Salzburg Interest Group on Integrated Systems

siGis-015-1998 June 30, 1998

ERC - Evolutionary Resample and Combine for Adaptive Parallel Training Data Set Selection

14th International Conference on Pattern Recognition
August 16-20, 1998, Brisbane, Australia

Reinhold Huber
reinhold.huber@dlr.de

Helmut A. Mayer
helmut@cosy.sbg.ac.at

Aero–Sensing Radarsysteme GmbH
c/o DLR Oberpfaffenhofen, Germany

Department of Computer Science
University of Salzburg, Austria

siGis
Salzburg Interest Group on Integrated Systems

Correspondence to:

Universität Salzburg
Institut für Computerwissenschaften und Systemanalyse
Jakob–Haringer–Straße 2
A–5020 Salzburg
Austria

ERC - Evolutionary Resample and Combine for Adaptive Parallel Training Data Set Selection

Reinhold Huber
Aero–Sensing Radarsysteme GmbH
c/o DLR Oberpfaffenhofen
82234 Weßling, Germany
reinhold.huber@dlr.de

Helmut A. Mayer
Department of Computer Science
University of Salzburg
5020 Salzburg, Austria
helmut@cosy.sbg.ac.at

Abstract

We introduce evolutionary resampling and combine (erc) – a genetic algorithm based selection scheme for training examples for a multilayer perceptron (MLP) classifier. The erc method is compared to various adaptive resample and combine techniques arc-fs, arc-lh and arc-x4. To diminish the dependencies on the size of the training data set (TDS) and the missing consideration of test set performance common to all arc methods we present erc being based on evaluation of performance on a validation data set (VDS). Combination of classifiers is performed by all arc methods so as to reduce classifiers' variance, thus, erc also utilizes classifier combination schemes. All algorithms are compared for a real–world problem, the classification of high resolution interferometric synthetic aperture radar (InSAR) data into several land–cover classes.

1. Introduction

Feature extraction and classification of SAR images is commonly based on combined point and texture classification [14]. This study uses a high resolution InSAR data set consisting of SAR backscatter image data and InSAR phases correlation (coherence image). Texture extraction is applied to both images prior to classification by a one–hidden–layer MLP. In order to identify feature subsets of most discriminating power, feature selection based on *Jeffreys–Matusita–Distance* (JMD) is applied.

MLPs are regarded as classifiers of low bias and high variance, thus, to decrease variance adaptive resampling and combine methods have been suggested [3]. As these schemes are inherently sequential and do not take generalizing ability of the classifier into account, we suggest parallel evolutionary resample and combine to optimize classifier performance on an independent VDS.

2. Feature extraction from InSAR data

Extraction of textual features from SAR backscatter and InSAR coherence data is based on local statistics of first and second order. Specifically, we employ the coefficient of variation c_v for texture description based on first order statistics. The c_v is simply estimated by $\hat{c}_v = s/\bar{x}$, where s is the standard deviation and $\bar{x}$ is the mean in an estimation window, in our case a 7×7 pixels square neighbourhood.

Among the numerous existing texture description methods based on second order statistics [15] we have chosen the method of *Eigenfilters* [1]. With this method, a neighbourhood vector $\vec{i}$ containing adjacent pixels for the pixel in question is constructed. An unbiased estimator $\hat{\mathbf{C}}$ of the variance–covariance matrix is obtained by

$$\hat{\mathbf{C}} = \frac{1}{N-1}\sum_N (\vec{i}-\vec{m})(\vec{i}-\vec{m})^T, \quad \text{with} \quad \vec{m} = \frac{1}{N}\sum_N \vec{i},$$

where N is the number of image pixels.

To obtain the principal components the eigenvalue problem $(\hat{\mathbf{C}} - \lambda\mathbf{I})\vec{e} = 0$, where $\mathbf{I}$ and $\vec{e}$ denote identity matrix and eigenvector is solved. To build the eigenimages the initial image is convolved with the filter masks provided by the eigenvectors ordered by decreasing eigenvalue. A quantitative description of the texture is provided by the texture energy measure E [8]. E is given by the summed absolute difference between individual eigenimage pixels x and its mean $\bar{x}$ in a square window $E = \sum |x - \bar{x}|$.

