MAX PLANCK INSTITUTE
FOR POLYMER RESEARCH

# Machine learning for energy landscapes

Tristan Bereau
Van 't Hoff Institute for Molecular Sciences & Informatics Institute
University of Amsterdam

Today

- Introduction to kernel-based ML
- Incorporation of physical symmetries, conservation laws

# Machine learning for energy landscapes

Tristan Bereau
Van 't Hoff Institute for Molecular Sciences & Informatics Institute
University of Amsterdam

MAX PLANCK INSTITUTE
FOR POLYMER RESEARCH

Today
- Introduction to kernel-based ML
- Incorporation of physical symmetries, conservation laws

# Machine learning for energy landscapes

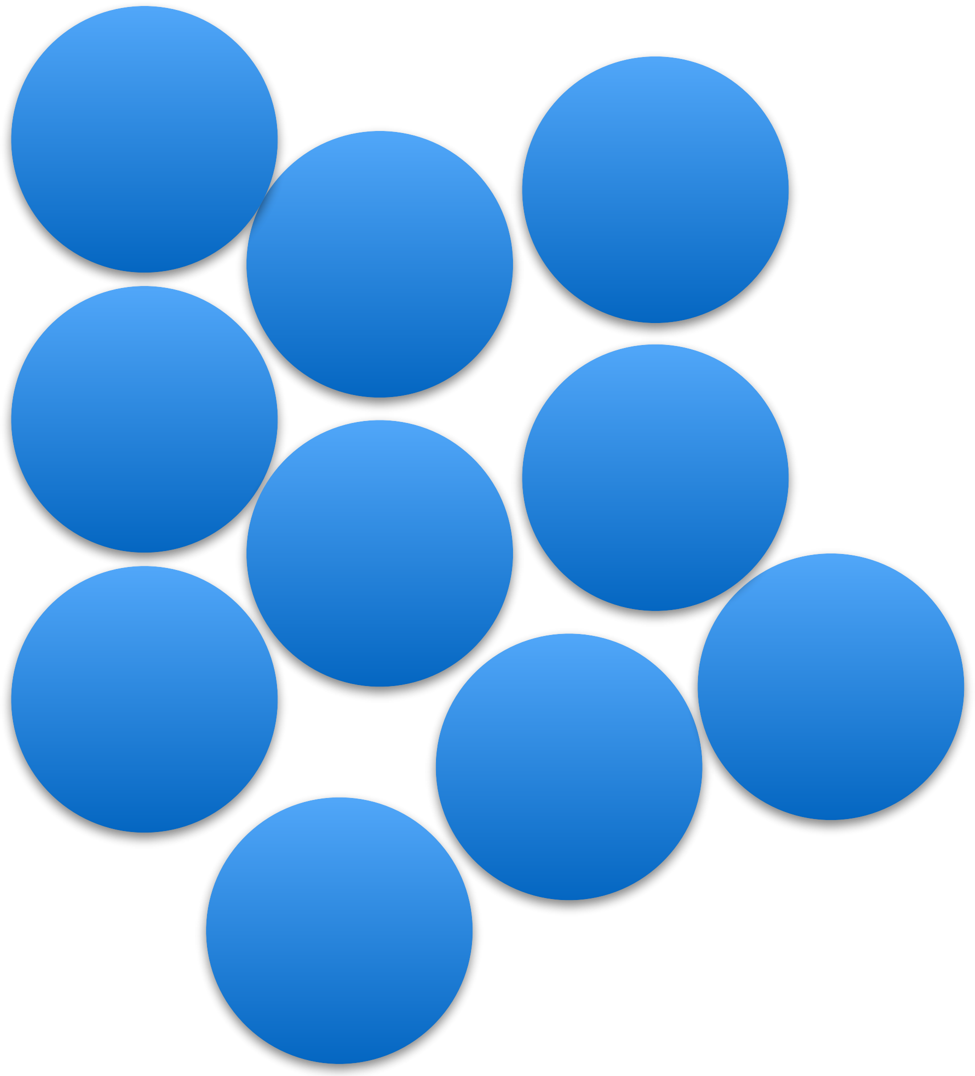
Tristan Bereau
Van 't Hoff Institute for Molecular Sciences & Informatics Institute
University of Amsterdam

Thursday
Supervised ML in chemistry and materials science

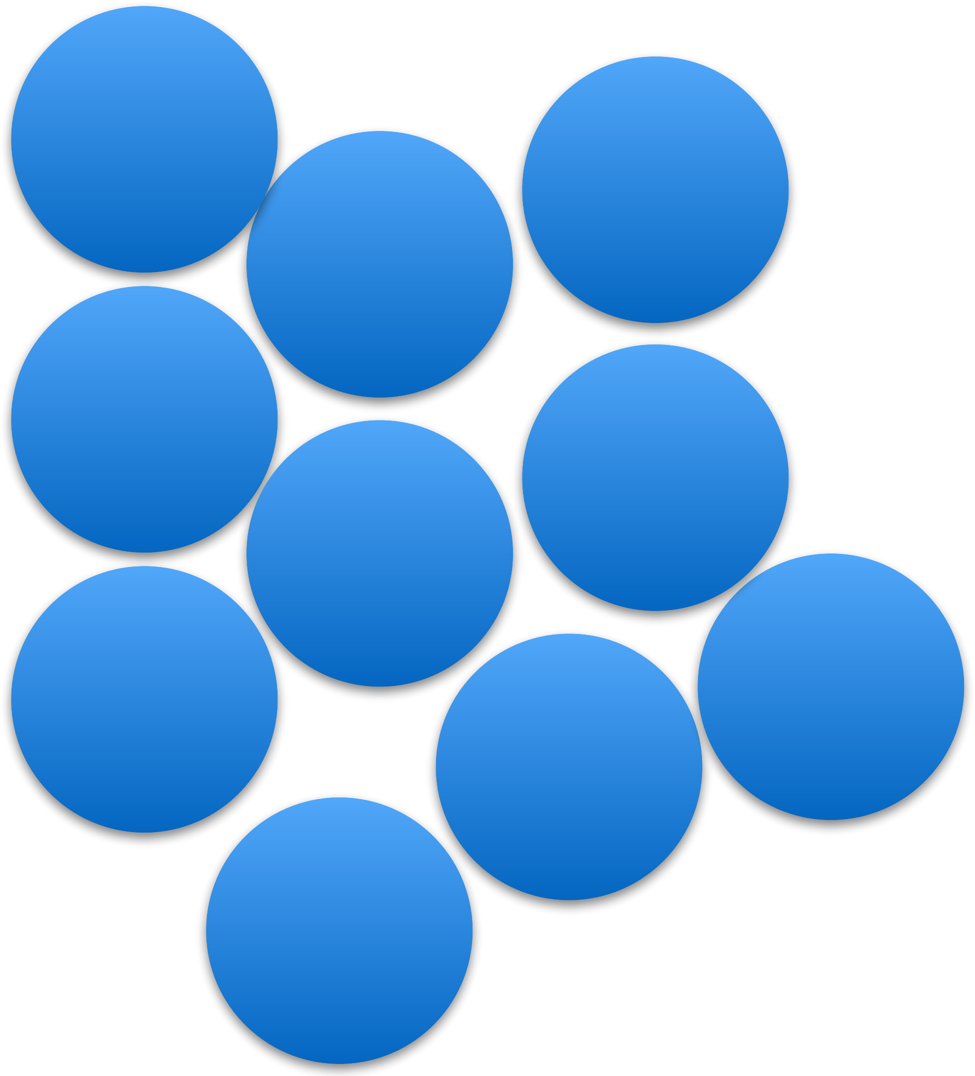

Kevin Jablonka


Mohamad Moosavi

# Molecular dynamics



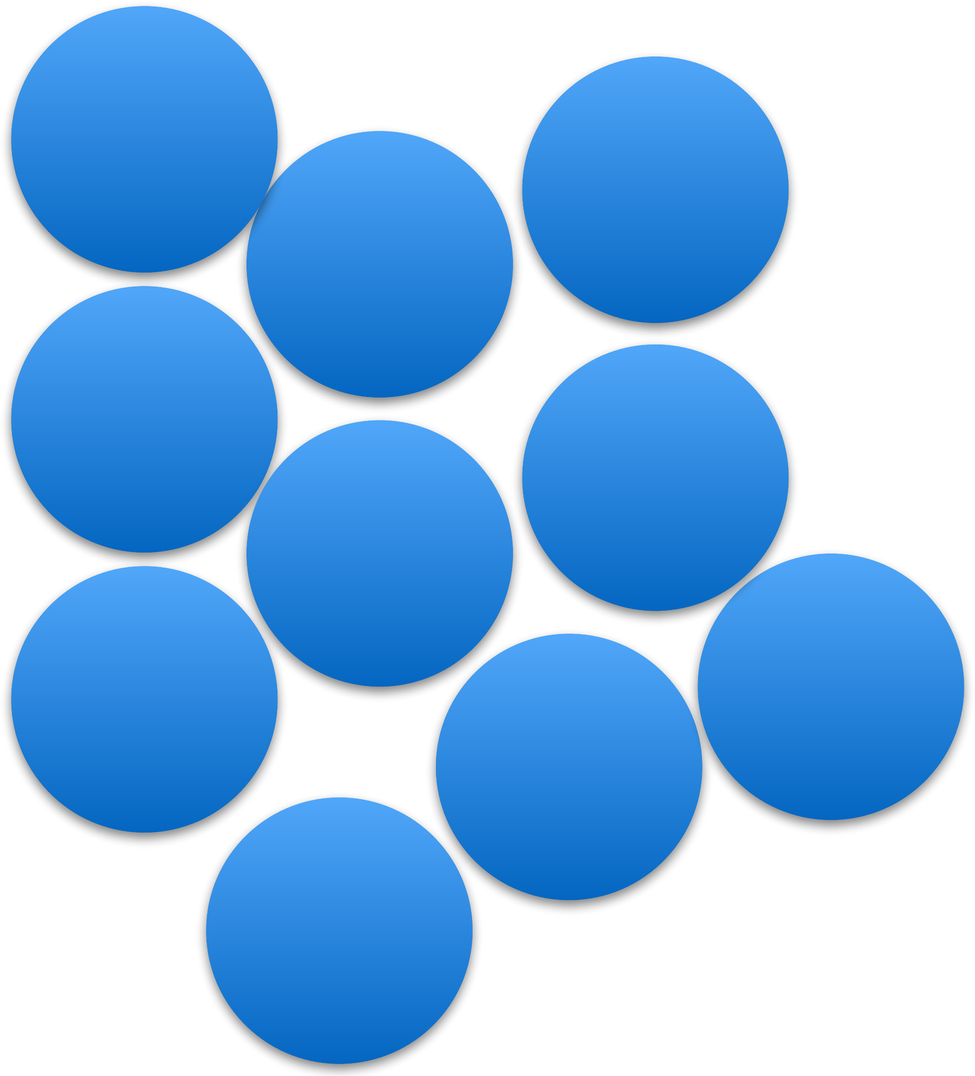
# Molecular dynamics



$$\mathbf{F} = m\mathbf{a}$$

THEORY GROUP

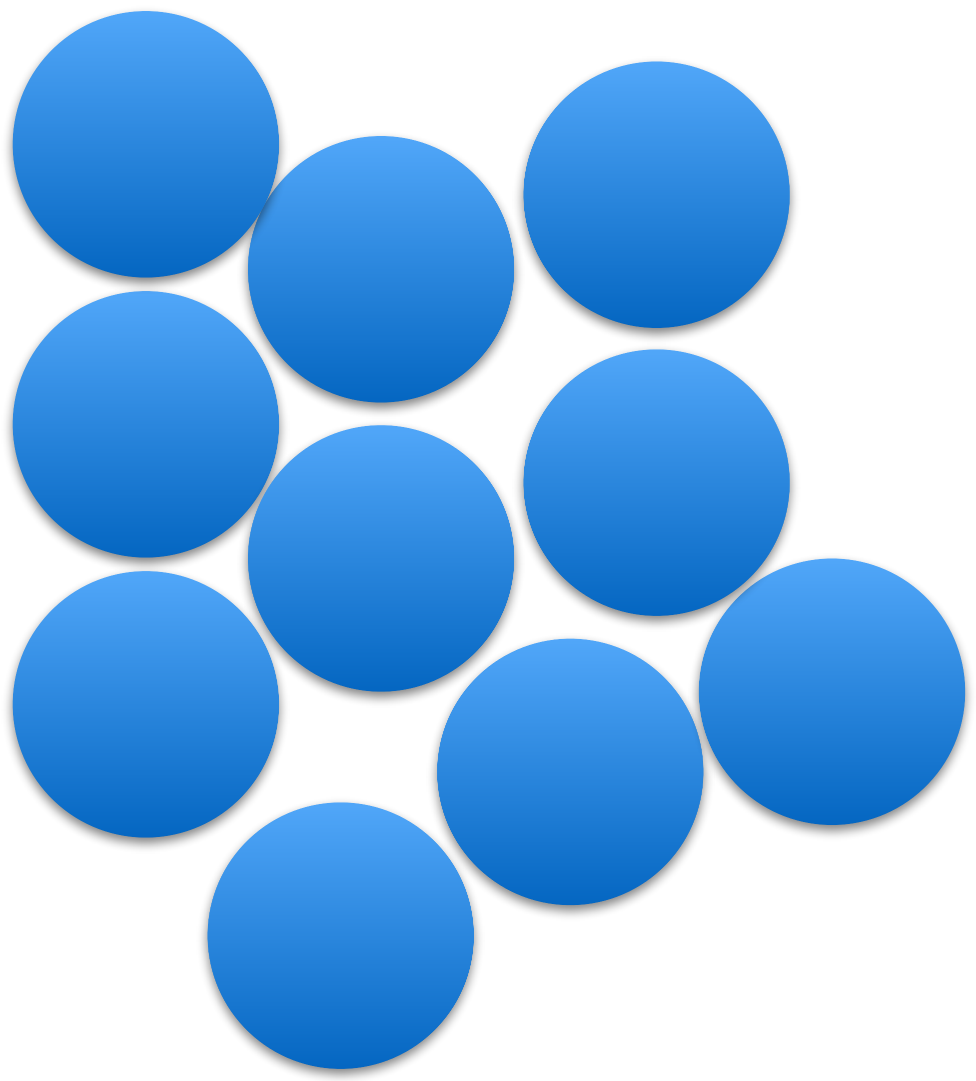
# Molecular dynamics



$$\mathbf{F} = m\mathbf{a}$$

Specify interparticle forces: “force field”

# Molecular dynamics

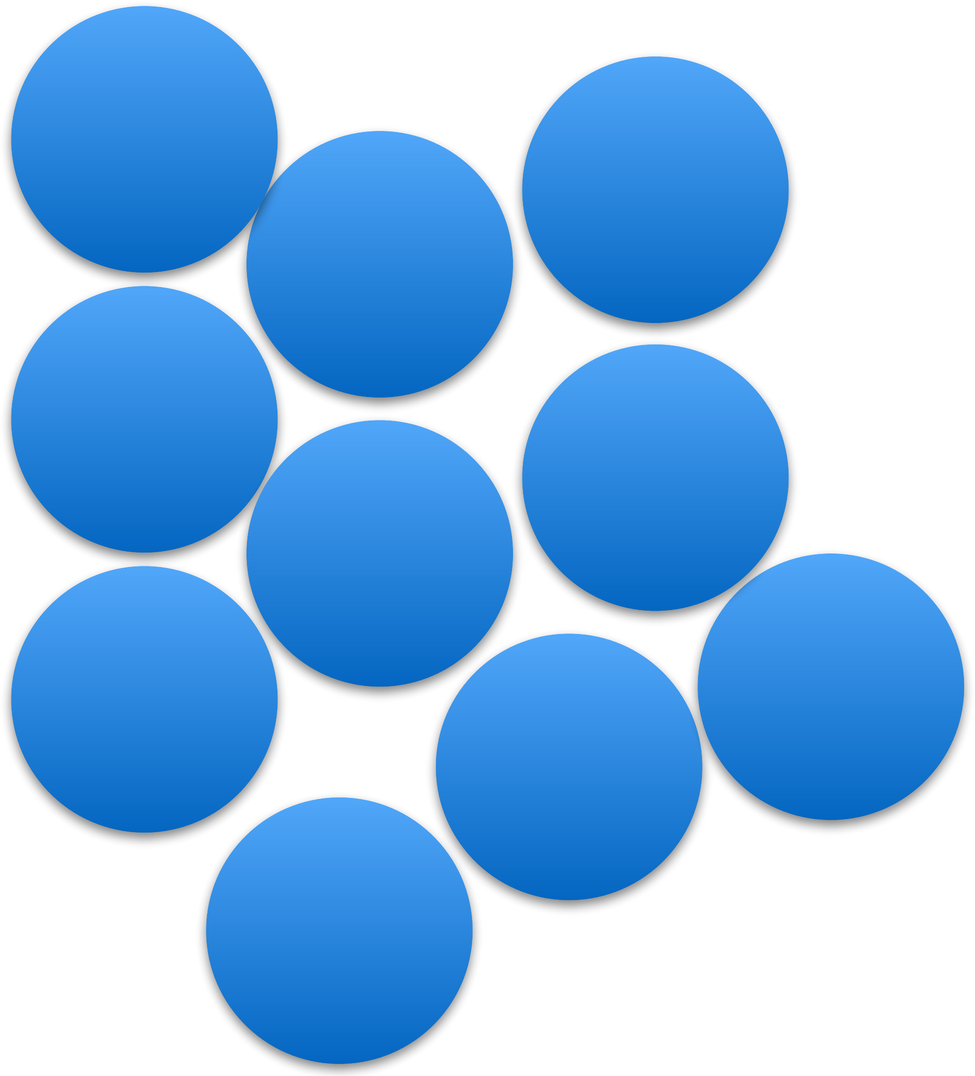


Numerically integrate
particle positions

$$\mathbf{F} = m\mathbf{a}$$

Specify interparticle
forces: "force field"

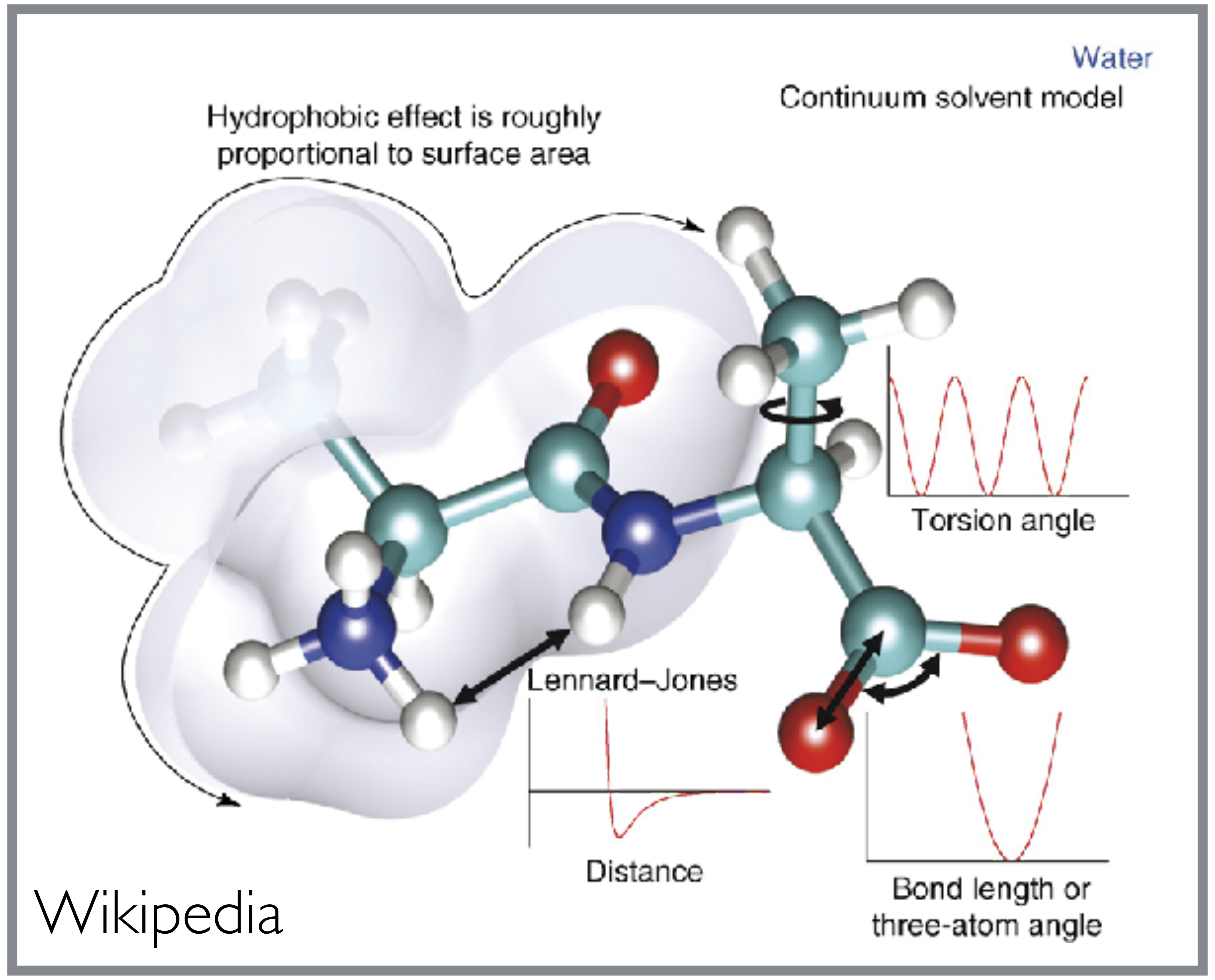
# Molecular dynamics



Numerically integrate particle positions

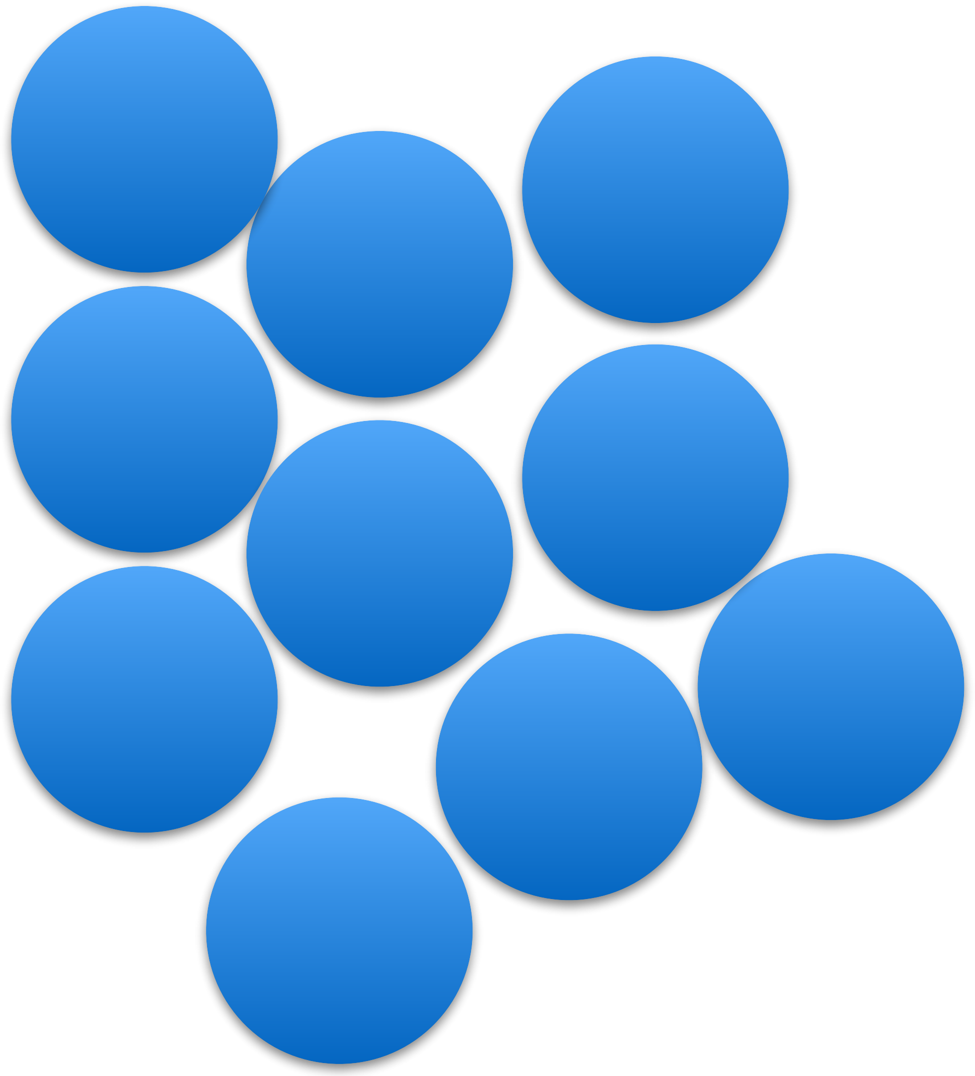
$$\mathbf{F} = m\mathbf{a}$$

Specify interparticle forces: “force field”



Wikipedia

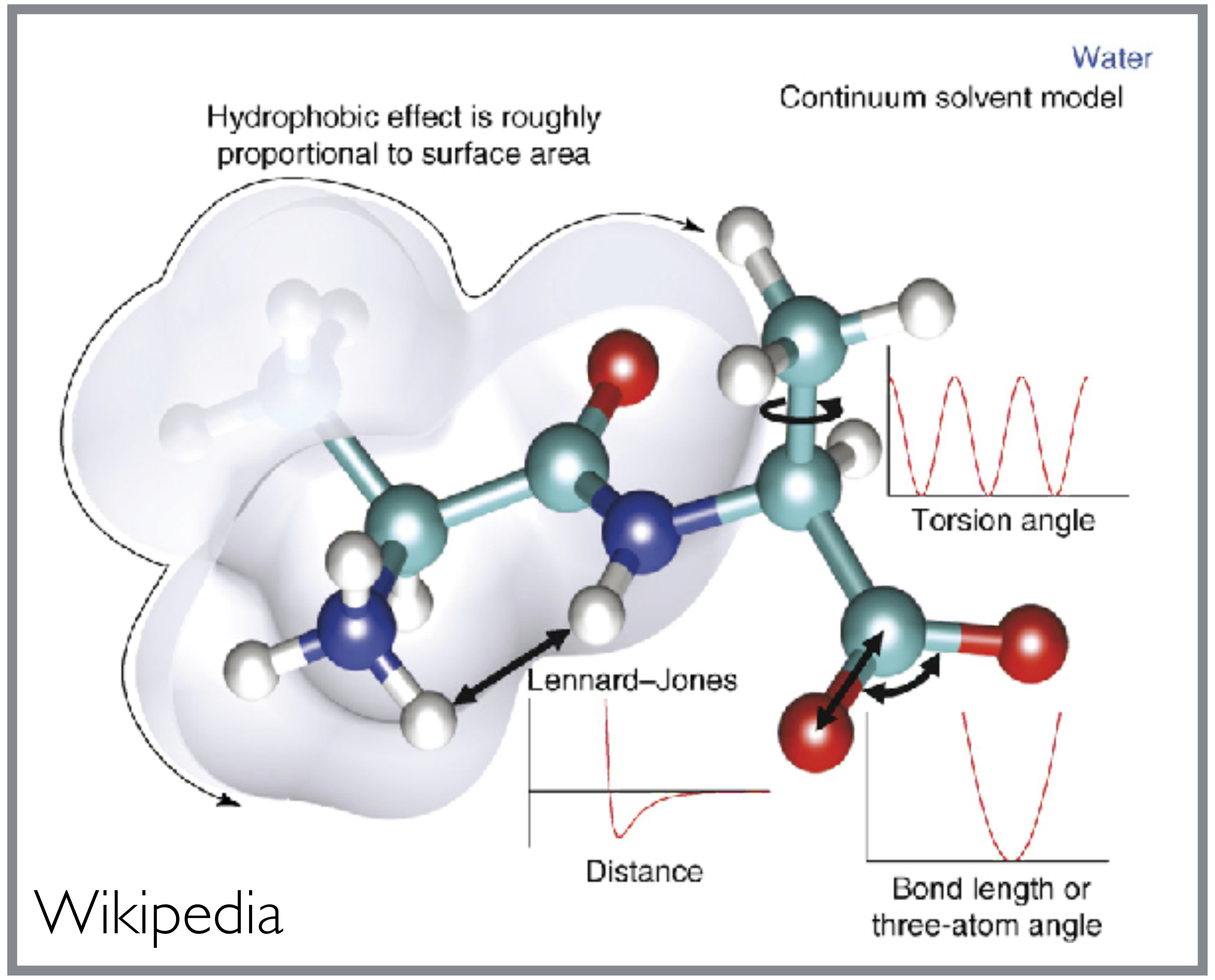
# Molecular dynamics



Numerically integrate particle positions

$$\mathbf{F} = m\mathbf{a}$$

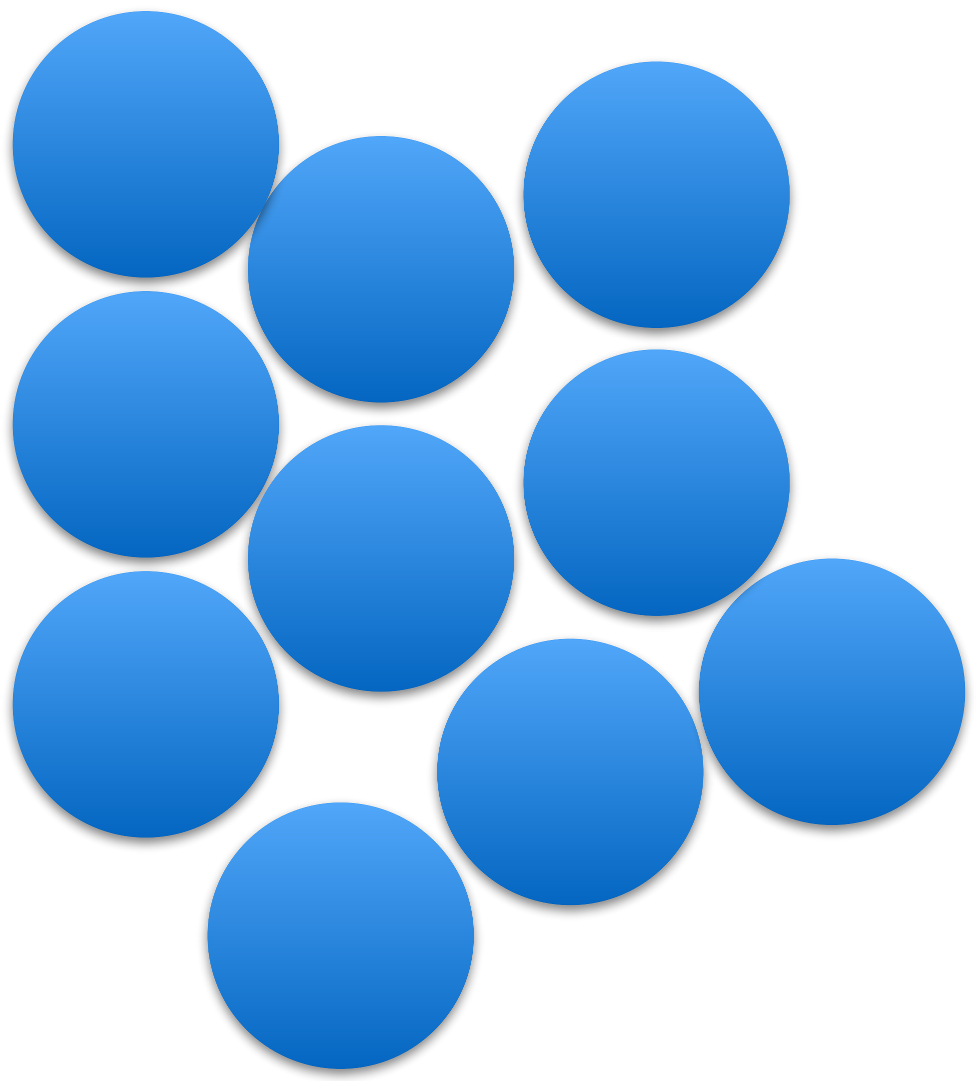
Specify interparticle forces: “force field”



Wikipedia

fs ps ns $\mu$s ms s

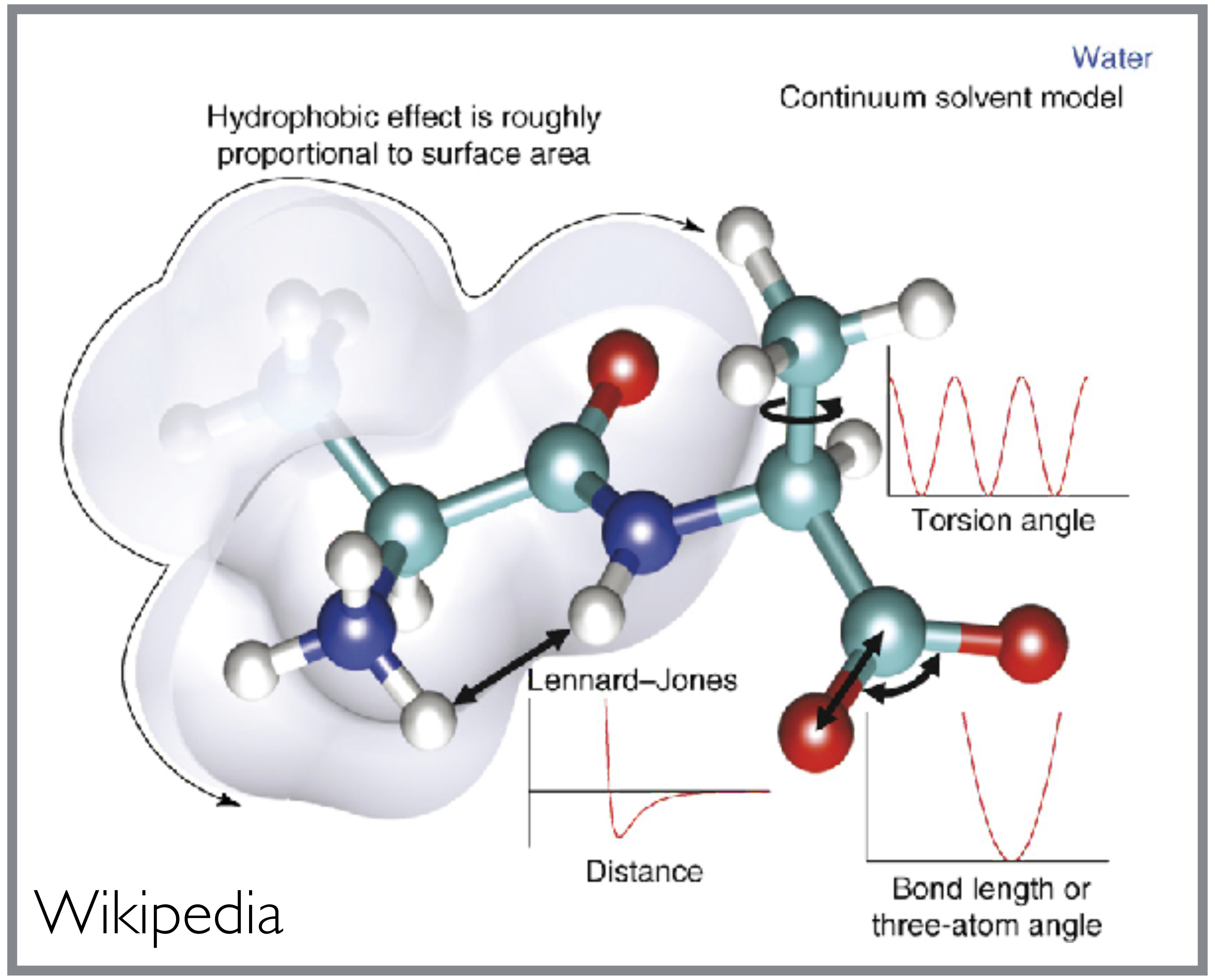
# Molecular dynamics



Numerically integrate
particle positions

$$\mathbf{F} = m\mathbf{a}$$

Specify interparticle
forces: “force field”

