Linear Classifier Combination and Selection Using Group Sparse Regularization and Hinge Loss

Mehmet Umut Sen*, Hakan Erdogan

Vision and Pattern Analysis Laboratory
Sabanci University - Faculty of Engineering and Natural Sciences
Istanbul, Turkey
Phone: +902164839000-2117 Fax: +902164839550

Abstract

The main principle of stacked generalization is using a second-level generalizer to combine the outputs of base classifiers in an ensemble. In this paper, after presenting a short survey of the literature on stacked generalization, we propose to use regularized empirical risk minimization (RERM) as a framework for learning the weights of the combiner which generalizes earlier proposals and enables improved learning methods. Our main contribution is using group sparsity for regularization to facilitate classifier selection. In addition, we propose and analyze using the hinge loss instead of the conventional least squares loss. We performed experiments on three different ensemble setups with differing diversities on 13 real-world datasets of various applications. Results show the power of group sparse regularization over the conventional l_1 norm regularization. We are able to reduce the number of selected classifiers of the diverse ensemble without sacrificing accuracy.

*Corresponding authors
Email addresses: umutsen@sabanciuniv.edu (Mehmet Umut Sen), haerdogan@sabanciuniv.edu (Hakan Erdogan)

Preprint submitted to Pattern Recognition Letters *September 28, 2012*

With the non-diverse ensembles, we even gain accuracy on average by using group sparse regularization. In addition, we show that the hinge loss outperforms the least squares loss which was used in previous studies of stacked generalization.

Keywords:
classifier combination, group sparsity, classifier selection, regularized empirical risk minimization, hinge loss

1. Introduction

Classifier ensembles aim to increase the efficiency of classifier systems in terms of accuracy at the expense of increased complexity and they are shown to obtain greater performance than single-expert systems for a broad range of applications. Among all theoretical and practical reasons to prefer using ensembles, which are categorized as *statistical*, *computational* and *representational* in [7], the most important ones are the statistical reasons. Since we are looking for the generalization performance (error in the test data) in pattern recognition problems, it is often very difficult to find the "perfect classifier", but by combining multiple classifiers, probability of getting closer to the perfect classifier is increased. An ensemble may not always beat the performance of the best single classifier obtained, but it will surely decrease the variance of the classification error. Some other reasons besides statistical reasons can be found in [7, 20].

The straightforward method to obtain an ensemble is using different classifier types or different parameters. Also training base classifiers with different subsets or samplings of data or features is used to obtain more diverse ensembles. In this work, we are not interested in the methods of obtaining the ensemble, but we investigate various linear combination types for a given set of base classifiers.

Base classifiers produce either label outputs or continuous valued outputs. For the former, combiners like majority voting or weighted majority voting are used. In the latter case, base classifiers produce continuous scores for each class that represent the degree of support for each class. They can be interpreted as confidences in the suggested labels or estimates of the posterior probabilities for the classes [13]. In this paper, we deal with the combination of continuous valued outputs.

Combination rules can be grouped into trainable vs. non-trainable. Learning the combiner from training data is shown to give better accuracy than non-trainable combiners. Among trainable combiners, such as stacked generalization (stacking) [33], decision templates [13] and Dempster-Shafer combination [22]; stacked generalization is deeply investigated and analyzed in the literature [33, 29, 14, 28, 24, 18, 5, 25, 21, 30, 16].

1.1. Stacked Generalization

The idea of stacking is to use the confidence scores that are obtained from base classifiers as attributes in a new training set keeping the original class labels and training a meta-classifier with this new dataset. Linear meta-classifiers have speed and complexity advantage over non-linear ones and are usually preferred in the literature. When initially introduced, stacking is used to combine the class predictions of the base classifiers [33]. Ting & Witten used confidence scores of base classifiers as input features and improved stacking's performance [29, 28]. Merz used stacking and correspondence analysis to model the relationship between the learning examples and their classification by a collection of learned models and used nearest neighbor classifier as the meta learner [18]. A pool of representations obtained by a genetic algorithm is used to train different classifiers in [19], which are then combined by vote rule. Dzeroski & Zenko used multi-response model trees as the meta-learner [5]. Seewald introduced stackingC, which improves stacking's performance further and reduces the computational cost by introducing class-conscious combination [24]. Sill incorporated meta-features with the posterior scores of base classifiers to improve accuracy [25]. Ledezma, used genetic algorithms to search for good stacking configurations [16]. Tang, re-ranked all possible class labels according to the scores and obtained a learner which outperforms all base classifiers [27].

Since training the base classifiers and the combiner with the same data samples will result in overfitting, a sophisticated cross-validation approach is applied to obtain the training data of the combiner (level-1 data). This procedure, called internal cross-validation, is described in section 2. After obtaining level-1 data, there are two main problems remaining for a linear combination: (1) Which type of combination method should be used? (2) Given a combination type, how should we learn the parameters of the combiner? For the former problem, Ueda [31] defined three linear combination types namely type-1, type-2 and type-3; for which, we use the descriptive names: weighted sum (WS), class-dependent weighted sum (CWS) and linear stacked generalization (LSG) respectively, and investigate all of them. LSG is used in [14, 28], and CWS combination is proposed in [29, 24]. For the second main problem described above, Ting & Witten proposed a multi-response linear regression algorithm for learning the weights [29]. Ueda in [31] proposed using minimum classification error (MCE) criterion for estimating optimal weights, which increased the accuracies. MCE criterion is an approximation to the zero-one loss function which is not convex, so finding a global optimizer is not always possible. Ueda derived algorithms for different types of combinations with MCE loss using stochastic gradient methods. Both of these studies ignored "regularization" which has a huge effect on the performance, especially if the number of base classifiers is large. Reid & Grudic in [21] regularized the standard linear least squares estimation of the weights with CWS and improved the performance of stacking. They applied l_2 norm penalty, l_1 norm penalty and linear combination of the two (elastic net regression). In this work, we propose maximum margin algorithms for learning the optimal weights. We work with the regularized empirical risk minimization framework [15] and use the hinge loss function with l_2 regularization, which corresponds to the support vector machines (SVM). We do not derive optimization algorithms for the solutions of the minimization problems, but state-of-the-art solutions of SVM in the literature can be modified for our problem.

1.2. Sparse Combination

Another issue, recently addressed in [34], is combination with a sparse weight vector so that we do not use all classifiers in the ensemble. Since we do not have to use classifiers which have zero weight on the test phase, overall test time will be much less. Zhang formulated this problem as a linear programming problem for only the WS combination type [34]. Reid used l_1 norm regularization for CWS combination [21]. In this paper, we investigate sparsity issues for all three combination types: WS, CWS and LSG. We use both l_1 norm and $l_1 - l_2$ norm for regularization in the objective function for CWS and LSG. Latter regularization results in group sparsity, which is deeply investigated and successfully applied to various problems recently [17].

1.3. Organization of the Paper

Throughout the paper, we used m for the classifier subscript, n for the class subscript, i for the data instance subscript, M, N and I for the number of classifiers, classes and data instances respectively. Datapoint subscript i is sometimes dropped for simplicity. In Section 2 we explain the cross-validation technique used in stacked generalization. In Section 3, we define the classifier combination problem formally and define three different combination types used in the literature, namely WS, CWS and LSG. In Section 4, we explain how the weights are learned using regularized empirical risk minimization framework with hinge loss and a regularization function. In Section 5, we define sparse regularization functions to enable classifier selection. In Section 6, the experimental setups are described. In Section 7, we present the results of our experiments and discuss them. Section 8 finishes the paper with concluding remarks.

