Relevance Units Machine for Classification

Mark Menor and Kyungim Baek
Department of Information and Computer Sciences
University of Hawai‘i at Mānoa
Honolulu, Hawai‘i 96822

***Abstract*—Classification, a task to assign each input instance to a discrete class label, is a prevailing problem in various areas of study. A great amount of research for developing models for classification has been conducted in machine learning research and recently, kernel-based approaches have drawn considerable attention mainly due to their superiority on generalization and computational efficiency in prediction. In this work, we present a new sparse classification model that integrates the basic theory of a sparse kernel learning model for regression, called relevance units machine, with the generalized linear model. A learning algorithm for the proposed model will be described, followed by experimental analysis comparing its predictive performance on benchmark datasets with that of the support vector machine and relevance vector machine, the two most popular methods for kernel-based classification.**

***Index Terms*—classification, sparse kernel model**

I. Introduction

Classification is one of the most studied problems in machine learning research and one that has kept a constant momentum in terms of research interest and theoretical novelty. Classification problems are ubiquitous in various fields of study including biomedical informatics. Examples of biomedical classification problems are to find the binding of a peptide to one of the pain receptors in the brain for drug development, protein structure prediction based on the primary sequence segments, MRI image analysis for brain tumor recognition, identifying protein-coding regions of DNA segments, and EEG signal analysis for brain computer interaction, just to list a few.

The main purpose of classification algorithms is to learn a classifier (or predictor) that assigns each input instance to one of a finite number of class labels. Over the last several decades, many classification algorithms have been proposed, such as decision trees [1]–[3], k-nearest neighbors [4], artificial neural networks [5], discriminant analysis [6], Bayesian networks [7], [8], and the support vector machine (SVM) [9].

Recently, considerable attention has focused on kernel-based learning approaches in supervised classification where they have shown state-of-the-art performance. A prediction model produced by kernel-based learning is often sparse, depending only on a subset of the kernel basis functions associated with some of the training samples. Of particular interest is the SVM, a sparse linearly-parameterized model, which has demonstrated great successes. The basic idea of the SVM is to find the optimal hyperplane decision boundary between the two data classes. By optimal, it means that the distance from the closest training examples to the separating hyperplane should be the largest possible, i.e. the margin between the two classes is maximized. Therefore, SVM minimizes the risk of overfitting and has potential for better classification accuracy from small training sets compared to other approaches to classification.

However, despite the current popularity and success, a number of significant and practical concerns of the SVM methodology have been identified: 1) The error/margin trade off term has to be selected often through a cross-validation procedure, which is a waste of both computation and data. 2) Although relatively sparse, the number of support vectors required tends to increase linearly to the size of the training set. 3) The prediction is not probabilistic, therefore it makes a hard decision [10].

The relevance vector machine (RVM), a Bayesian extension of the SVM, is a sparse learning algorithm that does not suffer from the limitations alluded to above [10], [11]. The RVM is capable of generating a fully probabilistic output and it has been shown empirically that, when solving the same problems, the RVM uses fewer relevance vectors than support vectors, making the classifier much sparser than the SVM-based one. This can lead to significant reduction in computational complexity of classification, while maintaining nearly identical performance to, if not better than, that of the SVM [10], [12], [13].

In this paper, we present a new classification model developed based on a sparse kernel regression model called relevance units machine (RUM), a recent variation of the RVM [14]. While RVM finds relevance vectors from the training samples like support vectors in the SVM, those vectors in the RUM are not necessarily from the training data. Those vectors, called relevance units, are in fact learnt during the training. The RUM also adopts the Bayesian inference framework to automatically learn all the parameters including the kernel parameters.

Based on the experimental results on regression problems, it has been argued that the RUM can generate an even sparser predictor than the RVM while retaining comparable performance [14]. However, the RUM learning framework has not been introduced for classification, therefore the development of RUM-based classifiers would provide insights for improving existing state-of-the-art classification methods. In this work, we present a RUM based classification model and evaluate its predictive performance through experimental studies with benchmark datasets.

