Predicting pH from SERS Data

Zhang Yichi
Supervisor: Prof. C.K.I. Williams

August 2014

1 Introduction

Blood tests are often used in health care to determine physiological and biochemical states of patients. But as we need to draw blood from patients first, it does take some time to get far results.

Recently, the chemists found that gold nanoshells can be used as intracellular sensors based on surface-enhanced Raman scattering (SERS) and these materials exhibit low toxicity to the cells of interest. This is a very good property since we can just put the nanoshell into the blood vessel of patients and measure redox potential or pH values of patients' blood instantly.

So when given the spectrum, the problem is how to predict the pH value. We have tried four regression methods, that is principal component regression (PCR), partial least squared regression (PLSR), lasso regression and kernel regression, on the data.

2 Data

There are 120 samples in the dataset from two chips. For each chip, there are five replications for twelve pH values, that is 60 samples. The spectrum is a 1044 dimension vector for each sample, and each dimension represents a Raman intensity for a given Raman shift.

There are two datasets we have got for experiments. The first dataset we used was produced with the order that pH value is increasing. We have found in the dataset that the intensity is lower and lower when the pH value is greater than 7 as shown in Figure 1.

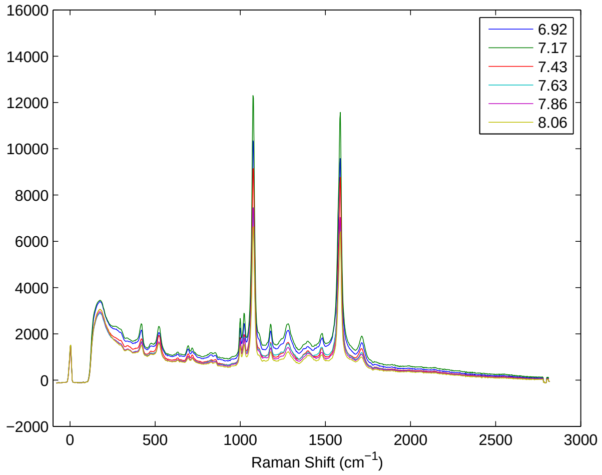

Figure 1: The mean spectrum for different pH values in the first dataset.

However, the chemists have told us that the intensity is lower and lower maybe due to the systematic loss of nanoparticles through time.

Thus, we got a another set of data with pH values measured in a randomized order produced by chemists. The experiments and analyses below are based on the randomized dataset.

2.1 Raw Data

As for the raw data, we plot the 5 replications in one plot for each pH values as shown in Figure 2.

