Proceedings of the 11th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining (PAKDD 2007), Nanjing, China, pp. 23-34.

Multi-Represented Classification based on Confidence Estimation

Johannes Aßfalg, Hans-Peter Kriegel, Alexey Pryakhin, Matthias Schubert

Institute for Informatics, Ludwig-Maximilians-University of Munich, Germany
{assfalg,kriegel,pryakhin,schubert}@dbs.ifi.lmu.de

Abstract. Complex objects are often described by multiple representations modeling various aspects and using various feature transformations. To integrate all information into classification, the common way is to train a classifier on each representation and combine the results based on the local class probabilities. In this paper, we derive so-called confidence estimates for each of the classifiers reflecting the correctness of the local class prediction and use the prediction having the maximum confidence value. The confidence estimates are based on the distance to the class border and can be derived for various types of classifiers like support vector machines, k-nearest neighbor classifiers, Bayes classifiers, and decision trees. In our experimental results, we report encouraging results demonstrating a performance advantage of our new multi-represented classifier compared to standard methods based on confidence vectors.

1 Introduction

In many application areas such as multimedia, biology, and medicine, objects can be described in multiple ways. For example, images can be described by color histograms or texture features or proteins can be described by text annotations, sequence descriptions, and 3D shapes. To classify these multi-represented objects, it is often useful to integrate as much information as possible because the representation providing the best suitable object description might vary from object to object. A simple way to combine multiple representations would be to span a feature space with respect to all features occurring in some representation. However, this approach induces treating sparse dimensions such as word occurrences in the same way as color distributions or texture descriptions. Therefore, established methods for classifier combination, which is also called classifier fusion, train a classifier on each representation and derive a global class prediction based on the class probabilities of each of these classifiers.

In this paper, we introduce a new method for combining local class predictions based on so-called confidence estimates. A confidence estimate reflects the degree of reliability for the class prediction of a given classifier. In contrast, the probability distributions used in the established methods for building combined classifiers represent the likelihood that an object o belongs to any of the possible classes. The difference becomes clear when considering the following two-class case. A classic probability estimate for class c_1 of 40 % implies that it would be better to predict class c_2 which must have correspondingly a probability estimate of 60 %. On the other hand, a confidence estimate of the class decision for class c_1 of 40 % only implies that the result of the classifier is rather unreliable. In multi-class problems, the difference can be seen more easily because there is only 1 confidence estimate for the class decision and not a separated probability estimation for each of the classes. In this paper, we argue that reliable confidence estimates are easier to derive and that a confidence estimate yields enough information for learning powerful multi-represented classifiers.

A second advantage of the proposed method is that confidence estimates can be derived from multiple types of classifiers such as Support Vector Machines (SVMs), Bayes classifiers, k-nearest-neighbor (kNN) classifiers and decision trees. Since the principle idea of deriving confidence estimates is the same for each of these classifiers, the confidence estimates are directly comparable even for different types of classifiers trained on varying feature spaces. This is not necessarily the case for probability estimation because the idea behind approximating the class probabilities is often quite different for various classifiers. Thus, the semantic of the probabilities is not necessarily comparable leading to suboptimal classification results.

To derive confidence estimates, we introduce the concept of the confidence range of a decision. The confidence range is the smallest range inside which the classified object could be moved in order to be assigned to a different class. In other words, the confidence range corresponds to the distance of an object to the closest class border. Therefore, deriving confidence estimates from SVMs can be done in a straightforward way. However, there still exist major differences between the probability estimation for a SVM as proposed in [1] and the confidence estimate employed in this paper. First of all, it is possible to derive a confidence estimate of less than 50 % for the predicted class, if it is quite uncertain that the prediction is correct. Additionally, the method proposed in [1] yields a solution for probability estimation in two class problems while our method using confidence estimates can be applied to an arbitrary number of classes. For employing other classifiers than SVMs, we will provide algorithms for several well-established classification methods like Bayes classifiers, decision trees, and kNN classifiers for deriving confidence ranges. Afterwards the confidence ranges are used to calculate confidence estimates which are finally used to find the global class decision. The main contributions of this paper are:

- A new method for combining classifiers based on the confidence estimates instead of complete distribution vectors.
- Methods for deriving confidence ranges for various classifiers such as decision trees, Bayes classifiers, or kNN classifiers.
- Methods for learning a function that derives confidence estimates from the derived confidence ranges.

