espace de dimension infinie. Les observations ne vont plus être des variables aléatoires réelles ou vectorielles mais des fonctions aléatoires (courbes, images, …). Ces données sont de plus en plus fréquentes dans le monde actuel, on les retrouve dans divers domaines scientifiques (biologie, chimie, climatologie, industrie nucléaire, etc). Vous pouvez voir ci-joint des graphiques représentant des données fonctionnelles.

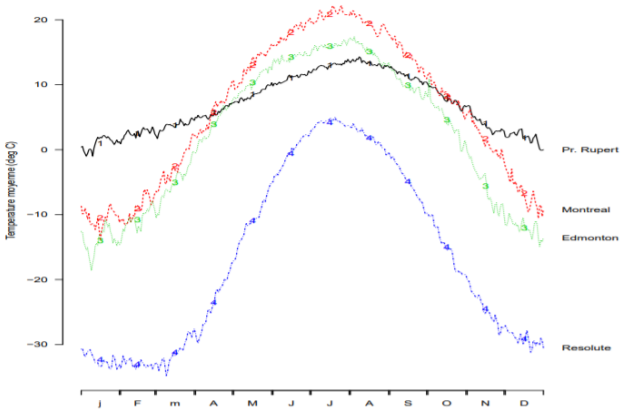


Figure 1 : Courbes de température au Canada

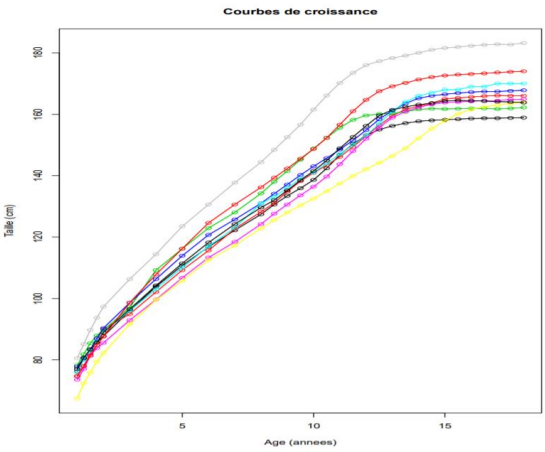


Figure 2 : Courbes de croissance de 10 américaines

Dans un second temps, nous devons expliquer le cadre mathématique dans lequel nous serons tout au long de ce document. Les données sont dans des espaces fonctionnels donc d'une dimension infinie. Nous disposons d'un ensemble d'apprentissage $(x_1, y_1), \ldots, (x_N, y_N)$, avec (x_i, y_i) des réalisations i.i.d du couple aléatoire (X,Y). X, la variable d'entrée, prends ses valeurs dans un espace de Hilbert arbitraire (<.,.> correspond à son produit scalaire). Y, la variable réponse, désigne les classes (les étiquettes) d'appartenance de X, ses valeurs sont {-1, 1}.

Enfin, nous allons étudier une méthode de classification supervisée. Le but est de déterminer le Y d'une nouvelle donnée X. Pour cela, l'algorithme va définir une règle de classification à partir des données d'apprentissages. Il pourra donc appliquer celle-ci sur des nouvelles entrées et obtenir leurs réponses.

Nous allons commencer, dans ce rapport, par un rappel sur la classification SVM (support vector machine ou machine à vecteurs de support). Ensuite, nous verrons comment adapter cette classification aux données fonctionnelles, notamment en utilisant des noyaux fonctionnels. Dans une dernière partie, nous présenterons les résultats que nous avons eu en appliquant la méthode vue sur des données du package fda.usc en R.

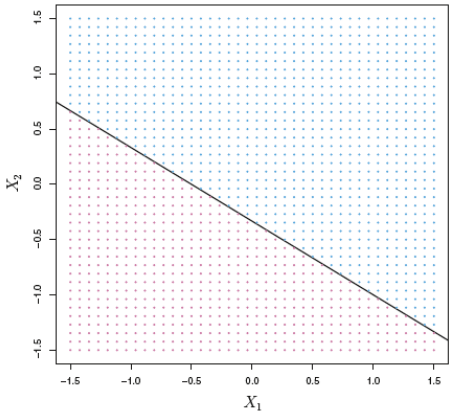
- Rappel sur la classification SVM (machine à vecteur de support) :

SVM (support vector machine ou machine à vecteurs de support) est une classification supervisée qui a vu le jour dans les années 1990. Il a été prouvé que les SVM fonctionnent dans divers domaines, et sont souvent considérés comme l'un des meilleurs classificateurs. Cette méthode repose sur la généralisation d'une approche simple et intuitive appelée le **classificateur à marge maximale.**

