Support Vector Neural Training.

Włodzisław Duch

***Index Terms*— Mulitlayer perceptrons, neural networks, backpropagation, active learning, machine learning.**

***Abstract*— Neural networks are usually trained on all available data. Support Vector Machines start from all data but near the end of the training use only a small subset of vectors near the decision border. The same learning strategy may be used in neural networks, independently of the actual optimization method used. Feedforward step is used to identify vectors that will not contribute to optimization. Threshold for acceptance of useful vectors for training is dynamically adjusted during learning to avoid excessive oscillations in the number of support vectors. Benefits of such approach include faster training, higher accuracy of final solutions and identification of a small number of support vectors near decision borders. Results on satellite image classification and hypothyroid disease obtained with this type of training are better than any other neural network results published so far.**

I. INTRODUCTION

Typical training error curve of the feedforward neural networks has exponential shape, showing rapid decrease of the error in initial epochs, followed by almost flat long tail, where the network error decreases very slowly. The training progresses according to the law of diminishing returns. In the final part of the training, presentation of most vectors that are far from decisions borders has almost no influence on the network parameters. In multi-layer perceptrons (MLPs) these vectors are in the region where outputs of neurons are close to 0 or 1, thus giving vanishing gradients. Only the vectors that are close to the decision border have significant influence, with largest gradients obtained after presentation of those vectors that lead to activations of some neurons near their threshold values, or to the outputs that are around the middle of the linear part of sigmoidal function.

In contrast to the neural learning support vector machines (SVMs, [1]) take into account initially all training vectors, but progressively the influence of those vectors that are far from decision borders is decreased and near the end of the training only a small percentage of vectors that are close to the decision hyperplane are left. This approach contributes not only to the increased speed near the end of training, but also to the higher accuracy that is finally achieved. Linear SVM algorithm with reasonably small margin discriminates between border vectors with greater precision than a perceptron. If the margin between the hyperplane and the vectors from two classes is small the long tails of sigmoidal output function contributing to the error function may shift the position of the hyperplane, and although the mean square error will decrease the number of classification errors may increase. We have observed such behavior [2] after initialization of the MLP network with parameters obtained from the linear discrimination analysis (LDA). Although MLPs converge quickly after such initialization the LDA hyperplane gave usually lower number of errors than the perceptron solution.

Author is with the Department of Informatics, Nicholaus Copernicus University, Grudziądzka 5, Toruń, Poland, http://www.phys.uni.torun.pl/kmk, and School of Computer Engineering, Nanyang Technological University, Singapore.

Only a few attempts to train neural networks using border vectors seem to be investigated so far. One way to select such vectors is to use distances between vectors of different classes [3]. For the two class problems, for each vector from the first class the closest vector from the second class is selected; then the process is repeated for the second class. A few closest vectors may be selected, and the selection process may be repeated several times to leave only the vectors that have close neighbors from the opposite class. For large databases this is costly, scaling with a square of the number of vectors, and it is not clear at which moment the switch from training using all vectors to training using just the border vectors should be made.

Sensitivity analysis in respect to input perturbations has been used [4], [5] to visualize and analyze decision borders. This is a form of active learning [6], in which the training algorithm has an influence on what part of the inputs space the information comes from. Zhang [7] has developed the selective incremental learning algorithm that starts from a random subset of the training set. After training the remaining examples available in the training set are evaluated, and those that give large error are added to the current training set. This algorithm has been tested on the binary majority function problem, non-linear function mapping and handwritten digit recognition, achieving significant accuracy gains. Röbel [8] has also described an incremental “dynamic selection algorithm" in which available patterns that give largest errors are incrementally added to the training set and evaluation of generalization is done using validation set. He has applied this algorithm to prediction of chaotic time series.

In this paper perhaps the simplest, but it seems that so far little explored, approach to active learning is investigated. Similarly as in the SVM approach [1] all data is initially used and after a few epochs vectors that do not contribute much to the next epoch training process are removed. The algorithm and properties of this approach are described in the next section. In the third section an illustration of the selection process is shown on a version of XOR data, and a few applications are presented showing the effectiveness of such training. The paper ends with a few conclusions.