In the next section, we specify a classification model that combines the RUM for regression with the generalized linear model. Assumptions on the parameters and the underlying distributions for prediction and the (hyper)priors are described in Section III, followed by the derivation of the learning algorithm for the proposed model in Section IV. Section V presents experimental studies and analyzes the results of benchmark comparisons with the SVM and RVM. Finally, we conclude our work with discussions on advantages and shortcomings of the current model, and future prospect of overcoming the limitations.

II. Model

We assume that the data is a set of N independent input vectors $X = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\} \subset \mathbb{R}^q$, along with their corresponding targets $\mathbf{t} = [t_1, t_2, \dots, t_N]^T$. Furthermore, we assume that the target values are binary, $t_i \in \{0, 1\}$. We use a model that combines the RUM for regression [14] with the generalized linear model, giving

$$\hat{t}(\mathbf{x}) = f\left(w_0 + \sum_{m=1}^{M} w_m k(\mathbf{x}, \mathbf{u_m}; \mathbf{\Theta})\right) \tag{1}$$

where $1 \le M \le N$, $f(\cdot)$ is an activation function, $k(\cdot, \cdot; \mathbf{\Theta})$ is a kernel function with a set of parameters $\mathbf{\Theta}$, the weights $\mathbf{w} = [w_0, w_2, \dots, w_M]^T$, and the units $\mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_M]$ with each $\mathbf{u}_i \in \mathbb{R}^q$. We call this the Classification Relevance Units Machine (CRUM) model.

The problem is to estimate from the data $(X, \mathbf{t})$ appropriate values of $\mathbf{w}$, $\mathbf{U}$, and $\mathbf{\Theta}$ to make good predictions $\hat{t}(\mathbf{x})$ for a fixed M. Typically $M \ll N$ to achieve a sparse solution. We use the logistic sigmoid function, $\sigma(a) = (1 + e^{-a})^{-1}$, as the activation function from here on.

III. Assumptions

Without loss of generality, we assume $w_0 = 0$ and set $\mathbf{w} = [w_1, w_2, \dots, w_M]^T$. Define

$$y_i = \sum_{m=1}^{M} w_m k(\mathbf{x}_i, \mathbf{u}_m; \mathbf{\Theta}) \tag{2}$$

and the design matrix $\mathbf{K}$ with elements

$$K_{ij} = k(\mathbf{x}_i, \mathbf{u}_j) \tag{3}$$

for $i \in \{1, 2, \dots, N\}$ and $j \in \{1, 2, \dots, M\}$.

We adopt a Bernoulli distribution for $P(t|\mathbf{x}, \mathbf{U}, \mathbf{w}, \mathbf{\Theta})$ as done with the RVM for classification [10], giving the following likelihood of the data $(X, \mathbf{t})$ given the model:

$$P(\mathbf{t}|X, \mathbf{U}, \mathbf{w}, \mathbf{\Theta}) = \prod_{n=1}^{N} \sigma(y_n)^{t_n} [1 - \sigma(y_n)]^{1-t_n} \tag{4}$$

A maximum likelihood estimate can be obtained by finding the parameters $\mathbf{w}$, $\mathbf{U}$, and $\mathbf{\Theta}$ that maximizes the likelihood function (4). However we specify further assumptions to add regularization terms and obtain a maximum a posteriori (MAP) estimate to mitigate overfitting.

We adopt a zero-mean isotropic Gaussian prior distribution over $\mathbf{w}$ to give preference to simple models,

$$p(\mathbf{w}|\alpha) = \mathcal{N}(\mathbf{w}|\mathbf{0}, \alpha^{-1}\mathbf{I}) \tag{5}$$

where $\mathbf{I}$ is the $M \times M$ identity matrix and $\alpha \in \mathbb{R}^+$ is the hyperparameter controlling the precision of the Gaussian over the weights. Note that this is not the Automatic Relevance Determination prior used in the RVM [10]. Driving w_i values to zero is not a goal since sparsity is instead achieved by selecting a small value for M.

Similarly we specify a Gaussian prior over $\mathbf{U}$ with

$$p(\mathbf{U}|r) = \left(\frac{r}{2\pi}\right)^{Mq/2} \exp\left\{-\frac{r}{2}\mathrm{tr}[\mathbf{U}^T\mathbf{U}]\right\} \tag{6}$$

where $r \in \mathbb{R}^+$ is the hyperparameter that controls the precision of the prior.

