Approximating Critical Temperatures of Superconductors Given Atomic and Chemical Features using Regularized Multiple Linear Regression

by Edward Serafimescu, UCLA '26

Abstract

The critical temperature is defined as the exact temperature limit to enact a moment of immense composition change that corresponds with unique property shifts in their conductivity and magnetic field strengths. It can prove beneficial to accurately determine moments of critical temperature as to mark the efficiency of each material in real-world or laboratory settings, a task possible through a productive model. To test the capabilities of machine learning algorithms, data from the University of California Irvine's Machine Learning Database was extracted and run through several linear regression models. The data contains the chemical formulas and atomic features of known superconductors ranging from atomic radius to thermal conductivity. The objective of the programs are to predict with some accuracy the critical temperature of each superconductor given relevant features. Using methods of regularization, the total error aims to be reduced and the most suitable model found. This task is mainly done to test the capability of python libraries to produce working machine learning models.

1. Introduction

This trial is aimed as assessing the performance of multi-dimensional linear models in reporting results relating to data connected to physics academia. The data is run through several modes of linear regression that moderate coefficient weight values produced from the linear equation such as Lasso, Ridge, and Elasticnet. Below the matrix equation for simple linear regression is defined as:

$$\hat{y} = \mathbf{w}^T \mathbf{x} \tag{1}$$

where the the vector **y** represents a prediction of the target feature given another feature vector *x* multiplied by a learned value weight matrix W.

The goal in producing a successful model is to minimize the loss function based on the given data. The formula is described as:

$$\ell(\mathbf{w}) = \sum (y - (\mathbf{w}^T \mathbf{x}))^2 \tag{2}$$

where ℓ represents the total sum of error generated by the difference of squares of y, the true value of critical temperature, and the predicted value. The most ideal model will produce the lowest possible error based on optimized coefficients and the data given. This is reached by taking the gradient of the loss function, setting the solution to zero, and solving the resulting equations based on their respective variables. With this process, the *d x d* matrix consisting of R real values for w^T can be deduced. The *2 x 2* example yields

$$w_0 = \bar{y} - w_1 \bar{x} \tag{3}$$

$$w_1 = \frac{\sum x_i y_i - \frac{1}{N} \sum x_i y_i}{\sum x_i^2 - \frac{1}{N} \sum x_i \sum x_i} \tag{4}$$

where w_0 and w_1 have become vectors of dimension i, equal to the total data points. The equation for w_0 reflects that the mean values for both feature vectors exist within the model. The equation for w_1 is also known as the covariance of x and y divided by the variance of x. This gives a standard mathematical notion to the idea of association by asserting the relationship to a changing x to a changing y. The equation answers the question to what amount of change can be expected in the target variable given the feature variable. This trend will continue even as more variables and subsequently dimensions are added to the model. The multivariate definition is

$$\mathbf{w} = (X^T X)^{-1} X^T \mathbf{y} \tag{5}$$

where X is a matrix of the data points. This definition of **w** will be used for all models in this program.

With these equations, the fundamentals behind linear regression are established. The data used in the program originates from the University of California Irvine Machine Learning database. The csv file is opened as a pandas dataframe and the target variable is specified. The process begins producing a correlation matrix among every feature in relation to the target feature. The value with the greatest magnitude of correlation would be the feature most likely to dramatically alter the prediction of the critical temperature. In Table 1, the list of features and their coefficients are shown. The feature of the highest magnitude is thermal conductivity, with a value of 0.72. To make a solid prediction, it is important that this feature be represented heavily in the models. Next a scatterplot is generated to view the data representation between critical temperature and thermal conductivity, shown in Figure 1. This will give an idea to how the data will appear in the models. The clustering of the data muddles the ability to predict across a variety of data points, but the objective remains to seek the lowest error. For this trial, all possible feature values will be used. Finally, the data undergoes a 80-20 split for training and test data respectively. The split created sets for both the prediction feature and the target feature.

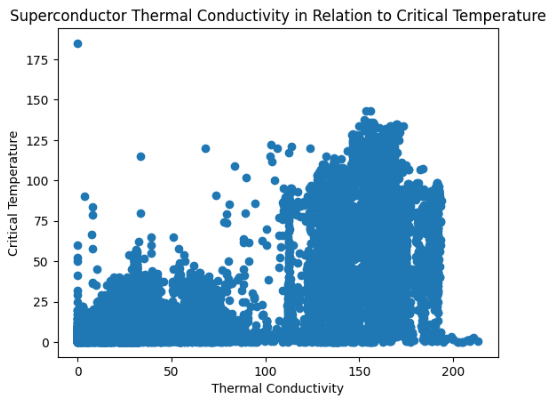

Figure 1: Initial Data Scatterplot

Feature	Correlation
wtd_mean_Valence	-0.632401
wtd_gmean_Valence	-0.615653
mean_Valence	-0.600085
gmean_Valence	-0.573068
gmean_Density	-0.541684
	...
std_ThermalConductivity	0.653632
range_atomic_radius	0.653759
range_ThermalConductivity	0.687654
wtd_std_ThermalConductivity	0.721271
critical_temp	1.000000

Table 1: Features and their correlation coefficients in relation to critical temperature, shown as 1.0. The feature of highest magnitude is most likely to influence the target value.