II. ACTIVE LEARNING BY DYNAMIC SELECTION OF TRAINING VECTORS.

The Support Vector Neural Training (SVNT) algorithm presented here is a modification of the standard backpropagation procedure. The goal is to reduce the amount of training data, findng only those training vectors that are really needed to support the training process. The neural network $F_k = M_k(\mathbf{X};\mathbf{W})$, for $k = 1\dots K$ (the number of outputs is equal to the number of classes), is a vector mapping on $F_k \in [0,1]$ that depends on parameters $\mathbf{W}$. These parameters are updated after presentation of the training data $\mathcal{T}$, depending on the difference between the target output values and the achieved network output $Y_k - M_k(\mathbf{X};\mathbf{W})$:

$$\Delta W_{ij} = -\eta \frac{\partial E(\mathbf{W})}{\partial W_{ij}} = \eta \sum_{k=1}^{K} (Y_k - M_k(\mathbf{X};\mathbf{W})) \frac{\partial M_k(\mathbf{X};\mathbf{W})}{\partial W_{ij}} \quad (1)$$

If the difference $\varepsilon(\mathbf{X}) = \sum_k |Y_k - M_k(\mathbf{X};\mathbf{W})|$ is suffciently small the pattern $\mathbf{X}$ will have negligible influence on the training process. Patterns that are close to the decision borders may give significant errors and thus should be used for training. A forward propagation step (computationally inexpensive) before each training epoch may determine which vectors have a chance to influence the training procedure. Vectors for which $\varepsilon(\mathbf{X}) > \varepsilon_{\min}$ are included in the current training set for the next epoch, other vectors have no chance to exert significant influence.

The number of vector selected for training depends critically on the $\varepsilon_{\min}$ threshold. If it is too low most vectors will be included, and the savings will be minimal. If it is too high only a few vectors that lead to largest errors will be included. On some data where many neurons have to adjust their parameters to separate well defined clusters this may be a good strategy. However, for noisy data or for strongly overlapping clusters convergence may become oscilatory, with the number of vectors selected in each training epoch changing rapidly.

This $\varepsilon_{\min}$ parameter may be automatically adjusted to avoid strong oscillations. In the first training epoch $\varepsilon_{\min} = 0$ and all training vectors are used. Different schedules may be proposed to change this parameter in such a way that only useful support vectors will be left at the end of training. In the simplest case $\varepsilon_{\min}$ is increased by $\Delta\varepsilon = 0.01$ after every epoch in which the accuracy has been increased. Increasing $\varepsilon_{\min}$ should lead to dynamic reduction of the number of vectors, until only the necessary support vectors close to the decision borders are left. Increasing this value further will remove important support vectors and the accuracy will decrease. Since there is no guarantee that convergence will be monotonic it may be necessary to step back and decrease $\varepsilon_{\min}$ threshold.

The number of currently selected support vectors is a more sensitive indicator that important support vectors have been removed than accuracy. If this number increases after the next selection by more than 5% of the current number of vectors than $\varepsilon_{\min}$ should be set slightly lower to stabilize the iterative process. and therefore it is set back by $\Delta\varepsilon$. To reduce oscillations the value of $\Delta\varepsilon$ is then reduced dividing it by 1.2. After a few oscillations this value stabilizes, but if the initial value of $\Delta\varepsilon$ is too large some oscillations in the number of support vectors, MSE and classification accuracy may persist. The number of vectors used for training may still decrease in the iterative process even if $\varepsilon_{\min}$ will stabilize, because minimization of the error function leads to growing weights. If weights are very large sigmoidal functions have effectively steeper slopes. This comes from the fact that $\sigma(\mathbf{W}\cdot\mathbf{X} - \theta) = \sigma(\beta(\mathbf{W}'\cdot\mathbf{X} - \theta'))$, where $\beta = ||\mathbf{W}||$, $\mathbf{W}' = \mathbf{W}/||\mathbf{W}||$ and $\theta' = \theta/||\mathbf{W}||$. The network provides sharper decision borders, leaving fewer support vectors that have $\varepsilon(\mathbf{X}) > \varepsilon_{\min}$; in the limit for separable data clusters the number of such vectors may go to zero.

This works well if the number of mislabeled patterns, or noisy patterns, giving large errors, is not too large, but it is clear that such patterns end up as support vectors. If all training data is used they may have smaller influence than if only support vectors are used for training. This problem is solved by excluding from training also the patterns that give very large errors, keeping only those for which $\varepsilon(\mathbf{X}) \in [\varepsilon_{\min}, \varepsilon_{\max}]$. In practice to avoid new parameters $\varepsilon_{\max} = 1 - \varepsilon_{\min}$ has been used. The $\varepsilon_{\min}$ threshold determines the margin of the size $1 - 2\varepsilon_{\min}$, centered at 1/2, for the output values. This may be translated into a margin around the decision surface near some points $\mathbf{X}$, using the curvature of the sigmoidal functions in this area. Vectors falling into this margin are used as support vectors for training while those outside are assumed to be already correctly handled. Thus the SVNT algorithm selects best support vectors for training, cleaning the data at the same time.

