Machine Learning for PES Approximation

Khachik Sargsyan

Sandia National Laboratories, Livermore, CA

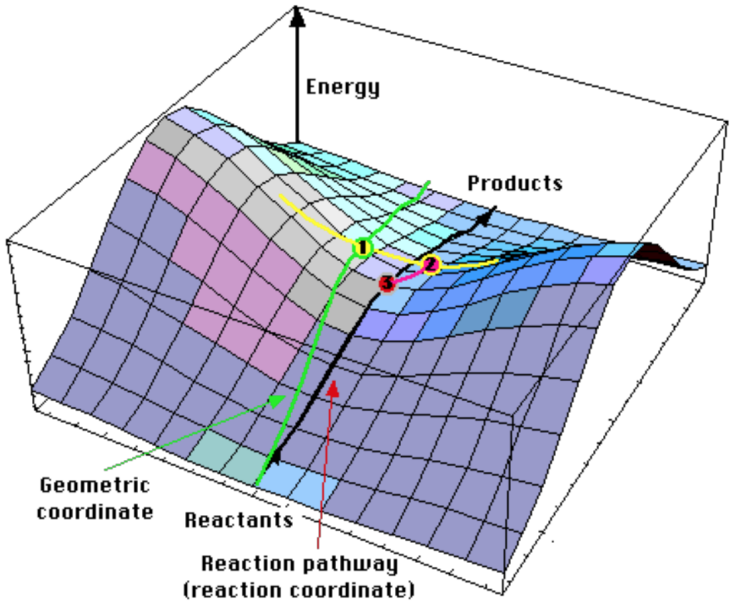
ECC Meeting
SNL-CA
October 3-5, 2018

Goals

- General: machine learning for rapid saddle point search
- Specific: machine learning for PES approximation
  - To feed into saddle point search algorithm
- So far:
  - Prashant Rai $\rightarrow$ investigation of low-rank approximations, with gradient information
  - Jonathan Vo $\rightarrow$ derivative-free optimization benchmark study on synthetic high-dimensional functions
  - Initial proof-of-concept code for derivative-free optimization $\rightarrow$ Eric Hermes
  - Scoping the problem, to complement Eric's work
  - Literature review, ML for PES

Main target: PES approximation

- Accurate and fast surrogates for PES to replace QM computations for studies requiring many PES inquiries
  - saddle point search, transition paths, barrier heights
  - rapid assessment of reaction characteristics
  - automate the discovery of reactive pathways



Physical vs Mathematical PES Apprx's

- Simplified energy expressions based on physical considerations
- Central idea: decompose into low-dim bonding of 2-,3-,4-body
- Classical force fields, reactive potentials
- (Embedded) atom method
- Brenner potential, Tersoff potential, Stillinger-Weber potential, Lennard-Jones potential
- Accuracy limited by the imposed functional form

Physical vs **Mathematical** PES Apprx's

- also called Machine Learning (ML); No direct physical meaning
- Huge growth last decade
  - Weighted interpolation [Ischtwan 1994; Dowes, 2007-09; Maisuradze, 2009]
  - Permutationally invariant polynomials [Xie, 2010]
  - Gaussian processes [Bartok, Csanyi 2010-15; Mills, 2012; Rupp, 2013; Cui, 2016; Uteva, 2017; Guan, 2018; Schmitz, 2018]
  - Low-rank tensor expansions [Jackle, 1996; Baranov, 2015; Rai, 2017, 2018]
  - Support vector machines, kernel regression [Le, 2009; Balabin, 2011; Dral, 2017]
  - Artificial neural networks (NN) [Blank, 1995; Tai No, 1997; Prudente, 1998; Lorenz, 2004; Witkoskie, 2005; Manzhos, 2006-09; Malshe, 2008; Le, 2009] [Behler, 2010-16; Handley, 2010, 2014; Jiang, 2013; Li, 2013; Dolgirev, 2016; Khorshidi, 2016; Peterson, 2016; Carr, 2016; Kolb, 2016; Shao, 2016; Chmiela, 2017; Cubuk, 2017; McGibbon, 2017; Smith, 2017; Schutt, 2017; Yao, 2017; Hajinazar, 2017; Bereau, 2018; Lubbers, 2018; Unke, 2018; Wang, 2018; Natarajan, 2018; Zhang, 2018; Onat, 2018]

ML Potentials

- Particularly useful when:
  - Many interaction types: covalent, ionic, metallic bonding, dispersion
  - Uncommon environments: amorphous systems, phase transitions
  - Making/breaking bonds
- Requirements:
  - Accurate
  - Fast-to-evaluate
  - Analytic derivatives
  - Systematic improvement
  - Reactive, i.e. enable making/breaking bonds
  - High-dimensional
  - Transferable/generalizable to unseen atomic configurations
  - **Translational, rotational, permutational invariance**

