

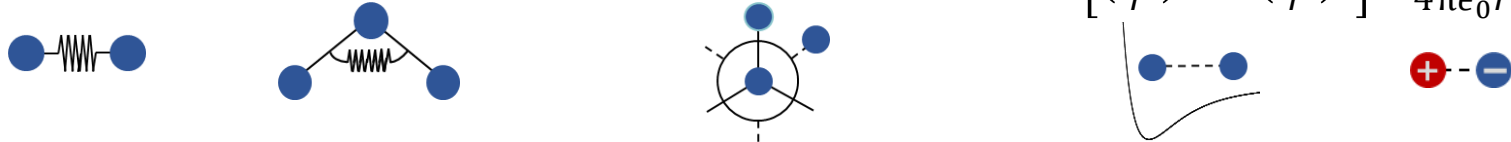
How to use the Force Field Kernel Mean Descriptor

Yoshihiro Hayashi
yhayashi@ism.ac.jp
2023/07/31

About the Force Field Kernel Mean Descriptor

Parameters of the molecular force field (GAFF2) for molecular dynamics (MD) simulations are encoded to construct the descriptor by the kernel mean embedding.

Force field of MD simulation (AMBER type)

$$U(\mathbf{r}) = K_{\text{bond}}(r - r_0)^2 + K_{\text{angle}}(\theta - \theta_0)^2 + K_{\text{dihedral}}[1 + \cos(n\phi - \delta)] + 4\epsilon_{ij} \left[\left(\frac{\sigma_{ij}}{r} \right)^{12} - \left(\frac{\sigma_{ij}}{r} \right)^6 \right] + \frac{q_i q_j}{4\pi\epsilon_0 r}$$


- Molecules in MD simulations are characterized by force field parameters that relate molecular geometries, covalent bonding interactions, van der Waals interactions, and Coulomb interactions.
- The parameters are the vectors of which length is different by molecules.
- The parameters are encoded to the fixed length vectors by the kernel mean embedding.

Def. Kernel mean (m_X)

$$m_X := E[\Phi(X)] = E[k(\cdot, X)] = \int k(\cdot, x) dP(x)$$

X : set of variable (force field parameters)

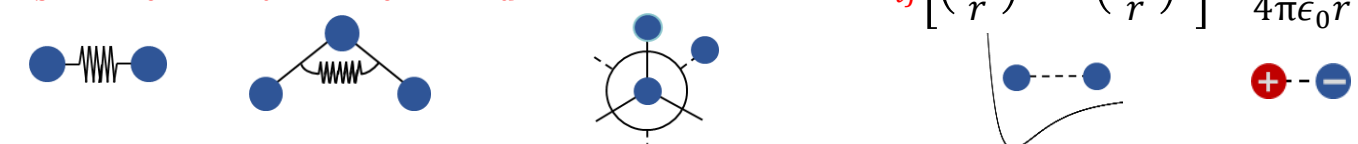
P : probability distribution

k : positive definite kernel (e.g. Gaussian function)

Finite sample approximation

$$m_X := \frac{1}{n} \sum_i^n k(\cdot, x_i)$$

Step 1: kernel mean embedding

$$U(\mathbf{r}) = K_b(r - r_0)^2 + K_a(\theta - \theta_0)^2 + K_d[1 + \cos(n\phi - \delta)] + 4\epsilon_{ij} \left[\left(\frac{\sigma_{ij}}{r} \right)^{12} - \left(\frac{\sigma_{ij}}{r} \right)^6 \right] + \frac{q_i q_j}{4\pi\epsilon_0 r}$$


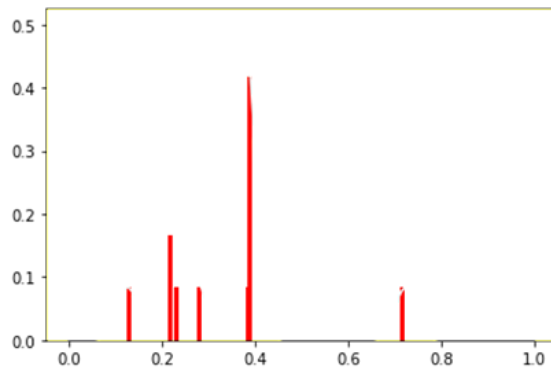
Kernel means are independently calculated for each of the 10 parameter classes.

atoms	Mass (m) Charge (q) Strength of vdW interaction (ϵ) Equilibrium distance of vdW interaction (σ)
bonds	Force constant of bonds (K_b), Equilibrium distance of bonds (r_0) Bond polarity ($ q_i - q_j $) (polar)
angles	Force constant of angles (K_a), Equilibrium angles (θ_0)
dihedrals	Force constant of dihedral angles (K_d)



