# Classification Methods and Inductive Learning Rules: What We May Learn From Theory

Cesare Alippi, *Fellow, IEEE,* and Pietro Braione

***Abstract*—*Inductive* learning methods allow the system designer to infer a model of the relevant phenomena of an unknown process by extracting information from experimental data. A wide range of inductive learning methods is nowadays available, potentially ensuring different levels of accuracy on different problem domains. In this critical review of theoretic results gained in the last decade, we address the problem of designing an inductive classification system with optimal accuracy when domain knowledge is limited and the number of available experiments is—possibly—small. By analyzing the formal properties of consistent learning methods and of accuracy estimators, we wish to convey to the reader the message that the common practice of aggressively pursuing error minimization with differentg training algorithms and classification families is unjustified.**

***Index Terms*—Image classification, intelligent systems, learning systems, neural networks, pattern classification.**

## I. Introduction

QUALITY expectations about modern industrial production lines are becoming more and more severe. Current research is investigating how an industrial process can be monitored and controlled in an effective way by exploiting the information produced by a stack of in-process sensors. Such an approach is potentially effective provided that, on one side, sensor data convey a sufficient amount of information and that, on the other side, a sensible monitoring and control subsystem can be designed so that sensor information can be effectively and optimally exploited. This, in turns, depends on our degree of knowledge of the process. Sometimes a formal description of the process is available in the form of a parametric model. In this case, the identification task is performed by estimating the parameters' values which minimize a suitable cost figure, accounting the loss we have or expect to have when we assume the obtained model as the true process. An important yet elusive cost figure is *risk*, which aims at measuring the difference between the true system behavior and the estimated one. In this article, we focus instead on the case, more and more common in industrial practice, where a formal process description is only partially available, or not available at all, because of limited knowledge of domain phenomena. In these cases, the system designer must *a priori* commit to a nonparametric model. An *inductive* learning method is a (non)parametric model structure, plus a criterion for tuning its degrees of freedom over experimental data produced by some unknown phenomenon.

The selection of a suitable inductive learning method is a crucial phase in any system design methodology. Today, the application designer can choose from many inductive learning methods, ranging from linear discriminant and probabilistic approaches, to artificial neural networks and support vector machines [4], or even from a number of integrated design methodologies [2], where several identification methods are elicited to lead from a set of raw data to a full industry-strong solution. Each approach behaves more or less well when applied to different problems in different domains.

Given these premises, it is no surprise that the following methodological issue recurrently arises both in literature and in industrial practice: by assuming a specific learning problem, and risk as the cost figure, how do we select the optimal inductive learning method among a number of candidates? In other words: how do we select the inductive learning method which, in probability, yields the most faithful model of the process? In this paper, we wish to convey to the Systems, Man and Cybernetics community the message that, when limited *a priori* information is available about the system that must be modelled, aggressively pursuing risk minimization for different learning rules and classifier families is not justified under the theoretical nor under the practical point of view, if we assume that learning method, classifier family and accuracy estimators are concretely chosen as exposed in next sections. Intuitively, the reason is twofold: on one side, we are not able to assess the accuracy of a solution with an arbitrary degree of confidence, given a limited number of experiments; on the other, there are some sufficient conditions, typically satisfied by any relevant learning method, ensuring that a learning process converges rapidly towards the optimal solution for that family of classifiers. For these reasons, we maintain that different learning methods which, in theory, may yield different approximations of the optimal solution on different classes of learning problems, can be considered equivalent in practice, under the uncertainty limits of the scenarios we are considering. This paper will focus on quality analysis applications, thus to the problem of designing binary (accept/reject) classification systems, but the results and conclusions we will expose are general enough to be comfortably adapted to any incarnation of the learning problem.

The paper is structured as follows. In Section II we formally define the learning problem, and we explain the features of a learning method ensuring that the solutions is able to produce have a high degree of accuracy. Three popular families of methods will be discussed in Section III, while in Section IV we

Manuscript received January 1, 2005; revised April 12, 2005. This paper was recommended by Associate Editor E. Trucco.

The authors are with the Dipartimento di Elettronica e Informazione, Politecnico di Milano, 20133 Milano, Italy (e-mail: cesare.alippi@polimi.it; braione@elet.polimi.it).

Digital Object Identifier 10.1109/TSMCC.2005.855508

address the problem of assessing the accuracy of the produced classifiers. Concluding remarks will be given in Section V.

## II. Inductive Learning

In this section, we follow the approach suggested by Vapnik [20], [21], and Devroye *et al*. [6] to the analysis of classification problems.

