Chapter 3

ACTIVE LEARNING

One of the goals when designing and training a neural network is to maximize or to improve generalization, which is, the ability to give accurate response to data that the NN has not seen as part of the training process. In conventional backpropagation learning, all the available data are presented to the network for training. Learning on all the training data can be quite problematic especially when there are redundant data in the training set. The computational cost of training the network in terms of training time can become high, especially if these redundant data are included in the training set.

Studies have shown that selecting the most informative data for training rather than presenting all the available data to the network improves, or at least maintains the generalization performance. Selecting data for training also reduces training time and the data needed for training [Zhang 1994, Plutowski *et al* 1993, Engelbrecht *et al* 1998, Engelbrecht *et al* 1999a, Röbel 1994a].

This chapter discusses the concept and advantages of using of active learning. The objective of this chapter is to present a new selective learning algorithm and also to compare this new algorithm with three additional active learning algorithms with reference to their respective generalization performance, overfitting characteristics.

computational complexity and convergence characteristics.

3.1 Introduction

There have been various research efforts on improving the learning of BPNNs. These research efforts include finding optimal weight initialization [Rumelhart *et al* 1986], optimal learning rate and momentum [Plaut *et al* 1986 , Weir 1990, Yu *et al* 1997], finding the optimal architectures [Le Cun 1990, Karnin 1990, Hirose *et al* 1991, Pelillo *et al* 1993, Engelbrecht *et al* 1996], using second order optimization techniques [Becker *et al* 1988], adaptive activation functions [Zurada 1992a, Engelbrecht *et al* 1995, Fletcher *et al* 1998] and active learning [Zhang 1994, Engelbrecht *et al* 1999a, Engelbrecht *et al* 1998, Röbel 1994a]. This chapter concentrates on active learning as an approach to improve performance of NNs. Active learning is a technique in which patterns that have the highest influence on weight changes are dynamically selected by the NN learner from a candidate set of training patterns. The network utilizes current attained knowledge about the tasks to be learned as encapsulated in the current weights to select the most informative training patterns. There are two main approaches to active learning:

- **Incremental learning,** where patterns are selected and removed from a candidate training set. The selected patterns are added or injected into the actual training set. The effect is that the actual training set grows as training progresses, while the candidate training set is pruned.
- **Selective learning,** where a subset of the training patterns that satisfy a selection criterion is selected from a candidate training set and used for training. Unlike the incremental learning, the candidate training set is not pruned. At each pattern selection interval, all the patterns in the candidate set have a

chance to be selected. The candidate set remain fixed while the size of the actual training set varies from time to time.

A brief outline of this chapter is as follows: Section 3.2 discusses the concept of active learning and section 3.3 presents a general algorithm for active learning. A new selective learning strategy for time series problems is presented and compared with three other active learning algorithms in section 3.4. Section 3.5 presents results obtained from the different learning algorithms. Finally, section 3.6 highlights the conclusions, comments and observations.

3.2 Concept of Active Learning

Active learning has emerged as an efficient alternative to improve the performance of multilayer layer NNs. Active learning refers to the selection of a subset of the available training data dynamically during training, where the subset contains the most informative data. The objective of active learning algorithms is to identify, and train on the most informative patterns in a candidate training set. Active learning *efficiently* selects *optimal* training patterns from available training patterns for training the network. Efficiency refers to the complexity of the pattern selection mechanism which should be minimized. Optimal patterns are patterns that have useful information about the current state of the network and such patterns bring the network closer to the target function. The network plays an active role in data selection. Rather than being a passive learner, the network utilizes information based on its current state to gather useful information for further training.

Active learning addresses two fundamental questions:

– Which of the patterns should be selected for training from the candidate set?

– When should an additional set of patterns be selected?

Answers to these questions have resulted in the design of different approaches to active learning. These approaches mainly use the error in prediction as selection criterion [Cohn 1994, Röbel 1994a, Zhang 1994, Plutowski *et al* 1993] or changes in outputs due to perturbations in input parameters [Engelbrecht *et al* 1998, Engelbrecht *et al* 1999a, Engelbrecht *et al* 1999c]. Pattern selection has been the focus of many research. Infact, active learning has been called various names such as query learning, incremental learning, selective learning and dynamic pattern selection. All these terms refer to the same basic concept of selecting a subset of the most informative patterns from the candidate training set. Active learning algorithms aim at:

– Improving, or at least maintaining the generalization ability of the network.

– Reducing the cost of training the network in terms of the number of patterns needed for training. But selecting these patterns should not exceed the gain in computational cost reduction achieved by reducing the training set size.

– Improving the speed of convergence. Convergence is the ability to achieve certain generalization levels. Active learning aims at increasing the probability that the network will converge to given generalization levels, and doing so in as less time as possible.

Section 3.2.1 presents an overview of different approaches to active learning.

3.2.1 Overview on Active Learning

Plutowski and White used error in prediction as their selection criterion [Plutowski *et al* 1993]. The integrated squared bias (ISB) is used as the error term. Patterns that maximize the decrement in the ISB of the network resulting from

adding such patterns to the training subset are selected for training. Additional patterns are selected when the training error on the current training subset is sufficiently small. While Plutowski and White used the bias term of the MSE as the selection criterion, Cohn used the variance term of the MSE [Cohn 1994]. In this case, the most informative patterns are those that maximize the change in variance of the network resulting from adding these patterns to the network. The learner (NN) selects an additional training pattern at each time step, or epoch. In Mackay's algorithm, information theory was used to select patterns for training [Mackay 1992]. However, Mackay applied his active algorithm within bayesian framework. Fukumizu selected patterns that minimize the estimation error i.e. the expected value of the MSE [Fukumizu 1996] for training. Sung *et al* also used the error function as their selection criterion, but they considered both the bias and variance term [Sung *et al* 1996]. Patterns that minimize the expected misfit, i.e. the total output uncertainty between the target and the estimated target function are selected for training. Seung *et al* developed an active learning algorithm which they called Query by Committee (QBS) [Seung *et al* 1992]. In QBC, the degree of disagreement among the committee of learners (students) serves as an estimate of information value. The query that has the maximum disagreement among the committee of learners is chosen for training. That is, QBC selects an input classified as positive by half of the committee and negative by the other half. Freund *et al* presented a more complete and general analysis of QBC using the batch training algorithm [Freund *et al* 1997]. Hara *et al* applied an active learning algorithm to pattern classification problems [Hara *et al* 1988]. Patterns selected for training are those patterns that are close to the boundary of the pattern classes. Cohn *et al* combined active learning with statistical models (gaussian and weighted regression) [Cohn *et al* 1996]. Patterns that give the lowest expected model variance are selected for training.

Similar to Plutowski *et al*, Sung and Cohn, Zhang used the error in prediction as selection criterion [Zhang 1994]. Patterns whose addition cause the maximum approximation error to the network, are selected for training. However, the relative value rather than the absolute value of the error term is used. Röbel used the same error criterion as Zhang in selecting the most informative patterns for training[Röbel 1994a]. Röbel and Zhangs' algorithms only differ in the criterion that triggers the subset selection. Training on the current subset continues until some criteria are triggered (refer to section 3.4).

The change in outputs due to perturbations in input parameters can also used as selection criterion. Engelbrecht proposed an active learning algorithm where patterns with the highest influence on the outputs are selected for training [Engelbrecht *et al* 1999a]. First order derivatives of the output units with respect to the input units are used to quantify the influence a pattern has on the outputs. Another active learning algorithm that uses output perturbation as selection criterion is the selective learning algorithm (SLA) [Engelbrecht *et al* 1999c], developed in this thesis. Patterns that influence the output most are selected more for training than patterns that have little influence on the output. The influence on the output is reflected in the next-time-change in output values. Thus, patterns that have the large next-time-change in output values are selected more into the current training set than patterns with small-time-change in output.

This thesis selects four active learning algorithms based on their selection criteria for comparison. Two of these algorithms uses the error in prediction as selection criterion while the other two algorithms uses changes in output as their selection criteria. The algorithms selected are:

1. error based criterion
   - Accelerated learning using active learning, developed by Zhang [Zhang 1994].
   - Dynamic pattern selection (DPS), developed by Röbel [Röbel 1994a].

2. output based criterion
   - Sensitivity analysis incremental learning (SAILA), developed by Engelbrecht [Engelbrecht *et al* 1999a].
   - Selective learning algorithm (SLA), developed in this thesis.

While Cohn, Plutowski *et al* and Sung *et al* based their selection criteria on information theory, Zhang has shown that their approach is similar to selection of patterns using the prediction error as selection criterion [Zhang 1994]. Selection of patterns using the largest error is computationally less expensive than using information theoretic approaches. For these reasons, this thesis chose the algorithms developed by Röbel [Röbel 1994a] and Zhang [Zhang 1994] in its comparison instead of the information theoretic approach. The next section presents a general algorithm for active learning.

3.3 General Algorithm for Active Learning

This section presents a general algorithm for active learning and then discusses the algorithm design issues. Let D_T be the current training set, which has all the patterns selected for training, D_C be the candidate training set, which contains all the available patterns and D_V be the validation set, which contains patterns not used as part of training and is used to test for overfitting.

A general algorithm for active learning is summarized below:

1. Initialize weights randomly as in conventional back propagation.
2. Select the most *informative* pattern(s) into training set D_T from D_C
   - for incremental learning, add the selected pattern(s) into D_T and remove them from D_C,

- for selective learning, select patterns into D_T.

3. Train the network for a training interval (i.e. adjust the weights) using D_T.
4. If the network has reached the desired accuracy or has reached the maximum number of epochs, stop training.
5. If a subset selection criteria triggers, repeat from step (2)
   otherwise repeat from step (3).

A training interval maybe one epoch for online training or ϵ epochs if batch training method is used.

Design issues in active learning algorithm

When designing and implementing an active learning algorithm, some issues have to be taken into consideration. One of these design issues is the number of patterns to select at each selection interval, referred to as the subset size. Although there is no heuristic to determine the size of the training subset, there are guidelines. For incremental learning, it is advisable to train with a small subset size because the subset grows during training. Selecting a small subset means patterns will be selected more often and thus increase the complexity of the network due to the cost of selecting the patterns. But, selecting a large number of patterns during training may defeat the aim of active learning, which is to reduce the number of patterns needed for training. Therefore, good selection criteria are desirable in selecting patterns that have useful information about the network. For selective learning, the subset size depends on the selection criteria. Therefore, selection criteria are important to ensure that the most informative patterns are selected for training.

