BMC Systems Biology

RESEARCH **Open Access**

Sparse representation approaches for the classification of high-dimensional biological data

Yifeng Li*, Alioune Ngom

From IEEE International Conference on Bioinformatics and Biomedicine 2012
Philadelphia, PA, USA. 4-7 October 2012

Abstract

Background: High-throughput genomic and proteomic data have important applications in medicine including prevention, diagnosis, treatment, and prognosis of diseases, and molecular biology, for example pathway identification. Many of such applications can be formulated to classification and dimension reduction problems in machine learning. There are computationally challenging issues with regards to accurately classifying such data, and which due to dimensionality, noise and redundancy, to name a few. The principle of sparse representation has been applied to analyzing high-dimensional biological data within the frameworks of clustering, classification, and dimension reduction approaches. However, the existing sparse representation methods are inefficient. The kernel extensions are not well addressed either. Moreover, the sparse representation techniques have not been comprehensively studied yet in bioinformatics.

Results: In this paper, a Bayesian treatment is presented on sparse representations. Various sparse coding and dictionary learning models are discussed. We propose fast parallel active-set optimization algorithm for each model. Kernel versions are devised based on their dimension-free property. These models are applied for classifying high-dimensional biological data.

Conclusions: In our experiment, we compared our models with other methods on both accuracy and computing time. It is shown that our models can achieve satisfactory accuracy, and their performance are very efficient.

Introduction

The studies in biology and medicine have been revolutionarily changed since the invents of many high-throughput sensory techniques. Using these techniques, the molecular phenomenons can be probed with a high resolution. In the virtue of such techniques, we are able to conduct systematic genome-wide analysis. In the last decade, many important results have been achieved by analyzing the high-throughput data, such as microarray gene expression profiles, gene copy numbers profiles, proteomic mass spectrometry data, next-generation sequences, and so on.

On one hand, biologists are enjoining the richness of their data; one another hand, bioinformaticians are being challenged by the issues of the high-dimensional data. Many of the analysis can be formulated into machine learning tasks. First of all, we have to face to the cures of high dimensionality which means that many machine learning models can be overfitted and therefore have poor capability of generalization. Second, if the learning of a model is sensitive to the dimensionality, the learning procedure could be extremely slow. Third, many of the data are very noise, therefore the robustness of a model is necessary. Forth, the high-throughput data exhibit a large variability and redundancy, which make the mining of useful knowledge difficult. Moreover, the observed data usually do not tell us the key points of the story. We need to discover and interpret the latent factors which drive the observed data.

Many of such analysis are classification problem from the machine learning viewpoint. Therefore in this paper, we focus our study on the classification techniques for high-dimensional biological data. The machine learning techniques addressing the challenges above can be

* Correspondence: li11112c@uwindsor.ca
School of Computer Science, University of Windsor, Windsor, Ontario, Canada

categorized into two classes. The first one aims to directly classify the high-dimensional data while keeping the good capability of generalization and efficiency in optimization. The most popular method in this class is the *regularized basis-expended linear model*. One example is the state-of-the-art *support vector machine* (SVM) [1]. SVM can be kernelized and its result is theoretically sound. Combining different regularization terms and various loss functions, we can have many variants of such linear models [2]. In addition to classification, some of the models can be applied to regression and feature (biomarker) identification. However, most of the learned linear models are not interpretable, while interpretability is usually the requirement of biological data analysis. Moreover, linear models can not be extended *naturally* to multi-class data, while in bioinformatics a class may be composed of many subtypes.

Another technique of tackling with the challenges is dimension reduction which includes feature extraction and feature selection. *Principal component analysis* (PCA) [3] is the oldest feature extraction method and is widely used in processing high-dimensional biological data. The basis vectors produced by PCA are orthogonal, however many patterns in bioinformatics are not orthogonal at all. The classic *factor analysis* (FA) [4] also has orthogonal constraints on the basis vectors, however its Bayesian treatment does not necessarily produce orthogonal basis vectors. Bayesian factor analysis will be introduced in the next section.

Sparse representation (SR) [5] is a parsimonious principle that a sample can be approximated by a sparse linear combination of basis vectors. Non-orthogonal basis vectors can be learned by SR, and the basis vectors may be allowed to be redundant. SR highlights the parsimony in representation learning [6]. This simple principle has many strengthes that encourage us to explore its usefulness in bioinformatics. First, it is very robust to redundancy, because it only select few among all of the basis vectors. Second, it is very robust to noise [7]. Furthermore, its basis vectors are non-orthogonal, and sometimes are interpretable due to its sparseness [8]. There are two techniques in SR. First, given a basis matrix, learning the sparse coefficient of a new sample is called *sparse coding*. Second, given training data, learning the basis vector is called *dictionary learning*. As dictionary learning is, in essence, a sparse matrix factorization technique, *non-negative matrix factorization* (NMF) [9] can be viewed a specific case of SR. For understanding sparse representation better, we will give the formal mathematical formulation from a Bayesian perspective in the next section.

This paper is the significant extension of our preliminary work presented in [10] where sparse representation is treated from regularization and optimization perspectives. In this paper, we formulate sparse representation from a Bayesian viewpoint. We show that using different prior distributions, we can obtain various sparse coding and dictionary learning models. Although there exists some works, for example [11], which apply sparse coding in the classification of biological data, to the best of our knowledge, this is the first time that sparse representation is intensively and systematically studied in the area of bioinformatics. This study has the following contributions:

1. We give a Bayesian treatment on the sparse representation, which is very helpful to understand and design sparse representation models.
2. Kernel sparse coding techniques are proposed for direct classification of high-dimensional biological data.
3. Fast parallel active-set algorithms are devised for sparse coding.
4. An efficient generic framework of kernel dictionary learning for feature extraction is proposed.
5. We reveal that the optimization and decision making in sparse representation is *dimension-free*.

We organize the rest of this paper as follow. We first introduce factor analysis and sparse representation from a Bayesian aspect. Classification method based on sparse coding is then introduced and the active-set methods are proposed for the optimization. Their kernel extensions are given as well. Then various dictionary learning models are given. After that, a generic optimization framework is devised to optimize these models. In the same section, dictionary-learning-based classification and its kernel extension are proposed as well. Then we describe our computational experiments on two high-dimensional data sets. Finally, conclusions and future works are drawn.

Related work from a Bayesian viewpoint

Both (sparse) factor analysis and sparse representation models can be used as dimension reduction techniques. Due to their intuitive similarity, it is necessary to give their definitions for comparison. In this section, we briefly survey the sparse factor analysis and sparse representation in a Bayesian viewpoint. The introduction of sparse representation is helpful to understand the content of the subsequent sections. Hereafter, we use the following notations unless otherwise stated. Suppose the training data is $D \in \mathbb{R}^{m \times n}$ (m is the number of features and n is the number of training instances (samples or observations)), the class information is in the n-dimensional vector $\boldsymbol{c}$. Suppose p new instances are in $B \in \mathbb{R}^{m \times p}$.

Sparse Bayesian (latent) factor analysis

The advantages of *Bayesian (latent) factor analysis model* [12] over likelihood (latent) factor analysis model are that

1. The knowledge regarding the model parameters from experts and previous investigations can be incorporated through the prior.
2. The values of parameters are refined using the current training observations.

The Bayesian factor analysis model [12] can be formulated as

$$(b|\mu, A, x, k) = \mu + Ax + \varepsilon, \tag{1}$$

where $b \in \mathbb{R}^{m \times 1}$ is an observed multivariate variable, $\mu \in \mathbb{R}^{m \times 1}$ is the population mean, $A \in \mathbb{R}^{m \times k}$ is *latent factor loading matrix*, and $x \in \mathbb{R}^{k \times 1}$ is *latent factor score* ($k \ll m$), and $\varepsilon \in \mathbb{R}^{m \times 1}$ is an idiosyncratic *error* term. This model is restricted by the following constraints or assumptions:

1. The error term is normally distributed with mean $\mathbf{0}$ and covariance $\mathbf{\Phi}$: $\boldsymbol{\varepsilon} \sim N(\mathbf{0}, \mathbf{\Phi})$. $\mathbf{\Phi}$ is diagonal on average.
2. The factor score vector is also normally distributed with mean $\mathbf{0}$ and identity covariance $\boldsymbol{R} = \boldsymbol{I}$: $\boldsymbol{x} \sim N(\mathbf{0}, \boldsymbol{R})$; and the factor loading vector is normally distributed: $\boldsymbol{a}_i \sim N(\mathbf{0}, \mathbf{\Delta})$ where $\mathbf{\Delta}$ is diagonal. Alternatively, the factor loading vectors can be normally distributed with mean $\mathbf{0}$ and identity covariance $\mathbf{\Delta} = \boldsymbol{I}$; and the factor score vector is normally distributed with mean $\mathbf{0}$ and diagonal covariance $\boldsymbol{R}$. The benefit of identity covariance either on $\boldsymbol{x}$ or $\boldsymbol{A}$ is that arbitrary scale interchange between $\boldsymbol{A}$ and $\boldsymbol{x}$ due to scale invariance can be avoided.
3. $\boldsymbol{x}$ is independent of $\boldsymbol{\varepsilon}$.

