Machine Learning Journal manuscript No.
(will be inserted by the editor)

Dual Coordinate Descent Methods for Logistic Regression and Maximum Entropy Models

Hsiang-Fu Yu · Fang-Lan Huang · Chih-Jen Lin

Received: date / Accepted: date

Abstract Most optimization methods for logistic regression or maximum entropy solve the primal problem. They range from iterative scaling, coordinate descent, quasi-Newton, and truncated Newton. Less efforts have been made to solve the dual problem. In contrast, for linear support vector machines (SVM), methods have been shown to be very effective for solving the dual problem. In this paper, we apply coordinate descent methods to solve the dual form of logistic regression and maximum entropy. Interestingly, many details are different from the situation in linear SVM. We carefully study the theoretical convergence as well as numerical issues. The proposed method is shown to be faster than most state of the art methods for training logistic regression and maximum entropy.

1 Introduction

Logistic regression (LR) is useful in many areas such as document classification and natural language processing (NLP). It models the conditional probability as:

$$\mathcal{P}_{\boldsymbol{w}}(y = \pm 1|\boldsymbol{x}) \equiv \frac{1}{1 + e^{-y\boldsymbol{w}^T\boldsymbol{x}}},$$

where $\boldsymbol{x}$ is the data, y is the class label, and $\boldsymbol{w} \in R^n$ is the weight vector. Given two-class training data $\{\boldsymbol{x}_i, y_i\}_{i=1}^l, \boldsymbol{x}_i \in R^n, y_i \in \{1, -1\}$, logistic regression minimizes the following regularized negative log-likelihood:

$$P^{\mathsf{LR}}(\boldsymbol{w}) = C\sum_{i=1}^{l} \log\left(1 + e^{-y_i\boldsymbol{w}^T\boldsymbol{x}_i}\right) + \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w}, \tag{1}$$

Department of Computer Science
National Taiwan University
Taipei 106, Taiwan
{B93107, D93011, CJLIN}@CSIE.NTU.EDU.TW

where $C > 0$ is a penalty parameter. Problem (1) is referred to as the primal form of logistic regression, as one may instead solve the following dual problem.

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & D^{\mathsf{LR}}(\boldsymbol{\alpha}) = \frac{1}{2}\boldsymbol{\alpha}^T Q \boldsymbol{\alpha} + \sum_{i:\alpha_i>0} \alpha_i \log \alpha_i + \sum_{i:\alpha_i<C} (C-\alpha_i)\log(C-\alpha_i) \\ \text{subject to} \quad & 0 \le \alpha_i \le C,\ i = 1, \ldots, l, \end{aligned} \tag{2}$$

where $Q_{ij} = y_i y_j \boldsymbol{x}_i^T \boldsymbol{x}_j\ \forall i, j$.[1] By defining $0 \log 0 = 0$, (2) becomes

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & D^{\mathsf{LR}}(\boldsymbol{\alpha}) = \frac{1}{2}\boldsymbol{\alpha}^T Q \boldsymbol{\alpha} + \sum_{i=1}^{l} \alpha_i \log \alpha_i + (C-\alpha_i)\log(C-\alpha_i) \\ \text{subject to} \quad & 0 \le \alpha_i \le C,\ i = 1, \ldots, l. \end{aligned} \tag{3}$$

We omit the derivation of the dual problem as later we show details for the more general maximum entropy model.

Numerous optimization methods have been applied to train logistic regression, as surveyed in, for example, Minka (2003). Most of them solve the primal problem. Darroch and Ratcliff (1972), Della Pietra et al. (1997), Goodman (2002), Jin et al. (2003) and many others have proposed iterative scaling methods. Huang et al. (2010) apply a coordinate descent approach. A quasi-Newton method is included in the comparison by Minka (2003). Komarek and Moore (2005) and Lin et al. (2008) propose truncated Newton techniques. Less efforts have been made to solve the dual problem. A few existing studies include Jaakkola and Haussler (1999) and Keerthi et al. (2005). These works, interested in kernel LR, consider the dual because of the easy embedding of kernels. In contrast to LR, for linear support vector machines (SVM),[2] optimization methods have been shown to be very effective for solving the dual problem (e.g., Hsieh et al., 2008). Note that LR is very related to linear SVM, which takes the following primal and dual forms:

$$\min_{\boldsymbol{w}} \quad P^{\mathsf{SVM}}(\boldsymbol{w}) = C \sum_{i=1}^{l} \max(1 - y_i \boldsymbol{w}^T \boldsymbol{x}_i, 0) + \frac{1}{2}\boldsymbol{w}^T \boldsymbol{w}, \tag{4}$$

and

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & D^{\mathsf{SVM}}(\boldsymbol{\alpha}) = \frac{1}{2}\boldsymbol{\alpha}^T Q \boldsymbol{\alpha} - \sum_{i=1}^{l} \alpha_i \\ \text{subject to} \quad & 0 \le \alpha_i \le C, \forall i, \end{aligned}$$

where $Q_{ij} = y_i y_j \boldsymbol{x}_i^T \boldsymbol{x}_j$. An overview of dual forms of various regularized classifiers including LR and SVM can be found in Zhang (2002) though it does not explore detailed optimization algorithms for each classifier.

Coordinate descent methods, a classic optimization approach, have been very successfully applied to solve the dual form of large linear SVM (Hsieh et al., 2008). Motivated by their work, in this paper we study if coordinate descent methods are useful for solving the dual problem of LR. Interestingly, we find that many details are different from the situation in support vector machines. In particular, numerical issues due to

[1] In this work we do not consider kernel LR.

[2] By linear SVM we mean that kernel tricks are not employed.

logarithmic evaluations must be properly handled. We carefully design a coordinate descent algorithm to avoid numerical difficulties and prove the convergence. The proposed method is shown to be faster than most state of the art methods for training logistic regression.

Maximum Entropy (ME) is a generalization of logistic regression for multi-class scenarios.[3] Thus we also study a coordinate descent method for the dual form of ME. ME models the conditional probability as:

$$\mathcal{P}_{\boldsymbol{w}}(y|x) \equiv \frac{\exp(\boldsymbol{w}^T \boldsymbol{f}(x,y))}{\sum_{y'} \exp(\boldsymbol{w}^T \boldsymbol{f}(x,y'))},$$

where x denotes a context, y is the label of the context, and $\boldsymbol{w} \in R^n$ is the weight vector. A function vector $\boldsymbol{f}(x,y) \in R^n$ indicates features extracted from the context x and the label y. Assume N training samples $\{(x,y)\}$ are given, and we have grouped x's to l unique contexts $\{x_i\}$ and calculate the empirical probability distribution $\tilde{\mathcal{P}}(x_i,y) = N_{x_i,y}/N$, where $N_{x_i,y}$ is the number of times that (x_i,y) occurs in the training data. ME minimizes the following regularized negative log-likelihood:

$$\begin{aligned} \min_{\boldsymbol{w}} \quad P^{\mathsf{ME}}(\boldsymbol{w}) &= -\sum_{i=1}^{l}\sum_{y} \tilde{\mathcal{P}}(x_i,y) \log \mathcal{P}_{\boldsymbol{w}}(y|x_i) + \frac{1}{2\sigma^2}\boldsymbol{w}^T\boldsymbol{w} \\ &= \sum_{i=1}^{l} \tilde{\mathcal{P}}(x_i) \log\left(\sum_{y} \exp(\boldsymbol{w}^T \boldsymbol{f}(x_i,y))\right) - \boldsymbol{w}^T\tilde{\boldsymbol{f}} + \frac{1}{2\sigma^2}\boldsymbol{w}^T\boldsymbol{w}, \end{aligned} \tag{5}$$

where σ is the penalty parameter similar to C in (1), $\tilde{\mathcal{P}}(x_i) = \sum_y \tilde{\mathcal{P}}(x_i,y)$ is the marginal probability of x_i, and

$$\tilde{\boldsymbol{f}} = \sum_{i=1}^{l}\sum_{y} \tilde{\mathcal{P}}(x_i,y)\boldsymbol{f}(x_i,y) \tag{6}$$

is the expected vector of $\boldsymbol{f}(x_i,y)$. For convenience, we assume that

$$y_i \in Y \equiv \{1,2,\ldots,|Y|\}.$$

Many optimization methods have been applied to train ME, as discussed in Malouf (2002), Gao et al. (2007), Huang et al. (2010) and references therein. Most existing methods solve the primal problem, though there are a few exceptions: Memisevic (2006) applies a two-level coordinate descent method. Collins et al. (2008) propose an exponentiated gradient (EG) algorithm for conditional random fields (CRF) and their methods can be modified for dual ME. In this paper, we extend the two-level coordinate descent method (Memisevic, 2006) to a numerically robust algorithm. Moreover, we carefully study the theoretical convergence.

This paper is organized as follows. In Section 2, we discuss basic concepts of coordinate descent methods and show some existing examples for SVM and primal LR. In Sections 3 and 4, we describe our proposed algorithms for LR and ME, respectively. A related optimization method for LR/ME duals is discussed in Section 5. In Section 6, we compare our method with state of the art implementations. Results show that the new methods are more efficient. We conclude our work in Section 7.

[3] See the derivation in Section 6.1 of Huang et al. (2010). If $\boldsymbol{x}_i \in R^n, \forall i$, are training instances, then in ME, $\boldsymbol{w} \in R^{n|Y|}$. LR formulation in (1) is a simplified form because its $\boldsymbol{w}$ has n instead of $2n$ elements.

2 Coordinate Descent Methods

This section gives an overview of coordinate descent methods by considering the following optimization problem with linear constraints:

$$\begin{aligned} \min_{\boldsymbol{\alpha}\in R^l} \quad & F(\boldsymbol{\alpha}) \\ \text{subject to} \quad & A\boldsymbol{\alpha} = \boldsymbol{b}, \quad \text{and} \quad \mathbf{0} \le \boldsymbol{\alpha} \le C\boldsymbol{e}, \end{aligned} \tag{7}$$

where $A \in R^{m\times l}$, $\boldsymbol{b} \in R^m$, $0 < C \le \infty$ and $\boldsymbol{e} \in R^l$ is the vector of all ones. Coordinate descent methods iteratively update a block of variables because optimizing all variables together is more difficult. At each iteration, a nonempty subset $B \subset \{1,\dots,l\}$ is chosen to construct the following sub-problem.

$$\begin{aligned} \min_{\boldsymbol{z}} \quad & F(\boldsymbol{\alpha}+\boldsymbol{z}) \\ \text{subject to} \quad & z_i = 0, \forall i \notin B, \\ & A\boldsymbol{z} = \mathbf{0}, \quad \text{and} \quad 0 \le \alpha_i + z_i \le C, \forall i \in B. \end{aligned} \tag{8}$$

That is, we consider changing $\boldsymbol{\alpha}_B$ using the solution of (8), while fixing all other elements.

The two design considerations for coordinate descent methods are how to select a block B and how to solve the sub-problem (8). We take SVM and primal LR as examples and discuss different situations.

2.1 Exactly Solving One-Variable Sub-Problem

If the sub-problem has a closed-form solution, we can exactly solve it without using optimization software. We discuss Hsieh et al. (2008) for dual SVM as an example. They restrict B to contain only one element and sequentially select an element from $\{1,\dots,l\}$. If α_i is being updated, the one-variable sub-problem is

$$\begin{aligned} \min_{z} \quad & D^{\mathsf{SVM}}(\alpha_1,\dots,\alpha_i+z,\dots,\alpha_l) \\ & = \frac{1}{2}Q_{ii}z^2 + \nabla_i D^{\mathsf{SVM}}(\boldsymbol{\alpha})z + \text{constant}, \\ \text{subject to} \quad & 0 \le \alpha_i + z \le C, \end{aligned} \tag{9}$$

where $\nabla_i D^{\mathsf{SVM}}(\boldsymbol{\alpha})$ is the ith component of the gradient. As (9) is a quadratic function of z, if $Q_{ii} > 0$, easily the solution is:

$$z = \min\left(\max\left(\alpha_i - \frac{\nabla_i D^{\mathsf{SVM}}(\boldsymbol{\alpha})}{Q_{ii}}, 0\right), C\right) - \alpha_i. \tag{10}$$

We need to calculate:

$$\nabla_i D^{\mathsf{SVM}}(\boldsymbol{\alpha}) = (Q\boldsymbol{\alpha})_i - 1 = \sum_{j=1}^{l} Q_{ij}\alpha_j - 1, \tag{11}$$

which costs $O(ln)$ for calculating the ith row of the matrix $Q\boldsymbol{\alpha}$. Such operations are expensive. Hsieh et al. (2008) propose an efficient way of $O(n)$ to calculate (11). This

technique is applied to our method for logistic regression. We will have a detailed discussion in Section 3. Algorithm 1 summarizes Hsieh et al. (2008)'s procedure.

In practice, for every round of going through l variables, Hsieh et al. (2008) randomly permute l indices to decide the order for update. They report this setting yields better convergence than sequential updates. In all coordinate descent methods we will discuss, this technique can be applied.

2.2 Approximately Solving One-Variable Sub-Problem

If the sub-problem does not have a closed-form solution, optimization methods must be used to solve the sub-problem. We show the work by Huang et al. (2010) as an example. They apply a one-variable coordinate descent method to solve the primal form of LR. If w_j is being updated, the sub-problem minimizes

$$\begin{aligned} g(z) &= P^{\mathsf{LR}}(\boldsymbol{w} + z\boldsymbol{e}_j) \\ &= \frac{z^2}{2} + zw_j + C\left(\sum_{i=1}^{l} \log(1 + \frac{e^{zx_{ij}} - 1}{1 + e^{-\boldsymbol{w}^T\boldsymbol{x}_i}}) - z\sum_{i=1,y_i=1}^{l} x_{ij}\right) + P^{\mathsf{LR}}(\boldsymbol{w}), \end{aligned} \tag{12}$$

where $\boldsymbol{e}_j$ is the indicator vector for the jth element. This sub-problem does not have a closed-form solution, so Huang et al. (2010) consider the Newton method with the following update rule:

$$z \leftarrow z - g'(z)/g''(z).$$

The first and second derivatives of $g(z)$ are respectively:

$$g'(z) = w_j + z + C\left(\sum_{i=1}^{l} \frac{x_{ij}e^{zx_{ij}}}{e^{zx_{ij}} + e^{-\boldsymbol{w}^T\boldsymbol{x}_i}} - \sum_{i=1,y_i=1}^{l} x_{ij}\right), \tag{13}$$

and

$$g''(z) = 1 + C\left(\sum_{i=1}^{l} \frac{x_{ij}^2 e^{-\boldsymbol{w}^T\boldsymbol{x}_i} e^{zx_{ij}}}{(e^{zx_{ij}} + e^{-\boldsymbol{w}^T\boldsymbol{x}_i})^2}\right). \tag{14}$$

If $\boldsymbol{w}^T\boldsymbol{x}_i$ is available, (13) and (14) cost $O(l)$. In particular, there are l exponential operations, each of which is much more expensive than a multiplication or a division on most computers. As each Newton update is not cheap, Huang et al. (2010) apply only one update and obtain an approximate solution of the sub-problem (12). They also need a line search procedure to guarantee the convergence.

Compared to Section 2.1, clearly the situation is more complicated if the sub-problem does not have a closed-form solution.

2.3 Constrained Problems and Using More Than One Variable

Examples in Sections 2.1 and 2.2 choose one variable at a time, so the sub-problem is simple. Instead, we can choose more than one variable. This is particularly needed for constrained problems as a one-variable update may fail to change the solution (i.e., $z = 0$ is optimal for the sub-problem). We show several examples in this section.