Another important issue to consider is when to select additional patterns. One or more subsetselection criteria can be used. These subsetselection criteria should ensure that the NN does not train too long on a training subset because the network may overfit the training subset. The network should also train long enough to acquire maximum information on the current training subset. Different algorithms have

different subsetselection criteria. Patterns can be selected at each training epoch [Cohn 1994, Engelbrecht *et al* 1999c]. This criterion can have high computational cost because the subset selection function is applied at each epoch. An alternative is to select at each ξ epochs. A reduction in computational cost will be achieved since the number of pattern selections is fewer. However, if the selection interval ξ is too large, overfitting of the training subset may occur. Patterns can also be selected using the error on the training and the validation set. New patterns are selected into D_T as soon as the error on the training or validation set reduces to a specified level [Zhang 1994, Engelbrecht *et al* 1999a].

Another criterion is to select new patterns as soon as the network overfits the training subset [Engelbrecht *et al* 1999a, Röbel 1994a]. Different algorithms use a criterion or combination of criteria to decide when to select additional patterns. The different subsetselection criteria used for the four selected active learning algorithms are discussed in more details in section 3.4.

3.4 A Comparative Study of Four Selected Active Learning Algorithms

The objective of this thesis is to compare selected active learning algorithms with reference to generalization performance and computational complexity. For this purpose, a new selective learning algorithm for time series problems developed in this thesis, sensitivity analysis for incremental learning [Engelbrecht *et al* 1999a], dynamic pattern selection of Röbel [Röbel 1994a] and accelerated learning using active example selection of Zhang [Zhang 1994], are compared with one another. This section presents an overview and critique of these algorithms.

3.4.1 A New Selective Learning Algorithm

Kohara presented an algorithm that performs pre-processing of the training set for time series problems [Kohara 1995]. Kohara's algorithm divides the training set into two sets. The one set contains all training patterns that reflect large-next-time changes in the time series, while the other contains patterns that reflect small-next-time changes. Kohara's algorithm assumes one output unit and also assumed a time ordering among the training patterns. Kohara uses target values to determine next-time changes. The next-time-change $\Upsilon_k^{(p)}$ for output k is defined as

$$\Upsilon_k^{(p)} = t_k^{(p+1)} - t_k^{(p)}$$

The two training sets remain fixed during training. During training, patterns are more frequently selected from the large-next-time changes set than from the small-next-time changes set. Therefore, Kohara's algorithm is not considered as an active learning algorithm, because the neural network plays no role in the selection of patterns. Kohara's approach is rather referred to as a training set manipulation technique.

A new output based selective learning algorithm is proposed in this thesis based on Kohara's algorithm. Instead of using the target values to construct the two training subsets, the actual outputs of the network are used. Therefore, next-time-change $\Psi_k^{(p)}$ for output k is defined as

$$\Psi_k^{(p)} = o_k^{(p+1)} - o_k^{(p)} \tag{3.1}$$

$\Psi_k^{(p)}$ can only be computed for the first $P_C - 1$ patterns, where P_C is the number of patterns in the candidate training set D_C. The division of the original training set into large- and small-next-time changes sets is done after each selection interval, which is one epoch.

More patterns (80%) from the large-next-time changes set are randomly selected than from the small-next-time changes (20%) into the current training subset D_T. In doing this, the two subsets reflect the current knowledge of the learner, in that the set reflects what the learner perceives as large and small changes. More patterns are selected from the large-next-day changes set, since these patterns contain the most information about the characteristics of the time series. A large change in the output values, causes a large change in the the weights and a large change in weights means more information is gained in bringing the network's output closer to the target function (refer to weight update equations (2.13) and (2.16)).

Active learning is introduced by calculating the next-time-changes based on the actual output of the network and not on the target output values. At each epoch, the current training subset is discarded and a new subset D_T is selected with training patterns. The training set D_T is then used for training.

The algorithm for SLA is summarized below:

1. Initialize weights, learning rate and momentum.
2. Calculate the output $o_k^{(p)}$ of the network for each pattern p. Then, calculate next-time-changes as

$$\Psi_k^{(p)} = o_k^{(p+1)} - o_k^{(p)} \ , \ \forall_p \in D_C$$

3. Separate patterns into a small-time and a large-time-change set:
   - calculate the average next-time-changes:

$$\overline{\Psi} = \frac{\sum_{p=1}^{P_C-1} \Psi^{(p)}}{P-1}$$

   - divide the candidate patterns into the two training subsets:
     Add all patterns p for which $\Psi_k^{(p)} > \overline{\Psi}$ to the large-change set and all other patterns into the small-change set.
4. Select the actual training set D_T to consist of

- 80% of the large-change set, randomly selected
- 20% of the small-change set, randomly selected.

5. Train the network for one epoch using D_T
6. Repeat steps (2 - 5) until the number of epochs exceeds the maximum number of epochs allowed.

The subset size depends mainly on the size of the large-next-time changes set. The performance of the proposed algorithm is compared to the other algorithms and the results are discussed in the section 3.5

3.4.2 Sensitivity Analysis Incremental Learning Algorithm

The second algorithm to be studied is an *incremental learning* approach to active learning which uses an output based selection criterion, referred to as sensitivity analysis incremental learning algorithm (SAILA). SAILA is developed by Engelbrecht [Engelbrecht *et al* 1999a]. In SAILA, the most informative patterns are perceived as those patterns that maximally influence the output of the NN in the presence of small input perturbations. First order derivatives of the output units with respect to the input units are used to compute the influence the pattern has on the output value of the function approximated by the network. Patterns with the highest sensitivity cause the largest change in weights (large change in weights achieve maximum gain in bringing the approximation closer to the true function). These patterns lie in the region of the peaks' derivatives. Thus, the partial derivatives $\frac{\partial o_k}{\partial z_i^{(p)}}$ is calculated for each input and output for each pattern. Training on such patterns yield better generalization and faster convergence [Engelbrecht *et al* 1999a]. The sensitivity of each pattern is determined by computing the informativeness of the pattern, as

$$\Phi^{(p)} = \max\{S_{o,k}^{(p)}\} \tag{3.2}$$

where

$$S_{o,k}^{(p)} = \sqrt{\sum_{i=1}^{I} (\frac{\partial o_k}{\partial z_i^{(p)}})^2} \tag{3.3}$$

The larger the value of $\Phi^{(p)}$, the more informative that pattern is. Pattern(s) with the largest absolute value of $\Phi^{(p)}$ are selected into D_T.

SAILA continues training on the training subset to achieve maximum gain from the patterns before selecting additional patterns. At the same time, the network must not be allowed to memorize the training subset. The network should therefore not spend too much time on the current training set. SAILA uses the following subset selection criteria:

1. The algorithm limits the number of training epochs on the current subset. The criterion ensures that the NN does not train indefinitely on the subset. Engelbrecht limited the number to 100 in his implementation.
2. If the error on D_T, or the validation set D_V, decreases sufficiently, a new subset is selected for training. The criterion ensures that the NN achieves sufficient gain on the current training subset before selecting additional patterns. In Engelbrecht's implementation, an additional pattern is selected into D_T as soon as the error on the D_T or D_V decreases by 80%.
3. A new subset is also selected if the average decrease in error on D_T and D_V since training started on the current subset is small. The criterion will prevent the learner from training on D_T with achieving too little gain.
4. If the error $\mathcal{E}_V$ on the validation set increases too much, a new subset is selected. The subset selection criterion prevents the NN from memorizing the current training subset by triggering a new subset selection as soon as overfitting of D_T is observed.

The sensitivity analysis incremental learning algorithm is summarized below [Engelbrecht *et al* 1999a]:

1. Initialize weights, momentum and learning rate. Initialize the subset size, P_{S_s}, i.e the number of patterns selected from the candidate set D_C. Construct the initial training subset $D_{S_o} \subset D_C$. Let $D_T \leftarrow D_{S_o}$ be the current training set.

2. Repeat

   - Repeat

     Train the NN on training set D_T

     until a termination criterion on D_T is triggered (as discussed above).

   - Compute the new training subset D_{S_s}

     * For each $p \in D_C$, compute the sensitivity matrix $S^{(p)}_{oz,ki}$ for sigmoid activation functions:

$$S^{(p)}_{oz,ki} = f^{(p)'}_{o_k} \sum_{j=1}^{J} w_{kj} f^{(p)'}_{y_j} v_{ji}$$

     * Compute the output sensitivity vector $\vec{S}^{(p)}_{o,k}$ for each $p \in D_C$

$$\vec{S}^{(p)}_{o,k} = ||S^{(p)}_{oz}||$$

     * Compute the informativeness $\Phi^{(p)}$ of each pattern $p \in D_C$ using

$$\Phi^{(p)} = \max_{k=1,\cdots,K} \{|S^{(p)}_{o,k}|\}$$

     * Find the subset D_{S_s} of the P_{S_s} most informative patterns as

$$D_{S_s} \leftarrow \{p \in D_C | \Phi^{(p)}_{\infty} = \max_{q=1,\cdots,P_C} \{\Phi^{(q)}_{\infty}\}; \forall q \in D_C, not \;\; yet \;\; selected\}$$

     where P_C is the number of patterns remaining in D_C. Then, let $D_T \leftarrow D_T \cup D_{S_s}$ and $D_C \leftarrow D_C - D_{S_s}$

   until convergence is reached.

In Engelbrecht's implementation, SAILA started training with one pattern, and selects only one new pattern at each subsetselection interval. Although the subset

selection criteria implemented by SAILA considers overfitting effect and generalization of the network, the cost of selecting patterns in SAILA could be high. Because SAILA has more criteria to implement when selecting new patterns for training. Results of SAILA when applied to function approximation and time series problems are presented in the section 3.5.

3.4.3 Dynamic Pattern Selection

The third active learning algorithm studied is Röbel's dynamic pattern selection technique (DPS). Unlike SAILA and SLA, DPS uses the error indication as selection criterion. DPS is an example of incremental learning , where informativeness of each pattern is measured using the prediction error. The prediction error is computed as $(t_k^{(p)} - o_k^{(p)})^2$. Patterns with the largest prediction error are the most informative and are selected for training.

