REFERENCES

[1] K. P. Bennett, "Decision tree construction via linear programming," in *Proc. 4th Midwest Artif. Intell. Cogn. Sci. Soc. Conf.*, M. Evans, Ed., Utica, IL, 1992, pp. 97–101.

[2] V. Cherkassky and F. Mulier, *Learning from Data – Concepts, Theory and Methods*. New York: Wiley, 1998.

[3] M. Craven and J. Shavlik, "Learning symbolic rules using artificial neural networks," in *Proc. 10th Int. Conf. Mach. Learn.*, Amherst, MA, 1993, pp. 73–80.

[4] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines*. Cambridge, U.K.: Cambridge Univ. Press, 2000.

[5] D. Decoste and B. Schölkopf, "Training invariant support vector machines," *Mach. Learn.*, vol. 46, no. 1-3, pp. 161–190, 2002.

[6] G. Fung and O. L. Mangasarian, "Proximal support vector machine classifiers," in *Proc. Knowl. Disc. Data Mining*, F. Provost and R. Srikant, Eds., San Francisco, CA, Aug. 26–29, 2001, pp. 77–86.

[7] G. Fung, O. L. Mangasarian, and J. Shavlik, "Knowledge-based nonlinear kernel classifiers," Data Mining Inst., Comput. Sci. Dept., Univ. Wisconsin, Madison, WI, Tech. Rep. 03–02, 2003 [Online]. Available: ftp://ftp.cs.wisc.edu/pub/dmi/tech-reports/03-02.ps

[8] G. Fung, O. L. Mangasarian, and J. Shavlik, "Knowledge-based support vector machine classifiers," in *Advances in Neural Information Processing Systems*, S. Becker, S. Thrun, and K. Obermayer, Eds. Cambridge, MA: MIT Press, 2003, vol. 15, pp. 521–528.

[9] M. A. Goberna and M. A. López, *Linear Semi-Infinite Optimization*. New York: Wiley, 1998.

[10] B. Haasdonk and H. Burkhardt, "Invariant kernel functions for pattern analysis and machine learning," *Mach. Learn.*, vol. 68, pp. 35–61, 2007.

[11] T. K. Ho and E. M. Kleinberg, Checkerboard Dataset, 1996 [Online]. Available: http://www.cs.wisc.edu/math-prog/mpml.html

[12] S. Y. Huang and Y.-J. Lee, "Theoretical study on reduced support vector machines," Nat. Taiwan Univ. Sci. Technol., Taipei, Taiwan, Tech. Rep., 2004.

[13] L. Kaufman, "Solving the quadratic programming problem arising in support vector classification," in *Advances in Kernel Methods – Support Vector Learning*, B. Schölkopf, C. J. C. Burges, and A. J. Smola, Eds. Cambridge, MA: MIT Press, 1999, pp. 147–167.

[14] Q. V. Le, A. J. Smola, and T. Gärtner, "Simpler knowledge-based support vector machines," in *Proc. 23rd Int. Conf. Mach. Learn.*, Pittsburgh, PA, 2006 [Online]. Available: http://www.icml2006.org/icml2006/technical/accepted.html

[15] Y.-J. Lee and O. L. Mangasarian, "RSVM: Reduced support vector machines," in *Proc. 1st SIAM Int. Conf. Data Mining*, Chicago, IL, Apr. 5–7, 2001 [Online]. Available: ftp://ftp.cs.wisc.edu/pub/dmi/tech-reports/99–10.ps

[16] Y.-J. Lee, O. L. Mangasarian, and W. H. Wolberg, "Breast cancer survival and chemotherapy: A support vector machine analysis," in *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*. Providence, RI: American Mathematical Society, 2000, vol. 55, pp. 1–10.

[17] R. Maclin, J. Shavlik, T. Walker, and L. Torrey, "A simple and effective method for incorporating advice into kernel methods," in *Proc. 21st Nat. Conf. Artif. Intell.*, 2006, pp. 427–432.

[18] O. L. Mangasarian, *Nonlinear Programming*. New York: McGraw-Hill, 1969, Reprint: SIAM Classic in Applied Mathematics 10, Philadelphia, PA, 1994.

[19] O. L. Mangasarian, J. W. Shavlik, and E. W. Wild, "Knowledge-based kernel approximation," *J. Mach. Learn. Res.*, vol. 5, pp. 1127–1141, 2004.

[20] O. L. Mangasarian and E. W. Wild, "Nonlinear knowledge in kernel approximation," *IEEE Trans. Neural Netw.*, vol. 18, no. 1, pp. 300–306, Jan. 2007.

[21] P. M. Murphy and D. W. Aha, UCI Machine Learning Repository, 1992 [Online]. Available: www.ics.uci.edu/~mlearn/MLRepository.html

[22] P. Niyogi, F. Girosi, and T. Poggio, "Incorporating prior information in machine learning by creating virtual examples," *Proc. IEEE*, vol. 86, no. 11, pp. 2196–2209, Nov. 1998.

[23] B. Schölkopf and A. Smola, *Learning with Kernels*. Cambridge, MA: MIT Press, 2002.

[24] V. N. Vapnik, *The Nature of Statistical Learning Theory*, 2nd ed. New York: Springer-Verlag, 2000.

[25] A. Wieland, Twin Spiral Dataset, [Online]. Available: http://www-cgi.cs.cmu.edu/afs/cs.cmu.edu/project/airepository/ai/areas/neural/bench/cmu/0.html

Local Classifier Weighting by Quadratic Programming

Hakan Cevikalp and Robi Polikar

***Abstract*—It has been widely accepted that the classification accuracy can be improved by combining outputs of multiple classifiers. However, how to combine multiple classifiers with various (potentially conflicting) decisions is still an open problem. A rich collection of classifier combination procedures—many of which are heuristic in nature—have been developed for this goal. In this brief, we describe a dynamic approach to combine classifiers that have expertise in different regions of the input space. To this end, we use local classifier accuracy estimates to weight classifier outputs. Specifically, we estimate local recognition accuracies of classifiers near a query sample by utilizing its nearest neighbors, and then use these estimates to find the best weights of classifiers to label the query. The problem is formulated as a convex quadratic optimization problem, which returns optimal nonnegative classifier weights with respect to the chosen objective function, and the weights ensure that locally most accurate classifiers are weighted more heavily for labeling the query sample. Experimental results on several data sets indicate that the proposed weighting scheme outperforms other popular classifier combination schemes, particularly on problems with complex decision boundaries. Hence, the results indicate that local classification-accuracy-based combination techniques are well suited for decision making when the classifiers are trained by focusing on different regions of the input space.**

