Using particle swarms for the development of QSAR models based on K-nearest neighbor and kernel regression

Walter Cedeño* & Dimitris K. Agrafiotis
3-Dimensional Pharmaceuticals, Inc., 665 Stockton Drive, Exton, PA 19341, USA.

Summary

We describe the application of particle swarms for the development of quantitative structure-activity relationship (QSAR) models based on k-nearest neighbor and kernel regression. Particle swarms is a population-based stochastic search method based on the principles of social interaction. Each individual explores the feature space guided by its previous success and that of its neighbors. Success is measured using leave-one-out (LOO) cross validation on the resulting model as determined by k-nearest neighbor kernel regression. The technique is shown to compare favorably to simulated annealing using three classical data sets from the QSAR literature.

Introduction

The increasing amount of information available in digital form has prompted the development of novel data mining methodologies that can process and interpret large volumes of data faster and with greater reliability. Artificial intelligence methods, such as artificial neural networks (ANN) [1], classification and regression trees (CART)[2], and k-nearest neighbor classifiers (KNN) [3], have been used extensively for this purpose [4, 5]. These methods are widely employed in computer-assisted drug design to correlate some measure of biological activity with a set of physicochemical, structural and/or electronic parameters (descriptors) of the compounds under investigation. It is assumed that the biological activity of a compound is related to its chemical structure, and can therefore be inferred from a carefully chosen set of descriptors. Since it is not possible to know in advance which molecular descriptors are most relevant to the problem at hand, a comprehensive set is usually employed, chosen based on experience, software availability, and computational cost.

However, is well known, both in the chemical and statistical fields, that the number of features used in a QSAR model can greatly affect its accuracy. The presence of noise and irrelevant or redundant features can cause the method to adapt to the idiosyncrasies of the individual samples and lose sight of the broad picture that is essential for generalization beyond the training set [6]. This problem is compounded when the number of observations is also relatively small, as is often the case in molecular design. If the number of variables is comparable to the number of training patterns, the parameters repeated. The large number of available descriptors also increases the risk of chance correlations [7]. Comparing the results against randomly generated results is commonly used to verify that this risk is sufficiently low and the model is reliable.

Feature selection is often used to remedy this situation and improve the accuracy of a classification or regression technique. Feature selection works by identifying a small subset of necessary and sufficient features that can be used as input to the underlying predictor. Feature selection algorithms can be divided into three main categories [8]: (1) those where the selection is embedded within the basic regression algorithm, (2) those that use feature selection as a filter prior to regression, and (3) those that use feature selection as a

*To whom correspondence should be addressed. E-mail: walter.cedeno@3dp.com.

wrapper around the regression. The latter has a long history in statistics and pattern recognition, and is the method of choice for QSAR.

Feature selection can be viewed as a heuristic search, where each state in the search space represents a particular subset of the available features. In all but the simplest cases, an exhaustive search of the state space is impractical, since it involves $n!/(n-r)!r!$ possible combinations, where n is the total number of available features and r is the number of features selected. Several search algorithms have been applied to this problem, ranging from simple greedy approaches such as forward selection or backward elimination [9], to more elaborate methodologies such as simulated annealing (SA) [10], evolutionary programming (EP) [11], genetic algorithms (GA) [12–15], artificial ants (AA) [16], and more recently binary particle swarms (PS) [17].

Feature selection is a special case of a more general technique known as feature weighting [18, 19]. In feature weighting, each feature is associated with a weight value that indicates the contribution of that feature in the learning algorithm. Weights are usually confined to the interval [0, 1]. In feature selection, a weight can either be 1 and 0 to indicate whether the feature is used in the model or not. The aim of feature weighting is to find the relative importance of each feature. In some cases, using partial information from each feature has been shown to benefit the learning algorithm [20–22].

