Chapter 8

Dynamic intelligence through Artificial Neural Networks (ANNs)

The concept of Artificial Neural Networks (ANNs) is derived from research into the working of the brain. The models of ANNs are algorithms for cognitive tasks such as learning and optimization (Müller et al., 1995). The use of neural networks in the broader field of operations research is reviewed by Burke and Ignizio (1992).

Potvin and Smith (2003) provide a brief historic review of the first introduction of Artificial Neural Network (ANN) in the late 1940s. After a devastating blow by counter-proving research in the 1960s, the field only reemerged in the early 1980s, with successes in the field of pattern recognition. The first attempt to solve combinatorial optimization problems using neural networks was made by Hopfield and Tank (1985) — solving only small instances of the TSP. Potvin (1993) later solved problems with 200 nodes. One of the first attempts to solve a VRP using an ANN was made by Matsuyama (1991).

The objective of employing ANNs in this thesis is not to compete with existing metaheuristics to *solve* a variation of the VRP. The emphasis is on employing ANNs to predict the best solution algorithm to use, given a new and unknown data set. Also, to learn from experience so that better predictions can be made in future. The candidate proceeds in this chapter to use a learning structure for predicting the best solution algorithm, and the approach can easily be extended to include parametric variations of an algorithm.

8.1 Learning structures

Russell and Norvig (2003) comprehensively address a number of learning structures. The first, and simplest form of learning, is done from observation alone. A *decision tree* is established from known results, and can be employed in making future decisions. The task of learning a function from example inputs and outputs is referred to as *inductive learning*, or *supervised learning*.

8.1.1 Bayesian networks

A *hypothesis*, in the learning context, is a probabilistic theory describing the problem domain. To use an example from the vehicle routing domain, a hypotheses would be to *solve the given problem using the TS*. On the other hand, *data* is the available evidence used to sustain a specific hypothesis. Again, using the vehicle routing example, evidence would be the fact that *customer input data is classified as a* c1 *problem* — geographically clustered with tight time windows.

Acid et al. (2004) used Bayesian networks to predict patient stays in an emergency medical service, and answer related management questions with regards to staff redistribution and possible reinforcement of both staff and equipment. Bayesian learning simply calculates the probability of each hypothesis, given the input data, and makes predictions on that basis (Russell and Norvig, 2003). Predictions are therefore made by using all the probabilities, albeit in a weighted manner, instead of just the single best hypothesis.

8.1.2 Artificial Neural Networks (ANNs)

According to Potvin and Smith (2003) the original objective of artificial neural networks were to provide a fundamentally new and different approach to information processing, especially when an algorithmic procedure for solving the problem was not known. But it is the ability that ANNs have to learn arithmetic or logical functions, due to McCulloch and Pitts (1943), that is of value to this thesis. Russell and Norvig (2003) confirm that ANNs are mostly used for *classification*, in the case of discrete hypotheses, or *regression* for continuous hypotheses. ANNs are powerful at finding nonlinear relationships in data without known structure, but require a lot of data to find such relationships.

This thesis attempts to algorithmically implement the ability of human decision makers to choose an appropriate solution algorithm when solving scheduling problems. The desire to understand the principles on which the human brain work is one of the motivating factors why researchers became interested in neural networks. Another motivation is the wish to build machines that are capable of performing complex, parallel processing of tasks for which the sequentially operating computers are not well-suited. Technology has changed dramatically and consequently made the motivating factors for ANNs a possibility.

8.2 Basic mechanisms of an ANN

Müller et al. (1995) defines a neural network in mathematical terms as a directed graph with the following properties:

- A state variable n_i associated with each node i in the graph. Nodes are also referred to as *neurons*.
- A real-valued weight w_{ik} associated with each edge between nodes i and k. Links are also referred to as *synapses*. The weight w_{ik} is interpreted as the influence that node k's output will have on node i.
- A real-valued bias ν_i associated with each node i, also referred to as the *activation threshold*.
- A transfer function $f_i\left[n_k, w_{ik}, \nu_i, (k \neq 1)\right]$ is defined for each node i. The function determines the state of the node as a function of its bias, the weights of incoming links, as well as the states of the nodes connected to it by the links. The transfer function usually takes the form $f\left(\sum_k w_{ik}n_k - \nu_i\right)$ and is either a discontinuous step function, or a smoothly increasing generalization known as a sigmoidal function. Examples of these shapes are illustrated in Figure 8.1.