Tout d'abord, nous allons définir et introduire le concept d'un hyperplan séparateur optimal. Nous posons l'hypothèse que les données d'origine sont séparables linéairement. Dans un espace à p dimensions, un hyperplan est un sous-espace affine de dimension p-1. Par exemple, si l'on se place en deux dimensions, l'hyperplan représente une droite. Pour p-dimension, nous pouvons définir mathématiquement l'hyperplan grâce à son équation :

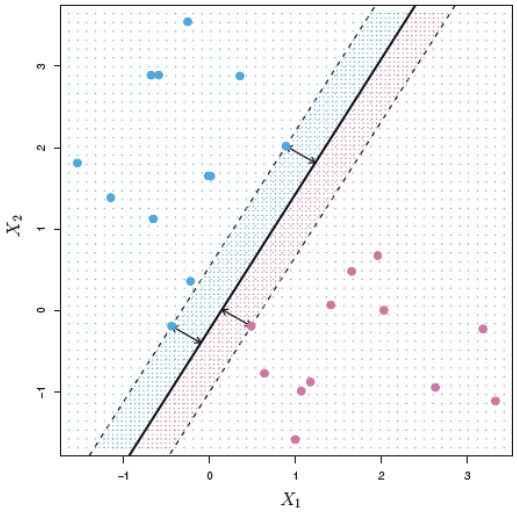
$$f(x) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_p X_p = 0$$

Si X ne satisfait pas l'équation de l'hyperplan, cela veut dire que l'observation se trouve d'un côté de celui-ci. Le signe de f(x) est très important, il permet de savoir de quel côté se positionne le point. Nous pouvons dire que l'hyperplan va donc diviser l'espace p-dimensionnel en deux parties. Vous pouvez voir ci-joint une représentation d'un hyperplan dans un espace bidimensionnel :



Dans le contexte d'une classification de données, nous devons réussir à choisir parmi tous les hyperplans envisageables celui qui sépare de façon optimale les différentes observations. En effet, la règle de classification, définie par l'hyperplan choisi, sera ensuite basée sur le signe de f(x). Si avec la nouvelle donnée on obtient f(x) positif alors l'étiquette prédite de cette observation est 1 (-1 si f(x) est négatif).

Le choix de l'hyperplan est fait à partir du concept de **l'hyperplan à marge maximale**. Nous parlerons alors de **classificateur à marge maximale**. La distance entre chaque observation d'entrainement et un hyperplan donné va être calculée. La plus petite distance correspond à la distance minimale entre les observations et l'hyperplan, cette distance est appelée la marge. L'hyperplan à marge maximale est celui qui maximise la marge, en d'autres termes c'est celui qui a la distance minimale aux exemples d'apprentissage la plus grande. Une fois que l'on a trouvé cet hyperplan à marge maximale, nous avons notre hyperplan séparateur optimal qui va définir la règle de classification. Nous pouvons schématiser cette méthode comme cela :



Mathématiquement, nous devons maximiser M, désignant la marge de l'hyperplan, tel que :

$$y_i(\beta_0 + \beta_1 X_{i,1} + \beta_2 X_{i,2} + \ldots + \beta_p X_{i,p}) > M \qquad ; i = 1, \ldots, n$$

$$sous\ contrainte : \sum_{i=1}^{p} \beta_j^2 = 1$$

L'encadré rouge représente la distance de l'observation X_i à l'hyperplan choisi.

Dans la pratique, il est très rare d'avoir des données qui se séparent linéairement de manière parfaite. De plus, le classificateur à marge maximale peut entraîner un sur-apprentissage (overfitting) quand p est très grand. C'est pour cela que nous allons adapter cette précédente méthode, et accepter des erreurs de classement sur l'apprentissage. Nous allons donc parler du **classificateur à vecteurs de support**.
Ce classificateur va choisir un hyperplan de manière à séparer correctement la plupart des observations d'entrainement mais il peut classer quelques points de façon erronée. Nous allons parler d'**hyperplan à marge souple**. On cherche à maximiser M tel que

$$y_i(\beta_0 + \beta_1 X_{i,1} + \beta_2 X_{i,2} + \ldots + \beta_p X_{i,p}) > M(1 - \varepsilon_i) \quad ; \ i = 1, \ldots, n$$

$$sous\ contrainte : \sum_{i=1}^{p} \beta_j^2 = 1 \ et \sum_{i=1}^{n} \varepsilon_i \leq C\,, \qquad \varepsilon_i \geq 0$$

Les variables ε_i sont appelées les variables ressort (slack en anglais). On a trois cas de classification d'une observation désormais :

- $\varepsilon_i = 0$: elle est du bon côté de l'hyperplan, en dehors de la marge.
- $0 < \varepsilon_i < 1$: elle est du mauvais côté de la frontière de la marge.
- $\varepsilon_i > 1$: elle est mal classée, se situe du mauvais côté de l'hyperplan.