***Index Terms*—Classification, classifier fusion, combining multiple classifiers, ensemble learning, local classification accuracy, neural networks, quadratic programming.**

I. INTRODUCTION

Ensemble (multiple classifier)-based systems have recently enjoyed increased attention due to their favorable classification accuracies over single-classifier systems on a wide spectrum of applications [1]–[5]. Ensemble systems improve generalization performance primarily by decreasing the variance in classification decisions. This is done by strategically combining a diverse set of classifiers. Different strategies can be followed to ensure the diversity among classifiers, e.g. classifiers with identical architectures can be trained on different subsets of training data, or classifiers with different architectures—such as neural networks with different number of layers/nodes, error goals, etc.—can be trained on the entire data set. Classifiers then make different errors on different instances, and a suitable combination of these classifiers' decisions can hence reduce the total error [1]. However, how to combine multiple classifiers with various (potentially conflicting) decisions is still an open problem.

There is a rich collection of classifier combination strategies in the literature. These strategies make different assumptions on classifier independence (dependent [6], [7] or independent [1], [8]), type of classifier output (crisp labels [9], posterior probabilities [1], [8]), aggregation strategies (global [1], [4], [7] or local [10], [11]), aggregation procedures (a function [4], [12], a neural network [10], an algorithm [9]), etc. These methods typically fall into one of two categories: classifier

Manuscript received November 9, 2007; revised July 29, 2008; accepted August 24, 2008. First published September 26, 2008; current version published October 8, 2008. This work was supported by the U.S. National Science Foundation under Grant ECS-0239090.

H. Cevikalp is with Electrical and Electronics Engineering Department, Eskisehir Osmangazi University, Meselik, 26480 Eskisehir, Turkey (e-mail: hakan.cevikalp@gmail.com).

R. Polikar is with the Electrical and Computer Engineering Department, Rowan University, Glassboro, NJ 08028 USA (e-mail: polikar@rowan.edu).

Color versions of one or more of the figures in this brief are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNN.2008.2005301

selection or classifier fusion. Classifier selection approaches use only one (or few) classifier(s) that is(are) predicted to be the most reliable in some sense [11], [13]. On the other hand, classifier fusion methods use some weighted average of many classifiers' outputs [1], [7]–[9], [12]–[14]. Such weights can be fixed for any given classifier (as in AdaBoost [7]), or weights can be dynamically updated for each query point. Classifier combination strategies also include hybrid approaches that combine classifier selection and fusion, Dempster–Shafer-based combination [15], [16], fuzzy integral, and decision templates [5], [6]. Although most classifier combination methods typically employ functions to aggregate classifier outputs, these outputs may also be treated as an input to a second-level classifier, which then makes the final decision [5], [10]. Comparison and theoretical analyses of these rules are discussed in [1], [4], [13], and [17].

In this brief, we restrict our attention to local dynamic classifier combination strategies and propose a dynamic weighting scheme to combine classifiers that are trained on different subsets of the training data. Dynamic classifier combination methods typically include a partition of the input space. The partitions can be defined based on the degree of agreement among classifiers [9], by dividing the input space into regions through clustering [6], or by using the neighborhood of the test sample [11]. In each case, a measure of competency of classifiers is estimated for each region, and the classification decisions are made by the most (more) competent classifier(s). For instance, Jacobs *et al.* [10] proposed adaptive mixtures of local experts where local experts are trained to learn from different subsets of the training data. The label assignment of a test sample is made through a gating network that decides which experts should be used for the final decision. Kuncheva [6] introduced a sophisticated dynamic method, which decides between using local classifier selection or decision templates based on a statistical significance test. Similarly, Woods *et al.* [11] proposed a simple heuristic approach, dynamic classifier selection by local accuracy (DCS-LA), for classifier selection based on the local accuracy estimation of the classifiers. In this work, we also follow a similar approach and compute local classification accuracies. However, unlike DCS-LA, which uses local accuracies to select a single classifier (hence, classifier selection), we use local accuracies to determine the best weights of the most competent classifiers. Therefore, the proposed method can be seen as a hybrid approach that combines the classifier selection and fusion. Moreover, the proposed approach also differs from other classifier fusion strategies in the way the weights are computed. Specifically, we propose to determine the weights by casting the problem as a quadratic optimization problem with a convex objective function. Hence, the determined weights are optimal for the given classifiers and the given query point to be classified, with respect to the chosen objective function.

The rest of this brief is organized as follows. In Section II, we first review DCS-LA method and then introduce the proposed method. Section III describes the experimental results. Finally, concluding remarks are given in Section IV.

II. Method

A. Problem Setting

Let us assume that we have an ensemble of classifiers, each member of which is trained to become an expert in some local region of the entire feature space. Such an ensemble can be generated in several different ways. For example, competition among classifiers can be encouraged by using a suitable objective function as in mixtures of local experts [10]. Similarly, neural network classifiers can be trained to become an expert for a particular class by changing the weight decay parameter as described in [12]. Boosting [18] also allows us to train classifiers with local expertise: AdaBoost iteratively trains a set of classifiers such that each classifier is dependent on the previous one and focuses on the instances misclassified by the previous classifiers. As a result, each classifier becomes an expert on different subsets of data. In our experiments, we use AdaBoost to train classifiers, however other algorithms may also be used.

