

# Predicting Alarms in Supermarket Refrigeration Systems Using Evolved Neural Networks and Evolved Rulesets

Dan W. Taylor, David W. Corne

University of Reading  
Reading, UK  
dan@jtl.co.uk, d.w.corne@reading.ac.uk

David L. Taylor, Jack Harkness

JTL Refrigeration Systems  
Newbury, UK  
david@jtl.co.uk, jack@jtl.co.uk

**Abstract:** Supermarkets suffer serious financial losses owing to problems with their refrigeration systems. Most refrigeration units have controllers which output "high-temperature" and similar alarms. We describe a system developed to predict alarm volumes from this data in advance, and compare evolved and backpropagation-trained neural networks, and evolved rulesets for this task.

## 1 Introduction

Supermarkets lose many millions of pounds each year due to stock losses from refrigerators in their stores. A typical UK supermarket may contain more than one hundred individual refrigerated cabinets, coldrooms and items of plant machinery which interact as part of a complex integrated refrigeration system within the store. Things very often go wrong with individual units (icing up of components, electrical or mechanical failure, and so forth...) or with components which serve a network of units (coolant tanks, pumps, compressors, and so on).

In the interests of public health, if a refrigerator is found to have failed to maintain an acceptable temperature then all food within that refrigerator must be destroyed. This translates into annual stock losses in the region of several millions of pounds for major supermarket chains.

In almost all supermarkets, refrigerated cabinets and coldrooms (henceforth coldspaces) are attached to a network of piping through which refrigerant is pumped. Heat from the coldspaces is absorbed by evaporating refrigerant which is then compressed and pumped to condensing units outside the store where the heat is expelled. A complex, networked control system is used to co-ordinate this process, allowing a constant temperature to be maintained in coldspaces whilst expending the minimum of energy.

Due to the system's highly integrated nature, a small fault in a single unit or item of machinery can have detrimental effects on the entire store, potentially endangering hundreds of thousands of pounds' worth of stock.

A number of companies exist who specialise in refrigeration control and monitoring equipment in the UK but this paper focuses on the systems designed, manufactured and

maintained by JTL Systems Ltd, based in Newbury, Berkshire, UK.

In order to safeguard stock, the control systems developed by JTL incorporate the facility to generate "alarms" which allow faults to be brought to the attention of store staff (usually by some form of audible and/or visual warning) and are also sent, via the telecommunications network, to a centralised alarm monitoring centre, run by JTL and based in Sunderland in the north east of England. Alarms received at the monitoring centre are recorded in a relational database and are brought to the attention of staff, who are trained to take the appropriate action (usually alerting store staff or the appropriate refrigeration maintenance company).

The monitoring centre, which was opened in September 2000, has proven to be successful in saving JTL's customers money in stock losses but it is accepted that more could be done to prevent stock loss if some means of *predicting* failure existed. In the near future we expect to develop valuable predictive systems which can spot patterns in temperature and pressure logs and predict (and therefore avoid) heavy losses owing to coolant leakage or faulty equipment. For example, all coldspaces must be defrosted every few hours as a matter of necessity. The curve of temperature change with time as a unit defrosts, and then as it recovers from defrost, reveals patterns which are discernible to some expert human eyes as potentially indicative of future problems. However, regular access to these data requires high bandwidth communication between in-store monitoring equipment and a centralised data warehouse which is not yet generally in place (but is underway, and we expect completion late in 2002).

