

Active Learning and Uncertainty Quantification for Machine Learning Interatomic Potentials

Khachik Sargsyan,
Katherine Johnston, Varuni Dantanarayana,
Habib N. Najm

Sandia National Laboratories, Livermore, CA

MMLDT-CSET
(virtual) San Diego
Sep 28, 2021



Sandia National Laboratories

Acknowledgements

- Aidan Thompson, Mary Alice Cusentino, Yoona Yang (Sandia)
- DOE, Office of Science,
 - Fusion Energy Sciences (FES)
 - Advanced Scientific Computing Research (ASCR)
 - Basic Energy Sciences (BES)



Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA-0003525.

- Motivation: potential energy surface approximation
 - Machine learning for interatomic potentials (MLIAP)
- Bayesian estimation of MLIAPs
 - Demo on Spectral Neighbor Analysis Potential (SNAP)
 - Importance of noise model, model error estimation
- Active learning (AL) strategies for ML
 - UQ helps AL
 - Query-by-committee
 - Preliminary results

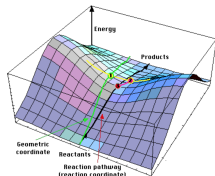
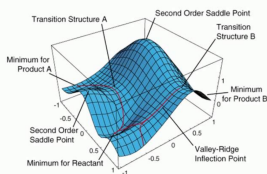
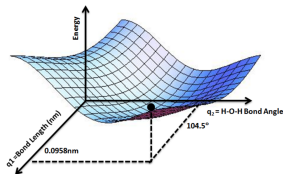
Target: Potential energy surface (PES) approximation

$$E = f(x)$$

x represents coordinates/descriptors

E is energy

- Accurate and fast surrogates for PES to replace quantum mechanical 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



ML Interatomic Potentials (MLIAP)

- Partition the interatomic interaction energy into individual contributions of the atoms
 - $E_{\text{total}} = \sum_{i=1}^N E_i$
- Assume flexible functional forms of each such contribution
 - Function of positions of the neighboring atoms
 - $O(100)$ parameters
- Require the energy, forces and/or stresses predicted by a MLIAP to be close to those obtained by a quantum mechanical model on some atomic configurations (a.k.a. training set)
 - This is supervised machine learning

MLIAP - desired features

- Good input descriptors
- Accurate, fast-to-evaluate, analytic derivatives
- High-dimensional, flexible functional form
- Transferable/generalizable to unseen atomic configurations
- Account for physics:
 - invariant with respect to translation, rotation, and reflection of the space, and also permutation of chemically equivalent atoms
- Locality (depend on surrounding atoms only within a finite cut-off radius), but remain smooth with respect to atoms entering and leaving the local neighborhood
- **Equipped with uncertainty estimate**

ML for PES, (growing) literature:

uncertainty estimation is largely lacking

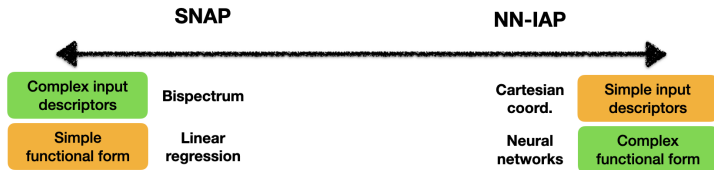
- Weighted interpolation [Ischtwan 1994; Doves, 2007-09; Maisuradze, 2009]
- Permutationally invariant polynomials [Xie, 2010]
- Gaussian processes [Bartok, Csanyi 2010-15; Mills, 2012; Rupp, 2013; Cui, 2016; Uteva, 2017; Guan, 2018; Schmitz, 2018]
- Low-rank tensor expansions [Jackle, 1996; Baranov, 2015; Rai, 2017, 2018]
- Support vector machines, kernel regression [Le, 2009; Balabin, 2011; Dral, 2017]
- 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]