Let $H = \{H_1, \ldots, H_L\}$ be a set of L locally expert classifiers trained on the given data by using one of the techniques mentioned above and $\{\omega_1, \ldots, \omega_C\}$ be a set of classes. We assume that all classifiers produce soft class labels. More specifically, for a given query point $\mathbf{x}_q \in \mathbb{R}^d$, each classifier H_i $(i = 1, \ldots, L)$ produces a C-dimensional hypothesis vector $\mathbf{h}_i(\mathbf{x}_q) = [h_{i1}(\mathbf{x}_q) h_{i2}(\mathbf{x}_q) \ldots h_{iC}(\mathbf{x}_q)]^\top$ where $h_{ij}(\mathbf{x}_q)$ represents the support given to class j by classifier i, which is often interpreted as an estimate of the posterior probability $p(\omega_j|\mathbf{x}_q)$ for class ω_j, satisfying $h_{ij}(\mathbf{x}_q) \in [0, 1]$ and $\sum_{j=1}^{C} h_{ij}(\mathbf{x}_q) = 1$. Some classifiers immediately offer estimates of posterior probabilities, such as the multilayer perceptron trained with backpropagation. Otherwise, one can compute these estimates of posterior probabilities by using the distances to the decision boundaries as described in [8]. Note that the decision of any classifier can be converted to a crisp label by using the maximum membership rule when crisp labels are needed for decision making

$$g\left(\mathbf{h}_i(\mathbf{x}_q)\right) = m \Leftrightarrow h_{im}(\mathbf{x}_q) = \max_{j=1,\ldots,C} \left\{h_{ij}(\mathbf{x}_q)\right\}. \tag{1}$$

For a given query sample $\mathbf{x}_q$ and an ensemble of classifiers H, our objective is to compute nonnegative weights to be used in combining classifier outputs to label the query. Formally, we search for a weight vector $\alpha(\mathbf{x}_q) = [\alpha_1 \ldots \alpha_L]^\top$ where $\alpha_i \geq 0$ and $\sum_{i=1}^{L} \alpha_i = 1$. Thus, our goal is to find the best interpolating hypothesis obtained from the classifier outputs. Once the weights are computed, the final support for each class ω_j (an estimate of posterior probability $p(\omega_j|\mathbf{x}_q)$) is computed as the weighted sum of the hypotheses of the classifiers in H

$$p(\omega_j|\mathbf{x}_q) = \sum_{i=1}^{L} \alpha_i h_{ij}(\mathbf{x}_q), \qquad j = 1, \ldots, C. \tag{2}$$

Due to constraints on the coefficients, the final hypothesis vector $\sum_{i=1}^{L} \alpha_i \mathbf{h}_i(\mathbf{x}_q)$ is a convex combination of the contributions of classifiers in the ensemble. Geometrically, the final hypothesis vector lies in the convex hull of the classifiers' output vectors considering these as data points in C-dimensional hypothesis space. Consequently, better classification performances can be obtained if the classifier outputs are diverse (i.e., they are far from each other in the C-dimensional space), which implies that the volume of the corresponding convex hull is large. On the other hand, if the classifiers' output vectors are similar (i.e., they are close to each other in the C-dimensional space), then the volume of the corresponding convex hull will be small giving rise to a combined output, which is also similar to the individual classifier outputs. During the decision making, the query is assigned to the class with the highest estimated posterior probability

$$\mathbf{x}_q \in \omega_m \Leftrightarrow p(\omega_m|\mathbf{x}_q) = \max_{j=1,\ldots,C} \left\{p(\omega_j|\mathbf{x}_q)\right\}. \tag{3}$$

Before discussing our proposed solution to this problem, we first review a related method called DCS-LA. Similar to the DCS-LA, our proposed method is also built on the concept of local accuracy of the classifiers in the ensemble. However, the proposed approach is designed to overcome certain limitations of DCS-LA, which we discuss below.

B. Dynamic Classifier Selection by Local Accuracy

DCS-LA estimates classifier accuracies in local regions of feature space near a query sample, and then employs the most locally accurate classifier for labeling the given query point [11]. In contrast to our proposed method, it does not need soft class labels because it uses crisp labels in the process. The local regions are defined in terms of the K-nearest neighbors of the query sample. To estimate the local accuracy, the authors described two approaches: 1) overall percentage of the correctly classified nearest neighbors and 2) local accuracy of the predicted class—given a classifier that assigns a query point to class ω_j, the nearest neighbors assigned to class ω_j are found, and the local accuracy of the classifier is then defined as the proportion of those points whose true labels are ω_j. The latter approach, also known as the *precision* in the context of information retrieval, is preferred by the authors and others [13] (hence, we will compare our results to those of this procedure). While these two approaches are individually analyzed and compared, their combination is not explored in [11]. Merging these two estimates of local accuracies into a unique framework is one of the features of the proposed method.

Furthermore, there may—and usually will be—several winners of the local accuracy contest, because both DCS-LA approaches use the crisp labels of the outputs. If these classifiers assign different labels to the query, making the final decision can be problematic. Tie breaking is typically handled heuristically by choosing the class that is selected most often among the tied classifiers. If a tie still exists, the classifiers with the next highest local accuracy are employed to break the tie. However, these series of heuristics do not always solve the problem because there is no guarantee that the ties will be broken by this procedure.

C. Local Classifier Weighting by Quadratic Programming

For a given query and an ensemble of classifiers, our objective is to determine nonnegative weights of classifiers in the ensemble. Intuitively, the weights should reflect the competence of the classifiers in the neighborhood of the query, and more accurate classifiers should be weighted more heavily for decision making. Tresp and Taniguchi [19] proposed a dynamic weighting function defined as the inverse of the variance depending on the query in the context of regression and they showed that the variance of a regression module can be used to assess the local competence of the module. In the classification context, however, variance-based weighting has serious drawbacks for minimization of classification errors [12]. Thus, the local accuracy of the classifiers should be defined based on the classification errors in the neighborhood of the query $\mathbf{x}_q$.

To determine the weights, we minimize the square of the Euclidean norms of the difference vectors between the weighted classifier outputs for training samples in the neighborhood of the query and corresponding target vectors. As in DCS-LA, local regions are defined in terms of the K-nearest neighbors of the query. As we show below, this leads to a convex quadratic programming problem and hence a global optimal solution exists. One may use other loss functions such as the one defined based on a smooth misclassification measure in [12]. However, an analytical solution often does not exist for such loss functions, and the final weights are usually computed using a gradient-descent procedure that may get stuck in a locally optimum solution. Furthermore, gradient-descent procedures introduce additional parameters to be fixed and they are very sensitive to initialization. Finding a good initialization point generally requires *a priori* information on the problem, which is typically not available on most practical applications. A poorly selected initialization, on the other hand, often leads to suboptimal solutions, or even divergence.