2. Internal Cross Validation

The basic idea of stacking is applying a meta-level (or level-1) generalizer to the outputs of base classifiers (or level-0 classifiers). For training the level-1 generalizer, we need the confidence scores (level-1 data) of the training data, but training the combiner with the same data instances which are used for training the base classifiers will lead to overfitting the database and eventually result in poor generalization performance. So we should split the dataset into two disjoint subsets for training the base classifiers and the combiner. But this partitioning leads to inefficient usage of the dataset. Wolpert deals with this problem by a sophisticated cross-validation method (internal CV), in which training data of the combiner is obtained by cross validation [33]. In k-fold cross-validation, training data is divided into k parts and each part of the data is tested with the base classifiers that are trained with the other $k-1$ parts of data. So at the end, each training instance's score is obtained from the base classifiers whose training data does not contain that particular instance. This procedure is repeated for each base classifier in the ensemble. We apply this procedure for the three different linear combination types.

3. Combination Types

3.1. Problem Formulation

In the classifier combination problem with confidence score outputs, input to the combiner are the posterior scores belonging to different classes obtained from the base classifiers. Let p_m^n be the posterior score of class n obtained from classifier m for any data instance. Let $\mathbf{p}_m = [p_m^1, p_m^2, \ldots, p_m^N]^T$, then the input to the combiner is $\mathbf{f} = [\mathbf{p}_1^T, \mathbf{p}_2^T, \ldots, \mathbf{p}_M^T]^T$, where N is the number of classes and M is the number of classifiers. Outputs of the combiner are N different scores representing the degree of support for each class. Let r^n be the combined score of class n and let $\mathbf{r} = [r^1, \ldots, r^N]^T$; then in general the combiner is defined as a function $g : \mathbb{R}^{MN} \to \mathbb{R}^N$ such that $\mathbf{r} = g(\mathbf{f})$. Let I be the number of training data instances, $\mathbf{f}_i$ contain the scores for training data point i obtained from base classifiers with internal CV and y_i be the corresponding class label; then our aim is to learn the g function using the data $\{(\mathbf{f}_i, y_i)\}_{i=1}^{I}$. On the test phase, label of a data instance is assigned as follows:

$$\hat{y} = \arg\max_{n \in [N]} r^n, \tag{1}$$

where $[N] = \{1, \ldots, N\}$. Among combination types, linear ones are shown to be powerful for the classifier combination problem. For linear combiners, the g function has the following form:

$$g(\mathbf{f}) = \mathbf{W}\mathbf{f} + \mathbf{b}. \tag{2}$$

In this case, we aim to learn the elements of $\mathbf{W} \in \mathbb{R}^{N \times MN}$ and $\mathbf{b} \in \mathbb{R}^N$. So, the number of parameters to be learned is $MN^2 + N$. This type of combination is the most general form of linear combiners and called type-3 combination in [31]. In the framework of stacking, we call it linear stacked generalization (LSG) combination. One disadvantage of this type of combination is that, since the number of parameters is high, learning the combiner takes a lot of time and may require a large amount of training data. To overcome this disadvantage, simpler but still strong combiner types are introduced with the help of the knowledge that p_m^n is the posterior score of class n. We call these methods weighted sum (WS) rule and class-dependent weighted sum (CWS) rule. These types are categorized as class-conscious combinations in [13].

3.2. Linear Combination Types

In this section, we describe and analyze three combination types, namely *weighted sum* rule (WS), *class-dependent weighted sum* rule (CWS) and *linear* *stacked generalization* (LSG) where LSG is already defined in (2).

3.2.1. Weighted Sum Rule

In this type of combination, each classifier is given a weight, so there are totally M different weights. Let u_m be the weight of classifier m, then the final score of class n is estimated as follows:

$$r^n = \sum_{m=1}^{M} u_m p_m^n = \mathbf{u}^T \mathbf{f}^n \quad , \quad n = 1, \ldots, N, \tag{3}$$

where $\mathbf{f}^n$ contains the scores of class n: $\mathbf{f}^n = [p_1^n, \ldots, p_M^n]^T$ and $\mathbf{u} = [u_1, \ldots, u_M]^T$. For the framework given in (2), WS combination can be obtained by letting $\mathbf{b} = 0$ and $\mathbf{W}$ to be the concatenation of constant diagonal matrices:

$$\mathbf{W} = [u_1 \mathbf{I}_N | \ldots | u_M \mathbf{I}_N], \tag{4}$$

where $\mathbf{I}_N$ is the $N \times N$ identity matrix. We expect to obtain higher weights for stronger base classifiers after learning the weights from the database.

3.2.2. Class-Dependent Weighted Sum Rule

The performances of base classifiers may differ for different classes and it may be better to use a different weight distribution for each class. We call this type of combination CWS rule. Let v_m^n be the weight of classifier m for class n, then the final score of class n is estimated as follows:

$$r^n = \sum_{m=1}^{M} v_m^n p_m^n = \mathbf{v}_n^T \mathbf{f}^n \quad , \quad n = 1, \ldots, N, \tag{5}$$

where $\mathbf{v}_n = [v_1^n, \ldots, v_M^n]^T$. There are MN parameters in a CWS combiner. For the framework given in (2), CWS combination can be obtained by letting

$\mathbf{b} = 0$ and $\mathbf{W}$ to be the concatenation of diagonal matrices; but unlike in WS, diagonals are not constant:

$$\mathbf{W} = [\mathbf{W}_1|\mathbf{W}_2|\dots|\mathbf{W}_M], \tag{6}$$

where $\mathbf{W}_m \in \mathbb{R}^{N\times N}$ are diagonal for $m = 1,\dots,M$.

3.2.3. Linear Stacked Generalization

This type of combination is the most general form of supervised linear combinations and is already defined in (2). With LSG, score of class n is estimated as follows:

$$r^n = \mathbf{w}_n^T\mathbf{f} + b_n \quad , \quad n = 1,\dots,N, \tag{7}$$

where $\mathbf{w}_n \in \mathbb{R}^{MN}$ is the n^{th} row of $\mathbf{W}$ and b_n is the n^{th} element of $\mathbf{b}$. LSG can be interpreted as feeding the base classifiers' outputs to a linear multi-class classifier as a new set of features. This type of combination may result in overfitting to the database and may yield lower accuracy than WS and CWS combination when there is not enough training data. From this point of view, WS and CWS combination can be treated as regularized versions of LSG. A crucial disadvantage of LSG is that the number of parameters to be learned is $MN^2 + N$ which will result in a long training period.

There is not a single superior one among these three combination types since results are shown to be data dependent [8]. A convenient way of choosing the combination type is selecting the one that gives the best performance in cross-validation.

4. Learning the Combiner

We use the regularized empirical risk minimization (RERM) framework [15] for learning the weights. In this framework, learning is formulated as an unconstrained minimization problem and the objective function consists of a summation of empirical risk function over data instances and a regularization function. Empirical risk is obtained as a sum of “loss” values obtained from each example. In general, we want to minimize the following objective function:

$$\phi(\mathbf{W}, \mathbf{b}) = \frac{1}{I} \sum_{i=1}^{I} \sum_{n=1}^{N} L(\mathbf{f}_i, y_i, n, \mathbf{w}_n) + \lambda R(\mathbf{W}). \tag{8}$$

where, L is the loss function. Different choices of loss functions and regularization functions correspond to different classifiers. Using the hinge loss function with l_2 norm regularization is equivalent to support vector machines (SVM). It has been shown in studies that the hinge loss function yields much better classification performance as compared to the least-squares (LS) loss function in general. Earlier classifier combination literature uses LS loss function [29, 28, 21], which is less favorable as compared to the hinge loss that we promote and use in this paper. Least-squares loss function is as follows:

$$L(\mathbf{f}_i, y_i, n, \mathbf{w}) = (s(y_i, n) - \mathbf{f}_i^T \mathbf{w}_n - b_n)^2, \tag{9}$$

where $s(y_i, n) = 1$ if $y_i = n$, -1 otherwise and b_n is the n^{th} element of $\mathbf{b}$. Instead of the s function, we can use the $\delta(y_i, n)$ which is zero if $y_i \neq n$ instead of -1. LS loss function forces the true class' scores to be one and wrong classes' scores to be zero or -1. This problem can be seen as a regression problem. Using least-squares with l_2 regularization is equivalent to applying least-squares support vector machine (LS-SVM) [26] to the level-1 data.

