A neural network representation of the potential energy surface in Si- and Si-Li systems

Brad Malone, Ekin Cubuk, and Efthimios Kaxiras

HARVARD
UNIVERSITY

The detailed simulation of many physical processes can be perfomed with knowledge of the potential energy surface (PES)

The detailed simulation of many physical processes can be perfomed with knowledge of the potential energy surface (PES)

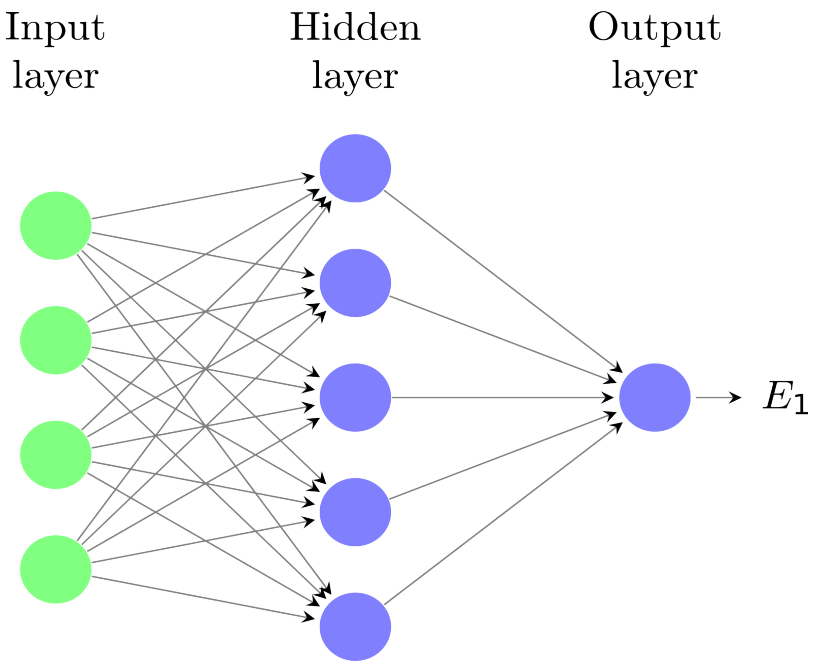
$$\{R_1 \ldots . R_N\} \xrightarrow{SE} E_{TOT}, \{F_1 \ldots . F_N\}$$

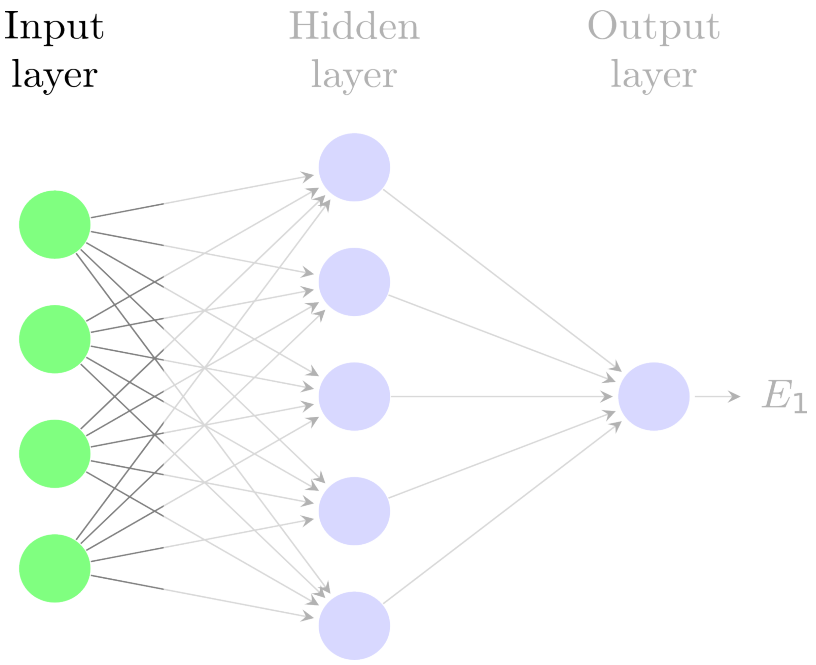
The detailed simulation of many physical processes can be perfomed with knowledge of the potential energy surface (PES)

$$\{R_1 \ldots . R_N\} \xrightarrow{SE} E_{TOT}, \{F_1 \ldots . F_N\}$$

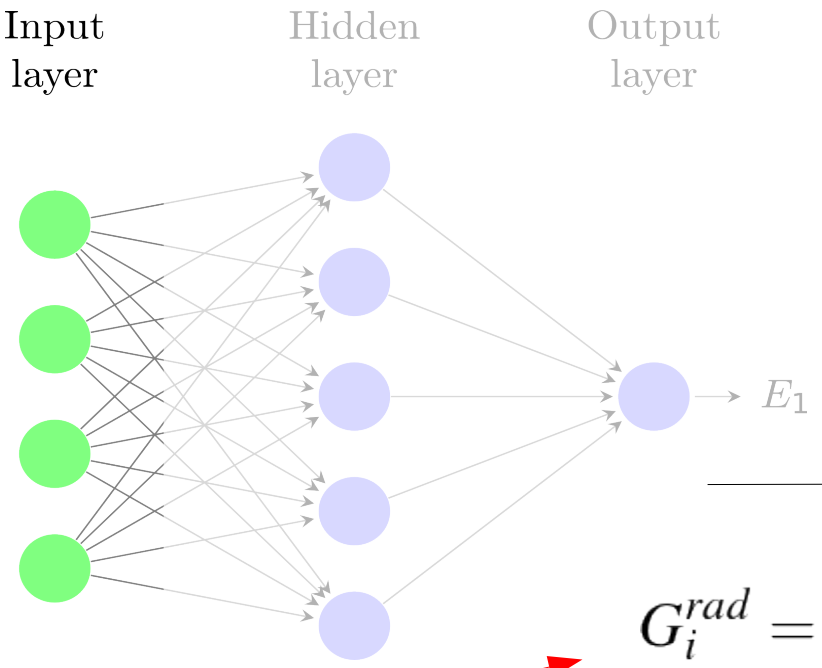
Empirical potentials can be very efficient, although with some loss of accuracy

