COMP6247(2020/21): Reinforcement and Online Learning Assessed Work: Recursive Least Squares

Yan Zhang (yz10u20@soton.ac.uk)

1 Linear Regression

1.1 Linear Regression estimated in closed form

Linear regression is a learning of the linear relationship of the target y and the explanatory variables. The learning problem could be expressed as

$$\hat{w} = arg\min_{w} \frac{1}{2} \parallel Xw - y \parallel^2 \tag{1}$$

By differentiating this expression w.r.t w, the closed-form solution could be obtained.

$$\hat{w} = (X^T X)^{-1} X^T y \tag{2}$$

Notice that the matrix $X^T X$ must be invertible, and this requires matrix X to have full rank and the number of samples(n) is much larger than the number of variables(p). However, when n ¡ p, the inverse is not unique, which is called under-constrained. Solving the inverse problem ($\hat{w} = (X^T X)^{-1} X^T y$) is said to be ill-posed, because $(X^T X)^{-1}$ doesn't exist. Thus, the data is chosen to have more observations than parameters. The result of the closed-form solution is shown in Figure 1(a). It has a good performance on the data that has a linear relationship.

1.2 Linear Regression using gradient descent

The goal of linear regression is trying to find a $\hat{y}$ that has a good estimation of the target y. And the difference between $\hat{y} = X\hat{w}$ and y is what needs to be minimized, where the loss function is

$$\ell = \frac{1}{n} \sum_{i=0}^{n} (y_i - \hat{y_i})^2 = \frac{1}{n} \sum_{i=0}^{n} (y_i - X\hat{w})^2 \tag{3}$$

By using gradient descent, w could be estimated by

$$w^{k+1} = w^k - \eta \nabla_w \ell = w^k - \eta X^T (y - Xw^k) \tag{4}$$

This process iterates over n times before the algorithm reaches convergence. The result of gradient descent is shown in Figure 1(b), and it reached convergence at iteration 80 with a low error of estimation.

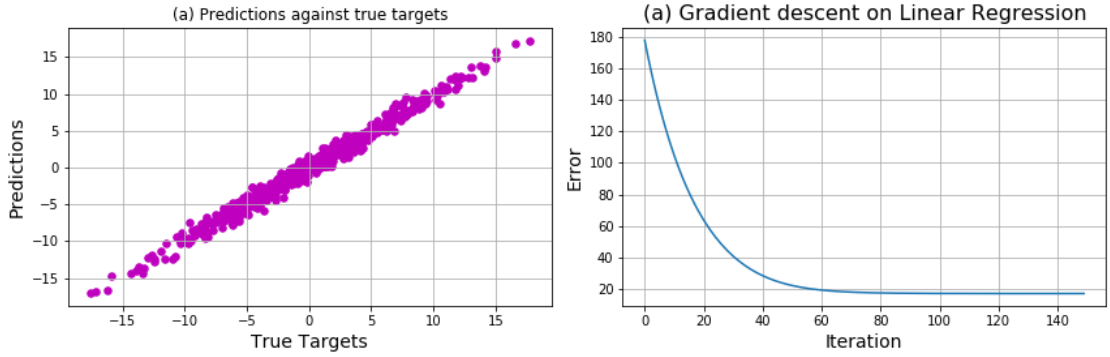

Figure 1: Linear regression estimated in closed form and using gradient descent

1.3 Linear regression using stochastic gradient descent

Stochastic gradient descent used another method to update the weights. Instead of training on the entire data set, SGD updates the weights sample by sample, which is

$$w^{k+1} = w^k - \eta(y_n - w^T x_n)x_n \tag{5}$$

The result of linear regression using SGD, where the learning rate was set as 0.05, is shown in Figure 2(a). It is evident that between iterations 250 and 2000, where we decide to stop will return answers that differ in error by about 20. The reason behind this is that the learning rate is too large so that SGD jumps too far and misses the area near the local minimum. This is the case of under-fitting. To solve the problem, one approach that could be used is to deploy an early-stopping method to the algorithm such that the learning will stop at around the 250th iteration. Alternatively, set the learning rate as 0.01, we will find the fluctuating trend in the learning curve disappeared and the learning converged at around the 500th iteration, which is shown in Figure 2(b).