For n training instances $\boldsymbol{D}$, we have the likelihood:

$$p(D|\mu, A, Y, \Phi, k) = \frac{1}{(2\pi)^{\frac{mn}{2}} |\Phi|^{\frac{n}{2}}} e^{-\frac{1}{2}\sum_{i=1}^{n} (d_i - \mu - Ay_i)^{\mathrm{T}} \Phi^{-1} (d_i - \mu - Ay_i)} \tag{2}$$

$$= \frac{1}{(2\pi)^{\frac{mn}{2}} |\Phi|^{\frac{n}{2}}} e^{-\frac{1}{2}\mathrm{tr}[(D - \mu\mathbf{1}^{\mathrm{T}} - AY)^{\mathrm{T}} \mathbf{\Phi}^{-1} (D - \mu\mathbf{1}^{\mathrm{T}} - AY)]}, \tag{3}$$

where tr($\boldsymbol{M}$) is the trace of square matrix $\boldsymbol{M}$.

The variants of Bayesian factor analysis models differ in the decomposition of the joint priors. The simplest one may be $p(\boldsymbol{\mu}, \boldsymbol{A}, \boldsymbol{Y}) = p(\boldsymbol{\mu})p(\boldsymbol{A})p(\boldsymbol{Y})$. Suppose k is fixed a priori. The posterior therefore becomes

$$p(\mu, A, Y|D, k) \propto p(D|\mu, A, Y, \mathbf{\Phi}, k)p(\mu)p(A)p(Y). \tag{4}$$

The model parameters are usually estimated via *Markov chain Monte Carlo* (MCMC) techniques.

Sparse Bayesian factor analysis model imposes a sparsity-inducing distribution over the factor loading matrix instead of Gaussian distribution. In [13], the following mixture of prior is proposed:

$$p(a_{ij}) = (1 - \pi_{ij})\delta_0(a_{ij}) + \pi_{ij}N(a_{ij}|0, 1), \tag{5}$$

where π_{ij} is the probability of a nonzero a_{ij} and $\delta_0(\cdot)$ is the Dirac delta function at 0. Meanwhile, $\boldsymbol{A}$is constrained using the lower triangular method. *Bayesian factor regression model* (BFRM) is the combination of Bayesian factor analysis and Bayesian regression [13]. It has been applied in oncogenic pathway studies [4] as a variable selection method.

Sparse representation

Sparse representation (SR) is a principle that a signal can be approximated by a sparse linear combination of dictionary atoms [14]. The SR model can be formulated as

$$\begin{aligned}(b|A, x, k) &= x_1 a_1 + \cdots + x_k a_k + \varepsilon \\ &= Ax + \varepsilon,\end{aligned} \tag{6}$$

where $A = [a_1, \cdots, a_k]$ is called *dictionary*, $\boldsymbol{a}_i$ is a dictionary atom, $\boldsymbol{x}$ is a sparse *coefficient vector*, and $\boldsymbol{\varepsilon}$ is an error term. $\boldsymbol{A}$, $\boldsymbol{x}$, and k are the model parameters. SR model has the following constraints:

1. The error term is Gaussian distributed with mean zero and isotropic covariance, that is $\boldsymbol{\varepsilon} \sim N(\mathbf{0}, \mathbf{\Phi})$ where $\mathbf{\Phi} = \varphi\boldsymbol{I}$ where φ is a positive scalar.
2. The dictionary atoms is usually Gaussian distributed, that is $\boldsymbol{a}_{\mathrm{i}} \sim N(\mathbf{0}, \mathbf{\Delta})$ where $\mathbf{\Delta} = \boldsymbol{I}$. The coefficient vector should follows a sparsity-inducing distribution.
3. $\boldsymbol{x}$ is independent of $\boldsymbol{\varepsilon}$.

Through comparing the concepts of Bayesian factor analysis and Bayesian sparse representation, we can find that the main difference between them is that the former applies a sparsity-inducing distribution over the factor loading matrix, while the later uses a sparsity-inducing distribution on the factor score vector.

Sparse representation involves sparse coding and dictionary learning. Given a new signal $\boldsymbol{b}$ and a dictionary $\boldsymbol{A}$, learning the sparse coefficient $\boldsymbol{x}$ is termed *sparse coding*. It can be statistically formulated as

$$(b|A) = Ax + \varepsilon. \tag{7}$$

Suppose the coefficient vector has Laplacian prior with zero mean and isotropic variance, that is $p(x|\Gamma) = L(\mathbf{0}, \Gamma) = \frac{1}{(2\gamma)^k} e^{-\frac{||x||_1}{\gamma}}$. The likelihood is Gaussian distributed as $p(b|A, x, \Phi) = N(Ax, \Phi) = \frac{1}{(2\pi)^{\frac{m}{2}} \phi^{\frac{m}{2}}} e^{-\frac{1}{2\phi}||b - Ax||_2^2}$.

The posterior is thus

$$p(x|A,b,\Phi,\Gamma) = \frac{p(b|A,x,\Phi,\Gamma)p(x|A,\Phi,\Gamma)}{p(b)} \propto p(b|A,x,\Phi)p(x|\Gamma). \quad (8)$$

Taking the logarithm, the above is thus

$$L(x) = \log p(b|A,x) + \log p(x) = -\frac{1}{2\phi}||b - Ax||_2^2 - \frac{||x||_1}{\gamma} + c, \quad (9)$$

where c is a constant term. We can see that maximizing the posterior is equivalent to minimizing the following task:

$$\min_x f(x) = \frac{1}{2}||b - Ax||_2^2 + \lambda||x||_1, \quad (10)$$

where $\lambda = \frac{\phi}{\gamma}$. Hereafter we call Equation (10) *l_1-least squares* (l_1 LS) sparse coding model. It is known as the l_1-regularized regression model in regularization theory. It coincides with the well-known *LASSO* model [15], which in fact is a *maximum a posteriori* (MAP) estimation.

Given training data $\boldsymbol{D}$, learning (or estimating) the dictionary $\boldsymbol{A}$, the coefficient vectors $\boldsymbol{Y}$, and the number of dictionary atoms k is called *dictionary learning*. Suppose k is given a priori, and consider the Laplacian prior over $\boldsymbol{Y}$and the Gaussian prior over $\boldsymbol{A}$, and suppose $p(A,Y) = p(A)p(Y) = \prod_{i=1}^{k}(p(a_i))\prod_{i=1}^{n}(p(y_i))$. We thus have the prior:

$$p(A,Y|\Delta,\Gamma) = \frac{1}{(2\pi)^{\frac{k}{2}}}e^{\sum_{i=1}^{k} -\frac{1}{2}||a_i||_2^2}\frac{1}{(2\gamma)^{kn}}e^{\sum_{i=1}^{n} -\frac{||y_i||_1}{\gamma}} \quad (11)$$

The likelihood is

$$p(D|A,Y,\Phi) = \frac{1}{(2\pi)^{\frac{mn}{2}}\phi^{\frac{mn}{2}}}e^{-\frac{1}{2\phi}\mathrm{tr}(||D-AY||_F^2)}. \quad (12)$$

The posterior is

$$p(A,Y|D,\Delta,\Gamma,\Phi) = \frac{p(D|A,Y,\Delta,\Gamma,\Phi)p(A,Y|\Delta,\Gamma,\Phi)}{p(D)} \quad (13)$$

$$\propto p(D|A,Y,\Phi)p(A|\Delta)p(Y|\Gamma). \quad (14)$$

Ignoring the normalization term (that is the marginal likelihood), the log-posterior is thus

$$L(A,Y) = -\sum_{i=1}^{n}\frac{1}{2\phi}||d_i - Ay_i||_2^2 - \sum_{i=1}^{k}\frac{1}{2}||a_i||_2^2 - \sum_{i=1}^{n}\frac{||y_i||_1}{\gamma} + c. \quad (15)$$

Therefore the MAP estimation of dictionary learning task is

$$\min_{A,Y} f(A,Y) = \sum_{i=1}^{n}\frac{1}{2}||d_i - Ay_i||_2^2 + \sum_{i=1}^{k}\frac{\alpha}{2}||a_i||_2^2 + \sum_{i=1}^{n}\lambda||y_i||_1, \quad (16)$$

where $\alpha = f$ and $\lambda = \frac{\phi}{\gamma}$. Equation (16) is known as a dictionary learning model based on l_1-regularized least squares.

In the literature, the kernel extension of sparse representation is realized in two ways. The first way is to use empirical kernel in sparse coding as in [16], where dictionary learning is not considered. The second way is the one proposed in [17], where dictionary learning is involved. However, the dictionary atoms are represented and updated explicitly. This could be intractable, as the number of dimensions of dictionary atoms in the feature space is very high even infinite. In the later sections, we give our kernel extensions of sparse coding and dictionary learning, respectively, where any kernel functions can be used and the dictionary is updated efficiently.

Sparse coding methods

Since, the l_1LS sparse coding (Equation (10)) is a two-sided symmetric model, thus a coefficient can be zero, positive, or negative [18]. In Bioinformatics, l_1LS sparse coding has been applied for the classification of microarray gene expression data in [11]. The main idea is in the following. First, training instances are collected in a dictionary. Then, a new instance is regressed by l_1LS sparse coding. Thus its corresponding sparse coefficient vector is obtained. Next, the regression residual of this instance to each class is computed, and finally this instance is assigned to the class with the minimum residual.

