UN MÉTODO EFICIENTE PARA RESOLVER UN PROBLEMA DE PROGRAMACIÓN CUADRÁTICA DERIVADO DEL APRENDIZAJE DE FUNCIONES DE DISTANCIA

Bac Nguyen*, *, Carlos Morell*,†, Bernard De Baets**,‡
*Facultad de Matemática Física y Computación, Universidad Central "Marta Abreu" de Las Villas, Cuba.
**Department of Mathematical Modelling, Statistics and Bioinformatics, Ghent University, Bélgica.

ABSTRACT
Quadratic programming is a special type of the mathematical optimization problem. It is the problem of optimizing a quadratic function of several variables subject to linear constraints on these variables. When the number of constraints, as well as the number of variables are huge, the general method is not computational tractable. However, in some specific domain of application, we can take some advantages from the problem to improve performance of optimization process. The main idea of this work is to propose a new approach to solve a special quadratic program with large scale constraints, which is formulated to solve a problem of distance metric learning.

KEYWORDS: quadratic programming, optimization, distance metric learning

MSC: 90C20

RESUMEN
La programación cuadrática es un tipo especial de problema de optimización matemática. Se trata de minimizar una función cuadrática de varias variables sujetas a restricciones lineales. Cuando el número de restricciones, así como el número de variables es muy grande, no es posible encontrar una solución utilizando un método de cálculo exacto. Sin embargo, en el ámbito específico de algunas aplicaciones, se puede tomar ventajas de las características particulares del problema para mejorar el rendimiento del proceso de optimización. La idea principal de este trabajo es proponer un nuevo enfoque para resolver un problema particular de programación cuadrática con restricciones a gran escala, que se formula para resolver un problema de aprendizaje de funciones de distancia.

1. INTRODUCCIÓN

La optimización es la búsqueda de la mejor solución (la óptima) a un problema. Esta solución óptima depende del área sobre la cual se está trabajando y puede significar encontrar el costo mínimo, los beneficios máximos, o la ruta mínima. Todos estos problemas y muchos otros pueden ser planteados de forma numérica, mediante un modelo de programación matemática. Motivado por estas razones y por la creación reciente de nuevos algoritmos ha surgido un nuevo interés en la materia. La programación cuadrática es un caso particular de la programación no lineal y por tanto los algoritmos clásicos, entiéndase: los métodos de punto

*nguyencongbacbk@gmail.com
†cmorellp@uclv.edu.cu
‡Bernard.DeBaets@ugent.be.

interior [24, 12, 18], de restricciones activas [13, 3], gradiente conjugado [22], estimación de los multiplicadores de Lagrange, etc, pueden ser empleados para su solución. Un problema de programación cuadrática es aquel que contiene una función objetivo cuadrática y restricciones lineales. En muchos campos de investigación donde se utiliza la optimización, tales como minería de datos, economía, finanzas y otras más, ella desempea un papel muy importante. En particular, en el aprendizaje automático se utiliza para el entrenamiento de las máquinas de vectores de soporte [6] y otros modelos afines.

Dentro de este campo, recientemente se ha comenzado a trabajar en el aprendizaje de funciones de distancia para su aplicación en dominios como la recuperación de información y clasificación [16, 15], visión por computadora [19, 11] y bioinformática [34, 26]. Varios son los modelos y disímiles las formas de aprender tales funciones de distancia a partir de los datos. Sirva de ejemplo las propuestas de [7, 10, 32, 25, 2, 20]. Usualmente, sus autores disean o adecuan una técnica de optimización a cada caso particular aprovechando las particularidades del modelo en cuestión. Los enfoques para aprender una métrica generalmente recaen en dos categorías: la optimización basada en los vectores propios y basada en optimización. Se mencionan aquí algunas técnicas que se aplican con éxito en el aprendizaje de funciones de distancia. Los métodos de optimización de valores propios se han utilizado para encontrar una transformación lineal del espacio de entrada, tales como Análisis de Componentes Principales (PCA) [17], Análisis Discriminante Lineal (LDA) [8], y Análisis de los Componentes Relevantes (RCA) [1, 30]. Por otra parte, se puede tratar de formular la función objetivo como un problema de optimización que entonces puede resolverse mediante algoritmos estándares. Geoldberger et al. [10] proponen usar el método del gradiente mientras que Xing et al. [33] y Weinberger et al. [32] aplican la técnica de descenso del gradiente proyectado. Por su parte Davis et al. [7] usan la proyección de Bregman y Shalev-Shwartz et al. [28] proponen la técnica de descenso de gradiente estocástico.

Schult y Joachims en [27] han propuesto un modelo de propósito general para el aprendizaje de funciones de distancia que se expresa mediante un problema de programación cuadrática. Para encontrar una solución particular, los autores emplean un paquete de resolución de propósito general cuyo funcionamiento se degrada muy rápidamente cuando se incrementa el número de restricciones o la cantidad de variables. En este trabajo se presenta un nuevo algoritmo para resolver este tipo particular de problema de optimización de una manera más eficiente. En la siguiente sección se introducen los conceptos básicos del modelo mientras que en la sección 3 se describe nuestra propuesta de solución. Por último, en la sección 4 se hace un análisis experimental que complementa el análisis de complejidad del método.

2. BÁSICOS

En esta sección se introduce el problema a tratar y se presentan las ideas fundamentales de la técnica de descenso por coordenadas que se empleará en la obtención de los resultados principales.

