Knowledge-Based Systems 71 (2014) 339–344

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Kernel ridge regression using truncated newton method

Maher Maalouf [a,*], Dirar Homouz [b]

[a] Industrial and Systems Engineering, Khalifa University, P.O. Box 127788, Abu Dhabi, United Arab Emirates
[b] Applied Mathematics and Science, Khalifa University, P.O. Box 127788, Abu Dhabi, United Arab Emirates

ARTICLE INFO

Article history:
Received 31 March 2014
Received in revised form 20 July 2014
Accepted 11 August 2014
Available online 26 August 2014

Keywords:
Regression
Least-squares
Kernel ridge regression
Kernel methods
Truncated Newton

ABSTRACT

Kernel Ridge Regression (KRR) is a powerful nonlinear regression method. The combination of KRR and the truncated-regularized Newton method, which is based on the conjugate gradient (CG) method, leads to a powerful regression method. The proposed method (algorithm), is called Truncated-Regularized Kernel Ridge Regression (TR-KRR). Compared to the closed-form solution of KRR, Support Vector Machines (SVM) and Least-Squares Support Vector Machines (LS-SVM) algorithms on six data sets, the proposed TR-KRR algorithm is as accurate as, and much faster than all of the other algorithms.

1. Introduction

Regression and classification are fundamental machine learning techniques to find patterns in data. Predictive tasks whose outcomes are quantitative (real numbers) are called regression, and tasks whose outcomes are qualitative (binary, categorical, or discrete) are called classification. The most fundamental method to address regression problems is the *least squares* (LS) method, while *logistic regression* (LR) is the fundamental method for classification. Some disadvantages of the LS method (over-fitting and multicollinearity) are addressed through the development of the method of *ridge regression* (RR) [1], which is based on the LS method. Kernel methods are some of the most successful for machine learning in recent years. One of the their advantages is extending linear algorithms to non-linear problems by the implementation of the kernel. *Support vector machines* (SVM), developed originally by Vapnik [2], is considered a state-of-the-art algorithm for both classification (SVC) and regression (SVR) [3] through its implementation of kernels. *Least squares support vector machines* (LS-SVM), developed by Suykens and Vanderwalle [4], is extended to solve regression problems. The LS-SVM method is easier to train and it converts the inequality constraints of SVM into equality constraints [5]. *Kernel ridge regression* (KRR) [6] extends the RR method to non-linear problems and is now an established data mining tool [7].

Each one of aforementioned methods has a limitation. LS linearity may be an obstacle to handling highly nonlinear small-to-medium size data sets [8]. The SVM method requires solving a constrained quadratic optimization problem with a time complexity of $O(N^3)$ where N is the number of training instances. The KRR method, in the form of ridge regression, is not sparse and requires all of the training instances in its model [8]. Like SVM, KRR has a time complexity of $O(N^3)$. Its computation can be slow due to the density of its matrices [8,5].

Komarek and Moore [9] are the first to show that the *truncated-regularized iteratively re-weighted least squares* (TR-IRLS) algorithm can be effectively implemented on LR to classify large and high dimensional data sets, and that it can outperform the support vector machine (SVM) algorithm. The TR-IRLS algorithm is based on the linear CG method, as described by Komarek [9]. Maalouf and Siddiqi [10] apply the LR truncated Newton method to large-scale imbalanced and rare events data using the *rare events weighted logistic regression* (RE-WLR) algorithm. Maalouf et al. [11] show the effectiveness of the linear CG in solving the *kernel logistic regression* (KLR) model through the *truncated regularized kernel logistic regression* (TR-KLR) algorithm. Furthermore, Maalouf and Trafalis [12] extended the TR-KLR model to imbalanced data through the *rare-event weighted kernel logistic regression* (RE-WKLR) algorithm. To the authors' knowledge, truncated Newton methods have not been fully utilized to solve KRR problems. A possible reason could be the notion that the stability of the CG method is not guaranteed when the data matrix is dense [8,13].

