International Journal of Computational Intelligence Research.
ISSN 0973-1873 Vol.3, No.1 (2007), pp. 31–35

RVM Ensemble for Text Classification

Catarina Silva[1,2] **and Bernardete Ribeiro**[2]

[1]School of Technology and Management of the Polytechnic Institute of Leiria
Morro do Lena - Alto do Vieiro, Portugal, P-2411-901 Leiria, Portugal
catarina@dei.uc.pt

[2]Department of Informatics Engineering, Center for Informatics and Systems
University of Coimbra, Polo II, P-3030-290 Coimbra, Portugal
bribeiro@dei.uc.pt

Abstract: **Automated classification of texts by their likeness or affinity has greatly eased the management and processing of the massive volumes of information we face everyday. Although Support Vector Machines (SVM) provide a state-of-the-art technique to tackle this problem, Relevance Vector Machines (RVM), which rely on Bayesian inference learning, offer advantages such as their capacity to find sparser and probabilistic solutions. A known problem with the Bayesian approaches, however, is their relative inability to scale to larger problems where millions of documents are involved as well as real-time user's requests.**

We propose an ensemble strategy to circumvent RVMs scalability problem by applying a divide-and-conquer technique to handle the overload of available data, where the training documents are divided amongst small RVM classifiers, then the ensemble combines their individual contributions. The solution achieved keeps a sparse decision function and is computationally efficient. Results with respect to Reuters-21578 clearly demonstrate the proposed strategy can surpass other techniques, in both in terms classification performance and response time.

Keywords: Text classification, Relevance Vector Machines, Ensembles, Scaling Machine Learning Algorithms.

I. Introduction

Text classification (TC) aims to automatically assign semantic categories to natural language text. In recent years, automated classification of texts into predefined categories has attracted considerable interest, due to the increasing volume of documents in digital form and the following need to organize them.

A recurrent problem in TC problems is the scale of the problems that usually causes problems for many standard learning algorithms. Documents are represented by vectors of numeric values, with one value for each word that appears in any training document, making it a large scale problem. High dimensionality both increases processing time and the risk of overfitting, i.e., that the learning algorithm will induce a classifier that reflects accidental properties of the particular training examples rather than the systematic relationships between the words and the categories [1]. To deal with this dimensionality problem, feature selection and dimension reduction methods are applied, such as, stopword removal, stemming and removing less frequent words. Yang [2] presents a noteworthy scalability analysis of classifiers in TC, including KNN and SVM. In [1] an empirical analysis of sparse Bayesian classifiers for TC is conducted, concluding that RVM constitutes a competitive approach with minor changes to Tipping's initial proposal. Bayesian learning algorithms, like RVMs allow the user to specify a probability distribution over possible parameter values of the learned classifier. This not only provides one solution to the overfitting problem (since the algorithm can use prior distribution to regularize the classifier), but the prior also provides a mathematically well-justified way to allow domain knowledge to influence the parameter values that result from learning.

Despite RVM potential advantages when compared with SVM, a known problem with Bayesian approaches is their relative inability to scale with large problems like TC. In fact for a training set of N examples, RVM training time is $O(N^3)$ and memory scaling $O(N^2)$, making RVM difficult to scale when there are many training examples. We use a divide and conquer strategy, where the training documents are divided amongst small RVM classifiers and an ensemble strategy combines their individual contribution to substanctially improve classification performance.

The rest of the paper is organized as follows. Section II has a brief review of RVMs with their mathematical formulation. In section III preliminary results achieved with RVM are introduced, comparing them with state-of-the-art results to attain a baseline comparison platform. Section IV proposes the RVM ensemble and includes the results achieved. Conclusions and future lines of research are drawn in section V.

II. Relevance Vector Machines

RVM, pioneered by Tipping [3] is a sparse learning algorithm, similar to the SVM in many respects but capable of delivering a fully probabilistic output. It is reported to have nearly identical performance to, if not better than, that of SVM [3]. SVM has several desirable properties [4]:

- It fits functions in high-dimensional feature spaces, through the use of kernels;
- Despite a possibly large space of functions available in feature space, good generalization performance is nevertheless achieved by margin maximization [5];
- It is sparse: Only a subset of training examples is retained at runtime, improving computational efficiency.

