Constrained Optimization Methods for Support Vector Machine Classification

Marylesa Howard · Johnathan M. Bardsley

Received: date / Accepted: date

Abstract The support vector machine (SVM) is a supervised classification method that separates a data set into classes using nonlinear boundaries. Construction of the SVM classifier yields a quadratic program (QP) with bound and equality constraints. We present two augmented Lagrangian (AL) optimization methods for numerically solving the SVM QP. One AL method incorporates both the bound and equality constraints into Lagrangian, allowing for the use of an unconstrained quadratic optimizer. The other AL method incorporates only the equality constraint into the Lagrangian and hence requires the solution of a bound constrained QP. We also present an adaptation of the scaled gradient projection (SGP) method of [3] for solving the SVM QP. Numerical examples on image data sets indicate that both AL methods and SGP work well, and that the AL methods outperform SGP and two other state-of-the-art methods that we include for comparisons.

Keywords support vector machines · constrained optimization · image classification

1 Introduction

Supervised image classification seeks to segment a given image into classes based on a priori knowledge of the properties of the pixels in each class. Classification techniques are used on a wide range of scientific problems. For example, ecologists classify aerial images in order to determine how landscapes
change over time [28], while the U.S. Post Office uses classification techniques for handwriting recognition [18]. Other non-imaging examples include spam detection systems for email and patient diagnosis in medicine [18].

Supervised classification methods require the user to provide the set of classes in the data set, as well as a training field for each class, which comprises the training data. For imaging problems, the training data are sets of pixels that correspond to each class in the image. A classifier is built from the training data, which is then applied to the test set. The test set is the set of remaining pixels in the image for which the class is unknown.

The support vector machine (SVM) is a well-known method for supervised classification and is well documented throughout the literature; see, e.g., [32, 33, 5, 9, 18, 13]. It requires the solution of a constrained quadratic minimization problem. For the SVM classifier with bound and linear equality constraints, we introduce an augmented Lagrangian technique similar to that of [22, 24], but that incorporates all constraints into the objective function allowing for the use of an unconstrained quadratic minimization algorithm such as conjugate gradient. Furthermore, we adapt the scaled gradient projection algorithm of [3] for use on the constrained SVM quadratic program. We emphasize that our focus is on the development of algorithms for solving the SVM quadratic program, and hence that is where the novelty in the work lies.

1.1 Support Vector Classifiers

We now give a brief introduction to support vector classifiers. We assume that each pixel in the image of interest has \( n \) bands, and hence the feature space is \( \mathbb{R}^n \). The idea behind the support vector classifier (SVC) is to seek a separating hyperplane of the training data in feature space and then apply that hyperplane to the test set, giving a classification of the image.

For simplicity and visualization purposes, we begin with a two class, \( n \)-banded image, following the work of [9, 18, 5]. Let the training data be the set \( \{ x_i, y_i \}, \ i = 1, \ldots, p \), where \( x_i \in \mathbb{R}^n \) is the data of a single \( n \)-banded pixel and \( y_i \in \{-1, +1\} \) represents its inclusion into one of the two classes. Let a linear classifier for this training data be defined by a vector \( w \in \mathbb{R}^n \), a scalar \( b \), and the decision rule

\[
G(\mathbf{x}) = \text{sign}(w' \mathbf{x} + b),
\]

which places \( \mathbf{x} \) into the -1 class if \( G(\mathbf{x}) \) is negative and into the +1 class if it is positive.

Assuming completely separable data, there exist \( w \) and \( b \) such that the hyperplane \( w' \mathbf{x} + b = 0 \) separates the training data. Figure 1 shows an example of the feature space for a two class, two-banded image with a separating hyperplane. Let \( d^+ \) (\( d^- \)) be the shortest distance from the separating hyperplane to the closest positive (negative) \( y_i \). The margin is defined as \( d^+ + d^- = 2/||w|| \).

Maximizing the margin is an intuitive approach that simultaneously distances all of the training data as far from the hyperplane as possible, thereby
Fig. 1 An example of a two class, two-banded image with a separating hyperplane in its feature space. The solid line is the separating hyperplane. The dotted line is the margin and training data that fall on the margin are called support vectors minimizing the likelihood of a misclassification. See Figure 2 for a visual demonstration of the optimality of the maximum margin hyperplane in terms of misclassification error.

Fig. 2 An example of a two-banded, two class data set with and without maximizing the margin of a separating hyperplane. Training data appear as stars, test data appear as closed and open circles for the two classes. Left: The maximized margin separating hyperplane from Figure 1 applied to the test data. Note that one data point is misclassified. Right: A separating hyperplane in which the margin is not maximized. Note that four data points are misclassified.

This paper is organized as follows for the convenience of the reader. We begin with the derivation of a bound and equality constrained SVC formulation in Section 2 for the convenience of the reader, including a brief description creating the SVM with kernels. Optimization techniques are discussed in Section 3, including the unconstrained augmented Lagrangian method developed by us. Numerical examples are then presented in Section 4, followed by conclusions in Section 5.
2 Support Vector Machines

Many variations on the SVC formulation exist [13,9], and each formulation has its own merits. Here we focus on a formulation that results in a quadratic program with bound and equality constraints. Following the derivation, we discuss the enlargement of the feature space to allow for the use of nonlinear boundaries in the classification of support vector machines.

