An Alternating Direction Method of Multipliers Derivation for Support Vector Machines

Jarod Acanfrio, MATP 4820

May 7, 2021

Abstract

In this project I derive the Alternating Direction Method of Multipliers algorithm for optimizing the weights and bias of the L1-loss Support Vector Machine. By representing the regularization of the weight vector as a quadratic form of the augmented weight vector and bias term, we can update the weight vector and bias term at the same time, while only penalizing the feature weights. The soft-margin violations were expressed as a single variable which was defined as a function of the weight vector and bias term, and was the main equality constraint in the algorithm. On the most complex data set my derivation performed much faster and to the same accuracy as the provided Augmented Lagrange Method.

1 Introduction

In this project we consider the problem of finding a hyperplane to separate a dataset with binary labels while having as little violation of the classification margin as possible. This can be expressed as:

$$\min_{\mathbf{w},b} \sum_{i=1}^{N} \max(0, 1 - y_i(\mathbf{w}^T\mathbf{x}_i + b)) + \frac{\lambda}{2}\|\mathbf{w}\|^2 \tag{1}$$

Where $\mathbf{w} \in \mathbb{R}^p$ is the feature weight vector, $b \in \mathbb{R}$ is the bias term of the classifier, $\lambda > 0$ is a regularization parameter, $y_i = \pm 1, \mathbf{y} \in \mathbb{R}^N$ is the label of the ith data point and the label vector respectively, and $\mathbf{x}_i \in \mathbb{R}^p$ is the ith column of $\mathbf{X} \in \mathbb{R}^{p\times N}$ our data matrix.

We penalize large weights to avoid overfitting our classifier, and also aim to minimize the sum of the individual margin violations. These margin violations can be thought of as the distance from the weighted score for the ith data point from the classification margin on the number line.

2 Reformulation

To approach the problem, we first need to reformulate the minimization into the standard form of the Alternating Direction Method of Multipliers:

$$\min_{\mathbf{x},\mathbf{z}} f(\mathbf{z}) + g(\mathbf{z}) \text{ s.t. } \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z} = \mathbf{c} \tag{2}$$

2.1 Additional Variables

To distinguish between our own reformulation of the problem, and the original problem, all variables we introduce are denoted in bold capital letters, except for the given data matrix. The augmented data matrix is distinguished with a hat. These variables can be both vectors and matrices; the dimension of each variable will be included with its definition. Common constant vectors such as $\vec{0}, \vec{1}, \mathbf{y}, \mathbf{x}_i$ are denoted as bold lower case letters. We define the following variables:

$$\begin{aligned}
\mathbf{e} \in \mathbb{R}^N &= \mathbf{1} \\
\hat{\mathbf{X}} \in \mathbb{R}^{(p+1)\times N} &= \begin{bmatrix} \mathbf{X} \\ \mathbf{e}^T \end{bmatrix} \\
\mathbf{B} \in \mathbb{R}^{(p+1)\times(p+1)} &= \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0}^T & 0 \end{bmatrix} \\
\mathbf{A} \in \mathbb{R}^{N\times(p+1)} &= \mathbf{y} \odot \hat{\mathbf{X}}^T \\
\mathbf{W} \in \mathbb{R}^{p+1} &= \begin{bmatrix} \mathbf{w} \\ b \end{bmatrix} \\
\mathbf{T} \in \mathbb{R}^N &= \mathbf{e} - \mathbf{A}\mathbf{W}
\end{aligned}$$

Where $\mathbf{e}$ is the one vector, $\hat{\mathbf{X}}$ is the data matrix with an augmented row of 1's, $\mathbf{B}$ is the matrix used in the quadratic expression of the weight vector regularization, $\mathbf{A}$ is the element wise product of each label and each component in the corresponding data point, $\mathbf{W}$ is the vector of the feature weights and added bias term, and $\mathbf{T}$ is a primal variable that we introduce.

We first consider the sum of the margin violations, and see that they can be rewritten as:

$$\begin{aligned}
\sum_{i=1}^{N} \max(0, 1 - y_i(\mathbf{w}^T\mathbf{x}_i + b)) &= \| \max(0, \mathbf{e} - \mathbf{y} \odot \hat{\mathbf{X}}^T\mathbf{W})\|_1 \\
&= \| \max(0, \mathbf{e} - \mathbf{A}\mathbf{W})\|_1 \\
&= \| \max(0, \mathbf{T})\|_1
\end{aligned}$$

Where $\odot$ is the element wise multiplication operator. We can now express our margin violation as a function of a single variable.

