On the Trade-Off Between Multi-Level Security Classification Accuracy and Training Time

Paal Engelstad
Oslo and Akershus University College of Applied Sciences (HiOA)
Oslo, Norway
e-mail: paal.engelstad@hioa.no

Abstract—Automatic security classification is a new research area about to emerge. It utilizes machine learning to assist humans in their manual classification. In this paper, we investigate the importance of the training time of the machine learner. To the best of our knowledge, this has not been analyzed in previous works. We compare various machine learning methods, including SVM, LASSO and the ensemble methods Adaboosting and Adabagging, with respect to their performance. The paper demonstrates that the computational cost of a method is an important part of its performance metric.

Keywords—Multi-level security, classification, machine learning, ensemble methods, feature selection, cross-domain information exchange.

I. INTRODUCTION

Security classification is about classifying information objects, such as documents and text messages, into different groups, e.g. “Secret”, “Confidential”, “Unclassified” etc. The concept is not only used by the military, government agencies and international organizations, but also by private corporations, e.g. see [1]. The classification indicates the sensitivity of the contents of different information objects and mandates how the information object shall be treated according to the governing security policy.

The content of an information object is typically classified by human inspection and assessment, and given a security label (e.g. "Public" vs "Confidential" or "Business Internal"). However, with an increasing amount of generated information, there is a need for tools that can assist with automatic security classification. As a result, research that explores the use of machine learning for automatic security classification of information objects is about to emerge.

A number of recent works have explored machine learning to this problem. Some works, such as [2], [3], [4] and [5], are in favor of using Support Vector Machines (SVM) or Least Absolute Shrinkage and Selection Operator (Lasso) [6] [7] over methods like k-Nearest Neighbor (kNN) and Naïve Bayes (NB) [8]. Other works, such as [9], emphasize that some ensemble methods, such as Adaboosting and Adabagging ([10] [11]), might perform comparably well or even better, despite their comparably larger computational overhead. Other ensemble methods, such as RandomForest, on the other hand, might not scale well to this problem [9].

In this paper, we take the training time into account when comparing the performance of the slowly trained ensemble methods, Adaboosting, and Adabagging, with the performance of the quickly trained methods, SVM and Lasso. To the best of our knowledge, such comparison has not yet been studied for this problem. This constitutes the main contribution of this paper.

II. MAIN EXPERIMENT

A. Experimental setup

The analyses presented in this paper are based on a benchmark experiment for binary classification described in detail in [3], [4] and [5]. Due to space limitations, the reader is referred to these references for an extensive description. The setup will be only briefly summarized in the following.

In our experiments we used a corpus of documents retrieved from the Digital National Security Archive - DNSA [12]. We used around 2800 documents in our analyses, where around half of the documents are "Classified", while the other part is "Unclassified". The document texts are treated as bag-of-words, where we apply word stemming to each word. Each document gives a term-vector with a dimensionality corresponding to the number of available words, and with each vector component providing the frequency of the word used, i.e. it is typically a sparse vector. Vectors of all documents comprise the Document-Term Matrix (DTM), where each row in the matrix is a word-frequency vector corresponding to one specific document. Some specific keyword terms were ignored. We ignore infrequently used words (used less than 15
times), reducing the numbers of columns in the DTM from around 23000 word stems, down to around 5840 term stems. Finally, we used "term frequency-inverse document frequency" (tfidt) for term weighting [3] [4].

For the analyses presented in this paper, we use two traditional non-ensemble methods, SVM and Lasso, as well as a the ensemble methods Adaboosting and Adabagging. Furthermore, we use a 10-fold cross-validation to decrease the variance in our results. This represents a difference from the analysis presented in [3], [4] and [5], where a 10-fold cross-validation on the training set was used only for tuning machine learner parameters.

III. RESULTS

A. Classification accuracy

Table I shows the main results derived according to the experimental setup described above. We observe in the left column (labeled "Acc") that AdaBoosting yields the highest classification accuracy, but that there are small differences in the results between the chosen algorithms.

B. ROC, AUC and False Positives

Previous works have noted that automatic security classification will not be used without human control, intervention and assistance [3] [4]. The reason is that automatic security classification is different from the vast number of machine learning scenarios in the way that misclassifications can have extremely damaging consequences. If an information object that contains classified information is mistakenly predicted to be unclassified, it might lead to information leakage. This can be especially critical in military settings.

A consequence of this is that only looking at the Classification Accuracy alone might not be sufficient. A further refinement is to study the importance of False Negatives (FN) and False Positives (FP) through the Receiver Operating Characteristic (ROC).

