Probabilistic Vector Machine: Scalability through Clustering Hybridization

Mihai Cimpoeşu, Andrei Sucilă, and Henri Luchian

Alexandru Ioan Cuza University, Faculty of Computer Science, Iasi, Romania
mihai.cimpoesu@info.uaic.ro

Abstract. In this paper, a hybrid clustering and classification algorithm is obtained by exploring the specific statistical model of a hyperplane classifier. We show how the seamless integration of the clustering component allows a substantial cost decrease in the training stage, without impairing the performance of the classifier. The algorithm is also robust to outliers and deals with training errors in a natural and efficient manner.

Keywords: algorithm, classification, clustering, hybrid, statistical model, probabilistic classifier.

1 Introduction

One side-effect of the need for accurate prediction in critical fields such as disease identification in biological data or identifying credit fraud in financial data has been a steady increase in the size of the datasets to be processed. Most prediction tasks require classification in order to sort and label dataset records. With the help of decision trees, a binary classifier often can be used to solve this type of problems. A binary classifier assigns either a positive or a negative label to each new unlabeled record, after having been developed using a training set of labeled records.

The introduction in [1], [2] and [3] of Support Vector Machine (SVM) and specifically, soft margin SVMs, has provided a good tool to deal with classification tasks. SVM is one of the most succesful algorithms in the field of classification, with numerous applications in text processing, bioinformatics and many other fields. However, the basic model of a soft margin SVM has some drawbacks. Some of these are: the inability to directly deal with outliers, defining the decision function based solely on border points and ignoring the distribution of the training set. These have been addressed by the introduction of error bounds in [4] and based on these bounds, a method for choosing parameters for SVMs has been presented in [5]. The bounds developed are not tight and overestimate the probability of error.

A serious problem with SVMs has been the handling of large databases, which becomes especially cumbersome as the training phase requires five fold cross-validation of parameters. The existing solvers, such as libSVM presented in [6] and [7], have a scalability problem, both in execution time and memory requirements which increase by an $O(n^2)$ factor in the database size. As a consequence, the authors of [8], [9] and [10] integrate clustering techniques in the training

L. Correia, L.P. Reis, and J. Cascalho (Eds.): EPIA 2013, LNAI 8154, pp. 187–198, 2013.

phase in order to reduce the size of databases and [11] changes the formulation of SVMs to a second order cone programming problem, to allow for better scalability with the increase of database size.

We propose an alternative hyperplane classifier, based on the Probabilistic Vector Machine (PVM) classifier, introduced in [12], which has a built–in approach to dealing with outliers and classification errors. This leads to a smaller number of parameters, which decreases the cost of the five fold cross-validation in the search of the effective parameters.

Although the classifier only requires to solve linear feasibility problems, these are dense and can lead to memory problems in the linear programming algorithms. Therefore, the training phase can become very expensive –similar to the SVM– and a reduction in the problem size is in order. Clustering and the PVM algorithm have a natural way of being hybridized, allowing for a unified approach of the classification problem. The idea of bringing clustering and classification together, applied on SVM, were pursued in [13], [14] and [15]. Besides applying clustering before training the PVM, we also bring them closer together by readjusting the cluster architecture during the training of the classification algorithm.

The paper is structured as follows: a quick presentation of the background on PVM in Section 2; a first variant of the PVM model, followed by the combined implicit clustering variant in Section 3; experimental results and a discussion based on these results in Section 4 and conclusions in Section 5.

2 Probabilistic Vector Machine

The classifier upon which we base our hybrid approach is the Probabilistic Vector Machine, which has been recently introduced in [12]. It is a statistical binary classifier which learns a separating hyperplane, used to make the decision whether to label a point positively or negatively. The hyperplane is chosen such as to optimize a statistical measure of the training data.

