École nationale supérieure des Mines de Nancy

Rapport de projet 3A

Pierre Gauthier

Algorithme d'apprentissage en chimie quantique et application au screening (sélection) de cellules photovoltaïques

Laboratoire : Institut Élie Cartan

Tuteurs : Jérémie Unterberg, Marianne Clausel, Dario Rocca

4 Fevrier 2019

Table des matières

Introduction :

Ce projet est conduit dans un cadre pédagogique en tant que projet de troisième année à l'école des Mines de Nancy avec pour tuteurs Marianne Clausel, Dario Rocca et Jérémie Unterberg. Il suit la publication scientifique de Mathias Rupp *Machine Learning for Quantum Mechanics in a Nutshell*. Le but du projet est la reproduction des résultats de cette étude, pour ensuite aller plus loin avec d'autres méthodes. Les codes du prjet ne sont pas en annexe, mais sont disponible sur le dépôt Github suivant : `https://github.com/pierrzacharias/Projet_3A`.

Chapitre 1

Contexte de l'étude

1.1 The Harvard Clean Energie Project

La publication de Mathias Rupp *Machine Learning for Quantum Mechanics in a Nutshell* s'inscrit dans la dynamique des différents travaux menés autour du vaste projet de recherche : *The Harvard Clean Energie Project*. Ce projet vise à concevoir une nouvelle gamme de panneaux solaires révolutionnaires conçus à partir de molécules organiques. Ces panneaux permettraient de s'affranchir des terres rares indispensables actuellement, et d'atteindre de meilleurs rendements énergétiques (15%) et d'être beaucoup plus économiques que les panneaux actuels par la facilité de production).
Toute la problématique du projet est de trouver de nouvelles molécules présentant des propriétés optimales pour une utilisation sur des panneaux solaires. Les équipes de Harvard sont donc à la recherche de la molécule présentant les caractéristiques optimales parmi un set de l'ordre de 3.5 millions de molécules.

Cependant calculer les propriétés des molécules amène à résoudre le problème à N corps, c'est à dire le système d'équation de Schrödinger qui ne comporte pas de solution analytique. Le développement de la théorie de la DFT (*density functional theory*) dans les années 80 pallie à cette difficulté, et permet de calculer les états d'énergies des molécule à partir de l'état fondamental.

Mais le temps de calcul pour étudier toute les molécules reste considérable au vu de leur grand nombre. Le *Clean Energie Project* a notamment en 2011 fait appel à un réseaux de crowd computing : ce sont des réseaux de bénévole mettant à disposition une partie de la puissance de calcul de leur ordinateur pour des projets à grande échelle du monde scientifique nécessitant un temps de calcul hors du communs.

L'article que cherchons à reproduire explore de nouvelles techniques de couplage entre la physique quantique et la prédiction de données pour s'affranchir des limitations de puissance de calculs et pouvoir explorer le maximum de configuration par extrapolation des résultats.

Nous allons ainsi nous appuyer sur un set de molécules dont les énergies d'atomisations ont étés calculés par la théorie de la DFT, et faire de la prédiction pour de nouvelle molécules à partir de ces dernièrse en employant les méthodes d'apprentissage automatique.

1.2 Descripteurs des molécules

Nous allons à présent nous intéresser aux données sur lesquelles nous allons effectuer nos prédictions, et quels descripteurs nous allons employer.

Initialement, nos données sont les coordonnées de chaque atome des molécules, et les énergies d'atomisations de celles-ci.
Nos données sont alors contenues dans un fichier au format .xyz. Chaque élément du fichier correspondant à une molécule est un tableau avec en première ligne le nombre d'atome de la molécule, en deuxième ligne l'énergie d'atomisation et le numéro de la molécule dans le fichier, et chaque ligne correspond ensuite aux coordonnées cartésienne de l'atome dans la molécule. A noter que ces coordonnées sont par rapport à un atome de référence.

Nombre d'atome			
numéro de la molécule	énergie d'atomisation		
atome 1	x(1)	y(1)	z(1)
atome 2	x(2)	y(2)	z(2)
.	.	.	.
.	.	.	.
.	.	.	.
atome n	x(n)	y(n)	z(n)

Par exemple pour la première molécule du fichier CH_4 que l'on traite :

5						
0001	-417.031					
C	1.04168000	-0.05620000	-0.07148000	1.04168200	-0.05620000	-0.07148100
H	2.15109000	-0.05620000	-0.07150000	2.13089400	-0.05620200	-0.07149600
H	0.67187000	0.17923000	-1.09059000	0.67859800	0.17494100	-1.07204400
H	0.67188000	0.70866000	0.64196000	0.67861300	0.69474600	0.62898000
H	0.67188000	-1.05649000	0.23421000	0.67861400	-1.03828500	0.22864100

FIGURE 1.1 – Premier élément du fichier .xyz correspondant à la première molécule. Le bloc de coordonnées à gauche correspond aux coordonnées du champ de force et le bloc de droite correspond aux coordonnées DFT.

Nous allons ensuite mettre en forme ces données avec un descripteur pour effectuer nos prédictions par la suite. Nous utilisons les matrices de Coulomb pour

représenter les molécules. Les matrices de Coulomb sont définies comme suit :

$$M_{ij} = \begin{cases} 0.5Z_i^{2.4} & i = j \\ \dfrac{Z_i Z_j}{||R_i - R_j||_2} & i \neq j \end{cases}$$

avec Z_i le numéro atomique correspondant et R_i la position des atomes.