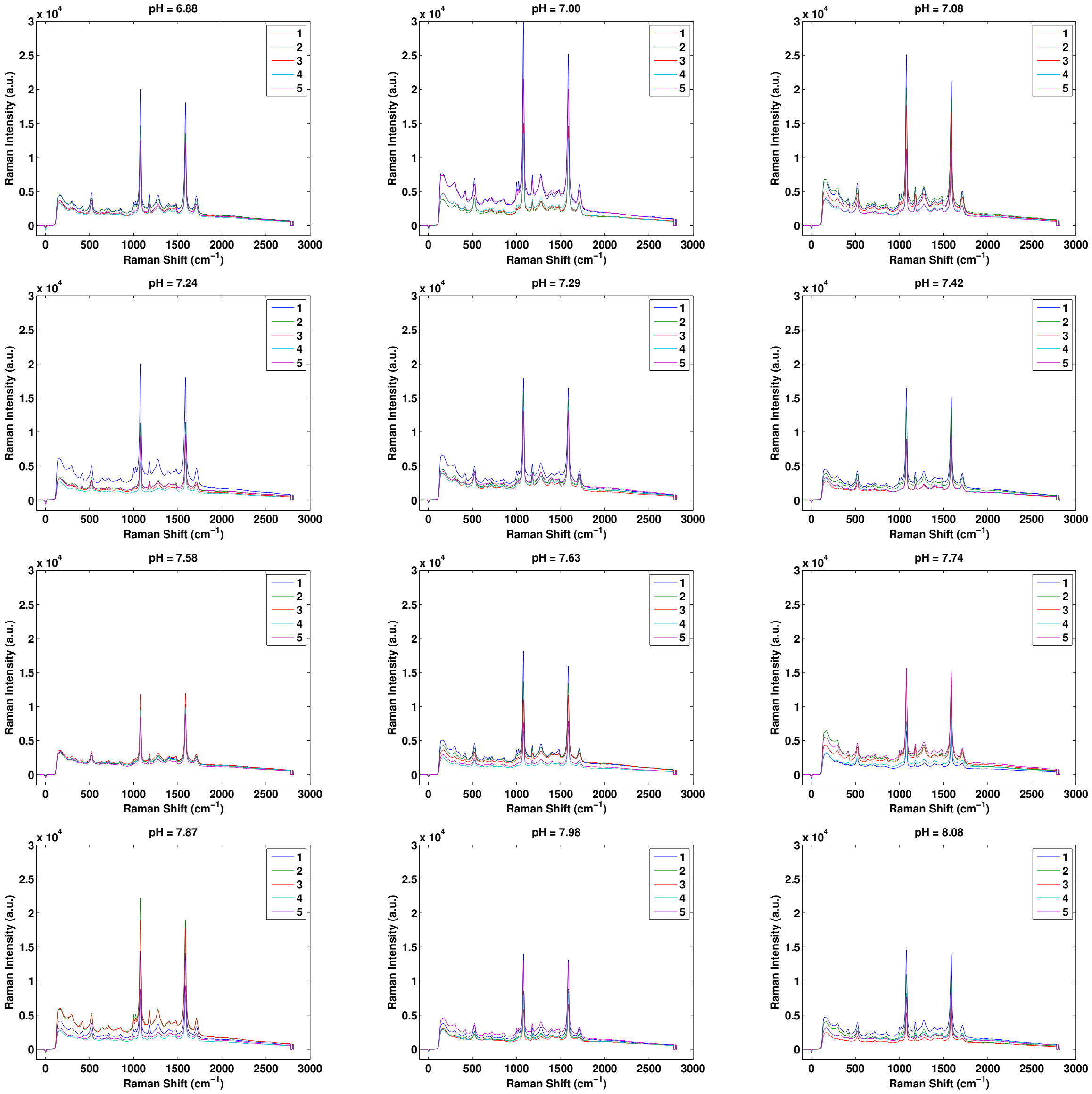

Figure 2: Plots for each pH values of raw data

We can find that when the pH value is increasing, not like in the previous dataset, the peak isn't lower and lower as the pH value is increasing all the time.

Meanwhile, we can find that for some pH values, the curves look dramatically different for the same pH value due to different nanoparticles densities on part of the experimental well. Suggested by chemists, normalization is a good way to avoid such variation. The method we choose to use is normalizing the total area under the curve.

2.2 Normalization

As mentioned in previous section, I have normalized the total area under the curve for every spectrum as shown in Figure 3.

