مدل سازی انرژی پیوندی مولکولی با استفاده از یادگیری ماشین

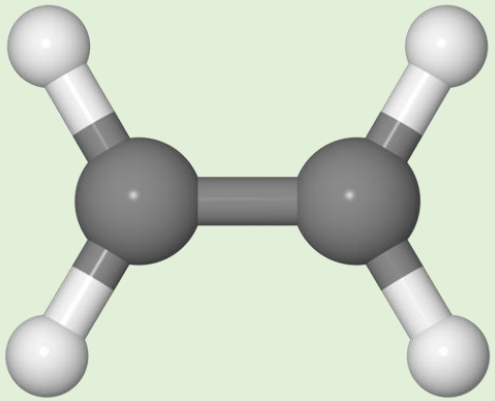
مجتبی اعلائی

دانشکده فیزیک دانشگاه صنعتی اصفهان
گروه شبیه سازی کوانتومی مواد
https://qsm.iut.ac.ir/

دومین کارگاه یادگیری ماشینی در فیزیک: کاربردها در ماده چگال

۱۱ تا ۱۳ مهرماه ۱۳۹۷
دانشکده فیزیک دانشگاه تهران

انگیزه و هدف



برای نمایش یک مولکول به مکان و نوع اتم‌ها نیاز داریم:

		x	y	z	
$Z_1=6$	C	0.93211000	0.03083000	-0.07433000	$\mathbf{R}_1$
$Z_2=6$	C	2.26316000	0.03083000	-0.07433000	$\mathbf{R}_2$
$Z_3=1$	H	0.38637000	0.96781000	-0.11896000	$\mathbf{R}_3$
$Z_4=1$	H	0.38637000	-0.90614000	-0.02971000	$\mathbf{R}_4$
$Z_5=1$	H	2.80890000	0.96781000	-0.11896000	$\mathbf{R}_5$
$Z_6=1$	H	2.80890000	-0.90614000	-0.02971000	$\mathbf{R}_6$

$\{Z_I, \boldsymbol{R}_I\}$

مکانیک کوانتومی

معادله شرودینگر

$$H\Psi = E\Psi$$

$$H(\{Z_I, \boldsymbol{R}_I\}) \xrightarrow{\Psi} E$$

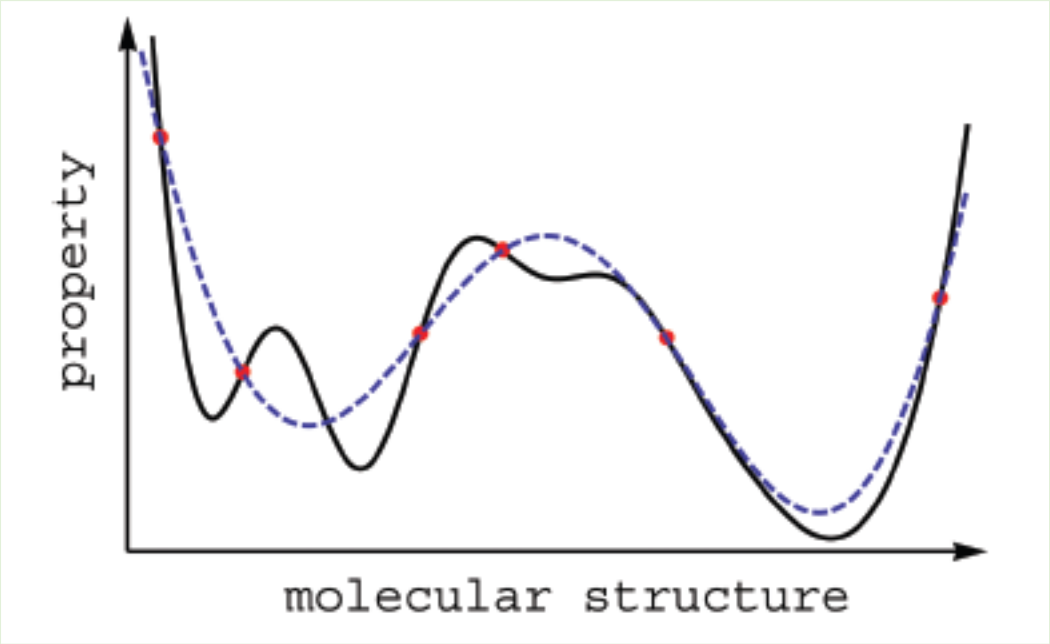
یادگیری ماشین

(ML)

$$\{Z_I, \boldsymbol{R}_I\} \xrightarrow{\text{(ML)}} E$$

انگیزه و هدف

$$\{Z_I, \boldsymbol{R}_I\} \xrightarrow{\text{ML}} \text{خواص مولکول}$$



1-M Rupp, International Journal of Quantum Chemistry 115 (16), 1058-1073

انگیزه و هدف

در این سخنرانی ما قصد داریم با استفاده از مقاله آموزشی-مروری که توسط Matthias Rupp نوشته شده است مقادیر انرژی پیوندی را برای مجموعه‌ای از مولکول‌های آلی پیش بینی کنیم:

TUTORIAL REVIEWS WWW.Q-CHEM.ORG International Journal of QUANTUM CHEMISTRY

Machine Learning for Quantum Mechanics in a Nutshell

Matthias Rupp*

Models that combine quantum mechanics (QM) with machine learning (ML) promise to deliver the accuracy of QM at the speed of ML. This hands-on tutorial introduces the reader to QM/ML models based on kernel learning, an elegant, systematically nonlinear form of ML. Pseudocode and a reference implementation are provided, enabling the reader to reproduce results from recent publications where atomization energies of small organic molecules are predicted using kernel ridge regression. © 2015 Wiley Periodicals, Inc.

DOI: 10.1002/qua.24954

فهرست مطالب در این سخنرانی

بخش تئوری

- رگرسیون خطی (Linear regression)
- رگرسیون خطی مقید (Linear ridge regression)
- نگاشت داده‌ها به یک فضای ویژگی (mapping data to a feature space)

بخش تعاریف و تکنیک

- ماتریس کولنی (Coulomb matrix)
- هسته‌ی گوسی (Gaussian Kernel)
- تخمین تشابه بین مولکول‌ها
- روش حل چولسکی

بخش عملی

- توضیح برخی از قسمت‌های یک برنامه برای پیش بینی انرژی پیوندی

بخش تئوری

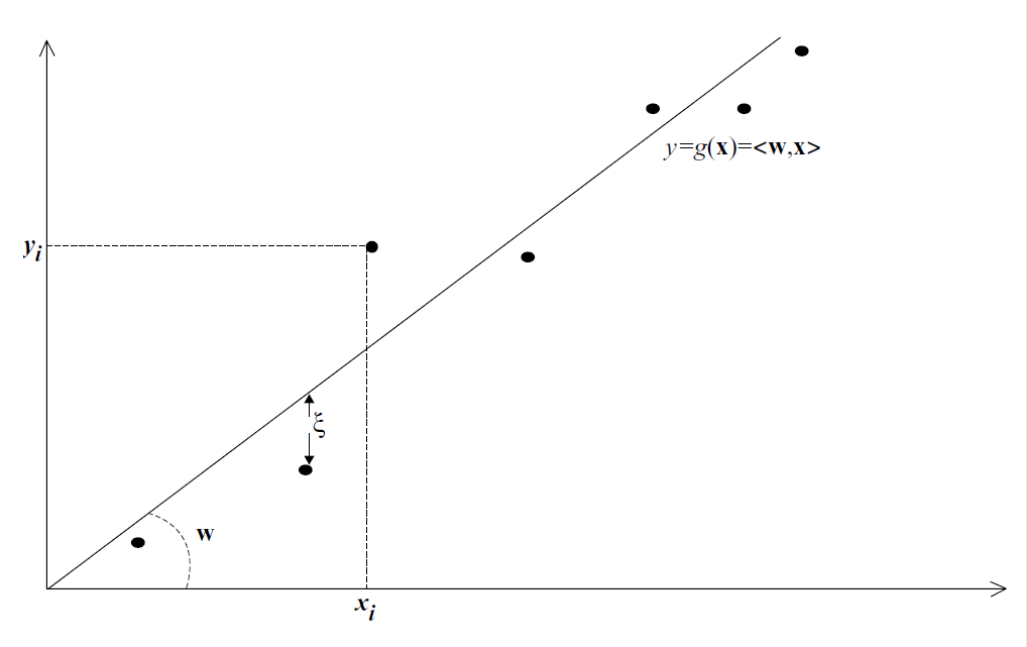
رگرسیون خطی
(Linear regression)