2.1 Derivation of an Bound-plus-Equality Constrained SVC

We assume a non-separable data set such that a hyperplane cannot partition the training data in feature space into its classes without misclassification. For non-separable data, the SVC seeks the hyperplane of maximal margin but introduces nonnegative slack variables $\xi_i$ into the constraints:

$$\begin{align*}
  w'x_i + b & \geq +1 - \xi_i \quad \text{for } y_i = +1, \\
  w'x_i + b & \leq -1 + \xi_i \quad \text{for } y_i = -1, \\
  \xi_i & \geq 0, \\
  \sum_{i=1}^{p} \xi_i & \leq \text{constant},
\end{align*}$$

for $i = 1, \ldots, p$, where the last constraint places a maximum on the magnitude of the training errors [18]. Maximizing the margin with these constraints corresponds to the optimization problem

$$\min_{w, b, \xi_i} \frac{1}{2}||w||^2 \quad \text{subject to } \begin{cases}
  y_i(w'x_i + b) - (1 - \xi_i) & \geq 0 \quad \forall \, i, \\
  \xi_i & \geq 0 \quad \forall \, i, \\
  \sum_{i=1}^{p} \xi_i & \leq \text{constant},
\end{cases} \quad (2)$$

We introduce Lagrange multipliers $\alpha_i$ and $\mu_i$ so that the primal problem (2) can be rewritten as

$$\min_{w, b, \xi_i} L_p = \frac{1}{2}||w||^2 + \sum_{i=1}^{p} (C - \mu_i)\xi_i - \sum_{i=1}^{p} \alpha_i(y_i(w'x_i + b) - (1 - \xi_i)) \quad \text{subject to } \alpha_i, \mu_i, \xi_i \geq 0, \quad (3)$$

where the cost parameter $C$ replaces the constant in (2) [18].

By calculating the Karush-Kuhn-Tucker (KKT) conditions for (3), we can convert the primal program into a dual quadratic program [26]. The Wolfe-dual [15] is derived by maximizing the primal objective function $L_p$ with respect to the Lagrange multipliers, subject to $\nabla_{w, b, \xi_i} L_p = 0$, and $\alpha_i, \mu_i \geq 0$. Solving the stationarity KKT condition, $\nabla_{w, b, \xi} L_p = 0$, yields

$$w = \sum_{i=1}^{p} \alpha_i y_i x_i, \quad \sum_{i=1}^{p} \alpha_i y_i = 0, \quad \text{and } \alpha_i = C - \mu_i, \quad \forall \, i. \quad (4)$$
Substituting (4) into (3) we obtain the quadratic dual objective function

\[ L_d = \sum_{i=1}^{p} \alpha_i - \frac{1}{2} \sum_{i=1}^{p} \sum_{j=1}^{p} \alpha_i \alpha_j y_i y_j x'_i x_j. \]

Notice that the Lagrange multipliers \( \mu_i \), slack variables \( \xi_i \), hyperplane vector \( w \), and intercept \( b \) do not appear in this final formulation. The dual of (3) is written as

\[
\min_{\alpha_i} \frac{1}{2} \sum_{i=1}^{p} \sum_{j=1}^{p} \alpha_i \alpha_j y_i y_j x'_i x_j - \sum_{i=1}^{p} \alpha_i \text{ subject to } \left\{ \begin{array}{l}
0 \leq \alpha_i \leq C \quad \forall i \\
\sum_{i=1}^{p} \alpha_i y_i = 0
\end{array} \right\}. \tag{5}
\]

The constraint \( 0 \leq \alpha_i \leq C \) comes from the fact that \( \alpha_i = C - \mu_i \) and \( \mu_i \geq 0 \). We will focus in this manuscript on solving the dual problem (5); however, it is also possible to solve the primal problem [7,19,16,30,4].

The dual optimization problem (5) is equivalent to solving (2), yet it remains to find \( w \) and \( b \) for the hyperplane. Given the solution \( \alpha^* \) of (5), by (4) we have

\[ w^* = \sum_{i \in S} \alpha^*_i y_i x_i, \tag{6} \]

where \( S = \{i|\alpha_i \neq 0\} \), while from the KKT complimentary slackness conditions \( \alpha_i (y_i (w^* x_i + b) - (1 - \xi_i)) = 0 \), we choose \( b^* \) to be the average of those for which \( \alpha_i \neq 0 \),

\[ b^* = \frac{1}{|S|} \sum_{i \in S} (1 - y_i (w^*)' x_i) / y_i. \tag{7} \]

Nonzero \( \alpha_i \) corresponds to support vector \( x_i \), which lies on the margin, thus indicating \( \xi_i = 0 \). Hence \( b^* \) is an average of all the intercepts suggested by the support vectors.

Using the estimated hyperplane, \( (w^*)' x + b^* \), the decision rule for classification is given by (1). Generalizing to more classes is a simple extension of the two class problem. Classification with \( m \) classes requires an associated hyperplane \( (w^*_i, b^*_i) \) for each class, \( i = 1, \ldots, m \). The decision rule is given by

\[ G(\mathbf{x}) = \arg \max_{1 \leq i \leq m} ((w^*_i)' \mathbf{x} + b^*_i). \]

Geometrically, this assigns new data \( \mathbf{x} \) to the class for which \( \mathbf{x} \) is furthest from the associated hyperplane.

