The role of feature selection in building pattern recognizers for computer-aided diagnosis

Clay Spence and Paul Sajda

National Information Display Laboratory
Sarnoff Corporation
Princeton, NJ 08543-5300, USA

ABSTRACT

In this paper we explore the use of feature selection techniques to improve the generalization performance of pattern recognizers for computer-aided diagnosis (CAD). We apply a modified version of the sequential forward floating selection (*SFFS*) of Pudil et al.[1] to the problem of selecting an optimal feature subset for mass detection in digitized mammograms. The complete feature set consists of multi-scale tangential and radial gradients in the mammogram region of interest. We train a simple multi-layer perceptron (MLP) using the SFFS algorithm and compare its performance, using a jackknife procedure, to an MLP trained on the complete feature set (35 features). Results indicate that a variable number of features is chosen in each of the jackknife sets (12 ± 4) and the test performance, A_z, using the chosen feature subset is no better than the performance using the entire feature set. These results may be attributed to the fact that the feature set is noisy and the data set used for training/testing is small. We next modify the feature selection technique by using the results of the jackknife to compute the frequency at which different features are selected. We construct a classifier by choosing the top N features, selected most frequently, which maximize performance on the training data. We find that by adding this "hand-tuning" component to the feature selection process, we can reduce the feature set from 35 to 8 features and at the same time have a statistically significant increase in generalization performance ($p < 0.015$).

Keywords: Feature Selection, Pattern Recognition, Computer-aided Diagnosis, Neural Network, Breast Cancer, Mammogram.

1. INTRODUCTION

Computer-aided diagnosis (CAD) systems typically can be broken down into three main processing stages. The first two stages are aimed at increasing the overall signal-to-noise levels in the image and applying simple rules/heuristics to define a set of candidate regions-of-interest (ROIs). These stages have adjustable parameters which typically are set to produce a very high sensitivity, usually at a cost of low specificity. The final stage is a statistical model, whose parameters are typically found using error-based optimization. The function of this last stage is to reduce false positives, i.e., increase specificity, without significant loss in sensitivity.

These statistical models or pattern recognizers, many times having a neural network architecture, are constructed by optimizing an error criterion given a set of training data. One important consideration when training pattern recognition systems is the need to identify a feature set which is best matched for discriminating between the objects of interest. Often a large set of features is initially defined with the hope being that the pattern recognizer can learn which features are important and which are not. However, recognizers trained with a large number of features are often plagued by the curse of dimensionality, eventually converging to a local minima solution which is suboptimal. To reduce the number of local minima and the contribution of clutter and/or noise in the input, techniques are needed to find a feature subset, i.e., a subspace with lower dimensionality, which maximizes the performance of the pattern recognizer.

In this paper we describe results of applying feature selection techniques to train pattern recognizers for detecting masses in mammogram regions-of-interest (*ROI*s). The feature selection technique we use is a modification of the sequential forward floating selection (*SFFS*) algorithm proposed by Pudil et al[1] to search a feature space for an optimal feature subset. We begin the paper by presenting the details of the SFFS search algorithm, as well as elaborating on the architecture of the pattern recognizer and the class of features to which it will be applied. We

Email: {cspence|psajda}@sarnoff.com

then present results of directly applying the SFFS feature selection algorithms for mammographic mass detection, discussing how noise and the size of the dataset effects these results. We then modify the technique and use the frequency of selected features to select the optimal feature subset. For cases of small and noisy datasets, this "hand-tuning', based on the frequency of feature selection, can be useful for reducing the number of required features and improving generalization performance.

2. FEATURE SELECTION ALGORITHM

Feature-selection algorithms usually involve maximizing (or minimizing) a criterion function whose value can be computed for any subset of the complete set of features. For example, one such criterion function is the log-likelihood. A feature selection algorithm might therefore search for the subset of features which maximizes the log-likelihood. One approach to finding the optimal feature subset is exhaustive search, but this almost always involves examining an enormous number of feature subsets, $n!$ for a set of size n, and thus is too slow. Another choice is the *branch and bound* search algorithm[2] which is optimal under certain conditions on the criterion function and is much faster than exhaustive search. However, it is still quite slow for moderately large feature sets.

Sequential forward selection (SFS)[3] and sequential backward selection (SBS)[4] are two very fast but sub-optimal algorithms. SFS begins with no features in the subset and proceeds to sequentially construct subsets by successively adding an additional feature. The subset S_k of size k is constructed by adding to the subset S_{k-1} of size $k-1$ that single feature that gives the best performance for the new subset. In this way, only $n(n+1)/2$ subsets must be examined. SBS works in an analogous manner, but starts with the entire set of features and successively removes features in searching for the optimal subset. These algorithms are sub-optimal because they construct nested subsets, i.e., they will choose subsets such that $S_{k-1} \subset S_k, \forall k$. This property does not necessarily hold for the optimal subsets. Unfortunately the departure from optimality can be large.

