Review of Large-Scale Coordinate Descent Algorithms for Multi-class Classification with Memory Constraints

by
Aleksandar Jovanovich

Submitted in Partial Fulfillment of the Requirements

for the Degree of

Master

of

Computing and Information Systems

Program

YOUNGSTOWN STATE UNIVERSITY

May, 2013

Review of Large-Scale Coordinate Descent Algorithms for Multi-class Classification with Memory Constraints

Aleksandar Jovanovich

Signature:

Aleksandar Jovanovich, Student Date

Approvals:

Alina Lazar, Thesis Advisor Date

Gwang-Hwa Chang Committee Member Date

Graciela Perera, Committee Member Date

Bryan DePoy, Interim Dean of School of Graduate Studies and Research Date

Abstract

Big data can impact the performance of many standard measures used for classification. Specifically the efficiency of multi-class classification algorithms when the dataset is too large to fit into limited memory available needs to be explored. Different algorithms with varying complexity have been proposed in the literature. Two of the most recognized classification algorithms, batch learning and online learning have emerged as the most consistent options when solving for a multi-class problems. Presently a gap in the documentation of such algorithms exists in the literature available. Furthermore, the recent development of the online multi-class solver warrants a detailed examination. This thesis will address both concerns, providing detailed documentation of the analysis, comparisons of the algorithms, plots of the results, as well as a discussion about the findings.

Acknowledgments

First, I would like to thank my parents for allowing me to realize my own potential. All the support they have provided me over the years was the greatest gift anyone has ever given me.

Also, I need to thank Carolyn O’Rourke my fiancée, who taught me the value of hard work and an education. Without her, I may never have gotten to where I am today.

I would also like to acknowledge my committee members: Dr. Graciela Perera and Guang-Hwa Chang, who graciously agreed to serve on my committee.

Finally, I would like to thank Alina Lazar, who took the time to share her knowledge and appreciation of machine learning. Also, for reading this thesis, for which I owe her a new box of pens.

Table of Contents

Abstract 1

Acknowledgments 2

Table of Contents 3

List of Figures 6

CHAPTER 1: INTRODUCTION 7

1.1 Subject 7

1.2 Purpose 8

1.3 Scope and Limitations 8

1.4 Plan of Development 9

CHAPTER 2: LITERATURE REVIEW 10

2.1 Definitions 10

2.2 Binary Classification 11

2.3 Multi-class Classification 12

2.3.1 One-Against-All 13

2.3.2 Pairwise comparison 14

2.4 Coordinate Descent 15

2.4.1 Line Search ... 16

2.5 Related Work ... 17

CHAPTER 3: METHODOLOGY ... 18

3.1 Block Minimization ... 18

3.1.1 Support Vector Machines Learning ... 21

3.2 Online Learning ... 24

3.2.1 Gradient Descent ... 27

CHAPTER 4: COMPARISON ... 29

4.1 Large-scale ... 29

4.1.1 SensIT Dataset ... 30

4.1.2 RCV1 Dataset ... 31

4.2 Study Design ... 33

4.3 Outcome Measures ... 33

4.4 Performance Measures ... 34

4.4.1 Accuracy ... 34

4.4.2 Training Time ... 35

4.5 Analysis ... 35

CHAPTER 5: RESULTS AND DISCUSSION ... 36

5.1 Datasets ... 36

5.1.1 Format ... 36

5.2 Performance Measures ... 38

5.2.1 Training Time ... 38

5.2.2 Accuracy ... 41

5.3 Discussion ... 41

Chapter 6: Conclusions ... 45

References ... 46

List of Figures

Figure 1: SensIT Class Frequency 30

Figure 2: RCV1 Class Frequency 31

Figure 3: Reuters Corpus Volume 1 Document 32

Figure 4: SensIT Training Time 38

Figure 5: RCV1 Training Time 39

Figure 6: SensIT Accuracy 40

Figure 7: RCV1 Accuracy 40

Figure 8: SensIT table of Statistics 42

Figure 9: RCV1 Table of Statistics 42

Figure 10: Average Run Time 43

Figure 11: Average Classification Accuracy 43

CHAPTER 1: INTRODUCTION

1.1 Subject

Big Data has affected the way statistical analysis is being conducted. Many real-world datasets contain hundreds or thousands of variables of interest which can contain hundreds of thousands or millions of records. Time spent on reading/writing between memory and disk becomes the bottleneck, rendering most algorithms inefficient (Yu et al. 2012). Even with the growing memory sizes of computers, a large data set can still be problematic. As a consequence, the complexity of analysis increasingly becomes unmanageable by using traditional machine learning algorithms. To extract useful knowledge from dense data makes the task of analysis time consuming. Coupled with the fact that most algorithms use a iteration process that cycles through the dataset multiple times, the process is seemingly impossible to finish.

In the past, classification models have been shown to handle large amounts of data well, and several optimization techniques have been applied to efficiently train data intensive models (Aly 2005). However the performance of these algorithms begins to decline when the data cannot be processed into memory (Yu et al. 2012). In these cases, training techniques that deal well with memory limitations become critical.

Recent progress has been made in the development of techniques to optimize over memory constrained systems. Presently there exists an influx of algorithms that optimize with processing constraints in the literature available. Yu et al. (2012) draws comparisons from large-scale solvers with or without memory constraints. The results from the comparison suggest that performance wise both cases yielded similar results. Similar findings were discussed in Langford and Zhang (2009), when online algorithms were tested. This thesis will review the methodology behind these more recent memory constrained optimization algorithms in a classification framework.

1.2 Purpose

Within the literature a gap exists in the documentation of large scale coordinate descent algorithms for multi-class classification with memory constraints. The purpose of this paper is to provide a significant review of the existing methods, providing details about the theory, implementation, and overall performance of the algorithms. In addition the algorithms will be compared, testing the speed and classification accuracy of the multi-class solvers. In the end this thesis is intended to "fill the gap" that is currently present in the literature.

1.3 Scope and Limitations

This research will provide significant contribution to the academic community. The review presented will detail the theories supporting multi-class classification, the framework of such algorithms, testing results, as well as a comparison of the performance measured. As the research is not limited to a single application, it will act as a reference to various implementations outside the scope of this review. Every study has some limitations that narrow the scope of a research. The limitations of this research are discussed as follows:

- As the study is based on two datasets, there is limitation of generalizing the results on the basis of a small sample size. It may not be appropriate to do so.

- The research is based on the premise that entire dataset cannot fit into memory. The availability of memory available will alter the performance of the multi-class solvers, which again makes the generalization part difficult as we are talking about all cases with memory constraints.

- Further the availability of time and resource poses another limitation as classification problems extend beyond the scope of this paper. It is not possible to conduct a research on whole of the multi-class classification and thus the findings of the study depend upon the availability of data with reference to time and accessibility.

1.4 Plan of Development

The paper is organized in the following manner: Chapter 2 provides a literature review of existing work. Chapter 3 presents methodology behind the approach taken in this thesis. In the next chapter, Chapter 4 draws comparison between the Large-Scale Coordinate Descent Algorithms for Multi-class Classification with Memory Constraints. Results and discussions follow in Chapter 5, and finally the conclusion in Chapter 6. This thesis lists references last.

CHAPTER 2: LITERATURE REVIEW

2.1 Definitions

Supervised learning is mainly concerned with predicting class information based on observed information. For example, predicting part-of-speech based on sentences. It employs statistical methods to construct a prediction rule from labeled training data.

Data is comprised of points, called instances that are described by their values on some set of features. The space that these instances live in is called the feature space, and is typically denoted by X. The *label* of an instance is the predicted value of X, and denoted as the space of possible labels Y (Bayes and Regression 2010).

Extension of a base *learning problem* is some unknown data distribution D over $X \times Y$, coupled with a loss function $\ell(y', y)$ measuring the loss of predicting y when the true label is y'. The *loss function is* a function that maps an event onto a real number intuitively representing some "cost" associated with the event (Li and Yang 2003). Typically it is used for parameter estimation, and the event in question is some function of the difference between estimated and true values for an instance of data.

