Learning Non-Linear Classifiers with a Sparsity Constraint using L1 Regularization

Mathieu Blondel
Kobe University
mblondel@ai.cs.kobe-u.ac.jp

Kazuhiro Seki
Kobe University
seki@cs.kobe-u.ac.jp

Kuniaki Uehara
Kobe University
uehara@kobe-u.ac.jp

ABSTRACT
When combined with kernels, Support Vector Machines (SVMs) often achieve outstanding accuracy. This comes, however, at the cost of very expensive predictions. To address this issue, we consider the problem of learning a non-linear classifier with a sparsity constraint. Using \( \ell_1 \)-regularization, we show that this reduces to a univariate parameter search problem, which we show how to solve efficiently. Experiments show that our approach, while leading to much sparser models, is competitive with unconstrained kernel SVMs, both in terms of accuracy and training time.

Categories and Subject Descriptors
I.2.7 [Artificial Intelligence]: Learning

General Terms
non-linear classification

Keywords
kernel methods, sparse methods

1. INTRODUCTION
When combined with kernels, Support Vector Machines (SVMs) often achieve outstanding accuracy, thanks to the induced non-linearity. However, it is well-known that the number of support vectors grows linearly with the number of training instances [5], resulting in high prediction cost. Observing that, in practice, a small number of support vectors are often sufficient to achieve good accuracy, several approaches imposing a hard-limit (an upper bound) on the number of support vectors have been proposed. Among the most popular ones are Matching Pursuit like approaches [6], that greedily select support vectors until the limit is reached and online budgeted approaches [3], that remove a support vector whenever necessary. In this paper, we propose a different approach based on \( \ell_1 \)-regularization. Using \( \ell_1 \)-regularization, learning a classifier with a sparsity constraint reduces to a simple univariate parameter search problem, which we show how to solve efficiently. Experiments show that our approach, while leading to much sparser models, is competitive with unconstrained kernel SVMs, both in terms of accuracy and training time.

2. LEARNING NON-LINEAR CLASSIFIERS WITH A SPARSITY CONSTRAINT

2.1 Objective function
Stated formally, given \( n \) training instances \( x_i \in \mathcal{X} \), their associated labels \( y_i \in \{-1, +1\} \), \( m \) basis vectors \( b_j \in \mathcal{X} \) and a similarity function \( s: \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R} \), our goal is to learn a classifier \( \hat{y}(x) = \sum_j \beta_j s(x, b_j) \) that minimizes the following objective:

\[
\begin{align*}
\min_{\beta \in \mathbb{R}^m} & \sum_{i=1}^n \max(1 - y_i \beta^T S_i, 0)^2 \\
\text{subject to} & \quad ||\beta||_0 \leq B,
\end{align*}
\]

where \( ||\beta||_0 \) is the \( \ell_0 \) norm or cardinality of \( \beta \) (the number of non-zero coefficients in \( \beta \)) and \( B \) is a user-specified upper bound. \( S \in \mathbb{R}^{n \times m} \) is the similarity matrix such that \( S_{ij} = s(x_i, b_j) \) and \( S_i \in \mathbb{R}^{m} \) is the \( i^{th} \) row of \( S \). The similarity matrix \( S \) plays a similar role to the kernel matrix in an SVM objective. However, it does not need to be positive semi-definite (or even, symmetric) for the objective to remain convex. Moreover, unlike support vectors, basis vectors need not be training instances nor labeled (e.g., they can be k-means centroids). \( \max(1 - y_i \hat{y}_i, 0)^2 \) is called the squared hinge loss. We choose this loss because it is differentiable and efficient to compute.

