1

Computational Intelligence Algorithms and DNA Microarrays

D.K. Tasoulis[1], V.P. Plagianakos[2], and M.N. Vrahatis[2]

[1] Institute for Mathematical Sciences, Imperial College London, South Kensington, London SW7 2PG, United Kingdom
d.tasoulis@imperial.ac.uk

[2] Computational Intelligence Laboratory, Department of Mathematics, University of Patras Artificial Intelligence Research Center (UPAIRC), University of Patras, GR–26110 Patras, Greece
{vpp,vrahatis}@math.upatras.gr

Summary. In this chapter, we present Computational Intelligence algorithms, such as Neural Network algorithms, Evolutionary Algorithms, and clustering algorithms and their application to DNA microarray experimental data analysis. Additionally, dimension reduction techniques are evaluated. Our aim is to study and compare various Computational Intelligence approaches and demonstrate their applicability as well as their weaknesses and shortcomings to efficient DNA microarray data analysis.

1.1 Introduction

The development of microarray technologies gives scientists the ability to examine, discover and monitor the mRNA transcript levels of thousands of genes in a single experiment. The development of technologies capable to simultaneously study the expression of every gene in an organism has provided a wealth of biological insight. Nevertheless, the tremendous amount of data that can be obtained from microarray studies presents a challenge for data analysis.

This challenge is twofold. Primarily, discovering patterns hidden in the gene expression microarray data across a number of samples that are correlated with a specific condition is a tremendous opportunity and challenge for functional genomics and proteomics [1–3]. Unfortunately, employing any kind of pattern recognition algorithm to such data is hindered by the *curse of dimensionality* (limited number of samples and very high feature dimensionality). This is the second challenge. Usually to address this, one has to preprocess the expression matrix using a dimension reduction technique [4] and/or to find a subset of the genes that correctly characterizes the samples. Note that this is not similar to *"bi-clustering"*, which refers to the identification of genes that exhibit similar behavior across a subset of samples [5, 6]. In this chapter we examine the application of various Computational Intelligence

D.K. Tasoulis et al.: *Computational Intelligence Algorithms and DNA Microarrays*, Studies in Computational Intelligence (SCI) **94**, 1–31 (2008)
www.springerlink.com

methodologies to face problems arising from the twofold nature of the microarray data. We also examine various manners to combine and interact algorithms towards a completely automated system.

To this end the rest of this chapter is structured as follows. Sections 1.2 and 1.3 are devoted to a brief presentation of Neural Networks as classification tools, Evolutionary Algorithms that can be used for dimension reduction, and their synergy. In Section 1.4 various feature selection and dimension reduction techniques are presented, starting from the Principal Component Analysis, continuing with several clustering algorithms, and finally we analyze hybrid approaches. In Section 1.5 using well known and publicly available DNA microarray problems, we study and examine feasible solutions to many implementation issues and report comparative results of the presented algorithms and techniques. The chapter ends with a brief discussion and some concluding remarks.

1.2 Neural Networks

Feedforward Neural Networks (FNNs) are parallel computational models comprised of densely interconnected, simple, adaptive processing units, characterized by an inherent propensity for storing experiential knowledge and rendering it available for use. FNNs have been successfully applied in numerous application areas, including DNA microarray data analysis [7].

To train an FNN, supervised training is probably the most frequently employed technique. The training process is an incremental adaptation of connection weights that propagate information between neurons. A finite set of arbitrarily ordered examples is presented at the input of the network and associated to appropriate references through an error correction process. This can be viewed as the minimization of an error measure, which is usually defined as the sum-of-squared-differences error function E over the entire training set:

$$w^* = \min_{w \in \mathbb{R}^n} E(w), \tag{1.1}$$

where $w^* = (w_1^*, w_2^*, \ldots, w_n^*) \in \mathbb{R}^n$ is a minimizer of E. The rapid computation of such a minimizer is a rather difficult task since, in general, the number of network weights is high and the corresponding nonconvex error function possesses multitudes of local minima and has broad flat regions adjoined with narrow steep ones.

Let us consider the family of gradient–based supervised learning algorithms having the iterative form:

$$w^{k+1} = w^k + \eta^k d^k, \qquad k = 0, 1, 2, \ldots \tag{1.2}$$

where w^k is the current weight vector, d^k is a search direction, and η^k is a *global* learning rate, i.e. the same learning rate is used to update all the weights of the network. Various choices of the direction d^k give rise to distinct algorithms. A broad class of methods uses the search direction $d^k = -\nabla E(w^k)$, where the gradient $\nabla E(w)$

can be obtained by means of back–propagation of the error through the layers of the network [8]. The most popular training algorithm of this class, named batch Back–Propagation (BP), minimizes the error function using the steepest descent method [9] with constant, heuristically chosen, learning rate η. In practice, a small value for the learning rate is chosen ($0 < \eta < 1$) in order to secure the convergence of the BP training algorithm and to avoid oscillations in a direction where the error function is steep. It is well known that this approach tends to be inefficient. This happens, for example, when the search space contains long ravines that are characterized by sharp curvature across them and a gently slopping floor.

Next, we give an overview of two neural network training algorithms: the Rprop algorithm and the adaptive online algorithm. Both algorithms have been used on DNA microarray problems. Rprop is one of the fastest and most effective training algorithms. On the other hand, adaptive online seems more suitable for this kind of problems, due to its ability to train FNNs using extremely large training sets.

1.2.1 The Rprop Neural Network Training Algorithm

The Resilient backpropagation (Rprop) [10] algorithm is a local adaptive learning scheme performing supervised training of FNNs. To update each weight of the network, Rprop exploits information concerning the sign of the partial derivative of the error function. The size of the weight change, Δw_{ij}, is determined by a weight specific update value, $\Delta_{ij}^{(t)}$, given by the following formula:

$$\Delta w_{ij}^{(t)} = \begin{cases} -\Delta_{ij}^{(t)}, & \text{if } \frac{\partial E^{(t)}}{\partial w_{ij}} > 0, \\ +\Delta_{ij}^{(t)}, & \text{if } \frac{\partial E^{(t)}}{\partial w_{ij}} < 0, \\ 0, & \text{otherwise,} \end{cases}$$

where $\partial E^{(t)}/\partial w_{ij}$ denotes the summed gradient information over all patterns of the training set (batch training). The second step of the Rprop algorithm is to determine the new update values, using the following formula:

$$\Delta_{ij}^{(t)} = \begin{cases} \eta^{+}\Delta_{ij}^{(t-1)}, & \text{if } \frac{\partial E^{(t-1)}}{\partial w_{ij}} \frac{\partial E^{(t)}}{\partial w_{ij}} > 0, \\ \eta^{-}\Delta_{ij}^{(t-1)}, & \text{if } \frac{\partial E^{(t-1)}}{\partial w_{ij}} \frac{\partial E^{(t)}}{\partial w_{ij}} < 0, \\ \Delta_{ij}^{(t-1)}, & \text{otherwise,} \end{cases}$$

where $0 < \eta^{-} < 1 < \eta^{+}$, i.e. each time the partial derivative with respect to w_{ij} changes its sign, which is an indication that the last update was too big and the algorithm has possibly overshot a local minimizer, the update value $\Delta_{ij}^{(t)}$ is decreased by η^{-}. If the derivative retains its sign, the update value is slightly increased to further accelerate convergence in shallow regions of the weight space.

In our experiments, the five parameters of the Rprop method were initialized using values commonly encountered in the literature. More specifically, the increase factor was set to $\eta^{+} = 1.2$; the decrease factor was set to $\eta^{-} = 0.5$; the initial

update value is set to $\Delta_0 = 0.07$; the maximum step, which prevents the weights from becoming too large, was $\Delta_{\max} = 50$; and the minimum step, which is used to avoid too small weight changes, was $\Delta_{\min} = 10^{-6}$.

1.2.2 The Adaptive Online Neural Network Training Algorithm

Despite the abundance of methods for learning from examples, there are only a few that can be used effectively for on–line learning. For example, the classic batch training algorithms cannot straightforwardly handle non–stationary data. Even when some of them are used in on–line training there exists the problem of "catastrophic interference", in which training on new examples interferes excessively with previously learned examples, leading to saturation and slow convergence [11].

Methods suited to on–line learning are those that can efficiently handle non–stationary and time–varying data, while at the same time, require relatively little additional memory and computation to process one additional example. The Adaptive Online Backpropagation (AOBP) algorithm [12,13] belongs to this class and can be used in on–line neural networks training. A high level description of the algorithm is given in Algorithm 1.

In the algorithm model η is the learning rate, K is the meta–learning rate and $\langle \cdot, \cdot \rangle$ stands for the usual inner product in $\mathbb{R}^n$. As the termination condition the classification error, or an upper limit to the error function evaluations can be used. The key features of this method are the low storage requirements and the inexpensive computations. Moreover, in order to calculate the learning rate for the next iteration, it uses information from the current, as well as, the previous iteration. This seems to provide some kind of stabilization in the calculated values of the learning rate, and previous experiments show that it helps the method to exhibit fast convergence and high success rate.

THE TRAINING ALGORITHM

0: Initialize the weights w^0, η^0, and K.
1: **Repeat**
2: Set $k = k + 1$
3: Randomly choose a pattern from the training set.
4: Using this pattern, calculate the error, $E(w^k)$ and then the gradient, $\nabla E(w^k)$.
5: Calculate the new weights using:
$w^{k+1} = w^k - \eta^k \nabla E(w^k)$
6: Calculate the new learning rate using:
$\eta^{k+1} = \eta^k + K\langle \nabla E(w^{k-1}), \nabla E(w^k) \rangle$
7: **Until** the *termination condition* is met.
8: **Return** the final weights w^{k+1}.

Algorithm 1: The Online Training Algorithm in Pseudocode

1.3 Evolutionary Algorithms

Evolutionary Algorithms (EAs) are stochastic search methods that mimic the metaphor of natural biological evolution. They operate on a population of potential solutions applying the principle of survival of the fittest to produce better and better approximations to a solution. At each generation, a new set of approximations is created by the process of selecting individuals according to their level of fitness in the problem domain and breeding them together using operators borrowed from natural genetics [14]. Many attempts have been made within the Artificial Intelligence community to integrate EAs and ANNs. We examine the application of EAs to microarray classification to determine the optimal, or near optimal, subset of predictive genes on the complex and large spaces of possible gene sets. Next we outline the Differential Evolution algorithm and its search operators.

