Generalized LARS as an Effective Feature Selection Tool for Text Classification With SVMs

S. Sathiya Keerthi SELVARAK@YAHOO-INC.COM
Yahoo! Research Labs, 210 S. DeLacey Avenue, Pasadena, CA-91105

Abstract

In this paper we generalize the LARS feature selection method to the linear SVM model, derive an efficient algorithm for it, and empirically demonstrate its usefulness as a feature selection tool for text classification.

1. Introduction

Text classification is an interesting collection of classification problems in which the number of features is large that often exceeds the number of training examples and, for which, linear classifiers such as logistic regression (Genkin et al., 2004), linear SVMs (Joachims, 1998) and regularized least squares work very well. For these problems, feature selection can be important, either for improving accuracy or for reducing the complexity of the final classifier. Filter methods such as information gain (Yang & Pedersen, 1997) and bi-normal separation (Forman, 2003) are usually employed to do feature selection.

LARS (Least Angle Regression and Shrinkage) is a 'semi-wrapper'[1] feature selection approach devised by Efron et al (Efron et al., 2004) for ordinary least squares. It is closely related to the Lasso model (Tibshirani, 1996) which corresponds to the use of L_1 regularizer for least squares problems. The great advantage of LARS is its ability to order variables according to their 'importance', while keeping the total computational cost small.

In this paper we generalize the LARS approach to linear SVMs, derive an efficient algorithm for it, along the lines of Rosset and Zhu's path tracking algorithm for L_1 regularized models (Rosset & Zhu, 2004), and point out that this algorithm is computationally practical, say for ordering and choosing the top 1000 features of a (binary) text classification problem. We evaluate the generalization performance (F-measure) on several text classification benchmark datasets and show that SVM with LARS generates an ordering of features that is much better than that obtained using information gain. Regularized least squares with LARS (Zou & Hastie, 2005; Efron et al., 2004) is also evaluated and shown to be inferior for text classification compared to SVM with LARS.

SVM with LARS can be viewed as an efficient approximate algorithm for a modified SVM formulation which uses both, an L_2 regularizer and an L_1 regularizer. (The algorithm essentially efficiently tracks the solution curve for all possible L_1 regularizer coefficient values.) As a part of our study we also evaluate the usefulness of keeping (or leaving out) the L_2 regularizer. When the L_2 regularizer is left out, the model with LARS is close in spirit to the sparse logistic regression model recently proposed (and shown to be very good) by Genkin et al (Genkin et al., 2004) for text classification. Keeping only the L_1 regularizer leads to an aggressive reduction of the training cost using a small subset of features. On some datasets this leads to an impressive performance while, on other datasets that need to keep a large number of features[2] for optimal performance, the aggressive feature removal property of the L_1 regularizer (acting alone) leads to a loss in performance. Thus, it is better to try both classifiers: one which keeps the L_2 regularizer and a second which leaves out the L_2 regularizer, and choose the better one based on validation.

The paper is organized as follows. Section 2 is the main section where the generalized LARS idea is explained and the SVM-LARS algorithm is derived. Section 3

[1]Unlike filter methods which order features by analyzing them independently, LARS sequentially chooses new features that are dependent on the features that are already chosen, and is 'wrapped around' the induction algorithm. However, it is not a full-fledged wrapper method since it is only based on optimizing the training set performance.

[2]The *20 Newsgroups* dataset is a good example of this case.

Appearing in *Proceedings of the 22nd International Conference on Machine Learning*, Bonn, Germany, 2005. Copyright 2005 by the author(s)/owner(s).

evaluates the various methods on 6 benchmark binary classification problems. Concluding remarks are given in section 4.

The following notations will be used in the paper. x' will denote the transpose of the vector/matrix x. x_j is the j-th component of the vector x. Given a set $\mathcal{A}$, $|\mathcal{A}|$ will denote the cardinality of $\mathcal{A}$. For $\beta \in R^m$ and $\mathcal{A} \subset \{1, \ldots, m\}$, $\beta_{\mathcal{A}}$ is the $\mathcal{A}$ dimensional vector containing $\{\beta_j, j \in \mathcal{A}\}$. If $q = \max_{p \in P} h(p)$ and $\bar{p} = \arg\max_{p \in P} h(p)$, then we say $\bar{p}$ defines q.

2. Generalized LARS Algorithm

Consider a (binary) classifier whose training can be expressed as

$$\min_{\beta} f(\beta) \tag{1}$$

where f is the sum of a regularizer term and a data-fit term, and the parameter vector β is an element of R^m. Let $g(\beta)$ denote the gradient of f. We are interested in the following two models.

Model 1. *Regularized Least Squares (RLS)*[3]

$$f(\beta) = f_{RLS}(\beta) = \frac{\lambda_2}{2}\|\beta\|^2 + \frac{1}{2}\sum_{i=1}^{n} r_i^2(\beta) \tag{2}$$

where $\|\beta\|$ is the L_2 norm of β, x_i is the i-th example's input vector, $r_i(\beta) = \beta' x_i - t_i$, $t_i \in \{1, -1\}$ is the target (1 denotes class 1 and -1 denotes class 2) for the i-th example, and n is the number of training examples.[4] The gradient of f_{RLS} is: $g_{RLS}(\beta) = \lambda_2\beta + X'(X\beta - t)$ where X is the $n \times m$ matrix whose rows have $\{x_i'\}$ and t is the target vector containing $\{t_i\}$.

