A Review of Some Machine Learning Systems

1. Introduction

Li et al. [10] made a comparison of two machine learning frameworks in the context of cloud computing. This study is a partial replication of their experiment on a much wider variety of machine learning systems. Both Li et al. and this current work can only hope to scratch the surface of the possibilities in machine learning. One of the key findings of this current research is that machine learning systems can vary wildly in terms of performance, implementation, usability and accuracy of results when applying the same technique to the same problem.

In the Li et al. experiment two commonly used machine learning systems, Apache Mahout [4] and Dato Graphlab [7] were compared on three machine learning problems. Two of the three problems were topic modelling problems that involved large amounts of text processing. The third was a collaborative filtering application that was more numerically focused. In this study two of these tests were replicated: a topic modelling problem, an analysis of server log files with an aim to correlate log entry topic with system failure, and a collaborative filtering task of predicting Twitter follower relationships. These two tasks were attempted on six machine learning systems: BDAS/Spark [3], Apache Mahout [4], Graphlab [7], Vowpal Wabbit [9], MADLib [11] and Mallet [12]. Table 1 is a summary of the results.

Time series is another important area of application in machine learning which was not covered in the Li et al. experiment. Two time series analysis systems Yahoo EGADS [20] and Numenta NuPIC [15] were tested on raw network and cpu performance data. Table 2 is a summary of the results.

Table 1: summary of topic modelling and collaborative filtering results

Toolkit	Ease of installation and use	Topic modelling prediction ranks (accuracy/speed)	Twitter follower prediction ranks (accuracy/speed)
Vowpal Wabbit 7.10.2	Easy	1/4	3/4
Mallet 2.0.6	Moderate	2/3	not tested
Graphlab 1.4.1	Difficult	3/2	2/3
MADLib 1.6.0	Difficult	4/5	not tested
BDAS/Spark 1.3.1	Very difficult	5/1	1/1
Mahout 0.11	Very difficult	failed/6	4/2

Table 2 time series results summary

Toolkit	Ease of installation and use	Time Series rank (accuracy/speed)
EGADS 0.1	Moderately Easy	1/1 (for best algorithm)
NuPIC 0.2.1	Difficult	2/2

Different machine learning systems have different objectives and design criteria. These differences can have a significant impact on the usability of any given system for any given task. Both Apache Mahout and its offshoot project BDAS/Spark are primarily designed to work in a distributed computing environment. Many of the other systems tested could be integrated into a distributed environment but worked just as well stand alone. MADLib was unique in that it was integrated into databases.

Of all the systems Graphlab was the only one that required a commercial license (an academic license was graciously provided by Dato Corporation for the duration of the course). Some systems were more numerically specialized. Spark, Mahout, Vowpal Wabbit and MADLib all seemed to be more focussed in this direction. Mallet was specifically designed for text based tasks.

2. Methods

Three problems were attempted. The three problems were:

1. use exception message topics as the basis for predicting system failures
2. use a graph to predict whether a given twitter user was following another twitter user
3. use time series analysis to predict network and CPU usage on a web server

Problem 1: predicting system failures from inferred error message topics

Problem 1 required log files to process. A partial data set from the original Li et al. experiment was provided by the University of Virginia. This consisted of one large 50MB log file that contained 191055 error and warning messages from a high performance compute cluster.

This large log file was split up in various ways as different machine learning systems required data in different formats. Each message, along with the contents of the message had a timestamp and a log level. The levels were, in increasing order of severity, DEBUG, INFO, WARN and ERROR.

For the purposes of replicating Li et al's work, each log message was labelled based on whether it represented an increase in error severity. Li et al's criterion for this was: a message represented an escalation if it was at a higher level than the median for the last 35 log messages. These escalation messages were labelled "1" and the other messages were labelled "0". Of the 191055 messages, 5492 (2.87%) messages met the criterion for error escalation.

To compare the effectiveness of the LDA topic weights generated by each system as predictors, Vowpal Wabbit's Logistic Regression implementation was exclusively used to generate predictions. This eliminated an extra unknown in the assessment. Using Vowpal Wabbit did not substantially affect processing time. 90% of the log entries were used as training data with the positive escalation examples (1) amplified such that they represented 50% of the examples. The generated LR model was tested on the remaining 10% of the data.

Problem 2: inferring Twitter followers

This problem was far less difficult to implement and test. Once again the University of Virginia provided the data needed. This data consisted of a graph of pairs of numeric Twitter identifiers representing actual Twitter users. The pairs represented the relation of the first Twitter user following the second. Collaborative filtering was used to generate a model of "follower" relationships with an input dataset of 3,084,481 pairs. The model was then used to predict relationships between a held out data set of 342,721 pairs.

Problem 3: network utilization time series predictions

The input data for this problem was one week of CPU and network monitoring data for a moderately busy web server. The data consisted of an epoch second timestamp and either the 5 minute rolling average of CPU load at that time or the number of packets sent in the time from the previous timestamp. The time differences were on the order of 2 to 5 seconds.

EGADS was compared with Numenta's NuPIC by running the input files through scripts which output the predictions as CSV format data consisting of three fields: epoch timestamp, actual value, predicted value. The differences in actual vs predicted values were calculated using a custom script *evaluate.sh* (in */srv/cal/src/egads* and */srv/cal/src/nupic/zabbix*). This script timed each run and checked the resulting output, calculating the mean error for each input file and algorithm.

