Capstone Project
Machine Learning
Engineer Nanodegree

Mohammad Rahimi
December 20, 2016

Force Field Optimization

Definition

Project Overview

Molecular dynamics (MD) is a computer simulation technique for studying structure and dynamics of complex systems with extreme detail--literally on scales where the motion of individual atoms can be tracked. MD acts as a bridge between experiment and theory to develop a realistic model to describe experimental measurements. We can test a model by conducting a simulation and comparing the simulation result with experimental measurements. At the same time, the hidden detail behind the experiment can be revealed. The main strength of MD method is the ability to determine the time evolution of a system at atomic-level resolution which brings new insights into mechanisms and processes that are not directly accessible through experiment. In addition, we are able to carry out simulations that are too difficult or impossible to be treated with standard experimental tools. Providing information at atomic-level resolution, MD has subsequently become an invaluable tool to facilitate and complement experimental studies in different research fields such as chemistry, biochemistry, and material science.

Molecular Dynamics simulation is in many respects very similar to real experiments. First, we need to prepare a sample of material that we want to study. A sample consists of initial positions and velocities of atoms in the system. By using the proper interaction potential, the classical equation of motion is solved numerically for all atoms in the system step-by-step. The simulation will be continued until the properties of the system no longer change with time, the so-called equilibrium state. After equilibration, the physical quantities are measured based on the positions and velocities of all atoms in the system. For example, the temperature of the system is a function of velocities of all atoms.

Molecular dynamics uses Newton's equations of motion to computationally simulate the time evolution of a set of interacting atoms. Such techniques are dependent on a description of how molecules will interact. In classical MD simulations, atoms are treated as classical objects, resembling soft ball. The interaction between soft balls is defined by the Lennard-Jones (LJ) potential as follows:

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

where σ is the radius of the soft balls, ϵ is the strength of the interaction, and r is the distance between the atoms. Using LJ potential leads to tremendous computational simplification. In chemistry and physics a description of the terms by which the particles interact is usually referred to as a force field, and the procedure to determine LJ parameters (σ and ϵ) to reproduce experimental data is referred to as force field optimization.

Optimizing a realistic LJ potential that would adequately mimic the real materials is nontrivial. The underlying physics of interaction of two atoms is too complex to describe with simple LJ potential. It has only two parameters which determine the length and energy scales. The potential is thus unique, and cannot reproduce all properties of real materials. Due to the complexity of the problem, an empirical approach is mostly used to optimize force field. In this technique, the optimization procedure is based on the reproduction of experimental measurements by simulating a small system. Via

an iterative approach, the LJ parameters are primarily optimized to reproduce experimental data. This strategy is significantly demanding since MD simulation is very expensive and simulation results are very sensitive to the LJ parameters; slightly change the LJ parameters has a significant influence on the simulation results. Due to the difficulty of empirical force field calculations and the complexity of defining realistic potential, it is difficult or impossible to reproduce more than two experimental data.

The empirical approach for a system with dissimilar atoms is more demanding than a system with similar atoms since the number of LJ parameters increase by increasing the type of atoms in the system. For example, in a system consists of two types of atoms A and B, three different interactions occur: two similar interactions between atoms of the same type A-A and B-B and a dissimilar interaction between atoms of different types A-B. Therefore, it is necessary to define 3 LJ potentials, leading to optimize six parameters. For simplicity, one can use the combination rules to define the LJ parameters for dissimilar interaction as follow:

$$\epsilon_{AB} = \sqrt{\epsilon_A \epsilon_B} \qquad \sigma_{AB} = \frac{\sigma_A + \sigma_B}{2}$$

Using the combination rules reduce the number of LJ parameters need to determine. However, it can affect the outcome of the simulation, and it is necessary to optimize force field for materials with the similar type of atoms.

Problem Statement

Calcium phosphate has many engineering and biomedical application, and we would like to use the MD simulation to detect the formation of calcium phosphate nanoparticles at atomic-level resolution. Calcium phosphate is made of two separate parts, calcium and phosphate. Phosphate is a kind of salt that consists of one central phosphorus atom surrounded by four oxygen atoms. Force field of pure phosphorus and oxygen atoms were developed and validated against experimental data. However, in order to simulate a phosphate we need to optimize existing force field to reproduce experimental measurements.

