A majority voting classifier with probabilistic guarantees

Giorgio Manganini, Alessandro Falsone, Maria Prandini

***Abstract*— This paper deals with supervised learning for classification. A new general purpose classifier is proposed that builds upon the Guaranteed Error Machine (GEM). Standard GEM can be tuned to guarantee a desired (small) misclassification probability and this is achieved by letting the classifier return an *unknown* label. In the proposed classifier, the size of the unknown classification region is reduced by introducing a majority voting mechanism over multiple GEMs. At the same time, the possibility of tuning the misclassification probability is retained. The effectiveness of the proposed majority voting classifier is shown on both synthetic and real benchmark data-sets, and the results are compared with other well-established classification algorithms.**

I. Introduction

Machine Learning (ML) techniques, see e.g. [1]–[5], aim at designing automatic procedures to make accurate prediction of future instances of some phenomenon based on a set of observations (the training data–set). To this end, every instance in the data–set is represented by a set of features. When instances in the training data–set are provided together with the corresponding output, and the ML technique exploits this information, then we are dealing with supervised learning. This work is concerned with supervised learning in the context of classification, where the output takes values in a finite set.

A classifier is a map associating to the features (input) some label (output). The goal of a supervised learning algorithm is to construct a classifier that provides highly accurate predictions when applied to new unseen instances, minimizing the number of misclassification (generalization error).

Various techniques have been developed for supervised classification within the fields of artificial intelligence (logical/symbolic techniques, like Decision trees [6] or learning set of rules [7]), neural–networks and statistics (Bayesian Networks [8], NNC [3], SVM [4] and SCM [9]). The interested reader is referred to [10] for an overview. The application domains of supervised classification include text categorization, fraud detection, machine vision, natural-language processing, and bio–informatics to name a few (see e.g. [11] and [12] for application–oriented papers).

In this work, we focus on a general purpose classifier, called Guaranteed Error Machine (GEM) [13]. A comparative discussion on alternative approaches proposed in the literature goes beyond the scope of this paper, and the interested reader is referred to [13]. The supervised learning algorithm that builds a GEM classifier processes the training data–set progressively, starting from one of the training data named the base sample. The main property of the GEM classifier is that, under general conditions, the statistics of its generalization error are universal, i.e., independent of the (unknown) mechanism that generates the data. This property allows the user to select a desired maximum level of generalization error and let the learning machine automatically adjust to meet it. In order to control the probability of misclassification, the GEM is allowed to return an $unknown$ label, expressing doubt on which label should be associated to a particular instance. The requirement on the accuracy level is actually met by modulating the size of the region where the machine returns the $unknown$ label.

Our main contribution consists in introducing a supervised learning algorithm for shrinking the unknown classification region as much as possible, while retaining some probabilistic guarantees on the generalization error of the resulting new GEM-based classifier. The idea is as simple as follows: building multiple GEMs based on the same training data–set by starting from different base samples, and then assigning to each new data the label that is voted by most GEMs, except for the $unknown$ label that is assigned only if all GEMs vote for it. Indeed, this reduces the region with the $unknown$ label by construction. As for the generalization error properties of the majority voting GEM, a conservative bound can be derived based on that of the standard GEM. In practice, numerical examples on both artificially generated and real data show that the generalization error is much lower than the a-priori bound, and that the majority voting GEM outperforms standard GEM also in that respect.

The rest of the paper is organized as follows. We start revising briefly the GEM classifier and its generalization properties in Section II. We then propose the new majority voting classifier and discuss its properties in Section III. In Section IV we present numerical examples assessing the performance of the majority voting classifier. Finally, in Section V some concluding remarks are drawn and possible extensions are described.

II. The GEM classifier

In this section we revise the GEM supervised learning algorithm proposed in [13] and introduce some basic notations and definitions.

Let $x \in \mathcal{X} \subseteq \mathbb{R}^d$ be a vector of features and $y = y(x) \in Y = \{0, 1\}$ the corresponding binary label. A classifier $\hat{y} = \hat{y}(x)$ provides an estimate for the label y of x and

This work is partially supported by the European Commission under the project UnCoVerCPS with grant number 643921.

