Conservation laws with ML for energy landscapes

Tristan Bereau
Max Planck Institute for Polymer Research
Mainz, Germany
Molecular dynamics
Molecular dynamics

\[ F = ma \]
Molecular dynamics

\[ \mathbf{F} = m \mathbf{a} \]

Specify interparticle forces: “force field”
Molecular dynamics

Specify interparticle forces: “force field”

\[ \mathbf{F} = ma \]

Numerically integrate particle positions
Molecular dynamics

Numerically integrate particle positions

\[ \mathbf{F} = m \mathbf{a} \]

Specify interparticle forces: “force field”

\[ E_{\text{total}} = \sum_{\text{bonds}} K_r (r - r_0)^2 + \sum_{\text{angles}} K_\theta (\theta - \theta_0)^2 + \sum_{\text{dihedrals}} \frac{\nu_i}{2} [1 + \cos(n\phi - \gamma)] + \sum_{i<j} \left( \frac{A_{ij}}{R_{ij}} - \frac{B_{ij}}{R_{ij}^6} + \frac{q_i q_j}{R_{ij}^{12}} \right) \]

Molecular dynamics

Numerically integrate particle positions

\[ E = \sum_{\text{bonds}} K_b (r - r_{eq})^2 + \sum_{\text{angles}} K_\theta (\theta - \theta_{eq})^2 + \sum_{\text{dihedrals}} \frac{\nu}{2} [1 + \cos(n\phi - \gamma)] + \sum_{i<j} \left[ \frac{A_i}{R_i R_j} - \frac{B_i}{R_i^n R_j^n} + \frac{q_i q_j}{\epsilon R_i R_j} \right] \]

Hydrophobic effect is roughly proportional to surface area

Continuum solvent model

Wikipedia

Durrant & McCammon, BMC Biol 9 (2011)
Molecular dynamics

Numerically integrate particle positions

\[ \mathbf{F} = ma \]

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”

Integration time step

\[ \text{fs} \quad \text{ps} \quad \text{ns} \quad \mu\text{s} \quad \text{ms} \quad \text{s} \]

Molecular dynamics

Numerically integrate particle positions

\[ \mathbf{F} = m \mathbf{a} \]

Specify interparticle forces: “force field”

Timescales of interest

<table>
<thead>
<tr>
<th>Integration time step</th>
<th>fs</th>
<th>ps</th>
<th>ns</th>
<th>µs</th>
<th>ms</th>
<th>s</th>
</tr>
</thead>
</table>
Molecular dynamics

Numerically integrate particle positions

\[ \mathbf{F} = m \mathbf{a} \]

Specify interparticle forces: “force field”

\[ E_{\text{total}} = \sum_{\text{bonds}} K (r - r_0)^2 + \sum_{\text{angles}} K (\theta - \theta_0)^2 + \sum_{\text{dihedrals}} \frac{V}{2} [1 + \cos(n\phi - \gamma)] + \sum_{i<j} \left[ \frac{A}{R_{ij}^{12}} - \frac{B}{R_{ij}^6} + q_i q_j \frac{1}{4\pi \varepsilon_0 R_{ij}} \right] \]

Timescales of interest

Emergent complexity

integration time step

fs ps ns μs ms s
Links to machine learning

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?
Disclaimer: only kernel methods covered

Kernel

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

Deep learning

- learns the representation
- complex mathematical structure
- data hungry
Kernel machine learning 101

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

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

Make a prediction:

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

Train your model:

\[ \alpha = (K + \lambda I)^{-1} U \]
Kernel machine learning 101

Define kernel

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

Make a prediction:

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

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:

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

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:

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

Train your model:

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

Conformational space missing from training
Extrapolation in machine learning

Define kernel

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

Make a prediction:

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

Train your model:

\[ \alpha = (K + \lambda I)^{-1} U \]
Bayesian inference

Bayes' formula

\[
p(f, f^* | y) = \frac{p(y | f) \ p(f, f^*)}{p(y)}
\]

- likelihood
- prior
- posterior

normalization


Ideally, we would like to know the error of our model on new data—predicting those is its purpose, after all. In statistical learning theory, this is measured by the risk of the model:

\[
R(f) = \mathbb{E}_{P_X, P_Y} L(Y, f(X))
\]

where \(P\) is the joint distribution of inputs and labels, and \(L: \mathbb{Y} \times \mathbb{Y} \to \mathbb{R}\) is a loss function measuring the error of a prediction. Eq. 20 is the expected error of \(f\). Unfortunately, \(P\) is usually not known, and \(R\) has to be estimated from a finite set of training data as the empirical risk:

\[
R_n(f) = \frac{1}{n} \sum_{i=1}^{n} L(y_i, f(x_i))
\]

Attributed to William of Ockham (early 14th century), but already known to Aristotle and Ptolemy in classical antiquity.