Phosphate has two types of atoms, and we need to optimize 4 LJ parameters; two LJ parameters for phosphorus and two LJ parameters for oxygen. As a first guess, we used the force field developed for pure phosphorus and oxygen atoms. Then we employed a grid search method to define a set of LJ parameters. For each set of LJ parameters, one MD simulation has been performed and physical properties of the system have been measured. The data set provides a large collection of LJ parameters and simulation results corresponding to each set of LJ parameters.

The main aim of this project is to employ machine learning algorithms to find a relationship between LJ parameters and simulation results, leading to the development of force field to reproduce experimental measurements. It is a supervised learning problem, more specifically a regression problem. The LJ parameters that we want to predict are called the dependent variables or target variables. In contrast to most of the supervised learning problem, here we have more than one target variable. The MD simulation results that we will use to predict the target variables are called independent variables or regressor variables. We also would like to study the influence of the number of training sample in the accuracy of our prediction. As mentioned before, MD simulation is very expensive, and developing a model to predict the LJ parameters with a limited number of training sample can significantly reduce the computational time.

Here, we will employ several algorithms for training, and we will evaluate the performance of each algorithm by calculating an appropriate score. To optimize the parameters of each algorithm, we will use grid search and cross-validation techniques. Finally, the optimized method will be used to predict the LJ parameters based on the experimental measurements. Using the prediction LJ parameters, we will perform MD simulation and calculate the accuracy of MD simulation by comparing simulation results and experimental measurements.

Metrics

To evaluate the performance of the prediction algorithms R squared, R^2, is calculated. R^2 describes the proportion of the variance in the target variable that is predictable from the independent variable. The best possible score is 1.0, and a model with the highest score has the best performance. The R^2 score is also used to study the performance of models as a function of number of training sample.

In the final step, we use the realistic data obtained from experimental measurement as a set of independent variable to predict target variables, and the predicted target variables are used for an MD simulation. To evaluate the accuracy of MD simulation against experiment data, we measure the accuracy as follows:

$$accuracy = \frac{experiment - MD\ simulation}{experiment} \times 100$$

Analysis

Data Exploration

The dataset is organized in the format of csv file and consists of 4355 rows and 14 columns, of which four columns contains LJ parameters, and ten columns are the result of the MD simulation. The dataset is completely clean, and there are no missing values. A description of each feature, along with its data type, is given in theTable 1, and the statistical analysis of the data is given in Table 2.

Feature	Data Type	Description
sigP	Continuous	sigma for phosphorus (LJ parameter)
epsiP	Continuous	epsilon for phosphorus (LJ parameter)
sigO	Continuous	sigma for oxygen (LJ parameter)
epsiO	Continuous	epsilon for oxygen (LJ parameter)
a	Continuous	Unit cell vector
b	Continuous	Unit cell vector
c	Continuous	Unit cell vector
alpha	Continuous	Angle between two unit cell vectors
beta	Continuous	Angle between two unit cell vectors
gamma	Continuous	Angle between two unit cell vectors
density	Continuous	Mass divided by volume defined by unit cell vectors
cleav	Continuous	
interface	Continuous	
solvation	Continuous	

Table 1 - Description of features of dataset

Two features alpha and beta are constant, indicating no relationship between these features and target variables. According to the basic statistical analysis, the features have different scaling, and it has a beneficial effect if we normalize and center the data (with mean 0 and variance 1). Especially, for Ridge and Lasso model where the penalty term might depend on the scaling of features.

Feature	MIN	MAX	MEAN	MEDIAN	STD
sigP	3.800000	4.800000	4.300092	4.300000	0.316242
epsiP	0.200000	0.300000	0.250007	0.260000	0.034161
sigO	3.000000	4.000000	3.500046	3.500000	0.316286
epsiO	0.070000	0.170000	0.119989	0.110000	0.034156
a	26.467610	30.469037	28.657301	28.658189	0.839512
b	26.445006	30.464088	28.657461	28.656921	0.839317
c	26.525660	31.864694	29.239715	29.190841	1.372049
alpha	90.0	90.0	90.0	90.0	0.0
beta	90.0	90.0	90.0	90.0	0.0
gamma	119.909643	120.049333	119.999809	119.999925	0.005926
density	2.357965	3.724957	2.909440	2.892776	0.305426
cleav	0.000000	1679.146074	1347.586270	1374.300229	234.986886
interface	-588.414643	16.102536	-227.372736	-211.415497	138.934645
solvation	-801.979098	-630.241372	-706.517259	-704.165108	39.046300

