8

Quantum-Inspired Particle Swarm Optimization for Feature Selection and Parameter Optimization in Evolving Spiking Neural Networks for Classification Tasks

Haza Nuzly Abdull Hamed[1,2], Nikola K. Kasabov[1] and Siti Mariyam Shamsuddin[2]
[1]*Auckland University of Technology*
[2]*Universiti Teknologi Malaysia*
[1]*New Zealand*
[2]*Malaysia*

1. Introduction

Particle Swarm Optimization (PSO) was introduced in 1995 by Russell Eberhart and James Kennedy (Eberhart & Kennedy, 1995). PSO is a biologically-inspired technique based around the study of collective behaviour in decentralized and self-organized animal society systems. The systems are typically made up from a population of candidates (particles) interacting with one another within their environment (swarm) to solve a given problem. Because of its efficiency and simplicity, PSO has been successfully applied as an optimizer in many applications such as function optimization, artificial neural network training, fuzzy system control. However, despite recent research and development, there is an opportunity to find the most effective methods for parameter optimization and feature selection tasks.
This chapter deals with the problem of feature (variable) and parameter optimization for neural network models, utilising a proposed Quantum–inspired PSO (QiPSO) method. In this method the features of the model are represented probabilistically as a quantum bit (qubit) vector and the model parameter values as real numbers. The principles of quantum superposition and quantum probability are used to accelerate the search for an optimal set of features, that combined through co-evolution with a set of optimised parameter values, will result in a more accurate computational neural network model. The method has been applied to the problem of feature and parameter optimization in Evolving Spiking Neural Network (ESNN) for classification. A swarm of particles is used to find the most accurate classification model for a given classification task. The QiPSO will be integrated within ESNN where features and parameters are simultaneously and more efficiently optimized. A hybrid particle structure is required for the qubit and real number data types. In addition, an improved search strategy has been introduced to find the most relevant and eliminate the irrelevant features on a synthetic dataset. The method is tested on a benchmark classification problem. The proposed method results in the design of faster and more accurate neural network classification models than the ones optimised through the use of standard evolutionary optimization algorithms.

This chapter is organized as follows. Section 2 introduces PSO with quantum information principles and an improved feature search strategy used later in the developed method. Section 3 is an overview of ESNN, while Section 4 gives details of the integrated structure and the experimental results. Finally, Section 5 concludes this chapter.

2. Particle swarm optimization

PSO is a population-based stochastic optimization technique. In common classifiers, PSO is a global optimization technique that is often used to seek a good set of weights. Similar to other evolutionary algorithms, PSO is initialized with a random population and searches for optimal solutions by updating the particles. Unlike Genetic Algorithms (GA), PSO has no evolution operators such as crossover and mutation. In PSO, the potential solutions, called particles, fly through the problem space by following the current optimum particles.

To create a swarm of $i = 1,...,N$ particles, at all points in time, each particle *i* has

1. A current position X_i or $X_n = \left(x_{i1},...,x_{iD}\right)$,
2. A record of the direction it follows to get to that position V_i or $V_n = (v_{i1},...,v_{iD})$,
3. A record of its own best previous position $pbest = (pbest_1,...,pbest_D)$,
4. A record of the best previous position of any member in its group $gbest = (gbest_1,...,gbest_D)$.

Given the current position of each particle, as well as the other information, the problem is to determine the change in direction of the particles. As mentioned above, this is done by reference to each particle's own experience and its companions. Its own experience includes the direction it comes from V_i and its own best previous position. The experience of others is represented by the best previous position of any member in its group. This suggests that each particle might move in

1. The same direction that it comes from V_i,
2. The direction of its best previous position $pbest - X_i$,
3. The direction of the best previous position of any member in its group $gbest - X_i$.

The algorithm supposes that the actual direction of change for particle *i* will be a weighted combination of (Shi & Eberhart, 1998);

$$V_n = W_x V_n + C_1 * r_1 * (G_{best,n} - X_n) + C_2 * r_2 * (P_{best,n} - X_n) \quad (1)$$

where

r_1 and r_2 are uniform random numbers between $[0,1]$,
$C_1 \rangle 0$ and $C_2 \rangle 0$ are constants called the *cognitive* and *social* parameters, and
$w \rangle 0$ is a constant called the *inertia* parameter.

For successive index periods (generations), n and $n+1$, the direction of change, i.e., the new position of the particle will simply be:

$$X_n = X_n + V_n \quad (2)$$

Given the initial values of X_i, V_i, $pbest$ and $gbest$, Equation (1) and Equation (2) will determine the subsequent path that each particle in the swarm will follow. To avoid particles flying beyond the boundary, the velocities of each dimension are clamped to a maximum velocity, $V_{\max}$. If the sum of accelerations causes the velocity of that dimension to exceed $V_{\max}$, which is a pre-defined parameter, then the velocity is limited to $V_{\max}$.

Obviously, from the above procedure, it seems that PSO shares many common features with GA. Both algorithms start with a randomly generated population, and have a fitness function to evaluate the population. Both methods update the population and search for the optimum solutions with random techniques. However, PSO does not have genetic operators like crossover and mutation. Particles update themselves with the internal velocity, and have memory as storage of history. In PSO, only *gbest* gives the information to others in the population, and it is a one -way information sharing mechanism. The evolution only looks for the best solution, and, in most cases, all the particles tend to converge to the best solution quickly.