As mentioned above, we promote to use the hinge loss function for the combiner. Using the hinge loss function with the l_2 norm regularization is equivalent to using Support Vector Machine classifier. SVMs were originally designed for binary classification and there are a lot of ongoing research on how to effectively extend it for multiclass classification. We use the method defined by Crammer and Singer [3]. With this method, we find the linear separating hyper-plane that maximizes the margin between true class and the most offending wrong class. When we apply this idea to our problem, we obtain the following unconstrained minimization problem for LSG:

$$\phi_{LSG}(\mathbf{W}, \mathbf{b}) = \frac{1}{I} \sum_{i=1}^{I} (1 - r_i^{y_i}(\mathbf{W}) + \max_{n \neq y_i} r_i^n(\mathbf{W}))_+ + \lambda R_{LSG}(\mathbf{W}), \qquad (10)$$

where $R_{LSG}(\mathbf{W})$ is the regularization function, $(x)_+ = \max(0, x)$ and $r_i^n(\mathbf{W})$ is the posterior score of data instance i for class n with the combiner $\mathbf{W}$:

$$r_i^n(\mathbf{W}) = \mathbf{w}_n^T \mathbf{f}_i + b_n. \qquad (11)$$

$\lambda \in \mathbb{R}$ in (10) is the regularization parameter which is usually learned by cross validation. The objective function given in (10) encourages the distance between the true class' score and the most offending wrong class' score to be larger than one. A conventional regularization function is the Frobenius norm of $\mathbf{W}$:

$$R_{LSG}(\mathbf{W}) = ||\mathbf{W}||_F^2 = \sum_{n=1}^{N} ||\mathbf{w}_n||_2^2, \qquad (12)$$

Equation (10) is given for LSG but it can be modified for other types of combinations using the unifying framework described in [8]. But we also give objective functions for WS and CWS explicitly. The objective function for WS is as follows:

$$\phi_{WS}(\mathbf{u}) = \frac{1}{I}\sum_{i=1}^{I}(1 - \mathbf{u}^T\mathbf{f}_i^{y_i} + \max_{n\neq y_i}(\mathbf{u}^T\mathbf{f}_i^n))_+ + \lambda R_{WS}(\mathbf{u}). \tag{13}$$

For regularization, we use the l_2 norm of $\mathbf{u}$: $R_{WS} = ||\mathbf{u}||_2^2$. For CWS, we have the following objective function:

$$\phi_{CWS}(\mathbf{V}) = \frac{1}{I}\sum_{i=1}^{I}(1 - \mathbf{v}_{y_i}^T\mathbf{f}_i^{y_i} + \max_{n\neq y_i}(\mathbf{v}_n^T\mathbf{f}_i^n))_+ + \lambda R_{CWS}(\mathbf{V}), \tag{14}$$

where $\mathbf{V} \in \mathbb{R}^{M\times N}$ contains the weights for different classes: $\mathbf{V} = [\mathbf{v}_1, \ldots, \mathbf{v}_N]$. As for LSG, conventional regularization function for CWS is the Frobenious norm of $\mathbf{V}$: $R_{CWS}(\mathbf{V}) = ||\mathbf{V}||_F^2$.

5. Sparse Regularization

In this section, we define a set of regularization functions for enforcing sparsity on the weights so that the resulting combiner will not use all the base classifiers leading to a shorter test time. This method can be seen as a classifier selection algorithm, but here classifiers are selected automatically and we cannot determine the number of selected classifiers beforehand. But we can lower this number by increasing the weight of the regularization function (λ). With sparse regularization, λ has two main effects on the resulting combiner. First, it will determine how much the combiner should fit the data. Decreasing λ results in more fitting the training data and decreasing it too much results in overfitting, on the other hand, increasing it too much prevents the combiner to learn from the data and the accuracy drops dramatically. Secondly, as mentioned before, it will determine the number of selected classifiers. As λ increases, the number of selected classifiers decreases.

5.1. Regularization with the l_1 Norm

The most successful approach for inducing sparsity is using the l_1 norm of the weight vector for WS [34]:

$$R_{WS}(\mathbf{u}) = ||\mathbf{u}||_1, \tag{15}$$

For CWS and LSG, we have the following sparse regularization functions:

$$R_{CWS}(\mathbf{V}) = ||\mathbf{V}||_{1,1} = \sum_{n=1}^{N} ||\mathbf{v}_n||_1, \tag{16}$$

$$R_{LSG}(\mathbf{W}) = ||\mathbf{W}||_{1,1} = \sum_{n=1}^{N} ||\mathbf{w}_n||_1. \tag{17}$$

If all weights of a classifier are zero, that classifier will be eliminated and we do not have to use that base classifier for a test instance, so that testing will be faster. But the problem with l_1-norm regularizations for CWS and LSG is that we are not able to use all the information from a selected base classifier, because a classifier may receive both zero and non-zero weights. To overcome this problem, we propose to use group sparsity, as explained in the next section.

5.2. Regularization with Group Sparsity

We define another set of regularization functions which are embedded by group sparsity [17] for LSG and CWS to enforce classifier selection. The main principle of group sparsity is enforcing all elements that belong to a group to be zero altogether. Grouping of the elements are done before learning. In classifier combination, posterior scores obtained from each base classifier form a group. The following regularization function yields group sparsity for LSG:

$$R_{LSG}(\mathbf{W}) = \sum_{m=1}^{M} ||\mathbf{W}_m||_F. \tag{18}$$

For CWS, we use the following regularization:

$$R_{CWS}(\mathbf{V}) = ||\mathbf{V}||_{1,2} = \sum_{m=1}^{M} ||\mathbf{v}^m||_2, \tag{19}$$

where $\mathbf{v}^m$ is the m^{th} row of $\mathbf{V}$, so it contains the weights of the classifier m. After the learning process, the elements of $\mathbf{v}^m$ for any m are either all zero or all non-zero. This leads to better performance than l_1 regularization for automatic classifier selection, as we show in Section 7. In the next section, we describe the setup of the experiments.

6. Experimental Setups

We have performed extensive experiments in 13 real-world datasets from the UCI repository [1] and other sources[1]. For a summary of the characteristics of the datasets and the sources, see Table 1. In order to obtain statistically significant results, we applied 5x2 cross-validation [6] which is based on 5 iterations of 2-fold cross-validation (CV). In this method, for each CV, data are randomly split into two stacks as training and testing, resulting in overall 10 stacks for each database.

We constructed three ensembles which differ in the construction method and their diversity. In the first ensemble, we construct 10 different subsets

[1] Code can be downloaded from http://myweb.sabanciuniv.edu/umutsen/research/

randomly which contain 80% of the original data. Then, 13 different classifiers are trained with each subset resulting in a total of 130 base classifiers. We used PR-Tools [23] and Libsvm toolbox [2] for obtaining the base classifiers. These 13 different classifiers are: normal densities based linear classifier (ldc), normal densities based quadratic classifier (qdc), nearest mean classifier (nmc), k-nearest neighbor classifier (knnc), polynomial classifier (polyc), general kernel/dissimilarity based classification (kernelc), normal densities based classifier with independent features (udc), Parzen classifier (parzenc), binary decision tree classifier (treec), linear perceptron (perlc), SVM with linear kernel, polynomial kernel, and radial basis function (RBF) kernel. We used default parameters of the toolboxes. Average test error percentages over 10 different subsets and 10 stacks of 5x2 CV of 13 different base classifier types are given in Table 2. In the second ensemble setup, we trained a total of 154 SVM's with different kernel functions and parameters. Latter method produces less diverse base classifiers as compared to the former one. Third ensemble setup is the same as the first one, except the perturbation of the base classifiers are obtained with Random Subspace method [10]. In this case, each subset is obtained by choosing half of the features randomly, then 13 classifiers are applied for each subset. For some datasets, LSG combination could not be performed because of memory limitations.