Instead, our present focus is on exploiting a regular stream of data which is already available to JTL. This is alarm data which is automatically dialled in by refrigeration controllers and received at the monitoring centre in Sunderland. It would be very useful to predict the general volume of alarms that are to be expected in the near future, since this impacts on issues of call-centre staffing and overtime, as well as providing useful staffing and management-related information for particular supermarket sites which are (predicted) to suffer heavy alarms in the near future. This

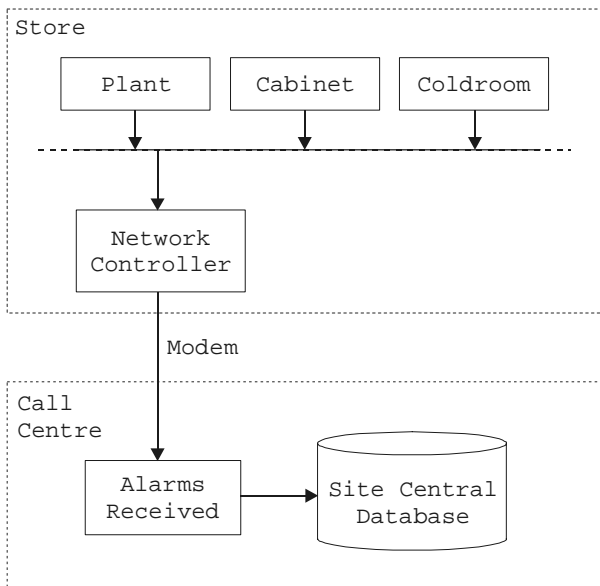
paper hence details the start of our ongoing exploration into using past alarm data to train prediction systems.

In the remainder of the paper, section 2 describes the alarms and the data available in more detail. Section 3 briefly describes the methods we tested on the alarm prediction problem in this paper: namely, neural networks (both evolved and trained by back propagation), and evolved simple rulesets. In section 4 we describe various experiments and their results, finally, section 5 contains a brief concluding discussion.

## 2 Call Centre Data

The data most readily available for study and on-line exploitation is in the form of a large database of *alarms*. In the context of refrigeration control systems, an alarm corresponds to the occurrence of an event which is deemed to be potentially threatening to stock or the operation of the refrigeration system itself.

The electronic controllers in every coldspace and item of plant machinery within a store (henceforth referred to as “control units” or just “units” for brevity) are linked to a serial communications network, supervised by a *Network Controller* which co-ordinates the operation of all control units within the store.



**Figure 1:** Flow of alarms from refrigeration equipment to the Site Central database

When the controller within, for example, a frozen food case senses a set of conditions which are outside its normal operating ranges it contacts the network controller and raises an alarm. On receipt of an alarm the network

controller waits for a specific time period to ensure that the alarm is not just a transient disturbance. After this delay an audible warning is sounded on the shop floor to alert store staff and the network controller uses its in-built modem to contact the alarm monitoring centre. Figure 1 illustrates the way that alarms are generated on site and passed to software at the monitoring centre.

When an alarm is received at the monitoring centre it is immediately stored within a relational database (known as Site Central). The Site Central database holds records corresponding to every alarm received at the monitoring centre as well as information on the monitored stores and the control units therein. An alarm record has these fields:

- Alarm message: a brief textual description of the alarm situation
- Unit and site numbers: used to identify the location and type of the unit which has raised the alarm
- Start and receipt times: the time the alarm was first detected and the time at which it was received at the monitoring centre

The Site Central database also stores information on the status of every unit at every store monitored including its model, the category of the fixture it controls (i.e. “Ice Cream Coldroom”, “Fish Chiller” etc.) and its current settings. This allows an accurate picture of the nature of the fault to be built up.

More than 400 stores are monitored, each with around 100 control units. On average, around 1500 alarms are received per day, though in the summer months, when refrigeration equipment is stretched by hot weather the monitoring centre has been known to receive up to 4000 alarms in a 24 hour period.

### 2.1 Alarm Totals Test Data

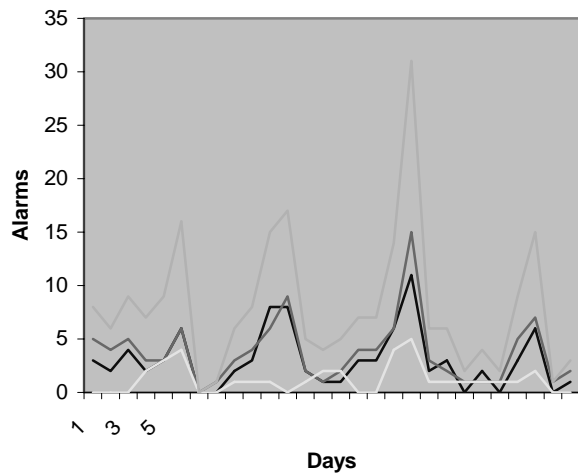
In the experiments to follow we used data from a single site (i.e. a single supermarket), which is known for rather poor management of its refrigeration installations, and we consider only daily totals of particular types of alarm. In particular, data from this site (which we will call site B) for a single day consisted of a 3-tuple:

(cabinet alarms, coldroom alarms, plant alarms)

indicating the total numbers of the respective alarm types received by the call centre on that day. Cabinets are refrigerated cases on the shop floor from which customers select their purchases. Coldrooms are large refrigerated spaces where stock is stored before being transferred to the shop floor and are accessible only to store staff (a typical coldroom may contain £25,000 worth of stock). *Plant* refers to machinery such as compressors and condensers which serve a number of units.

Data are available for this (and other) sites, stretching back almost two years (from when JTL first began to collect it).

In particular, at the time of experimenting we had 672 days' worth of data for site B. Figure 2 shows a representative four-week stretch of this data in full, indicating the daily total (the sum of each of the three types of alarm per day) over time, as well as the individual types. There is a clear cyclic trend which results from changing daily patterns which repeat weekly. For example, Fridays and Saturdays tend to be busiest, with a greater number of shoppers using the store, and hence a greater disturbance to the coldspaces (as shoppers rummage around selecting their purchases). This causes a heightened strain on the refrigeration system and thus a higher probability of alarms being raised.



**Figure 2:** Cabinet, plant and coldroom daily alarm totals (and their sum) over a four week period in 2001.

The clear weekly trend in the data led to some constraints on the setup of our experiments to train predictive systems. It would be of value to be able to predict the next days' total alarms at a particular site. This is valuable for two reasons: first, predictions of high alarm totals can influence management decisions to divert extra effort towards refrigeration system management the next day (both in store and at the alarm monitoring centre). Also, high alarm predictions can lead to the scheduling of pre-emptive maintenance visits by refrigeration engineers to discover and rectify the problems.

### 3 Evolving Neural Networks

This section gives details of how we produced various types of predictive model for the alarm-total prediction problem described in section 2. We compare and contrast three different techniques: neural networks trained by straightforward backpropogation (BP), neural networks

'trained' by using a hybrid of evolutionary search and backpropogation (EB), and simple predictive rulesets trained by evolutionary algorithms (Back, 1996; Fogel, 1995; Goldberg, 1989) (ER). Further details of the techniques are given in a moment, but we must also add here that two kinds of predictive model were trained in the case of BP and EB, and one type in the case of ER. The predictive models were either straightforward regression models (in which the neural network had a single output unit whose value was desired to match an actual number of alarm totals), or categorisation models whereby we tried to predict whether an alarm total would be low, medium, or high. The BP and EB methods were tried on both regression and categorised versions of the problem, while the ER method was suitable only for categorised data.

In the cases of BP and EB, we used straightforward three-layer (one hidden layer) neural networks (Rumelhart et al, 1986; Haykin, 1994). In BP, the networks were trained by backpropogation (Rumelhart et al, 1986), and in method EB, they were trained using a hybrid of evolutionary algorithms and backpropogation which was a simplified version of EP-net (Yao & Liu, 1995; 1996), in which the topology of the network was fixed and we evolved only the weights of the network. By exploring both BP and EB methods we were interested in the trade-off between speed of training and generalisation ability. It is well known that backpropogation trains a neural network relatively fast, however has serious problems with overfitting. Several studies in which neural networks are trained instead using an evolutionary algorithm, however, show advantages in performance despite slower training. Classic studies include Montana & Davis (1989) and Belew et al (1991), while a range of more recent examples include Cho & Cha (1996), Yao & Liu (1995; 1996) and Knowles & Corne (2000). Excellent reviews of this area appear in Yao (1999).

