Machine Learning, 46, 91–129, 2002
© 2002 Kluwer Academic Publishers. Manufactured in The Netherlands.

On the Dual Formulation of Regularized Linear Systems with Convex Risks

TONG ZHANG tzhang@watson.ibm.com
Mathematical Sciences Department, IBM T.J. Watson Research Center, Yorktown Heights, NY 10598 USA

Editor: Nello Cristianini

Abstract. In this paper, we study a general formulation of linear prediction algorithms including a number of known methods as special cases. We describe a convex duality for this class of methods and propose numerical algorithms to solve the derived dual learning problem. We show that the dual formulation is closely related to online learning algorithms. Furthermore, by using this duality, we show that new learning methods can be obtained. Numerical examples will be given to illustrate various aspects of the newly proposed algorithms.

Keywords: regulation, linear model, convex duality, support vector machine, logistic regression, augmented Lagrangian

1. Introduction

We consider linear prediction problems such as linear regression and classification. Although nonlinear models such as neural networks have been extensively studied in the past, there is renewed interest in linear prediction models, which include recently popularized kernel methods such as Gaussian processes and support vector machines (Cristianini & Shawe-Taylor, 2000).

From the theoretical point of view, this recent popularization of linear prediction methods is related to their theoretical tractability. In particular, there has been a substantial amount of work in generalization performance analysis for various linear function classes, which provides valuable insights into theoretical properties of linear models. On the other hand, nonlinear methods are often difficult to analyze.

From a more practical point of view, linear models can be used to learn nonlinear decisions, by simply using pre-determined nonlinear functions as features. One intelligent way to include such nonlinear functions as features is through the so-called kernel representation, such as in Gaussian processes or support vector machines. This type of kernel methods rely on a form of convex duality, which converts a linear model in the original (possibly infinite dimensional) "feature" space into a dual learning model in the corresponding (finite dimensional) dual "sample" space.

However, the dual representation in a kernel method requires a very specific form of linear model in the original feature space. A natural question to ask is whether it is possible to extend this convex duality, so that it is applicable to a more general family of linear models. This paper provides an affirmative answer to this question. In order to obtain such an extension, we introduce a general form of regularized linear prediction models with convex risks. These models include support vector machines and Gaussian processes as special cases. We then derive a dual representation of the proposed formulation, and study numerical algorithms to solve the dual learning problem. We show that this dual representation provides useful insights into existing schemes. In addition, new learning algorithms can be obtained from the dual formulation.

The paper is organized as follows. In Section 2, we introduce a general linear prediction model, motived from examples. The corresponding dual formulation will be obtained in Section 3. We propose a relaxation algorithm in Section 4 to solve the dual learning problem. In Section 5, we give a number of examples, and show that novel learning algorithms can be obtained from the dual formulation. In Section 6, Numerical experiments are used to illustrate various aspects of the newly proposed algorithms. Section 7 summarizes the paper.

2. Regularized generalized linear model

Assume we have a set of input vectors $x_1, \ldots, x_n$, with corresponding desired output variables $y_1, \ldots, y_n$. For a generalized linear model (McCullagh & Neldor, 1989), one is interested in obtaining a weight vector w such that $y \approx \phi(w^T x)$ for all future input data x, where ϕ is called a link function. The degree of such an approximation can be measured by a *loss function* $L(\phi(w^T x), y)$. A practical method to compute a weight vector $\hat{w}$ from the data is to find the minimum of the *empirical expectation* of the loss: $\hat{w} = \arg\min_w \frac{1}{n} \sum_{i=1}^n L(\phi(w^T x_i), y_i)$. For computational purposes, the loss function is often matched with the link function ϕ so that $L(\phi(w^T x_i), y_i)$ is convex (Helmbold, Kivinen, & Warmuth, 1999; McCullagh & Nelder, 1989). In general, one can assume that the weight $\hat{w}$ is obtained by

$$\hat{w} = \arg\min_w \frac{1}{n} \sum_{i=1}^n f(w^T x_i, y_i), \tag{1}$$

where $f(a, b)$ is a convex function of a.

In the literature, one often encounters a more general type of linear functional: $w^T x + b$, where b is called *bias*. However, one can easily convert this formulation into one in which b is zero. This is achieved by letting $\tilde{x} = [x, 1]$, and $\tilde{w} = [w, b]$: $\tilde{w}^T \tilde{x} = w^T x + b$. Therefore unless otherwise indicated, we assume a linear form with $b = 0$ throughout this paper.

In practice, one often encounters a situation that the dimension d of the input-vector space is larger than the training sample size. In this case, the original problem in (1) is singular. This is because there are infinitely many solutions of w such that $w^T x_i = \hat{w}^T x_i$ for all i, which implies that they have the same expected loss. To eliminate this problem of uncertainty, one may consider the following alternative formulation for regression problems:

$$\hat{w} = \arg\min_w \frac{1}{2} w^2, \quad \text{s.t. } w^T x_i = y_i \quad \text{for } i = 1, \ldots, n.$$

Similarly, one can consider the following alternative formulation for binary-classification problems with labels of ± 1:

$$\hat{w} = \arg\min_w \frac{1}{2} w^2, \quad \text{s.t. } w^T x_i y_i \geq 1, \quad \text{for } i = 1, \ldots, n.$$

In general, the above two formulations are quite sensitive to outliers since a single modification of any data point (x_i, y_i) can violate the equality (or inequality) constraint, which causes a large change in the estimated parameter $\hat{w}$. Another problem is that these constraints may become un-satisfiable. For example, this is likely to happen when the dimension d is smaller than the sample size n, The above mentioned problems can be remedied by using a penalty type regularization method:

$$\hat{w} = \arg\min_w \left[\frac{1}{n} \sum_{i=1}^{n} f(w^T x_i, y_i) + \frac{\lambda}{2} w^2 \right],$$

where for regression, $f(a, b) = (a - b)^2$ (ridge regression (Hoerl & Kennard, 1970)), and for classification, $f(a, b) = 1 - ab$ if $ab \le 1$ and $f(a, b) = 0$ otherwise (soft-margin SVM (Vapnik, 1998)).

The above formulations use square regularization conditions. However, in practice, other types of regularization are also useful. For example, in the maximum entropy framework for density estimation, one seeks a density vector $\hat{w}$ so that it is consistent with the data, and the relative entropy is maximized:

$$\hat{w} = \arg\max_w \sum_{j=1}^{d} -w_j \ln \frac{w_j}{\mu_j} \quad \text{s.t. } w^T x_i = y_i \quad \text{for } i = 1, \ldots, n,$$

where μ is a prior typically chosen to be uniform. We assume that w represents a probability measure. That is: $\sum_{j=1}^{d} w_j = 1$ and $w_j \ge 0$ for $j = 1, \ldots, d$. The classification version of maximum entropy has also been suggested (Jaakkola, Meila, & Jebara, 2000). As demonstrated in Section 5, it is related to Winnow online algorithms (Littlestone, 1988).

Another interesting regularization condition is the 1-norm regularization used in *basis pursuit* (Chen, Donoho, & Saunders, 1999), which in general can lead to a sparse weight vector $\hat{w}$ (thus automatic feature selection):

$$\hat{w} = \arg\min_w \|w\|_1, \quad \text{s.t. } w^T x_i = y_i \quad \text{for } i = 1, \ldots, n.$$

The 1-norm regularization condition has some interesting learning properties (Barron, 1993; Bartlett, 1998), and has also been suggested for classification (Mangasarian, 1999).

In general, all of the above mentioned methods can be put into a penalty type regularized form. Therefore in this paper, we consider the following system with convex loss that can result from a regularized version of the generalized linear model in (1):

$$\hat{w} = \arg\min_w \left[\frac{1}{n} \sum_{i=1}^{n} f(w^T x_i, y_i) + \lambda g(w) \right]. \tag{2}$$

We assume that $f(a, b)$ is a convex function of a, and g is a convex function of w. $\lambda > 0$ is a regularization parameter.

3. Duality

3.1. Duality in ridge regression

We would like to present a simple example to explain the concept of duality. Consider the square regularization $g(w) = \frac{1}{2}w^2$. Assume f is differentiable, then by differentiating (2) with respect to w, we see that at the optimal solution of (2), $\hat{w}$ has a representation of the form

$$\hat{w} = \frac{1}{\lambda n} \sum_{i=1}^{n} \hat{\alpha}_i x_i.$$

This representation leads to the idea of Reproducing Kernel Hilbert Space (RKHS) method (Wahba, 1999), and has recently been successfully applied to many learning problems including Gaussian processes and support vector machines (Cristianini & Shawe-Taylor, 2000).

Equation (2) is called primal formulation, which involves the primal variable w. $\hat{\alpha}$ is the dual variable. By using convex duality, it is possible to rewrite (2) in terms of $\hat{\alpha}$, which leads to the dual formulation of (2).

As an example, we consider the ridge regression problem

$$\hat{w} = \arg\min_{w} \left[\frac{1}{n} \sum_{i=1}^{n} (w^T x_i - y_i)^2 + \frac{\lambda}{2} w^2 \right].$$

By taking derivative with respect to w, and let $\hat{\alpha}_i = -2(w^T x_i - y_i)$, we obtain the following system of equations

$$\begin{aligned} \hat{\alpha}_i &= -2(\hat{w}^T x_i - y_i), \quad i = 1, \ldots, n, \\ \hat{w} &= \frac{1}{\lambda n} \sum_{i=1}^{n} \hat{\alpha}_i x_i. \end{aligned}$$

We can now eliminate $\hat{w}$ and obtain

$$\hat{\alpha}_i = -2\left(\frac{1}{\lambda n} \sum_{k=1}^{n} \hat{\alpha}_k x_j^T x_i - y_i \right), \quad i = 1, \ldots, n.$$

It is easy to verify that $\hat{\alpha}$ is the solution of the following dual optimization problem:

$$\hat{\alpha} = \arg\min_{\alpha} \left[\sum_{i=1}^{n} \frac{1}{n} \left(\frac{1}{4} \alpha_i^2 - \alpha_i y_i \right) + \frac{1}{2\lambda n^2} \sum_{i=1}^{n} \sum_{k=1}^{n} \alpha_i \alpha_k x_i^T x_k \right].$$

Since the dual formulation only requires the inner product $x_i^T x_k$, we may replace it by a symmetric positive (semi)-definite Kernel function $K(x_i, x_k)$, which leads to a form of Gaussian process (Cristianini & Shawe-Taylor, 2000).

For square regularization, a relatively general duality theorem was given in Jaakkola and Haussler (1999). They essentially extend the derivation presented in this section to include any smooth loss term f. Corresponding kernel methods can be obtained in this approach. Our goal is to present a much more general analysis, so that a dual representation of the general formulation (2) can be obtained, with general forms of loss term f and regularization term g.

3.2. A general dual representation

A general duality theory for (2) can be derived by using a procedure similar to what we have done in ridge regression. In summary, we differentiate with respect to w to obtain the first order condition at the optimal solution. An auxiliary variable α_i can then be associated with each data point x_i. The optimal solution $\hat{w}$ of the primal problem can be expressed as a function of the linear combination $\sum_{i=1}^{n} \alpha_i x_i$. $\hat{w}$ can then be eliminated to obtain a system of equations in α. Finally, we associate the solution of the system of equations to the solution of a dual optimization problem.

However, since the rigorous derivation requires the theory of convex duality, it is more convenient to directly introduce a dual variable, and then obtain the dual formulation from the convex duality point of view.

In order to be general, we take the standard convex analysis point of view described in Rockafeller (1970). We allow a convex function $p(u) : \mathcal{R}^d \to \mathcal{R}^+$ to take a value of $+\infty$, where $\mathcal{R}$ is the real line, and $\mathcal{R}^+$ denotes the extended real line $\mathcal{R} \cup \{+\infty\}$. However, we assume that convex functions do not achieve $-\infty$. We also assume that any convex function $p(u)$ in this paper contains at least one point u_0 such that $p(u_0) < +\infty$. Convex functions that satisfy these conditions are called *proper* convex functions. This definition is very general: virtually all practically interesting convex functions are proper. In this paper, we only consider *closed* convex functions. That is, $\forall u$, $p(u) = \lim_{\epsilon \to 0^+} \inf\{p(v) : \|v - u\| \le \epsilon\}$. This condition essentially means that the convex set above the graph of u: $\{(u, y) : y \ge p(u)\}$ is closed.

We also assume that (2) has a finite solution $\hat{w}$. However, we do not assume that the solution is unique.

We say a point is *feasible* with respect to a convex function p if $p(u) < +\infty$. From the computational point of view, a convex function p that takes values on the extended real line $\mathcal{R}^+$ can be regarded as a convex function that takes values on the real line $\mathcal{R}$, in the feasible domain $D_p = \{u : p(u) < \infty\}$. This domain D_p imposes a constraint on p.

