ESANN 2012 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 25-27 April 2012, i6doc.com publ., ISBN 978-2-87419-049-0.
Available from http://www.i6doc.com/en/livre/?GCOI=28001100967420.

The 'K' in K-fold Cross Validation

Davide Anguita, Luca Ghelardoni, Alessandro Ghio,
Luca Oneto and Sandro Ridella

University of Genova - Department of Biophysical and Electronic Engineering
Via Opera Pia 11A, I-16145 Genova - Italy

Abstract. The K-fold Cross Validation (KCV) technique is one of the most used approaches by practitioners for model selection and error estimation of classifiers. The KCV consists in splitting a dataset into k subsets; then, iteratively, some of them are used to learn the model, while the others are exploited to assess its performance. However, in spite of the KCV success, only practical rule-of-thumb methods exist to choose the number and the cardinality of the subsets. We propose here an approach, which allows to tune the number of the subsets of the KCV in a data–dependent way, so to obtain a reliable, tight and rigorous estimation of the probability of misclassification of the chosen model.

1 Introduction

The Support Vector Machine (SVM) [1] represents one of the state-of-the-art techniques in the framework of classification problems. The *training* (*TR*) phase consists in finding a set of parameters by solving a Convex Constrained Quadratic Programming (CCQP) problem, for which several effective techniques have been proposed throughout the years. However, the search of optimal parameters does not complete the learning phase of the SVM. In fact, a set of additional variables, namely the *hyperparameters*, needs to be tuned in order to select the best model, whose performance must be then assessed to guarantee the reliability of the results. The former step is known as the *Model Selection* (*MS*) phase, while the latter one is known as the *Error Estimation* (*EE*) phase.

Several techniques can be exploited for MS and EE in Support Vector Machines [2, 3]. A practical approach consists in using the K-fold Cross–Validation technique (KCV) [3, 4, 5], which allows both to tune the hyperparameters and to estimate the generalization error of the classifier. The available dataset is split into k folds, where, in turn, $(k-2)$ subsets are used for the TR phase and the remaining two are exploited for MS and EE purposes, respectively. Usually, the value of k is aprioristically chosen and fixed: however, the choice of k can severely influence both the performance of the selected model and the quality of the predicted error [6, 7]. Rule-of-thumb methods suggest to fix large values of k (5, 10 or 20), since it is usually preferable to exploit a larger number of patterns for training purposes, even at the expense of obtaining loose generalization error estimations [6].

However, in many fields, obtaining a tight and rigorous estimation of the generalization error of a classifier is not only an issue of academic interest, but also a way to guarantee, in a statistical sense, the reliability of the model. In these cases, smaller values of k should be preferred in order to reserve a larger

ESANN 2012 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 25-27 April 2012, i6doc.com publ., ISBN 978-2-87419-049-0.
Available from http://www.i6doc.com/en/livre/?GCOI=28001100967420.

fraction of samples for EE purposes. In many applications like, for example, legal practice [8], it is necessary to automatically find a trade–off between the percentage of data used to train the classifier, and the tightness of the estimated error. In this paper, we propose a modified, but statistically rigorous, KCV approach, which allows to implement this trade-off by considering k as an additional hyperparameter and tuning it during the MS phase, in a data–dependent way.

2 The Support Vector Machine: TR, MS and EE phases

Let us consider a binary classification problem, where $\mathcal{D}_n = \{(\boldsymbol{x}_i, y_i)\}$, $i = 1, \ldots, n$ is a dataset composed by n i.i.d. pairs, such that $\boldsymbol{x}_i \in \mathcal{X} \in \mathbb{R}^d$ and $y_i \in \mathcal{Y} \in \{\pm 1\}$. The relation between $\mathcal{X}$ and $\mathcal{Y}$ is encapsulated in an unknown distribution $\mathcal{P}(\mathcal{X}, \mathcal{Y})$, which originated the data. The goal of learning is to find a function $f : \mathbb{R}^d \to \mathcal{Y}_f \subseteq \mathbb{R}$, which approximates this relation. The SVM algorithm [1] can be exploited for this purpose, where the classifier is identified during the TR phase by solving the following CCQP problem [1]:

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j K(\boldsymbol{x}_i, \boldsymbol{x}_j) - \sum_{i=1}^{n} \alpha_i \\ & 0 \leq \alpha_i \leq C, \quad \sum_{i=1}^{n} y_i \alpha_i = 0, \end{aligned} \tag{1}$$

where C is one of the hyperparameters and $K(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is a kernel function. In this work we will focus on Gaussian kernels $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left(-\gamma \left\|\boldsymbol{x}_i - \boldsymbol{x}_j\right\|^2\right)$, where γ is an additional hyperparameter, but our results can be easily generalized to other kernel functions. After solving the CCQP problem, the output of the classifier can be computed as $f(\boldsymbol{x}) = \sum_{i=1}^{n} y_i \alpha_i K(\boldsymbol{x}_i, \boldsymbol{x}) + b$, where b is the bias.

The hyperparameter pair (C, γ) must be tuned during the MS phase. Though some rule-of-thumb methods have been suggested for deriving the hyperparameters in a very simple and efficient way (e.g. [9]), the most used and effective procedure is a grid search, where the CCQP problem is solved for several hyperparameter values and the models are compared by assessing their performance on previously unseen data. For this purpose, a usual approach consists in reserving a validation set $\mathcal{V}_m$, consisting in m patterns originated from the same $\mathcal{P}(\mathcal{X}, \mathcal{Y})$ but independent of the ones of $\mathcal{D}_n$. For every value of (C, γ), problem (1) is solved and the empirical error on $\mathcal{V}_m$ is computed:

$$\hat{L}_m(f) \triangleq \frac{1}{m} \sum_{i=1}^{m} \ell_H(f(\mathbf{x}_i), y_i), \quad \ell_H(f(\boldsymbol{x}), y) \triangleq \frac{1 - \operatorname{sign}[f(\boldsymbol{x})]\, y}{2}, \tag{2}$$

where ℓ_H is the loss function, which counts the number of misclassifications.

Once the MS phase concluded, the EE step consists in estimating the generalization error $L(f)$. Obviously, using $\hat{L}_m(f)$ as an estimation of $L(f)$ is not

ESANN 2012 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 25-27 April 2012, i6doc.com publ., ISBN 978-2-87419-049-0.
Available from http://www.i6doc.com/en/livre/?GCOI=28001100967420.

statistically correct and can lead to remarkable underestimations of the generalization error. Therefore, a further error estimation set $\mathcal{E}_s$, composed of s patterns and independent of both $\mathcal{D}_n$ (already used for training the model) and $\mathcal{V}_m$ (already exploited for selecting the hyperparameters), should be used for this purpose. As ℓ_H is a random variable which can assume only two different values and is distributed according to a Binomial distribution, we can exploit the Clopper-Pearson (CP) inequality [10] to upper–bound $L(f)$, which is statistically sound and provides the tightest error estimation [11]:

$$L(f) \leq L_{EE} \triangleq \max_{p} \left\{ p : \sum_{j=0}^{s\hat{L}_s(f)} \binom{s}{j} p^s (1-p)^{s-j} \geq \delta \right\}, \tag{3}$$

where $\hat{L}_s(f)$ is the empirical error rate of f on $\mathcal{E}_s$.

3 Complete K-fold Cross Validation

As three independent sets for TR, MS and EE could not be available in practical cases, the K-fold Cross Validation (KCV) procedure is often exploited [3, 4, 12, 5], which consists in splitting $\mathcal{D}_n$ in k subsets, where k is fixed in advance: $(k-2)$ folds are used, in turn, for the TR phase, one for the MS phase and one for the EE step. In particular, we define $\mathcal{D}_{n_{TR}}$ the TR set of $n_{TR} = n - 2(n/k)$ data, $\mathcal{D}_{n_{MS}}$ the MS set of $n_{MS} = n/k$ samples, and $\mathcal{D}_{n_{EE}}$ the EE set of $n_{EE} = n/k$ patterns. We also define $\hat{L}_{n_{MS}}(f)$ and $\hat{L}_{n_{EE}}(f)$ as the empirical errors, respectively, on the MS and the EE set. Unfortunately, a single splitting is often not sufficient to obtain reliable models and the corresponding error estimations, as reported in the recent literature: in particular, it can be shown that the variance of the error obtained by using the k folds, both during the MS and the EE phases, can be large in some cases [13]. As an alternative, we propose to use the Complete KCV (C-KCV) procedure [12], which consists in considering all the possible combinations of subsets in which the original dataset can be split, given a fixed number of folds k. The number of combinations is $l_C = \binom{n/k}{n}\binom{n/k}{n-(n/k)}$; however, in practice, a Monte Carlo procedure can be exploited and $l_{MC} << l_C$ combinations are actually used. It is worth noting that the C-KCV method is trivially parallelizable, so that novel high–performance multi–core computing architectures can be exploited to speed–up the procedure.

