Lecture 24
Kernel Methods and Gaussian Processes

Bayesian linear models; kernel; Gaussian processes; case study

Prof. David A. Kofke
CE 500 – Modeling Potential-Energy Surfaces
Department of Chemical & Biological Engineering
University at Buffalo

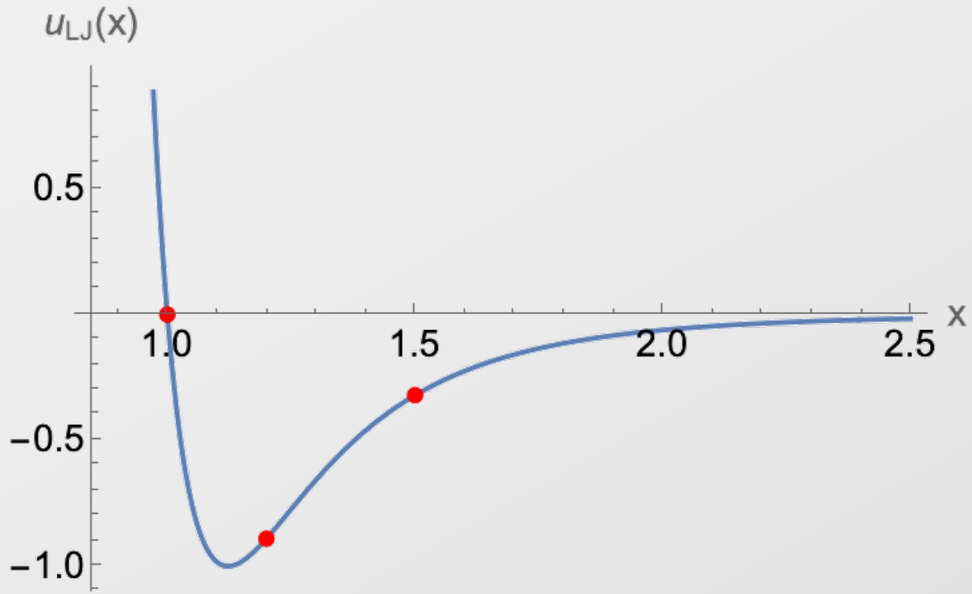
University at Buffalo
The State University of New York

Overview of development

- Given X, y data for quantity of interest
- Linear model in input variable (r)
  - Develop probabilities of weights w from prior and data, $p(w|X,y)$
  - Evaluate f_* via
- Project into feature space (introduce basis functions), $\phi(x)$
  - Follow same overall procedure as model in linear inputs
- Recognize that "dot products" of feature vectors is all that matters
- Develop approach that goes straight to modeling the dot product, $k(x, x')$

We will walk through the major elements using a simple, specific example

- Formulate a model that reproduces the Lennard-Jones potential based on 3 data points: $x = 1, 1.2, 1.5$



$$\begin{pmatrix} x^{(1)} \\ x^{(2)} \\ x^{(3)} \end{pmatrix} = \begin{pmatrix} 1.0 \\ 1.2 \\ 1.5 \end{pmatrix}$$

Start with a linear model in terms of the single input parameter *x*

- $u(x) = w_1 + w_2 x$

Input design matrix

$$X = \begin{pmatrix} 1 & x^{(1)} \\ 1 & x^{(2)} \\ 1 & x^{(3)} \end{pmatrix} = \begin{pmatrix} 1 & 1.0 \\ 1 & 1.2 \\ 1 & 1.5 \end{pmatrix}$$

Output

$$\mathbf{y} = \begin{pmatrix} u_{\mathrm{LJ}}(x^{(1)}) \\ u_{\mathrm{LJ}}(x^{(2)}) \\ u_{\mathrm{LJ}}(x^{(3)}) \end{pmatrix} = \begin{pmatrix} 0.00 \\ -0.89 \\ -0.32 \end{pmatrix}$$

For this example, we assume some noise in *y* values: $\sigma_n \equiv 0.1$

Weights (TBD)

$$\mathbf{w} = \begin{pmatrix} w_1 \\ w_2 \end{pmatrix}$$

Select a Gaussian prior with zero mean, zero correlation, variance 10

$$\Sigma_p = \begin{pmatrix} 10 & 0 \\ 0 & 10 \end{pmatrix}$$

- Posterior distribution of weights, via Bayes' rule, is Gaussian

$$p(\mathbf{w} \mid \mathbf{y}, X) \sim \mathcal{N}(\bar{\mathbf{w}}, A^{-1})$$

$$A^{-1} = (\sigma_n^{-2} X^\top X + \Sigma_p^{-1})^{-1} = \left(100. \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1.2 & 1.5 \end{pmatrix} \begin{pmatrix} 1 & 1.0 \\ 1 & 1.2 \\ 1 & 1.5 \end{pmatrix} + \begin{pmatrix} 0.1 & 0 \\ 0 & 0.1 \end{pmatrix} \right)^{-1} = \begin{pmatrix} 0.12 & -0.10 \\ -0.10 & 0.08 \end{pmatrix}$$

$$\bar{\mathbf{w}} = \sigma_n^{-2} A^{-1} X^\top \mathbf{y} = 100 \begin{pmatrix} 0.12 & -0.10 \\ -0.12 & 0.08 \end{pmatrix} \begin{pmatrix} 1 & 1 & 1 \\ 1.0 & 1.2 & 1.5 \end{pmatrix} \begin{pmatrix} 0.00 \\ -0.89 \\ -0.32 \end{pmatrix} = \begin{pmatrix} 0.13 \\ -0.10 \end{pmatrix}$$

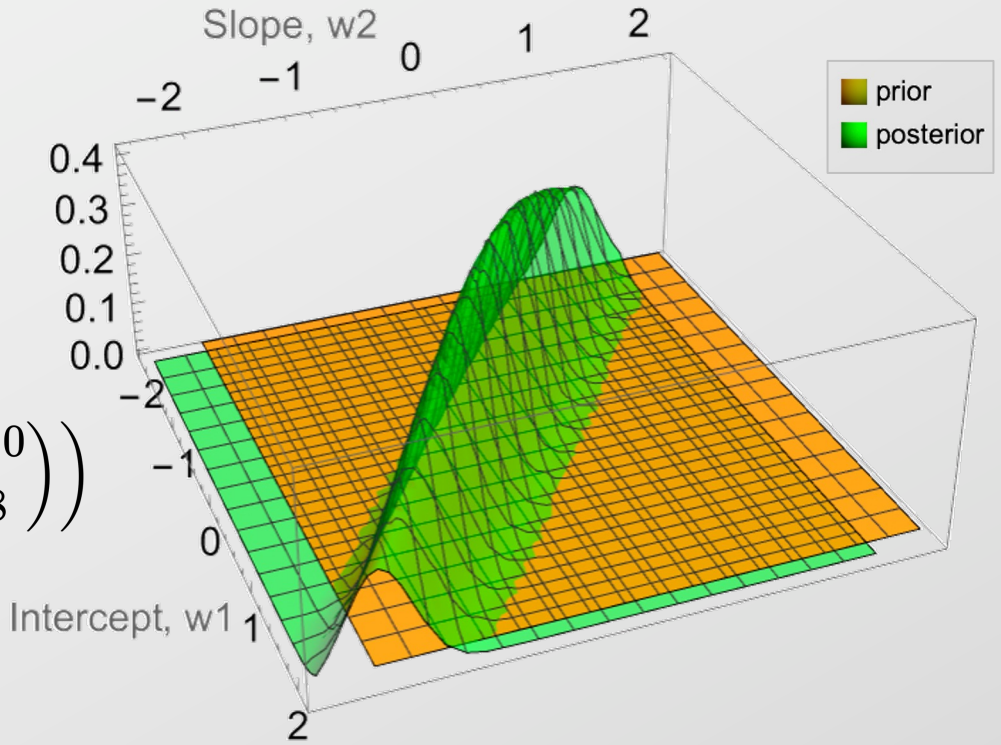
The posterior distribution of weights shows a strong correlation between slope and intercapt