However, there are also some disadvantages [3]:

- Although relatively sparse, SVM make unnecessary literal use of basis functions since the number of support vectors (SV) required typically grows linearly with the size of the training set. Some form of post-processing is often required to reduce computational complexity;
- Predictions are not probabilistic. In regression the SVM outputs a point estimate and in classification a 'hard' binary decision. Ideally we desire to estimate the conditional distribution in order to capture the uncertainty in out prediction.
- It is necessary to estimate the error/margin trade-off parameter C (and in regression the insensitivity parameter ϵ too). This generally entails a cross-validation procedure, which is wasteful both of data and computation.
- The kernel function must satisfy Mercer condition. That is, it must be the continuous symmetric kernel of a positive integer operator.

RVM's advantages rise due to its ability to yield a decision function that is much sparser than SVM, while maintaining its classification accuracy. This can lead to significant reduction in the computational complexity of the decision function, thereby making it more suitable for real-time applications [6].
The RVM was proposed by Tipping [3], as a Bayesian treatment of the sparse learning problem. The RVM preserves the generalization and sparsity of the SVM, yet it also yields a probabilistic output, as well as circumventing other limitations of SVM, such as the need for Mercer kernels and the definition of the error/margin trade-off parameter C.

A. Formulation

For an input vector $\mathbf{x}$, an RVM models the probability distribution of its label $d \in \{-1, +1\}$ using logistic regression:

$$p(d = 1|\mathbf{x}) = \frac{1}{1 + exp(-f_{RVM}(\mathbf{x}))}, \quad (1)$$

where $f_{RVM}(\mathbf{x})$, the classifier function, is given by

$$f_{RVM}(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i K(\mathbf{x}, \mathbf{x}_i), \quad (2)$$

where $K(.,.)$ is a kernel function, and $\mathbf{x}_i$, $i = 1, 2, ..., N$, are training samples. The parameters α_i, $i = 1, 2, ..., N$, in $f_{RVM}(\mathbf{x})$ are determined using Bayesian estimation, introducing a sparse prior on α_i. The parameters α_i are assumed to be statistically independent obeying a zero-mean Gaussian distribution with variance λ_i^{-1}, used to force them to be highly concentrated around zero, leading to very few nonzero terms. The α_i are then obtained by maximizing the posterior distribution of the class labels given the input vectors. This is equivalent to maximizing the objective function:

$$J(\alpha_1, \alpha_2, ..., \alpha_N) = \sum_{i=1}^{N} log\, p(d_i|\mathbf{x}_i) + \sum_{i=1}^{N} log\, p(\alpha_i|\lambda_i^*) \quad (3)$$

where the first summation term corresponds to the likelihood of the class labels, and the second term corresponds to the prior on parameters α_i, in which λ_i^* denotes the maximum a posteriori estimate of the hyper-parameter λ_i. In the resulting solution, only those samples associated with nonzero coefficients α_i, called relevant vectors (RV), will contribute to the decision function $f_{RVM}(\mathbf{x})$. In (2) the kernel function $K(.,.)$ forms expansion basis functions for $f_{RVM}(\mathbf{x})$, and is not limited by the Mercer's condition, unlike SVM. The Mercer's condition states that $K(.,.)$ must be a positive integral operator, that is, for every square-integrable function $g(.)$ defined on $\mathbb{R}^n$ the kernel $K(.,.)$ satisfies (4). As usual for text classification purposes, the linear kernel will be used.

$$\int\int K(\mathbf{x}, \mathbf{y}) g(\mathbf{x}) g(\mathbf{y}) d\mathbf{x} d\mathbf{y} \geq 0 \quad (4)$$

III. Preliminary Results

These results are presented as baseline comparison. Besides baseline RVM results, SVM results are also presented since SVM constitute the state-of-the-art machine learning technique in text classification. RVM models are as described in section II, while for SVM, SVM^{light} package by Joachims (http://svmlight.joachims.org/) was used. Before the results are presented, brief information about the data set and the performance criteria is given.

A. Data set

For the experiments, Reuters-21578 dataset (http://kdd.ics.uci.edu/databases/reuters21578/reuters21578.html) was used. It is a financial corpus with news articles averaging 200 words each. Reuters-21578 corpus has about 12000 classified stories into 118 possible categories. We use only 10 categories (earn, acq, money-fx, grain, crude, trade, interest, ship, wheat and corn), detailed in table 1 with the corresponding number of positive training and testing examples,

since they cover 75% of the items and constitute an accepted benchmark. The ModApte split was used, using 75% of the articles (9603 items) for train and 25% (3299 items) for test.

