

Support Vector Machine as Feature Selection Method in Classifier Ensembles

Jasmina Đ. Novakovic

Belgrade Business School, Belgrade, 11000, Serbia

Email: jasmina.novakovic@bbs.edu.rs

Abstract—In this paper, we suggest classifier ensembles that can incorporate Support Vector Machine (SVM) as feature selection method into classifier ensembles models. Consequences of choosing different number of features are monitored. Also, the goal of this research is to present and compare different algorithmic approaches for constructing and evaluating systems that learn from experience to make the decisions and predictions and minimize the expected number or proportion of mistakes. Experimental results demonstrate the effectiveness of selecting features with SVM in various types of classifier ensembles.

Index Terms—Classification accuracy, feature selection, classifier ensembles, machine learning, Support Vector Machine.

I. INTRODUCTION

A process that chooses a minimum subset of M features from the original set of N features, so that the feature space is optimally reduced according to a certain evaluation criterion can be defined as feature selection. Finding the best feature subset is usually intractable and many problems related to feature selection have been shown to be NP-hard.

Feature selection is an active field in computer science and it has been a fertile field of research and development since 1970's in statistical pattern recognition machine learning and data mining. It is a fundamental problem in many different areas, especially in forecasting, document classification, bioinformatics, and object recognition or in modelling of complex technological processes. In such applications, datasets with thousands of features are not uncommon and for some problems all features may be important, but for some target concepts, only a small subset of features is usually relevant.

Various aspects of feature selection have been studied. Search is a key topic in the study of feature selection [1] such as search starting points, search directions, and search strategies. Another important aspect is how to measure the goodness of a feature subset [2]. Algorithms for feature selection may be divided into filters [2, 3], wrappers and embedded approaches [4]. Filters methods evaluate quality of selected features, independently from the classification algorithm, wrapper methods require application of a classifier to evaluate this quality, while embedded methods perform feature selection during

learning of optimal parameters. According to class information availability in data, there are supervised feature selection approaches as well as unsupervised feature selection approaches.

Some classification algorithms have inherited ability to focus on relevant features and ignore irrelevant ones. Decision trees are primary example of a class of such algorithms, but also multi-layer perceptron neural networks, with strong regularization of the input layer, may exclude the irrelevant features in an automatic way [5]. Such methods may also benefit from independent feature selection. On the other hand, some algorithms have no provisions for feature selection. The k -nearest neighbour algorithm is one family of such methods that classify novel examples by retrieving the nearest training example, strongly relying on feature selection methods to remove noisy features.

The main aim of this paper was to experimentally verify the impact of SVM as feature selection method on classification accuracy with classifier ensembles. We use classifier ensembles, instead of individual classifier, because in many fields, multiple classifier system is more accurate and robust than an excellent single classifier.

In this study, we suggest classifier ensembles that can incorporate SVM as feature selection method into classifier ensembles models. The goal of this research is also to present and compare different algorithmic approaches for constructing and evaluating systems that learn from experience to make the decisions and predictions and minimize the expected number or proportion of mistakes.

The paper is organized as follows. In the next section we briefly described general issues concerning SVM. Section 3 gives a brief overview of adopted algorithms, namely, Bagging, AdaBoost, Rotation Forest, Dagging, Decorate, MultiBoost and LogitBoost. Section 4 discusses the results and investigates the performance of the proposed technique. Finally, concluding remarks are discussed in section 5.

II. SUPPORT VECTOR MACHINE

Various feature ranking and feature selection techniques have been proposed in the machine learning literature, which purpose is to discard irrelevant or redundant features from a given feature vector. In this paper, we consider evaluation of the practical usefulness of SVM as feature selection technique on classification

accuracy with classifier ensembles, as filter method which evaluate each subset.

SVM introduced by Vapnik [6], employs structural risk minimization whereby a bound on the risk is minimized by maximizing the margin between the separating hyperplane and the closest data point to the hyperplane. SVM as supervised learning methods that analyze data and recognize patterns, rigorously based on statistical learning theory simultaneously minimizes the training and test errors.

In 1963 Vapnik proposed original optimal hyperplane algorithm, which was a linear classifier, but often happens that in a finite dimensional space the sets to be discriminated are not linearly separable. In 1992, Boser, Guyon and Vapnik [7] suggested a way to create nonlinear classifiers by applying the kernel trick (originally proposed by Aizerman, Braverman and Rozonoer [8]) to maximum-margin hyperplanes. In this algorithm, every dot product is replaced by a nonlinear kernel function, which allows the algorithm to fit the maximum-margin hyperplane in a transformed feature space. To make the separation easier, it was proposed that the original finite-dimensional space be mapped into a much higher-dimensional space. Mapping into a larger space, cross products may be computed easily in terms of the variables in the original space, making the computational load reasonable.

SVM constructs a hyperplane or set of hyperplanes in a high dimensional space, which can be used for classification, regression, or other tasks. Many hyperplanes might classify the data; the best hyperplane is the one that represents the largest separation, or margin, between the classes. Generally, the larger the margin it is the lower the generalization error of the classifier. We choose the maximum-margin hyperplane, such the hyperplane in which the distance from it to the nearest data point on each side is maximized.

