Feature Selection for Nonlinear Kernel Support Vector Machines

Olvi L. Mangasarian and Edward W. Wild
Computer Sciences Department
University of Wisconsin
Madison, WI 53706
{*olvi,wildt*}*@cs.wisc.edu*

Abstract

An easily implementable mixed-integer algorithm is proposed that generates a nonlinear kernel support vector machine (SVM) classifier with reduced input space features. A single parameter controls the reduction. On one publicly available dataset, the algorithm obtains 92.4% accuracy with 34.7% of the features compared to 94.1% accuracy with all features. On a synthetic dataset with 1000 features, 900 of which are irrelevant, our approach improves the accuracy of a full-feature classifier by over 30%. The proposed algorithm introduces a diagonal matrix E with ones for features present in the classifier and zeros for removed features. By alternating between optimizing the continuous variables of an ordinary nonlinear SVM and the integer variables on the diagonal of E, a decreasing sequence of objective function values is obtained. This sequence converges to a local solution minimizing the usual data fit and solution complexity while also *minimizing the number of features used.*

1. Introduction

Feature selection is a fairly straightforward procedure for *linear* support vector machine (SVM) classifiers. For example, a 1-norm support vector machine linear classifier obtained by either linear programming or concave minimization will easily reduce features [1]. However, when similar techniques are applied to *nonlinear* SVM classifiers, the resulting reduction is *not* in the number of input space features but in the number of kernel functions needed to generate the nonlinear classifier [6]. This may be interpreted as a reduction in the dimensionality of the higher dimensional transformed space, but does not result in any reduction of input space features. It is precisely this reduction that we are after in this paper, namely, a reduced number of input space features that we need to input into a nonlinear SVM classifier. We shall achieve this by replacing the usual nonlinear kernel $K(A, A')$, where A is the $m \times n$ data matrix, by $K(AE, EA')$ where E is an $n \times n$ diagonal matrix of ones and zeros. The proposed algorithm alternates between computing the continuous variables (u, γ) of the nonlinear kernel classifier $K(x'E, EA')u - \gamma = 0$, by using linear programming, and the integer diagonal matrix E of ones and zeros by successive minimization sweeps through its components. The algorithm generates a decreasing sequence of objective function values that converge to a local solution that minimizes the usual data fit and the number of kernel functions used while *also* minimizing the number of features used. A possibly related result, justifying the use of reduced features, is that of random projection on a subspace of features for Gaussian mixtures which states that data from a mixture of k Gaussians can be projected into $O(log\ k)$ dimensions while retaining approximate separability of the clusters [3].

There has been considerable recent interest in feature selection for SVMs. Weston et al. propose reducing features based on minimizing generalization bounds via a gradient approach [19]. In [5], Frölich and Zell introduce an incremental approach based on ranking features by their effect on the margin. An approach based on a Bayesian interpretation of SVMs is presented by Gold et al. [7], and an approach based on smoothing spline ANOVA kernels is proposed by Zhang [20]. In [8], Guyon et al. use a wrapper method designed for SVMs. Another possibility is to use a filter method such as Relief [14] in conjunction with an SVM. None of these approaches utilize the straightforward and easily implementable mixed-integer programming formulation proposed here.

We briefly outline the contents of the paper now. In Section 2 we derive the theory behind our reduced feature support vector machine classifier and state our algorithm for generating our RFSVM and its convergence. Section 3 gives computational results on two publicly available datasets that show the effectiveness and utility of RFSVM. In particular, we will show that RFSVM is often able to reduce features by similar percentages to those of established

0-7695-3019-2/07 $25.00 © 2007 IEEE
DOI 10.1109/ICDMW.2007.30

IEEE computer society

techniques for linear classifiers, while maintaining the accuracy associated with nonlinear classifiers. We will further demonstrate that the feature selection and classification accuracy of RFSVM is comparable to two other feature selection methods which can be applied to nonlinear classification. Finally, we will show that RFSVM can handle problems with large numbers of features. Section 4 concludes the paper.

