Non-Negative Least Squares Methods for the Classification of High Dimensional Biological Data

Article *in* IEEE/ACM transactions on computational biology and bioinformatics / IEEE, ACM · April 2013

DOI: 10.1109/TCBB.2013.30 · Source: PubMed

2 authors:

Yifeng Li
National Research Council Canada
48 PUBLICATIONS **364** CITATIONS
SEE PROFILE

Alioune Ngom
University of Windsor
133 PUBLICATIONS **757** CITATIONS
SEE PROFILE

Non-Negative Least-Squares Methods for the Classification of High Dimensional Biological Data

Yifeng Li, *Student Member, IEEE,* Alioune Ngom, *Member, IEEE*

Abstract—Microarray data can be used to detect diseases and predict responses to therapies through classification models. However, the high dimensionality and low sample size of such data results in many computational problems such as reduced prediction accuracy and slow classification speed. In this paper, we propose a novel family of non-negative-least-squares classifiers for high dimensional microarray gene expression and comparative genomic hybridization data. Our approaches are based on combining the advantages of using local learning, transductive learning and ensemble learning, for better prediction performance. To study the performances of our methods, we performed computational experiments on seventeen well-known data sets with diverse characteristics. We have also performed statistical comparisons with many classification techniques including the well-performing SVM approach and two related but recent methods proposed in literature. Experimental results show that our approaches are faster and achieve generally a better prediction performance over compared methods.

Index Terms—Medicine, classifier design and evaluation, algorithms.

1 Introduction

Microarray gene-sample data [1] comprise the expression levels of genes measured at given experimental conditions and for a given number of samples of different classes. They has been extensively studied to detect diseases associated to gene mutations and genomic disorder, to predict responses to drug treatments, or to discover genomic etiology using supervised or unsupervised machine learning methods. Alternatively, *array-based comparative genomic hybridization* (aCGH) techniques [2] directly measure the DNA copy numbers, which in turn can be used to analyze biological samples. It has been attracting some researchers to combine gene expression and DNA copy numbers in cancer study.

Due to the expensive cost of microarray experiments, the number of samples is usually in the order of tens or hundreds while the number of genes (or probe sets) is in the order of thousands or even tens of thousands. Such low sample size and high dimensionality of such data lead to inaccurate prediction results and slow classification performance. Classifiers such as *Gaussian-distribution-based Bayesian* classifier, and *linear discriminant analysis* (LDA) may fail because the correlation or covariance matrix may be singular due to the high dimensionality of the data. *Artificial neural network* (ANN) will contain a large number of parameters, and hence, leading to overfitting issues and slow learning time. Some probabilistic graphical models such *Bayesian network* classifier will have difficulty to learn the reliable dependencies since the data are sparse, due to the limited number of samples for estimating the parameters. Another issue with microarray data arises from the inclusion of technical or artificial noise as well as the existence of irrelevant or redundant information within the data. *Tree-based* classifiers, for example, are sensitive to such problem and therefore may generalize poorly. Methods ranging from feature selection or extraction, outlier detection, and noise removal, to the design of new classifiers, to name a few, have been devised to improve a classifier's performance on such data. Without loss of generality, we represent a gene-sample data set by a matrix with the samples on the columns.

The *weighted random subspace* method was proposed in [3] for high dimensional data. This method is computationally expensive since it require thousands classifiers or more to learn a model with acceptable performance. Furthermore, the constrained quadratic programming devised to select among the given classifiers is too large to optimize efficiently. A hierarchical classifier (*PCAHIER*) was introduced by [4] to classify short prokaryotic metagenomic DNA fragments. *Principal component analysis* (PCA) is applied as dimension reduction method and then four layers of LDA classifiers are applied, based on the fact that a fragment has four hierarchical class labels. The advantage is that PCAHIER makes use of the hierarchical structure of the class labels. PCAHIER approach is, however, computationally costly since (1) many local classifiers are learned at each level, and (2) binary LDA classifiers are used in an one-versus-all strategy for learning multiple-class data. [5] used *Partial least squares* (PLS) approaches in order to reduce the dimension of microarray data. [6] proposed *Sparse PLS discriminant analysis* (SPLSDA) and *Sparse generalized PLS* (SGPLS) to simultaneously extract and

• *Y. Li and A. Ngom are with the School of Computer Science, University of Windsor, Ontario, Canada N9B 3P4.*
E-mail: {li11112c,angom}@uwindsor.ca

select features for classifying microarray data. However, it is time-consuming to estimate the parameters of these PLS models. A class of sparse approaches, the *non-negative matrix factorization* (NMF) method and its variants [7], have been applied to cluster either the genes or the samples given a microarray data [8] [9]. Studies have shown that NMF clustering approaches performs better than k-mean clustering methods [10]. NMF techniques are easy to implement, and are very fast when using alternative non-negative least squares optimization method (see Section 3.4). In addition, unlike the other clustering methods, NMF can be viewed as both crisp and fuzzy clustering method. NMF has also been used as dimension reduction model for classifying microarray data [11].

In this paper, we propose a family of fast *non-negative least squares* (NNLS) methods to classify high dimensional microarray data. Our methods combine the advantages of NMF, local learning, transductive learning, and ensemble learning. We then analyze in terms of predictive accuracy and learning time, the performance of our methods by applying them on a number of low-sample-size high dimensional microarray data sets. Our techniques can also be used on many types of data other than microarray data.

We give a brief review of related approaches in Section 2. The NNLS classifier coupled with two sparsity interpreters is given in Section 3.1. The local and transductive NNLS classifier is presented in Section 3.2. In Section 3.3, we introduce the repetitive local NNLS classifier. The optimization algorithms are given in Section 3.4. In Section 3.5 we reveal that the optimization and prediction are dimension-free, and our methods can be extended to kernel versions. The experimental comparison with the latest classification techniques for high dimensional biological data and local classifiers is reported in Section 4. Finally, we draw our conclusions and suggest avenues for further research.

2 Related Work and Insights

In this section, we review the concepts of sample selection, local learning, transductive learning, as well as ensemble learning. We also give further insights about them which contribute to our idea presented in Section 3.

2.1 Sample Selection and Local Learning

Learning on the whole training data might be too complicated due to the unknown distribution of data and the presence of noise or outliers. Therefore *sample selection* has been studied to select a meaningful portion from the whole sample set. Given labeled training data, inductive learning aims to learn a “ubiquitous” model from these training data and make prediction on future unknown samples. Readers are refereed to [12] for a detailed discussion as well as its interaction with feature selection. In an inductive learning, the idea of sample selection is that some samples are more important for (or, relevant to) the classification problem at hand than other samples. For example, *Support vector machine* (SVM) has been utilized to keep only the samples corresponding to the support vectors (that is, the hard samples) and then select the features given only these hard samples [13]. Alternatively, sample weighting scheme is used in ensemble learning [14] for selecting important samples. Unsupervised learning can also be performed to explore portions of the sample space. For example, *kd-tree* has been used to cluster the training data, such that feature selection is subsequently applied on subset of samples from the resulting clusters [15].

Similar with sample selection, *local learning* [16] [17] provides a much clearer confidence of learning on parts of data. In addition to the reason for sample selection, local learning is a divide-and-conquer scheme when the size of the data is too large to compute a model from. Discrete or smooth kernels are often used to select local samples. Discrete kernels select k samples within the kernel width. Smooth kernels, for instance *radial basis function* (RBF), give more weights to the neighbors in the vicinity of a sample. In fact, *k-nearest neighbors* (k-NN) and *decision tree* are the first-generation of local learning methods [18], and SVM using a RBF kernel is of the second generation of local learning approaches. In [16], a linear classifier is trained using only k nearest neighbors. *Local SVM* (LSVM) combines both local and transductive learning ideas [19] in which the similarity between an unknown sample and the training samples are computed by using RBF, so that the training phase pays more attention to samples closer to the unknown sample.