However, it is well known that directly minimizing an objective of the form of (1) is a combinatorial NP-hard optimization problem [1]. Therefore, we do not attempt to solve it directly. Instead, we consider the following \( \ell_1 \)-regularized objective:

\[
\begin{align*}
\min_{\beta \in \mathbb{R}^m} & \quad f(\beta) = ||\beta||_1 + C \sum_{i=1}^n \max(1 - y_i \beta^T S_i, 0)^2, \quad (2)
\end{align*}
\]

where \( C > 0 \) is a penalty parameter. If we regard each row \( S_i \) as a pseudo training instance, (2) reduces to a standard \( \ell_1 \)-regularized objective. Therefore, any existing algorithm may be used to solve it. However, most existing approaches such as interior point methods require that the entire \( S \) matrix fits in memory, which is unrealistic when \( n \) is large, due to \( S \)'s \( O(n^2) \) space complexity. In this paper, we use the...
coordinate descent algorithm of Yuan et al. [7]. The main advantage of coordinate descent here is that it requires only
one column of S per iteration. Therefore, similarly to
decomposition methods for SVMs, S never needs to be entirely
materialized in memory and the algorithm can scale to large
datasets.

2.2 A search algorithm

Objective (2) has the following desirable property: in-
creasing values of C lead to increasing ||β||0.1. This suggests
that we can cast the problem of learning a classifier with a
sparsity constraint as a univariate parameter search problem:
finding the value of C that yields the solution β with best
accuracy, subject to the constraint ||β||0 ≤ B.

Our basic strategy for carrying out this search efficiently
is to repeatedly identify an interval [Cmin, Cmax] between
which we sequentially search for the best value of C match-
ing the sparsity constraint. Our main motivation for search-
ing sequentially in the interval (rather than, e.g., using a
bisection) is because i) smaller values of C are faster to
train ii) it allows to use warm-start (initialize the current
model with the previous model). To decide the initial value
of Cmin, we use the following lemma.

**Lemma 1.** Let C0 = \min_{j} \frac{1}{2} \sum_{i} y_i S_{ij}. Then, given any
C ≤ C0, the optimal solution to (2) is β = 0.

Proof omitted due to lack of space. Lemma 1 is very useful
because it tells us that we should not bother to compute
the solution of (2) for any C ≤ C0, as the resulting model
would be empty and thus useless for prediction. We can
therefore set Cmin ← C0 + ϵ, where ϵ is a small value (e.g.,
ϵ = 10−12). However, for reasons pointed out previously,
it is not desirable to entirely pre-compute S. In practice,
observing that the exact C0 value is not critical, we use the
approximation C0 ≈ \min_{j} \frac{1}{2} \sum_{i} y_i S_{ij}, where A is a small
subset of \{1, ..., n\} (e.g., |A| = 60). For the initial value
of Cmax, we require the user to define a value such that
||β||0 ≥ B. This value can be set arbitrarily large and will
be refined by the algorithm later.

We present our procedure in Algorithm 1. We initial-
ize β with βmin, where βmin is initially set to 0. We
generate r uniformly spaced C values between the initial
Cmin and Cmax. For each C value, we compute the solu-
tion with warm-start. If the solution satisfies the constraint
||β||0 ≤ B, we measure its accuracy against a validation set
and keep it as the current best solution if the accuracy
improved. If the constraint is violated, we stop. In practice,
the initial interval [Cmin, Cmax] may be too large to find a
good solution. We would rather perform the search on the
interval [Cmin, Cmax] such that B − Δ ≤ ||β||0 ≤ B, where
B − Δ is a lower bound specified by the user. While search-
ing for the best value of C in the current interval, Algorithm
1 therefore updates, for future use, Cmin and Cmax so that
||β||0 is as close as possible to falling in B − Δ ≤ ||β||0 ≤ B.
In addition to β∗, Algorithm 1 returns βmin, Cmin and
Cmax. We can therefore call Algorithm 1 repeatedly on a
refined [Cmin, Cmax] interval. We call the number of times
Algorithm 1 is repeatedly called the number of passes. The
main advantage of the proposed algorithm is that, for each

```
Algorithm 1 Search algorithm

Input: B, Δ, r, βmin, Cmin, Cmax
Set β ← βmin
β∗ ← βmin
C ← Cmin
τ ← (Cmax − Cmin)/r
for i = 1 → r do
    Solve (2) using coordinate descent
    if ||β||0 ≤ B − Δ then
        Set Cmin ← C
        βmin ← β
    else if ||β||0 ≥ B then
        Set Cmax ← C
        break
end if
Measure accuracy of β on a validation set
Set β∗ to β if accuracy of β is better than of β∗
Set C ← C + τ
end for
Output: β∗, βmin, Cmin, Cmax
```

