On the Combination of Logistic Regression and Local Probability Estimates

Melanie Osl[1], Christian Baumgartner[1], Bernhard Tilg[1], and Stephan Dreiseitl[1,2]
[1]*Institute of Biomedical Engineering, University for Health Sciences, Medical Informatics and Technology, 6060 Hall in Tyrol, Austria*
[2]*Department of Software Engineering, Upper Austria University of Applied Sciences, 4232 Hagenberg, Austria*

Abstract—In this paper we give a survey of the combination of classifiers. We briefly describe basic principles of machine learning and the problem of classifier construction and review several approaches to generate different classifiers as well as established methods to combine different classifiers. Then, we introduce our novel approach to assess the appropriateness of different classifiers based on their characteristics for each test point individually.

***Index Terms*—classifier characteristics, classifier combination, ensemble classification, k-nearest neighbor classification, logistic regression, machine learning**

I. INTRODUCTION

The field of machine learning seeks to develop methods to learn from data. Therefore, computational and statistical methods are applied, which connects machine learning closely to statistics and theoretical computer science. Machine learning is divided into supervised and unsupervised learning. In supervised learning, a set of data points with known prediction values is given. The objective is to model the conditional distribution of the prediction values given the data points in order to make predictions about future data. According to the characteristics of the predicted values, supervised learning can be further classified into regression and classification. In regression, the prediction values are continuous. An example for a regression task is to predict a patient’s blood pressure based on clinical data and lab results. In classification, the predicted values are distinct classes. Examples for classification tasks are spam filtering, medical decision support, or fraud detection [1]. If no prediction value is given in the data, unsupervised learning methods are applied with the objective of grouping data points according to similar characteristics, e.g. to identify groups of interacting genes.

In the following we concentrate on classification, especially on two-class classification problems, as illustrated in Figure 1.

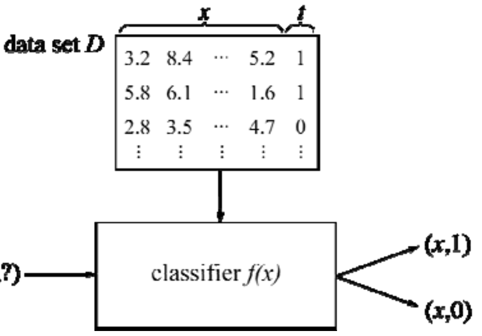

Fig. 1. Illustration of a classifier system (*t* is a class label, *x* is a vector of attribute values).

Formally, the given data set D consists of data points (x,t), where x is an n-dimensional vector of attributes and $t \in \{0,1\}$ is a class label. From this data set a model $f(x)$ of the conditional probability $P(t|x)$ is learned. The function $f(x)$ is also called classifier. In order to classify an unseen data point x_k the classifier computes $f(x_k)$ and assigns the data point to class $t_k = 0$ if $f(x_k) \geq 0.5$ and $t_k = 1$ if $f(x_k) < 0.5$. Some of the most popular classifiers are logistic regression, classification trees, artificial neural networks, k-nearest neighbors and support vector machines.

Logistic regression – Logistic regression searches for an optimal linear separation between two classes. It models the posterior probabilities of the classes in terms of a logit transformation to ensure that the probabilities sum up to one and remain in the range of 0 to 1. The parameters of a logistic regression are optimally adjusted by maximum likelihood [2].

Classification tree – Classification trees classify data points by sorting them down a tree where each node corresponds to a test on some attribute of the data point, and each branch corresponds to one of the possible values of this attribute. To build up the tree, all attributes are evaluated using a statistical test. The one which classifies the data set best is chosen to be placed at the root node. A descendant node for each possible value of the attribute is created, and the data points are sorted in. The process is repeated for each descendant node using the data points associated with this node [3].

Artificial neural network – Artificial neural networks consist of a set of artificial neurons organized in layers. The neurons of one layer serve as input for the neurons of the next layer and are therefore connected to each other. Each neuron takes a vector of weighted inputs, calculates a linear combination of these inputs and outputs a logistic transformation of this linear combination. The weights are adapted to minimize the deviation of the final output of the network to the expected output of the network [4].