Many local learning methods have been devised following its introduction in [16]; see for example [20], [21], and [22]. The partitioning scheme is a crucial step in local learning, in which a recursive tree or kernel is used to cluster the samples. Generally (but not strictly) speaking, if two data points are mutually closest to each other then they are expected to fall into the same cluster. *Profile SVM* (PSVM) [19] uses a supervised grouping method to split the data such that each subgroup equally includes data from both classes before training a linear SVM. Any clustering method can be used in the sample partitioning phase.

2.2 Transductive Learning

Local learners such as the k-NN based algorithm [16], LSVM [19], and PSVM [19], are *transductive* methods since unlabeled samples are involved in the partitioning phase before learning a model in each local region. Take the k-NN based algorithm for example, since each unlabeled but unknown sample is used to select a subset of the training set, the subsequent linear classifiers will makes use of the information contained in the unlabeled samples in order to learn a model. Given a set of labeled data $\boldsymbol{X}$ and a set of unlabeled data $\boldsymbol{S}$, transductive learning learns over the union of $\boldsymbol{X}$ and $\boldsymbol{S}$ in order to predict the correct class labels of the unlabeled samples $\boldsymbol{S}$ [23]. In inductive learning, supervised algorithm is performed only on the labeled data $\boldsymbol{X}$ in order to predicts the class labels of $\boldsymbol{S}$ and any unknown future sample. Transductive learning uses both labeled data $\boldsymbol{X}$ and unlabeled data $\boldsymbol{S}$, and it is also called *semi-supervised learning* [24] [25] since it makes use of both supervised and unsupervised methods.

The advantage of transductive learning is that unlabeled samples contribute to the classification, therefore more information is utilized. This is beneficial for increasing the prediction accuracy, particularly when there are few labeled training data and many unlabeled data available. The objective of transductive learning is different from that of inductive learning. Transductive learning aims to only classify the unlabeled data $\boldsymbol{S}$ and does not predict unlabeled samples not included in $\boldsymbol{S}$. Inductive learning learns a general model from $\boldsymbol{X}$ which is then used to predict the classes of $\boldsymbol{S}$ and any future unknown sample. One shortcoming of transductive learning is that it is time-consuming since it is performed again for every subset of never before seen unlabeled sample; unless an online update method can be devised to update the learned model quickly. Since inductive learning obtains a general model on the labeled training data, it is thus convenient for predicting any unknown samples. On the other hand, if the task is only to predict the available unlabeled samples once, and if the mixture of labeled and unlabeled data can be clustered well, transductive learning should be better than inductive learning. However, another drawback of transductive learning is that the incorrect prediction of the unlabeled data or ill clustering may propagate and then amplify mistakes in the subsequent prediction phase.

We can relax the single-class constraint in transductive learning, by applying state-of-the-art clustering algorithms, and then allow each resulting cluster to contain samples from more than one classes. This is because it is possible that the distributions of some classes may overlap while still being separable. However, clusters with single class label are still the most welcome. This relaxation can be viewed as a median between the partitioning and the agglomerative approaches. Since a cluster may have more than one unique class label, a classifier must be used in each such cluster.

2.3 Ensemble Learning

Local learning is also often combined with *ensemble learning* [26] which is composed of a committee of classifiers and a voting scheme for deciding the final prediction. In [3], each classifier is trained in a random subspace, and the class label of an unknown sample is decided by the weighted vote of many classifiers. As mentioned above, local learning is a divide-and-conquer scheme, and essentially ensemble learning often provides a necessary summary after local learning. *Mixtures of local experts* (MLE) proposed in [27] applies some neural networks, as local experts, each learning on different subregion of the learning space, and then a gating network decides which expert to switch to given a unknown sample. This gating network actually implicitly plays the role of a crisp kernel. Since competitive learning is used in MLE, each expert is actually expected to learn on a subgroup of the complete training data.

3 Methods

In this section, we shall describe our non-negative least-squares (NNLS), local NNLS, and repetitive local NNLS approaches sequentially, and then discuss the related optimization algorithms. Finally we devise the kernel extensions of our approaches.

3.1 NNLS Classifier

Sound performance of NMF has been reported in image processing and bioinformatics as clustering and dimension reduction methods [7] [11]. Suppose the training data and the unlabeled but unknown data are represented by $\boldsymbol{X} \in \mathbb{R}^{m\times n}$ and $\boldsymbol{S} \in \mathbb{R}^{m\times p}$, respectively, where each column is a sample, and each row corresponds to a feature. The class labels of these n training samples are in the vector $\boldsymbol{c} \in \{0, 1, \cdots, C-1\}^n$ where C is the number of classes. We can formulate NMF in the following equation

$$\min_{\boldsymbol{A},\boldsymbol{Y}} \frac{1}{2}\|\boldsymbol{X} - \boldsymbol{A}\boldsymbol{Y}\|_F^2, \text{ s.t. } \boldsymbol{A}, \boldsymbol{Y} \geq 0, \tag{1}$$

where, $\boldsymbol{X}$ must be non-negative and $\|\cdot\|_F^2$ is the *Frobenius* norm. Semi-NMF is expressed in the equation below

$$\min_{\boldsymbol{A},\boldsymbol{Y}} \frac{1}{2}\|\boldsymbol{X} - \boldsymbol{A}\boldsymbol{Y}\|_F^2, \text{ s.t. } \boldsymbol{Y} \geq 0, \tag{2}$$

where negative values are allowed in $\boldsymbol{X}$ and $\boldsymbol{A}$. Based on the fact that each sample (a column of $\boldsymbol{X}$ or $\boldsymbol{S}$) is approximated by a non-negative (and probably sparse) superposition of basis vectors (columns of $\boldsymbol{A}$), that is $\boldsymbol{x}_i \approx \boldsymbol{A}\boldsymbol{y}_i$, our idea is that, first, the basis matrix is replaced with the training data, and an unknown sample is therefore approximated by a non-negative and sparse combination of the training samples, that is $\boldsymbol{s}_i \approx \boldsymbol{X}\boldsymbol{y}_i$; then a sparsity interpreter is used to predict the class label through interpreting the sparse coefficient vector $\boldsymbol{y}_i$.

```
Algorithm 1 NNLS Classifier
Input: $X_{m\times n}$: training set with $m$ features and $n$ samples
       $c$: class labels of the training samples
       $S_{m\times p}$: $p$ unknown samples without labels
Output: $p$: predicted class labels of the $p$ unknown samples

  1. Solve the NNLS optimization problem:
       $\min_{Y} \frac{1}{2}\|S - XY\|_F^2, \text{ s.t. } Y \geq 0;$   (3)
  2. Predict the class label using a sparsity interpreter:
       $p_i \leftarrow MAX(y_i)$, or
       $p_i \leftarrow NS(y_i)$;
  3. return $p$.
```

Our approach is described in Algorithm 1. The non-negative coding step is essentially a multiple-columns non-negative least squares problem. We therefore call our method *non-negative least squares (NNLS)* classification approach. Its optimization is introduced in Section 3.4. Now, we give an example to explain how this step works in Equation (4) where $\boldsymbol{X} \in \mathbb{R}^{8\times 6}$ and $\boldsymbol{S} \in \mathbb{R}^{8\times 4}$, and $\boldsymbol{c} = [0, 0, 0, 1, 1, 1]^{\mathrm{T}}$. This example is represented in the form of $\boldsymbol{S} \approx \boldsymbol{X}\boldsymbol{Y}$. The first two unknown samples are actually from class 0, and the last two from class 1. The training and unknown samples contains a large portion of Gaussian noise. The fifth and sixth features are redundant features. It can be seen that the coefficient matrix is non-negative and sparse.