2.1. Aprendizaje de funciones de distancia mediante restricciones relativas

Dado un conjunto de instancias $\mathcal{X} = \{x_1, x_2, \ldots, x_N\} \subset \mathbb{R}^m$ y sus clases correspondientes $\mathcal{Y} = \{y_1, y_2, \ldots, y_N\}$, Schultz y Joachims en [27] proponen un enfoque para el aprendizaje de funciones de distancia sobre un conjunto de restricciones relativas de la forma:

$$\mathcal{P} = \{(i, j, k) : x_i, x_j, x_k \in \mathcal{X} \text{ y } x_i \text{ debe estar más cercano a } x_j \text{ que a } x_k\}. \tag{2.1}$$

La familia de funciones de distancia entre dos instancias $u, v \in \mathcal{X}$, parametrizada por dos matrices, A y W, tiene la forma:

$$\begin{aligned} d_{A,W}(u, v) &= (u - v)^T AWA^T(u - v) \\ &= (A^T u - A^T v)^T W(A^T u - A^T v) \\ &= w^T \Big((A^T u - A^T v) \circ (A^T u - A^T v) \Big), \end{aligned}$$

donde, A es una matriz de transformación lineal, W es una matriz diagonal, $\circ$ denotado el producto de Hadamard, y el vector w es la diagonal de la matriz W. En este problema $\|.\|_F$ denotada la norma de Frobenius, el objetivo en [27] es encontrar una matriz diagonal W semidefinida positiva ($\succeq 0$) tal que:

$$\begin{aligned} \min_{W} \quad & \frac{1}{2}\|AWA^T\|_F^2 \\ \text{s.a.} \quad & \forall(i,j,k) \in \mathcal{P} : d_{A,W}(x_i,x_k) - d_{A,W}(x_i,x_j) \geq 1 - \xi_{ijk} \\ & \xi_{ijk} \geq 0 \\ & W \succeq 0 \end{aligned} \tag{2.2}$$

Para expresar el problema (2.2) de una forma más simple, se emplean las siguientes notaciones:

$$\begin{aligned} L &= (A^TA) \circ (A^TA), \\ \Delta^{x_i,x_j} &= (A^Tx_i - A^Tx_j)^T \circ (A^Tx_i - A^Tx_j), \\ z_{ijk} &= \Delta^{x_i,x_k} - \Delta^{x_i,x_j}. \end{aligned}$$

La función objetivo $\frac{1}{2}\|AWA^T\|_F^2$ se convierte en $\frac{1}{2}w^TLw$ y las restricciones $d_{A,W}(x_i,x_k) - d_{A,W}(x_i,x_j) \geq 1 - \xi_{ijk}$ se convierten en: $w^Tz_{ijk} \geq 1 - \xi_{ijk}$, por:

$$\begin{aligned} d_{A,W}(x_i,x_k) - d_{A,W}(x_i,x_j) &= w^T\Big((A^Tx_i - A^Tx_k) \circ (A^Tx_i - A^Tx_k)\Big) - w^T\Big((A^Tx_i - A^Tx_j) \circ (A^Tx_i - A^Tx_j)\Big) \\ &= w^T\Delta^{x_i,x_k} - w^T\Delta^{x_i,x_j} \\ &= w^T\Big(\Delta^{x_i,x_k} - \Delta^{x_i,x_j}\Big) \\ &= w^Tz_{ijk}. \end{aligned}$$

Donde z_{ijk} y ξ_{ijk} son elementos de conjuntos numerables y, por tanto, se pueden reenumerar utilizando los sub-índices $\{1,2,\dots n\}$ (n es el tamao de $\mathcal{P}$). Según [27], el problema (2.2) se convierte ahora en:

$$\begin{aligned} \min_{w,\xi} \quad & \frac{1}{2}\vec{w}^TL\vec{w} + C\sum_{i=1}^{n}\xi_i \\ \text{s.a.} \quad & \vec{w}^Tz_i \geq 1 - \xi_i, i = 1,\dots,n \\ & \xi_i \geq 0, i = 1,...,n \\ & w_j \geq 0, j = 1,...,m \end{aligned} \tag{2.3}$$

El parámetro $C \in \mathbb{R}^+$ controla la penalización entre la función objetivo y las restricciones.

Aunque este método se valida mediante su empleo en problemas de recuperación de información, su generalidad permite que se utilice en otros contextos. Las restricciones en forma de tripletas se formulan a partir de los datos de aprendizaje y en cada contexto tienen un significado diferente. El punto débil es que la cantidad de restricciones puede crecer rápidamente (hasta un máximo de (N^3) restricciones para un conjunto de datos de N instancias). En la propuesta original se usa un solucionador de propósito general cuyo comportamiento se degrada rápidamente al crecer la dimensión del problema. Es por ello que en lo adelante nos concentramos en disear un método específico que aproveche las particularidades del problema (2.3) para obtener una solución de forma más eficiente.

2.2. Descenso por coordenadas

El método de descenso por coordenadas es una técnica de optimización muy popular para la solución de problemas convexos m-dimensionales. Su esquema básico se presenta en el Algoritmo 1. En cada iteración

interna, se fijan todas las variables excepto la variable activa x_i, y luego se optimiza la función con respecto a esta variable. En general, el método de descenso por coordenadas es eficiente si cada iteración simple se puede realizar con un costo mínimo. Esto tiene la ventaja de que no hay que tener en cuenta las interacciones o dependencias entre las variables. La convergencia del método ha sido analizada con anterioridad en trabajos tales como [23, 31, 29].

Algorithm 1 Algoritmo descenso por coordenadas

- Dado los valores iniciales en $k = 1$, $\vec{x}^0 = [x_1^0, x_2^0, ..., x_m^0]^T$
- Problema de optimización es:
  $\min_{x \in \mathbb{R}^m} \quad f(\vec{x})$
- **Mientras** el criterio de convergencia no se cumple (iteración externa)
  (1) **Para** $i \leftarrow 1 \dots m$ (iteración interna)
  $$x_i^k = \arg\min_y \quad f(x_1^k, \dots, x_{i-1}^k, y, x_{i+1}^{k-1}, \dots, x_m^{k-1})$$
  (2) $k \leftarrow k + 1$

Las aplicaciones de este método son variadas (véase por ejemplo [9, 35]). En el contexto de este artículo es de especial interés su aplicación en el entrenamiento de las máquinas de vectores de soporte lineales [14] y que ha servido de inspiración a nuestro trabajo. A continuación destacamos las semejanzas y diferencias de ambos enfoques. Para ello definiremos primeramente el problema planteado en [14].

