Puncturing Multi-Class Support Vector Machines

Fernando Pérez-Cruz and Antonio Artés-Rodríguez*

University Carlos III, Department of Signal Theory and Communications
Avda. Universidad 30, 28911 Leganés (Madrid) SPAIN
fernandop@ieee.org

Abstract. Non-binary classification has been usually addressed by training several binary classification when using Support Vector Machines (SVMs), because its performance does not degrade compared to the multi-class SVM and it is simpler to train and implement. In this paper we show that the binary classifiers in which the multi-classification relies are not independent from each other and using a puncturing mechanism this dependence can be pruned, obtaining much better multi-classification schemes as shown by the carried out experiments.

1 Introduction

Support Vector Machines (SVMs) have rapidly become one of the most common techniques for solving binary classification problems, due to its ease of implementation and its state-of-the-art results in large number of applications [4]. SVMs have been also used for multiple class problems in which the lables might belong to more than 2 classes. For such problems there are several approaches and none of them seems to be superior to the others. Basically, we can either solve several binary problems [1] or we can directly solve the multiclass problem by extending the SVM binary formulation [6], which it is very hard to implement and train. Among the binary settings, we can describe several approaches: the comparison of each class against all the others [5], known as one-vs-all; the comparison of each class against all the other classes individually [3], known as all-pairs, or the comparison of a subset of classes against the rest of them using error correcting codes [2], this last one presents as particular cases the previous two.

In the paper in which it was unified the binary approaches for multi-class problems [1], they propose that the best way to increase the performance of the overall system (reduce the class' mislabelling) is to increase the number of binary classifiers as much as one can, because being more classifiers to choose from it will make the appearance of mistakes harder. The problem with this approach is that the classifiers can not be regarded as independent from each other, which is a needed condition for error correcting codes to be effective [7], so increasing the number of them might not increase the system performance and it could even degrade it. To obtain a higher performance (lower probability of erroneous

* This work has been partially supported by CICYT grant TIC2000-0380-C03-03.

decisions) we propose in this paper a puncturing mechanism that will eliminate those classifiers that degrade the performance and obtaining, as a consequence, much less complex multi-classification schemes.

The rest of the paper is outlined as follows. In Section 2, the error correcting codes are reviewed and its application to non-binary problem resolution is presented. In Section 3, we discuss the limitation of error correcting codes for multi-class problems and how we can use puncturing to improve the system performance. Computers experiment with the vowel data set are shown in Section 4. We end this paper in Section 5 with some concluding remarks.

2 Channel Coding for SVMs

Error protection codes are a family of techniques for detecting and correcting channel errors in digital communications. In the general setting, we have several symbols, let say k, to be transmitted through a binary communication channel. Previous to transmission, the symbols have to be matched with a binary string that can be handled by the communication system. The shortest string that can be used to distinguish between k symbols is $\lceil \log_2 k \rceil$, but with a single mistake in the transmitted bits, due to a channel error, it will produce a symbol error. One can assign longer sequences to each symbol expecting that, if done right, the string of bits would present more differences with other symbol's strings and would be able to detect or correct errors produce by the digital communication channel and, as a consequence, the mistake will not be transferred to the symbol itself. The price to pay is that as the length of the string increases the bit error rate does (for a fixed symbol throughput), because the time dedicated to each bit decreases [7]. For each digital communication system there is an optimum between the length of the string and the correcting capabilities which is usually sought when designing the code.

The learning problem from samples for multiple outputs can be described as finding decision rules that separate with the least number of errors a given set of labeled samples $((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_l, y_l)$ for $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \{1, 2, \ldots, k\})$. If we assign a binary string to each class and train a binary classifier with all the samples for each label of the string, the analogy with the channel communication readily arises. We will slightly modify the binary string to a ternary one, following [1], the entries to these strings will be either ± 1 or 0 indicating, respectively, the label of each class in the binary problem (± 1) or that it is absent from it (0). We show in Figure 1 three different ways in which a four-class problem can be divided into binary classifiers.

In order to assign a class to a new sample, we need to compute the output for all binary classifiers and construct an n-dimensional vector to be compared to each class' string. The class selection can be done using either Hamming distance if the outputs of the classifiers are binary (ternary) or using Euclidean distance if these outputs have not been threshold previously. For the one-vs-all approach, it simplifies to finding the classifier with largest output.

	one-vs-all				all-pairs						H(7,4)						
	C1	C2	C3	C4	C1	C2	C3	C4	C5	C6	C1	C2	C3	C4	C5	C6	C7
Class 1	1	-1	-1	-1	1	1	1	0	0	0	-1	-1	-1	1	-1	1	1
Class 2	-1	1	-1	-1	-1	0	0	1	1	0	-1	-1	1	-1	1	1	-1
Class 3	-1	-1	1	-1	0	-1	0	-1	0	1	-1	1	-1	1	1	-1	-1
Class 4	-1	-1	-1	1	0	0	-1	0	-1	-1	1	-1	1	1	-1	-1	-1

Fig. 1. We show three type of transformations of a four-class problem into binary classifiers. The first on is the on-vs-all approach in which each class takes the positive label once and the rest takes the negative label. The second approach compares all pairs of two classes in each binary classifier. One class takes the positive label the other class the negative label and the rest of the classes are absent from training. The last one is a channel coding approach in which we have used some of the words of weight three of a cyclic Hamming code H(7,4), in which a set of labels is compared to the rest of them.