In this paper, we adapt the particle swarm search paradigm [23] to the problem of finding the optimal subset of feature weights for creating a QSAR model based on k-nearest neighbor and kernel regression [24, 25]. Particle swarms explore the state space through a population of individuals that move in a stochastic manner, influenced by their own memory as well as that of their peers. The location of each individual at any point in time represents a particular solution to the optimization problem. Although the particles move stochastically, they have a tendency to repeat past behaviors that have been rewarded by their environment, i.e., return towards promising states visited in the past either by themselves or by one of their neighbors. In the case at hand, a state represents a particular subset of features and their corresponding weights, and its fitness represents the quality of the KNN model derived from those weights, as measured by an appropriate crossvalidation statistic. KNN (or kernel regression) is a similarity-based approach that makes predictions by averaging the response values

of the k nearest neighbors to the query. To minimize the likelihood of over-fitting, the method described herein uses a combination of feature selection and feature weighting to construct the actual models. In the remaining sections, we provide the key algorithmic details of particle swarm optimization and kernel regression, discuss its advantages, and compare it to alternative methods based on simulated annealing using three classical data sets from the QSAR literature.

Methods

k-nearest neighbor and kernel regression

KNN is an intuitive method used extensively for classification. Given a pattern to classify, KNN works by selecting the k most similar patterns from a set of well-known classified data (training data) and choosing the class with the most representatives in the set. Similarity is typically measured by the Euclidean distance in some appropriate feature space or some other suitable metric. KNN is a lazy algorithm, i.e., it defers data processing until needed. The algorithm uses local information and adapts well to changes in the training data. Its main drawbacks are its susceptibility to noise and the curse of dimensionality. These can be alleviated using normalization and feature weighting to calculate the distance between patterns according to the equation:

$$d(p, q) = \sqrt{\sum_{i-1}^{n} w_i (p_i - q_i)^2} \qquad (1)$$

where n is the number of features, p_i and q_i are the i-th feature values for patterns p and q respectively, and w_i is the weight of the i-th feature. The implementation of KNN used in this work is based on k-d trees in order to reduce the computational cost associated with calculating distances [26].

Kernel regression is a closely related non-parametric technique that uses local information to obtain a prediction. The main difference from KNN is that the prediction for q is based on the weighted average of the response values of all the patterns p in the neighborhood of q. Usually, the kernel function gives more weight to points that are closer to q.

In this work, the patterns are chemical structures and the input features are a suitably chosen set of molecular descriptors. We use KNN to select the k most similar structures to a structure q and apply kernel

regression using the kernel function:

$$f(p, q) = \frac{1}{1 + d(p, q)} \qquad (2)$$

where $d(p,q)$ is the KNN distance as given in Equation 1. The prediction for q is obtained by:

$$y'(q) = \frac{\sum\limits_{i=1}^{k} y_i f(p_i, q)}{\sum\limits_{i=1}^{k} f(p_i, q)} \qquad (3)$$

where k is the number of nearest neighbors of q, p_i is the i-th nearest neighbor, y_i is the known response value of p_i, and $f(p, q)$ the kernel function given in Equation 2.

Particle swarms

Particle swarms is a relatively new optimization paradigm introduced by Kennedy and Eberhart [23]. The method is based on the observation that social interaction, which is believed to play a crucial role in human cognition, can serve as a valuable heuristic in identifying optimal solutions to difficult optimization problems. particle swarms explore the search space using a population of individuals, each with an individual, initially random, location and velocity vector. The particles then 'fly' over the state space, remembering the best solution encountered. Fitness is determined by an application-specific objective function $f(x)$. During each iteration, the velocity of each particle is adjusted based on its momentum and the influence of the best solutions encountered by itself and its neighbors. The particle then moves to a new position, and the process is repeated for a prescribed number of iterations. In the original PS implementation, the trajectory of each particle is governed by the equations:

$$\begin{aligned} v_i(t + 1) &= v_i(t) + \eta_1 \cdot r \cdot (p_i - x_i(t)) \\ &+ \eta_2 \cdot r \cdot (p_{b(i)} - x_i(t)) \end{aligned} \qquad (4)$$

and

$$x_i(t + 1) = x_i(t) + v_i(t + 1) \qquad (5)$$

where x_i and v_i are the current position and velocity of the i-th particle, p_i is the position of the best state visited by the i-th particle, $b(i)$ is the particle with the best fitness in the neighborhood of i, and t is the iteration number. The parameters η_1 and η_2 are called the cognitive and social learning rates, and determine the relative influence of the memory of the individual versus that of its neighborhood. In the psychological metaphor, the cognitive term represents the tendency of organisms to repeat past behaviors that have proven successful or have been reinforced by their environment, whereas the social term represents the tendency to emulate the successes of others, which is fundamental to human sociality. In effect, these terms introduce a tendency to sample regions of space that have demonstrated promise. r is a random number whose upper limit is a constant parameter of the system, and is used to introduce a stochastic element in the search process.

