

Receptive Field Optimisation and Supervision of a Fuzzy Spiking Neural Network

Abstract

This paper presents a supervised training algorithm that implements fuzzy reasoning on a spiking neural network. Neuron selectivity is facilitated using receptive fields that enable individual neurons to be responsive to certain spike train frequencies and behave in a similar manner as fuzzy membership functions. The connectivity of the hidden and output layers in the fuzzy spiking neural network (FSNN) is representative of a fuzzy rule base. Fuzzy C-Means clustering is utilised to produce clusters that represent the antecedent part of the fuzzy rule base that aid classification of the feature data. Suitable cluster widths are determined using two strategies; subjective thresholding and evolutionary thresholding respectively. The former technique typically results in compact solutions in terms of the number of neurons, and is shown to be particularly suited to small data sets. In the latter technique a pool of cluster candidates are generated using Fuzzy C-Means clustering and then a genetic algorithm is employed to select the most suitable clusters and to specify cluster widths. In both scenarios, the network is supervised but learning only occurs locally as in the biological case. The advantages and disadvantages of the network topology for the Fisher Iris and Wisconsin Breast Cancer benchmark classification tasks are demonstrated and directions of current and future work are discussed.

Keywords: Clustering methods; Receptive Fields; Evolutionary Algorithms; Spiking Neural Network; Supervised Learning

1. Introduction

The history of neural network research is characterised by a progressively greater emphasis paid to biological plausibility. The evolution of neuron modelling with regard to the complexity of computational units can be classified into three distinct generations (Maass, 1997). The third generation of neuron modelling (spiking neurons) is based on the realisation that the precise mechanism by which biological neurons encode and process information is poorly understood. In particular, biological neurons communicate using action potentials also known as spikes or pulses. The spatio-temporal distribution of spikes in biological neurons is believed to 'hold the key' to understanding the brain's neural code (DeWeese, 2006).

There exists a multitude of spiking neuron models that can be employed in spiking neural networks (SNNs). The models range from the computationally efficient on the one hand to the biologically accurate on the other (Izhikevich, 2004); the former are typically of the integrate-and-fire variety and the latter are of the Hodgkin-Huxley type. All the models in this range exploit time as a resource in their computations but vary significantly in the number and kinds of neuro-computational features that they can model (Izhikevich, 2004). The extensive amount and variety of neuron models exist in acknowledgement of the fact that there is a trade-off

between the individual complexity of spiking neurons and computational intensity.

In addition to the variety of neuron models, biological neurons can have two different roles to play in the flow of information within neural circuits. These two roles are excitatory and inhibitory respectively. Excitatory neurons are responsible for relaying information whereas inhibitory neurons locally regulate the activity of excitatory neurons. There is also experimental evidence to suggest that the interaction between these two types of neuron is responsible for synchronisation of neuron firing in the cortex (Börgers & Kopell, 2003). Ongoing physiological experiments continue to illuminate the underlying processes responsible for the complex dynamics of biological neurons.

The degree to which these complex dynamics are modelled in turn limits the size and computational power of SNNs. Therefore, it is imperative to determine which biological features improve computational capability whilst enabling efficient description of neuron dynamics. Ultimately neuro-computing seeks to implement learning in a human fashion. In any kind of algorithm where human expertise is implicit, fuzzy IF-THEN rules can provide a language for describing this expertise (Zadeh, 1965). In this paper, the rationale for the distribution of biologically-inspired computational

elements is prescribed by the implementation of fuzzy IF-THEN rules. This rationale will demonstrate how strictly biological models of neurons, synapses and learning can be assembled in a network topology using fuzzy reasoning. Two benchmark classification datasets are used to demonstrate the capabilities of the topology. The benchmarks are the Fisher Iris and Wisconsin Breast Cancer datasets.

In Section II, unsupervised and supervised learning methods, dynamic synapses and receptive fields (RF) are reviewed. A brief discussion of how fuzzy reasoning can provide a basis for structuring the network topology in such a way that these various elements are suitably utilised follows. Section III introduces a generic network topology and outlines the specific models and algorithms used to implement fuzzy reasoning. Section IV is concerned with pre-processing of benchmark data, Fuzzy C-Means clustering and thresholding to determine cluster size. Experimental results and remarks for the complex non-linear Iris classification problem using a subjective cluster thresholding approach are presented in Section V, and results from the evolutionary optimisation of the thresholding technique for the Wisconsin Breast Cancer dataset are presented in section VI. A discussion of generalisation and the main contribution of the work are outlined in section VII, and lastly conclusions and future research directions are presented in Section VIII.

2. Review

In this section, unsupervised and supervised learning methods, dynamic synapses and RFs are reviewed. Modelling synapses is an essential aspect of accurate representation of real neurons, and one of the key mechanisms to reproducing the plethora of neuro-computational features in SNNs. Learning in all generations of neural networks involves the changing of synaptic weights in the network in order for the network to 'learn' some input-output mapping. From a biologically plausible point-of-view synaptic modification in spiking neurons should be based on the temporal relationship between pre and post-synaptic neurons, in accordance with Hebbian principles (Hebb, 1949). In fact, Hebbian learning and its ability to induce long-term potentiation (LTP) or depression (LTD) provides the basis for most forms of learning in SNNs. Hebbian learning gains great computational power from the fact that it is a local mechanism for synaptic modification but also suffers from global stability problems as a consequence (Abbott & Nelson, 2000).

2.1. Unsupervised Learning

There are several learning algorithms that can be used to evoke LTP or LTD of synaptic weights. Spike-timing dependent plasticity (STDP) is arguably the most biologically plausible means of inducing LTP and LTD learning since it is a temporal interpretation of Hebb's well-known first generation learning rule (Bi & Poo, 1999; Abbott & Nelson, 2000). In terms of the temporal coding of spikes, it is the order of individual pre and post-synaptic spikes that determines whether the weight is increased or decreased using STDP. Stability of STDP can be ensured by placing limits in the strengths of individual synapses and a multiplicative form of the rule introduces an adaptive aspect to learning, resulting in progressively smaller weight updates as learning progresses.

Bienenstock, Cooper and Munro's model (BCM) (Bienenstock et al., 1982) compares correlated pre and post-synaptic firing rates to a threshold in order to decide whether to induce LTP or LTD. The threshold slides as a function of the post-synaptic firing rate in order to stabilise the model. Despite criticism for its lack of biological basis (Abbott & Nelson, 2000), BCM has been demonstrated to be related to STDP (Izhikevich, 2003). In particular, by restricting the number of pairings of pre and post-synaptic spikes included in the STDP rule, the BCM rule can be emulated using STDP.

BCM and STDP are of course unsupervised learning algorithms, and as such they do not obviously lend themselves to applications requiring a specific goal definition, since this requires supervised learning.

2.2. Supervised Learning

There are several methodologies to date for implementing supervised learning in SNNs:

- SpikeProp (Gradient Estimation) (Bohte et al., 2002)
- Statistical approach (Pfister et al., 2003)
- Linear algebra formalisms (Carnell & Richardson, 2005)
- Evolutionary Strategy (Beletreche et al., 2003)
- Synfire Chains (Sougne, 2000)
- Supervised Hebbian Learning (Ruf & Schmitt, 1997; Legenstein et al., 2005)
- Remote Supervision (Kasiński & Ponulak, 2005)

For a detailed review see (Kasiński & Ponulak, 2006).

SpikeProp (Bohte et al., 2002) is a gradient decent training algorithm for SNNs that is based on backpropagation. The discontinuous nature of spiking neurons causes problems with gradient descent algorithms, but SpikeProp overcomes this issue by only allowing each neuron to fire once and by training the neurons to fire at a desired time. However, if weight updates leave the neuron in a state such that it will not fire, the algorithm cannot restore the neuron to firing for any new input pattern. Additionally, since each neuron is only allowed to fire once, the algorithm can only be used in a time-to-first-spike coding scheme which means that it cannot learn patterns consisting of multiple spikes.

By employing a probabilistic approach to the Hebbian interaction between pre and post-synaptic firing, it is possible to produce a likelihood that is a smooth function of its parameters (Pfister et al., 2003). The aim of this, of course, is that this allows gradient descent to be applied to the changing of synaptic efficacies. This statistical approach employs STDP-like learning windows and an injected teacher current. Consequently, the method has been described (Kasiński & Ponulak, 2006) as a probabilistic version of Supervised Hebbian learning (Ruf & Schmitt, 1997; Legenstein et al., 2005). Experiments with this approach have been limited to networks consisting of only two spikes, so it is difficult to know how robust the technique would be for larger networks.

Linear algebra formulisms involving definitions of inner product, orthogonality and projection operations for spike time series form the backbone of Carnell and Richardson's work (Carnell & Richardson, 2005). The Gram-Schmidt process is used to find an orthogonal basis for the input time series subspace, and this is then used to find the subspace for the desired output. A batch style iterative process is described that seeks to then minimise the error between target and actual outputs by projecting the error into the input subspace. The Liquid State Machine (LSM) (Maass et al., 2002) is used for the experiments. Successful training is dependent on the variability of input spikes, but since the training requires batch learning the method is unsuitable for online learning (Carnell & Richardson, 2005).

Evolutionary strategies (ES) have been applied as a form of supervision for SNNs (Beletreche et al., 2003). ES differs from genetic algorithms in that they rely solely on the mutation operator. The accuracy of the resulting SNN provides the basis for determining the fitness function and the ES population was shown to produce convergence to an optimal solution. The learning capabilities of the ES were tested with the XOR and Iris benchmark classification problems.

The Spike Response Model was used to model the spiking neurons in a fully connected feed-forward topology. A limitation of this approach is that only the time-to-first-spike is considered by the ES (Beletreche et al., 2003). Additionally, as with all evolutionary algorithms the evolutionary process is very time-consuming and this renders them unsuitable for online learning.