Table 2 – Statistical analysis of dataset

Exploratory Visualization

To explore the dataset, we calculate the correlation matrix for all regressor variables. Noted that the two features alpha and beta are dropped from dataset since their values are constant. The correlation matrix (Fig 1) shows that the three variables a, b, and c are highly correlated to the density, which is expected since the density directly correlated to the volume of the MD simulation box and the volume is calculated by multiplying the three variables. The linear correlation between the three variables and density is obvious in scatter matrix plot (Fig 2). There is no correlation between gamma variable and other variables. It seems the value of gamma variable has small fluctuations around the mean value with standard deviation of 0.005926. The diagonal of the scatter plot represents the distribution density of the variables. According to these observations, we only have four variables that can help us to develop our prediction model.

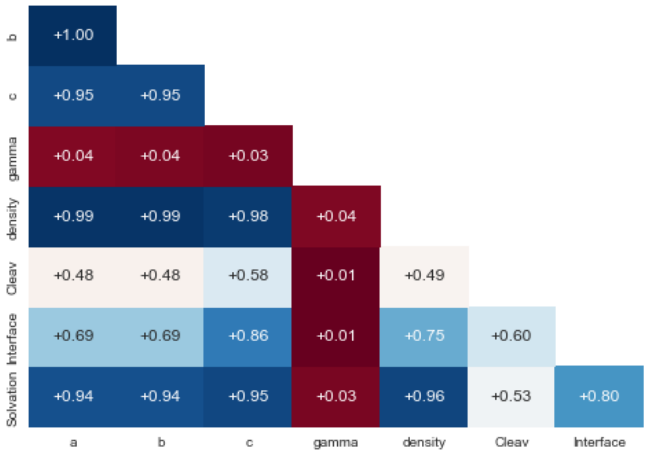

Figure 1 - Correlation matrix

In this project, we have four target variables; changing a regressor variable has an influence on all target variables. Therefore, it is difficult to visualize the relationship between regressor variables and target variables. Here, we try to simplify the problem by calculating the average of a regressor variable over two target variables to reduce the number of target variables from 4 to 2 variables. We use contour plot to show the relationship between one regressor variable and two target variables. For example, we calculate the average of density for a given set of sigP and epsiP, showing in the Fig. 3 up-left. Using all combination of target variables, we demonstrate how density depends on the target variables. Density is anticorrelated with sigP and sigO; reducing these two variables increases the density. It seems there is no correlation between density and epsiP.

