# Distributed Support Vector Machines via the Alternating Direction Method of Multipliers

Anders Poirel,
AM 229, Fall 2020

December 17, 2020

# 1 Alternating Descent Method of Multipliers

## 1.1 Motivation

The *Alternating Descent Method of Multipliers* (ADMM) algorithm is motivated by a desire to combine the best properties of two simpler algorithms while overcoming their respective limitations [1].
On one hand, the *dual descent* algorithm has subproblems that can be easily distributed. However, this comes at the cost of strong assumptions on the form of the problem. On the other hand, the *method of multipliers* (MM) modifies dual ascent to relax the required assumptions, but this sacrifies the ability to distribute parts of the algorithm.

## 1.2 Algorithm

The ADMM algorithm is used solves problems of the form

$$\begin{aligned} \min_{x,z} \quad & f(x) + g(z) \\ \text{subj. to} \quad & Ax + Bz = c \end{aligned} \tag{1}$$

where $x \in \mathbb{R}^m, z \in \mathbb{R}^n, A \in R^{m \times p}, B \in \mathbb{R}^{n \times p}, c \in \mathbb{R}^p$

Like MM, the ADMM algorithm actually solves an equivalent augmented problem

$$\begin{aligned} \min_{x,z} \quad & f(x) + g(z) + \frac{\rho}{2}\|Ax + Bz - c\|_2^2 \\ \text{subj. to} \quad & Ax + Bz = c \end{aligned} \tag{2}$$

Indeed, any satisfiable pair $x, z$ zeroes out the extra term in the objective, recovering problem (1).
The associated augmented Lagrangian is

$$L_\rho(x, z, \nu) = f(x) + g(z) + \nu^\top(Ax + Bz + c) + \frac{\rho}{2}\|Ax + Bz - c\|_2^2$$

The algorithm is given by

**Algorithm 1:** ADMM

**for** $k = 1, 2, \ldots$ **do**
  $x_{k+1} \leftarrow \operatorname{argmin}_x L_\rho(x, z_k, \nu_k)$
  $x_{k+1} \leftarrow \operatorname{argmin}_z L_\rho(x_{k+1}, z, \nu_k)$
  $\nu_{k+1} \leftarrow \nu_k + \rho(Ax_{k+1} + Bz_{k+1} - c)$
**end**

### 1.3 Properties of ADMM

**Convergence**
A major advantage of ADMM is that it converges in many common scenario where dual ascent does not. For instance, dual ascent does not converge if the objective is linear [1]. In fact if the problem satisfies the following assumptions

(1) epi(f) and epi(g) are closed, nonempty convex sets

(2) The Lagrangian of problem formulation 1 has a saddle point

Then that is, the points approach feasability and the objective approaches its optimum

$$Ax_k + Bz_k \to 0 \text{ as } k \to \infty$$
$$f(x_k) + g(z_k) \to p^* \text{ as } k \to \infty$$

These assumptions hold in many applications, including the example discussed in a later section [1].

**Seperability**
The function $f$ is *seperable* if

$$f(x) = \sum_{i=1}^{n} f + i(x_i)$$

where $x = (x_1, \ldots, x_n)$ and $x_i$ are subvectors of $x$. If either $f$ or $g$ in the ADMM problem (1) are seperable, then the $x$ and $z$- minimization steps in the algorithm can be decomposed into independant problems [1], which allows the algorithm to conduct these updates in parallel, yielding distributed solvers for many problems.

## 2 Distributed ADMM

### Global variable consensus ADMM

The power of the ADMM becomes clear when the objective of the minimization problem is *additive*, i.e. when it is of the form

$$\min \quad \sum_{i=1}^{N} f_i(x) \tag{3}$$

where each of the $f_i$ are convex. In this situation, one can derive a version of the ADMM algorithm that can be distributed, which can yield a sizeable improvement in performance. To obtain a distributed algorithm, the objective function should be *seperable*

Now, to transform an addditive objective into a seperable objective, the problem can be rewritten as a *global consensus problem* with "local variables" which are all constrained to be equal to the original variable $x$ [2],

$$\begin{aligned} \min \quad & \sum_{i=1}^{N} f_i(x_i) \\ \text{subj. to} \quad & x_i = z \quad \forall i = 1, \ldots, n \end{aligned} \tag{4}$$