Artificial neural network





Highly beneficial if inputs have obvious symmetries of the system built-in



Highly beneficial if inputs have obvious symmetries of the system built-in

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

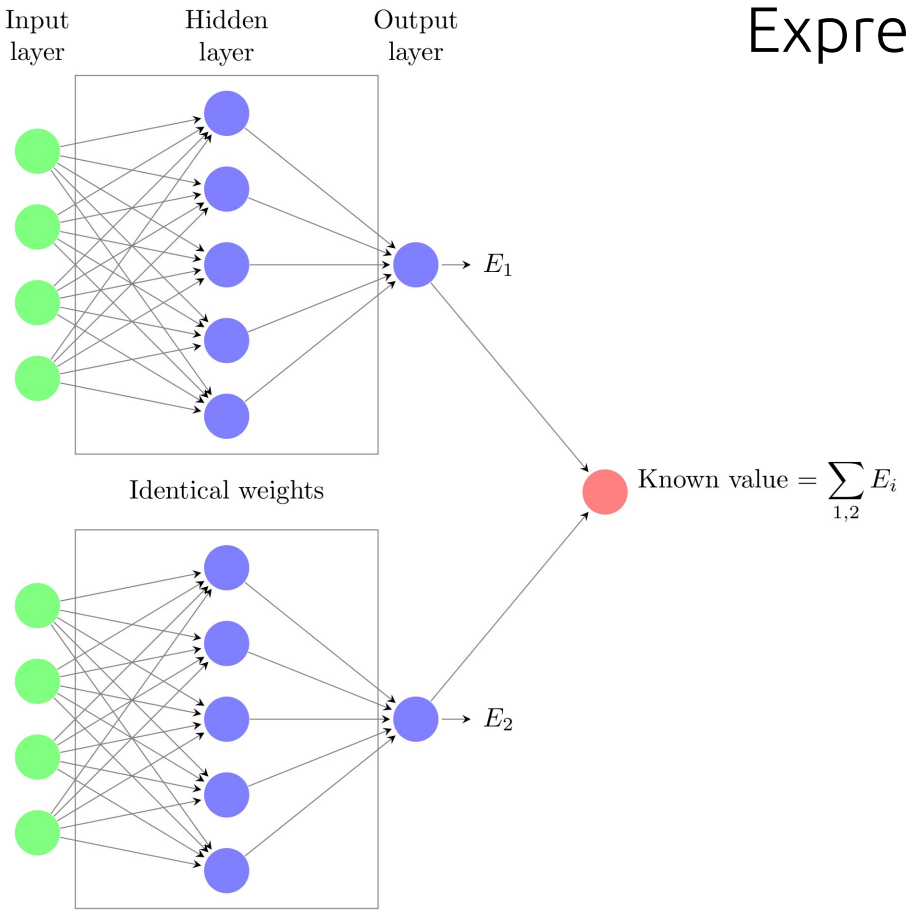
Radial functions

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

Angular functions

Cutoff functions

Behler, J. Chem. Phys. **134**,074106 (2011)



Express total energy as a sum of atomic contributions

$$E^{tot} = \sum_i E_i$$

Behler, Parrinello PRL **98**,146401 (2007)

Networks of this type have been successful, but...

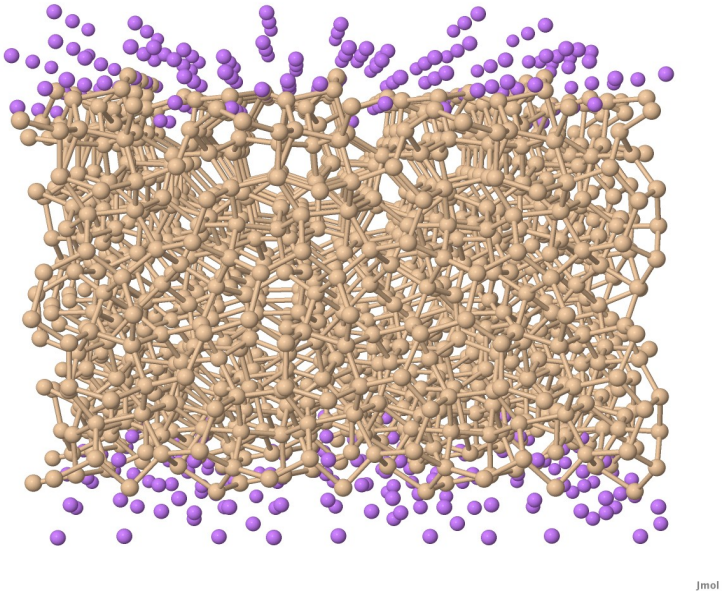
1). Currently limited to about 4 atomic types

2). Number and type of symmetry functions is ad-hoc

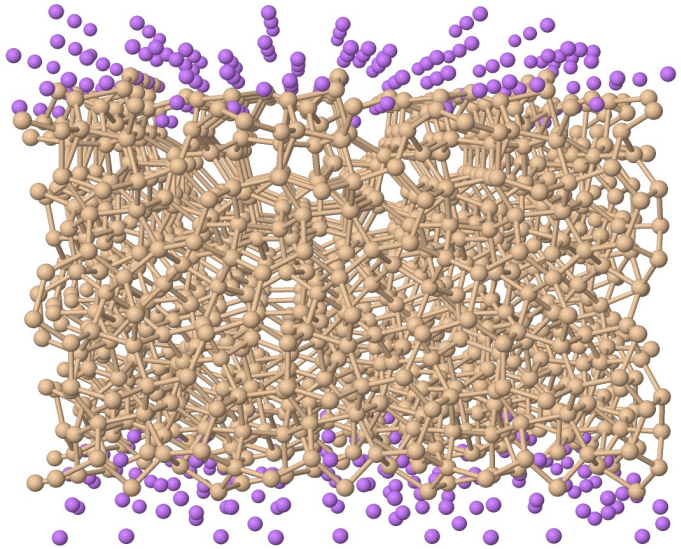
3). Extrapolation on unseen data can give large errors

4). Relationships between accuracy, data sizes, NN architectures, input space choices largely unexplored

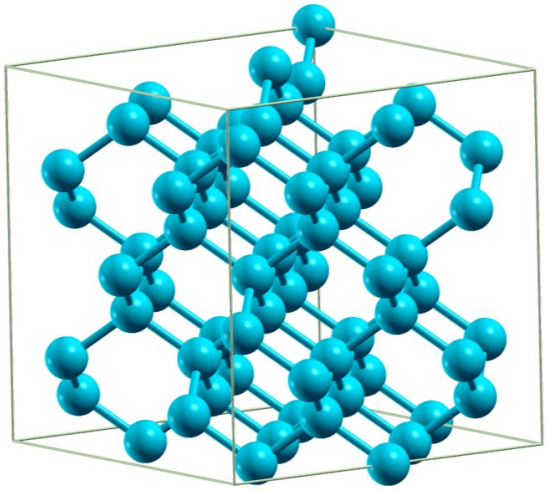
- Si-Li systems for battery applications



- Si-Li systems for battery applications



- Pure Si a good benchmark system

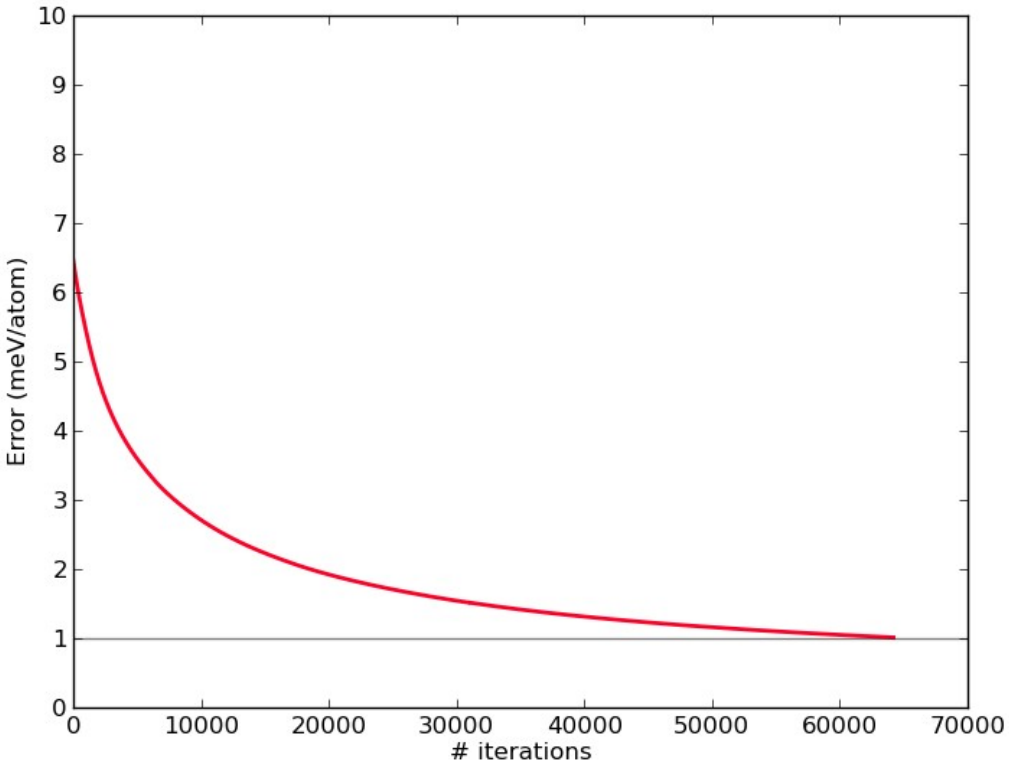


Does this work?