Dans la pratique, les calculs que nous mettrons en place pour la prédiction nécessiterons un nombre constant de descripteur par molécules, même si celle-ci n'ont pas le même nombre d'atomes et donc des matrice de Coulomb de taille différentes. Nous allons donc remplir pour chaque molécule une matrice de Coulomb de taille égale au nombre maximal d'atome que nous pouvons avoir parmi les molécules que l'on considère dans notre jeu de données, c'est à dire une matrice de taille $23times23$.

Les matrices de Coulomb sont symétriques ce qui fait que nous pouvons garder que la partie inférieure ou supérieure de la matrice pour la prédiction. Nous allons ainsi conserver la partir supérieure de la matrice de Coulomb dans un vecteur de taille $\frac{23\times(23+1)}{2} = 276$.

On veillera dans ce procédé à conserver les termes nuls de la matrice initiale de taille 23×23 pour que les descripteurs identiques puissent être également positionnée dans le vecteur final. On peut voir l'importance de cette remarque sur un court exemple sur deux molécules : CH et CH_2 :

Pour C-H :

	C	H		
C	a_1	b_1	0	0
H	b_1	a_2	0	0
	0	0	0	0
	0	0	0	0

Ce qui donne le vecteur (a_1, b_1, 0, 0, a_2, 0, 0, 0, 0, 0)

Pour H-C-H on retrouve les même termes :

	C	H	H	
C	a_1	b_1	c_1	0
H	b_1	a_2	c_2	0
H	c_1	c_2	a_3	0
	0	0	0	0

Ce qui donne le vecteur (a_1, b_1, c_1, 0, b_1, a_2, c_2,0 ,a_3, 0)

On voit dans cet exemple l'importance de conserver les termes nuls lors du remplissage du vecteur, pour que les termes égaux correspondants à des atomes dans des situations semblables soient positionnés à la même place, notamment les termes diagonaux qui ne dépendent que de l'atome lui-même. Sinon les modèles que nous mettrons en place pour la prédiction seront faussés.

De plus comme nous l'avons énoncé ,les coordonnées des atomes sont déterminées par rapport à un atome de référence ce qui fait que les matrices de coulomb ne sont pas insensibles aux inversions de ligne pour les modèle de régression. Pour que toute les matrices soient équivalentes de ce point de vue la, nous trions les lignes de la matrice par norme descendante, en inversant les lignes correspondantes pour conserver la symétrie de la matrice avant de prendre la partie diagonale supérieure.

Sur le Github `https://github.com/pierrzacharias/Projet_3A` la mise en forme des données telle que décrite dans cette section se trouve dans le fichier de code *ecriture_Matrice_de_Coulomb.py* dans le dossier *Ecriture_des_Matrices_de_Coulomb/*. Elle sont encodés en sortie dans le fichier *matrice_coulomb.txt*

Chapitre 2

Modèle de prédiction : Machine learning

Nous allons à présent introduire toute la théorie des outils d'apprentissage automatique que nous mettrons en œuvre pour la prédiction.

2.1 Support Vector Machines methods (SVM)

Nous considérons les données : $(x_i,\ y_i)_{1\leqslant i\leqslant N},\ x_i \in \mathbb{R},\ y_i \in \{-1,\ 1\}$.

Le probléme SVM vise à séparer les données en deux classes $+1$ et -1 à l'aide de la fonction $f(x) = w \cdot x + b$ ($b \in \mathbb{R},\ w \in \mathbb{R}^d$) telle que :

$$f(x) > 0 \Rightarrow x \in C_{+1}$$
$$f(x) < 0 \Rightarrow x \in C_{-1}$$

Nous voulons trouver l'hyperplan qui sépare le mieux nos données parmi tous ceux compatibles.

Pour juger la qualité d'un hyperplan en tant que séparateur on utilise la distance entre les données et l'hyperplan. Plus précisément, la « marge » d'un problème d'apprentissage est définie comme la distance entre l'hyperplan de séparation et l'individu le plus proche .
Pour un hyperplan $H = \{x \mid w^T x + b = 0\}$, on a :

$$\text{Marge}(H) = \min_{x_i}\ d(x_i,\ H)$$

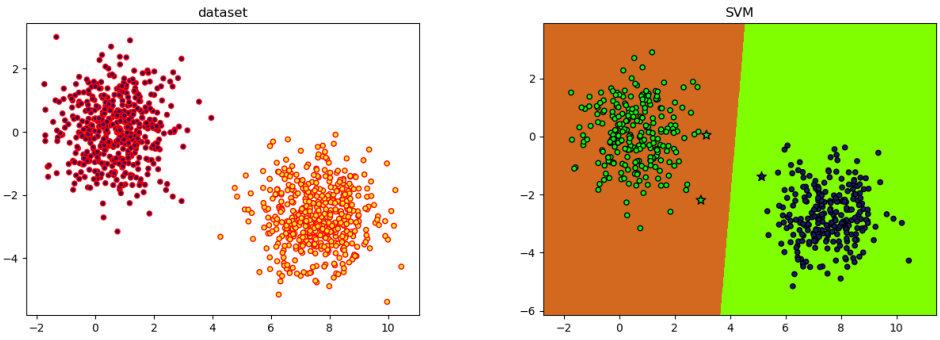

FIGURE 2.1 – séparation de données générées make_blobs du package dataset et séparation de l'espace en deux classes par la méthode des vecteurs support à l'aide de la fonction SVC du package sklearn. Les étoiles sont les vecteurs supports.