For most classical SVM software, they solve SVM with a bias term b. That is, $\boldsymbol{w}^T\boldsymbol{x}_i$ in (4) is replaced by $\boldsymbol{w}^T\boldsymbol{x}_i + b$. The dual problem then contains an equality constraint:

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & \frac{1}{2}\boldsymbol{\alpha}^T Q\boldsymbol{\alpha} - \sum_{i=1}^{l}\alpha_i \\ \text{subject to} \quad & \sum_{i=1}^{l} y_i\alpha_i = 0, \quad \text{and} \quad 0 \le \alpha_i \le C,\ \forall i. \end{aligned} \tag{15}$$

Due to the equality constraint, the sub-problem must contain at least two variables.

Another example needing more than one variable per sub-problem is multi-lass SVM. Assume there are $|Y|$ classes. Then $y_i \in \{1, \dots, |Y|\}$ instead of $\{1, -1\}$. We discuss the multi-class SVM approach by Crammer and Singer (2000) because its formulation is related to maximum entropy discussed later. The dual problem is

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & D^{\mathsf{CS}}(\boldsymbol{\alpha}) = \frac{1}{2}\sum_{y=1}^{|Y|}\sum_{i=1}^{l}\sum_{j=1}^{l}\alpha_{iy}\alpha_{jy}\boldsymbol{x}_i^T\boldsymbol{x}_j + \sum_{i=1}^{l}\sum_{y=1,y\neq y_i}^{|Y|}\alpha_{iy} \\ \text{subject to} \quad & \sum_{y=1}^{|Y|}\alpha_{iy} = 0,\ \forall i = 1, \dots, l, \quad \text{and} \\ & \alpha_{iy} \le C^y_{y_i},\ \forall i = 1, \dots, l,\ y = 1, \dots, |Y|, \end{aligned} \tag{16}$$

where

$$\boldsymbol{\alpha} = [\alpha_{11}, \dots, \alpha_{1|Y|}, \dots, \alpha_{l1}, \dots, \alpha_{l|Y|}]^T, \quad \text{and} \quad C^y_{y_i} = \begin{cases} 0 & \text{if } y_i \neq y, \\ C & \text{if } y_i = y. \end{cases}$$

The optimization problem (16) has $|Y|l$ variables. The l equalities imply that several variables must be chosen for a sub-problem. As each equality involves variables associated with an instance, Crammer and Singer (2000) decompose $\boldsymbol{\alpha}$ to l blocks with $\bar{\boldsymbol{\alpha}}_i = [\alpha_{i1}, \dots, \alpha_{i|Y|}]^T,\ i = 1, \dots, l$, and update one block at a time. The sub-problem is

$$\begin{aligned} \min_{\boldsymbol{z}} \quad & D^{\mathsf{CS}}(\bar{\boldsymbol{\alpha}}_1, \dots, \bar{\boldsymbol{\alpha}}_i + \boldsymbol{z}, \dots, \bar{\boldsymbol{\alpha}}_l) \\ & = \frac{1}{2}\sum_{y=1}^{|Y|}\boldsymbol{x}_i^T\boldsymbol{x}_i z_y^2 + \sum_{y=1}^{|Y|}\nabla_{iy}D^{\mathsf{CS}}(\boldsymbol{\alpha})z_y + \text{constant}, \\ \text{subject to} \quad & \sum_{y} z_y = 0 \quad \text{and} \quad 0 \le \alpha_{iy} + z_y \le C^y_{y_i},\ \forall y = 1, \dots, |Y|, \end{aligned} \tag{17}$$

where $\nabla_{iy}D^{\mathsf{CS}}(\boldsymbol{\alpha})$ is the partial derivative with respect to α_{iy}. Crammer and Singer (2000, Section 6) show that a closed-form solution of this sub-problem can be obtained in $O(|Y|\log|Y|)$ time. Alternatively, we can apply general optimization methods.

Algorithm 1 Dual coordinate descent method for linear SVM

1. Given initial $\boldsymbol{\alpha} \in [0, C]^l$.
2. While $\boldsymbol{\alpha}$ is not optimal
   - Choose an index i from $\{1, \ldots, l\}$.
   - Solve the sub-problem (9) exactly by the analytic form (10).
   - Update α_i .

Algorithm 2 Dual coordinate descent method for logistic regression

1. Given initial $\boldsymbol{\alpha} \in (0, C)^l$.
2. While $\boldsymbol{\alpha}$ is not optimal
   - Choose an index i from $\{1, \ldots, l\}$.
   - Solve the sub-problem (18) exactly or approximately.
   - Update α_i.

3 A Dual Coordinate Descent Method for Logistic Regression

We begin with discussing difficulties for applying coordinate descent methods for LR. Next we devise an effective method to solve the sub-problem and present our overall procedure. Earlier studies employing coordinate descent methods for dual LR include Minka (2003, Section 9) and Keerthi et al. (2005). We also discuss the differences between ours and their works.

3.1 Issues in Applying Coordinate Descent Methods for Logistic Regression

Since the dual form for LR is very close to SVM dual, naturally we try to extend existing methods for SVM (e.g., Algorithm 1). In the following we check if each step of Algorithm 1 is applicable to LR.

To give an initial $\boldsymbol{\alpha}$, Algorithm 1 allows any point in a closed interval $[0, C]^l$ and one often uses $\boldsymbol{\alpha} = \mathbf{0}$ due to the sparsity at the SVM dual optimal solution. However, for dual LR the objective function is not well defined at $\alpha_i = 0$ or $\alpha_i = C$. Therefore, an initial $\boldsymbol{\alpha}$ must be in an open interval $(0, C)^l$. Further, as $\lim_{\alpha_i \to 0^+} \alpha_i \log \alpha_i = 0$, it is unclear if an optimal solution occurs at $\alpha_i = 0$ or C. The following theorem shows that (3) attains a unique minimum in $(0, C)^l$:

Theorem 1 *The* LR *dual problem* (3) *attains a unique optimal solution* $\boldsymbol{\alpha}^*$ *and* $\boldsymbol{\alpha}^* \in (0, C)^l$.

The proof is in Appendix A.2. In Section 3.4, we discuss how to choose an appropriate initial point in $(0, C)^l$.

Another important difference from SVM is that the sub-problem no longer has a closed-form solution. If the ith variable is selected, the sub-problem is

$$\begin{aligned} \min_z \quad & g(z) \equiv (c_1 + z)\log(c_1 + z) + (c_2 - z)\log(c_2 - z) + \frac{a}{2}z^2 + bz \\ \text{subject to} \quad & -c_1 \le z \le c_2, \end{aligned} \tag{18}$$

where

$$c_1 = \alpha_i, c_2 = C - \alpha_i, a = Q_{ii}, \text{ and } b = (Q\boldsymbol{\alpha})_i.$$

Table 1: Cost of operations at a Newton iteration.

Operation	Cost
Constructing the sub-problem	$O(n)$
Finding Newton direction d	$O(1)$
Calculating $g(z^k + \lambda d)$ in line search	$O(1)$

This sub-problem has been studied in, for example, Keerthi et al. (2005) and Memisevic (2006).[4] We will discuss the difference between our approach and theirs.

If using Newton methods to solve (18), the update rule without considering the constraint $-c_1 \le z \le c_2$ is

$$z^{k+1} = z^k + d, \quad d = -\frac{g'(z^k)}{g''(z^k)}, \tag{19}$$

where k is the index of iterations and $\forall z \in (-c_1, c_2)$

$$g'(z) = az + b + \log\frac{c_1 + z}{c_2 - z}, \text{ and } \quad g''(z) = a + \frac{c_1 + c_2}{(c_1 + z)(c_2 - z)}. \tag{20}$$

To ensure the convergence of Newton methods, we often need a line search procedure to check the sufficient decrease of function values. For example, we may search for the first $\lambda = 1, \beta, \beta^2, \ldots$, such that

$$g(z^k + \lambda d) - g(z^k) \le \gamma\lambda g'(z^k)d, \tag{21}$$

where $\gamma, \beta \in (0, 1)$. In Keerthi et al. (2005), they suggest a combination of Newton and bisection methods to ensure the convergence, but details are not given. We give an implementation in Section 6.3 and compare it with our proposed method.

We can apply many or few Newton iterations to accurately or loosely solve the sub-problem, respectively. The decision relies on analyzing the cost per iteration; see Table 1. In the beginning, we must construct the sub-problem by calculating coefficients in (18). Since Q_{ii} can be pre-stored, the main cost is $O(nl)$ for calculating $(Q\boldsymbol{\alpha})_i$. The same operation is needed for SVM; see (11). To reduce the cost, we adopt a commonly used trick in linear SVM (e.g., Hsieh et al., 2008) by maintaining a vector:

$$\boldsymbol{w}(\boldsymbol{\alpha}) \equiv \sum_{i=1}^{l} y_i\alpha_i\boldsymbol{x}_i. \tag{22}$$

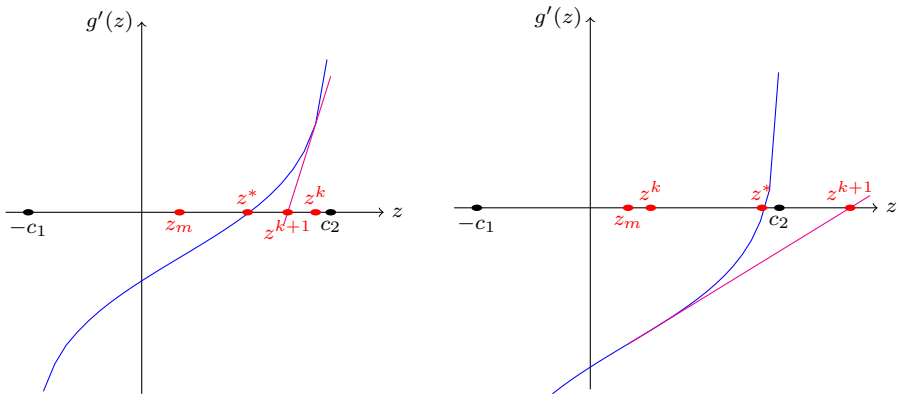
Then the cost is reduced to $O(n)$:

$$(Q\boldsymbol{\alpha})_i = \sum_{j=1}^{l} y_i y_j \alpha_j \boldsymbol{x}_j^T\boldsymbol{x}_i = y_i(\sum_{j=1}^{l} y_j\alpha_j\boldsymbol{x}_j^T)\boldsymbol{x}_i = y_i\boldsymbol{w}(\boldsymbol{\alpha})^T\boldsymbol{x}_i. \tag{23}$$

To apply (23), $\boldsymbol{w}(\boldsymbol{\alpha})$ should be maintained throughout the procedure. By

$$\boldsymbol{w}(\boldsymbol{\alpha} + z\boldsymbol{e}_i) = \boldsymbol{w}(\boldsymbol{\alpha}) + zy_i\boldsymbol{x}_i, \tag{24}$$

[4] Their sub-problem, though in the same form as (18), is from solving maximum entropy instead of logistic regression. See more discussion in Section 4.



(a) $z^k > z^*$ (b) $z^k < z^*$ and $z^k + d > c_2$

Fig. 1: Newton steps for finding a root of $g'(z)$. z^k is an initial point, z^{k+1} is derived from z^k by the Newton step, z^* is the optimizer, and $z_m \equiv (c_2 - c_1)/2$ is the mid-point of $(-c_1, c_2)$. Figure 1(a) shows that Newton step works fine with a good starting point. Figure 1(b) shows the situation that Newton step $z^k + d$ walks outside the interior.

where z is the solution of the sub-problem (18) and $\boldsymbol{e}_i$ is the indicator vector for the ith component, $\boldsymbol{w}(\boldsymbol{\alpha})$ can be maintained in $O(n)$ time. Hence constructing the sub-problem costs $O(n)$. From Table 1, the complexity of solving the sub-problem is

$$O(n) + \#\text{Newton steps} \times (O(1) + O(1) \times (\#\text{Line search steps}))\,. \tag{25}$$

Because of the cheap $O(1)$ cost for finding Newton directions and conducting line search, we should accurately solve the sub-problem. Interestingly, the situation is very different for solving primal LR via coordinate descent methods (Section 2.2). The sub-problem (12) does not have a closed-form solution either, but Huang et al. (2010) conduct only one Newton iteration (with line search). The reason is that both finding Newton directions and conducting line searches are expensive.

From (20), the time for calculating d is dominated by the log operation, which is much more expensive than addition and multiplication operations. In the line search procedure, calculating one function value $g(z + \lambda d)$ involves two log operations; see (18). Hence line search is more expensive than finding the Newton direction. In Section 3.2, we propose a modified Newton method so that line search is not needed but the convergence still holds. Moreover, our approach will take the constraint $-c_1 \le z \le c_2$ into consideration.

The discussion so far indicates that while LR dual is very close to SVM dual, many details in applying coordinate descent methods are different.

3.2 A Modified Newton Method for Solving the Sub-problem

We propose a modified Newton method for (18) without needing line search procedures. Besides, we properly handle the inequality constraint and establish the global

convergence. To begin, we follow Theorem 2 to show that the optimum of (18) is in the open interval $(-c_1, c_2)$:

Theorem 2 *The sub-problem* (18) *has a unique minimum* z^*. *Moreover,* $z^* \in (-c_1, c_2)$ *and* $g'(z^*) = 0$.

The proof is in Appendix A.3. We draw Figure 1 to analyze how Newton updates (19) may find a root of $g'(z)$. By considering two different situations, we can draw some crucial observations:

- From Figure 1(a), if z^k is on the "correct" side of z^*, then not only subsequent points generated by (19) are in $(-c_1, c_2)$, but also the Newton method converges to z^*.
- From Figure 1(b), if z^k is on the "wrong" side of z^*, then z^{k+1} by (19) may be outside $(-c_1, c_2)$.

We need a mechanism so that eventually all points are on the "correct" side of z^*. To do so a good understanding of "correct" and "wrong" sides is needed. Let $z_m \equiv (c_2 - c_1)/2$ be the mid-point of the interval $(-c_1, c_2)$. From Figure 1, we can see that $g'(z)$ is concave in $(-c_1, z_m]$, and convex in $[z_m, c_2)$.[5] The following theorem shows that we can check the position of z^* and z_m to see if z^k is on the correct side:

Theorem 3 *Let* z^* *be the optimizer of* (18) *and* $z_m = (c_2 - c_1)/2$. *If* $z^* \geq z_m$, *then* $\{z^k\}$ *generated by* (19) *converges to* z^* *for any starting point in* $[z^*, c_2)$. *If* $z^* \leq z_m$, *then* $\{z^k\}$ *converges to* z^* *for any starting point in* $(-c_1, z^*]$. *For any* z^k *satisfying these conditions, we say it is on the "correct" side of* z^*.

This theorem can be easily obtained by the standard convergence proof of Newton methods.[6]

For any z^k on the "wrong" side, there are two cases. The first one is $z^k + d \in (-c_1, c_2)$. If $z^k + d$ falls on the "correct" side, Theorem 3 implies that subsequent Newton updates converge. If $z^k + d$ is still on the "wrong" side, it at least gets closer to z^*. Thus we take $z^k + d$ as z^{k+1}. The second case is that $z^k + d \notin (-c_1, c_2)$. Because $(-c_1, c_2)$ is an open interval, it is not possible to do a direct projection. Assume $z^k + d \geq c_2$ as Figure 1(b). We propose finding a point z in $[z^k, c_2)$ closer to the "correct" side by

$$z^{k+1} = \xi z^k + (1 - \xi)c_2, \tag{26}$$

where $\xi \in (0, 1)$. For any z^k on the "wrong" side, we prove that the above setting eventually reaches a point on the "correct" side. Then this point can be considered as a starting point in Theorem 3 for the convergence.