Model 2. *SVM (with L_2 loss)*

$$f(\beta) = f_{SVM}(\beta) = \frac{\lambda_2}{2}\|\beta\|^2 + \frac{1}{2}\sum_{i \in I(\beta)} r_i^2(\beta) \tag{3}$$

where $I(\beta) = \{i : t_i r_i(\beta) < 0\}$ and the other quantities are as defined for f_{RLS}. It is easy to check that f_{SVM} is continuously differentiable and that its gradient is given by: $g_{SVM}(\beta) = \lambda_2\beta + X_I'(X_I\beta - t_I)$ where X_I is the matrix whose rows have $\{x_i', i \in I(\beta)\}$ and t_I is the vector containing $\{t_i, i \in I(\beta)\}$.

SVM with hinge loss corresponds to replacing the $r_i^2(\beta)$ term in (3) by $-2t_i r_i(\beta)$. When there are no severe outliers in the training set, the SVM models using the L_2 and hinge loss produce very similar classifiers. We work with L_2 loss because our ideas for feature selection only apply to models for which f is continuously differentiable. SVM with modified Huber loss, which also has this property, is another excellent alternative that can be used together with our LARS ideas. In this paper we will present our ideas only for the L_2 loss. By doing minor modifications, these ideas can be easily extended to the modified Huber loss.

Lasso (Tibshirani, 1996) and Elastic Net (Zou & Hastie, 2005) are feature selection tools that apply to f_{RLS}. (Lasso is designed for $\lambda_2 = 0$.) They do systematic feature selection by including an L_1 regularizer in the cost function:

$$\min \tilde{f}(\beta) = f(\beta) + \lambda_1\|\beta\|_1 \tag{4}$$

where $\|\beta\|_1$ is the L_1 norm of β. When λ_1 is large[5] $\beta = 0$ is the minimizer of $\tilde{f}$, which corresponds to the case of all variables being excluded. As λ_1 is decreased, more and more variables take positive values. When $\lambda_1 \to 0$, the solution of (4) approaches the minimizer of f.

In the above approach, as λ_1 is decreased, a β_j can move back and forth between zero and non-zero values more than once. LARS is a closely related, but computationally much simpler approach that was devised by Efron et al (Efron et al., 2004) for ordinary least squares problems (i.e., f_{RLS} with $\lambda_2 = 0$); this approach only keeps adding variables as a parameter similar to λ_1 is decreased.[6]

The basic idea behind LARS is simple. Though not mentioned in Efron et al. (2004), LARS can be easily extended to other models such as f_{SVM}. We will refer to the extension also as LARS. To understand the method, first note that $\beta^\star$ is a minimizer of f iff it is a (global) minimizer of the function $g_{\max}(\beta) = \max_j |g_j(\beta)|$. LARS starts from $\beta = 0$ and continuously decreases $g_{\max}$ by adding variables one at a time until $\beta^\star$ is reached. At $\beta = 0$ let j_1 be the index which defines $g_{\max}$. Take β_{j_1} as the first variable (at the current point it is the variable that contributes most to the decrease in f) to be included. Let $s_{j_1} = \text{sgn}(g_{j_1}(0))$. If we start from $\lambda_1 = g_{\max}(0)$ and track the curve defined by $g_{j_1} = \lambda_1 s_{j_1}$ with β_{j_1} and λ_1 as the only variables allowed to change, then $g_{\max}$ can be decreased.[7] This can be continued until we reach a $\tilde{\lambda}_1$ and $\tilde{\beta}$ where another variable index j_2 also defines $g_{\max}$, i.e., $|g_{j_2}(\tilde{\beta})| = |g_{j_1}(\tilde{\beta})| = \tilde{\lambda}_1$. At this

[3]The model is same as ridge regression on the classifier targets.

[4]We assume throughout this paper that the bias term is built into β and that it is also regularized.

[5]It is easy to see that, if $\lambda_1 > \max_j |g_j(0)|$, then the directional derivatives of the $\lambda_1\|\beta\|_1$ term at $\beta = 0$ dominate and so $\beta = 0$ is the minimizer of $\tilde{f}$.

[6]Since we use only LARS in this paper, we will call this parameter for LARS also as λ_1.

[7]At any stage of the LARS algorithm λ_1 equals $g_{\max}$. Thus, decreasing λ_1 causes $g_{\max}$ to decrease.

point LARS includes β_{j_2} as the new variable for inclusion and tracks the curve defined by the two equations, $g_{j_1} = \lambda_1 s_{j_1}$, $g_{j_2} = \lambda_1 s_{j_2}$, where $s_{j_2} = \mathrm{sgn}(g_{j_2}(\tilde{\beta}))$ and, β_{j_1}, β_{j_2} and λ_1 are the only variables allowed to change. The procedure is repeated to include more variables while $g_{\max}$ is continuously decreased.

Because the choice of new variables is done conditional on variables already included, LARS has the potential to be better than filter methods which analyze features independently. Since the importance of variables is judged by the size of the derivative of f, it is necessary to scale the variables uniformly. Most representations employed in text classification (binary, normalized binary, normalized tf-idf etc) have such a uniform scaling. They all give data values spread in the (0,1) range. Let us now give full details of the (generalized) LARS algorithm.

Algorithm LARS

1. Let $\bar{\beta} = 0$. Find $\bar{j} = \arg\max_j |g_j(\bar{\beta})|$. Let $\mathcal{A} = \{\bar{j}\}$, $s_{\bar{j}} = \mathrm{sgn}(g_{\bar{j}}(\bar{\beta}))$, $\bar{\lambda}_1 = |g_{\bar{j}}(\bar{\beta})|$. Go to step 2.

