On The Value of Leave-One-Out Cross-Validation Bounds

Jason D. M. Rennie
jrennie@csail.mit.edu

December 15, 2003

Abstract

A long-standing problem in classification is the determination of the regularization parameter. Nearly every classification algorithm uses a parameter (or set of parameters) to control classifier complexity. Cross-validation on the training set is usually done to determine the regularization parameter(s). [1] proved a leave-one-out cross-validation (LOOCV) bound for a class of kernel classifiers. [2] extended the bound to Regularized Least Squares Classification (RLSC). We provide the (trivial) extension to multiclass. Our contribution is empirical work. We evaluate the bound's usefulness as a selector for the regularization parameter for RLSC. We find that it works extremely poorly on the data set we experimented with (20 Newsgroups); the LOOCV bound consistently selects a regularization parameter that is too large.

1 Introduction

A common problem in training a classifier (and most any learner, for that matter) is the selection of the regularization parameter. All modern classifiers use a parameter to control the complexity of resulting model. The Support Vector Machine (SVM) [3] uses C to trade off between minimizing loss on the training data and minimizing the magnitude of the weight vector. Other kernel classifiers (such as Regularized Logistic Regression (RLR) and Regularized Least Squares (RLSC)) use a similar trade-off parameter. Naive Bayes [4] smooths probability estimates by adding α to training data counts. Decision tree algorithms [5] limit tree depth or prune back nodes. Setting the regularization parameter by minimizing loss/error or maximizing likelihood on the training data results in the most complex model. So, the regularization parameter is usually learned on a holdout set or via cross-validation.

An excellent method for learning the regularization parameter is leave-one-out cross-validation (LOOCV). Each training example is labeled by a classifier trained on all other training examples. While this is efficient to calculate for simple algorithms such as Naive Bayes, it is much less efficient for algorithms

such as the SVM, requiring the training of one classifier for each support vector. [6] showed that the LOOCV error for RLSC can be computed efficiently given knowledge of the diagonal entries of the inverse of an n-by-n matrix (n is the number of training examples). But, this is still impractical for large data sets. [1] proved LOOCV error bounds for a class of kernel classifiers including SVMs. [2] showed that this class of bounds are also applicable to RLSC.

[2] discusses the use of these bounds for estimation of the generalization error. But, our own experimental evidence indicates that these bounds are so poor as to be almost useless. We ran binary and multiclass text classification experiments on the 20 Newsgroups data set[1]. We found that the bounds greatly over-predicted error for smaller regularization values and consistenly selected too large a regularization value. In fact, you would do much better by simply selecting the largest of the regularization parameters that gave the lowest training error (usually many settings gave the same training error). We report these results.

2 Leave-One-Out Cross-Validation Bounds

Regularized Least Squares (RLSC) is a classification algorithm much like the Support Vector Machine and Regularized Logistic Regression. It minimizes a loss function plus a complexity penalty. A regularization parameter, λ, is used to regulate the complexity of the classifier (the magnitude of the weight vector). Given a matrix of training examples (one per row), X, and a vector of binary labels, $\mathbf{y}$, RLSC minimizes[2]

$$(Xw - \mathbf{y})^T(Xw - \mathbf{y}) + \lambda w^T w. \tag{1}$$

RLSC can be kernelized, using a weight vector on the examples, $\mathbf{c}$. The kernel form can be appealing even when using the linear kernel since it can reduce the number of parameters to be optimized (if the number of examples is less than the number of features). For a vector of weights on examples, $\mathbf{c}$, RLSC minimizes

$$(Kc - \mathbf{y})^T(Kc - \mathbf{y}) + \lambda c^T K c, \tag{2}$$

where K is the kernel matrix. The linear kernel is XX^T; setting $K = XX^T$ makes kernel RLSC equivalent to minimizing Equation 1.

[2] shows that one can bound the LOOCV error of RLSC by

$$|\mathbf{x}_i : y_i \left(\sum_{j \neq i} c_j K_{ij} \right) \leq 0|. \tag{3}$$

In other words, an example is classified correctly if, when its contribution to the output is discarded, the example is still classified correctly.