We generalize this methodology in the way that the sparse code can be obtained by many other regularization and constraints. For example, we can pool all training instances in a dictionary (hence $k = n$ and $\boldsymbol{A} = \boldsymbol{D}$), and then learn the non-negative coefficient vectors of a new instance, which is formulated as an one-sided model:

$$\min_x \frac{1}{2}||b - Ax\|_2^2 \text{ s.t. } x \geq 0. \quad (17)$$

We called this model the *non-negative least squares* (NNLS) sparse coding. NNLS has two advantages over l_1LS. First, the non-negative coefficient vector is more easily interpretable than coefficient vector of mixed signs, under some circumstances. Second, NNLS is a non-parametric model. From a Bayesian viewpoint, Equation (17) is equivalent to the MAP estimation with the same Gaussian error as in Equation (6), but the following discrete prior:

$$Pr(x) = \begin{cases} 0.5^k & \text{if } x \geq 0, \\ 0 & \text{otherwise.} \end{cases} \tag{18}$$

This non-negative prior implies that, the elements in $\boldsymbol{x}$ are independent, and the probability of $x_i = 0$ is 0.5 and the probability of $x_i > 0$ is 0.5 as well. (That is the probabilities of x_i being either 0 or positive are equal, and the probability of being negative is zero.) Inspired by many sparse NMFs, l_1-regularization can be additionally used to produce more sparse coefficients than NNLS above. The combination of l_1-regularization and non-negativity results in the l_1NNLS sparse coding model as formulated below:

$$\min_x \frac{1}{2}\|b - Ax\|_2^2 + \lambda^T x \text{ s.t. } x \geq 0. \tag{19}$$

We call Equation (19) the l_1*NNLS* model. It is more flexible than NNLS, because it can produce more sparse coefficients as controlled by λ. This model in fact uses the following prior:

$$p(x) = \begin{cases} \frac{1}{\gamma^k} e^{-\frac{\|x\|_1}{\gamma}} & \text{if } x \geq 0, \\ 0 & \text{otherwise.} \end{cases} \tag{20}$$

Now, we give the generalized sparse-coding-based classification approach in details. The method is depicted in Algorithm 1. We shall give the optimization algorithms, later, required in the first step. The NN rule mentioned in Algorithm 1 is inspired by the usual way of using NMF as a clustering method. Suppose there are C classes with labels 1, ..., C. For a given new instance $\boldsymbol{b}$, its class is $l = \arg\max_{i=1,\ldots,k} x_i$. It selects the maximum coefficient in the coefficient vector, and then assigns the class label of the corresponding training instance to this new instance. Essentially, this rule is equivalent to applying *nearest neighbor* (NN) classifier in the column space of the training instances. In this space, the representations of the training instances are identity matrix. The NN rule can be further generalized to the weighted *K-NN* rule. Suppose a K-length vector $\bar{x}$ accommodates the K-largest coefficients from x, and $\bar{c}$ has the corresponding K class labels. The class label of $\boldsymbol{b}$ can be designated as $l = \arg\max_{i=1,\ldots,C} s_i$ where $\delta_i(\bar{x})$. $\delta_i(\bar{x})$ is a K-length vector and is defined as

$$(\delta_i(\bar{x}))_j = \begin{cases} \bar{x}_j & \text{if } \bar{c}_j = i, \\ 0 & \text{otherwise.} \end{cases} \tag{21}$$

The maximum value of K can be k, the number of dictionary atoms. In this case, K is in fact the number of all non-zeros in $\boldsymbol{x}$. Alternatively, the *nearest subspace* (NS) rule, proposed in [19], can be used to interpret the sparse coding. NS rule takes the advantage of the discrimination of property in the sparse coefficients. It assigns the class with the minimum regression residual to $\boldsymbol{b}$. Mathematically, it is expressed as $j = \min_{1 \leq i \leq C} r_i(\boldsymbol{b})$ where $r_i(\boldsymbol{b})$ is the regression residual corresponding to the i-th class and is computed as $r_i(b) = \|b - A\delta_i(x)\|_2^2$, where $\boldsymbol{\delta_i(x)}$ is defined analogically as in Equation (21).

Algorithm 1 *Sparse-coding-based classification*

Input: $\boldsymbol{A}_{m\times n}$: n training instances, $\boldsymbol{c}$: class labels, $\boldsymbol{B}_{m\times p}$: p new instances

Output: $\boldsymbol{p}$: predicted class labels of the p new instances

1. Normalize each instance to have unit l_2-norm.
2. Learn the sparse coefficient matrix $\boldsymbol{X}$, of the new instances by solving Equation (10), (17), or (19).
3. Use a sparse interpreter to predict the class labels of new instances, e.g. the NN, K-NN, or NS rule.

Optimization

Active-set algorithm for l_1 LS

The problem in Equation (10) is equivalent to the following non-smooth unconstrained *quadratic programming* (QP):

$$\min_x \frac{1}{2} x^T H x + g^T x + \lambda \|x\|_1, \tag{22}$$

where $\boldsymbol{H}_{k\times k} = \boldsymbol{A}^T\boldsymbol{A}$, and $\boldsymbol{g} = -\boldsymbol{A}^T\boldsymbol{b}$. We thus know that the l_1LS problem is a l_1QP problem. This can be converted to the following smooth constrained QP problem:

$$\min_{x,u} \frac{1}{2} x^T H x + g^T x + \lambda^T u \text{ s.t. } -u \leq x \leq u, \tag{23}$$

where $\boldsymbol{u}$ is an auxiliary vector variable to squeeze $\boldsymbol{x}$ towards zero. It can be further written into the standard form:

$$\min_{x,u} \frac{1}{2}[x^T, u^T] \begin{bmatrix} H & 0_{k\times k} \\ 0_{k\times k} & 0_{k\times k} \end{bmatrix} \begin{bmatrix} x \\ u \end{bmatrix} + g^T x + \lambda^T u \tag{24}$$

$$s.t. \begin{bmatrix} I_{k\times k} & -I_{k\times k} \\ -I_{k\times k} & -I_{k\times k} \end{bmatrix} \leq 0,$$

where $\boldsymbol{I}$ is an identity matrix. Obviously, the Hessian in this problem is positive semi-definite as we always suppose $\boldsymbol{H}$ is positive semi-definite in this paper.

A general active-set algorithm for constrained QP is provided in [20], where the main idea is that a working set is updated iteratively until it meets the true active set. In each iteration, a new solution $\boldsymbol{x}_t$ to the QP constrained only by the current working set is obtained. If the update step $\boldsymbol{p}_t = \boldsymbol{x}_t - \mathbf{x}_{t-1}$ is zero, then Lagrangian multipliers of the current active inequalities are computed. If all these multipliers corresponding to the working set are non-negative, then the algorithm terminates with an optimal solution. Otherwise, an active inequality is dropped from the current working set. If the update step $\boldsymbol{p}_t$ is nonzero, then an update length α is computed using the inequality of the current passive set. The new solution is updated as $\boldsymbol{x}_t = \boldsymbol{x}_{t-1} + \alpha\boldsymbol{p}_t$. If $\alpha < 1$, then a blocking inequality is added to the working set.

To solve our specific problem efficiently in Equation (24), we have to modify the general method, because i) our constraint is sparse, for the i-th constraint, we have $x_i - u_i \leq 0$ (if $i \leq k$) or $-x_i - u_i \leq 0$ (if $i > k$); and ii) when u_i is not constrained in the current working set, the QP constrained by the working set is unbounded, therefore it is not necessary to solve this problem to obtain $\boldsymbol{p}_t$. In the later situation, $\boldsymbol{p}_t$ is unbounded. This could cause some issues in numerical computation. Solving the unbounded problem is time-consuming if the algorithm is unaware of the unbounded issue. If $\boldsymbol{p}_t$ contains positive or negative ∞, then the algorithm may crash.

We propose the revised active-set algorithm in Algorithm 2 for l_1LS sparse coding. To address the potential issues above, we have the following four modifications. First, we require that the working set is complete. That is all the variables in $\boldsymbol{u}$ must be constrained when computing the current update step. (And therefore all variables in $\boldsymbol{x}$ are also constrained due to the specific structure of the constraints in our problem.) For example, if $k = 3$, a working set {1, 2, 6} is complete as all variables, x_1, x_2, x_3, u_1, u_2, u_3, are constrained, while {1, 2, 4} is not complete, as u_3 (and x_3) is not constrained. Second, the update step of the variables that are constrained once in the working set are computed by solving the equality constrained QP. The variables constrained twice are directly set to zeros. In the example above, suppose the current working set is {1, 2, 4, 6}, then x_2, x_3, u_2, u_3 are computed by the constrained QP, while x_1 and u_1 are zeros. This is because the only value satisfying the constraint $-u_1 = x_1 = u_1$ is $x_1 = u_1 = 0$. Third, in this example, we do not need to solve the equality constrained QP with four variables. In fact we only need two variables by setting $u_2 = -x_2$ and $u_3 = x_3$. Forth, once a constraint is dropped from the working set and it becomes incomplete, other inequalities must be immediately added to it until it is complete. In the initialization of Algorithm 2, we can alternatively initialize $\boldsymbol{x}$ by 0's. This is much efficient than $\boldsymbol{x} = (\boldsymbol{H})^{-1}(-\boldsymbol{g})$ for large-scale sparse coding and very sparse problems.

