penalizedSVM: a R-package for feature selection SVM classification

Natalia Becker1,*, Wiebke Werft2, Grischa Toedt1, Peter Lichter1 and Axel Benner2

1Division Molecular Genetics and 2Division Biostatistics, INF 280, 69120 Heidelberg, Germany.

Received on February 20, 2009; revised on April 6, 2009; accepted on April 22, 2009

Advance Access publication April 27, 2009

Associate Editor: Jonathan Wren

To whom correspondence should be addressed.

© The Author 2009. Published by Oxford University Press. All rights reserved. For Permissions, please email: journals.permissions@oxfordjournals.org

Summary: Support vector machine (SVMs) classification is a widely used and one of the most powerful classification techniques. However, a major limitation is that SVM cannot perform automatic feature selection. To overcome this limitation, a number of penalized feature selection methods have been proposed. In the R package ‘penalizedSVM’ implemented penalization functions are used and one of the most powerful classification techniques, especially for high-dimensional data.

The command help (package=penalizedSVM) points to the available help pages.

Often we do not only require low prediction error but also we need to identify covariates playing an important role in discrimination between the classes and to assess their contribution to the classifier. To do so, a number of feature selection methods have been proposed for SVMs.

Feature selection methods can be subdivided into two classes: filter and wrapper methods (Blum and Langley, 1997; Weston et al., 2000). Filter methods, e.g. Recursive Feature Elimination (RFE) (Guyon et al., 2002) drop irrelevant features before the learning algorithm constructs the prediction rule. Wrapper methods provide the selection within the optimization procedure, which increases the prediction power (Markowetz and Spang, 2005).

The R package ‘penalizedSVM’ provides two wrapper feature selection methods for SVM classification using penalty functions. We implemented a new quick version of L1 penalty (LASSO). The second implemented method, Smoothly Clipped Absolute Deviation (SCAD) was up to now not available in R. Thus, this package makes feature selection SVM methods open to the wide R community.

The command help (package=penalizedSVM) points to the available help pages.

1 INTRODUCTION

Classification methods play an important role in data analysis in a wide range of scientific applications. Classification and prediction by support vector machines (SVM) is a widely used and one of the most powerful supervised classification techniques, especially for high-dimensional data.

Given a training dataset \( \{(x_1, y_1), \ldots, (x_n, y_n)\} \), \( x_i \in \mathbb{R}^d \), \( y_i \in \{-1, 1\} \) with \( x = (x_1, \ldots, x_d) \) —input data and \( y \) —class label. The linear SVM separate classes by a linear boundary

\[
\begin{align*}
    f(x) &= \sum_{j=1}^{d} w_j x_j + b, \\
    \text{where} \quad w = (w_1, \ldots, w_d) \text{ are the coefficients of the hyperplane and } b \text{ denotes the intercept of the hyperplane. The class assignment for a test dataset } x_{\text{test}} \text{ is given as } y_{\text{test}} = \text{sign}(f(x_{\text{test}})).
\end{align*}
\]

The problem of finding the optimal hyperplane with maximal margin is solved by convex optimization. Maximizing the margin can be achieved by solving

\[
\min_{w, b} \sum_{i=1}^{n} \left[ 1 - y_i f(x_i) \right] + \text{pen}_\lambda(w).
\]

The penalty term \( \text{pen}_\lambda(w) = \lambda \sum_{j=1}^{d} |w_j|^\gamma \) for SVM has the form of \( L_2 \) norm (‘ridge penalty’), which only shrinks the coefficients but does not set them exact to zero.

Often we do not only require low prediction error but also we need to identify covariates playing an important role in discrimination between the classes and to assess their contribution to the classifier. To do so, a number of feature selection methods have been proposed for SVMs.

Feature selection methods can be subdivided into two classes: filter and wrapper methods (Blum and Langley, 1997; Weston et al., 2000). Filter methods, e.g. Recursive Feature Elimination (RFE) (Guyon et al., 2002) drop irrelevant features before the learning algorithm constructs the prediction rule. Wrapper methods provide the selection within the optimization procedure, which increases the prediction power (Markowetz and Spang, 2005).

The R package ‘penalizedSVM’ provides two wrapper feature selection methods for SVM classification using penalty functions. We implemented a new quick version of \( L_1 \) penalty (LASSO). The second implemented method, Smoothly Clipped Absolute Deviation (SCAD) was up to now not available in R. Thus, this package makes feature selection SVM methods open to the wide R community.

The command help (package=penalizedSVM) points to the available help pages.