Main ingredients

- Training data $(\boldsymbol{x}_i, E_i)$ for $i = 1, \ldots, S$, and $\boldsymbol{x}_i \in R^{3N}$
  - *ab initio*, DFT
- Input representation, aka fingerprint, aka descriptor
  - Internal coordinates,
    $\boldsymbol{x}_i \rightarrow z_i \equiv$(bond length, bond angle, and dihedral angle).
    Dimensionality $3N \rightarrow 3N - 6$
  - Morse variables, from internuclear distance $p_{jk} = e^{-||\boldsymbol{x}_j - \boldsymbol{x}_k||/\gamma}$
  - Symmetry functions [Behler, 2010]
  - Invariant low-order polynomials [Xie, 2010]. Think of $x_i^2 y_i \rightarrow x_i^2 y_i + x_i y_i^2$
- Parameterized form of the approximation class

$$f_p(\cdot)$$

- Loss function

$$\min_p \sum_{i=1}^{N} \left(E_i - f_p(\boldsymbol{z}_i(\boldsymbol{x}))\right)^2$$

- Regularization
  - Parameter norm penalties; Early stopping; Multiple-fit validation

Permutationally invariant polynomials (PIP)

- Symmetrization of a primitive monomial basis
- Collect $N(N-1)/2$ internuclear distances

$$r_{jk} = ||\boldsymbol{x}_j - \boldsymbol{x}_k||, \text{ for } j,k = 1,\ldots,N \text{ and } j \neq k$$