3. Measurement criteria

For the log error prediction task, predictions were made on the held out data with Logistic Regression which outputs probabilities of a given example belonging to a given class - in this case the probability of the example representing an error escalation. The predictions were compared with the actual class of each log message and the Root Mean Squared Error (RMSE), F score and Receiver Operating Characteristic Curve (ROC curve) Area Under Curve (AUC) were generated for the predicted versus actual values. The unix *perf* program was used to compare the actual classes and generate the F, RMSE and AUC statistics. These statistics are defined by the formulas below [24 pp 172-180].

$$RMSE = \sqrt{\frac{\sum(actual - predicted)^2}{n}} \quad (1)$$

where n is the total number of predictions used for testing

and

$$F = 2 \times \frac{precision \times recall}{precision + recall} \quad (2)$$

where

$$recall = \frac{true\ positives}{true\ positives + false\ negatives} \quad (3)$$

$$precision = \frac{true\ positives}{true\ positives + false\ positives} \quad (4)$$

The ROC curve is a graph of precision or True Positives (TP) on the y axis against False Positives (FP) on the x axis [24 p 176].

$$TP = \frac{true\ positives}{true\ positives + false\ negatives} = precision \quad (5)$$

and

$$FP = \frac{false\ positives}{false\ positives + true\ negatives} \quad (6)$$

For the Twitter follower predictions RMSE was calculated on the test set. The test set was the same test set used by Li et al. Statistics were generated by each individual machine learning framework rather than a centralized test script.

For the time series data the actual and predicted values were compared using the RMSE and Mean Absolute Error (MAE) on the normalized difference between predicted and actual value. The MAE formula below is from Witten et al. [24 p 180].

$$MAE = \frac{\sum |actual - predicted|}{n} \quad (7)$$

The time to produce the LDA topic mixtures, Collaborative Filters and time series predictions was also recorded for each machine learning system. The timing measurements for the log analysis and Twitter follower tasks are somewhat misleading as they do not take into account the time spent on rework getting data into usable formats for the target systems. With the time series tools EGADS did not require any preparation but NuPIC required a 40 minute parameter optimization step before testing could begin.

4. Results

Problem 1: predicting system failures from inferred error message topics

The results of the error escalation predictions are summarized in Table 3. Table 4 shows ROC curves for the best and worst classifiers.

Table 3: summary of log escalation events based on LDA topic models

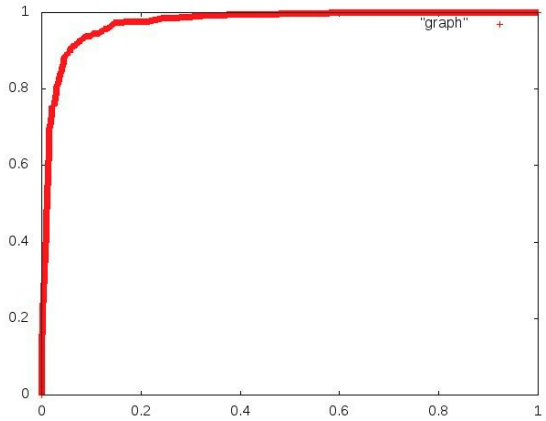
Toolkit	LDA Time (Prep/Run Min)	AUC	F	RMSE
Vowpal Wabbit	<2 / ~6	0.97233	0.54579	0.18589
Graphlab	<2 / ~1	0.91857	0.38564	0.28691
Mallet	<2 / ~2	0.91128	0.29617	0.29245
MADLib	~35 / ~8	0.83164	0.16673	0.40540
Spark (8 nodes)	~30 / ~1	0.58684	0.11094	0.17143
Spark (Single)	Failed	n/a	n/a	n/a
Mahout (8 nodes)	~30 / ~45 Lost data	n/a	n/a	n/a

The RMSE score for Spark was unexpected. On closer examination almost none of the Spark results were classed as being escalation events or “1” (79 predicted vs 584 actual). This contrasts with Vowpal Wabbit which predicted 1104 of the log entries as escalation events (compared with 524 actual).

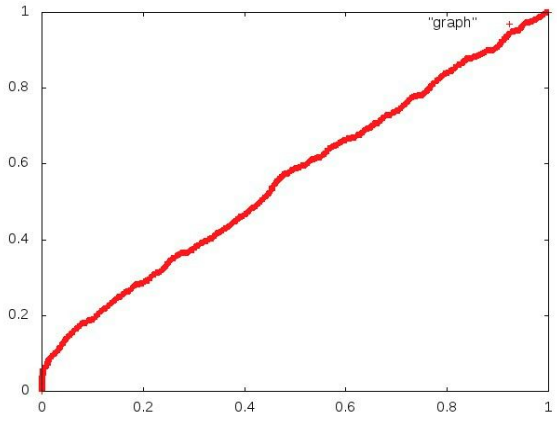
The Spark predictions exhibit a relatively small range compared to the other tools which may account for this discrepancy. The RMSE calculation is based on the squared difference of the probability, a number between 0 and 1, and the predicted class, either a 0 or a 1. Whereas a true positive is a frequency count not affected by scale.

