CSE 514A Programming Assignment

Kaitlin Day

ID #473393

# Introduction

## Problem Description

The data set provides characteristics of concrete used in civil engineering applications and a score for comprehensive concrete strength based on these characteristics. Understanding the impact and influence of individual, groups of, or all these characteristics on the concrete strength score can provide useful insights for optimizing concrete quality and type for future construction projects. Most buildings and large-scale infrastructure require concrete foundations to ensure structural integrity. Better awareness of how various characteristics contribute to concrete strength allows project managers to minimize costs and materials required for future projects and encourages continued innovation in building construction.

## Pre-Processing

### Check for Nulls

The data set is first preprocessed by checking for null values and none are found.

### Normalization (Raw Data -> New Data)

The data has been normalized by rescaling all values to the range [0, 1]. This is a common form of normalization and also beneficial when considering percent of total because the value is already a proportion of 1. This is completed by subtracting the mean of a column from each feature in that column and dividing that value by the standard deviation of the column. The standardized values are labeled as “x\_new” and “y\_new” in the code. Throughout the code, the data before normalizing is called referred to as “raw” and the normalized data is referred to as “new”.

### Splitting Training & Testing

Next, the raw and new data are split using the built in sklearn function “train\_test\_split” with a test size of 0.1255 which creates 900 training data points and the remaining 130 data points are testing data points.

Features vs. label before pre-processing:

Diagram

Description automatically generated

Features vs. label after pre-processing:

Diagram

Description automatically generated

## Algorithm Details

### Stopping Criterion

### The stopping criterion chosen for each of the models is a combination of the gradient value being less than 10e-6 and the iteration count exceeding 2000. Both of these cases are included in the while loop for each model because the raw and new data have very different values, so the stopping criterion that applies to raw data may not be as effective for new data. Together, the two stopping criterion ensure a thorough, yet reasonable complexity for each of the models created for both raw and new data in univariate and multivariate models.

### Parameter Update Procedure

The parameter procedure is performed using standard gradient descent, not stochastic gradient descent. This is largely done because stochastic gradient descent would drastically increase the volume of computations required to update m and b to values that would result in an acceptable MSE and variance explained. Therefore, I update m and b for each data point and use the final values as the optimal m and b of the model.

### Learning Rate

The learning rate for each model was selected using the best\_alphaUni and best\_alphaMulti methods which take in the x and y vectors to train the model. A vector of 5 alpha values evenly spaced from 1e-6 to 1e-5 for univariate models and 1e-7 and 1e-8 for multivariate models is initialized inside each method. For each value, a univariate or multivariate model is created and the MSE (initialized at 1000000) is compared to the current best MSE of other models with different alpha values. The value of the model with the lowest MSE is returned as the best\_alpha for the given x and y vectors. For univariate models, a separate value is determined for each feature.

## Algorithm Pseudo-Code

### Find best or values for given model

Check range of or values for each feature in univariate & entire model in multivariate

Compare MSE / MAE of models with different or values & chose model with lowest MSE / MAE

### Find m, b R2 & MSE for models w/ optimized or values

Gradient Descent algorithm:

Input: x, y (labels), feature number (only for univariate), (learning rate), (only for Ridge Regression)

Initialize parameters (m = 0, b = mean(y)) & other variables (sum\_m, sum\_b, dLdm, dLdb, iteration count)

Perform gradient descent while gradient > 10e-6 or iteration count < 2000

For each data point in x, find the predicted value using the current m & b: y = m \* x + b

Add to a sum

Add to a sum

Find final m & b gradient for all data points by dividing both sums by n

Update m:

Update b:

Set and

Calculate Variance, MSE & R2

Return final m, b, R2, and MSE

### Use final m & b values to evaluate model performanc on testing data

Initialize MSE, MAE (for MAE models), Variance, and R2 lists

Find the Variance, MSE, MAE, and R2 values for each model

* Variance is evaluated using the built in numpy function np.var( ) on the labels
* MSE is evaluated using the appropriate MSE function defined earlier in the code where
* R2 or variance explained is evaluated using the previous two values in the equation below

## MAE Implementation

The implementation of MAE is very similar to the algorithm for MSE with MAE as the loss function

This changes the gradient functions for the parameter weights and to

Additionally, the “best” alpha values used for each model are different. These values are found by comparing models with different alpha values like the MSE alpha selection method but uses MAE as the comparison measure rather than MSE. Therefore, the best alpha value for the model for each feature is the value that results in the lowest MAE out of the range of values specified in the method (chosen on trial and error).