مجموعه ی آموزشی (training set):

$$S = \{(\mathbf{x}_1, y_1), \ldots (\mathbf{x}_l, y_l)\}$$

$$\mathbf{x}_i \text{ from } X \subseteq \mathbb{R}^n$$

$$y_i \text{ from } Y \subseteq \mathbb{R}$$



برای یک داده جدید x، y را با یک تابع خطی پیش بینی می‌کنیم

$$\mathbf{x} = (x_1, x_2, \ldots, x_n)$$

$$y \approx g(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle = \sum_{i=1}^{n} w_i x_i$$

برای پیدا کردن w می توان تابع زیان (loss function) یا تابع هزینه (cost function) را کمینه کرد:

$$\mathcal{L}(g,S)=\mathcal{L}(\mathbf{w},S)=\sum_{i=1}^{l}(y_i-g(\mathbf{x}_i))^2=\sum_{i=1}^{l}\xi_i^2=\sum_{i=1}^{l}\mathcal{L}((\mathbf{x}_i,y_i),g)$$

تابع زیان (loss function)

با تعریف ξ به صورت زیر:

$$\xi=\mathbf{y}-\mathbf{X}\mathbf{w} \qquad \mathbf{X}=\begin{pmatrix}\boxed{\mathbf{x}_1}\\ \mathbf{x}_2\\ \vdots\\ \mathbf{x}_l\end{pmatrix}$$

بردار n مولفه‌ای

تابع زیان را می‌توان به صورت زیر نوشت:

$$\mathcal{L}(\mathbf{w},S)=\|\xi\|^2=(\mathbf{y}-\mathbf{X}\mathbf{w})^{\mathsf{T}}(\mathbf{y}-\mathbf{X}\mathbf{w})$$

با مشتق گیری از تابع زیان به نتیجه زیر می‌رسیم:

$$\mathbf{w}=(\mathbf{X}^{\mathsf{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathsf{T}}\mathbf{y}$$

روش حل اولیه (primal solution)

بیان مسئله با استفاده از تابع هسته (Kernel function):

$$\mathbf{w} = (\mathbf{X}^\intercal \mathbf{X})^{-1}\mathbf{X}^\intercal \mathbf{y} = \mathbf{X}^\intercal \boxed{\mathbf{X}(\mathbf{X}^\intercal \mathbf{X})^{-2}\mathbf{X}^\intercal \mathbf{y}} = \mathbf{X}^\intercal \alpha$$

بنابراین w را می توان به صورت زیر نوشت:

$$\mathbf{w} = \sum_{i=1}^{l} \alpha_i \mathbf{x}_i$$

و تابع g :

$$g(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle = \left\langle \sum_{i=1}^{l} \alpha_i \mathbf{x}_i, \mathbf{x} \right\rangle = \sum_{i=1}^{l} \alpha_i \langle \mathbf{x}_i, \mathbf{x} \rangle = \sum_{i=1}^{l} \alpha_i k(\mathbf{x}_i, \mathbf{x})$$

روش حل دوگانه (dual solution)

که

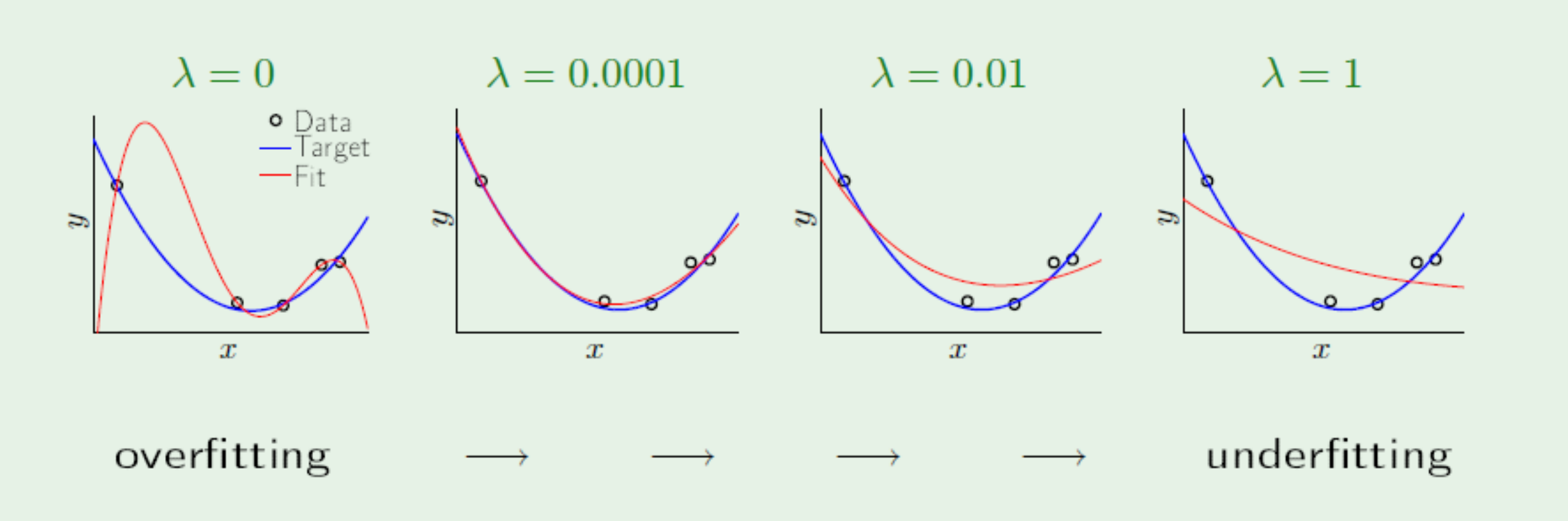
$$k(\mathbf{x}_i, \mathbf{x}) = \langle \mathbf{x}_i, \mathbf{x} \rangle$$

تابع هسته (هسته خطی یا linear kernel)

رگرسیون خطی مقید
(Linear ridge regression)

$$\min_{\mathbf{w}} \mathcal{L}_\lambda(\mathbf{w}, S) = \min_{\mathbf{w}} \lambda \|\mathbf{w}\|^2 + \sum_{i=1}^{l} (y_i - g(\mathbf{x_i}))^2$$

در اینجا λ یک پارامتر تنظیمی است



رگرسیون خطی مقید
(Linear ridge regression)

