Introducing GEMS – a Novel Technique for Ensemble Creation

Ulf Johansson[1], Tuve Löfström[1], Rikard König[1], Lars Niklasson[2]

[1]School of Business and Informatics, University of Borås, Sweden
[2]School of Humanities and Informatics, University of Skövde, Sweden
{ulf.johansson, tuve.lofstrom, rikard.konig}@hb.se
lars.niklasson@his.se

Abstract

The main contribution of this paper is to suggest a novel technique for automatic creation of accurate ensembles. The technique proposed, named GEMS, first trains a large number of neural networks (here either 20 or 50) and then uses genetic programming to build the ensemble by combining available networks. The use of genetic programming makes it possible for GEMS to not only consider ensembles of very different sizes, but also to use ensembles as intermediate building blocks which could be further combined into larger ensembles. To evaluate the performance, GEMS is compared to different ensembles where networks are selected based on individual test set accuracy. The experiments use four publicly available data sets and the results are very promising for GEMS. On two data sets, GEMS has significantly higher accuracy than the competing ensembles, while there is no significant difference on the other two.

Introduction

The primary goal when performing predictive modeling is to achieve high accuracy, i.e., a low error between the predicted value and the target value, when the model is applied to novel data. Although there are many available data mining techniques, Artificial Neural Networks (ANNs) are often used if there is no explicit demand requiring a transparent model. The motivation is that ANNs are known to often produce very accurate models in many diverse domains.

Within the research community it is, however, a well-known fact that the use of ensembles consisting of several models normally produces even higher accuracy, compared to single models; see e.g. the papers by Hansen and Salmon (1990) and Krogh and Vedelsby (1995). Despite this, the use of ensembles in applications is still limited. Two possible reasons for this are insufficient knowledge about the benefits of using ensembles and limited support in most data mining tools. In addition, even when ensembles are used, very simple designs are often preferred. A typical choice would be to train exactly five (or ten) ANNs with identical topology and simply average the output.

With this in mind, algorithms for constructing accurate ensembles should be of significant interest to both researchers and practitioners within the data mining community. The overall purpose of this paper is to suggest and evaluate a novel technique for automatic creation of ANN ensembles.

Background and related work

Although any algorithm for constructing ensembles must somehow determine ensemble members, the actual selection could be performed in many different ways. Standard techniques like bagging, introduced by Breiman (1996), and boosting, introduced by Shapire (1990), rely on resampling techniques to obtain different training sets for each of the classifiers. Both bagging and boosting can be applied to ANNs, although they are more common when using decision trees; see e.g. (Opitz and Maclin, 1999). Another option is to train each classifier independently and then either combine all classifiers or select a subset of classifiers to form the actual ensemble. Regardless of exactly how the actual creation is carried out, a very important part of all algorithms creating ensembles is how the evaluation of possible ensembles is performed.

Several approaches try to create ensembles by somehow applying genetic algorithms (GAs) to search for optimal ensembles. Zhou et al. (2001), (2002) proposed a method named GASEN, where several ANNs are trained before GAs are used to select an optimal subset of individual networks. In GASEN, the optimization is performed on individual ANNs and each ANN is coded (in the gene) as a real number denoting the benefit of including that ANN in the final ensemble. The optimization criterion (the fitness) is rather technical but boils down to accuracy on a hold-out (test) set. The number of ANNs in the ensemble can vary since all ANNs with strength values higher than a specific threshold (which is a pre-set parameter) are included in the ensemble.

Opitz and Shavlik (1996) proposed a method called ADDEMUP where the GA is used for creating new ANNs as parts of an ensemble. The size of the ensemble is predetermined and fixed. ADDEMUP uses a fitness function directly balancing accuracy against diversity, also using a test set.

We have recently proposed and evaluated a novel, but simpler, approach also based on GAs (Johansson, Löfström and Niklasson, 2005a). Here several individual ANNs are trained separately, on the same data set, and then GAs are used to directly find an accurate ensemble

from these ANNs. More specifically, each gene is represented as a sequence of zeroes and ones (a "bitstring") where the chromosomes correspond to a particular ANN. As expected a "1" would indicate that the specific ANN should be included in the ensemble. The optimization is performed on ensembles and the fitness is based directly on ensemble accuracy on training and/or test sets. The number of ANNs in the ensemble can vary since optimization is performed on the ensemble level. In that study we also evaluated numerous, more basic ways of creating ensembles, without the use of GAs. Although our novel technique performed best, an interesting result was that some extremely straightforward approaches came very close. More specifically; if we trained 50 ANNs with slightly different architectures (for details see the original paper) and used the ten ANNs with highest individual accuracy on the test set to form the ensemble, these ensembles turned out to have almost as high accuracy on the production set as the ones created using GAs.

