Investigation on House Price Prediction with Various Gradient Descent Methods

Yan SUN1,*
1University of Sydney Darlington Campus, Sydney, New South Wales 2006, Australia
*ysun0827@uni.sydney.edu.au

Abstract. As a branch of artificial intelligence, machine learning has received a lot of attention and has become a popular research field. It connects the huge database obtained from daily life with data prediction, which is of great help in decision making. In this project, we apply linear regression models and use different gradient descent methods, including general, mini-batch, and stochastic gradient descent, to optimize the error. We used convergence rate and accuracy as evaluation criteria, which will be explained via run time of convergence and mean square error respectively. For the regression models, both single-feature and multi-feature hypothesis models are investigated. In consideration of the risks of multicollinearity and overfitting, only simple linear model and its exponential transformation are adopted. The result indicates that mini-batch gradient descent has the biggest comprehensive advantage.

1. Introduction
The level of correlation between house prices and other factors is a popular and practical research topic. Constructing an accurate model for their relationship with minimized error is meaningful for daily life as the housing market is still one of the most essential activities in economic transactions. In this project, we use the House-price data set from Python scikit which contains 506 instances [1]. There are 13 features, representing the conditions of the house itself and the surrounding environment, that may influence the house price, 12 of which are continuous value and one of which is a categorical variable. We mainly focus on the relationship between house price and numeric features, assuming they are independent of each other. Table 1 shows the detail with X1 to X12 as predictors and Y as the dependent variable.

Table 1. 12 elements that influence house price

| Mathematical representation                                      | Input variables                                      |
|-----------------------------------------------------------------|------------------------------------------------------|
| X1                                                                | per capita crime rate by town                         |
| X2                                                                | proportion of residential land zoned for lots over 25,000 sq.ft. |
| X3                                                                | proportion of non-retail business acres per town      |
| X4                                                                | nitric oxides concentration (parts per 10 million)    |
| X5                                                                | average number of rooms per dwelling                  |
| X6                                                                | proportion of owner-occupied units built prior to 1940 |
| X7                                                                | weighted distances to five Boston employment centres  |
| X8                                                                | index of accessibility to radial highways             |
2. Methods

2.1 Algorithm

For the loss function, here we use mean squared error (MSE) to evaluate the difference between the actual value in the dataset and the expected value predicted by the pre-assumed equation. MSE is one of the most common functions in machine learning. The error indicates how close the fitted parameters are to the true value. As the name tells, it is the average squared distance between the predicted value and the true value [2].

\[ \text{MSE} = \frac{1}{n} \sum_{i=1}^{n} \left( y_i - (a \times x_i + b) \right)^2 \]  

(1)

The goal of the gradient descent algorithm is to optimize the function and get the minimum value. We calculate the partial gradient and get the ideal parameters by adjusting the learning rate to minimize the error. The learning rate is a parameter to decide the step size that the parameters are moving towards a better solution. In this project, three types of gradient descents will be discussed, which are general gradient descent, stochastic gradient descent, and mini-batch gradient descent. Although there are many differences between these three algorithms, the core idea is the same, which is to use the equation below to do iteration and minimize the error function:

\[ \theta_{new} = \theta_{old} - \alpha \times \frac{\partial F(\theta_0, \theta_1, \theta_2,...)}{\partial \theta} \]  

(2)

where \( \theta \) stands for parameters, \( \alpha \) stands for learning rate, and \( F \) stands for the loss function.

For the general gradient descent, the whole data will be used to compute the gradients at once in each iteration. Compared to the other two methods, general gradient descent has the most robust learning and gradient convergence. Therefore, it also has the smoothest convergence curve. Besides, there is only one loop and vectorization involved in the whole process. However, it is the slowest one when the data size is big.

For the mini-batch gradient descent, in each iteration, the dataset will be shuffled to get differently ordered data. Then the dataset will be divided into same-size-batches, and the computation will be implemented in batches. Since data is shuffled in each iteration, mini-batch gradient descent has a fluctuated curve [3].

For the stochastic gradient descent, in each iteration, the dataset will be shuffled to get differently ordered data. Then gradient computation will be implemented on each sample. Since each sample is randomly chosen, the error values can be very different, resulting in a fluctuated curve with a converging trend [4]. It can be regarded as a special case of mini-batch with batch size equals 1.

