Data Compression using Sparse Stochastic Gradient Descent

Scott Hoyland

STAT 672

May 10, 2021

Table of Contents

[Summary 3](#_Toc71398957)

[Introduction 3](#_Toc71398958)

[Methods 4](#_Toc71398959)

[Application 9](#_Toc71398960)

[Conclusion 14](#_Toc71398961)

[References 15](#_Toc71398962)

# Summary

The tradeoff in machine learning algorithms for large scale datasets is computational complexity versus accuracy. Sparse stochastic gradient descent lowers the computational complexity by reducing the communication cost of distributed stochastic descent through creating a subset of the gradient vector to zero based on an optimal probability algorithm. An experiment on the sparse stochastic gradient descent approximation of logistic regression validates the sparsification approach presented.

# Introduction

Algorithms that run on large datasets require a different tradeoff than algorithms that run on small datasets. For small datasets, algorithms are mostly focused on the tradeoff of approximation versus estimation, also called the bias/variance tradeoff. As the size of datasets increases, modern machine learning algorithms must balance the computational complexity versus accuracy tradeoff (Buttou and Bousquet). There is no strict cutoff for when a dataset grows to a size where it officially becomes a large dataset; rather, a dataset is considered a large dataset when the resource of time starts to become an issue in computation. The size where the dataset becomes a large dataset is dependent upon the type of algorithm being run and the machine that the algorithm is being run on. A dataset and an algorithm being run on a mobile phone have different resource constraints than the same dataset and algorithm being run on a supercomputer. One solution to the large-scale dataset tradeoff is to break algorithms down into individual parts and run the algorithm on a distributed system; however, the distributed system incurs its own costs in the communication between the different workers in the system. After each individual worker in the distributed system runs a piece of the algorithm on any local data, it must transfer the result to the common worker for consolidation and updating. Therefore, compared to a single-thread algorithm, distributed algorithms have additional communication costs at the start of the algorithm and at each updating step (David, A.). To be a more efficient distributed system, the algorithm may be written in a way that sacrifices some accuracy for an increase in computational efficiency. The communication cost between each distributed worker and the common worker can be reduced by either decreasing the size of the message or by splitting the message into multiple parts. Sparse stochastic gradient descent aims to achieve reduced communication cost through reducing the number of floating-point bits by reducing a subset of the updating gradient vector to zero.

In the rest of this report, I will examine the methods involved in gradient descent and how sparse stochastic gradient descent approaches the communication problem. I will run an experiment using the gradient descent approximation to logistic regression to demonstrate the computational complexity versus accuracy tradeoff. Finally, I will offer conclusions and ideas for future research.

# Methods

Gradient Descent is an optimization algorithm based on a convex function and updates its parameters iteratively to minimize a given function. It is an optimization algorithm to find a local minimum of a differentiable function. Gradient descent is used to find the optimal parameters that minimize the cost function of an algorithm. The gradient measures how much the output function changes as the input parameters are marginally changed. In this regard, the gradient is similar to the slope of a function. The larger the gradient, the faster the function can learn, but if the gradient is zero, the function stops learning. Mathematically, the gradient is the partial derivative of the cost function with respect to the function inputs. Figure 1 is a visual representation of the iteration optimization of gradient descent.

**Figure 1**

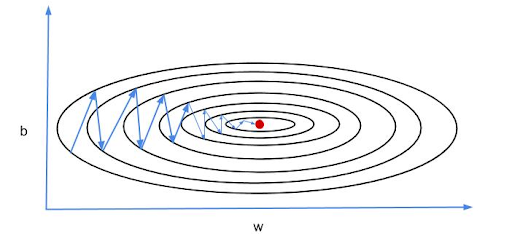


Figure 1: Each line segment represents an iteration of the gradient descent algorithm where the middle point of the shrinking ellipsoids represents the minimum of the cost function. The direction of the line segment represents the updating function of the gradient algorithm (*Gradient Descent with Momentum)*.