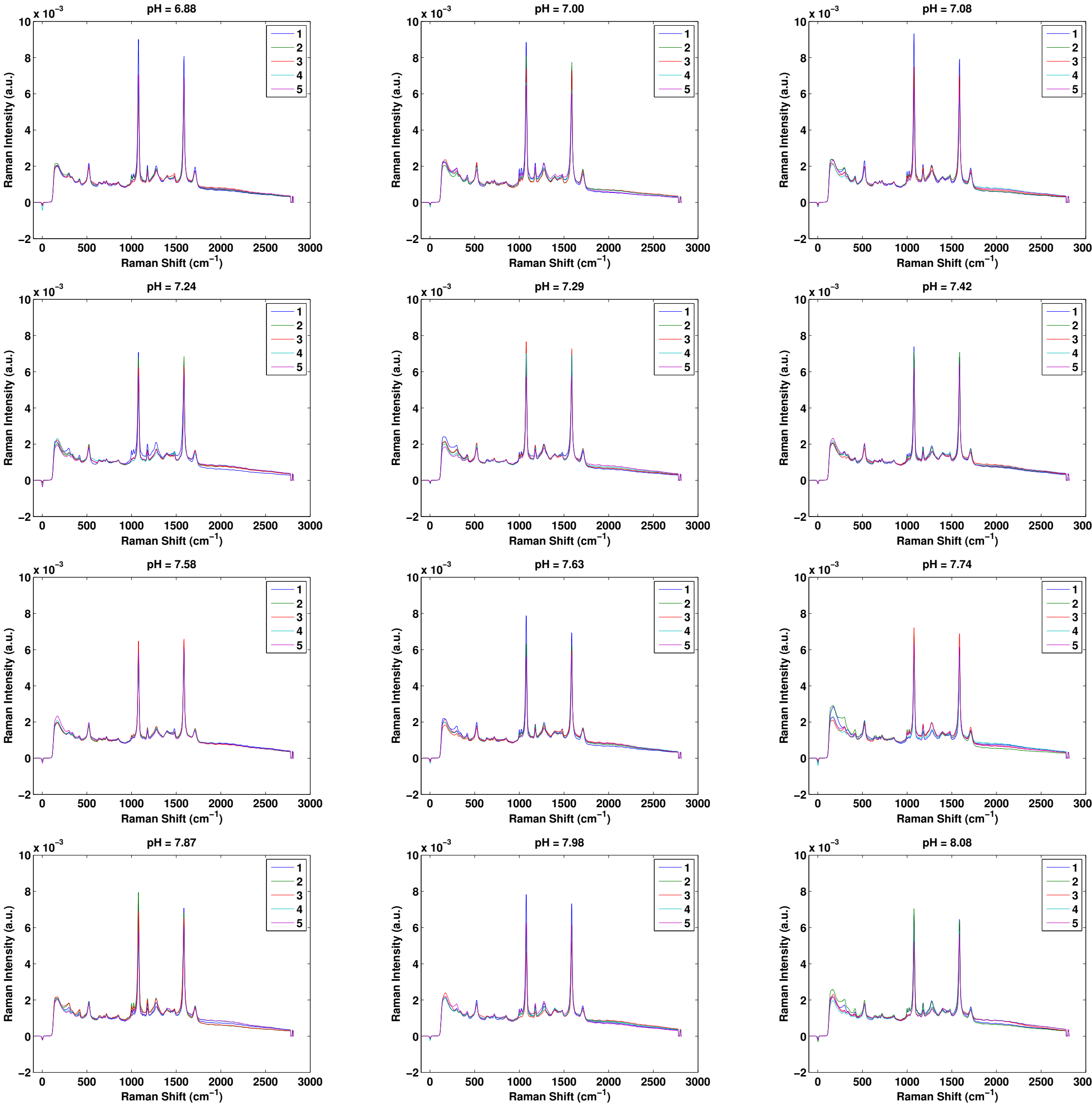

Figure 3: Plots for each pH values of data after normalization

As we can see from the figure, the normalization does eliminate the differences among samples of the same pH value.

3 Methods

There are only 60 samples for each chip. Since the number of sample is very small, we'll use cross validation to judge which method is better.

For cross validation, we divide samples into 5 folds. For each fold, no two samples with the same pH value. Every time, we use 4 folds for training and 1 fold for testing, and use standardized mean squared error (SMSE) and mean absolute error (MAE) to measure performance

$$\mathrm{SMSE} = \frac{\sum_{i=1}^{n}(y_i - \hat{y_i})^2}{\sum_{i=1}^{n}(y_i - \overline{y})^2}. \tag{1}$$

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y_i}| \tag{2}$$

Here, y_i represents the actual pH values of i-th spectrum in testing set, and $\bar{y}$ is mean of y_i. And $\hat{y_i}$ is the predicted pH values of i-th spectrum in testing set, and n is the size of testing set.