The concept of *binary switching* was first proposed in the decision element theory of McCulloch and Pitts (1943). Each one of the decision elements (neurons) $i = 1, \ldots, n$ can only take the output values $n_i \in \{0, 1\}$, where $n_i = 0$ represents the resting state, and $n_i = 1$ the active state of the neuron. The new input state of a neuron i, denoted by $h_i(t)$ at time t, is influenced by all other neurons. The initial combination was originally expressed linearly as

$$h_i(t) = \sum_j w_{ij}n_j(t)$$

using the same notation presented earlier with the exception of a time-dependent $n_j(t)$ denoting the state of neuron j at time t. If $n_i(t)$ denotes the output state of a neuron, the properties of the neural network is governed by the functional relation between $h_i(t)$ and $n_i(t+1)$. The simplest case sees a neuron become active if its input exceeds a certain

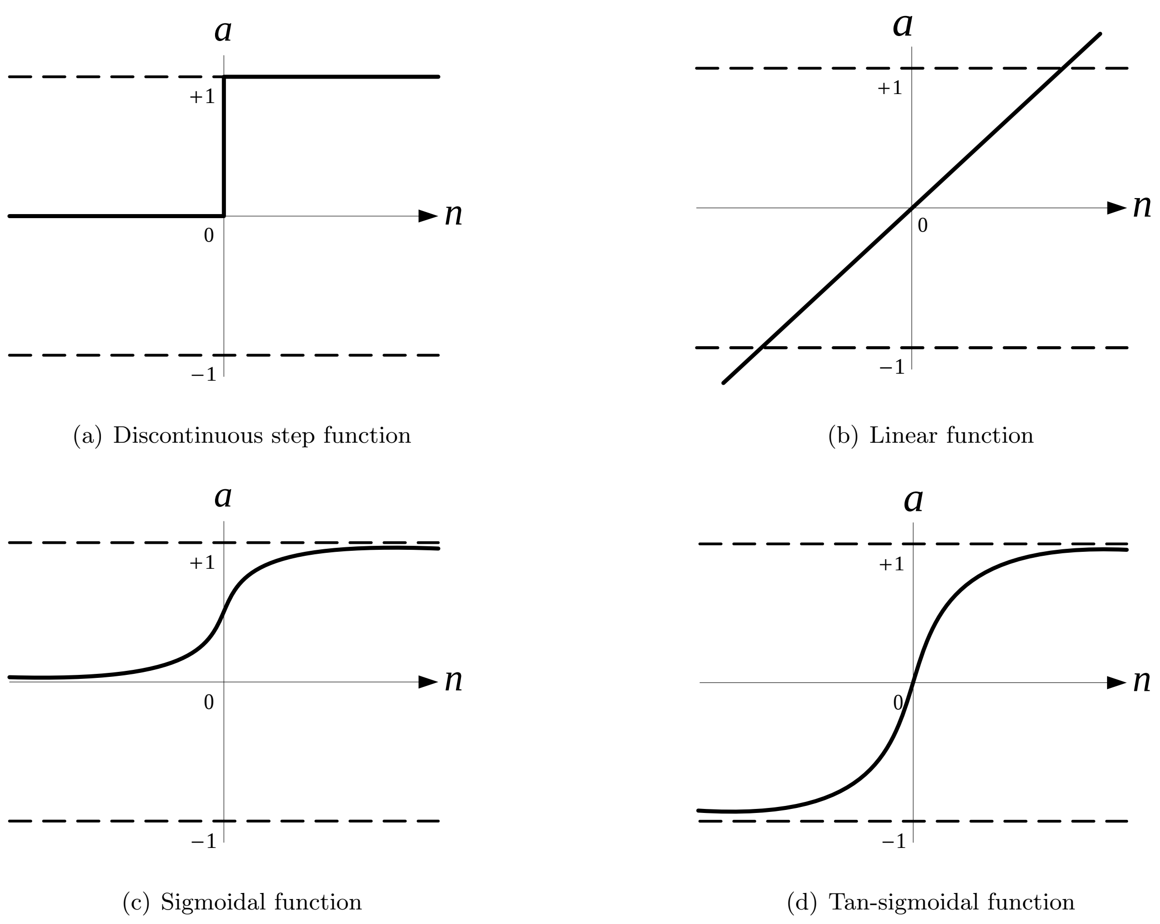

Figure 8.1: Typical transfer function shapes

threshold ν_i which may be unique to each neuron i. The law

$$n_i(t+1) = \theta\left(h_i(t) - \nu_i\right)$$

governs the network with $\theta\left(x\right)$ being the step function indicated in Figure 8.1(a), i.e. $\theta(x < 1) = 0$, and $\theta(x \geq 1) = 1$.