Molten Si system – 64 atom supercell @ 2000K

Does this work?

Molten Si system – 64 atom supercell @ 2000K



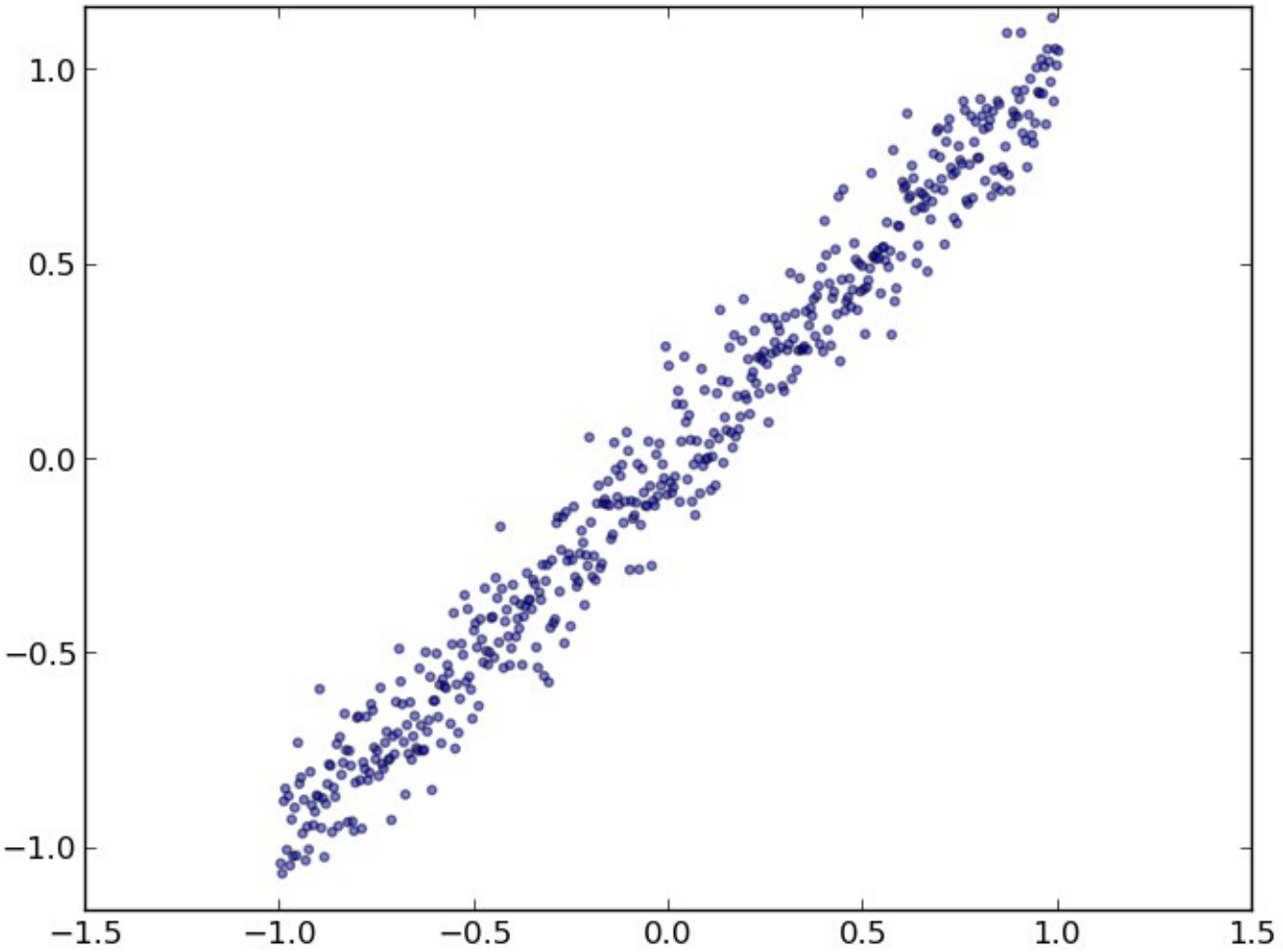
Training error =1.01 meV/atom

Test error = 1.1 meV/atom

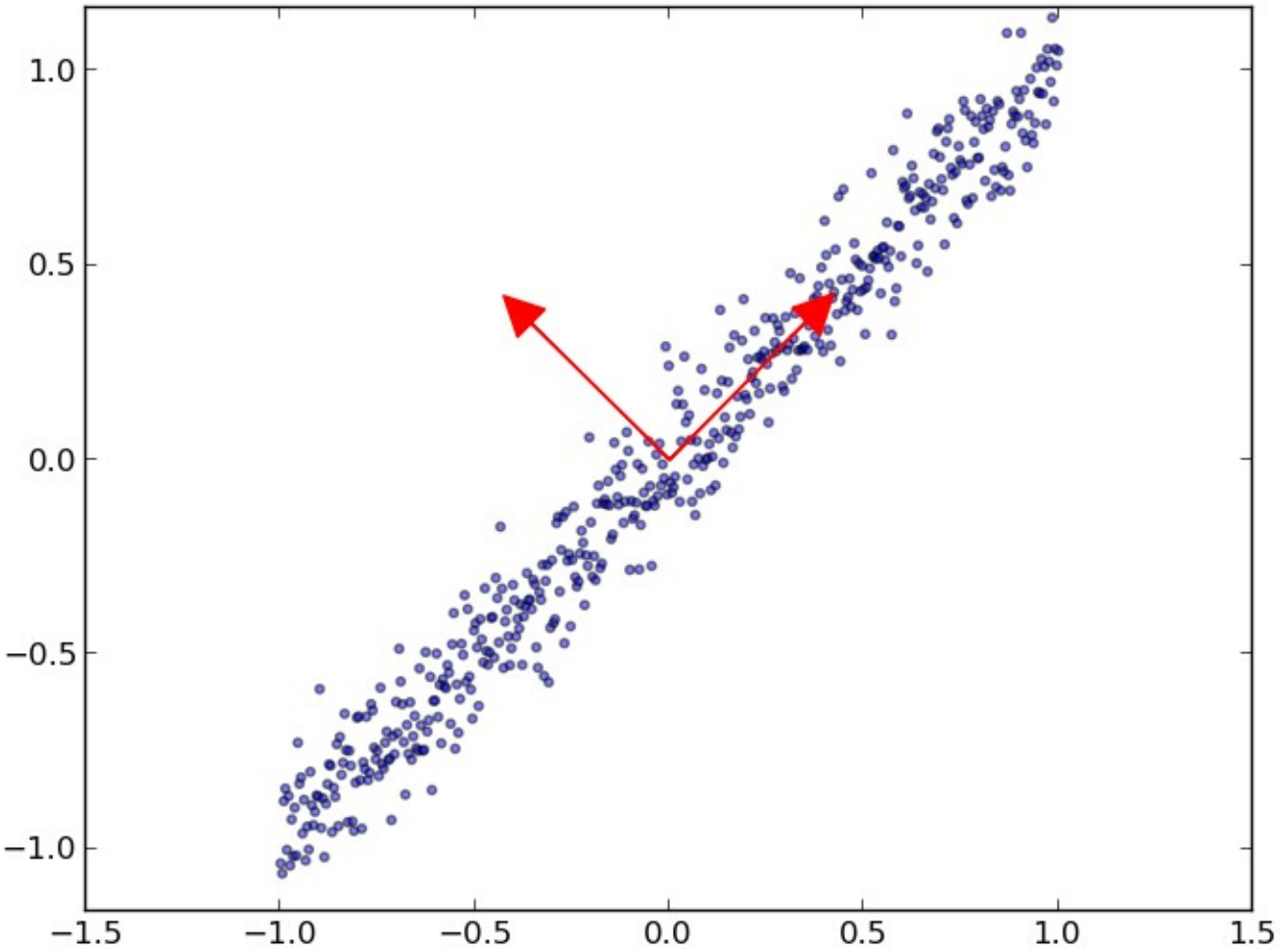
Neural net training can take a long time

