Fundamenta Informaticae XXI (2001) 1001–1028
IOS Press

Sensitivity Analysis for Selective Learning by Feedforward Neural Networks

AP Engelbrecht
Department of Computer Science
School of Information Technology
University of Pretoria
Pretoria, South Africa
engel@driesie.cs.up.ac.za

Abstract. Research on improving the performance of feedforward neural networks has concentrated mostly on the optimal setting of initial weights and learning parameters, sophisticated optimization techniques, architecture optimization, and adaptive activation functions. An alternative approach is presented in this paper where the neural network dynamically selects training patterns from a candidate training set during training, using the network's current attained knowledge about the target concept. Sensitivity analysis of the neural network output with respect to small input perturbations is used to quantify the informativeness of candidate patterns. Only the most informative patterns, which are those patterns closest to decision boundaries, are selected for training. Experimental results show a significant reduction in the training set size, without negatively influencing generalization performance and convergence characteristics. This approach to selective learning is then compared to an alternative where informativeness is measured as the magnitude in prediction error.

Keywords: Sensitivity Analysis, Dynamic Pattern Selection, Decision Boundaries, Pattern Informativeness, Feedforward Neural Networks

1. Introduction

In the words of Ockham, *"What can be done with fewer is done in vain with more"*, unnecessarily complex models should not be preferred to simpler ones - a very intuitive principle. In this paper, a neural network (NN) model is viewed as the NN architecture and the data used for learning. Model selection is then viewed as the process of designing an optimal NN architecture as well as the implementation of techniques to make optimal use of the available training data. Following from Ockham's philosophy is a preference then for both simple NN architectures and optimized training data. Usually, model selection techniques address only the question of which architecture best fits the task. This paper explores the question of which data best describes the task, and develops a sensitivity analysis data selection technique to make optimal use of the available training data. The paper specifically focusses on selective learning for feedforward neural networks.

Training data consists of input-target vector pairs, which is only a finite sample from the distribution describing the input space. The objective of NN training is to find a good approximation to the function that relates input vectors to corresponding target vectors, not only for the training space, but for the entire input space. That is, the trained NN should generalize well. This objective is achieved by iteratively adjusting weights using some optimization algorithm.

Gradient descent is one of the most popular optimization techniques used for NN training, resulting in the widely used backpropagation (BP) neural network [1]. Although backpropagation neural networks (BPNN) have been used successfully in many applications, they may suffer from problems inherent to gradient descent optimization. That is, convergence of standard BPNNs tends to be slow, and they often yield local optimum solutions. Consequently, generalization performance is reduced.

Much research has been done to improve the generalization performance and training time of NNs. Current research mostly concentrates on the optimal setting of initial weights [2, 3], optimal learning rates and momentum [4, 5, 6, 7], finding optimal NN architectures using pruning techniques [8, 9, 10, 11, 12, 13] and construction techniques [14, 15, 16], sophisticated optimization techniques [17, 18, 19, 20, 21, 22], and adaptive activation functions [23, 24, 25]. This paper presents an alternative approach to improve generalization and training time, i.e. *active learning using sensitivity analysis*.

Standard error back-propagating NNs are *passive learners*. These networks passively receive information about the problem domain, randomly sampled to form a fixed size training set. Random sampling is believed to reproduce the density of the true distribution. However, more gain can be achieved if the learner is allowed to use current attained knowledge about the problem to guide the acquisition of training examples. As passive learner, a NN has no such control over what examples are presented for learning. The NN has to rely on the teacher (considering supervised learning) to present informative examples.

The generalization abilities and convergence time of NNs are much influenced by the training set size and distribution. To generalize well, the training set must contain enough information to learn the task. Here lies one of the problems in model selection: the selection of concise training sets. Without prior knowledge about the learning task, it is very difficult to obtain a representative training set. Theoretical analyses provide a way to compute worst-case bounds on the number of training examples needed to ensure a specified level of generalization. A widely used theorem concerns the Vapnik-Chervonenkis (VC) dimension [26, 27, 28, 29]. This theorem states that the generalization error $\mathcal{E}_G$ of a learner with VC-dimension d_{VC} trained on P random examples will, with high confidence, be no worse than a bound of order d_{VC}/P. For NN learners, the total number of weights in a one hidden layer network is used as an estimate of the VC-dimension. This means that the appropriate number of examples to ensure an $\mathcal{E}_G$ generalization is approximately the number of weights divided by $\mathcal{E}_G$.

The VC-dimension provides overly pessimistic bounds on the number of training examples, often leading to an overestimation of the required training set size [27, 29, 30, 31, 32]. Experimental results have shown that acceptable generalization performances can be obtained with training set sizes much less than that specified by the VC-dimension [27, 31]. Cohn and Tesauro show that for experiments conducted, the generalization error decreases exponentially with the number of examples, rather than the $1/P$ result of the VC bound [27]. Sung and Niyogi derive a theorem which states that their Integrated Mean Squared Error active learning algorithm requires at most $\frac{1}{2}\ln(1/12\mathcal{E})$ examples to guarantee a NN output uncertainty less than $\mathcal{E}$ with high probability greater than $1 - \delta$. This is in contrast to passive learners needing at least $\frac{1}{\sqrt{48\mathcal{E}}}\ln(1/\delta)$ examples [33]. Experimental results by Lange and Männer show that more training examples do not necessarily improve generalization [34]. In their paper, Lange and Männer introduce the notion of a critical training set size. Through experimentation they found that examples beyond this critical size do not improve generalization, illustrating that an excess patterns have no real gain.

While enough information is crucial to effective learning, too large training set sizes may be of disadvantage to generalization performance and training time [32, 35]. Redundant training examples may be from uninteresting parts of input space, and do not serve to refine learned weights - it only introduces unnecessary computations, thus increasing training time. Furthermore, redundant examples might not be equally distributed, thereby biasing the learner.

The ideal then, is to implement structures to make optimal use of available training data. That is, to select for training only informative examples, or to present examples in a way to maximize the decrease in training and generalization error. To this extend, active learning algorithms have been developed.

An overview of active learning is presented in section 2, while section 3 introduces the sensitivity analysis selective learning algorithm. The concept of pattern informativeness is defined in section 3.1, and section 3.2 discusses the approach of pattern selection around decision boundaries. A mathematical model of selective learning based on sensitivity analysis is presented in section 3.3, and an algorithm is given in section 3.4. Section 3.5 compares the complexity of the selective learning algorithm with normal fixed set learning. Section 4 summarizes experimental results to illustrate the efficiency of the selective learning algorithm. SASLA is compared to fixed set learning and an error selection approach. The effects of the SASLA selection constant is also investigated.

2. Active Learning

Cohn, Atlas and Ladner define *active learning* (also referred to in the literature as dynamic pattern selection, example selection, sequential learning, query-based learning) *as any form of learning in which the learning algorithm has some control over what part of the input space it receives information* [36]. An active learning strategy allows the learner to dynamically select training examples, during training, from a candidate training set. The learner capitalizes on current attained knowledge to select examples that are most likely to solve the problem, or that will lead to a maximum decrease in error. Rather than passively accepting training examples from the teacher, the network is allowed to use its current knowledge about the problem to have some deterministic control over training examples, and to guide the search for informative patterns. By adding this functionality to a NN, the network changes from a passive learner to an active learner.

With careful dynamic selection of training examples, shorter training times and better generalization may be obtained. A reduction in the number of training examples reduces the total number of learning computations. Provided that the added complexity of the example selection method does not exceed this reduction in training computations, training time will be reduced [33, 32, 37]. Generalization can potentially be improved using active learning algorithms, provided that selected examples contain enough information to learn the task. Cohn [38] and Cohn, Atlas and Ladner [36] show through average case analyses that the expected generalization performance of active learning is significantly better than passive learning. Seung, Opper and Sompolinsky [39], Sung and Niyogi [33] and Zhang [32] report similar improvements in using active learning models. Results presented by Seung, Opper and Sompolinsky indicate that generalization error decreases more rapidly for active learning than for passive learning [39].

This paper identifies two main approaches to active learning, i.e. *incremental learning* and *selective learning*. Incremental learning starts training on an initial subset of a candidate training set. During training, at specified selection intervals (e.g. after a specified number of epochs, or when the error on the current training subset no longer decreases), further subsets are selected from the candidate examples using some criteria or heuristics (based on the current knowledge of the network), and added to the training set. The training set consists of the union of all previously selected subsets, while examples in selected subsets are removed from the candidate set. Thus, as training progresses, the size of the candidate set decreases while the size of the actual training set grows.

In contrast to incremental learning, selective learning selects at each selection interval a new training subset from the original candidate set, using the current knowledge of the network about the problem domain. Selected patterns are not removed from the candidate set. At each selection interval, all candidate patterns have a chance to be selected. The subset is selected and used for training until some convergence criteria on the subset is met (e.g. a specified error limit on the subset is reached, the error decrease per iteration is too small, the maximum number of epochs allowed on the subset is exceeded). A new training subset is then selected for the next training period. This process repeats until the NN is trained to satisfaction.

The main difference between these two approaches to active learning is that no examples are discarded by incremental learning. In the limit, all examples in the candidate set will be used for training. With selective learning, training starts on all candidate examples, and examples are discarded as training progresses.

Not much research has been done in selective learning. Hunt and Deller developed Selective Updating [37], where patterns that exhibit a high influence on weights, i.e. patterns that causes the largest changes in weight values, are selected from the candidate set and added to the training set. Patterns that have a high influence on weights are selected at each epoch by calculating the effect that patterns have on weight estimates. These calculations are based on matrix perturbation theory, where an input pattern is viewed as a perturbation of previous patterns. If the perturbation is expected to cause large changes to weights, the corresponding pattern is included in the training set. Selective Updating has the drawback of assuming uncorrelated input units, which is often not the case for practical applications.

Another approach to selective learning is simply to discard those patterns that have been classified correctly [40, 41]. The effect of such an approach is that the training set will include those patterns that lie close to decision boundaries. If the candidate set contains outlier patterns, these patterns will, however, also be selected. This error selection (ES) approach therefore requires a robust estimator (objective function) to be used in the case of outliers.

While only a few selective learning strategies have been developed for feedforward neural networks, several *training set manipulation* strategies have been developed. Training set manipulation strategies pre-process the training set to assign a specific order in which patterns will be selected for learning. This order is maintained during training, and does not change dynamically. No knowledge of the learner is used to perform this selection of patterns. What follows next is an overview of such training set manipulation techniques.

Ohnishi, Okamoto and Sugie suggested a method called Selective Presentation where the original training set is divided into two training sets. One set contains typical patterns, and the other set contains confusing patterns [42]. With “typical pattern” the authors mean a pattern far from decision boundaries, while “confusing pattern” refers to a pattern close to a boundary. The two training sets are created once before training. Generation of these training sets assumes prior knowledge about the problem, i.e. where in input space decision boundaries are. In many practical applications such prior knowledge is not available, thus limiting the applicability of this approach. The Selective Presentation strategy alternately presents the learner with typical and then confusing patterns. (The approach can be compared to boosting neural networks where later learners in an ensemble concentrates more on difficult patterns as identified by previously trained networks.) The learner therefore has no control over the patterns presented for training, and the two sets remain fixed during training. Selective Presentation do not adhere to the definition of active learning, and is not viewed as an active learning algorithm.

Kohara developed Selective Presentation Learning for forecasting applications [43]. Before training starts, the algorithm generates two training sets. The one set contains all patterns representing large next-day changes, while patterns representing small next-day changes are contained in the second set. Large-change patterns are then simply presented more often than small-change patterns (similar to Selective Presentation). Again, the learner plays no role in the pattern selection process, and each training set remains fixed during training. Selective Presentation Learning does not adhere to the definition of active learning.

Slade and Gedeon [44] and Gedeon, Wong and Harris [45] proposed Bimodal Distribution Removal, where the objective is to remove outliers from training sets during training. Frequency distributions of pattern errors are analyzed during training to identify and remove outliers. Although the NN uses current attained knowledge to prune outliers from the training set, this thesis does not consider Bimodal Distribution Removal as an active learning algorithm. It is rather a training set filtering algorithm. The NN still trains on all non-outlier training patterns whether they are informative or not. If the original training set contains no outliers, the method simply reduces to FSL with the added complexity of analyzing an error frequency distribution at each epoch.