Table 1: Train and test documents for Reuters-21578.

Category	Train	Test	Category	Train	Test
earn	2715	1044	trade	346	113
acq	1547	680	interest	313	121
money-fx	496	161	ship	186	89
grain	395	138	wheat	194	66
crude	358	176	corn	164	52

B. Performance Criteria

When the number of positive examples is reduced, common error or accuracy measures are not appropriate, since they value equally both false positives (negative testing examples classified as positive) and false negatives (positive testing examples classified as negatives). It is possible to define different weights to these errors and obtain useful measures, but the usual performance criteria in TC used are Recall and Precision. Recall is the percentage of total documents for the given topic that are correctly classified (5).

$$Recall = \frac{relevant\, retrieved}{relevant\, in\, the\, collection} \tag{5}$$

Precision is the percentage of predicted documents for the given topic that are correctly classified (6).

$$Precision = \frac{relevant\, retrieved}{total\, number\, of\, retrieved} \tag{6}$$

An alternative representation is the use of true positives (TP), false positives (FP) and false negatives (FN) as in (7).

$$Recall = \frac{TP}{TP+FN}; \qquad Precision = \frac{TP}{TP+FP} \tag{7}$$

The use of two different measures can make it difficult to compare classifiers. Thus, usually a combined F_β (8) measure is calculated. In the experiments presented ahead F_β was used with $\beta = 1$ in the testing set for each category.

$$F_\beta = \frac{(\beta^2+1)\times TP}{(\beta^2+1)\times TP + FP + \beta^2 \times FN} \tag{8}$$

$F1$ can be represented alternatively, and possibly clearly (9):

$$F1 = \frac{2 \times precision \times recall}{precision + recall} \tag{9}$$

As the TC multi-class problem has been sub-divided in several two-class problems, averaging has to be used to find total criteria values. There are two types of averaging: micro-averaging and macro-averaging. In micro-averaging, performance tables for each of the categories are added, and the criteria are computed. In macro-averaging, performance measures are computed separately for each category and the mean of the resulting performance is taken. The results throughout the paper use macro-averaging.

ROC curves will also be drawn to offer a visual evaluation of classifiers, plotting TP rate versus FP rate (10), depicting relative trade-offs between benefits (TP) and costs (FP).

$$TPrate = \frac{TP}{TP+FN}; \qquad FPrate = \frac{FP}{TN+FP} \tag{10}$$

C. Baseline Results

Baseline results were achieved using words that appear in more than 2, 190 or 500 documents, i.e., which have a Document Frequency (DF) over 2 (DF>2), 190 (DF>190) or 500 (DF>500), resulting in respectively 7573, 497 or 176 words used in each setting. This initial feature selection was carried out to reduce the dimension of the problem. For each DF threshold, training was carried out with 1000, 2000 and all training examples. For RVM it was not possible to gather results for the largest set, i.e., with 7573 words, neither for all training examples. The number of examples is proportional to computational load, and that much processing power on one machine was not attainable and is not reasonable to be considered available in a real-time situation. With this constraint, the use of all words also constitutes a problem, that arises when there is one or more words that do not appear in any of the documents chosen for training. The results are represented in terms of F1 where classification performance is concerned and in terms of Support Vectors (SV)/Relevant Vectors (RV) and CPU training time (in seconds) where solution complexity is concerned. Table 2 presents RVM baseline results. Tables 3 and 4 resume the macro-averaged results for baseline settings concerning F1 and vectors respectively. From these results a few conclusions can already be drawn: (i) Classification performance is very similar for both learning machines for the same settings; (ii) RVMs present a much smaller computational complexity of the decision function, using between 13% and 14% of the vectors SVM needs; (iii) RVMs are more robust to smaller training sets and smaller number of features; (iv) RVMs do not scale well with number of documents, doubling the number of examples, increases time between 6 and 12 times, depending on the features used.

IV. Ensemble Approach

An ensemble is started by creating base classifiers with necessary accuracy and diversity. Unlike the traditional approach of choosing the best performing learning machine, an ensemble strategy compares the performance of the combined output with the selection and use of the best one, in terms of classification performance. When using the same learning algorithm, different classifiers are generated by manipulating the training set, manipulating the input features, manipulating the output targets or injecting randomness in the learning algorithm.

To make use of all training examples usually available in TC, our approach consists in the construction of several smaller training sets (chunks) of 1000 and 2000 documents.