3.1 Linear Regression

The basic method of regression is linear regression. The simplest linear model is one that involves a linear combination of the spectrum

$$f(\mathbf{x}, \mathbf{v}) = v_0 + v_1 x_1 \ldots + v_D x_D, \tag{3}$$

where $\mathbf{x} = (x_1, \ldots, x_D)^T$ and here D is 1044 in our dataset. The key property of this model is that it is a linear function of the parameters $v_0, v_1, \ldots, v_D$.

However, it may be not possible for all the points representing all the spectra in the training set to be all on the same plane. So what we're going to do is to minimize the cost function

$$J(\mathbf{v}) = \sum_{i=1}^{m}(f(\mathbf{x}_i, \mathbf{v}) - y_i)^2, \tag{4}$$

which minimizes the total difference between the predicted pH value and the observed pH value. Here, $\mathbf{x}_i$ is a vector representing i-th spectrum in our dataset, and y_i is a scalar representing the pH value corresponding to i-th spectrum. This leads to a closed-form expression for the estimated value

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

which enables us to directly calculate the best $\mathbf{v}$ given the training data.

As the dimension of the spectrum is 1044 dimensions and the number of samples for one chip is only 60, it's not possible to directly use the method mentioned above, and there are two methods mentioned below which can resolve this issue.

3.1.1 Principal Component Regression

Principal component regression (PCR) is a regression analysis technique that is based on principal component analysis (PCA) [1].

PCA is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components. The number of principal components we used is less than the number of original variables. This transformation is defined in such a way that the first principal component has the largest possible variance, and each succeeding component in turn has the highest variance possible under the constraint that it is orthogonal to the preceding components.

Using the PCA first can make the dimension of spectrum less than 60, and then we can use traditional linear regression on the data.

3.1.2 Partial Least Squares Regression

Partial least squares regression (PLSR) [2] is a statistical method that has some relation to principal components regression. Instead of finding hyperplanes of minimum variance between the response and independent variables, it finds a linear regression model by projecting the predicted variables and the observable variables to a new space.

As PLS considers not only the spectra but also the pH values corresponding to them, PLSR has better performance in a lot of cases than PCR.

3.2 Lasso Regression

Lasso regression [3] is a regularized version of linear regression which can avoid over-fitting. It minimizes the cost function

$$J(\mathbf{v}) = \sum_{i=1}^{m} (f(\mathbf{x}_i, \mathbf{v}) - y_i)^2 + \alpha ||\mathbf{v}||. \tag{6}$$

Here, α is an important parameter to control the intensity of regularization. The larger α is, more values in vector $\mathbf{v}$ will be equal to 0 or nearly 0 for that the cost function is minimized.

3.3 Kernel Regression

Kernel regression [4] is quite a different method from the methods mentioned above.

Before introducing it, we'll introduce a method for classification called k-NN. In this method, for every new sample to be classified, we choose first k nearest samples for it and count which class most of the samples belong to. Normally, we choose Euclidean distance to calculate nearest samples.

And for regression, we cannot only count. We should combine the pH values of its neighbours together by

$$f_{\mathrm{KR}}(\mathbf{x}) = \frac{\sum\limits_{i=1}^{n} y_i k(\mathbf{x}, \mathbf{x}_i)}{\sum\limits_{i=1}^{n} k(\mathbf{x}, \mathbf{x}_i)}. \tag{7}$$

Here, $\mathbf{x}$ represents current spectrum and $\mathbf{x}_i$ represents the i-th spectrum in training set. We use Gaussian kernel for the weight of each pH values, and we can then predict the pH value for spectrum in testing set.

3.4 Gaussian Process Regression

Gaussian process [5] is a stochastic process whose realizations consist of random values associated with every point in a range of space such that each such random variable has a normal distribution.

Gaussian process regression is a very good way for regression. Not like the traditional linear regression, it is not related to some specific models, but can represent $f(\mathbf{x})$ obliquely.