Giorgio Manganini, Alessandro Falsone and Maria Prandini are with the Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, via Ponzio 34/5, 20133 Milano, Italy. {giorgio.manganini, alessandro.falsone, maria.prandini}@polimi.it

it errs on x if $y(x) \neq \hat{y}(x)$. Differently from most other classifiers, the GEM one may return an $unknown$ label expressing the inability to classify the sample, so that the label set is augmented to $Y \cup \{unknown\}$.

Now, let $\mathcal{E}_N = \{(x_1, y_1), \ldots, (x_N, y_N)\}$ be a data–set of N training samples, where $x_1, \ldots, x_N$ are independently extracted from $\mathcal{X}$ according to some probability distribution μ and $y_i = y(x_i)$. We denote by $\hat{y}_N(\cdot)$ the GEM classifier trained on these data.

The algorithm to build the GEM classifier takes as input the training data–set $\mathcal{E}_N$, a "base" instance $(x_B, y_B) \in \mathcal{E}_N$, and some integer parameter $k < N$. It then constructs a set of hyper-ellipsoidal regions $\mathcal{R} = \{\mathcal{R}_1, \ldots, \mathcal{R}_r\}$. Each region $\mathcal{R}_j$ has associated a label $\ell_j \in Y$. The union of the regions in $\mathcal{R}$ constitutes that part of the input space $\mathcal{X}$ where the GEM classifier assigns a proper label, whereas in the remaining (uncovered) part of the space the machine returns the label $unknown$. More precisely, the GEM classifier is defined as

$$\hat{y}_N(x) = \begin{cases} unknown & \text{if } x \in \mathcal{X} \setminus \bigcup_{j=1}^{r} \mathcal{R}_j \\ \ell_{q(x)} & \text{if } x \in \bigcup_{j=1}^{r} \mathcal{R}_j \end{cases}, \qquad (1)$$

where

$$q(x) = \min\{j : \text{ such that } x \in \mathcal{R}_j, j = 1, \ldots, r\}.$$

As for the construction of the regions in $\mathcal{R}$, starting from the base instance (x_B, y_B), the algorithm constructs an hyper-ellipsoidal set $\mathcal{R}_1 \subseteq \mathcal{X}$, that contains x_B and extends it until it touches another example $x_j \neq x_B$, with a different label $y_j \neq y_B$. All training samples included into $\mathcal{R}_1$ have the same label $\ell_1 = y_B$ of x_B and are removed from the training data–set, while the instances $(x, y) \in \mathcal{E}_N$ with x on the boundary $\Omega(\mathcal{R}_1)$ of the region $\mathcal{R}_1$ are marked as "active" points and added to some set Q. If the cardinality of Q satisfies $|Q| < k$ and the updated training data–set is not empty, then the active point farthest from x_B is selected to form the new base instance, and a new region is constructed. Parameter k acts on the size of the $unknown$ region, where a classification cannot be provided. Roughly speaking, larger values for k correspond to classifiers with smaller $unknown$ regions, but more prone to misclassification. Ensuring that $|Q| \leq k$ is the key property to prove the generalization properties of the GEM classifier in [13].

A flow diagram of the algorithm is shown in Figure 1. Note that, depending on the cardinality of set Q a different convex optimization problem is solved to construct the region $\mathcal{R}_j$ (parametrized by A_j^* and b_j^*). More specifically, problem (A) constructs hyper-ellipsoids containing x_B, problem (B) determines hyper-spheres containing x_B, and problem (C) finds hyper-spheres centered in x_B. For more details on the GEM algorithm the reader is referred to [13].

The *probability of error* (or generalization error) of the GEM classifier $\hat{y}_N(\cdot)$ in (1) is defined as

$$PE(\hat{y}_N) = \mu\left(\hat{y}_N(x) \in Y \ \wedge \ y(x) \neq \hat{y}_N(x)\right),$$

which is the probability that a proper label in Y is issued and that label is not correct. Note that the $unknown$ label is not considered, and hence it is not counted as an error when returned by the classifier.