Synfire chains (SFC) (Sougne, 2000) is a feed-forward multi-layered topology (chain) in which each pool of neurons in the chain (subnet) must fire simultaneously to raise the potential of the next pool of neurons enough so that they can fire. STDP and an additional non-Hebbian term form the basis of the learning rule. The learning rule is designed to regulate the firing activity and to ensure that the waves of synchronous firing make it to the output of the network. The ability of the network to learn an input-output mapping is sensitive to the number and diversity of connection delays. As with many existing supervised SNN learning algorithms this technique relies on time-to-first-spike encoding, and as such cannot learn patterns involving multiple spikes.

Supervised Hebbian Learning (SHL) (Ruf & Schmitt, 1997; Legenstein et al., 2005) is arguably the most biologically plausible supervised SNN learning algorithm. SHL simply seeks to ensure that an output neuron fires at the desired time, with the inclusion of a 'teaching' signal. Since the teaching signal comprises of intracellular synaptic currents, supervision may be envisioned as supervision by other neurons. It has been proven that SHL can learn to reproduce the firing patterns of uncorrelated Poisson spike trains. Thus it can learn patterns involving multiple spikes. However, it suffers from the limitation that even after the goal firing pattern has been reached; SHL will continue to change the weights. Thus constraints must be added to the learning rule to ensure stability (Legenstein et al., 2005).

The Remote Supervision Method (ReSuMe) is closely related to SHL but manages to avoid its drawbacks (Kasiński & Ponulak, 2005). The 'remote' aspect comes from the fact that teaching signals are not delivered as currents to the learning neuron (as with SHL). Instead a teaching signal and STDP-like Hebbian correlation are employed to co-determine the changes in synaptic efficacy. Two STDP-like learning windows are used to update synaptic weights. The first window increases the weight whenever there is a temporal correlation between the input and the desired output (teaching signal). The second learning window is an anti-STDP window which decreases the weight based on the correlation between the input and the actual output.

ReSuMe has thus far only been applied to LSM networks and has demonstrated that it can learn patterns involving multiple spikes to a high degree of accuracy (Kasiński & Ponulak, 2005). Irrespective of the particular neuron model used, the learning process converges quickly and in a stable manner.

Of all the supervised learning algorithms discussed, the ReSuMe approach is perhaps the most efficient. However, synaptic efficacy changes on a much shorter time-scale in biological neurons as well as over the longer time-scale of learning. Hence synapses should be modelled as dynamic not static entities.

2.3. Dynamic Synapses

Synaptic efficacy changes over very short-time scales as well as over the longer time-scale of training. The rate at which synaptic efficacy changes is determined by the supply of synaptic resources such as neuro-transmitter and the number of receptor sites. Dynamic synapse models are typically either deterministic (Tsodyks et al., 1998) or probabilistic (del Castillo & Katz, 1954). However, whichever model is preferred, it is important that the modelled magnitude of the post-synaptic response (PSR) changes in response to pre-synaptic activity (Furhmann et al., 2002). Furthermore, biological neurons have synapses that can either facilitate or depress synaptic transmission of spikes (Tsodyks et al., 1998).

Modelling the dynamics of limited synaptic resources makes neurons selective (sensitive) to particular spike frequencies. The filtering effects of dynamic synapses occur because there is a frequency of pre-synaptic spike trains that optimises the post-synaptic output (Thomson, 1995; Nätschläger & Maass, 2001). A likely explanation for this specificity of frequency is that for certain pre-synaptic spike train frequencies the synapse will not run out of resources whereas for another it probably will. Between these two pre-synaptic spike frequencies there will be an optimum state where the post-synaptic spike frequency is maximised. This means that certain neurons and synapses can potentially be targeted by specific frequencies of pre-synaptic spike trains. This phenomenon has been described as 'preferential addressing' (Nätschläger & Maass, 2001).

The dynamic synapse model used in this research has the following kinetic equations (Tsodyks et al., 1998):

$$\frac{dx}{dt} = \frac{z}{\tau_{rec}} - U_{SE} x(t_{sp}) \quad (1)$$

$$\frac{dy}{dt} = -\frac{y}{\tau_{in}} + U_{SE} x(t_{sp}) \quad (2)$$

$$\frac{dz}{dt} = \frac{y}{\tau_{in}} - \frac{z}{\tau_{rec}} \quad (3)$$

The three differential equations describe inactive, active and recovered states respectively. Each pre-synaptic spike arriving at time t_{sp} activates a fraction (U_{SE}) of synaptic resources, which then quickly inactivate with a time constant of a few milliseconds (τ_{in}) and recover slowly with a time constant of τ_{rec} . These equations are used to calculate the post-synaptic current, which is taken to be proportional to the fraction of resources in the active state:

$$I_{syn}(t) = A_{SE} y(t) \quad (4)$$

The post-synaptic membrane potential is calculated using a leaky-integrate and fire (LIF) passive membrane function given by:

$$\tau_{mem} \frac{dv}{dt} = -v + R_{in} I_{syn}(t) \quad (5)$$

where τ_{mem} is the membrane time constant, v is the membrane potential and R_{in} is the input resistance. The choice of parameters in the dynamic synapse model determines the type of synaptic response. The hidden layer neurons in the FSNN are connected to the input layer using facilitating synapses. Facilitating synapses are marginally more complicated to implement than depressing synapses because they involve the inclusion of another differential equation (Tsodyks et al., 1998):

$$\frac{dU_{SE}}{dt} = -\frac{U_{SE}}{\tau_{facil}} + U1(1 - U_{SE})\delta(t - t_{sp}) \quad (6)$$

where τ_{facil} is the facilitation time constant and $U1$ is the initial value of U_{SE} . For facilitating synapses, the equation describes the variable rate of consumption of synaptic resources.

Depressing synapses model the fact that biological neurons often only respond to the first few spikes in a spike train before running out of synaptic resources. These synapses are implemented using the dynamic synapse model (Tsodyks et al., 1998) by replacing Equation 6 with a constant value for U_{SE} . The action potential with these types of synapses is only significantly high in magnitude for a very short interval. Depressing synapses have been described as being coincidence detectors (Pantice et al., 2003) since they often require synchrony of post-synaptic potentials in order to cause firing of a post-synaptic neuron.

Constructing a network of neurons using synapses that operate at different frequency bands is desirable from the perspective of promoting neuron selectivity and richness of information flow. However, it is

particularly difficult to tune dynamic synapse models to operate at specific frequency bands by changing the various model parameters. One way to guarantee that synapses are responsive to certain frequencies is with the use of RFs.

2.4. Receptive Fields

As far back as 1953, experiments with retinal ganglion cells in the frog showed that the cell's response to a spot of light grew as the spot grew until some threshold had been reached (Barlow, 1953). The part of the visual world that can influence the firing of a neuron is referred to as the RF of the neuron (Rieke, 1997). In Barlow's work (Barlow, 1953), it was demonstrated that a spot of light within the centre of the RF produces excitation of the neuron, whereas when the spot of light is larger than the RF or outside the RF inhibition occurs.

The implications for SNNs are that RFs can be used in conjunction with neuron models to promote feature selectivity and hence enhance the 'richness' of information flow. It should be pointed out that RFs in this research behave in a similar manner to membership functions and, unlike the visual and auditory RFs found in the biology, the RFs in this research are being utilised to process purely numerical data.

3. FSNN (Fuzzy SNN) Topology

Biological neuron dynamics are determined by the relationships between spike trains, synaptic resources, post-synaptic currents and membrane potentials. Neuron selectivity can be further strengthened using RFs. The dilemma is in the way in which all these various elements can be combined in a logical way resulting in SNNs that provide insight in turn into the biological neuron's code, and are useful from an engineering perspective. Biological neurons obviously implement a form of human reasoning. Human reasoning is fuzzy in nature and involves a much higher level of knowledge representation (Zadeh, 1965). Fuzzy rules are typically defined in terms of linguistic hedges e.g. low, high, excessive, reduced etc. Taking a cue from fuzzy reasoning, the aim of this paper is to demonstrate how the components necessary to define a fuzzy rule in turn dictate the distribution of the various biologically plausible computational elements in an SNN. Fuzzy IF-THEN rules are of the form:

IF (x_1 is A_1) AND AND (x_N is A_N) THEN (y is Z) (7)

where x_1 to x_N represent the network inputs, A_1 to A_N represent hidden layer RFs and y is the network output. Figure 1 shows the FSNN topology.

Figure 1 Generic FSNN Topology

At first glance it seems like any other fully connected feed-forward topology. However, each layer uses various computational elements to manage the information flow and implement fuzzy reasoning.

The following subsections will outline the components in each of the layers in the three-layer SNN.

3.1. Input Layer

The function of the input neurons is to simply encode feature data into an appropriate frequency range. Spike trains are then generated from the data using a linear encoding scheme. There is always a significant amount of pre-processing of input data that is required in order to convert typically numerical feature data into temporal spike train data. Here the encoding scheme takes the frequency data points and converts them into an inter-spike interval (ISI) which is then used to create linear input spike trains. Thus each data point is encoded to a particular frequency, which is used to generate an input spike train with a constant ISI. It is then the task of the network to alter the flow of information depending on the frequency encoding of the input data i.e. to be frequency selective.

3.2. Hidden Layer

All of the synapses in the FSNN are dynamic. There are facilitating synapses between the input and hidden layers and depressing synapses between the hidden and output layers. Gaussian RFs are placed at every synapse between the input and the hidden neurons.

