Neural Networks 22 (2009) 623–632

Contents lists available at ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet

2009 Special Issue

Integrated feature and parameter optimization for an evolving spiking neural network: Exploring heterogeneous probabilistic models

Stefan Schliebs [a,*], Michaël Defoin-Platel [b], Sue Worner [c], Nikola Kasabov [a]

[a] Knowledge Engineering and Discovery Research Institute (KEDRI), Auckland University of Technology, New Zealand

[b] Biomathematics and Bioinformatics at Rothamsted Research, United Kingdom

[c] Lincoln University, Centre for Bioprotection, New Zealand

ARTICLE INFO

Article history:
Received 6 May 2009
Received in revised form 3 June 2009
Accepted 25 June 2009

Keywords:
Evolving spiking neural network
Quantum-inspired evolutionary algorithm
Multiple probabilistic model
Estimation of distribution algorithm

ABSTRACT

This study introduces a quantum-inspired spiking neural network (QiSNN) as an integrated connectionist system, in which the features and parameters of an evolving spiking neural network are optimized together with the use of a quantum-inspired evolutionary algorithm. We propose here a novel optimization method that uses different representations to explore the two search spaces: A binary representation for optimizing feature subsets and a continuous representation for evolving appropriate real-valued configurations of the spiking network. The properties and characteristics of the improved framework are studied on two different synthetic benchmark datasets. Results are compared to traditional methods, namely a multi-layer-perceptron and a naïve Bayesian classifier (NBC). A previously used real world ecological dataset on invasive species establishment prediction is revisited and new results are obtained and analyzed by an ecological expert. The proposed method results in a much faster convergence to an optimal solution (or a close to it), in a better accuracy, and in a more informative set of features selected.

1. Introduction

Recently spiking neural networks (SNN) (Gerstner & Kistler, 2002; Izhikevich, 2003) have been developed as biologically plausible connectionist models, which use trains of spikes for internal information representation. It was argued that SNN have at least similar computational power than the traditional Multi-Layer-Perceptron derivates (Maass, 1999). Nowadays many studies attempt to use Spiking Neural Networks (SNN) for practical applications, some of them demonstrating very promising results on solving complex real world problems. Substantial progress has been made in areas like speech recognition (Verstraeten, Schrauwen, & Stroobandt, 2005), learning rules (Bohte, Kok, & Poutré, 2002), associative memory (Knoblauch, 2005), and function approximation (Iannella & Kindermann, 2005), just to name a few. Based on Kasabov (2007) an evolving spiking neural network was proposed and applied to audio-visual pattern recognition (Wysoski, Benuskova, & Kasabov, 2006, 2008). A similar type of network was later used in the context of a taste recognition task (Soltic, Wysoski, & Kasabov, 2008).

With encouraging results, spiking neural networks were presented in the context of a feature selection problem (Schliebs, Defoin-Platel, & Kasabov, 2009). In this work a binary state-of-art optimization algorithm, namely the Versatile Quantum-inspired Evolutionary Algorithm (vQEA) (Defoin-Platel, Schliebs, & Kasabov, 2007), was combined with an Evolving Spiking Neural Networks(eSNN) (Wysoski et al., 2006). Through implementing quantum principles, vQEA evolves in parallel a number of independent probability vectors, that may interact at certain intervals with each other, forming a multi-model Estimation of Distribution Algorithm (EDA) (Defoin-Platel, Schliebs, & Kasabov, 2009).

Following the wrapper approach, vQEA was used to identify relevant feature subsets and simultaneously evolve an optimal eSNN parameter setting. This extended architecture was referred to as the Quantum-inspired SNN (QiSNN) framework. Applied to carefully designed benchmark data, containing irrelevant and redundant features of varying information quality, the QiSNN-based feature selection led to excellent classification results and an accurate detection of relevant information in the dataset.

The QiSNN framework was used on a case study of ecological modeling (Schliebs, Defoin-Platel, Worner, & Kasabov, 2009). Meteorological data, such as monthly and seasonal temperature, rain fall and soil moisture recordings for different geographical sites, were compiled from published results, and each global site was labeled according to the presence or absence of the

* Corresponding author.
E-mail addresses: sschlieb@aut.ac.nz, sschliebs@gmail.com (S. Schliebs), michael.defoinplatel@gmail.com (M. Defoin-Platel), worner@lincoln.ac.nz (S. Worner), nkasabov@aut.ac.nz (N. Kasabov).

0893-6080/$ – see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.neunet.2009.06.038

Mediterranean fruit-fly (a serious fruit pest). The study aimed towards the identification of important features relevant for predicting the presence/absence of this insect species. Results have been compared to the classical Naïve Bayesian Classifier (NBC) and obtained feature subsets were verified by an ecological expert.

