2014 International Joint Conference on Neural Networks (IJCNN)
July 6-11, 2014, Beijing, China

A Decomposition Method for Large-Scale Sparse Coding in Representation Learning

Yifeng Li
CMMT, Child and Family Research Institute
University of British Columbia
Vancouver, British Columbia, Canada
Email: yifeng@cmmt.ubc.ca

Richard J. Caron
Mathematics and Statistics
University of Windsor
Windsor, Ontario, Canada
Email: rcaron@uwindsor.ca

Alioune Ngom
School of Computer Science
University of Windsor
Windsor, Ontario, Canada
Email: angom@uwindsor.ca

***Abstract*—In representation learning, sparse representation is a parsimonious principle that a sample can be approximated by a sparse superposition of dictionary atoms. Sparse coding is the core of this technique. Since the dictionary is often redundant, the dictionary size can be very large. Many optimization methods have been proposed in the literature for sparse coding. However, the efficiency of the optimization for a tremendous number of dictionary atoms is still a bottleneck. In this paper, we propose to use decomposition method for large-scale sparse coding models. Our experimental results show that our method is very efficient.**

I. Introduction

Sparse representation is an important topic in representation learning [1]. *Sparse representation* (SR) is a principle that a multivariate sample can be approximated by a sparse linear combination of dictionary atoms [2], [3]. It can be formulated as $\boldsymbol{b} = x_1\boldsymbol{a}_1 + \cdots + x_k\boldsymbol{a}_k + \boldsymbol{\epsilon} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{\epsilon}$, where $\boldsymbol{A} = [\boldsymbol{a}_1, \cdots, \boldsymbol{a}_k]$ is called a *dictionary*, $\boldsymbol{a}_i$ is called a *dictionary atom* or *basis vector*, $\boldsymbol{x}$ is a sparse *coefficient vector*, and $\boldsymbol{\epsilon}$ is an error term. Sparse representation involves sparse coding and dictionary learning. Given a new signal $\boldsymbol{b}$ and dictionary $\boldsymbol{A}$, learning the sparse coefficient $\boldsymbol{x}$ is termed *sparse coding*. Given training data $\boldsymbol{D}$, learning the dictionary $\boldsymbol{A}$ is called *dictionary learning*. In the last decade, sparse representation has been successfully applied in signal processing [2], computer vision [4], machine learning [5], and bioinformatics [6].

The l_1-norm is among the most popular sparsity-inducing term in sparse coding. The l_1-regularized sparse coding model can be expressed as

$$\min_{\boldsymbol{x}} f(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}\|_2^2 + \lambda\|\boldsymbol{x}\|_1, \tag{1}$$

where $\lambda \geq 0$ is a pre-specified parameter to balance the trade-off between reconstructive error and sparsity. This model is called l_1-*least-squares* (l_1LS) sparse coding. From the Bayesian perspective, Equation (1) is a maximum *a posteriori* (MAP) estimation where the error is Gaussian and the prior of $\boldsymbol{x}$ is Laplacian. As has been shown in [7], [8], [5], non-negativity can also induce sparse coefficient. The non-negative sparse coding model can be rewritten as

$$\min_{\boldsymbol{x}} f(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}\|_2^2 \quad \text{subject to } \boldsymbol{x} \geq 0. \tag{2}$$

This model is called *non-negative least squares* (NNLS) sparse coding. It corresponds to the MAP estimation with Gaussian error and Bernoulli prior. The combination of the l_1-regularization and the non-negativity leads to the following l_1NNLS model:

$$\min_{\boldsymbol{x}} f(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}\|_2^2 + \lambda\|\boldsymbol{x}\|_1 \quad \text{subject to } \boldsymbol{x} \geq 0. \tag{3}$$

Dictionary learning is an alternating sparse coding problem in essence. Therefore, the optimization of sparse coding is the core task in sparse representation learning. Many optimization methods have been proposed in the literature for sparse coding. The l_1LS model is convex but non-smooth, it is thus challenging to optimize given a large-scale dictionary. The interior-point method [9] and proximal method [10] are two typical algorithms for l_1LS sparse coding. Active-set methods have also been proposed for l_1LS, NNLS, and l_1NNLS [11], [5], and other variants [12]. These methods are briefly reviewed in the next section. Most of them are inefficient for a very big dictionary.

In this paper, we propose decomposition methods for the l_1LS and NNLS sparse coding models. The contributions of this paper include:

1) We investigate the idea of decomposition in sparse coding. The decomposition method has been successfully applied in the optimization of support vector machine (SVM).
2) We design sequential minimal optimization (SMO) methods for l_1LS, NNLS, and l_1NNLS sparse coding models. This idea can be easily generalized for many other sparse models.

The rest of this paper is organized as follows. In Section II, we reformulate the sparse coding models into quadratic programmes, and reveal that the optimization of these models are dimension-free. In the same section, we then concisely survey the existing optimization algorithms. Our

978-1-4799-1484-5/14/$31.00 ©2014 IEEE

methods are described in Section III. The computational experiment is shown in Section IV. Finally, we draw our conclusions and mention future works.