In this work it should be noted that the Gaussian RFs operate in the frequency domain. Since the input data to the FSNN is encoded linearly as spike trains of particular frequencies, the RFs utilised in this work need to be selective to these differences in frequency. In the biology, RFs usually operate in the spatial domain, and relay spike trains of various frequencies depending on where an input stimulus is presented in relation to the position of the RF. In this work, since data is encoded into frequencies, the RFs are conceptualised as operating in the frequency domain.

The frequency RFs determine where an input frequency f_i is in relation to the central operating frequency of the RF F_O . The weight is then scaled by

an amount φ_{ij} when calculating the PSR. It is straight-forward for the synapse to measure the input frequency of the RF by measuring the ISI of the linear encoded spike trains. This whole process relates to the ‘IF (x_i is A_i)’ part of the fuzzy rule, where x_i is the input and A_i represents the RF. In this way the RF makes the synapse frequency selective in such a way that the numerous parameters in the dynamic synapse model do not have to be carefully tuned. This is aided by the fact that the synaptic weights of the hidden layer neurons are fixed and uniform in such a way that scaling of the weight by φ_{ij} produces the desired variation in firing frequency of the hidden layer neurons. This is further facilitated by keeping the frequency range of encoded spike trains sufficiently small, [10 40] Hz in this work. Although it should be understood that the weights do not need to be carefully tuned, A_{SE} values (see Equation 4) in the vicinity of 900 pA should suffice for most applications.

Employing facilitating synapses for this purpose makes the hidden layer synapses respond like a non-linear filter. A single hidden layer containing facilitating synapses can approximate all filters that can be characterised by the Volterra series (Maass & Sontag, 2000). Like the Taylor series, the Volterra series can be used to approximate the non-linear response of a system to a given input. The Taylor series takes into account the input to the system at a particular time, whereas the Volterra series takes into account the input at all other times. Such filtering capability has been recently reported in the literature. It is thought that the interplay between facilitation and depression is responsible for this capability, with low-pass (Dittman et al., 2000), high-pass (Fortune & Rose, 2001), and band-pass (Abbott & Regehr, 2004) filtering effects all having been observed.

Spiking neurons sum the post-synaptic potentials to calculate the membrane potential. The function of each hidden layer neuron is to impose the remaining part of the antecedent fuzzy IF-THEN rule, namely the conjunctive ‘AND’. This is not straight-forward since simply summing the post-synaptic potentials is tantamount to performing a disjunctive ‘OR’. The collective aim of the RFs connecting to each hidden layer neuron is to only allow spikes to filter through to the output layer when all of the input frequencies presented at each synapse are within the Gaussian RFs. This can be ensured by making the RF excitatory within the RF and inhibitory outside, as with the centre-surround RFs observed in retinal ganglion cells (Glackin et al., 2008a; Glackin et al., 2008b). In terms of biological plausibility, this gating property of RFs has been observed (Tiesinga et al., 2004). The degree of synchrony of inhibitory synaptic connections causes some RFs to act as

gates, transmitting spikes when synchrony is low, and preventing transmission of spikes when synchrony is high (Tiesinga et al., 2004). This has been observed in vitro by injecting currents into rat cortical neurons using a dynamic clamp and modifying the amount of inhibitory jitter. Of course, the gating property itself is unlikely to be as clear cut as traditional ‘AND’ gates, instead it is likely to be fuzzy in itself ranging between ‘OR’ and ‘AND’ on a continuous scale. In this paper, the RFs are designed to be conjunctive ‘AND’ gates. The reason for this is that there is currently no rationale for the use of any other types of gates in terms of clustering and because there are constraints on the number of RFs that can be implemented on the computer. To produce the ‘AND’ gates, the excitatory (positive) part of the Gaussian RF scales the weight in the range [0, 1], and the inhibitory (negative) part of the RF scales the weight in the range [0, (1 - N)], where N is defined as the number of input neurons. In this way, if even one input frequency lies outside the RF, the resultant inhibitory post-synaptic potential will have sufficient magnitude to ‘cancel out’ the post-synaptic potentials from all the other synapses connecting to the same hidden layer neuron. Linear spikes encoded with frequency x_i are routed to the RFs located at synapses w_{ij} . Each synapse w_{ij} contains a centre-surround RF of the kind shown in Figure 2.

Figure 2 The conjunctive ‘AND’ part of the fuzzy rule using excitatory/inhibitory receptive fields

The RFs are characterised by a central excitatory Gaussian with inhibitory surround. The activation of the RF φ_{ij} , scales the weight w_{ij} , according to the position of the input frequency x_i and the operating frequency of the RF f_{ij} . More formally, the activation φ_{ij} of the RF on the synapses w_{ij} is given by:

$$\varphi_{ij}(x_i - f_{ij}) = \begin{cases} e^{-\frac{(x_i - f_{ij})^2}{2\sigma_{ij}^2}} & \text{if } e^{-\frac{(x_i - f_{ij})^2}{2\sigma_{ij}^2}} > 0.01 \\ 1 - N & \text{if } e^{-\frac{(x_i - f_{ij})^2}{2\sigma_{ij}^2}} < 0.01 \end{cases} \quad (8)$$

where x_i is the frequency of the linear input spike train from input i , N is the size of the input layer, and f_{ij} is the operating frequency of the RF. σ_{ij}^2 is the width parameter of the RF located on synapse w_{ij} . The Gaussian RFs are asymptotic at $\varphi_{ij} = 0$, therefore an arbitrary tolerance of 0.01 on φ_{ij} is used to sharply implement the inhibitory part of the RF and prevent calculation of excessively small floating point numbers in MATLAB.

3.3. Output Layer

The aim of the depressing synapses connecting the hidden layer to the output layer is to produce spikes in the output in a regimented stable manner. It is then the task of the ReSuMe supervised learning algorithm (Kasiński & Ponulak, 2005) to associate the hidden layer neurons to the output layer neurons. Figure 3 illustrates the ReSuMe process. The learning windows are similar to STDP (W^d) and anti-STDP (W^{out}). The parameters s^d and s^{out} denote the time delays ($t^d - t^n$) and ($t^{out} - t^n$) respectively.

Figure 3 Illustration of the learning properties of ReSuMe (Remote Supervision). The learning rules are exponential functions that modify the synaptic weights according to the time difference between the desired spike time and input spike time (left plot); and according to the time difference between the output spike time and the input spike time (right plot).

A modified version of the ReSuMe algorithm was implemented using differential equations that enable the algorithm to be executed in a one-pass manner in much the same way that (Song et al., 2000) implemented the additive STDP rule. The equations used are shown in equations 7.

$$s^{out} \frac{dW^{out}}{dt} = -W^{out} \quad \text{and} \quad s^d \frac{dW^d}{dt} = -W^d \quad (7)$$

Thus performing the fuzzy inferencing between the hidden layer (antecedents), and the output layer (consequents).

In summary, the presented FSNN topology provides a rationale for the use of RFs, excitatory and inhibitory neurons, as well as facilitating and depressing synapses. The next section of the paper describes how such a network may be used to solve two complex non-linear benchmark classification problems.

4. Pre-processing the Data

To test the validity of the approach presented in this paper, two well-known benchmark classification datasets were utilised. The two benchmarks are the Fisher Iris (Fisher, 1936) and the Wisconsin Breast Cancer (Asuncion & Newman, 2007) datasets. Arguably the two datasets have been used repeatedly for testing classification capabilities for numerous numerical, statistical and machine learning

algorithms, and the plethora of benchmark results make them invaluable for refining new approaches.

4.1. Fisher Iris Data

The Iris classification problem (Fisher, 1936) is well-known in the field of pattern recognition. The data-set contains 3 classes of 50 types of Iris plant. The 3 species of plant are Iris Setosa, Iris Versicolour, and Iris Virginica. In the interests of clarity the three classes shall be referred to as class 1, class 2, and class 3 respectively. Class 1 is linearly separable from classes 2 and 3. Classes 2 and 3 are not linearly separable and make Iris a complex non-linear classification problem. The lack of sufficient training data, a problem which is exacerbated by partitioning for cross-validation tests for generalisation, makes training and particularly generalisation for Iris problematic. With a training set of 90 data points (2/3 of the data), 5 misclassifications in the training set will result in less than 90% accuracy, and with 60 data points in the testing (the remaining 1/3 of the data), 4 misclassifications results in a sub-95% testing error. In this way, classification problems involving larger data sets are 'more forgiving'.

4.2. Wisconsin Breast Cancer Data

The Wisconsin Breast Cancer Dataset (Asuncion & Newman, 2007) is another well-known benchmark classification problem. The Wisconsin breast cancer dataset contains 699 instances, with 458 benign (65.5%) and 241 (34.5%) malignant cases. The dataset has only 6 samples with missing data and these samples were removed. Each instance is described by 9 features with an integer value in the range 1-10 and a class label.

4.3. Positioning the receptive fields

The task of determining the number, position and spread of RFs (membership functions) is an important step in tuning a fuzzy system. This is because the maximum number of possible rules governing the fuzzy inferencing process is determined by $R = N^M$ where R is the number of rules, M is the number of membership functions, and N is the number of inputs. For the Iris classification task there are 4 inputs (4 features), therefore the number of rules (number of hidden layer neurons) is given by $R = 4^M$. This means that the number of possible rules grows rapidly as the number of membership functions increases. This phenomenon is known as 'rule explosion'. There are many methodologies for optimising RF placement.

Methodologies include statistical techniques and clustering algorithms (Abdelbar et al., 2006). For this dataset, Fuzzy C-Means (FCM) clustering was used to calculate the positions for RF placement that facilitate separation of the input feature data into their respective classes.