Let w be the coefficient vector of the function, let X be the data matrix with n observations and d parameters, let η be the learning rate, and let g(w) be the gradient vector, then gradient descent can be modeled by the equation

This equation models the updating of the coefficient vector in the direction of the steepest gradient. The learning rate, η, represents how much influence the gradient vector has on the updating of the coefficient vector. The larger the learning rate, the larger step size the gradient has on the updating of the coefficient vector. The choice of the learning rate determines the rate at which the gradient descent reaches its minimization point. If the learning rate is too large, there is a possibility that the updating function moves the coefficient vector past the minimization point. The smaller the learning rate, the less likely the updating function will pass the local minimum, but the longer it will take to reach the minimum of the optimization function.

Two types of gradient descent include batch gradient descent and stochastic gradient descent. In batch gradient descent, all the available data is included in the data matrix, X. In stochastic gradient descent, a subset of the observations from the data is used in the data matrix when performing gradient descent. The subset of the data can range from 1 to less than n where n is the total number of observations in the data. A stochastic gradient descent using 1 < k < n observations from the data can also be called min-batch gradient descent.

The calculation of the gradient, g(w), in the application of gradient descent can be computationally expensive. The algorithm must calculate an estimate using the current probability vector, calculate the gradient based on the current estimate, and update the coefficient vector based on the gradient. If the number of iterations to reach the minimum in gradient descent is n, then the total computational cost is n times higher than the computational cost to run one iteration of the algorithm. One way to improve the runtime of the algorithm is to run gradient descent in a distributed system. Each worker in the distributed system calculates the gradient based on the worker’s available data. In stochastic gradient descent, each worker has a subset of the data of size k/M, where M is the total number of workers. The gradient for each worker is averaged by a common worker using the formula

where gm(w) is the gradient vector calculated by the mth worker. The updating function for the distributed system becomes

While distributed gradient descent helps with the overall runtime of the algorithm, additional complexity is added through the communication cost of each distributed worker sending a gradient vector to the common worker. To lower the communication cost of distributed stochastic gradient descent, sparse stochastic gradient descent can be used. The sparse stochastic gradient descent uses the same subsetting method as stochastic gradient descent. The sparsification occurs in the communication of the gradient updating vector from the distributed worker to the common worker. Where distributed gradient descent uses the updating function above, the sparse stochastic gradient descent sparsifies gt(wt) by creating the gradient vector

where pi is the probability that element i of the gradient vector gt(wt) remains nonzero in the sparse gradient updating vector Qt(wt) (Wangni J., Wang J., Liu J., Zhang T). The variable Z is a binomial variable where P(Zi = 1) = pi. The scaling of the nonzero gradient elements by p keeps Qt(wt) an unbiased estimator of g(wt) since

In distributed learning, each worker in the system calculates a sparse stochastic gradient Qt(wt), which is communicated with the common system. The common worker then creates an average sparse stochastic gradient from the equation

where there is a total of M workers and m represents one worker in the distributed system. The function Qm(w) is the sparse gradient communicated from worker m to the common worker. The updating function then becomes

While the sparsification method described above reduces the communication cost for each node to the common system, it increases the total variance of the algorithm. The probability vector must be created to find the best tradeoff between the sparsity of the vector and the variance created. The optimal probability vector, p, can be approximated through the following algorithm proposed by Wangni J., Wang J., Liu J., Zhang T. Begin by creating an updating gradient, g(wt), and calculating a beginning probability algorithm

where the variable ρ represents the expected sparsity of the gradient. From the initial probability vector, p0, identify the elements that are not 1 and create the vector I that includes the vector position of each are those elements

Using the position vector, calculate a scaling coefficient

If the scaling coefficient is less than or equal to 1, c 1, then use p0 as the probability vector. If c is greater than 1, then update the probability vector using the following equation

Repeat this process, updating p each iteration until the scaling coefficient is less than or equal to 1. When c 1, then pj used in the calculation of c is the probability vector used in the sparsification of g(wt).

The theoretical bounds of the change in variance and computational cost using sparse stochastic gradient descent can be calculated based on the calculations by Wangni J., Wang J., Liu J., Zhang T. The expected sparsity of Qt(wt) can be upper-bounded by (1+ρ)s where ρ is the pre-defined sparsity parameter and s is the nonzero subset of gt(wt). The increase in variance brought on by the sparsification of the gradient vector is upper bounded by (1+ρ). However, the number of iterations needed to reach the same accuracy within the sparse stochastic gradient descent as in stochastic gradient descent may need to increase by up to a factor of (1+ρ). With the sparsification, the overall number of floating point numbers communicated is reduced by up to a factor of (1+ρ)s2/d. This theoretical lower bound on the number of floating points shows the lower communication cost between workers in the distributed system for the sparse stochastic gradient descent algorithm.