Next, we consider the penalized weight vector squared ℓ_2-norm from the original minimization in (1). We want to update both the weight vector and the bias term at the same time, while only penalizing the feature weight vector. The variable that allows us to do this is the $\mathbf{B}$ matrix, which lets us express the weight vector norm-squared as a quadratic matrix product. Had the bias term been added onto the weight vector, and we took the standard ℓ_2-norm squared of it, it would also be penalized.

Consider the variables defined above. Expanding out the product we show our proposed matrix product is equivalent to the weight vector regularization term:

$$\begin{aligned}
\mathbf{W}^T\mathbf{B}\mathbf{W} &= \begin{bmatrix} \mathbf{w} & b \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0}^T & 0 \end{bmatrix} \begin{bmatrix} \mathbf{w} \\ b \end{bmatrix} \\
&= \begin{bmatrix} \mathbf{w} & 0 \end{bmatrix} \begin{bmatrix} \mathbf{w} \\ b \end{bmatrix} \\
&= \mathbf{w}^T\mathbf{w} \\
&= w_1^2 + w_2^2 + ... + w_p^2 \\
&= \sum_{i=1}^{p} w_i^2 \\
&= \|\mathbf{w}\|^2
\end{aligned}$$

Having expressed each term in the original minimization as a function of a single variable, with a single equality constraint relating the two primal variables we can rewrite the minimization problem about $\mathbf{W}, \mathbf{T}$ in the standard form of the ADMM:

$$\min_{\mathbf{W}\in\mathbb{R}^{p+1}, \mathbf{T}\in\mathbb{R}^N} \frac{1}{2}\mathbf{W}^T\mathbf{B}\mathbf{W} + \frac{1}{\lambda}\|\max(0, \mathbf{T})\|_1 \text{ s.t. } \mathbf{A}\mathbf{W} + \mathbf{T} = \mathbf{e} \tag{3}$$

Where in the standard form:

$$\begin{aligned}
f(\mathbf{W}) &= \frac{1}{2}\mathbf{W}^T\mathbf{B}\mathbf{W} \\
g(\mathbf{T}) &= \frac{1}{\lambda}\|\max(0, \mathbf{T})\|_1 \\
\mathbf{A} &= \mathbf{A} \\
\mathbf{B} &= \mathbf{I} \\
\mathbf{c} &= \mathbf{e}
\end{aligned}$$

3 KKT System and Variable Updates

3.1 Augmented Lagrange Function

The Augmented Lagrange Function for the standard form ADMM is:

$$\mathcal{L}_\beta(\mathbf{x}, \mathbf{z}, \mathbf{y}) = f(\mathbf{x}) + g(\mathbf{z}) + \mathbf{y}^T(\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z} - \mathbf{c}) + \frac{\beta}{2}\|\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z} - \mathbf{c}\|^2$$

Where $\beta > 0$ is a penalty parameter. The updates performed at iteration k are:

$$\begin{cases} \mathbf{x}^{(k+1)} = \arg\min_{\mathbf{x}} \mathcal{L}_\beta(\mathbf{x}, \mathbf{z}^{(k)}, \mathbf{y}^{(k)}) \\ \mathbf{z}^{(k+1)} = \arg\min_{\mathbf{z}} \mathcal{L}_\beta(\mathbf{x}^{(k+1)}, \mathbf{z}, \mathbf{y}^{(k)}) \\ \mathbf{y}^{(k+1)} = \mathbf{y}^{(k)} + \beta(\mathbf{A}\mathbf{x}^{(k+1)} + \mathbf{B}\mathbf{z}^{(k+1)} - \mathbf{c}) \end{cases}$$

The Augmented Lagrange Function of our Support Vector Machine optimization is:

$$\mathcal{L}_\beta(\mathbf{W}, \mathbf{T}, \mathbf{u}) = \frac{1}{2}\mathbf{W}^T\mathbf{B}\mathbf{W} + \frac{1}{\lambda}\|\max(0, \mathbf{T})\|_1 + \mathbf{u}^T(\mathbf{T} + \mathbf{A}\mathbf{W} - \mathbf{e}) + \frac{\beta}{2}\|\mathbf{T} + \mathbf{A}\mathbf{W} - \mathbf{e}\|^2$$

3.2 KKT System

The KKT system can be easily expressed now that the problem is in the ADMM standard form:

$$\begin{cases} \mathbf{A}\mathbf{W} + \mathbf{T} - \mathbf{e} = \mathbf{0} & \text{Primal Feasibility} \\ \mathbf{B}\mathbf{W} + \mathbf{A}^T\mathbf{u} = \mathbf{0} & \text{Dual Feasibility 1} \\ \mathbf{0} \in \partial g(\mathbf{T}) + \mathbf{u} & \text{Dual Feasibility 2} \end{cases} \tag{4}$$

