A Local Expert Structure using Support Vector Machines

Matthias Rychetsky, Stefan Ortmann, Manfred Glesner
Darmstadt University of Technology
Institute for Microelectronic Systems
Karlstraße 15, D-64283 Darmstadt, Germany
Phone: +49 6151 164439, Fax: +49 6151 164936
email: Matthias.Rychetsky@mes.tu-darmstadt.de

ABSTRACT: In this paper we propose a method for the construction of a support vector machine (SVM) with local experts, using local properties of the data. To do this a SVM with large margin is first derived for the problem to learn. Then training points in the margin of this machine are used to build up several experts along the decision boundary. This is done by making a clustering in the projected high dimensional space of the SVM and using the clusters as training sets for the experts. For the local experts in this structure an upper bound on the generalization error is derived. We apply the algorithm on benchmark data and an automotive control task (knock detection) and compare the results to a normal support vector machine.

KEYWORDS: Support Vector Machines, Local Experts, Clustering, Engine Knock Detection

1 INTRODUCTION

Support Vector Machines are a powerful approach with many successful applications to construct classifiers, function approximation and time series prediction systems. Nevertheless they have some drawbacks: The number of support vectors for a real world problem can be very large, which results in high computational costs for on-line calculations (large number of kernel products have to be evaluated). Furthermore they lag an interpretability. Only from the choice of the support vectors some information can be drawn about the data. A hierarchical approach may help in both cases. It can reduce the number of calculations needed and it extracts some information about the data by the partition. This is why we propose a system which consists of local experts. These experts are only responsible for a small area in the input domain and therefore need not to be very complex (additionally experts can adapt their parameters to local properties of the data). In order to accelerate the system, experts are only placed directly at the class border (for a classification system). The remaining data is classified by a large margin SVM.

The described system is related to decision tree approaches, for which Bennett (1997) has derived a support vector technique. Kwok (1998) also proposed a system using SVM experts, which is very similar to the mixture of experts approach of Jacobs and Jordan (1991). But as we examined in our implementation this approach is not very stable in learning (oscillations during learning and some experts grow to infinity while others diminish.). This may be caused by the fact that the splitting of data to the gates is done implicitly by alternatingly learning one branch of the structure and fix the parameters of the other branch.

The paper has the following structure: First we give a brief introduction to SVMs, their properties and some important parameters. Then we present our local expert algorithm. We show how a simple SVM for the coarse grain decisions and the splitting of the data at the class border through clustering can be achieved. The following section shows an error bound for the experts. Furthermore a section gives a brief description how to extend the algorithm to regression. In the next section the performance of the system is demonstrated and finally we give conclusions and an outlook.

1.1 SHORT OVERVIEW OF SUPPORT VECTOR MACHINES

Suppose we have N given observations. Each observation consists of a pair of data: a data vector $\mathbf{x}_i \in R^n$, i=1,..,N and a class label $y_i \in \{-1,1\}$ for each vector. These data pairs build the training set Tr. Additionally, a statistically independent

but identical distributed validation set is available to tune parameters of the machine.
Support Vector Machines for classification (invented by Vladimir Vapnik (1982,1995)) in their simplest, linear form, build a hyperplane that separates the two given classes in the data with maximal margin. The output is given by:

$$u = w \cdot x - b \tag{1}$$

where **w** is the normal vector of the hyperplane. For the hyperplane u=0 the closest points lie on the planes u=+/-1 (if the data is linear separable). To maximize the margin (which is given by $1/\|\mathbf{w}\|_2$) one can solve the following optimization problem (primal problem):

$$\min_{w,\,b} \frac{1}{2}\|w\|^2 \quad \text{subject to} \quad y_i \cdot (w \cdot x_i - b) \geq 1,\ i = 1, \ldots, N \tag{2}$$

This could also be written as a Lagrangian by introducing Lagrange multipliers α for the constraints. Non-linearity is introduced in SVMs by a projection of the data through a function $\phi(\mathbf{x})$. Fortunately, one has not to use $\phi(\mathbf{x})$ but only a scalar product of the form $\phi(\mathbf{x}_1)\phi(\mathbf{x}_2)$ which can be calculated by a kernel function $K(\mathbf{x}_1,\mathbf{x}_2)$. This simplifies the calculation in learning and recall. The non separable case can be solved by introducing non-negative slack variables ξ in the constraints in equation 2. This modification suggested by Cortes and Vapnik (1995) allows, but penalizes, the failure of an example to reach the correct margin:

$$\min_{w,\,b,\,\xi} \frac{1}{2}\|w\|^2 + C\sum_i \xi_i \quad \text{subject to} \quad y_i(w \cdot x_i - b) \geq (1 - \xi_i) \quad i = 1, \ldots, N \quad \text{and} \quad \xi_i \geq 0 \quad i = 1, \ldots, N \tag{3}$$