A word about our notation and background material follows. All vectors will be column vectors unless transposed to a row vector by a prime superscript $'$. The scalar (inner) product of two vectors x and y in the n-dimensional real space R^n will be denoted by $x'y$ and the p-norm of x, $(\sum_{i=1}^{n}|x_i|^p)^{\frac{1}{p}}$, will be denoted by $\|x\|_p$. For a matrix $A \in R^{m\times n}$, A_i denotes the ith row of A, while $A_{\cdot j}$ denotes j-th column of A and A_{ij} denotes the element of A in row i and column j. A column vector of ones of arbitrary dimension will be denoted by e. The base of the natural logarithm will be denoted by ε. For $A \in R^{m\times n}$ and $B \in R^{n\times l}$, the kernel $K(A, B)$ is an arbitrary function that maps $R^{m\times n} \times R^{n\times l}$ into $R^{m\times l}$. In particular, if x and y are column vectors in R^n then, $K(x', y)$ is a real number, $K(x', A')$ is a row vector in R^m and $K(A, A')$ is an $m \times m$ matrix. We shall employ the widely used Gaussian kernel [18, 11, 15] $\varepsilon^{-\mu\|A_i-A_j\|_2^2}$, $i, j = 1, \ldots, m$ for all our numerical tests. The notation $E = diag(1 \text{ or } 0)$ denotes a diagonal matrix with ones or zeros on the diagonal while x_+ denotes the nonnegative vector generated from x by setting all its negative components to zero. Cardinality of a vector or matrix denotes the number of its nonzero components. The abbreviation "s.t." stands for "subject to."

2. Reduced Feature Support Vector Machine (RFSVM) Formulation and Algorithm

We consider a given set of m points in the n-dimensional input feature space R^n represented by the matrix $A \in R^{m\times n}$. Each point represented by A_i, $i = 1, \ldots, m$, belongs to class +1 or class -1 depending on whether D_{ii} is 1 or -1, where $D \in R^{m\times m}$ is a given diagonal matrix of plus or minus ones. We shall attempt to discriminate between the classes +1 and -1 by a nonlinear classifier induced by a *completely arbitrary* kernel $K(A, A')$ and parameters $u \in R^m$ and $\gamma \in R$, as follows. The classifier (1) is determined by the nonlinear surface:

$$K(x', A')u - \gamma = 0, \tag{1}$$

which classifies points as belonging to class +1 if $K(x', A')u - \gamma > 0$ and as class -1 if $K(x', A')u - \gamma < 0$.

The parameters $u \in R^m$ and $\gamma \in R$ in the classifier (1) are determined by solving the following linear program [11, 2, 15]:

$$\begin{array}{rl} \min\limits_{u,\gamma,y,s} & \nu e'y + e's \\ \text{s.t.} & D(K(A, A')u - e\gamma) + y \geq e, \\ & -s \leq u \leq s, \\ & y \geq 0. \end{array} \tag{2}$$

Here ν is a positive parameter that weights the misclassification error $e'y = \|y\|_1$ versus the model-simplifying regularization term $e's = \|u\|_1$. In fact, it turns out that minimizing $e's = \|u\|_1$ leads to a minimal number of kernel functions used in the classifier (1) by zeroing components of the variable u [6]. However, our primary concern here is to use as few components of the input space vector x as possible in the nonlinear classifier (1). We proceed to do that now as follows.

We introduce a diagonal matrix $E \in R^{n\times n}$ with ones or zeros on its diagonal. The zeros correspond to suppressed input space features and the ones correspond to features utilized by the nonlinear classifier (1) which we modify as follows:

$$K(x'E, EA')u - \gamma = 0. \tag{3}$$

In turn, the linear program (2) becomes the following mixed-integer nonlinear program:

$$\begin{array}{rl} \min\limits_{u,\gamma,y,s,E} & \nu e'y + e's + \sigma e'Ee \\ \text{s.t.} & D(K(AE, EA')u - e\gamma) + y \geq e, \\ & -s \leq u \leq s, \\ & y \geq 0, \\ & E = diag(1 \text{ or } 0), \end{array} \tag{4}$$

where σ is a positive parameter that that weights the feature suppression term $e'Ee = \sum_{i=1}^{n} E_{ii}$. Mixed-integer programs are basically NP-hard. However, we can easily obtain a local solution by fixing E and solving the resulting linear program (4) for (u, γ, y, s), then fixing (u, γ, y, s) and sweeping through the components of E altering them successively only if such alteration decreases the objective function. Repeating this process leads to the following algorithm which, in addition to suppressing input space features, suppresses components of the variable u because of the 1-norm term in the objective function and hence utilizes a minimal number of kernel function components $K(AE, (EA')_{\cdot j})$, $j = 1, \ldots, m$.

