



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

21 Novembre 2018

Table des matières

1	Contexte de l'étude	3
1.1	The Harvard Clean Energie Project	3
1.2	Descripteurs des molécules	5
2	Modèle de prédiction : Machine learning	8
2.1	Support Vector Machines methods (SVM)	8
2.2	Données non-linéairement séparables : Astuce du noyaux.	11
2.3	Variable d'écart	13
2.4	Application des méthode de noyaux à la Ridge Regression	14
3	Experimentations	15
3.1	Évaluations des modèles	15
3.2	noyaux pour la SVM	16
3.3	bases d'entraînement et de test des modèles	17
3.4	Résultats	19

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 pouvoir proposer des variations dans le traitement et étudier l'influence sur le résultat à travers les différents indicateurs que nous définirons.

Les codes sont en annexe, mais sont aussi 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 d'un vaste projet de recherche : *The Harvard Clean Energie Project*. Ce projet vise à concevoir une nouvelle gamme de panneaux solaires révolutionnaire conçus à partir de molécules organiques. Ces panneaux permettrait de s'affranchir des terres rares indispensable actuellement, et d'atteindre de meilleur rendement énergétique (15% selon une source, et d'être beaucoup plus économique 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 solutions analytiques. 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'énergie des molécules à partir de l'état fondamental.

Mais le temps de calcul pour étudier toutes les molécules reste considérable au vu de leur grand nombre. Le *Clean Energie Project* a notamment en 2011 fait appel à un réseau de crowd computing : ce sont des réseaux de bénévoles 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 commun.

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

Nous allons ainsi nous appuyer sur un set de molécules dont les énergies ont été calculées par la théorie de la DFT, et faire de la prédiction pour de nouvelles molécules à partir de ces dernières en employant la méthode de machine learning.

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 de la molécule, et l'énergie d'atomisation de celle-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 pour chaque ligne ensuite correspond aux coordonnées cartésiennes 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 CH₄ 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 :

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

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

Dans la pratique, les calculs que nous allons mettre en place pour la prédiction nécessite 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 variante. 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 ans notre jeu de données, c'est à dire un 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 CH₂ :

		C	H		
Pour 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 (a₁, b₁, 0, 0, a₂, 0, 0, 0, 0, 0, 0)

		C	H		
Pour H-C-H on retrouve les même termes :	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 (a₁, b₁, c₁, 0, b₁, a₂, c₂, 0, a₃, 0, 0)

On voit dans cet exemple l'importance de conserver les termes nuls lors du remplissage du vecteur, pour que les termes égaux correspondant à des atomes dans des situations semblables soient positionnée à la même place, notamment les termes diagonaux qui ne dépendent que de l'atome lui-même. Sinon les modèle 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 équivalents de ce point de vue la, nous trions les lignes de la matrice par norme descendante, en inversant les lignes correspondante 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

2.1 Support Vector Machines methods (SVM)

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

Le problème 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 :

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

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

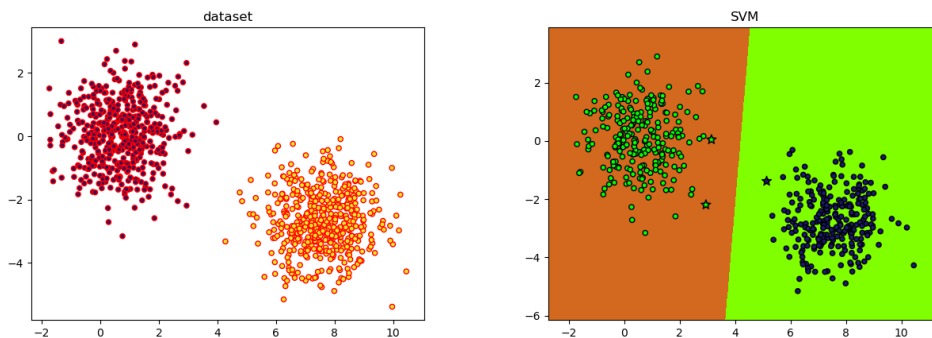


FIGURE 2.1 – sparation de données générées make_blobs du package dataset et séparation de l'espace en deux classe par la méthode des vecteurs support à l'aide de la fonction SVC du package sklearn. Les étoiles sont les vecteurs supports.