Given that $\hat{y}_N(\cdot)$ is derived based on the set $\mathcal{E}_N$ of randomly sampled training data, $PE(\hat{y}_N)$ is a random variable that depends on the data generation mechanism $\{\mu, y(\cdot)\}$. Interestingly, Theorem 1 in [13] provides a strict bound on the probability distribution of $PE(\hat{y}_N)$, which is independent on the data generation mechanism $\{\mu, y(\cdot)\}$. Specifically, let

$$F_{PE}(\epsilon) := \mu^N \{PE(\hat{y}_N) \leq \epsilon\}$$

be the probability that $PE(\hat{y}_N) \leq \epsilon$. Then, we have that

$$F_{PE}(\epsilon) \geq 1 - \sum_{i=0}^{k-1} \binom{N-1}{i} \epsilon^i (1-\epsilon)^{N-1-i}.$$

If the training data–set size N can be selected arbitrarily, then, one can choose the confidence parameter $\delta \in (0, 1)$ and determine N satisfying

$$\sum_{i=0}^{k-1} \binom{N-1}{i} \epsilon^i (1-\epsilon)^{N-1-i} \leq \delta, \qquad (2)$$

so that the misclassification probability $PE(\hat{y}_N)$ is guaranteed not to exceed ϵ with confidence not smaller than $1 - \delta$. Following [14], we can compute the following lower bound on the number of training data N that are needed for expression (2) to hold, as a function of ϵ, δ, and k:

$$N \geq 1 + \frac{1}{\epsilon}\left(k - 1 + \log\frac{1}{\delta} + \sqrt{2(k-1)\log\frac{1}{\delta}}.\right). \qquad (3)$$

The GEM classifier $\hat{y}_N(\cdot)$ depends on the training data–set $\mathcal{E}_N$ and on the sample selected as base instance. As suggested in [13], one can run the algorithm in Figure 1 multiple times starting from different base instances, and then select the best classifier (*i.e.*, the one with a smaller $unknown$ region) *a posteriori*. The probability that all the resulting M classifiers have a probability of error smaller than or equal to ϵ can be bounded as follows:

$$\begin{aligned} &\mu^N\left\{PE(\hat{y}_N^{(1)}) \leq \epsilon \wedge \cdots \wedge PE(\hat{y}_N^{(M)}) \leq \epsilon\right\} \\ &= 1 - \mu^N\left\{PE(\hat{y}_N^{(1)}) > \epsilon \vee \cdots \vee PE(\hat{y}_N^{(M)}) > \epsilon\right\} \\ &\geq 1 - \sum_{i=1}^{M} \mu^N\left\{PE(\hat{y}_N^{(M)}) > \epsilon\right\} \\ &\geq 1 - M\sum_{i=0}^{k-1} \binom{N-1}{i} \epsilon^i (1-\epsilon)^{N-1-i}. \end{aligned} \qquad (4)$$

The size N of the training data–set can then be determined based on (3) with δ replaced with $M\delta$.

III. Majority voting classifier

Voting denotes the simplest method of combining multiple classifiers [15]. In its simplest form, called plurality or majority voting, each classifier contributes with a single vote [16]. The labeling mechanism is decided by the majority of the votes, i.e., the label with the most votes is the final

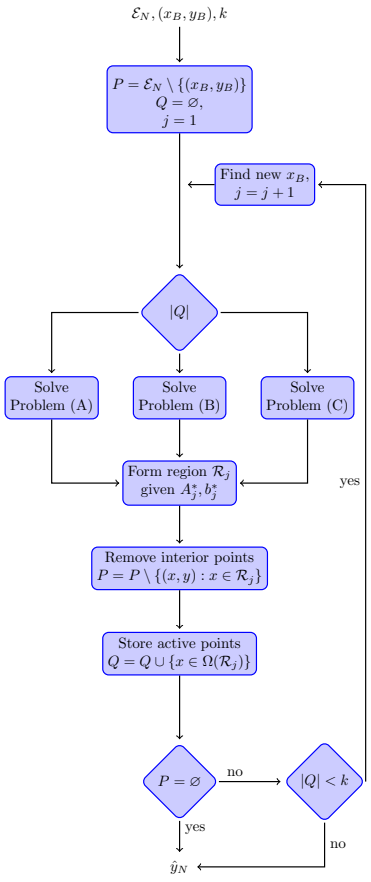

Fig. 1. Flow diagram of the algorithm to construct a GEM classifier.