Usually, the more up to the left the ROC curve is, the better, and therefore the Area Under the Curve (AUC) is a commonly used measure for how "good" a ROC curve is. The AUCs that we obtained for the different machine learners are given in Table I. It shows that the AUC performance of Lasso is comparably the highest relative to its classification accuracy. The AUC performances of the remaining machine learners are pretty much consistent with what their accuracies show.

Since we are especially concerned with the False Positives (i.e. the "False Unclassified" according to our definition here), one side of the ROC curves is more important than the other. Instead of looking at the skewness of the curves, we display the share of False Positives (FPs) in Table I. This metric is very important to many of the scenarios that automatic security classification is targeting. We observe that AdaBagging perform the best.

Nevertheless, due to space limitations we will mainly focus on the trade-off between the classification accuracy and the training time in the following. The approach we use below can easily be extended to studying a trade-off that involves for instance false positives instead of the classification accuracy. This is left for future work.

IV. TRAINING TIME

A. Computational overhead

To the best of our knowledge, previous works on Automatic Security Classification have not analyzed the computational time of various machine learners. The number of seconds to complete the training part of each machine learner is shown in Table I. Since the computational time can be scaled by the size of investment in hardware, we are mostly interested in the relative differences in performance. (The actual numbers shown in the table were derived on a modern MacBook Pro 2015 laptop computer.)

The results show that LASSO is extremely quick compared to the other alternatives. SVM is comparably slow, but much quicker than the remaining ensemble methods.

B. Why is training time essential?

In many of the scenarios that are targeted by Automatic Security Classification, there will be an ongoing production of new information objects that need to be labeled with its classification level. A machine learner is using the existing information objects as a training set, in order to predict the classification level of the new objects.

In a previous work [5], it has been shown that there can be an evolution of the topics that are described in the documents, as an “aging” of the information. The most recent of the existing documents are the ones in the training set that are the best predictors of the classifications of new objects, while the oldest documents in the training set might describe outdated topics or have outdated perspectives. It is important to get newly produced information objects into the training set, and train the machine learner regularly so that the machine learner always is trained with the most pertinent information. The computation

<table>
<thead>
<tr>
<th>Machine learner</th>
<th>Randomized in time</th>
<th>Acc</th>
<th>AUC</th>
<th>FPs</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>AdaBoosting</td>
<td>.808</td>
<td>.878</td>
<td>.082</td>
<td>13293</td>
<td></td>
</tr>
<tr>
<td>LASSO</td>
<td>.783</td>
<td>.869</td>
<td>.070</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>SVM</td>
<td>.792</td>
<td>.862</td>
<td>.064</td>
<td>913</td>
<td></td>
</tr>
<tr>
<td>AdaBagging</td>
<td>.777</td>
<td>.848</td>
<td>.052</td>
<td>11247</td>
<td></td>
</tr>
</tbody>
</table>
TABLE II
CLASSIFICATION ACCURACY WITH CHRONOLOGICALLY ORDERED DOCUMENTS.

<table>
<thead>
<tr>
<th>Machine learner</th>
<th>Chronological</th>
<th>Accuracy</th>
<th>95% conf.int.</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM</td>
<td>No</td>
<td>0.77</td>
<td>(0.74, 0.81)</td>
</tr>
<tr>
<td>SVM</td>
<td>Yes</td>
<td>0.70</td>
<td>(0.67, 0.73)</td>
</tr>
<tr>
<td>Lasso</td>
<td>No</td>
<td>0.78</td>
<td>(0.75, 0.82)</td>
</tr>
<tr>
<td>Lasso</td>
<td>Yes</td>
<td>0.74</td>
<td>(0.71, 0.77)</td>
</tr>
</tbody>
</table>

The time of the machine learner determines how quickly it is possible to update the machine learner with newly acquired information.

In such scenarios, all the documents in the training set are older than the documents that shall obtain a predicted security classification. To provide a pertinent analysis of the performance, one must create a similar experiment; i.e., all the documents in the training set must be older than all the documents in the test set. Nevertheless, many analyses, including the analysis we have presented above, uses a cross-validation technique where different folds are determined by randomly selecting documents to the training set and test set. This means that the information objects are not chronologically ordered in time in terms of the machine learning. This yields artificially good results, because the machine learner is not suffering from the aging of information between the training set and the test set.

We conducted an additional analysis where a 10-fold cross-validation is conducted on a training set (70% of the corpus) to tune the parameters, before the performance of the machine learner is determined on the test set (Table II). To demonstrate our point, it is sufficient to limit ourselves to Lasso and SVM, which both are quick machine learners that are easily tuned. Indeed, results in Table II show a considerable drop in performance with chronological ordering, i.e., when we ensure that all documents in the training set are older than all documents in the test set. (In fact, it is possible to reduce this performance drop by introduction of time interaction techniques [5], but this issue is out of scope here.)