Uncertainty estimation options

$$y \approx f_w(x)$$

- Bayesian inference: $\overbrace{P(w|y)}^{\text{Posterior}} \propto \overbrace{P(y|w)}^{\text{Likelihood}} \overbrace{P(w)}^{\text{Prior}}$
 - Markov chain Monte Carlo sampling of posterior PDF
- Variational methods: $w \sim N(\mu, \Sigma)$ and optimize μ, Σ .
 - Largely, this is also called Bayesian Neural Networks (BNN)
 - Stochastic gradient descent to minimize evidence lower bound (ELBO)
- Ensemble methods: many flavors.
 - Deep ensembles
 - Query-by-committee
 - Boosting/bagging

Range of options for IAPs



Focus on SNAP (Spectral Neighbor Analysis Potential)

- [Thompson et al., 2015] “Spectral neighbor analysis method for automated generation of quantum-accurate interatomic potentials”, *J Comp Phys*, 2015.

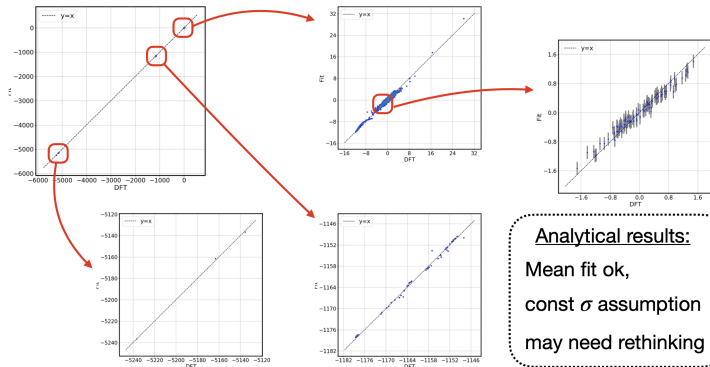
$$f(q) = \sum_{k=0}^K w_k B_k(q)$$

- Linear expansion in parameters w .
- Bayesian inference: both MCMC and analytical posterior PDFs are feasible

SNAP uncertainty with Tantalum data set

$$f(q) = \sum_{k=0}^K w_k B_k(q)$$

- Employed FitSNAP <https://github.com/FitSNAP/FitSNAP>
- Exact analytical Bayesian answer:
 $w \sim \mathcal{N}((B^T B)^{-1} B^T y, \sigma^2 (B^T B)^{-1})$
 - ... if Gaussian i.i.d. likelihood is used



- assumptions baked in likelihood form are crucial

Bayesian likelihood (aka data model) should be chosen with model error in mind

- Overall workflow below needs to be augmented with UQ



- The ‘classical’ Gaussian i.i.d. data model $y_i = f_w(q_i) + \sigma\epsilon_i$ does not work well for heterogeneous training sets
- Model error embedding [\[Sargsyan et. al., 2019\]](#) will be applied to augment MLIAP coefficients with uncertainties, leading to physics-based likelihoods via data model

$$y_i = f_{w+\sigma_w\xi}(q_i) + \sigma\epsilon_i$$

- For linear models, such as SNAP, this is equivalent to more flexible noise assumptions. Current work.

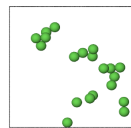
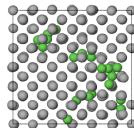
Training set selection is crucial

- Configurations chosen for training data influence results
- Example: W-H (tungsten/hydrogen) IAPs
- Initial IAPs resulted in hydrogen clusters in bulk tungsten, which should not occur
- Additional training data was generated and put into the training set
- Including these specific configurations prevented unphysical hydrogen clustering

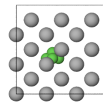
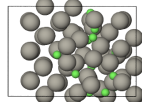
Grey: Tungsten

Green: Hydrogen

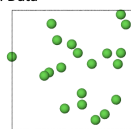
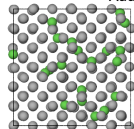
Initial Poor Hydrogen Clustering Behavior