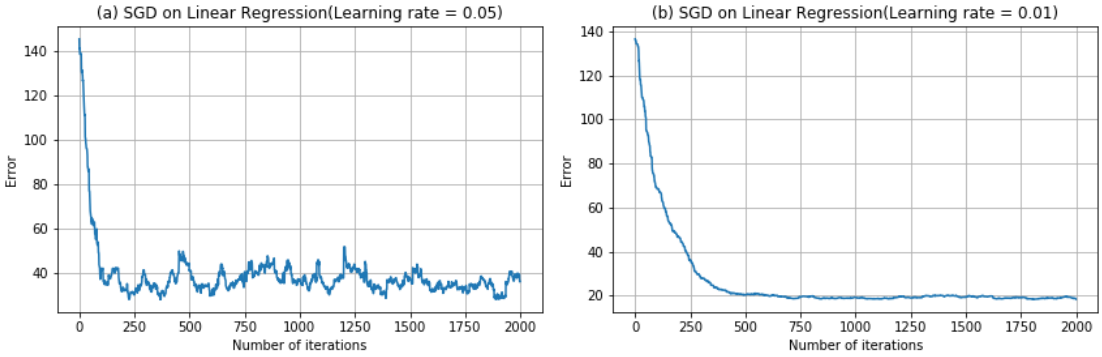

Figure 2: Learning curve of stochastic gradient descent with learning rate 0.05 and 0.01

2 Recursive Least Squares

In linear regression, the model is trained using all the data at one time. In this section, the data come one at a time instead of a bunch of data. Each time the data come, the weight is updated by performing linear regression using the new knowledge. To save computational cost, rank one update is applied to avoid inverse calculations, which is shown in (6). The inverse of $A + xx^T$ can the found without doing inverse calculations again.

$$(A + xx^T)^{-1} = A^{-1} - \frac{A^{-1}xx^T A^{-1}}{1 + x^T A^{-1} x} \tag{6}$$

As data arrive, what happened in the recent past is much more important than what happened in the distant past. Thus, the forget term λ is adapted to give exponentially less weight to older error samples. The smaller the λ is, the smaller the contribution of previous samples to the covariance matrix. In practice, λ is usually chosen between 0.98 and 1. In this report, λ is set as 0.99. The general update rule could be written as

$$w_n = w_{n-1} + k_n e(n) \tag{7}$$

where $k_n = \frac{\frac{1}{\lambda}P_{n-1}x_n}{1+\frac{1}{\lambda}x_n^T P_{n-1}x_n}$, $P_n = \frac{1}{\lambda}P_{n-1} - \frac{1}{\lambda}k_n x_n^T P_{n-1}$ and $e(n) = y_n - w_{n-1}^T x_n$.

The result of the self-implemented RLS applied on a linear data set(with dimension p=20, 50, 100, 200) is shown in Figure 3. It is evident that in the beginning, the error reduces sharply as a function of iteration and it fluctuates up and down as more data coming along. Also, It is found that the higher the dimension of the data is, the longer it will take before convergence.

3 Performance on UCI data set

To examine how well RLS can apply to a real-world problem, I download the computer hardware data set from UCI Machine Learning Repository and fit it into the RLS model. This data set has 6 response variables that measure machine cycle time,