SVM is explained below. Given training vectors $x_i \in \mathbb{R}^n, i = 1, \dots, l$, in the two-class case and the corresponding class labels decision $y_i \in \{1, -1\}$, the statement of SVM optimization for classification problems may be the following [9, 10]:

$$\min_{w, b, \xi} \frac{1}{2} w^T w + C \sum_{i=1}^l \xi_i \quad (1)$$

with constraints: $y_i(w^T \phi(x_i) + b) \geq 1 - \xi_i$, $\xi_i \geq 0, i = 1, \dots, l$. The dual problem definition is:

$$\min_{\alpha} \frac{1}{2} \alpha^T Q \alpha - e^T \alpha, 0 \leq \alpha_i \leq C, i = 1, \dots, l, \quad (2)$$

with constraints $y^T \alpha = 0$, where e is the vector of all ones, $C > 0$ is the upper bound, Q is a l by l positive semi definite matrix, $Q_{ij} \equiv y_i y_j K(x_i, x_j)$, and $K(x_i, x_j) \equiv \phi(x_i)^T \phi(x_j)$ is the kernel. Function ϕ transforms training vectors x_i into a higher (maybe infinite) dimensional space. The decision function is

$$\text{sgn}\left(\sum_{i=1}^l y_i \alpha_i K(x_i, x) + b\right) \quad (3)$$

The choice of kernel results in different kinds of SVM with different performance levels. The choice of the appropriate kernel for a specific application is often a difficult task. A necessary and sufficient condition for a kernel to be valid is that it must satisfy Mercer's theorem, but other than that, there is really no mathematically structured approach to prefer one kernel to the other.

III. CLASSIFIER ENSEMBLES

In machine learning and pattern recognition, in many fields multiple classifier system is more accurate and robust than an excellent single classifier, because one single classification system cannot always provide high classification accuracy. Classifier combination is an active field of research for the reason that a lot of theoretical and practical studies present the advantages of the combination paradigm over the individual classifier models. A great deal of study has gone into designing multiple classifier systems that are commonly called classifier ensembles. Seven approaches for constructing classifier ensembles are presented, which have been found to be accurate and computationally feasible across various data domains.

A. Bagging

The Bagging predictor firstly was introduced in 1996 by Breiman [11], who shows in his work that the Bagging predictors can push an unstable weak classifier to a significant step towards optimality. Bagging [11] takes bootstrap samples of objects and trains a classifier on each sample. The classifier votes are combined by majority voting. In some implementations of Bagging, classifiers produce estimates of the posterior probabilities for the classes. Posterior probabilities are averaged across the classifiers and the most probable class is assigned, called "average" aggregation of the outputs.

For $k = 1, 2, \dots, K$ Do
 Draw $N_{(B)}$ samples randomly and with replacement from the X .
 And form the learning set $L_{(B)}$.
 Train classification $\phi_k(L_{(B)})$ from the learning set $L_{(B)}$.
 Make a plurality vote for every ϕ_k of K classifications on the test set.
 The classification, which gets the most voting score, is the best one.

Fig.1. Bagging algorithm [12]

Bagging with average aggregation is used in this experiment. Each individual classifier is trained on a bootstrap sample, which causes the data distribution seen during training is similar to the original distribution. Therefore, the individual classifiers in a bagging ensemble have relatively high classification accuracy.

Only the proportion of different objects in the training samples encourages diversity between these classifiers. The bootstrap sampling appears to lead to ensembles of low diversity compared to other ensemble creating methods, even though the classifier models used in Bagging are sensitive to small changes in data. For that reason, Bagging requires larger ensemble sizes to perform well.

Bagging algorithm is showed in Fig. 1. A learning set consists of data $L := \{(y_n, x_n), n = 1, 2, \dots, N\}$, where $x_n \in X$ is the input and $y_n \in Y$ is the response from a predictor denoted by $\varphi(X, L)$. Suppose we are given k learning sets $\{L_k\}$ of N independent identical distribution observation from the same distribution of L . An obvious procedure to use the average to get a better estimate of the classification:

$$\varphi_A(X) = E_L \varphi(X, L) \quad (4)$$

where E_L denotes the expectation over L . Generally, we only have one learning set L , so, one resolvable way is to take bootstrap samples $\{L_{(B)}\}$ of $N_{(B)}$ cases, randomly, but with replacement from L , and form predictor set $\{\varphi(X, L_{(B)})\}$. This procedure is called “bootstrap aggregating” and use the acronym Bagging. In that case, the predictor $\bar{\varphi}$, which gets most voting score of test set, is the best classification.

B. AdaBoost

Boosting is a family of methods for improving the performance of a “weak” classifier by using it within an ensemble structure, the most prominent member of which is AdaBoost. In Boosting methods, a set of weights is maintained across the objects in the data set, so that objects that have been difficult to classify acquire more weight, forcing subsequent classifiers to focus on them. Mechanism of these methods consist of repeatedly running a given weak learning algorithm on various distributions over the training data, and then combining the classifiers produced by the weak learner into a single composite classifier.