Although $g(\mathbf{T})$ is convex, it is not smooth and cannot have a gradient defined. Instead, we have that at an optimal solution zero is contained in the set of vectors produced by the dual variable plus the subgradient at $\mathbf{T}^{(k)}$. A brief overview of subgradients provided by Ryan Tibshirani's course on Convex Optimization at Carnegie Mellon University was used to understand how to formally express this. We can express the dual feasibility about $\mathbf{W}$ as the gradient since the function is convex and smooth.

3.3 W Update

The update for our first primal variable is defined as:

$$\mathbf{W}^{(k+1)} = \arg\min_{\mathbf{W}} \mathcal{L}_\beta(\mathbf{W}, \mathbf{T}^{(k)}, \mathbf{u}^{(k)}) \tag{5}$$

Given that $f(\mathbf{W})$ is convex and smooth as it is quadratic, it is implied that at the updated value of $\mathbf{W}$, the optimality condition holds:

$$\begin{aligned} \vec{0} &= \nabla_{\mathbf{w}} \mathcal{L}_\beta(\mathbf{W}^{(k+1)}, \mathbf{T}^{(k)}, \mathbf{u}^{(k)}) \\ &= \mathbf{B}\mathbf{W}^{(k+1)} + \mathbf{A}^T\mathbf{u}^{(k)} + \beta\mathbf{A}^T(\mathbf{T} + \mathbf{A}\mathbf{W} - \mathbf{e}) \\ &= \mathbf{B}\mathbf{W}^{(k+1)} + \mathbf{A}^T\mathbf{u}^{(k)} + \beta\mathbf{A}^T\mathbf{T}^{(k)} + \beta\mathbf{A}^T\mathbf{A}\mathbf{W}^{(k+1)} - \beta\mathbf{A}^T\mathbf{e} \\ (\mathbf{B} + \beta\mathbf{A}^T\mathbf{A})\mathbf{W}^{(k+1)} &= \beta\mathbf{A}^T\mathbf{e} - \beta\mathbf{A}^T\mathbf{T}^{(k)} - \mathbf{A}^T\mathbf{u}^{(k)} \\ &= \beta\mathbf{A}^T(\mathbf{e} - \mathbf{T}^{(k)}) - \mathbf{A}^T\mathbf{u}^{(k)} \\ &= \mathbf{A}^T(\beta(\mathbf{e} - \mathbf{T}^{(k)}) - \mathbf{u}^{(k)}) \\ \mathbf{W}^{(k+1)} &= (\mathbf{B} + \beta\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T(\beta(\mathbf{e} - \mathbf{T}^{(k)}) - \mathbf{u}^{(k)}) \end{aligned}$$

Hence, we have a closed form solution for the $\mathbf{W}^{(k+1)}$ update. We can directly compute the next value of $\mathbf{W}$ in each iteration, and do not need to use an iterative subsolver algorithm to handle our first primal subproblem.

3.4 T Update

The update for our second primal variable is defined as:

$$\mathbf{T}^{(k+1)} = \arg\min_{\mathbf{T}} \mathcal{L}_\beta(\mathbf{W}^{(k+1)}, \mathbf{T}, \mathbf{u}^{(k)}) \tag{6}$$

Expanding out the Augmented Lagrange Function, and only keeping relevant terms with respect to the second primal subproblem we have:

$$\begin{aligned}
\mathbf{T}^{(k+1)} &= \arg\min_{\mathbf{T}} \frac{1}{2}(\mathbf{W}^{(k+1)})^T\mathbf{B}\mathbf{W}^{(k+1)} + \frac{1}{\lambda}\|\max(0,\mathbf{T})\|_1 + \mathbf{u}^T(\mathbf{T} + \mathbf{A}\mathbf{W}^{(k+1)} - \mathbf{e}) \\
&\quad + \frac{\beta}{2}\|\mathbf{T} + \mathbf{A}\mathbf{W}^{(k+1)} - \mathbf{e}\|^2 \\
&= \arg\min_{\mathbf{T}} \frac{1}{\lambda}\|\max(0,\mathbf{T})\|_1 + \mathbf{u}^T(\mathbf{T} + \mathbf{A}\mathbf{W}^{(k+1)} - \mathbf{e}) + \frac{\beta}{2}\|\mathbf{T} + \mathbf{A}\mathbf{W}^{(k+1)} - \mathbf{e}\|^2
\end{aligned}$$