Neural net training can take a long time

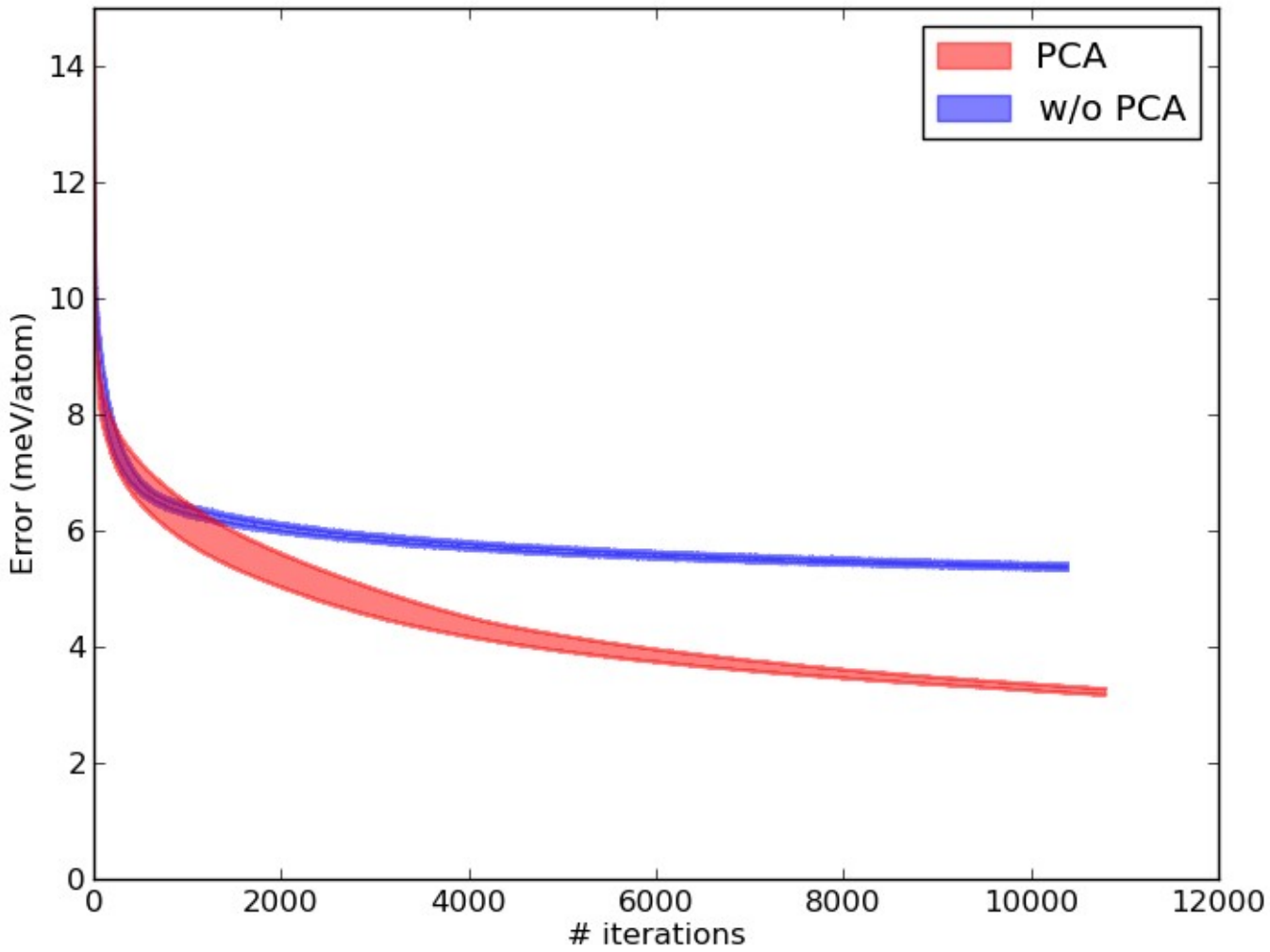
Use Principal Component Analysis (PCA)
on input space?

