Locally Linear Support Vector Machines and Other Local Models

Vojislav Kecman and J. Paul Brooks

Abstract— **The paper introduces various local models for solving machine learning (i.e., data mining) problems. In particular (and, due to their superior results) it focuses on a novel design of locally linear support vector machines classifiers. It presents them as powerful alternatives to the global (over the whole input space) nonlinear classifiers. Locally linear support vector machine (LL SVM) maximizes the margin in the original input features space and it never performs the nonlinear mapping to some kernel induced feature space. In performing such a task it uses only the *K* closest points to the query data point q. In this way it grasps the local decision function better than the standard global SVM does. This is shown to be a powerful approach when data are unevenly distributed in the input space and when a suitable decision function possesses different nonlinear characteristics in various parts of the input space. Experiments on eleven benchmark data sets display both the superior performance of LL SVMs as well as great performances of other classic locally linear classifiers. In addition, this is the first paper which proves the stability bounds for local SVMs and it shows that they are tighter than the ones for traditional, global, SVM. LL SVM is a natural classifier for multiclass problems which means that it can be easily adopted for solving regression tasks.**

I. Introduction

THERE is a well known proverb which says 'If it looks like a duck, swims like a duck and quacks like a duck, then it probably is a duck.' Such folk sayings can be believed or not but one thing is certain about them – they were learned the hard way, and this is why they must be trusted. However, there is another view about the statement above, which is that it expresses a classification (decision) rule based on a similarity, or closeness, measure. In the space of three features (attributes, inputs) being here the overall outlook, an ability to swim and a kind of sound that it produces, each bird (or, in fact, a broader class of animals) will be measured in respect how far its three features are from duck's ones. The smaller the distance between the 'duck point' and an 'unknown bird point', the higher the likelihood the unknown bird is a duck.

More formally, there is a broad range of algorithms which rely on the distance measure. In fact, all the machine learning algorithms start by measuring the distances of the known (training) data points with some metric.

Manuscript received January 24, 2010.

Vojislav Kecman is with Virginia Commonwealth University (VCU), 401 West Main Street, E4244, P.O. Box 843019, Richmond, VA, 23284-3019, USA (e-mail: vkecman@vcu.edu).

J. Paul Brooks is with the Department of Statistical Sciences and Operations Research. He is also a Fellow at Center for the Study of Biological Complexity, College of Humanities and Sciences at VCU, USA PO Box 843083, Richmond, VA 23284, (e-mail: jpbrooks@vcu.edu)

Here, we will use a simple, weighted, L_2 distance to find the K nearest neighbors (NNs) to the query $\mathbf{q}$, and to design a linear SVM (or, some other local classifier) for classifying the query point. This means that instead of one globally and nonlinearly (NL) acting SVM (or e.g., neural network) we will design many locally linear ones. Another difference between the NL global SVM and the LL SVM approach taken in this paper is that unlike the former, LL SVM will maximize the margin in the original input space. Sure, in the case that one is going to design a local nonlinear SVM (LNL SVM) classifier, the margins between the m classes within the K NN data chosen will be maximized in the kernel induced feature space again. (As for m, the following is valid $1 \leq m \leq M$, where m is the number of classes in K NNs sphere and M is a total number of classes in the given multiclass problem). The decision whether to construct locally linear or locally nonlinear SVM is the model builder's and in the rest of the paper both the notion of *local SVM* and the acronym L(N)L SVM refer to *Local Linear and/or Nonlinear SVM*. Interestingly enough, the experiments show that LL SVM usually outperforms LNL SVM. There is also theoretical evidence (shown here in section IV) indicating that L(N)L SVM is more stable than the traditional global NL SVM.

There has been a range of approaches based on a local classifier design. First, there are the ideas in [1] presenting an intuitive theoretical motivation why local models may be better by the statement that 'The use of b as an additional free parameter allows us to find deeper minima of the guaranteed risk'. In the previous sentence, b is a parameter defining the size of the neighborhood containing the closest neighbors; one approach is to let $b=K$, the number of nearest neighbors K, which will be used here. That paper was followed by an implementation of the local linear classifier in [2]. At that point, a model of choice was a classic linear classifier trained by implementing weight decay regularization. Following these two papers, a stronger theoretical foundation for local models was presented in [3]. The idea of performing local training has not always been considered as a proper one as explicitly stated in [4, page 253].