II. The Quadratic Programming Formulations and Related Works

A. The Optimizations Are Quadratic Programmes and Dimension-Free

The least squares problems in Equations (1), (2), and (3) are, in essence, *quadratic programming* (QP) problems. The l_1LS problem in Equation (1) can be formulated into the following l_1QP problem:

$$\min_{\boldsymbol{x}} f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^{\mathrm{T}}\boldsymbol{H}\boldsymbol{x} + \boldsymbol{g}^{\mathrm{T}}\boldsymbol{x} + \lambda\|\boldsymbol{x}\|_1, \tag{4}$$

where $\boldsymbol{H} = \boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}$, and $\boldsymbol{g} = -\boldsymbol{A}^{\mathrm{T}}\boldsymbol{b}$. Hereafter, we call the l_1LS model as l_1QP model. This non-smooth l_1QP problem can be further converted to the following problem that is smooth but constrained:

$$\begin{aligned} \min_{\boldsymbol{x},\boldsymbol{u}} f(\boldsymbol{x},\boldsymbol{u}) &= \frac{1}{2}\boldsymbol{x}^{\mathrm{T}}\boldsymbol{H}\boldsymbol{x} + \boldsymbol{g}^{\mathrm{T}}\boldsymbol{x} + \boldsymbol{\lambda}^{\mathrm{T}}\boldsymbol{u} \\ \text{subject to } & -\boldsymbol{u} \le \boldsymbol{x} \le \boldsymbol{u}, \end{aligned} \tag{5}$$

where $\boldsymbol{u}$ is an auxiliary variable, and $\boldsymbol{\lambda} = \{\lambda\}^k$. The NNLS problem in Equation (2) and l_1NNLS problem in Equation (3) can be reformulated into the following NNQP problem:

$$\begin{aligned} \min_{\boldsymbol{x}} f(\boldsymbol{x}) &= \frac{1}{2}\boldsymbol{x}^{\mathrm{T}}\boldsymbol{H}\boldsymbol{x} + \boldsymbol{g}^{\mathrm{T}}\boldsymbol{x} \\ \text{subject to } & \boldsymbol{x} \ge 0, \end{aligned} \tag{6}$$

where $\boldsymbol{H} = \boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}$, and $\boldsymbol{g} = -\boldsymbol{A}^{\mathrm{T}}\boldsymbol{b}$ for NNLS and $\boldsymbol{g} = \lambda - \boldsymbol{A}^{\mathrm{T}}\boldsymbol{b}$ for l_1NNLS. Hereafter, we call the NNLS and l_1NNLS models as NNQP models. The QP formulations above have two important advantages which are given in the following. First, we can clearly see that the optimization only requires the inner products $\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}$ and $\boldsymbol{A}^{\mathrm{T}}\boldsymbol{b}$. Therefore, the optimization is in fact *dimension-free*, that is the number of dimensions of the dictionary atoms does not affect the computation directly. Through replacing the inner products by kernels $k(\boldsymbol{A},\boldsymbol{A})$ and $k(\boldsymbol{A},\boldsymbol{b})$, we can easily obtain the *kernel sparse coding*. Second, by adding the term $\frac{\alpha}{2}\|\boldsymbol{x}\|_2^2$ in the objectives of Equations (1) and (3), we can obtain the elastic-net regularized sparse coding [13], which has important application for correlated variable selection. The Hessian in the QP formulations in Equations (4) and (6) now simply becomes $\boldsymbol{H} = \boldsymbol{A}^{\mathrm{T}}\boldsymbol{A} + \alpha\boldsymbol{I}$ where $\boldsymbol{I}$ is an identity matrix. Due to page limit, we do not discuss the second strength in this paper.

In the following, we briefly review three important existing works which solve either the least squares formulations or the QP formulations. If the original works solve least squares problems, we reformulate them into QP problems, so that we can discuss all the methods conveniently and consistently.

B. Interior-Point Method

The log-barrier interior-point method for the l_1LS problem (Equation (1)) is proposed in [9]. The basic idea is to convert a non-smooth problem into an unconstrained one. For the convenience of discussion, we extend this method to solve the l_1QP and NNQP problems. The interior-point method for l_1QP is introduced in the following. Due to similarity and page limit, we omit the interior-point method for NNQP problem. The problem in Equation (4) can be transformed into minimizing the unconstrained log-barrier function:

$$f(\boldsymbol{x},\boldsymbol{u}) = t(\frac{1}{2}\boldsymbol{x}^{\mathrm{T}}\boldsymbol{H}\boldsymbol{x} + \boldsymbol{g}^{\mathrm{T}}\boldsymbol{x} + \boldsymbol{\lambda}^{\mathrm{T}}\boldsymbol{u}) - \sum_{i=1}^{k}\log(u_i^2 - x_i^2). \tag{7}$$