Our experimental evaluation illustrate the capability of our new approach to improve the classification accuracy compared to combined classifiers that employ distribution vectors.

The rest of the paper is organized as follows. Section 2 surveys related work. In section 3, we introduce the general idea for our method of classifier combination. Afterwards, section 4 describes methods to derive confidence ranges for various classifiers and explains their use for deriving confidence estimates. The results of our experimental evaluation are shown in section 5. Section 6 concludes the paper with a summary and ideas for future work.

2 Related Work

In general, methods that employ multiple learners to solve a common classification problem are known as ensemble learning. An overview over ensemble learning techniques can be found in [2]. Within the area of ensemble learning our work deals with the subarea of classifier combination. The aim of classifier combination is to use multiple independently trained classifiers and combine their results to increase the classification accuracy in comparison to the accuracy of a single classifier. Combining classifiers to learn from objects given by multiple representations has recently drawn some attention in the pattern recognition community [3–5]. The authors of [3] developed a theoretical framework for combining classifiers which use multiple pattern representations. Furthermore, the authors propose several combination strategies like max, min, and, product rule. [4] describes so-called decision templates for combining multiple classifiers. The decision templates employ the similarity between classifier output matrices. In [5] the author proposes a method of classifier fusion to combine the results from multiple classifiers for one and the same object. Furthermore, [5] surveys the four basic combination methods and introduces a combined learner to derive combination rules increasing classification accuracy. All methods mentioned above assume that a classifier provides reliable values of the posteriori probabilities for all classes. Techniques for deriving probability estimates from various classifiers can be found in [1, 6]. Learning reliable probability estimates and measuring their quality is a rather difficult task, because the training sets are labeled with classes and not with class probability vectors. In contrast to these solutions, we propose a method that calculates a single confidence estimate reflecting the correctness of each particular class decision. A related subarea of ensemble learning is co-training or co-learning which assumes a semi-supervised setting. The classification step of co-training employs multiple independent learners in order to annotate unlabeled data. [7] and [8] were the first publications that reported an increase of classification accuracy by employing multiple representations. The most important difference of co-learning approaches to our new approach of multi-represented classification is that we do not consider a semi-supervised setting. Additionally, co-training retrains its classifiers within several iterations whereas the classifiers in our approach are only trained once. Recently, methods of hyper kernel learning [9] were introduced that are also capable of employing several representation in order to learn a classifier. In contrast to our method the hyper kernel learners optimize the use of several kernels that can be based on multiple representations within one complex optimization problem which is usually quite difficult to solve.

3 Confidence Based Multi-Represented Classification

In this section we will specify the given task of multi-represented classification and describe our new approach of using a single confidence value for each representation to derive global class decisions.

A multi-represented object o is given by an n-tuple $(o_1, \ldots, o_n) \in R_1 \times \ldots \times R_n$ of feature vectors drawn from various feature spaces $R_i = F_i \cup \{-\}$. F_i denotes the corresponding feature space of representation i and ”$-$” denotes that there is no object description for this representation. Missing representations are a quite common problem in many application areas and thus, should be considered when building a method. Obviously, for each object there has to be at least one $o_j \neq$ "$-$". For a given set of classes $C = c_i, \ldots, c_k$, our classification task can be described in the following way. Given a training set of multi-represented objects $TR \subset R_1 \times \ldots \times R_n$, we first of all train a classifier $CL_i : R_i \rightarrow C$ for each representation. Afterwards, we train a confidence estimator $CE_{CL_i} : R_i \rightarrow [0..1]$ based on a second training set TR_{conf} for each classifier which predicts the confidence of the class decision $CL_i(o_i)$ for each object o_i. Let us note that we employed cross validation for this second training since the number of training objects is usually limited. To combine these results, we employ the following combination method:

$$CL_{global}(o) = CL_{\underset{0 \leq j \leq n}{\operatorname{argmax}} \{CE_{CL_j(o)}\}}(o)$$

where o is an unknown data object.

