$|\beta_(t1)\beta_(t2)|$ I don't love In the plot of log(wavelength) vs flux, at some wavelength the point of different flux are very closed to each other, which means that these wavelengths flux have strong relationship with wavelength. Also the difference between max flux and min flux at each wavelength is descreasing first but then increasing. These suggest that ρ_w is neither smaller than the diffrence between two consecutive log(wavelength), which is 0.005, nor larger than the differnece between min and max of log(wavelength) in dataset, which is 0.2. So I assume ρ_w to be 0.05 With the same reason, ρ_t is smaller than 30 but larger than around 3. So I assume ρ_t to be 20
The other estimations are: $(\tau)^2$ is
0.0022, σ^2 is 0.0007, α is 1052, κ is 0.2, λ is 1.2 `what is the fuck`

`hahahahaha`

For those of you who want to know what we are about

$N(\mu,\sigma^2)$ Obviously, pointwise confidence interval is wider but the parameter wise confidence interval is more strict. The point wise confidence interval is achieved by finding the estimated sigma squared in the fit and constructing the confidence interval using t distribution. The parameter wise confidence interval is constructed by the equation of book (3.15); note that we can construct chisquared distribution for each β using the diagnal element of X^tX

Basically, I used three methods to tackle the problem, namely, linear regression, maximizing loglikelyhood and k nearest neighborhood method.

Method One: Optimizing loglikelihood

For optimizing loglikelyhood, we need to compute the loglikelyhood function and then optimize the function in order to attain the estimate of beta. In this situation, we denote digit Two as 0 and digit Three as 1. Then we can construct the likelyhood function based on binomial distribution and do optimization. It turns out the error rate for predicting test data of TWO is 0.0404, the error rate of predicting test data THREE is 0.048, and the total error rate is which is acceptable for me, and the general error rate is about 0.044. Besides, the error rate of predicting training data set is 0 for both numbers, which is perfect and reasonable since the likelyhood under optimized data is maximized to almost 1.

Method Two: Linear regression

I used the linear regression to fit the model without origin point since the image of both zipcode 2 and zipcode 3 have no information in the origin. After I got the fitted value of Y, we use the maximum fitted value of X from group two and the minimum of fitted value of X from group three as the threshold to classify the point. The training error for 2 is 0.055 the training error for 3 is 0.0015 and the total training error is around 0.028. On the other hand, the test error for digit two is 0.034 and the test error for three is 0.006; the total test error of linear prediction error is 0.08.

Method Three: k nearest neighbors

For k nearest neighborhood method, I used the built function knn in the class package to classify the data. It turns out that the training error for knn classifier is 0 when k is 1 and increases as k increases. When, k = 1, the test error of knn classifier is better than linear classifier since the error rate of testing 2 is 0.03, the error rate of testing 3 is 0.018 and the general error rate is 0.024 when k = 1. Similarly, the test error also increases as k increases, the further information is illustrated by the graph.

Problem 5

First of all, I plotted data and it turns out that I cannot really find out the linear relationship visually, so I decided to fit a naive regression model and find out the the leverage pts based on hat matrix, and it turns out that African elephant is the first leverage pt that I have discovered. Then I made the diagnostic plot of the naive fit, It turns out that human and Asian elephant are the other two leverage pts. Then I decided to remove these three points and look at the plot. Since there sort of a linear relationship, then I fitted the second linear model, but it turns out that the normality plot is not good. Thus, I decided to do transform the data in logscale and it turns out there is an obvious linear relationship based on scatter plot, and the diagnostic plot also indicates the linear fit is decent.

Problem One

(1)

Since SP500 index is calculated by the price of companies that are decided by the committee of StandardPoor using a linear relationship. However, the actual coefficients remain undecided. By the background knowledge, we can replicate the SP500 index by regression on the price of those companies. Since we want to build a sparse portfolio, we can use lasso to construct our sparse portfolio. In order to evaluate the quality of my model, firstly, I tested if I should use elastic model or pure lasso by using different alpha(0 ¡ alpha ¡= 1) to conduct regression. It turns out that the model is better when alpha is around 0.7. So I chose alpha to be 0.7.Then, I did cross validation and pick out the best lambda with samllest MSE; then I use that lambda to construct sparse portfolio. It turns out that my prediction of SP 500 index is very close to the actual value, which is verified by figure ”Best Lambda Fit” the lines(my prediction) matches perfectly with points(the actual value of SP500 index). As the result, we constructed a sparse portfolio where we use only the price of 58 companies to predict the sP500 index.