Support vector machine – Support vector machines extent the solution of an optimal linear separation problem to the situation where classes are not linearly separable. It produces nonlinear decision boundaries by constructing a linear boundary in a transformed attribute space. An optimal separation is found by maximizing the so-called margin, the largest possible distance between the separation hyperplane and the data points on either side. The points that lie on the margin are called support vectors and represent, as a linear combination, the solution of a convex optimization problem [5].

k-nearest neighbor – k-nearest neighbor classifiers search the training set for the k closest neighbors to a given test point according to a certain distance measure (e.g. Euclidean distance). Commonly, the majority class of the neighbors is assigned to the test point, but also a probability estimate given by the frequency of the test points for which $t_k = 1$ can be returned [6].

A. Classifier Evaluation

To assess the quality of a classifier, a common approach is to split the data set into a training set and a test set. The training set is used to build the classifier. Then the classifier is evaluated by measuring the percentage of correctly classified test points, referred to as accuracy of the classifier. However, this approach is not feasible for small data sets. To overcome this problem, the process of training and testing is repeated on different splits of the data. In k-fold cross-validation the data set is split into k parts, where k times a different part is used for testing and the remaining k-1 parts are used for training. The k accuracies obtained for the k folds are finally averaged. If the number of folds is chosen so that the test set of each fold consists of only one test point, the validation procedure is called leave-one-out cross-validation.

However, for data sets with highly skewed class distributions, accuracy was shown to be an inappropriate quality measure [7]. For example, a two class data set with 70 data points in one class and 30 data points in the other class would be classified with an accuracy of 70 percent by an uninformed classifier that predicts only based on class prevalence. An alternative measure to accuracy for assessing the quality of a classifier is the area under the ROC curve (AUC). The AUC for the example above is 0.5, which corresponds to the true situation that the classifier assigns classes to test points only randomly. Thus, AUC has found widespread use as the measure of choice for evaluating classifier performance in the machine learning literature [8].

Given m classifier outputs a_i for data points of class 0, and n classifier outputs b_j for data points of class 1, the AUC estimate $\hat{\theta}$ of the classifier can be shown to be equivalent to

$$\hat{\theta} = \frac{1}{m \cdot n} \sum_{i=1}^{n} \sum_{j=1}^{m} 1_{a_i < b_j} \qquad (1)$$

where $1_{(\cdot)}$ denotes the boolean indicator function that returns one if its argument is true, and zero otherwise [9]. It is immediately obvious that $\hat{\theta}$ is an unbiased estimator of the parameter $\theta = P(A < B)$, the probability that a randomly chosen element A of class 0 is ranked lower than a randomly chosen element B of class 1. The value of $\hat{\theta}$ can thus be used as an assessment of a classifier's discriminatory power (how well it can separate two classes). A perfect classifier has $\hat{\theta} = 1$, indicating that there exists a threshold along which both classes can be separated without error. An uninformed classifier that performs no better than chance has $\hat{\theta} = 0.5$.

B. Ensemble Classification

For the data set at hand, traditionally the best classifier is selected on the basis of a quality evaluation. Another possibility is to combine the collected classifiers to construct a classifier that performs better than any of the base classifiers [10,11]. Therefore the base classifiers have to be diverse but also comparable.

A consistent ensemble of classifiers can be generated e.g. by different initializations, different parameter choices or different architectures of the same classifier. More successful due to the fact that the classifiers are independent of each other is to generate different classifiers from different training sets. Well known examples are bagging [12] and boosting [13,14]. Bootstrap aggregating (short "bagging") creates n times a set of m data points by sampling uniformly with replacement from the original data set. Thus some data points may be included in the sample multiple times whereas other data points do not appear at all. In boosting, the single training sets differ systematically as the weights for each data points are changed based on the previous classification. Another way to force diversity is to build a set of classifiers on different sets of features [15,16], e.g. speech and image features in identification problems. Manipulating the output of the training data is particularly useful if the number of output classes is large. Then new learning problems can be constructed by randomly dividing the output classes into a smaller number of subsets of classes [17]. Of course a combination of different classifiers trained on the same features and by the same training set is also possible.