We can obtain its gradient as below

$$\boldsymbol{d} = \begin{bmatrix} (\frac{\partial f(\boldsymbol{x},\boldsymbol{u})}{\partial \boldsymbol{x}})^T \\ (\frac{\partial f(\boldsymbol{x},\boldsymbol{u})}{\partial \boldsymbol{u}})^T \end{bmatrix} = \begin{bmatrix} t(\boldsymbol{H}\boldsymbol{x} + \boldsymbol{g}) + 2[\frac{x_1}{u_1^2 - x_1^2} \cdots \frac{x_k}{u_k^2 - x_k^2}]^{\mathrm{T}} \\ t\boldsymbol{\lambda} - 2[\frac{u_1}{u_1^2 - x_1^2} \cdots \frac{u_k}{u_k^2 - x_k^2}]^{\mathrm{T}} \end{bmatrix}.$$

And its Hessian matrix is

$$\boldsymbol{H} = \begin{bmatrix} \frac{\partial f^2(\boldsymbol{x},\boldsymbol{u})}{\partial \boldsymbol{x}^2} & \frac{\partial f^2(\boldsymbol{x},\boldsymbol{u})}{\partial \boldsymbol{u}\partial \boldsymbol{x}} \\ \frac{\partial f^2(\boldsymbol{x},\boldsymbol{u})}{\partial \boldsymbol{x}\partial \boldsymbol{u}} & \frac{\partial f^2(\boldsymbol{x},\boldsymbol{u})}{\partial \boldsymbol{u}^2} \end{bmatrix},$$

where

$$\frac{\partial f^2(\boldsymbol{x},\boldsymbol{u})}{\partial \boldsymbol{x}^2} = t\boldsymbol{H} + 2\mathrm{diag}(\frac{u_1^2 + x_1^2}{(u_1^2 - x_1^2)^2} \cdots \frac{u_k^2 + x_k^2}{(u_k^2 - x_k^2)^2}),$$

$$\frac{\partial f^2(\boldsymbol{x},\boldsymbol{u})}{\partial \boldsymbol{u}^2} = 2\mathrm{diag}(\frac{u_1^2 + x_1^2}{(u_1^2 - x_1^2)^2} \cdots \frac{u_k^2 + x_k^2}{(u_k^2 - x_k^2)^2}),$$

and

$$\frac{\partial f^2(\boldsymbol{x},\boldsymbol{u})}{\partial \boldsymbol{x}\partial \boldsymbol{u}} = -4\mathrm{diag}(\frac{x_1 u_1}{(u_1^2 - x_1^2)^2} \cdots \frac{x_k u_k}{(u_k^2 - x_k^2)^2}).$$

Newton's method (a well-known second-order algorithm) can be applied to minimize the unconstrained log-barrier function given positive t [14]. As t increases to positive infinite, the solution converge to the optimal solution. Given a value of t, the convergence rate of Newton's method is super-linear or quadratic with heavy steps.

C. Proximal Method

As one of fast first-order methods, proximal method was proposed in [10] to solve the l_1LS problem. Compared with the (second-order) interior-point method as derived above, it has two advantages. First, Hessian is not recomputed in each step. Second, analytical solution can be obtained in each step. As in the interior-point method, we extend the proximal method for the l_1QP problem for the convenience of discussion. The main idea is in the following. The proximal method works in an iterative procedure. In each iteration, given the current estimate $\hat{\boldsymbol{x}}$, the new estimate

is the solution to the following first-order Taylor series of Equation (4):

$$\min_{\boldsymbol{x}} f(\hat{\boldsymbol{x}}) + \nabla q(\hat{\boldsymbol{x}})(\boldsymbol{x} - \hat{\boldsymbol{x}}) + \lambda\|\boldsymbol{x}\|_1 + \frac{L}{2}\|\boldsymbol{x} - \hat{\boldsymbol{x}}\|_2^2, \quad (8)$$

where $q(\hat{\boldsymbol{x}}) = \frac{1}{2}\hat{\boldsymbol{x}}^{\mathrm{T}}\boldsymbol{H}\hat{\boldsymbol{x}} + \boldsymbol{g}^{\mathrm{T}}\hat{\boldsymbol{x}}$ and $L > 0$ is a parameter. This is done iteratively until termination criteria are met. Equation (8) is equivalent to the following proximal operator:

$$\min_{\boldsymbol{x}} \frac{1}{2}\|\boldsymbol{x} - \bar{\boldsymbol{x}}\|_2^2 + \eta\|\boldsymbol{x}\|_1, \quad (9)$$

where $\bar{\boldsymbol{x}} = \hat{\boldsymbol{x}} - \frac{1}{L}\nabla q(\hat{\boldsymbol{x}})$, and $\eta = \frac{\lambda}{L}$. Unlike interior-point method with heavy Newton step in each iteration, it has an element-wise analytical solution: $x_i = \begin{cases} \text{sign}(\bar{x}_i)(|\bar{x}_i| - \eta) & \text{if } |\bar{x}_i| > \eta \\ 0 & \text{otherwise} \end{cases}$. The proximal method for l_1QP has *linear* convergence rate with swift steps.