Let $k(\cdot, b)$ be the dual transform of $f(\cdot, b)$ (see Appendix A):

$$k(v, b) = \sup_u (uv - f(u, b)),$$

which implies that $\forall i$:

$$f(w^T x_i, y_i) = \sup_{\alpha_i} (-k(-\alpha_i, y_i) - \alpha_i w^T x_i). \tag{3}$$

Using this formula, we can now introduce an auxiliary dual variable α, with component α_i for each data point x_i. Consider the following convex-concave function:

$$R(w, \alpha) = \frac{1}{n} \sum_{i=1}^{n} (-k(-\alpha_i, y_i) - \alpha_i w^T x_i) + \lambda g(w). \tag{4}$$

Using (3), the optimization problem (2) can be equivalently rewritten as

$$\hat{w} = \arg \inf_w \sup_\alpha R(w, \alpha). \tag{5}$$

It is valid to switch the order of $\inf_w$ and $\sup_\alpha$ in the above minimax convex-concave programming problem. A proof of this interchangeability, i.e. strong duality, is given in Appendix B. The proof shows that given $\hat{w}$ that is a solution of (5), there is a solution $\hat{\alpha}$ to the following problem

$$\hat{\alpha} = \arg \sup_\alpha \inf_w R(w, \alpha), \tag{6}$$

such that

$$\nabla g(\hat{w}) = \frac{1}{\lambda n} \sum_{i=1}^{n} \hat{\alpha}_i x_i, \tag{7}$$

where $\nabla g(\hat{w})$ denotes a subgradient of g at $\hat{w}$ (see Appendix A for definition). $\nabla g(\hat{w})$ becomes the gradient if g is differentiable at $\hat{w}$. At the fixed value $\hat{\alpha}$, $\hat{w}$ given in (7) minimizes (6).

To simplify notations, we now consider the dual transform of $g(\cdot)$:

$$h(v) = \sup_u (u^T v - g(u)),$$

where $h(\cdot)$ is also a convex function. By definition, (6) can be rewritten as the following dual formulation:

$$\hat{\alpha} = \arg \inf_\alpha \left[\frac{1}{n} \sum_{i=1}^{n} k(-\alpha_i, y_i) + \lambda h \left(\frac{1}{\lambda n} \sum_{i=1}^{n} \alpha_i x_i \right) \right]. \tag{8}$$

From (7) and Proposition A.1, we see that the optimal solution $\hat{w}$ to the primal problem (2) is given by

$$\hat{w} = \nabla h \left(\frac{1}{\lambda n} \sum_{i=1}^{n} \hat{\alpha}_i x_i \right). \tag{9}$$

We use ∇h to denote a subgradient of h at $\frac{1}{\lambda n} \sum_{i=1}^{n} \hat{\alpha}_i x_i$. As a comparison, we also have the following equation from Appendix B:

$$\hat{\alpha}_i = -f_1'(\hat{w}^T x_i, y_i) \ (i = 1, \ldots, n), \tag{10}$$

where f_1' denotes a subgradient of $f(a, b)$ with respect to a.

Since the role of w and α is symmetric in (4), the above derivation implies that if $\hat{\alpha}$ solves (8), then there is a subgradient ∇h such that (9) gives a solution of (2).

In the main body of this paper, we assume that h is differentiable for simplicity. This implies that the subgradient ∇h in (9) is unique. However, it is possible to handle the case that h is not differentiable every where. We leave the discussion of this situation in Appendix C.

4. Dual learning algorithm

We consider numerical algorithms for the dual learning problem (8). In this paper, we consider a special type of relaxation algorithm, which is called *Gauss-Seidel* method in numerical analysis. The algorithm cycles through components of the dual variable α, and optimizes one component at a time (while keeping others fixed). Since each dual component is associated with a data point, the algorithm is closely related to online learning algorithms in that it learns by looking at one data at a time.

We first consider the simple case that h is differentiable without any constraint. The Gauss-Seidel algorithm is given in Algorithm 1. We denote the dimension of w by d, and the j-th component of a data point x_i by x_{ij}.

Algorithm 1 (*Dual Gauss-Seidel*)

```
let $\alpha = \alpha^0$ and $v_j = \frac{1}{\lambda n} \sum_{i=1}^{n} \alpha_i^0 x_{ij}$for $j = 1, \ldots, d$
**for** $k = 1, 2, \ldots$
  **for** $i = 1, \ldots, n$
    find $\Delta\alpha_i$ by approximately minimizing
        $k(-\alpha_i - \Delta\alpha_i, y_i) + \lambda n h(v + \frac{1}{\lambda n}\Delta\alpha_i x_i)$        $(*)$
    update $v$: $v_j = v_j + \frac{1}{\lambda n}\Delta\alpha_i x_{ij}$      $(j = 1, \ldots, d)$
    update $\alpha$: $\alpha_i = \alpha_i + \Delta\alpha_i$
  **end**
**end**
let $w = \nabla h(v)$.
```

In Algorithm 1, the default choice of the initial value α^0 is zero. At each inner iteration i, one fixes all dual components α_k with $k \neq i$, and find an update $\Delta\alpha_i$ of α_i to reduce the dual objective function (8). If we ensure that the dual objective function is always reduced at each step, then the algorithm will converge.

The main attractive feature of Algorithm 1 is its simplicity. As we have mentioned earlier, it has a form that is similar to an online update algorithm. From the machine learning point of view, this implies that we are able to convert certain online algorithms into batch algorithms using the dual formulation. This point will be illustrated in Section 5.

A straight-forward method to solve $(*)$ approximately is gradient descent. In this paper, we are especially interested in this scheme since it yields a very simple form that is closely related to online updates (which usually can be regarded as a form of stochastic gradient descent). Furthermore, in gradient descent, different regularization conditions share the same update rule:

$$\frac{\Delta\alpha_i}{\lambda n} = -\eta_i\big(-k_1'(-\alpha_i, y_i) + w^T x_i\big), \tag{11}$$

where $w = \nabla h(v)$. $\eta_i > 0$ is an appropriately chosen learning rate.

Clearly, the only difference among algorithms with different regularization terms is the transfer function $w = \nabla h(v)$ given in (9). The form of gradient descent update in (11) remains unchanged.

Note that in some applications, k may contain a constraint. In this situation, we always update α_i so that the constraint is satisfied. The learning rate η_i can be selected in many ways. One way is to fix η_i at a pre-selected small value for all i. Another way is to choose η_i so that (11) corresponds to a form of Newton's method. In some cases, η_i can also be chosen to exactly minimize $(*)$. Examples will be given in Section 5.

If exact optimization is used in $(*)$, or if we appropriately choose sufficiently small η_i in (11), then α converges to the true optimal solution $\hat{\alpha}$ of (8). In addition, if the Hessian of (8) exists and is positive definite around $\hat{\alpha}$, then (8) can be locally approximated by a quadratic function. In this case, if we select η_i that is asymptotically less than twice the amount needed to optimize $(*)$ exactly, then locally around $\hat{\alpha}$, (11) can be regarded as a successive over (or under) relaxation method (SOR) (Golub & Van Loan, 1996) for a symmetric positive definite linear system. The convergence analysis of SOR implies that asymptotically, Algorithm 1 has a linear convergence rate. However, the rate of this linear convergence depends on the spectrum of the Hessian matrix of (8) at the optimal solution $\hat{\alpha}$, as well as η_i.

One may also consider more sophisticated numerical algorithms with asymptotically superlinear convergence rates, such as quasi-Newton or conjugate gradient (CG) methods (Fletcher, 1987). However, such methods do not have direct connections with online algorithms any more. From experiments in Section 5, we also observe that the generalization performance with the linear weight w obtained from Algorithm 1 quickly stabilizes after the first few iterations. This means that in practice, an asymptotically faster algorithm is not more attractive. Furthermore, in many interesting situations (such as in an SVM), k in (8) may contain constraints. It is relatively difficult to handle such constraints directly in a superlinear convergence optimization algorithm, unless additional Lagrangian multipliers are introduced. These additional Lagrangian multipliers introduce more complexity into numerical optimization, which may not be beneficial in many problems.

On the other hand, for simple problems such as dual ridge regression (or problems with sufficiently smooth dual objective functions in (8)), it is numerically preferable to use a preconditioned CG with a symmetrized version of Algorithm 1 as the preconditioner (Golub & Van Loan, 1996).

If h contains a constraint as discussed in Appendix C, the relationship (9) have to be replaced by (35). Furthermore, Algorithm 1 can fail in this case since we may not be able to modify any dual component α_i individually, so that the constraint is still satisfied. Modifications of Algorithm 1 are discussed in Appendix D.

5. Examples

We give example learning algorithms that can be obtained from the dual formulation (8). We first discuss impacts of different regularization conditions. We then combine them with different loss terms f to obtain various learning algorithms.

5.1. Some regularization conditions

Square regularization. One of the most important regularization conditions is the square penalty:

$$g(w) = \frac{1}{2} w^T K w, \tag{12}$$

where K is a symmetric positive definite operator. In many applications, one choose $K = I$—the identity matrix. The dual is given by

$$h(v) = \frac{1}{2} v^T K^{-1} v.$$

Therefore step $(*)$ in Algorithm 1 can be replaced by minimizing:

$$k(-\alpha_i - \Delta\alpha_i, y_i) + c_2 \Delta\alpha_i + \frac{c_1}{2} \Delta\alpha_i^2, \tag{13}$$

where

$$c_1 = \frac{1}{\lambda n} x_i^T K^{-1} x_i, \quad c_2 = x_i^T K^{-1} v = w^T x_i. \tag{14}$$

In a problem using regularization, the data dimension d can be very large. The computation of c_1 and c_2 can thus be time consuming. An interesting property of (13) is that no matter what the loss term f (and thus k) is, the data dependent computation is only in c_1 and c_2. Therefore, as long as c_1 and c_2 are obtained, the optimization of (13) can be considered as a small problem of constant size that is independent of the data dimension d. As a result, when d is large, we can afford to use more sophisticated nonlinear optimization methods in (13).

If we do not compute v (and w) explicitly, but rather express it as $\sum_{i=1}^n \frac{1}{\lambda n} \alpha_i x_i$, then $v^T x = \sum_{i=1}^n \frac{1}{\lambda n} \alpha_i x_i^T K^{-1} x$. In the computation of c_1 and c_2 in (13), we only need to evaluate inner products of the form $x^T K^{-1} y$. This observation implies that we can replace $x^T K^{-1} y$ with a symmetric positive definite kernel function $K(x, y)$, which results to a class of dual kernel methods. In this case, the computation of c_1 and c_2 in (13) are given by

$$c_1 = \frac{1}{\lambda n} K(x_i, x_i), \quad c_2 = \frac{1}{\lambda n} \sum_{k=1}^n \alpha_k K(x_i, x_k). \tag{15}$$

Specific examples of kernel methods can be found in Cristianini and Shawe-Taylor (2000), Jaakkola and Haussler (1999) and Vapnik (1998). This paper generalizes earlier works on kernel methods so that an arbitrary form of convex function $k(-\alpha_i, y_i)$ can be used.

Sparse regularization. We have mentioned that the 1-norm regularization condition $g(w) = \|w\|_1$ leads to a sparse $\hat{w}$. This phenomenon can be understood from the dual formulation. From Appendix E, we see that the dual h of g is $h(v) = 0$ with constraint $\max_j |v_j| \le 1$. If $|v_j| < 1$, then the j-th component of the subgradient $\nabla h(v)$ is zero. Let $\hat{v} = \frac{1}{\lambda n} \sum_{i=1}^n \hat{\alpha}_i x_i$ in (9), then the j-th component of $\hat{w}$ is exactly zero as long as $|\hat{v}_j| < 1$. This characteristics shows why the computed $\hat{w}$ is sparse in general.

From the computational point of view, the dual constraints $\max_j |v_j| \le 1$ require d Lagrangian multipliers, one for each component. To avoid the complication of dealing with these Lagrangian multipliers, we may include a quadratic term to g, and modify the sparse regularization condition as

$$g(w) = \epsilon \|w\|_1 + \frac{1}{2} w^2, \tag{16}$$

where $\epsilon > 0$ is a parameter to control the degree of sparsity. In this case,

$$h(v) = \sum_{j=1}^{d} h_j(v_j),$$

where

$$h_j(v_j) = \begin{cases} 0 & \text{if } |v_j| \le \epsilon, \\ \frac{1}{2}(|v_j| - \epsilon)^2 & \text{otherwise.} \end{cases}$$

Let $\hat{v} = \frac{1}{\lambda n} \sum_{i=1}^n \hat{\alpha}_i x_i$, then Eq. (9) becomes

$$\hat{w}_j = \begin{cases} 0 & \text{if } |\hat{v}_j| \le \epsilon, \\ \operatorname{sgn}(\hat{v}_j)(|\hat{v}_j| - \epsilon) & \text{otherwise.} \end{cases}$$

Clearly, a component of $\hat{w}$ is exactly zero as long as $|\hat{v}_j| \le \epsilon$. Therefore with this modified regularization term, the computed primal variable $\hat{w}$ is still sparse. In addition, the degree of sparsity can be controlled by adjusting ϵ.

Interestingly, the reason we provide for the potential sparsity of the primal estimate $\hat{w}$ is dual to the reason that a support vector machine formulation can lead to a sparse dual estimate $\hat{\alpha}$: observe that a support vector machine has a loss function $f(a, b)$ such that $f(a, b)$ is a constant in an interval of a. In such an interval, the subgradient $f_1'(a, b)$ becomes zero. By Eq. (10), we see that if $\hat{w}^T x_i$ belongs to the interval, then the dual component $\hat{\alpha}_i$ is exact zero.