Cloete and Ludik have done extensive research on *training strategies* [46]. Firstly, they proposed Increased Complexity Training where a NN first learns easy problems, and then the complexity of the problem to be learned is gradually increased The original training set is split into subsets of increasing complexity before training commences. A drawback of this method is that the complexity measure of training data is problem dependent, thus making the strategy unsuitable for some tasks. Secondly, Cloete and Ludik developed *incremental training strategies*, i.e. Incremental Subset Training and Incremental Increased Complexity Training. In Incremental Subset Training, training starts on a random initial subset. During training, random subsets from the original training set are added to the actual training subset. Incremental Increased Complexity Training is a variation of Increased Complexity Training, where the complexity ranked order is maintained, but training is not done on each complete complexity subset. Instead, each complexity subset is further divided into smaller random subsets. Training starts on an initial subset of a complexity subset, and is incrementally increased during training. Finally, Delta Training Strategies were proposed. With Delta Subset Training examples are ordered according to inter-example distance, e.g. Hamming or Euclidean distance. Different strategies of example presentations were investigated: smallest difference examples first, largest difference examples first, and alternating difference. The training strategies proposed by Cloete and Ludik do not adhere to the definition of active learning. The learner has no control over the training subsets created, since subsets are created before training starts.

Approaches to selective learning have also been developed for ensemble networks. Boosting is one such selective learning approach, where trained members of the ensemble filter patterns into easy and hard patterns [47]. New, untrained members of the ensemble then focus more on the hard patterns as selected by previously trained members. While the boosting strategy does not adhere to the definition of active learning (which states that the learner itself has control over which examples to train on), it is a form of dynamic pattern selection. Boosting networks is rather a cooperative learning strategy where the knowledge of already trained networks is used to provide untrained networks with their training patterns. The pattern selection rationale of boosting networks is similar to the error selection approach for feedforward networks, where emphasis is also given to hard patterns.

Selective learning strategies have not just been developed for feedforward neural networks, but also for other classification algorithms. For example, the C5 decision tree algorithm includes a windowing strategy to perform pattern selection in cases of extremely large data bases [48]. Support vector machines perform selective learning by finding the support vectors to define the separating hyperplanes through a quadratic programming approach [49].

Research on incremental learning for feedforward neural networks is more abundant than for selective learning. Most current incremental learning techniques have their roots in information theory, adapting Fedorov's optimal experiment design for NN learning [33, 38, 50, 51, 52]. The different information theoretic incremental learning algorithms are very similar, and differ only in whether they consider only bias, only variance, or both bias and variance terms in their selection criteria. These approaches are computationally very expensive, due to the required inversion of the Hessian matrix. Approximations to the information theoretical incremental learning algorithms have been developed by Zhang [32] and Röbel [31], where patterns with the largest error, i.e. the largest $\sum_{k=1}^{K}(t_k^{(p)} - o_k^{(p)})^2$, where $t_k^{(p)}$ and $o_k^{(p)}$ are respectively the target and output value of output unit k, are selected from the candidate set. Cohn, Atlas and Ladner [36] and Hwang, Choi, Oh and Marks [55] developed incremental learning algorithms for classification problems where patterns are selected from a region around decision boundaries.

This paper presents a selective learning algorithm which uses a sensitivity analysis of the NN output with respect to small parameter perturbations to dynamically select the most informative patterns for training. Only classification problems are considered, in which case the most informative patterns are those closest to decision boundaries. Without loss of generality, a three layer feedforward NN is assumed throughout this paper, with differentiable activation functions - sigmoid functions in this case.

3. Selective Learning using Sensitivity Analysis

Selective learning is an active learning strategy that effectively "prunes" the original training set during training. The NN uses its current learned knowledge to select at each selection interval a subset of informative patterns from the candidate training set. Training commences on the training subset until subset termination criteria are triggered, upon which a new training subset is selected, from the original candidate set.

Selective learning algorithms certainly make sense for application to classification problems only. In classification problems, a pattern can be identified as being classified correctly or not. If the learner is already sure of the classification of a pattern, there is no need to re-learn that pattern. However, during the learning process, the learner may become uncertain about a previously correct classification, in which case the corresponding pattern should be brought back into the training subset. A selective learning algorithm should therefore have a good understanding of what information must be used for training, and what information can be overlooked. It certainly makes sense that patterns which are most likely to help the NN solve the problem must be preserved during training. For classification problems, the objective of NN training is to find optimum decision boundaries in input space that give good generalization. The patterns that are more likely to contribute to this objective are patterns in a region close to a boundary - referred to as the *region of uncertainty* [36, 50]. In fact, studies have shown that training on patterns near boundaries generalizes better than networks trained on the same number of randomly chosen examples [32, 36, 53, 54, 55, 42].

This section presents the Sensitivity Analysis Selective Learning Algorithm (SASLA), which uses sensitivity analysis to select patterns in the region of a decision boundary. First order derivatives of the output units with respect to input units are used to determine how close a pattern lies to a decision boundary. Patterns which lie closest to the decision boundaries, which are the most informative patterns, are selected for training.

3.1. Pattern Informativeness

This section introduces the basic idea of the sensitivity analysis selective learning algorithm (SASLA). SASLA is based on, and built upon, the concept of *pattern informativeness*. A pattern that has a negligible effect on the NN outputs is said to be uninformative for learning purposes, while informative patterns have a strong influence on the NN outputs.

Definition 3.1. Pattern Informativeness: Define the informativeness of a pattern as the sensitivity of the NN output vector to small perturbations in the input vector. Let $\Phi^{(p)}$ denote the informativeness of pattern p. Then,

$$\Phi^{(p)} \doteq ||\vec{S}_o^{(p)}|| \tag{1}$$

where $\vec{S}_o^{(p)}$ is the output sensitivity vector for pattern p (defined in equation (3)), and $|| \bullet ||$ is any suitable norm.

This study suggests the maximum-norm,

$$\Phi_\infty^{(p)} = ||\vec{S}_o^{(p)}||_\infty = \max_{k=1,\cdots,K}\{|S_{o,k}^{(p)}|\} \tag{2}$$

where $S_{o,k}^{(p)}$ refers to the sensitivity of a single output unit o_k to changes in the input vector $\vec{z}$. In equations (1) and (2), the output sensitivity vector is defined as

$$\vec{S}_o^{(p)} = ||S_{oz}^{(p)}|| \tag{3}$$

where $S_{oz}^{(p)}$ is the output-input layer sensitivity matrix. Assuming sigmoid activation functions in the hidden and output layers, each element $S_{oz,ki}^{(p)}$ of the sensitivity matrix is computed using

$$S_{oz,ki}^{(p)} = (1 - o_k^{(p)})o_k^{(p)} \sum_{j=1}^{J} w_{kj}(1 - y_j^{(p)})y_j^{(p)} v_{ji} \tag{4}$$

where w_{kj} is the weight between output unit o_k and hidden unit y_j, v_{ji} is the weight between hidden unit y_j and input unit z_i, $o_k^{(p)}$ is the activation value of output o_k, $y_j^{(p)}$ is the activation of hidden unit y_j, and J is the total number of hidden units (including a bias unit to the output layer). Suitable norms for calculating the output sensitivity vector are the sum-norm, or the Euclidean-norm. That is, for each element k of $\vec{S}_o^{(p)}$,

$$S_{o,k}^{(p)} = ||S_{oz}^{(p)}||_1 = \sum_{i=1}^{I} |S_{oz,ki}^{(p)}| \tag{5}$$

or

$$S_{o,k}^{(p)} = ||S_{oz}^{(p)}||_2 = \sqrt{\sum_{i=1}^{I} (S_{oz,ki}^{(p)})^2} \tag{6}$$

where I is the total number of input units (including a bias unit to the hidden layer).

Using definition 3.1 and equation (2), a pattern is considered informative if any one, or more, of the output units is sensitive to small perturbations in the input vector. The larger the value of $\Phi_\infty^{(p)}$, the more informative is pattern p. To illustrate this idea, assume gradient descent is used to find optimal weight values. Then,

$$\Delta w_{kj}, \Delta v_{ji} \propto (t_k^{(p)} - o_k^{(p)}) \tag{7}$$

where $t_k^{(p)}$ is the target value output unit o_k for pattern p. Each new pattern can be viewed as a perturbation of a previously presented pattern. Let $\Phi_\infty^{(p)} = |S_{o,k}^{(p)}|$. Then, if $\Phi_\infty^{(p)}$ is large, the output value of o_k is significantly influenced, and the value of $(t_k^{(p)} - o_k^{(p)})$ changes significantly from the previous presentation. On the other hand, if $\Phi_\infty^{(p)}$ is small, no significant change in the activation value of $o_k^{(p)}$ will occur from the previously presented pattern. That is, the value of $(t_k^{(p)} - o_k^{(p)})$ does not change much, making pattern p an insignificant contributor to the determination of the gradient direction - and therefore being uninformative to the learning process.

3.2. Pattern Selection around Decision Boundaries

The objective of a NN classifier is to construct optimal decision boundaries over input space. Active learning algorithms which sample from a region around decision boundaries have been shown to refine boundaries, resulting in improved generalization performance. Paramount to the success of decision boundary active learning algorithms, is the method used to detect boundaries - if too complex, the model will be impracticle.

Engelbrecht developed a computationally inexpensive approach to locate and visualize decision boundaries [57, 58]. This approach computes for each pattern p the sensitivity $S_{oz,ki}^{(p)}$ of each output unit o_k to small perturbations in each input unit z_i, using equation (4). Patterns with high $S_{oz,ki}^{(p)}$ values lie closest to decision boundaries. The sensitivity analysis selective learning algorithm makes use of this fact to assign to each pattern a "measure of closeness" to decision boundaries. Patterns closest to decision boundaries are the most informative as calculated from equation (1). Selecting the most informative patterns therefore results in training only on patterns close to boundaries.

To illustrate the selection of patterns around decision boundaries, consider the artificial circle-in-a-box problem

$$\text{class} = \begin{cases} 0 & \text{if } \sqrt{z_1^2 + z_2^2} \leq 0.5 \\ 1 & \text{otherwise} \end{cases} \tag{8}$$

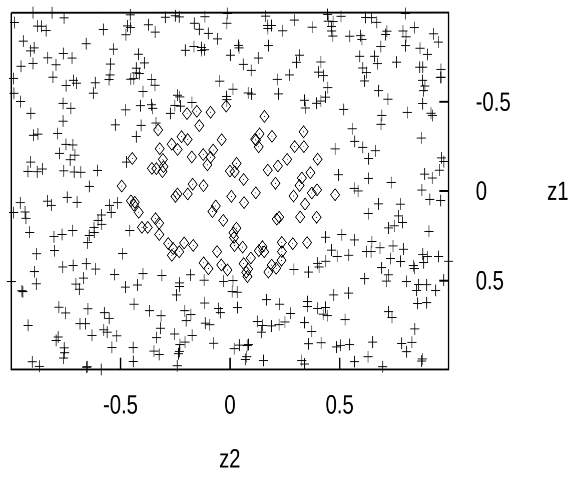
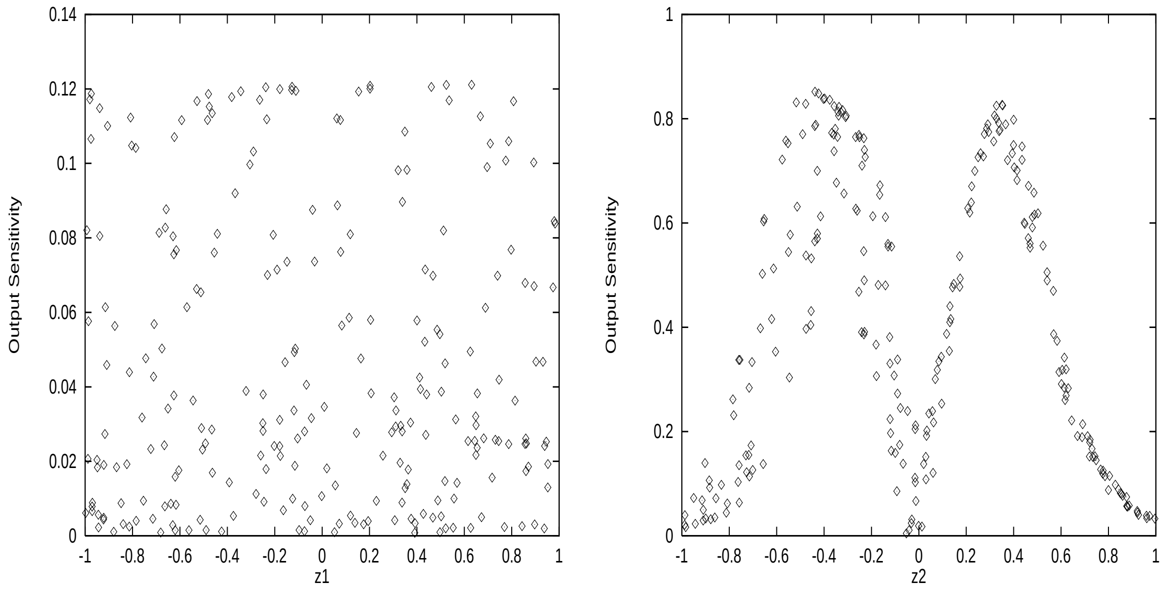


Figure 1. Circle classification problem defined in (8)

where z_1 and z_2 are the input parameters, sampled from an uniform distribution, i.e. $U \sim (-1, 1)$. Figure 1 illustrates this problem.