Let $S = \{(x_i, y_i | x_i \in \mathbb{R}^n), y_i \in \{\pm 1\}\}$ be the training set and let $S_+ = \{x_i \in S | y_i = 1\}$ and $S_- = \{x_i \in S | y_i = -1\}$be the training subsets comprized of positively and negatively labeled points. For a hyperplane determined by its normal, $w \in \mathbb{R}^n$, and its offset, or bias, $b \in \mathbb{R}$, we denote by E_+ and E_- the average signed distance to the hyperplane of the points in S_+ and S_- respectively. We also denote by σ_+, σ_- the average deviations from E_+ and E_- of the corresponding points. This can be written as:

$$E_+ = b + \frac{1}{|S_+|} \sum_{x_i \in S_+} < w, x_i >, \; E_- = -b - \frac{1}{|S_-|} \sum_{x_i \in S_-} < w, x_i >$$

$$\sigma_+ = \frac{1}{|S_+| - 1} \sum_{x_i \in S_+} |< w, x_i > + b - E_+|$$

$$\sigma_- = \frac{1}{|S_-| - 1} \sum_{x_i \in S_-} |< w, x_i > + b + E_-|$$

PVM searches for the hyperplane which optimizes the function:

$$\begin{cases} \min \max\{\frac{\sigma_+}{E_+}, -\frac{\sigma_-}{E_-}\} \\ E_+ \geq 1, E_- \leq -1 \end{cases} \tag{2.1}$$

This system is solvable through a series of linear feasibility problems which searches for the objective via bisection.

Nonlinear training may be achieved by using kernel functions. Let $\Phi : \mathbb{R}^n \to H$, where H is a Hilbert space, be the a projection function. Let $K : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ be the kernel function, defined as $K(u, v) =< \Phi(u), \Phi(v) >_H$. We search for w as a linear combination of the training points, $w = \sum_{i=1}^{m} \alpha_i \Phi(x_i)$.

Let $K_+^i = \frac{1}{|S_+|} \sum_{x_j \in S_+} K(x_i, x_j)$ and $K_-^i = \frac{1}{|S_-|} \sum_{x_j \in S_-} K(x_i, x_j)$. The positive and negative averages are, then:

$$E_+ = \sum_{i=1}^{m} \alpha_i K_+^i, E_- = -\sum_{i=1}^{m} \alpha_i K_-^i$$

The linear feasibility systems formulated by PVM, parameterized by $t \in [0, \infty)$:

$$Feas(t) = \begin{cases} \left|\sum_{x_i \in S} \alpha_i (K(x_i, x_j) - K_+^i)\right| \leq \sigma_+^j \ , \forall x_j \in S_+ \\ (|S_+| - 1)t \cdot \left(b + \sum_{x_i \in S} \alpha_i K_+^i\right) - \sum_{x_i \in S_+} \sigma_+^i \geq 0 \\ b + \sum_{x_i \in S} \alpha_i K_+^i \geq 1 \\ \\ \left|\sum_{x_i \in S} \alpha_i (K(x_i, x_j) - K_-^i)\right| \leq \sigma_-^j \ , \forall x_j \in S_- \\ (|S_-| - 1)t \cdot \left(-b - \sum_{x_i \in S} \alpha_i K_-^i\right) - \sum_{x_i \in S_-} \sigma_-^i \geq 0 \\ -b - \sum_{x_i \in S} \alpha_i K_-^i \geq 1 \end{cases} \tag{2.2}$$

where σ_+^j is a majorant for the deviation of $x_j \in S_+$ and σ_-^j is a majorant for the deviation of $x_j \in S_-$. The optimal hyperplane for System 2.1 may be found by searching the minimal value of t, denoted as $t_{optimal}$, for which the linear system $Feas(t)$ is still feasible.

3 Integration of a Data Clustering Step

Although PVM can be solved efficiently using any of the readily available linear solvers, it turns out that solving times can be quite long. This is in part due to the fact that system (2.2) is a dense system, which, when we deal with m entries, has $2m + 4$ equations, with $2m$ unknowns and a ratio of non zero terms of approximately 0.5 of the total size of the system. Another factor is that the systems required to be solved will always turn into degenerate ones, as –whilst $t \to t_{optimal}$– the feasible area shrinks to just one point.

The strong constraints on the feasibility systems, as well as the memory complexity explosion of simplex solvers naturally lead to the idea of sampling the training database.

The first thing to consider when reducing the training set is to observe that PVM is based on a statistical model. One can obtain the same separating hyperplane by sampling the initial training set such that only a small deviation of the statistical measures is induced. Suppose that the signed distances from the hyperplane is normally distributed, with average μ. Let E be the average signed distance of a sample set. E has the property that $\frac{(E-\mu)\sqrt{N}}{s}$ follows a Student-T distribution with $N-1$ degrees of freedom, where s is the standard deviation of the sample set and N is the sample set size. s has the property that $\frac{(N-1)s^2}{\sigma^2}$ follows a χ^2 distribution, where σ is the standard deviation of the whole population. Therefore, for very large databases, we may compute the required sample set size for a desired level of confidence.