In other words, we employ each classifier $CL_j(o)$ for deriving a class and afterwards determine the confidence of this class decision $CE_{CL_j}(o)$. As a global result, we predict the class c_i which was predicted by the classifier having the highest confidence estimate $CE_{CL_i}(o)$. To handle unknown representations, we define $CE_{CL_j}(-) = 0$. Thus a missing representation cannot have the highest confidence value.

4 Deriving Confidence Estimates

After describing our general pattern for multi-represented classification, we now turn to describing the method for deriving the confidence estimates. The main idea of our proposed confidence estimation is that the larger the area around an object for which the class prediction does not change, the larger is the confidence for the class decision. In other words, the more we can alter the characteristics of an object without changing its class, the more typical is this object for the given class in the given feature space. The confidence range can be determined by calculating the distance of the classified object o to the closest class border. Let us note that we can apply this idea regardless of the used classification method. To formalize the area around each object for which the class prediction remains unchanged, we define the confidence range of an object as follows:

Definition 1. *Let $o \in F$ be a feature vector and let $CL : F \rightarrow C$ be a classifier w.r.t. the class set C. Then the confidence range $CRange(o)$ is defined as follows:*

$$CRange(o) = \min \{\|v\| \; |v \in F \wedge CL(o) \neq CL(o + v)\}$$

The methods for deriving the confidence range are varying between the classification methods. For SVMs the method to extract a confidence range is straightforward. For the two-class case, we can use the output of the SVM as distance to the separating hyperplane. In the case of multi-class SVMs, the minimum distance to all of the used hyperplanes is considered. For other classification paradigms, the calculation is less straightforward. In general, the confidence range of an object o can be determined by taking the minimum distance for which the class prediction changes from class c_{pred} to some other class c_{other}. Thus, we can determine $CRange(o)$ by:

$$CRange(o) = \min_{c_{other} \in C \setminus c_{\{pred\}}} CRange_{c_{pred},c_{other}}(o)$$

where $CRange_{c_{pred},c_{other}}(o)$ is the distance of o to the class border between the predicted class c_{pred} and the class c_{other}. In the following, we will provide methods for approximating $CRange(o)$ for three well-established classification methods.

4.1 Bayes Classification

For a Bayes classifier over a feature space $F \subseteq \mathbb{R}^d$, each class c is determined by a prior $p(c)$ and a density function corresponding to $p(x|c)$ where $x \in F$. The class having the highest value for $p(c) \cdot p(x|c)$ is predicted. Thus, for each class c_{other}, we need to consider the following equation describing the class border between two classes:

$$\begin{aligned}
&p(c_{pred}) \cdot p(\boldsymbol{x}|c_{pred}) = p(c_{pred}) \cdot p(\boldsymbol{x}|c_{other}) \\
&\Rightarrow p(c_{pred}) \cdot p(\boldsymbol{x}|c_{pred}) - p(c_{pred}) \cdot p(\boldsymbol{x}|c_{other}) = 0
\end{aligned}$$

To determine the distance of an object o to this class border, we need to solve the following optimization problem:

$$\begin{aligned}
&\min_{x \in \mathbb{R}^d} d(o, x) \\
&s.t.\;\; p(c_{pred}) \cdot p(\boldsymbol{x}|c_{pred}) - p(c_{pred}) \cdot p(\boldsymbol{x}|c_{other}) = 0
\end{aligned}$$

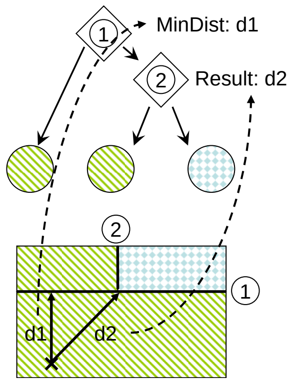
For example, the optimization problem for a general Gaussian distribution can be formulated as follows:

$$\begin{aligned}
&\min_{x \in \mathbb{R}^d} d(o, x) \\
&s.t.(x - \mu_1)^T \times (\Sigma_1)^{-1} \times (x - \mu_1) \\
&-(x - \mu_2)^T \times (\Sigma_2)^{-1} \times (x - \mu_2) - \ln \frac{p(c_1) \cdot \Sigma_2}{p(c_2) \cdot \Sigma_1} = 0
\end{aligned}$$

To solve this problem, we employed a gradient descent approach which is an iterative method for solving non linear optimization problems. Beginning at an initialized point, the direction of the steepest descent is determined. Then, a step in this direction is made whereas the step size is calculated by applying the Cauchy principle. The steps are repeated until the minimum is reached which usually occurs after a small number of iterations.

