Optimization Methods for Large-Scale Machine Learning

Aditi Gangrade (20162059)
Dishant Chauhan (20162109)
Rohit Tiwari (20162085)
Shreya Singh (20162060)

April 23, 2018

Abstract

This project aims at using second order Newton methods instead of the traditional gradient descent method for optimization of machine learning problems. In this report, we present a hybrid approach using the power of both, first order and second order methods for solving optimization problems, in order to reduce both time and space complexity.

1 Introduction

As the training data sets become large, the classical gradient methods tend to be slow in convergence. The goal of this project is to identify a suitable optimization method for a given cost function, and then to investigate a class of parallel optimization solvers that are suitable for large scale machine learning applications. Effectiveness of our algorithm was tested on existing datasets in terms of both space as well as time complexity.

2 Definitions

2.1 Optimization Problem

In mathematics and computer science, an optimization problem is the problem of finding the best solution from all feasible solutions. Optimization problems can be divided into two categories depending on whether the variables are continuous or discrete. An optimization problem with discrete variables is known as a discrete optimization.

2.2 Optimization Problems in Machine Learning

Optimization problems in machine learning arise through the definition of prediction and loss functions that appear in measures of expected and empirical risk that one aims to minimize. We first derive a cost function or the loss function for a given problem and then iterate over the various feasible solutions to find the most optimum one, (the one that minimizes the cost incurred).

2.3 Convex Functions

A real valued function, defined on an n-dimensional interval is called convex (convex downward or concave upward) if the line segment between any two points on the graph of the function lies above or on the graph. Formally, let $f : X \to R$ be a function
f is called convex if :

$$\forall x_1, x_2 \in X, \forall t \in [0,1]$$
$$f(tx_1 + (1-t)x_2) \leq tf(x_1) + (1-t)f(x_2)$$

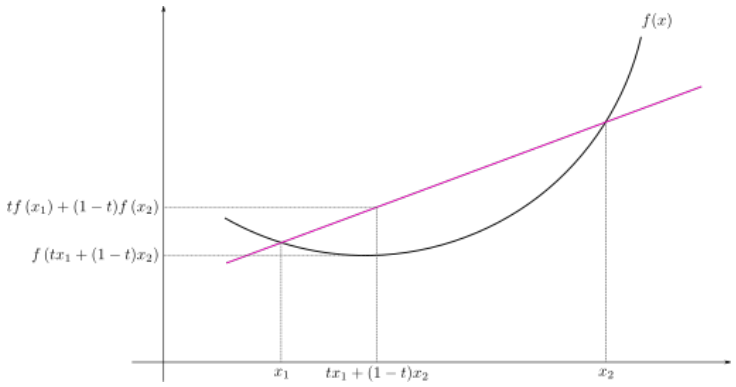

Figure 1: Convex Function

2.4 Predictor Function

A linear function of a set of parameters and independent variables, whose value is used to predict the outcome of dependent variable.
Matrix notation of a predictor function is :
for a data point i, where i = 1,...,n

$$f(i) = \theta^T . x_i$$

2.5 Logistic Regression

A statistical method for analyzing a dataset in which there are one or more independent variables that determine an outcome. Logistic regression transforms its output using the logistic sigmoid function to return a probability value which can then be mapped to two or more discrete classes.

2.6 Cost Function

It is a measure of how wrong the model is in terms of its ability to estimate the relationship between dependent and independent variables. This is a function which measure the difference between the estimator (the dataset) and the estimated value (the prediction). One common type of cost function is Mean Squared Error :

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - Y_i')^2$$

where Y_i is Estimator
and Y_i' is Estimated value

3 Approach

3.1 Basic Idea

Almost every machine learning problem revolves around minimizing a convex cost function. Traditional approaches use Gradient Descent method to accomplish this. However, as gradient descent is based on single derivative (i.e. the slope of the function at the given point), the convergence is slow. To overcome this issue, we move to a better approximation that takes into account the curvature details of the given cost function by computing the double derivative or the Hessian matrix.

Another issue that crops up while using the double derivative is the large memory space required by the Hessian matrix. As a solution to the memory issue, we do not store the complete Hessian and instead, calculate the matrix-vector product on the go, as required by the iterative solution of Conjugate Gradient Method.