Now, let $\mathbf{t}(\mathbf{x}_k) = [t_1 \dots t_C]^\top$ be a C-dimensional true class index (target) vector whose jth component is 1, with all others zero, if the kth nearest neighbor $\mathbf{x}_k$ comes from class ω_j. Then, the weight estimation problem can be formulated as

$$\min_{\alpha(\mathbf{x}_q)} \quad \sum_{k=1}^{K} \left\| \mathbf{t}(\mathbf{x}_k) - \sum_{i=1}^{L} \alpha_i \mathbf{h}_i(\mathbf{x}_k) \right\|^2 \qquad \text{s.t.} \quad \alpha_i \geq 0, \qquad \sum_{i=1}^{L} \alpha_i = 1. \tag{4}$$

Because $\sum_{i=1}^{L} \alpha_i = 1$, the associated error for a single neighbor $\mathbf{x}_k$ can be written as

$$\epsilon(\mathbf{x}_k) = \left\| \sum_{i=1}^{L} \alpha_i \left[\mathbf{h}_i(\mathbf{x}_k) - \mathbf{t}(\mathbf{x}_k) \right] \right\|^2. \tag{5}$$

This error term is equal to $\alpha(\mathbf{x}_q)^\top \mathbf{A}^{(k)} \alpha(\mathbf{x}_q)$ where the matrix $\mathbf{A}^{(k)} \in \mathbb{R}^{L \times L}$ is

$$\mathbf{A}^{(k)} = \left(A_{ij}^{(k)} \right)_{i,j=1,\dots,L} = \left\langle \mathbf{h}_i(\mathbf{x}_k) - \mathbf{t}(\mathbf{x}_k), \mathbf{h}_j(\mathbf{x}_k) - \mathbf{t}(\mathbf{x}_k) \right\rangle. \tag{6}$$

Here, $\langle .,. \rangle$ denotes the dot product between two vectors. Note that $\mathbf{A}^{(k)}$ is a symmetric positive-semidefinite matrix. If we let $\mathbf{A} = \sum_{k=1}^{K} \mathbf{A}^{(k)}$, then our initial weight estimation problem can be written as

$$\min_{\alpha(\mathbf{x}_q)} \quad \alpha(\mathbf{x}_q)^\top \mathbf{A} \alpha(\mathbf{x}_q) \qquad \text{s.t.} \quad \alpha_i \geq 0, \qquad \sum_{i=1}^{L} \alpha_i = 1. \tag{7}$$

This is a quadratic programming problem that can be solved using standard optimization techniques. Note that the Hessian matrix $\mathbf{A} \in \mathbb{R}^{L \times L}$ is a positive-semidefinite matrix, thus the cost function is convex and a global minimum exists. Moreover, the matrix $\mathbf{A}$ is more likely to be strictly positive definite if the multiplication of number of classes and the number of the nearest neighbors $(C \times K)$ is larger than the number of classifiers L. In this case, the solution is unique and it is guaranteed to be the global minimum. One may ask under what condition on the matrix $\mathbf{A}$, the best single classifier is also equivalent to the best combined classifier. To answer this question, we first define the best single classifier H_l such that $A_{ll} = \min_j(A_{jj})$. Then, the following observation adopted from [20] holds.

Observation 1: The best single classifier H_l in the ensemble is also the best combined classifier if and only if

$$A_{ll} \leq A_{il}, \qquad \text{for all} \quad i = 1, \dots, L. \tag{8}$$

D. Integrating Confidence of Classifier Decisions to the Objective Function

We note that the above-described estimation procedure does not utilize the information on whether classifier outputs agree on the label of $\mathbf{x}_q$. This can be problematic especially if the classifiers yielding the smallest error disagree on the label of the query. In such cases, the weights will be shared randomly among such classifiers (all yielding the smallest error) even though these classifiers assign different labels to $\mathbf{x}_q$. However, as empirically demonstrated in [11], local accuracy of a classifier in the ensemble with respect to the predicted class (i.e., precision), which can be seen as a measure of confidence of the classifier in its decision, is an important guide for the decision making. Using this measure for the classification may yield better results than using overall local classification accuracy only. Therefore, we add another

term to the cost function to include this information merging the two separate definitions of local accuracy discussed in [11]. The cost of this integration is one additional design parameter as described below.

Assume that a classifier H_i assigns $\mathbf{x}_q$ to the class ω_j. Then, the corresponding confidence of the classifier is estimated as

$$ch_i = \frac{n_{tp}}{n_{tp} + n_{fp} + n_{fn}} \tag{9}$$

where n_{tp} is the number of nearest neighbors correctly classified as class ω_j (true positives), n_{fp} is the number of nearest neighbors erroneously classified as class ω_j (false positives), and n_{fn} is the number of samples in class ω_j that are erroneously assigned to other classes (false negatives). This confidence measure is similar to the local class accuracy (precision) concept of DCS-LA with the exception that we also include n_{fn} term. The chosen weights must then maximize the weighted sum of confidences, thus the new weight estimation problem can be written as

$$\begin{aligned} \min_{\alpha(\mathbf{x}_q)} \quad & \alpha(\mathbf{x}_q)^\top \mathbf{A} \alpha(\mathbf{x}_q) - \gamma \mathbf{b}^\top \alpha(\mathbf{x}_q) \\ \text{s.t.} \quad & \alpha_i \geq 0, \qquad \sum_{i=1}^{L} \alpha_i = 1. \end{aligned} \tag{10}$$

where $\mathbf{b} = [ch_1 \dots ch_L]^\top$ is the column vector of class confidence measures, and γ is the regularization parameter that controls the tradeoff between the local classification accuracy and confidence of classifiers on the weight estimation. We call the proposed method local classifier weighting by quadratic programming (LCW-QP).