3. Feature selection

The primary features SAR backscatter and coherence together with the derived features from local statistics and principal components make up a potentially high–dimensional feature vector. The *"Curse of Dimensionality"* [2] and empirical results exploring feature subset selection techniques [13] suggest the construction of a feature vector of lower dimension. To identify the most relevant features among the large number of texture energy features derived from principal components images the average Jeffreys–Matusita–distance (JMD) is evaluated for all possible subsets of an initial subset containing the texture energy of first six principal components. These are the first six principal features contained 95.48 percent variation for the SAR image and 99.17 percent variation for the coherence image.

For multivariate Gaussian distributions JMD which measures the separability between two class conditional distributions, characterised by their mean vectors $\vec{m}_i, \vec{m}_k$ and covariance matrices Σ_i, Σ_k, is given by $J_{ik} = 2(1 - e^{-B_{ik}})$ [6] in which B_{ik} (Battacharyya–distance) is

$$B_{ik} = \frac{1}{8}(\vec{m}_i - \vec{m}_k)^t \Sigma^{-1} (\vec{m}_i - \vec{m}_k) + \frac{1}{2} \ln \left[\frac{|\Sigma|}{|\Sigma_i|^{\frac{1}{2}} |\Sigma_k|^{\frac{1}{2}}} \right]$$

$$\text{and} \qquad \Sigma = \frac{\sum_i + \sum_k}{2}.$$

Each subset is characterised by its average JMD. Only one texture energy feature was selected from SAR and coherence image, respectively, by ranking the subsets by their average JMD [4]. The features of largest frequency in the best subsets were retained, namely texture energy from the fourth principal component of SAR E_{S4} and the third principal component of coherence E_{C3}.

The result of feature extraction and selection is the 5–dimensional feature vector $\vec{x}$ containing the features SAR backscatter, InSAR coherence and the textural features $\hat{c}_v$, E_{S4}, and E_{C3}. Initial TDSs of different size with the four different land–cover classes, water, forest, built–up, and open area, were generated randomly for a study area of 768×768 pixels of airborne AeS-1 SAR data. The size of the training sets are 200, 2000 and 10000 patterns, each class is represented by an equal number of examples. Additionally, test and validation sets containing 2000 examples each and equal class size have been generated by random sampling. Figure 1 shows the five selected feature images.

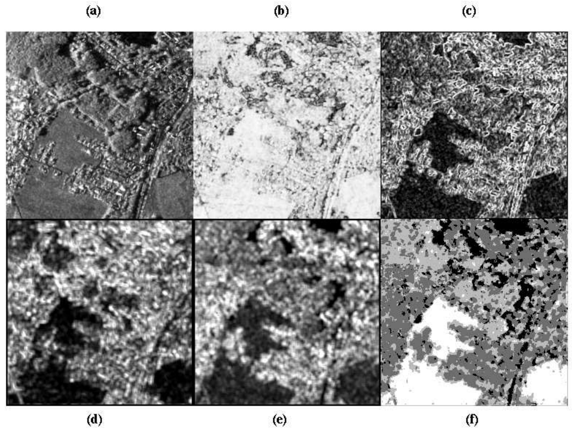

Figure 1. Feature images and multilayer perceptron classification: (a) SAR image, (b) InSAR coherence, (c) coefficient of variation, texture energy of (d) fourth principal component of SAR image and (e) third principal component of coherence, (f) classification.