2. Consider the curve P in (β, λ_1) space defined by

$$P = \{(\beta,\lambda_1) : g_j = s_j \lambda_1 \ \forall j \in \mathcal{A}, \ \beta_k = 0 \ \forall k \notin \mathcal{A}, \ \lambda_1 \le \bar{\lambda}_1\} \tag{5}$$

Find

$$\tilde{\lambda}_1 = \sup\{ \ \lambda_1 : (\beta,\lambda_1) \in P, \ |g_k| \ge \lambda_1 \text{ for some } k \notin \mathcal{A} \ \} \tag{6}$$

Let the corresponding β be $\tilde{\beta}$. If $\tilde{\lambda}_1 = 0$, stop. If $\tilde{\lambda}_1 \neq 0$, then by continuity arguments there exists $\tilde{k} \notin \mathcal{A}$ for which $|g_k| = \tilde{\lambda}_1$; obtain $\tilde{k}$ and go to step 3.

3. Let $\mathcal{A} := \mathcal{A} \cup \{\tilde{k}\}$, $\bar{\beta} = \tilde{\beta}$, $s_{\tilde{k}} = \mathrm{sgn}(g_{\tilde{k}}(\tilde{\beta}))$ and go back to step 2.

Consider a generic stage of the algorithm at which $\bar{\beta}$ satisfies $\bar{g}_j = s_j \lambda_1 \ \forall j \in \mathcal{A}$ and $\bar{\beta}_j = 0 \ \forall j \notin \mathcal{A}$. It is easy to see that $\bar{\beta}$ is the minimizer of $f - s'_{\mathcal{A}}\beta_{\mathcal{A}}$ where only $\beta_{\mathcal{A}}$ is varied, and $s_{\mathcal{A}}$ is the $\mathcal{A}$ dimensional vector containing $\{s_j, j \in \mathcal{A}\}$.

Suppose we decide to run the algorithm only upto a point where d variables have been chosen and so stop at the end of step 2 when $|\mathcal{A}| = d$ occurs. At that point what is the β that should be used as the parameter vector for the classifier? One idea is to use $\beta = \tilde{\beta}$. *This is the parameter vector we have used in all the computational experiments reported in this paper.* Since we are terminating the algorithm with only a partial set of variables, $g(\tilde{\beta}) \neq 0$. We can take the d parameters defined by $\mathcal{A}$, optimize f with respect to these parameters and obtain the parameter vector, $\hat{\beta}$. This is another worthwhile possibility for use in the final classifier. For ordinary least squares (f_{RLS} with $\lambda_2 = 0$), Efron et al (Efron et al., 2004) refer to this alternative as the OLS/LARS hybrid. One can also think of fundamentally altering the algorithm by solving $g_j = 0$, $j \in \mathcal{A}$ to obtain $\bar{\beta}$, $j \in \mathcal{A}$, and then choosing the next entering variable to be the one which has the largest magnitude gradient component. Such an aggressive forward selection scheme can be overly greedy, leading to a poor selection of features (Efron et al., 2004).

Let us now discuss the computer implementation of the above algorithm. Let us begin with $f = f_{RLS}$ since the ideas are simpler for it. Although details for the least squares case are given in Efron et al. (2004) and Zou and Hastie (2005), we repeat them here since they smoothly lead us to the details for $f = f_{SVM}$. Let $\beta_{\mathcal{A}}$ denote the $|\mathcal{A}|$ dimensional vector containing $\{\beta_j, j \in \mathcal{A}\}$. For $f = f_{RLS}$, the g_j are affine functions of $\beta_{\mathcal{A}}$ and so P is a linear curve. The direction of that linear curve, $\delta\beta_{\mathcal{A}}$ can be determined by solving the linear system,

$$H_{\mathcal{A}} \delta\beta_{\mathcal{A}} = s_{\mathcal{A}} \tag{7}$$

where: $H_{\mathcal{A}} = \lambda_2 I_{|\mathcal{A}|} + X'_{\mathcal{A}} X_{\mathcal{A}}$; $I_{|\mathcal{A}|}$ is the identity matrix of size $|\mathcal{A}|$; $X_{\mathcal{A}}$ is the $n \times |\mathcal{A}|$ submatrix of the data matrix X corresponding to the variables defined by $\mathcal{A}$; and $s_{\mathcal{A}}$ is a $|\mathcal{A}|$ dimensional vector containing $\{s_j, j \in \mathcal{A}\}$. The computational algorithm can now be easily given. By the way $\delta\beta_{\mathcal{A}}$ is defined in (7) note that $\beta_{\mathcal{A}}$ has to be moved in the negative $\delta\beta_{\mathcal{A}}$ direction in order to decrease λ_1.

Algorithm RLS-LARS

1. Let $\bar{\beta} = 0$. Compute $\bar{g} = -X't$ and find $\bar{j} = \arg\max_j |\bar{g}_j|$. Let $\mathcal{A} = \{\bar{j}\}$, $s_{\bar{j}} = \mathrm{sgn}(\bar{g}_{\bar{j}})$, $\bar{\lambda}_1 = |\bar{g}_{\bar{j}}|$, $L = \sqrt{H_{\mathcal{A}}}$ where $H_{\mathcal{A}} = \lambda_2 + X'_{\mathcal{A}} X_{\mathcal{A}}$. (Since $\mathcal{A}$ is a singleton, $H_{\mathcal{A}}$ is a real number. Throughout the algorithm L will denote the lower triangular Cholesky factorization of $H_{\mathcal{A}}$.) Go to step 2.