The learning problem is solved by a *learning algorithm,* which takes a set of labeled training instances of the form $(x, y) \in X \times Y$ and produces a predictor $f: X \rightarrow Y$.

$$\mathrm{E}_{(x,y)\sim D}\, \ell\,(f(x), y) \qquad (1)$$

The goal of the algorithm is to find the f such that, the expected loss is minimized (1).

2.2 Binary Classification

There are two base learning problems, defined for any feature space X: a regression and a binary classification problem. In this thesis only the binary classification problem is reviewed, more information about regression can be found in the work by Montgomery, Peck and Vining (2012).

A *binary classification problem* is defined by a distribution D over $X \times Y$, where the labels are assigned in only two classes, $Y = \{0, 1\}$. The goal of binary classification is to predict an unobserved binary label Y (Beygelzimer, Langford, and Zadrozny 2008).

$$e(h, D) = \boldsymbol{Pr}_{(x,y)\sim D} [\, h(x) \neq y] \tag{2}$$

By minimizing the *error rate* on D (2), a classifier $h: X \rightarrow Y$ is learned

A general binary classification method used to learn a real-valued loss function f(x), where $\mathbb{R}^d \rightarrow$ R such that a classification rule (3) is induced.

$$\mathrm{h}(x) = \begin{cases} 1 & f(\mathrm{x}) > 0 \\ -1 & otherwise \end{cases} \tag{3}$$

In supervised learning, the classifier $h(x)$, or its loss function $f(x)$, is learned from a set of labeled examples $\{(x1, y1) \dots (x_n, y_n)\}$ referred to as the *training dataset*. Its performance (classification error) should be evaluated on a separate set of correctly labeled data called the *testing dataset*.

2.3 Multi-class Classification

The principles of multi-class classification are founded upon the same principles of binary classification. Each case involves assigning a class label for each instance present. In the case of multi-class classification however, the label is assigned from a set that has more than two classes.

Formally, a k-class classification problem is defined by a distribution D over X × Y, where X is some feature space and Y = {1 . . . k} is the set of possible labels. The goal is to find a classifier h: X → Y minimizing the error rate on D (Beygelzimer, Langford and Zadrozny 2008).

A common approach used in multiclass learning is based on the reduction of a multi-class problem to a set of multiple binary classification problems (Bartlett and Tewari 2007). The process entails a transformation of a single problem into a set of multiple problems before being solved. Reduction approaches are best suited for binary learning algorithm, batch learning, online learning as well as any Bayesian based learning algorithm.

After the binary classification problems have been solved, the resulting set of binary classifiers must then be combined in some way. The studies conducted in this thesis will review the general framework of learning algorithms that unify all of these methods of reducing a multiclass problem to a binary problem.

2.3.1 One-Against-All

Perhaps the simplest such scheme is one-against-all (OAA). For a k-class problem the OAA method constructs k models, where each model separates a class from the rest. The ith model is trained with all of the binary instances pertaining to the ith class (Liu, Wang, and Zeng 2007). The final output of the one-against-all method is the class that corresponds to the highest output value. The general algorithm for OAA solvers is presented in Algorithm 1.

Algorithm 1 One-Against All

```
(Set of $k$-class training examples S, binary classifier learning algorithm $A$)
1. Set $S' = Ø$
2. For all $(x, y) \in S$
        2.1 For all $i \in \{1, . . . , k\}$
        2.2 Add a binary example $(x, i, 1(y = i))$ to $S'$
3. Return $h = A(S')$
```

Build k different binary classifiers. For the ith classifier, let the positive examples be all the points in class i, and let the negative examples be all the points not in class i. Let f_i be the ith classifier. Classify with:

$$f(x) = arg \max_{i} f_i(x) \quad (4)$$

The OAA approach to classification presented in Algorithm 1 does not learn *k* separate classifiers, only a single combined classifier on the union of all training data is learned (Rifkin and Klautau 2004). The binary reduction applied in this framework maps multi-class instances to binary instances, in principle transforming the original multiclass distribution *D* into an induced binary distribution *D′*. To draw a sample from *D′*, a multiclass example (*x*, *y*) from *D* is drawn as well as a random index i ∈ {1 . . . *n*}, and output (*x*, *i*›, 1(*y* = *i*)).

2.3.2 Pairwise comparison

Another approach used for multiclass to binary reductions is based on comparing only pairs of classes (Beygelzimer, Langford and Ravikumar 2009). The All-Pairs reduction starts by constructing 2 binary classifiers, one for every pair of classes. Given a training dataset *S* = {(*x*, *y*)}, the binary classifier for the (*i*, *j*) class pair is trained with dataset {(*x*, 1(*y* = *i*)) and y = *i* or *j*} to discriminate between classes *i* and *j*. Given a test instance, each of the binary classifiers predicts a winner amongst its two classes, and the class with the highest number of wins is chosen as the multiclass prediction, with ties broken randomly.

$$f(x) = arg\ max_i(\sum f_{ij}(x)) \tag{5}$$

Build *k* (*k*-1) classifiers, one classifier to distinguish each pair of classes *i* and *j*. Let f_{ij} be the classifier where class *i* are positive instances and class *j* are negative.

Beygelzimer, Langford, and Ravikumar (2009) propose the use of Error Correcting Tournaments (ECT) classifiers to solve for pairwise comparisons. The *m*-elimination tournament solves for the weighted classification problem in two phases. The first phase consists of *m* single-elimination tournaments over the *k* labels where a label is paired against another label at most once per round. Once an example has lost *m* times, it is eliminated and no longer influences. The second phase is a final elimination phase, where the winner is selected from the *m* winners of the first phase, usually by a single-elimination tournament that weighs the *m* winners based on the degree of redundancy.

2.4 Coordinate Descent

Algorithms use an optimization scheme to efficiently solve the multi-class learning problem and select the best label (with regard to some criteria) from some set of available alternatives (Bouman and Sauer 1996). In the simplest case, an optimization problem consists of maximizing or minimizing a real function by methodically choosing input values from within an allowed set and computing the value of the function. An optimization problem is solved when an element x_0 in D is found, given a function $f(x)$: $D \rightarrow \mathbb{R}$, such that for all x in D:

$$f(x_0) \leq f(x)\ (\textit{minimization}) \text{ or } f(x_0) \geq f(x)\ (\textit{maximization}) \tag{6}$$

Here the function $f(x)$ is termed the *objective function*. A possible solution that minimizes (or maximizes) the objective function is called an *optimal solution.* The solution to the optimization problem is generally located through an iterative process (Liu, Wang, and Zeng 2007). The procedure generates a sequence of improving approximate solutions for a class of problems and continues until some subsequence of iterations converges to the optimal solution.

2.4.1 Line Search

In optimization, the line search strategy is a basic iterative approach that finds an optimal solution to the objective function. The line search approach first finds a descent direction along which the objective function $f(x)$ will be reduced and then computes a step size that determines how far x should move along that direction. The descent direction is referred to as coordinate descent and can be computed by various methods, such as cyclical descent and gradient descent (Jovanovich and Lazar 2012).

Cyclical descent is based on the idea that the minimization of a multivariable function $f(x)$ can be achieved by minimizing it along one direction at a time with respect to the coordinate variables (Friedman, Hastie and Tibshirani 2010). Through cyclically iterative steps, each direction sequentially minimizes the objective function.

Gradient descent is an optimization algorithm that finds a local minimum of a function using the points in the direction of the greatest rate of increase of the scalar field (Langford, Li, and Zhang 2009). Steps are taken proportionally to the negative of the gradient of the function $f(x)$ at the current point.

2.5 Related Work

In this section related work is discussed pertaining to comparisons of Large-scale Classification problems. The topic has compiled a comprehensive library primarily for binary classification. As such there seems to be a gap in knowledge pertaining to multi-class classification. The techniques used for the comparisons were derived from strategies implemented in related work.