These equations can be transformed to a standard quadratic optimization problem. To optimize a support vector machine with non-linear decision function in the non separable case one has to maximize (dual problem):

$$L_D = \sum_i \alpha_i - \frac{1}{2}\sum_{i,\,j} \alpha_i \alpha_j y_i y_j K(x_i, x_j) \quad \text{subject to} \quad 0 \leq \alpha_i \leq C \quad \text{and} \quad \sum_i \alpha_i y_i = 0 \tag{4}$$

The parameter C controls the trade off between errors of the SVM on training data and margin maximization. Large C correspond to assigning a higher penalty to errors, and small C lead to wider margin. Our algorithm described in the next section utilizes C to construct more simple SVMs.
The output of a SVM is calculated by:

$$u(x, \alpha) = \sum_{\text{support vectors}} y_i \alpha_i K(x_i, x) - b \tag{5}$$

And the final decision then takes the form:

$$f(x, \alpha) = \text{sign}(u(x, \alpha)) \tag{6}$$

2 ALGORITHM

Because of reasons discussed in the introduction, the local expert algorithm presented here is based on clustering (and not on partition learning) in the feature space and learning in the subregions. This section starts with a description of the general structure of the local expert system and then points out how the k-means algorithm can be used for clustering.

2.1 GENERAL STRUCTURE

In figure 1 you can see the architecture of the local expert SVM. First this structure tries to classify data by a normal SVM which has been trained to have a large margin and only few support vectors. Only if the point which has to be classified has $|u(\mathbf{x})|<1$ (see equation 5) the local expert structure is started. In this case partitioning SVMs are used to select the local SVM experts. These SVMs then finally make the decision to which class **x** belongs.

The large margin SVM_{SP} is constructed as following: A first SVM_{RC} build using a C, which leads to sufficiently many

vectors inside the margin. Than by removing all training points inside this margin a new training set is build, which is used to construct a simple SVM_{SP} with only few support vectors. Vectors inside the margin of the first SVM_{RC} are than taken to build up the expert structure (that is the data which lies near the class border).

For expert construction first a clustering is done using the k-means algorithm in feature space as described below. To do this the number of clusters M has to be selected beforehand. To be able to partition data also in recall, a simple SVM (as few support vectors as possible) is constructed. Inside the clusters further SVMs are setup to do the final decision. The local expert algorithm is trained as described in table I.