4 Results

4.1 Principal Component Regression

To use PCR for regression, the first problem needed to be solved is determining how many principal components should be used. Before solving this problem, we can see the percent variance explained by the corresponding principal component in Figure 4.

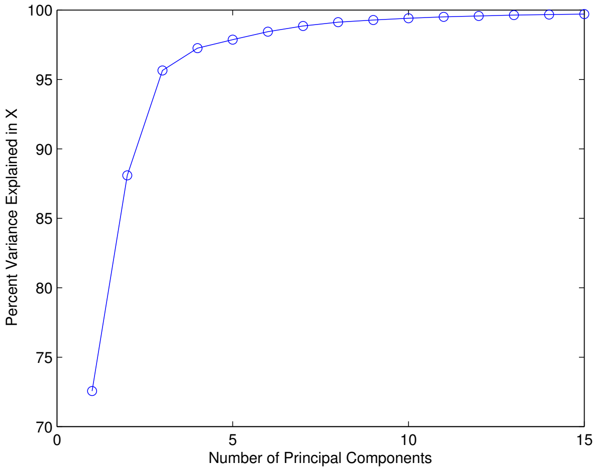

Figure 4: The percent variance explained by the corresponding principal component

We can see in the figure that when the number of principal components is greater than 3, the percent is greater than 95%. It is really enough to keep the feature of the dataset using only 3 principal components.

And to find the smallest SMSE, we choose to use the grid search. The result of grid search is shown in Figure 5.

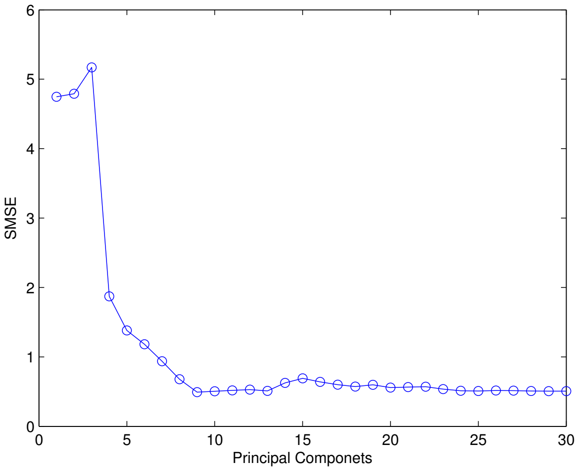

Figure 5: The result of grid search

We can find that the SMSE is smallest when the number of principal components is equal to 9. The result predicted by PCR with 9 principal components is shown in Figure 6. And the SMSE is 0.098476 and MAE is 0.090856.

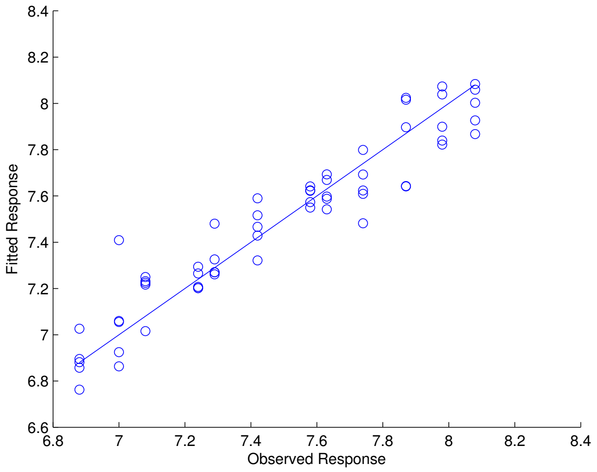

Figure 6: PCR with 9 principal components

The visualization of linear regression parameter $\mathbf{v}$ is shown in Figure 7.