2.1 Application in parameter optimization

In neural network models, an optimal combination of parameters can influence their performance. It is not feasible to manually adjust the parameters, particularly when dealing with different combinations for different datasets. Consequently, parameter optimization is vital and much research has been conducted on it (Bäck & Schwefel, 1993). Parameter optimization using PSO works when each particle in the swarm holds the parameter value. This value is considered a particle's position. Updating the particle's position means updating the parameter value. The process begins by initializing the population and all particles with random parameter values. Then, in every iteration, all particles' position are updated based on two factors; the *gbest* and *pbest* . Updating the parameter value based on these two values normally results in a better solution. This updating process is repeated in every iteration until stopping criteria is met, for example a desired fitness value or a maximum number of iterations.

2.2 Quantum inspired probability concept for feature selection

Feature optimization is considered as a crucial pre-processing phase in a classification task. This is because using a higher number of features does not necessarily translate into better classification accuracy. In some cases, having fewer significant features could help reduce the processing time and produce good classification results. Blum and Langley have classified the feature selection techniques into three basic approaches (Blum & Langley, 1997): Embedded approach adds or removes features in response to prediction error on new instances; Filter approach first selects features and then uses them in a classification model; Wrapper approach uses classification accuracy to evaluate features. However, the conventional PSO is inadequate for solving problems that require probability computation such as in the feature selection tasks. Therefore, the quantum information principle is embedded with principles of evolutionary computation in the PSO as a mechanism for feature probability calculation and consequent selection based on these probabilities.

Referring to (Narayanan, 1999; Kasabov, 2007), quantum computing principles have been seen as a source of inspiration for novel computational methods. Two famous quantum applications are factorisation problem (Shor, 1994) and Grover's database search algorithm (Grover, 1996).

According to the normal or classical computing concept, information is represented in bits where each bit must hold a value of either 0 or 1. However, in quantum computing, information is instead represented by a qubit in which a value of a single qubit could be 0, 1, or a superposition of both. Superposition allows the possible states to represent both 0 and 1 simultaneously based on its probability. The quantum state is modelled by the Hilbert space of wave functions and is defined as:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \tag{3}$$

where α and β are complex numbers defining probabilities at which the corresponding state is likely to appear when a qubit collapses, for instance, when reading or measuring. Probability fundamentals stated that $|\alpha|^2 + |\beta|^2 = 1$, where $|\alpha|^2$ gives the probability that a qubit is in the OFF (0) state and $|\beta|^2$ gives the probability that a qubit is in the ON (1) state. The probability of $\begin{bmatrix}\alpha\\ \beta\end{bmatrix}$ can be represented as quantum angle θ, where $\begin{bmatrix}\cos(\theta)\\ \sin(\theta)\end{bmatrix}$ satisfies the probability fundamental of $|\sin(\theta)|^2 + |\cos(\theta)|^2 = 1$. The θ parameter is normally been used in quantum inspired Evolutionary Algorithms (EA) to calculate and update probability. There are several studies where quantum computation acts as probability computation in EA. The Quantum Evolutionary Algorithm (QEA) was popularized by Han and Kim (Han & Kim, 2000). Since then, a lot of attention has been given by researchers around the world to this technique. Descended from the basic EA concept, QEA is a population-based search method which simulates a biological evolutionary process and mechanism, such as selection, recombination, mutation and reproduction. Each individual in a population is a possible solution candidate and is evaluated by a fitness function to solve a given task. However, instead of using normal real value values, information in QEA is represented in qubits. This probability presentation has a better characteristic of diversity than classical approaches. QEA have been reported to successfully solve complex benchmark problems such as numerical (da Cruz et al., 2006), multiobjective optimization (Talbi et al., 2006) and real world problems (Jang et al., 2004).

The quantum computation also has been extended to PSO and this is known as Quantum-inspired Particle Swarm Optimization (QiPSO) (Sun et al., 2004). The main idea of QiPSO is to update the particle position represented as a quantum angle θ. The common velocity update equation in conventional PSO is modified to get a new quantum angle which is translated to the new probability of the qubit by using the following formula:

$$\Delta\theta_n = w * \Delta\theta_{n_{t-1}} + c_1 * rand() * (\theta_{gbest_n} - \theta_n) + c_2 * rand() * (\theta_{pbest_n} - \theta_n) \tag{4}$$

Based on the new θ velocity, the new probability of α and β is calculated using a rotation gate as follows:

$$\begin{bmatrix}\alpha\\ \beta\end{bmatrix} = \begin{bmatrix}\cos(\Delta\theta) & -\sin(\Delta\theta)\\ \sin(\Delta\theta) & \cos(\Delta\theta)\end{bmatrix}\begin{bmatrix}\alpha_{t-1}\\ \beta_{t-1}\end{bmatrix} \tag{5}$$

In a feature selection task, each qubit denoted as quantum angle θ, represents one feature. In this case, the collapse qubit value 1 represents features selected while value 0 represents those not selected.

2.3 Enhancement for parameter optimization and feature selection

There are some problems when using QiPSO algorithms for parameter optimization and feature selection. Therefore, this chapter proposes an improved QiPSO algorithm called Dynamic Quantum-inspired Particle Swarm Optimization (DQiPSO). The problems include

a possibility of missing the optimal parameter value when using only binary QiPSO. As the information is represented in a binary structure, the conversion from binary to real value will lead to such problems, especially if the selected number of qubits representing the parameter value is insufficient. To overcome this problem, a combination of QiPSO and conventional PSO is proposed. The DQiPSO particle is divided into two parts: the first part uses quantum probability computation for feature selection and another part holds the real value for parameters as shown in Figure 1. This method not only effectively solves this problem, but also eliminates one parameter which holds number of qubits representing the parameter value.