Training continues on the current training subset until the subset starts to overfit. To measure overfitting, Röbel defined the generalization factor ρ as

$$\rho = \frac{E_V}{(E_T + E_C)}$$

where E_V, E_T and E_C are the error functions on the validation set, training subset and the candidate training set respectively. By requiring that $\rho \leq 1.0$, overfitting is prevented. A value of ρ greater than one means that the validation error is larger than the training the training error, hence bad generalization. New patterns are therefore selected into D_T when ρ grows beyond one. However, Röbel discovered that each pattern selected for training decreases the value of ρ to a minimum value before ρ slowly increases again and therefore takes a long time to reach the value of one. This means that the network will train too long on the current training subset if only the selection criterion of $\rho > 1.0$ is implemented. Thus, new patterns are also selected as soon as ρ reaches a minimum threshold value. For these purpose, a

threshold ϕ_ρ is defined as

$$\phi_\rho(\xi) = \min\{\phi_\rho(\xi - 1), \overline{\rho} + \sigma_\rho, 1.0\}$$

where ξ is the current training epoch, $\overline{\rho}$ and σ_ρ are the average and standard deviation of the generalization factor for M preceding epochs respectively. Röbel used 100 for the value of M. The DPS algorithm selects new patterns whenever $\rho(\xi) > \phi_\rho(\xi)$; $\rho(\xi)$ is the generalization factor for the current training epoch.

The algorithm for DPS is summarized below:

1. Initialize the weights and set threshold $\phi_\rho(\xi)$ to one with $\xi = 1$.
2. For each pattern p in D_C, calculate the SSE as $E^{(p)} = \sum_{k=1}^{K} (t_k^{(p)} - o_k^{(p)})^2$.
3. Select pattern(s) with the highest error $E^{(p)}$ into D_T and remove the selected pattern(s) from D_C.
4. Train the network.
5. If the number of epochs exceeds the maximum number of epochs or the error limit has been reached, stop training.
6. Calculate the generalization factor $\rho = \frac{E_V}{(E_T + E_C)}$
7. if ρ is greater than $\phi_\rho(\xi)$, then set $\phi_\rho(\xi + 1) = \min(\rho(\xi), 1.0)$ and repeat from step (2), otherwise set $\phi_\rho(\xi + 1) = \min\{\phi_\rho(\xi), \overline{\rho} + \sigma_\rho, 1.0\}$ and repeat from step (4).

Röbel used a subset size of one pattern and selects a pattern when the subsetselection criterion is triggered. The online cross validation technique is used to check for the generalization. That is, a separate data is used compute the validation error of the network. Additional overhead is incurred in DPS for implementing the cross validation technique. If the training data is limited, having a separate set for validation may not be feasible.

While the selection criterion is easy to compute, performance degrades if the training data have outliers and noise. Outliers will be selected as the most informative patterns, since these patterns have the maximum prediction errors. The network is then biased towards the outliers. Consequently, the ability of the network to generalize deteriorates. Results of DPS when implemented are discussed in section 3.5.

3.4.4 Accelerated Learning by Active Example Selection

The last algorithm to be studied is accelerated learning by active example selection (AL) proposed by Zhang [Zhang 1994]. AL is an incremental learning approach. AL selects as the most informative patterns, those patterns that have the maximum prediction error, where the prediction error is computed as $(t_k^{(p)} - o_k^{(p)})^2$.

New patterns are selected for training when the error on the training subset is reduced to a specified performance level κ, where κ is computed as

$$\kappa = \frac{J(I+K)}{\tau}$$

τ is the allowable error per connection. Zhang suggested a value of $\tau \in [100, 200]$.

Zhang motivates the selection criterion on the fact that the learning capacity of a NN is proportional to the total number of adjustable connections in the network, which is $J(I+K)$ [Zhang 1994].

The algorithm for Zhang's accelerated learning is

1. Initialize weights to small random values.
2. For each pattern p in D_C, compute the SSE as

$$E^{(p)} = \sum_{k=1}^{K} (t_k^{(p)} - o_k^{(p)})^2$$

3. Select pattern(s) with the highest error into D_T and remove from D_C.

4. Train the network.
5. If the maximum number of epochs is exceeded or if the error limit has been reached, stop training.
6. Compute $\kappa = \frac{J(K+I)}{\tau}$ and SSE on D_T as $E_T = \sum_{p=1}^{P} \sum_{k=1}^{K} (t_k^{(p)} - o_k^{(p)})^2$.
7. If $\kappa \leq E_T$ then repeat from step (2),
   otherwise repeat from step (4).

Even though AL's subset termination criterion is less complex and easy to compute, AL does not test for generalization of the network. Overfitting may therefore still occur. The subselection termination criterion depends on the architecture of the network being trained. If the wrong architecture is selected (either undersized or oversized) this criterion may not perform well. Due to the selection of patterns with the largest prediction error, the performance of AL may deteriorate in the presence of outliers and or noise.

The results and performance of AL are discussed in section 3.5.

3.5 Experimental Results

Four approximation and times series problems of varying complexity were used to test the performance of SLA, SAILA, DPS and AL. These problems differ in input dimensions and the number of hidden units needed to train the network. Table 3.1 shows a summary of the NN architecture used for these problems. In table 3.1, the architecture of a NN is referred to as I-J-K where I is the number of input units, J is the number of hidden units and K is the number of output unit i.e. the notation 2-5-1 means two input, five hidden and one output units are used.

All the available data was split into three sets: the candidate training set D_t, validation set D_V and generalization set D_G. The three sets were randomly created such

Problem	*Equation*	P_C-P_G-P_V	*Architecture*
F1	(3.3)	600 -200-200	2-5-1
TS1	(3.4)	600-200-200	1-5-1
TS2	(3.5)	600-200-200	2-5-1
TS3	(3.6)	140-60-60	10-10-1
TS4		600-200-200	2-5-1
TS5		600-200-200	2-7-1

Table 3.1: Summary of the functions and time series used

that

$$\begin{aligned} D_C \cap D_V &= \emptyset \\ D_C \cap D_G &= \emptyset \\ D_G \cap D_V &= \emptyset \end{aligned}$$

Let P_C be the number of training patterns in D_C, P_V the number of training patterns in D_V and P_G the number of patterns in test set D_G. Table 3.1 shows the size of these sets for each problem. D_C is the candidate training set from which training patterns are selected. D_V contains data used to determine the generalization factor during training. D_G contains data used to determine the generalization performance of the network.

The performance of the active learning algorithms was tested on clean and noisy data, as well as data containing outliers. Section 3.5.1 explains the experimental procedure, including a discussion of the performance criteria used to compare the learning algorithms. The results are compared in section 3.5.2.

The characteristics of the functions and time series used for experimentation are discussed next. The following functions and time series were used:

1. Function $F1$ is defined as (see figure 3.1(a))

$$F1 : F(z_1, z_2) = \frac{1}{2}(z_1^2 + z_2^2) \tag{3.4}$$

where $z_1, z_2 \sim U(-1, 1)$. All target values were scaled to the range [0,1].

2. Time series TS1 is a sine function defined as (see figure 3.1(b)),

$$TS1 : F(z) = \sin(2\pi z)e^{(-z)} + \zeta \tag{3.5}$$

where $z \sim U(-1, 1)$ and $\zeta \sim N(0, 0.1)$. Target values were scaled to the range [0,1].

3. Time series TS2 is the henon-map function defined as (refer to figure 3.1(c)),

$$\begin{aligned} TS2 : o_t &= z_t \\ z_t &= 1 + 0.3z_{t-2} + 1.4z_{t-1}^2 \end{aligned} \tag{3.6}$$

where $z_1, z_2 \sim U(-1, 1)$. The target values were scaled to the range [0,1].

4. Time series TS3 is a difficult time series, having 10 input parameters of which 7 are irrelevant (see figure 3.2(c)).

$$\begin{aligned} TS3 : o_t &= z_t \\ z_t &= 0.3z_{t-6} - 0.6z_{t-4} + 0.5z_{t-1} + 0.3z_{t-6}^2 - 0.2z_{t-4}^2 + \zeta_t \end{aligned} \tag{3.7}$$

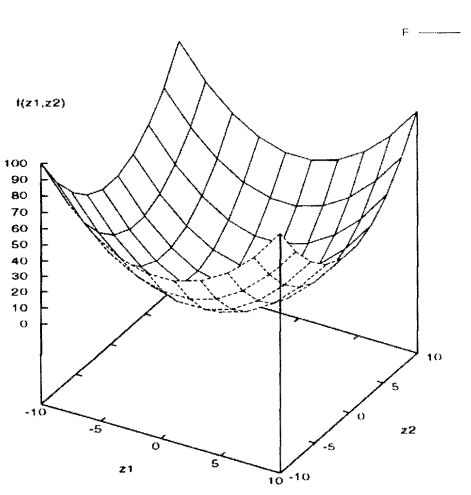
for $t = 1, \cdots, 10$, where $z_4, z_6, z_9 \sim U(-1, 1)$ and $\zeta_t \sim N(0, 0.05)$. All target values were scaled to the range [0,1].

5. Time series TS4 is a convolution of two discrete functions with outliers. Figure 3.2(a) shows an illustration of this function.

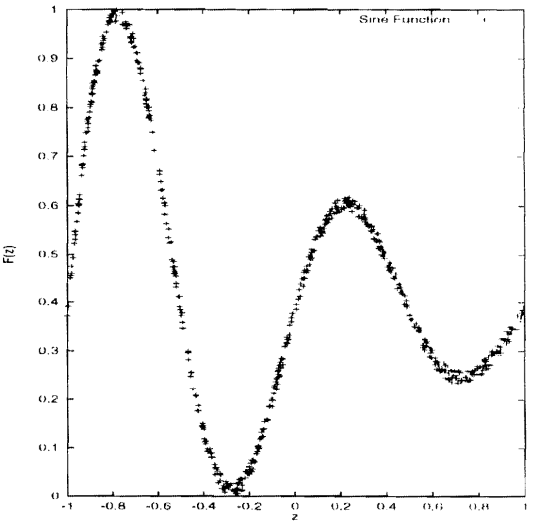
6. Time series TS5 is the sine function TS1 with 5% of the candidate training set consisting of outliers (see figure 3.2(b)).