### A. Learning Problem

Let us consider a stationary random pair, $z = \langle x, y \rangle, x \in \mathbf{X} \subset \mathbb{R}^{n_\mathbf{X}}, y \in \{0, 1\}$, and a set $\mathbf{F} = \{f(x, \alpha) \,|\, \alpha \in \Lambda \subset \mathbb{R}^{n_\Lambda}\}$ of real-valued functions, called the *hypothesis space*. $x$ is the input vector of acquired data, $y$ is the classification value, $\alpha \in \Lambda$ is a controllable parameter vector and the $\alpha$-indexed functions $f(x, \alpha) : \mathbf{X} \times \Lambda \to \{0, 1\}$ are called the *hypotheses*. As an example, if $\mathbf{F}$ is a family of artificial neural networks $\alpha$ is the vector containing all the trainable parameters of the network, like weights and biases. Finally, we denote by $L(y, f(x, \alpha))$ the *loss*, or discrepancy, function, expressing the *cost*, denoted by $L$, of observing $y$ instead of $f(x, \alpha)$. We define *risk* to be the expected value of the loss, $R(\alpha) = E[L(y, f(x, \alpha))]$, as a function of $\alpha$. In this framework, the *learning problem* can be cast in an optimization problem, requiring the identification of the $\alpha_0 \in \Lambda$ minimizing the risk functional being in possess of a finite data sample $S = \{\langle x_1, y_1 \rangle, \ldots, \langle x_n, y_n \rangle\}$, of cardinality $n$. The hypothesis space represents all the available solutions, or *classifiers*: the ultimate goal of training is to find a classifier with minimum risk.

In classifier design the cost function is generally the indicator over the event $y \neq f(x, \alpha)$, which assigns a unit cost to an incorrect classification performed by the selected hypothesis. In this case, risk reduces to the *error probability* $\mathrm{Err}(\alpha) = P\{y \neq f(x, \alpha)\}$. It is very often the case that $f(x, \alpha)$ has the form $\mathrm{sign}(\psi(x, \alpha) - 1/2)$, where $\psi(x, \alpha)$ is a real-valued continuous function; the surface described by equation $\psi(x, \alpha) = 1/2$ is named the *decision boundary* of $f(x, \alpha)$. It can be proved [8], [6] that there is an optimum decision boundary for any classification problem, called the *maximum a posteriori probability* (MAP), or *Bayes*, decision boundary $f_B(x) = \mathrm{sign}(P^{y\,|\,x}(1\,|\,x) - 1/2)$, whose risk we denote with $\mathrm{Err}_B$. Intuitively, this boundary achieves optimal separation of classes by minimizing the probability of the misclassification error which, however, cannot be completely eliminated in the general case. As a consequence, no classifier can be more precise that the MAP one [6], [8], whose knowledge is associated with the knowledge of the conditional probability distribution of $y$ with respect to $x$. Assuming that this classifier belongs to some hypothesis space is an *a priori* assessment of a property of the learning problem which, usually, we do not know and does not hold. Consequently, in the general case, we are unable to state whether the chosen hypothesis space contains the Bayes classifier or not, and therefore, we must expect that $\mathrm{Err}(\alpha_0) \geq \mathrm{Err}_B$. The quantities $\mathrm{Err}_B$ and $\mathrm{Err}(\alpha_0)$ are also known as the *inherent* and *language-intrinsic error*, respectively. If we rewrite $\mathrm{Err}(\alpha)$ as

$$\mathrm{Err}(\alpha) = (\mathrm{Err}(\alpha) - \mathrm{Err}(\alpha_0)) + (\mathrm{Err}(\alpha_0) - \mathrm{Err}_B) + \mathrm{Err}_B$$

we notice that three distinct fonts of error contribute to form $\mathrm{Err}(\alpha)$:

- the inherent error $\mathrm{Err}_B$, which depends only on the learning problem itself and that, for this reason, can be improved only by improving the problem itself (e.g., the process);
- the *approximation* error $\mathrm{Err}(\alpha_0) - \mathrm{Err}_B$, which depends on how the hypothesis space is close to the process generating the data;
- the *estimation error* $\mathrm{Err}(\alpha) - \mathrm{Err}(\alpha_0)$, which depends on the ability of the learning process to pick a hypothesis close to the language-intrinsic one.

### B. Learning Methods

An *inductive (learning) principle* defines how data are used to select a classifier from a given hypothesis space. A *learning algorithm* is an implementation of an inductive principle. Inductive principles establish a correlation between $\alpha$ and $S$ by defining a function $\alpha = \alpha(S)$, or more generally a probability distribution $P^{\alpha|S}$. In the former case we will name the inductive principle *deterministic*, *stochastic* in the latter. Be it deterministic or stochastic, an inductive principle determines a probability distribution over the post-training value of $\alpha$. Hence, we will make explicit the dependence of $\alpha, f(\alpha, x)$ and $\mathrm{Err}(\alpha)$ on the sample size $n$ defining, given an inductive principle, an *(inductive) rule* as a random sequence $\{f(x, \alpha_n) \,|\, n \geq 1\}$, where $n$ is, as usual, the sample size. We will name each function in the rule as $f_n$, and their probability of error as $\mathrm{Err}_n$.

The effectiveness of a learning method is determined by the speed with which it converges towards the optimum classifier, be it the language-intrinsic or the Bayes one as the number $n$ of available experiments increases. A rule is *consistent* if $\mathrm{Err}_n$ converges, as $n \to \infty$, to the optimum value $\mathrm{Err}_{\mathrm{opt}}$ (we may assume $\mathrm{Err}(\alpha_0)$ or $\mathrm{Err}_B$ as our optimum of choice, yielding a "relative" and an "absolute" definition of consistency respectively). We say to have *good generalization ability* (i.e., *fast convergence speed*) if a bound $g(n)$ exists for $P\{\mathrm{Err}_n > \mathrm{Err}_{\mathrm{opt}} + \epsilon\}$, such that $\lim_{n\to\infty} \exp(n) g(n) < \infty$ (*universally*, if convergence is independent on $P^{\langle x,y \rangle}$). We will name the function $g(n)$, the *convergence speed bound* of the rule.