In this study we want to extend the work presented in Schliebs et al. (2009) by studying the QiSNN framework on two benchmark and one real world problem. We will start our analysis by introducing a novel combined optimization algorithm, which allows us to explore heterogeneous search spaces simultaneously. The method uses a binary representation for optimizing feature subsets and a continuous representation for evolving appropriate real-valued configurations of a spiking network. Altogether four methods are here experimentally compared to each other: QiSNN as presented in Schliebs et al. (2009), the enhanced QiSNN using the combined optimization algorithm, a multi-layer perceptron and a classical NBC. A comprehensive analysis of the results obtained from the benchmarks experiments in terms of consistency of selected feature subsets, classification accuracy, computational complexity and evolution of parameters in the QiSNN framework is presented. Furthermore, we will point out some significant differences between QiSNN and its enhanced version. Finally we revisit the ecological dataset used in Schliebs et al. (2009) and new results are obtained and analyzed by an ecological expert.

In the following sections first the QiSNN framework is summarized. The novel continuous optimization method is introduced and the simultaneous exploration of a binary and a continuous search space is discussed. QiSNN is then experimentally studied, followed by an analysis and discussion of the obtained results.

2. Framework and implementation of QiSNN

Based on our previous results on eSNN and quantum inspired evolutionary algorithms (Benuskova & Kasabov, 2007; Defoin-Platel et al., 2009; Kasabov, 2007; Wysoski et al., 2008), here we propose and explore an integrative quantum inspired feature selection using the eSNN architecture, tightly coupled with the learning environment (the data).

2.1. The eSNN architecture

The eSNN architecture uses a computationally very simple and efficient spiking neural model, in which early spikes, received by a neuron, are more strongly weighted than later ones. The model was inspired by the neural processing of the human eye, which performs a very fast image processing. Experiments have shown that a primate only needs several hundreds of milliseconds to make reliable decisions about images that were presented in a test scenario (VanRullen & Thorpe, 2001). Since it is known that neural image recognition involves several succeeding layers of neurons, these experiments suggested that only very few spikes could be involved in the neural chain of image processing. In Thorpe (1997) a mathematical definition of these neurons was proposed and tested on some face recognition tasks, reporting encouraging experimental results. The same model was later extended to eSNN and used in Wysoski et al. (2006) and Wysoski (2008) to perform audio-visual face recognition.

Similar to other SNN approaches, a specific neural model, a learning method, a network architecture and an encoding from real values into spike trains need to be defined in an eSNN model. The neural model is given by the dynamics of the post-synaptic potential $u_i(t)$ of a neuron i:

$$u_i(t) = \begin{cases} 0 & \text{if fired} \\ \sum_{j \mid f(j) < t} w_{ji}\, m_i^{order(j)} & \text{else} \end{cases} \quad (1)$$

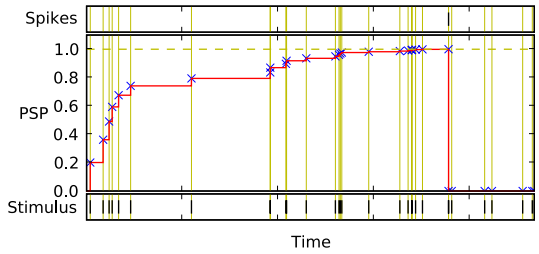


Fig. 1. Evolution of the post-synaptic potential (PSP) of a neural model used in QiSNN for a given input stimulus. If the potential reaches threshold θ a spike is triggered and the PSP set to 0 for the rest of the simulation, even if the neuron is still stimulated by incoming spike trains.

Algorithm 1 Training an Evolving Spiking Neural Network

Require: $m_l \in (0, 1)$, $s_l \in (0, 1)$, $c_l \in (0, 1)$, $l \in L$
1: initialize neuron repository $R_l = \{\}$
2: **for all** samples $X^{(i)}$ belonging to class l **do**
3: $w_j^{(i)} \leftarrow (m_l)^{order(j)}$, $\forall j \mid j$ pre-synaptic neuron of i
4: $u_{\max}^{(i)} \leftarrow \sum_j w_j^{(i)} (m_l)^{order(j)}$
5: $\theta^{(i)} \leftarrow c_l u_{\max}^{(i)}$
6: **if** $\min(d(w^{(i)}, w^{(n)})) > s_l$, $w^{(n)} \in R_l$ **then**
7: $w^{(n)} \leftarrow$ merge $w^{(i)}$ and $w^{(n)}$
8: $\theta^{(n)} \leftarrow$ merge $\theta^{(i)}$ and $\theta^{(n)}$
9: **else**
10: $R_l \leftarrow R_l \cup \{w^{(i)}\}$
11: **end if**
12: **end for**