In many situations, the objective has an extra term in $x$, which is rewritten in terms of $z$

$$\begin{aligned} \min \quad & \sum_{i=1}^{N} f_i(x_i) + g(z) \\ \text{subj. to} \quad & x_i = z \quad \forall i = 1, \ldots, n \end{aligned} \tag{5}$$

The augmented Lagrangian is then

$$L_\rho(x_1, \ldots, x_N, z, \nu) = g(z) + \sum_{i=1}^{N} \left( f_i(x_i) + (\nu_i)_k^\top (x_i - z) + \frac{\rho}{2} \|x_i - z\|_2^2 \right)$$

which yields the following ADMM algorithm after simplifying the z-update step [1]:

**Algorithm 2:** Global variable consensus ADMM

**for** $k = 1, 2, \ldots$ **do**

$(x_i)_{k+1} \leftarrow \text{argmin}_{x_i} \left( f_i(x_i) + (\nu_i)_k^\top (x_i - z_k) + \frac{\rho}{2} \|x_i - z_k\|_2^2 \right)$

$z_{k+1} \leftarrow \text{argmin}_z \left( g(z) + \frac{N\rho}{2} \|z - \overline{x}_{k+1} - (1/\rho)\overline{\nu}_k\|_2^2 \right)$

$(\nu_i)_{k+1} \leftarrow (\nu_i)_k + \rho((\beta_i)_k - z_{k+1})$

**end**

This algorithm performs $N$ distinct $x_i$ and $\nu_i$-updates at each step, each of which is can be done in parallel as they are independant of one-another. The $z$-update is handled by a central process, and coordinates the solutions of the subproblems solved in the $x, \nu$ updates.

### 2.1 Regularized Model Estimation

The ADMM form (1) is particularly natural for regularized model estimation problems, where the objective can be written as the sum of a "loss" and a "penalty" term, i.e.

$$\min_{\beta} \quad l(\beta, X, y) + r(\beta)$$

where $X \in \mathbb{R}^{m \times p}$ is a matrix of training examples, $y \in \mathbb{R}^m$ is the vector of training labels, and $\beta$ is a vector of model parameters. If the loss function $l$ and regularization function $r$ are respectively convex, the substitution $\beta = z$ immediately yields a problem in desired form (1)

$$\begin{aligned} \min_{\beta, z} \quad & l(\beta, X, y) + r(z) \\ \text{subj. to} \quad & \beta - z = 0 \end{aligned} \tag{6}$$

When $l$ is seperable, this problem can be solved using the global consensus ADMM algorithm by splitting the problem over blocks of data [1], such that

$$X = \begin{bmatrix} X_1 \\ \vdots \\ X_N \end{bmatrix}, \quad y = \begin{bmatrix} y_1, \\ \vdots \\ y_n \end{bmatrix}$$

where $X_i \in \mathbb{R}^{m_i \times p}$, $y \in \mathbb{R}^{m_i}$, $\sum_{i=1}^n m_i = m$, and $l_i$ is the loss functions on the $i^{th}$ block of data. The algorithm is then, following the result for problems of the form (5) and using uses the more compact scaled form described in [1],

**Algorithm 3:** Consensus global variable ADMM for regularizated model estimation

**for** $k = 1, 2, \dots$ **do**

$$(\beta_i)_{k+1} \leftarrow \operatorname{argmin}_{\beta_i} \left( (l_i(\beta_i, X_i, y_i) + (\rho/2)\|\beta_i - z_k + (u_i)_k\|_2^2 \right)$$

$$z_{k+1} \leftarrow \operatorname{argmin}_z \left( r(z) + \frac{N\rho}{2}\|z\|_2^2 - \overline{\beta}_{k+1} - \overline{u}_k \right)$$

$$(u_i)_{k+1} \leftarrow (u_i)_k + (\beta_i)_k - z_{k+1}$$

**end**

The next section demonstrates a worked out example of a straighforward application of this algorithm.