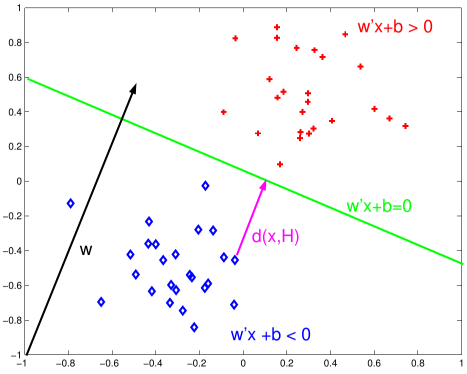

FIGURE 2.2 – Illustration de la distance à l'Hyperplan H.[Cours Cnam RCP209]

On définit alors les vecteurs supports comme les éléments les plus proches de part et d'autre de l'hyperplan de séparation. On voit intuitivement qu'ils déterminent la marge (Figure 2.4)

On peut ainsi exprimer la marge en fonction des vecteurs supports x_{vs} :

$$2 \times \text{Marge} = 2 \times d(x,\ H) = \frac{|w^T x_{vs} + b|}{||w||}$$

Nous prendrons dans la suite la quantité 2×Marge car cela ne change pas le problème de minimisation et simplifie les expressions. On impose pour les vecteur

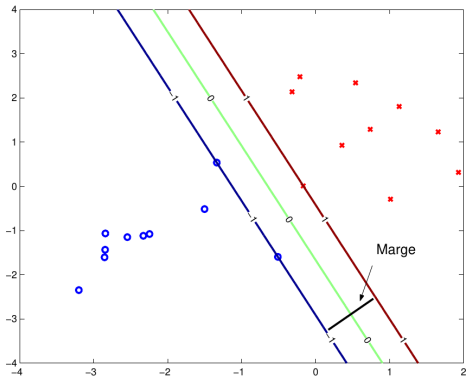

FIGURE 2.3 – Le meilleur hyperplan parmi tous ceux compatible est, comme on le voit ici, celui qui passe au « milieu »des données, et donc celui qui maximise la marge. C'est le séparateur de marge maximale.[Cours Cnam RCP209]

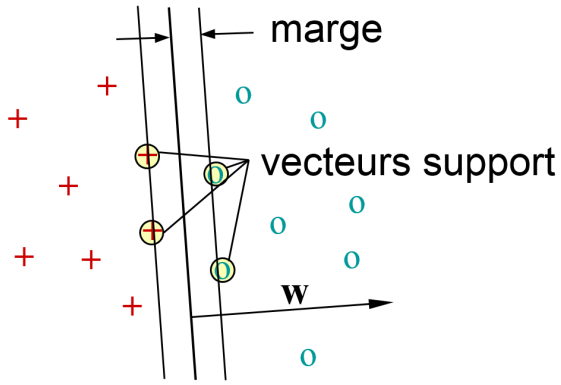

FIGURE 2.4 – Illustration de l'hyperplan à marge maximale et des vecteur supports associés.[Cours Cnam RCP209]

supports $|w^T x_{vs} + b| = 1$. La marge devient donc :

$$\text{Marge} = \frac{2}{||w||}$$

Sous l'hypothèse qu'il existe un hyperplan qui sépare nos données, trouver l'hyperplan qui maximise la marge revient à résoudre le problème suivant :

$$\begin{cases} \arg\min_{w,b} \dfrac{1}{2}||w||^2 \\ \forall\ 1 \leqslant i \leqslant N,\ y_i(w \cdot x_i + b) \geqslant 1 \end{cases}$$

On utilise le Lagrangien des conditions de Karush, Kuhn et Tucker, qui s'exprime sous la forme suivante :

$$L(w, b, \lambda_i) = \frac{1}{2}||w||^2 - \sum \lambda_i (y_i(w \cdot x_i + b) - 1)$$

Le problème de minimisation trouve sa solution pour les paramètres (w, b, λ) tels que :

$$\begin{cases} \dfrac{\partial L}{\partial b}(w^*, b^*, \lambda^*) = 0 \\ \dfrac{\partial L}{\partial w}(w^*, b^*, \lambda^*) = 0 \end{cases}$$

Ce qui amène à :

$$\begin{cases} \displaystyle\sum_{i=1}^{n} \lambda_i^* y_i = 0 \\ \displaystyle\sum_{i=1}^{n} \lambda_i^* y_i x_i = w^* \end{cases}$$

Ce qui permet d'obtenir l'expression du problème dual :

$$\begin{cases} \max\ L(\lambda) = \sum_i \lambda_i - \frac{1}{2} \sum_i \sum_j \lambda_i \lambda_j y_i y_j x_i \cdot x_j \\ \lambda_i \geqslant 0 \\ \sum_i \lambda_i y_i = 0 \end{cases}$$

avec le paramètre b^* que nous obtenons avec la relation $|x_{vs}^T w^* + b^*| = 1$ que nous avons fixée précédemment. La fonction de décision correspondant à la solution du problème de maximisation de la marge est :

$$f^*(x) = \sum_{i=1}^{n} \lambda_i^* y_i x_i^T x + b^*$$

2.2 Données non-linéairement séparables : Astuce du noyau.

Dans la section précédente nous avons considéré des données linéairement séparables, nous allons à présent nous intéresser au cas où il n'existe pas d'hyperplan qui puissent séparer nos données. L'idée principale de cette section est de pour pouvoir séparer nos données, nous passons dans un espace de dimension supérieure tel que nos données deviennent linéairement séparables comme cela est illustré sur la figure 2.6.

