Reducing the Effect of Out-Voting Problem in Ensemble Based Incremental Support Vector Machines

Zeki Erdem[1,4], Robi Polikar[2], Fikret Gurgen[3], and Nejat Yumusak[4]

[1] TUBITAK Marmara Research Center, Information Technologies Institute,
41470 Gebze - Kocaeli, Turkey
zeki.erdem@bte.mam.gov.tr
[2] Rowan University, Electrical and Computer Engineering Department,
210 Mullica Hill Rd., Glassboro, NJ 08028, USA
polikar@rowan.edu
[3] Bogazici University, Computer Engineering Department,
Bebek, 80815 Istanbul, Turkey
gurgen@boun.edu.tr
[4] Sakarya University, Computer Engineering Department,
Esentepe, 54187 Sakarya, Turkey
nyumusak@sakarya.edu.tr

Abstract. Although Support Vector Machines (SVMs) have been successfully applied to solve a large number of classification and regression problems, they suffer from the *catastrophic forgetting* phenomenon. In our previous work, integrating the SVM classifiers into an ensemble framework using Learn++ (SVMLearn++) [1], we have shown that the SVM classifiers can in fact be equipped with the incremental learning capability. However, Learn++ suffers from an inherent *out-voting* problem: when asked to learn new classes, an unnecessarily large number of classifiers are generated to learn the new classes. In this paper, we propose a new ensemble based incremental learning approach using SVMs that is based on the incremental Learn++.MT algorithm. Experiments on the real-world and benchmark datasets show that the proposed approach can reduce the number of SVM classifiers generated, thus reduces the effect of *out-voting* problem. It also provides performance improvements over previous approach.

1 Introduction

As with any type of classifier, the performance and accuracy of SVM classifiers rely on the availability of a representative set of training dataset. In many practical applications, however, acquisition of such a representative dataset is expensive and time consuming. Consequently, it is not uncommon for the entire data to be obtained in installments, over a period of time. Such scenarios require a classifier to be trained and incrementally updated as new data become available, where the classifier needs to learn the novel information provided by the new data without forgetting the knowledge previously acquired from the data seen earlier. We note that a commonly used procedure for learning from additional data, training with the combined old and new data, is not only a suboptimal approach (as it causes catastrophic forgetting), but it

W. Duch et al. (Eds.): ICANN 2005, LNCS 3697, pp. 607 – 612, 2005.

may not even be feasible, if the previously used data are lost, corrupted, prohibitively large, or otherwise unavailable. Incremental learning is the solution to such scenarios, which can be defined as the process of extracting new information without losing prior knowledge from an additional dataset that later becomes available. Various definitions, interpretations, and new guidelines of incremental learning can be found in [2] and references within.

Since SVMs are stable classifiers that use the global learning technique, they are prone to *catastrophic forgetting* phenomenon (also called unlearning) [3] which can be defined as the inability of the system to learn new patterns without forgetting previously learned ones. To overcome some drawbacks, various methods have been proposed for incremental SVM learning in the literature [4, 5]. In this work, we consider the incremental SVM approach based on incremental learning paradigm referenced within [2] and propose an ensemble based incremental SVM construction to solve the catastrophic forgetting problem and out-voting problem by reducing the number of SVM classifiers generated in ensemble.

2 Ensemble of SVM Classifiers

Learn++ uses weighted majority voting, where each classifier receives a voting weight based on its training performance [2]. This works well in practice even for incremental learning problems. However, if the incremental learning problem involves introduction of new classes, then the voting scheme proves to be unfair towards the newly introduced class: since none of the previously generated classifiers can pick the new class, a relatively large number of new classifiers need to be generated that recognize the new class, so that their total weight can out-vote the first batch of classifiers on instances coming from this new class. This in turn populates the ensemble with an unnecessarily large number of classifiers. The Learn++.MT algorithm, explained below, is specifically proposed to address this issue of classifier proliferation [6]. For any given test instance, it compares the class predictions of each classifier and cross-references them with the classes on which they were trained. Essentially, if a subsequent ensemble overwhelmingly chooses a class it has seen before, then the voting weights of those classifiers that have not seen that class are proportionally reduced.