Guide for Setting Design Parameters: For a given ensemble of classifiers, the proposed method has two design parameters: the number of the nearest neighbors K and the regularization parameter γ. The shape of the decision boundaries among classes, the type of the classifiers in the ensemble, and the sparsity of the training samples greatly affect the optimal values of these parameters. In other words, choosing best parameters is data and application dependent. Therefore, these parameters should be determined by fivefold or tenfold cross-validation or leave-one-out procedure. Because we have two parameters, this requires a search in a 2-D parameter space. We follow a global coarse-to-fine search to set parameter values. Specifically, we first determine the minimum and maximum values of design parameters that produce acceptable classification rates by coarsely searching over a wide range of the parameter space. Then, we construct a coarse grid over the unknown parameters using these computed values and finally perform a local search near the parameters yielding the best classification rate and compute the final best values.

III. Experiments

We tested the proposed method on binary and multiclass classification problems using several benchmark and real-world databases, comparing the approach to the maximum, average, sum, and product classifier combination rules, the weighted majority voting used in AdaBoost, and the DCS-LA method described earlier. In all experiments, the classifiers were trained using the AdaBoost distribution update rule, however, the classifiers so created were combined using the aforementioned methods, as well as the proposed approach. In addition, we also trained strong neural network classifiers with different architectures for United States Postal Office (USPS) handwritten digit database.[1] For binary classification tasks, decision stumps were used as weak base classifiers whereas (stronger) neural network classifiers were chosen in the multiclass classification problems. The number of base classifiers in the ensemble is fixed based on the performance of the weighted majority voting rule, which is the native combination rule used in AdaBoost. The individual decisions were converted to crisp labels by the maximum membership rule when crisp labels were needed for classifier combination. Unless stated otherwise, all design parameters and base classifier parameters were determined through fivefold cross validation. We followed a global coarse-to-fine search in the parameter space in each experiment and we covered values between 1 and 30 for K (1 and 50 for the USPS database), and values between 0 and 5 for γ.

[1]Available at ftp://ftp.kyb.tuebingen.mpg.de/pub/bs/data

A. Binary Classification Problems

In this group of experiments, we first conduct experiments on rotated Checkerboard data and investigate the effects of the algorithm parameters (K and γ) on the classification performance. Then, we test the generalization performance of the proposed approach on other two-class databases.

1) Experiments on the Rotated Checkerboard Data: We experimented with two variations of the rotated Checkerboard data, where each class covers specific regions of a 2-D input space with the second distribution being more challenging than the first one as shown in Fig. 1. An ensemble of $L = 50$ base classifiers were trained using decision stumps as base classifiers for the first distribution and $L = 90$ base classifiers were trained for the second. Because the decision stumps do not readily provide estimations of the posterior probabilities, the sigmoids of the distances to the decision boundaries were used to obtain estimation of posterior probabilities. To fix the parameter values, we randomly sampled 500 samples for training and 5000 samples for testing. This was repeated five times and parameters were set to the values yielding the best average classification rate. The best parameter values were found as $K = 5$ and $\gamma = 2$ for the first Checkerboard distribution and $K = 5$ and $\gamma = 1$ for the second. To test the generalization performance, we repeated the above procedure ten times and the final classification rates were determined by averaging the classification accuracies of each run. The results are given in Table I, where the asterisks indicate performance differences that are statistically significant at 5% level between the given method and the corresponding best result indicated in bold. Table I indicates that the proposed method significantly outperforms other combination rules. The improvement is particularly high for the more challenging second Checkboard data.

We have also conducted experiments to observe the effects of changing the regularization parameter and the number of the nearest neighbors on the generalization performance. To do so, we first fixed the number of the nearest neighbors to the best values found by the search scheme described earlier and then varied the regularization parameter value. The results are illustrated in Fig. 2(a). The best classification accuracy is obtained as 91.32% for $\gamma = 0$ and 87.09% for $\gamma = 0.5$ on the first and second Checkerboard distributions, respectively. These results imply that the method is not very sensitive to the changes on the regularization value. Then, we fixed the regularization parameter to those values yielding the best classification accuracies and varied the number of the nearest neighbors. The results are illustrated in Fig. 2(b). We note that the approach is indeed sensitive to the changes in the number of the nearest neighbors. The best classification rates are successively obtained as 92.81% and 87.35% for $K = 3$ on the first and second distributions, and they monotonically decrease as K further increases.

We should note that the improvement on the classification performance obtained by LCW-QP comes at a cost of additional computational complexity. In particular, additional computations are required for the nearest neighbor search and solving the quadratic programming problem. In the first Checkboard data classification problem using 50

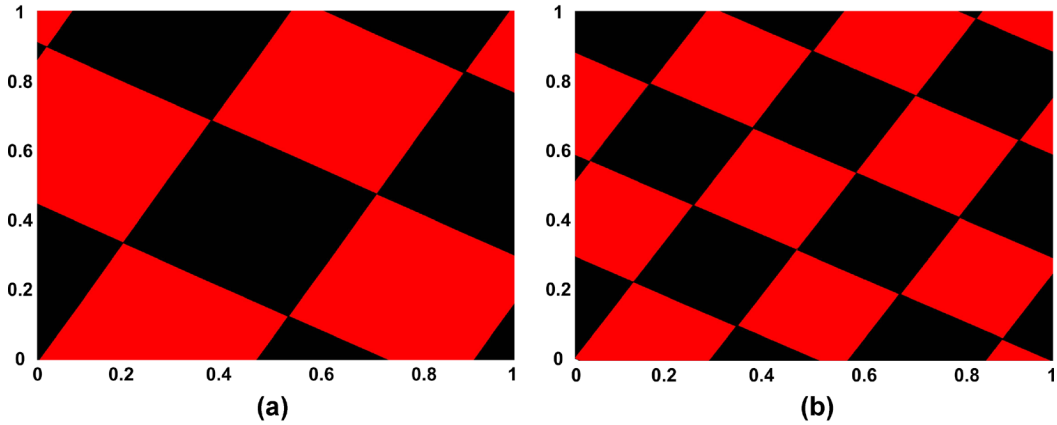

Fig. 1. Checkerboard database has two classes and each class covers the regions shaded in different colors. We sampled data points from two distributions—(a) and (b)—where the second distribution (b) is more challenging than the first one (a).