But it was Caianello (1961) that addressed *learning* through an algorithm that allow the determination of the synaptic strengths, w_{ij}, of the neural network. Rosenblatt (1962) introduced the *perceptron*, a layered neural network. The neurons of the output layer receives synaptic signals from the input layer, as illustrated in Figure 8.2.

Whereas the clustering of input data discussed in Chapter 7 was unsupervised, learning is supervised in the ANN suggested here. Dermuth et al. (2005) state that the event of training a neural network can be illustrated using Figure 8.3. The network is trained, or adjusted, until a particular input leads to a specific target output. This iterative process is referred to as *error back-propagation*, and minimizes the Mean Square Error (MSE), where

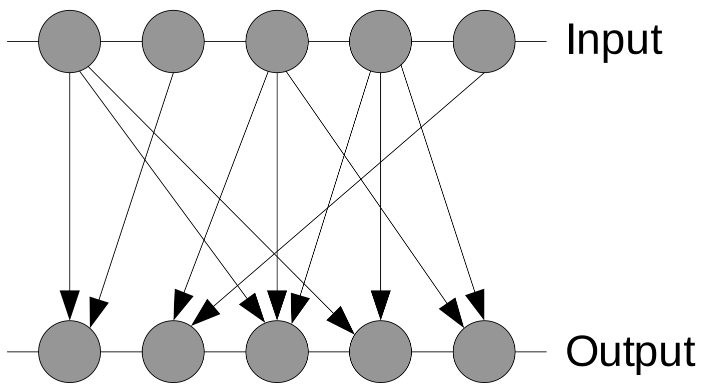

Figure 8.2: Simple layered perceptron (Müller et al., 1995)

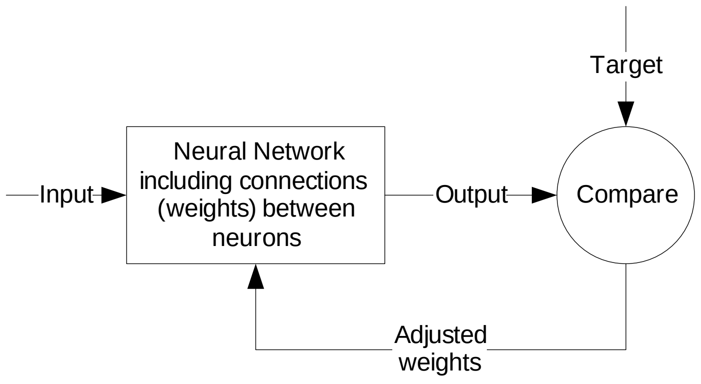
Target
Neural Network including connections (weights) between neurons
Input
Output
Compare
Adjusted weights

Figure 8.3: Training a neural network(Dermuth et al., 2005)

the error is the difference between the actual output and the target output. If t_k denotes the target output of an input element p_k, and a_k denotes the actual output of the network, then, for a training session with K input elements, the MSE is calculated as

$$\text{MSE} = \frac{1}{K}\sum_{k=1}^{K}(t_k - a_k)^2.$$

Karayannis and Venetsanopoulos (1993) divide ANN architectures into three basic categories, of which the first is the category most widely researched, and henceforth adopted in this thesis:

Feed-forward One of the earliest architectures consisting of one or more layers of processing units. Units belonging to neighboring layers are connected by sets of *synaptic weights*. The name *feed-forward* is illustrative of output layers of the network feeds the next layer of units in the network. Networks of this type can be trained to provide a desired response (solution algorithm) to a given input (customer data set).

Feed-back In this type of ANN, the input information defines the initial activity state of a feed-back system.

Self-organizing Humans' ability to use their past experience in order to adapt to unpredictable changes in their environment has lead to self-organizing — an adaption to the environment without the involvement of an external teacher.

8.3 Representation conventions

The notation used to express the ANN is due to Dermuth et al. (2005), and requires a brief introduction.

