基于核函数的稀疏属性选择算法

张善文 文国秋 张乐园 李佳烨

(广西师范大学计算机科学与信息工程学院广西多源信息挖掘与安全重点实验室 广西 桂林 541004)

摘 要 鉴于传统属性选择算法无法捕捉属性之间的关系的问题,文中提出了一种非线性属性选择方法。该方法通过引入核函数,将原始数据集投影到高维的核空间,因在核空间内进行运算,进而可以考虑到数据属性之间的关系。由于核函数自身的优越性,即使数据通过高斯核投影到无穷维的空间中,计算复杂度亦可以控制得较小。在正则化因子的限制上,使用两种范数进行双重约束,不仅提高了算法的准确率,而且使得算法实验结果的方差仅为0.74,远小于其他同类对比算法,且算法更加稳定。在8个常用的数据集上将所提算法与6个同类算法进行比较,并用SVM分类器来测试分类准确率,最终该算法得到最少1.84%,最高3.27%,平均2.75%的提升。

关键词 核函数,属性选择,稀疏,L_1 范数,$L_{2,1}$ 范数

中图法分类号 TP181 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2019.02.010

Sparse Feature Selection Algorithm Based on Kernel Function

ZHANG Shan-wen WEN Guo-qiu ZHANG Le-yuan LI Jia-ye

(Guangxi Key Lab of Multi-source Information Mining & Security, College of Computer Science and Information Engineering, Guangxi Normal University, Guilin, Guangxi 541004, China)

Abstract In view of the condition that the traditional feature selection algorithm can not capture the relationship between features, a nonlinear feature selection method was proposed. By introducing a kernel function, the method projects the original data set into a high-dimensional kernel space, and considers the relationship between sample features by performing operations in the kernel space. Due to the superiority of the kernel function, even if the data are projected into the infinite dimensional space through the Gaussian kernel, the computational complexity can be controlled to a small extent. For the limitation of the regularization factor, the use of two norms for double constraint not only improves the accuracy of the algorithm, but also makes the variance of the algorithm only be 0.74, which is much smaller than other similar comparison algorithms, and it is more stable. 6 similar algorithms were compared on 8 common data sets, and the SVM classifier was used to test the effect. The results demonstrate that the proposed algorithm can get the improvement by a minimum of 1.84%, a maximum of 3.27%, and an average of 2.75%.

Keywords Kernel function, Feature selection, Sparse, L_1-norm, $L_{2,1}$-norm

1 引言

在现代计算机中,随着大数据和社会的高速发展,原始数据的维度往往非常巨大。对于这样的高维数据,存储、应用以及传输的代价都极为昂贵。而最为关键的是,真正有效的属性所占比重并不高。为了能够有效地使用高维数据、释放存储空间、减少相对处理时间,对于高维数据,去除冗余属性、约减维度、提取重要属性,成为当前数据挖掘领域的一个重要问题。

常用的属性约减方法有两种:子空间学习和属性选择(下文简称"属选")方法[1-2]。其中,子空间学习法[3]一般是将高维空间向低维空间投影,从而获得特定映射,常见方法有局部保留投影[4]、主成分分析[5]算法等。属性选择算法,即从高维的数据中选取其中一个具有代表性的子集,这个子集包含了原始数据绝大多数重要的属性。在得到该子集的过程中,我们虽然去除了部分属性,但是由于去除的绝大部分都是冗余属性,因此之后的实验预测精度一般不会降低。更可贵的是,由于去除的冗余属性中包含相当一部分的噪音,因此实验分类准确度可能还会相应提升。其常见的方法有:t-test检验法[6]以及系数逻辑回归法[7]等。早期的属性选择算法一般首先使用一些度量的方法,如协方差、拉普拉斯算子[8]等,通过估算的方式来获得每个属性的权值,之后将得到的权值依照

到稿日期:2018-08-03 返修日期:2018-09-30 本文受国家自然科学基金(61170131,61263035,61573270,90718020),中国博士后基金(2015M570837),广西自然科学基金(2015GXNSFCB139011,2015GXNSFAA139306),国家重点研发计划资助项目(2016YFB1000905),广西科技基地与人才计划项目(Guike 541804573)资助。

张善文(1991-),男,硕士生,主要研究方向为数据挖掘、机器学习;**文国秋**(1987-),女,硕士,讲师,主要研究方向为基础数学、机器学习,E-mail:wenguoqiu2008@163.com(通信作者);**张乐园**(1995-),男,硕士生,主要研究方向为数据挖掘、机器学习;**李佳烨**(1993-),男,硕士生,主要研究方向为数据挖掘、机器学习。

其重要程度的不同进行重排，以此来筛选属性。

近期较为流行的属性选择的方法一般都是通过 L_1，$L_{2,1}$ 和 L_2 等范数来对目标函数式中的正则化因子进行稀疏处理，再根据正则化因子中元素的稀疏程度进行属性重要度探究，从而实现属选[9-10]。但是，因为其损失函数基于线性，所以往往不会考虑到数据属性之间的关系。因此，当数据属性之间有较强关系时，会损失很多有用的信息，这样在判断属性取舍时就会出现偏差，不能得到满意的效果。特别是在稳定性方面，从实验结果中可以看到，本文方法的标准差最小，稳定性最好。

研究表明，属性选择方法具有很好的可解释性，而核函数已在众多领域（如支持向量机）取得极好的效果。以支持向量机为例，原始数据通过核变换被投入高维空间。在原始空间中，对于无法线性可分的数据，在足够高维的空间中总是可以找到一个超平面将其分开，实现了高维空间的超平面可分。本文将核函数应用到属性选择算法中对原始数据进行核函数处理，获得非线性的损失函数，从而获取更加显著的属性选择效果。

由此，本文提出一种 SFSK（Sparse Feature Selection Kernel）属性选择算法。该算法将把原始空间的数据通过核函数映射到高维空间，并进行非线性运算，以考虑到数据各个属性之间的关系[11]，从而可以捕捉到更多的有用信息，提高算法的精确度，这是普通线性属选难以实现的。

2　相关算法背景及简介

2.1　符号注释

在本文中，$\boldsymbol{X}$ 矩阵的第 i 行和第 j 列分别使用符号 $\boldsymbol{X}^i$ 和 $\boldsymbol{X}_j$ 来表示，$\boldsymbol{W}_{(i)}$ 表示第 i 次迭代的 $\boldsymbol{W}$，$\boldsymbol{X}$ 矩阵的 F 范数表示为 $\|\boldsymbol{X}\|_F$，并且 $\|\boldsymbol{X}\|_F=\sqrt{\sum_i \|\boldsymbol{X}^i\|_2^2}$，$\nabla f(\alpha)$ 为 $f(\alpha)$ 对 α 的导数。文中各矩阵的维度分别如下：