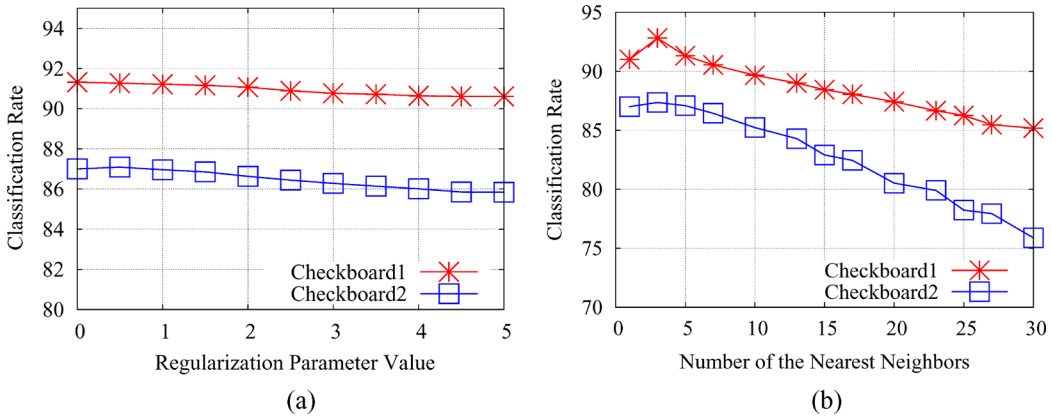

Fig. 2. Classification rates under variation in various parameters. (a) The classification rates as a function of the regularization parameter. (b) The classification rates as a function of the number of nearest neighbors.

base classifiers, it approximately takes 0.18 ms for the nearest neighbor search, 2.1 ms for evaluation of the base classifiers, and 34.2 ms for solving the quadratic programming problem on a 2.40-GHz machine with 4 GB of RAM. Testing times for evaluation of the base classifiers and quadratic programming problem depend on the number of the base classifiers in the ensemble. On the other hand, testing time of the nearest neighbor search depends on the training set size and the dimensionality of the input space. If the dimensionality and/or training set size are large, we can employ fast nearest neighbor search algorithms such as KD-trees [21] or locality sensitive hashing [22] to speed up the search procedure.

2) Experiments on Other Binary Classification Problems: In these experiments, we compared algorithms on five two-class databases: Iris,[2] Wisconsin Diagnostic Breast Cancer (WDBC), Pima, Banana, and Lithuanian Classes (LC) databases. Among these, the first three come from the University of California at Irvine (UCI) machine learning repository[3] and the remaining two databases are created using PRTOOLS Matlab toolbox.[4] The key features of these databases are summarized in Table II. In all experiments, we used fivefold cross validation to estimate design parameters and tenfold cross validation to assess the generalization performance.

[2]In fact, Iris database has three classes. One class is linearly separable from other two and the latter two are not separable from each other. We removed the linearly separable class in our experiments, leaving two classes for this experiment.

[3]From http://archive.ics.uci.edu/ml/

[4]From http://www.prtools.org/

TABLE I
CLASSIFICATION RATES (IN PERCENT) FOR THE CHECKERBOARD DATA

Method	Checkerboard 1	Checkerboard 2
LCW-QP, $K = 5$, $K = 5$	**91.07**, $\sigma = 0.72$	**86.96**, $\sigma = 1.14$
DCS-LA, $K = 5$, $K = 5$	89.42^{*}, $\sigma = 0.48$	78.88^{*}, $\sigma = 0.82$
Weighted Majority	71.88^{*}, $\sigma = 1.60$	59.11^{*}, $\sigma = 2.16$
Majority Rule	68.90^{*}, $\sigma = 2.94$	57.87^{*}, $\sigma = 2.01$
Max Rule	55.04^{*}, $\sigma = 2.32$	51.38^{*}, $\sigma = 1.03$
Sum Rule	53.45^{*}, $\sigma = 3.68$	52.58^{*}, $\sigma = 1.42$
Product Rule	53.52^{*}, $\sigma = 3.74$	52.55^{*}, $\sigma = 1.39$

TABLE II
KEY FEATURES OF SELECTED DATABASES

Databases	Data Set Size	Dimensionality
Banana	200	2
Iris	100	4
LC	200	2
Pima	768	8
WDBC	569	30

The results are given in Table III. The number of the base classifiers used in each ensemble is given in the square brackets for each database. The results suggest that when the classes are close to being linearly separable as in Iris and WDBC, most of the classifier combination strategies yield similar results, with DCS-LA having a slight lead. As the decision boundaries become more complex (Banana and

TABLE III
CLASSIFICATION RATES (IN PERCENT) ON SOME TWO-CLASS DATABASES

Method	Banana [40]	Iris [50]	LC [20]	Pima [90]	WDBC [100]
LCW-QP	**96.50**, $\sigma = 5.3$	94.00, $\sigma = 6.9$	**98.00** $\sigma = 2.5$	**77.75**, $\sigma = 3.7$	97.35*, $\sigma = 1.9$
DCS-LA,	96.00, $\sigma = 4.5$	**95.00**, $\sigma = 5.2$	97.50, $\sigma = 4.2$	63.73*, $\sigma = 6.5$	**99.13**, $\sigma = 1.2$
Weighted Majority	94.00, $\sigma = 5.1$	93.00, $\sigma = 6.7$	96.50, $\sigma = 4.1$	74.44*, $\sigma = 4.1$	97.73*, $\sigma = 1.1$
Majority Rule	94.50, $\sigma = 4.9$	92.00, $\sigma = 9.1$	97.00, $\sigma = 9.1$	68.25*, $\sigma = 4.2$	97.91*, $\sigma = 1.0$
Max Rule	39.50*, $\sigma = 6.4$	84.00*, $\sigma = 9.6$	89.50*, $\sigma = 4.3$	49.45*, $\sigma = 4.5$	84.18*, $\sigma = 4.4$
Sum Rule	80.50*, $\sigma = 6.8$	94.00, $\sigma = 6.9$	92.00*, $\sigma = 4.8$	69.34*, $\sigma = 4.8$	95.81*, $\sigma = 2.7$
Product Rule	69.50*, $\sigma = 8.9$	94.00, $\sigma = 6.9$	94.00*, $\sigma = 4.5$	64.24*, $\sigma = 3.6$	91.01*, $\sigma = 4.4$