Generated New Training Data Based on Poor Initial Performance

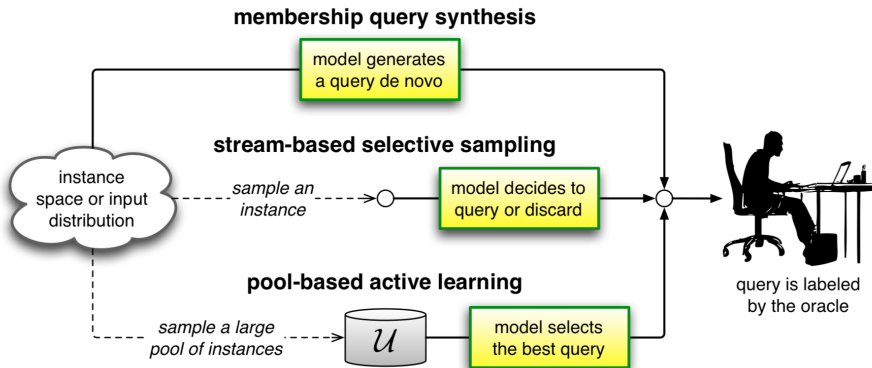


Improved Clustering Behavior with Additional Data



Results from Mary Alice Cusentino (SNL), using LAMMPS software.

Active Learning: selection of training configurations

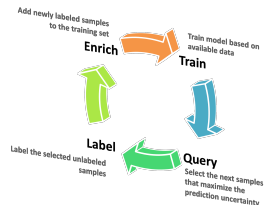


[B. Settles, "Active learning literature survey", Computer Sciences Technical Report 1648, University of Wisconsin-Madison, 2009]

Active Learning: selection of training configurations

Greater test accuracy with fewer training samples

- Two flavors of the challenge:
 - Interpolation: developing a reliable problem-specific MLIAP that would accurately interpolates within the training domain is nontrivial
 - Extrapolation: prediction outside the training domain is even harder
- Key: *query strategy*, whether to query high-fidelity quantum mechanical (QM) simulation or not.
 - If such decision can be made reliably, then one does not need to start with a very good training set



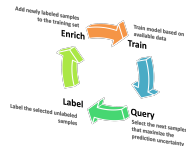
Query Strategies:

almost all rely on some form of uncertainty estimate

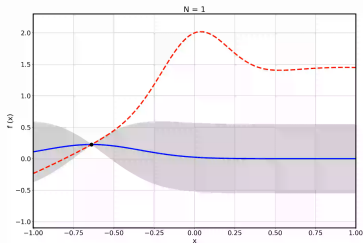
- **Uncertainty sampling:** an active learner queries the instances about which it is least certain how to label.
- **Query-by-committee:** committee of competing models, and pick a query about which they most disagree. Need a measure of disagreement.
- **Expected model change:** which query would lead to greatest model change, e.g. largest gradient length.
- **Variance Reduction and Fisher Information Ratio:** minimizing the variance component of generalization error estimate (via Fisher Information)
- **Estimated error reduction:** Estimate the expected future error that would result if some new instance x is labeled and added to training set, and then select the instance that minimizes that expectation.

Demonstration of AL

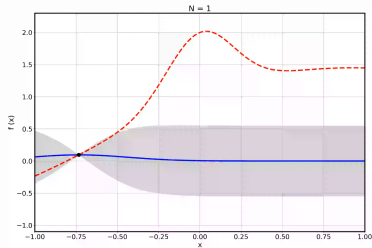
- Selecting one point at a time given the current uncertainty estimate



Naïve approach



Active approach

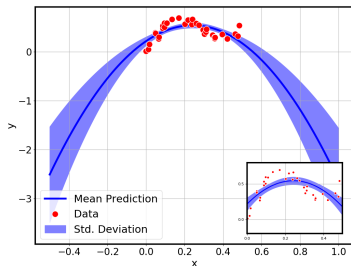


- Next: how to reliably estimate uncertainty?

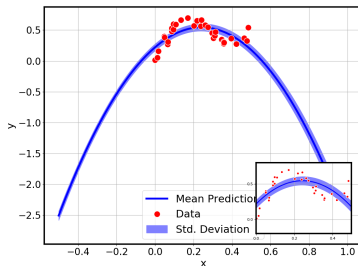
Polynomial fit: Extrapolation scenario

Order=2

True Posterior



Variational Posterior

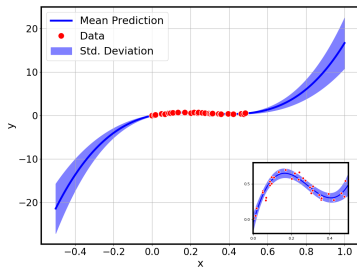


Variational posterior predictions heavily underestimate both interpolative and extrapolative errors.