Table 4: ROC curves for the topic modelling and prediction task

Best Vowpal Wabbit: AUC 0.97233



Worst Spark: AUC 0.58684



The listed times are the approximate processing times for preparing data and running it through the LDA algorithm to generate topic weights for each error message. The MADLib figure includes loading all the data into two Postgresql tables - one for documents and one for a dictionary. In a production database most likely the data might already be in the database. However, LDA requires generating statistics on the frequency of each word for all documents and for each individual document. The dictionary building step in MADLib turned out to be extremely time consuming (32 minutes) if all words were used in the dictionary. In comparison, commonly used text editing tools such as perl [25] can do the same job in a minute - as Figure 1 illustrates.

Figure 1: perl "one liner" dictionary builder and obfuscated code contest candidate

```
time perl -ne 'chomp;
($ts,$m)=($1,$2) if /\[(.*?)(WARN|ERROR|DEBUG|INFO)\]/;
if (/\[$ts$m\]/) {
   foreach (@msg) { next unless /\w/; s/\W+/_/g; $c{$_}++; $totals{$_}++; }
   print "$m | ",(join " ", map { "$_:$c{$_}" } sort { $c{$b} <=> $c{$a} } keys %c),"\n";
   %c=@msg=();
} else {
   push @msg, split /\s+/;
}
END { use Data::Dumper; print STDERR Dumper(\%totals) }' log.txt > docs.txt 2>dict.txt

real    1m2.760s
user    0m59.509s
sys     0m2.413s
```

For the topic modelling task all toolkits were tested with all words included to generate comparable results. In a production setting one would most likely remove common words before generating LDA topic weights. Filtering appeared to significantly affect overall running time in practice. As such these timing results should be taken as a "worst case" scenario.

Problem 2: inferring Twitter follower relationships

Generating the Collaborative Filtering models, for the toolkits that supported CF, was a much simpler process than handling the text data. However, MADLib and Mallet did not appear to have collaborative filtering implemented. Test results are summarized in Table 5.

In contrast, Graphlab had two different CF optimization schemes Alternating Least Squares (ALS) and Stochastic Gradient Descent (SGD). Both were tested. Note the drastic improvement in processing time with SGD. All other tools used ALS.

Spark dramatically improved over its performance in the LDA task and was the clear winner here with the lowest error and speediest processing time. Spark and Mahout used an 8 node

Hadoop cluster to generate these results. Spark's time is all the more impressive as it used the slower ALS optimization method.

Table 5: Collaborative filtering accuracy and speed (ordered by RMSE)

Toolkit	Spark	Graphlab ALS	Vowpal Wabbit	Graphlab SGD	Mahout
RMSE	0.04329	0.09434	0.09693	0.10724	0.11233
Time	0 m 50.636s	3 m 59.484s	4 m 29.904s	0 m 54.269s	2 m 22.028s

The timing data in this case only reflects running time of the optimization algorithms as there was no data manipulation involved.

Problem 3: network utilization time series predictions

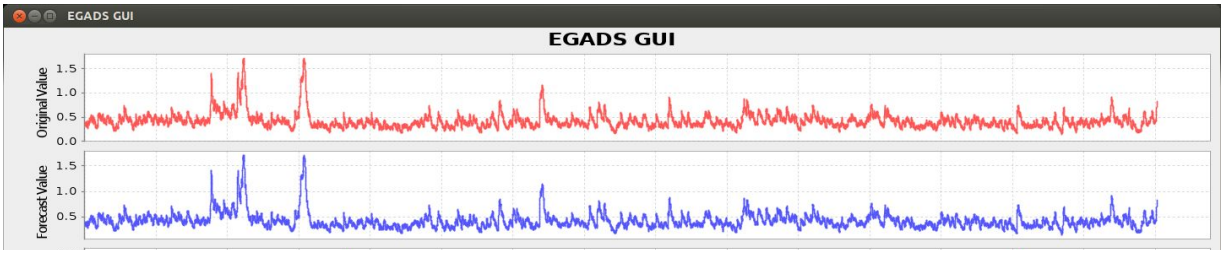
The time series predictions are summarized in Table 6. The listed results are for time series prediction of a quasi-periodic but somewhat chaotic network utilization data stream.

Table 6: EGADS + NuPIC incoming network traffic prediction running time and error (ordered by RMSE)

Algorithm	Running Time	RMSE	MAE
Double Exponential Smoothing Model	1 m 07.429s	0.0033893127	0.0004114775
Moving Average Model	0 m 20.258s	0.0118632667	0.0014783201
Weighted Moving Average Model	0 m 20.185s	0.0119737604	0.0014819254
NuPIC	2 m 21.865s	0.0125233871	0.0010758358
Triple Exponential Smoothing Model	1 m 55.244s	0.0129765293	0.0013357614
Simple Exponential Smoothing Model	0 m 20.073s	0.013142195	0.0015542917
Naive Forecasting Model	0 m 15.597s	0.0147225279	0.0017005605
Olympic Model	0 m 01.577s	0.0235406447	0.0010514938
Auto Forecast Model	3 m 05.014s	0.0235406447	0.0010514938
Polynomial Regression Model	0 m 01.787s	0.0235683265	0.0024430346
Regression Model	0 m 01.663s	0.023597017	0.00241754
Multiple Linear Regression Model	0 m 01.860s	0.023597017	0.00241754