Entropy regularization. Normalized entropy frequently occurs in distribution related estimation problems. Such a regularization condition can be written in general as:

$$g(w) = \sum_{j=1}^{d} w_j \ln \frac{w_j}{\mu_j}, \quad \text{s.t. } w_j \ge 0, \sum_{j=1}^{d} w_j = A,$$

where $\mu_j > 0$ gives the prior. $A > 0$ is a normalization constant given by $A = \sum_{j=1}^{d} \mu_j$. From Appendix E, the dual function is

$$h(v) = A \ln \sum_{j=1}^{d} \frac{\mu_j}{A} e^{v_j},$$

which leads to the following relationship in (9):

$$\hat{w}_j = A \frac{\mu_j \exp\left(\frac{1}{\lambda n} \sum_{i=1}^{n} \hat{\alpha}_i x_{ij}\right)}{\sum_{k=1}^{d} \mu_k \exp\left(\frac{1}{\lambda n} \sum_{i=1}^{n} \hat{\alpha}_i x_{ik}\right)}.$$

This formula corresponds to the exponentiated gradient (EG) transfer function in Kivinen and Warmuth (1997). Using this transfer function, we see that the gradient descent update (11) for the dual variable α in Algorithm 1 can be regarded as an exponentiated gradient descent rule in the primal variable w. Therefore this relationship suggests a systematic method to convert EG online methods into the corresponding batch methods.

In the online learning literature, besides the normalized entropy condition, one often considers the following unnormalized entropy regularization, which leads to EGU rules (Kivinen & Marmuth, 1997):

$$g(w) = \sum_{j=1}^{d} w_j \ln \frac{w_j}{e\mu_j}, \quad w_j \geq 0. \tag{17}$$

Its dual is

$$h(v) = \sum_{j=1}^{d} \mu_j \exp(v_j).$$

Using this dual function, Eq. (9) gives an EGU transfer function:

$$\hat{w}_j = \mu_j \exp\left(\frac{1}{\lambda n} \sum_{i=1}^{n} \hat{\alpha}_i x_{ij}\right).$$

The gradient descent update (11) on α can thus be regarded as an EGU update on $\hat{w}_j$.

q-norm regularization. We use a regularization condition of the form:

$$g(w) = \frac{1}{q'} \|w\|_q^{q'},$$

where $q, q' \in (1, +\infty)$. From Appendix E, we obtain its dual transform:

$$h(v) = \frac{1}{p'} \|v\|_p^{p'},$$

where $1/p + 1/q = 1$, and $1/p' + 1/q' = 1$. The transfer function in (9) becomes

$$\nabla h(v)_j = \operatorname{sgn}(v_j)|v_j|^{p-1}\|v\|_p^{p'-p},$$

where $\nabla h(v)_j$ denotes the j-th component of $\nabla h(v)$. This formulation is related to the so-called p-norm algorithm in online learning (Gentile & Littlestone, 1999; Grove, Littlestone, & Schuurmans, 1997), where the transfer function has a form of

$$w_j = \operatorname{sgn}(v_j)|v_j|^{p-1}\|v\|_p^{2-p}.$$

This particular transfer function clearly corresponds to the case $p' = 2$.

In addition to their relationship with online learning algorithms, this class of regularization conditions can have many practical applications. For example, if the data are bounded in p-norm, then it is reasonable to regularize using $\|w\|_q$ since the linear prediction $w^T x$ remains bounded by the Hölder's inequality. Furthermore, in many engineering applications, the solution w may be bounded. This requires an infinity-norm regularization, which can be approximated by a q-norm regularization with a large q. In some robust formulations, one may also prefer a regularization condition $\|w\|_q$ with $q < 2$.

5.2. *Some learning formulations*

Regression. We consider a generalized form of Vapnik's ϵ-sensitive loss with $\epsilon > 0$:

$$f(a, b) = \begin{cases} 0 & \text{if } |a - b| \le \epsilon, \\ \frac{1}{p}(|a - b| - \epsilon)^p & \text{otherwise,} \end{cases} \tag{18}$$

where $1 \le p < +\infty$. Its dual is

$$k(-\alpha_i, y_i) = \epsilon|\alpha_i| + \frac{1}{q}|\alpha_i|^q - y_i\alpha_i,$$

where $1/p + 1/q = 1$. If $\epsilon > 0$, then by (10), this formulation produces a sparse dual estimate.

In the general case with $p > 1$, the corresponding gradient descent in (11) becomes

$$\frac{\Delta\alpha_i}{\lambda n} = -\eta_i(\operatorname{sgn}(\alpha_i)(\epsilon + |\alpha_i|^{q-1}) + w^T x_i - y_i), \tag{19}$$

where $w = \nabla h(v)$.

If $p = 1$, $k(\cdot, \cdot)$ becomes a constrained convex function:

$$k(-\alpha_i, y_i) = \epsilon|\alpha_i| - y_i\alpha_i, \quad \alpha_i \in [-1, 1].$$

The corresponding gradient descent in (11) becomes

$$\frac{\Delta\alpha_i}{\lambda n} = -\eta_i(\operatorname{sgn}(\alpha_i)\epsilon + w^T x_i - y_i), \quad \alpha_i + \Delta\alpha_i \in [-1, 1], \tag{20}$$

where $w = \nabla h(v)$.

If $|w^T x_i - y_i| < \epsilon$, and w is close to the optimal solution $\hat{w}$, then the relationship (10) indicates that α_i should be (close to) zero. However, The right hand side of (19) or (20) may not be close to zero, which causes oscillation in the update $\Delta\alpha_i$. This problem can be avoided if we choose an update $|\Delta\alpha_i| \le |\alpha_i|$ when $|w^T x_i - y_i| < \epsilon$. Also if $\alpha_i = 0$ and $|w^T x_i - y_i| \ge \epsilon$, then $\mathrm{sgn}(\alpha_i)$ is not well-defined. We can set $\mathrm{sgn}(\alpha_i) = -\mathrm{sgn}(w^T x_i - y_i)$ in this situation.

If the square regularization (12) is employed, and $p = 1$ or $p = 2$ in (18), then the exact solution of $(*)$ can be obtained.

When $p = 1$, (13) becomes

$$\inf_{\Delta\alpha_i} \left[\epsilon|\alpha_i + \Delta\alpha_i| - y_i\Delta\alpha_i + c_2\Delta\alpha_i + \frac{c_1}{2}\Delta\alpha_i^2 \right] \quad \text{s.t.} \quad \alpha_i + \Delta\alpha_i \in [-1, 1].$$

Let

$$\Delta_1 = -\frac{\epsilon + c_2 - y_i}{c_1}, \quad \Delta_2 = -\frac{-\epsilon + c_2 - y_i}{c_1},$$

where c_1 and c_2 are given by (14) or (15). The exact solution of $(*)$ is

$$\Delta\alpha_i = \begin{cases} \min(\Delta_1, 1 - \alpha_i) & \text{if } \alpha_i + \Delta_1 \ge 0, \\ \max(\Delta_2, -1 - \alpha_i) & \text{if } \alpha_i + \Delta_2 \le 0, \\ -\alpha_i & \text{otherwise.} \end{cases}$$

When $p = 2$, (13) becomes

$$\inf_{\Delta\alpha_i} \left[\epsilon|\alpha_i + \Delta\alpha_i| + \frac{1}{2}(\alpha_i + \Delta\alpha_i)^2 - y_i\Delta\alpha_i + c_2\Delta\alpha_i + \frac{c_1}{2}\Delta\alpha_i^2 \right].$$

Let

$$\Delta_1 = -\frac{\epsilon + \alpha_i + c_2 - y_i}{c_1 + 1}, \quad \Delta_2 = -\frac{-\epsilon + \alpha_i + c_2 - y_i}{c_1 + 1},$$

where c_1 and c_2 are given by (14) or (15). The exact solution of $(*)$ is

$$\Delta\alpha_i = \begin{cases} \Delta_1 & \text{if } \alpha_i + \Delta_1 \ge 0, \\ \Delta_2 & \text{if } \alpha_i + \Delta_2 \le 0, \\ -\alpha_i & \text{otherwise.} \end{cases}$$

The case $p = 1$ corresponds to robust estimation. Its robustness is naturally explained in the dual formulation since $\alpha_i \in [0, 1]$ is bounded for each i. From the relationship $w = \nabla h(\frac{1}{\lambda n}\sum_{i=1}^n \alpha_i x_i)$, we see that the contribution of any single data point x_i in the summation cannot exceed $|\frac{1}{\lambda n} x_i|$. This means that a small percentage of outliers in the data do not have a significant impact on w.

Binary classification. In binary classification, $y_i = \pm 1$. We consider a general form of SVM style loss function ($1 \leq p < +\infty$):

$$f(a,b) = \begin{cases} 0 & \text{if } ab \geq 1, \\ \frac{1}{p}(1-ab)^p & \text{otherwise.} \end{cases} \tag{21}$$

Its dual is given by

$$k(-\alpha_i, y_i) = \frac{1}{q}|\alpha_i|^q - \alpha_i y_i \quad (\alpha_i y_i \geq 0),$$

where $1/p + 1/q = 1$.

For $p > 1$ with a general regularization transfer function ∇h, the corresponding gradient descent in (11) becomes

$$\frac{\Delta\alpha_i}{\lambda n} = -\eta_i(\operatorname{sgn}(\alpha_i)|\alpha_i|^{q-1} + w^T x_i - y_i), \quad (\alpha_i + \Delta\alpha_i)y_i \geq 0. \tag{22}$$

where $w = \nabla h(v)$.

When $p = 1$, the dual can be written equivalently as:

$$k(-\alpha_i, y_i) = -\alpha_i y_i \quad (\alpha_i y_i \in [0, 1]).$$

The corresponding gradient descent in (11) becomes

$$\frac{\Delta\alpha_i}{\lambda n} = -\eta_i(w^T x_i - y_i), \quad (\alpha_i + \Delta\alpha_i)y_i \in [0, 1], \tag{23}$$

where $w = \nabla h(v)$.

As a comparison, a family of online update algorithms for classification have been introduced in (Grove, Littlestone, & Schuurmans, 1997). This family of algorithms can be equivalently written as the following update at step $(*)$ in Algorithm 1:

$$\Delta\alpha_i = \begin{cases} 0 & \text{if } w^T x_i y_i > 0, \\ \eta y_i & \text{otherwise,} \end{cases} \tag{24}$$

where $w = \nabla h(v)$. $\eta > 0$ is a learning rate.

This online update does not correspond to a true gradient descent rule with respect to any dual formulation (8). However, it has a form that is very similar to the true gradient descent rule in (22) and (23). Conceptually, (22) and (23) can be regarded as different large margin versions of (24) using g-regularization.

From the online learning point of view, update rule (24) gives the Perceptron algorithm if we let $h(v) = \frac{1}{2}v^2$; it gives the Winnow family of online algorithms (Littlestone, 1988) if we let $\nabla h(v)$ take exponential forms; it gives the p-norm algorithm if h corresponds to the dual of a q-norm regularization term g.

Therefore, we can obtain regularized large margin versions of these online algorithms simply by choosing different regularization conditions in (22) and (23). For example, with entropy regularization, an additive update in α leads to an multiplicative update in w. This gives regularized Winnows. With q-norm regularization, we obtain large margin versions of a p-norm algorithm. Specifically, if we let $q = 2$, then we obtain large margin Perceptrons. In fact, this connection with the Perceptron update is already known. For example, see Platt (1999) and Cristianini and Shawe-Taylor (2000). Furthermore, we can use a transfer function $\nabla h(v)$ that vanishes in a region, which leads to a sparse representation on w.

In general, the above discussion shows that with the same gradient descent rule, we can modify the transfer function $w = \nabla h(v)$ to obtain different methods. Furthermore, the transfer function $w = \nabla h(v)$ can be chosen to enforce desired properties on w (such as sparsity, positivity, etc). Due to the simplicity of this approach, we may even conjecture that with a transfer function that does not correspond to the gradient of a convex function, the resulting method could still be interesting, although there is no convex duality theory to reconstruct the corresponding primal problem.

It is interesting to compare (22) and (23) to the corresponding update rules (20) and (20) for regression problem with $\epsilon = 0$. We observe that the only difference is the additional constraint $\alpha_i y_i \geq 0$ in classification. This observation can be applied to convert other dual regression formulations into classification formulations.

If we consider the square regularization (12) and let $p = 1$ or $p = 2$ in (21), then the exact solution of $(*)$ can be obtained.

When $p = 1$, (13) becomes

$$\inf_{\Delta\alpha_i} \left[-y_i \Delta\alpha_i + c_2 \Delta\alpha_i + \frac{c_1}{2} \Delta\alpha_i^2 \right] \quad \text{s.t. } (\alpha_i + \Delta\alpha_i) y_i \in [0, 1].$$

The exact solution of $(*)$ is

$$\Delta\alpha_i = y_i \max\left(\min\left(\alpha_i y_i - \frac{c_2 - y_i}{c_1} y_i, 1 \right), 0 \right) - \alpha_i, \tag{25}$$

where c_1 and c_2 are given by (14) or (15). In 1999, Platt also derived this update rule. However, his derivation employed Bregman's technique (Bregman, 1967) suitable only for some specific problems. In particular, the technique of convex duality which we have adopted in this paper generalizes Bregman's approach. This specific update formula (25) has also appeared in Jaakkola, Diekhans, and Haussler (2000).

When $p = 2$, (13) becomes

$$\inf_{\Delta\alpha_i} \left[\frac{1}{2} (\alpha_i + \Delta\alpha_i)^2 - y_i \Delta\alpha_i + c_2 \Delta\alpha_i + \frac{c_1}{2} \Delta\alpha_i^2 \right] \quad \text{s.t. } (\alpha_i + \Delta\alpha_i) y_i \geq 0.$$

The exact solution of $(*)$ is

$$\Delta\alpha_i = y_i \max\left(-\frac{\alpha_i + c_2 - y_i}{c_1 + 1} y_i, -\alpha_i y_i \right), \tag{26}$$

where c_1 and c_2 are given by (14) or (15).

