Mach Learn (2010) 80: 63–84
DOI 10.1007/s10994-010-5183-x

Classification with guaranteed probability of error

Marco C. Campi

Received: 1 July 2009 / Revised: 8 February 2010 / Accepted: 24 March 2010 /
Published online: 17 April 2010

Abstract We introduce a general-purpose classifier that we call the *Guaranteed Error Machine*, or GEM, and two learning algorithms that are used for the training of GEM, a *real GEM algorithm* and an *ideal GEM algorithm*. The real GEM algorithm is for use in real applications, while the ideal GEM algorithm is introduced as a theoretical tool. Differently from most learning machines, GEM has a ternary-valued output, that is besides 0 and 1 it can return an *unknown* label, expressing doubt. Our central result is that, under general conditions, the statistics of the generalization error of the GEM machine obtained with the ideal GEM algorithm is *universal*, in the sense that *it remains the same, independently of the (unknown) mechanism that generates the data*. As a consequence, the user can select a desired level of generalization error and the learning machine is automatically adjusted so as to meet this desired level, and no knowledge of the data generation mechanism is required in this process; the adjustment is achieved by modulating the size of the region where the machine returns the *unknown* label. The key-point is that no conservatism is present in this process because the statistics of the generalization error is known. We further show that the generalization error of the machine obtained with the real algorithm is always no larger than the generalization error of the machine obtained with the ideal algorithm. Thus, the generalization error computed for the latter can be rigorously used as a bound for the former, and, moreover, it provably provides tight evaluations in typical cases.

Keywords Computational learning theory · Guaranteed generalization error · Convex optimization · Invariant statistics · Ternary-valued classifiers

Editor: Nicolo Cesa-Bianchi.

M.C. Campi (✉)
Department of Electrical Engineering, University of Brescia, Via Branze 38, 25123 Brescia, Italy
e-mail: marco.campi@ing.unibs.it
url: http://www.ing.unibs.it/~campi

1 Introduction

1.1 Preliminaries and notations

Machine learning studies the problem of predicting the *class label* of *instances*. An instance x, $x \in \mathbb{R}^d$, is a vector of measured attributes, and the class label y is a discrete quantity,[1] belonging to a finite set, whose value is determined by x, that is there exists a function $y = y(x)$.[2] We here consider binary classification, i.e. $y \in \{0, 1\}$. 0 and 1 represent two different classes, whose meaning varies depending on the application at hand and can e.g. be sick or healthy, right or wrong, male or female. To an x, there is associated a $y \in \{0, 1\}$, but we have no access to the map $y = y(x)$, so, given an x, we have to guess its class y by means of some *classifier* $\hat{y} = \hat{y}(x)$. A classifier errs on x if $y(x) \neq \hat{y}(x)$, and the goal of machine learning is that of constructing classifiers that err as rarely as possible.

In *statistical* machine learning, it is assumed that x values occur according to a probability μ and the reliability of a classifier is measured by $\mu(y(x) \neq \hat{y}(x))$, the probability of the set in $\mathbb{R}^d$ where $y(x)$ and $\hat{y}(x)$ do not agree. $\mu(y(x) \neq \hat{y}(x))$ is called the *probability of error* (or *generalization error*) and it will be indicated in this paper with $PE(\hat{y})$.

Given $\hat{y}(\cdot)$, $PE(\hat{y})$ cannot be computed, for its computation would require knowledge of μ and $y(\cdot)$. Hence, selecting a good classifier cannot rely on direct minimization of this probability. Instead, we assume that one has access to a database of past examples $\mathcal{E}_N := (x_1, y_1), \ldots, (x_N, y_N)$, where the x_i's are independently extracted according to μ and $y_i = y(x_i)$, and is asked to select a classifier $\hat{y}_N(\cdot)$ based on $\mathcal{E}_N$. An algorithm is a rule that makes this selection.

See e.g. (Vapnik 1995, 1998; Devroye et al. 1996; Cristianini and Shawe-Taylor 2000; Schölkopf and Smola 2002) for general presentations of machine learning.

1.2 The stochastic nature of $PE(\hat{y}_N)$

Given a data generation mechanism, i.e. a pair $(\mu, y(\cdot))$, $PE(\hat{y})$ is a deterministic function that associates a number in $[0, 1]$ to any given classifier $\hat{y}(\cdot)$. If we further substitute $\hat{y}(\cdot)$ with $\hat{y}_N(\cdot)$, the classifier obtained from an algorithm, $PE(\hat{y}_N)$ becomes a random variable formed by the composition of $\hat{y}_N(\cdot)$, which depends on the random examples $\mathcal{E}_N$, with the deterministic function $PE(\cdot)$.[3] The dependence of $PE(\hat{y}_N)$ on $\mathcal{E}_N$ expresses the fact that different sets of examples can be more or less effective to form an estimate of $y(\cdot)$ and, when an algorithm is fed with $\mathcal{E}_N$, it returns a classifier $\hat{y}_N(\cdot)$ that more or less closely match $y(\cdot)$ depending on the seen $\mathcal{E}_N$.

In more formal terms, let us consider a given fixed data generation mechanism $(\mu, y(\cdot))$. This assigns a probabilistic model according to which each single example $(x_i, y_i) = (x_i, y(x_i))$ is drawn. From this model, a probabilistic model for $\mathcal{E}_N$ can be obtained by the independence of examples: a multi-extraction of instances $x_1, x_2, \ldots, x_N$ is selected in $(\mathbb{R}^d)^N = \mathbb{R}^d \times \cdots \times \mathbb{R}^d$ according to the product probability $\mu^N := \mu \times \cdots \times \mu$, and this multi-extraction maps into $\mathcal{E}_N := (x_1, y(x_1)), \ldots, (x_N, y(x_N))$. Thus, $\mathcal{E}_N$ is a random element defined over $((\mathbb{R}^d)^N, \mu^N)$. If we now choose an algorithm, to each $\mathcal{E}_N$ a classifier

[1] At times, a broader perspective where y is not discrete is also considered, see (Vapnik 1995).

[2] The approach of this paper carries over to the case when y takes value 0 or 1 with given probabilities—nonzero Bayes risk, see (vi) in Sect. 3.

[3] In the statistical learning literature, $PE(\hat{y}_N)$ is sometimes called the "conditional probability of error".

$\hat{y}_N(\cdot)$ is associated and in turn $PE(\hat{y}_N) = \mu(y(x) \neq \hat{y}_N(x))$ becomes a fixed number for any given $\mathcal{E}_N$. Thus, through $\mathcal{E}_N$, $PE(\hat{y}_N)$ is a random variable defined over $((\mathbb{R}^d)^N, \mu^N)$.

One fact that is important to make explicit is that $PE(\hat{y}_N)$ is a random variable whose probabilistic characteristics, its distribution in particular, depend on the unknown data generation mechanism $(\mu, y(\cdot))$. Thus, the same algorithm can generate classifiers that are likely to be more or less reliable depending on the context in which the algorithm is applied.

1.3 The guaranteed error machine

In this paper, we introduce $\{0, 1, \textit{unknown}\}$-valued classifiers that have a guaranteed probability of error, also called Guaranteed Error Machines (GEM). The learning algorithm for the GEM has a tunable parameter k which may be used to enlarge the region where the classifier does provide a classification, that is it returns a value 0 or 1. A large value of k leads to classifiers that are more likely to return a 0 or 1 value, but these classifiers misclassify more frequently, whereas smaller values of k correspond to more risk-averse classifiers paying emphasis on reducing the probability of misclassification, but also returning *unknown*'s with higher probability.

In more precise terms, let us start by generalizing to $\{0, 1, \textit{unknown}\}$-valued classifiers the definition of probability of error: define $PE(\hat{y}_N) = \mu(\hat{y}_N(x) = 0 \textit{ or } 1, \textit{ and } y(x) \neq \hat{y}_N(x))$, that is $PE(\hat{y}_N)$ is now the probability that an answer is issued and this answer is incorrect. First, we introduce an *ideal* GEM *algorithm*. For each value of the tunable parameter k, our Theorem 1 gives the *exact probability distribution* of $PE(\hat{y}_N)$, and shows that, under mild conditions, this distribution *does not depend on the data generation mechanism* $(\mu, y(\cdot))$, i.e. it is *universal*. This is in contrast with what happens in other machines, as discussed in Sect. 1.2. This deep theoretical result bears important practical implications: when selecting the tunable parameter k, the user has full control on how $PE(\hat{y}_N)$ distributes since this distribution does not depend on the unknown element of the problem, the data generation mechanism $(\mu, y(\cdot))$. Thus, the user can choose a level of risk and select that k that meets the selected risk level, even without any knowledge of $(\mu, y(\cdot))$. For simple-to-estimate data generation mechanisms, the algorithm will construct classifiers that return a 0 or 1 answer most of the time, while an *unknown* will be more often issued in the case of difficult-to-estimate data generation mechanisms. The very point is that no conservatism is present in this selection procedure since the algorithm pushes the classifier all the way down to the desired level of risk, automatically and without prior knowledge on the data generation mechanism.

The ideal GEM algorithm selects the size of the *unknown* region to tune the level of risk. Like pulling down one end of a rope wrapped around a pulley lifts the other end, similarly in the ideal GEM algorithm pulling down the level of risk determines an increase in the size of the *unknown* region. This tuning mechanism, however, is effective until the *unknown* region is present, that is it does not shrink to the empty set so that the classifier classifies the whole x space. In the ideal GEM algorithm, when this happens a misclassified region is artificially introduced to obtain the selected risk level.

