CS-328 Introduction to Data Science
Sampling and Sketching in Machine Learning Pipeline
Final Project Report

Heer Ambavi (16110062), Nisarg Ujjainkar (16110102), S Vinu Sankar (16110143)

July 2020

1 Introduction

Machine learning has become so computationally expensive as data is growing day by day. Hence, there is a need to make them data- and resource-efficient. When the number of data points or dimnension of the dataset is very high, researchers use methods such as sketching and sampling to reduce computations for machine learning training. Sparse projection or hashing is done to learn machine learning models in the dimensionally reduced space. Coreset construction is an efficient sampling method used for the data-efficient training of machine learning models. Coreset is the weighted subset of a full dataset that summarises the entire dataset. They can also be used to provably approximate machine learning models with error ϵ and probability (1-δ). In this report, we report the observations of experiments we have implemented coreset-based sampling techniques. For implementations, see our GitHub page[1].

2 Literature Review

Much work has been done on using sampling and sketching methods in Machine Learning Algorithms. Mirzasoleiman et al. [1] present CRAIG, a method to select a coreset of training data that closely estimates the full gradient. The work speeds up Gradient Descent by 6x for logistic regression and 3x for training deep neural networks.

Baykal et al. [2] provide a coreset construction algorithm for solving classification tasks using Support Vector Machines. In the work by Jiang et al. [3], compression method to efficiently handle a gradient for Distrbuted Machine Learning system is presented. Paul et al [4] use 3 different types of random projection on SVM matrices and compare their performance on sparse and non-sparse datasets. Weinberger et al. [5] provide theoritical bounds and empirical observations on feature hashing on multitask learning problem.

[1] https://github.com/vinusankars/Sampling-and-sketching-methods-for-machine-learning

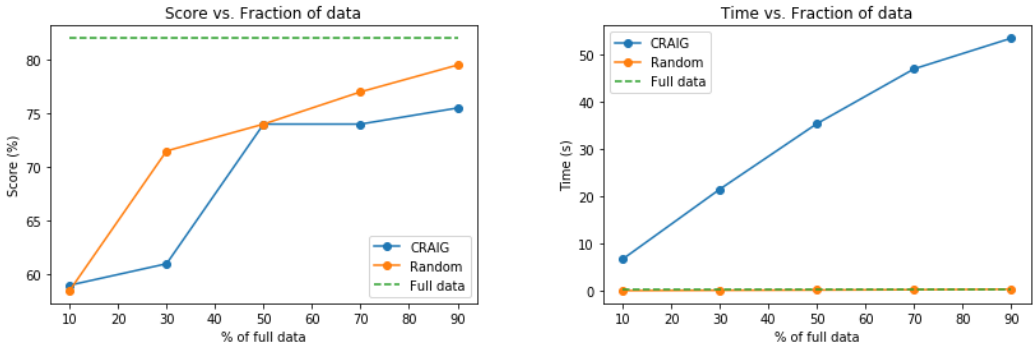
3 Experiments and Observations

3.1 Coreset-based sampling

3.1.1 Coresets for accelerated gradient descent

The experiment in this section is based on the work by Mirzasoleiman et al. [1] on using CoResets for Accelerated Incremental Gradient (CRAIG). This work proposes the first ever data-efficient training methodology for machine learning using a coreset-based approach. The objective of this paper is to find a smaller weighted subset of the full dataset, which can approximate the full gradient components as close as possible to the full dataset by maximizing a submodular function. The CRAIG algorithm is complementary to other standard IG algorithms such as SGD, SAGA, and SVRG.

For a training dataset V, the objective is to find a subset S with per-element weights $\{\gamma_i\}_{i=1}^{|S|}$. CRAIG uses a coreset-based approach to find $\mid S \mid$ medoids in the full gradient component space. This will help the training process speed-up by a factor of $\mid V \mid / \mid S \mid$. The paper transforms this problem into a submodular set cover problem, which is approximately solved using a greedy approach.