Fuzzy clustering is distinct from hard clustering algorithms such as K-means in that a data point may belong to more than one cluster at the same time. FCM clustering was first developed by Dunn in 1973 (Dunn, 1973) and later refined by Bezdek in 1981 (Bezdek, 1981). FCM is better at avoiding local minima than K-means but it is still susceptible to the problem in some cases. The algorithm for FCM is an iterative procedure that aims to minimise the following objective function.

$$J_m = \sum_{i=1}^m \sum_{j=1}^C u_{ij}^m \|x_i - c_j\|^2, (1 \leq m < \infty) \quad (8)$$

where m refers to the degree of fuzziness, and should be a real number greater than 1. It is commonly set to 2 in the absence of experimentation or domain knowledge. u_{ij} is the membership of any data point x_i in the cluster j . c_j is the centre of the cluster j , and $\|\cdot\|$ represents any norm that measures the similarity (distance) between the coordinate x_{ij} and the cluster centre. Through iteration of the algorithm, the membership is updated with the following function, until the termination criteria ε is met.

$$u_{ij} = \frac{1}{\sum_{k=1}^c \left\{ \frac{\|x_i - c_j\|^2}{\|x_i - c_k\|^2} \right\}^{\frac{2}{m-1}}}, \quad c_j = \frac{\sum_{i=1}^N u_{ij}^m x_i}{\sum_{i=1}^N u_{ij}^m} \quad (9)$$

The stopping criteria is given by:

$$\max_{ij} \left\{ |u_{ij}^{(k+1)} - u_{ij}^m| \right\} < \varepsilon, \quad (10)$$

ε is in the interval $[0,1]$, and k are the iteration steps.

The Iris training set was clustered with FCM in a five-dimensional sense (4 features and the class data). One advantage of FCM is that it can cope when clustering the input data with the class data without altering the class information very much in the resulting clusters. This was the case with the Iris data, for the most part resulting in clusters associated with a particular class. Of course, since FCM is a fuzzy clustering technique, data samples are allowed to belong to multiple classes with varying degrees of membership. Careful selection of appropriate cluster widths can ensure, where possible, that hidden layer clusters are associated with a single class.

MATLAB's FCM algorithm was used to perform the clustering.

4.4. Thresholding

The FCM program returns the cluster positions and a fuzzy membership matrix U for all the data samples in the training set. By setting appropriate thresholds on the fuzzy memberships to each cluster, it is possible to determine which data samples should be within each cluster and which should be excluded. Care should be taken with the thresholding to ensure that each data sample belongs to at least one cluster. Otherwise the networks' hidden layer will ignore the data sample. Figure 4 shows an example of thresholding the fuzzy membership matrix U obtained using MATLAB's FCM clustering algorithm.

Figure 4 Intuitive Thresholding Technique

The figure shows U on the left and the thresholds on the clusters on the right. The membership values are obtained by calculating the Euclidean distance between the data points (1 to 9 in this simple example) and the clusters (1 to 6). The Euclidean distances then are scaled by the sum of Euclidean distances in each column so that the total membership is equal to 1. Of course Euclidean distance is not the only distance measure that can be used to determine the distance between a cluster and a datapoint. Several other distance measures were investigated including (Manhattan and Mahalanobis distances to mention but a few) but subjectively Euclidean was preferred for its simplicity and ease of thresholding. Once the thresholding has determined which data samples should belong to each cluster (shaded in grey in Figure 4), the variances for each feature can be calculated from the features of each included data sample. These feature variances are then used to specify the widths of each Gaussian RF.

5. Fisher Iris Classification Results (Subjective Thresholding Technique)

The simple example shown in Figure 4 illustrates the thresholding technique for a small number of data points and clusters, but for larger sets optimisation of the thresholds is problematic. The subjective nature of the thresholding technique means that often there is a trade-off between the classification of data points crisply into particular classes and the number of data points included in the classification. Figure 5 shows the hidden layer output after thresholding has been performed for the Iris training dataset.

Figure 5 Hidden Layer Processing of Fisher Iris Training Data

The original ordering of the training data is used for clarity, with each successive 30 second interval corresponding to classes 1, 2 and 3. The order of the data does not affect the processing of the hidden layer dynamic synapses since synaptic resources are re-initialised after processing each data point. The hidden layer neurons have also been put in order of class. As can be seen from Figure 5, there are 3, 3 and 4 clusters (and hence hidden neurons) associated with classes 1, 2 and 3 respectively. The resultant classification of the data is clear to see from the figure, however there are some anomalies. The thresholding technique could not exclusively associate hidden neurons 5 and 6 to class 2 data points. Similarly, hidden neurons 9 and 10 could not be exclusively associated with class 3 data points without the hidden layer excluding some points altogether. In all, there were 9 data points (10% of the training set) that were not uniquely associated with one particular class. This means that 10% of the data in the training set is unresolved in terms of the FSNN associating the data correctly with the class data. It is the task of the ReSuMe supervised training regime to resolve this error.

5.1. Resume Training

The first step in implementing the ReSuMe training was to specify the supervisory spike trains to be used. The ReSuMe algorithm was 'forgiving' in this respect producing good convergence for a wide range of supervisory frequencies. Therefore supervisory spike trains of a 50 Hertz frequency were delivered to the appropriate class output neuron whenever the data sample belonged to that class. The supervisory signal for the other class neurons for the given data sample was of zero frequency. The learning windows were equal and opposite in magnitude producing a maximum possible weight update of 0.05. Each training data sample is deemed correctly classified when it produces the maximum number of output spikes at the correct output class neuron. The training resulted in 10/409 misclassified data samples (97.6% accuracy). Once training was completed, all weights were then fixed and the unseen testing data (274 samples) were presented to the network. During the testing phase 13/274 data points were misclassified (95.3% accuracy), showing good generalisation

6. Wisconsin Breast Cancer Classification Results (Evolutionary Threshold Optimisation)

When producing clusters for the Iris dataset, the class information was included in the clustering procedure as described in section 4.3. For the Wisconsin dataset it was desired to cluster the feature data in ignorance of the class information. Clustering without prior knowledge of class information is problematic because data points may be close together with regards to feature data but differ solely with regards to class information. Therefore any clusters generated may contain data points belonging to both classes. The task of the thresholding technique is complicated by this cluster 'fuzziness'. It was discovered that the subjective thresholding approach was time-consuming with regards to larger datasets. Preliminary experiments with FCM indicated that in order to achieve objective values comparable with the Iris dataset (with data scaled into the same frequency range), more clusters would be required to be generated. The larger number of thresholds accompanying these clusters further complicates the optimisation of the clusters. With MATLAB's FCM algorithm it is required to specify the number of clusters required in advance. However, specifying too many can create 'clusters' that contain only one data-point, obviously these types of clusters could have disastrous consequences for generalisation. It is desirable therefore to use an optimisation technique to determine the required number of clusters, and perform the thresholding for problems that involve larger datasets such as the Wisconsin Breast Cancer dataset.

A Genetic Algorithm (GA) was used to find a good solution to the thresholding problem for the Wisconsin dataset, thus optimising the configuration of cluster sizes. GAs are particularly useful for finding good solutions for problems with high dimensional parameter spaces (Goldberg, 1989; Holland, 1975). In the typical GA considered in this research, a genotype encodes the solution to the problem. The genotype is constructed from the individual cluster thresholds (alleles), and then a fitness function determines the merit of each solution to the problem by assigning a fitness value to each genotype. In this way, by utilising thresholds as the basis for the hidden layer RF configuration, individual chromosomes in the GA are compact and as such well-suited to evolutionary optimisation (Cantu-Paz and Kamath, 2005). The GA developed is of the indirect topological variety (Miller et al., 1989), since chromosomes only contain information about part of the structure of the FSNN.

The following equations define fitness. The membership matrix U returned from the FCM algorithm is first augmented with the threshold vector T (genotype) to give:

$$U = \left[\begin{array}{cccc|c} u_{1,1} & \cdots & u_{1,n} & T_1 \\ & \ddots & & \vdots \\ \vdots & & u_{i,j} & T_i \\ & & & \vdots \\ u_{SZH,1} & \cdots & u_{SZH,n} & T_{SZH} \end{array} \right] \quad (11)$$

where SZH is the size of the hidden layer (number of clusters) and n is the number of data points in the training set. A membership index matrix S is constructed according to the following rules:

$$S_{i,j} = \begin{cases} 1 & (u_{i,j} \geq T_i) \\ 0 & (u_{i,j} < T_i) \end{cases} \quad (12)$$

The elements of the thresholded-membership matrix V are then calculated using:

$$v_{i,j} = u_{i,j} S_{i,j} \quad (13)$$

It is known from the training data which elements of $v_{i,j}$ belong to which class, therefore we define $v^{(q)}$ as being an element of V belonging to class q . The fitness of each allele can now be evaluated:

$$F_i = \begin{cases} 1 - \left(\frac{\sum_{j=1}^n v_{i,j}^{(1)}}{\sum_{j=1}^n v_{i,j}^{(1)} + \sum_{j=1}^n v_{i,j}^{(2)}} \right) & \left(\sum_{j=1}^n v_{i,j}^{(1)} > \sum_{j=1}^n v_{i,j}^{(2)} \right) \\ 1 - \left(\frac{\sum_{j=1}^n v_{i,j}^{(2)}}{\sum_{j=1}^n v_{i,j}^{(1)} + \sum_{j=1}^n v_{i,j}^{(2)}} \right) & \left(\sum_{j=1}^n v_{i,j}^{(1)} < \sum_{j=1}^n v_{i,j}^{(2)} \right) \\ 1 & \left(\sum_{j=1}^n v_{i,j}^{(1)} = \sum_{j=1}^n v_{i,j}^{(2)} \right) \\ 0 & \left(\sum_{j=1}^n v_{i,j}^{(1)} = \sum_{j=1}^n v_{i,j}^{(2)} = 0 \right) \end{cases} \quad (14)$$

Finally the total fitness of each genotype is given by:

$$F = \sum_{i=1}^{SZH} F_i \quad (15)$$

According to equation (14) high fitness (a fitness value of zero) is awarded to those thresholds that crisply classify the data, whereas thresholds that classify data points belonging to both classes result in a fitness value close to 1 (low fitness), as do thresholds that do not classify the data at all. In this way the GA does not tell the clusters which class

they belong to, it simply rewards crispness of classification.

