Apprentissage sous contrainte

Baptiste Aussel et Albin Cintas

Juin 2022

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES TOULOUSE

INP TOULOUSE ENSEEIHT

Contents

1 Introduction

Le but de ce challenge est de modéliser la potentielle energie inter-atomic de petites molécules organiques. on note $r = r_1, r_2, ..., r_N$ les positions des atomes dans la molécule sur un espace 3D et $E(r)$ l'energie atomique de la configuration r. Le but est de prédire E(r) via des modèles différents, en utilisant les informations géométriques de la molécules données par r.
Nous observons une contrainte, la contrainte de symétrie :

$$(T_b(r)) = E(T_U(r)) = E(T_\sigma(r) = E(r)$$

Pour toute translation b, rotation U et permutation σ définie ci-dessous
$T_b(r) = r + b$, $T_U(r) = U(r)$ et $T_\sigma(r) = r_{\sigma(1)}, r_{\sigma(2)}, ..., r_{\sigma(n)}$

2 Jeu de donnée

2.1 QM7-X

Nous avons utilisé le jeu de donnée QM7-X pour modéliser nos molécules. Ce jeu de donnée contient 8466 strucutures de molécules. Toutes les molécules possèdent jusqu'a 7 atomes lourds(C,N,O,H,S,Cl). Afin de couvrir l'espace important des composés chimiques.
Nous obtenons donc la matrice de coulomb d'une molécule, sa masse et le nombre d'atome dans la molécule.
Afin d'évaluer l'erreur sur notre apprentissage nous utilisions la Root Mean Squared Error:

$$\sqrt{\frac{1}{D}\sum_{id=1}^{d}(E(r_{id}) - \hat{E}(r_{id}))^2}$$

2.2 La matrice de Coulomb

Dans un premier temps on s'intéresse à la représentation de ces molécules par matrice de Coulomb. Ces matrices exploitent la charge atomique et la distance entre les atomes dans une molécule. En effet, chaque terme est donné par :

$$C_{ij} = \begin{cases} 0.5 \times Z_i^{2.4} \text{ si } i = j \\ \frac{Z_i Z_j}{|R_i - R_j} \text{ sinon.} \end{cases}$$

La matrice de Coulomb a l'avantage d'être invariante en rotation et translation. Afin de respecter l'invariance par permutation, on procède au tri des colonnes de la matrice par leur norme l2. On tri simultannément leur lignes dans le même ordre. Ceci nous apporte alors l'invariance par permutation, et les contraintes physiques sont maintenant respectées.

2.3 Augmentation des données

Comme décrit dans [1], nous avons procédé à une augmentation de données consistant à bruiter la matrice de Coulomb en ajoutant un bruit gaussien à la norme des lignes, et à trier la nouvelle matrice obtenue par ligne. Les colonnes sont également permutées par la même permutation que celle des lignes.
Ceci nous permet alors de générer de nouvelles matrices de Coulomb qui viendront compléter notre jeu de données. L'algorithme est détaillé ci-dessous :

Algorithm for generating a random Coulomb matrix

1. Take any Coulomb matrix C among the set of matrices that are valid Coulomb matrices of M and compute its row norm $||C|| = (||C_1||, \ldots, ||C_d||)$.
2. Draw $n \sim \mathcal{N}(0, \sigma I)$ and find the permutation P that sorts $||C|| + n$, that is, find the permutation that satisfies $\text{permute}_P(||C|| + n) = \text{sort}(||C|| + n)$.
3. Permute C row-wise and then column-wise with the same permutation, that is, $C_{\text{random}} = \text{permutecols}_P(\text{permuterows}_P(C))$.

2.4 Many-body tensor

Nous avons ajouté à cela une nouvelle représentation très exploitée dans les études similaires à l'état de l'art. Celle-ci consiste à représenter chaque chaine d'atome de k éléments (chaque atome, chaque liaison, chaque triplet, etc...) par une distribution qui dépend de la charge atomique, et des distances. On peut alors représenter une molécule par un signal de la manière suivante :

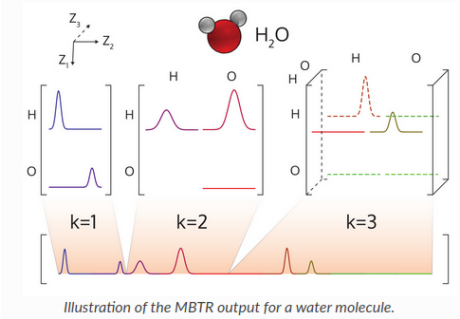

Illustration of the MBTR output for a water molecule.

Cette représentation nous permet de maintenir l'invariance par rotation, translation et permutation.

2.5 Implémentation

Nous avons implémenté ces représentations, ainsi qu'exploité les caractéristiques des atomes grâce aux librairies python DScribe et ASE.

3 différentes méthodes

Les résultats seront présentés dans la partie 4 : Comparaison des résultats.