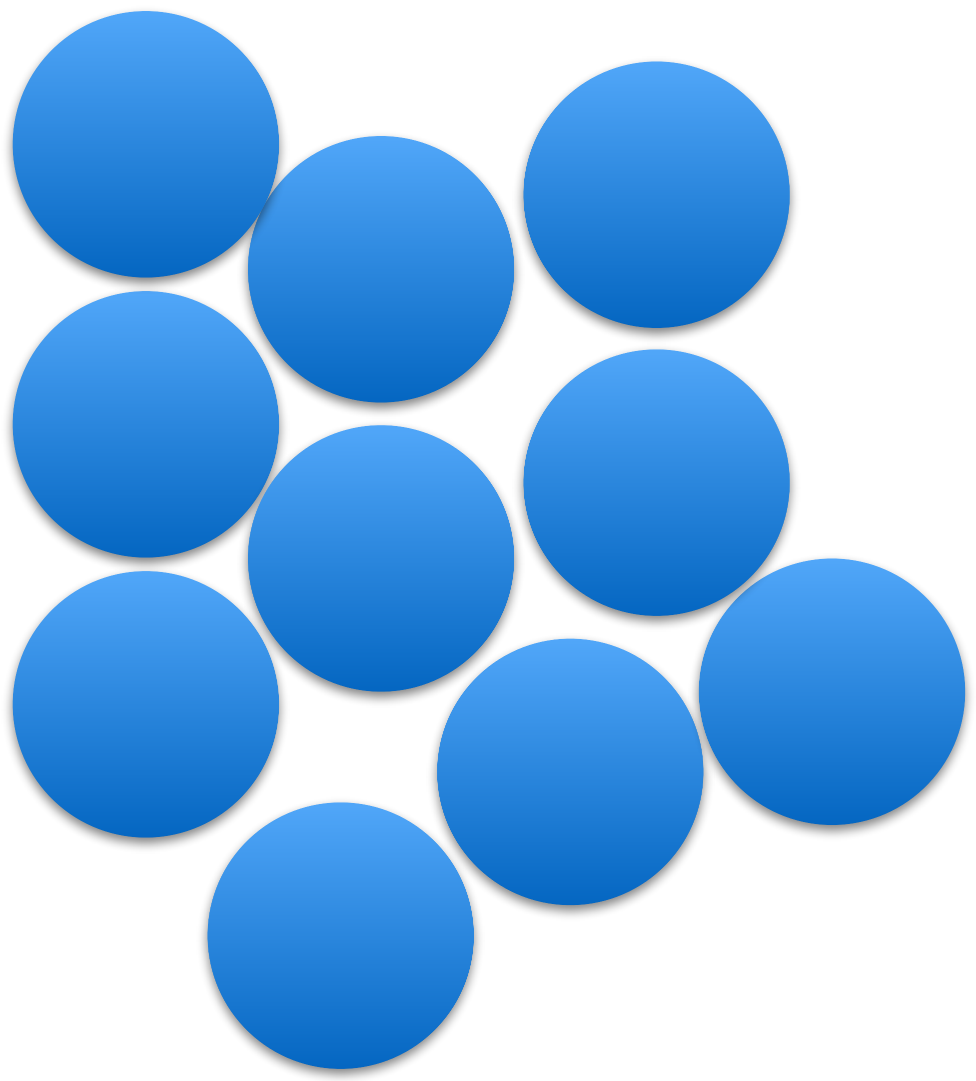


Wikipedia

integration
time step

fs ps ns $\mu$s ms s

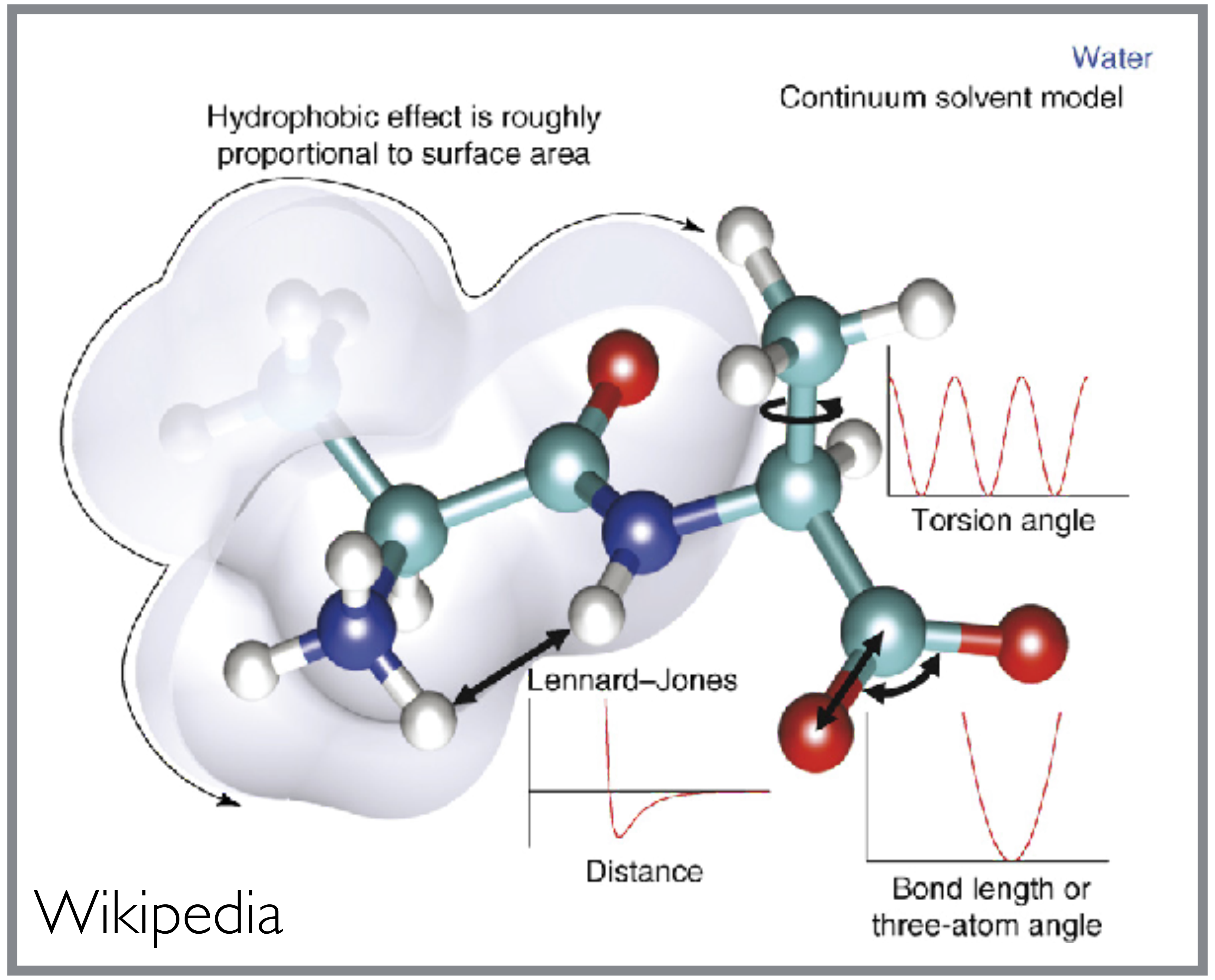
# Molecular dynamics



Numerically integrate particle positions

$$\mathbf{F} = m\mathbf{a}$$

Specify interparticle forces: “force field”

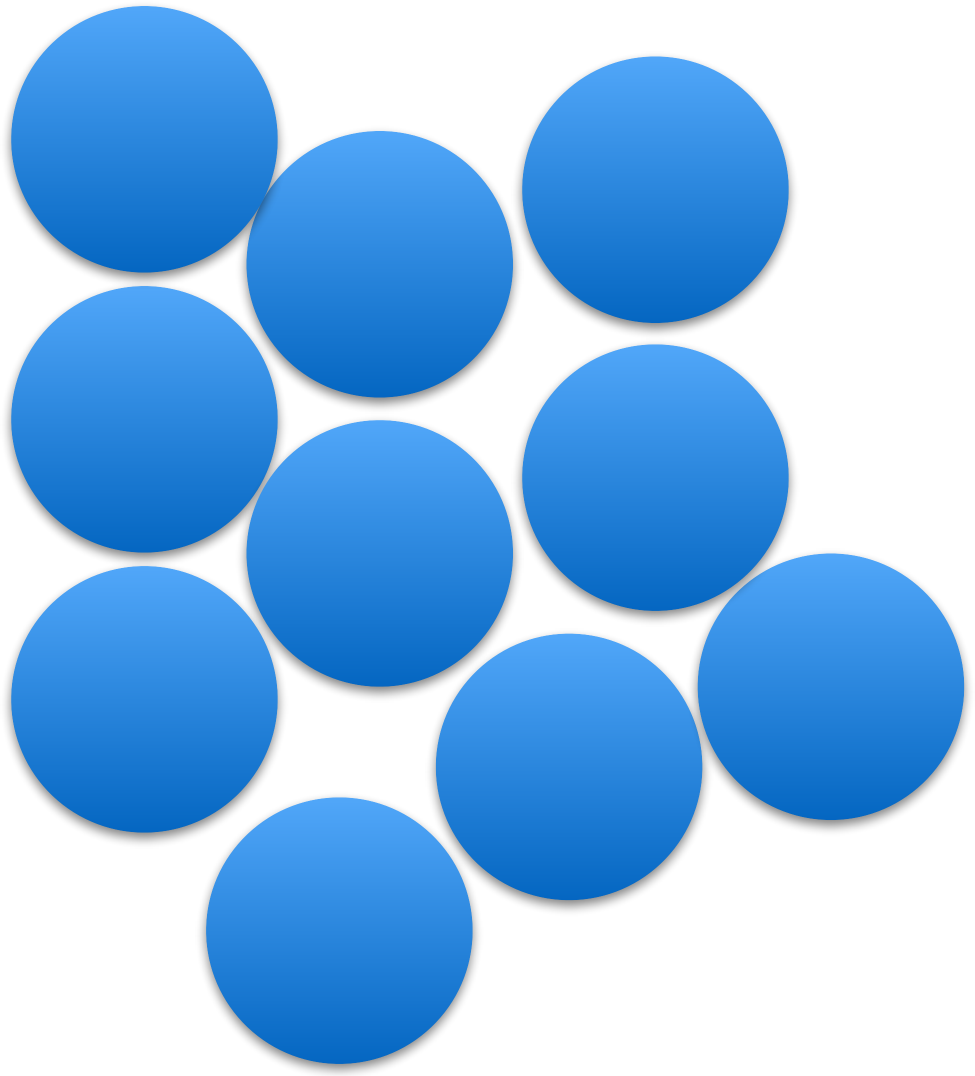



integration time step

Timescales of interest

fs ps ns $\mu$s ms s

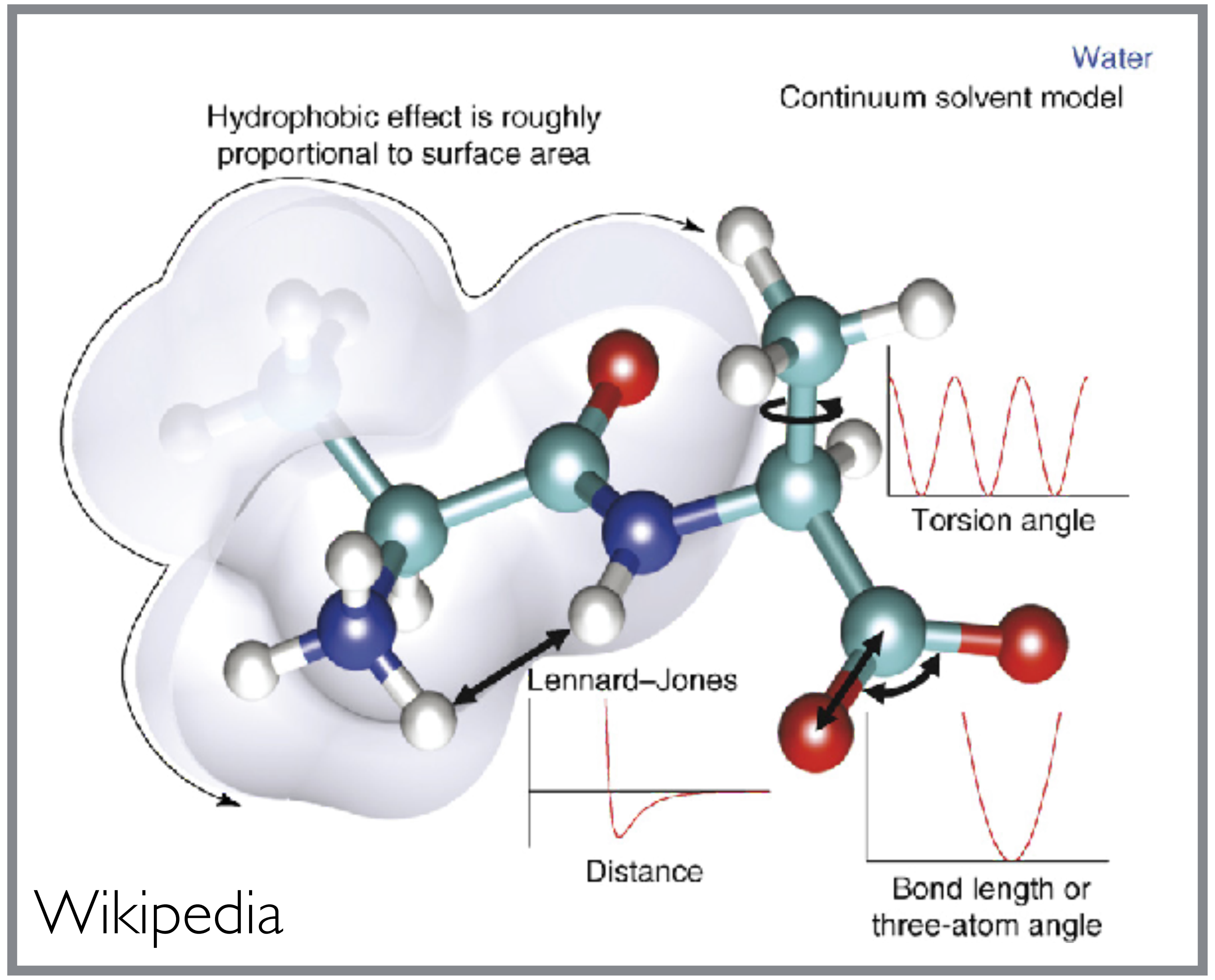
# Molecular dynamics



Numerically integrate particle positions

$$\mathbf{F} = m\mathbf{a}$$

Specify interparticle forces: “force field”




Wikipedia

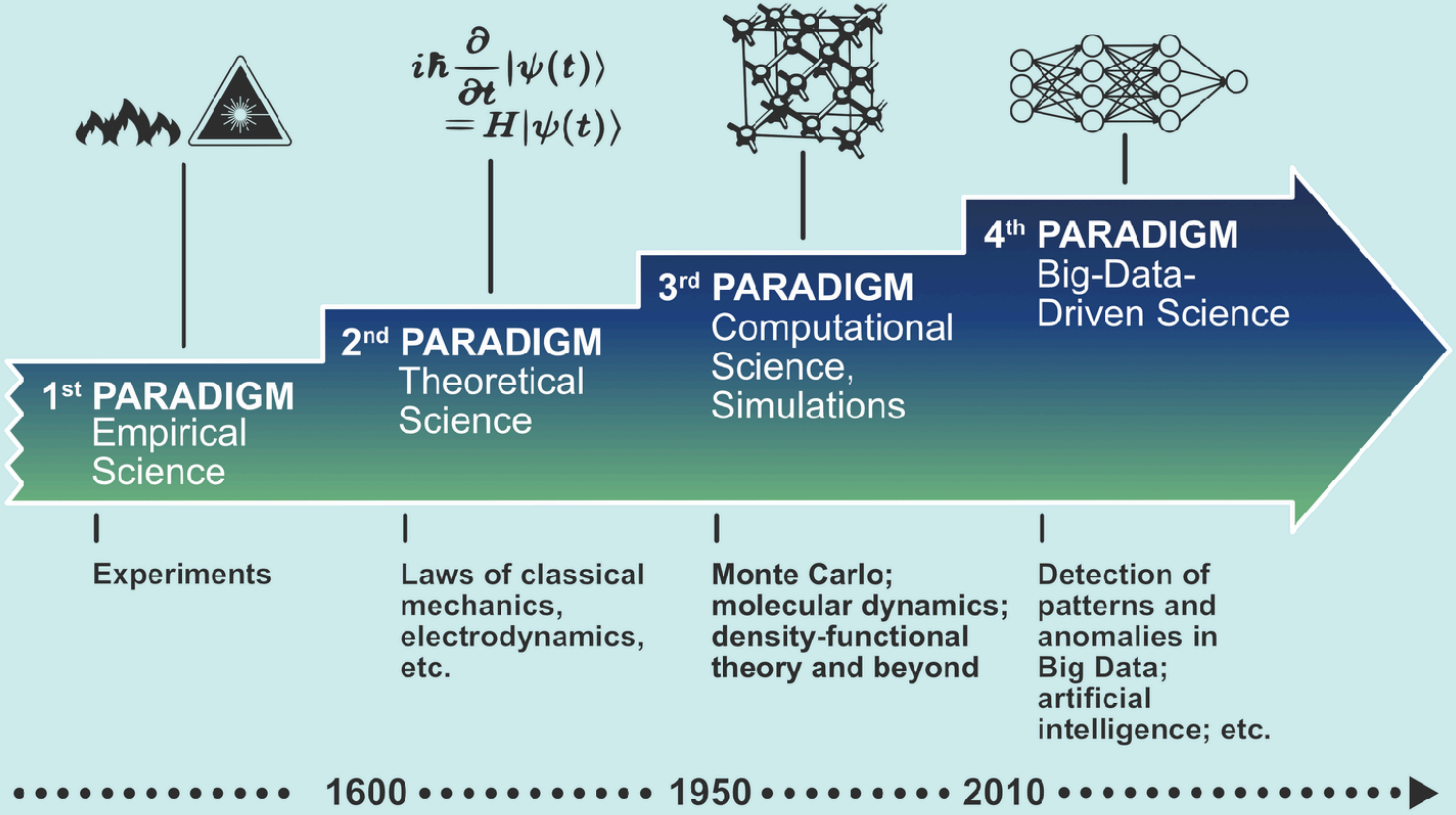
integration time step

**Emergent complexity**

Timescales of interest

fs ps ns $\mu$s ms s

# Data as the 4th pillar of science




BiG max MAX PLANCK RESEARCH NETWORK on big-data-driven materials science

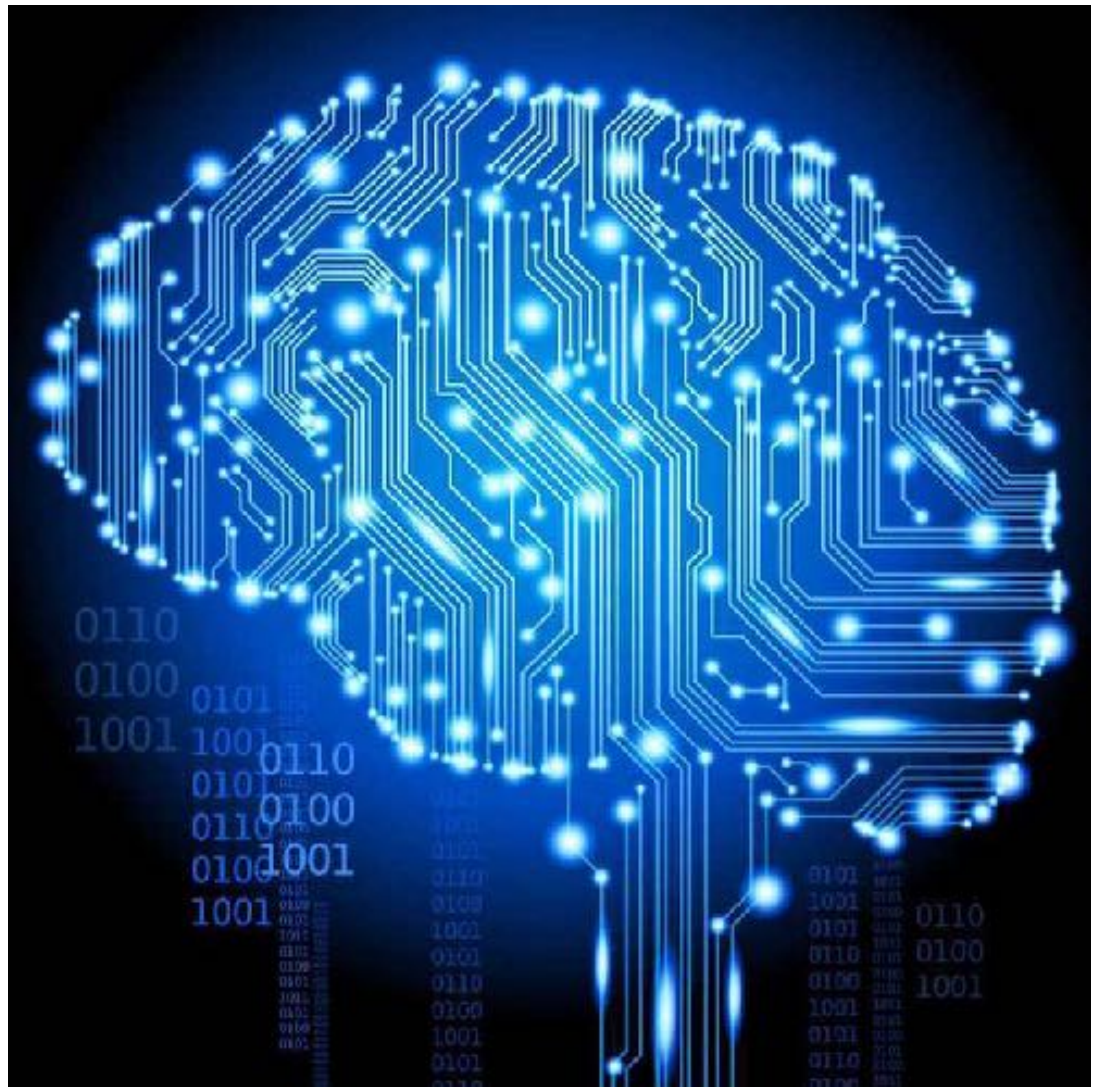
THEORY GROUP

A. Agrawal and A. Choudhary APL Mater. 4 053208 (2016); https://www.bigmax.mpg.de

# Data science

Hardware

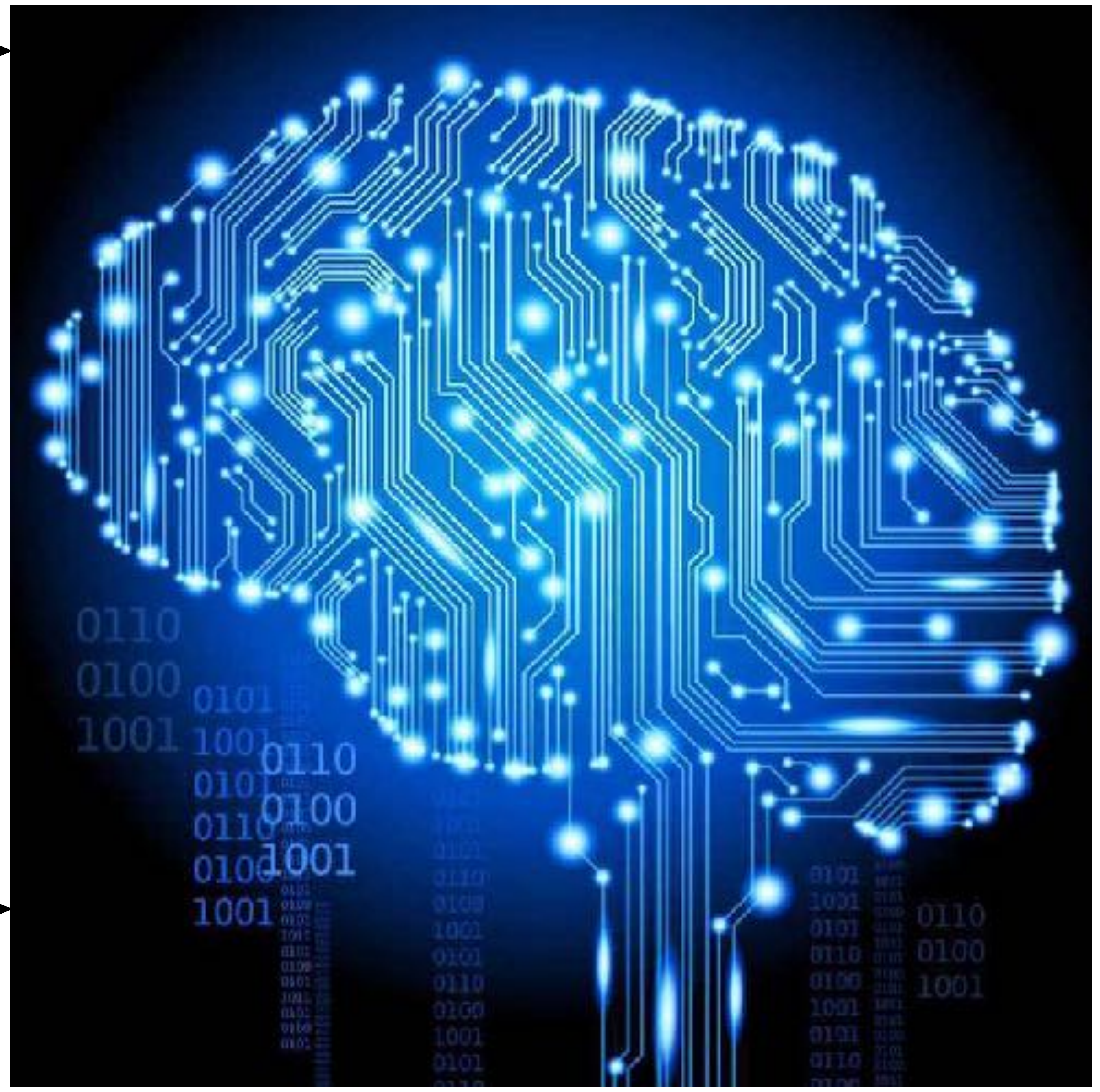
Machine learning



Databases

# Data science


Machine learning

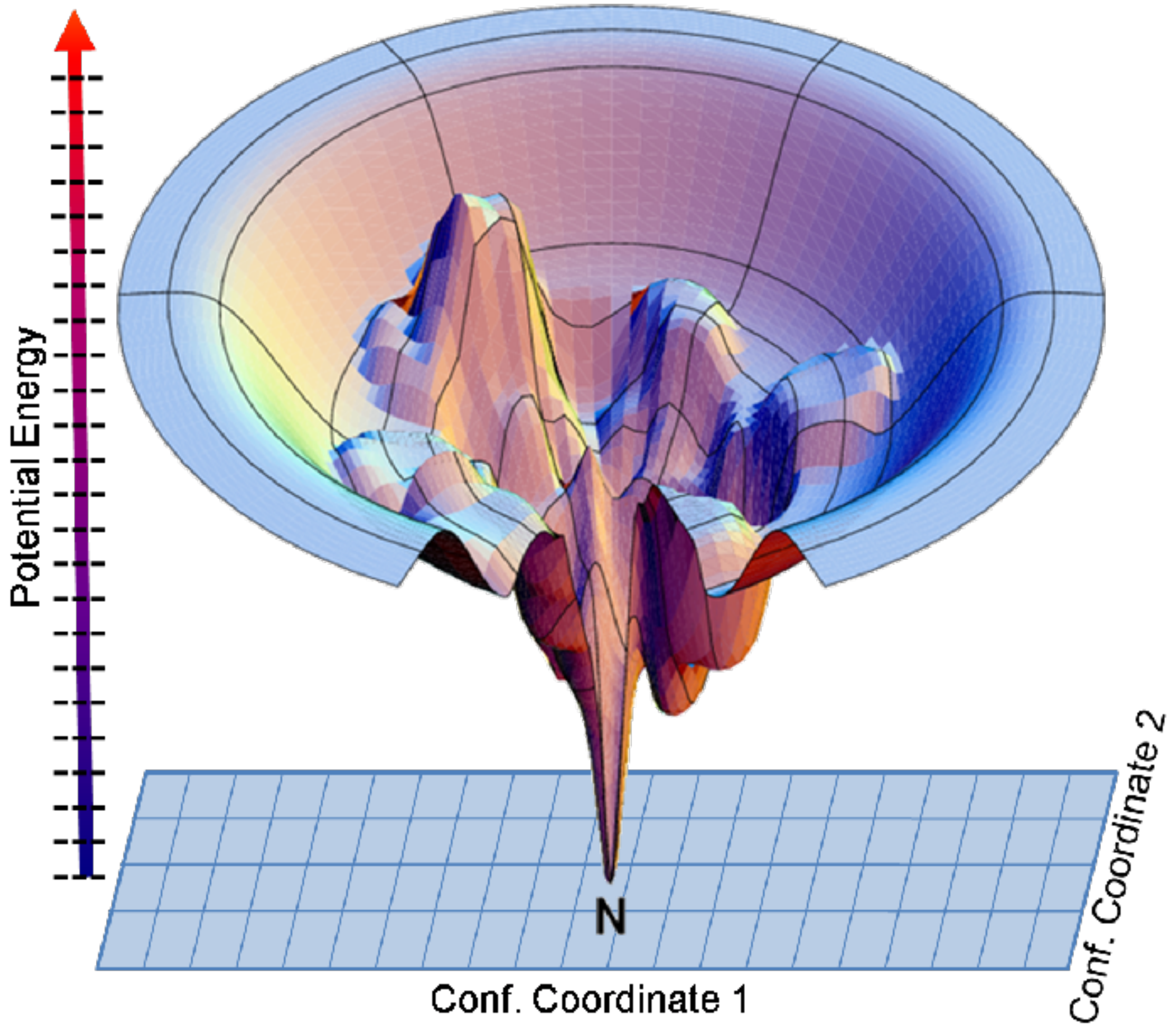



Databa

# Links to machine learning

## Interpolation of a high-dimensional function

Potential energy surface



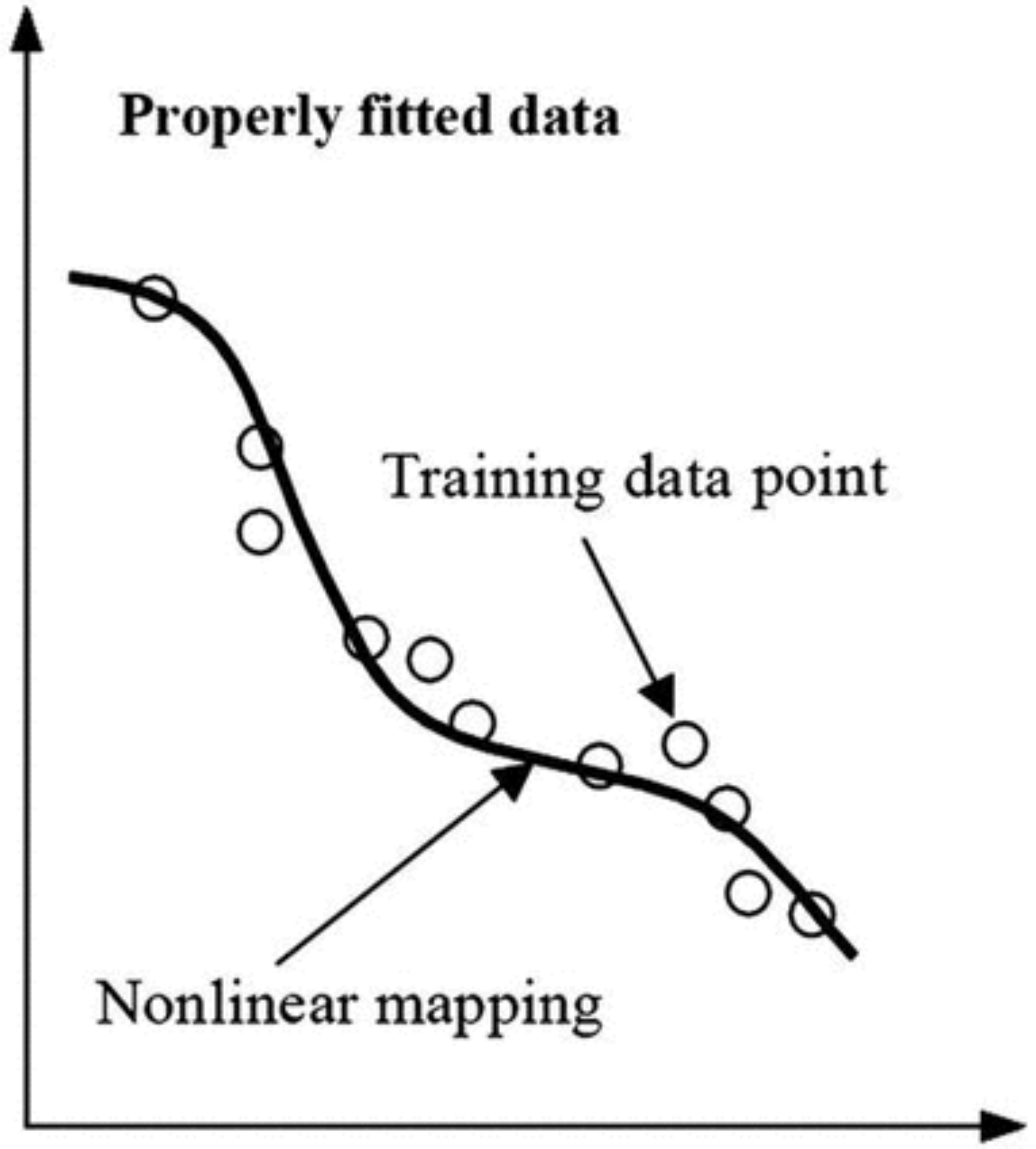

Can we build a more accurate PES?