Theorem 4 *Assume* $z^* \geq z_m$. *If we generate a sequence of Newton iterations by starting from* $z^k < z^*$ *(i.e.,* z^k *on the "wrong" side of* z^**), and applying the update rule:*

$$z^{k+1} = \begin{cases} z^k + d & \text{if } z^k + d < c_2, \\ \xi z^k + (1 - \xi)c_2 & \text{if } z^k + d \geq c_2, \end{cases}$$

then there is $k' > k$ *such that* $z^{k'} \geq z^*$. *That is,* $z^{k'}$ *is on the "correct" side. The situation for* $z^* \leq z_m$ *and* $z^k > z^*$ *is similar.*

[5] Formally, we can prove $g'''(z_m) = 0$, $g'''(z) > 0$ if $z > z_m$, and $g'''(z) < 0$ if $z < z_m$.

[6] For example, `http://planetmath.org/encyclopedia/NewtonsMethodWorksForConvexRealFunctions.html`

Algorithm 3 A modified Newton method to solve (18)

- Given coefficients: a, b, c_1, and c_2.
- Set initial $z^0 \in (-c_1, c_2)$.
- For $k = 0, 1 \dots$
  - If $g'(z^k) = 0$, break.
  - $d \leftarrow -g'(z^k)/g''(z^k)$.
  -
$$z^{k+1} = \begin{cases} z^k + d & \text{if } z^k + d \in (-c_1, c_2), \\ \xi z^k + (1-\xi)(-c_1) & \text{if } z^k + d \le -c_1, \\ \xi z^k + (1-\xi)c_2 & \text{if } z^k + d \ge c_2. \end{cases} \tag{30}$$

The proof is in Appendix A.4.

We describe the modified Newton method in Algorithm 3. The update rule is very simple and no line search is needed. Combining Theorems 3 and 4, the global convergence of Algorithm 3 is established.

Theorem 5 *The sequence* $\{z^k\}$ *generated by Algorithm 3 converges to the optimum* z^* *of* (18) *for any* $z^0 \in (-c_1, c_2)$.

The initial z^0 can be any value in $(-c_1, c_2)$, but we hope it is close to z^* for fast convergence of the Newton method. In the final stage of the decomposition method, α_i does not change much and $z^* \approx 0$, so $z^0 = 0$ is a reasonable choice. However, in the early stage of the decomposition method, this z^0 may be far away from z^*. While we cannot easily find a z^0 on the "correct" side, Theorem 3 indicates that z^0 should satisfy

$$z^0 \in \begin{cases} (-c_1, z_m) & \text{if } z^* \le z_m, \\ [z_m, c_2) & \text{if } z^* \ge z_m. \end{cases} \tag{27}$$

Later in Section 3.3 we show an easy way to check if $z^k \le z_m$ or not; see (34). Thus we use $z^0 = 0$ in general, but also ensure that z^0 satisfies (27). This is achieved by

$$z^0 = \begin{cases} (1-\xi_0)(-c_1) & \text{if } z^* \le z_m \le 0, \\ (1-\xi_0)(c_2) & \text{if } z^* \ge z_m \ge 0, \\ 0 & \text{otherwise.} \end{cases} \tag{28}$$

We explain that $0 < \xi_0 \le 0.5$ will let z^0 satisfy (27). If

$$-c_1 < z^* \le z_m \le 0 < c_2, \tag{29}$$

then $-(1-\xi_0)c_1 \in (-c_1, 0)$ and is closer to $-c_1$. Since z_m is the mid-point of $(-c_1, c_2)$, we have $-(1-\xi_0)c_1 \le z_m$. The situation for $z^* \ge z_m \ge 0$ is similar.

3.3 Numerical Difficulties

Unfortunately, a direct implementation of Algorithm 3 may face numerical difficulties. Keerthi et al. (2005) point out that when α_i is close to 0 or C, it may be difficult to reach a solution z^* satisfying

$$g'(z^*) = Q_{ii}z^* + (Q\boldsymbol{\alpha})_i + \log(\alpha_i + z^*) - \log(C - \alpha_i - z^*) \approx 0.$$

They explain that if C is large (say 10^5), $(Q\boldsymbol{\alpha})_i$ is large as well. Then $\alpha_i + z^*$ may be too small (e.g., e^{-10^5}) to be represented as a floating-point number. They propose some ways to handle such a situation. However, through experiments we find that even if C is as large as 10^5, $(Q\boldsymbol{\alpha})_i$ is generally much smaller (e.g., a few hundreds or thousands). The reason seems to be that from (23), $(Q\boldsymbol{\alpha})_i$ is the sum of positive and negative terms, so the value is not as large as α_i. Instead, we find that numerical difficulties occur because of catastrophic cancellations (i.e., subtraction between two nearly-equal floating-point numbers) when $\alpha_i + z$ is close to zero. That is, if $z \approx -\alpha_i$, the relative numerical error of calculating $\alpha_i + z$ can be large (Goldberg, 1991). Then $\log(\alpha_i + z)$ is erroneous. A common solution to avoid catastrophic cancellation is by some reformulations.

Let $Z_1 = c_1 + z$ and $s = c_1 + c_2$. An equivalent form to (18) is

$$\begin{aligned} \min_{Z_1} \quad & g_1(Z_1) = Z_1 \log Z_1 + (s - Z_1)\log(s - Z_1) + \frac{a}{2}(Z_1 - c_1)^2 + b_1(Z_1 - c_1) \\ \text{subject to} \quad & 0 \le Z_1 \le s, b_1 = b. \end{aligned}$$

Clearly, when $z \approx -c_1$,

$$s - Z_1 = c_2 - z \approx c_2 + c_1 = s \tag{31}$$

is far away from zero. Thus we avoid a catastrophic cancellation. However, a new subtraction $Z_1 - c_1$ occurs. In calculating the Newton direction, $Z_1 - c_1$ appears only in $g_1'(Z_1)$; see (32). If $Z_1 - c_1 \approx 0$, then $a(Z_1 - c_1) + b_1 \approx b_1$ and the large relative error in calculating $Z_1 - c_1$ does not cause serious problems.

Similarly, if $z \approx c_2$, we let $Z_2 = c_2 - z$ and adopt the following reformulation.

$$\begin{aligned} \min_{Z_2} \quad & g_2(Z_2) = Z_2 \log Z_2 + (s - Z_2)\log(s - Z_2) + \frac{a}{2}(Z_2 - c_2)^2 + b_2(Z_2 - c_2) \\ \text{subject to} \quad & 0 \le Z_2 \le s, b_2 = -b. \end{aligned}$$

Therefore, instead of minimizing on z, we now work on the distance from z to the lower (or upper) bound. To minimize $g_t(Z_t),\ t = 1, 2$ by the Newton method, we need the first and the second derivatives:

$$g_t'(Z_t) = \log\frac{Z_t}{s - Z_t} + a(Z_t - c_t) + b_t \quad \text{and} \quad g_t''(Z_t) = a + \frac{s}{Z_t(s - Z_t)}. \tag{32}$$

Next we check if $g_1(Z_1)$ or $g_2(Z_2)$ should be used. From the above discussion, $g_1(Z_1)$ aims to handle the situation of $z \approx -c_1$, while $g_2(Z_2)$ is for $z \approx c_2$. As $\{z^k\}$ generated by Algorithm 3 converges to z^*, most of the points in $\{z^k\}$ are close to z^*. Hence we can choose $g_1(Z_1)$ or $g_2(Z_2)$ based on z^*'s closeness to the two bounds:

$$z^* \text{ closer to } \begin{cases} -c_1 \\ c_2 \end{cases} \quad \Rightarrow \quad \text{choose } \begin{cases} g_1(Z_1), \\ g_2(Z_2). \end{cases} \tag{33}$$

To use (33), as z^* is unknown before applying the Newton method, we consider the following property:

$$z^* \text{ closer to } \begin{cases} -c_1 \\ c_2 \end{cases} \Leftrightarrow z^* \begin{cases} \le z_m \\ \ge z_m \end{cases} \Leftrightarrow g'(z_m) \begin{cases} \ge 0 \\ \le 0 \end{cases} \Leftrightarrow z_m \begin{cases} \ge -b/a, \\ \le -b/a. \end{cases} \tag{34}$$

Algorithm 4 A new modified Newton method for (18)

- Given coefficients: a, b, c_1, and c_2. Let $s = c_1 + c_2$
- $t \leftarrow \begin{cases} 1 & \text{if } z_m \geq \frac{-b}{a}, \\ 2 & \text{if } z_m < \frac{-b}{a}. \end{cases}$
- $Z_t^0 \in (0, s)$.
- For $k = 0, 1, \ldots$
  - If $g_t'(Z_t^k) = 0$, break.
  - $d \leftarrow -g_t'(Z_t^k)/g_t''(Z_t^k)$.
  - $Z_t^{k+1} = \begin{cases} \xi Z_t^k & \text{if } Z_t^k + d \leq 0, \\ Z_t^k + d & \text{otherwise.} \end{cases}$
- $\begin{cases} Z_2^k = s - Z_1^k & \text{if } t = 1, \\ Z_1^k = s - Z_2^k & \text{if } t = 2. \end{cases}$
- return (Z_1^k, Z_2^k).

The proof is in Appendix A.5. Thus the decision is by easily comparing z_m and $-b/a$.

Using $Z_1 = c_1 + z,\ Z_2 = c_2 - z$, and (34), a direct calculaton shows that the initial z^0 considered in (28) becomes

$$Z_1^0 = \begin{cases} \xi_0 c_1 & \text{if } c_1 \geq s/2, \\ c_1 & \text{otherwise,} \end{cases} \quad \text{and} \quad Z_2^0 = \begin{cases} \xi_0 c_2 & \text{if } c_2 \geq s/2, \\ c_2 & \text{otherwise.} \end{cases} \tag{35}$$

We also need to adjust (26), which handles the situation if $z^k + d > c_2$. Assume $g_2(Z_2)$ is used. Eq. (26) becomes

$$c_2 - Z_2^{k+1} = \xi(c_2 - Z_2^k) + (1 - \xi)c_2$$

and can be simplified to

$$Z_2^{k+1} = \xi Z_2^k.$$

The situation for $g_1(Z_1)$ is similar. By minimizing $g_1(Z_1)$ or $g_2(Z_2)$, Algorithm 3 becomes Algorithm 4. The returned values can be either (t, Z_t^k) or (Z_1^k, Z_2^k). We adopt the latter to avoid possible catastrophic cancellations in calculating c_1 and c_2 for the next sub-problem. See details in Section 3.4.

3.4 The Overall Procedure

Different from the situation in SVM, now $\boldsymbol{\alpha} = \mathbf{0}$ is not a valid starting point. A naive choice is to set $\alpha_i = C/2 \in (0, C),\ \forall i$. However, experiments show that this initialization is far away from the optimal solution. Note that for SVM, $\boldsymbol{\alpha} = \mathbf{0}$ is a reasonable choice because at the final solution many elements remain at zero (i.e., the solution is sparse). Though LR does not produce a sparse solution, we explain that many α_i values are small. From the optimality condition[7], the optimal $(\boldsymbol{w}, \boldsymbol{\alpha})$ satisfies

$$\alpha_i = \frac{C \exp(-y_i \boldsymbol{w}^T \boldsymbol{x}_i)}{1 + \exp(-y_i \boldsymbol{w}^T \boldsymbol{x}_i)},\ \forall i.$$

[7] We do not show the optimality condition for LR, but a similar form can be found in (75) for ME.

Algorithm 5 A dual coordinate descent method for logistic regression

- Set initial $\alpha_i = \min(\epsilon_1 C, \epsilon_2)\ \forall i$ and the corresponding $\boldsymbol{w} \leftarrow \sum_i \alpha_i y_i \boldsymbol{x}_i$.
- $\alpha'_i \leftarrow C - \alpha_i$ and $Q_{ii} \leftarrow \boldsymbol{x}_i^T \boldsymbol{x}_i\ \forall i$.
- While $\boldsymbol{\alpha}$ is not optimal

  For $i = 1, \ldots, l$

  1. Construct the sub-problem (18) for instance $\boldsymbol{x}_i$ by

  $$c_1 = \alpha_i, c_2 = \alpha'_i, a = Q_{ii}, \text{ and } b = y_i \boldsymbol{w}^T \boldsymbol{x}_i.$$

  2. Solve (18) by Algorithm 4 and get Z_1 and Z_2. Note that in Algorithm 4, $s \equiv c_1 + c_2 = C$.
  3. $\boldsymbol{w} \leftarrow \boldsymbol{w} + (Z_1 - \alpha_i) y_i \boldsymbol{x}_i$.
  4. $\alpha_i \leftarrow Z_1, \alpha'_i \leftarrow Z_2$.

As $\exp(-y_i \boldsymbol{w}^T \boldsymbol{x}_i)$ quickly decays to zero for negative $-y_i \boldsymbol{w}^T \boldsymbol{x}_i$, many correctly classified instances have their corresponding α_i/C close to zero. Therefore, similar to SVM, we should use an initial point close to the zero vector. We consider

$$\alpha_i = \min(\epsilon_1 C, \epsilon_2)\ \forall i, \tag{36}$$

where ϵ_1 and ϵ_2 are small positive values less than one. Keerthi et al. (2005) consider $\alpha_i = C/l^+$ if $y_i = 1$ and C/l^- if $y_i = -1$, where l^+ and l^- are the numbers of positive/negative data, respectively. Ours differs from them in ϵ_2, which ensures that the initial α_i is sufficiently small regardless of the C value.

In constructing the sub-problem (18), another catastrophic cancellation may occur. If $\alpha_i \approx C$, then calculating $c_2 = C - \alpha_i$ is a catastrophic cancellation. An erroneous c_2 then causes more numerical errors in subsequent calculations. To remedy this problem, a reformulation can be performed in the previous update of α_i: From the definition of Z_2 in Section 3.3,

$$Z_2 = c_2 - z = C - \alpha_i^{\text{old}} - z = C - \alpha_i^{\text{new}}.$$

Therefore, if earlier $g_2(Z_2)$ is considered, the returned Z_2 can be directly used as c_2 for the current sub-problem. Alternatively, if $g_1(Z_1)$ is used, we calculate $Z_2 = s - Z_1$ in the end of Algorithm 4. According to (31), this is not a catastrophic cancellation. The discussion here explains why we choose to output both (Z_1, Z_2) in Algorithm 4.

Algorithm 5 gives details of the proposed coordinate descent method for LR dual. To update $\boldsymbol{w}(\boldsymbol{\alpha})$ via (24), we need to obtain z, but Algorithm 4 gives only Z_1 and Z_2. We can consider either $Z_1 - \alpha_i$ or $C - Z_2$ though a catastrophic cancellation may occur. However, the situation seems to be less serious than that in Section 3.2, which involves a log operation after a catastrophic cancellation. Finally, the following theorem shows the linear convergence of Algorithm 5.

Theorem 6 *Let $\boldsymbol{\alpha}^s$ denote the vector in the beginning of each iteration in the while loop of Algorithm 5. The sequence $\{\boldsymbol{\alpha}^s\}$ globally converges to the unique optimum $\boldsymbol{\alpha}^*$. The convergence rate is at least linear: there are $0 < \mu < 1$ and an iteration s_0 such that*

$$D^{\mathsf{LR}}(\boldsymbol{\alpha}^{s+1}) - D^{\mathsf{LR}}(\boldsymbol{\alpha}^*) \le \mu (D^{\mathsf{LR}}(\boldsymbol{\alpha}^s) - D^{\mathsf{LR}}(\boldsymbol{\alpha}^*)), \forall s \ge s_0. \tag{37}$$

The proof is in Appendix A.6.