3.1 Using Weights

A first approach in reducing the database size is to combine the training with a clustering step. It is shown below that, by using weights in the training phase, one can obtain the same statistical model for the clustered set as that obtained for the original set.

Let $C_1, C_2, \ldots, C_k \subset S$ denote a partitioning of the training data with the property that a cluster contains only points of one label:

$$\begin{aligned} &C_1 \cup C_2 \cup \ldots \cup C_k = S \\ &C_i \neq \emptyset, \forall i \in \{1, \ldots, k\} \\ &C_i \cap C_j = \emptyset, \forall i, j \in \{1, \ldots, k\}, i \neq j \\ &y_j = y_l, \forall x_j, x_l \in C_i, \forall i \in \{1, \ldots, k\} \end{aligned}$$

Let $P_i, i \in \{1, \ldots, k\}$ be the centers of these clusters:

$$P_i = \frac{1}{|C_i|} \sum_{x \in C_i} x$$

In order to obtain the same statistical model, a weighted average is used instead of the simple average when training on the cluster centers. Let C_+ be the set of clusters which contain only positively labeled points and with C_- the set of clusters which contain only negatively labeled points.

Note that $\cup_{C \in C_+} C = S_+$ and $\cup_{C \in C_-} C = S_-$. The positive and negative averages are, then :

$$\begin{aligned} E_{C_+} &= b + \tfrac{1}{|S_+|} \textstyle\sum_{C_i \in C_+} |C_i| < w, P_i > \\ E_{C_-} &= -b - \tfrac{1}{|S_-|} \textstyle\sum_{C_i \in C_-} |C_i| < w, P_i > \end{aligned}$$

Given the way P_i are computed and since $\cup_{C \in C_+} C = S_+$, we have:

$$E_{C_+} = b + \frac{1}{|S_+|} \sum_{C_i \in C_+} |C_i| < w, P_i > \tag{3.1}$$

$$= b + \frac{1}{|S_+|} \sum_{C_i \in C_+} \left(|C_i| < w, \frac{1}{|C_i|} \sum_{x \in C_i} x > \right) \tag{3.2}$$

$$= b + \frac{1}{|S_+|} \sum_{C_i \in C_+} (\sum_{x \in C_i} < w, x >) \tag{3.3}$$

$$= b + \frac{1}{|S_+|} \sum_{x \in S_+} < w, x > \qquad = E_+ \tag{3.4}$$

and similarly, $E_{C_-} = S_-$. Hence, computing the average in a weighted manner leads to obtaining the same average as on the nonclustered set. The same weighing is used when the average deviations are computed:

$$\sigma_+ = \frac{1}{|S_+|-1} \sum_{C_i \in C_+} |C_i| \sigma_+^i$$
$$\sigma_- = \frac{1}{|S_-|-1} \sum_{C_i \in C_-} |C_i| \sigma_-^i$$

but in this case one no longer obtains the same average deviations as on the original set. This is due to the inequality:

$$\left| \sum_{x \in C_i} (< w, x > + b - E) \right| \leq \sum_{x \in C_i} | < w, x > + b - E| \tag{3.5}$$

which becomes a strict inequality when there are at least two members in the first sum with opposite signs.

The two sides in (3.5) are equal if all the terms $< w, x > + b - E$ have the same sign for all $x \in C_i$.

When using kernels, the clustering will take place in the projection space. It is important to reformulate the system such that the cluster centers themselves will no longer be explicitly used.