Can we **easily** build an accurate PES?

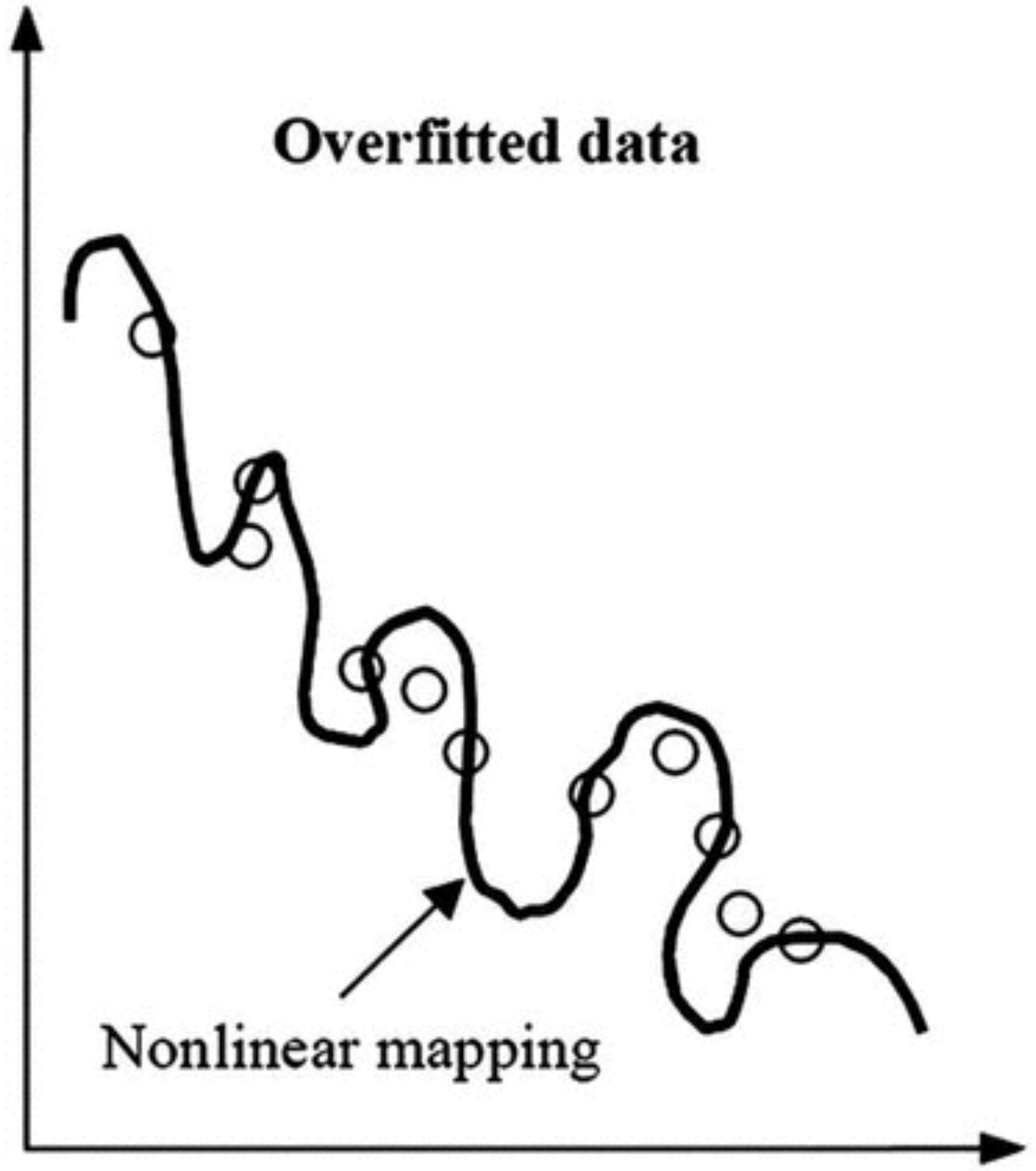
Can we make the numerical integration faster and/or more efficient?

…

# Teaser: let’s fit data points

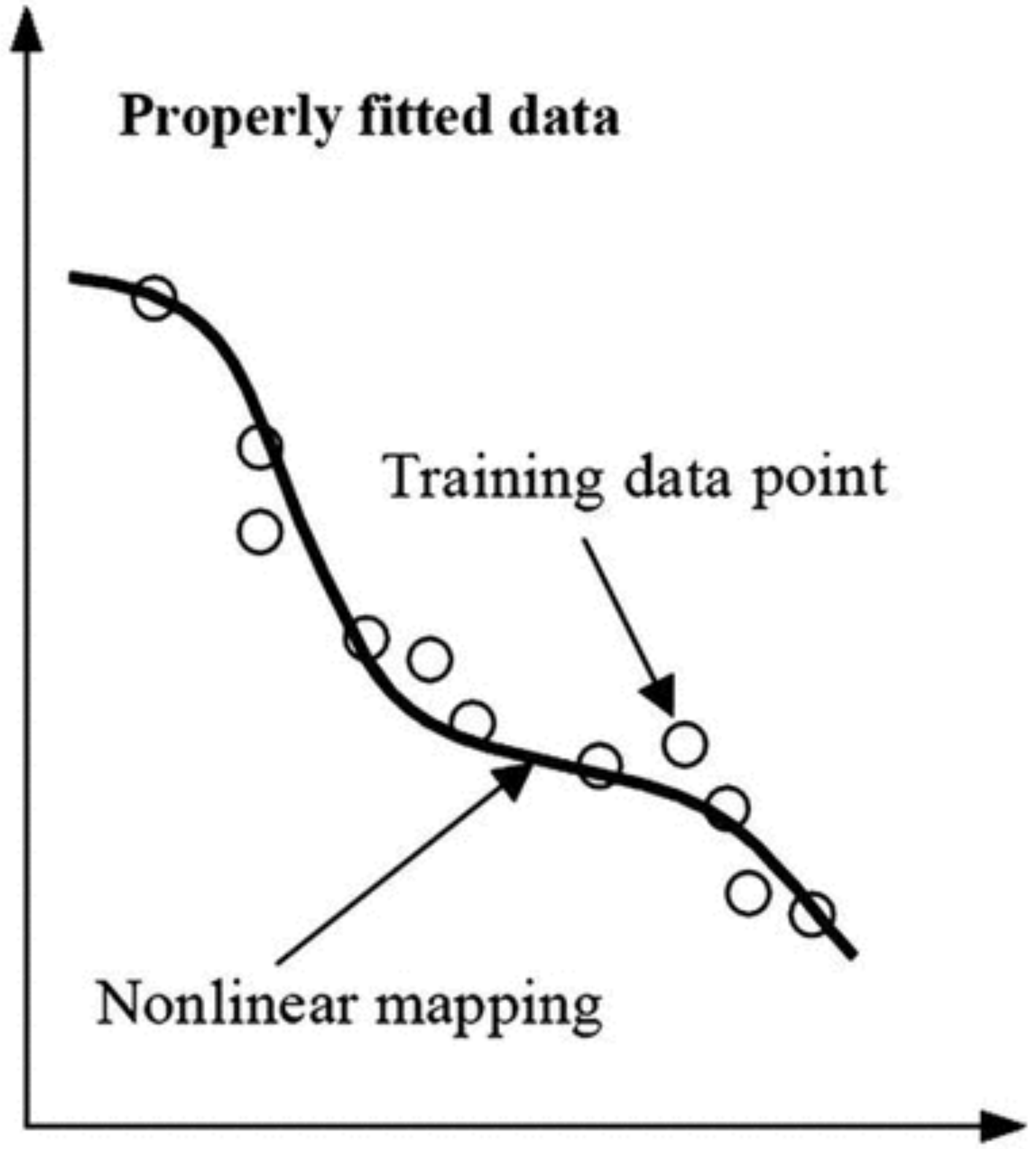



Good

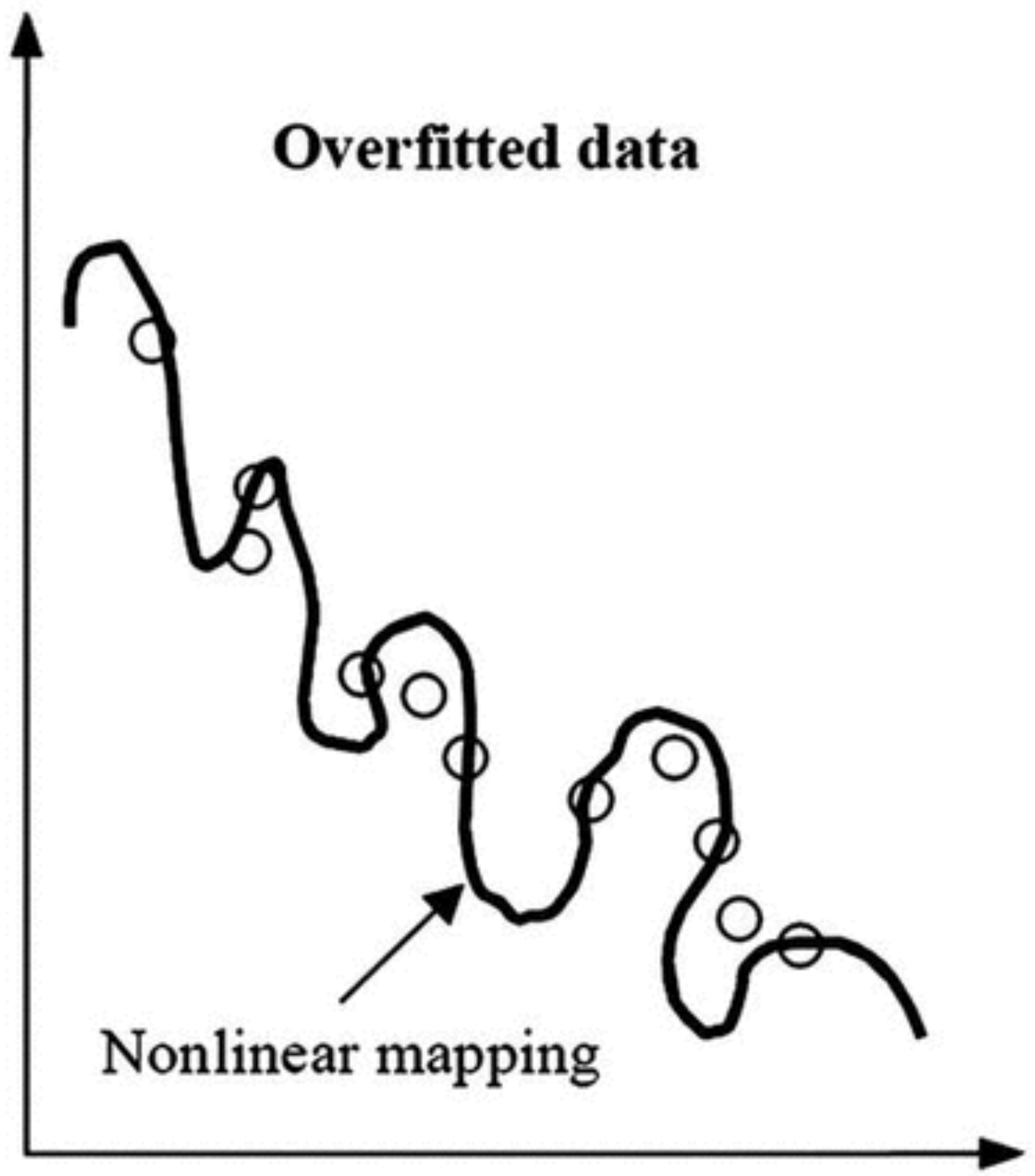



Not good

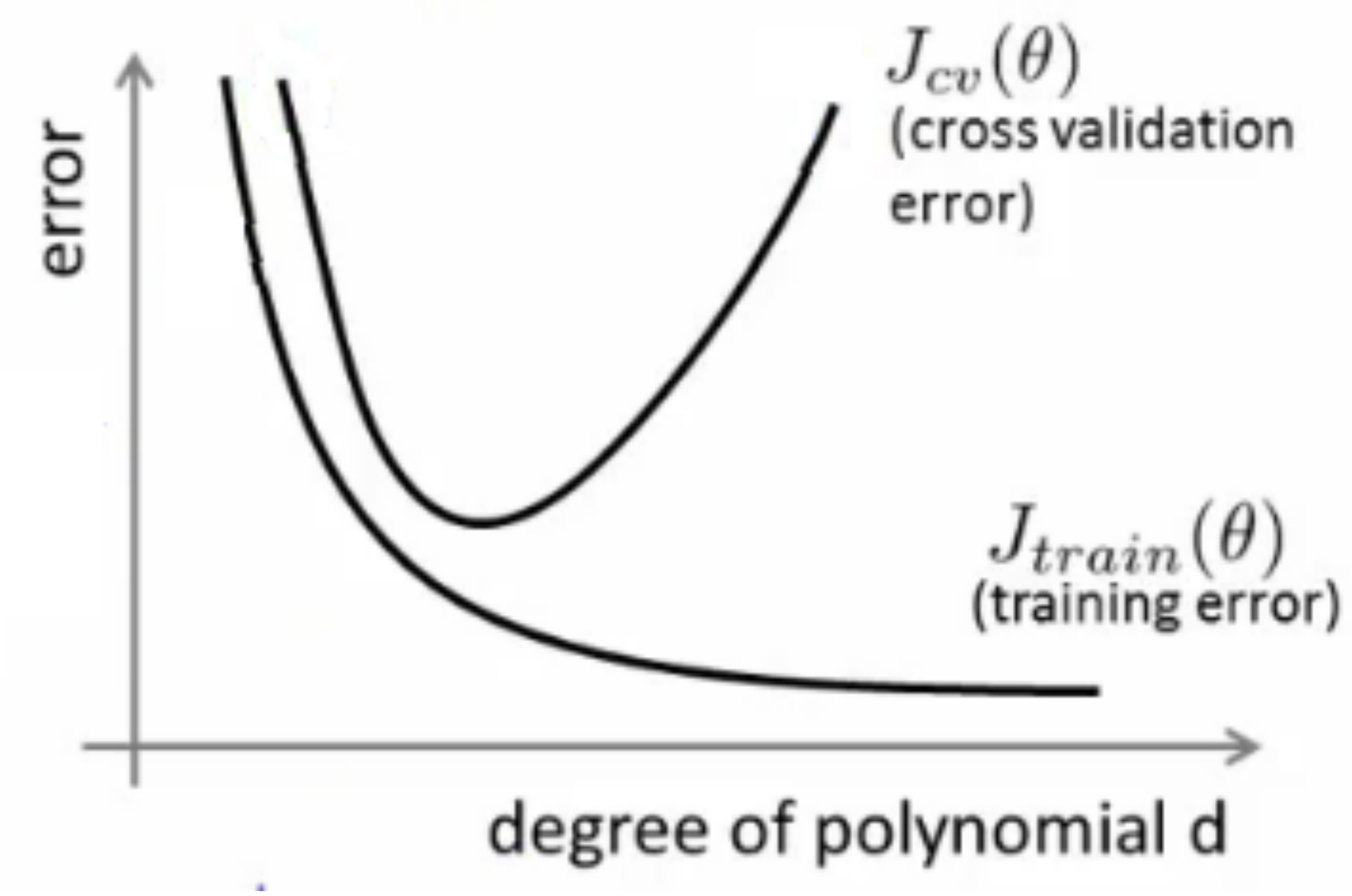
# Teaser: let's fit data points



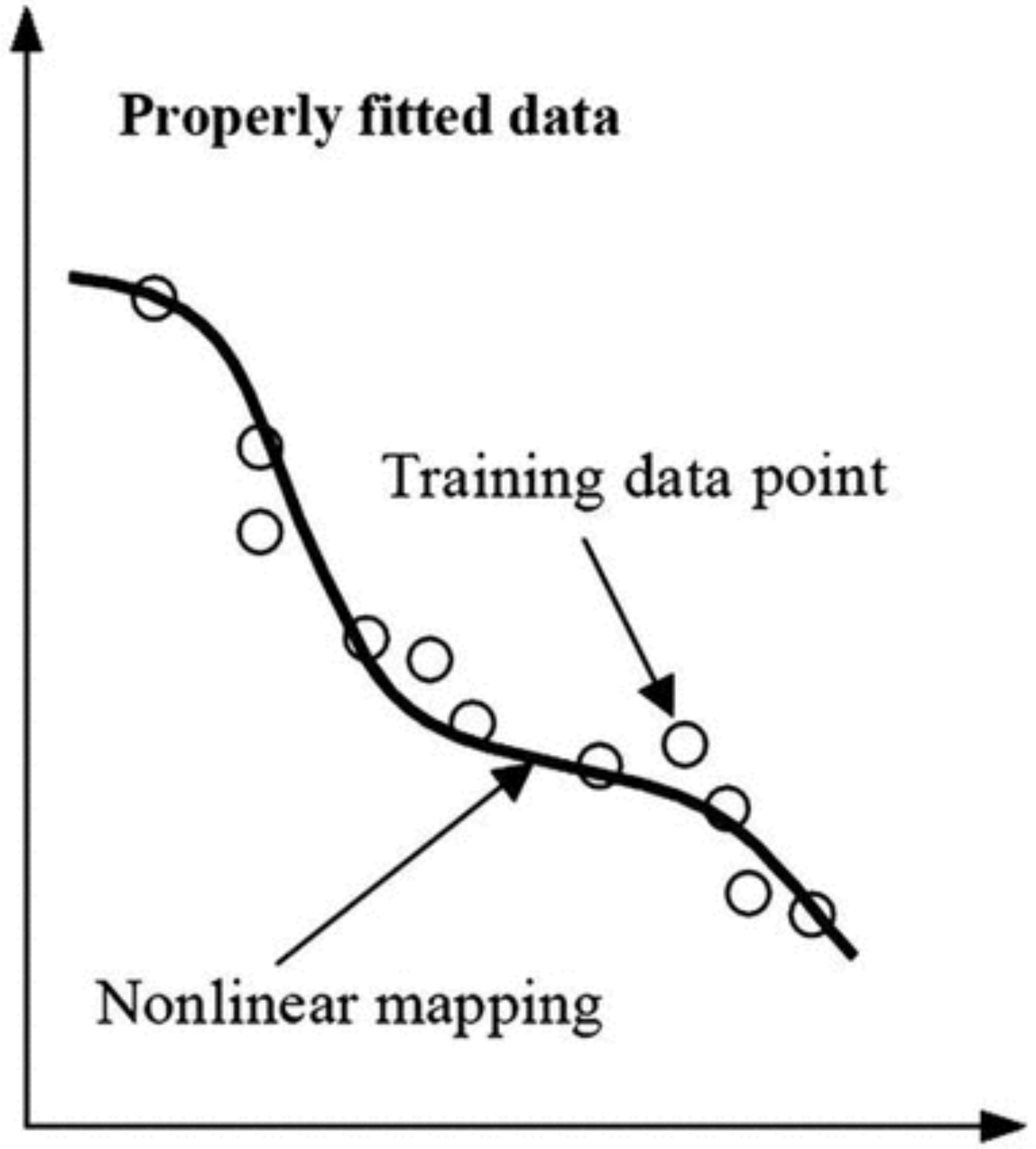

Good



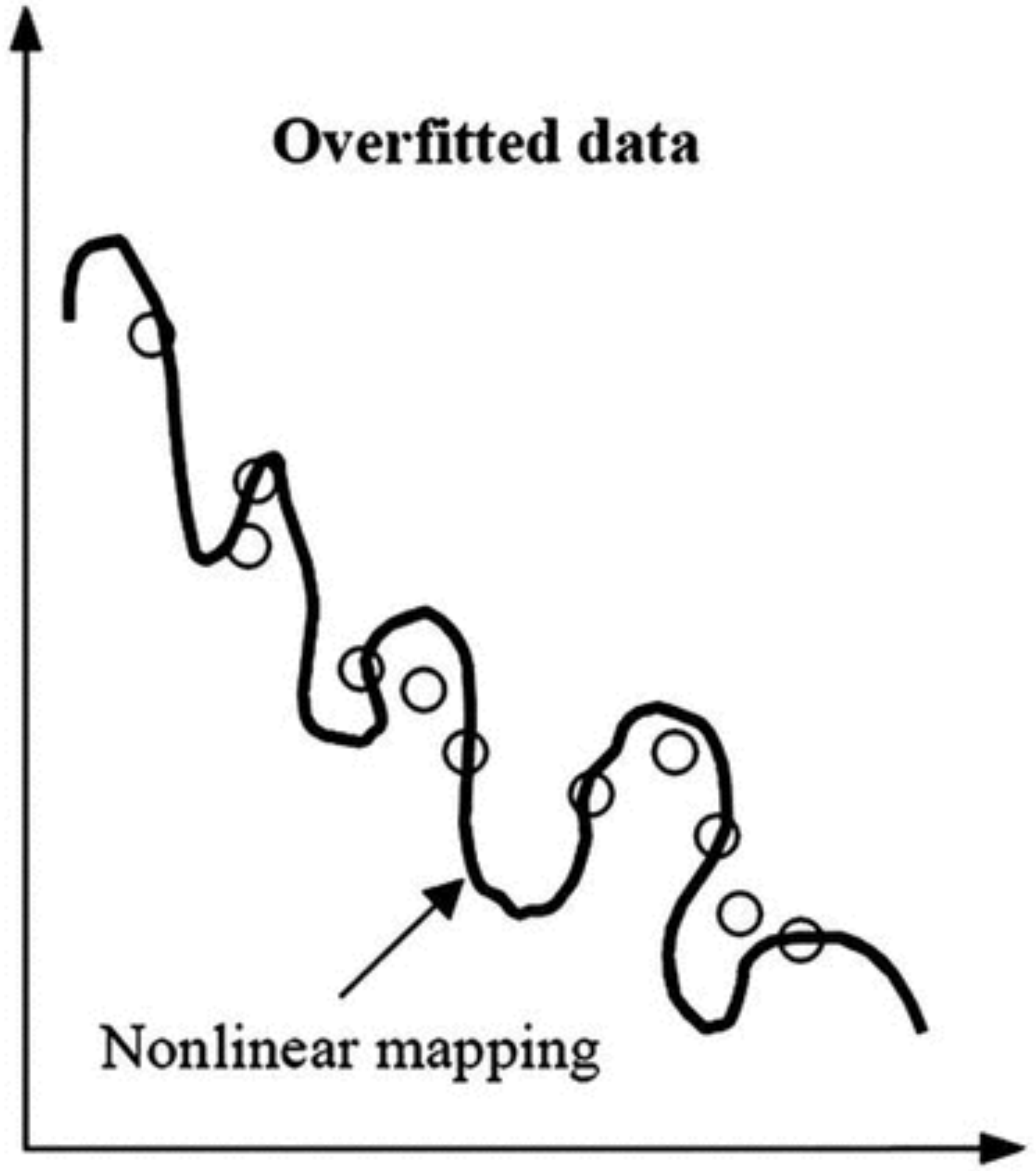

Not good



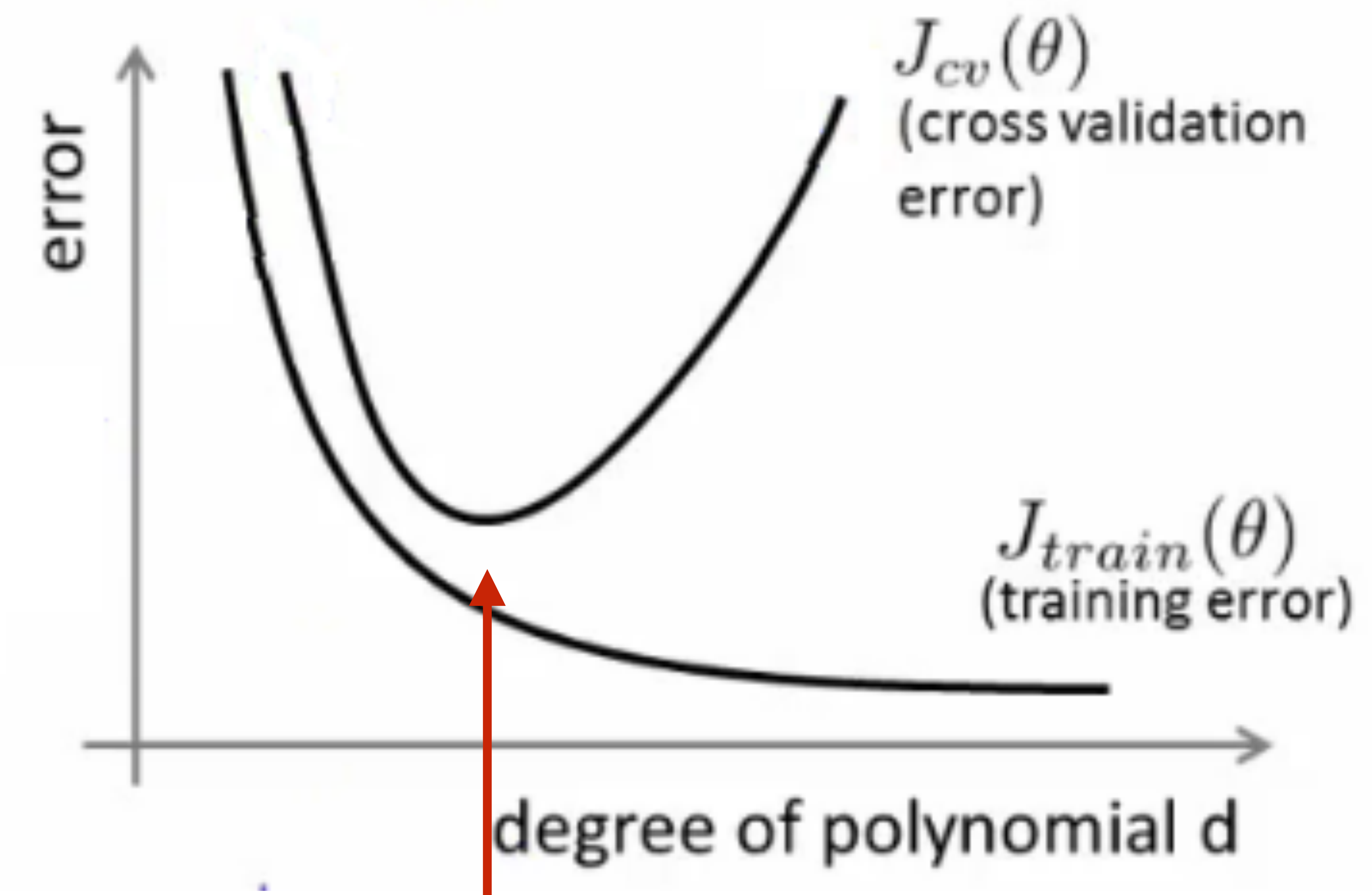
# Teaser: let’s fit data points




Good

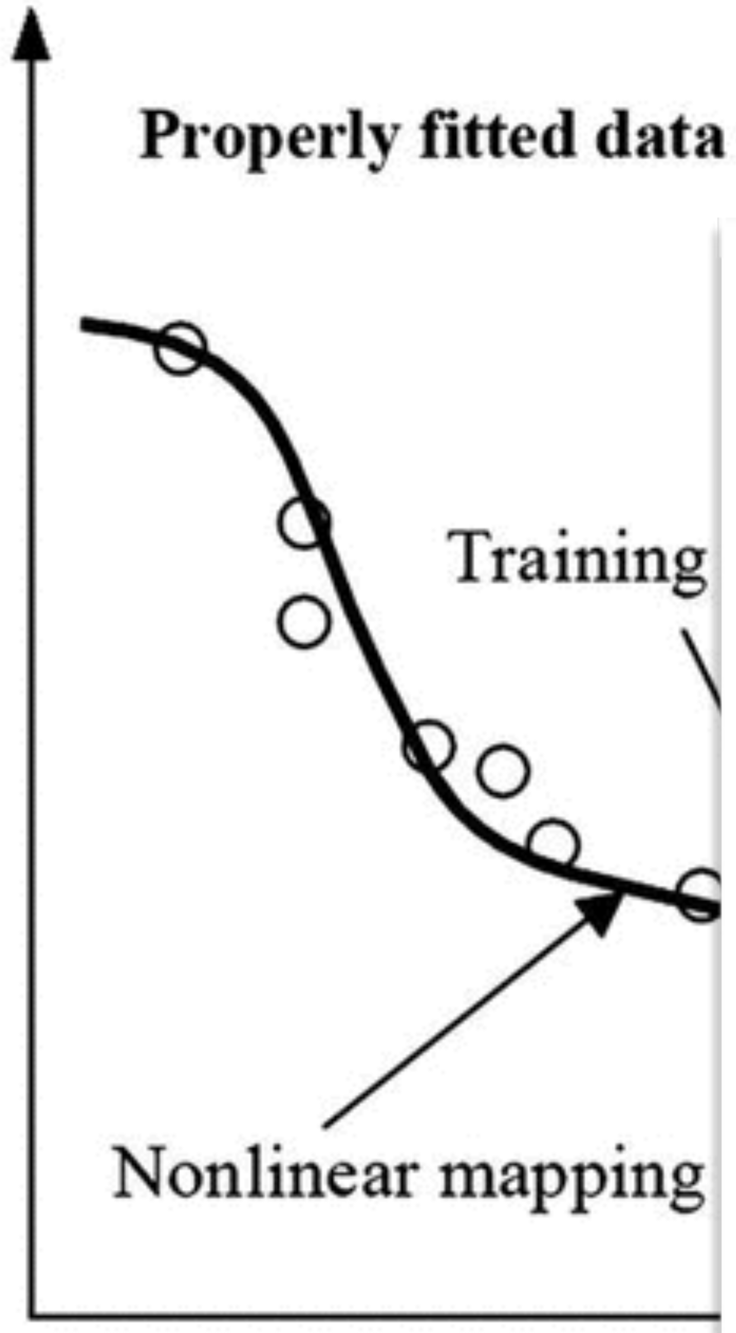



Not good

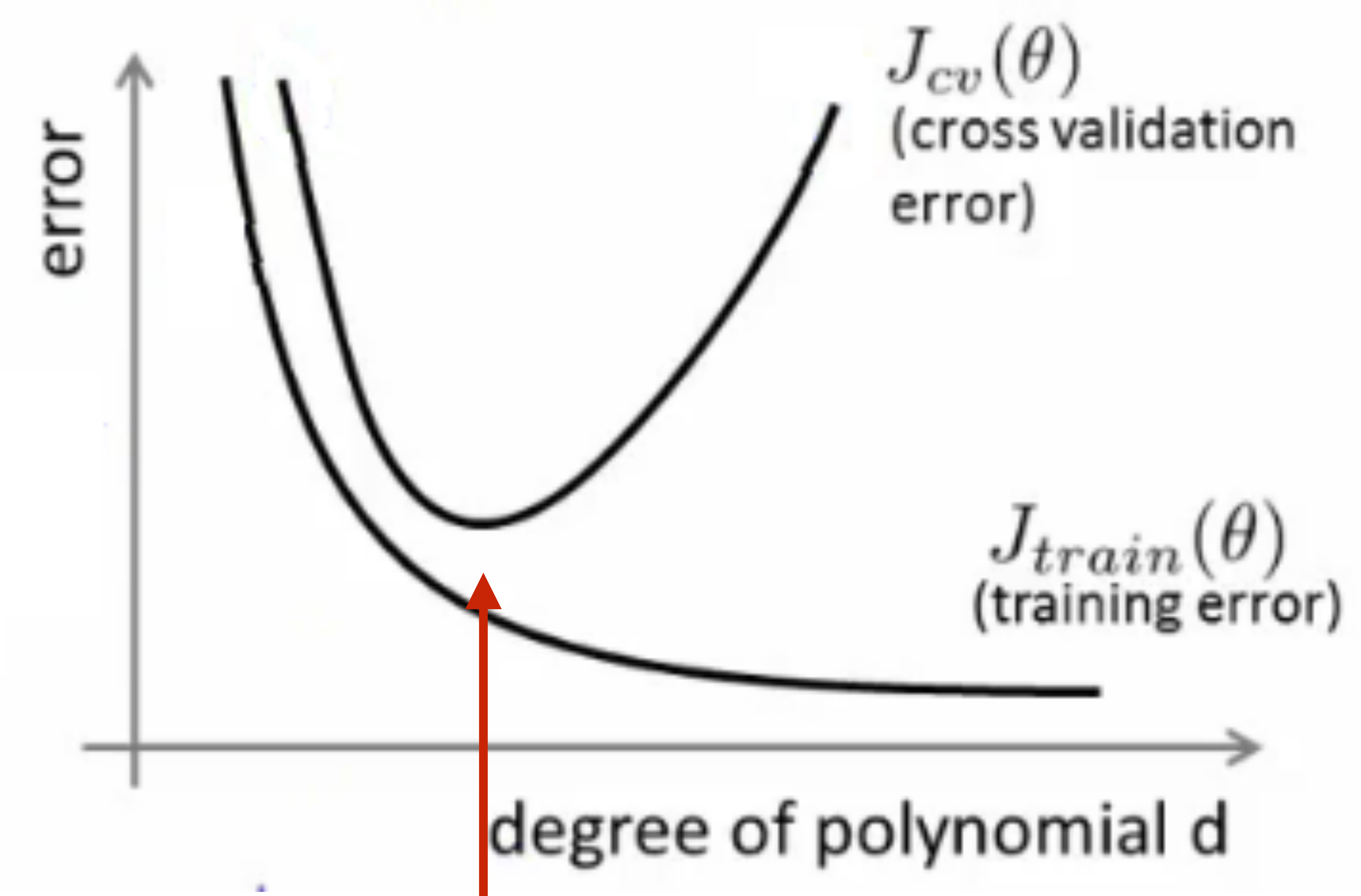



I want to be here

# Teaser: let's fit data points








I want to be here



ruthlessly stolen from A. von Lilienfeld

# Multivariate function approximation

Sparse data $\longrightarrow$ infer smooth function

# Multivariate function approximation

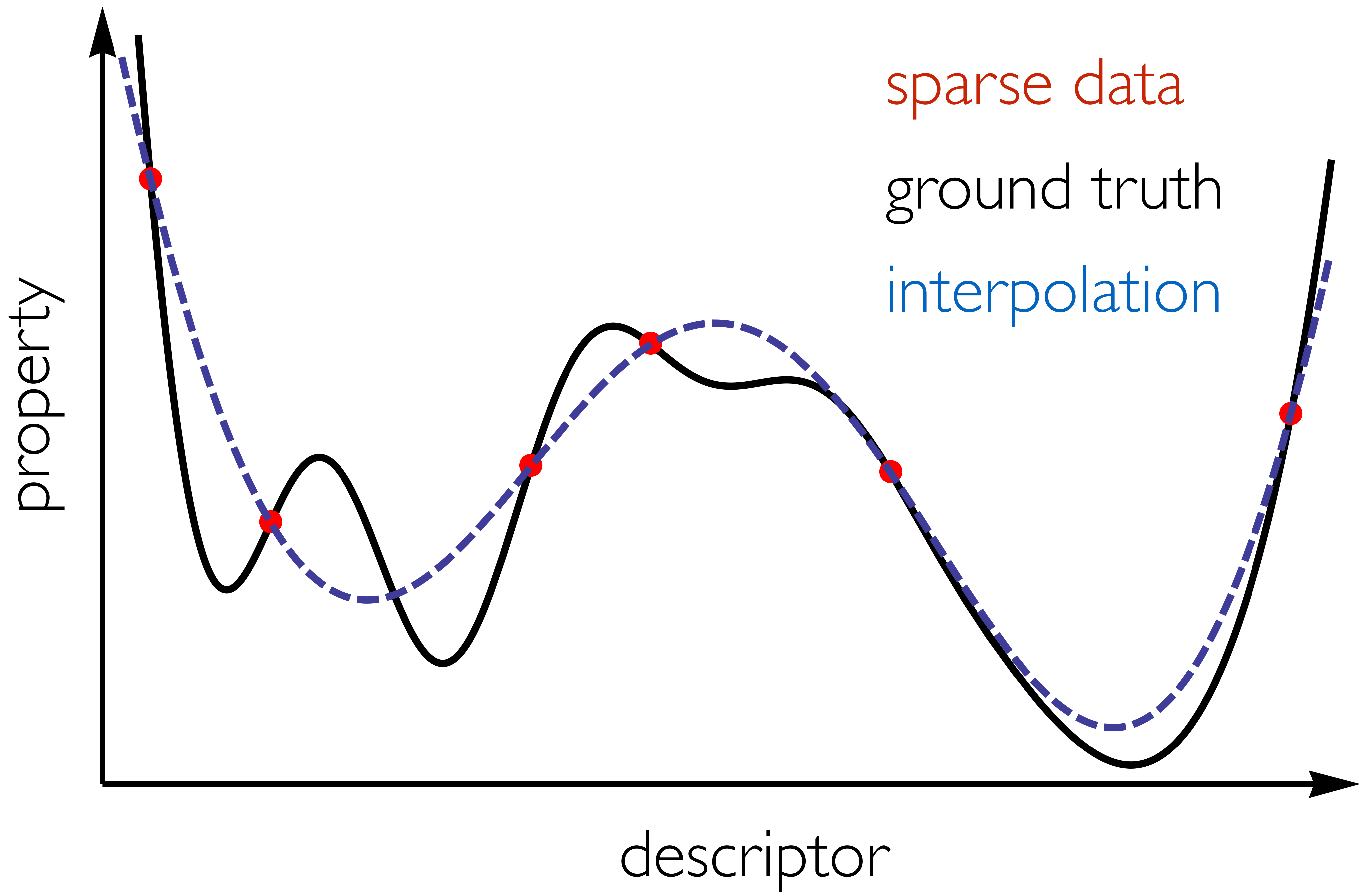
"inverse problems"