The Boosting algorithm takes as input a training set of m examples $S = \{(x_1, y_1), \dots, (x_m, y_m)\}$ where x_i is an instance drawn from some space X , and $y_i \in Y$ is the class label associated with x_i . In this research, is assumed that the set of possible labels Y is of finite cardinality k . The Boosting algorithm calls weak learning algorithm repeatedly in a series of rounds. On round t , the booster provides weak learning algorithm with a distribution D_t over the training set S . Weak learning algorithm computes a classifier or hypothesis $h_t: X \rightarrow Y$, which should misclassify a non trivial fraction of the training examples, relative to D_t . The goal of weak learner is to find a hypothesis h_t that minimizes the training error $\epsilon_t = \Pr_{i \sim D_t}[h_t(x_i) \neq y_i]$. Training error is measured with respect to the distribution D_t that was provided to the weak learner. This process continues for

T rounds. At last, the booster combines the weak hypotheses h_1, \dots, h_T into a single final hypothesis h_{fin} . In the Boosting algorithm the manner in which D_t is computed on each round, and how h_{fin} is computed are unspecified and these questions solve different Boosting schemes in different ways.

AdaBoost.M1 algorithm uses the simple rule present in Fig. 2, where the initial distribution D_1 is uniform over S so $D_1(i) = 1/m$ for all i . In this algorithm to update distribution, the weight of example i is multiplied by some number $\beta_t \in [0, 1]$ if h_t classifies x_i correctly, and otherwise the weight is left unchanged, and also divide by the normalization constant Z_t . Thus, “hard” examples, which tend often to be misclassified, get higher weight, and “easy” examples that are correctly classified by many of the previous weak hypotheses get lower weight. Accordingly, AdaBoost.M1 focuses the most weight on the examples that seem to be hardest for weak learning algorithm.

Input: sequence of m examples $\{(x_1, y_1), \dots, (x_m, y_m)\}$
with labels $y_i \in Y = \{1, \dots, k\}$
weak learning algorithm ψ
integer T specifying number of iterations ψ

Initialize: $D_1(i) = 1/m$ for all i . ψ

Do for $t = 1, 2, \dots, T$ ψ

1. ψ Call weak learning algorithm, providing it with the distribution D_t . ψ
2. ψ Get back hypothesis $h_t: X \rightarrow Y$. ψ
3. ψ Calculate the error of $h_t: \epsilon_t = \sum_{i: h_t(x_i) \neq y_i} D_t(i)$.
If $\epsilon_t > 1/2$, then set $T = t - 1$ and abort loop. ψ
4. ψ Set $\beta_t = \epsilon_t / (1 - \epsilon_t)$. ψ
5. ψ Update distribution $D_t: D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times$
 $\begin{cases} \beta_t & \text{if } h_t(x_i) = y_i \\ 1 & \text{otherwise} \end{cases}$
where Z_t is a normalization constant (chosen so that D_{t+1} will be a distribution). ψ

Output the final hypothesis:
 $h_{fin}(x) = \arg \max_{y} \sum_{t: h_t(x) = y} \log \frac{1}{\beta_t}$

Fig.2. AdaBoost.M1 algorithm [13]

The number β_t is a function of ϵ_t and the final hypothesis h_{fin} is a weighted vote of the weak hypotheses. The weight of hypothesis h_t is defined to be $\ln 1/\beta_t$ so that greater weight is given to hypotheses with lower error.

The success of AdaBoost algorithm has been explained, among others, with its diversity creating ability, which is an important property of a classifier ensemble [14]. This algorithm creates inaccurate classifiers by forcing them to concentrate on difficult objects and ignore the rest of the data, which led to large diversity that boosted the ensemble performance, often beyond that of Bagging. This leads us to the famous accuracy-diversity dilemma, because it seems that classifiers cannot be both very accurate and have very diverse outputs.

C. Rotation Forest

Rotation Forest is an ensemble method that trains classifiers independently, using a different set of extracted features for each of classifier, which is accurate and feasible across various data domains [15].

Rotation Forest randomly split the feature set into K subsets, which a base classifier uses as training data, and chose disjoint subsets to maximize the chance for high diversity. The filter used to project the data applied to each subset. Running the filter on a subset of classes instead of the whole set is done in a bid to avoid identical coefficients if the same feature subset is chosen for different classifiers.

Details about Rotation Forest algorithm are presented below. Assume that there are N training samples and n features in data set. Let X be the training sample set in a form of a $N \times n$ matrix, Y be the corresponding labels, where Y takes values from the set of class labels $\{w_1, \dots, w_c\}$, and F be the feature set. Assuming there are L decision trees in Rotation Forest, denoted by D1, ..., DL respectively, and the feature set split randomly into K subsets, we need to determine L and K in advance.