4 A Two-Level Dual Coordinate Descent Method for Maximum Entropy

Based on the experience for LR in Section 3, this section investigates a two-level dual coordinate descent method for ME. The outer level considers a block of variables at a time. The resulting sub-problem is then solved by an inner loop of coordinate descent updates. Our method extends that in Memisevic (2006), but we give more complete analysis.

4.1 Dual of ME and Coordinate Descent Methods

We derive in Appendix A.7 the following dual form for (5):

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & D^{\mathsf{ME}}(\boldsymbol{\alpha}) = \frac{1}{2\sigma^2}\boldsymbol{w}(\boldsymbol{\alpha})^T\boldsymbol{w}(\boldsymbol{\alpha}) + \sum_i \sum_{y:\alpha_{iy}>0} \alpha_{iy}\log\alpha_{iy} \\ \text{subject to} \quad & \sum_y \alpha_{iy} = \tilde{\mathcal{P}}(x_i) \quad \text{and} \quad \alpha_{iy} \geq 0 \ \forall i, y, \end{aligned} \tag{38}$$

where

$$\boldsymbol{w}(\boldsymbol{\alpha}) \equiv \sigma^2\left(\tilde{\boldsymbol{f}} - \sum_{i,y}\alpha_{iy}\boldsymbol{f}(x_i, y)\right) \tag{39}$$

and $\tilde{\boldsymbol{f}}$ is defined in (6). The vector $\boldsymbol{\alpha} \in R^{l|Y|}$ can be decomposed to l blocks

$$\boldsymbol{\alpha} = [\bar{\boldsymbol{\alpha}}_1, \ldots, \bar{\boldsymbol{\alpha}}_l]^T \text{ and } \bar{\boldsymbol{\alpha}}_i = [\alpha_{i1}, \ldots, \alpha_{i|Y|}]^T, \tag{40}$$

where $\bar{\boldsymbol{\alpha}}_i$ corresponds to the unique context x_i in the data set. If $\boldsymbol{w}^*$ and $\boldsymbol{\alpha}^*$ are respectively the optimal solution of primal and dual problems, then $\boldsymbol{w}(\boldsymbol{\alpha}^*) = \boldsymbol{w}^*$.

Eq. (39) is slightly different from the formulation considered in Lebanon and Lafferty (2002); Memisevic (2006); Collins et al. (2008), where

$$\boldsymbol{w}(\boldsymbol{\alpha}) \equiv \sigma^2 \sum_{i,y} \alpha_{iy}\left(\boldsymbol{f}(x_i, y_i) - \boldsymbol{f}(x_i, y)\right). \tag{41}$$

The difference is due to that these works additionally assume that there is a unique y_i for each x_i among all training data. That is, $\tilde{\mathcal{P}}(x_i, y_i) = \tilde{\mathcal{P}}(x_i)$ and $\tilde{\mathcal{P}}(x_i, y) = 0 \ \forall y \neq y_i$. Under this assumption and using the equality constraint in (38), (39) can be reduced to (41):

$$\begin{aligned} &\sigma^2\big(\tilde{\boldsymbol{f}} - \sum_{i,y}\alpha_{iy}\boldsymbol{f}(x_i, y)\big) = \sigma^2\big(\sum_{i,y}(\tilde{\mathcal{P}}(x_i, y) - \alpha_{iy})\boldsymbol{f}(x_i, y)\big) \\ &= \sigma^2\big(\sum_i \tilde{\mathcal{P}}(x_i)\boldsymbol{f}(x_i, y_i) - \sum_{i,y}\alpha_{i,y}\boldsymbol{f}(x_i, y)\big) = \sigma^2\sum_{i,y}\alpha_{iy}\left(\boldsymbol{f}(x_i, y_i) - \boldsymbol{f}(x_i, y)\right). \end{aligned}$$

Like the situation in LR, the following theorem shows that the optimal $\boldsymbol{\alpha}^*$ for (38) is in general an interior point.

Theorem 7 *The* ME *dual problem* (38) *attains a unique optimal solution* $\boldsymbol{\alpha}^*$ *and for any* i, y

$$\alpha^*_{iy}\begin{cases} = 0 & \text{if } \tilde{\mathcal{P}}(x_i) = 0, \\ \in (0, \tilde{\mathcal{P}}(x_i)) & \text{otherwise.} \end{cases}$$

The proof is in Appendix A.8.

Next we design a coordinate descent method to solve (38). We observe that (38) is very similar to (16) for multi-class SVM in several aspects. First, the $\boldsymbol{\alpha}$ vector can be decomposed to several blocks, and each block is associated with an x_i and all labels; see (40).[8] Second, each equality constraint corresponds to a single x_i. Therefore, we follow Memisevic (2006) and earlier SVM works (Crammer and Singer, 2000; Hsu and Lin, 2002; Keerthi et al., 2008) to consider variables associated with an x_i as a block. The sub-problem is:

$$\begin{aligned} \min_{\boldsymbol{z}} \quad & h(\boldsymbol{z}) \\ \text{subject to} \quad & \sum_y z_y = 0 \quad \text{and} \quad z_y \geq -\alpha_{iy} \ \forall y, \end{aligned} \tag{42}$$

where

$$\begin{aligned} h(\boldsymbol{z}) &\equiv D^{\mathsf{ME}}(\bar{\boldsymbol{\alpha}}_1, \ldots, \bar{\boldsymbol{\alpha}}_i + \boldsymbol{z}, \ldots, \bar{\boldsymbol{\alpha}}_l) \qquad (43) \\ &= \sum_y (\alpha_{iy} + z_y) \log(\alpha_{iy} + z_y) + \frac{1}{2\sigma^2} \left\| \boldsymbol{w}(\boldsymbol{\alpha}) - \sigma^2 \sum_y z_y \boldsymbol{f}(x_i, y) \right\|^2 + \text{constant} \\ &= \sum_y (\alpha_{iy} + z_y) \log(\alpha_{iy} + z_y) - \sum_y z_y \boldsymbol{w}(\boldsymbol{\alpha})^T \boldsymbol{f}(x_i, y) + \frac{\sigma^2}{2} \boldsymbol{z}^T K^i \boldsymbol{z} + \text{constant}, \end{aligned}$$

where $K^i \in R^{|Y| \times |Y|}$ is a matrix with $K^i_{yy'} = \boldsymbol{f}(x_i, y)^T \boldsymbol{f}(x_i, y'),\ \forall y, y' \in Y$.

4.2 Solving the Sub-problem

Clearly, (42) is very similar to the sub-problem in (17) because of the same equality constraint. Eq. (17) has a closed-form solution, but (42) has not due to the log terms in the objective function. Many optimization methods can be applied to solve (42). Collins et al. (2008) propose an exponentiated gradient (EG) method to get an approximate solution. We leave details of EG in Section 5. We follow Memisevic (2006) to use a coordinate descent method, so the procedure for solving (38) becomes a two-level coordinate descent method. Each step of the outer level considers variables associated with an x_i as a block and gets the sub-problem (42). The inner level then solves (42) via coordinate descent methods. Such two-level approaches have been considered in training SVM (e.g., Rüping, 2000; Pérez-Cruz et al., 2004).

To solve the sub-problem (42), each time we select two variables α_{iy_1} and α_{iy_2}. Using the equality constraint, we obtain a one-variable sub-problem:

$$\begin{aligned} \min_d \quad & h(\boldsymbol{z} + d\,(\boldsymbol{e}_{y_1} - \boldsymbol{e}_{y_2})) \equiv (\alpha_{iy_1} + z_{y_1} + d) \log(\alpha_{iy_1} + z_{y_1} + d) \\ & + (\alpha_{iy_2} + z_{y_2} - d) \log(\alpha_{iy_2} + z_{y_2} - d) \\ & + \left(\sigma^2 \left((K^i \boldsymbol{z})_{y_1} - (K^i \boldsymbol{z})_{y_2}\right) - \boldsymbol{w}(\boldsymbol{\alpha})^T \left(\boldsymbol{f}(x_i, y_1) - \boldsymbol{f}(x_i, y_2)\right)\right) d \qquad (44) \\ & + \frac{\sigma^2}{2} (K^i_{y_1 y_1} + K^i_{y_2 y_2} - 2K^i_{y_1 y_2}) d^2 + \text{constant} \\ \text{subject to} \quad & -\left(\alpha_{iy_1} + z_{y_1}\right) \leq d \leq \alpha_{iy_2} + z_{y_2}. \end{aligned}$$

[8] In fact, by defining $\boldsymbol{w}(\boldsymbol{\alpha}) = \begin{bmatrix} \sum_i \alpha_{i1} \boldsymbol{x}_i \\ \vdots \\ \sum_i \alpha_{i|Y|} \boldsymbol{x}_i \end{bmatrix}$, (16) also has a $\boldsymbol{w}(\boldsymbol{\alpha})^T \boldsymbol{w}(\boldsymbol{\alpha})$ term like (38).

Algorithm 6 Solving the sub-problem (42) by a coordinate descent method with maximal violating pairs. We assume the property (48).

- Given $\boldsymbol{\alpha}_i, \boldsymbol{w}(\boldsymbol{\alpha}), K^i_{yy},\ \forall y$.
- $\hat{\boldsymbol{z}}^0 \leftarrow \boldsymbol{\alpha}_i$.
- $v_y \leftarrow \boldsymbol{w}(\boldsymbol{\alpha})^T \boldsymbol{f}(x_i, y),\ \forall y$.
- Find the initial gradient
$$G_y \leftarrow \log(\hat{z}^0_y) + 1 - v_y,\ \forall y.$$
- For $k = 0, 1, 2 \ldots$,
  - If $\max_y G_y = \min_y G_y$, break
  - $y_1 \leftarrow \arg\max_y G_y, \quad y_2 \leftarrow \arg\min_y G_y$.
  - Calculate coefficients of (44) by using the variable $\hat{\boldsymbol{z}}$
$$\begin{aligned} a &\leftarrow \sigma^2 \left(K^i_{y_1 y_1} + K^i_{y_2 y_2}\right) \\ b &\leftarrow \sigma^2 \left((\hat{z}^k_{y_1} - \alpha_{iy_1}) K^i_{y_1 y_1} - (\hat{z}^k_{y_2} - \alpha_{iy_2}) K^i_{y_2 y_2}\right) - v_{y_1} + v_{y_2} \\ c_1 &\leftarrow \hat{z}^k_{y_1}, \quad c_2 \leftarrow \hat{z}^k_{y_2} \end{aligned} \tag{47}$$
  - Solve (44) by Algorithm 4 and get the optimal Z^*_1, Z^*_2.
  - $\hat{z}^{k+1}_{y_1} \leftarrow Z^*_1, \quad \hat{z}^{k+1}_{y_2} \leftarrow Z^*_2$.
  - Update the gradient
$$\begin{aligned} G_{y_1} &\leftarrow \log(\hat{z}^{k+1}_{y_1}) + 1 + \sigma^2 K^i_{y_1 y_1} (\hat{z}^{k+1}_{y_1} - \alpha_{iy_1}) - v_{y_1}, \\ G_{y_2} &\leftarrow \log(\hat{z}^{k+1}_{y_2}) + 1 + \sigma^2 K^i_{y_2 y_2} (\hat{z}^{k+1}_{y_2} - \alpha_{iy_2}) - v_{y_2}. \end{aligned}$$

By assigning

$$\begin{aligned} a &\leftarrow \sigma^2 \left(K^i_{y_1 y_1} + K^i_{y_2 y_2} - 2K^i_{y_1 y_2}\right) \\ b &\leftarrow \sigma^2 \left((K^i \boldsymbol{z})_{y_1} - (K^i \boldsymbol{z})_{y_2}\right) - \boldsymbol{w}(\boldsymbol{\alpha})^T \left(\boldsymbol{f}(x_i, y_1) - \boldsymbol{f}(x_i, y_2)\right) \\ c_1 &\leftarrow \alpha_{iy_1} + z_{y_1} \text{ and } \quad c_2 \leftarrow \alpha_{iy_2} + z_{y_2}, \end{aligned} \tag{45}$$

(44) is in the same form as (18), so Algorithm 4 can be applied.

There are many ways to select the two indices y_1 and y_2. In SVM, this issue, called the working set selection, has been thoroughly studied. For example, we can sequentially go through all pairs of indices. Alternatively, using gradient information (e.g., Joachims, 1998; Keerthi et al., 2001; Fan et al., 2005) may lead to faster convergence. Memisevic (2006) adopts the "maximal violating pair" (Keerthi et al., 2001) by selecting the two indices violating the optimality condition the most. From a proof similar to Theorem 1, the optimal $\boldsymbol{z}^*$ of (42) satisfies $z^*_y > -\alpha_{iy},\ \forall y$. Thus without considering inequality constraints, the optimality condition implies

$$\nabla_{z_y} h(\boldsymbol{z}^*) = \nabla_{z_{y'}} h(\boldsymbol{z}^*),\ \forall y, y',$$

where

$$\nabla_{z_y} h(\boldsymbol{z}) \equiv \log(\alpha_{iy} + z_y) + 1 + \sigma^2 (K^i \boldsymbol{z})_y - \boldsymbol{w}(\boldsymbol{\alpha})^T \boldsymbol{f}(x_i, y). \tag{46}$$

We can select the maximal violating pair by

$$y_1 = \arg\max_y \nabla_{z_y} h(\boldsymbol{z}) \text{ and } y_2 = \arg\min_y \nabla_{z_y} h(\boldsymbol{z}).$$

Algorithm 7 A coordinate descent method for the dual of ME (38)

- Set initial $\boldsymbol{\alpha}$ by (51).
- $\boldsymbol{w}(\boldsymbol{\alpha}) \leftarrow \sigma^2 \left(\sum_i \sum_y \left(\tilde{\mathcal{P}}(x_i, y) - \alpha_{iy} \right) \boldsymbol{f}(x_i, y) \right)$.
- While $\boldsymbol{\alpha}$ is not optimal
  For $i = 1, \ldots, l$
  - Solve the sub-problem (42) by Algorithm 6 and get the optimal $\hat{\boldsymbol{z}}^*$.
  - Update $\boldsymbol{\alpha}$ and $\boldsymbol{w}(\boldsymbol{\alpha})$ by (50).

