# Leave-one-out error and stability of learning algorithms with applications

André Elisseeff
Max Planck Institute for Biological Cybernetics
Spemannstrasse 38, 72076 Tuebingen Germany
*e-mail: andre.elisseeff@tuebingen.mpg.de*

Massimiliano Pontil
Department of Information Engineering, University of Siena
Via Roma 56, 53100 Siena Italy
*e-mail: pontil@dii.unisi.it*

**Abstract.** The leave-one-out error is an important statistical estimator of the performance of a learning algorithm. Unlike the empirical error, it is almost unbiased and is frequently used for model selection. We review attempts aiming at justifying the use of leave-one-out error in Machine Learning. We especially focus on the concept of stability of a learning algorithm and show how this can be used to formally link leave-one-out error to the generalization error. Stability has also motivated recent work on averaging techniques similar to bagging which we briefly summarize in the paper. The ideas we develop are illustrated in some details in the context of kernel-based learning algorithms.

## 1 Introduction

Assessing the performance of different learning algorithms on a dataset is at the core of Machine Learning. If this task is done properly, the best algorithm can be selected and the problem of generalization is partially solved. For that reason, many studies have focused on the design of scores that led to different model selection strategies. As examples, we could cite techniques based on minimum description length [37], margin bounds [7], bayesian evidence [31], metric structures [40], etc. A complete list would include many more techniques.

When it comes to practice, many practitioners use cross validation methods. This is remarkable from two related standpoints. First, cross validation has been seldom studied in the literature compared to some of the above mentioned techniques. Second, as described in [27], “In spite of the practical importance of this estimate [cross validation], relatively little is known about its theoretical properties”.

In this paper, we try to gather information about one particular instance of cross validation, namely the leave-one-out error, in the context of Machine Learning and mostly from stability considerations. By limiting the subject, we hope to provide a consistent view and we apologize for not quoting important works that we might miss.

*Outline of the paper*

After a general discussion about leave-one-out error (Section 1), we present different analysis trying to explain why and how this technique works (Section 2). In particular we focus on concepts of stability which allow to derive probabilistic bounds on the generalization error of learning algorithms. The bounds say that the difference between expected and leave-one-out error is small when the algorithm is stable. This is discussed in Section 3 where we also present other attempts to justify leave-one-out error. In Section 4 we illustrate the use of leave-one-out error and stability in the context of kernel machines, a family of learning algorithms which has gained great popularity over the past few years. In this case it is possible to derive estimates of the leave-one-out error which only require the knowledge of the machine trained once on the full dataset. In Section 5 we overview the use of leave-one-out error in learning problems other than classification and regression.

*Notation*

In the following, calligraphic font is used for sets and capital letters refer to numbers unless explicitly defined. Let $\mathcal{X}$ and $\mathcal{Y}$ be two Hilbert spaces and define $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$. $\mathcal{X}$ is identified as the input space and $\mathcal{Y}$ as the output space. Given a learning algorithm $A$, we define $f_{\mathcal{D}}$ to be the solution of the algorithm when the training set $\mathcal{D} = \{z_i = (x_i, y_i),\ i = 1, \ldots, m\} \in \mathcal{Z}^m$ drawn i.i.d. from a distribution $\mathbb{P}$ is used. $A$ is thus interpreted as a function from $\mathcal{Z}^m$ to $\mathcal{Y}^{\mathcal{X}}$ - the set of all functions from $\mathcal{X}$ to $\mathcal{Y}$ - and we use the notation: $A(\mathcal{D}) = f_{\mathcal{D}}$. We denote by $\mathcal{D}^i$ the training set obtained by removing the point $(x_i, y_i)$ from $\mathcal{D}$. $f_{\mathcal{D}}$ is sometimes denoted by $f$ and $f_{\mathcal{D}^i}$ by $f^i$. The expectation with respect to (w.r.t.) the sample $\mathcal{D}$ is denoted by $\mathbf{E}_{\mathcal{D}}[\cdot]$.

For any point $(x, y)$ and function $f$ we denote by $\ell(f(x), y)$ the error made when $f(x)$ is predicted instead of $y$ ($\ell$ is the loss function). We also sometimes write $\ell(f, z)$ instead, where $z = (x, y)$. The generalization error of $f$ w.r.t. loss $\ell$ is

$$R_{\text{gen}}(f) = \mathbf{E}_z\left[\ell(f, z)\right].$$

where $\mathbf{E}_z[\cdot]$ denotes the expectation w.r.t. $z$. We define as well the *empirical error* w.r.t. the same loss function by

$$R_{\text{emp}}(f) = \frac{1}{m} \sum_{i=1}^{m} \ell(f, z_i).$$

In the paper we focus on regression, $\mathcal{Y} = \mathbb{R}$ and on binary classification, $\mathcal{Y} = \{-1, 1\}$. In the latter case we denote by $\theta(\cdot)$ the Heavyside function and let the *misclassification error* of function $f$ be the generalization error of $f$ w.r.t. loss $\theta(-yf(x))$.

## 2 General observations about leave-one-out error

It seems that leave-one-out error (also called *deleted estimate* or *U-method*) has been defined in the late sixties/mid-seventies. It appears in different papers by Lachenbruch [29], Luntz and Brailovsky [30], Cover [10], and Stone [42]. For a learning algorithm $A$ producing an outcome $f_{\mathcal{D}}$, it is defined as

**Definition 2.1 (Leave-one-out error).**

$$R_{\text{loo}}(f_{\mathcal{D}}) = \frac{1}{m}\sum_{i=1}^{m} \ell(f^i, z_i)$$

and is supposed to be an "almost" unbiased estimate of the generalization error of $f_{\mathcal{D}}$. Because it seems to share many properties with a technique called Jackknife introduced by Tukey [43, 35], it is worth pointing out that the leave-one-out error is different. The latter concerns indeed a learning algorithm whose output is computed on a point that has not been used during training. The former consists in using repeatedly the whole training set but one point, computing many estimators and combining them at the end. This combination should lead to a new estimator whose bias is supposed to be low. The Jackknife can be used to derive an estimate of the generalization error based on the empirical error but things are then more complicated than for the leave-one-out error estimate (see, e.g., [36]).