Training Phase⁺

Given⁺

- X: the object in the training data set (an $N \times n$ matrix)⁺
- Y: the labels of the training set (an $N \times 1$ matrix)⁺
- L: the number of classifiers in the ensemble⁺
- K: the number of subsets⁺
- $\{w_1, \dots, w_c\}$: the set of class labels⁺

For $i = 1 \dots L$ ⁺

- Prepare the rotation matrix R_i^a ⁺
 - Split F (the feature set) into K subsets: $F_{i,j}$ (for $j = 1 \dots K$)⁺
 - For $j = 1 \dots K$ ⁺
 - Let $X_{i,j}$ be the data set X for the features in $F_{i,j}$ ⁺
 - Eliminate from $X_{i,j}$ a random subset of classes⁺
 - Select a bootstrap sample from $X_{i,j}$ of size 75% of the number of objects in $X_{i,j}$. Denote the new set by $X'_{i,j}$ ⁺
 - Apply PCA on $X'_{i,j}$ to obtain the coefficients in a matrix $C_{i,j}$ ⁺
 - Arrange the $C_{i,j}$ for $j = 1 \dots K$ in a rotation matrix R_i ⁺
 - Construct R_i^a by rearranging the columns of R_i to match the order of features in F.⁺
- Build classifier D_i using (XR_i^a, Y) as the training set⁺

Classification Phase⁺

- For a given x, let $d_{i,j}(xR_i^a)$ be the probability produced by the classifier D_i to the hypothesis that x belongs to class w_j . By the average combination method, calculate the confidence for each class as follows:⁺

$$\mu_j(x) = \frac{1}{L} \sum_{i=1}^L d_{i,j}(xR_i^a), j = 1, \dots, c \quad +$$

The test sample x is easily assigned to the class with the largest confidence.⁺

Fig.3. Rotation Forest algorithm [15]

Construction of the training set for an individual classifier D_i is in three steps. In step1, split F randomly into K disjointed subsets. Suppose that K is a factor of n so that each feature subset contains $M = n/K$ features. In step 2, denote by $F_{i,j}$ the jth subset of features for training set of classifier D_i . Select randomly for each subset, a

nonempty subset of classes, and then draw a bootstrap subset of objects with the size of 75 percent of the data set to form a new training set. After that, PCA is applied to the M features in $F_{i,j}$ and the selected subset of X.

Denoted the coefficients of PCA by $a_{i,j}^{(1)}, \dots, a_{i,j}^{(M)}$ each of size $M \times 1$. In step 3, arrange a sparse rotation matrix R_i with the obtained coefficients. The columns of R_i should be rearranged according to the original features and the rearranged rotation matrix is denoted by R_i^a . The transformed training set for classifier D_i is XR_i^a . All classifiers will be trained in parallel. In the classification phase of Rotation Forest, for a given x, let $d_{i,j}(xR_i^a)$ be the probability produced by the classifier D_i to the hypothesis that x belongs to class w_j . By the average combination method, calculate the confidence for each class as follows:

$$\mu_j(x) = \frac{1}{L} \sum_{i=1}^L d_{i,j}(xR_i^a), j = 1, \dots, c \quad (5)$$

The test sample x is easily assigned to the class with the largest confidence. The algorithm of Rotation Forest is showed on Fig. 3.

D. Dagging

Ting and Witten [16] introduced Dagging as ensemble method, which is very similar to Bagging, but instead of using bootstrap sampling it uses disjoint samples. The training set is partitioned into K subsets, and a base classifier generates a hypothesis for each subset. The final prediction is done with plurality vote as in Bagging. A further difference is Dagging uses no extra resources, since the same amount of examples are used as the training set. This ensemble method is useful for base classifiers that are quadratic or worse in time behavior, regarding number of instances in the training data.

In his research Ting and Witten were tested Bagging and Dagging with C4.5, considered unstable classifier, and Naive Bayes, considered stable classifier. Although Breiman reports that Bagging doesn't generally work well on stable classifiers, Ting and Witten conclude that Bagging as well as Dagging, do well on both unstable and stable algorithms.

E. Decorate

Decorate directly constructs diverse hypotheses using additional artificially-constructed training examples [17]. The diversity of an ensemble of classifiers is known to be an important factor in determining its generalization error. This classifier ensemble can use any strong learner (one that provides high accuracy on the training data) as a base classifier to build diverse committees.

In Decorate (Fig. 4), an ensemble is generated iteratively, first learning a classifier and then adding it to the current ensemble. The ensemble is initialized to contain the classifier trained on the given training data. In successive iteration, the classifiers are trained on the

original training data combined with some artificial data. In iteration, artificial training examples are generated from the data distribution; where the number of examples to be generated is specified as a fraction, R_{size} , of the training set size. The labels for these artificially generated training examples are chosen so as to differ maximally from the current ensemble's predictions. A new classifier is trained on the union of the original training data and the diversity data, thereby forcing it to differ from the current ensemble. As a result, adding this classifier to the ensemble should increase its diversity. To maintain training accuracy, a new classifier is rejected if adding it to the existing ensemble decreases its accuracy. Decorate is repeated this process until the desired committee size or exceed the maximum number of iterations are reached.