کمینه کردن تابع زیان با قید منجر به معادله زیر می شود:

$$\mathbf{X}^\intercal\mathbf{X}\mathbf{w}+\lambda\mathbf{w}=(\mathbf{X}^\intercal\mathbf{X}+\lambda\mathbf{I}_n)\mathbf{w}=\mathbf{X}^\intercal\mathbf{y}$$

$$\mathbf{w}=(\mathbf{X}^\intercal\mathbf{X}+\lambda\mathbf{I}_n)^{-1}\mathbf{X}^\intercal\mathbf{y}$$

$$g(\mathbf{x})=\langle\mathbf{w},\mathbf{x}\rangle=\mathbf{w}^\intercal\mathbf{x}=\mathbf{y}^\intercal\mathbf{X}(\mathbf{X}^\intercal\mathbf{X}+\lambda\mathbf{I}_n)^{-1}\mathbf{x}$$

روش حل اولیه (primal solution)

$$\mathbf{w}=\lambda^{-1}\mathbf{X}^\intercal(\mathbf{y}-\mathbf{X}\mathbf{w})=\mathbf{X}^\intercal\alpha$$

$$\Downarrow$$

$$\alpha=\lambda^{-1}(\mathbf{y}-\mathbf{X}\mathbf{w})$$

$$\Downarrow$$

$$\lambda\alpha=(\mathbf{y}-\mathbf{X}\mathbf{X}^\intercal\alpha)$$

$$\Downarrow$$

$$(\mathbf{X}\mathbf{X}^\intercal+\lambda\mathbf{I}_l)\alpha=\mathbf{y}$$

$$\Downarrow$$

$$\alpha=(\mathbf{G}+\lambda\mathbf{I}_l)^{-1}\mathbf{y}$$

$\mathbf{G}=\mathbf{X}\mathbf{X}^\intercal$ یا به صورت مولفه‌ای $\mathbf{G}_{i,j}=\langle\mathbf{x}_i,\mathbf{x}_j\rangle$

$$\Rightarrow\quad g(\mathbf{x})=\langle\mathbf{w},\mathbf{x}\rangle=\left\langle\sum_{i=1}^{l}\alpha_i\mathbf{x}_i,\mathbf{x}\right\rangle=\sum_{i=1}^{l}\alpha_i\langle\mathbf{x}_i,\mathbf{x}\rangle=\mathbf{y}^\intercal(\mathbf{G}+\lambda\mathbf{I}_l)^{-1}\mathbf{k}\qquad k_i=\langle\mathbf{x}_i,\mathbf{x}\rangle$$

روش حل دوگانه (dual solution)

رگرسیون خطی مقید
(Linear ridge regression)

چند نکته در مورد حل دوگانه:

1- تمام اطلاعات مجموعه‌ی آموزشی در ضرب داخلی بین تمام جفت نقاط آنها قرار داده می‌شود یعنی همان ماتریس $\mathbf{G}$

2- به طور مشابه برای پیش بینی در یک نقطه‌ی جدید $\mathbf{x}$ نیاز به ضرب داخلی این نقاط با نقاط مجموعه‌ی آموزشی هستیم

$$g(\mathbf{x}) = \sum_{i=1}^{\ell} \alpha_i k(\mathbf{x}_i, \mathbf{x})$$

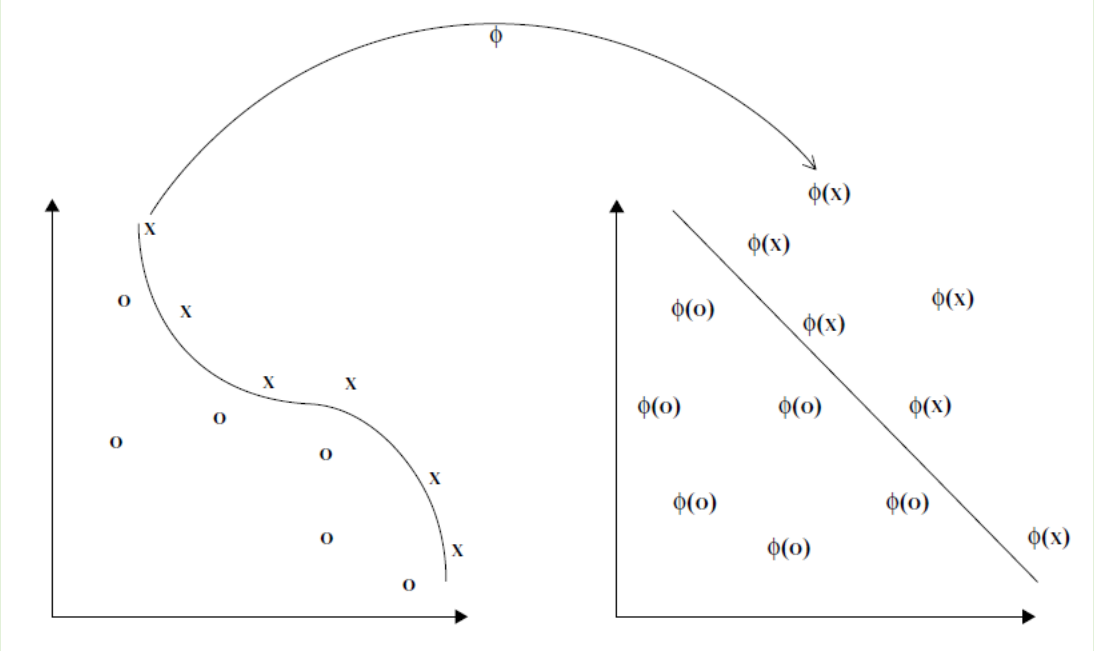
3- به ماتریس $\mathbf{G}$ ماتریس گرام (Gram matrix) گفته می‌شود.

نگاشت داده‌ها به یک فضای ویژگی
(mapping data to a feature space)

فضای ویژگی (feature space) $\phi : \mathbf{x} \in \mathbb{R}^n \mapsto \phi(\mathbf{x}) \in \mathbb{R}^N$

مثال:

$$\phi : \mathbf{x} = (x_1, x_2) \in \mathbb{R}^2 \mapsto \phi(\mathbf{x}) = (x_1^2, x_2^2, x_1 x_2, x_2 x_1) \in \mathbb{R}^4$$



دلیل استفاده از نگاشت: تبدیل یک مسئله غیرخطی به خطی.

نگاشت داده‌ها به یک فضای ویژگی
(mapping data to a feature space)

مشکل: در صورتیکه فضای ویژگی دارای ابعاد بسیار بزرگی باشد، کار با آن پردردسر خواهد شد.

راه حل: استفاده از ضرب داخلی در فضای ویژگی

روش: استفاده از (تابع) هسته

تعریف تابع هسته

$$k(\mathbf{x},\mathbf{z}) = \langle \phi(\mathbf{x}), \phi(\mathbf{z}) \rangle$$

مثال:

$$\phi : \mathbf{x} = (x_1, x_2) \in \mathbb{R}^2 \rightarrow \phi(\mathbf{x}) = (x_1^2, x_2^2, \sqrt{2}x_1x_2) \in \mathbb{R}^3$$

$$\begin{aligned}\langle \phi(\mathbf{x}), \phi(\mathbf{z}) \rangle &= \left\langle (x_1^2, x_2^2, \sqrt{2}x_1x_2), (z_1^2, z_2^2, \sqrt{2}z_1z_2) \right\rangle \\ &= x_1^2 z_1^2 + x_2^2 z_2^2 + 2x_1x_2z_1z_2 \\ &= (x_1z_1 + x_2z_2)^2 = \langle \mathbf{x}, \mathbf{z} \rangle^2\end{aligned}$$

بنابراین:

$$k(\mathbf{x},\mathbf{z}) = \langle \mathbf{x}, \mathbf{z} \rangle^2$$

پس لازم نیست برای محاسبه ضرب داخلی در فضای ویژگی، عمل نگاشت را انجام داد.