Pour ce faire nous allons remplacer le produit scalaire $x^T x$ dans l'expression de f par une fonction dite « noyau » : $K : \xi \times \xi \to \mathbb{R}$.
On donne quelques exemple de noyau courant en figure 2.6. Logiquement, comme nous cherchons à séparer les données, les noyaux prennent des valeurs importante pour des données proches.

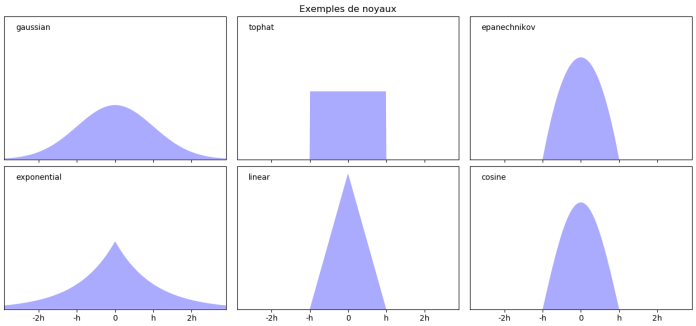

FIGURE 2.5 – Affichage de la densité pour des noyaux courant [Documentation scikit-learn]

Les fonctions noyaux K doivent vérifier les conditions de Mercer :

- K est continue symétrique
- $K(x_i, x_j)_{\ 1 \leqslant i,j \leqslant N}$ est une matrice définie positive

Le théorème de Mercer nous assure ainsi de l'existence d'un espace de Hilbert H et d'une fonction $\phi : \xi \to H$ telle que $K(x, y) = \langle \phi(x), \phi(y) \rangle$.

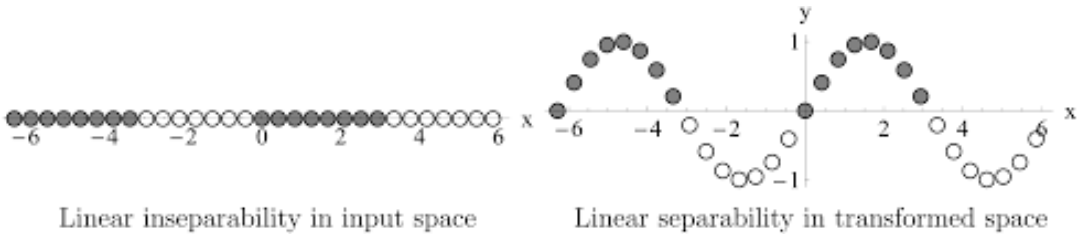

FIGURE 2.6 – Astuce du noyau : projeter les données dans un espace de dimension plus grande, où elles deviennent séparables linéairement.[Mathias Rupp•]

L'estimateur devient ainsi en effectuant le produit scalaire sur l'ensemble des

données $(x_i)_{1 \leqslant i \leqslant N}$:

$$f(\widetilde{x}) = \sum_{i=1}^{n} \alpha_i \mathrm{K}(x_i, \widetilde{x})$$

L'intérêt de la méthode est que l'on change d'espace par la fonction ϕ, mais que l'on a en pratique pas à calculer la fonction ϕ : on calcule directement la fonction K.

2.3 Variable d'écart

Nous allons à présent autoriser que certains points soient mal classés par l'hyperplan pour simplifier l'hyperplan de séparation et éviter des problème de sur-apprentissage.

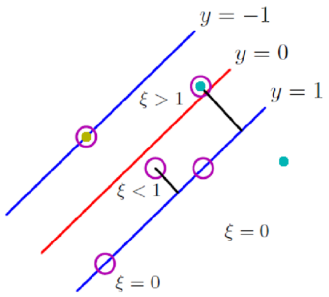

FIGURE 2.7 – [C. Bishop, Pattern Recognition and Machine Learning, Springer 2006]

Nous définissons les variables d'écart ξ_i comme suit :

- $\xi_i = 0$ si x_i est bien classé
- $\xi_i = |y_i - f(x_i)|$ si x_i est mal classé

Le problème de maximisation de la marge devient, en introduisant la constante de régularisation $C > 0$ qui pénalise d'autant plus l'erreur quelle est grande :

$$\begin{cases} \arg\min_{w,b} \dfrac{1}{2}||w||^2 + C \sum\limits_i \xi_i \\ \forall\ 1 \leqslant i \leqslant N,\ y_i(w \cdot \phi(x_i) + b) \geqslant 1 - \xi_i \\ \forall\ 1 \leqslant i \leqslant N, \xi_i \geqslant 0 \end{cases}$$