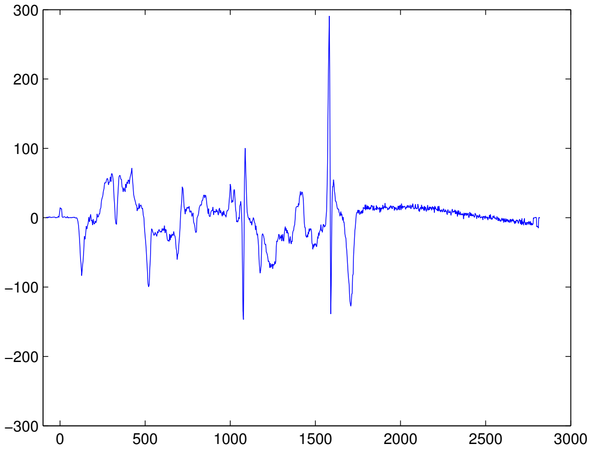

Figure 7: Plot of mean of $\mathbf{v}$ for PCR with 9 components

4.2 Partial Least Squares Regression

For PLSR, we also only consider the result on the data after normalization.

It's the same with PLSR that the first problem needed to be solved is determining how many principal components should be used. We can also see the percent variance explained by the corresponding principal component in Figure 8.

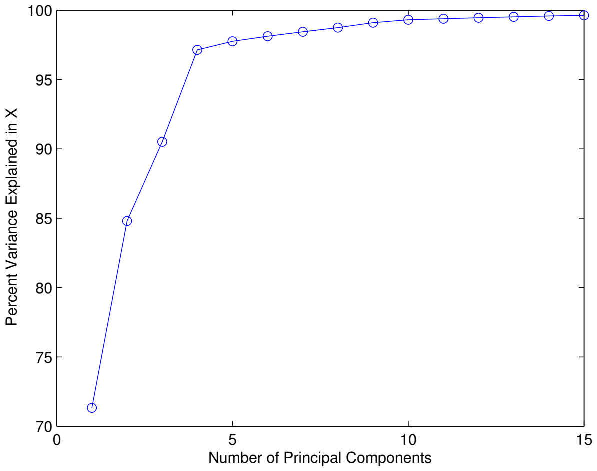

Figure 8: The percent variance explained by the corresponding principal component

We can see in the figure that when the number of principal components is greater than 4, the percent is greater than 95%. It is really enough to keep the feature of the dataset using only 4 principal components.

And we use the same method, grid search, to find the smallest SMSE. The result of grid search is shown in Figure 9.

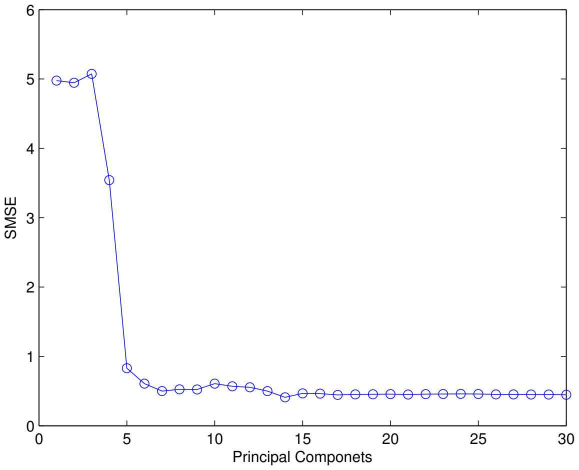

Figure 9: The result of grid search

We can find that the SMSE is smallest when the number of principal components is equal to 14. As we expected, this method has better performance than PCR, with SMSE 0.082091 and MAE 0.085736.

The result predicted by PLSR is shown in Figure 10.

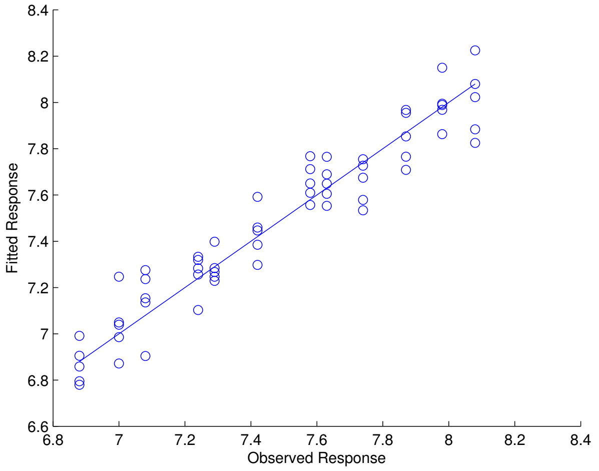

Figure 10: PLSR with 14 principal components

The visualization of linear regression parameter $\mathbf{v}$ is shown in Figure 11.