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 :

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

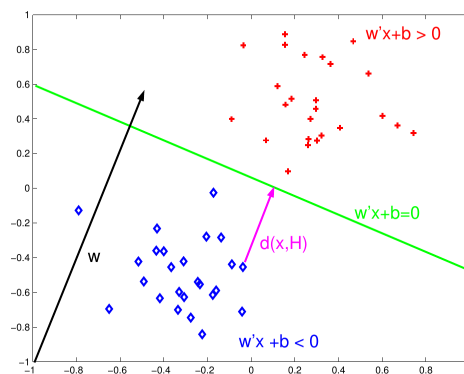


FIGURE 2.2 – Illustration de la distance à l'Hyperplan H.[Cours Cnam RCP209]

On définit alors les vecteurs supports comme les éléments les plus proches de part et d'autre de l'hyperplan de séparation qui 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 $2 \times \text{Marge}$ car cela ne change pas le problème de minimisation et simplifie les expressions. On impose pour les vecteurs supports $|w^T x_{vs} + b| = 1$. La marge devient donc :

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

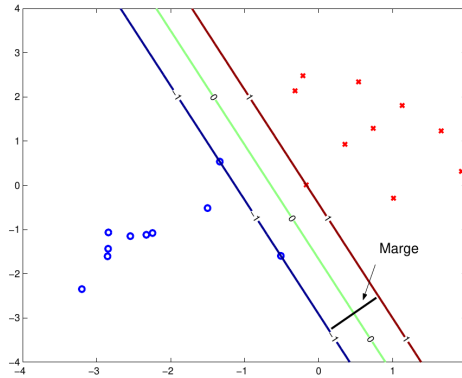


FIGURE 2.3 – Le meilleur hyperplan parmi tous ceux compatibles est, comme on le voit ici, celui qui passe au « milieu » des données, et donc celui qui maximise la marge. C'est le séparateur de marge maximale.[Cours Cnam RCP209]

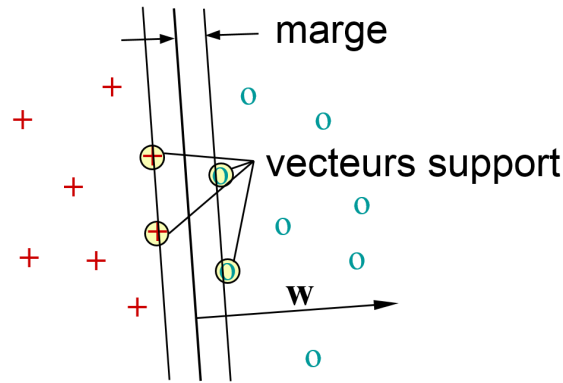


FIGURE 2.4 – Le meilleur hyperplan parmi tous ceux compatibles est, comme on le voit ici, celui qui passe au « milieu » des données, et donc celui qui maximise la marge. C'est le séparateur de marge maximale.[Cours Cnam RCP209]

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 :

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

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 :

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

Ce qui amène à :

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

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

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

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 noyaux.

Dans la section précédente nous avons considéré des données linéairement séparable, 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 pour pouvoir séparer nos données et de passer dans une base de dimension supérieure telle que nos données deviennent linéairement séparables comme cela est illustré sur la figure 2.5.

Pour ce faire nous allons remplacer le produit scalaire $x^T x$ dans l'expression de f par une fonction dite « noyaux » : $K : \xi \times \xi \rightarrow \mathbb{R}$.

On donne quelques exemple de noyaux courant en figure 2.6. Logiquement, comme nous cherchons à séparer les données, les noyaux prennent des valeurs importante pour des données proches.

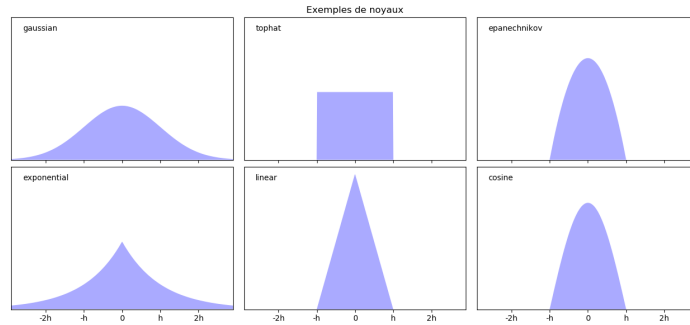


FIGURE 2.5 – Affichage de la densité pour des noyaux courant [Documentation scikit-learn]

Les fonctions noyaux K doivent vérifier les conditions de Mercer :

- K est continue symétrique
- $K(x_i, x_j)_{1 \leq i, j \leq N}$ 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 $\phi : \xi \rightarrow H$ telle que $K(x, y) = \langle \phi(x), \phi(y) \rangle$.