Our motivation for this study is based on the success and effectiveness of truncated Newton methods when applied to KLR

* Corresponding author.
E-mail addresses: maher.maalouf@kustar.ac.ae (M. Maalouf), dirar.homouz@kustar.ac.ae (D. Homouz).

http://dx.doi.org/10.1016/j.knosys.2014.08.012
0950-7051/

classification problems [11,12]. In this study we combine the speed of the truncated Newton techniques with the accuracy generated by the use of kernels for solving nonlinear KRR problems. As with our TR-KLR classification method, our proposed regression method, the TR-KRR algorithm, is easy to implement and requires solving only an unconstrained regularized optimization problem, thus providing a computationally more efficient alternative algorithm to SVM. The combination of regularization, approximate numerical methods, kernelization and efficient implementation are essential to enabling TR-KRR to be at once an effective and powerful regression method. We test the performance of TR-KRR on six data sets, one of which is simulated and the rest are real-life data sets. In as much as the use of truncated Newton methods has not been fully exploited in solving KRR models, it is our intention to provide further contribution.

In Section 2, we provide a brief description of the LS method. In Section 3, we derive the RR model. Sections 4 and 5 discuss the KRR model and the TR-KRR algorithm, respectively. Numerical results are presented in Section 6 and Section 7 states the conclusion.

2. Least squares method

Let $\mathbf{X}$ in $\mathbb{R}^{N\times d}$ be a data matrix where N is the number of training instances (examples) and d is the number of features (parameters or attributes), and $\mathbf{y}$ be a real-valued outcome vector. Let the set of training data be $\{(\mathbf{x}_1,y_1),\ldots,(\mathbf{x}_N,y_N)\}$, where each $\mathbf{x}_i$ in $\mathbb{R}^d$ (a row vector in $\mathbf{X}$) denotes a sample (instance) in the input space with a corresponding output y_i in $\mathbb{R}$, for $i=1,2,\ldots N$. The goal is to find a functional approximation, $f(\mathbf{x})$, for inputs outside of the training sample but hypothetically follow the same probability distribution function as the sample points. Mathematically,

$$f(\mathbf{x})=\langle\boldsymbol{\beta},\mathbf{x}\rangle+\beta_0 \quad (\boldsymbol{\beta},\mathbf{x}\in\mathbb{R}^d), \tag{1}$$

where $\boldsymbol{\beta}$ is the weight vector of the regression hyperplane and β_0 is the threshold with respect to the origin. The method of *least squares* (LS) is a well-known method of estimation [14,15]. The general linear model in matrix form is

$$\mathbf{y}=\mathbf{X}\boldsymbol{\beta}+\boldsymbol{\epsilon}, \tag{2}$$

where $\boldsymbol{\epsilon}=(\epsilon_1,\epsilon_2,\ldots,\epsilon_N)^{\mathrm{T}}$ in $\mathbb{R}$ is the error vector, with the assumptions that the errors have a constant variance, and are linearly independent and normally distributed. The vector $\boldsymbol{\beta}$ is the vector of unknown parameters such that $\mathbf{x}_i\leftarrow[1,\mathbf{x}_i]$ and $\boldsymbol{\beta}\leftarrow[\beta_0,\boldsymbol{\beta}^{\mathrm{T}}]$. From now on, the assumption is that the intercept is included in the vector $\boldsymbol{\beta}$. The LS method then estimates $\boldsymbol{\beta}$ by minimizing the sum of squared residuals (RSS),

$$RSS=\sum_{i=1}^{\ell}\epsilon_i^2=\epsilon^{\mathrm{T}}\epsilon=(\mathbf{y}-\mathbf{X}\boldsymbol{\beta})^{\mathrm{T}}(\mathbf{y}-\mathbf{X}\boldsymbol{\beta}). \tag{3}$$

This sum is often called the objective function. The same objective function can be also obtained by taking the natural log of probability distribution function of errors, in close similarity to the maximum likelihood approach. The solution, after obtaining the gradient vector and the Hessian matrix, and given the matrix $(\mathbf{X}^{\mathrm{T}}\mathbf{X})$ is non-singular, is

$$\hat{\boldsymbol{\beta}}=(\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{y}. \tag{4}$$

