Active learning for microarray data

D. Vogiatzis \textsuperscript{a,*}, N. Tsapatsoulis \textsuperscript{b}

\textsuperscript{a} University of Cyprus, Department of Computer Science, Kallipoleos 75, CY-1678 Nicosia, Cyprus
\textsuperscript{b} University of Peloponnese, Department of Telecommunications Science and Technology, End of Karaiskaki Street, 22100 Tripoli, Greece

Received 10 April 2006; received in revised form 8 September 2006; accepted 15 March 2007

Abstract

In supervised learning it is assumed that it is straightforward to obtain labeled data. However, in reality labeled data can be scarce or expensive to obtain. Active learning (AL) is a way to deal with the above problem by asking for the labels of the most “informative” data points. We propose an AL method based on a metric of classification confidence computed on a feature subset of the original feature space which pertains especially to the large number of dimensions (i.e. examined genes) of microarray experiments. DNA microarray expression experiments permit the systematic study of the correlation of the expression of thousands of genes.

Feature selection is critical in the algorithm because it enables faster and more robust retraining of the classifier. The approach that is followed for feature selection is a combination of a variance measure and a genetic algorithm.

We have applied the proposed method on DNA microarray data sets with encouraging results. In particular we studied data sets concerning: small round blue cell tumours (four types), Leukemia (two types), lung cancer (two types) and prostate cancer (healthy, unhealthy).

© 2007 Published by Elsevier Inc.

Keywords: Active learning; Microarray data; Classification confidence; Feature selection; Genetic algorithms

1. Introduction to active learning

The idea that a large set of labeled data is available for training a classifier under a supervised regime is often wrong. Data labeling can be a time consuming and expensive process. For instance in a microarray experiment in biology, a large data set is produced which represents the expression of hundreds or thousands genes (i.e. production of RNA) under different experimental conditions (see [1,2] for an overview of microarray experiments) and labeling exhaustively all the experimental samples can be very expensive or simply impossible. An alternative would be to start with a small set of labeled data, which may be easy to obtain, then to train a classifier with supervised learning. At this point a query mechanism pro-actively asks for the
labels of some of the unlabeled data; whence the name active learning. The query, formed by the classifier, implements a strategy to discover the labels of the most “informative” data points.

The concept of active learning is hardly new, an important contribution is in [3], where optimal data selection techniques for feed forward neural networks are discussed. In addition the authors show how the same approach for active data selection is presented in [4]. In particular three different techniques for maximizing the information gain are tested on an interpolation problem. In yet another approach, the geometry of the learning space is derived by computing the Voronoi tessellation, and the queries request the labels of data points at the borders of Voronoi regions [5]. The concept of active learning has also been applied in the context of support vector machines for text classification [6]. The method is based on selecting for labeling, data points that reduce the version space (the hyperplanes that separate the data classes) as much as possible.

In this study we apply the active learning approach in the construction of effective classifiers for microarray data. Active data selection is based on the confidence of classification of the previously launched data. That is, if a new datum is to be used for training then this datum should be the one that could improve the confidence of classification at most. Taking into account that microarray data are characterized by a very large number of dimensions (genes) the metric that is for the selection of the active learning data should be based on an optimal (or sub-optimal) subset of features to reduce complexity and retraining time [7,8]. For this purpose we propose a combination of a variance based feature selection and genetic algorithms which we compare against linear and non-linear principal component analysis.

The rest of the paper is organized as follows: Section 2 formally introduces the concept of active learning. Then in Section 3 the proposed genetic algorithm based feature selection is presented, as well as the principal component analysis (linear and non-linear versions). The experimental setting and results are presented in Section 4. Conclusions and the relevant discussion are provided in Section 5.

2. Active learning

2.1. Classification of microarray data

Let $V = \{v_i | i = 1, \ldots, N\}$ be a set of $N$ feature vectors $v_i = [v_{i,1} \ldots v_{i,d}]$ on $d$-dimensional Euclidean space $\mathcal{R}^d$. Let also $Y = \{y_i | i = 1, \ldots, N\}$ be a set of binary vectors $y_i = [y_{i,1} \ldots y_{i,c}]$ with $y_{i,j}$ be a binary variable.

In all dimensions but one are zero, this notation is influenced by the encoding of the output units in neural networks but the actual implementation can be made more efficient.