To optimize the algorithm, a number of steps are taken. First, the data set is preprocessed to detect and remove all dependent features, and also, rows containing all zeros. Second, as the Hessian calculation takes a lot of time, we move to a hybrid approach that calculates Hessian only for the first iteration. This gives a close approximation for Gradient Descent iterations and decreases the no. of iterations considerably. Third, the matrix-vector product calculation, as is independent for each row, is done using parallel processing i.e. for loop block for each row is run independently by the worker processes utilizing the power of multiprocessor systems.

3.2 Second Order Methods

Traditional optimization methods such as Gradient Descent methods use only first order derivative to find the optimum value. To address non-linearity of the objective functions (cost functions), we move over to second order methods. These methods improve convergence rates.

For minimizing a continuously differentiable function $F : R^d \rightarrow R$, using Gradient Descent method, the update equation used is stated below -

$$w_{k+1} \leftarrow w_k - \alpha_k \nabla F(w_k)$$

For the same update, if we use second order method, the update equation used is

$$w_{k+1} \leftarrow w_k - \alpha_k \frac{\nabla F(w_k)}{\nabla^2 F(w_1)}$$

Notice that for a convex function, the optimum value can be found out by single iteration starting from any initial w_1.

3.3 Algorithm

We now describe the algorithm in detail.

Algorithm 1 Hessian Free Optimization Method

1: Choose an initial weight vector w_1.
2: Choose constant $\rho \in (0,1)$.
3: Remove all linearly dependent features and rows of data with all zeros.
4: Compute prediction function, cost value and $\nabla F(w_1)$ for current weight vector.
5: Compute s by solving Hessian-free CG to solve

$$(\nabla^2 F(w_1))s = -\nabla F(w_1)$$

until $max_c g$ iterations have been performed or a trial solution yields

$$||(\nabla^2 F(w_k))s + \nabla F(w_k)||_2 \leq \rho ||\nabla F(w_k)||_2$$

6: Update the vector using the following equation

$$w_2 \leftarrow w_1 + s$$

7: **return** w_2 ▷ $w_2 = w_*$

4 Experiment

To compare the results of traditional approaches, with that discussed, we have performed a number of classification experiments using Logistic Regression on classical data sets. The data sets and the classification method have been discussed below. Here, we consider all operations are batch operations.

4.1 Logistic Regression

Logistic regression can be seen as a special case of the generalized linear model, analogous to linear regression. It estimates the probabilities of the outcome based upon the input variables and assigns the class that is more probable. The various equations used in Logistic Regression are stated below:

4.1.1 Prediction Function

The prediction function for Logistic Regression is given by,

$$h_\theta(x) = g(\theta^T x)$$
$$g(z) = \frac{1}{1+e^{-z}}$$
$$z = \theta^T x$$

Here, $g(z)$ represents the sigmoid function.

4.1.2 Cost Function

The cost function for Logistic Regression is given by,

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}[-y^{(i)} log(h_\theta(x^{(i)})) - (1-y^{(i)}) log(1-h_\theta(x^{(i)}))]$$

4.1.3 First Order Derivative

$$
\begin{aligned}
h_\theta(x) &= g(\theta^T x) \\
g(z) &= \frac{1}{1+e^{-z}} \\
z &= \theta^T x \\
\frac{\partial h_\theta(x)}{\partial \theta} &= (h_\theta(x))(1-h_\theta(x)).x \\
J(\theta) &= \frac{1}{m}\sum_{i=1}^{m}[-y^{(i)} log(h_\theta(x^{(i)})) - (1-y^{(i)}) log(1-h_\theta(x^{(i)}))] \\
\frac{\partial J(\theta)}{\partial \theta_j} &= \frac{1}{m}\sum_{i=1}^{m}[-y^{(i)}\frac{1}{h_\theta(x^{(i)})}h_\theta(x^{(i)})(1-h_\theta(x^{(i)}))x_j^{(i)} \\
&\quad -(1-y^{(i)})\frac{1}{(1-h_\theta(x^{(i)}))}h_\theta(x^{(i)})(1-h_\theta(x^{(i)}))(-x_j^{(i)})] \\
&= \frac{1}{m}\sum_{i=1}^{m}[-y^{(i)}(1-h_\theta(x^{(i)}))x_j^{(i)}) + (1-y^{(i)})h_\theta(x^{(i)})x_j^{(i)}] \\
&= \frac{1}{m}\sum_{i=1}^{m}[-x_j^{(i)}y^{(i)} + x_j^{(i)}y^{(i)}h_\theta(x^{(i)}) + x_j^{(i)}h_\theta(x^{(i)}) - x_j^{(i)}y^{(i)}h_\theta(x^{(i)})] \\
&= \frac{1}{m}\sum_{i=1}^{m}[h_\theta(x^{(i)}) - y^{(i)}]x_j^{(i)}
\end{aligned}
$$