Active-set algorithm for NNLS and l_1 NNLS

Both the NNLS problem in Equation (17) and the l_1NNLS problem in Equation (19) can be easily reformulated to the following *non-negative QP* (NNQP) problem:

$$\min \frac{1}{2}x^{\mathrm{T}}H\,x + g^{\mathrm{T}}x \,\text{s.t.}\; x \geq 0, \tag{26}$$

Algorithm 2 *Active-set l_1QP algorithm*

```
Input: Hessian $\boldsymbol{H}_{k\times k}$, vector $\boldsymbol{g}_{k\times 1}$, scalar $\lambda$
Output: vector $\boldsymbol{x}$ which is a solution to min $\frac{1}{2}x^{\mathrm{T}}Hx + g^{\mathrm{T}}x + \lambda^{\mathrm{T}}u$, s.t. $-u \leq x \leq u$
% initialize the algorithm by a feasible solution and complete working set
 $x = (\boldsymbol{H})^{-1}(-\boldsymbol{g})$; $\boldsymbol{u} = |\boldsymbol{x}|$;
 $\mathcal{R} = \{i, j | \forall i : \text{if } x_i > 0 \text{ let } j = k + i \text{ otherwise } j = i\}$; % initialize working set
 $\mathcal{P} = \{1 : 2k\} - \mathcal{R}$; % initialize inactive(passive) set
 while true do
   % compute update step
   let $\mathcal{R}_{\text{once}}$ be the indices of variables constrained once by $\mathcal{R}$;
   $\boldsymbol{p}_{2k\times 1} = 0$;
   $p_{x,\mathcal{R}_{\text{once}}} = \arg\min_q q^{\mathrm{T}}H_{\mathcal{R}_{\text{once}}}q + [H_{\mathcal{R}_{\text{once}}}x_{\mathcal{R}_{\text{once}}} + g_{\mathcal{R}_{\text{once}}} + \lambda e]^{\mathrm{T}}q$ where $e_i = 1$ if $u_{\mathcal{R}_{\text{once}},i} = x_{\mathcal{R}_{\text{once}},i}$, or -1 if $u_{\mathcal{R}_{\text{once}},i} = x_{\mathcal{R}_{\text{once}},i}$;
   if $\boldsymbol{p} = 0$ then
     obtain Lagrange multiplier $\boldsymbol{\mu}$ by solving
       $$A_{\mathcal{R}}^{T}\mu = -\begin{bmatrix} Hx + g \\ \lambda \end{bmatrix} \tag{25}$$
     where $\boldsymbol{A}$ is the constraint matrix in Equation (24)
     if $\mu_i \geq 0\ \forall i \in \mathcal{R}$ then
       terminate successfully;
     else
       $\mathcal{R} = \mathcal{R} - j$; $\mathcal{P} = \mathcal{P} + j$ where $j = \arg\min_{l\in\mathcal{R}} \mu_l$;
       add other passive constraints to $\mathcal{R}$ until it is complete;
     end if
   end if
   if $\boldsymbol{p} \neq 0$ then
     $\alpha = \min(1, \min_{i\in\mathcal{P}, a_i^{\mathrm{T}}p \geq 0} \frac{-a_i^{\mathrm{T}}[x; u]}{a_i^{\mathrm{T}}p})$;
     $[\boldsymbol{x}; \boldsymbol{u}] = [\boldsymbol{x}; \boldsymbol{u}] + \alpha\boldsymbol{p}$;
     if $\alpha < 1$ then
       $\mathcal{R} = \mathcal{R} + i$; $\mathcal{P} = \mathcal{P} - i$. where $i$ corresponds to $\alpha$;
     end if
   end if
 end while
```

where $\boldsymbol{H} = \boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}$, $\boldsymbol{g} = -\boldsymbol{A}^{\mathrm{T}}\boldsymbol{b}$ for NNLS, and $\boldsymbol{g} = -\boldsymbol{A}^{\mathrm{T}}\boldsymbol{b} + \boldsymbol{\lambda}$ for l_1NNLS.

Now, we present the active-set algorithm for NNQP. This problem is easier to solve than l_1QP as the scale of Hessian of NNQP is half that of l_1QP and the constraint is much simpler. Our algorithm is obtained through generalizing the famous active-set algorithm for NNLS by [21]. The complete algorithm is given in Algorithm 3. The warm-start point is initialized by the solution to the unconstrained QP. As in Algorithm 2, $\boldsymbol{x}$ can be alternatively initialized by 0's. The algorithm keeps adding and dropping constraints in the working set until the true active set is found.

```
Algorithm 3 Active-set NNQP algorithm
Input: Hessian $\boldsymbol{H}_{k\times k}$, vector $\boldsymbol{g}_{k\times 1}$
Output: vector $\boldsymbol{x}$ which is a solution to $\min \frac{1}{2}x^{\mathrm{T}}Hx + g^{\mathrm{T}}x$, s.t. $x \geq 0$
  $x = [(\boldsymbol{H})^{-1}(-\boldsymbol{g})]_{+}$; % $\boldsymbol{x} = [\boldsymbol{y}]_{+}$ is defined as $x_i = y_i$ if $y_i > 0$, otherwise $x_i = 0$
  $\mathcal{R} = \{i | x_i = 0\}$; % initialize active set
  $\mathcal{P} = \{i | x_i > 0\}$; % initialize inactive(passive) set
  $\boldsymbol{\mu} = \boldsymbol{Hx} + \boldsymbol{g}$; % the lagrange multiplier
  while $\boldsymbol{R}$ ¹ Æ and $\min_{i\hat{I}R} (\mu_i) < -e$ do
    % e is a small positive numerical tolerance
    $j = \arg\min_{i\hat{I}R} (\mu_i)$; % get the minimal negative multiplier
    $\mathcal{P} = \mathcal{P} + \{j\}; \mathcal{R} = \mathcal{R} - \{j\}$;
    $t_{\mathcal{P}} = (H_{\mathcal{P}})^{-1}(-g_{\mathcal{P}})$;
    $t_{\mathcal{R}} = 0$;
    while $\min t_{\mathcal{P}} \leq 0$ do
      $\alpha = \min_{i\in\mathcal{P}, t_i \leq 0} \frac{x_i}{x_i - t_i}$;
      $\mathcal{K} = \arg\min_{i\in\mathcal{P}, t_i \leq 0} \frac{x_i}{x_i - t_i}$; % there is one or several indices correspond to $\alpha$
      $x = x + \alpha(t - x)$;
      $\mathcal{P} = \mathcal{P} - -\mathcal{K}; \mathcal{R} = \mathcal{R} + \mathcal{K}$;
      $t_{\mathcal{P}} = (H_{\mathcal{P}})^{-1}(-g_{\mathcal{P}})$;
      $t_{\mathcal{R}} = \boldsymbol{0}$;
    end while
    $x = t$;
    $\mu = Hx + g$;
  end while
```

Parallel active-set algorithms

The formulations of l_1QP and NNQP sparse coding for p new instances are, respectively,

$$\min_{X,U} \sum_{i=1}^{p} \frac{1}{2} x_i^{\mathrm{T}} H x_i + g_i^{\mathrm{T}} x_i + \lambda^{\mathrm{T}} u_i, \tag{27}$$

s.t. $-U \leq X \leq U$,

and

$$\min_{X} \sum_{i=1}^{p} \frac{1}{2} x_i^{\mathrm{T}} H x_i + g_i^{\mathrm{T}} x_i \text{ s.t. } X \geq 0. \tag{28}$$

If we want to classify multiple new instances, the initial idea in [19] and [11] is to optimize the sparse coding one at a time. The interior-point algorithm, proposed in [22], is a fast large-scale sparse coding algorithm, and the proximal algorithm in [23] is a fast first-order method whose advantages have been recently highlighted for non-smooth problems. If we adapt both algorithms to solve our multiple l_1QP in Equation (27) and NNQP in Equation (28), it will be difficult to solve the single problems in parallel and share computations. Therefore, the time-complexity of the multiple problems will be the summation of that of the individual problems. However, the multiple problems can be much more efficiently solved by active-set algorithms. We adapt both Algorithms 2 and 3 to solve multiple l_1QP and NNQP in a parallel fashion. The individual active-set algorithms can be solved in parallel by sharing the computation of matrix inverses (systems of linear equations in essence). At each iteration, single problems having the same active set have the same systems of linear equations to solve. These systems of linear equations can be solved once only. For a large value p, that is large-scale multiple problems, active-set algorithms have dramatic computational advantage over interior-point [22] and proximal [23] methods unless these methods have a scheme of sharing computations. Additionally, active-set methods are more precise than interior-point methods. Interior-point methods do not allow $u_i^2 = x_i^2$ and u_i^2 must be always greater than x_i^2 due to feasibility. But $u_i^2 = x_i^2$ is naturally possible when the i-th constraint is active. $u_i = x_i = 0$ is reasonable and possible. Active-set algorithms do allow this situation.

Kernel extensions

As the optimizations of l_1QP and NNQP only require inner products between the instances instead of the original data, our active-set algorithms can be naturally extended to solve the kernel sparse coding problem by replacing inner products with kernel matrices. The NS decision rule used in Algorithm 1 also requires only inner products. And the weighted K-NN rule only needs the sparse coefficient vector and class information. Therefore, the classification approach in Algorithm 1 can be extended to kernel version. For narrative convenience, we also denote the classification approaches using l_1LS, NNLS, and l_1NNLS sparse coding as l_1LS, NNLS, and l_1NNLS, respectively. Prefix "K" is used for kernel versions.