We assume Gamma distributed hyperpriors for the hyperparameters α and r

$$p(\alpha) = \mathrm{Gamma}(\alpha|a, b) \tag{7}$$
$$p(r|c, d) = \mathrm{Gamma}(r|c, d) \tag{8}$$

where $\mathrm{Gamma}(x|a, b) = \Gamma(a)^{-1}b^a x^{a-1}e^{-bx}$ and $\Gamma(a) = \int_0^\infty t^{a-1}e^{-t}dt$. The a, b, c, and d parameters are fixed to small values to make the hyperpriors non-informative [10].

We obtain the posterior distribution by combining all our assumptions (4)–(8),

$$p(\mathbf{t}, \mathbf{U}, \mathbf{w}, \alpha, r|X, \mathbf{\Theta}) = P(\mathbf{t}|X, \mathbf{U}, \mathbf{w}, \mathbf{\Theta})p(\mathbf{U}|r)p(r)p(\mathbf{w}|\alpha)p(\alpha) \tag{9}$$

Let $\hat{t}_n = \hat{t}(\mathbf{x}_n)$ and $\mathbf{B} = \mathrm{diag}(\hat{t}_1(1-\hat{t}_1), \hat{t}_2(1-\hat{t}_2), \dots, \hat{t}_N(1-\hat{t}_N))$. We can marginalize $\mathbf{w}$ from (9) using a Laplace approximation [15] of the integral, as is done in the RVM, to obtain:

$$\begin{aligned} &p(\mathbf{t}, \mathbf{U}, \alpha, r|X, \mathbf{\Theta}) = \\ &\int P(\mathbf{t}|X, \mathbf{U}, \mathbf{w}, \mathbf{\Theta})p(\mathbf{w}|\alpha)p(\mathbf{U}|r)p(r)p(\alpha)d\mathbf{w} \qquad (10) \\ &\approx (2\pi)^{M/2}|\mathbf{\Sigma}|^{1/2}P(\mathbf{t}|X, \mathbf{U}, \mathbf{w}^*, \mathbf{\Theta})p(\mathbf{w}^*|\alpha)p(\mathbf{U}|r)p(r)p(\alpha) \end{aligned}$$

where the covariance of the Laplace approximation is

$$\mathbf{\Sigma} = (\mathbf{K}^T\mathbf{B}\mathbf{K} + \alpha\mathbf{I})^{-1} \tag{11}$$

and its mean $\mathbf{w}^*$ is the $\mathbf{w}$ that maximizes $P(\mathbf{t}|X, \mathbf{U}, \mathbf{w}, \mathbf{\Theta})p(\mathbf{w}|\alpha)$.

The MAP estimate of the parameters $\mathbf{w}$, $\mathbf{U}$, and $\mathbf{\Theta}$ are those that maximize (10).

IV. Learning Algorithm

We define our objective function as

$$L = -\ln p(\mathbf{t}, \mathbf{U}, \alpha, r | X, \mathbf{\Theta}) \tag{12}$$

and we use methods from unconstrained optimization for continuous functions to find the parameters $\mathbf{w}$, $\mathbf{U}$, and $\mathbf{\Theta}$ that minimize L. Typically such methods require computing the gradient of L that we will provide here.

To begin, we need to compute the mean $\mathbf{w}^*$ for the Laplace approximation. This is the $\mathbf{w}$ that maximizes

$$\ln[P(\mathbf{t}|X, \mathbf{U}, \mathbf{w}, \mathbf{\Theta}) p(\mathbf{w}|\alpha)] = \tag{13}$$
$$\sum_{n=1}^{N} [t_n \ln \hat{t}_n + (1 - t_n) \ln(1 - \hat{t}_n)] - \frac{\alpha}{2} \mathbf{w}^T \mathbf{w} + \text{const}$$

where $\hat{t}_n$ is the prediction on $\mathbf{x}_n$ using the current model estimates. The first term is the log-likelihood and second is the regularization term. Let $\hat{\mathbf{t}} = [\hat{t}_1, \hat{t}_2, \ldots, \hat{t}_N]^T$. Then the vector $\mathbf{w}^*$ can be computed by maximizing (13) with respect to $\mathbf{w}$ using the Newton-Raphson method with gradient

