Reading in Data

Input: filename, which is the string that represents the name of the files we want to read data from

Output: a matrix formed from number1, a sequence whose elements are numbers read from the file filename.

Method Used: file2url, whose argument is a string that represent file's name

urlopen, whose argument is a given url

readlines, which is used to get the original file in the form as lines

Recipe:

1. Create an empty sequence and use the variable number1 to refer to this sequence.
2. Make the filename become a url by calling the method file2url, and use the variable url to refer to this url
3. Open url by calling the method urlopen and assign the value of the opened file to the variable netfile
4. Read every line of netfile by calling the method readlines and iterate over each line in the netfile. Then use the variable line to refer to each line in netfile.

   Inside the iteration: 1. Split line into a sequence whose elements is the strings from the line , use the variable number to refer to this sequence.

   2. Iterate each number from 0 to the value of length of the number; assign the value to the variable index.

   Inside the iteration:

   Make the string become decimal numbers; Then assign the value of numbers to the elements whose index is index in the sequence number

   3. Add number into number1

5. Make number1 into a matrix, return the matrix formed by number1

Understanding the Model: Generating Predictions

Input: A linear model using the m*1 matrix of weights gotten from itself

Inputs, a m*n matrix of explanatory variables

Output: a n*1 matrix, which is the product of inputs and weights

Recipe:

1. Get the weights of the class LinearModel itself

   Return the product of inputs and weights

Understanding the Model: Calculating Prediction Error

Input: A linear model using the m×1 matrix of whose weights is gotten from itself

inputs, which is a n×m matrix of explanatory variables

actual_result, which is a n×1 matrix of the corresponding actual values for the measured variables

Output: the mean squared error between inputs and actual_result

Algorithum used: mse(), whose two arguments are two sequences, one is the expected result and another one is actual result. This function is used to calculate the mean squared error between expected result and actual result.

shape(), which is used to get a sequence whose first element represents the number of rows of the matrix while the second element represents the number of column of the matrix

generate_predictions(), which is used to predict a matrix of

measured variables given a matrix of input data. Its argument is the matrix inputs that represent input data

zip(), which is used to combine two sequences into one sequence whose elements is a sequence whose elements are from the original given two sequences. The arguments are two given sequences, result and expected.

Recipe:

1. At first, define a function mse() outside the class LinearModel, which is used to calculate mean squared error.

Inside the definition:

1. Create a variable sum_squares. Assign the value of 0 to the sum_squares.
2. Combine two given sequences into one sequence by calling the zip function, iterate the first element in the elements inside the sequence and assign its value to the variable num1, iterate the first element in the elements inside the sequence and assign its value to the variable num2.
3. Add the value of the variable sum_squares and the value of the square of the difference between num1 and num2.
4. Make sum_squares into a decimal number, divide it by the value of the length of the sequence expected. Assign the value of the quotient to the variable err.
5. Return the variable err

2. Create two empty sequences referred by the variable list1 and list2.
3. Call the method generate_predictions to generate predictions for the given inputs, assign the value to the variable inputs1
4. Iterate each number from 0 to the value of the number of rows of the matrix inputs1 (the value of the number of rows is the elements whose index is 1 in the sequence gotten by calling the method shape .

   Inside the iteration:

   1. Add the element whose index of row is index and whose index of column is 0 inside the matrix inputs1 into list1
   2. Add the element whose index of row is index and whose index of column is 0 inside the matrix actual_result into list2

5. Call the function mse() to calculate the mean square error between list1 and list2, return the result of the calculation.

Written component for fit least squares:

$$\frac{1}{n}(2w^T X^T X - 2y^T X) = 0$$

$$2w^T X^T X = 2y^T X$$

$$w^T X^T X = y^T X$$

① $X = 0$ all elements value in

② $X \neq 0$ all elements value in

$$w^T X^T = y^T$$

$$(Xw)^T = y^T$$

$$Xw = y$$

$$(X^T \cdot X) \cdot w = X^T y$$

$$(X^T \cdot X)^{-1} (X^T \cdot X) \cdot w = (X^T \cdot X)^{-1} \cdot X^T y$$

$$w = (X^T \cdot X)^{-1} \cdot X^T y$$

Discussion:

1. This is correct only in the given case because we know that, X represents a n*m matrix, y represents a n*1 matrix and w represents a m*1 matrix.

   Therefore yT*X is the product between a 1*n matrix and a n*m matrix, which is a 1*m matrix. Then when the product multiply w, the result is the product of a 1*m matrix and an m*1 matrix, which is a 1*1 matrix. At the right side, wT is a 1*m matrix and XT is an m*n matrix. As a result, their product is a 1*n matrix.

   Then the product of the 1*n matrix and y, a n*1 matrix, is also a 1*1 matrix. In another word, the dimension of the matrix at both sides of the equation is 1*1. So it can work in this case. However, in other cases, the dimensions may be quite different (e.g. all are n*1 matrix), which may cause the dimension error when bring multiplied.

2. As lameda increases, the value to be minimized will also increase. the weight vector output by the LASSO algorithm will increase as λ increases.
3. The fit LASSO Estimation

Yes, these weights also best predict the number of wins on the test 2001-2012 data.

The conclusion is : Compared with least square fit, Fit LASSO Shooting model is a better way to make the predictions.

4. We can determine the importance by evaluating the absolute value of the corresponding weights. The lager the absolute value is, the more important that statistic data is. Since the absolute value of weighs of put out is always close to zero in several predictions of LASSO Estimation based on different lamdas, therefore it is not very important. Because of the same reason, double play and assists are also not very important.