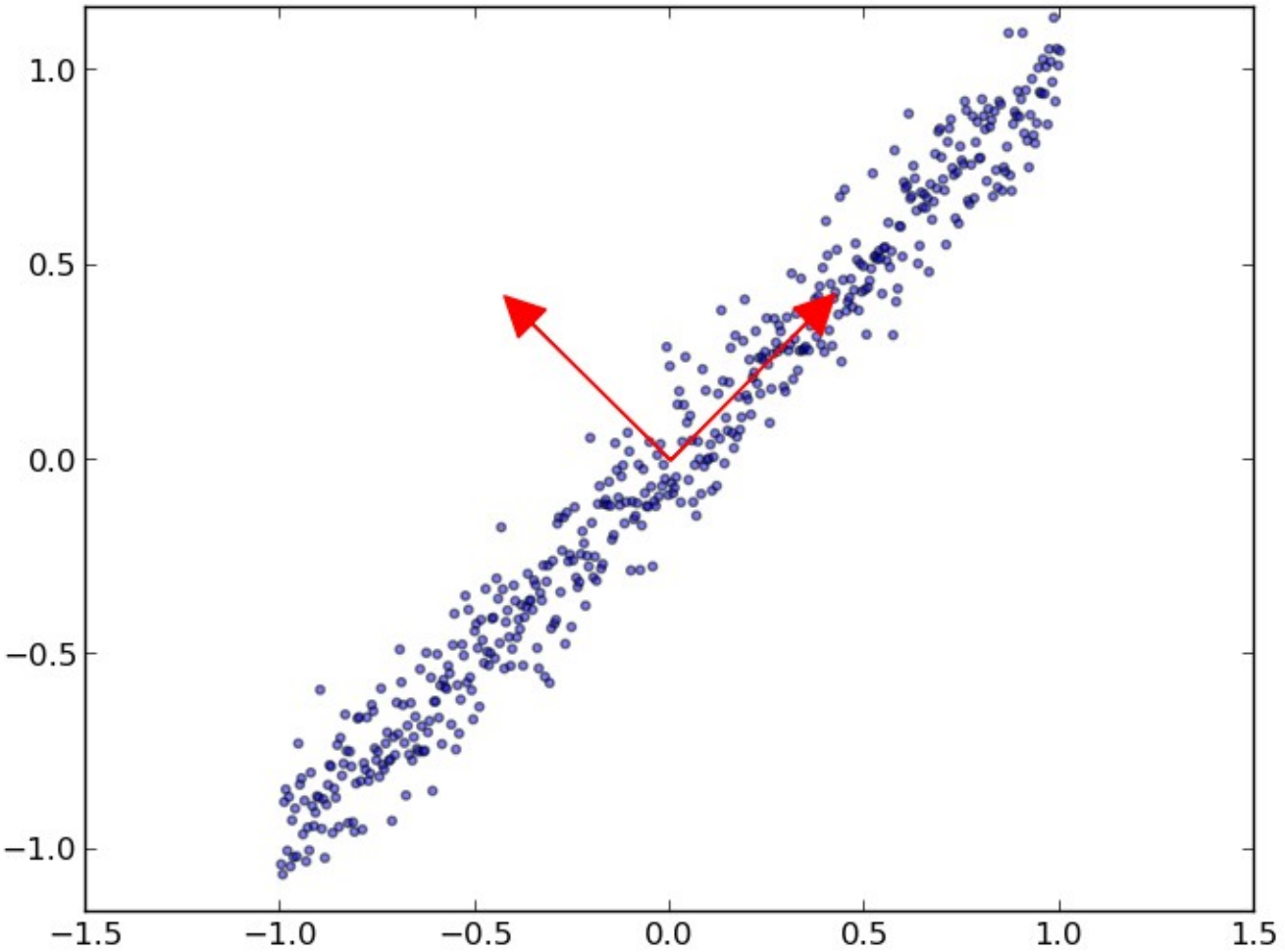


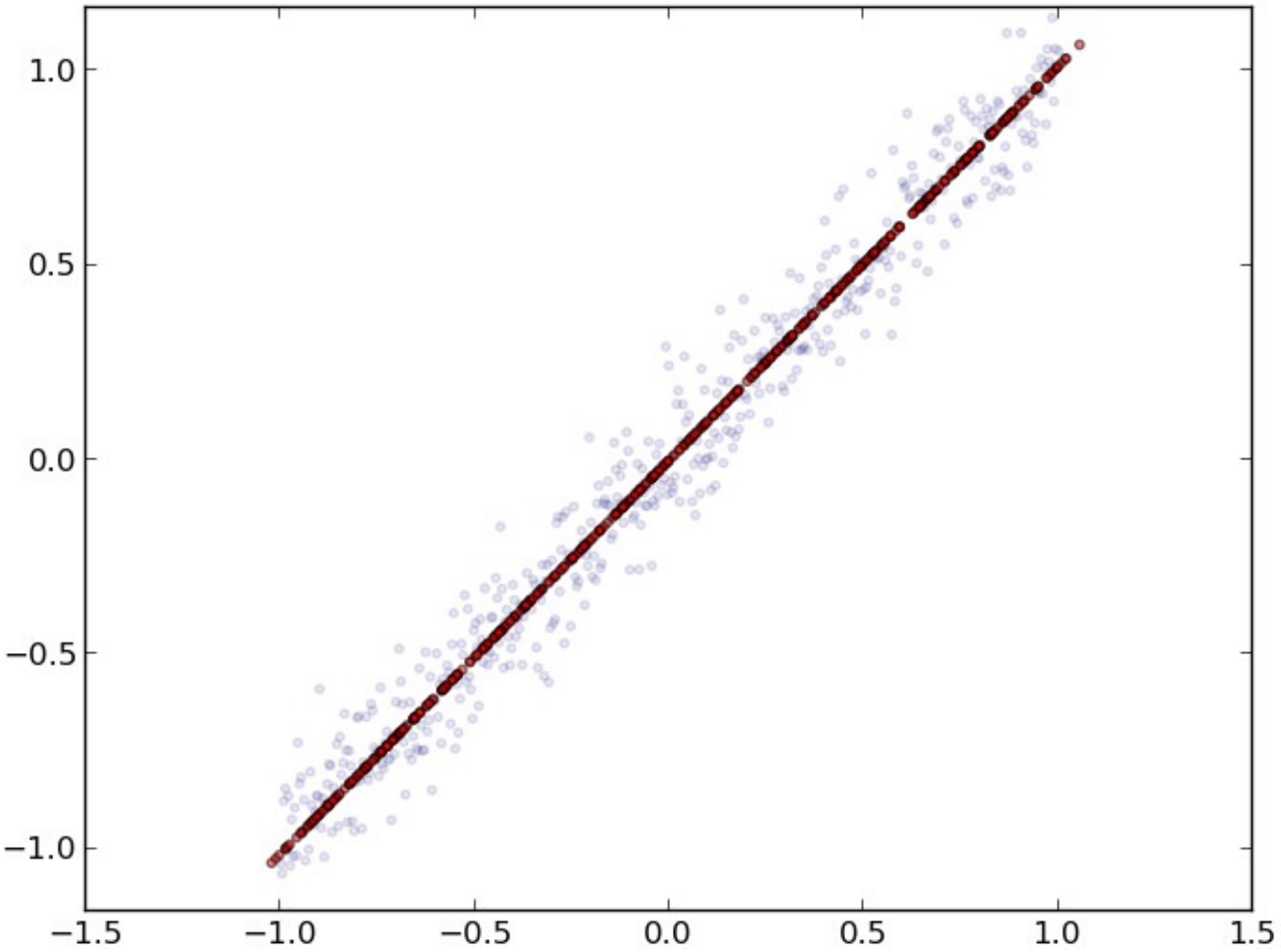
Data is highly correlated



Identification of new directions (principal components) such that the data is uncorrelated


PCA
w/o PCA
Error (meV/atom)
iterations
0
2
4
6
8
10
12
14
0
2000
4000
6000
8000
10000
12000


1.0
0.5
0.0
−0.5
−1.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5



Dimensionality reduction using most significant principal components

- Neural networks capable of reproducing DFT total energies to high precision
- PCA dramatically speeds up the training of NNs applied for these physical problems, allows for easier exploration of NN possibilities
- PCA may allow for dimensional reduction of input space, enabling the use of NNs to systems with a greater number of atomic types and/or improving the transferability

---

Torch7 machine learning library

XSEDE
Extreme Science and Engineering Discovery Environment

A neural network representation of the potential energy surface in Si- and Si-Li systems

Brad Malone, Ekin Cubuk, and Efthimios Kaxiras

HARVARD
UNIVERSITY

The Problem

Harvard

The detailed simulation of many physical processes can be perfomed with knowledge of the potential energy surface (PES)

A large number of important physical phenomena can be studied through the knowledge of the static energy of a system given its atomic configuration, which is known as the PES.. \

The Problem

Harvard

The detailed simulation of many physical processes can be perfomed with knowledge of the potential energy surface (PES)

$$\{R_1 \ldots . R_N\} \xrightarrow{SE} E_{TOT}, \{F_1 \ldots . F_N\}$$

Density functional theory is an efficient approximation for solving this problem. Nevertheless, ab initio MD is still much too expensive to address problems that are on very large length and time scales. For these more efficient approximations are needed.

The Problem

Harvard

The detailed simulation of many physical processes can be perfomed with knowledge of the potential energy surface (PES)

$$\{R_1 \ldots . R_N\} \xrightarrow{SE} E_{TOT}, \{F_1 \ldots . F_N\}$$

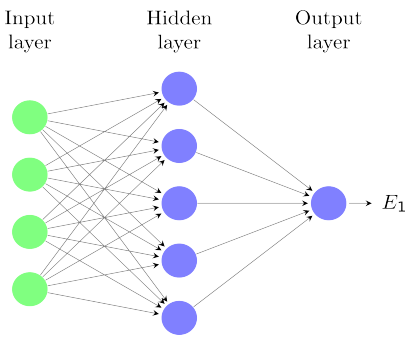
Empirical potentials can be very efficient, although with some loss of accuracy

MM 2014 BD Malone

Empirical potentials are an alternative option, although the functional form chosen can often limit the accuracy possible.

~~–------[[An popular example of this type is the Stillinger Weber potential, which is constructed from both two-body and 3-body terms. This potential favors the configuration when the cosine is -1/3, or in other words, bond angles of the tetrahedral configuration. This sort of potential may work very well for certain systems, such as cubic silicon, but its form may limit the type of physics you can observe within a MD simulation. In particular, if you want to study the phase transformation to a non-cubic polytype of Si, this potential may unfavorably bias against its formation.]]---~~

Artificial neural network

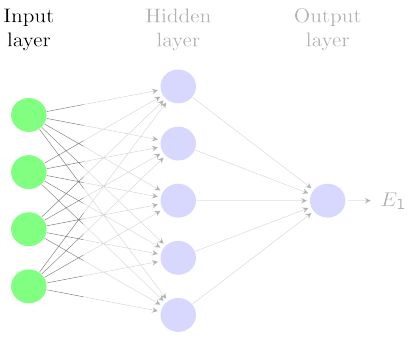


A different approach to this problem can be taken from the machine learning community through the use of the biologically-inspired neural networks, which is an extremely flexible approach in contrast to the fixed form of the empirical potentials and has in previous work shown to be capable of reproducing DFT total energies but at a fraction of the cost.