Dado un conjunto de vectores de entrada $\mathcal{X} = \{x_1, x_2, \dots, x_n\} \subset \mathbb{R}^m$ y sus clases correspondientes $\mathcal{Y} = \{y_1, y_2, \dots, y_n\} \in \{-1, +1\}$, las máquinas de vectores de soporte lineales resuelven el siguiente problema de optimización:

$$\begin{aligned} \min_{w,\xi} \quad & \frac{1}{2}\vec{w}^T\vec{w} + C\sum_{i=1}^{n}\xi_i \\ \text{s.a.} \quad & y_i\vec{w}^T x_i \geq 1 - \xi_i \\ & \xi_i \geq 0, i = 1, \dots, n \end{aligned} \tag{2.4}$$

donde ξ_i es la variable de holgura no negativa y C es un parámetro de penalización. La idea principal de [14] es resolver el problema dual y mantener una traza de los valores del vector $\vec{w}$ en el problema primal.

El método propuesto aquí sigue esta idea pero se aplica en el problema (2.3). Es fácil notar que ambos problemas, (2.3) y (2.4), son parecidos (haciendo el cambio $z_i \equiv y_i x_i$). Las diferencias radican en que el primer término de la función objetivo de (2.3) tiene insertada una matriz L semidefinida positiva y en el hecho de exigir la no negatividad de las componentes del vector $\vec{w}$. Esto implica que los problemas duales respectivos difieren y, por tanto, en la solución de (2.3) no se puede emplear directamente el método propuesto en [14]. En la siguiente sección se describe cómo adaptar este método para manejar estas diferencias.

3. MÉTODO PROPUESTO

En esta sección, se describe en detalle como se resuelve el problema (2.3) usando la técnica de descenso por coordenadas. Primeramente, se convierte el problema primal en su problema dual mediante la función dual de Lagrange. Luego, se aplica la técnica de descenso por coordenadas para resolver el problema dual. Afortunadamente, la solución del problema primal también se puede mantener durante el proceso de optimización.

Con el propósito de formular el problema dual de (2.3), se puede usar la función Lagrangiana $\mathcal{L}(w, \xi, \lambda, \mu, t)$:

$\mathbb{R}^m \times \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^m$, usando los multiplicadores de Lagrange $\vec{\lambda}, \vec{\mu}, \vec{t}$:

$$\mathcal{L}(\vec{w}, \vec{\xi}, \vec{\lambda}, \vec{\mu}, \vec{t}) = \frac{1}{2}\vec{w}^T L \vec{w} + C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\lambda_i(\vec{w}^T z_i - 1 + \xi_i) - \sum_{i=1}^{n}\mu_i\xi_i - \sum_{i=1}^{m} t_i w_i. \tag{3.5}$$

De esta forma se obtiene la función dual de Lagrange para el problema (2.3),

$$\begin{aligned} g(\vec{\lambda}, \vec{\mu}, \vec{t}) &= \inf_{\vec{w}\in\mathbb{R}^m, \vec{\xi}\in\mathbb{R}^n} \mathcal{L}(\vec{w}, \vec{\xi}, \vec{\lambda}, \vec{\mu}, \vec{t}) \\ &= \inf_{\vec{w}\in\mathbb{R}^m, \vec{\xi}\in\mathbb{R}^n} \left(\frac{1}{2}\vec{w}^T L \vec{w} + C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\lambda_i(\vec{w}^T z_i - 1 + \xi_i) - \sum_{i=1}^{n}\mu_i\xi_i - \sum_{i=1}^{m} t_i w_i\right). \end{aligned}$$

Para resolver (2.3), hay que buscar el punto de estacionario de $\mathcal{L}(w, \xi, \lambda, \mu, t)$, es decir, minimizar en las variables primales $\vec{w}, \vec{\xi}$ y maximizar en las variables duales $\vec{\lambda}, \vec{\mu}, \vec{t}$. Haciendo igual a cero la derivada de $\mathcal{L}(w, \xi, \lambda, \mu, t)$ con respeto a las variables primales, se obtienen las siguientes ecuaciones:

$$\frac{\partial \mathcal{L}(w, \xi, \lambda, \mu, t)}{\partial \vec{w}} = L\vec{w} - \sum_{i=1}^{n}\lambda_i z_i - \vec{t} = 0 \Rightarrow \vec{w} = L^{-1}\left(\sum_{i=1}^{n}\lambda_i z_i + \vec{t}\right) \tag{3.6}$$

$$\frac{\partial \mathcal{L}(w, \xi, \lambda, \mu, t)}{\partial \xi_i} = C - \lambda_i - \mu_i = 0 \Rightarrow C = \lambda_i + \mu_i, \quad i = 1, \ldots, n \tag{3.7}$$

La ecuación (3.7) implica que $0 \leq \lambda_i \leq C$ porque $\mu_i \geq 0, \forall i$. Sustituyendo (3.6) en la ecuación (3.5), finalmente, la función dual de Lagrange se expresa como:

$$\begin{aligned} g(\vec{\lambda}, \vec{\mu}, \vec{t}) &= \frac{1}{2}\vec{w}^T L \vec{w} + C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\lambda_i(\vec{w}^T z_i - 1 + \xi_i) - \sum_{i=1}^{n}\mu_i\xi_i - \sum_{i=1}^{m} t_i w_i \\ &= \frac{1}{2}\vec{w}^T L \vec{w} + \underbrace{\sum_{i=1}^{n}\xi_i(C - \lambda_i - \mu_i)}_{=0} - \left(\sum_{i=1}^{n}\lambda_i \vec{w}^T z_i + \vec{w}^T\vec{t}\right) + \sum_{i=1}^{n}\lambda_i \\ &= \frac{1}{2}\vec{w}^T L \vec{w} - \vec{w}^T\left(\sum_{i=1}^{n}\lambda_i z_i + \vec{t}\right) + \sum_{i=1}^{n}\lambda_i \\ &= \frac{1}{2}\vec{w}^T L \vec{w} - \vec{w}^T L \underbrace{L^{-1}\left(\sum_{i=1}^{n}\lambda_i z_i + \vec{t}\right)}_{=\vec{w}} + \sum_{i=1}^{n}\lambda_i \\ &= -\frac{1}{2}\vec{w}^T L \vec{w} + \sum_{i=1}^{n}\lambda_i \\ &= -\frac{1}{2}\left(L^{-1}\left(\sum_{i=1}^{n}\lambda_i z_i + \vec{t}\right)\right)^T L\left(L^{-1}\left(\sum_{i=1}^{n}\lambda_i z_i + \vec{t}\right)\right) + \sum_{i=1}^{n}\lambda_i \\ &= -\frac{1}{2}\left(\sum_{i=1}^{n}\lambda_i z_i^T\right) L^{-1}\left(\sum_{i=1}^{n}\lambda_i z_i\right) - \left(\frac{1}{2}\vec{t}^T L^{-1}\vec{t}\right) - \left(\vec{t}^T L^{-1}\sum_{i=1}^{n}\lambda_i z_i\right) + \sum_{i=1}^{n}\lambda_i \\ &= -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\lambda_i\lambda_j z_i^T L^{-1} z_j - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} t_i t_j L_{ij}^{-1} - \sum_{i=1}^{n}\lambda_i z_i^T L^{-1} t + \sum_{i=1}^{n}\lambda_i. \end{aligned}$$

