Support Vector Machines in Multisource Classification

Gisli Hreinn Halldorsson, Jon Atli Benediktsson and Johannes R. Sveinsson
Engineering Research Institute, University of Iceland, Hjardarhaga 2-6, 107 Reykjavik, Iceland
e-mail: {ghh, benedikt, sveinsso}@hi.is

***Abstract*— The use of Support Vector Machines (SVMs) for classification of multisource data is investigated. SVMs have been shown to have difficulties in classifying multiclass data. To overcome that, the multiclass classification problem considered here was reduced to multiple margin-based binary problems. Several possibilities of binary problems were investigated, including one-against-all, all-pairs and nearly random decompositions of the multiclass problem. To combine the outputs from the binary problems three approaches were tested: a) voting schemes, b) two loss functions, and c) decoding function based on conditional probability estimation. An extension of the radial basis function kernel for multisource data is also proposed. The kernel concentrates on local distance between features from each data source. The experimental results show the proposed approach to be appropriate for multisource data classification.**

I. Introduction

The combined use of multisource remote sensing and geographic data has been shown to offer improved accuracies in land cover classification [1]. For such classification, the conventional parametric statistical classifiers, which have been applied successfully in remote sensing for the last three decades, are not appropriate, since a convenient multivariate statistical model does not exist in general for such data. Several methods have been proposed to classify multisource data. These methods include, e.g., statistical methods based on consensus theory, neural networks and boosting methods [1]. Here, we investigate the use of Support Vector Machines (SVMs) for classification of multisource data. The paper is organized as follows. First, Support Vector Machines are discussed in Section 2. Then, output coding for multiclass problems is reviewed in Section 3. Experimental results are given in Section 4, and, finally, conclusions are drawn in Section 5.

II. Support Vector Machines

Support Vector Machines (SVMs) are extensively used as classification tools in a variety of areas [2]. They map an input pattern $\mathbf{x}$ into a high dimensional feature space ($\mathbf{z} = \varphi(\mathbf{x})$) and construct an optimal hyperplane defined by $g(\mathbf{z}) = \mathbf{w}^T\mathbf{z} + b = 0$ where ($\mathbf{w}$ is a vector and b is a bias parameter) to separate examples from the classes. For an SVM with L_1 norm, soft-margin formulation, the primal problem needs to be solved:

$$\begin{array}{c} \min\limits_{\mathbf{W},\xi_i} \Big\{ \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{N}\xi_i \Big\}, \\ y_i(\mathbf{w}^T\mathbf{z}_i + b) \geq 1 - \xi_i, \\ \xi_i \geq 0 \quad \forall \quad i \in \{1,\ldots,N,\} \end{array} \tag{1}$$

where $\mathbf{x}_i$ is the i-th pattern, y_i is the class label value that takes the values $\{\pm 1\}$, ξ_i is a slack variable, C is a regularization parameter, and N is the number of patterns. The above problem is solved computationally using the solution of its dual form

$$\begin{array}{c} \max\limits_{\alpha_i, i\in\{1,\ldots,N\}} \Big\{ \sum_{i=1}^{N}\alpha_i \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j k(\mathbf{x}_i,\mathbf{x}_j) \Big\}, \\ \sum_{i=1}^{N} y_i\alpha_i = 0, \qquad 0 \leq \alpha_i \leq C \quad i \in \{1,\ldots,N\} \end{array} \tag{2}$$

where $k(\mathbf{x}_i,\mathbf{x}_j)\varphi(\mathbf{x}_i)\varphi(\mathbf{x}_j)$ is the kernel function, which performs the nonlinear mapping. An example of a popular kernel function is the radial basis function (RBF):

$$k(\mathbf{x}_i,\mathbf{x}_j)\exp\Big(-\gamma||\mathbf{x}_i - \mathbf{x}_j||_2\Big).$$

In some cases it could be useful to design a kernel that is specified for the problem at hand. In this paper we consider multisource classification where each data source has different characteristics. Therefore, it could be reasonable to handle each source separately. We propose a kernel that is an extension of the RBF kernel, instead of trying to estimate distance between vectors over all sources with $||\mathbf{x}_i - \mathbf{x}_j||_2$ we concentrate on the local distance between features from each source. Finally, we sum over all the distances and weight each distance by the number of features for the source. This kernel can be expressed as

$$k(\mathbf{x}_i,\mathbf{x}_j)\exp\left\{-\gamma\sum_{k}\omega_k \sum_{l\in\text{source}_k}||x_i[l] - x_j[l]||_2\right\}$$

where ω_k is number of features from source k, and $l \in \text{source}_k$ is feature l in source k.