3.5.1 Experimental Procedure

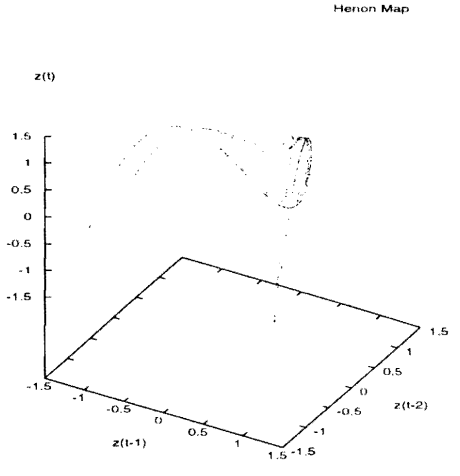
In order to obtain statistically valid assertions in comparing experimental results of the four learning algorithms, thirty simulations were performed for each problem.



(a) F1

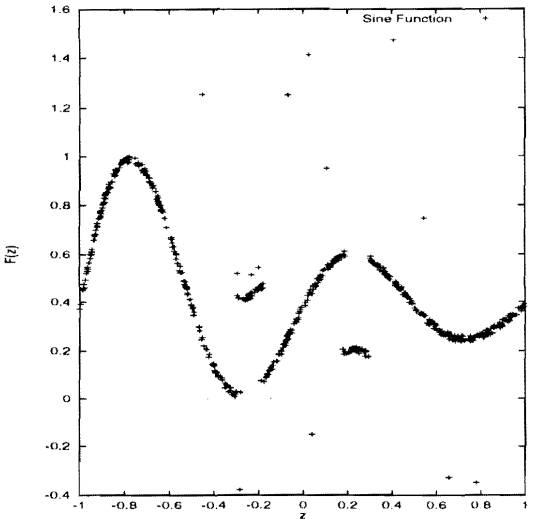


(b) TS1

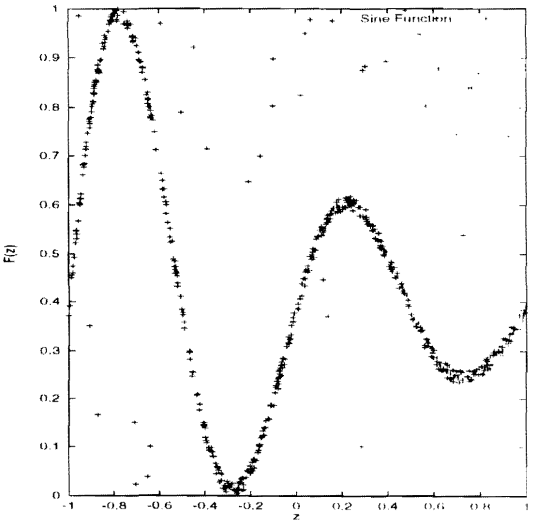


(c) TS2

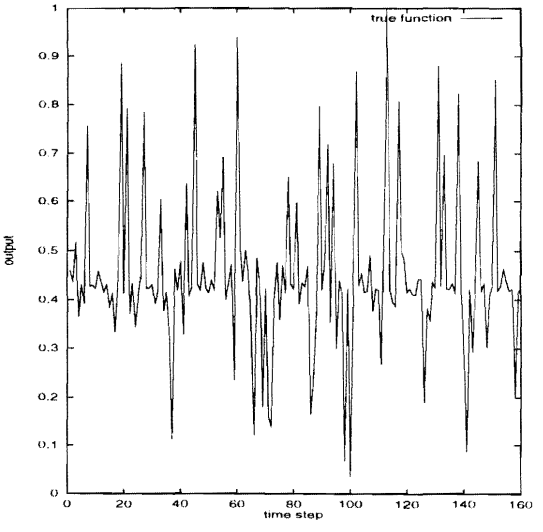
Figure 3.1: Function and Time series problems to be approximated



(a) TS4



(b) TS5



(c) TS3

Figure 3.2: Time series problems to be approximated

Online training was used for the active learning algorithms. The initial subset size for incremental learning algorithms consisted of one pattern and a subselection size of one pattern was used. Each simulations was executed for 2000 epochs. A learning rate 0.1 and momentum 0.9 were used for all the approximation problems . Results reported are averages over the 30 simulations together with 95% confidence intervals as obtained from the t-distribution.

The selective learning algorithm was not applied to F1, since F1 is not a time series problem. The τ value used in the subset selection criterion for AL was adjusted for each problem using a trial and error approach. For TS3, a high τ was used ($\tau = 1000$), a value of 100 was used for TS1, TS4 and TS5 while a value of 180 was used for TS2 and F1.

Performance measures

To evaluate the performance of each learning algorithm, the following performance criteria were used:

1. The mean squared error (MSE) was used as a measure of accuracy. The MSE measures how well a function is approximated by the network, and is defined as

$$MSE = \frac{\sum_{p=1}^{P} \sum_{k=1}^{K} \left(t_k^{(p)} - o_k^{(p)}\right)^2}{2\ K\ P}$$

.

A MSE value close to zero shows a small error between the target and the output function. The MSE over the three sets D_V, D_G and D_C were computed. The MSE over D_G, denoted by E_G provides an unbiased estimate of the generalization error since the patterns in D_G were not used for training.

2. Röbel's generalization factor ρ was used to measure overfitting effects. The generalization factor was computed as $\rho = \frac{E_V}{E_C}$, where E_C is the MSE over candidate training set D_C and E_V is the MSE over the validation set D_V. A network overfits when the value of ρ increases substantially above 1.

3. The computational complexity of learning algorithms was also used as performance criterion. For the purpose of this thesis, computational cost is measured as the number of calculations needed to train the network. Calculations include subtraction, multiplication, addition and division.

   At any epoch ξ, the cost C_{fc} of training a NN on a training set, is expressed as

   $$C_{fc} = (C_V + C_W) * P_T$$

   where C_V is the cost of updating weights between input and hidden units and C_W is the cost of updating weights between hidden and output units. P_T is the number of patterns in the training subset D_T. For conventional backpropagation with fixed set learning, $P_T = P_C$. Thus the cost of training C_{fsl} is computed as $C_{fsl} = (C_V + C_W) * P_C$.

   The costs of updating the weights are calculated as

   $$\begin{aligned} C_V &= C_v * (N_V) \\ C_W &= C_w * (N_W) \end{aligned}$$

   where C_v is the cost of updating a single weight between the input and hidden layers, C_w is the cost of updating a single weight between the hidden and output layers. C_V is the total cost of updating the weight connections between the input and the hidden layers, and C_W is the total cost of updating the weight connections between the hidden and output layer. N_V is the total number of connections between the input and hidden layers and N_W is the total number of connections between the hidden and output layers.

   The total number of connections N_V and N_W are expressed as

   $$N_V = (I + 1) * (J + 1)$$
   $$N_W = (J + 1) * (K)$$

   and

   $$C_v = 13$$

$$C_w = 11$$

Therefore,

$$\begin{aligned} C_V &= 13 * (I+1) * (J+1) \\ C_W &= 11 * (J+1) * K \end{aligned} \tag{3.8}$$

The cost of training a network using any active learning algorithm includes C_{fc}, the cost of selecting patterns for training and the cost of computing the subset termination criterion. Therefore, at any epoch ξ, the cost of training a network using SLA, SAILA, DPS and AL are:

$$\begin{aligned} C_{SL} &= C_{fc} + C_{sla} * P_C \\ C_{DP} &= C_{fc} + C_{dps} * (P_C - P_T) + (C_{S_{dps}} * P_T) \\ C_{AL} &= C_{fc} + C_{al} * (P_C - P_T) + (C_{S_{al}} * P_T) \\ C_{SA} &= C_{fc} + C_{sai} * (P_C - P_T) + (C_{S_{sai}} * P_T) \end{aligned}$$

For all the incremental learning algorithms, the subset selection criteria are tested on the remaining patterns in the candidate set D_C which is equal to $P_C - P_T$. Also, for incremental learning algorithms, an additional cost of selecting pattern is incurred when a pattern is selected.

C_{SL}, C_{SA}, C_{AL} and C_{DP} are the cost of training a network using SLA, SAILA, AL and DPS respectively. $C_{sla} = 15$ is the cost of computing the subset selection criteria for SLA, $C_{dps} = 11$ is the cost of computing the subset selection criteria for DPS, $C_{al} = 4$ is the cost of computing the subset selection criteria for AL and $C_{sai} = 18$ is the cost of computing the subset selection criteria for SAILA. $C_{S_{dps}} = 2$, $C_{S_{al}} = 2$ and $C_{S_{sai}} = 7$ are the cost of selecting patterns into D_T for DPS, AL and SAILA respectively.

Therefore,

$$C_{SL} = C_{fc} + 15P_C$$

$$\begin{aligned} C_{DP} &= C_{fc} + 11(P_C - P_T) + 2P_T \\ C_{AL} &= C_{fc} + 4(P_C - P_T) + 2P_T \\ C_{SA} &= C_{fc} * P_T + 18(P_C - P_T) + 7P_T \end{aligned} \tag{3.9}$$

From equation (3.9), the cost of training is directly proportional to the number of patterns selected for training. The more patterns are selected for training. the higher the computational cost. Initially, P_T for SLA is greater than the other algorithms because DPS, AL and SAILA are incremental learning algorithm and a small initial trainig set and subset size is used in the simulations. Thus, C_{SL} is expected to be greater than C_{AL}, C_{DPS} and C_{SA} initially. SAILA is computationally more expensive in selection criteria than the other algorithms because SAILA has more subset selection criteria to implement than the other algorithms.

Section 3.5.2 illustrates the costs for the different algorithms.

3.5.2 Results

This section presents the results of the simulations carried out on the active learning algorithms.

Training error

In order to compare the performance of the four active learning algorithms. the MSE over the candidate set D_C was computed for the simulations and the average calculated. Tables (3.2) and (3.3) show the result over 2000 epochs for clean data and data with noise and outliers.

For TS1, DPS had a very low error with the lowest variance which means that all the errors of the simulations for DPS were all closer to the average error of 0.0003. Although, SLA had a low error as well. However SLA had a large variance when compared to DPS. AL had the largest error with a very large variance.

