Report on Project 1

Zhunxuan Wang, 13300180086
School of Mathematical Sciences

October 9, 2017

1 Linear Regression and Nonlinear Bases

1.1 Adding a Bias Variable

Based on the model in the original function `leastSquares`

$$\hat{\mathbf{y}} = \mathbf{X}\hat{\mathbf{w}},$$

and the minimization of the loss

$$\min_{\mathbf{w}} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2$$

of which the explicit solution [2] is

$$\hat{\mathbf{w}} = (\mathbf{X}^\intercal \mathbf{X})^{-1} \mathbf{X}^\intercal \mathbf{y}. \tag{1}$$

I inserted a vector with all ones in the first column of $\mathbf{X}$, then the first dimension of $\hat{\mathbf{w}}$ would be fitted as the bias term

$$\hat{\mathbf{y}} = \begin{pmatrix} \mathbf{1} & \mathbf{X} \end{pmatrix} \begin{pmatrix} \beta \\ \hat{\mathbf{w}} \end{pmatrix}$$

where β is the bias term. The exact solution is in the same form as equation 1. And the updated plot is shown as follows

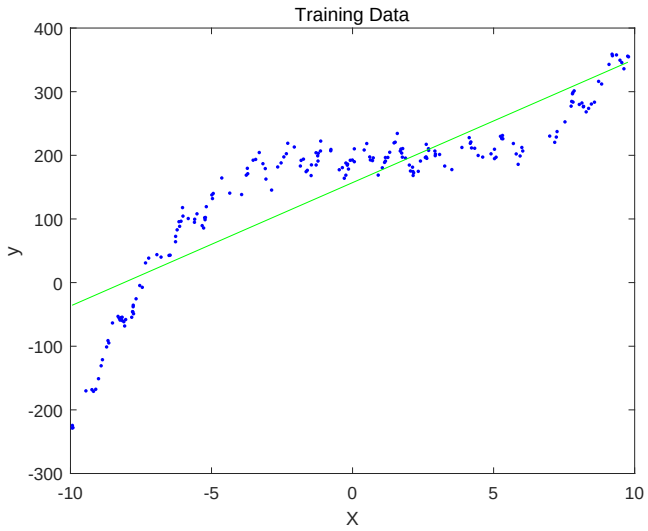

Figure 1: The Fitting (with Bias) Plot

The training error of the fitting above is 3551.35, and the test error is 3393.87.
The code of function `leastSquaresBias` is shown as follows

```
function [model] = leastSquaresBias(X, y)

% append a vector to the first column
X_app = [ones(size(X, 1), 1) X];

% calculate the estimated weight values
w_bias = (X_app' * X_app) \ X_app' * y;

% weight values with bias
model.w_bias = w_bias;
model.predict = @predict;

end

function [y_hat] = predict(model, X)

% append a vector to the first column
X_app = [ones(size(X, 1), 1) X];

% make prediction
y_hat = X_app * model.w_bias;

end
```

1.2 Polynomial Basis

Linear fitting does not work well when the complexity of the data is considerably high. Thus we consider fitting a polynomial model. E.g., when the degree of the polynomial is 3, this model will be

$$\hat{\mathbf{y}} = \mathbf{X}_{\text{poly}}\hat{\mathbf{w}}$$

where

$$\mathbf{X}_{\text{poly}} = \begin{bmatrix} 1 & x_1 & x_1^2 & x_1^3 \\ 1 & x_2 & x_2^2 & x_2^3 \\ \vdots & \vdots & \vdots & \vdots \\ 1 & x_n & x_n^2 & x_n^3 \end{bmatrix}.$$

The explicit solution would still be

$$\hat{\mathbf{w}} = \left(\mathbf{X}_{\text{poly}}^{\intercal}\mathbf{X}_{\text{poly}}\right)^{-1}\mathbf{X}_{\text{poly}}^{\intercal}\mathbf{y},$$

and the fitted curve is a cubic curve. It will be similar cases if the polynomial degree is greater.

