Fall 2023
MATH 309 Continuous Optimization Project Report:
Stochastic Gradient Descent

by Hannah Nadnaden, Nada Elsayed, Sherlyn Ng

1 Introduction

In machine learning, optimal model parameters are a fundamental requirement for achieving predictive accuracy and efficiency. As datasets expand in size and complexity, there is a greater need for utilizing more efficient optimization methods. While traditional approaches, such as the steepest descent (SD) method, prove effective on smaller scales, they often struggle to scale seamlessly to the dimensions demanded by large datasets. This project report investigates the application of stochastic gradient descent (SGD) and SD methods, focusing on predicting housing prices through multiple linear regression on a reasonably sized dataset.

The motivation for employing SGD in our learning problem is explored by comparing its features and performance with the SD method. The report outlines the mathematical foundations of both methods, the implementation details within our learning problem, and the experimental setup for both optimization methods. The chosen learning problem for this experiment involves predicting housing prices based on training datasets. The results of the implemented methods will be thoroughly analyzed and compared. The insights gained from these results will be crucial for understanding optimization techniques in machine learning and addressing any challenges that may arise.

2 Methodology

2.1 The Learning Problem

We employ the task of predicting housing prices through multiple linear regression as a means to illustrate the performance between SGD and the SD method. The dataset for this task encompasses 546 observations and incorporates 13 variables. Among these variables, six are numerical—denoted as x_1 to x_6. Additionally, the remaining seven variables are categorical, identified as x_7 to x_{12} introduced below.

- x_1 = price of the house
- x_2 = area of the house
- x_3 = number of house bedrooms
- x_4 = number of house bathrooms
- x_5 = number of house stories
- x_6 = number of house parking spots
- x_7 = does it have a basement (Yes/No)
- x_8 = does it have a hotwater heater (Yes/No)
- x_9 = does it have airconditioning? (Yes/No)
- x_{10} = is it connected to the main road? (Yes/No)
- x_{11} = house in a preferred neighborhood of the city (Yes/No)
- x_{12} = any guestroom offered? (Yes/No)

Thus, the model can expressed as a multiple linear regression, formulated as:

$$Y_i = \beta_0 + \beta_1 x_{2i} + \beta_2 x_{3i} + ... + \beta_{12} x_{12i} + \epsilon_i$$

where ϵ_i is the error term, assumed to follow a normal distribution with a mean of 0 and variance σ^2, Y_i is the response variable for the housing prices and x_{ki} are the features for the i-th house, $k = 2, ..., 12$.

2.2 Defining Objective Function

In the context of multiple linear regression for housing price prediction, the minimization problem involves finding the optimal parameters that minimize the difference between the predicted house prices and the actual prices in the dataset. The objective function for linear regression is typically the mean squared error (MSE) or the sum of squared errors, which represents the difference between predicted and actual prices for all the observations in the dataset.

For a dataset with n = 546 observations and m = 11 features, the objective function for the multiple linear regression can be expressed as:

$$\text{MSE} = J(\theta) = \frac{1}{2n} \sum_{i=1}^{n} (h_\theta(x^{(i)}) - y^{(i)})^2$$

Where:
- $J(\theta)$ is the cost function
- $(h_\theta(x^{(i)}))$ is the predicted price for the i-th observation using the parameter θ
- $y^{(i)}$ is the actual price for the i-th observation

For clarity, $h_\theta(x^{(i)})$ represents predicted output of the multiple linear regression function, also known as a hypothesis function. It predicts the housing prices based on the chosen features we included in the model parameterized by θ. θ here will represents the model parameters for $\beta_0, \beta_1, ..., \beta_{12}$ of our hypothesis function.

2.3 Stochastic Gradient Descent (SGD)

In machine learning, datasets often exhibit substantial scale both in terms of the number of training data points and the dimensionality of the data. A significant challenge in this context arises from the computational intensity associated with computing the gradient at a single point when dealing with an extensive sample size (n). This means computing the gradient for a single iteration will most likely be costly and time consuming. SGD counters this drawback by randomly picking an integer from the training data points at each iteration. Instead of computing the full gradient, it computes the gradient of a subset or single randomly chosen data point.