核函数化前：$\boldsymbol{X}\in\mathbb{R}^{n\times d}$，$\boldsymbol{Y}\in\mathbb{R}^{n\times c}$，$\boldsymbol{X}\in\mathbb{R}^{d\times c}$。

核函数化后：$\boldsymbol{K}^{(i)}\in\mathbb{R}^{n\times n}$，$\boldsymbol{Y}\in\mathbb{R}^{n\times c}$，$\boldsymbol{W}\in\mathbb{R}^{n\times c}$，$\alpha\in\mathbb{R}^{d\times 1}$。

2.2　传统算法

在传统的属性选择中，目标函数一般如下：

$$\min_{\boldsymbol{W}} \|\boldsymbol{Y}-\boldsymbol{X}\boldsymbol{W}\|_F^2+\alpha\|\boldsymbol{W}\|_{2,p}+\beta\phi(\boldsymbol{W}) \tag{1}$$

其中，$\boldsymbol{X}$ 是数据集，$\boldsymbol{Y}$ 是标签，$\phi(\boldsymbol{W})$ 是正则化因子，$\|\boldsymbol{Y}-\boldsymbol{X}\boldsymbol{W}\|_F^2$ 为损失函数，α 为调优参数。对于 $\boldsymbol{W}$ 的限制，一般通过不同的范数来实现，其中，L_0 范数可以直接得到最优解，但是优化求解 L_0 是 NP 难问题，暂时无法解决。L_1 范数是 L_0 范数的最优凸近似，L_1 范数的优化过程虽然比较困难，但仍有许多办法，比如软阈值等，其结果可以直接得到 0 元素。对于 L_2 范数，因其光滑且是凸函数，所以优化相对简单，直接求导即可。虽然 L_2 可以使元素变得很小，但是一般不会直接得到 0 元素。而 $L_{2,1}$ 范数可以使行元素直接得到 0，从而实现属性选择。

通常对于损失函数，需求出 $\boldsymbol{W}$ 的最小值：

$$\min_{\boldsymbol{W}} \|\boldsymbol{Y}-\boldsymbol{X}\boldsymbol{W}\|_F^2 \tag{2}$$

对 $\boldsymbol{W}$ 进行求导，再令导数为 0，即：

$$-\boldsymbol{X}^{\mathrm{T}}\boldsymbol{Y}+\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}\boldsymbol{W}=0 \tag{3}$$

$$\Leftrightarrow \boldsymbol{W}=(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X})^{-1}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{Y} \tag{4}$$

式(4)中有一个明显的问题需要解决，即如果 $\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}$ 不可逆，则 $\boldsymbol{W}$ 无法求出。针对这一问题，一般给出的解决方案是给 $\boldsymbol{W}$ 加上稀疏限制（以 L_2 范数为例），则式(1)变为：

$$\min_{\boldsymbol{W}} \|\boldsymbol{Y}-\boldsymbol{X}\boldsymbol{W}\|_F^2+\lambda\|\boldsymbol{W}\|_2^2 \tag{5}$$

因为使用了 L_2 范数，式(5)又被称为“岭回归”。对 $\boldsymbol{W}$ 求导并令导数为 0，可得：

$$-\boldsymbol{X}^{\mathrm{T}}\boldsymbol{Y}+\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}\boldsymbol{W}+\lambda\boldsymbol{W}=0\Leftrightarrow\boldsymbol{W}=(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}+\lambda\boldsymbol{I})^{-1}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{Y} \tag{6}$$

因为 $\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}$ 为半正定矩阵，$\lambda\boldsymbol{I}$ 是正定矩阵，所以 $\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}+\lambda\boldsymbol{I}$ 一定为可逆矩阵。

2.3　核函数

2.3.1　核函数介绍

核函数的定义为：核本质为函数 K，对于所有 $\boldsymbol{x}_1,\boldsymbol{x}_2\in\boldsymbol{X}$，均能满足核函数式，即：$\boldsymbol{x}_1,\boldsymbol{x}_2\in\boldsymbol{X}$，$K(\boldsymbol{x}_1,\boldsymbol{x}_2)=\langle\Gamma(\boldsymbol{x}_1),\Gamma(\boldsymbol{x}_2)\rangle$。其中，$\Gamma(\cdot)$是从原始输入空间 $\boldsymbol{X}$ 到内积特征空间 Γ 的映射，$\langle\cdot,\cdot\rangle$表示内积，$K\langle\cdot,\cdot\rangle$就是对应的核函数。

为了实现非线性变换，首先找到某个特征映射 Γ，由 Γ 映射得到 $\boldsymbol{X}\rightarrow\boldsymbol{F}$。即原始数据 $\boldsymbol{X}$ 从原始空间通过 Γ 映射到某个特征空间变成数据 $\boldsymbol{F}$。在特征空间中，原本线性不可分的数据可以找到超平面进行分割。但是，这里有一个显著的问题，即维度爆炸。因为特征映射 Γ，当 $\boldsymbol{X}$ 映射到特征空间中之后，原本的属性维度会被映射成极高维度，甚至无穷维度（比如高斯核）。这种情况会造成数据难以处理，甚至不可计算。

如果不使用核函数，我们需先计算线性映射 $\Gamma(\boldsymbol{x}_1)$ 和 $\Gamma(\boldsymbol{x}_2)$，然后再计算出这 2 个特征的内积；若使用核函数，直接计算出 $\Gamma(\boldsymbol{x}_1)$ 和 $\Gamma(\boldsymbol{x}_2)$ 的表达式 $K(\boldsymbol{x}_1,\boldsymbol{x}_2)=\langle\Gamma(\boldsymbol{x}_1),\Gamma(\boldsymbol{x}_2)\rangle$，就可以得到想要的结果，而且这个表达式往往非常简单，大大简化了计算。至此，我们已不必再关心特征映射 Γ 的形式，也不必担心映射之后的维度，只需要使用合适的核函数即可[12]。

在时间复杂度上，以一种多项式核为例：

$$K(\boldsymbol{x},\boldsymbol{z})=(\boldsymbol{x}^{\mathrm{T}}\boldsymbol{z})^2 \tag{7}$$

对式(7)展开得到：

$$K(\boldsymbol{x},\boldsymbol{z})=(\boldsymbol{x}^{\mathrm{T}}\boldsymbol{z})^2=(\sum_{i=1}^{n}\boldsymbol{x}_i\boldsymbol{z}_i)(\sum_{j=1}^{n}\boldsymbol{x}_j\boldsymbol{z}_j) \tag{8}$$

$$\Leftrightarrow\sum_{i=1}^{n}\sum_{j=1}^{n}(\boldsymbol{x}_i\boldsymbol{x}_j)(\boldsymbol{z}_i\boldsymbol{z}_j)=\langle\Gamma(\boldsymbol{x}),\Gamma(\boldsymbol{x})\rangle \tag{9}$$