As well as this objective, there are several constraints that need to be applied to the GA. The thresholds that are randomly initialised or mutated must be in the range $[0, 1]$ as numbers outside this range have no meaning in terms of fuzzy membership. Additionally, every data point must be processed by at least one cluster to ensure that data is not ignored by the hidden layer of the network. The FCM algorithm is first used to generate 100 cluster candidates. A population of 100 randomly generated solutions is evolved over 100 generations under the control of genetic operators. The principal operators are selection (preferential sampling), cross-over (recombination) and mutation. Figure 6 illustrates the flow of operation of the GA.

Figure 6 GA Flow of Operation

A population of 100 individuals are initialized in the range $[0, 1]$, every individual in the population is evaluated according to equations 11-15. Ranking of the individuals is then performed according to fitness. The two individuals with the highest fitness are termed elite individuals and are retained for at least one more generation. A cross-over fraction of 0.8 is used to determine that the next 78 (0.8 of the population of non-elite individuals) are used for recombination. Recombination or reproduction is performed by randomly selecting multiple cross-over points according to the uniform distribution. Then the remaining 20 individuals are mutated. Constrained mutation is employed by necessity as values for individual thresholds have no meaning outside the range $[0, 1]$, since this is the range of fuzzy membership of data-points to clusters. Thus the next generation is constructed and the algorithm iterates until the target fitness has been reached, or the fitness has not improved by a significant amount for 20 generations, or the maximum number of generations has been reached. Figure 7 shows a plot of the mean fitness against the fittest individual at every generation.

Figure 7 GA Best and Mean fitness for Wisconsin Breast Cancer training data (fold 1)

As can be seen from the figure the mean fitness approaches the best fitness when the GA approximates its best solution. It should be noted however, that since the GA initialises its thresholds randomly that it is important to perform multiple runs.

The hidden layer output resulting from the first fold of training data is shown in Figure 8.

Figure 8 Hidden layer output for Wisconsin Breast Cancer training set (fold 1)

The data is in class order for clarity and the classification from the hidden layer is clearly shown. The hidden layer neurons (plotted vertically in the figure) are active in the first approximately two thirds of the data (class 1) or in the final third. The degree of activity of the hidden layer neurons is shown by the density of the spikes. It is clear to see for each hidden layer neuron which class it is classifying, although there is of course a certain amount of fuzziness when a particular hidden layer neuron produces spikes across the whole range of input data (but favours one class over another in terms of density of spikes). Additionally, as can be seen from the figure the output of only 41 of the original clusters is presented. The reason for this is that the GA sets the cluster sizes of the other 59 clusters to 0; therefore they are removed from the implementation of the clusters in the hidden layer.

6.1. ReSuMe Training

As with the Iris problem, once the hidden layer RFs have been configured it is the task of the ReSuMe supervised learning to associate hidden layer neurons to output class neurons. The difference between the Subjective and Evolutionary approaches is that the latter is by definition an automated technique and as such cross-validation (CV) can be utilised to test for generalisation. Thus five-fold CV was implemented and the results are shown in Table 1.

Table 1 – Wisconsin Results

7. Discussion

Leave-one-out tests have been criticised as a basis for accuracy evaluation, with the conclusion that CV is more reliable (Kohavi, 1995). However, CV tests are also not ideal. Theoretically about 2/3 of results should be within a single standard deviation from the average, and 95% of results should be within two standard deviations, so in a 10-fold CV you should very rarely see results that are better or worse than two standard deviations. The main criticism of generalisation tests is that running them several times produces different solutions. Therefore, the search for the best generalisation estimator is still an open problem (Dietterich, 1998; Nadeau, 1999). Even the best accuracy and variance estimation is not sufficient, since performance cannot be characterised by a single number. It would be much better to provide full Receiver Operator Curves (ROC).

However, while ROC curves are considered to be superior it is useful to at least use generalisation tests that aid comparison to previous work on the benchmark datasets presented in this paper.

The presented FSNN topology is biologically based in a number of ways. As with all SNNs the topology uses the temporal encoding of data into spike trains. The topology contains facilitating excitatory and inhibitory dynamic synapses (hidden layer) that act as non-linear filters. There is even biological evidence to suggest that the gating phenomenon implemented in its purest sense in this work (i.e. as pure AND gates) exists in the biology. In particular, when there is synchrony between inhibitory connections (Tiesinga, 2004). However, it is unclear to what extent the formation of RFs in the biology is evolved or learned; at best the mechanisms governing their use are poorly understood. Certainly there are structural aspects of RFs that are likely evolved (for example retinal ganglion cell and photoreceptors) and there is evidence that RFs can adapt to new stimuli. However, study of this phenomenon is limited by experimental technique and development, and it is fair to say the field has considerable scope for further development.

8. Conclusions and Future Work

This paper presents a detailed review of relevant publications on existing training algorithms and associated biological models. To underpin the FSNN topology, results for the well-known Fisher Iris and Wisconsin Breast Cancer classification problems are presented. The FSNN topology provides a rationale for the assembly of biological components such as excitatory and inhibitory neurons, facilitating and depressing synapses, and RFs. In particular, the major contribution of this paper is how RFs may be configured in terms of excitation and inhibition to implement the conjunctive AND of the antecedent part of a fuzzy rule.

Currently, the work relies on the use of FCM clustering as a strategy for combating the rule-explosion phenomenon of fuzzy logic systems (FLS). Although fuzzy reasoning provides a rationale for the deployment of biological models, there are as yet no biological models for the positioning and size of RFs. Typically in FLSs this is done using purely computational techniques such as gradient-descent algorithms or intuitively. Often membership functions are chosen for the smoothness of control surfaces or for other reasons, it is as much art as science and very problem specific. Perhaps fortunately, neuroscientists are studying RF

formation more intensely than ever before and it seems plausible that a biological alternative to using such computational clustering algorithms may exist. Ideally, it would be preferable for the FSNN to determine the number, placement and spread of fuzzy clusters, without relying on external statistical or clustering techniques in reflection of the biology. For this reason, future work in recognition of the growing interest in biological RFs, will involve the development of dynamic RFs. The ultimate aim is to develop a biologically plausible FSNN that tunes itself using strictly biological principles, and in that regard this work represents a significant contribution.

References

- (1) Abbott, L. & Nelson, S. (2000). Synaptic plasticity: taming the beast. *Nature Neuroscience*, 3, 1178-1183.
- (2) Abbott, L. F. & Regehr, W. G. (2004). Synaptic computation. *Nature*, 431, 7010, 796-803.
- (3) Abdelbar, A., Hassan, D., Tagliarini, G. & Narayan, S. (2006). Receptive field optimization for ensemble encoding. *Neural Computing & Applications*, 15, 1-8.
- (4) Asuncion, A., Newman, D.J. (2007). UCI Machine Learning Repository. <http://www.ics.uci.edu/~mllearn/MLRepository.html>
- (5) Barlow, H. (1953). Summation and inhibition in the frog's retina. *The Journal of Physiology*, 119, 69.
- (6) Belatreche, A., Maguire, L., McGinnity, T. & Wu, (2003). A Method for the supervised training of Spiking Neural Networks. *IEEE Cybernetics Intelligence--Challenges and Advances (CICA)*, 39-44.
- (7) Bengio, Y. & Nadeau, C. (2003). Inference for the generalization error. *Machine Learning*, 52, 239-281
- (8) Bezdek, J. (1981). *Pattern Recognition with fuzzy objective function algorithms*. Kluwer Academic Publishers Norwell, MA, USA.
- (9) Bi, G. & Poo, M. (1999). Distributed synaptic modification in neural networks induced by patterned stimulation. *Nature*, 401, 6755, 792-795.
- (10) Bienenstock, E., Cooper, L. & Munro, P. (1982). Theory for the development of neuron selectivity: orientation specificity and binocular interaction in visual cortex. *Journal of Neuroscience*, 2, 32-48.
- (11) Bohte, S., Kok, J. & La Poutré, H. (2002). Error-backpropagation in temporally encoded networks of spiking neurons. *Neurocomputing*, 48, 17-37.
- (12) Borgers, C. & Kopell, N. (2003). Synchronization in Networks of Excitatory and Inhibitory Neurons with Sparse, Random Connectivity. *Neural Computation*, 15, 509-538.
- (13) Cantu-Paz, E. & Kamath, C. (2005). An empirical comparison of evolutionary algorithms and neural networks for classification problems. *IEEE Transactions on Systems, Man and Cybernetics*, 35, 5, 915-927.
- (14) Carnell, A. & Richardson, D. (2005). Linear algebra for time series of spikes. *Proceedings of the 13th European Symposium on Artificial Neural Networks (ESANN)*. 363-368.
- (15) del Castillo, J. & Katz, B. (1954). Quantal components of the end-plate potential. *The Journal of Physiology, Physiological Soc.*, 124, 560.
- (16) DeWeese, M. & Zador, A. (2006). Neurobiology: efficiency measures. *Nature*. 439, 936-42
- (17) Dietterich, T. (1998). Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, 10, 1895-1923.
- (18) Dittman, J. S., Kreitzer, A. C. & Regehr, W. G. (2000). Interplay between facilitation, depression, and residual Calcium at three presynaptic terminals. *Journal of Neuroscience*, 20, 4, 1374-1385.
- (19) Dunn, J. (1973). A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters. *Cybernetics and Systems*, 3, 32-57.
- (20) Fisher, R. (1936). The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7, 179-188.
- (21) Fortune, F. S., & Rose, G. J. (2001). Short-term synaptic plasticity as a temporal filter. *Trends in Neurosciences*, 24, 7, 381-385.
- (22) Fulmann, G., Segev, I., Markram, H. & Tsodyks, M. (2002). Coding of temporal information by activity-dependent synapses. *Journal of Neurophysiology*, 87, 140-148.
- (23) Glackin, C., McDaid, L., Maguire, L. & Sayers, H. (2008). Implementing Fuzzy Reasoning on a Spiking Neural Network. *Proceedings of the 18th International Conference on Artificial Neural Networks Part II*, 258-267.
- (24) Glackin, C., McDaid, L., Maguire, L. & Sayers, H. (2008). Classification using a fuzzy spiking neural network. *The 2008 UK Workshop on Computational Intelligence (UKCI)*.
- (25) Goldberg, D. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co. Inc., Boston, MA, USA.
- (26) Hebb, D. (1949). *The Organization of Behavior: A Neuropsychological Theory*. Wiley.
- (27) Holland, J. (1975). *Adaptation in natural and artificial systems*. Michigan Press, MI.
- (28) Izhikevich, E. M. & Desai, N. S. (2003). Relating STDP to BCM. *Neural Computation*, 15, 1511-1523.
- (29) Izhikevich, E. (2004). Which model to use for cortical spiking neurons? *IEEE Transactions on Neural Networks*, 15, 1063-1070.
- (30) Kasinski, A. & Ponulak, F. (2005). Experimental Demonstration of Learning Properties of a New Supervised Learning Method for the Spiking Neural Networks. *Proceedings of the 15th International Conference on Artificial Neural Networks*, 3696, 145-153.
- (31) Kasinski, A. & Ponulak, F. (2006). Comparison of supervised learning methods for spike time coding in spiking neural networks. *International Journal of Applied Mathematics and Computer Science*, 16, 101.
- (32) Kohavi, R. (1995). A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection. *International Joint Conference on Artificial Intelligence*, 14, 1137-1145.