Input:
BaseLearn - base learning algorithm
 T - set of m training examples $\langle (x_1, y_1), \dots, (x_m, y_m) \rangle$ with labels $y_j \in Y$
 C_{size} - desired ensemble size
 I_{max} - maximum number of iterations to build an ensemble
 R_{size} - factor that determines number of artificial examples to generate
 $i = 1$
 $trials = 1$
 $C_i = BaseLearn(T)$
 Initialize ensemble, $C^* = \{C_i\}$
 Compute ensemble error, $\epsilon = \frac{\sum_{x_j \in T, C^*(x_j) \neq y_j} 1}{m}$
 While $i < C_{size}$ and $trials < I_{max}$
 Generate $R_{size} \times |T|$ training examples, R , based on distribution of training data
 Label examples in R with probability of class labels inversely proportional to predictions of C^*
 $T = T \cup R \rightarrow$
 $C' = BaseLearn(T)$
 $C^* = C^* \cup \{C'\}$
 $T = T - R$, remove the artificial data
 Compute training error, ϵ' , of C^*
 If $\epsilon' \leq \epsilon$
 $i = i + 1$
 $\epsilon = \epsilon'$
 otherwise,
 $C^* = C^* - \{C'\}$
 $trials = trials + 1$

Fig.4. Decorate algorithm [17]

F. MultiBoost

MultiBoost is technique for combining Boosting and Wagging (which is in turn a variant of Bagging). Bagging and AdaBoost appear to operate by diverse mechanisms, have different effects, and both have greatest effect obtained from the first few committee members, which suggest that it might be possible to obtain benefit by combining them. Because the mechanisms are differing, their combination may out-perform either in isolation. AdaBoost reduces both bias and variance and Bagging mainly reduces variance, but more effective than AdaBoost at reducing variance. It is reason why their

combination may be able to retain AdaBoost's bias reduction while adding Bagging's variance reduction to that already obtained by AdaBoost.

The resulting MultiBoost algorithm is presented in Fig. 5. This algorithm besides the bias and variance reduction properties that may inherit from each of its constituent committee learning algorithms, has the potential computational advantage over AdaBoost that the sub-committee may be learned in parallel, although this would require a change to the handling of early termination of learning a sub-committee. The AdaBoost process is inherently sequential, minimizing the potential for parallel computation, but each classifier learned with Wagging is independent of the rest, allowing parallel computation, a property that MultiBoost inherits at the sub-committee level.

Input:
 S : a sequence of m labeled examples $\langle (x_1, y_1), \dots, (x_m, y_m) \rangle$ with labels $y_i \in Y$
 base learning algorithm *BaseLearn*
 integer T specifying the number of iterations
 vector of integers I_i specifying the iteration at which each subcommittee $i \geq 1$ should terminate
 $S' = S$ with instance weights assigned to be 1
 set $k = 1$
 For $t = 1$ to T
 If $I_k = t$ then
 reset S' to random weights drawn from the continuous Poisson distribution
 standardize S' to sum to n
 increment k
 $C_t = BaseLearn(S')$ (Step 8)
 $\epsilon_t = \frac{\sum_{j \in S', C_t(x_j) \neq y_j} weight(x_j)}{m}$ [the weighted error on the training set]
 if $\epsilon_t > 0.5$ then
 reset S' to random weights drawn from the continuous Poisson distribution
 standardize S' to sum to n
 increment k
 go to Step 8
 otherwise if $\epsilon_t = 0$ then
 set β_t to 10^{-10}
 reset S' to random weights drawn from the continuous Poisson distribution
 standardize S' to sum to n
 increment k
 otherwise,
 $\beta_t = \frac{\epsilon_t}{1 - \epsilon_t}$
 For each $x_j \in S'$
 divide $weight(x_j)$ by $2\epsilon_t$ if $C_t(x_j) \neq y_j$ and $2(1 - \epsilon_t)$ otherwise
 if $weight(x_j) < 10^{-8}$, set $weight(x_j)$ to 10^{-8}
 }
Output the final classifier:
 $C^*(x) = \underset{y \in Y}{argmax} \sum_{t: C_t(x) = y} \log \frac{1}{\beta_t}$

Fig.5. MultiBoost algorithm [18]

G. LogitBoost

LogitBoost performs classification using a regression scheme as the base learner, and can handle multi-class problems (Fig. 6). This classifier ensemble can do efficient internal cross-validation to determine appropriate number of iterations.

Start with weights:
 $\omega_{ij} = 1/N, i = 1, \dots, N, j = 1, \dots, J, F_j(x) = 0$ and
 $p_j(x) = 1/J \forall j.$
 Repeat for $m = 1, 2, \dots, M:$
 (a) Repeat for $j = 1, \dots, J:$
 (i) Compute working responses and weights in the j th class,⁴

$$z_{ij} = \frac{y_{ij} - p_j(x_i)}{p_j(x_i)(1 - p_j(x_i))},$$

$$\omega_{ij} = p_j(x_i)(1 - p_j(x_i)).$$
 (ii) Fit the function $f_{mj}(x)$ by a weighted least-squares regression of z_{ij} to x_i with weights $\omega_{ij}.$
 (b) Set $f_{mj}(x) \leftarrow \frac{j-1}{j} f_{mj}(x) - \frac{1}{j} \sum_{k=1}^J f_{mk}(x),$ and
 $F_j(x) \leftarrow F_j(x) + f_{mj}(x).$
 (c) Update $p_j(x).$
 Output the classifier $\arg \max_j F_j(x).$