2. (a) Solve $LL'\delta\beta_{\mathcal{A}} = s_{\mathcal{A}}$ (two triangular linear systems) to get the direction vector $\delta\beta_{\mathcal{A}}$. Then compute $\delta g = X'X_{\mathcal{A}}\delta\beta_{\mathcal{A}}$, the change in g caused by $\delta\beta_{\mathcal{A}}$.
   (b) For each $k \notin \mathcal{A}$: solve $\bar{g}_k + (\lambda_1 - \bar{\lambda}_1)\delta g_k = \lambda_1$ to get λ^+_{1k}; solve $\bar{g}_k + (\lambda_1 - \bar{\lambda}_1)\delta g_k = -\lambda_1$ to get λ^-_{1k}; and then set $\tilde{\lambda}_{1k} = \min_+\{\tilde{\lambda}^+_{1k}, \tilde{\lambda}^-_{1k}\}$ where $\min_+$ means that, when the minimum is taken, negative quantities are ignored. (As

λ_1 is decreased from $\bar{\lambda}_1$ and P is tracked, $\tilde{\lambda}_{1k}$ is the first λ_1 value at which $|g_k| = \lambda_1$ occurs.)

(c) Let $\tilde{\lambda}_1 = \max\{\tilde{\lambda}_{1k} : k \notin \mathcal{A}\}$ and $\tilde{k}$ be the k which defines $\tilde{\lambda}_1$. (If $\mathcal{A} = \{1, \ldots, m\}$ then simply set $\tilde{\lambda}_1 = 0$; in this case, $\tilde{k}$ is undefined.) Go to step 3.

3. Reset $\bar{\beta}_{\mathcal{A}} := \bar{\beta}_{\mathcal{A}} + (\tilde{\lambda}_1 - \bar{\lambda}_1)\delta\beta_{\mathcal{A}}$, $\bar{g} := \bar{g} + (\tilde{\lambda}_1 - \bar{\lambda}_1)\delta g$, and $\bar{\lambda}_1 := \tilde{\lambda}_1$. If $\bar{\lambda}_1 = 0$ stop. Else, set $\mathcal{A} := \mathcal{A} \cup \{\tilde{k}\}$, $s_{\tilde{k}} = \text{sgn}(\bar{g}_{\tilde{k}})$, incrementally update the cholesky factor L to include the new variable and go back to step 2.

In text classification problems the number of variables is large and so the above algorithm will be impractical if a complete ordering of the full set variables is to be carried out. But, there is rarely a need to run the above algorithm beyond the selection of about 500-1000 top variables. In many cases the classifier performance peaks within that selection; if the performance continues to rise even beyond, that is usually an indication that using the full set of variables is the best way to go. Therefore it is sufficient to run the algorithm till about 500-1000 variables are chosen.

Let us do a complexity analysis for an early termination of the algorithm when it is run till d variables are chosen. Let $m_{\mathcal{A}} = |\mathcal{A}|$ and n_{nz} denote the number of non-zero elements in X. In textual problems X is usually very sparse and so n_{nz} is a small fraction of nm (which is the value of n_{nz} for a dense X matrix). For our analysis we can take n and m to be smaller than n_{nz}. The main cost of the algorithm is associated with steps 2(a) and (3), which respectively have the complexities, $O(m_{\mathcal{A}}^2 + n_{nz})$ and $O(m_{\mathcal{A}}^2)$. Accumulating upto d variables we get the algorithm's complexity upto d variables as $O(d^3 + n_{nz}d)$. To get an idea of the times involved, take a problem with about 10,000 examples having 100 non-zero elements in each example, i.e., n_{nz} is about 1 million. If $d = 1000$ then d^3 and $n_{nz}d$ are about the same order and so the overall cost of running the algorithm upto 1000 variables is only about few times the cost of inverting a 1000 dimensional dense square matrix.

Let us now turn to the extension of the algorithm to SVM. For f_{SVM}, the g_j are piecewise-affine functions of $\beta_{\mathcal{A}}$ and so the curve P in (5) is piecewise-linear. Rosset and Zhu (Rosset & Zhu, 2004) discuss a range of models for which the overall curve parametrized with respect to λ_1 is piecewise-linear. Though they do not explicitly include f_{SVM} in their list of such models, all their ideas can be easily extended to f_{SVM} too. Let us now take up these details. Apart from identifying λ_1 values at which a new variable enters, we also need to locate points at which an example leaves or enters the active index set, I. The following algorithm gives all the details. Many of the steps are same as those in algorithm RLS-LARS. In the algorithm we use y_i to denote $t_i r_i$; thus, $y_i < 0$ means that $i \in I$.

Algorithm SVM-LARS

1. Let $\bar{\beta} = 0$, $I = \{1, \ldots, n\}$ and $\bar{y}_i = -1$, $i = 1, \ldots, n$. Compute $\bar{g} = -X't$ and find $\bar{j} = \arg\max_j |\bar{g}_j|$. Let $\mathcal{A} = \{\bar{j}\}$, $s_{\bar{j}} = \text{sgn}(\bar{g}_{\bar{j}})$, $\bar{\lambda}_1 = |\bar{g}_{\bar{j}}|$, $L = \sqrt{H_{\mathcal{A}}}$ where $H_{\mathcal{A}} = \lambda_2 + X'_{\mathcal{A}}X_{\mathcal{A}}$. Go to step 2.

2. (a) Solve $LL'\delta\beta_{\mathcal{A}} = s_{\mathcal{A}}$ to get the direction vector $\delta\beta_{\mathcal{A}}$. Then compute $\delta r = X_{\mathcal{A}}\delta\beta_{\mathcal{A}}$, and then, $\delta g = X'\delta r$.