Consistency is the first property we would expect from a learning method: the training phase of a consistent methods improves as a larger training dataset is available, while this is not necessarily true for an inconsistent one. A necessary condition for a learning method to be universally consistent is that its hypothesis space contains enough functions to approximate the decision boundary with an arbitrary degree of accuracy—intuitively, the hypothesis space is arbitrarily expressive. However, a classifier is always selected by means of a finite sample. This implies that the reliability of the selection procedure decreases with the size of the hypothesis space. Universal consistency with fast convergence speed is achieved by balancing the estimation and the approximation error for every value of $n$, in a way that neither component outgrows the other one. This is usually done by defining a hierarchy of subspaces, $\{\mathbf{F}_p \,|\, p \in \mathbb{N}\}$, with $\mathbf{F}_p \subseteq \mathbf{F}_{p+1}$, whose limit $\mathbf{F}^* = \bigcup_{p\in\mathbb{N}} \mathbf{F}_p$ contains enough

functions to approximate every decision boundary with arbitrary precision. If we consider as an example the space of the neural networks with one hidden layer of neurons, $\mathbf{F}_p$ usually contains all the networks with up to $p$ hidden units. By increasing $p$ we span the hierarchy and obtain more expressive classifiers at the expense of a larger model complexity. For every value of $n$ there is an optimum value of $p$, which we indicate by $p(n)$, allowing in probability the best tradeoff between the approximation and estimation error. Such optimum depends, in general, on the actual data distribution, and cannot therefore be easily determined. On the other hand, universal consistency is an asymptotic property, implying that it can be achieved when $p(n)$ has suitable asymptotic features, depending only on the hierarchy of subspaces and not on the learning problem. In general, suitable parameter selection criteria yield universally consistent rules, both towards Bayes and towards the language-intrinsic classifier. This is true for nearest neighbor classifiers, feed-forward artificial neural networks, and almost all effective classifiers used in the related literature. A comprehensive review of classification paradigms with their universally consistent rules is presented in Devroye, Györfi, and Lugosi [6]. An example of universally consistent rule for the $k$NN classifier will be given in Section III.

### C. Comments

The theory allows us to understand the intrinsic limits to learning. A learning problem is affected by three sources of error; of these, the inherent one is determined by the nature itself of the problem, and for this reason it cannot be improved by learning. The remaining error sources, i.e., the approximation and the estimation error, are the true object of any learning procedure. Asymptotically, they can both be controlled if the learning method has some basic consistency features (which most practical methods have). But when the available dataset is small, the dominating component of the learning error is determined, if the method is consistent, by the approximation error, i.e., by how well the hypotheses in the space can approximate the Bayes decision boundary. In other words, the error is mainly determined by the choice of the classifier rather than the training procedure. As a consequence, in absence of *a priori* information we have no basis to prefer a consistent learning method to another one.

## III. Classifier Families

One of the most important choices the designer of a learning system must face concerns the learning method to adopt. What clearly emerges from an analysis of the literature is that several combinations of structured hypothesis spaces and inductive principles have been experimented, and for many of them some statistical properties have been determined. In this article we consider three different classifier families: $k$NN classifiers, feed-forward artificial neural networks and support vector machines. Their properties have been studied in three different contexts—classical statistical estimation for $k$NN, Vapnik's Statistical Learning Theory for SVM and the theory of function approximation for artificial neural networks—and thus they are meaningful examples of different approaches to the learning problem.

*$k$NN:* $k$NN classifiers assign the class to a point in the input space by considering the $k$ patterns in $S$ that are closest to the point according to some metric, and adopting some tie-breaking procedure when $k$ is even. The parameters characterizing the family are the number $k$ of neighbours and the metric ($L_2$ norm is the common choice). It can be proved that the error probability of 1NN classifiers, as $n \to \infty$, is bounded by twice the inherent error [8]. For this reason, when the inherent error is expected to be small and many patterns are available, the 1NN classifier is quite a common solution. Should we consider a $k$ higher than 1? With small sample sizes 1NN yields a poor density estimate, excessively sensitive to local glitches, than a classifier with a slightly higher $k$. Conversely, a too high $k$ yields an oversimplified model until error becomes as bad as the lowest of the *a priori* probabilities when $k = n$. Stone [18] proved that all $k$NN rules where $k(n)$ grows less than linearly with $n$ are universally consistent.

The major drawbacks of the $k$NN paradigm are its computational load and its space occupation [6]. The basic $k$NN implementation has a $O(n^2)$ space occupation and a $O(knm)$ computational complexity, where $m$ accounts for the cost of one distance computation. Implementations exist with reduced computational complexity up to, with $k$ and $m$ fixed, $O(n^{1/m})$, while edited and condensed nearest neighbour paradigms, by storing only a subset of the available patterns, also reduce space occupation and improve class separability.