The first result that is generally given to motivate the leave-one-out error is the following:

**Theorem 2.1 (Luntz and Brailovsky [30]).** *The leave-one-out estimate is almost unbiased in the following sense:*

$$\mathbf{E}_{\mathcal{D}}\left[R_{\text{loo}}(f_{\mathcal{D}})\right] = \mathbf{E}_{\mathcal{D}'}\left[R_{\text{gen}}(f_{\mathcal{D}'})\right] \tag{1}$$

*where $\mathcal{D}'$ is set of $m-1$ points in $\mathcal{Z}$.*

On average, the leave-one-out error should be relatively informative about the generalization error of the same algorithm when trained on $m-1$ points. In practice, it is generally admitted that removing one point of the training set does not change much the outcome of the algorithm. In theory, such an assumption cannot be made so easily. Consider $\mathcal{D}' \subset \mathcal{D}$ a training set of $m-1$ points. If we assume that $f_{\mathcal{D}} \approx f_{\mathcal{D}'}$ for all training sets $\mathcal{D}$, then $R_{\text{emp}}(f_{\mathcal{D}}) \approx R_{\text{loo}}(f_{\mathcal{D}})$ and using the leave-one-out estimate is roughly the same as using the empirical error. The latter however is well known to be highly biased and does not give a good indication on the generalization performance of the algorithm. The apparent contradiction in this reasoning could have been removed if we had defined a more precise definition of what we mean by stable. For simplicity, let us consider until the end of the section that the learning problem is a classification problem. We define:

**Definition 2.2 (Hypothesis stability [14]).** *The hypothesis stability $\beta(f_{\mathcal{D}})$ of a symmetric*[1] *learning algorithm $A$ whose outcome is $f_{\mathcal{D}}$ is defined as:*[2]

$$\beta(f_{\mathcal{D}}) = \mathbf{E}_{\mathcal{D},z}\left[\left|\ell(f_{\mathcal{D}}, z) - \ell(f_{\mathcal{D}^i}, z)\right|\right]. \tag{2}$$

This definition captures a notion of stability that suits our purpose: if one point is removed, the difference in the outcome of the learning algorithm will be measured by the averaged absolute difference of the losses. We can now naturally relate the generalization error of $f_{\mathcal{D}}$ with the one of $f_{\mathcal{D}'}$. We have indeed:

$$\left|\mathbf{E}_{\mathcal{D}}\left[R_{\text{gen}}(f_{\mathcal{D}}) - R_{\text{gen}}(f_{\mathcal{D}'})\right]\right| = \left|\mathbf{E}_{\mathcal{D},z}\left[\ell(f_{\mathcal{D}}, z) - \ell(f_{\mathcal{D}'}, z)\right]\right| \leq \beta(f_{\mathcal{D}})$$

[1]An algorithm is said to be symmetric if its outcome, $f_{\mathcal{D}}$, does not change when the elements of the training set are permuted.

[2]Note that, since the algorithm is symmetric, when averaging over $\mathcal{D}$, specifying which point is left out is irrelevant: the r.h.s of Equation (2) is the same for all $i = 1, .., m$.

so that the bias of the leave-one-out error is bounded by the hypothesis stability of the learning algorithm. Note that the hypothesis stability does not imply that the empirical error is close to the leave-one-out.

The following example shows how to compute hypothesis stability for the $k-$nearest neighbor ($k-$NN) algorithm.

**Example 2.1 (Hypothesis Stability of $k-$NN).** *With respect to the classification loss, $k-$NN is $\frac{k}{m}$ stable. This can be seen via symmetrization arguments. For sake of simplicity, we give here the proof for the $1-$NN only. Let $v_i$ be the neighborhood of $z_i$ such that the closest point of the training set to any point in $v_i$ is $z_i$. The nearest neighbor machine computes its output via the following equation (we assume here that the probability that $\mathbf{x}_i$ appears twice in the training set is negligible):*

$$f_{\mathcal{D}}(\mathbf{x}) = \sum_{i=1}^{m} y_i \mathbf{1}_{\mathbf{x}\in v_i}(\mathbf{x})$$

*where $\mathbf{1}_A$ is the indicator function of set $A$. The difference between the losses $\ell(f_{\mathcal{D}}, z)$ and $\ell(f_{\mathcal{D}^i}, z)$ is then defined by the set $v_i$. Here we assume that $\ell$ is the classification loss. We have then:*

$$\mathbf{E}_z[|\ell(f_{\mathcal{D}^i}, z) - \ell(f_{\mathcal{D}}, z)|] \leq \mathbb{P}(v_i).$$

*Note that $v_i$ depends on $\mathcal{D}$. Now averaging over $\mathcal{D}$ we need to compute $\mathbf{E}_{\mathcal{D}}\left[\mathbb{P}(v_i)\right]$ which is the same for all $i$ because the $z_i$ are drawn i.i.d. from the same distribution. But, we have,*

$$1 = \mathbf{E}_{\mathcal{D},z}\left[|f_{\mathcal{D}}(\mathbf{x})|\right] = \mathbf{E}_{\mathcal{D},z}\left[\left|\sum_{i=1}^{m} y_i \mathbf{1}_{\mathbf{x}\in v_i}(\mathbf{x})\right|\right] = \mathbf{E}_{\mathcal{D},z}\left[\sum_{i=1}^{m} \mathbf{1}_{\mathbf{x}\in v_i}(\mathbf{x})\right].$$

*The last inequality comes from the fact that for fixed $\mathcal{D}$ and $z$, only one $\mathbf{1}_{\mathbf{x}\in v_i}(\mathbf{x})$ is non-zero. We have then:*

$$1 = \mathbf{E}_{\mathcal{D},z}\left[\sum_{i=1}^{m} \mathbf{1}_{\mathbf{x}\in v_i}(\mathbf{x})\right] = m\mathbf{E}_{\mathcal{D}}\left[\mathbb{P}(v_i)\right].$$

*So that: $\mathbf{E}_{\mathcal{D}}\left[\mathbb{P}(v_i)\right] = \frac{1}{m}$. And finally, the $1-$NN has a hypothesis stability bounded above by $1/m$.*

Thus for nearest neighbor methods, the leave-one-out error is truly almost unbiased: for the 1-NN, the average difference between $f_{\mathcal{D}'}$ and $f_{\mathcal{D}}$ is bounded by $1/m$. This statement does not hold for all algorithms: when the algorithm is unstable, the use of leave-one-out error is not recommended. Consider for instance a (quite stupid) algorithm whose outcome is the constant function $0$ when $m$ is odd and the constant function $1$ when $m$ is even. The leave-one-out estimate would then be totally wrong. One might object that this example is quite artificial but actually such instabilities might occur when the value of $m$ is small. A hard margin support vector machine (SVM) separating 3 points in a two dimensional input space is unstable: if one point is removed, the outcome of the SVM usually changes a lot.