## Ridge Regression Implementation

The implementation of Ridge Regression is very similar to the algorithm for MSE with an norm term added to the sum with a hyperparameter multiplied by the weight(s) of the features. As a result, the gradient functions for the parameter weights and to.

The optimal values for are computed using the same method as the MSE models because the effect of lambda on the step size is assumed to be negligent. However, the hyperparamter is tuned using a similar method to hyperparameter tuning for MSE and MAE. In the method, the used in the model with the lowest MSE is choses as the optimal hyperparameter for the feature or features in the model.

# Results

## MSE

Range of alpha values:

* Univariate Data: [1e-6, 1e-5]
* Multivariate Data: [1e-7, 1e-6]

### Raw Data

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Best alpha | MSE | Variance observed | Train R2  (Variance Explained) | Test R2  (Variance Explained) |
| Feature 1 | 3.25e-6 | 209.8614 | 300.8179 | 0.2283 | 0.3024 |
| Feature 2 | 3.25e-6 | 297.0304 | 300.8179 | 0.0119 | 0.0126 |
| Feature 3 | 3.25e-6 | 290.0713 | 300.8179 | 0.0061 | 0.0357 |
| Feature 4 | 1e-5 | 286.5146 | 300.8179 | 0.0652 | 0.0475 |
| Feature 5 | 5.5e-6 | 273.1951 | 300.8179 | 0.1206 | 0.0918 |
| Feature 6 | 1e-6 | 301.6408 | 300.8179 | 0.0013 | -0.0027 |
| Feature 7 | 1e-6 | 302.5149 | 300.8179 | 0.0024 | -0.0056 |
| Feature 8 | 3.25e-6 | 279.2512 | 300.8179 | 0.0847 | 0.0717 |
| Multivariate | 3.25e-7 | 176.5178 | 300.8179 | 0.4026 | 0.4132 |

### New Data

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Best alpha | MSE | Variance observed | Train R2  (Variance Explained) | Test R2  (Variance Explained) |
| Feature 1 | 7.75e-6 | 0.7534 | 1.0779 | 0.2385 | 0.3002 |
| Feature 2 | 7.75e-6 | 1.0518 | 1.0779 | 0.0167 | 0.0242 |
| Feature 3 | 7.75e-6 | 1.0481 | 1.0779 | 0.0079 | 0.0276 |
| Feature 4 | 7.75e-6 | 1.0104 | 1.0779 | 0.0866 | 0.0626 |
| Feature 5 | 7.75e-6 | 0.9451 | 1.0779 | 0.1350 | 0.1232 |
| Feature 6 | 7.75e-6 | 1.0281 | 1.0779 | 0.0235 | 0.0462 |
| Feature 7 | 7.75e-6 | 1.0651 | 1.0779 | 0.0299 | 0.0119 |
| Feature 8 | 7.75e-6 | 0.9479 | 1.0779 | 0.1045 | 0.1206 |
| Multivariate | 3.25e-7 | 0.4215 | 1.0779 | 0.5447 | 0.6089 |

## MAE

* Univariate Data: [1e-9, 1e-7]
* Multivariate Data: use value from Multivariate w/ MSE as loss function