In the case of ER, we use a straightforward evolutionary algorithm to evolve small sets of simple attribute-based rules. The enterprise of using evolutionary algorithms to generate rules or sets of rules is growing in importance and popularity. Freitas (2001) surveys this field, and the approach we use here is related to that used in Fidelis et al (2000), although we evolve simple sets of rules (called the Pittsburgh approach, stemming from the area of Learning Classifier Systems (Holland, 1988)). A ruleset is simply a collection of simple rules which attempt to 'cover' a dataset. For example, if we are trying to predict a category (low, medium, or high) for the total number of alarms that will happen on day 4 given the alarm totals for days 1, 2 and 3, then the following is a simple ruleset containing 4 rules.

- If (day1 = LOW) and (day2 = LOW) THEN day4 = LOW
- If (day1 = MED) THEN day4 = MED
- If (day3 = HIGH) THEN day4 = HIGH
- If (day2 = LOW) THEN day4 = MED

Two important factors about such rulesets are necessary to point out. First, there may be cases not covered – i.e. there

may easily be instances in the test data for which the ruleset provides no prediction (e.g. in this case there are many, including an instance where day1 is LOW and day2 is HIGH). Such a result must therefore be packaged with semantics which indicate what the ruleset's verdict is for such cases. In our case, the ruleset defaults to predicting the category most prevalent in the training set (usually LOW). It is of course valid to have "no prediction" as a default, however we chose this to enable proper comparison with EB and BP (the resulting neural networks also always provide a prediction). Second, there may be conflict within a ruleset. In this case, given a test instance with day1=MED and day3=HIGH, both the second and third rules above will yield a prediction for day4, but these predictions will be in conflict. Rulesets therefore need a conflict resolution mechanism. This can be very sophisticated (e.g. in the XCS classifier system this is done via a number of equations involving several factors of the rules involved including parameters relating to their 'experience' during training (Wilson, 1995; Butz & Wilson, 2000)), but in our case we use a very simple technique whereby each rule has an associated random weight between 0 and 1 (subject to mutation). In case of conflict, the highest-weighted rule wins (ties broken arbitrarily). For further work using more sophisticated evolutionary algorithms for evolving rulesets, see, for example, De Jong et al (1993) and Pei et al (1997).

## 4 Experiments and Results

### 4.1 Data Sets and Other Details

As described in section 2, we used alarms data from a particular supermarket site which we call B, which tends to have management problems with the monitoring and engineering/repair of its refrigeration equipment. The raw data consists of nearly two years' worth of readings of total alarms divided into *cabinet*, *coldroom*, and *plant* alarms. Such a triplet of such readings is available from site B for each of 672 days.

Taking account of the weekly cycle in the refrigeration load placed on these systems (with most people shopping on Friday and Saturday), we divided the data into seven data sets, data0, data1, ... data6, with the following meaning: 0 stands for Saturday, 1 for Sunday, 2 for Monday, and so forth. Dataset data $k$  was aimed at training the predictive model to predict the alarm total (or a category for the alarm total) on day  $k$  given the previous 7 days' broken down totals. Hence, for example, a predictive model trained with data2 would be used in practice only on Sunday evenings to try to predict an alarm total or category for the next day (Monday), given data for each day in the preceding week.

In the cases of BP and EB, the neural network therefore had 21 input units (seven days' worth of cabinet, coldroom and plant totals), either 20, 30, or 50 hidden units (depending on the experiment) and either one or three output units (depending on whether we wished to predict the total

(cabinet alarms, plant alarms and coldroom alarms added together) for the next day, or whether we were predicting a category (LOW, MED or HIGH) for that total. In the regression cases, we class a prediction as 'correct' if it was within 30% of the actual total. In the categorisation cases, a winner-takes-all strategy is used whereby we take the network's prediction to be indicated by which of the three output units (respectively representing LOW, MED and HIGH) has the highest activation.