Training data of the combiner is obtained by 4-fold internal CV. For each stack in 5×2 CV, 2-fold CV is used to obtain the optimal λ in the regularization function, i.e., λ which gives the best average accuracy in CV [2]. For the

[2] We searched for λ in $\{10^{-11}, 10^{-9}, 10^{-7}, 10^{-5}, 10^{-3}, 0.005, 0.01, 0.05, 0.1, 0.5, 1, 10\}$

minimization of the objective functions, we used the CVX-toolbox [9]. We use the Wilcoxon signed-rank test for identifying the statistical significance of the results with one-tailed significant level $\alpha = 0.05$ [4].

Table 1: Properties of the data sets used in the experiments [4]

DB	# of Instances	# of classes	# of features
Segment	2310	7	19
Waveform	5000	3	21
Robot	5456	4	24
Statlog	846	4	18
Vowel	990	11	10
Wine	178	3	13
Yeast	1484	9	8
Steel	1941	7	27
Svmguide4 [5]	612	6	10
Protein [6]	5000	3	352
Svmguide2 [7]	391	3	20
DNA [8]	3186	3	180
Cardio	2126	10	22

7. Results

First, we investigate the performance of the regularized learning of the weights with the hinge loss compared to the conventional least squares loss

[4]Full names of some datasets: *"Image Segmentation"* (*Segment*), *"Waveform Database Generator (Version 1)"* (*Waveform*), *"Wall-Following Robot Navigation Data"* (*Robot*), *"Statlog (Vehicle Silhouettes)"* (*Statlog*), *"Connectionist Bench (Vowel Recognition - Deterding Data)"* (*Vowel*), *"Steel Plates Faults"* (*Steel*), *"Cardiotocography"* (*Cardio*).

[5]Dataset is provided at [11]

[6]Dataset is provided at [32]

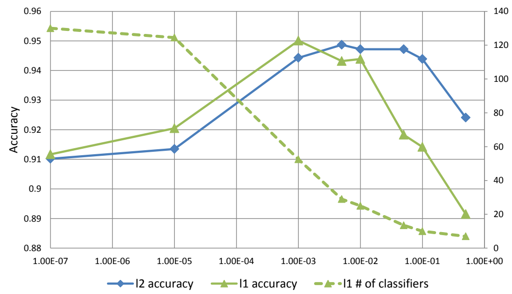
[7]Dataset is provided at [11]

[8]Dataset is provided at [12]

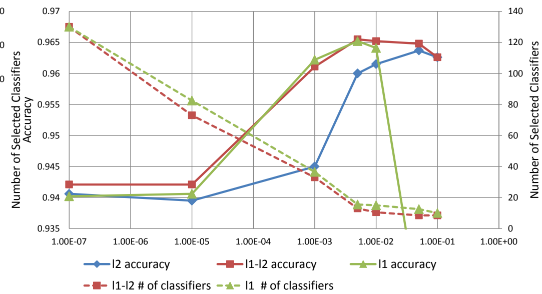
[21] and the multi-response linear regression (MLR) method which does not contain regularization [29] with the diverse ensemble setup described in Section 6. It should be noted that results shown here and in [21, 29] are not directly comparable since constructions of the ensembles are different. Error percentages of our method (hinge loss with l_2 regularization), least squares method, and MLR method for WS, CWS and LSG are given in Table 3. We also compared the results with simpler combination types, depicted in columns *EW*, *EW-Norm*, *EW-HP*, *WS-Simple*. Results for the simple sum rule, which is equivalent to using equal weights in the WS, are given in the column titled *EW*. *EW-Norm* is the simple sum rule with base classifier scores that are normalized to have mean zero and variance one. In *EW-HP*, base classifiers that have lower CV accuracy than the mean of all base classifier CV accuracies are not retained in the fusion. *WS-Simple* is a simple weighted-sum rule, where weight of each classifier is set to 4-fold CV accuracy of that base classifier. First entries in the boxes are the means of error percentages over 5×2 CV stacks and the second entries are the standard deviations. Star symbols (*) under the hinge loss column indicate that results of the hinge loss function are significantly different from the results of the least squares loss function with the corresponding combination type, i.e., WS, CWS, or LSG.

In most datasets, hinge loss function outperforms the LS loss function for the diverse ensemble. On almost all datasets, MLR method results in higher error percentages compared to other methods, and this shows the power of regularized learning, especially if the number of base classifiers is high. It should be noted that in [29], 3 base classifiers are used and here we use 130 base classifiers. *WS-Simple* results in the lowest error percentage for *Yeast* dataset, but this result is not statistically significant. For all other datasets except *Svmguide-2*, performance differences between the best method and all four simple combination types (*EW*, *EW-Norm*, *EW-HP*, *WS-Simple*) are statistically significant.

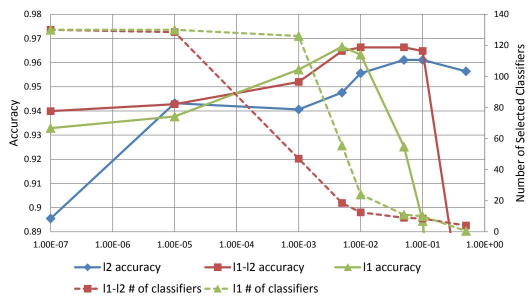
We also investigated the performance of sparse regularization with the hinge loss function. We used two different ensemble setups described in the beginning of this section. Regularization parameter λ given in the objective functions (10,13,14) is an important parameter and if we minimize the objective functions also over λ, the combiner will overfit the training data, which will result in poor generalization performance. Therefore, we used 2-fold cross-validation to learn the optimal parameter. We plot the relation of λ with accuracies and the number of selected classifiers for different regularizations with WS, CWS and LSG for the *Robot* dataset in Figures 1a, 1b and 1c respectively. In these figures, dashed lines correspond to the number of selected classifiers and solid lines correspond to the accuracies. The $l_1 - l_2$ label represents group sparsity. In all sparse regularizations, the best accuracies are obtained when most of the base classifiers are eliminated. For all regularizations, accuracies make a peak at λ values between 0.001 and 0.1. For l_1 norm regularization, accuracies drop dramatically with a small increase in λ. However, with group sparse regularization, accuracies remain high in a larger range for λ than that with the l_1 norm regularization. Thus the performance of l_1 regularization is more sensitive to the selection of λ. So we can say that the $l_1 - l_2$ norm regularization is more robust than the l_1 norm regularization. As the number of selected classifiers decreases, ac-



(a) WS combination



(b) CWS combination



(c) LSG combination

Figure 1: Accuracy and Number of selected classifiers vs. λ for WS, CWS and LSG combination of Robot data with the diverse ensemble setup

curacies increase for a large range of λ in general, but this increase in the accuracy cannot be attributed only to the classifier selection, because λ also determines how much the combiner should fit the data.

Next, we show the test results for all combination types with various regularization functions. Error percentages and corresponding number of selected classifiers (mean $\pm$ standard deviation) are shown in Table 4 for the diverse ensemble setup. In the significance column, denoted by *SIG*, the letters "a,b,c,d,e" denote that the accuracy-performances between (l_2, l_1) for WS, (l_2,$l_1 - l_2$), (l_1,$l_1 - l_2$) for CWS and (l_2,$l_1 - l_2$), (l_1,$l_1 - l_2$) for LSG are statistically significant respectively.