Yu et al. first introduced a comparison for large linear classification in his paper published in 2012. His comparison of SVM solvers and online-learning algorithms only extended to binary classification. The study defined one assumption that is significant in our comparison. Assuming that the amount of available memory is limited, entire datasets cannot be stored in memory, but can be stored in the disk of one computer. The size of the datasets used in this paper is large enough to satisfying this constraint, and must be accessed through the hard drive where they are stored.

Multi-class classification was addressed in the paper by Chang and Roth (2011). In the paper comparisons were drawn from both batch learning as well as online-learning algorithms. Unfortunately Vowpal Wabbit at that time was limited to binary classification and no comparisons were drawn. Instead the focus was primarily on block minimizing algorithms, of which LIBLINEAR was deemed superior to several other algorithms. Recently Jovanovich and Lazar (2013) implemented a comparison of batch vs. online learning in a multi-class setting with limited memory.

CHAPTER 3: METHODOLOGY

A more challenging situation for large linear classification is to deal with data sets that cannot fit in memory. Prevailing training algorithms often need to iteratively access data, so without enough memory, the training time will be huge (Yu et al. 2012). Handling data beyond the memory capacity remains a challenging research issue. According to Langford et al. (2009), present approaches to handle large data can be roughly categorized to two types. The first approach solves classification problems through batch learning. The second approach considers online learning algorithms. Because data may be used only once, this type of method can effectively handle the memory issue. Both algorithms attempt to solve the classification problem of assigning labels from Y to instances X (Bartlett and Tewari 2007).

3.1 Block Minimization

Optimization through block minimization has been used in the past to efficiently deal with data to large too fit into memory. The process entails using an optimization technique based on block minimization. The term “block” refers to partitions of the dataset that can be read through the memory available. The size and content of each block varies from approach to approach. Pérez-Cruz, Figueiras, and Artes (2004) propose the use of “double chunking,” where data is partitioned into both “large chunks” and “small chunks.” Another approach described by Chang and Roth (2011) uses selective sampling for block minimization. By selecting only significant instances, the goal is to minimize the size of data blocks and speed up the iteration process. The sizes of the blocks are determined by the known memory constraints. Each instance is read and randomly assigned a block.

Algorithm 2 explains the process for data splitting. Yu et al. (2012) suggest a framework for block minimization that is also used for testing in this thesis. In this approach the amount of memory available for processing correlates to the size of blocks. The framework consists of 2 steps that split the data and read the blocks into memory, before solving for classification through an iterative process. The algorithm can be summarized as following:

Algorithm 2 Framework for Block Splitting

```
1. Decide m and create $m$ empty files
2. For $i$ = 1…
        2.1 Convert $x_i$ to a binary format $x_{i.}$
        2.2 Randomly choose number $j$ {1... $m$}.
        2.3 append $x_i$ into the end of the $j^{th}$ file.
```

The goal is to solve this problem in a way that (a) allows for each block *B* stored in files to be handled individually by the processor or machine, and (b) does not involve transfer of the block over the network.

Block minimization is a classical method of machine learning (Bertsekas 1999). Each step of this method updates a block of features, and is modified here to corresponding blocks of a contiguous chunk of data. The block minimization framework is summarized in Algorithm 3. The step of working on a single block is denoted as an *inner iteration*, while the *m* steps of going over all blocks as an *outer iteration*.

Algorithm 3 Block Coordinate Descent

```
1. Split $\{1, . . . , l\}$ to $B_1 , . . . , B_m$ and store data into $m$ files accordingly.
2. Set initial $\alpha$ or $w$
3. For $k = 1,2, . . .$ (outer iteration)
        For $j = 1, 2, . . . , m$ (inner iteration)
        3.1. Read $x_r , \forall r \in B_j$ from disk
        3.2 conduct operation on $\{x_r \mid r \in B_j\}$
        3.3 Update $\alpha$ or w
```

Let $\{B_1. . . B_m \}$ be a partition of all data indices $\{1, . . . , l\}$. The block size is adjusted according to the memory constraint so that B_j can fit in memory. These *m* blocks, are stored as *m* files and are loaded when required (Chang and Roth 2011). Then at each step operations are implemented using one block of data, followed by an update to *w* or α according to if the primal or the dual problem is measured. The iteration round is then complete. The process will continue, repeating the iteration steps until the algorithm converges at the optimal solution.

3.1.1 Support Vector Machines Learning

One of the most prominent learning algorithms associated with block minimization is Support Vector Machines (SVMs) learning. Proposed by Cortes and Vapnik in 1995, the algorithm has since grown into one of the most widely used learning algorithm in the world (Bottou and Lin 2007). The implementation and broad uses of SVMs have been well documented in the years since past.

SVMs are studied in this work for the reason that it is one of the most used classifiers. Given a training set $\{(x_i , y_i)\}_{i=1}^{l}$, $x_i \in R^n$, $y^i \in \{-1, +1\}$, SVMs solve the following optimization problem:

$$\min_{w} \quad \frac{1}{2} w^{T} w + C \sum_{i=1}^{l} \xi(w; x_{i}, y_{i}) \tag{7}$$

where $\xi(w; x_{i}, y_{i})$ is a loss function, $C > 0$ is a penalty parameter (Chang et al. 2008). Equation (7) is often referred to as the primal form of SVMs.

In order to optimize through block minimization only the dual form of SVMs must be used (Chang and Roth 2011). By examining the dual form of the optimization problem the entire algorithm is written in terms of only inner products between input feature vectors. Updates to the weight vector w, which corresponds to the entire data set treating instances uniformly prevent the primal form of SVMs to be used (Shalev-Shwartz et al.

2007). Solving the dual problem generates efficient learning in very high dimensional spaces. The framework for solving the dual SVMs is presented in Algorithm 4.

```
Algorithm 4 Framework for Dual SVMs
(Details are shown for steps 3.2 and 3.3)
  3.2 Exactly or approximately solve the sub-problem (1) to obtain $d^*_{Bj}$
        Where $\alpha B_j \leftarrow \alpha B_j + d^*_{Bj}$
  3.3 Update w by (3)
```

Let $B_j = \{1 \, . \, . \, . \, , l\} \backslash B_j$ and $d\bar{B}_j$ be the sub-vector of d comprising d_i, $i \in B_j$. At each inner iteration step the following sub-problem is solved:

$$\min_{d\bar{B}j} f(\alpha + d)$$

Subject to: $d_{Bj} = 0 \text{ and } 0 \leq \alpha_i + d_i \leq C, \forall \in B_j$ (8)

αB_j is updated by solving (8) for each block. The iteration round k is then complete after w is updated. To update w, if d_{Bj} is an optimal solution for the sub-problem, then:

$$w \leftarrow w + \sum_{r \in Bj} d^*_r \, y_r \, x_r \tag{9}$$

while solving for the *objective function* (10).

$$w \equiv \sum_{i=1}^{l} \alpha_i y_i x_i \qquad (10)$$

The iteration process continues until optimization is reached, converging when one of two conditions is met (Yu et al. 2012). The first condition states that optimization is complete when the sub-problem for each block is solved and the solutions converge. The second condition is a stopping criterion. Usually a finite number of iterations are chosen, or an accuracy threshold is obtained.

LIBLINEAR addresses both conditions while solving for the sub-problem (8). The software contains a library with tools used for SVM classification when data cannot fit into memory (Yu et al. 2012). By sequentially selecting one variable for update and fixes others inside the block the memory constraint can be solved. The framework shown in this paper is explained by Yu et al. (2012). LIBLINEAR uses a SVM coordinate descent method and solver to update instances in block B_j before solving for algorithm 4.

Using reduction scheme discussed early (from multi-class to binary), the Dual SVM framework from Algorithm 4 is implemented. If data can fit in memory the optimization problem by Crammer and Singer (2002) can be solved by a dual coordinate descent method, which is available in LIBLINAR.