Let $\Phi : \mathbb{R}^n \to H$ be the implicit projection function and redefine P_i, the cluster centers as:

$$P_i = \frac{1}{|C_i|} \sum_{x \in C_i} \Phi(x)$$

The normal to the hyperplane then becomes:

$$w = \sum_{i=1}^{k} \alpha_i |C_i| P_i$$
$$= \sum_{i=1}^{k} \alpha_i \left(\sum_{x \in C_i} \Phi(x) \right)$$

The $< w, P_j >_H$ scalar products in H can be rewritten as:

$$\begin{aligned} < w, P_j >_H &= \sum_{i=1}^{k} \alpha_i \left(\sum_{x \in C_i} < \Phi(x), P_j >_H \right) \\ &= \sum_{i=1}^{k} \alpha_i \frac{1}{|C_j|} \sum_{x_i \in C_i, x_j \in C_j} < \Phi(x_i), \Phi(x_j) >_H \\ &= \frac{1}{|C_j|} \sum_{i=1}^{k} \alpha_i \sum_{x_i \in C_i, x_j \in C_j} K(x_i, x_j) \end{aligned}$$

Hence, the feasibility system (2.2) can be rewrittten without explicitly using the cluster centers. Instead, they can be replaced by the average of the projected points in the clusters, resulting in the following system for $Feas(t)$:

$$\begin{cases} E_+ = b + \sum_{i=1}^{k} \alpha_i \sum_{x_i \in C_i} K_+^i \\ \left| \frac{1}{|C_j|} \sum_{i=1}^{k} \alpha_i \sum_{x_i \in C_i, x_j \in C_j} K(x_i, x_j) + b - E_+ \right| \leq \sigma_+^j, \\ \forall C_j \in C_+ \\ \sigma_+ = \frac{1}{|S_+|-1} \sum_{C_i \in C_+} |C_i| \sigma_+^i \\ \sigma_+ \leq t E_+ \\ E_+ \geq 1 \\ E_- = -b - \sum_{i=1}^{k} \alpha_i \sum_{x_i \in C_i} K_-^i \\ \left| \frac{1}{|C_j|} \sum_{i=1}^{k} \alpha_i \sum_{x_i \in C_i, x_j \in C_j} K(x_i, x_j) + b + E_- \right| \leq \sigma_-^j, \\ \forall C_j \in C_- \\ \sigma_- = \frac{1}{|S_-|-1} \sum_{C_i \in C_-} |C_i| \sigma_-^i \\ \sigma_- \leq t E_- \\ E_- \geq 1 \end{cases} \tag{3.6}$$

In doing so, we eliminate the necessity of explicitly using the cluster centers, which leads to correct average computations even when using kernels. Furthermore, the advantages of using a clustered data set are preserved, as the actual problem size coincides with that of the clustered data set, due to the decrease of the number of unknowns in the linear feasibility problem.

3.2 Dividing Clusters

Due to the inequality (3.5), the average deviations used for the clustered set can substantially differ from the ones that would result on the unclustered data. Initial testing confirmed that this induces a negative effect marked by a severe degradation of the classification ability. Therefore, it is important to obtain a set of clusters which lead to relatively small differences between the members of inequality (3.5).

We define the relative induced distortion for cluster C_j, denoted $A(C_j)$, as:

$$1 - \frac{\left|\sum_{i=1}^{k} \alpha_i \sum_{x_i \in C_i, x_j \in C_j} K(x_i, x_j) + |C_j| b - |C_j| E\right|}{\sum_{x_j \in C_j} \left|b + \sum_{i=1}^{k} \alpha_i \sum_{x_i \in C_i} K(x_i, x_j) - E\right|} \tag{3.7}$$

where E is either E_+ or E_- depending on the labeling of the points in cluster C_j. This measure reflects the relative amount by which the average deviation differs in the clustered data from the nonclustered data.

For a cluster where all its points lie on the same side of E, the nominator and denominator in (3.7) will be equal and, hence, $A(C_j) = 0$. After training the classifier for a given cluster set, we will check which clusters have a relative induced distortion over a certain threshold and split each such cluster into two new clusters. For a cluster with high relative induced distortion, C_j, the two new clusters are:

$$C_{j_+} = \{x_j \in C_j | b + \sum_{i=1}^{k} \alpha_i \sum_{x_i \in C_i} K(x_i, x_j) \geq E\}$$
$$C_{j_-} = \{x_j \in C_j | b + \sum_{i=1}^{k} \alpha_i \sum_{x_i \in C_i} K(x_i, x_j) < E\}$$

Consequently:

$$C_{j_+} \neq \emptyset, C_{j_-} \neq \emptyset$$
$$C_{j_+} \cap C_{j_-} = \emptyset$$
$$C_{j_+} \cup C_{j_-} = C_j$$
$$A(C_{j_+}) = A(C_{j_-}) = 0$$

The idea is to cycle through training and splitting clusters until a sufficiently low error in the statistical model is reached. The only question that remains open is what would be a good halting criteria.