DPS achieved the smallest error for TS2, having a small variance. For TS3, all the algorithms had very low errors but SAILA had a high variance. DPS had the smallest error for F1 with a very small variance.

For TS4 and TS5, SLA achieved the smallest error with the lowest variance. AL had the largest error for TS4 and TS5. This is because AL selected and trained on just a single pattern for TS4 and an average of 4 patterns for TS5. Thus AL had high errors for TS4 and TS5.

The training errors for all the problems with noise and outliers were larger ($\times 10^2$) than for problems with clean data. DPS had the lowest average error for clean data while SLA had the lowest error for noisy data.

Generalization error

To compare the generalization ability of the four active learning algorithms, the MSE over the generalization set, E_G, was computed and the average over the 30 simulations was plotted as a function of number of epochs. Figures 3.3 and 3.4 illustrates the trend of the generalization errors for the entire training period.

DPS achieved a very low average error faster than the other algorithms for F1 (refer to figure 3.3(a)). However, both SAILA and AL achieved a comparable result to DPS at epoch 1000. From table 3.3, DPS had the lowest error with the a very small variance ($5.07E-05$) which means all errors of the simulations are closer to the average.

For TS1, SAILA initially had the highest generalization error but decreased to a low level of error (see figure 3.3(b)). SLA initially had the lowest average error, which can be explained by the fact that SLA used more patterns initially than the other algorithms (refer to figure 3.7(b)). Although SLA and DPS had small errors, DPS had the smallest variance and thus DPS achieved the smallest error. AL had the largest error after 2000 epochs with a large confidence interval.

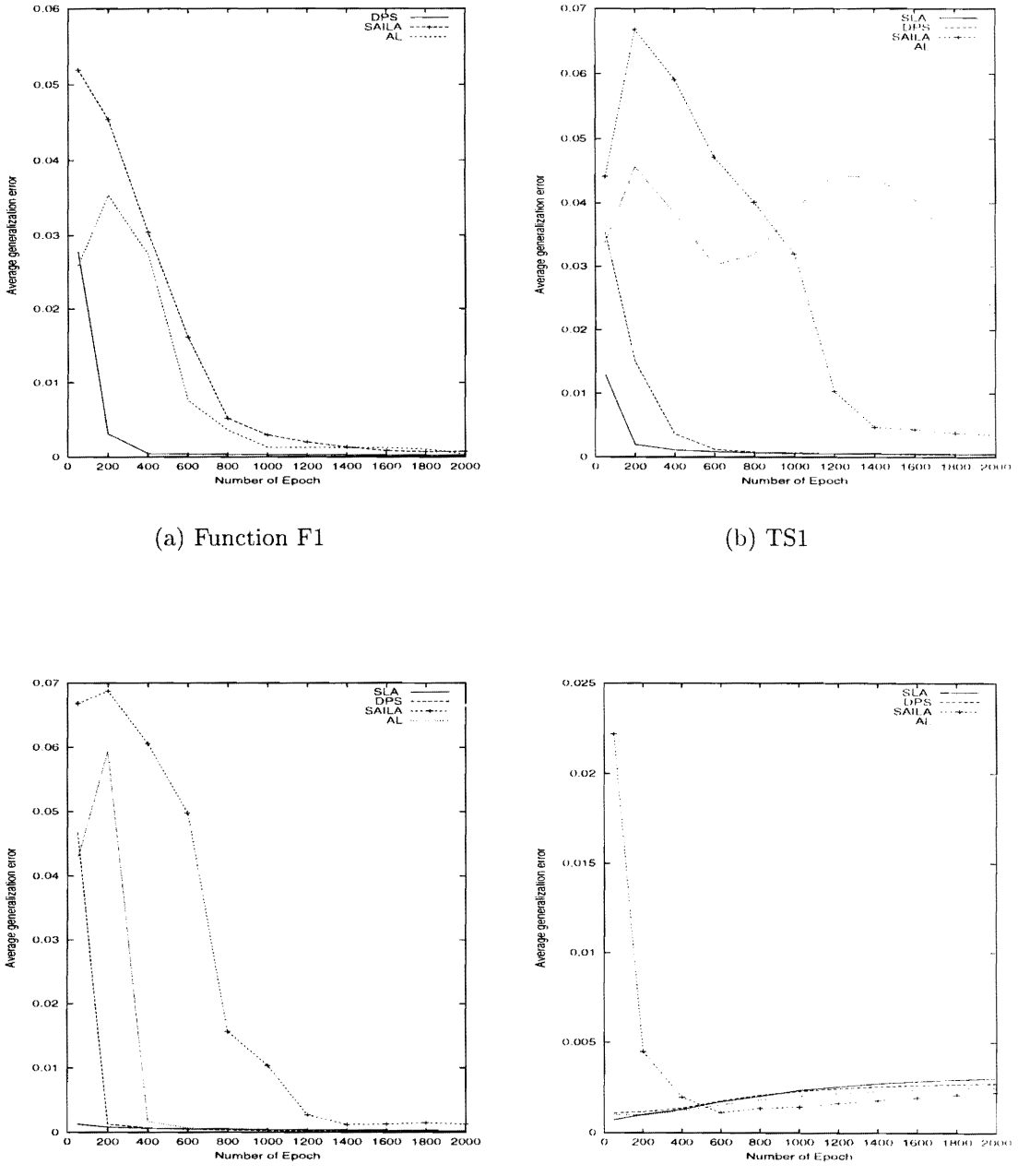
For TS2, DPS, AL and SLA achieved a very low average error before epoch 500.

Function	Röbel	Zhang	Selective Learning	Sensitivity Analysis
TS1				
Training Error	0.00036 ± 0.00017	0.02172 ± 0.04186	0.00045 ± 0.00035	0.00346 ± 0.00712
Generalization	0.00039 ± 0.0002	0.02241 ± 0.04191	0.00047 ± 0.00040	0.0035 ± 0.00737
Used Patterns	485.43 ± 234.79	4.73 ± 0.92	270.93 ± 3.48	571.67 ± 88.91
TS2				
Training Error	0.00014 ± 0.00011	0.00023 ± 0.00021	0.00029 ± 0.00038	0.00126 ± 0.00163
Generalization	0.00012 ± $0.25E-05$	0.00022 ± 0.00019	0.00029 ± 0.00037	0.00129 ± 0.00169
Used Patterns	411.77 ± 215.87	174.63 ± 61.48	272.57 ± 7.61	522.57 ± 173.37
TS3				
Training Error	0.00039 ± 0.00086	0.00044 ± 0.00091	0.00050 ± 0.00085	0.00068 ± 0.00146
Generalization	0.00275 ± 0.00155	0.00253 ± 0.00133	0.00302 ± 0.00138	0.00225 ± 0.00174
Used Patterns	180 ± 0	180 ± 0	78.17 ± 1.53	180 ± 0

Table 3.2: Comparison results over 2000 epochs for times series problems

Function	Röbel	Zhang	Selective Learning	Sensitivity Analysis
F1				
Training Error	0.000226 ± 5.07$E-05$	0.000412 ± 0.000366		0.000791 ± 0.001852
Generalization	0.000221 ± 5.2$E-05$	0.000392 ± 0.000347		0.000764 ± 0.001624
Used Patterns	320.2 ± 167.6698	82.8 ± 32.37935		445.1333 ± 121.476
TS4				
Training Error	0.01141 ± 0.00573	0.19935 ± 0.03093	0.00516 ± 0.00393	0.02828 ± 0.09522
Generalization	0.01077 ± 0.00534	0.19051 ± 0.02169	0.00478 ± 0.00349	0.02739 ± 0.09170
Used Patterns	493.03 ± 193.37	1 ± 0	245.23 ± 7.89	597.03 ± 27.41
TS5				
Training Error	0.00683 ± 0.00468	0.10278 ± 0.08731	0.00155 ± 0.00158	0.00562 ± 0.00768
Generalization	0.00714 ± 0.00489	0.09904 ± 0.08932	0.00158 ± 0.00142	0.00595 ± 0.00842
Used Patterns	103.5 ± 20.14	4.67 ± 1.35	269.13 ± 9.89	584 ± 93.4

Table 3.3: Comparison results over 2000 epochs for problems F1 and times series with noise and outliers



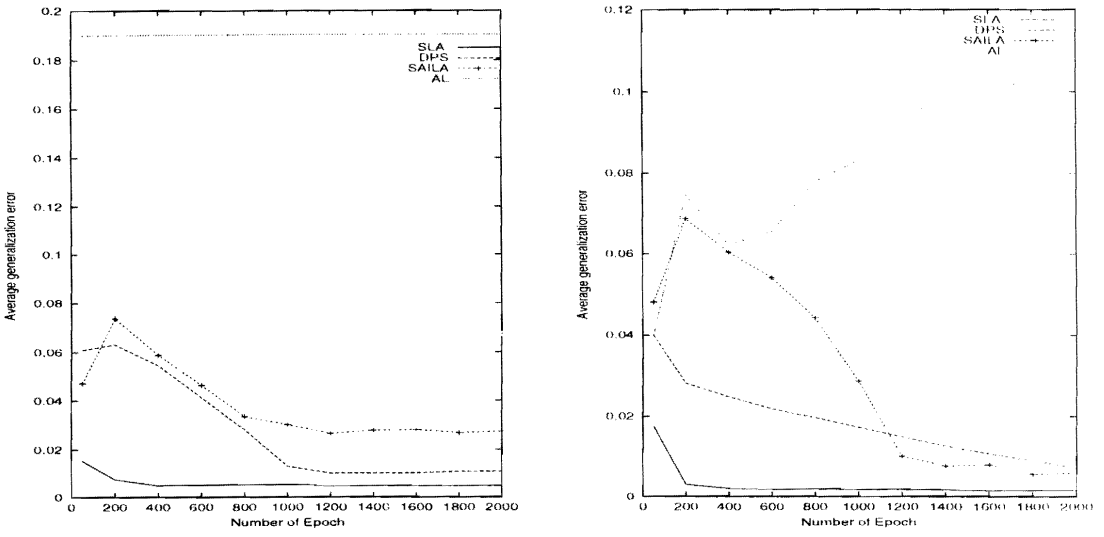
(a) Function F1

(b) TS1

(c) TS2

(d) TS3

Figure 3.3: Average generalization error vs epoch



(a) TS4 (b) TS5

Figure 3.4: Average generalization error vs epoch

SAILA was slower to achieve a comparable low error but SAILA had a low error by the end of training. From the table 3.2, DPS had the smallest error with a very small variance after 2000 epochs, implying that all errors of the simulations are closer to the average.