The training error and the test error against `deg` are shown as follows

deg	0	1	2	3	4	5	6	7	8	9	10
training	15480.52	3551.35	2167.99	252.05	251.46	251.14	248.58	247.01	241.31	235.76	235.07
test	14390.76	3393.87	2480.73	242.80	242.13	239.54	246.01	242.89	245.97	259.30	256.30

And the smoothed log-error plot of the table above is shown as follows

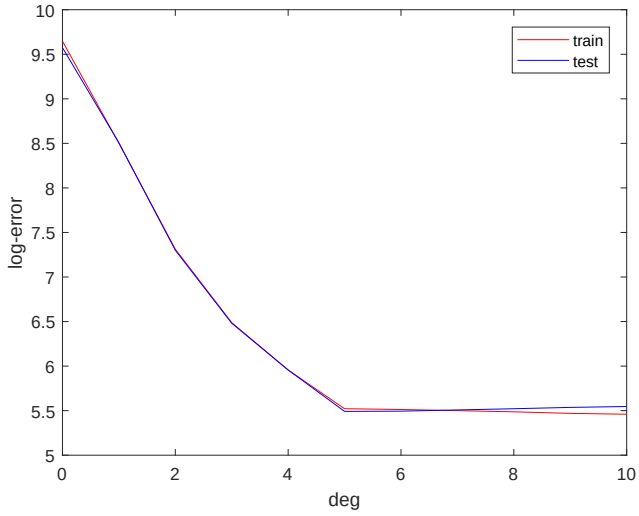

Figure 2: The Log-Error (against deg) Plot (smoothed)

As we observe from the table and the plot above, the errors showed a steep decreasing situation when `deg` was not greater than 3, then showed an unconspicuous down trend for training error, and down-up trend for test error (may be a trend of overfitting). The optimal `deg` for test error is 5, when the optimal test error is 239.54. The code is shown as follows

```
function [model] = leastSquaresBasis(X, y, deg)

% generate the polynomial-form matrix
X_app = zeros(size(X, 1), deg + 1);
for i = 0 : deg
  X_app(:, i + 1) = X .^ i;
end

% calculate the estimated weight values
w_bias = (X_app' * X_app) \ X_app' * y;

% weight values with bias
model.w_bias = w_bias;
model.deg = deg;
model.predict = @predict;

end

function [y_hat] = predict(model, X)

% generate the polynomial-form matrix
X_app = zeros(size(X, 1), model.deg + 1);
for i = 0 : model.deg
  X_app(:, i + 1) = X .^ i;
end

% make prediction
y_hat = X_app * model.w_bias;

end
```

2 Ridge Regression

For the ridge regression model, the hypothesis function is the same as linear regression

$$\hat{\mathbf{y}} = \mathbf{X}\hat{\mathbf{w}},$$

Taking the sum of the loss of LR and an L^2-regularization term as the loss function

$$J(\mathbf{X}, \mathbf{y};\ \mathbf{w}) = \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \delta^2\|\mathbf{w}\|_2^2,$$

hence

$$\hat{\mathbf{w}} = \arg\min_{\mathbf{w}} J(\mathbf{X}, \mathbf{y};\ \mathbf{w}).$$

We use matrix-form linear least squares method from [2]

$$\hat{\mathbf{w}} = \left(\mathbf{X}^\intercal\mathbf{X} + \delta^2\mathbf{E}\right)^{-1}\mathbf{X}^\intercal\mathbf{y}.$$

2.1 Preprocessing

We take the prostate cancer dataset, constructing a model to predict the 9-th variable a linear combination of the other 8. Before the model training, we shuffle the table, split the dataset to training and testing set, and take the standard score for each attribute of $\mathbf{X}$, and demean $\mathbf{y}$

$$(X_{\text{std}})_{i,j} = \frac{X_{i,j} - \bar{X}_j}{\sigma_j},\ (y_{\text{std}})_i = y_i - \bar{y},$$

where $\bar{X}_j$ and σ_j are the mean and the standard deviation of the j-th attribute in $\mathbf{X}$ respectively, $\bar{y}$ is the mean of $\mathbf{y}$.