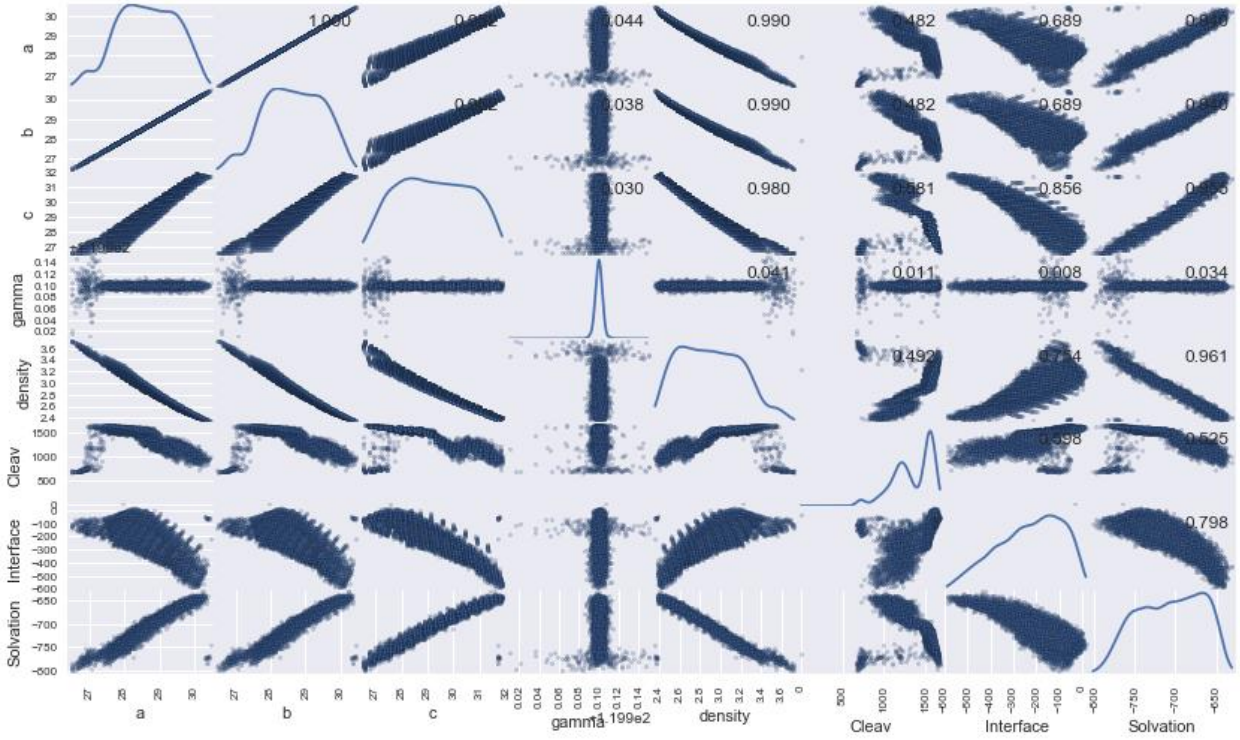

Figure 2 - Scatter matrix plot

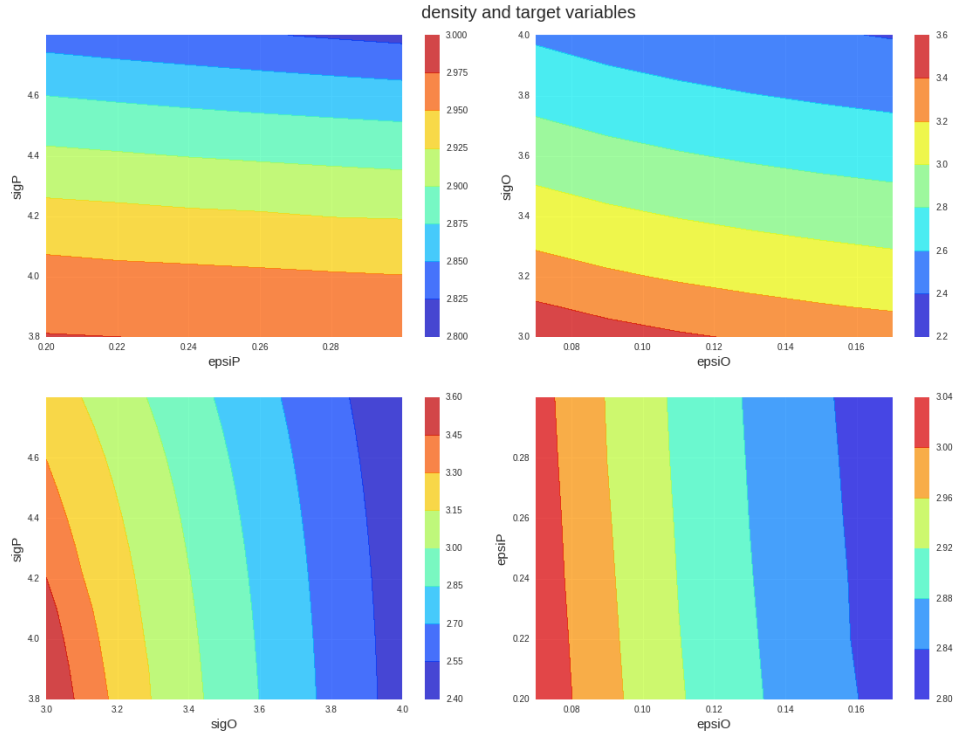

Figure 3 - Contour plot to illustrate the relationship between target variables and density

Algorithms and Techniques

Force field optimization is a regression supervised learning problem, and there are few algorithms available for training a regressor to learn from data. Two algorithms with different complexity are selected based on the nature of problem and training data. For each algorithm, a set of parameters is optimized by using grid search and cross validation technique.