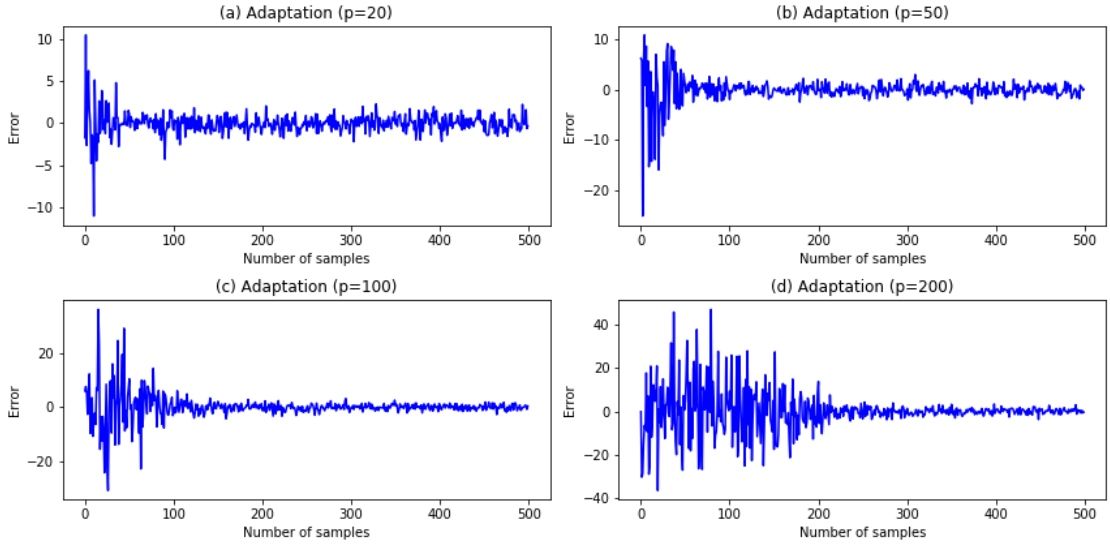

Figure 3: Relationship between the speed of convergence and the dimensionality of the problem

memory, and channels. These 6 variables are then used to form a linear approach to estimate the target(performance of computer). Initially, without doing any data preprocessing, the error function will go infinite after several iterations, which is a problem. This is because there is a vast difference in the range of each feature, these larger features will play a more decisive role when training the model, making the smaller features meaningless. Also, the vast difference between features and target makes the update of the weights huge and gradually boosts the weight to infinite. Therefore, I transform all features as well as the target by scaling each feature to [0,1] range. In such a way, the algorithm will converge much faster and avoid the boosting of the weights. After all the training process, the output could be inverse-transformed back into the original range.

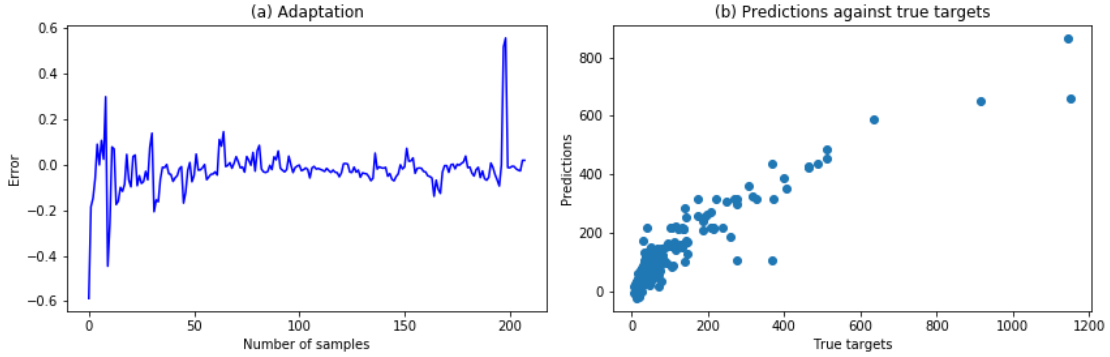

Figure 4: Performance of RLS on UCI Computer Hardware data set

Graphs (b) in Figures 4, 5, and 6 show how the predictions differ from the true target by using self-implemented RLS, SGD, and RLS from the pandasip package. Graphs (a) in Figures 4, 5, and 6 demonstrate the difference in the learning curve. A similar trend could be observed from these figures. RLS tends to have a sharp increase in the error in some iterations, which caused the outliers in (b). But generally, most predictions were made close to the true targets.