The SVTN algorithm proceeds as follows:

```
1) Initialize the network parameters $\mathbf{W}$, set $\Delta\varepsilon = 0.01$,
   $\varepsilon_{\min} = 0$, set $SV = \mathcal{T}$.
   Until no improvement is found in the last $N_{\text{last}}$ iterations
   do
2) Optimize network parameters for $N_{\text{opt}}$ steps on $SV$ data.
3) Run feedforward step on $\mathcal{T}$ to determine overall accu-
   racy and errors, take $SV = \{\mathbf{X}|\varepsilon(\mathbf{X}) \in [\varepsilon_{\min}, 1-\varepsilon_{\min}]\}$.
4) If the accuracy increases:
   compare current network with the previous best one,
   choose the better one as the current best (take lower
   MSE if the number of errors is identical);
   increase $\varepsilon_{\min} = \varepsilon_{\min} + \Delta\varepsilon$.
5) If the number of support vectors $|SV|$ increases on more
   than $0.05 \cdot |SV|$
   decrease $\varepsilon_{\min} = \varepsilon_{\min} - \Delta\varepsilon$;
   decrease $\Delta\varepsilon = \Delta\varepsilon/1.2$.
```

MSE error should always be determined on the whole data set. Variants of this algorithm may include changes of basic parameters, more sophisticated schedules of parameter changes, for example increase of the number of iterations N_{opt} between support vector selections that may speed up the final convergence, and many others. Of course various thresholds may also be built in the backpropagation procedure itself, but in this paper only the simplest solution that does not require changes to the network optimization procedure is investigated.

Regularization is a commonly used technique to prevent overfitting of complex neural models and improve generalization [9]. Adding sum of squares of weights, or similar such terms, to the error function obviously decreases the value of weights. In effect the slopes of sigmoidal functions are rather small, influencing the margin very strongly. For a single network node $\varepsilon(\mathbf{X}) = |Y_k - \sigma(\mathbf{W}\cdot\mathbf{X} - \theta)|$ is therefore always close to 0.5. The margin around the front of the

$\sigma(\mathbf{W}\cdot\mathbf{X}-\theta)=0.5$ hyperplane where $\varepsilon(\mathbf{X})>\varepsilon_{\min}$ is rather large, sometimes covering the whole data range. Qualitatively the situation is the same in MLPs, so most vectors will be used for training. Large regularization term may significantly increase the total value of the cost function and slow down the convergence, but it may improve the final results.

In particular implementation used in the experiments described below the optimization procedure runs for a small number of iterations (typically $N_{\text{opt}}=2-50$), and then selection of support vectors performed. After the number of selected vectors stabilizes the number of iterations performed between reductions may be increased, or a stopping criterion for optimization may be used to converge to the particular solution for reduced data. After subsequent reduction of the training set another solution may be found. Thus the algorithm explores various solutions in the parameter space, keeping the core support vectors and changing small percentage of these vectors after every restart. Convergence at the later stages may therefore be far from monotonic, and it is worthwhile to wait for some number of iterations before stopping. Network with the lowest number of the training errors, and among those with lowest MSE error, is returned at the end and used on the test data.

III. Numerical experiments

Implementation of the SVNT algorithm has been made using Netlab package [10]. All MLP networks were trained using Scaled Conjugated Gradient (SCG) optimization procedure. No modification of the optimization algorithm itself was done, only some outer loops were added to control its use.

In order to see that SVNT algorithm is indeed capable of selecting correct support vectors noisy version of XOR data has been created, with Gaussian clouds around the corners and two additional vectors per cluster defining decision borders (Fig. 1). This data may be perfectly separated with two hyperplanes. Correct support vectors have been selected by the GhostMiner data mining package implementation of the SVM algorithm, using Gaussian and quadratic kernels [11]. Parameter tuning was necessary for Gaussian kernel (optimal parameters are determined in an automatic way using crossvalidation by this GhostMiner software). Some kernels (linear, cubical) could not find the optimal solution for the range of parameters investigated.