نگاشت داده‌ها به یک فضای ویژگی
(mapping data to a feature space)

تعریف ماتریس گرام (Gram matrix) در فضای ویژگی

$$\mathbf{G}_{ij} = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle = k(\mathbf{x}_i, \mathbf{x}_j)$$

در این حالت اغلب به ماتریس گرام، ماتریس هسته (kernel matrix) گفته می‌شود و با $\mathbf{K}$ نمایش داده می‌شود

$$\mathbf{K} = \begin{pmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & k(\mathbf{x}_1, \mathbf{x}_2) & \cdots & k(\mathbf{x}_1, \mathbf{x}_\ell) \\ k(\mathbf{x}_2, \mathbf{x}_1) & k(\mathbf{x}_2, \mathbf{x}_2) & \cdots & k(\mathbf{x}_2, \mathbf{x}_\ell) \\ \vdots & \vdots & \ddots & \vdots \\ k(\mathbf{x}_\ell, \mathbf{x}_1) & k(\mathbf{x}_\ell, \mathbf{x}_2) & \cdots & k(\mathbf{x}_\ell, \mathbf{x}_\ell) \end{pmatrix}$$

با تشکیل ماتریس هسته می‌توان از روش دوگانه (که برای حل مسئله‌های خطی است) سود برد:

محاسبه α $\quad \alpha = (\mathbf{K} + \lambda \mathbf{I}_\ell)^{-1} \mathbf{y}$

پیش بینی برای یک نقطه جدید $(\mathbf{x})$ $\quad g(\mathbf{x}) = \sum_{i=1}^{\ell} \alpha_i k(\mathbf{x}_i, \mathbf{x})$

بخش تعاریف و تکنیک

ماتریس کولنی
(Coulomb matrix)

$$\{Z_I, \boldsymbol{R}_I\} \xrightarrow{\text{ML}} E$$

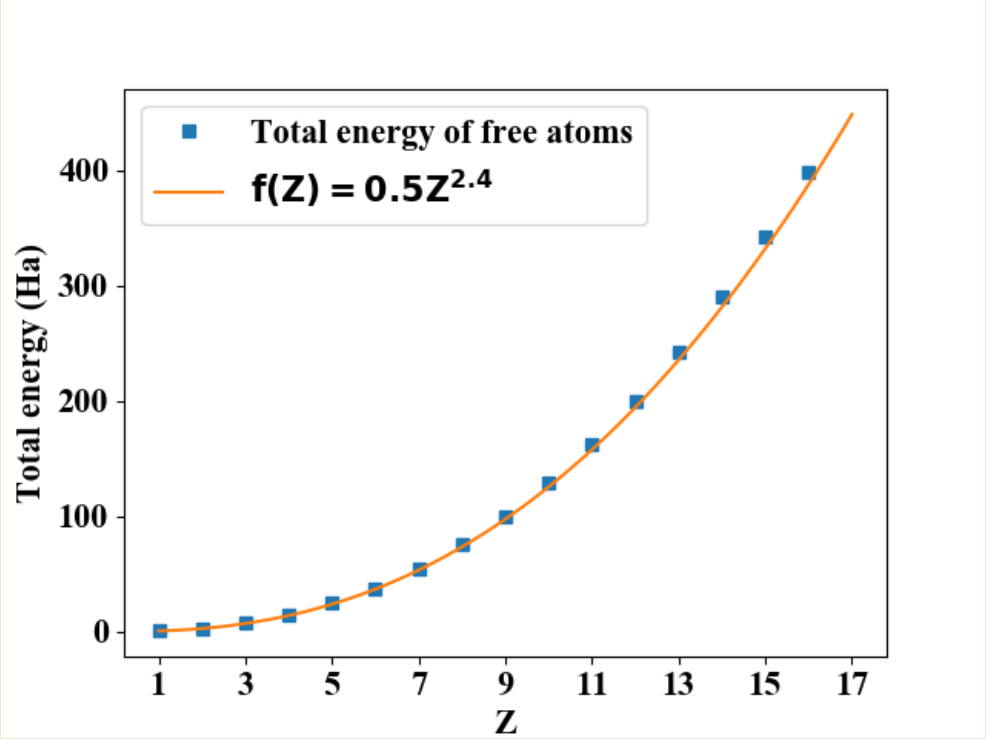
نمایش داده‌ها به صورت موثر (به منظور پیش بینی انرژی پیوند مولکولی)؟

ماتریس کولنی

$$\mathbf{M}_{ij} = \begin{cases} \boxed{0.5Z_i^{2.4}} & i = j \\ \dfrac{Z_i Z_j}{\| \mathbf{R}_i - \mathbf{R}_j \|} & i \neq j \end{cases}$$

انرژی کل اتم آزاد با عدد اتمی Z_i

ماتریس نسبت به چرخش و جابه‌جایی مختصات ناوردا است



هسته‌ی گوسی
(Gaussian Kernel)

در مسئله‌ی مورد بررسی (پیش بینی انرژی پیوندی مولکولی)، از هسته‌ی گوسی استفاده می‌کنیم:

$$k(\mathbf{x},\mathbf{z}) = \exp\left(-\frac{\|\mathbf{x}-\mathbf{z}\|^2}{2\sigma^2}\right)$$

آیا این تابع را می‌توان بر حسب ضرب داخلی در یک فضای ویژگی نوشت؟

$$k(\mathbf{x},\mathbf{z}) \overset{?}{=} \langle \phi(\mathbf{x}), \phi(\mathbf{z}) \rangle$$

برای ساده کردن مسئله فرض می‌کنیم $2\sigma^2 = 1$

$$\exp\left(-\frac{\|\mathbf{x}-\mathbf{z}\|^2}{2\sigma^2}\right) = \exp(-\|\mathbf{x}\|^2)\exp(-\|\mathbf{z}\|^2)\underbrace{\exp(2\|\mathbf{x}\|\|\mathbf{z}\|)}_{} = \exp(-\|\mathbf{x}\|^2)\exp(-\|\mathbf{z}\|^2)\sum_{k=0}^{\infty}\underbrace{\frac{2^k\|\mathbf{x}\|^k\|\mathbf{z}\|^k}{k!}}_{}$$

بنابراین اگر تابع نگاشت به صورت زیر تعریف شود (با فضای ویژگی بینهایت):

$$\phi(\mathbf{x}) = \exp(-\|\mathbf{x}\|^2)\left(1, \frac{\sqrt{2^1}}{1!}\|\mathbf{x}\|, \frac{\sqrt{2^2}}{2!}\|\mathbf{x}\|^2, \ldots, \frac{\sqrt{2^k}}{k!}\|\mathbf{x}\|^k, \ldots\right) \in \mathbb{R}^\infty$$

در نتیجه:

$$k(\mathbf{x},\mathbf{z}) = \exp\left(-\frac{\|\mathbf{x}-\mathbf{z}\|^2}{2\sigma^2}\right) = \langle \phi(\mathbf{x}), \phi(\mathbf{z}) \rangle$$

تخمین تشابه بین مولکول‌ها

تشابه در ماتریس کولنی دو مولکول می‌تواند معیار مناسبی از تشابه دو مولکول باشد. به این منظور ماتریس را به یک بردار تبدیل می‌کنیم و فاصله اقلیدسی این بردارها را معیاری از تفاوت دو مولکول در نظر می‌گیرم.