The ideal GEM algorithm is used in this paper merely as a theoretical tool, and it is not for use in real applications. We next introduce a *real* GEM *algorithm*, the algorithm used in practice. The machine obtained with the real GEM algorithm is identical to that constructed by the ideal GEM algorithm, except that the artificial misclassified region is not introduced. As a consequence, most of the time the two algorithms generate the same classifier,[4] while in

[4]They generate the same classifier whenever the *unknown* region does not become empty, a situation which is not uncommon, see also the literature on classifiers that can choose not to classify referenced in Sect. 1.4.

simple-to-estimate classification problems the real GEM algorithm may reach the condition of total classification (no *unknown*'s) even before hitting the selected level of risk. Hence, we show that the probability of error of the classifier obtained with the real GEM algorithm is always no more than that of the classifier obtained with the ideal GEM algorithm and the results valid for the ideal GEM algorithm can therefore be rigorously used to establish reliability results for the real GEM algorithm.

Thanks to the theoretical result that the distribution of $PE(\hat{y}_N)$ is universal, the GEM machine can be tuned so as to obtain a classified region as extended as it is possible compatibly to the selected level of risk. Still, whether the classified region is indeed satisfactorily large depends on the environment in which classification is performed and can only be evaluated a-posteriori, after the classifier is constructed. When conceiving the GEM machine, we designed it so that it generates wide regions of classification and we observed in experimental examples that the classified region tends to grow rapidly in the x domain. Moreover, we observed that when the GEM classifier eventually comes to the point of classifying all the observations, it had an empirical probability of error (evaluated by cross-validation) similar to other machine learning approaches, see the empirical results in Sect. 4. So, GEM seems to perform similarly to other methods in case of full classification, while it gains two important additional advantages: (i) theoretical and non-conservative a-priori guarantees of generalization are available; (ii) can be tuned by the user so as to trade part of its classification capability (i.e. it may output an *unknown*) for an increased level of generalization properties.

Further discussion on the theoretical results and on their use in practice is postponed to Sect. 3 after the GEM algorithms have been formally introduced in the next Sect. 2.

1.4 Bibliographical notes

Classifiers that can choose not to classify (like our *unknown* here) have been previously introduced by other authors in contexts different from ours. A Bayesian classifier with rejection option was considered as early as in (Chow 1957). See (Chow 1970; Gyorfi et al. 1978; Muzzolini et al. 1998; Ferri et al. 2003) for other studies. Theoretical results on *abstaining classifiers* are provided e.g. in (Rivest and Sloan 1988; Freund et al. 2004; Ruckert and Kramer 2004; Pietraszek 2005), see also Chap. 3 of (Vovk et al. 2005) where the efficiency of recursive conformal predictors that can output multiple labels set are studied. However, as recently noted in (Herbei and Wegkamp 2006; Bartlett and Wegkamp 2008), more theoretical work is needed, and indeed expected, in this context.

The results of this paper have some aspects in common with "conformal prediction", see (Vovk et al. 2005; Shafer and Vovk 2008). Conformal prediction operates on-line and generates predictions sequentially:[5] based on the examples $(x_1, y_1), \ldots, (x_t, y_t)$ and after seeing the next instance x_{t+1}, a prediction $\hat{y}_{t+1}$ of y_{t+1} is generated; the actual y_{t+1} is then observed and the predictor incurs an error if $y_{t+1} \neq \hat{y}_{t+1}$,[6] moreover the database of examples is updated to also include the new observation, i.e. the database is now $(x_1, y_1), \ldots, (x_{t+1}, y_{t+1})$; this same scheme is repeated at time $t+2, t+3, \ldots$. Grounded on a theoretically solid and yet flexible approach, conformal prediction generates hedged on-line predictions with a guaranteed level of cumulative errors. We here also generate hedged predictions, but in

[5] This framework is also called *transductive*, see Vapnik (1995, 1998).

[6] Similarly to our *unknown*, conformal predictors can return a multiple output $\{0, 1\}$ which never makes an error. So, we should more precisely write $y_{t+1} \notin \hat{y}_{t+1}$.

the more traditional *inductive* framework where examples are used off-line to determine a classifier. In this context, our Theorem 1 permits one to compute (ϵ, δ) guarantees in a PAC-learning sense.

The results of this paper also have connections with statistical tolerance limits. A tolerance limit is a set of a measurable space constructed from data that contains a portion of the total probability of the space (or "coverage") that distributes independently of the probability with which data are generated. According to this terminology, Theorem 1 states that the classifier constructed by GEM is indeed a tolerance limit in the space $\mathbb{R}^d \times \{0, 1\}$ for (x, y). Tolerance limits were introduced by Wilks (1941) in relation to the problem of partitioning the $[0, 1]$ segment into blocks. Similar to Wilks' approach, further generalizations provided by (Wald 1943; Tukey 1947; Fraser 1951; Fraser 1953; Kemperman 1956) are still based on order statistics. Existence of an ordering permits one to identify sets where data have to fall in order to obtain given coverages. In contrast, the approach of this paper is based on convex optimization and does not allow for such an interpretation.

Interestingly, in the 1950's, in addition to tolerance limits, expectation tolerance limits were also introduced, (Fraser and Guttman 1956; Fraser 1957), which have the property that the expectation of the coverage is independent of the probability with which data are generated. This is the same property that underpins the theoretical results valid for conformal predictors. So, tolerance limits are predecessors of the GEM, and expectation tolerance limits are the predecessors of conformal predictors.

2 The GEM algorithm

We first introduce the real GEM algorithm (henceforth also simply called the GEM algorithm), then followed by the ideal GEM algorithm obtained as a modification of the GEM algorithm. In the algorithms, k is an integer that has to be fixed in advance; depending on k, different probability distributions for $PE(\hat{y}_N)$ are obtained, as given in Theorem 1. We also recall that d is the size of x and N is the number of examples. Throughout, we assume $k \leq N - 1$.

THE GEM ALGORITHM

0. Let $x_B = x_1$, $j = 1$, $P = \{2, 3, \ldots, N\}$, and $Q = \emptyset$ (empty set).

1. **If** $|Q| \leq k - (d(d+1)/2 + d)$ ($|\cdot|$ means cardinality), go to point 2a;
elseif $k - (d(d+1)/2 + d) < |Q| \leq k - (d+1)$, go to point 2b;
else, go to point 2c.

2a. Solve the following convex optimization problem:

$$\begin{aligned} &\min_{A=A^T \in \mathbb{R}^{d\times d}, b\in\mathbb{R}^d} \ Trace(A) \\ &\text{subject to:} \qquad (x_i - x_B)^T A (x_i - x_B) + b^T (x_i - x_B) \geq 1, \\ &\qquad\qquad \text{for all } i \in P \text{ such that } y_i \neq y(x_B) \\ &\qquad\qquad \text{and } A \succeq 0 \quad (A \text{ positive semi-definite}). \end{aligned}$$

If more than one pair (A, b) solves the problem, take the pair (A, b) with smallest norm of b, $\|b\| = \sqrt{\sum_{r=1}^{d} b_r^2}$. If a tie still occurs, break it according to a lexicographic rule on the elements of A and b. Let (A^*, b^*) be the optimal solution.
Go to point 3.

2b. Solve the following convex optimization problem:

$$\begin{aligned} &\min_{a\geq 0, b\in\mathbb{R}^d} a \\ &\text{subject to: } a\cdot\|x_i - x_B\|^2 + b^T(x_i - x_B) \geq 1, \\ &\qquad\qquad \text{for all } i \in P \text{ such that } y_i \neq y(x_B). \end{aligned}$$

If more than one pair (a, b) solves the problem, take the pair (a, b) with smallest norm of b. Let (a^*, b^*) be the optimal solution, and define $A^* = a^* I$ (I = identity matrix). Go to point 3.

2c. Solve the following convex optimization problem:

$$\begin{aligned} &\min_{a\geq 0} \quad a \\ &\text{subject to: } a\cdot\|x_i - x_B\|^2 \geq 1, \quad \text{for all } i \in P \text{ such that } y_i \neq y(x_B). \end{aligned}$$

Let a^* be the optimal solution, and define $A^* = a^* I$ and $b^* = 0$.
Go to point 3.

3. Form the region $\mathcal{R}_j = \{x : (x - x_B)^T A^*(x - x_B) + (b^*)^T(x - x_B) < 1\}$, and let $\ell_j = y(x_B)$.

Update P by removing the indexes of the instances in $\mathcal{R}_j$. If P is empty, go to point 4; update $Q = Q \cup \{\text{indexes of the active instances}\}$, where the "active" instances are those that fall on the boundary of $\mathcal{R}_j$;
if $|Q| < k$, search for the "active" instance furthest away from x_B (if there is more than 1 instance at the furthest distance from x_B take any one of them) and rename as x_B this instance; let $j = j + 1$, and go to point 1;
else, go to point 4.

4. Define the classifier

$$\hat{y}_N(x) = \begin{cases} \textit{unknown}, & \text{if } x \notin \mathcal{R}_r, 1 \leq r \leq j \\ \ell_q, & \text{otherwise, with } q = \min r \text{ such that } x \in \mathcal{R}_r, 1 \leq r \leq j. \end{cases}$$

Points 2a, 2b, and 2c construct regions containing examples all having the same class label as the label $y(x_B)$ of the "base" instance x_B. Point 2a constructs regions more complex than 2b, which are in turn more complex than those in 2c: 2a constructs (hyper)ellipsoids containing x_B, those in 2b are (hyper)spheres containing x_B, while those in 2c are (hyper)spheres having x_B at their center. If P does not become empty, the procedure is halted when the total number of the active instances $|Q|$ reaches the selected bound k, and redirecting the algorithm to simpler constructions when $|Q|$ gets close to k as done in point 1 serves the purpose of exactly reaching k upon termination of the algorithm. As we shall see, this is the key-property to make the generalization error of the algorithm guaranteed.