Note that if $p = 1$, then $\alpha_i y_i \in [0, 1]$. If $p = 2$, α_i can be large. This difference means that the case $p = 1$ is more robust in the sense that an outlier will not have a significant impact on the primal estimate. However, for practical problems with few outliers, this difference is non-essential. In fact, we observe that in many problems, the $p = 2$ formulation is preferable.

Finally, we mention that the dual formulation can also be used to solve another popular binary classification method—logistic regression.

$$f(a, b) = \ln(1 + \exp(-ab)),$$

Its dual is

$$k(-\alpha_i, y_i) = \alpha_i y_i \ln(\alpha_i y_i) + (1 - \alpha_i y_i) \ln(1 - \alpha_i y_i) \quad \text{s.t. } \alpha_i y_i \in [0, 1].$$

It is interesting to observe that compared with the SVM style loss with $p = 1$ in (21), the dual of logistic-loss replaces the term $-\alpha_i y_i$ in the dual of SVM-loss by an entropy term. The $-\alpha_i y_i$ term favors $\alpha_i y_i \approx 1$, while the entropy term favors $\alpha_i y_i \approx 0.5$. Conceptually, the difference is quite small. Due to the constraint $\alpha_i y_i \in [0, 1]$, logistic regression is also robust to outliers.

We have seen that all classification loss-terms $f(a, b)$ studied in this section have duals $k(-\alpha_i, y_i)$ of the following shape: $k(-\alpha_i, y_i) = +\infty$ when $\alpha_i y_i < 0$; as $\alpha_i y_i$ increases, $k(-\alpha_i, y_i)$ first decreases at $\alpha_i y_i = 0$; it achieves the minimum at $\alpha_i y_i = a > 0$, and then increases. This means that a classification-loss favors a fixed positive value of $\alpha_i y_i = a$. The overall effect is to favor a linear weight of the form $w = \nabla h(\frac{a}{\lambda n} \sum_{i=1}^n x_i y_i)$, which can be regarded as a transformed centroid. This implies that the main difference among different classification loss-terms is how this transformed centroid is favored.

Un-regularized bias term. We have mentioned in Section 2 that one often encounters a linear model of the form $w^T x + b$. Although the bias b can be absorbed into the weight w by appending a constant 1 to x, it is typically un-regularized in the literature. Note that in the previous discussion, we essentially assume that b is also regularized. In our experience, this extra regularization does not introduce any practical difference for learning problems. However, it is still useful to consider the special case that b is un-regularized.

We replace $w^T x_i$ in (2) by $w^T x_i + b$. In this case, the regularization condition $g(w)$ is independent of b. The dual of $g(w)$ in the space $[w, b]$ has a form

$$\tilde{h}([v, v_b]) = \sup_{w,b} [w^T v + v_b b - g(w)].$$

Therefore if $v_b = 0$, $\tilde{h}([v, v_b]) = h(v)$ where $h(v)$ is the dual of $g(w)$; if $v_b \neq 0$, $\tilde{h}([v, v_b]) = +\infty$.

This means that the dual problem is still given by (8), but under an additional constraint

$$v_b = \frac{1}{\lambda n} \sum_{i=1}^n \alpha_i = 0.$$

From Appendix C and Appendix D, this constraint can be handled by solving the following augmented Lagrangian problem

$$\begin{aligned}\hat{\alpha} = \arg\inf_{\alpha} \Bigg[& \frac{1}{n}\sum_{i=1}^{n} k(-\alpha_i, y_i) + \lambda h\left(\frac{1}{\lambda n}\sum_{i=1}^{n}\alpha_i x_i\right) \\ & + s\sum_{i=1}^{n}\frac{1}{n}\alpha_i + \frac{1}{2\mu n}\left(\sum_{i=1}^{n}\alpha_i\right)^2\Bigg],\end{aligned} \tag{27}$$

where the Lagrangian parameter s is chosen so that $\sum_{i=1}^{n}\hat{\alpha}_i = 0$. The optimal primal solution $[\hat{w}, \hat{b}]$ is given by

$$\hat{w} = \nabla h\left(\frac{1}{\lambda n}\sum_{i=1}^{n}\hat{\alpha}_i x_i\right), \quad \hat{b} = s. \tag{28}$$

We may now use Algorithm 3 to solve (27). Similar to (11), the α-update in $(**)$ can be approximately solved using a gradient descent method:

$$\frac{\Delta\alpha_i}{\lambda n} = -\eta_i\left(-k_1'(-\alpha_i, y_i) + w^T x_i + s + \frac{1}{\mu}\sum_{k=1}^{n}\alpha_k\right), \tag{29}$$

where $w = \nabla h(v)$.

Compared with (11), the only difference we have is the extra two terms $s + \frac{1}{\mu}\sum_{i=1}^{n}\alpha_i$. The first term corresponds to the bias b. The second term favors a direction $\Delta\alpha_i$ that makes $|\sum_{i=1}^{n}\alpha_i|$ small.

A special but interesting case is the standard SVM formulation, where f is given by (21) with $p = 1$, and g is given by the square regularization (12). Similar to (25), we can obtain the exact solution to $(**)$ as

$$\Delta\alpha_i = y_i \max\left(\min\left(\alpha_i y_i - \frac{c_2 + s + \frac{1}{\mu}\sum_{i=1}^{n}\alpha_i - y_i}{c_1 + \frac{1}{\mu}} y_i, 1\right), 0\right) - \alpha_i, \tag{30}$$

where c_1 and c_2 are given by (14) or (15).

The s-update in Algorithm 3 is given by

$$s = s + \frac{\eta}{\mu}\sum_{i=1}^{n}\alpha_i. \tag{31}$$

Intuitively, this shifts the bias b towards the current estimate of the extra bias given by $\sum_{i=1}^{n}\alpha_i$. Due to this compensation, it has the effect of shrinking $\sum_{i=1}^{n}\alpha_i$ towards zero.

6. Experiments

In this section, we study various aspects of the proposed dual algorithms. Due to the generality of algorithms that can result from this formulation, it is not possible to study all specializations of (8) mentioned in this paper. Therefore, we only use a few example dual formulations to illustrate aspects of Algorithm 1 that are common to the general dual learning problem.

6.1. *Effect of different regularization conditions*

In the engineering literature, non-square regularization conditions are widely used. A specific choice of regularization condition reflects certain domain dependent data properties. For example, if sparse weight is required, then 1-norm regularization should be used. In machine learning, it is also known that different regularization should be employed in different situations. For example, the maximum entropy principle is used in density estimation. In online learning, if the data has many irrelevant attributes and the target linear separator is one-norm bounded, then the Winnow algorithm is preferred to the Perceptron algorithm.

The conclusion of this online learning analysis also holds in batch learning. In this section, we use an artificial binary classification problem to show that if the data are ∞-norm bounded, and the target hyperplane is sparse, then the regularized Winnow with entropy regularization is superior to the regularized Perceptron with 2-norm regularization. Note also that there is no obvious way to employ a kernel method that can enhance the simple 2-norm regularization. This example shows that in practice, different regularization conditions are suitable for different learning problems.

The data in this experiment are generated as follows. We select an input data dimension d, with $d = 500$ or $d = 5000$. The first 5 components of the target linear weight w are set to be ones; the 6th component is -1; and the remaining components are zeros. The bias b is -2. Data are generated as random vectors with each component randomly chosen to be either 0 or 1 with probability 0.5 each. Five percent of the data are given wrong labels (noise). The remaining data are given correct labels, but we remove data with margins that are less than 1. One thousand training and one thousand test data are generated.

Note that Winnow algorithms with entropy regularization require positive weights. A variant called balanced Winnow can be used to overcome this problem. We embed the input space into a higher dimensional space as: $\tilde{x} = [x, -x]$. This modification allows the positive weight Winnow algorithm for the augmented input $\tilde{x}$ to have the effect of both positive and negative weights for the original input x. Balanced Winnows are used throughout our experiments. As mentioned in Section 2, we append a constant 1 to each data point to compensate the effect of bias b, so that the decision rule can be written as $y \approx \mathrm{sgn}(w^T x)$ with zero bias.

We use UWin and NWin to denote the basic unnormalized and normalized Winnows respectively. LM-UWin and LM-NWin denote the corresponding large margin versions. The SVM style large margin Perceptron is denoted as LM-Perc.

We use 200 iterations over the training data for all algorithms. The initial values for the Winnows are set to be the priors: $\mu_j = 0.01$. Since our main purpose is to examine different

Table 1. Testset accuracy (in percentage) on the artificial dataset.

Dimension	Perceptron	LM-Perc	UWin	LM-UWin	NWin	LM-NWin
500	82.2	87.1	82.4	94.0	82.4	94.3
5000	67.9	69.8	69.7	87.4	69.7	88.6

regularization conditions, we chose the SVM style soft-margin loss in (21) with $p = 1$ for all regularized algorithms.

For online algorithms, we fix the learning rates at 0.01. For large margin Winnows, we use learning rates $\eta_i = 0.01$ in the gradient descent update (23). For (2-norm regularized) large margin Perceptron, we use the exact update given in (25).

Accuracies (in percentage) of different methods are listed in Table 1. For regularization methods, accuracies are reported with the optimal regularization parameters. The superiority of the regularized Winnows is obvious, especially for high dimensional data. Accuracies of regularized algorithms with different regularization parameters are plotted in figure 1. The behavior is very typical for regularized algorithms. In practice, the optimal regularization parameter can be found by cross-validation.

6.2. Convergence of the dual Gauss-Seidel algorithm

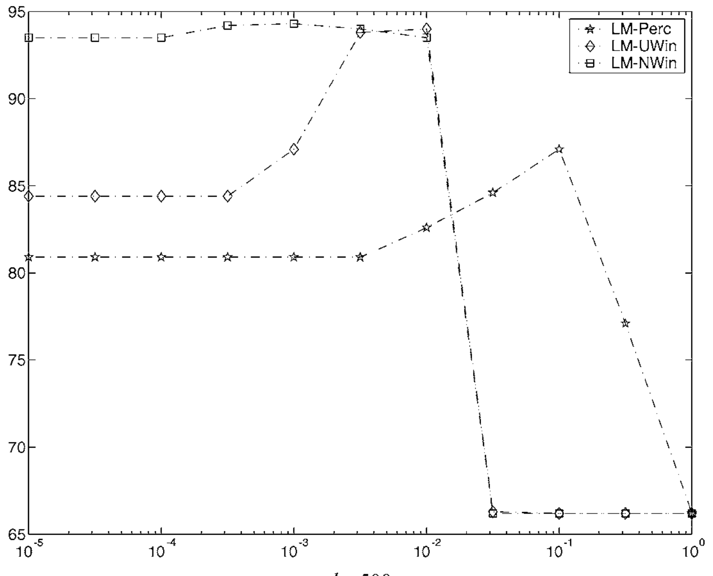
We study the convergence behavior of Algorithm 1. In this experiment, we use a more realistic dataset in text categorization, where support vector machines have been successfully applied.

In text categorization, it frequently happens that each category contains only a small percentage of documents. When this happens, the standard classification error measurement is often not very indicative, because one can get a low error rate simply by saying no item is in any class. The standard performance measures of a classification method are the precision and recall:

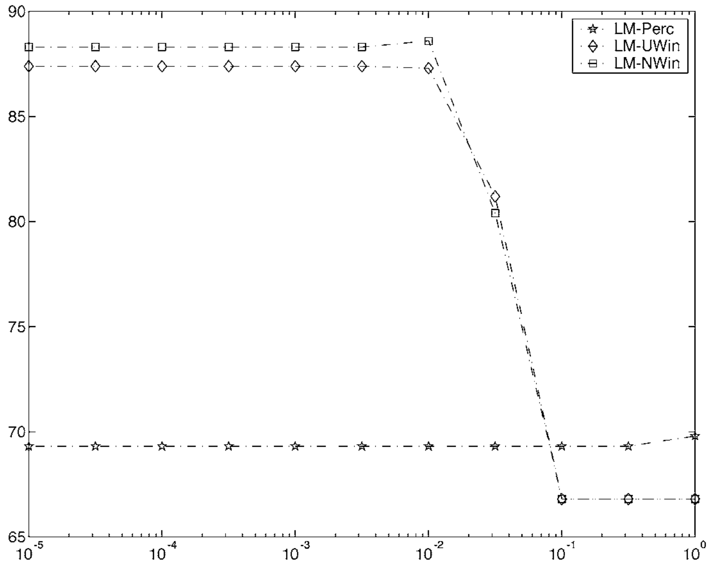
$$\text{precision} = \frac{\text{true positive}}{\text{true positive} + \text{false positive}} \times 100$$
$$\text{recall} = \frac{\text{true positive}}{\text{true positive} + \text{false negative}} \times 100$$

If a classification algorithm contains a parameter that can be adjusted to facilitate a trade-off between precision and recall, then one can compute the break-even point (BEP), where precision equals recall, as an evaluation criterion for the performance of a classification algorithm.

The standard data set for comparing text categorization algorithms is the Reuters set of news stories, publicly available at *http://www.research.att.com/~lewis/reuters21578.html*. We use Reuters-21578 with Mod-Apte split to partition the documents into training and validation sets. This dataset contains 118 non-empty classes (in addition, there are some



$d = 500$



$d = 5000$

Figure 1. Testset accuracy (in percentage) as a function of λ.

documents that are not categorized), with 9603 documents in the training set and 3299 documents in the test set.