Gaussian processes

\[ f \sim \mathcal{GP}(m, k) \]
Gaussian processes

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

\[ f(x) \quad \mu_i = m(x_i) \quad \Sigma_{ij} = k(x_i, x_j) \]

random variable: value of the stochastic function at \( x \)

mean covariance

Gaussian processes

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

\[ f(x) \]

random variable:
value of the stochastic function at \( x \)

\[ \mu_i = m(x_i) \quad \Sigma_{ij} = k(x_i, x_j) \]

mean

covariance

\[ K\alpha = p \]

target property

Gaussian processes

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

- \( f(x) \) is a random variable: value of the stochastic function at \( x \)
- \( \mu_i = m(x_i) \) is the mean
- \( \Sigma_{ij} = k(x_i, x_j) \) is the covariance

Nonuniform: replace by repulsive potential (Csanyi and coworkers, Clementi and Noé, …)

\[ K\alpha = p \]

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

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

Lagrangian:

$$L = T - V$$

action

microtrajectory

kinetic energy

potential energy
Mechanics 101: Principle of least action

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

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

\[ \mathcal{S}[x^*(t)] = 0 \]
Mechanics 101: Principle of least action

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

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

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

stationarity under small perturbations leads to Euler-Lagrange equations

\[ \delta \mathcal{S} = \int_{t_1}^{t_2} dt \, L(x^* + \varepsilon, \dot{x}^* + \dot{\varepsilon}, t) - L(x^*, \dot{x}^*, t) \]

\[ = \int_{t_1}^{t_2} dt \left( \varepsilon \frac{\partial L}{\partial x} + \dot{\varepsilon} \frac{\partial L}{\partial \dot{x}} \right) = \int_{t_1}^{t_2} dt \left( \varepsilon \frac{\partial L}{\partial x} - \varepsilon \frac{d}{dt} \frac{\partial L}{\partial \dot{x}} \right) = 0 \]

Lagrangian \( L = T - V \)
Mechanics 101: Principle of least action

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

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

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

stationarity under small perturbations leads to Euler-Lagrange equations

\[
\begin{align*}
\delta \mathcal{S} &= \int_{t_1}^{t_2} dt \left( \varepsilon \frac{\partial L}{\partial x} + \dot{\varepsilon} \frac{\partial L}{\partial \dot{x}} \right) - \int_{t_1}^{t_2} dt \left( \varepsilon \frac{\partial L}{\partial x} - \varepsilon \frac{d}{dt} \frac{\partial L}{\partial \dot{x}} \right) \\
&= \int_{t_1}^{t_2} dt \left( \varepsilon \frac{\partial L}{\partial x} + \dot{\varepsilon} \frac{\partial L}{\partial \dot{x}} \right) - \int_{t_1}^{t_2} dt \left( \varepsilon \frac{\partial L}{\partial x} - \varepsilon \frac{d}{dt} \frac{\partial L}{\partial \dot{x}} \right) = 0
\end{align*}
\]

integration by parts & \( \varepsilon(t_1) = \varepsilon(t_2) = 0 \)
From symmetries, to invariants, to conserved quantities

\[ \mathcal{S}[x(t), y(t), z(t)] = \int dt \frac{m}{2} (\dot{x}^2 + \dot{y}^2 + \dot{z}^2) - mgz \]
From symmetries, to invariants, to conserved quantities

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

Introduce constant translations along \(x\) and \(y\):

\[ \mathcal{S}[x(t) + x_0, y(t) + y_0, z(t)] = \int dt \frac{m}{2} (\dot{x}^2 + \dot{y}^2 + \dot{z}^2) - mgz \]

\[ = \mathcal{S}[x(t), y(t), z(t)] \]
From symmetries, to invariants, to conserved quantities