Algorithm 5: One-Against-All Multi-class Block Minimization Framework

```
(We assume the $k$ class labels are $1 \dots k$)
1. Split $\{1 \dots l\}$ to $B_1 \dots B_m$, and store data into $m$ files accordingly
2. Set initial $\alpha^1 \dots \alpha^K$ and $w^1 \dots w^K$ , where $k$ is the number of classes
3. For $k = 1, 2 \dots$ (outer iteration)
        For $j = 1 \dots m$ (inner iteration)
      3.1. Read $x_r$, $\forall r \in B_j$ from disk
      3.2. For $t = 1 \dots K$
            3.2.1. Use $B_j \equiv \{x_r \mid r \in B_j$ and $y_r = t\}$ as positive and $B_j \backslash B_j^t$ as negative data
            3.2.2. Conduct certain training operations, and update $\alpha^t$ and $w^t$
```

To apply the one-against-the rest approach for a k-class problem, k classifiers must be trained. An implementation to save the disk access time is to train k models together and not separate. Each block B_j to $B_j^1 \dots B_j^K$ is partitioned according to the class information. Then, K sub-problems are solved simultaneously using B_j^t as positive data and $B_j \setminus B_j^t$ as negative data to update vectors w^t and α^t.

3.2 Online Learning

Online learning algorithms were proposed as fast alternatives to SVMs. Online learning algorithms are used to efficiently classify data by building a weight model derived from sequentially received training instances. Compared to block minimization which solves for the sub-problem of each block, online learning updates instances through the use of a cache file. Each iteration round updates the cache file where the weight model is stored. The algorithm classifies each instance, and uses the new "instance-label pair" to update and improve the stored model (Tewari and Bartlett 2007). This method is expected to accurately predict the labels of instances that are not part of the training set. The framework for the general online learner proposed by Beygelzimer, Langford and Zadrozny (2008) is presented in Algorithm 6 below.

Algorithm 6 Online-learning

```
(In the setting of standard online learning, we are interested in sequential prediction
problems where repeatedly from $i = 1, 2 . . .$)
1. An unlabeled example $x_i$ arrives
2. We make a prediction based on existing weights $w_i \in R^d$
3. We observe $y_i$, let $z_i = (x_i, y_i)$, and incur some known loss $L(w_i, z_i)$ that is convex in
   parameter $w_i$
4. Update weights according to some rule $w_{i+1} \leftarrow f(w_i)$
```

By finding the update rule $f(x)$, the sum of loses will be bound (11).

$$\sum_{i=1}^{T} L(w_i, z_i) \tag{11}$$

More formally, an online algorithm descends through the dataset in a sequence of trials. Each trial can be decomposed into three steps. First the algorithm receives an instance. Second the algorithm predicts the label of the instance. Third the algorithm receives the true label of the instance (fan et al. 2008). The third stage is the most crucial as the algorithm can use this label feedback to update its hypothesis for future trials. The goal of the algorithm is to minimize some performance criteria. Because on-line learning algorithms continually receive label feedback, the algorithms are able to adapt and learn in difficult situations (Shalev-Shwartz et al. 2007). Many online algorithms can give strong guarantees on performance even when the instances are not generated by a distribution. As long as a reasonably good classifier exists, the online algorithm will learn to predict correct labels.

The concept of online learning has long been used by researchers. Various strategies were proposed to optimize online learning algorithms, most of which extend the original purpose of binary classification to multi-class learning. In the literature, the Lasso algorithm (Tibshirani, 1996) is commonly used to achieve optimization. The algorithm implements a loss function bound by a *convex constraint* $\|w\|_1 \leq s$. Furthermore a s*oft regularization constraint* can be bound by (12).

$$w = \underset{w}{argmin} \sum_{i=1}^{T} L(w_i, z_i) + g\|w\|_1 \tag{12}$$

More recently, Duchi and Singer (2008) propose a framework for empirical risk minimization with regularization called Forward Looking Sub gradients, where a regularized optimization problem is solved after every gradient-descent step. Shalev-Shwartz et al. (2007) exploited the dual formation of optimization to create a more efficient online learning algorithm. The Forgetron algorithm proposed by Dekel et al. (2006) is an online-learning algorithm that manages memory use by decaying the weights on previous examples and then rounding these weights to zero when they become small. The algorithm can perform well when a hyper plane exists that splits the data into two categories. This algorithm can be modified to allow infrequently change during the online learning trials. A Bayesian approach to learning is taken by Balakrishnan and Madigan (2008), where they approximate the posterior by a Gaussian distribution.

3.2.1 Gradient Descent

Online learning methods are very closely related to stochastic gradient methods, as they operate on only one single instance at each iteration step (Langford, Li, and Zhang 2009). Furthermore, many online learning rules can be perceived as an implementation of a stochastic gradient descent. Such methods have strong associations to the predictor, without minimize the SVM objective. Online Gradient Descent solves (7) with a different loss function (13).

$$max(-y_i w^T x_i, 0) \tag{13}$$

Both SVM (9) and SGD (13) maintain weight w, but they take different directions through the optimization process. The optimal solution is presented in (14).

$$w \leftarrow w - \eta \Delta w(y_i, x_i) \tag{14}$$

The parameter $\eta > 0$ in (14) is often referred to as the learning rate. $\eta\Delta w(y_i, x_i)$ is a sub-gradient of the objective function (15).

$$w^T w/2 + C max(1 - y_i w^T x_i, 0) \tag{15}$$

The above method has been widely used in online learning (Langford, Li, and Zhang 2009). One of the most widely successful implementation of online learning is Vowpal Wabbit (VW). John Langford and his colleagues at Yahoo! Research developed the package, a fast online-learning algorithm that uses stochastic gradient descent to handle very large datasets without ever needing to load the entire dataset into memory. The algorithm also requires less computational power and far fewer resources by learning through online gradient descent (Langford, Li, and Zhang, 2009). The algorithm includes functions for multi-class classification solvers OAA as well as ECT.

Vowpal Wabbit is said to be efficient for solving batch problems when optimization of the online algorithm over training data requires several iterations. The idea has been successfully applied to large-scale SVM formulations (Shalev-Shwartz et al., 2007).

CHAPTER 4: COMPARISON

In the present study, interest was primarily focused on the responsiveness of coordinate descent algorithms to train and predict classification of multi-class large-scale datasets.

4.1 Large-scale

Big Data is a collection of data sets so large and complex that it becomes difficult to process using traditional data processing applications (Howe et al. 2008). The challenges include capture, storage, search, analysis, and visualization. The trend to larger data sets is due to the additional information derivable from analysis of a single large set of related data.

Datasets grow in size in part because they are increasingly being gathered by a growing array of technology that includes: mobile devices, remote sensing, software logs, cameras, microphones, and wireless sensor networks Howe et al. (2008). The world's technological per-capita capacity to store information has roughly doubled every 3 years for the last 50 years, and as of 2012, every day 2.5 quintillion (2.5×1018) bytes of data were created.

In a research report Laney (2001) defined data growth challenges and opportunities as being three-dimensional*, i.e. increasing volume (amount of data), velocity (speed of data in and out), and variety (range of data types and sources).* Using this definition, the comparison presented will be based on two data sets of varying size and data types.

