# A Walk from 2-Norm SVM to 1-Norm SVM

Submitted for Blind Review

## Abstract

*The 1-norm SVM performs better than the standard 2-norm regularized SVM on problems domains with many irrelevant features. This paper studies how useful the standard SVM is in approximating the 1-norm SVM problem. To this end, we examine a general method that is based on iteratively re-weighting the features and solving a 2-norm optimization problem. The convergence rate of this method is unknown. Previous work indicates that it might require an excessive number of iterations. We study how well we can do with just a small number of iterations. In theory the convergence rate is fast, except for coordinates of the current solution that are close to zero. Our empirical experiments confirm this. In both real-world and synthetic problems with irrelevant features, already one iteration is often enough to produce accuracy as good as or better than that of the 1-norm SVM. Hence, it seems that in these problems we do not need to converge to the 1-norm SVM solution near zero values. The benefit of this approach is that we can build something resembling the 1-norm regularized solver based on any 2-norm regularized solver. This is quick to implement and the solution inherits the good qualities of the solver such as scalability and stability. For linear SVMs the recent advances in efficient solvers make this approach practical.*

## 1 Introduction

Minimizing empirical error over some model class is a basic machine learning approach which is usually complemented with regularization to counterattack overfitting. The support vector machine (SVM) is an approach for building a linear separator, which performs well in tasks such as letter recognition and text categorization. In this paper the linear SVM is defined to solve the following minimization problem:

$$\min_w \underbrace{\frac{1}{2}\|w\|^2}_{\text{regularizer}} + \underbrace{C\sum_{i=1}^{m} \operatorname{loss}(y_i, f_w(x_i))}_{\text{error}}\,. \tag{1}$$

Here $\{x_i, y_i\}_{i=1}^m \in \mathbb{R}^d \times \{-1, 1\}$ is a training set of examples $x_i$ with binary labels $y_i$. The classifier $f_w(x)$ that we wish to learn is $w \cdot x$ (plus unregularized bias, if necessary), where $w \in \mathbb{R}^d$ is the normal of the separating hyperplane. The function $\operatorname{loss}(y, f(x))$ is the hinge loss $\max(0, 1 - y\,f(x))$ (L1-SVM), or its square (L2-SVM).

The squared 2-norm $\|w\|^2$ is not always the best choice as the regularizer. In principle the 1-norm $\|w\|_1 = \sum_{i=1}^d |w_i|$ can handle a larger number of irrelevant features before overfitting [18]. In the context of least-squares regression Tibshirani [22] gives evidence that L1-regularization (lasso regression) is particularly well-suited when the problem domain has small to medium number of relevant features. However, for a very small number of relevant features a subset selection method outperformed L1-regularization in these experiments and for a large number of relevant features the L2-regularization (ridge regression) was the best.

In a 1-norm SVM [23] the regularizer is the 1-norm $\|w\|_1$. The resulting classifier is a linear classifier without an embedding to an implicit high-dimensional space given by a non-linear kernel. Some problem domains do not require a non-linear kernel function. For example, this could be the case if the input dimension is already large [14]. Furthermore, we can try to map the features to an explicit high-dimensional space if the linear classifier on original features is not expressive enough. For instance, we could map training examples to values of a kernel function evaluated at random points in the training set.

In this paper we study a simple iterative scheme which approaches the 1-norm SVM by solving a series of standard 2-norm SVM problems. Each 2-norm SVM solves a problem where the features are weighted depending on the solution of the previous 2-norm SVM problem. Hence, we will refer to this algorithm as the *re-weighting algorithm* (RW). More generally, we can apply it to minimize any convex error function regularized with 1-norm.

In this scheme most of the complexity resides in the regular SVM solver. Hence, the desired features of any standard SVM solver are readily available. Such features include performance, scalability, stability, and minimization of different objective functions (like L1-SVM and L2-SVM). Several fast approximate solvers for linear SVMs have been proposed recently [21, 13]. They are sufficiently quick to justify solving several linear SVM problems for a single 1-norm SVM problem. For example, Pegasos [21] trains the Reuters data set with ca. 800,000 examples and 47,000 sparse features in five seconds (does not count the time to read the data to memory).

Our contribution is three-fold. First, we provide theoretical results on the speed of the convergence. It is known that similar optimization methods are equivalent to the 1-norm solution [9]. Unfortunately, the convergence rate is unknown. Neither we are able to prove hard bounds on it. We can, though, provide intuition on the behavior of the convergence. More precisely, we will lower bound the decrease of the 1-norm objective in one iteration. This lower bound is higher when the current solution is poor in terms of the 1-norm objective function. On the other hand, the bound also depends on our current solution: it shows that the convergence is slow on coordinates that are already near zero.

The second contribution is that we experimentally demonstrate the efficiency of the resulting algorithm in the specific application of the SVMs. Because theoretical results do not guarantee the speed of convergence, we experiment on how many iterations one needs to run the algorithm. Previous work [20] has suggested that a similar algorithm needs up to 200 iterations to converge in the case of the multinomial logistic regression. However, the experiments are complicated by the fact that minimizing the objective is merely a proxy of the real target — generalization accuracy. When measuring accuracy, the 1-norm SVM solution does not necessarily give the best performance on problems with irrelevant features. Each iteration of the RW algorithm solves an optimization problem. Thus, the accuracy of the solution given by any iteration may be good even if it has not reached the 1-norm SVM optimum. In fact, previous work [8] argues that the best performance is often somewhere between the 1-norm and 2-norm optima.

We will experiment on real-world data sets. These include both ones with many irrelevant features and ones without. These experiments will demonstrate that in these data sets the convergence to the 1-norm SVM is not necessary. Surprisingly, the RW algorithm actually performs *better* than 1-norm SVM on problems with few relevant features. Furthermore, our experiments on synthetic data attempt to quantify this effect. They suggest that the performance of the RW algorithm is better than 1-norm SVM or standard SVM in problem domains with varying number of relevant features.

Finally, we provide a patch to `liblinear` [13] that implements the RW algorithm.