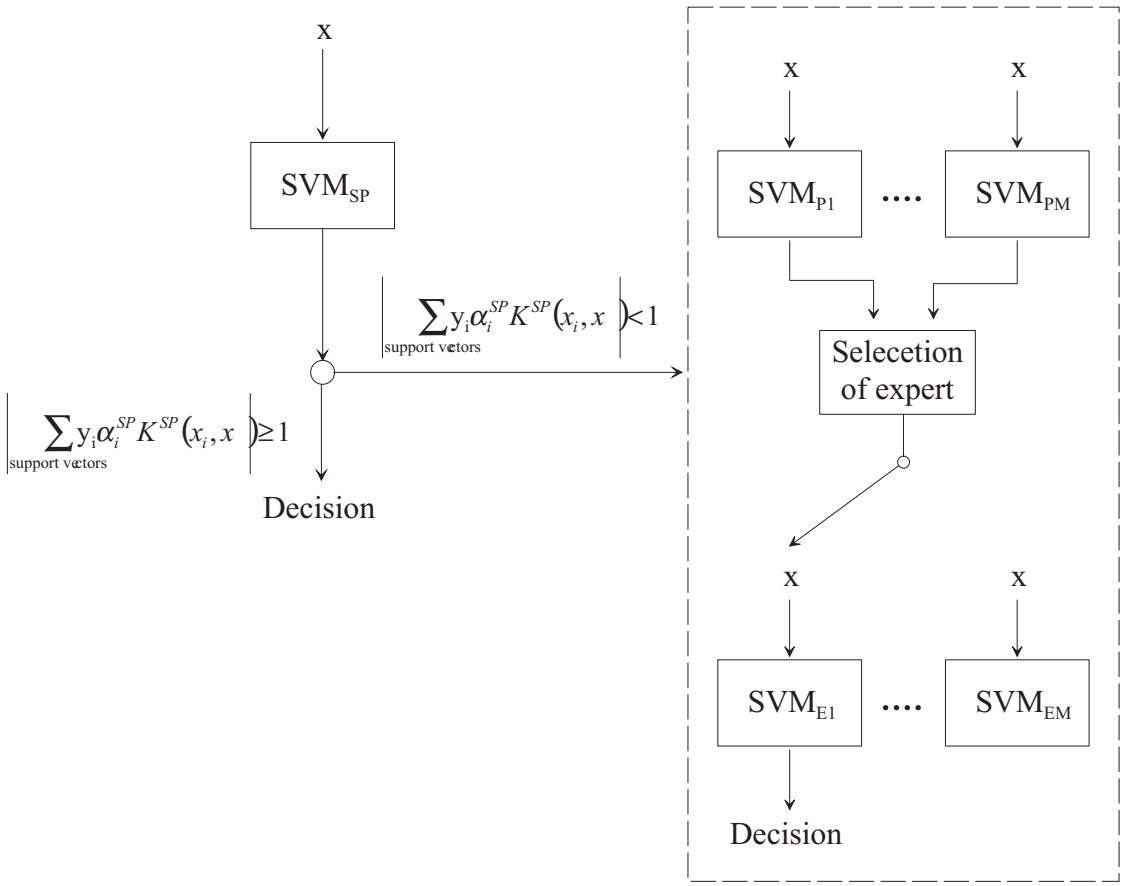

Figure 1: Architecture of local expert SVM

This approach of course only makes sense in those cases where a normal SVM would have a significantly large number of support vectors. Therefore the algorithm only should build experts (e.g. in a multi-class SVM), if the corresponding problem has vectors inside the margin of SVM_{RC} and could not be solved easily (with few support vectors) otherwise.

An important consequence of the local expert approach is, that it provides locally changing margins. That means depending on the noise level at this domain of the input space experts with different margins will be derived. This technique can also be used to adapt the kernel parameters (or even kernel types) to different domains of the input space, which may have changing properties. E.g. in regions with few vectors when using a Radial-Basis-Function kernel a large σ (and therefore a coarser approximation) can be applied. For the local experts one can chose smaller σ to utilize the denser data and get a better approximation of the class border.

Construct large margin SVM with few support vectors:

1. Derive the parameters (kernel type and parameters, C) for an optimal machine: SVM_{OPT}
2. Train SVM_{RC} with small C (compared to optimal SVM_{OPT}, if not SVM_{OPT} already had many SVs inside the margin)
3. Construct a new training set Tr_R by removing all samples which have a margin smaller than 1 (i.e. are part of set Tr_S) when classified with SVM_{RC}:

$$Tr_S = \{\{x_i, y_i\} \in Tr \,|\, (u(x_i) \cdot y_i < 1)\} \tag{7}$$

$$Tr_R = Tr \setminus Tr_S \tag{8}$$

4. Use Tr_R to train SVM_{SP} only with the points with margin larger or equal to 1.

Use samples in Tr_S to build experts and partitioning SVMs:

5. Cluster the samples in the margin with a number of clusters chosen a priori (M). This is done by applying the k-means algorithm to the data in the high dimensional projected space (see next section).
6. Use cluster sets C_1... C_M to train M experts SVM_{E1} ... SVM_{EM}. M has to be chosen sufficiently large that the complexity of experts SVM_E is not to high.
7. Train M SVMs which partition the clusters individually (SVM_{P1} ... SVM_{PM}). These SVMs select the experts in recall phase, and should therefore be simple.