Classification involves a random feature vector $\tilde{v}_i$, a random output class vector $\tilde{y}_i$, and a function (classifier) $\Psi : \mathcal{R}^d \rightarrow \{0, 1\}^c$ to serve as a predictor of $\tilde{y}_i$, which means that $\tilde{y}_i$ is to be predicted by $\Psi(\tilde{v}_i)$. Building a classifier refers to the process of creating an estimator function $\hat{\Psi}(V, Y)$ based on the sets $V, Y$. We denote by $V_k$ the subset of $V$ such that for each $\tilde{v}_i \in V_k$, $\Psi(\tilde{v}_i) = \tilde{y}_i$, where $\tilde{y}_i = [y_{i,1} \ldots y_{i,k} \ldots y_{i,c}]$, with $y_{i,j} = 0 \ \forall j \neq k$, and $y_{i,k} = 1$. That is, $V_k$ is the set of vectors in $V$ that belong to class $k$. Provided that each vector belongs to one and only one class the following relations hold:

$$\left( \bigcup_{k=1}^{c} V_k \right) = V, \quad \left( \bigcap_{k=1}^{c} V_k \right) = \emptyset.$$

In the particular case where the aim is to build a classifier for microarray data, $d$ corresponds to the number of genes involved in the experiment, $N$ is the total number of experiments (samples), and $c$ is the number of output classes to which the samples should be categorized. In microarray experiments it is common that $N \ll d$, which makes the construction of an effective classifier a hard task [7,8]. Dimensionality reduction or feature selection techniques are used to get around this problem. Feature selection refers to the estimation of a function $T : \mathcal{R}^d \rightarrow \mathcal{R}$ which transforms the microarray samples to vectors in a lower dimensional space. The most important constraint that is imposed in the estimation of function $T$ is the preservation (or even enhancement) of separability of samples belonging to different classes. It is possible that $T$ maps different microarray samples to the same vector in the lower dimensional space. However, in the current situation this is unlikely to occur, since the dimensionality of the input space $d$ is in the order of thousands whereas there are only a few tens of microarray samples. That is, the $\mathcal{R}^d$ is sparsely populated.

Please cite this article in press as: D. Vogiatzis, N. Tsapatsoulis, Active learning for microarray data, Appl. Math. Comput. (2007), doi:10.1016/j.amc.2007.03.009
Applying the transformation function \( T \) to all vectors of set \( V \) leads to the creation of set \( X = \{ \vec{x}_i \,|\, i = 1, \ldots, N \} \), with \( \vec{x}_i = [x_{i1} \ldots x_{in}] \), \( \vec{x}_i = T(\vec{u}) \). The problem of designing a classifier is, therefore, transformed to the estimation of function \( \hat{\Psi}(X, Y) \) based on the sets \( X, Y \). We present the feature selection method we follow in a subsequent section.

### 2.2. The proposed active learning method

It is presumed that we have a pool of labeled and unlabeled multidimensional data. The purpose is to reduce the error of a classifier (built with the labeled training set), by selectively asking for the labels of the unlabeled data. The algorithm is summarized below:

1. **Pre-process the data set with a view of reducing the dimensions.** Methods employed: linear PCA, kernel PCA, a variance based method and a GA based method.
2. **Train the classifier with the labeled data.** That is, create the estimator \( \hat{\Psi}(X, Y) \) of the classification function \( \Psi \).
3. **Active learning data selection:** The classifier forms a query to ask for the label of an unlabeled datum. Let \( X_k \) be the set of data classified, by the current classifier, to class \( k \). Let also \( \vec{x}_i \in \mathcal{R}^n \) be the \( i \)th vector in \( X_k \). If \( U \) is the set of unlabeled data \( \vec{u}_j \in \mathcal{R}^n \) is the \( j \)th vector in \( U \) then the following criterion is used to form a pool of candidate data \( U_\xi = \{ \vec{u}_{\xi,1} \ldots \vec{u}_{\xi,c} \} \) to be labeled and used for training,

\[
\vec{u}_{\xi,k} = \arg\min_{\vec{u}_j \in U} \left\{ \sum_{i=1}^{N_k} \| \vec{u}_j - \vec{x}_i \|^2 \right\}, \tag{1}
\]

where \( N_k \) is the cardinality of \( X_k \).