4.2 Decision Trees

For most decision trees, each node in the tree belongs to some discriminative function separating the training instances with respect to a single dimension of the feature space.




```
calculateCRange(Object o, Tree decisionTree)
  queue=getNewPriorityQueue(orderAscending);
  /*classify o, store omitted subtrees in queue with their MinDist to o */
  classID:=classify(o,decisionTree, queue);
  LOOP
    current=queue.removeFirst();
    if (current.isLeaf() AND current.getClass()!= classID) then
      return current.getMinDist();
     else
      foreach subtree ∈ current.getSubtrees()
        queue.insert(subtree, subtree.getMinDistTo(o));
     end if;
  end LOOP;
```

Fig. 1. Example and pseudo code for determining CRange for decision trees.

Therefore, each leaf of a decision tree corresponds to a hyper rectangle. However, to determine $CRange(o)$, it is not sufficient to calculate the minimum distance of o to the border of this hyper rectangle. If the neighboring leaf does correspond to the same class, the border of the leaf does not provide a class border. Therefore, for determining $CRange(o)$, we need to find the distance of object o to the closest leaf belonging to any other class than c_{pred}. To find this leaf node while traversing the tree for classification, we collect all subtrees that do not contain o, i.e. the subtrees that are omitted during the classification of o. Each of these subtrees is stored in a priority queue which is ordered by the minimum distance of o to any node of the subtree. After the classification algorithm reaches a leaf node (i.e. o is classified), we can process the priority queue containing the collected subtrees. If the first object in the queue is a leaf node, we determine whether the leaf corresponds to a class which is different to the class prediction. In this case, we can calculate $CRange(o)$ as the distance of o to the boundaries of this leaf. Since the priority queue is ordered by the minimum distance to o, there cannot exist any other leaf with a boundary closer to o than the calculated $CRange(o)$. If the top element of the queue is a subtree, we remove the tree from the queue and insert its descendants. Figure 1 illustrates an example for a small decision tree on the left side. The right side of Figure 1 describes the proposed algorithm in pseudo code. Let us note that calculating the minimum distance of o to any leaf node in the tree can be done by only considering the attributes which are encountered while traversing the tree. For each other attribute, the feature value of o must be contained within the range of the tree node.

4.3 *k*NN Classification

Though it is sufficient for finding an accurate class decision, determining the k nearest neighbors for an object o is not enough to determine $CRange(o)$. Since the k nearest neighbors do not necessarily contain representative objects of each of the classes, finding the class border for a kNN classifier needs to consider additional objects belonging to each of the classes. If the number of considered neighbors is one, the class borders are described by Voronoi cells around the training objects. In this case, we can easily calculate $CRange(o)$ on the basis of the particular $CRange_{c_{pred},c_{other}}(o)$. Thus, we only need to compare the distances to the class border which is determined by the nearest neighbor u_c of the predicted class c to any nearest neighbor $u_{\hat{c}}$ of the other classes $\hat{c}$. This distance can be calculated using the following lemma.

Lemma 1. *Let o be an object, let u_c be the nearest neighbor belonging to class $CL(o) = c$ and let u_{other} be the nearest object belonging to some other class $other \in C \setminus c$. Furthermore, let $d(x_1, x_2)$ be the Euclidian distance in $\mathbb{R}^d$. Then, $CRange_{c,other}(o)$ for a nearest neighbor classifier can be calculated as follows:*

$$CRange_{c,other}(o) = \frac{d(u_c, u_{other})^2 + d(u_c, o)^2 - d(u_{other}, o)^2}{2d(u_c, u_{other})} - \frac{d(u_c, u_{other})}{2}$$

A proof for this lemma can be found in [10].