The fundamental concept behind SGD lies in its approach to approximating the gradient of the objective function by leveraging a randomly chosen subset of the training data. This stochastic strategy mitigates the need to evaluate the gradient for every data point at each iteration, resulting in a substantial reduction in the number of iterations compared to other methods like SD. By utilizing a randomly chosen subset of the training data, SGD introduces variability, preventing the algorithm from getting stuck in local minima. This enhances its ability to explore the solution space effectively. Moreover, this stochastic property also renders SGD more amenable to parallelization due to its independence of mini-batch computations. Thus, leading to further improvements in computational efficiency and training speed.

The general structure of SGD can be referenced below:

---

1. Let $k = 0$ and $\vec{x}_0 = \vec{x}$
2. If $\nabla f(\vec{x}_k) < restol$ or $k >$ max number of iterations then return $\vec{x}_k$
3. Else

3.1 Select random subset of data with size M
3.2 Define objective function $J(x)$
3.3 $\nabla f(x_k) = \frac{1}{M} \sum_{j \in subset} \nabla J(x_k, y_j, x_j)$
3.4 $\vec{x}_{k+1} = \vec{x}_k - \alpha_k \nabla f(\vec{x}_k)$
3.5 Increment k and go to Step 1.

The update step in SGD is given by:

$$\vec{x}_{k+1} = \vec{x}_k - \alpha_k \nabla f(\vec{x}_k)$$

where:

1. x_k is the current solution,
2. $\nabla f(x_k)$ is the gradient of the objective function with respect to the parameters estimated on a randomly chosen subset from the dataset.
3. α_k is the step size.

The descent direction is represented by the negative gradient of the objective function with respect to the model parameters. This negative gradient points in the direction of steepest decrease of our objective function, that is the MSE. In this context, the descent direction is determined by the average of gradient over the randomly selected subset from the dataset. This introduces stochasticity which then introduces variability to escape local minima.

2.4 Steepest Descent (SD)

The SD Method, also referred to as the Batch Gradient Descent method, operates as an iterative optimization technique designed to locate the minimum of an objective function. It achieves this by systematically moving in the direction of the negative gradient of the objective function. In contrast to SGD, the SD method computes the gradient using the entire dataset at each iteration. This makes the update process deterministic rather than stochastic, because SD relies on the precise calculation of the gradient based on the dataset.

The method systematically adjusts the parameters of the model in a direction that minimizes the objective function. This makes it particularly effective for scenarios where the entire dataset can be feasibly processed within each iteration. In other words, SD is efficient when the dataset is smaller in scale. SD may exhibit slower convergence on large-scale datasets, compared to the more computationally efficient and hypothetically faster-converging SGD method.

The general structure of SD can be referenced below:

1. Let $k = 0$ and $\vec{x}_0 = \vec{x}$
2. If $\nabla f(\vec{x}_k) < restol$ or $k >$ max number of iterations then return $\vec{x}_k$
3. Else
3.1 $\vec{p_k} = \nabla f(\vec{x}_k)$
3.2 $\vec{x}_{k+1} = \vec{x}_k + \alpha_k \nabla f(\vec{x}_k)$
3.3 Increment k and go to Step 1.

There are a few essential elements in this method: the descent direction and the update step. Both elements always work simultaneously. The update step in the steepest descent direction (3.2) is given by:

$$\vec{x}_{k+1} = \vec{x}_k + \alpha_k \nabla f(\vec{x}_k)$$

where:

1. x_k is the current solution,
2. $\nabla f(x_k)$ is the gradient of the function at x_k,
3. α_k is the step size, also known as the learning rate.

The descent direction is the negative of the gradient $\nabla f(x_k)$, which will point in the direction of the steepest decrease of the objective function at the current solution x_k.

An optimal step size α_k selection is important to the overall convergence behaviour. If α_k is too small, convergence will be slow; if α_k is to big, it can cause overshooting of the minimizer and diverge. In the setting of large dataset, optimal step size selection can be obtained with adjustment strategies. One of them is implementing the backtracking line search to optimally select the step size.