Completing the square with the last two terms in the equation above, we can eliminate another term from the second primal subproblem:

$$\frac{\beta}{2}\|\mathbf{T}+\mathbf{A}\mathbf{W}^{(k+1)}-\mathbf{e}\|^2+\mathbf{u}^T(\mathbf{T}+\mathbf{A}\mathbf{W}^{(k+1)}-\mathbf{e}) = \frac{\beta}{2}\|\mathbf{T}+\mathbf{A}\mathbf{W}^{(k+1)}-\mathbf{e}+\frac{1}{2}\mathbf{u}^{(k)}\|^2-\frac{1}{4}\|\mathbf{u}^{(k)}\|^2$$

Plugging this back into the expression for the $\mathbf{T}$ update:

$$\begin{aligned}
\mathbf{T}^{(k+1)} &= \arg\min_{\mathbf{T}} \frac{1}{\lambda}\|\max(0,\mathbf{T})\|_1 + \mathbf{u}^T(\mathbf{T} + \mathbf{A}\mathbf{W}^{(k+1)} - \mathbf{e}) + \frac{\beta}{2}\|\mathbf{T} + \mathbf{A}\mathbf{W}^{(k+1)} - \mathbf{e}\|^2 \\
&= \arg\min_{\mathbf{T}} \frac{1}{\lambda}\|\max(0,\mathbf{T})\|_1 + \frac{\beta}{2}\|\mathbf{T} + \mathbf{A}\mathbf{W}^{(k+1)} - \mathbf{e} + \frac{1}{2}\mathbf{u}^{(k)}\|^2 - \frac{1}{4}\|\mathbf{u}^{(k)}\|^2 \\
&= \arg\min_{\mathbf{T}} \frac{1}{\lambda\beta}\|\max(0,\mathbf{T})\|_1 + \frac{1}{2}\|\mathbf{T} - (\mathbf{e} - \mathbf{A}\mathbf{W}^{(k+1)} - \frac{1}{2}\mathbf{u}^{(k)})\|^2 \\
&= \text{prox}_{\frac{1}{\lambda\beta}\|\max(0,\cdot)\|_1}(\mathbf{e} - \mathbf{A}\mathbf{W}^{(k+1)} - \frac{1}{2}\mathbf{u}^{(k)})
\end{aligned}$$

For the conservation of space later in the paper, let us define the vector passed to the proximal mapping as:

$$\mathbf{Q}^{(k)} \in \mathbb{R}^N = \mathbf{e} - \mathbf{A}\mathbf{W}^{(k+1)} - \frac{1}{2}\mathbf{u}^{(k)} \tag{7}$$

We are now tasked with solving for the general form of this proximal mapping.

3.4.1 Proximal Mapping for T Update

Consider a general input $\mathbf{a} \in \mathbb{R}^N$ to the proximal operator:

$$\begin{aligned}\text{prox}_{\frac{1}{\lambda\beta}\|\max(0,\cdot)\|_1}(\mathbf{a}) &= \text{argmin}_{\mathbf{T}} \frac{1}{\lambda\beta}\|\max(0,\mathbf{T})\|_1 + \frac{1}{2}\|\mathbf{T}-\mathbf{a}\|^2 \\ &= \arg\min_{\mathbf{T}} \frac{1}{\lambda\beta}\sum_{i=1}^{N}\max(0,T_i) + \frac{1}{2}\sum_{i=1}^{N}(T_i-a_i)^2\end{aligned}$$

For each component of $\mathbf{T}$ we solve the piecewise quadratic minimization:

$$\begin{aligned}T_i^{(k+1)} &= \arg\min_{T_i} \frac{1}{\lambda\beta}\max(0,T_i) + \frac{1}{2}(T_i-a_i)^2 \\ &= \arg\min_{T_i} \begin{cases} \frac{1}{2}T_i^2 + \frac{1}{2}a_i^2 - T_i a_i & T_i \le 0 \\ \frac{1}{2}T_i^2 + \frac{1}{2}a_i^2 - T_i a_i + \frac{1}{\lambda\beta}T_i & T_i > 0 \end{cases} \\ &= \arg\min_{T_i} \begin{cases} \frac{1}{2}T_i^2 - T_i a_i & T_i \le 0 \\ \frac{1}{2}T_i^2 + (\frac{1}{\lambda\beta} - a_i)T_i & T_i > 0 \end{cases}\end{aligned}$$

Denote $\gamma(T_i)$:

$$\gamma(T_i) = \begin{cases} \frac{1}{2}T_i^2 - T_i a_i & T_i \le 0 \\ \frac{1}{2}T_i^2 + (\frac{1}{\lambda\beta} - a_i)T_i & T_i > 0 \end{cases}$$

The value of T_i which is the argmin is the value of T_i where the partial derivative is equal to zero:

$$0 = \frac{\partial\gamma}{\partial T_i} = \begin{cases} T_i - a_i & T_i \le 0 \\ T_i + \frac{1}{\lambda\beta} - a_i & T_i > 0 \end{cases} \tag{8}$$

We discuss the possible cases of a_i.