(a) Score vs. fraction of data

(b) Time vs. fraction of data

Figure 1: Comparison of CRAIG with full dataset and random coreset data for the covtype.binary dataset. Time and score with varying size of coreset is compared here.

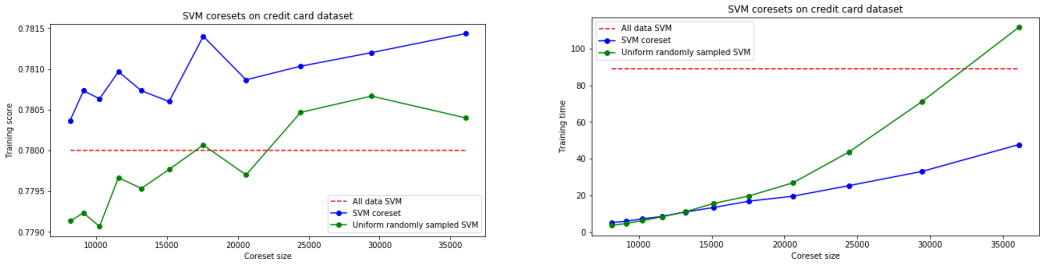
In our experiments, we try to implement a logistic regressor for covtype.binary[2] dataset classification problem using a gradient descent approach (for convex optimization). We faced many problems while implementing this experiment and failed to discover the empirical benefits claimed in the paper. The algorithm 1 in [1] claims it to be $O(\mid V \mid . \mid S \mid)$. But, our implementation, based on our understanding of the paper is $O(\mid V \mid^2 . \mid S \mid)$. The dataset we are using has $\mid V \mid = 581{,}012$ datapoints. With the bottle-neck over compute resources (Intel i5 CPU and 4 GB RAM), we chose to experiment with random 200 datapoints from the dataset. The results obtained are shown in Figure 1. Time reported for CRAIG is the total time for preprocessing and training added together.

3.1.2 Coresets for training support vector machines

In this experiment we implement coreset construction specifically for training Support Vector Machines (SVMs) based on the work by Bayka et al. [2]. In this work, they show that for a dataset with n datapoints and d dimensional inputs, a weighted $(\epsilon, \delta)-$coreset can be created using $O(log\ n(d\ log\ log\ n + log^2 n))$ datapoints. The

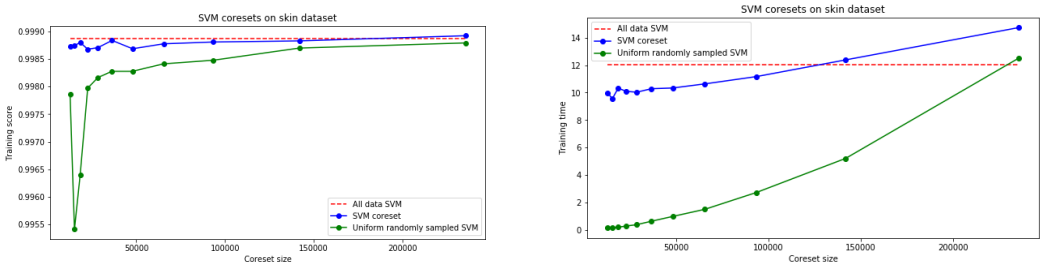
[2]https://www.csie.ntu.edu.tw/ cjlin/libsvmtools/datasets/binary.html

application of the work is shown by using it for training SVMs on the skin[3] and credit card[4] datasets. The skin dataset contains 245,057 4-D data points. The credit card dataset contains 36,000 24-D data points.



(a) Coreset size vs. score (b) Coreset size vs. time

Figure 2: Comparing the SVM coreset method with random coreset and full data on the credit card dataset.



(a) Coreset size vs. score (b) Coreset size vs. time

Figure 3: Comparing the SVM coreset method with random coreset and full data on the skin dataset.