The Differential Evolution Algorithm

Differential Evolution [15] is an optimization method, capable of handling non differentiable, nonlinear and multimodal objective functions. To fulfill this requirement, DE has been designed as a stochastic parallel direct search method, which utilizes concepts borrowed from the broad class of evolutionary algorithms. The method typically requires few, easily chosen, control parameters. Experimental results have shown that DE has good convergence properties and outperforms other well known evolutionary algorithms [15]. DE has been applied on numerous optimization tasks. It has successfully solved many artificial benchmark problems [16], as well as, hard real–world problems (see for example [17]). In [18] it was employed to train neural networks and in [19, 20] we have proposed a method to efficiently train neural networks having arbitrary, as well as, constrained integer weights. The DE algorithm has also been implemented on parallel and distributed computers [21, 22].

DE is a population–based stochastic algorithm that exploits a population of potential solutions, *individuals*, to effectively probe the search space. The population of the individuals is randomly initialized in the optimization domain with *NP*, *n*–dimensional vectors, following a uniform probability distribution and is evolved over time to explore the search space. *NP* is fixed throughout the training process. At each iteration, called *generation*, new vectors are generated by the combination of randomly chosen vectors from the current population. This operation in our context is referred to as *mutation*. The resulting vectors are then mixed with another predetermined vector – the *target* vector – and this operation is called *recombination*. This operation yields the so–called *trial* vector. The trial vector is accepted for the next generation depending on the value of the fitness function. Otherwise, the target vector is retained in the next generation. This last operator is referred to as *selection*.

The search operators efficiently shuffle information among the individuals, enabling the search for an optimum to focus on the most promising regions of the solution space. The first operator considered is mutation. For each individual x_g^i,

$i = 1, \ldots, NP$, where g denotes the current generation, a new individual v^i_{g+1} (mutant vector) is generated according to one of the following equations:

$$v^i_{g+1} = x^{\text{best}}_g + \mu(x^{r1}_g - x^{r2}_g), \tag{1.3}$$

$$v^i_{g+1} = x^{r1}_g + \mu(x^{r2}_g - x^{r3}_g), \tag{1.4}$$

$$v^i_{g+1} = x^i_g + \mu(x^{\text{best}}_g - x^i_g) + \mu(x^{r1}_g - x^{r2}_g), \tag{1.5}$$

$$v^i_{g+1} = x^{\text{best}}_g + \mu(x^{r1}_g - x^{r2}_g) + \mu(x^{r3}_g - x^{r4}_g), \tag{1.6}$$

$$v^i_{g+1} = x^{r1}_g + \mu(x^{r2}_g - x^{r3}_g) + \mu(x^{r4}_g - x^{r5}_g), \tag{1.7}$$

where x^{best}_g is the best member of the previous generation; $\mu > 0$ is a real parameter, called *mutation constant*, which controls the amplification of the difference between two individuals so as to avoid the stagnation of the search process; and $r_1, r_2, r_3, r_4, r_5 \in \{1, 2, \ldots, i-1, i+1, \ldots, NP\}$, are random integers mutually different.

Trying to rationalize the above equations, we observe that Equation (1.4) is similar to the crossover operator used by some Genetic Algorithms and Equation (1.3) derives from it, when the best member of the previous generation is employed. Equations (1.5), (1.6) and (1.7) are modifications obtained by the combination of Equations (1.3) and (1.4). It is clear that more such relations can be generated using the above ones as building blocks.

The recombination operator is subsequently applied to further increase the diversity of the mutant individuals. To this end, the resulting individuals are combined with other predetermined individuals, called the target individuals. Specifically, for each component l ($l = 1, 2, \ldots, n$) of the mutant individual v^i_{g+1}, we choose randomly a real number r in the interval $[0, 1]$. We then compare this number with the *recombination constant*, ρ. If $r \leqslant \rho$, we select, as the l–th component of the trial individual u^i_{g+1}, the l–th component of the mutant individual v^i_{g+1}. Otherwise, the l–th component of the target vector x^i_{g+1} becomes the l–th component of the trial vector. This operation yields the trial individual. Finally, the trial individual is accepted for the next generation only if it reduces the value of the objective function.

One problem when applying EAs, in general, is to find a set of control parameters which optimally balances the exploration and exploitation capabilities of the algorithm. There is always a trade off between the efficient exploration of the search space and its effective exploitation. For example, if the recombination and mutation rates are too high, much of the search space will be explored, but there is a high probability of losing good solutions. In extreme cases the algorithm has difficulty to converge to the global minimum due to the insufficient exploitation. Fortunately, the convergence properties of DE typically do not depend heavily on its control parameters.

Although, DE performs stably across the space of possible parameter settings, different operators may exhibit different convergence properties. More specifically, DE operators that use the best individual as a starting point for the computation of the mutant vector, constantly push the population closer to the location of the best computed point. On the other hand, operators that utilize many randomly chosen individuals for the computation of the mutant individual, greatly enhance the

exploration capability of the algorithm. In [23] we present a detailed study and experimental results on exploration vs. exploitation issues.

1.4 Feature Selection and Dimension Reduction Techniques

An important issue in any classification task is to define those features that significantly contribute to the classification of interest, while at the same time discarding the least significant and/or erroneous ones. This procedure is also referred to as dimension reduction. The problem of high dimensionality is often tackled by user specified subspaces of interest. However, user–identification of the subspaces is error–prone, especially when no prior domain knowledge is available. Another way to address high dimensionality is to apply a dimensionality reduction method to the dataset. Methods such as the Principal Component Analysis [24], optimally transform the original data space into a lower dimensional space by forming dimensions that are linear combinations of given attributes. The new space has the property that distances between points remain approximately the same as before. Alternatively, one can apply a clustering algorithm to the data set in order to reduce the dimensionality of the problem. To this end, the Principal Component Analysis, as well as several clustering algorithms used for dimension reduction are presented below.

1.4.1 Principal Component Analysis

In general, the Principal Component Analysis (PCA) is a powerful multivariate data analysis method [4]. Its main purpose is to reduce and summarize large and high dimensional datasets by removing redundancies and identifying correlation among a set of measurements or variables. It is a useful statistical technique that has found many applications in different scientific fields such as face recognition, image processing and compression, molecular dynamics, information retrieval, and gene expression analysis. PCA is mainly used in gene expression analysis in order to find an alternative representation of the data using a much smaller number of variables, as well as, to detect characteristic patterns in noisy data of high dimensionality. More specifically, PCA is a way of identifying patterns in data and expressing the data in such a way as to highlight their similarities and differences. Since patterns in high dimensional data can be hard to find, PCA is a powerful tool of analysis, especially when the visualization of the data is impossible.

Although PCA may succeed in reducing the dimensionality, the new dimensions can be difficult to interpret. Moreover, to compute the new set of dimensions information from all the original dimensions is required. The selection of a subset of attributes in the context of clustering is studied in [25, 26]. In the context of classification, subset selection has also been studied [24].

1.4.2 Reducing the Dimensions Using Clustering

Clustering can be defined as the process of "grouping a collection of objects into subsets or clusters, such that those within one cluster are more closely related to

each other than objects assigned to different clusters" [27]. Clustering is applied in various fields including data mining [28], statistical data analysis and social sciences [29], compression and vector quantization [30], global optimization [31, 32], image analysis, and others. Clustering techniques have been successfully applied to gene expression data [33–36] and have proved useful for identifying biologically relevant groupings of genes and samples [37].

Cluster analysis is one key step in understanding how the activity of genes varies during biological processes and is affected by disease states and cellular environments. In particular clustering can be used either to identify sets of genes according to their expression in a set of samples [34, 38], or to cluster samples into homogeneous groups that may correspond to particular macroscopic phenotypes [39]. The latter is in general more difficult, but is very valuable in clinical practice.

Identifying sets of genes that have a similar expression in a set of samples can lead to a successful dimension reduction technique. Aiming to this, clustering methodology can be applied to identify meaningful clusters of features (genes), and subsequently feature selection can be accomplished by selecting one or more representatives from each cluster. Such a selection can be based on the distance among the feature values and the identified cluster center. The feature with the minimum such distance from the cluster center can be a valid selection.

Although numerous clustering algorithms exist [40], mostly hierarchical clustering methods have been applied to microarray data. Hierarchical clustering algorithms construct hierarchies of clusters in a top–down (agglomerative) or bottom–up (divisive) fashion. This kind of algorithms have proved to give high quality results. One of the most representative hierarchical approaches is the one developed by Eisen et al. [34]. In that work, the authors employed an agglomerative algorithm and adopted a method for the graphical representation of the clustered dataset. This method has been widely used by many biologists and has become the most widely used tool in gene expression data analysis [33, 41, 42]. Nonetheless, the high sensitivity of agglomerative methods to small variations of the inputs and the high computational requirements, their usage is hindered in real applications, where the number of samples and their dimensionality is expected to be high (the cost is quadratic to the number of samples).

Partitioning clustering algorithms, start from an initial clustering (that may be randomly formed) and create flat partitionings by iteratively adjusting the clusters based on the distance of the data points from a representative member of the cluster. The most commonly used partitioning clustering algorithm is k–means. This algorithm initializes k centers and iteratively assigns each data point to the cluster whose centroid has the minimum Euclidean distance from the data point. Although, k–means type algorithms can yield satisfactory clustering results at a low cost, as their running time is proportional to kn, where n is the number of samples, they heavily depend on the initialization. Additionally, there is no automatic technique able to select the number of clusters k, but most of the times this is achieved by examining the results of successive re-executions of the algorithm.

Graph theoretical clustering approaches construct a proximity graph, in which each data point corresponds to a vertex, and the edges among vertices model their

proximity. Xing and Karp [43], developed a sample–based clustering algorithm named CLIFF (CLustering via Iterative Feature Filtering), which iteratively employs sample partitions as a reference to filter genes. The selection of genes through this approach relies on the outcome of an NCut algorithm, which is not robust to noise and outliers.