Fig.6. The LogitBoost (J classes) algorithm [19]

IV. OUTCOMES OF UNDERTAKING COURSEWORK

In this section, we will investigate the impact of data dimensionality reduction with SVM on classification accuracy with classifier ensembles. Consequences of choosing different number of features are monitored. Later on, comparisons of results of measuring the performance of classifiers are presented. For the purpose of this research, SVM classifier evaluates the worth of each feature. Features are ranked by the square of the weight assigned by the SVM [20]. The WEKA toolkit is used to analyze the dataset [21, 22].

Five real data sets in medical domains were used for tests, taken from the UCI repository of machine learning databases [23]. We used these data sets to compare results of classification with data dimensionality reduction by SVM in medical diagnosis.

To achieve the goal of classifier ensemble to produce a model (based on the training data) which predicts the target values of the test data given only the test data features; the following procedure is used. It consists of transform data to the appropriate format, conduct simple scaling on the data, use linear kernel $K(x_i, x_j) = x_i^T x_j$ for feature selection, and finally use one of the classifier ensembles for classification. We set classifier ensembles in following way:

- AdaBoost.M1 algorithm is used, which use the base classifier DecisionStump and reweighting, the number of iterations is set on 10, and weight threshold for weight pruning is set on 100.
- Bagging use the base classifier REPTree, size of each bag, as a percentage of the training set size is

set on 100, the out-of-bag error is not calculated, the number of iterations is set on 10.

- Rotation Forest use the base classifier J48, the number of iterations is set on 10, the filter used to project the data is set on Principal Components, and the percentage of instances to be removed is set on 50.
- Dagging use the base classifier SMO (John Platt's sequential minimal optimization algorithm for training a support vector classifier) and the number of folds to use for splitting the training set into smaller chunks for the base classifier is set on 10.
- Decorate use the base classifier J48, the desired number of member classifiers in the Decorate ensemble is set on 10 (larger ensemble sizes usually lead to more accurate models, but increases training time and model complexity) and the maximum number of Decorate iterations to run is set on 10.
- MultiBoostAB use the base classifier Decision Stump and reweighting, the number of iterations is set on 10, the number of subcommittees is set on 3, and weight threshold for weight pruning is set on 100.
- LogitBoost use the base classifier Decision Stump and reweighting, threshold on improvement in likelihood is set on -1.798E308, no cross-validation is performed, and the number of iterations is set on 10.

The experiment was first unleashed the ranking of importance of each feature in the data sets. Dimensionality reduction was done by reduced number of features in each data set; taking into account that the feature which has the lowest significance is first rejected. Then, the classification accuracy is measured by applying different classifier ensembles. Classifier ensembles were used for the good performance shown by the preliminary study, the high classification accuracy and high speed operation. After that, was analyzed the effect of reducing the number of features on the accuracy of each classifiers.

Our implementation is as follows. The training data is separated into 10 subsets of equal size in 10-fold cross-validation. Sequentially one fold is considered as the validation set and the rest are for training. The cross validation accuracy is the average of accuracy on predicting the validation sets. Results of classification accuracy, as a method for measuring the performance of SVM for five data sets in medical domains, are presented on Fig. 7, Fig. 8, Fig. 9, Fig. 10 and Fig. 11. Selecting appropriate features for a given data set, the reliability of classification for most of data sets and classifier ensembles is increased.

For Pima data set, with all classifier ensembles algorithms, using the SVM for feature selection, we get at least the same or higher classification accuracy. In most cases, five of seven cases, the greatest classification accuracy were achieved with five features, instead of the eight features as it was in the original data set. For Pima data set, maximum improvement of the classification accuracy with SVM as feature selection method,

compared to the classification accuracy without selection of features is as follows: AdaBoost is 1.56%, Bagging is 0.52%, Rotation Forest is 1.17%, Dagging without improvement, Decorate is 2.6%, MultiBoostAB without improvement and LogitBoost is 1.04%.