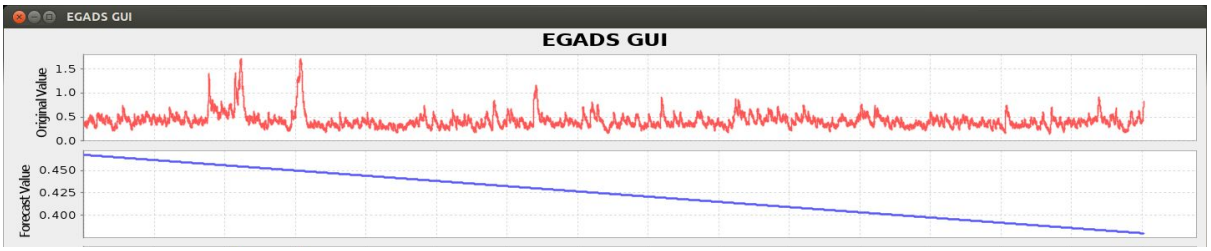
Figures 2,3 and 4 show examples of the predictions made for a CPU load average file.

Figure 2: Double Exponential Smoothing CPU timer series predictions (predictions are second row in blue)



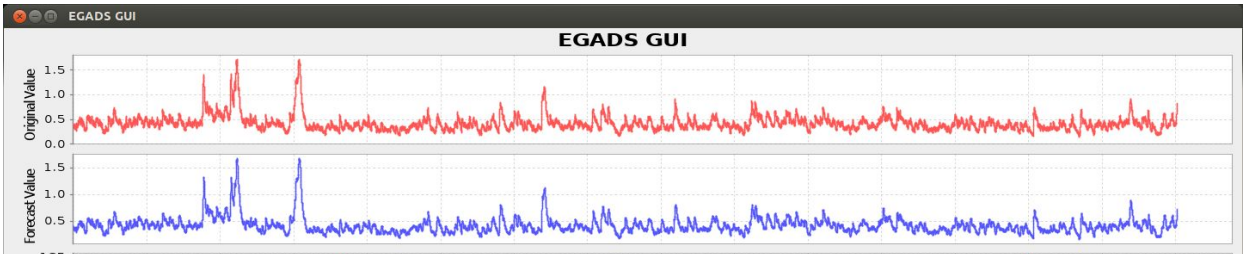
With EGADS models tended to swing wildly from being very close to the input to taking a “one size fits all” approach. As Figure 3 illustrates, the Multiple Linear Regression Model simply “predicted” a straight line - what one might expect from linear regression.

Figure 3: Multiple Linear Regression CPU time series predictions (predictions are second row in blue)



NuPIC was clearly in the middle of the pack with the network data. However, as can be seen in Figure 4, NuPIC was much better at predicting the smoother CPU load data. There is little observable difference between NuPIC and the best EGADS algorithm in this case.

Figure 4: NuPIC CPU time series prediction (predictions are second row in blue)



5. Discussion

While there are many tools for machine learning, these tools vary considerably in usability, quality and applicability for a given purpose. One tool that may be a poor choice for one task is the best for another. This variability is a reflection of the variety of machine learning techniques. As Domingos [8] put it: “There are literally thousands.” Adding to the complexity is the fact that these toolkits are evolving over time. Tools that may not work well now may be much more effective in future versions.

Mahout is a good example. In this review it was a disappointment but it has gone through at least one major refactoring in its history [4] and may go through more again. A limitation of the current review is that only older versions of Mahout, Hadoop and HDFS were were available for testing. Results may have been different with newer software.

A side effect of evolving software is documentation that is incomplete, confusing or just plain wrong. This turned out to be an issue with all platforms to some degree or another. This is perhaps to be expected in an arena where many practitioners are researchers or enthusiasts. Writing the documentation is easy to put off in projects under active development.

Builds too were often broken and in one case an error in the software had to be corrected. Table 7 summarizes build mitigations that were needed to proceed with testing.

Table 7: builds were often broken and needed to be modified to work

Toolkit	Build issues
BDAS/Spark	won't build with -Pyarn because of library version discrepancy
Graphlab	required licensed free edition
Vowpal Wabbit	one string bug which caused failures when input lines blank
NuPIC	cnp library failing - tool still works with it commented out
EGADS	have to comment out -Xlint in top level pom.xml to build
Weka	missing postgresql library in jar file causing failures

Furthermore, the lack of a standard input format for training and test data, combined with incomplete or missing documentation, was a major cause of delay. With the output data, mapreduce based tools such as Mahout and Spark left the output data out of order which made doing the order sensitive class assignments in the logging task much more difficult. While Spark's data could be restored to order, Mahout's could not. However, this issue was moot as during mapreduce Mahout would consistently lose two of the log messages. The resulting output was unusable.

In the log file classification task, once all the formatting bugs were worked out and testing could begin there was another hidden problem: one class clearly dominated the other. This difference was so extreme (33x) that it made the Logistic Regression fail on many platforms. It was a common occurrence for all predictions to be 0 - a motivation to use a separate tool to make the LR predictions. The Logistic Regression prediction generating script, *vw-lr.sh* [based on 17] in */srv/cal/src/virginia-datasets*, took care of creating a random sample (using a 90%-10 %split), amplifying the less numerous examples, randomizing the data running the training data trhough Vowpal Wabbit's LR implementation to create a model and using the model to test the held out examples. The *vw-lr.sh* script used the *perf* program was used to calculate accuracy scores and graph the results.