Backtracking line search starts with an initial guess for the step size, and iteratively reduces it until a stopping criterion or condition is satisfied. The stopping criterion is often defined by the Armijo-Goldstein condition such that it ensures sufficient decrease in the objective function. It checks if the actual reduction in function is proportional to the expected reduction. The Armijo-Goldstein condition is such that for some $c_1, c_2 \in (0, 1/2)$, choose an α_k that satisfies both:

$$f(\vec{x_k} + \alpha \vec{p_k}) \leq f(\vec{x_k}) + c_1 \alpha \nabla f_k^T \vec{p_k}$$
$$f(\vec{x_k} + \alpha \vec{p_k}) \geq f(\vec{x_k}) + (1 - c_2) \alpha \vec{p_k}^T \nabla f_k$$

3 Hypotheses and Expectation

The motivation behind this housing price prediction experiment is to observe and understand the convergence behavior of the two gradient descent methods. Based on the definition of SGD and SD, the expectation outcome inclines towards SGD converging faster than SD due to the use of random subsets of data in each iteration. This means SGD allows for quicker exploration of the parameter space. While the MSE on the training set may decrease rapidly initially for SGD, the final convergence may not be to the global minimum. This is because there may be more variability in the results due to the stochastic nature of SGD. On the contrary, SD with backtracking line search is expected to exhibit a more stable

convergence compared to SGD as the adaptive step size prevents overshooting of the minima. Although this implies the SD may take longer per iteration, the final convergence should be more reliable and may obtain even lower MSE on the testing set as compared to SGD.

It should be noted that the size of the dataset for this experiment may not be adequate to support the conventional expectation that SGD will converge faster than SD. With a relatively small dataset of 546 observations, both methods may struggle with overfitting because they are limited by the lack of complexity in the model that can be learned. Due to its inherent stochasticity, SGD may be more susceptible to overfitting due to its rapid adjustments to variability in a small dataset. This makes it weak to generalize unseen data. In contrast, SD with backtracking line search may be slightly more resistant to overfitting thanks to its controlled convergence behavior. Therefore, the balance between model complexity and the size of the dataset is crucial for accurately measuring the generalization performance of both SGD and SD.

Acknowledging the small size of the dataset in this experiment, our hypothesis for this experiment is that SD will demonstrate better generalization to the testing set due to its stable convergence behavior. This means that we expect to see SD converging faster than SGD in terms of the number of iterations it takes to converge, and may have more accurate convergence.

4 Experiment Setup

4.1 Data Processing

In the setting of machine learning, scaling features and normalizing data are crucial pre-processing steps. When features are not on a similar scale, the optimization process can be skewed, and it may take longer for the algorithm to converge. When the features with larger scales dominate the learning process, it becomes challenging for the algorithm to give appropriate consideration to features with smaller scales.

The dataset pertaining to this project has both numerical and categorical data, which will need to be processed for equalizing their feature scales. Numerical features typically have different scales, and some machine learning algorithms are sensitive to these differences. It can cause slow convergence, when the numerical variables are on vastly different scale. For example, the housing prices in the raw dataset were in the scale of millions, while some numerical features were in the scale of tens. On the other hand, categorical variables, which represent categories or labels, need to be converted into numerical form. One-hot encoding creates binary columns for each category, where the value 1 is assigned to "yes" and 0 is assigned to "no". Scaling methods, such as Z-score normalization can help bring numerical features and the encoded categorical features to a comparable scale.

By performing these pre-processing steps, we create a standardized and consistent input for our machine learning model. This process ensures that each feature contributes proportionally to the learning process.

4.2 Train and Test Data

For our model workflow, we divided our dataset into training and testing sets using an 80-20 split. This split ensures a representative random sample in both subsets and helps prevent biases caused by the order of the data. The training set, comprising 80% of the data, serves as the primary element for our model's learning process. The model will learn patterns and relationship between the features to make predictions through the training set. The testing set, comprising the remaining 20%, acts as unseen data that the model has not encountered during training. This set will help us gauge how well our model generalizes to

new data.