Case 1: $a_i \le 0$
In the case that $a_i \le 0$ we see that (8), $T_i \le 0$, would equal zero when $T_i = a_i \implies T_i \le 0$. This is in line with our assumption that $T_i \le 0$ in this part of the piecewise quadratic function. For (8), $T_i > 0$, the value of T_i that makes the partial derivative zero is $T_i = a_i - \frac{1}{\lambda}\beta \implies T_i < 0$. which violates our definition of T_i in this region. Hence, when $a_i \le 0$:

$$\text{prox}_{\frac{1}{\lambda\beta}\|\max(0,\cdot)\|_1}(a_i) = a_i, \forall a_i \le 0, i = 1, ..., N$$

Case 2: $a_i > 0$
For the case that $a_i > 0$ we again check to see the sign on values of T_i that cause the partial derivative to be zero. For equation (8), $T_i \le 0$, to be equal to zero we would have $T_i = a_i$. This means that $T_i > 0$, which directly violates the definition of T_i in this part of the piecewise function. For equation (8), $T_i > 0$,

we have that $T_i = a_i - \frac{1}{\lambda\beta}$ when the partial derivative is equal to zero. The sign of T_i for different values of a_i is:

$$\text{sign}(T_i) = \begin{cases} -1 & a_i \in (0, \frac{1}{\lambda\beta}) \\ 0 & a_i = \frac{1}{\lambda\beta} \\ 1 & a_i > \frac{1}{\lambda\beta} \end{cases}$$

These results only agree with the definition of T_i for the quadratic piecewise function when $a_i > \frac{1}{\lambda\beta}$. Values less than or equal to this, but greater than zero will form our third case of a_i considered. Hence, when $a_i > \frac{1}{\lambda\beta}$:

$$\text{prox}_{\frac{1}{\lambda\beta}\|\max(0,T_i)\|_1}(a_i) = a_i - \frac{1}{\lambda\beta}, \forall a_i > \frac{1}{\lambda\beta}, i = 1, ..., N$$

Case 3: $0 < a_i \leq \frac{1}{\lambda\beta}$
For the case that $0 < a_i \leq \frac{1}{\lambda\beta}$ we see that there is no part of the piecewise partial derivative that can be solved for a value of T_i that agrees with the constraints. Graphically, in this region each part of the piecewise that corresponds to a positive a_i value has its minimizer outside of the interval it is defined on.

Let us consider a point:

$$\hat{a}_i \in (0, \frac{1}{\lambda\beta}) = \frac{1}{\lambda\beta} - \delta, 0 < \delta < \frac{1}{\lambda\beta}$$

For $\hat{a}_i$, the part of the piecewise quadratic corresponding to $T_i \leq 0$ has its minimzer at $T_i^* > 0$. As a result, the closest we can get to T_i^* is at $T_i = 0$. Likewise, for $T_i > 0$ the minimizer of the quadratic piecewise part is at $T_i^* \leq 0$. Again, the closest we can get to T_i^* is at $T_i = 0$. Therefore, we have that:

$$\text{prox}_{\frac{1}{\lambda\beta}\|\max(0,T_i)\|_1}(a_i) = 0, \forall a_i \in (0, \frac{1}{\lambda\beta}), i = 1, ..., N$$

After considering all relevant cases, we can write the proximal mapping as:

$$\text{prox}_{\frac{1}{\lambda\beta}\|\max(0,\cdot)\|_1}(\mathbf{a}) = \begin{cases} a_i & a_i \leq 0 \\ 0 & 0 < a_i < \frac{1}{\lambda\beta} \\ a_i - \frac{1}{\lambda\beta} & a_i \geq \frac{1}{\lambda\beta} \end{cases}, \forall i = 1, ..., N$$