C est un paramètre appelé constante de tolérance (ou paramètre de régularisation) qu'on doit calibrer. Il doit être défini correctement pour obtenir un bon compromis biais-variance. Dans le cas où C est grand, cela veut dire qu'on autorise peu d'observations à être mal classées. Le biais est donc faible et la variance élevée.

A travers ces méthodes de classificateur à marge maximale ou à vecteur de support, nous avons défini le concept de la classification SVM linéaire. Effectivement, cette approche fonctionne de la même façon en choisissant le meilleur hyperplan à marge qui sépare les classes et auquel on a prédéfini un paramètre de tolérance. Néanmoins, il est possible dans la pratique d'avoir des données qui ne sont pas linéairement séparables. Le SVM non linéaire va être possible grâce à une transformation des données initiales et à l'utilisation de l'astuce du noyau.

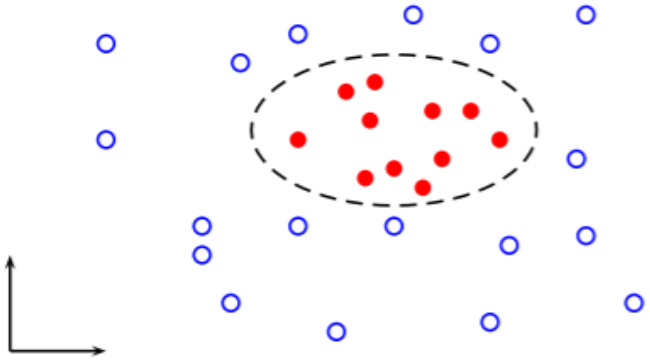


Figure 3 : Exemple graphique de données non séparable linéairement

Nous remarquons que les solutions vues précédemment dépendent essentiellement du produit scalaire entre les observations et ne dépendent pas des observations elles-mêmes. Nous pouvons modifier le classificateur à vecteur de support sous cette forme :

$$f(x) = \beta_0 + \sum_{i=1}^{n} \propto_i < X, X_i > \; ; \; i = 1, \dots, n$$

Notez que pour évaluer la fonction f(x), nous devons calculer le produit scalaire entre le nouveau point X et chacun des points d'apprentissage X_i.

Le SVM non linéaire va être une extension du classificateur à vecteurs de support qui résulte de la transformation de l'espace d'entrée en un espace de très grande dimension, éventuellement infini, muni d'un produit scalaire. On peut parler de l'espace de Hilbert. Ce nouvel espace sera notre espace de représentation. Cette redéfinition va être réalisée à l'aide d'une fonction ϕ, nommée la fonction de représentation. La sortie associée à une entrée x est le vecteur image ϕ(x) de dimension supérieure à l'espace d'origine. A la suite de cette transformation, les données vont devenir linéairement séparables et on envisagera d'appliquer la méthode de classification du SVM.

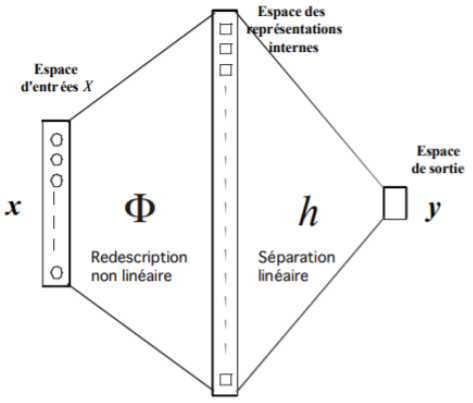


Figure 4 : Représentation graphique du passage par une redéfinition des données

Reprenons l'exemple graphique de données non linéaires montré ci-dessus. Les données sont en bi-dimensions (axe x et y), nous devons passer dans un espace de dimension trois pour réussir à les séparer. Schématiquement, on peut penser qu'on obtiendrait une représentation 3D avec les données centrées par rapport à l'axe z. La partie rouge se placera en haut de l'axe et celle en bleu en bas. Un hyperplan séparateur sera alors envisageable.

Notre problème d'optimisation va être équivalent à celui-ci :

$$\max_{\alpha} \quad \sum_{i=1}^{N} \alpha_i - \sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \langle \phi(x_i), \phi(x_j) \rangle_{\mathcal{H}}$$

$$\text{s.t.} \quad \sum_{i=1}^{N} \alpha_i y_i = 0,$$

$$0 \leqslant \alpha_i \leqslant C, \; 1 \leqslant i \leqslant N.$$

Le produit scalaire <ϕ(x_i), ϕ(x_j)> se calcule difficilement lorsqu'on est dans de grandes dimensions. Néanmoins, il est très important pour la détermination de notre règle de classification du SVM. Nous allons utiliser l'astuce du noyau pour simplifier les calculs dans l'espace de grande dimension, ce qui va permettre d'éviter de calculer ce produit scalaire. Cela consiste à remplacer le produit scalaire par une fonction noyau K : K(x_i, x_j) = <ϕ(x_i), ϕ(x_j)>. La fonction noyau quantifie la similarité de plusieurs observations, elle doit être positive et symétrique. Les fonctions noyaux les plus courantes sont de forme polynomiale, à base radiale ou sigmoïdes. La démarche sera de chercher des fonctions noyaux qui coïncident à un produit scalaire dans un certain espace de grande dimension qui agira comme l'espace de représentation. L'utilisateur de la méthode devra essayer différentes fonctions noyaux et choisir celle pour laquelle il obtient une bonne séparation des données pour utiliser le principe de SVM linéaire.