Recently, however, several algorithms performing a local classifier design have been proposed. Among them, there are two direct predecessors of this paper. In [5], the K-local Hyperplane Distance Nearest Neighbor algorithm (HKNN) is introduced. HKNN does not maximize the margin between classes directly. Instead, it approximates K data points belonging to each class by linear manifolds, and it calculates

distances between the query point **q** and each manifold. The algorithm performs quite well but there was a space for a significant improvement which is proposed in [6] and explored in [6]-[11] and [19] where the Adaptive Linear Hyperplane (ALH) algorithm is introduced and tested on more than 20 various benchmarking data sets for which it outperforms all the other data mining models (including SVMs, *K*-nearest neighbors (KNN), linear discriminant analysis (LDA), decision trees and HKNN among others), in terms of average accuracy over all the data sets [6]-[11] and [19]. ALH is a natural solver for multiclass classification problems and this led to its straightforward application for regression problems in [12] where *the regression tasks are reformulated as the multiclass classification problems*. ALH's powerful improvement is achieved: *a*) by introducing the feature weighting method and *b*) by a new approach for selecting nearest neighbors. Specifically, the *K* class prototypes are selected by using the *weighted* Euclidean distance metric, where the feature weight is estimated by using *the ratio of the between-group to within-group sums of squares*. (Some comments on weighting can be found in [19]). The very same features weighting will also be used in this paper. As for the choice of *K* nearest neighbors, unlike in HKNN which uses the *K* nearest neighbors from each class, in ALH the value of *K* refers to *K* nearest neighbors to **q**, regardless of their class. Hence, there can be from 0 to *K* NNs of a certain class within the K_m-neighborhood of **q.** It is straightforward to see that the procedure used here is also more sensible than the procedure used in HKNN when the number of samples in each class differ a lot.

The first part of the LL SVM algorithm is taken from the ALH algorithm and it applies weights to the features and chooses the *K* nearest neighbors in exactly same way as the ALH algorithm does. The only difference with ALH is that after *K* NNs data points to **q** are found, the linear SVM classifier (or some other local classifier either linear or nonlinear) will be designed over the *K* local data chosen, and the query **q** will be assigned to the class found by the obtained local classifier(s). Recently, in [15] and [16], two approaches which use 'a combination' of KNN and SVM have been presented. However, both are different in respect to LL SVM proposed here in two different ways. In [15], the local SVM has been created from a single element of each class and the classification is made by SVM if, for a query point **q**, its output is bigger than some given threshold. Otherwise the KNN classifier is used. This is far in both the spirit and the algorithm itself of what is proposed here. The approach in [16], called SVM-KNN, is closer to the ideas presented here, but there are few crucial differences worth of pointing out. SVM-KNN uses two levels of calculating distances between the query **q** and *K* nearest data; in the first step K_{sl} NNs to **q** are selected by some crude distance metric (L_2 has been mentioned as the crude one), and then *K* NNs out of K_{sl} points are chosen by using some 'accurate' distance function (e.g. tangent distance). After these two steps, a set of further steps is taken and DAGSVM from [17] is used for classifying a given query **q**. Note that DAGSVM searches for a margin in the kernel induced feature space and not in the original input space as LL SVM does. This is important because the maximal margin is defined in the *original input* features space and maximizing the margin in the *kernel induced* features space doesn't necessarily mean a creation of a maximal margin in the original input space. Next, the DAGSVM implements the pairwise construction of $m*(m-1)/2$ classifiers (where *m* stands for number of classes in *K* NNs), while LL SVM designs *m* 1-vs-all ones in solving multiclass classification problem (which is also known as the Winner-Takes-All, WTA, method). There is no significant difference between the two approaches in terms of accuracy, and, in addition, there is no problem whatsoever to use the pairwise strategy in LL models instead of WTA. Much more intriguing analysis and comments to the dilemma on how to approach multiclass classification problems and what scheme to use can be found in [18].