4. Arcing classifiers

Adaptive resample and combine (arc) is a technique to increase classifier performance. The average error Error$(C, N, \vec{x})$ of a classifier C for sample size N at point $\vec{x}x$ is characterised by a decomposition into bias Bias$(C, N, \vec{x})$ and variance $Var(C, N, \vec{x})$ term [7]:

$$\text{Error}(C, N, \vec{x}) = \text{Bias}(C, N, \vec{x})^2 + \text{Var}(C, N, \vec{x}).$$

Arcing tries to minimise Var$(C, N, \vec{x})$ by voting among classifiers.

All arcing algorithms use an initial TDS $T(\vec{x}, C)$ containing N examples which define an association between a feature vector $\vec{x}$ and M different classes C. The set T serves as the basis for estimating the class conditional densities $p(\vec{x}|C_m)$. Arcing starts by sampling with replacement from T with an initial probability of $P_1(n) = 1/N$. Arcing is iterated for K steps and sample probabilities are adapted due to specific criteria in order to focus the classifier on "hard to learn" examples. The selection probability of an example n in step k is denoted by $P_k(n)$. We shortly review the operation of the considered arcing methods (all methods start with $k = 1$):

arc-fs (called AdaBoost.M2 in [5]):

1. Sample with replacement from T with probabilities $P_k(n)$ and construct the classifier C_k using the resampled set T_k of size N.
2. Classify T_k using C_k and let $d(n) = 1$ if example n is incorrectly classified, else $d(n) = 0$.
3. Let $e_k = \sum P_k(n)d(n)$ and $\beta_k = (1 - e_k)/e_k$.
4. Update probabilities by $P_{k+1}(n) = P_k(n)\beta_k^{d(n)}$.
5. Normalize by $P_{k+1}(n) = P_{k+1}(n)/\sum_i P_{k+1}(i)$.
6. Let $k = k + 1$ and goto 1 if $k < K$.
7. Arc-fs combines $C_1..C_k$ by weighted voting assigning the vote $log(\beta_k)$ to classifier C_k.

arc-lh [9]:

1. Like arc-fs.
2. Let $e_k(n) = |\vec{t}(n) - \vec{o}(n)|^2$ where $\vec{t}(n)$ and $\vec{o}(n)$ are target and output of C_k.
3. Update probabilities $P_{k+1}(n) = P_k(n) + e_k(n)$.
4. Normalization $P_{k+1}(n) = P_{k+1}(n) / \sum_i P_{k+1}(i)$.
5. Let $k = k + 1$ and goto 1 if $k < K$.
6. Arc-lh combines $C_1..C_k$ by majority voting.

arc-x4 [3]:

1. Like arc-fs.
2. Classify T_k using C_k and let $m_k(n) = \sum_\kappa d_\kappa(n)$ being the number of misclassifications of example n using classifier C_κ for $\kappa \in 1..k$.
3. Update probabilities $P_{k+1}(n) = 1 + m_k(n)^4$.
4. Normalization $P_{k+1}(n) = P_{k+1}(n) / \sum_i P_{k+1}(i)$.
5. Let $k = k + 1$ and goto 1 if $k < K$.
6. Arc-x4 combines $C_1..C_k$ by majority voting.

5. Evolutionary resample and combine

Based on a method proposed in [11], we employed *Genetic Algorithms* (GA) for the evolutionary search of TDSs optimised towards the classification accuracy of the MLP on a validation set. The patterns constituting a complete TDS are selected in parallel by the GA, the MLP is trained with these individuals (TDSs), and TDS fitness is assigned according to the VDS performance. Hence, the quality of single training patterns are not explicitly evaluated, but rather their contribution and cooperation in a TDS for a well generalising MLP is awarded.

The current encoding of a TDS is simple and straightforward. Each training pattern in the initial TDS is represented by a single bit of the GA *Chromosome*. Thus, the number of bits (chromosome length l) is equal to the number of training patterns in the initial TDS. The actual TDS is built by selecting all the patterns whose corresponding bits are of value 1. Contrary to arcing, where the fixed size TDS contains a variety of identical patterns with high probability, ercing also enables the evolution of a TDS of variable size without any identical patterns ($0 - l$ patterns can be mapped to a TDS). The fitness of a TDS is given by $f = \frac{\sum_N 1-d(n)}{N}$ (overall accuracy). We will use the terms T–fitness and V-Fitness for the classification accuracy on TDSs and VDSs, respectively.