❖ SVM pour FDA

Après avoir expliqué le concept de la classification SVM, applicable sur des données linéaires et non linéaires, nous pouvons décrire l'utilisation de cette méthode pour analyser des données fonctionnelles. Effectivement, comme nous l'avons vu précédemment, le SVM peut être appliqué à des données à dimension infinie.

Cependant, si l'on applique directement le SVM linéaire aux données fonctionnelles, on remarque une mauvaise performance du classificateur. Cela est dû au fait que la pénalisation (vu précédemment ε_i) va être mal approximée si les courbes sont discrétisées par un très grand nombre de points. Il est alors plus intéressant d'utiliser le SVM non linéaire sur des données fonctionnelles. L'adaptation de la méthode SVM va donc se reposer sur une bonne définition de la **fonction noyau pour les données fonctionnelles**.

Il existe des fonctions noyaux standards applicables à n'importe quel espace d'Hilbert. Par exemple, on retrouve le noyau Gaussien ou le noyau polynomial. Néanmoins, l'utilisation directe de ces noyaux standards sur les données fonctionnelles ne prend pas en compte la nature fonctionnelle de celles-ci. Le classificateur n'est donc pas optimal. Il est très important de prendre en compte cet aspect fonctionnel des données.

L'article nous décrit alors plusieurs techniques pour déterminer les fonctions noyaux adaptées aux données fonctionnelles. La méthode principale consiste à transformer, à l'aide d'un opérateur P, X (les données fonctionnelles dans l'espace d'Hilbert) vers un autre espace D sur lequel un noyau K est défini. Le noyau réel Q sur X a pour formule : $Q(u, v) = K(P(u), P(v))$.

L'une des transformations employées pour avoir des noyaux adaptés est la **projection**. L'idée est de réduire la dimension de l'espace d'entrée. Nous notons V_d un sous-espace d-dimensionnel de X et $\{\psi_j\}$ avec j = 1, …, d la base orthonormée de cet espace. La transformation P_{V_d} est la projection orthogonale sur l'espace V_d :

$$P_{V_d}(x) = \sum_{j=1}^{d} \langle x, \Psi_j \rangle \Psi_j.$$

Nous pouvons déterminer deux solutions envisageables pour la transformation de l'espace d'entrée. Effectivement, le choix de l'espace V_d peut être dirigé dans un premier temps par une expertise des données fonctionnelles, et donc un choix précis sur la base liée à cet espace. On retrouve notamment la base de Fourier, la base d'ondelettes ou la base B-spline qui peuvent avoir des propriétés intéressantes pour un certain type de données. Par exemple, les spectres du proche infrarouge sont lisses en raison des propriétés physiques de la transmission et de la réflexion de la lumière. Pour analyser de telles données, un expert peut choisir un sous-espace spline avec une base B-spline. Celle-ci est reconnue pour son principe de lissage, et peut donc être adaptée aux propriétés des données fonctionnelles de la spectroscopie.

Cette méthode d'expertise va tirer parti de la nature fonctionnelle de la donnée, tout comme la seconde solution. Cette dernière consiste à adapter la fonction noyau de l'espace tout en cherchant les paramètres optimaux à déterminer pour réaliser la méthode de la projection.

Il est vrai que pour construire un classificateur SVM à partir de la méthode de projection, nous devons choisir plusieurs paramètres :

- La dimension du sous-ensemble V_d
- Le paramètre de régularisation C
- Le type de noyau universel K et ses paramètres (sigma pour un noyau gaussien)

Nous allons noter *A* la liste de ces paramètres. Notre objectif est alors de choisir la meilleure liste de paramètres $a \in A$ et donc le meilleur classificateur associé à cet ensemble. Nous allons diviser les entrées en deux ensembles : un ensemble de formation $\{(x_i , y_i), i = 1, \ldots , l_N \}$ et un ensemble de validation $\{(x_i , y_i), i = l_N +1, \ldots , N\}$. On applique le SVM sur l'ensemble de formation et on calcule alors la règle de classification notée f_a. L'ensemble de validation est utilisé pour sélectionner la valeur optimale de *a* déterminée par :

$$a^* = \arg\min_{a \in \mathcal{A}} \widehat{L} f_a + \frac{\lambda_a}{\sqrt{N - l_N}}, \qquad \text{avec} \qquad \widehat{L} f_a = \frac{1}{N - l_N} \sum_{n=l_N+1}^{N} \mathbb{I}_{\{f_a(x_n) \neq y_n\}},$$

Le terme λ_a représente un terme de pénalité pour éviter la sélection des modèles les plus complexes. Le classificateur SVM est alors choisi comme $f_N = f_{a^*}$.