Ce qui nous amène de la même manière au problème dual :

$$\begin{cases} \max L(\lambda) = \sum_i \lambda_i - \frac{1}{2} \sum_i \sum_j \lambda_i \lambda_j y_i y_j K(x_i, x_j) \\ \forall\ 1 \leqslant i \leqslant N, 0 \leqslant \lambda_i < C \\ \forall\ 1 \leqslant i \leqslant N, \sum_i \lambda_i y_i = 0 \end{cases}$$

2.4 Application des méthode de noyaux à la Ridge Regression

La régression linéaire classique d'observations $(x_i,\ y_i)_{1 \leqslant i \leqslant N}$ par une fonction $f(\widetilde{x}) = \langle w, \widetilde{x} \rangle$ correspond au problème de minimisation :

$$\arg\min_{w \in \mathbb{R}^n} \sum (f(x_i) - y_i)^2$$

La regression ridge consiste à ajouter une fonction de coût avec une norme $|.|_{L^2}$ à cette erreur avec un paramètre $\lambda \geqslant 0$:

$$\arg\min_{w \in \mathbb{R}^n} \sum (f(x_i) - y_i)^2 + \lambda ||w||_2^2$$

L'ajout de cette pénalisation dans une régression permet par la réduction des coefficients de palier à des descripteur corrélés. Cela permet également en introduisant un biais de réduire la variance, et ainsi de réduire également l'erreur totale du modèle comme somme du biais et de la variance.

En reprenant les parties précédentes avec l'astuce du noyau la fonction f devient $f(\widetilde{x}) = \sum_{i=1}^n \alpha_i \mathrm{K}(x_i, \widetilde{x})$,et le problème de minimisation devient

$$\arg\min_{\alpha \in \mathbb{R}^n} \sum (f(x_i) - y_i)^2 + \lambda ||f||_H^2$$

avec H l'espace de Hilbert explicité dans la partie 2.2.
On peut ainsi écrire matriciellement ce problème :

$$\arg\min_{\alpha \in \mathbb{R}^n} < K\alpha - y, K\alpha - y > + \lambda \alpha^T K \alpha$$

où $K \in \mathbb{R}^{n \times n}$ est la matrice du noyau $\quad K_{i,j} = K(x_i, x_j)$.
On trouve directement une solution analytique en prenant le gradient à 0 :

$$\alpha = (K + \lambda I)^{-1} y, \quad \text{avec } I \text{ la matrice identité}$$

On peut donc directement exprimer les coefficients α_i à partir du coefficient de pénalisation λ et c'est ce que nous faisons dans la suite.

Chapitre 3

Experimentations

3.1 Évaluations des modèles

Nous allons dans cette partie expliciter les outils que nous allons utiliser pour évaluer les performances des modèles en comparant les prédictions $f(x_i)$ et les données y_i. Nous utilisons des fonctions de coût qui représentent l'erreur du modèle, on cherche à les avoir donc les plus petites possible. Nous pouvons d'abord définir la somme des résidus au carré RSS (residual sum of squares) :

$$\text{RSS} = \sum_{i=}^{n} (y_i - f(x_i))^2$$

Le premier outil est la racine de la moyenne des sommes des résidus RMSE (root mean squared error) :

$$\text{RMSE} = \sqrt{\frac{1}{n} \sum_{i=}^{n} (y_i - f(x_i))^2}$$

Nous utilisons aussi la moyenne des erreurs absolues MAE (mean absolute error)

$$\text{MAE} = \frac{1}{n} \sum_{i=}^{n} |y_i - f(x_i)|$$

L'intérêt d'utiliser deux fonctions de coût différentes et qu'elles ne donnent pas le même poids au résidus, ici l'erreur RMSE par la mise au carré va pénaliser encore plus fortement les résidus importants.

Nous utilisons également le coefficient de corrélation R^2 définis par

$$(1 - R^2) \sum_{i=1}^{n} (\overline{y} - y_i)^2$$

où $\overline{y}$ est la moyenne des observation y.

Nous pouvons interpréter R^2 comme la proportion de la variance de nos données qui sont expliqués par le modèle. Plus R^2 est proche de 1, plus le modèle est explicatif.

3.2 noyaux pour la SVM

Nous explicitons les différents noyaux K que nous allons utiliser dans nos modélisations.

Noyau Gaussien Nous définissons le noyau gaussien définis par :

$$K(x,z) = \exp -\frac{||x-z||_2^2}{2\sigma^2} \quad \sigma \geqslant 0$$

Ce noyau comporte un hyperparamètre σ à déterminer.

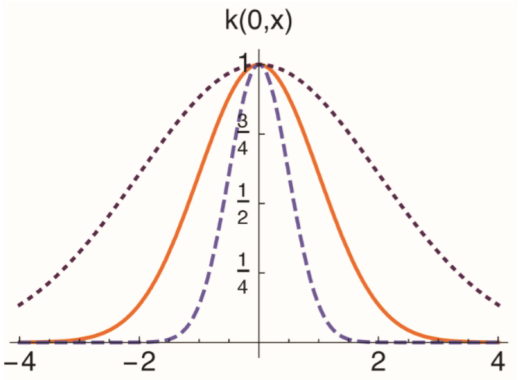

FIGURE 3.1 – noyau gaussien avec $\sigma = 0.5$ $(- -)$, 1 $(—)$, 2 $(\cdots)$ [Mathias Rupp *Machine Learning for Quantum Mechanics in a Nutshell*]

Noyau linéaire Nous définissons le noyau linéaire par :

$$K(x,z) = \langle x, z \rangle$$

Ce noyau ne fait pas subir de changement de dimensions, il fait comme si nous n'appliquions pas la méthode du noyau.