Eq. (1) indicates that the candidate data are selected in such a way to be representative of the already created classes and not outliers. In this way the stability of classifier parameters is preserved during retraining while at the same time the information presented to the network is the most informative with respect to the already classified data. Active learning involves the use of only one datum \( \vec{u}_{\xi,a} \) from the pool of candidate data \( U_\xi \). Hence, the final selection is made based on one of the following criteria, depending on whether the aim is to enhance the robustness of the classifier or its discrimination ability:

- **Variation 1:** The next datum \( \vec{u}_{\xi,a} \) to be labeled is selected from class \( a \) (provided that there are non-labeled data for this class), where \( a \) is the class that presents the highest classification error based on the already classified data,

\[
a = \arg\max_k \left\{ \sum_{\vec{x}_i \in X_k} \| \hat{\Psi}(\vec{x}_i) - \vec{y}_i \|^2 \right\}, \tag{2}
\]

where \( \vec{y}_i \) is the expected output for datum \( \vec{x}_i \in X_k \).

- **Variation 2:** For each datum \( \vec{u}_{\xi,a} \in U_\xi \) try to predict its class based on a \( k \) nearest neighbours (\( k \)NN) scheme. If the vast majority of the neighbouring vectors suggest a certain class, then there is no need to use this datum for active learning since the new information it presents to the classifier is low. On the other hand the vector \( \vec{u}_{\xi,a} \in U_\xi \) whose output class prediction, based on the \( k \)NN scheme, presents the highest uncertainty (i.e., its immediate neighbours cannot clearly suggest a class for it) is the most informative (among all data in \( U_\xi \)) for the classifier.

(4) Re-train the classifier with the added datum \( \vec{u}_{\xi,a} \); if the classification error for the specific category (category \( a \)) decreases then repeat Step (2). Otherwise, roll-back to the previous classifier, remove \( \vec{u}_{\xi,a} \) from \( U_\xi \), and repeat Variation 2.

(5) The algorithm terminates, when the user decides that the overall classifier’s performance is good enough, or when there can be no more labels.

To summarise, the rationale of the above algorithm is to detect interesting data, i.e. data essential for reducing the classification error either for increasing robustness (Variation 1) or for increasing classification performance (Variation 2) by selecting data for re-training whose class prediction confidence (based on the already trained classifier) is low.
3. Feature selection

Constructing efficient classifiers from micro-array datasets is prohibited, in most cases, by the fact that these datasets involve a large number of features but only a small number of samples. Therefore, the generalization ability of the learned classifiers is, in general, poor. Feature selection is an obvious choice for dimensionality reduction.

As already stated we combine our algorithm for active learning with feature selection so as to enable faster and more robust retraining of the classifier. Our favorite feature selection method is a combination of a variance based measure and a genetic algorithm. However, for comparison purposes, we include in our study two well known dimensionality reduction methods, the linear principal component analysis (l-PCA) and kernel PCA (k-PCA), a form of non-linear PCA.

At this point we should make a distinction between feature selection and dimensionality reduction. In feature selection the selected features preserve their physical meaning, i.e. we can say, for example, which genes’ (features’) expression help us in the diagnosis of a particular disease (class). In dimensionality reduction we transform our features into another space, taking for example linear or non-linear combinations of them, in which the classification problem could be easier to handle. However, the physical meaning of the transformed features is hard to be identified.

3.1. The proposed feature selection method

Feature selection is often split into two categories: the filter and wrapper methods. In the filter method, features are selected without regard for classifier design, for instance, by choosing features most correlated with the labels or via mutual information. In the wrapper method, features are selected in conjunction with classifier design [9]. When there is a very large number of features, such as in the case of gene expressions on a microarray, the methods are used in conjunction. First, a filtering method is used and then some selection method involving classification is employed on the preliminarily reduced set.

Our feature selection method involves both methods: A feature pre-selection strategy that is based on the maximum variability principle, and an innovative genetic algorithm optimization scheme which involves an radial basic function (RBF) classifier. This combination leads to near-optimal feature sub-spaces with respect to the generalization ability of the classifier. Applying GAs, without a feature pre-selection step, could lead to optimal sub-space selection but convergence to this optimal solution is very slow due to the large number of features involved.