In the Matlab program where we implement this model, we utilize the `cvpartition` function with the `'HoldOut'` type, ensuring a straightforward 80-20 split. This method of partitioning is chosen for its simplicity and suitability for our specific model needs. Additionally, we use the `training` function in conjunction with `cvpartition` to obtain logical indices corresponding to the training set. These indices are then used to extract the training set from our dataset.

4.3 Objective Function Implementation

As discussed earlier, the objective function for this model is to minimize the MSE of the predicted price and the actual price.

1: **function** findMSE (θ, X, Y)
2:
3: $m \leftarrow length(Y)$
4:
5: predictions $\leftarrow X * \theta$
6:
7: squaredError $\leftarrow (predictions - y)^2$
8:
9: Mean Squared Error $\leftarrow \frac{1}{2m} * sum(squaredError)$

The function computes the predicted house prices using the linear model defined by the parameters θ and the input features X. It then calculates the squared differences between these predictions and the actual prices, response variable y. The final step involves computing the MSE, a measure that quantifies the average squared difference between predicted and actual values. A lower MSE signifies a more accurate and reliable predictive model.

4.4 Implementing SGD

The SGD function is designed to iteratively minimize the MSE between predicted and actual housing prices in training data with respect to θ, the model parameters.

```
function SGD(X, Y, α, maxk, restol)
    θ ← small random values
    price history ← matrix of 1's

    for a maximum of maxk times: do
        shuffle the training data to introduce randomness
        for each observation in shuffled data: do
            predictions ← training data x * θ
            error ← predictions - training data y
            gradient ← updates in the direction that minimizes it
            θ ← updated based on error in the negative direction of gradient

        price history(i) ← findMSE(θ, X, Y)
        if MSE is below restol: then
            declare convergence
        if exhausted all maxk iterations: then
            break the loop, and return; no convergence observed
return θ, price history
```

The model parameters are updated using SGD update rule. The gradient is computed based on the randomly selected observation, and the model parameters are adjusted in the direction that minimizes the error.

4.5 Implementing SD

In this specific implementation, the algorithm efficiently updates model parameters to converge towards the optimal values that yield the lowest MSE using backtracking line search method.

```
function STEEPESTDESCENT(X, Y, maxk, restol)
    θ ← zeros
    price history ← to check MSE over iteraionts

    for a maximum of maxk times: do
        gradient ← updates gradient of MSE with respect to θ
        use Backtrack Line Search method to find optimal step size α
        θ ← updated based on α and negative gradient (descent direction)

        if MSE is below restol: then
            declare convergence
        if exhausted all maxk iterations: then
            break the loop, and return; no convergence observed
return θ, price history
```

The parameter update step adjusts each θ value in the opposite direction of gradient with optimal step size α evaluated by the line search function. We utilize the backtracking method to scale for an optimal step size in each iteration. The line search function, with the backtracking method, is a useful addition to the SD function, as it enables dynamic adaptation of the model's features and ensures that the algorithm progresses toward minimizing the MSE.

4.6 Analysis Structure

For analysis, a few metrics that will determine the methods' convergence behaviours are number of iterations and the summary of their fit for the model.