Kennedy defined four models of PS. The full model, which places equal influence to the cognitive and social influence, the social-only model, which involves no cognitive learning, the cognitive-only model, which has no social component, and the self-less model, which is a social-only model in which the individual is excluded from consideration in determining its neighborhood's best. The neighborhood represents a subset of the population surrounding a particular particle. The neighborhood size defines the extent of social interaction, and can range from the entire population to a small number of neighbors on either side of the particle (i.e. for the i-th particle, a neighborhood size of 3 would represent particles $i - 1$, i, and $i + 1$).

The present work employs Shi and Eberhart's[27,28] variant of the PS algorithm, which makes use of an inertia weight, ω, to dampen the velocities during the course of the simulation, and allow the swarm to converge with greater precision:

$$\begin{aligned} v_i(t + 1) &= \omega v_i(t) + \eta_1 \cdot r \cdot (p_i - x_i(t)) \\ &+ \eta_2 \cdot r \cdot (p_{b(i)} - x_i(t)) \end{aligned} \qquad (6)$$

Larger values of ω induce larger transitions and thus enable global exploration, whereas lower values facilitate local exploration and fine-tuning of the current search area.

The location vector for each particle corresponds to the feature weights used in KNN. The swarm searches for the best subset of features and corresponding weights that minimize the regression error in the training data. All weights are limited to values in the range [0, 1]. Location vectors that move outside this range are normalized at run time. In the present study, only a fixed number of features with the highest weight values were used to construct the actual models. Thus, whereas all feature weights are updated according to Equations 5 and 6, only the features with the highest weights are used in the KNN model.

Simulated annealing

Simulated annealing is a global, multivariate optimization technique based on the Metropolis Monte-Carlo search algorithm [29]. The method starts from an initial random state, and walks through the state space associated with the problem of interest by generating a series of small, stochastic steps. As with particle swarms, an objective function maps each state into a value that measures its energy or fitness. While downhill transitions are always accepted, uphill transitions are accepted with a probability that is inversely proportional to the energy difference between the two states. This probability is computed using Metropolis' acceptance criterion:

$$p = e^{-\Delta E / KT} \tag{7}$$

where K is a constant used for scaling purposes and T is an artificial temperature factor that controls the ability of the system to overcome energy barriers. The temperature is systematically adjusted during the simulation in a manner that gradually reduces the probability of high-energy transitions. In this study, we used a Gaussian cooling schedule with a half-width of 5 deviation units.

To circumvent the problem of assigning the appropriate value for K and to ensure that the transition probability is properly controlled, an adaptive approach is used. In this approach, K is not a true constant, but rather it is continuously adjusted during the course of the simulation based on a running estimate of the mean transition energy [30]. In particular, at the end of each transition, the mean transition energy is updated, and the value of K is adjusted so that the acceptance probability for a mean uphill transition at the final temperature is 0.1%. In general, schedules that involve more extensive sampling at lower temperatures seem to perform best, although it is also important that sufficient time must be spent at higher temperatures so that the algorithm does not get trapped into local minima.

In the problem at hand, a state represents the set of feature weights used in KNN. The objective is to minimize the regression error in the training data. Two types of steps were evaluated: (1) a feature is randomly selected and assigned a new random weight in the interval [0, 1], and 2) all feature weights are adjusted by a random value in the interval [−0.25, 0.25], followed by truncation if they exceed the specified boundaries [0, 1].

Table 1. Data set size and model parameters used.

Data set	Total samples	Total features	Selected features	Hidden neurons
AMA	31	53	3	3
BZ	57	42	6	2
PYR	74	27	6	2

Data sets

The methods were tested on three well-known data sets: antifilarial activity of antimycin analogues (AMA) [9, 11–13, 31], binding affinities of ligands to benzodiazepine/GABAA receptors (BZ) [13, 32], and inhibition of dihydrofolate reductase by pyrimidines (PYR) [33]. These data sets have been the subject of extensive QSAR studies, and have served as a test bed for many feature selection algorithms. To allow comparison with previous neural network-based approaches, the number of input features were taken from the literature. These details are summarized in Table 1. In all three cases, the descriptor data were normalized to [0, 1] prior to modeling.