SVNT algorithm with selection of support vectors after every second iteration, and $\Delta\varepsilon=0.01$ converges to optimal solution in about 2/3 of the runs. The number of support vectors is rapidly reduced to 8, and stays at this level for some time (Fig.2). This stability results from the fact that the number of vectors available for training is very small, so only after a longer training weights become sufficiently large to reduce the margin around the decision border so that some of these vectors become also excluded (see Fig.2, right).

Increasing the number of iterations between restarts to $N_{\text{opt}}=5$ or more leads to convergence in almost all runs. Since the noisy XOR problem is quite simple taking larger N_{opt} values leads to convergence too fast giving $\varepsilon_{\min}$ no chance to increase to sufficiently large values (0.05 is usually

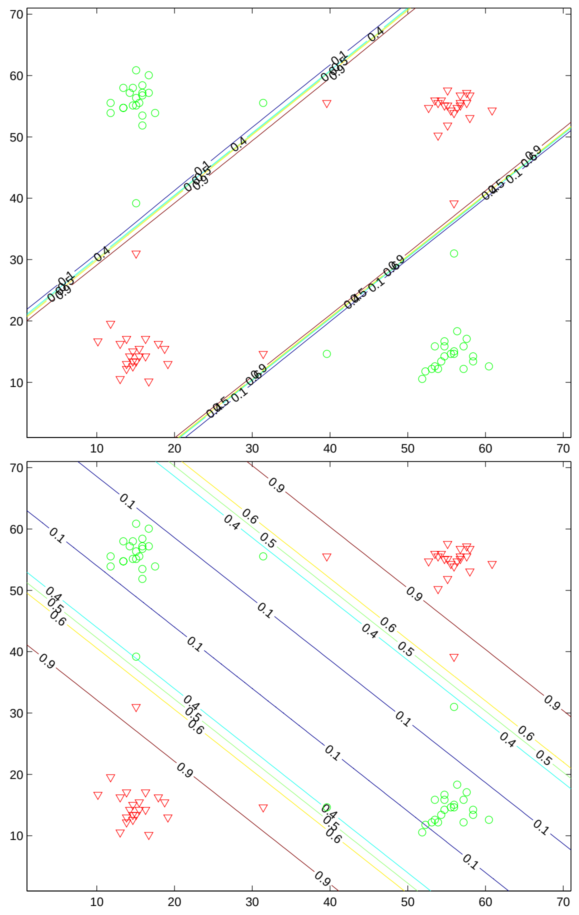

Fig. 1. Noisy XOR data with additional 8 vectors added to enforce unique borders; two perfect solutions to this problem, the lower one with wider margin achieved by adding regularization term.

sufficient) that enable reduction of the number of support vectors to the minimal number. Adding regularization term slows down the convergence, but results in more unsuccessful runs and increases the margin.

Satellite Image data consists of the multi-spectral values of pixels in the 3x3 neighborhoods in a small section (82x100) of an image taken by the Landsat Multi-Spectral Scanner. The intensities are one-byte numbers (0-255), the training set includes 4435 samples and the test set 2000 samples. Central pixel in each neighborhood is assigned to one of the six classes: red soil (1072), cotton crop (479), grey soil (961), damp grey soil (415), soil with vegetation stubble (470), and very damp grey soil (1038 training samples). This dataset is rather difficult to classify, with strong overlaps between some classes. Test set accuracies reported in the Statlog project [13] ranged between 71% (Naive Bayes method) to 91% for the k-Nearest Neighbor classifier. MLPs were at the level of 86%, and RBF at 88% accuracy on the test set.

This dataset has been re-analyzed with a number of methods available in the Ghostminer package [11]. Many other results for this dataset may be found in the Statlog book [13]. Best results (Table I) were achieved with the k-Nearest Neighbors classifier with small k (automatic selection using crossvalidation tests found optimal k=3), suggesting that rather