Noyau Lagrangien Nous définissons le noyau gaussien définis par :

$$K(x,z) = \exp -\frac{||x-z||_1}{\sigma} \quad \sigma \geqslant 0$$

Ce noyau comporte un hyperparamètre σ à déterminer.

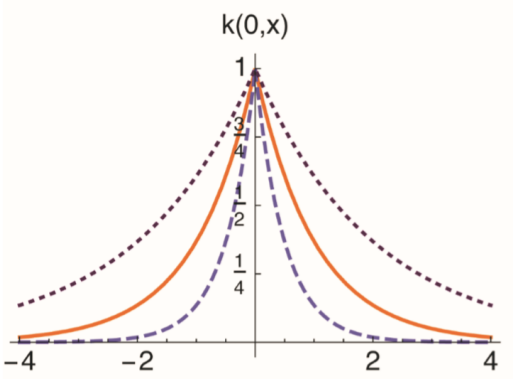

FIGURE 3.2 – noyau laplacien avec $\sigma = 0.5$ $(- -)$, 1 $(—)$, 2 $(\cdots)$ [Mathias Rupp *Machine Learning for Quantum Mechanics in a Nutshell*]

Hyperparamètres Nous voyons que nous avons globalement deux hyperparmètres à déterminer dans les modèle utilisant la kernel regression ridge :

1. λ de la ridge régression
2. σ du noyau gaussien

Nous n'utilisons pas la marge permissive avec une pénalisation ajustable décrite dans la partie 2.3.

Nous allons rechercher le meilleur modèle pour prédire les énergies d'atomisations des molécules ce qui revient à déterminer le meilleur couple d'hyperparamètres (λ, σ) sur un ensemble Ω en utilisant les outils d'estimation de l'erreur d'écrits dans la section 3.1.

3.3 bases d'entraînement et de test des modèles

Nous allons détailler notre manière de procéder visant à obtenir le meilleur modèle de régression pour prédire l'énergie d'atomisation des molécules.

Tout d'abord nous allons séparer notre dataset en une base d'apprentissage sur laquelle entrainer nos modèles, et une base de test où nous évaluerons les performances des modèle avec les outils décrits section 3.1.

Notre dataset contient un peu plus de 7000 molécules, nous le partitionnons en un training set de 1000 molécules, et un testing set contenant le reste des molécules.

Nous décomposons encore cette base d'entrainement training set de taille 1000 en une base de taille 900 pour le choix des hyperparamètres et une base de taille 100 sur laquelle nous allons vérifier la pertinence du choix de nos hyperparamètres avant de passer au modèle global, pour se prémunir contre des problème de sur-apprentissage.

Après sélections de nos hyperparamètres, nous entrainons le modèle final sur l'ensemble du training set de 1000 molécules et nous confrontons nos prédictions aux molécules du testing set.

Par ailleurs si nous nous intéressons à la répartition du nombre d'atome qui ne sont pas des atomes d'hydrogènes par molécules dans notre dataset, nous pouvons voir sur la Figure 3.3 que cette répartition est trés inégale. En pratique nous prendrons toutes les molécules qui comprennent moins de 5 non-H atomes dans la base d'apprentissage training set car leur nombre est trés réduit (il y en a 59), même si cela augmente l'erreur de notre modèle car nos prédictions se font alors uniquement sur des molécules comportant plus de 5 atomes qui ne sont pas des atomes d'hydrogènes.

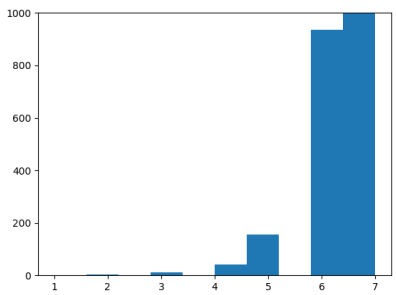

FIGURE 3.3 – Histogramme du nombre d'atome non-H par molécule

3.4 Résultats

Sur le Github `https://github.com/pierrzacharias/Projet_3A`, les simulations effectuées dans cette section sont réalisées dans *implementation_KKR.py* dans le fichier *Methode_Kernel_Ridge_Regression/*.
Les résultats des simulations sont stockés dans des fichiers texte tels que *Resultat_RMSE_RBF.txt* (pour l'erreur RMSE avec le noyau gaussien), et la mise en forme des données pour l'affichage et réalisé dans *lecture_donnee.py*.

Résultats pour le noyau gaussien On affiche les résultats des calculs des erreurs sur une grille $(\lambda, \gamma) = [-30, -10]^2$.
On fait la distinction entre σ utilisé dans l'article, et γ utilisé dans le code avec $\gamma = \frac{1}{2\sigma^2}$, ce qui fait que si on compare les graphiques d'erreurs avec ceux de l'article de M. Rupp, les erreurs sont inversées sur l'axe des abscisses.