8.3.1 Network architectures

The architecture of a network describes how many layers a network has, the number of neurons in each layer, each layer's transfer function, and how layers are connected to each other (Dermuth et al., 2005). The challenge is to find the most appropriate architecture for the problem of identifying the most appropriate routing algorithm, and predicting the objective function value. As a general rule, the more neurons in a hidden layer, the more powerful the network.

Consider Figure 8.4 to be a simple neuron i with vector input

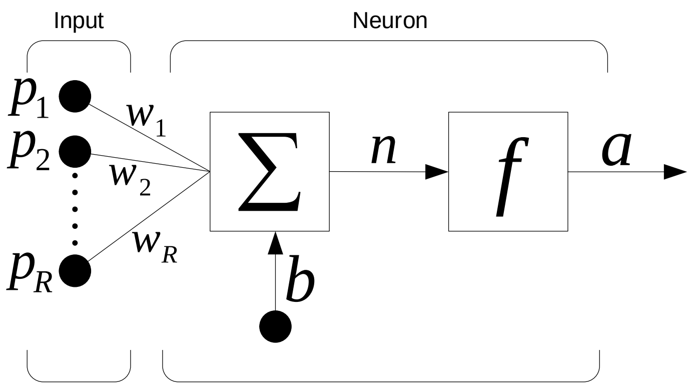

Figure 8.4: A basic neuron

$\boldsymbol{p} = \{p_1, p_2, \ldots, p_j, \ldots, p_R\}$ where R is number of input elements. Associated with each edge between input element j and neuron i is the real-valued weight $\boldsymbol{w} = \{w_{i,1}, w_{i,2}, \ldots, w_{ij}, \ldots, w_{i,R}\}$. The summing junction indicated by $\boxed{\sum}$ has as its input the dot product of the single row matrix $\boldsymbol{w}$ and the input vector $\boldsymbol{p}$, $\boldsymbol{wp}$, as well as the scalar bias, b. In this thesis the bias is initially set to zero, and is adapted during training. The net input to the transfer function, n, is calculated as

$$n_i = \sum_j w_{ij} p_j + b.$$

The transfer function is indicated by $\boxed{f}$, and may be either a discontinuous step function, pure linear function, or the sigmoidal function indicated in Figure 8.1. An abbreviated notation is suggested for a single neuron by Dermuth et al. (2005), and is provided in Figure 8.5.

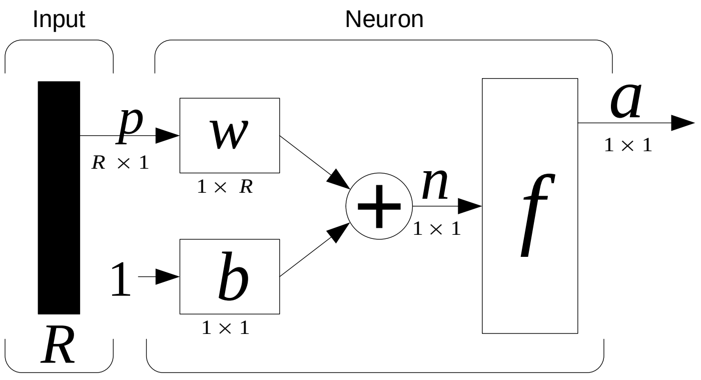

Figure 8.5: A basic neuron using the abbreviated notation

The benefit of the abbreviated notation becomes apparent when s neurons are combined in a single layer. The layer is illustrated in its basic form by Figure 8.6, and in its abbreviated form by Figure 8.7.

A class of problems, referred to as linearly inseparable problems, for example the exclusive-or (XOR) function, cannot be represented by a single perceptron. It was the identification of such inseparable problems that halted neural network research during the 1960s. The introduction of multiple layers of neurons avoids nonrepresentable problems. Hence, the last notational introduction is the existence of multiple layers. To distinguish between the weight matrices, output vectors, etc., for each layer, the layer's number is indicated as a superscript to the variable of interest. Figures 8.8 and 8.9 provides the basic, and abbreviated notation for a three-layered network. The weight $w^{2,1}_{4,3}$ is interpreted as being the weight of the 3rd output neuron from layer 1 to the 4th neuron from layer 2. In Figure 8.9 the layer between the input layer 1 and the output layer 3 is referred to as the *hidden* layer. Russell and Norvig (2003) state the advantage of hidden layers as the ability to enlarge the hypothesis space that the network can represent.