Sparse data $\longrightarrow$ infer smooth function

# Multivariate function approximation

"inverse problems"

Sparse data → infer smooth function

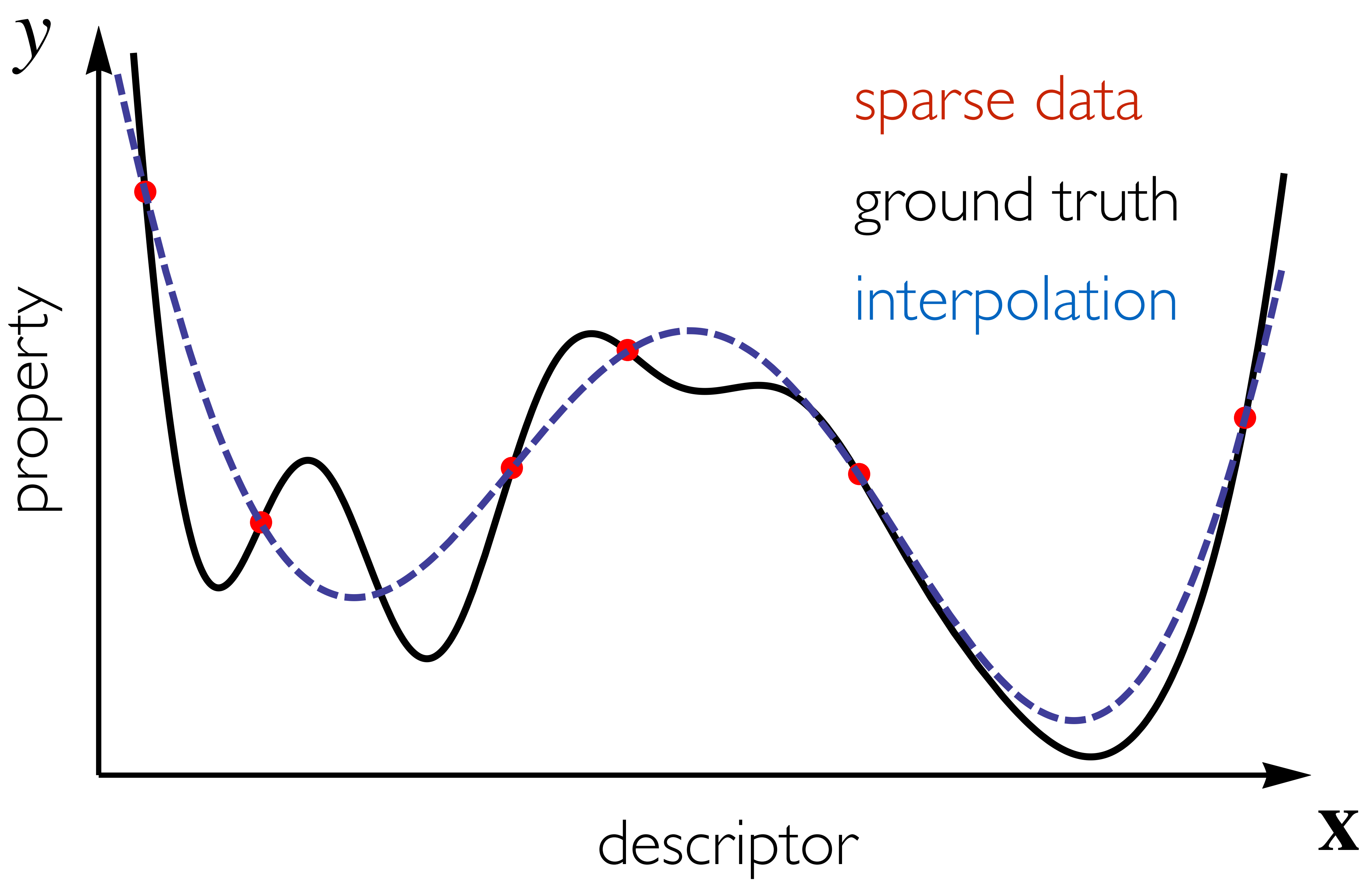



 Rupp, *International Journal of Quantum Chemistry* **115** (2015)

# Multivariate function approximation

"inverse problems"

Sparse data → infer smooth function




**Regression**:

prediction of $f : \mathbb{R}^d \to \mathbb{R}$

based on noisy data points

$$D = (\mathbf{X}, \mathbf{y}) = \{\mathbf{x}_n, y_n\}_{n=1}^{N}$$

$$y_n = f(\mathbf{x}_n) + \varepsilon$$

What is $f$?

THEORY GROUP Rupp, *International Journal of Quantum Chemistry* **115** (2015)

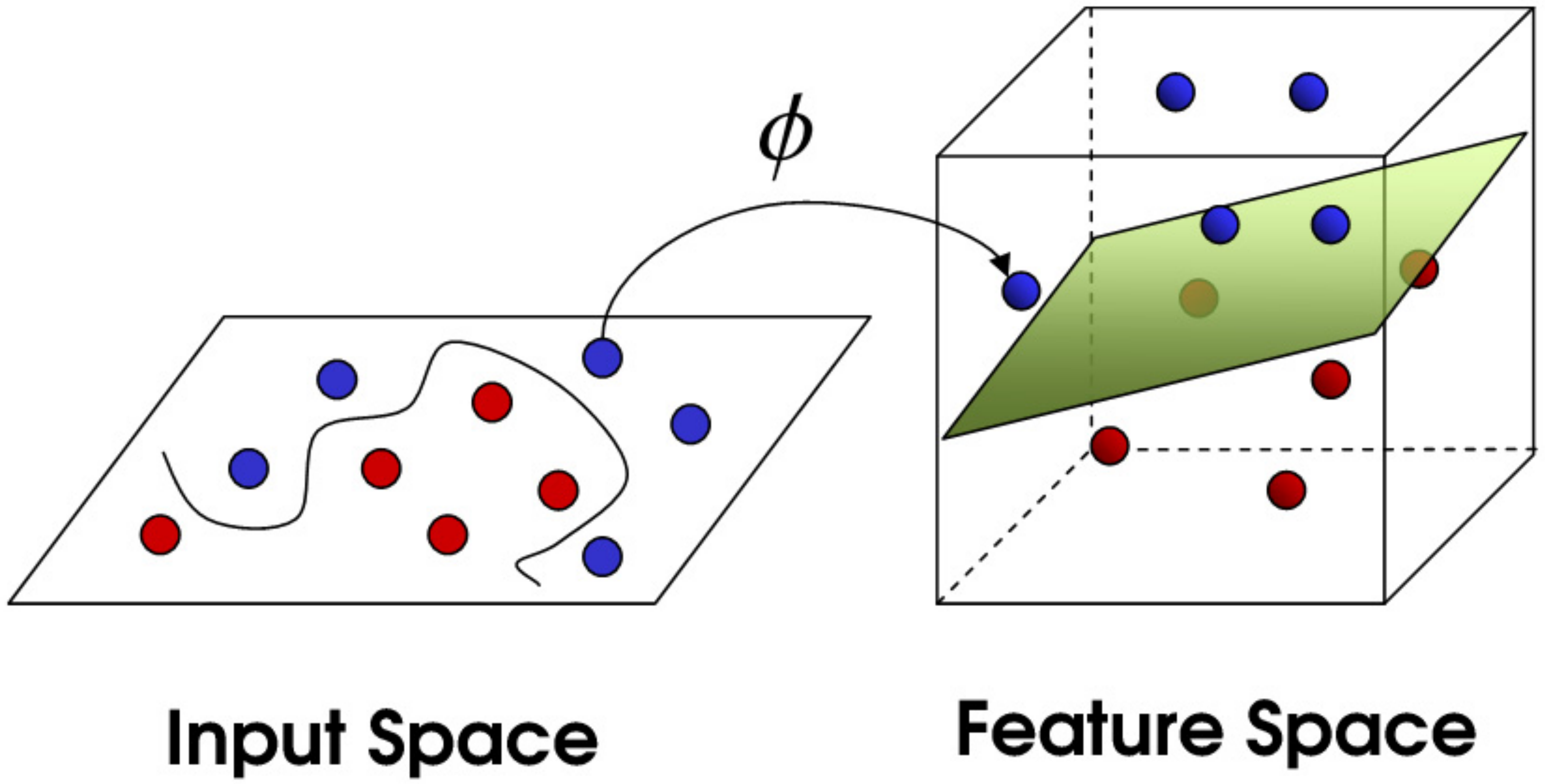
# Kernel methods are vintage

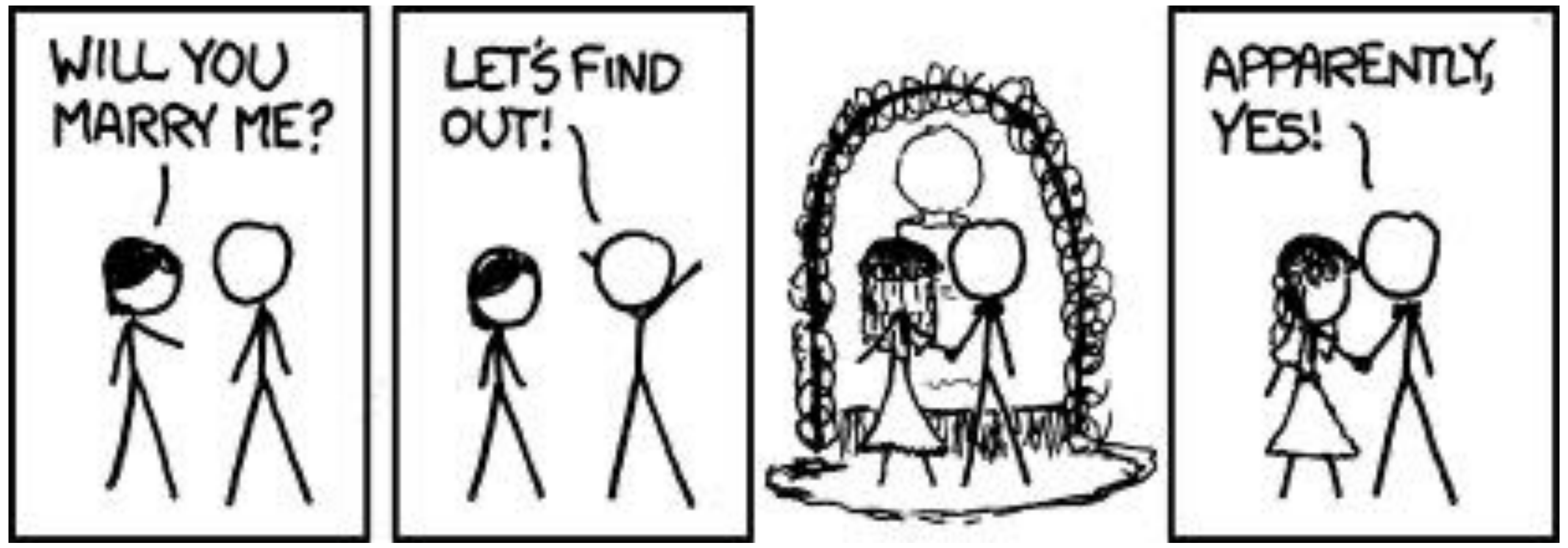
Kernel
- needs a representation
- linear algebra
- can be efficient with small data

Deep learning
- learns the representation
- complex mathematical structure
- data hungry

# Learning from experience

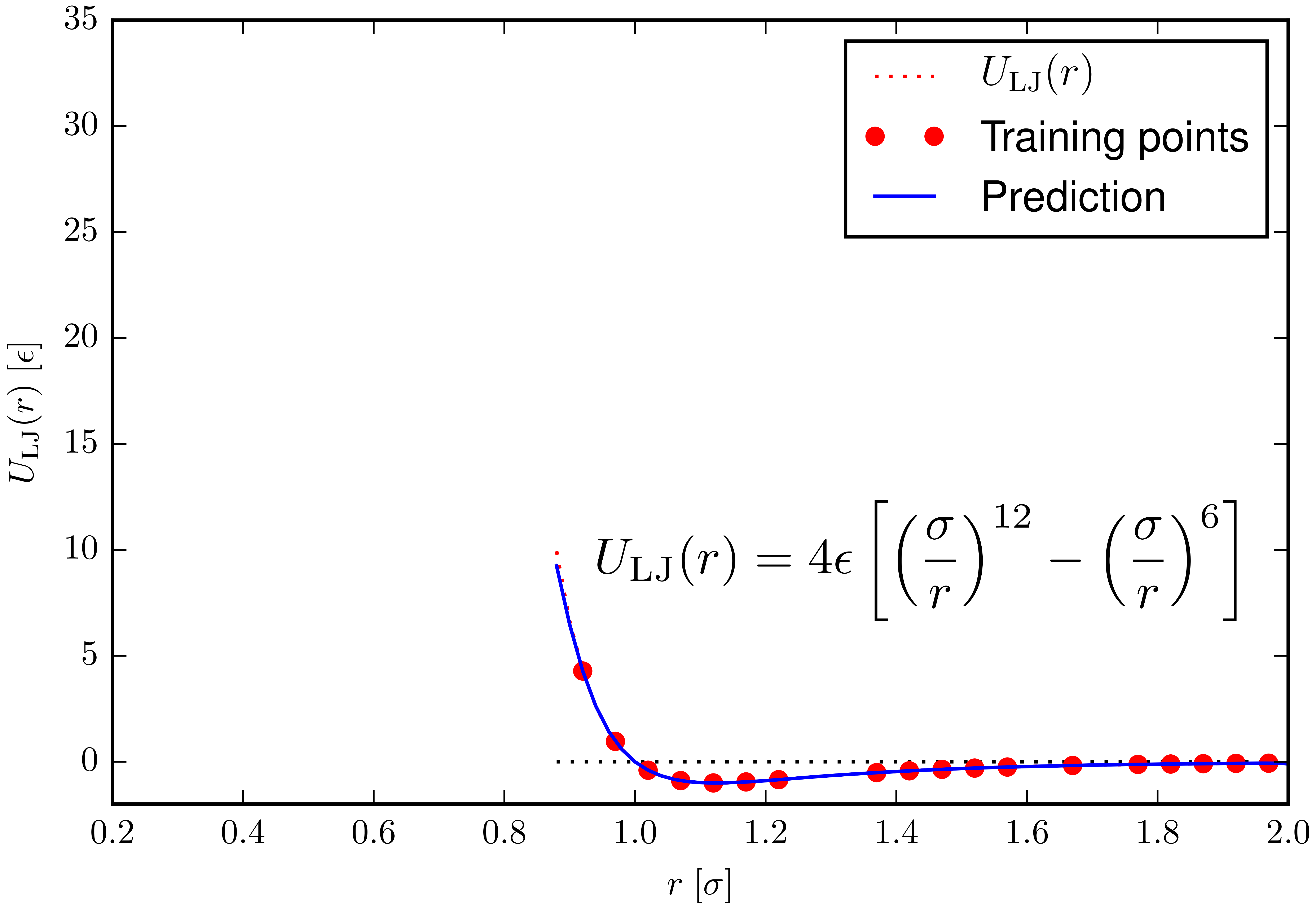
Inductive (based on examples)



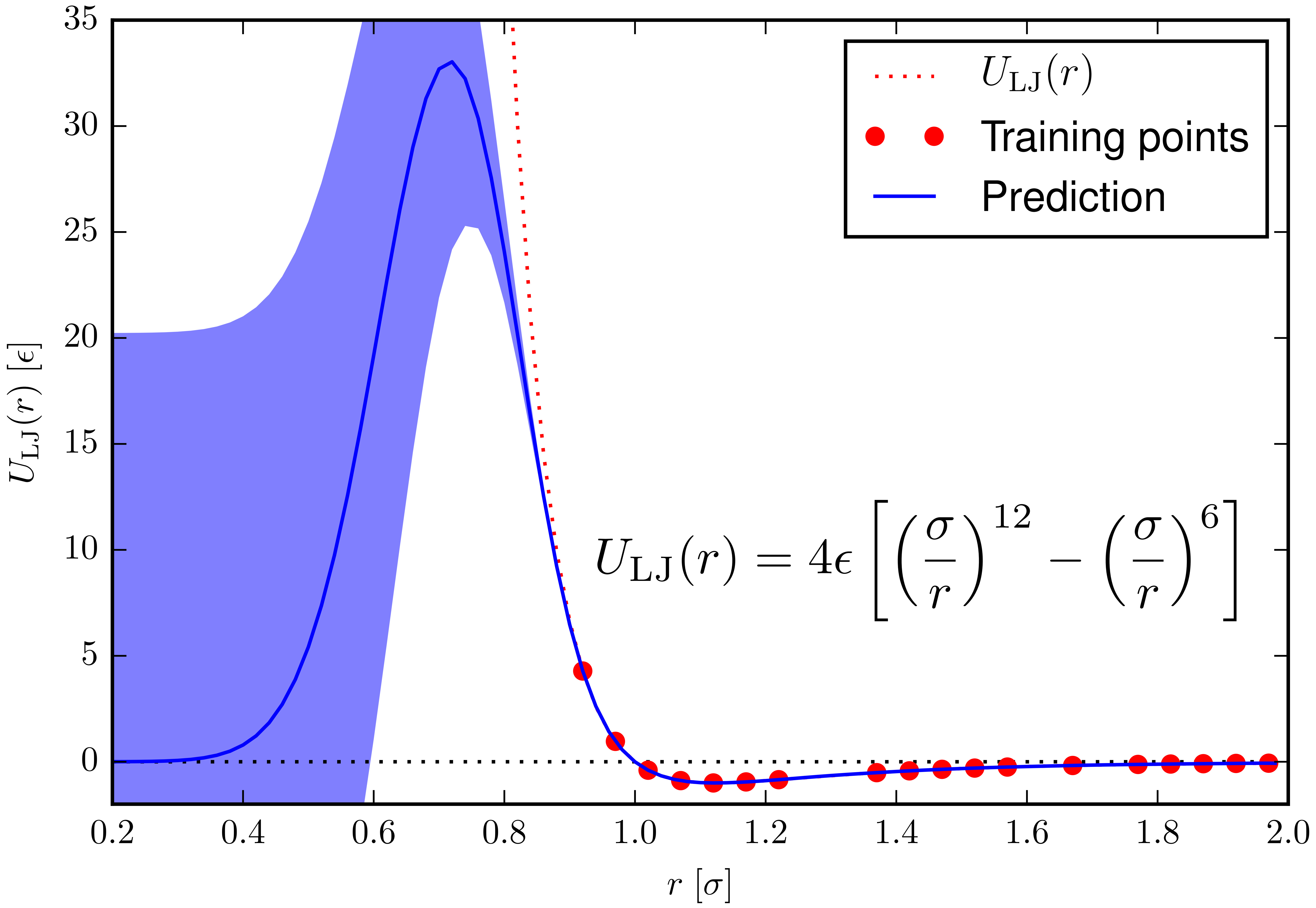

source: xkcd

# Extrapolation in machine learning




THEORY GROUP

# Extrapolation in machine learning




THEORY GROUP

# Bayesian inference

**Prediction** $\left\{ \begin{array}{l} \text{Prior beliefs} \\ \\ \text{Sampled data} \end{array} \right.$

THEORY GROUP

# Bayesian inference

**Prediction** $\left\{ \begin{array}{l} \text{Prior beliefs} \\ \\ \text{Sampled data} \end{array} \right.$

$f$ training values

$f^*$ test values

$y$ observations

THEORY GROUP

# Bayesian inference

**Prediction** $\begin{cases} \text{Prior beliefs} \\ \text{Sampled data} \end{cases}$

$f$ training values

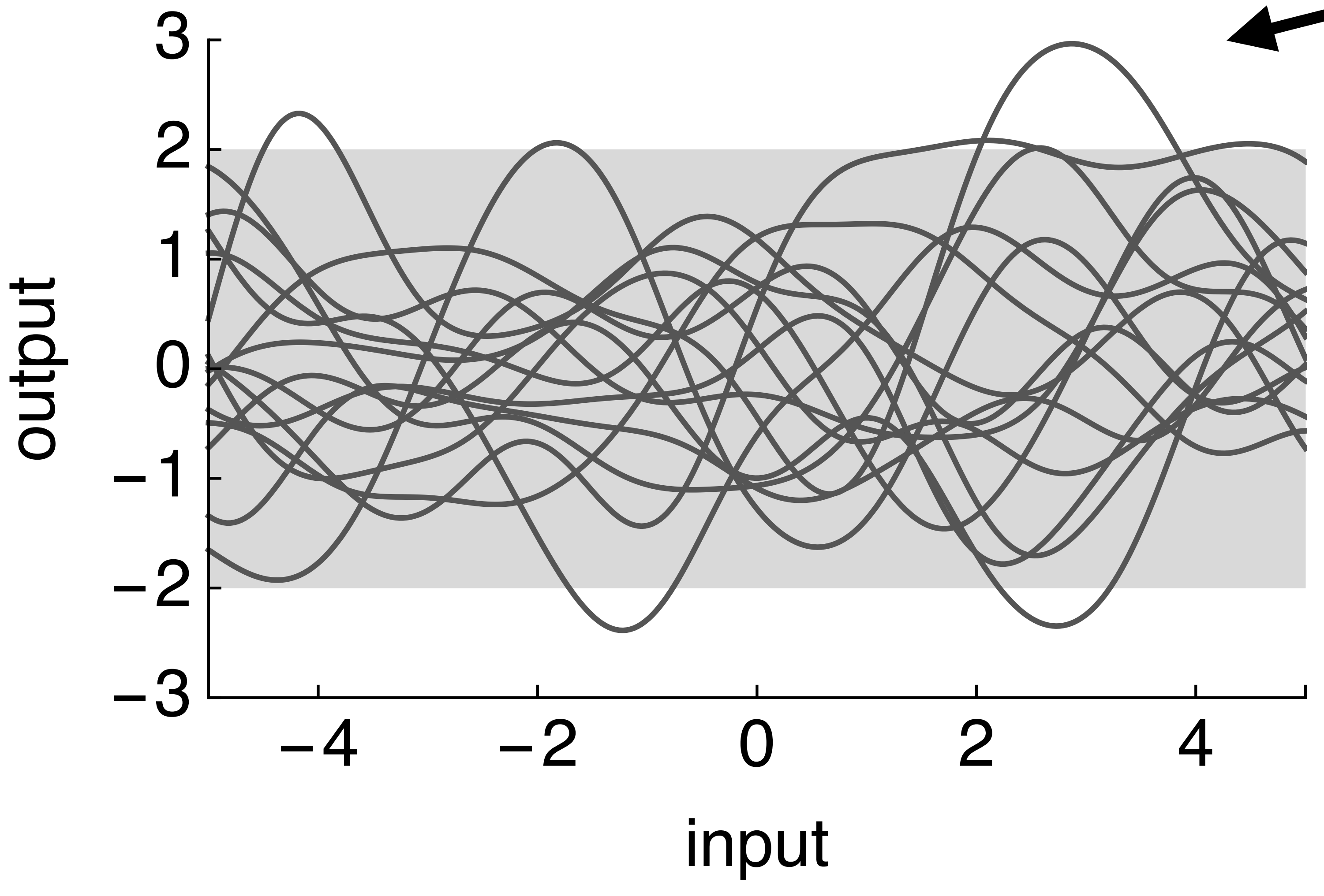
$f^*$ test values

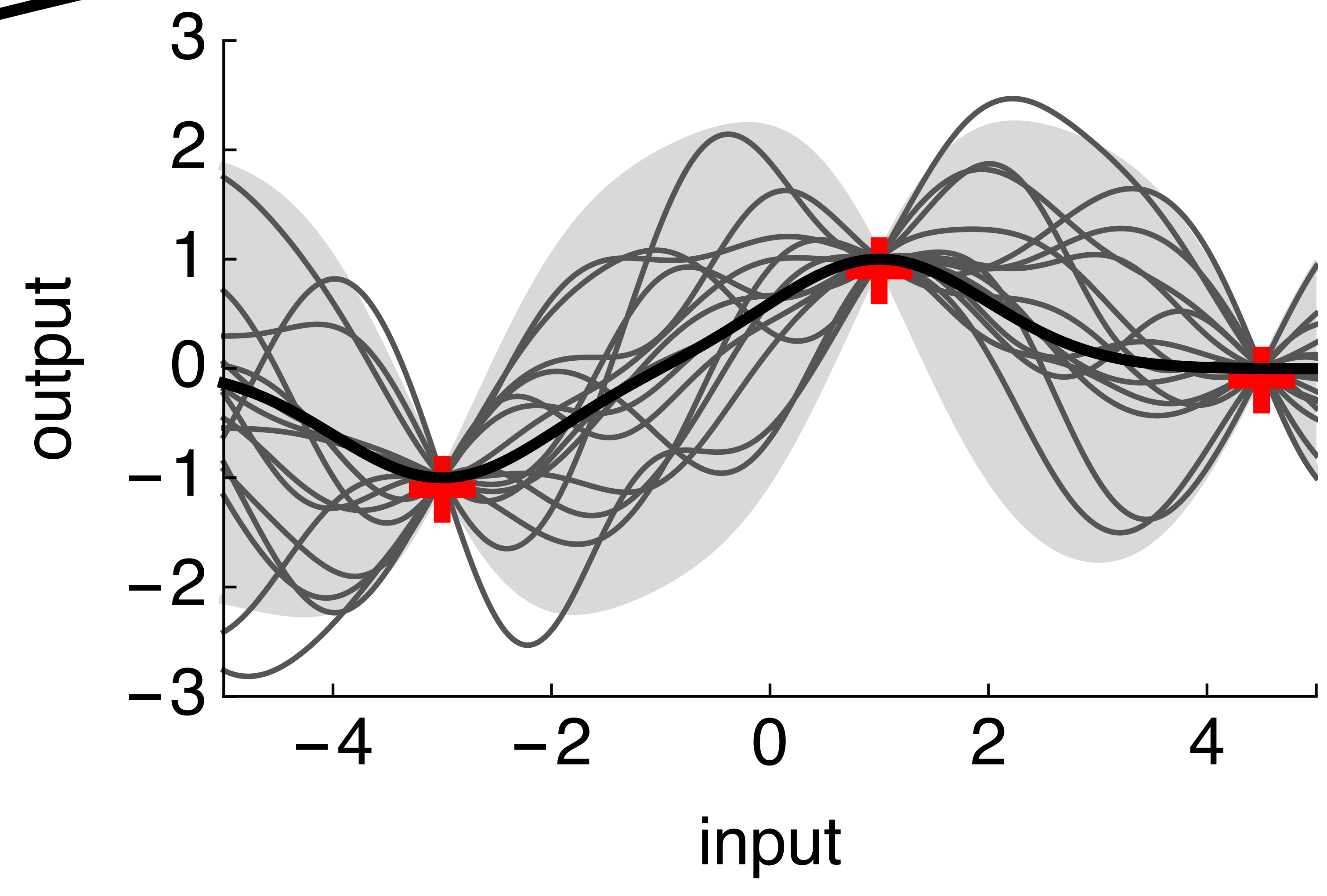
$y$ observations

**Bayes' formula**

$$\underbrace{p(f,f^*\,|\,y)}_{\text{posterior}} = \frac{\overbrace{p(y\,|\,f)}^{\text{likelihood}}\ \overbrace{p(f,f^*)}^{\text{prior}}}{\underbrace{p(y)}_{\text{normalization}}}$$

# Bayesian inference




**Bayes' formula**

likelihood

prior

$$p(f,f^*|y) = \frac{p(y|f)\;p(f,f^*)}{p(y)}$$

posterior

normalization

Rasmussen, *Advanced lectures on machine learning*. Springer, 63-71 (2004)

# Gaussian processes

$$f \sim \mathcal{GP}(m, k)$$



Rasmussen, *Advanced lectures on machine learning*. Springer, 63-71 (2004)

# Gaussian processes

$$f \sim \mathcal{GP}(m, k)$$

$f(x)$

random variable:
value of the
stochastic function
at *x*

$\mu_i = m(x_i)$

mean

$\Sigma_{ij} = k(x_i, x_j)$

covariance



Rasmussen, *Advanced lectures on machine learning*. Springer, 63-71 (2004)

# Gaussian processes

$$f \sim \mathcal{GP}(m, k)$$

$f(x)$

random variable: value of the stochastic function at *x*

$\mu_i = m(x_i)$

mean

$\Sigma_{ij} = k(x_i, x_j)$

covariance

kernel

target property

$$\mathbf{K}\alpha = p$$



Rasmussen, *Advanced lectures on machine learning*. Springer, 63-71 (2004)

# Linear-ridge vs kernel-ridge regression

Linear-ridge regression

$$\mathbf{A}x = b$$

kernel-ridge regression (ML)