The tests, especially of the topic modelling task, were not intended to determine optimal models. Instead the focus was on comparing performance. Using all the words in the log file inputs is not typically done in practice . However, this ensured that all the toolkits had the same input. Similarly, the number of topics used in LDA is a parameter that is provided to the algorithm. Typically the number of topics tuned by running the LDA algorithm on larger and larger numbers of topics until the measured perplexity of the resulting topic mixtures for the documents in the collection is at a minimum. As LDA is an unsupervised algorithm there is no direct way to know the optimal number of topics from the algorithm.

Most of the toolkits that implemented LDA used a technique called Gibbs Sampling to assign topics to words (see the Glossary below for an explanation of how Gibbs Sampling works). However, Spark used a different technique, Expectation Maximization (EM). This may have skewed results. Perhaps EM is more affected by the large amount of redundant words in the unfiltered input data.

Scaling of the LDA algorithm from one to eight servers did not improve the processing time with Mahout. As Spark failed in standalone mode it was not possible to confirm if this was the case in general. Spark's LDA implementation, executed In a distributed context, did outperform the other frameworks but not by an eight-fold margin as might be expected. For a relatively small dataset such as the log entry dataset, the overhead of mapreduce may overwhelm any improvement in processing time from parallelization.

Between platforms, there was considerable variability in the per log entry topic mixtures produced. Each document had 20 numbers associated with it. Each number proportional to the likelihood of that topic for that document. Some toolkits such as MADLib output the topic weights as integers and others output them as fractions. Ranges varied considerably. Table 8 illustrates the average per document range of values for the topic mixtures generated by each toolkit.

Table 8: scale of LDA generated topic weights used as LR inputs

System	Av Per Row Max-Min (Normalized)	Av Per Row Max-Min (Unnormalized)
MADLib	0.0190	62.49
Spark	0.3775	0.3776
Vowpal Wabbit	0.0272	66.4334
Mallet	0.5694	0.5694
Graphlab	0.3986	0.2915
Mahout	0.7602	0.7602

How similar were the topic weights? Table 9 shows the average estimated correlation between topics for the different toolkits using two different methods to estimate the correlations. The numbers in the parentheses represent the average of the maximal correlations between topics for pairs of toolkits. This should represent the upper bound of how related the LDA topic weights for the two toolkits might be. The second is an estimate based on a greedy algorithm where

topics are paired based on the maximum correlation between topics which have not already been selected. While this is not the optimal value for the likely topic correlation it takes into account conflict situations where a single topic has two or more approximately equal correlates. Note how Vowpal Wabbit correlates relatively poorly with the others but nevertheless performed the best in the tests.

Table 9: estimated between-platform LDA topic correlations

LDA topic correlation matrix - numbers in parentheses represent average of the upper bound r					
	Graphlab	MADLib	Mallet	Spark	Vowpal Wabbit
Graphlab	1	0.4464 (0.5716)	0.4168 (0.5378)	0.5247 (0.6091)	0.2179 (0.3118)
MADLib		1	0.4742 (0.6648)	0.6290 (0.7659)	0.2343 (0.4475)
Mallet			1	0.5261 (0.6247)	0.2674 (0.3593)
Spark				1	0.3164 (0.4734)
Vowpal Wabbit					1

How valid are the topics generated by LDA? Some research has questioned the assumption that optimal topic mixtures are truly distinguishable by human observers. For example, Chang et al. [28] showed subjects lists of topic words identified using LDA topic modelling. These words were combined with extra words not related to the topics. The study participants had a great deal of difficulty identifying the added words. However, in another study 90% agreement was found between expert human judges and topic modelling software [29]. There may be some types of texts that are easier to model than others. Limiting the types of words used by passing the text through a part of speech tagger and keeping only nouns may also affect the quality of results from LDA. However, with some texts, such as the log messages used in this study, making those distinctions may be more difficult.

In comparison to the topic modelling task, the Collaborative Filtering task was relatively straightforward. Both Mahout and Spark were able to run the task flawlessly and quickly. Spark was clearly fastest and the most accurate of all the tools tested. This was a good illustration how application specific these tools can be in practice.

The time series tests focused on an entirely different aspect of machine learning - unsupervised pattern detection. EGADS was far simpler to use than NuPIC. However, unlike NuPIC, EGADS implements numerous time series and anomaly detection algorithms. Some testing would be needed to determine which is better for a specific application.

NuPIC attempts to mimic the actual working of the cerebral cortex [15]. It was much more complicated to operate, requiring a parameter building step which typically took at least 40 minutes to complete. The parameters generated were specific the input data being tested. Even

with parameter tuning NuPIC was disappointing both in performance and accuracy. Contrary to claims made by the NuPIC authors [], the predictions actually got worse when the same model was applied to the same data several times.