*Neural Networks:* Feedforward neural networks are grounded in the theory of continuous function approximation [11]. A fundamental theorem from Cybenko [5] states that the class of the networks with one hidden layer of sigmoid neurons allows to approximate every continuous function uniformly over a bounded set, with an arbitrary degree of precision. It can be proved that this property ensures sufficient expressive power to build universally consistent rules [7]. It is therefore no surprise that practitioners have focused their attention on this class of neural networks to solve several applications. The space occupation of a neural network linearly depends on the number of hidden neurons. When we apply Makhoul's remarks on the shatter capacity of the multilayer perceptron we may obtain an approximate asymptotic upper bound of $O(i \cdot n^{1/i})$, where $i$ is the number of inputs [1], [12]. The main issue of neural networks commonly used in practice is perhaps the lack of a computationally efficient method to minimize their empirical error. The algorithms used in practice are usually suboptimal gradient descent heuristics, minimizing mean-square error rather than empirical error itself. Such algorithms often get stuck in local optima, thus hindering the effectiveness of the training phase. Faragó and Lugosi recently proposed an algorithm that finds the network minimizing the empirical error with a complexity exponential both in the number of hidden neurons and in the number of inputs, thus useless in all the practical cases [7].

*Support Vector Machines:* Support vector machines (SVMs) [3], [4], [5] [20], [21] are linear classifiers in high-dimensional spaces; the idea behind the SVM paradigm is that, if we define a (usually nonlinear) map from the input space to a *feature space* $\mathbf{H}$, the sample $S$ can be separated by a hyperplane in $\mathbf{H}$ even if a separating hyperplane does not exist in $\mathbf{X}$. In the

subset of all the $S$-separating hyperplanes we choose the *optimal separating* hyperplane as the one which maximizes *margin*. Margin is defined as the distance between the hyperplane and the closest between the positive and the negative cluster, $S^+ = \{\langle x, 1\rangle \in S\}$ and $S^- = \{\langle x, 0\rangle \in S\}$.

Determining the optimal separating hyperplane is a quadratic programming problem. However, mapping the input space to a feature space may make this problem computationally hard. At the purpose SVMs exploit Mercer's theory of kernel functions. A *kernel* function is a symmetric function $\ker : \mathbf{X} \times \mathbf{X} \to \mathbb{R}^+$ having the property of being the inner product $\phi(x_1) \cdot \phi(x_2)$ for some map $\phi : \mathbf{X} \to \mathbf{H}$: its knowledge is sufficient to determine the hyperplane in the feature space separating the patterns. Polynomial functions and Gaussian RBF functions are example of kernel functions, and are indeed very popular. Moreover, the problem is relaxed to allow a hyperplane to be found when the sample cannot be perfectly separated even in the feature space. This is the case of $C$-SVMs, which modify the cost function by adding a term which depends on the number of patterns falling inside the margin set. A $C$ hyperparameter can be used to determine a tradeoff between the two terms of the cost function: a higher $C$ biases the learning process towards a smaller classification error over the training data, rather than on a bigger margin.

The resulting hyperplane in the feature space has form

$$f(x, \alpha) = \operatorname{sign}\left(\ker\left(x, \sum_i \lambda_i x_i\right) + b\right)$$
$$= \operatorname{sign}\left(\sum_i \lambda_i \ker(x, x_i) + b\right)$$

with $\langle x_i, y_i\rangle \in S$ and $i = 1 \ldots n; \lambda_i$ are suitable multipliers. We will call this solution the *generalized optimal separating hyperplane*. We notice that $f(x, \alpha)$ has, indeed, the form of a linear discriminant function where the scalar product of $x$ for the weight vector $w$ has been replaced by the corresponding scalar product in the feature space, $\ker(x, w)$. The generalized optimal separating hyperplane is determined only by the patterns for which $\lambda_i \neq 0$, which are called the *support vectors*. In the case of $C$-SVMs, the patterns which are associated to a $\lambda$ such that $0 < |\lambda| < C$ lie on the margin, while patterns for which $|\lambda| = C$ lie inside the margin, possibly misclassified. This usually means that the support vectors are a small portion of the training data. In a sense, SVMs resemble condensed nearest neighbour classifiers, where the decision surface is calculated as a function of the few patterns that are close to it. A hyperplane in the feature space is the image of a nonlinear function in the input space, whose shape depends on the kernel function.

An attractive feature of SVMs is the fact that a quadratic programming problem is convex, and therefore, every local solution is also global (unique if the Hessian is positive definite). The training phase of a SVM is computationally intensive when $n$ is high, but is amenable to parallelization [10], [16], [17], [19] and always yields the optimal solution. A SVM is structurally similar to a feedforward neural networks with one layer of hidden neurons, where a hidden neuron performs the kernel product of the input with a support vectors. Since support vectors are usually a small portion of the training data, the spatial complexity of a SVM is usually low.

The $C$-SVM paradigm exposes as parameters $C$ and the parameters determining the shape of the kernel (e.g., spread for the Gaussian kernel). It is not immediately clear how the parameters are related to the complexity of the resulting machine; here we note that, by increasing the degree of a polynomial kernel, the dimension of the minimal embedding space grows, and that by reducing the spread of a RBF kernel a higher shattering capacity is achieved as closer points can be discriminated. For its very definition, a higher $C$ will penalize training error more than a low margin, and thus will bias the training phase towards a higher model complexity. Authors argue that the training procedure itself of SVMs is assimilable to the Structural Risk Minimization learning principle, introduced by Vapnik [21], therefore, ensuring fast convergence of the training error [22].