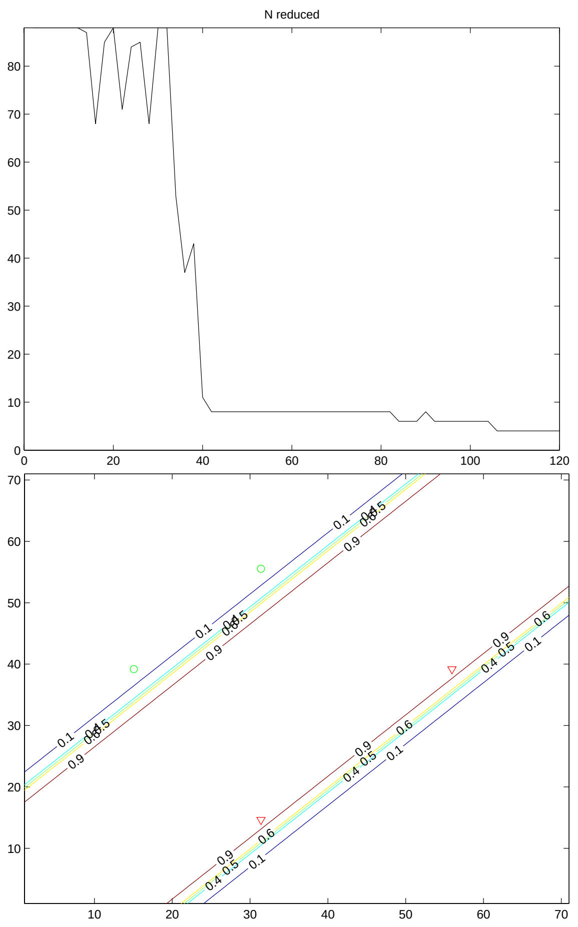

Fig. 2. Typical run on the noisy XOR data; left, the number of support vectors decreasing during iterations; right, the final solution with only support vectors shown.

complex decision borders are needed. From the description of this dataset one may conclude that dimensionality reduction is rather unlikely. Creating a network with 36 hidden nodes, one per each output, and adding a regularization term with $\alpha = 0.5$ to avoid overfitting rather, led to very good results. The maximum variance of the hidden node responses reached 0.85, but for 5 nodes it was of the order of 0.001, and for another 4 nodes it was below 0.1, showing that such network is slightly too large, but most neurons are fully utilized. Minimal MSE and minimal number of training errors has been achieved after 1300 iterations using all 4435 training vectors. The test accuracy of this network is 5% higher than the Statlog result for MLP network, and about 3% better than the best SVM solution. No other classification system applied to this data has obtained such good result.

Even higher accuracy may be achieved using SVNT algorithm using a fraction of all training vectors. Dependence of accuracy on the fixed ε threshold is shown in Figure 3. Best results are achieved with automatic determination of relevant threshold. In particular large classes 1 and 5 may be reduced to 10% without any degradation of accuracy, while some classes retain almost all their vectors. Confusion matrices in all runs looked very similar. With regularization at $\alpha = 0.5$ level no overfitting is observed, the training results is well correlated

TABLE I
RESULTS FOR THE SATELLITE IMAGE DATA. OPTIMAL PARAMETERS HAVE BEEN DETERMINED USING CROSSVALIDATION ON THE TRAINING SET.

System and parameters	Train accuracy	Test accuracy
MLP, 36 nodes, $\alpha = 0.5$+SVNT	96.5	91.3
MLP, 36 nodes, $\alpha = 0.5$	96.0	91.0
kNN, k=3, Manhattan	–	90.9
FSM neurofuzzy, learn 0.95	95.1	89.7
kNN, k=1, Euclidean	–	89.4
SVM Gaussian kernel (best)	91.6	88.4
RBF, Statlog result	88.9	87.9
MLP, Statlog result	88.8	86.1
C4.5 tree	96.0	85.0
SSV tree	90.9	84.3

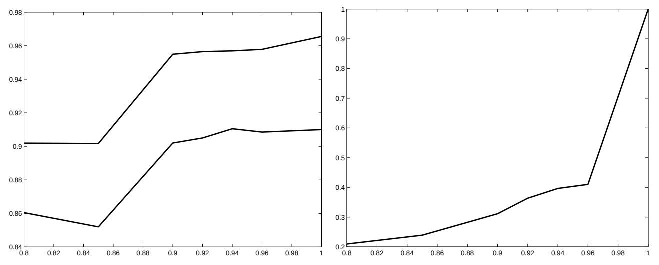

Fig. 3. Convergence of a network with 36 hidden neurons on the Satellite Image data for different thresholds ε: left, training (upper) and test (lower curve) accuracy; right, number of support vectors.

with the test results – the number of training errors in Fig. 4 goes below the number of test errors without any increase in the latter.