Another graph theoretical algorithm, CLICK (CLuster Identification via Connectivity Kernels) [35], tries to recognize highly connected components in the proximity graph as clusters. The authors demonstrated the superior performance of CLICK to the approaches of Eisen et al. [34], and the Self Organizing Map [44] based clustering approach. However, as claimed in [1], CLICK has little guarantee of not generating highly unbalanced partitions. Furthermore, in gene expression data, two clusters of co–expressed genes, C1 and C2, may be highly intersected with each other. In such situations, C1 and C2 are not likely to be split by CLICK, but would be reported as one highly connected component.

Finally, Alter et al. [45], by examining the projection of the data to a small number of principal components obtained through a Principal Component Analysis, attempt to capture the majority of gene variations. However, the large number of irrelevant genes does not guarantee that the discriminatory information will be highlighted to the projected data. For an overview of the related literature see [1–3, 46].

Below, we briefly describe five well–known clustering algorithms, namely, a) the unsupervised k–windows clustering algorithm [47,48]. (UkW) b) the Density–Based Spatial Clustering of Applications with Noise (DBSCAN) clustering algorithm [49], c) the Principal Direction Divisive Partitioning (PDDP) clustering algorithm [50], d) the fuzzy c–means (FCM) clustering algorithm [51], and e) the Growing Neural Gas (GNG) [52]. Note that the UkW, DBSCAN and GNG, apart from identifying the clusters, are also able to approximate the number of clusters present in the data set; thus no special knowledge about the data is required. However, PDDP and FCM need explicit determination of the cluster number. The PDDP, algorithm has also the ability to endogenously handle the large dimensionality since it is based on the PCA technique.

Unsupervised k–Windows Clustering Algorithm

One of the most important class of clustering algorithms are the density based methods [53–55], especially for data of low attribute dimensionality [56–58]. These methods operate by identifying regions of high density in dataset objects, surrounded by regions of low density. One recently proposed technique in this class is the "Unsupervised k–Windows" (UkW) [48], that utilizes hyperrectangles to discover clusters. The algorithm makes use of techniques from computational geometry and encapsulates clusters using linear containers in the shape of d–dimensional hyperrectangles that are iteratively adjusted with movements and enlargements until a certain termination criterion is satisfied [48, 59]. Furthermore, with proper tuning, the algorithm is able to detect clusters of arbitrary shapes [59].

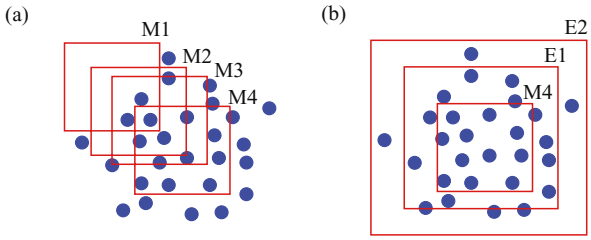

Fig. 1.1. (a) Sequential movements M2, M3, M4 of initial window M1. (b) Sequential enlargements E1, E2 of window M4

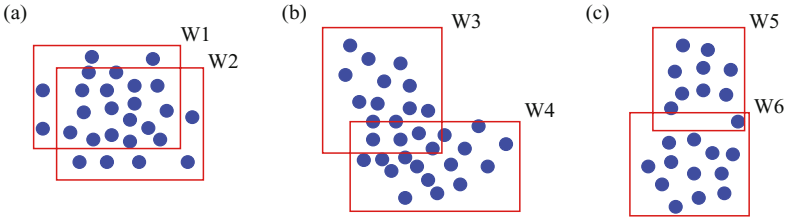

Fig. 1.2. (a) W1 and W2 satisfy the similarity condition and W1 is deleted. (b) W3 and W4 satisfy the merge operation and are considered to belong to the same cluster. (c) W5 and W6 have a small overlap and capture two different clusters

UkW aims at capturing all objects that belong to one cluster within a d–dimensional window. Windows are defined to be hyperrectangles (orthogonal ranges) in d dimensions [48]. UkW employs two fundamental procedures: *movement* and *enlargement*. The movement procedure aims at positioning each window as close as possible to the center of a cluster. The enlargement process attempts to enlarge the window so that it includes as many objects from the current cluster as possible. The two steps are illustrated in Figure 1.1.

A fundamental issue in cluster analysis, independent of the particular clustering technique applied, is the determination of the number of clusters present in a dataset. For instance well–known and widely used iterative techniques, such as the k–means algorithm [60] as well as the fuzzy c–means algorithm [51], require from the user to specify the number of clusters present in the data prior to the execution of the algorithm. UkW provides an estimate for the number of clusters that describe a dataset. The key idea is to initialize a large number of windows. When the movement and enlargement of all windows terminate, all overlapping windows are considered for merging by considering their intersection. An example of this operation is exhibited in Figure 1.2. For a detailed description of the algorithm see [59].

The DBSCAN Clustering Algorithm

The DBSCAN [49] clustering algorithm relies on a density–based notion of clusters and is designed to discover clusters of arbitrary shape as well as to distinguish

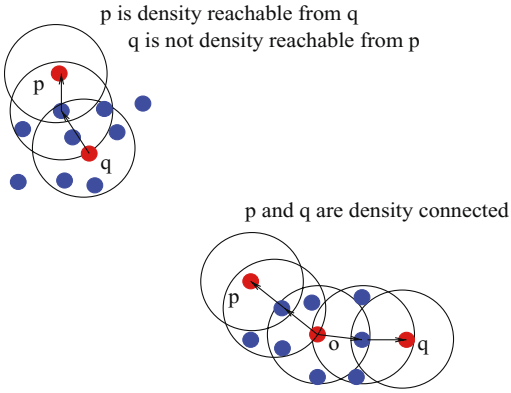

Fig. 1.3. An example of "Density–Reachable" and "Density Connected" points

noise. More specifically, the algorithm is based on the idea that in a neighborhood of a given radius (*Eps*) for each point in a cluster at least a minimum number of objects (*MinPts*) should be contained. Such points are called core points and each point in their neighborhood is considered as "Directly Density–Reachable" from that. Consequently the algorithm uses the notion of density reachable chains of objects; i.e. a point q is "Density–Reachable" from a point p, if there is a chain of objects $p_1,\ldots,p_k$ such that $p_1 = q$, $p_k = p$ and p_{i+1} is "Directly Density–Reachable" from p_i for $i = 1,\ldots,k$. Finally, a point p is defined as "Density Connected" to a point q, if there is a point o that both p,q are "Density–Reachable" from that. Fig 1.3, illustrates an example of these definitions.

Using the above described definitions, the algorithms considers as a cluster the subset of points from the dataset that are "Density–Reachable" from each other and additionally each pair of points inside the cluster is "Density Connected". Any point of the dataset not in a cluster is considered as noise.

To discover the clusters the algorithm retrieves density–reachable points from the data by iteratively collecting directly density–reachable objects. The algorithm scans the *eps*, neighborhood of each point in the database. If that neighborhood has more than *MinPts* points a new cluster C containing them is created. Then, the neighborhood of all points q in C which have not yet been processed is checked. If the points in neighborhood of q are more than *MinPts*, then those which are not already contained in C are added to the cluster and their neighborhood will be checked in a subsequent step. This procedure is iterated until no new point can be added to the current cluster C. In Fig 1.4, an example of the result of the DBSCAN algorithm is demonstrated, for three clusters of different sizes, with convex and non–convex shape. Additionally, some of the neighborhoods are depicted, to better illustrate the operation of the algorithm. For a detailed description of the algorithm see [54].

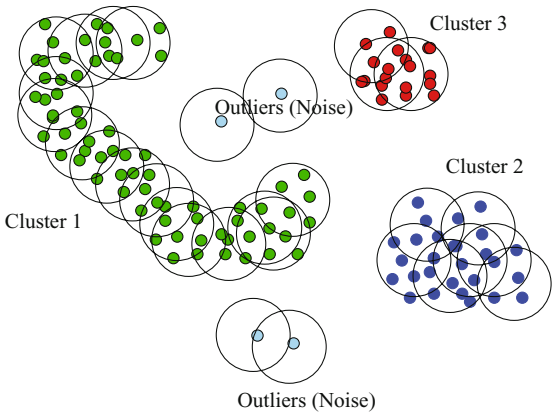

Fig. 1.4. An example of the result of the DBSCAN algorithm

The Fuzzy *c*–Means Clustering Algorithm

The Fuzzy c–Means (FCM) algorithm [51], considers each cluster as a fuzzy set. It firstly initializes a number of c prototype vectors (centroids) p^j over the dataset. The centroids represent the center of the clusters. Next it computes a degree of membership for every data vector x^i at each cluster using the membership function:

$$\mu_j(x^i) = \left(\sum_{l=1}^{c} \left(\frac{\|x^i - p^j\|}{\|x^i - p^l\|} \right)^{1/r-1} \right)^{-1},$$

which takes values in the interval $[0,1]$, where $r \in (1,\infty)$ determines the fuzziness of the partition. If r tends to 1_+, then the resulting partition asymptotically approaches a crisp partition. On the other hand, if r tends to infinity, the partition becomes a maximally fuzzy partition. Finally, the c prototypes are updated using the following equation:

$$P^j = \frac{\sum_{i=1}^{n} \left[m_j(x^i) \right]^r x^i}{\sum_{i=1}^{n} \left[m_j(x^i) \right]^r}.$$

This procedure is iteratively repeated until the measure of the distortion:

$$d = \sum_{j=1}^{c} \sum_{i=1}^{n} \left[m_j(x^i) \right]^r \|x^i - p^l\|^2,$$

changes less than a user defined threshold.

1.4.3 The PDDP Clustering Algorithm

The PDDP algorithm [50], is a divisive clustering algorithm. The key component in this algorithm is the computation of the principal directions of the data. Starting with

an initial cluster of all the data points, the algorithm iteratively splits the clusters. The use of a distance or similarity measure is limited to deciding which cluster should be split next, but the similarity measure is not used to perform the actual splitting. In detail, all the data points are projected onto the leading eigenvector of the covariance matrix of the data. Based on the sign of that projection the algorithm splits an initial cluster into two. This fact enables the algorithm to operate on extremely high dimensional spaces. PDDP, as well as PDDP(l) [61], which is a recent generalization of PDDP, does not provide a direct estimation for the number of clusters. Proposed methods that provide such estimations through these algorithms are based on scattering of the data around their centroids. Nonetheless, they tend to overestimate the true number of clusters resulting in rigid clustering [50, 61].