### A. Comments

All the families presented in this section allow to generate consistent rules, and this is the case with many other families we do not consider in this article. Consistent rules converge quickly towards the optimum classifier: when the sample is large, thus, consistent rules have comparable accuracy. What really differentiates a learning method from another one is how it behaves when the training sample is small and the approximation error is prevalent. This piece of information can, in principle, be exploited when sufficient *a priori* knowledge over the learning problem allows to choose the family ensuring optimal approximation ability. When such knowledge is not available, as it often happens in practice, we have no sound basis to prefer a method to another one, at least for what concerns accuracy.

## IV. Accuracy Estimation and Comparison

In the previous sections, we remarked how one of the most valuable features of a classifier is how accurately it is able to separate "positive" from "negative" cases, and how this feature is quantified as risk. In the previous sections we have seen how theory allows us to understand under which conditions we may expect that, the more patterns we use to infer a classifier, the better the obtained classifier is. This is a strong motivation for using all the available patterns to infer the classifier. However, the data used to infer a classifier cannot be used to estimate its accuracy without introducing a bias in the estimation itself. In other words, a tradeoff must be assessed between the confidence in a classifier's accuracy, and the confidence in the estimated value of such feature. In this section, we will show that such issue strongly affects the task of estimating the risk of a classifier obtained by means of an inductive learning method, especially when minimization of some error estimator is used as an inductive principle.

### A. Estimators

Many estimators for the whole-sample true error have been proposed in literature, mostly based on *resampling*. Resampling

estimators build one or more pairs of samples $\langle S_E, S_D \rangle$ from the available experimental dataset $S$. The $S_D$ sample is used to induced a classifier whose accuracy is estimated by applying it over $S_E$. A component in the bias and variance of the estimate is determined by the resampling operations themselves. We will consider the following methods.

- *Apparent Error Rate* (AER), or *resubstitution*: The whole sample $S$ is used both to infer a classifier and to estimate its error probability.
- *Sample Partitioning* (SP): $S_D$ and $S_E$ are obtained by randomly splitting $S$ in two disjoint subsets. $S_D$ is used to induce a classifier and $S_E$ to estimate its accuracy.
- *Leaving-One-Out* (LOO): $S_E$ contains one pattern in $S$, and $S_D$ contains the remaining $n-1$ patterns. The procedure is iterated $n$ times by holding out each pattern in $S$, and the resulting $n$ estimates are averaged.
- *$w$-fold Crossvalidation* ($w$CV): $S$ is randomly split into $w$ disjoint subsets of equal size. For each subset the remaining $w-1$ subsets are merged to form $S_D$ and the reserved subset is used as $S_E$. The resulting $w$ estimates are averaged. This procedure can be iterated and the results averaged when $w \ll n$ in order to reduce the random resampling variance. This estimate is a generalization of LOO.

The AER estimate is, in most cases, strongly optimistically biased since most learning methods use AER (the training error) to tune the classifier's parameters. Bounds for the AER estimate can be given in many cases, but this estimate is normally too biased to be useful in practice.

The properties of the SP estimate are related to the fact that, by randomly splitting a sample $S$ in two subsamples $S_D$ and $S_E$, two mutually independent identically distributed (i.i.d.) datasets are obtained. If the cardinality of $S_D$ is $d$, the SP estimator yields an unbiased estimate of $\mathrm{Err}_d$, which can be considered as a biased estimate of the full sample error $\mathrm{Err}_n$. The holdout bias, which makes the SP estimate a pessimistic one for universally consistent rules, can be reduced by raising the $d/n$ ratio. The main shortcoming of the SP estimator is, precisely, the difficulty of correctly balancing the holdout bias with the estimator variance when $n$ is low, as bias decreases with $d/n$ while variance increases. According to Higleyman's analysis [9] the $d/n$ ratio must be higher than 0.5, but it is common practice to lower this ratio up to 25% to reduce holdout bias [13]. SP is an alternative when $n$ is sufficiently high to have both low holdout bias and low variance.

The $w$CV and LOO estimates are strictly related with holdout. LOO can, in fact, be considered as the maximally iterated minimum holdout bias estimate, where iteration is used to reduce the variance of estimating over a singleton data set. It is based on the assumption that the learning method is *stable* under the perturbation caused by deleting one pattern from the sample $S$, i.e., a classifier structurally similar to that obtainable from the whole sample is returned when the inductive principle is applied to the perturbed dataset. The LOO method yields an unbiased estimate of $\mathrm{Err}_{n-1}$, which is close to $\mathrm{Err}_n$ except for very low values of $n$. The main drawbacks of the LOO method are its high variance, which is the counterpart of its low bias, and the high computational load it requires as $n$ training phases must be performed.