6. Conclusion

This study compared machine learning tools with a variety of design philosophies and target applications on three distinct tasks. There was no clear best choice for all tasks on all dimensions. For less common applications, such as LDA, the usability, performance and accuracy of each tool varied substantially. For more common tasks such as CF and LR the tools tended to behave more consistently. This study focused more on what distinguished each toolkit than on what they had in common.

Overall, Vowpal Wabbit was the easiest to work with for optimization problems. EGADS was the easiest to work with for time series data. The most difficult platforms to use were those that were designed for distributed environments. This is unfortunate as many real world machine learning applications involve large scale distributed data sets.

Li et al's results were not entirely replicated. In particular, the Mahout test failed and no usable data was generated from it. Perhaps, given different input data or a newer version of Mahout the results would have been better.

That there was no standard input or output formats for any of the platforms reviewed suggests that people who use one tool tend to only use that tool. However, designing machine learning tools for inter-operation would be a welcome development. Algorithms should be independent of data input. Models should have a common, human readable, format. And results should be evaluated using common evaluation tools such as the *perf* program used in this study.

As the results of the same algorithm on the same input data could be wildly inconsistent, inter-tool validation is essential to improve accuracy. Also, a little competition could motivate developers to improve the platforms.

Finally, these tools need to work seamlessly at varying scales. Developing a solution on an individual system should not require substantial changes to work on a multi-host distributed network. From the results of this study scaling is not a problem that has been gracefully solved yet in machine learning. How these pressing issues will be resolved as the industry evolves remains to be seen.

7. Glossary

Numerical modelling:

Collaborative Filtering

A matrix factorization algorithm where an *m* by *n* matrix M of relationships between *m* entities (such as movie viewers) and *n* related entities (such as movies) are factorized into two matrices, one *n* by *f* say A and the other *f* by *m* say B such that AB approximately equals M. The rows in A are called “factor vectors” of the *n* entities in M (the movies) and the columns in B are the same for the *m* related entities (the viewers). []

Alternating Least Squares

An optimization algorithm used for collaborative filtering where matrix A in the Collaborative Filtering description above is used to estimate matrix B then matrix B is used to estimate A iteratively until error terms do not further decrease. The error in the estimates is the square of the difference between the actual rating and the estimated rating (either the product of the factor vector for the viewer and the movies matrix A or the factor vector for the movie times the viewers matrix B). In this way each matrix A and B can be checked against the ground truth before being updated. [19]

Stochastic Gradient Descent

An optimization algorithm where an objective function (an error function) is nudged to a central minimum. At this minimum the slope of the derivative of the objective function is 0. Individual weights for individual features are updated based on a learning rate and the difference between the partial derivative of the objective function with respect to that given feature and 0. In Stochastic Gradient Descent each example is used to optimize only one feature in the linear equation greatly speeding up the convergence process on large datasets. [24 p 236, 242]

Logistic Regression

A logistic function is a linear function that is transformed into a sigmoid function (“S” shaped) that can model the probability of a class: as the output of the linear function increases the sigmoid function gets closer to 1. As the output of the linear function becomes more negative the output of the sigmoid function becomes closer to 0 [24 p 236]. The linear function combines a feature vector with weights relating to the value of that feature in predicting the class. Gradient Descent is often used to optimize the weights for the feature vector. Once optimized the logistic function can be used to predict the probability of a given set of features representing an example of a specific class. This type of model can only distinguish between two classes. [24 pp 125-127]

Latent Dirichlet Allocation

Is an unsupervised learning method for assigning classes to objects. For example, when a document is processed using Latent Dirichlet Allocation what is allocated are “topics” of words in that document. Counting these “topics” for a given document gives a probability mixture representing the relationship between that document and the topics. The allocation is done using some form of Markov Chain Monte Carlo sampling such as Gibbs Sampling. Words that feature prominently in specific topics are inferred from the number of times they are assigned a given topic. “Latent” refers to the fact that the nature of the topics is not known ahead of time. Words, like documents, also have probability mixtures indicating which topics they belong to and a single word can be representative to a greater or lesser degree to multiple topics. [6]

Gibbs Sampling

A form of Markov Chain Monte Carlo sampling where the current class of an item is determined from the conditional probability of the item being in that class given the class assignments of the other items in the collection. For words in Latent Dirichlet Allocation this would mean the topics assigned to the other words. Each word has a single topic assigned to it in Gibbs Sampling. Initially all the words are assigned random topics and as the Gibbs Sampling algorithm runs, topics are randomly reassigned based on the relative likelihood of a word being associated with any of a given set of topics. The likelihood of any given topic being determined by how many words are assigned that topic in general and in that particular document. In practice how this is done is to calculate the relative proportions of each topic (e.g. 30% topic 1, 25% topic 2, 45% topic 3), pick a random fraction from 0-1 and pick the topic based on where the random number falls (e.g. if the random number was 0.56 then topic 3 would be chosen). This reassignment process in turn changes the conditional probabilities of the topic assignments. Gibbs sampling is a generalization of a Bernoulli process to more than two possible outcomes. It is also often used to model responses on multi-item questionnaires in social sciences. [23]