With these results in mind, together with the fact that several (if not most) existing techniques use test set accuracy to select ensemble members, we decided to look into the importance of test set accuracy in the next study, (Johansson, Löfström and Niklasson, 2005b). Here the evaluation boiled down to whether the correlation between test set accuracy and production set accuracy is high enough to motivate its use as selection criterion. The somewhat surprising result was that the correlation between accuracy on one hold-out set (the test set) and another hold-out set (the production set) often was very low. As a matter of fact, in our experiments there was, in general, absolutely nothing to gain from using an ensemble with high test set accuracy compared to a random ensemble.

Method

In this section we first introduce a novel technique named GEMS (Genetic Ensemble Member Selection) for the creation of ANN ensembles. In the second part, we describe the details regarding the experiments conducted. Since GEMS consists of two steps, each requiring several design choices and parameters, we start with a brief description of the main characteristics.

In the first step of GEMS a number of ANNs are trained and stored in a pool. Each ANN uses a 1-of-C coding (i.e. a localist representation) so the number of output units is equal to the number of classes. The activation level of the output units for a specific ANN is termed its *result vector*. In the second step Genetic Programming (GP), is used to create the actual ensemble. When using GP, the ensembles are coded as (genetic) programs, each individual representing a possible combination of the available ANNs. More specifically; each ensemble is represented as a tree, where the internal nodes contain operators while the leaves must be either ANNs from the pool or (random) constants. At the moment, GEMS has only two operators; FACT and AVG. FACT is used to multiply a result vector with a constant while AVG averages the result vectors from its children.

It should be noted that this in fact means that GEMS builds ensembles using a mix of smaller ensembles and single ANNs as building blocks. Fig. 1 shows a GEMS ensemble coded in the tree format described above. This very small, sample, ensemble uses only three ANNs and the result is the average of ANN3 (multiplied with a factor 0.8) and the average of ANN1 and ANN2.

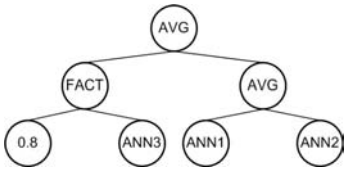

Fig. 1: A sample GEMS ensemble

GEMS

This study consists of two experiments. The number of available ANNs is 50 and 20, respectively. In both experiments, half of the ANNs has one hidden layer, while the other half has two hidden layers. Each ANN is a fully connected multi-layer perceptron network, with slightly randomized architecture. For an ANN with only one hidden layer the number of hidden units is determined from (1) below.

$$h = \sqrt{(v * c)} + \left\lfloor rand \cdot \sqrt{(v * c)} \right\rfloor \quad (1)$$

where v is the number of input variables and c is the number of classes. *rand* is a random number in the interval [0, 1]. For ANNs with two hidden layers the number of units in each hidden layer is determined from (2) and (3) below.

$$h_2 = \left\lfloor \frac{3}{2} c \right\rfloor \quad (2)$$

$$h_1 = \frac{3}{2} h \quad (3)$$

where c again is the number of classes and h is calculated using (1). All networks were trained with the Levenberg-Marquardt backpropagation algorithm.

When performing GP the two most important parameters are the representation language used and the fitness function. In this study we wanted to keep both as simple as possible. With this in mind the function and terminal sets are:

$F = \{AVG, FACT\}$

$T = \{ANN_1, ANN_2, \ldots, ANN_n, \Re\}$

where $\Re$ is a random number in the interval [0, 1].

$\Re$ is only used as a scaling factor together with the FAC operator.

The fitness function is based on three components. The first component adds a constant value to the fitness for each pattern in the training set that is correctly classified. The second component is identical to the first with the exception that it uses the test set. The third component is a penalty for longer programs that adds a negative constant value to the fitness for each part of the program.