Dictionary learning methods

We pursue our dictionary-learning-based approach for biological data, based on the following two motivations. First, since sparse-coding-only approach is a lazy learning, the optimization can be slow for large training set. Therefore, learning a concise dictionary is more efficient for future real-time applications. Second, dictionary learning may capture hidden key factors which correspond to biological pathways, and the classification performance may hence be improved. In the following, we first give the dictionary learning models using Gaussian prior and uniform prior, respectively. Next, we give the classification method based on dictionary learning. We then address the generic optimization framework of dictionary learning. Finally, we show that the kernel versions of our dictionary learning models and the classification approach can be easily obtained.

Dictionary learning models

Now we give our dictionary learning models using Gaussian prior and uniform prior over the dictionary atoms, respectively. Both priors aims to get rid off the arbitrary scale interchange between dictionary and coefficient. Suppose $\boldsymbol{D}_{m\times n}$ is the data of n training instances, and the dictionary $\boldsymbol{A}$ to be learned has k atoms. If the Gaussian prior in Equation (6) is used on the dictionary atom, our dictionary learning models of l_1LS, NNLS, and l_1NNLS are expressed as follow, respectively:

$$l_1LS : \min_{A,Y} \frac{1}{2} \parallel D - AY \parallel_F^2 + \frac{\alpha}{2}\text{trace}(A^{\text{T}}A) + \lambda \sum_{i=1}^{n} \parallel y_i \parallel_1, \quad (29)$$

$$NNLS : \min_{A,Y} \frac{1}{2} \parallel D - AY \parallel_F^2 + \frac{\alpha}{2}\text{trace}(A^{\text{T}}A) \quad (30)$$

s.t. $Y \geq 0$,

and

$$l_1NNLS : \min_{A,Y} \frac{1}{2} \parallel D - AY \parallel_F^2 + \frac{\alpha}{2}\text{trace}(A^{\text{T}}A) + \sum_{i=1}^{n} \lambda^{\text{T}} y_i \quad (31)$$

s.t. $Y \geq 0$.

The strength of the Gaussian prior based dictionary learning is that it is flexible to control the scales of dictionary atoms. However, it has two model parameters, which increase the model selection burden in practice. Alternatively, in order to eliminate the parameter α, we design an uniform prior over the dictionary which is expressed as

$$Pr(a_i) = \begin{cases} p \text{ if } \parallel a_i \parallel_2 = 1, \\ 0 \text{ otherwise}, \end{cases} \quad (32)$$

where p is a constant. That is the feasible region of the dictionary atoms is a hypersphere centered at origin with unit radius, and all the feasible atoms have equal probability. The corresponding dictionary learning models are given in the following equations, respectively:

$$l_1LS : \min_{A,Y} \frac{1}{2} \parallel D - AY \parallel_F^2 + \lambda \sum_{i=1}^{n} \parallel y_i \parallel_1 \quad (33)$$

s.t. $a_i^{\text{T}} a_i = 1,\ i = 1, \cdots, k,$

$$NNLS : \min_{A,Y} \frac{1}{2} \parallel D - AY \parallel_F^2 \quad (34)$$

s.t. $a_i^{\text{T}} a_i = 1, \quad i = 1, \cdots, k;\ Y \geq 0,$

and

$$l_1NNLS : \min_{A,Y} \frac{1}{2} \parallel D - AY \parallel_F^2 + \sum_{i=1}^{n} \lambda^{\text{T}} y_i \quad (35)$$

s.t. $a_i^{\text{T}} a_i = 1, \quad i = 1, \cdots, k;\ \ Y \geq 0.$

A generic optimization framework for dictionary learning

We devise block-coordinate-descent-based algorithms for the optimization of the above six models. The main idea is that, in the next step, $\boldsymbol{Y}$ is fixed, and the inner product $\boldsymbol{A}^{\text{T}}\boldsymbol{A}$, rather than $\boldsymbol{A}$ itself, is updated; in the next step, $\boldsymbol{Y}$ is updated while fixing $\boldsymbol{A}^{\text{T}}\boldsymbol{A}$ (a sparse coding procedure). The above procedure is repeated until the termination conditions are satisfied.

Now, we show that $\boldsymbol{A}$ can be analytically obtained. For normal prior over dictionary atoms, the optimization of finding $\boldsymbol{A}$ in Equations (29), (30), and (31) is to solve

$$\min_A f(A) = \frac{1}{2} \parallel D - AY \parallel_F^2 + \frac{\alpha}{2}\text{trace}\,(A^{\text{T}}A) \quad (36)$$

Taking the derivative with respect to $\boldsymbol{A}$ and setting it to zero, we have

$$\frac{\partial f(A)}{\partial A} = AYY^{\text{T}} - DY^{\text{T}} + \alpha A = 0. \quad (37)$$

We hence have

$$A = DY^{\ddagger}, \quad (38)$$

where $\boldsymbol{Y}^{\ddagger} = \boldsymbol{Y}^{\text{T}}(\boldsymbol{Y}\boldsymbol{Y}^{\text{T}} + \alpha\boldsymbol{I})^{-1}$. The inner product $\boldsymbol{A}^{\text{T}}\boldsymbol{A}$ can thus be updated by

$$R = A^{\text{T}}A = (Y^{\ddagger})^{\text{T}}D^{\text{T}}DY^{\ddagger}. \quad (39)$$

We also can compute $\boldsymbol{A}^{\text{T}}\boldsymbol{D}$ by

$$A^{\text{T}}D = (Y^{\ddagger})^{\text{T}}D^{\text{T}}D. \quad (40)$$

For the uniform prior as in Equation (32), updating unnormalized $\boldsymbol{A}$ while fixing $\boldsymbol{Y}$ in Equations (33), (34), and (35) is to solve the generalized least squares:

$$\min_A f(A) = \frac{1}{2} \parallel D - AY \parallel_F^2. \quad (41)$$

Taking derivative with respect to $\boldsymbol{A}$ and setting it to zero, we have

$$A = DY^{\dagger}, \quad (42)$$

Algorithm 4 The generic dictionary learning framework

```
Input: $K = D^{T}D$, dictionary size $k$, $\lambda$
Output: $R = A^{T}A$, $Y$
nitialize $Y$ and $R = A^{T}A$ randomly;
```

```
$r_{prev}$ = $Inf$; % previous residual
for $i = 1 : maxIter$ do
  update $Y$ by solving the active-set based $l_1$LS, NNLS, or $l_1$NNLS sparse coding algorithms;
  if Gassian prior over $A$ then
    update $R = Y^{\dagger T} D^T D Y^{\dagger}$;
  end if
  if uniform prior over $A$ then
    update $R = Y^{\dagger T} D^T D Y^{\dagger}$;
    normalize $R$ by $R = R./\sqrt{\mathrm{diag}(R)\,\mathrm{diag}(R)^T}$;
  end if
  if $i == maxIter$ or $i \bmod l = 0$ then
    % check every $l$ iterations
    $r_{cur} = f(A, Y)$; % current residual of a dictionary learning model
    if $r_{prev} - r_{cur} \le e$ or $r_{cur} \le e$ then
      break;
    end if
    $r_{prev} = r_{cur}$;
  end if
end for
```

where $Y^{\dagger} = Y^{T}(YY^{T})^{-1}$. The inner products of $R = A^TA$ and A^TD are computed similarly as for the Gaussian prior. The normalization of R is straightforward. We have $R = R./\sqrt{\mathrm{diag}(R)\,\mathrm{diag}(R)^T}$, where ./ and $\sqrt{\bullet}$ are element-wise operators. Learning the inner product A^TA instead of A has the benefits of dimension-free computation and kernelization.

Fixing A, Y can be obtained via our active-set algorithms. Recall that the sparse coding only requires the inner products A^TA and A^TD. As shown above, we find that updating Y only needs its previous value and the inner product between training instances.

Due to the above derivation, we have the framework of solving our dictionary learning models as illustrated in Algorithm 4.

Classification approach based on dictionary learning

Now, we present the dictionary-learning-based classification approach in Algorithm 5. The dictionary learning in the training step should be consistent with the sparse coding in the prediction step. As discussed in the previous section, the sparse coding in the prediction step needs the inner products A^TA, B^TB and A^TB which actually is $Y^{\ddagger T}D^TB$ or $Y^{\dagger T}D^TB$.

```
Algorithm 5 Dictionary-learning-based classification
Input: $D_{m\times n}$: $n$ training instances, $c$ the class labels, $B_{m\times p}$: $p$ new instances, $k$: dictionary size
Output: $p$: the predicted class labels of the $p$ new instances
{training step:}
 1: Normalize each training instance to have unit $l_2$ norm.
 2: Learn dictionary inner product $A^TA$ and sparse coefficient matrix $Y$ of training instances by Algorithm 4.
 3: Train a classifier $f(\theta)$ using $Y$ (in the feature space spanned by columns of $A$).
{prediction step:}
 1: Normalize each new instance to have unit $l_2$ norm.
 2: Obtain the sparse coefficient matrix $X$ of the new instances by solving Equation (27), or (28).
 3: Predict the class labels of $X$ using the classifier $f(\theta)$ learned in the training phase.
```

Kernel extensions

For Gaussian dictionary prior, the l_1LS based kernel dictionary learning and sparse coding are expressed in the following, respectively:

$$\min_{A_\phi, Y} \frac{1}{2} \| \phi(D) - A_\phi Y \|_F^2 + \frac{\alpha}{2}\mathrm{trace}(A_\phi^T A_\phi) + \lambda \| Y \|_1, \quad (43)$$

$$\min_{X} \frac{1}{2} \| \phi(B) - A\phi X \|_F^2 + \lambda \| X \|_1,$$

where $f(\cdot)$ is a mapping function. Equations (30), (31), (33), (34), (35) and their sparse coding models can be kernelized analogically. As we have mentioned that the optimizations of the six dictionary learning models, only involves inner products of instances. Thus, we can easily obtain their kernel extensions by replacing the inner products with kernel matrices. Hereafter, if dictionary learning is employed in sparse representation, then prefix "DL" is used before "l_1LS", "NNLS", and "l_1NNLS". If kernel function other than the linear kernel is used in dictionary learning, then prefix "KDL" is added before them.