Following the work of [16], where evolved MLPs in the last generation are used for majority voting as opposed to the prevailing practice of taking the MLP with the best fitness, we investigated three ways to exploit the knowledge accumulated by evolution. For the classification of a test set *erc–sb* employs the single best MLP discovered, *erc–vb* the g (number of generations) best MLPs, and *erc–vl* all MLPs of the last generation. The MLPs are combined by majority voting.

For TDS evolution experiments presented below we used a generational GA with basic operators implemented in the parallel netGEN system [10] evolving MLP topologies and/or TDSs on interconnected workstations.

6. Results

With all experiments the architecture of the MLP has been fixed to a $5 - 10 - 4$ one–hidden–layer fully connected network, and all MLP trainings have been conducted using the RPROP algorithm [12]. Figure 2 gives the performance of the classifiers C_k for the three considered arcing methods. Figure 3 depicts evolution of T– and V-fitness for ercing. Table 1 and Table 2 summarise the voting performance on the test set for arcing and ercing, respectively.

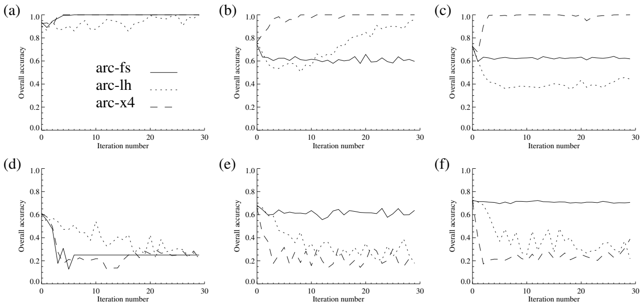

Figure 2. Overall accuracy of arc-fs, arc-lh and arc-x4 on TDSs and test sets versus iteration number: TDS (a) 200, (b) 2000, (c) 10000, test set (d) 200, (e) 2000, (f) 10000 examples

TDS	arc–fs	arc–lh	arc–x4
Size	Acc	Acc	Acc
200	0.3275	0.5250	0.6105
2000	0.6805	0.6710	0.6590
10000	0.7240	0.7205	0.7215

Table 1. Overall accuracy of the voting performance of arc-fs, arc-lh and arc-x4 on the test set for different TDS sizes.

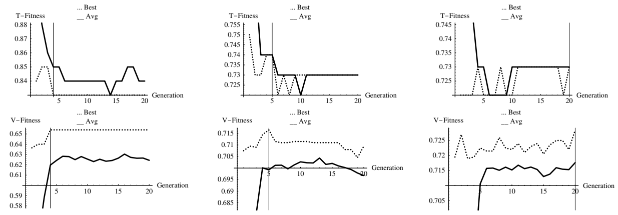

Figure 3. Best and average fitness on TDSs of different size (T–Fitness) and on the validation set (V–Fitness) per generation (initial TDS sizes 200, 2000, 10000 from left to right).

Initial TDS	erc–sb	erc–vb	erc–vl
Size	Acc	Acc	Acc
200	0.6510	0.6510	0.6585
2000	0.7105	0.7175	0.7080
10000	0.7200	0.7210	0.7150

Table 2. Overall accuracy of erc of the single best MLP (erc–sb), voting of best MLPs (erc–vb), and voting of the last generation (erc–vl) for different initial TDS sizes.

7. Discussion and conclusion

With the small initial TDS arcing proliferated low classification performance results, see figure 2 (b). The bad performance of arc–fs (see table 1) seems to stem from the weighted voting scheme, as the performance graphs during training of arc-fs and arc-lh (majority voting) are very similar. Based on the observation that arcing performance increases remarkably with initial TDS size as shown in figures 2 (e) and (f) and table 1, it becomes obvious that the heuristic probability update rules demanding a (re)normalisation of probabilities induce a dependency on initial TDS size. This effect is much smaller with ercing (see table 2), although, the search space increases exponentially with initial TDS size.