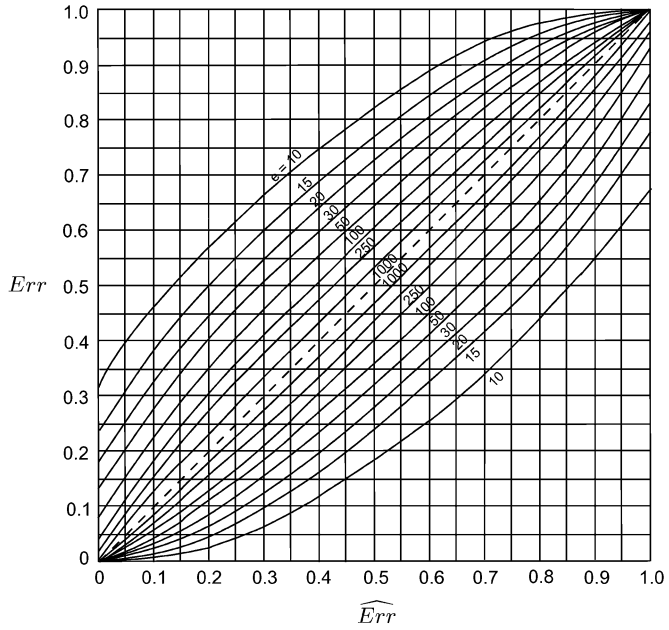


Fig. 1. Binomial law confidence interval for $\eta = 0.05$. The interval is determined by intersecting the curves labeled with the size of the estimation sample $S_E, e \stackrel{\text{def}}{=} n - d$, with the vertical line at the abscissa corresponding to the estimated error $\widehat{\mathrm{Err}}$. The upper and the lower bounds of the interval are read on the $Y$ axis (adapted from Higleyman [9]).

### B. Confidence and Accuracy Comparison

The problem of determining a parameter's confidence interval or a rejection region for a given hypothesis can be solved if we have an estimator for the parameter whose distribution is known. The number of classification errors over a sample not used to induce the classifier is a binomially distributed random variable [8]. But usually patterns do not come for free and reliable error estimates can be obtained only through resampling.

If resamples were independent we might state that, by applying the law of large numbers, resampling estimators are approximately normally distributed. Unfortunately, this is not the case, and we cannot plug in the classical confidence intervals and tests the error estimates obtained via resampling without the important caveat that, in general, we will not obtain the textbook significance. The methods we will introduce in this subsection are all affected by this issue.

*1) Confidence Intervals:* In Fig. 1 a nomograph of the 0.05-level confidence interval for binomially distributed error count is reported—i.e., in the assuption that $S_E$ is obtained by means of sample partitioning. Let us now consider the general case, and denote with $\widehat{\mathrm{Err}}$ the estimated error obtained by means of some error estimator, e.g., LOO. A textbook formula yielding an approximated $\eta$-level binomial confidence interval is

$$\widehat{\mathrm{Err}} \pm \left( \frac{0.5}{\hat{e}} + \phi_{1-\eta/2}\, s_1 \right) \tag{1}$$

where $\phi$ is the cumulative standard normal distribution, $\hat{e}$ is the equivalent test sample size (for LOO it is $\hat{e} = n$, for SP it is of course $\hat{e} = e \stackrel{\text{def}}{=} n - d$), and $s_1 \stackrel{\text{def}}{=} \sqrt{\widehat{\text{Err}}(1 - \widehat{\text{Err}})/\hat{e}}$.

According to Martin and Hirschberg [14], the recommended confidence interval for small samples is obtained by assessing the Jeffreys' *a priori* distribution over $\text{Err}(\alpha)$

$$\left(\widehat{\text{Err}} + \frac{a}{2\hat{e}}\right) \pm \phi_{1-\eta/2}\, s_2 \tag{2}$$

where $a \stackrel{\text{def}}{=} \sqrt{2}(\hat{e} - 2\hat{e}_{mc})\phi_{1-\eta/2}/(\hat{e} + 3)$, $s_2 \stackrel{\text{def}}{=} \sqrt{\widehat{\text{Err}}(1 - \widehat{\text{Err}})/(\hat{e} + 2.5)}$, and $\hat{e}_{mc}$ is the number of pattern in the estimation subsample which are incorrectly classified by the classifier.

*2) Performance Comparison:* Given two classifiers we can roughly compare them against classification performance by directly comparing their estimated error. The main drawback of this method is that a difference in the estimates can be more or less meaningful according to the accuracy of the estimates themselves and to their mutual correlation. This does not prevent learning methods using direct comparison of estimated performances to be universally consistent, given that the simple consistency requirements exposed formerly are fulfilled, but asymptotic results do not hold in general when the comparison is performed over finite validation data. As a consequence, the procedure exposed above offers no guarantee of viability, and we should more correctly perform a comparison through a hypothesis test with a given level of confidence.

Let us suppose we are given two classifiers $f_1, f_2$, and a finite sample $S$. We consider the problem of determining whether $f_1$ and $f_2$ differ for their accuracy or not (the null hypothesis being the latter one). By assuming that both classifiers do not depend on $S$ (i.e., the patterns in $S$ have not been used one or both the classifiers), we can exploit the fact that the error count estimates over $S$ for the two classifiers, $\widehat{\text{Err}}_1$ and $\widehat{\text{Err}}_2$ respectively, are binomially distributed. In this case, if $X_i \sim \mathcal{N}(\mu_i, \sigma_i^2)$, $i = 1, 2$, then $X_1 + X_2 \sim \mathcal{N}(\mu_1 + \mu_2, \sigma_1^2 + \sigma_2^2)$. Under the conditions allowing the application of the normal approximation ($n > 30$ is a popular heuristics) we can use a standard Gaussian test for $\widehat{\text{Err}}_1 - \widehat{\text{Err}}_2$, and derive an $\eta$-significance reject region [14]

$$|\widehat{\text{Err}}_1 - \widehat{\text{Err}}_2| > \frac{s_3}{\sqrt{n}}\, \phi_{1-\eta/2} \tag{3}$$

where

$$s_3^2 = \left(\frac{1}{e_{mc,1}} + \frac{1}{e_{mc,2}}\right) \widehat{e}(1 - \widehat{e}) \quad \widehat{e} \stackrel{\text{def}}{=} \frac{e_{mc,1} + e_{mc,2}}{e_1 + e_2} \tag{4}$$

the $e_i$'s are, as usual, the size of the subsamples used to calculate the error estimate $\widehat{\text{Err}}_i$, and $e_{mc,i}$ is the number of estimation patterns incorrectly classified by the classifier $i$. The approximation is good when $e_1$ and $e_2$ are comparable. This textbook test is sufficiently accurate for most practical purposes when compared to the exact binomial test.