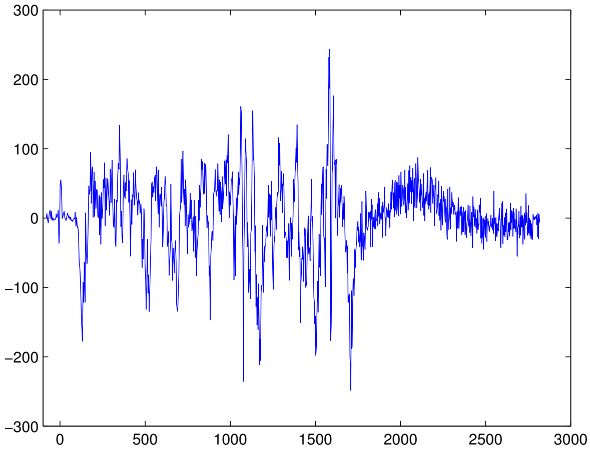

Figure 11: Plot of mean of $\mathbf{v}$ for PLSR with 14 components

As we can see, the curve is not that smooth like that of $\mathbf{v}$ for PCR and thus it's not easy to further analyse it. So we try lasso regression in the next section to make the curve smoother.

4.3 Lasso Regression

To make the curve smoother for further analysis and avoid the potential over-fitting problem, lasso regression is used to replace the linear regression in PCR and PLSR.

Figure 12 shows the change of SMSE as the parameter α of lasso regression is increasing.

As we can see in the figure, both the SMSE of PCR and PLSR are increasing. That means there are no over-fitting in the two regression methods before, and smoothing the curve of $\mathbf{v}$ is not a good way to improve the performance of those methods.

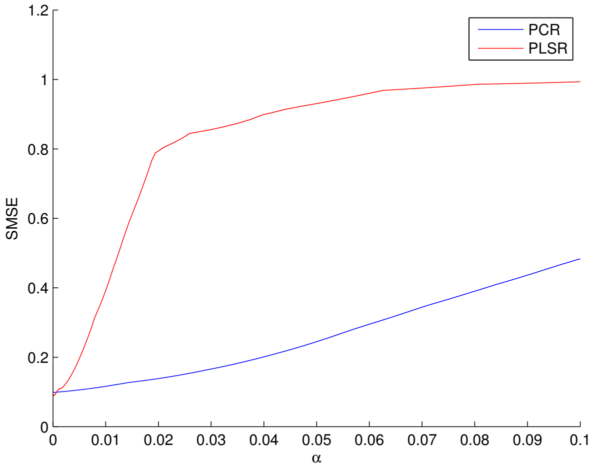

Figure 12: The relationship between SMSE and α

Moreover, we can find that the SMSE of PLSR increases much faster than that of PCR. As we know from the visualization of $\mathbf{v}$ shown before, the curve of $\mathbf{v}$ for PCR is much smoother than that of PLSR, so the $\alpha||\mathbf{v}||$ part of the cost function may not affect the performance of PCR that much as α is increasing.

4.4 Kernel Regression

To find the best α in kernel regression, we also use grid search to determine which σ should be chosen in Gaussian kernel.

Unfortunately, the performances are very poor no matter which σ is chosen. When I try support vector regression with rbf kernel on the dataset, the performance is also very poor. The linear kernel, however, has a very good performance on this dataset. That may mean non-linear regression model may not fit the data in this dataset.

4.5 Gaussian Process Regression

To use Gaussian process, we should determine two functions first, that is mean function and covariance function. And to do regression work, we should also determine which likelihood function is used and the inference method to find hyperparameters.