To overcome some of the sub-optimal behavior of the SFS and SBS algorithms, Pudil et al[1] proposed modifications of the SFS and SBS algorithms they called *floating* sequential algorithms. The forward and backwards variants thus have the acronyms *SFFS* and *SBFS*, respectively. The SFFS algorithm is similar to the SFS algorithm, except that, when a new subset S_k is constructed, the algorithm enters a loop in which it constructs smaller subsets from S_k by sequentially removing one feature at a time. This loop is terminated when removing a feature from subset S_l results in a subset S'_{l-1} that is not better than the previous best subset S_{l-1}. (Note that l can equal k.) It then resumes constructing larger subsets, but from S_l. In pseudo-code, the SFFS algorithm is (defining the empty set to be worse than any non-empty set of features):

SFFS Algorithm:
$k \leftarrow 0$
$S_0 \leftarrow \emptyset$
Loop — SFS
    Find feature $f \notin S_k$ that makes $S_{k+1} = S_k \cup f$ optimal.
    $k \leftarrow k + 1$
    Loop — backtrack
        Find feature $f \in S_k$ that makes $S'_{k-1} = S_k \backslash f$ optimal.
        If S'_{k-1} is better than S_{k-1}
            $S_{k-1} \leftarrow S'_{k-1}$
            $k \leftarrow k - 1$
        else
            Leave backtrack loop.
    Endloop — backtrack
While $k < n$

SBFS is similar. Note that the algorithm gives an approximately optimal subset of each size $k \in \{1, \ldots, n\}$. In general, the performance does not need to improve with the size of the subset, in fact subsets of different sizes are never compared with each other. In some experiments, these algorithms gave near-optimal results and were very fast.[1,5]

There can arise situations in which SFFS chooses an inferior feature subset, even among those for which it computed the criterion function. The backtracking procedure can cause it to "forget" a superior subset and not be

able to recover it. Consider, for example, the following sequence of steps in a "fictitious" run of the SFFS algorithm:

Step	Action	k	S_k	Criterion Function
0:	Add feature 0	1	$\{0\}$	0
1:	Add feature 1	2	$\{0,1\}$	1
2:	Add feature 2	3	$\{0,1,2\}$	3
3:	Add feature 3	4	$\{0,1,2,3\}$	7
4:	Remove feature 0	3	$\{1,2,3\}$	4
5:	Remove feature 1	2	$\{2,3\}$	2
6:	Add feature 4	3	$\{2,3,4\}$	5
7:	Add feature 5	4	$\{2,3,4,5\}$	6

At step 6 it has become impossible to recover the subset chosen at step 3, unless it is recovered during backtracking from a yet larger feature set. As a result the set S_4 obtained at the end of step 7 is worse than the S_4 that was already checked at step 3. Notice that this can only occur when the algorithm backtracks by two or more features.

A simple solution to this problem is the following; If some subsets of size k have had their criterion function computed, keep the best of these. At any step we keep the best of those subsets of the current size that have been checked. Thus the algorithm becomes:

SFFS Algorithm (Modified):
$k \leftarrow 0$
$S_0 \leftarrow \emptyset$
Loop — SFS
    Find feature $f \notin S_k$ that makes $S'_{k+1} = S_k \cup f$ optimal.
    If there is no previous S_{k+1} or S'_{k+1} is better than S_{k+1}
        $S_{k+1} \leftarrow S'_{k+1}$
    $k \leftarrow k + 1$
    Loop — backtrack
        Find feature $f \in S_k$ that makes $S'_{k-1} = S_k \backslash f$ optimal.
        If S'_{k-1} is better than S_{k-1}
            $S_{k-1} \leftarrow S'_{k-1}$
            $k \leftarrow k - 1$
        else
            Leave backtrack loop.
    Endloop — backtrack
While $k < n$

We implemented the algorithm in the scripting language *Python*, which simplifies the book-keeping. In practice, the modified algorithm tended to be faster since there was a reduction in the amount of backtracking needed to finding a better subset.

3. PATTERN RECOGNIZERS

The pattern recognizer we used to evaluate the modified SFFS algorithm was a simple multilayer perceptron (one hidden layer containing two sum-and-sigmoid neurons). The pattern recognizers were trained with a quasi-Newton algorithm, implemented in a sequential quadratic programming routine from the commercial NAG subroutine library. This is a batch algorithm, i.e., the quantity being minimized is the error summed over all examples in the training set.