Another case of failure for the leave-one-out estimate has been pointed out by Shao [41] who studied model selection with a linear regression method. Shao proved that, asymptotically when $m$ tends to infinity, the probability to select the optimal model based on the leave-one-out error is not equal to one: model selection based on leave-one-out estimate tends to choose unnecessarily large models. This failure is not the privilege of the leave-one-out estimate. The latter is indeed equivalent to other criterions that inherit as well from this inconsistency. These criterions are the Akaike information criterion [1] and Mallows's $Cp$ [32], which are proved to be equivalent to leave-one-out in the particular setting of quadratic linear regression when $m$ tends to infinity.

A common belief is that the leave-one-out estimate has a large variance: when different training sets are sampled from the same distribution, the variance of the leave-one-out error computed over these samplings is generally larger than 10-fold cross validation. This statement has been observed practically by Kohavi [26]. The latter also showed examples where leave-one-out error fails completely due to the instability of the learning algorithm. Theoretically, the variance of the leave-one-out error has been computed in the book of Vapnik [44] (p.236) in the special case of a gaussian noise and for linear models. It is shown that the variance of the leave-one-out error in this setting is equivalent to the variance of a hold-out estimate.[3]

At last, except for few special cases, the leave-one-out estimate is very time-consuming. For that reason, many works have been produced in order to derive easy-to-compute bounds. These bounds seem however to be specific to linear models. This rules out many algorithms for which the leave-one-out error will still be long to compute.

Despite all these handicaps, the leave-one-out estimate is used by many practitioners. It is generally believed as to be a fairly good estimator albeit not the best and it has been used successfully for model selection (see for instance the work of Chapelle *et al.* [8]). All these facts might then seem contradictory. Following the no free lunch theorem [48], we are tempted to conclude like Goutte [20] that leave-one-out is as bad as any other estimator but that we have not found practical problems for which it completely fails. Coming back to stability considerations, this observation might indicate that most methods that are currently used by practitioners are actually stable with respect to the removal of one point in the training set.

## 3 Theoretical attempts to justify the use of leave-one-out error

In the previous section we have mainly focused on basic facts that are observed from practice or that have been deduced from the classical statistical framework. In this section, we will present theoretical approaches that have been inspired by machine learning and which seem to have been directed by the need to understand why this estimate was better than empirical error.

### *3.1 Early work in non-parametric statistics*

In the late seventies, Rogers, Devroye and Wagner wrote a series of papers about $k$-nearest neighbor ($k-$NN), local, histograms and potential rules. For $k-$NN and $k-$local rules (i.e. rules which compute their output from the $k$ closest points in the training set), they derived exponential bounds on the probability that the leave-one-out error[4] deviates from the generalization error [14].

[3]The number of points held out must be greater than $m$ minus the dimension of the input space and under general conditions, the training set is the same set used to compute the leave-one-out error. Note that in his book, Vapnik uses the term "moving control estimator" rather than leave-one-out estimate.

[4]Note that the leave-one-out error for $k-$NN is very easy to compute.

**Theorem 3.1 (Devroye and Wagner [14]).**

$$\mathbb{P}\left(R_{\text{loo}}\left(k\text{--}NN\right) - R_{\text{gen}}\left(k\text{--}NN\right) > \epsilon\right) \le 2e^{-m\epsilon^2/18} + 6e^{-m\epsilon^3/(108k(2+\gamma_d))} \tag{3}$$

*where $\gamma_d$ is the maximum number of distinct points in $\mathbb{R}^d$ which can share the same nearest neighbor.*

The capacity term also called VC-dimension[45] that generally occurs in statistical learning theory bounds is replaced here by a metric concept $\gamma_d$ whose value seems quite difficult to compute (see [14]). This value does not depend on $m$ but goes to infinity when $d$ tends to infinity (it is lower bounded by the $d$). This means that such bound does not motivate the use of $k-$NN in infinite dimensional reproducing kernel Hilbert spaces, that is, $k-$NN used with kernels (see Section 4).

For more general classifiers, Devroye, Rogers and Wagner showed that:

**Theorem 3.2 (Devroye *et al.* [13, 38]).**

$$\mathbf{E}_{\mathcal{D}}\left[\left(R_{\text{loo}}\left(f_{\mathcal{D}}\right) - R_{\text{gen}}\left(f_{\mathcal{D}}\right)\right)^2\right] \le \frac{1}{m} + \mathbb{P}\left(f_{\mathcal{D}} \neq f_{\mathcal{D}^i}\right). \tag{4}$$

These results have been extended to potential function rules that include classifiers based on Parzen window density estimators (see [15] for more details). In order to get a small variance, this bound suggests that the stability should be small. Since with Tchebytchev's inequality a bound on the variance induces a bound on the difference between leave-one-out and generalization error, it is straightforward to relate stability to generalization and to support the common intuition that if an algorithm is stable then its leave-one-out error is close to its generalization. Such stability considerations have been reconsidered in the late nineties with the work of Kearns and Ron [25].