$$\begin{pmatrix} 2.0702 & 1.4154 & 0.2012 & 3.4851 \\ -1.4899 & -0.8002 & -0.5951 & 1.3299 \\ 4.3718 & 6.9345 & -0.9424 & 0.8629 \\ 4.1365 & 2.9243 & -1.7334 & -0.6178 \\ -0.2267 & 0.1533 & 2.3851 & -0.2490 \\ 0.2735 & 0.1955 & -0.0560 & -0.9819 \\ 4.8281 & 6.2445 & 1.8495 & 1.1942 \\ -1.9287 & -2.8046 & 2.4307 & 3.2563 \end{pmatrix} \approx$$

$$\begin{pmatrix} 2.7206 & 2.1430 & 0.1428 & 2.0088 & -0.0913 & 1.9284 \\ -1.9554 & 0.3454 & -1.9641 & -0.4381 & 1.2974 & 0.2921 \\ 5.5972 & 7.0132 & 6.7965 & -1.4534 & 0.1980 & 0.6267 \\ 1.5594 & 3.7783 & 2.7041 & -3.9547 & -2.1347 & -2.2222 \\ 0.1129 & 0.1157 & 0.1575 & 1.7358 & 0.9402 & 1.0519 \\ -0.9253 & -0.1314 & -0.7121 & -0.4539 & -0.4043 & -0.1376 \\ 3.8031 & 4.9755 & 4.3159 & 0.3239 & 0.8172 & 2.5853 \\ -1.7990 & -2.6933 & -3.7117 & 1.7895 & 2.8704 & 5.2992 \end{pmatrix}$$

$$\begin{pmatrix} 0.2500 & 0 & 0 & 0 \\ 0.5863 & 0.5568 & 0 & 0.1091 \\ 0.0230 & 0.5175 & 0.0336 & 0 \\ 0 & 0.0286 & 0.3624 & 0.0572 \\ 0 & 0 & 0.1586 & 0 \\ 0 & 0.1551 & 0.2586 & 0.6216 \end{pmatrix} \quad (4)$$

The prediction step requires a sparse interpreter on the sparse coefficient vector in order to predict the class of a unknown sample. We give two interpreters in this paper. The first one is called the *MAX* rule which is inspired by the usual way of using NMF as a clustering method. For a new sample, it selects the maximum coefficient in the coefficient vector, and then assign the class label of the corresponding training sample to this new sample. Essentially, this rule is equivalent to apply 1-NN classifier in the column space of the training samples. In this space, the representation of the training data is actually an identity matrix. In the example above, we can use this rule to predict all the classes of the unknown samples correctly. In much noisy case, incorrect decision might be made by the MAX rule. The second interpreter is called the *nearest subspace* (NS) rule which is proposed in [28] to interpret sparse coding in face recognition. NS rule takes advantage of the discriminating property of sparse coefficients, and is generally more robust to noise than the MAX rule. Their experimental comparison is shown in Section 4. Suppose there are C classes with labels $0, 1, \cdots, C-1$. For a given new sample $\boldsymbol{s}$, after obtaining its coefficient vector $\boldsymbol{y}$, the regression residual corresponding to class i is computed as

$$r_i(\boldsymbol{s}) = \frac{1}{2}\|\boldsymbol{s} - \boldsymbol{X}\delta_i(\boldsymbol{y})\|_2^2,$$

where $\boldsymbol{\delta}_i(\boldsymbol{s}) : \mathbb{R}^n \to \mathbb{R}^n$ returns the coefficients for class i. Its j-th element is defined by

$$(\delta_i(\boldsymbol{s}))_j = \begin{cases} y_j & \text{if } \boldsymbol{x}_j \text{ in class } i, \\ 0 & \text{otherwise.} \end{cases}$$

Finally, a class label q is assigned to $\boldsymbol{s}$, where

$$q = \min_{0 \le i \le C-1} r_i(\boldsymbol{s}).$$

For instance, the residuals of $\boldsymbol{S}$ in Equation (4) is shown in Table 1. According to the residuals, the first two samples are predicted to be in Class 0, and the last two samples are in Class 1. Hereafter, we denote the NNLS approach using MAX rule as *NNLS-MAX*, and that using NS rule as *NNLS-NS*.

TABLE 1
The Residuals of S in (4)

Sample	Class 0	Class 1
$\boldsymbol{s}_1$	2.5370	8.3597
$\boldsymbol{s}_2$	1.7087	10.4016
$\boldsymbol{s}_3$	4.4854	2.2324
$\boldsymbol{s}_4$	5.2291	3.0576

From the above example we can see that the NNLS classifier is suitable for classifying data where the number of features is greater than the number of samples, and is robust to noise and redundance. Also, our approach is a non-parametric multi-class classifier, unlike SVM which needs to find the optimal hyperplane and the support vectors between two classes. Furthermore, our NNLS method is essentially a local learner since only the training samples with non-zero coefficients contribute to the prediction. However, it is very different from k-NN. k-NN obtains its k nearest neighbors based on a distance metric, whereas NNLS finds its local regressors based on the contribution of a sample to the reduction of the current regression residual. Suppose $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ are very similar training samples and both are the closest to a unknown sample $\boldsymbol{s}$ in Euclidean space. k-NN ($k \geq 2$) will choose both, while NNLS only selects one of them, since including another one additionally does not contribute to reducing the residual. Instead, NNLS would further search other samples according to the residual. Additionally, k has to be pre-specified before running k-NN, but NNLS adaptively finds its local regressors according to the sparse coefficient vector.

3.2 Local NNLS Classifier

Interesting but complicated patterns may be hidden within microarray data. Local learners may approximate this complexity by learning in local regions. In order to emphasize the advantages of local learning, we propose a generalized local version of NNLS which is described in Algorithm 2. Please note that the NNLS classifier given in the previous section is a special case of this approach, because its cluster number can be viewed as one. Given $\boldsymbol{X}$ containing training samples in columns, the classification task is to categorize the unknown samples $\boldsymbol{S}$. Our idea of solving this task is the following. First, the union of $\boldsymbol{X}$ and $\boldsymbol{S}$ is clustered by NMF (if it is non-negative) or semi-NMF (if it is of mixed signs). Then, for each cluster, if there are no unknown samples in this cluster, simply skip this iteration; if this cluster does not contain any training sample, a NNLS classifier is applied over the whole training set $\boldsymbol{X}$ to classify the unknown samples in this cluster; if there are both training samples and unknown samples in this cluster, then a NNLS classifier learns on these training samples and predict the classes of these unknown samples. This approach is also transductive as unlabeled but unknown samples are involved in the partitioning phase. This method is named *local NNLS* (LNNLS) classification approach. We denote it as LNNLS-MAX and LNNLS-NS for MAX and NS rules, respectively.

The main differences between this approach and PSVM are that i) PSVM has to conduct a supervised partitioning as the learning of a linear SVM in a subgroup require (balanced) samples from both classes, while LNNLS uses unsupervised clustering which neither deteriorate the natural structure of the data, nor require classifiers for some pure clusters; and ii) PSVM is a binary approach and hence computationally costly in the case of many classes, whereas LNNLS is appropriate for multi-class data.