Using a 2-3-1 NN architecture, trained on 250 patterns (using normal fixed set learning), figures 2 and 3 visualize the boundaries for this problem. These figures plot the values of $S_{oz,ki}^{(p)}$ for each pattern p. Figure 2 illustrates the boundaries for inputs z_1 and z_2 after 50 epochs. At this point no boundaries have formed for z_1, while two clear boundaries formed for z_2. Note the peaks of high sensitivity values for input z_2 at approximately $z_2 = -0.4$ and $z_2 = 0.4$. Figure 3 illustrates two boundaries for z_1 after 500 epochs at approximately $z_1 = -0.45$ and $z_2 = 0.48$ (similar results are obtained for z_2). Figure 3 shows that only a small percentage of the training patterns in the candidate training set lie close to the boundaries, suggesting that we may benefit from discarding patterns far away from boundaries. Figures 4(a) and 4(b) illustrate the patterns used for training as selected by SASLA for epoch 200 and epoch 500 respectively. Comparison of these figures with figure 1, which represents the candidate training set, reveals a substantial reduction in the number of training patterns. These figures also show the distribution of the selected patterns around the -0.5 and 0.5 boundaries. SASLA retains those patterns important to form the decision boundaries. From figure 4 we see that those patterns which are most distant from the boundary are discarded from epoch 200 to epoch 500.

Other learning algorithms have been developed that use different approaches to select patterns near decision boundaries. Ohnishi, Okamoto and Sugie use known characteristics of the problem to select patterns around the decision boundaries once before training [42]. Selective Sampling, developed by



(a) Input z_1 (b) Input z_2

Figure 2. Boundaries for circle classification problem after 50 epochs

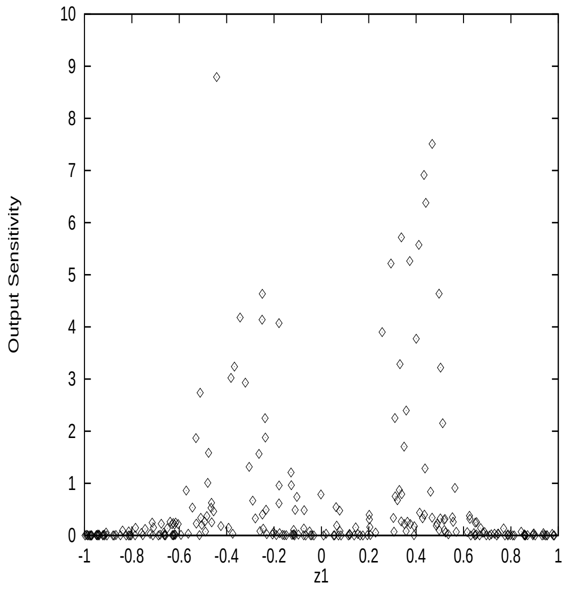
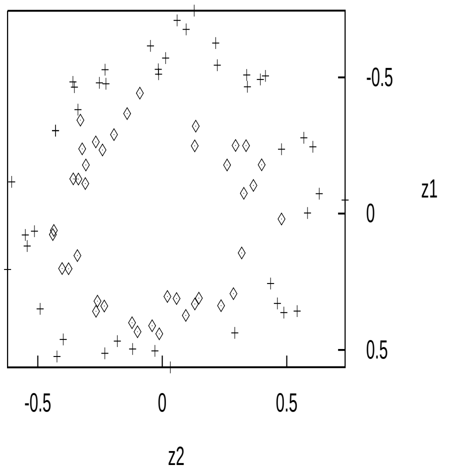
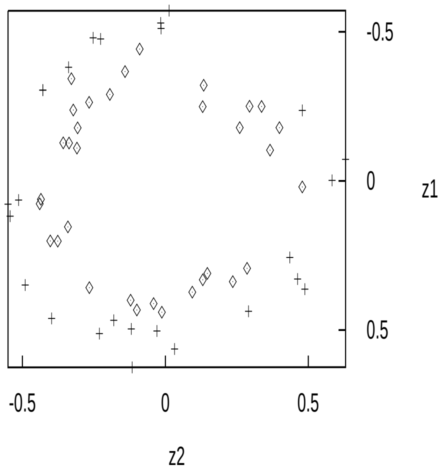


Figure 3. Boundaries for circle classification problem after 500 epochs



(a) At epoch 200



(b) At epoch 500

Figure 4. Selected training patterns at different epochs for the circle classification problem

Cohn, Atlas and Ladner, uses distribution information from the environment to calculate a region of uncertainty around boundaries, and select patterns from these regions only [36]. However, in most applications prior knowledge is not available. Baum finds boundaries using an iterative search and selects patterns in the region of these boundaries [54]. In Query-Based Learning, a NN inversion algorithm is used together with selective sampling to generate new patterns around the decision boundaries to further refine these boundaries [55].

The sensitivity analysis approach to locate patterns close to decision boundaries offers a less complex approach, making use of derivative information already calculated during learning.

3.3. Mathematical Model

This section defines the SASLA pattern selection operator, $\mathcal{A}$, and shows how the operator is used to generate a new training subset. Define the SASLA operator as

$$\mathcal{A}(D_C, \mathcal{F}(D_T; W)) = \{p \in D_C | \Phi_{\infty}^{(p)} > \Psi(\vec{\Phi}_{\infty})\} \tag{9}$$

where D_C is the set of candidate training patterns, W represents the network weights, $\mathcal{F}(D_T; W) : \mathbb{R}^I \rightarrow \mathbb{R}^K$ is the current function approximated by the NN that relates the I-dimensional input space to the K-dimensional target space, using only the training patterns in the training set D_T. $\Phi_\infty^{(p)}$ is defined in equation (2), and $\vec{\Phi}_\infty$ is the vector

$$\vec{\Phi}_\infty = (\Phi_\infty^{(1)}, \ldots, \Phi_\infty^{(p)}, \ldots, \Phi_\infty^{(P_C)}) \tag{10}$$

The function Ψ implements the rule used to select patterns. For the SASLA implementation presented in this paper,

$$\Psi(\vec{\Phi}_\infty) = (1-\beta)\overline{\Phi}_\infty \tag{11}$$

where β is the subset selection constant (discussed below), and $\overline{\Phi}_\infty$ is the average pattern informativeness,

$$\overline{\Phi}_\infty = \frac{\sum_{p=1}^{P_C} \Phi_\infty^{(p)}}{P_C} \tag{12}$$

where P_C is the total number of patterns in the candidate training set D_C. Patterns with informativeness a factor larger than the average informativeness over all patterns are therefore selected.

The subset selection constant β is crucial to the efficacy of the algorithm. This selection constant, which lies in the range $[0, 1]$, is used to control the region around decision boundaries within which patterns will be considered as informative. The larger the value of β, the more patterns will be selected. If β is too small, only a few patterns will be selected which may not include enough information to form boundaries. A conservative choice of β close to 1 improves the chances of selecting patterns representing enough information about the target concept, ensuring most of the candidate patterns to be included in the initial training subset. A conservative value for β does, however, not mean a small reduction in training set size. As training progresses, more and more patterns become uninformative, resulting in larger reductions in training set size. Section 4 shows that for such a conservative β, the training set is substantially reduced early in training. If $\beta = 1$, SASLA simply generalizes to normal fixed set learning (FSL). Section 4 empirically investigates the influence of the β parameter.

At each subset selection interval τ_s (corresponding to epoch ξ_s), let the new subset be $D_{S_s} = \mathcal{A}(D_C, \mathcal{F}_{NN}(D_T; W))$. That is, select from the original candidate set D_C the subset $D_{S_s} \subseteq D_C$. Then, let the training set $D_T = D_{S_s}$ for the next training interval. Note that, after subset selection, the NN does not train on the difference $D_C - D_{S_s}$. The actual training set for the current training interval is reduced by $|D_C - D_{S_s}|$ patterns.

3.4. Selective Learning Algorithm

The sensitivity analysis selective learning algorithm is outlined below. Note that the algorithm makes no reference to a specific optimization method or objective function. Although SASLA depends on the choice of activation functions (requiring differentiable activation functions) and NN architecture, the formulation is general to illustrate the algorithm's applicability to any activation function and architecture. It is only step 2.b.i that changes for different activation functions and architectures.

```
1. Initialize weights and learning parameters.
   Initialize the pattern selection constant, $\beta = 0.9$ for a conservative choice.
   Construct the initial training subset, $D_{S_0} = D_C$.
   Let the training set $D_T = D_{S_0}$.
2. Repeat
   (a) Repeat
       Train the NN on training subset $D_T$
       until a termination criterion on $D_T$ is triggered.
   (b) Compute the new training subset $D_{S_s}$ for the next subset selection interval $\tau_s$:
       i. For each $p \in D_C$, compute the sensitivity matrix $S_{oz,ki}^{(p)}$ using equation (4) for sigmoid
          activation functions.
       ii. Compute the output sensitivity vector $\vec{S}_o^{(p)}$ for each $p \in D_C$ from equation (3).
       iii. Compute the informativeness $\Phi^{(p)}$ of each pattern $p \in D_C$ using equation (2).
       iv. Compute the average pattern informativeness from equation (12).
       v. Apply operator $\mathcal{A}$ in equation (9) to find the subset $D_{S_s}$ of most informative patterns.
          Then, let $D_T = D_{S_s}$.
   until convergence is reached.
```

Design issues specific to SASLA are discussed next:

- **Initial subset selection and size**: The purpose of the initial subset is to initiate training. As soon as training starts, the NN starts to build its knowledge of the problem. Only then can pattern selection be applied effectively to refine the NN's knowledge. SASLA starts training on the entire candidate training set. The initial training subset is therefore just D_C, consisting of $P_C = |D_C|$ patterns.

- **Subset termination criteria**: Learning each training subset to the same accuracy as required for the complete set of training data may cause overfitting of the subsets. The network may converge to an unacceptable local optimum solution, instead of generalizing to a global optimum solution. The efficiency of active learning algorithms relies on subset termination criteria to signal the selection of a new subset. The term *subset selection interval* refers to the "time" (epoch) at which a termination criterion is triggered and a next subset is selected. The current implementation of SASLA selects a new training subset at each epoch, allowing for many opportunities to make optimal use of the NN's knowledge in the search for the most informative patterns. Section 4 shows that, even for subset selection at each epoch, the total number of calculations is reduced compared to that of FSL. To further reduce computations, subsets can be selected at longer time intervals, provided that mechanisms are implemented to prevent overfitting.

- **Subset size**: Subset sizes do have an influence on the efficiency of the active learning algorithm. Smaller subsets give more opportunity to select informative patterns, but increase the number of selection intervals. For computationally expensive selection criteria, for example methods based on information theory, too many selection intervals may adversely affect convergence times. For

SASLA, the size of each subset may differ from one selection interval to the next, since subset sizes depend on the subset selection constant, β, and pattern informativeness as obtained from the knowledge embedded in the current NN weights.

The usual termination criteria are used, i.e. training stops when the maximum number of epochs is exceeded, or when the mean squared error (MSE) is lower than the given threshold, or when the percentage correctly classified patterns is higher than the given threshold.

3.5. Model Complexity

An important requirement for any new learning algorithm is that its complexity should not be unacceptably higher than existing algorithms. If possible, the complexity should be reduced. The complexity of the proposed selective learning algorithm is explored in this section and compared to that of FSL. For the purposes of this exposition, complexity is expressed as the number of calculations and comparisons made during one training sweep through the training set (one epoch). Calculations include additions, subtractions, multiplications and divisions.