[1] `We use 20news-bydate.tar.gz` from `http://www.ai.mit.edu/~jrennie/20Newsgroups`.

[2] We discard the $\frac{1}{l}$ constant that [2] uses.

We perform multiclass RLSC using one-versus-all (OVA). For each class, we build a binary classifier that uses in-class examples as positive and out-of-class examples as negative. A new example gets the label of the binary classifier that gives the largest (most positive) output. [2] finds that this is an effective way to use RLSC for multiclass classification. It is trivial to extend the LOOCV error bounds given by [2] to the OVA multiclass case. Let c_{kj} be the j^{th} example weight for the k^{th} binary classifier. Multiclass LOOCV error is bounded by

$$|\mathbf{x}_i : y_i \neq \arg\max_k \sum_{j \neq i} c_{kj} K_{ij}|. \tag{4}$$

$\sum_{j \neq i} c_{kj} K_{ij}$ is the example i classification output for the k^{th} label, without the contribution from example i. This is a direct extension of the bound given by [2].

3 Experiments

To evaluate the goodness of these LOOCV error bounds, we ran text classification experiments on the 20 Newsgroups data set. 20 Newsgroups (as the title suggests) is a collection of newsgroup posts from 20 different newsgroups from the mid 1990s. There are approximately 1000 posts per newsgroup. We use the "20news-bydate" version, which has dupliates removed, posts are sorted within each newsgroup by date into train/test sets and newsgroup-identifying headers are discarded. Our pre-processing consisted of (in order) (1) splitting on space characters, (2) lower-casing all alphabetic characters, (3) discarding tokens of length 25 or greater, (4) converting all digits to 'D', and (5) removing non-alphanumeric beginning and ending characters. We computed a document vector for each post, consisting of the number of times each word occured in that post. We transformed each count by $\log(x+1)$ and then normalized each vector to (L2-norm) length 1. We did not stem or use a stop list. A pre-processed (ready for Matlab) version of the data set, along with code we used for RLSC is available at `http://www.ai.mit.edu/~jrennie/matlab/`.

We ran two sets of experiments, one 20 class experiment using the entire data set and 100 binary experiments using random pairs of classes. In each case, we tested a wide range of lambda values, and, for each, calculated error on the training set, error on the test set and the LOOCV error bound. We found the results to be surprising. We didn't expected the LOOCV error bound to closely approximate the test error, but we did expected it to provide a good choice of lambda. This was never the case. In fact, the LOOCV bound consistently chose too high of a lambda. Test errors for the lambda with minimum LOOCV bound were much higher (in one case, almost an order of magnitude higher) than the lowest test error. We could have almost always done better (!) by simply chosing the largest lambda from the set of lambdas that gave the lowest error (often, many different lambdas yielded zero training error).

Table 1 gives the results for the full 20 class classification task. The lowest test error (15.1%) was achieved with a setting of $\lambda = 10^{-3}$. The LOOCV error

λ	Train	Test	LOOCV
10^1	0.4019	0.5027	0.4618
1	0.2159	0.3449	0.3109
10^{-1}	0.0704	0.2111	**0.2456**
10^{-2}	0.0121	0.1625	0.4187
10^{-3}	0.0004	**0.1514**	0.6959
10^{-4}	0.0002	0.1658	0.7600
10^{-5}	**0.0001**	0.1778	0.7731
10^{-6}	0.0003	0.1977	0.7879

Table 1: Shown is classification error for different values of lambda on the full 20 newsgroups classification task. The first column gives the regularization parameter used. The second column gives training error. The third column gives test error. The fourth column gives the LOOCV error bound. The LOOCV error bound greatly over-estimates generalization error for small values of lambda.