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

Introduce constant translations along \( x \) and \( y \):

\[ \mathcal{S}[x(t) + x_0, y(t) + y_0, z(t)] = \int dt \frac{m}{2} (\dot{x}^2 + \dot{y}^2 + \dot{z}^2) - mgz \]

\[ = \mathcal{S}[x(t), y(t), z(t)] \]

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

\[ \frac{\partial L}{\partial x} - \frac{d}{dt} \frac{\partial L}{\partial \dot{x}} = 0 \]
From symmetries, to invariants, to conserved quantities

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

Introduce constant translations along \( x \) and \( y \):

\[ \mathcal{S}[x(t) + x_0, y(t) + y_0, z(t)] = \int dt \frac{m}{2} \left( \dot{x}^2 + \dot{y}^2 + \dot{z}^2 \right) - mgz \]

\[ = \mathcal{S}[x(t), y(t), z(t)] \]

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

\[ \frac{\partial L}{\partial x} - \frac{d}{dt} \frac{\partial L}{\partial \dot{x}} = 0 \]

\[ \frac{\partial L}{\partial \dot{x}} = m\dot{x} = \text{const.} \]

Translational invariance implies linear momentum conservation.
From symmetries to conserved quantities (cont’d)

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

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**
2 ways of encoding symmetries:
- Representation
- ML model
Encoding symmetries in the representation
Translational and rotational symmetries

Behler-Parrinello

\[ G_i^1 = \sum_{j \neq i} e^{-\eta (R_{ij} - R_i)^2} f_c (R_{ij}) \]

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

Coulomb matrix

\[ C_{ij} = \begin{cases} \frac{1}{2} Z_i^{2.4} & \forall \ i = j \\ \frac{Z_i Z_j}{|R_i - R_j|} & \forall \ i \neq j \end{cases} \]


Encoding symmetries in the ML model
Encoding symmetries in ML models using group theory

Action of group $G$ on input sample $x \mapsto T_g(x)$
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? $f(T_g(x)) = f(x) \forall g \in G$

$$k(x, x') = k(T_g(x), T_g(x'))$$
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

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

Example of symmetrized kernel

SOAP kernel/representation*

\[ S(\hat{R}) \equiv S(\rho, \hat{R}\rho') = \int dr \rho(r)\rho'(\hat{R}r) = \sum_{i,i'} \sum_{l,m} D_{lm'lm''}^{l'}(\hat{R}) \int dr c_{i,lm}^* (r) c_{i',lm'}^* (r) \int d\hat{r} Y_{lm}(\hat{r}) Y_{lm'}(\hat{r}) \]

\[ k(\rho, \rho') = \int |S(\rho, \hat{R}\rho')|^n d\hat{R} = \int d\hat{R} \left| \int \rho(r)\rho'(\hat{R}r)dr \right|^n \]

*Smooth Overlap of Atomic Positions: is a distance metric between two samples
Invariant vs. covariant properties

Tensorial property (e.g., dipole moment, force) rotates with the sample
Invariant vs. covariant properties

Tensorial property (e.g., dipole moment, force) rotates with the sample.
Invariant vs. covariant properties

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

Encode rotational properties of the target property in the kernel
Covariant kernels

Encode rotational properties of the target property in the kernel

\[ \hat{f}(S\rho \mid \mathcal{D}) = S\hat{f}(\rho \mid \mathcal{D}) \]

Force prediction
Descriptor
Training data
Transformation (rotation/inversion)

Covariant kernels

Encode rotational properties of the target property in the kernel

\[ \hat{f}(S\rho \mid \mathcal{D}) = S\hat{f}(\rho \mid \mathcal{D}) \]

“Transform the configuration, and the prediction transforms with it”
Covariant kernels

\[ K(S_\rho, S'_\rho') = SK(\rho, \rho')S'^T \]