$$\mathbf{K}^T(\mathbf{t} - \hat{\mathbf{t}}) - \alpha \mathbf{w} \tag{14}$$

and the Hessian

$$\mathbf{K}^T \mathbf{B} \mathbf{K} + \alpha \mathbf{I} \tag{15}$$

Now that we have $\mathbf{w}^*$ we can compute the partial derivatives of L as follows,

$$\frac{\partial L}{\partial \alpha} = -\frac{1}{2} \left(\frac{M}{\alpha} - \mathbf{w}^{*T} \mathbf{w}^* - \operatorname{tr}[\mathbf{\Sigma}] \right) \tag{16}$$

$$\frac{\partial L}{\partial U_{ij}} = r U_{ij} - \sum_{n=1}^{N} (t_n - \hat{t}_n) w_j^* \frac{\partial k(\mathbf{x}_n, \mathbf{u}_j; \mathbf{\Theta})}{\partial U_{ij}} \tag{17}$$

$$\frac{\partial L}{\partial r} = \frac{1}{2} \operatorname{tr}\left[\mathbf{U}^T \mathbf{U}\right] - \frac{Mq}{2r} \tag{18}$$

$$\frac{\partial L}{\partial \mathbf{\Theta}} = \sum_{n=1}^{N} (\hat{t}_n - t_n) \operatorname{tr}\left[\mathbf{w}^{*T} \frac{\partial \mathbf{k}(\mathbf{x}_n)}{\partial \mathbf{\Theta}} \right] \tag{19}$$

where $\mathbf{k}(\mathbf{x}) = [k(\mathbf{x}, \mathbf{u}_1; \mathbf{\Theta}), k(\mathbf{x}, \mathbf{u}_2; \mathbf{\Theta}), \ldots, k(\mathbf{x}, \mathbf{u}_M; \mathbf{\Theta})]^T$. Note that the terms in L associated with $p(\alpha)$ and $p(r)$ were dropped by assuming that they are non-informative. Also the term in L associated with $|\mathbf{\Sigma}|^{1/2}$ was dropped for simplicity in (17) and (19). Therefore these are actually the partials of the negative log of (9) with respect to U_{ij} and $\mathbf{\Theta}$, respectively. We observe good predictive performance despite these simplifications.

We chose to implement this minimization using an alternating variables method where only one of the parameters or hyperparameters ($\mathbf{w}$, α, $\mathbf{U}$, r, or $\mathbf{\Theta}$) is optimized at a time while holding all other parameters and hyperparameters fixed. This method cycles through each parameter and hyperparameter estimator repeatedly until the estimates converge. While this approach leads to slower convergence due to the ignorance of any correlation between the parameters and hyperparameters [16], this simplification allows us to work with smaller Hessian matrices. Also, we have observed lower predictive performance with estimates gained using a trust region Newton method [17] over all parameters and hyperparameters simultaneously under the same initial values. Therefore it appears, at least on the datasets we have tested, that the alternating variables method leads to better predictive solutions.

The MAP estimate of $\mathbf{w}$ is $\mathbf{w}^*$, obtained using the Newton-Raphson method. The estimates for α and r are obtained by setting their partial derivatives (16) and (18) to zero, giving

$$\alpha^* = \frac{M - \alpha \operatorname{tr}[\mathbf{\Sigma}]}{\mathbf{w}^T \mathbf{w}} \tag{20}$$

$$r^* = \frac{Mq}{\operatorname{tr}[\mathbf{U}^T \mathbf{U}]} \tag{21}$$

A trust region Newton method [17] is used to obtain the MAP estimates of $\mathbf{U}$ and $\mathbf{\Theta}$ using (17) and (19) to compute the partial derivatives of L with respect to $\mathbf{U}$ and $\mathbf{\Theta}$, respectively, and using finite differencing to approximate the Hessian in both cases.