Once the optimum d^* of (44) is obtained, for the next coordinate descent step we need the new $\nabla h(\boldsymbol{z})$ for selecting the maximal violating pair. As $\boldsymbol{w}(\boldsymbol{\alpha})^T \boldsymbol{f}(x_i, y)$ is considered as a constant in (46), the main cost is on updating $K^i \boldsymbol{z}$ to $K^i(\boldsymbol{z} + d(\boldsymbol{e}_{y_1} - \boldsymbol{e}_{y_2}))$. The vector $K^i \boldsymbol{z}$ should be maintained as it is also used in (45). Therefore, each iteration to solve (42) requires

$$\text{cost for } K^i_{yy_1} d \text{ and } -K^i_{yy_2} d, \ \forall y \ + \ \text{cost for finding pairs} \ + \ \text{cost for solving (44)}.$$

The first term needs $|Y|$ inner products as in general storing $K^i, \ \forall i$ is not possible. This is much more expensive than the second term involving only finding the largest/smallest entries of $|Y|$ values. Moreover, solving (44) is cheap due to the small number of Newton updates. The discussion raises a question if using/maintaining the gradient is cost-effective. For SVM, the same reason leads Hsieh et al. (2008, Section 4) to suggest that for linear SVM we should avoid using gradients for selecting working sets. Fortunately, for most ME applications, features often specify an indicator function of properties of x_i and a class y (Jurafsky and Martin, 2008), so

$$\boldsymbol{f}(x, y)^T \boldsymbol{f}(x, y') = 0, \quad \text{if } y \neq y'. \tag{48}$$

Thus $K^i_{yy'} = 0$ if $y \neq y'$ and (46) is reduced to

$$\nabla_{z_y} h(\boldsymbol{z}) = \log(\alpha_{iy} + z_y) + 1 + \sigma^2 K^i_{yy} z_y - \boldsymbol{w}(\boldsymbol{\alpha})^T \boldsymbol{f}(x_i, y). \tag{49}$$

As $K^i_{yy}, \forall y$ can be pre-stored, the cost for calculating the gradient is significantly reduced to constant time. Therefore, using gradients for the working set selection is very suitable for most ME applications.[9]

For practical implementations, we must handle the numerical issue discussed in Section 3.2 when solving (44). If using Algorithm 4, what we have obtained are Z_1^* and Z_2^*:

$$Z_1^* = \alpha_{iy_1} + z_{y_1} + d^*, \qquad Z_2^* = \alpha_{iy_2} + z_{y_2} - d^*.$$

Therefore, instead of maintaining the vector $\boldsymbol{z}$, we work on $\hat{\boldsymbol{z}} \equiv \bar{\boldsymbol{\alpha}}_i + \boldsymbol{z}$. From (45) and (48), the coefficients of problem (44) using the variable $\hat{\boldsymbol{z}}$ are described in (47). Algorithm 6 gives details for solving (42). In particular, it shows the loop to update the vector $\hat{\boldsymbol{z}}$. Note that $\boldsymbol{w}(\boldsymbol{\alpha})^T \boldsymbol{f}(x_i, y), \ \forall y$ is a constant vector independent of the loop, so we pre-calculate and pre-store it as a vector $\boldsymbol{v}$. The following theorem shows that Algorithm 6 solves (42):

Theorem 8 *The sequence* $\{\hat{\boldsymbol{z}}^0, \hat{\boldsymbol{z}}^1, \ldots\}$ *generated by Algorithm 6 converges to* $\boldsymbol{\alpha}_i + \boldsymbol{z}^*$, *where* $\boldsymbol{z}^*$ *is the optimum of* (42).

The proof is omitted because it is very similar to Theorem 1 in Keerthi et al. (2005).

[9] If using a heap structure for the gradient, then maintaining the heap and getting the maximal violating pair cost only $O(\log |Y|)$. However, this is only useful when $|Y|$ is large.

Algorithm 8 A randomized online EG algorithm (Collins et al., 2008)

- Given maxTrial and a learning rate $\eta_i = 0.5\ \forall i = 1, \ldots, l$. Set initial $\boldsymbol{\alpha}$.
- $\boldsymbol{w}(\boldsymbol{\alpha}) \equiv \sigma^2 \sum_{i,y} \alpha_{iy} \left(\boldsymbol{f}(x_i, y_i) - \boldsymbol{f}(x_i, y)\right)$.
- While $\boldsymbol{\alpha}$ is not optimal
  - Randomly choose i from the set $\{1, \ldots, l\}$.
  - trial $= 0$
  - While trial $<$ maxTrial
    - Calculate α'_{iy} by (52).
    - If $D^{\mathsf{ME}}(\boldsymbol{\alpha}') - D^{\mathsf{ME}}(\boldsymbol{\alpha}) \leq 0$
      - $\eta_i \leftarrow 1.05\eta_i$.
      - Update $\boldsymbol{\alpha}$ and $\boldsymbol{w}(\boldsymbol{\alpha})$ by Eqs. similar to (50).
      - Break.
    - Else
      - $\eta_i \leftarrow \eta_i/2$.
    - trial $\leftarrow$ trial $+ 1$.

4.3 The Overall Procedure

The overall procedure to solve ME dual is in Algorithm 7. Under the coordinate descent setting, we sequentially update $\bar{\boldsymbol{\alpha}}_i$ by solving (42). Once (42) is solved and $\hat{\boldsymbol{z}}^* = \boldsymbol{z}^* + \bar{\boldsymbol{\alpha}}_i$ is obtained, $\boldsymbol{\alpha}$ and $\boldsymbol{w}(\boldsymbol{\alpha})$ are respectively updated by

$$\begin{aligned} \bar{\boldsymbol{\alpha}}_i &\leftarrow \hat{\boldsymbol{z}}^*, \\ \boldsymbol{w}(\boldsymbol{\alpha}) &\leftarrow \boldsymbol{w}(\boldsymbol{\alpha}) - \sigma^2 \sum_y \left(\hat{z}^*_y - \alpha_{iy}\right) \boldsymbol{f}(x_i, y). \end{aligned} \tag{50}$$

This calculation needs to access $\boldsymbol{f}(x_i, y),\ \forall y$. As finding $\boldsymbol{w}(\boldsymbol{\alpha})^T \boldsymbol{f}(x_i, y),\ \forall y$ before solving (42) requires the same data access, the update in (50) is affordable.

Regarding the initial point, similar to the case in LR, $\boldsymbol{\alpha} = \mathbf{0}$ is not a valid point. Memisevic (2006) simply sets $\alpha_{iy} = \tilde{\mathcal{P}}(x_i)/|Y|$ to satisfy the equality constraint $\sum_y \alpha_{iy} = \tilde{\mathcal{P}}(x_i)$. From the optimality condition (75), we think that α_{iy} should be related to $\tilde{\mathcal{P}}(x_i, y)$. For each i, we consider two cases based on the unseen label set $E_i \equiv \{y \mid \tilde{\mathcal{P}}(x_i, y) = 0\}$ and heuristically set

$$\alpha_{iy} = \begin{cases} \tilde{\mathcal{P}}(x_i, y) & \text{if } |E_i| = 0, \\ \begin{cases} (1-\epsilon)\tilde{\mathcal{P}}(x_i, y) & \forall y \notin E_i \\ \frac{\epsilon}{|E_i|}\tilde{\mathcal{P}}(x_i) & \forall y \in E_i \end{cases} & \text{if } |E_i| \neq 0, \end{cases} \tag{51}$$

where ϵ is a small positive value. The following theorem shows that Algorithm 7 solves (38).

Theorem 9 *The sequence generated by Algorithm 7 converges to the optimum $\boldsymbol{\alpha}^*$ of* (38).

The proof is in Appendix A.9.

5 A Related Method

In this section, we describe an existing method which also solves dual ME.

5.1 Exponentiated Gradient Method

Collins et al. (2008) propose batch and online exponentiated gradient (EG) algorithms for CRF. Their methods are applicable to ME as ME is a special case of CRF. Here we discuss only their online EG algorithm, as it is more related to our coordinate descent methods. At each iteration an example i is randomly chosen from $\{1, \ldots, l\}$ and $\bar{\boldsymbol{\alpha}}_i$ is updated to $\bar{\boldsymbol{\alpha}}'_i$ by the following way.

$$\alpha'_{iy} = \frac{\alpha_{iy} \exp(-\eta_i \nabla_{iy})}{\sum_{y'} \alpha_{iy'} \exp(-\eta_i \nabla_{iy'})}, \ \forall y, \tag{52}$$

where

$$\nabla_{iy} \equiv \frac{\partial D^{\mathsf{ME}}(\boldsymbol{\alpha})}{\partial \alpha_{iy}} = 1 + \log \alpha_{iy} + \boldsymbol{w}(\boldsymbol{\alpha})^T \left(\boldsymbol{f}(x_i, y_i) - \boldsymbol{f}(x_i, y)\right) \tag{53}$$

and $\eta_i > 0$ is a learning rate. Note that we follow Collins et al. (2008) to use $\boldsymbol{w}(\boldsymbol{\alpha})$ in (41).

To improve the convergence, Collins et al. (2008) adaptively adjust the learning rate η_i for each instance. If the function value does not decrease, they iteratively halve η_i at most maxTrial times (maxTrial is set by users). Finally, they slightly increase η_i to avoid it being too small. The detailed procedure is in Algorithm 8.

The most expensive operation in Algorithm 8 is to calculate the function difference. Using (38),

$$\begin{aligned}
& D^{\mathsf{ME}}(\boldsymbol{\alpha}') - D^{\mathsf{ME}}(\boldsymbol{\alpha}) \\
=& \sum_y \alpha'_{iy} \log \alpha'_{iy} - \sum_y \alpha_{iy} \log \alpha_{iy} \\
& + \frac{1}{2\sigma^2} \left(\|\boldsymbol{w}(\boldsymbol{\alpha}) + \sigma^2 \sum_y (\alpha'_{iy} - \alpha_{iy}) \boldsymbol{f}(x_i, y)\|^2 - \|\boldsymbol{w}(\boldsymbol{\alpha})\|^2 \right) \\
=& \sum_y \alpha'_{iy} \log \alpha'_{iy} - \sum_y \alpha_{iy} \log \alpha_{iy} + \sum_y (\alpha'_{iy} - \alpha_{iy}) \boldsymbol{w}(\boldsymbol{\alpha})^T \boldsymbol{f}(x_i, y) \\
& + \frac{\sigma^2}{2} (\bar{\boldsymbol{\alpha}}'_i - \bar{\boldsymbol{\alpha}}_i) K^i (\bar{\boldsymbol{\alpha}}'_i - \bar{\boldsymbol{\alpha}}_i).
\end{aligned} \tag{54}$$

The vector $\boldsymbol{w}(\boldsymbol{\alpha})$ is maintained in a way similar to (50), so the most expensive operation in (54) is for inner products between features (see the last term). If the condition (48) holds and $K^i_{yy}, \ \forall y$ are pre-calculated, then (54) needs $O(|Y|)$ time. Thus each of the maxTrial iterations in Algorithm 8 costs $O(|Y|)$, comparable to each coordinate descent step in Algorithm 6.

EG differs from our Algorithm 6 mainly on solving the sub-problem (42). Ours more accurately solves the sub-problem, while EG uses only the update rule (52). Therefore, EG's convergence may be slower. However, EG's implementation is easier and we do not observe numerical difficulties such as catastrophic cancellations described in Section 3.3.

6 Experiments

In this section, we investigate the performance of the proposed coordinate descent methods for logistic regression and maximum entropy. We consider two types of NLP

Table 2: Statistics of data (real-valued features). l: number of instances, n: number of features, #nz: number of total non-zero feature values, and C: best regularization parameter from five-fold cross validation.

Problem	l	n	#nz	C
a9a	32,561	123	451,592	4
real-sim	72,309	20,958	3,709,083	8
yahoo-japan	176,203	832,026	23,506,415	4
rcv1	677,399	47,236	49,556,258	8

applications. One is logistic regression for data with real-valued features and the other is maximum entropy for 0/1-featured data. Programs used for experiments are available at

`http://www.csie.ntu.edu.tw/~cjlin/liblinear/exp.html`.

We run all experiments on a 64-bit machine with Intel Xeon 2.0GHz CPU and 32GB main memory.

6.1 Logistic Regression for Data Classification

We compare the following implementations. The first two solve the dual, while the other three solve the primal.

1. CDdual: the dual coordinate descent method in Algorithm 5.
2. CDdual-ls: the same as CDdual except that the sub-problem (18) is approximately solved by one Newton update with line search; see (21). The setting is similar to that in Section 2.2 for primal LR. We use $\beta = 0.5$ and $\gamma = 0.01$.
3. CDprimal: a primal coordinate descent method for logistic regression; see Section 2.2.
4. EG: an online exponentiated gradient implementation for LR; see Section 5.1.
5. LBFGS: a limited memory quasi Newton method for general unconstrained optimization problems (Liu and Nocedal, 1989).
6. TRON: a trust region Newton method for logistic regression (Lin et al., 2008).

Our implementations are extended from the framework used in Huang et al. (2010). We consider four data sets. All of them except yahoo-japan are available at LIBSVM data set.[10] Data statistics and the regularization parameter C (obtained by cross validation) are in Table 2. The initial $\boldsymbol{w}$ of the three primal-based methods is $\mathbf{0}$. For CDdual, CDdual-ls and EG, the dual-based methods, the initial solution is via (36) with $\epsilon_1 = 10^{-3}$ and $\epsilon_2 = 10^{-8}$. All three coordinate descent methods (CDdual, CDdual-ls, CDprimal) apply the random permutations of indices; see the explanation in Section 2.1. For CDdual, we set $\xi = 0.1$ in Algorithm 4. For the stopping condition of Algorithm 4, we use $|g_t'(Z_t)| \le \epsilon'$, where ϵ' is set to 10^{-2} initially and is gradually reduced to $\epsilon' = 10^{-8}$. This strategy saves Newton iterations in the early stage.

We begin with checking training time versus the relative difference of the function value to the optimum:

$$\frac{P^{\mathsf{LR}}(\boldsymbol{w}) - P^{\mathsf{LR}}(\boldsymbol{w}^*)}{P^{\mathsf{LR}}(\boldsymbol{w}^*)}, \tag{55}$$

[10] `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets`

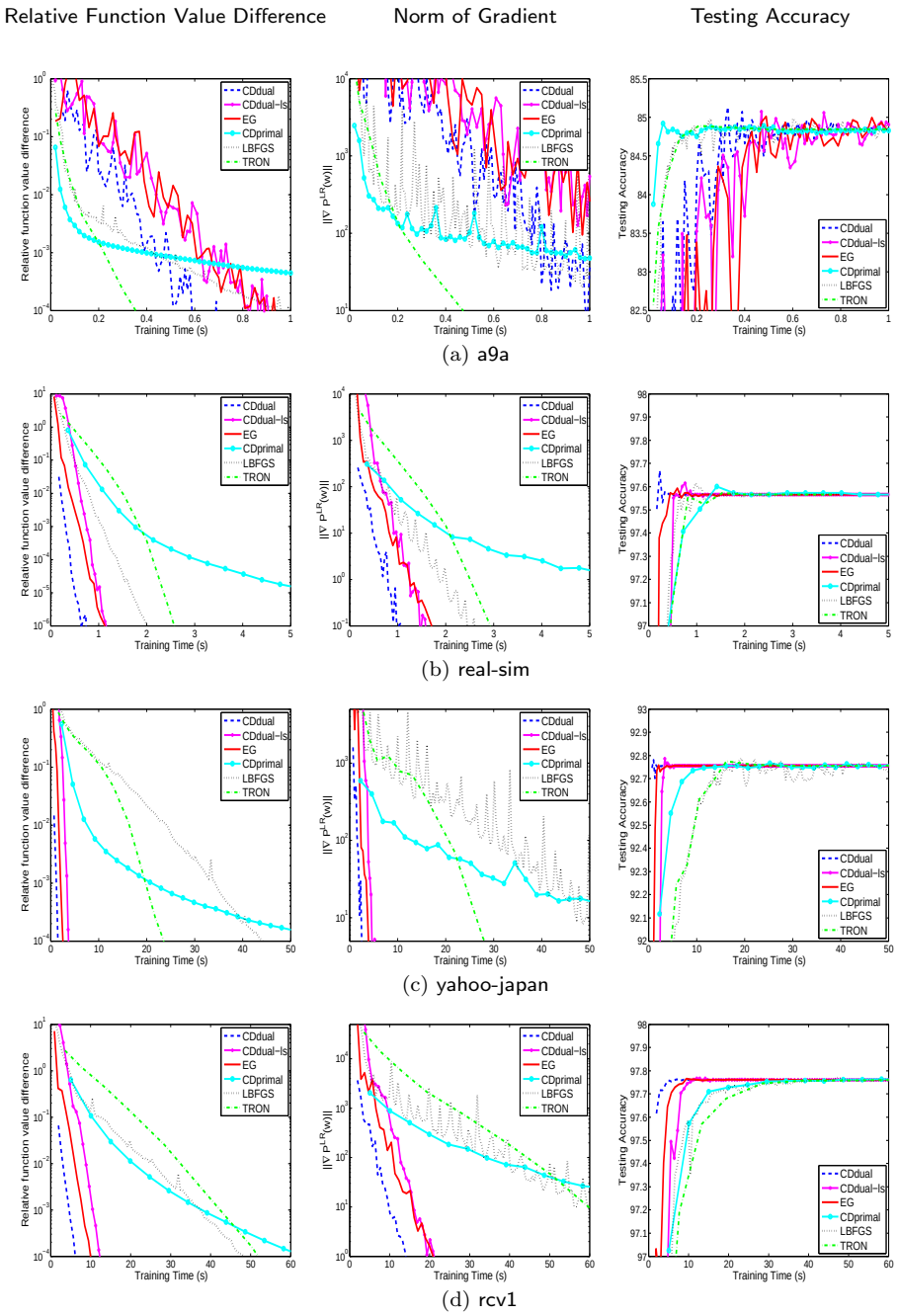


Fig. 2: Results for logistic regression on real-valued document data. The first column shows time versus the relative function difference (55). The second and third columns show $\|\nabla P^{\mathsf{LR}}(\boldsymbol{w})\|$ and testing performances along time, respectively. Time is in seconds.