As noted, the evolutionary algorithm used to evolve neural networks was a simplified version of EP-net which hybridised BP and an EA. Since certain aspects of the way we did the hybridisation are being considered for a patent application by the sponsoring company, we unfortunately cannot disclose here the precise algorithm.

When using ER, the rules in a ruleset had interval-based attributes and made a category prediction. That is, a single rule is simply a conjunction of intervals, one for each of the 21 inputs, followed by a category. An example such rule with just 3 intervals rather than 21 is as follows:

[1—10] [5—7] [0—2] [0—2] [0—2] [1—8] MED

which would be interpreted as "if there were between 1 and 10 cabinet alarms on day 1 AND between 5 and 7 coldroom alarms on day 1 AND ... AND between 1 and 8 plant alarms on day 2, then the total on day 3 will be MED".

A chromosome (individual) in the evolutionary algorithm (Fogel, 1995) used in ER was simply a straightforward representation of a ruleset, consisting of a fixed number of rules, with 21 integer intervals and a category represented for each of them, along with a real-valued weight between 0 and 1 for conflict resolution. A straightforward evolutionary algorithm (Back, 1996; Fogel, 1995; Goldberg, 1989) was used, with steady-state reproduction (Syswerda, 1991), binary tournament selection, and using only a mutation operator which either randomly changed the weight of a randomly chosen rule in a ruleset (using a Gaussian deviate with step size 0.1) or randomly changed an interval or category in a randomly chosen rule in a ruleset.

Finally, despite having nearly two years of totals data available, breaking this down sensibly into models tuned towards predicting alarms for particular days of the week led to 7 relatively small datasets. There are 96 weeks covered by the data, and hence 96 available training patterns. Such a light dataset is unsuitable for standard cross-validation studies, and we therefore used the 0.632 bootstrap method (Efron, 1979). In brief, this involves constructing a training set from a dataset of size  $n$  by taking  $n$  samples with replacement, and using the unchosen samples as the test set (typically the training set will therefore contain around 63.2% of the data). After training, training set error and test set error are then combined and weighted to yield a statistically justified estimate (after repeating several times with different sampled training sets) of the model's performance on unseen data. Test set error

when small datasets are involved is often pessimistic owing to the considerable potential for overfitting. Efron’s bootstrap method yields a result that somewhat alleviates the pessimism without being unduly optimistic.

#### 4.2 Predicting Raw Daily Totals

First we attempted to predict the total alarms for day  $k$  from the previous 7 days’ broken down totals. Hence we trained 7 predictive models, one for each day  $k$ . Ruleset evolution (ER) was not used here. Instead, we compare BP and EB for each of the seven desired predictive models, and we also examine results for different numbers of hidden nodes.

Data6: Predicting alarm totals for Fridays			
Method	#hidden	%-correct	Stdev (%)
BP — EB	20	59.1 – 60.0	1.8—2.2
BP — EB	30	60.0—60.4	1.9—1.9
BP — EB	50	58.1—60.7	2.1—1.7
Data0: Predicting alarm totals for Saturdays			
BP — EB	20	63.2—65.9	2.3 —1.9
BP — EB	30	63.0—65.5	2.5—1.5
BP — EB	50	61.1—65.1	3.0—2.3
Data2: Predicting alarm totals for Mondays			
BP — EB	20	64.2—69.4	3.0—2.1
BP — EB	30	63.2—67.5	2.5—2.1
BP — EB	50	62.8—66.0	3.9—2.2

**Table 1:** results for BP and EP for the Friday, Saturday and Monday predictive models with varying numbers of hidden units. Each row gives results arising from ten runs each of BP and EB on the same dataset for a particular number of hidden units. E.g. for the Saturday data, and using networks with 30 hidden units, BP attained a mean predictive accuracy of 63.0%, with a standard deviation of 2.5%.

