Initializing RBF-Networks with Small Subsets of Training Examples

Miroslav Kubat and Martin Cooperson, Jr.
Center for Advanced Computer Studies
University of Southwestern Louisiana, Lafayette, LA 70504-4330
{mkubat,mxc6421}@cacs.usl.edu

Abstract

An important research issue in RBF networks is how to determine the gaussian centers of the radial-basis functions. We investigate a technique that identifies these centers with carefully selected training examples, with the objective to minimize the network's size. The essence is to select three very small subsets rather than one larger subset whose size would exceed the size of the three small subsets unified. The subsets complement each other in the sense that when used by a nearest-neighbor classifier, each of them incurs errors in a different part of the instance space. The paper describes the example-selection algorithm and shows, experimentally, its merits in the design of RBF networks.

Introduction

Radial-basis-function (RBF) networks, such as the one depicted in Figure 1, are used to approximate functions $f : R^m \rightarrow R^p$ by appropriate adjustments of the parameters in the formula $f_j(\mathbf{x}) = \Sigma_{ij} w_i \varphi_i(\mathbf{x})$, where $\mathbf{x} = (x_1, \ldots, x_n)$ is an input vector and each $\varphi_i(\mathbf{x}) = \exp - \frac{\|x - \mu_i\|^2}{2\sigma_i^2}$ is an RBF function. The parameters to be determined by learning are the gaussian centers μ_i, the variances σ_i^2, and the weights w_i. Since the weights are easy to determine (e.g. by linear regression) and the variances do not pose any major challenge, the main difficulty is presented by the centers μ_i. Existing methods associate these vectors with the gravity centers of data clusters (Moody and Darken, 1989; Musavi et al., 1992), with hyperrectangles defined on the instance space by decision-tree induction (Kubat, 1998), or with vectors determined by AI search techniques (Chen, Cowan, and Grant, 1991; Cheng and Lin, 1994). A method that adds one neuron at a time with subsequent tuning of the centers was developed by Fritzke (1993, 1994). Techniques to adjust the centers by learning have been studied also by Wettschereck and Dietterich (1992).

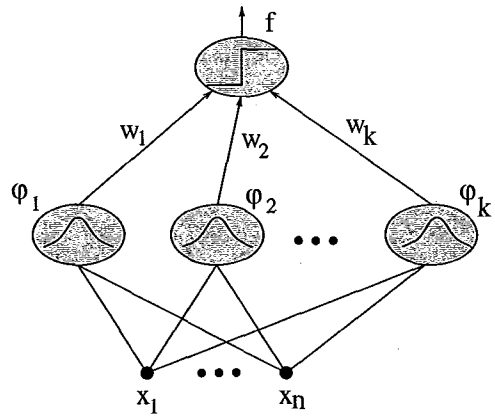

Figure 1: A Radial Basis Function Network

In this paper we focus on *2-class pattern-recognition* in which the network's output is reduced to a boolean variable, $f : R^n \rightarrow [-1, 1]$, and we explore the approach that associates each μ_i with one training example (Poggio and Girosi, 1990). Since turning *all* training examples into gaussian centers would create an unnecessarily large network, Lowe (1989) and Lee (1991) recommend selecting only a random subset. This is still unsatisfactory in applications where many examples are noisy, redundant, or non-representative. We surmise that choosing only those examples that have particular merit will make it possible to model the given pattern with a *smaller* network.

The search for representative examples has received attention among researchers that study *edited nearest-neighbor* classifiers, notably Hart (1968), Gates (1972), Wilson (1972), Ritter et al. (1975), Tomek (1976), and, more recently, Cortez and Vapnik (1995), and Wilson and Martinez (1997). Their algorithms can remove many noisy and redundant examples at the price of high computational costs—typically at least $O(mn^2)$, where m is the number of attributes and n is the number of examples. Moreover, these authors primarily concentrate on how to find "consistent" subsets that, when

used by a nearest-neighbor rule, give the same classifications as the original set. In the search for gaussian centers for RBF networks, consistency is not requried because the weights and non-linearities can make up for imperfections of the selected examples.

We suggest a novel technique that, at costs $O(mn)$, sorts out just a few representative training examples. The idea is that, rather than a single subset with N examples, we create three small subsets, S_1, S_2, and S_3, such that $|S_1 \cup S_2 \cup S_3| < N$. These subsets complement each other in the sense that, when employed by a 1-nearest-neighbor classifier, each of them tends to err in a different part of the instance space so that occasional mistakes of one classifier can be outvoted by the others.