Table 2: F1, Relevant Vectors and CPU training time (in seconds) for Baseline RVM using 1000 or 2000 documents and words with document frequency over 500 and 190.

RVM	1000 documents			2000 documents			RVM	1000 documents			2000 documents		
DF>500	F1	RV	CPU	F1	RV	CPU	DF>190	F1	RV	CPU	F1	RV	CPU
earn	94.65%	23	111	95.95%	36	828	earn	95.70%	23	95	97.52%	35	1218
acq	87.72%	25	93	89.80%	41	616	acq	86.69%	26	100	91.57%	49	1257
money-fx	46.43%	16	81	57.25%	34	626	money-fx	45.32%	19	96	57.14%	35	1193
grain	77.58%	19	126	80.61%	25	849	grain	72.86%	22	96	76.52%	28	1110
crude	68.17%	19	78	72.90%	27	606	crude	64.11%	16	90	69.51%	28	1125
trade	48.28%	20	104	43.96%	25	558	trade	42.46%	19	103	50.27%	37	1076
interest	59.65%	13	78	54.44%	22	584	interest	45.40%	16	84	58.29%	35	1133
ship	37.84%	15	98	62.28%	22	583	ship	37.84%	15	78	66.21%	26	1130
wheat	81.43%	12	83	79.14%	14	503	wheat	71.01%	19	90	75.18%	19	1133
corn	57.45%	14	112	62.14%	23	542	corn	62.63%	15	82	67.37%	27	952
average	65.92%	17.6	96.4	69.85%	26.9	629.5	average	62.40%	19.0	91.4	70.96%	31.9	1132.7

Table 3: Summary of F1 macro-averaged results for baseline settings.

$DF > 500$					$DF > 190$					$DF > 2$		
1000		2000		all	1000		2000		all	1000	2000	all
SVM	RVM	SVM	RVM	SVM	SVM	RVM	SVM	RVM	SVM	SVM	SVM	SVM
59.39%	65.92%	66.39%	69.85%	73.16%	57.79%	62.40%	66.89%	70.96%	76.14%	57.65%	68.76%	79.88%
1	1.11	1	1.05	1	1	1.09	1	1.06	1	1	1	1

Thus, two ensembles were tested one with 7 RVM classifiers trained with one chunk of 1000 documents each (from now on referred to as 7x1000) and another with 3 RVM classifiers trained with one chunk of 2000 documents each (from now on referred to as 3x2000). After the 7 or 3 RVM classifiers that make up the ensembles are trained, a majority voting scheme is implemented to determine the ensemble output decision, taking as output value the average value of the classifiers that corroborated the majority decision. As the number of chunk classifiers is odd, a draw is never reached.

A. Experimental results

Table 5 presents the F1 results for each RVM classifier for the 7x1000 ensemble and the final ensemble output performance. Table 6 displays F1 results, but for the 3x2000 classifier. On average both settings show a definite performance improvement of 9% for ensemble 7x1000 and of 5% for ensemble 3x2000. Peak improvements can go up to 42%, as for trade category on ensemble 7x1000. These gains are possible by the use of all training examples to determine the final classification. Baseline results decide based only on part of the available training documents. The ensemble approach makes it possible to use all documents, retaining all relevant information for classification. Figures 1 and 2 display ROC curves that compare for crude category the ensemble result with the best RVM classifier that constitutes the ensemble. Fig. 1 refers to the comparison between the best chunk of 1000 documents for crude category, i.e, chunk 3 with F1 of 67.10%, and the 7x1000 ensemble result for crude, i.e., with F1 of 70.99%. AUC values are in the southeast corner of each figure, confirming F1 results.

Table 5: F1 values for ensemble of 7 chunks of 1000 documents approach with Majority Voting (MV).

Category	**Maximum**	**Average**	**Ensemble Voting**
earn	96.77%	96.12%	97.69%
acq	90.36%	88.84%	94.97%
money-fx	68.29%	60.54%	71.81%
grain	84.83%	79.16%	81.62%
crude	75.08%	70.13%	78.34%
trade	66.67%	62.46%	70.25%
interest	67.02%	62.10%	70.47%
ship	66.67%	60.43%	77.99%
wheat	85.71%	80.83%	81.48%
corn	70.83%	64.06%	66.67%
average	77.22%	72.46%	79.13%

Table 6: F1 values for ensemble of 3 chunks of 2000 documents approach with Majority Voting (MV).