- (33) Legenstein, R., Naeger, C. & Maass, W. (2005). What Can a Neuron Learn with Spike-Timing-Dependent Plasticity? *Neural Computation*, 17, 2337-2382.
- (34) Maass, W. (1997). Networks of spiking neurons: The third generation of neural network models. *Neural Networks*, 10, 1659-1671.
- (35) Maass, W. & Sontag, E. (2000). Neural Systems as Nonlinear Filters. *Neural Computation*, 12, 1743-1772.
- (36) Maass, W., Natschlagler, T. & Markram, H. (2002). Real-Time Computing Without Stable States: A New Framework for Neural Computation Based on Perturbations. *Neural Computation*, 14, 2531-2560.
- (37) Miller, G., Todd, P., & Hegde, S. (1989). Designing neural networks using genetic algorithms. *Proceedings of the third international conference on Genetic Algorithms*, 379-384.
- (38) Natschlagler, T. & Maass, W. (2001). Finding the Key to a Synapse. *Advances in Neural Information Processing Systems*, 138-144.
- (39) Pantic, L., Torres, J. & Kappen, H. (2003). Coincidence detection with dynamic synapses. *Network Computation in Neural Systems*, 14, 17-33.
- (40) Pfister, J.; Barber, D. & Gerstner, W. (2003). Optimal Hebbian Learning: A Probabilistic Point of View. *Lecture Notes in Computer Science*, 92-98.
- (41) Rieke, F. (1997). *Spikes: Exploring the Neural Code*. MIT Press.
- (42) Ruf, B. & Schmitt, M. (1997). Learning Temporally Encoded Patterns in Networks of Spiking Neurons *Neural Processing Letters*, 5, 9-18.
- (43) Song, S., Miller, K. & Abbott, L. (2000). Competitive Hebbian learning through spike-timing-dependent synaptic plasticity. *Nature Neuroscience*, 3, 919-926.
- (44) Sougné, J. (2000). A Learning Algorithm for Spine Chains. *Connectionist Models of Learning, Development and Evolution: Proceedings of the Sixth Neural Computation and Psychology Workshop*, 23.
- (45) Thomson, A. (1997). Synaptic interactions in neocortical local circuits: dual intracellular recordings in vitro. *Cerebral Cortex*, 7, 510-522.
- (46) Tiesinga, P., Fellous, J., Salinas, E., José, J. & Sejnowski, T. (2004). Inhibitory synchrony as a mechanism for attentional gain modulation. *Journal of Physiology*, 98, 296-314.
- (47) Tsodyks, M., Pawelzik, K. & Markram, H. (1998). Neural Networks with Dynamic Synapses. *Neural Computation*, 10, 821-835.
- (48) Zadeh, L. (1965). Fuzzy sets. *Information and Control*, 8, 338-353.

PDF Editor with Free Writer and Tools

Table 1 Wisconsin Results

Fold	1	2	3	4	5	Average (%)	Standard Deviation
Number of Clusters	35	40	41	41	43		
Training (%)	98.5	98.5	98.8	98.5	97.5	98.4	0.45
Testing (%)	94.8	95.9	97.8	97.0	97.0	96.5	1.04