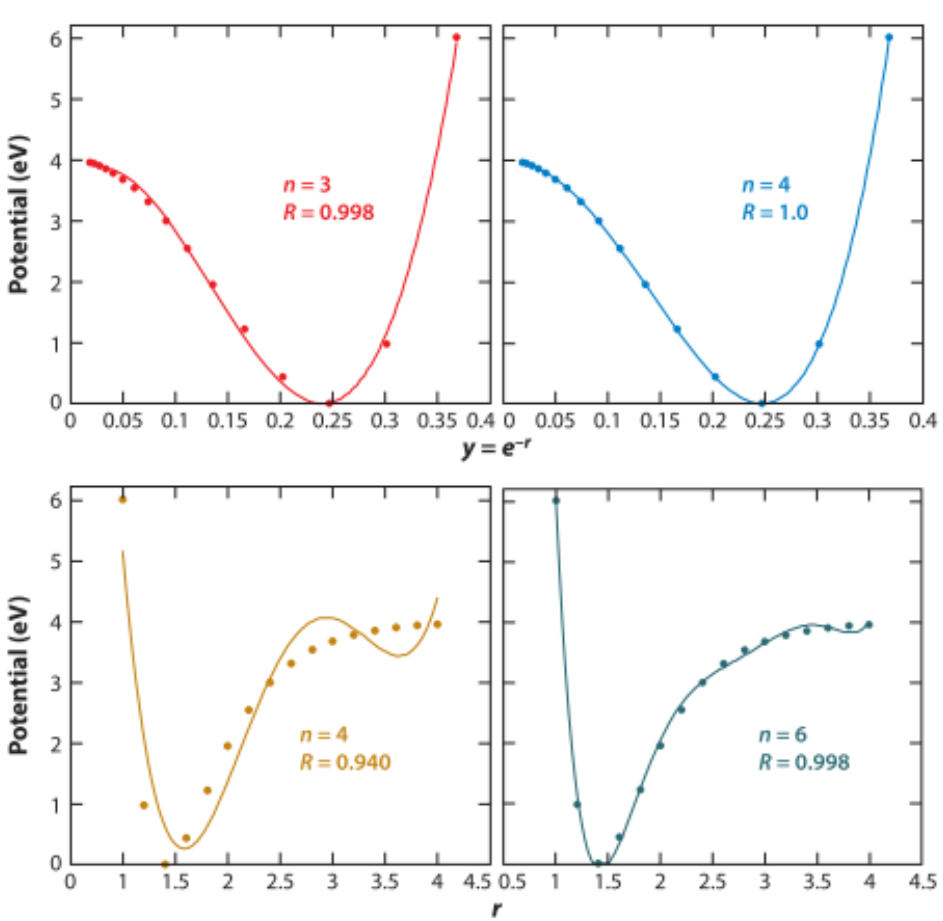
- E.g. tetratomic example

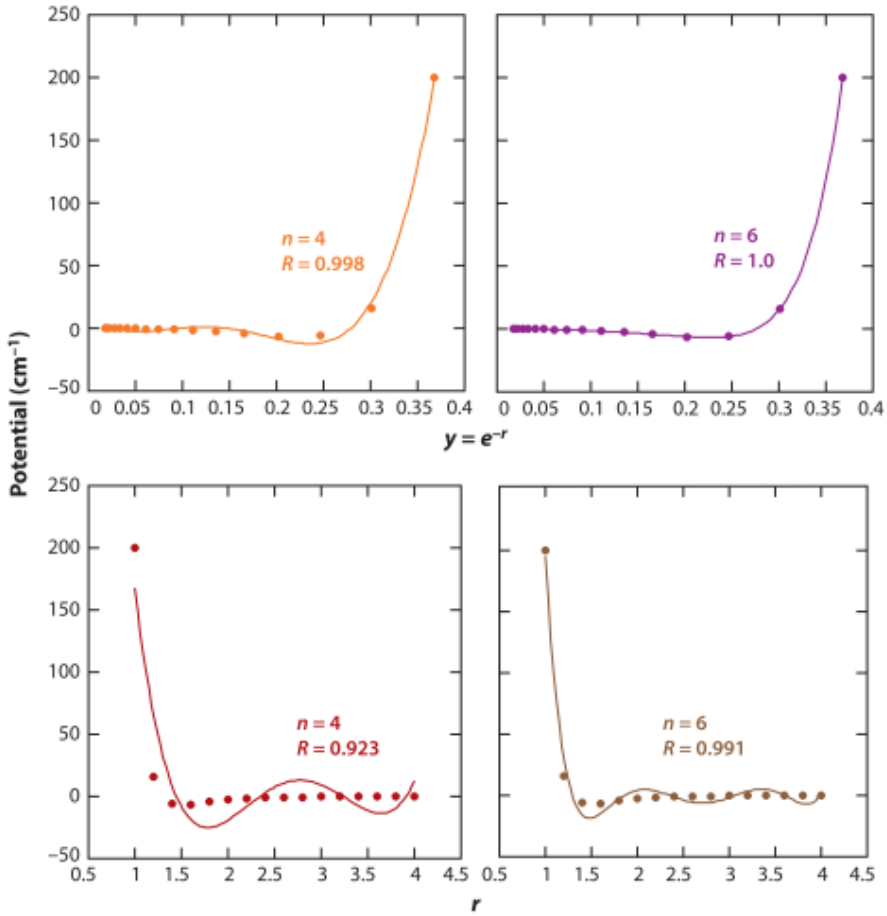
$$(z_1, z_2, z_3, z_4, z_5, z_6) = (r_{12}, r_{13}, r_{14}, r_{23}, r_{24}, r_{34})$$

- Polynomial basis $z_1^3 z_2^2 = r_{12}^3 r_{13}^2$ is replaced with $r_{12}^3 r_{13}^2 + r_{12}^2 r_{13}^3$
- A lot of bookkeeping. Scalability is not clear, but...
- Can be used as a preconditioning for NN [Li, 2013; Jiang, 2014, Kolb, 2016], or GP [Qu, 2018]

... actually, use Morse variables instead

- Collect $N(N-1)/2$ internuclear distances $r_{jk} = ||\boldsymbol{x}_j - \boldsymbol{x}_k||$
- Enumerate $\boldsymbol{y} = e^{-r_{jk}/\gamma}$