1.4.4 Growing Neural Gas

GNG [52] is an incremental neural network. It can be described as a graph consisting of k nodes, each of which has an associated weight vector, w_j, defining the node's position in the data space and a set of edges between the node and its neighbors. During the clustering procedure, new nodes are introduced into the network until a maximal number of nodes is reached. GNG starts with two nodes, randomly positioned in the data space, connected by an edge. Adaptation of weights, i.e. the nodes position, is performed iteratively. For each data object the closest node (winner), s_1, and the closest neighbor of a winner, node s_2, are determined. These two nodes are connected by an edge.

An age variable is associated with each edge. At each learning step the ages of all edges emanating from the winner are increased by 1. When the edge connecting s_1 and s_2 is created its age is set to 0. By tracing the changes of the age variable inactive nodes are detected. Any nodes having no emanating edges and edges exceeding a maximal age are removed.

The neighborhood of the winner is limited to its topological neighbors. The winner and its topological neighbors are moved in the data space toward the presented object by a constant fraction of the distance, defined separately for the winner and its topological neighbors. There is no neighborhood function or ranking concept. Thus, all topological neighbors are updated in the same manner.

1.4.5 A Hybrid Approach

The PCA technique optimally transforms the data set, with limited loss of information, to a space of significantly lower dimension. However, it is a global technique in the sense that does not deal with special characteristics that might exist in different parts of the data space.

To deal with this it is possible to hybridize a clustering algorithm and PCA. Firstly, the entire data set is partitioned into clusters of features, and next, each feature cluster can be independently transformed to a lower dimension space through the PCA technique. This application of the PCA is local and has the potential of better adapting to the special characteristics that might exist in the data set.

The techniques reported in this section should not be confused with any kind of *bi-clustering* approach [5, 6]. In the latter case the aim is to find subsets of genes that exhibit a similar behavior for a subset of samples. However the techniques reported below aim to either organize the genes in groups and infer compact representation for each group. Either they aim to recognize clusters of samples that have a physical common character. Sometimes to achieve this the compact representations inferred from the initial procedure are employed but this is quite different to the bi-clustering point of view.

1.5 Experimental Analysis

In this Section we initially describe the microarray problems used in the remaining of this chapter. Then, we perform an extensive evaluation of various clustering algorithms for supervised as well as unsupervised classification of the data sets. Subsequently, we implement and test FNN classifiers combined with clustering methods and the PCA dimension reduction technique. Finally, we report results from a hybrid approach that utilizes EAs for gene selections and FNNs for classification.

1.5.1 DNA Microarray Problems

The evaluation of all the Computational Intelligence algorithms presented in this chapter is performed through the following well–known and publicly available data sets:

(a) The ALL–AML data set [39]. This study examines mRNA expression profiles from 72 leukemia patients to develop an expression–based classification method for acute leukemia. In the data set each sample is measured over 7129 genes. The first 38 samples were used for the clustering process (train set), while the remaining 34 were used to evaluate the clustering result (test set). The initial 38 samples contained 27 acute myeloid leukemia (ALL) samples and 11 acute lymphoblastic leukemia (AML) samples. The test set contained 20 ALL samples and 14 AML samples. The data set is available at:
http://www.broad.mit.edu/cancer/pub/all_aml

(b) The COLON data set [33] consists of 40 tumor and 22 normal colon tissues. For each sample there exist 2000 gene expression level measurements. The data set is available at:
http://microarray.princeton.edu/oncology

(c) The PROSTATE data set [62] contains 52 prostate tumor samples and 50 non-tumor prostate samples. For each sample there exist 6033 gene expression level measurements. It is available at:
http://www.broad.mit.edu/cgi-bin/cancer/datasets.cgi

(d) The LYMPHOMA dataset [41] that contains 62 samples of the 3 lymphoid malignancies samples types. The samples are measured over 4026 gene expression levels. This dataset is available at:
http://genome-www.stanford.edu/

All the data sets contain a relatively large number of patients and have been well characterized and studied. Notice that no additional preprocessing or alteration was performed to the data, except for the application of the methods described in this chapter.

1.5.2 Evaluation of the Clustering Algorithms

In the literature, both supervised and unsupervised classifiers have been used to build classification models from microarray data. Supervised classifiers employ predefined information about the class of the data to build the classification model. On the other hand, no class information is necessary to the unsupervised methods.

To investigate the performance of the clustering algorithms on gene expression microarray data we primarily employ the data set from the ALL–AML microarray problem. We performed two independent sets of experiments. In the first set, the clustering methodology was applied on two previously published gene subsets as well as their union. The comparative results assess the comparative performance of the clustering algorithms.

In the second set of experiments, we do not use class information for the gene selection. To this end, to reduce the dimensionality of the problem we use the PCA technique, as well as, dimension reduction through clustering. This second set of experiments is closer to real life applications where no class information is a priori known. Moreover, the hybridization of clustering and the PCA is evaluated. The hybrid scheme seems also able to provide results equivalent to those obtained with the supervised gene selection. Thus, this scheme is also applied on the remaining three data sets for further evaluation.

Clustering Based on Supervised Gene Selection

Generally, in a typical biological system, it is often not known how many genes are sufficient to characterize a macroscopic phenotype. In practice, a working mechanistic hypothesis that is testable and largely captures the biological truth, seldom involves more than a few dozens of genes. Therefore, identifying the relevant genes is critical [43]. Initially, we intended to study the performance of the UkW clustering algorithm, so we applied it over the complete ALL–AML train set. The algorithm was applied to the measurements of the 7129 genes, as well as various randomly selected gene subsets having from 10 to 2000 genes each. The algorithm produced clusters that often contained both AML and ALL samples. Typically, at least 80% of all the samples that were assigned to a cluster were characterized by the same leukemia type.

To improve the quality of the clustering, it proved essential to identify sets of genes that significantly contribute to the partition of interest. Clearly, there exist many such sets and it is difficult to determine the best one. To this end, we tested the UkW clustering algorithm on two previously discovered subsets of significant genes. The first set has been published in the original paper of Golub et al. [39] (we call it $GeneSet_1$), while the second set was proposed by Thomas et al. [63] ($GeneSet_2$).

Each dataset contains 50 genes. Furthermore, we tested the clustering algorithms on the union of the above gene sets ($GeneSet_3$), consisting of 72 genes.

In [39] $GeneSet_1$ was constructed by electing 50 highly correlated genes with the ALL–AML class distinction. Next, the authors used a Self Organizing Map [64] based clustering approach, to discover clusters on the training set. SOM automatically grouped the 38 samples into two classes, one containing 24 out of the 25 ALL samples, and the other containing 10 out of the 13 AML samples.

Regarding the second set of genes ($GeneSet_2$), the 50 most highly correlated genes with the ALL–AML class distinction (top 25 differentially expressed probe sets in either sample group) have been selected. More specifically, the selection approach is based on well–defined assumptions, uses rigorous and well–characterized statistical measures, and tries to account for the heterogeneity and genomic complexity of the data. The modelling approach uses known sample group membership to focus on expression profiles of individual genes in a sensitive and robust manner, and can be used to test statistical hypotheses about gene expression. For more information see [63].

Applying the UkW algorithm on those 3 gene train sets, each produced 6 clusters containing ALL or AML samples. Table 1.1 exhibits the results. More specifically, the algorithm using $GeneSet_1$ discovered 4 ALL clusters and 2 AML clusters (3 misclassifications), while using $GeneSet_2$ discovered 4 clusters containing only ALL samples and 2 clusters containing only AML samples (0 misclassifications). The algorithm discovered 4 ALL clusters and 2 AML clusters (1 misclassification) when applied to $GeneSet_3$. $GeneSet_2$ yielded the best results in the training set (followed by $GeneSet_3$).

Table 1.1. The performance of the UkW algorithm for the different train sets

Clustering result for the train set $GeneSet_1$ ALL accuracy: 87.5% — AML accuracy: 100%						
Leukemia type	ALL Clusters				AML Clusters	
	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 1	Cluster 2
ALL	4	4	12	4	3	0
AML	0	0	0	0	4	7
Clustering result for the train set $GeneSet_2$ ALL accuracy: 100.0% — AML accuracy: 100%						
Leukemia type	ALL Clusters				AML Clusters	
	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 1	Cluster 2
ALL	10	3	10	4	0	0
AML	0	0	0	0	8	3
Clustering result for the train set $GeneSet_3$ ALL accuracy: 95.83% — AML accuracy: 100%						
Leukemia type	ALL Clusters				AML Clusters	
	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 1	Cluster 2
ALL	8	9	5	4	0	1
AML	0	0	0	0	7	4

Table 1.2. The performance of the UkW algorithm for the different test sets

Clustering result for the test set $GeneSet_1$ ALL accuracy: 60.00% — AML accuracy: 92.85%						
Leukemia type	ALL Clusters				AML Clusters	
	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 1	Cluster 2
ALL	2	0	7	3	8	0
AML	1	0	0	0	8	5
Clustering result for the test set $GeneSet_2$ ALL accuracy: 100% — AML accuracy: 78.57%						
Leukemia type	ALL Clusters				AML Clusters	
	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 1	Cluster 2
ALL	8	0	9	3	0	0
AML	0	0	3	0	8	3
Clustering result for the test set $GeneSet_3$ ALL accuracy: 90% — AML accuracy: 100%						
Leukemia type	ALL Clusters				AML Clusters	
	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 1	Cluster 2
ALL	10	4	3	1	0	2
AML	0	0	0	0	5	9

Table 1.3. Comparative results for the test set $GeneSet_3$

	Misclassified samples		Number of clusters		Accuracy (%)	
	train set	test set	train set	test set	AML	ALL
DBSCAN	1	3	4	4	78.5	100
FCM	1	2	4	4	85.7	100
GNG	1	3	3	3	78.5	100
PDDP	2	4	6	6	71.4	100
UkW	1	2	6	6	100.0	90.0

To further evaluate the clustering results each sample from each test set was assigned to one of the clusters discovered in the train set according to its distance from the cluster center. Specifically, if an ALL (AML) sample from the test set was assigned to an ALL (AML, respectively) cluster then that sample was considered correctly classified. From the results exhibited in Table 1.2 it is evident that using the clustering from $GeneSet_1$ one AML and eight ALL samples from the test set were misclassified, resulting in a 73.5% correct classification. The clusters discovered using $GeneSet_2$ resulted in three misclassified AML samples (91.2% correct classification), while $GeneSet_3$ clusters yielded the best performance with only two misclassified ALL samples (94.1% correct classification).