where w_{ji} is the weight of a pre-synaptic neuron j, $f(j)$ the firing time of j, and $m_i \in (0, 1)$ a parameter of the model, namely the modulation factor. Function $order(j)$ represents the rank of the spike emitted by neuron j. For example, a rank $order(j) = 0$ would be assigned, if neuron j is the first among all pre-synaptic neurons that emits a spike. In a similar fashion the spikes of all pre-synaptic neurons are ranked and then used in the computation of u_i. A neuron i fires a spike when its potential has reached a certain threshold θ. After emitting a spike, the potential is reset to $u_i = 0$. Each neuron is allowed to emit only a single spike at most. The threshold $\theta = c\, u_{\max}$ is set to a fraction $c \in (0, 1)$ of the maximum potential $u_{\max}$ possible by a neuron. In Fig. 1 the change of the post-synaptic potential for this neural model is presented, when a series of input spikes (stimuli) are presented to the different synapses of this neuron.

An evolving neural network architecture using the above model along with a learning algorithm was proposed in Wysoski et al. (2006, 2008). The method successively creates a repository of trained output neurons during the presentation of training samples. For each training sample a new neuron is trained and then compared to the ones already stored in the repository. If a trained neuron is considered to be too similar (in terms of its weight vector) to the ones in the repository (according to a specified similarity threshold s), the neuron will be merged with the most similar one. Otherwise, the trained neuron is added to the repository as a new output neuron. The merging is implemented as the (running) average of the connection weights, and the (running) average of the two firing threshold. Because of the incremental evolution of output neurons it is possible to accumulate knowledge as it becomes available. Hence a trained network is able to learn new data without the need of re-training already learned samples. The procedure is described in detail in Algorithm 1.

The encoding of input values seems to be a critical factor in all SNN approaches. Several encoding mechanisms for SNN have been proposed, such as frequency mappings, Poisson processes

and rank order encoding. Another approach is the population encoding which distributes a single input value to multiple neurons and hence may cause the excitation and firing of several responding neurons. Our implementation is based on arrays of receptive fields as described in Bohte et al. (2002). Receptive fields allow the encoding of continuous values by using a collection of neurons with overlapping sensitivity profiles. Each input variable is encoded independently by a group of M one dimensional receptive fields. For a variable n an interval $[I^n_{\min}, I^n_{\max}]$ is defined. The Gaussian receptive field of neuron i is given by its center $\mu_i = I^n_{\min} + (2i-3)/2 * (I^n_{\max} - I^n_{\min})/(M-2)$ and width $\sigma = 1/\beta(I^n_{\max} - I^n_{\min})/(M-2)$, with $1 \leq \beta \leq 2$. Parameter β directly controls the width of each Gaussian receptive field.

2.2. Wrapper approach

The eSNN may be used to address feature subset selection (FSS) problems following the well known wrapper approach. A wrapper contains a general optimization algorithm interacting with an induction method (classifier), also *cf.* Fig. 2. The optimization task consists in a proper identification of an optimal feature subset, which maximizes the classification accuracy determined by the inductor. An eSNN operates here as the induction method, while, due to its interesting properties in terms of solution quality and convergence speed, the previously proposed Versatile Quantum-inspired Evolutionary Algorithm (vQEA) (Defoin-Platel et al., 2007) was used as the optimization algorithm. A vQEA evolves in parallel a number of independent probability vectors, which interact at certain intervals with each other, forming a multi-model Estimation of Distribution Algorithm (EDA) (Defoin-Platel et al., 2009). The binary nature of vQEA fits well to the feature selection problem we want to apply it on.

2.3. Integrated feature and parameter optimization

Manual fine-tuning the neuronal parameters can quickly become a challenging task (Wysoski, 2008). An alternative proposed in Valko, Marques, and Castelani (2005), is to optimize both the set of features and the neuronal parameters, in a simultaneous way. The selection of the fitness function was identified to be a crucial step for the successful application of such an embedded approach. In the early phase of the optimization the parameters are selected randomly. As a result it is very likely that a setting is selected for which the classifier is unable to respond to any input presented, which corresponds to flat areas in the fitness landscape. Hence a configuration that will allow the network to fire (even if not correctly) represents a huge (local) attractor in the search space, which could be difficult to escape in later iterations of the search. In Valko et al. (2005) a linear combination of several sub-criteria was used to avoid a too rugged fitness landscape. Nevertheless we can not confirm that the use of much simpler fitness functions led to any problems in our experiments. Using the classification accuracy on testing samples seemed to work well as it is presented in this and previous papers. All parameters, namely modulation factor m_l, similarity threshold s_l, potential fraction c_l, $\forall l \in L$ of eSNN were included in the search space of the optimization method.