TABLE IV
CLASSIFICATION RATES FOR THE VOC DATABASE

Methods	Classification Rate (%)
LCW-QP, $K = 5$	**93.07**, $\sigma = 0.83$
DCS-LA, $K = 5$	90.43*, $\sigma = 1.66$
Weighted Majority	80.93*, $\sigma = 1.11$
Max Rule	74.47*, $\sigma = 3.46$
Majority Rule	82.55*, $\sigma = 1.60$
Sum Rule	81.90*, $\sigma = 1.35$
Product Rule	66.26*, $\sigma = 4.90$

TABLE V
CLASSIFICATION RATES WITH WEAK CLASSIFIERS ON THE USPS DATABASE

Methods	Classification Rate (%)
LCW-QP, $K = 20$	92.83
DCS-LA, $K = 10$	88.94
SB Classifier	88.74
Weighted Majority	87.54
Max Rule	91.23
Majority Rule	91.73
Sum Rule	91.68
Product Rule	91.63

LC databases), the proposed method takes the lead and finally it significantly outperforms all other methods when the classes are very hard to separate as in the Pima database.

B. Multiclass Classification Problems

1) Experiments on the Volatile Organic Compound Database: The Volatile Organic Compound (VOC) database comes from a real-world application, where the goal is to determine the identity of a volatile organic compound, detected by an array of six quartz crystal microbalance-type sensors. There are five VOCs of interest (ethanol, octane, toluene, xylene, and trichloroethylene). The identification must be invariant to the concentration of the VOC, which makes this problem a challenging one, as different VOCs at different concentrations may—and typically have—similar responses from different sensors. The small size of this 6-D, five-class data set with 384 samples, also adds another challenge. Additional information and the database itself are available online.[5]

We trained $L = 10$ classifiers using AdaBoost distribution update rule. Because this is a relatively challenging multiclass problem, decision stumps were not appropriate as base classifiers. Therefore, we used the feedforward multilayer perceptron neural network (MLPNN) classifier as base classifiers. We deliberately left the MLPNNs undertrained via early stopping and small architecture (ten hidden-layer nodes). The number of the nearest neighbors was determined as $K = 5$, and the control parameter γ was determined as one through fivefold cross validation. Also, due to particularly challenging small size of this data set, we used leave-one-out to assess the classification accuracy. This process was also repeated five times and the final classification accuracy was determined by averaging the classification accuracies obtained in each run. The classification results are given in Table IV. As before, the asterisks indicate performance differences that are statistically significant at 5% level between the given method and the corresponding result in bold. We observe that the proposed method LCW-QP significantly outperforms other classifier combining strategies.

2) Experiments on the USPS Database: The USPS database contains 9298 16×16 gray-scale images of handwritten digits where 7291 images are allocated for training+validation and the remaining 2007 for testing. Because the training, validation, and test sets are fixed, design parameters are set by using allocated validation set and the generalization performances are assessed on the allocated test set. We performed two different types of experiments. In the first one, we trained weak classifiers, whereas in the second experiment, we trained strong neural network classifiers with different architectures to observe the generalization performance of the proposed method on strong classifiers.

For the first experiment, we trained $L = 20$ MLPNN classifiers. Each MLPNN had one hidden layer with ten nodes, with all layers using sigmoidal activation functions. The number of the nearest neighbor was set to $K = 20$, and the control parameter to $\gamma = 0.5$. The classification rates are given in Table V. In the table, we also give the classification rate of the single best (SB) classifier in the ensemble. Except for the weighted majority rule, all other classifier combination methods improve the recognition accuracy over the best classifier in the ensemble. The recognition accuracies of classifier combination schemes are mostly similar. This indicates that the weak classifiers are not adequately diverse, and most of the misclassified samples overlap. Nevertheless, the proposed method takes the advantage of small diversity among the classifiers and achieves the best performance among all other classifier combination strategies. In the second experiment, we trained $L = 5$ strong classifiers, one MLPNN and four radial basis function (RBF) neural network classifiers. The MLPNN had one hidden layer with 100 nodes. The sigmoidal functions were used in the hidden and output layers. For the RBF neural network classifiers, we used spherical Gaussian function $R_j = \exp(-\|\mathbf{x} - \mu_j\|^2/\sigma_j)$, $j = 1, \ldots, h$, as hidden unit function, where h represents the number of hidden units. To ensure the diversity among the classifiers, each RBF classifier was initialized with different number of hidden units (h) at different positions. The number and the initial positions of the hidden units were determined by using the support vector machines (SVMs) [23]. We first trained one-against-all SVM [24] classifiers using linear kernel, polynomial kernels with degree $n = 2, 3$, and the Gaussian kernel. Then, extracted support vectors were used to initialize the centers of RBF neural network classifiers. The initial widths of all hidden functions were fixed to $\sigma_j = 1.5 d_{\min}$, $j = 1, \ldots, h$, where $d_{\min}$ is the minimum distance among all support vectors belonging to different classes. The control parameter γ was again set to 0.5. All parameters were chosen using the error rates of the allocated validation

[5]Available at http://users.rowan.edu/~polikar/RESEARCH/vocdb.html

TABLE VI
CLASSIFICATION RATES WITH STRONG CLASSIFIERS ON THE USPS DATABASE

Methods	Classification Rate (%)
LCW-QP, $K = 40$	**96.31**
DCS-LA, $K = 40$	96.06
SB Classifier	96.21
Weighted Majority	95.62
Max Rule	94.87
Majority Rule	96.06
Sum Rule	95.62
Product Rule	95.62

set. After fixing classifier parameters, the validation set is added to the training set and the classifiers are retrained using this new training set. Classification rates are given in Table VI.

The minimum error rate is obtained by the proposed weighting scheme using $K = 40$ nearest neighbors. Using other classifier combination schemes yields worse recognition accuracies than the accuracy of the best classifier in the ensemble. It should be noted that the majority, sum, and product rules smooth out the differences among the classifiers because these combination rules implicitly assign *static* weights to all classifiers. Consequently, it is not possible to choose the best classifier for a given sample, resulting in poorer results. On the other hand, the proposed scheme successfully chooses the best classifiers for a particular sample and ignores the contribution of other less accurate classifiers by assigning *dynamic* (instance specific) weights to all classifiers. As a result, an improvement is obtained over the best single classifier in the ensemble.