由式(8)和式(9)可以得出，只计算原始向量内积平方的时间复杂度为 $O(n)$，而计算映射后特征的内积的复杂度为 $O(n^2)$。虽然不同的核函数的计算复杂度不同，但由此例可以明显看出，核函数对于精简时间复杂度有巨大优势。

2.3.2　核函数用法实例

对于数据集 $\boldsymbol{X}\in\mathbb{R}^{n\times d}$（$n$ 为样本数量，d 为属性数量），我们按照属性来拆分样本。因此，$\boldsymbol{X}$ 变为$(\boldsymbol{x}_1,\boldsymbol{x}_2,\boldsymbol{x}_3,\cdots,\boldsymbol{x}_d)$，取其中一个 $\boldsymbol{x}_i\in\mathbb{R}^{n\times 1}(i=1,2,\cdots,d)$，再对 $\boldsymbol{x}_i$ 之间的每个元素做核变换。这里以高斯核为例，高斯核：$k(\boldsymbol{x}^i,\boldsymbol{x}^j)=\mathrm{e}^{-\|\boldsymbol{x}^i-\boldsymbol{x}^j\|^2/2\sigma^2}$。即：$k(\boldsymbol{x}_i^1,\boldsymbol{x}_i^1)$，$k(\boldsymbol{x}_i^1,\boldsymbol{x}_i^2)$，$k(\boldsymbol{x}_i^1,\boldsymbol{x}_i^3)$，$\cdots$，由此原本的属性样本 $\boldsymbol{x}_i$ 依次通过核变换得到如下核矩阵：

$$\begin{bmatrix} k(\boldsymbol{x}_i^1,\boldsymbol{x}_i^1) & \cdots & k(\boldsymbol{x}_i^1,\boldsymbol{x}_i^n) \\ \vdots & \ddots & \vdots \\ k(\boldsymbol{x}_i^n,\boldsymbol{x}_i^1) & \cdots & k(\boldsymbol{x}_i^n,\boldsymbol{x}_i^n) \end{bmatrix} \tag{10}$$

由此，取一个数据集 $\boldsymbol{X}$，原来的 d 个属性向量 $\boldsymbol{x}_i(\boldsymbol{x}_i\in\mathbb{R}^{n\times 1}(i=1,2,\cdots,d))$转换成 d 个核矩阵 $\boldsymbol{K}_i(\boldsymbol{K}_i\in\mathbb{R}^{n\times n},i=1,2,\cdots,d)$。

3 算法描述与优化

3.1 目标函数

通过式(7)对原线性损失函数 $\|\boldsymbol{Y}-\boldsymbol{XW}\|_F^2$ 中的 $\boldsymbol{X}$ 进行核函数处理,因需求得 $\boldsymbol{W}$ 的最小值,故原损失函数变为:

$$\min_{\boldsymbol{W}}\|\boldsymbol{Y}-\sum_{i=1}^{d}\boldsymbol{K}^{(i)}\boldsymbol{W}\|_F^2 \tag{11}$$

显然,式(8)中 $\boldsymbol{W}$ 已经不能做到对数据集的属性选择。因此,需要引入新的变量 $\boldsymbol{\alpha}$,并且使用 L_1 范数来对 $\boldsymbol{\alpha}$ 进行稀疏限制。因此,得到新的公式:

$$\min_{\boldsymbol{W},\alpha}\|\boldsymbol{Y}-\sum_{i=1}^{d}\boldsymbol{\alpha}_i\boldsymbol{K}^{(i)}\boldsymbol{W}\|_F^2+\lambda\|\boldsymbol{\alpha}\|_1 \tag{12}$$

其中,λ 为调节参数,用于控制 $\boldsymbol{\alpha}$ 的稀疏性。为了使得整个目标函数的收敛性效果达到最好,从而使整个算法的稳定性达到最佳,尽量使 $\boldsymbol{W}$ 稀疏。因此,对 $\boldsymbol{W}$ 赋予两个正则项 $L_{2,1}$ 和 L_1 来进行约束,至此,得到如下目标函数:

$$F(\boldsymbol{\alpha})=\min_{\boldsymbol{W},\alpha}\|\boldsymbol{Y}-\sum_{i=1}^{d}\boldsymbol{\alpha}_i\boldsymbol{K}^{(i)}\boldsymbol{W}\|_F^2+\lambda_1\|\boldsymbol{W}\|_1+\lambda_2\|\boldsymbol{W}\|_{2,1}+\lambda_3\|\boldsymbol{\alpha}\|_1 \tag{13}$$

其中,参数 λ_1 用于调节 $\boldsymbol{W}$ 的整体稀疏性,λ_2 用于调节 $\boldsymbol{W}$ 的整行稀疏性,λ_3 用于调节 $\boldsymbol{\alpha}$ 的稀疏性。L_1 范数已被证明可以使回归结果生成稀疏的回归稀疏[13]。因此,λ_3 的大小即决定了 $\boldsymbol{\alpha}$ 的稀疏程度。当 $\boldsymbol{\alpha}$ 中某个元素为 0,即表示不选择这个核矩阵(属性);当 $\boldsymbol{\alpha}$ 中某个元素不为 0 则表示选取该属性。

3.2 算法优化

$\boldsymbol{\alpha}$ 与 $\boldsymbol{W}$ 均为独立的变量,所使用的稀疏限制也不相同,因为核函数的加入,它们在化简与优化方面也与常见的算法有较大差别。并且根据实际情况的不同,为了使得算法总体收敛得更快,对 $\boldsymbol{\alpha}$ 和 $\boldsymbol{W}$ 选择不同的优化方式。

3.2.1 固定 $\boldsymbol{\alpha}$,优化 $\boldsymbol{W}$

令 $\boldsymbol{P}=\sum_{i=1}^{d}\boldsymbol{\alpha}_i\boldsymbol{K}^{(i)}$,对目标函数(13)进行化简:

$$\min_{\boldsymbol{W}}\|\boldsymbol{Y}-\sum_{i=1}^{d}\boldsymbol{\alpha}_i\boldsymbol{K}^{(i)}\boldsymbol{W}\|_F^2+\lambda_1\|\boldsymbol{W}\|_1+\lambda_2\|\boldsymbol{W}\|_{2,1}$$
$$\Leftrightarrow\min_{\boldsymbol{W}}\|\boldsymbol{Y}-\boldsymbol{PW}\|_F^2+\lambda_1\|\boldsymbol{W}\|_1+\lambda_2\|\boldsymbol{W}\|_{2,1} \tag{14}$$

虽然式(14)是凸函数,但是 L_1,$L_{2,1}$ 范数对 $\boldsymbol{W}$ 的正则化都是非光滑的,为此,本文设计一种方法来对 $\boldsymbol{W}$ 快速求解[14]。具体步骤如下:

首先对 $w_i(1\leqslant i\leqslant c)$ 求偏导,并且令偏导数为 0,得到:

$$\boldsymbol{PP}^{\mathrm{T}}w_i-\boldsymbol{P}\boldsymbol{y}_i+\lambda_1\boldsymbol{D}_iw_i+\lambda_2\hat{\boldsymbol{D}}w_i=0 \tag{15}$$

$\boldsymbol{D}_i(1\leqslant i\leqslant c)$ 是一个对角矩阵,它的每个元素都为 $\frac{1}{2\|w_i\|}$;$\hat{\boldsymbol{D}}$ 也为一个对角矩阵,它的元素为 $\frac{1}{2\|w^i\|_2}$。之后可以得到最终的结果:

$$w_i=(\boldsymbol{PP}^{\mathrm{T}}+\lambda_1\boldsymbol{D}_i+\lambda_2\hat{\boldsymbol{D}})^{-1}\boldsymbol{P}y_i \tag{16}$$

因为 $\boldsymbol{D}$ 和 $\hat{\boldsymbol{D}}$ 均依赖于 $\boldsymbol{W}$,所以需要依赖迭代算法来求解最优的 $\boldsymbol{W}$。本文算法的收敛性证明如下:

根据式(16),可以得到:

$$\boldsymbol{W}^{(t+1)}=\min_{w}\mathrm{Tr}(\boldsymbol{P}^{\mathrm{T}}\boldsymbol{W}-\boldsymbol{Y})^{\mathrm{T}}+\lambda_1\sum_{1}^{c}w_i^{\mathrm{T}}\boldsymbol{D}_i^{(t)}w_i+\lambda_2\mathrm{Tr}\,\boldsymbol{W}^{\mathrm{T}}\hat{\boldsymbol{D}}^{(t)}\boldsymbol{W} \tag{17}$$

因此,可以得到如下收敛步骤:

$$\mathrm{Tr}(\boldsymbol{P}^{\mathrm{T}}\boldsymbol{W}^{(t+1)}-\boldsymbol{Y})^{\mathrm{T}}(\boldsymbol{P}^{\mathrm{T}}\boldsymbol{W}^{(t+1)}-\boldsymbol{Y})+\lambda_1\sum_{i=1}^{c}(w_i^{(t+1)})^{\mathrm{T}}\boldsymbol{D}_i^{(t)}w_i^{(t+1)}+\lambda_2\mathrm{Tr}(\boldsymbol{W}^{(t+1)})^{\mathrm{T}}\hat{\boldsymbol{D}}^{t}\boldsymbol{W}^{(t+1)}\leqslant\mathrm{Tr}(\boldsymbol{P}^{\mathrm{T}}\boldsymbol{W}^{(t)}-\boldsymbol{Y})^{\mathrm{T}}(\boldsymbol{P}^{\mathrm{T}}\boldsymbol{W}^{(t)}-\boldsymbol{Y})+\lambda_1\sum_{i=1}^{c}(w_i^{(t)})^{\mathrm{T}}\boldsymbol{D}_i^{(t)}w_i^{(t)}+\rho_2\mathrm{Tr}(\boldsymbol{W}^{(t)})^{\mathrm{T}}\hat{\boldsymbol{D}}^{t}\boldsymbol{W}^{(t)}$$

$$\Rightarrow\mathrm{Tr}(\boldsymbol{P}^{\mathrm{T}}\boldsymbol{W}^{(t+1)}-\boldsymbol{Y})^{\mathrm{T}}(\boldsymbol{P}^{\mathrm{T}}\boldsymbol{W}^{(t+1)}-\boldsymbol{Y})+\lambda_1\sum_{i=1}^{d}\sum_{j=1}^{c}(\frac{(w_{ij}^{(t+1)})^2}{2\|w_{ij}^{(t)}\|}-\|w_{ij}^{(t+1)}\|+\|w_{ij}^{(t+1)}\|)+\lambda_2\sum_{k=1}^{d}(\frac{\|(w^{(t+1)})^k\|_2^2}{2\|(w^{(t)})^k\|_2}-\|(w^{(t+1)})^k\|_2+\|(w^{(t+1)})^k\|_2)$$

$$\leqslant\mathrm{Tr}(\boldsymbol{P}^{\mathrm{T}}\boldsymbol{W}^{(t)}-Y)^{\mathrm{T}}(\boldsymbol{P}^{\mathrm{T}}\boldsymbol{W}^{(t)}-Y)+\rho_1\sum_{i=1}^{d}\sum_{j=1}^{m}(\|w_{ij}^{(t)}\|+\frac{(w_{ij}^{(t)})^2}{2\|w_{ij}^{(t+1)}\|}-\|w_{ij}^{(t)}\|)+\rho_2\sum_{k=1}^{d}(\|(w^{(t)})^k\|_2+\frac{\|(w^{(t)})^k\|_2^2}{2\|(w^{(t)})^k\|_2}-\|(w^{(t)})^k\|_2)$$

$$\Rightarrow\mathrm{Tr}(\boldsymbol{P}^{\mathrm{T}}\boldsymbol{W}^{(t+1)}-\boldsymbol{Y})^{\mathrm{T}}(\boldsymbol{P}^{\mathrm{T}}\boldsymbol{W}^{(t+1)}-\boldsymbol{Y})+\lambda_1\sum_{i=1}^{d}\sum_{j=1}^{m}\|w_{ij}^{(t+1)}\|+\lambda_2\sum_{k=1}^{d}\|(w^{(t+1)})^k\|_2$$
$$\leqslant\mathrm{Tr}(\boldsymbol{P}^{\mathrm{T}}\boldsymbol{W}^{(t)}-\boldsymbol{Y})^{\mathrm{T}}(\boldsymbol{P}^{\mathrm{T}}\boldsymbol{W}^{(t)}-\boldsymbol{Y})+\rho_1\sum_{i=1}^{d}\sum_{j=1}^{m}\|w_{ij}^{(t)}\|+\rho_2\sum_{k=1}^{d}\|(w^{(t)})^k\|_2$$

根据文献[15],对于任意的向量 w 和 w_0,均可以得到 $\|w_2\|-\frac{\|w\|_2^2}{2\|w_0\|_2}\leqslant\|w_0\|_2-\frac{\|w_0\|_2^2}{2\|w_0\|_2}$,因此本算法在迭代中每次都会减少目标的值,最终使其收敛。

$\boldsymbol{W}^{(t)}$,$\boldsymbol{D}_i^{(t)}(1\leqslant i\leqslant c)$ 以及 $\tilde{\boldsymbol{D}}^{(t)}$ 满足式(16),并且因为目标函数为凸函数,所以局部最优解即为全局最优解。

