Generalization improvement in multi-objective learning

Lars Gräning, Yaochu Jin, Bernhard Sendhoff

2006

To appear: WCCI'06, Vancouver, IEEE Press, July 16-21, 2006

Generalization Improvement in Multi-Objective Learning

Lars Gräning, Yaochu Jin, *Senior Member, IEEE*, and Bernhard Sendhoff, *Senior Member, IEEE*

Abstract— **Several heuristic methods have been suggested for improving the generalization capability in neural network learning, most of which are concerned with a single-objective (SO) learning tasks. In this work, we discuss generalization improvement in multi-objective learning (MO). As a case study, we investigate the generation of neural network classifiers based on the receiver operating characteristics (ROC) analysis using an evolutionary multi-objective optimization algorithm. We show on a few benchmark problems that for MO learning such as the ROC based classification, the generalization ability can be more efficiently improved within a multi-objective framework than within a single-objective one.**

I. Introduction

One of the main challenge in neural network learning is to improve the generalization capability of neural networks. A lot of heuristic methods have been suggested to improve the generalization capability of learning models by avoiding over-fitting. An overview of the existing techniques for generalization improvement, e.g., early stopping, regularization, network pruning and weight decay can be found in [3] or [21]. Almost all existing methods consider single-objective learning only, probably because classical learning algorithms, e.g., the gradient-based learning methods, cannot deal with multiple learning objectives efficiently. Although evolutionary algorithms have widely been employed for optimizing the weights as well as the topology of neural networks [24], most evolutionary learning algorithms are also based on a single criterion.

In contrast, most machine learning problems have more than one objective. In general, they can be categorized into two groups. In the first category, the algorithm has to optimize only one objective, e.g., the mean squared error. To improve the generalization ability of the learning model, additional objectives, e.g., a regularization term, has to be considered in the learning algorithm. Thus, though the learning problem *per se* has only one target, it can be better handled if it is considered as a multi-objective problem. In the second category, machine learning tasks themselves have more than one objective. For example in receiver operating characteristics (ROC) analysis of classifiers [7], one needs to maximizing the true positive rate (TPR) and minimize the false positive rate (FPR) at the same time. Other learning algorithms that minimizes multiple error measures have also been investigated [22]. Another example is sequential learning, where more than one task needs to be learned. The main challenge here is the learning of a new task may cause the forgetting of the previously learned tasks, which is often known as catastrophic forgetting [17].

Existing SO learning approaches often combine the multiple objectives into a scalar cost function to make the learning problem tractable. In the recent years, multi-objective machine learning has received increasing attention, see [13] for an overview, partly due to the great success of evolutionary multi-objective algorithms [5]. However, most existing research is concerned with the first category of multi-objective learning with a few exceptions. One of the early papers where the optimization of neural networks has been considered as multi-objective task came from the field of medical diagnostics [16], where the weights of neural classifiers are optimized with respect to both TPR and FPR. A recent paper [6] adapted the weights of neural classifiers based on the ROC and extended the algorithm to multi-class problems.

In this work we attempt to investigate the generalization issue for the second category of multi-objective learning problems. An evolutionary MOO algorithm is employed for simultaneously optimizing the structure and the weights of neural networks for solving binary classification tasks, where the TPR is to be maximized and the FPR is to be minimized. We improve the generalization ability by perturbing the training patterns during optimization. We show that by using a multi-objective learning framework, the generalization ability can be more efficiently improved compared to an evolutionary single-objective optimization algorithm, where multiple objectives are combined to form a scalar cost function.

Section II of the paper describes existing work on multi-objective learning in more detail. Section III presents the evolutionary multi-objective algorithm for optimization of neural classifiers. Two methods for generating perturbations to training patterns are also described briefly. Simulation studies are conducted in Section IV, where we show that the generalization ability can be more effectively improved if the MO learning approach is adopted. A summary of the work is provided in Section V.

II. Generalization and Multi-objective learning

A. Generalization by Multi-objectivization

The main goal in machine learning is to find learning models (e.g., neural networks) that are optimal with respect to the training data as well as to unseen data for one or multiple learning objectives. Of course there is in principal no knowledge about unseen data available. That is the reason why one has to control the learning process in order to learn a general approximation of the underlying problem and to avoid over-fitting to the training data.