Traditionally the results of the different base classifiers, which can be class labels or class posterior probabilities, are combined by combinatorial functions. The majority voting is the simplest of all combinatorial functions. The class which is most often predicted by the base classifiers is selected. Using functions like sum, product, average or median to combine the results of the base classifiers presumes that the base classifiers are independent of each other, which is hardly ever the case. However, if an ensemble consists of similar base classifiers with independent noise behavior, the errors are averaged out by the summation. Statistical combination methods, e.g. Bayesian combination [18], are another possibility to build an ensemble classifier. Finally, learning a meta-classifier based on the outputs of different classifiers, called stacking [19], can also be seen as a decision logic for a multi-classifier system.

Beside training diverse base classifiers in parallel and combining them afterwards, a multi-classifier system can also be built of cascading base classifiers or hierarchically. Cascading classifiers pass the classification results from classifiers to classifiers (which take them as input) until the final output is obtained through the final classifier in the chain. However, the major disadvantage of this approach is that the later classifiers are unable to correct mistakes made by the earlier classifiers. The most prominent hierarchical classifier systems are decision trees. A particular advantage of hierarchical classifier systems is that they allow to introduce error checking.

Like experts in a committee, also classifiers in an ensemble have different areas of expertise. By assigning equal weights to each of them at combination these differences in skills are neglected. Therefore, weighted combinations methods are used, where the weights reflect the importance or significance assigned to the classifier. A more objective measure reflecting the confidence into a classifier is its quality found by an independent evaluation set.

C. Classifier Characteristics

In contrast to the methods mentioned above, our approach assesses the confidence into different classifiers based on their characteristics and for each test point individually. Thus the combination of the classifiers emphasizes their strengths while diminishing their weaknesses. We elaborate this concept to combine logistic regression and *k*-nearest neighbor classification.

Logistic regression (LR) models have a long history as the primary tool for supervised classification problems [20,21,22]. For an *n*-dimensional data set containing two classes, the logistic regression model depends on an *n*-dimensional parameter vector β and a scalar β_0 in the functional form

$$P(t=1 \mid x, \beta, \beta_0) = \frac{1}{1+e^{-(\beta \cdot x + \beta_0)}} \qquad (2)$$

Here, $x \in \mathbb{R}^n$ is alternatively called covariate vector or input vector, and $t \in \{0,1\}$ is the class label. The LR model thus provides the probability that a given vector x belongs to class 1. LR models are examples of discriminative models, because they offer a functional representation of a discriminatory line—e.g., where $P(t = 1| x,\beta,\beta_0) = 0.5$—that separates the two classes in a data set. The LR model output for a data point x depends on the distance of x from the discriminatory line: data points x that are far from the discriminatory line have model outputs close to zero (for x on one side of the line) and close to one (for x on the other side). Data points close to the discriminatory line have outputs close to 0.5. Generally, the parameters of an LR model are estimated by maximum likelihood, i.e., by minimizing the negative log likelihood

$$\sum_{k=1}^{m} \log(1+e^{(-2t_k+1)(\beta \cdot x_k + \beta_0)}) \qquad (3)$$

for an *m*-element data set of input/label pairs (x_k,t_k). LR models are linear in the parameters, and can thus only model separating $(n-1)$-dimensional hyperplanes in *n*-space.

With the same notation as above, a *k*-nearest neighbor (*k*NN) classifier output can be calculated as

$$P(t=1 \mid \{x_i, t_i\}) = \frac{1}{k}\sum_{j=i_1}^{i_k} t_j \qquad (4)$$

where $i_1,\ldots,i_k$ denote the indices of the k points in $x_1,\ldots,x_m$ that are closest (usually in Euclidean distance) to the given data point x. The probability estimate for x is thus given by the local posterior probability in the vicinity of x.

LR models occupy one end of the spectrum spanned by the bias variance tradeoff [23]: Due to their linear nature, LR models have high bias, but little variance, compared to other (nonlinear) machine learning algorithms. On the other end of the spectrum, we can find so-called memory-based classifiers such as *k*-nearest neighbors. These classifiers are not models in the strict sense of the word, because they do not build a functional or probabilistic representation of the data. For *k*NN, the data is the model, implying that there is high variability (and low bias) between different (finite) data samples drawn from the same data distribution. Thus, these two classification methodology complement each other well and combine the advantages of both rigid model structure (by the LR component) and local flexibility (by the *k*NN component) in one classification structure. The goal of this combination is to obtain better classification performance.

