CHAPTER 10

USING COMPUTATIONAL INTELLIGENCE FOR COMPUTER-AIDED DIAGNOSIS OF SCREEN FILM MAMMOGRAMS

Walker H. Land, Jr.
Departments of Bioengineering and Computer Science, Binghamton University, Binghamton, NY

Daniel W. McKee
Department of Computer Science, Binghamton University, Binghamton, NY

Frances R. Anderson
Our Lady of Lourdes Memorial Hospital, Binghamton, NY

Timothy Masters
TMAIC, Brackney, PA

Joseph Y. Lo
Duke University Medical Center, Durham, NC

Mark Embrechts
Decision Sciences and Engineering Department, Rensselaer Polytechnic Institute, Troy, NY

John Heine
Department of Radiology, Moffitt Cancer and Research Center and Univ. of South Florida, Tampa, FL

CONTENTS

10.1 Introduction / 323
10.2 Evolutionary Programming and Evolutionary Computation / 323
  10.2.1 Evolutionary process / 324
  10.2.2 Duke University database / 325
  10.2.3 Results / 326
  10.2.4 Discussion / 332
10.3 Evolutionary Programming/Adaptive Boosting Hybrid / 332
  10.3.1 Adaptive boosting / 333
  10.3.2 Development and refinement of the hybrid / 334

321
10.1 INTRODUCTION

This chapter describes paradigms and hybrids applied to the computer aided diagnosis (CAD) of breast cancer using mammogram screen film data. The chapter is organized into the following sections:

- Evolutionary programming (EP)/evolutionary strategies (ES) family of neural networks (NNs), derived and evaluated using statistical cross-validation techniques with results measured by ROC analysis.
- EP/adaptive boosting (AB) hybrid-derived family of NNs, with CAD results.
- EP approach, which evolves into not only NN parameters, but also the architecture, with CAD results.
- EP-derived support vector machines (SVMs), with CAD results.
- SVM/GRNN oracle hybrid formulation, including the development of the probabilistic neural network (PNN) and the generalized regression neural network (GRNN), with CAD performance results.
- Partial least squares (PLS) and kernel-PLS (K-PLS), with CAD performance results.
- Modular artificial intelligence (AI) system design and the use of knowledge engineering (KE) and knowledge representation (KR), with CAD performance results.

This research has been supported by several sponsors, whose acknowledgments are included at the end of appropriate sections.

10.2 EVOLUTIONARY PROGRAMMING AND EVOLUTIONARY COMPUTATION

In recent years, artificial neural network (ANN) techniques have dominated the relatively new field of CAD, particularly in the diagnosis of breast cancer.\(^1\)\(^-\)\(^6\) Most of these studies relied on a classic ANN paradigm, namely, the single-hidden-layer, fully-interconnected, feed-forward, error-back propagation network, which used sigmoid activation functions. Although versatile and popular, this classic ANN approach has many limitations. In particular, the gradient descent technique used to train network weights is susceptible to entrapment in local minima. Furthermore, the number of hidden nodes are fixed and chosen arbitrarily. To achieve a desired level of performance, too many hidden nodes frequently are used, resulting in over-fitting of the training cases, which compromises the network’s ability to generalize to new cases it has not seen before.

To address these limitations, the evolutionary computing (EC) paradigm\(^7\)\(^,\)\(^8\) was investigated as an alternative to the classic ANN paradigm. The EC paradigm is a stochastic optimization technique, consisting of a blend of EP and ES, which numerically addresses (but is not immune from) the problem of entrapment in local minima.
Using available mammographic findings and patient history, researchers applied these techniques to the problem of predicting whether a breast lesion was benign or malignant. Mammographic findings were used because mammography is the most widely used radiologic modality for the early detection of breast cancer. However, only 15–34% of women who undergo a breast biopsy for a mammographically suspicious, nonpalpable lesion actually have breast cancer.\(^9,^{10}\) Thus, 66–85% of the biopsies performed today could be avoided accurately if these lesions could be classified accurately using the information from a mammogram.

### 10.2.1 Evolutionary Process

The evolutionary process evolves parameters for a family of fixed-architecture NNs. A generic description of the selection process used, called rank order selection, is as follows. First, a population of trial solutions (candidate NN architectures and associated parameter values) is randomly configured. Each of these architectures (parents) then is copied and mutated (children) and all parents and children are scored using an objective (or fitness) function. These scored architectures are rank ordered based on the results of tournament selection methodologies, where the best performing architecture is placed first and the poorest performing architecture is placed last. The lower 50% of these candidate parents and children are “killed off” to retain the original population size. The remaining better-fit architectures are again mutated, and the fitness process continued for a prespecified number of generations. As long as the selection and mutation process results in population elements that increase in fitness, then the average and maximum population fitness will asymptotically increase to some maximum value as the population is evolved over several generations. The two key components of this evolutionary process are tournament selection (with and without replacement) and mutation.

In tournament selection without replacement, each NN is taken, one at a time, and compared with, say, 10 other randomly selected networks from the combined parent and child populations. If the chosen network produces a smaller classification error on the training data than the other 10, it is declared the winner of the round and its win count is increased. If the chosen network does not beat all 10 competitors, then the competitor with the most accurate classification score has its win count increased. This process is repeated \(2^m\) times, once for each element in the combined parent and child population. The set of \(2^m\) population elements then are reordered, where the NN with the largest win count is placed first and the NN with the lowest win count placed last. The \(m\) lowest-performing networks (50% of the population elements) then are discarded to reduce the population to its original size.

In tournament selection with replacement, each network from the \(2^m\) population elements is compared to 10 other randomly selected NNs from the same population. If it beats all 10, it retains its place in the hierarchy. If a competitor’s performance exceeds this selected network (performs with minimum classification error on the training data), then this competitor replaces the originally selected network.
in the hierarchy. The impact of this replacement is twofold: it makes this “better” performer more likely to be chosen as a competitor for the remaining NNs, and its duplication in the population allows it to have more “children” in the next generation. The tournament process is repeated for all $2m$-population NN elements. The population then is reordered (based on win count), where the best performing network is placed first and the poorest performing network placed last. The population of $2m$ is pruned to $m$ by discarding the bottom 50%, which are the poorest performers.

The mutation process utilizes both random and bias mutation of several self-adaptive components (i.e., weights and biases). A mutation process taken from evolutionary strategies is used, and consequently, integrates the mutation process with the selection process described previously (usually used in evolutionary programming). First, the variances for each weight and bias component of a fixed configuration NN are mutated using the following expression:

$$
\sigma'_i = \sigma_i \exp \left[ \frac{1}{\sqrt{2n}} N(0, 1) + \frac{1}{\sqrt{2\sqrt{n}}} N_i(0, 1) \right],
$$

(10.1)

where $n$ is the total number of weights and biases comprising the network. (Note, here $n$ is not the population size, but is the length of the $\sigma$ vector for each network.) $N(0, 1)$ is a standard normal variable sampled once for all $n$ parameters of the $\sigma$ vector. $N_i(0, 1)$ is a standard normal random variable sampled once for each of the $n$ parameters in the $\sigma$ vector.

The second step of this more rigorous mutation process is the updating of all weights and biases for all networks in the evolving population. If the vector $x_i$ denotes these elements for each of the individual networks, the update process will be accomplished as follows:

$$
x'_i = x_i + C \sigma'_i,
$$

(10.2)

where $i$ is the $i$th component of the $x$ vector, $x$ is the vector containing the current weights and biases, $\sigma'_i$ is a vector containing the variances computed by the previously described process, and $C$ is a standard Cauchy random variable. This new $x'_i$ vector is computed in turn for the weights and biases for each of the $i$ child elements of the $n + 1$st generation.

### 10.2.2 Duke University database

The Duke University mammographic database, which is the first of the databases used to evaluate some of the to-be-described diagnostic and classification paradigms, consists of 500 cases of nonpalpable, mammographically suspicious breast lesions selected randomly from patients seen at Duke University Medical Center.
Each case underwent needle localization and excisional biopsy, resulting in definitive histopathologic diagnosis. Of these 500 cases, 326 (65%) were benign. The mean age was 55.5 years, with a range of 24–86 years.

For each case, ten mammographic findings were extracted by one of four experienced radiologists blinded to biopsy outcome. These findings included three pertaining to calcifications, four pertaining to masses, and three miscellaneous findings. Findings were encoded according to the BI-RADS™ lexicon, a standard adopted by the American College of Radiology to improve consistency and accuracy of mammogram interpretation. In addition, six variables pertaining to patient history (age, personal history of breast cancer, family history of breast cancer, etc.) also were recorded for each case. Each of the sixteen variables were linearly scaled into floating-point numbers between 0 and 1, with greater values corresponding to a priori increased likelihood of malignancy (Table 10.1). This database has been used successfully in several experiments involving NNs to predict breast cancer.11–13

10.2.3 RESULTS

To judge the efficacy of the EC-derived NNs’ performance, extensive k-fold cross-validation evaluations were performed. Cross-validation is a statistical process whereby a data set of limited size may be cleverly partitioned to better evaluate the evolved NN performance against a given data set. Using the mammographic data set containing 500 cases, five-fold cross-validation experiments were designed by the researchers as follows. The 500 cases were partitioned into 5 groups of 100 each. The first 100 cases were “held out” as the validation set, and the remaining

<table>
<thead>
<tr>
<th>Input No.</th>
<th>Variable name</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>I0</td>
<td>Calcification distribution</td>
<td>Mammographic calcification findings</td>
</tr>
<tr>
<td>I1</td>
<td>Calcification number</td>
<td></td>
</tr>
<tr>
<td>I2</td>
<td>Calcification description</td>
<td></td>
</tr>
<tr>
<td>I3</td>
<td>Mass margin</td>
<td>Mammographic mass findings</td>
</tr>
<tr>
<td>I4</td>
<td>Mass shape</td>
<td></td>
</tr>
<tr>
<td>I5</td>
<td>Mass density</td>
<td></td>
</tr>
<tr>
<td>I6</td>
<td>Mass size</td>
<td></td>
</tr>
<tr>
<td>I7</td>
<td>Quadrant</td>
<td>Mammographic miscellaneous findings</td>
</tr>
<tr>
<td>I8</td>
<td>Associated findings</td>
<td></td>
</tr>
<tr>
<td>I9</td>
<td>Special cases</td>
<td></td>
</tr>
<tr>
<td>I10</td>
<td>Age</td>
<td>Patient history variables</td>
</tr>
<tr>
<td>I11</td>
<td>Family history</td>
<td></td>
</tr>
<tr>
<td>I12</td>
<td>History of benign biopsy</td>
<td></td>
</tr>
<tr>
<td>I13</td>
<td>History of breast cancer</td>
<td></td>
</tr>
<tr>
<td>I14</td>
<td>Menopause</td>
<td></td>
</tr>
<tr>
<td>I15</td>
<td>Hormone treatment</td>
<td></td>
</tr>
</tbody>
</table>
400 cases were used as the training/evolving set. The family of 100 fixed architecture NNs were each trained using the remaining 400 cases. The held-out 100-case set was used as the validation set, and the ROC $A_z$ index (see brief description of ROC curves below) computed for the best performing network. This allows the network to be trained on the most possible data, while reducing the likelihood that partitioning the data has introduced a bias into the results (as could happen if the “difficult” cases were all in the training set, and the validation set contained all “easy” cases).

The “vanilla” ROC curve is a plot of the true positive ratio (TPR) as a function of the false-positive ratio (FPR) (vertical and horizontal axes, respectively), both of which are in the $[0, 1]$ closed interval. The area under this curve (ROC index or $A_z$ index or AUC) represents an overall performance “averaged” over all possible (TPR, FPR) operating points. The statistically derived ROC curves are used to assess the effectiveness of a classification procedure and are obtained by evaluating the system under all possible threshold settings. The probability of detecting a malignancy is traded off as a function of the likelihood of obtaining a false-positive result. As the threshold value is lowered, the EC-derived NN will correctly identify a greater number of malignancies, but this comes at the expense of a higher false-positive rate. Conversely, raising the threshold value can lower the false-positive rate; but this, in turn, decreases the sensitivity.

Continuing with the five-fold cross-validation description, the second set of 100 cases was held out while the first 100 and last 300 were used as the training set. The best of this different family of NNs was evaluated using the second set of held-out cases and the ROC index computed. This process was continued, using the third, fourth, and fifth sets of 100 held-out cases for validation sets, and the ROC indices were computed.

10.2.3.1 Sensitivity experiments

This section delineates a set of sensitivity experiments conducted to find the best performing EP-evolved NN as measured by the ROC $A_z$ index. These experiments were conducted in separate phases: a design phase and an optimization phase. A discovery phase consisted of identifying the ranges of system parameters that would produce evolved families of NNs that provided average ($A_z \approx 0.70–0.75$) performance.

10.2.3.2 Design phase

The population size (i.e., number of networks in a given generation) was varied between 50 and 500. Contrary to intuition, larger populations did not necessarily perform better. This may be due to the ratio between the population size and the tournament arena size. Additional analysis was done holding the sample size for tournament selection (arena size) at 10% of the population. This improved NN performance for very small populations (50 networks), but actually decreased the network performance achieved over larger populations.
The range of values used as weights in the initial population was decreased from the defaults of ±0.5 to ±0.25, and then increased to ±1.0, ±2.0, and ±3.0. Asymmetric initializations (−1.0, 0) and (0, 1.0) were also tried. There was no discernable correlation between these initial weights and network performance. It was expected that the influence of this parameter would decrease as the number of generations increased, but that a larger initial range could give more diverse initial networks and might cover a larger potion of the solution space.

The default values for the initial sigma weights were (0.009, 0.02). The lower bound was decreased to 0.002, and the upper bound was varied from 0.02 to 0.04 for both the original and modified values of the lower bound. Like the multilayer feed-forward network (MLFN) initial weights, it was expected that with a significant number of generations, this adjustment to the initial sigma values would have little effect. By adjusting the initialization range to (0.002, 0.03), network performance achieved after 250 generations improved slightly. Further experimentation is needed to determine whether this improvement is statistically significant. Larger initialization ranges showed poorer performance, demonstrating that the default range is close to the optimum, and greater diversity in the population does not necessarily imply better performance.

The number of generations was varied from 100 to 1000. With the default values for all other parameters, the best-performing NN occurred after 265 generations. The best error improved slightly during the remaining generations, but validation performance did not improve. $A_z$ indices appeared to correlate with validation performance (this was not always the case while varying other parameters). During the optimization phase, a more exhaustive analysis of the effect of this parameter on the final $A_z$ index was completed.

The number of neurons on the hidden layer was varied from 1 to 5. As expected, the validation performance after 250 generations generally decreased with more than 2 neurons in the hidden layer. The exception to this was that a network with 5 neurons in the hidden layer did particularly well.

Additional experimentation was done with higher numbers of generations to ensure that the poorer performance was due to the network “learning” too much about the training set, and not due to the more complex networks taking longer to evolve. This was seen to be the case as the $A_z$ index began to drop with higher numbers of generations in the networks.

The 5-neuron model that performed well was subjected to a 2-fold and a 5-fold cross-validation. The results indicated that the exceptional result from the original data set was likely due to a bias in the arrangement of the data, as the overall performance under 5-fold cross-validation was less than the 2-neuron model under the same validation. This was further confirmed by a cursory investigation of networks with 6 or 7 neurons in the hidden layer, which quickly began memorization of the training set and performed poorly upon validation. The 5-neuron model did outperform the 2-neuron model on validation set “a,” arguably the most difficult. Further research into how to capitalize on this improvement with the difficult data, while minimizing the negative effects of this model on easier data, could improve overall performance of the NN.
Experiments were tried in which a rank-order cutoff selection was substituted for tournament selection with replacement. As with the initial validation set, performance achieved under the cutoff method was superior, but \( A_z \) indices were lower. It was expected that the tournament selection results would be superior, as that method allows networks that have the potential to evolve into better performing networks to survive. This proved to be the case under 5-fold cross-validation after 250 generations. However, the cutoff method achieved better results than tournament selection when reevaluated after 500 generations.