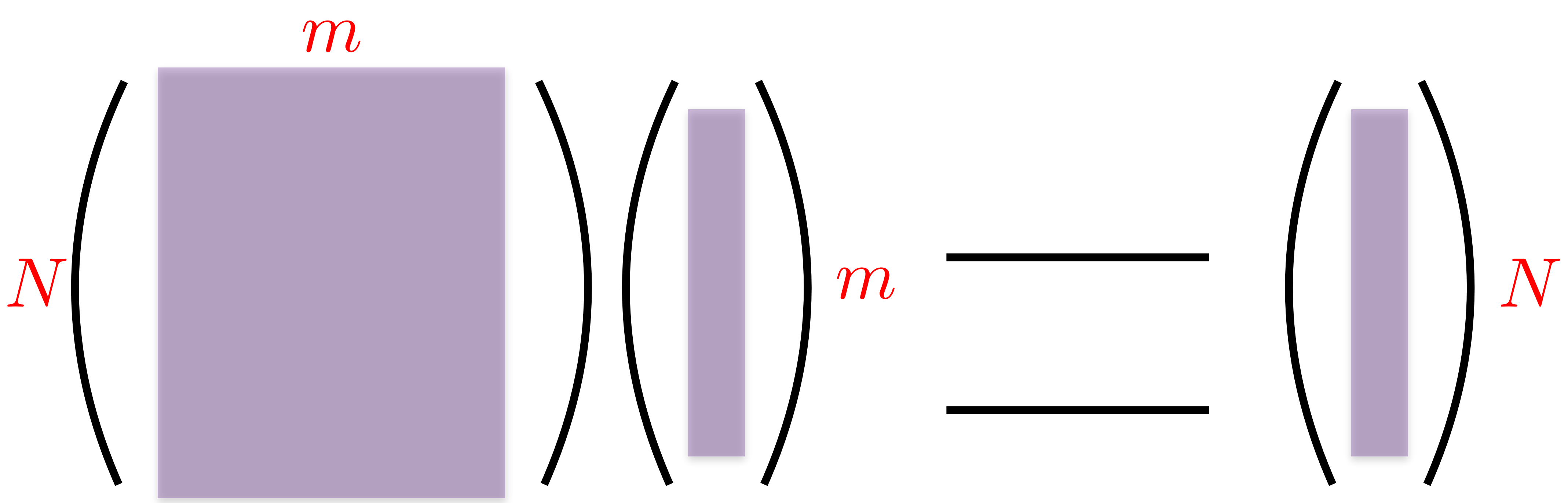
$$\mathbf{K}\alpha = p$$

THEORY GROUP

# Linear-ridge vs kernel-ridge regression

## Linear-ridge regression

$$\mathbf{A}x = b$$



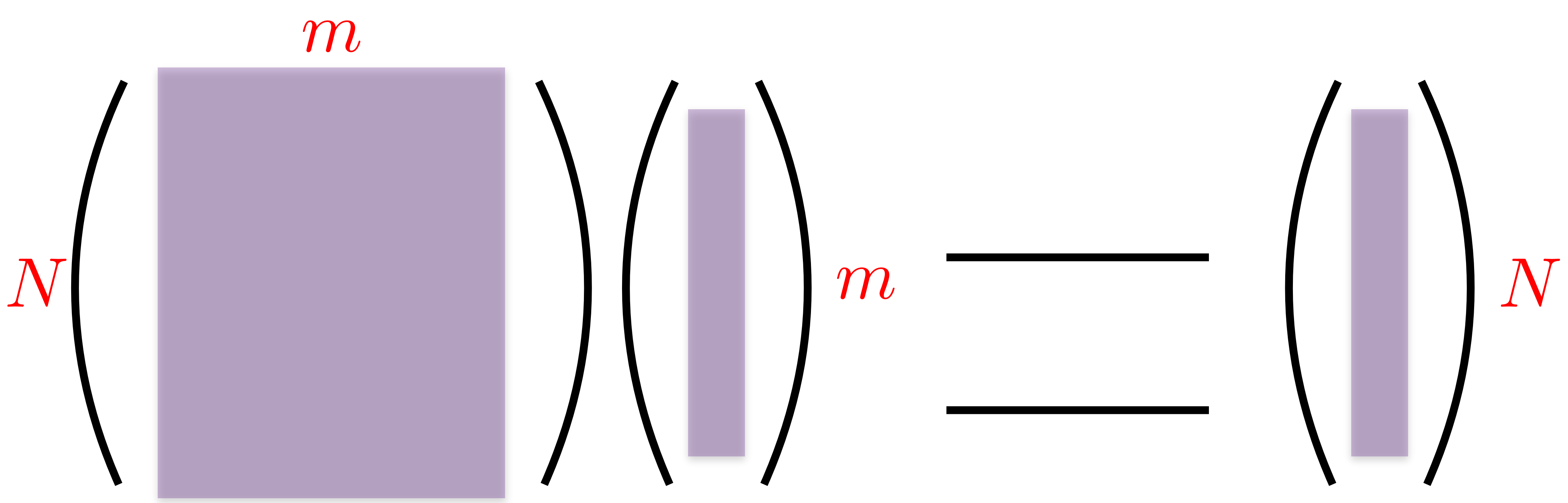

## kernel-ridge regression (ML)

$$\mathbf{K}\alpha = p$$

# Linear-ridge vs kernel-ridge regression

Linear-ridge regression

$$\mathbf{A}x = b$$




in general: $m \ll N$

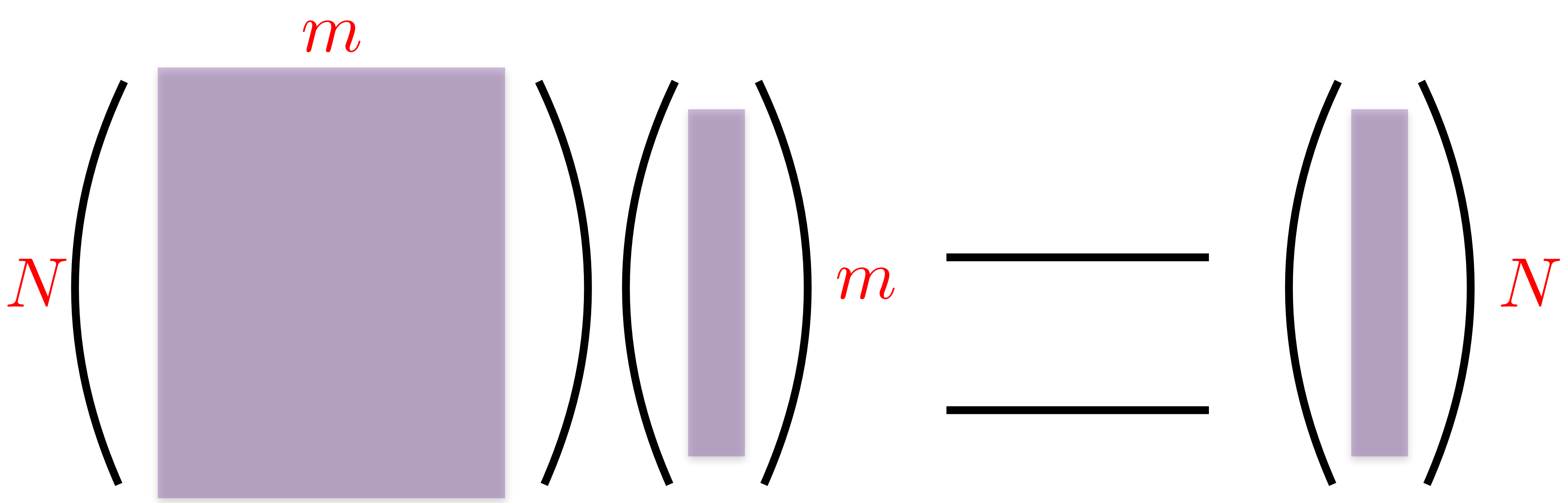
kernel-ridge regression (ML)

$$\mathbf{K}\alpha = p$$

# Linear-ridge vs kernel-ridge regression

## Linear-ridge regression

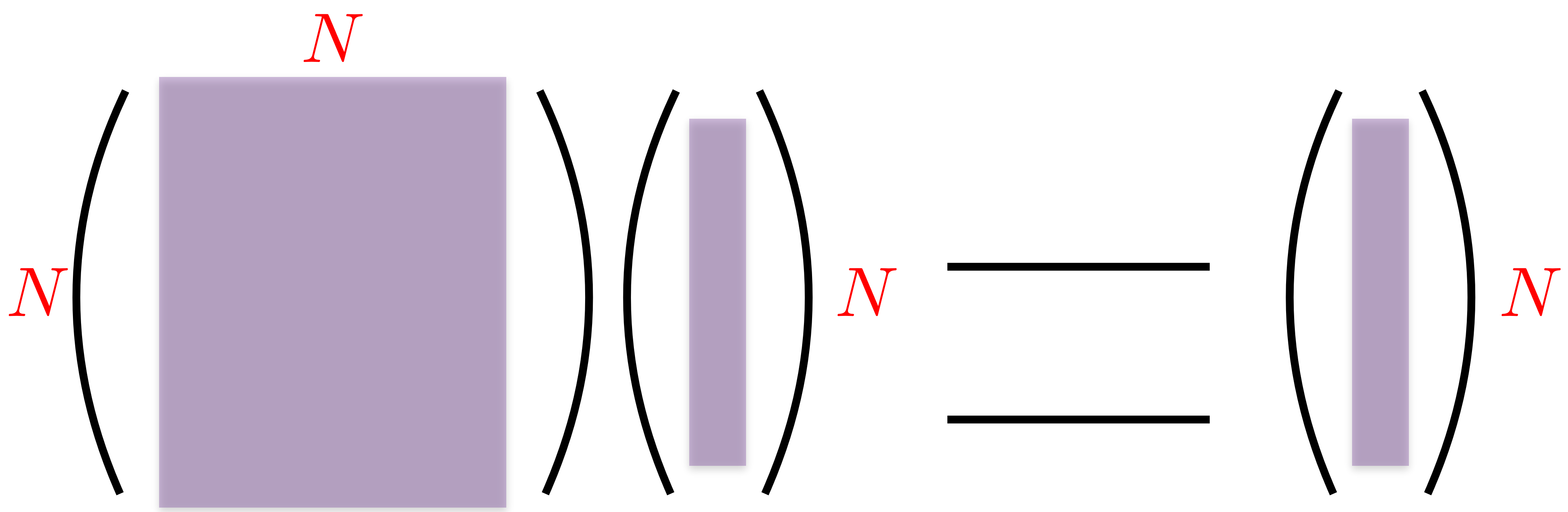
$$\mathbf{A}x = b$$




in general: $m \ll N$

## kernel-ridge regression (ML)

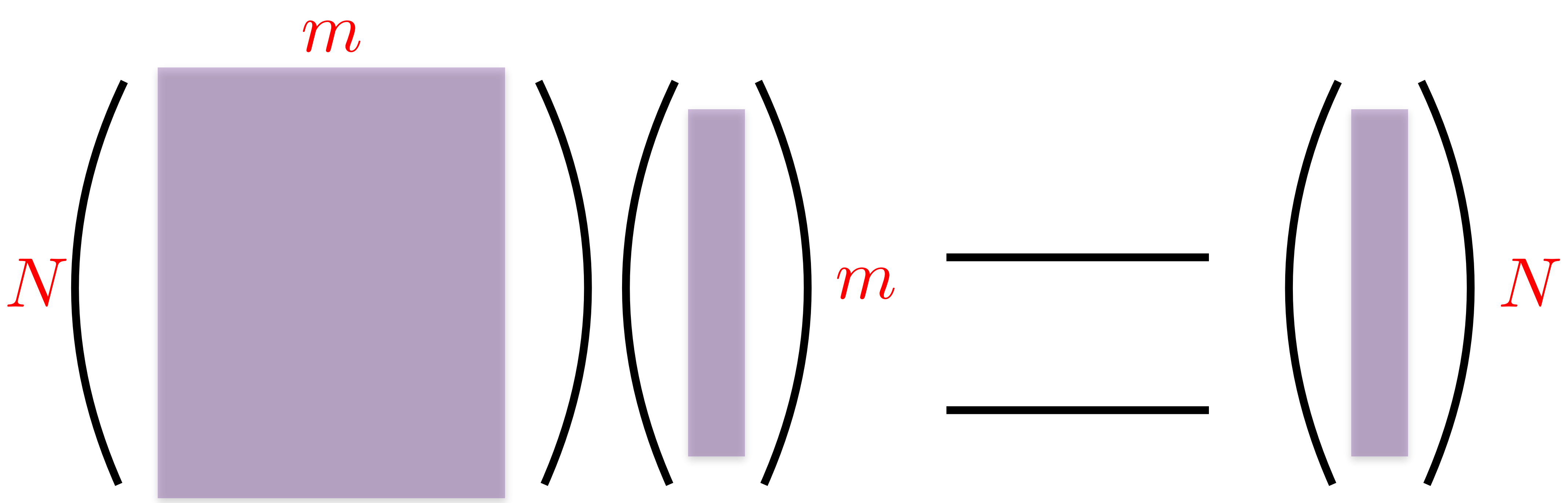
$$\mathbf{K}\alpha = p$$



# Linear-ridge vs kernel-ridge regression

## Linear-ridge regression

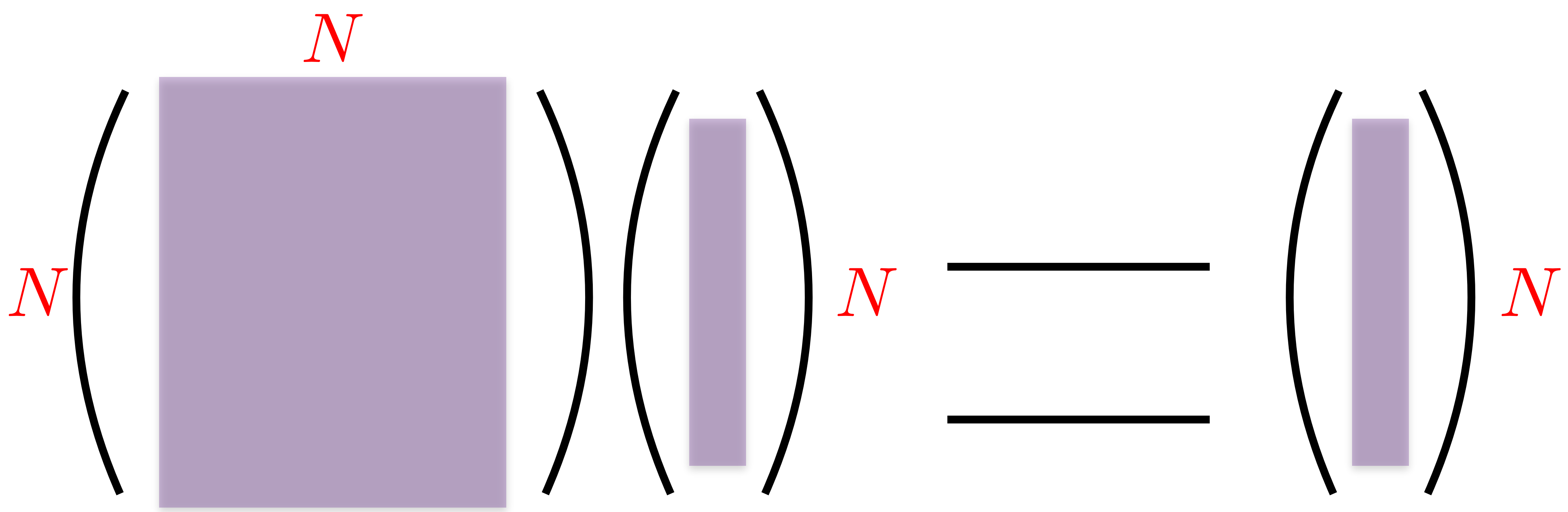
$$\mathbf{A}x = b$$




in general: $m \ll N$

## kernel-ridge regression (ML)

$$\mathbf{K}\alpha = p$$

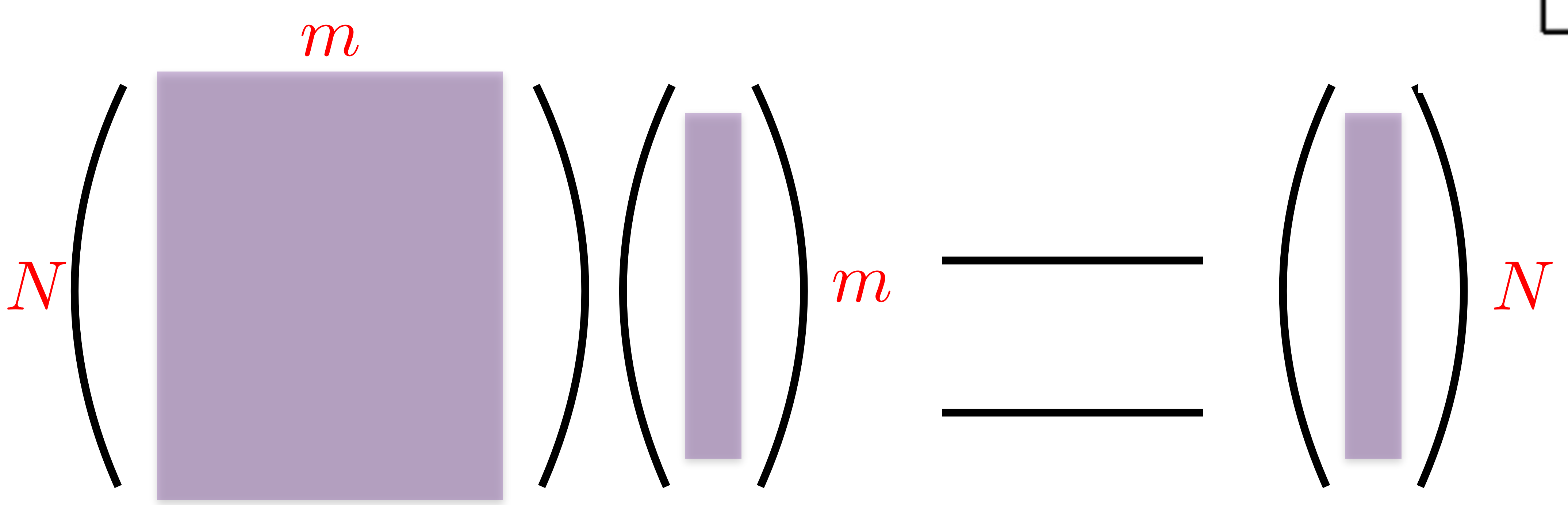



$$\begin{aligned} K_{ij} &= K_{ij}(\mathbf{x}_i, \mathbf{x}_j) \\ &= K_{ij}(|\mathbf{x}_i - \mathbf{x}_j|) \\ &= \exp\left(-\frac{|\mathbf{x}_i - \mathbf{x}_j|}{\sigma}\right) \end{aligned}$$

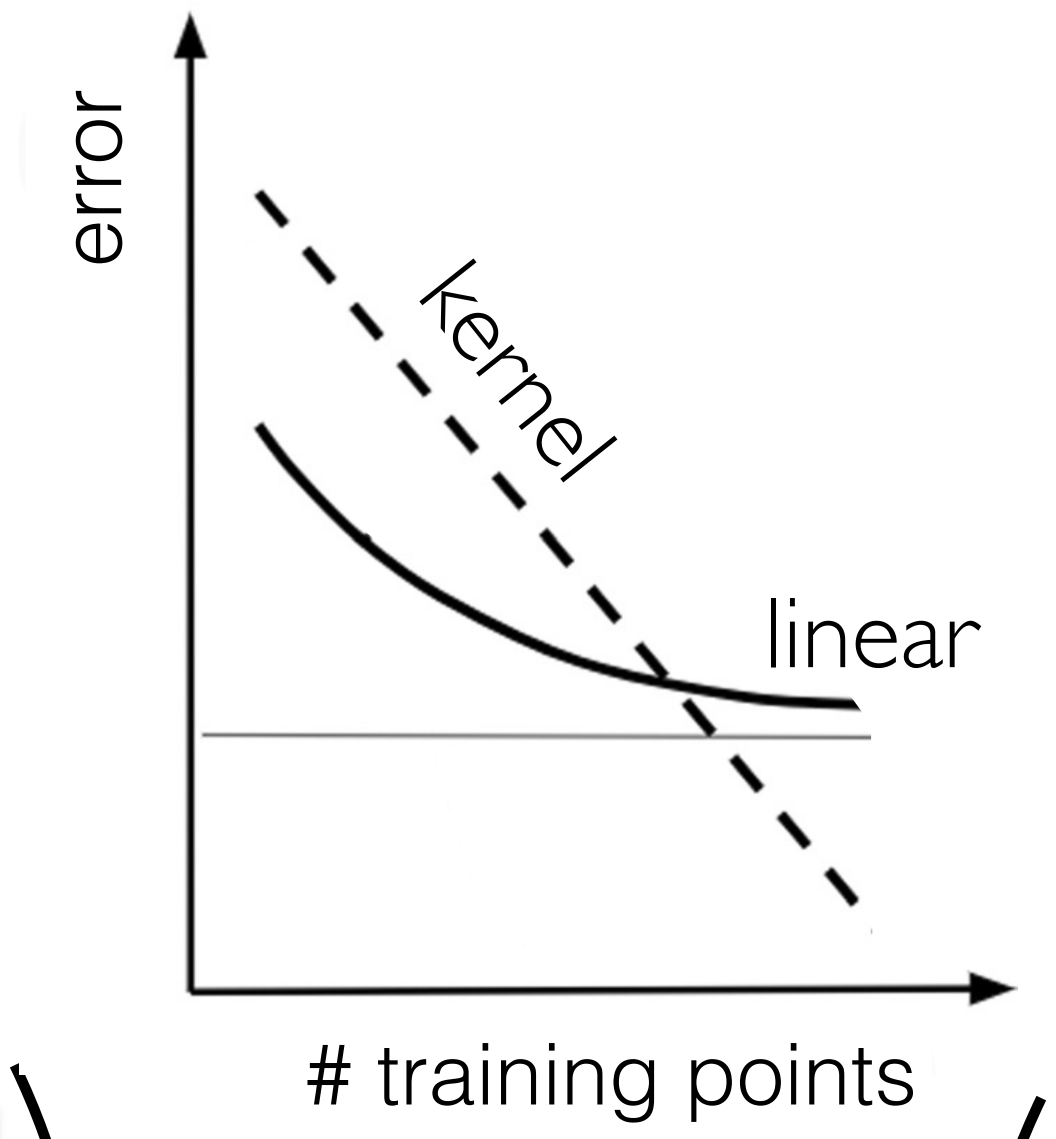
# Linear-ridge vs kernel-ridge regression

Linear-ridge regression

$$\mathbf{A}x = b$$

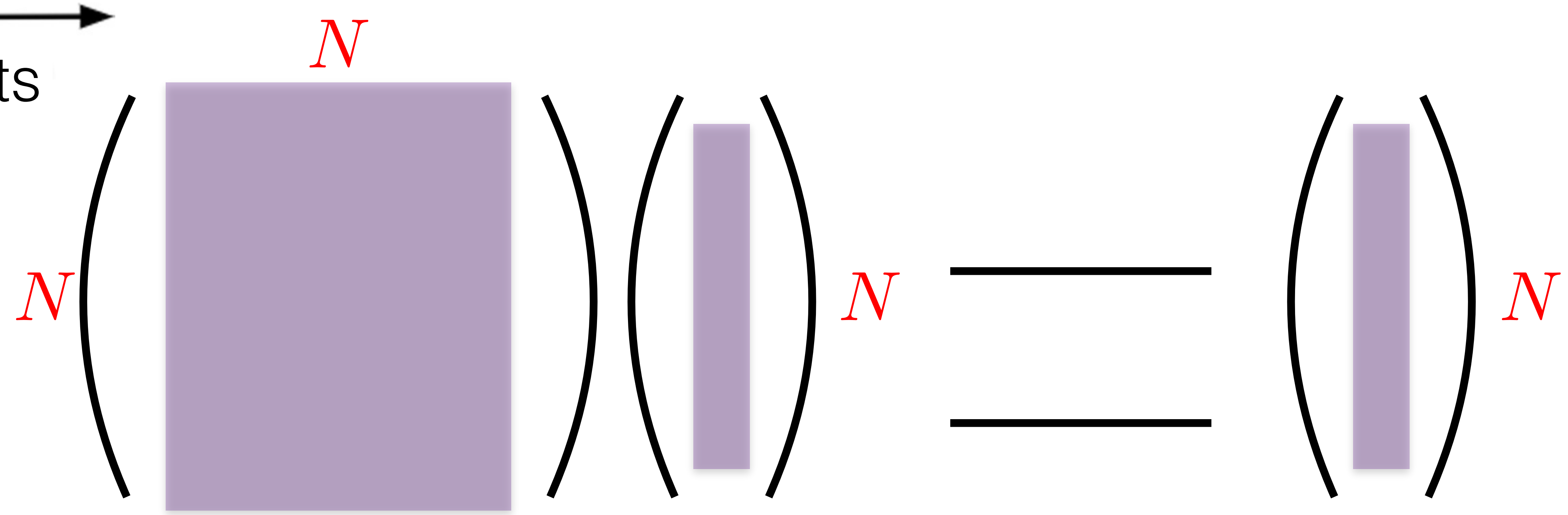



in general: $m \ll N$




kernel-ridge regression (ML)

$$\mathbf{K}\alpha = p$$

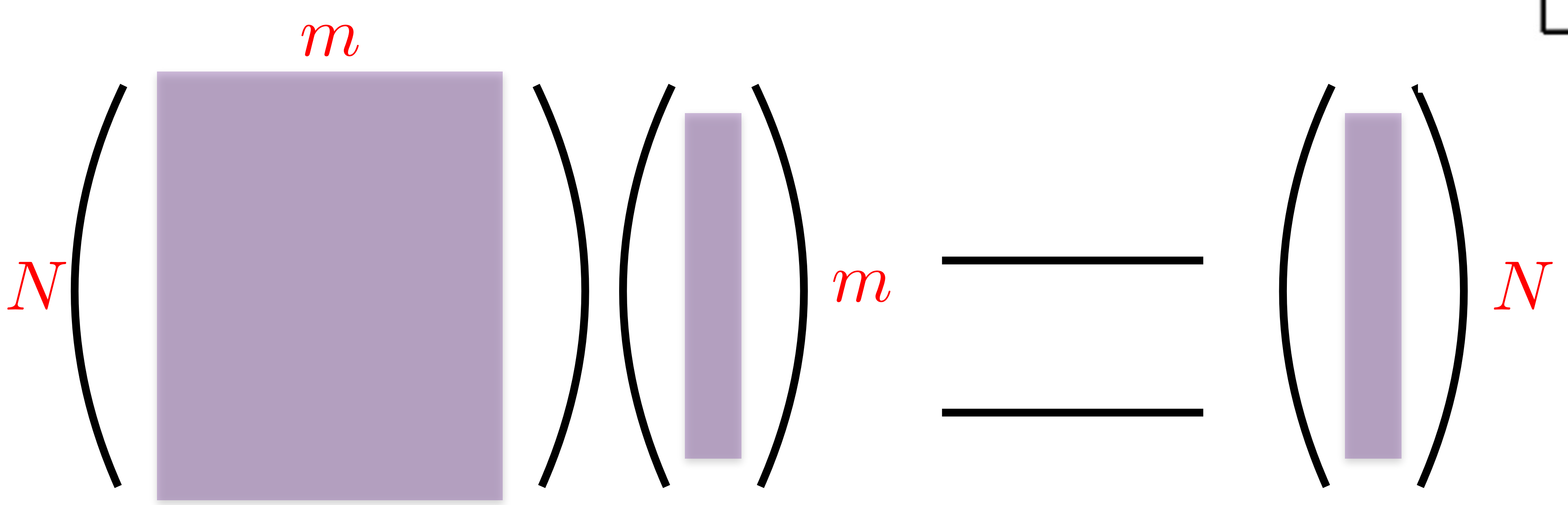



$$\begin{aligned} K_{ij} &= K_{ij}(\mathbf{x}_i, \mathbf{x}_j) \\ &= K_{ij}(|\mathbf{x}_i - \mathbf{x}_j|) \\ &= \exp\left(-\frac{|\mathbf{x}_i - \mathbf{x}_j|}{\sigma}\right) \end{aligned}$$

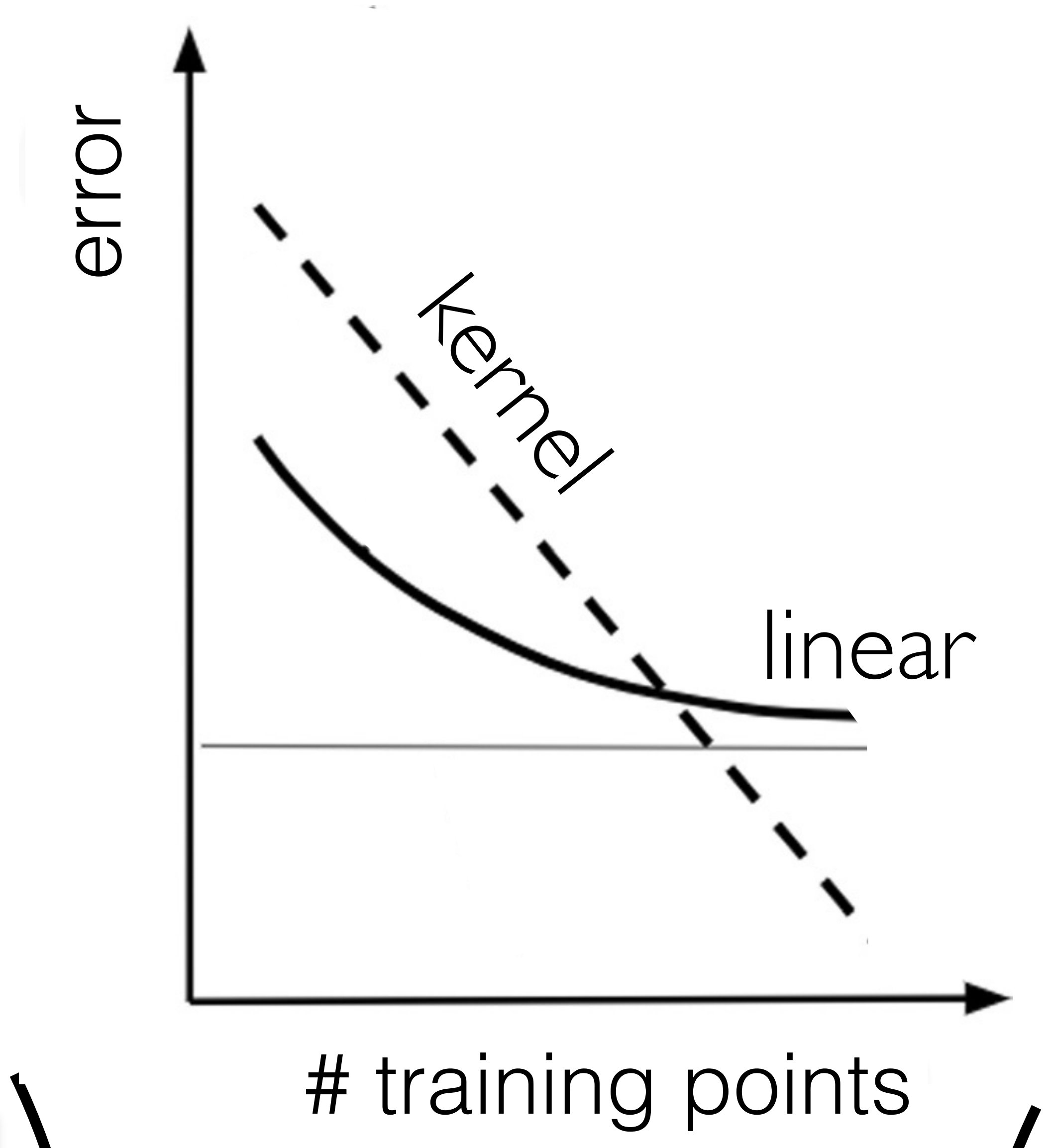
# Linear-ridge vs kernel-ridge regression

Linear-ridge regression

$$\mathbf{A}x = b$$

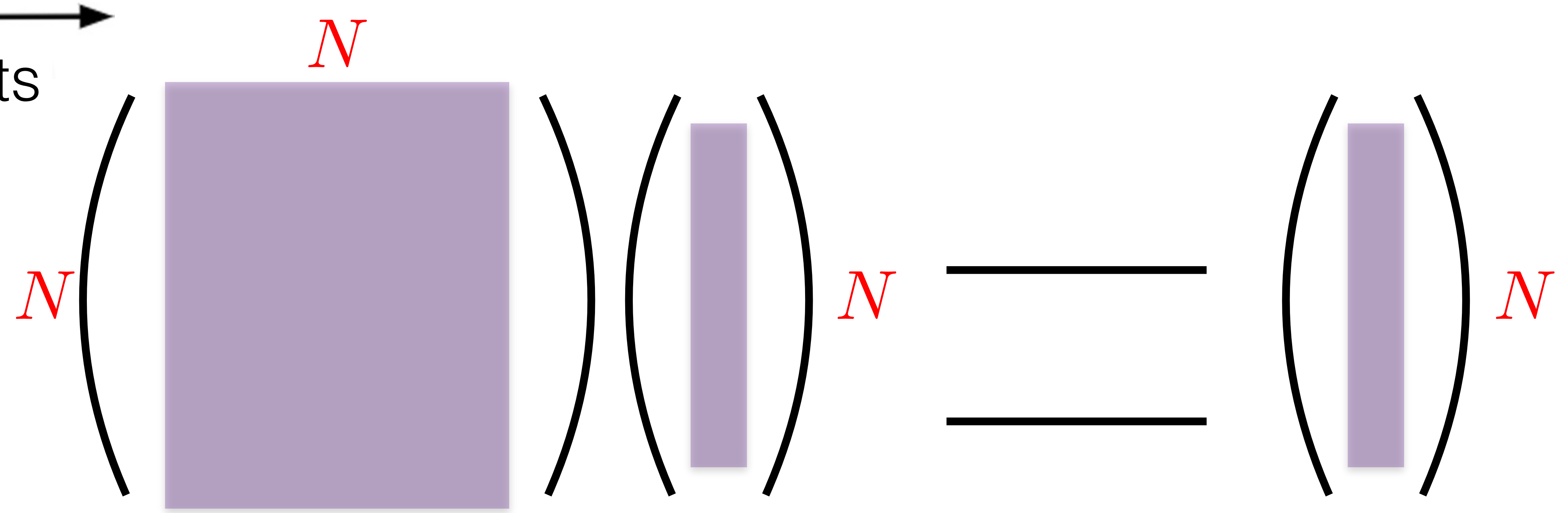



in general: $m \ll N$

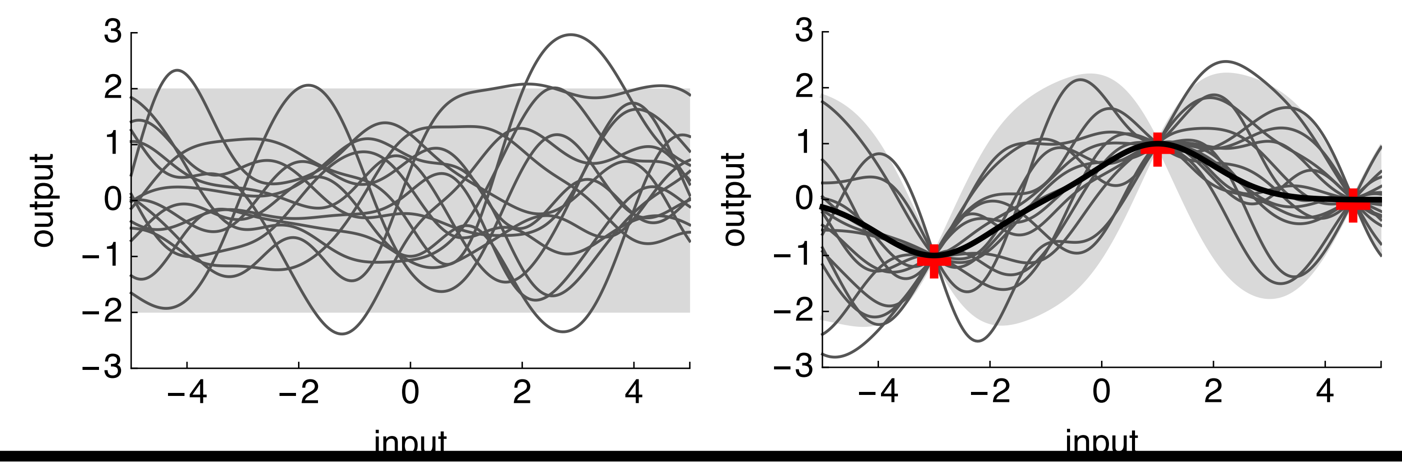



kernel-ridge regression (ML)

$$\mathbf{K}\alpha = p$$




$$\begin{aligned} K_{ij} &= K_{ij}(\mathbf{x}_i, \mathbf{x}_j) \\ &= K_{ij}(|\mathbf{x}_i - \mathbf{x}_j|) \\ &= \exp\left(-\frac{|\mathbf{x}_i - \mathbf{x}_j|}{\sigma}\right) \end{aligned}$$