Table I: ALGORITHM for a support vector machine with local experts

2.2 K-MEANS CLUSTERING IN FEATURE SPACE

In order to find clusters in the high dimensional feature space, which can be used to train the local experts, the k-means algorithm has been adapted (for k-means see Duda and Hart (1973)). This adaptation works without calculating the clusters means explicitly in the input space (which are here anyway not needed as output). This is necessary, because it is hard to calculate the reverse projection from a known point in the high dimensional space to a point in the input space (The reduced set method proposed by Schölkopf (1997) and Burges (1996) calculates a set of few new SVs in the input space, but here it would be computationally much to complex to do this). The mean of a set of points in the feature space, which is needed for k-means, can be calculated by:

$$\phi(x^m) = \frac{1}{N} \sum_{i=1}^{N} \phi(x_i) \tag{9}$$

where $\mathbf{x}^m$ is the mean point which is transformed by ϕ (the mean of the transformed set). For the k-means algorithm distances have to be calculated. This can be done in the feature space by:

$$\left\|\phi(x_1) - \phi(x_2)\right\|^2 = K(x_1, x_1) - 2K(x_1, x_2) + K(x_2, x_2) \tag{10}$$

In order to calculate the distance from a cluster mean one has to calculate:

$$\left\|\phi(x^m) - \phi(x_a)\right\|^2 = K(x^m, x^m) - 2K(x^m, x_a) + K(x_a, x_a) \tag{11}$$

$\phi(\mathbf{x}^m)$ as well as $K(\mathbf{x}^m,\mathbf{x}^m)$ are not directly know, but by using equation 9 this can be expanded to:

$$\left\|\phi(x^m) - \phi(x_a)\right\|^2 = \phi(x^m) \cdot \phi(x^m) - 2(\phi(x^m) \cdot \phi(x_a)) + K(x_a, x_a)$$

$$= \frac{1}{N} \sum_{i=1}^{N} \phi(x_i) \cdot \left(\frac{1}{N} \sum_{i=1}^{N} \phi(x_i) \right)^T - 2\frac{1}{N} \sum_{i=1}^{N} \phi(x_i)\phi(x_a) + K(x_a, x_a)$$

$$= \frac{1}{N}\sum_{i,j=1}^{N} K(x_i, x_j) \cdot -2\frac{1}{N}\sum_{i=1}^{N} K(x_i, x_a) + K(x_a, x_a) \tag{12}$$

Using this equation the k-means algorithm is given in table II.

---

1. Initialize the cluster mean points $\phi(\mathbf{x}^m)$ with M randomly chosen points from Tr_S (or the first k points of Tr_S)

2. repeat

Group all patterns with closest cluster mean into clusters $C_1 \ldots C_M$:

for all $\mathbf{x}_i$ do

Assign $\mathbf{x}_i$ to cluster j where $\min_j \left\| \phi(x^m) - \phi(x_a) \right\|^2$

end

Implicitly calculate new cluster means $\phi(\mathbf{x}^m)$: By using equation 12 the distances to the new cluster means can be stored as an expression of kernel products of known data points.

until there is no change in cluster assignments from one iteration to the next

---

Table II: ALGORITHM for k-means in feature space

3 A BOUND FOR THE GENERALIZATION PERFORMANCE

The algorithm belongs to the general class of local algorithms. Vapnik and Bottou (1995, 1993) showed how to generalize the bound for the problem of global risk minimization to local risk minimization (for a bounded function) by using a neighborhood function (which can be e.g. an gaussian kernel or a hard threshold function). This can be used to derive a local error bound for the local expert structure, because the partitioning SVMs can be seen as neighborhood functions which select the local SVMs. Vapnik's neighborhood function $K(x,x_0,\beta)$ of a point x_0 (using a locality parameter $\beta \in (0, \infty)$) satisfies two conditions:

$$0 \le K(x, x_0, \beta) \le 1 \quad \text{and}$$

$$K(x_0, x_0, \beta) = 1 \tag{13}$$

Vapnik suggested as vicinity function e.g. the "hard threshold" (which is 1 if $\|x-x_0\|<\beta/2$, otherwise 0) or a "soft threshold" (radial basis function). In our case there is no explicit x_0 and β is not directly available. That means the area for which an "expert" is responsible is not as simple as in Vapnik's neighborhood functions. The vicinity of an expert is give implicitly by the path through the classification system. The shape of the expert area Ω is given by the margin=1 surface of SVM_{RC} and the SVM_P decision surfaces. Nevertheless the conditions in equation 13 are fulfilled for $K(x,\Omega)$: only one expert at a time is responsible for Ω. That means:

$$K(x, \Omega) = \begin{cases} 1 & \text{if } x \in \Omega \\ 0 & \text{otherwise} \end{cases} \tag{14}$$