4. FEATURES

The complete feature set included 35 features extracted from the mammogram regions of interest (*ROIs*) (235 ROIs; 99 True positives, 106 False positives. Half this data was used for training and the other half for testing.*)

*Dataset provided courtsey of R2 Technology Inc.

Filter	Tap number 0	1	2	3
Gaussian (even)	1.0	0.60653	0.13534	0.011109
Grad-Gauss (odd)	0.0	0.60653	0.27068	0.099981

Table 1. Filter taps for gradient features. The other half of the taps are given by the indicated symmetry.

These features are multi-scale and represent tangential and radial gradients in the image. The gradient filters are seven-by-seven tap approximations to a gradient-of-Gaussian filter,

$$G(\vec{x}) = (\hat{n} \cdot \nabla) e^{-\|\vec{x}\|^2/(2\sigma^2)} \propto (\hat{n} \cdot \vec{x}) e^{-\|\vec{x}\|^2/(2\sigma^2)} \tag{1}$$

where $\vec{x}$ is the displacement from the filter center, and $\hat{n}$ is a unit vector in the direction along which we compute the gradient. For any $\hat{n}$, the gradient filter is just a linear combination of the gradients in the horizontal and vertical directions, with coefficients given by the sine and cosine of the direction angle, i.e., the components of $\hat{n}$. The horizontal and vertical gradients are separable, so that we can compute them by first filtering with a one-dimensional filter in the horizontal direction, and then filtering with a one-dimensional filter in the vertical direction. It therefore has relatively low computational cost. The filter in one of the directions is a Gaussian while the other is the derivative of a Gaussian. The actual scaling of the two is not relevant since we will scale the features before using them. The tap weights that were actually used are shown in Table 1.

For each image we computed the gradients in the radial and tangential directions by first computing the gradients in the horizontal and vertical directions, and then computing their linear combination at each pixel, with weights given by the sine and cosine of the direction angle from the tentative mass center to the pixel. These gradients were computed at several levels in the Gaussian pyramid of the ROI image. The resulting radial gradient is the inward-pointing gradient.

In practice, the gradients in a fixed direction at different levels can be adequately computed by simply blurring and sub-sampling the zeroth-level gradient image, since the blurring operation is a convolution with a Gaussian filter, and the convolution of the gradient image is equivalent to convolving the original image with a gradient-of-Gaussian filter in which the Gaussian is broader. When directly computing the gradient from an image, rather than by blurring a higher-resolution gradient image, the pixel at the mass center was assigned the value zero, since there is no radial or tangential direction there.

The features for the pattern recognizers were the pixel values of the feature images at the tentative mass centers. To have non-zero values for these features, we only used the feature images that were pyramid reductions of higher-resolution gradient or squared-gradient images.

To motivate this choice, consider the radial gradient images. A mass will have a more-or-less bright blob, and so the inward radial gradient will be positive at the edge of the mass. The center pixel of a reduced radial gradient image is essentially the radial gradient averaged over a region of some size given by the width of the Gaussian blurring filter. The different pyramid levels give averaging regions with different sizes. Because of the edge of the mass, the pixel at the center of the mass in each blurred radial gradient image will be relatively large if the Gaussian's width (the extent of blurring) matches the mass size. The pixel value will be relatively smaller for other Gaussian widths.

The average of the tangential gradient at the central pixel is always zero, since it is the sum of changes in brightness as one moves around a circle, summed with some weights over circles of different radii. The starting value is the ending value, so the summed change must be zero. Therefore we only used the square of the tangential gradient. This was computed at each of several levels in the pyramid. As noted earlier, the central pixel is always zero, so we constructed Gaussian pyramids of each squared tangential gradient image, and used all but the highest resolution level in each of these feature pyramids. Together these levels give the tangential energy at several scales or frequencies, averaged over regions with different sizes.

To summarize, we used the following features:

- The radial gradient computed at the highest resolution and blurred using pyramid reduction to levels 1–7. The central pixel of these lower-resolution images were used to give 7 features.