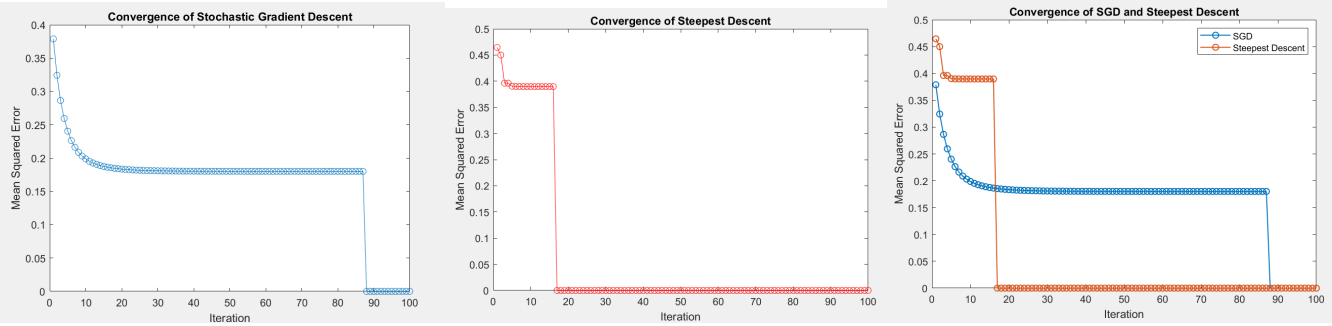
We create line plots to display the convergence behaviours for SGD and SD with respect to the MSE and the number of iterations. These convergence plots serve as valuable visual tools for understanding and interpreting the behaviour of our optimization algorithms. By charting the MSE over the course of iterations, we gain insights into the convergence patterns. The trajectories of the plots allow us to observe how quickly each algorithm approaches a minimum and whether the convergence is steady or not. A smooth, consistent descent may suggest stable convergence, while erratic behavior might indicate challenges such as overshooting or oscillation between minima.

To understand how both methods handle prediction, we utilize scatter plots comparing actual and predicted housing prices. These scatter plots serve as diagnostic tools for assessing the predictive performance of our housing price prediction model. By plotting the actual housing prices against the predicted prices, we can visually inspect the relationship between the two variables. The scatter plots offer a clear depiction of how well our model captures the underlying patterns in the data. We can also identify overfitting from the scatter plots. If the model has overfit the training data, the scatter plot may reveal overly complex patterns. Signs of overfitting might include tight clusters or fitting to outliers in the training set. Conversely, a well-generalizing model should exhibit a more even spread and distribution of predicted prices relative to the actual prices in the scatter plot.

5 Results

In this section, we present and analyze the results obtained from our housing price prediction experiment. The primary objective of the experiment was to investigate the convergence behavior and predictive performance of two gradient descent methods.

We begin by exploring the convergence behavior, where the MSE on the training set was monitored over the course of iterations for both SGD and SD.

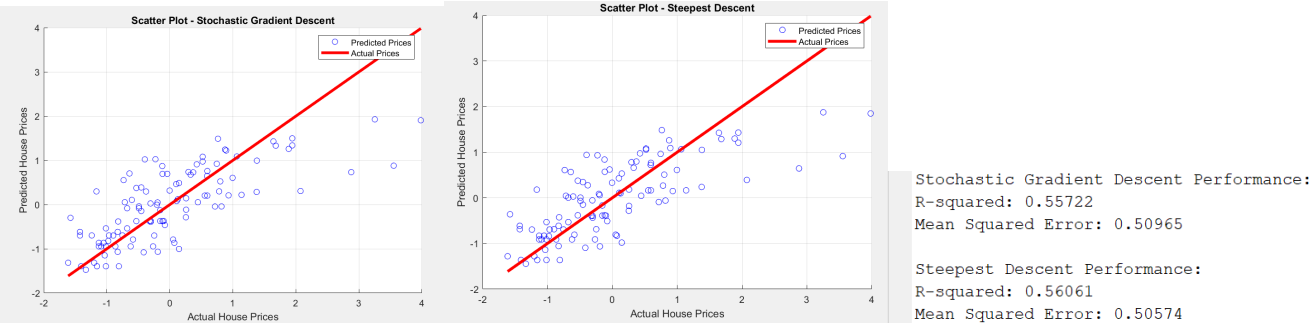


(a) (b) (c)

Figure 1

Comparing figure (a) and (b), SGD converges at the 87th iteration, while SD converges at the 16th iteration. This is consistent with our expectation that SD converges faster than SGD. Moving to Figure (c), we present a superimposed view of the two convergence behaviors. As anticipated, SGD demonstrates rapid decrease in MSE during the initial iterations, reflecting its ability to quickly adapt to random subsets of data. However, the stochastic property of SGD introduces variability, preventing an immediate convergence to a local minimum. On the contrary, SD with backtracking line search exhibits a more controlled and steady convergence. It converges once the minimum MSE is identified. This demonstrates the effectiveness of the adaptive step size in preventing overshooting and oscillations.