Due to the relatively small sample size of microarray data, some readers might be concerned that learning in local regions may worsen the learning performance since the size of a cluster is smaller. We address this concern with the following three points. First, the optimization of NMF and NNLS needs only the inner products of the samples rather than the original high dimensional vectors. and thus, the classification is in fact dimension-free. Replacing the inner products with kernel matrices, we can obtain kernel NMF and kernel NNLS, though we have only investigated linear kernel so far. Please see Section 3.5 for details on this point. Second, our LNNLS method does not simply cluster and classify. Applying a classifier in a non-homogeneous cluster (i.e., a cluster containing labeled samples from different classes) is the final step of prediction. For homogeneous clusters, that is when all labeled samples are from the same class c, we can directly assign the class c as the label of the unlabeled samples contained in such clusters. For clusters in which all samples are unlabeled, the all training samples will be required for classifying the unlabeled samples. Third, whether the sample size is sufficient or not is a statistical question, it also depends on the biological and experimental complexity (please see [29] and [30] and references therein). Despite their small sample size, microarray data may still contain enough information needed for discovering the hidden patterns with a statistical learning model. In Section 4.3 we will empirically show that the minimum number of training samples required by NNLS to obtain significant accuracy is generally very small on microarray data.

3.3 Repetitive LNNLS Classifier

The performance of the LNNLS classifier depends on the clustering algorithm (NMF or semi-NMF). Due to the non-convexity of NMF optimization and to the random initializations, different executions of NMF may result in different factors and therefore we may obtain different clustering results. We thus propose the *repetitive LNNLS* (RLNNLS) classification approach, in Algorithm 3, in which we perform LNNLS learning on the data $maxR$ times. The final class labels of the unlabeled samples are then decided through a voting process on the decisions returned by the $maxR$ LNNLS classifiers. *Majority rule* is used as our voting method. Thus, this technique falls in the domain of ensemble learning [26]. From now on, RLNNLS coupled with MAX and NS rules are denoted as *RLNNLS-MAX* and *RLNNLS-NS*, respectively.

Algorithm 2 *LNNLS Classifier*

```
Input: $\boldsymbol{X}_{m\times n}$: training set with $m$ features and $n$ samples
       $\boldsymbol{c}$: class labels of the training samples
       $\boldsymbol{S}_{m\times p}$: $p$ unknown samples without labels
       Clust_Method: clustering method; NMF, semi-NMF
Output: $\boldsymbol{p}$: predicted class labels of the $p$ unknown samples

  $[\mathbf{X}_i, \mathbf{S}_i]_{i=1}^{i=k} \leftarrow Clustering([\boldsymbol{X}, \boldsymbol{S}], Clust\_Method)$;
     $\{\mathbf{X}_i$ : set of labeled samples in $i$-th cluster$\}$
     $\{\mathbf{S}_i$ : set of unlabeled samples in $i$-th cluster$\}$

  for $i \leftarrow 1$ to $k$ do
    if $\boldsymbol{S}_i = \emptyset$ then
      continue;
    end if

    if $\boldsymbol{X}_i = \emptyset$ then
      $\boldsymbol{p}_i \leftarrow NNLS(\boldsymbol{X}, \boldsymbol{c}, \boldsymbol{S}_i)$;
          $\{\boldsymbol{p}_i$ : predicted labels of samples in $\boldsymbol{S}_i\}$
      continue;
    end if

    if $Homogeneous(\boldsymbol{X}_i)$ then
      $\boldsymbol{p}_i \leftarrow \boldsymbol{c}_i$;
          $\{\boldsymbol{c}_i$ is the unique class label appearing in $\boldsymbol{X}_i\}$
      continue;
    end if
    $\boldsymbol{p}_i \leftarrow NNLS(\boldsymbol{X}_i, \boldsymbol{c}_i, \boldsymbol{S}_i)$;
  end for

  return $\boldsymbol{p}$.
```

Algorithm 3 *RLNNLS Classifier*

```
Input: $\boldsymbol{X}_{m\times n}$: training set with $m$ features and $n$ samples
       $\boldsymbol{c}$: class labels of the training samples
       $\boldsymbol{S}_{m\times p}$: $p$ unknown samples without labels
Output: $\boldsymbol{p}$: predicted class labels of the $p$ unknown samples

  for $r \leftarrow 1$ to $maxR$ do
    $\mathbf{p}_r \leftarrow LNNLS(\boldsymbol{X}, \boldsymbol{c}, \boldsymbol{S})$;
        $\{\mathbf{p}_r$ : predicted labels at the $r$-th iteration$\}$
  end for

  return $\boldsymbol{p} \leftarrow Majority\_Vote(\mathbf{p}_1, \ldots, \mathbf{p}_{maxR})$.
```

3.4 Algorithms

The core optimization method in the above approaches is that of NMF. For NNLS classifier, we only need to set the basis matrix as the training data, and update the coefficient matrix. For LNNLS, NMF is required as a clustering algorithm. The NMF clustering algorithm is summarized in the below:

1) Each column of $\boldsymbol{X}$ is a sample, and its rows correspond to features, these samples will be clustered into $k \leq n$ clusters.
2) Each column of $\boldsymbol{A}$ is called a meta-sample, and corresponds to a cluster.
3) The i-th sample $\boldsymbol{x}_i$ is approximated by a non-negative linear combination of the meta-samples, and the coefficients are in the i-th column of $\boldsymbol{Y}$, that is
$$\boldsymbol{x}_i = [\boldsymbol{a}_1, \boldsymbol{a}_2, \cdots, \boldsymbol{a}_k][y_{1i}, y_{2i}, \cdots, y_{ki}]^{\mathrm{T}}$$
.
4) $\boldsymbol{x}_i$ is assigned to the q-th cluster, where $q = \arg\max_{1 \leq j \leq k} y_{ji}$.

The optimization task expressed in Equation (1) uses the Euclidean distance, other objectives corresponding to other distance metrics are discussed in the literature [31]. This optimization is not convex, thus block-coordinate-decent algorithms are proposed in literature. There are mainly three iterative algorithms to optimize this task: *multiple-update rule* (MUR) [7] [32], *alternating least squares* (ALS) [33], and *alternating non-negative least squares* (ANLS) [34]. In each iteration of ANLS, fixing $\boldsymbol{A}$ (or $\boldsymbol{Y}$) and solving $\boldsymbol{Y}$ (or $\boldsymbol{A}$) is a NNLS subproblem. Optimal solution of this subproblem can be found by the standard NNLS algorithm [35] which is an active-set algorithm. The convergence properties of these methods have been discussed in [34]. It has been shown that the MUR algorithm does not always converge to a stationary point within a realistic number of iterations, that the convergence of the ALS algorithm is hard to analyze, and that the ANLS algorithm converges to a stationary point since an optimal solution for each subproblem at each iteration can be found. It has also been proven that the ANLS algorithm takes much less amount of iterations and computing time than MUR. In this paper, we use the ANLS-based NMF algorithm, which is shown in Algorithm 4. This ANLS approach is based on the *fast computational NNLS* (FCNNLS) optimization algorithm of [36], which in turns dramatically improves the speed of the standard NNLS algorithm.

An alternating update rule using pseudo-inverse has been proposed in [10] for solving the semi-NMF problem of in Equation (2), that is when $\mathbf{A}$ is allowed to contain negative values. Although the fitting residual does monotonically decrease under the update rule of $\boldsymbol{Y}$, the optimal solution to the subproblem is not guaranteed. Therefore, if we follow the same theory as in [34], the algorithm in [10] is not even guaranteed to converge to a stationary point. Another drawback of this algorithm is that the update rule based methods take larger number of iterations and longer time than ANLS algorithm to meet the same termination criteria. The FCNNLS algorithm can find an optimal solution to this subproblem at each iteration, and hence, we propose to replace the update rule of $\boldsymbol{Y}$ with FCNNLS, so that our solver as described in Algorithm 5 can converge to a stationary point.