When S_i's satisfying the complementarity condition have been found, their examples are pooled, $S = S_1 \cup S_2 \cup S_3$, and the elements of S are used as the gaussian centers $\boldsymbol{\mu_i}$'s. We hypothesize that the complementary behavior of S_i's ensures that the resulting RBF network will have favorable classification accuracy despite its small size. Whether this expectation is realistic will be examined by a series of simple experiments with synthetic as well as benchmark data. Prior to that, however, the next section explains the details of the example-selection algorithm.

Description of the Algorithm

The gaussian centers $\boldsymbol{\mu_i}$ are identified with carefully selected training examples in the subsets S_1, S_2, and S_3. Each S_i, when used by a 1-nearest-neighbor algorithm, defines a subclassifier C_i. The subsets S_i's are to be selected in a manner that ensures that each C_i will be prone to err on different examples. Here, our inspiration comes from the recent work on combining expert decisions (Vovk 1990; Breiman 1996) and, indirectly, from the essence of the boosting algorithm (Schapire, 1990).

Each S_i contains only very few training examples (say 2 or 3), and care is taken that both classes (positive and negative) are represented in each S_i. Applying C_i's to example $\mathbf{x}$ results in three class labels, and the final decision is achieved by *voting* of the triplet of C_i's. This will correctly classify any example that has been correctly classified by C_1 *and* C_2. Conversely, any example that has been misclassified by both C_1 and C_2 will be misclassified by the voting triplet regardless of the behavior of C_3. Therefore, C_3 should maximize its performance on those examples where C_1 and C_2 disagree. Let the error rates of C_1, C_2, and C_3 be denoted ε_1, ε_2, and ε_3, and let ε_k denote the percentage of examples misclassified by both C_1 and C_2. (Remember that the error rates of C_1 and C_2 will be measured on the entire training set, but ε_3 will be measured on those examples where C_1 and C_2 disagree.)

Figure 2 clarifies the notation and illustrates the fact that the error rate (P_E) of voting consists of ε_k, increased by the percentage of those examples on which C_3 is wrong while C_1 and C_2 disagree ($\varepsilon_1 + \varepsilon_2 - 2\varepsilon_k$):

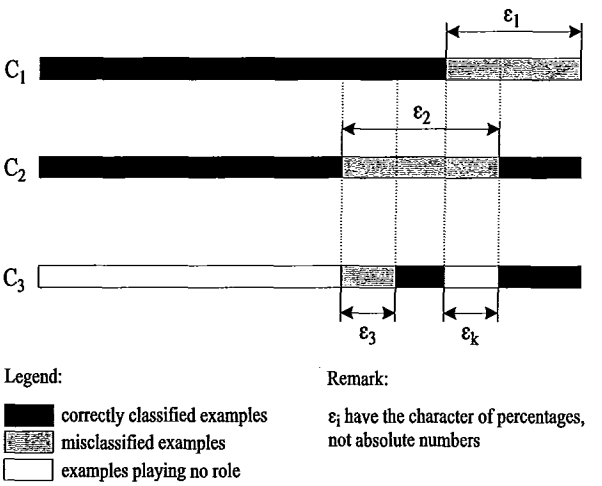

Figure 2: The behavior of the three classifiers

$$P_E = \varepsilon_k + \varepsilon_3(\varepsilon_1 + \varepsilon_2 - 2\varepsilon_k) \qquad (1)$$

In the event of $\varepsilon_1 = \varepsilon_2 = \varepsilon_3 = \varepsilon$, Equation 1 degenerates into $P_E = \varepsilon_k + \varepsilon(\varepsilon + \varepsilon - 2\varepsilon_k) = \varepsilon_k(1 - 2\varepsilon) + 2\varepsilon^2$. Then, $P_E \leq \varepsilon$ for any $\varepsilon_k < \varepsilon$, and the voting is guaranteed to outperform any C_i. The expression turns into $P_E = 2\varepsilon^2$ for $\varepsilon_k = 0$, and it turns into $P_E = \varepsilon$ for $\varepsilon_k = \varepsilon$. The best accuracy for $\varepsilon_1 = \varepsilon_2 = \varepsilon_3 = \varepsilon$ is therefore achieved when C_1 and C_2 never misclassify the same example ($\varepsilon_k = 0$). Conversely, P_E reaches its maximum when $\varepsilon_k = \varepsilon$.