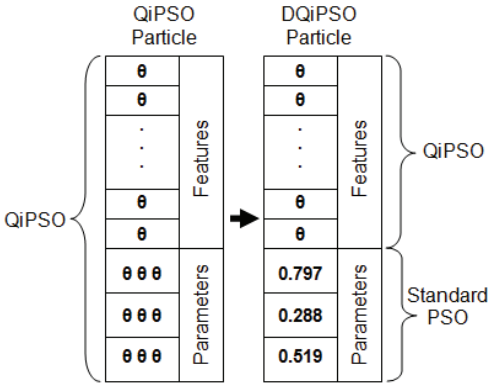


Fig. 1. The proposed hybrid particle structure in DQiPSO

In addition, the search strategy of QiPSO is based on random selection at the beginning of the process. Each particle will update itself based on the best solution subsequently found. A major problem with this approach is the possibility of not selecting the relevant features at the beginning; other particles in the entire process are thus affected. This is due to each particle updating its information without relevant features. Therefore, a new strategy is proposed in which five types of particles in the DQiPSO are considered. Apart from the normal particle, referred to as the Update Particle, which renews itself based on *pbest* and *gbest* information, four new types of particles are added to the swarm. The first type is the Random Particle, which will randomly generate new sets of features and parameters in every iteration to increase the robustness of the search. The second type is the Filter Particle, which selects one feature at a time and feeds it to the network and calculates the fitness value. This process is repeated for each feature. Any features with above average fitness will be considered as relevant. This method is targeted at linear separation problems. The third particle type is the Embed In Particle in which input features are added to the network one by one. If a newly added feature improves fitness, it will be considered a relevant feature. Otherwise, the feature will be removed. The final particle type is the Embed Out Particle which starts the identification process with all features fed to the network to get the initial fitness value. These features are gradually removed one by one. If removing a feature causes decrement of the fitness value, then this feature will be considered relevant and hence will be kept. Otherwise, the feature will be considered irrelevant and removed.

The main idea behind Filter, Embed In and Embed Out particles is to identify the relevance of each feature and to reduce the number of candidates until a small subset remains. For

subsequent iterations, features considered relevant will be selected randomly to find the best combination of significant features. This strategy helps to solve unevaluated relevant features, while reducing the search space and facilitating the optimizer in finding relevant features faster. Similar to the standard PSO in updating the particles, if a new particle is found to be the best solution, then it will be stored as a *gbest* . In this scenario, some improvements have also been proposed for the update strategy. This includes replacing the *gbest* particle with a new particle if the fitness value is higher or equivalent, but with a lower number of selected features. Due to the robust search space provided by DQiPSO, fewer particles are needed to perform the optimization tasks; hence, less processing time can be achieved. The structure of this strategy is illustrated in Figure 2.

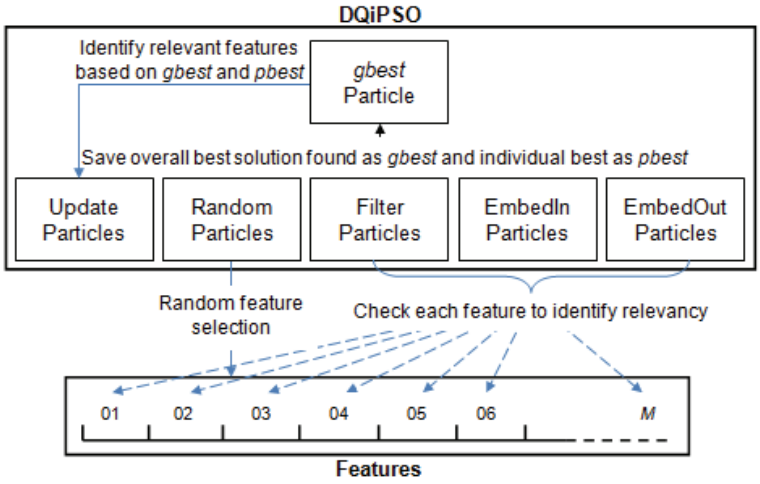


Fig. 2. DQiPSO feature selection strategy

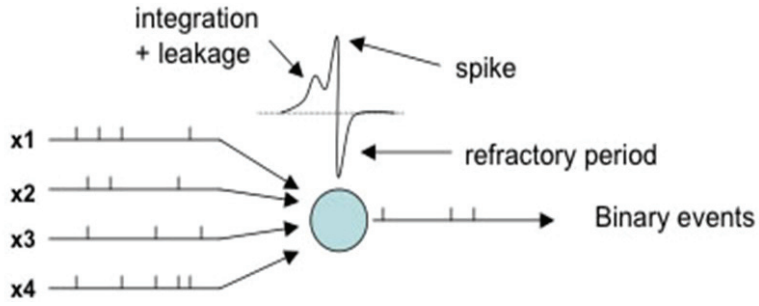
3. Evolving spiking Neural Networks

Many successful Artificial Neural Network (ANN) models have been developed and applied for learning from data and for generalization to new data (Arbib, 2003). Applications include: classification, time series prediction, associative storage and retrieval of information, robot and process control, medical and business decision support, and many others (Arbib, 2003). Most of these ANN use simple and deterministic models of artificial neurons, such as the McCulloch and Pitts model (McCulloch & Pitts, 1943). They also use rate coded information representation, where average activity of a neuron or an ANN is represented as a scalar value. Despite the large structural diversity of existing ANN, the limited functionality of the neurons and connections between them has constrained the scope of applications of ANN and their efficiency when modelling large scale, noisy, dynamic and stochastic processes such as ecological, environmental, physical, biological, cognitive, and others.