We have chosen to base the fitness on both the test set and the training set since that strategy proved to be most successful in our first study regarding ensembles; see (Johansson, Löfström and Niklasson, 2005a). It is, however, far from trivial exactly how we should balance accuracy on test samples against accuracy on training samples. Whether to use a penalty for larger ensembles and, if so, the appropriate magnitude, is another tricky question. Obviously, the constants used in the fitness function will significantly affect the behavior of the GP. In this initial GEMS study we elected to set the constants to 1, 3 and 0.01 respectively; resulting in the fitness function given in (4).

$$f = \#correct_{train} + 3 \cdot \#correct_{test} - \frac{1}{100} size \qquad (4)$$

Crossover and mutation are performed as usual; with the addition that it is ensured that an offspring always is a correct program. With the representation language chosen, this means that the FACT node must have exactly one (random) constant child node. The other child could be a single ANN (a leaf node) or an AVG node. For an AVG node both children could be single ANN terminals, another AVG node or a FACT node.

With this representation language GEMS has the ability to combine the available ANNs in a huge number of ways. During evolution GEMS is actually using genetic blocks representing ensembles to create new ensembles. This extreme flexibility is a key property of the GEMS technique. Naturally the GP itself also has several parameters. The most important are given in Table 1 below.

Parameter	Value
Crossover rate	0.8
Mutation rate	0.001
Population size	500
Generations	500
Creation depth	8
Creation method	Ramped half-and-half
Elitism	Yes

Table 1: GP parameters

Experiments

The four data sets used in this study are all publicly available from the UCI Repository (Blake and Merz, 1998). For a summary of the data set characteristics, see Table 2. Cont is the number of continuous input variables. Cat is the number of categorical input variables and Total is the total number of input variables.

Data set	Instances	Classes	Cont	Cat	Total
CMC	1473	3	2	7	9
TAE	151	3	1	4	3
Tic-Tac-Toe	958	2	0	9	9
Vehicle	846	4	18	0	18

Table 2: Data sets

For each data set 40 runs were measured. Before each run the data set was randomly divided into four parts; a training set (50% of the patterns) used to train the ANNs, a validation set (10%) used for early stopping, a test set (20%) used to select ensembles and a production set (20%). The production set is of course a hold-out set used exclusively to measure performance. To evaluate GEMS we also created four competing ensembles on each run. Each competing ensemble consists of a fixed number of ANNs, based on test set accuracy; i.e. an ensemble consisting of five ANNs includes the best five individual ANNs, measured on the test set. The exact number of ANNs in the fixed ensembles is given in Table 3 below.

Ensemble	#ANNs in first experiment (total 20 ANNs)	#ANNs in second experiment (total 50 ANNs)
Quarter	5	13
Half	10	25
Three-quarter	15	39
All	20	50

Table 3: Number of ANNs in fixed ensembles

In this study, the output from a fixed ensemble is always the average of the output from all members. Since 1-of-C coding is used, the unit (or index in the result vector rather) with the highest (averaged) output finally determines the predicted class.

Results

Given the limited space, we elect to present results from only one data set and experiment in great detail. We choose the experiment with 50 ANNs and the Tic-Tac-Toe data set. All other results will be presented in summarized form only, as mean values. Table 4 and 5 show both test and production results from all 40 runs.

Run	Quarter	Half	Three-quarter	All	GEMS
1	0.865	0.849	0.839	0.833	0.901
2	0.875	0.859	0.844	0.833	0.917
3	0.880	0.870	0.875	0.859	0.953
4	0.865	0.870	0.865	0.854	0.901
5	0.870	0.865	0.849	0.844	0.906
6	0.859	0.854	0.844	0.833	0.932
7	0.828	0.818	0.802	0.792	0.891
8	0.906	0.911	0.891	0.891	0.938
9	0.865	0.865	0.849	0.823	0.927
10	0.917	0.911	0.927	0.917	0.948
11	0.943	0.911	0.880	0.859	0.969
12	0.927	0.891	0.865	0.849	0.964
13	0.844	0.844	0.823	0.818	0.917
14	0.880	0.870	0.870	0.859	0.901
15	0.911	0.891	0.880	0.870	0.948
16	0.906	0.891	0.865	0.844	0.917
17	0.859	0.844	0.818	0.818	0.901
18	0.917	0.891	0.854	0.839	0.938
19	0.927	0.896	0.885	0.885	0.958
20	0.896	0.859	0.844	0.839	0.943
21	0.859	0.844	0.828	0.833	0.901
22	0.932	0.901	0.896	0.885	0.948
23	0.859	0.849	0.854	0.828	0.906
24	0.911	0.891	0.880	0.859	0.953
25	0.870	0.870	0.849	0.839	0.932
26	0.922	0.911	0.901	0.880	0.948
27	0.854	0.844	0.818	0.802	0.906
28	0.948	0.917	0.859	0.839	0.964
29	0.875	0.849	0.844	0.844	0.922
30	0.880	0.854	0.849	0.828	0.922
31	0.891	0.859	0.833	0.813	0.927
32	0.880	0.849	0.823	0.802	0.917
33	0.859	0.849	0.854	0.849	0.901
34	0.865	0.859	0.859	0.854	0.917
35	0.917	0.896	0.880	0.865	0.953
36	0.922	0.911	0.875	0.865	0.974
37	0.849	0.839	0.818	0.807	0.917
38	0.938	0.911	0.901	0.875	0.974
39	0.922	0.917	0.896	0.885	0.953
40	0.922	0.906	0.896	0.865	0.958
Mean	**0.890**	**0.875**	**0.860**	**0.847**	**0.932**