individual optimization problem during the search, coordi-
nate descent is run to convergence and the solution is exact.

3. EXPERIMENTS

We compared our proposed method with LaSVM [2] and a
dual SVM algorithm [4], where we used early stopping (ES)
when the upper-bound B was reached. We also compared
to optimizing (2) using the same early stopping and LaSVM
with unlimited number of support vectors (SV). All meth-
ods used greedy selection of the support/basis vectors in a
similar fashion to LaSVM, as well as debiasing. The dataset
we used are summarized in Table 1. #SV was obtained by
LaSVM (unlimited). The kernel/similarity function used is
the RBF kernel \(\kappa(x_i, x_j) = \exp(-\gamma||x_i - x_j||^2)\). We set r
to 50, Δ to B/10, Cmax to 0.5 and made 3 passes. γ and C
(for the baselines only) were set by cross-validation. Table 2
compares accuracy and training time when B = 200. Figure
1 compares accuracy for increasing values of B.

4. REFERENCES

[1] E. Amaldi and V. Kann. On the approximability of
minimizing nonzero variables or unsatisfied relations in

classifiers with online and active learning. *Journal of

classification on a budget. In *Proceedings of Neural

S. Sundararajan. A dual coordinate descent method for
large-scale linear svm. In *Proceedings of the 25th ICML


comparison of optimization methods and software for
large-scale l1-regularized linear classification. *J. Mach.
Table 1: Datasets

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Train</th>
<th>Test</th>
<th>Features</th>
<th>#SV</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adult</td>
<td>32,561</td>
<td>16,281</td>
<td>122</td>
<td>11,809</td>
</tr>
<tr>
<td>Banana</td>
<td>3,975</td>
<td>1,325</td>
<td>2</td>
<td>1,000</td>
</tr>
<tr>
<td>USPS</td>
<td>7,291</td>
<td>2,007</td>
<td>256</td>
<td>293</td>
</tr>
<tr>
<td>USPS+Noise</td>
<td>7,291</td>
<td>2,007</td>
<td>256</td>
<td>2,949</td>
</tr>
</tbody>
</table>

Table 2: Test accuracy (%), rank and training time (seconds) comparison

<table>
<thead>
<tr>
<th>B = 200</th>
<th>Adult</th>
<th>Banana</th>
<th>USPS</th>
<th>USPS+Noise</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proposed</td>
<td>84.49 (3)</td>
<td>89.80 (2)</td>
<td>98.77 (2)</td>
<td>98.45 (2)</td>
</tr>
<tr>
<td>ℓ₁-regularized (ES)</td>
<td>4.96</td>
<td>1.68</td>
<td>1.54</td>
<td>1.43</td>
</tr>
<tr>
<td>Dual SVM (ES)</td>
<td>82.48 (5)</td>
<td>88.17 (4)</td>
<td>98.35 (4)</td>
<td>92.40 (5)</td>
</tr>
<tr>
<td>LaSVM (ES)</td>
<td>82.70 (4)</td>
<td>87.67 (5)</td>
<td>98.27 (5)</td>
<td>98.04 (3)</td>
</tr>
<tr>
<td>LaSVM (unlimited)</td>
<td>85.09 (1)</td>
<td>90.03 (1)</td>
<td>99.30 (1)</td>
<td>99.05 (1)</td>
</tr>
</tbody>
</table>

Figure 1: Test accuracy (%) comparison for increasing values of the upper-bound B.