There is one more important difference between LL SVM and SVM-KNN. When a small *K* is used, according to [16], SVM-KNN behaves as a KNN classifier. This is never the case for L(N)L SVM even if using very small number of NNs to **q**. Namely, whatever *K* is used, the final labeling of **q** is always done by the separation hyperplane created by local SVM classifier. Just as an example, consider the case where there are 4 elements of class *i* and only 1 element from class *j* in $K = 5$ NNs sphere around **q**. The particular query **q** will be labeled based on what side of the separation hyperplane created by L(N)L SVM it lies. This means that it will never be labeled as belonging to class *i* just because there are 4 times more *i* points than *j* ones in a **q**'s 5 NNs sphere.

The paper continues as follows: section 2 introduces a sketch of the proposed weighting and *K* NN selection scheme used here (while the details can be seen in [6] and in [19]) and it also presents the strategy how L(N)L SVM is designed. Section 3 shows the accuracies of six local classifiers and it compares their performances with a few other well known classifiers. Section 4 presents the proofs of the stability (generalization ability) of L(N)L SVM classifiers. Finally, the conclusions and suggestions for future works are given.

II. L(N)L SVM For Classification Tasks

In the general supervised machine learning problems, a training set of, say, *l* instances (samples, measurements) with *d* input features is given. Each training instance belongs to one of a small set of labels with *M* classes, and it can be denoted as $\mathbf{x}_i = (x_{i1}, \ldots, x_{id})^T$ with known class label $y_i = c$, for $i = 1, \ldots, l$ and $c = 1, \ldots, M$. The objective is to predict the class label of an unlabeled query input vector **q**.

In the L(N)L SVM approach, *K* NNs of **q** are first selected, then a L(N)L SVM is constructed for the selected *K* NNs and the class label of **q** is assigned depending on what side of the separation hyperplane(s) (in original input feature

space for LL SVM and in kernel induced feature space for LNL SVM) the query **q** lies. In choosing K NNs the number K refers to K NNs to a query **q** from all the classes. Hence, there can be from 0 to K members of a certain class within the K-neighborhood of **q**.

L(N)L SVM borrows the procedure for finding K NNs to **q** from its preceding ALH algorithm by considering the features weights. In the training stage of L(N)L SVM, the associated weight for each feature is computed by using the ratio of the between-group to within-group sums of squares. Specifically, the feature weights are computed as follows,

$$r_j = \frac{\sum_i \sum_c I\left(y_i = c\right)\left(\bar{x}_{cj} - \bar{x}_j\right)^2}{\sum_i \sum_c I\left(y_i = c\right)\left(x_{ij} - \bar{x}_{cj}\right)^2}\ , \tag{1}$$

where $I(.)$ denotes the indicator function, $\bar{x}_{cj}$ denotes the j-th component of a class centroid of class c and $\bar{x}_j$ denotes the j-th component of the grand class centroid. The feature weight can then be given by the exponential weighting scheme on the normalized r_j,

$$w_j = \frac{\exp\left(TR_j\right)}{\sum_{j=1}^{d} \exp\left(TR_j\right)}\ , \tag{2}$$

where $R_j = r_j / \max(r_j)$, and T is a positive parameter that controls the influence of R_j on w_j. If $T = 0$, then $w_j = 1/d$ and the differences between the R_js are ignored. On the other hand, when T is large, a change in R_j will be exponentially reflected in w_j. Equation (2) is known as a softmax, i.e., multiple logistic, function which ensures that all the weights are between zero and one and that their sum equals one. The feature weights w_j computed from the training set will be used in the test stage too. The resulting feature weights are then used to compute all the distances given as,

$$D\left(\mathbf{x}_i, \mathbf{q}\right) = \sqrt{\sum_{j=1}^{d} w_j \left(x_{ij} - q_j\right)^2}\ . \tag{3}$$

In the NNs selection stage, the K NNs of the query **q** are selected using the K smallest weighted Euclidean distances between **q** and all the data points x_i (i = 1, …, l). Having K NNs chosen, a labeling of the unknown query data **q** is done as follows:

i) if all K NNs belong to same class c, **q** is labeled as belonging to class c i.e., $y_q = c$,
ii) if there are members of only two classes in K NNs, a single L(N)L SVM is designed and the label of the query **q** depends upon on what side of the separating hyperplane **q** lies,
iii) if there are members of more than two classes in K NNs, say all the data points in **q**'s K NNs sphere belong to m classes, then m 1-vs-all L(N)L SVMs are constructed and the query **q** takes the label of the SVM classifier which produces the biggest output given that **q** is the input. (This is a WTA approach, but in this step some other strategy can be implemented for solving multiclass classification task instead).