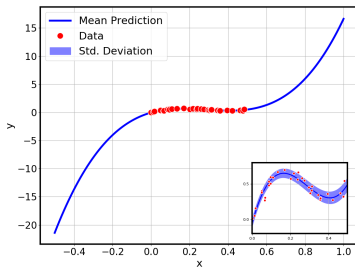
Polynomial fit: Extrapolation scenario

Order=3

True Posterior



Variational Posterior

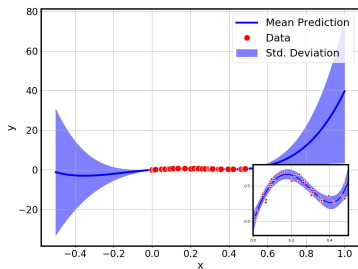


Variational posterior predictions heavily underestimate both interpolative and extrapolative errors.

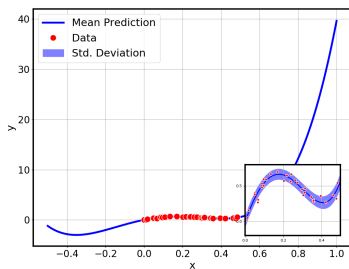
Polynomial fit: Extrapolation scenario

Order=4

True Posterior



Variational Posterior

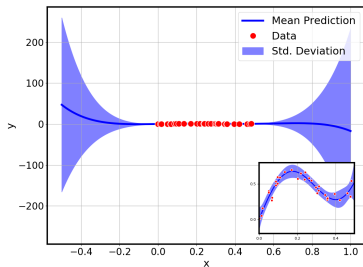


Variational posterior predictions heavily underestimate both interpolative and extrapolative errors.

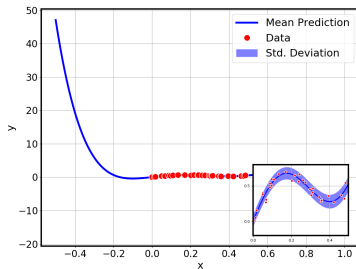
Polynomial fit: Extrapolation scenario

Order=5

True Posterior



Variational Posterior

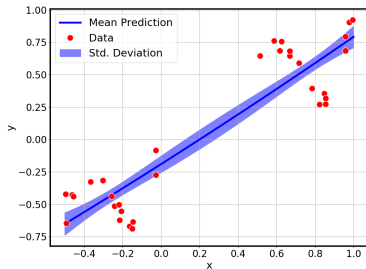


Variational posterior predictions heavily underestimate both interpolative and extrapolative errors.

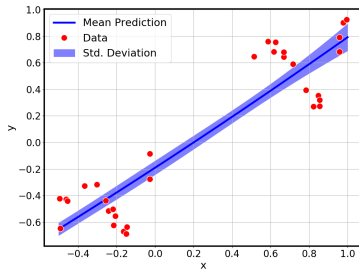
Polynomial fit: Interpolation scenario

Order=2

True Posterior



Variational Posterior

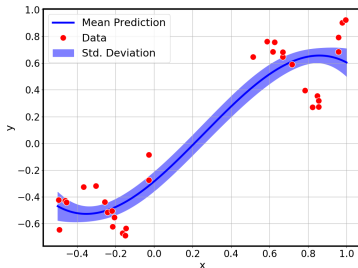


Variational posterior predictions heavily underestimate both interpolative and extrapolative errors.