Table 1 summarises the results. Results are only given for the three most interesting days (in terms of alarm frequency). Fridays and Saturdays are very busy for refrigeration maintenance in the supermarket sector, and as discussed earlier it is valuable regarding call centre staffing and also supermarket site management to predict how busy it will be beforehand. Mondays tend to be quite and have low volumes of alarms. The number of shoppers on a Monday, and thus the load on the refrigeration system, is much lower than on other days of the week, especially Friday, Saturday and Sunday, which are all very busy.

The only clear trend we can observe is that the EB method consistently outperforms BP. Otherwise there is no discernible pattern in terms of performance against number

of hidden units, or variance in performance. The difficulty of this learning task is clearly beyond the extra leeway provided by extra hidden units in this type of network. Considering various in-house issues, we have settled on an accuracy of 75% or above as indicating a useful result which can be exploited in JTL’s daily call-centre operations and protocols. Disappointingly, but understandably given the low volume of data, this is not quite achieved. The chief reason for this may be that the data used are not sufficient for useful patterns to be determined (e.g. weather data, and/or a larger window size may be more helpful).

Data6: Predicting alarm totals for Fridays			
Method	#hidden/rules	%-correct	Stdev (%)
BP — EB	20	73.6—76.6	2.5—2.9
BP — EB	30	74.1—77.9	3.1—2.5
BP — EB	50	73.7—77.6	2.9—2.6
ER	10	70.3	3.4
ER	20	73.4	3.5
ER	50	75.5	2.3

**Table 2:** results for BP, EP and ER for the Friday predictive model with varying numbers of hidden units for BP and EB and varying numbers of rules in a ruleset for ER. Each row gives results arising from ten runs each of BP and EB (or just of ER) on the same dataset for a particular number of hidden units (or ruleset size).

#### 4.3 Predicting Categorised Alarm Totals

In these experiments, each of BP, EB and ER were tested on their ability to predict categorisations of the alarm total for the day of interest. In each dataset, alarm totals were categorised into LOW, MED and HIGH in such a way that there was a roughly equal number of cases in each category. The results are summarised in Table 2. By reducing the difficulty of the learning task somewhat, the three-category problem leads to better accuracies, and again the results are unsurprising in regard to the trends observed. Now we can note that the 75% target is achieved for Friday predictions using EB; BP approaches this performance, but not quite achieves it. In all this is a useful result, since the high-volume days (Friday and Saturday especially) are precisely those for which we could exploit fairly accurate predictions. For example, with more rules allowed in a ruleset, greater accuracy is generally achieved. With just a small number of rules, accuracy is not poor, but is easily outperformed by BP and EB. However (which is the point), such a small ruleset contains human-understandable cues to the factors which seem to underpin the predictions. Unfortunately the precise rules discovered by the process, some of which are quite instructive and useful, are somewhat commercially sensitive

and we cannot publish them. However we will note that several obvious rules typically appear in evolved rulesets, for example predicting a low alarm total if all alarm figures in the previous few days were below a small threshold.

## 5 Conclusion

We have begun to apply various machine learning techniques to the production of predictive models of supermarket site refrigeration alarms. The essence is to be able to predict today the number of alarms that will be signalled ‘tomorrow’ at a supermarket site, so that action can be taken (if the prediction is of a particularly high value, then that is possibly indicative of some major fault brewing in the refrigeration network, and engineers may be called in advance), or staffing decisions appropriately revised (at both site and call-centre) in the light of the refrigeration systems’ expected demand for attention.

In terms of assessing the machine learning techniques themselves, we generally find concurrence with the emerging expectations that evolved neural networks outperform backpropagation trained networks, and that simple evolved rulesets give lower accuracies on the whole, but provide useful human-understandable rules of thumb.

In terms of the application, the results are slightly disappointing since the accuracies are generally below a useful level from a business viewpoint. However this is not always the case, and this general result is also not particularly surprising given the small amount of training data available, and the clear possibility that additional types of salient data will be necessary to ground the pattern recognition. But since systems are coming in place to regularly collect such further and more extensive data, we feel certain that machine learning will be in place in the near future at JTL’s call centre to predict alarm totals (among other things) at chosen sites, as well as to predict staffing needs at the call centre between 1 and 2 days in advance.