3.1.1. Feature filtering step

Let \( V = \{ \tilde{v}_i | i = 1, \ldots, N \} \) be the initial set of training vectors, with \( \tilde{v}_i \in \mathbb{R}^d \), and \( Y = \{ \tilde{y}_i | i = 1, \ldots, N \} \) be the corresponding set of output binary vectors \( \tilde{y}_i \in \{0, 1\}^c \). The mean vector of set \( V \) is given by

\[
\bar{m} = [m_1, m_2, \ldots, m_d] = \frac{1}{N} \sum_{i=1}^{N} \tilde{v}_i.
\]

We denote by \( V_k \subset V \) the subset of vectors in \( V \) that belong to class \( k \). For each subset \( V_k \) we compute the mean and standard deviation vectors as follows:

\[
\bar{m}^k = [m^k_1, m^k_2, \ldots, m^k_d] = \frac{1}{N_k} \sum_{i=1}^{N_k} \tilde{v}_i^k,
\]

where \( \tilde{v}_i^k \) is the \( i \)-th vector and \( N_k \) the cardinality of \( V_k \).

\[
\sigma^k = [\sigma^k_1, \sigma^k_2, \ldots, \sigma^k_d],
\]

\[
\sigma^k_j = \sqrt{\frac{1}{N_k - 1} \sum_{i=1}^{N_k} (m^k_j - \tilde{v}^k_{ij})^2},
\]

where \( \sigma^k_j \) is the standard deviation of the \( j \)-th feature (gene) computed over set \( V_k \), \( \tilde{v}^k_{ij} \) is the \( j \)-th element of the \( i \)-the vector of \( V_k \) and \( d \) is the number of dimensions.
The inter-class spread of mean class vectors is given by,
\[ s = \frac{s_1 s_2 \cdots s_d}{C_1 \cdots C_1 C_1 \cdots C_1} \]  
\[ s_j = \sqrt{\frac{1}{c-1} \sum_{k=1}^{c} (m_j - m_j^k)^2}, \]  
where \( c \) is the number of classes.

Feature pre-selection is performed by selecting the \( L \) features, \( L \ll d \), that maximize the following inter-class variability criterion:
\[ c_r = \arg \max_i \left( \frac{s_i}{\epsilon + \| m_i \| + \frac{1}{\epsilon} \sum_{k=1}^{c} s_i^k} \right) \]  
where \( \epsilon \) is a small positive constant.

3.1.2. Genetic algorithm

The result of feature pre-selection described in the previous section is the creation of a set of vectors \( \mathbf{B} = \{ \mathbf{b}_i | i = 1, \ldots, N \} \), with \( \mathbf{b}_i \in \mathbb{R}^d \). The next step is to build an RBF classifier \( \mathbf{\Psi}_B \) based on sets \( \mathbf{B}, \mathbf{Y} \). We use the classifier \( \mathbf{\Psi}_B \) in the genetic algorithm optimization scheme to form the optimality criterion. It should be made clear, however, that the chromosomes in the GA algorithm encode input dimensions and not RBF functions. That is, with the use of GAs the set of input vectors \( \mathbf{B} \) is altered (because of the change in input dimensions that are included) while the classifier \( \mathbf{\Psi}_B \) is kept unchanged. Therefore, the classifier \( \mathbf{\Psi}_B \) acts, actually, as a metric. Input values corresponding to excluded input dimensions are zeroed before feeding the RBF neurons. We should note, here, that in a future extension co-optimization based on input dimensions and the \( \mathbf{\Psi}_B \) will be examined. That is, for a particular set \( \mathbf{B} \) created by altering input dimensions, that are taken into account, a different classifier \( \mathbf{\Psi}_B \) will be designed.

Genetic algorithms are adaptive optimization methods that resemble the evolution mechanisms of biological species [10]. Feature selection is one of the areas that GAs present excellent performance. The main advantages of GAs are that they:

- do not require the continuity of parameter space and
- are able to efficiently search over a wide range of parameters/parameter sets.

In a GA, the search begins from a population of possible solutions (in our case binary strings of length \( L \), with ones denoting selection and zeros denoting not selection), and not just one possible solution. Thus, the search will not be trapped in a local optimum, especially if significant diversity exists among the various solutions. The population GAs tends to evolve towards increasingly better regions of the search space through the use of certain randomized processes, called “genetic operators”. Typical genetic operators are the selection, mutation and recombination. The selection process chooses strings with better objective function value and reproduces them more often than their counterparts with worse objective function value. Thus, a new population is formed consisting of the strings that perform better in their environment. The recombination (cross-over) operator allows for the mixing of parental information, which is then passed to their descendants. The initial population is randomly acquired; this means that the first and major degree of diversity is introduced in this stage of the GA. The second and lesser degree of diversity is introduced when the mutation operator acts upon each string of the population. The whole evolution process stops after a predefined maximum number of iterations (generations) is reached.