In general, we are able to use much less base classifiers with sparse regularizations with the cost of a small decrease in the accuracies. For LSG, average error percentage of group sparsity is a little less than that of the l_1 norm regularization. But the number of selected base classifiers is much less. So if classifier selection is desired, we suggest to use either CWS or LSG combination with $l_1 - l_2$ regularization. If training time is also crucial, CWS with $l_1 - l_2$ regularization seems to be the best option.

Error percentages and number of selected classifiers for the non-diverse ensembles are given in Tables 5. We also compared with the test error percentages of base classifiers which has highest CV accuracy, under the column "BC". With the non-diverse ensembles we are even able to increase the accuracy with much less number of base classifiers with sparse regularization in CWS and LSG. For LSG combination, $l_1 - l_2$ regularization results in lower error percentages than l_1 regularization on four datasets with lower number of base classifiers except the *Waveform* dataset. In general, the number of selected base classifiers of $l_1 - l_2$ regularization is much less than that of l_1 regularization. Except the *Statlog* dataset, the lowest error percentages are obtained with the sparse combinations with much less base classifiers than that of l_2 regularization which uses 154 base classifiers. If we compare different combination types with the l_2 norm, on average we see that, unlike in the diverse ensemble setup, WS and/or CWS outperforms LSG in all databases. We can conclude that if the posterior scores obtained from base classifiers are correlated, non-complex combiners, such as WS and CWS, are more powerful since complex combiners may result in overfitting.

Results for the third ensemble (random subspace) is presented at Table 6. We see similar results with the diverse ensemble setup, but in general, random subspace methods yields higher error rates than the diverse ensemble setup.

8. Conclusion

In this paper, we suggested using group sparse regularization for learning the parameters of linear combiners in stacked generalization. Results indicate that group sparse regularization outperforms the conventional l_1 norm regularization, and we can use smaller number of base classifiers with a small sacrifice in the accuracy with the diverse ensemble, so that the test time is shortened. With the non-diverse ensemble setup, we even obtain better accuracies using sparse regularizations on some datasets. We also proposed using the hinge loss function in the regularized empirical risk minimization framework, and we are able to obtain better accuracies with the hinge loss function than conventional least-squares estimation of the weights. We performed experiments for three different combination types and compared them. If training time is important, we suggest using the CWS type combination. And if test time is also important, we suggest using group sparse regularization.

9. Acknowledgments

This research is supported by The Scientific and Technological Research Council of Turkey (TUBITAK) under the scientific and technological research support program (code 1001), project number 107E015 entitled "Novel Approaches in Audio Visual Speech Recognition".

References

[1] Asuncion, A., Newman, D., 2007. UCI machine learning repository.

[2] Chang, C.C., Lin, C.J., 2001. LIBSVM: a library for support vector machines. Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

[3] Crammer, K., Singer, Y., 2000. On the learnability and design of output codes for multiclass problems, in: In Proceedings of the Thirteenth Annual Conference on Computational Learning Theory, pp. 35–46.

[4] Demšar, J., 2006. Statistical comparisons of classifiers over multiple data sets. J. Mach. Learn. Res. 7, 1–30.

[5] Deroski, S., Zenko, B., 2002. Is combining classifiers better than selecting the best one, in: Proceedings of the Nineteenth International Conference on Machine Learning, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA. pp. 123–130.

[6] Dietterich, T.G., 1998. Approximate statistical tests for comparing supervised classification learning algorithms. Neural Computation 10, 1895–1923.

[7] Dietterich, T.G., 2000. Ensemble methods in machine learning, in: International Workshop on Multiple Classifier Systems, Springer-Verlag. pp. 1–15.

[8] Erdogan, H., Sen, M., 2010. A unifying framework for learning the linear combiners for classifier ensembles, in: Pattern Recognition (ICPR), 2010 20th International Conference on, pp. 2985 –2988.

[9] Grant, M., Boyd, S., 2011. CVX: Matlab software for disciplined convex programming, version 1.21. Available at http://cvxr.com/cvx.

[10] Ho, T.K., 1998. The random subspace method for constructing decision forests. IEEE Trans. Pattern Anal. Mach. Intell. 20, 832–844.

[11] Hsu, C.W., Chang, C.C., Lin, C.J., 2000. A practical guide to support vector classification.

[12] Hsu, C.W., Lin, C.J., 2002. A comparison of methods for multiclass support vector machines.

[13] Kuncheva, L.I., 2004. Combining Pattern Classifiers: Methods and Algorithms. Wiley-Interscience.

[14] LeBlanc, M., Tibshirani, R., 1993. Combining Estimates in Regression and Classification. Technical Report. Journal of the American Statistical Association.

[15] Lecun, Y., Chopra, S., Hadsell, R., Huang, F.J., Bakir, G., Hofman, T., Schlkopf, B., Smola, A., (eds, B.T., 2006. A tutorial on energy-based learning, in: Predicting Structured Data, MIT Press.

[16] Ledezma, A., Aler, R., Sanchis, A., Borrajo, D., 2010. Ga-stacking: Evolutionary stacked generalization. Intell. Data Anal. 14, 89–119.

[17] Majumdar, A., Ward, R., 2009. Classification via group sparsity promoting regularization, in: Acoustics, Speech and Signal Processing, 2009. ICASSP 2009. IEEE International Conference on, pp. 861 –864.

[18] Merz, C.J., Stolfo, S., 1998. Using correspondence analysis to combine classifiers, in: Machine Learning, Kluwer Academic Publishers. pp. 33–58.

[19] Nanni, L., Lumini, A., 2009. A genetic encoding approach for learning methods for combining classifiers. Expert Syst. Appl. 36, 7510–7514.

[20] Polikar, R., 2006. Ensemble based systems in decision making. Ieee Circuits And Systems Magazine 6, 21–45.

[21] Reid, S., Grudic, G., 2009. Regularized linear models in stacked generalization, in: Proceedings of the 8th International Workshop on Multiple Classifier Systems, Springer-Verlag, Berlin, Heidelberg. pp. 112–121.

[22] Rogova, G., 1994. Combining the results of several neural network classifiers. Neural Netw. 7, 777–781.

[23] R.P.W., D., P., J., P., P., E., P., D., d.R., D.M.J., T., S., V., 2007. PRTools4.1, A Matlab Toolbox for Pattern Recognition. Delft University of Technology.

[24] Seewald, A.K., 2002. How to make stacking better and faster while also taking care of an unknown weakness, in: Proceedings of the Nineteenth International Conference on Machine Learning, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA. pp. 554–561.

[25] Sill, J., Takács, G., Mackey, L., Lin, D., 2009. Feature-weighted linear stacking. CoRR abs/0911.0460.

[26] Suykens, J., Vandewalle, J., 1999. Least squares support vector machine classifiers. Neural Processing Letters 9, 293–300. 10.1023/A:1018628609742.

[27] Tang, B., Chen, Q., Wang, X., Wang, X., 2010. Reranking for stacking ensemble learning, in: Proceedings of the 17th international conference on Neural information processing: theory and algorithms - Volume Part I, Springer-Verlag, Berlin, Heidelberg. pp. 575–584.

[28] Ting, K.M., Witten, I.H., 1997. Stacking bagged and dagged models, in: In Proc. 14th International Conference on Machine Learning, Morgan Kaufmann. pp. 367–375.

[29] Ting, K.M., Witten, I.H., 1999. Issues in stacked generalization. J. Artif. Int. Res. 10, 271–289.

[30] Todorovski, L., Deroski, S., 2000. Combining multiple models with meta decision trees, in: Zighed, D., Komorowski, J., Zytkow, J. (Eds.), Principles of Data Mining and Knowledge Discovery. Springer Berlin / Heidelberg. volume 1910 of *Lecture Notes in Computer Science*, pp. 69–84. 10.1007/3-540-45372-5-6.