Therefore the following bound of Vapnik (1995) holds with probability $1-\eta$ for the experts in the proposed system:

$$R_E(\alpha, \Omega) \le \frac{\frac{1}{N}\sum_{i=1}^{N} L(y_i, f(x_i, \alpha))K(x_i, \Omega) + \mathcal{E}(N, h_\Sigma)}{\left(\frac{1}{N}\sum_{i=1}^{N} K(x_i, \Omega) - \mathcal{E}(N, h_\omega)\right)_+} \tag{15}$$

using $\varepsilon(N, h_\Sigma) = \sqrt{\frac{Nh(\ln(2N/h + 1) - \ln\eta/2)}{N}}$ where h_Σ is the Vapnik-Chervonenkis (VC) dimension of the set of functions $L(y_i, f(x_i, \alpha))K(x_i, \Omega)$ and h_ω is the VC dimension of the set of functions $K(x_i, \Omega)$. As loss function L usually in pattern recognition a indicator function is used. This can be e.g.:

$$L(y, f(x, \alpha)) = \begin{cases} 0 & \text{if} \quad y = f(x, \alpha) \\ 1 & \text{if} \quad y \neq f(x, \alpha) \end{cases} \tag{16}$$

4 REGRESSION USING LOCAL EXPERT SVM

The algorithm is of course not limited to classification tasks but can also be applied for regression. In support vector regression analysis (as in other approaches) the problem appears that different densities of the data in the input domain could not be well approximated. At top of figure 2 such a task and its approximation using a RBF Kernel with different width can be seen (here the ε-insensitive region, which is normally used in SVM regression, is set to zero for simplicity). Obviously both approximation are not optimal, since for small σ the regression in domains with dense data is well but at the border to sparse regions oscillations occur. Additionally in the sparse regions the RBF kernel tends to have one "bump" for every sample. On the other hand the large σ performs well in sparse regions. Now by using a local expert SVM this can be overcome.

The classification algorithm can be adapted in different ways: E.g. if an RBF kernel is used, one can construct SVM_{OPT} and SVM_{RC} in the same way as described above. But it is also possible to use σ for this task. The partitioning method described for the classification case should be changed a little to better accommodate with regression: The partition should be based directly on the output SVM_P and not on the decision SVM_{RC} margin<1. Therefore in the training phase all data is collected which has a large deviation from the SVM_{RC} regression function (e.g. α at bound C). This data can be clustered and used for training of the partitioning SVMs.

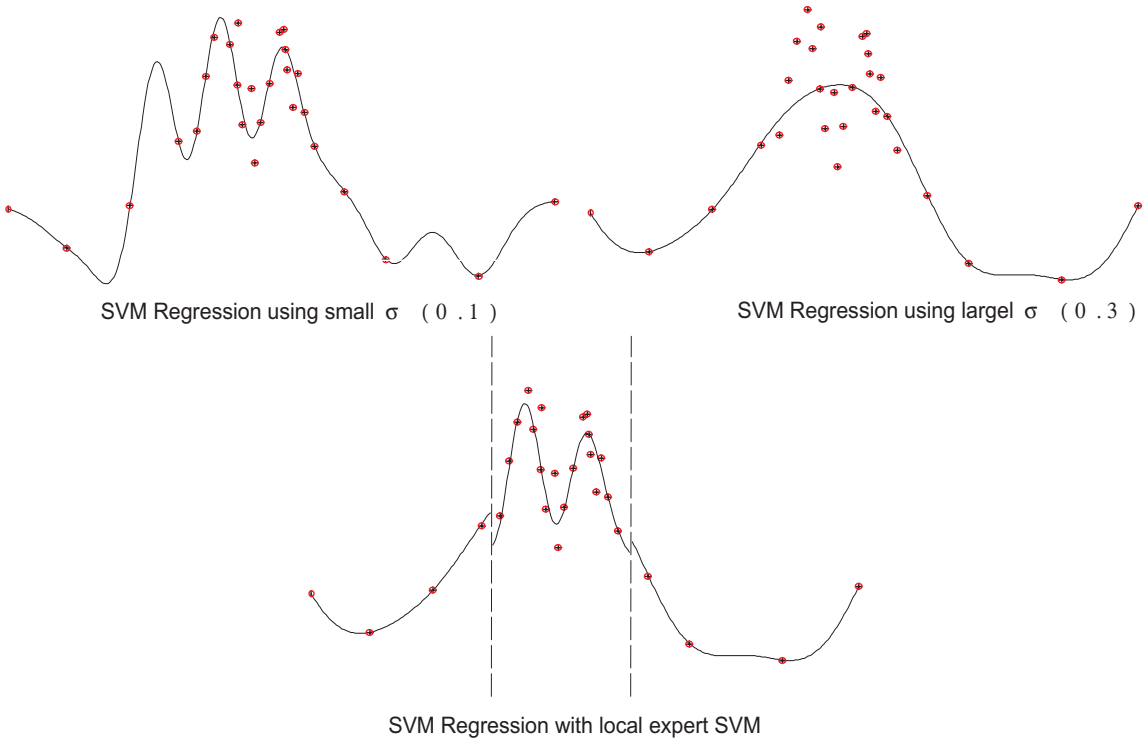

Figure 2: Example application of a local expert SVM to regression problems with different data densities