❖ Applications sur des données

Dans un premier temps, nous avons créé des données pour réaliser un SVM. Nous avons donc utilisé des données « classiques » et la fonction **svm** du package « **e1071** » que nous avons étudié en cours.

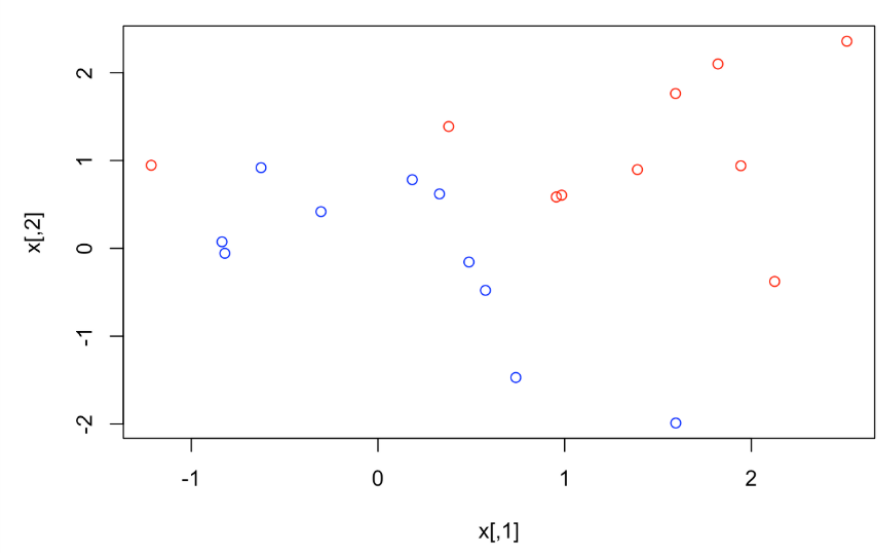
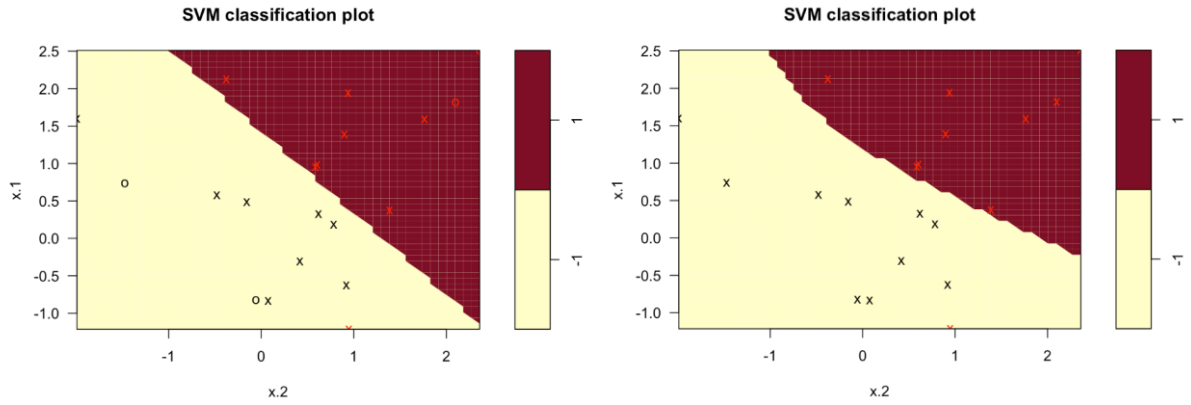


Figure 5 : Aperçu des données que nous avons créées pour réaliser un SVM

En premier, nous avons appliqué une fonction noyau linéaire. Le résultat est visible sur la photo de gauche. Sur celle de droite, nous avons réalisé un SVM avec une fonction radiale.



Nous avons appliqué pour les deux classifications, cost = 1, c'est-à-dire que notre constante de tolérance C vaut 1. Dans les deux cas, l'algorithme identifie bien deux classes, une qui vaut 1 et l'autre qui vaut -1. En affichant la matrice de confusions des deux méthodes, nous avons remarqué un meilleur résultat avec une fonction radiale. Ci-joint, vous trouverez les résultats des matrices : (en colonnes nous avons les prédictions faites par la méthode ; en lignes ce sont les étiquettes théoriques)

	Fonction linéaire		Fonction radiale	
	-1	1	-1	1
-1	8	2	10	0
1	1	9	1	9

Effectivement, avec une fonction radiale, on remarque qu'il n'y a qu'une observation qui est mal classée (le point est prédit avec une étiquette -1 alors qu'il devrait être situé en 1). Alors que dans l'autre application, on retrouve trois observations qui ne sont pas correctement prédites.