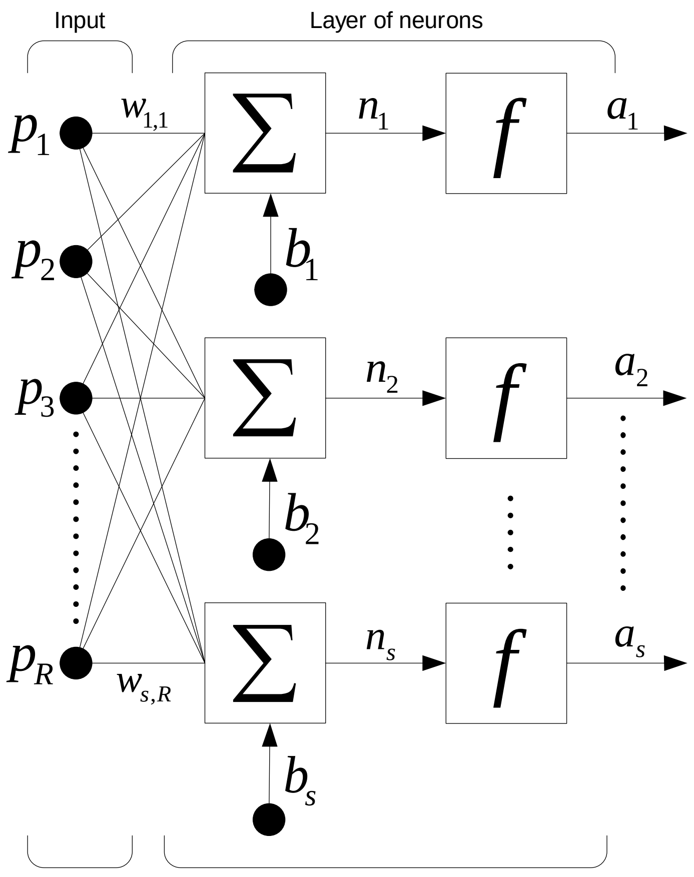

Figure 8.6: A layer of neurons

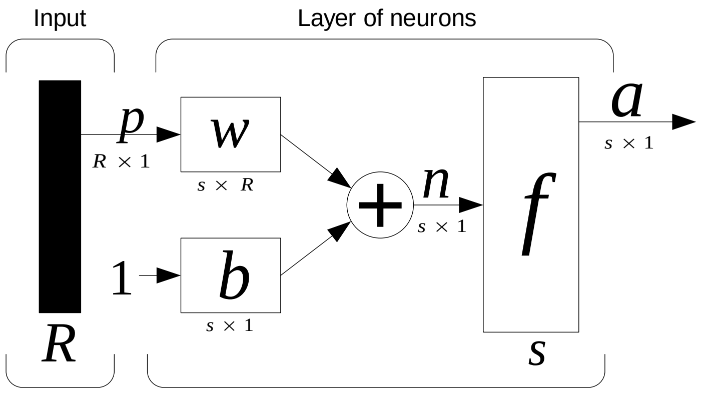

Figure 8.7: A layer of neurons using the abbreviated notation

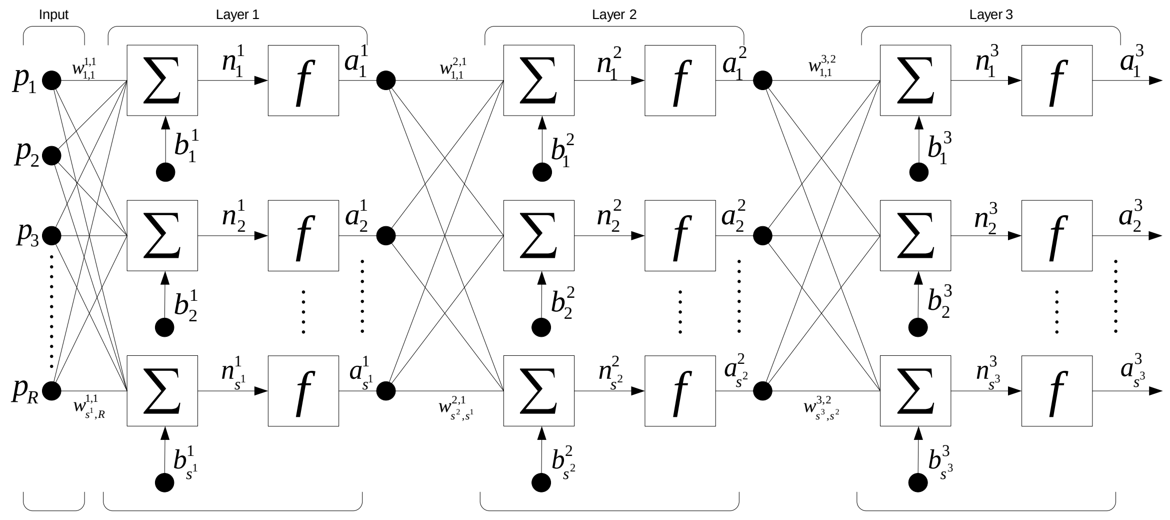

Figure 8.8: A three-layer network

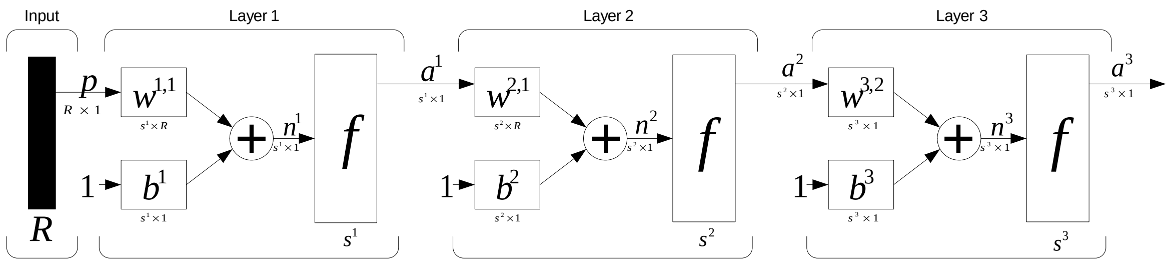

Figure 8.9: A three-layer network using the abbreviated notation

The concept of *back-propagation* determines which hidden layer is responsible for the error. Each hidden node j is believed to be responsible for some fraction of the error, denoted by Δ_i, in each of the output nodes to which it connects. Thus, the Δ_i values are divided according the the strength of the connection between the hidden node and the output node. The error portions are propagated back to provide the Δ_j for the hidden layer. The back-propagation is repeated until the first (earliest) hidden layer is reached.

8.3.2 Data structures