Algorithm 4 *NMF Algorithm*

```
Input: $X_{m\times n}$: a non-negative matrix
       $k$: the number of basis vectors
Output: $A_{m\times k}$: the non-negative basis matrix
        $Y_{k\times n}$: the non-negative coefficient matrix

  $Y \leftarrow rand(k,n)$;
    {Randomly initialize $Y$, (0, 1)-uniform distribution}

  repeat
    $A \leftarrow FCNNLS(Y, X)$;
    $Y \leftarrow FCNNLS(A, X)$;
  until $\frac{1}{2}\|X - AY\|_F^2 \leq \varepsilon$

  return $A$ and $Y$.
```

Algorithm 5 *Semi-NMF Algorithm*

```
Input: $X_{m\times n}$: a non-negative matrix
       $k$: the number of basis vectors
Output: $A_{m\times k}$: the non-negative basis matrix
        $Y_{k\times n}$: the non-negative coefficient matrix

  $Y \leftarrow rand(k,n)$;
    {Randomly initialize $Y$, (0, 1)-uniform distribution}

  repeat
    $A \leftarrow XY^{\dagger}$;
      {$Y^{\dagger}$: Moore-Penrose pseudo-inverse of $Y$}
    $Y \leftarrow FCNNLS(A, X)$;
  until $\frac{1}{2}\|X - AY\|_F^2 \leq \varepsilon$

  return $A$ and $Y$.
```

3.5 Kernel Extensions

In this section, we show that NNLS and the NMF methods need only the inner products of the samples rather the high-dimensional sample data set itself. Two benefits can be achieved using the inner products. First, classification and clustering become dimension-free, and second, corresponding kernel extensions can be obtained through replacing the inner products with kernel matrices.

The non-negative least squares is essentially a non-negative quadratic programming. To see this, we rewrite Equation (3) as follows:

$$\min_{\boldsymbol{Y}} \sum_{i=1}^{p} \frac{1}{2}\boldsymbol{y}_i^{\mathrm{T}}\boldsymbol{H}\boldsymbol{y}_i + \boldsymbol{c}_i^{\mathrm{T}}\boldsymbol{y}_i, \text{ s.t. } \boldsymbol{Y} \geq 0, \tag{5}$$

where, $\boldsymbol{H} = \boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}$ and $\boldsymbol{c}_i = -\boldsymbol{X}^{\mathrm{T}}\boldsymbol{s}_i$. Vector $\boldsymbol{s}_i$ is the i-th column of $\boldsymbol{S}$ in Equation (3). From Equaion (5), we find that the optimization and the classification tasks only need the inner products of training and test samples. If we replace the inner products with kernel matrices then we obtain kernel versions of our NNLS classifiers.

NMF clustering is employed in the LNNLS approaches, in which the inner products of samples are also used in Algorithm 5 as well. This is proven in the following.

To obtain the clustering results, only $\boldsymbol{Y}$ is required as output. $\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}$ and $\boldsymbol{A}^{\mathrm{T}}\boldsymbol{X}$ are needed to update $\boldsymbol{Y}$. We have $\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A} = (\boldsymbol{Y}^{\dagger})^{\mathrm{T}}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}\boldsymbol{Y}^{\dagger}$ and $\boldsymbol{A}^{\mathrm{T}}\boldsymbol{X} = (\boldsymbol{Y}^{\dagger})^{\mathrm{T}}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}$. Therefore, to obtain the new $\boldsymbol{Y}$ in the current iteration, we need only the inner products $\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}$ and the matrix $\boldsymbol{Y}$ obtained from previous iteration. If $\boldsymbol{A}$ is not directly needed in the post-analysis, we can return $\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}$ and $\boldsymbol{Y}$ in the output. Therefore, we can obtain the kernel NMF version by replacing these inner products with kernel matrices.

4 Experiment

TABLE 2
High Dimensional Microarray Data Sets.

Data	+/-	#Classes	#Features	#Samples
Adenoma [37]	±	2	7457	18+18=36
Breast [38]	+	2	24481	44+34=74
Colon [39]	+	2	2000	40+22=62
DLBCL-NIH [40]	±	2	7399	102+138=240
Leukemia [41]	±	2	7129	47+25=72
Lung [42]	±	2	7129	10+86=96
Medulloblastoma [43] [8]	+	2	5893	25+9=34
Prostate [44]	±	2	12600	59+77=136
ALL [45]	±	6	12625	15+27+64+20+43+79=248
ALLAML [41] [8]	+	3	5000	19+8+11=38
Breast5 [46]	±	5	13582	39+22+53+31+13=158
CNS [47]	±	5	7129	10+10+10+4+8=42
MLL [48]	±	3	12582	24+20+28=72
SRBCT [49]	+	4	2308	23+8+12+20=63
Bladder [50]	±	2	2143	32+16=48
BreastBerkeley [51]	±	2	2149	72+69=141
Melanoma [52]	±	2	3649	35+43=78

In order to investigate the classification performance of our family of NNLS classifiers for high dimensional biological data, we ran it on 14 microarray gene expression data sets (including 8 binary-class and 6 multi-class data sets) and 3 binary-class array-based comparative genomic hybridization (aCGH) data sets. The aCGH technique is used to identify DNA copy numbers. These data are summarized in Table 2. The aCGH data sets are in the last block of the table. The numbers of features vary from 2000 to 24481 among these 17 data sets. The numbers of samples vary from 34 to 248. The gene expression intensities should be naturally non-negative. However, due to preprocessing when producing the data, some data have negative components. 5 out of these 14 data sets, are non-negative. The aCGH data are presented as $\log_2$-ratio, therefore have both positive and negative values. In this study, for non-negative data, NMF is used to cluster the data; otherwise, semi-NMF is employed to do the same task. The optimal number of clusters was obtained by linear search, though a model selection method proposed in [8] is widely used. This is because we observed that the optimal number of clusters suggested by that model selection method does not always results in the highest classification accuracy. We used linear kernel in our NNLS family. We compared our NNLS family with three classical local learners, namely, k-NN, SVM with RBF kernel, and LSVM. We implemented all these methods in MATLAB. The recently proposed SPLSDA and SGPLS were also compared with our methods. We use the SPLS package (version 2.1-1) in R which is available from [6]. Parameters of all methods were selected by linear or grid search. 4-fold *cross-validation* (CV) was used to split a whole data set into labeled training set and unlabeled test set. We defined the *accuracy* of a given classifier as the ratio of the number of correctly predicted test samples to the total number of test samples. For each data set, 4-fold CV was 20 times, and the averages and standard deviations over the 20 runs were computed. Fig. 1 shows the results on six data sets: Breast, Colon, Prostate, ALL, Breast5, and BreastBerkeley. SGPLS fails on the ALL data set which has a very large number of features. LSVM gives better performance compared with SVM on Breast, ALL, and BreastBerkeley, and gives very poor performance on Colon and Prostate. From all six groups of results, it can be seen that medium performance was obtained by k-NN. It can also be seen that our NNLS classifiers generally perform very well. We can also see that NNLS-NS has better accuracies than NNLS-MAX on Prostate and Breast5. We can observe that LNNLS-NNLS obtained better results than NNLS-NS. Furthermore, the RLNNLS-NS classifier has better accuracies than LNNLS-NS on Breast, Prostate, and ALL; meanwhile comparable results were obtained by both of them over the remaining three data sets.