[31] Ueda, N., 2000. Optimal linear combination of neural networks for improving classification performance. IEEE Trans. Pattern Anal. Mach. Intell. 22, 207–215.

[32] Wang, J.Y., 2002. Application of Support Vector Machines in Bioinformatics. Master's thesis.

[33] Wolpert, D.H., 1992. Stacked generalization. Neural Netw. 5, 241–259.

[34] Zhang, L., Zhou, W.D., 2011. Sparse ensembles using weighted combination methods based on linear programming. Pattern Recognition 44, 97 – 106.

Table 2: Error percentages for base classifiers in the diverse ensemble setup (*mean* ± *standard deviation*)×100.

DB	ldc	qdc	nmc	knnc	polyc	kernelc	udc	parzenc	treec	perlc	svm-linear	svm-poly	svm-rbf
Segment	8.80 ± 0.58	13.01 ± 1.79	27.61 ± 1.95	**5.57 ± 0.69**	13.30 ± 1.50	8.77 ± 1.08	21.32 ± 1.88	9.60 ± 2.14	12.73 ± 2.24	57.98 ± 5.54	29.23 ± 4.86	57.93 ± 10.98	50.26 ± 10.38
Waveform	14.59 ± 0.77	15.80 ± 0.56	19.98 ± 0.51	14.70 ± 0.56	14.65 ± 0.79	**13.58 ± 0.69**	19.09 ± 0.47	16.37 ± 0.59	29.22 ± 0.95	19.28 ± 2.99	13.37 ± 0.75	20.11 ± 2.32	13.09 ± 0.62
Robot	34.62 ± 0.99	31.33 ± 0.93	44.72 ± 0.89	15.41 ± 0.70	35.61 ± 0.98	20.49 ± 1.13	47.25 ± 1.38	16.37 ± 0.68	**5.87 ± 0.79**	42.72 ± 4.02	29.48 ± 0.99	31.73 ± 0.65	27.48 ± 1.00
Statlog	22.47 ± 1.68	**17.51 ± 1.34**	61.24 ± 1.29	39.31 ± 2.15	23.05 ± 1.91	30.37 ± 1.74	54.42 ± 3.08	39.08 ± 2.06	33.29 ± 2.49	29.82 ± 3.90	64.58 ± 4.08	67.78 ± 6.91	63.95 ± 5.41
Vowel	41.44 ± 2.81	20.34 ± 2.35	43.53 ± 2.99	**8.06 ± 1.51**	55.36 ± 3.30	22.52 ± 2.44	36.33 ± 4.20	8.14 ± 1.47	42.96 ± 4.09	65.80 ± 4.00	59.12 ± 6.08	65.13 ± 6.69	65.03 ± 6.83
Wine	2.91 ± 2.28	8.28 ± 5.68	27.93 ± 3.32	32.15 ± 4.02	**2.78 ± 1.98**	27.90 ± 3.47	2.91 ± 2.20	31.65 ± 3.41	16.52 ± 5.10	3.08 ± 2.31	36.93 ± 6.79	50.08 ± 15.74	46.43 ± 17.28
Yeast	**41.73 ± 1.35**	79.80 ± 10.37	49.82 ± 1.27	43.76 ± 1.32	47.48 ± 1.24	42.14 ± 1.24	80.94 ± 11.23	44.16 ± 1.64	56.44 ± 2.93	59.08 ± 9.28	43.01 ± 1.62	45.91 ± 3.17	43.38 ± 1.83
Steel	**32.59 ± 0.90**	40.28 ± 2.12	83.87 ± 2.32	52.10 ± 1.43	34.99 ± 1.52	50.92 ± 1.28	42.29 ± 2.11	52.16 ± 3.36	41.30 ± 2.33	44.51 ± 4.70	51.63 ± 2.22	65.91 ± 2.73	51.34 ± 1.23
Svmguide4	28.39 ± 5.23	**22.28 ± 3.26**	79.31 ± 2.33	67.87 ± 2.80	35.41 ± 4.68	61.71 ± 2.77	53.41 ± 7.03	80.41 ± 1.58	31.29 ± 4.58	45.48 ± 9.45	76.19 ± 7.11	81.79 ± 2.15	77.55 ± 6.18
Protein	39.81 ± 0.88	51.74 ± 0.96	38.79 ± 0.96	49.18 ± 1.30	39.62 ± 0.88	42.86 ± 1.05	52.40 ± 3.19	56.59 ± 1.18	58.86 ± 1.05	43.01 ± 1.23	39.85 ± 0.80	53.14 ± 8.91	**37.80 ± 1.16**
Svmguide2	19.85 ± 1.99	43.61 ± 13.40	25.17 ± 2.68	24.97 ± 2.59	20.59 ± 2.46	**17.59 ± 1.93**	22.08 ± 2.10	23.27 ± 2.30	37.99 ± 3.29	25.97 ± 3.70	20.28 ± 2.07	23.10 ± 3.16	21.31 ± 2.19
DNA	6.72 ± 0.59	34.52 ± 1.15	11.06 ± 0.57	12.82 ± 1.66	6.35 ± 0.54	9.64 ± 0.81	**5.44 ± 0.47**	27.99 ± 1.27	19.52 ± 1.61	9.66 ± 0.95	8.86 ± 0.46	12.98 ± 1.69	5.98 ± 0.69
Cardio	**26.86 ± 1.07**	28.65 ± 1.93	62.74 ± 1.66	32.02 ± 1.44	35.01 ± 1.91	32.53 ± 1.58	41.59 ± 2.99	36.46 ± 2.32	40.58 ± 3.82	39.88 ± 5.15	44.93 ± 1.94	55.57 ± 3.42	45.50 ± 1.78

Table 3: Error percentages in the diverse ensemble setup (*mean* ± *standard deviation*)×100.