Note an important fact in building local classifiers – if an *LL SVM classifier* is designed over K NNs to **q**, the *margin between the classes will be maximized in the original input features space*. In other words, in each part of the input space the classifiers with guaranteed maximal local margins (the sizes of which will usually be controlled by penalty parameter C) are created for given query data points. This is a great advantage in respect to global NL SVMs designed over the whole input space and over all the training data set. The experimental results shown below will confirm the soundness of local linear models idea and the superiority of LL SVMs in respect to many other classifiers. However, there is a tiny price to pay for an improved performance. In a standard NL SVM design there are two tuning parameters; kernel's 'shape' parameter (which is usually either the order of polynomial or variance of Gaussian kernel for the two the most popular kernels), and a penalty parameter C. In training LL SVM there are three parameters; number of nearest neighbors K, feature's weighting parameter T and locally linear SVM's penalty parameter C. On the other hand, the SVM's training over small number of K data is much faster than the one over much bigger set of all training data points, and at this point it seems that a training phase of LL SVM will be usually faster. In application, i.e., in a test phase, the classic SVM will have some CPU time advantage because in the case of LL SVM one will have to find K NNs first, to design LL SVM next (however, this time with the best, K, T and C, from the training phase, which will be fast), and to classify **q**. The CPU time issues mentioned will be the subjects of future investigations.

Sure, a model builder can opt for designing local NL SVM over the K selected NNs to query **q**, instead of a LL SVM. In this case the local margin will be maximal in the chosen kernel induced feature space and the maximal separation in the original input features space will not be guaranteed. Similarly, the model builder can decide to design any other classifier over K NNs. The first obvious choice is a classic linear classifier with weight decay. This has already been investigated in [2] but without input features weighting. In the present paper, we will show how the feature weighting helps in achieving higher accuracy rates for all local models. Obviously, any weighting scheme can be applied within the local approach presented here, but the ratio of the between-group to within-group sums of squares seems to be exceptionally good for capturing relevancies of input features.

III. Experimental Results and Comparisons

In this section, the comparisons of six locally designed classifiers with eight competing classifiers, including KNN, Linear Discriminant Analysis (LDA), SVM, Nearest Feature

Line (NFL), K Local Hyperplane Nearest Neighbor (HKNN), Nearest Neighbor Line (NNL), Center-based Nearest Neighbor (CNN) and ALH, by using leave-one-out cross-validation technique, are presented. More precisely, after introducing the data sets used in Table 1, we will show the average performances of eight classifiers as given in [6] and [19] first, and then we will compare performances of six local classifiers with the two best performing classifiers (ALH and global, i.e., standard, SVM trained over whole training data set), from [6] and [19]. Eleven real,

TABLE I
ELEVEN CLASSIFICATION DATA SETS

Data set	# samples	# features	# classes
Iris	150	4	3
Glass	214	9	6
Vote	232	16	2
Wine	178	13	3
Teach	151	5	3
Sonar	208	60	2
Cancer	198	32	2
Dermatology	366	33	6
Heart	297	13	5
Prokaryotic	997	20	3
Eukaryotic	2427	20	4

benchmarking, data sets are used in experiments. The first nine data sets are taken from the UCI Machine Learning Repository [20] (http://archive.ics.uci. edu/ml/), and the last two ones are the benchmarking data sets for protein subcellular localization constructed by Reinhardt and Hubbard [21]. The basic information about all the data sets is summarized in Table 1. Table 2 shows the average results over 11 data sets for 8 classifiers and it is reproduced from [6] and [19]. We use the *leave-one-out cross-validation* (LOO CV) procedure for hyper-parameters determination and for the accuracy estimation of all classifiers over all datasets. (Only the results for protein subcellular localization while using SVM and NFL are taken from literature).