The first type of input vector, denoted by $\boldsymbol{p}$, occurs concurrently without a particular sequence of the elements of the vector, as is the case in this thesis. All inputs (geographical dispersion, time windows, and demand characteristics) occur concurrently, and need not be presented to the network in a specific order. On the other hand, a network for sequentially input vectors contains delays to ensure that input vectors are received in a specific order by the network.

8.4 Proposed network structure

The proposed network is illustrated in its abbreviated form in Figure 8.10. The network

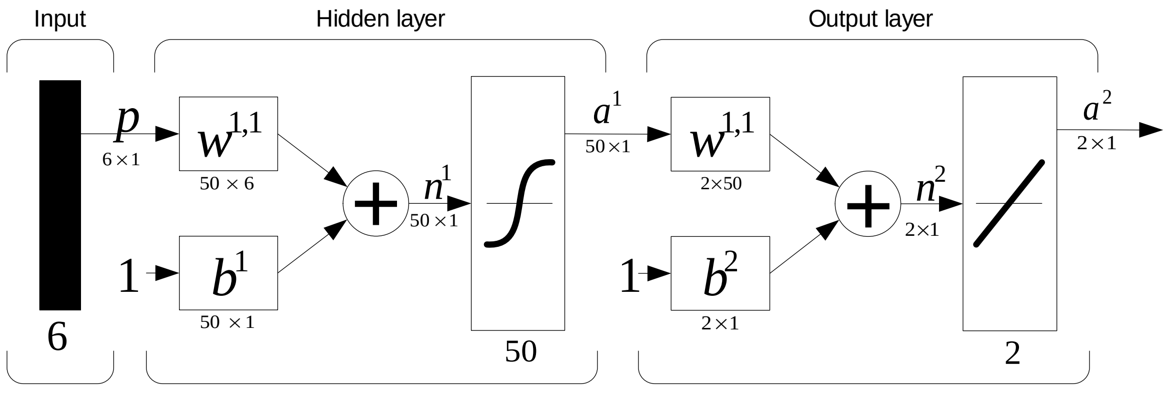

Figure 8.10: Proposed network in abbreviated form

consists of an input layer with six input elements, each depicting a specific characteristic of the problem set:

Number of clusters The optimal number of clusters when using fuzzy c-means clustering and the Xie-Beni index. The index value is the same as the extended Xie-Beni index, V_{XB}^{+}, with a fuzzy factor of $m = 2$.