For TS3, the generalization error for all the algorithms increased as the number of epochs increased except SAILA (see figure 3.3(c)).

From the table 3.3, SLA had the lowest generalization errors for TS4 and TS5. While AL had the largest error for TS4 and TS5. AL did not learn the functions (refer to figure 3.4(a)). AL selected a few patterns for training, thus had little information about the time series to be approximated and therefore AL had a bad generalization.

DPS had the lowest generalization errors for functions with clean data while SLA had the lowest generalization errors for functions with noise and outliers. Although DPS had better generalization with clean functions than SLA, DPS used more patterns than SLA to achieve the low generalization error in all the problems. AL had very large generalization errors for TS1, TS4 and TS5. This bad generalization can be

attributed to the extremely small training set sizes used by AL which is an indication of an inferior subset selection criterion. The subset selection criterion depends on the number of connections in the network. Redundant or irrelevant weights in the network will make the value of the performance level κ very large which can cause the network to train on the current training subset D_T too long without selecting additional patterns. Thus the network selects a few patterns, hence having insufficient information to train the network. On the other hand, too few weights in the network can make κ small. Thus, the network selects patterns more often than are needed for training.

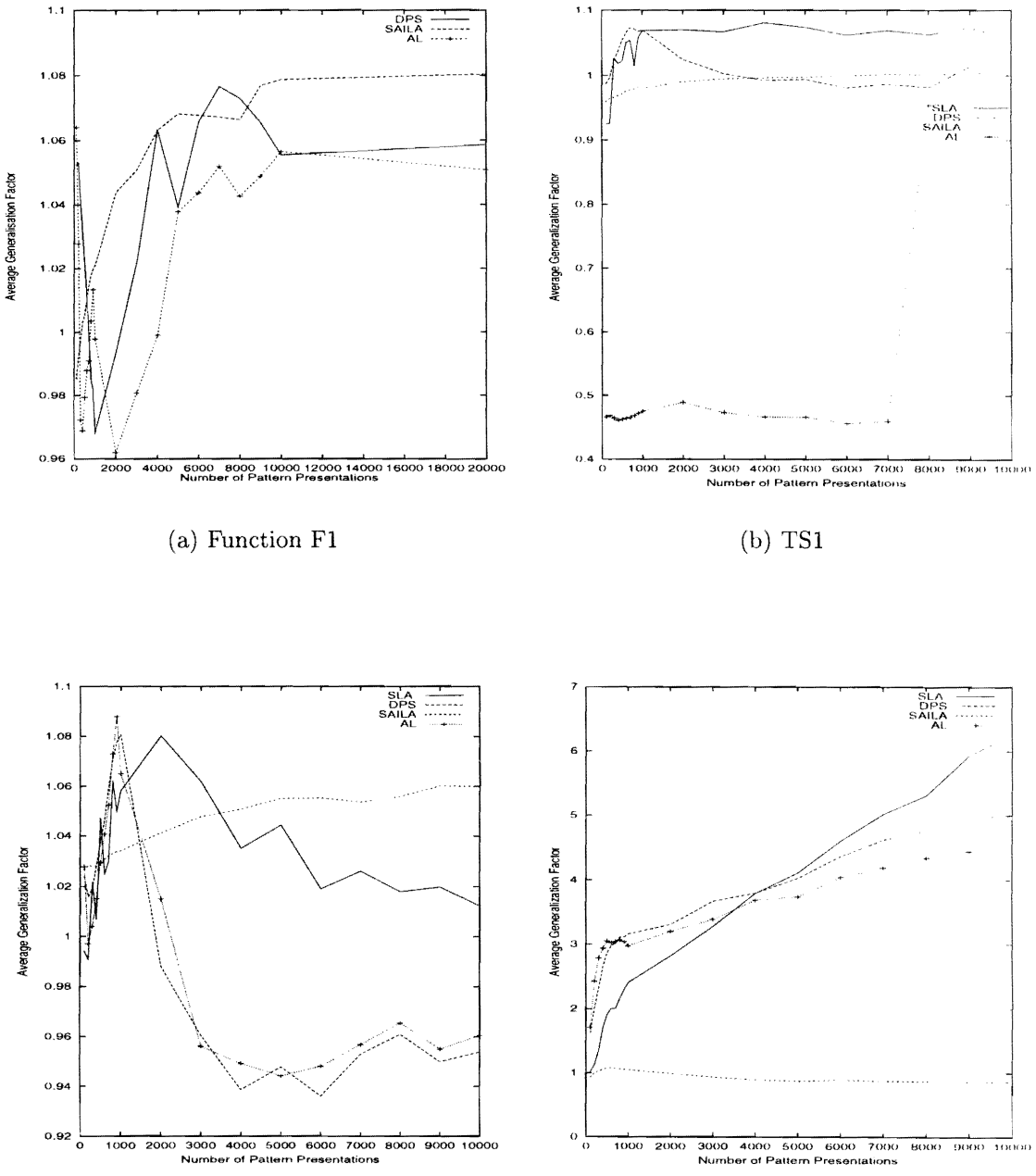
Overfitting effects

The average generalization factor ρ for all the problems were computed over the 30 simulations. Figures 3.5 and 3.6 show the charts for the average generalization factors. The average generalization factors were plotted as function of pattern presentations. A pattern presentation represents one weight update.

TS3 was the only function for which all the algorithms except SAILA, overfitted. SAILA had an average generalization factor of less than one, while the other algorithms had high generalization factors. For the entire training period for TS4, AL had a generalization factor constantly larger than 1, indicating that AL overfitted TS4. For the other functions, the average generalization factor values fluctuated. The flunctuation is due to the overfitting of a training subset until new patterns are selected for training. When new patterns are selected, the overfitting of the training subset is reduced. The average generalization factor for all the algorithms (except TS3) were slightly over one, and indicating a mild case of overfitting.

Computational costs

The computational costs for AL, DPS, SLA and SAILA were computed using equation (3.9) for specified epochs. The costs are plotted as a function of epochs as illustrated in figures (3.9) and (3.10).



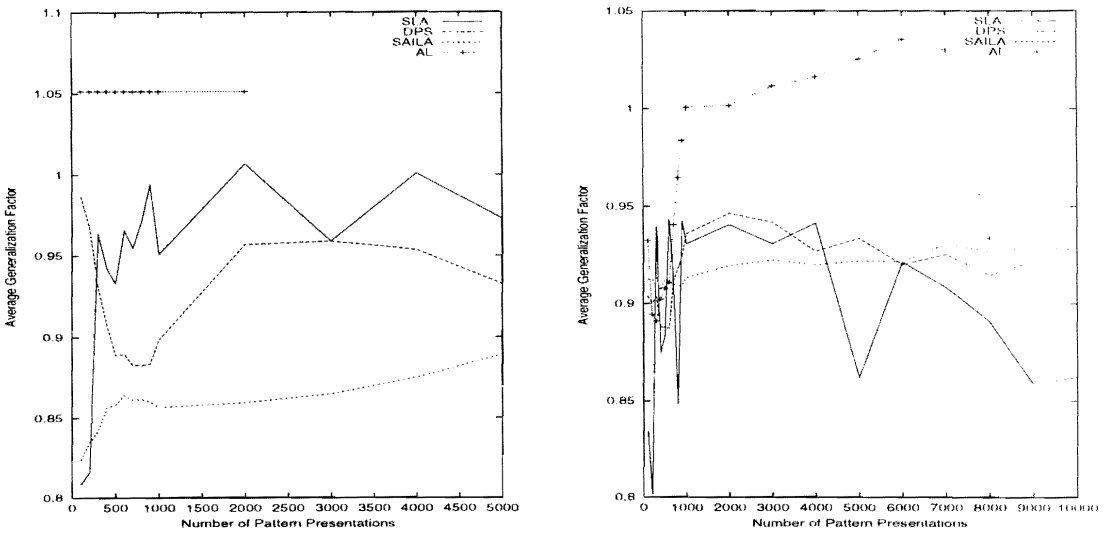
(a) Function F1

(b) TS1

(c) TS2

(d) TS3

Figure 3.5: Average generalization factor vs pattern presentations



(a) TS4 (b) TS5

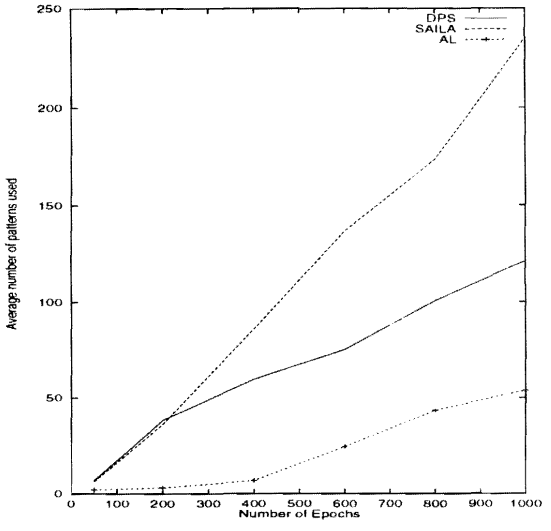
Figure 3.6: Average generalization factor vs pattern presentations

SAILA has the most expensive and AL has the least expensive subset selection criteria. However, AL performed badly (refer to tables 3.2 and 3.3). For all the approximation problems, DPS, AL and SAILA had increasing costs because they are incremental learning algorithms. More patterns were used as training progressed (refer to figures (3.7) and (3.8)).