III. Output Coding for Multiclass Problems

So far, we have discussed binary classification, where the class labels y can only take two values, ± 1. Suppose now that we are faced with a multiclass learning problem in which each label y is chosen from a set $\mathcal{Y}$ of cardinality $M = |\mathcal{Y}| > 2$. How can a binary margin-based learning algorithm (such as SVMs) be modified to handle M-class problem?

Several solutions have been proposed for the above problem [11]. Many involve reducing the multiclass problem, in one way or another, to a set of binary problems. Perhaps the simplest approach is to construct a set of binary classifiers $f^1,\ldots,f^M$, each trained to separate one class form the rest, and combine them by doing the multiclass classification

according to the maximal output before applying the sgn function [3], i.e.,

$$\hat{y} = \arg\max_{j=1,\ldots,M} g^j(\mathbf{x}) = \sum_{i=1}^{N} y_i \alpha_i^j k(\mathbf{x}, \mathbf{x}_i) + b_i, \quad f^j(\mathbf{x}) = \text{sgn}(g^j(\mathbf{x})). \tag{3}$$

The above approach is called the *one-against-all* approach.

Another approach, suggested by Hastie and Tibshirani [4], is to use the binary learning algorithm to distinguish each pair of classes. Thus, for each distinct pair, $r_1, r_2 \in \mathcal{Y}$, we run the learning algorithm on a binary problem in which examples labeled $y = r_1$ are considered positive, and those labeled $y = r_2$ are negative. All other examples are simply ignored. When we try to classify a test pattern, we evaluate all $\binom{M}{2}$ binary classifiers, and classify according to which of the classes gets the highest number of votes. A vote for a given class is assigned when a classifier puts a pattern into the class. We call this the *all-pairs* approach.

A more general method on handling multiclass problems was given by Dietterich and Bakiri [5]: Generate S binary problems by splitting the original set of classes into two subsets and train binary classifiers $f^1, \ldots, f^S$ for each problem. Each class corresponds to a unique vector $\{\pm 1\}^S$ and for M classes we obtain a "coding matrix" $\mathbf{M} \in \{\pm 1\}^{M \times S}$. Given a example $\mathbf{x}$, we then predict the label y for which row of matrix $\mathbf{M}$ is "closest" to output vector $\mathbf{f}(\mathbf{x}) = f^1(\mathbf{x}), \ldots, f^S(\mathbf{x})$ using the Hamming distance to estimate which row is "closest." This is the method of error correcting output codes (ECOC) based on the Hamming distance.

The ECOC method has been shown to produce good results in multiclass cases. However, it has been pointed out that it does not make use of the margin, which is a crucial quantity for classifiers. Recently [6], a version was developed that replaces the Hamming-based decoding with a more sophisticated scheme that takes margins into account. Coding matrix is used but taken from a larger set $\{-1, 0, +1\}^{M \times S}$. That is, some of the entries in $\mathbf{M}(r, s)$ may be zero, indicating that we do not care how classifier f^s categorizes examples with label r. Consequently, we are training classifiers f^s where $s \in [1, \ldots, S]$ is provided with labeled data of the form $\{\mathbf{x}_i, \mathbf{M}(y_i, s)\}$ for all examples in the training set but omits all examples for which $\mathbf{M}(y_i, s) = 0$. In order to choose a label r or a row in the matrix $\mathbf{M}$ that is most consistent (closest) with the magnitude of the predictions $\mathbf{g}(\mathbf{x}) = g^1(\mathbf{x}), \ldots, g^S(\mathbf{x})$, a distance is calculated with

$$d_L(\mathbf{M}(r), \mathbf{g}(\mathbf{x})) \sum_{s=1}^{S} L(\mathbf{M}(r, s) g^s(\mathbf{x})), \tag{4}$$

where $\mathbf{M}(r)$ is row r in matrix $\mathbf{M}$ and L is a loss function. The simplest loss function one can use is the linear loss for which $L(z) = -z$ but several other choices are possible like $L(z) = e^{-z}$ [7]. Then the label $\hat{y}$ is predicted with

$$\hat{y} = \arg\min_{r=1,\ldots,M} d_L(\mathbf{M}(r), \mathbf{g}(\mathbf{x})). \tag{5}$$