### 2.2 Expanding the Feature Space with Kernels

For the SVC, the decision function is a linear function of the data, creating linear decision boundaries in the input feature space. This is not always optimal. Nonlinear boundaries arise when the Euclidean inner-product \( x'_i x_j \), appearing in the above dual objective function, is replaced by the evaluation of a kernel function, \( K(x_i, x_j) \), yielding the support vector machine (SVM). The
advantage of using kernels is that nonlinear decision boundaries are computed cheaply [9].

The underlying idea is illustrated as follows. First, we transform the training data using basis functions: \( h(x_i) = (h_1(x_i), \ldots, h_l(x_i)) \) for \( i = 1, \ldots, p \). We then apply the SVC to the transformed data, yielding the dual function for (5) as

\[
L_d = \sum_{i=1}^{p} \alpha_i - \frac{1}{2} \sum_{i=1}^{p} \sum_{j=1}^{p} \alpha_i \alpha_j y_i y_j \langle h(x_i), h(x_j) \rangle,
\]

with hyperplane solution

\[
(w^*)' h(x) + b^* = \sum_{i=1}^{p} \alpha_i y_i \langle h(x_i), h(x) \rangle + b^*.
\]

Notice that in all of the above, the basis functions appear only within an inner product, and hence it suffices to know the kernel function defined by

\[
K(x, y) = \langle h(x), h(y) \rangle,
\]

which computes inner products in the transformed space. The following three kernels were the first ones used in pattern recognition and are still popular:

- \( d^{th} \) Degree polynomial: \( K(x, y) = (1 + \langle x, y \rangle)^d \),
- Radial basis: \( K(x, y) = e^{-\sigma ||x-y||^2} \),
- Neural Network: \( K(x, y) = \tanh(\kappa_1 \langle x, y \rangle + \kappa_2) \).

Figure 3 visually demonstrates the improvement in classification error for non-separable data using the radial basis kernel versus no kernel.

In enlarged feature space, the cost parameter \( C \) (the upper-bound on the inequality constraint in (5)), is more understandable in the role it plays. With complete separation often plausible in the enlarged feature space, a larger value of \( C \) corresponds to greater number of support vectors (fewer positive slack variables \( \xi_i \)) and thus a wiggly, possibly overfit boundary in the original feature space [18]. A smaller value of \( C \) corresponds to fewer support vectors and a smoother boundary, allowing more positive values of \( \xi_i \).

3 Bound and Linear Equality Constrained Quadratic Programming Methods

We have presented an SVM formulation in Section 2 and now rewrite it in matrix notation for implementation in optimization algorithms. The bound and equality constrained quadratic program (5) can be written in the form

\[
\min_{\alpha} f(\alpha) = \frac{1}{2} \alpha' A \alpha - \alpha' b
\]

s.t. \( l \leq \alpha \leq u \) and \( y' \alpha = 0 \),

\[
(10)
\]
Fig. 3 Training data for three classes are indicated by the large circles, triangles, and squares. Classification for $\mathbb{R}^2$ is given, where the smaller dots correspond to the large circle class, the smaller triangles correspond to the large triangle class, and the empty space corresponds to the square class. The left image gives classification for the training set using linear boundaries (no kernel) while the right image indicates classification using a radial basis kernel with $\sigma = 0.2$. Both classifications use $C = 10^5$. Notice the linear decision boundaries misclassify four of the training data points due to non-separability while the nonlinear decision boundaries correctly classify the training data.

where $\mathbf{\alpha} = (\alpha_1, \ldots, \alpha_p)'$, $\mathbf{b} = (1, \ldots, 1)'$, $\mathbf{l} = (0, \ldots, 0)'$, $\mathbf{u} = (C, \ldots, C)'$, $\mathbf{y} = (y_1, \ldots, y_p)'$, and

$$\mathbf{A} = \begin{bmatrix} y_1 \mathbf{x}_1' \\ \vdots \\ y_p \mathbf{x}_p' \end{bmatrix} \begin{bmatrix} y_1 \mathbf{x}_1 & \ldots & y_p \mathbf{x}_p \end{bmatrix}.$$ 

While there are many approaches for solving the SVM problem as mentioned above, here we focus solely on the dual formulation provided in Section 2. Methods such as cutting planes and the stochastic sub-gradient take advantage of the primal formulation [7,19,16,4,30] but are not discussed here. Interior point methods which exploit the low rank structure of the problem are another common approach but are not discussed here [14,12]. While we incorporate the use of projected Newton and conjugate gradient algorithms into the augmented Lagrangian algorithms, various other active set methods are available [23,31,29] for solving the dual formulation.