Se quiere buscar w y $\vec{\xi}$ que minimicen la función objetivo, y $\vec{\lambda}$, μ, t que maximicen $g(\vec{\lambda}, \mu, t)$. Haciendo $f(\vec{\lambda}, t) = -g(\vec{\lambda}, \mu, t)$ el problema primal (2.3) se convierte en el problema dual equivalente:

$$\begin{aligned} \min_{\vec{\lambda},t} \quad & f(\vec{\lambda}, t) = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\lambda_i\lambda_j z_i^T L^{-1} z_j + \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} t_i t_j L_{ij}^{-1} + \sum_{i=1}^{n}\lambda_i z_i^T L^{-1} t + \sum_{i=1}^{n}\lambda_i \\ \text{s.a.} \quad & C \geq \lambda_i \geq 0, i = 1, \ldots, n \\ & t_j \geq 0, j = 1, \ldots, m \end{aligned} \tag{3.8}$$

Este problema dual es otro problema de optimización, y aunque la función objetivo es más complicada que la del problema primal sus restricciones son muchos más simples. Según [4], la función dual de Lagrange $g(\vec{\lambda}, \mu, t)$ es una función cóncava aún cuando la función primal no lo sea. Esto implica que la función $f(\lambda, t)$ es convexa, es decir, (3.8) es un problema de programación cuadrática. Para resolver este problema, se divide el problema dual en sub-problemas más pequeos y, mediante la técnica de descenso por coordenadas, se encuentra la solución analítica para un sub-problema en cada iteración.

El proceso de optimización empieza a partir de un valor inicial $(\vec{\lambda}^0, t^0) \in \mathbb{R}^{n+m}$ y se genera una sucesión de vectores $\{(\lambda^k, t^k)\}_{k=0}^{\infty}$. Llamaremos iteración externa al procedimiento que actualiza (λ^k, t^k) a partir de (λ^{k-1}, t^{k-1}). Cada iteración externa ejecuta $(n + m)$ iteraciones internas en las que se resuelve (3.8) en términos de la variable $\vec{\lambda}$ y luego en término de t. Por otro lado, en cada iteración interna se generan los vectores $\vec{\lambda}^{k,i}$ para $i = 1, \ldots, n$ y $t^{k,j}$ para $j = 1, \ldots, m$ tal que:

$$\begin{aligned} \vec{\lambda}^{k,i} &= \left[\lambda_1^{k+1,i}, \ldots, \lambda_{i-1}^{k+1,i}, \lambda_i^{k,i}, \ldots, \lambda_n^{k,i}\right]^T, \\ t^{k,j} &= \left[t_1^{k+1,j}, \ldots, t_{j-1}^{k+1,j}, t_j^{k,j}, \ldots, t_m^{k,j}\right]^T. \end{aligned}$$

donde k denomina el índice de la iteración externa, i, j denominan los índices de la iteración interna. Además, se tiene que $\vec{\lambda}^{k,1} = \vec{\lambda}^k, \vec{\lambda}^{k,n+1} = \vec{\lambda}^{k+1}$ y $t^{k,1} = t^k, t^{k,m+1} = t^{k+1}$. A continuación, se describe en detalle el método para resolver simultáneamente $\vec{\lambda}^k$ y t^k en cada iteración externa.

3.1. Encontrar los valores de $\vec{\lambda}^k$

Para comenzar, se calculan en cada iteración los valores $\vec{\lambda}^{k,i}$. Para actualizar $\vec{\lambda}^{k,i+1}$ a partir de $\vec{\lambda}^{k,i}$, se resuelve el siguiente problema de una variable:

$$\begin{aligned} \min_{d\in\mathbb{R}} \quad & f(\vec{\lambda}^{k,i} + de_i, t^k) \\ \text{s.a.} \quad & C \geq d + \lambda_i^k \geq 0 \end{aligned} \tag{3.9}$$

donde $e_i = [0, \ldots, 1, \ldots, 0]^T$. La función objetivo en (3.9) es una función cuadrática con respeto a la variable d:

$$f(\lambda^{k,i} + de_i, t^k) = \frac{1}{2}\left(z_i^T L^{-1} z_i\right) d^2 + \nabla_i f(\vec{\lambda}^{k,i}, t^k) d + \delta, \tag{3.10}$$

donde δ es un término constante, $\nabla_i f(\vec{\lambda}^{k,i}, t^k)$ es la i-ésima componente del gradiente ∇f, y se define como (para los casos $i \leq n$):

$$\begin{aligned} \nabla_i f(\vec{\lambda}^{k,i}, t^k) &= \sum_{j=1}^{n}\lambda_j^{k,i} z_i^T L^{-1} z_j + z_i^T L^{-1} t - 1 \\ &= z_i^T L^{-1}\left(\sum_{j=1}^{n} z_j \lambda_j^{k,i} + t\right) - 1 \\ &= z_i^T w - 1. \end{aligned} \tag{3.11}$$