In realistic applications, $\varepsilon_k = 0$ is hard to achieve if $\varepsilon_1 = \varepsilon_2$. In the search for the most appropriate C_2, the condition $\varepsilon_k = 0$ can often be satisfied only at the price of significantly increased ε_2, which means that the trade-off between ε_2 and ε_k has to be considered. Suppose the learner has chosen some C_1, and then succeeded in finding C_2 such that ε_k is low, while ε_2 is high. What if another classifier, C_2', can be found such that $\varepsilon_2' < \varepsilon_2$, and $\varepsilon_k' > \varepsilon_k$? How to decide whether the improvement in ε_2 outweighs the loss in ε_k? A guideline is offered by the following lemma:

Lemma 1. *Given fixed ε_1 and ε_3, assume two candidates for the second classifier, C_2 and C_2', such that $\varepsilon_2' = \varepsilon_2 + \Delta\varepsilon_2$ and $\varepsilon_k' = \varepsilon_k + \Delta\varepsilon_k$. Define $\alpha = \frac{1-2\varepsilon_3}{\varepsilon_3}$. For the error rates, P_E and P_E', of C_2 and C_2', the following equivalence holds:*

$$P_E' < P_E \iff \Delta\varepsilon_2 < -\alpha\Delta\varepsilon_k$$

Proof. Equation 1 establishes for the overall error rate:

Table 1: The algorithm for the selection of example subsets, S_1, S_2 and S_3, that complement each other when used by a nearest-neighbor classifier.

1. Make K_1 choices of S_1, and then select the one that minimizes ε_1.
2. Select randomly S_2 and S_3 and establish the initial value of α.
3. Make K_2 choices of an alternative for S_2. Whenever $\Delta\varepsilon_2 < -\alpha\Delta\varepsilon_k$ is satisfied, replace S_2 with this new candidate.
4. Make K_3 choices of S_3, and select the one that minimizes ε_3. Unless a stopping criterion is satisfied, update $\alpha = (1 - 2\varepsilon_3)/\varepsilon_3$, and return to step 3.

$$\begin{aligned} P'_E &= \varepsilon'_k + \varepsilon_3(\varepsilon_1 + \varepsilon'_2 - 2\varepsilon'_k) \\ &= \varepsilon_k + \Delta\varepsilon_k + \varepsilon_3(\varepsilon_1 + \varepsilon_2 + \Delta\varepsilon_2 - 2\varepsilon_k - 2\Delta\varepsilon_k) \\ &= P_E + \Delta\varepsilon_k + \varepsilon_3(\Delta\varepsilon_2 - 2\varepsilon_k) \end{aligned}$$

From here, $P'_E < P_E$ iff $\Delta\varepsilon_k + \varepsilon_3(\Delta\varepsilon_2 - 2\Delta\varepsilon_k) < 0$. This inequality is satisfied whenever $\varepsilon_2 < -\Delta\varepsilon_k \frac{1-2\varepsilon_3}{\varepsilon_3} = -\alpha\Delta\varepsilon_k$.

Q.E.D.

For illustration, suppose that some initial choice for C_2 implies $\varepsilon_k = 0.05$ and $\varepsilon_2 = 0.35$, and let $\alpha = 3$. This gives $P_E = \alpha\varepsilon_k + \varepsilon_2 = 0.15 + 0.35 = 0.50$. Then, another classifier, C'_2, is found, entailing error rates $\varepsilon'_2 = 0.18$ and $\varepsilon'_k = 0.10$. According to the previous lemma, this new classifier is better ($P'_E < P_E$) because $\Delta\varepsilon_2 = -0.17$, which is less than $-\alpha\Delta\varepsilon_k = -3 \cdot 0.05 = -0.15$. Indeed, it is easy to verify that $P'_E = \alpha\varepsilon'_k + \varepsilon'_2 = 3 \cdot 0.10 + 0.18 = 0.48 < P_E$.

Suppose that C_1 and C_2 have been chosen. Then, C_3 is selected and the values of ε_3 and α determined. The learner will search for some C'_2 that satisfies the condition $\Delta\varepsilon_2 < -\alpha\Delta\varepsilon_k$. Lemma 1 guarantees that the replacement of C_2 by C'_2 reduces P_E. The search is repeated, with the error rate P_E gradually decreasing, until no C'_2 satisfying the condition of Lemma 1 can be found. The learner will then find a C_3 that miminizes ε_3. As a result, ε_3 might depart from its initial value, which chanes α. The program will return to the previous step to further adjust C_2, and the procedure is repeated several times, terminating when no reasonable improvement of P_E is observed.