Once the initial population has been created the process of creating new generations starts and consists, typically, of three stages:

1. A fitness value (measure of “optimality”) of each string in the random population is calculated.
Genetic operators, corresponding to mathematical models of simple laws of nature, like reproduction, crossover and mutation are applied to the population and result in the creation of a new population. The new population replaces the old one.

In our case the coding that has been chosen models the presence (indicated by one in the string) or absence (indicated by zero in the string) of the corresponding input dimension (feature). The fitness function \( F \) that is used is given by

\[
F(\Psi_B) = \frac{1}{N} \sum_{b \in B} \| \Psi_B(b) - \bar{y} \|,
\]

where \( N \) is the cardinality of sets \( B \) and \( Y \).

The objective is to find the binary string that minimises the fitness function \( F(\Psi_B) \). The realisation of the genetic operators reproduction, mutation and crossover is as follows:

**Reproduction.** The fitness function \( F(\Psi_B) \) is used in the classical “roulette” wheel reproduction operator that gives higher probability of reproduction to the strings with better fitness according to the following procedure:

1. An order number, \( q \), is assigned to the population strings. That is \( q \) ranges from 1 to \( P_N \), where \( P_N \) is the size of population.
2. The sum of fitness values (\( F_{\text{sum}} \)) of all strings in the population is calculated.
3. The interval \([0, F_{\text{sum}}]\) is divided into \( P_N \) sub-intervals each of one being \([SF_{q-1}, SF_q]\) where

\[
SF_{q-1} = \sum_{j=1}^{q-1} F_j \quad \text{for } q > 1 \quad \text{and } SF_{q-1} = 0 \quad \text{for } q = 0 \quad \text{or } q = 1.
\]

\[
SF_q = \sum_{j=1}^{q} F_j \quad \text{for every } q,
\]

while \( F_j \) is the value of fitness function for the \( j \)th string.
4. A random real number \( R_0 \) lying in the interval \([0,F_{\text{sum}}]\) is selected.
5. The string having the same order number as the subinterval of \( R_0 \) is selected.
6. Steps (4) and (5) are repeated \( P_N \) times in order to produce the intermediate population to which the other genetic operators will be applied.

**Crossover.** Given two strings of length \( k \) (parents) an integer number is randomly selected. The two strings retain their gene values up to gene \( r \) and interchange the values of the remaining genes creating two new strings (offspring).

**Mutation.** This operator is applied to each gene of a string and it alters its content, with a small probability. The mutation operator is actually a random number that is selected and depending on whether it exceeds a predefined limit it changes the value of a gene.

### 3.2. Linear principal components analysis

Principal Components analysis (PCA) is a method for reducing the dimensionality of the data by finding some linear transformation of the co-ordinate system such that the variance of the data along some new dimensions is suitably small and so those particular new dimensions can be ignored [11]. PCA first finds some dimension along which the data varies as much as possible. Thus it seeks a transformation of the axes \( y = \sum a_x x \), such as the variance of the original data points in direction \( y \) is as large as possible. Since the transformation can be arbitrarily rescaled, it is usual to add the constraining that \( \sum a^2 = 1 \). Having found such a direction, it then finds another direction, orthogonal to the first, along which the data varies as much as possible. Then it finds a third direction orthogonal to the first two and so on. The method is as follows. Suppose we have a set of \( m \) points, each \( n \)-dimensional,

---

Please cite this article in press as: D. Vogiatzis, N. Tsapatsoulis, Active learning for microarray data, Appl. Math. Comput. (2007), doi:10.1016/j.amc.2007.03.009
The feature space can be represented as

\[ \mathbf{p}_1 = [p_{11} p_{12} p_{13} \cdots p_{1n}], \]

\[ \mathbf{p}_2 = [p_{21} p_{22} p_{23} \cdots p_{2n}], \]

\[ \vdots \]

\[ \mathbf{p}_m = [p_{m1} p_{m2} p_{m3} \cdots p_{mn}]. \]

Then the matrix \( \mathbf{C} \) is a symmetric matrix. The \( i \)-th row and \( j \)-th column is \( c_{ij} = \frac{1}{m-1} \sum_{k=1}^{m} (p_{ki} - \mu_i)(p_{kj} - \mu_j) \). Because \( \mathbf{C} \) is symmetric (and real-valued) it has \( n \) eigenvectors and eigenvalues and the eigenvalues corresponding to different eigenvalues are mutually orthogonal to each other. The eigenvectors taken in the order of size of the eigenvalues are the directions we are looking for. Moreover, the size of the eigenvalues shows the proportionate variation of the original points in the direction of the corresponding eigenvector; thus dimensionality reduction can be achieved by ignoring those eigenvector corresponding to suitably small eigenvalues. Let \( \mathbf{E} \) be the matrix, where each row is an eigenvector. It is an \( n \times n \) matrix, the transformed data \( \mathbf{Y} \) can be obtained as \( \mathbf{Y} = \mathbf{M} \cdot \mathbf{E} \).