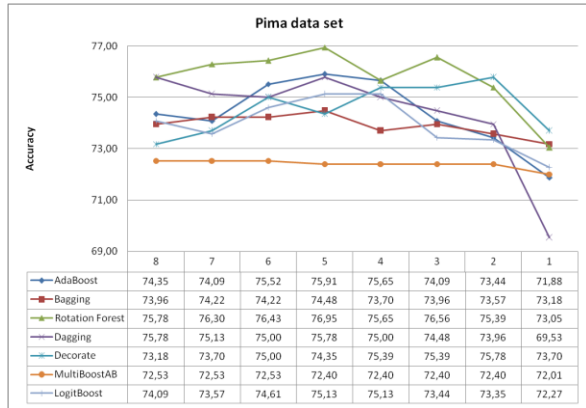


Fig.7. Pima data set and impact of selected features on classification accuracy

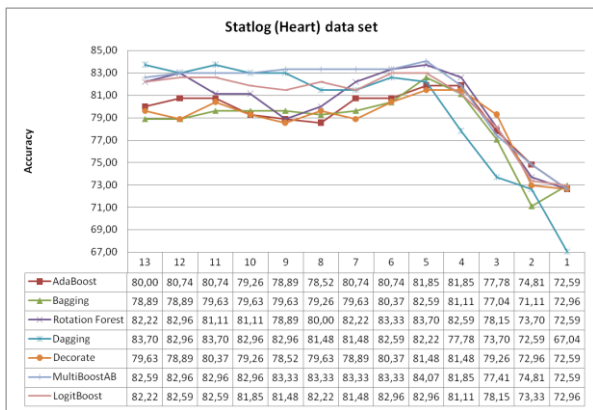


Fig.8. Statlog(Heart) data set and impact of selected features on classification accuracy

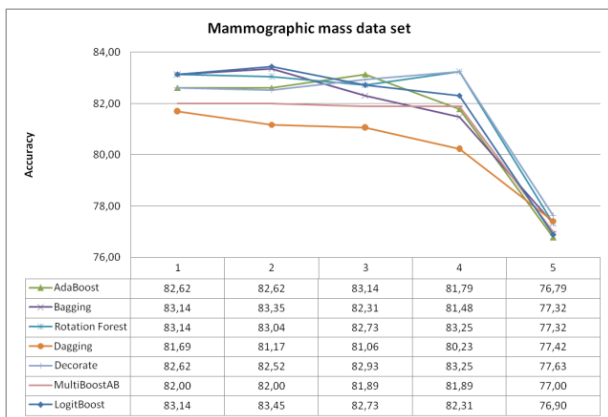


Fig. 9. Mammographic mass data set and impact of selected features on classification accuracy

In the case of Statlog (Heart) data set, for all algorithms, using the SVM for feature selection method, we get at least the same or higher classification accuracy. In all cases except in one case, the highest classification accuracy was achieved with five features, instead of the thirteen features as it was in the original data set. For

Statlog (Heart) data set, maximum improvement of the classification accuracy with SVM as feature selection method, compared to the classification accuracy without selection of features is as follows: AdaBoost is 1.85%, Bagging is 3.7%, Rotation Forest is 1.48%, Dagging without improvement, Decorate is 1.85%, MultiBoostAB is 1.48% and LogitBoost is 0.74%.

In the case of Mammographic mass data set, in all algorithms, using the SVM for feature selection method, we get at least the same or higher classification accuracy, except for Dagging classifier ensembles which leads to less accuracy of classification. In three cases of seven cases, classification accuracy was achieved with four features, instead of the five features as it was in the original data set. For Mammographic mass data set, maximum improvement of the classification accuracy with SVM as feature selection method, compared to the classification accuracy without selection of features is as follows: AdaBoost is 0.52%, Bagging is 0.21%, Rotation Forest is 0.11%, Dagging without improvement, Decorate is 0.63%, MultiBoostAB without improvement and LogitBoost is 0.31%.



Fig.10. Liver data set and impact of selected features on classification accuracy

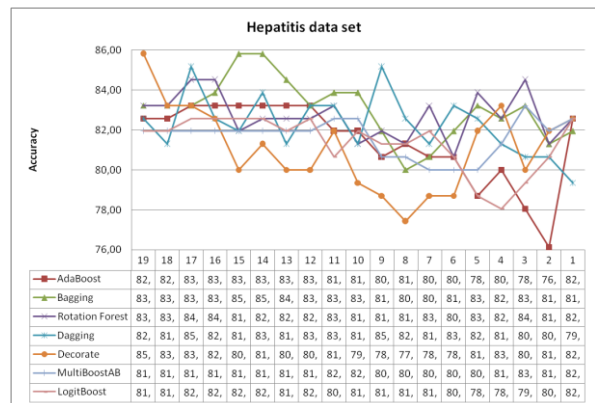


Fig.11. Hepatitis data set and impact of selected features on classification accuracy

For Liver data set, in all algorithms, using the SVM for feature selection method, we get at least the same or higher classification accuracy, except for Bagging, Rotation Forest and Decorate classifier ensembles. In three cases of seven cases, the classification accuracy was

achieved with five features, instead of the six features as it was in the original data set. For Liver data set, maximum improvement of the classification accuracy with SVM as feature selection, compared to the classification accuracy without selection of features is as follows: AdaBoost is 3.19%, Bagging without improvement, Rotation Forest without improvement, Dagging without improvement, Decorate without improvement, MultiBoostAB is 1.74% and LogitBoost is 2.61%.

For Hepatitis data set, for all algorithms, using the SVM for feature selection method, we get at least the same or higher classification accuracy, except for Decorate classifier ensembles. The largest classification accuracy was achieved with different numbers of features. With the decrease of the number of features, the classification accuracy of some algorithms oscillates around some value. For this data set, with a very small number of selected features, high classification accuracy was achieved. For Hepatitis data set, maximum improvement of the classification accuracy with SVM as feature selection method, compared to the classification accuracy without selection of features is as follows: AdaBoost is 0.65%, Bagging is 2.58%, Rotation Forest is 1.29, Dagging is 2.58%, Decorate without improvement, MultiBoostAB is 1.29% and LogitBoost is 0.64%.