4.1 Statistical Comparison

We used the Friedman test with post-hoc Nemenyi test to further statistically compare all the classifiers over their mean accuracies on the 17 data sets. Friedman test is a non-parametric approach which compares multiple classifiers on multiple data sets [53]. It was recommended by [53] because it is simple, safe, and robust. The null hypothesis of our Friedman test is that all classifiers are equivalent in term of error rate. Once it is rejected, Nemenyi test is used to find the difference among the classifiers. In Nemenyi test, if the distance between the average ranks of a pair of classifiers are larger than the *crucial difference* (CD), then we say that they are significantly different. The significance level was set to $\alpha = 0.05$ in our experiment. The null hypothesis was rejected in our test, and thus the post-hoc Nemenyi test was conducted. According to the test results, we have the following interpretations which are consistent with our above observations from Fig. 1.

NNLS using NS rule is significantly better than the one using MAX rule. NNLS-NS is significantly better than k-NN. There is a large difference between the average ranks of NNLS-NS and SVM, though the difference is not significant at the current significance level. If we increase α slightly, then NNLS-NS and SVM will be significantly different. The average rank of LNNLS is not worse than NNLS. RLNNLS-NS obtained the best average rank. It is significantly different from LNNLS-MAX, SVM, k-NN, and NNLS-MAX. Finally, the average ranks of SPLSDA and SGPLS are between those of RLNNLS-NS

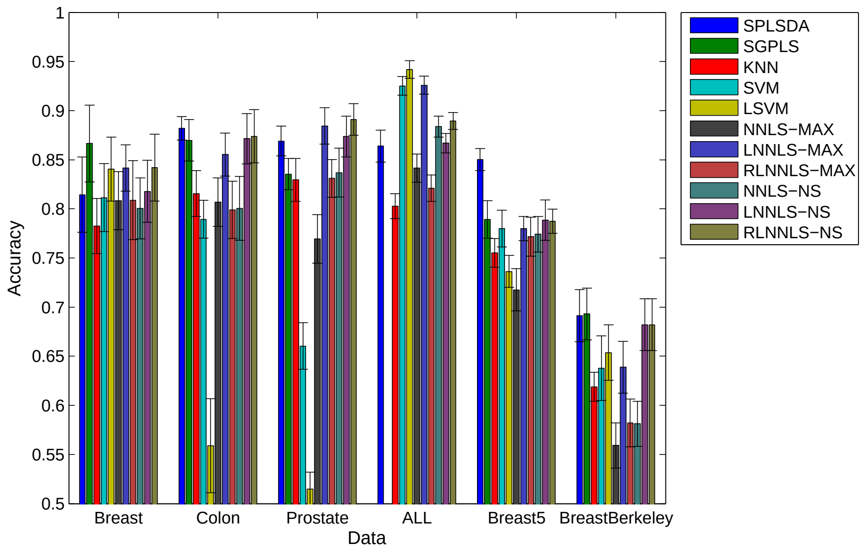

Fig. 1. Classification Performance on Six Data Sets. 1) SGPLS fails on ALL which has multiple classes and a very large number of features. 2) LSVM perform better than SVM on Breast, ALL, and BreastBerkeley, and performs very poor on Colon and Prostate. 3) Medium performance was obtained by k-NN. 4) NNLS classifiers generally perform very well. 5) NNLS-NS performs better than NNLS-MAX on Prostate and Breast5. 6) LNNLS-NNLS obtains better results than NNLS-NS. 7) The RLNNLS-NS gives better accuracies than LNNLS-NS on Breast, Prostate, and ALL; comparable results were obtained by both over the rest.

and LNNLS, and current test does not find significant difference among them.

4.2 Computing Time

The averaged elapsed execution time in seconds of each method was also recorded as a measure of performance. All experiments were performed on an Intel machine (Core TM i7-2600, 3.400 GHz, CPU with 8 GB RAM, with 64-bit Windows 7 Professional operation system). All methods, except the SPLS methods, were implemented in the language MATLAB, 64-bit version 7.11. The SPLS methods was implemented in the language R, 64-bit version 2.12. The computing times of LNNLS and RLNNLS do not include the time of clustering, because clustering can be done only once and then the results can be saved in memory or hard drive. During CV and the iterations of CV, the same clustering results can be reused through simply changing the roles (training or test) of the samples. In real-world applications, to predict the class labels of new but unlabeled samples, the computing times may increase since clustering must be redone. *Online* updating methods can be investigated to reduce the computing time of NMF clustering; see [54] for an example of an online NMF. Fig. 2 shows the computing times of the methods on the six data sets; $\log_2 times$ are given for a better visual comparison.

First, it can be clearly seen that our NNLS approaches are much more efficient than the PLS methods over large or multi-class data. For instance on ALL, the computing time of SGPLS is as 2^{10} and 2^7 times as NNLS-NS and RLNNLS-NS, respectively. Second, our inductive and local NNLS methods are also consistently faster than local SVM. Local SVM becomes computationally costly when the number of classes is large. For example over Breast5 and ALL, it performs slower than our RLNNLS methods. One of the reason for the efficiency of our methods, compared with PLS and LSVM, is because our techniques are naturally multi-class methods, while the supervised PLS and LSVM are binary models which have to use one-against-all, one-against-one, or other strategies for multi-class task. Therefore their computing time is significantly increased in the multi-class case.

4.3 Estimation of Data-Size Requirement

We estimated the minimum data size (that is, the number of training samples) required to obtain significant accuracy, in order to explore the practical reason for the high performance of our approaches. We implemented the permutation-test-based method proposed in [29]. We set the significance level to $\alpha = 0.05$ in these experiments. The minimum data size of NNLS-NS and SVM for each data set is listed in Table 3. In the experiments above, all classifiers obtained poor performance on DLBCL-NIH. Through investigating the data-size requirements, we found that the sample size provided in this data set was not sufficient to obtain a significant accuracy. In Table 3, we can see that NNLS-NS generally needs very few training samples in order to achieve significant accuracy. This also is a possible reason of why LNNLS methods obtained good accuracies, in practice, as they perform prediction via clustering.

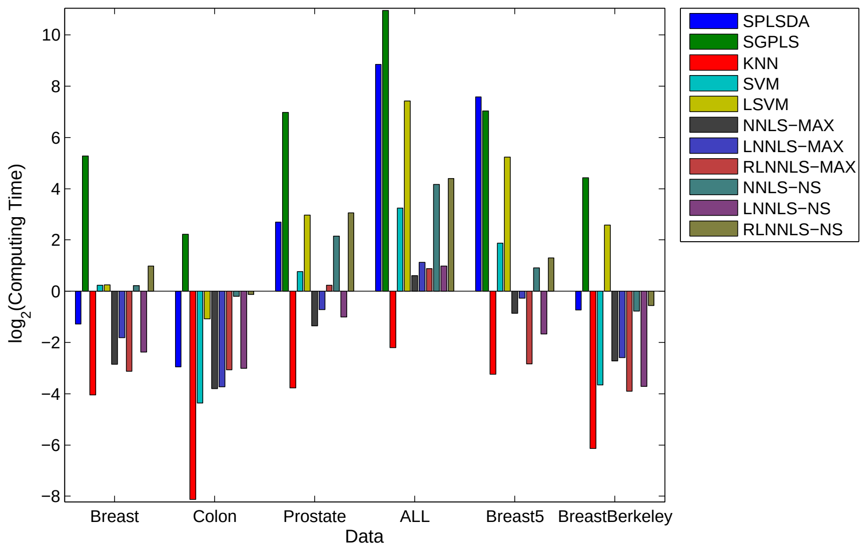

Fig. 2. Computing Times on Six Data Sets. 1) PLS methods are inefficient on large or multi-class data, such as Prostate, ALL, and Breast5. 2) LSVM is also slow. 3) NNLS techniques are generally more efficient than the PLS and LSVM classifiers.

TABLE 3
The Minimum Data Size Required to Obtain Significant Accuracy ($\alpha = 0.05$). NNLS-NS method only requires a very small number of training samples in order to obtain significant accuracy.