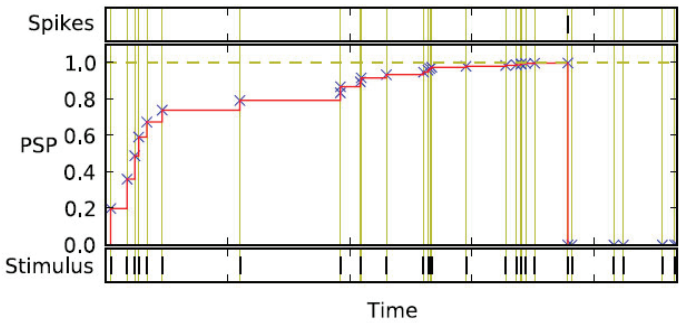
Recently new knowledge about neuronal, genetic and quantum levels of information processing in biological neural networks has been discovered. For example, whether a neuron spikes or not at any given time could depend not only on input signals but also on

gene and protein expression (Kojima & Katsumata, 2009), physical properties of connections (Huguenard, 2000), probabilities of spikes being received at the synapses, emitted neurotransmitters, open ion channels and others. Many of these properties have been mathematically modelled and used to study biological neurons (Gerstner & Kistler, 2002), but have not been properly utilised for more efficient ANN for complex AI problems.

Spiking Neural Networks (SNN) models are made up of artificial neurons that use trains of spikes to represent and process pulse coded information. In biological neural networks, neurons are connected at synapses and electrical signals (spikes) pass information from one neuron to another. SNN are biologically plausible and offer some means for representing time, frequency, phase and other features of the information being processed. A simplified diagram of a spiking neuron model is shown in Figure 3a. Figure 3b shows the mode of operation of a spiking neuron, which emits an output spike when the total spiking input – Post Synaptic Potential (PSP), is larger than a spiking threshold.



(a)



(b)

Fig. 3. (a) A simplified diagram of a spiking neuron model. (b) A spiking neuron emits an output spike when the total spiking input – Post Synaptic Potential (PSP), is larger than a spiking threshold.

Based on the SNN, Evolving SNN (ESNN) was introduced (Wysoski et al., 2006) where SNN evolve their structure through fast one-pass learning from data and have the potential for solving complex problems. ESNN have been applied for classification tasks, such as face recognition, person authentication based on audiovisual information, taste recognition. They achieved better results than previously published models. The ESNN architecture consists of an encoding method for real value data to spike time, neuron model and learning method.

3.1 Information encoding methods

The information in ESNN is represented as spikes; therefore, input information must be encoded in spike pulses. The well-known encoding technique for ESNN is the Population Encoding (Bohte et al., 2002). Population Encoding distributes a single input value to multiple pre-synaptic neurons. Each pre-synaptic neuron generates a spike at firing time.The firing time is calculated using the intersection of Gaussian function. The centre of the Gaussian function is calculated using Equation (6) and the width is computed using Equation (7) with the variable interval of $\left[I_{\min}, I_{\max}\right]$. The parameter β controls the width of each Gaussian receptive field.

$$\mu = I_{\min} + (2 * i - 3) / 2 * (I_{\max} - I_{\min}) / (M - 2) \tag{6}$$

$$\sigma = 1 / \beta (I_{\max} - I_{\min}) / (M - 2) \quad where \quad 1 \le \beta \le 2 \tag{7}$$

The illustration of this encoding process is shown in following figure.

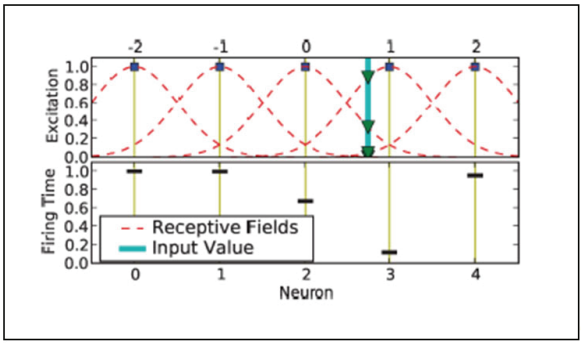


Fig. 4. Population Encoding Method.

3.2 Neuron models

Most of the SNN models have been well explained by Gerstner and Kistler (Gerstner & Kistler, 2002). For the ESNN, Thorpe's neuron model (Thorpe, 1997) has been selected because of its effectiveness and simplicity. The fundamental concept of this model is that the earlier spikes received by a neuron have a stronger weight compared with later spikes. Once the neuron reaches a certain amount of spikes and the Post-Synaptic Potential (PSP) exceeds the threshold value, it fires and becomes disabled. The neuron in this model can only fire once. The computation of the PSP of neuron i is presented in Equation (8).

$$PSP_i = \begin{cases} 0 & if \quad fired \\ \sum w_{ji} * Mod_i^{order(j)} & else \end{cases} \tag{8}$$

where w_{ji} being the weight of pre-synaptic neuron j . Mod_i being a parameter called modulation factor with an interval of [0,1]and $order^{(j)}$ representing the rank of the spike emitted by the neuron. The $order^{(j)}$ starts with 0 if it spikes first amongst all pre-synaptic neurons and increases according to firing time.