El problema (3.9) tiene la solución trivial $d = 0$ si y solo si:

$$\nabla_i^P f(\lambda^{k,i}, t^k) = 0, \tag{3.12}$$

donde $\nabla_i^P f(\lambda, t)$ es el gradiente proyectado [21], que se calcula de la forma siguiente:

$$\nabla_i^P f(\lambda, t) = \begin{cases} \nabla_i f(\lambda, t) & , \text{ si } 0 < \lambda_i < C \\ \min(0, \nabla_i f(\lambda, t)) & , \text{ si } \lambda_i = 0 \\ \max(0, \nabla_i f(\lambda, t)) & , \text{ si } \lambda_i = C. \end{cases}$$

Si el gradiente proyectado $\nabla_i^P f(\lambda^{k,i}, t^k)$ es cero, esto implica que la solución actual $\lambda^{k,i}$ es óptima. Solo se actualiza $\vec{\lambda}^{k,i}$ si $\nabla_i^P f(\lambda^{k,i}, t^k) \neq 0$. En este caso, debido a que la función (3.10) es cuadrática y sólo depende de la variable d, debemos considerar el valor de $z_i^T L^{-1} z_i$. Si $z_i^T L^{-1} z_i > 0$ entonces valor de d donde (3.10) se hace mínimo es:

$$d = -\frac{\nabla_i f(\vec{\lambda}^{k,i}, t^k)}{z_i^T L^{-1} z_i}.$$

Luego, tenemos que restringir la solución $(\lambda_i^{k,i} + d)$ del problema (3.9) en el intervalo $[0, C]$ para garantizar la factibilidad de la solución para el problema dual.

$$\lambda_i^{k,i+1} = \min\left(\max\left(\lambda_i^{k,i} - \frac{\nabla_i f(\vec{\lambda}^{k,i}, t^k)}{z_i^T L^{-1} z_i}, 0\right), C\right). \tag{3.13}$$

Por otro lado, si $z_i^T L^{-1} z_i = 0$, esto implica que $z_i = 0$. Por (3.11), se tiene que $\nabla_i f(\vec{\lambda}^{k,i}, t^k) = -1$. Entonces la solución óptima para el problema (3.9) es $\lambda_i^{k,i+1} = C$ (si se define $1/0 = +\infty$, también se puede calcular $\lambda_i^{k,i+1}$ por la ecuación (3.13)). Para calcular $\nabla_i f(\vec{\lambda}^{k,i}, t^k)$ se necesita actualizar w durante todo el proceso de optimización. Esto se puede realizar fácilmente si se tienen λ_i^{old}, λ_i como los valores de antes y después de la actualización respectivamente:

$$w \leftarrow w + (\lambda_i - \lambda_i^{old}) L^{-1} z_i.$$

Aquí aparece otro problema: el cálculo de $L^{-1} z_i$ en cada iteración interna tiene una complejidad temporal de $\mathcal{O}(m^2)$. Esto se puede realizar de manera más eficiente si se precalculan los valores de $L^{-1} z_i$ antes del proceso de optimización. De esta forma se reduce la complejidad temporal en cada iteración desde $\mathcal{O}(m^2)$ hasta $\mathcal{O}(m)$ para actualizar los valores de w.

3.2. Encontrar los valores de t^k

En esta parte se describen los pasos necesarios para encontrar t^k. El procedimiento es muy similar al realizado para calcular $\vec{\lambda}^k$. Después del cálculo de los valores de λ^{k+1}, para actualizar desde $t^{k,j}$ hasta $t^{k,j+1}$, se resuelve el siguiente problema de optimización de una variable :

$$\begin{aligned} \min_{d \in \mathbb{R}} \quad & f(\lambda^{k+1}, t^{k,j} + d e_j) \\ \text{s.a.} \quad & d + t_j^k \geq 0 \end{aligned} \tag{3.14}$$

La función a minimizar en (3.14) es también una función cuadrática en la variable d:

$$f(\vec{\lambda}^{k+1}, t^{k,j} + d e_j) = \frac{1}{2} L_{jj}^{-1} d^2 + \nabla_{n+j} f(\vec{\lambda}^{k+1}, t^{k,j}) d + \delta \tag{3.15}$$

donde δ es un término constante y $\nabla_{n+j} f(\vec{\lambda}^{k+1}, t^{k,j})$ es $(n+j)$-ésima componente del gradiente ∇f definida como:

$$\nabla_{n+j} f(\vec{\lambda}^{k+1}, t^{k,j}) = \sum_{i=1}^{m} L_{ji}^{-1} t_i^{k,j} + \left[\underbrace{L^{-1} \sum_{i=1}^{n} \lambda_i^{k+1} z_i}_{=T(\vec{\lambda})} \right]_j$$
$$= \sum_{i=1}^{m} L_{ji}^{-1} t_i^{k,j} + \left[T(\vec{\lambda}) \right]_j . \tag{3.16}$$

Al igual que lo sucedido con el problema (3.9), (3.14) tiene la solución trivial $d = 0$ si y solo si:

$$\nabla_{n+j}^{P} f(\lambda^{k+1}, t^{k,j}) = 0, \tag{3.17}$$

donde $\nabla_{n+j}^{P} f(\lambda^{k+1}, t^{k,j})$ es el gradiente proyectado [21], que se calcula de la forma siguiente:

$$\nabla_{n+j}^{P} f(\lambda, t) = \begin{cases} \nabla_{n+j} f(\lambda, t) & , \text{ si } t_j > 0 \\ \min(0, \nabla_{n+j} f(\lambda, t)) & , \text{ si } t_j = 0 \end{cases}$$

Cuando (3.17) no se cumple entonces $L_{jj}^{-1} > 0$ debido a que L es una matriz definida positiva y por ello el punto óptimo que minimiza la función (3.15) es:

$$d = -\frac{\nabla_{j+n} f(\vec{\lambda}^{k+1}, t^{k,j})}{L_{jj}^{-1}}.$$

Luego, se restringe la solución $(t_j^{k,j} + d)$ del problema (3.14) en el intervalo $[0, +\infty)$ para garantizar la factibilidad de la solución para el problema dual.

$$t_j^{k,j+1} = \max\left(t_j^{k,i} - \frac{\nabla_{j+n} f(\vec{\lambda}^{k+1}, t^{k,j})}{L_{jj}^{-1}}, 0 \right). \tag{3.18}$$