In various experiments, changing the schedule of ε reduction and the number of iterations between the reductions training accuracies as high as 96.5% were achieved, with corresponding test set results of 91.3%. In this case (see Fig. 4) the final number of support vectors was 2075, still rather large, because using 1507 vectors (for the final $\varepsilon = 0.935$) resulted with only slightly worse training results (157 instead of 154 errors), with small improvement of the test results (175 instead of 175 errors).

Strong overlap of classes for this dataset requires relatively large number of support vectors. Other data with strong imbalance between classes allow for much larger reduction of the support vector set. Hypothyroid dataset [12] has for a long time been a challenge to neural networks. This data has been collected during two years of real medical screening tests for thyroid diseases and therefore contains mostly healthy cases. Among 3772 cases there are only 93 cases of primary hypothyroid, and 191 of compensated hypothyroid, the remaining 3488 cases are healthy cases. 3428 cases are provided for testing, with similar class distribution 73, 177, and 3178. 21 attributes (15 binary, 6 continuous) are given, but only two of the binary attributes (on thyroxine, and thyroid surgery) contain any useful information [**?**], therefore the number of attributes used below has been reduced to eight.

This datasets presents quite different challenge. All kNN results are in this case rather poor; using all features they are

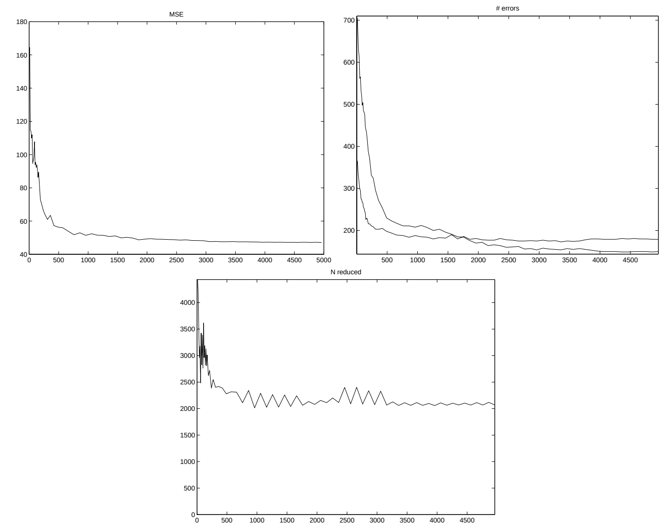

Fig. 4. Convergence of the same network with dynamic ε threshold determination: upper left – MSE, upper right – training (upper) and test (lower curve) accuracy; bottom, number of support vectors.

only slightly above the base rate, while using 8 features best test result is 97.3%. MLPs trained using all tricks of the trade (local learning rates, genetic optimization), and the cascade correlation algorithm still reach only 98.5% percent accuracy on the test set [16]. Best results (Table II) are obtained with a few optimized logical rules, or very simple decision trees [15], [14]. This shows the need of logical, sharp decision borders. In contrast to the Satellite Image problem there is no chance here to get good results using large, complex network with regularization. A small network with large weights, providing sharp decision borders, is needed. Unfortunately finding good solutions with gradient-based backpropagation methods for small networks is not easy, global optima of the cost function may be found in small, narrow wells in the parameter space. This problem may be remedied using global optimization techniques (see for example analysis in [17]). Here the simplest solution is used, multistart optimization, selecting from ten runs a network with lowest MSE after 2000 iterations and converging it until no improvement in MSE is found in the last 1000 iterations.

Networks with 3 neurons do not give good results, therefore at least 4 neurons are needed (this is also necessary to obtain 4 logical rules [14]). More neurons will lead to faster convergence. For example, one network with 12 neurons, optimized using SCG procedure that was run for 100 iterations and then run again, found a solution with zero errors on training and 40 errors on the test (98.83% accuracy) after less than 5000 iterations. This seems to be the best result obtained with MLP so far on this dataset. Using 4 neurons only in 20 runs the number of training errors has never reached zero, and the number of iterations needed for convergence was always quite large, reaching 50000 or more. The best result running SCG optimization repeatedly for 100 iterations was found after 15000 iterations, with MSE about three times larger than for 12 neurons (further training for more than 1000 iterations always returned higher MSE). Although 7 errors were left on the training set (99.81% accuracy) only 26 errors were made on the test set (99.24% accuracy). finding such good results may require many starts.