(2)

In this case, I divided the time range of stock into intervals where each one contains 60 days of record(The last one has only 22), then I constructed a function that figures out the sparse portfolio by lasso, guarantees the quality of model by cross-validation and returns the names as well as corresponding coefficients of companies in the portfolio. It turns out that our portfolio is not stable enough, for example, the number of companies included in the portfolio varies with time, which is illustrated in the figure with name "Number of Companies in Portfolio". Even if for the same company, we have different weights(coefficients) over time, which may increase the cost transaction when we actually put the model into application
In order to get a portfolio that changes little overtime, we can add penalty of $|\beta_t - \beta_s|$ and it means that we penalize the model with unstable coefficients. I believe it would help us to figure out the portfolio that changes little over time.

(3)

In this case, we can modify the penalty of lasso for a little bit. Since we don't want to have negative values, then we change the penalty of negative β to be infinity while leaving the positive β unchanged. The optimization problem (minimizing argmin$squarederror + lassopenalty$) would come up with the result with only positive βs.

(4)

Since the relationship between SP500 return and the price of every stock is no longer linear, then I expect that we may need to expand the number of stocks we include in the portfolio in order to have a better prediction. It turns we do need a larger portfolio, specifically, I need 403 companies in my portfolio of tracking SP500 returns while I only need 58 companies to track the SP500 index.
In terms of the stability of sparse model of tracking SP500 returns, our result is pretty stable since we include a large portion of companies in our sparse portfolio. In other words, we are more likely to include major representative companies of the stock market in most of time. Thus, the sparse portfolio for replicating returns is fairly stable compared with the sparse portfolio for replicating index.

(5)

Since we do not need to concern about transaction problems, we can use the price of all the companies on our list to predict SP500 index using linear regression. I divided my dataset into training and test dataset. It turns out the prediction is really good where the squared error is about 274.9754. Using the same training and testing dataset, I tried lasso, ridge regression and elastic-net model, but it turns out linear regression has the smallest test error.

Problem Three

For this problem, I used LDA, QDA, logistic model and logistic with lasso to analyze the data. I notice that the model could be better(we do not need all of the predictors to do classification) if we use step-wise or stage-wise selection to select predictors carefully. However, we are comparing the quality of classification method, as long as we use the same model to all of the method, we can have a relative comparison. In order to compare the quality of classifiers, I used two ways:
1). split the data into training set and test set and calculate the traning error and test error for every method. Compare training and test error.
2). figure out the false positive and false negative index among different classifiers.
In conclusion:
In terms of training error and test error(check plot "Training and Test Error"), it seems that LDA outweighs the other three methods and Logistic-lasso also have a good prediction behaviour. However, Logistic-lasso have the best false positive error(check the plot "False Positive False Negative") while QDA has the best false negative error. Back to the practical situation of diagnosing cancer, the cost of diagnosing a non-cancer patient as having cancer is much larger than diagnosing a cancer patient as not having cancer. Thus, we want to elminate false positive error. Hence, we are better off with logistic with lasso model.

First Part

It turns out we do need a larger portfolio, specifically, I need a large portion of companies in my portfolio in order to track SP500 returns while I only need 58 companies to track the SP500 index.

Stability

In terms of the stability of sparse model of tracking SP500 returns every 60 day, our result is not stable although we do require more companies compared with tracking the index. By the plot "Number of Compnaies in Portfolio SP500 Return",we still have a fluctuating sparse portfolio. Possibly we need to change the penalty in order to really fix the problem.

Introduction

Concrete is one of the most widely used artificial construction materials nowadays. Among all of the benchmarks evaluating the quality of a kind of concrete, people are more concerned about the compressive strength, which is determined by the indegredients and age of concretes. Thus, we want to build a regression model which can help us to predict the compressive strength of a concrete based on its composition and age. Specifically, we have 1030 records of lab data of conrete including the concrete's composition as well as age. The response variable is conceret compressive strength (MPa), and predictors includes cement, fly ash, water, coarse aggregate and so on.

Methods

First of all, it is indicated by the abstract of data that compressive strength is a highly non linear response, so I think it is pretty clear that if we use all of the predictors, we are not likely to have a perfect linear model to fit the data. Thus, I considered to use a sparse model, the intention is that high dimensional non linear points can be better fitted using linear method in a low dimension. For example, any non linear two dimensional data once projected to one dimension is just a line. The first thing I considered is lasso, but if I use the whole data set to construct a lasso regression model, none of the predictors will be dropped.Then I tried to conduct PCA and then use linear method on the projected data without dropping predictor. However, the bad screeplot as well as the complicated interepretation pushed me to turn to other methods. Hence, I decided to drop the predictor mannually before lasso since I notice there are three predictors with many 0 entries and then conduct lasso.

Secondly, the non linear relationship reminds me of the polynomial regression; the polynomial fit may have a better fit as well as prediction than normal linear method. Thus the second method I tried is to fit a polynomial regression on the data. But I am afraid that we may potentially add more noise to our model by using normal polynomial regression, then I decided to use a sparse polynomial regressionby combining polynomial regression with lasso.