one. In the following we propose a novel classifier which involves multiple GEMs and incorporate a majority voting mechanism.

Suppose to design M GEMs by running M times the algorithm in Figure 1 from different base instances (x_B, y_B) in the same training dataset $\mathcal{E}_N$. Let $\ell^{(m)}(x) = \hat{y}_N^{(m)}(x)$ be the label associated to sample $x \in \mathcal{X}$ by the m–th GEM, with $m = 1, \ldots, M$. Given a label $c \in Y$, we define $\mathbf{1}_c(\ell)$ to be the indicator function, i.e.,

$$\mathbf{1}_c(\ell) = \begin{cases} 1, & \text{if } \ell = c \\ 0, & \text{otherwise} \end{cases},$$

for any $\ell \in Y \cup \{unknown\}$. A simple majority voting mechanism [16] prescribes to select the label for a sample $x \in \mathcal{X}$ according to

$$c^\star(x) = \arg\max_{c \in Y} \sum_{m=1}^{M} \mathbf{1}_c(\ell^{(m)}(x)),$$

with the understanding that if there is more than one maximizer, the smallest one is chosen.

The proposed majority voting classifier is then given by

$$\hat{y}_{N,M}(x) = \begin{cases} unknown, & \text{if } \ell^{(m)}(x) = unknown, \\ & \forall m \in \{1, \ldots, M\}, \\ c^\star(x), & \text{otherwise.} \end{cases} \quad (5)$$

and returns the label in Y that is most voted, and the unknown label only when all GEMs return unknown.
The unknown classification region of the majority voting classifier is thus the intersection of the unknown classification regions of all M GEMs and as such it is reduced.

Proposition 3.1 below shows the probabilistic guarantees of the majority voting classifier.

Proposition 3.1: The majority voting classifier (5) constructed using M GEMs, each one with probability of error bounded by ϵ with confidence not smaller than $1 - \delta$, has a probability of error bounded by $M\epsilon$ with confidence not smaller than $1 - M\delta$. Formally, if

$$\mu^N \left\{ PE(\hat{y}_N^{(m)}) \leq \epsilon \right\} \geq 1 - \delta, \ m = 1, \ldots, M,$$

then

$$\mu^N \left\{ PE(\hat{y}_{N,M}) \leq M\epsilon \right\} \geq 1 - M\delta. \quad (6)$$

The proof of Proposition 3.1 straightforward given that the probability of error of the majority voting classifier is upper bounded by the probability that at least one GEM makes an error. In turn, this probability is smaller than or equal to $M\epsilon$ in the case when all M GEMs have a probability of error of at most ϵ, which happens with confidence larger than or equal to $1 - M\delta$ (see (4)).

Note that the bound (6) on the misclassification probability depends on the number M of GEMs used to construct the majority voting classifier through a scaling factor on the confidence parameter and on the probability of error parameter as well. Hence, to guarantee a-priori that the probability of error of the majority voting classifier is upper bounded by $\bar{\epsilon}$ with confidence $1 - \bar{\delta}$, one has to replace δ with $\bar{\delta}/M$ and ϵ with $\bar{\epsilon}/M$ when calculating N through (3). In Section IV we report some numerical examples that show the effectiveness of the proposed majority voting classifier in shrinking the $unknown$ region. These examples also reveal that the bound in (6) is deem over conservative and one can in fact use $\bar{\epsilon}$ in place of $\bar{\epsilon}/M$ when calculating N through (3) in order to have that the actual probability of error is lower than or equal to $\bar{\epsilon}$, like in the single GEM.