All the details of the software used and the simulations results shown in Table 2 can be found in [6] and [19]. In addition to the models shown in Table II, a series of experiments has been performed by using classification and regression trees (CART) algorithm which produced the weakest results on the data sets used here [19].

TABLE II
AVERAGE PERFORMANCES OF 8 CLASSIFIERS ON 11 DATA SETS

	KNN	LDA	SVM	NFL	HKNN	NNL	CNN	ALH
Average	83.7	77.6	**84.0**	81.8	83.6	82.0	81.3	**86.9**

Table 3 shows the performance of ALH, SVM and *six local classifiers* for all eleven data sets. Results for ALH are taken from [6] and [19] and the results for SVM are improved over the ones in [6] and [19] by using Active Set based SVM classifier from [23] and [24].

There are several interesting results in Table 3. First, LL SVM is the best classifier averaging over all 11 data sets used. Next, it is a winning model in eight out of 11 data sets and it was close second in three cases only. Third, the closest model to LL SVM is the ALH algorithm which is also a local modeling approach. Fourth, global modeling approaches,

TABLE III
PERFORMANCES OF LOCAL CLASSIFIERS AND COMPARISONS WITH TRADITIONAL SVM*

	SVM	ALH	LL SVM	LL SVM $T=0$	L LIN	L LIN $T=0$	L LIN W.D.	LNL SVM P2
Iris	97.3	97.3	**98.7**	97.3	96.7	**98.7**	**98.7**	97.3
Glass	71	75.7	**77.6**	73.4	72.9	70.6	73.8	76.2
Vote	**97**	**97**	**97**	**97**	96.1	96.6	96.1	**97**
Wine	97.2	**99.4**	**99.4**	**99.4**	97.8	97.8	98.9	**99.4**
Teach	67.6	**74.8**	72.2	64.9	70.9	64.9	71.5	71.5
Sonar	89.9	**93.8**	93.3	91.4	92.3	90.4	92.8	89.9
Cancer	**84.3**	82.8	82.8	82.3	77.3	75.3	77.3	82.8
Dermatology	98.1	98.1	**98.9**	98.6	97.5	96.7	97.5	98.1
Heart	58.6	60.3	**61.6**	**61.6**	56.9	56.9	59.3	**61.6**
Prokaryotic	91.4	**91.7**	**91.7**	**91.7**	91.6	91.4	91.6	91.2
Eukaryotic	79.4	85.3	**85.6**	85.1	84.3	84.3	84.9	85.4
Average	84.7	86.9	**87.2**	85.7	84.9	84.0	85.7	86.4

* $T=0$ means no features weighting, L Lin is a classic linear classifier, L Lin W.D. stands for L classifier with weight decay and LNL SVM P2 is a local NL SVM with 2nd order polynomial kernel.

represented by SVMs here, can't compete with local classifiers. SVMs results are behind all but one locally trained model. Fifth, features weighting always helps which can be followed by comparing LL SVM with and without weighting ($T = 0$), as well as looking at local, standard, linear classifier with and without weighting. Sixth, the local NL SVM by using second order polynomial kernel is close third over all eleven data sets; just hinting that maximizing the margin in a kernel induced feature space may be only suboptimal. Seventh, the fourth best model is the local linear classifier with weight decay which was also the approach taken in the early paper [2], and supports the argument that local models may be a good alternative. Some theoretical insight into why LL SVM performs better than global NL SVM is given in the next section.

IV. STABILITY PROPERTIES OF L(N)L SVM

Now, we establish the stability of L(N)L SVM and derive some theoretical evidence that the method can be more stable than global (traditional) SVM. (Note, the tighter the stability bound, the better the generalization). Our results follow from Rademacher Theory as developed in [25] and shown in [26].