Table 4: Tic-Tac-Toe test set results using 50 ANNs

In this experiment, GEMS ensembles have, on average, more than 4 % higher test set accuracy than the second best ensemble, which is the Quarter ensemble.

Run	Quarter	Half	Three-quarter	All	GEMS
1	0.811	0.811	0.811	0.805	0.853
2	0.847	0.826	0.816	0.821	0.874
3	0.821	0.789	0.779	0.768	0.874
4	0.847	0.847	0.837	0.826	0.853
5	0.842	0.816	0.821	0.821	0.853
6	0.858	0.853	0.842	0.842	0.889
7	0.811	0.821	0.800	0.789	0.853
8	0.889	0.879	0.879	0.868	0.900
9	0.858	0.858	0.853	0.837	0.853
10	0.889	0.874	0.853	0.853	0.916
11	0.932	0.895	0.853	0.853	0.942
12	0.847	0.853	0.821	0.811	0.837
13	0.858	0.837	0.832	0.816	0.900
14	0.816	0.795	0.805	0.795	0.832
15	0.853	0.847	0.837	0.842	0.863
16	0.816	0.816	0.800	0.795	0.842
17	0.895	0.895	0.879	0.874	0.905
18	0.858	0.847	0.847	0.842	0.874
19	0.879	0.842	0.816	0.795	0.900
20	0.853	0.842	0.805	0.821	0.905
21	0.837	0.853	0.821	0.821	0.868
22	0.884	0.884	0.863	0.863	0.900
23	0.821	0.816	0.811	0.800	0.853
24	0.816	0.795	0.779	0.774	0.842
25	0.853	0.842	0.842	0.847	0.884
26	0.905	0.879	0.874	0.874	0.863
27	0.858	0.863	0.826	0.816	0.884
28	0.900	0.895	0.874	0.863	0.958
29	0.863	0.868	0.842	0.858	0.863
30	0.842	0.811	0.789	0.774	0.884
31	0.916	0.916	0.900	0.884	0.932
32	0.911	0.858	0.832	0.837	0.911
33	0.947	0.921	0.926	0.900	0.932
34	0.858	0.847	0.842	0.853	0.853
35	0.900	0.874	0.837	0.847	0.942
36	0.837	0.832	0.826	0.805	0.858
37	0.842	0.821	0.805	0.805	0.879
38	0.895	0.900	0.879	0.874	0.921
39	0.895	0.884	0.874	0.863	0.884
40	0.921	0.895	0.879	0.879	0.926
Mean	**0.864**	**0.852**	**0.838**	**0.833**	**0.884**

Table 5: Tic-Tac-Toe production set results using 50 ANNs

On the production set, the accuracy of the GEMS ensembles is also considerably higher than all the competing ensembles.

From Table 4 it is very clear that GEMS is more than able to achieve high accuracy on the fitness data. Another illustration of this is shown in Fig. 2, where test set accuracy vs. production set accuracy are plotted for the GEMS and the Quarter ensemble.

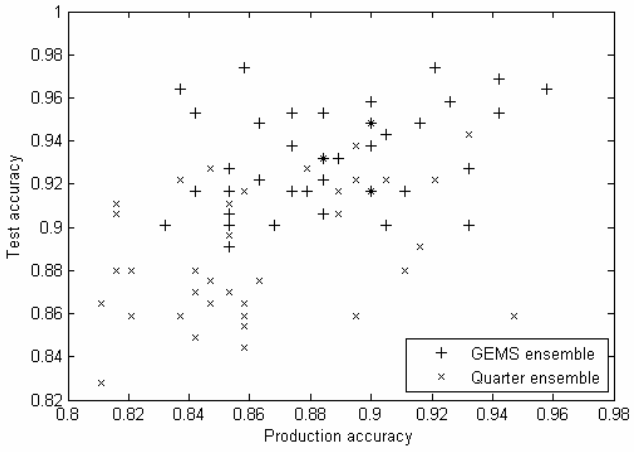

Fig. 2: Test set accuracy vs. production set accuracy

From Fig. 2 it is obvious that GEMS ensembles are much more accurate on the test set, compared to the other ensembles (here Quarter). As a matter of fact this holds for all data sets; see Table 6 and Table 7 below. For the TAE and Tic-Tac-Toe data sets, this advantage transfers to the production set. Unfortunately, this is not the case for CMC and Vehicle. We believe this to be an important observation, although we are not sure about the reason.