The relatively small dataset size is a critical factor that may have skewed our observations for both convergence behaviors. In a larger dataset scenario, both SGD and SD may exhibit vast difference in convergence behaviors and generalization patterns. The constraints imposed by the dataset size amplify the risk of overfitting, especially considering the complexity of the housing price prediction task.



(a) (b) (c)

Figure 2

The two scatterplots, Figure 2(a) and 2(b), do not differ much in terms of the spread of their respective predicted prices. We also observe the presence of outliers in both scatterplots—these outliers are data points that deviate significantly from the general trend. Their existence is indicative that both methods struggle with overfitting by attempting to fit the training data too closely.

This observation is supported by the R-squared and MSE values shown in Figure 2(c). R-squared measures the proportion of the variance in actual housing prices that is predicted from the model's features, gauging how well the model captures and accounts for the observed variability. The R-squared output in Figure (c) indicates R-squared values of 0.55722 for SGD and 0.56061 for SD. This indicates that approximately

55.7% and 56.1% of the variance in the actual housing prices can be explained by the features used in SGD and SD respectively. Both methods exhibit immaterial differences in R-squared values, indicating a comparable ability to account for variability in actual housing prices based on the features.

As for MSE, both SGD and SD show relatively close MSE values (SGD: 0.50965 and SD: 0.50574). This observation implies that both methods achieve a comparable level of accuracy in predicting housing prices. To scrutinize the values, the MSE for SD is slightly smaller. This aligns with our expectation that the method may attain an even lower MSE compared to SGD, given the backtracking line search adaptation that prevents overshooting the minima.

6 Conclusion

With the motivation of understanding the comparative performance of SGD and SD for predicting housing prices, our experiment revealed insights that may deviate from conventional expectations. The results of this experiment may not align precisely with the expected norms in larger-scale datasets; specifically, SGD converges faster than SD. However, this experiment brings awareness that when a small dataset is used for such a model, the issue of overfitting arises regardless of the different gradient descent methods. This is evidenced by the presence of outliers in the scatter plots, magnifying overfitting due to the constraints of a limited dataset.

The key takeaway from our experiment is that the lack of data poses a significant challenge to both SGD and SD. The observed overfitting issues underscore the importance of dataset size in training accurate and generalizable models. Obtaining an adequate and large-scale dataset from open-source databases is challenging. Settling with a relatively small dataset of 546 observations in this experiment clearly imposes constraints on the generalization of our conclusions. It is crucial to acknowledge that the limitations imposed by our dataset size may not necessarily generalize to larger-scale scenarios. The dynamics of large-scale datasets may introduce additional factors that influence the comparative performance of SGD and SD. This can potentially lead to more contrasting outcomes in terms of computational performance and the rate of convergence.

Looking forward, addressing the lack of data should enable a more robust assessment of SGD and SD methods under more diverse scenarios. The limitations encountered in our experiment, stemming from the relatively small dataset of 546 observations, highlight the need for caution when generalizing conclusions on the two methods. The contrast in computational efficiency and convergence behavior between these two methods may become more pronounced on a grander implementation scale. Exploring avenues for dataset augmentation and diversifying the datasets become significant to build the foundation for a more comprehensive understanding of how SGD and SD respond to challenges in larger datasets.

7 References

1. Stochastic Gradient Descent: A Basic Explanation. [https://mohitmishra786687.medium.com/stochastic-gradient-descent-a-basic-explanation-cbddc63f08e0]

2. Dealing with Large Datasets: the Present Conundrum. [https://towardsdatascience.com/title-86a91890b5c6]

3. Stochastic gradient descent [https://optimization.cbe.cornell.edu/index.php?title=Stochastic_gradient_descent]

4. Math309 Fall 2023, Inexact Line Search(1)[https://canvas.sfu.ca/courses/79324/pages/course-schedule-lectures-and-assignments]