Transformations (rotation/inversion)

\[ K(\rho, \rho') = \int dR Rk_b(\rho, R\rho') \]

\[ K^\mu(\rho, \rho') = \frac{1}{L} \sum_{ij} \phi(r_i, r_j) r_i \otimes r_j^T \]
Extension to higher-order tensors

\[
K^Q(S \rho, S' \rho') = S'S^T K^Q(\rho, \rho') SS'T
\]

\[
K^Q(\rho, \rho') = \frac{1}{L} \sum_{ij} (R_i^T \otimes R_j) \Phi(r_i, r_j) (R_i^T \otimes R_j)
\]

aligns \(r_i\) onto \(\hat{z}\)

\[
\Phi(r_i, r_j) = e^{-\frac{\alpha_{ij}^2}{4\sigma^2}} \int d\alpha \int d\beta \int d\gamma \frac{\sin \beta}{8\pi^2} r_{ij} \cdot \cos \beta
\]

\[
\begin{pmatrix}
\varphi_1 & 0 & 0 & 0 & \varphi_2 & 0 & 0 & 0 & 0 \\
0 & \varphi_1 & 0 & -\varphi_2 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \varphi_3 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -\varphi_2 & 0 & \varphi_1 & 0 & 0 & 0 & 0 & 0 \\
\varphi_2 & 0 & 0 & 0 & \varphi_1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \varphi_3 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & \varphi_3 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \varphi_3 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & \varphi_4 & 0 \\
\end{pmatrix}
\]
Application to intermolecular interactions
Transferable model for intermolecular interactions

Physics-based models
- Encode laws, symmetries
- Little chemical information

Data-driven models
- Need to *learn* laws, symmetries
- Interpolate across chemistry

Compositional (e.g., benzene vs. chlorobenzene)
Transferable model for intermolecular interactions

Physics-based models
- Encode laws, symmetries
- Little chemical information

Data-driven models
- Need to *learn* laws, symmetries
- Interpolate across chemistry

Any small molecule made of H, C, O, N
neutral compounds

Compositional (e.g., benzene vs. chlorobenzene)
Physics-based aspect

Long-ranged

- Static electrostatics
- Many-body dispersion
- Polarization

Short-ranged

- Charge penetration
- Repulsion
- (Charge transfer)

**Physics-based aspect**

### Long-ranged

**Perturbation theory**
- Static electrostatics
- Many-body dispersion
- Polarization

\[ V_{\text{Coulomb}}(r) = \frac{1}{4\pi \epsilon_0} \frac{q_i q_j}{r} \]

### Short-ranged

**Overlap models**
- Charge penetration
- Repulsion
- (Charge transfer)

\[ S_{ij} = \int d^3r \rho_i(r) \rho_j(r) \]

Physics-based aspect

Long-ranged

• Static electrostatics
• Many-body dispersion
• Polarization

Use ML to predict atoms-in-molecules properties

• Multipole moments
• Hirshfeld ratios
• Atomic density widths/populations

Short-ranged

• Charge penetration
• Repulsion
• (Charge transfer)

Physics-based aspect

Long-ranged
- Static electrostatics
- Many-body dispersion
- Polarization

Short-ranged
- Charge penetration
- Repulsion
- (Charge transfer)

Multipole moments

Hirshfeld ratios

Use ML to predict atoms-in-molecules properties
- Multipole moments
- Hirshfeld ratios
- Atomic density widths/populations

Static multipole electrostatics

\[ V_{\text{Coulomb}}(r) = \frac{1}{4\pi\varepsilon_0} \frac{q_i q_j}{r} \]
Static multipole electrostatics

\[ V_{\text{Coulomb}}(r) = \frac{1}{4\pi \varepsilon_0} \frac{q_i q_j}{r} \]

\[ 4\pi \varepsilon_0 \Phi(r) = \frac{q}{R} + \frac{\mu_\alpha R_\alpha}{R^3} + \frac{1}{3} \Theta_{\alpha\beta} \frac{3R_\alpha R_\beta - R^2 \delta_{\alpha\beta}}{R^5} + \ldots \]

Stone, *The Theory of Intermolecular Forces*
Bereau and Meuwly, *Many-Body Effects and Electrostatics in Biomolecules*
Static multipole electrostatics

\[ V_{\text{Coulomb}}(r) = \frac{1}{4\pi \varepsilon_0} \frac{q_i q_j}{r} \]

\[ 4\pi \varepsilon_0 \Phi(r) = \frac{q}{R} + \frac{\mu_\alpha R_\alpha}{R^3} + \frac{1}{3} \Theta_{\alpha \beta} \frac{3R_\alpha R_\beta - R^2 \delta_{\alpha \beta}}{R^5} + \ldots \]