4.1.4 Hessian

$$
\begin{aligned}
\frac{\partial J(\theta)}{\partial \theta_j} &= \frac{1}{m}\sum_{i=1}^{m}[h_\theta(x^{(i)}) - y^{(i)}]x_j^{(i)} \\
\frac{\partial^2 J(\theta)}{\partial \theta_j \partial \theta_k} &= \frac{\partial \frac{1}{m}\sum_{i=1}^{m}[h_\theta(x^{(i)})x_j^{(i)} - y^{(i)}x_j^{(i)}]}{\partial \theta_k} \\
&= \frac{1}{m}\sum_{i=1}^{m}[h_\theta(x_j^{(i)})(1-h_\theta(x_j^{(i)}))x_k^{(i)}x_j^{(i)}]
\end{aligned}
$$

4.2 Data Sets

The various data sets that have been used to test results are stated below:

4.2.1 Student Admission

Given historical data from previous applicants, the objective is to determine each applicant's chance of admission based on their results on two exams. For each training example, you have the applicant's scores on two exams and the admissions decision. The task is to build a classification model that estimates an applicant's probability of admission based the scores from those two exams.

4.2.2 Diabetes

The Pima Indians Diabetes Dataset [2] involves predicting the onset of diabetes within 5 years in Pima Indians given medical details. It is a binary (2-class) classification problem. The number of observations for each class is not balanced. There are 768 observations with 8 input variables and 1 output variable. Missing values are believed to be encoded with zero values.

4.2.3 Sonar

The Sonar Dataset [3] involves the prediction of whether or not an object is a mine or a rock given the strength of sonar returns at different angles. It is a binary (2-class) classification problem. The number of observations for each class is not balanced. There are 208 observations with 60 input variables and 1 output variable.

4.2.4 Bank Note Authentication

The Banknote Dataset [4] involves predicting whether a given banknote is authentic given a number of measures taken from a photograph. It is a binary (2-class) classification problem. The number of observations for each class is not balanced. There are 1,372 observations with 4 input variables and 1 output variable.

4.2.5 Ionosphere

The Ionosphere Dataset [5] requires the prediction of structure in the atmosphere given radar returns targeting free electrons in the ionosphere. It is a binary (2-class) classification problem. The number of observations for each class is not balanced. There are 351 observations with 34 input variables and 1 output variable.

The results and comparisons on time taken and accuracy of the two approaches have been discussed in the next section.

5 Observations and Results

The below screen shots show the execution time and accuracy of our approach as opposed to the traditional Gradient Descent approach as followed by the *fminunc* function in MATLAB. As can be seen, accuracy similar to that of the *fminunc* function is achieved in single iteration.

5.1 Student Admission Data Set

Criteria	*fminunc*	Proposed Algorithm
# of iterations	37	1
Time elapsed (in sec)	0.03	0.09
Accuracy (%)	89	90
Cost at w_*	0.203498	0.331473