We next describe how to choose M so as to obtain a majority voting classifier that is structurally guaranteed to never provide $unknown$ as a label for the data in the training data–set.
We start by setting $M = 1$ and constructing a GEM starting from a random data point in $\mathcal{E}_N$ as base instance. Then, we construct the set of $unknowns$ as

$$\mathcal{U}(M) = \{(x_i, y_i) \in \mathcal{E}_N : \hat{y}_{N,M}(x_i) = unknown\}.$$

If $\mathcal{U}(M)$ is not empty, then we increase M by one and construct an additional GEM instance to be incorporated

into the majority voting classifier. The procedure is repeated iteratively until $\mathcal{U}(M)$ is empty. The obtained value for M is denoted as $M^\star$ and the corresponding classifier as $M^\star$-GEM majority voting classifier. The whole procedure is summarized in Algorithm 1.

Algorithm 1 $M^\star$–GEM majority voting classifier

Input: Training data–set $\mathcal{E}_N = \{(x_i; y_i) | x_i \sim \mu\}_{i=1}^N$

1: $M \leftarrow 0$
2: $\mathcal{U}(M) \leftarrow \mathcal{E}_N$
3: **repeat**
4: $\quad (\bar{x}; \bar{y}) \leftarrow$ an element from $\mathcal{U}(M)$
5: $\quad M \leftarrow M + 1$
6: $\quad \hat{y}_N^{(M)}(\cdot) \leftarrow$ GEM instance starting from $(\bar{x}; \bar{y})$
7: $\quad \mathcal{U}(M) \leftarrow \{(x_i, y_i) \in \mathcal{E}_N : \hat{y}_{N,M}(x_i) = unknown\}$
8: **until** $\mathcal{U}(M) = \varnothing$
9: $M^\star = M$

Output: $\hat{y}_{N,M^\star}(\cdot)$

The $M^\star$–GEM majority voting classifier guarantees by construction zero unknowns over the data–set. In order to determine N, one would need to know *a-priori* what the value of $M^\star$ will be. Given that the worst case scenario consists in having $M^\star = N$, the confidence level for all GEM instances can be set to $1 - \bar{\delta}/N$ to ensure a confidence level $1 - \bar{\delta}$ for the $M^\star$–GEM majority voting classifier.

IV. Numerical examples

This section provides empirical evidence of the effectiveness of the proposed classifier and of the theoretical results in Section III. To this aim, we use two types of data–sets. The first one is obtained artificially from $(\mu, y(\cdot))$, where μ is a uniform distribution on $[0, 1]^n$, and $y(\cdot)$ is given by:

$$y(x) = \begin{cases} 1 & \text{if } 1 - \frac{\sqrt{2}}{2} \leq x_i \leq 1 \quad \forall i = 1 \dots n \\ 0 & \text{otherwise} \end{cases}, \qquad (7)$$

where x_i is the i-th component of x. Figure 2 shows $N = 1000$ samples extracted from μ for the case $n = 2$.

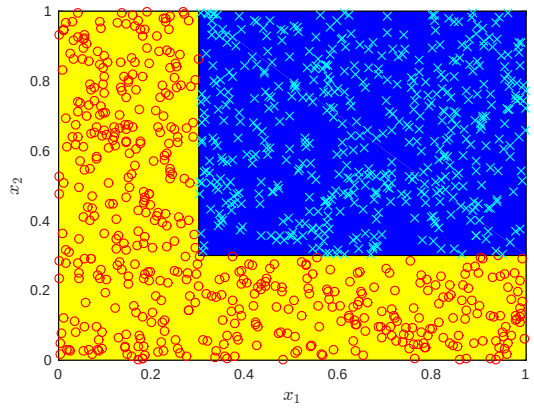

Fig. 2. Synthetic training data–set.

Then, additional empirical results are computed on real and publicly available data–set: Glass, BreastW, Haberman, Pima, Bupa and Credit. All these data–sets are obtained from the machine learning repository at UCI [17], except for the Glass data–set which can be downloaded from [18]. For each data–set, we removed all samples with missing features or contradictory labels (this occurred only for a few samples in the Haberman data–set).