   (b) For each $k \notin \mathcal{A}$: solve $\bar{g}_k + (\lambda_1 - \bar{\lambda}_1)\delta g_k = \lambda_1$ to get λ_{1k}^+; solve $\bar{g}_k + (\lambda_1 - \bar{\lambda}_1)\delta g_k = -\lambda_1$ to get λ_{1k}^-; and then set $\tilde{\lambda}_{1k} = \min_+\{\tilde{\lambda}_{1k}^+, \tilde{\lambda}_{1k}^-\}$.

   (c) Let $\tilde{\lambda}_1 = \max\{\tilde{\lambda}_{1k}, k \notin \mathcal{A}\}$ and $\tilde{k}$ be the k which defines $\tilde{\lambda}_1$. (If $\mathcal{A} = \{1, \ldots, m\}$ then simply set $\tilde{\lambda}_1 = 0$; in this case, $\tilde{k}$ is undefined.)

   (d) For each $i = 1, \ldots, n$, compute $\delta y_i = t_i \delta r_i$ and λ_{1i}, the solution of $\bar{y}_i + (\lambda_1 - \bar{\lambda}_1)\delta y_i = 0$. (As $\beta_{\mathcal{A}}$ is changed from $\bar{\beta}_{\mathcal{A}}$ along the line defined by the $\delta\beta_{\mathcal{A}}$ direction, λ_{1i} is the λ_1 value at which the i-th example will hit the margin plane corresponding to it.)

   (e) Find $\hat{\lambda}_{1a}$, the first $\lambda_1 \leq \bar{\lambda}_1$ at which an example from group I moves out of I, i.e., $\hat{\lambda}_{1a} = \max\{\lambda_{1i} : i \in I, \delta y_i < 0\}$. Let $\hat{i}_a$ be the $i \in I$ that defines $\hat{\lambda}_{1a}$. (If, either $\hat{\lambda}_{1a} < 0$ or $\nexists\ i \in I$ such that $\delta y_i < 0$, then simply reset $\hat{\lambda}_{1a} = 0$. For this case, the value of $\hat{i}_a$ is immaterial and can be left undefined.)

   (f) Find $\hat{\lambda}_{1b}$, the first $\lambda_1 \leq \bar{\lambda}_1$ at which an example from outside group I moves into I, i.e., $\hat{\lambda}_{1b} = \max\{\lambda_{1i} : i \notin I, \delta y_i > 0\}$. Let $\hat{i}_b$ be the $i \notin I$ that defines $\hat{\lambda}_{1b}$. (If, either $\hat{\lambda}_{1b} < 0$ or $\nexists\ i \notin I$ such that $\delta y_i > 0$, then simply reset $\hat{\lambda}_{1b} = 0$. For this case, the value of $\hat{i}_b$ is immaterial and can be left undefined.)

   (g) If $\hat{\lambda}_{1a} \geq \hat{\lambda}_{1b}$ then set $\hat{\lambda}_1 = \hat{\lambda}_{1a}$, $\hat{i} = \hat{i}_a$; else set $\hat{\lambda}_1 = \hat{\lambda}_{1b}$, $\hat{i} = \hat{i}_b$;

   (h) If $\tilde{\lambda}_1 \geq \hat{\lambda}_1$ go to step 3; else go to step 4.

3. Reset $\bar{\beta}_{\mathcal{A}} := \bar{\beta}_{\mathcal{A}} + (\tilde{\lambda}_1 - \bar{\lambda}_1)\delta\beta_{\mathcal{A}}$, $\bar{y} := \bar{y} + (\tilde{\lambda}_1 - \bar{\lambda}_1)\delta y$, $\bar{g} := \bar{g} + (\tilde{\lambda}_1 - \bar{\lambda}_1)\delta g$, and $\bar{\lambda}_1 = \tilde{\lambda}_1$. If $\bar{\lambda}_1 = 0$ stop. Else, set $\mathcal{A} := \mathcal{A} \cup \{\tilde{k}\}$, $s_{\tilde{k}} = \text{sgn}(\bar{g}_{\tilde{k}})$, incre-

mentally update the cholesky factor L to include the new variable and go back to step 2.

4. Reset $\bar{\beta}_{\mathcal{A}} := \bar{\beta}_{\mathcal{A}} + (\hat{\lambda}_1 - \bar{\lambda}_1)\delta\beta_{\mathcal{A}}$, $\bar{y} := \bar{y} + (\hat{\lambda}_1 - \bar{\lambda}_1)\delta y$, $\bar{g} := \bar{g} + (\hat{\lambda}_1 - \bar{\lambda}_1)\delta g$, and $\bar{\lambda}_1 = \hat{\lambda}_1$. If $\hat{\lambda}_1 = 0$ stop. Else, do the following. If $\hat{i} \in I$, remove $\hat{i}$ from I, incrementally update the cholesky factor L (see text below) to remove the effect of example $\hat{i}$; else, include $\hat{i}$ in I, incrementally update the cholesky factor L (see text below) to include the effect of example $\hat{i}$. Go back to step 2.

The Cholesky update in step 4 can be implemented as follows. Let $\hat{x}$ be the $|\mathcal{A}|$ dimensional vector formed by taking the $\hat{i}$-th example, $x_{\hat{i}}$ and keeping only the elements corresponding to $\mathcal{A}$. Then the update corresponds to forming the Cholesky factors of $LL' \pm \hat{x}\hat{x}'$. This can be accomplished in $O(|\mathcal{A}|^2)$ effort.