Lars Gräning, Yaochu Jin and Bernhard Sendhoff are with Honda Research Institute Europe GmbH, Carl-Legien-Strasse 30, D-63073 Offenbach (Main), Germany (email: lars.graening@honda-ri.de; yaochu.jin@honda-ri.de; bernhard.sendhoff@honda-ri.de)

In SO learning, many methods for improving generalization, such as early stopping, weight decay, regularization and adding artificial noise during training have been suggested. In SO learning, the underlying problem is often defined by one scalar objective function, e.g., the mean squared error. As commonly known the complexity of neural networks plays a crucial role in improving generalization capability and neural networks with an overly complex structure are more likely to overfit the training data than neural networks with a simpler structure. To control the complexity, regularization techniques can be used in SO learning tasks to penalize the complexity by adding a penalty term to the original cost function:

$$f = E + \lambda\Omega,$$

where E defines the objective function and Ω the complexity term. One drawback of this regularization technique is that the parameter λ has to be determined to control the impact of penalization. As shown in [11], regularization can be formulated as a multi-objective optimization problem, where one objective is to minimize the error metrics itself and the second one is to minimize the complexity:

$$f1 = E$$
$$f2 = \Omega.$$

The MOO algorithm will result in a set of neural networks that trade off between accuracy and complexity. By analyzing the neural networks on the Pareto front, neural networks from which interpretable rule can be extracted [12], or neural networks that generalize well [14], can be identified.

Another idea of using MOO for improving the generalization capability is described in [1], where the available training data D is divided into two data sets, D_1 and D_2. After that two objectives can be formulated, where each objective is to minimize the error on each data set:

$$f1 = E(D_1)$$
$$f2 = E(D_2).$$

This approach may work only for relatively small neural networks, otherwise the algorithm may result in one neural network that overfits both data sets.

B. Generalization Improvement in Multi-objective Learning

As previously discussed, there are multi-objective problems where more than one objective has to be considered without taking generalization into account. In this case, generalization must be addressed explicitly. As indicated in [8], a network may overfit the training data with respect to one objective, but not with respect to another. Thus, it becomes difficult to define a single stop criterion if early stopping is adopted to avoid overfitting. In [8], two techniques for improving the generalization capability of neural networks in multi-objective learning have been suggested. The first approach is inspired by the cross-validation method from single-objective learning, where the training data set is divided into a training data set and a validation data set. Potential solutions which are non-dominated by the solutions in the training archive are tested on the validation archive. Only if the found solution is also non-dominated by the solutions in the validation archive, the solution is added to the validation data set. The second method is based on bootstrapping techniques, where the training data set is bootstrapped to generate n subsets. The worst fitness on all bootstrapped subsets becomes the final fitness value.

Another idea to prevent overfitting in multi-objective learning is suggested in [23], where an MO learning algorithm for solving face detection tasks has been studied. The training data set has also been divided into a training set and a validation set. The training data set is used to evaluate the fitness, whereas the validation data set is used for life-time learning. The cross-validation approach between learning and evolution is an improvement compared to other methods but it has also been noticed that improving generalization capability should be focus of further research.

Similar to SO learning algorithms, an additional objective to regulate for example the complexity of neural networks can be introduced explicitly to improve the generalization ability in multi-objective learning.

III. Evolutionary Algorithm for the Optimization of Neural Classifiers

In this section the evolutionary algorithm for the optimization of neural classifiers is described, which has been used both for the single-objective as well as the multi-objective learning algorithms. The evolutionary algorithm is combined with a lifetime learning algorithm and both the weights and the structure of the neural networks are optimized. Fig. 1 illustrates the major elements of the a generic evolutionary algorithm.

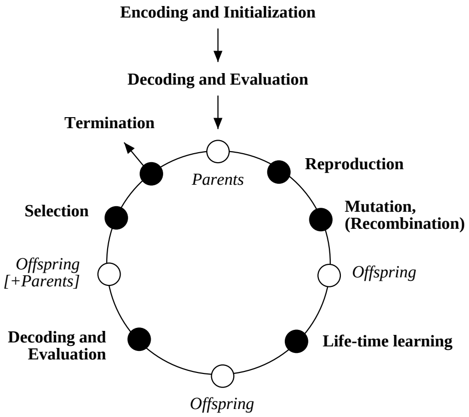

Fig. 1. A generic evolutionary algorithm including lifetime learning.

All individual $p_i = p_i(C_i, W_i), i = 1, \ldots, \mu$ of the parent population are initialized randomly. A direct encoding scheme has been used to encode the neural classifiers. A connection matrix C and a weight matrix W are stored in the genotype of the individual as shown in Fig. 2, where the connection matrix is encoded as a binary string and the weight matrix as a string of real-valued numbers. In this work feed-forward neural networks with one hidden layer and sigmoid-like activation functions are used. A threshold $\vartheta \in [-1, 1]$ is applied to the continuous output of the neural network to derive class membership. The threshold remains constant during optimization ($\vartheta = 0$).