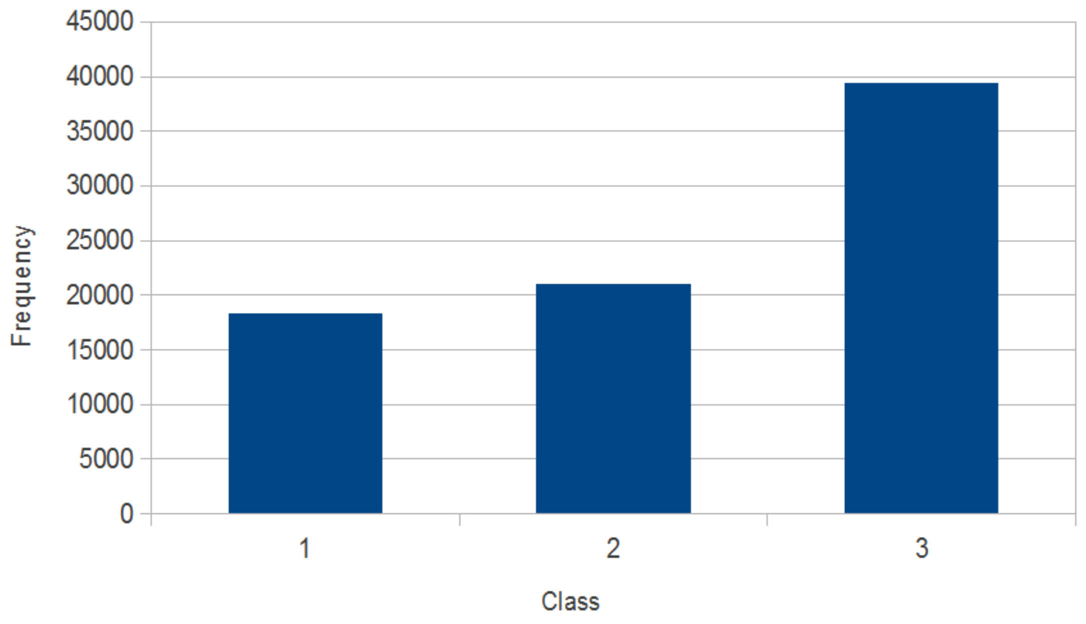

Figure 1: SensIT Class Frequency

4.1.1 SensIT Dataset

The SensIT Vehicle dataset is comprised of instances labeled as one of three classes. The data was extracted from sensor data collected during a real world experiment carried out at Twenty nine Palms, CA in November of 2001 (Duarte and Hu 2004). The sensors were used to obtain both acoustic and seismic activity from vehicles in the vicinity. Each vehicle was driven around a road while sensors collected information as they passed. Classes included in the training set presented in Figure 1 included: AAV3 (class 1), DW3 (class 2), and a third class for noise (class 3). In total there are 78,823 training samples, 19,705 testing samples and 50 features were extracted.

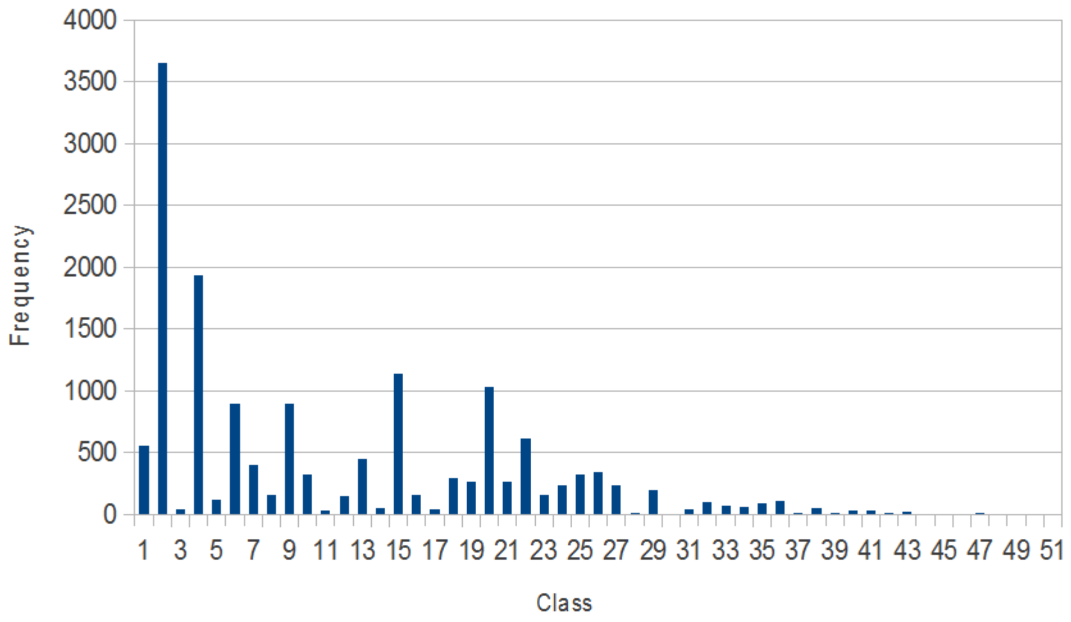

Figure 2: RCV1 Class Frequency

4.1.2 RCV1 Dataset

The RCV1 dataset (Figure 2) was used in part due to its 53-class problem. The RCV1 dataset is one of the most widely used test collection for text categorization research (Lewis et al. 2004). Reuters is the largest text and television news agency in the world. The editorial division produces some 11,000 stories a day in 23 languages. Stories are both distributed in real time and made available through online databases. The stories cover the range of content typical of a large English language international newswire, and vary from a few hundred to several thousand words in length. Figure 3 shows an example story.

```xml
<?xml version="1.0" encoding="iso-8859-1" ?>
<newsitem itemid="2330" id="root" date="1996-08-20"
xml:lang="en">
<title>USA: Tylan stock jumps; weighs sale of company.</title>
<headline>Tylan stock jumps; weighs sale of company.</headline>
<dateline>SAN DIEGO</dateline>
<text>
<p>The stock of Tylan General Inc. jumped Tuesday after the maker
of
process-management equipment said it is exploring the sale of the
company and added that it has already received some inquiries
from
potential buyers.</p>
<p>Tylan was up $2.50 to $12.75 in early trading on the Nasdaq
market.</p>
<p>The company said it has set up a committee of directors to
oversee
the sale and that Goldman, Sachs &amp; Co. has been retained as
its
financial adviser.</p>
</text>
<copyright>(c) Reuters Limited 1996</copyright>
<metadata>
<codes class="bip:countries:1.0">
<code code="USA"> </code>
</codes>
<codes class="bip:industries:1.0">
<code code="I34420"> </code>
</codes>
<codes class="bip:topics:1.0">
<code code="C15"> </code>
<code code="C152"> </code>
<code code="C18"> </code>
<code code="C181"> </code>
<code code="CCAT"> </code>
</codes>
<dc element="dc.publisher" value="Reuters Holdings Plc"/>
<dc element="dc.date.published" value="1996-08-20"/>
<dc element="dc.source" value="Reuters"/>
<dc element="dc.creator.location" value="SAN DIEGO"/>
<dc element="dc.creator.location.country.name" value="USA"/>
<dc element="dc.source" value="Reuters"/>
</metadata>
</newsitem>
```

Figure 3: Reuters Corpus Volume 1 Document

RCV1 is extracted from one of the online databases. It was intended to consist only of English language stories produced by Reuter's journalists between August 20, 1996, and August 19, 1997. It contains in total 534,135 manually categorized newswire stories. The dataset is divided into 518,571 training documents and 15,564 test documents, and contains 47,236 features (Lewis et al. 2004). Preparation of the dataset involved substantial verification and validation of the content, as well as attempts to remove false or duplicated documents.

4.2 Study Design

The study was designed to review not only the theory but the implementation and efficiency of the large-scale coordinate descent learning algorithms for multi-class classification problems when memory is limited. To ensure optimum performance and reliability the algorithms were compiled in C++ and run under the Linux operating system. To ensure the memory constraint condition, the hardware was scaled down to an Asus 5750G laptop. The CPU processor was limited to 2.2 GHz and only 6 GB of DDR3 was made available.

4.3 Outcome Measures

Classification predictions and processing time for training was recorded at the end of every iteration step. They were obtained from the best of three runs using the minimum value measure. In addition an overall assessment of efficiency was obtained for the optimum solution at the stopping threshold of 78 iterations by ranking the performance measures using a 3 point response scale ranging from best to worst.

4.4 Performance Measures

Performance measurement estimates the parameters under which algorithms converge at the optimal solution. The efficiency of the algorithms can be viewed as a process to reduce resource consumptions, including training time and accuracy.

4.4.1 Accuracy

It is surprisingly difficult to arrive at an adequate definition and measurement of accuracy. The best test of a classifier's value is its future performance (generalization), i.e. correctly classifying instances. Generalization is linked to classifier design, implementation and testing (Joachims 2001). Overall, complex classifiers fit 'noise' in the training data, consequently lowering the accuracy when presented with instances. Every so often it is necessary to accept reduced accuracy on the training data if it leads to increased accuracy.