Table 6 summarizes the results for the first experiment (20 ANNs in pool) for all data sets. The value for each ensemble is the average over all 40 runs.

Data set	Quarter		Half		Three-quarter		All		GEMS	
	Test	Prod	Test	Prod	Test	Prod	Test	Prod	Test	Prod
CMC	.574	.552	.569	.555	.563	.555	.553	.552	.605	.554
TAE	.622	.519	.603	.526	.582	.528	.552	.529	.702	.548
Tic-Tac-Toe	.868	.856	.861	.850	.851	.844	.839	.839	.906	.865
Vehicle	.866	.837	.864	.845	.857	.845	.852	.845	.892	.845
Mean	**.732**	**.691**	**.724**	**.694**	**.713**	**.693**	**.699**	**.691**	**.776**	**.703**

Table 6: Results using 20 ANNs

On two of the data sets, TAE and Tic-Tac-Toe, GEMS clearly outperforms the other ensembles. A pair-wise t-test, between GEMS and Half, shows that the difference is statistically significant; the p-values for TAE and Tic-Tac-Toe are 0.024 and 0.008, respectively. On CMC and Vehicle all five techniques show very similar results.

Table 7 summarizes the results for the second experiment (50 ANNs) including the Tic-Tac-Toe results presented in detail above.

Data set	Quarter		Half		Three-quarter		All		GEMS	
	Test	Prod	Test	Prod	Test	Prod	Test	Prod	Test	Prod
CMC	.569	.551	.564	.552	.556	.550	.551	.550	.589	.552
TAE	.609	.503	.580	.503	.548	.508	.524	.513	.724	.536
Tic-Tac-Toe	.890	.864	.875	.852	.860	.838	.847	.833	.932	.884
Vehicle	.863	.847	.859	.847	.854	.845	.846	.845	.897	.846
Mean	**.733**	**.691**	**.719**	**.688**	**.704**	**.685**	**.692**	**.685**	**.786**	**.705**

Table 7: Results using 50 ANNs

The results are quite similar to the first experiment. The GEMS ensembles on TAE and Tic-Tac-Toe are again significantly better than all other ensembles. The p-values between GEMS and Quarter are for TAE 0.024 and for Tic-Tac-Toe $4.2 \cdot 10^{-7}$. The results for CMC and Vehicle are, however, once again almost identical.

A comparison of the results from the two experiments shows that there is no significant difference between using 20 or 50 ANNs, and that this holds for both the fixed ensembles and for GEMS.

The ensembles constructed by GEMS are extremely varied in both size and shape. The length penalty does, however, put a big pressure on the evolution, often resulting in remarkably small ensembles. As a matter of fact, it is not uncommon with ensembles using less than 5 ANNs. Below are sample ensembles (shown in the internal format used by GEMS) from two different runs. The first ensemble, shown in Fig. 3, is rather small and the second (shown in Fig. 4) is of about average size.

```
(avg(avg(ann2)(ann3))(avg(ann37)(ann8)))
```

Fig. 3: Small sample ensemble in GEMS internal format

```
(avg(avg(avg(*(0.898)(avg(ann7)(ann18)))(
*(0.874)(avg(ann15)(ann14))))(avg(*(0.227
)(avg(ann15)(ann18)))(*(0.717)(avg(ann15)
(ann4)))))(*(0.574)(avg(avg(ann6)(ann1))(
*(0.186)(ann13)))))
```

Fig. 4: Average-size sample ensemble

The possibility of using a specific ANN more than once in an ensemble is often utilized. The ensemble in Fig. 4 is one example, since it uses ann15 three times.

Conclusions

From the results it is obvious that GEMS is an interesting technique that should be further evaluated. In this first study we consistently choose very simple parameter settings. Despite this, GEMS clearly outperformed the other ensembles on two data sets. We believe that different parameter settings could significantly increase the performance of GEMS, although it might require fine-tuning for each data set. In addition the versatile nature of

GP makes it very easy to include new operators; e.g. a "majority vote" node.