The algorithm that employs this analysis for the choice of complementary instance-based classifiers is summarized in Table 1. The precise values of the K_i values were: $K_1 = 10, K_2 = 5$, and $K_3 = 10$. This means that in the first step, 10 different (random) subsets S_1 are considered, and the one with lowest ϵ_1 value is chosen. The program generates the initial S_2 and S_3 subsets (so as to calculate the initial value of α), and then cycles through steps 3 and 4 until some termination criterion is reached. In our experiments, we simply stopped the program after 5 visits at steps 3 and 4. In each of these visits, $K_2 = 5$ random candidate S_2 subsets were generated. Whenever the condition $\Delta\varepsilon_2 < -\alpha\Delta\varepsilon_k$, the current S_2 is replaced with the new candidate. The program then generates $K_3 = 10$ random candidates for S_3 and chooses the one that has the lowest ϵ_3.

The number of subsets considered by this algorithm is $K_1 + 5(K_2 + K_3) = 10 + 5(5 + 10) = 85$ which, together with the initial choices of S_2 and S_3, gives 87. Evaluation of each subset on the training set is linear in the number of examples. More precisely, if each S_i contains three examples, then the overall computational costs are upper bounded by $87 \times 3 \times n$, where n is the number of training examples. The real costs are somewhat lower because S_3 is evaluated only on those examples where the first two subclassifiers disagree.

Experiments

Our objective is to select a compact subset of the training examples that will be identified with the gaussian centers, $\boldsymbol{\mu_i}$, of a radial-basis function network. In our experiments, the weight vector $\mathbf{w}$ was obtained using the statistical linear-regression technique of pseudoinverse matrices (see, e.g., Duda and Hart 1973). More specifically, if $\mathbf{X}$ is a matrix whose each row contains the outputs of the RBF units, and $\mathbf{b}$ is the classification vector whose elements are 1 for positive examples and -1 for negative examples, then $\mathbf{w} = \mathbf{X^P b}$ where $\mathbf{X^P}$ is a pseudoinverse matrix of $\mathbf{X}$. The variance, σ_i^2, of the i-th RBF is set to $\sigma_i^2 = \sqrt{d}$ where d is the Euclidean distance of the center that is closest to $\boldsymbol{\mu_i}$.

Our question is whether our example-selection method can really lead to smaller RBF networks with still high classification accuracy. In the search for an answer, we experimented with simple synthetic data with known characteristics, and then proceeded to experiments with more realistic benchmark domains.

In the sequel, an RBF network that has been created from randomly selected examples will be referred to by the acronym RANDEX. A RBF network that has been created from examples selected by the algorithm described in the previous section will be referred to by the acronym SELEX.

Synthetic Data

To develop initial insight into the behavior of SELEX, we experimented with synthetic data in which the decision surface between positive and negative examples is known. We worked with three noise-free artificial domains, created as follows:

1. *Hyperbola.* Examples are generated as pairs, (x, y), of uniformly distributed numbers. All points satis-

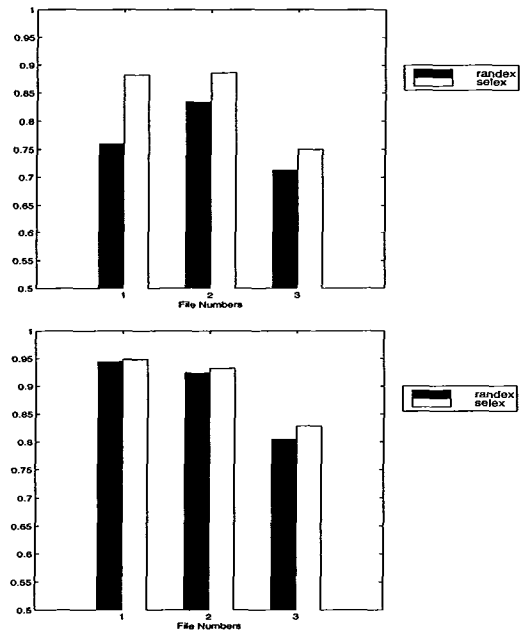

Figure 3: Classification accuracies in synthetic domains. Left to right: hyperbola, hypercubes, hyperspheres. Top: 6 gaussian centers. Bottom: 15 centers.

fying $3x^2 - 3y^2 < 1$ are positive; all other points are negative.