3.2.2 固定 $\boldsymbol{W}$,优化 $\boldsymbol{\alpha}$

在下文中需对 $f(\boldsymbol{\alpha})$ 的求导,故固定 $\boldsymbol{W}$ 可忽略关于 $\boldsymbol{W}$ 的正则项。原式(13)中 $F(\boldsymbol{\alpha})$ 可分为如下两个部分:

$$F(\boldsymbol{\alpha})=f(\boldsymbol{\alpha})+\lambda_3\|\boldsymbol{\alpha}\|_1 \tag{18}$$

$$f(\boldsymbol{\alpha})=\sum_{i=1}^{n}\boldsymbol{y}_i\boldsymbol{y}_i^{\mathrm{T}}-2\boldsymbol{\alpha}^{\mathrm{T}}\sum_{i=1}^{n}\boldsymbol{S}^{(i)}\boldsymbol{y}_i^{\mathrm{T}}+\boldsymbol{\alpha}^{\mathrm{T}}\sum_{i=1}^{n}(\boldsymbol{S}^{(i)}(\boldsymbol{S}^{(i)})^{\mathrm{T}})\boldsymbol{\alpha} \tag{19}$$

获得 $f(\alpha)$ 的具体步骤如下:

$$\min_{\boldsymbol{\alpha}}\|\boldsymbol{Y}-\sum_{i=1}^{d}\boldsymbol{\alpha}_i\boldsymbol{K}^{(i)}\boldsymbol{W}\|_F^2 \tag{20}$$
$$\Leftrightarrow\min_{\boldsymbol{\alpha}}\|\boldsymbol{Y}-(\alpha_1\boldsymbol{K}^{(1)}\boldsymbol{W}+\cdots+\alpha_d\boldsymbol{K}^{(d)}\boldsymbol{W})\|_F^2 \tag{21}$$

由 $\boldsymbol{Q}^{(i)}=\boldsymbol{K}^{(i)}\boldsymbol{W}\in\mathbb{R}^{n\times c}$,得到:

$$\Leftrightarrow\min_{\boldsymbol{\alpha}}\|\boldsymbol{Y}-(\boldsymbol{\alpha}_1\boldsymbol{Q}^{(1)}+\cdots+\boldsymbol{\alpha}_d\boldsymbol{Q}^{(d)})\|_F^2 \tag{22}$$

$$\Leftrightarrow\min_{\alpha}\sum_{i=1}^{n}\|\boldsymbol{y}_i-(\boldsymbol{\alpha}_1\boldsymbol{q}_{i,\cdot}^{(1)}+\cdots+\boldsymbol{\alpha}_d\boldsymbol{q}_{i,\cdot}^{(d)})\|_2^2,\boldsymbol{q}_{i,\cdot}^{(1)}\in\mathbb{R}^{1\times c} \tag{23}$$

$$\Leftrightarrow\min_{\boldsymbol{\alpha}}\sum_{i=1}^{n}\|\boldsymbol{y}_i-\boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{S}^{(i)})\|_2^2,\boldsymbol{S}^{(i)}\in\mathbb{R}^{d\times c} \tag{24}$$

$$\Leftrightarrow\min_{\boldsymbol{\alpha}}\sum_{i=1}^{n}\|\boldsymbol{y}_i-\boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{S}^{(i)}\|_2^2,\boldsymbol{S}^{(i)}=\begin{pmatrix}\boldsymbol{q}_{i,1}^{(1)} & \boldsymbol{q}_{i,c}^{(1)}\\ \boldsymbol{q}_{i,1}^{(d)} & \boldsymbol{q}_{i,c}^{(d)}\end{pmatrix}\in\mathbb{R}^{d\times c} \tag{25}$$

$$\Leftrightarrow\min_{\boldsymbol{\alpha}}\sum_{i=1}^{n}\|\boldsymbol{y}_i^{\mathrm{T}}-(\boldsymbol{S}^{(i)})^{\mathrm{T}}\boldsymbol{\alpha}\|_2^2 \tag{26}$$

$$\Leftrightarrow\min_{\boldsymbol{\alpha}}\sum_{i=1}^{n}(\boldsymbol{y}_i\boldsymbol{y}_i^{\mathrm{T}}-2\boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{S}^{(i)}\boldsymbol{y}_i^{\mathrm{T}}+\boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{S}^{(i)}(\boldsymbol{S}^{(i)})^{\mathrm{T}}\boldsymbol{\alpha}) \tag{27}$$

$$\Leftrightarrow\min_{\boldsymbol{\alpha}}\sum_{i=1}^{n}\boldsymbol{y}_i\boldsymbol{y}_i^{\mathrm{T}}-2\boldsymbol{\alpha}^{\mathrm{T}}\sum_{i=1}^{n}\boldsymbol{S}^{(i)}\boldsymbol{y}_i^{\mathrm{T}}+\boldsymbol{\alpha}^{\mathrm{T}}\sum_{i=1}^{n}(\boldsymbol{S}^{(i)}(\boldsymbol{S}^{(i)})^{\mathrm{T}})\boldsymbol{\alpha} \tag{28}$$

可以注意到,式(18)中,$f(\alpha)$ 是凸的并且可微。而 $\lambda_3\|\boldsymbol{\alpha}\|_1$ 虽然也是凸的,但并不光滑,所以无法直接优化。因此这里使用 PGD[15] 算法来优化 $\boldsymbol{\alpha}$。

在每一次的迭代中,更新 $\boldsymbol{\alpha}$ 值的问题即是下式的最优化问题。

$$\boldsymbol{\alpha}_{t+1}=\arg\min_{\boldsymbol{w}} \boldsymbol{G}_{\eta(t)}(\boldsymbol{\alpha},\boldsymbol{\alpha}_t) \tag{29}$$

其中，$\boldsymbol{G}_{\eta(t)}(\boldsymbol{W},\boldsymbol{W}(t))$定义为：

$$\boldsymbol{G}_{\eta(t)}(\boldsymbol{\alpha},\boldsymbol{\alpha}_t)=f(\boldsymbol{\alpha}_t)+\langle\nabla f(\boldsymbol{\alpha}_t),\boldsymbol{\alpha}-\boldsymbol{\alpha}_t\rangle+\frac{\eta(t)}{2}\|\boldsymbol{\alpha}-\boldsymbol{\alpha}_t\|^2+\lambda_3\|\boldsymbol{\alpha}\|_1 \tag{30}$$

其中，$\eta(t)$为调优参数，t 为迭代次数。

本文中$\nabla f(\boldsymbol{\alpha}_t)$定义如下：

$$\nabla(\boldsymbol{\alpha}_t)=-2\sum_{i=1}^{n}\boldsymbol{S}^{(i)}\boldsymbol{x}_i^{\mathrm{T}}+(2\sum_{i=1}^{n}(\boldsymbol{S}^{(i)}(\boldsymbol{S}^{(i)})^{\mathrm{T}})\boldsymbol{\alpha}_t \tag{31}$$