3.1 Les modèles de régression

Pour commencer nous avons d'abord essayer d'implémenter des méthodes de régressions linéaires en utilisant la liste du nombre d'atome et la masse de la molécule (le rajout de la masse n'est pas très utile car dépendant du nombre de d'atome dans la molécule).

3.1.1 Régression linéaire

La premiere méthode que nous avons implémenté est une régréssion linéaire classique. L'augmentation de données apporte une amélioration sur le score de cette méthode.

3.1.2 Régression Ridge

Ensuite nous avons implémenté une régression ridge. Suite à une gridSearch les meilleurs paramètres que nous avons trouvé sont ceux ci : kernel='rbf', $\gamma = 0.1$, $\alpha = 1$. Le kernel gaussien permettait selon [2] d'obtenir de bons résultats, mais nous n'avons pas réussi à l'exploiter.

3.1.3 Régression XGBoost

Pour finir avec les régresseurs nous avons utilisé XGBoost, ridge. Cette méthode de Gradient Boosting n'a pas donné de résultats à la hauteur de ceux de la régression linéaire.

3.2 Les réseaux de neurones

3.2.1 Les CNN 1D

On teste un réseau convolutionel 1D sur les matrices de Coulomb vectorisées, concaténées à une liste contenant le cardinal de chaque atome (C, H, O, S, N, Cl). On choisit empiriquement le réseau suivant :

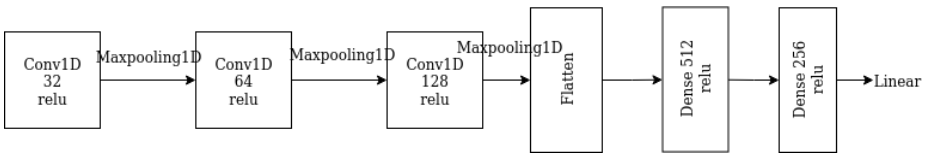

Figure 2: Réseau convolutionnel 1D

3.2.2 Les CNN 2D

On implémente ici un réseau convolutionel 2D sur la représentation MBTR des **liaisons entre 2 atomes**. En effet, en discretisant les signaux en 100 points (= 100 canaux), on peut appliquer un réseau convolutionnel. On choisit empiriquement le réseau suivant :

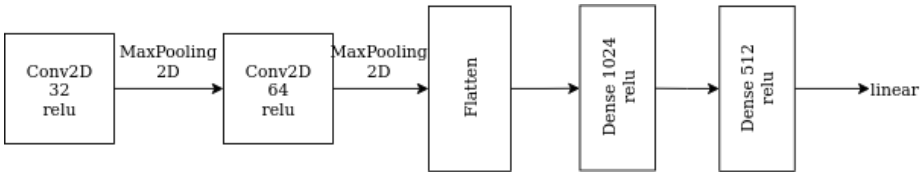

Figure 3: Réseau convolutionnel 2D

3.2.3 La fusion

En partant du constat que nos deux réseaux précédents apportaient chacun des résultats satisfaisants, on a opté pour un modèle qui réunit ces deux réseaux convolutionnels. On récupère les deux couches flatten de chacun des réseaux convolutionnels, et on les concatène, puis on les passe dans des couches denses. On peut représenter le réseau de la manière suivante :

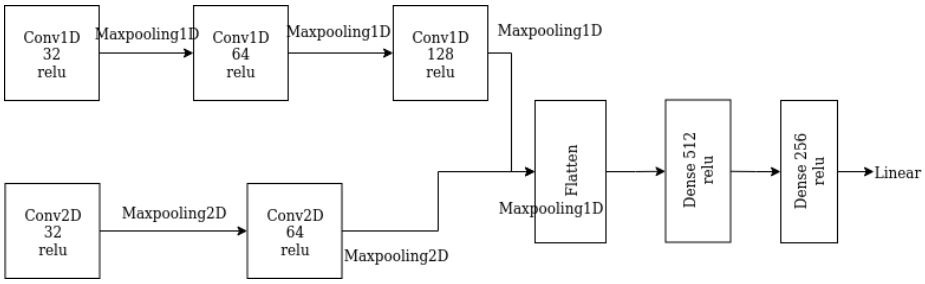

Figure 4: Fusion des réseaux convolutionnels

4 Comparaison des résultats

On a séparé nos données en un jeu de données d'entrainement et un jeu de données de validaton. On a considéré 20% du jeu de données comme données de validation.

	Epoch	Validation
Rég Lin	x	0.34
Ridge	x	large sup than 1
XGBoost	x	0.55
CNN 1D	1500	0.15
CNN 2D	1500	0.12
Fusion	1500	0.09

Table 1: scores sur les données de validation

5 Conclusion

En conclusion notre modèle le plus efficace est la fusion des 2 réseaux de neurones tout en utilisant la réprésentation MBTR de nos données. On s'est cependant rendu compte sur les données de test sur la plateforme Kaggle, que notre modèle a du mal à généraliser. En effet, les score RMSE kaggle étaient en général plus élevés que ceux trouvés sur notre jeu de validation.