Expectation Maximization

In the context of Latent Dirichlet Allocation, Expectation Maximization, like Alternating Least Squares above, updates topic probability mixtures for individual terms based on topic mixtures for documents containing those terms and documents are updated in turn based on the topic mixtures for terms. Expectation Maximization treats each topic as a cluster and each document (or word) is assigned a probability or *weight* for being part of that cluster. These weights are determined via an optimization process where the mean and standard deviation of each topic probability distribution is calculated. The mean and standard deviation are used to calculate the likelihood of an individual document or word is related to a given topic. If we assume the probability distributions for topics are normal, the normal function can be used to make the likelihood estimate. Multiplying these likelihoods together (or summing -1 times the logs of the probabilities) can be used to estimate if one set of topic-document probability distributions is a better match for the data than another. Each topic is updated with new probabilities for each word and each

document in turn and the likelihood estimate is calculated. When the likelihood estimate stops changing significantly (as determined by a predefined cutoff) then the algorithm stops. In practice convergence happens quickly. [24 p 288]

Perplexity

A measure of how likely a given model predicts the data seen. Typically perplexity is measured in bits. One way to calculate it is as the sum of the measured probability of the evidence times -1 times the log base 2 of the estimated probability of the evidence. Lower perplexity indicates a more accurate model. [22]

Time series models:

Many of the EGADS tools use code from the openforecast project [16]. References, unless otherwise noted are the openforecast source or the EGADS source [20] (also found in */srv/cal/src/egads/src/main/java/com/yahoo/egads/models/tsmm/* on cloudbig).

Auto Forecast Model

An amalgam of all the following models - run each and pick the best result.

Double Exponential Smoothing Model

Uses two smoothing factors alpha and gamma. Alpha is used to calculate what is called the *level* $L_t = \alpha Y_t + (1-\alpha)[L_{t-1} + T_{t-1}]$ and gamma is used to calculate the *trend* $T_t = \gamma[L_t - L_{t-1}] + (1-\gamma)T_{t-1}$. These are combined to produce an estimate at time t: $Y_t = L_{t-1} + T_{t-1}$ [13][14]

Moving Average Model

Sums the values over a given period with each value being given a constant weight that depends on the size of the period. [26]

Multiple Linear Regression Model

Create a “best fit hyperplane” of the values in the dataset using Gaussian Elimination to generate coefficients for an equation of the form “Y = mx + b” with each data point tuple being a row in the matrix. For the time series data the “x” is the position of the measurement in the time series.

Naive Forecasting Model

A moving average model with the number of periods set to 1.

Olympic Model

Look back a preset number of “weeks” and average the values, dropping outliers.

Polynomial Regression Model

Similar to Multiple Linear Regression, uses Gaussian Elimination to find coefficients for a modelling equation. In this case each row of the matrix is the independent variable to the power of the column with the last column being the original variable. EGADS uses a polynomial of order 3 (i.e. y = x + x^2 + x^3)

Regression Model

Makes a regression equation using the standard least squares method. [27]

Simple Exponential Smoothing Model

Similar to the Moving Average model except that the weights are nonlinear. Using a constant, alpha, future values are related to past values by the formula $Y_{t+1} = \alpha Y_t + (1-\alpha) Y_{t-1}$. EGADS uses an alpha of 0.75 for forecasting.

Triple Exponential Smoothing Model

Adds a seasonal index to Double Exponential Smoothing with a third factor. The seasonal index looks back a "year" into the data $SI_t = \beta \frac{Y_{actual\ at\ t}}{Y_{predicted\ at\ t}} + (1-\beta) SI_{t-1}$

Weighted Moving Average Model

Similar to the Moving Average Model except that multiple weights are explicitly stated, one for each time period in the past for the average. EGADS uses 0.75, 0.25 as their weights.

<u>NuPIC Cortical Learning Algorithms:</u>

Reference [15].

Swarming Algorithm

Not strictly a part of NuPIC's Cortical Learning Algorithms, the Swarming Algorithm is a simplified genetic algorithm where a series vectors of parameters of interest are formed and evaluated against some criterion. In NuPIC's case these parameters are used to test predictive ability. At each iteration all vectors are evaluated and the best vector is selected. The other vectors' parameters are modified by applying a "velocity" which is designed to nudge the vectors towards the current best candidate. In this way multiple sets of parameters can be evaluated at once. This method is useful if the evaluations are somewhat unpredictable - in contrast to Gradient Descent which assumes a single optimum value to its evaluation function.

Spatial Pooling Algorithm

Is a method for representing external inputs. It is implemented as a 3 dimensional array where the dimensions are determined by the Swarming Algorithm. The columns in the array represent distinct sets of neurons that act together somewhat like "bits" in that they represent a single 1 or 0 in the array. The spatial pooler ensures that similar external

stimuli are represented in a similar way. However, it also ensures that only a small percentage of columns need to be active to correctly represent the stimulus.

Temporal Pooling Algorithm

Builds on the Spatial Pooling Algorithm to identify patterns in time. It is the key to NuPIC making predictions of future events. External stimuli that have been seen together become associated with each other. Elements in the columns of cells record other cells that are active when a stimulus occurs. When the same cells are activated they in turn re-activate cells they have coincidentally seen active when they were active. These reactivations are the basis of predictions. If enough cells reactivate a cell it is assumed to be on for the next iteration of the algorithm. If not enough cells reactivate a cell it is turned off. Because only some columns need to be active to represent an input, NuPIC can make an educated guess as to what future values of an input are likely to be with incomplete or ambiguous information.