Stone, The Theory of Intermolecular Forces
Bereau and Meuwly, Many-Body Effects and Electrostatics in Biomolecules
Static multipole electrostatics

\[
V_{\text{Coulomb}}(r) = \frac{1}{4\pi \varepsilon_0} \frac{q_i q_j}{r}
\]

\[
4\pi \varepsilon_0 \Phi(r) = \frac{q}{R} + \frac{\mu_\alpha R_\alpha}{R^3} + \frac{1}{3} \frac{3R_\alpha R_\beta - R^2 \delta_{\alpha\beta}}{R^5} + \ldots
\]

**Figure 1.1:** Field-line representations in Figure 1.1. A more convenient linear combination of pairs, hydrogen bonding, etc. to better describe halogenated compounds.

![Image](image.png)

**Image Caption:**

1. Monopole
2. Dipole
3. Quadrupole

**Image Details:**

- Dipole, quadrupole rotate with the sample.

---

**References:**

- Stone, *The Theory of Intermolecular Forces*
- Bereau and Meuwly, *Many-Body Effects and Electrostatics in Biomolecules*
Easier to learn H,O than C,N

Bereau, DiStasio Jr., Tkatchenko, von Lilienfeld, JCP 148, 241706 (2018); JCP Editor's Choice 2018
Multipoles: Correlation curves

(a) Monopole

(b) Dipole

(c) Quadrupole

C, N have more complex valencies
Intermolecular energy across conformations and composition

7 global parameters

Jurecka, Sponer, Cerny, Hobza, PCCP (2006)
Paton and Goodman, J Chem Inf Model (2009)

Bereau, DiStasio Jr., Tkatchenko, von Lilienfeld, JCP 148, 241706 (2018); JCP Editor’s Choice 2018
Energy conservation
Learning a vector field: matrix-valued kernels

Up to now: learning a scalar field \( f : \mathbb{R}^n \rightarrow \mathbb{R} \)

\[
f(x) = \sum_{i=1}^{N} \alpha_i K(x, x_i)
\]

\[
\alpha_i \in \mathbb{R}
\]

\[
K : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}
\]
Learning a vector field: matrix-valued kernels

Up to now: learning a scalar field $f : \mathbb{R}^n \rightarrow \mathbb{R}$

$$f(x) = \sum_{i=1}^{N} \alpha_i K(x, x_i)$$

$\alpha_i \in \mathbb{R}$

$K : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}$

Learning a vector field $f : \mathbb{R}^n \rightarrow \mathbb{R}^n$

$$f(x) = \sum_{i=1}^{N} \alpha_i K(x, x_i)$$

$\alpha_i \in \mathbb{R}^n$

$K : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}^{n \times n}$

Matrix-valued kernel

Learning a vector field \( f : \mathbb{R}^n \rightarrow \mathbb{R}^n \)

\[
f(x) = \sum_{i=1}^{N} \alpha_i K(x, x_i)
\]

\[
Nn = Nn \quad Nn \times Nn
\]

\( \alpha_i \in \mathbb{R}^n \)

\( K : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}^{n \times n} \)

By default: components learned \textit{independently}. \textbf{No prior on the vector field.}
Matrix-valued kernel

Learning a vector field \( f : \mathbb{R}^n \rightarrow \mathbb{R}^n \)

\[
f(x) = \sum_{i=1}^{N} \alpha_i K(x, x_i)
\]

\( N \times N \)

\( N \times N \times n \)

\( \alpha_i \in \mathbb{R}^n \)

By default: components learned independently. **No prior on the vector field.**

Time invariance leads to an **energy-conserving** force field (curl-free):

\[
\nabla \times f = 0
\]