For those unfamiliar with NN, let me give you a quick summary. The NN is constructed with a certain topology and with some parameters known as weights, marked by the lines between nodes. By learning on a set of examples, these weights can be chosen to minimize the error. A suitably trained NN can then predict output quantities from inputs that it's never seen before.

Input layer

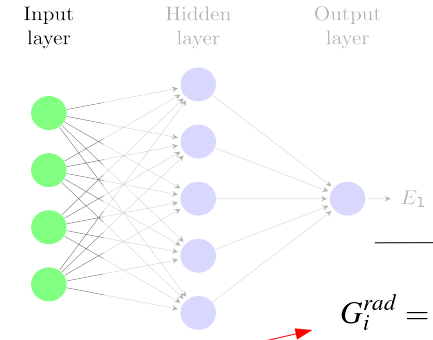
Harvard



Highly beneficial if inputs have obvious symmetries of the system built-in

MM 2014 BD Malone

Now for this input vector the naïve thing to do is to use the cartesian coordinates of the atoms.. However, this is not invariant to translations or rotations of the system, and so it's not a convenient set of functions to describe our system. For this we use so-called symmetry functions which describe the atomic environment around each atom



Highly beneficial if inputs have obvious symmetries of the system built-in

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

Radial functions

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

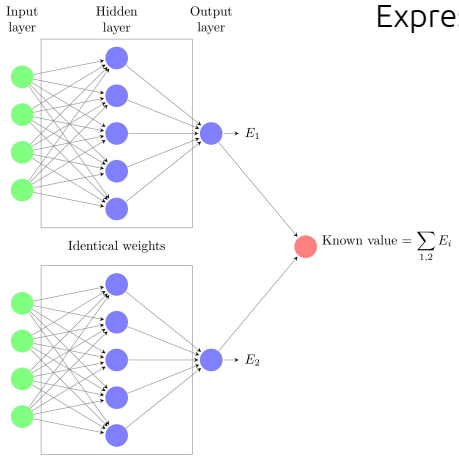
Behler, J. Chem. Phys. **134**,074106 (2011)

Angular functions

Cutoff functions

Behler Parrinello topology

Harvard

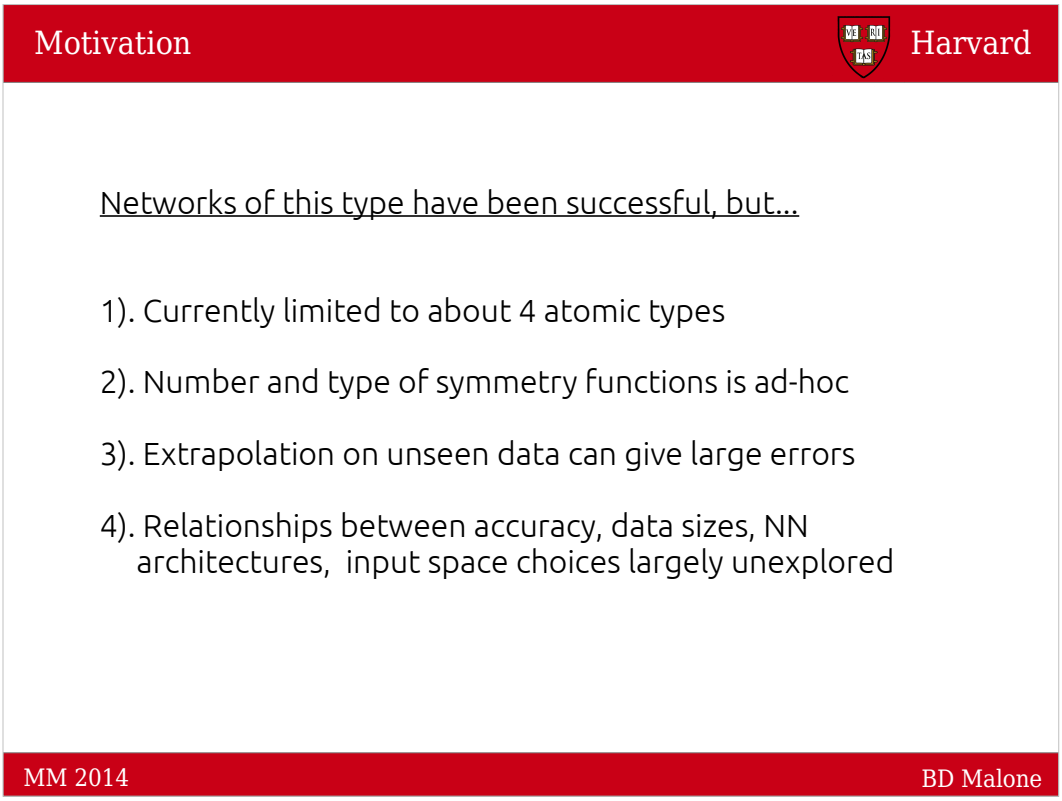


Express total energy as a sum of atomic contributions

$$E^{tot} = \sum_i E_i$$

Behler, Parrinello PRL **98**,146401 (2007)

We use a topology originally introduced by Behler and Parrinello in which we have coupled NN, each of which is supposed to calculate the energy contribution from a single atom to the total energy. If two atoms have the same environment, they should give the same energy, and so these weights are shared. The benefit of this is that it's invariant to permutations and that a trained network can be used for other system sizes by simply adding or subtracting the correct number of individual atom networks.



Now it's been found that networks of this type can be successful, but there are a number of open problems. One, these networks are believed to only be feasible for up to about 4 atomic types, The reason for this is that the number of symmetry functions used to describe the multicomponents system grows rapidly w/ the # of types.

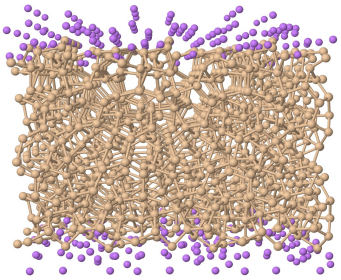
Additionally, the number and type of symmetry functions is currently ad-hoc, and it's unclear if they can improved.

As is common with NN, the network can have large extrapolation errors when it sees an environment it hasn't seen before. Minimizing these possibilities, or at least detecting them, is an important part of building trust with the NN black box.

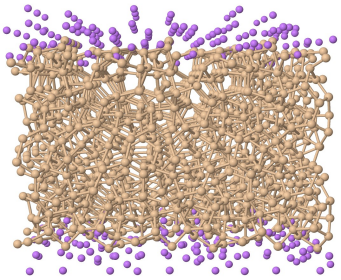
Finally, the relationships between the accuracy obtainable, the data sizes, NN architectures, and input space choices is largely unexplored.

,

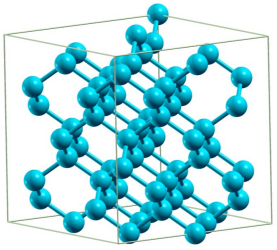
- Si-Li systems for battery applications



- Si-Li systems for battery applications



- Pure Si a good benchmark system



Does this work?

Molten Si system – 64 atom supercell @ 2000K

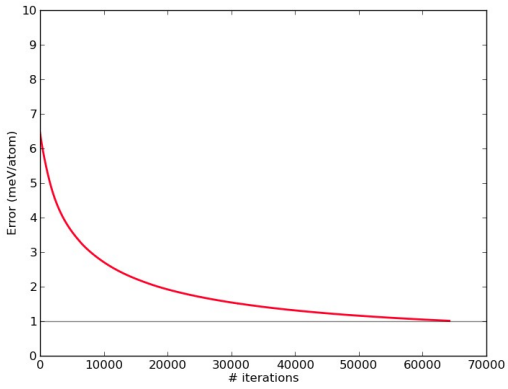
The first question is, whether we can get this method to work. For this we generated around 20,000 MD snapshots of molten Si DFT data. We chose a high temperature of 2000K for this so that we get a wide range of atomic environments over the course of the simulation. This data provides for a nice playground to work within.