Arguably the most important ability of GEMS is the possibility to use any combination of the available ANNs; including the option to use specific ANNs several times in one ensemble. Perhaps it would be more correct to describe GEMS as a Meta ensemble builder, since its building blocks in fact are ensembles.

Discussion and future work

First of all it must be noted that GEMS, in this study, is only compared to other ensembles. Very often novel ensemble techniques are instead compared to single models. To us it is, however, evident that the use of an ensemble is clearly superior to single models and therefore such comparisons are left out.

On the other hand, it seems to be extremely hard to come up with a technique that is always able to obtain ensembles significantly better than straightforward choices. Part of this is probably due to the fact that test set accuracy does not seem to be the silver bullet it is often assumed to be. As a matter of fact, the standard procedure of using test set accuracy when comparing models must be questioned. We all agree that the overall goal is to achieve high accuracy on unseen data, so naturally the best possible test seems to be to measure exactly that, accuracy on unseen data. This reasoning, however, has a devious shortcoming; the real issue is how a model chosen from accuracy on a test set would perform on *yet novel data*. If we use a test set to somehow choose one model over another, the underlying assumption must be that there is a high correlation between accuracy on that test set and accuracy on another set of unseen data; i.e. the production set. If this assumption does not hold, there is obviously little to gain from using a test set as a basis for ensemble construction.

With this in mind, one very interesting observation is the fact that although GEMS consistently had much higher accuracy on the test set (compared to the other ensembles) this property was not always preserved in the production set. Even though the fitness function used all available data (i.e. both training and test data) and a length penalty was used to encourage smaller ensembles, the most probable explanation is that the GP has overfit the test data. Just to iterate this important point; at the moment GEMS will always have very high accuracy on the part of the data set covered by the fitness function, but this does not necessarily carry over to the production set. How to deal with this problem is the top priority for future studies. At the moment we are considering two different strategies.

The first is the very straightforward choice to dispose of the test set altogether. We believe that the best use of the data set might be to use all available data for both ANN training and GP evolution. One micro technique to enforce some diversity among the networks could be to train each ANN using only part of the available training data (e.g. 70%) and randomize the exact patterns for each network.

The second strategy we consider is to change the GP training regime to avoid overspecialization on a specific part of the data. One option is to use something similar to standard boosting and another is to constantly alter the fitness set by randomly adding and removing data patterns between generations. Using either of these regimes should favor more general ensembles, and hopefully that should carry over to the production set.

Acknowledgement
This work has partly been made possible by a grant from the Knowledge Foundation, Sweden, to support a research program on information fusion and data mining.

References

C. L. Blake and C. J. Merz. *UCI Repository of machine learning databases*. University of California, Department of Information and Computer Science, 1998.

L. Breiman, 1996. Bagging predictors. *Machine Learning*, 24(2), pp. 123-140.

L. K. Hansen and P. Salamon, 1990. Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(10), pp. 993-1001.

U. Johansson, T. Löfström and L. Niklasson, 2005a. Obtaining Accurate Neural Network Ensembles. *International Conference on Computational Intelligence for Modelling Control and Automation – CIMCA 2005*. In press.

U. Johansson, T. Löfström and L. Niklasson, 2005b. Accuracy on a Hold-out Set: Neither the Goal nor the Way. Under review.

A. Krogh and J. Vedelsby, 1995. Neural network ensembles, cross validation, and active learning. *Advances in Neural Information Processing Systems*, Volume 2, pp. 650-659, San Mateo, CA, Morgan Kaufmann.

D. Opitz and R. Maclin, 1999. Popular ensemble methods: an empirical study. *Journal of Artificial Intelligence Research*, 11, pp. 169-198.

D. Opitz and J. Shavlik, 1996. Actively searching for an effective neural-network ensemble. *Connection Science*, 8(3/4), pp. 337-353.

R. Shapire, 1990. The strength of weak learnability. *Machine Learning*, 5(2), pp. 197-227.

Z.-H. Zhou, J.-X. Wu, Y. Jiang and S.-F. Chen, 2001. Genetic algorithm based selective neural network ensemble. *17th International Joint Conference of Artificial Intelligence*, vol. 2, pp. 797-802, Seattle, WA.

Z.-H. Zhou, J.-X. Wu, and W. Tang, 2002. Ensembling Neural Networks: Many Could Be Better Than All, *Artificial Intelligence*, vol. 137, no. 1-2, pp. 239-263, Elsevier.