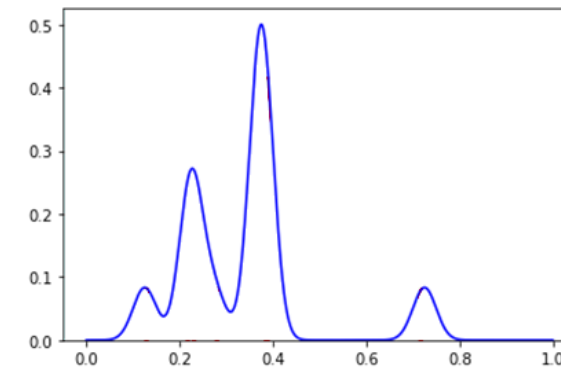
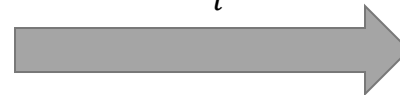
$$\begin{bmatrix} m_m \\ m_q \\ m_\epsilon \\ m_\sigma \\ m_{Kb} \\ m_{r0} \\ m_{polar} \\ m_{Ka} \\ m_{\theta0} \\ m_{Kd} \end{bmatrix}$$

Dimension: $10 \times N$



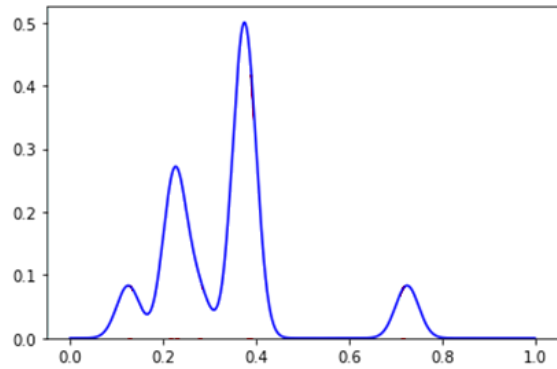
Distribution of parameters $P(x)$

$$m_x := \frac{1}{n} \sum_i^n k(\cdot, x_i)$$

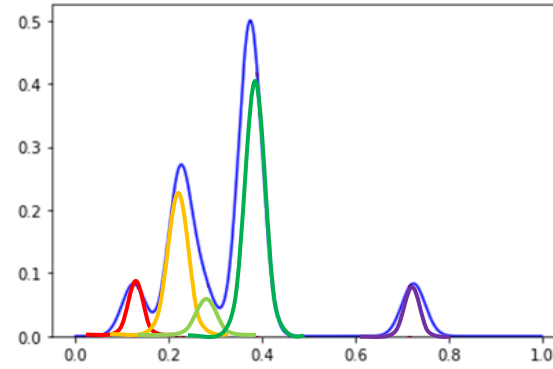


Kernel mean m_x (function)

Step 2: discretization

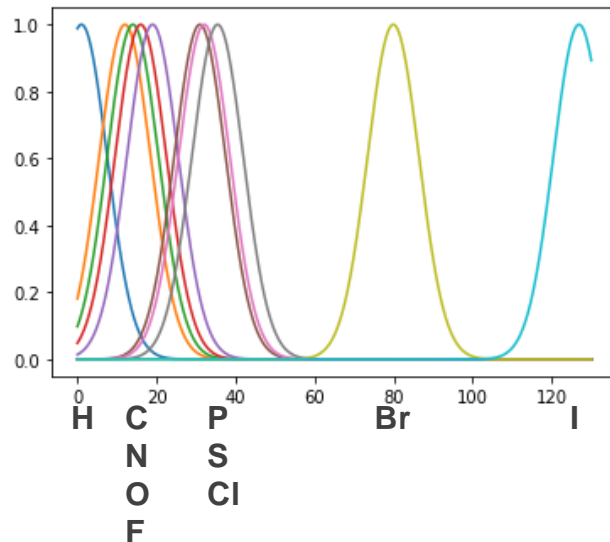


Kernel mean m_x (function)



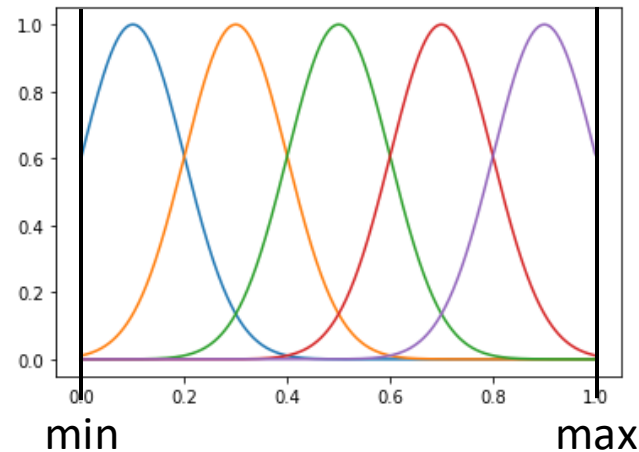
Discretization by N Gaussian functions

Mass



Discretizing at corresponding to atomic mass (10 points)