The structure of this paper is as follows. In Section 2 we first cover the theoretical background of this work. Section 3 presents the algorithm and provides both intuition and theory on why it works. Section 4 concerns empirical behavior of the algorithm, including results on the speed of convergence and how the number of relevant features affects the performance. In Section 5 we discuss how the results in this paper relate to previous work. Finally, Section 6 concludes this work.

## 2 Theoretical Background

In this section we introduce the theoretical background on how 2-norm regularization has been used to solve 1-norm regularized problems. Consider a simple linear regression problem, where we wish to minimize over $w$ the following expression for continuous outputs $y_i$:

$$\underbrace{\|w\|_1 + C\sum_{i=1}^{m}(y_i - w\cdot x_i)^2}_{E_{\text{squared}}(w)}\,. \tag{2}$$

This is the 1-norm regularized least squares proposed by Tibshirani [22]. To the best of our knowledge Grandvalet and Canu [9, 10] were the first to suggest a connection between 1-norm regularization and 2-norm regularization. They showed that 1-norm regularized least squares regression equals 2-norm regularization of a certain error function. This minimization was over additional relevance parameters that weighted the features in the error function. These parameters were constrained by a condition that limited their 2-norm. However, their work does not give the same updates as the ones in this paper.

Independently from the Grandvalet and Canu's approach an expectation maximization (EM) algorithm has been used to justify similar re-weighting algorithms as the one studied in this paper. Minimizing (2) is equivalent to computing the maximum a posteriori (MAP) estimate of the weight vector $w$ given a Laplacian prior on it and Gaussian noise on the outputs of the assumed underlying model $w\cdot x$ (and appropriate variances). Figueiredo [6] derived an EM algorithm for computing the MAP estimate of the weight vector $w$ for the Laplacian prior, and arrived at the following iterative updates (the derivation uses hidden hyper-parameters on the variances of the items in $w$):

$$w^{(t+1)} = \arg\min_{w} \frac{1}{2}\sum_{i=1}^{d}\frac{w_i^2}{|w_i^{(t)}|} + E_{\text{squared}}(w), \tag{3}$$

where $w^{(t)}$ is the value of the weight vector in the $t$-th iteration and $d$ is the dimension of the example space. The above expression reduces a 1-norm regularized least-squares problem to 2-norm regularized least-squares problem, where the feature $i$ is weighted with $\sqrt{|w_i^{(t)}|}$. Similar justification through the EM algorithm could be derived for other error functions. For example, logistic regression regularized with 1-norm corresponds to the MAP estimate of $w$ with Laplacian prior and logistic loss function. The work of Argyriou

**Algorithm 1** 1-norm SVM via 2-norm SVM

```
Input: a training set $\{(x^i, y^i)\}_{i=1}^m$ and number of iterations $N$.
Output: a weight vector $w$.
Initialize vector $v^{(1)}$ to all ones.
for $t = 1$ to $N$ do
    for each example $x^i$ do
        for each coordinate $j$ do
            Set $x'^i_j := x^i_j v^{(t)}_j$.
        end for
    end for
    $w^{(t)} :=$
    solution to SVM with examples $\{(x'^i, y^i)\}_{i=1}^m$.
    for each coordinate $i$ do
        Set $v^{(t+1)}_i := \sqrt{|w^{(t)}_i v^{(t)}_i|}$.
    end for
end for
for each coordinate $i$ do
    Set $w_i := w^{(N)}_i v^{(N)}_i$.
end for
Return $w$
```

et al. [1] also implies an algorithm with the same updates as in (3). These updates are the ones that we use, though our error function is different.

## 3 The Re-Weighting Algorithm

Algorithm 1 re-weights the features in each iteration, which makes it possible to use the SVM solver as a black-box. In short, during $t$-th iteration the RW algorithm multiplies the feature $i$ with a weight $v^{(t)}_i$. Then it obtains a 2-norm SVM solution $w$ from these weighted features. The new weights $v^{(t+1)}_i$ are set to $\sqrt{|w_i v^{(t)}_i|}$.

We could improve the performance of the algorithm by tailoring the SVM solver. Here we are interested in simplicity rather than performance optimizations that have unclear value. Note that the algorithm is in fact oblivious to the choice of the error function, so the SVM solver could be either L1-SVM or L2-SVM (or any other convex error for that matter).

### 3.1 Intuition

Figure 1 gives a graphical justification for the benefits of L1-regularization over L2-regularization. In it the contour of the red tilted square finds a sparser solution, because it is more spiky in directions where the weights are zero. The blue dashed ellipse shows what effect weighting of the features has. If weighted correctly, the optimization with the squeezed ellipse will approximate L1-regularized solution better than the L2-regularized one.

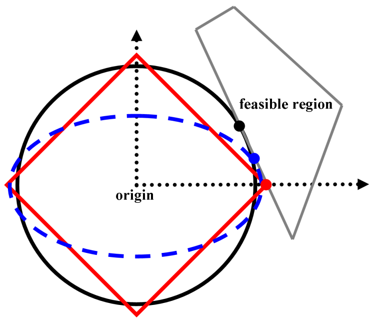

**Figure 1. 2-norm contour as the black circle, 1-norm as the red tilted square, and scaled 2-norm with the blue dashed ellipse.**

What is not apparent in Figure 1 is that if we know a non-optimal feasible point, then we can always choose the relative lengths of the axes of the ellipsis so that the L1-regularized objective value will decrease. This will be shown in the following theory section.

**Remark:** Figure 1 also suggests that the RW algorithm has difficulties to converge a coordinate to zero, because of the dull corners of the squeezed ellipse. The following theory section will also quantify this effect.

### 3.2 Theory

For vectors $w$ and $v$ we use $w \otimes v$ to denote an element-wise product (Hadamard product), where the $i$th coordinate $(w \otimes v)_i$ is $w_i v_i$. The absolute value $|w|$ of a vector $w$ is a vector containing the absolute values of the original vector: $|w|_i = |w_i|$. The error function $E(w)$ denotes the error given by a weight vector $w$. The modified error function $E_v(w)$ denotes the error, where features are weighted with $|v|$, i.e., $E_v(w) = E(w \otimes |v|)$. When the specific norm of $\|w\|$ is not indicated, it is always the 2-norm.