SVM algorithm with 8 selected features and Gaussian kernels requires optimization of the C and bias parameters, achieving best results for $C = 1000$ and bias$= 0.05$, found by crossvalidation on the training set. With the number of support vectors between 115 and 143 for each class versus two other classes the test result was 98.4%. Using Minkovsky kernel $e^{||\cdot||_a}$, optimizing the exponent ($a = 1$ works best) and $C = 1000$, bias$= 0.15$, leads to much better results, with 100% accuracy on the training set and 99.18% accuracy on the test (only 22 vectors were wrongly classified, and 6 vectors were left unclassified). The number of support vectors for each class versus two other classes ranged from 131 to 190.

Experiments with SVNT algorithm showed that with automatic determination of ε zero errors may be reached on the training set using the same type of training as before. A rather flat error curve has been reached after 27400 iterations, with only two errors on the training, 34 errors on the test set. The ε has stabilized at 0.048 and 67 support vectors were selected. If the training is continued further MSE drops very slowly, and after 43500 iterations zero training errors is reached, with 37 test errors and 45 support vectors (7, 21, and 17 respectively, from the three classes). The number of iterations may be large, but with less than 100 vectors for training and a very small network time to do such calculations is quite short. The final weights grow to quite high values, of the order of 100, showing that this problem has an inherent logical structure: forced to take binary decisions (sick or healthy) medical experts probably used in almost all cases threshold values for different tests. Convergence is quite slow because very little is gained by very large increase of the weights.

TABLE II
RESULTS FOR THE HYPOTHYROID DATASET.

Method	% train	% test	Ref.
C-MLP2LN rules	99.89	99.36	[14]
CART tree	99.79	99.36	[15]
SSV tree	99.79	99.33	[14]
MLP+SCG, 4 neurons	99.81	99.24	this work
SVM Minkovsky kernel	100.0	99.18	this work
MLP+SCG, 4 neurons, 67 SV	99.95	99.01	this work
MLP+SCG, 4 neurons, 45 SV	100.0	98.92	this work
FSM 10 rules	99.60	98.90	[14]
MLP+SCG, 12 neurons	100.0	98.83	this work
Cascade correlation	100.0	98.5	[16]
MLP+backprop	99.60	98.5	[16]
SVM Gaussian kernel	99.76	98.4	[14]
k-NN, k=1, 8 features	–	97.3	this work
Naive Bayes	97.0	96.1	[15]

These results represent probably the limit of accuracy that may be achieved for this data set. Significant improvement over previously published neural network results has been achieved with standard SCG batch optimization procedure. This type of training is not as effective as on-line training, as has been verified experimentally recently [18]. It should therefore be worthwhile to use the SVNT also in on-line learning procedures.

IV. Discussion and conclusions

Reduction of the size of the dataset used for training leads to increased accuracy and higher speed of learning. Although many algorithms for active learning exist the simplest one, based on rejection of training vectors that have no chance to contribute to the weight changes, seems to be quite effective. Using a conservative threshold $\varepsilon \approx 1$, leads only to speed increase, without affecting the convergence. In this paper the network training has been simply continued on reduced training data, but trivial modifications to the standard training algorithm may be done to reject training vectors in on-line learning.

The SVNT algorithm is especially useful in the cases when one class dominates over all others. The number of data samples from each class near the end of the training seems to be well balanced and may include only a small fraction of the original training set. This leads to significant improvement in classification of unbalanced data. In case of hypothyroid problem the number of support vectors was amazingly small, about 1.2% of the whole training data. Unfortunately these support vectors cannot be saved and used directly in other classification algorithms because they are selected from vectors that have been standardized using the whole training data. Thus they have to be remembered together with standardization coefficients that should be applied to all data. Training other systems on support vectors selected by SVNT and validating results on the remaining training data should be quite effective.

Depending on the rejection threshold, MSE, the number of errors and the number of selected vectors may oscillate. Large oscillations are damped by the dynamic rejection threshold update in the SVNT algorithm, but small oscillations may actually be useful. Reduction of the traing set introduces a stochastic element to the training, pushing the system out of the local minima. Comparing this approach to the algorithms based on evolutionary or on other global optimization algorithms it is clear that at the initial stages convergence is fast and that all good models need to share roughly the same characteristics. There is no reason to do much exploration of the solutions space, gradient-based methods are much faster. Near the end of the training gradients are small and wider exploration of the parameter space is worthwhile to fine tune the decision borders. This is exactly what SVNT does.