## Acknowledgements

We acknowledge the support of the Teaching Company Directorate (via the DTI and EPSRC) for funding this project. The second author also thanks Evosolve Ltd for partial financial support. Finally, we are most grateful for the positive comments of two referees and for the negative but valid opinion of a third.

## Bibliography

Bartlett, P. and Downs, T. (1990) “Training a neural network with a genetic algorithm.” *Technical Report, Dept. of Electrical Engineering., University of Queensland.*

Back, T. (1996) *Evolutionary Algorithms in Theory and Practice*, Oxford University Press: New York.

Belew, R.K., McInerney, J. and Schraudolph, N.N. (1991) “Evolving networks: using genetic algorithm with connectionist learning.” *Technical Report #CS90-174, Computer Science and Engineering Dept., University of California at San Diego.*

Butz, M.V. and Wilson, S.W. (2000), “An Algorithmic Description of XCS”, *Technical report 2000017*, ILLIGAL Laboratory, University of Michigan, IL, USA.

Davis, L.(ed.) (1991) *Handbook of Genetic Algorithms* Van Nostrand Reinhold.

De Jong, K.A., Spears, W. & Gordon, D.F. (1993) “Using genetic algorithms for concept learning”, *Machine Learning* **13**: 161—188.

Efron, B. (1979) “Bootstrap methods: another look at the jackknife”, *Annals of Statistics* **7**:1—26.

Fogel, D.B. (1995) *Evolutionary Computation: Towards a new philosophy of machine intelligence*, IEEE Press.

Freitas, A.A. (2002) “A Survey of Evolutionary Algorithms for Data Mining and Knowledge Discovery”, in Ghosh, A. and Tsutsui, S. (eds.) *Advances in Evolutionary Computation*, Springer, to appear. Preliminary version available from <http://www.ppgia.pucpr.br/~alex/papers.html>

Goldberg, D.E. (1989) *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley.

Haykin, S. (1994) *Neural Networks: A Comprehensive Foundation*. Prentice-Hall, Inc.

Knowles, J.D. and Corne, D.W. (2000) “Evolving Neural Networks for Cancer Radiotherapy”, in L. Chambers (ed.), *Practical Handbook of Genetic Algorithms: Applications, 2<sup>nd</sup> Edition*. Chapman Hall/CRC Press, pp. 443—488.

Montana, D. and Davis, L. (1989) “Training feedforward neural networks using genetic algorithms.” *Proceedings of the Eleventh International Conference on Artificial Intelligence*, pp 762-767.

Pei, M., Goodman, E.D. & Punch, W.F. (1997) “Pattern discovery from data using genetic algorithms”, *Proc. 1<sup>st</sup> Pacific Asia Conf. on Knowledge Discovery and Data Mining (PAKDD-97)*.

Rumelhart, D.E. , Hinton, G.E. and Williams, R.J. (1986) “Learning representations by back-propagation of errors.” In *Parallel Distributed Processing: Exploration in the Microstructure of Cognition* (D.E. Rumelhart and J.L. McClelland, eds.), Vol. 1, Chapter 8, MIT Press.

Wilson, S.W. (1995) “Classifier fitness based on accuracy”, *Evolutionary Computation* **3**(2):149—175.

Syswerda, G. (1991) “A study of Reproduction in Generational and Steady-State Genetic Algorithms.” *Foundations of Genetic Algorithms*, G. Rawlins, ed. Morgan-Kaufmann, pp 94-101.

Yao, X. and Liu, Y. (1995) “A New Evolutionary System for Evolving Artificial Neural Networks.” *IEEE Transactions on Neural Networks*, **8**(3).

Yao, X and Liu, Y. (1996) “A Population-Based Learning Algorithm Which Learns Both Architectures and Weights of Neural Networks.” *Chinese Journal of Advanced Software Research*, **3**(1).

Yao, X (1999) “Evolving Artificial Neural Networks”, *Proceedings of the IEEE*, **87**(9):1423—1447.