According to the assumptions in the paper, the dataset is preprocessed to make have a zero mean and unit variance. The sensitivities of data points are tightly upper bounded using Assumption 6 mentioned in the paper to get importance values. These importance values along with a multinomial sampling is used to create the coreset. They also show the guarantee for creating an ϵ-coreset with a probability of $(1 - \delta)$ using this approach. The results and analysis for this experiment is shown in Figure 2 and Figure 3. The use of SVM coresets is displaying a significant speed up in training with very less or no drop in accuracy score.

4 Additional Study

4.1 Sketching Data-structures

Sketches of a data are a special kind of data structures that can represent large data in sublogarithmic or constant space. This can reduce the memory required for temporary storage for data, by a great extent. These kinds of data structures are very helpful when we have a stream kind of data. Here are a few sketching data structures that are commonly used today:

4.1.1 Count-Min Sketch

This data structure is in the form of a table[6], where every column is a hash function and the hash function can map a given event to an index of the column. Every cell

[3]https://archive.ics.uci.edu/ml/datasets/Skin+Segmentation/
[4]https://archive.ics.uci.edu/ml/datasets/default+of+credit+card+clients

is a counter. If an event comes, we pass it to all the hash functions and increment the counter of corresponding cells. To query an event, we again pass it to the hash functions and get the values of the cells the the event maps to. We then return the minimum of all the cells.

Here, the hash functions could map multiple event to same index and as the number of events increase, collisions also increase, which reduces the accuracy. Still, count min sketches are very efficient when it comes to space saving.

4.1.2 Hyper loglog Estimation

This is a from of hash function[7] which maps a given event to binary form and tries to estimate the number of unique data points. It leverages the fact that one in 2^z events gets mapped to binary form which begins with z zeros.

4.1.3 Bloom Filter

This data-structure is useful in finding whether a data-point belongs to a certain set[8]. It is in the form of a bit string with m bits and k hash functions ($k < m$). If we want to put a data point into a set, we pass it to the k hash functions. All of them return a different bit position and we set all of those k bit positions to 1. When querying, a point belongs to a set iff all the k bits are 1.
All the three above mentioned data-structures together could be used to store an approximate sketch of the data without loosing much on the accuracy, but saving a lot of memory.

References

[1] Baharan Mirzasoleiman, Jeff Bilmes, and Jure Leskovec, "Coresets for data-efficient training of machine learning models," *arXiv preprint arXiv:1906.01827*, 2019.

[2] Cenk Baykal, Lucas Liebenwein, and Wilko Schwarting, "Training support vector machines using coresets," *CoRR*.

[3] Jiawei Jiang, Fangcheng Fu, Tong Yang, and Bin Cui, "Sketchml," *Proceedings of the 2018 International Conference on Management of Data - SIGMOD 18*, 2018.

[4] Malik Magdon-Ismail Saurabh Paul, Christos Boutsidis and Petros Drineas, "Random projections for support vector machines," *Proceedings of Machine Learning Research, Volume 31: Artificial Intelligence and Statistics, 29-1 May 2013*.

[5] Kilian Weinberger, Anirban Dasgupta, John Langford, Alex Smola, and Josh Attenberg, "Feature hashing for large scale multitask learning," *Proceedings of the 26th Annual International Conference on Machine Learning - ICML 09*, 2009.

[6] Karan Shukla, "Big data with sketchy structures, part 1 — the count-min sketch," *medium*, 2018.

[7] Philippe Flajolet, Éric Fusy, Olivier Gandouet, and Frédéric Meunier, "HyperLogLog: the analysis of a near-optimal cardinality estimation algorithm," in *AofA: Analysis of Algorithms*, Philippe Jacquet, Ed., Juan les Pins, France, June

2007, vol. DMTCS Proceedings vol. AH, 2007 Conference on Analysis of Algorithms (AofA 07) of *DMTCS Proceedings*, pp. 137–156, Discrete Mathematics and Theoretical Computer Science.

[8] Karan Shukla, "Big data with sketchy structures, part 2 — hyperloglog and bloom filters," *medium*, 2018.

[9] Badih Ghazi and Joshua R. Wang, "Recursive sketches for modular deep learning," 2019.

[10] Max Pagels, "Introducing one of the best hacks in machine learning: the hashing trick," 2017.