8. References

[1] Akyildiz, B. Alternating Least Squares Method for Collaborative Filtering. http://bugra.github.io/work/notes/2014-04-19/alternating-least-squares-method-for-collaborative-filtering/. Accessed August 7, 2015.

[2] Amplab UC Berkeley. Berkeley Data Analytics Stack. https://amplab.cs.berkeley.edu/software/. Accessed June, 2015.

[3] Apache Foundation. Spark MLib. http://spark.apache.org/mllib. Accessed June, 2015.

[4] Apache Foundation. What is apache mahout? http://mahout.apache.org. Accessed June, 2015.

[5] Bradley, J. (March 25, 2015). Topic modeling with LDA: MLlib meets GraphX. https://databricks.com/blog/2015/03/25/topic-modeling-with-lda-mllib-meets-graphx.html. Accessed August 8, 2015.

[6] Chen, E. (2011). What is Latent Dirichlet Allocation? http://blog.echen.me/2011/08/22/introduction-to-latent-dirichlet-allocation/. Accessed August 9, 2015.

[7] Dato Incorporated. Graphlab create. https://dato.com/products/create/. Accessed June, 2015.

[8] Domingos, P. (2012). A few useful things to know about machine learning. Communications of the ACM 55(10). October 2012.

[9] Langford, T. Vowpal wabbit. https://github.com/JohnLangford/vowpal_wabbit/wiki. Accessed June, 2015.

[10] Li K., Gibson, C., Ho, D., Zhou, Q. Kim, J., Buhusi, O., Brown, D. E., Gerber, M. (2013). Assessment of machine learning algorithms in cloud computing frameworks. IEEE Systems and Information Engineering Design Symposium 2013.

[11] Madlib.net. MADlib: Big data machine learning in SQL for data scientists. http://madlib.net/. Accessed June, 2015.

[12] McCallum, A. K. (2002). MALLET: A machine learning for language toolkit. http://mallet.cs.umass.edu. Accessed June, 2015.

[13] Minitab Inc. (2014). What are the level and trend components for double exponential smoothing?

http://support.minitab.com/en-us/minitab/17/topic-library/modeling-statistics/time-series/time-series-models/what-are-level-and-trend-components/. Accessed August 7, 2015.

[14] NIST. Forecasting with double exponential smoothing (LASP). http://www.itl.nist.gov/div898/handbook/pmc/section4/pmc434.htm]. Accessed August 7, 2015.

[15] Numenta.org. Numenta platform for intelligent computing. http://numenta.org/. Accessed June, 2015.

[16] Openforecast. Openforecast CVS source repository. http://openforecast.cvs.sourceforge.net/viewvc/openforecast/src/net/sourceforge/openforecast/models/. Accessed August 8, 2015.

[17] Popel, M. Stackoverflow.com (July 9, 2014). How to perform logistic regression using Vowpal Wabbit. http://stackoverflow.com/questions/24634602/how-to-perform-logistic-regressionusing-vowpal-wabbit. Accessed June 15, 2015.

[18] University of Waikato. Weka 3: Data mining software in java. http://www.cs.waikato.ac.nz/ml/weka/. Accessed June, 2015.

[19] Apache Mahout. Introduction to ALS Recommendations with Hadoop. https://mahout.apache.org/users/recommender/intro-als-hadoop.html. Accessed June, 2015.

[20] Nikolay Laptev, Saeed Amizadeh, Youssef Billawala. (May 14, 2015). Announcing the Open Source of EGADS: A Scalable, Configurable, and Novel Anomaly Detection System. http://labs.yahoo.com/news/announcing-the-open-source-of-egads-a-scalable-configurable-and-novel-anomaly-detection-system/ and https://github.com/yahoo/egads. Accessed June, 2015.

[21] Weinberger, P. (2015). Using NuPIC. https://github.com/numenta/nupic/wiki/Using-NuPIC

[23] Wikipedia.org. Perplexity. https://en.wikipedia.org/wiki/Perplexity#Perplexity_per_word. Accessed August 8, 2015.

[23] Wikipedia.org. Gibbs sampling. https://en.wikipedia.org/wiki/Gibbs_sampling. Accessed August 7, 2015.

[24] Witten, I., Frank, E., Hall, M. (2011). Data mining: Practical machine learning tools and techniques, 3rd Edition. Burlington, MA: Morgan Kaufman.

[25] Perl.com. The perl programming language. https://www.perl.org/. Accessed August 9, 2015.

[26] Wikipedia.org. Moving average model. https://en.wikipedia.org/wiki/Moving-average_model. Accessed August 8, 2015.

[27] Wolfram.com. Least squares fitting. http://mathworld.wolfram.com/LeastSquaresFitting.html. Accessed August 9, 2015.

[28] Chang, J., Boyd-Graber, J., Gerrish, S., Wang, C. and D. M. Blei (2009). Reading tea leaves: How humans interpret topic models. http://machinelearning.wustl.edu/mlpapers/paper_files/NIPS2009_0125.pdf. Accessed August 9, 2015.

[29] Chanen, A., Patrick, J. (2007). Measuring correlation between linguists' judgements and latent dirichlet allocation topics. http://www.aclweb.org/anthology/U07-1005. Accessed August 9, 2015.