Category	**Chunk 1**	**Chunk 2**	**Chunk 3**	**MV**
earn	97.52%	96.58%	96.84%	97.36%
acq	91.57%	89.15%	89.01%	92.84%
money-fx	57.14%	61.76%	68.18%	71.86%
grain	76.52%	73.31%	74.70%	80.45%
crude	69.51%	58.70%	69.40%	75.45%
trade	50.27%	58.93%	63.67%	66.67%
interest	58.29%	57.30%	52.63%	67.37%
ship	66.21%	67.55%	76.54%	80.25%
wheat	75.18%	78.69%	70.87%	79.71%
corn	67.37%	62.37%	62.00%	63.64%
average	70.96%	70.43%	72.38%	77.46%

Table 4: Summary of number of vectors macro-averaged results for baseline settings.

$DF > 500$					$DF > 190$					$DF > 2$		
1000		2000		all	1000		2000		all	1000	2000	all
SVM	RVM	SVM	RVM	SVM	SVM	RVM	SVM	RVM	SVM	SVM	SVM	SVM
128.3	17.6	210.2	26.9	519.2	142.7	19.0	229.5	31.9	542.9	177.6	281.1	618.2
1	0.14	1	0.13	1	1	0.13	1	0.14	1	1	1	1

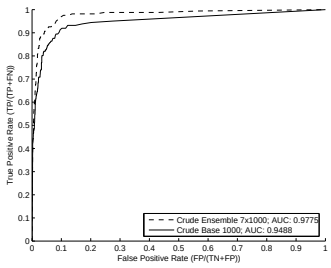

Figure. 1: ROC curves for crude category: comparison with ensemble of 7 chunks of 1000 documents.

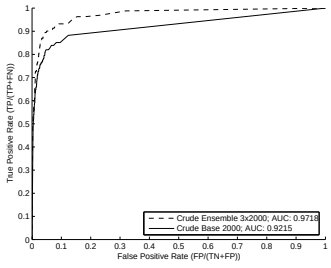

Figure. 2: ROC curves for crude category: comparison with ensemble of 3 chunks of 2000 documents.

V. Conclusions and Future Work

We have presented an RVM ensemble able to manage large datasets for Text Classification (TC). Two ensemble settings were defined using a divide-and-conquer technique, where training documents were first divided amongst small RVM classifiers, then an ensemble voting combined their individual contributions. Both settings show performance improvements compared with the best individual classifier, asserted not only by F1 performance measures, but also by ROC curves and the area under the curve (AUC). The approach produces useful sparse and probabilistic decision functions, coupled with a surplus in performance quality on Reuters-21578, gained by the use of all available training documents. Moreover the fast response time attained may lead to the distributed deployment of successful real-time applications.
Future research is expected in the refinement of the base classifiers. Instead of general-purpose classifiers, specific tasks will be assigned, such as, rare-class classification. This strategy will make the ensemble more heterogeneous where data is concerned, and also were functionality is concerned.

Acknowledgments

CISUC - Center of Informatics and Systems of University of Coimbra and Portuguese Foundation for Science and Technology Project POSI/SRI/41234 /2001 are gratefully acknowledged for partial financing support.

References

[1] S. Eyheramendy, A. Genkin, W. Ju, D. Lewis, D. Madigan, ”Sparse Bayesian Classifiers for Text Classification”, *Journal of Intelligence Community R&D*, 2003.

[2] Y. Yang, J. Zhang, B. Kisiel, ”A Scalability Analysis of Classifiers in TC”, *SIGIR*, ACM, 2003, pp 96-103.

[3] M. Tipping, ”Sparse Bayesian Learning and the Relevance Vector Machine”, *Journal of Machine Learning Research I*, 2001, pp 211-214.

[4] O. Williams, A. Blake, R. Cipolla, ”Sparse Bayesian Learning for Efficient Visual Tracking”, *IEEE Transactions on Pattern Analysis and Machine Intelligence.*, Vol. 27, No. 8, 2005, pp 1292-1304.

[5] B. Schölkopf, C. Burges, A. Smola, ”Adv. in Kernel Methods: Support Vector Machines”, MIT Press, 1998.

[6] L. Wei, Y. Yang, R. Nishikawa, M. Wernick and A. Edwards, ”RVM for Automatic Detection of Clustered Microcalcifications”, *IEEE Transactions on Medical Imaging*, Vol. 24, No. 10, 2005, pp 1278-1285.