The micro-averaged (that is, the precision and the recall are computed by using statistics summed over the confusion matrices of all classes) performances of the algorithms over

Table 2. Micro-averaged break-even performance on Reuters.

Algorithm	Perceptron	LM-Perc	UWin	LM-UWin	NWin	LM-NWin	Spars
BEP	78.5	85.7	79.2	87.0	79.2	87.1	86.7

all 118 classes are reported in Table 2. The results are comparable with the state of the art achieved on Reuters. Each BEP is computed by first adjusting the linear decision threshold individually for each category to obtain a BEP for the category, and then micro-averaged over all 118 classes. The regularization parameters are determined by cross-validation.

The "Spars" algorithm in Table 2 is a version of sparse regularization in (16) with $\epsilon = 0.1$. To be consistent with other algorithms, we still use the SVM style soft-margin loss in (21) with $p = 1$. This algorithm is solved by using the gradient descent update (23) with $\eta_i = 0.01$.

We use binary word occurrences as input features, and append a constant feature of 1. As explained before, the constant feature is used to offset the effect of the bias b. The step-up of each algorithm is the same as that for the artificial dataset in Section 6.1. Note that in this experiment, no feature selection or stop-word removal is used. Due to the presence of some irrelevant features, the Winnow family of algorithms has a slight advantage.

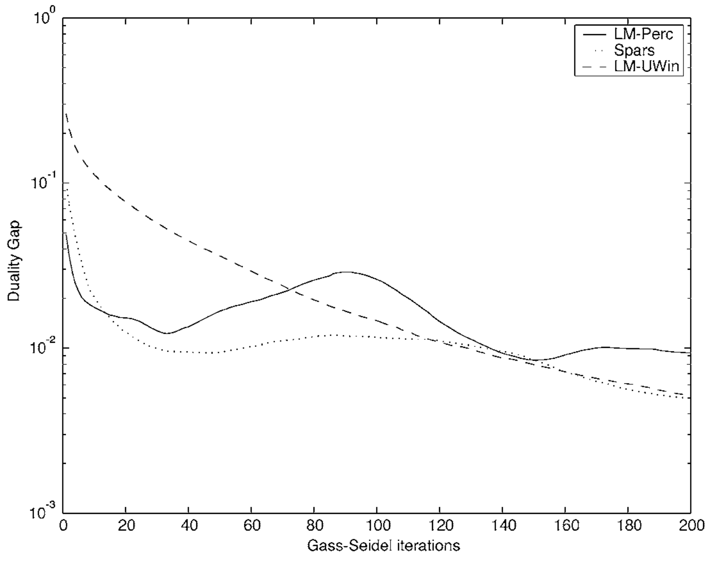
Since the "Spars" algorithm produces a sparse weight, it has an effect of automatic feature selection. Therefore in this case, it is not surprising to see that the algorithm achieves a better performance than a large margin Perceptron. Furthermore, the average non-zeros in linear weights obtained from the Spars algorithm is 400 over all 118 categorizes (and 1500 over top 10 categories). This is a significant reduction compared with the 45000 features we start with.

Convergence curve. We study the convergence behavior of Algorithm 1 as the number of Gauss-Seidel iterations increases. Two measures are used: duality gap and BEP on the test set. Let $\tilde{w}$ and $\tilde{\alpha}$ be the current estimate of the optimal primal and dual variables. The duality gap measurement is based on the fact that

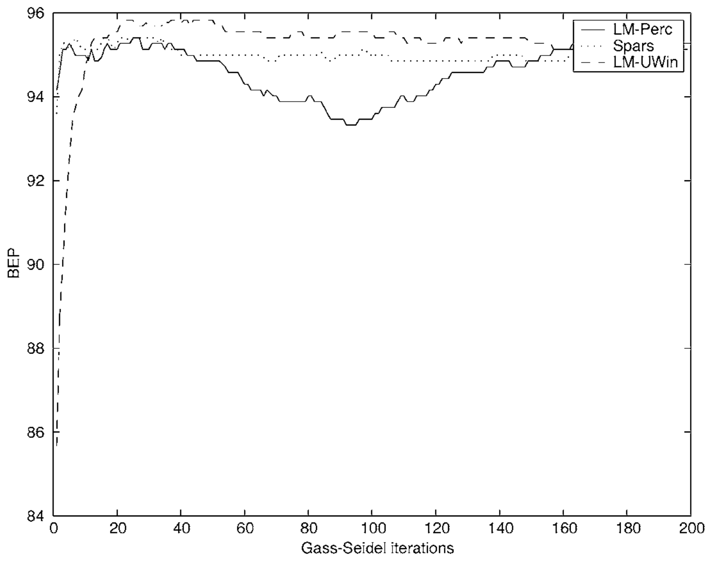
$$G(\tilde{w}, \tilde{\alpha}) = \sup_{\alpha} R(\tilde{w}, \alpha) - \inf_{w} R(w, \tilde{\alpha}) \geq 0.$$

$G(\tilde{w}, \tilde{\alpha})$ is zero when $\tilde{w}$ is an optimal solution to (2), and $\tilde{\alpha}$ is an optimal solution to (8). By definition, the duality gap $G(\tilde{w}, \tilde{\alpha})$ equals the sum of the value of (2) at $\tilde{w}$, and the value of (8) at $\tilde{\alpha}$.

We only report results on the Reuters "acq" and "trade" categories. The "acq" category has a BEP of about 95; and "trade" has a BEP of about 75. These choices thus reflect convergence behaviors of Algorithm 1 for relatively clean data and for relatively noisy data. To save space, we only consider LM-Perc, LM-NWin, and Spars. Since LM-Perc has been used in practice, and corresponds to the exact optimization of $(*)$, we regard it as the base-line for the purpose of studying behaviors of the gradient descent rule (11) for a general formulation in (8). We would also like to point out that with the default initial values, the corresponding duality gaps for all algorithms considered in this section are ones.



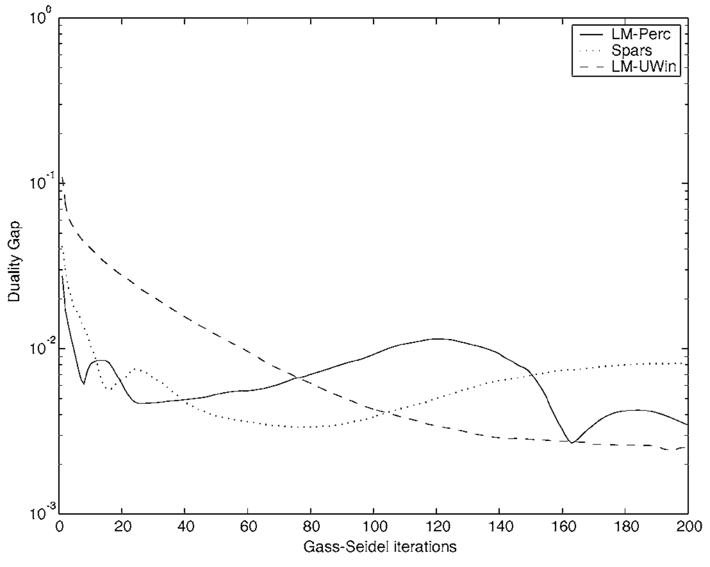
duality gap curves



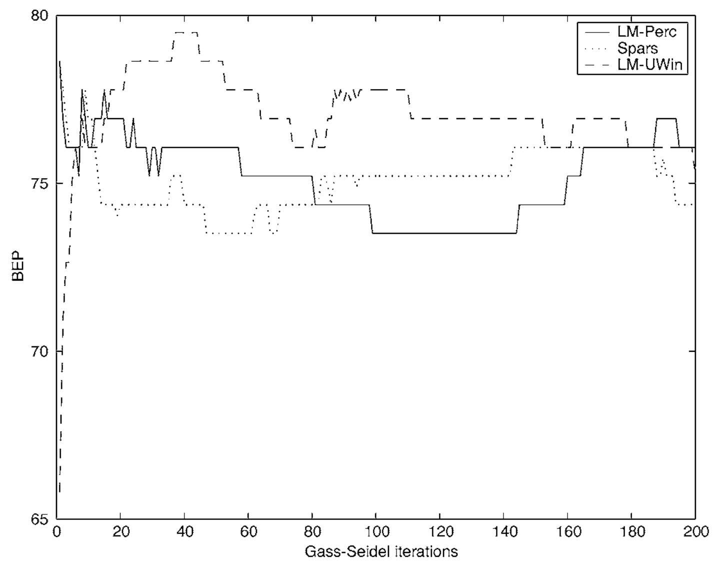
test-set BEP curves

Figure 2. Convergence curves on the "acq" category.

Figure 2 and figure 3 show the convergence behavior of the algorithms, measured by duality gap and test-set BEP performance. They reflect typical behavior of the dual Gauss-Seidel algorithm. Results from the two figures are consistent, and they reveal a number of interesting properties.



duality gap curves



test-set BEP curves

Figure 3. Convergence curves on the "trade" category.

One observation is that for every algorithm, the duality gap quickly drops from the initial value of 1 after the first few iterations over the data. At the same time, the classification performance goes up to a level that is more or less stabilized. That is, further optimization has a relatively small impact on the classification accuracy. We can also see that the algorithms converge at similar rates, even though some use gradient descent, and some exact optimization.

It is also interesting to observe that the duality gap for the LM-Perc algorithm does not always decrease. However, as a result of using the exact update rule (25), the dual objective function value in (8) always decreases (which we have verified numerically). From the figures, we also see that this unstable behavior of duality gap is correlated with a deterioration of classification performance. In the next experiment, we show that this non-monotonic duality gap problem can be remedied by using a smaller update step size. Even though this problem does not lead to any contradiction (the duality gap will eventually goes to zero), it is not desirable. We do not have a definite explanation for this phenomenon. Our only conjecture is that due to the constraint $\alpha_i \in [0, 1]$ for each α_i (which introduces some non-smoothness), the large step size given by the exact update rule (25) makes α_i to oscillate.

Influence of learning rate. We study the impact of different learning rates on the three algorithms. To save space, we only use the "trade" category in Reuters.

For the LM-UWin and Spars algorithms, we plot the convergence curves with learning rates $\eta_i = 0.0001, 0.001, 0.01, 0.1$. The results are shown in figure 5 and figure 6. We see that even with every small learning rates, these algorithms converge rather quickly. The Winnow algorithm is quite stable with respect to these learning rates. However, for the Spars algorithm, with a large learning rate such as 0.1, the algorithm oscillates.

For the LM-Perc algorithm, we use an update that is a scale t of the exact solution (25), where $t = 0.01, 0.1, 1, 10$. The update is modified from (25) as below:

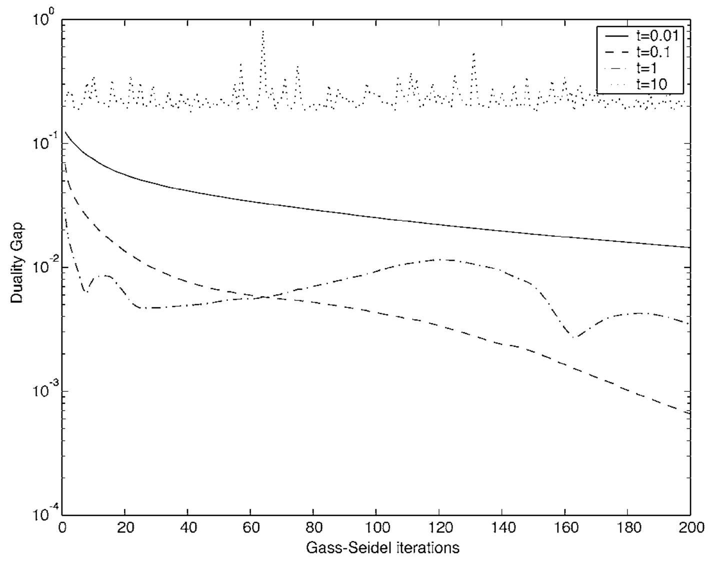
$$\Delta\alpha_i = y_i \max\left(\min\left(\alpha_i y_i - t\frac{c_2 - y_i}{c_1} y_i, 1\right), 0\right) - \alpha_i.$$

Interestingly, if we use a step-size that is ten time smaller than the exact solution, the convergence curve becomes much smoother (see figure 4). This indicates that it is desirable to use a learning rate that is smaller than the exact solution of $(*)$ given in (25).

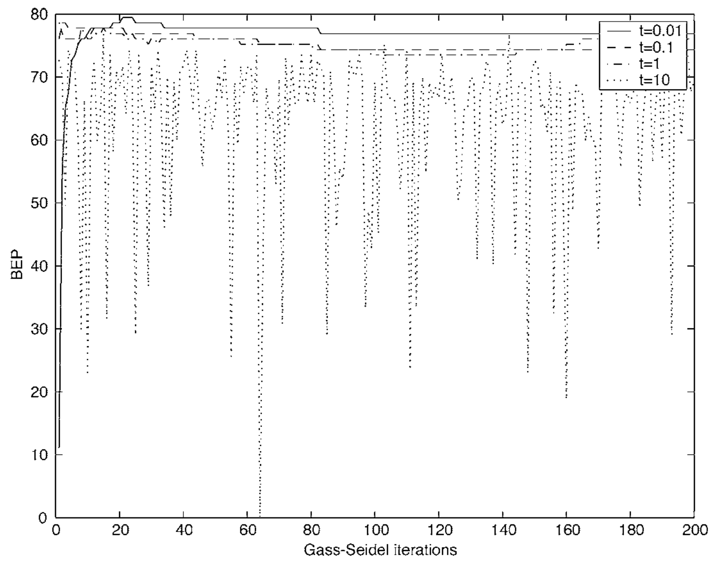
This experiment indicates that we can use an update that is ten or even a hundred time smaller than the update that solves $(*)$ exactly (or the update from Newton's method). A smaller update will not significantly slow down the convergence. However, it is dangerous to use a large learning rate. Therefore in practice, one should make sure that the selected learning rate is not larger than the update implied from Newton's method. Roughly speaking, for entropy regularized algorithms, the rate should be inversely proportional to $\|x\|_\infty^2$. For 2-norm regularized algorithms, the rate should be inversely proportional to $\|x\|_2^2$. It is usually not difficult to find a good learning rate for a specific application, as long as appropriate care is taken to prevent oscillation or slow convergence.