#### 3.2.1 Hard-margin

For clarity let us first cover the hard-margin SVM, in which each example $(x_i, y_i)$ implies a constraint: $y_i(w \cdot x_i) \geq 1$. The hard-margin SVM minimizes $\|w\|^2/2$ over these constraints. The following theorem gives a partial motivation for the minimization over weighted features. However, it does not guarantee that we find a better solution in each iteration. The theorem assumes that our current solution is a vector $v \otimes |v|$. We then minimize over the weighted problem where the weight on the $i$th feature is $|v_i|$. Now, one possible solution to the weighted problem is to set the solution $w$

to $v$. Then the new solution to the original problem equals the previous solution $v \otimes |v|$. However, Theorem 1 tells that *if* the minimization finds a solution $w$ to the weighted problem such that $\|w\|_2 < \|v\|_2$, then we obtain a solution $w \otimes |v|$ with a lower 1-norm. The subsequent Theorem 3 will add the necessary steps to guarantee that under certain conditions we will find such solution $w$.

**Theorem 1.** *Let $u$ be any vector and define $v$ as the weight vector induced by $u$, i.e., $u = v \otimes |v|$. Assume that $w$ satisfies all constraints $y_i(w \cdot (x_i \otimes |v|)) \geq 1$. The weighted problem implies a new solution $u^{new} = w \otimes |v|$ for the original unweighted problem. Now, if $\|w\|^2 < \|v\|^2$, then $\|u^{new}\|_1 < \|u\|_1$.*

*Proof.* The vector $u^{\text{new}}$ satisfies the hard-margin constraints because $w$ satisfies the constraints weighted with $v$. The following equations show that the 1-norm of $u^{\text{new}}$ is strictly smaller than $\|u\|_1$:

$$\begin{aligned}
\|u^{\text{new}}\|_1 &= \left\|\sqrt{u^{\text{new}}}\right\|_2^2 \\
&= \underbrace{|w| \cdot |v| \leq \||w|\|_2 \, \||v|\|_2}_{\text{Cauchy-Schwartz inequality}} \\
&= \underbrace{\|w\|_2 \, \|v\|_2 < \|v\|_2^2}_{\text{we assume } \|w\|_2 < \|v\|_2} = \|u\|_1 .
\end{aligned}$$

□

### 3.2.2 Soft-margin and Generalization to Any Error Function

Let us now generalize Theorem 1 to the minimization of L1-regularizer plus any error function, including the hinge loss of the SVM.

**Theorem 2.** *If $u = v \otimes |v|$ and $u^{new} = w \otimes |v|$ and*

$$\frac{1}{2}\|w\|^2 + E_v(w) < \frac{1}{2}\|v\|^2 + E_v(v), \qquad (4)$$

*then $u^{new}$ is a better solution to the L1-regularized optimization than $u$:*

$$\|u^{new}\|_1 + E(u^{new}) < \|u\|_1 + E(u).$$

*Furthermore, the 1-norm objective decreases by at least as much as the weighted objective in (4).*

The proof of Theorem 2 is given in Appendix A.

### 3.2.3 Convex Error Gives Convergence

Theorem 2 does not state that we will find a vector $w$ with smaller weighted objective value than $v$ even if there is a solution $u^\star$ with a smaller value to the L1-regularized error. Theorem 3, though, will show that an iteration finds a solution with smaller weighted objective value. The theorem assumes that the error function is convex and that the current solution has no zero in the coordinates, where $u^\star$ has a non-zero value. Therefore, Theorems 2 and 3 together say that an iteration of the RW algorithm decreases the value of the L1-regularized objective function.

We will use a scaled 2-norm $\|w\|_u = \sum_{i=1}^{d} w_i^2/|u_i|$. A substitution $w := w' \otimes |v|$ shows that regularization with $\|w\|_u/2$ equals the weighted objective $\|w'\|^2/2 + E_v(w')$, if $u = v \otimes |v|$. Intuitively $\|w\|_u$ approximates $\|w\|_1$ in a neighborhood of $u$, which yields the following theorem.

**Theorem 3.** *Let $h$ be a vector which is normalized so that $\|h\|_u = 1$. Let vector $c^\star h$, where $c^\star$ is a scalar, denote any direction in which the L1-regularized objective value decreases when starting from $u$. Hence,*

$$\|u + c^\star h\|_1 + E(u + c^\star h) < \|u\|_1 + E(u).$$

*Then the weighted objective function decreases to that same direction, i.e., there is a scalar $c > 0$ such that*

$$\frac{1}{2}\|u + c\,h\|_u + E(u + c\,h) + \frac{1}{2}c^2 \leq \frac{1}{2}\|u\|_u + E(u).$$

*More precisely, the step size $c$ is at least the minimum of $c^\star$ and*

$$-\left(\sum_{i=1}^{d} \textit{sign}\,(u_i)\,h_i\right) + \frac{1}{c^\star}\left(E\,(u) - E\,(u + c^\star\,h)\right).$$

The proof of Theorem 3 is also given in Appendix A.

Theorem 3 gives us some insight to the speed of convergence. It and Theorem 2 together show that the 1-norm objective function decreases by at least $c^2/2$ in one iteration. Let us now derive a more intuitive approximation to the expression of the step size $c$. If we assume that for all $i$ the signs of $h_i$ and $u_i$ differ, then $-c^\star\left(\sum_{i=1}^{d} \text{sign}\,(u_i)\,h_i\right) = \|u\| - \|u + c^\star\,h\|_1$. This approximation is good, if the 1-norm of the solution is an important factor in the optimization. Hence,

$$c \gtrapprox \min\left(\frac{\textit{Obj}\,(u) - \textit{Obj}\,(u^\star)}{c^\star}, c^\star\right),$$

where *Obj*$(x)$ is the 1-norm objective function. Thus, L1-regularized error drops quickly if our current solution $u$ is poor in comparison to optimal $u^\star$. On the other hand, $c$ has an inverse dependence on $c^\star$. This implies that convergence is slow along a coordinate, in which our current solution is already close to zero. Therefore, it might be impossible to obtain hard limits to the convergence rate, because the RW algorithm has trouble in converging near-zero coordinates. The next section experimentally tests how well we can manage with a small number of iterations.