Table 3: CDdual for LR with different ξ. The table shows time in seconds to reduce the relative difference to the optimal function value to be less than 0.01. We boldface the best approach. Clearly, the running time is not sensitive to the choice of ξ.

Problem	$\xi = 0.1$	$\xi = 0.5$	$\xi = 0.9$
a9a	0.30	**0.29**	0.31
real-sim	**0.24**	**0.24**	**0.24**
yahoo-japan	1.02	**1.01**	1.02
rcv1	**3.56**	3.59	3.65

where $\boldsymbol{w}^*$ is the optimal solution of (1). As $\boldsymbol{w}^*$ is not available, we obtain a reference point satisfying $\|\nabla P^{\mathsf{LR}}(\boldsymbol{w})\| \leq 0.01$. We use primal objective values even for dual solvers because from a dual solution it is easy to estimate a primal solution by (22). In contrast, finding a corresponding dual solution from a given primal vector $\boldsymbol{w}$ is more difficult. Results of (55) are in the first column of Figure 2. Next, we check these methods' gradient values in the second column of Figure 2, as $\|\nabla P^{\mathsf{LR}}(\boldsymbol{w})\| = 0$ implies that $\boldsymbol{w}$ is the global minimum. We are also interested in the time needed to achieve a reasonable testing result. The third column of Figure 2 presents testing accuracy versus training time. Note that (55) and $\|\nabla P^{\mathsf{LR}}(\boldsymbol{w})\|$ in Figure 2 are both log scaled.

From Figure 2, CDdual and CDdual-ls are more efficient than other solvers on all problems except a9a. Note that a9a has much fewer features than data points. For such problems solving the primal problem may be more suitable because the number of variables is the same as the number of features. We observe that CDdual is always faster than CDdual-ls, a result consistent with the analysis in Section 3. CDprimal is worse than CDdual because of its slower convergence and higher cost per iteration. From the discussion in Section 2.2 and (25), for every round of going through all variables, CDprimal (n variables) and CDdual (l variables) respectively need

$$O(nl) \quad \text{and} \quad O(l \times \#\text{Newton Steps})$$

exp/log operations, where #Newton steps is the average number of Newton updates in Algorithm 4. We experimentally observe that for all problems except a9a, to go through all variables once, CDprimal is at least six times more expensive than CDdual. Regarding the three dual-based methods CDdual, CDdual-ls and EG, CDdual is generally faster. For TRON and LBFGS, they are Newton and quasi-Newton methods respectively, so fast final convergence is observed. However, since they take significant efforts at each iteration, they fail to generate a reasonable model quickly. From the experiment results, CDdual converges as fast as TRON and LBFGS, but also performs well in early iterations.

We find that different initial z's in the Newton method for CDdual cause different running time. Using $z = 0$ is the best because near the optimum, $\boldsymbol{\alpha}$ is not changed much and z is close to zero. Regarding the parameter ξ in CDdual, Table 3 shows that the running time is not sensitive to the choice of ξ. This is because the operation in (26) takes only a small portion of the total running time.[11]

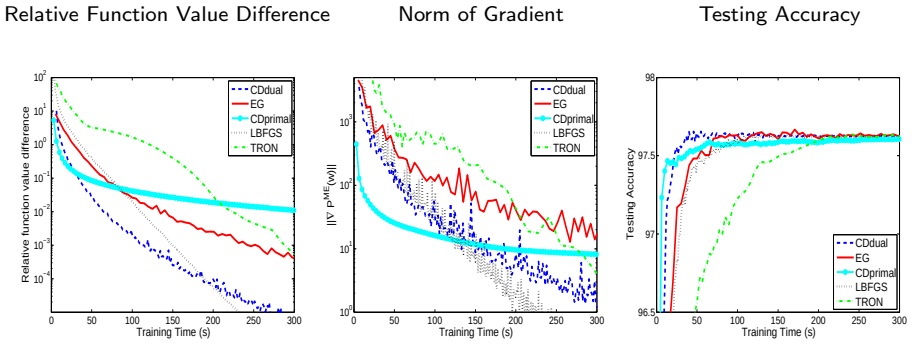
6.2 ME for 0/1-featured Data in NLP

We apply ME models to part of speech (POS) tagging and chunking tasks following the setting in Huang et al. (2010). It is based on the OpenNLP package (Baldridge et al.,

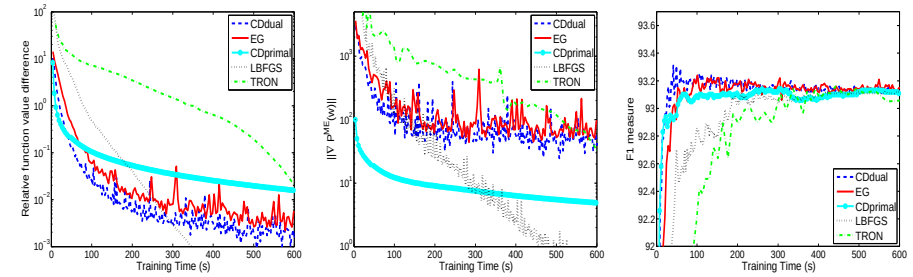
[11] Note that (26) is used only if $z^k + d \notin (-c_1, c_2)$.

Table 4: Statistics of NLP data (0/1 features). l: number of contexts, $|Y|$: number of class labels, n: number of features, and #nz: number of total non-zero feature values

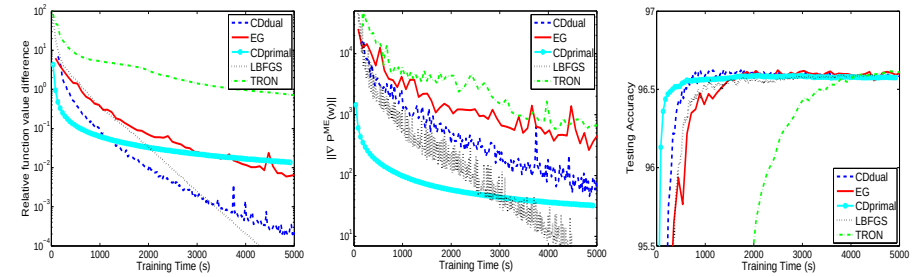
Data set	l	$\|Y\|$	n	#nz
CoNLL2000-P	197,979	44	168,674	48,030,163
CoNLL2000-C	197,252	22	273,680	53,396,844
BROWN	935,137	185	626,726	601,216,661



(a) CoNLL2000-P



(b) CoNLL2000-C



(c) BROWN

Fig. 3: Results for maximum entropy on 0/1-featured data. The first column shows time versus the relative function difference (55). The second and third columns show $\|\nabla P^{\mathsf{ME}}(\boldsymbol{w})\|$ and testing performances along time, respectively. Time is in seconds.

2001), which extracts binary features and predicts the tag sequences by the method in Ratnaparkhi (1998). We use CoNLL2000 shared task data[12] for chunking (denoted as CoNLL2000-C) and POS tagging (CoNLL2000-P), and BROWN corpus[13] for POS tagging. Table 4 lists the statistics of data sets.

We compare the following methods: CDdual, CDprimal, LBFGS, TRON and EG. CDdual-ls is not included because it is shown in Section 6.1 to be slower than CDdual. CDdual and EG solve the dual problem, while the others solve the primal. We use the regularization parameter $\sigma^2 = 10l$. As Huang et al. (2010) report under this value, ME achieve good testing performances. The initial $\boldsymbol{w}$ of primal-based methods is $\mathbf{0}$. For CDdual and EG, the initial $\boldsymbol{\alpha}$ is set by (51) with $\epsilon = 10^{-10}$. Figure 3 shows the results of the relative function difference to the optimum, the gradient $\|\nabla P^{\mathsf{ME}}(\boldsymbol{w})\|$, and the testing accuracy.

For the function value, results in Figure 3 are different from Figure 2, in which CDdual is the fastest all the time. Now CDprimal is the fastest in the beginning, but has the slowest final convergence. CDdual is only slightly slower than CDprimal in the very early stage, but its final convergence is much better. Moreover, LBFGS may surpass CDdual in the final stage. Regarding the two dual-based methods CDdual and EG, CDdual is generally faster. Overall, the proposed CDdual method is competitive for these data sets.

6.3 A Comparison between Algorithm 4 and a Strategy of Combining Bisection and Newton Methods

In Section 3.2, we use Newton methods to solve the sub-problem (18). If z^k is on the "wrong" side of z^* , we use the technique (30) and prove that in a finite steps a point on the "correct" side will be obtained. Here we experiment with an alternative strategy by using a bisection method to find a point on the "correct" side of z^* before Newton updates.

Since $g'(z^*) = 0$ and $g'(z)$ is increasing, (33) and (34) imply that a point on the "correct" side of z^* satisfies

$$g'(z) \begin{cases} \le 0 & \text{if } t = 1, \\ \ge 0 & \text{if } t = 2. \end{cases} \tag{56}$$

From the fact $g_1'(Z_1) = g'(z)$ and $g_2'(Z_2) = -g'(z)$, (56) becomes

$$g_t'(Z_t) \le 0. \tag{57}$$

Simple calculations show that $g_t'(0) = -\infty$ and $g_t'(s/2) \ge 0$. Therefore, starting from a point in $(0, s/2]$, the bisection method sequentially cut the point to half until (57) is safisfied. See Algorithm 9 for details. In our implementation, (35) is used as the initial point of the bisection procedure.

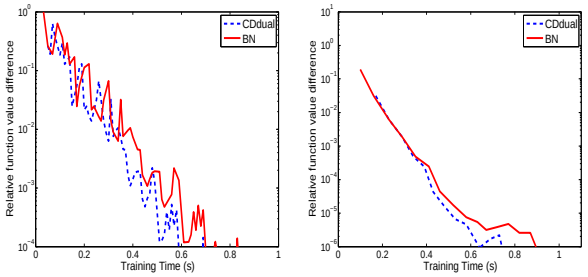
We refer to the strategy of combining bisection and Newton methods as BN. In Figure 4, we compare BN and CDdual. Note that BN is the same as CDdual except that (18) is solved by Algorithm 9. We can see that CDdual has slightly better final convergence. The reason seems to be that Algorithm 4 takes Newton updates regardless of whether the current z^k is on the "correct" side of z^* or not. The only exception is that the point after update is outside the interval $(-c_1, c_2)$; see (30).

[12] `http://www.cnts.ua.ac.be/conll2000/chunking`

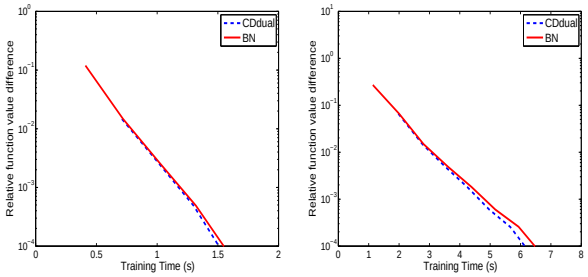
[13] `http://www.nltk.org`

Algorithm 9 A combination of bisection and Newton methods to solve (18)

- Given coefficients: a, b, c_1, and c_2.
- $t \leftarrow \begin{cases} 1 & \text{if } z_m \geq \frac{-b}{a}, \\ 2 & \text{if } z_m < \frac{-b}{a}. \end{cases}$
- $Z_t^0 \in (0, s/2)$.
- While 1
  - If $g_t'(Z_t^0) \leq 0$, break.
    Else, $Z_t^0 \leftarrow Z_t^0/2$.
- For $k = 0, 1, \ldots$
  - If $g_t'(Z_t^k) = 0$, break.
  - $Z_t^{k+1} \leftarrow Z_t^k - g_t'(Z_t^k)/g_t''(Z_t^k)$.
- $\begin{cases} Z_2^k = s - Z_1^k & \text{if } t = 1, \\ Z_1^k = s - Z_2^k & \text{if } t = 2. \end{cases}$
- return (Z_1^k, Z_2^k).



(a) a9a (b) real-sim



(c) yahoo-japan (d) rcv1

Fig. 4: A comparison between BN and CDdual on logistic regression for real-valued document data. The figures show time versus the relative function value difference defined in (55). Time is in seconds.

7 Discussion and Conclusions

We have illustrated in various places that this work is related to some our earlier developments. Table 5 summarizes their relationship.

In summary, motivated from the success of coordinate descent methods for solving SVM dual, in this work we study if similar methods can be used for LR and ME duals. An important lesson learned is that some algorithmic and implementation details are different from SVM. This is mainly because log operations are involved in LR and ME

Table 5: The relationship between this work and our earlier developments.

	2-class		multi-class	
	SVM	LR	SVM	ME
Primal	Chang et al. (2008)	Huang et al. (2010)		Huang et al. (2010)
Dual	Hsieh et al. (2008)	This paper	Keerthi et al. (2008)	This paper

dual problems. We carefully address theoretical and numerical issues of the coordinate descent procedure. Experiments indicate that the proposed method is faster than state of the art methods for logistic regression and maximum entropy.

Acknowledgements The authors thank anonymous reviewers for helpful comments. This work was supported in part by the National Science Council of Taiwan via the grant 98-2221-E-002-136-MY3.

References

Jason Baldridge, Tom Morton, and Gann Bierner. OpenNLP package, 2001. URL `http://opennlp.sourceforge.net/`.

Dimitri P. Bertsekas. *Nonlinear Programming*. Athena Scientific, Belmont, MA 02178-9998, second edition, 1999.

Kai-Wei Chang, Cho-Jui Hsieh, and Chih-Jen Lin. Coordinate descent method for large-scale L2-loss linear SVM. *Journal of Machine Learning Research*, 9:1369–1398, 2008. URL `http://www.csie.ntu.edu.tw/~cjlin/papers/cdl2.pdf`.

Michael Collins, Amir Globerson, Terry Koo, Xavier Carreras, and Peter Bartlett. Exponentiated gradient algorithms for conditional random fields and max-margin Markov networks. *Journal of Machine Learning Research*, 9:1775–1822, 2008.