For each dataset ($\mathrm{D_k}$), the inputs to the algorithm are (i) a sequence of m training data instances x_i along with their correct labels y_i, (ii) a classification algorithm, and (iii) an integer T_k specifying the maximum number of classifiers to be generated using that database. If the algorithm is seeing its first database ($k=1$), a data distribution (D_t), from which training instances will be drawn, is initialized to be uniform, making the probability of any instance being selected equal. If $k>1$ then a distribution initialization sequence initializes the data distribution. The algorithm adds T_k classifiers to the ensemble starting at $t=eT_k+1$, where eT_k denotes the current number of classifiers in the ensemble. For each iteration t, the instance weights, w_t, from the previous iteration are first normalized to create a data distribution D_t. A classifier, h_t, is generated from a subset of $\mathrm{D_k}$ that is drawn from D_t. The error, ε_t, of h_t is then calculated; if $\varepsilon_t > ½$, the algorithm deems the current classifier, h_t, to be weak, discards it, and returns and redraws a training dataset, otherwise, calculates the normalized classifica-

tion error, $\beta_t = \varepsilon_t/(1-\varepsilon_t)$, since for $0 < \varepsilon_t < ½$, $0 < \beta_t < 1$. The class labels of the training instances used to generate this classifier are then stored. The *dynamic weight voting* (*DWV*) algorithm is called to obtain the composite classifier, H_t, of the ensemble. H_t represents the ensemble decision of the first t hypotheses generated thus far. The error of the composite classifier, E_t is then computed and normalized. The instance weights w_t are finally updated according to the performance of H_t such that the weights of instances correctly classified by H_t are reduced and those that are misclassified are effectively increased. This ensures that the ensemble focus on those regions of the feature space that are not yet learned, performing the incremental learning [6].

Given a set of training samples x_i, i=1,…,m, where $\boldsymbol{x}_i \in \boldsymbol{R}^n$ is input patterns, y_i, (i=1,..,m), is the class labels, the SVM classifier function is formulated in terms of kernels functions, such as radial basis function and polynomial:

$$h(x) = sign\left(\sum_{i=1}^{m} \alpha_i y_i K(x_i, x) - b\right). \tag{1}$$

where b is the bias and α_i are the coefficients that are maximized by Lagrangian [7,8]. The final composite SVM classifier is obtained using the *DWV* algorithm for Learn++.MT algorithm, as follows [6]:

$$H_{final}(x_i) = \arg\max_{c} \sum_{t:h_t(x_i)=c} W_t \tag{2}$$

Where c = 1,2,…C is classes, and $W_t = \log(1/\beta_t)$ is the SVMs classifier weights.

3 Simulation Results

Proposed incremental learning approach of SVM ensemble using Learn++.MT has been tested on several datasets. We use the SVMLearn++.MT notation for proposed approach for consistency. Due to space limitations, we present results on one benchmark dataset and one real-world application as explained following sections. We used the LIBSVM library [9] as SVM solver. The Gaussian kernel functions were used in our experiments. We utilized the cross-validation technique with 5-folds to jointly select the SVM parameters, which are the regularization constant C and the RBF width σ.

3.1 Optical Character Recognition Dataset

The Optical Character Recognition dataset is a benchmark dataset from UCI machine learning repository. The OCR dataset features 10 classes (digits 0 ~ 9) with 64 attributes. The dataset was divided into four sets, to create three training subset (**DS1~3**) and a test subset (**Test**), whose distribution can be seen in Table 1. We evaluated the incremental learning capability and also the performance of SVMLearn++ and SVMLearn++.MT on a fixed number of classifiers to allow for a fair comparison. Each algorithm was used to generate seven classifiers with the addition of each dataset, giving a total of 21 classifiers in three training sessions. The data distribution was deliberately made rather challenging, specifically designed to test the ability of proposed approach to learn *multiple* new classes at once with each additional dataset while retaining the knowledge of previously learned classes. In this incremental learning problem, instances from only six of the ten classes are employed in each subsequent dataset resulting in a rather difficult problem.