The first example plays a special role in the construction. To make the algorithm independent of the ordering of the training examples one can conceive to test one by one all examples in the role of first base example and then select that example that provided the best performance (fewer *unknowns*). While this is a sensible way of proceeding, one has to keep in mind that Theorem 1 below has to be used with care in this case and one has to multiply the right-hand-side of (1) by N, the number of different possible choices of first base example, obtaining a bound instead of an equality in (1).

Altogether, the classifier has the structure of a data-dependent decision list, see e.g. (Anthony 2004; Marchand and Sokolova 2005; Klivans and Servedio 2006).

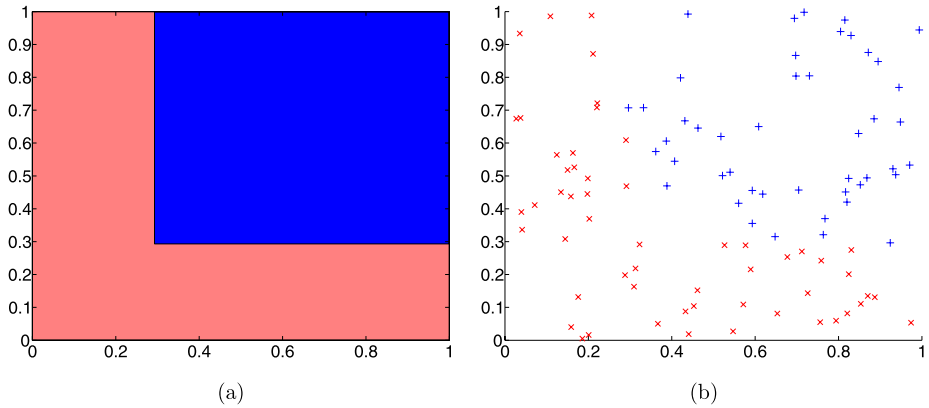

Fig. 1 **a** Function $y(x)$, upper square has label 1. **b** Data set: $\times = 0$, $+ = 1$

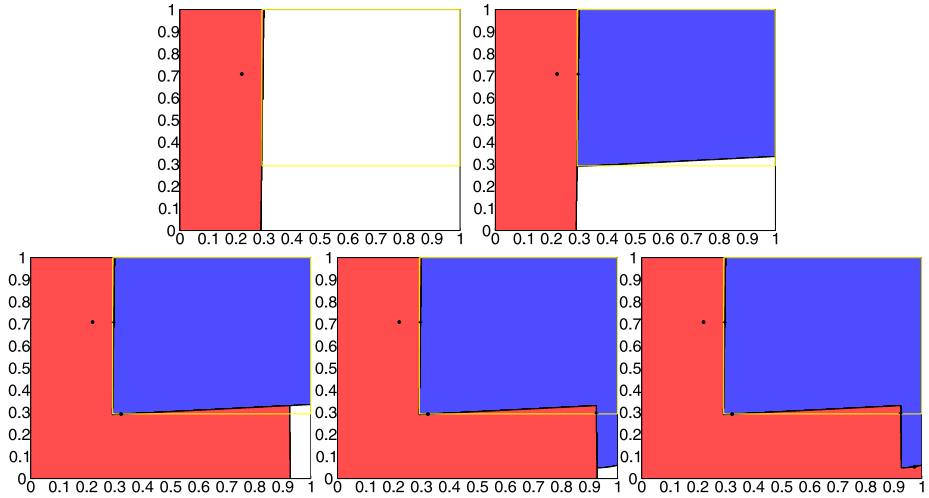

Fig. 2 The regions constructed by the GEM algorithm

To help the reader understand the algorithm operation, a toy example is provided in Figs. 1 and 2. Figure 1(a) represents the function $y(x)$; $N = 100$ examples were extracted according to a uniform distribution on $[0, 1]^2$ and they are shown in Fig. 1(b). We set $k = 5$ and executed the GEM algorithm. The algorithm performed one construction according to point 2a, two constructions according to 2b, and two constructions according to 2c, determining each time $n = 1$ active instances. Figure 2 displays the regions constructed in succession by the algorithm. No *unknown* region was left at the end.

If the GEM algorithm terminates at point 3 with P empty, then it is easy to see that all the $\mathbb{R}^d$ space of x is classified. The ideal GEM algorithm is defined as a modification of the real GEM algorithm, where the modification is effective only when P becomes empty:

THE IDEAL GEM ALGORITHM

Add at the end of point 4 of the real GEM algorithm the following part:
"If P is empty, construct the largest open ball $\mathcal{B}$ centered in x_1 that contains other (besides x_1) $k - |Q| - 1$ of the instances x_i, $i \in \{2, 3, \ldots, N\} - Q$. So, if e.g. $k = 20$ and $|Q| = 18$, $\mathcal{B}$ contains just one instance and, by enlarging $\mathcal{B}$ as much as possible, it touches another instance at its boundary. Then, let $Q = Q \cup \{$indexes of the instances inside and on the boundary of $\mathcal{B}\}$, and redefine $\hat{y}_N(x) = 1 - y(x)$ for all x in $\mathcal{B}$ except for the instances x_i, $i \in \{2, 3, \ldots, N\} - Q$, for which we maintain a correct classification $\hat{y}_N(x_i) = y(x_i)$; thus, $\hat{y}_N(x)$ misclassifies all $x \neq x_i$ in $\mathcal{B}$."

The ideal GEM algorithm cannot be used in practice since the redefinition of $\hat{y}_N(x)$ in the ball $\mathcal{B}$ requires knowledge of $y(x)$;[7] and, moreover, it would not make sense in practice to deliberately misclassify in $\mathcal{B}$! We show in Theorem 1 that, for the ideal GEM algorithm, $PE(\hat{y}_N)$ is exactly distributed as a Beta variable, independently of $(\mu, y(\cdot))$. We also show that this Beta can be used to bound the misclassification of the machine obtained with the real GEM algorithm, which always misclassifies no more than the machine obtained with the ideal GEM algorithm. Since the modification in the ideal GEM algorithm is effective only in the special condition of total classification, for normally assumed risk levels the two algorithms behave the same and the Beta distribution provides tight risk evaluations for the real GEM algorithm.

Theorem 1 *Suppose that the probability μ according to which x values are extracted has density. Then, the probability distribution of $PE(\hat{y}_N)$ for the ideal GEM algorithm is given by*

$$F_{PE}(z) := \mu^N\{PE(\hat{y}_N) \leq z\} = \sum_{i=k}^{N-1} \binom{N-1}{i} z^i (1-z)^{N-1-i}. \tag{1}$$

Note that $F_{PE}(z)$ does not depend on the data generation mechanism $(\mu, y(\cdot))$.

Moreover, this $F_{PE}(z)$ "dominates" the probability distribution of $PE(\hat{y}_N)$ for the real GEM algorithm, in the sense that $\mu^N\{PE(\hat{y}_N) \leq z\} \geq \sum_{i=k}^{N-1} \binom{N-1}{i} z^i (1-z)^{N-1-i}$ for this algorithm.

In the theorem, $\mu^N = \mu \times \cdots \times \mu$ is the product probability according to which the examples $\mathcal{E}_N = (x_1, y_1), \ldots, (x_N, y_N)$ are extracted. The right-hand-side of (1) is a $Beta(k, N-k)$ distribution (see e.g. Schervish 1995), and in words the theorem says that the probability of error $PE(\hat{y}_N)$ of the GEM machine obtained with the ideal GEM algorithm distributes as a $Beta(k, N-k)$ variable, irrespective of the data generation mechanism $(\mu, y(\cdot))$. Moreover, for the real GEM algorithm, $PE(\hat{y}_N) \leq z$ holds with probability no less than that for the ideal GEM algorithm. To visualize the result, the density of a $Beta(k, N-k)$ is depicted in Fig. 3 for a few values of N and k.

The proof of Theorem 1 is given in Sect. 5. In the next section, we discuss the practical use of the GEM algorithm, while Sect. 4 presents some empirical results with real data.

[7] In conformal prediction, (Vovk et al. 2005; Shafer and Vovk 2008), the empty prediction $\emptyset$ has been introduced; $\emptyset$ always incurs an error. We could have similarly used the empty prediction and output $\emptyset$ instead of $1 - y(x)$ in $\mathcal{B}$, resulting in an implementable algorithm. Nonetheless, in this paper we have preferred not to enlarge the label set of the classifier by adding $\emptyset$ since sticking to 0, 1, *unknown* is easier and since the ideal algorithm is here seen as a theoretical tool not for real implementation.