Since the selective learning operator $\mathcal{A}$ is applied to each pattern in the candidate set D_C, a computational cost is assigned to each pattern presentation. Let $\mathcal{C}_{FSL}^{(p)}$ and $\mathcal{C}_{SASLA}^{(p)}$ respectively denote the cost per pattern presentation for FSL and SASLA. Then,

$$\mathcal{C}_{FSL}^{(p)} = \mathcal{C}_W^{(p)} + \mathcal{C}_V^{(p)} \tag{13}$$

and

$$\mathcal{C}_{SASLA}^{(p)} = \begin{cases} \mathcal{C}_{\mathcal{A}}^{(p)} + \mathcal{C}_W^{(p)} + \mathcal{C}_V^{(p)} & \text{if pattern } p \text{ is selected} \\ \mathcal{C}_{\mathcal{A}}^{(p)} & \text{if pattern } p \text{ is not selected} \end{cases} \tag{14}$$

where $\mathcal{C}_W^{(p)}$ is the cost of updating weights between the hidden and output layer for one pattern, and $\mathcal{C}_V^{(p)}$ is the cost of updating weights between the input and hidden layer. The term $\mathcal{C}_{\mathcal{A}}^{(p)}$ represents the cost to calculate the informativeness of pattern p and to make the selection decision. Equation (13) illustrates a fixed computational cost for each pattern presentation for FSL, while the computational cost per pattern presentation for SASLA depends on whether the pattern is selected for inclusion in the training subset. The total number of pattern presentations for FSL after ξ epochs is

$$\mathcal{T}_{FSL}(D_C) = \sum_{\varepsilon=1}^{\xi} P_C = \xi P_C \tag{15}$$

For SASLA, the total number of pattern presentations is

$$\begin{aligned} \mathcal{T}_{SASLA}(D_C) &= \mathcal{T}_{SASLA}(D_{S_0}) + \mathcal{T}_{SASLA}(D_{S_1}) + \cdots + \mathcal{T}_{SASLA}(D_{S_S}) \\ &= \sum_{s=1}^{S} \sum_{\varepsilon=\xi_{s-1}}^{\xi_s} P_{S_s} = \sum_{s=1}^{S} (\xi_s - \xi_{s-1}) P_{S_s} \end{aligned} \tag{16}$$

From equations (13) to (16), the total cost for all SASLA pattern presentations is

$$\mathcal{C}_{SASLA} = \mathcal{C}_{FSL} + \mathcal{C}_{\mathcal{A}} - (\mathcal{T}_{FSL}(D_C) - \mathcal{T}_{SASLA}(D_C))(\mathcal{C}_W^{(p)} + \mathcal{C}_V^{(p)}) \tag{17}$$

where the term $\mathcal{T}_{FSL}(D_C) - \mathcal{T}_{SASLA}(D_C)$ is the difference in the total number of pattern presentations between FSL and SASLA after ξ epochs, and $\mathcal{C}_{SASLA}$ and $\mathcal{C}_{FSL}$ respectively represent the total training cost of SASLA and FSL; $\mathcal{C}_{\mathcal{A}}$ is the total cost of the selective learning operator. From equation (17), a cost saving by SASLA is achieved when $\mathcal{C}_{SASLA} - \mathcal{C}_{FSL} < 0$. If $\mathcal{C}$ denotes the cost saving, then

$$\mathcal{C} = \mathcal{C}_{\mathcal{A}} - (\mathcal{T}_{FSL}(D_C) - \mathcal{T}_{SASLA}(D_C))(\mathcal{C}_W^{(p)} + \mathcal{C}_V^{(p)}) < 0 \tag{18}$$

If equation (18) becomes true, SASLA is more cost effective than FSL.

For one pattern presentation, $\mathcal{C}_W^{(p)} = 2K(J+1)(7+I+J)$ calculations, and $\mathcal{C}_V^{(p)} = (2+5K)(I+1)(J+1)$ calculations, where I, J and K are respectively the number of input, hidden and output units (considering weight updates using gradient descent).

Referring to the sensitivity analysis selective learning algorithm in section 3.4, the total cost of applying the operator is

$$\mathcal{C}_{\mathcal{A}} = S \times (\mathcal{C}_{S_{oz}} + \mathcal{C}_{\vec{S_o}} + \mathcal{C}_{\vec{\Phi}} + \mathcal{C}_{\overline{\Phi}} + \mathcal{C}_{\Psi}) \tag{19}$$

where S is the total number of subset selection intervals $\tau_1, \cdots, \tau_s, \cdots, \tau_S$. In equation (19), $\mathcal{C}_{S_{oz}}$ is the total cost of calculating the sensitivity matrix for all candidate patterns. That is, $\mathcal{C}_{S_{oz}} = P_C \mathcal{C}_{S_{oz}^{(p)}}$. Referring to equation (4), the cost $\mathcal{C}_{S_{oz}^{(p)}}$ of calculating the sensitivity matrix for a single pattern is simply $3IJK$ calculations, since the derivatives $f_{o_k}^{'}$ and $f_{y_j}^{'}$ are already calculated during training. Therefore,

$$\mathcal{C}_{S_{oz}} = P_C \times C_{S_{oz}^{(p)}} = P_C \times (3IJK) \tag{20}$$

The total number of calculations per selection interval to compute the output sensitivity vector for all candidate patterns is (from equation (3))

$$\mathcal{C}_{\vec{S_o}} = P_C \times (IK) \tag{21}$$

To compute the informativeness of all candidate patterns, the total number of comparisons is (from equation(2))

$$\mathcal{C}_{\vec{\Phi}} = P_C \times (K) \tag{22}$$

To compute the average pattern informativeness, the cost is (from equation (12))

$$\mathcal{C}_{\overline{\Phi}} = P_C + 1 \tag{23}$$

Finally, the cost of applying the selection operator is (from equations (9) and (11))

$$\mathcal{C}_{\Psi} = P_C + 2 \tag{24}$$

Substitution of equations (20) to (24) into equation (19) yields

$$\mathcal{C}_{\mathcal{A}} = S(P_C(3IJK + IK + K + 2) + 3) \tag{25}$$

Which gives a cost saving of

$$\begin{aligned} \mathcal{C} &= S(P_C(3IJK + IK + K + 2) + 3) \\ &- (\xi P_C - \mathcal{T}_{SASLA}(D_C))(2K(J+1)(7+I+J) + (2+5K)(I+1)(J+1)) \end{aligned} \tag{26}$$

An analysis of equation (26) is presented next to establish under which conditions the cost saving is negative or positive.

- In the worst case, all candidate patterns lie close to decision boundaries such that SASLA prunes no patterns during training. In this case $\mathcal{C} = \mathcal{C}_{\mathcal{A}}$. Hence, SASLA is computationally more complex than FSL when all patterns lie close to boundaries.
- Since $3IJK + IK + K + 2$ is much less than $2K(J+1)(7+I+J) + (2+5K)(I+1)(J+1)$, with a difference of $21JK + 4IJK + 2KJ^2 + 18K + 16IK + 2IJ + 2I + 2J$, SASLA will be computationally less expensive than FSL when the difference $\xi P_C - \mathcal{T}_{SASLA}(D_C)$ is large. As illustrated in section 4, this is the case for all the experiments investigated.
- The complexity of SASLA can further be reduced by using fewer subset selection intervals, instead of selecting a new training subset at each epoch.
- Even for small candidate training set sizes, SASLA can be less complex than FSL since the cost for weight updates is much larger than the cost of applying the selective learning operator.

4. Results

This section first evaluates the performance of SASLA against that of normal fixed set learning (FSL). Eight real-world problems of varying complexity are used to test the performance of the proposed sensitivity analysis selective learning algorithm, as summarized in table 1. These problems differ in candidate training set size, input and output dimensions and the characteristics of data values (i.e. continuous, discrete, binary, whether missing values occur). Missing values are replaced by the average value for that attribute during a data pre-processing phase. Table 2 lists for each problem the learning parameters used, including the learning rate, momentum, training and test set sizes, and the number of epochs the networks were trained. Secondly, the effect of varying values of the selection constant β is investigated using two of the classification problems. Finally, SASLA is also compared with error selection.

Each problem investigated is referred to as an *experiment*. One NN training and testing session of an experiment is referred to as a *simulation*. For each experiment, 50 simulations are executed for each of the learning models whose performance is studied, that is 50 FSL and 50 SASLA simulations. This choice of the number of simulations allows the normality assumption and faithful comparison of the means [59]. For each learning model, the performance under oversized NN architectures is investigated, in order to investigate the overfitting characteristics of that learning model. Refer to a *simulation pair* as one simulation from each of the learning models with the same simulation number. NNs corresponding to the simulations of a simulation pair have the same architecture, learning parameters (learning rate and momentum), initial random weights, and training and test sets.

For each experiment, the original data set is scaled such that all input values are in the range $[-1, 1]$, and all outputs are in the range $[0.1, 0.9]$. Fifty training and test set pairs are then randomly generated from the original data set. Each simulation uses a different training and test set pair. Training continues for a fixed number of epochs (refer to table 2), and is not stopped when a specified error limit is reached. This is to allow the study of the overfitting effects of the different learning models.

SASLA, FSL and ES use on-line learning where weights are updated after each pattern is presented. Patterns are selected randomly from the training set during training. The learning rate and momentum for each experiment are listed in table 2. For the first set of experiments, SASLA uses a conservative subset selection constant of $\beta = 0.9$, and a new training subset is selected after each epoch. Since this section

Table 1. Characteristics of data sets used to test SASLA

Problem	**Attribute Types**	**NN Architecture**	**Missing Values**	**Class Distribution**	**Source**
circle	continuous	2-3-1	none	class1 - 75% class2 - 25%	artificial
breast cancer	discrete	10-15-1	yes	benign - 65.5% malignant - 34.5%	[60]
diabetes	continuous	8-40-1	none	class1 - 65.1% class2 - 34.9%	[60]
iris	continuous	4-10-3	none	setosa - 33.3% versicolor - 33.3% virginica - 33.3%	[61]
wine	continuous	13-10-3	none	class1 - 33.1% class2 - 39.9% class3 - 27.0%	[61]
glass	continuous	9-30-6	none	class1 - 32.7% class2 - 35.5% class3 - 7.9% class4 - 6.1% class5 - 4.2% class6 - 13.6%	[60]
hepatitis	binary continuous	19-20-1	in most attributes	class1 - 79.87% class2 - 20.13%	[61]
thyroid	binary continuous	21-20-3	none	class1 - 2.31% class2 - 5.11% class3 - 92.58%	[60]

deals with classification problems only, the number of correctly classified patterns is used as measure of training accuracy and generalization. For the purposes of this exposition assume that a pattern is correctly classified if for each output k, $((o_k \geq 0.7 \text{ and } t_k = 0.9) \text{ or } (o_k \leq 0.3 \text{ and } t_k = 0.1))$.

4.1. SASLA vs Fixed Set Learning (FSL)

This section compares the performances of SASLA (with selection constant $\beta = 0.9$) to that of FSL. The generalization performance, overfitting effects, complexity and convergence of the two learning algorithms are compared.

Tables 3 and 4 present extensive summaries of the results obtained for all problems considered. Table 3 lists for each problem the following statistics as obtained at the final training epoch (as given in table 2):

Table 2. Learning parameters for SASLA, FSL and ES experiments

Problem	**Train Set/ Test Set**	**Number of Epochs**	**Learning Rate**	**Momentum**
circle	250/150	500	0.1	0.9
breast cancer	480/120	1000	0.1	0.9
diabetes	560/140	5000	0.1	0.9
iris	120/30	1000	0.1	0.9
wine	142/36	500	0.1	0.9
glass	172/42	2000	0.1	0.6
hepatitis	123/31	200	0.1	0.9
thyroid	4000/3000	300	0.05	0.5

- The average training error, $\overline{\mathcal{E}}_T$, and the average generalization, $\overline{\mathcal{E}}_G$, over the 50 simulations, as the percentage correctly classified patterns. Also included are the 95% confidence intervals computed from the *t*-distribution.

- The best average generalization, $\overline{\mathcal{E}}_G^{best}$, and the number of pattern presentations needed to reach this best generalization.

- The average generalization factor, $\overline{\rho}$, over the 50 simulations, together with 95% confidence intervals. The generalization factor is computed per simulation as $\rho = \mathcal{E}_G/\mathcal{E}_T$, where $\mathcal{E}_G$ is the error on the test set (generalization), and $\mathcal{E}_T$ is the error on the training set. Since the error is expressed as the percentage correctly classified patterns, the ideal performance is when $\mathcal{E}_G \geq \mathcal{E}_T$; thus, when $\rho = \mathcal{E}_G/\mathcal{E}_T \geq 1$. Usually, $\mathcal{E}_G < \mathcal{E}_T$, but when $\mathcal{E}_G$ is much less than $\mathcal{E}_T$ it indicates severe overfitting of the training set. The objective is therefore to obtain a generalization $\mathcal{E}_G$ as close to $\mathcal{E}_T$ as possible and therefore ρ close to 1. A higher generalization factor ρ thus indicates less overfitting of the training set.