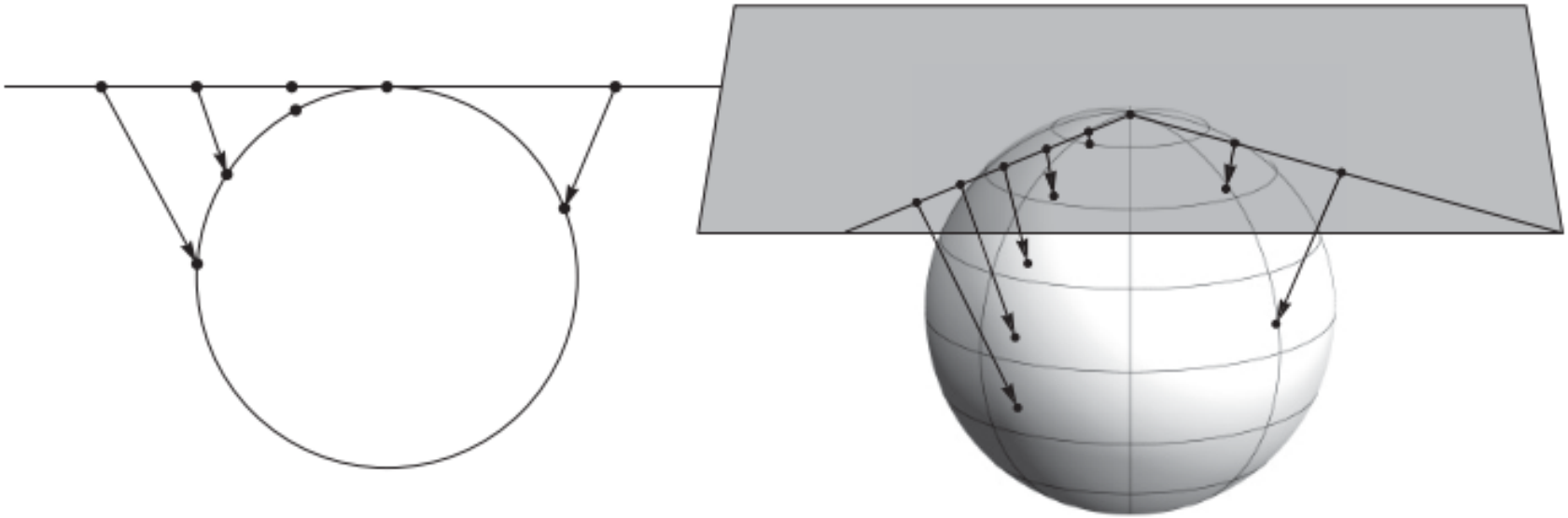




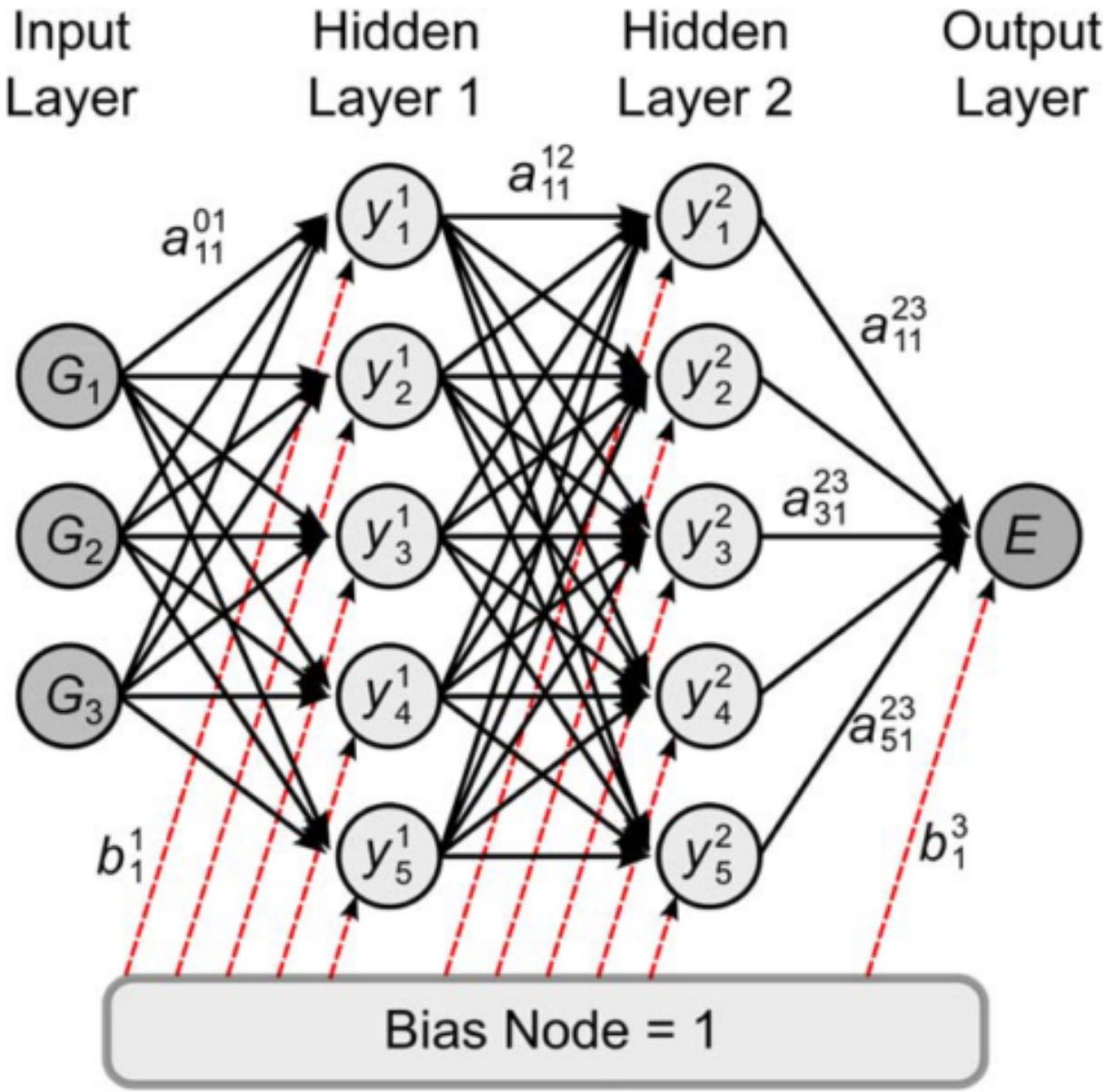
[Qu,Yu, Bowman, ARPC, 2018]

Gaussian Processes

- aka kriging
- Non-parametric way of fitting
- Usually more accurate but requires matrix inversion.
- Both computation and evaluation scales badly
- Provides uncertainty in the estimate (nobody uses it though)
- GAP (Gaussian approximation potentials) [Bartok, Csanyi, 2010-15]: emphasis on input descriptor choice - uses bispectrum