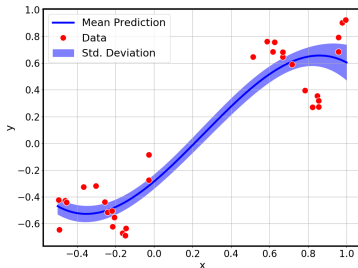
Polynomial fit: Interpolation scenario

Order=3

True Posterior



Variational Posterior

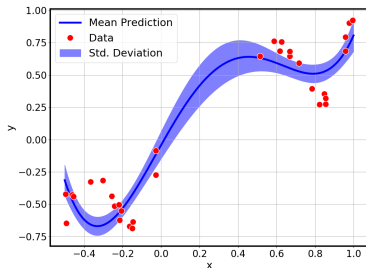


Variational posterior predictions heavily underestimate both interpolative and extrapolative errors.

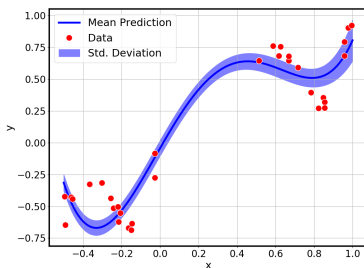
Polynomial fit: Interpolation scenario

Order=4

True Posterior



Variational Posterior

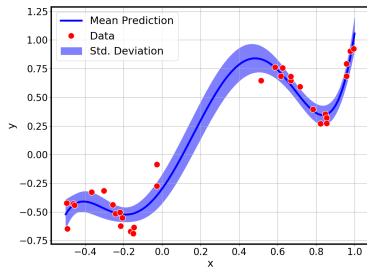


Variational posterior predictions heavily underestimate both interpolative and extrapolative errors.

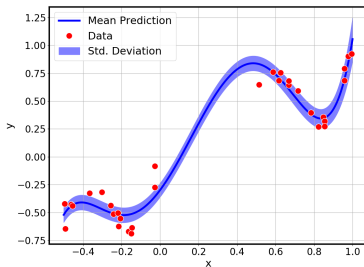
Polynomial fit: Interpolation scenario

Order=5

True Posterior



Variational Posterior

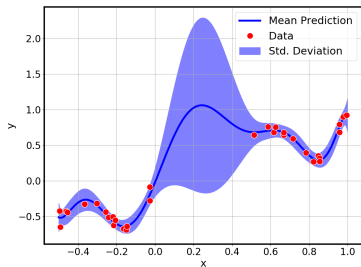


Variational posterior predictions heavily underestimate both interpolative and extrapolative errors.

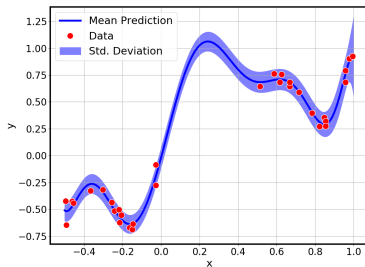
Polynomial fit: Interpolation scenario

Order=10

True Posterior



Variational Posterior

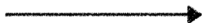


Variational posterior predictions heavily underestimate both interpolative and extrapolative errors.

Uncertainty-enabling wrappers over PyTorch modules

Deterministic

`torch.nn.module`



Probabilistic

`wrapper(torch.nn.module)`

Option 1: ensemble NN

```
nn_ens = EnsRegr(torch.nn.module, nens=111)
```

```
class EnsRegr():
    def __init__(self, nnmodule, nens=1, verbose=False):
        self.nnmodule = nnmodule
        self.verbose = verbose
        self.nens = nens
```

Option 2: NN learning with MCMC

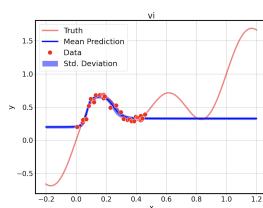
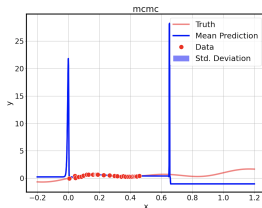
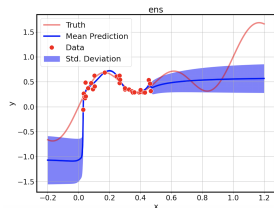
```
nn_mcmc = MCMCRegr(torch.nn.module)
```

```
class MCMCRegr():
    def __init__(self, nnmodule, verbose=True):
        self.nnmodule = nnmodule
        self.verbose = verbose
```