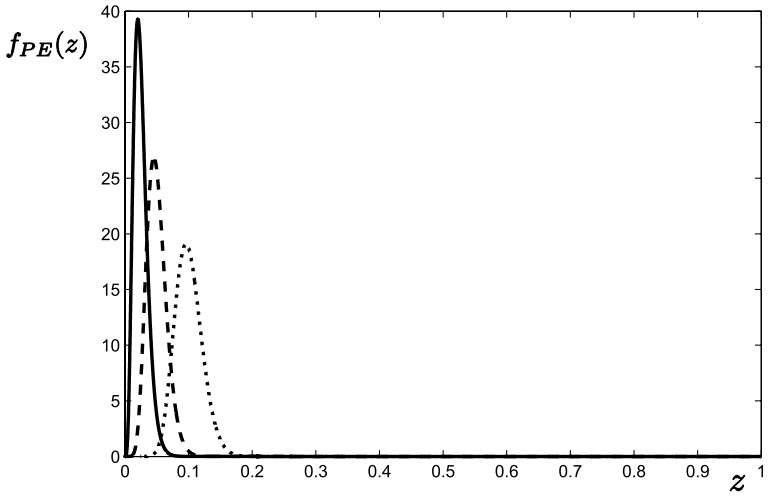

Fig. 3 The $Beta(k, N-k)$ density. $N = 200$, $k = 5$ (*solid*), 10 (*dashed*), 20 (*dotted*)

3 Practical use of the GEM algorithm

A number of remarks on the interpretation of the result presented in the previous section, as well as on its practical use, are in order.

(i) *The theoretical result*. The fact that with the ideal GEM algorithm $PE(\hat{y}_N)$ has a $Beta(k, N-k)$ distribution that does not depend on how the examples are generated is, we believe, a deep theoretical result. It can be phrased by saying that the distribution of $PE(\hat{y}_N)$ is *universal*. The $Beta(k, N-k)$ distribution also tightly describes the behavior of the real GEM algorithm, as confirmed by the empirical results in Sect. 4, and, in any case, it provides theoretically guaranteed lower bounds for the probability that $PE(\hat{y}_N) \leq z$.

(ii) *A Monte-Carlo test*. A Monte-Carlo test was performed for the $y(x)$ function of Fig. 1(a). $N = 200$ examples were extracted $M = 1000$ times. For each multi-extraction of 200 examples, we constructed the classifier $\hat{y}_{200}$ with the ideal GEM algorithm with $k = 5$ and then computed the corresponding $PE(\hat{y}_{200})$. Note that this computation is here possible due to the artificial nature of the problem example, i.e. $(\mu, y(\cdot))$ is known. The histogram obtained from the $M = 1000$ trials is shown in Fig. 4 against the theoretical $Beta(5, 200-5)$ density. Should we have extracted x values according to a probability density different from the uniform one, or should the $y(x)$ function have been a different one, the density of $PE(\hat{y}_{200})$ would have remained the same.

(iii) *Practical use of the result*. The $Beta(k, N-k)$ distribution can be used in different ways.

A first use is that one selects a risk level ϵ and requires that $PE(\hat{y}_N) \leq \epsilon$ holds with high probability $1-\delta$. Since the $Beta(k, N-k)$ tail is very thin, $PE(\hat{y}_N) \leq \epsilon$ can be enforced with such a high level of probability that the complement event that $PE(\hat{y}_N) > \epsilon$ looses any practical relevance. To appreciate this behavior, in Table 1 we give the probability that $PE(\hat{y}_N) > \epsilon$ for $\epsilon = 5\%$ and $N = 1000$ for different values of k.

For a quick evaluation of the largest k such that $\mu^N\{PE(\hat{y}_N) > \epsilon\}$ is a rare event with probability no more than a given δ, one can resort to the Chernoff bound for the *Beta* tail, see

Fig. 4 Histogram of $PE(\hat{y}_{200})$

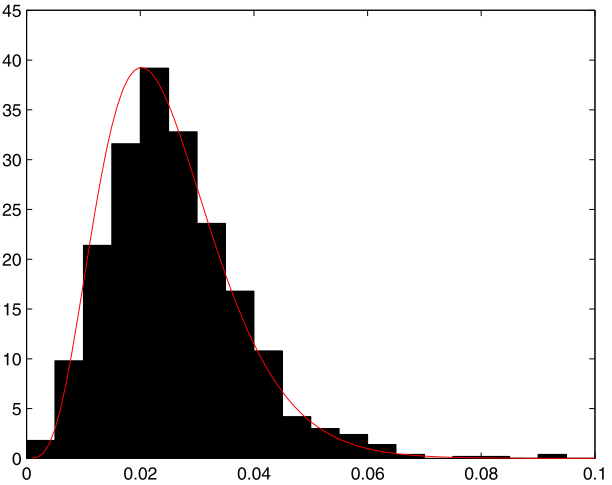

Table 1 $\delta := \mu^N\{PE(\hat{y}_N) > 5\%\}$ for different values of k, $N = 1000$

k	1	2	3	4	5
δ	5.5×10^{-23}	2.9×10^{-21}	7.9×10^{-20}	1.4×10^{-18}	1.9×10^{-17}

k	10	15	20	25	30
δ	5.4×10^{-13}	9.7×10^{-10}	2.9×10^{-7}	2.4×10^{-5}	7.3×10^{-4}

(Chernoff 1952) for the original reference or e.g. (Vidyasagar 1997), yielding the following corollary to Theorem 1:

Corollary 1 *Under the assumptions in Theorem 1, given $\epsilon, \delta \in (0,1)$ we have that $\mu^N\{PE(\hat{y}_N) \leq \epsilon\}$ holds with probability $1-\delta$ both for the ideal and the real GEM algorithm provided that*

$$k \leq \frac{1}{2}(N-1)\epsilon + \ln \delta. \tag{2}$$

Proof Note that (2) implies

$$\ln \delta \geq k - \frac{1}{2}(N-1)\epsilon \geq k - \frac{1}{2}(N-1)\epsilon - 1 - \frac{(1-k)^2}{2(N-1)\epsilon} = -\frac{[(N-1)\epsilon + (1-k)]^2}{2(N-1)\epsilon}, \tag{3}$$

so that $\delta \geq \exp(-\frac{[(N-1)\epsilon+(1-k)]^2}{2(N-1)\epsilon})$. On the other hand, the Chernoff bound (Chernoff 1952 or Vidyasagar 1997) says that the tail for $z > \epsilon$ of a $Beta(k, N-k)$ distribution is no more than $\exp(-\frac{[(N-1)\epsilon+(1-k)]^2}{2(N-1)\epsilon})$, so leading to the conclusion that

$$\delta \geq \text{“tail for } z > \epsilon \text{ of a } Beta(k, N-k)\text{”} \geq \mu^N\{PE(\hat{y}_N) > \epsilon\}. \qquad \square$$

For e.g. $N = 1000$, $\epsilon = 5\%$ and $\delta = 2.4 \times 10^{-5}$, (2) gives $k = 14$ (compare with Table 1 where the real k is 25).

A second use of the $Beta(k, N-k)$ distribution is to tune k so that the total probability of seeing N examples and that the next $(N+1)$-th example is misclassified is below a fixed threshold. This probability is given by the mean of $PE(\hat{y}_N)$, that is by the mean of the $Beta(k, N-k)$ distribution, which is (see e.g. Schervish 1995):

$$E[PE(\hat{y}_N)] = \frac{k}{N}. \tag{4}$$

Thus, with e.g. $N = 1000$ examples one can be 5% confident that the 1001-st example will not be misclassified if $k = 50$.

(iv) *An impossibility result*. Obtaining a result similar to Theorem 1 for $\{0, 1\}$-valued classifiers is impossible. Theorem 1 says that $PE(\hat{y}_N)$ distributes as a $Beta(k, N-k)$ variable and, therefore, the probability of the event $\{PE(\hat{y}_N) > z\}$ can be tuned to be small at will by suitably selecting k and N. In contrast, for $\{0, 1\}$-valued classifiers the following FACT holds:

Fact *Let $\rho > 0$ be an arbitrary small number. For any N and for any algorithm generating $\{0, 1\}$-valued classifiers, there exists a data generation mechanism $(\mu, y(\cdot))$ such that*

$$\mu^N\{PE(\hat{y}_N) > \epsilon\} \geq 0.5 - \epsilon - \rho.$$

This FACT immediately follows from Theorem 7.1 in (Devroye et al. 1996), a theorem originally proven in (Devroye 1982). Thus, with e.g. $\rho = 0.1$ we see that $\mu^N\{PE(\hat{y}_N) > \epsilon = 5\%\} \geq 0.5 - 0.05 - 0.1 = 0.35$ and no algorithm exists guaranteeing that the probability of error is below 5% with high probability close to 1, no matter how large N is. This is not a surprising fact: for any large N, there are difficult-to-estimate data generation mechanisms $(\mu, y(\cdot))$ where the algorithm fails to behave satisfactorily, if the algorithm is not allowed to issue the *unknown* symbol.

(v) *More general algorithms*. An inspection of the proof of Theorem 1 reveals that the key-property of the ideal GEM algorithm to make the result valid is that, upon termination, the algorithm has stored k active instances in Q. Other constructions than those in points 2a, 2b, and 2c of the algorithm permits to obtain this result. For example, one can introduce one more construction where the elements of A and b are constrained in such a way that the total number of degrees of freedom of the corresponding optimization problem is, say, p and enter this point only if $|Q| \leq k - p$. Theorem 1 easily extends to cover this case. More generally, along extensions similar to that just traced, the analysis of this paper carries over to a whole family of variants of the GEM algorithm.