10.2.3.3 Training and validation data sets

The data set was broken into five groups of 100 records each, labeled a, b, c, d, and e, where a, b, c, and d were the first, second, third, and fourth 100 elements of the training set, and e was the validation set. These were combined to form two groups of 250 each, with group a250 consisting of the elements from a, b, and the first half of c, and group b250 consisting of the remaining elements.

The complete set of records also was split into two separate files, each containing only benign or only malignant cases. Since the number of malignant cases overall was about 35%, a training set of 250 cases with a 50/50 mix of benign and malignant cases was assembled (using the first 125 cases of each type). The remaining 250 cases were used for validation. These two sets then were reversed to see the impact of a different benign/malignant mix on network performance.

It was expected and confirmed that using all benign or all malignant cases in a training set would cause the network to quickly converge to one that always provided the same diagnosis, regardless of the inputs. The experiments with different benign/malignant ratios had poorer performance than consistent ones. This may be due to the small number of cases available, or due to training with a non-representative ratio. Surprisingly, although the validation results were very poor (58–72%), the \( A_z \) indices were not as low as expected (0.77–0.79). For the remainder of the experiments, the labeled groups of 100 and 250 were used.

Based on the design-phase experiments, the values in Table 10.2 were chosen as a starting point for optimization. A 5-fold cross-validation computed with these values yielded an average \( A_z = 0.76913 \).

<table>
<thead>
<tr>
<th>Table 10.2 Initial parameters for optimization phase.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameter</td>
</tr>
<tr>
<td>Population size</td>
</tr>
<tr>
<td>MLFN initialization range</td>
</tr>
<tr>
<td>Sigma weights initialization range</td>
</tr>
<tr>
<td>Number of generations</td>
</tr>
<tr>
<td>Number of neurons in hidden layer</td>
</tr>
<tr>
<td>Selection method</td>
</tr>
</tbody>
</table>
Values that worked well for experiments in the design phase were subjected to 5-fold cross-validation (using each of a, b, c, d, and e as validation sets and the remaining 400 as a training set), to verify that values were indeed “good.” Each good value was taken as a default fixed starting point, and the parameters varied individually and then fixed one-by-one at the point that gave the highest $A_z$ index.

As a baseline, 5-fold cross-validation was done with the random seed fixed and the program defaults for all other values. The result was an average $A_z = 0.81866$.

The first parameter to be varied was the selection method. Tournament selection was tried with arena sizes of 8, 10, and 12, yielding $A_z$ indices of 0.80956, 0.81712, and 0.77213. Since the arena size of 10 gave the best results, this was used with tournament selection for the remainder of the optimization phase.

The next parameter to be varied was number of neurons in the hidden layer. Values of one and three for this parameter yielded $A_z$ indices of 0.81245 and 0.79253, respectively. Neither of these was as good as the previous mark (with two neurons in the hidden layer), so the number of neurons in the hidden layer was left at 2.

Population size was the next parameter to be varied. Populations of 125, 150, and 200 yielded $A_z$ indices of 0.81584, 0.79093, and 0.83727. Since the population of 200 yielded the best result, populations of 175 and 225 also were tried, yielding $A_z$ indices of 0.81258 and 0.77807. Since both of these results were poorer than the 0.83727 achieved with 200, the population was fixed at 200 for the remainder of the optimization.

The initialization range for the MLFN weights also was varied. Ranges of $(-0.05, 0.05)$, $(-0.25, 0.25)$, $(-0.75, 0.75)$, and $(-1.0, 1.0)$ yielded $A_z$ indices of 0.81943, 0.78706, 0.82144, and 0.76388. Since none of these changes were able to improve on the previous best result, the MLFN initialization range was left at $(-0.5, 0.5)$.

The next parameter to be varied was the sigma (mutation vector) initial weights. The range was reduced to the original setting of $(0.009, 0.02)$, and then each of the endpoints was changed individually, i.e., $(0.009, 0.03)$ and $(0.002, 0.02)$. The resulting $A_z$ indices were 0.81310, 0.79772, and 0.78742. Since none of these produced better results than the range $(0.002, 0.03)$, this range was kept for the final optimization.

The final parameter to be fixed was the number of generations (Fig. 10.1). Validation was performed after each generation (default is every 5) to ensure that the best network was found. $A_z$ indices after 100, 250, 750, and 1000 generations were 0.73568, 0.80883, 0.83553, and 0.83709.

An analysis of the validation performance (Fig. 10.2) of individual folds indicated that one of the sets experienced a marked improvement between 500 and 600 generations, while one had a slight decrease in performance (likely due to the beginning of memorization of the training set), and the other three remained constant over this interval. One validation set also had a substantial jump in performance at 935 generations, but this increase was more than offset by the decrease
Figure 10.1 ROC $A_z$ index as function of number of generations for the 5 folds.

Figure 10.2 ROC $A_z$ indices for best performing neural network over the five-fold cross-validation experiments.
in performance of the other sets by that time. The optimum value for the number of generations was moved to 600, yielding an average $A_z$ index of 0.83986, and a standard deviation of 0.0525.

Returning the validation check to the default of once every five generations (which substantially reduces the time necessary to complete experiments) had no significant impact on the results, giving an $A_z$ average of 0.84261 with a standard deviation of 0.0525.

10.2.4 Discussion

The best network achieved a balance between the “easy” and “difficult” sets in the 5-fold cross-validation. In general, parameter changes that improved performance on the more difficult data also decreased performance on the easier sets. Finding which changes maximized performance of the more difficult sets while minimizing the cost of those changes to performance on the easier data allowed the NN to achieve an average $A_z$ index of 0.84261, with a standard deviation of 0.0525. Under 2-fold cross-validation, these settings produced an average $A_z$ index of 0.80425.

In ongoing work, EP performance will be established using a reduced set of input variables for the evolved NN. Mammographic calcification findings, mass findings, certain miscellaneous findings and certain patient history variables will be used rather than the complete set delineated in Table 10.3. Additional hybrid evaluations will incorporate the frequency of linkage from each of the 16 discriminators obtained from a previous sensitivities analysis.

10.3 Evolutionary Programming/Adaptive Boosting Hybrid

While the evolutionary computation techniques described in Section 1 can rapidly find a “good” solution to an optimization problem, they provide no guarantee of reaching a globally optimal solution. Adaptive boosting (AB), described in this section, complements evolutionary computation by combining several evolved solutions to maximize performance, while also providing a bound on the error of its classifications.

<table>
<thead>
<tr>
<th>Table 10.3 Parameters for best performing neural network.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Best network parameters</td>
</tr>
<tr>
<td>Population size</td>
</tr>
<tr>
<td>MLFN initialization range</td>
</tr>
<tr>
<td>Sigma weights initialization range</td>
</tr>
<tr>
<td>Number of generations</td>
</tr>
<tr>
<td>Number of neurons in hidden layer</td>
</tr>
<tr>
<td>Selection method, arena size</td>
</tr>
</tbody>
</table>
10.3.1 Adaptive Boosting

The adaptive boosting algorithm AdaBoost, introduced by Freund and Schapire,\textsuperscript{14} takes the output from a “weak” learning algorithm—a method able to provide slightly better than random performance—and “boosts” the performance of that algorithm. This boosting is accomplished by successively applying the algorithm to available training data using a system of weights which are dynamically updated based on the weak learner’s performance. The final prediction is formed from the intelligent combination of outputs from the weak learners. This boosting method has the advantage over similar, earlier methods in that it requires no prior knowledge of the performance of the weak learning method—thus, the “adaptive” in adaptive boosting.

The pseudocode for this algorithm is given in Fig. 10.3. The algorithm takes as input a training set of \( m \) elements \((x_1, y_1), \ldots, (x_m, y_m)\) where \( x_i \) is a member of some domain or instance space \( X \), and each label \( y_i \) is in some label set \( Y \).

**Given:** \((x_1, y_1), \ldots, (x_m, y_m)\) where \( x_i \in X, y_i \in Y = \{-1, +1\}\)

**Initialize** \( D_1(i) = 1/m \).

**For** \( t = 1, \ldots, T \):

- Train weak learner using distribution \( D_t \).
- Get weak hypothesis \( h_t : X \rightarrow \{-1, +1\} \) with
  \[
  \text{error } \varepsilon_t = \sum_{i: h_t(x_i) \neq y_i} D_t(i).
  \]
- Choose \( \alpha_t = \frac{1}{2} \ln \left( \frac{1 - \varepsilon_t}{\varepsilon_t} \right) \).
- Update:
  \[
  D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} 
  e^{-\alpha_t} & \text{if } h_t(x_i) \neq y_i \\
  e^{\alpha_t} & \text{if } h_t(x_i) = y_i
  \end{cases},
  \]
  \[
  = \frac{D_t(i) e^{[-\alpha_t y_i h_t(x_i)]}}{Z_t},
  \]
  where \( Z_t \) is a normalization factor chosen such that \( D_{t+1} \) will be a distribution.

**Output** the final hypothesis:

\[
H(x) = \text{sign} \left( \sum_{t=1}^{T} \alpha_t h_t(x) \right).
\]

**Figure 10.3** Freund and Schapire’s AdaBoost algorithm.
In its most basic form, $Y$ is a simple binary classifier such as $Y = \{0, 1\}$, or $Y = \{-1, +1\}$. AdaBoost repeatedly calls a weak learner in a series of $T$ rounds, while maintaining a distribution or set of weights ($D_t$) over the training set. These weights initially are set equally; but on successive rounds, the weights of incorrectly classified cases are increased, while the weights of correctly classified cases are decreased. The result of this redistribution is forcing of the weak learner to focus on the more “difficult” cases in the training set.

The weak learner must find a weak hypothesis $h_t : X \rightarrow \{-1, +1\}$ that is appropriate for the distribution $D_t$ for that round. The fitness of the weak hypothesis is measured by its error with respect to the distribution on which the weak learner was trained. A separate parameter $\alpha_t$ is chosen such that a smaller error produces a larger $\alpha$, and as the error approaches random guessing ($\varepsilon \rightarrow 1/2$), $\alpha$ approaches zero. The distribution then is updated by using the rule shown in Fig. 10.3. The effect of this update is to increase importance for the next round of cases misclassified (and decrease importance of cases correctly classified) in the current round.

The $\alpha$ then is used as a weight in combining the weak learner’s hypotheses from the $T$ rounds to generate a composite prediction $H$, which is a weighted majority vote of the $T$ weak hypotheses. Freund and Schapire show that if each weak hypothesis is slightly better than random guessing, the training error drops exponentially.

10.3.2 Development and Refinement of the Hybrid

The first hybrid developed was based on the AdaBoost algorithm as described previously, calling an EP-based weak learner similar to those described in Section 10.2. Figure 10.4 is typical of the results achieved from this hybrid. Note that the $A_2$ area achieved in the first few rounds is relatively poor (70–80%), especially compared to results achieved without boosting. Although the adaptive boosting quickly improved the results, the starting point for this improvement consistently was around 70%. Even using architectures and parameters that performed very well for the EP process alone (average $A_2$ area of approximately 84.3%), the average $A_2$ area obtained after one “round” of boosting dropped substantially.

A thorough review of the algorithm as implemented revealed the cause for the loss. The basic form of the algorithm (Fig. 10.4) requires the weak learner to generate a hypothesis of the form $h : X \rightarrow \{-1, +1\}$. The EP-based weak learner used here generated outputs over the closed continuous interval $[0, 1]$, using a sigmoid activation function. These outputs were mapped to the set $\{-1, +1\}$ by application of a simple threshold at 0.5. Because of this process, information contained in the weak learner was being lost. For example, a result very close to 0.5 from the weak learner would indicate that it was unable to extract sufficient information to accurately classify that sample; however, the threshold application would “force” the output to one side or the other, losing the information that the weak learner was “unsure” of in that case.
Schapire and Singer\textsuperscript{16} discuss a generalization of AdaBoost that allows weak hypotheses of the form $h: X \rightarrow \mathbb{R}$. Restricting this hypothesis to $h: X \rightarrow [-1, +1]$ (e.g., each sample $x_i$ maps to a real number on the continuous closed interval bound by $-1$ and $+1$), one can place a firm upper bound on the potential training error, while retaining all of the information content delivered by the weak learner. In this case, the sign of the weak learner’s output would indicate the weak learner’s prediction for that sample, while the magnitude of that output can be viewed as a “confidence” of the weak learner in that prediction. This particular choice of range for the hypothesis function fits well with a hyperbolic tangent activation function (which asymptotically approaches $-1$ or $+1$ as the input to the function approaches negative or positive infinity). Thus, the EP-based weak learner was reworked to provide output over this range.

By replacing the final hypothesis from the AdaBoost algorithm with the simple weighted average,

$$H(x) = \frac{\sum_{i=1}^{T} \alpha_i h_i(x)}{\sum_{i=1}^{T} \alpha_i},$$  \hspace{1cm} (10.3)

one can maintain, with arbitrary precision, the information content gained from the boosting process. This allows a threshold setting to be chosen as appropriate for a
particular application, rather than accepting the default placement of this decision threshold at zero.

Finally, although the choice of $\alpha$ in the AdaBoost algorithm guarantees that $\alpha \geq 0$ as long as the error of the weak learner is less than 0.5, under more difficult distributions, the EP-based weak learner was unable to generate a hypothesis with an error of less than 0.5. In this case, the algorithm would produce a negative value for $\alpha$. One could argue that an error of greater than 0.5 implies a negative correlation between the hypothesis and the actual solution space—and therefore a negative value for $\alpha$ will allow information to be extracted from that negative correlation. However, the combination of “inaccurate” weak hypotheses (performing so poorly that their predictions must be inverted) with a group of more “accurate” hypotheses would seem to serve only to diffuse their accuracy.

Since a hypothesis with an error of 0.5 will yield $\alpha = 0$, such a hypothesis is meaningless in the weighted average calculation of the final boosted hypothesis. A floor was placed on the calculation of the $\alpha$ parameter such that any weak learner that had an error greater than 0.5 would also have $\alpha = 0$. A similar mechanism is employed in the AdaBoost. R algorithm\textsuperscript{14} to stop the boosting process if the error of the weak learner is greater than 0.5.

A review of the revised hybrid showed that it was now possible to achieve much higher $A_z$ results in the first few rounds of boosting. This allowed further investigation into the potential benefits of applying adaptive boosting to an optimized EP-derived network.