In summary, the pseudocode for the overall algorithm is as follows:

```
1: $\mathbf{w}, \alpha, \mathbf{U}, r, \mathbf{\Theta} \leftarrow$ initial values
2: while $\mathbf{w}, \alpha, \mathbf{U}, r, \mathbf{\Theta}$ have not converged do
3:    Compute design matrix $\mathbf{K}$ using (3)
4:    $\mathbf{w} \leftarrow$ Newton-Raphson using (13)–(15)
5:    $\alpha \leftarrow$ Equation (20)
6:    $\mathbf{U} \leftarrow$ trust region Newton using (12) and (17)
7:    $r \leftarrow$ Equation (21)
8:    $\mathbf{\Theta} \leftarrow$ trust region Newton using (12) and (19)
9: end while
```

V. Experimental Results

For all experiments we used the Gaussian kernel

$$k(\mathbf{x}, \mathbf{u}; \mathbf{\Theta} = \{\sigma\}) = \exp\left\{ -\frac{1}{\sigma^2} (\mathbf{x} - \mathbf{u})^T (\mathbf{x} - \mathbf{u}) \right\} \tag{22}$$

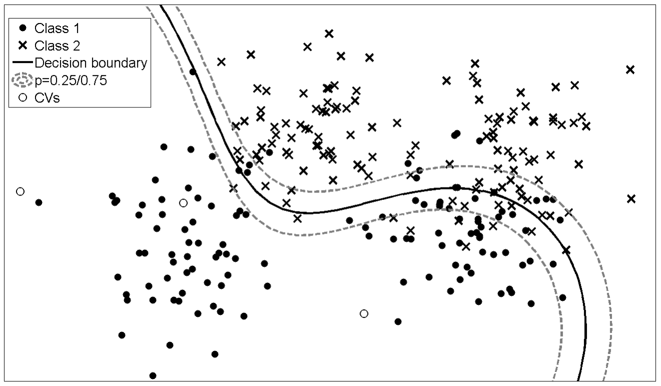
where σ is called the width of the kernel. We will use the term critical vector (CV) to refer to the basis vectors with non-zero w_i terms. The CVs are also called support vectors (SV) in the SVM model [9], [18], relevance vectors (RV) in the RVM model, and relevance units (RU) in the regression RUM and CRUM models.

A. Ripley's Synthetic Data

For visualization, we analyze two-dimensional data generated from mixtures of two Gaussians from [19] whose Bayes error is about 8%. For a comparison, both the RVM and CRUM models are trained on a set of 250 examples and

TABLE I
BENCHMARKING RESULTS OF THE SVM, RVM, AND CRUM. NOTE THAT THE PERFORMANCE REPORTED FOR THE CRUM IS FROM THE MODEL WITH THE BEST OBSERVED AIC, NOT WITH THE LOWEST ERROR RATE.

Dataset	N	N_{test}	d	SVM err%	RVM err%	CRUM err%	#SVs	#RVs	#RUs
Pima Diabetes	200	332	8	20.1	19.6	22.0	109	4	2
Banana	400	4900	2	10.9	10.8	13.4	135.2	11.4	7
Breast Cancer	200	77	9	26.9	29.9	28.6	116.7	6.3	2
Titanic	150	2051	3	22.1	23.0	23.4	93.7	65.3	1
Waveform	400	4600	21	10.3	10.9	10.9	146.4	14.6	1
German	700	300	20	22.6	22.2	22.0	411.2	12.5	5
Image	1300	1010	18	3.0	3.9	3.1	166.6	34.6	4



(a) CRUM with M = 3

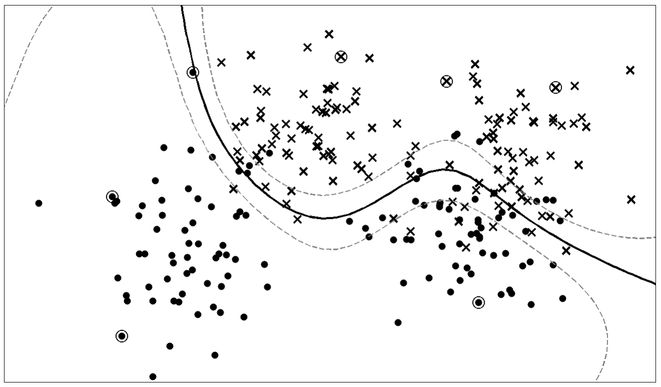

(b) RVM

Fig. 1. Plots of the CRUM and RVM predictions on Ripley's data. The solid curve represents the chosen decision boundary at $\hat{t}(\mathbf{x}) = 0.5$, while the dashed curves represent the boundary at values 0.25 and 0.75. The CRUM and RVM have their 3 and 7 CVs, respectively, indicated by the empty circles.