- Prior

$$p(\mathbf{w}) \sim \mathcal{N}\left(\begin{pmatrix}0\\0\end{pmatrix}, \begin{pmatrix}10 & 0\\0 & 10\end{pmatrix}\right)$$

- Posterior

$$p(\mathbf{w} \mid \mathbf{y}, X) \sim \mathcal{N}\left(\begin{pmatrix}0.13\\-0.10\end{pmatrix}, \begin{pmatrix}0.12 & -0.10\\-0.10 & 0.08\end{pmatrix}\right)$$



The estimated from the fit at any point is given as a Gaussian distribution

- Average the output from all possible linear models w.r.t. the Gaussian posterior

$$p(f_* \mid \mathbf{x}_*, X, \mathbf{y}) = \int p(f_* \mid \mathbf{x}_*, \mathbf{w}) p(\mathbf{w} \mid \mathbf{y}, X) d\mathbf{w}$$

Gaussian with stdev σ_n

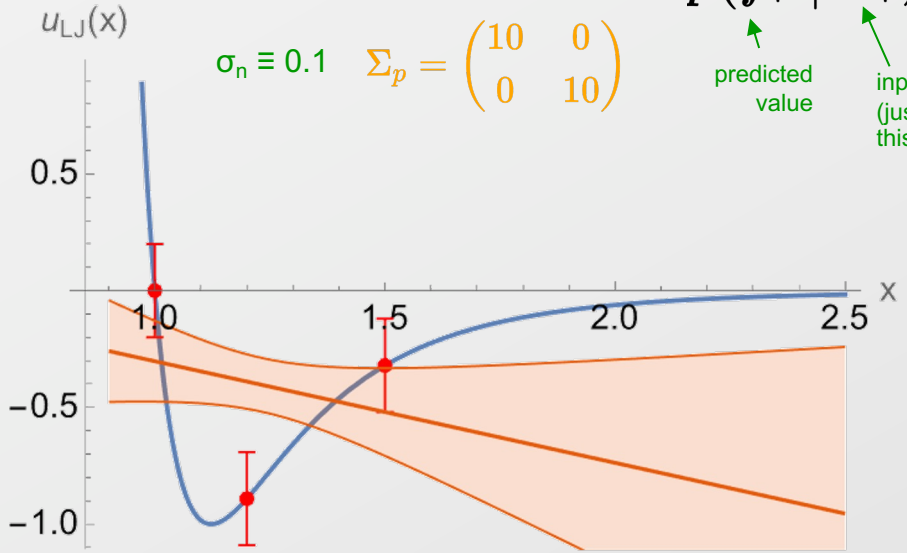
From previous slide

predicted value

input vector (just a scalar for this example)

$$= \mathcal{N}(\underbrace{\mathbf{x}_*^\top \bar{\mathbf{w}}}_{\text{mean}}, \underbrace{\mathbf{x}_*^\top A^{-1} \mathbf{x}_*}_{\text{Variance}})$$

depends on x*



All uncertainties represented as 2 standard deviations

$$\mathcal{N}(0.13 - 0.44x, 012 - 0.19x + 0.08x^2)$$

Explore effects of parameters in Mathematica…

We can get a more effective model by projecting into feature space (i.e., adopting a basis)

- Add just one more function: $u(x) = w_1 x^{-12} + w_2 + w_3 x$

$$\phi(x) = \begin{pmatrix} x^{-12} \\ 1 \\ x \end{pmatrix}$$

Input design matrix

$$\Phi = \begin{pmatrix} (x^{(1)})^{-12} & 1 & x^{(1)} \\ (x^{(2)})^{-12} & 1 & x^{(2)} \\ (x^{(3)})^{-12} & 1 & x^{(3)} \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1.0 \\ 0.11 & 1 & 1.2 \\ 0.008 & 1 & 1.5 \end{pmatrix}$$

Output

$$\mathbf{y} = \begin{pmatrix} u_{\mathrm{LJ}}(x^{(1)}) \\ u_{\mathrm{LJ}}(x^{(2)}) \\ u_{\mathrm{LJ}}(x^{(3)}) \end{pmatrix} = \begin{pmatrix} 0.00 \\ -0.89 \\ -0.32 \end{pmatrix}$$

For this example, we assume some noise in *y* values: $\sigma_n \equiv 0.1$

Weights (TBD)

$$\mathbf{w} = \begin{pmatrix} w_1 \\ w_2 \\ w_3 \end{pmatrix}$$

Select a Gaussian prior with zero mean, zero correlation, variance 10

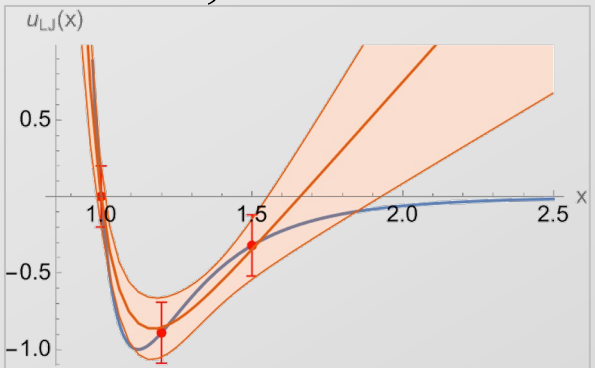
$$\Sigma_p = \begin{pmatrix} 10 & 0 & 0 \\ 0 & 10 & 0 \\ 0 & 0 & 10 \end{pmatrix}$$

- Posterior distribution of weights, via Bayes' rule, is Gaussian

$$p(\mathbf{w} \mid \mathbf{y}, X) \sim \mathcal{N}(\bar{\mathbf{w}}, A^{-1})$$

$$A^{-1} = (\sigma_n^{-2}\Phi^\top\Phi + \Sigma_p^{-1})^{-1} = \begin{pmatrix} 0.06 & -0.15 & 0.11 \\ -0.15 & 0.52 & -0.37 \\ 0.11 & -0.37 & 0.27 \end{pmatrix}$$

$$\bar{\mathbf{w}} = \sigma_n^{-2} A^{-1} X^\top \mathbf{y} = (1.45 \quad -3.67 \quad 2.21)^\top$$



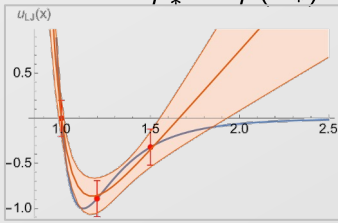
Model predictions can be expressed in terms of an inner product of a modified feature vector