Charge, ϵ , σ , K_b , r_0 , polar, K_a , θ_0 , K_d



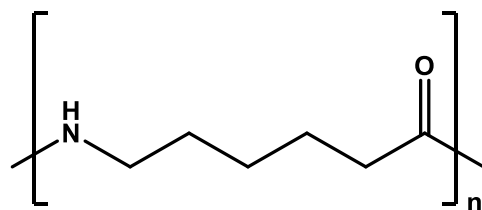
Discretizing at grid points
(Default: 20 grid points)

Hyperparameters

- Number of Gaussian in discretization
- Positions of the discretization points
- Band width of Gaussian function

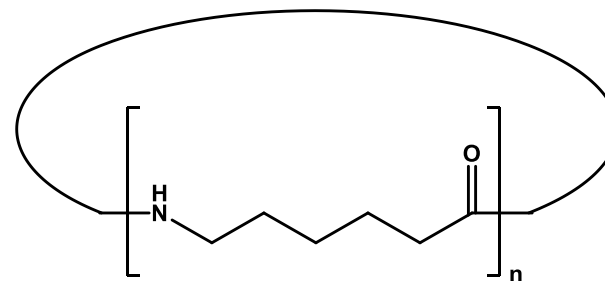
Preprocessing for polymers

- In the preprocessing for the polymer descriptor, the polymer SMILES is transformed to the macrocyclic oligomer.



CCCCC(=O)N

NCCCCC(=O)



- Appropriate representation of connected units
- Without terminal groups

Sample code of force field descriptor for RadonPy 0.2.x

```
import pandas as pd
from radonpy.ff.gaff2_mod import GAFF2_mod
from radonpy.ff.descriptor import FF_descriptor

mp = 10      # parallel number
n = 10       # cyclic oligomerization for polymer
nk = 20      # number of Gaussian in discretization (Hyperparameter)
sigma = 1/nk/2 # band width of Gaussian kernel (Hyperparameter)
mu = None    # list of discretizing points (Hyperparameter)

ff_desc = FF_descriptor(GAFF2_mod(), polar=True)
desc = ff_desc.ffkm_mp(smiles_list, mp=mp, nk=nk, s=sigma, mu=mu, cyclic=n)

desc_names = ff_desc.ffkm_desc_names(nk=nk)
FFKM_data = pd.DataFrame(desc, columns=desc_names, index=index_list)
```

FF_descriptor(GAFF2_mod(), polar=True)

GAFF2_mod(): the instance of force field class in RadonPy

polar: Add polar in FFKM descriptor ($\Delta q_{i,j} = |q_i - q_j|$)

ff_desc.ffkm_mp(smiles_list, mp=None, nk=20, s=None, mu=None, cyclic=10)

calculation of force field kernel mean (FFKM) descriptor, return variable is a 2D numpy.ndarray

smiles_list: List of SMILES to calculate FFKM descriptor

mp: number of parallel (None means using all CPU core)

nk: number of Gaussian kernels in discretization (default: 20)

mu: list of positions of the discretization points (default: None, this means discretizing at grid points)

s: band width of the Gaussian kernel (default: None, this means $1/(nk^2)$)

cyclic: for preprocessing of polymer SMILES, specifies the degree of polymerization in the generation of a macrocyclic oligomer. (default: 0, means that do not generate a macrocyclic oligomer)

ff_desc.ffkm_desc_name(nk=20)

Returns the name list of each feature value in FFKM descriptor.

Sample code of force field descriptor for RadonPy 0.3.x

Bold and underline means the change from RadonPy 0.2.x.

```
import pandas as pd
from radonpy.ff.gaff2_mod import GAFF2_mod
from radonpy.ff.descriptor import FF_descriptor

mp = 10      # parallel number
n = 10       # cyclic oligomerization for polymer
nk = 20      # number of Gaussian in discretization (Hyperparameter)
sigma = 1/nk/2 # band width of Gaussian kernel (Hyperparameter)
mu = None    # list of discretizing points (Hyperparameter)

ff_desc = FF_descriptor(GAFF2_mod(), polar=True)
desc = ff_desc.ffkm_mp(smiles_list, mp=mp, nk=nk, s=sigma, mu=mu, cyclic=n)
```

FF_descriptor(GAFF2_mod(), polar=True)

GAFF2_mod(): the instance of force field class in RadonPy

polar: Add “polar” feature value in FFKM descriptor ($\Delta q_{i,j} = |q_i - q_j|$)

ff_desc.ffkm_mp(smiles_list, mp=None, nk=20, s=None, mu=None, cyclic=10)

calculation of force field kernel mean (FFKM) descriptor, return variable is a **pandas.DataFrame**

smiles_list: List of SMILES to calculate FFKM descriptor

mp: number of parallel (None means using all CPU core)

nk: number of Gaussian kernels in discretization (default: 20)

mu: list of positions of the discretization points (default: None, this means discretizing at grid points)

s: band width of the Gaussian kernel (default: None, this means **1/(nk*sqrt(2))**)

cyclic: for preprocessing of polymer SMILES, specifies the degree of polymerization in the generation of a macrocyclic oligomer. (default: 0, means that do not generate a macrocyclic oligomer)