COMP6247 Lab 1: Recursive Least Squares

Wei Chien Teoh (Eugene)
wct1c16@soton.ac.uk

22 February 2021

1 Introduction

This report presents to findings and results for Lab 1 of COMP6247 of University of Southampton [1]. The code implementation is stored in a Github repository [2].

2 Task 1

The task 1 of the exercise is to implement a simple linear regression on synthetic data solved with three different methods, namely: (i) closed form (pseudo-inverse); (ii) gradient descent (GD); (iii) stochastic gradient descent (SGD). The synthetic data generated involves a design matrix $X \in \mathbb{R}^{N \times p}$ and a set of weights $w_* \in \mathbb{R}^p$. In this example, N (number of sample) and p (number of features) are set to 500 and 30 respectively. The data are generated from a gaussian distribution $X, w_* \sim \mathcal{N}(0.5, 1)$ using Numpy. The target data y is defined as the product Xw_* added with noise also sampled from a gaussian distribution $\mathcal{N}(0, 1)$.

Figure 1 shows the predicted results against the true values of targets and weights. Although the variance of the plots are large, there exist a linear relationship between true values and the prediction. This means that the closed form solution successfully provided a solution for the linear regression. The result plots for GD and SGD are very similar to that of Figure 1, hence they will be excluded in this report.

Table 1 shows the pearson correlation of the above three methods of estimating the solution for linear regression. All three methods show promising and very close results. Thus, it can be concluded that all three methods successfully estimated the linear regression. Out of the three, the closed form solution performed the best.

The loss function of linear regression is convex, hence there exist a closed form solution. However, if the number of features p is large, the pseudo-inverse may be expensive to compute. The time complexity of matrix inversion using the Gaussian elimination is $\mathcal{O}(n^3)$, hence it does not scale with larger matrices.

Figure 2 shows the training error against the number of iterations for two variants of gradient descent, batch and stochastic. Figure 2a shows the batch gradient descent training with the learning rate set to 10^{-4} for 500 iterations. Figure 2b shows the stochastic gradient descent training with learning rate set to 0.005 for 5000 iterations.

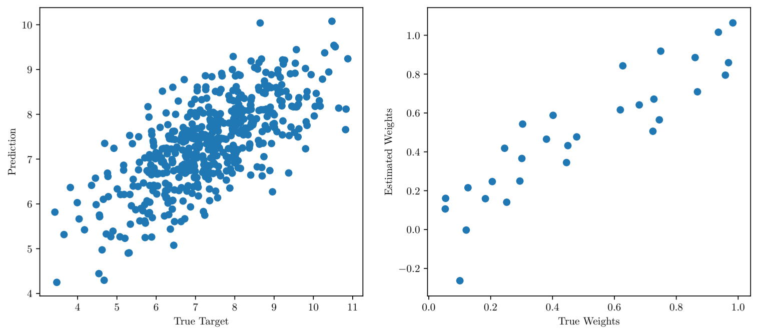


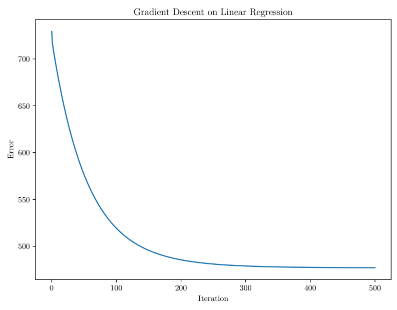
Figure 1: Results of Linear Regression estimated with closed form. (Left) true value of target against prediction. (Right) true values of weights against estimated weights.

Table 1: Pearson correlation of predicted values against ground truth.

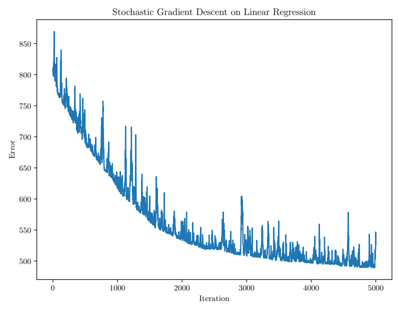
	Target	Weights
Ground Truth	1.000000	1.000000
Closed Form	0.704550	0.901110
GD	0.704474	0.901734
SGD	0.695540	0.911628

With batch gradient descent, it requires less iterations than stochastic gradient descent to achieve the convergence of the loss function. It also shows a smoother convergence with less fluctuation of error. However, as the number of data samples N gets larger, the derivative of the loss function with respect to the weights, $\nabla_{\boldsymbol{w}} E$ will be more expensive to compute.

Stochastic gradient descent was proposed to provide a scalable solution with large N. Figure 2b show the training error across the number of iterations. SGD provides similar performance to batch gradient descent, but with much more fluctuation. If the training is stopped at a training iteration with high fluctuation, it will cause the error at that iteration to be large. Figure 3 shows visual intuition of gradient descent updates close the convergence of loss function. The fluctuations of Figure 2b are caused by oscillations between the valley of the convex loss function.



(a) Batch Training



(b) Stochastic Training

Figure 2: Gradient Descent Training of Linear Regression.

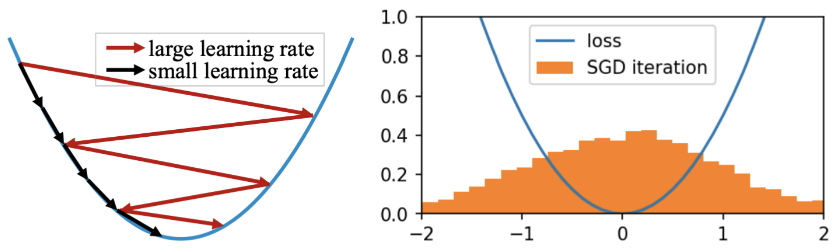


Figure 3: (Left) Effects of magnitude of learning rate on gradient descent update. (Right) Distribution of SGD loss convergence. (Sourced from [3])

3 Task 2

4 Task 3

5 Task 4

References

[1] Mahesan Niranjan. *COMP6247(2020/21): Reinforcement and Online Learning, Recursive Least Squares*. School of Electronics and Computer Science, University of Southampton.

[2] Eugene Teoh. *COMP6247 Labs*. 22 February 2021. URL: `https://github.com/eugeneteoh/COMP6247-Labs`.

[3] Kangqiao Liu, Liu Ziyin, and Masahito Ueda. *Noise and Fluctuation of Finite Learning Rate Stochastic Gradient Descent*. 2021. arXiv: `2012.03636 [stat.ML]`.