Table 1. OCR data distribution

Class	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10
DS1	250	250	250	0	0	250	250	250	0	0
DS2	150	0	150	250	0	150	0	150	250	0
DS3	0	150	0	150	400	0	150	0	150	400
Test	110	114	111	114	113	111	111	113	110	112

Results from this test are shown in Tables 2 and 3. Each row shows class-by-class generalization performance of the ensemble on the test data after being trained with dataset $\mathbf{DS_k}$, k=1,2,3. The last two columns are the average overall generalization performance (**Gen.**) over 20 simulation trials (on the entire test data which includes instances from all ten classes), and the standard deviation (**Std.**) of the generalization performances.

Table 2. SVMLearn++ with RBF kernel ($\sigma = 0.1$, $C = 1$) results on OCR dataset

	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	Gen.	Std.
DS1	99%	100%	100%	-	-	98%	100%	99%	-	-	60%	0.04%
DS2	99%	73%	100%	44%	-	98%	68%	99%	47%	-	63%	1.54%
DS3	99%	100%	100%	93%	14%	97%	100%	99%	90%	13%	80%	4.17%

Table 3. SVMLearn++.MT with RBF kernel ($\sigma = 0.1$, $C = 1$) results on OCR dataset

	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	Gen.	Std.
DS1	99%	100%	100%	-	-	98%	100%	99%	-	-	59%	0.05%
DS2	99%	34%	99%	97%	-	93%	20%	99%	60%	-	59%	0.43%
DS3	99%	98%	95%	97%	89%	53%	100%	52%	95%	90%	85%	0.56%

SVMLearn++ was able to learn, the new classes 4 and 9, only poorly after they were introduced in **DS2** but able to learn them rather well, when further trained with these classes in **DS3**. Similarly, it performs rather poorly on classes 5 and 10 after they are first introduced in **DS3**, though it is reasonable to expect that it would do well on these classes with additional training. We note however, SVMLearn++.MT was able to learn new class quite well in first attempt. Finally, recall that the generalization performance of the algorithm is computed on the entire test data which included instances from all classes. This is the reason that the generalization performance is only around 59% after the first training session, since the algorithm has seen only six of the ten classes in the test data. Both SVMLearn++ and SVMLearn++.MT exhibit the ability of incremental learning and an overall increase of generalization performance as new datasets are observed. However, SVMLearn++.MT is able to learn better than SVMLearn++ as shown in Table 2 and 3.

3.2 Volatile Organic Compounds Dataset

The Volatile Organic Compounds (VOC) dataset is a real world dataset that consist of 5 classes (toluene, xylene, hectane, octane and ketone) with 6 attributes coming from

six (quartz crystal microbalance type) chemical gas sensors. The dataset was split into three training and a test dataset. The distribution of the data is given in Table 4, where a new class was introduced with each dataset.

Table 4. VOC data distribution

Class	C1	C2	C3	C4	C5
DS1	20	0	20	0	40
DS2	10	25	10	0	10
DS3	10	15	10	40	10
Test	24	24	24	40	52

In this experiment, both algorithms were incrementally trained with three subsequent training datasets. Each algorithm was employed to create as many classifiers as necessary to obtain their maximum performance. As shown in Tables 5 and 6, based on an average of 30 trials, SVMLearn++ generated a total of 33 classifiers to achieve its best performance; however SVMLearn++.MT not only produced a 5% better generalization performance with only 10 classifiers, but it also provided a significantly more stable improvement as seen from the reduced standard deviation.