It is important to note that the appropriateness of LR and *k*NN depends on the local topology around the data point to be classified. An example for two different degrees of appropriateness for LR and *k*NN is shown in Figure 2(a) and 2(b), respectively. In Figure 2(a) it can be seen that there are several possibilities to separate the two classes, depicted by x and o. The different discriminatory lines are illustrated as dashed lines. A data point at the position of the solid dot near the center of the figure has a similar prediction value for each discriminatory line. In contrast, the prediction for the solid dot in the lower third of the figure differs for the different discriminatory lines. Hence, the application of LR is more appropriate for the upper data point than for the lower one. How appropriate the application of *k*NN for a data point is depends on the distance to its *k* nearest neighbors. In Figure 2(b) the *k* nearest neighbors (in this example is $k = 4$) of the data point at the position of the solid dot near the center of the figure are closer than the *k* nearest neighbors of the data point at the position of the solid dot in the upper left corner of the figure. Hence, the application of *k*NN is more appropriate for the lower data point than for the upper data point.

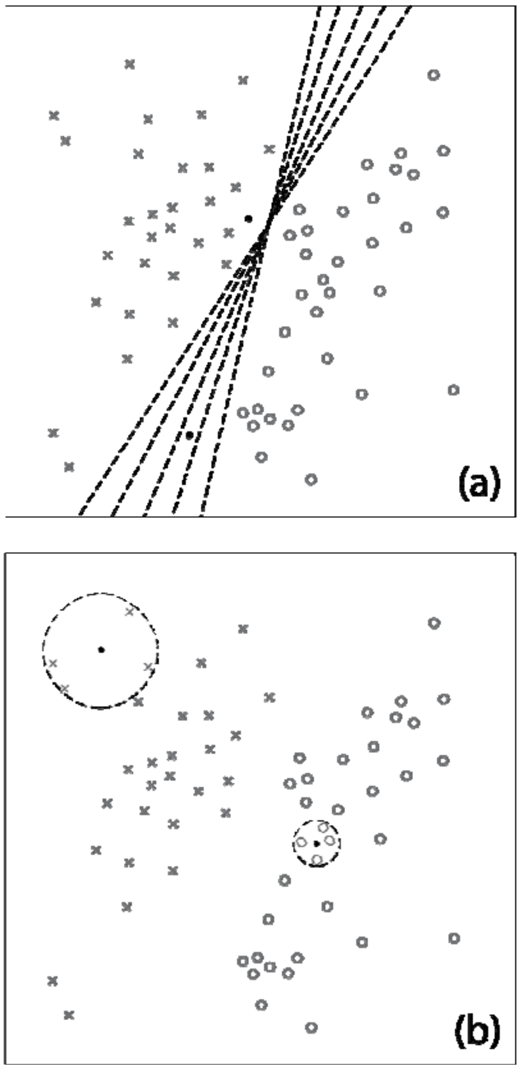

Fig. 2. Illustration of the topological dependency of LR (a) and kNN (b) appropriateness.

Our approach to combining LR and *k*NN models is thus by individually weighting the contributions of both models for each data point. We assign a larger factor to the model we consider more appropriate for a data point, and a smaller factor to the other model. The methodology to assess the appropriateness of a model for a given data points is different for LR and *k*NN.

II. CLASSIFIER COMBINATION

For LR, given a training set, we generate a subset of this training set containing 90% of the data points, and train an LR model on this reduced set. We do this 10 times for a total of 10 LR models that are all slightly different. We then generate the output of each of these LR models on every element of the training set, thus obtaining a standard deviation of model outputs for every data point. Although these standard deviations are likely smaller than those obtained from an independent test set, their average can nevertheless serve as a benchmark that allows us to judge how appropriate the LR models are. The reasoning is that a data point with large standard deviation over all LR models is one for which there is large variability between the models (high variance), and the model output should not be relied upon as much as for a data point with lower variance. The mean over all standard deviations on the training set is used to provide a scale information, as otherwise there is no way to know what constitutes a large or small standard deviation, respectively.