In addition to the averages and confidence intervals, table 3 also presents estimates of the difference in performance between SASLA and FSL for the training error, generalization and generalization factor. For example, considering the training errors $\mathcal{E}_{T_i}^{SASLA}$ and $\mathcal{E}_{T_i}^{FSL}$, for each simulation i, the estimate

$$\overline{\sigma} = \frac{1}{50}\sum_{i=1}^{50}(\mathcal{E}_{T_i}^{SASLA} - \mathcal{E}_{T_i}^{FSL}) \tag{27}$$

is computed to obtain a 95% confidence interval, using the *t*-distribution. These estimates can be used to determine if there is a significant difference in performance between SASLA and FSL. The performance difference estimates are the first entries for each problem (the values between parentheses).

Table 4 summarizes the following information for each learning algorithm and problem at the final epoch:

- The average number of patterns, $\overline{P}_T$, selected from the candidate set and used for training, together with 95% confidence intervals for SASLA.
- The average number of pattern presentations, together with 95% confidence intervals for SASLA.
- The average saving in computational cost achieved by using SASLA, with 95% confidence intervals.

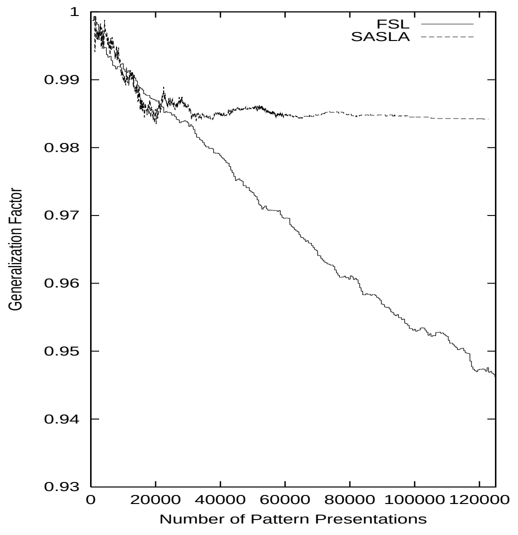
With reference to generalization performance, it is not possible to label any of the training methods as being superior, since they produced approximately the same average generalization and best generalization performances. Considering the training error, FSL achieved higher accuracies than SASLA for all problems. Note that even though FSL achieved higher accuracies on the training set, SASLA had better generalization performances for the *breast cancer, diabetes* and *hepatitis* problems, while having approximately the same generalization performance for the *iris, wine* and *thyroid* problems. It is only for the *glass* problem that FSL obtained better generalization than SASLA.

The consequence of FSL having better training accuracies than SASLA, but approximately the same generalization performance, is that FSL tends to overfit the training data. Table 3 shows that FSL overfitted more at the final epoch than SASLA for the *breast cancer, diabetes, iris* and *hepatitis* problems. Figure 5 illustrates the overfitting effects of the two algorithms for these problems, where the generalization factor is plotted as a function of the number of pattern presentations. Figure 5 shows that FSL increasingly overfits as training progresses, while the overfitting characteristics of SASLA stabilize. The two algorithms had the same generalization factors for the other problems.

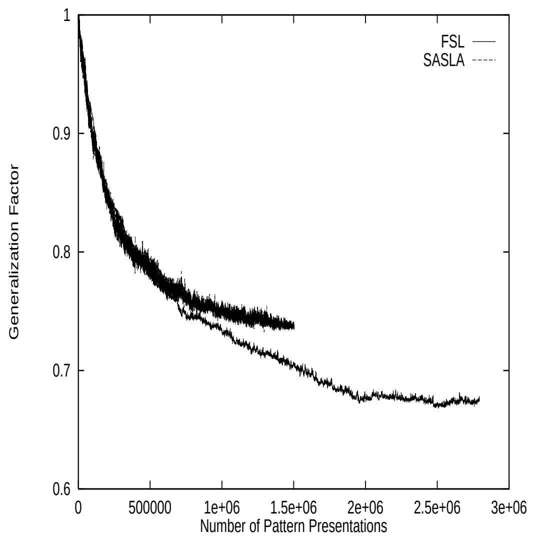
The larger overfitting effects for FSL can be explained by the fact that FSL trains on the entire training set, which, for real-world problems, usually contains outlier and noisy patterns. Dynamic pattern selection around decision boundaries reduces the effects of outliers and patterns with a very large noise component, since these patterns lie the furthest away from decision boundaries compared to "normal" patterns, and will have a smaller chance to be selected. Since overfitting occurs when the network starts to memorize individual case for these experiments), this explains why SASLA exhibits less overfitting than FSL.

While SASLA and FSL showed approximately the same generalization performance, SASLA needed much less pattern presentations than FSL to reach the same generalization performance levels. This conclusion is supported by the fourth and fifth columns of table 3. These columns show that SASLA needed substantially less pattern presentations to reach its best generalization performance than FSL (except for the *iris* problem).

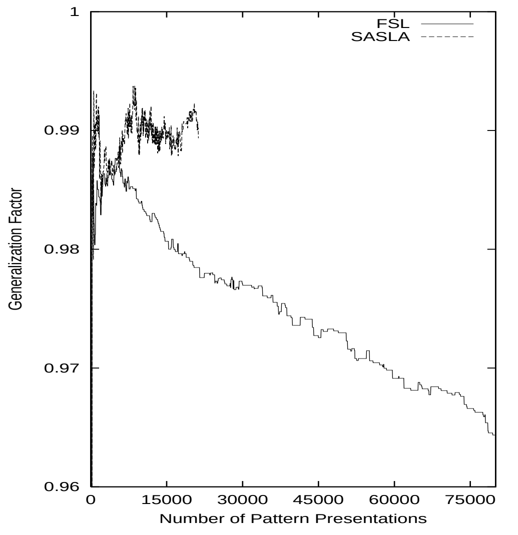
Tables 3 and 4 show that SASLA required less pattern presentations to reach good generalization levels compared to FSL. But does this mean that SASLA is computationally less complex than FSL, taking in consideration the added complexity of applying the pattern selection operator? Using the number of calculations per epoch as measure of computational complexity, equation (26) expresses the saving in the number of calculations by using SASLA. The last column in table 4 lists the saving in computational cost (at the final epoch) by using SASLA. Note that substantial savings in the number of calculations were achieved for all problems, except the *glass* problem. Even for the conservative subset selection constant, $\beta = 0.9$, SASLA was computationally more feasible than FSL very early in training, as indicated in table 5 which lists the last epoch at which FSL was, on average, computationally less expensive than SASLA. Although SASLA showed to be computationally more complex than FSL for the *glass* problem, SASLA still used less training presentations than FSL for 2000 training epochs.



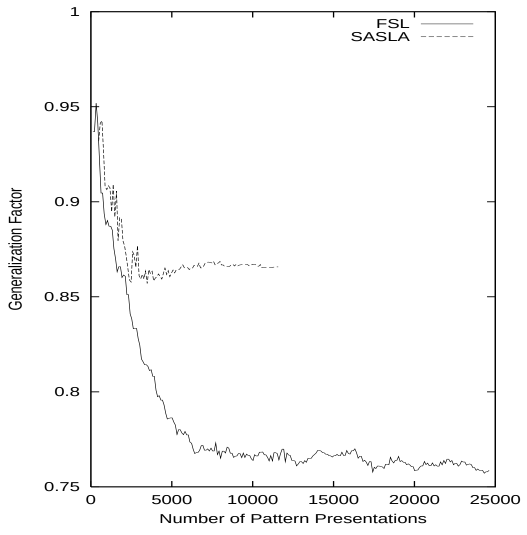
(a) Breast Cancer



(b) Diabetes



(c) Iris



(d) Hepatitis

Figure 5. Average generalization factor vs pattern presentations for real-world problems

(The next section shows that SASLA saved on computational costs for lower β values but at decreased accuracy.) The larger cost for the *glass* problem is attributed to the slow and small decrease in the training set size, such that the computational cost of applying the selection operator is larger than the saving due to the decreased training set size.

For the other problems, these very large computational cost savings were made possible by the large reduction in training set size through application of the selective learning operator. For these problems, the training set size reduced exponentially. The number of training patterns was rapidly reduced early in training, after which the training set size asymptotically converged to the set of patterns that defines the decision boundaries. The size of the thyroid candidate training set was, however, not that rapidly reduced. Table 6 shows for selected epochs the percentage reduction of the original training set.

Finally, figure 6 compares the convergence performance of SASLA with that of FSL. This figure plots the percentage of the 50 simulations that did not converge to specific generalization levels. SASLA performed exceptionally well for the *breast cancer* and *diabetes* problems, having much more converged simulations than FSL, especially for the higher generalization levels. For the *iris* problem SASLA performed in general better than FSL, having more converged simulations for most of the generalization levels. It is not *thyroid* problems. However, notice that SASLA tended to have more converged simulations for the higher generalization levels than FSL. Also, where FSL had more converged simulations, it was by a small percentage. FSL showed better convergence results for the *wine* and *glass* problems.

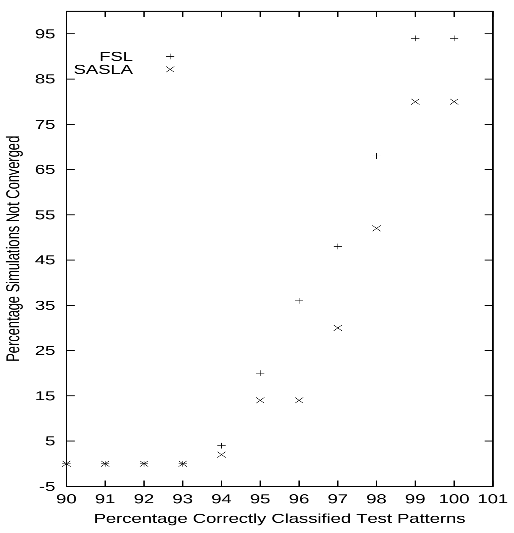
4.2. Selection Constant's Effects

While the results presented in the previous section are for $\beta = 0.9$, this section illustrates the effect of varying β values on performance. For this purpose the *glass* and *wine* problems are used, and 50 simulations executed for $\beta = 0.1, 0.3, 0.5$ and 0.7. The results, which include averages over the 50 simulations and associated 95% confidence intervals, are summarized in tables 7 and 8 for the *glass* problem (at epoch 2000), and tables 9 and 10 for the *wine* problem (at epoch 500).

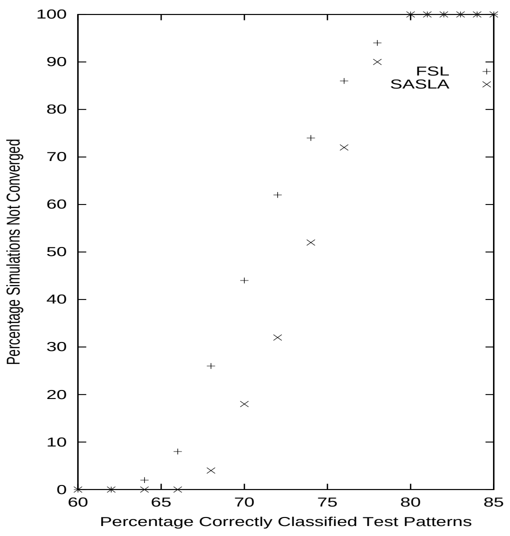
Tables 7 and 9 show that the lower the value of β, the worse the accuracy of SASLA: The average training error and generalization over the 50 simulations deteriorated for both problems, while the best generalization also decreased for the *glass* problem. Also note that higher β values achieved generalization accuracies larger than the training accuracy. As the value of β becomes smaller, the difference between generalization and training accuracy becomes smaller. Lower β values do, however, save on computations, due to the fact that less patterns are selected for the candidate set. However, the savings in computational cost using low β values do not justify the loss in accuracy for the *glass* problem, while the *iris* problem's decrease in accuracy is acceptable. The larger savings in computational cost for smaller β values are due to the larger decreases in training set sizes compared to larger β values. Tables 11 and 12 illustrate for the *glass* and *wine* problems how lower β values decreased the training set size for selected epochs.

Tables 13 and 14 show that the smaller the value of β the more simulations do not converge to generalization levels specified in the table, indicating that the lower β values do not select enough information to achieve the higher generalization accuracies.