2. *Hypercubes.* Examples are 5-tuples of uniformly distributed numbers. Three 5-dimensional hypercubes are created. All points inside any hypercube are positive; all other points are negative.
3. *Hyperspheres.* Examples are 5-tuples of uniformly distributed numbers. Three 5-dimensional hyperspheres are created. All points inside any hypersphere are positive, all other points are negative.

In all these three domains, care is taken that the positive and negative examples are equally represented in the data.

One of the advantages of experiments with synthetic data is that arbitrarily large sets of examples can be generated to warrant statistical significance of the results. In the experiments reported below, we always used 500 training examples and 500 testing examples. Given the relative simplicity of the decision surfaces (and the fact that the examples are noise-free), these numbers apear to be sufficient for good estimates of classification performance.

We carried out two sets of experiments with SELEX, differing in the sizes of the subsets S_i. In the first series, we required that $|S_i| = 2$, which led to RBF networks with $3 \times 2 = 6$ gaussian centers. In the second series, we required that $|S_i| = 5$ which led to RBF networks with $3 \times 5 = 15$ gaussian centers. By way of reference, we organized the experiments with RANDEX as follows: for each training set, we conducted 5 separate runs, each with a different set of randomly selected examples (6 or 15 examples), and averaged the results.

The results are graphically displayed in Figure 3. The bars show the classification accuracies observed on testing examples (classification acccuracy being defined as the percentage of correctly classified examples among all examples).

Expectedly, an increased number of gaussian centers (from $3 \times 2 = 6$ to $3 \times 5 = 15$ vectors) means higher classification accuracy in RANDEX as well as in SELEX. Also expectedly, SELEX outperformed RANDEX in all experimental domains. What is more important, however, is the fact that the margin is more clearly pronounced in the case of 6 centers than in the case of 15 centers (reaching 13% in the domain *hyperbola*). This observation corroborates the intuitive expectation that the example-selecting algorithm described in the previous section improves classification accuracy especially in very small RBF networks. When the number of centers is high, even random selection will be sufficiently representative, and the advantage of SELEX dissipates.

Note that the domain *hyperspheres* was especially difficult for the RBF network. Many more gaussian centers would be necessary if the network were to achieve satisfactory performance. For this reason, SELEX in this domain clearly outperformed RANDEX, even in the case of 15 centers.

Benchmark Data

To make sure that our conjectures extend also to real-world domains, we experimented with the benchmark data files from the Machine Learning Database Repository of the University of California, Irvine (Blake, Keogh, and Merz, 1998). The particular choice of testbeds was constrained by our focus on two-class problems and by the requirement that the examples be described mainly by numeric attributes. Occasional boolean attributes were treated as numeric attributes, acquiring value 1 for *true* and 0 for *false*. The upper part of Figure 4 summarizes the experimental data files by providing information about the size of each file, the number of attributes, and the percentage of examples that are labeled by the majority class (the right-most column).

The first seven datasets have two classes. The last three domains (*balance, glass,* and *wine*) originally involved more than two classes, but we turned them into two-class problems. Specifically, the classes in the *balance* domain say whether the balance is tilted left, right, or whether there is no tilt at all. We combined `tilt-right` and `tilt-left` into one class. We aggregated the 7 classes in the *glass* domain into into `windows` (the first four classes) versus `non-windows`. In the *wine* domain, the second class and the third class were combined into one.

	Dataset	#ex.	# att.	majority
1	Hepatitis	80	19	83.75
2	B. Cancer WI 1	683	9	65.01
3	B. Cancer WI 2	569	30	62.74
4	Pima	768	8	65.10
5	Ion	351	34	64.10
6	Balance	625	4	92.16
7	Echocardiogram	61	12	72.13
8	Glass	214	9	76.17
9	Liver	345	6	57.97
10	Wine	178	12	68.85