### New Data

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Best alpha | MAE | MSE | Variance observed | Train R2  (Variance Explained) | Test R2  (Variance Explained) |
| Feature 1 | 7.525e-7 | 0.6961 | 0.7635 | 1.0779 | 0.2318 | 0.2917 |
| Feature 2 | 7.525e-7 | 0.8101 | 1.0718 | 1.0779 | 0.0100 | 0.0056 |
| Feature 3 | 5.05e-7 | 0.8086 | 1.0611 | 1.0779 | 0.0052 | 0.0156 |
| Feature 4 | 7.525e-7 | 0.7872 | 1.0257 | 1.0779 | 0.07661 | 0.0484 |
| Feature 5 | 1e-6 | 0.7602 | 0.9775 | 1.0779 | 0.1210 | 0.0931 |
| Feature 6 | 1e-6 | 0.7839 | 1.0519 | 1.0779 | 0.0123 | 0.0241 |
| Feature 7 | 5.05e-7 | 0.8078 | 1.0743 | 1.0779 | 0.0273 | 0.0034 |
| Feature 8 | 5.05e-7 | 0.7527 | 0.9722 | 1.0779 | 0.1017 | 0.0981 |
| Multivariate | 7.75e-7 | 0.5275 | 0.4415 | 1.0779 | 0.5346 | 0.5904 |

## Ridge Regression

Lambda Optimization

* Univariate Data: [7.75e-9, 7.75e-7]
* Multivariate Data: use value from Multivariate w/ MSE as loss function

### New Data

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Best lambda | Best alpha | MSE | Variance observed | Train R2  (Variance Explained) | Test R2  (Variance Explained) |
| Feature 1 | 7.75e-8 | 7.525e-7 | 1.0596 | 1.0779 | 0.0963 | 0.2318 |
| Feature 2 | 7.75e-8 | 7.525e-7 | 1.0797 | 1.0779 | 0.0068 | 0.0077 |
| Feature 3 | 7.75e-8 | 5.05e-7 | 1.0799 | 1.0779 | 0.0064 | 0.0282 |
| Feature 4 | 7.75e-8 | 7.525e-7 | 1.0754 | 1.0779 | 0.0347 | -0.0051 |
| Feature 5 | 7.75e-8 | 1e-6 | 1.0713 | 1.0779 | 0.0119 | -0.0052 |
| Feature 6 | 7.75e-8 | 1e-6 | 1.0784 | 1.0779 | 0.0024 | 0.0261 |
| Feature 7 | 7.75e-8 | 5.05e-7 | 1.0797 | 1.0779 | 0.0244 | -0.0065 |
| Feature 8 | 7.75e-8 | 5.05e-7 | 1.0779 | 1.0779 | 0.0848 | 0.1067 |
| Multivariate | 3.25e-10 | 3.25e-7 | 0.4215 | 1.0779 | 0.5447 | 0.6089 |

## Plots of Data w/ Model

### MSE on Raw Data

Graphical user interface

Description automatically generated

### MSE on New Data

Graphical user interface, application, Excel

Description automatically generated

### MAE on New Data

Graphical user interface, application, Excel

Description automatically generated

### Ridge Regression on New Data

### Graphical user interface, application, Excel Description automatically generated

# Discussion

## Model Comparision

### Variance Explained (R2 values)

When looking at both the raw and new data, the best models were the multivariate models that contain all 8 features, rather than the univariate models that only contain one feature. This is expected because the more features that make up a model, the higher the likelihood that the model complexity will match the complexity and trends in the training and testing data it predicts.

Among the univariate models created on raw data, the models with the lowest MSE are built using features 1, 5, and 8 (Cement, Superplasticizer, and Age) with MSE values of 209.8614, 273.1951, and 279.2512 respectively. These models also had the highest variance explained (R2)values of 0.3024, 0.0918, and 0.0717respectively, which is expected because variance explained depends on MSE and variance observed, and variance observed is constant across all models. Similarly, for the univariate models on new data, the best features were 1, 5, and 8 with MSE values of 0.7534, 0.9451, and 0.9479 and R2 values of 0.3002, 0.1232, and 0.1206 respectively. The multivariate models using MSE as the loss function have a R2 of 0.4132 for raw data and 0.6089 for new data. Based on the R2 values for raw and new data, the new data models were better predictors of cement concrete strength, so preprocessing the data strengthened our models.