3.3 Learning method

Learning in ESNN is a complex process since information is represented in spikes, which is time dependence. Spike Time Dependent Plasticity (STDP) is a form of Hebbian Learning where spike time and transmission are used in order to calculate the change in the synaptic weight of a neuron. If a pre-synaptic spike arrives at the synapse before the postsynaptic action potential, the synapse is potentiated; if the timing is reversed, the synapse is depressed (Markram et al., 1997).

The One-Pass Algorithm is the learning algorithm for ESNN which follows both the SDTP learning rule and the time-to-first spike learning rule (Thorpe,1997). In this algorithm, each training sample creates a new output neuron. The trained threshold values and the weight pattern for that particular sample are stored in the neuron repository. However, if the weight pattern of the trained neuron greatly resembles a neuron in the repository, it will merge into the most similar one. The merging process involves modifying the weight pattern and the threshold of the merged neurons to the average value. Otherwise, it will be added to the repository as a newly trained neuron. The major advantage of this learning algorithm is the ability of the trained network to learn incrementally new samples without retraining.

3.4 ESNN structure

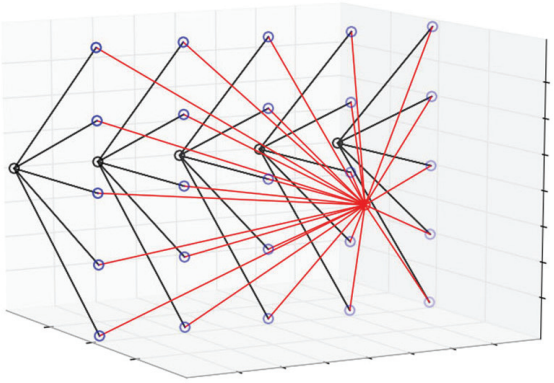


Fig. 5. A simplified ESNN structure

In general, each input neuron in the ESNN (black neuron in Figure 5) is connected to multiple pre-synaptic neurons (blue neurons). This process will transform the input values into a highly dimensional structure where each pre-synaptic neuron generates a certain spike at a firing time. The firing time is calculated using the intersection of the Gaussian function with the input value. Based on the firing time, a weight for each connection to the output neuron (red neuron) is generated. In the training process, the output neuron stores the computed weight of all pre-synaptic neurons, a threshold value to determine when the output neuron will spike and the class label the input sample belongs to. In the testing process, similar to the training process, each testing sample is encoded to spikes by the multiple pre-synaptic neurons. Then, the Post-Synaptic Potential (PSP) of the output class neurons is calculated. Once the neuron reaches a certain amount of spikes and the PSP exceeds the threshold value, it fires an output spike and becomes disabled. The testing

sample belongs to an output class of the output neuron which fires first among all output neurons. A detailed ESNN training algorithm follows.

Algorithm 1 ESNN Training Algorithm

1: Initialize neuron repository $R = \{\ \}$

2: Set ESNN parameters $Mod = [0,1]$, $C = [0,1]$ and $Sim = [0,1]$

3: **for every** input sample i that belongs to the same output class **do**

4: Encode input samples into firing time of pre-synaptic neurons j
Create a new output neuron for this class and calculate the connection weights as follows: $w_j = (Mod)^{order(j)}$

5: Calculate $PSP_{\max(i)} = \sum w_j * Mod^{order(j)}$

6: Get PSP threshold value $\chi_i = PSP_{\max(i)} * C$

7: **if** the new neuron weight vector <= Sim of trained output neuron weight vector in R **then**

8: Merge the weights and the threshold of the new neuron with the most similar neuron in the same class output group

9: $$w = \frac{w_{new} + w * N}{N + 1}$$

10: $$\chi = \frac{\chi_{new} + \chi * N}{N + 1}$$

11: where N is the number of all previous merges of the merged neuron

12: **Else**

13: Add the new neuron to the output neuron repository R

14: **end if**

15: **end for** (Repeat to all input samples for other output classes)

4. Integrated structure for parameter optimization and feature selection

An integrated structure is presented in which the features and parameters are optimized simultaneously, and this leads to better optimization. This experiment further tests the efficiency of DQiPSO in selecting the most relevant features and also optimizing the ESNN parameters. Here the DQiPSO optimizes the ESNN parameters: Modulation Factor (Mod), Proportion Factor (C) and Similarity (Sim), as well as identifies the relevant features. All particles are initialized with random value and subsequently interact with each other based on classification accuracy. Since there are two components to be optimized, each particle is divided into two parts. The first part of each hybrid particle holds the feature mask where information is stored in a string of qubits. Another part holds parameters of ESNN. The proposed integrated framework is shown in Figure 6.