- Here is the probability distribution of the prediction

$$f_* \mid \mathbf{x}_*, X, \mathbf{y} \sim N(\sigma_n^{-2}\boldsymbol{\phi}_*^\top A^{-1}\Phi\mathbf{y}, \boldsymbol{\phi}_*^\top A^{-1}\boldsymbol{\phi}_*)$$

$$\phi(x) = \begin{pmatrix} x^{-12} \\ 1 \\ x \end{pmatrix}$$

$$\boldsymbol{\phi}_* \equiv \boldsymbol{\phi}(\mathbf{x}_*)$$



- This can be written equivalently as

$$K = \Phi\Sigma_p\Phi^\top$$

$$f_* \mid \mathbf{x}_*, X, \mathbf{y} \sim \mathcal{N}\left(\boldsymbol{\phi}_*^\top\Sigma_p\Phi^\top\left(K + \sigma_n^2 I\right)^{-1}\mathbf{y}\right.$$
$$\left.\boldsymbol{\phi}_*^\top\Sigma_p\boldsymbol{\phi}_* - \boldsymbol{\phi}_*^\top\Sigma_p\Phi^\top\left(K + \sigma_n^2 I\right)^{-1}\Phi\Sigma_p\boldsymbol{\phi}_*\right)$$

$$\Sigma_p = \begin{pmatrix} 10 & 0 & 0 \\ 0 & 10 & 0 \\ 0 & 0 & 10 \end{pmatrix}$$

- Everything is in terms of $\phi\ \Sigma_p\ \phi$

$$\Phi = \begin{pmatrix} (x^{(1)})^{-12} & 1 & x^{(1)} \\ (x^{(2)})^{-12} & 1 & x^{(2)} \\ (x^{(3)})^{-12} & 1 & x^{(3)} \end{pmatrix} = \begin{pmatrix} \phi(x^{(1)})^\top \\ \phi(x^{(2)})^\top \\ \phi(x^{(3)})^\top \end{pmatrix}$$

$$\Phi^\top = \begin{pmatrix} \phi(x^{(1)}) & \phi(x^{(2)}) & \phi(x^{(3)}) \end{pmatrix}$$

- Modified feature vector

$$\boldsymbol{\psi}(\mathbf{x}) \equiv \Sigma_p^{1/2}\boldsymbol{\phi}(\mathbf{x}) = \sqrt{10}\begin{pmatrix} x^{-12} & 1 & x \end{pmatrix}^\top$$

Model predictions can be expressed in terms of an inner product of a modified feature vector

$$f_* \mid \mathbf{x}_*, X, \mathbf{y} \sim \mathcal{N}\Big(\boldsymbol{\phi}_*^\top \Sigma_p \Phi^\top \big(K + \sigma_n^2 I\big)^{-1} \mathbf{y} \quad K = \Phi \Sigma_p \Phi^\top$$

$$\boldsymbol{\phi}_*^\top \Sigma_p \boldsymbol{\phi}_* - \boldsymbol{\phi}_*^\top \Sigma_p \Phi^\top \big(K + \sigma_n^2 I\big)^{-1} \Phi \Sigma_p \boldsymbol{\phi}_*\Big)$$

- It's all dot products

$$\psi(\mathbf{x}) = \begin{pmatrix} \sqrt{10}\, x^{-12} \\ \sqrt{10} \\ \sqrt{10}\, x \end{pmatrix} \qquad \boldsymbol{\psi}(x) \cdot \boldsymbol{\psi}(x') = 10 + 10(xx')^{-12} + 10xx'$$

$$\boldsymbol{\phi}_*^\top \Sigma_p \Phi^\top = \begin{pmatrix} \boldsymbol{\psi}(x_*) \cdot \boldsymbol{\psi}(x^{(1)}) & \boldsymbol{\psi}(x_*) \cdot \boldsymbol{\psi}(x^{(2)}) & \boldsymbol{\psi}(x_*) \cdot \boldsymbol{\psi}(x^{(3)}) \end{pmatrix}$$

$$\boldsymbol{\phi}_*^\top \Sigma_p \boldsymbol{\phi}_*^\top = \boldsymbol{\psi}(x_*) \cdot \boldsymbol{\psi}(x_*)$$

$$K = \begin{pmatrix} \boldsymbol{\psi}(x^{(1)}) \cdot \boldsymbol{\psi}(x^{(1)}) & \boldsymbol{\psi}(x^{(1)}) \cdot \boldsymbol{\psi}(x^{(2)}) & \boldsymbol{\psi}(x^{(1)}) \cdot \boldsymbol{\psi}(x^{(3)}) \\ \boldsymbol{\psi}(x^{(2)}) \cdot \boldsymbol{\psi}(x^{(1)}) & \boldsymbol{\psi}(x^{(2)}) \cdot \boldsymbol{\psi}(x^{(2)}) & \boldsymbol{\psi}(x^{(2)}) \cdot \boldsymbol{\psi}(x^{(3)}) \\ \boldsymbol{\psi}(x^{(3)}) \cdot \boldsymbol{\psi}(x^{(1)}) & \boldsymbol{\psi}(x^{(3)}) \cdot \boldsymbol{\psi}(x^{(2)}) & \boldsymbol{\psi}(x^{(3)}) \cdot \boldsymbol{\psi}(x^{(3)}) \end{pmatrix}$$

$$\begin{pmatrix} x^{(1)} \\ x^{(2)} \\ x^{(3)} \end{pmatrix} = \begin{pmatrix} 1.0 \\ 1.2 \\ 1.5 \end{pmatrix}$$

We can bypass the features and weights prior (Σ_p): Define model directly in term of the inner product

- "kernel trick" $\quad \psi(x) \cdot \psi(x') \rightarrow k(x, x')$

$$f_* \mid \mathbf{x}_*, X, \mathbf{y} \sim \mathcal{N}\Big(\boxed{\phi_*^\top \Sigma_p \Phi^\top}\left(K + \sigma_n^2 I\right)^{-1}\mathbf{y}$$

$$\boxed{\phi_*^\top \Sigma_p \phi_*} - \boxed{\phi_*^\top \Sigma_p \Phi^\top}\left(K + \sigma_n^2 I\right)^{-1}\boxed{\Phi \Sigma_p \phi_*}\Big)$$

$$\phi_*^\top \Sigma_p \Phi^\top \rightarrow \begin{pmatrix} k(x_*, x^{(1)}) & k(x_*, x^{(2)}) & k(x_*, x^{(3)}) \end{pmatrix} \equiv k_*^\top$$

$$\phi_*^\top \Sigma_p \phi_*^\top \rightarrow k(x_*, x_*)$$

$$K \rightarrow \begin{pmatrix} k(x^{(1)}, x^{(1)}) & k(x^{(1)}, x^{(2)}) & k(x^{(1)}, x^{(3)}) \\ k(x^{(2)}, x^{(1)}) & k(x^{(2)}, x^{(2)}) & k(x^{(2)}, x^{(3)}) \\ k(x^{(3)}, x^{(1)}) & k(x^{(3)}, x^{(2)}) & k(x^{(3)}, x^{(3)}) \end{pmatrix}$$

← All training data

We can bypass the features and weights prior (Σ_p): Define model directly in term of the inner product