(vi) *The case of stochastic $y(x)$*. So far we have assumed that, given an x, $y(x)$ is univocally determined. Generalizing this situation, we can now consider the possibility that $y(x)$ is not totally determined by x and that it can take value 0 or 1 according to a random scheme. To formalize this situation, introduce the space $\mathbb{R}^d \times \{0, 1\}$ for the examples (x, y) and let now μ be a probability defined over $\mathbb{R}^d \times \{0, 1\}$ according to which the examples are extracted. In this context, $\hat{y}_N(\cdot)$ is still a classifier constructed according to the GEM algorithms that associates just one label (0, 1, or *unknown*) to each x, and we let $PE(\hat{y}_N) = \mu((x, y) : \hat{y}_N(x) = 0 \text{ or } 1, \text{ and } y \neq \hat{y}(x))$. The proof of Theorem 1 remains substantially unchanged, and one can prove that the theorem's result carries over to this case.

4 Empirical results

We present empirical results obtained by applying GEM to some publicly available data sets, Glass, Iris, BreastW, Haberman, Pima, Bupa, Credit. The first data set has been obtained from http://www.cs.utsa.edu/~bylander/cs4793/, while the other six have been downloaded from the UCI machine learning repository, (Asuncion and Newman 2007).

Tables 2–4 display the results obtained with Glass and Iris, where with Iris we have classified in Table 3 Iris Setosa vs other Iris types (Versicolor and Virginica) and in Table 4 we have eliminated the examples of Iris Setosa from the data set and have classified Iris Versicolor vs Iris Virginica. k is the tunable parameter in the GEM algorithm; *# of errors* is the total number of errors in a 10-fold cross validation. Precisely, we left out each time a number of examples equal to the integer part of the total number of examples divided by 10, and these examples were used as a test set. *# of errors* was then computed as the sum of the errors found in the test set at each trial. In parentheses, we have reported the ratio $\frac{\#\ of\ errors}{total\ \#\ of\ test\ examples}$, an estimate of the expected probability of error; *# of unknowns* and *# of correct* is obtained similarly to *# of errors* summing the number of unknowns and of correctly classified examples in the 10 test sets; the last row gives the theoretical value of $E[PE(\hat{y}_N)]$, computed using (4), so e.g. figure 2.04% in Table 2 is the ratio $3/147$, where $3 = k$ and 147 is the number of training examples (total number of examples, 163, minus examples in the test set, 16). This theoretical value should be compared with the empirical ratio $\frac{\#\ of\ errors}{total\ \#\ of\ test\ examples}$, and we notice a tight adherence between the two. The theoretical value is guaranteed and when the empirical ratio exceeds the theoretical value we have to impute this to the stochastic fluctuation of the former. In a real application, the user selects a reliability level and tunes k accordingly, guided by the theory. The algorithm adjusts the learning machine automatically to hit the desired reliability level, and the user can a-posteriori inspect the number of unknowns. A large number of unknowns indicates that the classification problem was a difficult one and the user can select his/her favorite compromise between reliability and number of unknowns.

In Tables 5 through 9 we have similar results for BreastW, Haberman, Pima, Bupa, Credit. Strictly speaking, for these data sets Theorem 1 is not applicable because some attributes are discrete and therefore μ has no density. However, an inspection of the theorem proof reveals that the existence of the density serves only to prevent degenerate situations from occurring where some instances fall exactly on the boundary of the region generated by other instances, a fact that generally does not happen even when μ has no density. Again we notice a good adherence between the theoretical value and the practical behavior, showing the robustness of the theoretical result.

We further compared GEM with other methods, the nearest-neighbor classifier (NNC, Devroye et al. 1996), the support vector machine (SVM, Vapnik 1998; Cristianini and

Table 2 Glass, # of examples = 163

k	3	9	10	20	30	40	44
# of errors	3 (1.88%)	8 (5.00%)	10 (6.25%)	16 (10.00%)	26 (16.25%)	47 (29.38%)	48 (30.00%)
# of unknowns	127	103	113	75	35	1	0
# of correct	30	49	37	69	99	112	112
$E[PE(\hat{y}_N)]$	2.04%	6.12%	6.80%	13.61%	20.41%	27.21%	29.93%

Table 3 Iris Setosa, # of examples = 149

k	2	3	4
# of errors	2 (1.43%)	3 (2.14%)	3 (2.14%)
# of unknowns	5	4	0
# of correct	133	133	137
$E[PE(\hat{y}_N)]$	1.48%	2.22%	2.96%

Table 4 Iris Versicolor vs Iris Virginica, # of examples = 99

k	3	4	5	6	10
# of errors	1 (1.11%)	3 (3.33%)	5 (5.56%)	6 (6.67%)	8 (8.89%)
# of unknowns	73	64	7	4	0
# of correct	16	23	78	80	82
$E[PE(\hat{y}_N)]$	3.33%	4.44%	5.56%	6.67%	11.11%

Table 5 BreastW, # of examples = 683

k	5	10	15	20	25	30	35
# of errors	5 (0.74%)	12 (1.76%)	20 (2.94%)	25 (3.68%)	26 (3.82%)	29 (4.26%)	31 (4.56%)
# of unknowns	378	347	148	106	49	16	0
# of correct	297	321	512	549	605	635	649
$E[PE(\hat{y}_N)]$	0.81%	1.63%	2.44%	3.25%	4.07%	4.88%	5.69%

Table 6 Haberman, # of examples = 294

k	10	20	40	60	80	90	100
# of errors	12 (4.14%)	24 (8.28%)	41 (14.14%)	59 (20.34%)	85 (29.31%)	95 (32.76%)	101 (34.83%)
# of unknowns	252	217	159	105	39	15	0
# of correct	26	49	90	126	166	180	189
$E[PE(\hat{y}_N)]$	3.77%	7.55%	15.09%	22.64%	30.19%	33.96%	37.74%

Shawe-Taylor 2000) with $C = \infty$ and with finite C, and the simple set covering machine (SCM, Marchand and Shawe-Taylor 2002; Hussain et al. 2007) of type c (conjunction) and d (disjunction) and with finite p and s (which provide better performance than the SCM with

Table 7 Pima, # of examples = 768

k	10	30	50	100	150	200	250
# of errors	9 (1.18%)	28 (3.68%)	49 (6.45%)	114 (15.00%)	158 (20.79%)	209 (27.50%)	241 (31.71%)
# of unknowns	724	657	601	402	244	92	0
# of correct	27	75	110	244	358	459	519
$E[PE(\hat{y}_N)]$	1.45%	4.34%	7.23%	14.45%	21.68%	28.90%	36.13%

Table 8 Bupa, # of examples = 345

k	10	15	30	50	90	120	135
# of errors	11 (3.24%)	16 (4.71%)	31 (9.12%)	50 (14.71%)	89 (26.18%)	122 (35.88%)	129 (37.94%)
# of unknowns	308	303	255	203	104	15	0
# of correct	21	21	54	87	147	203	211
$E[PE(\hat{y}_N)]$	3.22%	4.82%	9.65%	16.08%	28.94%	38.59%	43.41%

Table 9 Credit, # of examples = 653

k	10	30	50	80	100	135	150
# of errors	12 (1.85%)	39 (6.00%)	66 (10.15%)	100 (15.38%)	117 (18.00%)	153 (23.54%)	153 (23.54%)
# of unknowns	597	445	339	199	138	13	0
# of correct	35	166	245	351	395	484	497
$E[PE(\hat{y}_N)]$	1.70%	5.10%	8.50%	13.61%	17.01%	22.96%	25.51%

infinite p and s). Since all these methods do not use unknowns, to perform a comparison we chose the k value in GEM to be the first k giving 0 unknowns. Moreover, as explained above, with GEM at each trial we left out as test examples a number of examples equal to the integer part of one tenth of the total number of the examples, so that summing up all the test examples we do not reach exactly the total size of the data set. Thus, to provide a fair comparison with the other methods, we have multiplied the number of errors observed with GEM by the ratio between the total number of examples divided by the total number of test examples. Table 10 gives the results, where the figures for NNC, SVM and SCM have been taken from (Marchand and Shawe-Taylor 2002). It may seem that GEM does not compare favorably with the other methods in most of the cases. However, it is important to remark that the reported figures for SVM and SCM are those that achieved the smallest 10-fold cross validation error among an exhaustive scan of many values of the free parameters (the kernel parameter γ and the soft margin parameter C for SVM, and parameters p and s for SCM). Thus, these parameters are "tuned" to the data set and the figures in Table 10 are underestimates of the real generalization error. On the contrary, with GEM *we displayed the results*

Table 10 *# of errors*, comparison of different methods

	GEM	NNC	SVM ($C=\infty$)	SVM (finite C)	SCM type c	SCM type d
Glass	48.2	36	42	34	35	36
BreastW	31.1	29	27	19	18	16
Haberman	103.6	107	111	71	71	93
Pima	243.5	247	243	203	189	206
Bupa	130.9	124	121	107	109	106
Credit	153.7	214	205	190	198	195

obtained in one single algorithmic run and the obtained figures are reliable estimates of the generalization error, as confirmed by their adherence to the theoretical value of $E[PE(\hat{y}_N)]$.

5 Proof of Theorem 1

We start with some set-theory results.

Consider a set Z and two maps m and t as follows.

Let k be a fixed integer. Given a finite set of at least k points of Z, say $\{z_1, z_2, \ldots, z_p\}$ with $p \geq k$, $m\{z_1, z_2, \ldots, z_p\}$ extracts k of these points and returns the set that contains the extracted points. However, we do not require that m is always defined, that is there can be sets $\{z_1, z_2, \ldots, z_p\}$ for which $m\{z_1, z_2, \ldots, z_p\}$ is not defined (this will help us in the construction of m for the classification problem, see later on). What we instead require is that m is always defined when $p = k$, in which case $m\{z_1, z_2, \ldots, z_k\} = \{z_1, z_2, \ldots, z_k\}$ since m extracts a k-dimensional subset.

