Implementing linear SVM using quadratic programming

Toby Dylan Hocking
toby.hocking@inria.fr
November 23, 2012

The R package \texttt{quadprog} provides the function \texttt{solve.QP(D, d, A, b_0)}, which solves the following optimization problem:

$$
\min_{b \in \mathbb{R}^v} \quad \frac{1}{2} b' Db - d'b \\
\text{subject to} \quad A'b \succeq b_0
$$

where $D \in \mathbb{R}^{v \times v}$, $d \in \mathbb{R}^v$, $A \in \mathbb{R}^{v \times k}$, $b_0 \in \mathbb{R}^k$, $v$ is the number of optimization variables, $k$ is the number of inequality constraints, and $x \succeq y$ is componentwise inequality, which implies $x_i \geq y_i$ for all $i$. We call this the “standard form” of a quadratic program. \texttt{solve.QP} is a general-purpose quadratic programming solver that can be used for many things, but here we will use it to solve several formulations of linear Support Vector Machines (SVM).

1 Download and install R and \texttt{quadprog}

If you use Linux install R using your package manager if possible. On other systems you can download R from \url{http://cran.univ-lyon1.fr/}

Once R is installed, the \texttt{quadprog} package can be downloaded and installed using the following command line in R:

$$
> \text{install.packages("quadprog")}
$$

Then you can use the \texttt{library(quadprog)} command to get access to the \texttt{solve.QP} function.

2 Writing linear SVM in standard form

For a set of training points $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ with $x_i \in \mathbb{R}^p$ and $y_i \in \{-1, 1\}$, we stack the $x_i$ and $y_i$ to form the matrix $X \in \mathbb{R}^{n \times p}$ and the class vector $y \in \{-1, 1\}^n$. The linear $C$-SVM classifier for a new point $x \in \mathbb{R}^p$ is given by $f(x) = \beta_0 + \beta' x$, where $\beta_0 \in \mathbb{R}$ and
\( \beta \in \mathbb{R}^p \) are found as the solution to the following optimization problem:

\[
\begin{align*}
\min_{\beta_0 \in \mathbb{R}, \beta \in \mathbb{R}^p, \xi \in \mathbb{R}^n} & \quad C \sum_{i=1}^{n} \xi_i + \frac{1}{2} \beta' \beta \\
\text{subject to} & \quad \forall i, \xi_i \geq 0 \\
& \quad \forall i, \xi_i \geq 1 - \beta_0 y_i - \beta' x_i y_i 
\end{align*}
\]

(2)

Before using the solver, provide answers to the following questions:

1. Write expressions for the standard form vectors and matrices \( b, D, d, A, b_0 \) in terms of the model parameters \( \beta_0, \beta, \xi, C \) and data \( X, y \). How many optimization variables are there in terms of \( n \) and \( p \)? How many constraints? Hint: start with \( b = [ \beta_0 \ \beta \ \xi ]' \).

2. How would you calculate the model parameters after the solver gives you \( b \)? Discuss the intercept \( \beta_0 \), the normal vector \( \beta \), the margin, and the support vectors. Hint: the support vectors are the training points \( i \) that satisfy \( y_i f(x_i) \leq 1 \).

3. Given a new data point \( x \in \mathbb{R}^p \), how would you predict its class?

Now write a function \texttt{linear.svmqp}(\( X, y, C \)) that implements this optimization problem using \texttt{solve.QP} for a matrix of points \( X \in \mathbb{R}^{n \times p} \), a vector of labels \( y \in \{-1, 1\}^n \), and the margin size parameter \( C \in \mathbb{R} \). The function should return a list of estimated coefficients \( \beta_0, \beta \), size of the margin, and matrix of support vectors.

3 Kernel SVM primal problem

SVM can be extended by introducing a kernel function \( \kappa : \mathbb{R}^p \times \mathbb{R}^p \to \mathbb{R} \) and then calculating the kernel matrix \( K \in \mathbb{R}^{n \times n} \) where \( K_{ij} = \kappa(x_i, x_j) \). Then we replace all the inner products with kernel evaluations, giving the prediction function \( f(x) = \beta_0 + \sum_{i=1}^{n} \alpha_i \kappa(x, x_i) \) and the following optimization problem:

\[
\begin{align*}
\min_{\beta_0 \in \mathbb{R}, \alpha \in \mathbb{R}^n, \xi \in \mathbb{R}^n} & \quad C \sum_{i=1}^{n} \xi_i + \frac{1}{2} \alpha' K \alpha \\
\text{subject to} & \quad \forall i, \xi_i \geq 0 \\
& \quad \forall i, \xi_i \geq 1 - y_i \beta_0 - y_i \alpha' K_i 
\end{align*}
\]

(3)

where \( K_i = [ \kappa(x_1, x_i) \ldots \kappa(x_n, x_i) ]' \) is the \( i \)-th column of \( K \).

For linear SVM, we have \( \kappa(x_i, x_j) = x_i' x_j \), so \( K = XX' \). Answer the same questions as above for this optimization problem. Write a function \texttt{linear.kernel.svm.primal.qp}(\( X, y, C \)), as above, that implements it.
4 Kernel SVM dual problem

The optimization problem can be simplified by taking the dual:

$$\min_{\alpha \in \mathbb{R}^n} \frac{1}{2} \alpha' K \alpha - \alpha' y$$

subject to  \( \sum_{i=1}^{n} \alpha_i = 0 \) and  \( \forall i, \ 0 \leq y_i \alpha_i \leq C \)  \hspace{1cm} (4)

Answer the same questions as above for this problem, then write a function `linear.kernel.svm.dual.qp(X, y, C)` which implements it.

5 Comparison

Use simulated data in 2D or `data(ALL, package="ALL")` to compare the results of each function above to the results obtained from the `ksvm()` function from `library(kernlab)`. Refer to the TP to see how to simulate data, install the ALL package, and use the `ksvm()` function: [http://cbio.ensmp.fr/~jvert/teaching/2011mines/index.html](http://cbio.ensmp.fr/~jvert/teaching/2011mines/index.html)

1. For \( X, y, C \) fixed, do all the functions yield the same support vectors, margin, and parameters \( \beta_0, \beta, \alpha \)? Do the estimated functions \( f(x) \) agree? If not, can you propose changes to the algorithms to make the parameterizations agree?

2. How many parameters are involved in each optimization problem, in terms of \( n \) and \( p \)? Under what circumstances is it advantageous to use each algorithm?

3. How long does each algorithm take? Hint: you can use the `system.time()` function to get the time it takes to execute each algorithm.