Computational experiments

Two high-throughput biological data, including a microarray gene expression data set and a protein mass spectrometry data set, are used to test the performance of our methods. The microarray data set is a collection of gene expression profiles of breast cancer subtypes [24]. This data set includes 158 tumor samples from five subtypes measured on 13582 genes. The mass spectrometry data set is composed of 332 samples from normal class and prostate cancer class [25]. Each sample has 15154 features, that is the mass-to-charge ratios. Our experiments are separated into two parts. The performance of sparse coding for direct classification is first investigated with respect to accuracy and running time. Then our dimension reduction techniques using dictionary learning are tested.

Sparse coding for direct classification

When dictionary learning was not involved, the dictionary was "lazily" composed by all the training instances available. In our experiment, the active-set optimization methods for l_1LS, NNLS, and l_1NNLS were tested. The weighted *K*-NN rule and NS rule, mentioned in Algorithm 1, were compared. We set *K* in the *K*-NN rule to the number of all training instances, which is an extreme case as opposite to the NN rule. Linear and *radial basis function* (RBF) kernels were employed. We compared our active-set algorithms with the interior-point [22] method and proximal [23] method for l_1LS sparse coding (abbreviated by l_1LS-IP and l_1LS-PX). Benchmark classifiers, including *k*-NN and SVM using RBF kernel, were compared. We employed four-fold *cross-validation* to partition a data set into training sets and test sets. All the classifiers ran on the same training and test splits for fair comparison. We performed 20 runs of cross-validation and recorded the averages and standard deviations. Line or grid search was used to select the parameters of a classifiers.

The average accuracies of all classifiers with the corresponding standard deviations on both data sets are compared in Figure 1, from which we have four observations. First, the weighted *K*-NN rule obtained comparable accuracies with the NS rule. The advantage of the *K*-NN rule over the NS rule is that the former predicts the class labels based on the sparse coefficient vector solely, while the later has to use the training data to compute regression residuals. Therefore the *K*-NN rule is more efficient and should be preferred. Second, on the Prostate data, the sparse coding method l_1LS and Kl_1LS achieved the best accuracy. This convinces us that sparse coding based classifiers can be very effective for classifying high-throughput biological data. Third, the non-negative models including NNLS, l_1NNLS and their kernel extensions achieved competitive accuracies with the state-of-the-art SVM on both data set. Fourth, the l_1LS sparse coding using our active-set algorithm had the same accuracies as that using the interior-point algorithm and proximal algorithm on Breast data. But on Prostate data, the proximal method yielded a worse accuracy. This implies that our active-set method converges to the global minima as the interior-point method, while performance may be deteriorated by the approximate solution obtained by the proximal method in practice.

The mean running time (in second) of cross-validation are shown in Figure 2. For better comparison, logarithm of base two was taken on the results. First of all, we can clearly see that the interior-point method is very slow for the l_1LS sparse coding. Second, our active-set method is more efficient than the proximal method on Breast data. This is because i) active-set methods are usually the fastest ones for quadratic and linear programmes of small and median size; and ii) expensive computations, like

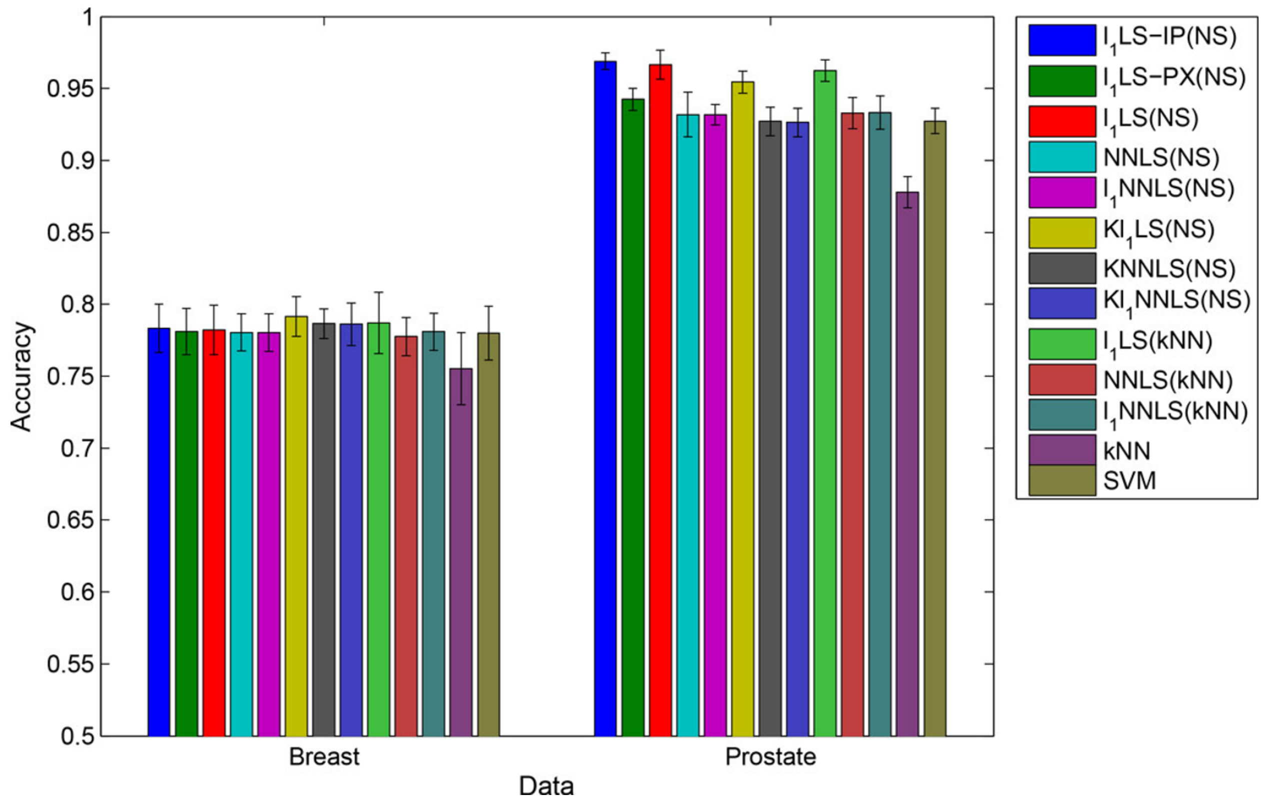

Figure 1 Mean accuracies and standard deviations of sparse coding and benchmark methods.