Design matrix-valued kernel that is also curl free.
Recall the (translation-invariant) kernel:

\[ K(\mathbf{r}, \mathbf{r}') = \exp \left( -\frac{\|\mathbf{r} - \mathbf{r}'\|^2}{2\sigma^2} \right) = \phi \left( \|\mathbf{r} - \mathbf{r}'\| \right) \]

scalar RBF \( \Phi : \mathbb{R}^n \rightarrow \mathbb{R} \)
Recall the (translation-invariant) kernel:

\[ K(\mathbf{r}, \mathbf{r}') = \exp\left( -\frac{\|\mathbf{r} - \mathbf{r}'\|^2}{2\sigma^2} \right) = \phi\left( \|\mathbf{r} - \mathbf{r}'\| \right) \]

Construct matrix-valued RBF \( \Phi : \mathbb{R}^n \rightarrow \mathbb{R}^{n \times n} \) from a scalar RBF \( \phi(\mathbf{x}) = \phi(\|\mathbf{x}\|) \)

Apply linear differential operator: \( \Phi(\mathbf{x}) := (\mathbb{L} \phi)(\mathbf{x}) \)

Enforcing structure onto the vector field: Matrix-valued radial basis functions

Recall the (translation-invariant) kernel:

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

scalar RBF $\Phi: \mathbb{R}^n \rightarrow \mathbb{R}$

Construct matrix-valued RBF $\Phi: \mathbb{R}^n \rightarrow \mathbb{R}^{n \times n}$ from a scalar RBF $\phi(x) = \varphi(||x||)$

Apply linear differential operator: $\Phi(x) := (\mathcal{L}\phi)(x)$

Example: curl-free

$$\left(H\phi\right)_{ij} := \frac{\partial^2 \phi}{\partial x_i \partial x_j}$$

Vector fields: curl-free and divergence-free

\[ \Phi_{df}(x) = (H\phi)(x) - \text{Tr}\{(H\phi)(x)\} \cdot \mathbb{I} \]

\[ \Phi_{cf}(x) = - (H\phi)(x) \]

Figure 1. Learning a vector field decomposition: samples, learned field, divergence- and curl-free parts.
Kernel learning of a 2D potential

\[
\frac{V(x, y)}{k_B T} = \frac{1}{50}(x - 4)(x - 2)(x + 2)(x + 3) + \frac{1}{20}y^2 + \frac{1}{25}\sin(3(x + 5)(y - 6))
\]

Standard kernel

\[ q = \begin{pmatrix} 2(x + y) \\ x - y \end{pmatrix} \]

Learn the instantaneous forces

\[ (\{ f^x \}, \{ f^y \}) \]

and assume independence

\[ K(q, q') = \exp \left( -\frac{(q - q')^2}{2\sigma^2} \right) \]

\[ f^x_j = \sum_i \alpha^x_i \left( K(q_i, q_j) + \lambda \right) \]

\[ f^y_j = \sum_i \alpha^y_i \left( K(q_i, q_j) + \lambda \right) \]
Energy-conserving kernel

Learning in the gradient domain:

\[
\text{Cov} \left( \frac{\partial E(q_i)}{\partial r^k}, \frac{\partial E(q_j)}{\partial r^l} \right) = \frac{\partial q}{\partial r^k_i} \cdot \frac{\partial^2 K(q_i, q_j)}{\partial q \partial q} \frac{\partial q}{\partial r^l_j} + \frac{\partial K(q_i, q_j)}{\partial q} \cdot \frac{\partial^2 q}{\partial r^k_i \partial r^l_j}
\]

Mathias, Master Thesis, Bonn (2015); Glielmo et al., arXiv:1905.07626; Csanyi; Tkatchenko; Müller…
Energy-conserving kernel

Learning in the gradient domain:

\[
\text{Cov} \left( \frac{\partial E(q_i)}{\partial r^k}, \frac{\partial E(q_j)}{\partial r^l} \right) = \frac{\partial q}{\partial r^k_i} \cdot \frac{\partial^2 K(q_i, q_j)}{\partial q \partial q} \frac{\partial q}{\partial r^l_j} + \frac{\partial K(q_i, q_j)}{\partial q} \cdot \frac{\partial^2 q}{\partial r^k_i \partial r^l_j}
\]

\[ K_{\text{Hess}}(q^k_i, q^l_j) \]
Energy-conserving kernel

Learning in the gradient domain:

\[
\text{Cov}\left(\frac{\partial E(q_i)}{\partial r^k}, \frac{\partial E(q_j)}{\partial r^l}\right) = \frac{\partial q}{\partial r^k_i} \cdot \frac{\partial^2 K(q_i, q_j)}{\partial q \partial q} \frac{\partial q}{\partial r^l_j} + \frac{\partial K(q_i, q_j)}{\partial q} \cdot \frac{\partial^2 q}{\partial r^k_i \partial r^l_j}
\]

\[K(q, q') = \exp\left(-\frac{(q - q')^2}{2\sigma^2}\right)\]

\[K_{\text{Hess}}(q^k_i, q^l_j)\]

\[K_{\text{Hess}}(q^k_i, q^l_j) = \frac{\partial q}{\partial r^k_i} \cdot \frac{\partial q}{\partial r^l_j} \cdot \frac{\partial}{\partial q} \frac{\partial}{\partial q} \exp\left(-\frac{(q_i - q_j)^2}{2\sigma^2}\right)\]

\[= \frac{\partial q}{\partial r^k_i} \cdot \frac{\partial q}{\partial r^l_j} \cdot \frac{1}{\sigma^2} \left(1 - \frac{(q_i - q_j)^2}{\sigma^2}\right) \exp\left(-\frac{(q_i - q_j)^2}{2\sigma^2}\right)\]

Mathias, Master Thesis, Bonn (2015); Glielmo et al., \textit{arXiv:1905.07626}; Csanyi; Tkatchenko; Müller…
Energy-conserving kernel

Learning in the gradient domain:

\[
\text{Cov} \left( \frac{\partial E(q_i)}{\partial r^k}, \frac{\partial E(q_j)}{\partial r^l} \right) = \frac{\partial q}{\partial r^k} \cdot \frac{\partial^2 K(q_i, q_j)}{\partial q \partial q} \frac{\partial q}{\partial r^l} + \frac{\partial K(q_i, q_j)}{\partial q} \cdot \frac{\partial^2 q}{\partial r^k \partial r^l}
\]

\[K(q, q') = \exp \left( -\frac{(q - q')^2}{2\sigma^2} \right)\]

\[K_{\text{Hess}}(q^k_i, q^l_j) = \frac{\partial q}{\partial r^k_i} \cdot \frac{\partial q}{\partial r^l_j} \cdot \frac{\partial}{\partial q} \frac{\partial}{\partial q} \exp \left( -\frac{(q_i - q_j)^2}{2\sigma^2} \right)\]

\[= \frac{\partial q}{\partial r^k_i} \cdot \frac{\partial q}{\partial r^l_j} \frac{1}{\sigma^2} \left( 1 - \frac{(q_i - q_j)^2}{\sigma^2} \right) \exp \left( -\frac{(q_i - q_j)^2}{2\sigma^2} \right)\]
Energy-conserving kernel

Learning in the gradient domain:

\[
\text{Cov} \left( \frac{\partial E(q_i)}{\partial r^k}, \frac{\partial E(q_j)}{\partial r^l} \right) = \frac{\partial q}{\partial r_k^i} \cdot \frac{\partial^2 K(q_i, q_j)}{\partial q \partial q} \frac{\partial q}{\partial r^l_j} + \frac{\partial K(q_i, q_j)}{\partial q} \cdot \frac{\partial^2 q}{\partial r_k^i \partial r^l_j}
\]

\[K(q, q') = \exp \left( -\frac{(q - q')^2}{2\sigma^2} \right)\]

\[K_{\text{Hess}}(q_i^k, q_j^l) \]

\[K_{\text{Hess}}(q_i^k, q_j^l) = \frac{\partial q}{\partial r_k^i} \cdot \frac{\partial q}{\partial r^l_j} \frac{\partial}{\partial q} \frac{\partial}{\partial q} \exp \left( -\frac{(q_i - q_j)^2}{2\sigma^2} \right)\]

\[= \frac{\partial q}{\partial r_k^i} \cdot \frac{\partial q}{\partial r^l_j} \frac{1}{\sigma^2} \left( 1 - \frac{(q_i - q_j)^2}{\sigma^2} \right) \exp \left( -\frac{(q_i - q_j)^2}{2\sigma^2} \right)\]

enables energy conservation

Mathias, Master Thesis, Bonn (2015); Glielmo et al., arXiv:1905.07626; Csanyi; Tkatchenko; Müller…
Energy-conserving kernel