We state our algorithm now. More implementation details are given in Section 3.

Algorithm 2.1. Reduced Feature SVM (RFSVM) Algorithm

(1) Pick a random $E = diag(1 \text{ or } 0)$ with cardinality of E inversely proportional to σ. Pick a fixed integer k, typically very large, for the number of sweeps through E, and a stopping tolerance tol, typically $1e-6$.

(2) Solve the linear program (4) for a fixed E and denote its solution by (u, γ, y, s).

(3) For $\ell = 1, \ldots, kn$ and $j = 1 + (\ell - 1) \bmod n$:

(a) Replace E_{jj} by 1 if it is 0 and by 0 if it is 1.

(b) Compute:

$$f(E) = \nu e'(e - D(K(AE, EA')u - e\gamma))_+ + \sigma e'Ee,$$

before and after changing E_{jj}.

(c) Keep the new E_{jj} only if $f(E)$ decreases by more than tol. Else undo the change in E_{jj}. Go to (a) if $j < n$.

(d) Go to (4) if the total decrease in $f(E)$ is less than or equal to tol in the last n steps.

(4) Solve the linear program (4) for a fixed E and denote its solution by (u, γ, y, s). Stop if objective function decrease of (4) is less than tol.

(5) Go to (3).

Remark 2.2. *We note that $f(E)$ in the RFSVM algorithm is equivalent to $\nu e'y + \sigma e'Ee$ when y takes on its optimal value generated by the first and the next-to-the-last sets of constraints of (4). Note that $f(E)$ still depends on E even for the case when $\sigma = 0$.*

We establish now convergence of the RFSVM algorithm for $tol = 0$, however computationally we use $tol = 1e-6$.

Proposition 2.3. RFSVM Convergence
For $tol = 0$, the nonnegative nonincreasing values of the sequence of objective function values $\{\nu e'y^r + e's^r + \sigma e'E^r e\}_{r=1}^{r=\infty}$, where the superscript r denotes iteration number of step (4) of Algorithm 2.1, converge to $(\nu e'\bar{y} + e'\bar{s} + \sigma e'\bar{E}e)$ where $(\bar{u}, \bar{\gamma}, \bar{y}, \bar{s}, \bar{E})$ is any accumulation point of the sequence of iterates $\{u^r, \gamma^r, y^r, s^r, E^r\}$ generated by Algorithm 2.1. The point $(\bar{u}, \bar{\gamma}, \bar{y}, \bar{s}, \bar{E})$ has the following local minimum property:

$$\begin{array}{rcl} (\nu e'\bar{y} + e'\bar{s} + \sigma e'\bar{E}e) = \min\limits_{u,\gamma,y,s} & \nu e'y + e's + \sigma e'\bar{E}e & \\ \text{s.t.} \quad D(K(A\bar{E}, \bar{E}A')u - e\gamma) + y & \geq & e \\ -s \leq u & \leq & s \\ y & \geq & 0, \end{array} \tag{5}$$

and for $p = 1, \ldots, n$:

$$f(\bar{E}) \leq f(E), \text{ for } E_{pp} = 1 - \bar{E}_{pp}, E_{jj} = \bar{E}_{jj}, j \neq p. \tag{6}$$

Proof. That the sequence $\{\nu e'y^r + e's^r + \sigma e'E^r e\}_{r=1}^{r=\infty}$ converges follows from the fact that it is nonincreasing and bounded below by zero. That (5) is satisfied follows from the fact that each point of the sequence $\{u^r, \gamma^r, y^r, s^r, E^r\}$ satisfies (5) with $(\bar{u}, \bar{\gamma}, \bar{y}, \bar{s}, \bar{E})$ replaced by $\{u^r, \gamma^r, y^r, s^r, E^r\}$ on account of step (4) of Algorithm 2.1. That (6) is satisfied follows from the fact that each point of the sequence $\{E^r\}$ satisfies (6) with $(\bar{E})$ replaced by $\{E^r\}$ on account of step (3) of Algorithm 2.1. Hence every accumulation point $(\bar{u}, \bar{\gamma}, \bar{y}, \bar{s}, \bar{E})$ of $\{u^r, \gamma^r, y^r, s^r, E^r\}$ satisfies (5) and (6). □