As remarked above, in general, k is aprioristically chosen: typical values are 5, 10 and 20, which allow to reserve a large fraction of data for the TR phase at the expense of obtaining a looser generalization error estimation, since few data can be used for EE purposes. We propose, instead, to consider the number of folds as a hyperparameter, which can assume any value in the set $k \in \{3, \ldots, n\}$, so to automatically weigh up the number of patterns used to create the model and the percentage of $\mathcal{D}_n$, which is exploited to estimate the classifier performance on previously unseen data. Therefore, in our case, where a Gaussian kernel is used, we have to tune the triple (C, γ, k).

ESANN 2012 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 25-27 April 2012, i6doc.com publ., ISBN 978-2-87419-049-0.
Available from http://www.i6doc.com/en/livre/?GCOI=28001100967420.

The algorithm for implementing the C-KCV procedure, where k is tuned accordingly to a set of data, is detailed in Algorithm 1. For every value of the hyperparameters triple, the available set is randomly split l_{MC} times into the three subsets, whose cardinalities depend on k. Then, at every step of the Monte Carlo procedure, a classifier f_{TR} is trained on $\mathcal{D}_{n_{TR}}$ and the empirical error on $\mathcal{D}_{n_{MS}}$ is computed. As the validation sets are characterized, for different values of k, by different cardinalities, in order to fairly compare the models during the MS phase, we apply the CP bound for MS purposes as well, obtaining the index of performance $L_{MS}(f_{TR})$. Note that, during the MS phase, we do not minimize the empirical error on the validation set: the exploitation of the CP bound allows, instead, to prevent overfitting. Subsequently, in order to exploit the largest set of data which allows to guarantee, at the same time, the statistical rigorousness of the approach, as also suggested in [4], we create a new set by joining $\mathcal{D}_{n_{TR}}$ and $\mathcal{D}_{n_{MS}}$ and retrain a classifier on it (f^*_{MS}). The obtained model is then used for the EE phase, where the CP bound of Eq. (3) is exploited, and added to a set of classifiers $\mathcal{F}$. Once the MS phase ends, the best triple (C^*, γ^*, k^*) is available as an output of the algorithm, as well as the best set of models $\mathcal{F}^*$ and the corresponding rigorous error estimation $L(f) \leq L^*_{EE}$. Note that, every time a new pattern must be classified, a model must be randomly chosen within $\mathcal{F}^*$ in order to guarantee the statistical rigorousness of the error estimation bound.

Algorithm 1 Modified Complete KCV learning procedure

$L^*_{MS} = +\infty$
for all $C \in [0, +\infty)$, $\gamma \in [0, +\infty)$, $k \in \{3, \ldots, n\}$ **do**
  $L_{MS} = 0$, $L_{EE} = 0$, $\mathcal{F} = \varnothing$
  for $i = 1 \rightarrow l_{MC}$ **do**
    $\{\mathcal{D}_{n_{TR}}, \mathcal{D}_{n_{MS}}, \mathcal{D}_{n_{EE}}\} = \text{Random Split}(\mathcal{D}_n, k)$
    $f_{TR} = SVM(\mathcal{D}_{n_{TR}}, C, \gamma)$
    Compute $\hat{L}^{(i)}_{n_{MS}}(f_{TR})$, $L^{(i)}_{MS} = \text{Clopper}\left(\hat{L}^{(i)}_{n_{MS}}(f_{TR})\right)$
    $L_{MS} = L_{MS} + L^{(i)}_{MS}$
    $f^*_{MS} = SVM(\mathcal{D}_{n_{TR}} \cup \mathcal{D}_{n_{MS}}, C, \gamma)$
    Compute $\hat{L}^{(i)}_{n_{EE}}(f^*_{MS})$, $L^{(i)}_{EE} = \text{Clopper}\left(\hat{L}^{(i)}_{n_{EE}}(f^*_{MS})\right)$
    $L_{EE} = L_{EE} + L^{(i)}_{EE}$, $\mathcal{F} = \mathcal{F} \cup \{f^*_{MS}\}$
  end for
  if $L_{MS} < L^*_{MS}$ **then**
    $L^*_{MS} = L_{MS}$, $L^*_{EE} = \frac{L_{EE}}{l_{MC}}$, $\mathcal{F}^* = \mathcal{F}$, $\{C^*, \gamma^*, k^*\} = \{C, \gamma, k\}$
  end if
end for