### 10.3.3 Results

Values that worked well for the EP process alone in previous experiments (see Table 10.4) were used as a starting point for a number of 5-fold cross-validation sensitivity experiments. These sensitivity experiments were aimed at trying to find a “near-optimal” EP configuration to use as a weak learner for the boosting process. Although this optimization initially focused on the ROC $A_z$ area achieved under 5-fold cross-validation, we were particularly interested in achieving the best possible performance at higher sensitivities. Other measures of performance used to evaluate these candidate EP configurations were the specificity and the positive predictive value (PPV). Table 10.4 summarizes the performance of two of the best EP

<table>
<thead>
<tr>
<th>Table 10.4 Optimized EP-derived MLFN—no boosting.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Average $A_z$</strong></td>
</tr>
<tr>
<td><strong>Specificity @ 100% sensitivity</strong></td>
</tr>
<tr>
<td><strong>PPV @ 100% sensitivity</strong></td>
</tr>
<tr>
<td><strong>Specificity @ 98% sensitivity</strong></td>
</tr>
<tr>
<td><strong>PPV @ 98% sensitivity</strong></td>
</tr>
<tr>
<td><strong>Specificity @ 95% sensitivity</strong></td>
</tr>
<tr>
<td><strong>PPV @ 95% sensitivity</strong></td>
</tr>
</tbody>
</table>
configurations, without boosting. A review of these results showed that although Candidate 2 had a higher overall $A_z$ area, Candidate 1 had superior performance at the higher sensitivities of 100%, 98%, and 95%. Both of these candidate configurations used a population of 200 networks evolved for 600 generations. The architecture was a multi-layer feed-forward network with one hidden layer containing one node. Candidate 1 (the better PPV performer) used tournament selection without replacement, while Candidate 2 (the better $A_z$ performer) used tournament selection with replacement.

Because the focus was on performance of the EP/AB hybrid at high sensitivities, candidate 1 was used as the starting point for a new round of sensitivity experiments. During these experiments, the modified AdaBoost algorithm was applied to ascertain what improvements it could offer for the various performance measures. With Candidate 1 as the weak learner in the hybrid, the hybrid’s performance was only slightly better than the EP process alone for most performance measures, although substantial improvements were made to the specificity and PPV at 98% sensitivity (see Table 10.5).

Because the best PPV and Specificity performance at high sensitivities did not directly correlate with $A_z$ performance (Table 10.5), additional sensitivity experiments were completed to explore configurations that might provide higher PPV and specificity values, even if the $A_z$ area achieved was lower. Through these experiments, a configuration was found that provided even better performance at the high-sensitivity levels. Table 10.6 summarizes the performance obtained by increasing the EP-based weak learner’s population size to 300, and reducing the number of generations to 350. Within 15 rounds of boosting, solutions were found that could provide an average PPV of 51.8% and specificity of 48.3% at 100% sensitivity.

### 10.3.4 Discussion

The $A_z$ area achieved in this case (0.851366) barely passed some of the nonboosted EP results, and the $A_z$ area achieved by the configuration described without boost-

<table>
<thead>
<tr>
<th>Table 10.5 Effect of boosting on optimized EP-derived network.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td>---------------------------</td>
</tr>
<tr>
<td>Average $A_z$</td>
</tr>
<tr>
<td>Specificity @ 100% sensitivity</td>
</tr>
<tr>
<td>PPV @ 100% sensitivity</td>
</tr>
<tr>
<td>Specificity @ 98% sensitivity</td>
</tr>
<tr>
<td>PPV @ 98% sensitivity</td>
</tr>
<tr>
<td>Specificity @ 95% sensitivity</td>
</tr>
<tr>
<td>PPV @ 95% sensitivity</td>
</tr>
</tbody>
</table>
ing (0.802294) was much lower than the optimized EP-only results. This lead to the conclusion that, while there existed some level of correlation between the $A_z$ performance and specificity and PPV performance, it was a somewhat loose correlation. While better $A_z$ values do imply better specificity and PPV performance over the full range of threshold settings, better $A_z$ values do not necessarily imply better PPV and specificity values at high sensitivities. The EP process used in the hybrid was a simple accuracy measure in its selection of which networks would survive to the next generation. Replacing this fitness function with one that takes into account the PPV performance of the candidates on the training data may allow improvement in results.

With this hybrid, one also is forced to make another performance trade-off in selecting the number of nodes in the hidden layer for the EP process. While limiting the architecture to one node in the hidden layer (thus, restricting the amount the network can learn about the training data), the ability of the network to overfit the training data is reduced, resulting in better generalization performance. However, when the training cases are weighted by the boosting algorithm, this architecture often is unable to “learn” enough to provide a solution with better than 50% accuracy. Adding a second node to the hidden layer allows the network to learn more, increasing the chance of finding a better-than-random solution for the difficult distributions. But this improvement comes at a price—performance on simpler distributions decreases, likely due to a slight overfitting of the training data. A better alternative may be in a hybrid that uses the simpler architecture with its better generalization when possible, and adds nodes to the hidden layer in cases where the simpler architecture fails to find a reasonable solution.

Although there is still room for improvement in this hybrid, the results show the potential for this technology to aid physicians in reducing the number of questionable mammogram cases referred for biopsy. Maintaining 100% sensitivity (missing no cancers), the best hybrid could help identify an additional 157 benign cases, reducing the number of biopsies by 31%, and increasing the positive PPV of mammography by 45–50%.

<table>
<thead>
<tr>
<th>Sensitivity</th>
<th>Metric</th>
<th>Fold 1</th>
<th>Fold 2</th>
<th>Fold 3</th>
<th>Fold 4</th>
<th>Fold 5</th>
<th>Average</th>
<th>Std. dev.</th>
<th>Round</th>
</tr>
</thead>
<tbody>
<tr>
<td>100% PPV</td>
<td>47.56</td>
<td>62.50</td>
<td>58.62</td>
<td>46.38</td>
<td>44.16</td>
<td>51.84</td>
<td>8.17</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>Specificity</td>
<td>36.07</td>
<td>55.38</td>
<td>65.15</td>
<td>45.59</td>
<td>39.39</td>
<td>48.32</td>
<td>11.94</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>97% PPV</td>
<td>47.50</td>
<td>70.83</td>
<td>58.93</td>
<td>48.44</td>
<td>53.23</td>
<td>55.79</td>
<td>9.56</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Specificity</td>
<td>40.98</td>
<td>69.23</td>
<td>63.64</td>
<td>54.41</td>
<td>54.55</td>
<td>56.56</td>
<td>10.75</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>95% PPV</td>
<td>50.00</td>
<td>70.21</td>
<td>58.18</td>
<td>51.72</td>
<td>54.24</td>
<td>56.87</td>
<td>8.07</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Specificity</td>
<td>39.34</td>
<td>78.46</td>
<td>65.15</td>
<td>58.82</td>
<td>59.09</td>
<td>60.17</td>
<td>14.11</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>$A_z$ area</td>
<td>0.780</td>
<td>0.911</td>
<td>0.850</td>
<td>0.847</td>
<td>0.869</td>
<td>0.851</td>
<td>0.047</td>
<td>8</td>
<td></td>
</tr>
</tbody>
</table>
10.4 Another Evolutionary Programming Approach

As described in Section 10.2, the EP paradigm is a stochastic optimization technique which is resistant (but not immune) from entrapment in local minima. Moreover, the version described here has a unique ability to prune input and hidden nodes, which reveals information content of the data and helps with the problem of overfitting. While the EP system in Section 10.2 attempted to optimize the parameters of an ANN, given a fixed structure (e.g., a set number of input, hidden, and output nodes), it also is possible to use the EP system to reduce the complexity of these ANN models. This section describes an EP process for optimizing both the ANN architecture and parameters.

10.4.1 Modified Evolutionary Programming Process

The EP process used in Section 10.2 was modified to develop relatively simple, straightforward ANNs through a process that optimizes both the structure and parameters of the network. The EP process may add or remove hidden nodes, and by making or breaking the connections from input nodes, also may effectively add or remove inputs as well. In addition, the EP process may change the values of any connection weight. The number of initial input and output nodes are fixed, but the number of hidden nodes may vary randomly within preset constraints. The number of connection weights is also random, given that there are no links to an input node from an output node, and that there is at most one link between any two nodes. The result is a feed-forward, recurrent network that can have connections from an input to hidden or output nodes, or from a hidden node to itself, other hidden nodes, or outputs.

The EP process is based on the same generic algorithm previously described. To initialize the process, a population of “parent” solutions is selected randomly from within preset architectural constraints and available parameters. Each ANN has a random number of hidden nodes, random number of connections, and uniformly random weight values. The parent solutions are mutated to produce a new generation of “child” solutions. The mutations include parametric mutations, which alter connection weight values, and structural mutations, which alter the number of hidden nodes and the presence of links in the network. The resulting child solutions are scored according to a fitness function, such as classification accuracy. The top 50% of parent and child solutions with the best fitness scores are selected as the next generation of parents, while the remaining are discarded. The process is repeated for a preset number of generations, with better solutions emerging through survival of the fittest.

In previous work, EP systems were developed using a database of 500 patient cases (see Section 10.2). Performance of the EP system was slightly worse than the classic optimized ANN, although statistically significant. Moreover, performance of the EP process was extremely sensitive to factors such as the data sampling
used to generate training versus validation subsets, the random number seed used to initialize the algorithm, and the number of evolved generations.

Based on this previous work, it was concluded that the EP was not fully optimized, and that this was at least in part due to excessive degrees of freedom. A current study sought to address these problems. The number of cases was increased substantially. The number of input features to the ANN was pruned using a priori knowledge based on both clinical relevance and previous ANN studies. Finally, the constraint on the maximum number of hidden nodes was tightened. All of these factors should reduce the degrees of freedom and help to improve stability of the EP performance.

10.4.2 MATERIALS AND METHODS

10.4.2.1 Clinical database

The EP process was applied to a recently enlarged database of mammographic findings and patient history data. This database consisted of 882 cases of non-palpable, mammographically suspicious breast lesions that were biopsied to yield a definitive diagnosis of benign or malignant. Of the 882 cases, 326 (65%) were benign. For each case, one of four experienced radiologists, blinded to biopsy outcome, extracted ten mammographic findings according to the standardized BI-RADS™ lexicon. In addition, six variables pertaining to patient history were recorded, resulting in a total of sixteen variables. Each variable was linearly scaled into floating-point numbers between 0 and 1, with greater values corresponding to a priori increased likelihood of malignancy. Preliminary attempts at building EP networks for all sixteen variables confirmed that there were too many degrees of freedom, resulting in poor and unstable performance.

To reduce the degrees of freedom, the sixteen variables were trimmed to eleven using a priori knowledge based on clinical relevance of the findings and previous efforts at building ANNs with a reduced number of inputs. One mammographic finding (location) and four history variables (family history of breast cancer, history of previous benign biopsy, menopause, and hormonal treatment) were excluded. The remaining eleven input variables are listed in Table 10.7.

In a previous study, only 500 cases were available. These were split into two subsets by attempting to equalize the mean values of all sixteen variables. This led to large imbalances in many key features, particularly the prevalence of malignant cases. Since the EP process was designed to optimize accuracy, which depends on this prevalence, results were unstable. To address that problem, the database was enlarged to 882 cases, divided into two subsets of 441. Random reshuffling was used to minimize differences in the means of four key fields: calcification description, mass margin, age, and biopsy outcome (3% root-mean-squared error of mean values for all four fields).
Table 10.7 Input variables for the ANN.

<table>
<thead>
<tr>
<th>Node #</th>
<th>Variable</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>I0</td>
<td>Calc. distribution</td>
<td>Mammographic</td>
</tr>
<tr>
<td>I1</td>
<td>Calc. number</td>
<td>calcification findings</td>
</tr>
<tr>
<td>I2</td>
<td>Calc. description</td>
<td></td>
</tr>
<tr>
<td>I3</td>
<td>Mass margin</td>
<td>Mammographic mass</td>
</tr>
<tr>
<td>I4</td>
<td>Mass shape</td>
<td>findings</td>
</tr>
<tr>
<td>I5</td>
<td>Mass density</td>
<td></td>
</tr>
<tr>
<td>I6</td>
<td>Mass size</td>
<td></td>
</tr>
<tr>
<td>I7</td>
<td>Associated findings</td>
<td>Mammographic misc.</td>
</tr>
<tr>
<td>I8</td>
<td>Special cases</td>
<td>findings</td>
</tr>
<tr>
<td>I9</td>
<td>Age</td>
<td>History</td>
</tr>
<tr>
<td>I10</td>
<td>History of benign biopsy</td>
<td>variables</td>
</tr>
</tbody>
</table>

10.4.2.2 Evolutionary programming

The EP system was optimized in two steps. First, the basic evolutionary constraints were established, including such factors as the maximum number of hidden nodes and the number of evolution generations. This produced a family of networks which tended to perform well. Second, a sensitivity analysis was performed by varying the random-number seed to evaluate which inputs were utilized most consistently.

In the first step, the degrees of freedom were reduced substantially from those used in the previous study. The maximum number of hidden nodes was reduced from ten to four, and the maximum number of nodes which could be added or deleted at any generation, was reduced from five to two. Performance was plotted as a function of the number of generations, and the stopping point was reduced from 600 generations to 180, accordingly. This improved performance and also helped with the saturation of ANN output values noted in the previous study.

In the second step, the constraints previously described were kept constant while the random-number seed used to initialize the EP was varied over twenty independent runs. Runs resulting in training accuracies less than 70% were discarded. For each run, the final EP architecture was displayed using a custom-written Java applet. A record was kept of all inputs which were linked either directly to the output node or indirectly via one or more hidden nodes, which in turn were linked to the output.

10.4.3 Results

10.4.3.1 Sample network architectures

The different random-number seeds yielded different network architectures within the constraints specified. The best performers were consistently perceptrons, with
no hidden nodes. A typical perceptron is shown in Fig. 10.5. In this case, the network utilized nine of the eleven inputs, but the weights for the mass findings (denoted as b, c, and d) clearly were dominant.

A very different kind of network is shown in Fig. 10.6, which demonstrates all the possible types of connections, including d, which connects an input directly to the output; a and b, which connect inputs to a hidden node; 1, which is a recurrent connection from the hidden node H12 to itself; k and j from a hidden node to another hidden node; and m and n from hidden nodes to the output.

In the second step, a sensitivity analysis was performed on the inputs to determine how frequently they were linked or utilized in twenty independent runs resulting from different random-number seeds. The results are shown in Table 10.8. For the most part, the rank order of these inputs was reasonable. The most unexpected finding was that the history of previous breast cancer was used in 95% of networks (19 of 20). The other two most frequently used were the mass margin (95%) and patient age (90%). Two other mass findings also were used at least 50% of the time. In contrast, all three calcification findings, as well as special cases and associated findings were all used less than 50% of the time.

10.4.4 DISCUSSION

This EP system was investigated as an alternative to the better known classic single-hidden-layer back-propagation ANN. In a preliminary study, EP based on a smaller database produced similar results to the classic ANN, but its behavior was not
stable. In the current study, attempts were made to improve the data-sampling scheme, increase the size of the database, and reduce degrees of freedom in the EP. Once the EP was partially optimized with respect to these new constraints, a family of EP runs was developed to assess the relative importance of the input findings.

The EP system was constrained to a maximum of four hidden nodes and eleven inputs, which represented a substantial simplification over the previous networks.

\[
a: 0.241425 \\
b: 0.316068 \\
c: -0.460892 \\
d: 6.26557 \\
e: 1.84233 \\
f: 0.727262 \\
g: 1.50675 \\
h: 2.05996 \\
i: 0.810306 \\
j: -7.83693 \\
k: 5.75446 \\
l: 1.3966 \\
m: 0.764884 \\
n: 0.896691
\]

**Figure 10.6** A complicated EP-generated network with hidden nodes.

**Table 10.8** Frequency of linkage for each input in sensitivity analysis.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Frequency (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>History of breast cancer</td>
<td>95</td>
</tr>
<tr>
<td>Mass margin</td>
<td>95</td>
</tr>
<tr>
<td>Age</td>
<td>90</td>
</tr>
<tr>
<td>Mass size</td>
<td>60</td>
</tr>
<tr>
<td>Mass density</td>
<td>55</td>
</tr>
<tr>
<td>Number of calcifications</td>
<td>40</td>
</tr>
<tr>
<td>Mass shape</td>
<td>40</td>
</tr>
<tr>
<td>Calcification description</td>
<td>35</td>
</tr>
<tr>
<td>Special cases</td>
<td>35</td>
</tr>
<tr>
<td>Associated findings</td>
<td>15</td>
</tr>
<tr>
<td>Calcification distribution</td>
<td>10</td>
</tr>
</tbody>
</table>
Contrary to expectations, many of the best performing EP networks were single-layer perceptrons, although the typical performance of these perceptrons was considerably worse than that of the optimized fully connected classic ANN. Further work must be done to assess the linear separability of this database, using both EP with better optimization as well as classic optimized ANNs with no hidden nodes.