D. Active-Set Method

Active-set algorithms have been used, in [11], to solve the l_1QP and NNQP problems. The basis idea of active-set algorithms is in the following. A working set is first initialized. Then variables are added to or deleted from the working set in an iterative procedure until the optimality conditions are satisfied. In each iteration, a system of linear equations, corresponding to the working set, need to be solved. When solving the sparse coding of multiple new samples, one advantage of the active-set methods are that their common system of linear equations can be solved once only, and can be shared among the multiple sparse coding procedures. It makes active-set methods very suitable to solve multiple sparse coding as well as the corresponding dictionary learning problems. Active-set methods are usually very efficient for small and medium-scale problems, because they usually have linear convergence rate, and solving the system of linear equations of a small number of variables is not expensive. The worst case of active-set algorithms is exponential, however this case is very rare. For example, it is known that the Simplex algorithm for linear programming is an active-set method. Although the worst case of Simplex is exponential, it is usually the fastest algorithm for small and median-scale linear programming.

III. Method

In this section, we give our decomposition method for l_1QP and NNQP respectively. Decomposition method [15] is first devised in the optimization of large-scale SVM which is a QP problem constrained by equality and bound constraints. The basic idea of the decomposition method is, in fact, an implementation of the block-coordinate-descent scheme [16]. The decomposition method works in an iterative procedure. In each iteration, a few number of variables violating optimality conditions (e.g. *Karush-Kuhn-Tucker* (KKT) conditions) are included in a working set, while the rest are fixed. Only the variables in the working set are updated by a solver. This procedure iterates until no coefficient violates the KKT conditions. Because the objective is decreased in each iteration, the convergence is guaranteed. *Sequential minimal optimization* (SMO) [17] is the extreme case of the decomposition method for SVM, where only the minimal number of variables (two variables) are updated. One of the features of the SMO method is that the sub-problem with two only variables can be solved analytically.

A. Decomposition Method for l_1QP

Let $\mathcal{A}$ be the set of a few working variables, and $\mathcal{P}$ the set of fixed variables. The decomposition of $f(\boldsymbol{x})$ in Equation (4) can be decomposed as

$$\begin{aligned} f(\boldsymbol{x}_{\mathcal{A}}) = &\frac{1}{2}[\boldsymbol{x}_{\mathcal{A}}^{\mathrm{T}}, \boldsymbol{x}_{\mathcal{P}}^{\mathrm{T}}] \begin{bmatrix} \boldsymbol{H}_{\mathcal{A}\mathcal{A}} & \boldsymbol{H}_{\mathcal{A}\mathcal{P}} \\ \boldsymbol{H}_{\mathcal{P}\mathcal{A}} & \boldsymbol{H}_{\mathcal{P}\mathcal{P}} \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_{\mathcal{A}} \\ \boldsymbol{x}_{\mathcal{P}} \end{bmatrix} \\ &+ [\boldsymbol{g}_{\mathcal{A}}^{\mathrm{T}}, \boldsymbol{g}_{\mathcal{P}}^{\mathrm{T}}] \begin{bmatrix} \boldsymbol{x}_{\mathcal{A}} \\ \boldsymbol{x}_{\mathcal{P}} \end{bmatrix} + \lambda\| \begin{bmatrix} \boldsymbol{x}_{\mathcal{A}} \\ \boldsymbol{x}_{\mathcal{P}} \end{bmatrix} \|_1 \quad (10) \\ = &\frac{1}{2}\boldsymbol{x}_{\mathcal{A}}^{\mathrm{T}}\boldsymbol{H}_{\mathcal{A}\mathcal{A}}\boldsymbol{x}_{\mathcal{A}} + (\boldsymbol{H}_{\mathcal{A}\mathcal{P}}\boldsymbol{x}_{\mathcal{P}} + \boldsymbol{g}_{\mathcal{A}})^{\mathrm{T}}\boldsymbol{x}_{\mathcal{A}} \\ &+ \lambda\|\boldsymbol{x}_{\mathcal{A}}\|_1 + \text{constant}. \quad (11) \end{aligned}$$

We can see that the sub-problem corresponding to the working set is a l_1QP problem as well. From this point, a solver of l_1QP is still required. In this paper, we focus on the extreme case of Equation (11): the SMO method. It can be easily seen that the minimal size of $\mathcal{A}$ is *one* and $\mathcal{P}$ contains the reminder $k - 1$ variables. Without loss of generality, we denote such variable by x_1. This results in the following sub-problem:

$$\min_{x_1} f(x_1) = \frac{1}{2}h_{11}x_1^2 + b_1x_1 + \lambda|x_1|, \quad (12)$$

where $b_1 = \boldsymbol{H}_{1\mathcal{P}}\boldsymbol{x}_{\mathcal{P}} + g_1$. This SMO method is a coordinate descent method [16] in essence.