Due to its binary nature, vQEA required the conversion of bit strings into real values. It was experimentally shown that a small number of Gray-coded bits seemed sufficient to approximate meaningful parameter configurations of the eSNN method. Nevertheless the use of a binary optimizer for a real-valued search space appears unsatisfactory. Each real-valued parameter needs to be encoded by a number of bits. For the mapping of bit strings into a real value additional computational resources are necessary. Furthermore a granularity is introduced into the search interval.

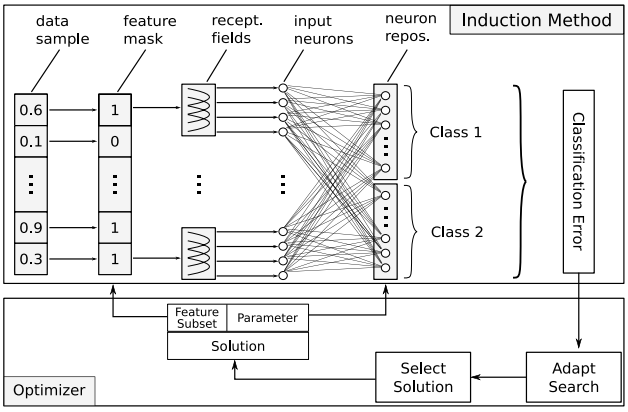


Fig. 2. The QiSNN framework of tightly coupled feature selection and parameter optimization of eSNN, integrated with the data.

Since a single continuous variable is represented by many bits, a binary optimization method has to operate on more variables, compared to a continuous optimizer. Thus scaling problems can be expected for binary optimization, especially in the context of high-dimensional problems, that need a precise optimization of real-valued search variables. Furthermore neighboring solutions in the continuous domain might not be neighbors in their binary representation. Exploring the local neighborhood of a solution may require the optimizer to flip many bits at the same time, which will encourage premature convergence and promote the phenomenon of hitch-hiking. Hence we will extend vQEA in this study towards continuous search spaces and use a combined representation for the simultaneous exploration of a binary landscape and the continuous landscape.

The complete QiSNN framework used in this study is summarized in Fig. 2.

3. Extending vQEA for continuous optimization

Based on the vQEA (Defoin-Platel et al., 2007), we propose here an extension of the algorithm to allow exploration of continuous search spaces. vQEA has been developed as a binary optimization method, which employs some quantum computing principles to enhance classical evolutionary algorithms. The method was studied in Defoin-Platel et al. (2009) and it was shown that vQEA belongs to the class of Estimation of Distribution Algorithms (EDA). The study revealed that the core of vQEA maintains a *multiple probabilistic model*, which is quite in contrast to other typical EDA, like *e.g.* Probabilistic Incremental Learning (PBIL), Univariate Marginal Distribution Algorithm (UMDA), and compact Genetic Algorithm (cGA), see Lozano, Larra naga, Inza, and Bengoetxea (2006) for an excellent overview of these algorithms. In these methods only a *single* probabilistic model is evolved during the optimization process. In vQEA each model explores the search space independently, but it may exchange information at pre-defined intervals with the other models. The algorithm is population based and each individual manages its own probabilistic model. The individuals itself are organized in groups, hence introducing an important structure into the population.

Several advantages of this multi-model and the structured population have been identified. vQEA uses an implicit adaptive learning rate, which makes it robust to its parameter configuration. It was shown that a certain parameter setting is suitable for a variety of problem classes and sizes. Furthermore, the multi-model approach allows a finite number of decision errors, which makes vQEA robust against fitness noise. It was demonstrated that vQEA performs better in terms of speed and solution quality than other first-level EDA, especially when links are introduced between

variables (epistasis). The method was compared to a number of Evolutionary Algorithms on several different benchmark problems. Finally, using several probabilistic models allows a more diverse exploration of the search space than just using a single one.

In the following section we extend the binary multi-model EDA (*i.e.* vQEA) towards the area of continuous search spaces. Since all key characteristics of vQEA will be still present in the proposed algorithm, we expect similar advantages of this method in comparison to other evolutionary methods, such as Genetic Algorithm (GA), Particle Swarm Optimization (PSO) and Differential Evolution (D/E).

3.1. Continuous-value multi-model EDA

The probabilistic model in vQEA is based on a Bernoulli random variable for each bit, which is referred to as a qbit according to the used quantum computing metaphor. Sampling from such a string of qbits results in the creation of a bit string, which in turn can be evaluated by the corresponding fitness function. Since we want to consider continuous search spaces now, we have to replace the Bernoulli distribution by a continuous one, such that it becomes possible to sample real values instead of discrete ones. A number of approaches concerning how to employ such distributions and how to model them have been studied in literature. Generally they are based on Gaussian distributions (Bosman & Thierens, 2000; Gallagher & Frean, 2005; Gallagher, Frean, & Downs, 1999; Mininno, Cupertino, & Naso, 2008; Yuan & Gallagher, 2003), histograms (Yuan & Gallagher, 2003), or interval representations (Servet, Travé-Massuyès, & Stern, 1998).