This performance drop underlines the significance of the information aging, and how important it is to frequently update the machine learner. The consequences of not updating the machine learner is illustrated in Fig. 1. We observe that the classification accuracy of the first documents to be classified is quite good. However, as more and more documents are to be classified, the training set gets more and more outdated, and the classification accuracy is decreasing. It is most easily seen by the red line, which shows how the moving average is decreasing. However, after a certain time, the red line stabilizes. This indicates that the training set consists of both a time-independent basis and a component that is highly time relevant. To always be operating in the left side of the figure where the classification accuracy is at its highest, one needs to update the machine learner sufficiently often.

In scenarios where the information aging is relatively high, the quickest machine learners in Table I, such as LASSO, might perform better than the slowest ensemble methods even though the latter displays good accuracy, since their training gets more outdated than the training of the quickest methods.

It is possible to obtain these performance gains by artificially speeding up the machine learning by reducing the size of the DTM. A feature selection (FS) method can be used to reduce the number of columns of the DTM by ignoring the least significant word stems. However, too aggressive features selection might punish the performance of the machine learner. In summary, there is a trade-off between the performance gain of the FS (through higher speed and thus having an up-to-date test set) and the performance loss of the FS (through lower overall precision of the information extracted from the training set). This trade-off will be analyzed in further depth in the following.

C. Different Feature Selection (FS) methods

Information Gain (IG), Chi-square analysis (CHI) and Document Frequency (DF) represent some of the different approaches for determining which features to select and
which to ignore [13]. We will base our analysis on IG in the following, since previous works on automatic security classification have shown that IG is a good candidate for FS [2] [4].

D. Performance losses of FS

Fig. 2 shows how the classification accuracy deteriorates as the feature selection gets more and more aggressive (i.e., as we move more and more to the left on the curves in the figure). Note that the x-axis is logarithmic; the figure shows that one might reduce the size of the DTM quite significantly (down to around 3% of its original size) without losing too much performance. As we reach the black dashed vertical line (which is situated at $x = 10^{-1.5} = 0.0316$) the classification accuracy begins to take a toll. We also observe that the different machine learners are punished comparably equally.

In the following we will focus primarily on the two methods AdaBoosting and Lasso. AdaBoosting is chosen, because it is the method in Table I with the highest classification accuracy, while Lasso is the method in the table with the quickest learning time. Fig. 3 shows the classification accuracy for AdaBagging and Lasso. It is the same as curves as in Fig. 2, however, here it is shown on a linear x-axis. We can easily do our analysis below on the real curves, but it is quite convenient to work on fitted curves instead, since they are easy to illustrate. Therefore, we fitted the growth curves using logistic growth functions. The fitted curves are shown in dashed lines in the figure. Without loss of generality, the conclusions of our trade-off analysis below will be the same disregarding if we are using fitted curves or sufficiently statistically smoothed curves.

E. Performance gains of FS

Fig. 2 shows that the calculation speeds of all the displayed machine learners are nearly directly proportional to the size of the DTM (i.e., here we use a linear scale on the x-axis).

This shows it is possible to scale the training time on demand by means of applying feature selection. Furthermore, the training time is reduced proportionally with the level of feature selection that is being applied. This makes it possible to describe the classification accuracy as a function of
the training time. Since we know the difference in training time between different machine learners, it is possible to compare their performance, as well as their internal trade-off between classification accuracy and training time.

Fig. 4 shows the classification accuracy for AdaBoosting and Lasso, i.e. the same curve fittings as displayed in Fig. 3. However, in Fig. 4 the classification accuracy is now shown as a function of the training time. This is possible, since the training time is given directly by the level of feature selection.

In the figure, the training time is shown in terms of time units (which can be arbitrarily chosen, e.g. as seconds, minutes or hours, depending on the actual hardware used, the real size of the problem etc.) Both AdaBoosting (blue line) and Lasso (red line) are first growing quickly, and then the classification accuracy is stabilizing as the level of feature selection is reduced. However, since the training of Lasso is so much quicker than that of AdaBoosting, the growth part of Lasso is hardly shown in the figure - it is barely seen as a vertical line on the left side in the figure - and the performance of Lasso appears mostly as only a horizontal line.

Fig. 4 also shows fitting of information aging curves (similar to the curve fitting shown in Fig. 1). For illustration, two different scenarios are displayed: The solid black curve shows the quick information aging, while the dashed black curve shows an information aging that is comparably slower.