As for map t, t maps any finite set of exactly k points of Z into a (possibly infinite) subset $A \subseteq Z$.

Example 1 Let $Z = \mathbb{R}$. Given a set of reals $\{z_1, z_2, \ldots, z_p\}$, we can take $m\{z_1, z_2, \ldots, z_p\} = \{z_{min}, z_{max}\}$, where z_{min} and z_{max} are the min and max numbers in the set. So, $k = 2$ here. Also, we define $t\{z_1, z_2\} = [z_1, z_2]$, the interval connecting the two points.

Example 2 (sketched) In the classification problem of this paper, Z is the instance space, m is the map that selects the "active" instances and t returns the region in Z where the classifier provides a correct classification. Details are provided later on in this section.

In what follows, R and S indicate two finite subsets of Z, both containing at least k points of Z. We consider three classes of pairs (R, S):

$$\mathcal{A} = \{(R, S) \text{ such that: } m(R) \text{ and } m(S) \text{ are defined, } R \subseteq S, \text{ and } S \subseteq t(m(R))\},$$

$$\mathcal{B} = \{(R, S) \text{ such that: } m(R) \text{ and } m(S) \text{ are defined, } R \subseteq S, \text{ and } m(S) = m(R)\},$$

$$\mathcal{C} = \{(R, S) \text{ such that: } m(R) \text{ and } m(S) \text{ are defined, } R \subseteq S, \text{ and } m(S) \subseteq R\}.$$

The following proposition establishes a link among $\mathcal{A}$, $\mathcal{B}$, and $\mathcal{C}$.

Proposition 1 *$\mathcal{A} = \mathcal{B}$ if and only if $\mathcal{A} = \mathcal{C}$.*

Example 3 (Example 1 continued) The reader may want to verify that $\mathcal{A} = \mathcal{B}$ and also $\mathcal{A} = \mathcal{C}$ in Example 1.

Proof of Proposition 1 (a) Suppose $\mathcal{A} = \mathcal{B}$.

To prove that $\mathcal{A} = \mathcal{B} \Rightarrow \mathcal{A} = \mathcal{C}$, we show in (a.1) that $\mathcal{A} \subseteq \mathcal{C}$ and in (a.2) that $\mathcal{C} \subseteq \mathcal{A}$.

(a.1) If $(R, S) \in \mathcal{A}$, then: $m(S) = [\text{since } \mathcal{A} = \mathcal{B}] = m(R) \subseteq R$, i.e. $(R, S) \in \mathcal{C}$.

(a.2) If $(R, S) \in \mathcal{C}$, then

$$m(S) \subseteq R. \tag{5}$$

Moreover,

$$S \subseteq t(m(S)), \tag{6}$$

as it follows from observing that $(S, S) \in \mathcal{B}$ so that, being $\mathcal{B} = \mathcal{A}$, $(S, S) \in \mathcal{A}$ and (6) follows. Thus,

$$R \subseteq S \subseteq [\text{use (6)}] \subseteq t(m(S)) = t(m(m(S))), \tag{7}$$

where the last equality is true since m extracts a subset of k points, so that $m(m(S))$ is the extraction of a subset of k points from $m(S)$ which is already a set of k points, and, hence, $m(m(S)) = m(S)$.

Equations (5) and (7) together imply that $(m(S), R) \in \mathcal{A}$ and, owing to that $\mathcal{A} = \mathcal{B}$, we obtain $(m(S), R) \in \mathcal{B}$, i.e.

$$m(m(S)) = m(R),$$

which in turn gives $m(S) = m(R)$, that is $(R, S) \in \mathcal{B}$. Since $\mathcal{B} = \mathcal{A}$, the thesis that $(R, S) \in \mathcal{A}$ is proven.

(b) Suppose now that $\mathcal{A} = \mathcal{C}$.

To prove that $\mathcal{A} = \mathcal{C} \Rightarrow \mathcal{A} = \mathcal{B}$, we show in (b.1) that $\mathcal{A} \subseteq \mathcal{B}$ and in (b.2) that $\mathcal{B} \subseteq \mathcal{A}$.

(b.1) If $(R, S) \in \mathcal{A}$, then $S \subseteq t(m(R)) = t(m(m(R)))$, so that $(m(R), S) \in \mathcal{A}$. But $\mathcal{A} = \mathcal{C}$ and, therefore, $(m(R), S) \in \mathcal{C}$ yielding $m(S) \subseteq m(R)$, from which $m(S) = m(R)$ since both sets contain k points. Whence, $(R, S) \in \mathcal{B}$.

(b.2) If $(R, S) \in \mathcal{B}$, then $m(S) = m(R) \subseteq R$ and we have that $(R, S) \in \mathcal{C} = \mathcal{A}$.

Summarizing, in (a) we have proven that $\mathcal{A} = \mathcal{B} \Rightarrow \mathcal{A} = \mathcal{C}$ and in (b) the opposite implication that $\mathcal{A} = \mathcal{C} \Rightarrow \mathcal{A} = \mathcal{B}$, so that the proposition statement that $\mathcal{A} = \mathcal{B}$ and $\mathcal{A} = \mathcal{C}$ are equivalent is established. □

We are now in a position to state the following fundamental result.

Theorem 2 *Consider a set Z and two maps m and t as above. Introduce a probability measure μ on Z according to which z_i points are independently extracted and suppose that*:

(i) *μ has no concentrated mass, i.e. $\mu(z) = 0, \forall z \in Z$;*
(ii) *for any fixed p, $m\{z_1, z_2, \ldots, z_p\}$ is defined except for at most an exceptional set with zero probability μ^p;*
(iii) *$\mathcal{A} = \mathcal{B}$.*

Given an integer $M \geq k$, consider the function $\xi : Z^M \to [0, 1]$ defined through: $\xi(z_1, z_2, \ldots, z_M) = \mu(t(m\{z_1, z_2, \ldots, z_M\}))$. Then, ξ has a Beta$(M + 1 - k, k)$ distribution, independently of the probability μ.

Example 4 (Example 3 continued) In the situation of Example 3, the theorem simply says that if points are extracted in $\mathbb{R}$ through a probability with no concentrated mass, then $\mu(t(m\{z_1, z_2, \ldots, z_M\})) = \mu(t\{z_{min}, z_{max}\}) = \mu[z_{min}, z_{max}]$, i.e. the mass inside the interval $[z_{min}, z_{max}]$, is a random variable (it is random through the random extractions of $z_1, z_2, \ldots, z_M$) that distributes according to a $Beta(M-1, 2)$.

Proof of Theorem 2 A $Beta(M+1-k, k)$ distribution has density $f_{Beta}(z) = \binom{M}{k-1}(M-k+1)z^{M-k}(1-z)^{k-1}$. Its moments are

$$\int_0^1 z^j f_{Beta}(z)\,\mathrm{d}z = \int_0^1 z^j \cdot \binom{M}{k-1}(M-k+1)z^{M-k}(1-z)^{k-1}\,\mathrm{d}z = \frac{\binom{M+j-k}{j}}{\binom{M+j}{M}}, \tag{8}$$

as it can be verified by integration by parts. We want to establish the result that the moments of ξ are indeed given by the right-hand-side of (8), viz.

$$E[\xi^j] = \frac{\binom{M+j-k}{j}}{\binom{M+j}{M}}, \quad j = 1, 2, \ldots, \tag{9}$$

which is enough to prove that ξ is distributed as a $Beta(M+1-k, k)$ since the distribution of a random variable with compact support (the support of ξ is [0, 1]) is completely determined by its moments (see e.g. Corollary 1, §12.9, Chap. II of Shiryaev 1996).

To prove (9), argue as follows: $\xi = \mu(t(m\{z_1, z_2, \ldots, z_M\}))$ is the probability that one more point z extracted according to probability μ falls in $t(m\{z_1, z_2, \ldots, z_M\})$, so that ξ^j is the probability that j more points independently extracted all fall in $t(m\{z_1, z_2, \ldots, z_M\})$, i.e.

$$\xi^j = \mu^j\big(z_{M+1}, \ldots, z_{M+j} \in t(m\{z_1, z_2, \ldots, z_M\})\big),$$

or, more compactly with the notation $z_m^n = \{z_m, z_{m+1}, \ldots, z_n\}$,

$$\xi^j = \mu^j\left(z_{M+1}^{M+j} \subseteq t(m(z_1^M))\right).$$

$E[\xi^j]$ is the mean of ξ^j when z_1^M is let vary according to the probability μ^M. Hence, using notation Z_m^n for the domain $Z \times \cdots \times Z$ in which z_m^n vary, we have

$$\begin{aligned}
E[\xi^j] &= \int_{Z_1^M} \xi^j\,\mathrm{d}\mu^M \\
&= \int_{Z_1^M} \mu^j\left(z_{M+1}^{M+j} \subseteq t(m(z_1^M))\right)\mathrm{d}\mu^M \\
&= [\mathbb{I}(A) = \text{indicator function of set } A] \\
&= \int_{Z_1^M}\left[\int_{Z_{M+1}^{M+j}} \mathbb{I}\left(z_{M+1}^{M+j} \subseteq t(m(z_1^M))\right)\mathrm{d}\mu^j\right]\mathrm{d}\mu^M \\
&= \int_{Z_1^{M+j}} \mathbb{I}\left(z_{M+1}^{M+j} \subseteq t(m(z_1^M))\right)\mathrm{d}\mu^{M+j}.
\end{aligned}$$