The appropriateness of *k*NN probability estimates is based on the distances of the *k* nearest neighbors to a data point. If these distances are large, we consider the probability estimate to be less reliable than if the distances are small. This is because for larger distances, the neighbors are not as close, and the *k*NN output is not as good an estimate of the local probability as in the case when the neighbors are in the immediate vicinity of the data point. To obtain an average distance value that can serve as a benchmark in the same way as the mean standard deviation for LR models, we calculate the distance of all data points in the training set to their *k* nearest neighbors. We then take the average of this distribution of distances, and judge the appropriateness of the *k*NN contribution by relating the *k*NN distance of a particular data point to the mean distance.

The precise manner in which LR and *k*NN estimates are combined is as follows: Let $\bar{\sigma}$ denote the mean standard deviation of LR model outputs and $\bar{d}$ the mean distances in the *k*NN part, both as described above. Also, for the *i*th data point in the test set, let lr_i denote the LR model output (the mean of all 10 trained models), σ_i the standard deviation of the 10 values, nn_i the *k*NN probability estimate, and d_i the distance to the *k* nearest neighbors. We measure the (in)appropriateness of the two components LR and *k*NN for this data point as

$$app_{LR} = \frac{\sigma_i}{\bar{\sigma}} \quad \text{and} \quad app_{kNN} = \frac{d_i}{\bar{d}} \qquad (5)$$

respectively. Note that a high value of one of these parameters means that the respective model is not appropriate, as the point displays above average standard deviation or distance. The contribution of each model is then weighted with the relative inappropriateness of the other model: This means that a point for which the LR output is highly inappropriate will assign most of its weight to the *k*NN component, and vice versa.

The heuristic to calculate a posterior class membership probability by combining all these pieces of information is

$$P(t=1 \mid x_i, lr_i, nn_i, app_{LR}, app_{kNN}) = \frac{app_{kNN}}{app_{LR} + app_{kNN}} lr_i + \frac{app_{LR}}{app_{LR} + app_{kNN}} nn_i \qquad (6)$$

Note that by using a convex combination of the two model contributions, we again obtain a probability estimate that is in the range of 0 to 1.

III. EXPERIMENTAL SETUP

We validated our approach on two different data sets: one synthetically generated data set that allows us to graphically demonstrate the feasibility of our method, and one real-world data set containing clinical information about patients with pigmented skin lesions.

Synthetic data - The synthetic data set, of which the training set is depicted in Figure 3, consists of two samples drawn from two different Gaussian mixture models with different class means and covariance matrices. Class 0 comprises 400 data points shown as o; class 1 consists of 600 data points marked by x. The dashed line shows the class separation by the logistic regression model at a threshold of p = 0.5. The two highlighted o points are used to provide, in Section IV, two examples of how our algorithm weights the contributions of LR and *k*NN differently, based upon our assessment of the appropriateness of the two components.

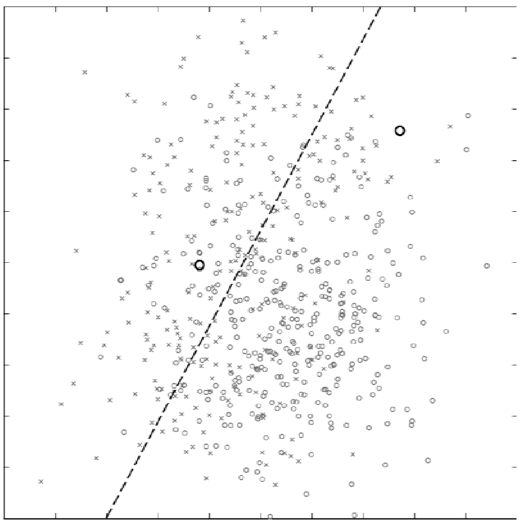

Fig. 3. Illustration of the synthetic data set.