6.3. Augmented Lagrangian

In this experiment, we study the convergence behavior of augmented Lagrangian for problems with un-regularized bias term. We use the standard SVM formulation with α-update



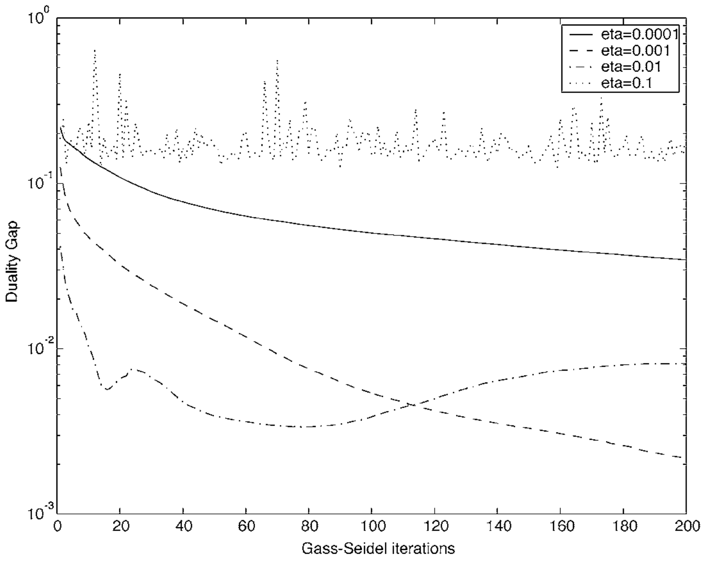
duality gap curves



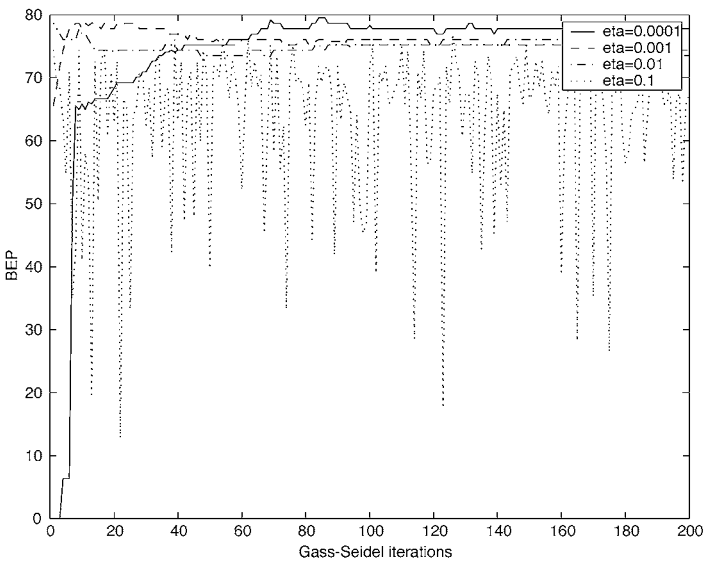
test-set BEP curves

Figure 4. Impact of learning rate for LM-Perc on the "trade" category.

given by (30), and s-update given by (31). From the previous experiment, we know that the exact α-update rule does not give a very desirable convergence behavior. Therefore we use an update that is ten times smaller than the update given in (30). This choice leads to a more desirable convergence curve.



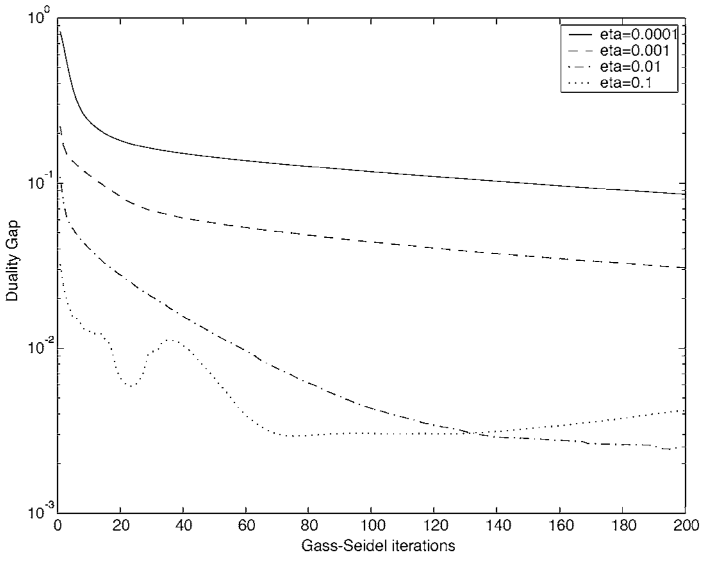
duality gap curves



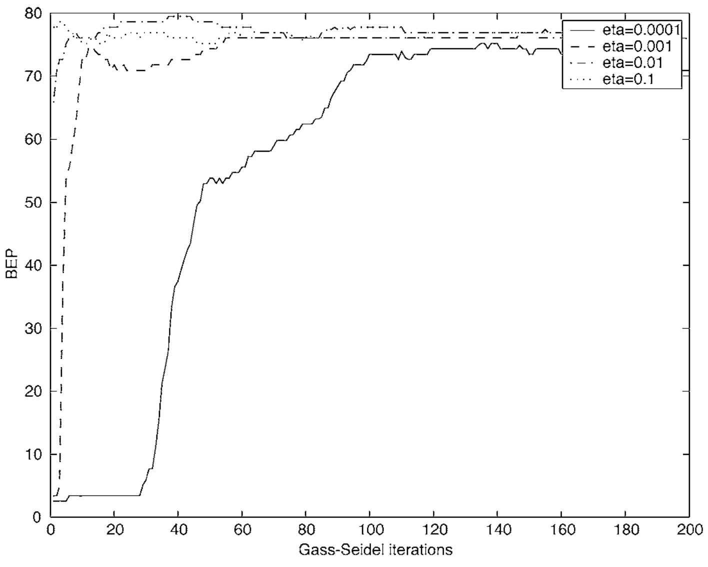
test-set BEP curves

Figure 5. Impact of learning rate for Spars on the "trade" category.

We study the convergence of duality gap and the quantity $|\sum_{i=1}^{n} \frac{\alpha_i}{\lambda n}|$ for the feasibility condition $\sum_{i=1}^{n} \frac{\alpha_i}{\lambda n} = 0$. Since the feasibility condition is not satisfied during the computation, formula (8) cannot be used to calculate the duality gap—it gives the value $+\infty$ when the feasibility condition is not satisfied. We thus consider the following modification:



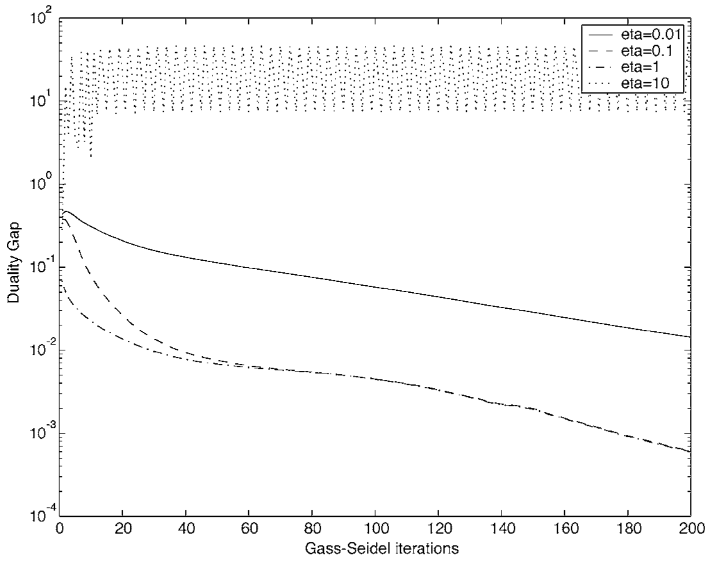
duality gap curves



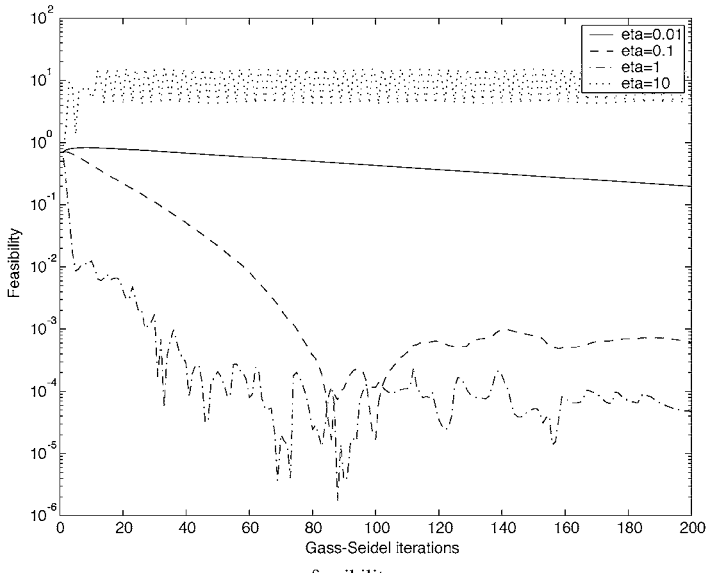
test-set BEP curves

Figure 6. Impact of learning rate for LM-UWin on the "trade" category.

with the current Lagrangian multiplier s, we set $b = s$ in the primal formulation (2), where we replace $w^T x_i$ by $w^T x_i + b$. The dual of (2) with fixed b is given by (27) with the augmented Lagrangian term $\frac{1}{2\mu n}(\sum_{i=1}^{n} \alpha_i)^2$ dropped. Their duality gap can be obtained accordingly.



duality gap curves



feasibility curves

Figure 7. Convergence of augmented Lagrangian with different s-update learning rates on the "trade" category.

We still use the "trade" category in Reuters in order to be consistent with other experiments. μ is chosen to be 1. We vary the s-update learning rate η in the s-update formula (31), and plot the corresponding convergence curves in figure 7. As we can see, with a large η, the algorithm becomes unstable. With an appropriate η, the algorithm converges fast (in terms of both feasibility and duality-gap). It seems that the choice of $\eta = 0.1$ is reasonable in many applications. The reason that the feasibility curve stops converging after a few dozen iterations (with $\eta = 0.1, 1$) is due to the error associated with the more slowly convergent duality gap, which appears as noise in the feasibility curve.

Compared with some earlier approaches to solve an SVM (such as SMO (Platt, 1999)), an advantage of the augmented Lagrangian approach is its simplicity and generality. There is also a very clear intuition for the α-update and the s-update in this scheme. In an augmented Lagrangian algorithm, we do not require the feasibility condition $\sum_{i=1}^{n} \alpha_i = 0$ to be satisfied. This allows simple α-update rules. However, in SMO style algorithms, this dual feasibility condition $\sum_{i=1}^{n} \alpha_i = 0$ should always be satisfied. As a result, each time two data points have to be selected simultaneously for optimization. This data selection task has to be implemented carefully, and can be quite time consuming.

7. Conclusion

Kernel methods have been successfully applied in many applications. Their foundation relies on the dual representation of a 2-norm regularized linear prediction method. In engineering, other regularized formulations have been used. It is thus useful to study dual representations of these methods.

In this paper, we consider a general form of regularized linear prediction methods. We extend the duality analysis in 2-norm regularization, and obtain a convex duality for linear models under general regularization conditions. Our discussion is very general. We allow a convex function to take the $+\infty$ value, which can be equivalently regarded as a constraint on the function. The only essential assumption required in this analysis is that the primal form has a feasible solution.

We show that for a general regularized linear system with convex risk in (2), a dual formulation (8) exists. An optimal solution $\hat{w}$ to the primal problem and the corresponding optimal solution $\hat{\alpha}$ to the dual problem satisfy the relationships (9) and (10). Therefore we can solve the dual learning problem and then obtain the primal solution through (9).

In Section 4, we present a Gauss-Seidel style numerical algorithm to solve the dual problem. There are two distinctive properties of this algorithm. The first property is that the algorithm updates the dual variable by examining one data at a time, which is very similar to an online algorithm. This update style establishes a connection between certain online learning methods and dual formulations of batch learning methods. The second property is that if we write a gradient descent rule to approximately solve each Gauss-Seidel step, algorithms with different regularization conditions have the same basic form of gradient descent rule (11). Their difference is reflected by the transfer function (9) which maps the dual variable into the primal variable. This unique feature implies that we can essentially keep the same update rule, but use a different transfer function to obtain the effect of different regularization. From a practical point of view, the transfer function (9) can be used to enforce various conditions on w such as positivity and sparsity.