Theorem 1([26], Theorem 4.9): For a class of functions F, let $\hat{R}(f_l)$ be the empirical loss of a decision function f_l selected from F based on a training set of size l. For any $\delta \in (0,1)$, with probability at least $1-\delta$,

$$R(f_l) \le \hat{R}(f_l) + \hat{C}(F) + 3\sqrt{\frac{\ln(2/\delta)}{2l}} \tag{4}$$

where $\hat{C}(F)$ is the Rademacher complexity of F. We now derive an upper bound on the Rademacher complexity of $F^{K,C}$, the class of all L(N)L SVM decision functions with

parameters K and C, where K is the number of nearest neighbors used for training a local classifier, and C (a.k.a. the penalty parameter) represents the tradeoff between maximizing margin and minimizing error in each local SVM classifier.

Proposition 1: Let $k(\cdot,\cdot)$ be the kernel function used when training an L(N)L SVM classifier. Then,

$$\hat{C}(F^{K,C}) \le \frac{4\sqrt{CK}}{l}\sqrt{\sum_{i=1}^{l} k(x_i,x_i)}\,. \tag{5}$$

Proof:

$$\begin{aligned}
\hat{C}(F^{K,C}) &= E_\sigma\left[\sup_{f\in F^{K,C}}\left|\frac{2}{l}\sum_{i=1}^{l}\sigma_i f_l^{K,C}(x_i)\right|\right] \\
&= E_\sigma\left[\sup_{f\in F^{K,C}}\left|\frac{2}{l}\sum_{i=1}^{l}\sigma_i \left\langle w, \Phi(x_i)\right\rangle\right|\right] \\
&\le \frac{2}{l}E_\sigma\left[\sup_{w}\|w\|\left\|\sum_{i=1}^{l}\sigma_i\Phi(x_i)\right\|\right] \\
&\le \frac{2}{l}E_\sigma\left[2\sqrt{CK}\left\|\sum_{i=1}^{l}\sigma_i\Phi(x_i)\right\|\right] \\
&\le \frac{2}{l}\left(2\sqrt{CK}\right)\sqrt{\sum_{i=1}^{l}k(x_i,x_i)}
\end{aligned} \tag{6}$$

The first line is the definition of Rademacher complexity, where the σ_i are Rademacher random variables taking values $\{\pm 1\}$. In the fourth line, we bound the value of the norm of the coefficients of the hyperplane in feature space for each local SVM. There is one local SVM for each observation. To obtain the bound, observe that for the primal SVM optimization problem for neighborhood i,

$$\begin{aligned}
&\min \frac{1}{2}\|w_i\|^2 + C\sum_{j=1}^{K}\xi_j \\
&s.t.\ y_j(w_i^T\Phi(x_j)+b) \ge 1-\xi_j\,, \\
&\xi_j \ge 0,\ j=1,\ldots,k
\end{aligned} \tag{7}$$

we can obtain a feasible solution by setting $w = 0$, $b = 1$, and $\xi_j = 2$ for all j; therefore, the optimal objective is at most $2CK$.

The other steps in the proof follow from the presentation in [25], Theorem 4.12.

End of proof

If we derive bounds for global (traditional) SVM in the same fashion (with a fixed C), we would find the following upper bound

$$\|w\| \le 2\sqrt{C\min\{N_+,N_-\}}\,, \tag{8}$$

where N_+ and N_- are the number of positive and negative observations, respectively. The bound in Proposition 1 for the Rademacher complexity would be the same except that the term $\sqrt{CK}$ would be replaced by the larger value $\sqrt{C\min\{N_+,N_-\}}$.

Thus we see that *the stability bounds for L(N)L SVM are tighter than that for traditional SVM* by a factor of