Validation index The Xie-Beni validation index vlaue, V_{XB}, calculated using (7.7).

Time window width (mean) The arithmetic mean width of customer time windows expressed as a fraction of the time window width of the depot.

Time window width (stdev) The standard deviation of the customer time window widths when expressed as a fraction of the depot's time window width.

Demand (mean) The mean customer demand.

Demand (stdev) The standard deviation in the data set containing all customer demands.

The network boasts a single hidden layer with arbitrary quantity of 50 neurons and a tan-sigmoid transfer function as indicated in 8.1(d) generating outputs in the range $(-1, 1)$. The output layer has two elements and a pure linear transfer function a indicated in Figure 8.1(b). The linear transfer function in the output layer allows outputs to take on any value. The first output element is equal to 1 if TS is the proposed solution algorithm to be used, and 2 if the GA is proposed. The second output element is the predicted objective function value when using the proposed solution algorithm. The actual training set is provided in Appendix F.

8.5 Training the neural network

One of two training styles can be used. In *batch* training, weights and biases are only updated once all inputs have been presented to the network. In this thesis all training data is presented to the network before weights are adjusted. The second *incremental* training style sees weights and biases of the network updated each time an input is presented to the network.

The convergence of the error during training is illustrated with Figure 8.11. As the

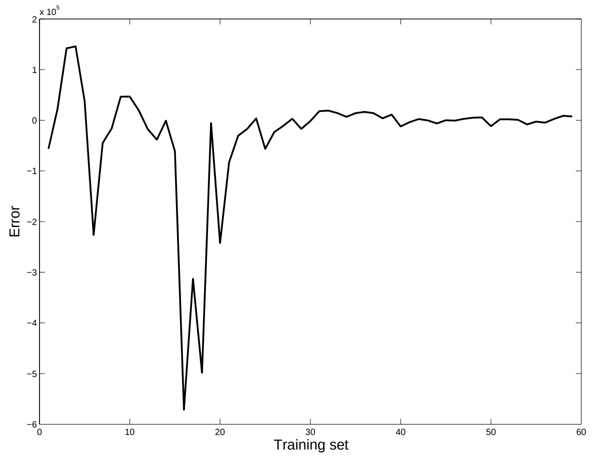

Figure 8.11: Convergence of the objective function error towards zero

training progresses, the network's ability to accurately predict the objective function value increases. It can be seen in the figure that the error converges towards zero.

Properly trained backpropagation networks tend to give reasonable answers when represented with inputs that they have never seen. Typically when an input is presented that is similar to an input on which the network was trained, the output will be similar to the correct output provided during training. This generalization makes it possible to train a neural network on a representative set of input/target pairs and get good results without training the network on *all* possible input/output pairs, as this will not be possible in practice. All possible customer location and demand configurations in the vehicle routing context cannot be solved in reasonable time to train the network.

According to Dermuth et al. (2005), an *epoch* is the time allowed to present the set of training data to the network and the calculation of new weights and biases. The regression analysis result after training the network for 500 epochs is indicated in Figure 8.12. Even

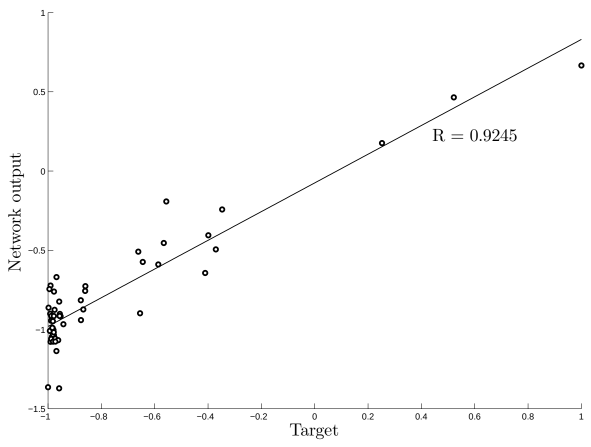

Figure 8.12: Regression results for the trained network

with training data that is not very rich, i.e. only a single instance occurs suggesting the use of GA as a solution algorithm, the network's ability to attain the target is fair, with a correlation coefficient of $R = 0.9245$.