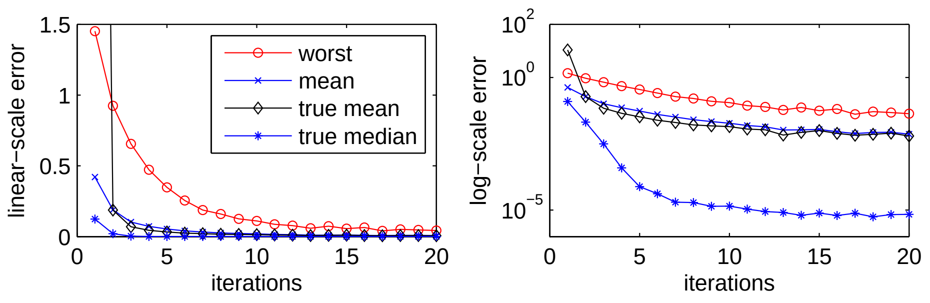

**Figure 2. The behavior of the objective with different number of iterations. The first two curves are derived from worst-case performances over $C$ on different data sets at the 20th iteration. The true mean and median are derived over all data sets and all values of the tradeoff parameter $C$.**

## 4 Empirical Behavior of Re-Weighting

### 4.1 Speed of Convergence

Let us present empirical evidence on how fast the RW algorithm converges to the 1-norm SVM solution. The experiments were performed with 16 data sets selected from UCI machine learning repository and Broad Institute Cancer Program. The data sets from UCI are `abalone`, `glass`, `segmentation`, `australian`, `ionosphere`, `sonar`, `bupa liver`, `iris`, `vehicle`, `wine`, `ecoli`, `wisconsin`, `german`, and `page`. The two data sets from Broad Institute are `leukemia` and `DLBCL`.[1]

For each data set we let the regularization parameter $C$ obtain powers of ten from $10^{-3}$ to $10^3$. In each iteration we recorded the objective value of the 1-norm regularized L1-SVM. For measuring the convergence we used `svmlight` [15] with default arguments for most runs (see below for a discussion on high values of $C$). We also measured the optimal value of the objective. For this we used a linear program and the `linprog` optimizer from Matlab (with default settings this is the `lipsol` solver).

Figure 2 gives a summary of the findings. The plotted objective value is $(\text{attained objective value} - \text{optimal value})/(\text{optimal value})$. For each data set, we selected the *worst* convergence over $C$ at the 20th iteration. From these 16 curves we formed two curves, the worst-case and the mean. Additionally, we show the mean and median over all problem domains and values of $C$ (true mean and true median).

The plots show that in absolute terms the convergence is fast on an average problem. We can see this from the behavior of the true median. However, the worst-case curve never decreases below 0.1. The worst convergence was obtained for the gene expression data sets, which have thousands of features out of which many are zero at the optimum. Thus, the small non-zero errors over many features lead to slow convergence in the objective.

We had a problem with `svmlight` for high values of the trade-off parameter $C$. The objective we measure is the primal objective. However, `svmlight` actually optimizes the dual objective to a given error [15]. If zero hinge loss is attainable, then the primal objective is unstable for high values of the parameter $C$. This is because the difference between hinge loss of 0 and 0.01 is large after multiplied with $C = 1,000$. However, this does not appear in the error of the dual. Therefore, we used more strict error parameters in two of the problem domains (`leukemia` and `DLBCL`) for value 100 of the parameter $C$.

### 4.2 Accuracy of Classifiers

Let us turn our attention to our true objective: building accurate classifiers. In this section we chart out how quickly the accuracy changes during iterations. The experiments include both problem domains with many relevant features and those that have many irrelevant features. This selection criterion should guarantee that there is a difference between accuracies of 1-norm and 2-norm SVM.

#### 4.2.1 Datasets

`Reuters`[2] is a text categorization data set and `Reuters-sampled` is a synthetic problem domain obtained from it; each experiment samples 250 examples from the original training set. We form a binary classification task by training the CCAT category versus all other categories. `Gisette`[3] is a digit classification task, which

[1]Available from `http://www.ailab.si/orange/`.

[2]Available from `http://jmlr.csail.mit.edu/papers/volume5/lewis04a/`

[3]Available from `http://www.nipsfsc.ecs.soton.ac.uk/datasets/`

**Table 1. The number of examples and features in each data set. Number of examples in format X/Y denotes X examples in the given training set and Y examples in the given test set.**

| DATA SET | EXAMPLES | FEATURES |
|---|---|---|
| REUTERS | 23,149/199,328 | 47,236 |
| GISETTE | 6,000/1,000 | 5,000 |
| DLBCL | 77 | 7,070 |
| LEUKEMIA | 72 | 5,147 |
| LUNG | 203 | 12,600 |
| PROSTATA | 102 | 12,533 |
| SRBCT | 83 | 2,308 |
| SONAR | 208 | 59 |

contains many irrelevant features, because $50\%$ of its features are synthetic noise features. We also experiment with several gene expression data sets[4] which should have many irrelevant features. Recall that the gene expression data sets had a slow convergence in the experiments of the previous section. The data sets are are `DLBCL`, `Leukemia`, `Lung`, `Prostata`, and `SRBCT`. Additionally we experiment on classical `Sonar` data set from UCI. Table 1 summarizes the properties of these data sets.