More empirical tests are needed to evaluate support vector neural training approach, but the ability to find correct support vectors (XOR data), handle multiclass problems with strong class overlaps (satellite image data), and excellent results on the unbalanced data (hypothyroid) show the potential of this idea. The algorithm may be applied to any type of feedforward network, not only MLPs. A few issues remain for further investigation, such as the optimal scheme for reduction of thresholds, implementation in on-line learning algorithms, using different neural optimization procedures, and improving convergence in the final stages of learning, when flat plateaus of the error function are reached. One idea to speed up convergence is based on principal components in the weight space, calculated from several epochs (M. Kordos, W. Duch, in preparation). Two principal components capture almost all variance, therefore the error surface may be mapped in two dimensions and larger jump towards minimum may be done. This should be especially effective in cases where the main improvements come from growth of network weights.

Acknowledgments

Dr Norbert Jankowski has developed the SVM implementation in the Ghostminer package, has done preliminary calculations on the hypothyroid data, and has helped me to use it.

References

[1] B. Schölkopf, and A.J. Smola, *Learning with Kernels. Support Vector Machines, Regularization, Optimization, and Beyond.* Cambridge, MA: MIT Press, 2001.

[2] W. Duch, R. Adamczak, and N. Jankowski, "Initialization and optimization of multilayered perceptrons". Proc. 3rd Conf. on Neural Networks and Their Applications, Kule, Poland, pp. 105-110, 1997.

[3] W. Duch, "Similarity based methods: a general framework for classification, approximation and association," Control and Cybernetics, vol. 29 (4), pp. 937-968, 2000.

[4] A. P. Engelbrecht, "Sensitivity Analysis for Decision Boundaries," Neural Processing Letters, vol. 10(3), pp. 253-266, 1999.

[5] A. P. Engelbrecht, "Sensitivity Analysis for Selective Learning by Feedforward Neural Networks," Fundamenta Informaticae, vol. 45(1), pp. 295-328, 2001.

[6] D. Cohn, L. Atlas, and R. Ladner, "Improving Generalization with Active Learning," Machine Learning, vol. 15, pp. 201-221, 1994.

[7] B-T. Zhang, "Accelerated Learning by Active Example Selection", International Journal of Neural Systems, vol. 5(1), pp. 67-75, 1994.

[8] A. Röbel, "The Dynamic Pattern Selection Algorithm: Effective Training and Controlled Generalization of Backpropagation Neural Networks", Technical Report, Institute für Angewandte Informatik, Technische Universität Berlin, pp. 497-500, 1994.

[9] C. Bishop, *Neural networks for pattern recognition*. Oxford: Clarendon Press, 1994.

[10] I. Nabnay, and C. Bishop, NETLAB software, Aston University, Birmingham, UK, 1997. http://www.ncrg.aston.ac.uk/netlab/

[11] W. Duch, N. Jankowski, K. Grąbczewski, A. Naud, and R. Adamczak, Ghostminer software, http://www.fqspl.com.pl/ghostminer/

[12] C.L, Blake, and C.J. Merz, UCI Repository of machine learning databases, http://www.ics.uci.edu/ mlearn/MLRepository.html. University of California, Irvine, Dept. of Information and Computer Science, 1998-2003.

[13] D. Michie, D.J. Spiegelhalter, and C.C. Taylor, *Machine Learning, Neural and Statistical Classification*. London, UK: Ellis Horwood, 1994.

[14] W. Duch, R. Adamczak, and K. Grąbczewski, "A New Methodology of Extraction, Optimization and Application of Crisp and Fuzzy Logical Rules," *IEEE Transactions on Neural Networks*, vol. 12, pp. 277–306, 2001.

[15] S.M. Weiss, and I. Kapouleas, "An empirical comparison of pattern recognition, neural nets and machine learning classification methods", in: *Readings in Machine Learning*, eds. J.W. Shavlik, T.G. Dietterich, Morgan Kauffman Publ, CA 1990

[16] W. Schiffman, M. Joost, and R. Werner, "Comparison of optimized backpropagation algorithms". Proc. of European Symposium on Artificial Neural Networks, De facto Publications, Brussels 1993, pp. 97-104

[17] Y. Shang, and B.W. Wah, "Global optimization for neural network training". IEEE Computer, vol. 29, pp. 45-54, 1996.

[18] D.R. Wilson, and T.R. Martinez, "The general inefficiency of batch training for gradient descent learning." Neural Networks, vol. 16, pp. 1429-1451, 2003.