La complejidad temporal para calcular $\nabla_{j+n} f(\vec{\lambda}^{k+1}, t^{k,j})$ es $\mathcal{O}(m)$. Esto se debe a que el primer término $\sum_{i=1}^{m} L_{ji}^{-1} t_i^{k,j}$ en (3.16) puede calcularse en $\mathcal{O}(m)$ y el segundo término $\left[T(\vec{\lambda}) \right]_j = \left[L^{-1} \sum_{i=1}^{n} \lambda_i^{k,n+1} z_i \right]_j$ se puede calcular en $\mathcal{O}(1)$ si se precalcula $T(\vec{\lambda})$ (un vector de longitud m). Este término es dependiente de $\vec{\lambda}$ y por ello tenemos que actualizar este valor durante todo el proceso de optimización. Después de cada iteración, a partir del nuevo valor $\vec{\lambda}^{k,i}$ calculado, se actualiza $T(\lambda)$

$$T(\vec{\lambda}) \leftarrow T(\vec{\lambda}) + (\lambda_i - \lambda_i^{old}) L^{-1} z_i, \tag{3.19}$$

y ello implica una complejidad temporal de $\mathcal{O}(m)$ si se precalculan los valores de $L^{-1} z_i$.

Se denota t^{old} al vector t antes de actualizarse. Entonces, después de actualizar t, tenemos que actualizar w por:

$$w \leftarrow w + L^{-1}(t - t^{old}). \tag{3.20}$$

La condición de parada para el algoritmo 2 puede ser $\|f(\vec{\lambda}, t) - f^{old}(\vec{\lambda}, t)\| \leq \epsilon$, y el método propuesto converge en $\mathcal{O}(\log(1/\epsilon))$ iteraciones (la demostración puede encontrarse en [14]).

Según [5], la velocidad de convergencia del método descenso por coordenadas se puede mejorar mediante la técnica de permutación aleatoria de un subproblema. Eso significa que se hace una permutación del orden en que se resolverá cada coordenada. Formalmente, el orden inicial $(1, \ldots, n)$ se permuta a $(\pi(1), \ldots, \pi(n))$ y se resuelve entonces siguiendo este último orden. El método propuesto se describe en el Algoritmo 2. Los valores iniciales pueden ser $\vec{\lambda} = [0, \ldots, 0]^T$ y $t = [0, \ldots, 0]^T$, lo que implica que los valores iniciales de w y T también son cero en ambos casos. La complejidad temporal en cada iteración es $\mathcal{O}((m+n) * m)$.

Después del cálculo de w mediante el Algoritmo 2, los valores del vector $\vec{\xi}$ se calculan de la siguiente forma:

$$\xi_i = \max(0, 1 - w^T z_i), \quad i = 1, \ldots, n$$

Algorithm 2 Algoritmo de descenso por coordenadas para resolver problema dual (3.8)

- Dados los valores iniciales de $\vec{\lambda}, t$ y los valores correspondientes
  $t \leftarrow L^{-1} \sum_{i=1}^{n} \lambda_i z_i$
  $w \leftarrow L^{-1} (\sum_{i=1}^{n} \lambda_i z_i + t)$
  $cnt \leftarrow 1$

- **Mientras** $\|f(\vec{\lambda}, t) - f^{old}(\vec{\lambda}, t)\| > \epsilon$ y $cnt \leq$ número máximo de iteraciones (iteración externa)
  $cnt \leftarrow cnt + 1$
  (a) Se permuta aleatoriamente $(1, \ldots, n)$ a $(\pi(1), \ldots, \pi(n))$
  (b) **Para cada** $i = \pi(1), \ldots, \pi(n)$ (iteración interna)
    (1) $G_\lambda \leftarrow z_i^T w - 1$
    (2) $PG_\lambda \leftarrow \begin{cases} G_\lambda & , \text{ si } 0 < \lambda_i < C \\ \min(0, G_\lambda) & , \text{ si } \lambda_i = 0 \\ \max(0, G_\lambda) & , \text{ si } \lambda_i = C \end{cases}$
    (3) **Si** $|PG_\lambda| \neq 0$, **entonces**
      $\lambda_i^{old} \leftarrow \lambda_i$
      $\lambda_i \leftarrow \min\left(\max\left(\lambda_i - \frac{G_\lambda}{z_i^T L^{-1} z_i}, 0\right), C\right)$
      $w \leftarrow w + (\lambda_i - \lambda_i^{old}) L^{-1} z_i$
      $T \leftarrow T + (\lambda_i - \lambda_i^{old}) L^{-1} z_i$
  (c) $t^{old} \leftarrow t$
  (d) Se permuta aleatoriamente $(1, \ldots, m)$ a $(\pi(1), \ldots, \pi(m))$
  (e) **Para cada** $j = \pi(1), \ldots, \pi(m)$ (iteración interna)
    (1) $G_t \leftarrow \sum_{i=1}^{m} L_{ji}^{-1} t_i^{k,j} + T_j$
    (2) $PG_t \leftarrow \begin{cases} G_t & , \text{ si } t_j > 0 \\ \min(0, G_t) & , \text{ si } t_j = 0 \end{cases}$
    (3) **Si** $|PG_t| \neq 0$, **entonces**
      $t_j \leftarrow \max\left(t_j - \frac{G_t}{L_{jj}^{-1}}, 0\right)$
  (d) $w \leftarrow w + L^{-1}(t - t^{old})$

4. EXPERIMENTOS Y EVALUACIONES

En esta sección, se investiga el rendimiento del método propuesto (FastQP). Se compara la exactitud del método FastQP con el método estándar (QPC) para resolver un problema de programación cuadrática que

surge cuando se aprenden funciones de distancia según la formulación de [27]. El método estándar emplea la técnica del conjunto activo y su implementación está accesible en *http://sigpromu.org/quadprog/*. Este método resuelve el problema siguiente:

$$\begin{aligned} \min_{x} \quad & \frac{1}{2}x^T Q x + c^T x \\ \text{s.a.} \quad & Ax \leq b, \qquad l \leq x \end{aligned}$$