Dans un second temps, nous avons utilisé le package « **fda.usc** » de R. Celui-ci nous permet d'avoir accès à des données fonctionnelles et d'utiliser des fonctions telles que **classif.svm** pour réaliser la classification. Nous avons décidé de travailler sur les données appelées « phoneme » (phonème en français). Cette base de données regroupe un échantillon d'apprentissage et un échantillon de test, les deux sont composés de données fonctionnelles. L'échantillon d'apprentissage est constitué de 250 courbes discrétisées chacune en 150 points. Les données de l'échantillon d'apprentissage sont représentées sur la photo suivante.

Chaque courbe va être relié à une classe, nous en trouvons 5 : « sh » (1), « iy » (2), « dcl » (3), « aa » (4) et « ao » (5). Nous distinguons ces groupes sur la photo, chacun caractérisé par une couleur respective (vert, bleu turquoise, bleu foncé, rose, noir).

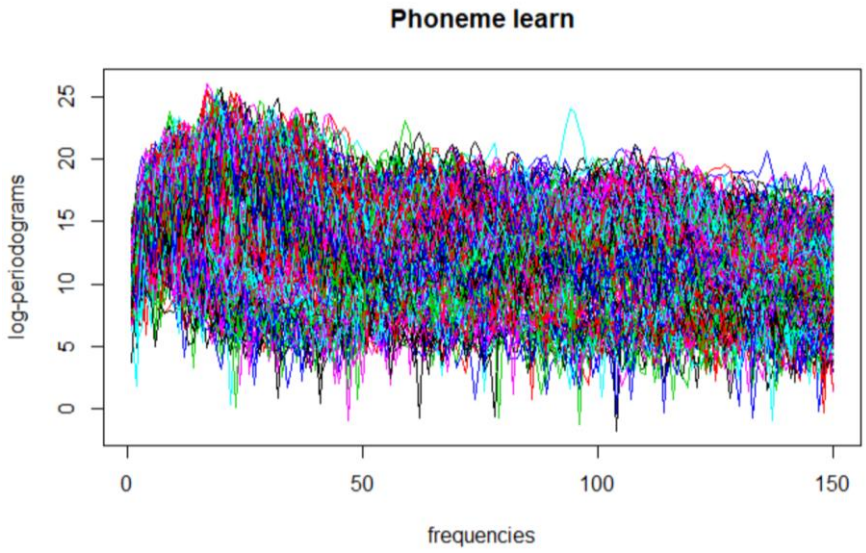


Figure 6 : Représentation des données fonctionnelles Phonème, package fda.usc

Nous avons réalisé différentes classifications SVM en spécifiant plus ou moins les paramètres. Un premier essai est l'application de la méthode SVM sans préciser les paramètres. Nous obtenons les résultats ci-joint :

```
-Probability of correct classification by group (prob.classification):
   1    2    3    4    5
0.98 1.00 0.96 0.80 0.78

-Confusion matrix between the theoretical groups (by rows)
  and estimated groups (by column)

    1  2  3  4  5
  1 49  1  0  0  0
  2  0 50  0  0  0
  3  0  2 48  0  0
  4  0  0  0 40 10
  5  0  0  0 11 39

-Probability of correct classification:  0.904
```

Le premier élément indique le pourcentage de bonne classification pour chaque classe. Nous observons que les courbes appartenant à la seconde classe « iy » sont toute bien classées. La classe 5 « ao » est celle qui obtient le plus bas pourcentage, cela veut dire qu'on arrive mal à identifier les courbes appartenant à ce son. La matrice de confusion indique bien que 11 courbes sont estimées appartenant à « aa » (4) alors qu'elles devraient être identifiées comme « ao » (5). Les erreurs étant principalement entre la classe 4 et 5, leurs pourcentages sont donc proches et le nombre de confusions dans la matrice aussi. Les sons « aa » et « ao » se ressemblent et sont donc difficilement estimés par le classificateur. On peut identifier

quelques erreurs aussi pour la classe 1 et 3 mais elles restent minimes par rapport aux autres. Cependant, dans la globalité nous obtenons un résultat de classification qui est relativement bon 0,904, proche de 1.

Par la suite, nous avons réalisé plusieurs tests sur les différents paramètres pour voir leur effet sur la classification. Le paramètre « Weight » peut être égal à « egal » ou « inverse ». Dans le premier cas, on garde les mêmes poids de pondération pour chaque observation, dans le second on inverse la probabilité de pondération. Ce paramètre peut aussi être un vecteur de taille n où on attribue une valeur pour chaque courbe.
Le paramètre « type » est déterminé par défaut comme « 1vsall ». Cela veut dire que le schéma de probabilité maximale qui est appliqué nécessite des classificateurs binaires G. Si on mets « majoritaire », un système de vote est appliqué, cela nécessite des classificateurs binaires G(G-1)/2.
Enfin le paramètre « laplace » peut être équivalent à une valeur si on souhaite qu'un lissage de Laplace soit effectué sur nos données.