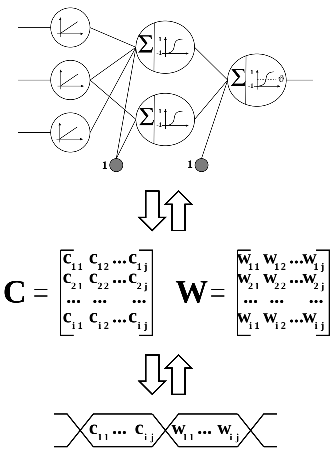

Fig. 2. Encoding of neural network classifiers.

A tournament selection with tournament size two has been employed to create parent individuals for the next generation from a combination of the parent and offspring of the current generation. During the selection, two parents are chosen randomly and the "fitter" one is copied to the next generation. Recall that the definition of a fitter solution is different in SO and MO learning, which will be discussed in greater detail later. Afterwards each offspring $o_j = o_j(C_j, W_j), j = 1 \ldots \lambda$ is mutated randomly using one out of the following five mutation operators:

- add a neuron,
- delete a neuron,
- add a connection,
- delete a connection and
- jog weights.

No recombination operator has been adopted. The weights of the neural classifiers are modified in two ways. At first, the "jog weights" mutation operator is used to perform a global search in the weight-space by adding a normally distributed random number to all weights. Then, a number of life-time learning iterations are embedded in order to do a more local search within the weight-space. Here, an improved version of the RProp algorithm [10] is used, which adapts the weights of the neural network by minimizing the mean squared error. After learning, the weights are coded back into the genotype of the offspring, based on the paradigm of Lamarckian evolution [9].

IV. Single-objective Optimization of Neural Classifiers

The main difference between SO and MO learning algorithms is the number of objectives that define the quality of an individual. In SO learning, the fitness is defined by a scalar value. In this work, the Matthews correlation coefficient (MCC) [19] is used to determine the quality of the neural classifiers:

$$Q_{MCC} = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP + FN)(TP + FP)(TN + FP)(TN + FN)}},$$

where the correct classifications are defined by the number of true positives (TP) and the number of true negatives (TN), whereas the misclassifications made by the classifier are defined by the number of false positives (FP) and the number of false negatives (FN). The MCC evaluates the correlation between the prediction of the neural classifier $\vec{y} \in \{-1, 1\}^N$ and the correct class $\vec{c} \in \{-1, 1\}^N$, where N is the number of patterns in a data set. The objective is to maximize the MCC $f = max\{Q_{MCC}\}, Q_{MCC} \in [-1, 1]$.

During the tournament selection, the one with a larger Q_{MCC} is considered as fitter and wins the tournament.

The SO learning algorithm results in one optimized classifier. A ROC curve is then generated by varying the threshold ϑ between 0 and 1. During operation, one can choose a threshold depending on one's preference over TPR and FPR. The preference often depends on the costs of misclassifications.

V. Multi-objective Optimization of Neural Classifiers

The MO learning algorithm simultaneously optimizes the TPR and FPR. Two objectives can be formulated by maximizing the TPR

$$f_1 = max\{TPR\} = max\left\{\frac{TP}{TP + FN}\right\},$$

and minimizing the FPR

$$f_2 = min\{FPR\} = min\left\{\frac{FP}{TN + FP}\right\},$$

where $TP + FN$ is the number of positive patterns ($c^{(k)} = 1$) and $TN + FP$ is the number of negative patterns ($c^{(k)} = -1$) in a data set. Consequently the fitness of an individual that describes the quality of the neural classifier is a vector $\vec{f}$ containing two elements, the TPR and the FPR.

Besides the number of objectives an MO learning algorithm differs from a SO learning algorithm mainly in the selection method. While in SO learning, it is straight-forward to choose the "fitter" individual during the tournament selection, and pass it to the next generation, it is more complicated when more than one objective has to be considered. In this work, the crowded tournament selection suggested in NSGA-II algorithm [4] has been adopted. In the crowded tournament selection, the union of the parent and offspring individuals are sorted according their rank based on the concept of

Pareto-dominance. Besides, a crowding distance reflecting how "crowded" it is near each individual is calculated to promote the diversity of the population. The Pareto-dominance can be defined as follows for a minimization problem:

> *A vector* $\vec{f}^* = (f_1^*, \ldots, f_m^*)$ *is said to dominate a vector* $\vec{f} = (f_1, \ldots, f_n)$, *if* $f_i^* \leq f_i, \forall i = 1, 2, \ldots, m$ *and* $\exists j \in \{1, \ldots, m\}$, *such that* $f_j^* < f_j$.