5 EXPERIMENTS

Here we show results for generalization performance and runtime complexity (number of kernel calculations/support vectors needed) for the recall step. The experiments show that number of kernel calculations can be significantly reduced without loosing too much generalization performance. For the local expert structure the number of evaluated kernel functions depend also on the data used for recall. Only if a vector lies in the margin of SVM_{SP}, the evaluation of the experts is started (therefore the mean number of support vectors is not an integer). The C parameters of the SVMs have been optimized for maximum generalization performance. The degree of the polynomial or sigma for the RBF kernel was chosen to be constant for all classes.

	Class 1	Class 2	Class 3	Error
SVM (p=6 C=1)	**4**	**25,5**	**20**	**3/75**
local expert SVM M=2 (C_{RC}=1 p_{SP}=6 C_{SP}=100 p_P =6 C_p =1000 p_E =6 C_E =5)	**4 SP: 4 No Experts**	**9,80 SP: 6,5 P1: 3,5 E1: 0 E2: 7,5**	**6,45 SP: 4 P1: 3,5 E1: 3 E2: 7**	**1/75**
local expert SVM M=2 (C_{RC}=1 p_{SP}=6 C_{SP}=100 p_P =6 C_p =1000 p_E =4 C_E =5)	4 SP: 4 No Experts	10,44 SP: 6,5 P1: 4,5 E1: 0 E2: 10,5	6,71 SP: 4 P1: 3,5 E1: 3 E2: 7,5	1/75
local expert SVM M=3 (C_{RC}=1 p_{SP}=6 C_{SP}=100 p_P =6 C_p =1000 p_E =6 C_E =5)	4 SP: 4 No Experts	11,75 SP: 6,5 P1: 3,5 P2: 3,5 P3: 4 E1: 0 E2: 2 E3:7,5	9,48 SP: 4 P1: 4 P2: 5 P3: 7 E1: 2,5 E2: 4 E3: 6,5	3/75

Table III: Results for iris data (polynomial)

An example for the iris data set in table III shows results for a polynomial kernel. The data set contains 75 training and 75 test examples. Tests have been carried out using two-fold cross-validation (using test-set as training-set and vice versa), every row represents results (mean of Ts/Tr and Tr/Ts) for one setting. The columns labeled as class 1 to 3 show the number of support vectors for the SVMs constructed. The last column shows the generalization error. The number of local experts is shown together with the machine parameters in the first column.

In table IV results using a radial-basis-function kernel and using the same setting as described before can be seen. The local expert structure is in both cases (polynomial and RBF kernel) able to build a system which performs slightly better and has fewer support vectors.

In our research project we developed an engine knock detection system for combustion engine control using advanced neural structures. Knocking is an undesired fast combustion which damages the engine. In order to approach we collected a large database with different engine states (2000 and 4000 rounds per minute; non-knocking, borderline- and hard-knocking). To determine the true knocking state a pressure sensor is used in every cylinder of the four cylinder engine. But normally only an accelerometer signal is available (because of the costs). From this sensor signal 24 features were generated. The task is to classify the current knocking condition (3 classes). The features were calculated using as a first step 8 frequency bands of a fast-fourier-transform. The second step is a statistical analysis of the energy in these 8 bands (which are calculated for one combustion) over a window (here 20 cycles). As second order features we used the mean, variance and third moment.