In general, the cases in this database can be loosely separated into three groups: those characterized mammographically by mass findings alone, those with calcification findings, and the remainder with both or neither. Moreover, it is well known that for computer models and radiologists alike, the classification of mass cases is a much easier task compared to the classification of calcification cases. Finally, recent work with a smaller subset of these cases indicated that patient history findings other than age had limited value for the task of computer-aided diagnosis. This motivated the a priori elimination of all history findings other than age and history of breast cancer in the current study.

In light of these observations, the rank ordering of inputs as shown in Table 10.8 revealed some interesting trends. First, the usefulness of mass findings in general, and mass margin in particular, was confirmed. The mass margin was linked in 95% of the runs. Likewise, age was linked in 90% of the runs. This confirmed the significance of these two variables. The biggest surprise was that history of breast cancer also was used in 95% of the runs. The EP system may have recognized this fact and taken advantage of it, although further study is warranted to confirm this.

On the negative side, it was somewhat disappointing that all of the calcification findings, which uniquely characterize the mammographic appearance of more than one-third of these lesions, were used less than half of the time. The EP system essentially “gave up” on these cases, and instead focused its efforts on the more amenable mass cases. The lack of predictive power in the calcification findings is confirmed by the equally poor performance of classic ANN classifiers and expert radiologists for these cases.

Finally, there are many issues of stability that remain to be resolved. In ongoing work, the EP will continue to be optimized through further reductions in the degrees of freedom. Another key improvement will be to change the performance metric used to determine fitness during the evolutionary process. In most EP studies to date, accuracy was used as this metric, but in practice the area under the ROC curve is more clinically relevant. In the previous study, the authors demonstrated certain discrepancies in the two metrics, particularly as optimizing accuracy often led to poor ROC performance. The EP system will be modified accordingly to directly optimize the ROC area.

ACKNOWLEDGMENTS

The work reported in Section 10.4 was supported by a Whitaker Foundation grant RG 97-0322 and NIH/NCI R29-CA75547. JYL also wishes to acknowledge the efforts of his colleagues in the Department of Radiology, Duke University Medical
Center, including data collection over the past five years by the attending radiologists of the Breast Imaging Section (particularly Jay A. Baker, M.D., former section chief Phyllis J. Kornguth, M.D., and current section chief Mary Scott Soo, M.D.) and data entry efforts by staff of the Digital Imaging Research Division (particularly Käthe Douglas and division director Carey E. Floyd, Jr., Ph.D.).

10.5 EP- Derived Support Vector Machines

In the previous sections, a system for evolving multilayered feed-forward neural networks (MLFNs) was described. The EP methodology has been extended and applied to a broad spectrum of techniques. In this section, the discussion is focused on a support vector machine (SVM) for breast cancer classification, and an EP process developed to find near-optimal parameters for evolving that family of SVMs.

Most iterative methods in current use to optimize SVM parameters are time consuming processes which can yield suboptimal values, resulting in performance degradation. In many cases, optimizing the parameters required by the SVM involves a simple “trial-and-error” approach. This frequently requires the user to perform a battery of different tests with no real guarantee of producing the best possible SVM. To address this time-consuming problem of optimizing the parameters characterizing SVMs, an EP methodology was developed and applied. Several benefits resulted from this approach: (1) reduction of “user” interaction, (2) a theory that has wide applicability to several kinds of environments, and (3) comparable classification performance for the test cases that have been evaluated.

10.5.1 SVMs and Sequential Minimum Optimization

Several references are available that provide extensive information about development of the mathematical foundation of SVMs.18–23 The key concepts of SVMs are covered in these references and will not be discussed here, except as they pertain to explanation of the theoretical concept of why SVMs will provide a global minimum, whereas NNs cannot.

10.5.1.1 Structural risk minimization versus empirical risk minimization

Assume there exist $N$ observations from a screen film mammogram dataset. Each observation (or training example) consists of a vector $\mathbf{x}_i$ containing the input pattern and a corresponding known classification $y_i$. The objective of the learning machine would be to formulate a mapping $\mathbf{x}_i \rightarrow y_i$. Now, consider a set of functions $f(x, \alpha)$ with adjustable parameters $\alpha$ that define a set of possible mappings $\mathbf{x} \rightarrow f(\mathbf{x}, \alpha)$. Here, $\mathbf{x}$ is given and $\alpha$ is chosen. In the case of a traditional NN of fixed architecture, the $\alpha$ values would correspond to the weights and biases.
The quantity $R(\alpha)$, known as the expected (or true) risk, associated with learning machines is defined as

$$R(\alpha) = \int \frac{1}{2} |y - f(x, \alpha)| p(x, y) \, dx \, dy,$$

where $p(x, y)$ is an unknown probability density function from which the examples were drawn. This risk function is the expected (or true) value of the test (or validation) error for a trained learning machine. It may be shown that the best possible generalization ability of a learning machine is achieved by minimizing $R(\alpha)$, the expected (or true) risk. This generalization bound, for binary classification, holds with the probability of at least $1 - \eta$ ($0 \leq \eta \leq 1$) for all approximating functions that minimize the expected (or true) risk. Thus, the true risk $R(\alpha)$ can be bound by

$$R(\alpha) \leq R_{\text{emp}}(\alpha) + \sqrt{\frac{h \log(2N/h) + 1 - \log(\eta/4)}{N}}.$$  

(10.5)

The first term on the right side of Eq. (10.5) is known as the “empirical risk.” The empirical risk $R_{\text{emp}}(\alpha)$ is defined as

$$R_{\text{emp}}(\alpha) = \frac{1}{2N} \sum_{i=1}^{N} |y_i - f(x_i, \alpha)|.$$  

(10.6)

The function is a measure of the error rate for the training set for a fixed, finite number of observations. This value is fixed for a particular choice of $\alpha$ and a given training set $(x_i, y_i)$. The second term in Eq. (10.5) is the “Vapnik-Chervonenkis (VC) confidence interval.” This term is a function of the number of training cases $N$, the probability value $\eta$, and the VC dimension $h$. The VC dimension is the maximum number of training cases that can be learned by a learning machine without error for all possible labeling of the classification functions $f(x, \alpha)$, and is, therefore, a measure of the capacity of the learning machine. In traditional NN implementations, this confidence interval is fixed by choosing a network’s architecture a priori. The function is generally minimized by obtaining a local minimum by minimizing the empirical risk through adjustment of weights and biases. Consequently, NNs are trained based on the empirical risk minimization (ERM) principle.

In a SVM design and implementation, not only is the empirical risk minimized, but the VC confidence interval is also minimized by using the principles of structural risk minimization (SRM). Therefore, SVM implementations simultaneously minimize the empirical risk as well as the risk associated with the VC confidence interval, as defined in Eq. (10.5). Equation (10.5) also shows that as $N \to \infty$, the empirical risk approaches the true risk because the VC confidence interval approaches zero. The reader may recall that obtaining larger and larger sets of valid training data sometimes would produce (with a great deal of training experience)
a better performing NN, which resulted from classical training methods. This restriction is not incumbent on the SRM principle, and is the fundamental difference between training NNs and training SVMs. Finally, because SVMs minimize the true risk, they provide a global minimum.

In the context of classification of mammograms as benign or malignant, the objective of a SVM is to construct an “optimal hyperplane” as the decision surface such that the margin of separation between the classes is maximized. SVMs are based on the fundamental ideas of: (1) structural/empirical risk minimization (SRM/ERM), as previously described, (2) the VC dimension, (3) the constrained optimization problem, and (4) the SVM decision rule. Because populations of SVMs are being configured, computational time must be minimized, which means that the “standard” stochastic gradient ascent and conjugate gradient methods generally used to solve the dual LaGrangian must be replaced.

10.5.1.2 Sequential minimum optimization

The authors used the sequential minimum optimization (SMO) method, which substantially reduced the computation time required to develop families of learning machines. In the SMO algorithm, Platt searches through the feasible region of the dual problem and maximizes the dual LaGrangian (or objective function) by optimizing two of the LaGrange multipliers at a time (with the other multipliers fixed). Heuristics are used to select the two multipliers for optimization, and SMO is essentially a “hill-climbing” process that may be summarized as follows (where $\alpha_i$ is the LaGrange multiplier for the $i$th training vector, and $E_i$ is the prediction error on the $i$th training vector):

Choose $\alpha_2$ by

Iterate over all cases where $0 < \alpha < C$.

If that fails to make progress, iterate over all training examples.

If that fails to make progress, the problem has been solved.

For the chosen $\alpha_2$, choose an $\alpha_1$ by

Choose the case with the largest $|E_1 - E_2|$.

Iterate through the cases with unbound $\alpha$.

Iterate through all the training cases.

Second, solving for the LaGrange multipliers does not determine the bias or threshold $b$. This may be accomplished as follows:

SMO recomputes $b$ after each optimization to ensure that the Karush-Kuhn-Tucker (KKT) necessary conditions are satisfied for the two optimized examples.

$b$ is set so that $0 < \alpha_1 < C$, $f(x_i) = y_i$, where $f(x_i)$ is the output of the SVM decision rule for some input vector $x_i$, and $y_i$ is the desired (“true”) output value corresponding to the $i$th training vector.

If both $\alpha_1$ and $\alpha_2$ are at bounds, then there is a range of valid values for $b$, and SMO sets $b$ to the midpoint of that range.
10.5.2 EVOLUTIONARY PROCESS APPLIED TO SUPPORT VECTOR MACHINES

The evolutionary process (EP), as implemented, closely parallels the process described in Section 10.2, and optimally determines the parameters for a family of SVMs, each of which uses the Gaussian radial basis function (GRBF). Sigma is bounded by $0 < \sigma < 10$, whereas the regularization parameter is bounded by $0 < C < 1000$. In the EP process used here, a population of trial solutions (candidate SVMs and associated parameter values) is randomly configured. A copy of each of these “parent” architectures is mutated using a new mutation process designed for this application, and all parents and mutated “children” are scored using an objective function. These scored learning machines are then rank ordered based either on the results of a tournament selection methodology, or based strictly on their fitness scores using the objective function. Once so ordered, the “most fit” 50% are kept, while the poorer performing 50% are discarded to maintain the original population size. The remaining architectures become the parents for the next generation. This mutation and selection process continues for a prespecified number of generations.

Optimum values of both $\sigma$ and $C$ were established by using nested subsets of successively smaller values in the mutation process for successive generations. When the value for $\sigma$ is chosen, the value for $C$ then is generated. Each value of $\sigma$ generated has an associated subrange of $C$, and these values of $\sigma$ and $C$ are then properly paired.

10.5.3 DUKE UNIVERSITY AND UNIVERSITY OF SOUTH FLORIDA DATA SETS

The data set used in previous studies (Section 10.2) was augmented with a larger data set for this study; however, the larger data set had fewer discriminators available. To allow cases from both data sets to be used together, a new version of the Duke University data set was created containing only those discriminators common to both data sets.

Both the “full-up” (16 discriminators) and reduced (7 discriminators) Duke data sets contain 500 cases of nonpalpable, mammographically suspicious breast lesions. Table 10.9 shows all 16 discriminators and the discriminators reduced to 7 (underlined), 6 of which pertain to the standardized BI-RADS™ lexicon, and one clinical history variable, of all of which were scaled between $-1$ and $1$.

The University of South Florida Digital Database for Screening Mammography (USF DDSM) data set contains screening mammograms obtained from 1988 to 1999 at four hospitals. The images in the USF DDSM data set were digitized from film. They were not from full-field digital mammograms (FFDM). About 50% of the 1979 cases are benign and about 50% are malignant. The Duke and USF DDSM data sets were created from separate efforts at each location. However, both
Table 10.9 Complete and reduced data sets.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>Calcification distribution</td>
<td>Mammographic calcification findings</td>
</tr>
<tr>
<td>Calcification number</td>
<td></td>
</tr>
<tr>
<td>Calcification description</td>
<td></td>
</tr>
<tr>
<td>Mass margin</td>
<td>Mammographic mass findings</td>
</tr>
<tr>
<td>Mass shape</td>
<td></td>
</tr>
<tr>
<td>Mass density</td>
<td></td>
</tr>
<tr>
<td>Mass size</td>
<td></td>
</tr>
<tr>
<td>Quadrant</td>
<td>Mammographic miscellaneous findings</td>
</tr>
<tr>
<td>Associated findings</td>
<td></td>
</tr>
<tr>
<td>Special cases</td>
<td></td>
</tr>
<tr>
<td>Age</td>
<td>Patient history variables</td>
</tr>
<tr>
<td>Family history</td>
<td></td>
</tr>
<tr>
<td>History of benign biopsy</td>
<td></td>
</tr>
<tr>
<td>History of breast cancer</td>
<td></td>
</tr>
<tr>
<td>Menopause</td>
<td></td>
</tr>
<tr>
<td>Hormone treatment</td>
<td></td>
</tr>
</tbody>
</table>

data sets contain the same six BI-RADS™ discriminators plus age. Consequently, the authors essentially have five independent data sets (four comprising the USF DDSM data set and the Duke data set), which allow for realistic cross-institutional variability tests, as well as provide a measure of how well the various models can “scale up” to larger and more diverse data sets.

10.5.4 EXPERIMENTS TO ESTABLISH POPULATION SIZE AND NUMBER OF GENERATIONS

A number of five-fold cross-validation experiments were performed to establish appropriate parameters for the EP process. To ascertain the number of SVMs required for a given generation, population sizes were varied from 20 to 75. These experiments showed that using a smaller-size population would cause the EP process to converge too rapidly, with the undesirable result of a poor overall $A_z$. Increasing the population size to about 50 SVMs corrected this premature convergence problem. Further increases of population size to 75 provided little, if any, additional improvement in the overall ROC $A_z$.

For the reduced Duke data set, the number of generations was varied between 50 and 150. Using a population size of 50, the best performing SVM resulted at the 150th generation. However, the asymptotic growth rate in the $A_z$ between the 50th and 100th generation was significant, but increased less dramatically for the remainder of the generations. While the highest root-mean-square (RMS) value for $A_z$ over all five folds was 84.23%, achieved at the 150th generation, the RMS $A_z$ was 82.21% by the 100th generation. Furthermore, the specificity was the same for the 100th and 150th generation (at 100% sensitivity). Specificity represents the percentage of cases within the total number of cases predicted as having a cancerous
lesion by the SVM that were not “false alarms.” Consequently, slightly better performing learning machines will produce higher values of specificity at 95–100% sensitivities. The PPV is defined as the likelihood that a signal of an event is correctly associated with the event, provided the signal occurred. Therefore, better performing learning machines also will provide higher PPVs.

10.5.5 RESULTS

10.5.5.1 Results using the reduced Duke data set

Table 10.10 depicts $A_z$ values as a function of population size and number of generations. It shows the percentage of performance improvement in overall $A_z$, specificity, and PPV when compared to SVMs whose optimum parameters were determined iteratively.