λ	Train	Test	LOOCV
10^1	0.2838	0.3137	0.2993
1	0.1072	0.1453	0.1325
10^{-1}	0.0173	0.0579	**0.0533**
10^{-2}	0.0006	0.0341	0.0736
10^{-3}	**0.0000**	**0.0329**	0.1885
10^{-4}	0.0000	0.0365	0.2755
10^{-5}	0.0000	0.0391	0.2962
10^{-6}	0.0000	0.0402	0.2984

Table 2: Shown is classification error averaged over 100 random binary classification tasks for the 20 Newsgroup data set. Each round, we randomly selected two (different) classes and trained a binary classifier. Each entry in the table is averaged over 100 rounds. The first column gives the regularization parameter used. The second column gives training error. The third column gives test error. The fourth column gives the LOOCV error bound. The LOOCV error bound greatly over-estimates generalization error for small values of lambda.

bound gave the lowest error for $\lambda = 10^{-1}$, a parameter setting which gave a much higher test error (24.6%). Even the parameter setting that gave the lowest training error, $\lambda = 10^{-5}$, yielded a better test error (17.8%).

Table 2 gives the averaged results for 100 binary classification tasks. Binary classifiers were trained and tested on random pairs of classes from the 20 Newsgroups data set. We made no effort to avoid repeating pairings. The LOOCV error bound consistently over-estimated generalization error for small values of lambda. In some cases, the setting selected by the LOOCV error bound yielded a test error close to the smallest test error. Twice, it even chose the setting that gave the lowest test error. One such case was talk.politics.guns vs. talk.religion.misc. The LOOCV error bound was lowest for $\lambda = 10^{-2}$. This and a setting of $\lambda = 10^{-3}$ both gave the lowest test error, 8.1%. However, it was much more common for the LOOCV error bound to select a setting with a much higher test error than the best setting. For example, in the rec.sport.baseball vs. talk.religion.misc task, the LOOCV error bound was lowest for $\lambda = 10^{-1}$, a test error of 5.7%. The lowest test error was 1.2%, given by $\lambda = 10^{-3}$. In nearly every case, we could have done better using the training error to select the value of lambda[3]. For every pairing, some value of lambda achieved zero training error. The largest lambda with zero training error gave test error as low or lower than the lambda selected by the LOOCV error bound in all but one case. This is borne out in the table. A setting of $\lambda = 10^{-3}$ gave an average test error of 3.5%. A setting of $\lambda = 10^{-1}$ gave the lowest average LOOCV error bound, but much higher average test error, 6.6%.

4 Summary

Since every classification algorithm uses some sort of regularization parameter, it is important to be able to learn that parameter in a reasonable way. [1] proved LOOCV error bounds for a certain class of kernel classifiers. [2] extended those bounds to RLSC. We ran experiments on the 20 Newsgroups data set with RLSC using a wide range of regularization values. We found that the LOOCV error bound consistently overestimated error for small values of the regularization parameter and did a very poor job of selecting the right regularization parameter. We limited our experiments to a single text classification data set, so it may be that these results are uncharacteristic of other domains and possibly other text data sets.

References

[1] Tommi Jaakkola and David Haussler. Probabilistic kernel regression models. In *Proceedings of the Seventh International Workshop on Artificial Intelligence and Statistics*, 1998.

[3] We don't recommended this as a general approach for selecting the regularization parameter. We found it curious that training error worked as well as it did!

[2] Ryan Rifkin. *Everything Old Is New Again: A Fresh Look at Historical Approaches in Machine Learning*. PhD thesis, Massachusetts Institute of Technology, 2002.

[3] Christopher J. C. Burges. A tutorial on Support Vector Machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

[4] Jason D. M. Rennie, Lawrence Shih, Jaime Teevan, and David R. Karger. Tackling the poor assumptions of naive bayes text classifiers. In *Proceedings of the Twentieth International Conference on Machine Learning*, 2003.

[5] Tom Mitchell. *Machine Learning*. McGraw-Hill Companies, Inc., 1997.

[6] P. J. Green and B. W. Silverman. *Nonparametric Regression and Generalized Linear Models: A Roughness Penalty Approach*. Chapman and Hall, 1994.