2.2 Training and Testing

By tuning the regularization parameter δ^2 from 10^{-2} to 10^4 and training, we have the regularization path of each parameter and the training and test error against $\log_{10}\delta^2$ for ridge regression

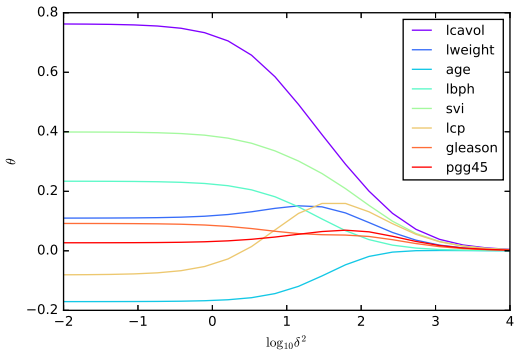

Figure 3: Regularization Path of Each Parameter

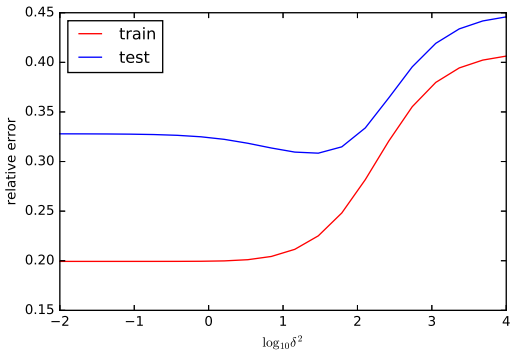

Figure 4: The Error Plot against $\log_{10}\delta^2$

As we observe from the plots above, the regularization path of each parameter shows a steady state when $\log_{10}\delta^2 < 0$ (so as the errors), and then a convergence trend to zero. The errors show a increasing trend when $\log_{10}\delta^2 > 1.5$, and the test error is permanently greater than the training error in this experiment.

2.3 Cross Validation

Applying the k-fold cross validation [1] (set $k = 10$) to choose the parameter

- Fix the regularization parameter δ^2.
- Randomly divide the dataset into k sets with similar size.
- Take each single set as the test set, and take the rest folds as the training set, iteratively.
- After k times of training, calculate the mean of test errors in the k times.
- Try other regularization parameter δ^2, and repeat the steps obtaining the mean of test errors.
- Criterion to choose δ^2: Choose the parameter with the minimum mean of test errors.

Tuning the regularization parameter δ^2 from 10^{-2} to 10^4 and training, we have the plot of means of test errors against $\log_{10} \delta^2$

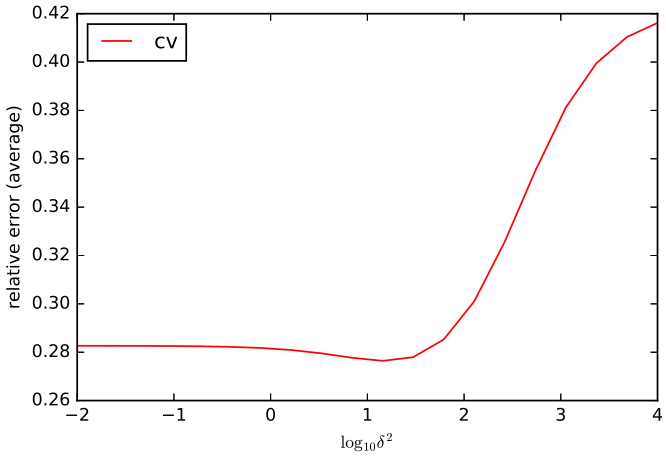

Figure 5: Mean relative error against $\log_{10} \delta^2$

On this dataset, based on the the criterion, optimal $\delta^2 = 10^{1.158}$, when the mean relative error is 0.276422.

References

[1] Ron Kohavi et al. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Ijcai*, volume 14, pages 1137–1145. Stanford, CA, 1995.

[2] Robert H Shumway and David S Stoffer. *Time series analysis and its applications: with R examples*. Springer Science & Business Media, 2010.