- "kernel trick" $\quad \boldsymbol{\psi}(x) \cdot \boldsymbol{\psi}(x') \rightarrow k(x, x')$

$$f_* \mid \mathbf{x}_*, X, \mathbf{y} \sim \mathcal{N}\left(k_*^\top \left(K + \sigma_n^2 I\right)^{-1} \mathbf{y},\right.$$
$$\left. k(x_*, x_*) - k_*^\top \left(K + \sigma_n^2 I\right)^{-1} k_*\right)$$

$$\boldsymbol{\phi}_*^\top \Sigma_p \Phi^\top \rightarrow \begin{pmatrix} k(x_*, x^{(1)}) & k(x_*, x^{(2)}) & k(x_*, x^{(3)}) \end{pmatrix} \equiv k_*^\top$$

$$\boldsymbol{\phi}_*^\top \Sigma_p \boldsymbol{\phi}_*^\top \rightarrow k(x_*, x_*)$$

$$K \rightarrow \begin{pmatrix} k(x^{(1)}, x^{(1)}) & k(x^{(1)}, x^{(2)}) & k(x^{(1)}, x^{(3)}) \\ k(x^{(2)}, x^{(1)}) & k(x^{(2)}, x^{(2)}) & k(x^{(2)}, x^{(3)}) \\ k(x^{(3)}, x^{(1)}) & k(x^{(3)}, x^{(2)}) & k(x^{(3)}, x^{(3)}) \end{pmatrix}$$

← All training data

The kernel function, freed from the features, can have any functional form that works

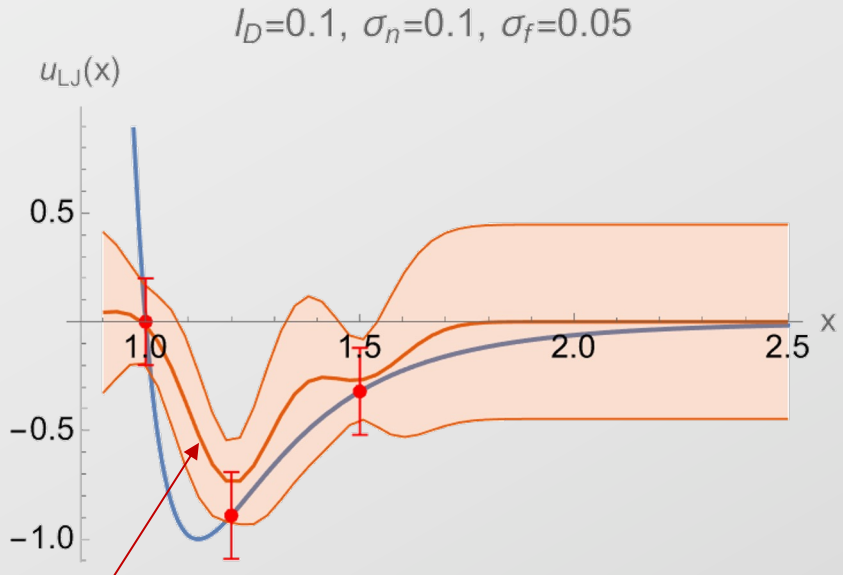
- The Radical Basis Function (RBF) is a popular choice

$$k(x, x') = \sigma_f^2 \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2\ell^2}\right)$$

- Application to LJ example

$$f_* \mid \mathbf{x}_*, X, \mathbf{y} \sim \mathcal{N}(k_*^\top \left(K + \sigma_n^2 I\right)^{-1} \mathbf{y},$$
$$k(x_*, x_*) - k_*^\top \left(K + \sigma_n^2 I\right)^{-1} k_*)$$

$$K = \begin{pmatrix} 0.05 & 0.007 & 0 \\ 0.007 & 0.05 & 0.0005 \\ 0 & 0.0005 & 0.05 \end{pmatrix} \quad k_* = \begin{pmatrix} k(x_*, x^{(1)}) \\ k(x_*, x^{(2)}) \\ k(x_*, x^{(3)}) \end{pmatrix} \quad \mathbf{y} = \begin{pmatrix} 0.00 \\ -0.89 \\ -0.32 \end{pmatrix}$$



Mean as function of x_*

$$k_*^\top \left(K + \sigma_n^2 I\right)^{-1} \mathbf{y} = e^{-50x_*^2}\left(1.6 \times 10^{-23} e^{100x_*} - 4.0 \times 10^{-32} e^{120x_*} - 3.6 \times 10^{-50} e^{150x_*}\right)$$

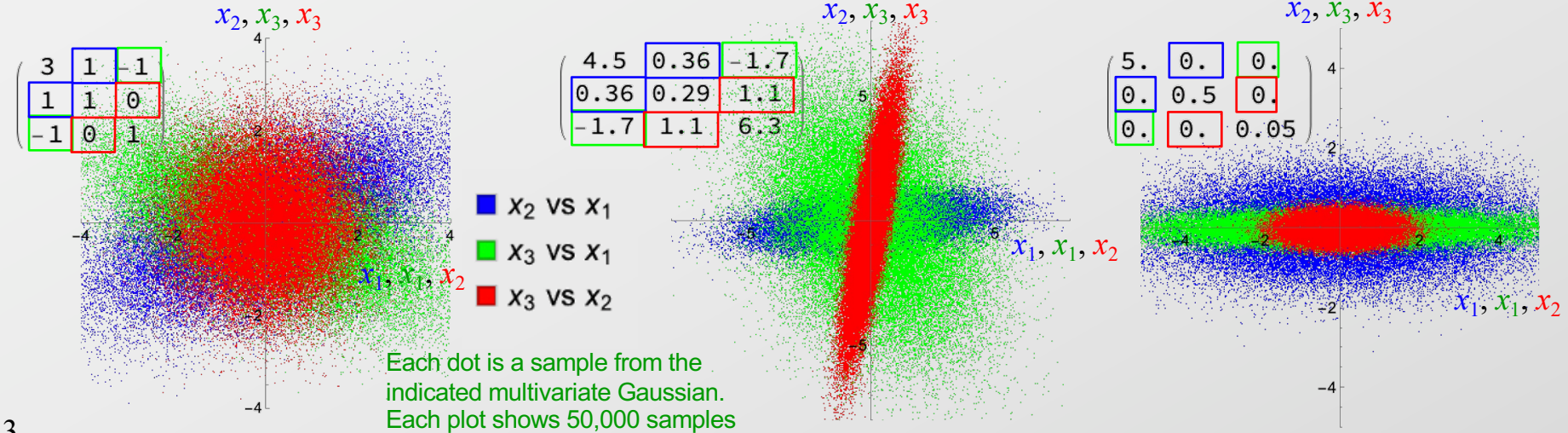
This is a *kernel method*. It may instead be seen as a *Gaussian process*. First, review multivariate Gaussian

$$\mathbf{X} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$$

$$\boldsymbol{\mu} = \begin{pmatrix} \mu_1 \\ \vdots \\ \mu_p \end{pmatrix} \qquad \Sigma = \begin{pmatrix} \sigma_{11} & \cdots & \sigma_{1p} \\ \vdots & \ddots & \vdots \\ \sigma_{p1} & \cdots & \sigma_{pp} \end{pmatrix}$$

- The probability density function is

$$f_{\mathbf{X}}(\mathbf{x}) = (2\pi)^{-k/2}(\det \boldsymbol{\Sigma})^{-1/2} \exp\left(-\tfrac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\top}\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right)$$