Neural network potentials



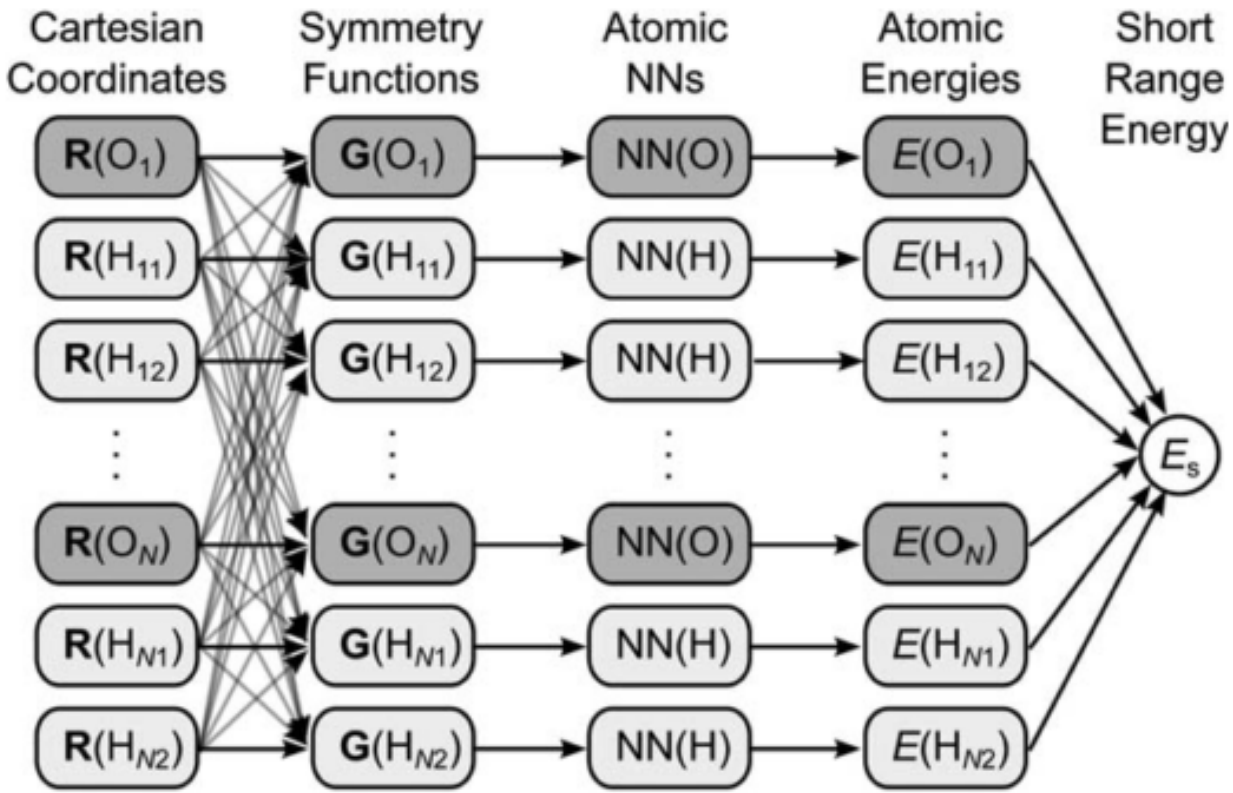
Neural network potentials

- Pros
  - High accuracy
  - Much cheaper than electronic structure calculations
  - Black-box: no-knowledge of the functional form of PES
  - No system-specific modifications required
  - Forces often computable by chain rule
- Cons
  - Slower than parameterized, classical force fields
  - Extrapolation questionable
  - Overfitting happens
  - Default implementation does not allow incremental addition of atoms
  - Training difficult
  - Low-to-moderate dimensions

High-dimensional NN

[Behler, 2010-]

- Map Cartesian coordinates to symmetry descriptors
- Write the total energy as a sum over all atoms $E_S = \sum_{i=1}^{N} E_i$
- Separate NN architecture for each atom



Software

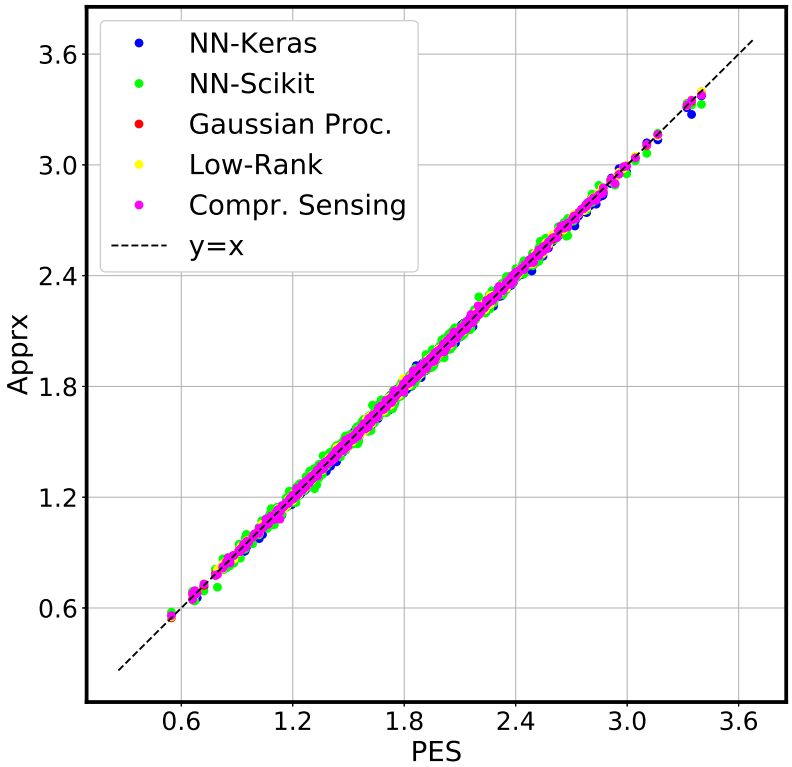
- **ænet** The Atomic Energy Network; Artrith et. al
  `http://ann.atomistic.net/`; Fortran/C/(some) Python;
  Neural Network
- **MLatom** A Package for Atomistic Simulations with Machine Learning; Dral et. al
  `http://mlatom.com/`; Pre-compiled binaries on Linux;
  Kernel Ridge Regression
- **Amp** Atomistic Machine-learning Package; Andrew Peterson and Alireza Khorshidi
  `https://amp.readthedocs.io/`; Python, some Fortran, ASE;
  Neural Network
- **MLIP** Machine Learning of Interatomic Potentials; Shapeev et. al
  `http://gitlab.skoltech.ru/shapeev/mlip/`; C++, LAMMPS integration;
  Moment tensor potential (MTP), active learning
- **MSA** Monomial Symmetrization for PES Fitting; Bowman et. al
  `https://scholarblogs.emory.edu/bowman/msa/`; Fortran/C++/Python;
  Permutationally invariant polynomials
- **DeePMD-kit** Deep learning package for many-body PE and MD; Wang and E
  `https://github.com/deepmodeling/deepmd-kit`; Python/C++/Tensorflow;
  Neural Network
- **GAP** Gaussian Approximation Potentials; Csanyi and Bartok
  `http://www.libatoms.org/`; Fortran;
  Gaussian processes