	Class 1	Class 2	Class 3	Error
SVM (σ=2,0 C=100)	**3**	**18,5**	**10**	**2/75**
local expert SVM M=2 (C_{RC}=100 σ_{SP}=2,0C_{SP}=1000 σ_P =2,0 C_p =1000 σ_E =2,0 C_E =100)	3 SP: 3 No Experts	9,61 SP: 7 P1: 4,5 E1: 4 E2: 5,5	5,86 SP: 5 P1: 3 E1: 1 E2: 3,5	6/75
local expert SVM M=2 (C_{RC}=10 σ_{SP}=2,0 C_{SP}=100 σ_P =2,0 C_p =1000 σ_E =2,0 C_E =100)	**3 SP: 3 No Experts**	**11,10 SP: 6 P1: 2,5 E1: 7,5 E2: 7,5**	**7,23 SP: 4 P1: 4,5 E1: 5,5 E2: 5,5**	**1,5/75**
local expert SVM M=3 (C_{RC}=10 σ_{SP}=2,0 C_{SP}=100 σ_P =2,0 C_p =1000 σ_E =2,0 C_E =100)	5 SP: 5 No Experts	19,10 SP: 4 P1: 5,5 P2: 3,5 P3: 8 E1: 0 E2: 3 E3: 7,5	11,28 SP: 6 P1: 3,5 P2: 3 P3: 10,5 E1: 5,5 E2: 4 E3: 5	4/75

Table IV: Results for iris data (RBF)

Because of the high non-linearity of the problem, neural networks are very promising and we showed already their high performance in this application (see e.g. Ortmann and Rychetsky (1998, 1997)). As conventional neural nets have some disadvantages (problem of capacity control; no error bounds available; convergence problems) we focused on Support Vector Machines approaches which provide improvements in these topics (they outperformed our other solutions in Rychetsky and Ortmann (1999)). Some results for the FILU data set using the local expert SVM are shown in table V. Here we show only results for 4000 RPM, because this is the more complex case with a higher noise level. We used 800 vectors for training and 1671 for testing.

	Class 1	Class 2	Class 3	Error
SVM (σ=2,0 C=∞)	64	158	84	112/1671
SVM (σ=2,0 C=1000)	**66**	**160**	**91**	**104/1671**
local expert SVM M=2 (C_{RC}=10 σ_{SP}=2,0C_{SP}=100 σ_P =2,0 C_p=10000 σ_E =2,0 C_E =200)	51,88 SP: 38 P1: 13 E1: 19 E2: 33	87,58 SP: 49 P1: 19 E1: 99 E2: 49	47,67 SP: 39 P1: 13 E1: 24 E2: 40	148/1671
local expert SVM M=2 (C_{RC}=10 σ_{SP}=2,0C_{SP}=100 σ_P =2,0 C_p=10000 σ_E =2,0 C_E =100)	52,31 SP: 38 P1: 13 E1: 48 E2: 43	92,12 SP: 49 P1: 19 E1: 115 E2: 52	48,38 SP: 39 P1: 13 E1: 30 E2: 42	150/1671
local expert SVM M=2 (C_{RC}=5 σ_{SP}=2,0C_{SP}=100 σ_P =2,0 C_p=10000 σ_E =2,0 C_E =200)	56,74 SP: 39 P1: 17 E1: 42 E2: 40	92,46 SP: 39 P1: 19 E1: 129 E2: 39	56,18 SP: 42 P1: 12 E1: 24 E2: 62	146/1671

Table V: Results for FILU 4000 data

	Class 1	Class 2	Class 3	Error
local expert SVM M=3 (C_{RC}=5 σ_{SP}=2,0C_{SP}=1000 σ_P =2,0 C_P=10000 σ_E =2,0 C_E=200)	61,94 SP: 39 P1: 11 P2: 20 P3: 14 E1: 12 E2: 29 E3: 38	93,89 SP: 39 P1: 22 P2: 16 P3: 26 E1: 80 E2: 21 E3: 66	60,24 SP: 42 P1: 9 P2: 22 P3: 15 E1: 17 E2: 26 E3: 41	138/1671
local expert SVM M=4 (C_{RC}=5 σ_{SP}=2,0C_{SP}=1000 σ_P =2,0 C_p=10000 σ_E =2,0 C_E=200)	69,49 SP: 39 P1: 12 P2: 15 P3: 35 P4:15 E1: 20 E2: 13 E3: 34 E4: 16	100,40 SP: 39 P1: 17 P2: 12 P3: 17 P4: 53 E1: 50 E2: 20 E3: 48 E4: 54	63,43 SP: 42 P1: 9 P2: 18 P3: 18 P4: 26 E1: 17 E2: 26 E3: 14 E4: 25	140/1671
local expert SVM M=3 (C_{RC}=5 σ_{SP}=2,0C_{SP}=1000 σ_P =2,0C_p=10000 σ_E =2,0 C_E=400)	**60,38 SP: 39 P1:11 P2:20 P3:14 E1:11 E2:22 E3:33**	**91,57 SP: 39 P1:22 P2:16 P3:26 E1:72 E2:21 E3:66**	**59,48 SP: 42 P1: 9 P2:22 P3:15 E1:17 E2:24 E3:36**	**133/1671**