2 METHODS

LASSO (\( L_1 \)): the use of a \( L_1 \) penalty function is a common approach in regression originally proposed by Tibshirani (1996). Tibshirani described a technique, called the LASSO for ‘least absolute shrinkage and selection operator’, for parameter estimation. Bradley and Mangasarian (1998) adapted the \( L_1 \)-regularization to SVM. The penalization term has the form

\[
\text{pen}_\lambda(w) := \lambda \sum_{j=1}^{d} |w_j|.
\]

As a result of singularity of the \( L_1 \) penalty function, the \( L_1 \) SVM can automatically select genes by shrinking the small coefficients of the hyperplane to exactly zero. Thus, the \( L_1 \) SVM is an effective feature selection tool. Recently, Fung and Mangasarian (2004) have published a fast \( L_1 \) SVM modification using a Newton Linear Programming SVM (NLPSVM). We implemented this modification in the package.

SCAD: it is a non-convex penalty function first proposed by Fan (1997) and discussed in Fan and Li (2001). Zhang et al. (2006) combined SVM with SCAD for feature selection. The penalization term for SCAD SVM has the form

\[
\text{pen}_\lambda(w) := \sum_{j=1}^{d} \text{pen}_{\lambda j}(w_j).
\]
To illustrate the use of ‘penalized SVM’, an example session with fit.scad

\[
\text{> ns.g = 10, seed=124)}
\]

\[
\text{> test}
\]

number of features.

\[
\text{at all. To validate the prediction, we used test data with the same}
\]

predictor could also be the null model, i.e. containing no feature

\[
\text{was described by only a subset of features. Note that the final}
\]

cross-validation. After training the model, the separating hyperplane

\[
\text{validation. Optionally, this can be combined with an outer}
\]

and eventually the use of outer cross-validation, the calculation time

\[
\text{can increase from seconds to several minutes. In the example above,}
\]

the computation time was 80 s using a desktop computer with 2 GB

RAM and 3 GHz CPU speed.

3 AN EXAMPLE SESSION

To illustrate the use of ‘penalized SVM’, an example session with simulated data is shown. First, we simulated training data with \( n = 200 \) samples, each with \( ng = 1000 \) features, where only \( nsg = 10 \) out

\[
\text{of 1000 features have an impact on the class labels.}
\]

\[
\text{> train<-sim.data(n = 200, ng = 1000, nsg = 10, seed=123)}
\]

Since SVM algorithms are very sensitive to the choice of the tuning parameters, the user should predefine a possible set of tuning parameters, \( \Lambda = \{\lambda_k, k = 1, \ldots, K\} \).

\[
\text{After defining a set of candidate tuning parameters}
\]

\[
\text{> Lambda.scad <- seq(0.01, 0.05, 0.01)}
\]

we trained the classifier using the SCAD penalty (for \( L_1 \) see the help page ?svm.fs).

\[
\text{> fit.scad<- svm.fs(x=train$x), y=train$y,}
\]

\[
\text{fs.method="scad", lambda1.set=Lambda.scad)}
\]

The final tuning parameter was selected using generalized cross-validation. Optionally, this can be combined with an outer \( k \)-fold cross-validation. After training the model, the separating hyperplane was described by only a subset of features. Note that the final predictor could also be the null model, i.e. containing no feature at all. To validate the prediction, we used test data with the same number of features.

\[
\text{> test<- simulate.data(n = 200, ng = 1000,}
\]

\[
\text{nsg = 10, seed=124)}
\]

The predicted classes for the test data result from

\[
\text{> predict(fit.scad,new.data=test$x),}
\]

\[
\text{new.data.labels=test$y)}
\]

The function predict(fit) predicts class labels for a new set of samples. In case the true class labels for test samples are available, the prediction accuracy, misclassification table, sensitivity and specificity are also calculated. In this example, the misclassification error was 21% with sensitivity 74% and specificity 83%.

Depending on sample size, number of tested tuning parameters and eventually the use of outer cross-validation, the calculation time can increase from seconds to several minutes. In the example above, the computation time was 80 s using a desktop computer with 2 GB RAM and 3 GHz CPU speed.

ACKNOWLEDGEMENTS

The development of the SCAD and its original implementation in MATLAB is the work of H. H. Zhang. The MATLAB implementation of the \( L_1 \) norm SVM was kindly provided by Glenn M. Fung and O. L. Mangasarian.

We are grateful to the editor and the referees for their helpful comments.

Conflict of Interest: none declared.

REFERENCES