We consider a continuous EDA based on Gaussian distributions here. For each dimension j of the continuous search space and for each probabilistic model i, a random variable following a Gaussian distribution is evolved. Therefore the distribution is fully described by two parameters: The mean $\mu_i^{(j)}$ and the standard deviation $\sigma_i^{(j)}$. In each generation samples are drawn forming real-valued vectors, whose quality can be evaluated by the fitness measure. An update rule is then applied to update $\mu_i^{(j)}$ and $\sigma_i^{(j)}$ to move the search towards promising areas in the search space, making higher quality solutions more likely to be sampled in the next generation. We will first describe the basic structure of algorithm in detail, followed by the presentation of the chosen update rule.

The overall structure of the proposed extension is almost identical to vQEA. Like vQEA also the continuous version is a population-based search method. Its behavior can be decomposed in three different interacting levels: Individual, group and population level.

Individuals. The lowest level corresponds to *individuals*. An individual i at generation t contains a probabilistic model $P_i(t)$ and two real-valued strings $C_i(t)$ and $A_i(t)$. More precisely P_i corresponds to a string of N pairs of values $(\mu_i^{(j)}, \sigma_i^{(j)})$:

$$P_i = P_i^1 \dots P_i^N = \begin{bmatrix} \mu_i^{(1)} & \dots & \mu_i^{(N)} \\ \sigma_i^{(1)} & \dots & \sigma_i^{(N)} \end{bmatrix}. \quad (2)$$

The pair $(\mu_i^{(j)}, \sigma_i^{(j)})$ corresponds to the parameters of the distribution of the jth variable of the ith probabilistic model. Each variable in P_i is sampled according to $\mu_i^{(j)}$ and $\sigma_i^{(j)}$, so that C_i represents a configuration in the search space whose quality can be determined using a fitness function f. In most continuous optimization problems, the variables have a specific domain of definition. Without loss of generality we assume each $c_i^{(j)} \in C_i$ to be defined in to the interval $[-1, 1]$. As a consequence, each $c_i^{(j)} \in C_i$ follows a *truncated normal distribution* in the range $[-1, 1]$. Truncated normals can be sampled using a simple numerical procedure and the technique is widely adopted in pseudo-random number generation, see *e.g.* Geweke (1991) for an efficient implementation.

To each individual i a solution A_i is attached acting as an attractor for P_i. Every generation C_i and A_i are compared in terms of their fitness. If A_i is better than C_i (*i.e.* $f(A_i) > f(C_i)$ assuming a maximization problem), an update operation is applied on the corresponding model P_i. The update will move the mean values of the probabilistic model P_i slightly towards the attractor A_i. The choice of a suitable model update operation is critical for the working of the algorithm. We will elaborate the details of the model update in Section 3.1.1.

The update policy of an attractor A_i can follow two distinctive strategies. In the original QEA (Han & Kim, 2002) an *elitist* update strategy was used, in which the attractor A_i is replaced by C_i only if C_i is better than A_i in terms of fitness. In a *non-elitist* update strategy (firstly introduced in Defoin-Platel et al. (2007)) C_i replaces A_i at every generation. The choice of the update policy has great consequences for the algorithm and changes its behavior completely. To emphasize the importance of the update rule the non-elitist version of QEA has been proposed as *Versatile QEA* (vQEA) as the attractors are able to change every generation and therefore demonstrate a very high volatility. Since no experimental condition could be identified that favored the elitist attractor update policy, we will concentrate on the non-elitist version during the course of this paper.

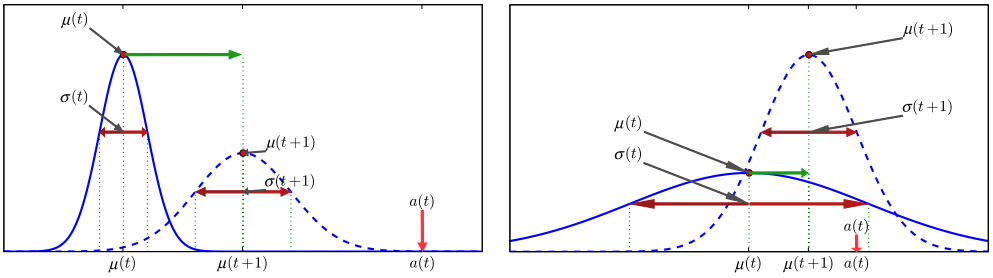
Groups. The second level corresponds to *groups*. The population is divided into g groups each containing k individuals having the ability of synchronizing their attractors. For that purpose, the best attractor (in terms of fitness) of a group, noted B_{group}, is stored at every generation and is periodically distributed to the group attractors. A parameter S_{local} is introduced, which controls the phase of local synchronization, *i.e.* a local synchronization event is triggered in every S_{local}-th generation.