1- Linear Regression

a- *Description:* Linear regression is an approach for modeling the linear relationship between the target variables and regressor variables. It fits a linear model to minimize the residual sum of squares between the target variables in the dataset, and the predicted value by the linear approximation. The linear model is highly biased, and it can only explain the linear relationship between the target variables and regressor variables. The model is extended to describe non-linear relationship by using a polynomial kernel. Employing polynomial function increase the flexibility of the prediction model, leading to the overfitting of the predicted values. To control the flexibility of the model more accurately, we use the techniques that regularize the coefficient estimates such as Ridge, Lasso, and ElasticNet. These techniques impose a penalty on the size of predicted model's coefficients. The Ridge and Lasso solve the minimization of the least-squares penalty with $\lambda \sum \beta_i^2$ and $\lambda \sum |\beta_i|$, respectively. The β is the coefficient and λ is a tuning parameter controls the flexibility of the model. The ElasticNet technique combines the two Ridge and Lasso technique.

b- *sklearn class:* linear_model, preprocessing, pipeline

c- *Default parameters selected for search:*

a- Ridge
alpha : [0, 0.1,0 .2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0]

b- Lasso
alpha : [0, 0.1,0 .2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0]

c- ElasticNet
Alpha : [0, 0.1,0 .2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0]
L1_ratio : [0, 0.1,0 .2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0]

d- PolynomialFeatures
degree : [1, 2, 3]
interaction_only : [True, False]

2- **Gradient Boosting Regressor (GBR)**

Description: In general boosting methods combine weak learners into a single strong learner, in an iterative fashion. Here we restrict our discussion of boosting to the context of decision trees. GBR tries to minimize the mean squared error by growing multiple decision trees. Unlike fitting a single large decision tree to the data, which amounts to fitting the data hard and potentially overfitting, the boosting approach instead learns slowly. At each stage, GBR fits a tree to the residuals from the model as the response and, then, adds this new decision tree into the fitted function to update the residuals. By fitting small trees to the residuals, GBR slowly improves the model in areas where it does not perform well.

d- *sklearn class:* ensemble, multioutput

e- *Default parameters selected for search:*
max_depth : [2, 3, 4, 5, 6, 7]
n_estimators : [100, 200, 300]
max_features : [1, 2, 3, 4]

Benchmark

We use a linear regression with four target variables and four regressor variables as a benchmark. The dataset is split into the training (90%) and testing (10%) dataset. The training dataset is used to train the models, and the testing dataset is used to calculate the R^2 score. Using default values for the model, we obtain 0.48 for R^2 score and 0.007 for mean squared

error. This simple algorithm provides a reasonable result and runs extremely fast. We will use this benchmark to estimate the improvement of the prediction model when using the most complex methods. The goal is to get the better score by employing more complex model and optimizing the parameters.

In this problem, each target value has R^2 score, and the total score is the average of all score with uniform weight. The R^2 score and mean squared error (MSQ) of all target variables are reported in table 3.

	siqP	epsiP	siqO	epsiP
R2	0.778	-0.007	0.955	0.196
MSQ	0.022	0.001	0.005	0.001

Table 3 - r2 score for all target variables obtained from linear regression

Methodology

Data Preprocessing

The dataset is the result of the MD simulations, and it is well organized. Therefore, no preprocessing work is required to clean the data or fill in missing values. We only drop the unnecessary features discussed in the analysis section from the dataset. Then, the dataset is normalized since the scale of values differs amongst the features. After this step, each column has zero mean and unit variance.

Implementation

To implement the learning algorithm to predict the target variables, in the first step, we split the dataset into a training set and a testing set using the function train_test_split from the package sklearn.cross_validation. 90% of the data is used for training the model, while 10% is reserved for testing the model once it has been trained. This split is chosen due to the small number of data in the dataset. Note that this split happens after the preprocessing of dataset explained before.

Once the dataset is split into training and testing sets, the regression model can be built and trained. It is necessary to optimize the parameters of the model to enhance the performance of the prediction models and obtain the best performance. We employ the grid search technique from the sklearn.grid_search package to optimize the parameters. Several parameters are incorporated into the GridSearchCV function. The grid search is performed using K-fold cross-validation (k = 5) with each possible permutation of the input parameters to obtain the best set of input parameters. The input parameters were described for each algorithm before.

The linear regression model has the flexibility to fit nonlinear data by constructing non-linear kernel. Here, we use the make_pipline function from package sklearn.pipline to feed a non-linear kernel to linear regression and build a single object representing a simple non-linear regression. The PolynomialFeature function is used as a non-linear kernel from Sklearn.preprocessing package, and we built a polynomial regression. This technique can also be applied to the Ridge and Lasso model. The flexibility of polynomial function defined by the degree of polynomial function is optimized by using grid search technique.

In contrast to the linear regression model, the GBR model doesn't support multiple outputs. Therefore, the GBR is required to extend to support multi-target regression. The MultiOutRegressor is used along with GBR to train a dataset with multiple outputs. This strategy consists of fitting one regressor per target.