## *3.2 Relation to VC-theory*

The work of Kearns and Ron aims at proving sanity check bounds for leave-one-out error. They prove that the bounds derived for leave-one-out error are not worse than those derived for empirical error. This may seem quite reassuring in the sense that theory does not say that leave-one-out is worse than empirical error. Their result is based on the following definition of stability:

**Definition 3.1 (Error Stability [25]).** *We say that a deterministic (and symmetric) algorithm $A$ has error stability $(\beta_1, \beta_2)$ if:*

$$\mathbb{P}\left(\left|R_{\text{gen}}\left(f_{\mathcal{D}}\right) - R_{\text{gen}}\left(f_{\mathcal{D}^i}\right)\right| \ge \beta_2\right) \le \beta_1.$$

They showed that this notion combined with another notion of "overestimating the empirical error" which controls how much the leave-one-out error overestimates the empirical error, leads to the following bound: $\forall \delta > 0$, with probability $1-\delta$ over the sampling of the training set, assuming that $f_{\mathcal{D}}$ is an algorithm minimizing the empirical error over a set of functions with VC dimension $d$ (see [45] for an account about Vapnik Chervonenkis dimension and the theory that is derived), we have:

$$\left|R_{\text{loo}}\left(f_{\mathcal{D}}\right) - R_{\text{gen}}\left(f_{\mathcal{D}}\right)\right| \le \left(8\sqrt{\frac{(d+1)(\ln(9m/d)+2)}{m}}\right)/\delta. \tag{5}$$

This bound is very similar to those that are derived for the empirical error. Actually it is not so surprising since the analysis of Kearns and Ron is based on VC-theory. They also showed that there exists an algorithm minimizing the empirical error over a set of VC dimension $d$ for which the left-hand side of Eq. (5) is lower bounded in $\Omega(d/m)$. This statement is not contradictory with bounds on the leave-one-out error for 1-nearest neighbor. It just means that the latter is not this algorithm. On the other hand, it shows that if a bound on the difference between the leave-one-out and the generalization error is found then it must be very specific on the algorithm or it will be like for the empirical error, in $\Omega(d/m)$.

Another related contribution to the theory of cross validation is by Holden [23]. Unfortunately, the results do not apply to leave-one-out error and holds only for cross validation when the fold left out is sufficiently large.

It seems that so far the best success that theory has been able to achieve for leave-one-out error has been met by the work of Devroye, Rogers and Wagner and by the notion of stability.

## *3.3 Stability*

Recently, stability considerations have been revisited by Bousquet and Elisseeff [5] who proved exponential bounds for stable algorithms.

**Definition 3.2 (Uniform stability [5]).** *Let $f_{\mathcal{D}}$ be the outcome of a symmetric and deterministic learning algorithm. We define the uniform stability $\beta(f_{\mathcal{D}})$ with respect to a loss function $\ell$ by*

$$\beta(f_{\mathcal{D}}) = \sup_{\mathcal{D}} \|\ell(f_{\mathcal{D}}, .) - \ell(f_{\mathcal{D}^i}, .)\|_\infty \tag{6}$$

¿From this definition, it is possible to prove the following theorem:

**Theorem 3.3 (Bousquet and Elisseeff [5]).** *Let $f_{\mathcal{D}}$ be the outcome of an algorithm with uniform stability $\beta(f_{\mathcal{D}})$ with respect to a loss function $\ell$ such that $0 \leq \ell(f_{\mathcal{D}}, y) \leq M$, for all $y \in \mathcal{Y}$ and all set $\mathcal{D}$. Then, for any $m \geq 1$, and any $\eta \in (0,1)$, the following bound holds with probability at least $1-\eta$ over the random draw of the sample $\mathcal{D}$,*

$$\mathbb{P}\left(\left|R_{\text{gen}}\left(f_{\mathcal{D}}\right) - R_{\text{loo}}\left(f_{\mathcal{D}}\right)\right| \geq \beta_m + \epsilon\right) \leq e^{-2m\epsilon^2/(4m\beta(f_{\mathcal{D}})+M)^2} \tag{7}$$

The bound presented in this theorem is interesting only if the stability $\beta(f_{\mathcal{D}})$ decreases as $1/m^a$ with $a > 1/2$.

The uniform stability of regularization algorithms such as regularization networks [33] or support vector machines [4, 45] are computed in [5]. This extends the results of Devroye, Rogers and Wagner to other learning techniques but at the cost of a more restrictive definition of stability. Hypothesis stability is indeed upper bounded by uniform stability and classic algorithms such as $k-$NN do not have an interesting uniform stability with respect to the classification loss: it is equal to 1 for the $1-$NN. It seems however that uniform stability is the cost to pay to get exponential bounds rather than polynomial as it is the case for the hypothesis stability.

A less restrictive notion of stability that has been introduced lately by Kutin and Niyogi [28] might provide a better notion than uniform stability.

**Definition 3.3 (Partial stability [28]).** *Let $f_{\mathcal{D}}$ be the outcome of a symmetric and deterministic learning algorithm $A$. We say that $A$ is $(\delta, \beta)$ partially stable with respect to a loss function $\ell$ if:*

$$\mathbb{P}\left(\forall i \in \{1, \ldots, m\},\ \|\ell(f_{\mathcal{D}}, .) - \ell(f_{\mathcal{D}^i}, .)\|_\infty \leq \beta\right) \geq 1 - \delta. \tag{8}$$

Kutin and Niyogi have derived bounds for the empirical error but it is possible to easily extend their results to bound the leave-one-out error as in (7), except that the uniform stability is replaced by the partial stability. We refer the reader to [28] for more details.

We see that there exist many notions of stability that lead to bounds on the generalization error in terms of the leave-one-out estimate. The choice of the stability measure is guided by the desired type of bound (Chebytchev's type with Hypothesis stability and exponential bounds with Uniform and partial stability) and the knowledge of the learning algorithm which might be translated into a bound over its stability. In next section, we show as an aside a way of improving the stability of a learning algorithm by averaging techniques.