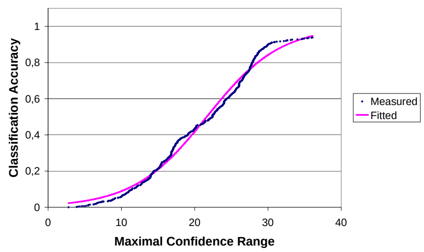
Unfortunately, $CRange_{c,other}(o)$ is much more complicated to calculate for $k > 1$ because this would require to calculate Voronoi cells of the order k. Since this would cause a very time consuming calculations, we propose a heuristic method to approximate $CRange(o)$ for the case of $k > 1$. The idea of our heuristic is to determine the set U_c^k consisting of the k closest objects for each class c. Note that the union of these sets obviously contains the k nearest neighbors as well. For each class, we now build the weighted centroid. The weights for determining the centroid are derived by the inverse squared distance to the classified object o, in order to mirror the well-known weighted decision rule for kNN classifiers. Formally, these class representatives are defined as follows:

$$Rep_c^k(o) = \sum_{u_i \in U_c^k} \frac{1}{d(o, u_i)^2} \cdot u_i \cdot \frac{1}{\sum_{u_i \in U_c^k} \frac{1}{d(o,u_i)^2}}$$

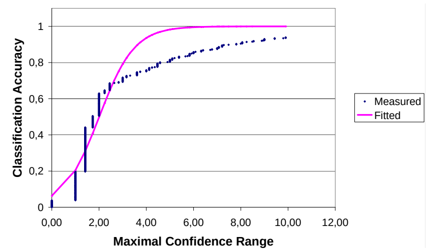
After having calculated a representative for each class, we can proceed as in the case for $k = 1$. Let us note that using this heuristic, an object might have a negative distance to the class if it is placed on the wrong side of the estimated class border. However, this only occurs if the distance to the border is rather small and thus, the class decision is more or less unreliable.

4.4 From Ranges to Confidence Estimates

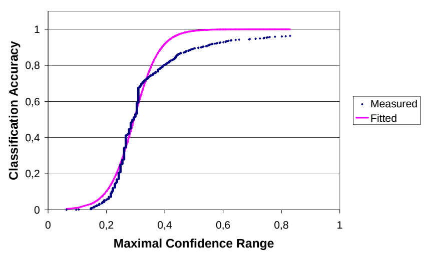
Our goal is to compare the results of various classifiers trained on varying feature spaces and employing various classification paradigms. However, the derived confidence ranges do not represent a comparable and accurate confidence estimate so far. To transform the confidence ranges into usable confidence estimates, we must cope with the following two problems. First the confidence ranges are distances in different feature spaces and thus, a comparably small distance in R_1 might induce a much higher confidence than a larger confidence range in representation R_2. Obviously, we have to learn which confidence range induces a high likelihood for a correct class prediction. A second problem we have to cope with is noise. In a noisy representation, an object having a comparably high confidence range might still be classified with comparably



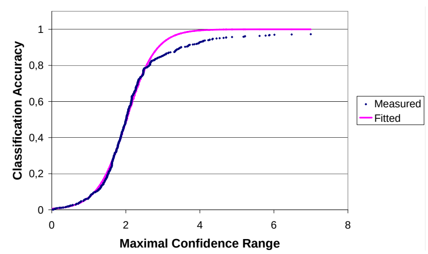
(a) Naive Bayes



(b) Decision Tree



(c) kNN



(d) SVM

Fig. 2. Relationship between confidence range and classification accuracy.

low confidence. Therefore, the confidence estimates should mirror the global reliability of the classifier as well as the confidence range of the individual object.