A. Synthetic data–set

In Figure 3 we report the number of $unknown$ obtained with the GEM and the majority voting classifier using M GEMs. In both cases we fixed $N = 1000$ and a confidence $1 - \delta$ with $\delta = 10^{-5}$. We can clearly see from Figure 3, that the number of $unknowns$ decreases as a function of k when we increase the number M of GEMs in the majority voting classifier.

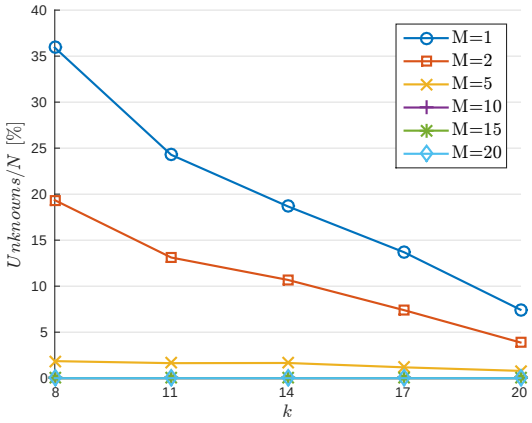

Fig. 3. Percentage of *unknowns* as a function of k and M.)

Note that for a given k the number of $unknowns$ decreases with M. Though this effect is an obvious consequence of using multiple GEMs and the same result could be achieved increasing the value of k for the original GEM, the impact on the value of the misclassification probability is different. As a matter of fact, by looking at Figure 4, one can see that as M increases not only the number of $unknowns$ but also the empirical error probability decreases in the M-GEM classifier. Given that each of the M GEMs has a theoretical misclassification level ϵ, Figure 4 provides empirical evidence that the bound in Proposition 3.1 is conservative, and that, by increasing M, we also gain in accuracy and not only in a reduction of the $unknown$ region.

Figure 5 plots the number of $unknowns$ obtained with the standard GEM and the majority voting classifier using M GEMs when the dimensionality of the problem n is increased. Classifiers are constructed by setting $N = 1000$, $\delta = 10^{-5}/M$, and $\epsilon = 6\%$ for each GEM. Note that standard GEM is obtained by setting $M = 1$. Clearly both classifiers suffer the growth in dimensionality, but the impact on the number of unknowns returned gets lower and lower as we increase the value of M.

We also compare the performance of the majority voting classifier using M–GEMs with the solution proposed in (4) which prescribes to create a certain number of GEMs starting

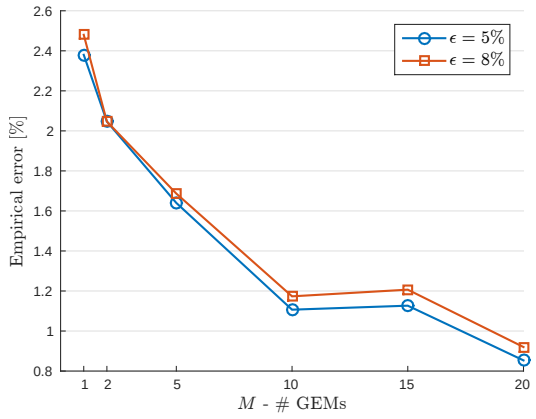

Fig. 4. Empirical error in percentage as a function of M and ϵ.