Here m is the number of objectives. In Fig. 3, the performance of three classifiers A, B and C is visualized in the ROC space in order to illustrate the concept of dominance. By definition it can be seen that classifier A dominates classifier C because classifier A is better in the TPR as well as in the FPR. Classifier A and classifier B are not dominated by any other classifiers. In this sense, they belong to the set of non-dominated solutions. Among the non-dominated solutions, one cannot say one solution is better than another. In this example, we cannot say that solution A is better than B nor that classifier B is better than A. If a solution is non-dominated in the entire feasible solution space, then the solution is said to be Pareto-optimal, and the union of all Pareto-optimal solutions is termed as the Pareto front. The goal of multi-objective learning algorithm is to approximate the Pareto front.

The readers are referred to [4] for the details of the crowded tournament selection.

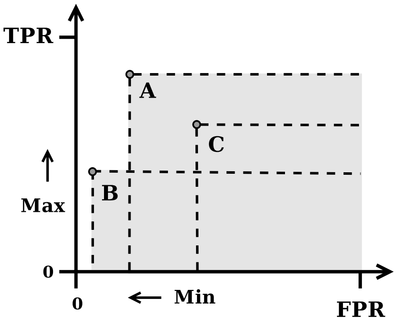

Fig. 3. Three solutions are shown in the objective space. Solutions A and B dominate solution C. The solutions A and B belong to the set of non-dominated solutions.

A number of non-dominated classifiers can be acquired in the MO learning approach. An even larger set of classifiers can thus be obtained by varying the threshold of all non-dominated solutions.

VI. Noise Based Generalization Improvement

One heuristic method that can directly be applied to multi-objective learning is the perturbation of training patterns during learning. Perturbing the training patterns results in a noisy fitness function and only neural networks that are robust against small changes in the training patterns will be competitive against other networks. In this way, overfitting of the training data can be avoided. Two approaches to perturbing the training patterns are investigated in this work. In the first approach, Gaussian noise is added to the features of the training patterns, whereas in the second approach the training patterns are perturbed by means of averaging training patterns in a local neighborhood (which should be in the same class). In both approaches one has to assume that small variations in the features of the training patterns do not change their class membership.

A. Additive Gaussian Noise

The training patterns for learning neural networks are presented in form of input/output pairs, where $\vec{x}_k$ is the input and c_k is the desired output. In each generation a new artificial training data set is generated by adding a normal distributed random number to each element of the input vector $\vec{x}_k$, this can be written as

$$\tilde{x}_i^{(k)} = x_i^{(k)} + \mathcal{N}(0, \sigma_k^2),$$

where σ_k^2 is the standard deviation of the Gaussian distribution. There are two ideas of choosing the standard deviation σ_k^2. An intuitive idea is to choose a predefined value. A better idea is to choose a relative value depending on the distribution of the training patterns within the feature space. Here we take the minimal distance between the k-th pattern and the remaining patterns in the feature space:

$$\sigma_k^2 = \frac{1}{\xi} \min_{l, l \neq k} \left\{ \sqrt{\sum_{i=1}^{N} (x_i^{(k)} - x_i^{(l)})^2} \right\},$$

where parameter ξ controls the impact of noise, N is the number of all patterns. It is more feasible to pre-define such a parameter than to pre-define the standard deviation directly because this parameter can be chosen independent of the distribution of the training patterns. After adding noise to the input of each training pattern the perturbed data set can be used to determine the fitness of the neural classifiers. So in each generation a new data set is presented to the classifiers for fitness evaluation.

B. Adding Noise by Means of Averaging

In the second approach, new artificial patterns are generated by averaging the original patterns within a local neighborhood. This can be formulated as

$$\hat{x}_i^{(k)} = \frac{1}{\nu + 1} \sum_{l=0}^{\nu} x_i^{(l)},$$

where ν defines the local neighborhood and is determined randomly between $\nu = 0$ and $\nu = \nu_{max}$. If $\nu = 0$ the original pattern is added to the new data set, otherwise the mean of the local neighborhood is calculated for each input dimension and the pattern with the average inputs $\underline{\hat{x}}^{(k)}$ is added to the new training data set. The parameter ν_{max} has to be determined before optimization. Finally the new data set is used for fitness evaluation.