Melanoma data - The real-world data set was collected at the pigmented lesion unit of the Department of Dermatology at the Medical University of Vienna, Austria. It is divided into the three classes common naevi, dysplastic naevi, and melanomas. The class distribution (gold standard) is 1290 patients for which a suspicious lesion was diagnosed as a common naevus, 224 patients with dysplastic naevi, and 105 patients with melanomas. For each of the 1619 patients, five clinical data items were recorded: the skin type according to Fitzpatrick, personal and family history with regard to melanoma, whole body naevus count, and skin damage due to sun exposure. In our experiments, we investigated the dichotomous problem of distinguishing patients with common naevi and dysplastic naevi from those with melanomas, based only on their clinical information. We had previously investigated the performance of a number of machine learning algorithms on an extension of this data set, which also included lesion features obtained by dermoscopy (epiluminescence microscopy) [24].

The data sets were split into training and test set. We used 60% of the data to train the algorithm, and 40% to test it. For the synthetic data set, the training set contained a total number of 240 data points from class 0, and 360 data points from class 1. The test set contained the remainder: 160 data points from class 0, and 240 data points from class 1. For the melanoma data set, the training set consisted of 909 data points from class 0 (common naevi and dysplastic naevi) and 63 data points from class 1 (melanomas). The test set consisted of the remaining 605 data points from class 0 and 42 data points from class 1. To consider the influence of different data set splits on our algorithm, we applied our approach on 20 different splits. This means that we randomly generated 20 different training and test sets with class distributions as described above. We calculated the area under the ROC curve for all of these splits and for all three algorithms (LR and *k*NN considered separately, and combined by our method). Our final performance numbers are the means of AUCs over all data set splits.

IV. RESULTS AND DISCUSSION

In Table 1, we compare our results with those of LR and *k*NN applied separately. We fixed the value of k = 10 in our k-nearest neighbor calculations. The AUC values of LR for each of the 20 splits were calculated by averaging the posterior class membership probabilities of all 10 models generated for assessing the variability of LR model outputs. The entries in Table 1 are the average (over 20 splits) of these averages (over 10 models). Besides these mean AUC values, we also report the standard deviations (SD) for LR, *k*NN, and our method of combination (denoted by "comb" in the table) on the synthetic and biomedical data set. It can be seen that our method achieves a higher classification performance for both data sets.

	synthetic data		melanoma data	
	AUC	SD	AUC	SD
LR	0.774	0.013	0.677	0.032
*k*NN	0.776	0.011	0.747	0.035
comb	0.794	0.011	0.785	0.038

Tab. 1. Performance comparison.

The advantage and novelty of our approach to data classification can be attributed to the heuristic method we use to combine LR model outputs and *k*NN local probability estimates. To illustrate this point, we consider the two data points highlighted in Figure 3. Both of the points belong to class 0. In this figure, the LR model classifies all points to the left of the dashed discriminatory line as belonging to class 1, and all points to the right of the line as belonging to class 0. In the following, we will use the notation of Equations (5) and (6). For the training set shown in Figure 3, the average standard deviation over the 10 LR models was $\bar{\sigma} =$

0.0071, and the average distance of a data point to its k = 10 nearest neighbors was $\overline{d}$ = 3.8872.

The first point (on the left side of the LR discriminatory line) was misclassified by the LR model, since it is on the wrong side of the line (the LR model output was lr_i = 0.6282). In contrast, the *k*NN probability estimate was nn_i = 0.2, which is closer to the true value of 0. The LR variability estimate of this data point was σ_i = 0.0067, and the *k*NN distance was d_i = 3.1074. The two measures of appropriateness, as defined in Equation (5), were thus app_{LR} = 0.9437 and app_{kNN} = 0.7994. This means that the LR model is less appropriate than the *k*NN part; the LR component therefore only contributes 0.4586 to the combined model, as calculated by the weighting factors in Equation (6). The remaining weight of 0.5414 is assigned to the *k*NN contribution, resulting in an overall combined model output of 0.3964. By weighting the correct *k*NN contribution more heavily than the incorrect LR contribution, our method was able to correct the error of the LR model.

The second data point (on the right side of the discriminatory line) was classified correctly by the LR model, with a value of lr_i = 0.2577. For this point, the *k*NN estimate is incorrect, with nn_i = 0.8. Because of the values σ_i = 0.0114 and d_i = 8.1644, which result in appropriateness estimates of app_{LR} = 1.6056 and app_{kNN} = 2.1003, the incorrect contribution of the *k*NN component is weighted less heavily than the correct contribution of the LR component. The combined model estimate was 0.4927. The *k*NN contribution was downweighted, and not large enough to cause an incorrect classification at the p = 0.5 LR threshold.