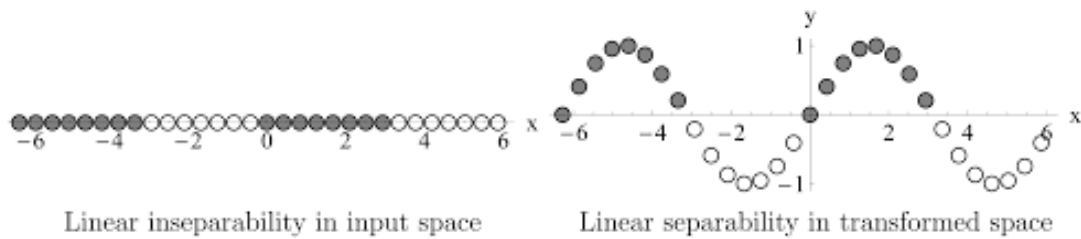


FIGURE 2.6 – Astuce du noyaux : projeter les données dans un espace de dimension plus grande, où elles deviennent séparables linéairement.[Mathias Rupp *Machine Learning for Quantum Mechanics in a Nutshell*]

L'estimateur devient ainsi en effectuant le produit scalaire sur l'ensemble des données $(x_i)_{1 \leq i \leq N}$:

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

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 avec le produit scalaire.

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 problèmes d'overfitting.

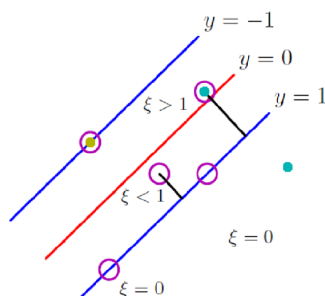


FIGURE 2.7 – [C. Bishop, Pattern Recognition and Machine Learning, Springer 2006]

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 :

$$\begin{cases} \arg \min_{w,b} \frac{1}{2} \|w\|^2 + C \sum_i \xi_i \\ \forall 1 \leq i \leq N, y_i(w \cdot \phi(x_i) + b) \geq 1 - \xi_i \\ \forall 1 \leq i \leq N, \xi_i \geq 0 \end{cases}$$

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

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

2.4 Application des méthode de noyaux à la Ridge Regression

La régression linéaire classique d'observations $(x_i, y_i)_{1 \leq i \leq N}$ par une fonction $f(\tilde{x}) = \langle w, \tilde{x} \rangle$ 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 $|\cdot|_{L^2}$ à cette erreur avec un paramètre $\lambda \geq 0$:

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

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 noyaux la fonction f devient $f(\tilde{x}) = \sum_{i=1}^n \alpha_i K(x_i, \tilde{x})$, 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 :

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

où $K \in \mathbb{R}^{n \times n}$ est la matrice du noyaux $K_{i,j} = K(x_i, x_j)$.

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 les plus petites possible. Nous pouvons d'abord définir la somme des résidus au carré RSS (residual sum of squares) :

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

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

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

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

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

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

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

où \bar{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.

Noyaux Gaussien Nous définissons le noyaux gaussien définis par :

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

Ce noyaux comporte un hyperparamètre σ à déterminer.

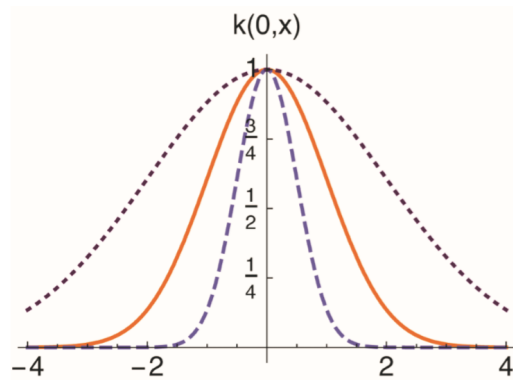


FIGURE 3.1 – noyaux gaussien avec $\sigma = 0.5$ (\cdots), 1 ($—$), 2 ($- - -$) [Mathias Rupp *Machine Learning for Quantum Mechanics in a Nutshell*]

Noyaux linéaire Nous définissons le noyaux linéaire par :

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

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

Noyaux Lagrangien Nous définissons le noyaux gaussien définis par :

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

Ce noyaux comporte un hyperparamètre σ à déterminer.