Refinement

The refinement step is to simply dig deeper into the model and the parameter space to develop the best possible regression model. Based on the nature of the problem and the goal of the project, the non-linear regression model is selected for further refinement. The advantage of non-linear regression model over GBR model is that it has the potential to be trained

with a limited number of training sample. As it was mentioned, the dataset is the result of the MD simulation which is very expensive, and reducing the number of dataset will save a lot of computational hours.

To take advantage of both Ridge and Lasso algorithm, we use ElasticNet method with polynomial kernel. The ElasticNet uses two penalty terms separately, and the coefficient of each term should be optimized to obtain the best performance. Additional parameters for polynomial function also should be tuned. Theses parameters were reported in the algorithm section.

Results

Model Evaluation and Validation

To evaluate the performance of all prediction model after parameters optimization, we calculate the score for all target values as well as a uniform average of them (Fig 4). According to our calculations, the linear model, used as the benchmark, provides a reasonable result for SigP and SigO but it can't predict the epsiP and epsiO. Using ridge and lasso models, we cannot improve the performance of the prediction model, showing that the linear model doesn't suffer the overfitting problem. The polynomial function is used as a kernel for linear regression, ridge, and lasso, to study more complex models with non-linear behavior. Similar to the linear models, these non-linear models can't predict epsiP, and we obtain a high negative score for epsiP with polynomial regression and polynomial ridge models, leading to a small score for uniform average. Noted that, we didn't include these two models in figure 4 for clarity. However, the polynomial lasso increases the score by 13%. The polynomial Elastinet which combines the two ridge and lasso models slightly improves the performance. We obtain the best performance by using GBR model, indicate that there is a complex and nonlinear relationship between the target variables and regressor variables.