Comparing the univariate models on new data using MAE as the loss function reveals that the features with the best models are 1, 5, and 8 (Cement, Superplasticizer, and Age). The MAE values of for each model are 0.6961, 0.7602, and 0.7527 and they have R2 values of 0.2917, 0.0931, and 0.0981 respectively. These models are the sane top performing models as the algorithm using MSE as the loss function, indicating that these features are likely the strongest predictors of concrete strength, especially Cement. Predictably, the multivariate model performed much better than the univariate models with an MSE of 0.4415 and R2 of 0.5904.

The Ridge Regression top performing univariate models are built using features 1 and 8 with MSE values of 1.0596 and 1.0779 and R2 values of 0.2318 and 0.1067 respectively. Interestingly, feature 5’s model does not perform well in comparison to other features as it did with MSE and MAE. This may indicate that feature 5 may not have as significant of a contribution to Concrete Strength as previously believed based on the MSE and MAE models. Despite this variation from previous results, the multivariate regression model still performs noticeably better than the univariate models with a R2 of 0.6089.

### Training vs. Testing Models

For most of the univariate models and all the multivariate models, the model performed better on the testing data than on the training data. This is an indication of good model performance because the models are not overfit to the training data but have enough information from the training data to accurately predict unseen data. This conclusion is demonstrated by the increased R2 values in the testing data models compared to the training data models.

### Hyperparamter Tuning

When tuning the hyperparameter to find an optimal learning rate for updating the weight parameters and intercept, trial and error was implemented to find the best range of values to select from. For univariate MSE, the value is selected from 5 evenly spaced values in the range [1e-6, 1e-5]. For multivariate MSE, the alpha value was lowered because there are more features in the model, so a smaller step size allows to for fine tuning of all features without any stepping too far. The multivariate MSE is selected from 5 evenly spaced values in the range [1e-7, 1e-6].

In univariate MAE, the value is selected from 5 evenly spaced values in the range [1e-8, 1e-6] and the multivariate MAE value is selected from 5 evenly spaced values in the range [1e-7, 1e06].

In univariate Ridge Regression, the values used are the same values from the MSE univariate model there was not a large difference in the computed optimal values so using a previously established variable is more efficient. The optimal value is selected from 5 evenly spaced values in the range corresponding to the feature. The result is the same = 7.75e-8 for each univariate model. This is clearly the optimal value because is not on the boundary of the range. In the multivariate model, many were tested using the from the multivariate model. There was not a significant different between the models with varying value of , so a value smaller than with a reasonable performance was chosen.

## Conclusions

The factors that best predict concrete comprehensive strength are Cement (feature 1), Superplasticizer (feature 5), and Age (feature 8). The most influential factor is Cement because it consistently had the highest R2 value among all univariate models, followed by Age, and finally by Superplasticizer. Therefore, when mixing concrete to optimize comprehensive strength, it is most important to consider Cement where higher values result in higher strength (positive parameter m value). Next, the Age of the mixture should be considered where the younger the mixture, the stronger the Concrete (negative parameter m value). Then, the Superplasticizer amount should be optimized where a higher concentration in the Concrete results in stronger Concrete (positive parameter m value). The remaining characteristics cannot be ignored completely because statistical significance tests have not been performed to determine whether they can be dropped from the multivariate models and not affect the accuracy. Therefore, the remaining 5 features should be considered with equal weight when predicting Concrete Comprehensive Strength.

# Instrucrtions

### Open the Python Jupyter notebook (.ipynb) in your desired interface tool (Jupyter Notebook, Visual Studio Code, Linux, etc.)

### Select “Cell” -> “Run All” from the Menu Bar

### \*\* It will take a few minutes to run all the way through but results will be printed after each cell as the program runs!