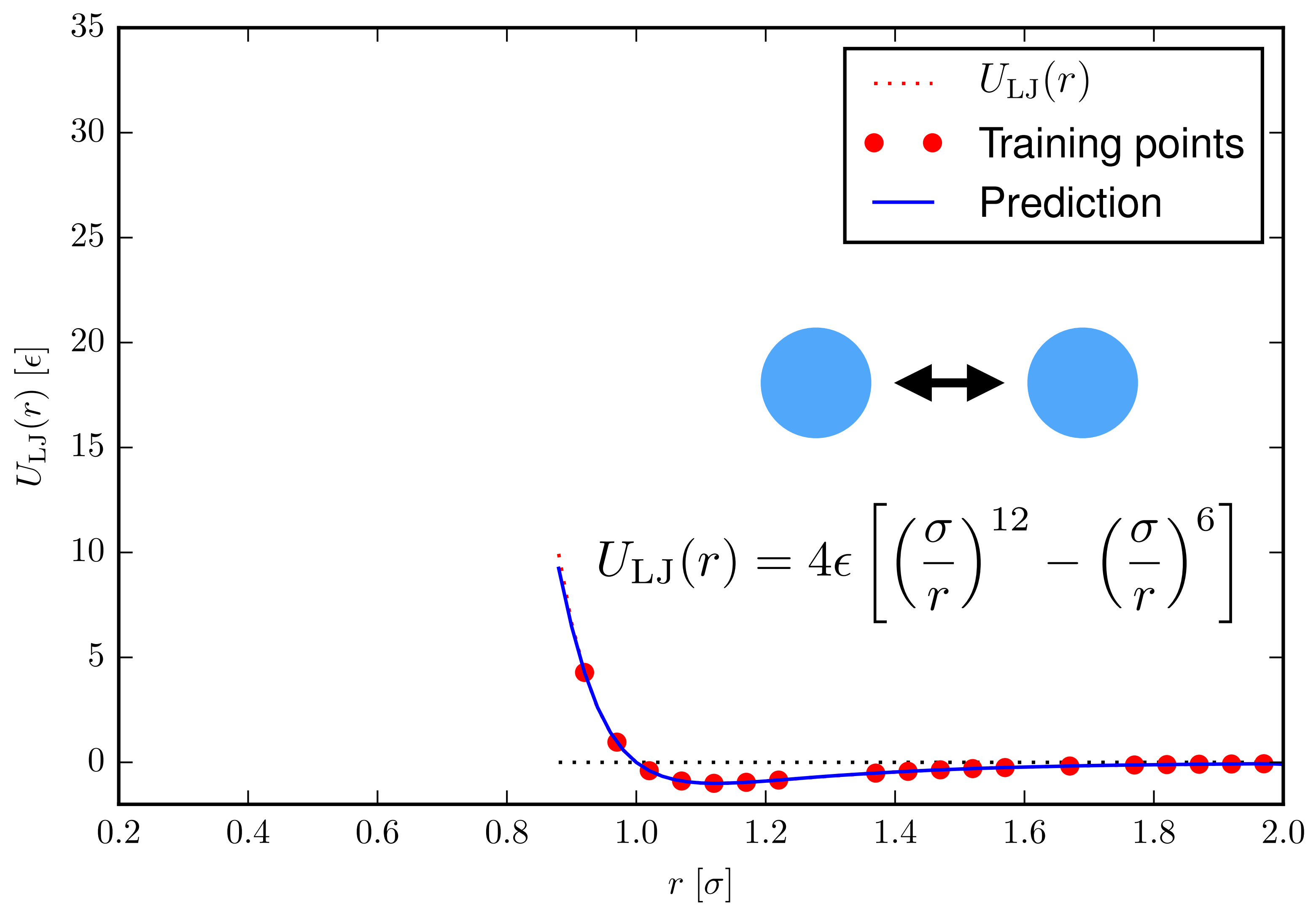

measures similarity between data points

THEORY GROUP

# Kernel machine learning 101

1) Define representation and kernel

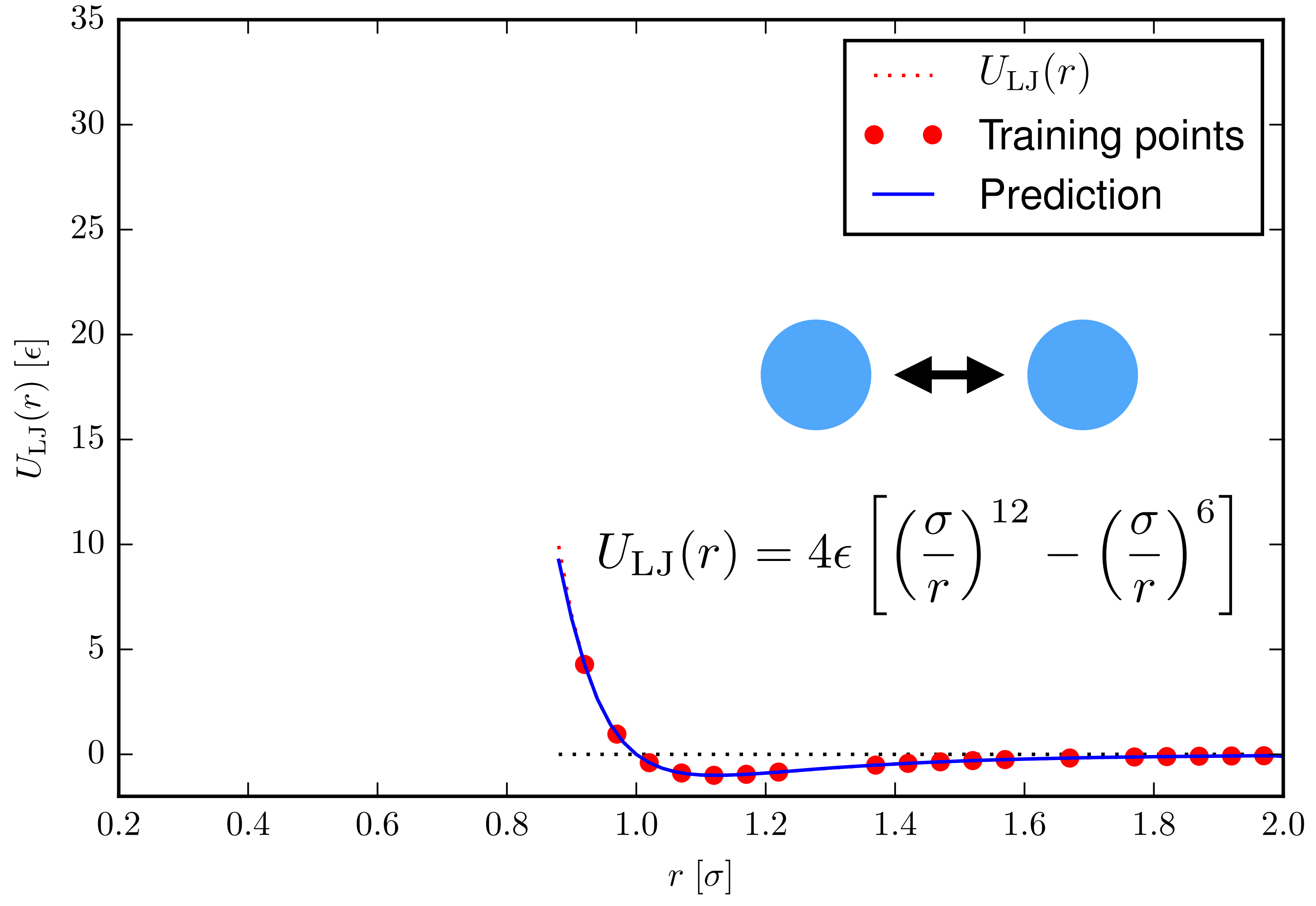
$$K(r, r') = \exp\left(-\frac{(r - r')^2}{2\sigma^2}\right)$$



# Kernel machine learning 101

1) Define representation and kernel

$$K(r, r') = \exp\left(-\frac{(r - r')^2}{2\sigma^2}\right)$$




2) Train your model:

$$\cancel{K\alpha = U}$$

Inverse is ill defined

$$(\mathbf{K} + \lambda \mathbf{I})\,\alpha = U$$

Regularization:
"hyperparameter"
scales noise level

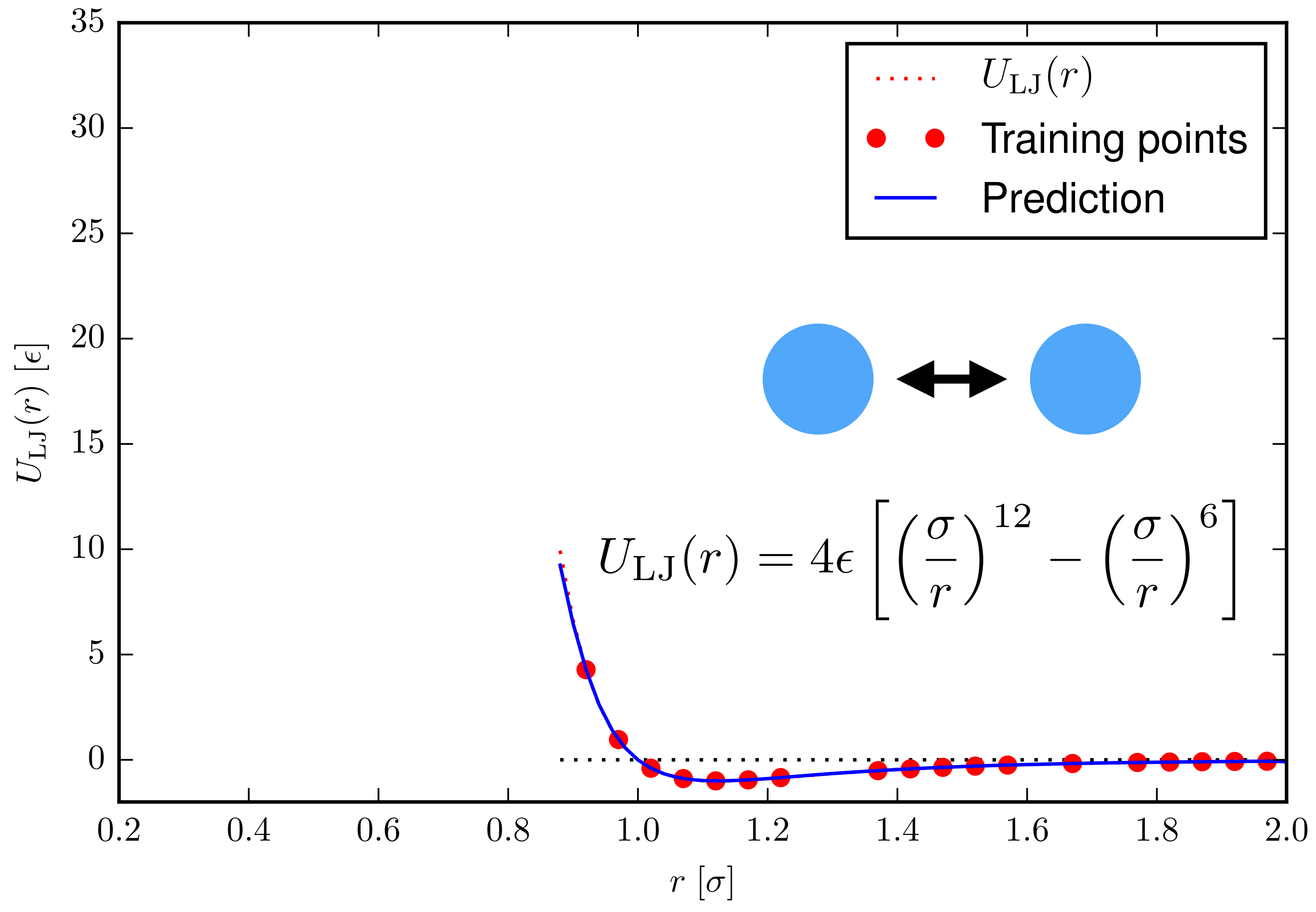
$$\alpha = (K + \lambda \mathbb{I})^{-1} U$$

Optimize weight coefficients on training set

# Kernel machine learning 101

1) Define representation and kernel

$$K(r, r') = \exp\left(-\frac{(r-r')^2}{2\sigma^2}\right)$$




2) Train your model:

$$\cancel{K\alpha = U}$$

Inverse is ill defined

$$(\mathbf{K} + \lambda \mathbf{I})\,\alpha = U$$

Regularization: "hyperparameter" scales noise level

$$\alpha = (K + \lambda \mathbb{I})^{-1} U$$

Optimize weight coefficients on training set

3) Make a prediction:

$$U(r) = \sum_i \alpha_i K(r_i^*, r)$$

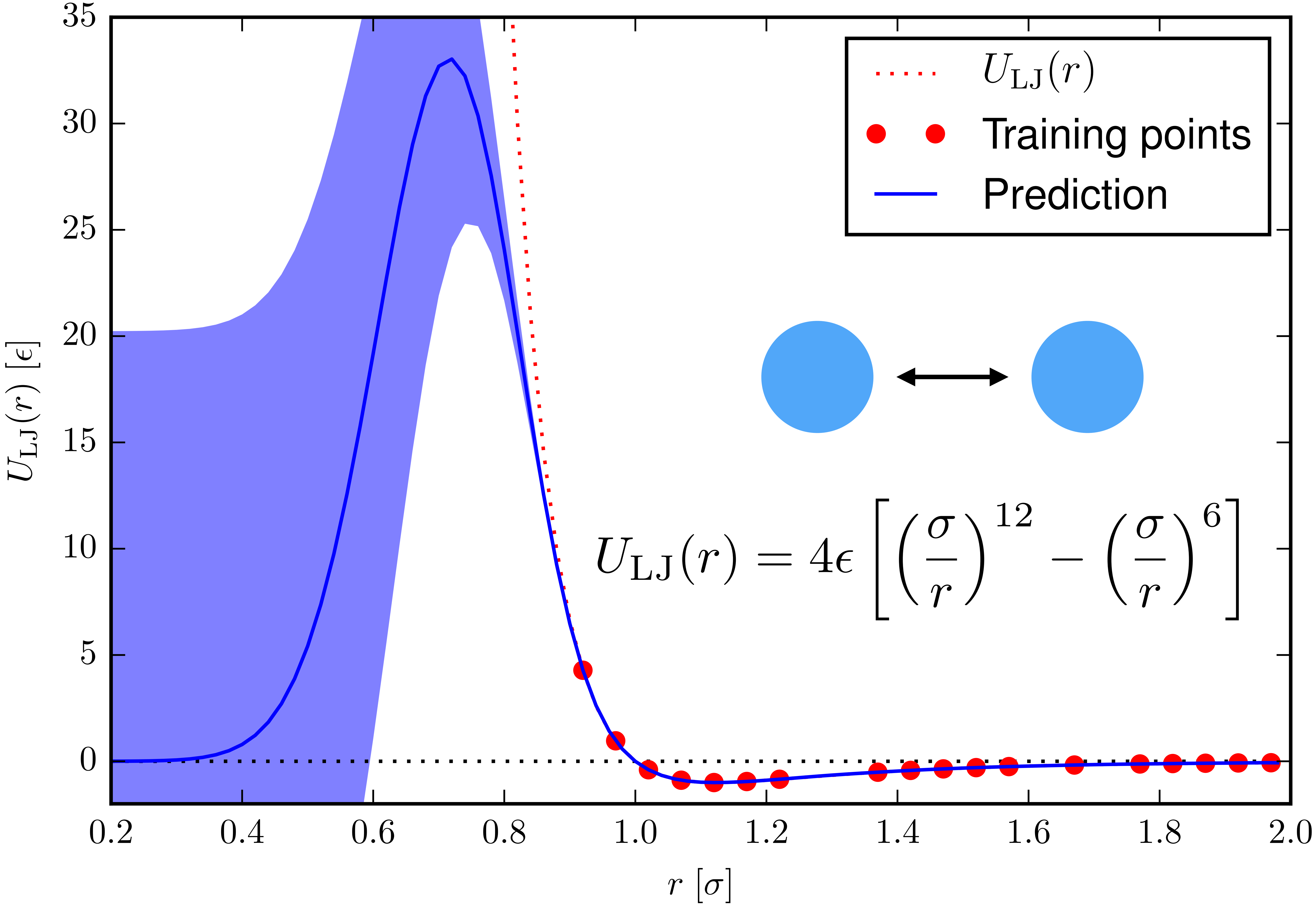
Predicted energy

Training points

Query sample

THEORY GROUP

# Extrapolation in machine learning




Define kernel

$$K(r, r') = \exp\left(-\frac{(r - r')^2}{2\sigma^2}\right)$$

Make a prediction:

$$U(r) = \sum_i \alpha_i K(r_i^*, r)$$

Predicted energy

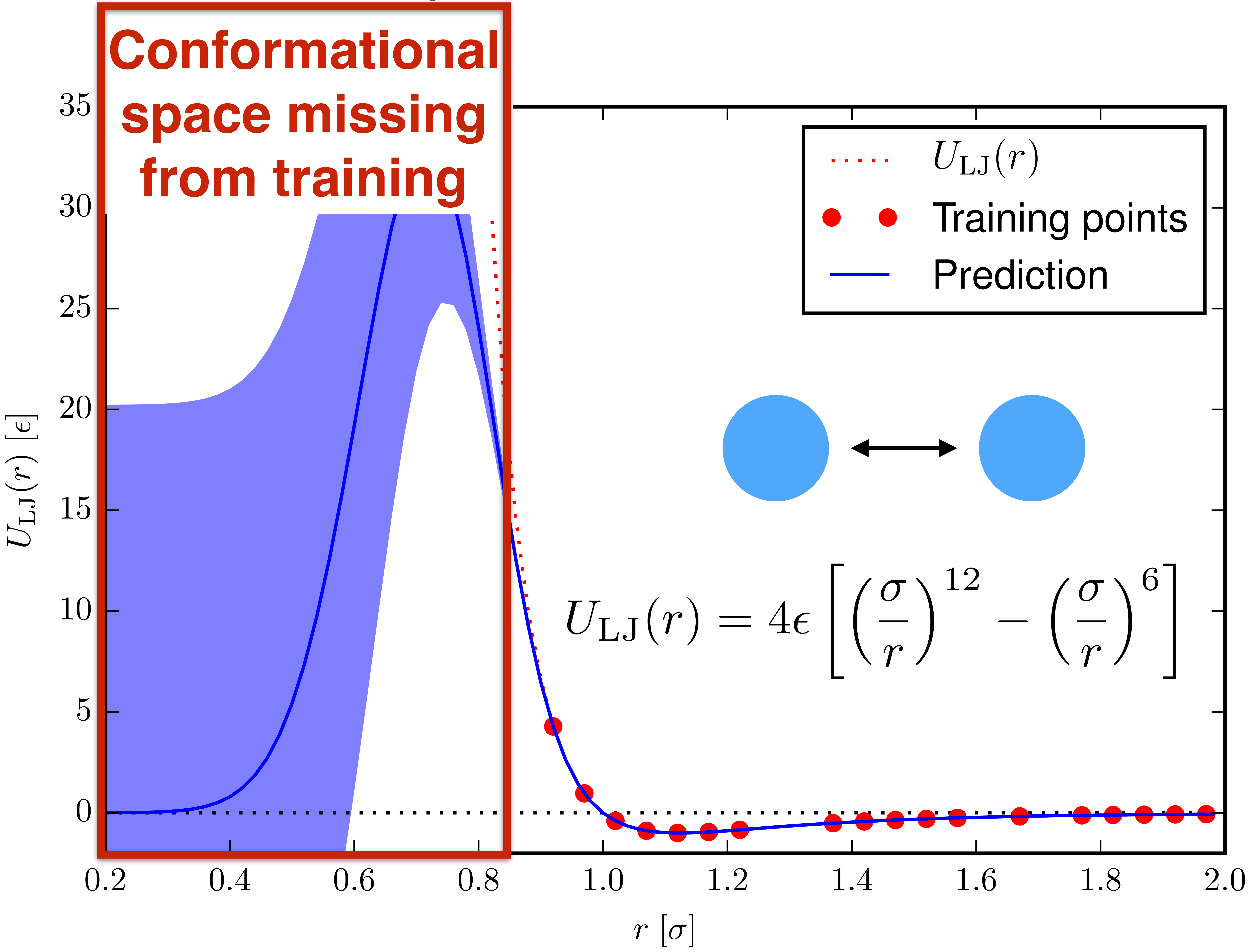
Training points

Query sample

Train your model:

$$\alpha = (K + \lambda \mathbb{I})^{-1} U$$

# Extrapolation in machine learning




Define kernel

$$K(r, r') = \exp\left(-\frac{(r-r')^2}{2\sigma^2}\right)$$

Make a prediction:

$$U(r) = \sum_i \alpha_i K(r_i^*, r)$$

Predicted energy

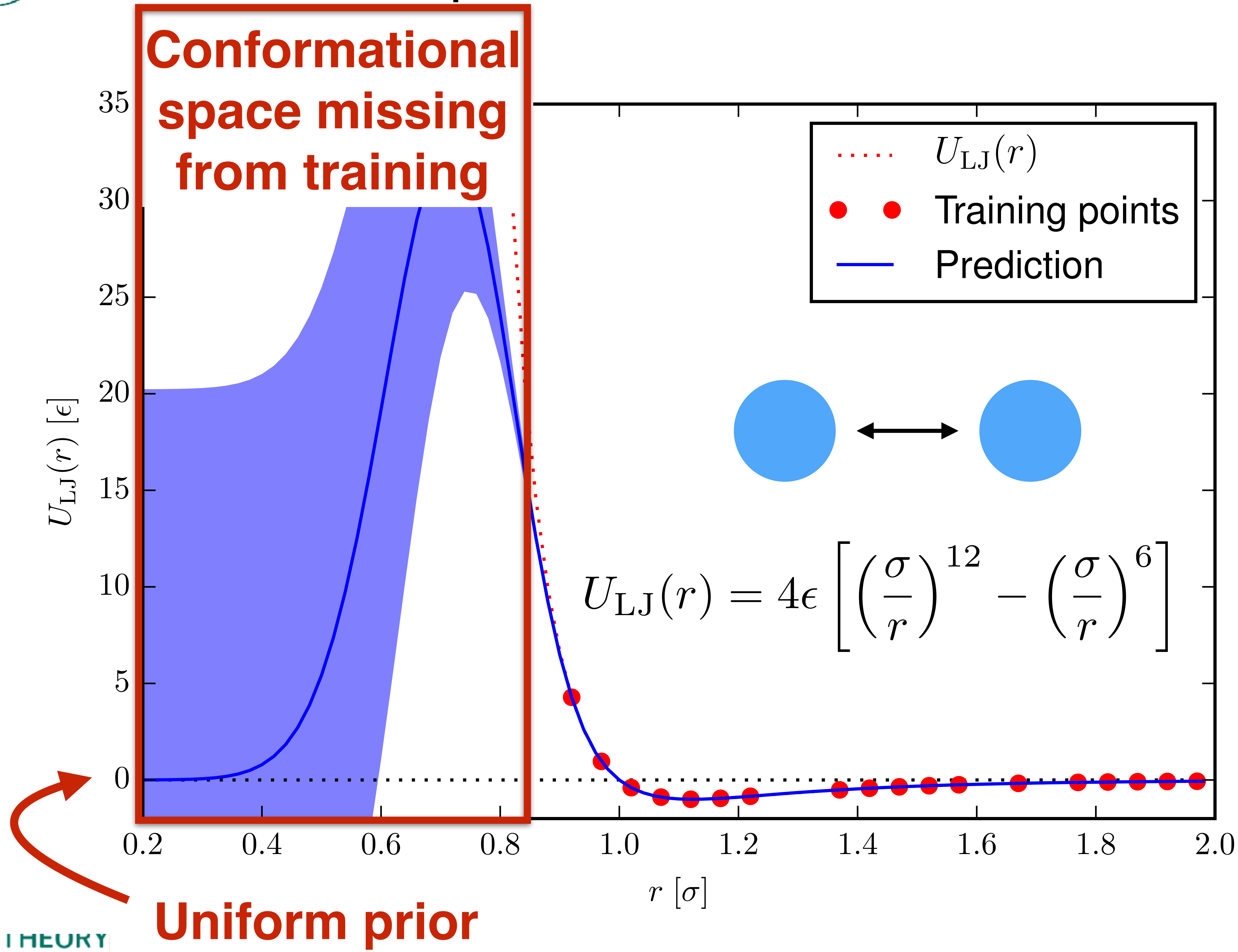
Training points

Query sample

Train your model:

$$\alpha = (K + \lambda\mathbb{I})^{-1} U$$

# Extrapolation in machine learning




Define kernel

$$K(r, r') = \exp\left(-\frac{(r-r')^2}{2\sigma^2}\right)$$

Make a prediction:

$$U(r) = \sum_i \alpha_i K(r_i^*, r)$$

Predicted energy

Training points

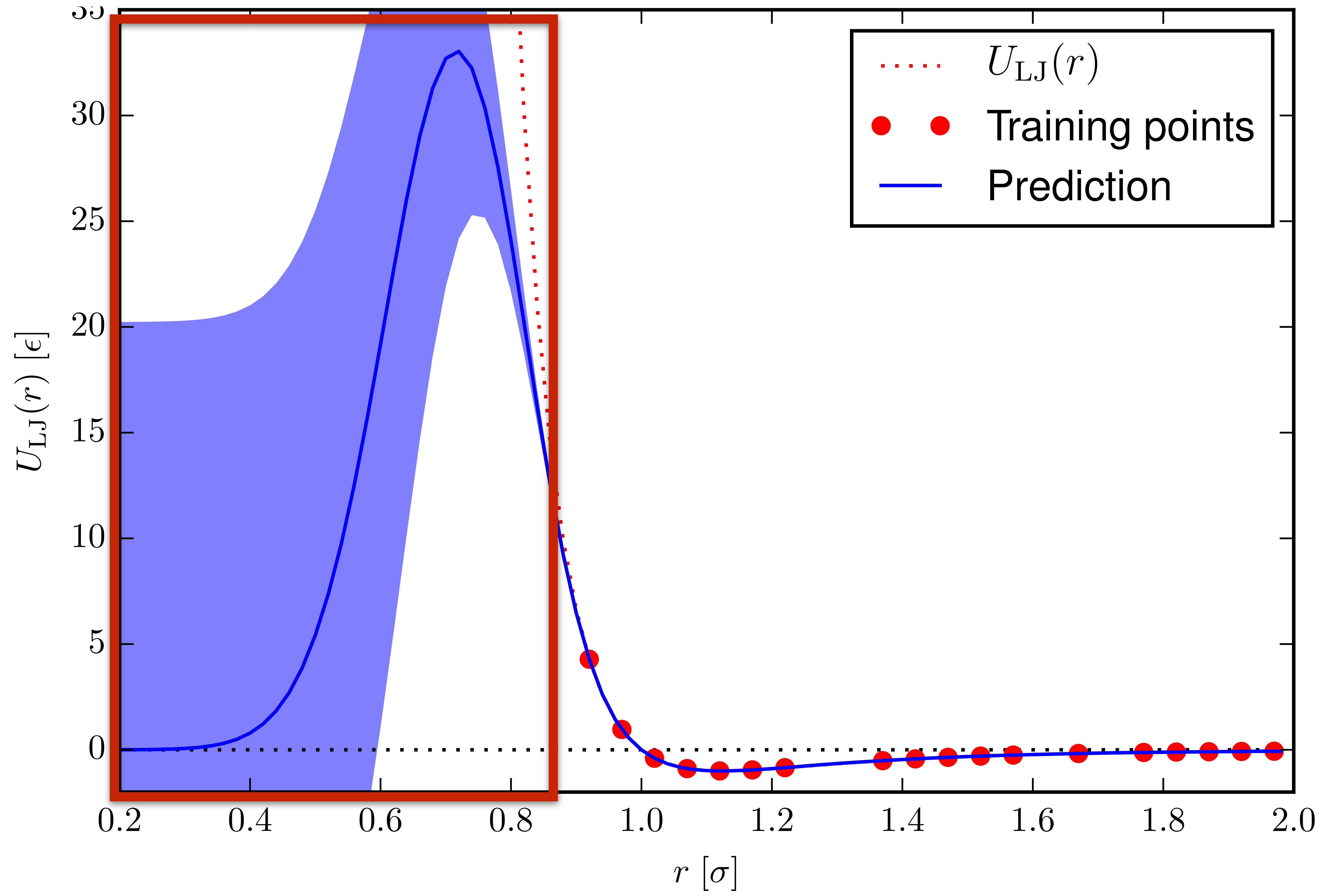
Query sample

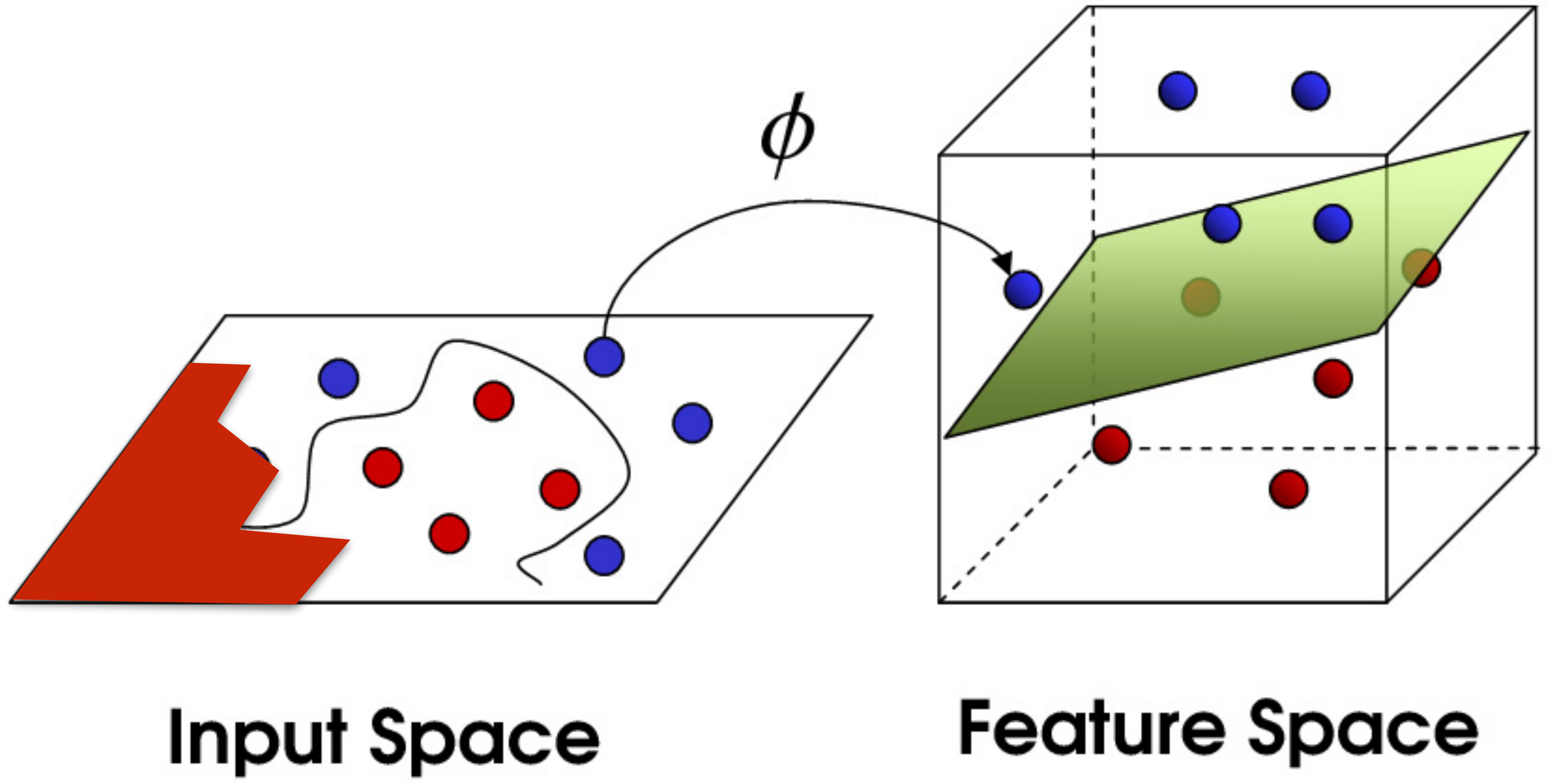
Train your model:

$$\alpha = (K + \lambda\mathbb{I})^{-1} U$$

# Linking conformational and interpolation spaces

**Conformational space missing from training**




ML training set size is limited (kernels!)