By weighting the contributions of both classifiers for each data point we consider their characteristics. Being limited to linear discriminatory hyperplanes, LR models have less variance than *k*NN classifiers. However, with increasing distances of the k nearest neighbors, the class prediction of a *k*NN classifier becomes less accurate. In this case, the LR output is considered to be more appropriate. On the other hand, if there is (relatively) high variability between the 10 LR models we generate, we assess the local probability estimate of *k*NN more appropriate.

Our experiments showed that our method of combining LR and *k*NN classifiers achieves a higher classification performance than the individual classifiers, both on a synthetically generated data set and on a biomedical data set. The improvement on the biomedical data set was higher than the improvement on the synthetic data set. We speculate that this may be due to the data set characteristics: The synthetic data set consists only of real-valued features, whereas the melanoma data set contains real, ordinal and categorical features. We presently investigate which data set characteristics may have an influence on the success of our method.

Possible directions for future research lie in the investigation of different heuristics for assessing the variability of LR models (possibly by bootstrapping methods), and in alternative methods to improve the *k*NN component of the model, either by applying adaptive *k*NN methods [25], or by weighting the distance calculations by the LR β coefficients.

V. SUMMARY

In this paper we presented the context for combining classifiers. We reviewed several approaches to generate different classifiers as well as established methods to combine different classifiers. Then we introduced a novel ensemble method to weight different classifiers based on their characteristics and for each test point individually. We elaborate this approach by combining a rigid LR model with the local flexibility of a *k*NN classifier. The individual weighting of both models for each given data point exploits the strength of LR, having little variance, as well as the strength of *k*NN, having low bias. Our experimental results on a synthetic and a biomedical data set confirm the feasibility and power of our approach. The combination of the models achieves, on both data sets, a higher classification performance than the individual models.

ACKNOWLEDGEMENTS

This work was supported by the Austrian Genome Program (GEN-AU), project Bioinformatics Integration Network (BIN III).

REFERENCES

[1] Christian Baumgartner, Christian Böhm, Daniela Baumgartner, et al., “Supervised machine learning techniques for the classification of metabolic disorders in newborns,” Bioinformatics, vol. 20, pp. 2985-2996, 2004.

[2] S. Dreiseitl and L. Ohno-Machado, “Logistic regression and artificial neural network classification models: a methodology review,” Journal of Biomedical Informatics, vol. 35, pp. 352–359, 2002.

[3] J. Quinlan, C4.5: Programs for Machine Learning, Morgan Kaufmann Publishers, 1993.

[4] C. Bishop, Neuronal Networks for Pattern Recognition, Oxford University Press, 1995.

[5] N. Cristianini and J. Shawe-Taylor, An Introduction to Support Vector Machines, Cambridge University Press, 2000.

[6] T. Mitchell, Machine Learning, McGraw Hill, 1997.

[7] F. Provost, T. Fawcett, and R. Kohavi, “The case against accuracy estimation for comparing induction algorithms,” Proceedings of the 15th International Conference on Machine Learning, 1998, pp. 445-453.

[8] A. Bradley, “The use of the area under the ROC curve in the evaluation of machine learning algorithms,” Pattern Recognition, vol. 30, pp. 1145-1159, 1997.

[9] D. Bamber, “The area above the ordinal dominance graph and the area below the receiver operating characteristic graph,” Journal of Mathematical Psychology, vol. 12, pp. 387-415, 1975.

[10] K. Chen, L. Wang, H. Chi, “Methods of combining multiple classifiers with different features and their application to text-independent speaker identification,” International Journal of Pattern Recognition and Artificial Intelligence, vol. 11, pp. 417-445, 1997.

[11] Y.S. Huang, C.Y. Suen, “A method of combining multiple experts for the recognition of unconstrained handwritten numerals,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol.17, pp. 90-94, 1995

[12] L. Breiman, “Bagging predictors,” Machine Learning, vol.24, pp. 123-140, 1996.