The solid red curve in Fig. 6 also shows the performance of Lasso as a result of both the underlying performance of Lasso from previous curve fittings (dashed red curve) and from the reduction in performance due to the information aging (dashed black curve), similar to the performance reduction demonstrated in Fig. 1). We observe that when the learning is sufficiently slow, the quickness of Lasso in terms of training time is not sufficient to outweigh the higher classification accuracy of AdaBoosting. Indeed, by selecting an appropriate level of feature selection according to the green vertical line (at the max value of x=884) in Fig. 6, AdaBoosting is still outperforming Lasso in for any possible outcome on the solid red line.
In a scenario with quicker information aging, the result might be different. Here the blue curve in Fig. 7 shows the performance of AdaBoosting as a result of both the underlying performance of AdaBoosting from previous curve fittings (dashed blue curve) and from the reduction in performance due to the information aging (dashed black curve). We observe that AdaBoosting is punished for having a large computational overhead. The excessive training time of AdaBoosting means that AdaBoosting has to work on a comparably old and out-aged corpus compared to Lasso. When the information aging is quick, the quickness of Lasso in terms of training time makes Lasso perform better than AdaBoosting, by selecting Lasso to operate quite much to the left on the solid red curve. Then, Lasso outperform AdaBoosting for any for any possible outcome on the solid blue line.

This example shows how it is possible to directly compare machine learner algorithm with respect to both the training time and the classification accuracy. It also shows the direct trade-off between training time and the classification accuracy. It is the level of information aging in the corpus that decides the optimal trade-off between the two metrics (cf. the vertical green line in Fig 6).

V. TUNING OF THE ENSEMBLE SIZES

In this section we show that for the ensemble methods, AdaBoosting and AdaBagging, we may speed up the learning time by reducing the ensemble size. However, a lower ensemble size might hurt the classification accuracy, similar to the use of FS.

Due to the time consumption of tuning a full-scale DTM with respect to the ensemble size, we conduct a tuning after having carried out a feature selection down to a share of \( x = 10^{-1.5} = 0.0316 \) (vertical black dashed line in Fig. 2). In Fig. 8 we see that the performance suffers when selecting an ensemble size of less than 20 (while we chose the default ipred value in R of 25). After this point, the performance gain of increasing the ensemble size decreases more and more.

As for the computational overhead, on the contrary, the speeds of all the ensemble methods studied are - as expected - linearly dependent on the size of the ensemble (Fig. 9). The non-ensemble methods, LASSO and SVM, on the other hand, are not dependent on an ensemble. For comparison, their performance is shown as horizontal lines in Figs. 8 and 9.

This means that for the ensemble methods we can use the ensemble size in a similar analysis as we did for feature selection, i.e. we can use it to convert the method onto a time scale and use it to trade off between training time and performance. If the information aging is quick, it makes sense to reduce the ensemble sizes to optimize performance. This corresponds to the optimum found for the feature selection (green vertical line in in Fig. 6). Due to space limitations, extending the analysis above to also cover adjustment of the ensemble size is left for future work.

VI. CONCLUSIONS AND FUTURE WORK

Different performance metrics of machine learning, such as the classification accuracy and the training time, is often considered incomparable. However, through the concept of information aging, we demonstrate how the two are
closely connected in scenarios where the information aging of the corpus implies that the training has to be carried out frequently.

Furthermore, this paper demonstrates that in such scenarios it is possible to compare different machine learners with respect to both the classification accuracy and the training time. In our scenario AdaBoosting performed best when the information aging was relatively slow, while with a high level of information aging the quickness of Lasso outweighed its slightly inferior classification accuracy in the base scenario, i.e. when the training of both AdaBoosting and Lasso are not speeding up by feature selection. Moreover, it is possible to analyse the trade-off between the two, e.g. by using feature selection and the classification accuracy as a function of the features selection to form a common time reference for all the comparisons. Indeed, it was shown how the exact level of feature selection can be chosen to derive with the best possible performance. (For instance, for AdaBoosting this optimum was shown by the green vertical line in Fig. 6).

Due to the space limitations of this paper, the issue has only been touched upon briefly, focusing primarily on the classification accuracy while ignoring the AUC and share of false positives. Extending the analysis to a trade-off between the training time and the AUC or the share of False Positives is straightforward. Furthermore, due to space limitations the analysis was primarily limited to the use of the curve fittings, while analysis with the statistically smoothed curves were not presented. Complementing the presented analysis with the statistically smoothed curves is also left for inclusion in a future work.

The paper finally shows that a similar analysis can be conducted by replacing the feature selection with reduction of the ensemble size for the ensemble methods. Extending our analysis to also cover adjustment of ensemble sizes is left for future work.

ACKNOWLEDGEMENT

This work was supported by the University Graduate Center (UNIK).

REFERENCES