Option 3: NN learning with VI

```
nn_vi = VIRegr(torch.nn.module)
```

```
class VIRegr():
    def __init__(self, nnmodule, verbose=False):
        self.bmodel = BNet(nnmodule)
        self.verbose = verbose
```

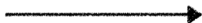


- MCMC struggles with complex NNs; VI underestimates; Ensembles do well

Uncertainty-enabling wrappers over PyTorch modules

Deterministic

`torch.nn.module`



Probabilistic

`wrapper(torch.nn.module)`

Option 1: ensemble NN

```
nn_ens = EnsRegr(torch.nn.module, nens=111)
```

```
class EnsRegr():
    def __init__(self, nnmodule, nens=1, verbose=False):
        self.nnmodel = nnmodule
        self.verbose = verbose
        self.nens = nens
```

Option 2: NN learning with MCMC

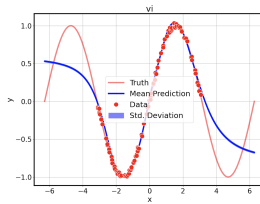
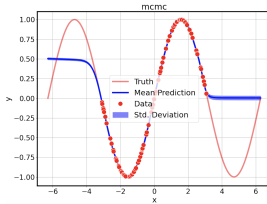
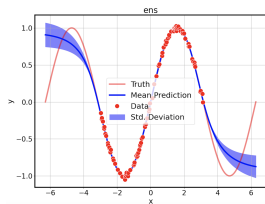
```
nn_mcmc = MCMCRegr(torch.nn.module)
```

```
class MCMCRegr():
    def __init__(self, nnmodule, verbose=True):
        self.nnmodule = nnmodule
        self.verbose = verbose
```

Option 3: NN learning with VI

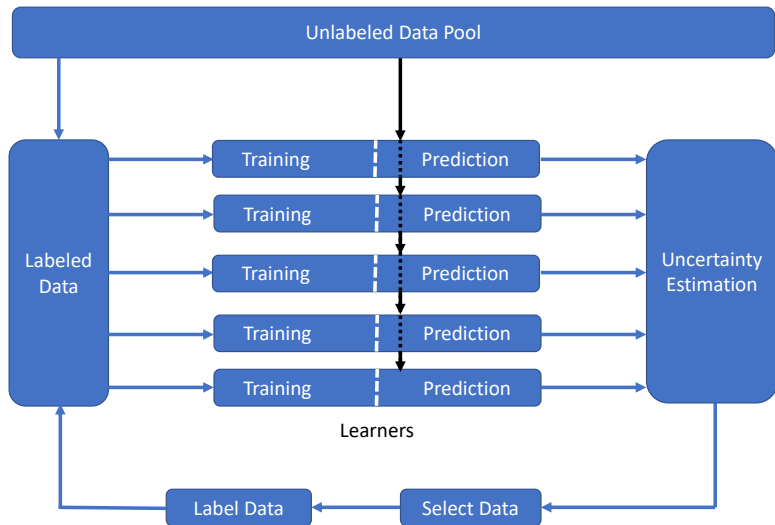
```
nn_vi = VIRegr(torch.nn.module)
```

```
class VIRegr():
    def __init__(self, nnmodule, verbose=False):
        self.bmodel = BNet(nnmodule)
        self.verbose = verbose
```



- MCMC struggles with complex NNs; VI underestimates; Ensembles do well

Query-by-Committee (QBC): algorithm sketch

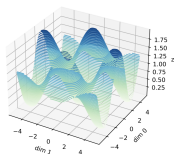


Query-by-Committee (QBC): algorithm outline

- Start with a large pool of P unlabeled points
- Select a training set of N points from the pool
- Launch K learners, each with fN randomly-chosen training points
 - Random sampling with replacement
 - Selection of fraction f determines data size per learner
 - diversity vs data size tradeoff
- Evaluate the learners' performance at all points in the pool
- Select M points from the pool, having highest 'disagreement', & add them to the training set
 - M choice, size of batch added per query, low error vs optimal choice
 - K -means clustering to discover geometry of selected data
 - Distribute data from clusters evenly among learners
 - Add fM points per learner with replacement
- Re-train, and repeat query to evaluate learners performance on prediction of unlabeled data in pool