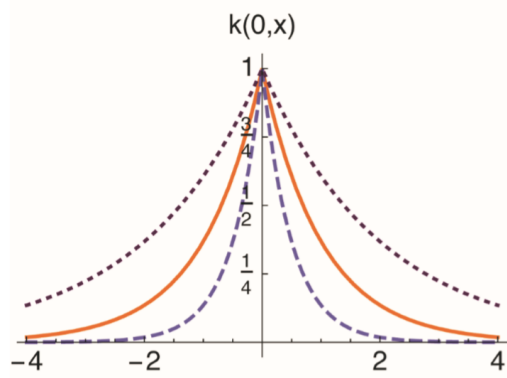


FIGURE 3.2 – noyaux laplacien avec $\sigma = 0.5$ (— —), 1 (—), 2 ($\cdot \cdot \cdot$) [Mathias Rupp *Machine Learning for Quantum Mechanics in a Nutshell*]

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

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

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

Nous allons ainsi dans la recherche du meilleur modèle pour prédire les énergie d'atomisation des molécules nous particulièrement 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 ainsi en un training set de 1000 molécules, et un testing set contenant le reste des molécules.

Nous décomposons encore cette base d’entraînement 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 d’overfitting.

Après sélections de nos hyperparamètres, nous entraînons 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’hydrogène par molécules dans notre dataset, nous pouvons voir sur la Figure 3.3 que cette répartition est très 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 très 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.

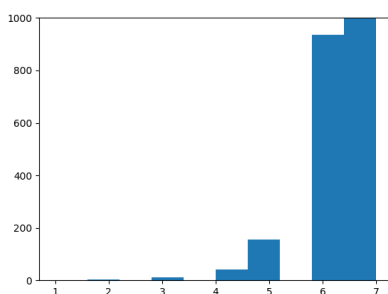


FIGURE 3.3 – Histogramme du nombre d’atome non-H par molécule

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 noyaux gaussien), et la mise en forme des données pour l'affichage et réalisé dans *lecture_donnee.py*.

Résultats pour le noyaux 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 $\gamma = \frac{1}{2\sigma^2}$

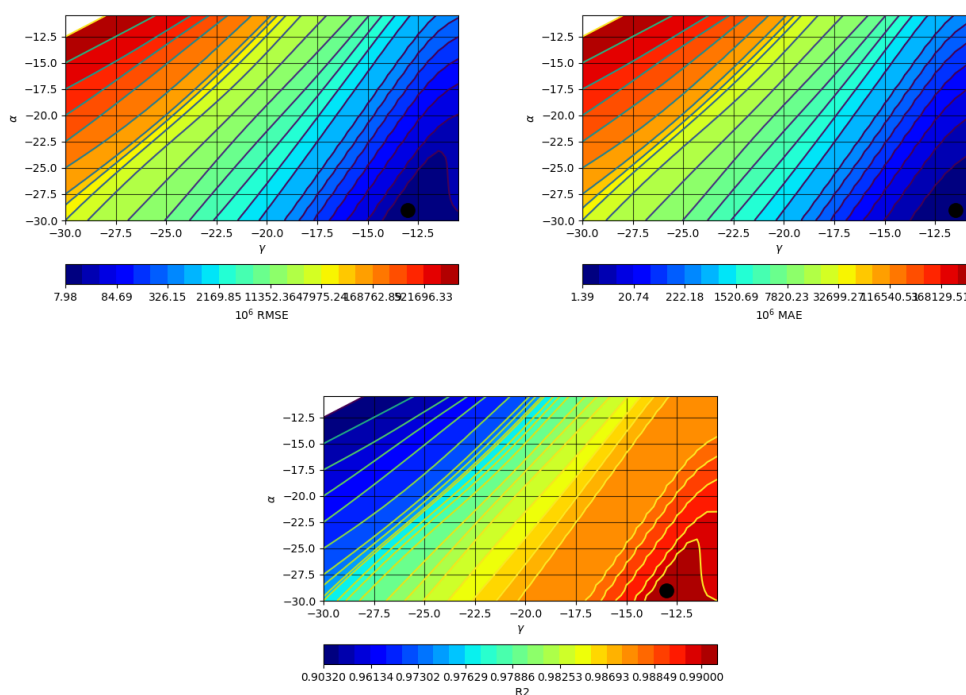


FIGURE 3.4 – Résultat des modèles pour un noyau gaussien, le point noir en $[-13, -29]$ correspond au minimum de l'erreur de l'erreur RMSE sur la grille

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

Erreur Utilisé	RMSE	MAE	R ²
KRR avec noyaux gaussien	10.3356	11.0503	0.9977
KRR avec noyaux laplacien	10.3356	11.0503	0.9977