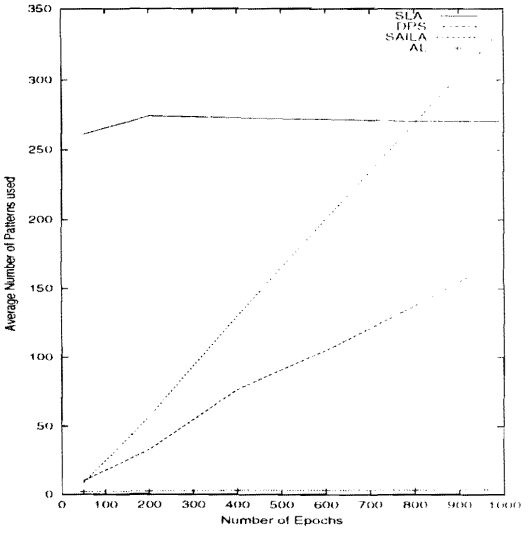
For F1 and TS2, AL had the smallest cost (see figure 3.9(a) and (b)). These small costs can be attributed to the cheap cost of the subset selection criterion as well as the fact that AL used the smallest number of patterns for training.

Despite the fact that AL has the cheapest subset selection criterion and a simple selection criterion, AL had the highest cost for TS3. This is because AL selected all the patterns in D_C within a short training interval (by epoch 400). SLA initially had the highest cost (first 350 epochs). However, at epoch 1000, the cost was half of the other algorithms.

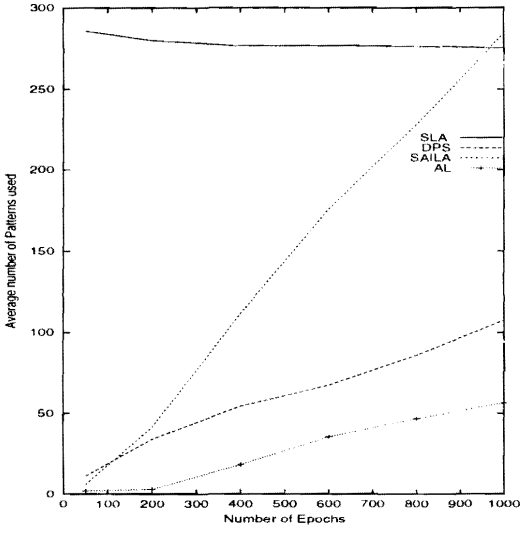
For all the functions approximated, SLA initially had a higher training cost than the other algorithms - almost four times the training cost of other algorithms, because



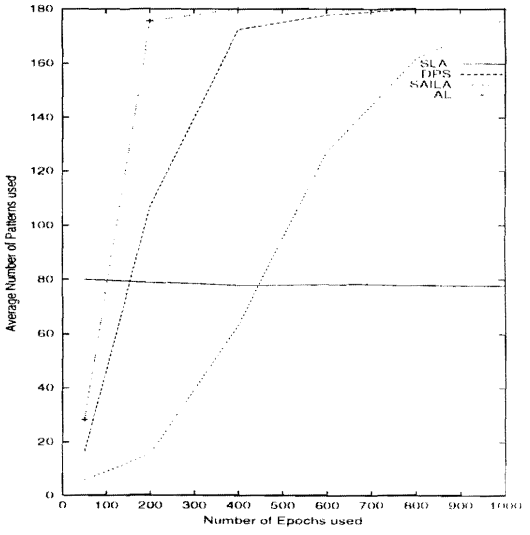
(a) Function F1



(b) TS1

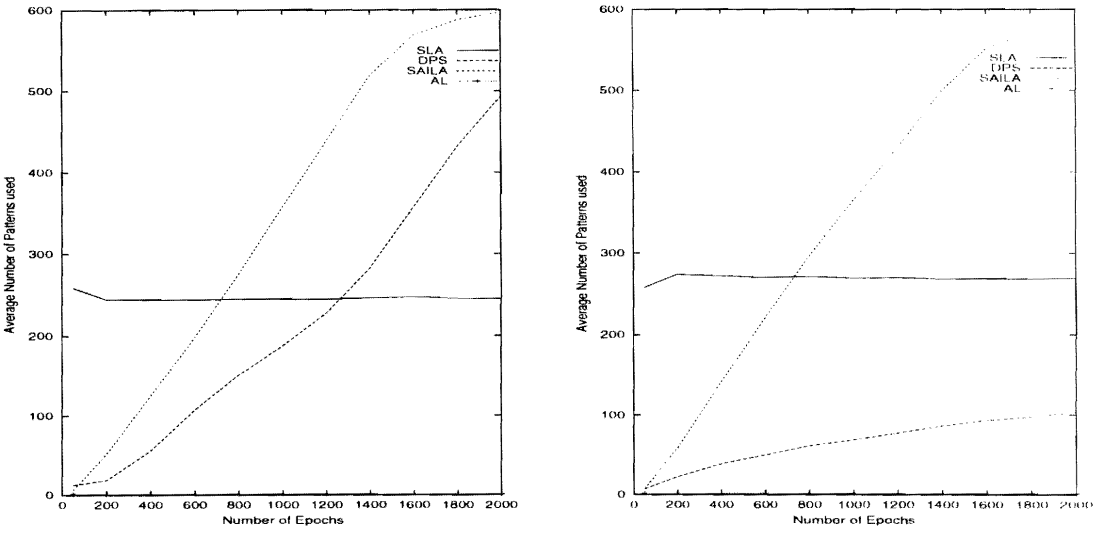


(c) TS2



(d) TS3

Figure 3.7: Average number of patterns used per epoch



(a) TS4 (b) TS5

Figure 3.8: Average number of patterns used per epoch

SLA is a selective approach (see figure (3.9) and (3.10)).

From tables (3.2) and (3.2), SLA used less number of patterns after 2000 epochs than the other algorithms, thus SLA was computationally less expensive.

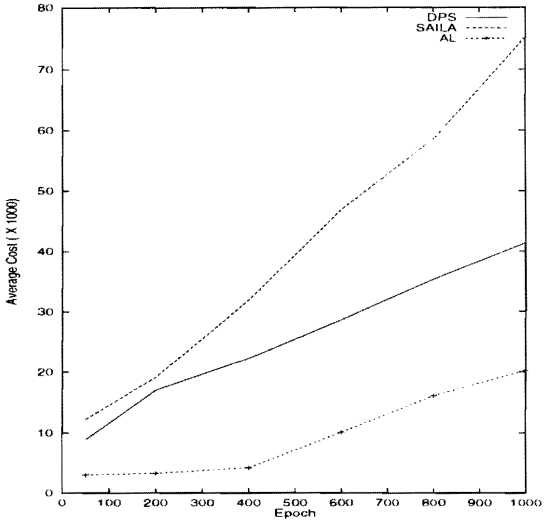
Convergence

The convergence performance of the four active learning algorithms are compared in figures 3.11 and 3.12. These figures plot the percentage of simulations that reached specific generalization errors.

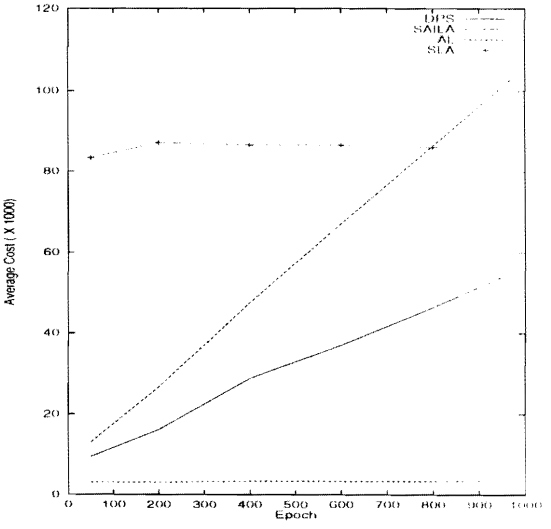
For F1, DPS had the best convergence, all the simulations converged to a very low error of 0.0004. AL also had a good convergence, more than half of the simulations converged to 0.0004 (refer to figure 3.11(a)).

None of AL's simulations converged to the specified error level for TS2. SLA and DPS achieved good convergence for TS2, as more than half of their simulations converged to a low error (refer to figure 3.11(b)).

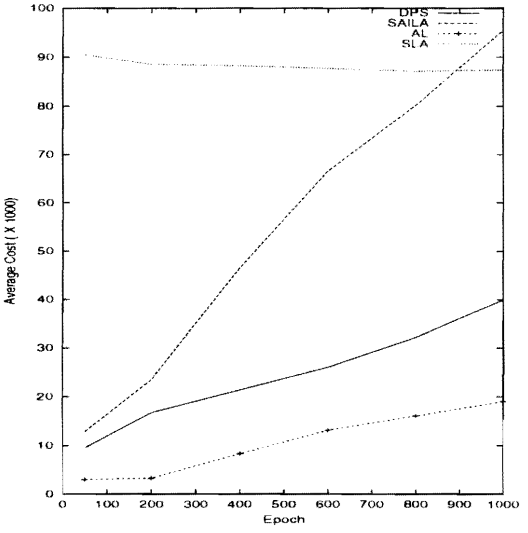
For TS2, DPS had the best generalization, most of all the simulations converged to a



