Learning to Predict the Leave–one–out Error

Koji Tsuda[*†], Gunnar Rätsch[*], Sebastian Mika[*], Klaus–Robert Müller[*+]
[*]GMD FIRST, Kekuléstr. 7, 12489 Berlin, Germany
[+]University of Potsdam, Am Neuen Palais 10, 14469 Potsdam
[†]Electrotechnical Laboratory, 1–1–4, Umezono, Tsukuba, 305–0031, Japan
{tsuda, raetsch, mika, klaus}@first.gmd.de

1 Introduction

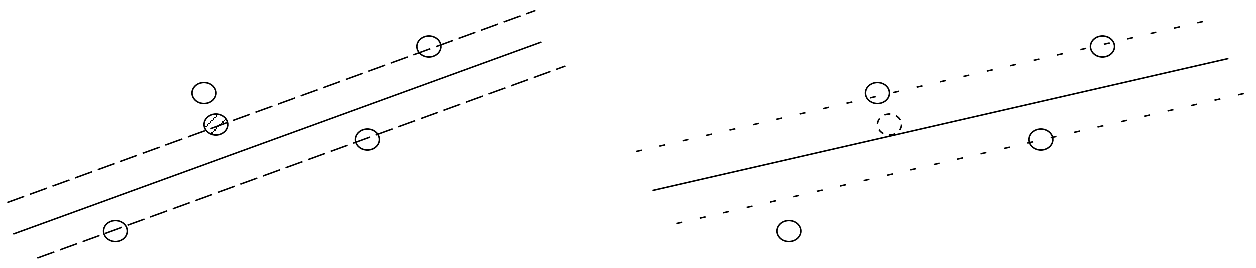
Model selection methods try to find a reasonably good estimate of the generalization error to select the proper hyper–parameters of a learning machine. The leave–one–out (LOO) error has the advantage that it is an (almost) unbiased estimator of the true generalization error. Its computation is, unfortunately, in most practical cases prohibitively slow. Starting from the LOO–estimation approach taken in [3] for Support Vector Machines (SVMs), we propose to use geometrical features from the learning problem to cast the leave–one–out error approximation problem into a – fast solvable – *classification* problem.

2 Predicting the Leave–one–out Error

Let $\mathbf{Z}$ be a sample of size ℓ, where each pattern is defined as $\boldsymbol{z}_i = (\boldsymbol{x}_i, y_i)$ $(\boldsymbol{x}_i \in \mathbb{R}^n, y_i \in \{\pm 1\})$. Furthermore, define $\mathbf{Z}_p = \{\boldsymbol{z}_i \in \mathbf{Z}, i \neq p\}$ and $f^p = \mathcal{L}(\mathbf{Z}_p)$, i.e. f^p is the decision obtained by learning using the learning algorithm $\mathcal{L}$ with the p-th sample left out of the training set. The LOO error is defined as $\epsilon^{loo}(\mathbf{Z}) = \frac{1}{\ell}\sum_{p=1}^{\ell} \Psi(-y_p f^p(\boldsymbol{x}_p))$, where $\Psi(q) = 1$ for $q > 0$ and $\Psi(q) = 0$ otherwise. As $-y_p f^p(\boldsymbol{x}_p)$ is positive only if f^p commits an error on $\boldsymbol{x}_p$, $\epsilon^{loo}(\mathbf{Z})$ is the average number of patterns that are misclassified when they are left out. In this paper we consider kernel classifiers: $f(\boldsymbol{x}) = \sum_{i=1}^{m} \alpha_i k(\boldsymbol{x}, \boldsymbol{x}_i)$, where $\boldsymbol{\alpha}$ is the weight vector and $k(\cdot,\cdot)$ is the kernel function computing a scalar product in feature space [1]. In sparse kernel classifiers, as e.g. Support Vector Machines [1] and Linear Programming Machines (LPMs) [2], often *only a few* patterns (called support vectors (SVs)), are concerned with LOO error, because removing a non–SV does not change the boundary.

To predict the LOO error, it is sufficient to predict the result of the LOO procedure for each support vector. Let $r_p \in \{0, 1\}$ describe the result of the LOO procedure for a given training sample $\boldsymbol{x}_p$, i.e. $r_p = \Psi(-y_p f^p(\boldsymbol{x}_p))$. The idea is to replace $y_p f^p(\boldsymbol{x}_p)$ by an estimate g_p, i.e. to model the LOO procedure by a function $\tilde{r}_p = \Psi(-g_p)$. In [3] an estimation method for SVMs based on the *span* of a SV – which depends on the distance of the SV to the other SVs – was proposed. The estimate has been proven to be exact if the set of support vectors does not change when the pattern is left out from the training set. However, in some cases this assumption may not hold and other information, e.g. about the non–SVs, can help to improve the performance (cf. Fig. 1).