In Table 1.3 we present comparative results from the test set $GeneSet_3$ only, as all the clustering algorithms exhibited improved classifications performance on this dataset. The best performance was achieved by the UkW algorithm and the FCM, followed by the DBSCAN and GNG algorithms. Notice that the FCM requires from the user to supply the number of clusters (supervised clustering) and that the

DBSCAN algorithm did not classify seven samples of the train set and five samples of the test set (all of them belonging in the AML class), since it characterized them as outliers.

Although the PDDP algorithm exhibited the worst classification performance, it must be noted that it was the only algorithm capable of using all the 7129 genes to cluster the samples. Using the complete set of genes, the PDDP algorithm misclassified two training set samples and eight test set samples.

Clustering Based on Unsupervised Gene Selection

Not using the class information to perform gene selection, we have to resort to unsupervised methods. We employ the UkW algorithm, for this task since it proved quite successful in the previous set of experiments. More specifically, the UkW algorithm was applied over the entire data set to select clusters of genes. Feature selection was accomplished by extracting from each cluster one representative feature (gene), based on the Euclidean distance among the feature values and the identified cluster center. The feature with the minimum distance from the cluster center was selected. This approach produced a new subset containing 293 genes ($GeneSet_4$).

The UkW algorithm was then applied on $GeneSet_4$ to group the samples. The results are illustrated in Table 1.4. From this table it is evident that high classification accuracy is possible even when class information is not known. Specifically, UkW exhibited accuracy of 93.6% and 76% for the ALL and the AML samples, respectively.

A second set of experiments is performed using the PCA technique for dimension reduction. A common problem when using PCA is that there is no clear answer to the question of how many factors should be retained for the new data set. A rule of thumb is to inspect the *scree plot*, i.e. plot all the eigenvalues in decreasing order. The plot looks like the side of a hill and "scree" refers to the debris fallen from the top and lying at its base. The scree test suggests to stop analysis at the point the mountain (signal) ends and the debris (error) begins. However, for the considered problem the scree plot was indicative, but not decisive. The scree plot, exhibited in Figure 1.5 (left), suggests that the contributions are relatively low after approximately ten components. In our experiments, we tried all the subsets using factors from 2 to 70. The classification accuracy is shown in Figure 1.5 (right). The best performance was attained when 25 factors were used (84.72%).

Although, the PCA technique tries to limit the loss of information, the classification accuracy is significantly lower, compared to the results obtained by supervised

Table 1.4. The performance of the UkW algorithm for the $GeneSet_4$ data set

Clustering result for the set $GeneSet_4$ ALL accuracy: 93.61% — AML accuracy: 76%						
Leukemia type	ALL Clusters					AML Cluster
	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5	Cluster 1
ALL	12	5	8	16	3	3
AML	2	0	3	0	1	19

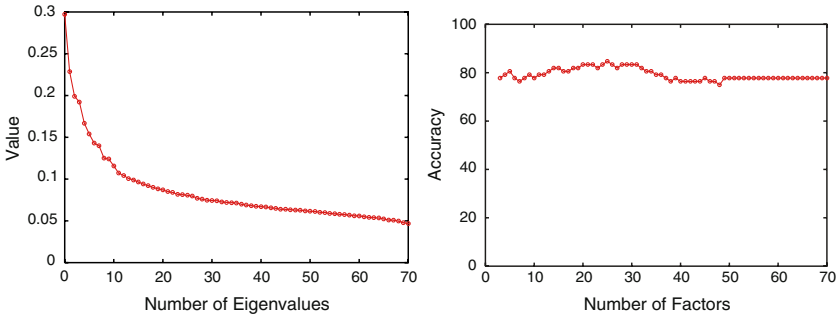

Fig. 1.5. Plot of the 70 first eigenvalues in decreasing order (*left*) and the corresponding classification accuracies (*right*)

Table 1.5. The performance of the UkW algorithm for the $GeneSet_5$ data set

Clustering result for the set $GeneSet_5$ ALL accuracy: 97.87% — AML accuracy: 88%						
Leukemia type	ALL Clusters				AML Clusters	
	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 1	Cluster 2
ALL	7	14	14	11	1	0
AML	0	0	3	0	13	9

gene selection, Next, we study the hybridization of the clustering the PCA technique, with the aim to obtain more informative representations of the data.

To this end, the entire data set is firstly partitioned into clusters of features using the UkW algorithm. Next, each feature cluster is independently transformed to a lower dimension space through the PCA technique. Regarding the number of factors selected from each cluster many approaches could be followed. In our experiments only two factors from each cluster were selected, resulting in $GeneSet_5$. Experiments conducted using scree plots exhibited identical results. Our experience is that the number of selected factors from each cluster is not critical, since the entire data set has already been partitioned to a specific cluster number determined by the algorithm itself. Finally, the algorithm is again applied to group the samples into clusters and the results are exhibited in Table 1.5. The UkW exhibited accuracy 97.87% and 88% for the ALL and the AML samples, respectively.

Overall, the obtained experimental results regarding the various gene subsets indicate that using $GeneSet_1$ and $GeneSet_2$, yields very satisfactory results. The best results were obtained using the union of the genes in $GeneSet_1$ and $GeneSet_2$. The drawback of this feature selection scheme is that it relies on human expertise ($GeneSet_1$) and requires class information ($GeneSet_2$) to construct the final dataset ($GeneSet_3$). On the other hand, performing unsupervised gene selection using either PCA or UkW may result in a lower classification accuracy.

The hybridization of the two approaches yielded results comparable to those obtained through the first three gene sets. The main drawback of this approach is that it requires information from all the genes.

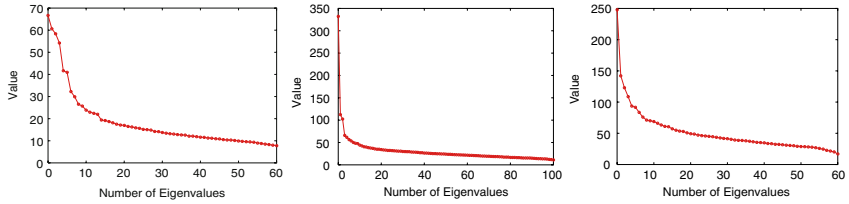

Fig. 1.6. Plot of the first eigenvalues in decreasing order, for the COLON (*left*), PROSTATE (*middle*) and the LYMPHOMA (*right*) datasets

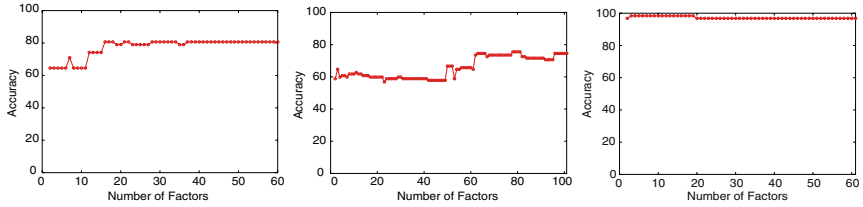

Fig. 1.7. Classification accuracy for all the factors, for the COLON (*left*), PROSTATE (*middle*) and the LYMPHOMA (*right*) datasets

To further investigate the efficiency of the hybridization scheme we compare it against the PCA dimension reduction technique on the COLON, the PROSTATE and the LYMPHOMA microarray data sets. While the hybrid approach automatically determines the number of reduced dimensions only the screen plot can provide such an information for the PCA technique. Although the scree plots, reported in Figure 1.6, provide an indication they are not conclusive. Generally, in all three cases the contributions are relatively low after approximately twenty components.

In our experiments, we tried all available factors for each dataset and utilized the UkW algorithm for the classification. The classification accuracy of the UkW clustering algorithm for each of the three datasets and all the available factors is reported in Figure 1.7. For the COLON dataset the best classification accuracy obtained was 80.64% employing 16 factors. For the PROSTATE dataset the best result was 82.35% classification accuracy, using 71 factors. Finally, for the LYMPHOMA dataset the best result was 98.38% classification accuracy using only 3 factors.

The results of the hybrid approach, for the three datasets, are presented in Table 1.6. As it is evident, the classification accuracy of the resulting partitions increases in all three cases. The high number of factors that the hybrid scheme decides to use, does not impose a problem to the algorithm since they originate in different clusters, and they are not correlated to each other. Furthermore, the additional advantage of the automatic determination of the required factors, exhibits a robust result that is not possible through the PCA technique. The classification accuracies obtained are considered very high, in comparison to other previously published approaches [65].

Table 1.6. The performance of the hybrid approach for the COLON, PROSTATE and LYMPHOMA datasets

Dataset	Number of Factors Used	Classification Accuracy (%)
COLON	229	82.25
PROSTATE	84	83.3
LYMPHOMA	103	99.01

1.5.3 Using Clustering and Feedforward Neural Networks Classifiers

Although the Feedforward Neural Networks (FNNs) trained using the PCA projection of the dataset can provide high classification accuracy, there is no straightforward interpretation of the new dimensions. Consequently, to compute features for a new patient, information from all the genes is required. On the other hand, the clustering algorithms identify a subset of genes that significantly contribute to the partition of interest. Thus, only the expression levels of the selected genes are needed for the future operation of the system. Unfortunately, there exist many such subsets and it is difficult for any clustering algorithm to determine the best one.