8.6 Integrating the neural network

Neural networks require rich training sets to ensure a well-trained network with accurate prediction rates. A structure for the integrated *intelligent agent* is proposed in Algorithm 8.1. Due to the computational burden being in excess of 1500 seconds, a problem set is only evaluated every ϑ instance. During the evaluation, a problem is solved using both the TS and the GA solution approaches. The problem is analyzed as per the original training of the network. The target for the problem is established as the best of either the TS or GA result. While the target is reported as the problem set's solution, the problem set's analysis, as well as the target, is added to the training set, and the network is retrained on the newly adapted training set.

Algorithm 8.1: The intelligent agent

```
   Input: Network update frequency function, $f(\theta)$
   Input: Problem set, $P$
1  load global iteration number, $\theta$
2  $\vartheta \leftarrow f(\theta) = \max\left\{1, \left\lfloor \ln\left(\frac{\theta}{10}\right)\right\rfloor\right\}$
3  if $\frac{\theta}{\vartheta} = \left\lfloor \frac{\theta}{\vartheta} \right\rfloor$ then
4      $x_1 \leftarrow$ Solve $P$ using TS
5      $x_2 \leftarrow$ Solve $P$ using GA
6      $x^\star \leftarrow \min_{i=\{1,2\}} \{x_i\}$
7      report solution $x^\star$
8      load training set
9      training set $\leftarrow$ training set $\cup \{P \oplus x^\star\}$
10     retrain neural network with updated training set
11     save neural network
12     save training set
13 else
14     load neural network
15     $\hat{x} \leftarrow$ simulated output from neural network with input $P$
16     Report solution $\hat{x}$
17 endif
18 $\theta \leftarrow \theta + 1$
19 save $\theta$
```

The network update frequency, ϑ, is a parameter that is dynamically adjusted as a function of the global iteration number, θ. This ensures that the network is initially retrained after every iteration, and the update frequency is decreased as the network becomes more adapted to the environment. In this thesis the candidate assumes that an intelligent routing agent is implemented in an environment in which customer demand characteristics are fairly stable. A network update frequency function, denoted by $f(\theta)$, is proposed in (8.1).

$$\vartheta = \max\left\{1, \left\lfloor \ln\left(\frac{\theta}{10}\right)\right\rfloor\right\} \tag{8.1}$$

The update frequency, ϑ, is illustrated in Figure 8.13 as a function of the global iteration number, θ.

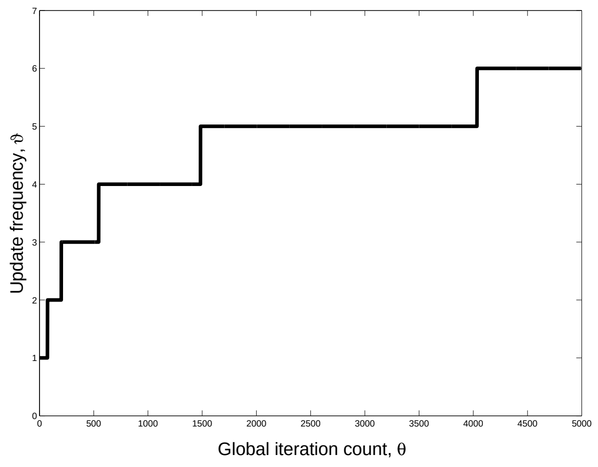

Figure 8.13: Update frequency function

8.7 Conclusion

Planning, according to Russell and Norvig (2003), is the task of coming up with a sequence of actions that will achieve a goal. In the context of this thesis, the goal is to solve any given real life vehicle routing and scheduling problem as best possible. In this chapter, an intelligent agent is proposed in the form of an algorithm that utilizes an Artificial Neural Network (ANN) to predict the best solution algorithm to use to solve a given routing problem. Strategies and results from previous chapters in this thesis, i.e. the Tabu Search (TS) and Genetic Algorithm (GA) metaheuristics, fuzzy c-means clustering, and the current neural network were all integrated within the proposed agent. A conclusive discussion on the value of this contribution, as well as suggested future endeavors are indicated in the next chapter.