Each dot is a sample from the indicated multivariate Gaussian. Each plot shows 50,000 samples

A *Gaussian process* is the extension of a multivariate Gaussian to a continuum of variables

$$\begin{pmatrix} 4.5 & 0.36 & -1.7 \\ 0.36 & 0.29 & 1.1 \\ -1.7 & 1.1 & 6.3 \end{pmatrix}$$

$p = 3$

$$\begin{pmatrix} 1.6 & 0.27 & 0.49 & 0.052 \\ 0.27 & 1.1 & 0.36 & 0.64 \\ 0.49 & 0.36 & 1.0 & 0.92 \\ 0.052 & 0.64 & 0.92 & 1.1 \end{pmatrix}$$

$p = 4$

$$\begin{pmatrix} 0.91 & 0.31 & 0.59 & 0.39 & 0.69 & 0.26 & 0.80 & 0.20 \\ 0.31 & 1.4 & 0.10 & 0.13 & 0.56 & 0.22 & 0.21 & 0.56 \\ 0.59 & 0.10 & 1.5 & 0.53 & 0.49 & 0.66 & 0.36 & 0.21 \\ 0.39 & 0.13 & 0.53 & 1.0 & 0.34 & 0.28 & 0.60 & 0.099 \\ 0.69 & 0.56 & 0.49 & 0.34 & 1.2 & 0.36 & 0.23 & 0.43 \\ 0.26 & 0.22 & 0.66 & 0.28 & 0.36 & 1.0 & 0.65 & 0.76 \\ 0.80 & 0.21 & 0.36 & 0.60 & 0.23 & 0.65 & 1.5 & 0.47 \\ 0.20 & 0.56 & 0.21 & 0.099 & 0.43 & 0.76 & 0.47 & 0.95 \end{pmatrix}$$

$p = 8$

$p = 15$

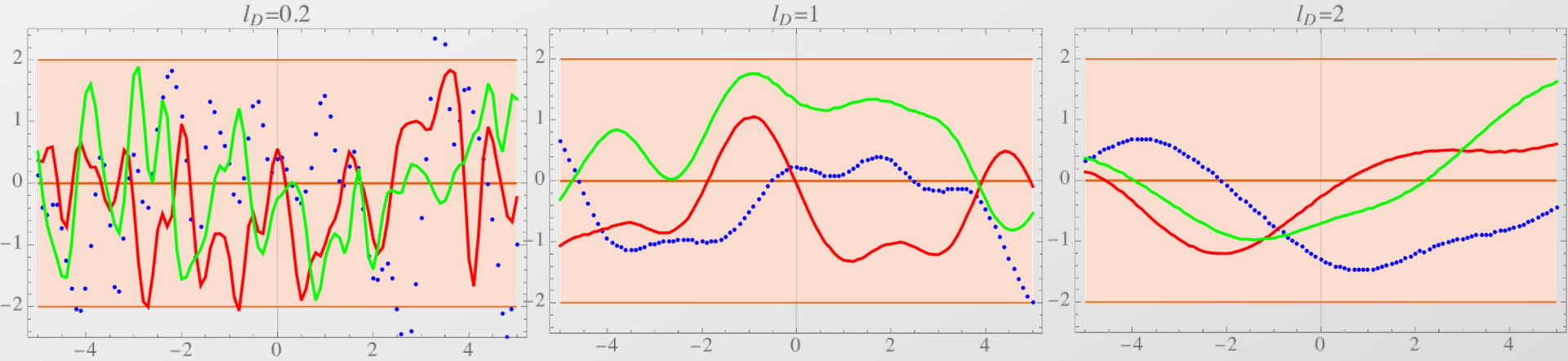
$p = 50$

$$\mathrm{cov}(i, j) = e^{-(i-j)^2/8}$$

When modeling via a Gaussian process, we adopt a Gaussian prior with covariance $k(x, x')$

$$\mathrm{cov}(x, x') = k(x, x') = \sigma_f^2 \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2\ell_{\mathrm{D}}^2}\right)$$

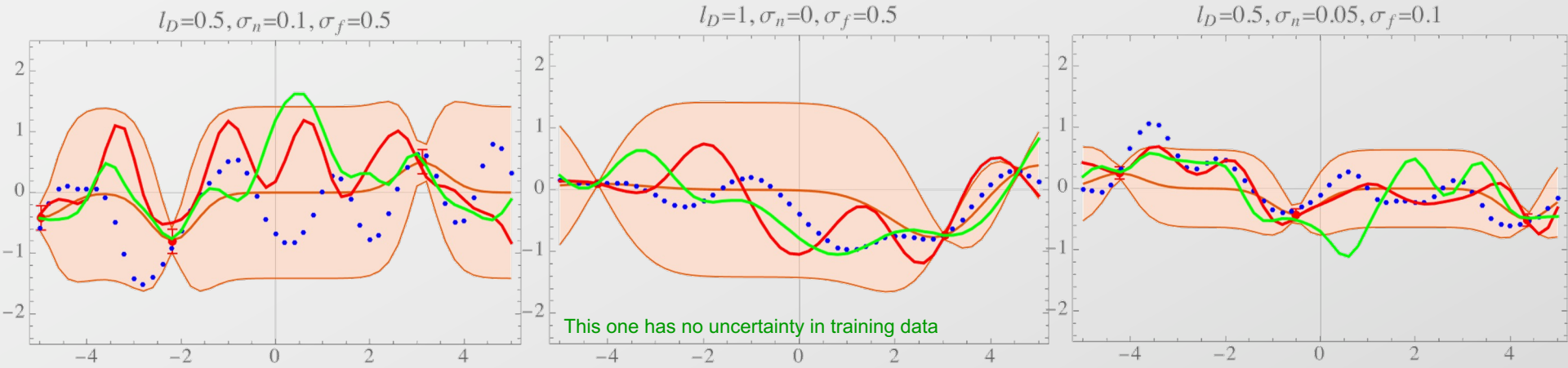
The decay length ℓ_{D} affects the smoothness of the sampled functions



The posterior is developed from Bayes' rule. It is the same Gaussian as from the kernel method

$$f_* \mid \mathbf{x}_*, X, \mathbf{y} \sim \mathcal{N}\left(k_*^\top \left(K + \sigma_n^2 I\right)^{-1} \mathbf{y},\right.$$
$$\left. k(x_*, x_*) - k_*^\top \left(K + \sigma_n^2 I\right)^{-1} k_* \right)$$