### 3.2.1. PCA amendment for microarray data

Principal component analysis in the investigation of microarray experiments was introduced in [12]. One of the peculiarities in microarray data sets is that the number of dimensions is usually two orders of magnitude more than the number of data, thus the covariance matrix is \( \mathbf{C} \in \mathbb{R}^{m \times m} \), and \( m \) is usually a few thousands, which renders the diagonalisation problematic. Also there are singularity problems that may occur when computing the eigenvectors of \( \mathbf{C} \), to avoid singularity we would have to have \( m > n^2 \).

The problem of diagonalisation of \( \mathbf{C} \) can be solved through the singular value decomposition of \( \mathbf{M} \). In particular, the eigenvectors of the \( n \times n \) matrix \( \mathbf{M}^T \mathbf{M} \) can be computed through those of the \( m \times m \) \( \mathbf{M}^T \mathbf{M} \), which is computationally simpler since \( m \ll n \). If the rank of matrix \( \mathbf{M} \) is \( r \) with \( r \leq m \) then according to the theory of singular value decomposition matrix \( \mathbf{M} \) can be expressed as

\[ \mathbf{M} = \sum_{k=1}^{r} \sqrt{\lambda_k} \cdot \mathbf{u}_k \cdot \mathbf{v}_k^T, \tag{11} \]

where \( \mathbf{u}_k \) and \( \mathbf{v}_k \) are the eigenvectors of the matrices \( \mathbf{M}^T \mathbf{M} \) and \( \mathbf{M} \mathbf{M}^T \) respectively, and \( \lambda_k \) the eigenvalues of \( \mathbf{M} \mathbf{M}^T \). Multiplying both members of Eq. 11 with \( \mathbf{v}_k \) we obtain an alternative way of computing the eigenvectors \( \mathbf{u}_k \),

\[ \frac{1}{\sqrt{\lambda_k}} \mathbf{M} \cdot \mathbf{v}_k = \mathbf{u}_k. \tag{12} \]

### 3.3. Non linear principal component analysis

The linear version of PCA is well suited for dimensionality reduction in gaussian type distributions, recall that the new dimensions are linear combinations of the old ones. If this condition fails, for instance in a hyperbolic – type data distribution, then PCA could not produce meaningful principal components. The Kernel principal components analysis (k-PCA), is a non-linear kernel method for extraction of non-linear principal components from a data set in which the \( n \)-dimensional input vectors are non-linearly mapped from their original space \( \mathbb{R}^n \) to a high dimensional feature space \( F \) where linear PCA is performed [13]. The mapping to the feature space can be represent as

\[ \Phi : \mathbb{R}^n \rightarrow F. \tag{13} \]
Again, we make the assumption that the data in feature space are centered, i.e. $\sum_{i=1}^{M} \Phi(\tilde{p}_i)$. Now, the covariance matrix in feature space $F$ is

$$\tilde{\mathbf{C}} = \frac{1}{m-1} \sum_{j=1}^{m} \Phi(\tilde{p}_j) \Phi(\tilde{p}_j)^T,$$

(14)

which can be diagonalised with non-negative eigenvalues satisfying,

$$\lambda \mathbf{V} = \tilde{\mathbf{C}} \mathbf{V}.$$

(15)

It is possible to compute dot-products without explicitly mapping into the high dimensional feature space, thus using the following form of dot products we employ the kernel trick,

$$K(\tilde{x}, \tilde{y}) = \phi(\tilde{x}) \cdot \phi(\tilde{y}).$$

(16)

Among the widely used kernels there is the Gaussian kernel

$$K(\tilde{x}, \tilde{y}) = \exp \left(\frac{-||\tilde{x} - \tilde{y}||^2}{2\sigma^2}\right),$$

(17)

and the polynomial kernel, $k(\tilde{x}, \tilde{y}) = (\tilde{x} \cdot \tilde{y})^d$. Note that for $d = 1$, we obtain a kernel that covers linear PCA as a special case. Kernel based PCA has been widely used as a pre-processing method in the field of data mining, as well as in study of microarray data [14].