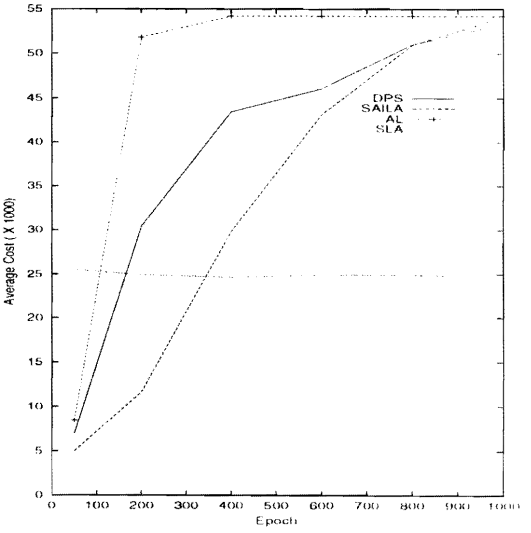
(a) Function F1



(b) TS1

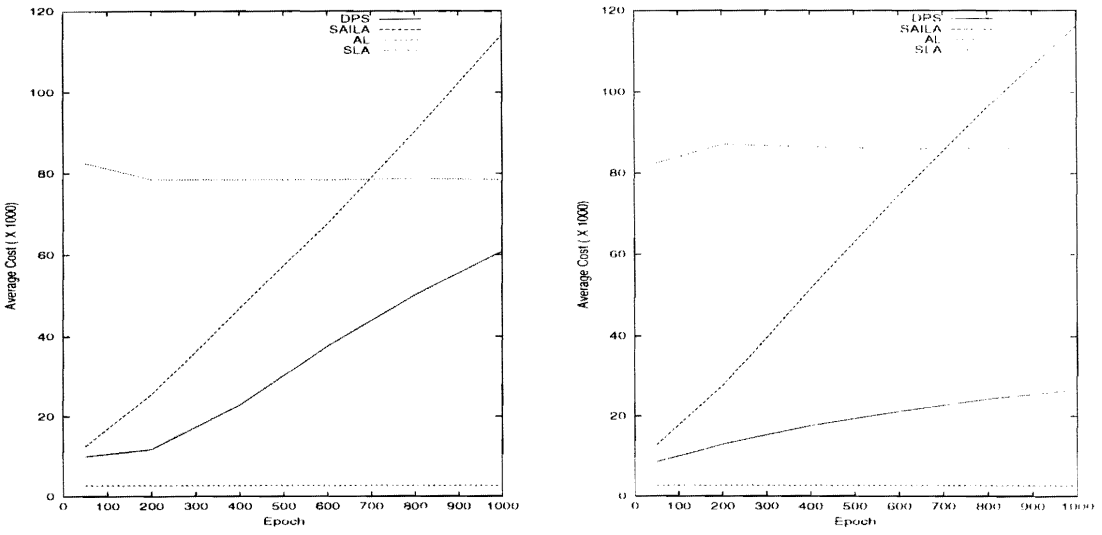


(c) TS2



(d) TS3

Figure 3.9: Average computational cost per epoch



(a) TS4 (b) TS5

Figure 3.10: Average computational cost per epoch

very low error (0.0002). SLA and AL also had good convergence (see figure 3.11(c)).

While the other algorithms had few converged simulations at 0.002, almost half of SAILA's simulations converged to this error (refer 3.11(d)).

AL had bad generalization for TS4 and TS5. None of AL's simulations converged to the specified error levels for TS4 while only a few converged for TS5.

SLA had the best generalization for TS4, with all the simulations converging to a low error of 0.004 (refer to figure 3.12(a)). SAILA had a good convergence with 40% of the simulations converging to this error of 0.004. Only a few of DPS's simulation and none of AL's simulation converged at this point.

SLA also had the best generalization for TS5. Almost all the simulations (74%) converged to a error level of 0.005 while only a few of the other algorithms simulations converged to this error level (see figure 3.12(b)).

SLA had the best convergence for data with outliers and noise. DPS had the best convergence for clean data, although SLA had good convergence for clean data. SAILA

had a good convergence for TS3. For all the sine functions (TS1, TS4 and TS5), AL had bad convergence, none of its simulations converged to the specified error levels. The errors specified for data with outliers and noise were larger than the errors specified for clean data. This is because the performance of all the algorithms were degraded in the presence of noise and outliers.

3.6 Conclusion

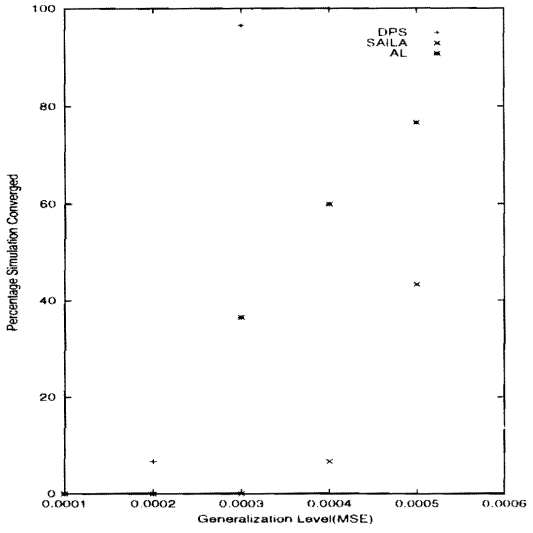
The objectives of the chapter were to present a new learning algorithm (SLA) and also to compare four active learning algorithms with respect to their accuracy, convergence and the complexity on both clean and noisy data as well as overfitting effects for the problems were also examined.

The results presented showed that AL was unstable, producing good results for the henon-map and F1 only. The bad training behavior can be attributed to the extremely small training set sizes used by AL, which is an indication of an inferior subset selection trigger.

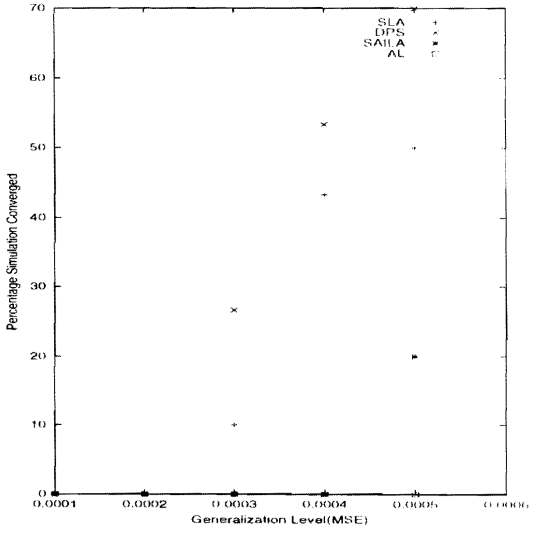
DPS and SLA performed very similar on the clean data, while SLA outperformed all the other algorithms on the noisy and outliers training data. The sensitivity analysis approach (SAILA) performed well under the occurrence of outliers and noisy time series, and very well for the complex function TS3. SAILA performed better than AL in TS1, TS4 and TS5, but worse than SLA and DPS. SAILA is computationally more expensive, requiring larger training subsets than the other algorithms.

As is expected, the performance of the error selection approaches degraded under the occurrence of outliers and noise. The degradation is due to the early selection of outliers, since outliers result in the largest prediction errors.

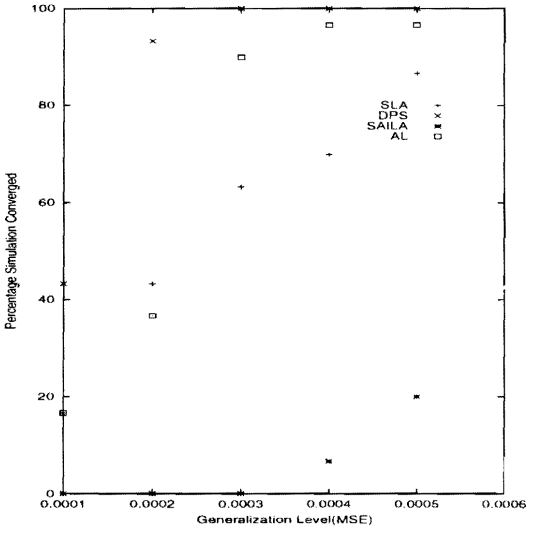
The comparison above showed that SLA had the best generalization performance, and lowest complexity. The selective learning approach (SLA) produced better accuracy



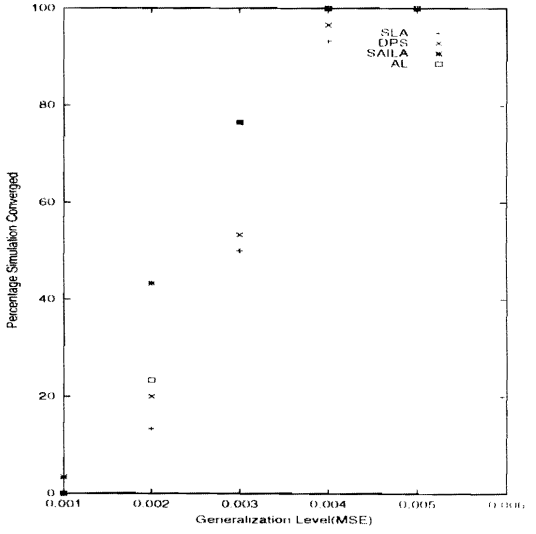
(a) F1



(b) TS1

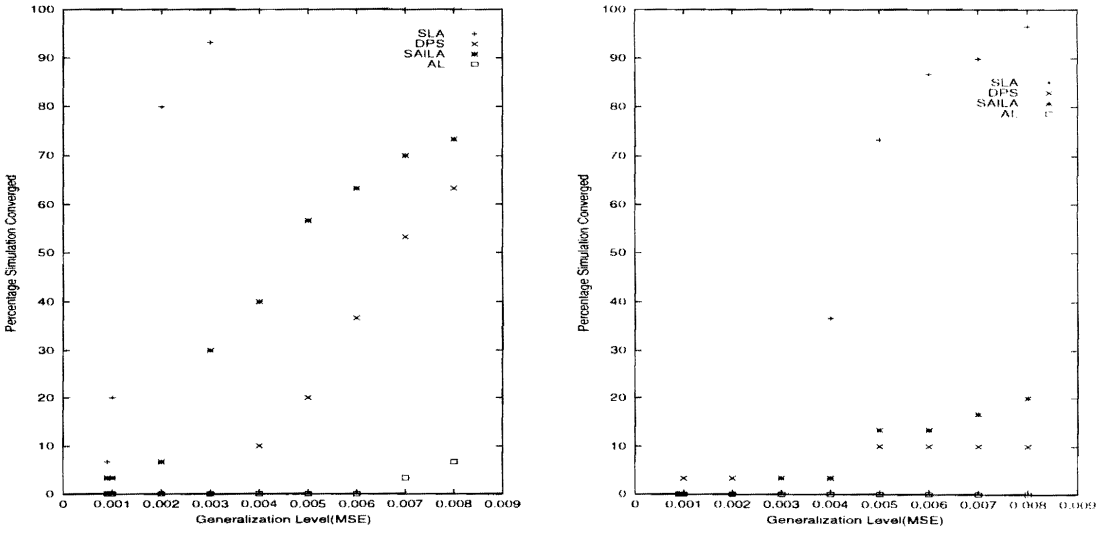


(c) TS2



(d) TS3

Figure 3.11: Percentage simulations that converged



(a) TS4 (b) TS5

Figure 3.12: Percentage simulations that converged

than the other approaches, and showed to be more robust in the occurrence of outliers and noise.