After training the classifier with a cluster set, the actual σ_+, σ_- for the nonclustered sets can be calculated. Let σ_+^c, σ_-^c be the average deviations obtained for the clustered set and let σ_+^n, σ_-^n be the average deviations for the nonclustered set. Let:

$$t_{opt}^c = \max\{\frac{\sigma_+^c}{E_+}, \frac{\sigma_-^c}{E_-}\}$$
$$t_{opt}^n = \max\{\frac{\sigma_+^n}{E_+}, \frac{\sigma_-^n}{E_-}\}$$

Note that, when the relative distortion of all clusters is 0, the only difference between $t_{optimal}$ and t_{opt}^n is the subspace of H where w is sought. When training over nonclustered data, the dimension of the subspace, W_{train}, is at most m, the number of training points. When training over clustered data, some points are enforced to have the same weight in the expression of w. The subspace in which we search for w, $W_{train_{clst}}$, is k-dimensional, where k is the number of clusters, and $W_{train_{clst}} \subset W_{train}$.

From (3.5) it follows that $t_{opt}^c \leq t_{opt}^n$. As the hyperplane obtained in the clustered form satisfies the conditions of system (2.1), but is not necessarily optimal for system (2.1), $t_{optimal} \leq t_{opt}^n$ will also hold.

Suppose that the linear subspace generated by the implicit cluster centers is identical to the linear subspace generated by the points themselves, $W_{train_{clst}} =$

W_{train}. As the statistical model obtained in the clustered form always has lower average deviations than the non clustered form, we will also have $t^c_{opt} \leq t_{optimal}$. Furthermore, under this condition we have:

$$t^c_{opt} \leq t_{optimal} \leq t^n_{opt}$$

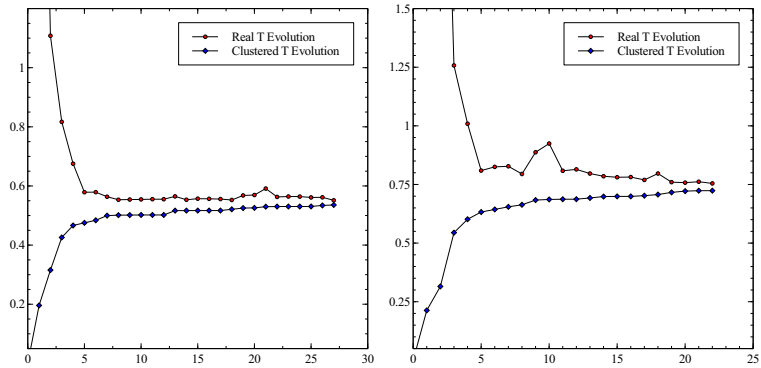
Notice that, according to Section 2, t^n_{opt} is a good indicator for the probability of error. We will use the relative difference between t^c_{opt} and t^n_{opt} as a halting criteria for our algorithm. The complete procedure for training with implicit clusters is presented in Algorithm 1.

Algorithm 1. Algorithm for training with implicit clusters

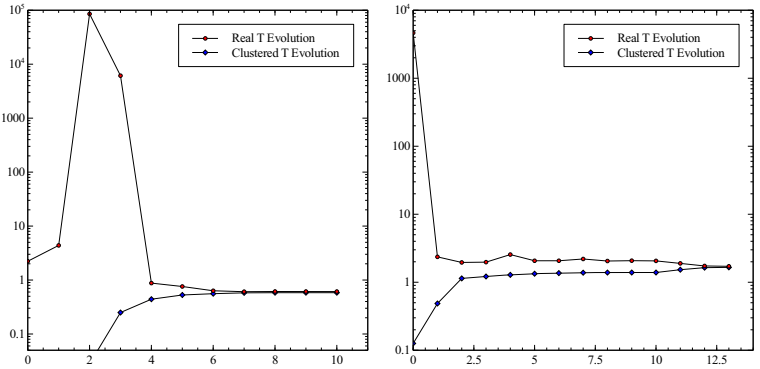
```
Function bool SplitClusters(double maxDistortion)
retValue ← FALSE
for i = Clusters.count - 1; i ≥ 0; i−− do
   if Distortion(Clusters[i]) ≥ maxDistortion then
      Clusters.Append(Clusters[i].SplitPositive())
      Clusters.Append(Clusters[i].SplitNegative())
      Clusters.Erase(i)
      retValue ← TRUE
   end if
end for
return retValue
EndFunction
maxDistortion ← 1.0
distortionDecreaseRate ← 0.9
ϵ ← 0.05
while true do
   TrainUsingCurrentClusters()
   RecomputeStatistics(t^c_opt, t^n_opt)
   if t^n_opt < t^c_opt · (1 + ϵ) then
      break
   end if
   while !SplitClusters(maxDistortion) do
      maxDistortion∗ = distortionDecreaseRate
   end while
end while
```