The best performing SVM was achieved after 150 generations using a population size of 50. This SVM’s performance was compared with the SVM whose parameters were optimized using a “standard” iterative method. The results are depicted in Table 10.11, and demonstrate that the EP-derived SVM significantly outperformed the baseline learning machine in all categories except one: the PPV at 95% sensitivity. Here, only a slight decrease in performance was observed. However, a 1.4% performance improvement that results from the iteratively derived SVM parameters cannot be accepted at the expense of misdiagnosing 5% of malignant cases as benign. One should note that minimum user interaction is required to generate the EP-configured SVM, which required inputting values for the population size as well as the number of generations desired. The manually configured SVM required a team of two people to perform a large number of experiments over several days.

Figure 10.7 compares ROC curves for the EP-configured SVM and the SVM configured using the iterative method. In this case, the SVM configured using the iterative method outperformed the EP-configured SVM when the sensitivity was below 40%. However, at clinically relevant higher sensitivities, the EP-configured SVM generated a lower number of “false alarms” while detecting all cancerous lesions.

Table 10.10 Performance of EP derived SVMs for varying populations and generations.

<table>
<thead>
<tr>
<th>Population size</th>
<th>Number of generations</th>
<th>Best average $A_z$ achieved</th>
<th>Specificity at 100% sensitivity</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>50</td>
<td>83.21%</td>
<td>41.59%</td>
</tr>
<tr>
<td>20</td>
<td>100</td>
<td>83.25%</td>
<td>43.78%</td>
</tr>
<tr>
<td>50</td>
<td>100</td>
<td>84.21%</td>
<td>48.36%</td>
</tr>
<tr>
<td>50</td>
<td>150</td>
<td>84.23%</td>
<td>48.52%</td>
</tr>
<tr>
<td>75</td>
<td>100</td>
<td>84.22%</td>
<td>48.51%</td>
</tr>
<tr>
<td>75</td>
<td>150</td>
<td>84.23%</td>
<td>48.52%</td>
</tr>
</tbody>
</table>
Table 10.11 Performance comparison of EP-configured and manually configured SVMs.

<table>
<thead>
<tr>
<th>Measure of performance (MOP)</th>
<th>EP-configured SVM</th>
<th>SVM parameters configured iteratively</th>
<th>Percent improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Az</td>
<td>84.23%</td>
<td>82.30%</td>
<td>2.345%</td>
</tr>
<tr>
<td>Specificity at 100% Sen.</td>
<td>48.52%</td>
<td>33.40%</td>
<td>45.269%</td>
</tr>
<tr>
<td>Specificity at 95% Sen.</td>
<td>52.94%</td>
<td>45.00%</td>
<td>17.64%</td>
</tr>
<tr>
<td>PPV at 100% Sen.</td>
<td>47.76%</td>
<td>44.90%</td>
<td>6.369%</td>
</tr>
<tr>
<td>PPV at 95% Sen.</td>
<td>49.20%</td>
<td>49.90%</td>
<td>-1.403%</td>
</tr>
</tbody>
</table>

Finally, Table 10.12 demonstrates that using the “full-up” Duke data set of 16 variables and the reduced set of seven variables produced comparable results. It is of note that the specificity at 95% sensitivity and PPV at both 100% and 95% sensitivities are superior for the reduced data set. This result is attributed to the error topology being less complex for the 7-variable data set than the 16-variable set. Consequently, the EP process could more accurately establish the SVM learning machine parameter values defining the global minimum for this topology.

10.5.6 Interinstitutional Variability and “Scaling Up” to Larger Data Sets

This section describes performance results using SVMs as well as the EP/AB hybrid CAD paradigms to measure performance with the larger USF DDSM data.
Table 10.12 Performance comparison of SVMs trained with 16 variables vs 7 variables.

<table>
<thead>
<tr>
<th>Measure of performance (MOP)</th>
<th>EP-configured SVM using all variables in data set (%)</th>
<th>EP-configured SVM using 7 variables in data set (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A_z$</td>
<td>84.23</td>
<td>83.446</td>
</tr>
<tr>
<td>Specificity at 100% Sen.</td>
<td>48.52</td>
<td>48.188</td>
</tr>
<tr>
<td>Specificity at 95% Sen.</td>
<td>52.94</td>
<td>56.908</td>
</tr>
<tr>
<td>PPV at 100% Sen.</td>
<td>47.76</td>
<td>54.458</td>
</tr>
<tr>
<td>PPV at 95% Sen.</td>
<td>49.20</td>
<td>58.612</td>
</tr>
</tbody>
</table>

set, the “full-up” Duke data set, and the reduced Duke data set. Measuring both the SVM kernel learning machine performance and the EP/AP hybrid performance under these conditions is very important. To be useful in a clinical environment, any CAD paradigm must work reasonably well at many different institutions under less-than-perfect circumstances for data collection.

10.5.6.1 Performance results from EP-derived SVM parameters for the USF DDSM data set

This section describes SVM performance obtained by the GRBF SVM learning machine. SVM parameters were derived by the EP process and results were achieved with the USF DDSM data set, where the learning machine was evaluated using 5-fold cross-validation.

The results of the experiment are depicted in Table 10.13. An average specificity of 16% was achieved at the nearly perfect sensitivity of 98% (missing 2% of the cancers). Specificity improved to approximately 33% when the sensitivity was decreased to 95% (missing 5% of the cancers). The overall ROC area index was 80%.

One objective of this set of experiments was to measure the GRBF performance using a large data set of approximately 2000 cases. As such, this experiment provided a preliminary test of the hypothesis that the GRBF learning machine kernel had the potential to generalize to patient data interpreted by different radiologists from different institutions.

10.5.6.2 Performance of the EP/AB hybrid with reduced Duke and USF DDSM data sets

For comparison purposes, the EP/AB hybrid described in Section 10.3 also was tested against this larger data set. $A_z$ indices, specificities, and PPVs at clinically relevant sensitivities (95% and 100%) were measured, with hybrid architectures containing one, two or, three neurons in the hidden layer.
Results are depicted in Table 10.14 for one, two, and three neurons in the hidden layer of a seven-input, one-output, EP/AB-derived NN using the USF DDSM data set. Based on specificity and overall $A_z$ index as the critical measures of performance (MOPs), the architecture with three nodes in the hidden layer was the most accurate. However, all results demonstrated that the hybrid can “scale up” to a larger data set by increasing the specificity to 4.80% at 100% sensitivity (missing no cancers) and 25% at 95% sensitivity (missing 5% of the cancers). The PPV for all architectures was 50% or greater. Because the data set used was derived using data from four hospitals, the results provide preliminary evidence to support the contention that the hybrid has some measure of adaptability to interinstitutional variability. Further work is warranted to assess whether the hybrid can provide similar performance using additional data sets.

### Table 10.13 EP-derived SVM performance with DDSM USF data set.

<table>
<thead>
<tr>
<th>Train USF/validate USF</th>
<th>Fold 1</th>
<th>Fold 2</th>
<th>Fold 3</th>
<th>Fold 4</th>
<th>Fold 5</th>
<th>Average</th>
<th>RMS</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A_z$</td>
<td>80.5595</td>
<td>79.4507</td>
<td>81.4693</td>
<td>80.8615</td>
<td>80.6</td>
<td>80.5882</td>
<td>80.5908</td>
</tr>
<tr>
<td>Specificity at 100% sensitivity</td>
<td>0.5263</td>
<td>2.46305</td>
<td>3.48837</td>
<td>0.9389</td>
<td>0.4629</td>
<td>1.5759</td>
<td>1.9803</td>
</tr>
<tr>
<td>Specificity at 97% sensitivity</td>
<td>27.8947</td>
<td>18.2266</td>
<td>30.814</td>
<td>20.1878</td>
<td>23.61</td>
<td>24.1466</td>
<td>24.5959</td>
</tr>
<tr>
<td>Specificity at 95% sensitivity</td>
<td>32.6316</td>
<td>29.064</td>
<td>38.9535</td>
<td>36.6197</td>
<td>26.39</td>
<td>32.7317</td>
<td>33.0585</td>
</tr>
<tr>
<td>PPV at 100% sensitivity</td>
<td>52.1519</td>
<td>49.3606</td>
<td>57.4359</td>
<td>46.4467</td>
<td>45.43</td>
<td>50.1650</td>
<td>50.3514</td>
</tr>
<tr>
<td>PPV at 98% sensitivity</td>
<td>56.4246</td>
<td>53.3708</td>
<td>60.6061</td>
<td>49.5868</td>
<td>48.35</td>
<td>53.6676</td>
<td>53.8550</td>
</tr>
<tr>
<td>PPV at 97% sensitivity</td>
<td>59.3472</td>
<td>53.8073</td>
<td>64.6884</td>
<td>51.1494</td>
<td>51.32</td>
<td>56.0624</td>
<td>56.3061</td>
</tr>
<tr>
<td>PPV at 95% sensitivity</td>
<td>60.4938</td>
<td>56.0976</td>
<td>66.9811</td>
<td>56.3107</td>
<td>51.82</td>
<td>58.3406</td>
<td>58.5646</td>
</tr>
</tbody>
</table>

### Table 10.14 EP/AB Hybrid results from USF DDSM data set using 5-fold cross-validation.

<table>
<thead>
<tr>
<th>Sensitivity</th>
<th>1 node</th>
<th>2 nodes</th>
<th>3 nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>100%</td>
<td>Specificity 1.96%</td>
<td>Specificity 3.71%</td>
<td>Specificity 4.80%</td>
</tr>
<tr>
<td>PPV</td>
<td>50.27%</td>
<td>PPV 50.72%</td>
<td>PPV 51.03%</td>
</tr>
<tr>
<td>98%</td>
<td>Specificity 8.046%</td>
<td>Specificity 12.81%</td>
<td>Specificity 11.71%</td>
</tr>
<tr>
<td>PPV</td>
<td>51.50%</td>
<td>PPV 52.87%</td>
<td>PPV 52.53%</td>
</tr>
<tr>
<td>95%</td>
<td>Specificity 17.15%</td>
<td>Specificity 22.81%</td>
<td>Specificity 24.94%</td>
</tr>
<tr>
<td>PPV</td>
<td>53.40%</td>
<td>PPV 55.18%</td>
<td>PPV 55.82%</td>
</tr>
</tbody>
</table>

$A_z$ Index 65.88% 73.33% 74.01%
Results with the Duke data set of 500 cases and only 7 discriminators are depicted in Table 10.15. Note that the specificities and $A_z$ indices are much higher. This result is expected because the BI-RADS™ and age discriminators came from one institution. The EP/AB hybrid provided 31% specificity at 100% sensitivity; however, the architecture with two nodes in the hidden layer provided 47.5% specificity at 95% sensitivity. Furthermore, an $A_z$ index of $\approx 80\%$ showed that the hybrid could provide “adequate” diagnostic performance, using the limited number of 7 discriminators.

10.5.6.3 Measuring SVM learning machine interinstitutional generalization capability

Choosing a good kernel is essential for obtaining the best performance from an SVM. In this section, the authors explore which kernels have the capability to generalize well to a patient data set compiled from different institutions. To address this, the authors used both the Duke and USF DDSM datasets. First, the learning machine kernels were trained using the 500-case reduced Duke data set and validated using the USF DDSM data set. Next, the learning machine kernels were trained using the USF DDSM data set and validated using the reduced Duke data set. These experiments were a good test to measure the generalization capability of the SVM designs because they were trained using the information from one institution, and validated using information from a completely different institution.

Six separate SVMs, each represented by a different kernel, were evaluated. These kernels were: GRBF; Scholkopf (S2000) kernel; polynomial kernels of degrees 2, 3, and 4; and the dot product. The MOPs used were the $A_z$ index; partial $A_z$ ($A_{z,90}$—the top 10% of the ROC curve); specificity; and PPV at 95%, 97%, 98%, and 100% sensitivities. MOP results are depicted in Table 10.16, when the separate SVMs were trained using the reduced Duke data set and validated using the reduced DDSM USF data set.

Using the ROC $A_z$ index, the clinically relevant partial $A_z$ (equivalent to average specificity for all sensitivities above 90%), specificity at 98% sensitivity, and specificity at 95% sensitivity, a partial $A_z$ greater than 25% was achieved for all learning machines except the polynomial of degree two (Table 10.16). For 98%

<p>| Table 10.15 Five-fold cross-validation hybrid results with the reduced Duke data set. |
|---------------------------------|---------------------------------|---------------------------------|</p>
<table>
<thead>
<tr>
<th></th>
<th>1 node</th>
<th>2 nodes</th>
<th>3 nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>100% Sensitivity</td>
<td>Specificity 31.46%</td>
<td>Specificity 36.73%</td>
<td>Specificity 30.94%</td>
</tr>
<tr>
<td>PPV</td>
<td>48.19%</td>
<td>PPV 45.78%</td>
<td>PPV 47.65%</td>
</tr>
<tr>
<td>98% Sensitivity</td>
<td>Specificity 31.45%</td>
<td>Specificity 36.73%</td>
<td>Specificity 30.94%</td>
</tr>
<tr>
<td>PPV</td>
<td>48.19%</td>
<td>PPV 45.78%</td>
<td>PPV 47.65%</td>
</tr>
<tr>
<td>95% Sensitivity</td>
<td>Specificity 37.53%</td>
<td>Specificity 47.52%</td>
<td>Specificity 41.49%</td>
</tr>
<tr>
<td>PPV</td>
<td>49.70%</td>
<td>PPV 49.43%</td>
<td>PPV 51.5%</td>
</tr>
<tr>
<td>$A_z$ Index</td>
<td>77.54%</td>
<td>75.70%</td>
<td>79.31%</td>
</tr>
</tbody>
</table>
sensitivity, specificities were equal to, or above, 15% for all learning machines except the polynomials of degrees two and three. In addition, specificities were in the 27–33% range at 95% sensitivity for all learning machines except the polynomial of degree two and dot product kernels.

These results demonstrate that the learning machines can potentially “scale up” to data sets from different institutions and, consequently, have the potential capability to generalize to address interinstitutional variability. While one would prefer to obtain comparable results at 100% sensitivity, previous research demonstrated that “delayed diagnosis of probably benign cancers has a minimal impact on the treatment of those cancers.”25,26 Clearly, additional study is required to ascertain whether “delayed diagnosis for cases identified by computer models as very likely benign would have a similarly minimal impact.”26

Table 10.17 displays the results from a set of experiments where six SVM kernels were trained using the USF DDSM data set and validated with the Duke data set. With the same MOP, results show a partial $A_z$ ranging from 44% to 46% for the GRBF, S2000, and polynomial of degree-one kernels. At 98% sensitivity, the specificities were 30–38% for these same kernels; and at 95% sensitivity, the specificities improved to 41–56%. Better generalization was expected for this set of experiments because all cases in this data set were scored by radiologists at one institution.