Table 1: Comparison of MATLAB function *fminunc* and proposed algorithm

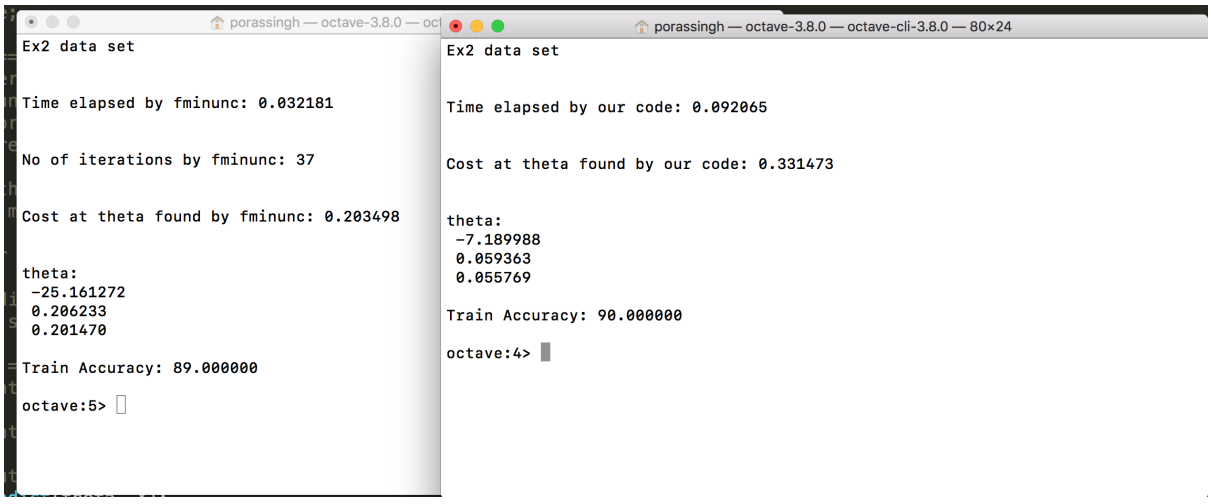

Figure 2: Results of *fminunc* on left and the proposed algorithm on right on *Student Admission* data set.

5.2 Diabetes Data Set

Criteria	*fminunc*	Proposed Algorithm
# of iterations	79	1
Time elapsed (in sec)	0.07	15.40
Accuracy (%)	78.25	77.99
Cost at w_*	0.470993	0.491225

Table 2: Comparison of MATLAB
function *fminunc* and proposed algorithm

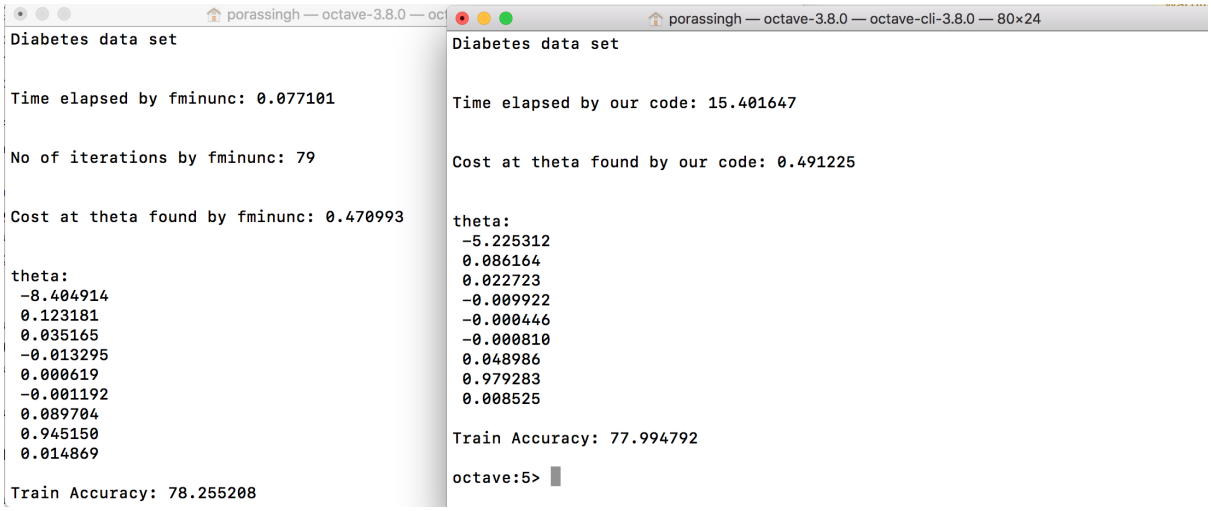

Figure 3: Results of *fminunc* on left and the proposed algorithm on right on *Diabetes* data set.

5.3 Sonar Data Set

Criteria	*fminunc*	Proposed Algorithm
# of iterations	248	1
Time elapsed (in sec)	0.21	389.40
Accuracy (%)	92.30	87.50
Cost at w_*	0.159572	0.360533