Implementation details

All programs were implemented in the C++ programming language and are part of the DirectedDiversity® software suite [34]. All calculations were carried out on a Dell Inspiron 8100 laptop computer equipped with a 1.133 GHz Pentium IV Intel processor running Windows 2000 Professional.

Results and discussion

Our study was designed to: (1) establish a reasonable set of parameters for KNN and PS, (2) determine if PS offers any advantages over other search algorithms such as simulated annealing and random search (RS), and (3) determine whether KNN offers any advantages over the substantially more popular artificial neural networks. In particular, we compare the results to those obtained from a recent study of the relative performance of simulated annealing, particle swarms and artificial ant colonies using classical feed-forward neural networks. In the following discussion, each method is denoted by the abbreviations of the search engine and the regression technique, separated by a

hyphen (e.g., KNN-PS is used to indicate a *k*-nearest neighbor model derived by particle swarms).

Following common practice, the cross-validated correlation coefficient, R_{CV} resulting from leave-one-out (LOO) cross-validation was used to define the quality of the resulting models. This value is based on the training correlation coefficient R given by:

$$R = \frac{N \sum\limits_{i=1}^{N} y_i \tilde{y}_i - \sum\limits_{i=1}^{N} y_i \sum\limits_{i=1}^{N} \tilde{y}_i}{\sqrt{\left[N \sum\limits_{i=1}^{N} y_i^2 - \left(\sum\limits_{i=1}^{N} y_i \right)^2 \right] \left[N \sum\limits_{i=1}^{N} \tilde{y}_i^2 - \left(\sum\limits_{i=1}^{N} \tilde{y}_i \right)^2 \right]}} \quad (8)$$

The LOO cross-validation coefficient is obtained by systematically removing one of the patterns from the training set, building a model with the remaining cases, and predicting the activity of the removed case using the optimized weights. This is done for each pattern in the training set, and the resulting predictions are compared to the measured activities to determine their degree of correlation. Since in KNN the patterns in the training data do not participate in the prediction of their own response values (i.e., the method does not involve any training), the training R corresponds to the LOO crossvalidated R_{CV}.

We first investigated the effect of k, the neighborhood size in KNN, on the performance of the regression model. This was accomplished by varying k between 1 and 10 for each data set, keeping all other parameters fixed. The LOO cross-validation coefficient was averaged over 50 independent optimization runs, each starting from a different random number seed. The results for KNN-PS are illustrated in Figure 1. In general, a neighborhood size between 1 and 4 produced, on average, better models than larger values, which is consistent with prior experience. Small values for k are usually better, but the exact value may differ from one data set to another. Similar results were observed for the two techniques based on simulated annealing.

To assess the efficiency of KNN-PS, we compared it against random search (RS) and two different implementations of simulated annealing which differed only in the logic used to modify the current state (see Methods section). All four algorithms were limited to 10,000 objective function evaluations. Preliminary calculations suggested that the best results for particle swarms were obtained when the number of steps and population size were both set to 100. The results are summarized in Table 2.

Close inspection of these results leads to several interesting observations. First, the number of function evaluations that we chose for each run (10,000) allows us to compare the techniques in three different settings. Given the number of selected features, there are a total of 23,426, 5,245,786, and 296,010 possible feature subsets for AMA, BZ and PYR, respectively, which means that 10,000 states represent, at best, 42.7%, 0.2%, and 3.4% of all possible subsets for each of these data sets. This allows us to evaluate the performance of the algorithm with high (AMA), medium (PYR) and low (BZ) subset sampling rates, though in all cases the state space is substantially larger since the weights of the selected features need to be refined by the optimizer in order to obtain the optimum model.

For the AMA data set, only PS was able to separate itself from the other techniques. Not only did it find a better model, but also produced a completely different set of features and neighborhood size. The AMA results reveal the importance of the weight values for this particular data set. Although the difference between the best models found by SA1, SA2 and RS is small, slight differences between the weight vectors produced a better model for SA2. The differences between the weights are more noticeable for the PS model, where the values are 1.000, 0.526 and 0.372 for features 15, 31 and 5, respectively. A similar situation arises for the other two data sets as well. This suggests that PS is naturally proficient at working in continuous search spaces, which is, after all, the type of problem that the method was originally designed to address.