El problema (2.3) se puede reescribir como:

$$\min_{w,\xi} \quad f(\vec{w},\vec{\xi}) = \frac{1}{2}\begin{bmatrix} w^T & \xi^T \end{bmatrix}\begin{bmatrix} L & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 \end{bmatrix}\begin{bmatrix} w \\ \xi \end{bmatrix} + \begin{bmatrix} C & \dots & C \end{bmatrix}\begin{bmatrix} w \\ \xi \end{bmatrix}$$

$$\text{s.a} \quad -\begin{bmatrix} z_1^T & e_1^T \\ z_2^T & e_2^T \\ \vdots & \vdots \\ z_n^T & e_n^T \end{bmatrix}\begin{bmatrix} w \\ \xi \end{bmatrix} \leq \begin{bmatrix} -1 \\ -1 \\ \vdots \\ -1 \end{bmatrix}, \qquad \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \leq \begin{bmatrix} w \\ \xi \end{bmatrix}$$

Haciendo el cambio de variables: $Q = \begin{bmatrix} L & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 \end{bmatrix}$, $A = -\begin{bmatrix} z_1^T & e_1^T \\ z_2^T & e_2^T \\ \vdots & \vdots \\ z_n^T & e_n^T \end{bmatrix}$, $c = \begin{bmatrix} C \\ C \\ \vdots \\ C \end{bmatrix}$ $b = \begin{bmatrix} -1 \\ -1 \\ \vdots \\ -1 \end{bmatrix}$, $l = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$ y $x = \begin{bmatrix} w \\ \xi \end{bmatrix}$, se obtiene la forma estándar de programación cuadrática para el problema (2.3).

Para este estudio se emplearon conjuntos de datos reconocidos internacionalmente, descritos en la Tabla 1. Estos 5 conjuntos de datos provienen del depósito de datos para aprendizaje automático KEEL [1](*Knowledge Extraction based on Evolutionary Learning*). Todos los experimentos se hacen en el ambiente Matlab, con un Sistema Operativo Windows 8, 64 bits, 10G Ram, Core i5 y 4 CPU, 3.4GHz. Las tripletas $(i, j, k) \in \mathcal{P}$ se forman a partir de un conjunto de datos de entrenamiento $\mathcal{X} = \{x_1, x_2, \dots, x_N\}$ y sus correspondientes clases $\mathcal{Y} = \{y_1, y_2, \dots, y_N\}$, N es el número de instancias. Se seleccionan las tripletas (i, j, k) de manera aleatoria, si $y_i = y_j$ y $y_i \neq y_k$, se agrega la tripleta (i, j, k) al conjunto $\mathcal{P}$. El número m es la cantidad de variables, n es la cantidad de restricciones formadas en el conjunto $\mathcal{P}$. Se realizaron 200 iteraciones para ambos algoritmos y el criterio de parada fue $\|f - f^{old}\| \leq 10^{-6}$, donde f y f^{old} son valores de la función objetivo en dos titeraciones consecutivas.

#	Datos	N	m	n
1.	appendicitis	106	7	318
2.	balance	625	4	1875
3.	ionosphere	351	33	1053
4.	movement_libras	360	90	1080
5.	vowel	990	13	2970

Table 1: Descripción de los datos

La Tabla 2 presenta los resultados experimentales llevados a cabo con los conjuntos de datos seleccionados. Se consideran dos medidas de evaluación del desempeo: el costo y el tiempo. El costo es el valor mínimo

[1]Accesibles en http://sci2s.ugr.es/keel/datasets.php

de la función objetivo mientras que el tiempo se mide en segundos. Se hacen dos experimentos distintos considerando dos matrices L diferentes: a la izquierda aparecen los resultados cuando L es la matriz identidad (I) y a la derecha cuando L es una matriz definida positiva aleatoria[2]. Los resultados muestran que el método propuesto logra alcanzar el mínimo en una fracción del tiempo que demora el método de propósito general. Cuando aumenta la densidad de la matriz L, los valores mínimos obtenidos por nuestro enfoque son aún más pequeos que los obtenidos por el método alternativo mientras que se mantiene la ventaja en cuanto al tiempo.

#	I				L			
	Valor de la función objetivo		Tiempo (segundos)		Valor de la función objetivo		Tiempo (segundos)	
	FastQP	QPC	FastQP	QPC	FastQP	QPC	FastQP	QPC
1.	316.14066	316.14066	**0.00729**	0.28516	**317.34498**	317.56939	**0.00968**	0.15562
2.	1760.39844	1760.39844	**0.00075**	108.35541	**1834.51979**	1834.81564	**0.00139**	106.39375
3.	919.87418	919.87418	**0.01019**	20.43069	**1044.14268**	1050.83556	**0.00452**	21.08289
4.	748.91836	748.91836	**0.00752**	23.93697	**1079.86011**	1079.96339	**0.01725**	30.47952
5.	2829.6551	2829.6551	**0.00103**	284.21132	**2964.17972**	2967.58653	**0.00193**	256.99621

Table 2: Resultados de los experimentos

5. CONCLUSIÓN

En este trabajo se propone un nuevo método para resolver un problema de programación cuadrática que surge al aprender una función de distancia a partir de datos. Este método resuelve el problema dual usando la técnica de descenso por coordenadas. Con esta técnica se logra un mejoramiento en complejidad temporal cuando se compara con un solucionador general de problemas de este tipo. La implementación es sencilla y provee una forma eficiente para la resolución del problema de optimización. Los resultados experimentales permiten comprobar la efectividad de la propuesta al tratar un problema de aprendizaje de funciones de distancia.

RECEIVED: FEBRUARY, 2014.
REVISED: APRIL, 2015.

REFERENCIAS

[1] BAR-HILLEL, A., HERTZ, T., SHENTAL, N. and WEINSHALL, D., (2005): Learning a Mahalanobis Metric from Equivalence Constraints. **Journal of Machine Learning Research**, 6, 937-965.

[2] BELLET, A., HABRARD, A. and SEBBAN, M., (2013): A survey on metric learning for feature vectors and structured data. **Technical Report, arXiv:1306.6709**.

[3] BONNANS, J. F., GILBERT, J. C., LEMARÉCHAL, C. and SAGASTIZABAL, C., (2003): Numerical optimization: theoretical and practical aspects. **Springer Science & Business Media**.