It is important to note that by repeating steps (3) and (4) of Algorithm 2.1, a feature dropped in one sweep through the integer variables may be added back in another cycle, and conversely. Thus, our algorithm is not merely a naïve greedy approach because the choices of one iteration may be reversed in later iterations, and we have observed this phenomenon in our experiments. However, cycling is avoided by choosing $tol > 0$, which ensures that the sequence of objective values generated by Algorithm 2.1 is strictly decreasing. It is also important to note that when changing the integer variables in step (3), only the objective function needs to be recomputed, which is much faster than solving the linear program in step (4). In fact, as we shall discuss in Section 3, we have found that the cycle through the integer variables in step (3) tends to be repeated more often than the linear program of step (4). We turn now to computational testing of our approach.

3. Computational Results

We illustrate the effectiveness of our Reduced Feature SVM (RFSVM) on two datasets from the UCI Machine Learning Repository [13] and on synthetic data generated using Michael Thompson's NDCC generator [17]. The UCI datasets are used to compare the feature selection and classification accuracy of RFSVM to the following two algorithms: recursive feature elimination (RFE), a wrapper method designed for SVMs [8], and Relief, a filter method [14]. A feature-reducing linear kernel 1-norm SVM (SVM1) [1], and a nonlinear kernel 1-norm SVM (NKSVM1) [11] with no feature selection are used as baselines. The synthetic NDCC data is used to illustrate the effectiveness of RFSVM on problems with large numbers of features, including a problem with 1000 features, 900 of which are irrelevant.

3.1 UCI Datasets

We use the UCI datasets to compare RFSVM to two other algorithms. RFE and Relief are used to illustrate how

RFSVM maintains classification accuracy for different degrees of feature selection. SVM1 and NKSVM1 are used to establish baselines for feature selection and classification accuracy. For the sake of efficiency, we use the experimental methodology described below to compare the algorithms. We first briefly describe RFE and Relief.

3.1.1. RFE. Recursive Feature Elimination (RFE) is a wrapper method designed for SVMs [8]. First an SVM (u, γ) is learned using all features, then features are ranked based on how much the margin $u'K(A, A')u$ changes when each feature is removed separately. Features which have a small effect on the margin are considered less relevant. A given percentage of the least relevant features are removed, and the entire procedure is repeated with the remaining features. In our experiments, we remove one feature at a time until the reported number of features is reached. Note that our RFSVM procedure uses the objective value $f(E)$ to determine whether to include or remove each feature, and removes or keeps features if the objective function decreases by more than a threshold, without first ranking the features. Furthermore, once a feature is removed by RFE it is never again considered for inclusion in the final classifier, while any feature removed during a sweep through the integer variables E in our Algorithm 2.1 may be included by a later sweep.

3.1.2. Relief. Relief is a filter method for selecting features [14]. Features are ranked by computing weights as follows. For a randomly chosen training example, find the nearest example with the same class (the *nearest hit*), and the nearest example in the other class (the *nearest miss*). Then update the weight of each feature by subtracting the absolute value of the difference in feature values between the example and the nearest hit, and adding the absolute value of the difference between the example and the nearest miss. This procedure is then repeated several times, with a different random example each time. Features with high weight are considered more relevant. Relief may be used with any binary classification algorithm, but in the following we use it exclusively with a 1-norm Gaussian kernel SVM.

3.1.3. Methodology. To save time, we tuned each algorithm by using $\frac{1}{11}$ of each dataset as a tuning set, and performed ten-fold cross validation on the remaining $\frac{10}{11}$. The tuning set was used to choose the parameters ν and μ on the first fold, and the chosen parameters were then used for the remaining nine folds. In order to avoid bias due to the choice of the tuning set, we repeated the above procedure five times using a different, randomly selected, tuning set each time. This procedure allows us to efficiently investigate the behavior of the feature selection algorithms RFSVM, RFE, and Relief on the datasets. Since the algorithms exhibit similar behavior on the datasets, we believe that our results support the conclusion that RFSVM is effective for learning nonlinear classifiers with reduced input space features.