$$\sqrt{\frac{\min\{N_+,N_-\}}{K}}\,. \tag{9}$$

V. Some Final Remarks on Local SVMs and Other Local Models

Local models show better experimental performance than traditional globally acting SVM classifiers trained and designed on all the data and acting over the whole input space. In addition, the stability bounds for L(N)L SVM are tighter than that for traditional SVM. These characteristics are due to the ability of local classifiers to model complex decision functions by a collection of less complex local linear and/or nonlinear approximation. Local models are ideal for small and medium, high dimensional and sparse data sets often found in health sciences, bioinformatics and related areas. However, faced with huge data sets there is a tiny computational price to pay during the training originating from the need to select K NNs which basically requires finding all the distances between all the data in order to select K nearest neighbors to each data point. This is very well known issue with *KNN* based classifiers and there are many proposals to speed up process of finding K NNs. Any of the method proposed (and there have been quite a few recently) can be applied here too. There is however good news in the availability of massive computing boards having hundreds (and, at the moment, going to thousands) of computing cores by which the mathematically simple floating point calculations of distances can be executed very fast. Thus, very soon, building local classifiers will be a fast process even faced with huge data sets.

The local classifiers proposed here are good remedies for a very well known soft spot of the traditional KNN classifiers which is that they have been heavily influenced by noisy data. Building the local SVMs by using penalty parameter C, and allowing in this way outlying points on the wrong side of the margin, increases the accuracy of local models significantly. In addition, there is one more property of K NN classifier which is both good and not-that-good. Namely, unlike at the traditional SVMs, the information presented in training data is never lost in the approach presented here (this is a good part) because all the data are saved on the machine (which is not-that-good part).

Finally, we remark on solving regression tasks by using local SVMs and/or other local classifiers. The approach presented here can be readily applied to the regression problems in the manner as presented in [12]. L(N)L SVM

algorithm is a natural tool for solving multiclass problems. This property makes it very eligible for solving regression problems where the basic difference in respect to the classification is that the output values are real numbers. The direct way of transferring the regression problem into the classification task is to perform the discretization of the target vector **y** into a set of *N* classes. This is similar (in fact, it is almost same as) defining the 'ε-insensitivity zone' (a.k.a. ε-tube) which controls the accuracy of the approximation of the traditional SVMs in a regression. However, unlike in later, the L(N)L SVM algorithm (after the discretization) solves the regression problem as the multiclass classification task. In such a formulation, the use of the L(N)L SVM is natural.

VI. Conclusion

The paper introduces locally linear and/or nonlinear SVM classifier algorithms as well as the other local models for solving classification problems. L(N)L SVMs are natural solvers of multiclass classification tasks. The algorithms use the ratio of the between-group to within-group sums of squares for features weighting but a model presented here works for any other weighting scheme. LL SVM creates a local SVM model over *K* NNs data to the query **q** and it displays great performance in beating traditional SVM, ALH and eleven other algorithms averaging over eleven benchmarking datasets. This is attributed to the ability of local classifiers to model complex decision functions by a collection of less complex local approximations. In addition, if LL SVMs are designed they create guaranteed maximal margin classifiers in the original input features space. Theoretical evidence provided in the paper shows that the stability bounds for L(N)L SVM are tighter than that for traditional SVM. Due to the natural capacity for solving multiclass problems L(N)L SVM is also an ideal tool for solving regression tasks after transforming them into multiclass classification problems.

Acknowledgment

The paper is an independent extension of the previous joint research on ALH algorithm done by Tao Yang (while working on his PhD thesis) and the first author, who gratefully acknowledges Dr. Yang for his tireless efforts in developing novel machine learning tools.

References

[1] V. Vapnik. "Principles of risk minimization for learning theory", *Advances in Neural Information Processing Systems* (*NIPS*), 4, pp. 831-838, 1991.

[2] L. Bottou and V. Vapnik. "Local learning algorithms," *Neural Computation*, 4(6), pp. 888-901, 1992.

[3] V. Vapnik. Statistical learning theory. New York, Wiley, 1998.

[4] C. Bishop. Neural Networks for Pattern Recognition, Oxford University Press, 1996.

[5] P. Vincent, and Y. Bengio, K-local hyperplane and convex distance nearest neighbor algorithms, *Advances in Neural Information Processing Systems (NIPS)*, 14, pp. 985–992, 2001.

[6] T. Yang and V. Kecman, Adaptive local hyperplane classification. *Neurocomputing* 71, pp. 3001-3004, 2008.

[7] Yang T., Kecman V., Face recognition with adaptive local hyperplane algorithm, *Pattern Analysis & Applications, Springer-Verlag*, Vol. 13, Nr. 1, pp. 79-83, 2010.