For all the trainings, we used an *distortionDecreaseRate* of 0.9. This parameter controls how fast the maximum relative induced distortion will decrease. Smaller values for this determine a faster termination of the algorithm, but increase the final number of clusters. The reason for this behaviour is that, when the *distortionDecreaseRate* is smaller the subsequent values obtained for *maxDistortion* are less coarse and, hence, may miss the value for which the number of clusters is minimal.

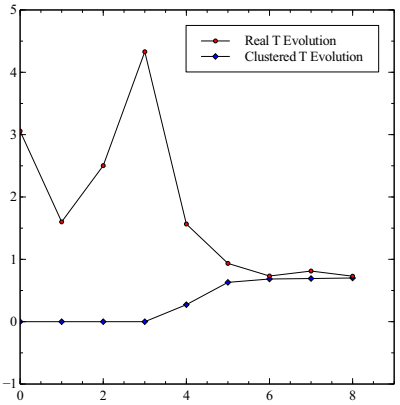
In Figure 1 we show how t^c_{opt} and t^n_{opt}, labeled as Clustered T and Real T, evolve over successive iterations on the tested databases.



(a) Credit Data **(b)** Heart Disease Wisconsin



(c) Ionosphere **(d)** Liver Buppa



(e) Spect Heart Train

Fig. 1. Evolution of t^c_{opt}(Clustered T) and t^n_{opt}(Real T). The X axis represents the iteration count, the Y axis the values for t^c_{opt} and t^n_{opt}.

4 Results and Discussion

For the *TrainUsingCurrentClusters* procedure of Algorithm 1 we have used the GNU Linear Programming Kit (GLPK), which is freely available. A version of the algorithm, with a few extra features which are not discussed in this paper, can be found at `https://code.google.com/p/dpvm/`. This code also functions with CPLEX.

We tested the classifier on 5 databases taken from the UCI ML repository. The datasets used can be found at `http://archive.ics.uci.edu/ml/`.

On each of the databases, 30 random splits were used separately, with each split containing 80% of the data for training phase and 20% for the testing phase. The results presented are averages over these 30 runs.

For reference, we also included in the comparison the soft margin SVM, which we have trained using the same kernel. We have used five fold cross-validation in order to determine the optimal value of the SVM parameter which controls error tradeoff in the objective. This parameter is usually denoted as C. We then used the obtained value for running the trainings on the 30 slices. The results

Table 1. Results for the Clustering PVM with cluster splitting

Problem Name	SVM	PVM	$t_{optimal}$	C-PVM	t^n_{opt}
Credit Screening	76.98	84.13	0.5397	84.76	0.5514
Heart Disease Cleaveland	54.23	82.11	0.7437	80.91	0.7543
Ionosphere	64.28	82.18	0.5922	82.13	0.6112
Liver	68.45	66.91	1.6927	66.31	1.715
Heart Spect Train	76.45	62.31	0.7018	66.90	0.7545

in Table 1 are obtained using an ϵ value of 0.05 in Algorithm 1. There is almost no decrease in accuracy and the value of t^n_{opt} closely follows that of $t_{optimal}$.

For $\epsilon = 0$, the final number of clusters increases only slightly, but the overall running time increases considerably as in the final iterations only a very small clusters require splitting. However, for $\epsilon = 0$, we obtain a completely identical model for the clustered and nonclustered training, giving perfectly identical results in the testing phase. This implies that the separating hyperplane is equally well expressed using only the subspace of the clustered data.

Table 2 shows how the average number of clusters obtained for a problem over the course of the 30 slices compares to the original size of the problem. These were obtained for $\epsilon = 0.05$. The ratio of clustered versus unclustered data size is, on average, 0.37.