مرحله 1:

ابتدا ماتریس کولنی را بر حسب اندازه برداری (norm) ردیف‌های آن به صورت نزولی مرتبه می‌کنیم. این کار را به گونه‌ای انجام می‌دهیم که در هر جابه‌جایی سطرها، ستون‌ها با همان اندیس‌ها نیز جابه‌جا شود (تا تقارن ماتریس کولنی از بین نرود).

$$
\begin{array}{c c}
 & \begin{array}{ccc} H_1 & C & H_2 \end{array} \\
\begin{array}{c} H_1 \\ C \\ H_2 \end{array} &
\begin{pmatrix}
\mathbf{M}_{H_1H_1} & \mathbf{M}_{H_1C} & \mathbf{M}_{H_1H_2} \\
\mathbf{M}_{CH_1} & \mathbf{M}_{CC} & \mathbf{M}_{CH_2} \\
\mathbf{M}_{H_2H_1} & \mathbf{M}_{H_2C} & \mathbf{M}_{H_2H_2}
\end{pmatrix}
\end{array}
\xrightarrow{\text{جابه‌جایی سطر اول و دوم}}
\begin{pmatrix}
\mathbf{M}_{CH_1} & \mathbf{M}_{CC} & \mathbf{M}_{CH_2} \\
\mathbf{M}_{H_1H_1} & \mathbf{M}_{H_1C} & \mathbf{M}_{H_1H_2} \\
\mathbf{M}_{H_2H_1} & \mathbf{M}_{H_2C} & \mathbf{M}_{H_2H_2}
\end{pmatrix}
\xrightarrow{\text{جابه‌جایی ستون اول و دوم}}
\begin{array}{c c}
 & \begin{array}{ccc} C & H_1 & H_2 \end{array} \\
\begin{array}{c} C \\ H_1 \\ H_2 \end{array} &
\begin{pmatrix}
\mathbf{M}_{CC} & \mathbf{M}_{CH_1} & \mathbf{M}_{CH_2} \\
\mathbf{M}_{H_1C} & \mathbf{M}_{H_1H_1} & \mathbf{M}_{H_1H_2} \\
\mathbf{M}_{H_2C} & \mathbf{M}_{H_2H_1} & \mathbf{M}_{H_2H_2}
\end{pmatrix}
\end{array}
$$

دلیل مرتب کردن: اگر دو مولکول کاملا یکسان یا بسیار شبیه به هم در داده‌ها وجود داشته باشد، با مرتب کردن یکسان بودن آنها در ماتریس کولنی به راحتی قابل تشخیص است. در واقع با این کار ناوردایی ماتریس کولنی، نسبت به اندیس گذاری اتم‌ها، اعمال می‌شود.

تخمین تشابه بین مولکول‌ها

مرحله 2:

تبدیل ماتریس به یک ماتریس پایین مثلثی

$$\begin{pmatrix} \mathbf{M}_{CC} & \mathbf{M}_{CH_1} & \mathbf{M}_{CH_2} \\ \mathbf{M}_{H_1C} & \mathbf{M}_{H_1H_1} & \mathbf{M}_{H_1H_2} \\ \mathbf{M}_{H_2C} & \mathbf{M}_{H_2H_1} & \mathbf{M}_{H_2H_2} \end{pmatrix} \longrightarrow \begin{pmatrix} \mathbf{M}_{CC} & 0 & 0 \\ \mathbf{M}_{H_1C} & \mathbf{M}_{H_1H_1} & 0 \\ \mathbf{M}_{H_2C} & \mathbf{M}_{H_2H_1} & \mathbf{M}_{H_2H_2} \end{pmatrix}$$

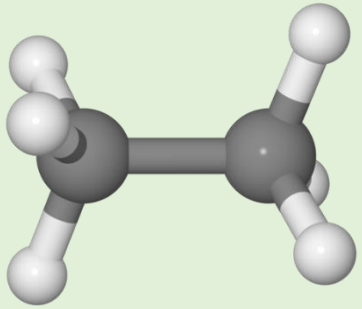
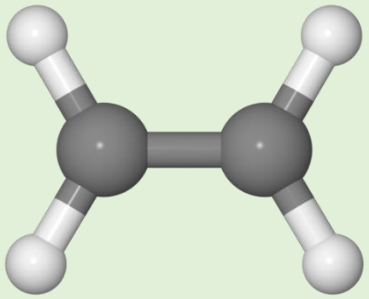
مرحله 3:

تبدیل ماتریس پایین مثلثی به یک بردار

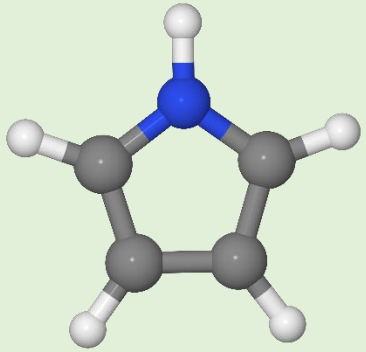
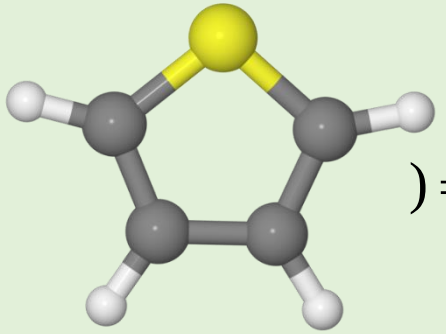
$$\begin{pmatrix} \mathbf{M}_{CC} & 0 & 0 \\ \mathbf{M}_{H_1C} & \mathbf{M}_{H_1H_1} & 0 \\ \mathbf{M}_{H_2C} & \mathbf{M}_{H_2H_1} & \mathbf{M}_{H_2H_2} \end{pmatrix} \longrightarrow (\mathbf{M}_{CC} \quad \mathbf{M}_{H_1C} \quad \mathbf{M}_{H_1H_1} \quad \mathbf{M}_{H_2C} \quad \mathbf{M}_{H_2H_1} \quad \mathbf{M}_{H_2H_2})$$

تخمین تشابه بین مولکول‌ها

مثال: فاصله اقلیدسی بین دو مولکول C_2H_4 و C_2H_6

$d(\mathbf{M}, \mathbf{M}') = d($  ,  $) = 4.986$

مثال: فاصله اقلیدسی بین دو مولکول C_4NH_5 و C_4SH_4

$d(\mathbf{M}, \mathbf{M}') = d($  ,  $) = 335.831$

تخمین تشابه بین مولکول‌ها با استفاده از هسته‌ی گوسی

با استفاده از تابع گوسی می‌توان تشابه را به صورت واضح‌تری به صورت عددی نشان داد:

$$d(\mathbf{M},\mathbf{M}') \approx 0 \Rightarrow \exp(-\frac{d(\mathbf{M},\mathbf{M}')^2}{2\sigma^2}) \approx 1$$

$$d(\mathbf{M},\mathbf{M}') \gg \sigma \Rightarrow \exp(-\frac{d(\mathbf{M},\mathbf{M}')^2}{2\sigma^2}) \approx 0$$

تخمین انرژی پیوندی با استفاده از هسته‌ی گوسی

$$E^{\text{est}}(\mathbf{M}) = \sum_{i=1}^{N} \alpha_i \exp\left[-\frac{d(\mathbf{M},\mathbf{M}_i)^2}{2\sigma^2}\right]$$

$$\min_{\alpha} \sum_i (E^{\text{est}}(\mathbf{M_i}) - E_i^{\text{ref}})^2 + \lambda \sum_i \alpha_i^2$$

$$\alpha = (\mathbf{K} + \lambda \mathbf{I})^{-1} \mathbf{E}^{\text{ref}}$$

$$K_{ij} = \exp[-d(\mathbf{M}_i, \mathbf{M}_j)^2 / (2\sigma^2)]$$

Kernel Ridge Regression (KRR)

$$g(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle = \sum_{i=1}^{l} \alpha_i k(\mathbf{x}_i, \mathbf{x})$$

$$\min_{\mathbf{w}} \mathcal{L}_\lambda(\mathbf{w}, S) = \min_{\mathbf{w}} \lambda \| \mathbf{w} \|^2 + \sum_{i=1}^{l} (y_i - g(\mathbf{x_i}))^2$$

$$\alpha = (\mathbf{G} + \lambda \mathbf{I}_l)^{-1} \mathbf{y}$$

$$\mathbf{G}_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$$

Kernel Ridge Regression (KRR)