### 4.2.2 Experimental setup

For `Reuters` and `Gisette` we use the given split into training set and a test or validation set. For the other domains we perform 30 experiments, in which we randomly split the data set half and half into a training set and a test set. We select the parameter $C$ with 5-fold cross-validation. Different iterations of the RW algorithm may use different $C$. The best value for $C$ is the rounded-down median of those values that attain the best cross-validated error. The range of $C$ is the powers of ten in $[10^{-9}, 10^{2}]$ for gene expression data sets and in $[10^{-5}, 10^{5}]$ for the other data sets (the best accuracy is on these intervals for all algorithms). The 2-norm SVM solver is `liblinear` [13]. We use default settings, except that the solver is set to L1-SVM dual optimizer. The default settings include an additional bias feature that has a constant value of 1. The 1-norm SVM is solved with Matlab, as in the previous section. We train the most frequent label versus the remaining labels, if the data set contains more than two labels. Each example in the data set `Gisette` is normalized to unit norm and each feature in the gene expression data sets is normalized to zero mean and unit variance.

[4]Available from `http://www.ailab.si/orange/`

Table 2 gives the results. Let us discuss a few observations. First, the difference in accuracy during iterations is small in `Reuters`, but large in `Reuters-sampled`. This suggests that if a problem domain has a large number of examples then the regularization has only a small effect. Second, the best accuracy in gene expression data sets is in between the solutions of 2-norm SVM and 1-norm SVM. In some of these data sets the difference between 1-norm SVM and the RW algorithm is surprisingly large. Friedman and Popescu [8] make a similar observation in their experiments with linear regression on both synthetic and proteomics data.

Of course, these experiments still leave open the question on how to determine the right number of iterations. In our experiments, a good number of iterations was easy to find with cross-validation. We already perform a cross-validation over the trade-off parameter $C$. Hence, we have an access to a table that gives the cross-validated error for each value of $C$ and for each iteration.

## 4.3 The Effects of the Number of Relevant Features

The results in Table 2 suggest that the RW algorithm is competitive with the 1-norm SVM already when run with a low number of iterations. However, this evidence comes from a relatively small number of problem domains. It is unclear how the number of relevant features affects the performance. Hence, we perform experiments on synthetic data in order to shed some light on this issue.

We generate synthetic data sets as follows. A data set consists of 100 examples with 200 features each. The examples are equally split between the two binary labels ($y = \pm 1$). Each label is distributed as a spherical unit-variance Gaussian. These distributions are slightly apart, because we set their means to be at a distance of three from each other. A parameter $d_{\mathrm{rel}}$ tells how many relevant features a data set has. The difference in means is equally divided into exactly $d_{\mathrm{rel}}$ features. Hence, $d - d_{\mathrm{rel}}$ features are identically distributed in both of the labels, and $d_{\mathrm{rel}}$ features are at different means $1.5\, y/\sqrt{d_{\mathrm{rel}}}$. Otherwise the experimental setting is similar to that in the previous section. Figure 3 gives the results.

An important subtlety arose in the experiments, so we present two graphs in Figure 3. In the left graph the 1-norm SVM solver performs well when the number of relevant features is high. This is not because 1-norm SVM objective function is good in these problems. Rather this performance is a property of a particular implementation. The reason is that in these problems the optimal weight vector $w$ was the zero vector. If the optimal $w$ is the zero vector and the solver allows an approximation error, then the 1-norm objective does not determine the resulting classifier. Rather,

**Table 2. Classification accuracies of the RW algorithm and 1-norm SVM runs with different numbers of iterations. The empirical standard deviation of the accuracy is given. The $i$-th iteration of the RW algorithm is denoted by RW($i$). For `Gisette` and `Reuters` our 1-norm SVM solver runs out of memory.**

| DATA SET | 2-NORM SVM | RW(2) | RW(3) | RW(5) | RW(10) | 1-NORM SVM |
|---|---|---|---|---|---|---|
| REUTERS | 93.4 | 93.5 | 93.5 | 93.4 | 93.3 | NA |
| REUTERS-SAMPLED | $85.9 \pm 0.2$ | $84.9 \pm 0.3$ | $83.7 \pm 0.3$ | $82.2 \pm 0.3$ | $80.6 \pm 0.3$ | $72.8 \pm 0.8$ |
| GISETTE | 97.8 | 98.3 | 98.5 | 98.2 | 98.1 | NA |
| DLBCL | $68.6 \pm 1.0$ | $91.4 \pm 0.7$ | $93.2 \pm 0.7$ | $93.5 \pm 0.8$ | $91.7 \pm 1.1$ | $85.3 \pm 1.3$ |
| LEUKEMIA | $82.7 \pm 1.0$ | $95.2 \pm 0.6$ | $96.0 \pm 0.5$ | $95.8 \pm 0.6$ | $94.7 \pm 0.8$ | $90.1 \pm 1.5$ |
| LUNG | $92.9 \pm 0.4$ | $95.1 \pm 0.3$ | $95.0 \pm 0.4$ | $94.5 \pm 0.5$ | $93.8 \pm 0.6$ | $92.6 \pm 0.5$ |
| PROSTATA | $90.8 \pm 0.6$ | $91.4 \pm 0.5$ | $92.0 \pm 0.5$ | $92.5 \pm 0.5$ | $91.6 \pm 0.6$ | $88.5 \pm 0.8$ |
| SRBCT | $88.8 \pm 1.1$ | $98.1 \pm 0.4$ | $98.7 \pm 0.3$ | $98.0 \pm 0.5$ | $96.8 \pm 0.6$ | $95.7 \pm 0.5$ |
| SONAR | $73.3 \pm 0.7$ | $73.0 \pm 0.6$ | $73.1 \pm 0.7$ | $72.8 \pm 0.8$ | $72.6 \pm 0.7$ | $72.2 \pm 0.8$ |

the behavior of the solver determines which weight vector $w$ it finds among the feasible ones around the zero vector. Hence, a small difference in $w$ can result in huge difference in accuracy. In the synthetic experiments with high number of relevant features the typical 1-norm SVM solution $w$ had a 1-norm $\|w\|_1$ of $10^{-12}$ and a hinge-loss of 67, but the generalization error was far from 0.5.