Koby Crammer and Yoram Singer. On the learnability and design of output codes for multiclass problems. In *Computational Learing Theory*, pages 35–46, 2000.

John N. Darroch and Douglas Ratcliff. Generalized iterative scaling for log-linear models. *The Annals of Mathematical Statistics*, 43(5):1470–1480, 1972.

Stephen Della Pietra, Vincent Della Pietra, and John Lafferty. Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19 (4):380–393, 1997.

Rong-En Fan, Pai-Hsuen Chen, and Chih-Jen Lin. Working set selection using second order information for training SVM. *Journal of Machine Learning Research*, 6:1889–1918, 2005. URL `http://www.csie.ntu.edu.tw/~cjlin/papers/quadworkset.pdf`.

Jianfeng Gao, Galen Andrew, Mark Johnson, and Kristina Toutanova. A comparative study of parameter estimation methods statistical natural language processing. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics (ACL)*, pages 824–831, 2007.

David Goldberg. What every computer scientist should know about floating-point arithmetic. *ACM Computing Surveys*, 23(1):5–48, 1991.

Joshua Goodman. Sequential conditional generalized iterative scaling. In *Proceedings of the 40th Annual Meeting of the Association of Computational Linguistics (ACL)*, pages 9–16, 2002.

Cho-Jui Hsieh, Kai-Wei Chang, Chih-Jen Lin, S. Sathiya Keerthi, and Sellamanickam Sundararajan. A dual coordinate descent method for large-scale linear SVM. In *Pro-*

ceedings of the Twenty Fifth International Conference on Machine Learning (ICML), 2008. URL `http://www.csie.ntu.edu.tw/~cjlin/papers/cddual.pdf`.

Chih-Wei Hsu and Chih-Jen Lin. A comparison of methods for multi-class support vector machines. *IEEE Transactions on Neural Networks*, 13(2):415–425, 2002.

Fang-Lan Huang, Cho-Jui Hsieh, Kai-Wei Chang, and Chih-Jen Lin. Iterative scaling and coordinate descent methods for maximum entropy. *Journal of Machine Learning Research*, 11:815–848, 2010. URL `http://www.csie.ntu.edu.tw/~cjlin/papers/maxent_journal.pdf`.

Tommi S. Jaakkola and David Haussler. Probabilistic kernel regression models. In *Proceedings of the 1999 Conference on AI and Statistics*. Society for Artificial Intelligence in Statistics, 1999.

Rong Jin, Rong Yan, Jian Zhang, and Alex G. Hauptmann. A faster iterative scaling algorithm for conditional exponential model. In *Proceedings of the Twentieth International Conference on Machine Learning (ICML)*, 2003.

Thorsten Joachims. Making large-scale SVM learning practical. In Bernhard Schölkopf, Christopher J. C. Burges, and Alexander J. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*, Cambridge, MA, 1998. MIT Press.

Daniel Jurafsky and James H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition*. Prentice Hall, second edition, 2008.

S. Sathiya Keerthi, Shirish Krishnaj Shevade, Chiranjib Bhattacharyya, and Karuturi Radha Krishna Murthy. Improvements to Platt's SMO algorithm for SVM classifier design. *Neural Computation*, 13:637–649, 2001.

S. Sathiya Keerthi, Kaibo Duan, Shirish Shevade, and Aun Neow Poo. A fast dual algorithm for kernel logistic regression. *Machine Learning*, 61:151–165, 2005.

S. Sathiya Keerthi, Sellamanickam Sundararajan, Kai-Wei Chang, Cho-Jui Hsieh, and Chih-Jen Lin. A sequential dual method for large scale multi-class linear SVMs. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2008. URL `http://www.csie.ntu.edu.tw/~cjlin/papers/sdm_kdd.pdf`.

Paul Komarek and Andrew W. Moore. Making logistic regression a core data mining tool: A practical investigation of accuracy, speed, and simplicity. Technical Report TR-05-27, Robotics Institute, Carnegie Mellon University, 2005.

Guy Lebanon and John Lafferty. Boosting and maximum likelihood for exponential models. In *Advances in Neural Information Processing Systems 14*. MIT Press, 2002.

Chih-Jen Lin, Ruby C. Weng, and S. Sathiya Keerthi. Trust region Newton method for large-scale logistic regression. *Journal of Machine Learning Research*, 9:627–650, 2008. URL `http://www.csie.ntu.edu.tw/~cjlin/papers/logistic.pdf`.

Dong C. Liu and Jorge Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45(1):503–528, 1989.

Zhi-Quan Luo and Paul Tseng. On the convergence of coordinate descent method for convex differentiable minimization. *Journal of Optimization Theory and Applications*, 72(1):7–35, 1992.

Robert Malouf. A comparison of algorithms for maximum entropy parameter estimation. In *Proceedings of the 6th conference on Natural language learning*, pages 1–7. Association for Computational Linguistics, 2002.

Roland Memisevic. Dual optimization of conditional probability models. Technical report, Department of Computer Science, University of Toronto, 2006.

Thomas P. Minka. A comparison of numerical optimizers for logistic regression, 2003. URL `http://research.microsoft.com/~minka/papers/logreg/`.

Fernando Pérez-Cruz, Aníbal R. Figueiras-Vidal, and Antonio Artés-Rodríguez. Double chunking for solving SVMs for very large datasets. In *Proceedings of Learning 2004, Spain*, 2004.

Adwait Ratnaparkhi. *Maximum Entropy Models For Natural Language Ambiguity Resolution*. PhD thesis, University of Pennsylvania, 1998.

Stefan Rüping. mySVM - another one of those support vector machines, 2000. Software available at `http://www-ai.cs.uni-dortmund.de/SOFTWARE/MYSVM/`.

Tong Zhang. On the dual formulation of regularized linear systems with convex risks. *Machine Learning*, 46(1–3):91–129, 2002.

A Appendix

A.1 Lemma 1

We need the following lemma to prove subsequent theorems.

Lemma 1 *Let $f(x)$ be a continuous function over $[a, b]$. If*

1. *f is differentiable in (a, b),*
2. *$\lim_{x \to a^+} f'(x) = -\infty$,*
3. *$\lim_{x \to b^-} f'(x) = \infty$,*

then there are $x_a, x_b \in (a, b)$ such that $f(x_a) < f(a)$ and $f(x_b) < f(b)$. That is, any minimizer of f must be an interior point.

Proof. From the second condition, there exists $x_a \in (a, b)$ such that $f'(x) < 0\ \forall x \in (a, x_a)$. By the Mean-Value Theorem, there exists $\epsilon \in (a, x_a)$ such that

$$f(x_a) = f(a) + f'(\epsilon)(x_a - a).$$

Then $f(x_a) < f(a)$ due to $f'(\epsilon) < 0$ and $(x_a - a) > 0$. By similar arguments, there is $x_b \in (a, b)$ such that $f(x_b) < f(b)$.

A.2 Proof of Theorem 1

By defining $0 \log 0 = 0$, $D^{\mathsf{LR}}(\boldsymbol{\alpha})$ is a continuous function on a closed set $[0, C]^l$. Hence a minimum in $[0, C]^l$ exists. We prove that any minimizer $\boldsymbol{\alpha}^* \in (0, C)^l$. Suppose that $\alpha_i^* = 0$ for some i. Consider the following one-variable problem

$$\begin{aligned} \min_z \quad & g(z) = D^{\mathsf{LR}}(\alpha_1^*, \ldots, \alpha_i^* + z, \ldots, \alpha_l^*) \\ & = z \log z + (C - z) \log(C - z) + (\boldsymbol{x}_i^T \boldsymbol{x}_i) z^2 + (Q\boldsymbol{\alpha}^*)_i z + \text{constant} \\ \text{subject to} \quad & 0 \le z \le C. \end{aligned}$$

By Lemma 1, there is $z^* \in (0, C)$ such that $g(z^*) < g(0) = D^{\mathsf{LR}}(\boldsymbol{\alpha}^*)$, which contradicts that $\boldsymbol{\alpha}^*$ is a minimizer. By the same arguments, we can get that $\alpha_i^* < C\ \forall i$.

Next we show the uniqueness by claiming that $D^{\mathsf{LR}}(\boldsymbol{\alpha})$ is strictly convex in $(0, C)^l$. The Hessian $\nabla^2 D^{\mathsf{LR}}(\boldsymbol{\alpha})$ of (3) is the sum of a positive semi-definite matrix Q and a diagonal matrix with positive entries $C/(\alpha_i(C - \alpha_i))\ \forall i$. Thus $\nabla^2 D^{\mathsf{LR}}(\boldsymbol{\alpha})$ is positive definite and $D^{\mathsf{LR}}(\boldsymbol{\alpha})$ is strictly convex. Then the uniqueness of the optimum is obtained.

A.3 Proof of Theorem 2

Since $g(z)$ satisfies all three conditions in Lemma 1, immediately we have $z^* \in (-c_1, c_2)$. The optimality condition and the property $z^* \in (-c_1, c_2)$ then imply $g'(z^*) = 0$.

A.4 Proof of Theorem 4

To begin, we list four important properties for the function $g(z)$:

$$g'(z_1) < g'(z_2), \quad \text{if } z_1 < z_2, \tag{58}$$

$$g''(z_1) < g''(z_2), \quad \text{if } z_m \le z_1 < z_2, \tag{59}$$

$$g''(z_1) > g''(z_2), \quad \text{if } z_1 < z_2 \le z_m, \tag{60}$$

$$g''(z) > 0, \quad \forall z. \tag{61}$$

We prove the results for the situation $z^* \ge z_m$ as the proof for the other situation is similar.

If the result does not hold, then starting from z^k, we have

$$z^{k+s} < z^*, \ \forall s = 0, 1, 2, \ldots. \tag{62}$$

From z^{k+s} to z^{k+s+1}, two update rules may be applied:

$$z^{k+s+1} = z^{k+s} - g'(z^{k+s})/g''(z^{k+s}), \tag{63}$$

$$z^{k+s+1} = \xi z^{k+s} + (1-\xi)c_2. \tag{64}$$

Using (58) and (62), $g'(z^{k+s}) < g'(z^*) = 0$. With (61) and $\xi > 0$, both update rules lead to

$$z^{k+s+1} > z^{k+s}, \ \forall s. \tag{65}$$

We claim that the number of updates via (64) must be finite. Otherwise, since

$$c_2 - z^{k+s+1} = \xi(c_2 - z^{k+s})$$

if (64) is taken, an infinite number of updates via (64) and the property in (65) will cause that $\{z^{k+s}\}$ converges to c_2. As $z^* < c_2$ by Theorem 2, $\{z^{k+s}\}$ will eventually be larger than z^* and the assumption (62) is violated. Therefore, we can let k_0 be the starting index so that all z^{k_0+s}, $\forall s$ are generated by (63).

We then claim that there exists $k_1 \ge k_0$ such that $z^{k_1} \ge z_m$. If such k_1 does not exist, then

$$z^{k_0+s} \le z_m, \ \forall s. \tag{66}$$

Consider the difference between two consecutive iterations:

$$\{\bar{z}^{k_0+s} \mid \bar{z}^{k_0+s} \equiv z^{k_0+s+1} - z^{k_0+s} = -g'(z^{k_0+s})/g''(z^{k_0+s})\}.$$

From (60), (61), and (66) we have $0 < g''(z^{k_0+s}) < g''(z^{k_0})$. With (58) and (66),

$$\bar{z}^{k_0+s} = \frac{-g'(z^{k_0+s})}{g''(z^{k_0+s})} > \frac{-g'(z_m)}{g''(z^{k_0})} > 0, \ \forall s.$$

However, $\{\bar{z}^{k_0+s}\}$ should approach 0 as $\{z^{k_0+s}\}$ is a convergent sequence following from the increasing property (65) and the boundedness (62). Therefore (66) is wrong and k_1 exists such that $z^{k_1} \ge z_m$.

By the Mean-Value Theorem, (63) and (65), there is $\tilde{z} \in (z^{k_1}, z^{k_1+1})$ such that

$$\begin{aligned} g'(z^{k_1+1}) &= g'(z^{k_1}) + g''(\tilde{z})\frac{-g'(z^{k_1})}{g''(z^{k_1})} \\ &= g'(z^{k_1})\left(1 - \frac{g''(\tilde{z})}{g''(z^{k_1})}\right) > 0. \end{aligned}$$

The inequality comes from $g'(z^{k_1}) < 0$ by (58) and (62), and $g''(\tilde{z}) > g''(z^{k_1})$ by $z^{k_1} \ge z_m$ and (59). As $g'(z^{k_1+1}) > 0$ implies $z^{k_1+1} > z^*$, we obtain a contradiction to (62). Thus there is k' such that $z^{k'} \ge z^*$ and the proof is complete.

A.5 Proof of (34)

The first relationship follows from the fact that z_m is the middle points of $(-c_1, c_2)$. The second relationship comes from $g'(z)$ is an increasing function. For the third relationship, from (20), $g'(z_m) = az_m + b$. With the property that $g'(z)$ is increasing, we have

$$g'(z_m) \begin{cases} \geq 0 & \text{if } z_m \geq -b/a, \\ \leq 0 & \text{if } z_m \leq -b/a. \end{cases}$$

A.6 Proof of Theorem 6

We consider the analysis in Luo and Tseng (1992), which studies coordinate descent methods for problems in the following form:

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & g(E\boldsymbol{\alpha}) + \boldsymbol{b}^T\boldsymbol{\alpha} \\ \text{subject to} \quad & L_i \leq \boldsymbol{\alpha}_i \leq U_i, \end{aligned} \tag{67}$$

where g is a proper closed convex function, E is a constant matrix and $L_i \in [-\infty, \infty)$, $U_i \in (-\infty, \infty]$ are lower/upper bounds. They establish the linear convergence of the coordinate descent method if (67) satisfies the following conditions:

1. E has no zero column.
2. The set of optimal solutions for (67), denoted by A^*, is nonempty.
3. The domain of g is open, and g is strictly convex and twice continuously differentiable on its domain.
4. $\nabla^2 g(E\boldsymbol{\alpha}^*)$ is positive definite for all $\boldsymbol{\alpha}^* \in A^*$.

We explain that dual LR satisfies all the above conditions. Define E as an $(n + l) \times l$ matrix

$$E \equiv \begin{bmatrix} y_1\boldsymbol{x}_1, \ldots, y_l\boldsymbol{x}_l \\ I_l \end{bmatrix}, \tag{68}$$

where I_l is the identity matrix. Let g be the following function:

$$g(\begin{bmatrix} \boldsymbol{w} \\ \boldsymbol{\beta} \end{bmatrix}) \equiv \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + \sum_{i=1}^{l} \beta_i \log \beta_i + (C - \beta_i)\log(C - \beta_i), \tag{69}$$

where $(\boldsymbol{w}, \boldsymbol{\beta}) \in$ an open domain $R^n \times (0, C)^l$, and $\boldsymbol{b} = \mathbf{0}$, $L_i = 0,\ U_i = C, \forall i$. Then $D^{\mathsf{LR}}(\boldsymbol{\alpha}) = g(E\boldsymbol{\alpha}) + \boldsymbol{b}^T\boldsymbol{\alpha}$ and (3) is the same as (67). Obviously E contains no zero column. For the set of optimal solutions, the unique minimum $\boldsymbol{\alpha}^*$ exists by Theorem 1 and satisfies $0 < \alpha_i^* < C,\ \forall i$. The function g is closed because it is twice continuously differentiable on its open domain. The matrix $\nabla^2 g(\begin{bmatrix} \boldsymbol{w} \\ \boldsymbol{\beta} \end{bmatrix})$ is diagonal and has positive entries:

$$\nabla^2_{ii} g(\begin{bmatrix} \boldsymbol{w} \\ \boldsymbol{\beta} \end{bmatrix}) = \begin{cases} 1 & \text{if } i = 1, \ldots, n, \\ \frac{C}{\beta_j(C - \beta_j)} & \text{if } i = n + j,\ j = 1, \ldots, l. \end{cases}$$

Hence g is strictly convex and $\nabla^2 g(E\boldsymbol{\alpha}^*)$ is positive definite. All conditions are satisfied and the linear convergence is obtained.