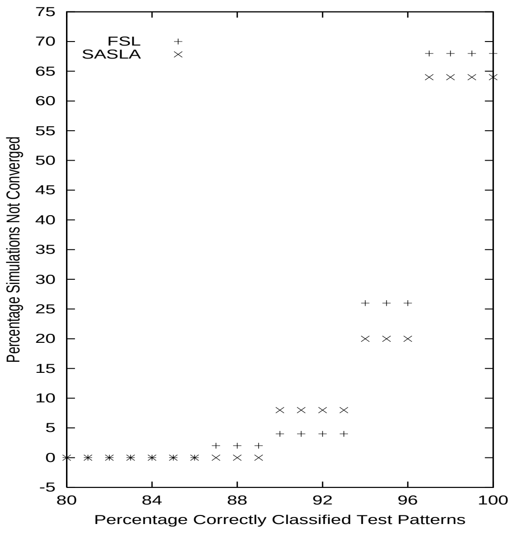
What can be concluded from the above experiments is that the best value for β is problem dependent. In selecting a value for β the trade-off between computational cost savings and decreased performance needs to be considered. It is suggested that future research includes an investigation into adaptive β values, where the value of β increases when performance deteriorates, and decreases when performance



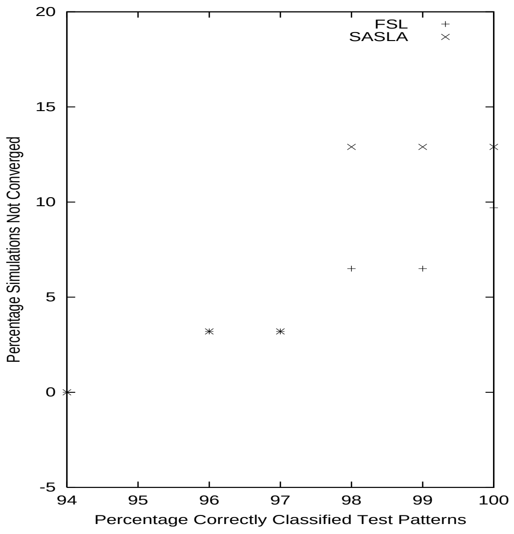
(a) Breast Cancer



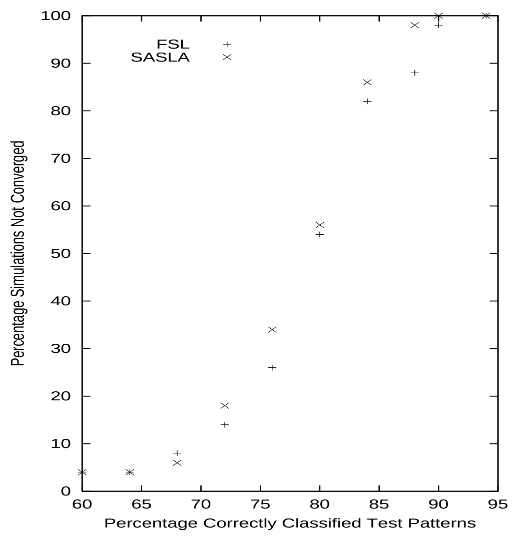
(b) Diabetes



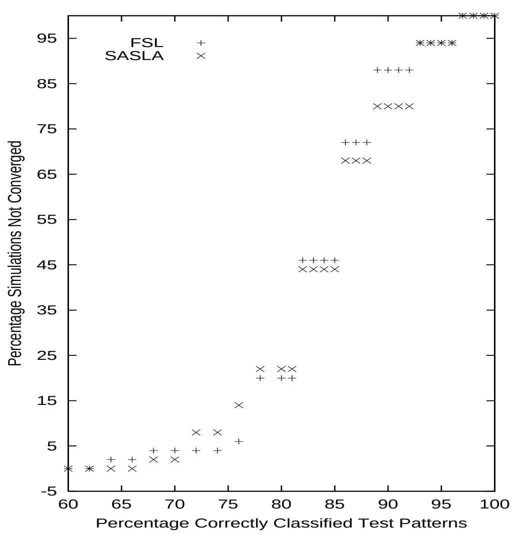
(c) Iris



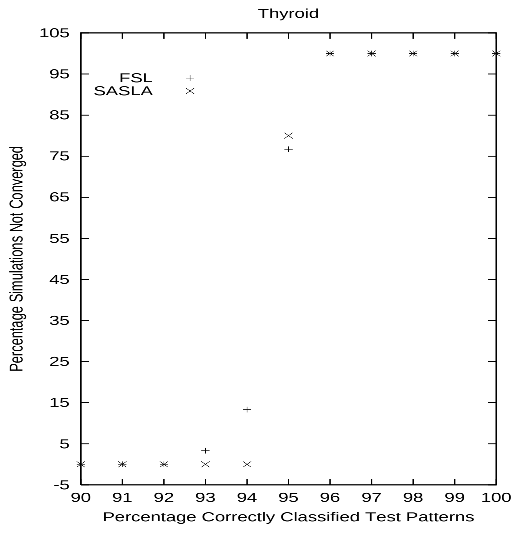
(d) Wine



(e) Glass



(f) Hepatitis



(g) Thyroid

Figure 6. Percentage simulations that did not converge to generalization levels for real-world problems

improves.

4.3. SASLA vs Error Selection (ES)

The reader may now ask why don't we simply select those patterns from the candidate set that are not yet correctly classified. While this approach will be computationally less expensive than SASLA, due to the simple selection operator, the error selection approach steps into trouble when the candidate set contains outlier patterns (assuming a non-robust estimator such as sum squared error is used [62, 63]). Since outlier patterns will be classified incorrectly, an error selection approach will select these outlier patterns for training. For sensitivity analysis patterns selection, on the other hand, the chance that outliers will be included in the training set is reduced, since the selection of patterns depends on how far they lie from the decision boundaries - outliers lie further away from decision boundaries than "normal" patterns. SASLA might initially select these outlier patterns, but will discard them as decision boundaries get more refined during training.

The objective of this section is to compare results obtained using an error selection scheme with that of SASLA. For this purpose the *glass* and *wine* problems, as well as the *circle* problem with outliers added to the candidate set (5% of the candidate set), were used to train neural networks based on the following pattern selection schema: if there exists an output unit such that $(t_k^{(p)} = 0.9)\&\&(o_k^{(p)} < 0.7)$ or $(t_k^{(p)} = 0.1)\&\&(o_k^{(p)} > 0.3)$, then select pattern p for training (binary 0 is represented as 0.1, and binary 1 is represented as 0.9). This selection criterion coincides with the assumption as to when a pattern should be accepted as being correctly classified.

Table 15 summarizes the error performance results for the ES approach in comparison with SASLA for $\beta = 0.9$, while table 16 and table 17 respectively summarizes the complexity and convergence results. For all problems SASLA showed to have significantly better generalization performance than ES. Table 15 shows that both SASLA and ES have been influenced by the occurrence of outliers for the *circle* problem. The results show, however, that ES is even more affected by the outliers, having an average generalization of 73.4%, compared to 79% of SASLA. On clean data, SASLA achieved and average generalization of 90.4%, with ES having an 88.4% generalization. From table 16 it is evident that ES is computationally less expensive than SASLA, using substantially less training patterns than SASLA. This very large reduction in training set size by ES compared to SASLA is a cause of the worse ES generalization performance, since the reduced training set may not contain sufficient patterns to refine the boundaries.

SASLA showed to have better convergence characteristics than ES. Table 17 illustrates that ES had substantially more simulations that did not converge to the different generalization levels.

In conclusion, although ES is computationally more efficient than SASLA, the better convergence and generalization results of SASLA make it the preferred selective learning algorithm.

4.4. Conclusions

This paper presented a new selective learning algorithm for feedforward neural networks that uses pattern sensitivity information to dynamically select those training patterns that contain the most information about the position of decision boundaries. The paper considers selective learning as a type of active learning algorithm, where active learning is considered as the ability of a learner to control itself what patterns are used for training.

Although, from the problems investigated in this section, it is not possible to label either this selective learning algorithm, SASLA, or FSL as having a superior generalization performance, SASLA showed to be more robust to overfitting. Furthermore, SASLA requires less pattern presentations to reach specified generalization levels. A major advantage of SASLA is the exponential reduction in training set size, and consequently a very large saving in computational costs. SASLA also shows a more favorable convergence rate than FSL for most of the problems, especially for higher generalization levels.

It was shown that the best value for the selection constant β is in fact problem dependent. Smaller β values do reduce computational effort substantially, but at the cost of degraded generalization performance.

Not much research has been done in selective learning. The available literature does not contain sufficient information to reproduce experiments for comparative purposes. However, a comparison between SASLA and ES has been done, which showed, for the problems investigated, SASLA to have significantly better generalization and convergence results, while ES did save on computations. The *circle* problem, with outliers added, illustrated ES to be affected more by the occurrence of outliers than SASLA.

It is proposed that future research includes an investigation into the effects of dynamic β values, and larger subset selection intervals. Such a study may even further reduce the complexity of SASLA and increase its generalization performance.

References

[1] Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning Internal Representation by Error Propagation, in: *Parallel Distributed Processing* (D.E. Rumelhart, J.L. McClelland, Eds.), Vol. 1, MIT Press, Cambridge, Mass, 1986.

[2] Denoeux, T., Lengellé, R.: Initializing Back Propagation Networks using Prototypes, *Neural Networks*, **6**(3), 1993, 351-363.

[3] Wesseles, L.F.A., Barnard, E.: Avoiding False Local Minima by Proper Initialization of Connections, *IEEE Transactions on Neural Networks*, **3**(6), 1992, 899-905.

[4] Magoulas, G.D., Vrahatis, M.N., Androulakis, G.S.: Effective Backpropagation Training with Variable Stepsize, *Neural Networks*, **10**(1), 1997, 69-82.

[5] Salomon, R., Van Hemmen, J.L.: Accelerating Backpropagation through Dynamic Self-Adaptation, *Neural Networks*, **9**(4), 1996, 589-601.

[6] Weir, M.K.: A Method for Self-Determination of Adaptive Learning Rates in Back Propagation, *Neural Networks*, 1990, 371-379.

[7] Yu, X-H., Chen, G-A.: Efficient Backpropagation Learning using Optimal Learning Rate and Momentum, *Neural Networks*, **10**(3), 1997, 517-527.

[8] Cibas, T., Fogelman Soulié, F., Gallinari, P., Raudys, S.: Variable Selection with Neural Networks, *Neurocomputing*, **12**, 1996, 223-248.

[9] Engelbrecht, A.P., Cloete, I.: A Sensitivity Analysis Algorithm for Pruning Feedforward Neural Networks, *IEEE International Conference in Neural Networks*, Washington, Vol. 2, 1996, 1274-1277.

[10] Fletcher, L., Katkovnik, V., Steffens, F.E., Engelbrecht, A.P.: Optimizing the Number of Hidden Nodes of a Feedforward Artificial Neural Network, *IEEE World Congress on Computational Intelligence, International Joint Conference on Neural Networks*, Anchorage, Alaska, 1998, 1608-1612.

[11] Hassibi, B., Stork, D.G., Wolff, G.: Optimal Brain Surgeon: Extensions and Performance Comparisons, in: *Advances in Neural Information Processing Systems* (J.D. Cowan, G. Tesauro, J. Alspector, Eds), Vol. 6, 1994, 263-270.

[12] Le Cun, Y., Denker, J.S., Solla, S.A.: Optimal Brain Damage, *Advances in Neural Information Processing systems* (D. Touretzky, Ed), Vol. 2, 1990, 598-605.

[13] Zurada, J.M., Malinowski, A., Usui, S.: Perturbation Method for Deleting Redundant Inputs of Perceptron Networks, *Neurocomputing*, **14**, 1997, 177-193.

[14] Hirose, Y., Yamashita, K., Hijiya, S.: Back-Propagation Algorithm which Varies the Number of Hidden Units, *Neural Networks*, **4**, 1991, 61-66.

[15] Kwok, T-Y., Yeung, D-Y.: *Constructive Feedforward Neural Networks for Regression Problems: A Survey*, Technical Report HKUST-CS95-43, Department of Computer Science, The Hong Kong University of Science & Technology, 1995.

[16] Lim, S-F., Ho, S-B.: Dynamic Creation of Hidden Units with Selective Pruning in Backpropagation, *World Congress on Neural Networks*, Vol. 3, 1994, 492-497.