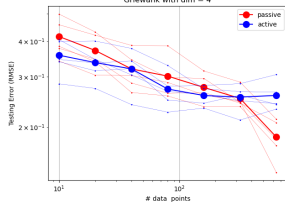
QBC: Griewank test function

Griewank: dim = 2

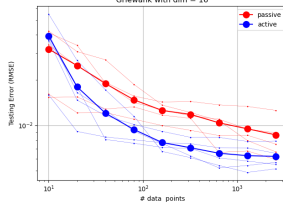


$$f(\mathbf{x}) = 1 + \sum_{i=1}^d \frac{x_i^2}{4000} - \prod_{i=1}^d \cos\left(\frac{x_i}{\sqrt{i}}\right)$$

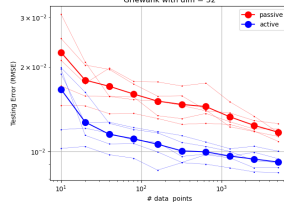
Griewank with dim = 4



Griewank with dim = 16



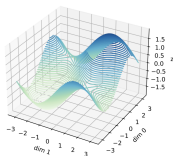
Griewank with dim = 32



- Efficiency of active learning improves with higher dimension.

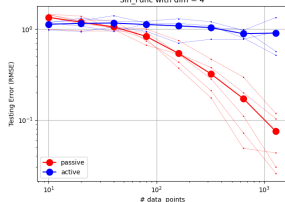
QBC: Sine test function

Sin_Func: dim = 2

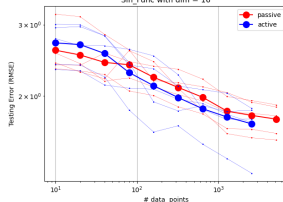


$$f(\mathbf{x}) = \sin\left(\sum_{i=1}^d x_i\right)$$

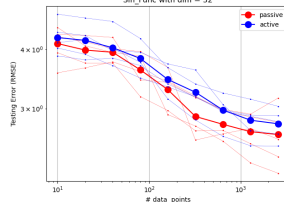
Sin_Func with dim = 4



Sin_Func with dim = 16



Sin_Func with dim = 32



- In low-d, large pool size causes newly selected points to cluster.
- Potential solution: sample according to PDF $e^{-std(x)}$ to concentrate new points near high uncertainty region, but select elsewhere, too.

- Potential energy surface (PES) approximation:
 - Machine learning for interatomic potentials is the cornerstone of PES approximation
- Uncertainty estimation
 - Bayesian inference: careful with likelihood assumptions; does not always work
 - Variational methods: underestimate/homogenize the uncertainty
 - Ensemble learning: mostly empirical, but they work!
- Active learning
 - Heavily relies on predictive uncertainty estimates
 - Query-by-committee (QBC) shows promising results on test models

Literature

Model error embedding

- [Sargsyan et al., 2019] “Embedded model error representation for Bayesian model calibration”, *Int. J. Uncertain. Quantif.*, 9(4), 2019.

MLIAPs

- [Thompson et al., 2015] “Spectral neighbor analysis method for automated generation of quantum-accurate interatomic potentials”, *J Comp Phys*, 2015.
- [J. Behler, 2014] “Representing potential energy surfaces by high-dimensional neural network potentials”, *J. Phys.: Condens. Matter*, 26, 2014.

Active learning

- [B. Settles, 2009] “Active learning literature survey”, *Comp Sci Tech Report 1648*, University of Wisconsin-Madison, 2009.

Active learning for MLIAPs

- [E. Podryabinkin, A. Shapeev, 2017] “Active learning of linearly parametrized interatomic potentials”, *Comp Mat Sci*, 140, 2017.
- [J. Vandermause et al., 2020] “On-the-fly active learning of interpretable Bayesian force fields for atomistic rare events”, *npj Computational Materials*, 6, 2020.