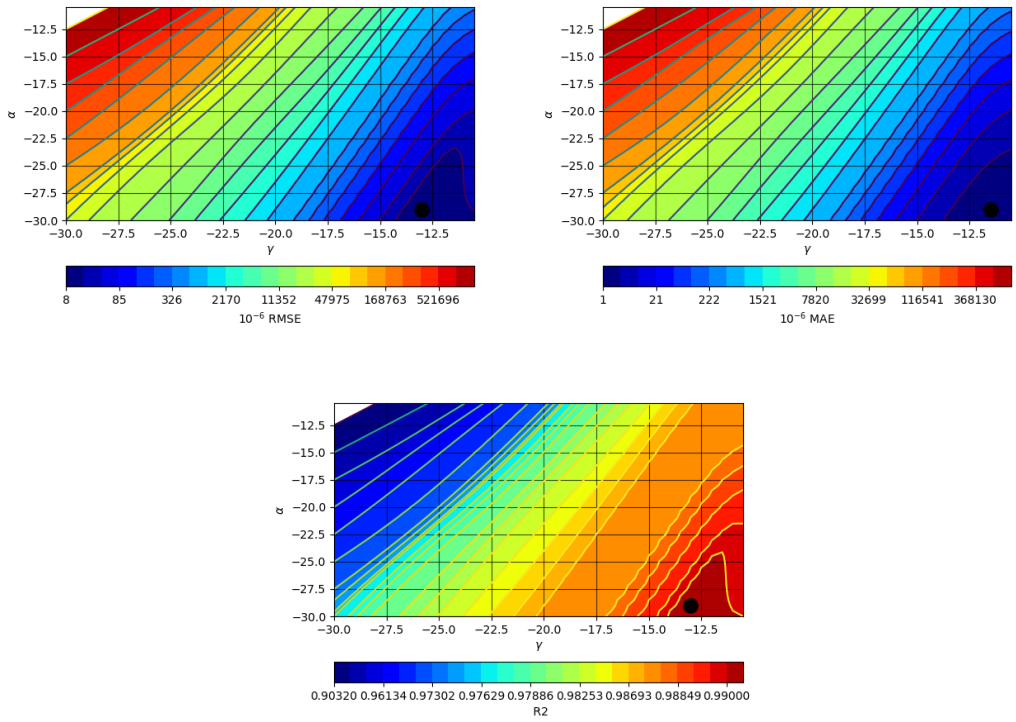

FIGURE 3.4 – Résultat des modèles pour un noyau gaussien, les points noirs correspondent au minimum de l'erreur

Résultats pour le noyau Laplacien : De même sur une grille $[-40, -20] \times [-30, -5]$, avec $\gamma = \frac{1}{\sigma}$ Les séparation pour le R^2 sont mal placés, cela est du au

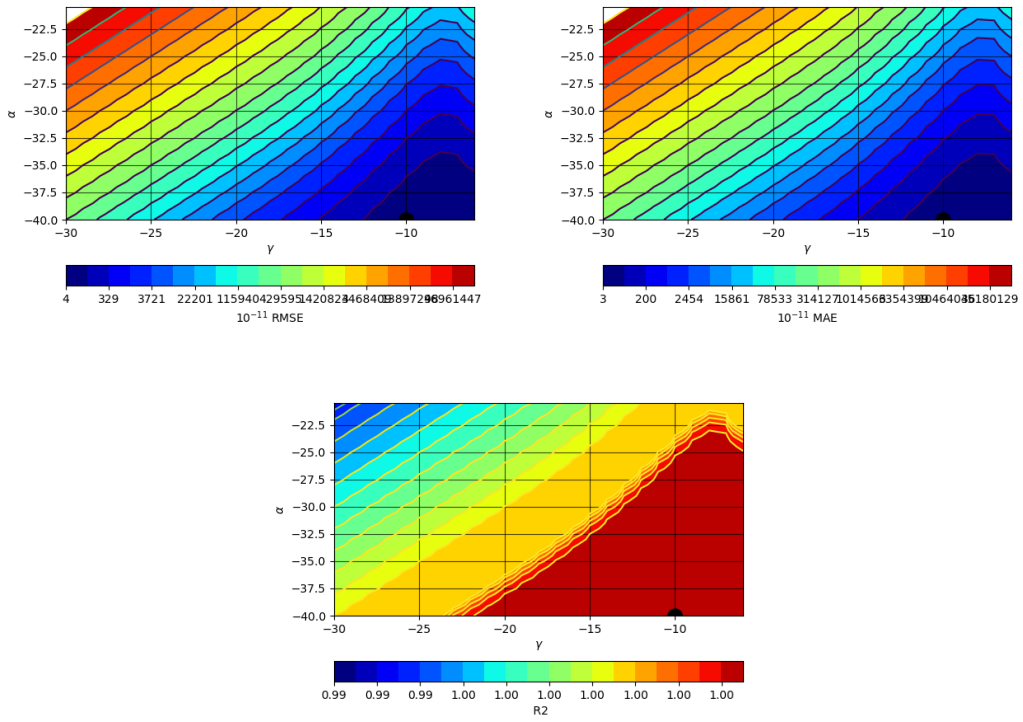

FIGURE 3.5 – Résultat des modèles pour un noyau laplacien, les point noir correspondent au minimum de l'erreur

fait qu'il converge très vite vers la valeur 1.