In Section 5, we give a number of dual learning formulations. We show that this duality provides useful insights into learning algorithms such as robustness and sparsity, etc. We show that using an entropy regularization term, the gradient descent in the dual variable leads to an exponentiated gradient descent in the primal variable. This provides a batch learning foundation for online exponentiated gradient descent algorithms (Kivinen & Warmuth, 1997). By using the relationship of dual representation and online learning, we are able to convert general online updates for linear classification in Grove, Littlestone, and Schuurmans (1997) into corresponding large margin batch algorithms. In particular, Perceptron can be converted into regularized Perceptron (SVM) and Winnow can be converted into regularized Winnow.

In Section 5, we examine various aspects of the newly proposed dual algorithms. We show that the choice of regularization condition can be important in machine learning. This provides a reason for investigating issues related to non-square regularization. We also show that Algorithm 1 can converge very fast in practice, even with a small learning rate in the gradient descent update rule (11). The generality, algorithmic simplicity and the nice convergence behavior of the augmented Lagrangian method in Algorithm 3 also suggests that it can be regarded as an alternative to previously proposed SVM algorithms.

In summary, the convex duality presented in this paper generalizes the convex duality in kernel methods. We show that the derived dual formulation has many interesting properties. It leads to interesting new learning algorithms, and it provides useful insights into existing learning formulations.

Appendix

A. *Dual transform and subgradient*

The material in this appendix can be found in Rockafellar (1970). We include them here for completeness.

Let p be a closed proper convex function, as those considered in this paper. We define its *dual* q as

$$q(v) = \sup_u (u^T v - p(u)).$$

It is well known that q is a closed proper convex function, and its dual is p:

$$p(u) = \sup_v (u^T v - q(v)).$$

In this paper, we use $\nabla p(u)$ to denote a *subgradient* of a convex function p at u, which is a vector that satisfies the following condition:

$$\forall u', \quad p(u') \geq p(u) + \nabla p(u)^T (u' - u).$$

The set of all subgradients of p at u is called the *subdifferential* of p at u and is denoted by $\partial p(u)$.

The following fact is very useful in our discussion. It can be found in Rockafellar (1970), Section 23.

Proposition A.1. *Let p and q be dual transforms, then the following conditions are equivalent:*

1. $v_0 \in \partial p(u_0)$.
2. $u^T v_0 - p(u)$ *achieves its supremum in u at u_0.*
3. $p(u_0) + q(v_0) = u_0^T v_0$.
4. $u_0 \in \partial q(v_0)$.

Proof: The subgradient inequality defining condition 1 can be rewritten as $u^T v_0 - p(u) \leq u_0^T v_0 - p(u_0)$. By definition, this is the same as condition 2. Condition 2 is the same as condition 3 by the definition of dual transform. Finally note that condition 3 is symmetric in p and q. Thus dually, its equivalence with condition 1 implies its equivalence with condition 4. □

B. *Strong duality*

It is well-known (for example, see (Rockafellar, 1970) Section 36) that the duality gap

$$G = \inf_w \sup_\alpha R(w, \alpha) - \sup_\alpha \inf_w R(w, \alpha) \geq 0.$$

Clearly, the non-negativity of duality gap implies that if there exist $\hat{w}$ and $\hat{\alpha}$ such that

$$R(\hat{w}, \hat{\alpha}) = \sup_\alpha R(\hat{w}, \alpha) = \inf_w R(w, \hat{\alpha}), \tag{32}$$

then $\hat{w}$ is a solution to (5), and $\hat{\alpha}$ is a solution to (6). It also implies that it is valid to interchange the order of $\inf_w$ and $\sup_\alpha$ in (5) since $G = 0$. If (32) holds, then $(\hat{w}, \hat{\alpha})$ is called a saddle point.

Let $\hat{w}$ be a solution to (5). In the following, we construct $\hat{\alpha}$ that satisfies (7) and (32). By definition, a convex function achieves its minimum at a point where 0 is a subgradient. Using the subgradient algebra in Rockafellar (1970), Section 23 (specifically Theorem 23.8 and Theorem 23.9), we obtain

$$\frac{1}{n} \sum_{i=1}^n f_1'(\hat{w}^T x_i, y_i) x_i + \lambda \nabla g(\hat{w}) = 0,$$

where f_1' denotes a subgradient of $f(a, b)$ with respect to a, and ∇g denotes a subgradient of g. Now, let $\hat{\alpha}_i = -f_1'(\hat{w}^T x_i, y_i)$, we obtain:

$$\nabla g(\hat{w}) = \frac{1}{\lambda n} \sum_{i=1}^n \hat{\alpha}_i x_i.$$

This gives (7). We only need to show that (32) holds.

By Proposition A.1 and (7), $\hat{w}$ achieves the minimum of

$$g(w) - w^T \frac{1}{\lambda n} \sum_{i=1}^{n} \hat{\alpha}_i x_i.$$

That is,

$$R(\hat{w}, \hat{\alpha}) = \inf_{w} R(w, \hat{\alpha}).$$

Also by the construction of $-\hat{\alpha}_i$ as a subgradient of $f(\cdot, y_i)$ at $\hat{w}^T x_i$ and Proposition A.1, we know that for all i, $\hat{\alpha}_i$ achieves the maximum of

$$-\alpha_i \hat{w}^T x_i - k(-\alpha_i, y_i).$$

That is,

$$R(\hat{w}, \hat{\alpha}) = \sup_{\alpha} R(\hat{w}, \alpha).$$

This finishes the proof.

C. *Dual constraint*

If h is not differentiable at every feasible point, then the subgradient of h may not be unique. This causes problems when we attempt to obtain the primal solution $\hat{w}$ of (2) from (9). This situation is related to the degeneracy of Eq. (7) at $\hat{w}$. The following situations may occur when a convex function $p(v_0)$ is not differentiable at a point v_0:

- p is continuous but not differentiable.
- p is not continuous at v_0. In this case, p achieves $+\infty$ in any neighborhood of v. That is, u_0 belongs to the boundary of the feasible region $D_p = \{v : p(v) < \infty\}$.

In this paper, we are not interested in the first situation since the chance this happens at the optimal solution in (9) is very rare. Furthermore, one can always slightly modify h so that it does not contain any continuous non differentiable point. However, in practice, the second situation can occur frequently. Since finding a feasible solution of the dual optimization problem (8) is equivalent to minimizing (8) in D_h, it is quite possible that the optimal solution lies on the boundary of D_h.

In order to obtain a useful computational procedure, we impose the following assumptions:

- There is a convex function $\bar{h}$ such that $\bar{h} = h$ in D_p, and $\bar{h}$ is differentiable on the boundary of D_p.

- We assume that the convex set D_h can be equivalently written as the following constraints:

  $$C_1 v - b_1 = 0, \quad c_2(v) \leq 0.$$

  C_1 is a matrix, b is a vector, and c_2 is a differentiable vector valued function with convex components. Note that since D_h is convex, therefore an associated non-trivial equality constraint has to be linear: the line segment connecting two points that satisfy the equality constraint also satisfies the constraint.

To solve (8) numerically, we use the Lagrangian multiplier method, which is very popular in numerical optimization (Fletcher, 1987). We introduce Lagrangian multiplier vectors s_1 and s_2, where s_1 corresponds to the equality constraint $C_1 v - b_1 = 0$, and $s_2 \geq 0$ corresponds to the inequality constraint $c_2(u) \leq 0$. We modify h using the Lagrangian variables as:

$$h_{s_1,s_2}(v) = \bar{h}(v) + s_1^T(C_1 v - b_1) + s_2^T c_2(v).$$

Clearly, $h_{s_1,s_2}(v) \leq h(v)$ for all v. Let $g_{s_1,s_2}(u)$ be the dual transform of $h_{s_1,s_2}(v)$, then by definition, $g_{s_1,s_2}(u) \geq g(u)$ for all u.

We now consider Lagrangian variables s_1 and s_2 that solve (8). That is, the solution $\hat{\alpha}(s_1, s_2)$ of

$$\hat{\alpha}(s_1, s_2) = \arg\inf_{\alpha} \left[\frac{1}{n} \sum_{i=1}^{n} k(-\alpha_i, y_i) + \lambda h_{s_1,s_2}\left(\frac{1}{\lambda n} \sum_{i=1}^{n} \alpha_i x_i \right) \right] \tag{33}$$

belongs to D_h, under the complementarity condition:

$$s_2 \cdot c_2\left(\frac{1}{\lambda n} \sum_{i=1}^{n} \alpha_i x_i \right) = 0, \quad s_2 \geq 0. \tag{34}$$

We use $\cdot$ to denote component-wise multiplication of two vectors.

It is easy to check that if s_1 and s_2 solve (8), then $\hat{\alpha}$ in (33) also solves (8). This is because $h_{s_1,s_2} \leq h$, and at the optimal solution $\hat{\alpha}$ of (33), $h_{s_1,s_2} = h$.

The existence of such Lagrangian multipliers s_1 and s_2 that solve (8) forms a central topic in mathematical programming. It can be regarded as a part of the general convex duality theory (see Rockafellar (1970), Section 28). For simplicity, we skip the related analysis.

We now assume that there exist s_1 and s_2 that solve (8). We also assume that h_{s_1,s_2} is differentiable at $\frac{1}{\lambda n} \sum_{i=1}^{n} \hat{\alpha}_i x_i$, so that its subgradient is unique. Then (9) can be modified to obtain a solution of (2) as

$$\hat{w} = \nabla h_{s_1,s_2}\left(\frac{1}{\lambda n} \sum_{i=1}^{n} \hat{\alpha}_i x_i \right). \tag{35}$$

The proof is given below.

Since h_{s_1,s_2} is differentiable, vector $\hat{w}$ given in (35) solves the primal problem (2) with g replaced by g_{s_1,s_2}. We have

$$\frac{1}{n}\sum_{i=1}^{n} f(\hat{w}^T x_i, y_i) + \lambda g_{s_1,s_2}(\hat{w}) = -\left[\frac{1}{n}\sum_{i=1}^{n} k(-\hat{\alpha}_i, y_i) + \lambda h_{s_1,s_2}\left(\frac{1}{\lambda n}\sum_{i=1}^{n}\hat{\alpha}_i x_i\right)\right].$$

This equality follows from the fact that at the optimal solution, the duality gap is zero (see Appendix B).

Note that $h_{s_1,s_2}(v) \le h(v)$ for all v by construction. Therefore by the definition of dual transformation, $g_{s_1,s_2}(u) \ge g(u)$ for all u. Since

$$h_{s_1,s_2}\left(\frac{1}{\lambda n}\sum_{i=1}^{n}\hat{\alpha}_i x_i\right) = h\left(\frac{1}{\lambda n}\sum_{i=1}^{n}\hat{\alpha}_i x_i\right),$$

we obtain

$$\frac{1}{n}\sum_{i=1}^{n} f(\hat{w}^T x_i, y_i) + \lambda g(\hat{w}) \le -\left[\frac{1}{n}\sum_{i=1}^{n} k(-\hat{\alpha}_i, y_i) + \lambda h\left(\frac{1}{\lambda n}\sum_{i=1}^{n}\hat{\alpha}_i x_i\right)\right].$$

This inequality means that the duality gap is less than or equal to zero. However, since a duality gap can never be negative, the equality must hold. This implies that $\hat{w}$ is a solution to (2).

D. *Augmented Lagrangian algorithm*

Assume that h contains a constraint in (8). We would like to extend Algorithm 1 so that the Gauss-Seidel style update is preserved. By Appendix C, if we can find Lagrangian variables s_1 and s_2 that solve (8), then we can directly replace h in (8) by h_{s_1,s_2}, and use Algorithm 1 to obtain the optimal solutions $\hat{\alpha}$ and $\hat{w}$.

The difficulty is to determine the appropriate Lagrangian multipliers s_1 and s_2. We would like to use an algorithm that automatically adjusts α, s_1 and s_2. The general idea is to use a two step procedure:

1. Fix s_1 and s_2. Use Algorithm 1 to obtain the optimal $\hat{\alpha}(s_1, s_2)$ in (33).
2. Check if $\hat{\alpha}(s_1, s_2)$ satisfies the complementarity condition (34), and the feasibility condition $\hat{\alpha}(s_1, s_2) \in D_h$. Update s_1 and s_2 accordingly to reduce violations.

To implement the procedure, even though one may directly work with the Lagrangian h_{s_1,s_2}, it is numerically preferable to work with an augmented Lagrangian function. The resulting method is called *augmented Lagrangian method*, which has been used in state of art numerical optimization packages such as LANCELOT (Conn, Gould, & Toint, 1992). The basic augmented Lagrangian update works only for equality constrained systems, although it can be modified to deal with inequality constraints (for example, see Conn, Vicente, and Visweswariah (1999) for a recent development).

For simplicity, in this paper, we only give numerical procedures for the equality constrained situation. We consider the following form of equality constraint for D_h:

$$Cv - b = 0. \tag{36}$$

We now consider a modification of h by augmented Lagrangian as

$$\bar{h}_{s,\mu}(v) = \bar{h}(v) + s^T(Cv - b) + \frac{\lambda n}{2\mu}(Cv - b)^2,$$

where $\mu > 0$ is a positive real number. The third term is used to stabilize the system: $\bar{h}_{s,\mu}(v)$ is strictly convex in a direction that is perpendicular to the constraint surface $Cv - b$. This extra stability is helpful in numerical computation.