Population. The set of all $p = g \times k$ individuals forms the *population* and defines the topmost level of the multi-model approach. As for the groups, the individuals of the population can synchronize their attractors, too. For that purpose, the best attractor (in terms of fitness) among all groups, noted B_{global}, is stored every generation and is periodically distributed to the group attractors. A parameter S_{global} is introduced, which controls the phase of global synchronization, *i.e.* a global synchronization event is triggered in every S_{global}-th generation.

3.1.1. Model update

The update of the probabilistic model is particularly interesting, since it governs how the search space is explored by the algorithm. Several continuous EDA have been proposed in literature (Bosman & Thierens, 2000; Mininno et al., 2008; Mühlenbein, Mahnig, & Rodriguez, 1999; Sebag & Ducoulombier, 1998; Yuan & Gallagher, 2003), along with a number of different update rules, *e.g.* Gallagher and Frean (2005) and Yuan and Gallagher (2003). The common principle of all these continuous EDA is based on the sampling of a population. In vQEA (and thus also its extension) the situation is very different, since only a *single* solution (for each probabilistic model) is sampled in every iteration. Hence the model update can not rely on the density of a population, but has to use a single attractor instead to perform the desired update.

We formulate here an appropriate update rule for the probabilistic models. Updating the mean $\mu^{(j)}$ in the Gaussian variable j appears to be straight-forward. We adopt a mean shift towards the value of the current attractor $a^{(j)}$ at location j, which is quite similar to the mean update used in methods mentioned



(a) Update operation for distant attractors.

(b) Update operation for close attractors.

Fig. 3. The figure presents the update operation for a single Gaussian random variable. For each update the distance $d = a(t) - \mu(t)$ between the attractor $a(t)$ and the mean $\mu(t)$ of the Gaussian is computed at generation t. (a) If $d \geq \sigma(t)$ the attractor is considered distant. We interpret that situation by assuming that $\mu(t)$ does not represent a promising area in the search space. In this case the mean $\mu(t)$ is strongly shifted towards the attractor, while at the same time the standard deviation $\sigma(t)$ is increased to allow a wider search in the fitness landscape. (b) On the other hand, if the attractor is inside the boundaries defined by $\sigma(t)$, *i.e.* $d < \sigma(t)$, we assumed that $\mu(t)$ is already in a promising area of the search space. The algorithm starts to localize the search by shifting $\mu(t)$ only slightly towards the direction of the attractor, while decreasing $\sigma(t)$ at the same time.

above. Depending on the distance $d = a^{(j)} - \mu^{(j)}$, a shift $\Delta\mu^{(j)}$ is computed at generation t:

$$\Delta\mu^{(j)}(t) = \frac{2}{1 + e^{-5d}} - 1 \tag{3}$$

which is then used to perform the update:

$$\mu^{(j)}(t+1) = \mu^{(j)}(t) + \theta_\mu \Delta\mu^{(j)}(t). \tag{4}$$

In Eq. (4) a parameter θ_μ is introduced, which we will refer to as the learning rate of the mean. We note that θ_μ corresponds to the maximum mean shift in a single generation.

For the update of the standard deviation $\sigma^{(j)}$ we will exploit the idea that $\sigma^{(j)}$ should decrease whenever $\mu^{(j)}$ represents a "promising" area in the fitness landscape. We assume $\mu^{(j)}$ to be "fit" when $|d| < \sigma^{(j)}$. Thus, if the attractor $a^{(j)}$ is close to $\mu^{(j)}$ (within the boundaries defined by $\sigma^{(j)}$), the standard deviation $\sigma^{(j)}$ is decreased. It is noteworthy that solutions fulfilling this condition are more likely to be sampled, than other solutions, which means that on average $\sigma^{(j)}$ will decrease. Attractors that are more distant to $\mu^{(j)}$ and thus $|d| \geq \sigma^{(j)}$, will cause an increase of $\sigma^{(j)}$, since it can be assumed that $\mu^{(j)}$ does not represent a promising area in the landscape. We define the standard deviation shift $\Delta\sigma^{(j)}$ at generation t as:

$$\Delta\sigma^{(j)}(t) = \frac{1}{1 + e^{-10(\sigma^{(j)}(t) - 0.5)}} \tag{5}$$

and then use it to perform the update:

$$\sigma^{(j)}(t+1) = \begin{cases} \sigma^{(j)}(t) - \theta_\sigma \Delta\sigma^{(j)}(t) & \text{if } |d| < \sigma^{(j)} \\ \sigma^{(j)}(t) + \theta_\sigma \Delta\sigma^{(j)}(t) & \text{otherwise.} \end{cases} \tag{6}$$