Synthèse des résultats Nous regroupons les résultats des erreurs pour la prédiction sur le jeu de données complet :

Erreur utilisée	RMSE	MAE	R^2
KRR avec noyau gaussien	9.6371	7.6962	0.9977
KRR avec noyau laplacien	5.4019	3.4555	0.9993

Je ne présente pas mes résultats pour le noyau linéaire car ceux-ci ne présentent pas d'intérêt particulier par-rapport aux méthodes à noyaux classiques.

En comparant avec la publication de Mathias Rupp, nous observons des résultats d'erreurs légèrement meilleurs sur la prédiction sur l'ensemble du jeux de données.

Nous pouvons afficher les prédictions par rapport aux énergie d'atomisation calculées :

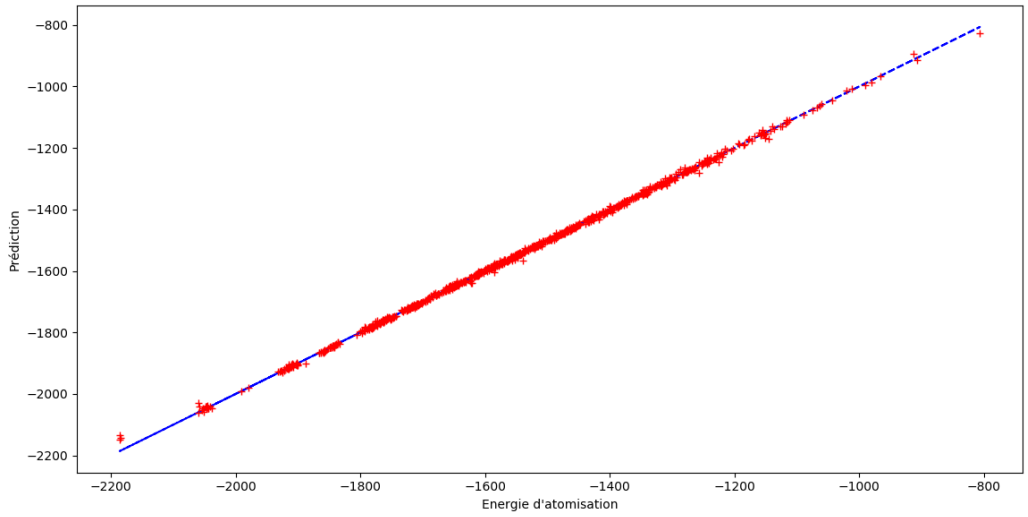

FIGURE 3.6 – Affichage des prédiction par rapport aux énergie d'atomisations, et de la courbe $y = x$

Nous constatons que les prédictions restent très près de la courbe $y = x$, ce qui nous confirme dans la pertinence du modèle que nous avons développé.

Chapitre 4

Perspectives

Dans ce chapitre nous explorons différentes perceptives du projet que nous n'avons pas mit en oeuvre.

4.1 Random Forest

Après avoir reproduit les résultats escomptés, nous aurions pu concevoir des modèles de prédiction différents des méthodes à noyaux et notamment utiliser des Random Forest. J'ai effectué une recherche de modèle utilisant des Random Forest trop superficielle pour que je la présente ici. Les principaux paramètres à considérer dans ces modèle sont la taille des arbres et le nombre de nœuds. Les modèle que j'ai construis m'ont permis d'atteindre une erreur RMSE avec la prédiction sur le jeu de données complet de 14 pour 90 arbres avec 25 nœuds. Cela est supérieur aux erreurs que nous avons obtenus avec les méthodes à noyaux, mais cela reste un modèle assez grossier et nous pouvons probablement obtenir de meilleurs erreurs en optimisant les paramètres du modèle.

4.2 Descripteurs Chimique

Un autre axe de développement du projet afin d'améliorer la prédiction est d'employer de nouveaux descripteurs. En effet les matrice de Coulombs présentent l'avantage de pouvoir reconstituer les molécules, mais ne caractérise pas les groupes chimiques caractéristiques comme par exemple -OH qui identifient un comportement particulier , ce qui va donc influencer les niveaux d'énergies. Typiquement, les chaîne carbonées sont classées en : méthane CH_4,butane C_4H_10,éthylène C_2H_4, benzène C_6H_6, éthanol C_2H_6O, et choline $C_5NH_{14}O$.

Conclusion :

Nous pouvons conclure que ce projet à réalisé ces objectif initiaux de mettre en œuvre les méthodes de régression à noyaux pour reproduire les résultats de la publication scientifique de Mathias Rupp.
Cependant nous ne somme pas allé plus loin que l'article en explorant de nouvelles méthodes d'apprentissage automatique comme les Random Forests.

Bibliographie

[1] M. Rupp. Int. J. Quantum Chem. 2015, 115, 1058–1073. DOI : 10.1002/qua.24954

[2]Cours du Cnam Paris (code Cnam RCP209) *Apprentissage, réseaux de neurones et modèles graphiques* disponible en libre accès sur `cedric.cnam.fr/vertigo/Cours/ml2/`

[3]T. Hastie, R. Tibshirani, J. Friedman, The Elements of Statistical Learning. Data Mining, Inference, and Prediction, 2nd ed., Springer : New York, 2009.