M. Rupp et. al, PRL 108, 058301 (2012)

حل $(\mathbf{G}+\lambda\mathbf{I}_l)\alpha=\mathbf{y}$ به روش تجزیه چولسکی

معادله $(\mathbf{G}+\lambda\mathbf{I}_l)\alpha=\mathbf{y}$ در واقع یک معادله خطی به صورت $\mathbf{Ax}=\mathbf{b}$ است که چهار روش برای حل آن می‌توان در نظر گرفت:

1- محاسبه وارون $\mathbf{A}$
2-روش حذف گوسی
3-روش تجزیه $\mathbf{A}$ به ماتریس‌های بالا مثلثی و پایین مثلثی (LU decomposition)
4-روش تجزیه چولسکی (cholesky decomposition)

مقایسه سرعت این روش‌ها:

روش 1 > روش 2 > روش 3 > روش 4

روش حل چولسکی

شرط استفاده از روش چولسکی

1- ماتریس $\mathbf{A}$ باید متقارن باشد

2- ماتریس متقارن $\mathbf{A}$ یک ماتریس معین مثبت (positive definite) باشد یعنی برای هر بردار $\mathbf{x} \neq 0$ $\quad \mathbf{x}^{\mathsf{T}}\mathbf{A}\mathbf{x} > 0$

تجزیه یک ماتریس معین مثبت

می‌توان نشان داد که یک ماتریس معین مثبت را می‌توان به صورت ضرب یک ماتریس پایین مثلثی (یا بالا مثلثی) در ترانهاده‌اش نوشت:

$$\mathbf{A} = \mathbf{L}\mathbf{L}^{\mathsf{T}}$$

روش تجزیه چولسکی

$$\mathbf{A}\mathbf{x} = \mathbf{b} \qquad \mathbf{L}\underbrace{\mathbf{L}^{\mathsf{T}}\mathbf{x}}_{\mathbf{z}} = \mathbf{b} \qquad \begin{cases} \mathbf{L}\mathbf{z} = \mathbf{b} \\ \mathbf{L}^{\mathsf{T}}\mathbf{x} = \mathbf{z} \end{cases}$$

به دلیل اینکه ماتریس L یک ماتریس پایین مثلثی و ترانهاده‌ی آن یک ماترس بالا مثلثی حل معادله اول و دوم فقط با جایگزینی انجام می‌شود.

می‌توان نشان داد که ماتریس $\mathbf{K}$ و در نتیجه ماتریس $\mathbf{K} + \lambda\mathbf{I}$ ماتریس معین مثبت است.

پیش بینی در روش حل دوگانه

پیش بینی برای داده‌های جدید $(\mathbf{x}')$

$$g(\mathbf{x}') = \sum_{i=1}^{l} \alpha_i k(\mathbf{x}_i, \mathbf{x}')$$

اگر برای m داده بخواهیم پیش بینی انجام دهیم می‌توان داده‌ها را به صورت ماتریسی زیر در نظر گرفت:

$$\mathbf{X}' \in \mathbb{R}^{m \times n} \qquad \mathbf{X}' = \begin{pmatrix} \boxed{\mathbf{x}_1'} \\ \mathbf{x}_2' \\ \vdots \\ \mathbf{x}_m' \end{pmatrix}$$

بردار n مولفه‌ای

اگر ماتریس $\mathbf{L}$ را به صورت زیر تعریف کنیم:

$$\mathbf{L}_{ij} = k(\mathbf{x}_i, \mathbf{x}_j')$$

در نتیجه نتایج پیش بینی را می‌توان با استفاده از ضرب ماتریس زیر بدست آورد:

$$g(\mathbf{X}') = \mathbf{L}^{\mathrm{T}} \alpha$$

بخش عملی

سیمای کلی برنامه

خواندن داده‌های 7102 مولکول آلی از فایل "dsgdb7ae2.xyz"

↓

جدا کردن 1000 مولکول به عنوان مجموعه‌ی آموزشی (training set)

↓

تشکیل ماتریس کولنی و سپس بردار کولنی برای تمام مولکول‌ها

↓

تشکیل ماتریس هسته $(\mathbf{K})$ برای مجموعه‌ی آموزشی

↓

حل معادله $(\mathbf{K}+\lambda\mathbf{I})\alpha=\mathbf{E}^{\mathrm{ref}}$

↓

تخمین

$$E^{\mathrm{est}}(\mathbf{M})=\sum_{i=1}^{N}\alpha_i\exp\left[-\frac{d(\mathbf{M},\mathbf{M}_i)^2}{2\sigma^2}\right]$$

خواندن داده‌های 7102 مولکول آلی از فایل "dsgdb7ae2.xyz"

تعداد اتم‌ها در هر مولکول

انرژی پیوندی مولکول بر حسب kcal/mol

اندیس مولکول

```
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
8
0002 -711.117
C    0.99571000  0.01149000 -0.09922000   0.99591400  0.01151100 -0.09922100
C    2.51489000  0.01148000 -0.09922000   2.51468600  0.01146600 -0.09922600
H    0.61911000  0.74910000 -0.83887000   0.59725900  0.72987700 -0.81959600
H    0.61911000  0.28325000  0.90938000   0.59725900  0.27617000  0.88310600
H    0.61909000 -0.99785000 -0.36818000   0.59727800 -0.97153100 -0.36116700
H    2.89151000  1.02083000  0.16973000   2.91332200  0.99450900  0.16271900
H    2.89149000 -0.26027000 -1.10783000   2.91334100 -0.25319200 -1.08155300
H    2.89149000 -0.72612000  0.64042000   2.91334100 -0.70690000  0.62114800
....
....
```

مختصات بهینه شده در DFT

مختصات تعادلی اتم‌ها (xyz) در هر مولکول با استفاده از میدان نیرو بر حسب آنگستروم

خواندن داده‌های 7102 مولکول آلی از فایل "dsgdb7ae2.xyz"

1-مختصات (مربوط به میدان نیرو)، نوع اتم‌ها، انرژی اتم و تعداد اتم‌ها باید از فایل " dsgdb7ae2.xyz " خوانده شوند

2-می‌توان به جای مختصات به دست آمده از میدان نیرو از مختصات که در DFT بدست آمده است استفاده کرد منتها به منظور سازگاری تا آخر محاسبات باید از مختصات DFT استفاده کرد.

3-اندیس گذاری‌ها در برخی مواقع درست انجام نشده است (مثلا اندیس 98 وجود ندارد) بنابراین بهتر است اندیس گذاری را خودمان از ابتدا انجام دهیم.