3.1 The Augmented Lagrangian

The augmented Lagrangian (AL) method is also known as the method of multipliers and solves the constrained optimization problem

$$\min_{\mathbf{\alpha}} f(\mathbf{\alpha}) \quad \text{subject to} \quad \begin{cases} h_i(\mathbf{\alpha}) = 0 \quad \forall \ i \in \mathcal{E} \\ g_i(\mathbf{\alpha}) \leq 0 \quad \forall \ i \in \mathcal{I} \end{cases},$$

where $\mathcal{E}$ and $\mathcal{I}$ are the sets of equality and inequality constraints, respectively. The general idea behind the AL method is to incorporate some or all of the
constraints into the objective function \( f(\alpha) \) \[2,25\]. We present AL formulations for the SVM optimization problem (10) here.

In \[22,24\], the equality constraint is incorporated into the Lagrangian function, leaving only the bound constraints. In \[24\], gradient projection conjugate gradient (GPCG) \[23\] is used to solve the resulting bound constrained quadratic program, whereas in \[22\] an approximate problem is solved analytically. Here we follow the approach of \[24\] but use projected Newton \[20\] in place of GPCG as it reduces computational time and is easier to implement. We also take the AL algorithm one step further and incorporate both the equality and bound constraints into the Lagrangian, allowing us to construct an iterative algorithm requiring the solution of a sequence of unconstrained quadratic programs, which can be solved with less sophisticated methods.

Before continuing, we note that the LANCELOT algorithm is a well-known method for solving general problems of the form (11) using the AL technique \[8\]. Due to the size of the class of problems in (11), the algorithm is necessarily complex. Since our focus is on the much smaller subclass of problems of the form (10), we are able to present two AL algorithms that are much simpler than LANCELOT.

### 3.1.1 The Bound Constrained Augmented Lagrangian

In the bound constrained AL technique, we add a quadratic penalty to the Lagrangian function. For (10), the AL function is given by

\[
L(\alpha, \lambda, \mu) = f(\alpha) + \lambda y'\alpha + \frac{\mu}{2}(y'\alpha)^2,
\]

where \( \lambda \) is a Lagrange multiplier and \( \mu > 0 \) is a penalty parameter.

Note that as \( \mu \) increases, failure to satisfy the equality constraint \( y'\alpha = 0 \) is increasingly penalized, forcing the optimization towards the feasible region. One might think that adding the quadratic penalty term alone to \( f \) would be enough, however the resulting problem is ill-conditioned as \( \mu \to \infty \) and a systematic perturbation to approximate minimizers results. These issues do not arise if we use (12) \[2,25\].

To create the bound constrained AL algorithm, we fix \( \lambda_k \) and \( \mu_k \) at each iteration and minimize (12) with respect to \( \alpha \) subject to \( l \leq \alpha \leq u \). The KKT conditions state that at the solution \((\alpha^*, \lambda^*, \mu^*)\)

\[
0 = \nabla_{\alpha} L(\alpha^*, \lambda^*, \mu^*) = f'(\alpha^*) + \lambda^*y + \mu^*(y'\alpha^*)
\]

\[= f'(\alpha^*) + y\lambda^*, \]

and for each iteration \( \alpha^k \),

\[
0 \approx \nabla_{\alpha} L(\alpha^k, \lambda_k, \mu_k) = f'(\alpha^k) + y'(\lambda_k + \mu_k(y'\alpha^k)).
\]

Equating (13) and (14) motivates the update

\[
\lambda_{k+1} = \lambda_k + \mu_k y'\alpha^k.
\]
Bertsekas discusses selecting a penalty parameter in [2], noting that $\mu_k$ needs to increase fast enough to have a decent convergence rate but slow enough to reduce ill-conditioning. Various updates for penalty parameter $\mu_k$ are given in [2], one of which we use here: let $\beta > 1$ and $0 < \gamma < 1$, then

$$
\mu_{k+1} = \begin{cases} 
\beta \mu_k & \text{if } ||y'\alpha^k|| > \nu||y'\alpha^{k-1}||, \\
\mu_k & \text{if } ||y'\alpha^k|| \leq \nu||y'\alpha^{k-1}||, 
\end{cases}
$$

(16)

with a recommended choice of $\nu = 0.25$, and we take $\beta = 3$. This increases the penalty parameter by a factor of $\beta$ only if the constraint violation has not decreased by a factor of $\nu$ from the previous minimization.

We can now present the first AL method, which solves (10) by cyclically minimizing $L(\alpha, \lambda_k, \mu_k)$ with respect to $\alpha$ subject to $l \leq \alpha \leq u$ using projected Newton [20], or one of many other bound constrained optimizers [23–25], and by updating $\lambda_k$ and $\mu_k$ via (15) and (16), respectively. In our implementation of projected Newton, we chose a projected gradient norm and step norm stopping tolerance of $10^{-6}$. Here the projected gradient is defined as

$$
[\nabla_p f(\alpha)]_i = \begin{cases} 
[A\alpha - b]_i & \text{if } \alpha_i \in (l_i, u_i), \\
\min\{[A\alpha - b]_i, 0\} & \text{if } \alpha_i = l_i, \\
\max\{[A\alpha - b]_i, 0\} & \text{if } \alpha_i = u_i. 
\end{cases}
$$

(17)

Pseudocode for this approach is presented in Algorithm 1. Convergence of this algorithm is proved in [25].

Algorithm 1. The Bound Constrained Augmented Lagrangian Method

Given $\mu_0 \geq 0$ and initial points $\alpha^0$ and $\lambda_0$, set $k = 0$.