So we constructed a NN using the scheme discussed, and here is our training error as a number of training iteration. As you can see, after training for about 65k iterations we get down to 1 meV/atom training error, with the test error only slightly larger. As you can see the error is still going down, but at this point we're already at an accuracy which is lower than a majority of DFT calculations are even converged wrt to.

It's another question as to how accurate this network would be on any possible reasonable Si configuration, but we can see that this approach works well and can learn the DFT Hamiltonian over this space.

Does this work?

Molten Si system – 64 atom supercell @ 2000K



Training error =1.01 meV/atom
Test error = 1.1 meV/atom

So we constructed a NN with 2 hidden layers, 35 nodes per layer, and tried to learn the SE over this set of input, and here is our training errors as a function of training iteration.

Neural net training can take a long time

In general NN can take a long time to train. This can be particularly annoying if we want to look at the behavior of the NN as a function of the # and type of inputs, the size of the network, the type of data in the training set, etc. since each of these tests could take a long time to reach a reasonable level of convergence. Thus anything we can do to speed up the convergence is very helpful in exploring these possibilities.

Training speedup?

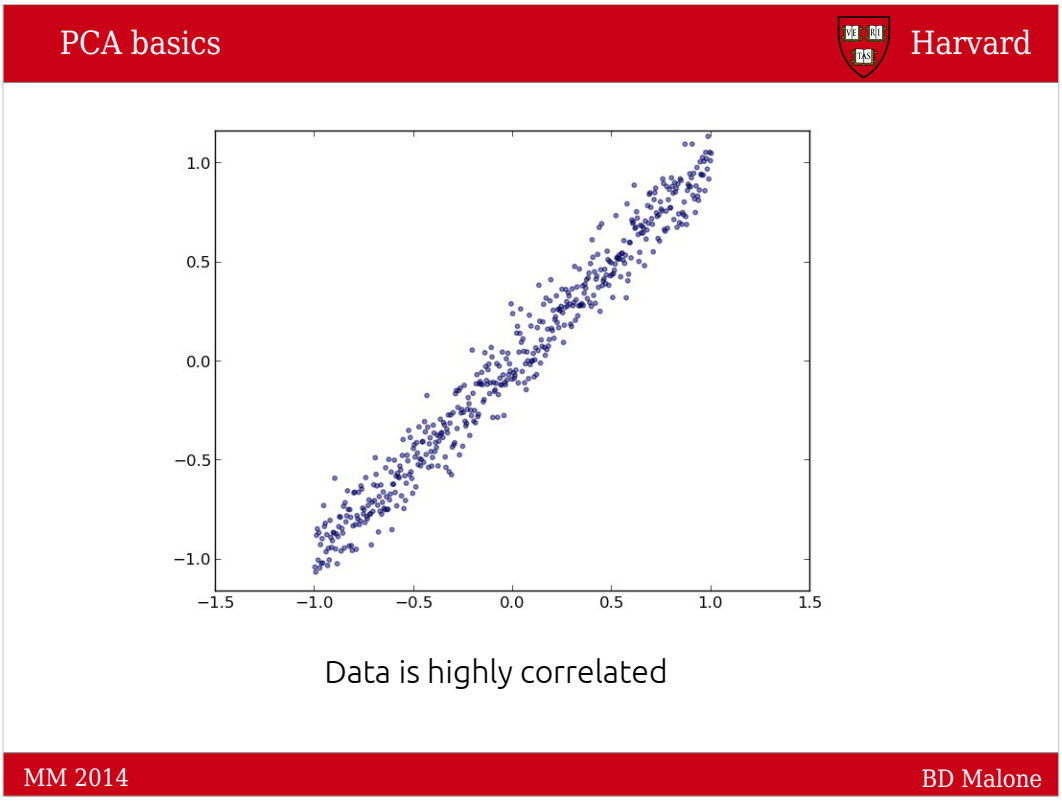
Harvard

Neural net training can take a long time

Use Principal Component Analysis (PCA)
on input space?

In general NN can take a long time to train. This can be particularly annoying if we want to look at the behavior of the NN as a function of the # and type of inputs, the size of the network, the type of data in the training set, etc. since each of these tests could take a long time to reach a reasonable level of convergence. Thus anything we can do to speed up the convergence is very helpful in exploring these possibilities.

PCA basics

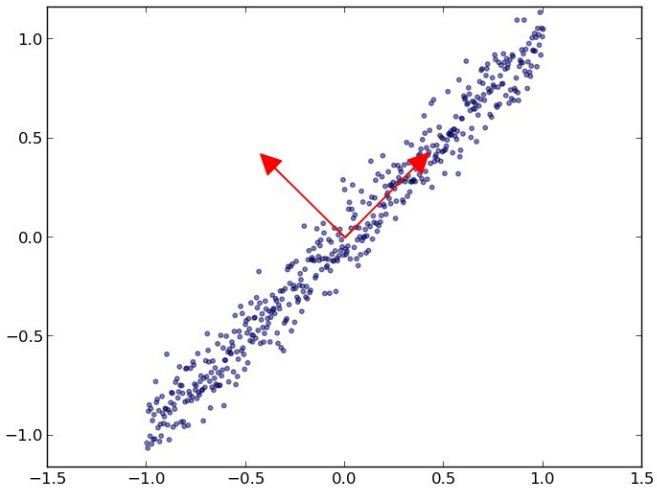


MM 2014 BD Malone

One thing we tried along these lines is to use PCA on the input symmetry functions rather than use the symmetry functions themselves as was previously done. PCA is a technique which creates new inputs from a linear combination of the old such that the new ones are uncorrelated with each other. In the ML community this has been seen to be useful in some cases for training NNS, as it can help the neural net learn the information more easily because it's a simpler optimization problemi if the inputs don't display correlations.