### *3.4 Stability of averaging techniques*

Suppose that a learning algorithm $A$ has uniform stability $\beta_m = O(1/m^a), a > 0$. When $a = 1$ we say that $A$ is stable. In this case Theorem 3.3 implies that

$$|R_{\text{gen}}(f_{\mathcal{D}}) - R_{\text{loo}}(f_{\mathcal{D}})| = O(1/\sqrt{m}).$$

If $A$ is not stable, a way to stabilize it is by averaging its solution trained on small bootstrap subsets of the training set [17]. In particular, consider the function

$$F^k_{\mathcal{D}}(x) = \mathbf{E}_S[f_S(x)] \tag{9}$$

where $\mathbf{E}_S$ denotes the expectation with respect to $k$ points sampled in $\mathcal{D}$ with the uniform distribution, i.e. $S$ contains $k$ points of $\mathcal{D}$ obtained by uniform sampling without replacement. When $k = m$ this method is Breiman's bagging [6], this variation is named *subagging*. The next theorem provides a link between the uniform stability of the average combination and that of algorithm $A$.

**Theorem 3.4 (Evgeniou *et al.* [17]).** *Assume that $f_{\mathcal{D}}$ has stability $\beta_m$. Then, the stability, $\hat{\beta}_m$ of the average combination $F^k$ in Eq. (9) is bounded as*

$$\hat{\beta}_m \leq \frac{k}{m}\beta_k.$$

Therefore, the average combination is always stable in the sense that $\hat{\beta}_m = O(1/m)$. In practice the average combination is replaced by a finite combination. In this case the stability theory does not hold because of the randomization of the solution. Recent work in [16] extends the stability concepts to randomized algorithms and presents a closer look at averaging techniques.

## 4 Kernel machines

In this section we focus on kernel machines [45, 18, 11, 39], a family of learning algorithms based on reproducing kernel Hilbert spaces. We begin by recalling the main features of kernel machines. Then, we review leave-one-out error and stability results.

### *4.1 Background on kernel machines*

Kernel machines are the minimizers of regularization functionals of the form:

$$\frac{1}{m}\sum_{i=1}^{m}\ell(f(x_i),y_i)+\lambda\|f\|_K^2 \tag{10}$$

where $f$ is a function $\mathcal{X}\to\mathbb{R}$ belonging to a reproducing kernel Hilbert space (RKHS) $\mathcal{H}_K$ defined by a symmetric and positive definite kernel $K: X\times X\to\mathbb{R}$, and $\|f\|_K^2$ is the norm of $f$ in this space. More precisely, the RKHS is the completion of the span of the set $\{K(x,\cdot), x\in\mathcal{X}\}$ with the inner product induced by

$$\langle K(x,\cdot),K(t,\cdot)\rangle = K(x,t). \tag{11}$$

Thus, if $t_1,\dots,t_n\in\mathcal{X}$, the function $\sum_{i=1}^{n}c_iK(t_i,x)$ belongs to the RKHS and its squared norm is $\sum_{i,j=1}^{n}c_ic_jK(t_i,t_j)$. In particular $\|K(x,\cdot)\|^2=K(x,x)$. The next fundamental property of $\mathcal{H}_K$ follows from Equation (11):

$$f(x)=\langle f,K(x,\cdot)\rangle.$$

for every $f\in\mathcal{H}$, $x\in X$. Using Cauchy-Schwartz inequality we obtain another important property of the RKHS:

$$|f(x)|=|\langle f,K(x,\cdot)\rangle|\le\|f\|_K\sqrt{K(x,x)}. \tag{12}$$

The RKHS norm is a measure of smoothness of the function, e.g. a Sobolev norm and, so, the regularization parameter $\lambda>0$ trades-off between small empirical error and smoothness of the solution. A correct choice of $\lambda$ prevents from overfitting. In practice $\lambda$ is selected by means of cross validation.

RKHS are discussed in [3]. A nice introduction relevant to kernel machines can be found in [12]. For more information on positive definite kernels see [19].

The loss function $\ell$ is assumed to be convex. Its choice determines different learning techniques, each leading to a different learning algorithm (for computing the coefficients $\alpha_i$ - see below). Important cases are regularization networks [33] and support vector machines (SVMs) [45]. They are summarized in Table 1. When $X=\mathbb{R}^n$ and $K(x,t)$ is the euclidean dot product and $\ell$ is the square loss, we have the older ridge regression [22].

| Learning method | $\ell(f,y)$ | $S(\alpha)$ | Constraints |
|---|---|---|---|
| Regularization network | $(y-f)^2$ | $y_i\alpha_i-\frac{1}{4C}\alpha_i^2$ | NO |
| SVM classification | $(1-yf)_+$ | $\alpha_i$ | $0\le\alpha_i\le C$ |
| SVM regression | $(\lvert y-f\rvert-\epsilon)_+$ | $y_i\alpha_i-\epsilon\lvert\alpha_i\rvert$ | $\lvert\alpha_i\rvert\le C$ |
| Logistic regression | $\ln(\tanh(yf))$ | $h(\frac{y_i}{C}\alpha_i)$ | $0\le\alpha_i\le C$ |

Table 1: Some kernel machines used in practice. Function $(z)_+$ equals $z$ when $z>0$ and zero otherwise. Function $h$ is the binary entropy: $h(z)=-z\ln(z)-(1-z)\ln(1-z)$. Logistic regression is discussed in [24].

Under rather general conditions the solution of Equation (10) is of the form[5]

$$f(x)=\sum_{i=1}^{m}\alpha_iK(x_i,x). \tag{13}$$

[5]For simplicity, we assume that the bias term is incorporated in the kernel $K$.

The coefficients $\alpha_i$ in Equation (13) are learned by solving the optimization problem:

$$\max_\alpha \left\{ \sum_{i=1}^m S(\alpha_i) - \frac{1}{2} \sum_{i,j=1}^m \alpha_i \alpha_j K_{ij} \right\} \tag{14}$$

where $S(\cdot)$ is a concave function whose form depends on $\ell$, and we used the shorthand $K_{ij}$ for $K(x_i, x_j)$. In particular, in the case of SVM classification $S(\alpha) = \alpha$ if $\alpha \in [0, C]$ and infinite otherwise. Here $C = 1/(2m\lambda)$. Thus, SVMs solve the following quadratic programming problem

$$\begin{array}{l} \max_\alpha \left\{ \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i,j=1}^m \alpha_i \alpha_j K_{ij} \right\} \\ \text{subject to}: \ 0 \le \alpha_i \le C, \ i = 1, \ldots, m. \end{array} \tag{15}$$

The points for which $\alpha_i > 0$ are called support vectors. They are those points for which $y_i f(x_i) \le 1$[6].