With shuffled data and batches, mini-batch gradient descent can be considered as the intermediate method between general and stochastic gradient descent with advantages of both. It is efficient with large data. Similar to stochastic gradient descent it prevents gradient descent from stuck in the local minimum while has more relatively robust gradient descent convergence.

Stochastic gradient descent helps avoid local minimum due to the oscillation [5]. For the same reason, it is usually used in higher-dimensional data. Therefore, it does not have significant advantages in low-dimensional data and will not be used in the univariate model. Besides, compared to general gradient
descent, it is more efficient but takes longer to achieve gradient and error convergence. Also, it does not benefit from vectorization since there is only one point in each epoch.

2.2 Process

The very first step is model and predictors selection. There are four models fitted in this project. For the first two models, we focus on the single relationship between the house price (y) and the percentage of the lower status population (x12). We construct 12 scatter plots between the house price and each variable and the one with x12 shows the most significant linear or exponential tendency. Therefore, x12 is chosen to be the predictor and simple linear regression and exponential regression are adopted for these first two univariate models. For the exponential regression model, it is the same as the former model except we take logarithmic transformation on the value of house price.

For other predictors, x1, x5, x6 also more or less shows a linear or exponential tendency with house price. Therefore, for the other two models, we still use a simple linear regression model and an exponential regression model. And we choose x1, x5, x6, and x12 as predictors.

Meanwhile, to check whether these four predictors are more correlated to the dependent variable, the coefficient of determination is constructed to obtain the R² value. It is a common measurement to study the correlation between variables, ranging from 0 to 1. The closer the R square value is to 1, the stronger the correlation is [6].

\[ R^2 = 1 - \frac{SS_{\text{res}}}{SS_{\text{tot}}} = 1 - \frac{\sum (y_i - \hat{y}_i)^2}{\sum (y_i - \bar{y})^2} = 1 - \frac{\sum e_i^2}{\sum (y_i - \bar{y})^2} \] (3)

\[ SS_{\text{res}} \] and \( \sum e_i^2 \) stand for the residual sum of squares; \( SS_{\text{tot}} \) stands for the total sum of squares; \( y_i \) stands for each dependent variable in the dataset; \( \bar{y} \) stands for the mean value of the dependent variable and \( \hat{y}_i \) stands for each fitted value of \( y_i \).

As Table 1 shows, the R² values of the four chosen predictors are relatively higher than others, which support our choice according to the scatter plots. Although \( X_3 \) has a higher R² value than \( X_1 \) and \( X_6 \), we can tell from figure-5 that the relationship between it and \( Y \) is clearly not linear and does not apply to our models. Therefore, we have \( X_1, X_5, X_6, X_{12} \) as the predictors of the third and the fourth multivariate model.
After selecting models and independent variables, we apply general gradient descent to the first two univariate models and all three gradient descents to the last two multivariate models. For the general gradient descent, the whole process only needs to be run once, since there is no random selection involved. We always get the same MSE value. However, for the mini-batch gradient descent and the stochastic gradient descent, a mean MSE error value is required since the data is randomly ordered in each iteration. Hence 10 runs are implemented for each algorithm. We record the running time and MSE error of each operation and compute the mean running time and error.
3. Results and Discussion

Table 3. Comparison between two models for $x_{12}$ and $y$

|          | Model 1                              | Model 2                              |
|----------|--------------------------------------|--------------------------------------|
|          | $y = 0.948 * x_{12} + 34.528$        | $\log(y) = -0.046 * x_{12} + 3.614$ |
| Running Time (sec) | 148.1                               | 45.6                                 |
| Mean Squared Error  | 38.48                               | 33.58                                |

Figure 3. Learning curves of model 1 (a) and model 2 (b) for general gradient descent, x-axis represents iteration, y-axis represents MSE.