In Eq. (6) a parameter θ_σ is used, which we will refer to as the learning rate of the standard deviation. Again we want to note that θ_σ corresponds to the maximum standard deviation shift in a single generation. In order to avoid divergent behavior of the algorithm, *i.e.* $\sigma^{(j)}$ increases indefinitely, we restrict the domain of $\sigma^{(j)}$ by defining upper and lower bounds, such that $\sigma_{\min} \leq \sigma^{(j)} \leq \sigma_{\max}$.

In Fig. 3 the principle of the update rule is summarized. Distant attractors (relative to the current mean of the PDF) result in a large mean shift, while at the same time the standard deviation is increased, *cf.* Fig. 3(a). For close attractors the mean shift is small and the standard deviation is decreased, *cf.* Fig. 3(b).

It is important to note, that the probabilistic update operator described above, is similar to the rotation gate used in QEA. As shown in Defoin-Platel et al. (2009) the size of an update step using the rotation gate depends on the convergence of the probabilistic model. This phenomenon was described as a form of deceleration of the algorithm before convergence. The sigmoid shape of the standard deviation update adopts a similar strategy, since also here the size of the shift $\Delta\sigma^{(j)}$ decreases with increasing convergence of the algorithm.

3.2. Combined search spaces

Many real-world problems require the exploration of combined search spaces: a binary and a continuous space. An example is the parallel evolution of the topology and the weight matrix of a neural network. Here the topology is encoded as a bit string, where "1" represents a present connection between two neurons and "0" encodes its absence. Another example is the wrapper based feature selection, where the presence/absence of a feature requires a binary search space, while appropriate configurations for the classification method may correspond to a continuous landscape.

It is now possible to employ vQEA on combined search spaces with two types of representation. Each representation uses its corresponding update operator to drive the probabilistic model towards promising areas in the search space. In every generation the models are sampled and then evaluated by a single fitness measure. The fitness evaluation uses the sampled binary and continuous solution part to determine the quality of the combined solution. According to the fitness of the obtained solution the models are updated. This extended vQEA allows us to enhance the original QiSNN.

We emphasize that the extended vQEA is similar to a collaborative coevolutionary algorithm (Potter & Jong, 2000). The evolution of the two representations proceeds more or less independently from each other. Both use their own solution representations and update operators and may explore their search space with different learning rates. Despite their independent evolution, both representations share a single fitness function. The binary and continuous sub-solutions are the components of a combined solution, and both parts need to collaborate in order to maximize their fitness.

4. Experiments

We study the enhanced version of QiSNN on two benchmark problems. The first benchmark is referred to as the two-spiral-problem, on which the original QiSNN was investigated before, *cf.* Schliebs et al. (2009). This problem is composed of two-dimensional data forming two intertwined spirals and was firstly introduced in Lang and Witbrock (1988). It requires learning of a highly non-linear separation of the input space. The data was frequently used as a benchmark for neural networks, including the analysis of the eSNN method itself (Schliebs et al., 2009; Wysoski, 2008). Since the data contains only two relevant dimensions it was extended by adding redundant and random information. The importance of the redundant features was varied: Features range from mere copies of the original two spirals to completely random ones. The inherent information of a feature decreases when stronger noise is applied. A detailed description of the data

generation can be found in Schliebs et al. (2009). The dataset contains seven redundant two-dimensional spiral points x'_i, y'_i, for each point a different noise strength is used, totalling in 14 redundant features. Additionally four random features $r_1, \ldots, r_4$ were included. Together with the two relevant features of the spirals (x and y) the dataset contained 20 features in 400 samples.

The second benchmark is the uniform hypercube dataset, to our best knowledge firstly introduced in Estevez, Tesmer, Perez, and Zurada (2009). The problem consists of two classes of 400 samples. For each sample a five-dimensional vector $(r_1, \ldots, r_5)$ is drawn from a uniform distribution. A given pattern belongs to class 1 if $r_i < \gamma^{i-1} * \alpha$ for $i = 1, \ldots, 5$ and to class 2 otherwise. The parameters were chosen to be $\gamma = 0.8$ and $\alpha = 0.5$. The entire dataset consists of 40 features, five relevant, 30 random and five redundant ones. The latter are linear combinations of the relevant features perturbated by additive Gaussian noise of increasing level. The dataset was balanced. A more detailed explanation of the data generation can be found in Estevez et al. (2009).