1) Analytical Solution: The SMO problem in Equation (12) can be solved analytically. Now let's separate the interval into $x_1 \geq 0$ and $x_1 \leq 0$. For $x_1 \geq 0$, the objective $f(x_1)$ becomes

$$f(x_1) = \frac{1}{2}h_{11}x_1^2 + (b_1 + \lambda)x_1. \quad (13)$$

Taking its first-order derivative and setting it to zero, we have

$$\frac{\partial f(x_1)}{\partial x_1} = h_{11}x_1 + (b_1 + \lambda) = 0. \quad (14)$$

We thus have $x_1^{(+)} = \frac{-b_1 - \lambda}{h_{11}}$. Therefore, for interval $x_1 \geq 0$, we have the optimal solution:

$$x_1^{(+*)} = \begin{cases} x_1^{(+)} & \text{if } x_1^{(+)} \geq 0 \\ 0 & \text{otherwise} \end{cases}. \quad (15)$$

Similarly, for $x_1 \leq 0$, the objective $f(x_1)$ becomes

$$f(x_1) = \frac{1}{2}h_{11}x_1^2 + (b_1 - \lambda)x_1. \tag{16}$$

We thus have $x_1^{(-)} = \frac{-b_1+\lambda}{h_{11}}$. And for interval $x_1 \leq 0$, we have the optimal solution:

$$x_1^{(-*)} = \begin{cases} x_1^{(-)} & \text{if } x_1^{(-)} \leq 0 \\ 0 & \text{otherwise} \end{cases}. \tag{17}$$

By considering both positive and negative interval together, the optimal solution to Equation (12) is the one, among $x_1^{(+*)}$ and $x_1^{(-*)}$, which obtains the minimum objective value, that is $x_1^* = \arg\min_{\{x_1^{(+*)}, x_1^{(-*)}\}} f(x_1)$.

We note that $x_1^{(+)} \geq 0$ is equivalent to $b_1 \leq -\lambda$, and $x_1^{(-)} \leq 0$ is equivalent to $b_1 \geq \lambda$. We can state that if $b_1 \leq -\lambda$ or $b_1 \geq \lambda$ the solution to Equation (12) is $x_1^* = \frac{-b_1-\lambda}{h_{11}}$ or $x_1^* = \frac{-b_1+\lambda}{h_{11}}$, respectively. Otherwise, $x_1^* = 0$. Therefore, the solution to Equation (12) can be equivalently written as

$$x_1^* = \begin{cases} \frac{-\text{sign}(b_1)(|b_1|-\lambda)}{h_{11}} & \text{if } |b_1| \geq \lambda \\ 0 & \text{otherwise} \end{cases}. \tag{18}$$

From this, we can obtain a general proposition below which is very useful:

Proposition 1. *The solution to the following problem*

$$\min_x f(x) = x^2 + bx + \lambda|x| \tag{19}$$

is analytically

$$x^* = \begin{cases} -sign(b)(|b| - \lambda) & \textit{if } |b| \geq \lambda \\ 0 & \textit{otherwise} \end{cases}. \tag{20}$$

2) Select x_1 Which Violates The Optimality Condition: The optimality condition of Equation (4) is

$$\boldsymbol{x}^{\mathrm{T}}\boldsymbol{H} + \boldsymbol{g}^{\mathrm{T}} + \nabla\lambda\|\boldsymbol{x}\|_1 = 0. \tag{21}$$

However because $\lambda\|\boldsymbol{x}\|_1$ is not differentiable, we need to resort to the concept of subgradient. We hence have

$$s_i = \boldsymbol{H}_{i:}\boldsymbol{x} + g_i = \begin{cases} \lambda & x_i < 0 \\ -\lambda & x_i > 0 \\ \in [-\lambda, \lambda] & x_i = 0 \end{cases}, \tag{22}$$

where $\boldsymbol{H}_{i:}$ is the i-th row of $\boldsymbol{H}$, similarly $\boldsymbol{H}_{:i}$ is the i-th column of $\boldsymbol{H}$. The algorithm is in the following. We iteratively select a variable x_1 which violates the optimality condition in Equation (22). In an iteration, x_1 is updated analytically as in Equation (18). If all variables satisfy the optimality condition, the algorithm terminates. In order to maximize the effort of violating variable selection, the one which violates the optimality condition the most should be preferred. The extent of violation is measured by the difference, as given below, between s_i and its desired values:

$$d_i = \begin{cases} |s_i - \lambda| & x_i < 0 \\ |s_i + \lambda| & x_i > 0 \\ |s_i| - \lambda & x_i = 0 \end{cases}. \tag{23}$$

3) Update $\boldsymbol{s}$ and Compute b_1: In each iteration after updating x_1, the vector $\boldsymbol{s}$ (defined in Equation 22) should be updated in order to obtain the optimality condition and select the new violating variable x_1. Intuitively, $\boldsymbol{s}$ can be updated by its definition in Equation (22). However, it would take linear time to update each element s_i. In fact, if we keep a record of its previous value (denoted by s_i'), s_i can be updated in constant time. We denote $\boldsymbol{x}'$ be the coefficients before updating x_1, and $\boldsymbol{x}$ be the coefficients after updating x_1. We know that $s_i' = h_{i1}x_1' + \boldsymbol{H}_{i\mathcal{P}}\boldsymbol{x}_{\mathcal{P}}' + g_i$, $s_i = h_{i1}x_1 + \boldsymbol{H}_{i\mathcal{P}}\boldsymbol{x}_{\mathcal{P}} + g_i$, and $\boldsymbol{H}_{i\mathcal{P}}\boldsymbol{x}_{\mathcal{P}}' = \boldsymbol{H}_{i\mathcal{P}}\boldsymbol{x}_{\mathcal{P}}$. We thus can update s_i by the following equation which takes constant time:

$$s_i = h_{i1}(x_1 - x_1') + s_i'. \tag{24}$$

Similar idea also applies to the computation of b_1 before updating x_1. According to the definition in Equation (12), b_1 can be updated in linear time. However, it can actually be updated in constant time as well. We know that $s_1' = h_{11}x_1' + \boldsymbol{H}_{1\mathcal{P}}\boldsymbol{x}_{\mathcal{P}}' + g_1 = h_{11}x' + b_1$. We thus can update b_1 in constant time:

$$b_1 = s_1' - h_{11}x'. \tag{25}$$

4) Initialize $\boldsymbol{x}$ and $\boldsymbol{s}$: Before the iterative update of the method, $\boldsymbol{x}$ and $\boldsymbol{s}$ have to be initialized. We can initialize $\boldsymbol{x}$ and $\boldsymbol{s}$ by zeros and $\boldsymbol{g}$, respectively. This initialization makes the following iterative update very efficient. This is because $\boldsymbol{x}$ is eventually sparse, which means that $\boldsymbol{x} = \boldsymbol{0}$ is a very good approximate. This initialization strategy may also apply to many other sparse coding methods, for example active-set methods.

B. Decomposition Method for NNQP

Now we concisely derive the decomposition method for NNQP. The decomposed objective of NNQP can be formulated into

$$f(\boldsymbol{x}_{\mathcal{A}}) = \frac{1}{2}\boldsymbol{x}_{\mathcal{A}}^{\mathrm{T}}\boldsymbol{H}_{\mathcal{A}\mathcal{A}}\boldsymbol{x}_{\mathcal{A}} + (\boldsymbol{H}_{\mathcal{A}\mathcal{P}}\boldsymbol{x}_{\mathcal{P}} + \boldsymbol{g}_{\mathcal{A}})^{\mathrm{T}}\boldsymbol{x}_{\mathcal{A}} + \text{constant}. \tag{26}$$

Since this objective is constrained only by nonnegativity, its SMO case also has only *one* variable in $\mathcal{A}$. Without loss of generality, we denote such variable as x_1 as above. This results in the following problem:

$$\min_{x_1} f(x_1) = \frac{1}{2}h_{11}x_1^2 + b_1x_1 \tag{27}$$
$$\text{subject to } x_1 \geq 0,$$

where $b_1 = \boldsymbol{H}_{1\mathcal{P}}\boldsymbol{x}_{\mathcal{P}} + g_1$. It is easy to obtain the analytical solution to Equation (27):

$$x_1^* = \begin{cases} \frac{-b_1}{h_{11}} & \text{if } b_1 \leq 0 \\ 0 & \text{otherwise} \end{cases}. \tag{28}$$

The Lagrangian function of Equation (6) is

$$L(\boldsymbol{x}, \boldsymbol{s}) = \frac{1}{2}\boldsymbol{x}^{\mathrm{T}}\boldsymbol{H}\boldsymbol{x} + \boldsymbol{g}^{\mathrm{T}}\boldsymbol{x} - \boldsymbol{s}^{\mathrm{T}}\boldsymbol{x}, \tag{29}$$

where $\boldsymbol{s}$ is the vector of dual variables. Therefore, the corresponding KKT conditions of Equation (6) are

$$\begin{cases} \boldsymbol{s} = \boldsymbol{H}\boldsymbol{x} + \boldsymbol{g} \\ \boldsymbol{s} * \boldsymbol{x} = 0 \\ \boldsymbol{x} \geq 0 \\ \boldsymbol{s} \geq 0. \end{cases} \tag{30}$$

From the KKT conditions, we can find that $s_i = \boldsymbol{H}_{i:}\boldsymbol{x} + g_i$ is the Lagrangian multiplier of the i-th primal variable, x_i. For the optimal x_i, the corresponding s_i must fulfil the following conditions:

$$\begin{cases} s_i * x_i = 0 \\ s_i \geq 0 \end{cases}. \tag{31}$$

The SMO-based NNQP works in an iterative procedure. Before the iterative loop, $\boldsymbol{x}$ can initialized by zeros, and $\boldsymbol{s}$ is therefore initialized by $\boldsymbol{g}$. After that, variables violating the KKT conditions in Equation (31) are updated in an iterative loop until all variables fulfil the KKT conditions. In each iteration, we need to find one variable, which violates the KKT condition, to update using Equation (28); before updating x_1, b_1 can be computed by Equation (25); after updating x_1, $\boldsymbol{s}$ can be updated as in Equation (24). In our implementation, we select the variable which violates the KKT conditions the most. The magnitude of violation is measured by the following equation:

$$d_i = \begin{cases} 0 & \text{if } x_i = 0 \text{ and } s_i \geq 0 \\ |s_i| & \text{if } x_i = 0 \text{ and } s_i < 0 \\ |s_i| & \text{if } x_i > 0 \end{cases}. \tag{32}$$