Table 5. SVMLearn++ with RBF kernel ($\sigma = 3$, $C = 100$) results on VOC dataset

	C1	C2	C3	C4	C5	Gen.	Std.
DS1(5)	91%	-	95%	-	99%	58%	1.62%
DS2(10)	97%	91%	81%	-	95%	70%	1.84%
DS3(18)	93%	99%	94%	68%	76%	83%	8.19%

Table 6. SVMLearn++.MT with RBF kernel ($\sigma = 3$, $C = 100$) results on VOC dataset

	C1	C2	C3	C4	C5	Gen.	Std.
DS1(6)	93%	-	89%	-	99%	58%	1.67%
DS2(2)	96%	93%	88%	-	95%	70%	1.45%
DS3(2)	95%	94%	100%	99%	73%	88%	1.37%

4 Conclusions

In this paper, we presented a new ensemble based incremental SVM learning algorithm, SVMLearn++.MT, using Learn++.MT. SVMLearn++.MT with RBF kernel functions has been tested on one real world dataset and one benchmark dataset. The results show that while SVM classifier can be equipped with the incremental learning capability, dealing with *catastrophic forgetting* problem, SVMLearn++.MT reduces the effect of *out-voting* problem, and also provides performance improvements over SVMLearn++.

It is also worth noting that, SVMLearn++.MT is more robust than SVMLearn++. One of the reasons why SVMLearn++ is having difficulty in learning a new class

when first presented is due to difficulty in choosing the strength of the base classifiers. If we choose too weak classifiers, the algorithm is unable to learn. If we choose too strong classifiers, the training data are learned very well, resulting in very low β values which then causes very high voting weights, and hence even a more difficult *out-voting* problem. Since the SVM classifiers are strong classifiers, we have shown that SVMLearn++.MT, by significantly reducing the effect of the *out-voting* problem, improves the robustness of the algorithm, as the new algorithm is substantially more resistant to more drastic variations in the SVM classifier architecture and parameters (regularization constant C and kernel parameters).

Acknowledgements

This work is supported in part by the National Science Foundation under Grant No. ECS-0239090, "CAREER: An Ensemble of Classifiers Approach for Incremental Learning". Z.E. would like to thank Mr. Michael Muhlbaier and Mr. Apostolos Topalis graduate students at Rowan University, NJ, for their invaluable suggestions and assistance.

References

1. Z. Erdem, R. Polikar, F. Gurgen, N. Yumusak, "Ensemble of SVMs Classifier for Incremental Learning", Proc. of 6th Int. Workshop on Multiple Classifier Systems (MCS 2005), Springer-Verlag LNCS, Vol:3541, pp:246-256, Seaside, CA, USA, 13-15 June 2005.
2. R. Polikar, L. Udpa, S. Udpa, V. Honavar. "Learn++: An incremental learning algorithm for supervised neural networks." IEEE Transactions on Systems, Man, and Cybernetics. Part C: Applications and Reviews 31.4 (2001):497-508.
3. N. Kasabov, "Evolving Connectionist Systems: Methods and Applications in Bioinformatics, Brain Study and Intelligent Machines", Springer Verlag, 2002.
4. L. Ralaivola, F. d'Alché-Buc, "Incremental Support Vector Machine Learning: a Local Approach", In Proceedings of ICANN'01, Vienna, Austria, (2001)
5. C. P. Diehl and G. Cauwenberghs, "SVM Incremental Learning, Adaptation and Optimization", Proc. IEEE Int. Joint Conf. Neural Networks (IJCNN 2003), Portland OR, (2003).
6. M. Muhlbaier, A. Topalis, R. Polikar, Learn++.MT: A New Approach to Incremental Learning, 5th Int. Workshop on Multiple Classifier Systems (MCS 2004), Springer LINS vol. 3077 , pp. 52-61, Cagliari, Italy, June 2004.
7. V. Vapnik, Statistical Learning Theory. New York: Wiley, 1998.
8. N. Cristianini, J. Shawe-Taylor, An Introduction to Support Vector Machines and Other Kernel-based Learning Methods, Cambridge University Press, 2000.
9. C.-C. Chang, C.-J. Lin, "LIBSVM: A library for support vector machines", http://www.csie.ntu.edu.tw/~cjlin/libsvm