PS also obtained the best model for the BZ data set, though in this case there is a more notable difference between the models derived by the four methods. Three features, 2, 3 and 9, are present in the best models found by all four methods, whereas two additional features, 1 and 14, are shared by three of these models. Once again, the PS method seems to be more effective in adjusting the weights in order to obtain a better solution. The BZ data set seems to be the most challenging for all methods. The variation in the neighborhood size and features employed by the best models seems to suggest that better solutions might be obtained by increasing the number of function evaluations or by trying alternative kernel functions. On the contrary, for the PYR data set, all methods converged to the same subset of features (0, 4, 9, 10, 11 and 19) and neighborhood size, but PS produced a slightly better solution due to better optimization of the weights (Figure 2).
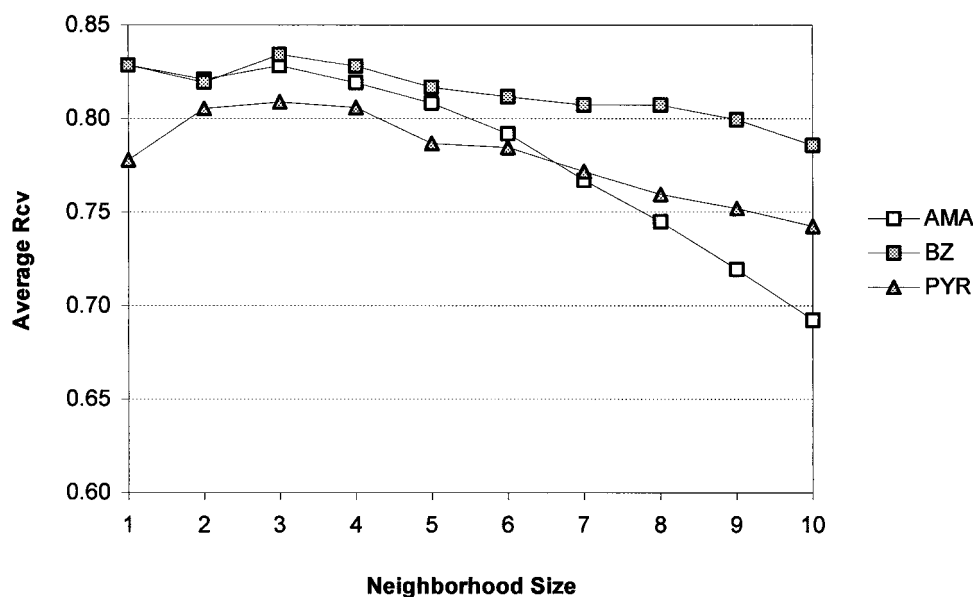
Figure 1. Effect of neighborhood size *k* on the generalization performance of KNN-PS.

A third goal of this work was to compare the results obtained by kernel regression to those obtained by neural networks. Since ANNs are unstable predictors, the learning (R) and generalization (R_{CV}) performance of the best ANN models discovered by each optimization technique were established by running several training and cross-validation experiments, each starting with a different random number seed, and averaging the respective statistics (50 runs were used to compute R, and 10 to compute R_{CV}). The same procedure was followed for all three data sets under investigation. A summary of the results for the ANN work can be found in Table 3.

Comparing the results obtained with ANN models (Table 3) to the present approach involving KNN and PS is not straightforward. The PS method described here does feature weighting, which is a far more complex problem than the feature selection employed with ANNs. The method not only selects the best features for the kernel model, but identifies their relative importance as well. Of course, one could argue that ANNs perform feature weighting implicitly by adjusting the synaptic weights during training, thus simplifying the selection task. Another difference has to do with the underlying regression techniques. ANN is a supervised learning technique and requires training prior to use, whereas KNN does not require training and relies on a prudent choice of the neighborhood size *k*, the kernel function, and the weight vector. The results in Tables 2 and 3 (summarized in

Figure 3) reveal that KNN-PS outperformed the ANN-based approaches for the AMA and PYR data sets, but did not do so for BZ.

Although these results do not allow any definitive conclusions to be drawn, they demonstrate the potential of KNN-PS for building QSAR models with good generalization power. Moreover, the method provides the researcher with a weight vector that can be used to determine the relative importance of each feature and, along with the inherent simplicity of the underlying model, which is based on molecular similarity, offer an additional level of interpretability that cannot be achieved with ANNs. On the other hand, KNN-based methods are computationally more intensive at prediction time, since they require a nearest neighbor search for every query structure. This problem is offset by substantial savings during the actual construction of the model, since there is no training involved and it is much easier to incorporate new data as they become available.