When the classifiers are inferred from the same data a paired test is more appropriate. Plugging the $w$CV estimator in the classical paired Student's t test we obtain the following approximation:

$$\left|\widehat{\text{Err}}_1^{w\text{CV}} - \widehat{\text{Err}}_2^{w\text{CV}}\right| > \frac{s_4}{\sqrt{v}} t_{1-\eta/2}(v - 1) \tag{5}$$

where $v \stackrel{\text{def}}{=} n/w$

$$s_4^2 \stackrel{\text{def}}{=} \frac{1}{v}\sum_{i=1}^{v}\left(\left(\widehat{\text{Err}}_1^{i\text{FD}} - \widehat{\text{Err}}_2^{i\text{FD}}\right) - \left(\widehat{\text{Err}}_1^{w\text{CV}} - \widehat{\text{Err}}_2^{w\text{CV}}\right)\right)^2 \tag{6}$$

and $\widehat{\text{Err}}^{i\text{FD}}$ is the error estimate produced by the $i$th fold of the cross-validation procedure. This approximation is, in fact, a very rough way to compare how different inference methods behave over the same data. The roughness depends on the fact that it implicitly assumes the $\widehat{\text{Err}}^{i\text{FD}}$ to be independent, and we know that this is not the case. In [14], the authors remark how estimation of variance performed through paired crossvalidation is biased and highly variable, and reject this test as misleading. Moreover, the authors express their belief that no reliable hypothesis test really exists in this case, nor our survey found better methods than the ones here reported.

### C. Comments

Determining how an error estimator is correlated to the estimated parameter (the classifier's error) is especially important when the sample is small. Unfortunately, there are few theoretical guarantees. A guideline is given by the stability principle, requiring that the classifier(s) used to estimate accuracy are structurally similar to the classifier(s) whose accuracy must be estimated. Anyway, the confidence over the resulting estimated values is usually very low, even if we optimistically assume the ideal confidence (see, e.g., Higleyman [9], also reported in Fukunaga [8]). Using these estimates to compare different classifiers against accuracy would be misleading to say the less, and in fact the problem of determining a sensible hypothesis test under the conditions we are outlining is still open.

## V. Conclusion

In this paper, we showed how a statistical approach to the problem of learning from data highlights the intrinsic limits the accuracy of the classifier obtained with a limited amount of information over a given problem domain suffers. We cannot effectively learn from data without any explicit or implicit *a priori* commitment over how the phenomena we observe are to be interpreted and modeled. Such a commitment allows us to prefer a formal model of the phenomena to another one, but at the same time it limits the validity of our result, which hold only as far as our assumption hold. The availability of new information allows us to loosen our *a priori* in the long run. In no case we can improve over the intrinsic limit given by our assumptions.

Another relevant issue is the mutual dependence of learning and performance estimation. Consistent learning methods usually converge quickly to the language-intrinsic error. This only depends on the hypothesis space, e.g., on the class of functions that our classification method is able to implement. What

we are not able to know is how different methods, i.e., different class of functions, are more or less suitable to approximate the Bayes decision boundary. Even if a method were sensibly better than another one, it would not be possible to observe this through the accuracy estimates, because of their low reliability as the sample is also used to induce the classifier. It would indeed be possible if the sample were sufficiently large, but in this case it would be possible to relax our *a priori*, choose a more complex (i.e., more expressive) family of learning functions, more expressive or more appropriate to the specific learning problem, and obtain a better classifier.

A third remark concerns the choice of the complexity of a family of learning functions, given the size of the training set. An excessively small family (e.g., a neural network with a low number of hidden neurons) usually implies a high approximation error, as the language-intrinsic classifier is far from the Bayes one. Conversely, an excessively complex family (e.g., a neural network with an excessively high number of hidden neurons) has enough espressivity to reduce the approximation error, but the training algorithm will not be able to control the estimation error, i.e., to select the best classifier in it. While acknowledging that the purpose of every learning methodology is striving towards a suitable tradeoff, our previous discussion suggests that these two adverse situations have different importance. In the former case, the source of error is determined by our *a priori* choice of the hypothesis space complexity, thus on the constraints we decide to impose over the learning methodology. In the latter case, the source of error is completely random and uncontrollable.

For these reasons, we maintain that the practice of aggressively minimizing the estimated error is useless, because, given that some basic consistency requirements are met, all the learning methods behave similarly when the sample is big, and when the sample is small we are unable to state effectively which behaves better. A simple model should be always preferred to a more complex one, as the cost of complexity in terms of computation time and silicon area has no payback in accuracy—on the contrary, a complex model is likely *less* accurate than a simple one with comparable error count. We do not consider this as a limitation: freed from the hassle of accuracy, the designer of a learning system may focus on the features which are directly quantifiable and comparable: cost, area occupation, time latency.