将式(30)中 $\boldsymbol{\alpha}$ 的独立条件忽略，得到：

$$\boldsymbol{\alpha}_{t+1}=\pi_{\eta(t)}(\boldsymbol{\alpha}_t)=\arg\min_{w}\frac{1}{2}\|\boldsymbol{W}-\boldsymbol{U}_t\|_F^2+\frac{\lambda_3}{\eta(t)}\|\boldsymbol{\alpha}\|_1 \tag{32}$$

为了便于计算，上式中 $\boldsymbol{U}_t=\boldsymbol{\alpha}_t-\frac{1}{\eta(t)}\nabla f(\boldsymbol{\alpha}_t)$

由于 $\boldsymbol{\alpha}_{t+1}$[16] 的每行可分性，我们可以按行来分别更新 $\boldsymbol{\alpha}$ 的权重，即：

$$\boldsymbol{\alpha}_{t+1}=\arg\min_{\boldsymbol{\alpha}}\frac{1}{2}\|\boldsymbol{\alpha}^i-u_t^i\|_2^2+\frac{\lambda}{\eta(t)}\|\boldsymbol{\alpha}^i\|_2 \tag{33}$$

由此，可以最终得到式(18)的闭式解：

$$\boldsymbol{\alpha}_{t+1}^i=\begin{cases}u^i-\lambda/\eta, & \lambda/\eta<u^i\\ 0, & |u^i|\leqslant\lambda_3/\eta\\ u^i+\lambda/\eta, & u^i<-\lambda_3/\eta\end{cases} \tag{34}$$

根据文献[17]中的定理，设$\{\boldsymbol{W}(t)\}$是由优化算法所生成的序列，并且对于 $\forall t\geqslant 1$，可推导得出：

$$\partial(\boldsymbol{\alpha}(t))-\partial(\boldsymbol{\alpha}*)\leqslant\frac{2\gamma L\|\boldsymbol{\alpha}(1)-\boldsymbol{\alpha}^*\|_F^2}{(t+1)^2} \tag{35}$$

其中，γ 是为正的预定义常数，L 是 $f(\boldsymbol{\alpha})$中梯度的李普希茨常数，$\boldsymbol{W}^*=\arg\min_{\boldsymbol{w}}\partial(\boldsymbol{W})$。从而可知近梯度算法的收敛率为 $O(\frac{1}{t^2})$，t 即为算法迭代次数。

本文算法的具体执行伪代码如算法 1 所示。

算法 1　基于核函数的稀疏属性选择算法

输入：数据集 $\mathbf{X}\in\mathbb{R}^{n\times d}$与标签 $\mathbf{Y}\in\mathbb{R}^{n\times d}$，调优参数 $\lambda_1,\lambda_2,\lambda_3$

输出：特征权值 $\boldsymbol{\alpha}$

1. 初始化 $\mathbf{W}_{(0)}$，通过 $\mathbf{W}_{(0)}$求出 $\mathbf{D}_{(0)}$与$\overset{\wedge}{\mathbf{D}}_{(0)}$。
2. 基于式(7)，将数据 $\mathbf{X}$ 变成 $\mathbf{K}^{(i)}$，i∈1→d。
3. 由 $\mathbf{P}=\sum_{i=1}^{d}\boldsymbol{\alpha}_i\mathbf{K}^{(i)}$得到 $\mathbf{P}$。

Repeat.

4. 通过式(16)更新出 $\mathbf{W}_{(1)}$。
5. 初始化 $\boldsymbol{\alpha}_{(0)}$，把 $\mathbf{W}_{(1)}$与 $\boldsymbol{\alpha}_{(0)}$代入式(16)中。
6. 基于式(21)、式(22)获得 $\mathbf{Q}$。
7. 基于步骤 6 以及式(23)得到 $\mathbf{S}$。
8. 利用 $\mathbf{Q}$ 与 $\mathbf{S}$ 获得 f($\boldsymbol{\alpha}$)，进而求导获得∇f($\boldsymbol{\alpha}$)。
9. set $\eta(t)=\eta(t-1)$，将∇f($\boldsymbol{\alpha}$)代入 $\mathbf{U}$ 计算 $\boldsymbol{\alpha}_{t+1}=\arg\min_{\mathbf{W}}G_{\eta(t)}(\boldsymbol{\alpha},\boldsymbol{\alpha}_t)$。
10. 对步骤 9 不断按行更新，通过式(33)最终得到闭式解式(34)。
11. Until 式(13)收敛。

4　实验结果与分析

4.1　实验数据集与相关对比算法

为了对算法进行充分测试，实验将在 8 个数据集上进行。本文选取的数据集均出自 UCI[18] 数据库。数据集的具体情况如表 1 所列。

表 1　数据集详细信息

Table 1　Detailed information of data set

数据集	样本数	属性数	类别
Arrhythmia	452	279	13
Clean	476	167	2
Colon	62	2000	2
Movements	360	90	15
Ecoli	336	343	8
Sonar	208	60	2
Glass	219	9	6
Yale	165	1024	15

为了更加公平、准确地测量算法在不同数据维度上的表现，在数据集的选择上，使属性维度的区分非常大。其中 Sonar，Movements，Glass 为百维以下的低维数据；Arrhythmia，Clean，Ecoli 为百维以上千维以下的中维数据；Colon，Yale 为千维以上的高维数据。

实验选择 6 种对比算法与本文提出的算法进行比较，选取的对比算法均为经典的或近年来优秀的属性选择算法。RFS[19] 方法：经典的有监督属性选择算法。其损失函数与正则化项均采用 $L_{2,1}$ 范数来进行约束，使得函数具有更好的鲁棒性并且可以有效地区分冗余属性。CSFS[20] 方法：通过凸优化公式来实现属性选择的算法。算法本身是半监督多标签的，但本次实验中传入的数据均含有标签，并且是单标签。MCFS[21] 方法：一种通过子空间学习从而考虑到属性之间的关系的半监督属性选择方法。HSICLasso[22] 方法：一种通过 L_1 正则化来捕获输入数据之间非线性关系的有监督核函数属性选择算法。SOGFS[23] 方法：通过学习得到一个最优化图的结构，从而进行属性选择的无监督算法。UDFS[24] 方法：通过线性分类器和 $L_{2,1}$ 正则化来得到一个虚拟分类标签的无监督属性选择算法。

4.2　实验与结果分析

本文所有实验都在 Win7 系统下，使用 MATLAB2014a 软件完成。实验将使用 10-fold cross-valind(十折交叉验证)方法把实验数据分为训练集和测试集两部分，再通过使用 LIBSVM[25] 工具箱进行 SVM 分类，以分类的准确率来确定算法属选的效果。为保证实验可靠性，所有实验均在同一环境下进行，最终会得到 10 次运行的实验结果，取其均值再加减均标准差以评估每种算法的性能。每种算法在 8 个数据集上的实验结果对比如图 1－图 8 所示。实验准确率和均值统计结果如表 2 所列。