3. Ridge regression

3.1. Ridge regression in the primal

One of the drawbacks of the method of least-squares is poor estimation of the regression coefficients, which could make the absolute values of the least-squares estimates too large and unstable [16]. Ridge regression "shrinks" the least-squares coefficients through the addition of a regularization parameter, thus minimizing the following objective function [17]:

$$f(\beta)=\frac{1}{2}(\mathbf{y}-\mathbf{X}\boldsymbol{\beta})^{\mathrm{T}}(\mathbf{y}-\mathbf{X}\boldsymbol{\beta})+\frac{\lambda}{2}\beta^T\beta, \tag{5}$$

where $\lambda\geqslant 0$ is the regularization parameter, and it is usually user-defined. The parameter λ is important in determining the bias-variance trade-off of an estimator [18,19]. When λ is very small, there is less bias but more variance. Larger values of λ, however, lead to more bias but less variance [20]. Therefore, the inclusion of regularization in the ridge regression model is very important to reduce any potential inefficiency. Furthermore, the addition of the regularization parameter makes the problem nonsingular, even if $\mathbf{X}^{\mathrm{T}}\mathbf{X}$ is not in full rank [17].

The gradient is obtained by differentiating the objective function in (5) with respect to $\boldsymbol{\beta}$, which vanishes by applying the first order condition $\nabla f(\beta)=\mathbf{0}$, yielding

$$\beta=(\mathbf{X}^{\mathrm{T}}\mathbf{X}+\lambda\mathbf{I}_d)^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{y}, \tag{6}$$

where $\mathbf{I}_d$ is a $d\times d$ identity matrix.

3.2. Ridge regression in the dual

Let α be a dual variable such that β can be a linear combination of the data points, then

$$\beta=\mathbf{X}^{\mathrm{T}}\alpha, \tag{7}$$

making the general linear model be

$$\mathbf{y}=\mathbf{X}\mathbf{X}^{\mathrm{T}}\alpha+\boldsymbol{\epsilon}=\mathbf{G}\alpha+\boldsymbol{\epsilon}, \tag{8}$$

where $\mathbf{G}=\mathbf{X}\mathbf{X}^{\mathrm{T}}$ is a symmetric Grammian matrix. The objective function in (5) can be minimized with respect to α by invoking (7), such that

$$f(\alpha)=\frac{1}{2}(\mathbf{y}-\mathbf{G}\alpha)^{\mathrm{T}}(\mathbf{y}-\mathbf{G}\alpha)+\frac{\lambda}{2}\alpha^{\mathrm{T}}\mathbf{G}\alpha. \tag{9}$$

Applying the first order condition $\nabla f(\alpha)=\mathbf{0}$ gives the dual solution

$$\boldsymbol{\alpha}=(\mathbf{G}+\lambda\mathbf{I}_N)^{-1}\mathbf{y}, \tag{10}$$

where $\mathbf{I}_N$ is now an $N\times N$ identity matrix.

4. Kernel Ridge Regression (KRR)

The linear transformation in (7) can be replaced with a more general non-linear mapping function, $\phi(\cdot)$, which maps the data from a lower dimensional space into a higher one, such that

$$\phi:\mathbf{x}\in\mathbb{R}^d\rightarrow\phi(\mathbf{x})\in\mathbb{F}\subseteq\mathbb{R}^{\Lambda}. \tag{11}$$

The goal for choosing the mapping ϕ is to convert nonlinear relations between the response variable and the independent variables into linear relations. Usually, the transformations $\phi(.)$ are often unknown. However, the solution to the regression problem depends only on the dot product in the feature space, as in the formulation of the dual problem in (8). The dot product can be expressed in terms of the input vectors through the kernel function.

Now, to avoid the curse of dimensionality of the nonlinear transformation, a kernel function in the form of the dot product, $\mathbf{K}=K(\mathbf{x}_i,\mathbf{x}_j)=\langle\phi(\mathbf{x}_i),\phi(\mathbf{x}_j)\rangle$, which measures the similarity between two vectors is introduced. This kernel is a transformation function that must satisfy Mercer's condition [21]. The importance of the kernel lies in identifying nonlinear functional relations between one selected variable and the remaining features [22]. The kernel function maps the input vectors to a higher dimensional