خواندن داده‌های 7102 مولکول آلی از فایل "dsgdb7ae2.xyz"

```
class Molecule:
    """
    Molecule class contains following properties a molecule:
    n:    number of atoms in the molecule
    ind:  index of molecule
    en:   atomization energy of the molecule
    atom: type of atoms in the molecule
    pos:  position (force field and DFT) of atoms in the molecule
    nonh: number of non-H atoms in the molecule
    """
    def __init__(self,n,ind, en,atom, pos,nonh):
        self.n=n
        self.ind=ind
        self.en=en
        self.atom=atom
        self.pos=pos
        self.nonh=nonh
    def __lt__(self, other):
       return self.n < other.n
```

برای خوانا بودن و راحتی برنامه نویسی
یک کلاس به اسم Molecule ایجاد می‌کنیم

خواندن داده‌های 7102 مولکول آلی از فایل "dsgdb7ae2.xyz"

```
molecule=[]   # A list to save properties of each molecule
dataxyz=open("dsgdb7ae2.xyz","r")

while True:
    line = dataxyz.readline()
    if not line:
        break
    pos=[]
    atom=[]
    words=line.split()
    if len(words)==1:
      n=int(words[0])
      line = dataxyz.readline()
      words=line.split()
      ind=int(words[0])
      ae=float(words[1])
      for i in range(n):
        line = dataxyz.readline()
        words=line.split()
        atom.append(words[0])
        pos.append(words[1:])
    m=Molecule(n,ind,ae,atom,pos,1)
    molecule.append(m)
dataxyz.close()
```

در این قسمت از برنامه یک متغیر لیست به اسم molecule ایجاد می‌کنیم و با خواندن داده‌ها و سپس ایجاد نمونه (instance) از کلاس Molecule برای هر مولکول آن را در لیست molecule ذخیره می‌کنیم.

ایجاد نمونه از مولکول

اضافه کردن آن به لیست مولکول‌ها

خواندن داده‌های 7102 مولکول آلی از فایل "dsgdb7ae2.xyz"

تصحیح اندیس مولکول‌ها

```
N=len(molecule)
print("Number of molecule=", N)
#some index of molecules are not correct (for example  there is a wrong index for 89th molecule)
for j in range(N):
  # correct index of molecules (re-indexing)
  molecule[j].ind=j
```

خواندن داده‌های 7102 مولکول آلی از فایل "dsgdb7ae2.xyz"

مقدار دهی به پارامتر nh که تعداد اتم‌های غیر از هیدروژن را نشان می‌دهد.

```
for j in range(N):
  nh=0
  n=molecule[j].n
  for i in range(n):
    if molecule[j].atom[i]=='H':
        nh=nh+1
  molecule[j].nonh=molecule[j].n-nh

#backup the molecules list
molecule_backup=molecule.copy()
```

جدا کردن 1000 مولکول به عنوان مجموعه‌ی آموزشی (training set)

ابتدا برای تمرین، جدول 3 مقاله را بدست می‌آوریم:

Table 3. *Distribution of molecular size* in the provided dataset, measured by number of non-H atoms.

non-H atoms	1	2	3	4	5	6	7	$\sum$
molecules	1	3	12	43	157	935	5951	7102

جدا کردن 1000 مولکول به عنوان مجموعه‌ی آموزشی (training set)

```
n1=0;n2=0;n3=0;n4=0;n5=0;n6=0;n7=0
for j in range(N):
    if (molecule[j].nonh) == 1:
        n1=n1+1
    if (molecule[j].nonh) == 2:
        n2=n2+1
    if (molecule[j].nonh) == 3:
        n3=n3+1
    if (molecule[j].nonh) == 4:
        n4=n4+1
    if (molecule[j].nonh) == 5:
        n5=n5+1
    if (molecule[j].nonh) == 6:
        n6=n6+1
    if (molecule[j].nonh) == 7:
        n7=n7+1
print("Table 3:")
print("non-H atoms", " 1","2","3"," 4"," 5","  6","  7")
print("n moulecules", n1,n2,n3,n4,n5,n6,n7)
```

باز تولید جدول 3 مقاله

جدا کردن 1000 مولکول به عنوان مجموعه‌ی آموزشی (training set)

در حالتی که داده‌ها همگن باشند، ایده‌آل انتخات تصادفی از بین داده‌ها برای ایجاد مجموعه‌ی آموزشی است ولی همانطور که از جدول 3 مشخص است تعداد مولکول‌هایی که تعداد اتم‌های غیر از هیدروژن آنها 4 و کمتر از 4 است بسیار کم است (برابر با 59 مولکول).
بنابراین ابتدا این 59 مولکول را به مجموعه آموزشی اضافه می‌کنیم

```
molecule_t=[]
ind_train=[]   # a list to save index of training set

#first we select all of molecules which have 4 to 1 non-H atoms
for j in range(N):
    if molecule[j].nonh <= 4:
        molecule_t.append(molecule[j])
        ind_train.append(molecule[j].ind)

#remove these molecule from the main list
k=len(molecule_t)
for j in range(k):
    molecule.remove(molecule_t[j])
```

لیست اندیس مولکول‌های مجموعه‌ی آموزشی

اضافه کردن اندیس‌های مولکول‌هایی که تعداد اتم‌های غیر هیدروژن آنها کمتر از 5 است به لیست اندیس مولکول‌های مجموعه‌ی آموزشی.

تعداد مولکول‌هایی که تعداد اتم‌های غیر هیدروژن آنها کمتر از 5 است.

جدا کردن 1000 مولکول به عنوان مجموعه‌ی آموزشی (training set)

```
# sort remaining molecules by number of atoms:
molecule.sort()
#select molecules by a step=(N-k)/(1000-k) ~8
step=(N-k)/(1000-k)
select_list=np.round(np.arange(0,N-k,step))
select_list=select_list.astype(int)
for j in select_list:
    molecule_t.append(molecule[j])
    ind_train.append(molecule[j].ind)

#remove these molecule from the main list
k1=len(molecule_t)
for j in range(k,k1):
    molecule.remove(molecule_t[j])

# save remaining molecules to a new list
molecule_out=molecule.copy()

ind=list(range(N))
for j in ind_train:
   ind.remove(j)
ind_predict=ind      # a list of remaining molecule as the prediction set
```

منظم کردن مولکول‌های باقیمانده بر حسب تعداد اتم‌هایشان

انتخاب مولکول‌ها باقیمانده (59–7102) با گامی به اندازه step

تشکیل یک لیست از اندیس مولکول‌های باقیمانده (1000–7102) به منظور استفاده برای پیش بینی انرژی پیوندی و در نتیجه آزمودن روش یادگیری ماشین به کار گرفته شده

تشکیل ماتریس کولنی و سپس بردار کولنی برای تمام مولکول‌ها

```
del molecule

molecule=molecule_backup.copy()

natoms=[]
for i in range(N):
  natoms.append(molecule[i].n)
nmax=max(natoms) # the maximum number of atoms in a molecule
del natoms
xyz=np.zeros([N,nmax,3])
Z=np.zeros([N,nmax])
CM=np.zeros([N,nmax,nmax])
```

آرایه برای ذخیره سازی مکان‌های اتم‌های مولکول‌ها

آرایه برای ذخیره سازی عدد اتمی اتم‌های هر مولکول

آرایه برای ذخیره سازی ماتریس کولنی

N: تعداد کل مولکول‌ها
Nmax: بیشینه تعداد اتم‌ها در یک مولکول
3: برای ذخیره مولفه‌های x,y,z

تشکیل ماتریس کولنی و سپس بردار کولنی برای تمام مولکول‌ها

تبدیل نمادهای اتمی به عدد اتمی

```
def atomicnumber(name):
  """
  convert atom name to its atomic number
  """
  if name=='H':
    return 1
  if name=='C':
    return 6
  if name=='N':
    return 7
  if name=='O':
    return 8
  if name=='S':
    return 16
```