	Diabetes	Heart	Card
Features	8	35	51
N_H^{max}	30	20	10
D_{Train}	576	690	518
N_{pos}	197	301	227
N_{neg}	379	389	291
D_{Test}	192	230	172
N_{pos}	70	110	80
N_{neg}	122	120	92

TABLE I

NUMBER OF PATTERNS, NUMBER OF FEATURES AS WELL AS THE MAXIMAL NUMBER OF HIDDEN NEURONS FOR EACH BENCHMARK PROBLEM.

VII. MAIN RESULTS

A. *Experimental Setup*

Both the SO learning algorithm and the MO learning algorithm are applied to three binary classification benchmark problems taken from the UCI database [20], namely Diabetes, Heart and Card. The Diabetes problem is to decide whether a human is diabetes positive or not. The Heart problem also comes from the field of medical diagnosis. Based on the given features the classifier has to diagnose a heart disease. The Card problem is a different one. Based on the given features the task is to determine whether a credit card can be approved or not. All of them are binary classification problems with different numbers of features and different numbers of samples in the available data set. The data set D of each problem is divided into a training set D_{Train} and a test set D_{Test}. A list of the database sizes and the class distribution within the training and test data sets are shown in Table I. The given number of features for each problem and the maximum number of hidden nodes are listed as well. Depending on the classification problem the number of hidden neurons is bounded by N_H^{max}. N_H^{max} is chosen so that the maximal number of free parameters in the network is smaller than or equal to the number of training patterns but no less than $N_H^{max} = 30$.

The size of the parent population is set to $\mu = 50$. In each generation $\lambda = 50$ offspring are reproduced and then mutation and life-time learning is applied to the offspring. The number of iterations for life-time learning is set to 200. Each optimization run proceeds for 300 generations. In order to get significant results, each optimization run is performed 10 times, which results in 10 ROC curves. To capture all ROC curves a median curve with a confidence band, which is generated by using fixed width bands [18], is calculated. Fig. 4 illustrates an example, where the ROC curves, the median curve and the confidence interval are plotted. The confidence band captures about 90 percent of all ROC curves.

B. *Comparing the SO and MO Learning Algorithms*

As previously mentioned, each run of the SOO algorithm results in one neural classifier. By varying the threshold of the obtained classifier, we can obtain one ROC curve. Thus, 10 ROC curves can be obtained from 10 runs.

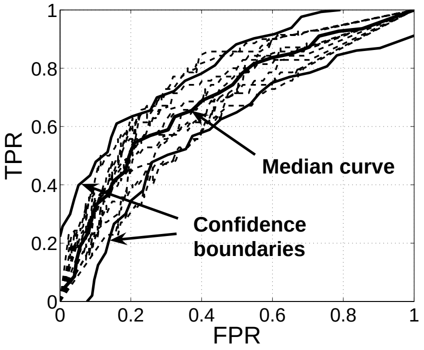

Fig. 4. The ROC curves from 10 optimization runs. The ROC curves from independent optimization runs can be captured by a median curve with a confidence band.

The MOO algorithm results in a set of neural classifiers with different structures and different weights showing different TPR and TFR tradeoffs. Additionally the threshold of each resulting classifier is varied, which results in a set of ROC curves in each optimization run. From these ROC curves, we can generate one ROC curve representing the non-dominated solutions among all the generated ROC curves. Thus, both the SO and the MO learning algorithms result in one ROC curve from one optimization run.

The resulting ROC curves on the training data for the three problems are presented in Fig. 5. From the figures, we see that the performance of the SO learning and MO learning are comparable, all showing good performance on the training data.

As we know, good performance on the training data does not necessarily mean good performance on unseen data. The test performance of the SO and the MO learning algorithms from ten optimization runs is depicted in Fig. 6. Comparing the performance on the training data and test data, it is clear to see that overfitting has occurred in both SO and MO learning. In the following, we attempt to address overfitting by adding noise to the training data.

C. *Additive Gaussian Noise*

In this section, we compare the SO and the MO learning algorithms when the training patterns are perturbed by adding Gaussian noise to the input vectors of the training patterns during fitness evaluation. The parameter ξ that influences the impact of noise is empirically set to $\xi = 0.3$. Fig. 7 shows the performance of the resulting classifiers on the test data set for all three classification problems. As can be seen, the MO learning algorithm shows better performance than the SO learning algorithm on all test problems. The ROC curves generated using the MO learning algorithm from the ten optimization runs are no worse than those using the SO learning algorithm. Additionally it is noticed that the width of the confidence band of ROC curves from the MO learning algorithm, especially for the Card problem, is smaller than that from the SO algorithm. We can thus conclude that the