In order to understand the relationship between confidence ranges and the reliability of the class prediction, we performed several experiments. First, we partitioned the training data of each representation into three folds. We used two folds to train the classifier and used the remaining fold, the test fold, for calculating the confidence ranges. Afterwards, we formed a curve expressing the accuracy of the class predictions for all objects having a smaller $CRange(o)$ then the given x-value in the graph. More precisely, for each test object o a set of objects was determined containing all objects u in the test fold for which $CRange(u) \leq CRange(o)$. Then, the classification accuracy for each of these subsets was calculated providing the y-value for the x-value $CRange(o)$. We observed small classification accuracies for subsets having a small maximal confidence range. The accuracy is improved with increasing maximal confidence range values and finally reaches the accuracy observed on the complete test set. Furthermore, the graph displayed a sigmoidal pattern, i.e. there is a range of values where a small increase of the confidence ranges results in a high increase of classification accuracy. The results of the above described experiments are presented in Figure 2. The curve labeled with 'Measured' corresponds to these observed accuracy values while the curve labeled with 'Fitted' displays the function we will introduce in the following to model

Table 1. Description of the data sets.

	DS1	DS2	DS3	DS4	DS5	DS6	DS7
Name	Oxidore-ductase	Trans-ferase	Transporter Activity	Protein Binding	Enzyme Regularization	Sonar	Wave 5000
No. of Classes	20	37	113	63	19	2	3
No. of Objects	1051	2466	2218	4264	1046	208	5000

this behavior. As can be seen in Figure 2, the measured values form a sigmoidal shape for all examined classification techniques.

Based on these observations we developed a technique to calculate confidence estimates for each classifier. These confidence estimates range from 0 to 1 and thus, unlike confidence ranges, the confidence estimates are directly comparable to each other. Since the confidence estimates cannot become larger than the classification accuracy on the complete test set, the noise level in each representation is considered as well. In the following, the calculation of the confidence estimates is described in detail.

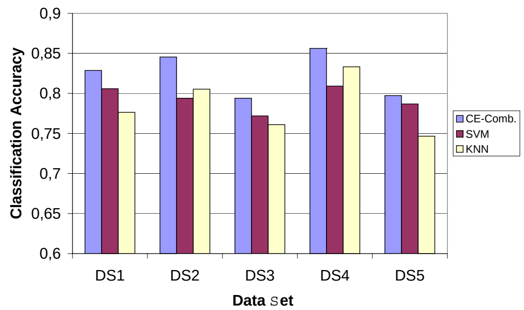
1. For a given classifier CL_j, perform a 3-fold cross validation with the training data in order to yield confidence range/accuracy pairs as described above.
2. A suitable optimization algorithm (e.g. Levenberg-Marquardt algorithm [11]) is used to determine the parameters α_j and β_j that minimize the least squares error for the sigmoid target function $accuracy_j(o) = \frac{1}{1+exp(\alpha_j \times CRange_j(o)+\beta_j)}$ given the observed pairs of confidence ranges and classification accuracy.
3. For classifier CL_j and object o the confidence estimate $CE_{CL_j}(o)$ can finally be calculated as:

$$CE_{CL_j}(o) = \frac{1}{1 + exp(\alpha_j \times CRange_j(o) + \beta_j)}$$

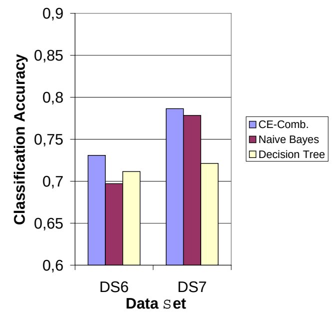
The derived confidence estimates are now used for classifier combination as described in section 3, i.e. the classification result based on the representation yielding the highest confidence estimate is used as the global prediction of the combined classifier.

5 Experimental Evaluation

For our experimental evaluation, we implemented our new method for classifier combination and confidence estimation in Java 1.5. All experiments were performed on a workstation featuring two Opteron processors and 8 GB main memory. For comparing our method to the established methods for classifier combination as described in [5], we used the J48 (decision tree), Naive Bayes, SMO (SVM), and IBK (kNN classifier) classifiers provided by the WEKA machine learning package [12]. The WEKA implementation provides probabilities for each of the classes which were combined using the minimum, the maximum, the product, and average. For example, a joined confidence vector v is constructed by taking the average class probability for each class c_i over all representation R_j as ith component v_i. For our confidence estimates we used the same classifiers and additionally implemented our new method. For fitting the sigmoid functions we used the method introduced in [11].