Though the complexity of one loop of steps 2-4 of SVM-LARS is same as the complexity of one loop of steps 2-3 of RLS-LARS, the overall complexity of SVM-LARS is higher than that of RLS-LARS due to the number of loops executed. Typically, examples only move out of I as the algorithm proceeds (note that, at the beginning, $I = \{1, \ldots, n\}$). For our analysis here, let us assume that this is true. Let n_{ch} be the number of examples that move out of I by the time d variables have been included. The total number of loops of steps 2-4, then, is $d + n_{ch}$. The total cost of the algorithm is bounded by $O((n_{nz} + d^2)(n_{ch} + d))$. Let n_{sv} denote the number of support vectors (i.e., $|I|$) in the SVM which uses all the variables. Then we have $n_{ch} \leq (n - n_{sv})$. This bound gives us a good estimate of n_{ch}.

For problems in which n_{ch} is large (n_{sv} is small) the extra cost of SVM-LARS over RLS-LARS (i.e., $O((n_{nz} + d^2)n_{ch})$) can be big. There is a simple way to reduce this extra cost considerably while bringing in some approximateness to the algorithm that is not serious. For lack of space, we skip all these details here. We have also developed an efficient method for tracking an approximation of the leave-one-out error as a function of λ_1. For lack of space we skip these details too. This approximation serves well for selecting the number of features to use.

3. Empirical Analysis

In this section we empirically evaluate the usefulness of LARS as a feature selection tool for SVM and RLS, and, in the process, also compare SVM and RLS as well as study the usefulness of keeping (or leaving out) the L_2 regularizer term in f. Although we have conducted the evaluation on many datasets, here we only report representative results on the following datasets.[8]

- *Fbis*-4: 1 binary problem of the *Fbis* dataset corresponding to class 4 (with 387 examples) against the rest; $n = 2463$; $m = 2000$.
- *La1*-2, *La1*-4: 2 binary problems of the *La1* dataset corresponding to class 2 (with 555 examples) and class 4 (with 943 examples) against the rest; $n = 3204$; $m = 31472$.
- *Ohscal*-7: 1 binary problem if the *Ohscal* dataset corresponding to class 7 (with 1037 examples) against the rest; $n = 11162$; $m = 11465$.
- *Reuters-Money-fx* : 1 binary problem of the *Reuters* dataset corresponding to the class *money-fx* (with 717 examples) against the rest; $n = 12902$; $m = 37207$.
- *News-3* : 1 binary problem of the *20 Newsgroups* dataset corresponding to class 3 (with 1000 examples) against the rest; $n = 19928$; $m = 62061$.

All these problems have a reasonably rich number of examples in each class. We study generalization performance as a function of the number of features. F-measure is used as the generalization performance metric. To evaluate the set of features obtained by LARS we compare it (in terms of the F-measure) against the set of features obtained by ordering according to information gain (IG).[9]

The following six methods were compared: SVM-LARS ($\lambda_2 = 1$); SVM-LARS ($\lambda_2 = 0$); SVM-IG; RLS-LARS ($\lambda_2 = 1$); RLS-LARS ($\lambda_2 = 0$); and, RLS-IG. Note that, setting $\lambda_2 = 0$ with SVMs does define a meaningful model since the L_1 regularizer is present. For RLS, setting $\lambda_2 = 0$ corresponds to the Lasso model (Tibshirani, 1996). For the case of non-zero λ_2, we did not tune λ_2, mainly to keep the computations

[8] *Fbis*, *La1* and *Ohscal*, which use binary representation for the words, are as in Forman (2003). The *Reuters* dataset is taken from (Lewis, 2004) and *20 Newsgroups* is as in Rennie and Rifkin (2001). For these two datasets we use normalized tf-idf representation.

[9] One could also consider other good filter methods such as BNS (Bi-Normal Separation) (Forman, 2003). We tried BNS, but it did not perform as well as IG on the datasets chosen by us. This is not inconsistent with the findings of Forman (Forman, 2003) who found BNS to be better, overall, on a range of problems in which many had large class skew. The binary problems chosen by us do not have big class skew.

manageable; also, it has been generally observed that, for text classification problems $\lambda_2 = 1$ is a good choice that gives a performance close to optimal. For each classifier used in our study we do a post-processing tuning of the threshold to optimize the F-measure. The validation outputs needed for doing this were obtained using the LOO approximation mentioned in section 2 for LARS and using 10-fold cross validation for IG ordering.[10]

Since the training set/testing set split can cause variability that needs to be shown for a proper comparison of the methods, we set-up the comparison experiments as follows. Each binary dataset was randomly divided into 4 (classwise stratified) groups; 4 runs were made, each time keeping one group as the testing set and the remaining 3 groups as the training set. Thus, for each classifier method and a given number of features selected, we get 4 F-measure values as estimates of generalization performance. Let $F_{\min}$, F_{mean} and $F_{\max}$ denote the minimum, mean and maximum of these four values. The experiment was repeated for 10 different choices of the random 4-groupings; $F_{\min}$, F_{mean} and $F_{\max}$ were averaged over the 10 runs to get the averaged values, $avF_{\min}$, avF_{mean} and $avF_{\max}$. While avF_{mean} gives an estimate of the expected F-measure value, $avF_{\max} - avF_{\min}$ gives us an estimate of its variability. For each method, we plot $avF_{\min}$, avF_{mean} and $avF_{\max}$ as a function of the number of features chosen. The analysis was done upto a maximum of 2000 chosen features. We also ran SVM and RLS using all the features in a given dataset and $\lambda_2 = 1$ to get another baseline for comparison. Only for the $Reuters - Money - fx$ dataset, we did not form any random groupings; going by standard practice for the Reuters dataset, we simply use the ModApte train/test split once.