4. Experiments

The experiments conducted aimed to compare the reduction of the classification error on active learning under different methods of feature selection (and dimensionality reduction). Feature selection, is a form of pre-processing a data set. After this step, the active learning algorithm progressively asks for the labels of all data in the pool. In each step the selected datum is the one which minimises: $\|\tilde{u}_i - \tilde{x}_j\|_2 \forall \xi, i$, where $\tilde{u}$ is an unlabeled datum from a pool, $\tilde{x}$ is a labeled datum from the training set, $k$ denotes the category where the classifier has the highest Root Mean Square Error (RMSE). Thus we evaluated variation 1, as described in Section 2. As performance measure of the active learning algorithm we employed the average cumulative error per query (CEQ) which is defined as

$$\text{CEQ} = \frac{\sum_{j=1}^{U_k} \text{RMSE}(j)}{U_k},$$

(18)

where $U_k$ is the cardinality of the pool data set, $\text{RMSE}(j)$ is the root mean square error of the classifier at step $j$, where $j$ unlabeled data have been introduced in the training set.

4.1. Expression data sets

Four different labeled data sets from microarray experiments were used. The first data set was obtained from “The Microarray Project cDNA Library” http://research.nhgri.nih.gov/microarray/Supplement/. The rest of the data sets were obtained from the Gene Expression Datasets collection http://sdmc.lit.org.sg/GEDatasets. The first data set is about small round blue cell tumours (SRBCT), investigated with cDNA microarrays containing 2308 genes, over a series of 63 experiments. The 63 samples included tumour biopsy material and cell lines from four different types: 23 Ewing’s sarcoma (EWS), 20 rhabdomyo sarcoma (RMS), 12 neuroblastoma (NB) and 8 Burkitt’s lymphoma (BL). There are also available 20 samples (6 EWS, 3 BL, 6 NB and 5 RMS) for testing [15]. The provenance of the second data set is also from oligonucleotide microarrays, with a view of distinguishing between acute lymphoblastics leukemia (ALL) and acute myeloid leukemia (AML). The training data set consisted of 38 bone marrow samples (27 ALL, 11 AML) from 7130 human genes. The test data set consisted of 34 samples (20 ALL, 14 AML) [16]. The third data set also stems from a microarray experiment and consists of lung malignant pleural mesothelioma (MPM) and adenocarcinoma (ADCA) samples [17]. The training set consists of 32 samples (16 MPM and 16 ADCA)
each class from 12,534 human genes. The test set consists of 149 samples (15 MPM and 134 ADCA). The fourth microarray data set concerns the classification of tumour vs. non-tumour samples from 12,600 genes in a study of prostate cancer. The training set consists of 102 samples (52 tumour and 50 healthy) while the testing set consists of 34 samples (25 tumour and 9 healthy) [18].

4.2. Experimental set up

First, the data sets were normalised in the interval [0, 1]. Then each data set was split into training, pool and testing subsets. The first is used for training the classifier. Under the active learning regime a query asks for the labels of specific data from the pool. The best datum (according to the query asked) receives a label and it is subsequently integrated into the training data set. The testing subset is used for independent control (see Table 1).

We have compared four different methods of feature selection against the original features on active learning across four microarray data sets. In particular the linear PCA (l-PCA), the kernel PCA (k-PCA), a variance based method, and the proposed genetic algorithm feature selection methods were investigated. In linear PCA the eigenvectors that corresponded to at least 1% of the variance of the data were retained. In kernel PCA a gaussian kernel was used. In the genetic algorithm, a population of 30 strings, with a mutation probability of 0.7 was employed.

Table 1
Data sets characteristics

<table>
<thead>
<tr>
<th>Data set</th>
<th>Number of features (genes)</th>
<th>Number of samples</th>
<th>Number of training</th>
<th>Number of pool</th>
<th>Number of test</th>
<th>Classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRBC</td>
<td>2308</td>
<td>83</td>
<td>13</td>
<td>52</td>
<td>20</td>
<td>4</td>
</tr>
<tr>
<td>Leukemia</td>
<td>7130</td>
<td>72</td>
<td>19</td>
<td>19</td>
<td>117</td>
<td>2</td>
</tr>
<tr>
<td>Lung</td>
<td>12534</td>
<td>181</td>
<td>16</td>
<td>16</td>
<td>117</td>
<td>2</td>
</tr>
<tr>
<td>Prostate</td>
<td>12600</td>
<td>136</td>
<td>52</td>
<td>52</td>
<td>34</td>
<td>2</td>
</tr>
</tbody>
</table>

Fig. 1. RMSE on data sets. (—): all features (no feature selection); (○): k-PCA, (×–×): l-PCA; (⊙): GA; (×–×): variance method.