For all the algorithms, we chose ν and the Gaussian kernel parameter μ from the set $\{2^i | i \in \{-7, \ldots, 7\}\}$. For each dataset, we evaluated the accuracy and number of features selected at $\sigma \in \{0, 1, 2, 4, 8, 16, 32, 64\}$. The diagonal of E was randomly initialized so that $\max\{\frac{n}{\sigma}, 1\}$ features were present in the first linear program, where n is the number of input space features for each dataset. As σ increases, the penalty on the number of features begins to dominate the objective. We only show values of σ for which we obtained reliable results. For RFE, we removed 1 feature per iteration. For Relief, we used 1000 iterations to determine the feature weights.

3.1.4. Results and discussion. Figure 1 gives curves showing the accuracy of RFSVM versus the number of input space features used on the Ionosphere and Sonar datasets. Each point on the curve is obtained by averaging five ten-fold cross validation experiments for a fixed σ. The square points denote the accuracy of NKSVM1, an ordinary nonlinear classifier which uses all the input space features. The points marked by triangles represent the accuracy and feature reduction of SVM1, a linear classifier which is known to reduce features [1]. Results for RFE are denoted by '+', and results for Relief are denoted by '♦' while results of our RFSVM algorithm are denoted by circles. Note that RFSVM is potentially able to obtain a higher accuracy than the linear classifier using approximately the same number of features on the Ionosphere and Sonar datasets. Note also that even for $\sigma = 0$, RFSVM was able to reduce features based only on decrease in the objective term $e'y$. RFSVM is comparable in both classification accuracy and feature selection to RFE and Relief.

To illustrate the efficiency of our approach, we report the CPU time taken on the Ionosphere dataset. On this dataset, the RFSVM algorithm required an average of 6 cycles through the integer variables on the diagonal of the matrix E, and the solution of 3 linear programs. The averages are taken over the classifiers learned for each fold once the parameters were selected. Using the MATLAB profiler, we found that the CPU time taken for one complete experiment on the Ionosphere dataset was 60.8 minutes. The experiment required 1960 runs of the RFSVM algorithm. Of this time, approximately 75% was used in evaluating the objective function, and 15% was used in solving linear programs. Our experience with the RFSVM algorithm is that the bottleneck is often the objective function evaluations rather than the linear programs, which suggests that significant speedups could be obtained by using more restrictive settings of the number of sweeps k and the tolerance tol for decreasing $f(E)$. These measurements were taken using

MATLAB 7.2 [12] under CentOS Linux 4.3 running on an Intel 3.0 GHz Pentium 4. The linear programs were solved using CPLEX 9.0 [9] and the Gaussian kernels were computed using a compiled function written in C.

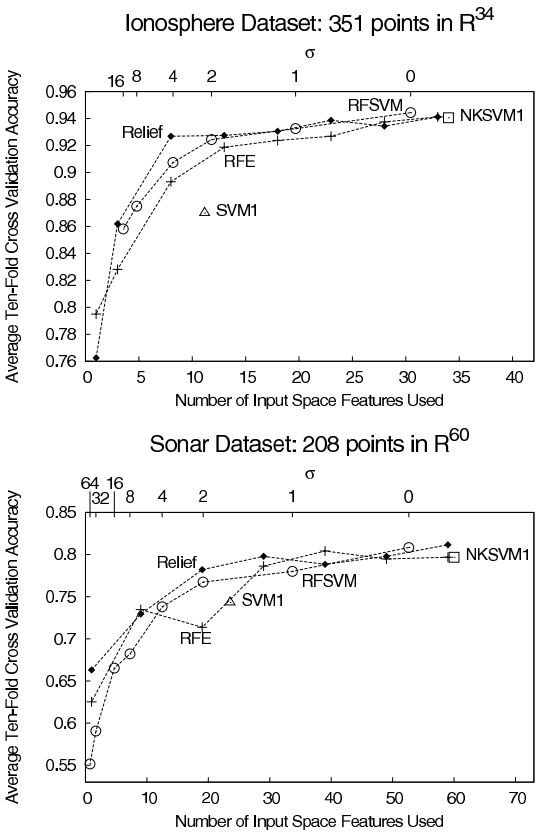

Figure 1. Ten-fold cross validation accuracy versus number of features used on the Ionosphere and Sonar datasets. Results for each algorithm are averages over five ten-fold cross validation experiments, each using a different $\frac{1}{11}$ of the data for tuning only, and the remaining $\frac{10}{11}$ for ten-fold cross validation. Circles mark the average number of features used and classification accuracy of RFSVM for each value of σ. '+', '♦', '□', and '△' represent the same values for RFE, Relief, NKSVM1, and SVM1, respectively.