In the right graph we set the accuracy of all weight vectors with 1-norm smaller than 0.0001 to 50% correct (which equals the accuracy of zero weight vector) for cross-validation purposes. This shows how the accuracy of the 1-norm SVM dramatically changes. The 1-norm SVM is better only when the data set has a single relevant feature, and even this difference disappears if we execute two iterations with the RW algorithm.

## 5 Related Work and Discussion

### 5.1 Related Methods

Zhu et al. [23] put forward the 1-norm SVM and solved the optimization problem with a linear program. A linear programming solution is only available for the L1-SVM objective function, because the L2-SVM objective function is non-linear. Mangasarian [17] describes a specialized solver for the 1-norm linear programming problem. In it the problem is transformed to an unconstrained quadratic problem for which there are efficient solvers.

Breiman [4] is to the best of our knowledge the first one to suggest a RW algorithm similar to the one studied in this paper. His non-negative garotte solves a linear regression problem by first solving an ordinary least squares problem, which gives a solution $w$. This solution is then used to weight another regression problem, where the $i$th feature is weighted with $w_i$. The solution to this new regression problem is limited to a positive weight vector, and the sum of these weights is constrained.

Another algorithm that weights features is the hybrid SVM studied by Zou [24]. It first computes 2-norm SVM solution $w$. Then it weights the feature $i$ with $|w_i|^{(\text{parameter})}$ and solves this new problem with 1-norm SVM solver. In Zou's experiments the hybrid SVM performed better than 1-norm SVM.

Several papers study applications of optimizing the 1-norm regularized error with a 2-norm regularized solver. Grandvalet and Canu [11] derive an algorithm that finds a non-linear kernel in which each feature in the *input* space is automatically weighted to match relevance. Argyriou et al. [1, 2] apply the method to the problem of multitask learning. Rakotomamonjy et al. [19] use a similar method to learn at the same time a classifier and a kernel which is a convex combination of other kernels. This is called the multiple kernel problem.

### 5.2 Discussion on Performance

The iterative RW algorithm relies on a 2-norm SVM solver. Hence, the performance of this solver is important. Recently several fast approximate solvers for the linear 2-norm SVM have been developed. `Svm-perf` [16] is the first linear-time linear SVM solver. Another, more recent cutting plane algorithm is OCAS [7]. Shalev-Shwartz et al. [21] as well as Bottou and Bousquet [3] propose algorithms based on online stochastic gradient descent. These algorithms do not converge well to an exact solution, but they quickly find an approximate solution. Chapelle [5] studies a Newton method and `liblinear` [13] is a package containing several algorithms, including a L1-SVM solver based on dual coordinate descent. These algorithms scale

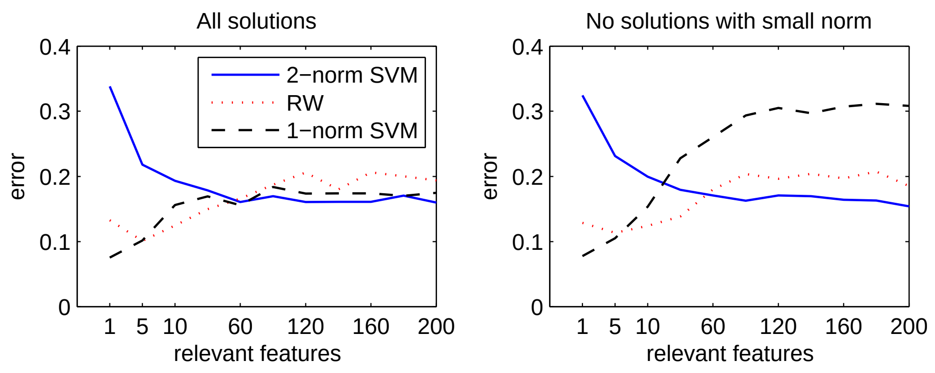

**Figure 3. How the error behaves as a function of relevant features. The number of features is 200. The solvers are the 2-norm SVM, the RW algorithm with one iteration, and the 1-norm SVM.**

easily to large data sets containing hundreds of thousands training examples. Typically the training time is bounded by the time needed to read the input from a file.

In this paper we did not perform comprehensive experiments on the run time of the RW algorithm. Instead we gave evidence on how many iterations the algorithm needs. Thus, we can approximate the run time in units of "2-norm SVM problem". This is more informative than measuring run times which are influenced by factors such as the termination criteria of the optimization and whether we use a subsample of the training set.

The experiments in Section 4 were performed by repeatedly calling the same implementation of either `svmlight` or `liblinear` with differently weighted inputs. However, we also integrated the RW algorithm directly to `liblinear` to assure ourself that our intuitions on performance are correct. As an example, ca. 200,000 examples from the Reuters data set took 33 seconds to train with a 2-norm SVM and 38 seconds to train with one additional weighted iteration (we set $C$ to one). The fact that reading the data from a file took 27 seconds explains the small difference between these run-times. Hence, weighting the features did not significantly affect the run-time. The computer on which we ran the experiments was a 2,8 Ghz Pentium 4 with 1 GB of main memory.

In general, L1-regularized optimization has a reputation for being more difficult than L2-regularized optimization. The non-differentiability of 1-norm is not as such a sufficient reason for this, because we can approximate the absolute value $|x|$ with a smoother function like $\sqrt{x^2 + \epsilon}$, where $\epsilon > 0$ is a small value. However, the current theoretical upper bounds [12] in online convex optimization suggest that the convexity of the objective is related to how quickly an online stochastic gradient descent algorithm converges.