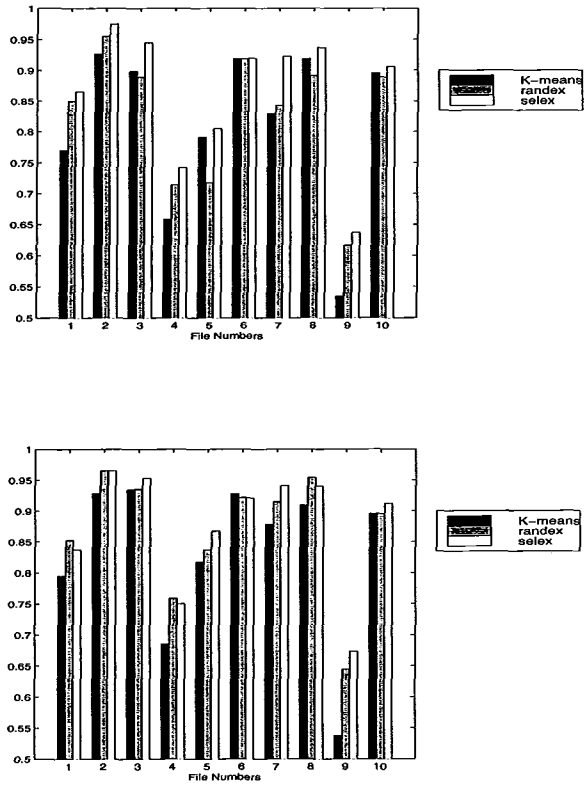

Figure 4: Classification accuracies observed in benchmark domains. Upper graph: 6 gaussian centers. Lower graph: 15 gaussian centers. For comparison, the results of initialization by k-means ($k = 6$ and $k = 15$) are shown.

These data files are not large enough to warrant statistically safe estimates of classification accuracies. Since the popular t-tests are unreliable in random subsampling of data files of this size, and since N-fold cross-validation on some of the smaller domains would entail very high standard errors, we opted for a compromise: we used 10-fold cross-validation, repeated 5 times, each time for a different partitioning of the data, and then averaged the accuracies measured in different runs. In each experiment, 90% of the examples were used for training, and the remaining 10% for testing. The variances then became reasonably small. In view of the fact that we are interested only in general tendencies, the chosen experimental methodology is sufficiently sound.

The SELEX and RANDEX experiments were organized analogically as in the case of synthetic data. This means that in SELEX, the sizes of the individual subsets were $|S_i| = 2$ or $|S_i| = 5$, meaning that the resulting RBF network would contain 6 and 15 units, respectivelly. The RANDEX results are averages from 5 random runs, each time with 6 (or 15) randomly selected examples used for $\boldsymbol{\mu_i}$.

For reference, the graphs in Figure 4 show also the performance of RBF networks that have been initialized by clustering techniques (each $\boldsymbol{\mu_i}$ is identified with the center of gravity of a cluster found by the k-means algorithm)[1] where, again, $k = 6$ or $k = 15$. The reader can see that in most domains the clustering-based initialization is a clear loser among the three techniques. From the results, one can therefore conclude that to identify $\boldsymbol{\mu_i}$'s with selected training examples is a more promising approach.

Further on, our example-selecting mechanism gives in nearly all domains better results than random selection of examples (the only exception being *balance*). Again, the margin is reduced when the number of centers increases to 15.

Conclusions

One of the main issues in RBF networks is how to define the gaussian centers of the individual radial-basis functions. One of the possibilities is to identify these centers with representative training examples. A natural requirement is that the ensuing network should be as small as possible to prevent overfitting and to minimize classification costs.

Our solution is based on a simple heuristic that says that rather than a single subset containing N examples, one should search for three smaller subsets, S_1, S_2, and S_3, such that $|S_1 \cup S_2 \cup S_3| < N$. The sets should complement each other in the sense that, when employed by a 1-nearest-neighbor classifier, each would lead to errors in a different part of the instance space.

This approach turned out to outperform random selection of examples. Another observation is that (at

[1]Clustering techniques did not make much sense in the synthetic data because the examples were uniformly distributed.

least in the benchmark domains we used) initialization by selected training examples gave better results than the more expensive clustering technique suggested by Moody and Darken, (1989) and Musavi et al. (1992).

On the other hand, the achieved performance does not seem to reach the classification accuracies reported by Kubat (1998) for his decision-tree based initialization. However, a great advantage of SELEX is the compactness of the resulting network. In the decision-tree based approach, a typical network contained dozens of neurons.

The encouraging results suggest that example-selecting techniques (used in RBF initialization) deserve further attention. We recommend that instead of just triplets of subsets, one should study the behavior of the more general K-tuples ($K > 3$). This will lead to more general version of Equation 1 and of Lemma 1.