3.5 u Update

The **u** update is simply the current iterate plus the primal residual at the end of our primal updates multiplied by the penalty parameter:

$$\mathbf{u}^{(k+1)} = \mathbf{u}^{(k)} + \beta(\mathbf{T}^{(k+1)} + \mathbf{A}\mathbf{W}^{(k+1)} - \mathbf{e})$$

3.6 Complete Iterative Update

Over each iteration we perform the following updates in the order presented, as we know that alternating the first primal variable to be updated each iteration doesn't guarantee convergence.

$$\mathbf{W}^{(k+1)} = (\mathbf{B} + \beta \mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T (\beta(\mathbf{e} - \mathbf{T}^{(k)}) - \mathbf{u}^{(k)}) \tag{9}$$

$$\mathbf{T}^{(k+1)} = \text{prox}_{\frac{1}{\lambda\beta}\|\max(0,\cdot)\|_1}(\mathbf{Q}^{(k)}) = \begin{cases} Q_i^{(k)} & Q_i^{(k)} < 0 \\ 0 & 0 \leq Q_i^{(k)} \leq \frac{1}{\lambda\beta} \\ Q_i^{(k)} - \frac{1}{\lambda\beta} & Q_i^{(k)} > \frac{1}{\lambda\beta} \end{cases}, \forall i = 1, ..., N \tag{10}$$

$$\mathbf{u}^{(k+1)} = \mathbf{u}^{(k)} + \beta(\mathbf{T}^{(k+1)} + \mathbf{A}\mathbf{W}^{(k+1)} - \mathbf{e}) \tag{11}$$

4 Residuals and Stopping Conditions

The primal residual is computed as the violation of our affine equality constraint:

$$\mathbf{r}_p^{(k+1)} = \mathbf{T}^{(k+1)} + \mathbf{A}\mathbf{W}^{(k+1)} - \mathbf{e}$$

For the dual residual, we recall the optimality conditions about $\mathbf{W}$, as well as the definiton of the $\mathbf{u}$ update. In theory, at an optimal $\mathbf{W}^{(k+1)}$ the following optimality condition should hold:

$$\begin{aligned}
\vec{0} &= \mathbf{B}\mathbf{W}^{(k+1)} + \mathbf{A}^T\mathbf{u}^{(k)} + \beta\mathbf{A}^T(\mathbf{T}^{(k)} + \mathbf{A}\mathbf{W}^{(k+1)} - \mathbf{e}) \\
&= \mathbf{B}\mathbf{W}^{(k+1)} + \mathbf{A}^T(\mathbf{u}^{(k+1)} - \beta(\mathbf{T}^{(k+1)} + \mathbf{A}\mathbf{W}^{(k+1)} - \mathbf{e})) + \beta\mathbf{A}^T(\mathbf{T}^{(k)} + \mathbf{A}\mathbf{W}^{(k+1)} - \mathbf{e}) \\
&= \mathbf{B}\mathbf{W}^{(k+1)} + \mathbf{A}^T\mathbf{u}^{(k+1)} - \beta\mathbf{A}^T(\mathbf{T}^{(k+1)} + \mathbf{A}\mathbf{W}^{(k+1)} - \mathbf{e}) + \beta\mathbf{A}^T(\mathbf{T}^{(k)} + \mathbf{A}\mathbf{W}^{(k+1)} - \mathbf{e}) \\
&= \mathbf{B}\mathbf{W}^{(k+1)} + \mathbf{A}^T\mathbf{u}^{(k+1)} + \beta\mathbf{A}^T(\mathbf{T}^{(k)} - \mathbf{T}^{(k+1)})
\end{aligned}$$

We can see that the first two terms are the same as in the KKT system, (4), for the first dual feasibility equation about $\mathbf{W}$. The term $\beta\mathbf{A}^T(\mathbf{T}^{(k)} - \mathbf{T}^{(k+1)})$ is not part of the KKT system, and is the residual of the dual feasibility conditions.

$$\mathbf{r}_d^{(k+1)} = \beta\mathbf{A}^T(\mathbf{T}^{(k)} - \mathbf{T}^{(k+1)})$$

The stopping condition of the algorithm is that both of the following are satisfied:

$$\|\mathbf{T}^{(k+1)} + \mathbf{A}\mathbf{W}^{(k+1)} - \mathbf{e}\|_2 \leq \text{tol}, \|\beta\mathbf{A}^T(\mathbf{T}^{(k)} - \mathbf{T}^{(k+1)})\|_2 \leq \text{tol}$$