# Application

One of the main considerations in a bank’s decision to offer a loan is the probability that the applicant will be able to pay back the loan. On loan applications, banks collect a variety of information such as loan amount, applicant income, and credit score. In addition, loan officers can collect information about the applicant such as if the applicant comes alone or accompanied to a meeting, what time of day the loan meeting takes place, and what day of the week the loan meeting takes place. Using this information, and other information collected about the applicant, banks create a holistic picture of the applicant’s life and can use this information to inform loan decisions. After a loan is granted, the bank keeps track of whether a loan recipient misses a payment or not. I will be using a dataset of loan applicant data and payment information from Kaggle.com to test sparse stochastic gradient descent approximation of logistic regression. The data contains over 300,000 observations and over 100 different features. The features in the original dataset that are factors were transformed into leave-one-out indicator variables for a final feature set size of 226. The features that were continuous variables were normalized. Some variables were available as mean, median, and mode, and the median values were chosen for this analysis. Logistic regression approximation was appropriate for this data because the dependent variable is a binary variable representing whether the loan recipient missed a loan payment. The logistic regression estimated the probability that an applicant missed a loan payment based on the information gathered by the bank.

The gradient descent approximation of the logistic regression is based on the Sigmoid function

where Zt is the linear combination of the estimated coefficients and the independent variables plus an intercept term given by the equation

The gradient descent loss function for logistic regression is:

The loss function produces an optimal gradient by taking the partial derivative of the loss function with respect to w­t. First, taking the logarithm of the prediction function for logistic regression and rearranging gives

Plugging in the following into the loss function, L(wt), defined above gives

Taking the partial derivative of the previous equation with respect to the coefficient vector wt gives

which is the function of the updating gradient, g(wt).

Using logistic regression with a coefficient requires an estimate of the coefficient and the corresponding updating function. One way to view this is to add an additional column to the data matrix (X) of a vector of size n containing the value 1 repeated. The derivation for the updating would be the same as above, and the broken out updating function for the coefficient (bt) would be

The resulting logistic regression gradient descent updating function would then be

g(w;b) = .

The probability of an applicant missing a loan payment was estimated using logistic regression, batch gradient descent, stochastic gradient descent, and sparse stochastic gradient descent. Logistic regression was run using the R function **glm**. The algorithms for the three types of gradient descent were written in R in a distributed manner with the **foreach** function within the **foreach** package combined with the **doParallel** package.

Each type of gradient descent used a learning rate of 0.01, and the stochastic gradient descent and sparse stochastic gradient descent used a subset percentage of 0.33. The sparse stochastic gradient descent used a sparse coefficient of 0.5. A misclassification rate was calculated on all the data for each of the algorithms using the final updated coefficient vector. The convergence of the algorithm was based on the change in the misclassification rate. The time to convergence was used as a proxy for the communication cost of using a distributed algorithm. The number of iterations shows how many updating functions were calculated before the algorithm reached convergence. The number of iterations to convergence is highly dependent on the data and the learning rate and is not sufficient on its own to choose one algorithm over another. Therefore, each algorithm was run 45 times to show the difference in runtime if each of the algorithms were run an equal number of times.

The results of the algorithms are summarized in Table 1. As expected, each of the three types of gradient descent converged faster than the logistic regression function took to run, and the misclassification rate for the three types of gradient descent is higher than the misclassification rate for the logistic regression. The differences in runtime and accuracy show the computational complexity versus accuracy trade off on the missed loan payment data. The runtime for the batch gradient descent was 72% faster than the runtime for logistic regression, the runtime for the stochastic gradient descent was 26% faster than the runtime for logistic regression, and the runtime for sparse stochastic gradient descent was 97% faster than the logistic regression. The massive improvement for the sparse stochastic gradient descent came mostly from the fast convergence the algorithm had for this data. I believe the algorithm to find the optimal probability vector based on the relative size of each element in the gradient vector helped the sparse stochastic gradient descent to converge faster than the batch gradient descent or the stochastic gradient descent. In the applicant data, there may be a few highly influential factors that that optimal probability algorithm was able to identify with fewer updating steps than either the batch gradient descent or stochastic gradient descent could. While the sparse stochastic gradient descent is expected to increase the variance by up to 1+ρ, 1.5 in this case, the misclassification rate for sparse stochastic gradient descent was equivalent to the misclassification rate of the batch gradient descent and the stochastic gradient descent.