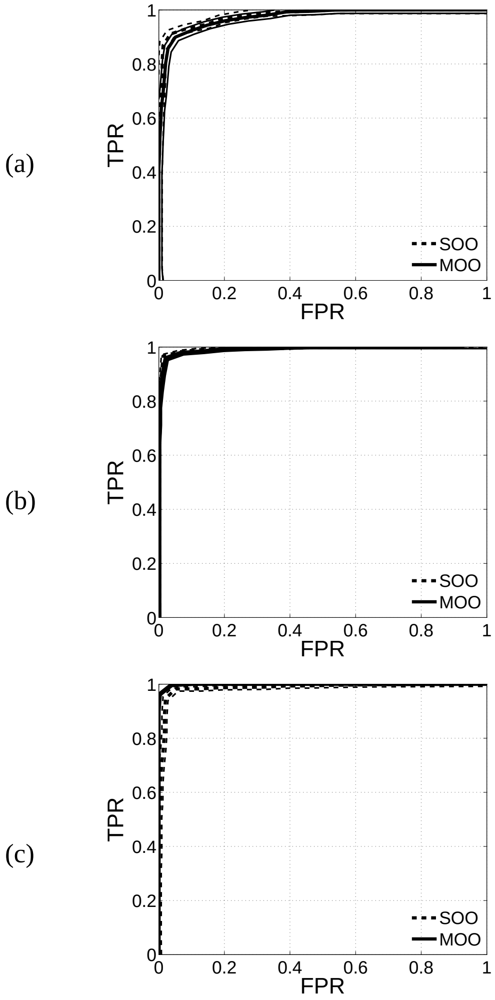

Fig. 5. ROC curves on the training data from the SO and MO learning algorithms. (a) Diabetes, (b) Heart, and (c) Card.

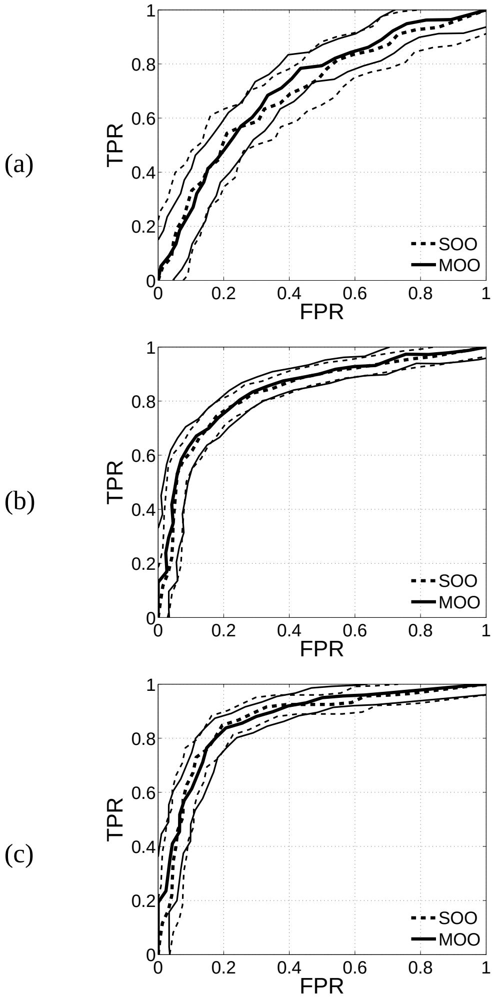

Fig. 6. ROC curves on the test data from the SO and MO learning algorithms. (a) Diabetes, (b) Heart, and (c) Card.

MO learning algorithm is more robust than the SO learning algorithm in avoiding overfitting.

D. Adding Noise by Averaging

In this approach, the training patterns are perturbed by averaging a pre-defined number of training patterns within a local neighborhood. The parameter that defines the size of the neighborhood is set to $\nu_{max} = 10$. The optimization runs are performed under the same conditions as in the experiments above.

Fig. 8 shows the results for the three classification problems. It can be seen that the MO learning algorithm improves the generalization capability compared to the SO learning algorithm in all the three problems. Comparing the results in Fig. 8 with the results in Fig. 7, we notice that adding Gaussian noise to the training patterns and perturbing the training patterns by means of averaging show similar results, though the former appears a little better.