DB	Hinge Loss with l_2 regularization			LS Loss with l_2 regularization			MLR			EW	EW-Norm	EW-HP	WS-Simple
	WS	CWS	LSG	WS	CWS	LSG	WS	CWS	LSG				
Segment	5.02 ± 0.88 *	3.90 ± 1.00	3.60 ± 1.05	6.34 ± 0.78	**3.54 ± 0.82**	3.57 ± 0.96	7.20 ± 1.02	6.66 ± 6.64	61.28 ± 9.35	7.37 ± 1.03	7.57 ± 1.36	5.69 ± 0.72	6.81 ± 1.06
Waveform	13.20 ± 0.69	**13.05 ± 0.72** *	**13.05 ± 0.65** *	13.19 ± 0.73	13.17 ± 0.72	13.18 ± 0.69	13.33 ± 0.68	14.10 ± 0.56	18.40 ± 7.06	14.17 ± 0.60	13.47 ± 0.65	13.26 ± 0.61	14.06 ± 0.62
Robot	3.95 ± 0.42 *	2.59 ± 0.33	2.61 ± 0.28 *	5.29 ± 0.61	2.55 ± 0.30	**2.53 ± 0.31**	5.05 ± 0.62	2.58 ± 0.30	3.19 ± 0.49	18.58 ± 0.61	23.75 ± 0.93	10.72 ± 1.11	16.43 ± 0.52
Statlog	16.34 ± 1.15 *	**16.12 ± 1.53** *	16.36 ± 1.67	16.78 ± 1.62	16.74 ± 1.91	16.88 ± 1.71	17.73 ± 2.11	58.01 ± 15.38	75.72 ± 6.18	23.03 ± 2.33	25.77 ± 1.32	19.03 ± 1.80	19.86 ± 2.12
Vowel	13.84 ± 2.73	7.66 ± 2.29 *	**6.32 ± 1.99**	13.90 ± 2.63	6.42 ± 2.06	6.46 ± 2.22	17.15 ± 2.31	10.08 ± 1.75	9.76 ± 1.14	14.53 ± 3.30	16.34 ± 2.74	10.16 ± 2.11	11.84 ± 2.23
Wine	1.57 ± 1.09	**1.12 ± 1.40** *	1.69 ± 1.32	2.36 ± 1.54	2.13 ± 2.21	2.13 ± 1.79	3.71 ± 2.31	8.20 ± 16.19	2.47 ± 1.66	2.81 ± 1.52	4.49 ± 1.67	1.24 ± 1.35	2.13 ± 1.35
Yeast	40.36 ± 1.21	40.23 ± 1.29	40.32 ± 1.19	40.26 ± 1.06	40.62 ± 1.44	40.94 ± 1.70	41.05 ± 1.04	53.11 ± 6.88	74.45 ± 6.42	40.26 ± 1.10	46.99 ± 9.24	40.44 ± 0.86	**40.22 ± 0.97**
Steel	29.85 ± 1.86 *	28.27 ± 1.38 *	**27.41 ± 1.22**	30.73 ± 2.02	27.52 ± 1.17	27.64 ± 1.47	30.35 ± 1.34	51.40 ± 14.66	77.12 ± 7.82	31.57 ± 2.07	34.76 ± 1.11	31.84 ± 1.79	30.61 ± 1.60
Svmguide4	18.14 ± 2.40	17.39 ± 2.16	17.25 ± 2.14	18.99 ± 2.77	17.52 ± 1.97	**17.12 ± 2.47**	18.50 ± 3.10	72.81 ± 4.46	32.48 ± 3.87	24.74 ± 3.47	25.88 ± 4.05	21.57 ± 3.96	21.27 ± 3.29
Protein	36.96 ± 0.89	36.40 ± 0.82	36.24 ± 0.83	36.93 ± 0.67	36.14 ± 0.91	**36.11 ± 0.93**	37.35 ± 0.69	39.10 ± 1.80	49.74 ± 10.26	37.09 ± 0.73	39.58 ± 0.98	36.52 ± 0.80	36.81 ± 0.79
Svmguide2	17.24 ± 2.41	17.96 ± 4.82	**17.24 ± 2.23**	17.70 ± 1.95	17.34 ± 2.24	17.60 ± 2.42	22.20 ± 3.98	45.23 ± 17.81	45.12 ± 11.39	17.39 ± 1.75	30.33 ± 15.72	17.50 ± 2.13	17.44 ± 1.79
DNA	4.93 ± 0.37	4.61 ± 0.49	4.48 ± 0.57 *	4.97 ± 0.46	4.49 ± 0.62	4.66 ± 0.65	5.19 ± 0.50	**4.43 ± 0.54**	4.51 ± 0.61	5.12 ± 0.44	6.84 ± 0.67	5.58 ± 0.49	5.17 ± 0.47
Cardio	17.90 ± 1.09	17.95 ± 0.98 *	-	17.66 ± 0.98	19.78 ± 1.01	-	**17.22 ± 0.62**	25.63 ± 7.57	-	18.58 ± 0.61	29.31 ± 1.66	23.75 ± 0.85	25.15 ± 1.00

Table 4: Error percentages and number of selected classifiers (out of 130) for sparse regularizations with the diverse ensemble setup ($mean \pm standard\ deviation$)$\times 100$. Bold values are the lowest error percentages and number of selected classifiers of sparse regularizations (l_1 or $l_1 - l_2$ regularizations)

DB	Error percentages								Number of Selected Classifiers					SIG
	WS		CWS			LSG			WS	CWS		LSG		
	l_2	l_1	l_2	l_1	$l_1 - l_2$	l_2	l_1	$l_1 - l_2$	l_1	l_1	$l_1 - l_2$	l_1	$l_1 - l_2$	
Segment	5.02 ± 0.88	4.90 ± 0.99	3.90 ± 1.00	3.62 ± 0.62	3.74 ± 0.40	3.60 ± 1.05	3.79 ± 1.05	**3.29 ± 0.55**	**21.50 ± 4.62**	63.50 ± 25.72	30.80 ± 34.92	97.40 ± 24.40	80.40 ± 14.93	
Waveform	13.20 ± 0.69	13.38 ± 0.70	13.05 ± 0.72	13.46 ± 0.74	13.42 ± 0.76	13.05 ± 0.65	13.33 ± 0.71	**13.24 ± 0.64**	36.60 ± 49.44	23.30 ± 37.59	47.00 ± 57.31	**11.20 ± 2.30**	12.10 ± 5.38	bd
Robot	3.95 ± 0.42	4.00 ± 0.38	2.59 ± 0.33	2.57 ± 0.35	**2.49 ± 0.33**	2.61 ± 0.28	2.54 ± 0.35	2.52 ± 0.32	41.80 ± 9.02	18.60 ± 5.97	14.00 ± 4.55	18.50 ± 4.53	**13.30 ± 2.63**	c
Statlog	16.34 ± 1.15	**17.19 ± 1.63**	16.12 ± 1.53	17.45 ± 1.74	17.33 ± 1.42	16.36 ± 1.67	17.40 ± 1.34	17.45 ± 1.51	36.10 ± 34.75	14.30 ± 10.85	49.20 ± 56.13	30.60 ± 36.31	**11.20 ± 12.42**	abd
Vowel	13.84 ± 2.73	14.40 ± 2.27	7.66 ± 2.29	7.62 ± 2.02	7.17 ± 1.50	6.32 ± 1.99	**6.18 ± 1.19**	6.79 ± 1.17	108.90 ± 44.48	37.80 ± 32.62	57.30 ± 62.64	128.00 ± 6.32	**13.80 ± 3.99**	a
Wine	1.57 ± 1.09	2.13 ± 1.63	1.12 ± 1.40	2.25 ± 1.18	**1.91 ± 1.30**	1.69 ± 1.32	2.25 ± 1.59	2.36 ± 1.54	130.00 ± 0.00	121.30 ± 18.60	117.10 ± 40.44	93.50 ± 58.86	**91.60 ± 61.83**	d
Yeast	40.36 ± 1.21	**40.38 ± 1.06**	40.23 ± 1.29	42.40 ± 4.10	41.19 ± 1.57	40.32 ± 1.19	48.09 ± 18.30	41.67 ± 1.31	119.10 ± 34.47	121.00 ± 28.46	40.40 ± 47.33	130.00 ± 0.00	**9.80 ± 3.46**	bd
Steel	29.85 ± 1.86	30.00 ± 2.61	28.27 ± 1.38	28.31 ± 1.39	**27.41 ± 1.21**	27.41 ± 1.22	28.09 ± 1.03	27.50 ± 1.24	41.90 ± 32.05	42.10 ± 6.85	35.30 ± 8.10	51.00 ± 16.62	**35.20 ± 11.93**	bc
Svmguide4	18.14 ± 2.40	18.53 ± 3.39	17.39 ± 2.16	18.79 ± 3.45	**18.07 ± 2.31**	17.25 ± 2.14	23.66 ± 19.44	18.14 ± 2.50	46.00 ± 31.94	40.40 ± 32.61	45.60 ± 46.14	51.90 ± 40.21	**19.90 ± 17.37**	d
Protein	36.96 ± 0.89	36.89 ± 0.76	36.40 ± 0.82	36.66 ± 0.95	**36.32 ± 1.03**	36.24 ± 0.83	38.48 ± 5.03	36.59 ± 1.07	95.20 ± 45.21	23.50 ± 7.71	47.90 ± 57.70	33.60 ± 35.55	**17.30 ± 14.70**	
Svmguide2	17.24 ± 2.41	17.80 ± 2.35	17.96 ± 4.82	19.13 ± 2.60	17.96 ± 4.43	17.24 ± 2.23	19.49 ± 2.63	**17.39 ± 2.00**	70.00 ± 63.28	72.10 ± 61.45	10.30 ± 10.30	69.90 ± 63.97	**6.10 ± 3.18**	e
DNA	4.93 ± 0.37	5.08 ± 0.52	4.61 ± 0.49	4.82 ± 0.47	**4.79 ± 0.61**	4.48 ± 0.57	5.06 ± 0.67	5.13 ± 0.68	97.20 ± 37.51	97.00 ± 33.66	107.60 ± 29.17	80.20 ± 26.83	**65.40 ± 12.47**	ad
Cardio	17.90 ± 1.09	**18.05 ± 1.17**	17.95 ± 0.98	19.89 ± 0.93	18.98 ± 0.56	-	-	-	61.50 ± 12.54	51.50 ± 10.24	35.70 ± 8.30	18.50 ± 4.53	**13.30 ± 2.63**	bc