From the previous sections, we know that a linear mean function like

$$m(\mathbf{x}) = \sum_{i=1}^{n} a_i x_i + a_0 \tag{8}$$

may be more suitable for our dataset, so this function is the only one we have chosen for later experiments.

As for covariance function, two functions have been chosen for experiments. The first one is squared exponential covariance function

$$k(\mathbf{x}_p, \mathbf{x}_q) = \sigma_0^2 \exp\Big[-\frac{(\mathbf{x}_p - \mathbf{x}_q)' P^{-1} (\mathbf{x}_p - \mathbf{x}_q)}{2}\Big]. \tag{9}$$

Here, P is equal to λ times the unit matrix. And both σ_0 and λ are hyperparameters. The second one is Matern covariance function

$$k(\mathbf{x}_p, \mathbf{x}_q) = \sigma_0^2 f(\sqrt{d}r) \exp(-\sqrt{d}r). \tag{10}$$

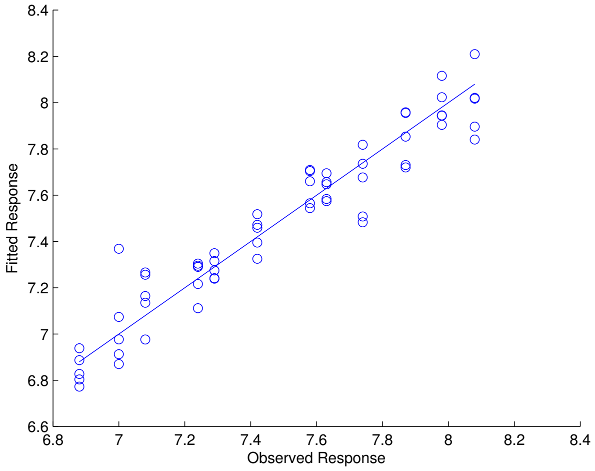

Figure 13: Result of Gaussian process with the best SMSE

Here, d is set to be 5 in our experiments, and function f is set to be $f(t) = \frac{1+t+t^2}{3}$. And r is the distance $\sqrt{(\mathbf{x}_p - \mathbf{x}_q)' P^{-1} (\mathbf{x}_p - \mathbf{x}_q)}$, while P is the same matrix as above. So it's the same that both σ_0 and λ are hyperparameters.

And since we're going to do regression work, we choose the traditional Gaussian function for likelihood function

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\frac{(x-\mu)^2}{2\sigma^2}\right]. \tag{11}$$

Here, μ is the mean and σ is the standard deviation. But only the standard deviation is in the hyperparameters.

And we have choose some different inference methods to get optimized hyperparameters. Table 1 shows the result for squared exponential covariance function, while Table 2 shows the result for Matern covariance function. Note that the two tables of results produced by experiments with the same mean function and likelihood function.

No.	GP01	GP02	GP03	GP04	GP05	GP06
Method	infExact	infEP	infLaplace	infVB	infKL	infLOO
SMSE	0.085254	0.084868	0.085691	0.084877	0.085814	0.089122
MAE	0.085065					0.088626

Table 1: Result for squared exponential covariance function

No.	GP07	GP08	GP09	GP10	GP11	GP12
Method	infExact	infEP	infLaplace	infVB	infKL	infLOO
SMSE	0.086177	0.084075	0.083821	0.083809	0.084479	0.089021
MAE	0.085909					0.088690

Table 2: Result for Matern covariance function

The result predicted by GP10 which has the best SMSE is shown in Figure 13.

References

[1] Ian Jolliffe. *Principal Component Analysis*. Wiley Online Library, 2005.

[2] Paul Geladi and Bruce R Kowalski. Partial least-squares regression: a tutorial. *Analytica Chimica Acta*, 185:1–17, 1986.

[3] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

[4] Elizbar A Nadaraya. On estimating regression. *Theory of Probability & Its Applications*, 9(1):141–142, 1964.

[5] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian processes for machine learning*. MIT Press, 2006.