In the case of the square loss, it can be easily verified that the coefficients $\alpha_i$ solve a linear system of equations:

$$(K + \lambda m I)\alpha = \mathbf{y} \tag{16}$$

where $\mathbf{y} = (y_1, \ldots, y_m)$ and, abusing notation, we denoted by $K$ the $m \times m$ matrix with elements $K_{ij} = K(x_i, x_j)$.

## 4.2 Leave-one-out for the square loss

For regularization networks the $\alpha$ parameters are the solution of the linear system (16). It is then possible to use standard linear algebra results to obtain

**Lemma 4.1.** *Let $f$ be the minimizer of the functional in Eq. (10), where we choose $\ell$ to be the square loss and $f^i$ the minimizer of the same functional when point $i$ is removed from the training set. Let $A = K(K + \lambda I)^{-1}$. Then*

$$y_i - f^i(x_i) = \frac{y_i - f(x_i)}{1 - A_{ii}}.$$

The interesting feature of this result is that it allows an exact computation of the leave-one-out error on the base of the solution obtained by training on the full dataset only. The following theorem is an immediate consequence of the lemma.

**Theorem 4.1.** *Under the same notation of Lemma 4.1, the leave-one-out error of a regularization network is*

$$\frac{1}{m} \sum_{i=1}^m (y_i - f^i(x_i))^2 = \frac{1}{m} \sum_{i=1}^m \left( \frac{y_i - f(x_i)}{1 - A_{ii}} \right)^2. \tag{17}$$

When the training set is large, computing $A_{ii}$ may be time consuming. In this case it is convenient to use the generalized cross-validation approximation [46]

$$\frac{1}{m} \sum_{i=1}^m (y_i - f^i(x_i))^2 \approx \frac{1}{m} \sum_{i=1}^m \left( \frac{y_i - f(x_i)}{1 - \frac{\text{trace} A}{m}} \right)^2.$$

For more information on generalized cross-validation see [46].

[6]In some special cases, which rarely occur in practice, it may be that $y_i f(x_i) = 1$ and $\alpha_i = 0$.

### *4.3 Bounds on the leave-one-out error and stability*

When the loss function in Eq. (10) is different from the square loss, the leave-one-out error cannot be computed on the base of the solution trained on the full training set only. However, it is possible to derive an upper bound on the error which achieves this. The derivation is based on the following lemma:

**Lemma 4.2 (Zhang [49]).** *Suppose function $S$ in (14) is concave. Let $f$ be the solution of (14) and $f^i$ the solution of the same problem when point $i$ is removed from the training set. Then*

$$\|f - f^i\|_K \leq |\alpha_i| \sqrt{K(x_i, x_i)}.$$

Now, property (12) implies that

$$|f(x_i) - f^i(x_i)| \leq |\alpha_i| K(x_i, x_i). \tag{18}$$

This immediately gives:

**Theorem 4.2.** *The leave-one-out error of a kernel machine which solves problem (14) is upper bounded as*

$$\frac{1}{m} \sum_{i=1}^{m} \ell(f^i(x_i), y_i) \leq \frac{1}{m} \sum_{i=1}^{m} \max_{|\mu| \leq 1} \ell(f(x_i) + \mu \alpha_i K(x_i, x_i), y_i).$$

*In particular, for binary classification*

$$\frac{1}{m} \sum_{i=1}^{m} \theta(-y_i f^i(x_i)) \leq \frac{1}{m} \sum_{i=1}^{m} \theta(|\alpha_i| K(x_i, x_i) - y_i f(x_i)).$$

Similar results were derived in [24, 8] for classification. In this case, last inequality says that a data point is counted as a leave-one-out error if it is either misclassified by $f$ or if by removing its contribution to $f$ changes the classification. For SVMs, the number of leave-one-out errors is no more than the number of support vectors. However if $|\alpha_i| K(x_i, x_i) \ll 1$, support vectors which are correctly classified are likely not to be counted as leave-one-out errors. Thus, the leave-one-out error is close to the empirical error if $C\kappa \ll 1$, with $\kappa = \sup_x K(x, x)$. In this case the SVM is stable. Since $C \propto 1/\mu$, increasing $\lambda$ increases stability. This is something we could expect, because the larger $\lambda$ the smoother the solution. The following result makes this intuition precise.

**Theorem 4.3 (Bousquet and Elisseeff [5]).** *The uniform stability of a kernel machine w.r.t the loss function $\ell$ is bounded as*

$$\beta_m \leq \frac{\kappa \sigma^2}{2m\lambda}$$

*where $\kappa = \sup_{x \in X} K(x, x)$, and $\sigma$ is a constant such that $|\ell(f, y) - \ell(g, y)| \leq \sigma |f - g|$, for every $y \in \mathbb{R}$ and $f, g \in \mathcal{H}_K$.*

Notice that the uniform stability of a kernel machines depends on how the regularization parameter scales with $m$. If $\lambda$ does not depend on $m$ we have $\beta_m = O(1/m)$. Usually, however, $\lambda$ decreases with $m$ and the kernel machine is not stable. In this case averaging helps to improve stability. We discussed this in section 3.4. Reference [5] contains more information about the computation of the constant $\sigma$ appearing in Theorem 4.3.

## 5 The use of leave-one-out error in other learning problems

Leave-one-out error is useful to study learning problems more than the standard classification and regression ones. This section analyzes two important cases in the context of kernel-based learning algorithms.