Table 4. Comparison between two models for $x_1, x_5, x_6, x_{12}$ and $y$

|          | Model 3                              | Model 4                              |
|----------|--------------------------------------|--------------------------------------|
|          | $y = a * x_1 + b * x_5 + c * x_6 + d * x_{12}$ | $\log(y) = a * x_1 + b * x_5 + c * x_6 + d * x_{12}$ |
| Running Time (sec) | 110.8                               | 109.3                                |
| Mean Squared Error  | 29.87                                | 38.12                                |
| General Gradient Descent | 50.3                                | 57.0                                 |
| Mean Squared Error  | $1.81 \pm 6.43$                     | $0.07 \pm 0.72$                     |
| Stochastic Gradient Descent | 41.6                                | 41.0                                 |
| Mean Squared Error  | $32.99 \pm 0.01$                    | $0.09 \pm 0.01$                     |
| Mini-Batch Gradient Descent |                                    |                                       |
Figure 4. Learning curves of model 3 (a) and model 4 (b) for general gradient descent, x-axis represents iteration, y-axis represents MSE

Figure 5. Learning curves of model 3 (a) and model 4 (b) for stochastic gradient descent, x-axis represents iteration, y-axis represents the mean MSE value of 10 runs

Figure 6. Learning curves of model 3 (a) and model 4 (b) for mini-batch gradient descent, x-axis represents iteration, y-axis represents the mean MSE value of 10 runs

It is clear that the stochastic gradient descent of model 4 gives us the best result among all the attempted models in terms of running time and total error. The stochastic model guarantees us that the model is both taking more than one variable, which improves the total $R^2$ value and also makes sure that the minima that we are predicting is actually a global minimum by randomizing from a greater set. Thus, the final predicted model for our study is:
\[
\log(Y) = -0.16 \cdot X_1 + 0.48 \cdot X_2 - 0.0008 \cdot X_5 - 0.005 \cdot X_{12}
\] (4)

Besides, we can also tell from the table above that mini-batch gradient descent has a comprehensive advantage on efficiency and error, while general gradient descent has the smoothest convergence curve and the lowest efficiency, which corresponds to their features which are mentioned before. However, the superiority on stochastic and mini-batch gradient descent might be more obvious on higher dimensional data, especially for mini-batch gradient descent.

We have also attempted using all 13 variables to fit the regression model, but there is a serious multicollinearity problem, which means that more than one variable is correlated to each other and mask away from the true regression model [7-10].

4. Conclusions
The result of this investigation suggests that the best-fitted model will be model 4, the exponential transformation of the multi-feature model with the conditions consisting percentage of lower status population, crime rate, average number of rooms per dwelling and proportion of owner-occupied units built prior to 1940.

Moreover, by comparing the mean square error and run, stochastic gradient descent reaches the relatively greatest precision within the shortest time, however, with very oscillating convergence because of the small batch size, while mini-batch gradient descent has the most comprehensive advantage.

During the algorithm implementation, we encounter multicollinearity when fitting models with more predictors. We tried to eliminate the multicollinearity with regularization but it did not work out. A possible reason is limited data size. A possible solution is to use truncated gradient method. We leave this problem as our future work. Besides, for the stochastic gradient descent, we can use some gradient descent optimization algorithms to achieve better convergence.

References
[1] Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.
[2] Allen, David M. "Mean square error of prediction as a criterion for selecting variables." Technometrics 13.3 (1971): 469-475.
[3] Hinton, Geoffrey, Nitish Srivastava, and Kevin Swersky. "Neural networks for machine learning lecture 6a overview of mini-batch gradient descent." Cited on 14.8 (2012).
[4] Bottou, Léon. "Stochastic gradient learning in neural networks." Proceedings of Neuro-Nimes 91.8 (1991): 12.
[5] Bottou L. (2012) Stochastic Gradient Descent Tricks. In: Montavon G., Orr G.B., Müller KR. (eds) Neural Networks: Tricks of the Trade. Lecture Notes in Computer Science, vol 7700. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-35289-8_25
[6] Miles, Jeremy. "R squared, adjusted R squared." Wiley StatsRef: Statistics Reference Online (2014).
[7] Alin, Aylin. "Multicollinearity." Wiley Interdisciplinary Reviews: Computational Statistics 2.3 (2010): 370-374.
[8] Melkumova, L. E., and S. Ya Shatskikh. "Comparing Ridge and LASSO estimators for data analysis." Procedia engineering 201 (2017): 746-755.
[9] Langford, J. "Sparse Online Learning via Truncated Gradient." Advances in Neural Information Processing Systems 21 - Proceedings of the 2008 Conference. N.p., 2009. 905–912. Web.
[10] Ruder, Sebastian. "An overview of gradient descent optimization algorithms." arXiv preprint arXiv:1609.04747 (2016).