We can now modify (33) using the augmented Lagrangian function as follows:

$$\hat{\alpha}_\mu(s) = \arg\inf_\alpha \left[\frac{1}{n}\sum_{i=1}^n k(-\alpha_i, y_i) + \lambda \bar{h}_{s,\mu}\left(\frac{1}{\lambda n}\sum_{i=1}^n \alpha_i x_i\right)\right]. \tag{37}$$

Since the term $\frac{\lambda n}{2\mu}(Cv-b)^2$ vanishes when α is feasible (i.e. $\alpha \in D_h$), a Lagrangian parameter s such that $\hat{\alpha}_\mu(s) \in D_h$ is also a Lagrangian parameter in (33) that solves (8). This implies that if $\hat{\alpha}_\mu(s)$ is feasible, $\hat{\alpha}_\mu(s)$ is a solution of (8), and

$$\hat{w} = \nabla \bar{h}\left(\frac{1}{\lambda n}\sum_{i=1}^n \hat{\alpha}_\mu(s)_i x_i\right) + C^T s \tag{38}$$

gives a solution of (2).

The following algorithm is a standard Augmented Lagrangian update that solves (8) under constraint (36) (cf. Fletcher (1987), p. 292):

Algorithm 2 (*Dual Augmented Lagrangian*)

```
let $\alpha = \alpha^0$, $v_j = \frac{1}{\lambda n}\sum_{i=1}^n \alpha_i^0 x_{ij}$ for $j = 1, \ldots, d$, $s = 0$, and $\mu = \mu^0 > 0$
let $\Delta s = +\infty$
**for** $k = 1, 2, \ldots$
  solve (8) with $h(\cdot)$ replaced by $h_{s,\mu}(\cdot)$ in Algorithm 1
    (use the current $(\alpha, v)$ as the initial point, and update $(\alpha, v)$)
  **if** $\|Cv - b\|_\infty > 0.5\|\Delta s\|_\infty$
    $\Delta s = Cv - b$
    $\mu = \mu/4$
  **else**
    $\Delta s = Cv - b$
    $s = s + \frac{\lambda n}{\mu}\Delta s$
  **end**
**end**
$\hat{w} = \nabla h(v) + C^T s$.
```

To understand the intuition behind the update formula

$$s_{new} = s_{old} + \frac{\lambda n}{\mu} \Delta s \qquad (\Delta s = C v_{old} - b)$$

used in Algorithm 2, we observe that $\forall v_{new} = v_{old} + \Delta v$:

$$s_{new}^T (C v_{new} - b) - s_{old}^T (C v_{new} - b) = \frac{\lambda n}{\mu} \big(\Delta s^T \Delta s + \Delta s^T C \Delta v \big).$$

Due to the term $\Delta s^T C \Delta v$, the solution v_{new} with the new Lagrangian parameter s_{new} favors a change Δv that is in the opposite direction of Δs. In fact, by comparing the value of the old solution of (37) before the s-update, and that of the new solution after the s-update, we can see that the new solution of (37) leads to a change $\Delta v = v_{new} - v_{old}$ such that $\Delta s^T C \Delta v < 0$. Note that $(C(v_{new} - b))^2 = (C(v_{old} - b))^2 + 2\Delta s^T C \Delta v + O(\Delta v^2)$. Therefore under the assumption that Δv is small, the s-update formula attempts to make the constraint $Cv - b = 0$ to be more satisfied with the new Lagrangian parameter s_{new}.

A nice property of the Augmented Lagrangian approach is that if the augmented dual problem can be solved by Algorithm 1 to a sufficiently high accuracy, then the step $\mu = \mu/4$ can only be executed a finitely number of times. This means that upon termination, μ is bounded away from zero.

However, an disadvantage of Algorithm 2 is that the augmented dual problem needs to be solved fairly accurately, which may require a significant amount of iterations in Algorithm 1. In practice, it is useful to update s more often, but with a smaller step size.

Algorithm 3 (*Modified Dual Augmented Lagrangian*)

```
let $\alpha = \alpha^0$, $v_j = \frac{1}{\lambda n} \sum_{i=1}^n \alpha_i^0 x_{ij}$ for $j = 1, \ldots, d$, $s = 0$, and $\mu = \mu^0 > 0$
let $\Delta s = +\infty$
for $k = 1, 2, \ldots$
  for $i = 1, \ldots, n$
    find $\Delta\alpha_i$ by approximately minimizing
        $k(-\alpha_i - \Delta\alpha_i, y_i) + \lambda n \bar{h}_{s,\mu}(v + \frac{1}{\lambda n} \Delta\alpha_i x_i)$        $(**)$
    update $v$: $v_j = v_j + \frac{1}{\lambda n} \Delta\alpha_i x_{ij}$      $(j = 1, \ldots, d)$
    update $\alpha$: $\alpha_i = \alpha_i + \Delta\alpha_i$
  end
  update $s$: $s = s + \eta \frac{\lambda n}{\mu} (Cv - b)$
end
$\hat{w} = \nabla h(v) + C^T s$.
```

For simplicity, we do not update μ in Algorithm 3. The positive number $\eta > 0$ in the s-update is a learning rate. It can be shown that as long as η is sufficiently small, the algorithm converges to a correct solution. Furthermore, under moderate regularity conditions, the asymptotic rate of convergence, which depends on η, is linear. Due to the space limitation, we skip the analysis.

An advantage of Algorithm 3 is its clear intuition: the update formula for α is to minimize the augmented Lagrangian objective function (37); the update formula for s is to make the constraint (36) more satisfied.

E. Examples of dual pairs

We use $p(u)$ to denote a convex function with variable u, and $q(v)$ to denote its dual transform with dual variable v.

For many simple convex functions, their dual transforms can be expressed in closed forms. It is often not difficult to calculate the dual transform $q(v)$ from $p(u)$. As an example, we consider $p(u) = \frac{1}{2}u^T K u$, where K is a symmetric positive definite operator. Given v, assume that the maximum of $\sup_u u^T v - p(u)$ is achieved at u_0, then $v = \nabla p(u_0) = K u_0$. Therefore $u_0 = K^{-1}v$. This implies that $q(v) = u_0^T v - p(u_0) = \frac{1}{2}v^T K^{-1} v$.

In the following, we list a number of duality pairs. The p-norm $\|\cdot\|_p$ of a vector is defined as $\|u\|_p = (\sum_i \|u_i\|^p)^{1/p}$.

1. $p(u) = \frac{1}{2}u^T K u$; $\quad q(v) = \frac{1}{2}v^T K^{-1} v$.
   K is a symmetric positive definite operator.
2. $p(u) = \frac{1}{p'}\|u\|_p^{p'}$; $\quad q(v) = \frac{1}{q'}\|v\|_q^{q'}$.
   $p \ge 1$ and $q \ge 1$ are dual pairs: $1/p + 1/q = 1$; $p' \ge 1$ and $q' \ge 1$ are dual pairs: $1/p' + 1/q' = 1$.
3. $p(u) = \sum_j u_j \ln \frac{u_j}{e\mu_j}$ $(u_j \ge 0)$; $\quad q(v) = \sum_j \mu_j \exp(v_j)$.
   $\{\mu_j > 0\}$ is a set of positive prior.
4. $p(u) = \sum_j u_j \ln \frac{u_j}{\mu_j}$ $(u_j \ge 0, \sum_j u_j = A)$; $\quad q(v) = A \ln(\sum_j \frac{\mu_j}{A} \exp(v_j))$.
   $\{\mu_j > 0\}$ is a set of positive prior such that $\sum_j \mu_j = A$.
5. $p(u) = \ln(1 + \exp(u))$; $\quad q(v) = v \ln v + (1 - v)\ln(1 - v)$ $(0 \le v \le 1)$.
6. $p(u) = -\ln u$; $\quad q(v) = -1 - \ln(-v)$.
7. $p(u) = \frac{1}{2}u^2$ if $|u| \le 1$, and $p(u) = |u| - \frac{1}{2}$ otherwise;
   $q(v) = \frac{1}{2}v^2$ $(|v| \le 1)$.
8. $p(u) = 0$ if $|u| \le \epsilon$, and $p(u) = \frac{1}{p}(|u| - \epsilon)^p$ otherwise;
   $q(u) = \epsilon|v| + \frac{1}{q}|v|^q$.
   $\epsilon > 0$. $p \ge 1$ and $q \ge 1$ are dual pairs: $1/p + 1/q = 1$.
9. $p(u) = \frac{1}{2}u^2$ if $0 \le u \le 1$, $p(u) = u - \frac{1}{2}$ if $u > 1$, and $p(u) = 0$ if $u < 0$;
   $q(v) = \frac{1}{2}v^2$ $(0 \le v \le 1)$.
10. $p(u) = \frac{1}{p}u^p$ if $u \ge 0$ and $p(u) = 0$ otherwise;
    $q(v) = \frac{1}{q}v^q$ $(v \ge 0)$.
    $p \ge 1$ and $q \ge 1$ are dual pairs: $1/p + 1/q = 1$.

In the calculation of dual transformation, it is also convenient to note that if

$$p'(u) = ap(S^T u + b),$$

where S is a non-singular linear transform, then the dual transform q' of p' is

$$q'(v) = aq\left(\frac{1}{a}S^{-T}v\right) - b^T S^{-T} v.$$

Acknowledgment

The author would like to thank suggestions from anonymous referees that have greatly enhanced the presentation of this paper.

References

Barron, A. R. (1993). Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Transactions on Information Theory, 39:3*, 930–945.

Bartlett, P. L. (1998). The sample complexity of pattern classification with neural networks: The size of the weights is more important than the size of the network. *IEEE Transactions on Information Theory, 44:2*, 525–536.

Bregman, L. M. (1967). The relaxation method of finding a common point of convex sets and its application to the solution of problems in convex programming. *USSR Computational Mathematics and Mathematical Physics, 7*, 200–217.

Chen, Scott Shaobing, Donoho, David L., & Michael A. (1999). Saunders. Atomic decomposition by basis pursuit. *SIAM J. Sci. Comput., 20:1*, 33–61.

Conn, A. R., Gould, N. I. M., & Toint, Ph. L. (1992). *LANCELOT*. Berlin: Springer-Verlag. A Fortran package for large-scale nonlinear optimization (release A).

Conn, Andrew R., Vicente, Luís N., & Visweswariah, Chandu. (1999). Two-step algorithms for nonlinear optimization with structured applications. *SIAM J. Optim., 9:4*, 924–947.

Cristianini, Nello & Shawe-Taylor John. (2000). *An introduction to support vector machines and other kernel-Based learning methods*. Cambridge, UK: Cambridge University Press.

Fletcher, R. (1987). *Practical methods of optimization*, 2nd ed. Chichester: New York, Wiley.

Gentile, C. & Littlestone, N. (1999). The robustness of the p-norm algorithms. In *COLT'99* (pp. 1–11).

Golub, G. H. & Van Loan, C. F. (1996). *Matrix computations*, 3rd ed. Baltimore, MD: Johns Hopkins University Press.

Grove, A. J., Littlestone, N., & Schuurmans, D. (1997). General convergence results for linear discriminant updates. In *Proc. 10th Annu. Conf. on Comput. Learning Theory* (pp. 171–183).

Helmbold, D. P., Kivinen, J., & Warmuth, M. K. (1999). Relative loss bounds for single neurons. *IEEE Transactions on Neural Networks, 10:6*, 1291–1304.

Hoerl, A. E. & Kennard, R. W. (1970). Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics, 12:1*, 55–67.

Jaakkola, T. & Haussler, D. (1999). Probabilistic kernel regression models. In *Proceedings of the 1999 Conference on AI and Statistics*.

Jaakkola, Tommi, Meila, Marina, & Jebara, Tony. (2000). Maximum entropy discrimination. In S. A. Solla, T. K. Leen, & K.-R. Müller (Eds.). *Advances in neural information processing systems 12* (pp. 470–476). Cambridge, MA: MIT Press.

Jaakkola, T. S., Diekhans, M., & Haussler, D. (2000). A discriminative framework for detecting remote protein homologies. *Journal of Computational Biology, 7*, 95–114.

Kivinen, J. & Warmuth, M. K. (1997). Additive versus exponentiated gradient updates for linear prediction. *Journal of Information and Computation, 132*, 1–64.

Littlestone, N. (1988). Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning, 2*, 285–318.

Mangasarian, O. L. (1999). Generalized support vector machines. In B. Schökopf A. J. Smola, P. Bartlett, & D. Schuurmans, (Eds.). *Advances in large margin classifiers* (pp. 135–146). Cambridge, MA: MIT Press.

McCullagh, P. & Nelder, J. A. (1989). *Generalized linear models*. London: Chapman & Hall.

Platt, John C. (1999). Fast training of support vector machines using sequential minimal optimization. In Bernhard Schölkopf, Christopher J. C. Burges, & Alexander J. Smola (Eds.). *Advances in kernel methods: support vector learning* (Ch. 12). Cambridge, MA: MIT Press.

Rockafellar, R. Tyrrell. (1970). *Convex analysis*. Princeton, NJ: Princeton University Press.

Vapnik, V. N. (1998). *Statistical learning theory*. New York: John Wiley & Sons.

Wahba, Grace. (1999). Support vector machines, reproducing kernel Hilbert spaces, and randomized GACV. In Bernhard Schölkopf, Christopher J. C. Burges, & Alexander J. Smola (Eds.). *Advances in kernel methods: support vector learning* (Ch. 6). Cambridge, MA: MIT Press.

Received March 2, 2000
Revised November 10, 2000
Accepted March 9, 2001
Final manuscript April 10, 2001