The implementation of KNN-PS that we used allows the selection of models with a predefined minimum and maximum number of features. Only features with weights above a given threshold (up to the predefined maximum) were chosen to be part of the model. Given this flexibility, we performed some experiments where we allowed solutions containing between 1 and 10 features with a threshold of 0.1. Although the results are not shown, models with up to 10 features produced slightly better LOO R_{CV} stat-
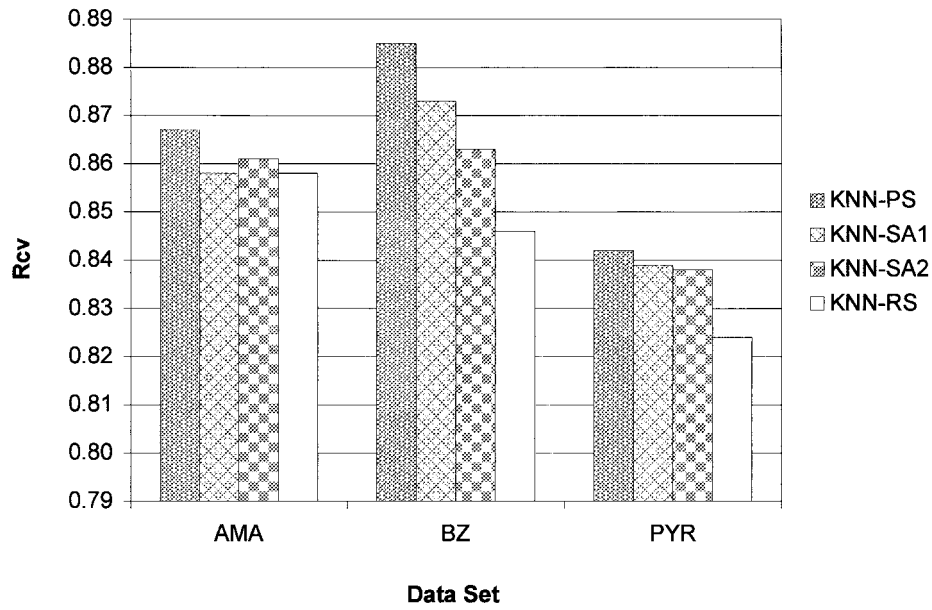
Figure 2. LOO cross-validation statistics for the best models discovered by KNN-PS, KNNSA1, KNN-SA2, and KNN-RS for the AMA, BZ, and PYR data sets.

Table 2. Top models selected by particle swarms (PS), simulated annealing (SA1 and SA2), and random search (RS) using *k*-nearest neighbor kernel regression.

Data set	Method	Selected variables	Variable weights	K	$\mu(R_{CV})$
AMA	PS	15, 31, 5	1.000, 0.526, 0.372	3	0.867
AMA	SA1	49, 18, 19	0.974, 0.937, 0.912	1	0.858
AMA	SA2	49, 19, 18	1.000, 1.000, 0.794	1	0.861
AMA	RS	18, 49, 19	0.983, 0.975, 0.922	1	0.858
BZ	PS	9, 3, 17, 1, 13, 2	1.000, 0.674, 0.539, 0.501, 0.498,	2	0.885
BZ	SA1	10, 27, 2, 9, 3, 14	0.962, 0.960, 0.952, 0.893, 0.777,	2	0.873
BZ	SA2	14, 3, 2, 9, 26, 1	1.000, 1.000, 1.000, 1.000, 0.679,	3	0.863
BZ	RS	22, 9, 3, 2, 14, 1	0.984, 0.979, 0.975, 0.899, 0.881,	4	0.846
PYR	PS	10, 4, 19, 0, 9, 11	1.000, 0.852, 0.806, 0.636, 0.596,	3	0.842
PYR	SA1	4, 10, 9, 19, 11, 0	0.995, 0.988, 0.984, 0.924, 0.825,	3	0.839
PYR	SA2	10, 4, 19, 9, 0, 11	1.000, 1.000, 0.916, 0.756, 0.727,	3	0.838
PYR	RS	17, 10, 19, 0, 4, 11	0.986, 0.982, 0.906, 0.832, 0.815,	3	0.824

Table 3. Top neural network models selected by the binary particle swarm (PS), simulated annealing (SA), and artificial ant algorithm (AA) for each of the three data sets.