Now, let $I = \{i_1, \ldots, i_M\}$ be a generic subset of M indexes from $\{1, 2, \ldots, M+j\}$ and let $\mathcal{I}$ be the family of all possible choices of I ($\mathcal{I}$ contains $\binom{M+j}{M}$ elements). Moreover,

let $\bar{I} = \{1, 2, \ldots, M + j\} - I$. Since all z_i are extracted independently and with the same distribution, each group of M points z_i has identical statistical properties as any other group; therefore, if we indicate with z^I the set of points $z_i, i \in I$, we have that

$$\int_{Z_1^{M+j}} \mathbb{I}\left(z_{M+1}^{M+j} \subseteq t(m(z_1^M))\right) \mathrm{d}\mu^{M+j} = \int_{Z_1^{M+j}} \mathbb{I}\left(z^{\bar{I}} \subseteq t(m(z^I))\right) \mathrm{d}\mu^{M+j}, \quad \forall I \in \mathcal{I}.$$

Whence,

$$\begin{aligned} E[\xi^j] &= \int_{Z_1^{M+j}} \mathbb{I}\left(z_{M+1}^{M+j} \subseteq t(m(z_1^M))\right) \mathrm{d}\mu^{M+j} \\ &= \frac{1}{\binom{M+j}{M}} \sum_{I \in \mathcal{I}} \int_{Z_1^{M+j}} \mathbb{I}\left(z^{\bar{I}} \subseteq t(m(z^I))\right) \mathrm{d}\mu^{M+j} \\ &= \frac{1}{\binom{M+j}{M}} \int_{Z_1^{M+j}} \sum_{I \in \mathcal{I}} \mathbb{I}\left(z^{\bar{I}} \subseteq t(m(z^I))\right) \mathrm{d}\mu^{M+j}. \end{aligned} \tag{10}$$

The indicator function in the integrand requires that $z^{\bar{I}} \subseteq t(m(z^I))$. On the other hand, $(z^I, z^I) \in \mathcal{B}$ which, by the fact that $\mathcal{B} = \mathcal{A}$, yields $z^I \subseteq t(m(z^I))$. Therefore, we can add the indexes $i \in I$ in the indicator function without any change of the result and write

$$\mathbb{I}\left(z^{\bar{I}} \subseteq t(m(z^I))\right) = \mathbb{I}\left(z_1^{M+j} \subseteq t(m(z^I))\right).$$

Further invoking Proposition 1, we see that the theorem condition (iii) that $\mathcal{A} = \mathcal{B}$ implies that $\mathcal{A} = \mathcal{C}$, so a pair (R, S) is in $\mathcal{A}$ if and only if it is in $\mathcal{C}$. Thus, with $R = z^I$ and $S = z_1^{M+j}$, we have

$$\mathbb{I}\left(z_1^{M+j} \subseteq t(m(z^I))\right) = \mathbb{I}\left(m(z_1^{M+j}) \subseteq z^I\right).$$

Substituting in turn the last two equations in (10), we come to the result that

$$E[\xi^j] = \frac{1}{\binom{M+j}{M}} \int_{Z_1^{M+j}} \sum_{I \in \mathcal{I}} \mathbb{I}\left(m(z_1^{M+j}) \subseteq z^I\right) \mathrm{d}\mu^{M+j}. \tag{11}$$

To complete the computation of $E[\xi^j]$, observe now that points $z_1, z_2, \ldots, z_{M+j}$ are with probability 1 all distinct since $\mu(z) = 0$, $\forall z \in Z$ (assumption (i)), and therefore the sum under the integral amounts to compute the number of times that a subset z^I of M points from a set z_1^{M+j} of $M + j$ distinct points contains a fixed set $m(z_1^{M+j})$ of k points in the set z_1^{M+j}. Since these k points are fixed, we have to decide which j points among the remaining $M + j - k$ in z_1^{M+j} have to be left out of z^I, and the number of possible different choices is $\binom{M+j-k}{j}$. Substituting in (11) yields (9), and this completes the proof. □

We shall now prove Theorem 1 as an application of Theorem 2.

We start by considering the ideal GEM algorithm.

Throughout, (x_1, y_1) is regarded as a given initial example which we suppose fixed, and derive the distribution of $PE(\hat{y}_N)$ as a function of the remaining $N - 1$ examples $(x_2, y_2), \ldots, (x_N, y_N)$. In more technical terms, we derive the conditional distribution of $PE(\hat{y}_N)$ given (x_1, y_1). This distribution turns out to be a $Beta(k, N - k)$ regardless of (x_1, y_1), so that, by integration with respect to (x_1, y_1), the distribution of $PE(\hat{y}_N)$ is proven to be a $Beta(k, N - k)$.

The data generation mechanism $(\mu, y(\cdot))$ has to be thought of as given and fixed throughout (even though it is unknown). Therefore an example $(x_i, y_i) = (x_i, y(x_i))$ is determined once x_i has been assigned.

For easy reference, we notice that Theorem 2 will be applied to establish Theorem 1 with the following positions:

- $Z = \mathbb{R}^d$, the instance space;
- the probability μ of Theorem 2 is the probability μ with which x_i instances are observed. Note that condition (i) of Theorem 2 is satisfied because μ in Theorem 1 has density;
- $z_i = x_{i+1}$, $i = 1, 2, \ldots$ (remember that x_1 plays a special role of "initial" instance);
- for any $p \geq k$, suppose we apply the ideal GEM algorithm with $N = p + 1$. Then, upon termination, the algorithm has stored in Q at least k indexes taken from $\{2, 3, \ldots, p + 1\}$ and, whenever they are exactly k, we define m through relation $m\{x_2, x_3, \ldots, x_{p+1}\} = \{x_i, i \in Q\}$. Later, we shall prove that Q contains exactly k indexes with probability 1, that is condition (ii) in Theorem 2 is fulfilled;
- given k examples with instances $x_{i_1}, x_{i_2}, \ldots, x_{i_k}$ (besides the initial example (x_1, y_1)), $t(x_{i_1}, x_{i_2}, \ldots, x_{i_k})$ is by definition the set in $\mathbb{R}^d$ that is correctly classified or classified as *unknown* by the classifier constructed by the ideal GEM algorithm applied to $x_{i_1}, x_{i_2}, \ldots, x_{i_k}$;
- $M = N - 1$.

By these positions, we have that $PE(\hat{y}_N) = 1 - \mu(t(m\{x_2, x_3, \ldots, x_N\}))$, which simply means that the probability of error of the classifier is 1 minus the probability μ of the region where it correctly classifies, or classifies as *unknown*. If we prove the applicability of Theorem 2, then $\mu(t(m\{x_2, x_3, \ldots, x_N\}))$ distributes as a $Beta(N - k, k)$ so that the complementary variable $PE(\hat{y}_N)$ distributes as a $Beta(k, N - k)$ and the proof for the ideal GEM algorithm is complete. We therefore see that all that remains to be proven is that conditions (ii) and (iii) of Theorem 2 hold in our present context of application.

To prove (ii), we start off by showing the validity of the following property

(P) the optimization program in point 2a of the GEM algorithm returns the same solution if we remove all the constraints $(x_i - x_B)^T A(x_i - x_B) + b^T(x_i - x_B) \geq 1$, $i \in P$ such that $y_i \neq y(x_B)$, but at most $(d(d + 1)/2 + d)$ of them, suitably selected.

This property is a consequence of the fact that this program is a convex program in $(d(d + 1)/2 + d)$ variables, where $d(d + 1)/2$ are the free parameters of A (remember that A is symmetric) and d are the free parameters of b. To formally show property (P), assume for the sake of contradiction that no choice of at most $(d(d + 1)/2 + d)$ constraints exists that maintains the solution unaltered. For any given $i \in P$ such that $y_i \neq y(x_B)$, consider the set H_i of (A, b), $A \succeq 0$, pairs that satisfy the associated constraint:

$$H_i = \left\{(A, b) \text{ such that } (x_i - x_B)^T A(x_i - x_B) + b^T(x_i - x_B) \geq 1\right\}.$$

Also consider one more set H_0 of (A, b), $A \succeq 0$, pairs defined as

$$H_0 = \{(A, b) \text{ that outperforms the solution of the program in point 2a}\}.$$

In this latter definition "outperforms" means that the $Trace(A)$ is lower than $Trace(A^*)$, or, for equal $Trace$, b has lower norm than b^*, or, for equal b norm, the lexicographic rule favors (A, b) over (A^*, b^*). An easy inspection shows that all these sets are convex, i.e. if (A_1, b_1) and (A_2, b_2) belong to one set, also $\alpha(A_1, b_1) + (1 - \alpha)(A_2, b_2)$, $\alpha \in [0, 1]$, belong to the

same set. Moreover, the intersection of any $(d(d+1)/2+d)+1$ of these sets is always non-empty. Indeed, the intersection of $(d(d+1)/2+d)+1$ sets H_i, $i \neq 0$, is clearly non-empty since an (A, b) with $b = 0$ and $A = aI$ with a large enough belongs to the intersection. If instead H_0 is in the group of $(d(d+1)/2+d)+1$ sets, then there are only $(d(d+1)/2+d)$ sets of the type H_i, $i \neq 0$, but then, by the assumption made for the sake of contradiction, the solution of the program with only the constraints associated with these $(d(d+1)/2+d)$ sets will be different from the solution of the whole program, and therefore this solution will outperform the solution of the whole program. This means that this solution is also in H_0, proving that the intersection of all the $(d(d+1)/2+d)+1$ sets is non-empty in this case too. Now, resorting to Helly's theorem (see e.g. Rockafellar 1970) yields that the intersection of all the sets, viz. $\cap_{i\in P, y_i\neq y(x_B)} H_i \cap H_0$, is non-empty. This last relation means that we can find a pair (A^{**}, b^{**}) which is simultaneously in all the sets H_i, $i \in P$, $y_i \neq y(x_B)$, and therefore it satisfies all constraints $(x_i - x_B)^T A(x_i - x_B) + b^T(x_i - x_B) \geq 1$, and that is also in H_0, and therefore it outperforms (A^*, b^*). But (A^*, b^*) is the optimal solution, and this is a contradiction. Hence, we have proven (P).