# 3 Support Vector Classifiers

## 3.1 Problem formulation

Letting $x_1, \dots x_m \in \mathbb{R}^p$ be the training examples, and $y_1, \dots, y_N \in \{-1, 1\}$ be the training labels, the standard form of the binary *support vector classifer* (SVC) problem is given in [3] by

$$\begin{aligned} \operatorname{argmin}_{\beta, \beta_0, \xi} \quad & \|\beta\|_2^2 & \\ \text{subj. to} \quad & y_i(x_i^\top \beta + \beta_0) \geq 1 - \xi & \forall i = 1, \dots, m \\ & \xi_i \geq 0 & \forall i = 1, \dots, m \\ & \sum_{i=1}^m \xi_i \leq C & C \text{ constant} \end{aligned} \tag{7}$$

Equivalently, we can formulate this as

$$\begin{aligned} \operatorname{argmin}_{\beta, \beta_0, \xi} \quad & \frac{1}{2}\|\beta\|_2^2 + C\sum_{i=1}^m \xi_i & \\ \text{subj. to} \quad & y_i(x_i^\top \beta + \beta_0) \geq 1 - \xi_i & \forall i = 1, \dots, m \\ & \xi_i \geq 0 & \forall i = 1, \dots, m \end{aligned} \tag{8}$$

Indeed, this minimum is defined iff $\sum \xi_i$ is finite.

### 3.2 Motivation

In many potential applications of the SVC, there are a large number of training samples (say, millions), each of relatively modest dimensionality - i.e. $N$ is much larger than $p$. While standard solvers for linear support vector classifers perform well in this situation [4]. the standard `libSVM` solvers for nonlinear kernels struggle on problems of this scale when the data is non-sparse [5].

The approach presented here uses the alternating descent method of multipliers to derive a parallelizable algorithm for fitting support vector classifiers, which will be capable of handling this type of large-scale problem. While only the linear classifier is shown here due to space constraints, extending this technique to non-linear kernels is straightforward [6].

### 3.3 Derivation of a distributed algorithm for SVC

The first goal is to rewrite the SVC problem (8) in a more convenient form to apply the global consensus variable ADMM algorithm. Formulation (8) is equivalent to

$$\begin{aligned} \text{argmin}_{\beta,\beta_0,\xi} \quad & \frac{1}{2}\|\beta\|_2^2 + C\sum_{i=1}^{m}\xi_i \\ \text{subj. to} \quad & \xi_i = [1 - y_i(\beta^\top x_i + \beta_0)]_+ \end{aligned}$$

where $[1 - y_i(\beta^\top x_i + \beta_0)]_+ = \max(0, 1 - y_i(\beta^\top x_i + \beta_0))$ . This is then equivalent to

$$\begin{aligned} \text{argmin}_{\beta,\beta_0,\xi} \quad & \frac{1}{2C}\|\beta\|_2^2 + \sum_{i=1}^{m}\xi_i \\ \text{subj. to} \quad & \xi_i = [1 - y_i(\beta^\top x_i + \beta_0)]_+ \end{aligned}$$

Now, minimizing over $\xi$, this problem is equivalent to

$$\text{argmin}_{\beta,\beta_0} \quad \sum_{i=1}^{m}[1 - y_i(\beta^\top x_i + \beta_0)]_+ \frac{1}{2C}\|\beta\|_2^2 \tag{9}$$

Indeed, to minimize $\sum \xi_i$ it suffices to take the smallest $\xi_i$ allowed by the constraints, that is $[1-y_i(\beta^\top x_i+\beta_0)]_+$. This corresponds to the *penalization* *method formulation of the SVC* given in [3].

The problem is now in the desired "loss + penalty" form. The term $\sum_{i=1}^{n}[1 - y_i(\beta^\top x_i + \beta_0)]_+$ is seperable in $x_1, \ldots, x_m$. Furthermore, both terms are convex in $\beta$ as $[1 - y_i(\beta^\top x_i + \beta_0)]_+$ is the pointwise maximum of two convex functions.