		Feature subset		All features	
Set	k_{Best}	A_z^{Train}	A_z^{Test}	A_z^{Train}	A_z^{Test}
0	5	0.83	0.64	0.80	0.72
1	15	0.95	0.67	0.94	0.71
2	16	0.92	0.63	0.91	0.69
3	9	0.88	0.63	0.87	0.73
4	8	0.86	0.73	0.89	0.73
5	17	0.92	0.77	0.79	0.75
6	11	0.93	0.68	0.90	0.64
7	17	0.93	0.73	0.86	0.72
8	15	0.89	0.74	0.94	0.77
9	10	0.86	0.73	0.88	0.70
Ave	12 ± 4	0.89 ± 0.04	0.70 ± 0.05	0.88 ± 0.05	0.72 ± 0.04

Table 2. Performance comparison between pattern recognizers trained with all features versus those trained on "optimal" subsets chosen by the feature selection process. Performance results are A_z values for both training and test data. k_{Best} is the number of features in the best subset.

- The squared tangential gradient was computed at each of levels 0–6. A Gaussian pyramid was constructed from each of these to a resolution equal to level 7 in the pyramid of the original image. In each pyramid the highest-resolution feature was not used. The central pixel of each of the other images was used as a feature, giving $7 - l$ features from levels $l + 1$ through 7 of the level-l squared tangential gradient. The total number of features for $0 \leq l \leq 6$ is 28.

Thus there are a total of 35 features.

5. EXPERIMENTS

Using the log-likelihood as the criterion function for the feature selection algorithm, we computed a ten-fold cross-validation estimate of the likelihood, i.e., the training set T was divided into ten nearly equal disjoint subsets T_i, and the network was trained ten times, once on each of the sets $T \backslash T_i$. Each of these training runs began from a common random initial weight vector. The summed cross-entropy error[†] on set T_i was then computed for the network trained on set $T \backslash T_i$, and this error was then summed over i. This ten-fold cross-validation procedure was repeated with five different initial random weight vectors. Weight-decay regularization was used to try to avoid over-fitting, i.e., a penalty term of the form $\frac{\lambda}{2}\|\mathbf{w}\|^2$ was added to the cross-entropy error to give the objective function being minimized during training. λ was adjusted to give the smallest cross-validation error. Thus, when combined with the feature-selection algorithm, both the regularization constant λ and the feature subset were being chosen to minimize the cross-validation error estimate.

The procedure described above was repeated as a "jackknife" procedure on ten different random splits of the total data set (i.e. split into training and test sets). To assess the quality of the results we computed the area A_z under the receiver operating curve (ROC).

6. RESULTS

6.1. Performance on selected feature subsets

The values of A_z on both the training and test sets for each jackknife split are shown in Table 2. Several aspects of these results are noteworthy;

1. Although one would expect over-fitting to be a problem with more features and hence more parameters, the use of more features did not result in better performance on the training set.

[†] The cross-entropy error is the negative log-likelihood for binary classification problems in which the network is being used as a model of the probability of class membership, i.e., in which the network output is considered to be the parameter of a Bernoulli distribution.[6]

Set	Min. occurrence frequency	No. features	A_z^{Train}	A_z^{Test}
A	9	1	0.56 ± 0.05	0.47 ± 0.03
B	7	3	0.74 ± 0.07	0.61 ± 0.07
C	6	5	0.79 ± 0.07	0.66 ± 0.07
D	5	8	0.86 ± 0.04	0.75 ± 0.05
E	4	15	0.85 ± 0.06	0.73 ± 0.04

Table 3. ROC performance figures of pattern recognizers constructed using the most frequently selected features. The frequencies were computed from the jackknife experiment, having 10 random splits into training and test sets.

Set	Feature subset "D" A_z^{Test}	All features A_z^{Test}	Difference
0	0.79	0.72	0.07
1	0.72	0.71	0.01
2	0.64	0.69	-0.05
3	0.80	0.73	0.07
4	0.80	0.73	0.07
5	0.77	0.75	0.02
6	0.73	0.64	0.09
7	0.73	0.72	0.01
8	0.77	0.77	0.00
9	0.77	0.70	0.07

Table 4. ROC performance figures (across the 10 subsets of the jackknife) of the pattern recognizer trained with feature subset "D" (8 features) versus all 35 features. The performance (A_z^{Test}) using subset "D" is better then when all 35 features are used for training (p=0.015).

2. The different feature subsets chosen across the jackknife did not improve performance over a pattern recognizer trained with all the features. In fact, the average A_z on the test data for the pattern recognizers trained on selected feature subsets was worse than for the pattern recognizers trained on all the data (though the difference was not statistically significant).

3. There is little consistency in the number of features chosen from one jackknife split to another.

6.2. Frequency of feature selection

We used the results of the jackknife experiments to compute the frequency with which each feature was included in a best feature subset. Although the size of the selected subset varied, some features were chosen more consistently than others. Intuitively, it would seem that the more often a feature is chosen, the more relevant information it carries. A feature subset made up of the more frequently-chosen features should therefore be expected to have good generalization performance. Table 3 contains the results of a jackknife study on the smallest five such feature subsets. The best subset (set "D") according to the jackknife study contained all of those features that were selected in half or more of the jackknife splits of the data, and actually does give better performance, on average, than using all features. It also contains only about one quarter of the total set of features.