Table 3: Comparison of MATLAB
function *fminunc* and proposed algorithm

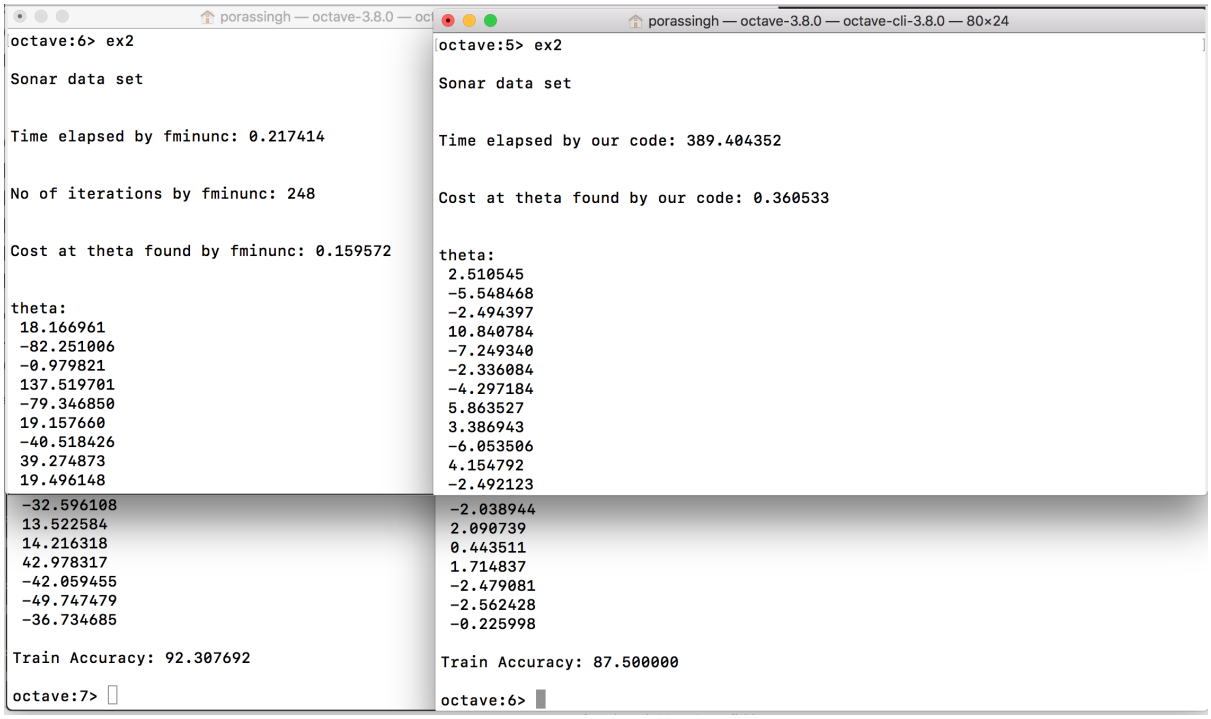

Figure 4: Results of *fminunc* on left and the proposed algorithm on right on *Sonar* data set.

5.4 Bank Note Authentication Data Set

Criteria	*fminunc*	Proposed Algorithm
# of iterations	56	1
Time elapsed (in sec)	0.05	5.00
Accuracy (%)	99.12	97.66
Cost at w_*	0.018557	0.195644

Table 4: Comparison of MATLAB function *fminunc* and proposed algorithm

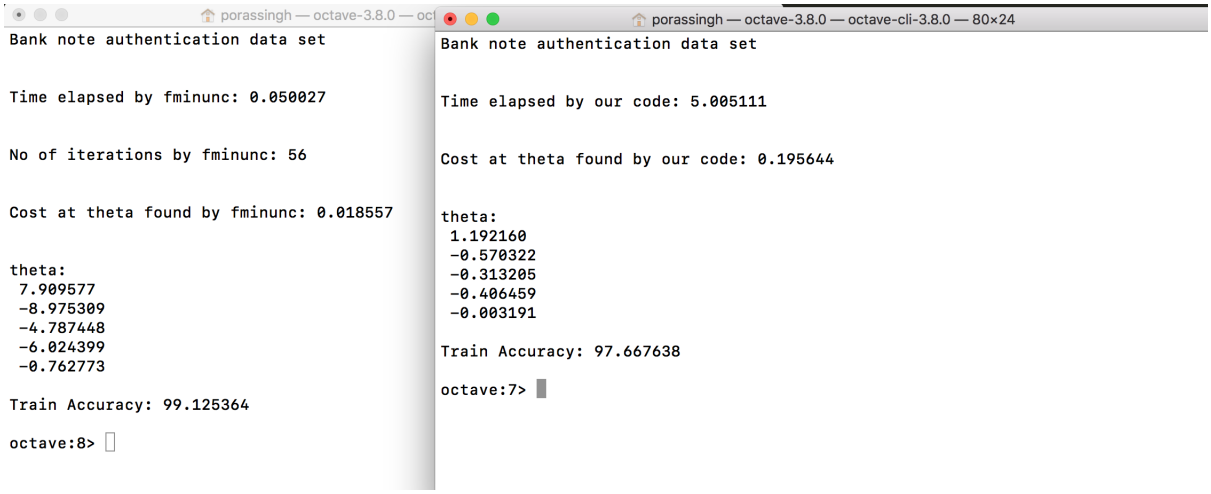

Figure 5: Results of *fminunc* on left and the proposed algorithm on right on *Bank Note Authentication* data set.