**Table 1**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Regression Type**  **(Gradient Descent ran in parallel using 4 cores)** | **Misclassification Rate** | **Number of Iterations to Convergence** | **Runtime** | **Runtime Percent Improvement Compared to Logistic Regression** |
| Logistic Regression | 7.96% | - | 62s | - |
| Batch Gradient Descent | 8.11% | 17 | 17.3s | 72% |
| Stochastic Gradient Descent | 8.11% | 45 | 45.5s | 26% |
| Sparse Stochastic Gradient Descent | 8.11% | 2 | 2.01s | 97% |

Table 1: The misclassification error, number of iterations to convergence, runtime, and percent runtime improvement for logistic regression, batch gradient descent, stochastic gradient descent, and sparse stochastic gradient descent.

The low number of iterations needed for the sparse stochastic gradient descent did not provide enough information on how the sparse stochastic gradient descent decreases complexity over many additional iterations compared to the batch gradient descent and the stochastic gradient descent. Therefore, each algorithm was run for 45 iterations, where one iteration is the updating of the gradient function by the common worker. Table 2 summarizes the results for this test. As expected, the sparse stochastic gradient descent had a lower runtime for 45 iterations than either the batch gradient descent or the stochastic gradient descent. The sparse stochastic gradient descent had a runtime 13% faster than the stochastic gradient descent and 9% faster than the batch gradient descent. While runtime is a proxy for the number of floating-point numbers communicated, we can expect that the sparse stochastic gradient descent communicated, at most, (1+ρ)2s/d (or 0.01s) fewer floating-point numbers.

**Table 2**

|  |  |  |
| --- | --- | --- |
| **Gradient Descent Type** | **Runtime of 45 iterations** | **Percent Improvement over the Slowest Algorithm** |
| Batch Gradient Descent | 43.39 | 4.5% |
| Stochastic Gradient Descent | 45.45 | 0% |
| Sparse Stochastic Gradient Descent | 39.53 | 13.0% |

Table 2: The runtime of 45 iterations of the batch gradient descent, stochastic gradient descent, and the sparse stochastic gradient descent and the percent improvement in runtime for the batch gradient descent and sparse stochastic gradient descent compared to the stochastic gradient descent.

# Conclusion

The sparse stochastic gradient descent algorithm is one way to address the computational complexity versus accuracy tradeoff, which can be applied to nearly all machine learning algorithms. The idea in sparse stochastic gradient descent is to reduce a subset of the gradient vector to zero, reducing the floating-point bit communication cost within a distributed system. The sparse stochastic gradient descent reduces communication costs by, at most, (1+ρ)2s/d while having a variance increase upper bounded by (1+ρ). The example approximating logistic regression shows that sparse stochastic gradient descent can obtain the same misclassification rate as batch gradient descent and stochastic gradient descent while running faster over the same number of iterations as either of the other two gradient descent algorithms. Sparse stochastic gradient descent could be even more efficient than the algorithm I created for this report based on coding techniques and programming language used. The communication of the sparse gradient could be even more efficient if only nonzero elements and their positions were sent from each worker in the distributed system to the common worker.

# References

*Buttou, L. and Bousquet, O. The Tradeoffs of Large Scale Learning https://proceedings.neurips.cc/paper/2007/file/0d3180d672e08b4c5312dcdafdf6ef36-Paper.pdf*

*David, A. Communication Cost in Parallel Machines. http://people.cs.aau.dk/~adavid/teaching/MTP-06/03a-MVP06-notes.pdf*

*Dutta, Gaurav. Loan Defaulter EDA in a Real Business Scenario. https://www.kaggle.com/gauravduttakiit/loan-defaulter*

*Gradient Descent with Momentum*. http://dataplusplus.ca/blog/2017/gradient-descent-with-momentum

Wangni J., Wang J., Liu J., Zhang T. Gradient Sparsification for Communication-Efficient Distributed Optimization. https://arxiv.org/pdf/1710.09854.pdf