However, there exists problems in how a model is overfit or underfit. This occurs as a model's coefficients may become too optimized on only a certain amount of the the total data, the training data, and may not be able to accurately predict test cases. This results in incorrect coefficient estimators. To avoid this, the trial code will use methods of regularization.

2. Using Regularization

Regularization is a method applied to the loss function to artificially boost or dampen weight values to avoid overfitting. For this lab, the focus will be on three methods: Ridge, LASSO, and ElasticNet.

2.1. *Ridge Regression*

Ridge regression uses a Euclidean vector weight sum to create a modified loss function. Multiplied by a determined constant alpha, the weights would decay exponentially based on their importance in the overall model. For this example the program only uses one feature, so a higher alpha value of ten will be used due to the large amount of data. The modified loss function is as follows:

$$\ell_r(w_i) = ||y - w_i x_i||^2 + \alpha ||w_i||^2 \tag{6}$$

where w_i represents a certain weight in the vector $\mathbf{w}$. Using scikit-learn's Ridge class, the parameter of alpha was specified with a value of ten. The model is created using the train data and a predict function called using the test data as a parameter. Next, another import statement from scikit.metrics featuring a mean squared error function is called with test-target data and the generated prediction. For Ridge regression, the total error was 310.68. This value and all future values will be rounded to the nearest hundredth.

2.2. *LASSO Regression and GridSearchCV*

Similar to Ridge regression, LASSO alters the loss function but relies on the sum of the absolute value of the weights as opposed to squares. This is known as the Manhattan vector magnitude.

$$\ell_l(w_i) = ||y - w_i x_i||^2 + \alpha |w_i| \tag{7}$$

This ensures features of lesser importance to the target variable are quickly zeroed out as the weights are altered. While this may reduce redundant features, important features may be removed and thus make an inaccurate model. This would hopefully make features of high correlation like thermal conductivity more visible. Following the same process and alpha value as Ridge, the mean squared error totaled as 391.97. In the hopes of reducing this number, both GridSearchCV and KFold classes were imported. The former exhaustively searches for all possibilites of a model using its parameters, and in this case, the aid of KFold. KFold does cross-validation across a given number of validation sets spanning the total data to find the best alpha value since ten proved poor. This search found that an alpha value of two was the best possible fit given the scope of the search. Repeating the process, the new error was 350.78: a considerable improvement.

2.3. *ElasticNet*

Lastly, ElasticNet combines both the regularization techiques of Ridge and LASSO regression. For this program, the ratio between both L1 and L2 regression with be kept the same. Thus the formula culminates as:

$$\ell_e(w_i) = ||y - w_i x_i||^2 + \alpha |w_i| + ||y - w_i x_i||^2 + \alpha ||w_i||^2 \tag{8}$$

Keeping the alpha value at ten, the model is created and the MSE calculated. The total error is 367.83. To attempt to lower the error, another KFold search is done using ElasticNetCV. Obtaining a ndarray of all alphas, the minimum was chosen. The error proved to be 479.03, meaning the search was too broad and one that may yield smaller alphas may prove too costly.

Model	MSE
Linear Regression	304.11
Ridge Regression	310.68
LASSO Regression	391.97
LASSO with GridSearch	350.78
ElasticNet	367.83
ElasticNet with ElasticNetCV	479.03

Table 2: Table comparing calculated models to the resulting mean squared error.

3. Analysis

To see the effectiveness of the models, the MSE of each will be compared.

Another class, representing standard linear regression as shown by the unmodified loss function, is the control. Despite regularization aiming to lower overall error, the control proved to have the least at 304.11. Thus, it seems that for the data given, regularization likely lost meaningful information or added too much bias from the features when attempting to generalize by decreasing estimator variance.

4. Conclusions

To finish, it seems that for multiple linear regression, the unregularized model proved most effective. An explanation may first come from the amount of data as well as it's tendency to be clustered. Ridge regression likely boosted the weights slightly due to higher values on the right side of the scatterplot. LASSO likely did not decay the weights nearly as much as Ridge due to the amount of data, skewing the result. ElasticNet probably convoluted these issues. Since the program served mostly to test models, it is understandable that the results were not ideal. Restricting to standard multiple linear regression for a task of a quantum scale is overly generalized at best. In the future, more complex models and less features to counter the multitude of possible irrelevances may prove the best in finding a model for tangible predictions.