Use physics to reduce the interpolation space

# Symmetries and conservation laws

# Mechanics 101: Principle of least action

$$\mathcal{S}[x(t)] = \int_{t_1}^{t_2} \mathrm{d}t \, L[x(t), \dot{x}(t), t]$$

action

microtrajectory

kinetic energy

potential energy

Lagrangian $L = T - V$

# Mechanics 101: Principle of least action

$$\mathcal{S}[x(t)] = \int_{t_1}^{t_2} \mathrm{d}t\, L[x(t), \dot{x}(t), t]$$

action

microtrajectory

kinetic energy

potential energy

Lagrangian $L = T - V$

Hamilton's principle: system minimizes action (variational principle)

$$\dot{\mathcal{S}}[x^*(t)] = 0$$

# Mechanics 101: Principle of least action

$$\mathcal{S}[x(t)] = \int_{t_1}^{t_2} \mathrm{d}t\, L[x(t), \dot{x}(t), t]$$

action

microtrajectory

kinetic energy

potential energy

Lagrangian $L = T - V$

Hamilton's principle: system minimizes action (variational principle)

$$\dot{\mathcal{S}}[x^*(t)] = 0$$

stationarity under small perturbations leads to Euler-Lagrange equations

$$\delta\mathcal{S} = \int_{t_1}^{t_2} \mathrm{d}t\, L(x^* + \varepsilon, \dot{x}^* + \dot{\varepsilon}, t) - L(x^*, \dot{x}^*, t)$$

$$= \int_{t_1}^{t_2} \mathrm{d}t \left( \varepsilon \frac{\partial L}{\partial x} + \dot{\varepsilon} \frac{\partial L}{\partial x} \right) = \int_{t_1}^{t_2} \mathrm{d}t \left( \varepsilon \frac{\partial L}{\partial x} - \varepsilon \frac{\mathrm{d}}{\mathrm{d}t} \frac{\partial L}{\partial \dot{x}} \right) = 0$$

# Mechanics 101: Principle of least action

$$\mathcal{S}[x(t)] = \int_{t_1}^{t_2} \mathrm{d}t\, L[x(t), \dot{x}(t), t]$$

action

microtrajectory

kinetic energy

potential energy

Lagrangian $L = T - V$

Hamilton's principle: system minimizes action (variational principle)

$$\dot{\mathcal{S}}[x^*(t)] = 0$$

stationarity under small perturbations leads to Euler-Lagrange equations

$$\delta\mathcal{S} = \int_{t_1}^{t_2} \mathrm{d}t\, L(x^* + \varepsilon, \dot{x}^* + \dot{\varepsilon}, t) - L(x^*, \dot{x}^*, t)$$

$$= \int_{t_1}^{t_2} \mathrm{d}t \left( \varepsilon \frac{\partial L}{\partial x} + \dot{\varepsilon} \frac{\partial L}{\partial x} \right) = \int_{t_1}^{t_2} \mathrm{d}t\, \boxed{\left( \varepsilon \frac{\partial L}{\partial x} - \varepsilon \frac{\mathrm{d}}{\mathrm{d}t} \frac{\partial L}{\partial \dot{x}} \right) = 0}$$

integration by parts &
$\varepsilon(t_1) = \varepsilon(t_2) = 0$

THEORY GROUP

# From symmetries, to invariants, to conserved quantities

$$\mathcal{S}[x(t), y(t), z(t)] = \int \mathrm{d}t \, \frac{m}{2} \left(\dot{x}^2 + \dot{y}^2 + \dot{z}^2\right) - mgz$$

THEORY GROUP

# From symmetries, to invariants, to conserved quantities

$$\mathcal{S}[x(t), y(t), z(t)] = \int \mathrm{d}t \, \frac{m}{2}\left(\dot{x}^2 + \dot{y}^2 + \dot{z}^2\right) - mgz$$

Introduce constant translations along $x$ and $y$:

$$\begin{aligned}\mathcal{S}[x(t) + x_0, y(t) + y_0, z(t)] &= \int \mathrm{d}t \, \frac{m}{2}\left(\dot{x}^2 + \dot{y}^2 + \dot{z}^2\right) - mgz \\ &= \mathcal{S}[x(t), y(t), z(t)]\end{aligned}$$

THEORY GROUP

# From symmetries, to invariants, to conserved quantities

$$\mathcal{S}[x(t), y(t), z(t)] = \int \mathrm{d}t \, \frac{m}{2}\left(\dot{x}^2 + \dot{y}^2 + \dot{z}^2\right) - mgz$$

Introduce constant translations along $x$ and $y$:

$$\begin{aligned}\mathcal{S}[x(t) + x_0, y(t) + y_0, z(t)] &= \int \mathrm{d}t \, \frac{m}{2}\left(\dot{x}^2 + \dot{y}^2 + \dot{z}^2\right) - mgz \\ &= \mathcal{S}[x(t), y(t), z(t)]\end{aligned}$$

(Translational) symmetry leaves the action invariant.
It leaves the Euler-Lagrange equation unchanged:

$$\frac{\partial L}{\partial x} - \frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial L}{\partial \dot{x}} = 0$$

THEORY GROUP

# From symmetries, to invariants, to conserved quantities

$$\mathcal{S}[x(t), y(t), z(t)] = \int \mathrm{d}t \, \frac{m}{2}\left(\dot{x}^2 + \dot{y}^2 + \dot{z}^2\right) - mgz$$

Introduce constant translations along $x$ and $y$:

$$\begin{aligned}\mathcal{S}[x(t) + x_0, y(t) + y_0, z(t)] &= \int \mathrm{d}t \, \frac{m}{2}\left(\dot{x}^2 + \dot{y}^2 + \dot{z}^2\right) - mgz \\ &= \mathcal{S}[x(t), y(t), z(t)]\end{aligned}$$

(Translational) symmetry leaves the action invariant.
It leaves the Euler-Lagrange equation unchanged:

$$\frac{\partial L}{\partial x} - \frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial L}{\partial \dot{x}} = 0$$

$$\frac{\partial L}{\partial x} = 0 \qquad \frac{\partial L}{\partial \dot{x}} = m\dot{x} = \text{const}\,.$$

Translational invariance implies linear momentum conversation

# From symmetries to conserved quantities (cont'd)

$$\mathcal{S}[\mathbf{r}(t)] = \int \mathrm{d}t \, \frac{m}{2}\dot{\mathbf{r}}^2 - V(r)$$

**Rotational symmetry**

Apply transformation $\mathbf{r} \to \mathbf{r}'$

where $\mathbf{r}'(t) = R\mathbf{r}(t) = \mathbf{r}(t) + \alpha \times \mathbf{r}(t)$

One can show that

$$\mathcal{S}[\mathbf{r}(t) + \alpha \times \mathbf{r}(t)] = \mathcal{S}[\mathbf{r}(t)]$$

Conservation of angular momentum

**Time translation**

Apply transformation $\mathbf{r} \to \mathbf{r}'$

where $\mathbf{r}'(t+\epsilon) = \mathbf{r}(t)$

One can show that

$$\mathcal{S}[\mathbf{r}'(t+\epsilon)] = \mathcal{S}[\mathbf{r}(t)]$$

(up to a boundary term)

Conservation of energy



Bañados & Reyes, *arXiv:1601.03616v3*

# Noether's theorem

**To every differentiable symmetry generated by local actions there corresponds a conserved quantity**

3 examples:

- Translational symmetry: **Linear momentum conservation**
- Rotational symmetry: **Angular momentum conservation**
- Time translation: **Energy conservation**

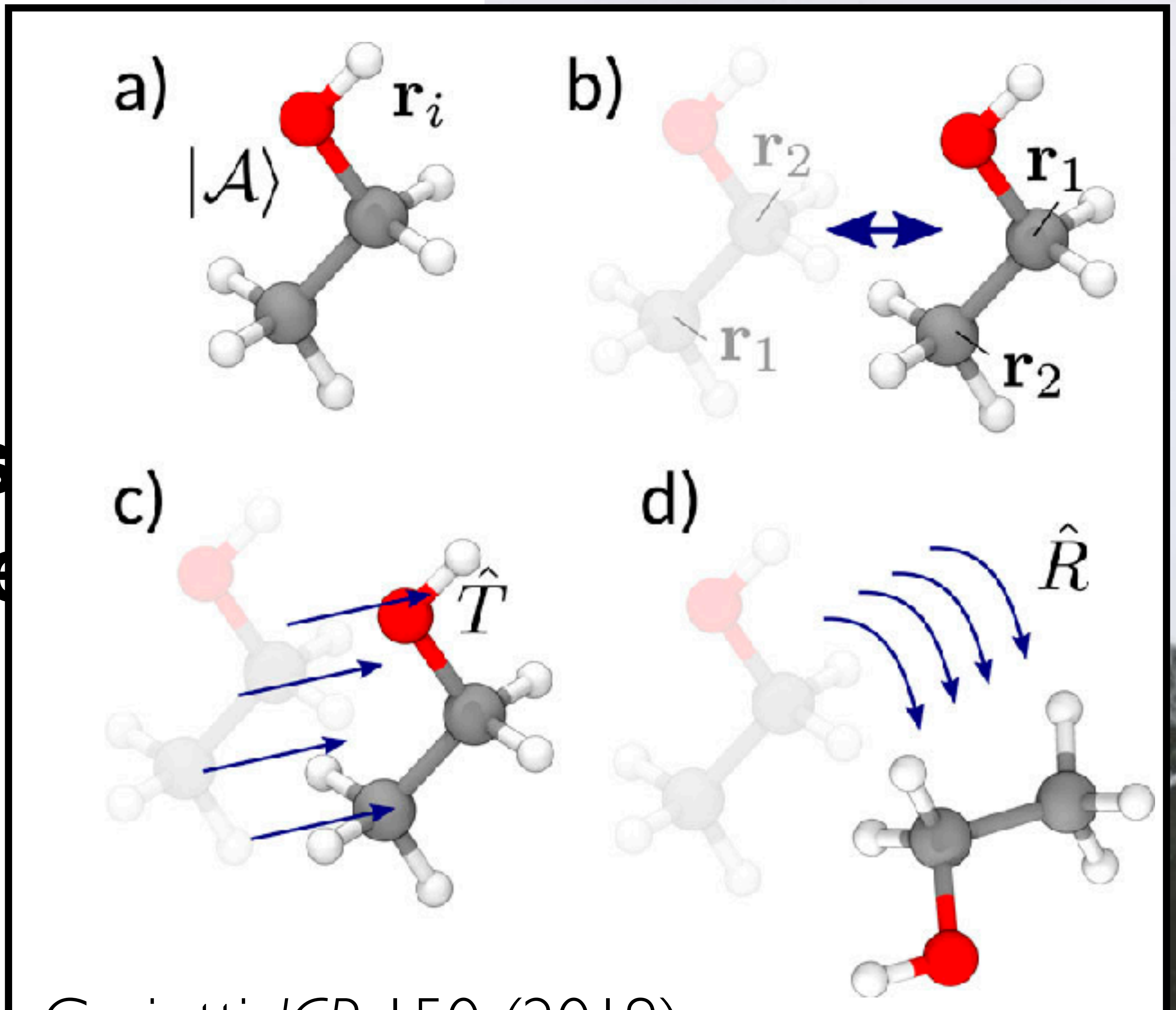
THEORY GROUP

# Noether's theorem

**To every differentiable symmetry generated by local actions there corresponds a conserved quantity**

3 examples:

- Translational symmetry: **Linear momentum cons**
- Rotational symmetry: **Angular momentum conse**
- Time translation: **Energy conservation**




Ceriotti, *JCP* 150 (2019)

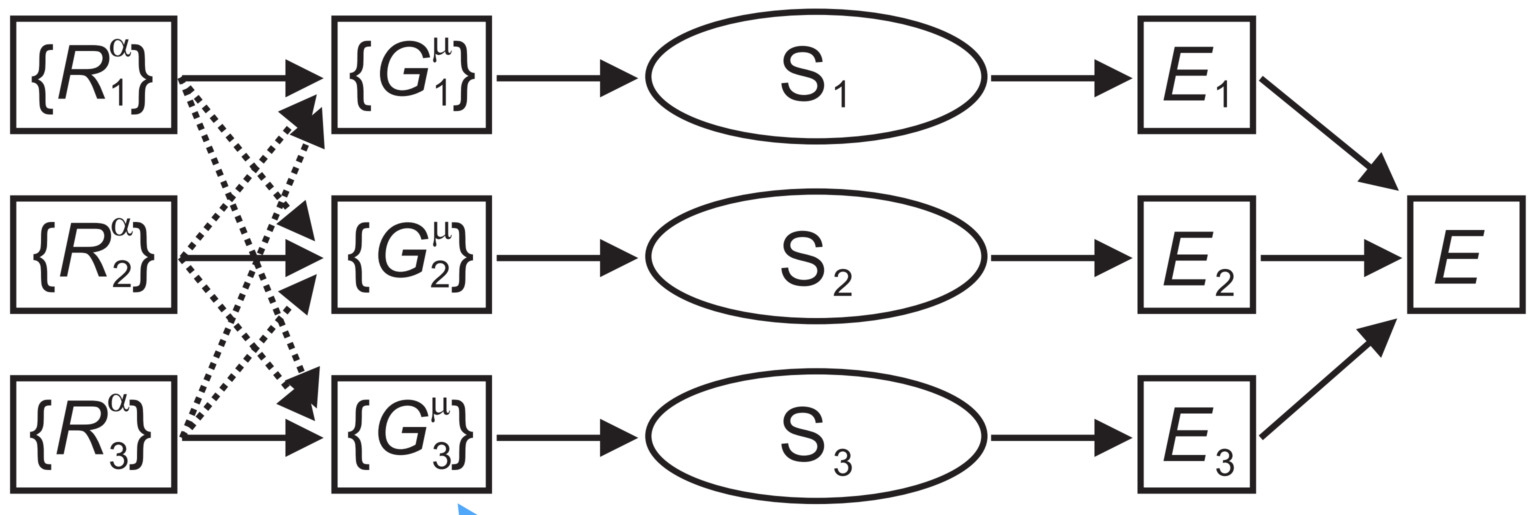
# 2 ways of encoding symmetries:

- Representation
- ML model

# Encoding symmetries in the representation

# Translational and rotational symmetries

## Behler-Parrinello




Cartesian coordinates

Symmetry functions

$$G_i^1 = \sum_{j \neq i}^{\text{all}} e^{-\eta(R_{ij}-R_s)^2} f_c(R_{ij})$$

Distances

$$G_i^2 = 2^{1-\zeta} \sum_{j,k \neq i}^{\text{all}} (1 + \lambda \cos\theta_{ijk})^\zeta \times e^{-\eta(R_{ij}^2+R_{ik}^2+R_{jk}^2)} f_c(R_{ij}) f_c(R_{ik}) f_c(R_{jk})$$

Angles

Behler & Parrinello, *Phys Rev Lett* **98** (2007)

## Coulomb matrix

$$C_{ij} = \begin{cases} \frac{1}{2} Z_i^{2.4} & \forall \;\; i = j \\ \dfrac{Z_i Z_j}{|\mathbf{R}_i - \mathbf{R}_j|} & \forall \;\; i \neq j \end{cases}$$

Distances

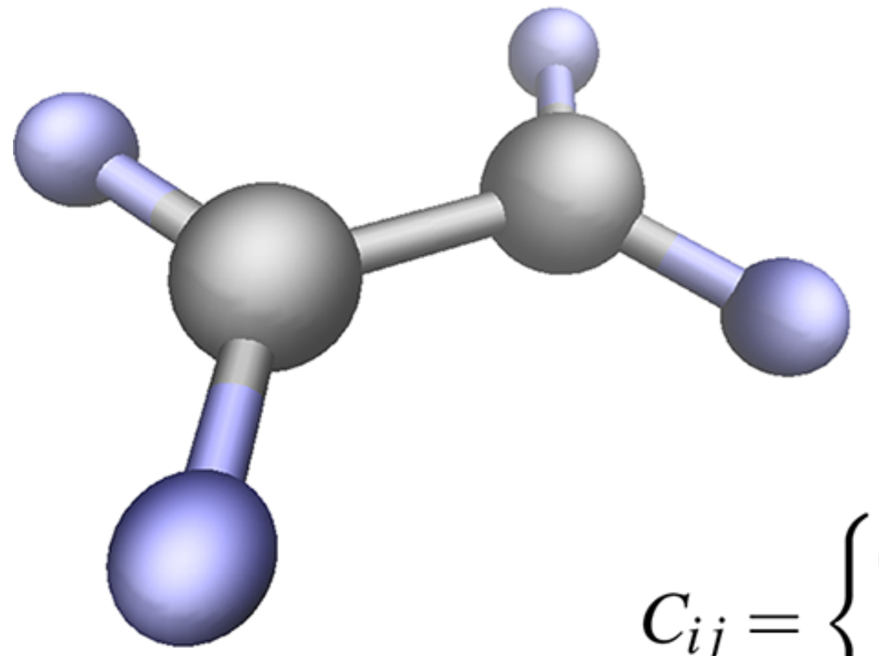
Rupp, Tkatchenko, Müller, von Lilienfeld, *Phys Rev Lett*, **108** (2012)

THEORY GROUP

# Representation: the Coulomb matrix

Symmetries of the representation should emulate symmetries of the system

1. Translation
2. Rotations
3. Mirror reflection



$\longrightarrow$

$$\mathbf{C} = \begin{array}{c|cccccc|} & \mathrm{H} & \mathrm{H} & \mathrm{C} & \mathrm{C} & \mathrm{H} & \mathrm{H} \\ \hline \mathrm{H} & 0.5 & 0.3 & 2.9 & 1.5 & 0.2 & 0.2 \\ \mathrm{H} & 0.3 & 0.5 & 2.9 & 1.5 & 0.2 & 0.2 \\ \mathrm{C} & 2.9 & 2.9 & 36.9 & 14.3 & 1.5 & 1.5 \\ \mathrm{C} & 1.5 & 1.5 & 14.3 & 36.9 & 2.9 & 2.9 \\ \mathrm{H} & 0.2 & 0.2 & 1.5 & 2.9 & 0.5 & 0.3 \\ \mathrm{H} & 0.2 & 0.2 & 1.5 & 2.9 & 0.3 & 0.5 \\ \hline \end{array}$$

$$C_{ij} = \begin{cases} 0.5\, Z_i^{2.4} & \forall i = j \\ \frac{Z_i Z_j}{|\mathbf{R}_i - \mathbf{R}_j|} & \forall i \neq j. \end{cases}$$

~ Coulomb's law $E = \dfrac{q_i q_j}{|\mathbf{r}_i - \mathbf{r}_j|}$

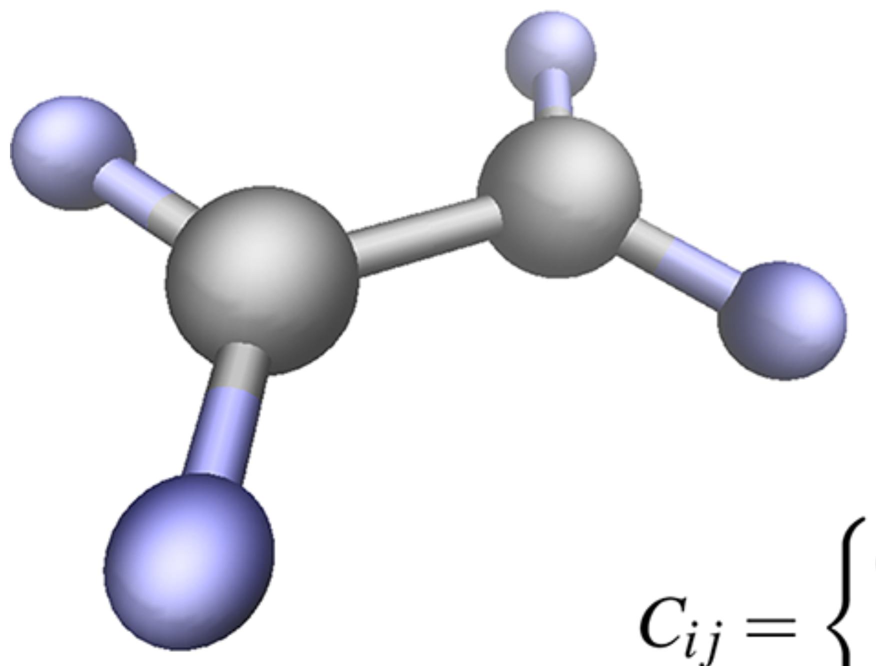
THEORY GROUP

Hansen et al., *J Chem Theory Comput*, **9** (2013)

# Representation: the Coulomb matrix

Symmetries of the representation should emulate symmetries of the system

1. Translation
2. Rotations
3. Mirror reflection



$$\longrightarrow \quad \mathbf{C} = \begin{array}{c} \\ \mathrm{H} \\ \mathrm{H} \\ \mathrm{C} \\ \mathrm{C} \\ \mathrm{H} \\ \mathrm{H} \end{array} \begin{array}{cccccc} \mathrm{H} & \mathrm{H} & \mathrm{C} & \mathrm{C} & \mathrm{H} & \mathrm{H} \\ \hline 0.5 & 0.3 & 2.9 & 1.5 & 0.2 & 0.2 \\ 0.3 & 0.5 & 2.9 & 1.5 & 0.2 & 0.2 \\ 2.9 & 2.9 & 36.9 & 14.3 & 1.5 & 1.5 \\ 1.5 & 1.5 & 14.3 & 36.9 & 2.9 & 2.9 \\ 0.2 & 0.2 & 1.5 & 2.9 & 0.5 & 0.3 \\ 0.2 & 0.2 & 1.5 & 2.9 & 0.3 & 0.5 \end{array}$$

$$C_{ij} = \begin{cases} 0.5\, Z_i^{2.4} & \forall i = j \\ \frac{Z_i Z_j}{|\mathbf{R}_i - \mathbf{R}_j|} & \forall i \neq j. \end{cases}$$

~ Coulomb's law $E = \dfrac{q_i q_j}{|\mathbf{r}_i - \mathbf{r}_j|}$

**Problems:**

1. **Dimensionality from # atoms**
2. **Ordering of the atoms**

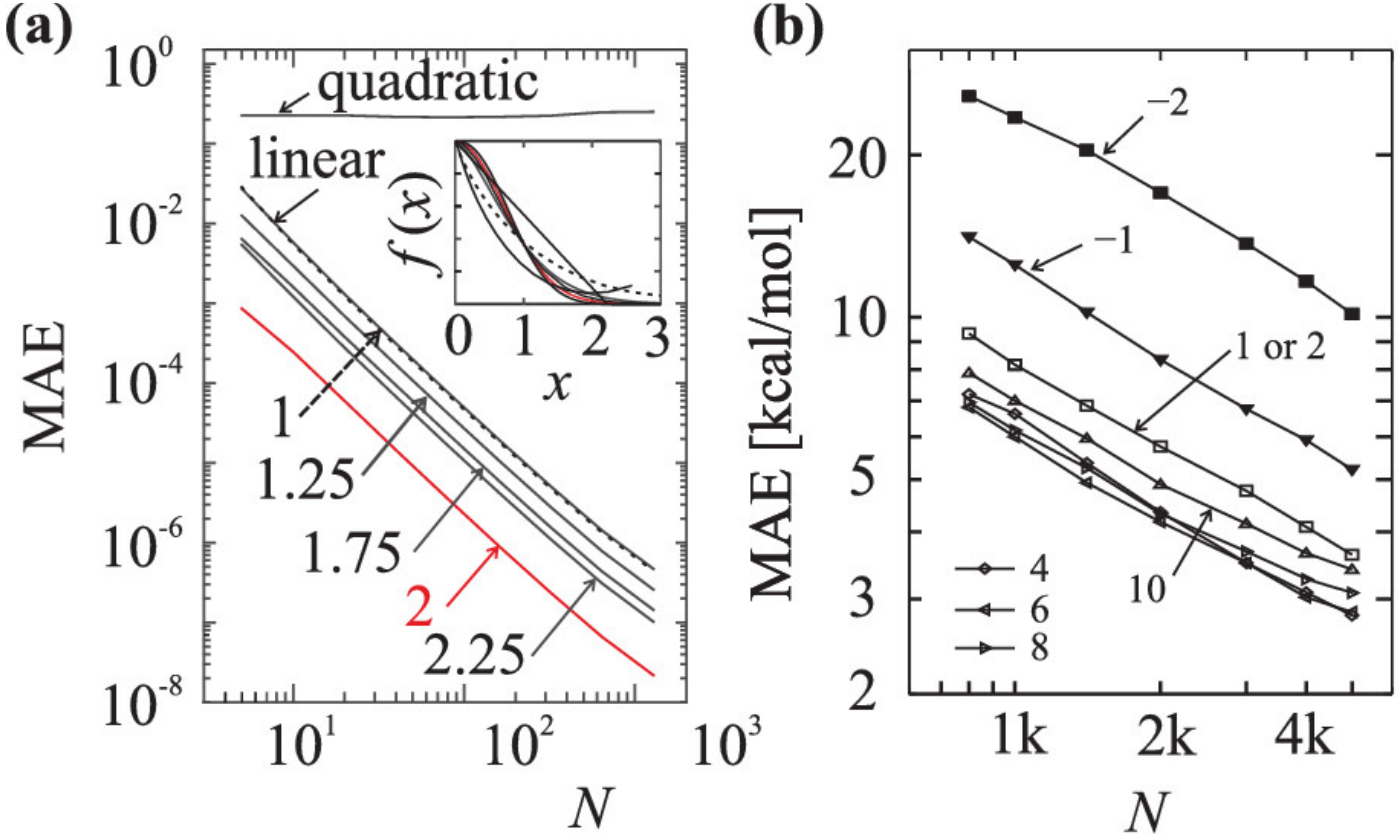


Hansen et al., *J Chem Theory Comput*, **9** (2013)

# Optimizing the representation links to the physics

Learning a Gaussian function

Learning atomization energies




Huang and von Lilienfeld, *J Chem Phys* **145** (2016)

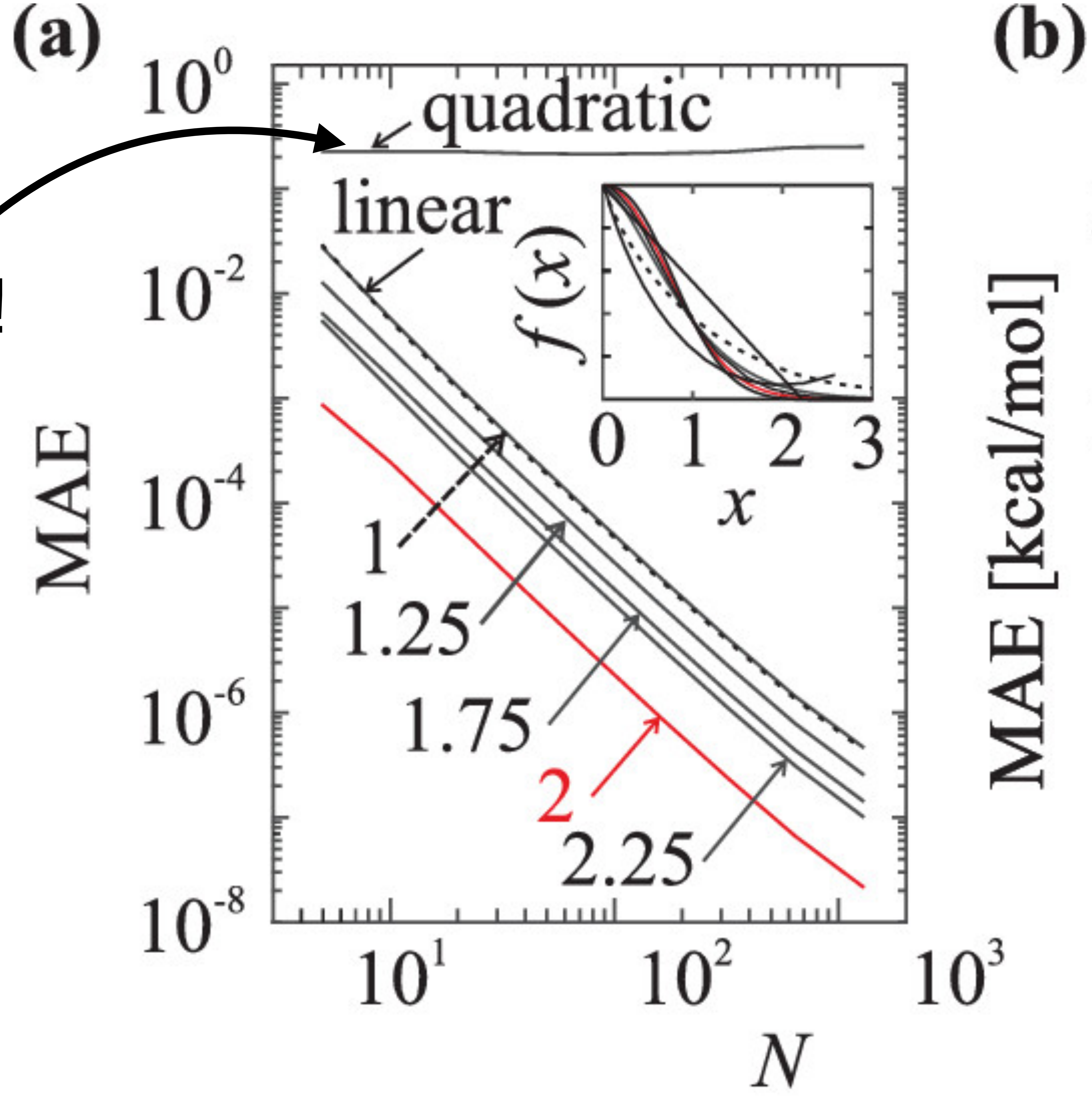
# Optimizing the representation links to the physics

Max-Planck-Institut für Polymerforschung
Max Planck Institute for Polymer Research

Learning a Gaussian function

Learning atomization energies

Non-unique representation!






Huang and von Lilienfeld, *J Chem Phys* **145** (2016)

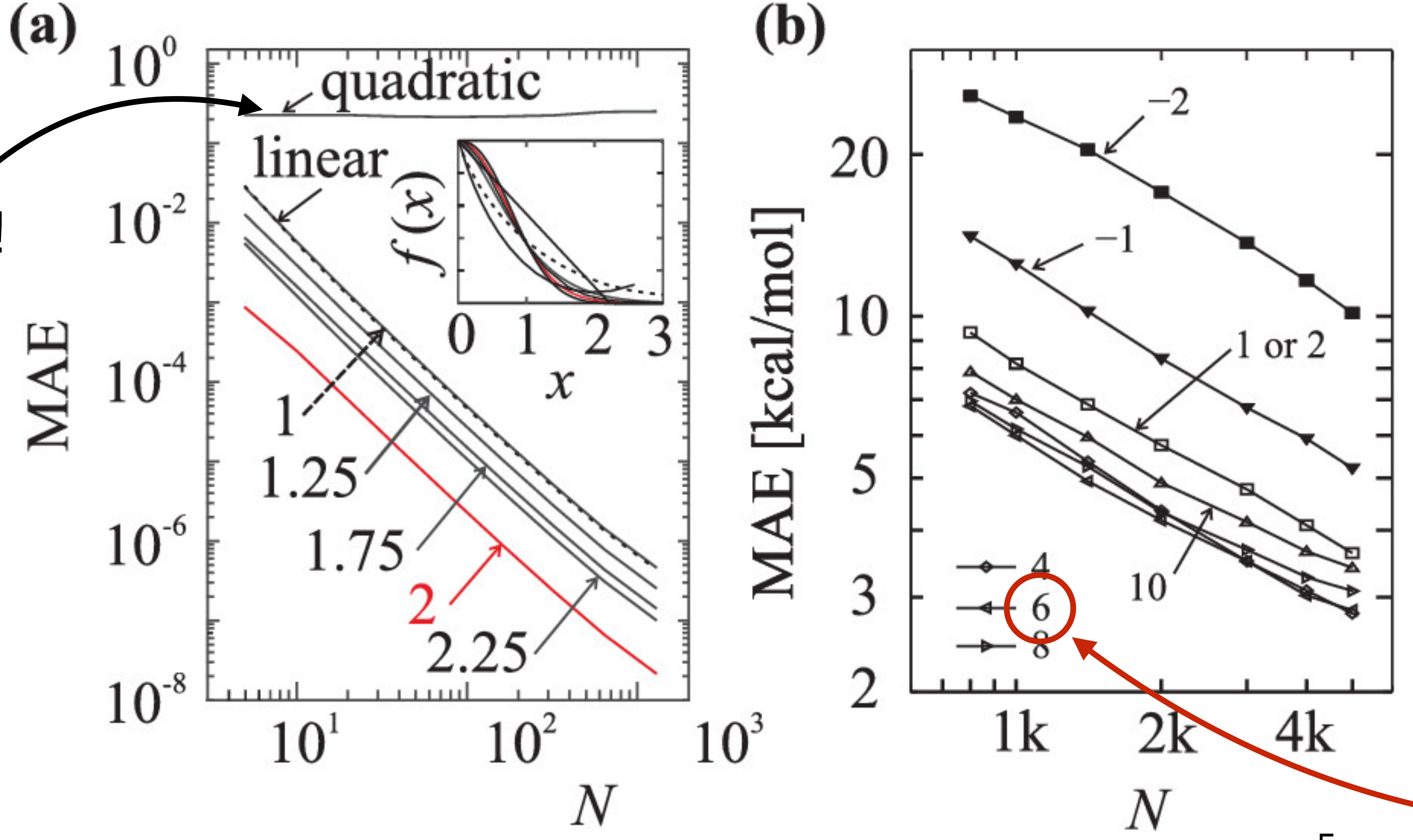
# Optimizing the representation links to the physics



Learning a Gaussian function

Learning atomization energies

Non-unique representation!




$$U_{\mathrm{LJ}}(r) = 4\epsilon\left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right]$$

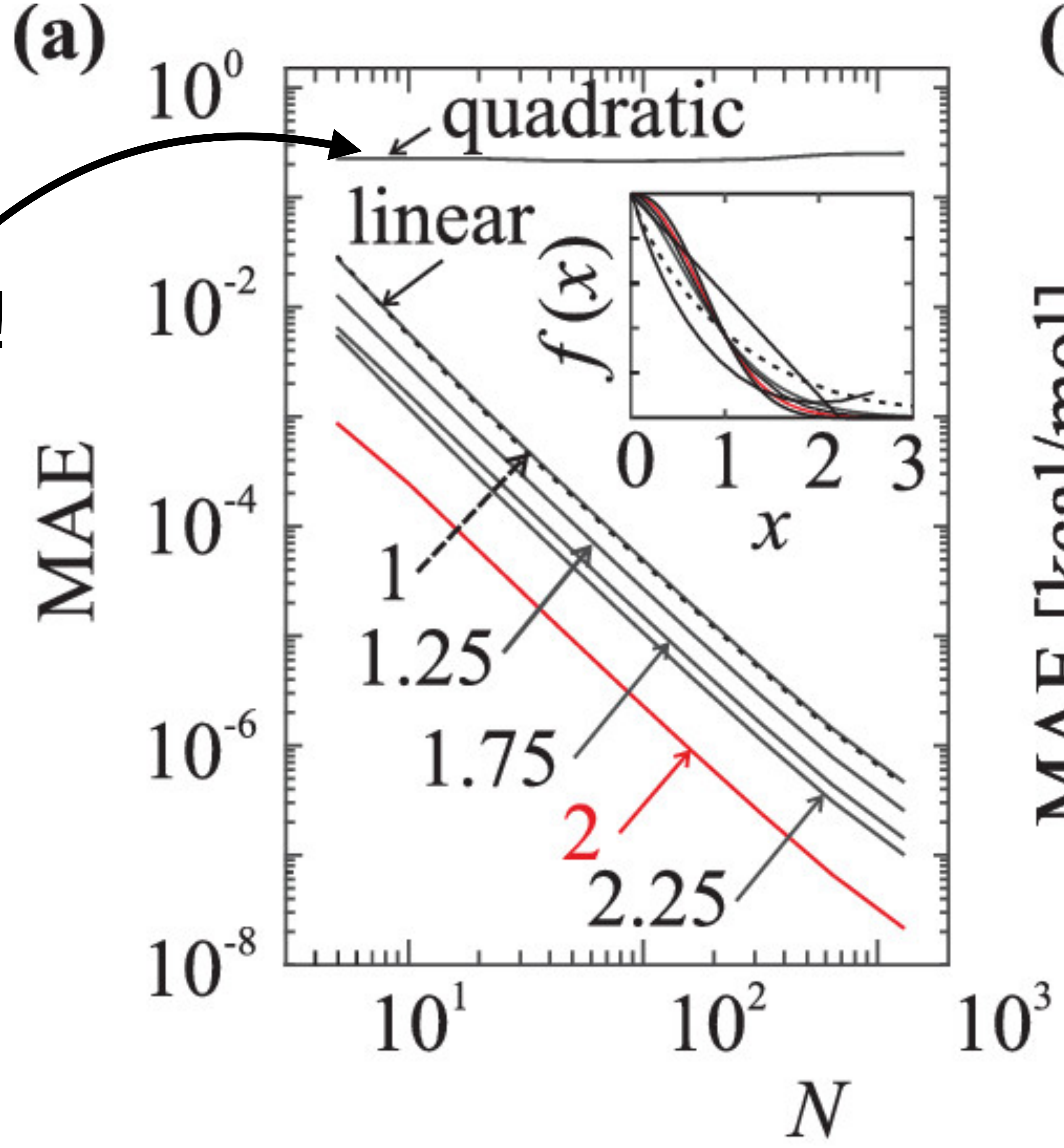
THEORY GROUP

Huang and von Lilienfeld, *J Chem Phys* **145** (2016)

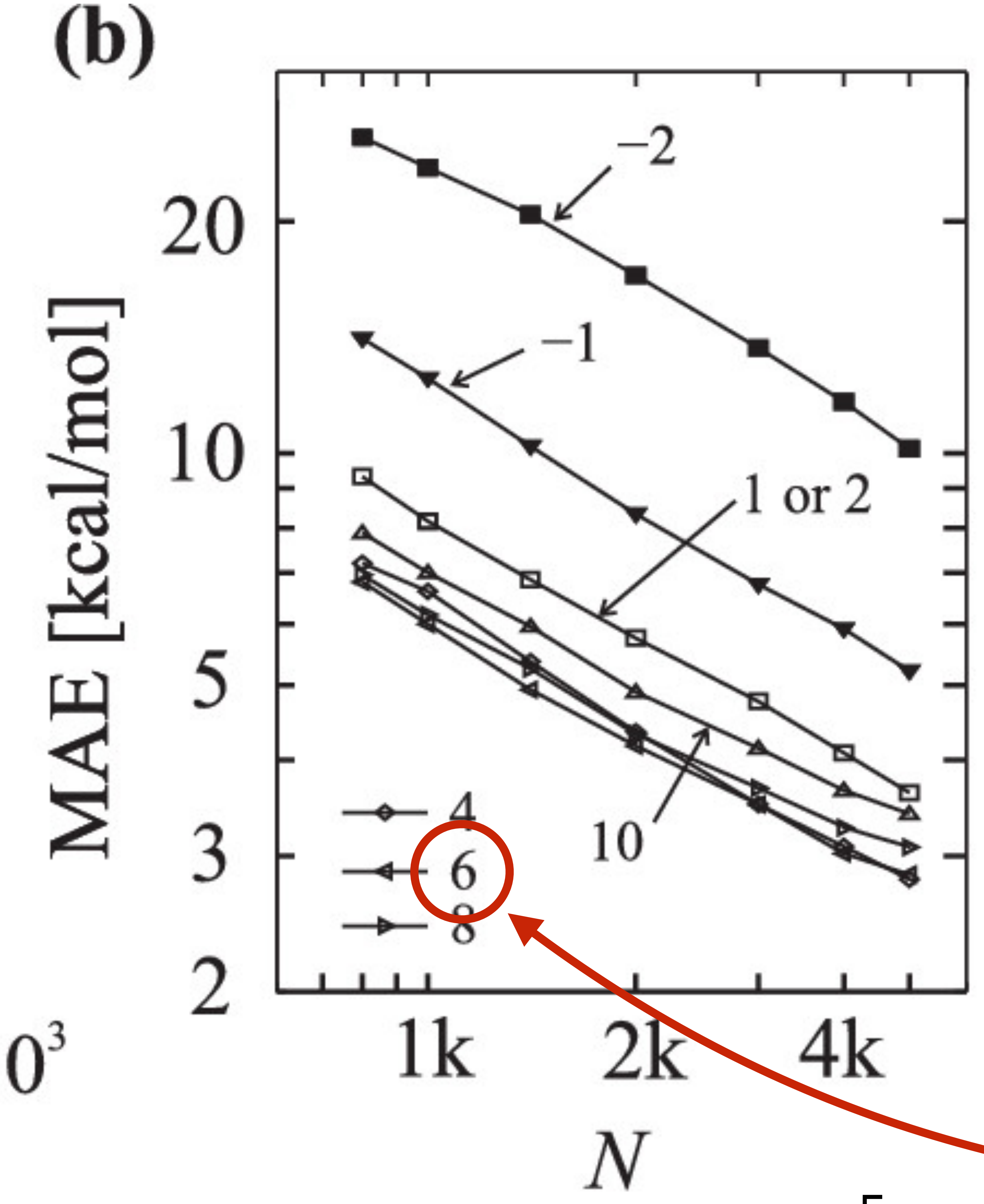
Max-Planck-Institut für Polymerforschung
Max Planck Institute for Polymer Research

Learning a Gaussian function

Non-unique representation!