The first step towards the implementation of such a system is to apply a clustering algorithm over the entire training sets. Dimension reduction is performed by selecting a representative feature from each identified feature cluster, as usual. The representative features will be used as input to the FNN classifier. To this end, the (supervised) FCM and the (unsupervised) UkW clustering algorithms were applied on the three data sets mentioned above. Since the number of clusters, c, present in each data set is unknown, all possible values from 3 to 30 were tried for the FCM algorithm. On the other hand, the UkW clustering algorithm was executed only once and it provided 14 features for the COLON data set, 18 features for the PROSTATE data set, and 22 features for the ALL–AML data set.

Consequently, an FNN having two hidden layers consisting of 5 neurons each, was trained using the Rprop and the AOBP training algorithms to classify the features of the data sets. In the experiments, we performed random splitting of the data into learning and test sets. Specifically, the data was partitioned randomly into a learning set consisting of two-thirds of the whole set and a test set consisting of the remaining one-third. To reduce the variability, the splitting was repeated 50 times as in [65]. For each splitting 50 independently initialized FNNs were trained, resulting in a total of 2500 experiments. The comparative results for the three problems considered here are illustrated using boxplots in Figures 1.8, 1.9, and 1.10, respectively. Each boxplot depicts the obtained values for the classification accuracy, in the 2500 experiments. The box has lines at the lower quartile, median, and upper quartile values. The lines extending from each end of the box (whiskers) indicate the range covered by the remaining data. The outliers, i.e. the values that lie beyond the ends of the whiskers, are represented with crosses. Notches represent a robust estimate of the uncertainty about the median. From these figures it is evident that the UkW algorithm exhibited the best performance. The mean classification success for each problem was 65.9%, 73.5%, and 69.2%, clearly above the mean classification success of FCM regardless

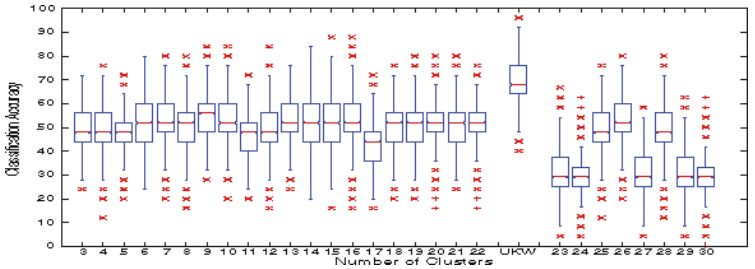

Fig. 1.8. COLON: Classification accuracy of FNNs incorporating the FCM and the UkW clustering algorithms

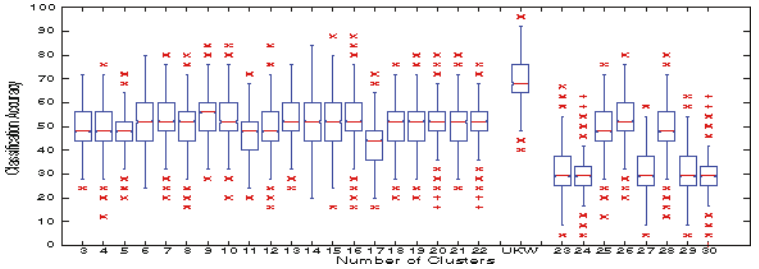

Fig. 1.9. PROSTATE: Classification accuracy of FNNs incorporating the FCM and the UkW clustering algorithms

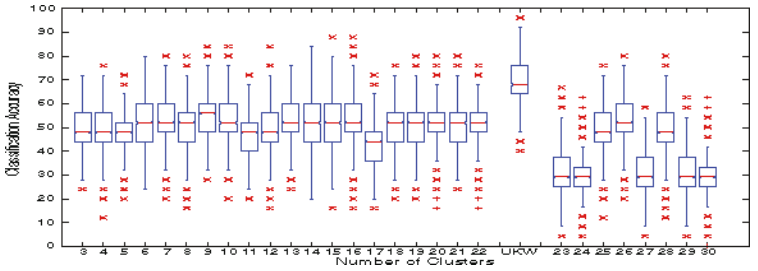

Fig. 1.10. ALL–AML: Classification accuracy of FNNs incorporating the FCM and the UkW clustering algorithms

the value of c. Moreover, FCM's results were heavily dependent on the number of features selected.

In spite of UkW algorithm's good results, this first set of experiments revealed the limitation of the direct application of any clustering algorithm, since even better classification accuracy is possible (see for example [65]). As a next step, we examine the classification accuracy on PCA derived features. Since for the for the considered problems the scree plots were indicative, but not decisive (see Figure 1.6) for the number of factors to use, we tried all of them from 3 to 30. As above, 50

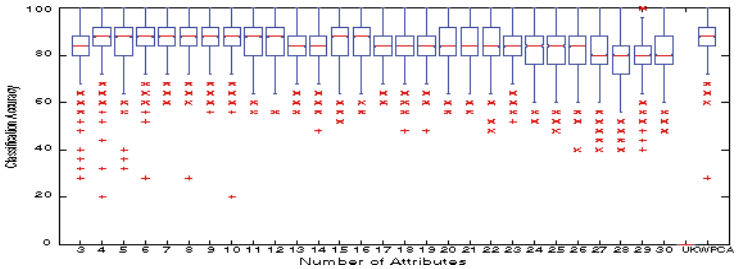

Fig. 1.11. COLON: Classification accuracy of FNNs incorporating the PCA technique and the proposed UkWPCA scheme

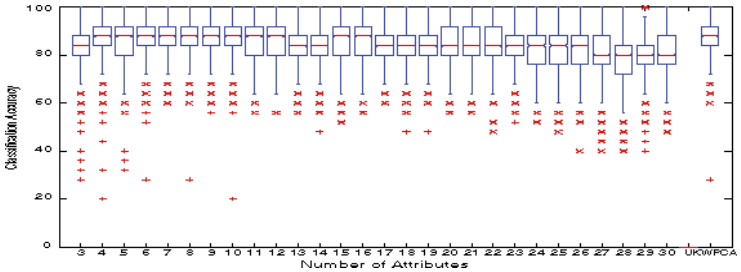

Fig. 1.12. PROSTATE: Classification accuracy of FNNs incorporating the PCA technique and the proposed UkWPCA scheme

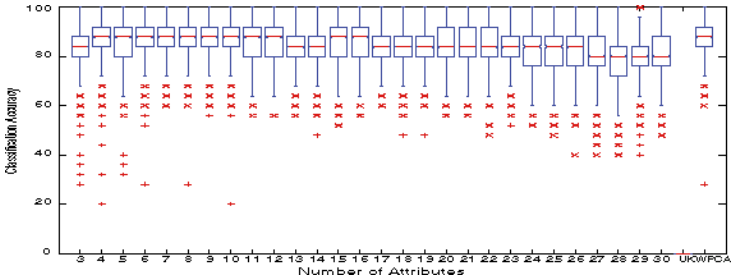

Fig. 1.13. ALL–AML: Classification accuracy of FNNs incorporating the PCA technique and the proposed UkWPCA scheme

random splittings of the data were performed and 50 independently initialized FNNs were trained. The results from the 2500 experiments for each problem are illustrated in Figures 1.11, 1.12, and 1.13, respectively. The results show that the classification accuracy depends on the number of factors used and that the best results do not exactly match the scree plot indication. Although, the FNNs trained using the PCA projection of the data set, in general, provide high classification accuracy, there is no straightforward way to select the right number of factors for each problem. FNNs using features computed by the PCA technique exhibited mean classification

accuracy 79.1%, 86.5%, and 88.5%, for the optimal selection of the number of factors.

The above discussion suggests that the UkW algorithm is capable of automatically identifying meaningful groups of features, while the PCA technique optimally transforms the data set, with limited loss of information, to a space of significantly lower dimension. Since both properties are desirable for an automatic classification system, we next examine the classification accuracy using the hybrid approach to reduce the dimension.

As in the previous sets of experiments, we performed 50 random splittings of the data set and consequently 50 independently initialized FNNs were trained using the Rprop algorithm. The classification accuracy of the proposed system (UkWPCA) is illustrated in the last column of Figures 1.11, 1.12, and 1.13, respectively. To summarize, the UkW algorithm automatically provided a good approximation of the number of clusters present in the data sets, while the PCA technique transformed the discovered clusters resulting in the most informative features. FNNs trained using these features had the highest classification accuracy and the most robust performance. Specifically, the mean classification accuracies for the three problems considered here were 80.3%, 87.1%, and 87.4%, respectively.

1.5.4 Using Evolutionary Algorithms and Feedforward Neural Networks Classifiers

Here, we propose the application of EAs to microarray classification to determine the optimal, or near optimal, subset of predictive genes on the complex and large space of possible gene sets. Although a vast number of gene subsets are evaluated by the EA, selecting the most informative genes is a non trivial task. Common problems include the existence of: a) relevant genes that are not included in the final subset, because of the insufficient exploration of the gene pool, b) significantly different subsets of genes being the most informative as the evolution progresses, and c) many subsets that perform equally well, as they all predict the test data satisfactorily. From a practical point of view, the lack of a unique solution does not seem to present a problem.

The EA approach we propose utilizes the DE algorithm and maintains a population of trial gene subsets, imposes random changes on the genes that compose those subsets, and incorporates selection (driven by a neural network classifier) to determine which are the most informative ones. Only those genes are maintained in successive generations; the rest are removed from the trial pool. At each iteration, every subset is given as input to an FNN classifier and the effectiveness of the trained FNN determines the fitness of the subset of genes. The size of the population and the number of features in each subset are parameters that we explore experimentally. For the experiments reported in this chapter we employed Equation (1.3) as the main DE search operator.

For the approach discussed above, each population member represents a subset of genes, so a special representation and a custom fitness function must be designed. When seeking subsets containing n genes, each individual consists of n integers. The

first integer is the index of the first gene to be included in the subset, the second integer denotes the number of genes to skip until the second gene to be included is reached, the third integer component denotes the number of genes to skip until the third included gene, and so on. This representation was necessary in order to avoid multiple inclusion of the same gene. Moreover, a version of DE that uses integer vectors has been proposed and thoroughly studied in previous studies [19, 20, 22].