For the six binary problems and the various methods, Figure 1 gives plots of the F-measure statistics as a function of the number of features chosen. For the LARS approaches, the number of features is a non-linear scaling of λ_1. (Note that, as λ_1 decreases more features get included.) Also, since features are sequentially added one at a time, the performance values for LARS are available for every integer value of the number of features. Since we are using F-measure, bigger values mean better performance. For the IG ordering we fixed $\lambda_2 = 1$ and only did the experiments for the following values of the number of features: 2, 4, 8, 16, 32, 64, 128, $200 + k * 150$, $k = 0, 1, \ldots, 12$. Figure 1 also gives the statistics associated with the classifier which uses all features and has $\lambda_2 = 1$.

We can group the findings under three headings.

LARS versus IG. Although the initial, small set of features chosen by IG is very good (sometimes these initial features are even quite better than those chosen by LARS), LARS usually does much better in the middle phase where the performance either peaks or is approaching the peak. This superiority is very striking in some cases; see, for instance, the performance on *Ohscal*-7. In several cases, peak performance is attained only when the number of features chosen is very large. Even in such problems, LARS usually does much better in pointwise comparison at various values of the number of features. This win makes LARS to be very useful in situations where there is a need to build classifiers using only a restricted number of features. *La1*-2 is one such case.

SVM versus RLS. In many cases, the performance of RLS is severely degraded by the inclusion of a large number of features. This is regardless of whether LARS or IG was used for feature selection. Rarely, a lesser degree of such degradation occurs with SVM too (see *Ohscal*-7, for instance). In terms of the peak performance achieved, SVM usually did quite better; also, SVM achieved the peak with less number of features.

Keeping ($\lambda_2 = 1$) versus Leaving out ($\lambda_2 = 0$) the L_2 regularizer. In the initial phase of feature selection, the two classifiers are usually identical. This is because, in this phase, the data-fit term gets the maximum importance. After the gradient of the data-fit term has reached small values, the classifier with the L_2 regularizer term concentrates on bridging the effects of the regularizer term and the data-fit term, while the classifier without the L_2 regularizer term continues to pick up features to aggressively reduce the gradient of the data-fit term towards zero, leading to overfitting and a severe loss in generalization performance. It can also be seen that the number of features at which the $\lambda_2 = 0$ classifier falls to low values is generally much smaller for SVM than for RLS. This can be easily explained by the fact that, while RLS always concentrates on all the examples, the SVM only has to fit the active examples (the elements of I) in its 'least squares' process. Thus the SVM can achieve this fit using much less features.

In some cases, peak generalization performance is achieved in the initial phase of feature selection itself; see *Fbis*-4, for instance. In such cases the absence of the L_2 regularizer is harmless. But, in several cases

[10]Like Genkin et al (Genkin et al., 2004) one could also simply use the training outputs to adjust the threshold. This will reduce the computational cost, but the performance of the chosen threshold will be slightly inferior.

(see the SVM case in *La1*-2, for example) the performance of the classifier with the L_2 regularizer keeps rising as more features are added. For such problems, leaving out the L_2 regularizer is clearly a poor choice.

There are also some interesting cases (*Reuters-Money-fx* ; *News-3*) where, in the initial phase of feature selection, leaving out the L_2 regularizer gives much better performance.[11] Genkin et al (Genkin et al., 2004) obtained very good results on the Reuters' binary problems using their logistic regression method with the L_1 regularizer. Our LARS method corresponding to leaving out the L_2 regularizer is very similar to their method; the two methods mainly differ in the loss function employed and the way λ_1 is tuned. Our experiments indicate that it is safer to try, both $\lambda_2 = 1$ and $\lambda_2 = 0$, and choose the better one, say, based on cross validation.

4. Conclusion

In this paper we applied generalized LARS to linear SVMs and showed that this leads to effective feature selection. SVM-LARS is close in spirit to an SVM model in which both, L_2 and L_1 regularizers are present. An important advantage of the SVM-LARS algorithm is its ability to finely track the solution with respect to λ_1 without losing efficiency. The model without the L_2 regularizer is also an interesting model that is worth considering. We are working on developing an efficient algorithm for applying generalized LARS to logistic regression. For this case, the solution curve with respect to λ_1 is a nonlinear curve, and hence the algorithmic aspects are more challenging.

In our empirical study we used only the standard 'Bag-Of-Words' (BOW) representation for forming the features. It is also possible to include other derived features, say, the distributional word cluster features considered by Bekkerman et al (Bekkerman et al., 2003), who found that their cluster representation is better than that of BOW on some datasets, but worse on others. By putting the BOW features and the derived features together and using LARS to do feature selection, there is hope for getting the best properties of both representations.

[11]This can be very useful in situations where there is a need to work with a limited number of features. But, if that is not the case, keeping the L_2 regularizer could be the better option. Note, for example, in the case of *News-3* that, the performance of the L_2 regularizer using all features is much better than the peak performance of the classifier with $\lambda_2 = 0$.

References

Bekkerman, R., El-Yaniv, R., Tishby, N., & Winter, Y. (2003). Distributional word clusters vs. words for text categorization. *Journal of Machine Learning Research*, *3*, 1183–1208.

Efron, B., Hastie, T., Johnstone, T., & Tibshirani, R. (2004). Least angle regression. *Annals of Statistics*, *32*, 407–499.

Forman, G. (2003). An extensive empirical study of feature selection metrics for text classification. *Journal of Machine Learning Research*, *3*, 1289–1305.

Genkin, A., Lewis, D. D., & Madigan, D. (2004). Large-scale bayesian logistic regression for text categorization.