The starting number of clusters has been set to 3% of the original dataset size. We have experimented with different numbers of starting clusters, but have found only a small variation in both the final number of clusters and the accuracy of the resulting classifier.

Table 2. Original and clustered dataset sizes

Problem Name	Original Size	Average Clustered Size
Credit Screening	690	225.03
Heart Disease Cleaveland	303	104.83
Ionosphere	351	145.7
Liver	341	67.4
Heart Spect Train	80	48.17

5 Conclusions

We have shown how, through exploitation of the specific structure of the PVM classifier, hybridization with a clustering step is possible, designed to reduce the size of the training problem whilst maintaining the properties of the statistical classifier, namely its good generalization ability and its natural stability to outliers. One important observation is that the number of clusters does not increase linearly with the size of the original dataset. The reduction is significant as the simplex solvers used are sensible to the size of the problem, in terms of memory and running time.

Future development on the matter will search for a suitable way of unifying clusters after the split procedure in order to further reduce the size of the problem. We also plan to develop a distributed solver for the problem for tackling very large databases.

Future research will also target the clustering method used, as K-means is not necessarily optimal and was chosen in this research only for its ease of usage.

Acknowledgements. This work has been supported by the European Social Fund in Romania, under the responsibility of the Managing Authority for the Sectoral Operational Programme for Human Resources Development 2007-2013 [grant POSDRU/CPP 107/DMI 1.5/S/ 78342].

References

1. Vapnik, V.N., Boser, B.E., Guyon, I.: A training algorithm for optimal margin classifiers. In: COLT 1992 Proceedings of the Fifth Annual Workshop on Computational Learning, Pittsburgh, PA, USA, vol. 5, pp. 144–152. ACM, New York (1992)
2. Cortes, C., Vapnik, V.N.: Support-vector networks. Machine Learning 20(3), 273–297 (1995)
3. Vapnik, V.N.: Statistical learning theory. John Wiley and Sons Inc. (1998)
4. Chapelle, O., Vapnik, V.N.: Bounds on error expectation for support vector machines. Neural Computation 12(9), 2012–2036 (2000)
5. Chapelle, O., Vapnik, V.N.: Choosing multiple parameters for support vector machines. Machine Learning 46(1-3), 131–159 (2001)

6. Fan, R.-E., Chen, P.-H., Lin, C.-J.: Working set selection using second order information for training support vector machines. The Journal of Machine Learning Research 6, 1889–1918 (2005)
7. Chang, C.-C., Lin, C.-J.: Libsvm: a library for support vector machines. ACM Transactions on Intelligent Systems and Technology 2(3) (2011)
8. Yu, H., Yang, J., Han, J.: Classifying large data sets using svms with hierarchical clusters. In: Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, pp. 306–315. ACM, New York (2003)
9. Ligges, U., Krey, S.: Feature clustering for instrument classification. Computational Statistics 23, 279–291 (2011)
10. Awad, M., Khan, L., Bastani, F., Yen, I.-L.: An effective support vector machines (svm) performance using hierarchical clustering. In: Proceedings of the 16th IEEE International Conference on Tools with Artificial Intelligence, Boca Raton, FL, USA, pp. 663–667. IEEE Computer Society, Washington (2004)
11. Nath, J.S., Bhattacharyya, C., Murty, M.N.: Clustering based large margin classification: A scalable approach using socp formulation. In: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Philadelphia, USA, pp. 374–379. ACM, New York (2006)
12. Sucilă, A.: A Distributed Statistical Binary Classifier. Probabilistic Vector Machine. Ph.D. Thesis, Alexandru Ioan Cuza University, Faculty of Computer Science (2012)
13. Cervantes, J., Li, X., Yu, W.: Support vector machine classification based on fuzzy clustering for large data sets. In: Gelbukh, A., Reyes-Garcia, C.A. (eds.) MICAI 2006. LNCS (LNAI), vol. 4293, pp. 572–582. Springer, Heidelberg (2006)
14. Cervantes, J., Li, X., Yu, W., Li, K.: Support vector machine classification for large data sets via minimum enclosing ball clustering. Neurocomputing 71(4-6), 611–619 (2008)
15. Nath, J.S., Bhattacharyya, C., Murty, M.N.: Clustering based large margin classification: a scalable approach using socp formulation. In: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2006, pp. 674–679. ACM, New York (2006)