Focusing on the generalization of a classifier differs from traditional statistical approaches which are usually judged by coefficient p-values or some overall goodness of fit such as R^2 (Stevens 2012). The statistical focus relates to the fit of the data to some pre-defined model and does not explicitly test performance on future data, generally because of the assumptions made about the parameters estimated by the statistics.

4.4.2 Training Time

Besides the classification performance, the training time is a second key factor that affects the aptness of a classification algorithm regarding an unknown dataset. An algorithm with a slightly lower accuracy is maybe preferred if its training time is significantly lower (Reif, Shafait and Dengel 2011). Additionally, an estimation of the required training time of a pattern recognition task is very useful if the result has to be available in a certain amount of time.

4.5 Analysis

Analysis of Large-Scale Coordinate Descent Algorithms for Multi-class Classification with Memory Constraints can be broken down into two main components, testing over SensIT dataset and testing over RCV1 dataset. The size of the SensIT dataset (training instances and features) is significantly smaller than RCV1, providing a baseline for performance measures. Pechyony, Shen and Jones state that using more samples and adding more features will boost performance in optimization. Following their lead, the comparisons made in this thesis follow the same structure.

Both LIBLINEAR and VOWPAL WABBIT support the coordinate descent algorithms discussed in this study. Multi-class Online Solvers OAA and ECT are implemented in VW, while Batch OAA is the primary solver for LIBLINEAR.

CHAPTER 5: RESULTS AND DISCUSSION

5.1 Datasets

Before the experiment can be run using the LIBLINEAR, the training dataset must be partitioned into smaller blocks. The optimal number of partitions was discovered to be 4 when training over the SensIT dataset and 8 blocks for the RCV1 dataset. Each block contains an even distribution of instances spread out across the partitions. Vowpal Wabbit on the other hand uses a cache file which fulfills the memory constraint without any parameter adjustments. The dataset does not need to be split or compressed; VW can access each instance without reading the entire datasets into memory.

5.1.1 Format

Both Datasets used in this thesis are available for download from LIBSVM Datasets webpage (Chang and Lin 2011). To date RCV1 is the largest available multi-class classification dataset in the database. This page contains a database of various classification, regression, and multi-label datasets stored in LIBSVM format (16).

*<class/target> [<attribute number> :< attribute value>]** (16)

For most sets, the scale for each attribute is set to [-1, 1] or [0, 1]. The testing data (if provided) is adjusted accordingly. The format of the data corresponds to LIBSVM which is able to be read by LIBLINEAR.

Vowpal Wabbit however reads data with a slightly different input format. The raw (plain text) input data for VW should have one example per line (Langford, Li, and Zhang 2009). Each example should be formatted according to the input format in (17).

[Label] [Importance [Tag]]|Namespace Features |Namespace Features... (17)

Where *Namespace = String [: Value]* and *Features = (String [: Value])**

In order to convert the LIBSVM formatted dataset to VW input format a script was written in C (18). The script can be implemented to seamlessly convert any LIBSVM formatted dataset (i.e. binary and multi-class classification, regression, and multi-label) into VW input format.

```
Zless [dataset] $1 | sed –e ‘s/\s/ | /’                (18)
```

To create a label dataset to be used for predication accuracy, the script (19) was also created and written in C.

```
Cat [dataset] | cut –d ‘ ‘ –f 1 > [output.txt]          (19)
```

5.2 Performance Measures

The performance measures for each iteration step are compared between the algorithms discussed in the review. VW uses a Gradient Descent method that is implemented in both VW OAA and VW ECT. Comparisons are made from the multi-class learners using gradient descent. Online learning VW OAA will also be compared to the batch learning LIBLINEAR OAA in terms of optimization approaches for limited memory constraints.

5.2.1 Training Time

Figure 4 shows the processing time per iterations over the SensIT dataset. The dataset represents a simple scenario where binary classification is extended into a 3-class problem.

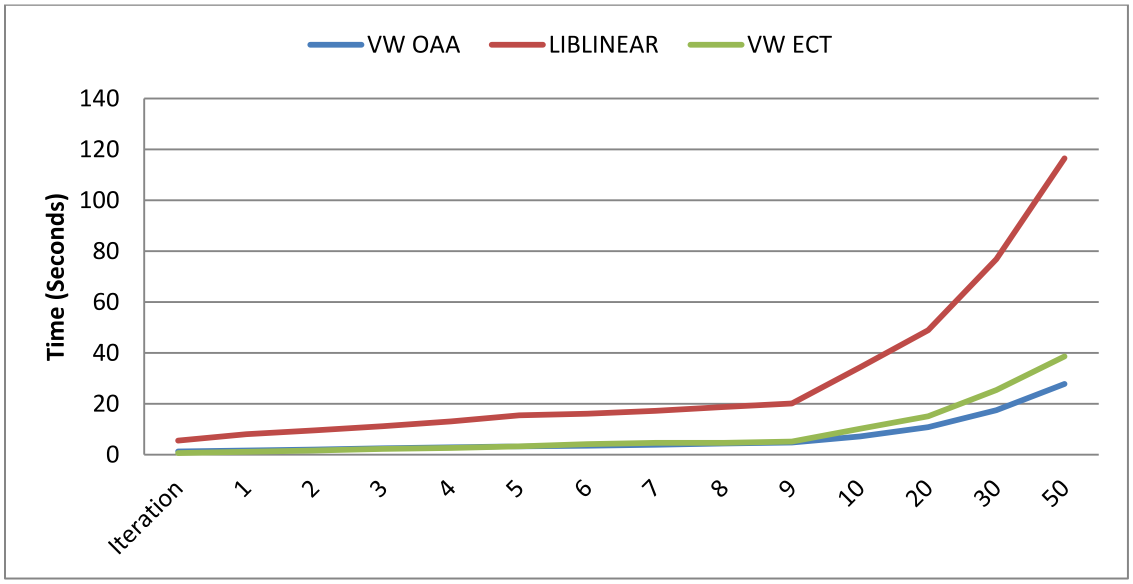

Figure 4: SensIT Training Time

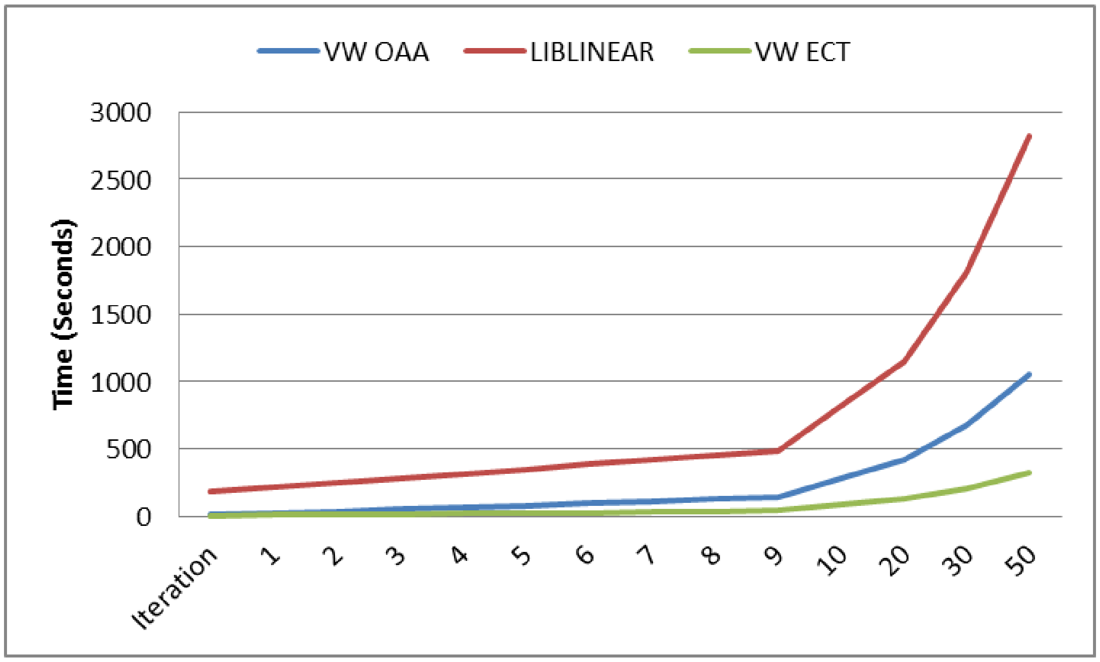

Figure 5: RCV1 Training Time

On average for Figure 4, the runtime per iteration for Vowpal Wabbit solving with OAA was the fastest averaging 0.36 seconds, while VW ETC came in second averaging 0.47 seconds. LIBLINEAR was a distant third averaging 1.48 seconds.