Table 5: Error percentages and number of selected classifiers (out of 154) for sparse regularizations with the non-diverse ensemble setup ($mean \pm standard\ deviation$)$\times 100$. Bold values are the lowest error percentages and number of selected classifiers of sparse regularizations (l_1 or $l_1 - l_2$ regularizations)

DB	Error percentages									Number of Selected Classifiers					SIG
	WS		CWS			LSG			BC	WS	CWS		LSG		
	l_2	l_1	l_2	l_1	$l_1 - l_2$	l_2	l_1	$l_1 - l_2$		l_1	l_1	$l_1 - l_2$	l_1	$l_1 - l_2$	
Segment	4.48 ± 0.64	4.49 ± 0.71	4.28 ± 0.71	**4.21 ± 0.80**	4.33 ± 0.74	4.42 ± 0.63	9.78 ± 17.11	4.35 ± 0.75	4.28 ± 0.75	29.40 ± 8.13	13.60 ± 6.93	8.40 ± 6.93	51.80 ± 70.80	**2.60 ± 2.67**	
Waveform	13.19 ± 0.71	**13.12 ± 0.81**	13.22 ± 0.78	13.33 ± 0.75	13.24 ± 0.78	13.20 ± 0.73	13.20 ± 0.70	13.25 ± 0.68	13.30 ± 0.80	32.40 ± 64.10	51.60 ± 70.98	95.60 ± 75.54	**5.10 ± 3.18**	36.70 ± 62.37	
Robot	8.02 ± 0.62	7.98 ± 0.70	7.98 ± 0.62	8.13 ± 0.40	**7.94 ± 0.49**	7.99 ± 0.69	8.13 ± 0.55	7.99 ± 0.56	8.54 ± 0.56	43.80 ± 16.19	30.60 ± 10.20	28.30 ± 16.57	20.50 ± 15.71	**13.70 ± 3.40**	ce
Statlog	18.70 ± 2.19	18.87 ± 2.05	18.56 ± 2.05	**18.77 ± 1.74**	19.24 ± 1.81	19.41 ± 1.31	19.17 ± 2.17	19.10 ± 1.56	19.62 ± 1.79	18.90 ± 9.46	14.50 ± 11.98	8.90 ± 9.64	25.30 ± 46.07	**7.80 ± 5.03**	c
Vowel	7.70 ± 2.05	9.88 ± 3.46	7.45 ± 2.23	6.34 ± 2.29	**6.08 ± 2.37**	8.71 ± 2.34	7.72 ± 2.24	6.10 ± 2.30	6.06 ± 2.26	69.20 ± 59.27	8.70 ± 10.12	3.00 ± 2.83	125.70 ± 45.82	**1.40 ± 0.70**	bde
Wine	9.10 ± 2.72	8.88 ± 2.45	10.56 ± 3.86	8.65 ± 3.05	**8.20 ± 3.63**	11.57 ± 3.75	8.65 ± 2.31	8.99 ± 2.95	8.54 ± 3.79	65.00 ± 73.24	39.30 ± 60.96	**21.90 ± 46.09**	34.80 ± 62.84	78.90 ± 79.19	bd

Table 6: Error percentages and number of selected classifiers (out of 154) for sparse regularizations with the Random Subspace ensemble setup (*mean* ± *standard deviation*)×100. Bold values are the lowest error percentages and number of selected classifiers of sparse regularizations (l_1 or $l_1 - l_2$ regularizations)

DB	Error percentages								Number of Selected Classifiers					SIG
	WS		CWS			LSG			WS	CWS		LSG		
	l_2	l_1	l_2	l_1	$l_1 - l_2$	l_2	l_1	$l_1 - l_2$	l_1	l_1	$l_1 - l_2$	l_1	$l_1 - l_2$	
Waveform	13.93 ± 0.72	14.02 ± 0.89	13.89 ± 0.77	14.01 ± 0.86	**13.82 ± 0.74**	-	-	-	25.50 ± 18.72	27.70 ± 2.91	**20.60 ± 3.72**	-	-	
Robot	2.49 ± 0.71	2.61 ± 0.77	2.14 ± 0.63	2.13 ± 0.70	**2.09 ± 0.69**	-	-	-	51.90 ± 13.84	18.60 ± 8.58	**15.80 ± 9.75**	-	-	
Statlog	20.80 ± 3.08	21.77 ± 2.98	19.31 ± 2.52	**19.36 ± 2.66**	19.72 ± 3.23	19.27 ± 3.00	19.48 ± 2.87	19.50 ± 2.97	48.80 ± 46.39	24.80 ± 14.06	**17.00 ± 9.63**	35.50 ± 31.66	20.30 ± 19.82	
Vowel	12.77 ± 3.01	13.58 ± 3.06	6.08 ± 2.19	7.72 ± 1.94	**6.22 ± 1.47**	6.16 ± 1.85	6.75 ± 2.00	6.30 ± 1.35	114.00 ± 34.37	22.70 ± 3.43	12.70 ± 7.29	92.90 ± 44.16	**10.80 ± 5.03**	c
Wine	2.58 ± 1.30	3.37 ± 1.67	2.13 ± 1.24	7.42 ± 5.03	**2.13 ± 1.35**	2.25 ± 1.67	3.03 ± 1.50	2.92 ± 2.44	70.90 ± 62.35	82.10 ± 61.89	84.20 ± 59.65	**14.40 ± 5.34**	17.30 ± 15.76	c
Steel	28.22 ± 1.69	28.16 ± 1.47	26.66 ± 1.20	27.23 ± 1.41	**27.03 ± 1.83**	27.17 ± 1.47	27.38 ± 1.17	27.99 ± 1.35	48.10 ± 17.14	38.60 ± 6.82	30.80 ± 13.78	50.70 ± 29.45	**22.00 ± 18.57**	
Svmguide4	20.85 ± 3.53	21.01 ± 4.01	18.76 ± 3.02	19.15 ± 3.97	**18.46 ± 3.87**	18.95 ± 3.00	20.03 ± 3.32	22.25 ± 3.95	47.00 ± 8.52	32.10 ± 7.13	27.40 ± 8.91	51.80 ± 29.23	**9.40 ± 4.50**	d
protein	37.64 ± 0.83	37.81 ± 0.90	37.02 ± 0.77	37.01 ± 0.80	**36.85 ± 0.72**	-	-	-	73.30 ± 39.80	37.80 ± 11.11	**32.70 ± 7.36**	-	-	
svmguide2	17.95 ± 3.12	**17.90 ± 2.46**	17.65 ± 2.28	18.98 ± 2.30	19.74 ± 2.38	17.08 ± 2.15	20.05 ± 2.96	19.33 ± 3.05	81.90 ± 62.11	18.20 ± 12.94	13.40 ± 11.85	12.20 ± 11.28	**10.10 ± 3.84**	bd
DNA	5.27 ± 0.66	5.38 ± 0.63	4.77 ± 0.66	**4.85 ± 0.42**	4.88 ± 0.54	-	-	-	**26.90 ± 9.72**	34.80 ± 5.41	29.30 ± 2.95	-	-	