\[ K(q, q') = \exp \left( -\frac{(q - q')^2}{2\sigma^2} \right) \]

\[ \mathbf{q} = \begin{pmatrix} 2(x + y) \\ x - y \end{pmatrix} \quad \frac{\partial \mathbf{q}}{\partial x} = \begin{pmatrix} 2 \\ 1 \end{pmatrix} \quad \frac{\partial \mathbf{q}}{\partial y} = \begin{pmatrix} 2 \\ -1 \end{pmatrix} \]

\[ \mathbf{q} = \begin{pmatrix} x \\ y \end{pmatrix} \quad \frac{\partial \mathbf{q}}{\partial x} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad \frac{\partial \mathbf{q}}{\partial y} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \]

\[ \mathbf{K}_{\text{Hess}}(q_i^k, q_j^l) = \left. \frac{\partial \mathbf{q}}{\partial r_i^k} \cdot \frac{\partial \mathbf{q}}{\partial r_j^l} \frac{\partial}{\partial \mathbf{q}} \frac{\partial}{\partial \mathbf{q}} \exp \left( -\frac{(q_i - q_j)^2}{2\sigma^2} \right) \right|_{\mathbf{r}_i = \mathbf{r}_j} \]

\[ = \left. \frac{\partial \mathbf{q}}{\partial r_i^k} \cdot \frac{\partial \mathbf{q}}{\partial r_j^l} \frac{1}{\sigma^2} \left( 1 - \frac{(q_i - q_j)^2}{\sigma^2} \right) \exp \left( -\frac{(q_i - q_j)^2}{2\sigma^2} \right) \right|_{\mathbf{r}_i = \mathbf{r}_j} \]

enables energy conservation
Standard kernel

Number of training samples

Force MAE \( f / r \)

-4 -2 0 2 4

-4 -2 0 2 4

\(-\partial E\)

\(-\partial E\)

\(f^x (ML)\)

\(f^y (ML)\)

Standard kernel
Energy-conserving ML

Number of training samples

Force MAE $f/\nu$

- $f^x$ (ML standard)
- $f^y$ (ML standard)
- $f$ (ML energy conserving)

Graphs showing $f^x$ and $f^y$ for different training samples.
Comparison between ML models

Learning from only 50 samples

Energy-conserving force field useful in the low-data regime
Building symmetries in ML force fields

Incorporate symmetries and conservation laws to minimize the training data and represents the rank relevant within the scope of the training dataset are successfully ignored.

We take advantage of the fact that the minimal pairwise assignment matrices and yields a model with the exact same number of parameters as a 200 picosecond DFT MD trajectory at 500 K following the Boltzmann distribution. Subsequently, a globally consistent permutation graph is constructed that jointly assigns all configurations even if they do not form a chemically feasible transformation that de

Statistical

\[ \hat{f}_F(x) = \sum_i^{M} \sum_l^{3N} \sum_q^{S} (P_q \alpha_i)_l \frac{\partial}{\partial x_l} \nabla \kappa(x, P_q x_i) \]

- Incorporate symmetries and conservation laws to minimize the training data
- Reduces dataset size: can target extremely accurate quantum chemistry

Chmiela et al., Nat. Comm. 9:3887 (2018)
Building symmetries in ML force fields

Operators in quantum machine learning: Response properties in chemical space

Cite as: J. Chem. Phys. 150, 064105 (2019); doi: 10.1063/1.5053562
Submitted: 23 August 2018 • Accepted: 19 November 2018 •
Published Online: 12 February 2019

Anders S. Christensen, Felix A. Faber, and O. Anatole von Lilienfeld

Affiliations
Department of Chemistry, University of Basel, Basel, Switzerland

Electronic mail: anatole.vonlilienfeld@unibas.ch

Chmiela et al., Nat. Comm. 9:3887 (2018)
Conclusions

Extrapolation in ML models of energy landscapes
Can lead to catastrophic physics

Take advantage of symmetries
Noether: symmetry leads to conservation law

\[ K(S\rho, S'\rho') = SK(\rho, \rho')S'^T \]

Build symmetries in ML model
Work with subset of kernels that a priori satisfy conservation law