[8] Yang T., Kecman V., Adaptive local hyperplane algorithm for learning small medical data sets, *Expert Systems, The Journal of Knowledge Engineering, Wiley Interscience, Blackwell Publishing*, Vol. 26, No. 4, pp. 355-359, 2009.

[9] G.C. Chen, J. Warren, T. Yang, and V. Kecman, Adaptive *K*-Local Hyperplane (AKLH) Classifiers on Semantic Spaces to Determine Health Consumer Webpage Metadata. In the *21st IEEE International Symposium on Computer-Based Medical Systems*, pp. 287-289, 2008.

[10] Yang T., Kecman V., Classification by ALH-Fast Algorithm. In the *Fifth International Symposium on Neural Networks (ISNN 2008),* Special Issue (to appear in an international journal), Beijing, China, 2008.

[11] V. Kecman, T. Yang, Protein Fold Recognition with Adaptive Local Hyperplane Algorithm, *IEEE Symposium Series on Computational Intelligence 2009, IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology (IEEE CIBCB 2009)*, Proceedings, paper #6006, pp.75-78, Nashville, TN, USA, 2009.

[12] V. Kecman, T. Yang, Adaptive Local Hyperplane for Regression Tasks, *Proc. of The 2009 International Joint Conference on Neural Networks (IJCNN)*, Atlanta, GA, pp. 1566-1570, 2009.

[13] Kecman V., P. J. Brooks, Locally Linear Support Vector Machines, INFORMS Annual Meeting, TA04, Joint Session ICS/DM Optimization in Data Mining/Machine Learning, Presenter P. J. Brooks, Oct. 10-14, San Diego, CA, 2009

[14] C. Domeniconi, D. Gunopulos, and J. Peng, Large Margin Nearest Neighbor Classifiers, *IEEE Trans. on NN.,* vol. 16, No. 4, pp. 899-909, 2005.

[15] X.Q. Shen, Y.P. Lin, Gene expression data classification using SVM-KNN classifier, In Proceedings of *IEEE International Symposium on Intelligent Multimedia, Video and Speech Processing*, pp. 149-152, Hong Kong, 2004.

[16] H. Zhang, A. Berg, M. Maire, and J. Malik. SVM-KNN: Discriminative nearest neighbor classification for visual category recognition, *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2, pp. 2126-2136 2006.

[17] J. C. Platt, N. Cristianini, and J. Shawe-Taylor. Large margin DAGs for multiclass classification *Advances in Neural Information Processing Systems* (*NIPS*), 12, pp. 547-553, 1999.

[18] R. Rifkin and A. Klautau, In defense of one-vs-all classification, *J. Mach. Lear. Res.*, vol. 5, pp. 101–141, 2004.

[19] T. Yang, The University of Auckland, PhD Thesis, submitted, 2009.

[20] C. Merz, P. Murphy, U.C. Irvine Repository of Machine Learning Databases, Department of Information and Computer Science, Irvine, CA, University of California, 1996.

[21] Reinhardt A., Hubbard T., Using neural networks for prediction of the subcellular location of proteins. *Nucleic Acids Research* 26, pp.2230-2236, 1998.

[22] O.G. Okun, K-local hyperplane distance nearest neighbor algorithm and protein fold recognition. *Pattern Recognition and Image Analysis* 16, pp.19-22, 2006.

[23] M. Vogt and V. Kecman, An Active-Set Algorithm for Support Vector Machines in Nonlinear System Identification. In Frank Allgöwer and Michael Zeitz, editors, *Proceedings of the 6th IFAC Symposium on Nonlinear Control Systems (NOLCOS 2004)*, September 1–3, Stuttgart, Germany, pp. 495–500, Oxford, UK, Elsevier Science, 2004.

[24] M. Vogt, Support Vector Machines for Identification and Classification Problems in Control Engineering, PhD thesis, TU Darmstadt, 2005.

[25] P. Bartlett and S. Mendelson, Rademacher and Gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research* 3, pp. 463-482, 2002.

[26] J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis,*Cambrige UP, 2004.