## REFERENCES


[1] C. Alippi, P. Braione, V. Piuri, and F. Scotti, "A methodological approach to multisensor classification for innovative laser material processing units," in *Proc. 18th Instrumentation and Measurement Technology Conference (IMTC)*, vol. 3, May 2001, pp. 1762–1767.

[2] C. Alippi, V. Piuri, and F. Scotti, "A high-level synthesis of embedded composite systems for environmental applications," in *Proc. IEEE International Workshop on Advanced Environmental Sensing and Monitoring Technologies*, Milan, Italy, 2001.

[3] C. J. C. Burges, "A tutorial on support vector machines for pattern recognition," *Data Min. Knowledge Discovery*, vol. 2, no. 2, pp. 121–167, 1998.

[4] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*. Cambridge, U.K.: Cambridge University Press, 2000.

[5] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Math. Contr., Signals Syst.*, vol. 2, pp. 303–314, 1989.

[6] L. Devroye, L. Györfi, and G. Lugosi, *A Probabilistic Theory of Pattern Recognition*. New York: Springer, 1996.

[7] A. Faragö and G. Lugosi, "Strong universal consistency of neural network classifiers," *IEEE Trans. Inform. Theory*, vol. 39, no. 4, pp. 1146–1151, Jul. 1993.

[8] K. Fukunaga, *Introduction to Statistical Pattern Recognition*. New York: Academic, 1972.

[9] W. H. Higleyman, "The design and analysis of pattern recognition experiments," *Bell Syst. Tech. J.*, vol. 41, pp. 723–744, 1962.

[10] T. Joachims, "Making large-scale SVM learning practical," in *Advances in Kernel Methods—Support Vector Learning*, B. Schölkopf, C. J. C. Burges, and A. J. Smola, Eds. Cambridge, MA: MIT Press, 1998.

[11] A. N. Kolmogorov, "On the representation of continuous functions of many variables by superposition of continuous functions of one variable and addition," *Doklady Akademii Nauk USSR*, vol. 114, pp. 953–956, 1957.

[12] J. Makhoul, "Pattern recognition properties of neural networks," in *Proc. IEEE Workshop on Neural Networks for Signal Processing*, 1991, pp. 173–187.

[13] J. K. Martin and D. S. Hirschberg, "Small sample statistics for classification error rates, I: Error rate measurements," Tech. Rep. 96-21, ICS Dept. Univ. California Irvine, 1996.

[14] —, "Small sample statistics for classification error rates, II: Confidence intervals and significance," Tech. Rep. 96-22, ICS Dept. Univ. California Irvine, 1996.

[15] K.-R. Müller, S. Mika, G. Rätsch, K. Tsuda, and B. Schölkopf, "An introduction to kernel-based learning algorithms," *IEEE Trans. Neural Networks*, vol. 12, no. 2, pp. 181–201, Mar. 2001.

[16] E. Osuna, R. Freund, and F. Girosi, "An improved training algorithm for support vector machines," in *Proc. IEEE VII Workshop on Neural Networks for Signal Processing*, Piscataway, NJ, 1997, pp. 276–285.

[17] J. C. Platt, "Fast training of SVMs using sequential minimal optimization," in *Advances in Kernel Methods – Support Vector Learning*, B. Schölkopf, C. J. C. Burges, and A. J. Smola, Eds. Cambridge, MA: MIT Press, 1998.

[18] C. Stone, "Consistent nonparametric regression," *Ann. Stat.*, vol. 8, pp. 1348–1360, 1977.

[19] V. Vapnik, *Estimation of Dependencies Based on Empirical Data*. Nauka, Moscow, 1979. (In Russian). English translation. New York. Springer Verlag, 1982.

[20] ——, *The Nature of Statistical Learning Theory*. New York: Springer-Verlag, 1995.

[21] ——, *Statistical Learning Theory*. New York: Wiley, 1998.

[22] ——, "An overview of statistical learning theory," *IEEE Trans. Neural Networks*, vol. 10, no. 5, pp. 988–999, Sep. 1999.



**Cesare Alippi** (SM'94–F'06) received the Dr.Ing. degree in electronic engineering (summa cum laude) in 1990 and the Ph.D. degree in computer engineering in 1995, both from Politecnico di Milano, Milan, Italy. He has completed research work in computer sciences at the University College, London, U.K., and the Massachussets Institute of Technology, Cambridge.

Currently, he is a Full Professor of information processing systems at the Politecnico di Milano. His research interests include application-level analysis and synthesis methodologies for embedded systems, computational intelligence, and wireless sensor networks. His research results have been published in more that 120 technical papers in international journals and conference proceedings.

Dr. Alippi is Associate Editor of the IEEE TRANSACTIONS ON NEURAL NETWORKS and the IEEE TRANSACTIONS ON INSTRUMENTATION AND MEASUREMENT.



**Pietro Braione** received the Dr.Eng. degree in computer science engineering in 2000 and the Ph.D. degree in information technology in 2004, both from Politecnico di Milano, Milan, Italy.

His research interests include machine learning, computational intelligence, and statistical learning theory.