[4] BOYD, S. and VANDENBERGHE, L., (2009): Convex Optimization. **Cambridge University Press**.

[5] CHANG, K. W., HSIEH, C. J. and LIN, C. J., (2008): Coordinate descent method for large-scale l2-loss linear support vector machines. **The Journal of Machine Learning Research**, 9,1369-1398.

[6] CORTES, C. and VAPNIK, V., (1995): Support-vector networks. **Machine Learning**, 20,273-297.

[2]Se genera aleatoriamente una matriz A de rango m, luego se calcula $L = (A^T A) \circ (A^T A)$.

[7] DAVIS, J., KULIS, B., SRA, S. and DHILLON, I., (2007): Information-theoretic metric learning. **Proceedings of the 24th International Conference on Machine Learning (ICML)**.

[8] FISHER, R. A., (1936): The use of multiple measurements in taxonomic problems. **Annals of eugenics**, 7, 179-188.

[9] FRIEDMAN, J., HASTIE, T., HÖFLING, H. and TIBSHIRANI, R., (2007): Pathwise coordinate optimization. **The Annals of Applied Statistics**, 1, 302-332.

[10] GOLDBERGER, J., ROWEIS, S., HINTON, G. and SALAKHUTDINOV, R., (2004): Neighbourhood Components Analysis. **Advances in Neural Information Processing Systems**, 17, 513-520.

[11] GUILLAUMIN, M., VERBEEK, J. and SCHMID, C., (2009): Is that you? Metric learning approaches for face identification. **Computer Vision, 2009 IEEE 12th International Conference**, 498-505.

[12] HELMBERG, C., RENDL, F., VANDERBEI, R. J. and WOLKOWICZ, H., (1996): An interior-point method for semidefinite programming. **SIAM Journal on Optimization**, 6, 342-361.

[13] HINTERMÜLLER, M., ITO, K. and KUNISCH, K., (2002): The primal-dual active set strategy as a semismooth newton method. **SIAM Journal on Optimization**, 13, 865-888.

[14] HSIEH, C. J., CHANG, K. W., LIN, C. J., KEERTHI, S. S. and SUNDARARAJAN, S., (2008): A Dual Coordinate Descent Method for Large-scale Linear SVM. **Proceedings of the 25th International Conference on Machine Learning**, 408-415.

[15] HUA, G., BROWN, M. and WINDER, S., (2007): Discriminant Embedding for Local Image Descriptors. **Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference**, 1-8.

[16] JAIN, P., KULIS, B. and GRAUMAN, K., (2008): Fast image search for learned metrics. **Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference**, 1-8.

[17] JOLLIFFE, I., (2005): Principal Component Analysis. **Wiley Online Library**.

[18] KIM, S. J., KOH, K., LUSTIG, M., BOYD, S. and GORINEVSKY, D., (2007): An interior-point method for large-scale l 1-regularized least squares. **Selected Topics in Signal Processing, IEEE Journal of**, 1, 606-617.

[19] KOESTINGER, M., HIRZER, M., WOHLHART, P., ROTH, P. M. and BISCHOF, H., (2012): Large scale metric learning from equivalence constraints. **IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2012**, 2288 - 2295.

[20] KULIS, B., (2012): Metric learning: A survey. **Foundations & Trends in Machine Learning**, 5, 287-364.

[21] LIN, C. J. and MORÉ, J. J., (1999): Newton's method for large bound-constrained optimization problems. **SIAM Journal on Optimization**, 9, 1100-1127.

[22] LUENBERGER, D. G., (1973): Introduction to linear and nonlinear programming. **Addison-Wesley Reading, MA**, 28.

[23] LUO, Z. Q. and TSENG, P., (1992): On the convergence of the coordinate descent method for convex differentiable minimization. **Journal of Optimization Theory and Applications**, 72, 7-35.

[24] MEHROTRA, S., (1992): On the implementation of a primal-dual interior point method. **SIAM Journal on Optimization**, 2, 575-601.

[25] PARAMESWARAN, S. and WEINBERGER, K. Q., (2010): Large margin multi-task metric learning. **Advances in Neural Information Processing Systems**, 1867-1875.

[26] SAIGO, H., VERT, J. P. and AKUTSU, T., (2006): Optimizing amino acid substitution matrices with a local alignment kernel. **BMC Bioinformatics**, 7, 246.

[27] SCHULTZ, M. and JOACHIMS, T., (2004): Learning a Distance Metric from Relative Comparisons. **Advances in Neural Information Processing Systems (NIPS)**, 41.

[28] SHALEV-SHWARTZ, S., SINGER, Y. and NG, A. Y., (2004): Online and batch learning of pseudo-metrics. **Proceedings of the Twenty-First International Conference on Machine Learning**, 94. ACM.

[29] SHALEV-SHWARTZ, S. and ZHANG, T., (2013): Stochastic dual coordinate ascent methods for regularized loss. **The Journal of Machine Learning Research**, 14, 567-599.

[30] SHENTAL, N., HERTZ, T., WEINSHALL, D. and PAVEL, M., (2002): Adjustment Learning and Relevant Component Analysis. **Springer Berlin Heidelberg**, 2353, 776-790.

[31] TSENG, P., (2001): Convergence of a block coordinate descent method for nondifferentiable minimization. **Journal of Optimization Theory and Applications**, 109, 475-494.

[32] WEINBERGER, K. Q. and SAUL, L. K., (2009): Distance metric learning for large margin nearest neighbor classification. **The Journal of Machine Learning Research**, 10, 207-244.

[33] XING, E. P., JORDAN, M. I., RUSSELL, S. and NG, A. (2002): Distance metric learning with application to clustering with side-information. **Advances in Neural Information Processing Systems**, 505-512.

[34] XIONG, H. and CHEN, X. W., (2006): Kernel-based distance metric learning for microarray data classification. **BMC Bioinformatics**, 7, 299.

[35] YUAN, G. X., HO, C. H. and LIN, C. J., (2012): An improved glmnet for l1-regularized logistic regression. **The Journal of Machine Learning Research**, 13, 1999-2030.