5 Code and Output Analysis

5.1 Algorithm Implementation

My algorithm performs all computation within the main while-loop. The W update had a closed form solution and didn't need to make a function call to

generate the new iterate. By avoiding the function call I was able to eliminate the need to allocate and delete space in the scope of the function during each iteration; this was done with the goal of improving algorithm speed. Additionally all constant matrix products, and inverses are computed at the start of the script and stored globally, so the same computation is not made every iteration. Profiling the code to look for bottlenecks, my $\mathbf{W}$ update is the slowest line, taking up about 80% of the processing time. Having stored all constants and inverses for the update, I cannot make this calculation any faster. My $\mathbf{T}$ update is a simple for-loop over all elements, which updates each component according to its proximal mapping. While this could have been moved outside of the main function, there would still be the overhead operations and memory allocations associated with the function call. In doing a simple parameter optimization I found that $\lambda = 0.1$ and $\beta \in [2, 2.5]$ worked best, with a β in this range speeding up the algorithm convergence on some test cases by a factor of 2 to 3 times.

In running tests on my solver, I decided to set all tolerances to $1e-4$. The analysis of the algorithm was primarily focused on the case where there was no maximum iteration condition on the outer loop. I was curious about how the algorithm would converge and behave overtime. All reported graphs are for tests with no maximum iteration condition. For data on trials with this condition, there is a "MI" after the algorithm name.

5.2 Rho02 Test Case

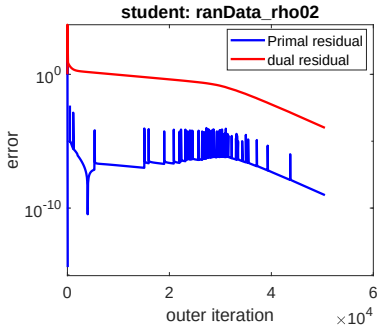

Figure 1: Student rho02

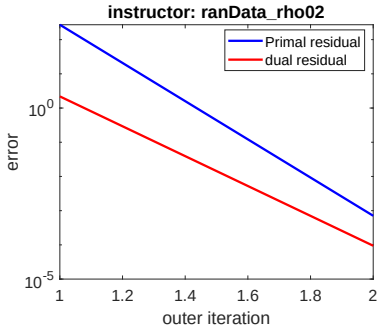

Figure 2: Instructor rho02

Test	Algorithm	Out Iter.	PR	DR	Acc.	Time (s)
rho02	ADMM	47198	$9.31e-10$	$9.99e-5$	97.5%	146.75
rho02	ADMM MI	1000	$6.27e-6$	1.75	97.0%	2.98
rho02	ALM	1	$7.08e-4$	$9.73e-5$	97.5%	0.72

On the rho02 data set, after running until both residuals are below the tolerance, my algorithm achieves the same accuracy as the instructors. It is worth

noting that my algorithm converges significantly slower than the instructors algorithm, taking over two minutes. Looking at my algorithm with the maximum iteration condition, it achieves a 97% accuracy in 1000 iterations over 2.98 seconds, which is not bad. The primal residual is under the tolerance, but the dual residual is 4 orders of magnitude away from the tolerance.

Looking at the residual graphs, it is apparent that the dual residual in my algorithm gradually converges, and that the primal residual sees frequent brief spikes. The primal residual spikes to below $1e-10$ and then shoots back up before stabilizing, and converging. The instructors residuals converge to below the tolerance after a single outer iteration.

5.3 Rho08 Test Case

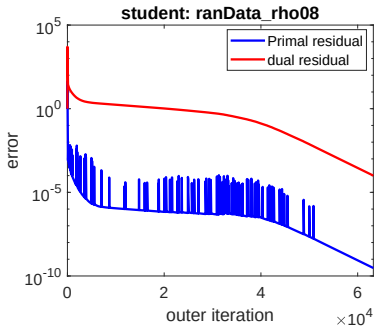

Figure 3: Student rho08

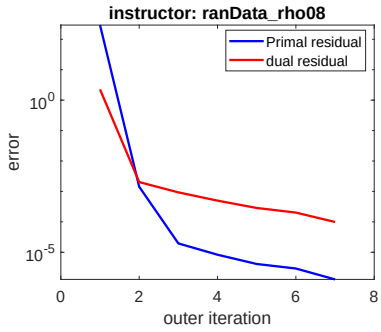

Figure 4: Instructor rho08

Test	Algorithm	Out Iter.	PR	DR	Acc.	Time (s)
rho08	ADMM	63236	$2.99e-10$	$9.99e-5$	88.0%	205.39
rho08	ADMM MI	1000	$2.68e-5$	3.37	84.5%	3.11
rho08	ALM	8	$3.36e-7$	$9.99e-5$	88.0%	1.98