通过图 1－图 8，我们可以非常直观地看到在每个数据集上，SFSK 算法与各对比算法完成属性选择后的 SVM 分类准确率的比较。

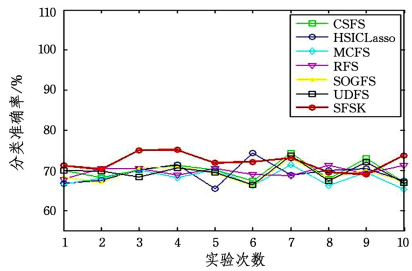

图 1　各算法在 Arrhythmia 上的实验结果

Fig. 1　Experimental results of each algorithm on Arrhythmia

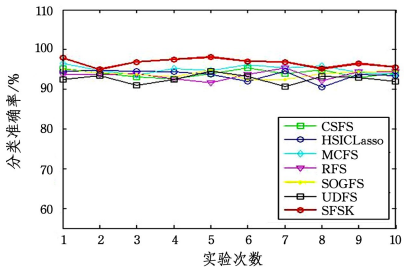

图 2 各算法在 Clean 上的实验结果

Fig. 2 Experimental results of each algorithm on Clean

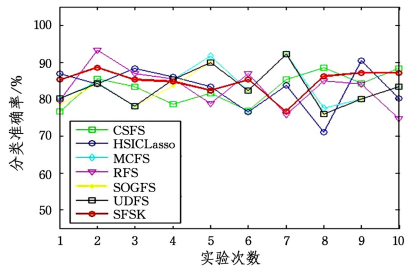

图 3 各算法在 Colon 上的实验结果

Fig. 3 Experimental results of each algorithm on Colon

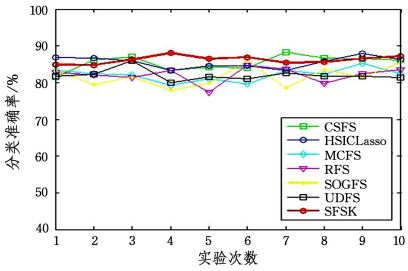

图 4 各算法在 Ecoli 上的实验结果

Fig. 4 Experimental results of each algorithm on Ecoli

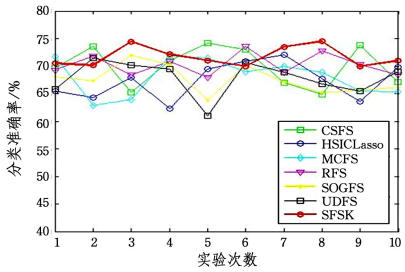

图 5 各算法在 Glass 上的实验结果

Fig. 5 Experimental results of each algorithm on Glass

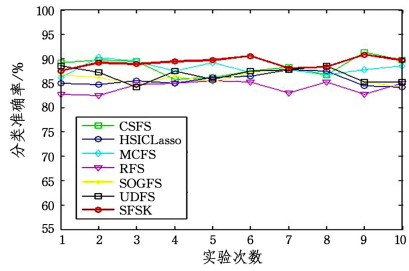

图 6 各算法在 Movements 上的实验结果

Fig. 6 Experimental results of each algorithm on Movements

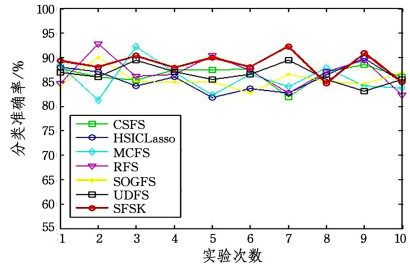

图 7 各算法在 Sonar 上的实验结果

Fig. 7 Experimental results of each algorithm on Sonar

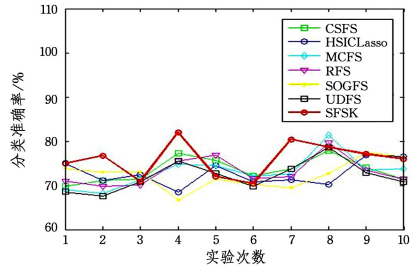

图 8 各算法在 Yale 上的实验结果

Fig. 8 Experimental results of each algorithm on Yale

在低维的 3 个数据集(Sonar,Movements,Glass)上,SFSK 算法大部分情况下的结果优于其他对比算法,这反映出了本文的算法由于通过核函数考虑到了更多的信息,因此在选择属性时更加精准。在中维的 3 个数据集(Arrhythmia,Clean,Ecoli)上,本文算法不仅每次分类准确率都要高过大部分对比算法,而且图形几乎成一条直线,算法的稳定性再一次得到了很好的验证。在高维数据集(Colon,Yale)上,由于冗余属性和噪音属性较多,因此每种算法的属选效果都有不同程度的波动,但是 SFSK 算法的表现依然是最优秀的。

表 2 实验准确率与均值统计结果

Table 2 Statistical results of experimental accuracy and mean values

数据集	CSFS	MCFS	HSICLasso	SOGFS	UDFS	RFS	SFSK
Arrhythmia	69.93±0.41	68.18±0.69	69.24±0.87	69.04±0.78	69.46±0.84	69.72±0.85	**72.07±0.48**
Clean	94.01±0.76	94.98±0.79	93.61±0.63	93.84±0.87	92.60±1.04	93.54±0.83	**96.63±0.36**
Colon	82.90±1.47	83.52±1.49	83.04±1.74	82.83±1.16	83.16±1.15	83.04±1.25	**84.85±1.12**
Movements	88.38±0.64	88.05±1.34	85.66±0.65	86.22±1.41	86.77±1.10	84.19±0.53	**89.22±0.63**
Ecoli	85.37±0.48	82.13±0.96	85.53±0.94	81.65±1.85	81.97±1.85	82.09±0.72	**86.20±0.80**
Sonar	86.52±0.87	85.77±1.33	85.49±1.19	85.51±1.55	86.52±1.31	86.94±1.75	**88.65±1.13**
Glass	69.95±0.95	68.05±2.20	67.33±1.53	67.57±0.89	67.85±1.50	70.21±1.83	**71.71±0.76**
Yale	73.43±1.95	73.12±1.53	72.71±1.32	72.41±1.36	72.07±1.85	73.12±1.66	**75.91±0.69**
平均	81.31±0.94	80.47±1.29	80.32±1.10	79.88±1.23	80.05±1.33	80.35±1.17	**83.15±0.74**

通过表 2 的分析可以得出,与 HSICLasso 算法相比平均提高了 2.83%,证明了本文提出的算法比其他核函数方法更可靠。与 MCFS 方法相比,本文算法的分类准确率平均提高了 2.68%,这说明本方法相较于子空间学习,有更好的提升。与 SOGFS 相比,本文算法的分类准确率提高了 3.27%,与 UDFS 方法相比也提升了 3.10%,这显示出核函数在高维空