1. Use projected Newton to find the approximate minimizer $\alpha^{k+1}$ of $L(\alpha, \lambda_k, \mu_k)$ subject to $l \leq \alpha \leq u$ with starting point $\alpha^k$ and with a projected gradient norm and step norm stopping tolerance of $10^{-6}$.

2. If $||y'\alpha^k|| < 10^{-6}$, stop with approximate solution $\alpha^{k+1}$.

3. Update the Lagrange multiplier via (15) to obtain $\lambda_{k+1}$, and choose a new penalty parameter $\mu_{k+1}$ via (16).

4. Set $k = k + 1$ and go to Step 1.

3.1.2 The Unconstrained Augmented Lagrangian Method

Bertsekas indicates in [2] that bound constraints may be included into the AL function using slack variables $s_i$ and $t_i$ satisfying $l_i - \alpha_i + s_i^2 = 0$ and $\alpha_i - u_i + t_i^2 = 0$. Then (10) can be alternatively written

$$
\min_{\alpha} \ f(\alpha) = \frac{1}{2}\alpha'A\alpha - \alpha'b \\
\text{subject to } y'\alpha = 0 \\
l_i - \alpha_i + s_i^2 = 0, \quad \text{for } i = 1, \ldots, p, \\
\alpha_i - u_i + t_i^2 = 0, \quad \text{for } i = 1, \ldots, p.
$$

(18)
Next, we incorporate all of the constraints into the Lagrangian to obtain
\[
\mathcal{L}(\alpha, s, t, \lambda, \gamma, \mu) = f(\alpha) + \lambda y'\alpha + \frac{\mu}{2}(y'\alpha)^2 \\
+ \sum_{i=1}^{p} \left[ \gamma_l(i - \alpha_i + s_i^2) + \gamma_u(i - u_i + t_i^2) \right] \\
+ \frac{\mu}{2} \sum_{i=1}^{p} \left[ (i - \alpha_i + s_i^2)^2 + (i - u_i + t_i^2)^2 \right],
\]
with \( s = (s_1, \ldots, s_p)' \), \( t = (t_1, \ldots, t_p)' \). Each iteration of the AL method updates the Lagrange multiplier parameters \( \lambda \) and \( \gamma = (\gamma'_l, \gamma'_u)' \), with \( \gamma'_l = (\gamma_{l_1}, \ldots, \gamma_{l_p})' \), \( \gamma'_u = (\gamma_{u_1}, \ldots, \gamma_{u_p})' \), and quadratic penalty parameter \( \mu \).

Bertsekas [2] notes that minimization of (19) with respect to the slack variables \( s_i, t_i \) yields a closed form expression for these variables. Specifically, the constrained minimizer of \( \mathcal{L} \) with respect to \( s_i \) satisfies
\[
\gamma_l + \mu(s_i + l_i - \alpha_i) = 0,
\]
giving the constrained minimum \( s_i = \max\{0, -(\frac{\gamma_l}{\mu} + l_i - \alpha_i)\} \). We substitute this, as well as the equivalent expression for \( t_i \), into (19) to obtain
\[
\mathcal{L}(\alpha, \lambda, \gamma, \mu) = f(\alpha) + \lambda y'\alpha + \frac{\mu}{2}(y'\alpha)^2 \\
+ \frac{1}{2\mu} \sum_{i=1}^{p} \left[ \max\{0, \gamma_l + \mu(l_i - \alpha_i)\}^2 - \gamma_l^2 \right] \\
+ \frac{1}{2\mu} \sum_{i=1}^{p} \left[ \max\{0, \gamma_u + \mu(\alpha_i - u_i)\}^2 - \gamma_u^2 \right].
\]
Notice the similarity between (20) and (12), yet we now have included both equality and bound constraints in the Lagrangian.

To obtain linear approximations of the max function in (20), we assume \( \alpha = \alpha^k \) and define
\[
[I_{l,k}]_{ii} = \begin{cases} 0 & \text{if } \gamma_l + \mu(l_i - \alpha^k_i) < 0 \\ 1 & \text{otherwise} \end{cases},
\]
\[
[I_{u,k}]_{ii} = \begin{cases} 0 & \text{if } \gamma_u + \mu(\alpha^k_i - u_i) < 0 \\ 1 & \text{otherwise} \end{cases}.
\]
Then (20) can be approximated by the quadratic function
\[
\mathcal{L}_k(\alpha) = \frac{1}{2}\alpha'\hat{A}_k\alpha - \alpha'\hat{b}_k,
\]
where
\[
\hat{A}_k = A + \mu_k(yy' + I_{l,k} + I_{u,k}),
\]
and
\[
\hat{b}_k = b - \lambda_k y + I_{l,k}(\mu_k l + \gamma^k_l) + I_{u,k}(\mu_k u - \gamma^k_u).
\]
This form of the AL function disregards all non-$\alpha$ terms in (20). The benefit of (22) is that an unconstrained quadratic optimizer, such as the conjugate gradient method or even a direct solver such as Gaussian elimination for small-scale problems, can be used to compute its minimizer, yielding the updated approximation $\alpha^{k+1}$.