4.1. Setup

For the combined optimization method we chose a population structure of ten individuals organized in a single group, which is globally synchronized every generation. This setting was reported to be generally superior for a number of different benchmark problems (Defoin-Platel et al., 2009). In the case of the spiral dataset the learning rate for the binary rotation gate was set to $\theta = \pi/50$. For the rate of the mean and standard deviation shift we chose $\theta_\mu = 0.1$ and $\theta_\sigma = 0.1$ respectively. The algorithm was allowed to evolve over a total number of 400 generations. Due to its larger problem size 1000 generations were computed for the hypercube problem, using once more $\theta = \pi/50$ for the binary learning rate and $\theta_\mu = 0.1$ and $\theta_\sigma = 0.05$ for the continuous update operator.

In order to allow a fair comparison between the classification methods used in this study, one has to decide for an appropriate parameter configurations for each classifier. In contrast to the other classifiers, NBC does not require the tuning of any parameters. To setup the MLP we have experimented on a subset of the datasets containing the relevant features only. By changing the number of hidden neurons, the learning rate, and the momentum term a satisfying configuration, in terms of classification accuracy, was experimentally obtained by systematic trial and error. The results of the parameter study for the MLP on the spiral dataset are presented in Fig. 4. The finally chosen setting is based on a tradeoff between computational cost and classification accuracy. The additional cost of more hidden neurons is not worth the slight increase of accuracy reported in Fig. 4. Using 10-fold cross-validation the chosen configuration of MLP achieved a satisfying accuracy of 0.849 (standard deviation 0.0634) on the spiral dataset containing the two relevant features only. When applied to the full dataset using all 20 features, the same configuration resulted in an accuracy of 0.611 (0.0608). Thus, appropriate feature selection does improve the performance of MLP, which is the key principle exploited in the wrapper approach. Finding an appropriate setting for the spiral problem appeared to be more difficult, in contrast to the other benchmark. For the latter problem changes in the configuration did not seem to impact the performance of the classifier too much. Thus we decided to use the same parameter setting for both problems. The common error back-propagation learning algorithm was used to train the network, connection weights were initialized with small uniform random numbers in the range $[-0.25, 0.25]$.

Most of the parameters of QiSNN are optimized during the evolutionary process. For each class $l \in L$ three parameters exist: The modulation factor m_l, the similarity threshold s_l, and the proportion factor c_l. Since both problems contain two classes, six parameters are involved in the QiSNN framework used here. In terms of the population encoding we found especially that the number of receptive fields needs careful consideration, since it affects the resolution for distinguishing between different input variables. After some preliminary experiments we decided for 20 receptive fields in case of the spiral data and five receptive fields for the hypercube. The Gaussian centers were uniformly distributed over the search interval and the variance was set to $\beta = 1.5$.

Fig. 4. The figure shows the accuracy levels achieved by 32 different configurations of a multi-layer perceptron on the two-spiral dataset. Each point represents the average of the accuracies obtained in a 10-fold cross-validation experiment, error bars indicate the standard deviation. All configurations use neurons with sigmoid transfer functions, trained in 500 epochs. The lower curve (green triangles) represents the accuracy of the MLP when all 20 features are included in the dataset, the upper curve (black squares) the accuracy when only the relevant features are used. The circles (red) indicate the finally chosen configuration for the experiments performed in this study, which is a satisfying compromise between computational cost and classification quality.

In order to guarantee statistical relevance, 30 independent runs for each investigated classification method were performed. In every generation all samples of the dataset were randomly shuffled and divided into training and testing samples, according to a ratio of 0.75. For the computation of the classification error we determined the ratio between correctly classified samples and the total number of testing samples.

4.2. Results

We discuss the results on the two-spiral problem first, *cf.* Fig. 5. Fig. 5(a)–(d) present the evolution of the average best feature subset in every generation using the enhanced and original QiSNN, MLP and NBC respectively. The color of a point in these diagrams reflects how often a specific feature was selected at a certain generation: The lighter the color the more often the corresponding feature was selected. It can clearly be seen that independent of the used algorithm a large number of features has been discarded during the evolutionary process. Furthermore all algorithms clearly identify the features x and y to be relevant. All methods except the enhanced QiSNN select some redundant and/or irrelevant features, too.

Particularly interesting is the order in which the features have been discarded by each algorithm. Both QiSNN (Fig. 5(a) and (d)) rejected the four random features $r_1, \ldots, r_4$ containing no information almost immediately in less than 20 generations. The redundant features x'_i, y'_i were then rejected one after the other, according to the strength of the inherent noise: The higher the noise, the earlier a feature is identified as irrelevant. We note the improved performance of the enhanced QiSNN, which is clearly able to reject all redundant features in most of the runs. Fig. 5(e) compares the evolution of the number of selected features during