(a) Protein data sets



(b) UCI data sets

Fig. 3. Classification accuracy of CE-Comb. compared to separated classification.

We tested our new ensemble method on 5 multi-represented data sets describing proteins (DS1-DS5). The 5 test beds consist of 19 to 113 Gene Ontology [13] classes. The corresponding objects were taken from the SWISS-PROT [14] protein database and consist of a text annotation and the amino acid sequence of the proteins. In order to obtain a flat class-system with sufficient training objects per class, the original data was adapted. We employed the approach described in [15] to extract features from the amino acid sequences. The basic idea is to use local (20 amino acids) and global (6 exchange groups) characteristics of a protein sequence. To construct a meaningful feature space, we formed all possible 2-grams for each kind of characteristic, which yielded the 436 dimensions of our sequence feature space. For text descriptions, we employed a TFIDF [16] vector for each description that was built of 100 extracted terms. Since these data sets could only be tackled by the more flexible SVMs and kNN classifiers, we could not test the Naive Bayes and the J48 decision tree classifiers on these problems. Therefore, by splitting the wave-5000 (DS7) and the sonar data set (DS6) from the well-known UCI machine learning repository [17] vertically, we generated two additional multi-represented. Thus, we derived for each data set two representations containing only half of the attributes of the original data set. An overview of our 7 test beds is given in Table 1.

For our experiments, we first of all classified each representation separately using several classifiers. Afterwards, we combined the best classification method for each representation using our new method based on confidence estimates (CE-Comb.) and the 4 standard methods mentioned above which are based on probability vectors. For the UCI data sets, we tested only Naive Bayes and J48 because these data sets were chosen to provide an example for these two types of classifiers.

Our first result compares the classification accuracy of our new combination method with the classification accuracy achieved in the separated representations. Figure 3 displays the achieved classification accuracies. In all 7 data sets our new combination method achieved a higher classification accuracy than both corresponding classifiers which rely on only a single representation. Thus, using the additional information of both representations was always beneficial.

Table 2. Accuracies for different combination methods.

	DS 1	DS 2	DS 3	DS 4	DS 5	DS 6	DS 7
CE-Comb.	**0.8287**	**0.8454**	**0.7939**	**0.8562**	**0.7973**	**0.7692**	**0.7954**
Product	0.7761	0.8092	0.7633	0.8302	0.7514	0.7307	0.7794
Sum	0.8072	0.8051	0.7768	0.8142	0.7877	0.7307	0.7808
Min	0.7761	0.8053	0.7610	0.8332	0.7466	0.7307	0.7786
Max	0.8058	0.7939	0.7718	0.8090	0.7868	0.7307	0.7806

In Table 2 we compare our new method to the established methods of classifier combination. The accuracies which where achieved using our confidence estimates are shown in the first row. For all data sets our new method for classifier combination outperforms the established approaches. The improvement by using our new combination method was up to 4 % in data sets DS3 and DS6. Let us note that the classifiers in each representation for all types of ensembles were always exactly the same. Thus, the improvement is achieved by the different combination method only.

Table 3. Comparing various combinations.

	DS 1	DS 2	DS 3	DS 4	DS 5
kNN+SVM	0.8116	0.8215	0.7831	0.8485	0.7782
SVM+kNN	**0.8287**	**0.8454**	**0.7939**	**0.8562**	**0.79732**
SVM+SVM	0.8097	0.836	0.7921	0.8410	0.7906
kNN+kNN	0.789	0.8215	0.7889	0.8499	0.7667

Our final result illustrates that the capability to combine classifiers of different types proves to be beneficial on real world data sets. We tested all possible combinations of SVMs and kNN classifiers for the 5 protein data sets. The results are displayed in Table 3. For all data sets, the combination of using a linear SVM for text classification and a nearest neighbor classifier for sequence classification proved to yield the best accuracy. Thus, our new method is capable of exploiting different types of classifiers which often yields a better ensemble than using only classifiers of one and the same type.