A loss function for the margin presents some advantage over the standard Hamming distance because it can encode the confidence of each classifier in the coding matrix. However, the confidence is a relative quantity, i.e., the range of the values of the margin may vary with the classifier. Thus, just using a linear loss function may introduce some bias in the final classification in the sense that classifiers with a large output range will receive a higher weight [7]. Another approach is to estimate the conditional probability of each output code bit O_s given the magnitude of the prediction $g^s(\mathbf{x})$, i.e., $P(O_s | g^s(\mathbf{x}))$. To do so, one can try to fit a parametric model on the output produced by a n-fold cross validation procedure. Platt [8] suggested to use a sigmod model:

$$P(O_s | g^s(\mathbf{x})) = \frac{1}{1 + \exp\{A_s g^s(\mathbf{x}) + B_s\}}, \tag{6}$$

motivated by the fact that the relationship between an SVM scores and empirical probabilities $P(O_s | g^s(x))$ appears to be sigmoidal for many data sets. Then, the conditional probability of an, SVM score y given code bits $O_1, \ldots, O_S$ in class r is

$$P(y = r | \mathbf{g}) = P(O_1 = \mathbf{M}(r, 1), \ldots, O_S \mathbf{M}(r, S) | \mathbf{g}) + \alpha \tag{7}$$

where α is a constant that collects the probability mass dispersed on the $2^S - M$ invalid codes. Assuming that O_s and O'_s ($O_s \neq O'_s$) are conditionally independent given $\mathbf{x}$ for each s and s', respectively, the likelihood that the resulting output code word is r can be written as

$$P(y = r | \mathbf{g}) = \prod_{s=1}^{S} P(O_s = \mathbf{M}(r, s)) + \alpha. \tag{8}$$

In the case, if α is small, the distance function will be

$$d(\mathbf{M}(r), \mathbf{g}) \approx -\log\{P(y = r | \mathbf{g})\}. \tag{9}$$

IV. Experiments

A. Colorado data set

Classification was performed on a data set consisting of the following 4 data sources:

- Landsat MSS data (4 spectral channels).
- Elevation data (in 10 m contour intervals,1 data channel).
- Slope data (0-90 degrees in 1 degree increments, 1 data channel).
- Aspect data (1-180 degrees in 1 degree increments, 1 data channel).

The area used for classification is mountainous area in Colorado. Ground reference data are available with 10 ground-cover classes. One is water; the others are forest types [1].

B. Implementation remarks

The two kernels introduced in Section II were used for classification along with most of the the output coding methods of Section III. The training accuracy was estimated with 4-fold cross validation using different kernel parameters γ and cost parameters C for each binary classifier. Therefore, for each binary classifier we tried $\gamma = [2^{-4}, 2^{-3}, 2^{-2}, \ldots, 2^5]$ and $C = [2^{-1}, 2^0, 2^{-1}, \ldots, 2^5]$, and selected the parameters that

gave the best training accuracy. The experiments were implemented in Matlab, python and C++ using LIBSVM [9]. For the output decoding, two loss function were tested: $L(z)e^{-z}$ and $L(z) = \log(1 + e^{-2z})$. In the voting approach, if two classes had identical votes, the one with the smaller index was selected. For probability decoding, 5-fold cross validation on the training set was used to fit the sigmoid model and transform the SVM scores to probabilities. The problem of designing an optimal binary code matrix $\mathbf{M} \in \{\pm 1\}^{M \times S}$ is NP-Complete [10]. For this reason we decided to select our code matrix $\mathbf{M} \in \{-1, 0, 1\}^{10 \times 50}$ from 10000 random generated code matrices where the probabilities for -1 and 1 were $\frac{1}{4}$, respectively. The "best" matrix has the greatest row and column separations and does not have rows or columns that contain only zeros. This approach of finding the code matrix is called *nearly random* below.

TABLE I

CLASSIFICATION ACCURACIES IN PERCENTAGE FOR DIFFERENT COMBINATIONS OF CODE MATRIX AND OUTPUT DECODING USING THE ORIGINAL RBF KERNEL.

Code Matrix	**Output Decoding**	**Training OA.**	**Test OA.**
One-Against-All	Max. Output	86.11	74.98
All-Pairs	$L(z) = e^{-z}$	85.22	74.68
	$L(z) = \log(1 + e^{-2z})$	86.01	74.98
	Voting	85.12	74.68
	Probability	85.71	74.78
Nearly Random	$L(z) = e^{-z}$	88.00	74.28
	$L(z) = \log(1 + e^{-2z})$	88.00	74.58
	Probability	87.50	75.07

TABLE II

CLASSIFICATION ACCURACIES IN PERCENTAGE FOR DIFFERENT COMBINATIONS OF CODE MATRIX AND OUTPUT DECODING USING THE EXTENDED RBF KERNEL.