Sparse Linear Model

Reasoning

After we read in all of the data, the first impression is that some of ingredients have a number of 0 input while the others does not. Thus we made a table called "zeros" to summarize this infromation. It turns out that Blast Furnace Slag, Fly Ash and Superplasticizer are the three ingredients containing many 0 values; thus, these three ingredients are more likely to be flexible in our model construction. In other words, we usually do not produce concrete without the other components but we could or possibly can build better concrete without Blast Furnace Slag, Fly Ash and Superplasticize. Hence, my intention is to sort of classify the concrete into 5 categories and use different sparse models to predict the compressive stength of different concerets.

Procedure

By the intention I mentioned before, I need to select different categories of data in order to conduct my sparse method.
By the correlation plot, we do not have very correlated predictors, so I did not go throught different combination of three selected predictors(the low correlation indicates that we should treat their effects independently) and only classfied the data into 5 categories:
Non zero: Eliminate the 0 values in predictors Blast Furnace Slag, Fly Ash and Superplasticizer and leave only non zero entries. I have 209 obervations, and I denote it as wo.zero(without zero).
All Three: I selected the data that does not contain any 0 values of Blast Furnace Slag, Fly Ash and Superplasticizer. I denoted it as wo.three.
With Blast Furnace Slag non zero: I selected the part containing all of the nonzero values of Blast Furnace Slag and dropped the Fly Ash and Superplasticizer. I denote it as wo.BFS.
Similarly, I selected the data wo.FA(Fly Ash) and wo.S(superplasticizer).
Now, I use the 5 data sets to do lasso regression respetively and figure out their fitted errors. Specifically, I wrote a function called sparse model that help me to fit a lasso model with the best lambda(based on leave one out cross validation)Then, I use the model constructed by one data set to predict the compressive strength of the other data set and calculate the prediction error. Finally, I plotted the error in the figure "Summary of Prediction and Fitted Errors" where the row name represents the training data set I used to construct the model and column name indicates the test data set I used to calculate prediction error.

Conclusion From Sparse Model

Basically, since I can not fit a consistently good sparse model, I used 5 sparse models based on different data set and compare their prediction error on the different kinds of data set I constructed. By my model construction, I manually drop predictors before I use lasso to do model construction.
Based on the plot, the diagnal entries are just fitted error since we use the training data set as the test data set.
If we want to predict the compressive strength of a concrete which includes all three components: Blast Furnace Slag, Fly Ash and Superplasticizer. I would use the model constructed with all three predictors by lasso since it has the samllest prediction error of 7.96 among all 5 models. If we want to predict the compressive strength of a concrete without component Superplasticizer, then I would use either the model without the three components or with all of the three components since they give prediction errors 4.35 and 4.37 respectively.
In conclusion, the linear sparse model can be generally

Polynomial Model With Lasso

Reasoning

Since the orginal data is said to have non linear relationship, I believe polynomial would be better than straight line in fitting a non linear relationship. Besides, I would like to see if I can find a generally better model so that I do not need to separate the data into different parts and use different models to fit the data.

Procedure

Basically, I constructed a function called make poly which helps me to design the data with specific n degree of design matrix required for polynomial regression. Then I put it into a lasso model and use the cross validated lambda to fit a polynomial sparse model. However, it is hard to decide which degree I should use. So I did a for loop to run over all degrees from 1 to 12(which is the maximum number I can use based on the warning output). Then I plotted the figure "Cross Validate Error Over Different Degree of Polynomials". It turns out degree of ten hits almost the bottom of the error(degree 12 is slightly better but it does not worth to add 16 predictors for a slightly better model).

Conclusion of Sparse Polynomial

After I fit a twn degree sparse polynomial regression, I have kept all of my predictors. It seems the model has not been over fitted yet even if I have 81 predictors in the model. However, I feel this is possible because I may need more predictors to linearly represent the data better. No matter sparse polynomial regression or lasso regression, all the method we used is limited in linear scenario. Although the polynomial fitting is better, I may need to find some other non linear methods for the data set from the perspective of better prediction.

Final Conclusion

There are several conclusions I have drawn by doing the project:
One: Linear method is not enough for us to deal with many practical problems where there are no obvious indications that linear method would be a good choice. If we used linear methods to solve non linear problem, we either need to reduce the dimension of our design matrix or increase the dimension of our design matrix in order to have a better fit.
Two: we need to consider prediction error and fitted error together in order to compare different methods. Some time a better fit could be an indicator that we are not able to have better prediction. For example, I tried the k nearest neighbor and did not propose it because we are able to have a great fit if k equals to the number of data points but the prediction is just horrible.

Three: Lasso can help us to do model selection but it is not able to find the best combination of predictors, especially from the perspective of prediction. If we can not drop any predictor from a linear model, then lasso is no better than a linear model fit.
Four: even if for the same data set, we may be able to have a better model fitting by classifying data into different parts and fit each part with slightly different model.