Figure 5 is consistent with the performances from the algorithms established with the tests run over the SensIT dataset in Figures 4. Average runtime per iteration for VW OAA increased to 27.26 seconds while LIBLINEAR took an average of 36 seconds per iteration. VW ECT managed an astounding 4.12 seconds per iteration. VW OAA was by far the most efficient classifier.

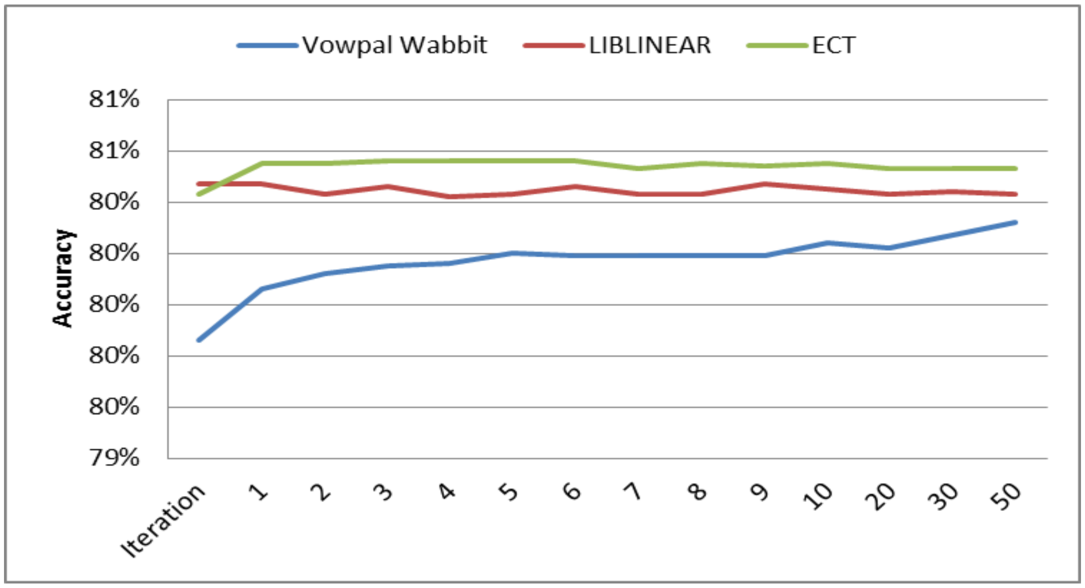

Figure 6: SensIT Accuracy

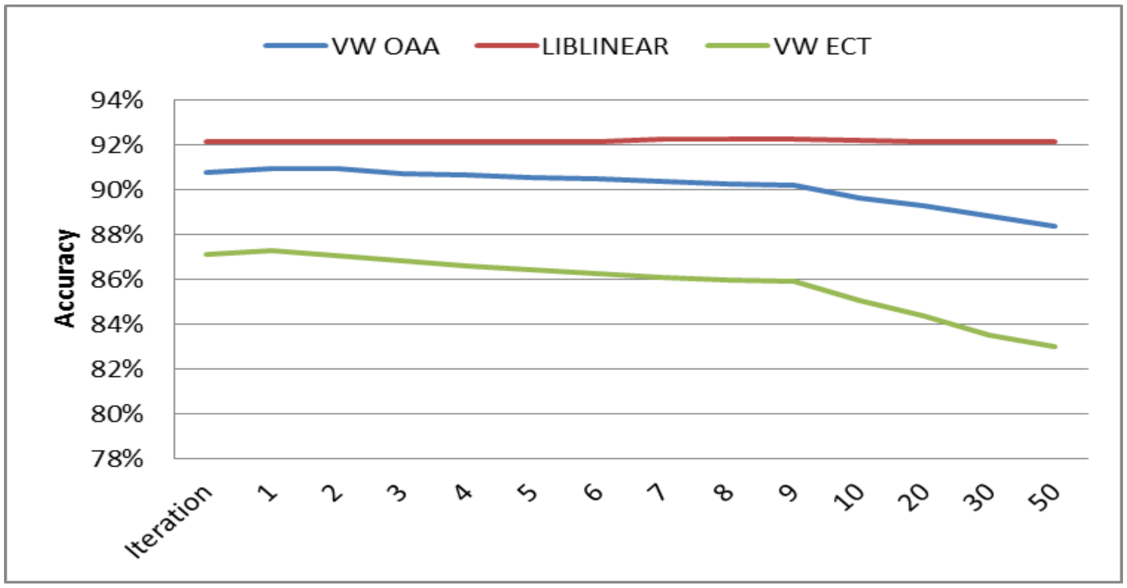

Figure 7: RCV1 Accuracy

5.2.2 Accuracy

While all algorithms scored over 80% over the SensIT dataset in Figure 6, accuracy classifying testing instances VW ETC scored highest most consistently. The biggest differences recorded can be seen between the first 5 iteration rounds. This is where the optimization approaches can be distinguished. Vowpal Wabbit ETC begins to maintain a constant accuracy as LIBLINEAR begins to degrade. As the Vowpal Wabbit algorithm passes through the dataset and the weights are updated from the initial label, the accuracy of classification begins to rise.

Moving on to the bigger dataset presented in Figure 7, the results indicate increased performance in both accuracy and speed. Classification accuracy raised an average of 8% when comparing the results of Figure 6 to 7. The larger dataset has had a positive impact on the accuracy rate of classification for the two algorithms. In this case, the lager dataset yields the highest accuracy rate when using LIBLINEAR. This is contrary to the results found in the SensIT experiment, in which VW ETC yielded the highest classification accuracy. In fact VWETC scored the lowest accuracy this time around.

5.3 Discussion

In both tests VW was 4 times as fast as LIBLINEAR. Both algorithms achieved higher levels of performance solving for the larger RCV1 classification problem. Figures 8 and 9 provide tables with a more detailed look.

SensIT Dataset								
Vowpal Wabbit			LIBLINEAR			ECT		
Iteration	Time	Acc	Iteration	Time	Acc	Iteration	Time	Acc
1	1.28	79.86%	1	5.52	80.47%	1	0.58	80.43%
2	1.64	80.06%	2	8.00	80.47%	2	1.15	80.55%
3	2.05	80.12%	3	9.53	80.43%	3	1.62	80.55%
4	2.46	80.15%	4	11.25	80.46%	4	2.23	80.56%
5	2.87	80.16%	5	13.03	80.42%	5	2.69	80.56%
6	3.29	80.20%	6	15.43	80.43%	6	3.30	80.56%
7	3.56	80.19%	7	16.09	80.46%	7	4.17	80.56%
8	3.95	80.19%	8	17.27	80.43%	8	4.71	80.53%
9	4.40	80.19%	9	18.76	80.43%	9	4.60	80.55%
10	4.81	80.19%	10	20.14	80.47%	10	5.14	80.54%
20	7.13	80.24%	20	34.28	80.45%	20	10.17	80.55%
30	10.83	80.22%	30	48.93	80.43%	30	15.10	80.53%
50	17.52	80.27%	50	76.93	80.44%	50	25.44	80.53%
78	27.81	80.32%	78	116.56	80.43%	78	38.68	80.53%

Figure 8: SensIT table of Statistics

RCV1 Dataset								
VW OAA			LIBLINEAR			VW ECT		
Iteration	Time	Acc	Iteration	Time	Acc	Iteration	Time	Acc
1	13.38	90.74%	1	180	92.13%	1	4.438	87.10%
2	27.85	90.94%	2	217	92.12%	2	8.749	87.25%
3	40.27	90.93%	3	251	92.12%	3	12.58	87.02%
4	53.67	90.72%	4	284	92.12%	4	15.89	86.80%
5	68.16	90.65%	5	318	92.12%	5	20.67	86.61%
6	82.01	90.55%	6	351	92.14%	6	23.14	86.43%
7	94.48	90.48%	7	384	92.15%	7	28.61	86.24%
8	114.12	90.34%	8	418	92.22%	8	33.72	86.10%
9	126.81	90.25%	9	451	92.23%	9	37.98	85.94%
10	143.59	90.16%	10	484	92.23%	10	40.38	85.88%
20	279.09	89.64%	20	816	92.18%	20	83.36	85.02%
30	416.09	89.29%	30	1147	92.14%	30	133.537	84.34%
50	681.63	88.79%	50	1811	92.14%	50	205.133	83.51%
78	1052.34	88.34%	78	2825	92.14%	78	322	82.99%

Figure 9: RCV1 Table of Statistics

Iteration runtime of both algorithms experienced an increase in time when compared to SensIT results. VW ECT was able to quickly optimize over the 78 step threshold limit, averaging only 4 seconds per round. Average runtime from both datasets is presented in Figure 10.