Let us now focus on the custom fitness function. Initially, the k–nearest neighbors (KNN) classifier was used as a fitness function to evaluate the fitness of each gene subset. KNN classification is based on a distance function such as the Euclidean distance or Pearson's correlation that is computed for pairs of samples in n–dimensional space. Each sample is classified according to the class memberships of its k–nearest neighbors, as these are determined by the distance function. KNN has the advantage of simplicity and it usually performs well on data sets that are not linearly separable. However, our preliminary experimental results indicated that although the evolutionary algorithm produces gene subsets that help the KNN classifier to achieve high classification accuracy on the training samples, KNN fails to correctly classify the test data.

Thus we decided to use FNNs instead of the KNN classifier. The utilization of FNNs as fitness function greatly improved the classification accuracy of this approach. An FNN was trained using each subset of genes and the fitness of the subset is scored by analyzing how well the FNN separates the training data into separate classes. One third of the data set is used as a training set for the FNN and one third is used to measure the classification accuracy of the FNN classifier. The remaining patterns of the data set are kept to estimate the classification capability of the final gene subset.

Below, we report the experimental results. We have tested and compared the performance of the this approach on many publicly available microarray data sets. Here we report results from the COLON and the PROSTATE data sets. Since the appropriate size of the most predictive gene set is unknown, DE was employed for various gene set sizes ranging from 10 to 100 with a step of 10. The FNN used at the fitness function consisted of two hidden layers with eight and seven neurons, respectively. The input layer contained as many neurons as the size of the gene set. One output neuron was used at the output layer whose value for each sample determined the network classification decision. Since both problems had two different classes for the patterns, a value lower than 0.5 regarded the pattern to belong to the first class; otherwise regarded it to belong to the second class.

For each different gene set size the data were partitioned randomly into a learning set consisting of two–thirds of the whole set and a test set consisting of the remaining one third, as already mentioned. The one third of the training set was used by the Rprop and the AOBP algorithms to train the FNNs. The performance of the respective gene set was measured according to the generalization of the trained FNN on the rest of the training set. Both the Rprop and the AOBP training algorithms exhibited stable performance and are suitable for this kind of tasks. Note that, the test set was only used to evaluate the classification accuracy that can be obtained using the final gene set discovered by the DE algorithm. To reduce the variability,

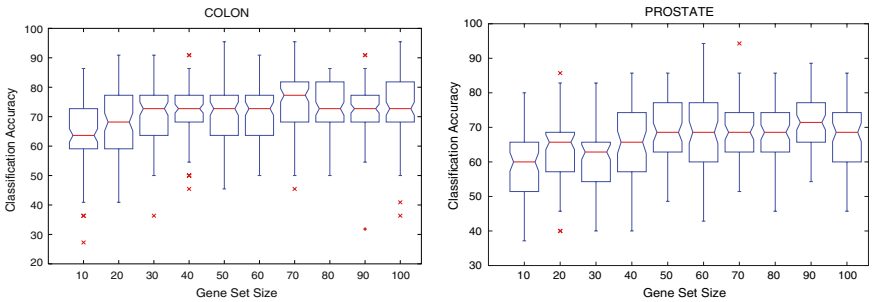

Fig. 1.14. Classification accuracy obtained by FNNs trained using the DE selected gene set for the COLON (*left*) and PROSTATE (*right*) datasets

the splitting was repeated 10 times and 10 independent runs were performed each time, resulting in a total of 100 experiments, for gene set size.

The classification accuracy of the proposed system is illustrated using boxplots in Figure 1.14. Each boxplot depicts the obtained values for the classification accuracy, in the 100 experiments. As demonstrated, using a gene set size of 50–80 for the COLON dataset the algorithm managed to achieve the best results. The same is achieved for the PROSTATE dataset for a gene set size ranging from 40 to 60. The experimental results are comparable to those obtained by other approaches [65, 66].

1.6 Concluding Remarks

Although the classification of the data obtained from microarray studies is very important in medical diagnosis of many diseases, it still presents a challenge for data analysis. This is due to the tremendous amount of available data (typically several Gigabytes of data), the redundant, erroneous or incomplete data sets, and the high dimensionality. Thus, the application of techniques for dimension reduction and/or selection of subsets of informative genes are essential to counter this very difficult problem. The selection of gene subsets that retain high predictive accuracy for certain cell–type classification, poses a central problem in microarray data analysis. The application and combination of various Computational Intelligence methods holds a great promise for automated feature selection and classification.

To summarize, in this chapter we have presented, implemented and tested supervised clustering algorithms, unsupervised clustering algorithms, the Principal Component Analysis dimension reduction technique, Feedforward Artificial Neural Networks, Evolutionary Algorithms, and hybrid approaches. Our goal was to evaluate and compare various approaches in an attempt to investigate their weaknesses and their shortcomings with respect to DNA microarray data analysis and classification.

Neural Networks have traditionally been used by the research community due their easiness of implementation and their high quality results. Their application to the microarray data, needs the existence of a preprocessing phase that would

reduce the dimension of the learning space. Using either Evolutionary techniques in a supervised manner or unsupervised cluster analysis significant results can be obtained. However, the unsupervised characteristics of the latter approach provide an intuitive advantage. On the other hand, using cluster analysis directly to infer knowledge without resorting to an external trainer as in the Neural Network case, also seems quite promising.

Among the different clustering algorithms studied, the density based approaches provided the best results. The experimental results of the DBSCAN, the UkW and the GNG algorithms are indicative of how effective is their feature of automatic discovery of the cluster number. However, in the case of GNG the numerous parameters seem to deteriorate its clustering accuracy. All the above comments are true in the case that user-defined information about the most important subset of genes is used.

In the case, that no such information is available our first resort is the PDDP clustering algorithm, that can be directly applied to the original high dimensional space. However, it results in low performance clustering, which can be attributed to its crude splitting technique. Nevertheless, by borrowing the idea of PDDP, that is to apply PCA to different parts of the data space, we can design a hybrid method that is completely automated and does not require any kind of external steering. To this end, we examined how the hybrid techniques can further improve the classification accuracy of traditional classifiers such as Neural Networks. These results are not restrictive to FNNs, but can be straightforwardly extended to other types of classifiers, such as Support Vector Machines, Probabilistic Neural Networks, etc.

Briefly, we can claim that the reported experimental results indicate that there exists no unique and clear solution to this hard real–life problem. One must try different approaches in order to gain insight and better analyze the DNA microarray data. However, Computational Intelligence techniques are clearly capable of:

(a) exhibiting high classification success rates,
(b) having completely automatic operation,
(c) discovering the subsets of features that contribute significantly,
(d) constructing non–linear relationships between the input and the output.

Thus, even when compared against the best known alternative methods, Computational Intelligence techniques seem to prevail. Extensive experiments on publicly available microarray datasets indicate that the approaches proposed and studied here are fast, robust, effective and reliable. However, further testing on bigger data sets from new microarray studies is necessary before we can establish a general, flexible all-purpose methodology.

References

1. Jiang, D., Tang, C., Zhangi, A.: Cluster analysis for gene expression data: A survey. IEEE Transactions on Knowledge and Data Engineering **16**(11) (2004) 1370–1386
2. Larranaga, P., Calvo, B., Santana, R., Bielza, C., Galdiano, J., Inza, I., Lozano, J.A., Armananzas, R., Santafe, G., Perez, A., Robles, V.: Machine learning in bioinformatics. Briefings in Bioinformatics **7**(1) (2006) 86–112

3. Statnikov, A., Aliferis, C.F., Tsamardinos, I., Hardin, D., Levy, S.: A comprehensive evaluation of multicategory classification methods for microarray gene expression cancer diagnosis Bioinformatics **21**(5) (2005) 631–643
4. Wall, M., Rechtsteiner, A., Rocha, L.: Singular value decomposition and principal component analysis. In: A Practical Approach to Microarray Data Analysis. Kluwer (2003) 91–109
5. Van Mechelen, I., Bock, H.H., De Boeck, P.: Two-mode clustering methods:a structured overview. Statistical Methods in Medical Research **13**(5) (2004) 363–394
6. Kung, S.Y., Mak, M.W.: A Machine Learning Approach to DNA Microarray Biclustering Analysis. In: Proceedings of the IEEE International Workshop on Machine Learning for Signal Processing, (2005) 314–321
7. Wang, Z., Wang, Y., Xuan, J., Dong, Y., Bakay, M., Feng, Y., Clarke, R., Hoffman, E.P.: Optimized multilayer perceptrons for molecular classification and diagnosis using genomic data. Bioinformatics **22**(6) (2006) 755–761
8. Rumelhart, D., Hinton, G., Williams, R.: Learning internal representations by error propagation. MIT Press Cambridge, MA, USA (1986)
9. Gill, P., Murray, W., Wright, M.: Practical optimization. London: Academic Press, (1981)
10. Riedmiller, M., Braun, H.: A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In: Proceedings of the IEEE International Conference on Neural Networks, San Francisco, CA. (1993) 586–591
11. Sutton, R., Whitehead, S.: Online learning with random representations. Proceedings of the Tenth International Conference on Machine Learning (1993) 314–321
12. Magoulas, G., Plagianakos, V.P., Vrahatis, M.N.: Development and convergence analysis of training algorithms with local learning rate adaptation. In: IEEE International Joint Conference on Neural Networks (IJCNN'2000), **1** (2000) 21–26.
13. Plagianakos, V.P., Magoulas, G., Vrahatis, M.N.: Global learning rate adaptation in on-line neural network training. In: Second International ICSC Symposium on Neural Computation (NC'2000). (2000)
14. Bäck, T., Schwefel, H.: An overview of evolutionary algorithms for parameter optimization. Evolutionary Computation **1**(1) (1993) 1–23
15. Storn, R., Price, K.: Differential evolution – a simple and efficient adaptive scheme for global optimization over continuous spaces. Journal of Global Optimization **11** (1997) 341–359
16. Storn, R., Price, K.: Minimizing the real functions of the icec'96 contest by differential evolution. In: IEEE Conference on Evolutionary Computation. (1996) 842–844
17. DiSilvestro, M., Suh, J.K.: A cross-validation of the biphasic poroviscoelastic model of articular cartilage in unconfined compression, indentation, and confined compression. Journal of Biomechanics **34** (2001) 519–525
18. Ilonen, J., Kamarainen, J.K., Lampinen, J.: Differential evolution training algorithm for feed forward neural networks. Neural Processing Letters **17**(1) (2003) 93–105
19. Plagianakos, V.P., Vrahatis, M.N.: Neural network training with constrained integer weights. In Angeline, P., Michalewicz, Z., Schoenauer, M., Yao, X., Zalzala, A., eds.: Proceedings of the Congress of Evolutionary Computation (CEC'99). IEEE Press (1999) 2007–2013
20. Plagianakos, V.P., Vrahatis, M.N.: Training neural networks with 3–bit integer weights. In Banzhaf, W., Daida, J., Eiben, A., Garzon, M., Honavar, V., Jakiela, M., Smith, R., eds.: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'99). Morgan Kaufmann (1999) 910–915