Learning atomization energies




**Empirically test for relevant physics**

$$U_{\mathrm{LJ}}(r) = 4\epsilon\left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right]$$

THEORY GROUP

Huang and von Lilienfeld, *J Chem Phys* **145** (2016)

# Encoding symmetries in the ML model

# Encoding symmetries in ML models using group theory

Action of group $G$ on input sample $x \mapsto T_g(x)$


Risi Kondor, *Group theoretical methods in machine learning, PhD thesis* (2008)

# Encoding symmetries in ML models using group theory

Action of group $G$ on input sample $x \mapsto T_g(x)$

Can we find a kernel that is invariant to this group action? $f(T_g(x)) = f(x) \forall g \in G$

$$k(x, x') = k(T_g(x), T_{g'}(x'))$$

Risi Kondor, *Group theoretical methods in machine learning, PhD thesis* (2008)

# Encoding symmetries in ML models using group theory

Action of group $G$ on input sample $x \mapsto T_g(x)$

Can we find a kernel that is invariant to this group action? $f(T_g(x)) = f(x) \forall g \in G$

$$k(x, x') = k(T_g(x), T_{g'}(x'))$$

To ensure invariance, symmetrize the kernel

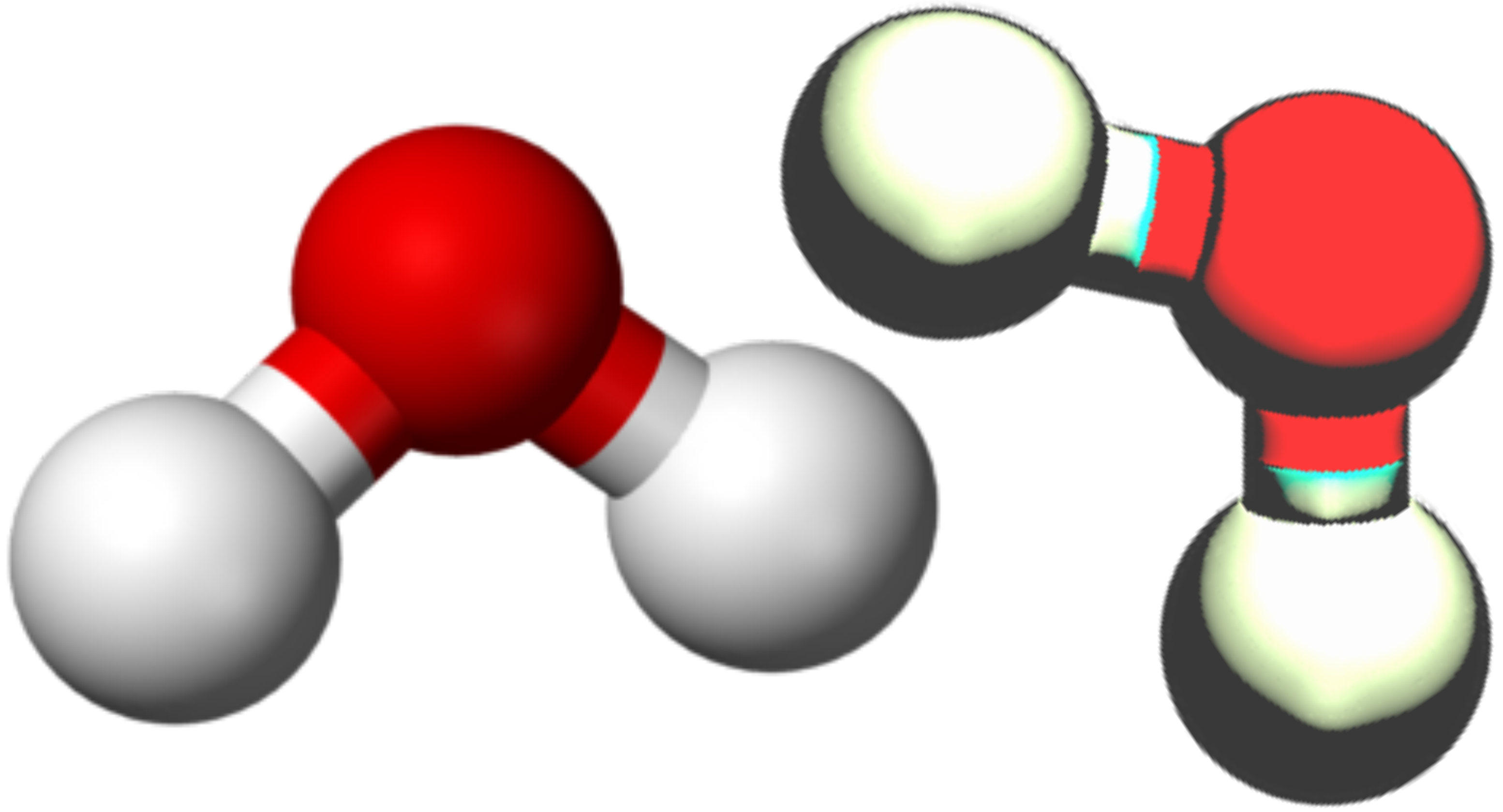
$$\boxed{k^G(x, x') = \frac{1}{|G|} \sum_{g \in G} k(x, T_g(x'))}$$

Risi Kondor, *Group theoretical methods in machine learning, PhD thesis* (2008)

# Example of symmetrized kernel

Vanilla/naïve kernel

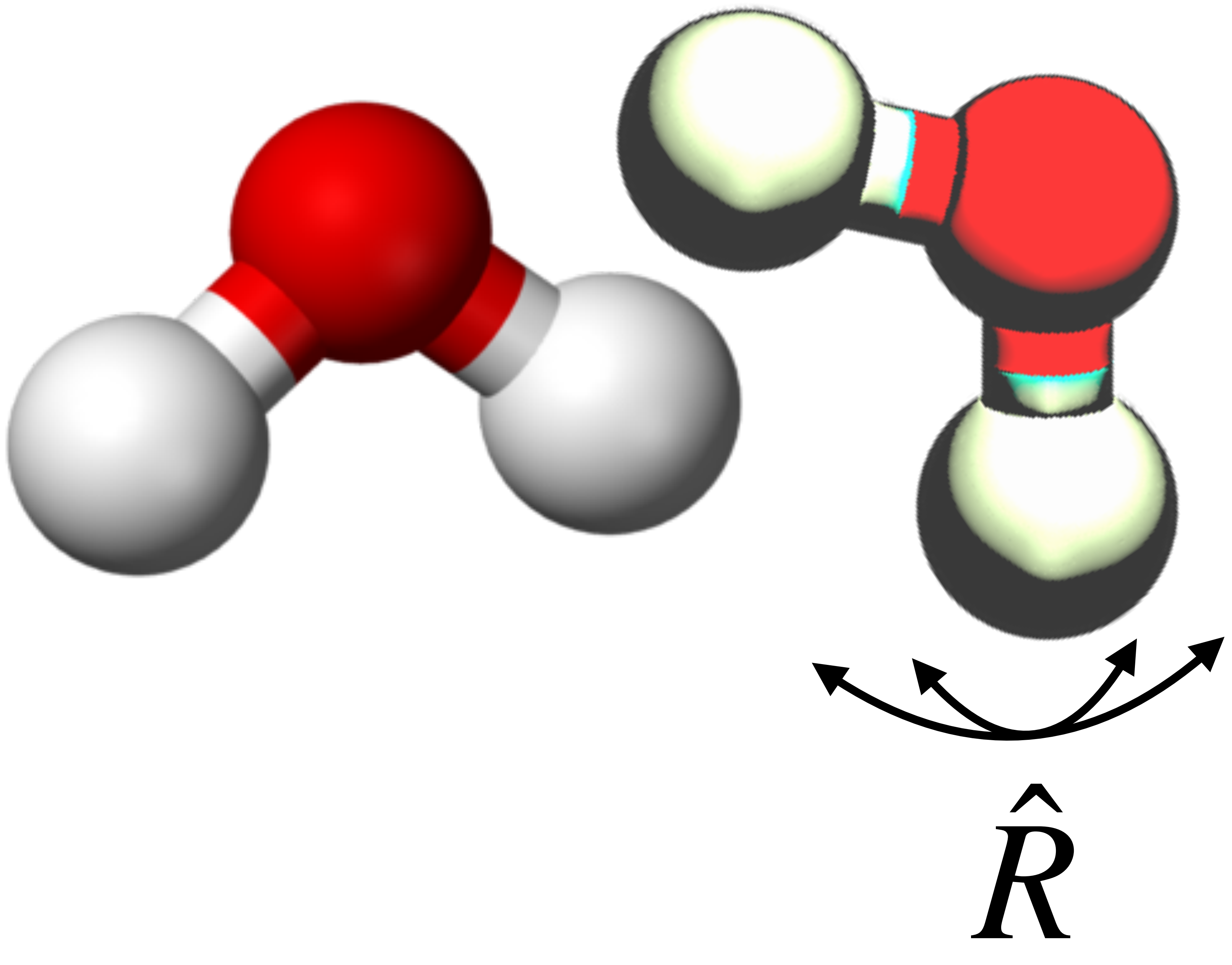
$$k(\rho, \rho') = \int \mathrm{d}\mathbf{r}\rho(\mathbf{r})\rho(\mathbf{r}')$$



# Example of symmetrized kernel

Vanilla/naïve kernel

$$k(\rho, \rho') = \int \mathbf{d}\mathbf{r}\rho(\mathbf{r})\rho(\mathbf{r}')$$




SOAP kernel/representation*

$$k(\rho, \rho') = \int \left| S(\rho, \hat{R}\rho') \right|^n \mathrm{d}\hat{R} =$$

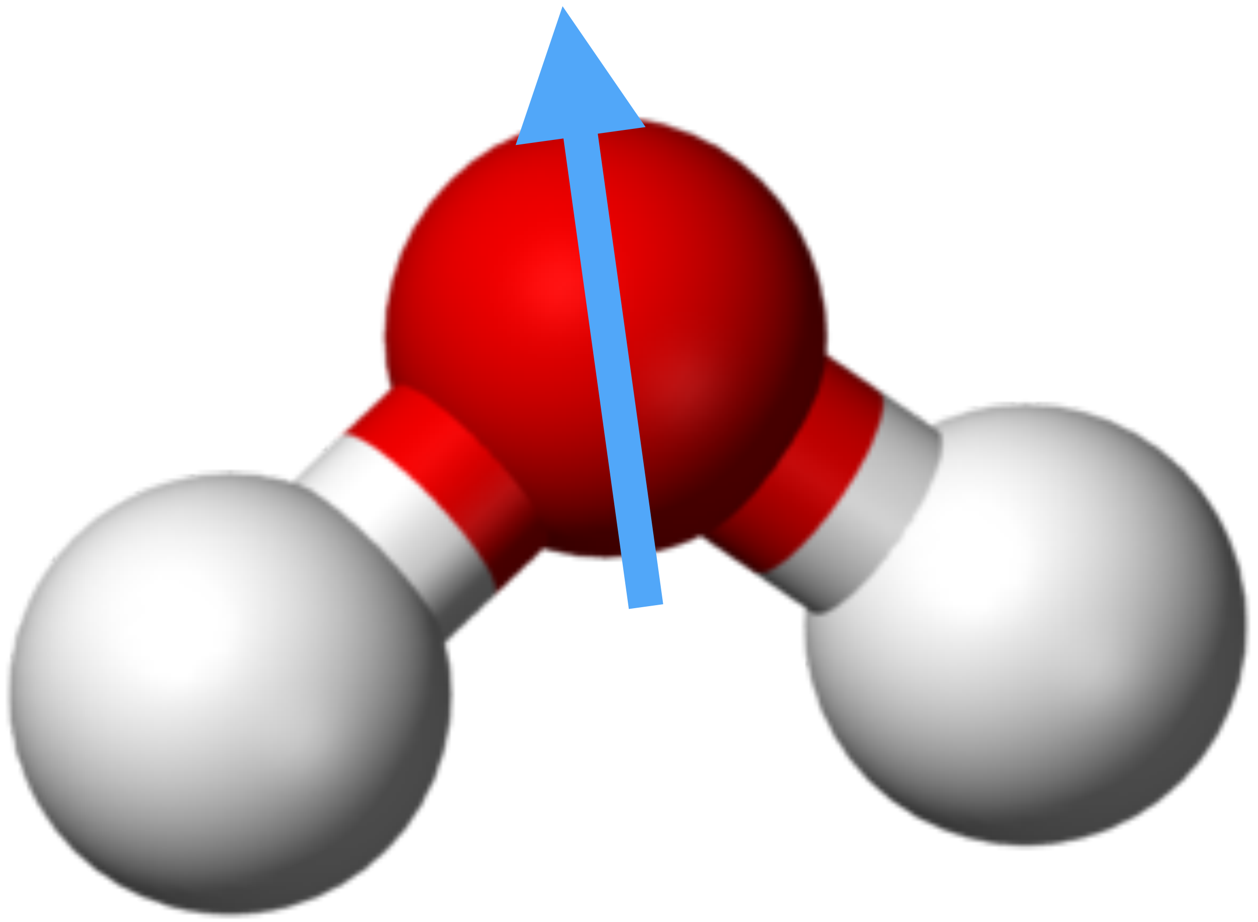
$$= \int \mathrm{d}\hat{R} \left| \int \rho(\mathbf{r})\rho'(\hat{R}\mathbf{r})\mathrm{d}\mathbf{r} \right|^n$$

*Smooth Overlap of Atomic Positions: is a distance metric between two samples


Bartók, Kondor, Csányi, *Phys Rev B* **87** (2013)

# Invariant vs. covariant properties

Tensorial property (e.g., dipole moment, force) **rotates** with the sample

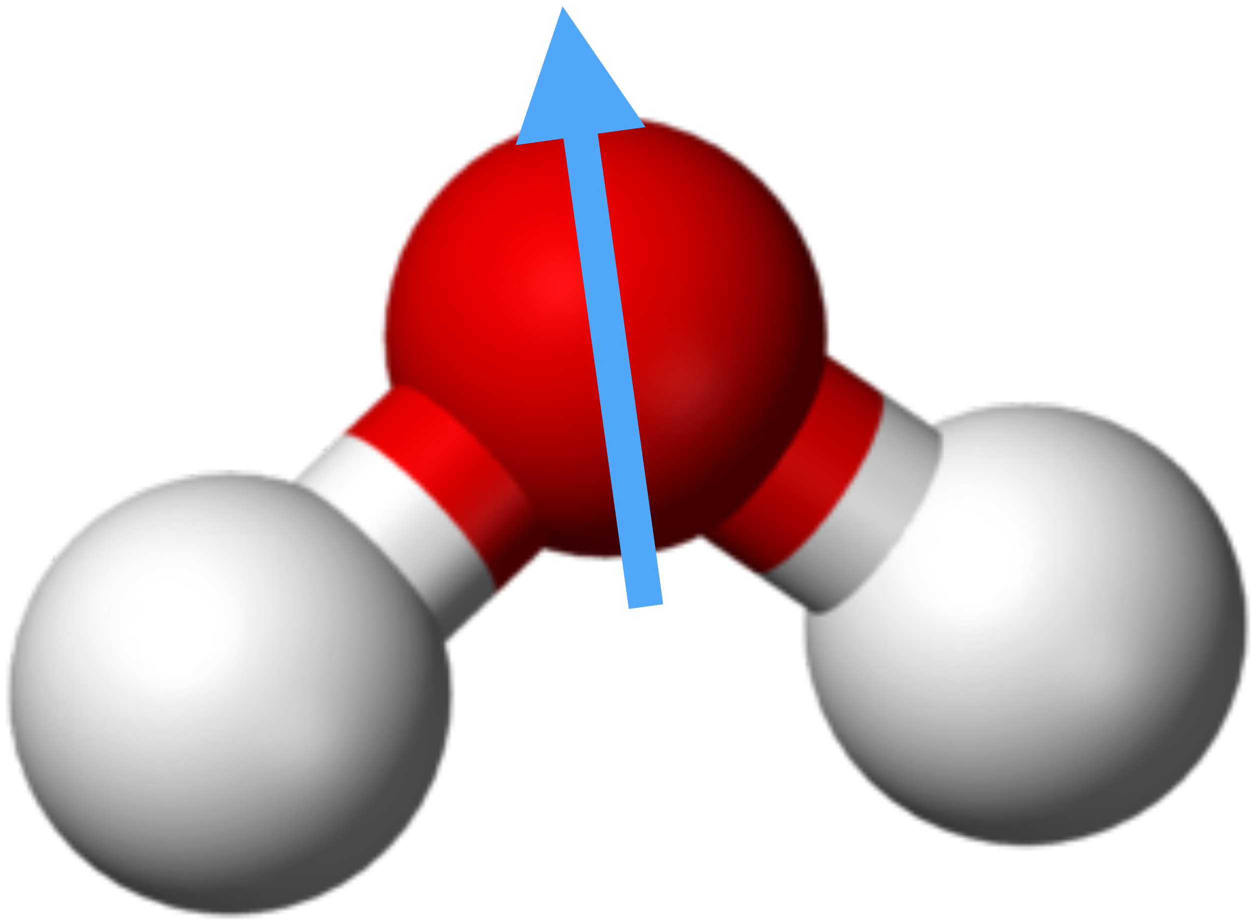


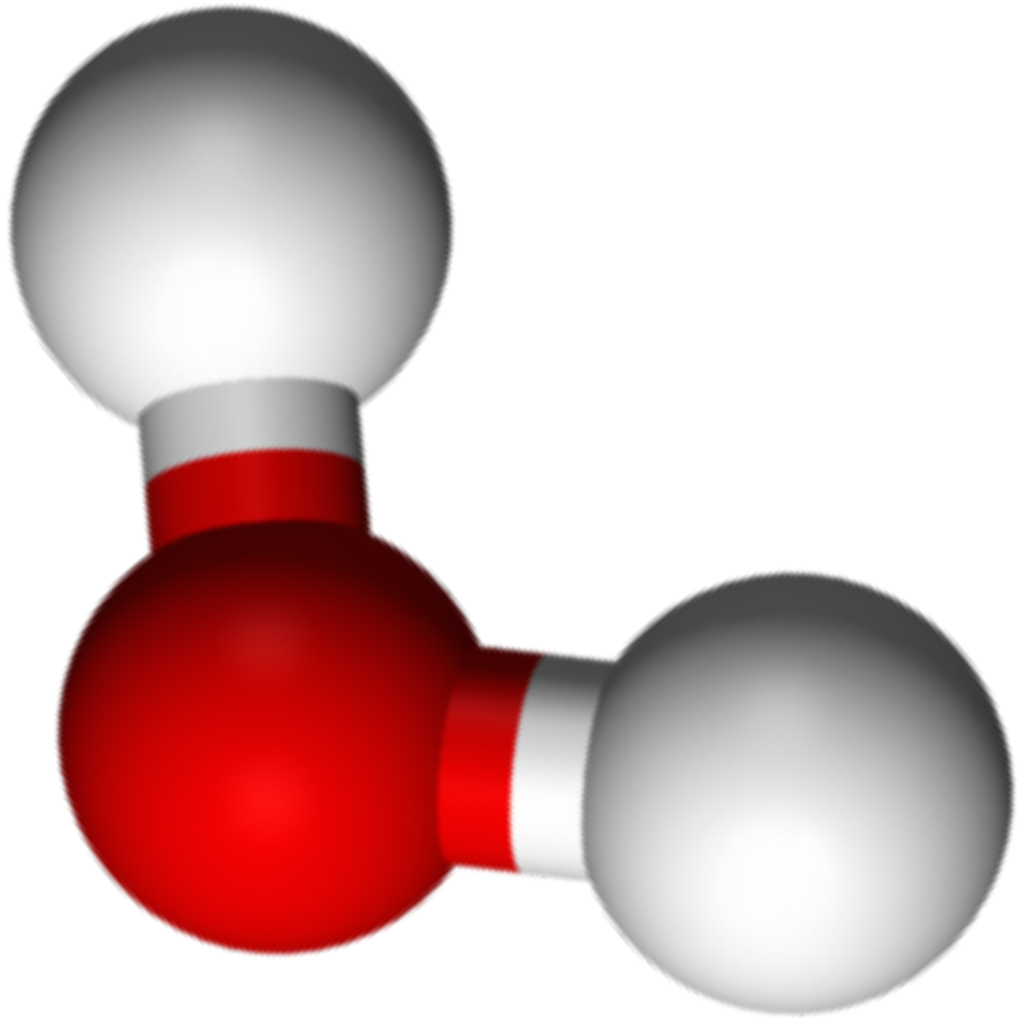


Glielmo, Sollich, De Vita, *Phys Rev B* **95** (2017)

# Invariant vs. covariant properties

Tensorial property (e.g., dipole moment, force) **rotates** with the sample





THEORY GROUP

Glielmo, Sollich, De Vita, *Phys Rev B* **95** (2017)

# Invariant vs. covariant properties

Tensorial property (e.g., dipole moment, force) **rotates** with the sample

Glielmo, Sollich, De Vita, *Phys Rev B* **95** (2017)

# Invariant vs. covariant properties

Tensorial property (e.g., dipole moment, force) **rotates** with the sample

“Build kernel so as to encode the rotational properties of the target property”

Glielmo, Sollich, De Vita, *Phys Rev B* **95** (2017)

# Covariant kernels

Encode rotational properties of the target property in the **kernel**

Glielmo, Sollich, De Vita, *Phys Rev B* **95** (2017)

# Covariant kernels

Encode rotational properties of the target property in the **kernel**

$$\hat{\mathbf{f}}(\mathcal{S}\rho \mid \mathcal{D}) = \mathbf{S}\hat{\mathbf{f}}(\rho \mid \mathcal{D})$$

Force prediction

Descriptor

Training data

Transformation (rotation/inversion)

THEORY GROUP

Glielmo, Sollich, De Vita, *Phys Rev B* **95** (2017)

# Covariant kernels

Encode rotational properties of the target property in the **kernel**

$$\hat{\mathbf{f}}(\mathcal{S}\rho \mid \mathcal{D}) = \mathbf{S}\hat{\mathbf{f}}(\rho \mid \mathcal{D})$$

Force prediction

Descriptor

Training data

Transformation (rotation/inversion)

"Transform the configuration, and the prediction transforms with it"

THEORY GROUP

Glielmo, Sollich, De Vita, *Phys Rev B* **95** (2017)

# Covariant kernels

$$\mathbf{K}(\mathcal{S}\rho, \mathcal{S}'\rho') = \mathbf{S}\mathbf{K}(\rho, \rho')\mathbf{S}'^{\mathrm{T}}$$

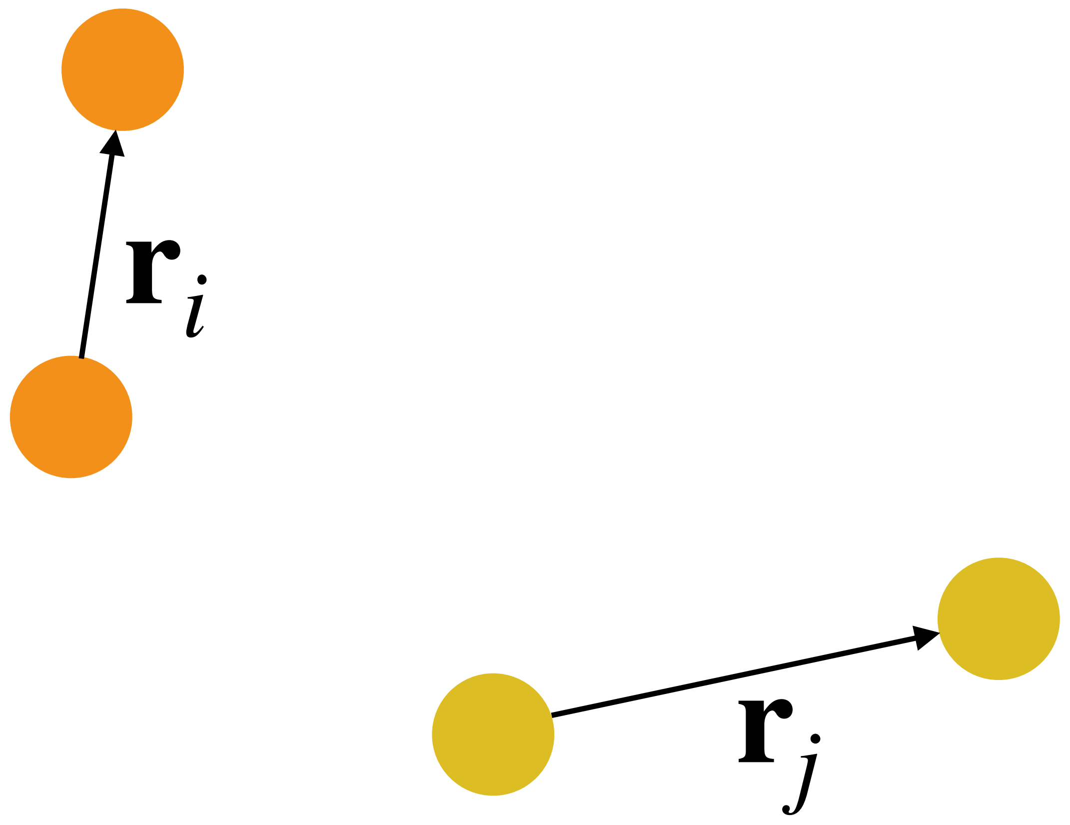
Kernel

Transformations (rotation/inversion)

Configurations

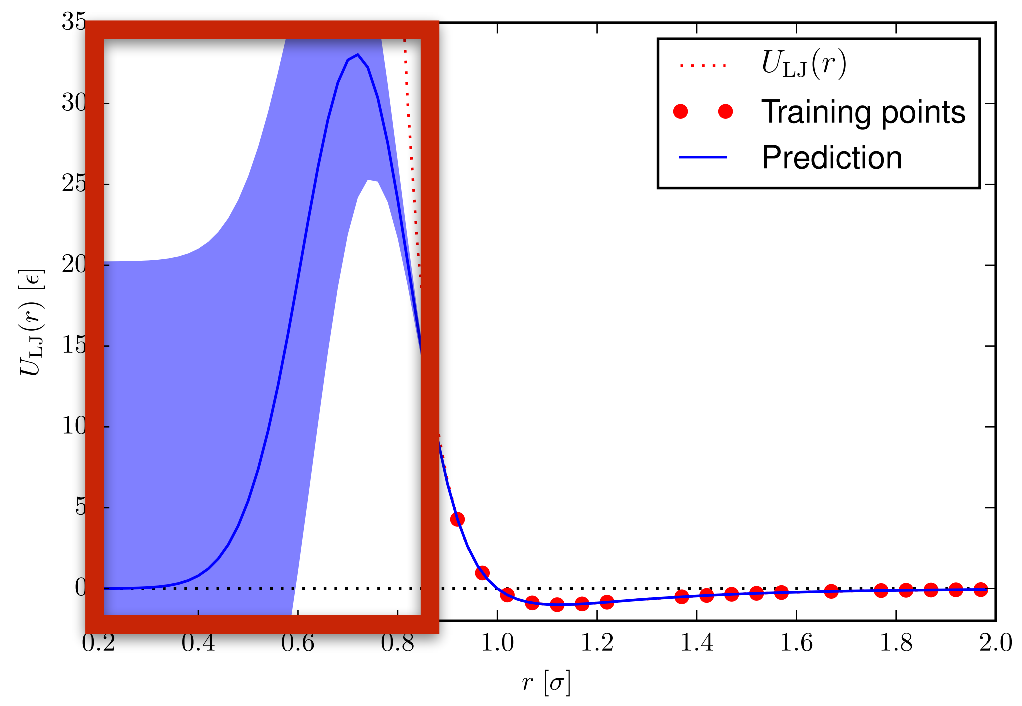
$$\mathbf{K}(\boldsymbol{\rho}, \boldsymbol{\rho}') = \int d\mathcal{R}\,\mathbf{R}k_b(\boldsymbol{\rho}, \mathcal{R}\boldsymbol{\rho}')$$

$$\mathbf{K}^{\mu}(\rho, \rho') = \frac{1}{L}\sum_{ij}\phi(r_i, r_j)\boxed{\mathbf{r}_i \otimes \mathbf{r}_j'^{\mathrm{T}}}$$




Glielmo, Sollich, De Vita, *Phys Rev B* **95** (2017)

# Conclusions

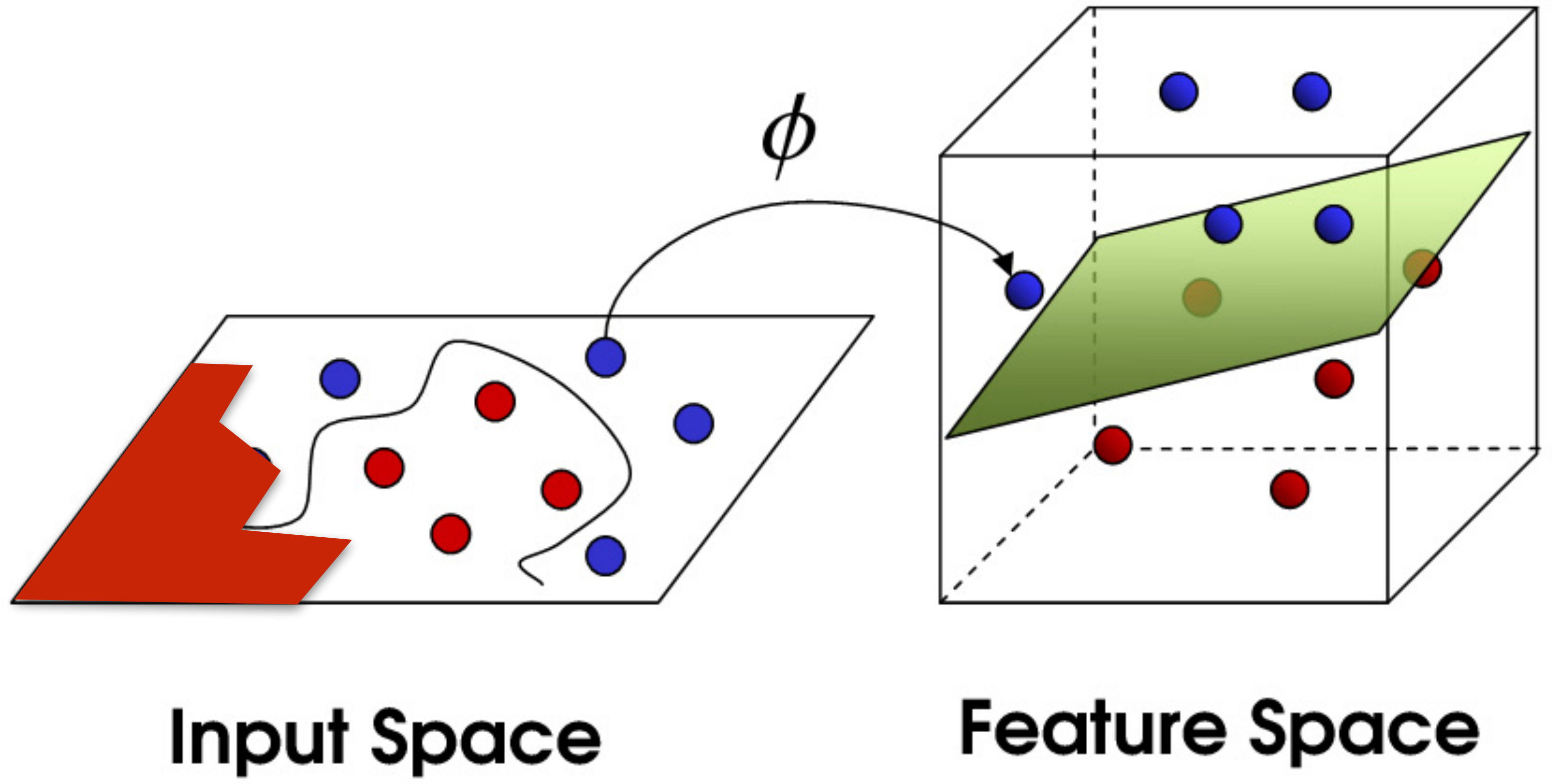



**Extrapolation in ML models of energy landscapes**

Can lead to catastrophic physics

**Take advantage of symmetries**

Noether: symmetry leads to conservation law




$$\mathbf{K}(\mathcal{S}\rho, \mathcal{S}'\rho') = \mathbf{S}\mathbf{K}(\rho, \rho')\mathbf{S}'^{\mathrm{T}}$$

**Build symmetries in ML model**

Work with subset of kernels that a priori satisfy conservation law