تشکیل ماتریس کولنی و سپس بردار کولنی برای تمام مولکول‌ها

```
for j in range(N):
  for i in range(molecule[j].n):
      Z[j,i]=atomicnumber(molecule[j].atom[i])
      xyz[j,i,0]=float(molecule[j].pos[i][0])
      xyz[j,i,1]=float(molecule[j].pos[i][1])
      xyz[j,i,2]=float(molecule[j].pos[i][2])
      #xyz[j,i,0]=float(molecule[j].pos[i][3])   #dft positions
      #xyz[j,i,1]=float(molecule[j].pos[i][4])   #dft positions
      #xyz[j,i,2]=float(molecule[j].pos[i][5])   #dft positions
ang2bohr=1.8897261
xyz=xyz*ang2bohr
for j in range(N):
  for i in range(molecule[j].n):
    for k in range(molecule[j].n):
      if i==k:
        CM[j,i,k]=0.5*Z[j,i]**2.4
      else:
        CM[j,i,k]=Z[j,i]*Z[j,k]/la.norm(xyz[j,i,:]-xyz[j,k,:])
```

تشکیل ماتریس کولنی

تشکیل ماتریس کولنی و سپس بردار کولنی برای تمام مولکول‌ها

مرتب کردن ماتریس کولنی براساس نرم ردیف‌های (و هم‌زمان مرتب کردن ستون‌ها برای حفظ تقارن ماتریس) به صورت نزولی

مرتب کردن اندیس سطرها به حسب اندازه نرم آنها (به صورت صعودی)

معکوس کردن ترتیب در اندیس‌ها

```
for j in range(N):
    CMj=CM[j,:,:]
    indexlist = np.argsort(la.norm(CMj,axis=1))
    indexlist_r=indexlist[::-1]
    sym_sortedCM = CMj[indexlist_r][:,indexlist_r]
    CM[j,:,:]=sym_sortedCM
```

مرتب کردن ماتریس بر حسب نرم ردیف‌ها (سطرها) به صورت نزولی

سپس ستون‌های ماتریس که سطرهای آن مرتب شده است را به صورت نزولی مرتب می‌کنیم

https://stackoverflow.com/questions/36885418/sort-symmetric-numpy-2d-array-by-function-norm

تشکیل ماتریس کولنی و سپس بردار کولنی برای تمام مولکول‌ها

تشکیل بردار یک بعدی از روی ماتریس کولنی

```
I=[]
J=[]
for i in range(nmax):
  for j in range(i+1):
    I.append(i)
    J.append(j)
indices=(I,J)

CM_j=CM[0,:,:]
CM_vec=CM_j[indices]
cm_vec_size=CM_vec.size # (23+22+21+20+...+1=276 or
nmax*(nmax+1)/2)
CM_vec=np.zeros([N,cm_vec_size])

for j in range(N):
   CM_j=CM[j,:,:]
   CM_vec[j,:]=CM_j[indices]
```

تشکیل ماتریس هسته (K) برای مجموعه‌ی آموزشی

محاسبه فاصله بین بردارها برای استفاده در محاسبه ماتریس هسته

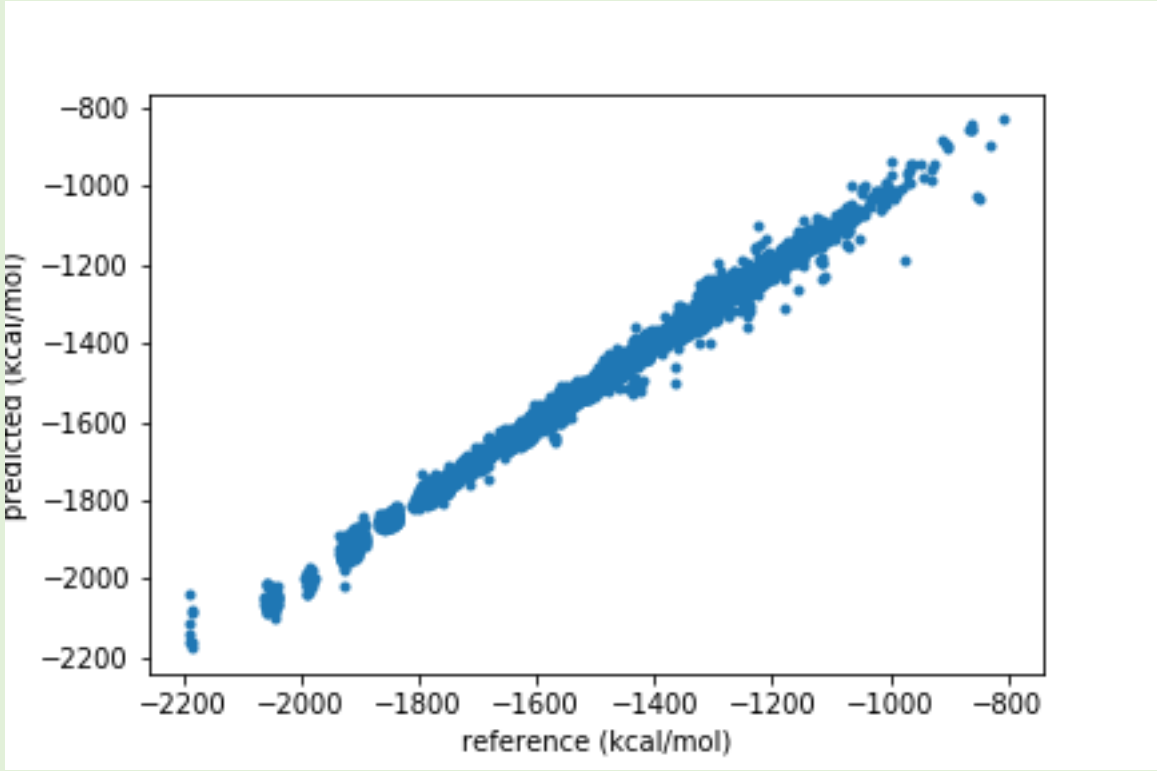
```
CM_vec_d=np.zeros([N,N])
for j in range(N):
    for k in range(j):
        CM_vec_d[j,k]=la.norm(CM_vec[j,:]-CM_vec[k,:])
        CM_vec_d[k,j]=CM_vec_d[j,k]
```

ادامه محاسبات به عنوان تمرین

نتایجی که باید بدست آورید:

1- نمودارپیش بینی شده‌ی انرژی پیوندی مولکول‌ها (که توسط یادگیر ماشین بدست آمده) بر حسب داده‌های انرژی پیوندی مولکول‌ها

2-محاسبه RMSD که باید مقداری برابر با 17.559 بدست آید.



ادامه محاسبات به عنوان تمرین

یک نکته:

اصولا باید برای پارامترهای σ و λ (که در مقاله به عنوان hyperparameters شناخته می‌شوند) بهینه سازی انجام شود که برای این بهینه سازی از یک مجموعه‌ی کوچک‌تر (به نام Hold-out set) استفاده می‌شود. ما در اینجا فقط از نتایج بهینه شده‌ای که در مقاله بدست آمده است استفاده می‌کنیم

```
lambda_p=2.0**(-22)  # is about 2.38418e-07 ~ 10e-6.5
sigma_p=2.0**(9.5)
```

منابع:

1- مقالات:

1-M Rupp, International Journal of Quantum Chemistry 115 (16), 1058-1073

2-M. Rupp et. al, Physical review letter 108, 058301 (2012)

2-کتاب:

1- John Shawe-Taylor, Kernel Methods for Pattern Analysis (2011)

3-سخنرانی:

1-Yaser Abu Mostafa, Learning from Data
2-Andrew Ng, Machine Learning

با تشکر فراوان