evaluated on a test set of 1000 examples. A CRUM was estimated with values of M from 1 through 8 and it was observed that $M = 3$ achieved the best Akaike Information Criterion (AIC) and Bayesian Information Criterion (BIC) [15] values. The decision boundary used here is $\hat{t}(\mathbf{x}) = 0.5$. This CRUM model achieves 9.6% error on the test set and is plotted in Figure 1a. The RVM model was estimated using a kernel width of 0.5, resulting in 7 CVs and a test error of 9.9%. The RVM was generated using the SparseBayes software from http://www.vectoranomaly.com. The plotted RVM result is shown in Figure 1b. The CRUM achieves slightly better predictive performance with a sparser model than the RVM.

B. Benchmark Comparisons

We ran a benchmark test to compare the classification results of the CRUM model to the previously published results of the RVM and SVM models on a variety of datasets. All datasets are taken from a collection compiled by Rätsch et al. [20] except the Pima Diabetes dataset which is taken from [19]. In this test, all problems are binary classification and both the RVM and CRUM use a decision boundary at the predicted value of 0.5 in all cases. The benchmark datasets used in the comparisons are listed below.

- **Pima Diabetes:** This is a collection of measurements taken from female patients at least 21 years old of Pima Indian heritage, and the problem is to predict the onset of diabetes mellitus.
- **Banana:** A set of two-dimensional samples generated by Rätsch et al. [20].
- **Breast Cancer:** This dataset contains measurements of nine attributes for prediction of the prognosis of breast cancer recurrence.
- **Titanic:** With this dataset, the task is to predict the survival of a passenger of the historical ship Titanic based on some attributes of the individuals.
- **Waveform:** This dataset contains a variety of noisy attributes and the task is to determine the class of the wave.
- **German:** This dataset contains credit data to use to classify a customer as good or bad credit risks.
- **Image:** This is a set of image segmentations described by numeric-valued attributes and the task is to determine what the image segments represent.

Table I lists the size of training and test datasets N and N_{test}, respectively, data dimension d, the observed error rates on the independent test dataset, and the number of CVs used in the SVM, RVM, and CRUM models. Unlike the other methods, CRUM requires that we specify M, the number of CVs. For each dataset we trained a CRUM for $M = 1$, $\lfloor C/2 \rfloor$ and $\lceil C \rceil$, where C is the the number of RVs reported. The AIC and BIC of these models were computed and further values of M were selected based on which interval could heuristically contain the optimal AIC or BIC. Note that what is reported in Table I is the CRUM model with the best observed AIC, not with the lowest error rate. With the exception of Pima Diabetes dataset, the reported error and number of CVs for the SVM and RVM are averages over 10 training/test splits. However we only report the error and the number of CVs for the CRUM on a single training/test split for computational reasons.

The results indicate that CRUM can achieve much sparser (less CVs) solutions than both the RVM and SVM with little sacrifice to predictive performance. In particular, there is a significant decrease in the number of CVs in the Titanic, Waveform, and Image datasets compared to the RVM and SVM; note that, for Titanic and Waveform, only one CV is enough to offer comparable results. A sparser solution means that the summation in (1) contains significantly less terms and is therefore more computationally efficient than the equivalent RVM or SVM. This feature is of great importance in running predictions on very large datasets, which is often the case in many biomedical informatics applications.

VI. Conclusion

The CRUM model has several advantages. It can be highly sparse and yet leads to very good generalization, as implied by the benchmark comparisons shown in Table I. Additionally, a user can choose to select smaller M to generate more compact models in sacrifice of the performance, if deemed desirable. This is not possible with the RVM, but can be done with the SVM through manipulation of the complexity control parameter. The model is also probabilistic and allows more flexibility than the SVM in choosing an appropriate decision function. The kernel parameters are estimated during the training unlike both the SVM and RVM where they must be specified. An appropriate M still needs to be selected, but this can be done using the AIC or BIC in a process simpler than cross-validation.