4 Experimental Results and Future Work

We present in this section the results, obtained by applying our technique to the datasets proposed by G. Rätsch for benchmarking Machine Learning algorithms

ESANN 2012 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 25-27 April 2012, i6doc.com publ., ISBN 978-2-87419-049-0.
Available from http://www.i6doc.com/en/livre/?GCOI=28001100967420.

Dataset [14]	k^*	L_{EE}	$\hat{L}_{n_t}(\mathcal{F}^*)$	$L_{EE}^{k=5}$	$\hat{L}_{n_t}^{k=5}(f_{k=5}^*)$	$L_{EE}^{k=10}$	$\hat{L}_{n_t}^{k=10}(f_{k=10}^*)$
banana	3.9±0.5	15.9±1.0	11.0±0.4	17.3±1.4	11.1±0.3	21.8±2.0	11.4±0.5
breast cancer	3.4±0.5	36.3±1.0	26.1±1.9	39.3±1.1	27.0±3.3	46.5±3.1	26.8±3.7
diabetis	3.2±0.3	29.4±0.9	23.3±1.0	31.2±1.0	23.3±1.6	34.5±1.1	23.6±1.4
flare solar	3.7±0.9	38.5±0.6	33.0±1.4	39.6±1.0	33.1±1.4	42.9±0.8	32.9±1.3
german	3.9±0.8	30.1±1.1	23.5±1.0	30.5±0.9	24.1±1.2	34.6±1.4	24.0±1.9
heart	3.4±0.5	27.8±1.8	17.5±2.5	30.7±2.5	18.2±2.8	37.9±3.4	18.4±2.6
image	6.3±1.3	5.7±0.4	2.7±0.3	5.6±0.3	2.8±0.3	6.5±0.4	2.6±0.3
ringnorm	3.1±0.2	4.6±0.6	1.7±0.1	6.2±0.5	1.6±0.1	10.3±1.3	1.8±0.4
splice	5.7±0.9	16.2±0.6	11.5±0.5	16.2±0.9	11.7±0.5	17.6±0.9	11.6±0.5
thyroid	3.3±0.3	12.4±0.9	5.2±2.1	16.1±1.8	6.5±1.8	24.7±1.6	6.0±2.2
titanic	3.2±0.4	34.9±1.8	22.8±0.4	39.5±2.1	22.6±0.4	46.7±1.7	22.7±0.6
twonorm	3.2±0.3	6.3±0.7	2.7±0.1	7.3±0.7	2.7±0.1	10.7±0.8	2.6±0.2
waveform	4.0±0.7	16.2±1.3	10.4±0.5	17.0±2.0	10.6±0.7	21.1±2.0	10.4±0.2

Table 1: Results (in percentage) obtained with the modified C-KCV procedure.

[14]. All the datasets consists in 20 or 100 random splits[1] of an original set of samples into sets for learning ($\mathcal{D}_n$) and for testing purposes ($\mathcal{T}_{n_t}$, n_t samples). The first three columns of Table 1, report the values of k^*, the generalization error L_{EE}^*, obtained during the learning phase, and the actual error $\hat{L}_{n_t}(\mathcal{F}^*)$ performed on the test data $\mathcal{T}_{n_t}$ by the classifiers included in $\mathcal{F}^*$. The subsequent columns show the comparison with the results obtained by applying the conventional KCV approach, when k is fixed to the typical values 5 and 10. The reported values represent both the estimated error $L_{EE}^{k=5,10}$ and the misclassification rate on the test set $\hat{L}_{n_t}^{k=5,10}(f_{k=5,10}^*)$.