For these examples, three (x,y) training points are generated at random



More generally, the kernel *k*(x,x') can be interpreted as a distance measure

- This is consistent with its connection to the dot product

$$\psi(x) \cdot \psi(x') \rightarrow k(x, x')$$

  - Smaller dot product ➔ x and x' are more orthogonal

- For RPG kernel, covariance depends only on separation, and decreases with increasing distance

p = 15

1.0	0.78	0.37	0.11	0.018	0.0019	0.00012	4.8×10^{-6}	1.1×10^{-7}	1.6×10^{-9}	0	0	0	0	0
0.78	1.0	0.78	0.37	0.11	0.018	0.0019	0.00012	4.8×10^{-6}	1.1×10^{-7}	1.6×10^{-9}	0	0	0	0
0.37	0.78	1.0	0.78	0.37	0.11	0.018	0.0019	0.00012	4.8×10^{-6}	1.1×10^{-7}	1.6×10^{-9}	0	0	0
0.11	0.37	0.78	1.0	0.78	0.37	0.11	0.018	0.0019	0.00012	4.8×10^{-6}	1.1×10^{-7}	1.6×10^{-9}	0	0
0.018	0.11	0.37	0.78	1.0	0.78	0.37	0.11	0.018	0.0019	0.00012	4.8×10^{-6}	1.1×10^{-7}	1.6×10^{-9}	0
0.0019	0.018	0.11	0.37	0.78	1.0	0.78	0.37	0.11	0.018	0.0019	0.00012	4.8×10^{-6}	1.1×10^{-7}	1.6×10^{-9}
0.00012	0.0019	0.018	0.11	0.37	0.78	1.0	0.78	0.37	0.11	0.018	0.0019	0.00012	4.8×10^{-6}	1.1×10^{-7}
4.8×10^{-6}	0.00012	0.0019	0.018	0.11	0.37	0.78	1.0	0.78	0.37	0.11	0.018	0.0019	0.00012	4.8×10^{-6}
1.1×10^{-7}	4.8×10^{-6}	0.00012	0.0019	0.018	0.11	0.37	0.78	1.0	0.78	0.37	0.11	0.018	0.0019	0.00012
1.6×10^{-9}	1.1×10^{-7}	4.8×10^{-6}	0.00012	0.0019	0.018	0.11	0.37	0.78	1.0	0.78	0.37	0.11	0.018	0.0019
0	1.6×10^{-9}	1.1×10^{-7}	4.8×10^{-6}	0.00012	0.0019	0.018	0.11	0.37	0.78	1.0	0.78	0.37	0.11	0.018
0	0	1.6×10^{-9}	1.1×10^{-7}	4.8×10^{-6}	0.00012	0.0019	0.018	0.11	0.37	0.78	1.0	0.78	0.37	0.11
0	0	0	1.6×10^{-9}	1.1×10^{-7}	4.8×10^{-6}	0.00012	0.0019	0.018	0.11	0.37	0.78	1.0	0.78	0.37
0	0	0	0	1.6×10^{-9}	1.1×10^{-7}	4.8×10^{-6}	0.00012	0.0019	0.018	0.11	0.37	0.78	1.0	0.78
0	0	0	0	0	1.6×10^{-9}	1.1×10^{-7}	4.8×10^{-6}	0.00012	0.0019	0.018	0.11	0.37	0.78	1.0

We are much more interested in cases where the *x* data are multidimensional

- E.g, **x** is formed from the coordinates of several atoms
- The RBF kernel is written to accommodate this

$$k(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2\ell_{\mathrm{D}}^2}\right)$$

  - ℓ_{D} might be different for different components of **x**
- Nothing else changes for this generalization

Gaussian processes form *nonparametric models*. There are no parameters to fit

- Hyperparameters are tuned to improve performance
  - $\{\ell_{\mathrm{D}}\}, \sigma_f, \sigma_n$ $\quad k(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2\ell_{\mathrm{D}}^2}\right)$
- Hyperparameter optimization is performed by maximizing log-likelihood of observing the training data

$$\log p(\mathbf{y} \mid X) = -\tfrac{1}{2}\mathbf{y}^\top (K + \sigma_n^2 I)^{-1}\mathbf{y} - \tfrac{1}{2}\log|K + \sigma_n^2 I| - \tfrac{n}{2}\log 2\pi$$

- Calculation of a model estimate is just a dot product with *all* of the *n* training data

$\left(k(\mathbf{x}_*, \mathbf{x}^{(1)}) \quad \cdots \quad k(\mathbf{x}_*, \mathbf{x}^{(n)})\right)$ 1× *n*

n × 1; compute only once for all $\boldsymbol{x}_*$

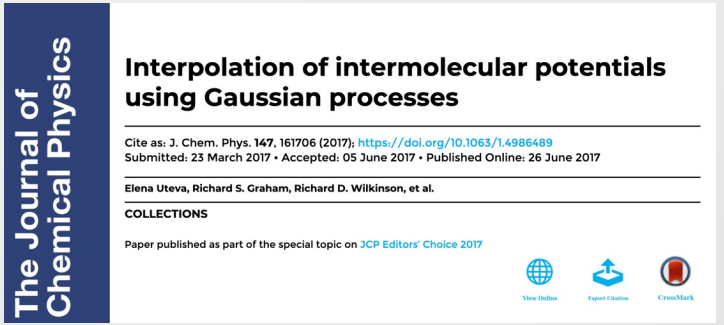
$$f_* \mid \mathbf{x}_*, X, \mathbf{y} \sim \mathcal{N}\left(k_*^\top \underbrace{\left(K + \sigma_n^2 I\right)^{-1}}_{n \times n} \underbrace{\mathbf{y}}_{n \times 1},\ k(x_*, x_*) - k_*^\top \left(K + \sigma_n^2 I\right)^{-1} k_*\right)$$

Other concepts are of importance in application of ML to computational chemistry

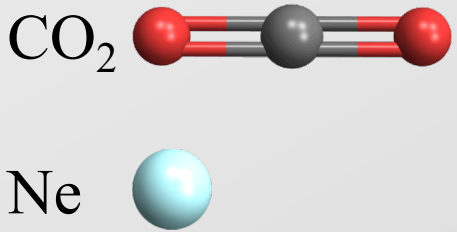
- Data augmentation
  - Exploiting symmetries in the physical system to generate new data without additional calculation, by permuting the elements of the **x** vector that leaves the system effectively unchanged
    - E.g., swapping coordinates of two oxygen atoms in CO_2
- Transfer learning
  - Using a pre-trained model as a start to training a similar model
    - e.g., a NN is trained on low-level quantum chemical data and then improved by fewer higher-level training data
- Active learning
  - Using uncertainty in estimate from Gaussian process to determine whether to do new calculations to generate additional training data

Case study: Gaussian-process modeling of CO2-Ne pair potential from ab initio training data

- Steps performed in study
  - Select configurations for data generation
  - Perform ab initio energy calculations
  - Tabulate ($\mathbf{x}$, y) data, form into training and test sets
  - Optimize hyperparameters via maximization of log-likelihood
  - Evaluate via RMSD of test set
  - Apply to calculation of virial coefficients, which can be compared to experiment



Rigid model

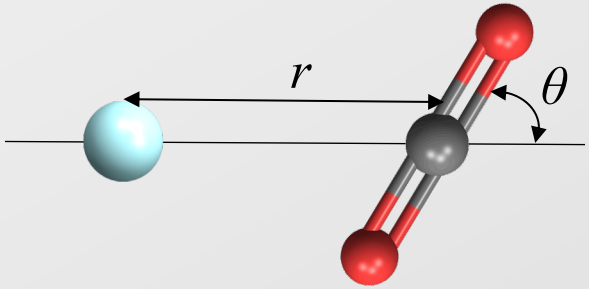


Data generation aims to sample a broad, homogeneous representation of configurations

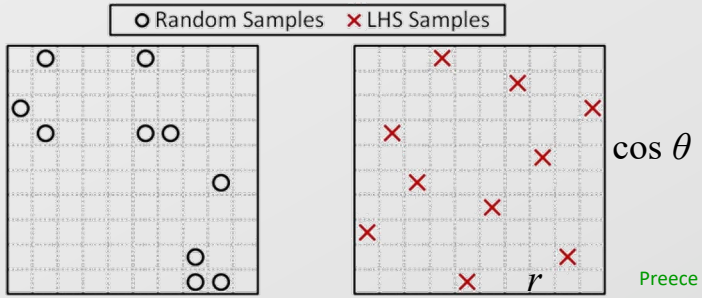
- Only two coordinates are needed to specify configuration

TABLE I. Coordinates for the test (grid or LHC) data for each system.