3.2 NDCC Data

The NDCC dataset generator creates datasets by placing normal distributions at the vertices of concentric 1-norm cubes [17]. The resulting datasets are not linearly separable, thus making them attractive testbeds for nonlinear classifiers. We create datasets to test feature selection by

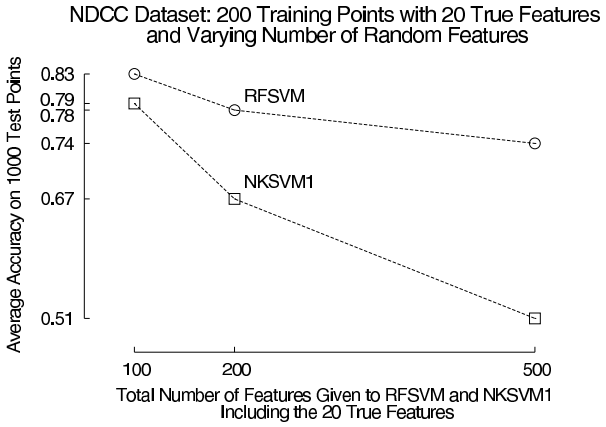

Figure 2. RFSVM1 and NKSVM1 on NDCC data with 20 true features and 80, 180, and 480 irrelevant random features. Each point is the average of the test set accuracy over two independently generated datasets.

adding random normal features to an NDCC dataset and then normalizing all features to have mean 0 and standard deviation 1. The order of the features is shuffled. Each dataset has 200 training points, 200 tuning points, and 1000 testing points. Accuracy of RFSVM and NKSVM1 on the dataset is measured by choosing ν and μ from the set $\{2^i | i \in \{-7, \ldots, 7\}\}$ using the tuning set, and then evaluating the chosen classifier on the 1000 testing points. To save time, we arbitrarily set σ in RFSVM to 1 before performing any experiments.

Figure 2 shows a comparison of RFSVM and NKSVM1 on NDCC data with 20 true features as the number of irrelevant features increases. Note that the accuracy of NKSVM1 decreases more than the accuracy of RFSVM as more irrelevant features are added. When 480 irrelevant features are added, the accuracy of RFSVM is 74%, 45% higher than NKSVM1.

We also investigated the performance of RFSVM on NDCC data with 1000 features, 900 of which were irrelevant. To improve the running time of RFSVM on problems with such large numbers of features, we implemented optimizations which took advantage of the form of the Gaussian kernel. We also used the Condor distributed computing system [10], which allowed us to evaluate Algorithm 2.1 for several tuning parameters simultaneously. Over 10 datasets, the average classification accuracy of RFSVM was 70%, while the average classification accuracy of NKSVM1 was 53%. Thus, the feature selection provided by RFSVM leads to a 32% improvement over a classifier with no feature selection. We expect that even better accuracy could be obtained by tuning σ, and heuristics to choose σ are an

important topic of future research.

When using Condor, we used the freely available CLP linear programming solver [4] to solve the linear programs and the MATLAB compiler version 4.5 to produce a standalone executable which ran Algorithm 2.1 for given values of ν and μ. On the same machine described above, the average time to run this executable for the parameters chosen by the tuning procedure was 115 seconds. Further speedups may be possible for some kernels, including the Gaussian kernel, by using approximations such as [16].

4. Conclusion and Outlook

We have presented a new approach to feature selection for nonlinear SVM classifiers for a completely arbitrary kernel. Our approach is formulated as an easily implementable mixed-integer program and solved efficiently by alternating between a linear program to compute the continuous parameter values of the classifier and successive sweeps through the objective function to update the integer variables representing the presence or absence of each feature. This procedure converges to a local minimum that minimizes both the usual SVM objective and the number of input space features used. Our results on two publicly available datasets and synthetic NDCC data show that our approach efficiently learns accurate nonlinear classifiers with reduced numbers of features. Extension of RFSVM to regression problems, further evaluation of RFSVM on datasets with very large numbers of features, use of different strategies to update the integer variables, and procedures for automatically choosing a value of σ for a desired percentage of feature reduction are important avenues of future research.