As with the update rule for $\lambda$ in (15), the update rule for $\gamma$ is derived from the stationarity KKT condition, requiring

$$0 = \nabla_{\alpha} \mathcal{L}(\alpha^*, \lambda^*, \gamma^*, \mu^*) = \nabla f(\alpha^*) + y'(\lambda^* + \mu^* \gamma^*) + \sum_{i=1}^{p} (-\gamma^*_{l_i} + \gamma^*_{u_i})$$

and noting that at each iteration $k$,

$$0 \approx \nabla_{\alpha} \mathcal{L}(\alpha^k, \lambda^k, \gamma^k, \mu_k) = \nabla f(\alpha^k) + y(\lambda^k + \mu_k y' \alpha^k)$$

Equating (23) and (24), we see the update rule for $\lambda$ is still given by (15), while the update for the Lagrangian parameter $\gamma$ is given by

$$\begin{align*}
\gamma_{l_i}^{k+1} &= \max\{0, \gamma_{l_i}^k + \mu_k (l_i - \alpha_i^k)\}, \quad i = 1, \ldots, p, \\
\gamma_{u_i}^{k+1} &= \max\{0, \gamma_{u_i}^k + \mu_k (\alpha_i^k - u_i)\}, \quad i = 1, \ldots, p.
\end{align*}$$

The pseudocode for this AL method is given in Algorithm 2, which closely resembles Algorithm 1, with the added update for the Lagrangian parameter vector $\gamma$. We use conjugate gradient to minimize (22) with relative residual norm stopping tolerance $10^{-6}$, where the relative residual norm is defined $r_n = ||b - A\alpha^k||/||b||$.

**Algorithm 2 Unconstrained Augmented Lagrangian Method**

Given $\mu_0 \geq 0$ and initial points $\alpha^0, \lambda_0$ and $\gamma_0$, set $k = 0$.

1. Fix constant indicator matrices (21) using $\alpha^k$, then use conjugate gradient to find the approximate minimizer $\alpha^{k+1}$ of $\mathcal{L}_k(\alpha)$ in (22), with starting point $\alpha^k$ and with relative residual norm stopping tolerance of $10^{-6}$.
2. If $|y'\alpha^k| < 10^{-6}$ and bound constraints hold, stop with approximate solution $\alpha^{k+1}$.
3. Update Lagrange multipliers via (15) and (25) to obtain $\lambda_{k+1}$ and $\gamma_{k+1}$, and choose a new penalty parameter $\mu_{k+1}$ via (16).

4. Set $k = k + 1$ and go to Step 1.

Bertsekas notes in [2] that iterates $\alpha^k$ generated by Algorithm 2 converge to the solution $\alpha^*$ of the bound and equality constrained problem (18), and the updates to the Lagrange multipliers in (15) and (25) converge to $\lambda^*$ and $\gamma^*$, respectively.

To the best of our knowledge, the unconstrained augmented Lagrangian method represented by Algorithm 2 has not been applied to the SVM optimization problem.

3.2 Scaled Gradient Projection

The scaled gradient projection (SGP) algorithm was proposed for image restoration applications in [3]. We note that to our knowledge, this method has not been implemented for the solution of SVMs. We do so here.

To apply SGP to (10), we must make adjustments to the method presented in [3] and hence we present a majority of the algorithm here. We define the current iterate $\alpha^k$, the feasible set

$$\Omega = \{\alpha \in \mathbb{R}^N | l \leq \alpha \leq u, \ y^\prime \alpha = 0\},$$

and the projection operator

$$P_\Omega(\alpha) \equiv \arg \min_{\beta \in \Omega} ||\beta - \alpha|| = \arg \min_{\beta \in \Omega} \left(\frac{1}{2} \beta^\prime \beta - \beta^\prime \alpha\right). \quad (26)$$

The authors of [3] suggest a more general scaled projection, but we found (26) worked best.

Next we define

$$\beta^k = P_\Omega(\alpha^k - \lambda_k [A\alpha^k - b]), \quad (27)$$

to be the projection of the negative gradient path onto $\Omega$, where $\lambda_k$ is chosen by the Barzilai and Borwein-like steplength selection method described in [3]. If $\beta^k = \alpha^k$, we have $\alpha^k$ as a stationary point, otherwise the descent direction is taken as $d^k = \beta^k - \alpha^k$, and a backtracking line search is used to ensure a sufficient decrease in the objective function value at the current iterate over the last $M$ iterations. For $M = 1$, this backtracking loop is simply the Armijo rule [1]. These steps completely describe one iteration of the SGP method in [3].

The projection remains to be defined and is where our derivation differs from [3]. Given the quadratic programming problem (10), the SGP projection (27) is equivalent to the solution of the constrained and strictly convex quadratic program

$$\min_{\alpha} \frac{1}{2} \alpha^\prime \alpha - \alpha^\prime z \quad \text{s.t.} \quad l \leq \alpha \leq u \quad \text{and} \quad y^\prime \alpha = 0, \quad (28)$$
where \( z = \alpha^k - \lambda_k[A\alpha^k - b] \).