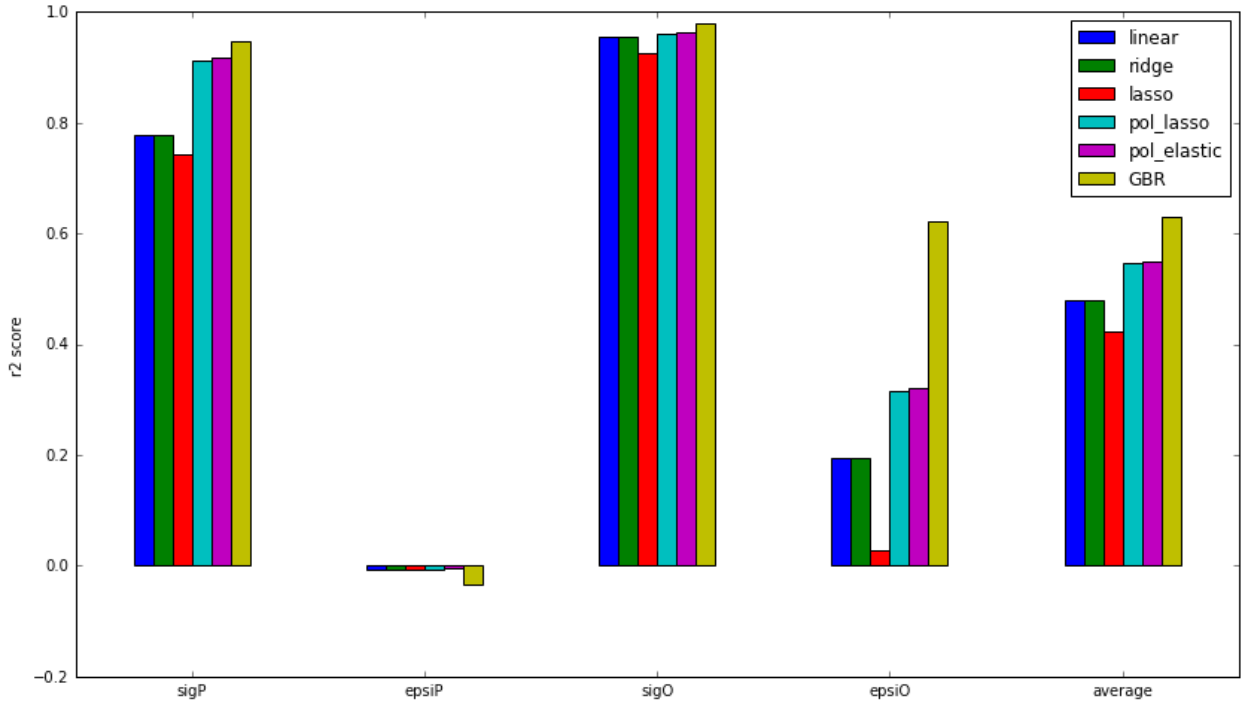

Figure 4 - r2 score for several prediction models

Two best models, polynomial Elastinet and GBR models, are selected for further evaluation and study. The sensitivity of the two prediction models to the subset of the data used for training is measured and plotted in figure 4. As expected for Elasticnet, the both training and testing scores converge to the target value very fast, indicating that the model does not depend on the number of training sample and using more data doesn't improve the performance. In contrast to the Elasticnet, the performance of the GBR strongly depends on the number of training data, and using more data might improve the performance of the model. Overall the performance of the GBR is better than the Elasticnet even for the small number of data. In addition, the standard deviation from the mean value for the GBR model is much smaller than the elasticnet, indicating that using different sample doesn't have a significant influence on the prediction result.

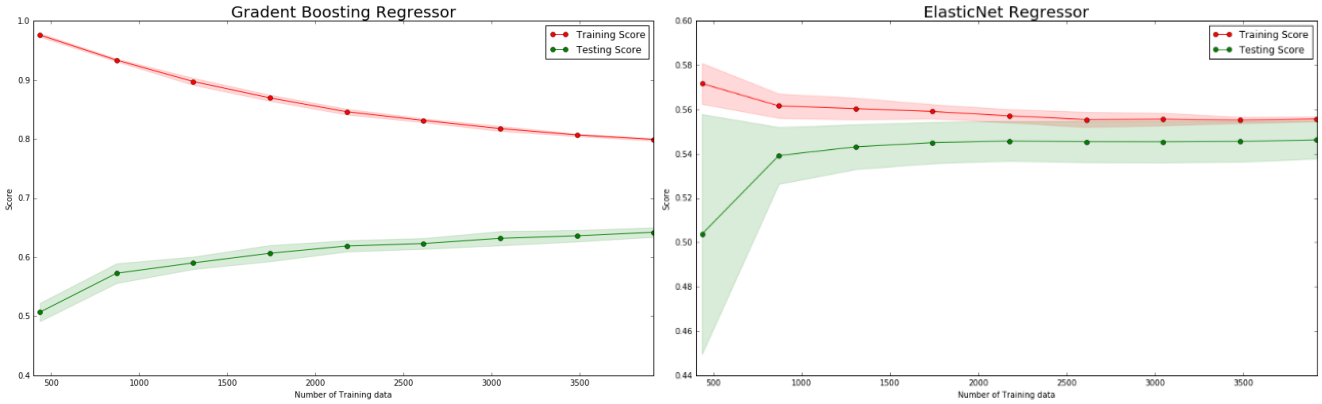

Figure 5 - learning curve for two best prediction models

Justification

As it was discussed in the previous section, using the polynomial Elasticnet and GBR improves the performance of the prediction model compare to the linear regression defined as the benchmark. All model can predict the sigP and sigO in an acceptable range, and employing complex models slightly improve the results. The complex models, especially GBR model, produces a reasonable result for epsiO while the benchmark can't predict the target value in an acceptable range, indicating that the relationship between epsiO and regressor variables is too complex to describe by a simple model such as linear regression. It is interesting that none of the prediction models can predict the value epsiP. As it was shown in the analysis section for density variable, there is no direct correlation between epsiP and regressor variables, leading to the small score for the prediction models.

Conclusion

Free-Form Visualization

The goal of this project was to optimize the MD parameters to reproduce the experimental by using the machine learning algorithms. In order to justify whether developed models can predict the MD parameters, we feed the experimental measurements to the trained prediction models. The MD parameters predicted by the models are used as an MD parameters, and we perform MD simulations. The accuracy of each simulation is calculated based on the function defined in the metric section. Noted that, the best possible score for accuracy is 0, and a model with the lowest score has the best performance. All MD simulations can reproduce density and solvation reasonably. They underestimate Cleav and overestimate Interface. The MD simulation used GBR prediction has the best performance.