### Table 10.17 Reduced Duke validation results with SVM kernels trained on DDSM USF data set.

<table>
<thead>
<tr>
<th>Kernel</th>
<th>$A_z$</th>
<th>100% sensitivity</th>
<th>98% sensitivity</th>
<th>97% sensitivity</th>
<th>95% sensitivity</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$A_{90}$</td>
<td>spec (%)</td>
<td>ppv (%)</td>
<td>spec (%)</td>
<td>ppv (%)</td>
</tr>
<tr>
<td>Rbf</td>
<td>0.798</td>
<td>0.286</td>
<td>0.20</td>
<td>49.82</td>
<td>15.69</td>
</tr>
<tr>
<td>S2000</td>
<td>0.793</td>
<td>0.270</td>
<td>1.41</td>
<td>50.13</td>
<td>14.29</td>
</tr>
<tr>
<td>Poly1</td>
<td>0.787</td>
<td>0.255</td>
<td>0.60</td>
<td>49.92</td>
<td>18.41</td>
</tr>
<tr>
<td>Poly2</td>
<td>0.718</td>
<td>0.182</td>
<td>0.00</td>
<td>49.77</td>
<td>7.85</td>
</tr>
<tr>
<td>Poly3</td>
<td>0.796</td>
<td>0.279</td>
<td>0.50</td>
<td>49.90</td>
<td>8.85</td>
</tr>
<tr>
<td>Dot</td>
<td>0.777</td>
<td>1.21</td>
<td>50.08</td>
<td>17.71</td>
<td>54.15</td>
</tr>
</tbody>
</table>
10.5.7 DISCUSSION

The research reported in this section used two data sets containing approximately 2500 cases, collected at different institutions. Various SVM designs, in addition to the EP/AB hybrid, showed the potential to address problems of interinstitutional variability and “scaling up” to larger data sets. Both types of paradigms achieved almost the same performance level using 6 BI-RADS™ discriminators plus age when compared to the “full up” complement of 10 BI-RADS™ and 6 clinical history discriminators. Moreover, the EP-derived SVMs generally performed with the same accuracy as the SVMs whose parameters were derived by classical numerical iterative methods. Use of the EP approach to derive the SVMs showed several advantages. There was a significant reduction of “user” interaction, resulting in hours to complete the task, as opposed to a week, which is required for the numerical iterative approach. The general theory has wide applicability to several kinds of environments and could be used by other researchers regardless of the application. In addition, comparable classification performance on the test cases verified that the new, faster approach to SVM derivation does not degrade performance of the established, but cumbersome, approach. Results of the current research demonstrate clinical feasibility of developing SVM-derived software to assist in breast cancer detection.

ACKNOWLEDGMENTS

The work reported in this section was supported by the New York State Breast Cancer Research and Education Fund through Department of Health Contract C017939. The following Binghamton University Department of Computer Science graduate students also are acknowledged for their contributions: Ming Kuo Song, Mohan Mathews, Yatendra Khandelwal, Daniel Goldman, Nirav Patel, Sushant Shah, Arpan Acharya, V.M. Bharathi, Senthil Mehalingam, Avinash Sridhar, and Richard Gonzalez, along with undergraduate student James B. McLaughlin.

10.6 A SUPPORT VECTOR MACHINE (SVM) GENERALIZED REGRESSION NEURAL NETWORK (GRNN) ORACLE

This section describes a more complex application of SVMs, called an SVM generalized regression neural network (GRNN) oracle hybrid. Until recently, the standard technique for dealing with multiple prediction models was to compare their performance and select the model that performed best, discarding the others. It is now well known that results almost always are better if all the models are used and their predictions intelligently combined. The authors designed and performed preliminary evaluation of a GRNN oracle that uses gate variables to intelligently combine outputs of the competing models. This GRNN oracle was developed to obtain accurate diagnoses of difficult cases, which may be embedded in a large
data set containing cases from many institutions. Once the expected errors of individual diagnostic models are estimated, they are used to compute weights for each model. When an unknown case is processed, gate variables are used by the GRNN oracle to decide which diagnostic models are likely to be best for that particular case. The chosen models are then weighted more heavily than likely inferior models. Selection of the proper gate variables is essential to successful operation of the oracle.

The final prediction of the GRNN oracle is a linear combination of the competing models, which have the desirable property of unbiased predictions. The linear combination of unbiased estimators having minimum mean squared error uses weights proportional to the reciprocal of each estimator’s variance. When the predicted squared error is used in place of the variance, it may be shown that the models with the largest mean squared errors have the smallest effect on the GRNN oracle’s prediction, while the models with the smallest mean squared error have the largest effect on the oracle’s output. Consequently, because this oracle is an asymptotically optimal predictor and fully nonlinear, the error predictions can be expected to be very good, so long as one has representative training data. Since the error predictions are used in an optimal manner, in general, it can be expected that the final predictions will be superior to those obtained from any single model.

10.6.1 Mathematical Foundations of the PNN, GRNN, and the GRNN Oracle

To understand the GRNN oracle, one first must understand the probabilistic neural network (PNN) and the GRNN.

10.6.1.1 PNN

The PNN is a classifier that makes effective use of the shapes of population distributions. Given a point that previously has not been classified, the PNN determines the probability that the point belongs to a particular class, based on populations of points whose class identity is known a priori. The PNN makes this determination in two steps. First, it uses the multivariate Parzen density to estimate (a constant multiple of) the density function of each population at the known point. It then computes the Bayesian probability of membership in each population and assigns the unknown to the class having the highest probability.

Suppose there is a training set composed of \( n \) cases. Each case \( i (i = 1, \ldots, n) \) consists of \( p \) predictor variables: \( x_{ij}, j = 1, \ldots, p \). These predictor variables in some way determine the relative efficacy of the prediction models. The observed values of predictor variables for the unknown case are: \( x_j, j = 1, \ldots, p \). The weighted Euclidean distance function often is employed as

\[
D_x(x, x_i) = \sum_{j=1}^{p} \left( \frac{x_j - x_{ij}}{\sigma_j} \right)^2.
\]

(10.17)
The (unnormalized) density function then is given by Parzen’s formula as

\[ g(x) = \frac{1}{n} \sum_{i=1}^{n} \exp[-D(x, x_i)]. \]  

(10.18)

This procedure is repeated for each of the \( K \) classes, giving the unnormalized density \( g_k(x) \) for \( k = 1, \ldots, K \). The Bayesian probability that the unknown case was drawn from class \( k \) is as follows:

\[ P[x \in k] = \frac{g_k(x)}{\sum_i g_i(x)}. \]  

(10.19)

From this formulation, one can see that the PNN has three useful properties. First, under reasonable assumptions and as the training set size increases, the PNN is an asymptotically optimal classifier. Second, the PNN is fully nonlinear. It neither imposes constraints of linearity on the model (as does ordinary discriminate analysis) nor requires linear separability like many other classifiers. Third, the PNN has the ability to compute Bayesian probabilities, including use of priors if desired, which is a great advantage in many practical applications.

10.6.1.2 The GRNN

The GRNN is essentially a PNN that has been modified to perform as a general function mapper rather than classifier. A vector \( x \) of \( p \) independent variables is being used to predict a dependent scalar variable \( y \). If one knows the joint density of these quantities, the prediction having minimum expected squared error is given by the conditional expectation

\[ E_{Y|X} = \frac{\int_{-\infty}^{\infty} y \cdot f_{XY}(x, y)dy}{\int_{-\infty}^{\infty} f_{XY}(x, y)dy}. \]  

(10.20)

In reality, this density is estimated by treating the quantity \((x, y)\) as a single vector and using Parzen’s method to estimate the joint density based on a set of training data. The mathematics are easiest (and performance usually best) if a Gaussian kernel is employed. Most applications require separate sigma weights for each independent variable. This implies the distance functions

\[ D_x(x, x_i) = \sum_{j=1}^{p} \left( \frac{x_j - x_{ij}}{\sigma_j} \right)^2 \]  

(10.21)

and

\[ D_y(y, y_i) = \left( \frac{y - x_i}{\sigma_y} \right)^2, \]  

(10.22)
where \( \mathbf{x} \) is the \( i \)th training vector. Parzen’s estimator of the joint density is given by
\[
g(\mathbf{x}, y) = \frac{1}{nc_x c_y} \sum_{i=1}^{n} \exp[-D_x(\mathbf{x}, \mathbf{x}_i) - D_y(y, y_i)]
\]
\[
= \frac{1}{nc_x c_y} \sum_{i=1}^{n} \exp[-D_x(\mathbf{x}, \mathbf{x}_i)] \exp[-D_y(y, y_i)].
\tag{10.23}
\]

The two normalizing constants that appear in Eq. (10.23) are needed to ensure that the joint density integrates to unity. The normalizer for \( y \) is
\[
c_Y = \int_{-\infty}^{\infty} \exp(-D_y(y, 0)) dy
\tag{10.24}
\]
and the normalizer for \( x \) is defined in the corresponding multivariate way.

When the predictor shown in Eq. (10.20) is modified by replacing the exact densities with the Parzen approximators shown previously, one arrives at
\[
\hat{y}(\mathbf{x}) = \frac{N(\mathbf{x})}{D(\mathbf{x})}.
\tag{10.25}
\]
The numerator and denominator of this expression are given by
\[
N(\mathbf{x}) = \int_{-\infty}^{\infty} \sum_{i=1}^{n} \exp[-D_x(\mathbf{x}, \mathbf{x}_i)] \exp[-D_y(y, y_i)] dy
\]
\[
= \frac{1}{nc_x c_y} \sum_{i=1}^{n} \exp[-D_x(\mathbf{x}, \mathbf{x}_i)] \int_{-\infty}^{\infty} y \cdot \exp[-D_y(y, y_i)] dy
\]
\[
= \frac{1}{nc_x} \sum_{i=1}^{n} y_i \exp[-D_x(\mathbf{x}, \mathbf{x}_i)]
\tag{10.26}
\]
and
\[
D(\mathbf{x}) = \int_{-\infty}^{\infty} \frac{1}{nc_x c_y} \sum_{i=1}^{n} \exp[-D_x(\mathbf{x}, \mathbf{x}_i)] \exp[-D_y(y, y_i)] dy
\]
\[
= \frac{1}{nc_x c_y} \sum_{i=1}^{n} \exp[-D_x(\mathbf{x}, \mathbf{x}_i)] \int_{-\infty}^{\infty} \exp[-D_y(y, y_i)] dy
\]
\[
= \frac{1}{nc_x} \sum_{i=1}^{n} \exp[-D_x(\mathbf{x}, \mathbf{x}_i)],
\tag{10.28}
\]
respectively.
When the expressions for the numerator and the denominator are inserted into Eq. (10.25), the constants cancel each other and the fundamental equation of the GRNN

\[ \hat{y}(x) = \frac{\sum_{i=1}^{n} y_i \exp[-D_x(x, x_i)]}{\sum_{i=1}^{n} \exp[-D_x(x, x_i)]} \]  

(10.29)

is left.

10.6.1.3 GRNN-based oracle

An overview of the GRNN-based oracle is presented in Fig. 10.8.

Suppose there are two or more prediction models of any type, each of which predicts the same scalar output variable. If there are multiple outputs, a separate oracle should be used for each. Neither the nature of these models nor their inputs is important. One or more gate variables, whose values presumably have an effect on deciding which of the competing models is most valid, are important. For present purposes, the individual prediction models are assumed to be black boxes that work reasonably well.

The authors’ goal was to design an oracle that used gate variables to intelligently combine the outputs of competing models. Once the expected error of each prediction model is estimated, these expected errors are used to compute the weights for each model. When an unknown case is processed, the gate variables

Figure 10.8 GRNN oracle.
are used by the GRNN to decide which models are likely to be best for that particular case. These models are weighted more heavily than the likely inferior models. In particular, with a training set composed of \( n \) cases, each case \( i \) \((i = 1, \ldots, n)\) consists of \( p \) gate variables: \( x_{i,j} \) where \( j = 1, \ldots, p \). These gate variables in some way determine the relative efficacy of the prediction models. The \( m \) competing prediction models provide outputs \( q_{i,k} \) where \( k = 1, \ldots, m \). The desired output (target value) is \( y_i \). For the gate variables and model outputs, only one subscript is used when referring to a trial case that is to be evaluated: \( x_j \) where \( j = 1, \ldots, p \), are the values of the observed gate variables, and \( q_k \) where \( k = 1, \ldots, m \) are the computed outputs of the \( m \) competing prediction models.

The weighted Euclidean distance (as determined by the gate variables) is defined between training case \( i \) and the trial case as in Eq. (10.21). The GRNN’s predicted squared error for model \( k \) may be derived as

\[
\hat{e}_k(x) = \frac{\sum_{i=1}^{n} (y_i - q_{i,k})^2 \exp[-D_x(x, x_i)]}{\sum_{i=1}^{n} \exp[-D_x(x, x_i)]}. \tag{10.30}
\]

It is desired that final prediction be a linear combination of the outputs of the competing models, as

\[
\hat{y}(x) = \sum_{k=1}^{m} w_k q_k. \tag{10.31}
\]

If the models have the (desirable) property that their predictions are unbiased, this property is maintained if and only if the following condition is imposed:

\[
\sum_{k=1}^{m} w_k = 1. \tag{10.32}
\]

The linear combination of unbiased estimators having minimum mean squared error uses weights proportional to the reciprocal of each estimator’s variance. If the predicted squared error is used in place of the variance, the following formula for the weights is derived:

\[
w_k = \frac{1/\hat{e}_k}{\sum_{\ell=1}^{m} 1/\hat{e}_\ell}. \tag{10.33}
\]

The GRNN is trained (e.g., the \( p \) sigma weights are optimized) in the usual cross-validation manner. To evaluate the quality of a sigma vector, a case is removed from the training set and the formulas previously shown are used first to estimate the competing models’ errors, then compute the \( w_k \) weights, weight the competing models to get the grand prediction, and compute the error of this grand prediction (it minus the true value of the omitted case). This procedure is repeated for each training case. The sigma vector providing minimum root-mean-square
(RMS) error is then found. A powerful hybrid training method combining gradient descent and differential evolution has been used to effectively train the sigma values of GRNN oracles.\textsuperscript{27}

### 10.6.2 Results and Description of SVM Models

This section describes an independent analysis that verifies results previously presented by Land et al.\textsuperscript{28} The GRNN oracle inputs were obtained from several different SVM kernels, including polynomial kernels of degrees 1 through 3, Gaussian radial basis function (GRBF), spline kernels of degrees 1 through 3, and sum and product kernels based on polynomial and GRBF kernels.

Table 10.18 depicts the results for a differential evolution (DE) crossover constant of 0.8. DE evolution is an extension of genetic algorithms (GAs), and was used to train the GRNN oracle. Differential evolution is similar to ordinary genetic optimization in that it starts with a collection of parameter sets that will be called the source population. The individuals comprising this population are combined with each other via crossover and are subjected to mutation to produce members of the destination population. Members of the destination population, taken as a group, are generally expected to be superior to members of the source population. By repeating this process enough times, the best member of the final population is anticipated to be close to the global optimum. More information on DE may be found in Land et al.\textsuperscript{28}

Table 10.18 shows that the DE crossover constant had little, if any, effect on the MOP. An $A_Z$ of approximately 82\% was obtained. A partial $A_Z$ of 32\% was obtained, which means that on average, approximately 432 women could have avoided biopsy. In addition, a specificity of approximately 5.6\% was obtained at 100\% sensitivity. This increased to approximately 36\% when sensitivity was decreased to 95\%. Finally, a PPV of 51\% was achieved at 100\% sensitivity. PPV

<table>
<thead>
<tr>
<th>Gate variables</th>
<th>Age/ mass margin</th>
<th>Age/ mass margin</th>
<th>Age/ mass margin</th>
</tr>
</thead>
<tbody>
<tr>
<td>Crossover constant</td>
<td>0.1</td>
<td>0.8</td>
<td>0.9</td>
</tr>
<tr>
<td>$A_Z$</td>
<td>0.812778</td>
<td>0.8161822</td>
<td>0.814936</td>
</tr>
<tr>
<td>$A_{.90}$</td>
<td>0.320024</td>
<td>0.320082</td>
<td>0.3200318</td>
</tr>
<tr>
<td>100% spec</td>
<td>0.0559456</td>
<td>0.0559456</td>
<td>0.0523343</td>
</tr>
<tr>
<td>100% ppv</td>
<td>0.516704</td>
<td>0.4822912</td>
<td>0.485322</td>
</tr>
<tr>
<td>98% spec</td>
<td>0.286576</td>
<td>0.231224</td>
<td>0.231224</td>
</tr>
<tr>
<td>98% ppv</td>
<td>0.550725</td>
<td>0.5266578</td>
<td>0.5277578</td>
</tr>
<tr>
<td>97% spec</td>
<td>0.306578</td>
<td>0.253323</td>
<td>0.244365</td>
</tr>
<tr>
<td>97% ppv</td>
<td>0.534545</td>
<td>0.5323213</td>
<td>0.5323213</td>
</tr>
<tr>
<td>95% spec</td>
<td>0.365421</td>
<td>0.3237252</td>
<td>0.3237252</td>
</tr>
<tr>
<td>95% ppv</td>
<td>0.569588</td>
<td>0.5539338</td>
<td>0.5543545</td>
</tr>
</tbody>
</table>
Table 10.19 Comparison of GRNN oracle and SVM performance.