While the resulting model is extremely sparse and leads to a very efficient classifier, the major disadvantage of the CRUM is the computational complexity of the current learning algorithm. For memory complexity, the CRUM requires the storage for an $N \times M$ design matrix and a Hessian matrix of dimension $Mq \times Mq$ in the estimation of $\mathbf{U}$. This means that the space complexity of the CRUM is higher than that of the RVM if $Mq > N$. The time complexity of inverting the Hessian matrices used to estimate $\mathbf{w}$ and $\mathbf{U}$, and the approximation of the Hessian itself using finite differencing in the $\mathbf{U}$ estimation are computationally expensive. Potential avenues to reducing the complexity of the algorithm in the future include using more crude but cheaper approximations to the Hessian and the use of other unconstrained optimization algorithms such as L-BFGS [21] that do not explicitly form or store the complete Hessian.

Acknowledgment

This work was supported by U.S. Public Health Services grants P20RR016467 and P20RR018727 from the National Center for Research Resources, National Institutes of Health. The content is solely the responsibility of the authors and do not necessarily represent the official views of the National Center for Research Resources, National Institutes of Health.

References

[1] S. K. Murphy, "Automatic construction of decision trees from data: A multi-disciplinary survey," *Data Mining and Knowledge Discovery*, vol. 2, pp. 345–389, 1997.

[2] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and Regression Trees*. Monterey, CA: Wadsworth and Brooks, 1984.

[3] J. R. Quinlan, *C4.5: Programs for Machine Learning*. San Francisco: Morgan Kaufmann, 1993.

[4] T. M. Cover and P. E. Hart, "Nearest neighbor pattern classification," *IEEE Transactions on Information Theory*, vol. 13, no. 1, pp. 21–27, 1967.

[5] G. P. Zhang, "Neural networks for classification: A survey," *IEEE Transactions on Systems, Man and Cybernetics, Part C (Applications and Reviews)*, vol. 30, no. 4, pp. 451–462, 2000.

[6] J. H. Friedman, "Regularized discriminant analysis," *Journal of the American Statistical Association*, vol. 84, no. 405, pp. 165–175, 1989.

[7] F. V. Jensen, *An Introduction to Bayesian Networks*. Secaucus, NJ: Springer-Verlag New York, Inc., 1996.

[8] M. G. Madden, "The performance of bayesian network classifiers constructed using different techniques," in *Proceedings of the ECML/PKDD-03, Workshop on Probabilistic Graphical Models for Classification*, 2003, pp. 59–70.

[9] V. N. Vapnik, *Statistical Learning Theory*. New York: Wiley, 1998.

[10] M. E. Tipping, "Sparse bayesian learning and the relevance vector machine," *Journal of Machine Learning Research*, vol. 1, pp. 211–244, 2001.

[11] M. E. Tipping and A. C. Faul, "Fast marginal likelihood maximisation for sparse bayesian models," in *Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics*, C. M. Bishop and B. J. Frey, Eds., 2003.

[12] Z. Chen and H. Tang, "Sparse bayesian approach to classification," in *IEEE Proceedings of Networking, Sensing and Control*, 2005, pp. 914–917.

[13] X.-M. Xu, Y.-F. Mao, J.-N. Xiong, and F.-L. Zhou, "Classification performance comparison between rvm and svm," in *2007 IEEE International Workshop on Anti-counterfeiting, Security, Identification*, 2007, pp. 208–211.

[14] J. Gao and J. Zhang, "Sparse kernel learning and the relevance units machine," in *PAKDD'09*, 2009.

[15] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[16] R. Fletcher, *Practical Methods of Optimization*, 2nd ed. Wiley, 2000.

[17] T. F. Coleman and Y. Li, "An interior, trust region approach for nonlinear minimization subject to bounds," *SIAM Journal on Optimization*, vol. 6, pp. 418–445, 1996.

[18] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proceedings of the fifth annual workshop on Computational learning theory*, 1992.

[19] B. D. Ripley, *Pattern Recognition and Neural Networks*. Cambridge University Press, 1996.

[20] G. Rätsch, T. Onoda, and K.-R. Müller, "Soft margins for adaboost," *Machine Learning*, vol. 42, no. 3, pp. 287–320, 2001.

[21] D. C. Liu and J. Nocedal, "On the limited memory method for large scale optimization," *Mathematical Programming B*, vol. 45, no. 3, pp. 503–528, 1989.