PDFill PDF Editor with Free Writer and Tools

Figure 1 Generic FSNN Topology

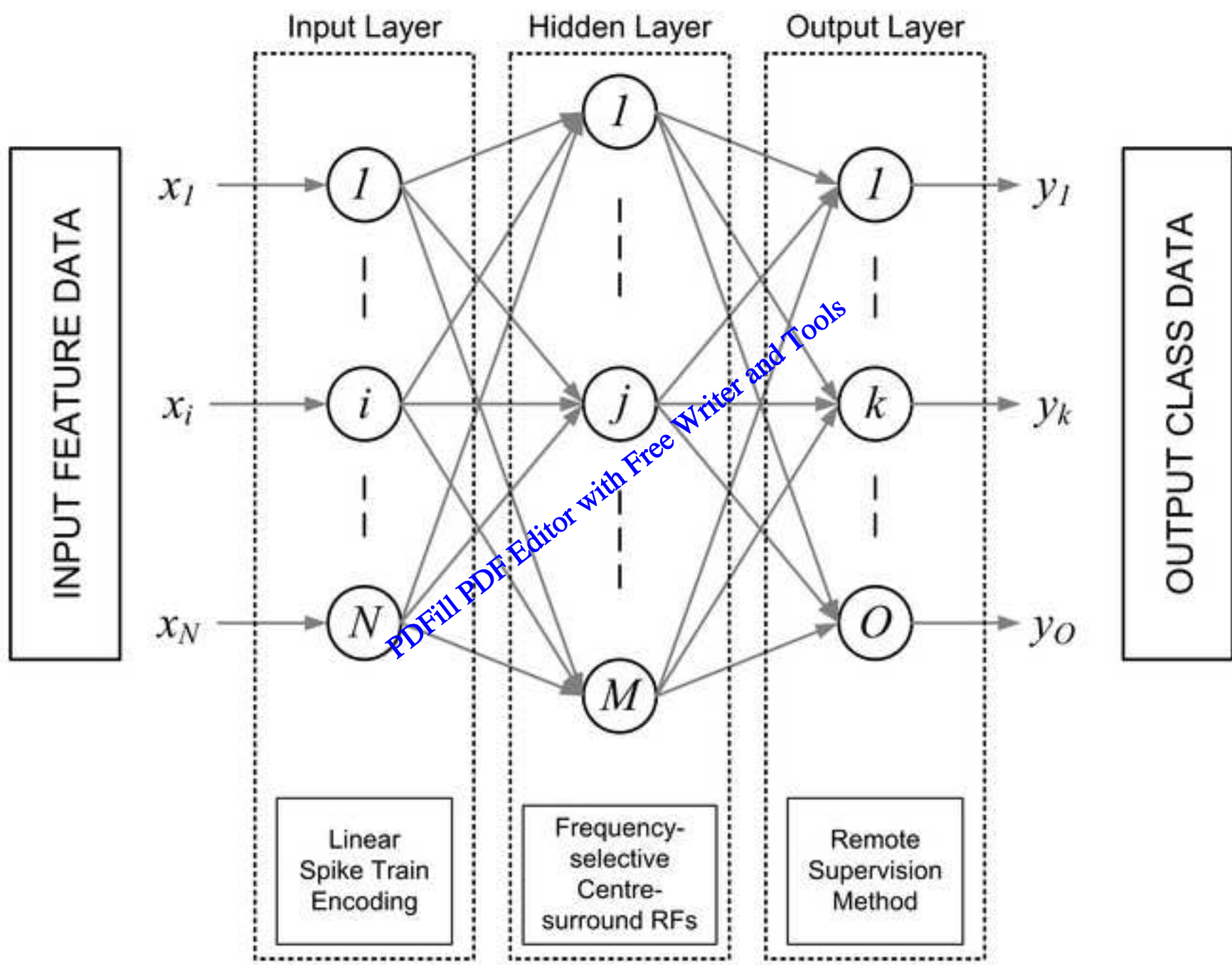


Figure 2 Excitatory/inhibitory receptive field

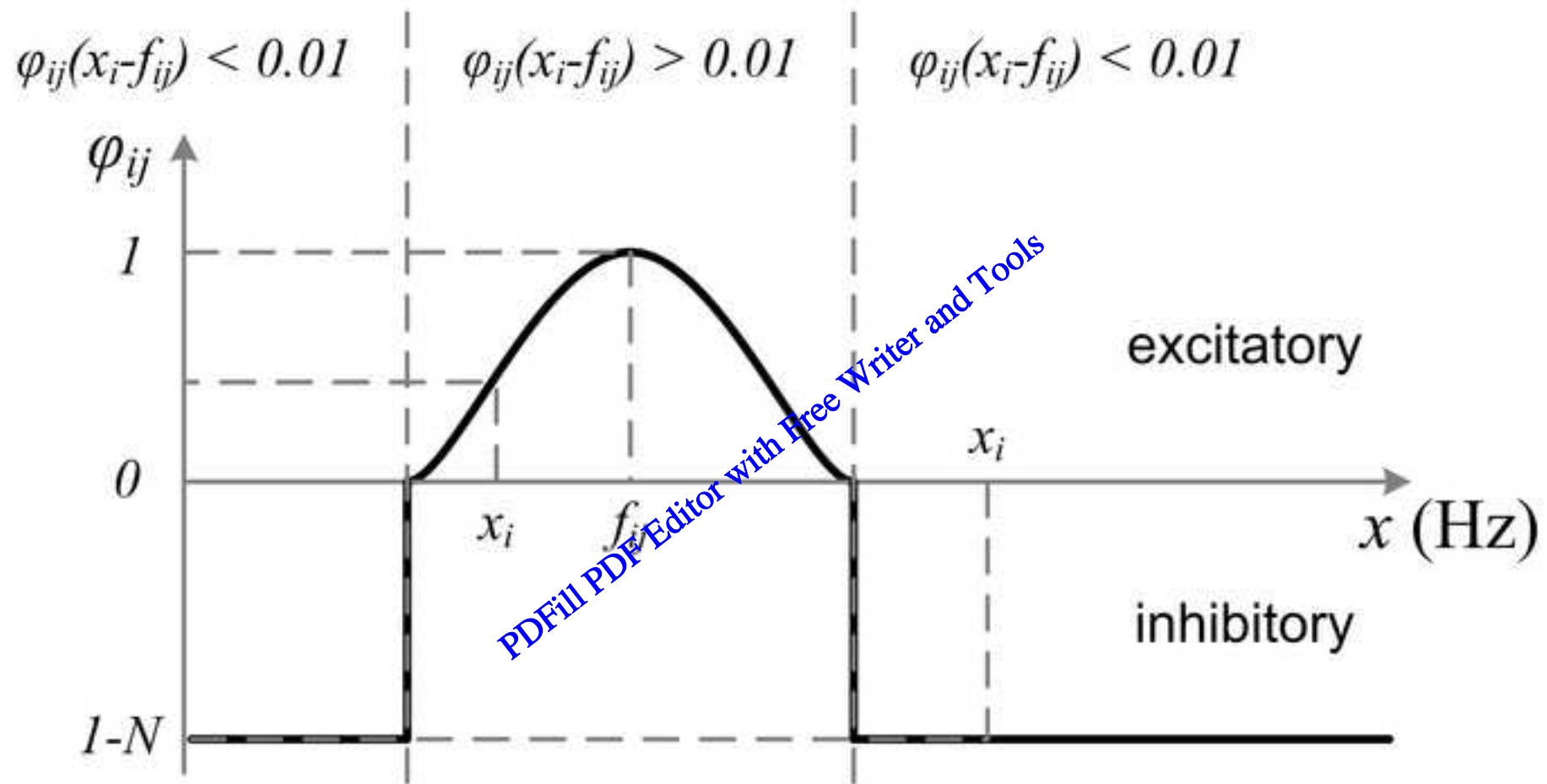


Figure 3 ReSuMe Learning

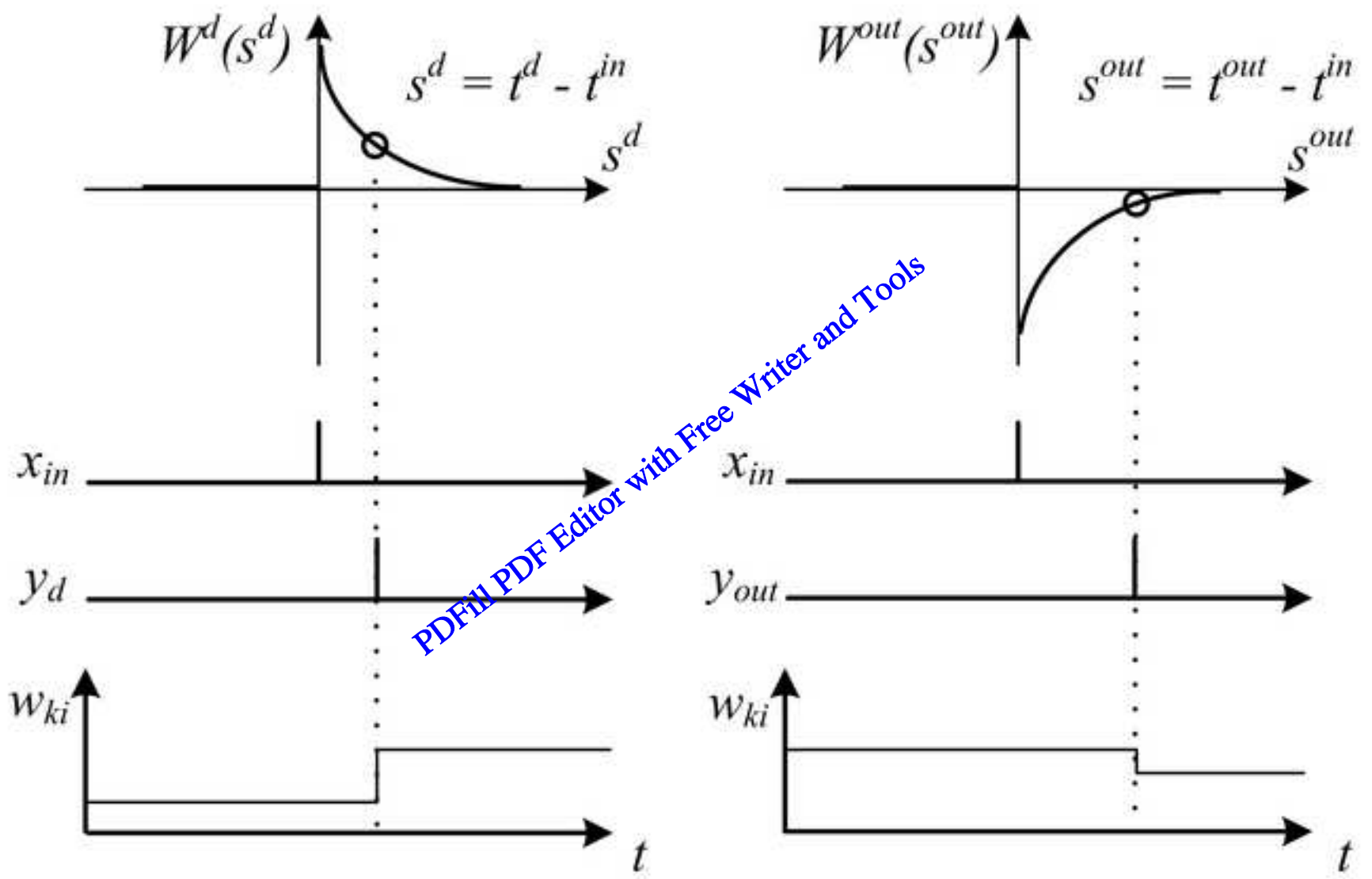


Figure 4 Intuitive Thresholding Technique

		Data									Thresholds	
		CLASS 1			CLASS 2			CLASS 3				
		1	2	3	4	5	6	7	8	9		
Clusters	CLASS 1	1	0.05	0.32	0.54	0.00	0.23	0.18	0.09	0.21	0.08	0.31
		2	0.46	0.03	0.28	0.00	0.10	0.00	0.09	0.08	0.07	0.27
	CLASS 2	3	0.18	0.00	0.00	0.23	0.32	0.11	0.01	0.01	0.01	0.23
		4	0.23	0.33	0.00	0.57	0.01	0.44	0.01	0.01	0.01	0.44
	CLASS 3	5	0.08	0.15	0.00	0.00	0.21	0.00	0.23	0.14	0.75	0.22
		6	0.00	0.00	0.00	0.00	0.13	0.00	0.96	0.84	0.00	0.84