Our ML Arsenal

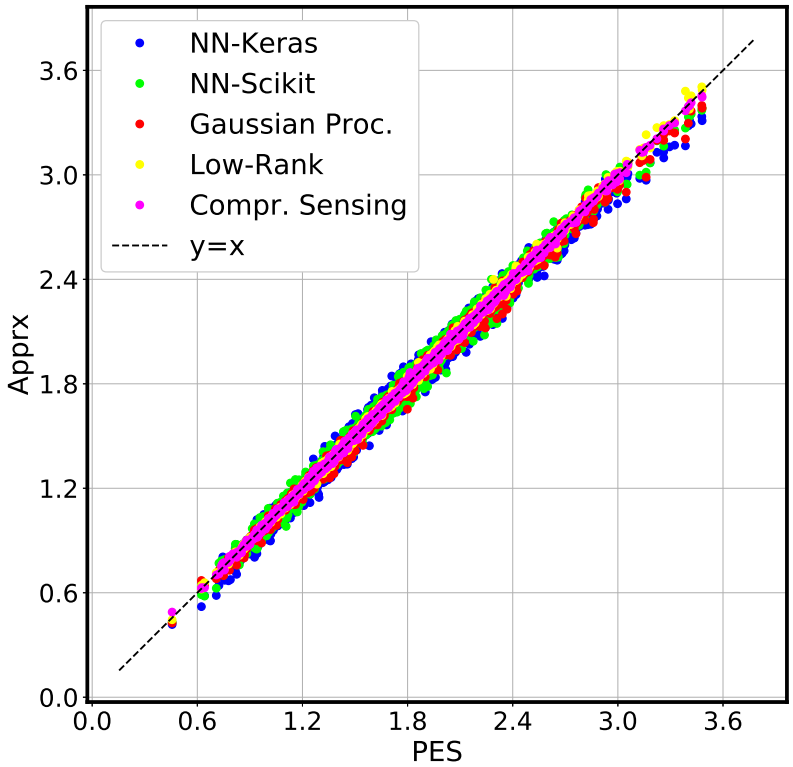
- Sparse Regression $f(x) = \sum_{k=0}^{K} c_k \Psi_k(x)$
  - Polynomial basis (UQTk, www.sandia.gov/uqtoolkit)
  - Radial basis functions (UQTk)
- Low-Rank Tensor Expansion $f(x) = \sum_{r=1}^{R} c_r \prod_{j=1}^{d} \Psi_{rj}(x_j)$
  - Canonical format (UQTk)
  - Tensor-train format - current (CCC, https://github.com/goroda/Compressed-Continuous-Computation)
- Gaussian processes $P(f|\mathcal{D})$
  - Hierarchical correction (UQTk)
  - Flexible kernels (Scikit-learn)
- Neural Networks $f(x) = ...W_3\sigma(W_2\sigma(W_1x + b_1) + b_2) + b_3$
  - Scikit-learn
  - Keras
  - PyTorch - current

Quick and dirty example: CH_2CHO

- 800 training points (PES evaluations)