Although not included in our results, preliminary experiments showed that if the size of the S_i's was two, the classifier accuracy sometimes suffered. Increasing the size from three to five did not significantly improve the results with other datasets. We therefore left the size of the S_i's at three because this seemed optimal for the majority of datasets involved. Additionally, similar preliminary experiments suggested the number of randomly generated candidates used in choosing the S_1, S_2 and S_3 subsets were optimal for the given benchmark datasets.

Apart from that, we observed that different domains called for different sizes of S_i's. This means that one should look for heuristics that would suggest the ideal size for the given problem. Flexibility can be further enhanced by allowing that each S_i has a different size.

References

Blake, C., Keogh, E., and Merz, C.J. (1998). UCI Repository of machine learning databases [www.ics.uci.edu/ mlearn/MLRepository.html]. Irvine, CA: University of California, Department of Information and Computer Science.

Breiman, L. (1996). Bagging Predictors. *Machine Learning*, 24, pp. 123–140.

Broomhead, D.S. and Lowe D.(1988). Multivariable Functional Interpolation and Adaptive Networks. *Complex Systems*, 2, 321–355.

Chen, S., Cowan, C.F.N., and Grant, P.M. (1991). Orthogonal Least Squares Learning Algorithm for Radial Basis Function Networks. *IEEE Transactions on Neural Networks*, 2, 302–309.

Cheng, Y.-H. and Lin, C.-S. (1994). A Learning Algorithm for Radial Basis Function Networks: with the Capability of Adding and Pruning Neurons. *Proceedings of the IEEE*, 797–801.

Cortes, C. and Vapnik, V. (1995). Support Vector Networks. *Machine Learning*, 20, 273–279.

Duda R.O. and Hart, P.E. (1973). *Pattern Classification and Scene Analysis.* John Wiley & Sons, New York.

Fritzke, B. (1993). Supervised Learning with Growing Cell Structures. In *Advances in Neural Information Processing Systems 6*, J. Cowan, G. Tesauro, and J. Alspector (eds.), San Mateo, CA: Morgan Kaufmann, pp.255–262

Fritzke, B. (1994). Fast Learning with Incremental RBF Networks. *Neural Processing Letters*, 1 ,2–5

Gates, G.W. (1972). The Reduced Nearest-Neighbor Rule. *IEEE Transactions on Information Theory*, IT-18, pp.431–433.

Hart, P.E. (1968). The Condensed Nearest-Neighbor Rule. *IEEE Transactions on Information Theory*, IT-14, pp. 515–516.

Kubat, M. (1998). Decision Trees Can Initialize Radial-Basis Function Networks. *IEEE Transactions on Neural Networks*, 9, pp. 813–821.

Moody, J.E. (1989). Fast Learning in Multi-Resolution Hierarchies. *Advances in Neural Information Processing Systems*, vol. 1 (pp. 29–39), Morgan Kaufmann, San Francisco, CA.

Musavi, M.T., W. Ahmed, K.H. Chan, K.B. Faris, and D.M. Hummels (1992). On the Training of Radial Basis Function Classifiers. *Neural Networks*, 5, 595–603.

Ritter, G.L., Woodruff, H.B., Lowdry, S.R., and Isenhour, T.L. (1975). An Algorithm for a Selective Nearest-Neighbor Decision Rule. *IEEE Transactions on Information Theory*, IT-21, pp. 665–669.

Schapire, R.E. (1990). The Strength of Weak Learnability. *Machine Learning*, 5,197–227.

Tomek I. (1976). Two Modifications of CNN. *IEEE Transactions on Systems, Man and Communications*, SMC-6, 769–772.

Vovk V. G. (1990). Aggregating Strategies. In *Proceedings of the Third Annual Workshop on Computational Learning Theory,* pp. 371-383.

Wettschereck D. and Dietterich T.G. (1992). Improving the Performance of Radial Basis Function Networks by Learning Center Locations. *Advances in Neural Information Processing Systems 4*, (J.E. Moody, S.J. Hanson, and R.P. Lippmann, eds.), 1133–1140. San Mateo, CA: Morgan Kaufmann

Wilson, D.L. (1972). Asymptotic Properties of Neraest Neighbor Rules Using Edited Data. *IEEE Transactions on Systems, Man, and Cybernetics*, 2, pp.408–421.

Wilson, D.R. and Martinez, T.R. (1997). Instance Pruning Techniques. *Proceedings of the 14th International Conference on Machine Learning.* Morgan Kaufmann.