[17] Battiti, R.: First- and Second-Order Methods for Learning: Between Steepest Descent and Newton's Method, *Neural Computation*, **4**, 1992, 141-166

[18] Engelbrecht, A.P., Ismail, A.: Training Product Unit Neural Networks, *Stability and Control: Theory and Applications*, **2**(1-2), 1999, 59-74.

[19] Møller, M.F.: A Scaled Conjugate Gradient Algorithm for Fast Supervised Learning, *Neural Networks*, **6**, 1993, 525-533.

[20] Rosen, B.E., Goodwin, J.M.: Optimizing Neural Networks using Very Fast Simulated Annealing, *Neural, Parallel & Scientific Computations*, 1997.

[21] Tang, Z., Koehler, G.J.: Deterministic Global Optimal FNN Training Algorithm, *Neural Networks*, **7**(2), 1994, 301-311.

[22] Van den Bergh, F., Engelbrecht, A.P.: Cooperative Learning in Neural Networks using Particle Swarm Optimizers, *South Africam Computer Journal*, **26**, 2000, 84-90.

[23] Engelbrecht, A.P., Cloete, I., Geldenhuys, J., Zurada, J.M.: Automatic Scaling using Gamma Learning for Feedforward Neural Networks, *International Workshop on Artificial Neural Networks*, Torremolinos, Spain, in: *"From Natural to Artificial Neural Computing"* in the series Lecture Notes in Computer Science (J. Mira, F. Sandoval, Eds), Vol. 930, 1995, 374-381.

[24] Fletcher, G.P., Hinde, C.J.: Learning the Activation Function for the Neurons in Neural Networks, *International Conference on Artificial Neural Networks*, Vol. 1, 1994, pp 611-614.

[25] Zurada, J.: Lambda Learning Rule for Feedforward Neural Networks, *Proceedings of the IEEE International Conference in Neural Networks*, San Francisco, 1992.

[26] Abu-Mostafa, Y.S.: The Vapnik-Chervonenkis Dimension: Information versus Complexity in Learning, *Neural Computation*, Vol. 1, 1989, 312-317.

[27] Cohn, D., Tesauro, G.: Can Neural Networks do Better than the Vapnik-Chervonenkis Bounds?, *Advances in Neural Information Processing Systems* (R. Lippmann, J. Moody, D.S. Touretzky, Eds), Vol. 3, 1991, 911-917.

[28] Hole, A.: Vapnik-Chervonenkis Generalization Bounds for Real Valued Neural Networks, *Neural Computation*, **8**, 1996, 1277-1299.

[29] Opper, M.: Learning and Generalization in a Two-Layer Neural Network: The Role of the Vapnik-Chervonenkis Dimension, *Physical Review Letters*, **72**(13), 1994, 2133-2166.

[30] Gu, H., Takahashi, H.: Estimating Learning Curves of Concept Learning, *Neural Networks*, **10**(6), 1997, 1089-1102.

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

[32] Zhang, B-T.: Accelerated Learning by Active Example Selection, *International Journal of Neural Systems*, **5**(1), 1994, 67-75.

[33] Sung, K.K., Niyogi, P.: *A Formulation for Active Learning with Applications to Object Detection*, AI Memo No 1438, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 1996.

[34] Lange, R., Männer, R.: Quantifying a Critical Training Set Size for Generalization and Overfitting using Teacher Neural Networks, *International Conference on Artificial Neural Networks*, Vol. 1, 1994, 497-500.

[35] Lange, S., Zeugmann, T.: Incremental Learning from Positive Data, *Journal of Computer and System Sciences*, **53**, 1996, 88-103.

[36] Cohn, D., Atlas, L., Ladner, R.: Improving Generalization with Active Learning, *Machine Learning*, **15**, 1994, 201-221.

[37] Hunt, S.D., Deller J.R.(jr): Selective Training of Feedforward Artificial Neural Networks using Matrix Perturbation Theory, *Neural Networks*, **8**(6), 1995, 931-944.

[38] Cohn, D.A.: *Neural Network Exploration using Optimal Experiment Design*, AI Memo No 1491, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 1994.

[39] Seung, H.S., Opper, M., Sompolinsky, H.: Query by Committee, *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*, 1992, 287-299.

[40] Barnard, E.: Performance and Generalization of the Classification Figure of Merit Criterion Function, *IEEE Transactions on Neural Networks*, **2**(2), 1991, 322-325.

[41] Hampshire, J.B., Waibel, A.H.: A Novel Objective Function for Improved Phoneme Recognition using Time-Delay Neural Networks, *IEEE Transactions on Neural Networks*, **1**(2), 1990, 216-228.

[42] Ohnishi, N., Okamoto, A., Sugiem, N.: Selective Presentation of Learning Samples for Efficient Learning in Multi-Layer Perceptron, *International Joint Conference on Neural Networks* Vol. 1, 1990, 688-691.

[43] Kohara, K.: Selective Presentation Learning for Forecasting by Neural Networks, *International Workshop on Applications of Neural Networks in Telecommunications*, Stockolm, Sweden, 1995, 316-323.

[44] Slade, P., Gedeon, T.D.: Bimodal Distribution Removal, *International Workshop on Artificial Neural Networks,* in: *"New Trends in Neural Computation,"* in the series Lecture Notes in Computer Science (J. Mira, J. Cabestany, A. Prieto, Eds), Springer-Verlag, Berlin, 1993, 249-254.

[45] Gedeon, T.D., Wong, P.M., Harris, D.: Balancing Bias and Variance: Network Topology and Pattern Set Reduction Techniques, *International Workshop on Artificial Neural Networks*, in: *"From Natural to Artificial Neural Computing,"* in the series Lecture Notes in Computer Science (J. Mira, F. Sandoval, Eds), Vol. 930, 1995, 551-558.

[46] Ludik, J.: *Training, Dynamics, and Complexity of Architecture-Specific Recurrent Neural Networks*, PhD Thesis, Department of Computer Science, University of Stellenbosch, February 1995.

[47] Drucker, H.: Boosting using Neural Networks, in: *Combining Artificial Neural Nets, Perspectives in Neural Computing* (A. Sharkey, Ed), Springer, 1999, 51-78.

[48] Quinlan, J.R.: *C4.5: Programs for Machine Learning*, Morgan Kaufmann, 1993.

[49] Stitson, M.O., Weston, J.A.E., Gammerman, A., Vovk, V., Vapnik, V.: *Theory of Support Vector Machines*, Technical Report CSD-TR-96-17, Department of Computer Science, Royal Holloway University of London, 1996.

[50] MacKay, D.J.C.: *Bayesian Methods for Adaptive Models*, PhD Thesis, California Institute of Technology, 1992.

[51] Fukumizu, K.: Active Learning in Multilayer Perceptrons, *Advances in Neural Information Processing Systems* (D.S. Touretzky, M.C. Mozer, M.E. Hasselmo, Eds), Vol. 8, 1996.

[52] Plutowski, M., White, H.: Selecting Concise Training Sets from Clean Data, *IEEE Transactions on Neural Networks*, **4**(2), 1993, 305-318.

[53] Ahmad, S., Tesauro, G.: Scaling and Generalization in Neural Networks: A Case Study, *Proceedings of the 1988 Connectionist Models Summer School*, Morgan Kaufmann, 1989, 3-10.

[54] Baum, E.B.: Neural Net Algorithms that Learn in Polynomial Time from Examples and Queries, *IEEE Transactions on Neural Networks*, **2**(1), 1991, 5-19.

[55] Hwang, J-N., Choi, J.J., Oh, S., Marks, R.J. II: Query-Based Learning Applied to Partially Trained Multilayer Perceptrons, *IEEE Transactions on Neural Networks*, **2**(1), 1991, 131-136.

[56] Engelbrecht, A.P., Cloete, I.: Selective Learning using Sensitivity Analysis, *IEEE World Congress on Computational Intelligence, International Joint Conference on Neural Networks*, Anchorage, Alaska, 1998, 1150-1155.

[57] Engelbrecht, A.P., Cloete, I.: Feature Extraction from Feedforward Neural Networks using Sensitivity Analysis, *International Conference on Systems, Signals, Control, Computers*, Durban, South Africa, Vol. 2, 1998, 221-225.

[58] Engelbrecht, A.P.: Sensitivity Analysis for Decision Boundaries, *Neural Processing Letters*, **10**(3), 1999, 253-266.

[59] Mitchell, T.M.: *Machine Learning*, MacGraw Hill, 1997.

[60] Prechelt, L.: *PROBEN1 - A Set of Neural Network Benchmark Problems and Benchmarking Rules*, Technical Report 21/94, Fakultät für Informatik, Universität Karslruhe, 1994.

[61] University California Irvene Machine Learning Repository,
http://www.ics.uci.edu/m̃learn/MLRepository.html

[62] Hoaglin, D.C., Mosteller, F., Tukey, J.W.: *Understanding Robust and Exploratary Data Analysis*, John Wiley & Sons, 1983.

[63] Huber, P.J.: *Robust Statistics*, John Wiley & Sons, 1981.

Table 3. Comparison of SASLA ($\beta = 0.9$) and FSL error performance measures

Problem	$\overline{\mathcal{E}}_T$	$\overline{\mathcal{E}}_G$	$\overline{\mathcal{E}}_G^{best}$	**Presen-tation**	$\overline{\rho}$
breast cancer	(-0.048 ± 0.004)	(0.016 ± 0.009)			(0.064 ± 0.009)
SASLA	$95.17 \pm 0.42\%$	$93.65 \pm 0.71\%$	96.2%	1530	0.984 ± 0.006
FSL	$99.98 \pm 0.02\%$	$92.03 \pm 0.76\%$	96.2%	2880	0.921 ± 0.007
diabetes	(-0.070 ± 0.014)	(0.002 ± 0.017)			0.058 ± 0.020
SASLA	$82.98 \pm 0.84\%$	$60.87 \pm 1.42\%$	66.0%	20480	0.735 ± 0.019
FSL	$89.97 \pm 8.52\%$	$60.47 \pm 1.31\%$	64.3%	192660	0.676 ± 0.020
iris	(-0.033 ± 0.008)	(-0.009 ± 0.017)			(0.027 ± 0.014)
SASLA	$95.22 \pm 0.56\%$	$93.93 \pm 1.44\%$	94.4%	20470	0.989 ± 0.017
FSL	$98.52 \pm 0.63\%$	$94.8 \pm 0.98\%$	95.2%	7320	0.963 ± 0.010
wine	(-1.283 ± 0.262)	(-0.645 ± 0.521)			(0.005 ± 0.006)
SASLA	$98.33 \pm 0.33\%$	$98.89 \pm 0.64\%$	100.0%	21317	1.006 ± 0.007
FSL	$99.46 \pm 0.27\%$	$99.53 \pm 0.47\%$	100.0%	28189	1.001 ± 0.006
glass	(-2.733 ± 1.353)	(-3.714 ± 1.455)			(-0.0144 ± 0.026)
SASLA	$69.06 \pm 1.07\%$	$70.90 \pm 1.83\%$	88.1%	281485	1.029 ± 0.030
FSL	$71.79 \pm 1.23\%$	$74.62 \pm 1.79\%$	90.4%	337120	1.043 ± 0.033
hepatitis	(-0.103 ± 0.007)	(0.017 ± 0.020)			(0.107 ± 0.022)
SASLA	$89.70 \pm 0.74\%$	$77.56 \pm 2.21\%$	77.9%	7580	0.865 ± 0.026
FSL	$99.98 \pm 0.03\%$	$75.85 \pm 2.08\%$	76.7%	14023	0.759 ± 0.021
thyroid	(-0.018 ± 0.017)	(-0.018 ± 0.017)			(-0.0002 ± 0.0018)
SASLA	$93.39 \pm 1.73\%$	$92.82 \pm 1.75\%$	94.2%	70900	0.994 ± 0.005
FSL	$95.22 \pm 0.29\%$	$94.65 \pm 0.30\%$	94.5%	1104000	0.994 ± 0.002