	Test grid or Latin hypercube			
System	Coordinate	Range	Spacing	Test points
CO_2–Ne	r	1.5–10 Å	0.116 Å	1 122
	$\cos\theta$	0–1	0.05	

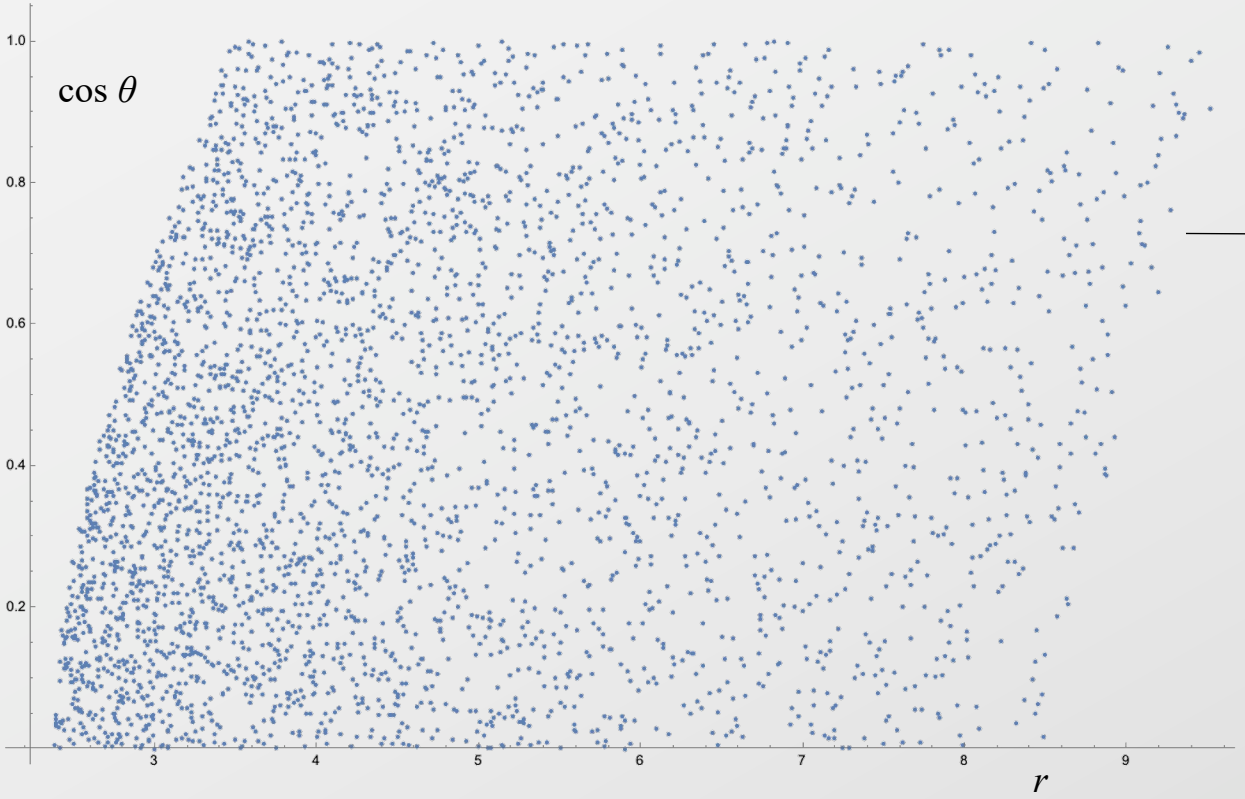


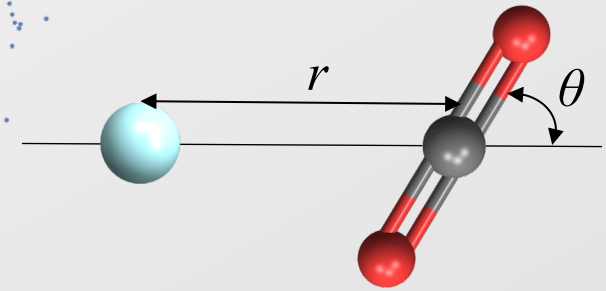
- A *latin hypercube* is used find random but well dispersed configurations



Preece & Milanović, 10.1109/TPWRS.2015.2417204

Data generation aims to sample a broad, homogeneous representation of configurations





Energy and geometry filters are applied to eliminate irrelevant data

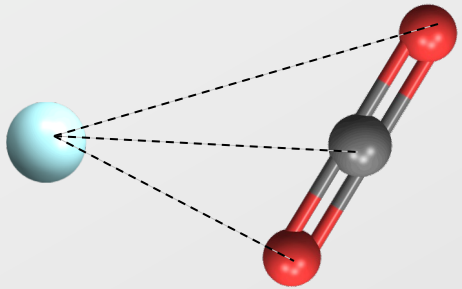
High-level ab initio calculations of the energy are performed for each configuration

- MP2 theory
- aug-cc-pVTZ basis set
- 1122 configurations from LHC
- Molpro software

X data provides an overspecified representation of configuration

- **x** is formed from three distances

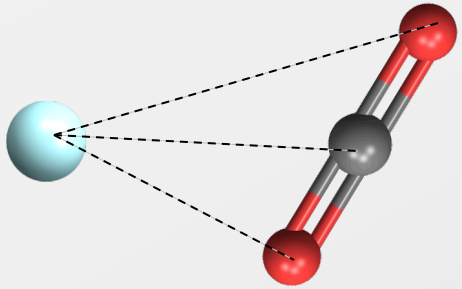
$1/r_{\text{O1-Ne}}$	$1/r_{\text{O2-Ne}}$	$1/r_{\text{C-Ne}}$	energy
0.350294	0.32761	0.321279	0.0000884092
0.131045	0.130434	0.128681	-2.93045×10^{-6}
0.140514	0.159722	0.124223	-6.68865×10^{-6}
0.368198	0.352168	0.326241	0.00075408
0.204061	0.260179	0.166751	−0.0000987712
0.402868	0.393012	0.341903	0.0038665
0.290997	0.27914	0.272258	−0.00026795
0.232019	0.311852	0.18388	−0.00016284
0.113186	0.121581	0.104731	-1.46951×10^{-6}
0.247046	0.252365	0.224879	−0.00013464
0.134051	0.13521	0.129853	-3.39012×10^{-6}
⋮	⋮	⋮	⋮

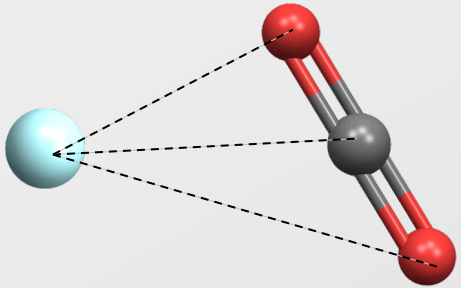


Also, it is found that performance improves by using $1/r$ rather than r to form the **x** vector