Letting \( \alpha^* \) be the solution of (28), under the KKT first order optimality conditions there exist Lagrange multipliers \( \lambda^* \in \mathbb{R} \) and \( \mu^*, \gamma^* \in \mathbb{R}^N \) such that the original constraints hold, as well as

\[
\alpha^* - z + \lambda^* y - \mu^* + \gamma^* = 0,
\]

\( \mu^*, \gamma^* \geq 0, \mu^* \circ \alpha^* = 0 \) and \( \gamma^* \circ (\alpha^* - \mu) = 0 \), where ‘\( \circ \)’ denotes component-wise multiplication. We can solve for \( \alpha^*, \mu^*, \) and \( \gamma^* \) as functions solely of \( \lambda^* \) using all but the original equality constraint:

\[
\begin{align*}
\alpha_i^* &= \max \{ l_i, \min \{ z_i - \lambda^* y_i, u_i \} \}, \\
\mu_i^* &= \max \{ 0, -(z_i - \lambda^* y_i) \}, \\
\gamma_i^* &= \begin{cases} 0 & \text{if } z_i - \lambda^* y_i \leq u_i \\ u_i + z_i - \lambda^* y_i & \text{otherwise} \end{cases}.
\end{align*}
\]

Thus, to solve the KKT system, we must find \( \lambda^* \) satisfying the remaining constraint

\[
\sum_{i=1}^{p} \alpha_i^*(\lambda^*) y_i = 0. \tag{30}
\]

The solution \( \lambda^* \) of (30) can be computed using one of many root finding algorithms. We use MATLAB’s \texttt{fzero}, while in [10] a secant-based method is introduced. Note, the solution \( \alpha^* \) defined by (29) and (30) is the desired SPG projection of (27).

Convergence of SGP is proved in [3]. The additional termination criterion for SGP used are a maximum number of iterations, or a projected gradient norm or step norm tolerance of \( 10^{-6} \), with the projected gradient defined in (17). Pseudocode for SGP can be found below. In our trials, initial parameter values are taken as \( \eta = 10^{-4}, \theta = 0.4, \lambda_{\min} = 10^{-5} \) and \( \lambda_{\max} = 10^5 \), as suggested in [3].

\textbf{Algorithm 3 Scaled Gradient Projection (SGP)}

\begin{enumerate}
\item Select \( \lambda_k \in [\lambda_{\min}, \lambda_{\max}] \) using the Barzilai and Borwein-like steplength selection method defined in [3].
\item Project \( \beta^k = P_\Omega(\alpha^k - \lambda_k \nabla f(\alpha^k)) \). If stopping tolerances are met, stop with approximate solution \( \alpha^{k+1} \).
\item Compute descent direction \( d^k = \beta^k - \alpha^k \).
\item Set \( \rho_k = 1 \) and \( f_{\max} = \max_{0 \leq j \leq \min(k, M-1)} f(\alpha^{k-j}) \).
\item Backtracking loop: If \( f(\alpha^k + \rho_k d^k) > f_{\max} + \eta \rho_k \nabla f(\alpha^k)' d^k \), then set \( \rho_k = \theta \rho_k \) and return to step 5.
\item Set \( \alpha^{k+1} = \alpha^k + \rho_k d^k \) and \( k = k + 1 \).
\end{enumerate}
4 Numerical Results

We now present two images for SVM classification using the methods presented here for comparison. Computational time, recorded in seconds, is the average of five trials. Initial parameter values for the bound constrained and unconstrained AL algorithms are set at $\lambda_0 = 5$ and $\mu_0 = 10$. Initial Lagrange vector $\gamma^0 = 1$ is used for the unconstrained AL algorithm. For all algorithms, regardless of constraints, the initial guess for $\alpha^0$ is chosen such that $(\alpha^0)'y = 0$.

Each classification is compared with Matlab’s built-in quadratic program solver `quadprog`, which uses an active-set method described in [17]. Furthermore, we also compare each classification with the widely-known sequential minimal optimization (SMO) algorithm, introduced by Platt in [27] as a simple, fast, and easy-to-implement technique for solving a bound and equality constrained SVM quadratic program. SMO has become a popular algorithm in the SVM literature and is often used as a comparative algorithm to test new methods [9, 6, 11, 21]. SMO solves a sequence of two-dimensional subproblems to obtain asymptotic convergence. Each subproblem requires the computation of updates to two Lagrange multipliers, one of which violates the KKT conditions. This method and its pseudocode are presented in [27].

We use 10-fold cross-validation to measure the performance of the algorithms [18, 28]. The $k$-fold cross-validation method partitions the training data set into $k$ approximately equal-sized parts. The first part is set aside as the new testing set and the remaining $k - 1$ parts become the new training set. The classifier is created with this adjusted training set and is used to classify the testing set. For example, when $k = 10$, this is equivalent to using 90% of the training set to classify the remaining 10% of the training set. Since the actual classification of the training set is known, the classification of the new testing set can be compared to the truth and the percent of misclassified pixels is noted. This process is repeated $k$ times (folds) so that each of the $k$ sets is used exactly once as a testing set. The average percent of misclassified pixels is reported. The recommended $k$ for $k$-fold cross validation is $k = 5$ or 10 [18].