5.5 Ionosphere Data Set

Criteria	*fminunc*	Proposed Algorithm
# of iterations	190	1
Time elapsed (in sec)	0.16	210.02
Accuracy (%)	93.73	90.02
Cost at w_*	0.158212	0.319442

Table 5: Comparison of MATLAB function *fminunc* and proposed algorithm

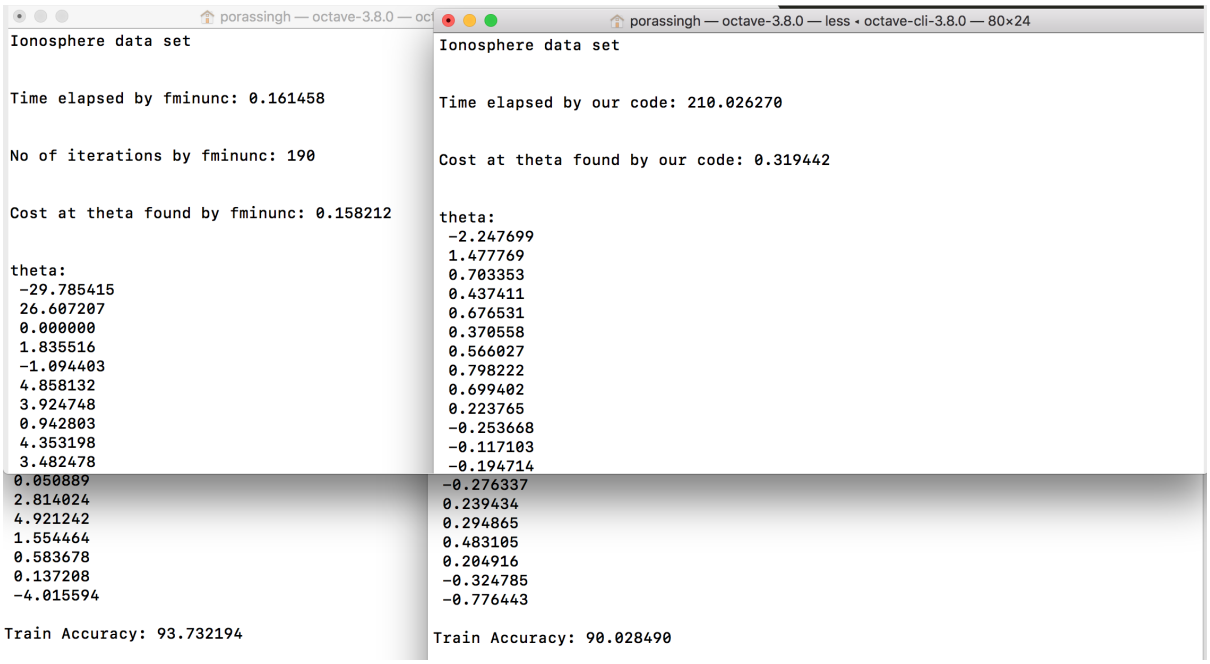

Figure 6: Results of *fminunc* on left and the proposed algorithm on right on *Ionosphere* data set.

References

[1] Leon Bottou, Frank E. Curtis, Jorge Nocedal, *Optimization Methods for Large-Scale Machine Learning*, June 6, 2017

[2] https://raw.githubusercontent.com/jbrownlee/Datasets/master/pima-indians-diabetes.data.csv

[3] https://archive.ics.uci.edu/ml/machine-learning-databases/undocumented/connectionist-bench/sonar/sonar.all-data

[4] http://archive.ics.uci.edu/ml/machine-learning-databases/00267/data_banknote_authentication.txt

[5] https://archive.ics.uci.edu/ml/machine-learning-databases/ionosphere/ionosphere.data