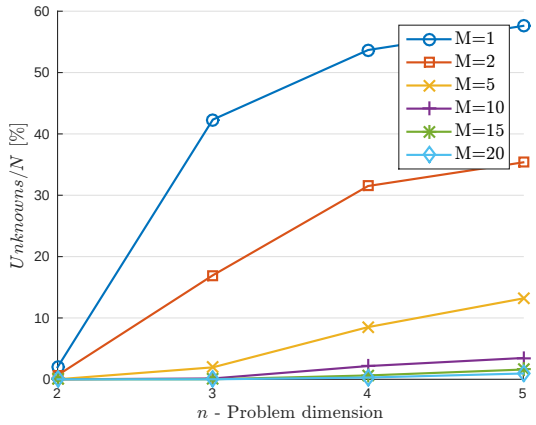

Fig. 5. Number of *unknowns* in percentage as a function of problem dimensionality and M.

from different base instances and then use the one with the smallest *unknown* region. For a given value of M, we construct M GEMs using $\delta/M = 10^{-5}/M$ as confidence parameter, an $\epsilon = 8\%$ for the accuracy parameter. For each value of M we run 30 trials starting from different M-tuples of base instances. Denote with $\mathrm{UK}_{\min}$ the number of *unknowns* obtained using the solution in (4) and with UK_M the number of *unknowns* obtained using the M–GEM. Figure 6 represents the mean (dots), the 25-th and 75-th percentiles (boxes), and min/max values (whiskers), over the 30 trials, of the quantity $\mathrm{UK}_{\min} - \mathrm{UK}_M$, which is the difference between the number of *unknowns* obtained with the two approaches. One can notice from Figure 6 that, as M increases from 2 on, all trials exhibit less $unknowns$ than the solution in (4). Indeed, this shows that using the majority voting of M GEMs is more convenient than selecting the best GEM out of M.

B. Real data–sets

We next compare the performance of the majority voting classifier with M GEMs (M=5 and 10) and $M^\star$–GEM against other well-known classification algorithms, like the nearest–neighbor classifier (NNC [3]), the support–vector

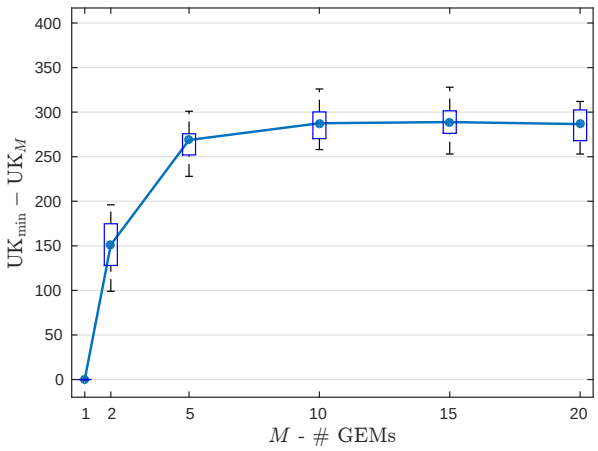

Fig. 6. $\mathrm{UK}_{\min} - \mathrm{UK}_M$ as a function of M ($N = 1000$).

machine (SVM [4]) and the set covering machine (SCM [9], [19]).

For all the data–sets, we use the 10–fold cross validation error as an estimate of the generalization error. Precisely, each time one–tenth of the data is used as a test set for the classifier obtained by using as training data–set the remaining nine–tenths. Thereupon, the operation is repeated for ten times using as a test set another tenth separated from the previous. The values reported in Table I and II are expressed as the total number of errors found in the test set at each trial, averaged on 30 independent execution of (M–, $M^\star$–)GEM majority voting classifiers starting from different tuples of base instances.

As for NNC, SVM and SCM, the results in Table I are taken from [9]. For the SVM, the parameter C refers to the soft margin; for the SCM, type c and d refer to machines based on conjunction or disjunction of boolean-based features (see [9]). In the comparison with the GEM classifier (in all its variants), it is important to remark that these parameters for SVM and SCM are the ones that gave the smallest 10-fold cross validation error among an exhaustive scan of many values for these parameters. Since all these methods do not return the $unknown$ label, to perform a comparison we chose the k value so that the standard GEM gives no $unknowns$. We then set, for each GEM composing the M–GEM and $M^\star$–GEM majority voting classifier, the same theoretical probability of error ϵ determined by the aforementioned k value, with the confidence parameter set equal to δ/M. First of all, by inspecting Table I, it is interesting to notice the positive impact of the majority voting mechanism on the performance of the GEM classifier. In all the data–sets, 5–GEM outperforms the standard GEM, and both do not return any $unknown$ label. The 10–GEM classifier reduces the empirical error probability with respect to the 5–GEM. Moreover, in the Credit data–set, both the 5–GEM and the 10–GEM achieve a better score than SCM, which reaches better results than all the other classifiers on the remaining data–sets.