By analysing the experimental results, it is clear that the optimal values of k mainly lie between 3 and 4. Note that this choice always guarantees the tighter generalization error estimation value, at the expense of some occasional and, eventually, slight increase of the misclassification rate on the test set. From these experiments, we can also claim that $k = 4$ could represent a good choice, if we had to select an a–priori value for the KCV procedure. However, it is clear that the best value of k is highly data-dependent, thus justifying the use of the proposed approach, even at the expense of considering k as a further hyperparameter to tune, which affects the computational effort required by the entire learning procedure.

It is also worth noting that the generalization error estimations, despite being obtained with the tightest bound for Binomial distributions, are sometimes loose if compared with the error rates, obtained on the test sets $\mathcal{T}_{n_t}$: unfortunately, this issue is not easy to circumvent, as CP represents the tightest available bound in literature for binary classification problems when a hard loss function is used. An open problem is hence to investigate the possibility of using different loss functions and different bounds to reduce the looseness of the estimations.

[1]For computational reasons only the first 10 realizations are used in our experiments.

ESANN 2012 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 25-27 April 2012, i6doc.com publ., ISBN 978-2-87419-049-0.
Available from http://www.i6doc.com/en/livre/?GCOI=28001100967420.

5 Conclusions

We proposed a modified Complete K-fold Cross Validation approach, which allows to automatically trade-off the percentage of data, used to train a classifier, and the tightness of the estimated error, by considering the number of folds as a hyperparameter to be tuned during the Model Selection phase. While k, in practice, is usually set to some fixed number, we have shown through some tests on well-known benchmarking datasets that the proposed approach allows to obtain tighter generalization error bounds, only at the expense of an occasional and, eventually, slight increase of the misclassification rate on the test set. However, further work has to be carried on, targeted towards tightening the discrepancy between the predicted and the actual performance of a model.

References

[1] V.N. Vapnik. *Statistical learning theory*. Wiley-Interscience, 1998.

[2] O. Chapelle, V. Vapnik, O. Bousquet, and S. Mukherjee. Choosing multiple parameters for support vector machines. *Machine Learning*, 46(1):131–159, 2002.

[3] I. Guyon, A. Saffari, G. Dror, and G. Cawley. Model selection: beyond the bayesian–frequentist divide. *The Journal of Machine Learning Research*, 11:61–87, 2010.

[4] D. Anguita, Ghio A., S. Ridella, and D. Sterpi. K-fold cross validation for error rate estimate in support vector machines. In *Proc. of the Int. Conf. on Data Mining*, 2009.

[5] T.G. Dietterich. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural computation*, 10(7):1895–1923, 1998.

[6] C.W. Hsu, C.C. Chang, and C.J. Lin. A Practical Guide to Support Vector Classification. Technical report, 2000.

[7] T. Hastie, R. Tibshirani, J. Friedman, and J. Franklin. The elements of statistical learning: data mining, inference and prediction. *The Mathematical Intelligencer*, 27(2):83–85, 2005.

[8] L. Roberge, S. B. Long, and D. B. Burnham. Data warehouses and data mining tools for the legal profession: using information technology to raise the standard of practice. *Syracuse Law Review*, 2002.

[9] B. L. Milenova, J. S. Yarmus, and M. M. Campos. Svm in oracle database 10g: Removing the barriers to widespread adoption of support vector machines. In *Proc. of the 31st Int. Conf. on Very Large Data Bases*, pages 1152–1163, 2005.

[10] C.J. Clopper and E.S. Pearson. The use of confidence intervals for fiducial limits illustrated in the case of the binomial. *Biometrika*, 1934.

[11] D. Anguita, L. Ghelardoni, A. Ghio, and S. Ridella. A survey of old and new results for the test error estimation of a classifier. *Journal of Artificial Intelligence and Soft Computing Research (in press)*.

[12] R. Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *International joint Conference on artificial intelligence*, volume 14, pages 1137–1145, 1995.

[13] S. Arlot and A. Celisse. A survey of cross-validation procedures for model selection. *Statistics Surveys*, 4:40–79, 2010.

[14] G. Rätsch, T. Onoda, and K.R. Müller. Soft margins for adaboost. *Machine Learning*, 42(3):287–320, 2001.