3 Puncturing

The question that readily arises is how do I choose the strings to obtain the best performance. In [1], the authors propose that the larger the code is the better, because they use codes much longer than the minimum necessary one and the larger ones perform best. But seldom the code with largest number of classifier is the best. The selection of large codewords can be explained by the limiting trade off parameter. In channel coding this parameter is the bit energy (probability of channel errors) and in non-binary classification is the available computational resources, which is a far less limiting factor. So under this viewpoint, one would usually tend to use as many binary classifiers as he/she can handle. But it can be seen that as the number of classifiers increases the performance of the overall system does not always improves and it can even get worse, which does not follow what would be expected from coding theory. This fact can be discussed over the independence of the binary classifiers. Coding theory in order to assert that the symbol error rate decreases with increasing code length needs to assure that the bit error rate in the channel is independent bit by bit. But we cannot assert this point over a set of binary classifiers that have been trained with the same set of samples (with different labels). We cannot expect that with an increasing number of binary classifiers the overall probability of error will be reduced and could even expect an increase in it as some classifiers might present erroneous decisions for symbols in the borderline between its true class and an incorrect decision.

In most error correcting codes we have enough redundancy, so if one of the bits are missing, we can still decode without error the transmitted symbol. This applies as well for the error correcting codes for non-binary classification, for example, if we delete the 5^{th} classifier in the all-pair codes in Figure 1, the classes 2 and 4 are still distinguishable, although there is not a direct comparison between them. This technique is known in channel coding as puncturing and it is widely use for convolutional and block codes. The idea is that one can remove

one/several of the redundancy bits that the sequence is still decodable without error, without considering the non-transmitted bits. The puncturing, when used over a communication system, removes part of the redundancy, increasing the energy of the remaining bits[1]. The advantage of this technique is that it can be used to prune the pernicious classifiers, that do to provide an improve in each class determination.

The puncturing mechanism, that we propose to provide a better performance, relies in a iterative pruning of the worst classifier until the best set of binary classifier has been found. We start with the n classifiers and measure the performance of all the combinations of $n-1$ classifiers, keeping the set of classifiers with least number of errors in each iteration. This procedure is repeated until there is not any classifier left and we have n set of binary classfiers with a decreasing number of them to choose from. Then, the best set of classifiers is selected comparing the best performance of all the remaining subset of classifiers.

4 Computer Experiments

We have taken the vowel data set from the UCI Machine Learning Repository for testing the puncturing of SVM binary classifiers for multiple classes problem. We have selected this problem because the best result reported in UCI is a 44% of classification error using nearest neighbour classifier and this data set has been solved using channel codes in [1] and the best achieved result was a 39% using an all-pairs code and binary SVMs with a polynomial kernel of fourth degree. Also, this is one of the hardest multiple class problems because one has to distinguish between 11 different vowels pronounce by 15 people only 6 times. The training and test sets divides the people in 2 sets, respectively, with 8 and 7 in each one. The difficult part of this task is to extract the characteristics that separates the different vowels instead of separating the speakers. In order to train the classfier we used the 10 features and whether the speaker was a woman or a man, all the features where preprocess to present zero mean and unit standard deviation.

We have use this data set and have used as code: BCH(7,4), BCH(15,4), BCH(31,5), BCH(63,7), one-vs-all and all-pairs. The number of binary classifiers is k for the one-vs-all, $\binom{k}{2}$ for all-pairs and for the BCH codes is the first number in its descriptor. The BCH can be easily generated as shown in [7] and for selecting the string for each vowel we have selected randomly among all the possible codewords, deleting first the all zero (-1) codeword. We have used an RBF kernel, $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\right)$, and have use 8-fold cross validation (each time one of the training person was left a side) to select the hyperparamters σ and C. The hyper-parameters have been chosen in two different ways. First by using the same σ and C for all the and seeking for the

[1] This is necessary because the tools to construct good block codes do not allow to use an arbitrary length for them.

best global behaviour (Equal). Second, by treating each binary classifier independently and setting the best σ and C for each one of them and afterwards their outputs were joined together to form the output of the multi-class problem (Individual). We show in Table 1 the results for these two approaches, for both the outputs of the binary SVMs were considered to be a real number and we used Euclidean distance to compare between the n-dimensional output word and the valid codewords.

	on-vs-all	all pairs	BCH(7,4)	BCH(15,4)	BCH(31,5)	BCH(63,7)
Equal	43.5%	39.4%	44.4%	48.1%	45.5%	42.0%
Individual	**42.4%**	**39.2%**	**42.4%**	**39.8%**	**41.8%**	**41.6%**

Table 1. We show the error rate for the six selected codes and the two ways the hyperparameters where chosen.