IV. SUMMARY AND CONCLUSION

In this brief, we proposed a local dynamic weighting system to linearly combine classifiers. For a given query sample, we estimate local classification accuracy of each classifier and classifier confidences for labeling the query. The local regions are defined in terms of the K-nearest neighbors of the query. Estimating local weights of classifiers is formulated as a convex quadratic optimization problem. The optimization returns nonnegative weight coefficients such that the most locally accurate classifiers contribute more to the decision making whereas the contribution of less accurate classifiers is largely ignored. There are two parameters to be fixed in the proposed method: the number of the nearest neighbors K and the regularization parameter γ. Although the method's generalization performance greatly depends on the number of chosen nearest neighbors, the regularization parameter does not appear to have the same impact on the performance. In most cases, the best values of γ were small, suggesting that the classifiers yielding the smallest error on the nearest neighbors typically agree on the label of the query point, which in turn reduces the effect of classifier confidence term in the optimization procedure. Nevertheless, small improvements were obtained by nonzero values of γ in our experiments. Hence, keeping the classifier confidence term makes the method more flexible and versatile for various classification tasks.

To demonstrate the efficacy of the proposed weighting scheme, we tested it on both binary and multiclass classification problems. The proposed scheme along with DCS-LA outperformed other existing classifier combination methods in all cases verifying that local classification-accuracy-based combination methods are better suited when classifiers are trained by focusing on different regions of the input space. Compared to the other local combination rule DCS-LA, the proposed method generally yielded better results, particularly when the decision boundaries are highly nonlinear and complex.

REFERENCES

[1] J. Kittler, R. P. W. Duin, and J. Matas, "On combining classifiers," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, no. 3, pp. 226–239, Mar. 1998.

[2] R. Polikar, A. Topalis, D. Parikh, D. Green, J. Frymiare, and C. M. Clark, "An ensemble based data fusion approach for early diagnosis of Alzheimer's disease," *Inf. Fusion*, vol. 9, pp. 83–95, 2008.

[3] R. Polikar, "Ensemble based systems in decision making," *IEEE Circuits Syst. Mag.*, vol. 6, no. 3, pp. 21–45, 2006.

[4] G. Fumera and F. Roli, "A theoretical and experimental analysis of linear combiners for multiple classifier systems," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 6, pp. 942–956, Jun. 2005.

[5] L. I. Kuncheva, "Decision templates for multiple classifier fusion: And experimental comparison," *Pattern Recognit.*, vol. 34, pp. 299–314, 2001.

[6] L. I. Kuncheva, "Switching between selection and fusion in combining classifiers: An experiment," *IEEE Trans. Syst. Man Cybern. B, Cybern.*, vol. 32, no. 2, pp. 146–156, Apr. 2002.

[7] Y. Freund and R. E. Schapire, "A decision theoretic generalization of on-line learning and application to boosting," *J. Comput. Syst. Sci.*, vol. 55, pp. 119–139, 1997.

[8] D. M. J. Tax, M. van Breukelen, R. W. Duin, and J. Kittler, "Combining multiple classifiers by averaging or multiplying," *Pattern Recognit.*, vol. 33, pp. 1475–1485, 2000.

[9] T. K. Ho, J. J. Hull, and S. N. Srihari, "Decision combination in multiple classifier systems," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 16, no. 1, pp. 66–75, Jan. 1994.

[10] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton, "Adaptive mixtures of local experts," *Neural Comput.*, vol. 3, pp. 79–87, 1991.

[11] K. Woods, W. P. Kegelmeyer, and K. Bowyer, "Combination of multiple classifiers using local accuracy estimates," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 19, no. 4, pp. 405–410, Apr. 1997.

[12] N. Ueda, "Optimal linear combination of neural networks for improving classification performance," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 2, pp. 207–215, Feb. 2000.

[13] L. I. Kuncheva, "A theoretical study on six classifier fusion strategies," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 2, pp. 281–286, Feb. 2002.

[14] J. Grim, J. Kittler, P. Pudil, and P. Somol, "Multiple classifier fusion in probabilistic neural networks," *Pattern Anal. Appl.*, vol. 5, pp. 221–233, 2002.

[15] G. Rogova, "Combining the results of several network classifiers," *Neural Netw.*, vol. 7, pp. 777–781, 1994.

[16] Y. Lu, "Knowledge integration in a multiple classifier system," *Appl. Intell.*, vol. 6, pp. 75–86, 1996.

[17] K. Tumer and J. Ghosh, "Analysis of decision boundaries in linearly combined neural classifiers," *Pattern Recognit.*, vol. 29, pp. 341–348, 1996.

[18] Y. Freund and R. E. Schapire, "Experiments with a new boosting algorithm," in *Proc. Int. Conf. Mach. Learn.*, 1996, pp. 148–156.

[19] V. Tresp and M. Taniguchi, "Combining estimators using non-constant weighting functions," in *Advances in Neural Information Processing Systems*, G. Tesauro, D. Touretzky, and T. Leen, Eds. Cambridge, MA: MIT Press, 1995, vol. 7, pp. 419–426.

[20] L. Breiman, "Stacked regressions," *Mach. Learn.* vol. 24, no. 1, pp. 49–64, 1996.

[21] J. H. Friedman, J. L. Bentley, and R. A. Finkel, "An algorithm for finding best matches in logarithmic expected time," *ACM Trans. Math. Software*, vol. 3, pp. 209–226, 1977.

[22] P. Indyk, "Nearest neighbors in high-dimensional spaces," in *Handbook of Discrete and Computational Geometry, Chapter 39*, J. E. Goodman and J. O'Rourke, Eds., 2rd ed. Boca Raton, FL: CRC Press, 2004.

[23] B. Scholkopf, K. K. Sung, C. J. C. Burges, F. Girosi, P. Niyogi, T. Poggio, and V. Vapnik, "Comparison support vector machines with Gaussian kernels to radial basis function classifiers," *IEEE Trans. Signal Process.*, vol. 45, no. 11, pp. 2758–2765, Nov. 1997.

[24] C. W. Hsu and C. J. Lin, "A comparison of methods for multiclass support vector machines," *IEEE Trans. Neural Netw.*, vol. 13, no. 2, pp. 415–425, Mar. 2002.