IV. Computational Experiments

We tested our SMO methods on the large-scale microarray meta-analysis data given in [18]. This data set has 5456 samples including healthy tissue samples and cancer samples from 13 different tissues (thus 26 classes in total). The number of dimensions (genes) of each sample is 9471. We randomly selected 100 samples as test set, and let the rest 5356 samples as training set. In few cases where the computation was very costly, we only used 10 test samples. We put all the training samples in the dictionary $\boldsymbol{A}_{9471\times 5356}$. Therefore, the Hessian is of size 5356 by 5356. We used linear kernel for all methods in our experiments (other kernels can, of course, be used consistently for all methods). We compared our SMO methods with proximal method and active-set methods. Due to the large numbers of dictionary size and dimensions (the implementation in [9] is not dimension-free), the interior-point method proposed in [9] crashed our computer. We recorded the wall-clock computing time, in seconds, and the number of iterations of the tests, as the value of λ increased. The average results of each test sample are given in Tables I and II. The corresponding sparsity is also given in the first table. The sparsity is the percentage of zeros in a coefficient vector. The result of sparsity in Table I was calculated based on the result of SMO. It is defined as $\frac{\text{sum}(\boldsymbol{x} < \varepsilon \max(\boldsymbol{x}))}{k}$, where we set $\varepsilon = 0.001$ in our experiment.

From Table I, we have the following observations. First, for all methods the computing time and numbers of iterations decrease gradually as the value of λ increases. Second, for the non-smooth l_1QP model, our SMO method is much faster than proximal and active-set methods. Third, for the smooth NNQP model, SMO is also efficient, though the active-set method keeps its efficiency. Fourth, the sparsity increases as the value of λ rises. When $\lambda = 0$, only the non-negativity induces the sparsity in the NNQP model. We can see that very high sparsity can be obtained by using the non-negativity solely.

Table I
Mean Computing Time (in Seconds) of Each Test Sample When Using 5356 Samples as Training Set

λ	l_1QP				NNQP		
	SMO	proximal	active-set	sparsity	SMO	active-set	sparsity
0	-	-	-	-	17.99	0.49	0.9954
0.0001	273.78	554.37	-	0.9478	17.94	0.48	0.9954
0.001	22.30	557.20	-	0.9950	18.02	0.49	0.9954
0.01	20.75	558.47	-	0.9956	17.66	0.49	0.9956
0.1	16.20	550.69	-	0.9968	13.67	0.34	0.9968
0.2	12.93	635.15	-	0.9975	10.99	0.29	0.9975
0.3	10.02	615.11	1669.63	0.9980	8.48	0.22	0.9980
0.4	7.83	572.55	1426.41	0.9984	6.58	0.17	0.9984
0.5	6.39	513.75	1157.20	0.9987	5.42	0.14	0.9987
0.6	5.17	441.12	923.95	0.9989	4.40	0.12	0.9989
0.7	3.91	357.81	684.89	0.9992	3.31	0.10	0.9992
0.8	2.86	254.41	553.94	0.9994	2.42	0.07	0.9994
0.9	1.70	97.00	395.52	0.9996	1.44	0.05	0.9996

From Table II, the active-set algorithms converge to the optimal solution in a few iterations, but the computational cost of each iteration is expensive, and Hessian matrix is involved in the computation. Our method and the proximal method have a larger number of iterations, but the computational cost of each iteration is very low. Both methods have closed-form update rules. The operation on Hessian is avoided in our SMO methods. Finally, we have to mention that our SMO methods can obtain identical solutions with active-set methods, which corroborates that the decomposition methods converge well. However, the proximal method can only obtain approximate results in many cases, because

it is a first-order method that needs a large number of iterations to converge.

Table II
MEAN NUMBER OF ITERATIONS OF EACH TEST SAMPLE WHEN USING 5356 SAMPLES AS TRAINING SET

λ	l_1QP			NNQP	
	SMO	proximal	active-set	SMO	active-set
0	-	-	-	53033	29.37
0.0001	682153	29250	-	52994	29.36
0.001	55612	29427	-	52981	29.30
0.01	51438	29451	-	51439	28.23
0.1	40068	29302	-	40068	20.07
0.2	31962	34334	-	31962	15.43
0.3	24899	33212	13.30	24899	12.20
0.4	19348	30902	11.40	19348	9.80
0.5	15927	27752	9.40	15927	8.22
0.6	12853	23834	7.80	12853	6.88
0.7	9680	19236	6.00	9680	5.51
0.8	7092	13144	5.00	7092	4.30
0.9	4210	5152	3.80	4210	3.09

Additionally, we tested the prediction accuracies of our l_1QP and NNQP models for classifying the microarray gene profiles. We used our SMO methods in the optimizations. The interior-point and proximal methods are assumed to obtained the same solution as the SMO methods. The mean accuracies of 10-fold cross-validation are given in Table III. We can see that both models obtained very high accuracies. The NNQP model with $\lambda = 0$ obtained the best result on this data. This observation suggests that we need to try the easier NNQP model first before resorting to l_1QP model, when applying sparse coding techniques in problems, for example classification.