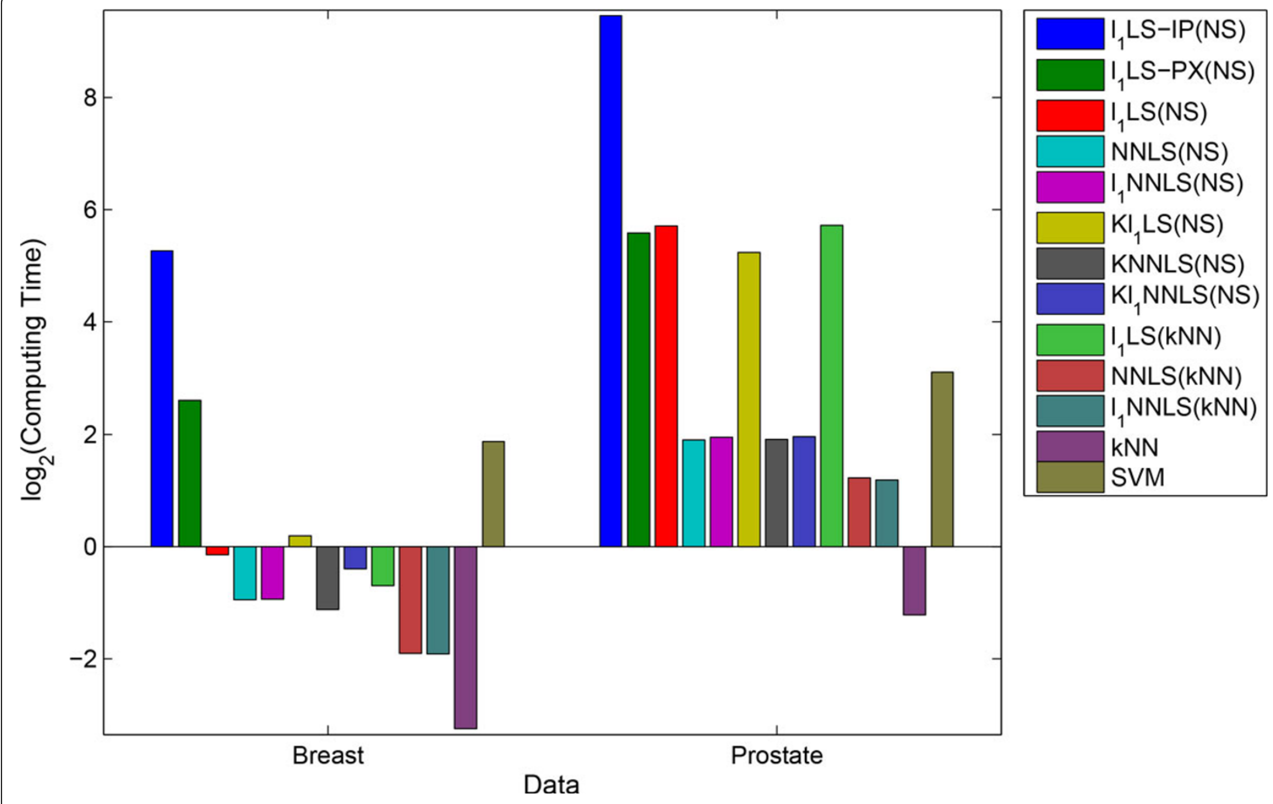

Figure 2 **Log2 computing time of sparse coding and benchmark methods**.

solving systems of linear equations, can be shared in the active-set method. Third, NNLS and l_1NNLS have the same time-complexity. This is reasonable, because both can be formulated to NNQP problem. These non-negative models are much simpler and faster than the non-smooth l_1LS model. Hence, if similar performance can be obtained by l_1LS and the non-negative models in an application, we should give preference to NNLS and l_1NNLS.

Dictionary-learning for feature extraction

The performance of various dictionary learning models with linear and RBF kernels were investigated on both Breast and Prostate data sets. The Gaussian-prior based and uniform-prior based dictionary learning models were also compared. Again, our active-set dictionary learning was compared with the interior-point [22] method and proximal method [23]. The semi-NMF based on multiplicative update rules [26] is also included in the competition. As in the previous experiment, four-fold cross-validation was used. All methods ran on the same splits of training and test sets. We performed 20 runs of cross-validation for reliable comparison. After feature extraction by using dictionary learning on the training set, linear SVM classifier was learned on the reduced training set and used to predict the class labels of test instances.

In Figure 3, we show the mean accuracy and standard deviation of 20 results for each method. First, we can see that the models with Gaussian prior on dictionary atoms obtained similar accuracies as the uniform prior. Second, with the comparison to sparse coding methods on Breast data as given Figure 1, we can see that dictionary learning increases the prediction accuracy. Third, from the comparison of Figures 3 and 1, we find that the dictionary learning based methods - DL-NNLS and DL-l_1NNLS, obtained similar accuracies as the sparse coding methods - NNLS and l1NNLS. This convinces us that dictionary learning is a promising feature extraction technique for high-dimensional biological data. On Prostate data, we can also find that the accuracy obtained by DL-l_1LS is slightly lower than l_1LS. This is may be because the dictionary learning is unsupervised. Fourth, using the model parameters, DL-l_1LS using active-set algorithm obtained higher accuracy than DL-l_1LS-IP and DL-l_1LS-PX on Prostate data. The accuracy of DL-l1LS is also slightly higher than that of DL-l_1LS-IP on Breast data. Furthermore, the non-negative DL-NNLS yielded the same performance as the well-known semi-NMF, while further corroborates the satisfactory performance of our dictionary learning framework. Finally, the kernel dictionary learning models achieved similar performance as their linear counterparts. We believe that

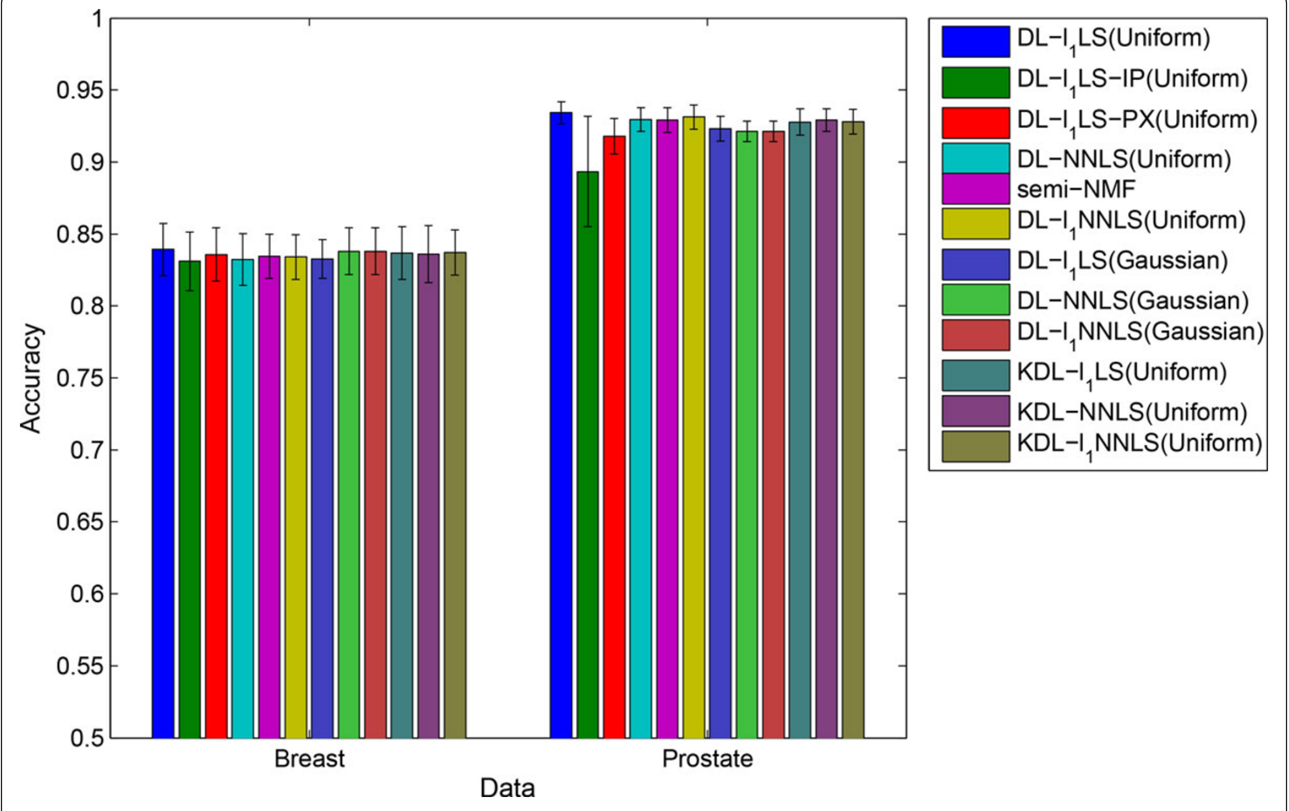

Figure 3 Mean accuracies and standard deviations of dictionary learning methods.

the accuracy could be further improved by a suitably selected kernel.

We compare the mean computing time of all the feature extraction methods in Figure 4. First, we can see that DL-l_1LS using active-set algorithm is much more efficient than DL-l_1LS-IP, DL-l_1LS-PX, and semi-NMF using multiplicative update rules. Second, the non-negative dictionary learning models are more efficient than the l_1-regularized models. Therefore as in the sparse coding method, priority should be given to the non-negative models when attempting to use dictionary learning in an application.

Conclusions

In this study, l_1-regularized and non-negative sparse representation models are comprehensively studied for the classification of high-dimensional biological data. We give a Bayesian treatment to the models. We prove that the sparse coding and dictionary learning models are in fact equivalent to MAP estimations. We use different priors on the sparse coefficient vector and the dictionary atoms, which lead to various sparse representation models. We propose parallel active-set algorithms to optimize the sparse coding models, and propose a generic framework for dictionary learning. We reveal that the sparse representation models only use inner products of instances. Using this dimension-free property, we can easily extend these models to kernel versions. With the comparison with existing models for high-dimensional data, it is shown that our techniques are very efficient. Furthermore, our approaches obtained comparable or higher accuracies. In order to promote the research of sparse representation in bioinformatics, the MATLAB implementation of the sparse representation methods discussed in this paper can be downloaded at [27].

Our Bayesian treatment may inspire the readers to try other prior distributions in order to design new sparse representation models. It also helps to discover the similarity and difference between sparse representation and other dimension reduction techniques. Our kernel versions can also be used to classify tensor data where an observation is not a vector but a matrix or tensor [28]. They can also be applied in the biomedical text mining and interaction/relational data where only the similarities between instances are known.