A.7 The Derivation of Dual ME

For convenience, we define some notation:

- $l =$ the number of unique x_i,
- $\tilde{\mathcal{P}}_i = \tilde{\mathcal{P}}(x_i)$, $\boldsymbol{f}_{iy} = \boldsymbol{f}(x_i, y)$, and

– $\tilde{\boldsymbol{f}} = \sum_{i,y} \tilde{\mathcal{P}}(x_i, y)\boldsymbol{f}(x_i, y)$.

The primal ME problem in (5) can be written as the following equality-constrained form:

$$\begin{aligned} \min_{\boldsymbol{w},\boldsymbol{\xi}} \quad & \frac{1}{2\sigma^2}\boldsymbol{w}^T\boldsymbol{w} + \sum_i \tilde{\mathcal{P}}_i \log \sum_y \exp(\xi_{iy}) - \boldsymbol{w}^T\tilde{\boldsymbol{f}} \\ \text{subject to} \quad & \xi_{iy} = \boldsymbol{w}^T\boldsymbol{f}(x_i, y)\ \forall y \in Y, i = 1, \ldots, l. \end{aligned} \tag{70}$$

The Lagrangian for (70) is:

$$\begin{aligned} L(\boldsymbol{w}, \boldsymbol{\xi}, \boldsymbol{\alpha}) &= \frac{1}{2\sigma^2}\boldsymbol{w}^T\boldsymbol{w} + \sum_i \tilde{\mathcal{P}}_i \log \sum_y \exp(\xi_{iy}) - \boldsymbol{w}^T\tilde{\boldsymbol{f}} - \sum_i \sum_y \alpha_{iy}(\xi_{iy} - \boldsymbol{w}^T\boldsymbol{f}_{iy}) \\ &= L^*(\boldsymbol{w}, \boldsymbol{\alpha}) + \sum_i L_i(\boldsymbol{\xi}_i, \bar{\boldsymbol{\alpha}}_i), \end{aligned}$$

where

$$\begin{aligned} L^*(\boldsymbol{w}, \boldsymbol{\alpha}) &\equiv \frac{1}{2\sigma^2}\boldsymbol{w}^T\boldsymbol{w} + \sum_i \sum_y \alpha_{iy}\boldsymbol{w}^T\boldsymbol{f}_{iy} - \boldsymbol{w}^T\tilde{\boldsymbol{f}}, \text{ and} \\ L_i(\boldsymbol{\xi}_i, \bar{\boldsymbol{\alpha}}_i) &\equiv \tilde{\mathcal{P}}_i \log \sum_y \exp(\xi_{iy}) - \sum_y \alpha_{iy}\xi_{iy},\ i = 1, \ldots, l. \end{aligned}$$

The dual problem is

$$\max_{\boldsymbol{\alpha}} \inf_{\boldsymbol{w},\boldsymbol{\xi}} L(\boldsymbol{w}, \boldsymbol{\xi}, \boldsymbol{\alpha}) = \max_{\boldsymbol{\alpha}} \left(\inf_{\boldsymbol{w}} L^*(\boldsymbol{w}, \boldsymbol{\alpha}) + \sum_i \inf_{\boldsymbol{\xi}_i} L_i(\boldsymbol{\xi}_i, \bar{\boldsymbol{\alpha}}_i) \right). \tag{71}$$

For $\inf_{\boldsymbol{w}} L^*(\boldsymbol{w}, \boldsymbol{\alpha})$, the minimum is obtained by

$$\nabla_{\boldsymbol{w}} L^*(\boldsymbol{w}, \boldsymbol{\alpha}) = \frac{1}{\sigma^2}\boldsymbol{w} + \sum_i \sum_y \alpha_{iy}\boldsymbol{f}_{iy} - \tilde{\boldsymbol{f}} = \boldsymbol{0}$$

By representing the minimum as a function of $\boldsymbol{\alpha}$, we have

$$\inf_{\boldsymbol{w}} L^*(\boldsymbol{w}, \boldsymbol{\alpha}) = -\frac{1}{2\sigma^2}\boldsymbol{w}(\boldsymbol{\alpha})^T\boldsymbol{w}(\boldsymbol{\alpha}), \text{ where } \boldsymbol{w}(\boldsymbol{\alpha}) = \sigma^2 \left(\tilde{\boldsymbol{f}} - \sum_i \sum_y \alpha_{iy}\boldsymbol{f}_{iy} \right). \tag{72}$$

To minimize $L_i(\boldsymbol{\xi}_i, \bar{\boldsymbol{\alpha}}_i)$, we check several cases depending on the value of $\bar{\boldsymbol{\alpha}}_i$. The first case considers $\bar{\boldsymbol{\alpha}}_i$ satisfying

$$\bar{\boldsymbol{\alpha}}_i \geq \boldsymbol{0} \text{ and } \sum_y \alpha_{iy} = \tilde{\mathcal{P}}_i. \tag{73}$$

Let $F_i \equiv \{y \mid \alpha_{iy} > 0\}$.

$$\begin{aligned} \inf_{\boldsymbol{\xi}_i} L_i(\boldsymbol{\xi}_i, \bar{\boldsymbol{\alpha}}_i) &= \inf_{\xi_{iy}: y \in F_i} \left(\left(\inf_{\xi_{iy}: y \notin F_i} \tilde{\mathcal{P}}_i \log \sum_y \exp(\xi_{iy}) \right) - \sum_{y \in F_i} \alpha_{iy}\xi_{iy} \right) \\ &= \inf_{\xi_{iy}: y \in F_i} \left(\tilde{\mathcal{P}}_i \log \left(\sum_{y \in F_i} \exp(\xi_{iy}) + \sum_{y \notin F_i} \inf_{\xi_{iy}} \exp(\xi_{iy}) \right) - \sum_{y \in F_i} \alpha_{iy}\xi_{iy} \right) \\ &= \inf_{\xi_{iy}: y \in F_i} \left(\tilde{\mathcal{P}}_i \log \sum_{y \in F_i} \exp(\xi_{iy}) - \sum_{y \in F_i} \alpha_{iy}\xi_{iy} \right). \end{aligned} \tag{74}$$

The optimality condition implies any minimizer $\boldsymbol{\xi}_i^*$ satisfies that for all $y \in F_i$:

$$\nabla_{\xi_{iy}} L_i(\boldsymbol{\xi}_i^*) = -\alpha_{iy} + \frac{\tilde{\mathcal{P}}_i \exp(\xi_{iy}^*)}{\sum_{y' \in F_i} \exp(\xi_{iy'}^*)} = 0. \tag{75}$$

Thus

$$\xi^*_{iy} = \log \alpha_{iy} + \log \sum_{y' \in F_i} \exp(\xi^*_{iy'}) - \log \tilde{\mathcal{P}}_i.$$

By embedding $\boldsymbol{\xi}^*_i$ into $L_i(\boldsymbol{\xi}_i, \bar{\boldsymbol{\alpha}}_i)$ and using (73), (74) becomes

$$\begin{aligned}
&\inf_{\boldsymbol{\xi}_i} L_i(\boldsymbol{\xi}_i, \bar{\boldsymbol{\alpha}}_i) \\
=&\tilde{\mathcal{P}}_i \log \sum_{y \in F_i} \exp(\xi^*_{iy}) - \sum_{y \in F_i} \left(\alpha_{iy} \log \alpha_{iy} + \alpha_{iy} \log \sum_{y' \in F_i} \exp(\xi^*_{iy'}) - \alpha_{iy} \log \tilde{\mathcal{P}}_i\right) \\
=&- \sum_{y \in F_i} \alpha_{iy} \log \alpha_{iy} + \tilde{\mathcal{P}}_i \log \tilde{\mathcal{P}}_i.
\end{aligned} \tag{76}$$

If $\bar{\boldsymbol{\alpha}}_i$ does not satisfy (73), then either

$$\text{there is } \alpha_{iy'} < 0 \quad \text{or} \quad \sum_y \alpha_{iy} \neq \tilde{\mathcal{P}}_i.$$

If there is $\alpha_{iy'} < 0$, we consider a point $\boldsymbol{\xi}_i$ with $\xi_{iy} = \epsilon$ if $y = y'$ and 0 otherwise. Then,

$$\inf_{\boldsymbol{\xi}_i} L_i(\boldsymbol{\xi}_i, \bar{\boldsymbol{\alpha}}_i) \leq \lim_{\epsilon \to -\infty} \left(\tilde{\mathcal{P}}_i \log(|Y| - 1 + \exp(\epsilon)) - \alpha_{iy'}\epsilon\right) = -\infty. \tag{77}$$

If $\sum_i \alpha_{iy} \neq \tilde{\mathcal{P}}_i$, we consider $\xi_{iy} = \epsilon, \forall y$ to obtain

$$\begin{aligned}
\inf_{\boldsymbol{\xi}_i} L_i(\boldsymbol{\xi}_i, \bar{\boldsymbol{\alpha}}_i) &\leq \inf_{\epsilon} \left(\tilde{\mathcal{P}}_i \log(|Y| \exp(\epsilon)) - \epsilon \sum_y \alpha_{iy}\right) \\
&= \tilde{\mathcal{P}}_i \log |Y| + \inf_{\epsilon} \epsilon \left(\tilde{\mathcal{P}}_i - \sum_y \alpha_{iy}\right) = -\infty.
\end{aligned} \tag{78}$$

Combining (72), (76), (77) and (78),

$$\begin{aligned}
&\inf_{\boldsymbol{w}, \boldsymbol{\xi}} L(\boldsymbol{w}, \boldsymbol{\xi}, \boldsymbol{\alpha}) \qquad (79) \\
=&\begin{cases}
-\frac{1}{2\sigma^2} \boldsymbol{w}(\boldsymbol{\alpha})^T \boldsymbol{w}(\boldsymbol{\alpha}) - \sum_i \left(\sum_{y: \alpha_{iy} > 0} \alpha_{iy} \log \alpha_{iy} + \tilde{\mathcal{P}}_i \log \tilde{\mathcal{P}}_i\right) & \text{if } \sum_y \alpha_{iy} = \tilde{\mathcal{P}}_i \ \forall i, \boldsymbol{\alpha} \geq \mathbf{0}, \\
-\infty & \text{otherwise.}
\end{cases}
\end{aligned}$$

As the dual problem defined in (71) maximizes the value (79) by adjusting $\boldsymbol{\alpha}$, we will not consider the situation with the value $-\infty$. Then the dual problem can be written as (38).

A.8 Proof of Theorem 7

By defining $0 \log 0 = 0$, $D^{\mathsf{ME}}(\boldsymbol{\alpha})$ is a continuous function on a closed set. Hence a minimum exists. We first show the interior property. If $\tilde{\mathcal{P}}(x_i) = 0$, then $\alpha^*_{iy} = 0$ follows from constraints of (38). If $\tilde{\mathcal{P}}(x_i) > 0$, we prove the result by contradiction. If there exists $\alpha^*_{iy_1} = 0$, then we can find another $\alpha^*_{iy_2} > 0$ due to the constraint $\sum_y \alpha^*_{iy} = \tilde{\mathcal{P}}_i$. We consider a problem by fixing all variables except α_{iy_1} and α_{iy_2}.

$$\begin{aligned}
\min_z \quad & g(z) = D^{\mathsf{ME}}(\bar{\boldsymbol{\alpha}}_1, \ldots, \bar{\boldsymbol{\alpha}}_i + (\boldsymbol{e}_{y_1} - \boldsymbol{e}_{y_2})z, \ldots, \bar{\boldsymbol{\alpha}}_l) \\
& = z \log z + (\alpha^*_{iy_2} - z) \log(\alpha^*_{iy_2} - z) + \frac{a}{2} z^2 + bz + \text{ constant} \\
\text{subject to} \quad & 0 \leq z \leq \alpha^*_{iy_2},
\end{aligned}$$

where $\boldsymbol{e}_{y_1}$ and $\boldsymbol{e}_{y_2}$ are indicator vectors,

$$a \equiv \sigma^2 \left(K^i_{y_1 y_1} + K^i_{y_2 y_2} - 2K^i_{y_1 y_2} \right) \text{ and } \quad b \equiv -\boldsymbol{w}^T(\boldsymbol{\alpha}^*) \left(\boldsymbol{f}(x_i, y_1) - \boldsymbol{f}(x_i, y_2) \right).$$

By Lemma 1, there is $z^* \in (0, \alpha^*_{iy_2})$ such that $g(z^*) < g(0) = D^{\mathsf{ME}}(\boldsymbol{\alpha}^*)$, which contradicts the fact that $\boldsymbol{\alpha}^*$ is the minimum. Therefore, $\alpha^*_{iy} > 0 \ \forall y$. The constraints in (38) then imply $\alpha^*_{iy} < \tilde{\mathcal{P}}(x_i) \ \forall y$, so $\alpha^*_{iy} \in (0, \tilde{\mathcal{P}}(x_i)) \ \forall i, y$.

We then show the uniqueness by the strict convexity of $D^{\mathsf{ME}}(\boldsymbol{\alpha})$ over $(0, \infty)^{l|Y|}$. $D^{\mathsf{ME}}(\boldsymbol{\alpha})$ can be decomposed into two parts. The first part is

$$\frac{1}{2\sigma^2} \boldsymbol{w}(\boldsymbol{\alpha})^T \boldsymbol{w}(\boldsymbol{\alpha}) = \frac{1}{2\sigma^2} \|\tilde{\boldsymbol{f}} - \mathcal{F}\boldsymbol{\alpha}\|^2, \tag{80}$$

where $\mathcal{F}$ is a $n \times l|Y|$ matrix and each column is $\boldsymbol{f}(x_i, y)$. The Hessian of (80) is a positive semi-definite matrix $\mathcal{F}^T\mathcal{F}$. The Hessian of the second part is a diagonal matrix with positive elements $1/\alpha_{iy} \ \forall i, y$. Therefore, $D^{\mathsf{ME}}(\boldsymbol{\alpha})$ is strictly convex for all interior $\boldsymbol{\alpha}$, so the uniqueness is obtained.

A.9 Proof of Theorem 9

We apply Proposition 2.7.1 by Bertsekas (1999), which gives the convergence of coordinate descent methods for the following problem:

$$\begin{aligned} \min \quad & D(\boldsymbol{\alpha}) \\ \text{subject to} \quad & \boldsymbol{\alpha} \in A_1 \times \cdots \times A_l, \end{aligned} \tag{81}$$

where A_i is a closed convex set. Sequentially a block of variables over A_i is updated and it is required that the minimum of each sub-problem is uniquely attained.

Problem (38) is in the form of (81) as we can define the following closed and convex set:

$$A_i \equiv \{\bar{\boldsymbol{\alpha}}_i \in [0, \tilde{\mathcal{P}}(x_i)]^{|Y|} \mid \boldsymbol{e}^T \bar{\boldsymbol{\alpha}}_i = \tilde{\mathcal{P}}(x_i)\},$$

Moreover, a proof similar to Theorem 7 shows that for each sub-problem (42), the minimum is uniquely attained. Therefore, Algorithm 7 converges.