Figure 5 Iris Hidden Layer Processing

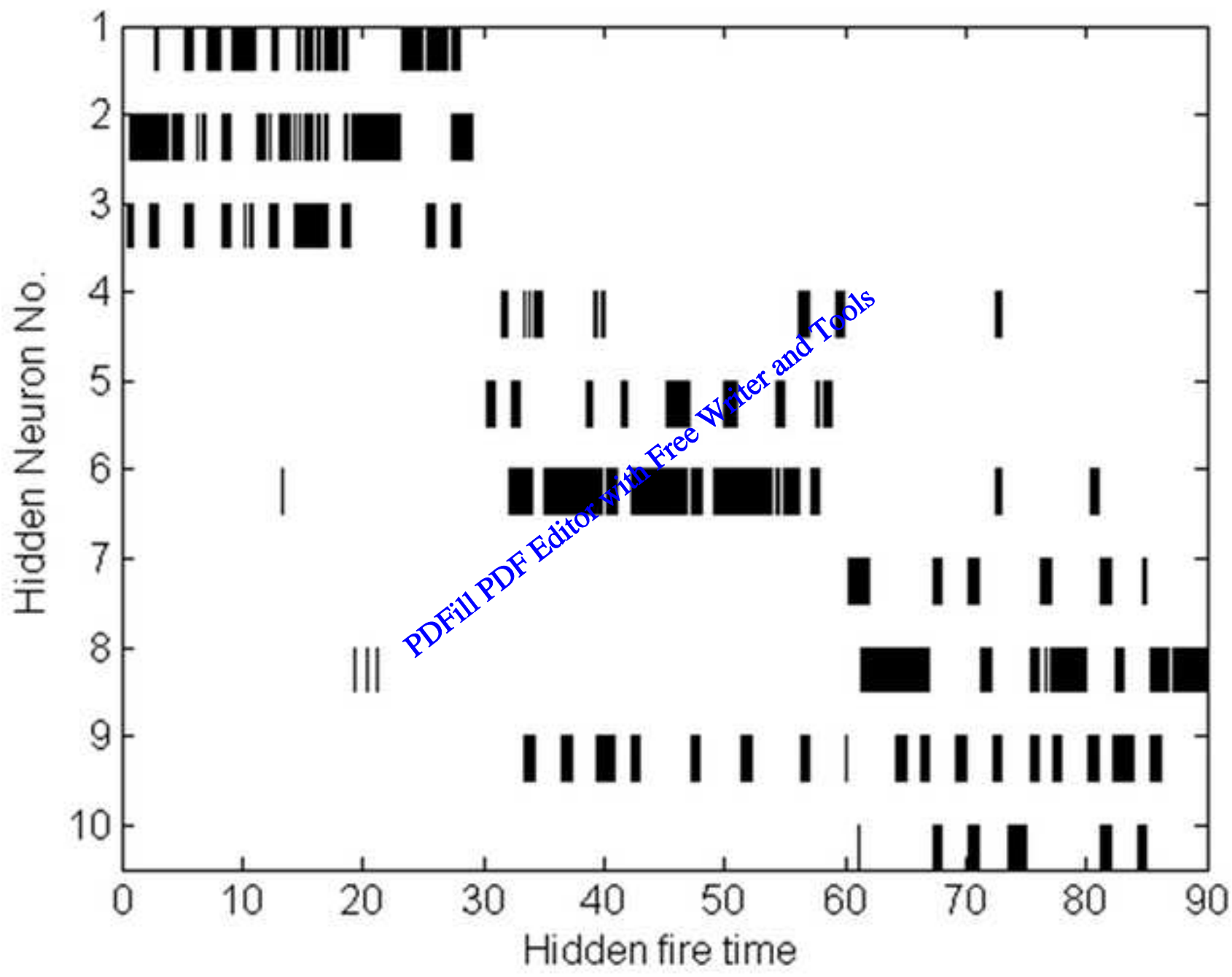


Figure 6 GA Flow of Operation

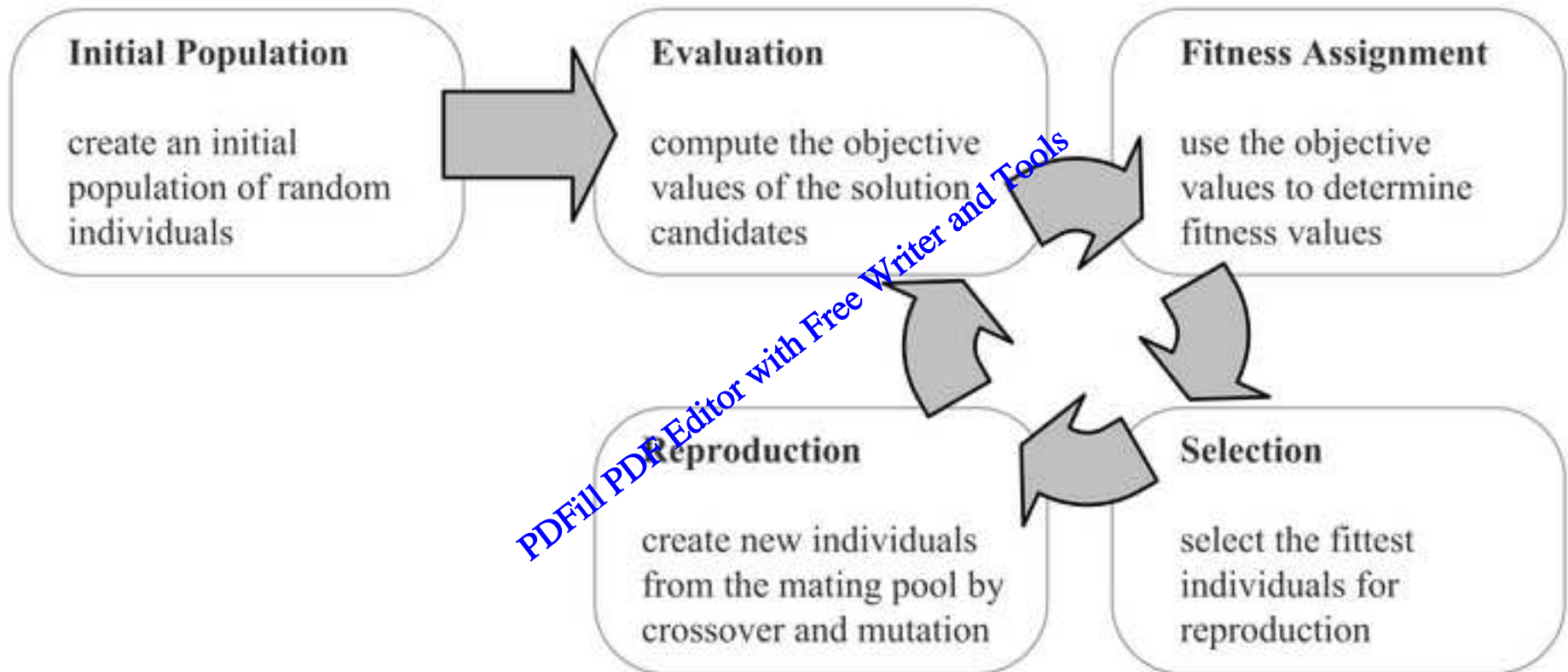


Figure 7 Wisconsin GA Fitness

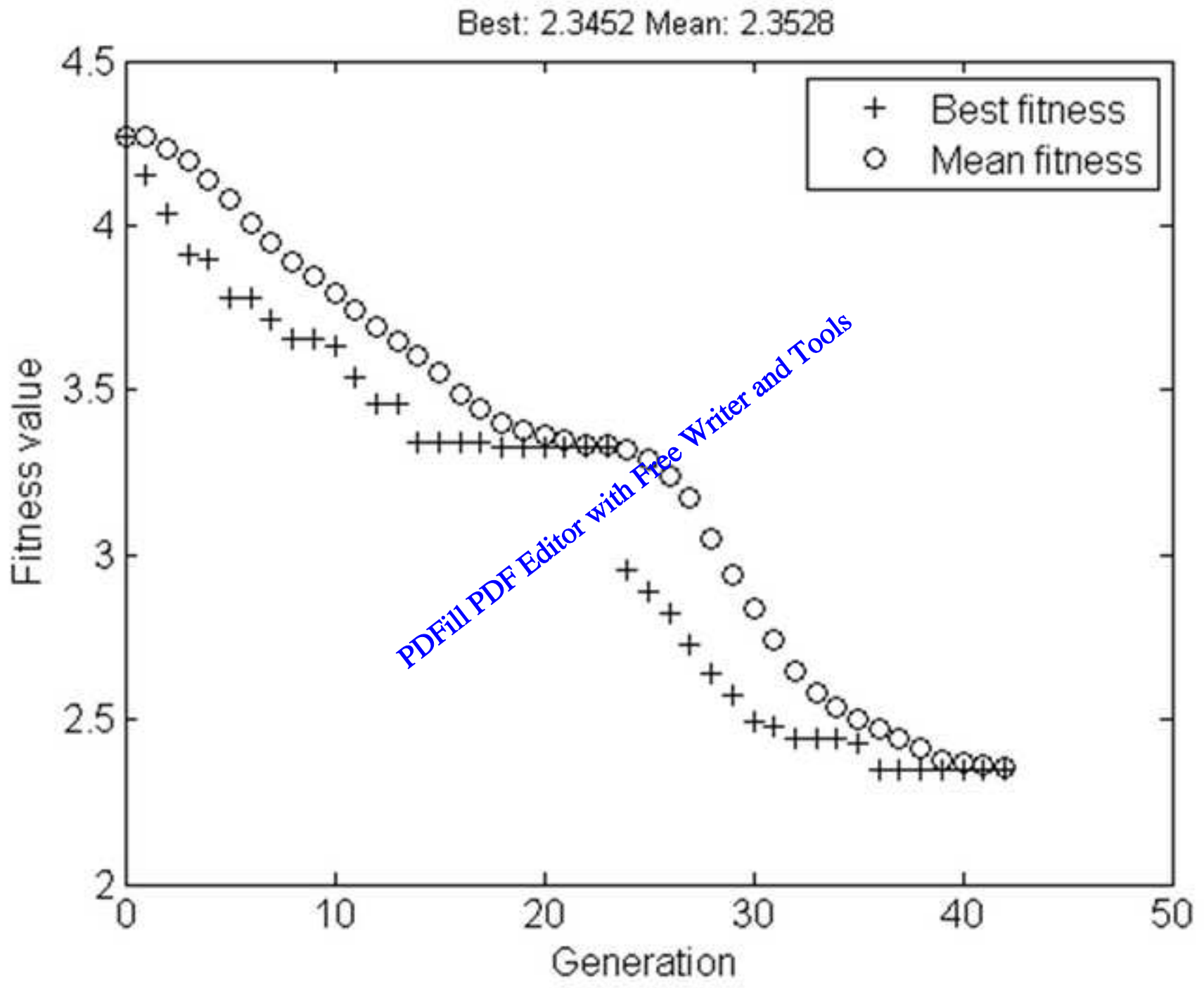


Figure 8 Wisconsin Hidden Layer Processing