### 5.1 Transduction

The transduction problem is discussed in [45]. Like in the supervised learning problem, we collect a training set $\mathcal{D}$, but our goal is now to compute only the outputs of a finite set $X_0 = \{x'_1, \ldots, x'_\ell\} \subset X$ (the unlabelled set). When the size of $X_0$ is small the problem can be significantly different form the standard learning problem. An extreme case is $X_0 = \{x'\}$. In this case we only care to compute the optimal output of $x'$ as opposed to the problem of computing a function with optimal average performance on future data. [9] proposes to compute the unknown outputs by minimizing the leave-one-out error of the set $\mathcal{T} = \{(x_1, y_1), \ldots, (x_m, y_m), (x'_1, y'_1), \ldots, (x'_\ell, y'_\ell)\}$. They use ridge regression with $m + \ell$ basis function. Each base is a Gaussian centered on a data point. This is the same as using a linear kernel $K$ in those basis. The advantage of this approach is that the leave-one-out error can be computed as in Equation (17). Let $\mathbf{y}' = (y'_1, \ldots, y'_\ell)$ be the outputs of the unlabelled set, and $\mathbf{y}^0$ the outputs assigned to the same set by means of standard ridge regression on the training set. [9] propose to compute $\mathbf{y}'$ as the minimizer of

$$Loo(\mathbf{y}') + \gamma \|\mathbf{y}' - \mathbf{y}^0\|^2$$

The second term serves as a regularization term which tends to favor solutions close to the standard ridge regression one. [9] validates this method on two datasets showing an improvement w.r.t. standard ridge regression.

We remark that a related but different problem is that of learning from partially labelled data. In this case the training set is formed of both labelled and unlabelled data and the goal is to compute a function which does well on average. Such a system would be very useful in real applications where labelled data are an expensive commodity. A possibility would be to compute the outputs of the unlabelled points as before and then to retrain on the extended training set.

### 5.2 Feature selection and rescaling

In feature selection and rescaling, an additional set of parameters $\sigma$ is introduced which multiplies the coordinate of $x$ (we assume here $\mathcal{X} = \mathbb{R}^n$) and the goal is to compute the optimal value of these parameters. In feature selection $\sigma \in \{0, 1\}^n$, while in feature rescaling $\sigma \in \mathbb{R}^n$. We focus on the second problem. This also serves as an intermediate step to solve the first problem, since we can select the features whose computed scaling parameters are larger than a threshold.

[8] discusses feature rescaling with SVMs for binary classification. Their algorithm consists of (usually) few iterations with a stopping criteria. First the SVM is trained on the initial data. Then, the scaling parameters are updated by performing a gradient step so that leave-one-out error decreases. These two steps are then repeated till a minimum of the leave-one-out error is reached. An interesting finding was that rather simplified leave-one-out bounds are

sufficient for computing a good solution. In particular the bounds in Theorem 4.2 can be further upper bounded by $R^2\|f\|_K^2$, where $R^2$ is the maximum value of $K(x,x)$ among the set of support vectors. This has also the advantage that $R^2\|f\|_K^2$ can be differentiated exactly – see [8] for more details. [47] contains more experiments on the feature selection problem. [21] discusses a similar feature selection method which is applied to image classification problems.

Note that the leave-one-out bounds may be also useful to adapt/build a kernel function in a way similar to the feature selection problem. For example the kernel function could be a convex combination of some known kernel functions. Then, the coefficients in the convex combination could be computed by means of the algorithm above, with an additional positivity constraint.

## 6 Discussion

We have reviewed existing results on the leave-one-out error in Machine Learning. In particular we discussed attempts aimed at relating leave-one-out error to generalization error. These attempts may or may not explain the success of leave-one-out error in practical problems, but we hope at least they will motivate further studies on this interesting topic.

### *6.1 Sensitivity analysis, stability, and learning*

Leave-one-out error and stability are a “specific” instance of the more general problem of studying how the solution of learning algorithm changes in response to some perturbations of its parameters.

The leave-one-out perturbation consists in removing one point from the training set. The results in Section 3 indicate that there is a general relation between the stability of a learning algorithm and its generalization error. Other kind of perturbations may be important to better understand the properties of the learning algorithm, both statistical and numerical. [34] studies how the SVM solution changes when the regularization parameter is modified but it would be also interesting to know the sensitivity of the solution w.r.t. modification of the outputs of the training points, kernel parameters, etc.

### *6.2 Open problems*

We conclude with few open questions.

**Question 1 (consistency of leave-one-out error)** We have seen that stability considerations provide, for a fixed algorithm, sufficient conditions for the leave-one-out error to tend to the generalization error. What are the general necessary conditions? The answer would give interesting insights into which properties of an algorithm are required for its leave-one-out estimate to be close to its generalization error.

**Question 2 (empirical stability)** Can we use leave-one-out error to say something practical about stability? In particular, is the difference between leave-one-out error and training error a good estimator of the “stability” of a learning algorithm? An answer to this question would also help to apply the stability bounds to other learning algorithms such as neural networks and decision trees where it seems difficult to compute hypothesis stability.

**Question 3 (stability and invariances)** Is there a way to incorporate prior knowledge via stability? For instance, would it help to add virtual inputs and to constraint the algorithm to be very stable on these new inputs, e.g. when one point is removed from the training set, the algorithm should give the same output on the virtual samples.

*Acknowledgements:* Alessio Ceroni read the paper and made useful remarks.

## References

[1] H. Akaike. Information theory and an extension of the maximum likelihood principle. In B.N. Petrov and F. Csaki, editors, *2nd Int. Symp. on Inform. Theory*, pages 267–281. Akademia Kiado, Budapest, 1973.

[2] S. Andonova, A. Elisseeff, T. Evgeniou, and M. Pontil. A simple algorithm for learning stable machines. In *15-th European Conference on Artifi cial Intelligence*, 2002.

[3] N. Aronszajn. Theory of reproducing kernels. *Trans. Amer. Math. Soc.*, 686:337–404, 1950.

[4] B. Boser, I. Guyon, and V. Vapnik. A training algorithm for optimal margin classifi ers. In *Fifth Annual Workshop on Computational Learning Theory*, pages 144–152, Pittsburgh, 1992. ACM.

[5] O. Bousquet and A. Elisseeff. Stability and generalization. *J. of Machine Learning Res.*, 2:499–526, 2002.

[6] L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[7] O. Chapelle and V. Vapnik. Model selection for support vector machines. In T.K. Leen S.A. Solla and K.-R. Muller, editors, *Advances in Neural Information Processing Systems*. MIT Press, 2000.

[8] O. Chapelle, V. Vapnik, O. Bousquet, and S. Mukherjee. Choosing multiple parameters for support vector machines. *Machine Learning*, 46(1-3):131–159, 2002.