4.1 Cat

The image of the cat displayed in Figure 4 is a gray-scale representation of a three banded RGB image of size $196 \times 293 \times 3$. Four classes are chosen for the training set as indicated in Figure 4: the cat, the black background, the gray background, and the carpet. A total of 181 training pixels are used, as indicated in the right image of Figure 4.

For trials on the cat image, the radial basis kernel with parameter $\sigma = 0.001$ is used, along with parameter $C = 10$. The classification scheme is more sensitive to the value of $\sigma$ than to $C$. Specifically, a change in $\sigma$ by an order of magnitude in either direction results in a poor classification of the image, while reducing $C$ by an order of magnitude yields a similar poor classification. Increasing $C$ does not affect the classification.
Classification of the cat image is presented in Figure 4. Table 1 summarizes the average CPU time and its standard deviations for five trials of the SVM formulation with the optimization techniques presented above. Note that the AL and SMO algorithms classify the image faster than the other two methods, with the unconstrained AL method leading in computational time. The 10-fold cross-validation error for each algorithm is 0.00%, most likely due to the fact that the selected training data was uniform and spatially well-separated. However, note that in the image, not every pixel is correctly classified, e.g. in the upper-left some carpet pixels are classified as cat pixels.

Table 1: CPU times for the SVM classification of the cat image

<table>
<thead>
<tr>
<th>Method</th>
<th>CPU avg.</th>
<th>Std. Dev.</th>
</tr>
</thead>
<tbody>
<tr>
<td>AL w/ PN</td>
<td>2.9604</td>
<td>0.0139</td>
</tr>
<tr>
<td>AL w/ CG</td>
<td>1.8451</td>
<td>0.1252</td>
</tr>
<tr>
<td>SGP</td>
<td>5.4044</td>
<td>0.0168</td>
</tr>
<tr>
<td>SMO</td>
<td>2.5585</td>
<td>0.1121</td>
</tr>
<tr>
<td>quadprog</td>
<td>4.7019</td>
<td>0.0273</td>
</tr>
</tbody>
</table>
4.2 Plumeria Flower

The image of the plumeria flower in Figure 5 is a gray-scale representation of an RGB image of size 302 × 369 × 3. Six classes are indicated by training data pixels in Figure 5: background grass, leaf, stem, bud, flower center and flower petal. There are a total of 300 training data pixels.

![Plumeria Flower Image](image)

**Fig. 5** Left: Original 302 × 369 × 3 plumeria flower image. Right: Training data for the six classes in the flower image, indicated in numerical order as background grass, leaf, stem, bud, flower center and flower petal. Bottom: The SVM classification of the image.

Classification with the bound and equality constrained SVM formulation is shown in Figure 5. For all SVM classifications, $C = 50$ and a radial basis kernel is used with $\sigma = 0.0005$. Decreasing $C$ by an order of magnitude results in a poor classification while an increase of the same yields a near identical classification, with classification only changing slightly in the lower left-hand corner of the image. Changing $\sigma$ by an order of magnitude in either direction results in a poor classification. Thus the classification is somewhat sensitive to the choice of both $\sigma$ and $C$.

Results from classification of the flower image are summarized in Table 2 with CPU time averaged over five trials, along with corresponding standard deviation. Notice the AL and SMO algorithms are competitive in computational time; however, the unconstrained AL algorithm outperforms the remaining
Table 2  CPU times for the SVM classification of the flower image

<table>
<thead>
<tr>
<th>Method</th>
<th>CPU avg.</th>
<th>Std. Dev.</th>
</tr>
</thead>
<tbody>
<tr>
<td>AL w/ PN</td>
<td>7.0759</td>
<td>0.0410</td>
</tr>
<tr>
<td>AL w/ CG</td>
<td>6.2379</td>
<td>0.0137</td>
</tr>
<tr>
<td>SGP</td>
<td>12.6780</td>
<td>0.0304</td>
</tr>
<tr>
<td>SMO</td>
<td>8.5972</td>
<td>0.1583</td>
</tr>
<tr>
<td>quadprog</td>
<td>27.9666</td>
<td>0.2339</td>
</tr>
</tbody>
</table>

algorithms in computational time. The 10-fold cross-validation error of the flower image for each algorithm is 0.00%, most likely for the same reasons as the previous example.

5 Conclusion

In this paper we have introduced an unconstrained augmented Lagrangian (AL) technique for numerically solving the bound and equality constrained quadratic program (QP) that arises in support vector machine (SVM) classification problems. We use the term unconstrained because the algorithm requires only the use of the conjugate gradient method or some other unconstrained quadratic minimization solver. We have also presented a bound constrained AL method for solving the SVM QP, which requires the minimization of bound constrained QP subproblems. Finally, we have presented an extension of the scaled gradient projection (SGP) method of [3] for solving the SVM QP. We compare these three methods with Matlab’s built-in constrained quadratic solver, quadprog, and with the sequential minimal optimization algorithm of [27] on the SVM QP problems arising in two image classification examples. In both cases, the unconstrained AL method is the most efficient. Moreover, it is straightforward to implement, suggesting that it will be useful to the wider SVM community.

References

31. Shilton, A., Palaniswami, M., Member, S., Ralph, D., Tsoi, A.C., Member, S.: Incremental training of support vector machines. IEEE Transactions on Neural Networks 16, 114–131 (2005)