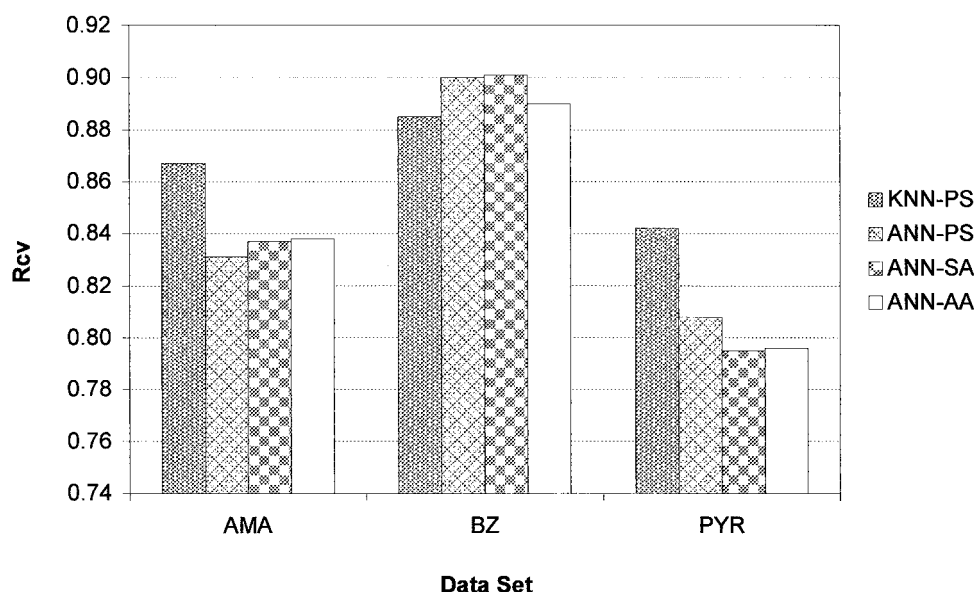| Data set	ANN-PS variables	$\mu(R_{CV})$	ANN-SA	$\mu(R_{CV})$	ANN-AA variables	$\mu(R_{CV})$
AMA	31,35,49	0.831	31,37,49	0.837	31,37,49	0.838
BZ	1,4,6,9,20,21	0.900	1,4,5,9,20,23	0.901	0,1,5,9,11,14	0.890
PYR	0,5,10,16,19,22	0.808	1,2,3,5,19,22	0.795	1,2,3,5,19,22	0.796

Figure 3. LOO cross-validation statistics for the best models discovered by KNN-PS, ANN-PS, ANN-SA and ANN-AA for the AMA, BZ and PYR data sets. For ANN-PS, ANN-SA and ANN-AA, the reported value represents the average over 10 independent crossvalidation runs.

istics. More experiments must be conducted in order to determine conclusively if the results are due to the inherent sampling efficiency of KNN-PS. One possible improvement is to include a bias term in the objective function in order to favor more parsimonious solutions involving small numbers of features.

The success demonstrated in the present study provides motivation for exploring further improvements of the technique. One obvious variation is to replace our simplistic kernel with a more sophisticated function. Another possibility is to improve the ability of particle swarms to create niches and converge to multiple solutions during the same run. This will improve the ability of the technique to detect and avoid local minima, and hopefully lead to better predictors.

Conclusions

A promising new algorithm based on particle swarms has been applied to the construction of QSAR models based on *k*-nearest neighbor and kernel regression. When compared to other search algorithms wrapped around the KNN learner, the PS method found better models for all three data sets used in the study. An advantage of PS lies in its ability to successfully adjust the weight values to improve the predictions of the model. When compared to neural networks, the method was able to identify models with better

generalization power as measured by leave-one-out cross-validation in two out of three data sets tested. The two main advantages of KNN over ANN are the ability to add more data to the model without retraining, and infer the relative importance of the selected features based on their respective weights, thus aiding in the interpretation of the model. Further empirical work is necessary to fully assess the effects of the other parameters, such as the kernel function, neighborhood and population size, etc. As in most published QSAR studies of this kind, the number and variety of data sets tested was small, and the true robustness of the algorithm can only be assessed by extensive validation with additional real-world examples.