While arcing focuses only on “hard to learn” patterns (arc–lh even more on the difficult patterns generating highest error), ercing concentrates on the best ensemble of patterns. These also include difficult patterns, as T–fitness decreases with the number of generations depicted in figure 3. The arcing performance heavily relies on combining of and voting among classifiers, whereas ercing searches for single good classifiers. Combination of the evolutionary classifiers generates only slightly different results (sometimes worse than the single best classifier), but this is also due to the fact that the number of generations has been kept low. Generally, ercing is robust with respect to initial TDS size (compare table 2) and quickly discovers (often in the first few generations) TDSs being able to train a well performing MLP (see figure 3).

References

[1] F. Ade. Characterization of Textures by ’Eigenfilters’. *Signal Processing*, 5:451–457, 1983.

[2] C. M. Bishop. *Neural Networks for Pattern Recognition*. Clarendon Press, 1995.

[3] L. Breiman. Arcing Classifiers. Technical Report 460, Department of Statistics, University of California, Berkeley, CA, USA, Feb. 1996.

[4] L. V. Dutra and R. Huber. Feature Extraction and Selection for ERS–1/2 InSAR Classification. *submitted to International Journal of Remote Sensing*, 1998.

[5] Y. Freund and R. E. Schapire. Experiments with a New Boosting Algorithm. In *Proceedings of the International Conference on Machine Learning*, 1996.

[6] K. Fukunaga. *Introduction to Statistical Pattern Recognition*. Academic Press, 1990.

[7] S. Geman, E. Bienenstock, and R. Doursat. Neural Networks and the Bias/Variance Dilemma. *Neural Computation*, 4(1):1–58, 1992.

[8] K. I. Laws. Rapid Texture Identification. In *Proceedings of SPIE Image Processing for Missile Guidance*, volume 238, pages 376–380, 1980.

[9] F. Leisch and K. Hornik. ARC-LH: A New Adaptive Resampling Algorithm for Improving ANN Classifiers. In M. Mozer, M. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems*, volume 9. MIT Press, Cambridge, MA, USA, 1997.

[10] H. A. Mayer, R. Huber, and R. Schwaiger. Lean Artificial Neural Networks - Regularization Helps Evolution. In *Proceedings of the 2nd Nordic Workshop on Genetic Algorithms and their Applications*, pages 163–172, 1996.

[11] H. A. Mayer and R. Schwaiger. Towards the Evolution of Training Data Sets for Artificial Neural Networks. In *Proceedings of the 4th IEEE International Conference on Evolutionary Computation*, pages 663–666. IEEE Press, 1997.

[12] M. Riedmiller and H. Braun. A Direct Adaptive Method for Faster Backpropagation Learning: The RPROP Algorithm. In *Proceedings of International Conference on Neural Networks*, pages 586–591, San Francisco, CA, USA, 1993.

[13] W. Siedlecki and J. Sklansky. On Automatic Feature Selection. *International Journal of Pattern Recognition and Artificial Intelligence*, 2(2):197–220, 1988.

[14] A. H. S. Solberg and A. K. Jain. Texture Fusion and Feature Selection Applied to SAR Imagery. *IEEE Transactions on Geoscience and Remote Sensing*, 35(2):475–479, Mar. 1997.

[15] M. Tuceryan and A. K. Jain. Texture Analysis. In C. Chen, L. Pau, and P. Wang, editors, *Handbook of Pattern Recognition and Computer Vision*, pages 235–276. World Scientific, 1993.

[16] X. Yao and Y. Liu. Ensemble Structure of Evolutionary Artificial Neural Networks. In *Proceedings of the 3rd IEEE International Conference on Evolutionary Computation*, pages 659–664. IEEE Press, 1996.