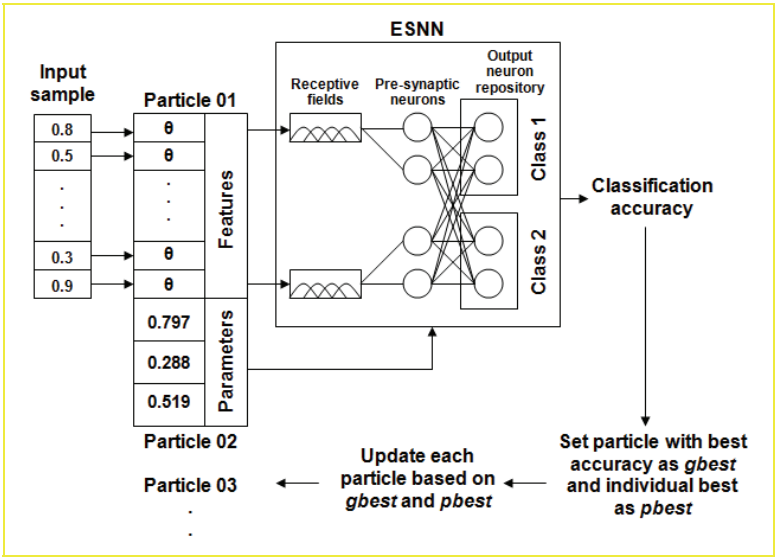


Fig. 6. An integrated ESNN-DQiPSO framework for feature selection and parameter optimization.

4.1 Setup

The proposed integrated ESNN-DQiPSO method was tested on a Uniform Hypercube dataset (Estavest et al., 2009). Thirty features were created where only 10 are the relevant features, where a sample belongs to class 1 when $r_i < \gamma^{i-1} * \alpha$ for i = 1 till 10. Parameters chosen were $\gamma = 0.8$ and $\alpha = 0.5$. The features which are not relevant to determining the output class consist of 10 random features with the random value of $[0,1]$, and 10 redundant features copied from relevant features with an addition of Gaussian noise of 0.3. The features were arranged randomly to simulate the real world problem where relevant features are scattered in the dataset as follows:

Features	Arrangement
Relevant	02, 04, 09, 10, 11, 15, 19, 20, 26, 30
Redundant	03, 07, 12, 14, 17, 18, 21, 25, 27, 28
Random	01, 05, 06, 08, 13, 16, 22, 23, 24, 29

Table 1. Feature arrangement

The problem consists of 500 samples, equally distributed into two classes. It was applied to the proposed framework and compared with the QiPSO method and ESNN with standard PSO. However, because standard PSO is inadequate for feature selection, it only optimizes the ESNN parameters. Based on the preliminary experiment, 20 ESNN's pre-synaptic neurons were chosen. For the DQiPSO, 18 particles were used, consisting of six Update, three Filter, three Random, three Embed In and three Embed Out. For the QiPSO, 20

particles were used. C_1 and C_2 were set to 0.05 to balance the exploration between *gbest* and *pbest* with the inertia weight w = 2.0. Ten-fold cross validations were used and the average result was computed in 500 iterations.

4.2 Results

Figure 7 illustrates the comparison of selected features from DQiPSO and QiPSO during the learning process. The lighter colour means more frequent corresponding features are

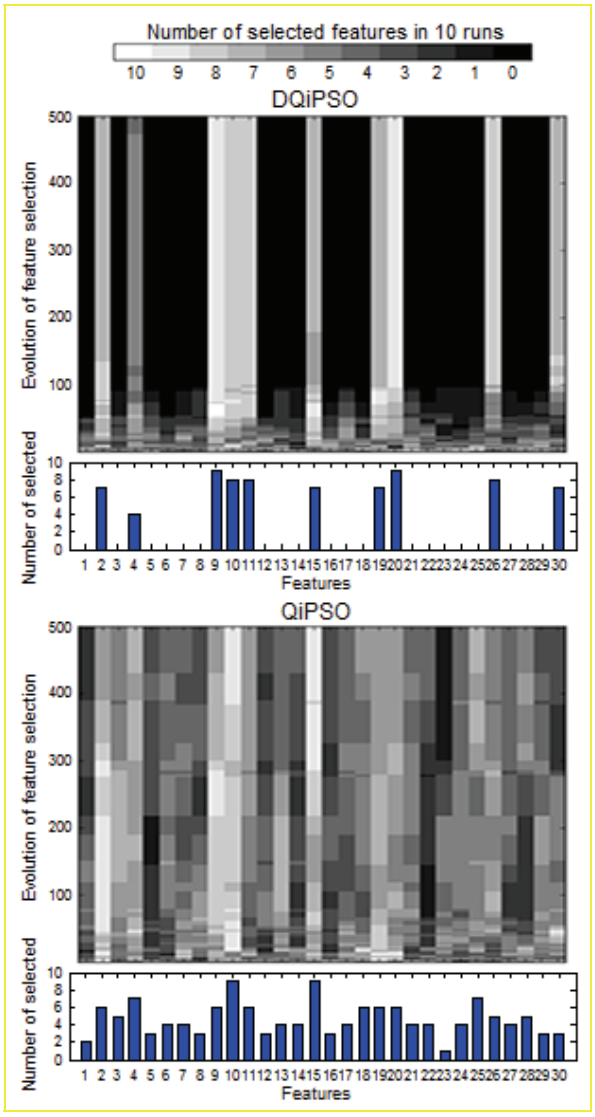


Fig. 7. Evolution of feature selection

selected and darker means otherwise. All the features have been ranked based on the number of selected features from 10 runs to determine their relevancy. From the figure, 10 relevant features which contained the most information can be clearly identified and are constantly being selected by DQiPSO. In contrast, the redundant and random features are completely rejected during the optimization process. DQiPSO takes less than 100 iterations to identify the relevant and irrelevant features. Based on the feature ranking, the most relevant features found are: Feature 9 and Feature 20, followed by Feature 10, Feature 11, Feature 26, Feature 2, Feature 15, Feature 19, Feature 30 and Feature 4. In contrast, the ability of the QiPSO to reject the irrelevant features is unsatisfactory. Most of the irrelevant features are still being selected, which contributes to the low classification accuracy and increased computation time. The most relevant features found by QiPSO are Feature 10 and 15, followed by Feature 4, Feature 25, Feature 2, Feature 9, Feature 11, Feature 18, Feature 19 and Feature 20. Other features are occasionally selected and can be considered as irrelevant features by QiPSO. Some relevant features are also being regarded as irrelevant due to the number of selected is low, while some irrelevant features which contain no information are considered as relevant by QiPSO. This situation has affected the results and overall classification performance of the ESNN-QiPSO.