Data augmentation exploits symmetries to provide additional data for free

- These have different **x** but the same energy





Gaussian process kernel includes the symmetry of the molecules

- Different ℓ_{D} for each coordinate

$$\kappa(\mathbf{x},\ \mathbf{x}') = \sigma_f^2 \prod_{i=1}^{N_D} \exp\left[-\frac{(x_i - x_i')^2}{2l_i^2}\right]$$

- Kernel is a sum over symmetric transformations

$$k_{\mathrm{sym}}(\mathbf{x}, \mathbf{x}') = \sum_{g \in G} \kappa(g\mathbf{x}, \mathbf{x}')$$

Agreement with test data improves with size of training set

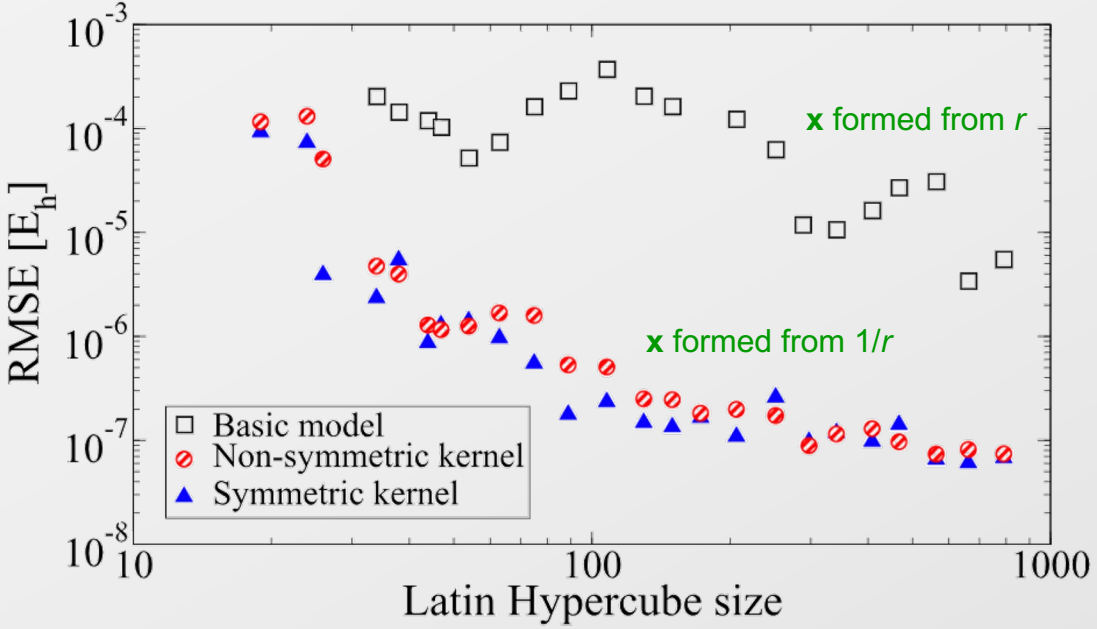
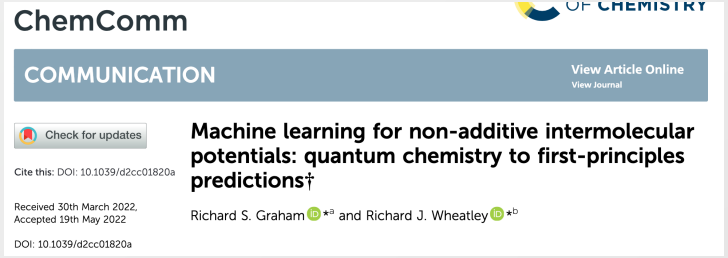


FIG. 1. RMSE against LHC size for CO_2–Ne. The lowest energy in the grid data is $-2.90 \times 10^{-4}\ E_h$.

A follow-up study for CO2-Ar shows excellent agreement with experiment

- CCSD(T) theory, extrapolation to infinite basis set
- 2-and 3-body potentials used to compute mixture virial coefficients up to 5th order


ChemComm

COMMUNICATION

View Article Online
View Journal

Check for updates

Cite this: DOI: 10.1039/d2cc01820a

Received 30th March 2022,
Accepted 19th May 2022

DOI: 10.1039/d2cc01820a

Machine learning for non-additive intermolecular potentials: quantum chemistry to first-principles predictions†

Richard S. Graham *a and Richard J. Wheatley *b

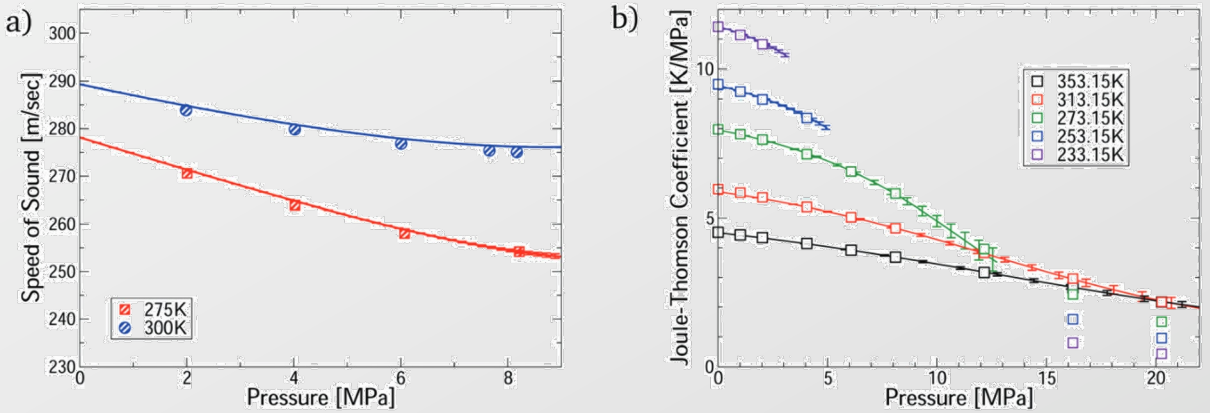


Fig. 3 Comparison of the first-principles predictions for CO_2–Ar mixtures *via* our virial calculations. (a) Speed of sound measurements[32] at ϕ_{CO2} = 0.5; and (b) Joule–Thomson coefficient measurements[33] at ϕ_{CO2} = 0.464. All predictions are converged with respect to number of virial terms.

Suggested Reading/Viewing

- C. E. Rasmussen & C. K. I. Williams, *Gaussian Processes for Machine Learning*, the MIT Press, 2006. Chapters 1 and 2.
  - www.GaussianProcess.org/gpml
- *Quantum Chemistry in the Age of Machine Learning*, edited by P. O. Dral
  - *Chapter 9. Kernel Methods,* Max Pinheiro Jr. and Pavlo O. Dral
  - *Chapter 10. Bayesian Inference*, Wei Liang and Hongsheng Dai
  - Posted on UBLearns
- A Visual Exploration of Gaussian Processes
  - https://distill.pub/2019/visual-exploration-gaussian-processes/