Please cite this article in press as: D. Vogiatzis, N. Tsapatsoulis, Active learning for microarray data, Appl. Math. Comput. (2007), doi:10.1016/j.ijar.2007.03.009
A support vector machine (SVM), with a polynomial kernel of degree 3 was used as the classifier of the active learning scheme [19]. SVMs have an excellent theoretical background, they achieve very good generalization results when compared to neural networks and they can handle data sets of the dimensionality (number of features) that is relevant to the current work. For instance, the smallest data set has 2308 dimensions and training a multilayered perceptron would take a prohibitively long time. Training on all data sets always resulted in learning 100% of the training sample.

The results are reported in Figure 1 where we depict the testing set error (RMSE) against the number of queries. The results are averages over 100 experiments. In each experiment the whole data set is shuffled while the number of data in the training, pool and test sets remains the same. In Table 2 the average cumulative error per query is reported for each data set. The last column of Table 2 records the error for all features. Table 3 reports the number of features that were selected, by the various methods that were evaluated.

All experiments were carried out on the Matlab 6.5 platform, with code developed by the authors along with the OSU SVM classifier http://sourceforge.net/projects/svm/ and the STPR toolboxes http://cmp.felk.cvut.cz/xfrancv/stprtool/.

5. Discussion and conclusions

Microarray experiments belong to a class of high throughput techniques that became recently available in biology. They permit the automatic study of the expression pattern of thousands of nucleotide sequences under different experimental conditions. However the characterisation of experiments is based on human expertise, and thus it is expensive, time consuming and occasionally error prone. In active learning the classifier decides which of the data (microarray experiments) are most informative, and asks for their label instead of blindly relying on a data set as is the case in supervised learning.

Feature selection as a pre-processing step has positive consequences in subsequent steps of the analysis and in particular in active learning. Training can be much faster, often more accurate and most important for our study: the error is reduced more rapidly as more data labels are acquired. Thus a tiny percentage of the original features (or of a combination thereof) is sufficient to accurately classify the data. For instance, in the Prostate cancer data set, 34 (out of 12,600) genes are sufficient to distinguish between tumour and non-tumour samples. The best results we have achieved regarding the feature selection (or dimensionality reduction) were 0.91%, 0.22%, 0.33% and 0.27% for the SRBCT, Leukemia, Lung and Prostate data sets respectively. Similar percentages for feature selection have been obtained in supervised learning for microarray data in other studies [15].
It should be noted that the features selected by the variance and the GA methods, are subsets of the original features. In contrast the l-PCA (k-PCA) selected features are linear (non-linear) combinations of the original features. This can be very important, in applications where the individual genes that are most characteristic for each class have to be identified.

When comparing different feature selection methods with regard to the cumulative error per query (CEQ) under the active learning scheme, the GA method outperformed all other methods in the Leukemia and Lung data sets. In the other two data sets k-PCA or variance are better but the number features discovered by GA is less than the number of features discovered by the other methods. We should note however, that the variance method has been evaluated independently, but it is also the first step in the GA method. Moreover, overall the number of features discovered by GAs is less than that discovered by other methods (again the only exception occurs with k-PCA on the Leukemia set, but the CEQ in k-PCA pre-processing is worse).

The non-linear version of PCA (k-PCA) projects non-linearly (in our case with a Gaussian kernel) the data to a feature space $F$ and then performs l-PCA on $F$. k-PCA is always better than l-PCA either with regard to the number of features discovered or to the CEQ. Thus l-PCA seems to be the worst method for pre-processing data for active learning. This is probably explainable by the nature of the data distribution. As it has been mentioned, l-PCA (linear PCA) is based on the assumption that there is an underlying gaussian distribution, otherwise the results are poor. The low performance of l-PCA seems to be corroborated by a similar study, which concerned microarray data clustering [20]. The relevant results (albeit on different datasets) indicated that independent component analysis (ICA) is superior when compared with PCA. ICA finds statistically independent components, whereas PCA discovers uncorrelated components. Independence is a stronger condition than uncorrelatedness, and are only equivalent in gaussian distributions. Furthermore, in the same study non-linear ICA had superior performance to linear ICA, which is further evidence that linear techniques are not well suited to microarray data.

References