Unfortunately this result cannot be easily transfered to other kernel classifiers (as e.g. LPMs). Thus, other ways to approximate the LOO–error have to be found. The main idea of this contribution is to interpret the LOO error estimation as a learning task, i.e. to construct a classifier which, based on suitable features, predicts whether a pattern will be misclassified if left out from the training set: without actually performing the LOO procedure. Here, the local geometry around a SV, including non–SVs, is of primal importance. For example, if there is a non–SV nearby, this vector will replace the removed support vector, and if the distance between the two samples is sufficiently small, the boundary will not change significantly (Fig. 1). Details about which features are found to be useful will be presented in the workshop.



(a) Decision boundary in full problem

(b) Decision boundary in LOO problem

Figure 1: Consider the case where a non–SV is near the SV of interest as shown in (a). In the leave–one–out procedure (b) the boundary is re–estimated without this SV, but the non–SV takes its role and the boundary does not change significantly. The LOO boundary could show a large difference, if the closeby non–SV would be absent. This example shows the importance of taking the local geometry around the SVs into account.

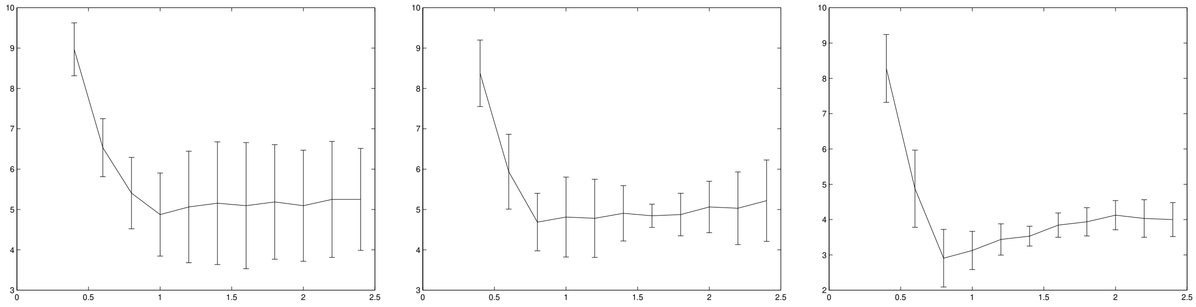


Figure 2: (Left) Estimated generalization error of LPM against kernel width. (Middle) Leave–one–out error actually calculated. (Right) The estimate of LOO error predicted by our method. All lines show averages over 10 different training/test sample splits and the error bars show the standard deviations over the trails.

3 Experiments

As one application of our approach, we consider multi–class pattern recognition problems. To solve c–class problems, it is common to use c classifiers, each of which is trained to discriminate one class against all other classes. Here, the hyper–parameters of each classifier have to be set properly by some model selection method. However, it usually takes prohibitively long to perform the leave–one–out procedure in all classifiers for all possible hyper–parameter settings. To cope with this problem, the leave–one–out procedure is performed with respect to only a few classes, and our LOO classifier learns from on these training cases. Then, the hyper–parameters of new classifiers for additional classes can efficiently be set based on the predictions of our leave–one–out classifier. Fig. 2 shows an example of a RBF kernel width selection experiment in a 3D object recognition data set using LPMs. We computed the generalization error prediction for different kernel parameters (width) using the LOO classifier and compared them with the "true" generalization error measured empirically on the respective test set. Our method shows excellent model selection performance as it reliably captures the minimum of the generalization error. Further application results will be presented at the workshop.

References

[1] B.E. Boser, I.M. Guyon, and V.N. Vapnik. A training algorithm for optimal margin classifiers. In D. Haussler, editor, *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*, pages 144–152, Pittsburgh, PA, 1992.

[2] E.J. Bredensteiner and K.P. Bennet. Feature minimization within decision trees. *Computational Optimization and Applications*, 10:110–126, 1997.

[3] V.N. Vapnik and O. Chapelle. Bounds on error expectation for support vector machines. *Neural Computation*, 2000. In press.