Figure 8 shows the results of parameter optimization. All parameters evolve steadily towards a certain optimal value, where the correct combination together with the selected relevant features leads to a better classification accuracy. In terms of the classification result, the average accuracy for ESNN-DQiPSO is 99.25% with the result of every single run consistently above 98%. For the ESNN-QiPSO algorithm, the average accuracy is 96.57%. The proposed DQiPSO and QiPSO methods are able to select relevant features with few or occasionally no irrelevant features, while simultaneously providing nearly optimal parameter combinations in the early stage of learning. This situation leads to acceptably

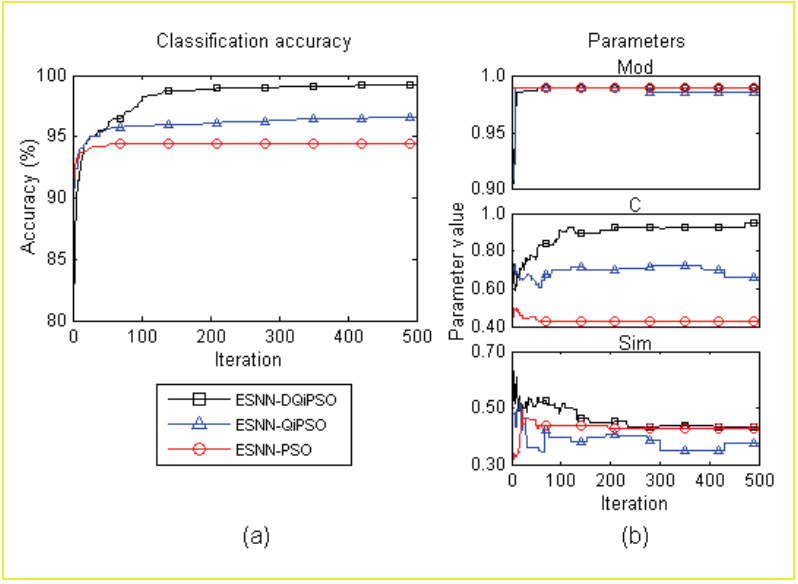


Fig. 8. a) Classification accuracy and b) Parameter optimization result

high average accuracy at the beginning of the learning process. For the ESNN-PSO algorithm, although the classification accuracy is 94.43%, this algorithm is entirely dependent on the parameter optimization which has affected the results, giving the lowest accuracy. The testing results for ESNN-DQiPSO, ESNN-QiPSO and ESNN-PSO are 95.99%, 91.58% and 83.93% respectively.

5. Conclusion and future research

This chapter has introduced a new PSO model and has shown how this optimizer can be implemented for parameter optimization and feature selection. Since feature selection is a unique task involving probability, quantum computation has been embedded into PSO and has been applied to an ESNN for classification. The new method results in a more efficient classification ESNN model with optimal features selected and parameters optimised.
Future work is planned to improve the proposed optimization method and to apply it to the Probabilistic Spiking Neural Networks (PSNN) (Kasabov, 2010). In this PSNN, not only features will be represented by a quantum bit vector, but also all connections between the neurons. A neuronal connection is either existent (1) or nonexistent (0), or in another interpretation – either propagating a spike or not propagating it. A quantum bit vector would be a suitable representation of all connections that can be optimized using the modified PSO. Each particle will be divided into three parts; the first two parts use quantum probability computation for feature and connection selection and the last part holds the real value for the parameters. It is to be believed that the proposed method will be able to optimize the given problem in a far more efficient way.

6. References

Arbib, M. (2003). *The Handbook of Brain Theory and Neural Networks*, MIT Press, ISBN 978-0-262-01148-8, Cambridge, MA, USA.

Bäck, T. & Schwefel, H. (1993). An Overview of Evolutionary Algorithms for Parameter Optimization. *Evolutionary Computation*, Vol. 1, No. 1, (March 1993), pp. 1-23, ISSN 1063-6560.

Blum, L.A. & Langley, P. (1997). Selection of Relevant Features and Examples in Machine Learning. *Artificial Intelligence*, Vol. 97, No. 1-2, (December 1997), pp. 245–271, ISSN 0004-3702.

Bohte, S.M.; Kok, J.N. & Poutre, H.L. (2002). Error-Backpropagation in Temporally Encoded Networks of Spiking Neurons. *Neurocomputing*, Vol. 48, No. 1-4, (October 2002), pp. 17-37, ISSN 0925-2312.

da Cruz, A.V.A.; Vellasco, M.M.B. & Pacheco, M.A.C. (2006). Quantum Inspired Evolutionary Algorithm for Numerical Optimisation, *Proceedings of IEEE Congress on Evolutionary Computation*, pp. 2630-2637, ISBN 0-7803-9487-9, Vancouver, Canada, July 16-21, 2006.