Thus substituting $z = \beta$,

$$\begin{aligned} \operatorname{argmin}_{\beta,\beta_0} \quad & \sum_{i=1}^{n}[1 - y_i(\beta^\top x_i + \beta_0)]_+ \frac{1}{2C}\|\beta\|_2^2 \\ \text{subj. to} \quad & x_i - z_i = 0 \quad \forall i = 1, \ldots, m \end{aligned} \tag{10}$$

This problem can be solved using the global variable consensus ADMM algorithm, where $X_i = \begin{bmatrix} x_{i1} & \ldots & x_{im_i} \end{bmatrix}^\top \in \mathbb{R}^{m_i \times p}$

---

**Algorithm 4:** Naive global variable consensus ADMM for SVC

---

**for** $k = 1, 2, \ldots$ **do**

$(\beta_i)_{k+1} \leftarrow$

$\operatorname{argmin}_{\beta_i} \left( \sum_{j=1}^{m_i}[1 - y_{ij}(\beta_i^\top x_{ij} + \beta_0)]_+ + (\rho/2)\|\beta_i - z_k + (u_i)_k\|_2^2 \right)$

$z_{k+1} \leftarrow \operatorname{argmin}_z \left( \frac{1}{2C}\|z\|_2^2 + \frac{N\rho}{2}\|z\|_2^2 - \overline{\beta}_{k+1} - \overline{u}_k \right)$

$(u_i)_{k+1} \leftarrow (u_i)_k + (\beta_i)_k - z_{k+1}$

**end**

---

This can be simplified further by solving the $z$-update analytically. Rewrite the right-hand side as

$$h(z) = \frac{1}{2C} z^\top I z + (z - \overline{\beta}_{k+1} - \overline{u}_k)^\top I (z - \overline{\beta}_{k+1} - \overline{u}_k)$$

Then, solve

$$\begin{aligned} 0 &= \nabla_z h(z) \\ 0 &= z^\top \left( \frac{1}{c} + N\rho \right) + N\rho \left( -\overline{\beta}_{k+1} - \overline{u}_k \right) \\ z &= \frac{N\rho}{1/C + N\rho} \left( \overline{\beta}_{k+1} + \overline{u}_k \right) \end{aligned}$$

This yields the final form of the ADMM algorithm for this problem,

---

**Algorithm 5:** Global variable consensus ADMM for SVC

---

**for** $k = 1, 2, \ldots$ **do**

$(\beta_i)_{k+1} \leftarrow$

$$\operatorname{argmin}_{\beta_i} \left( \sum_{j=1}^{m_i} [1 - y_{ij}(\beta_i^\top x_{ij} + \beta_0)]_+ + (\rho/2)\|\beta_i - z_k + (u_i)_k\|_2^2 \right)$$

$$z_{k+1} \leftarrow \frac{N\rho}{1/C + N\rho}(\overline{\beta}_{k+1} + \overline{u}_k) \; (u_i)_{k+1} \leftarrow (u_i)_k + (\beta_i)_k - z_{k+1}$$

**end**

---

As in the general case, the $u_i$- and $\beta_i$-updates are parallelizable. Notice that the minimization problem in the $\beta_i$ subproblems step resembles formulation (9) of the SVC. Indeed, one can treat this this as a modified SVM problem and use existing single-process solvers [1].
This pattern of subproblems having the same form as the original problem appears frequently in applications of ADMM. In this sense, ADMM can be seen as a technique to extend nonparallel methods to large-scale problems [1].

## References

[1] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, “Distributed optimization and statistical learning via the alternating direction method of multipliers,” *Found. Trends Mach. Learn.*, vol. 3, p. 1–122, Jan. 2011.

[2] A. Nedic and A. Ozdaglar, “Cooperative distributed multi-agent optimization,” *Convex optimization in signal processing and communications*, vol. 340, 2010.

[3] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* Springer Science & Business Media, 2009.

[4] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, “Liblinear: A library for large linear classification,” *Journal of machine learning research*, vol. 9, no. Aug, pp. 1871–1874, 2008.

[5] C.-C. Chang and C.-J. Lin, "Libsvm: A library for support vector machines," *ACM transactions on intelligent systems and technology (TIST)*, vol. 2, no. 3, pp. 1–27, 2011.

[6] P. A. Forero, A. Cano, and G. B. Giannakis, "Consensus-based distributed support vector machines," *Journal of Machine Learning Research*, vol. 11, no. 5, 2010.