Table 1: Comparison between self-implemented RLS, SGD and RLS from pandasip package in terms of R-squared score

Method	R-squared score
Self-implemented RLS	0.82
SGD	0.73
Pandasip RLS	0.68

R^2, also named coefficient of determination, is a way of measuring how much variation of a dependent variable is explained

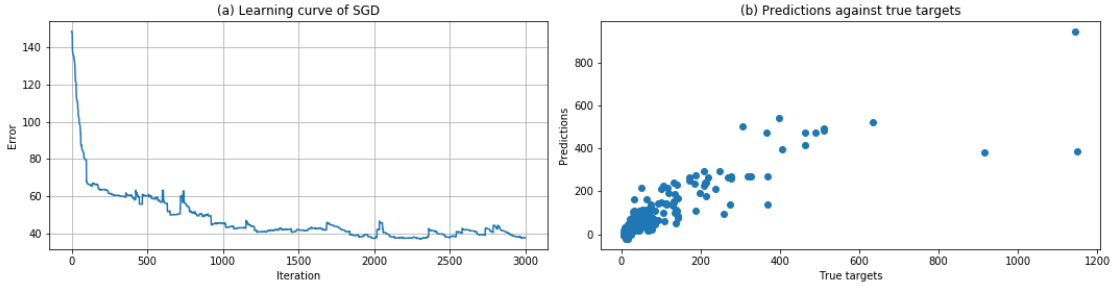

Figure 5: Performance of stochastic gradient descent on UCI Computer Hardware data set

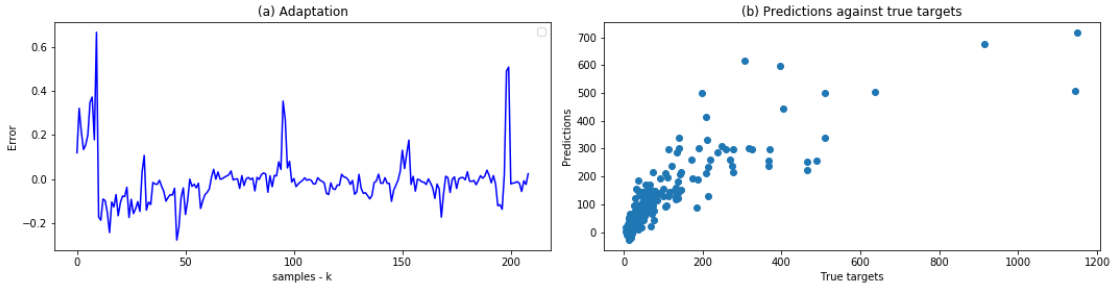

Figure 6: Performance of RLS from Pandasip toolbox on UCI Computer Hardware data set

by the independent variables in the regression model. In other words, it describes the proportion of the outputs that could be explained by the inputs. R^2 could be ranged from -1 to 1, and the closer it is to 1, the better the model is. The R^2 score of the three methods is shown in Table 1. It is found that the self-implemented RLS has the highest R^2 score (0.82), and RLS from Pandasip only reaches a score of 0.68

4 Recursive least squares filter

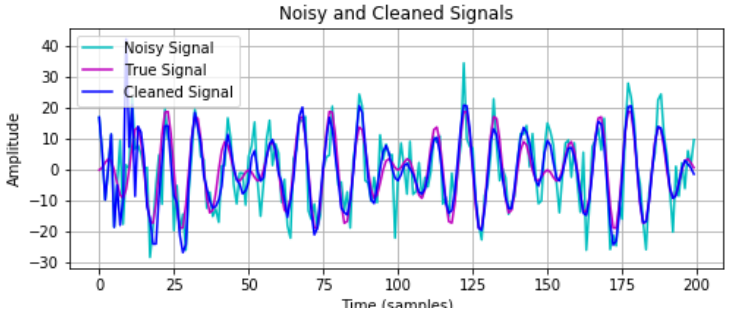

Figure 7: Adaptive noise cancellation using RLS filter

In this section, the RLS is used to adapt the weights of the filter, replacing LMS. Both LMS and RLS will lead to increased complexity and computational cost. But RLS will make convergence faster and the error smaller. The difference between LMS and RLS is that LMS used a gradient-based approach to perform the adaption, differently, RLS recursively finds the filter weights that minimized the weighted cost function by appropriately selecting the filter weights and updating the filter as new data arrive. With the forgetting term λ, RLS can attach more importance to the newly arrived data, and the older data could be de-emphasized.