PCA basics

Harvard

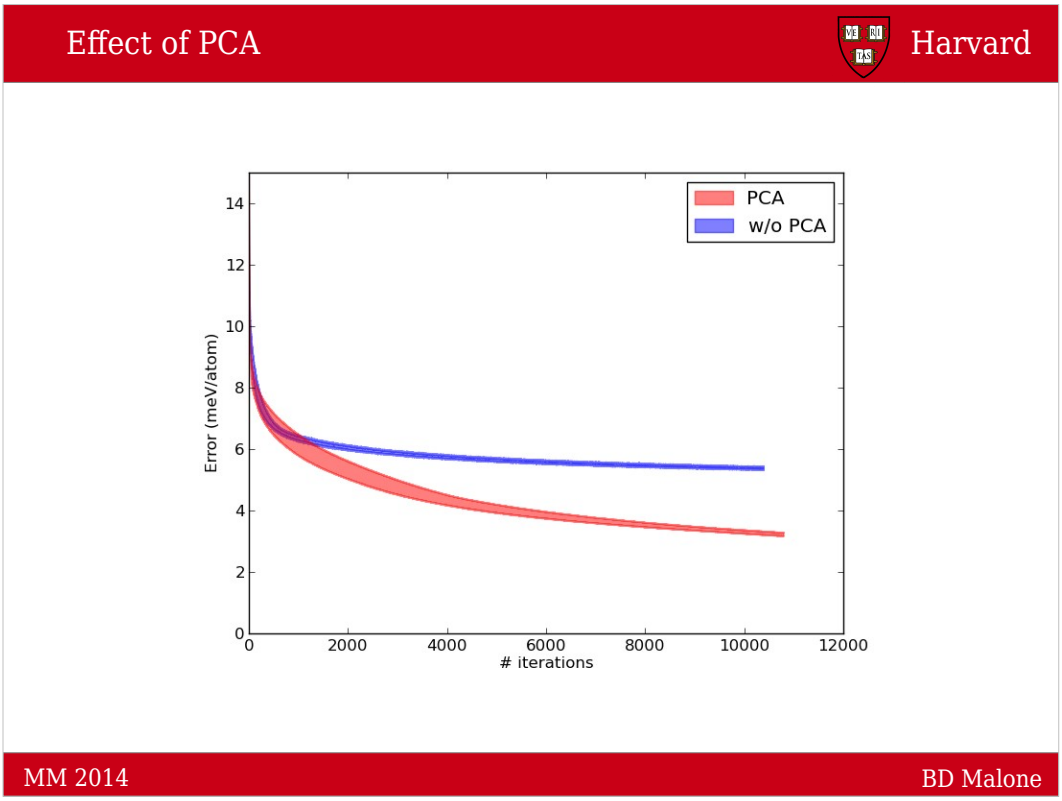


Identification of new directions (principal components) such that the data is uncorrelated

MM 2014

BD Malone

One thing we tried along these lines is to use PCA on the input symmetry functions rather than use the symmetry functions themselves as was previously done. PCA is a technique which creates new inputs from a linear combination of the old such that the new ones are uncorrelated with each other. In the ML community this has been seen to be useful in some cases for training NNS, as it can help the neural net learn the information more easily because it's a simpler optimization problemi if the inputs don't display correlations.



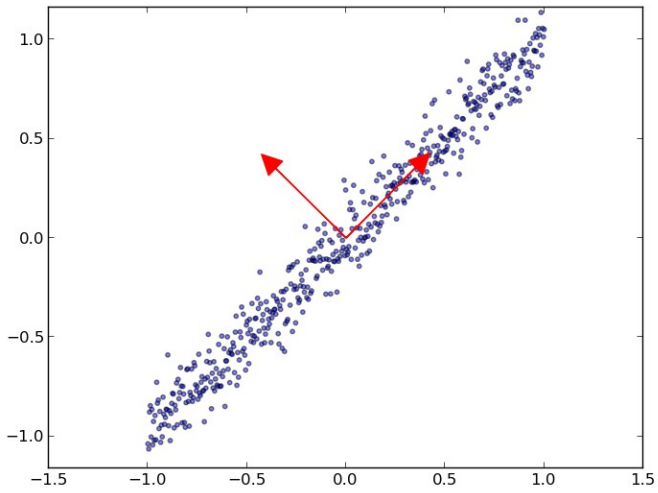
Now we've done the same thing on our symmetry function input space, which in this case is 28-dimensional rather than just 2, but the idea is the same.

Here are some results showing the effect of PCA on the training. The blue shaded region corresponds to the range of training errors corresponding to 5 independent trainings of the NN w/out using PCA. The red corresponds to the same but after applying PCA to the input space. You can see that the training begins reaching low values of error much more rapidly in the PCA case than in the non-PCA case.

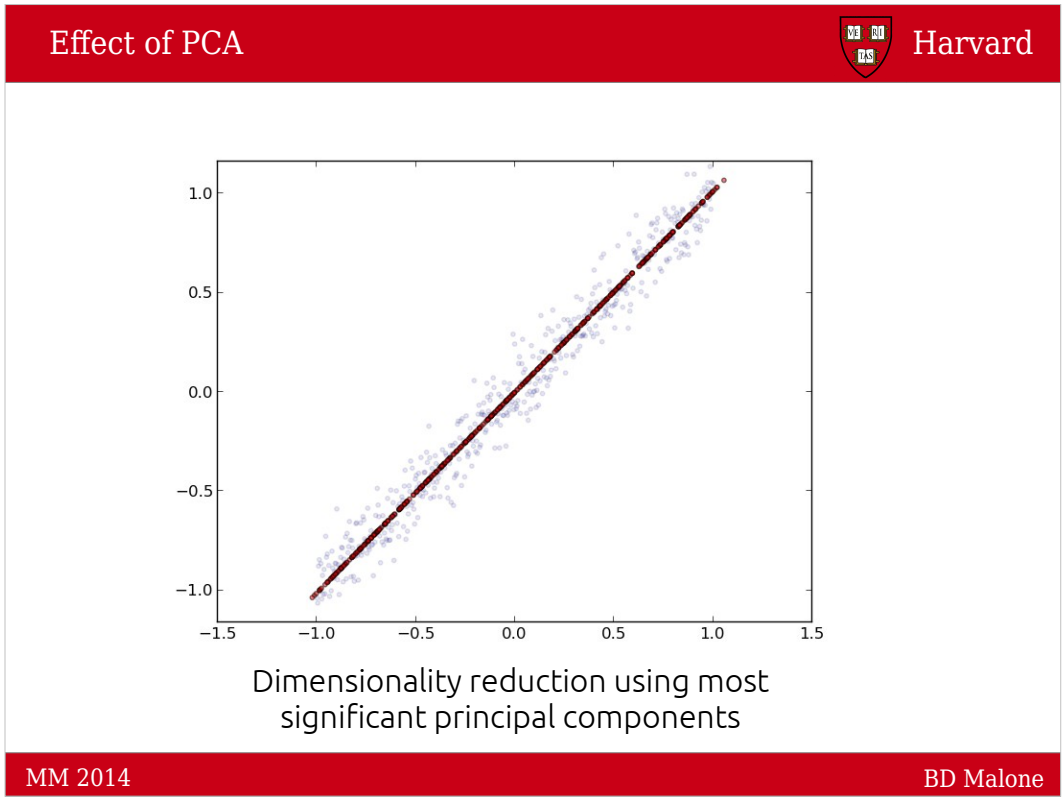
In addition to this improvement in training speed, in performing PCA we realized another thing: that many of the symmetry functions used in previous works actually contain some functions with are linearly dependent upon some of the others, which is as correlated as you can get. After PCA pointed this out, it was easy to show this algebraically as well. So in addition to quicker learning, PCA can point out highly redundant input that will enlarge the # of weights you need to optimize without providing you much benefit.

Effect of PCA

Harvard



Another possibility which is opened up with the use of PCA is the possibility of dimensionality reduction. So back to the 2d example, we had these two principal components. And PCA will rank these in order of greatest to least variance, in this case this would one be the greatest and this the least.



If we drop the component with least variance and convert the data back to our original xy space, we get the following picture. The data is represented by this line of points from their projection onto the greatest principal component. So we've lost some information from our original plot, but the trade off is that we've gone from a 2D to a 1D description.

We are now looking into doing this with the symmetry functions in our NN. The motivation in doing this with the hope that more compact representations will allow for easier training, less data, and better transferability since extrapolation problems might be less pronounced on lower dimensionality spaces.

There's good reason to think we can do this at least to some degree. In the above 2D example I threw out about 1% of the variance in dropping that principal component. In our NN trainings thsufar, we throw out only 1 10-billonth of a percent of the variance by dropping the last of the symmetry functions. So I think it's very unlikely that this will hurt us in the NN performance. And if it does, that's quite an interesting outcome in and of itself.

Conclusions

Harvard

- Neural networks capable of reproducing DFT total energies to high precision
- PCA dramatically speeds up the training of NNs applied for these physical problems, allows for easier exploration of NN possibilities
- PCA may allow for dimensional reduction of input space, enabling the use of NNs to systems with a greater number of atomic types and/or improving the transferability

---

Torch7 machine learning library

XSEDE

Extreme Science and Engineering Discovery Environment

One of the obvious questions is whether this will work. Now in the past it has been shown that errors on the order of 5 meV/atom have been achieved