Acknowledgments

We thank Hector Corrada Bravo, Greg Quinn, and Nicholas LeRoy for their assistance with Condor. The research described in this Data Mining Institute Report 06-03, July 2006 and revised June 2007, was supported by National Science Foundation Grants CCR-0138308 and IIS-0511905.

References

[1] P. S. Bradley and O. L. Mangasarian. Feature selection via concave minimization and support vector machines. In J. Shavlik, editor, *Proceedings 15th International Conference on Machine Learning*, pages 82–90, San Francisco, California, 1998. Morgan Kaufmann. ftp://ftp.cs.wisc.edu/math-prog/tech-reports/98-03.ps.

[2] P. S. Bradley, O. L. Mangasarian, and D. R. Musicant. Optimization methods in massive datasets. In J. Abello, P. M. Pardalos, and M. G. C. Resende, editors, *Handbook of Massive Datasets*, pages 439–472, Dordrecht, Netherlands, 2002. Kluwer Academic Publishers. ftp://ftp.cs.wisc.edu/pub/dmi/tech-reports/99-01.ps.

[3] S. Dasgupta. Learning mixtures of Gaussians. In *IEEE Symposium on Foundations of Computere Science (FOCS) 1999*, pages 634–644, 1999.

[4] J. Forrest, D. de la Nuez, and R. Lougee-Heimer. *CLP User Guide*, 2004. http://www.coin-or.org/Clp/userguide/index.html.

[5] H. Fröhlich and A. Zell. Feature subset selection for support vector machines by incremental regularized risk minimization. In *International Joint Conference on Neural Networks (IJCNN)*, volume 3, pages 2041–2046, 2004.

[6] G. Fung and O. L. Mangasarian. A feature selection Newton method for support vector machine classification. *Computational Optimization and Applications*, 28(2):185–202, July 2004. ftp://ftp.cs.wisc.edu/pub/dmi/tech-reports/02-03.ps.

[7] C. Gold, A. Holub, and P. Sollich. Bayesian approach to feature selection and parameter tuning for support vector machine classifiers. *Neural Networks*, 18(5-6):693–701, 2005.

[8] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik. Gene selection for cancer classification using support vector machines. *Machine Learning*, 46(1-3):389–422, 2002.

[9] ILOG, Incline Village, Nevada. *ILOG CPLEX 9.0 User's Manual*, 2003. http://www.ilog.com/products/cplex/.

[10] M. Litzkow and M. Livny. Experience with the condor distributed batch system. In *Proceedings IEEE Workshop on Experimental Distributed Systems*, pages 97–101, Hunstville, AL, October 1990. IEEE Compter Society Press.

[11] O. L. Mangasarian. Generalized support vector machines. In A. Smola, P. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 135–146, Cambridge, MA, 2000. MIT Press. ftp://ftp.cs.wisc.edu/math-prog/tech-reports/98-14.ps.

[12] MATLAB. *User's Guide*. The MathWorks, Inc., Natick, MA 01760, 1994-2006. http://www.mathworks.com.

[13] P. M. Murphy and D. W. Aha. UCI machine learning repository, 1992. www.ics.uci.edu/~mlearn/MLRepository.html.

[14] M. Robnik-Šikonja and I. Kononenko. Theoretical and empirical analysis of ReliefF and RReliefF. *Machine Learning*, 53(1-2):23–69, 2003.

[15] B. Schölkopf and A. Smola. *Learning with Kernels*. MIT Press, Cambridge, MA, 2002.

[16] Y. Shen, A. Y. Ng, and M. Seeger. Fast gaussian process regression using kd-trees. In *NIPS 18*, 2006. http://ai.stanford.edu/~ang/papers/nips05-fastgaussianprocess.pdf.

[17] M. E. Thompson. NDCC: Normally distributed clustered datasets on cubes, 2006. www.cs.wisc.edu/dmi/svm/ndcc/.

[18] V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, New York, 1995.

[19] J. Weston, S. Mukherjee, O. Chapelle, M. Pontil, T. Poggio, and V. Vapnik. Feature selection for SVMs. In *NIPS*, pages 668–674, 2000.

[20] H. H. Zhang. Variable selection for support vector machines via smoothing spline ANOVA. *Statistica Sinica*, 16(2):659–674, 2006.