We will apply our technique to other high-throughput data, such as microarray epigenomic data, gene copy number profiles, and sequence data. We will impose both sparse-inducing prior on dictionary atoms and coefficients. Inspired by Bayesian factor analysis, we will investigate the variable selection methods using sparse dictionary. The

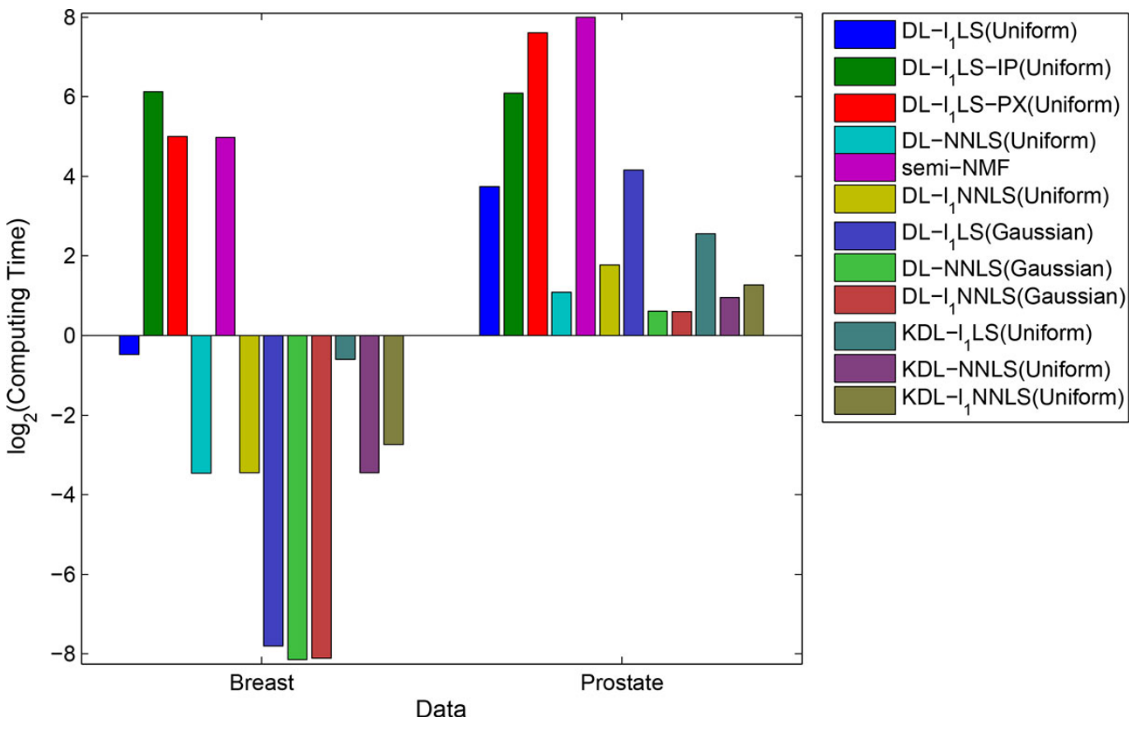

Figure 4 Log2 computing time of dictionary learning method.

sparse dictionary analysis would help us to uncover the biological patterns hidden in the high-dimensional biological data. Furthermore, combining Bayesian sparse representation and Bayesian regression leads to Bayesian sparse representation regression model, which is very helpful for designing supervised dictionary learning. Finally we should mention that we are working on a decomposition method for sparse coding which is efficient on large-scale biological data where there are at least thousands of samples.

Competing interests
The authors declare that they have no competing interests.

Authors' contributions
YL proposed the original idea, did literature survey, implemented the methods, conducted the experiments, and wrote the first draft of this paper. AN supervised the whole research, gave constructive suggestions, and finished the final version of this paper.

Acknowledgements
This article is based on "Fast Sparse Representation Approaches for the Classification of High-Dimensional Biological Data", by Yifeng Li and Alioune Ngom which appeared in Bioinformatics and Biomedicine (BIBM), 2012 IEEE International Conference on. © 2012 IEEE [10]. This research has been partially supported by IEEE CIS Walter Karplus Summer Research Grant 2010, Ontario Graduate Scholarship 2011-2013, and The Natural Sciences and Engineering Research Council of Canada (NSERC) Grants #RGPIN228117-2011.

Declarations
The publication costs for this article were funded by Dr. Alioune Ngom with his Natural Sciences and Engineering Research Council of Canada (NSERC) Grants #RGPIN228117-2011.

This article has been published as part of *BMC Systems Biology* Volume 7 Supplement 4, 2013: Selected articles from the IEEE International Conference on Bioinformatics and Biomedicine 2012: Systems Biology. The full contents of the supplement are available online at http://www.biomedcentral.com/bmcsystbiol/supplements/7/S4.

Published: 23 October 2013

References
1. Furey T, Cristianini N, Duffy N, Bednarski D, Schummer M, Haussler D: **Support vector machine classification and validation of cancer tissue samples using microarray expression data.** *Bioinformatics* 2000, **16(10)**:906-914.
2. Li Y, Ngom A: **The regularized linear models and kernels toolbox in MATLAB.** *Tech. rep., School of Computer Science* University of Windsor, Windsor, Ontario; 2013 [https://sites.google.com/site/rlmktool].
3. Wall M, Rechtsteiner A, Rocha L: **Singular value decomposition and principal component analysis.** In *A Practical Approach to Microarray Data Analysis.* Kluwer;Berrar D, Dubitzky W, Granzow M, Norwell, MA 2003:91-109.
4. Carvalho C, Chang J, Lucas J, Nevins J, Wang Q, West M: **High-dimensional sparse factor modeling: applications in gene expression genomics.** *Journal of the American Statistical Association* 2008, **103(484)**:1438-1456.
5. Elad M: *Sparse and Redundant Representations: From Theory to Applications in Signal and Image Processing* New York: Springer; 2010.
6. Bengio Y, Courville A, Vincent P: **Representation learning: a review and new perspectives.** *Arxiv* 2012, 1206.5538v2.
7. Elad M, Aharon M: **Image denoising via learned dictionaries and sparse representation.** *CVPR, IEEE Computer Society* Washington DC: IEEE; 2006, 895-900.
8. Li Y, Ngom A: **The non-negative matrix factorization toolbox for biological data mining.** *BMC Source Code for Biology and Medicine* 2013, **8**:10.
9. Lee DD, Seung S: **Learning the parts of objects by non-negative matrix factorization.** *Nature* 1999, **401**:788-791.
10. Li Y, Ngom A: **Fast sparse representation approaches for the classification of high-dimensional biological data.** *Bioinformatics and*

Biomedicine (BIBM), 2012 IEEE International Conference on: 4-7 October 2012 2012, 1-6.

11. Hang X, Wu FX: **Sparse representation for classification of tumors using gene expression data.** *J. Biomedicine and Biotechnology* 2009, **2009**:ID 403689.
12. Rowe D: *Multivariate Bayesian Statistics: Models for Source Separation and Signal Unmixing* Boca Raton, FL: CRC Press; 2003.
13. West M: **Bayesian factor regression models in the "large p, small n" paradigm.** *Bayesian Statistics* 2003, **7**:723-732.
14. Bruckstein AM, Donoho DL, Elad M: **From sparse solutions of systems of equations to sparse modeling of signals and images.** *SIAM Review* 2009, **51**:34-81.
15. Tibshirani R: **Regression shrinkage and selection via the lasso.** *Journal of the Royal Statistical Society. Series B (Methodological)* 1996, **58**:267-288.
16. Yin J, Liu X, Jin Z, Yang W: **Kernel sparse representation based classification.** *Neurocomputing* 2012, **77**:120-128.
17. Gao S, Tsang IWH, Chia LT: **Kernel sparse representation for image classification and face recognition.** *ECCV* Springer; 2010, 1-14.
18. Olshausen B, Field D: **Sparse coding with an overcomplete basis set: a strategy employed by V1?** *Vision Research* 1997, **37(23)**:3311-3325.
19. Wright J, Yang A, Ganesh A, Sastry SS, Ma Y: **Robust face recognition via sparse representation.** *TPAMI* 2009, **31(2)**:210-227.
20. Nocedal J, Wright SJ: *Numerical Optimization.* 2 edition. New York: Springer; 2006.
21. Lawson CL, Hanson RJ: *Solving Least Squares Problems* Piladelphia: SIAM; 1995.
22. Kim SJ, Koh K, Lustig M, Boyd S, Gorinevsky D: **An interior-point method for large-scale *l*1-regularized least squares.** *IEEE J. Selected Topics in Signal Processing* 2007, **1(4)**:606-617.
23. Jenatton R, Mairal J, Obozinski G, Bach F: **Proximal methods for hierarchical sparse coding.** *JMLR* 2011, **12(2011)**:2297-2334.
24. Hu Z, *et al*: **The molecular portraits of breast tumors are conserved across microarray platforms.** *BMC Genomics* 2006, **7**:96.
25. Petricoin EI, *et al*: **Serum proteomic patterns for detection of prostate cancer.** *J. National Cancer Institute* 2002, **94(20)**:1576-1578.
26. Ding C, Li T, Jordan MI: **Convex and semi-nonnegative matrix factorizations.** *TPAMI* 2010, **32**:45-55.
27. **The sparse representation toolbox in MATLAB.** [https://sites.google.com/site/sparsereptool].
28. Li Y, Ngom A: **Non-negative matrix and tensor factorization based classification of clinical microarray gene expression data.** *Bioinformatics and Biomedicine (BIBM), 2010 IEEE International Conference on: 18-21 December 2010* 2010, 438-443.

doi:10.1186/1752-0509-7-S4-S6
Cite this article as: Li and Ngom: Sparse representation approaches for the classification of high-dimensional biological data. *BMC Systems Biology* 2013 **7**(Suppl 4):S6.