[9] O. Chapelle, V. Vapnik, and J. Weston. Transductive inference for estimating values of functions. In *NIPS-12*, 2000.

[10] T.M. Cover. Learning in pattern recognition. In S. Watanabe, editor, *Methodologies of Pattern Recognition*, pages 111–132. Academic Press, 1969.

[11] N. Cristianini and J. Shawe-Taylor. *Introduction to Support Vector Machines*. Cambridge University Press, 2000.

[12] F. Cucker and S. Smale. On the mathematical foundations of learning. *Bull. Amer. Math. Soc.*, 39(1):1–49, 2002.

[13] L.P. Devroye and T.J. Wagner. Nonparametric discrimination and density estimation. Technical Report 183, Information Systems Research Laboratory, University of Texas, Austin, 1976.

[14] L.P. Devroye and T.J. Wagner. Distribution-free inequalities for the deleted and holdout error estimates. *IEEE Trans. Inform. Theory*, 25(2):202–207, 1979.

[15] L.P. Devroye and T.J. Wagner. Distribution-free performance bounds for potential function rules. *IEEE Trans. Inform. Theory*, 25(5):601–604, 1979.

[16] A. Elisseeff, T. Evgeniou, and M. Pontil. Stability of randomized algorithms with an application to bootstrap methods. Preprint, 2002.

[17] T. Evgeniou, M. Pontil, and A. Elisseeff. Leave one out error, stability, and generalization of voting combinations of classifi ers. *Machine Learning*, 2002. To appear.

[18] T. Evgeniou, M. Pontil, and T. Poggio. Regularization networks and support vector machines. Advances in Computational Mathematics, 13, pp.1-50, 2000.

[19] C.H. FitzGerald, C. A. Micchelli, and A. Pinkus. Functions that preserves families of positive defi nite functions. *Linear Algebra and its Appl.*, 221:83–102, 1995.

[20] C. Goutte. Note on free lunches and cross-validation. *Neural Computation*, 9(6):1245–1249, 1997.

[21] B. Heisele, T. Poggio, and M. Pontil. Face detection in still gray images. AI-Memo 1687, MIT, 2000.

[22] A.E. Hoerl and R. Kennard. Ridge regression: Biased estimation for nonorthogonal problmes. *Technometrics*, (12):55–67, 1970.

[23] S.B. Holden. Pac-like upper bounds for the sample complexity of leave-one-out crossvalidation. In *Ninth Annual Conference on Computational Learning Theory*, pages 41–50. ACM Press, 1996.

[24] T. Jaakkola and D. Haussler. Probabilistic kernel regression models. In *Proc. of Neural Information Processing Conference*, 1998.

[25] M. Kearns and D. Ron. Algorithmic stability and sanity check bounds for leave-one-out cross validation bounds. *Neural Computation*, 11(6):1427–1453, 1999.

[26] R. Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *IJCAI*, pages 1137–1145, 1995.

[27] S. Kulkarni, G. Lugosi, and S. Venkatesh. Learning pattern classifi cation—a survey. *1948–1998 Special Commemorative Issue of IEEE Transactions on Information Theory*, 44:2178–2206, 1998.

[28] S. Kutin and P. Niyogi. Almost-everywhere algorithmic stability and generalization error, 2002. Technical report TR-2002-03, University of Chicago.

[29] P.A. Lachenbruch. An almost unbiased method for the probability of misclassifi cation in discriminant analysis. *Biometrics*, 23:639–645, 1967.

[30] A. Luntz and V. Brailovsky. On estimation of characters obtained in statistical procedure of recognition (in russian). *Technicheskaya Kibernetica*, 3, 1969.

[31] D. MacKay. *Bayesian Methods for Adaptive Models*. PhD thesis, 1992.

[32] C.L. Mallows. Some comments on Cp. *Technometrics*, 15(4):661–675, 1973.

[33] T. Poggio and F. Girosi. Regularization algorithms for learning that are equivalent to multilayer networks. *Science*, 247:978–982, 1990.

[34] M. Pontil and A. Verri. Properties of support vector machines. *Neural Computation*, (10):955–974, 1998.

[35] M.H. Quenouille. Approximate tests of correlation in time-series. *J. Roy. Statist. Soc. B*, 11:68–84, 1949.

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

[37] J. Rissanen. Modeling by shortest data description. *Automatica*, 14:465–471, 1978.

[38] W.H. Rogers and T.J. Wagner. A fi nite sample distributio-free preformance bound for local discrimination rule. *Annals of Statistics*, 6:506–514, 1978.

[39] B. Scholkopf and A. Smola. *Learning with Kernels*. The MIT Press, Cambridge, MA, 2002.

[40] D. Schuurmans. A new metric-based approach to model selection. In *AAAI/IAAI*, pages 552–558, 1997.

[41] J. Shao. Linear model selection by cross-validation. *J. Amer. Statist. Assoc.*, 88:486–494, 1993.

[42] M. Stone. Cross-validatory choice and assessment of statistical predictions (with discussion). *Journal of the Royal Statistical Society B*, 36:111–147, 1974.

[43] J.W. Tukey. Bias and confi dence in not-quite large samples. *Annals of Math. Stat.*, 29, 1958.

[44] V.N. Vapnik. *Estimation of Dependencies Based on Empirical Data*. Springer, N.Y., 1982.

[45] V.N. Vapnik. *Statistical Learning Theory*. John Wiley & Sons, N.Y., 1998.

[46] G. Wahba. Splines models for observational data. *Series in Applied Mathematics, SIAM, Philadelphia*, 59, 1990.

[47] J. Weston, S. Mukherjee, O. Chapelle, M. Pontil, T. Poggio, and V. Vapnik. Feature selection for support vector machines. In *NIPS-13*, 2001.

[48] D.H. Wolpert and W.G. Macready. No free lunch theorems for search. Technical Report SFI-TR-95-02-010, Santa Fe, NM, 1995.

[49] T. Zhang. A leave-one-out cross-validation bound for kernel methods with applications in learning. In *14th Annual Conference on Computational Learning Theory*, pages 427–443, 2001.