Data	NNLS-NS	SVM
Adenoma	6	2
Breast	10	12
Colon	8	12
DLBCL-NIH	-	-
Leukemia	6	12
Lung	15	11
Medulloblastoma	12	2
Prostate	9	11
ALL	6	10
ALLAML	5	10
Breast5	5	11
CNS	5	14
MLL	4	18
SRBCT	4	10
Bladder	10	8
BreastBerkeley	20	11
Melanoma	6	4

5 Conclusion

In this study, we propose a family of NNLS approaches to classify high dimensional biological data. Our methods take advantages of local learning, transductive learning, and ensemble learning principles. Statistical tests on the experimental results shows that our inductive NNLS classifier outperforms k-NN. Our transductive and local NNLS methods can obtain performance comparable with their corresponding inductive versions. The repetitive LNNLS-NS obtained the best rank and outperforms SVM and k-NN. NNLS-NS methods also achieve accuracies comparable to those of the latest sparse PLS method while being much faster. We have shown that our NNLS approaches need only the inner products of the samples, and therefore, their optimization and prediction tasks are dimension-free. Hence, our methods can be kernelized. We estimated the minimum number of samples required to obtain a significant accuracy, and our results imply that NNLS-NS only needs a very small number of training samples for significant accuracy.

Our main motivation was also to highlight the advantages of combining the principles of local learning, transductive learning, and ensemble learning, when learning on (large) high-dimensional data. The idea of cluster-wise classification can also be used in any learning task in which the data sets are high-dimensional and contain un-labeled data. Our future work is to improve the accuracies and computing time of the clustering algorithms. In particular, we plan to devise online clustering approaches in order to avoid *re-clustering* the data whenever new un-labeled data are given. Simultaneous feature selection and clustering will also be investigated for better performance, in terms of both accuracy and computing times.

Acknowledgments

We would like to thank all reviewers who gave us valuable comments. They helped us to improve this paper in both theory and experiment. This research has been supported by IEEE CIS Walter Karplus Summer Research Grant 2010, Ontario Graduate Scholarship 2011-2013, and Canadian NSERC Grants #RGPIN228117-2011.

References

[1] A. Zhang, *Advanced Analysis of Gene Expression Microarray Data*, Singapore: World Scientific, 2009.

[2] M.A. van de Wiel, F. Picard, W.N. van Wieringen, and B. Ylstra, "Preprocessing and downstream analysis of microarray DNA copy number profiles," *Briefings in Bioinformatics*, vol.12, no.1, pp. 10-21, 2010.

[3] X. Li and H. Zhao, "Weighted random subspace method for high dimensional data classification," *Statistics and Its Interface*, vol. 2, pp. 153-159, 2009.

[4] H. Zheng and H. Wu, "Short prokaryotic fragment binning using hierarchical classifier based on linear discriminant analysis and principle component analysis," *Journal of Bioinformatics and Computational Biology*, vol. 8, no. 6, pp. 995-1011, 2010.

[5] G. Fort and S. Lambert-Lacroix, "Classification using partial least squares with penalized logistic regression," *Bioinformatics*, vol. 21, no. 7, pp. 1104-1111, 2005.

[6] D. Chung and S. Keles, "Sparse partial least squares classification for high dimensional data," *Statistical Applications in Genetics and Molecular Bioinformatics*, vol. 9, no. 1, pp. article 17, 2010. Software Available at http://cran.r-project.org/web/packages/spls

[7] D.D. Lee and S. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, pp. 788-791, 1999.

[8] J.P. Brunet, P. Tamayo, T.R. Golub, and J.P. Mesirov, "Metagenes and molecular pattern discovery using matrix factorization," *PNAS*, vol. 101, no. 12, pp. 4164-4169, 2004.

[9] H. Kim and H. Park, "Sparse non-negatice matrix factorization via alternating non-negativity-constrained least squares for microarray data analysis," *Bioinformatics*, vol. 23, no. 12, pp. 1495-1502, 2007.

[10] C. Ding, T. Li, and M.I. Jordan, "Convex and semi-nonnegative matrix factorizations," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 1, pp. 45-55, 2010.

[11] Y. Li and A. Ngom, "Non-negative matrix and tensor factorization based classification of clinical microarray gene expression data," *IEEE International Conference on Bioinformatics & Biomedicine*, 2010, pp.438-443.

[12] A. Blum and A. Langley, "Selection of relevant features and examples in machine learning," *Artificial Intelligence*, vol. 97, pp. 245-271, 1997.

[13] P.A. Mundra, and J.C. Rajapakse, "Gene and sample selection for cancer classification with support vectors based t-statistic," *Neurocomputing*, vol. 73, no. 13-15, pp. 2353-2362, 2010.

[14] R.E. Schapire, "The strength of weak learnability," *Machine Learning*, vol. 5, pp. 197-227, 1990.

[15] H. Liu, H. Motoda, and L. Yu, "A selective sampling approach to active feature selection," *Artificial Intelligence*, vol. 159, pp. 4974, 2004.

[16] L. Bottou and V.N. Vapnik, "Local learning algorithms," *Neural Computation*, vol. 4, no. 6, pp. 888-900, 1992.

[17] V.N. Vapnik, "Principles of risk minimization for learning theory," *Advances in Neural Information Processing Systems*, 1992, vol. 4, pp. 831-838.

[18] J.H. Friedman, "Local learning based on recursive covering," Technical Report, Deptment of Statistics, Stanford University, Stanford, CA, 1996.

[19] H. Cheng, P.-N. Tan, and R. Jin, "Efficient algorithm for localized support vector machine," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 4, pp. 537-549, 2010.

[20] C.G. Atkeson, A.W. Moore, and S. Schaal, "Locally weighted learning," *Artificial Intelligence Review*, vol. 11, pp. 11-73, 1997.

[21] A. Sierra and C.S. Cruz, "Global and local neural network ensembles," *Pattern Recognition Letters*, vol. 19, no. 8, pp. 651-655, 1998.

[22] J. Peng and B. Bhanu, "Local discriminative learning for pattern recognition," *Pattern Recognition*, vol. 34, pp. 139-150, 2001.

[23] V. Vapnik, *Statistical Learning Theory*, New York: Wiley, 1998, pp. 339-371.

[24] O. Chapelle, B. Schölkopf, and A. Zien, *Semi-Supervised Learning*, Cambridge, MA: MIT Press, 2006, pp. 453-472.

[25] X. Zhu, and A.B. Goldberg, *Introduction to Semi-Supervised Learning*, CA: Morgan & Claypool, 2009, pp. 9-19.

[26] L. Rokach, "Ensemble-based classifiers," *Artificial Intelligence Review*, vol. 33, no. 1-2, pp. 1-39, Feb. 2010.

[27] R.A. Jacobs, I.J. Michael, S.J. Nowlan, and C.E. Hinton, "Adaptive mixtures of local experts," *Neural Computation*, vol. 3, pp. 79-87, 1991.

[28] J. Wright, A Y. Yang, A. Ganesh, S.S. Sastry, and Y. Ma. "Robust face recognition via sparse representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 2, pp. 210-227, 2009.

[29] S. Mukherjee, P. Tamayo, S. Rogers, R. Rifkin, A. Engle, C. Campbell, T.R. Golub, and J.P. Mesirov, "Estimating dataset size requirements for classifying DNA microarray data," *Journal of Computational Biology*, vol. 10, no. 2, pp. 119-142, 2003.

[30] C. Bi, M. Becker, and S. Leeder, "Derivation of minimum best sample size from microarray data sets: a Monte Carlo approach," *IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology*, 2011, pp. 1-6.