Acknowledgements

References

1. Neural Networks in QSAR and drug design, Academic Press: New York, 1996.
2. Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J., *Classification and Regression Trees*. Wadsworth Intl. Group: Belmont, CA, 1984.
3. Parzen, E., Ann. Math. Stat. 33 (1962) 1065–1076.
4. van de Waterbeemd, H., Ed. *Chemometric Methods in Drug Design*, VCH: Weinheim, 1995, Vol. 2.
5. Hansch, L., Leo, C., In American Chemical Society: Washington, DC, 1995.
6. Manallack, D.T., Ellis, D.D., Livingston, D.J.J., Med. Chem., 37 (1994) 3758–3767.
7. Topliss, J.G., Edwards, R.P.J., Med. Chem., 22 (1979) 1238–1244.
8. John, G., Kohavi, R., Pfleger, J. In Machine Learning: Proceedings of the 11-th International Conference, Morgan-Kaufmann, 1994, pp. 121–129.
9. Selwood, D.L., Livingston, D.J., Comley, J.C.W., O'Dowd, A.B., Hudson, A.T., Jackson, P., Jandu, K.S., Rose, V.S., Stables, J.N.J., Med. Chem., 33 (1990) 136–142.
10. Sutter, J.M., Dixon, S.L., Jurs, P.C.J., Chem. Inf. Comput. Sci., 35 (1995) 77–84.
11. Luke, B.T.J., Chem. Inf. Comput. Sci., 34 (1994) 1279–1287.
12. Rogers, D.R., Hopfinger, A.J.J., Chem. Inf. Comput. Sci., 34 (1994) 854–866.
13. So, S.-S., Karplus, M.J., Med. Chem., 38 (1996) 5246–5256.
14. Yasri, A., Hartsough, D.J., Chem. Inf. Comput. Sci., 41 (2001) 1218–1227.
15. Hasegawa, K., Miyashita, Y., Funatsu, K.J., Chem. Inf. Comput. Sci., 37 (1997) 306–310.
16. Izrailev, S., Agrafiotis, D.K., SAR and QSAR in Environ. Res., 2001.
17. Agrafiotis, D.K., Cedeno, W.J., Med. Chem., 45 (2002) 1098–1107.
18. Wettschereck, D., Aha, D.W., Mohri, T., Artif. Intell. Review 11 (1997) 273–314.
19. Aha, D.W., In Liu, H., Motoda, H., Eds. *Feature extraction, construction and selection: a data mining perspective*, Kluwer: Norwell, MA, 1998.
20. Kohavi, R., Langley, P., Yun, Y. In European Conference on Machine Learning (ECML97), 1997.
21. Reymer, M.L., Punch, W.F., Goodman, E.D., Kuhn, L.A., Jain, A.K., IEEE Trans. Evol. Comput., 4 (2000) 164–171.
22. Komosinski, M., Krawiec, K., Artif. Intel. Med., 19, 25–38.
23. Kennedy, J., Eberhart, R. In *IEEE International Conference on Neural Networks*, IEEE Service Center: Perth, Australia, 1995, pp. 1942–1948.
24. Nadaraya, E.A. Theory of Probability and its Applications, 9 (1964) 141–142.
25. Watson, G.S. Sankhya – Ind. J. Stat. A, 26, 359–372.
26. Agrafiotis, D.K.J., Chem. Inf. Comput. Sci., 39 (1999) 51–58.
27. Shi, Y.H., Eberhart, R. In IEEE International Conference on Evolutionary Computation: Anchorage, AL, 1998.
28. Shi, Y.H., Eberhart, R. In 7th. Annual Conference on Evolutionary Programming: San Diego, CA, 1998.
29. Kirkpatrik, S., Gelatt, C.D.J., Vecchi, M.P., Science, 220 (1983) 671.
30. Agrafiotis, D.K.J., Chem. Inf. Comput. Sci., 37 (1997) 841–851.
31. Wikel, J.H., Dow, E.R., Bioorg. Med. Chem. Lett., 3 (1993) 645–651.
32. Maddalena, D.J., Johnson, G.A.R.J., Med. Chem. 1995, 38.
33. Hirst, J.D., King, R.D., Sternberg, M.J.E.J., Comput.-Aided Mol. Design, 8 (1994) 405–420.
34. Agrafiotis, D.K., Bone, R.F., Salemme, F.R., Soll, R.M. United States, 1995.