Joachims, T. (1998). Text categorization with support vector machines: Learning with many relevant features. *Proceedings of the Tenth European Conference on Machine Learning (ECML)* (pp. 137–142).

Lewis, D. D. (2004). *Reuters-21578 text categorization test collection: Distribution 1.0 readme file (v 1.3)* (Technical Report). www.daviddlewis.com/resources/.

Rennie, J. D. M., & Rifkin, R. (2001). *Improving multiclass text classification with the support vector machine* (Technical Report). Artificial Intelligence Lab, MIT.

Rosset, S., & Zhu, J. (2004). Piecewise linear regularized solution paths.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, B*, *58*, 267–288.

Yang, Y., & Pedersen, J. O. (1997). A comparative study on feature selection in text categorization. *Proceedings of the 14th International Conference on Machine Learning* (pp. 412–420).

Zou, H., & Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society, B.*

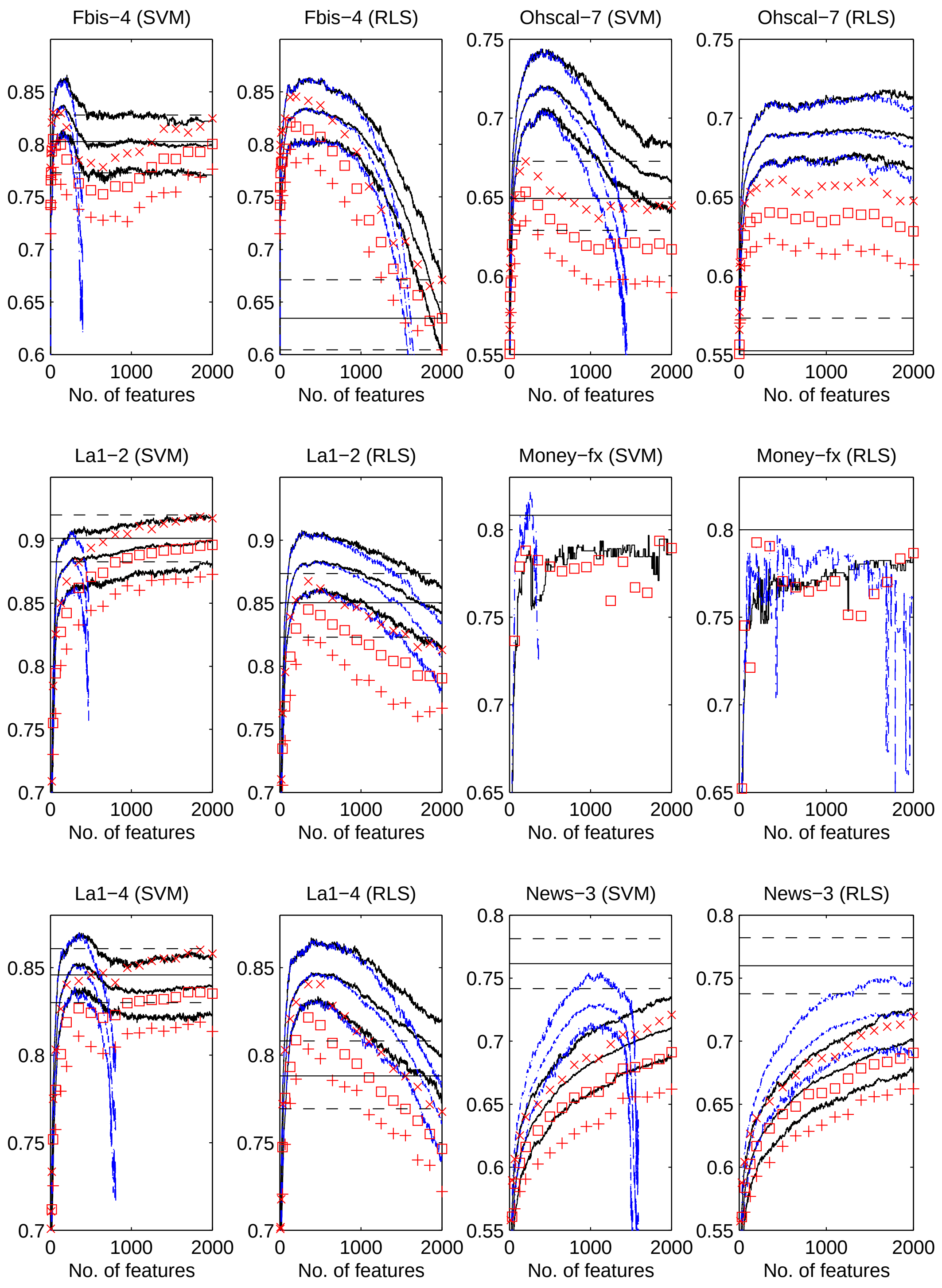

Figure 1. F-measure, as a function of the number of features chosen, for the six binary problems. For each problem there are two plots: the left one has the SVM results and the right one has the RLS results. In each plot, the *avF*-measure values of LARS with $\lambda_2 = 1$ are given as continuous (black) lines and the avF-measure values of LARS with $\lambda_2 = 0$ are given by (blue) broken lines. The avF_{min}, avF_{mean} and avF_{max} values for the IG ordering with $\lambda_2 = 1$ are respectively shown by the (red) symbols, $+$, $\square$ and $\times$. The avF_{min} and avF_{max} values for the classifier with $\lambda_2 = 1$, using all the features are shown by dotted (black) horizontal lines; the corresponding avF_{mean} value is shown by a continuous horizontal (black) line.