On the rho08 data set, my algorithm achieves the same accuracy as the instructors when run until both residuals converge, but takes even longer to converge than on rho02, needing over three minutes. For this data set, the erratic early stage behavior of the primal residual no longer happnes, but we still observe the frequent spikes in the error. Overall both residuals are stable and gradually converge. Introducing the maximum iteration condition, my algorithm scores a few accuracy points lower than the instructors, correctly classifying 84.5% of data points. It achieves this in 3.11 seconds, with a primal residual below the tolerance, and the dual residual again 4 orders of magnitude away from the tolerance. The instructors algorithm also took slightly longer than on the rho02 data set, but still ran significantly faster than my own.

5.4 Spam Test Case

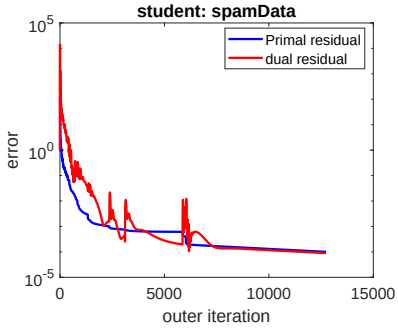

Figure 5: Student spam

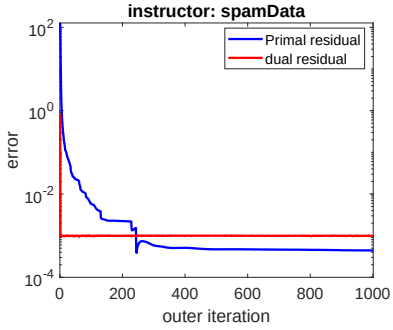

Figure 6: Instructor spam

Test	Algorithm	Out Iter.	PR	DR	Acc.	Time (s)
spam	ADMM	12733	$9.99e-5$	$8.96e-5$	79.7%	36.78
spam	ADMM MI	1000	$4.31e-3$	$1.06e-1$	79.9%	2.81
spam	ALM	1000	$4.41e-4$	$9.94e-4$	79.7%	200.94

Interestingly enough, my algorithm is superior on the spam data set. When allowed to run until both residuals are below the tolerance, my algorithm converges in 36.78 seconds, with a 79.7% accuracy. Including the maximum iteration condition, my algorithm achieves a 79.9% accuracy in 2.81 seconds, lowering the primal and dual residual to $4.31e-3, 1.06e-1$ respectively. While still not below the tolerance, this run had the best performance out of all three on the data set. In this test case my primal residual did not show any spikes and consistently decreased over the iterations. The dual residual bounced around quite a bit, experiencing its largest point of instability right after the primal residual took its last steep decline. The instructors dual residual almost converged immediately, and the primal approached the tolerance over the maximum number of iterations, but neither fully converged.

5.5 Thoughts and Observations

On the easier data sets, the instructors ALM implementation ran much faster and just as accurate as my ADMM. Surprisingly, my algorithm ran in less than a minute on the spam data. In all cases, when the maximum iteration condition was included my algorithm achieved only slighly lower accuracy in two cases, and a higher accuracy in the third. It has been noted by Boyd, Parikh, Chu, Peleato, Eckstein (2010) that the Alternating Direction Method of Multipliers does not converge to high accuracy very quickly, although this wasn't the case with my algorithm. They further make the point that ADMM will converge to a reasonable accuracy within tens of iterations, and that machine learning

parameter optimization problems don't typically benefit from super accurate fits as they will still have a relatively similar classification error

I am curious as to why my primal residuals frequently spiked in the rho data sets. I wonder if my formulation works better for certain structures of problems or data.

6 Conclusion

In this project I formulated an equivalent problem to the Support Vector Machine objective function, and applied the Alternating Direction Method of Multipliers to solve it. My derivation utilized a quadratic form representation of the feature weight vector ℓ_2-norm squared to update both the weights and the bias at the same time, while only regularizing the parameter weights in the vector. Using introduced variables, the margin violation is written as a function of a single variable, and its proximal mapping is computed. After testing the algorithm against the instructors on various data sets, the implemented ADMM consistently achieves the same accuracy as the professors, and is significantly faster on the most challenging data set, running six times faster.

References

[1] Boyd, S., Parikh, N., Chu, E., Peleato, B., Eckstein, J. (2011). *Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers*. Foundations and Trends in Machine Learning.

[2] Tibshirani, R. (2019). *Subgradients*. Convex Optimization 10-725, Fall 2019, Carnegie Mellon University.