From this results one can notice that it is a more wiser approach to train each classifier indepently and afterwards join their outputs together than seeking for the best parameters all together, because the tuning of each codeword individually get the best classifier for each set of labels. This can be explained if one sees each binary classifier as a hard-decisor, in this case the output of one classifier is not affected by rest of them and the best will be to train each one individually and also it is widely known that using soft outputs leads to better error correcting performance [7]. It can bee seen that the Achived results with the all-pairs and BCH(15,4) codes are as good as the best result reported in [1], where it was proposed the use of channel coding for non-binary classification with SVMs.

We have punctured the 6 proposed multi-class classifiers using the procedure described in the previous section for both ways of hyperparameters selection. We use the same 8-fold cross validation scheme and we show the result over the 7 people that were neither use for hyperparameter nor for classifier selection in Table 2.

	on-vs-all	all pairs	BCH(7,4)	BCH(15,4)	BCH(31,5)	BCH(63,7)
Equal	43.5% (11)	29.0% (29)	42.4% (6)	40.5% (8)	37.0% (9)	36.4% (30)
Individual	42.4% (11)	**24.9% (20)**	42.4% (7)	37.2% (10)	34.4% (10)	32.4% (14)

Table 2. The error rate for the six punctured codes are shown. The number in brackets indicates the remaining binary classifiers.

All the classifiers have provided a much better result after they were punctured. This is due to the binary classifiers with worst classification properties were pruned and the errors they induced were now corrected by the remaining classifers. The best result in this case is given by the all-pair approach in which we have a reduction of the test error over 14 percentual points and have reduced the number of binary classifiers from 55 to 21. We show in Table 3 the 21 pairs of classifiers and the used hyper-parameters.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
i	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
I	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
E	-1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
A	0	-1	0	0	0	-1	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0
a:	0	0	-1	0	0	0	-1	0	0	0	0	0	1	1	1	0	0	0	0	0	0
Y	0	0	0	-1	0	0	0	0	0	-1	0	0	-1	0	0	1	0	0	0	0	0
O	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0
C:	0	0	0	0	0	0	0	-1	-1	0	-1	0	0	-1	0	0	-1	0	0	1	0
U	0	0	0	0	0	0	0	0	0	0	0	-1	0	0	-1	0	0	0	0	0	1
u:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	0	-1	0	-1	-1
3:	0	0	0	0	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	0	0
C	.23	.23	.23	.23	.23	.5	.23	.23	.23	.23	.23	.23	5	.5	1	.23	10	5	.5	.5	10
σ	$\sqrt{\frac{d}{2}}$	$\sqrt{d}$	$\sqrt{\frac{d}{2}}$	$\sqrt{\frac{3d}{4}}$	$\sqrt{\frac{3d}{4}}$	$\sqrt{d}$	$\sqrt{d}$	$\sqrt{d}$	$\sqrt{d}$	$\sqrt{\frac{d}{2}}$	$\sqrt{d}$	$\sqrt{d}$	$\sqrt{\frac{3d}{4}}$	$\sqrt{d}$	$\sqrt{\frac{3d}{4}}$	$\sqrt{\frac{d}{2}}$	$\sqrt{d}$	$\sqrt{d}$	$\sqrt{\frac{d}{2}}$	$\sqrt{d}$	$\sqrt{\frac{d}{2}}$

Table 3. We show the 21 classifiers that present the best solution for the all-pairs code, which was the best one. We also show which were the hyper-parameters chosen by cross validation for each one of the binary classifiers, where d is the number of features (11).

5 Conclusions

In this paper we have reviewed the channel coding approach for multi-class problems using binary classifiers. We have shown the similarities but have also pointed out the differences. Among the differences we have underlined that the binary classifiers are not independent from each other and that as the number of them increases, the performance does not always improves. In order to solve this limitation, we have employed one of the tools from channel coding, known as puncturing, to prune the binary classifiers that do not improve the system performance and have shown that the probability of error can be significantly reduced and obtain, as a consequence, far less complex multi-classification schemes.

References

1. E. Allwein, R. Schapire, and Y. Singer. Reducing multiclass to binary: A unifying approach for margin classifiers. *Machine Learning Research*, p. 113-141, 2000.
2. T. G. Dietterich and G. Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, 2:263–286, 1995.
3. T. Hastie and R. Tibshirani. Clasification by pairwise coupling. *The annals of statistics*, 26(2):451–471, 1998.
4. B. Schoelkopf and A. Smola. *Learning with kernels.* M.I.T. Press, 2001.
5. B. Schölkopf, K.-K. Sung, C. J.C. Burges, F. Girosi, P. Niyogi, T. Poggio, and V. N. Vapnik. Comparing support vector machines with Gaussian kernels to radial basis function classifiers. *IEEE Trans. on Signal Processing*, 45(11):2758–2765, Nov. 1997.
6. J. Weston and C. Watkins. Multi-class support vector machines. Technical Report CSD-TR-98-04, Royal Holloway, University of London, UK, 1998.
7. S. B. Wicker. *Error Control Systems.* Prentice Hall, 1995.