Table 4. Comparison of SASLA $(\beta = 0.90)$ and FSL computational complexity

Problem	$\overline{P}_T$	**Total presented patterns**	**Cost Saving** $(\times 10^6)$
breast cancer			
SASLA	63.66 ± 8.20	$75\,250.70 \pm 8\,667.86$	$-690.871331 \pm 19.554722$
FSL	480	$480\,000$	
diabetes			
SASLA	304.00 ± 11.20	$1\,501\,858.66 \pm 41\,254.48$	$-6\,488.901950 \pm 292.618026$
FSL	560	$2\,800\,000$	
iris			
SASLA	22.50 ± 1.52	$26\,310.36 \pm 1\,575.59$	-172.210654 ± 3.656954
FSL	120	$120\,000$	
wine			
SASLA	78.45 ± 2.88	$45\,117.77 \pm 1\,068.71$	-32.810974 ± 4.913917
FSL	142	$710\,000$	
glass			
SASLA	131.42 ± 1.67	$287\,906.74 \pm 1\,847.39$	176.860996 ± 49.938683
FSL	172	$344\,000$	
hepatitis			
SASLA	46.64 ± 3.5	$10\,746.30 \pm 692.67$	-38.909426 ± 3.374711
FSL	123	$24\,600$	
thyroid			
SASLA	$2\,706.80 \pm 209.34$	$848\,795.13 \pm 19\,710.10$	$-264.849156 \pm 274.009798$
FSL	$4\,000$	$1\,200\,000$	

Table 5. Last epoch at which FSL is less expensive than SASLA

Problem	**Epoch**
breast cancer	4
diabetes	15
iris	5
wine	8
glass	-
hepatitis	15
thyroid	218

Table 6. Training set reduction by SASLA ($\beta = 0.9$)

	Percentage reduction after epoch			
Problem	**25**	**100**	**200**	**300**
breast cancer	$70.85 \pm 2.73\%$	$81.48 \pm 2.43\%$	$83.85 \pm 2.27\%$	$85.26 \pm 1.91\%$
diabetes	$20.57 \pm 3.47\%$	$30.69 \pm 2.63\%$	$38.05 \pm 1.83\%$	$41.08 \pm 1.99\%$
iris	$64.17 \pm 1.40\%$	$74.32 \pm 1.26\%$	$77.90 \pm 1.23\%$	$79.22 \pm 1.28\%$
wine	$19.21 \pm 1.57\%$	$32.44 \pm 1.62\%$	$37.47 \pm 1.72\%$	$40.87 \pm 1.94\%$
glass	$0.69 \pm 0.23\%$	$4.40 \pm 0.43\%$	$7.49 \pm 0.57\%$	$9.76 \pm 0.67\%$
hepatitis	$52.23 \pm 3.97\%$	$59.46 \pm 3.12\%$	$62.00 \pm 2.84\%$	-
thyroid	$17.45 \pm 3.43\%$	$30.21 \pm 4.84\%$	$35.33 \pm 6.35\%$	$32.33 \pm 5.23\%$

	Percentage reduction after epoch			
Problem	**500**	**1000**	**2000**	**5000**
breast cancer	$86.02 \pm 1.78\%$	$86.74 \pm 1.71\%$	-	-
diabetes	$45.03 \pm 1.86\%$	$47.70 \pm 1.85\%$	$48.07 \pm 2.21\%$	$45.71 \pm 2.0\%$
iris	$80.20 \pm 1.40\%$	$81.25 \pm 1.27\%$	-	-
wine	$45.07 \pm 2.11\%$	-	-	-
glass	$11.76 \pm 0.72\%$	$17.68 \pm 1.04\%$	$23.63 \pm 1.01\%$	-
hepatitis	-	-	-	-
thyroid	-	-	-	-

Table 7. Comparison of error performance measures for different β values for the *glass* problem

β	$\overline{\mathcal{E}}_T$	$\overline{\mathcal{E}}_G$	$\overline{\mathcal{E}}_G^{best}$	**Presen-tation**	$\overline{\rho}$
0.9	$69.06 \pm 1.07\%$	$70.90 \pm 1.83\%$	88.10%	281485	1.029 ± 0.030
0.7	$66.43 \pm 1.37\%$	$68.10 \pm 1.82\%$	85.71%	248760	1.031 ± 0.036
0.5	$62.73 \pm 1.76\%$	$64.38 \pm 1.70\%$	83.33%	200510	1.032 ± 0.036
0.3	$59.95 \pm 1.41\%$	$60.90 \pm 2.11\%$	80.95%	177310	1.023 ± 0.045
0.1	$57.26 \pm 1.19\%$	$57.10 \pm 1.99\%$	78.57%	121385	0.999 ± 0.035

Table 8. Comparison of computational complexity for different β values for the *glass* problem

β	$\overline{P}_T$	**Total presented patterns**	**Cost Saving** $(\times 10^6)$
0.7	111.60 ± 1.75	$252\,744.90 \pm 2\,124.33$	$-773.633863 \pm 57.424777$
0.5	94.48 ± 1.98	$214\,170.60 \pm 2\,405.99$	$-1\,816.374341 \pm 65.038596$
0.3	81.30 ± 1.94	$180\,981.50 \pm 2\,468.39$	$-2\,713.542092 \pm 66.725490$
0.1	69.20 ± 3.03	$152\,859.90 \pm 2\,698.49$	$-3\,473.725183 \pm 72.945519$

Table 9. Comparison of error performance measures for different β values for the *wine* problem

β	$\overline{\mathcal{E}}_T$	$\overline{\mathcal{E}}_G$	$\overline{\mathcal{E}}_G^{best}$	**Presen-tation**	$\overline{\rho}$
0.9	$98.33 \pm 0.33\%$	$98.89 \pm 0.64\%$	100.0%	21317	1.006 ± 0.007
0.7	$97.56 \pm 0.36\%$	$97.97 \pm 0.86\%$	100.0%	9823	1.004 ± 0.011
0.5	$97.01 \pm 0.46\%$	$97.42 \pm 0.91\%$	100.0%	9956	1.004 ± 0.012
0.3	$96.95 \pm 0.41\%$	$97.14 \pm 0.94\%$	100.0%	8958	1.002 ± 0.012
0.1	$96.64 \pm 0.45\%$	$96.22 \pm 0.95\%$	100.0%	9597	0.996 ± 0.013

Table 10. Comparison of computational complexity for different β values for the *wine* problem

β	$\overline{P}_T$	**Total presented patterns**	**Cost Saving** $(\times 10^6)$
0.9	78.45 ± 2.88	$45\,117.77 \pm 1\,068.71$	-32.810974 ± 4.913917
0.7	60.42 ± 1.94	$34\,624.29 \pm 749.06$	-81.060013 ± 3.444165
0.5	51.39 ± 1.35	$28\,384.00 \pm 1\,138.55$	-109.752868 ± 5.235048
0.3	45.48 ± 1.35	$22\,697.76 \pm 1\,311.81$	-125.124575 ± 2.123308
0.1	40.97 ± 1.15	$22\,112.00 \pm 821.32$	-138.591524 ± 3.776447

Table 11. Training set reduction for different β values for the *glass* problem

	Reduction at epoch				
β	**50**	**100**	**500**	**1000**	**2000**
0.9	$1.14 \pm 0.30\%$	$2.97 \pm 0.43\%$	$7.76 \pm 0.51\%$	$12.86 \pm 1.06\%$	$18.36 \pm 1.15\%$
0.7	$7.03 \pm 0.62\%$	$9.80 \pm 0.84\%$	$21.66 \pm 1.14\%$	$28.35 \pm 1.27\%$	$50.55 \pm 1.02\%$
0.5	$16.69 \pm 1.46\%$	$21.84 \pm 1.06\%$	$37.24 \pm 1.31\%$	$39.15 \pm 1.61\%$	$45.07 \pm 1.15\%$
0.3	$27.66 \pm 1.47\%$	$33.29 \pm 0.84\%$	$47.84 \pm 0.95\%$	$51.21 \pm 1.49\%$	$52.73 \pm 1.13\%$
0.1	$37.30 \pm 2.06\%$	$41.13 \pm 1.43\%$	$54.95 \pm 1.66\%$	$59.91 \pm 1.47\%$	$59.77 \pm 1.76\%$

Table 12. Training set reduction for different β values for the *wine* problem

	Reduction at epoch			
β	**25**	**50**	**100**	**500**
0.9	$19.21 \pm 1.57\%$	$27.37 \pm 1.42\%$	$32.44 \pm 1.62\%$	$45.07 \pm 2.11\%$
0.7	$37.89 \pm 1.54\%$	$43.98 \pm 1.48\%$	$48.64 \pm 1.19\%$	$57.97 \pm 1.35\%$
0.5	$48.62 \pm 1.04\%$	$54.02 \pm 1.20\%$	$57.95 \pm 0.97\%$	$63.81 \pm 0.95\%$
0.3	$56.47 \pm 0.71\%$	$61.22 \pm 0.79\%$	$64.63 \pm 0.75\%$	$68.36 \pm 0.81\%$
0.1	$60.97 \pm 0.65\%$	$65.58 \pm 0.67\%$	$68.04 \pm 0.73\%$	$71.15 \pm 0.81\%$

Table 13. Comparison of convergence results for different β values for the *glass* problem (% of simulations that did converge to the specified generalization levels)

	Generalization Levels						
β	**60%**	**65%**	**70%**	**75%**	**80%**	**85%**	**90%**
0.9	0%	0%	10%	29%	65%	90%	98%
0.7	2%	6%	18%	46%	80%	96%	100%
0.5	2%	20%	38%	70%	94%	100%	100%
0.3	8%	28%	76%	88%	98%	100%	100%
0.1	16%	52%	84%	94%	100%	100%	100%

Table 14. Comparison of convergence results for different β values for the *wine* problem (% of simulations that did converge to the specified generalization levels)

	Generalization Levels			
β	**94%**	**96%**	**98%**	**100%**
0.9	0%	4%	12%	12%
0.7	0%	6%	38%	38%
0.5	0%	10%	36%	36%
0.3	4%	12%	42%	42%
0.1	4%	12%	38%	38%

Table 15. Comparison of SASLA ($\beta = 0.9$) and ES error performance measures

Problem	$\overline{\mathcal{E}}_T$	$\overline{\mathcal{E}}_G$	$\overline{\mathcal{E}}_G^{best}$	**Presen-tation**	$\overline{\rho}$
glass	(9.779 ± 1.118)	(10.143 ± 1.750)			(0.0005 ± 0.029)
SASLA	$69.06 \pm 1.07\%$	$70.90 \pm 1.83\%$	88.10%	281485	1.029 ± 0.030
ES	$59.28 \pm 1.15\%$	$60.76 \pm 2.27\%$	78.60%	192125	1.029 ± 0.043
wine	(0.060 ± 0.022)	(0.079 ± 0.029)			(-0.0144 ± 0.026)
SASLA	$98.33 \pm 0.33\%$	$98.89 \pm 0.64\%$	100.0%	21317	1.006 ± 0.007
ES	$92.29 \pm 0.02\%$	$90.97 \pm 0.03\%$	100.0%	5704	0.985 ± 0.020
circle	(0.039 ± 0.055)	(0.039 ± 0.052)			(0.0003 ± 0.027)
SASLA	$75.51 \pm 4.40\%$	$79.28 \pm 3.68\%$	94.08%	73264	1.059 ± 0.034
ES	$71.61 \pm 3.52\%$	$73.39 \pm 3.17\%$	86.51%	25992	1.059 ± 0.028

Table 16. Comparison of SASLA ($\beta = 0.9$) and ES computational complexity

Problem	$\overline{P}_T$	**Total presented patterns**	**Cost Saving** $(\times 10^6)$
glass			
SASLA	131.42 ± 1.67	$287\,906.74 \pm 1\,847.39$	176.860996 ± 49.938683
ES	67.92 ± 1.88	$183\,540.80 \pm 2\,037.27$	$-4\,333.405094 \pm 55.071407$
wine			
SASLA	78.00 ± 2.88	$45\,117.77 \pm 1\,068.71$	-32.810974 ± 4.913917
ES	11.16 ± 3.46	$7\,766.29 \pm 486.73$	-290.322597 ± 2.237983
circle			
SASLA	153.93 ± 14.19	$100\,459.60 \pm 4\,430.68$	-1.540722 ± 0.797522
ES	79.63 ± 12.81	$31\,324.80 \pm 1\,031.40$	-13.985036 ± 0.185652

Table 17. Comparison of SASLA ($\beta = 0.9$) and ES convergence results (% of simulations that did converge to the specified generalization levels)

Problem	**Generalization levels**						
glass	**60%**	**65%**	**70%**	**75%**	**80%**	**85%**	**90%**
SASLA	0%	0%	10%	30%	64%	90%	98%
ES	10%	18%	62%	90%	100%	100%	100%
wine	**90%**	**95%**	**96%**	**97%**	**98%**	**99%**	**100%**
SASLA	0%	0%	4%	4%	12%	12%	12%
ES	10%	48%	48%	78%	78%	78%	78%
circle	**75%**	**80%**	**85%**	**90%**	**95%**		
SASLA	0%	6%	40%	86%	100%		
ES	0%	14%	84%	100%	100%		