Based on (P), we prove now property (ii). Suppose first that the algorithm last executes, among 2a, 2b, and 2c, point 2a before termination. Since point 2a is entered only when $|Q| \leq k - (d(d+1)/2+d)$, having $|Q| > k$ after executing point 2a implies that the algorithm has found more than $(d(d+1)/2+d)$ active instances while executing point 2a. But we have shown in (P) that at most $(d(d+1)/2+d)$ constraints determine the solution and, to have more than $(d(d+1)/2+d)$ active instances, at least one of the other instances must fall exactly on the boundary of the region generated by the at most $(d(d+1)/2+d)$ instances that determine the solution, a fact that happens only with probability 0 since μ has density.

A similar rationale permits one to conclude that $|Q| > k$ only happens with probability 0 when the algorithm last executes points 2b or 2c. We have therefore proven that $|Q| = k$ with probability 1 upon termination of the algorithm and condition (ii) is established.

We finally prove condition (iii), i.e. that $\mathcal{A} = \mathcal{B}$.

Suppose that a set of instances S has a subset R, that $m(S)$ and $m(R)$ are defined, and that S is in $t(m(R))$, i.e. $(R, S) \in \mathcal{A}$. This means that the examples whose instances are in $S - R$ are correctly classified, or classified as *unknown*, by the classifier constructed with the examples with instances in R. If so, an easy inspection of the GEM algorithm reveals that $m(S) = m(R)$, so that $(R, S) \in \mathcal{B}$. Vice versa, suppose $(R, S) \in \mathcal{B}$. The region of correct classification, or classified as *unknown*, for the classifier generated by the examples with instances in S contains S itself since the ideal GEM algorithm does not misclassify any example. Thus,

$$
\begin{aligned}
S \subseteq\ & \text{region of correct classification, or classified as } \textit{unknown}, \\
& \quad \text{for the classifier generated by the examples with instances in } S \\
=\ & t(m(S)) \\
=\ & [\text{since } (R, S) \in \mathcal{B} \text{ implies } m(S) = m(R)] \\
=\ & t(m(R)),
\end{aligned}
$$

and it remains proven that $(R, S) \in \mathcal{A}$. We have therefore proven condition (iii) that $\mathcal{A} = \mathcal{B}$, and this completes the proof for the ideal GEM algorithm.

The result that the probability distribution of $PE(\hat{y}_N)$ for the ideal GEM algorithm dominates that for the real GEM algorithm immediately follows from observing that $PE(\hat{y}_N)$ is, by construction, always no bigger for the latter than for the former.

Acknowledgements This research was supported by MIUR (Ministero dell'Istruzione, dell'Universita' e della Ricerca). I feel indebted with two anonymous reviewers for insightful comments that helped improve this manuscript. A special thank goes to Dr. A. Care' for performing the empirical experiments of Sect. 4. An openly distributed software that implements GEM is available at http://bsing.ing.unibs.it/~algo.care/GEM/.

References

Anthony, M. (2004). Generalization error bounds for threshold decision lists. *Journal of Machine Learning Research*, *5*, 189–217.

Asuncion, A., & Newman, D. J. (2007). *UCI machine learning repository*. University of California, Irvine, School of Information and Computer Sciences. URL: http://www.ics.uci.edu/~mlearn/MLRepository.html.

Bartlett, P. L., & Wegkamp, M. H. (2008). Classification with a reject option using a hinge loss. *Journal of Machine Learning Research*, *9*, 1823–1840.

Chernoff, H. (1952). A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *Annals of Mathematical Statistics*, *23*, 493–507.

Chow, C. K. (1957). An optimum character recognition system using decision functions. *IRE Transactions on Electronic Computers*, *6*, 247–254.

Chow, C. K. (1970). On optimum recognition error and reject tradeoff. *IEEE Transactions on Information Theory*, *16*, 41–46.

Cristianini, N., & Shawe-Taylor, J. (2000). *An introduction to support vector machines and other kernel-based learning methods*. Cambridge: Cambridge University Press.

Devroye, L. (1982). Necessary and sufficient conditions for the almost everywhere convergence of nearest neighbor regression function estimates. *Zeitschrift fur Wahrscheinlichkeitstheorie und Verwandte-gebiete*, *61*, 467–481.

Devroye, L., Gyorfi, L., & Lugosi, G. (1996). *A probabilistic theory of pattern recognition*. New York: Springer.

Ferri, C., Hernandez-Orallo, J., & Salido, M. A. (2003). Volume under the roc surface for multi-class problems. In *14th European conference on machine learning* (pp. 108–120).

Fraser, D. A. S. (1951). Sequentially determined statistically equivalent blocks. *Annals of Mathematical Statistics*, *22*, 372–381.

Fraser, D. A. S. (1953). Nonparametric tolerance regions. *Annals of Mathematical Statistics*, *24*, 44–55.

Fraser, D. A. S. (1957). *Nonparametric methods in statistics*. New York: Wiley.

Fraser, D. A. S., & Guttman, I. (1956). Tolerance regions. *Annals of Mathematical Statistics*, *27*, 162–179.

Freund, Y., Mansour, Y., & Schapire, R. E. (2004). Generalization bounds for averaged classifiers. *Annals of Statistics*, *32*(4), 1698–1722.

Gyorfi, L., Gyorfi, Z., & Vajda, I. (1978). Bayesian decision with rejection. *Problems of Control and Information Theory*, *8*, 445–452.

Herbei, R., & Wegkamp, M. H. (2006). Classifcation with reject option. *Canadian Journal of Statistics*, *34*(4), 709–721.

Hussain, Z., Laviolette, F., Marchand, M., Shawe-Taylor, J., Brubaker, S. C., & Mullin, M. D. (2007). Revised loss bounds for the set covering machine and sample-compression loss bounds for imbalanced data. *Journal of Machine Learning Research*, *8*, 2533–2549.

Kemperman, J. H. B. (1956). Generalized tolerance limits. *Annals of Mathematical Statistics*, *27*, 180–186.

Klivans, A. R., & Servedio, R. A. (2006). Toward attribute efficient learning of decision lists and parities. *Journal of Machine Learning Research*, *7*, 587–602.

Marchand, M., & Shawe-Taylor, J. (2002). The set covering machine. *Journal of Machine Learning Research*, *3*, 723–746.

Marchand, M., & Sokolova, M. (2005). Learning with decision lists of data-dependent features. *Journal of Machine Learning Research*, *6*, 427–451.

Muzzolini, R., Yang, Y. H., & Pierson, R. (1998). Classifier design with incomplete knowledge. *Journal of American Statistical Association*, *31*, 345–369.

Pietraszek, T. (2005). Optimizing abstaining classifiers using roc analysis. In *22nd European conference on machine learning* (pp. 665–672).

Rivest, R. L., & Sloan, R. (1988). Learning complicated concepts reliably and usefully. In *Proceedings of the 1st annual workshop on computational learning theory* (pp. 69–79), San Mateo, CA, USA.

Rockafellar, R. T. (1970). *Convex analysis*. Princeton: Princeton University Press.

Ruckert, U., & Kramer, S. (2004). Towards tight bounds for rule learning. In *Proceedings of the 21st international conference on machine learning*, Banff, Alberta, Canada.

Schervish, M. J. (1995). *Theory of statistics*. New York: Springer.
Schölkopf, B., & Smola, A. J. (2002). *Learning with kernels*. Cambridge: MIT Press.
Shafer, G., & Vovk, V. (2008). A tutorial on conformal prediction. *Journal of Machine Learning Research*, *9*, 371–421.
Shiryaev, A. N. (1996). *Probability*. New York: Springer.
Tukey, J. W. (1947). Nonparametric estimation II. Statistically equivalent blocks and tolerance regions—the continuous case. *Annals of Mathematical Statistics*, *18*, 529–539.
Vapnik, V. N. (1995). *The nature of statistical learning theory*. New York: Springer.
Vapnik, V. N. (1998). *Statistical learning theory*. New York: Wiley.
Vidyasagar, M. (1997). *Theory of learning and generalization: with applications to neural networks and control systems*. London: Springer.
Vovk, V., Gammerman, A., & Shafer, G. (2005). *Algorithmic learning in a random world*. New York: Springer.
Wald, A. (1943). An extension of Wilks' method for setting tolerance limits. *Annals of Mathematical Statistics*, *14*, 45–55.
Wilks, S. S. (1941). Determination of sample sizes for setting tolerance limits. *Annals of Mathematical Statistics*, *12*, 91–96.