A côté de l'utilisation des fonctions svm prédéfinies dans le logiciel de programmation, nous avons essayé d'utiliser et de comprendre les bases Fourier et B-spline. Nous avons créé pour cela des bases grâce aux fonctions « create.bspline.basis » et « create.fourier.basis ». Elles prennent en paramètre l'intervalle sur lequel les objets de la donnée fonctionnelle (phonème) peuvent être évalués. Nous avons également fait varier le paramètre « nbasis », qui caractérise le nombre de fonctions de base que nous souhaitons. Nous avons fait avec nbasis égal à 5, 51 ou 121. Cette variable doit être obligatoirement un nombre impair.

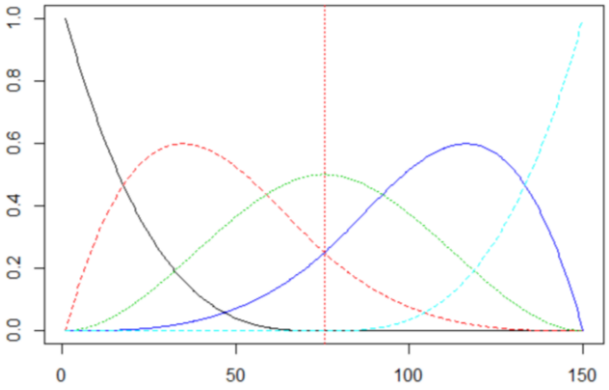


Figure 7 : Représentation d'une base B-spline avec 5 fonctions

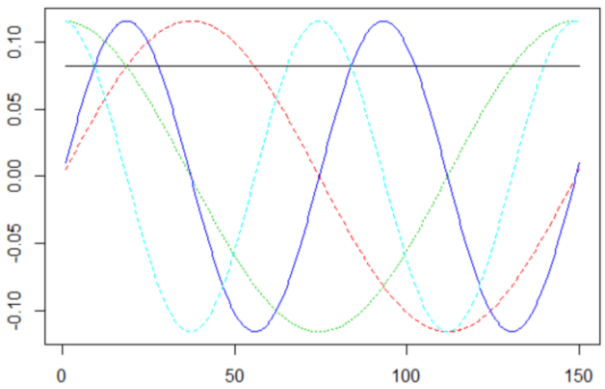


Figure 8 : Représentation d'une base Fourier avec 5 fonctions

Nous avons ensuite lissé une des courbes de la base de données appartenant au son « sh » à l'aide des bases que nous avons créé précédemment (photo ci-dessous). Nous pouvons remarquer que plus le nombre de fonctions pour définir la base est grand plus la représentation va être semblable à la courbe initiale. En analysant les résultats des deux bases pour un même nombre de fonctions, nous pouvons voir que la base Fourier est plus adaptée à nos données fonctionnelles.