We tested to see if feature set "D" consistently had a better performance (larger A_z), compared to the recognizer trained on all 35 features, across the 10 subsets of the jackknife. Since the same subsets were used for both "D" and the 35 feature case, we computed the difference in A_z for each jackknife subset. Table 4 shows the A_z values and differences. The difference between A_z on set "D" and A_z for all 35 features is positive for nine of the ten jackknife sets. The t-test on the differences indicates a positive mean ($p = 0.015$). Thus the smaller feature set "D" appears to have a statistically significant performance advantage over all 35 features, albeit it is a small one.

7. DISCUSSION

Several factors might account for why direct application of feature selection did not dramatically improve detector performance.

- It may be that the relevant information is well-distributed across many features in this problem, making feature selection more difficult.
- Although we have some confidence that cross-validation gives reasonable estimates of generalization performance, it is a noisy estimate, i.e., its variance can be high. Thus in choosing a feature subset the "noise" in the features can force the algorithm into choosing a sub-optimal set. This may have more serious consequences for sequential feature selection algorithms, since a poor choice at one step affects subsequent choices.
- Even if the cross-validation estimate is a "reasonable" error estimate, the location of its optimum in some parameter space, e.g., the set of feature subsets, may not be a reasonable estimate of the location of the true optimum.
- For each feature subset, we chose a regularization constant λ to optimize the cross-validation error estimate. In this study, the best value for λ was much larger when the input was the full feature set than when it was the optimal subset. It may be that this caused the network to average over more features, so that the effect of the noise in any one feature was minimized. Our results certainly support the notion that regularization has significant benefit, even if there is still some over-fitting.

All of these problems are affected by the size of the data set, which is relatively small in our case (235 examples, only 118 used for training). For very large data sets, the network training algorithm should be able to learn the relevance of individual features, so feature selection would be unnecessary. Presumably,for intermediate sized data sets, straightforward feature selection would be beneficial. We intend to study this when we acquire more data.

The second part of our study suggests a reasonable approach for applying feature selection for small and noisy datasets — find those features that occur most frequently in the feature subsets selected in a jackknife study, and then construct a new subset from the top N that are most frequently chosen. This bit of "hand-tuning" did result in smaller feature subsets which also had small improvements in generalization performance. In addition, one would tend to have more confidence in pattern recognizers constructed with a smaller feature subset, as long as some procedure such as the jackknife study indicated that the generalization performance does not decrease. However, further work is needed to determine if the generalization performance can be *dramatically* improved using these feature selection methods.

8. CONCLUSIONS

We have studied the benefits and drawbacks of feature selection techniques for improving the detection of masses in mammographic ROIs. We have concluded that, for small and noisy data sets, care is needed since, in some cases, the feature selection process may decrease generalization performance.

An alternative method of using feature selection, useful when the data is limited and/or noisy, is to perform a jackknife study to compute the frequency at which certain features are selected. Features selected more frequently presumably carry the majority of the information that is relevant for the classification problem. One can then form a subset from these features by hand. In applying this type of feature selection, we observed a small but significant benefit in terms of improved generalization performance. Intuitively, we expect better generalization performance from the pattern recognizer trained using the smaller feature subset.

9. ACKNOWLEDGMENTS

This work was supported by the National Information Display Laboratory. We would like to thank R2 Technology Inc. for supplying us with the mammographic ROI data.

REFERENCES

1. P. Pudil, J. Novovičova, and J. Kittler, "Floating search methods in feature selection," *Pattern Recognition Letters* **15**, pp. 1119–1125, Nov. 1994.
2. P. Narendra and K. Fukunaga, "A branch and bound algorithm for feature subset selection," *IEEE Trans. Comput.* **26**, pp. 917–922, 1977.
3. A. Whitney, "A direct method of nonparametric measurement selection," *IEEE Trans. Comput.* **20**, pp. 1100–1103, 1971.
4. T. Marill and D.M.Green, "On the effectiveness of receptors in recognition system," *IEEE Trans. Inform. Theory* **9**, pp. 11–17, 1963.
5. A. Jain and D. Zongker, "Feature selection: Evaluation, application, and small sample performance," *IEEE Trans. PAMI* **19**, pp. 153–158, Feb. 1997.
6. C. M. Bishop, *Neural Networks for Pattern Recognition*, Oxford University Press, New York, 1995.