Code Matrix	**Output Decoding**	**Training OA.**	**Test OA.**
One-Against-All	Max. Output	89.39	77.15
All-Pairs	$L(z) = e^{-z}$	85.02	77.65
	$L(z) = \log(1 + e^{-2z})$	85.32	77.84
	Voting	84.33	78.14
	Probability	84.82	77.05
Nearly Random	$L(z) = e^{-z}$	86.21	78.34
	$L(z) = \log(1 + e^{-2z})$	86.41	78.44
	Probability	84.82	77.45

C. Results and Discussions

Tables I and II summarize the overall accuracies using the original RBF and the extended kernels. As can be seen from the tables, it is not clear which combination of code matrix and output decoding is superior in terms of overall accuracy. But as pointed out in [11], the total training time for all-pairs is smaller than for one-against-all even though more classifiers need to be trained in all-pairs because each SVM optimization problem is smaller in that method. It is also noteworthy that using the nearly random approach to find the code matrix is not only computationally demanding but also does not take into account the underlying affinities among the individual classes (or meta-classes), e.g., their closeness or amount of separation [12]. Comparing the classification results in Tables I and II it is clear that the extended RBF kernel outperforms the original RBF in all cases in terms of test accuracies. The main drawback is that the extended RBF kernel is computationally a bit more demanding than the original RBF kernel, and has not been proven to fulfill the Mercer's conditions [3].

V. CONCLUSION

In this paper, we have compared methods for SVM output coding and proposed an extension of the RBF kernel for multisource data, which concentrates on the local distance between features from each source. According to our results, using an SVM with an extended RBF kernel seems to be a promising way to classify multisource data and is comparable in terms of overall accuracies to other advanced methods, which have been used to classify the data set [1].

ACKNOWLEDGMENT

The Colorado data set was originally acquired, preprocessed and loaned to us by Dr. Roger Hoffer of Colorado State University. Access to the data set is gratefully acknowledged. The research was funded in part by the Assistantship fund of the University of Iceland, The Research fund of the University of Iceland and the Icelandic Research Council.

REFERENCES

[1] G.J. Briem, J.A. Benediktsson and J.R. Sveinsson, "Multiple Classifiers Applied to Multisource Remote Sensing Data," *IEEE Trans. Geoscience and Remote Sensing*,vol. 40, pp. 2291-2299, 2002.

[2] V. Vapnik, *Statistical Learning Theory*, Wiley, New York 1998.

[3] B. Schlkopf and A.J. Smola, "Learning with Kernels, Support Vector Machines, Regularization, Optimaization, and Beyond," MIT Press, Cambridge 2002.

[4] T. Hastie and R. Tibshirani, "Classification by Pairwise Coupling," *The Annals of Statistics*, vol. 26(2), pp. 451-471, 1998.

[5] T.G. Dietterich and G. Bakiri, "Solving Multi-Class Learning Problems via Error-Correcting Output Codes," *Journal of Artifical Intelligence Research*, pp. 263-286, 1995.

[6] E.L. Allwein, R.E. Schapire, and Y. Singer, "Reducing Multiclass to Binary: A Unifying Approach for Margin Classifiers," *Journal of Machine Learning Research*, pp. 113-141, 2000.

[7] A. Passerini, M. Pontil and P. Frasconi, "From Margins to Probabilities in Multiclass Learning Problems," Proc. 15th *European Conf. on Artificial Intelligence*, 2002.

[8] J.C. Platt, "Probabilistic Outputs for Support Vector Machines and Comparisons to Regulaized Likelihood Methods," *Advances in Large Margin Classifiers*, A. Somola *et al.* (eds.) MIT Press, Cambridge 1998.

[9] C-C Chang and C-J Lin, LIBSVM: A library for support vector machines. Available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`, 2002.

[10] K. Crammer and Y. Singer, "On the Learnability and Design of Output Codes for Multiclass Problems," *Computational Learning Therory*, pp. 35-42, 2000.

[11] C-W Hsu and C-J Lin, "A Comparison of Methods for Multiclass Support Vector Machines," *IEEE Trans. Neural Networks*, vol. 13, pp. 415-425, 2002.

[12] J.T. Morgan, A. Henneguelle, M.M. Crawford, J. Ghosh, and A. Neuenschwander, "Adaptive Feature Spaces for Land Cover Classification with Limited Ground Truth Data," *Multiple Classifier Systems, MCS 2002*, LNCS 2364, Springer, Berlin 2002.