[31] A. Cichocki, R. Zdunek, A.H. Phan, and S. Amari, *Nonnegative Matrix and Tensor Factorizations: Applications to Exploratory Multi-way Data Analysis and Blind Source Separation*, West Sussex: Wiley, 2009.

[32] D.D. Lee and S. Seung, "Algorithms for non-negative matrix factorization," *Advances in Neural Information Processing Systems*, 2001, vol. 13, pp. 556-562.

[33] M.W. Berry, M. Browne, A.N. Langville, V.P. Pauca, and R.J. Plemmons, "Algorithms and applications for approximate nonnegative matrix factorization," *Computational Statistics & Data Analysis*, vol. 52, pp. 155-173, 2007.

[34] H. Kim and H. Park, "Nonnegative matrix factorization based on alternating nonnegativity constrained least squares and active set method," *SIAM Journal on Matrix Analysis and Applications*, vol. 30, no. 2, pp. 713-730, 2008.

[35] C.L. Lawson and R.J. Hanson, *Solving Least Squares Problems*, Piladelphis: SIAM, 1995, pp.160-165.

[36] M.H. Van Benthem and M.R. Keenan, "Fast algorithm for the solution of large-scale non-negaive constrained least squares problems," *Journal of Chemometrics*, vol. 18, pp. 441-450, 2004.

[37] D.A. Notterman, *et al.*,"Transcriptional gene expression profiles of colorectal adenoma, adenocarcinoma, and normal tissue examined by oligonucleotide arrays," *Cancer Research*, vol. 61, pp. 3124-3130, 2001. Data Available at http://genomics-pubs.princeton.edu/oncology

[38] L.J. Van't Veer, *et al.*,"Gene expression profiling predicts clinical outcome of breast cancer," *Nature*, vol. 415, no. 6871, pp. 530-536, 2002. Data Available at http://bioinformatics.nki.nl/data/van-t-Veer_Nature_2002

[39] U. Alon, *et al.*,"Broad patterns of gene expression revealed by clustering of tumor and normal colon tissues probed by oligonucleotide arrays," *PNAS*, vol. 96, no. 12, pp. 6745-6750, 1999. Data Available at http://genomics-pubs.princeton.edu/oncology

[40] A. Rosenwald, *et al.*, "The use of molecular profiling to predict survival after chemotherapy for diffuse large-B-cell lymphoma," *The New England Journal of Medicine*, vol. 346, no. 25, pp.1937-1947, 2002. Data Available at http://llmpp.nih.gov/DLBCL

[41] T.R. Golub, *et al.*,"Molecular classification of cancer: class discovery and class prediction by gene expression monitoring," *Science*, vol. 286, no. 15, pp. 531-537, 1999. Data Available at http://www.broadinstitute.org/cgi-bin/cancer/datasets.cgi

[42] D.G. Beer, *et al.*, "Gene-expression profiles predict survival of patients with lung adenocarcinoma", *Nature Medicine*, vol. 8, no. 8, pp. 816-823, 2002. Data Available at https://array.nci.nih.gov/caarray/project/beer-00153

[43] S.L. Pomeroy, *et al.*, "Gene expression-based classification and outcome prediction of central nervous system embryonal tumors", *Nature*, vol. 415, no. 6870, pp. 436-442, 2002. Data Available at http://www.broadinstitute.org/mpr/CNS

[44] D. Singh, *et al.*, "Gene expression correlates of clinical prostate cancer behavior," *Cancer Cell*, vol. 1, no. 2, pp. 203-209, 2002. Data Available at http://www.broadinstitute.org/cgi-bin/cancer/datasets.cgi

[45] E.J. Yeoh, *et al.*,"Classification, subtype discovery, and prediction of outcome in pediatric acute lymphoblastic leukemia by gene expression profiling," *Cancer Cell*, vol. 1, pp. 133-143, 2002. Data Available at www.stjuderesearch.org/site/data/ALL1

[46] Z. Hu, *et al.*, "The molecular portraits of breast tumors are conserved across microarray platforms", *BMC Genomics*, vol. 7, no. 96, doi:10.1186/1471-2164-7-96, 2006. Data Available at http://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE1992

[47] S.L. Pomeroy, *et al.*,"Prediction of central nervous system embryonal tumour outcome based on gene expression," *Nature*, vol. 415, pp. 436-442, 2002. Data Available at http://www.broadinstitute.org/cgi-bin/cancer/datasets.cgi

[48] S.A. Armstrong, *et al.*,"MLL translocations specify a distinct gene expression profile that distinguishes a unique leukemia," *Nature Genetics*, vol. 30, pp. 41-47, 2002. Data Available at http://www.broadinstitute.org/cgi-bin/cancer/datasets.cgi

[49] J. Khan, *et al.*,"Classification and diagnostic prediction of cancers using gene expression profiling and artificial neural networks," *Nature*

Medicine, vol. 7, no. 6, pp. 673-679, 2001. Data Available at http://research.nhgri.nih.gov/microarray/Supplement

[50] N. Stransky, *et al.*, “Regional copy number-independent deregulation of transcription in cancer,” *Nature Genetics*, vol. 38, pp. 1386-1396, 2006. Data Available at http://microarrays.curie.fr/publications/oncologie_moleculaire/bladder_TCM and http://www.cs.princeton.edu/~zbarutcu/hhcrf/HHCRF_matlab.zip

[51] K. Chin, *et al.*, “Genomic and transcriptional aberrations linked to breast cancer pathophysiologies,” *Cancer Cell*, vol. 10, no. 6, pp. 529-541, 2006. Data Available at http://cancer.lbl.gov/breastcancer/data.php and http://www.cs.princeton.edu/~zbarutcu/hhcrf/HHCRF_matlab.zip

[52] J. Trolet, *et al.*, “Genomic profiling and identification of high-risk uveal melanoma by array CGH analysis of primary tumors and liver metastases,” *Invest Ophthalmol Vis Sci.*, vol. 50, no. 6, pp. 2572-2580, 2009. Data Available at http://cbio.ensmp.fr/~frapaport/CGHfusedSVM and http://www.cs.princeton.edu/~zbarutcu/hhcrf/HHCRF_matlab.zip

[53] J. Demšar, ”Statistical comparisons of classifiers over multiple data sets,” *Journal of Machine Learning Research*, vol. 7, pp. 1-30, 2006.

[54] J. Driesen and H. Van hamme, ”Modelling vocabulary acquisition, adaptation and generalization in infants using adaptive Bayesian PLSA,” *Neurocomputing*, vol. 74, no. 11, pp. 1874-1882, 2011.

Yifeng Li Yifeng Li received his Bachelor’s and Master’s Degrees from Shandong Polytechnic University, China, in 2006, and 2009, respectively. Now he is a Ph.D. candidate with School of Computer Science, University of Windsor, Canada. His research interests include sparse representation and probabilistic graphical models in machine learning, high dimensional data analysis and systems biology in bioinformatics, and numerical optimization.

Alioune Ngom Alioune Ngom is an associate professor at the University of Windsor, Ontario, Canada. Prior to joining the University of Windsor, he held the position of an assistant professor at the Department of Mathematics and Computer Science at Lakehead University, Thunder Bay, Ontario, Canada, from 1998 to 2000. During his short stay at Lakehead University, he cofunded Genesis Genomics Inc., in 1999, a biotechnology company that specializes in the analysis of mitochondrial genome and the design of biomarkers for the early detection of cancer. His main research interests include but are not limited to computational intelligence and machine learning methods and their applications in computational biology and bioinformatics problems such as microarray analysis, protein analysis, oligonucleotide selection, bioimage analysis, and gene regulatory network analysis. He is a member of IEEE.