Average Time	VW OAA	LIBLINEAR	VW ECT
SensIT	0.36s	1.5s	0.49s
RCV1	13.49s	36.22s	4.13s
Total	**6.925s**	**18.86s**	**2.31s**

Figure 10: Average Run Time

Overall performance measures indicate the VW's use of gradient descent is better suited for large-scale classification with limited memory constraints. Figures 8 and 9 highlight in blue the highest accuracy achieved for the algorithms. In the latter case, LIBLINEAR achieved the highest accuracy rating recorded after iteration Round 9, where accuracy peaked at 92.23%. The classification accuracy improved on average in Figure 11.

Average Accuracy	VW OAA	LIBLINEAR	VW ECT
SensIT	80.16%	80.44%	80.54%
RCV1	90.13%	92.16%	85.80%
Total	**85%**	**86%**	**83%**

Figure 11: Average Classification Accuracy

Lower rates of classification accuracy can be contributed to the smaller training dataset size and lack of features. Frequency distribution presented in Figures 1 and 2 are unbalanced, as such the classification accuracy over the smaller SensIT data set was lower than the larger RCV1 dataset. The increase in instances correlated with the accuracy of overall classification. As more instances were added, classification accuracy improved.

Chapter 6: Conclusions

The following question about Fast Learning large-scale multi-class classification can be answered:

Question: Which algorithm is most efficient when there are constraints to the memory? When compared, Vowpal Wabbit is the most efficient multi-class classification algorithm. The results from the SensIT test case suggested that Vowpal Wabbit was the quicker algorithm while maintaining a slightly lower accuracy percentile than LIBLINEAR. Moving from the SensIT dataset to the larger RCV1, the results remained consistent. We have concluded that Vowpal Wabbit had a slight advantage in overall efficiency when there is a constraint placed of computer memory size. The Vowpal Wabbit OAA multi-class solver was by far the most efficient, ranking high in every test.

Performance measured from the implemented algorithms yielded relatively close results. Due to the small size of the experiment, further testing is needed for a thorough comparison. Testing over datasets that are more expansive, both in sample and feature size could be used for a more significant experiment. However, the size of the datasets used in this paper is adequate to provide conclusive results for the comparisons made.

References

Aly, M. 2005. Survey on multiclass classification methods. *Neural networks,* 1-9.

Argyriou, A., Herbster, M., and Pontil, M. 2005. Combining graph Laplacians for semi-supervised learning.

Bayes, N. and Regression, L. 2010. Fundamental Statistical Techniques. *Handbook of Natural Language Processing*, *2*: 189.

Bottou, L. and Lin, C. J. 2007. Support vector machine solvers. *Large scale kernel machines,* 301-320.

Bouman, C. A. and Sauer, K. 1996. A unified approach to statistical tomography using coordinate descent optimization. Image Processing, IEEE Transactions, 5(3): 480-492.

Chang, K. W., and Roth, D. 2011. Selective block minimization for faster convergence of limited memory large-scale linear models. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining,* 699-707.

Chang, C. and Lin, C. 2011. LIBSVM: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2(3): 27:1--27:27. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm

Cortes, C. and Vapnik, V. 1995. Support-vector networks. *Machine learning*, 20(3): 273-297.

Dekel, O. 2008. From online to batch learning with cutoff-averaging.

Douglas, L 2001. 3D Data Management: Controlling Data Volume, Velocity and Variety.

Duarte, M. F., and Hen Hu, Y. 2004. Vehicle classification in distributed sensor networks. *Journal of Parallel and Distributed Computing*, *64*(7): 826-838.

Fan, R. E., Chang, K. W., Hsieh, C. J., Wang, X. R., and Lin, C. J. 2008. LIBLINEAR: A library for large linear classification. *The Journal of Machine Learning Research,* 9: 1871-1874.

Friedman, J., Hastie, T., and Tibshirani, R. 2010. Regularization paths for generalized linear models via coordinate descent. *Journal of statistical software*, 33(1): 1.

Francis, M. 2012. Future telescope array drives development of exabyte processing.

Howe, D., Costanzo, M., Fey, P., Gojobori, T., Hannick, L., Hide, W., and Rhee, S. Y. 2008. Big data: The future of biocuration. *Nature*, 455: 47-50.

Joachims, T. 2001. A statistical learning learning model of text classification for support vector machines. *In Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, 128-136.

A. Jovanovich and A. Lazar 2013. Comparison of Fast Learning Large Scale Multi-Class Classification. *Midwest Artificial Intelligence and Cognitive Science Conference 2013.*

A. Jovanovich and A. Lazar 2012. Comparison of Optimization Methods for L1-regularized Logistic Regression. *Midwest Artificial Intelligence and Cognitive Science Conference 2012.*

Langford, J., Li, L., and Zhang, T. 2009. Sparse online learning via truncated gradient. *The Journal of Machine Learning Research, 10:* 777-801.

Lewis, D., Yang, Y., Rose, T., and Li, F. 2004. RCV1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research,* 5: 361-397.

Li, F. and Yang, Y. 2003. A loss function analysis for classification methods in text categorization. *In Machine Learning-International Workshop Then Conference.* 20: 472.

Liu, Y., Wang, R., and Zeng, Y. S. 2007. An Improvement of One-Against-One Method for Multi-class Support Vector Machine. In *Machine Learning and Cybernetics, 2007 International Conference* 5: 2915-2920.

Menon, A. K. 2009. Large-scale support vector machines: algorithms and theory. *Research Exam, University of California, San Diego*.

Montgomery, D. C., Peck, E. A., and Vining, G. G. 2012. Introduction to linear regression analysis. 821.

Pechyony, D., Shen, L., and Jones, R. Solving Large Scale Linear SVM with Distributed Block Minimization.

Pérez-Cruz, F., Figueiras, A., and Artes, A. 2004. Double chunking for solving SVMs for very large datasets.

Reif, M., Shafait, F., and Dengel, A. 2011. Prediction of classifier training time including parameter optimization. In *KI 2011: Advances in Artificial Intelligence,* 260-271.

Rifkin, R. and Klautau, A. 2004. In defense of one-vs-all classification. *The Journal of Machine Learning Research*, 5: 101-141.

Shalev-Shwartz, S., Singer, Y., and Srebro, N. 2007. Pegasos: Primal estimated sub-gradient solver for svm. In *Proceedings of the 24th international conference on Machine learning,* 807-814.

Stevens, James P. Applied multivariate statistics for the social sciences. Routledge Academic, 2012.

Tewari, A. and Bartlett, P. L. 2007. On the consistency of multiclass classification methods. *Journal of Machine Learning Research,* 8: 1007-1025.

Yu, H. F., Hsieh, C. J., Chang, K. W., and Lin, C. J. 2012. Large linear classification when data cannot fit in memory. *ACM Transactions on Knowledge Discovery from Data, TKDD 5*(4): 23.