21. Tasoulis, D.K., Pavlidis, N.G., Plagianakos, V.P., Vrahatis, M.N.: Parallel differential evolution. In: IEEE Congress on Evolutionary Computation (CEC 2004), **2** (2004) 2023–2029
22. Plagianakos, V.P., Vrahatis, M.N.: Parallel evolutionary training algorithms for 'hardware-friendly' neural networks. Natural Computing **1** (2002) 307–322
23. Tasoulis, D.K., Plagianakos, V.P., Vrahatis, M.N.: Clustering in evolutionary algorithms to efficiently compute simultaneously local and global minima. In: IEEE Congress on Evolutionary Computation. Volume 2., Edinburgh, UK (2005) 1847–1854
24. John, G., Kohavi, R., Pfleger, K.: Irrelevant features and the subset selection problem. In: International Conference on Machine Learning. (1994) 121–129
25. Aggarwal, C., Wolf, J., Yu, P., Procopiuc, C., Park, J.: Fast algorithms for projected clustering. In: 1999 ACM SIGMOD international conference on Management of data, ACM Press (1999) 61–72
26. Agrawal, R., Gehrke, J., Gunopulos, D., Raghavan, P.: Automatic subspace clustering of high dimensional data for data mining applications. In: 1998 ACM SIGMOD international conference on Management of data, ACM Press (1998) 94–105
27. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning. Springer-Verlag (2001)
28. Fayyad, U., Piatetsky-Shapiro, G., Smyth, P.: Advances in Knowledge Discovery and Data Mining. MIT Press (1996)
29. Aldenderfer, M., Blashfield, R.: Cluster Analysis. Volume 44 of Quantitative Applications in the Social Sciences. SAGE Publications, London (1984)
30. Ramasubramanian, V., Paliwal, K.: Fast *k*-dimensional tree algorithms for nearest neighbor search with application to vector quantization encoding. IEEE Transactions on Signal Processing **40**(3) (1992) 518–531
31. Becker, R., Lago, G.: A global optimization algorithm. In: Proceedings of the 8th Allerton Conference on Circuits and Systems Theory. (1970) 3–12
32. Torn, A., Zilinskas, A.: Global Optimization. Springer-Verlag, Berlin (1989)
33. Alon, U., Barkai, N., Notterman, D., K.Gish, Ybarra, S., Mack, D., Levine, A.: Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide array. Proc. Natl. Acad. Sci. USA **96**(12) (1999) 6745–6750
34. Eisen, M., Spellman, P., Brown, P., Botstein, D.: Cluster analysis and display of genome-wide expression patterns. Proc. Natl. Acad. Sci. USA **95** (1998) 14863–14868
35. Shamir, R., Sharan, R.: Click: A clustering algorithm for gene expression analysis. In: 8th International Conference on Intelligent Systems for Molecular Biology (ISMB 00), AAAI Press (2000)
36. Tavazoie, S., Hughes, J., Campbell, M., Cho, R., Church, G.: Systematic determination of genetic network architecture. Nature Genetics volume **22** (1999) 281–285
37. Tasoulis, D.K., Plagianakos, V.P., Vrahatis, M.N.: Unsupervised clustering in mRNA expression profiles. Computers in Biology and Medicine **36**(10) (2006)
38. Wen, X., Fuhrman, S., Michaels, G., Carr, D., Smith, S., Barker, J., Somogyi, R.: Large-scale temporal gene expression mapping of cns development. Proceedings of the National Academy of Science USA **95** (1998) 334–339
39. Golub, T., Slomin, D., Tamayo, P., Huard, C., Gaasenbeek, M., Mesirov, J., Coller, H., Loh, M., Downing, J., Caligiuri, M., Bloomfield, C., Lander, E.: Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring. Science **286** (1999) 531–537
40. Jain, A., Murty, M., Flynn, P.: Data clustering: a review. ACM Computing Surveys **31**(3) (1999) 264–323

41. Alizadeh, A., et al.: Distinct types of diffuse large b-cell lymphoma identified by gene expression profiling. Nature **403**(6769) (2000) 503–511
42. Perou, C., Jeffrey, S., de Rijn, M.V., Rees, C., Eisen, M., Ross, D., Pergamenschikov, A., Williams, C., Zhu, S., J.C. Lee, D.L., Shalon, D., Brown, P., Botstein, D.: Distinctive gene expression patterns in human mammary epithelial cells and breast cancers. Proc. Natl. Acad. Sci. USA **96** (1999) 9212–9217
43. Xing, E., Karp, R.: Cliff: Clustering of high–dimensional microarray data via iterative feature filtering using normalized cuts. Bioinformatics Discovery Note **1** (2001) 1–9
44. Tamayo, P., Slonim, D., Mesirov, Q., Zhu, J., Kitareewan, S., Dmitrovsky, E., Lander, E., Golub, T.: Interpreting patterns of gene expression with self-organizing maps: Methods and application to hematopoietic differentiation. Proc. Natl. Acad. Sci. USA **96** (1999) 2907–2912
45. Alter, O., Brown, P., Bostein, D.: Singular value decomposition for genome-wide expression data processing and modeling. Proc. Natl. Acad. Sci. USA **97**(18) (2000) 10101–10106
46. Szallasi, Z., Somogyi, R.: Genetic network analysis – the millennium opening version. In: Pacific Symposium of BioComputing Tutorial. (2001)
47. Tasoulis, D.K., Vrahatis, M.N.: Unsupervised distributed clustering. In: Proceedings of the IASTED International Conference on Parallel and Distributed Computing and Networks, Innsbruck, Austria (2004) 347–351
48. Vrahatis, M.N., Boutsinas, B., Alevizos, P., Pavlides, G.: The new *k*-windows algorithm for improving the *k*-means clustering algorithm. Journal of Complexity **18** (2002) 375–391
49. Sander, J., Ester, M., Kriegel, H.P., Xu, X.: Density-based clustering in spatial databases: The algorithm gdbscan and its applications. Data Mining and Knowledge Discovery **2**(2) (1998) 169–194
50. Boley, D.: Principal direction divisive partitioning. Data Mining and Knowledge Discovery **2**(4) (1998) 325–344
51. Bezdek, J.: Pattern Recognition with Fuzzy Objective Function Algorithms. Kluwer Academic Publishers (1981)
52. Fritzke, B.: Growing cell structures a self-organizing network for unsupervised and supervised learning. Neural Netw. **7**(9) (1994) 1441–1460
53. Ankerst, M., Breunig, M.M., Kriegel, H.P., Sander, J.: Optics: Ordering points to identify the clustering structure. In: Proceedings of ACM-SIGMOD International Conference on Management of Data. (1999)
54. Ester, M., Kriegel, H., Sander, J., Xu, X.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: Proceedings of the 2nd Int. Conf. on Knowledge Discovery and Data Mining. (1996) 226–231
55. Procopiuc, C., Jones, M., Agarwal, P., Murali, T.: A Monte Carlo algorithm for fast projective clustering. In: Proc. 2002 ACM SIGMOD, New York, NY, USA, ACM Press (2002) 418–427
56. Berkhin, P.: A survey of clustering data mining techniques. In Kogan, J., Nicholas, C., Teboulle, M., eds.: Grouping Multidimensional Data: Recent Advances in Clustering. Springer, Berlin (2006) 25–72
57. Jain, A.K., Murty, M.N., Flynn, P.J.: Data clustering: a review. ACM Computing Surveys **31**(3) (1999) 264–323
58. Tan, P.N., Steinbach, M., Kumar, V.: Introduction to Data Mining. Pearson Addison-Wesley, Boston (2005)

59. Tasoulis, D.K., Vrahatis, M.N.: Novel approaches to unsupervised clustering through the *k*-windows algorithm. In Sirmakessis, S., ed.: Knowledge Mining. Volume 185 of Studies in Fuzziness and Soft Computing. Springer-Verlag (2005) 51–78
60. Hartigan, J., Wong, M.: A *k*-means clustering algorithm. Applied Statistics **28** (1979) 100–108
61. Zeimpekis, D., Gallopoulos, E.: PDDP(l): Towards a Flexing Principal Direction Divisive Partitioning Clustering Algorithms. In Boley, D., Dhillon, I., Ghosh, J., Kogan, J., eds.: Proc. IEEE ICDM '03 Workshop on Clustering Large Data Sets, Melbourne, Florida (2003) 26–35
62. Singh, D., et al.: Gene expression correlates of clinical prostate cancer behavior. Cancer Cell **1** (2002) 203–209
63. Thomas, J., Olson, J., Tapscott, S., Zhao, L.: An efficient and robust statistical modeling approach to discover differentially expressed genes using genomic expression profiles. Genome Research **11** (2001) 1227–1236
64. Kohonen, T.: Self–Organized Maps. Springer Verlag, New York, Berlin (1997)
65. Ye, J., Li, T., Xiong, T., Janardan, R.: Using uncorrelated discriminant analysis for tissue classification with gene expression data. IEEE/ACM Transactions on Computational Biology and Bioinformatics **1**(4) (2004) 181–190
66. Plagianakos, V.P., Tasoulis, D.K., Vrahatis, M.N.: Hybrid dimension reduction approach for gene expression data classification. In: International Joint Conference on Neural Networks 2005, Post-Conference Workshop on Computational Intelligence Approaches for the Analysis of Bioinformatics Data. (2005)