间实现的非线性运算较好地考虑到了属性之间的关系。而且由于 $L_{2,1}$ 范数与 L_1 的双重稀疏，效果提升显著。需要特别注意的是，本文方法的标准差为 0.74，在所有算法中最小，而其他 6 个对比算法，除了 CSFS 方法的标准差为 0.94 以外，其他均超过 1.00，这是由于核函数变换充分考虑到了数据属性之间的关系，从而捕捉到了更多高效能线性属性选择算法无法捕捉的信息，这不仅在体现在分类准确率上，更体现在算法的稳定性上。

结束语　本文所提出的核函数属性选择算法 SFSK 通过对原始数据进行核函数处理，实现了数据在高维空间中的线性可分，从而提升了属性选择的效果。实验结果表明，所提算法的属选效果相当显著。在之后的学习中，将尝试把子空间学习、超图、低秩等[26-27]有效手段加入算法中。但因核函数的加入会增加算法复杂度，所以也将尝试使用更好的优化方式来减少算法的时间复杂度与空间复杂度。

参 考 文 献

[1] ZHU X, SUK H I, SHEN D. Matrix-Similarity Based Loss Function and Feature Selection for Alzheimer's Disease Diagnosis[C]// Computer Vision and Pattern Recognition. IEEE, 2014: 3089-3096.

[2] GU Q, LI Z, HAN J. Joint feature selection and subspace learning[C]// International Joint Conference on Artificial Intelligence. AAAI Press, 2011: 1294-1299.

[3] ZHU X, HUANG Z, CHENG H, et al. Sparse hashing for fast multimedia search[J]. Acm Transactions on Information Systems, 2013, 31(2): 1-24.

[4] ZHU X, HUANG Z, YANG Y, et al. Self-taught dimensionality reduction on the high-dimensional small-sized data[J]. Pattern Recognition, 2013, 46(1): 215-229.

[5] PYATYKH S, HESSER J, ZHENG L. Image noise level estimation by principal component analysis[J]. IEEE Transactions on Image Processing A Publication of the IEEE Signal Processing Society, 2013, 22(2): 687-699.

[6] KONIETSCHKE F, PAULY M. Bootstrapping and permuting paired t-test type statistics[J]. Statistics & Computing, 2014, 24(3): 283-296.

[7] LIIMATAINEN K, HEIKKILÄ R, YLIHARJA O, et al. Sparse logistic regression and polynomial modelling for detection of artificial drainage networks[J]. Remote Sensing Letters, 2015, 6(4): 311-320.

[8] BENABDESLEM K, HINDAWI M. Constrained laplacian score for semi-supervised feature selection[C]// Machine Learning and Knowledge Discovery in Databases－European Conference Proceedings. DBLP, 2011: 204-218.

[9] ZHANG S, CHENG D, ZONG M, et al. Self-representation nearest neighbor search for classification[J]. Neurocomputing, 2016, 195(C): 137-142.

[10] DENG Z, ZHANG S, YANG L, et al. Sparse sample self-representation for subspace clustering[J]. Neural Computing & Applications, 2018, 29(11): 43-49.

[11] VARMA M, BABU B R. More generality in efficient multiple kernel learning[C]// International Conference on Machine Learning. ACM, 2009: 1065-1072.

[12] COMANICIU D, RAMESH V, MEER P P. Kernel-Based Object Tracking[J]. Pattern Analysis & Machine Intelligence, 2003, 25(5): 564-575.

[13] GONG Y H, ZONG M, ZHU Y H, et al. Knn regression based on mixed-norm reconstruction[J]. Computer Applications and Software, 2016(2): 232-236. (in Chinese)
龚永红，宗鸣，朱永华，等. 基于混合模重构的 kNN 回归[J]. 计算机应用与软件，2016(2)：232-236.

[14] WANG H, NIE F, HUANG H, et al. Sparse multi-task regression and feature selection to identify brain imaging predictors for memory performance[C]// International Conference on Computer Vision. 2011: 557.

[15] GU Q, LI Z, HAN J. Linear discriminant dimensionality reduction[C]// Joint European Conference on Machine Learning and Knowledge Discovery in Databases. Springer Berlin Heidelberg, 2011: 549-564.

[16] ZHU X, ZHANG L, HUANG Z. A sparse embedding and least variance encoding approach to hashing[J]. IEEE Transactions on Image Processing, 2014, 23(9): 3737-3750.

[17] ZHU X, SUK H I, SHEN D. A Novel Multi-relation Regularization Method for Regression and Classification in AD Diagnosis[C]// International Conference on Medical Image Computing and Computer-Assisted Intervention. Springer International Publishing, 2014: 401-408.

[18] UCI repository of machine learning datasets [EB/OL]. [2016-05-27]. http://archive.icsuci.edu/ml.

[19] NIE F, HUANG H, CAI X, et al. Efficient and robust feature selection via joint $\ell 2,1$ -norms minimization[C]// International Conference on Neural Information Processing Systems. Curran Associates Inc., 2010: 1813-1821.

[20] CHANG X, NIE F, YANG Y, et al. A convex formulation for semi-supervised multi-label feature selection[C]// Twenty-Eighth AAAI Conference on Artificial Intelligence. AAAI Press, 2014: 1171-1177.

[21] CAI D, ZHANG C, HE X. Unsupervised feature selection for multi-cluster data[C]// ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, 2010: 333-342.

[22] YAMADA M, JITKRITTUM W, SIGAL L, et al. High-Dimensional Feature Selection by Feature-Wise Non-Linear Lasso[J]. Neural Computation, 2012, 26(1): 185-207.

[23] NIE F, ZHU W, LI X. Unsupervised feature selection with structured graph optimization[C]// Thirtieth AAAI Conference on Artificial Intelligence. AAAI Press, 2016: 1302-1308.

[24] YANG Y, SHEN H T, MA Z, et al. l 2,1 -norm regularized discriminative feature selection for unsupervised learning[C]// International Joint Conference on Artificial Intelligence. AAAI Press, 2011: 1589-1594.

[25] LIBSVM-ALibrary for Support Vector Machinces [EB/OL]. [2015-04-10]. http://www/csie.ntu.edu.tw/～cjlin/libsvm.

[26] ZHAO Z, HE X, CAI D, et al. Graph Regularized Feature Selection with Data Reconstruction[J]. IEEE Transactions on Knowledge & Data Engineering, 2016, 28(3): 689-700.

[27] XUE H, SONG Y, XU H M. Multiple Indefinite Kernel Learning for Feature Selection[C]// Twenty-Sixth International Joint Conference on Artificial Intelligence. 2017: 3210-3216.