Table V: Results for FILU 4000 data

6 CONCLUSIONS AND OUTLOOK

We have presented and demonstrated a new technique which trains several simple experts based on SVMs and aggregates them to a complex classification system. One disadvantage can be identified in the relatively complex design and parameter choice procedure, but the training time grows not disproportional because of the reduced data sets which are used to train local and partitioning SVMs. These SVMs are normally far less complex and therefore fast to optimize. It has been shown that the number of support vectors could be decreased significantly (and therefore the computing costs in recall) with only a small loss in generalization performance. An important consequence of this approach is that it provides locally changing margins. That means depending on the noise level at this domain of the input space experts with different margins will be derived. Figure 2 shows also a problem of the current method: at the border between local experts and the global SVM the regression (and also the decision surface in classification) is not smooth. To remove this a smoothing factor could be introduced, e.g. by weighting the two neighboring SVM with the margin of the partition SVM.

Our approach corresponds to the boosting strategy of Drucker *et al.* (1993), i.e. to focus on samples that are difficult to be classified. These samples can be found especially in the class border region. In contrast to known algorithms, e.g. adaboost from Freund and Schapire (1996) our approach explicitly identifies the critical boundary samples in a first step using a SVM technique, followed by a tree like divide-and-conquer strategy applying feature space clustering.

7 REFERENCES

K. P. Bennett and J. Blue, 1997, “A Support Vector Machine Approach to Decision Trees”. R.P.I Math Report No. 97-100, Rensselaer Polytechnic Institute, Troy, NY.

L. Bottou and V. N. Vapnik, 1992, “Local learning algorithms”. *Neural Computation,* 4(6):888-900.

C. Burges, 1996, “Simplified Support Vector decision rules”. 13th International Conference on Machine Learning.

C. Burges and B. Schölkopf, 1997, “Improving the accuracy and speed of support vector machines”. M. Mozer, M. Jordan, and T. Petsche (eds.): Neural Information Processing Systems, Vol. 9. MIT Press, Cambridge, MA.

C. Burges, 1998, “A tutorial on Support Vector Machines for Pattern Recognition”. Data Mining and Knowledge Discovery.

C. Cortes and V. Vapnik, 1995, “Support Vector Networks”. Machine Learning, 20:273-297.

H.Drucker, R.Schapire, P.Simard, 1993, “Boosting Performance in Neural Networks”. International Journal of Pattern Recognition and Artificial Intelligence, Vol. 7, No. 4, p. 705-719.

R.O. Duda and P.E. Hart, 1973, “Pattern Classification and Scene Analysis”. Wiley, New York, NY.

Y.Freund, R.E.Schapire, 1996, “Experiments with a New Boosting Algorithm”. Proceedings of the Thirteenth International Conference on Machine Learning, p. 148-156.

J.T. Kwok, 1998, “Support Vector Mixture for Classification and Regression Problems”. Proceedings of the International Conference on Pattern Recognition (ICPR), pp.255-258, Brisbane, Australia.

S. Ortmann, M.Rychetsky, et al., 1998, “Engine Knock Estimation using Neural Networks based on a real-world Database”. Tech. Paper 989513, SAE International Congress, Detroit.

M. Rychetsky, S.Ortmann, M.Glesner, 1997, “Application of hierarchical mixtures of experts networks to engine knock detection”. Proceedings of EUFIT ‘97, p.1714-1718, Aachen.

M. Rychetsky, S. Ortmann, M. Glesner, 1999, “Support Vector Approaches for Engine Knock Detection”. International Joint Conference on Neural Networks, Washington.

R. Jacobs, M. Jordan et al., 1991, “Adaptive mixtures of local experts”. Neural Computation, 3(1):79-87.

V. Vapnik, 1982, “Estimation of Dependences Based on Empirical Data”. Springer Verlag New York, Inc., USA.

V. Vapnik, 1995, “The Nature of Statistical Learning Theory”. Springer Verlag New York, Inc., USA.

V. Vapnik and L. Bottou, 1993, “Local algorithms for pattern recognition and dependencies estimation”. Neural Computation, 5(6):893-909.