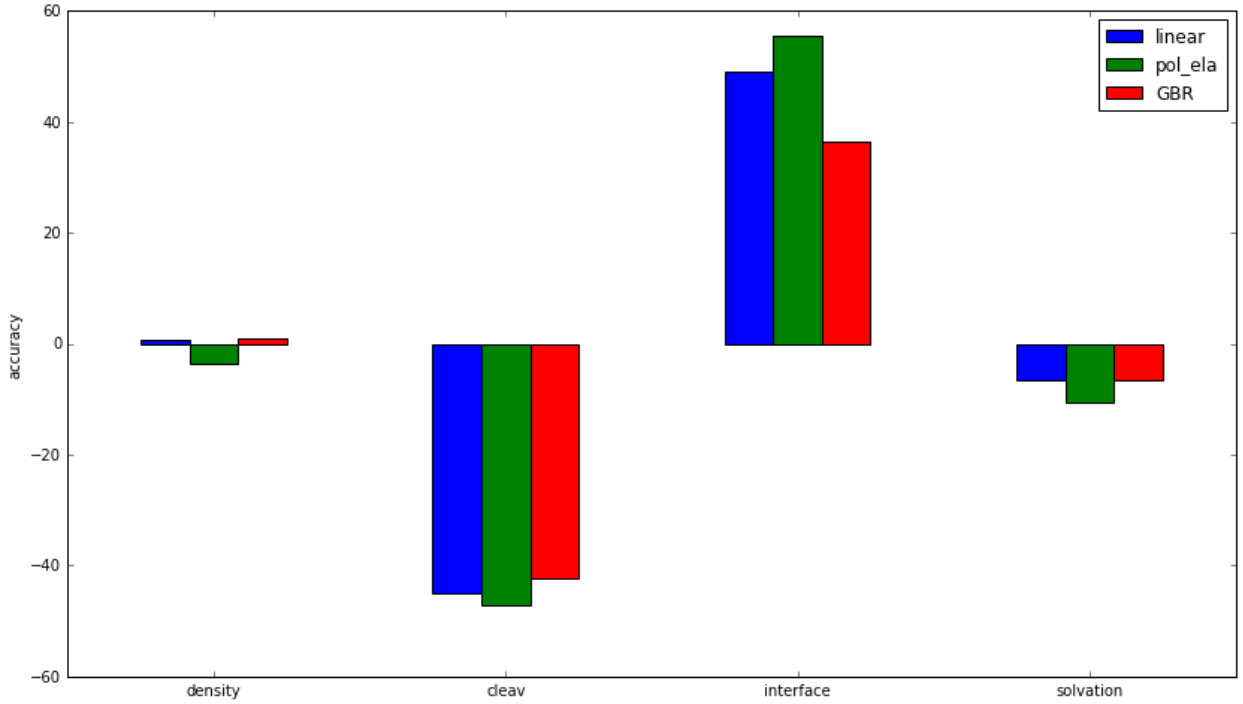

Figure 6 - accuracy of MD simulation against experimental data used MD parameters obtained from different prediction models

Reflection

The project started with a statistical analysis of the dataset obtained from MD simulations. Based on the observations, we have decided to drop some variables from dataset to reduce the number of features. In addition. It was determined that centering and normalization of data have a beneficial for prediction models. Once the preprocessing was done, the data set was split into training and test sets; 10% of the data was reserved for testing the models. Next, several regression models were trained and optimized by using grid search and k-fold cross-validation. Once several regression models were trained, the results of the models on the test data were calculated and compared. It was shown that the GBR model is the best model to use for this problem, although the polynomial with Elasticnet algorithm also produced good results. Finally, the MD parameters predicted with three best prediction models were used to perform MD simulations, and the result obtained from MD simulation compared with the experiment data by calculating accuracy. The MD simulation used MD parameters predicted by GBR could reproduce experimental data more accuracy than other methods.

To the best of our knowledge, the machine learning algorithm has never been used for force field optimization, and the methods developed in this project provides a new technique for computational physical scientists to reproduce the experimental data by performing MD simulations. In addition, this project provides new insight into the relationship between MD parameters and MD simulation result. We are confident that our findings have the potential to publish in a peer-reviewed article.

Improvement

The prediction results might be improved by applying a few improvements:

1- The performance of the best prediction model GBR strongly depends on the number of training data and using more data for training improves the performance of the model.

2- The available data contains four target variables and four regressor variables, however, we found that one target variable is independence to the features, leading to reduce the performance of the prediction models. Adding new features might improve the prediction models.
3- Due to the complexity of the problem, using other model based on the decision tree such as xgboost might lead to better result.