Eberhart, R. & Kennedy, J. (1995). A New Optimizer Using Particle Swarm Theory, *Proceedings of Sixth International Symposium on Micro Machine and Human Science*, pp. 39-43, ISBN 0-7803-2676-8, Nagoya, Japan, October 4-6, 1995.

Estavest, P.; Tesmer, M.; Perez, C.; & Zurada, J. (2009). Normalized Mutual Information Feature Selection. *IEEE Transactions on Neural Networks*, Vol. 20, No. 2, (February 2009), pp. 189-201, ISSN 1045-9227.

Gerstner, W. & Kistler. W. (2002). *Spiking Neuron Models: An Introduction*, Cambridge University Press, ISBN 0521890799, New York, NY, USA.

Grover, L. K. (1996). A Fast Quantum Mechanical Algorithm for Database Search, *Proceedings of 28th annual ACM symposium on Theory of computing*, pp. 212–219, ISBN 0-89791-785-5, Philadelphia, USA, May 22-24, 1996.

Han, K.H. & Kim, J.H. (2000). Genetic Quantum Algorithm and its Application to Combinatorial Optimization Problem, *Proceedings of 2000 Congress on Evolutionary Computation*, pp. 1354-1360, ISBN 0-7803-6375-2, La Jolla, CA , USA, July 16-19, 2000.

Huguenard, J.R. (2000). Reliability of Axonal Propagation: The Spike Doesn't Stop Here. *Proceedings of the National Academy of Sciences of the United States of America*, Vol. 97, No. 17, (August 2000), pp. 9349-9350, ISSN 1091-6490.

Jang, J.S.; Han, K.H. & Kim, J.H. (2004). Face Detection Using Quantum-Inspired Evolutionary Algorithm, *Proceedings of IEEE Congress on Evolutionary Computation*, pp. 2100-2106, ISBN 0-7803-8515-2, Portland, OR, USA, June 19-23, 2004.

Kasabov, N. (2007). *Evolving Connectionist Systems* (2nd ed.), Springer-Verlag, ISBN 978-1-84628-345-1, Secaucus, NJ, USA.

Kasabov, N. (2010). To Spike or Not To Spike: A Probabilistic Spiking Neural Model. *Neural Networks*, Vol. 23, No. 1, (January 2010), pp. 16-19, ISSN 0893-6080.

Kojima, H. & Katsumata, S. (2008). Analysis of Synaptic Transmission and Its Plasticity by Glutamate Receptor Channel Kinetics Models and 2-Photon Laser Photolysis, *Proceedings of International Conference on Neural Information Processing*, pp. 88-94, ISBN 3-642-02489-0, Auckland, New Zealand, November 25-28, 2008.

Markram, H.; Lubke, J.; Frotscher, M. & Sakmann, B. (1997). Regulation of Synaptic Efficacy by Coincidence of Postsynaptic Aps and Epsps. *Science*, Vol. 275, No. 5297, (January 1997), pp. 213-215, ISSN 0193-4511.

McCulloch, W.S. & Pitts, W. A. (1943). Logical Calculus of the Ideas Immanent in Nervous Activity. *Bulletin of Mathematical Biology*, Vol. 5, No. 4. (December 1943), pp. 115-133, ISSN 00074985.

Narayanan, A. (1999). Quantum Computing for Beginners, *Proceedings of Congress on Evolutionary Computation*, pp. 2231-2238, ISBN 0-7803-5536-9, Washington, DC , USA, July 6-9, 1999.

Shi, Y. & Eberhart, R. (1998). A Modified Particle Swarm Optimizer, *Proceedings of IEEE International Conference on Evolutionary Computation*, pp. 69–73, ISBN 0-7803-4869-9, Anchorage, USA, May 4-9, 1998.

Shor, P.W. (1994). Algorithms for Quantum Computation: Discrete Log and Factoring, *Proceedings of IEEE Symposium on Foundations of Computer Science*, pp. 124-134, ISBN 0-8186-6580-7, Santa Fe, NM, USA, November 20-22, 1994.

Sun, J.; Feng, B. & Xu, W. (2004). Particle Swarm Optimization with Particles Having Quantum Behavior, *Proceedings of Congress on Evolutionary Computation*, pp. 325-331, ISBN 0-7803-8515-2, Portland, Oregon, USA, June 20-23, 2004.

Talbi, H.; Draa, A. & Batouche, M. (2006). A Novel Quantum Inspired Evaluation Algorithm for Multisource Affine Image Registration. *The International Arab Journal of Information Technology*, Vol. 3, No. 1, (January 2006), pp. 9-15, ISSN 1683-3198.

Thorpe, S.J. (1997). How Can the Human Visual System Process a Natural Scene in Under 150ms? Experiments and Neural Network Models, In: *Proceedings of European Symposium on Artificial Neural Networks*, Verleysen, M. (ed.), D-Facto public, ISBN 2-9600049-7-3, Bruges, Belgium.

Wysoski, S. G.; Benuskova, L. & Kasabov, N. (2006). On-Line Learning with Structural Adaptation in a Network of Spiking Neurons for Visual Pattern Recognition, In: *Proceedings of 2006 International Conference on Artificial Neural Networks, Lecture Notes in Computer Science, Vol. 4131*, Kollias, S. et al. (Eds), pp. 61-70, Springer-Verlag, ISBN 978-3-540-38625-4, Berlin Heidelberg.