[13] Y. Freund and R.E. Schapire, “Experiments with a new boosting algorithm,” Proceedings of the 13th International Conference on Machine Learning, pp. 148-156, 1996.

[14] Y. Freund and R.E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," Journal of Computer and System Sciences, vol. 55, pp. 119-139, 1997.
[15] K. Cherkauer, "Human expert-level performance on a scientific image analysis task by a system using combined artificial neural networks," Working Notes of the AAAI Workshop on Integrating Multiple Learned Models, pp. 15-21, 1996.
[16] R. Ranawana und V. Palade, "A neural network based multi-classifier system for gene identification in DNA sequences," Neural Computing and Applications, vol. 14, pp. 122-131, 2005.
[17] T.G. Dietterich and G. Bakiri, "Solving multiclass learning problems via error-correcting output codes," Journal of Artificial Intelligence Research, vol. 2, pp. 263-286, 1995.
[18] Z. Ghahramani and H. Kim, "Bayesian Classifier Combination," Gatsby Technical Report, 2003.
[19] D.H. Wolpert, "Stacked generalization," Neural Networks, vol. 5, pp. 241-259, 1992.
[20] S. Dreiseitl and L. Ohno-Machado, "Logistic regression and artificial neural network classification models: a methodology review," Journal of Biomedical Informatics, vol. 35, pp. 352–359, 2002.
[21] F. Harrell, Regression Modeling Strategies: With Applications to Linear Models, Logistic Regression, and Survival Analysis, Springer, 2001.
[22] D. Hosmer and S. Lemeshow, Applied Logistic Regression, John Wiley & Sons, 2nd Edition, 2000.
[23] T. Hastie, R. Tibshirani, and J. Friedman, The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Springer, New York, 2001.
[24] S. Dreiseitl, L. Ohno-Machado, S. Vinterbo, H. Billhardt, and M. Binder, "A comparison of machine learning methods for the diagnosis of pigmented skin lesions," Journal of Biomedical Informatics, vol. 34, pp. 28–36, 2001.
[25] T. Hastie and R. Tibshirani, "Discriminant adaptive nearest neighbor classification," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 18, pp. 607–616, 1996.

Melanie Osl, MSc, received her MSc degree from the University of Medical Informatics and Technology (UMIT) in Hall, Austria. She is currently a PhD student at the Institute of Biomedical Engineering at UMIT. Her research interests include knowledge discovery and data mining in biomedicine, clinical bioinformatics and machine learning.

Christian Baumgartner, PhD, is Associate Professor of Biomedical Engineering and head of the Institute for Biomedical Engineering at the University of Health Sciences, Medical Informatics and Technology (UMIT). He received his MSc and PhD degree (1998) in Biomedical Engineering at Graz University of Technology, Austria, his habilitation (2006) at UMIT, and is the author of more than 50 publications in referred journals, book sections and conference proceedings. From 2007-2008 he was a visiting scientist at the Barnett Institute of Chemical and Biological Analysis, Northeastern University, Boston, MA and Harvard Medical School. His main research interests include data mining and knowledge discovery in biomedicine, clinical bioinformatics, and computational systems biology.

Bernhard Tilg, PhD, is Full Professor of Medical Informatics at the University for Health Sciences, Medical Informatics and Technology (UMIT), Hall in Tirol, Austria. He was the rector of UMIT between 2004 and 2008. In 1991 and 1995 he got the diploma and the doctor degree in electrical engineering from Graz University of Technology, respectively. Before February 2002, he was an Associate Professor for Biomedical Engineering at Graz University of Technology. His main research interests are biomedical signal processing and imaging, physiological modeling and simulation, inverse problems and estimation theory, and biomedical sensors.

Stephan Dreiseitl, PhD, received his MSc and PhD degrees from the University of Linz, Austria, in 1993 and 1997, respectively. He worked as a visiting researcher at the Decision Systems Group/Harvard Medical School before accepting a post as professor at the Upper Austria University of Applied Sciences in Hagenberg, Austria, in 2000. He is also an adjunct professor at the University of Health Sciences, Medical Informatics and Technology in Hall, Austria, and adjunct faculty at the Decision Systems Group/Harvard Medical School in Boston, USA. His research interests lie in the development of machine learning models and their application as decision support tools in biomedicine.