VIII. Conclusions

In this paper we investigate improvement of generalization ability of neural classifiers with multiple learning objectives. We give a brief review of the techniques for addressing overfitting in single-objective learning and stress that not all these techniques can directly be transplanted to multi-objective learning algorithms. Then, the perturbation method, which can directly be used in MO learning, is empirically studied in order to avoid overfitting in generating neural classifiers based on ROC analysis . Two similar approaches for perturbing the training patterns are investigated for evolutionary single-objective and multi-objective learning. In the first approach, Gaussian noise is added to the input of the training patterns, whereas in the second approach artificial training patterns are generated by means of averaging. In both approaches the MO learning algorithm shows better results than the SO learning algorithm on test data. The results indicate that Pareto-based multi-objective learning approaches should be preferred over single-objective learning

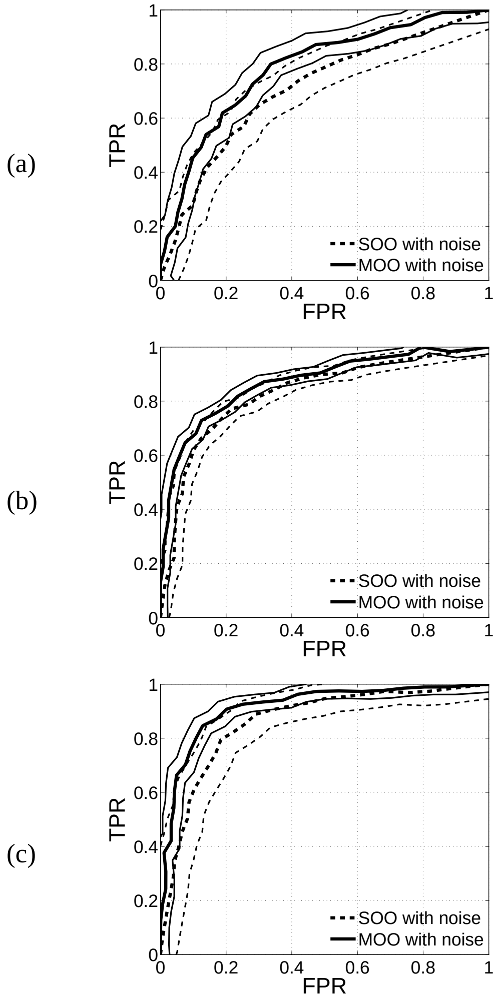

Fig. 7. Test performance using the SO and MO learning algorithms, where the training patterns are perturbed by Gaussian noise during evaluation. (a) Diabetes, (b) Heart, (c) Card.

algorithms that sum up the multiple learning objectives, particularly when noise is added into training patterns to avoid overfitting.

One weakness of the proposed MO learning method is that the life-time learning is still based on a scalar cost function that minimizes the mean squared error instead of minimizing TPR and FPR simultaneously. This can be addressed, e.g., by using a learning algorithm that maximizes the TPR or minimizes FPR randomly. Besides, it remains to be clarified if better performance can be obtained to perturb the training data during the life-time learning. Third, the noise based methods should be compared to other methods, e.g., suggested in [8] and [23].

REFERENCES

[1] H.A. Abbass, Pareto neuro-evolution: Constructing ensemble of neural networks using multi-objective optimization, In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC2003)*, vol. 3, pp. 2074–2080, IEEE-Press, 2003

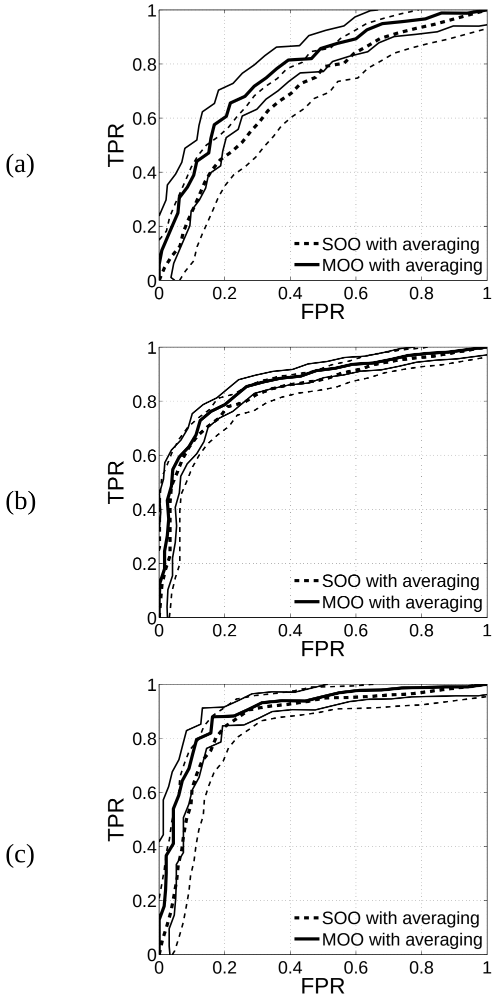

Fig. 8. Test performance using the SO and MO learning algorithms by perturbing the training patterns through averaging. (a) Diabetes, (b) Heart, and (c) Card.