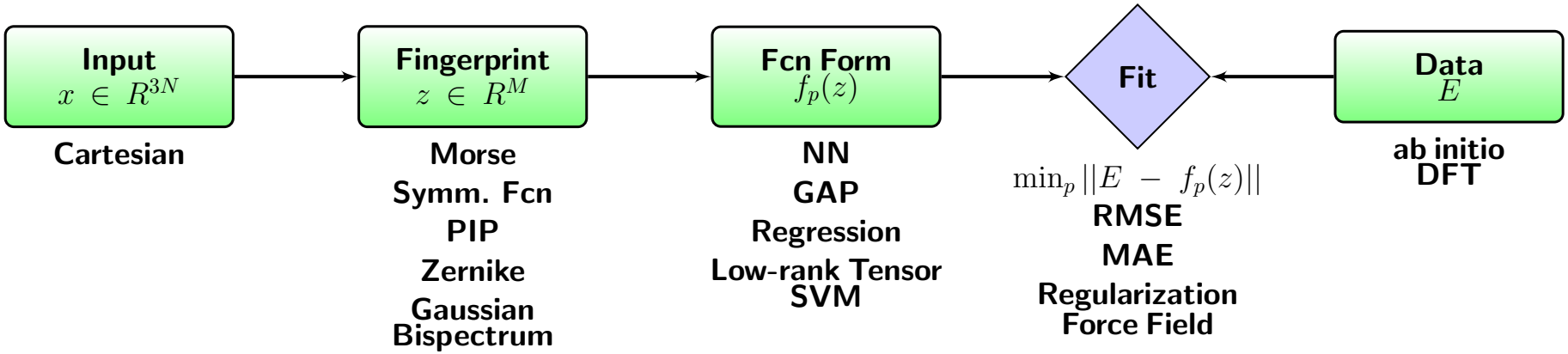


Training

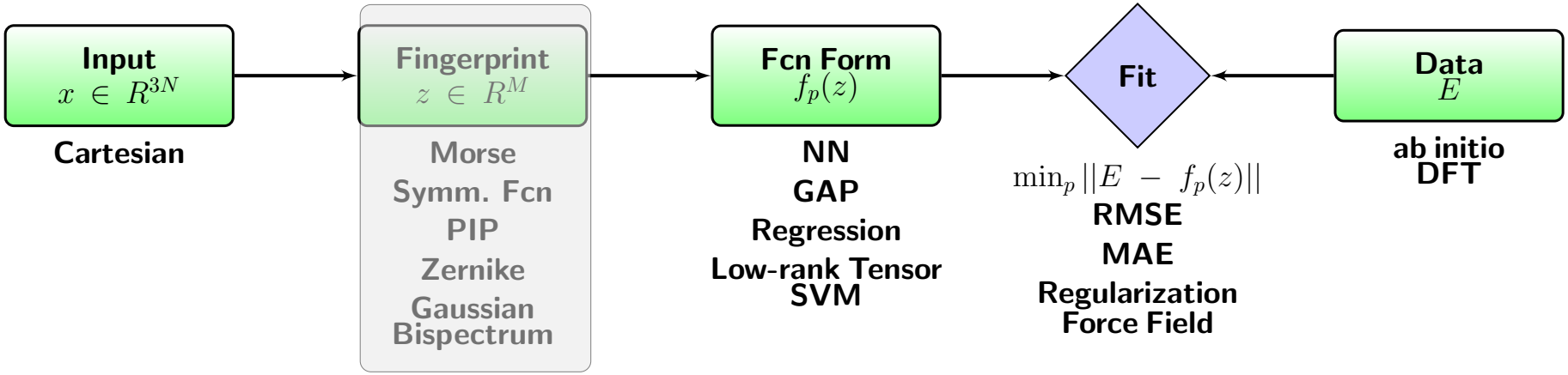


Test

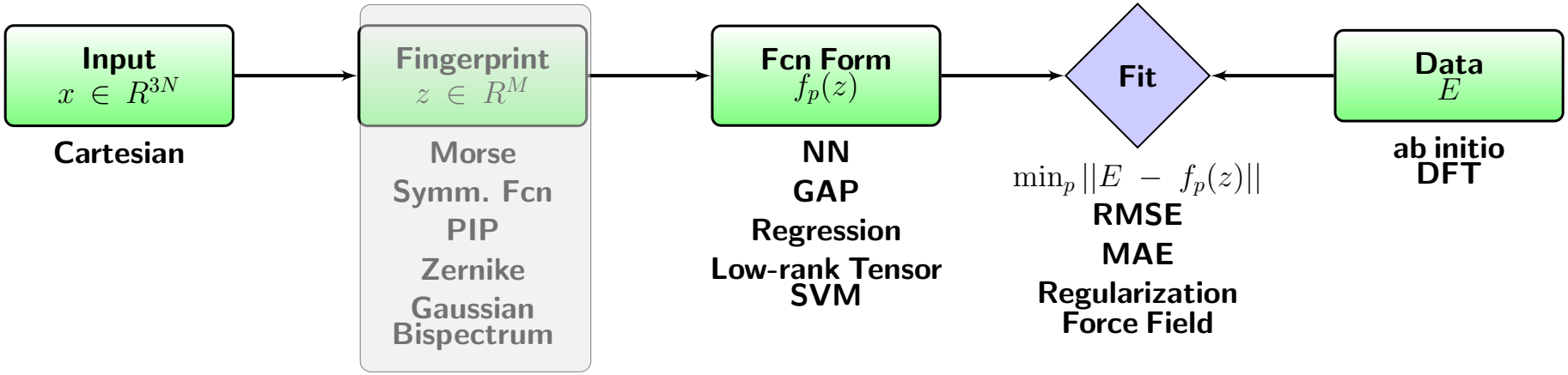
Plans and Big Picture



Plans and Big Picture

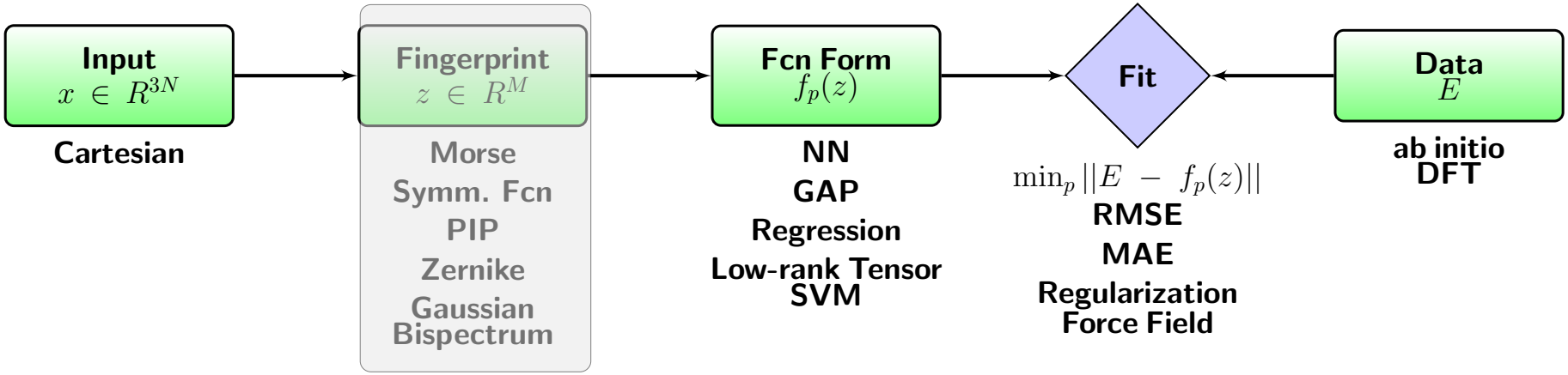


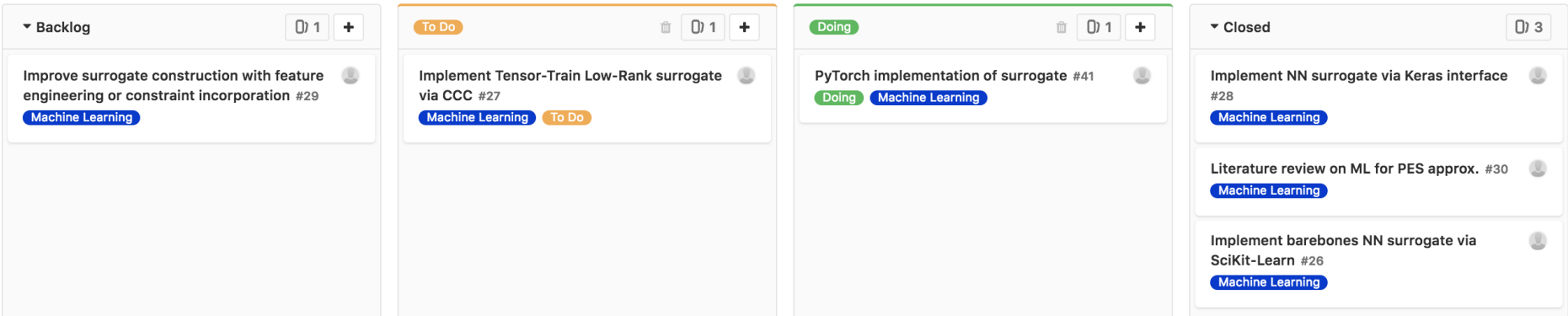
Plans and Big Picture



- Build modular library for 'supervised ML'
- Automate the fitting process
- Apply across multiple classes of structures
- Adaptive sampling of input configurations
- Leverage tools from UQTk (www.sandia.gov/UQToolkit) and from existing codes
- for NN: evaluate the GPU needs
- Connect (asynchronously) to Sella

Plans and Big Picture





Literature

- J. Behler, “Constructing High-Dimensional Neural Network Potentials: A Tutorial Review”, *International Journal of Quantum Chemistry*, 115(16), 2015.
- Z. Xie, J. Bowman, “Permutationally Invariant Polynomial Basis for Molecular Energy Surface Fitting via Monomial Symmetrization”, *Journal of Chemical Theory and Computation*, 6, 2010.
- A. Bartok, G. Csànyi, “Gaussian Approximation Potentials: A Brief Tutorial Introduction”, *International Journal of Quantum Chemistry*, 115, 2015.
- A. Khorshidi, A. Peterson, “Amp: A modular approach to machine learning in atomistic simulations”, *Computer Physics Communications*, 207, 2016.

- P. Rai, K. Sargsyan, H. Najm, M. Hermes, S. Hirata, “Low-rank canonical-tensor decomposition of potential energy surfaces: Application to grid-based diagrammatic vibrational Green’s function theory”, *Molecular Physics*, 115(17-18), 2017.
- P. Rai, K. Sargsyan, H. Najm, “Compressed sparse tensor based quadrature for vibrational quantum mechanics integrals”, *Computer Methods in Applied Mechanics and Engineering*, 336, 2018.
- K. Sargsyan, C. Safta, H. Najm, B. Debusschere, D. Ricciuto, P. Thornton, “Dimensionality reduction for complex models via Bayesian compressive sensing”, *International Journal for Uncertainty Quantification*, 4(1), 2014.