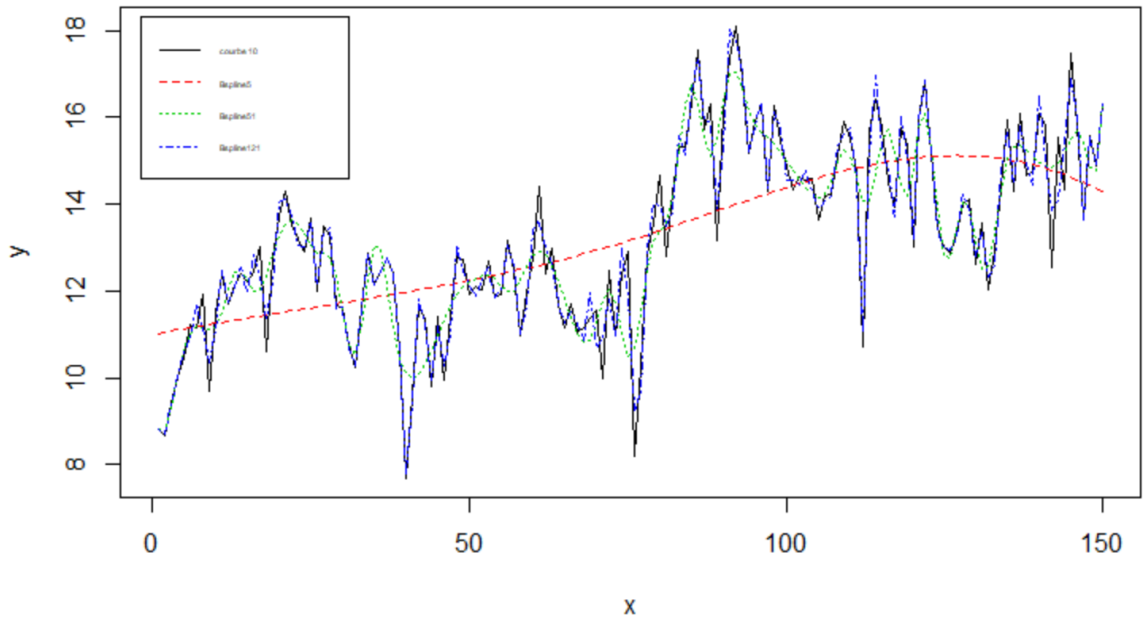


Figure 9 : Représentation du lissage par fonctions splines d'une courbe appartenant au son « sh »

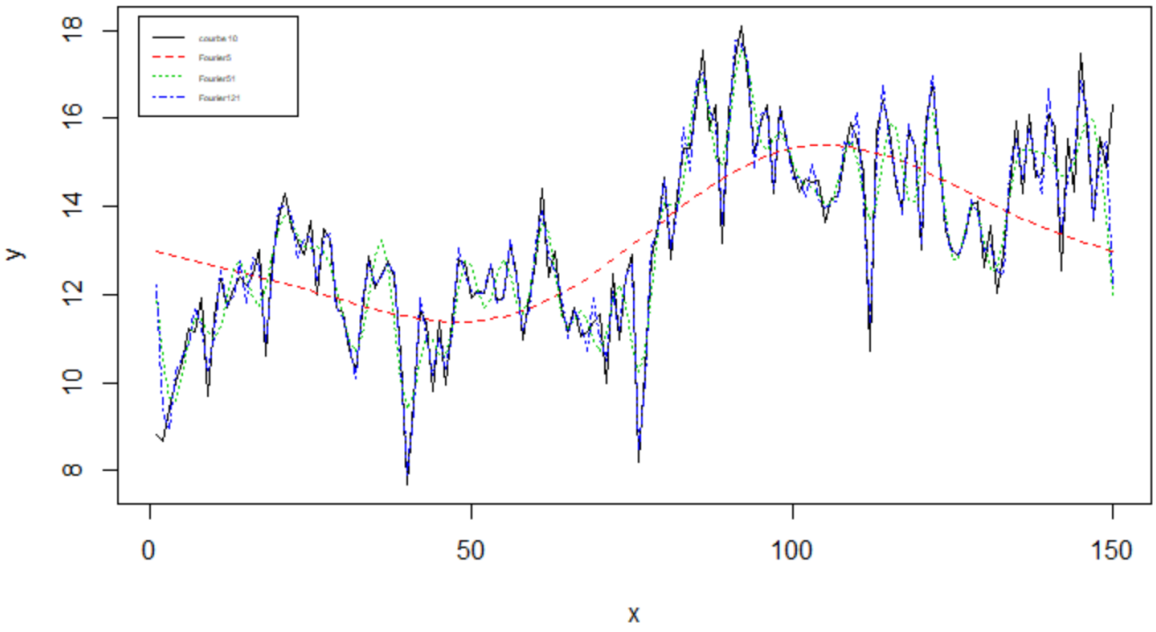


Figure 10 : Représentation du lissage par transformé de Fourier d'une courbe appartenant au son « sh »

En conclusion, l'article scientifique que nous avons étudié nous a permis de voir comment utiliser les SVM pour une classification de données fonctionnelles. Nous avons vu qu'il était plus favorable d'adapter les fonctions noyaux à ces données que d'appliquer directement le SVM linéaire. Pour cela, nous avons donc défini les fonctions noyaux basées sur la méthode de projection mais également les fonctions noyaux choisies avec un point de vue expert sur les données initiales. Enfin, nous avons appliqué la méthode sur R. Nous pouvons dire que cette méthode nous a permis de comprendre la notion de données fonctionnelles et surtout d'apprendre à les analyser en prenant en compte leur aspect fonctionnel, ce qui est très important pour une bonne analyse de ce type de données.

Références :

- Support vector machine for functional data classification Fabrice Rossi, Nathalie Villa
- https://perso.univ-rennes2.fr/system/files/users/fromont_m/Apprentissage.pdf
- https://fr.slideshare.net/tuxette/analyse-de-donnes-fonctionnelles-par-machines-vecteurs-de-support-svm
- https://www.lri.fr/~antoine/Papers/SVM-synthese.pdf
- http://gchagny.perso.math.cnrs.fr/CoursFDA_slides_Chap1.pdf?fbclid=IwAR3UqeICPG-xfrM7-43UNa6X9YHs-w0MQCSJZ9rjHQac2eJ16fsAGPKQQFo
- Statistical Computing in Functional Data Analysis: The R Package fda.usc ; Manuel Febrero-Bande et Manuel Oviedo de la Fuente