V. CONCLUSION

This paper investigates the impact of feature selection with SVM on classification accuracy with classifier ensembles. Classifier ensembles with SVM as feature selection method has used on five medical data sets. These results evaluated and compared choosing different number of selected features. Experimental results demonstrate the effectiveness of selecting features with SVM in various types of classifier ensembles.

REFERENCES

- [1] Doak J. An evaluation of feature selection methods and their application to computer security. Technical report, Davis CA: University of California, Department of Computer Science, 1992.
- [2] Almuallim H, Dietterich T G. Learning with many irrelevant features. In: Proc. AAAI-91, Anaheim, CA, 1991, 547-552.
- [3] Kira K, Rendell L A. The feature selection problem: traditional methods and a new algorithm. In: Proc. AAAI-92, San Jose, CA, 1992, 122-126.
- [4] Blum A I, Langley P. Selection of relevant features and examples in machine learning. *Artificial Intelligence*, vol. 97, 1997, 245-271.
- [5] Duch W, Adamczak R, Grabczewski, K. A new methodology of extraction, optimization and application of crisp and fuzzy logical rules. *IEEE Transactions on Neural Networks*, vol. 12, 2001, 277-306.
- [6] Vapnik V N. *The Nature of Statistical Learning Theory*. Information Science and Statistics, Springer, 1st edition, 1995.
- [7] Boser B E, Guyon I M, Vapnik V N. A Training Algorithm for Optimal Margin Classifiers. In: Proceedings of the Fifth Annual Workshop of Computational Learning Theory (COLT), 1992.
- [8] Aizerman M, Braverman E, Rozonoer L. Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control* 25: 821-837, 1964.
- [9] Cortes C, Vapnik V. Support-vector network, *Machine Learning*, 20:273-297, 1995.
- [10] Vapnik V. *Statistical Learning Theory*. Wiley, New York, NY, 1998.
- [11] Breiman L. Bagging Predictors, *Machine Learning*, vol. 24, no. 2, 123-140, 1996.
- [12] Dong L, Yuan Y, Cai Y. Using Bagging Classifier to Predict Protein Domain Structural Class. *Journal of Biomolecular Structure & Dynamics*, Volume 24, Issue Number 3, 2006.
- [13] Freund Y, Schapire R E. Experiments with a New Boosting Algorithm. *ICML*, 1996.
- [14] Kuncheva L. Diversity in Multiple Classifier Systems (editorial). *Information Fusion*, vol. 6, no. 1, 3-4, 2004.
- [15] Rodriguez J J, Kuncheva L I, Alonso C J. Rotation Forest: A New Classifier Ensemble Method. In: *IEEE Transactions on pattern analysis and machine intelligence*, vol. 28, no. 10, October 2006.
- [16] Ting K M, Witten I H. Stacking Bagged and Dagged Models. In: *Fourteenth international Conference on Machine Learning*, San Francisco, CA, 367-375, 1997.
- [17] Prem Melville, Raymond J. Mooney. Constructing Diverse Classifier Ensembles using Artificial Training Examples. *Proceedings of the IJCAI-2003*, pp.505-510, Acapulco, Mexico, August 2003.
- [18] Webb G I. *MultiBoosting: A Technique for Combining Boosting and Wagging*. Machine Learning, 40, 159-39, Kluwer Academic Publishers, Boston, 2000.
- [19] Friedman J, Hastie T, Tibshirani R. Additive logistic regression: A statistical view of Boosting. *The Annals of Statistics* 2000, vol. 28, No. 2, 337-407.
- [20] Guyon I, Weston J, Barnhill S, Vapnik V. Gene selection for cancer classification using support vector machines. *Machine Learning*, 46:389-422, 2002.
- [21] Weka: Data Mining Software in Java, <http://www.cs.waikato.ac.nz/ml/weka/>
- [22] Yugal Kumar, G. Sahoo. Study of Parametric Performance Evaluation of Machine Learning and Statistical Classifiers. *International Journal of Information Technology and Computer Science (IJITCS)*, vol. 5, no. 6, pp. 57-64, May 2013.
- [23] Frank A, Asuncion A. *UCI Machine Learning Repository* [<http://archive.ics.uci.edu/ml>]. Irvine, CA: University of California, School of Information and Computer Science, 2010.



Jasmina Đ Novakovic is a professor of computer science at Belgrade Business School, Belgrade, Serbia. She received B.S. and M.S. in computer science from University of Belgrade in 1989 and 2000 respectively and also Ph. D. in computer science from University of Megatrend in 2003. Her current research interests include artificial intelligence methods in decision making. She teaches following courses: Informatics, E-business, Computer applications and Intelligent decision support systems. Jasmina Đ Novakovic is the author of five textbooks and more than 70 scientific papers in peer reviewed Journals and Conferences.