For instance, consider the special case of SVMs. Shalev-Shwartz et al. [21] derived an upper bound on the number of iterations that an online stochastic gradient algorithm needs in order to arrive to a specified additive error. Their upper bound depends on the fact that the SVM objective is strongly convex. A $\lambda$-strongly convex function is one which cannot be approximated anywhere well with a linear function. More precisely, a linear approximation at the point $x$ of $\lambda$-strongly convex function must differ from the true value by at least $\lambda \left\|x - x'\right\|^2 /2$ at any other point $x'$. Bottou and Bousquet [3] arrive to a similar conclusion. They, however, use unrelated methods. They show that the convergence depends on the curvature of the objective function near the optimum (they define the curvature using the Hessian of the objective function). Note that both of these results depend on defining the objective function in such a way that the error in it does not scale with the number of training examples $m$.

## 6 Conclusion

We studied how a simple iterative re-weighting algorithm performs in problem domains with irrelevant features. In theory the re-weighting algorithm converges to a value that is close to the 1-norm SVM solution. The experimental results indicated that a small number of iterations is enough to attain the best accuracy. In fact, in many problem domains the re-weighting algorithm outperformed the 1-norm SVM and the standard SVM. However, a close convergence to 1-norm SVM might require many more iterations. This work suggests that we can use popular 2-norm SVM solvers to derive a solver that is more resilient to irrelevant features. Hence, the good properties of these solvers are available for problem domains that contain such.

In the synthetic experiments we faced a problem in the comparison of the algorithms. The accuracy of our 1-norm SVM solver was high even in problem domains with many relevant features. This was not because the objective function of the 1-norm SVM was good. Rather, an approximation in the solution vector around the zero made all solutions feasible. Hence, the performance depended on the particular implementation.

## References

[1] A. Argyriou, T. Evgeniou, and M. Pontil. Multi-task feature learning. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems*, volume 19, pages 41–48. MIT Press, Cambridge, MA, 2007.

[2] A. Argyriou, T. Evgeniou, and M. Pontil. Convex multi-task feature learning. *Machine Learning*, 73(3):243–272, 2008.

[3] L. Bottou and O. Bousquet. The tradeoffs of large scale learning. In J. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems*, volume 20, pages 161–168. NIPS Foundation, 2008.

[4] L. Breiman. Better subset regression using the nonnegative garrote. *Technometrics*, 37(4):373–384, 1995.

[5] O. Chapelle. Training a support vector machine in the primal. *Neural Computation*, 19(5):1155–1178, 2007.

[6] M. A. T. Figueiredo. Adaptive sparseness for supervised learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25:1150–1159, 2003.

[7] V. Franc and S. Sonnenburg. Optimized cutting plane algorithm for support vector machines. In *Proceedings of the 25th International Conference on Machine Learning*, pages 320–327, New York, NY, 2008. ACM.

[8] J. H. Friedman and B. E. Popescu. Gradient directed regularization for linear regression and classification. Technical report, Stanford University, 2004.

[9] Y. Grandvalet. Least absolute shrinkage is equivalent to quadratic penalization. In *Perspectives in Neural Computing*, volume 1, pages 201–206. Springer, 1998.

[10] Y. Grandvalet and S. Canu. Outcomes of the equivalence of adaptive ridge with least absolute shrinkage. In D. A. C. Michael J. Kearns, Sara A. Solla, editor, *Advances in Neural Information Processing Systems*, volume 11, pages 445–451, Cambridge, MA, USA, 1999. MIT Press.

[11] Y. Grandvalet and S. Canu. Adaptive scaling for feature selection in SVMs. In S. T. S. Becker and K. Obermayer, editors, *Advances in Neural Information Processing Systems*, volume 15, pages 553–560. MIT Press, Cambridge, MA, 2003.

[12] E. Hazan, A. Agarwal, and S. Kale. Logarithmic regret algorithms for online convex optimization. *Machine Learning*, 69(2-3):169–192, 2007.

[13] C.-J. Hsieh, K.-W. Chang, C.-J. Lin, S. S. Keerthi, and S. Sundararajan. A dual coordinate descent method for large-scale linear SVM. In *Proceedings of the 25th International Conference on Machine Learning*, pages 408–415, New York, NY, 2008. ACM.

[14] C. W. Hsu, C. C. Chang, and C. J. Lin. A practical guide to support vector classification. Technical report, Department of Computer Science and Information Engineering, National Taiwan University, Taipei, 2003.

[15] T. Joachims. Making large-scale support vector machine learning practical. In *Advances in Kernel Methods: Support Vector Learning*, pages 169–184. MIT Press, Cambridge, MA, USA, 1999.

[16] T. Joachims. Training linear SVMs in linear time. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 217–226, New York, NY, 2006. ACM.

[17] O. L. Mangasarian. Exact 1-norm support vector machines via unconstrained convex differentiable minimization. *Journal of Machine Learning Research*, 7:1517–1530, 2006.

[18] A. Y. Ng. Feature selection, $L_1$ vs. $L_2$ regularization, and rotational invariance. In *Proceedings of the 21st International Conference on Machine Learning*, pages 78–85, New York, NY, 2004. ACM.

[19] A. Rakotomamonjy, F. R. Bach, S. Canu, and Y. Grandvalet. SimpleMKL. *Journal of Machine Learning Research*, 9:2491–2521, November 2008.

[20] M. Schmidt, G. Fung, and R. Rosales. Fast optimization methods for L1 regularization: A comparative study and two new approaches. In *Proceedings of the 18th European Conference on Machine Learning*, pages 286–297, Berlin, Heidelberg, 2007. Springer-Verlag.

[21] S. Shalev-Shwartz, Y. Singer, and N. Srebro. Pegasos: Primal Estimated sub-GrAdient SOlver for SVM. In *Proceedings of the 24th International Conference on Machine Learning*, pages 807–814, New York, NY, 2007. ACM.

[22] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B*, 58:267–288, 1996.

[23] J. Zhu, S. Rosset, T. Hastie, and R. Tibshirani. 1-norm support vector machines. In S. Thrun, L. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems*, volume 16. MIT Press, Cambridge, MA, 2004.

[24] H. Zou. An improved 1-norm SVM for simultaneous classification and variable selection. In M. Meila and X. Shen, editors, *Proceedings of the 11th International Conference on Artificial Intelligence and Statistics*, volume 2, pages 675–681, 2007.