<table>
<thead>
<tr>
<th>Difference (GRNN-SVM) (%)</th>
<th>GRNN (%)</th>
<th>SVM (%)</th>
<th>% Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Az</td>
<td>81.612</td>
<td>81.22</td>
<td>0.41</td>
</tr>
<tr>
<td>Az 90</td>
<td>32.02</td>
<td>29.61</td>
<td>2.40</td>
</tr>
<tr>
<td>100% spec</td>
<td>5.55</td>
<td>3.27</td>
<td>2.29</td>
</tr>
<tr>
<td>100% ppv</td>
<td>48.23</td>
<td>47.59</td>
<td>0.64</td>
</tr>
<tr>
<td>98% spec</td>
<td>22.15</td>
<td>17.85</td>
<td>4.30</td>
</tr>
<tr>
<td>98% ppv</td>
<td>52.67</td>
<td>51.33</td>
<td>1.39</td>
</tr>
<tr>
<td>97% spec</td>
<td>24.23</td>
<td>23.46</td>
<td>0.77</td>
</tr>
<tr>
<td>97% ppv</td>
<td>53.16</td>
<td>52.81</td>
<td>0.35</td>
</tr>
<tr>
<td>95% spec</td>
<td>32.38</td>
<td>29.44</td>
<td>2.94</td>
</tr>
<tr>
<td>95% ppv</td>
<td>56.96</td>
<td>54.27</td>
<td>2.69</td>
</tr>
</tbody>
</table>

increased to 56% when the sensitivity was decreased to 95%. The results of this independent analysis are consistent with those described in Land et al.28 and, thus, verify operation of the SVM/GRNN hybrid using the integrated mammogram data set.

Table 10.19 depicts performance of the oracle as compared to that of the best performing SVM. A 71% performance improvement (from 3.27% to 5.55%) is noted in specificity at 100% sensitivity, while a 24% performance improvement (from 17.8% to 22.5%) is observed in specificity at 98% sensitivity. In general, improvements ranging from 0.5% to 24% were observed when performance of the GRNN oracle was compared to that of the SVM. This is not surprising when one observes that SVMs can be trained to a global minimum, given that learning machine parameters are properly selected. These results also are consistent with results reported in Land et al.28

10.6.3 DISCUSSION

The main objective of the research reported in this section was to perform an independent analysis to verify previous results.28 This included comparing performance of the GRNN oracle with the best performing SVM. An improvement from approximately 0.5% to 24% in diagnostic accuracy was observed when the GRNN oracle was compared to the SVM. These results also are consistent with results reported previously.28

ACKNOWLEDGMENTS

The work reported in Section 10.6 was supported in part by the New York State Breast Cancer Research and Education Fund through Department of Health Contract C017939.
10.7 Partial Least Squares (PLS) and Kernel-Partial Least Squares (K-PLS)

Currently, there are several commercially available automated imaging-checking systems that are used to aide radiologists with interpretation of screen film mammograms (SFMs). These systems show considerable promise, but also are associated with false-positive (FP) interpretation errors.\textsuperscript{29,30} To the best of the authors’ knowledge, there are no CAD systems commercially available that provide diagnostic interpretations rather than visual alerts.

Although screening mammography is useful for detecting nonpalpable breast cancers, most such lesions that are biopsied will be benign.\textsuperscript{31} After 10 mammograms, evidence indicates that the cumulative false-positive probability for screening mammography is about 49\%, and about 19\% of women without breast cancer will have undergone an unnecessary biopsy.\textsuperscript{32} This study also showed that for every $100 spent on screening, an additional $33 was spent related to the evaluation of false-positive lesions.\textsuperscript{31}

In this section, the authors present initial results for two automated classifiers based on PLS methodology. The input features to the classifier are BI-RADS\textsuperscript{TM} mass descriptors, as obtained from radiologists’ interpretations, family history of breast cancer, and age. Inclusion of the BI-RADS\textsuperscript{TM} descriptors and medical history in automated classification processing was previously called “case-based reasoning” by Floyd et al.,\textsuperscript{31} and several applications of computational intelligence to breast cancer screening have been published, many of which were discussed in earlier sections of this chapter.

Two novel approaches, PLS and K-PLS, were applied to CAD of breast cancer. This approach is based on optimization of the PLS algorithm for linear regression and the K-PLS algorithm for nonlinear regression. This section contains a discussion of the background of PLS and K-PLS, MOP frequently used in regression analysis, and preliminary results obtained by applying these approaches to the CAD of breast cancer.

10.7.1 Background

10.7.1.1 PLS regression and K-PLS

PLS regression was conceived by the Swedish statistician Herman Wold for econometrics modeling of multivariate time series. The first PLS publication was a sociology application in 1975.\textsuperscript{33} Herman’s son, Svante Wold, applied PLS to chemometrics in the 1980s.\textsuperscript{34,35} Currently, PLS is one of the most popular and powerful tools in chemometrics, mainly because of the quality of building models using many variables with colinearity. The mathematics behind PLS is not transparent and is outlined in Wold.\textsuperscript{35}

PLS can be viewed as a better version of principal component analysis, where the data first are transformed into a different and nonorthogonal basis, similar to
principal component analysis (PCA), and only a few (the most important) PLS components (or latent variables) are considered for building a regression model. The difference between PLS and PCA is that the new set of basis vectors (similar to the eigenvectors of \( X^T X \) in PCA) is not a set of successive orthogonal directions that explain the largest variance in the data, but actually are a set of conjugate gradient vectors to the correlation matrices that span a Krylov space.\(^{36}\) As with PCA, the basis vectors can be peeled off from the data matrix \( X \) successively in the nonlinear iterative partial least squares (NIPALS) algorithm\(^{37}\) by approximating \( X \) as \( X = TP^T \) (similar to \( X = TB \) in PCA, where \( T \) contains the scores and \( B \) is a matrix of eigenvectors corresponding to the largest eigenvalues for \( X^T X \)).

PLS regression is one of the most powerful data-mining tools for large data sets with an overabundance of descriptive features. The NIPALS implementation of PLS is elegant and fast. One of the unique characteristics of PLS is the ability to reconstruct logical flow implications and distinguish cause and consequence relationships from a database.\(^{38}\)

What makes PLS especially interesting for CAD and data-mining applications is a recent extension to nonlinear PLS or kernel PLS,\(^{39-41}\) which incorporates a kernel transform, similar to SVMs. The nonlinear K-PLS has recently been explained in the context of NNs (i.e., perceptrons and radial basis functions).\(^{39}\)

The K-PLS method not only can be reformulated to resemble SVMs, but it also can be interpreted as a kernel and centering transformation of the descriptor data followed by a regular PLS method.\(^{40}\) K-PLS was first introduced in the context of working with linear kernels on data sets with more descriptor fields than data to make the PLS modeling more efficient.\(^{42}\) Early applications of K-PLS were done mainly in this context.\(^{43-45}\)

10.7.1.2 Regression models based on direct kernels

Kernel transformation is an elegant way to make a regression model nonlinear. A kernel is a matrix containing similarity measures in a data set, either between the data of the data set itself, or with other data (e.g., support vectors\(^{46,47}\)). A classical application of a kernel is the correlation matrix used for determining the principal components in principal component analysis, where the feature kernel contains linear similarity measures between (centered) attributes. In SVMs, the kernel entries are similarity measures between data, rather than features, and these similarity measures usually are nonlinear, unlike the dot product similarity measure used previously to define a kernel. There are many possible nonlinear similarity measures, but to be mathematically tractable, a kernel has to satisfy certain conditions, the so-called Mercer conditions\(^{46,47}\)

\[
\kappa_{nn} = \begin{bmatrix}
    k_{11} & k_{12} & \cdots & k_{1n} \\
    k_{21} & k_{22} & \cdots & k_{2n} \\
    \vdots & \vdots & \ddots & \vdots \\
    k_{n1} & k_{n2} & \cdots & k_{nn}
\end{bmatrix}
\]

(10.34)
This expression introduces the general structure for a data kernel matrix, $K_{nn}$, for $n$ data. The kernel matrix is a symmetrical matrix, where each entry contains a similarity (linear or nonlinear) between two data vectors. There are many different possibilities for defining similarity metrics, such as the dot product, which is a linear similarity measure, and the radial basis function (RBF) kernel, which is a nonlinear similarity measure. The RBF kernel is the most widely used nonlinear kernel, and the kernel entries are defined by

$$k_{ij} = e^{-\frac{\|\mathbf{x}_j - \mathbf{x}_i\|^2}{2\sigma^2}}. \quad (10.35)$$

Note that in the kernel definition, the kernel entry contains the square of the Euclidean distance (or two norm) between data points, which is a dissimilarity measure (rather than a similarity) in a negative exponential. The negative exponential also contains a free parameter $\sigma$, which is the Parzen window width for the RBF kernel. The proper choice for selecting the Parzen window is usually determined by an additional tuning; also hyper tuning, on an external validation set.

In this discussion the kernel transformation is applied as a data transformation in a separate preprocessing stage. The data actually are replaced by a nonlinear data kernel, and a traditional linear predictive model is applied. Methods where a traditional linear algorithm is used on a nonlinear kernel transformation of the data are introduced in this section as “direct kernel methods.” The elegance and advantage of such methods are that the nonlinear aspects of the problem are captured entirely in the kernel and are transparent to the applied algorithm. If a linear algorithm was used before introducing the kernel transformation, the required mathematical operations remain linear. This shows how linear methods such as principal component regression, ridge regression, and PLS can be turned into nonlinear direct kernel methods by using exactly the same algorithm and code; only the data are different, and the kernel transformation of the data is operated on, rather than the data themselves.

To make out-of-sample predictions on true test data, a similar kernel transformation must be applied to the test data. The idea of direct kernel methods is illustrated in Fig. 10.9, by showing how any regression model can be applied to kernel-transformed data. One could also represent the kernel transformation in a NN type of flow diagram and the first hidden layer would yield the kernel-transformed data, and the weights in the first layer would be only the descriptors of the training data. The second layer contains the weights that can be calculated by applying PLS. When an RBF kernel is used, this type of NN would look very similar to a RBF NN, except that the weights in the second layer are calculated differently.

10.7.1.3 Dealing with bias: centering the kernel

There is an important detail required to make direct kernel methods work. By applying weights to the kernel-transformed data, there is no constant offset term or bias. The bias term does not have to be explicitly incorporated in the model when
Kernel centering can be summarized as follows. A straightforward way for kernel centering is to first subtract the average from each column of the training data kernel, and store this average for later recall, when centering the test kernel. The second step is going through the newly obtained vertically centered kernel again, row by row, and subtracting the row average form each horizontal row.

The kernel of the test data must be centered in a consistent way, following a similar procedure. In this case, the stored column centers from the kernel of the training data will be used for the vertical centering of the kernel of the test data. The vertically centered test kernel then is centered horizontally—for each row. The average of the vertically centered test kernel is calculated and each horizontal entry of the vertically centered test kernel is substituted by that entry minus the row average.

The advantage of the kernel-centering algorithm recipe introduced in this section is that it also applies to rectangular data kernels. The flow chart for preprocessing the data, applying a kernel transform on this data, and centering the kernel for the training, validation, and test data is shown in Fig. 10.10.

10.7.1.4 Metrics for assessing the model quality

Generally, ROC curves are used as MOPs to assess CAD performance, including $A_z$ index and partial $A_z$ ($A_z, 90$). However, regression analysis uses other MOPs, which may not be as familiar to the reader as the MOP previously described. Two popular regression MOPs are: $r^2$, defined as the correlation coefficient squared between the target values and the regression prediction, and the “Press R squared,” or “goodness,” denoted by $R^2$. For completeness, these MOPs and their derivative definitions are described in this section.
In the case of a classification problem that would be relatively easy, one would present the number of hits and misses in the form of a confusion matrix. For a regression problem, a common way to capture the error is by the root-mean-square error (RMSE) index which is defined as the average value of the squared error (either for the training set or the test set) according to

$$\text{RMSE} = \sqrt{\frac{1}{n} \sum_i (\hat{y}_i - y_i)^2}. \quad (10.36)$$

Although the RMSE is an efficient way to compare the performance of different prediction methods on the same data, it is not an absolute metric in the sense that the RMSE will depend on how the response for the data was scaled. To overcome this handicap, additional error measures can be introduced that are less dependent on the scaling and magnitude of the response value. One metric that can be used for assessing the quality of a trained model is $r^2$ according to

$$r^2 = \frac{\sum_{i=1}^{n_{\text{train}}} (\hat{y}_i - \bar{y})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n_{\text{train}}} (\hat{y}_i - \bar{y})^2 \cdot \sum_{i=1}^{n_{\text{train}}} (y_i - \bar{y})^2}}, \quad (10.37)$$

where $n_{\text{train}}$ represents the number of data points in the training set, $r^2$ takes values between zero and unity, and the higher the $r^2$ value, the better the model. An obvious drawback of $r^2$ for assessing model quality is that $r^2$ only expresses a linear correlation, indicating how well the predictions follow a line if $\hat{y}$ is plotted as a function of $y$. Although one would expect a nearly perfect model when $r^2$ is unity, this is not always the case. A second and more powerful measure to assess the quality of a trained model is $R^2$, often used in chemometric modeling, which
is defined as
\[ R^2 = 1 - \frac{\sum_{i=1}^{n_{\text{train}}} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{n_{\text{train}}} (y_i - \bar{y})^2}. \] (10.38)

\( R^2 \) is considered a better measure than \( r^2 \) because it also accounts for residual error. Higher values for \( R^2 \) imply better models. In certain cases, the \( R^2 \) metric actually can be negative. The \( R^2 \) metric is generally smaller than \( r^2 \). For large data sets, \( R^2 \) tends to converge to \( r^2 \), and the comparison between \( r^2 \) and \( R^2 \) for such data often reveals hidden biases.