Table III
MEAN PREDICTION ACCURACIES OF THE l_1QP AND NNQP MODELS USING 10-FOLD CROSS-VALIDATION

λ	l_1QP	NNQP
0	-	0.9762
0.5	0.9727	0.9727
0.9	0.9654	0.9654

V. CONCLUSION

The optimization of sparse coding models is the bottleneck of applying sparse representation to large-scale data. In this paper, we propose decomposition method for the optimization. We especially investigate its extreme case – the sequential minimal optimization, where only one variable is updated by a closed-form rule while the rest are fixed in an iteration. Our experiments on a very large data show that our SMO methods are very efficient, particularly in the non-smooth l_1QP model. Although our idea of decomposition is applied for two typical sparse coding models, it could be easily generalized to many other sparse models that may be more complicated and deliberated. The implementation of the SMO methods can be found in our sparse representation toolbox [19] (see functions `l1QPSMO`, `NNQPSMO`, and `KSRC` therein). We hope more efficient algorithms for large-scale data can be inspired by the idea of decomposition (coordinate descent or block coordinate descent).

As future work, our methods will be tested on larger data. The SMO methods will be used in dictionary learning as well. The decomposition method will also be explored for hierarchical sparse coding which is a non-smooth problem. Furthermore, improvement could be made for the case of sparse Hessian. Sparse Hessian can be obtained by some kernels, for example radial basis function. It is also necessary to investigate and realize decomposition method based on block coordinate descent in order to improve the efficiency. Finally, we have to mention that, since sparse coding models belong to neural networks, parallel algorithms (as in deep net learning) should be investigated.

ACKNOWLEDGMENT

This research has been supported by OGS 2011-2013, and NSERC Grants #RGPIN228117-2011.

REFERENCES

[1] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[2] A. M. Bruckstein, D. L. Donoho, and M. Elad, "From sparse solutions of systems of equations to sparse modeling of signals and images," *SIAM Review*, vol. 51, no. 1, pp. 34–81, 2009.

[3] M. Elad, *Sparse and Redundant Representations: From Theory to Applications in Signal and Image Processing*. New York: Springer, 2010.

[4] J. Wright, A. 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.

[5] Y. Li and A. Ngom, "Classification approach based on non-negative least squares," *Neurocomputing*, vol. 118, pp. 41–57, 2013.

[6] ——, "Non-negative least squares for the classification of high dimensional biological data," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 10, no. 2, pp. 447–456, 2013.

[7] P. Hoyer, "Modeling receptive fields with non-negative sparse coding," *Neurocomputing*, vol. 52-54, pp. 547–552, 2003.

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

[9] S. J. Kim, K. Koh, M. Lustig, S. Boyd, and D. Gorinevsky, "An interior-point method for large-scale $l1$-regularized least squares," *IEEE J. Selected Topics in Signal Processing*, vol. 1, no. 4, pp. 606–617, 2007.

[10] R. Jenatton, J. Mairal, G. Obozinski, and F. Bach, "Proximal methods for hierarchical sparse coding," *Journal of Machine Learning Research*, vol. 12, no. 2011, pp. 2297–2334, 2011.

[11] Y. Li and A. Ngom, "Sparse representation approaches for the classification of high-dimensional biological data," *BMC Systems Biology*, vol. 7, no. Suppl 4, p. S6, 2013.

[12] H. Lee, A. Battle, R. Raina, and A. Ng, "Efficient sparse coding algorithms," in *NIPS*. NIPS, December 2006, pp. 801–808.

[13] H. Zou and T. Hastie, "Regularization and variable selection via the elastic net," *Journal of the Royal Statistical Society - Series B: Statistical Methodology*, vol. 67, no. 2, pp. 301–320, 2005.

[14] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, UK: Cambridge University Press, 2004.

[15] T. Joachims, "Making large-scale support vector machine learning practical," in *Advances in Kernel Methods: Support Vector Learning*, B. Scholkopf, C. Burges, and A. Somla, Eds. MIT, 1998, ch. 11, pp. 169–184.

[16] D. Bertsekas, *Nonlinear Programming*, 2nd ed. Belmont, MA: Athena Scientific, 2008.

[17] J. Platt, "Fast training of support vector machines using sequential minimal optimization," in *Advances in Kernel Methods: Support Vector Learning*, B. Scholkopf, C. Burges, and A. Somla, Eds. MIT, 1998, ch. 12, pp. 185–208.

[18] N. Dawany, W. Dampier, and A. Tozeren, "Large-scale integration of microarray data reveals genes and pathways common to multiple cancer types," *International Journal of Cancer*, vol. 128, no. 12, pp. 2881–2891, 2011.

[19] Y. Li and A. Ngom, "The sparse representation toolbox in MATLAB," https://sites.google.com/site/sparsereptool.