## A Proofs

In this Appendix we prove Theorems 2 and 3 presented in Section 3.2.

### A.1 Proof of Theorem 2

First use the definition of $u^{\text{new}}$:

$$\|u^{\text{new}}\|_1 + E(u^{\text{new}}) = |w| \cdot |v| + E_v(w).$$

Then apply the Cauchy-Schwartz inequality on $|w| \cdot |v|$:

$$|w| \cdot |v| + E_v(w) < \|w\| \, \|v\| + E_v(w).$$

Use the assumption that $\frac{1}{2}\|w\|^2+E_v(w) < \frac{1}{2}\|v\|^2+E_v(v)$ to obtain:

$$\begin{aligned}&\|w\|\,\|v\| + E_v(w)\\ &\quad < \quad \|w\|\,\|v\| + \frac{1}{2}\left(\|v\|^2 - \|w\|^2\right) + E_v(v)\,.\end{aligned}$$

Note that this step guarantees that the 1-norm objective will decrease by at least as much as the weighted objective.

Shuffle some terms in the latter expression to obtain:

$$\|v\|^2 - \frac{1}{2}\left(\|v\|^2 - 2\|w\|\,\|v\| + \|w\|^2\right) + E_v(v)\,.$$

Complete the square which gives us the following:

$$\|v\|^2 - \frac{1}{2}\underbrace{(\|v\| - \|w\|)^2}_{\geq 0} + E_v(v) < \|v\|^2 + E_v(v)\,.$$

Now, the definition of $v$ was that $u = v \otimes |v|$ which implies that $E(u) = E_v(v)$. So,

$$\|v\|^2 + E_v(v) = \|u\|_1 + E(u)\,.$$

The claim follows.

## A.2 Proof of Theorem 3

Our goal is to find a lower bound to the difference

$$\underbrace{\frac{1}{2}\|u\|_u + E(u)}_{f(0)} - \left(\underbrace{\frac{1}{2}\|u + c\,h\|_u + E(u + c\,h)}_{f(c)}\right)$$

for some value of $c$. We will do this in three steps. In step 1 we will derive an upper bound $g(c)$ to $f(c)$. In step 2 we will compute $c' \geq 0$ that minimizes the upper bound $g(c')$. In step 3 we will find that for $c'' = \min(c^\star, c')$:

$$f(0) - f(c'') \geq f(0) - g(c'') \geq \frac{1}{2}c''^2$$

and the claim follows.

**Step 1**. First write out the norm

$$\begin{aligned}&\frac{1}{2}\|u + c\,h\|_u\\ &\quad = \quad \frac{1}{2}\sum_{i=1}^{d}\left(\frac{u_i^2}{|u_i|} + 2c\,\mathrm{sign}(u_i)\,h_i + \frac{c^2 h_i^2}{|u_i|}\right)\\ &\quad = \quad \frac{1}{2}\|u\|_1 + c\left(\sum_{i=1}^{d}\mathrm{sign}(u_i)\,h_i\right) + \frac{1}{2}c^2. \qquad (5)\end{aligned}$$

Approximate $E(u + c\,h)$ as a function of $c$. Use Jensen's inequality on the convex error function to obtain an upper bound

$$E(u + c\,h) \leq \left(1 - \frac{c}{c^\star}\right)E(u) + \frac{c}{c^\star}E(u + c^\star\,h). \qquad (6)$$

This upper bound holds for all $c$ on interval from zero to $c^\star$. Define the upper bound $g(c)$ as the sum of (5) and (6).

**Step 2**. Minimize the convex upper bound $g(c)$ with respect to $c$ to yield the following value $c'$:

$$c' = -\left(\sum_{i=1}^{d}\mathrm{sign}(u_i)\,h_i\right) + \frac{1}{c^\star}(E(u) - E(u + c^\star\,h)).$$

Now use the assumption that $u + c^\star\,h$ is a better solution than $u$ to the L1-regularized error:

$$\begin{aligned}&\|u\|_1 + E(u)\\ &\quad > \quad \|u + c^\star\,h\|_1 + E(u + c^\star\,h)\\ &\quad \geq \quad \|u\|_1 + c^\star\left(\sum_{i=1}^{d}\mathrm{sign}(u_i)\,h_i\right) + E(u + c^\star\,h).(7)\end{aligned}$$

Inequality (7) is true, because if signs of $u_i$ and $h_i$ are the same, then $|u_i + c^\star\,h_i| = |u_i| + c^\star|h_i|$. Otherwise, $|u_i + c^\star\,h_i| \geq |u_i| - c^\star|h_i|$. Inequality (7) implies that:

$$\sum_{i=1}^{d}\mathrm{sign}(u_i)\,h_i < \frac{1}{c^\star}\left(E(u) - E(u + c^\star\,h)\right).$$

Plugging the above inequality to (7) shows that $c'$ is positive. Hence, (6) holds for the minimum of $c'$ and $c^\star$.

**Step 3.** Set $c'' = \min(c^\star, c')$. As a final step, compute the difference $f(0) - g(c'')$. Do this in parts. First, (5) shows that

$$\frac{1}{2}\|u\|_u - \frac{1}{2}\|u + c''\,h\|_u = -c''\left(\sum_{i=1}^{d}\mathrm{sign}(u_i)\,h_i\right) - \frac{1}{2}c''^2.$$

Second, (6) shows that

$$E(u) - E(u + c''\,h) \geq \frac{c''}{c^\star}(E(u) - E(u + c^\star\,h)).$$

Hence we obtain a lower bound:

$$\begin{aligned}&-c''\left(\sum_{i=1}^{d}\mathrm{sign}(u_i)\,h_i\right) + \frac{c''}{c^\star}\left(E(u) - E(u + c^\star\,h)\right) - \frac{1}{2}c''^2\\ &= c''\,c' - \frac{1}{2}c''^2 \geq \frac{1}{2}c''^2,\end{aligned}$$

because of the definition of the $c'$.