[2] P. Baldi and S. Brunak and Y. Chauvin and C. A. F. Andersen and H. Nielsen, Assessing the accuracy of prediction algorithms for classification: An overview, *Bioinformatics*, vol. 16, no. 5, pp. 412–424, Oxford University Press, May 2000

[3] C. M. Bishop, *Neural Networks for Pattern Recognition*, Clarendon Press, Oxford New York, 1995

[4] K. Deb and S. Agrawal and A. Pratap and T. Meyarivan, A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II, *Proceedings of Parallel Problem Solving from Nature*, PPSN VI, Springer, pp.849-858, 2000

[5] K. Deb, *Multi-Objective Optimization Using Evolutionary Algorithms*, Wiley, 2001

[6] R.M. Everson and J.E. Fieldsend, Multi-objective optimization for receiver operating characteristic analysis, in *Multi-objective Machine Learning*, Y. Jin (Ed.), Springer, Feb. 2006

[7] T. Fawcett, *ROC graphs: Notes and practical considerations for data mining researchers*, Technical report, HP Laboratories Palo Alto, 2003

[8] J.E. Fieldsend and S. Singh, Pareto evolutionary neural networks, *IEEE Transactions on Neural Networks*, vol. 16, no. 2, pp. 338–354, 2005

[9] M. Hüsken and B. Sendhoff, Evolutionary optimization for problem classes with Lamarckian inheritance, In *Seventh International Conference on Neural Information Processing (ICONIP 2000) -Proceedings*, Edited by Soo-Young Lee, vol. 2, pp. 897–902, 2000

[10] C. Igel and M. Hüsken, Improving the RProp learning algorithm,

In *Second International Symposium on Neural Computation*, ICSC, Academic Press, pp. 115–121, 2000

[11] Y. Jin and T. Okabe and B. Sendhoff, Neural network regularization and ensembling using multi-objective evolutionary algorithms, *Proceedings of Congress on Evolutionary Computation*, pp.1-8, Portland, 2004

[12] Y. Jin, B. Sendhoff, E. Körner. Evolutionary multi-objective optimization for simultaneous generation of signal-type and symbol-type representations. *Evolutionary Multi-Criterion Optimization*, LNCS 3410, pages 752–766, 2005

[13] Y. Jin (editor). Multi-objective Machine Learning. Springer, Berlin, 2006

[14] Y. Jin, Simultaneous generation of accurate and interpretable neural network classifiers. In: *Multi-Objective Machine Learning*, Y. Jin (ed.), pp.291-312, 2006

[15] K.W.C Ku and M.W. Mak, Knowledge incorporation through lifetime learning, In *Knowledge Incorporation in Evolutionary Computation*, Y. Jin (ed.), Springer, pp.359-384, 2004

[16] M.A. Kupinski and M.A. Anastasio, Multiobjective genetic optimization of diagnostic classifiers with implications for generating receiver operating characteristic curves, *IEEE Transactions on Medical Imaging*, vol. 18, no. 8, pp.675-685, 1999

[17] M. McCloskey and N.J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. *The Psychology of Learning and Motivation*, 24:109–165, 1989

[18] S.A. Macskassy and F. Provost. ROC confidence bands: Methods and an empirical evaluation, *Workshop on ROC Analysis in AI*, 61–70, 2004

[19] P. Baldi et al, Assessing the accuracy of prediction algorithms for classification: An Overview. *Bioinformatics Review*, 16(5):412–424, 2000

[20] D.J. Newman and S. Hettich and C.L. Blake and C.J. Merz, UCI - repository of machine learning database, 1998

[21] R. D. Reed and R. J. Marks II, *Neural Smithing - Supervised Learning in Feedforward Artificial Neural Networks*, The MIT Press, Cambridge Massachusetts, 1999

[22] Y. Wang and F. Wahl. Multiobjective neural network for image reconstruction, *IEEE Proceedings - Vison, Image and Signal Processing*, vol. 144, no. 4, pp. 233–236, 1997

[23] S. Wiegand and C. Igel and U. Handmann, Evolutionary multi-objective optimization of neural networks for face detection, *International Journal of Computational Intelligence and Applications*, vol. 4, no. 3, pp. 237–253, 2004

[24] X. Yao, Evolving artificial neural networks, *Proceedings of the IEEE*, 87(9), pp. 1423–1447, September 1999