6 Conclusions

In this paper we describe a new method for classifier combination. The two main aspects of our new approach are the following. First of all, the global class decision is not dependent on complete probability distributions over all classes but depends only on a confidence estimate that the given classification result is indeed correct. The second aspect is that the used confidence estimates are not limited to a particular type of classifier. By introducing the general concept of confidence ranges, it is possible to generate comparable confidence estimates for different types of classifiers and varying feature spaces. To derive these confidence estimates, we provide algorithms to calculate confidence ranges for kNN classifiers, decision trees, and Bayes classifiers. The confidence ranges are transformed into meaningful confidence estimates using a trained sigmoid function. Our experimental evaluation shows that our new method is capable of outperforming established methods based on probability vectors. Additionally, we observed that it is sometimes indeed useful to use different types of classifiers for classifying different representations.

In our future work, we are going to extend our new idea for classifier combination to other methods of ensemble learning like co-training. Measuring the agreement between the used classifiers by means of our new confidence estimates might yield an improvement compared to established methods using probability vectors.

References

1. Platt, J.: "Probabilistic outputs for support vector machines and comparison to regularized likelihood methods". In: Advances in Large Margin Classifiers, MIT Press. (1999) 61–74
2. Valentini, G., Masulli, F.: "Ensembles of learning machines". Neural Nets WIRN (2002)
3. Kittler, J., Hatef, M., Duin, R., Matas, J.: "On Combining Classifiers". IEEE Transactions on Pattern Analysis and Machine Intelligence **20** (1998) 226–239
4. Kuncheva, L., Bezdek, J., Duin, R.: "Decision Templates for Multiple Classifier Fusion: an Experimental Comparison". Pattern Recognition **34** (2001) 299–314
5. Duin, R.: "The Combining Classifier: To Train Or Not To Train?". In: Proc. 16th Int. Conf. on Pattern Recognition (ICPR'02), Quebec City, Canada). (2002) 765–770
6. Zadrozny, B., Elkan, C.: "Obtaining calibrated probability estimates from decision trees and naive Bayesian classifiers". In: Proc. 18th Int. Conf. on Machine Learning, San Francisco, CA. (2001) 609–616
7. Yarowsky, D.: Unsupervised word sense disambiguation rivaling supervised methods. In: ACL. (1995) 189–196
8. Blum, A., Mitchell, T.: "Combining labeled and unlabeled data with co-training". In: Proc. of the eleventh annual conference on Computational learning theory (COLT '98), New York, NY, USA (1998) 92–100
9. Ong, C.S., Smalo, A.: "Machine Learning with Hyperkernels". In: Proc. of the 20th Int. Conf. (ICML 2003), Washington, DC, USA. (2003) 576–583
10. Kriegel, H.P., Schubert, M.: "Advanced Prototype Machines: Exploring Prototypes for classification". In: in Proc. 6th SIAM Conf. on Data Mining, Bethesda, MD. USA. (2006) 176–188
11. Levenberg, K.: "A method for the solution of certain problems in least squares". Quart. Appl. Math. **2** (1944) 164–168
12. Witten, I., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques with Jave Implementations. Morgan Kaufmann (1999)
13. The Gene Ontology Consortium: "Gene Ontology: Tool for the Unification of Biology". Nature Genetics **25** (2000) 25–29
14. Boeckmann, B., Bairoch, A., Apweiler, R., Blatter, M.C., Estreicher, A., Gasteiger, E., Martin, M., Michoud, K., O'Donovan, C., Phan, I., Pilbout, S., Schneider, M.: "The SWISS-PROT Protein Knowledgebase and its Supplement TrEMBL in 2003". Nucleic Acid Research **31** (2003) 365–370
15. Deshpande, M., Karypis, G.: "Evaluation of Techniques for Classifying Biological Sequences". In: Proc. of the 6th Pacific-Asia Conf. on Advances in Knowledge Discovery and Data Mining (PAKDD '02), London, UK (2002) 417–431
16. Salton, G.: Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer. Addison-Wesley (1989)
17. University of Irvine. http://www.ics.uci.edu/ mlearn/MLRepository.html, UCI Machine Learning Repository. (2005)