For assessing the quality of a validation set or test set, the authors will introduce similar metrics, \( q^2 \) and \( Q^2 \), where \( q^2 \) and \( Q^2 \) are defined as \( 1 - r^2 \) and \( 1 - R^2 \) for the data in the test set. For a model that perfectly predicts on the test data, one would expect \( q^2 \) and \( Q^2 \) to be zero. The reason for introducing metrics that are symmetric between the training set and the test set is to avoid confusion. \( Q^2 \) and \( q^2 \) values will always apply to a validation set or a test set, and one would expect these values to be quite low for a good predictive model. \( R^2 \) and \( r^2 \) values will always apply to training data, and should be close to unity for a good training model. An additional metric that can be useful for classification problems is the area under the ROC curve.\(^{52,53}\) This metric can be extended to regression problems as well.\(^{48}\)

### 10.7.1.5 Data conditioning and preprocessing

It is customary in predictive modeling to normalize or Mahalanobis scale all the data first before any further operations are attempted. Normalizing refers to subtracting the average from each of the descriptive model features or attributes, and dividing each feature entry by the standard deviation. This procedure must be carried out consistently for training and validation, in addition to test data, as outlined in Fig. 10.10. The division by the standard deviation in Mahalanobis scaling has the advantage that the model becomes independent of the metric in which the data were expressed. The centering of the data aspect in Mahalanobis scaling is not always desirable, and generally depends on the nature of the data and the machine-learning model. It has been observed by the authors that data centering in large data sets with many sparse binary features can lead to a serious deterioration of the model. Whether or not data should be centered before proceeding with the kernel transformation can be determined heuristically by observing model performance on an external validation set, with and without centering.

### 10.7.2 Scaled Moffitt data set

The feasibility study described in this section represents a standard case-control design with the aim of making benign-malignant predictions. The cases are from women with biopsy-verified malignant masses and benign breast lesions. For this
work, breast masses define the lesion, while the presence of calcifications, singular or clustered, is excluded. The benign controls consisted of cases determined to be benign by biopsy and those determined to be benign after two years of follow-up without biopsy. A total of 175 cases were used. There were 55 malignant cases, 40 benign controls determined by biopsy, and 80 benign controls followed up without biopsy. The data were collected from the diagnostic center at the Moffitt Cancer Center & Research Institute.

The standard BI-RADS™ descriptors used as input features were mass shape, mass margin, overall breast composition, and density. In addition to these BI-RADS™-derived features, the presence or absence of a halo around a tumor, family history of breast cancer, and age were used as input features. Mass shape is a five-category rating: round, oval, lobulated, irregular, and architectural distortion. Mass margin is a five-category rating: circumscribed-well defined, microlobulated, indistinct, obscured, or speculated. Density is a four-category rating in the vicinity of the mass relative to the surrounding area: high, low, equal, or fat-containing radiolucent. Overall breast composition is a four-category rating: entirely fat, scattered fibroglandular tissue, heterogeneously dense, and extremely dense.

10.7.3 Preliminary results

Table 10.20 depicts the results achieved from the SVM kernels, PLS, and K-PLS agents. K-PLS has an $A_z$ value of 0.907, which is better than PLS and almost as good as the other SVM kernels. The K-PLS $A_{z90}$ value is much better than all of the agents at 0.612, except the SVM-Poly kernel at 0.638. K-PLS still is faster than all of the SVM kernels and can accurately learn, using data sets that contain large nonlinear components. Note that all of the learning-machine paradigms significantly outperformed the standard three-layer NN trained by back propagation.

The ROC curves in Figs. 10.11 and 10.12 for PLS and K-PLS, respectively, are somewhat jagged because of the small validation data set sizes. However, the agents still learn and predict correctly. These curves are expected to become much smoother as the data-set size increases. Note that the $R^2$ is 0.89 for the PLS ROC curve, while the comparable value is 0.9 for the K-PLS ROC curve.

<table>
<thead>
<tr>
<th></th>
<th>KPLS</th>
<th>PLS</th>
<th>SVM</th>
<th>SVM</th>
<th>SVM</th>
<th>Neural network</th>
</tr>
</thead>
<tbody>
<tr>
<td>MOP $\sigma = 4.8$</td>
<td>S2000</td>
<td>Rbf</td>
<td>polynomial</td>
<td>3-layer ANN trained by back propagation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$A_z$</td>
<td>0.907</td>
<td>0.892</td>
<td>0.907</td>
<td>0.923</td>
<td>0.918</td>
<td>0.80</td>
</tr>
<tr>
<td>$A_{z90}$</td>
<td>0.612</td>
<td>0.55</td>
<td>0.494</td>
<td>0.571</td>
<td>0.638</td>
<td>0.80</td>
</tr>
</tbody>
</table>
Once again, the ROC curves for all three of the SVMs were not very good when the smaller data set was used. The $A_z$ results, however, are well within reason and are accurate. The SVM-RBF kernel performed the best of all the SVMs with an $A_z$ value of 0.923. Fig. 10.13 depicts the average ROC $A_z$ values of the fivefolds for the s2000, rbf, and polynomial kernels.

10.7.4 DISCUSSION

The results described in this section demonstrate that both the K-PLS and PLS paradigms achieved comparable results when compared to three separate SVMs, where the SVMs were known to be trained to a global minimum, using the same data set. The average performance of the three separate SVMs were $A_z = 0.9168$ and an average partial $A_z (A_z;90) = 0.5684$. These results compare favorably with the K-PLS paradigm, which obtained an $A_z = 0.907$ and partial $A_z = 0.6123$. The PLS paradigm provided comparable results. Of note is that both the K-PLS and
PLS paradigms outperformed the ANN by about 14%, with the best $A_z$ index obtained $A_z \approx 0.907$ compared to the ANN $A_z \approx 0.8$. The K-PLS and PLS results are expected to improve as the authors become more proficient with the theory supporting the new paradigms. Preliminary “Press R squared” values were 0.8911 for the PLS ROC curve and 0.9 for the K-PLS ROC curve. Both of these values are in agreement with other MOP results used to evaluate these new CAD algorithms.

Finally, as in all detection problems, the false-positive (FP) and true-positive (TP) rates were related, in that reducing the FP rate normally reduces the TP rate and vice versa. A classification also may be considered as a detection task of finer discrimination. Therefore, the long-term goal of the PLS-based classification work presented in this section is based on the premise of addressing two important problems in mammography today. First, an automated classification stage of processing may be added to a given automated checking system to reduce the FP interpretation rate. Second, a real-time classification system may be used to reduce the number of screening mammogram interpretations that result in recommendations for further
diagnostics, independent of the initial detection scheme. Future work will include further testing of these classifiers with larger data sets, in addition to including a more complete set of risk factors for breast cancer.

ACKNOWLEDGMENTS

The research presented in Section 10.7 was supported in part by the New York State Breast Cancer Research and Education Fund through Department of Health Contract C017939. It also was supported in part by the NCI under contract number R21CA082639.

10.8 ARTIFICIAL INTELLIGENCE, KNOWLEDGE REPRESENTATION, AND KNOWLEDGE ENGINEERING

The objective of this section is to demonstrate that one can formulate a CAD artificial intelligence (AI) system as a set of replaceable modular units, where specific components of these modular may contain ANNs; EP-derived components; multivariate PNNs, SVMs, and ROC components used for performance evaluation; etc. Also included is a research study to demonstrate the value of using both knowledge representation (KR) and knowledge engineering (KE) to “clean” a screen film mammogram database.
10.8.1 OVERVIEW OF MODULAR ARTIFICIAL INTELLIGENCE SYSTEM

This section highlights the components of the AI system designed for this research. These components are: (1) environment, “cleaned” by KR, KE, and environment findings; (2) agents and training methods; and (3) diagnostic results measured by ROC analyses.

As depicted in Fig. 10.14, a modular AI design provides a modular replacement for each of the components previously described. Another data set could replace the one used in the research described in this section, but one must ensure that elements in the data set are properly configured for the agent and training methods currently comprising these replaceable system components. For example, if an EP family of derived three-layer ANNs represents the agents, the data set must be scaled to the activation function used in these ANNs. If the agent is replaced by an EP/adaptive boosting (EP/AB) hybrid or a specific SVM kernel, then the elements of the data set must be scaled to comply with the different activation function used by the EP/AB hybrid or SVM kernel. Note that this last example replaced not only the agent but also the training method. In like manner, the data set can remain the same and the agent and training method can be modularly replaced. An example of this would be replacing either the EP or EP/AB hybrid by a PNN, which would require that the training set of cases first be grouped into benign and malignant topologies, as required by PNN processing, in addition to addressing the scaling problem. Finally, regardless of the environment, agent, and/or training modular component used, the ROC curves may be used as clinically relevant MOPs.

**Figure 10.14** Overview of AI system.
10.8.2 ENVIRONMENT

The environment consisted of two original data sets: USF DDSM containing six BI-RADS\textsuperscript{17} indicators and patient age, and the Duke data set containing ten BI-RADS\textsuperscript{TM} indicators and six clinical-history variables. Appropriate variables were extracted from the Duke data set to match it with the USF DDSM data set. Variables used in this study are shown in Table 10.21.

A KE activity insures that information for use with intelligent systems has been suitably identified and transformed as knowledge-domain discriminators appropriate to the environment, agent, and eventual use. Based on the error levels encountered with the breast cancer data, it was decided to reexamine the data sets with respect to the environment and agents. Both data sets had to be “cleaned.”

With the Duke data set, each value in each case was tracked back to the original data set. These values were verified to insure that they were representative of the transcribed data set. Much effort was expended during the generation of the original data set to remove cases where the information was incomplete or inconclusive. Dependent on the agent and the learning method, data-set values were transformed from the native coding to values appropriate for training. These values were further analyzed with regard to appropriateness of the transformations performed.

10.8.2.1 Environment findings

The original Duke data set was found to correctly represent the transcribed data. No obvious transcription or transformation errors were detected. Previous research using this data set showed that of the 16 inputs, only seven were significant in training of the system. The original Duke data set was transformed to the revised Duke data set using the seven significant inputs.

The revised Duke data set contained 25 noncharacterizable sets, comprised of 64 data elements. A noncharacterizable data set is a collection of two or more data elements that contain the same input values with two or more unique output values (see example in Table 10.22).

Note that the seven input values in Table 10.22 are identical, while the output value varies. KE attempts to eliminate noncharacterizable data sets, as these contribute systemic errors that retard the learning process. Depending on the envi-

<table>
<thead>
<tr>
<th>Variable</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>Calcification distribution and description</td>
<td>Mammographic calcification findings</td>
</tr>
<tr>
<td>Mass margin and shape</td>
<td>Mammographic mass findings</td>
</tr>
<tr>
<td>Associated findings, special cases</td>
<td>Mammographic miscellaneous findings</td>
</tr>
<tr>
<td>Age</td>
<td>Clinical history variables</td>
</tr>
</tbody>
</table>
Table 10.22 Example of a noncharacterizable set.

<table>
<thead>
<tr>
<th>Calcification distribution</th>
<th>Calcification description</th>
<th>Mass margin</th>
<th>Mass shape</th>
<th>Associated findings</th>
<th>Special findings</th>
<th>Age</th>
<th>Classification</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>13</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>56</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>13</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>56</td>
<td>1</td>
</tr>
</tbody>
</table>

Environment, the handling of these sets varies from (1) removing all data elements in a noncharacterizable set, (2) removing some data elements to generate a characterizable set, (3) altering the output value to represent a noncertainty value rather than a certainty value [e.g., giving all elements in the set a 0.5 (unknown) rather than a 0 or 1], or (4) generating another output value to use in training that would represent uncertainty.

It was concluded that the environment was not consistently defined by this set of discriminators, which is not unusual for intelligent systems, especially in biological domains. This conclusion was based on the finding that reducing the number of inputs from 16 to 7 generated a subset of noncharacterizable data sets (which were removed) that manifest systemic errors, but there was no significant increase in systemic errors. Given the options available, it was decided that the best course of action would be to eliminate the noncharacterizable sets, as stated previously. The Duke data set, therefore, was trimmed from 500 cases to 424 cases.

A similar process was performed on the USF DDSM data set, which only contains the seven inputs. Of the 1978 cases, there were 180 noncharacterizable sets consisting of 815 data elements, which was well within expectations. Since the environment is not fully defined by the input discriminators, one can expect a nonlinear increase in cases that belong to a noncharacterizable set. The USF data set, therefore, was trimmed from 1978 cases to 1163 cases.

10.8.2.2 Agents and training methods

The agents incorporated into the AI system were the PNN and several SVM kernels. A summary of the theoretical background of these agents is presented in Sections 10.5.1 (SVM) and 10.6.1.1 (PNN). Numerical methods were used to train the different agents. The MOP used to evaluate accuracy of the AI agents was the well-known and previously discussed ROC analysis and will not be discussed further here.

10.8.3 Results

CAD results using: (1) a multivariate PNN, and (2) two SVM kernels are depicted in Table 10.23, before and after the KE/KR activity. The approximate average $A_z$ performance improvement was 10%, and the average partial $A_z$ performance improvement was 50%.
Table 10.23 \( A_z \) and partial \( A_z \) results with the KE/KR approach.

<table>
<thead>
<tr>
<th></th>
<th>Before</th>
<th>After</th>
<th>% Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>PNN</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( A_z )</td>
<td>0.805543</td>
<td>0.873832</td>
<td>8.48</td>
</tr>
<tr>
<td>( A_z 90 )</td>
<td>0.347916</td>
<td>0.511734</td>
<td>47.09</td>
</tr>
<tr>
<td>SVM-RBF kernel</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( A_z )</td>
<td>0.812</td>
<td>0.895362</td>
<td>10.27</td>
</tr>
<tr>
<td>( A_z 90 )</td>
<td>0.345</td>
<td>0.489207</td>
<td>41.80</td>
</tr>
<tr>
<td>SVM-S2000 kernel</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( A_z )</td>
<td>0.817</td>
<td>0.9101842</td>
<td>11.41</td>
</tr>
<tr>
<td>( A_z 90 )</td>
<td>0.351</td>
<td>0.5596394</td>
<td>59.44</td>
</tr>
</tbody>
</table>

10.8.4 DISCUSSION

It was determined that by using modern data KE methods, SVM performance can be improved, as measured by the MOP of specificity, sensitivity, \( A_z \), partial \( A_z \), and PPV. The increase in performance using \( A_z \) as the main test parameter showed
a 9.18% improvement over the original raw data set. In addition, specificity at 98% sensitivity increased by 55%. By using the SVM or K-PLS, fast and accurate results can be achieved. This is crucial for proper diagnosing of breast cancer, particularly if the PLS and/or K-PLS algorithm has to be trained or retrained in real time.

ACKNOWLEDGMENTS

This work was supported in part by the New York State Breast Cancer Research and Education Fund through Department of Health Contract C017939.

REFERENCES


39. R. Rosipal and L. Trejo, “Kernel partial least squares regression in reproducing
40. K. Bennett and M. Embrechts, “An optimization perspective on kernel partial
least squares regression.” *Advances in Learning Theor: Methods, Models and
41. M. Embrechts and K. Bennett, “Sparse-kernel PLS for molecular drug design.”
42. F. Lindgren, P. Geladi, and S. Wold, “The kernel algorithm for PLS.”
43. S. Ren and L. Gao, “Simultaneous spectrophotometric determination of copper
44. L. Gao and S. Ren, “Simultaneous spectrometric determination of manganese,
zinc and cobalt by kernel partial least-squares method.” *J. Autom. Chem.* 20(6),
45. L. Gao and S. Ren, “Simultaneous spectrophotometric determination of four
46. N. Cristianini and J. Shawe-Taylor, *Support Vector Machines and Other
Kernel-Based Learning Methods*, Cambridge University Press, Cambridge,
UK (2000).
47. B. Schölkopf and A. Smola, *Learning with Kernels*, MIT Press, Cambridge,
MA (2002).
48. J. Bi and K. Bennett, “Regression error characteristic curves.” *Proceedings of
49. V. Froelicher, K. Shetler, and E. Ashley, “Better decisions through science:
53. J. Elmore, M. Barton, V. Moceri, S. Polk, P. Arena, and S. Fletcher, “Ten-
year risk of false positive screening mammograms and clinical breast exami-