In Table II we test the influence of the error probability

TABLE I

COMPARISON OF DIFFERENT METHODS IN TERMS OF AVERAGE NUMBER OF ERRORS.

	GEM	5–GEM	10–GEM	NNC	SVM ($C = \infty$)	SVM (finite C)	SCM type c	SCM type d
Glass	47	37.7	36.4	36	42	34	35	36
BreastW	38.1	27.6	26.3	29	27	19	18	16
Haberman	104.7	95.9	95.8	107	111	71	71	93
Pima	261.3	228.4	216.1	247	243	203	189	206
Bupa	136.2	123.8	123.6	124	121	107	109	106
Credit	157.2	118.2	108.5	214	205	190	198	195

parameter on the performance of the 10–GEM and M^*–GEM, both in terms of number of errors and $unknown$ labels, by setting it to $\epsilon/2$ for each GEM used in the 10–GEM and in the M^*–GEM. While the majority mechanism is still able to reduce the number of errors compared to the standard GEM, the a-priori fixed number $M = 10$ of GEMs employed in the 10–GEM results in a non zero number of $unknowns$ (reported in brackets in Table II). The Pima dataset is illustrative of this issue: 10–GEM ends up with the same number of errors of $M^\star$–GEM, but it carries also 40 $unknown$ labels over $N = 768$ instances. On the other hand, M^*–GEM never returns the $unknown$ label, while it is able to reach the same number of errors or even less. In particular, M^*–GEM achieves better results than SCM on Glass and Credit data–sets. Finally, we should point out that SVM and SCM are "tuned" (as stated before) to the data–set and hence their total number of errors are underestimates of the real generalization error. This is not the case for the M– and $M^\star$–GEM majority voting classifiers.

V. CONCLUSIONS

In this work we analyzed the general–purpose GEM classifier that was recently proposed in the literature. Starting from the observation that probabilistic guarantees on the GEM misclassification error can be provided, but at the price of eventually returning the *unknown* label, we developed a majority voting classifier involving multiple GEMs to shrink the *unknown* classification region. We derived probabilistic guarantees on its misclassification error. Numerical tests on both artificial and real benchmark data-sets show that the proposed majority voting classifier is effective, also compared to other well-known state-of-the-art classifiers.

Future research will regard the application of the majority voting classifier to control policy design for a system with a finite control space, where one can compute the optimal action to be applied for a finite number of state samples only, and then identify the optimal map by solving a classification problem [20]. In this setting, a policy can indeed be interpreted as a classifier that associates to each state a label representing the control action, and the existence of a region with an *unknown* label is undesirable.

TABLE II

AVERAGE NUMBER OF ERRORS AND UNKNOWNS FOR $\epsilon/2$.

	10–GEM	$M^\star$–GEM
Glass	35.8 (1.2)	34.6
BreastW	27.9 (1.49)	27
Haberman	88.81 (15)	92.1
Pima	212.8 (40)	212.2
Bupa	118.3 (24.6)	115.4
Credit	104.6 (3.8)	105.4

REFERENCES

[1] V. N. Vapnik, "The nature of statistical learning theory," 1995.

[2] V. N. Vapnik and V. Vapnik, *Statistical learning theory*. Wiley New York, 1998, vol. 1.

[3] L. Devroye, *A probabilistic theory of pattern recognition*. Springer Science & Business Media, 1996, vol. 31.

[4] N. Cristianini and J. Shawe-Taylor, *An introduction to support vector machines and other kernel-based learning methods*. Cambridge university press, 2000.

[5] B. Schölkopf and A. J. Smola, *Learning with kernels: Support vector machines, regularization, optimization, and beyond*. MIT press, 2002.

[6] S. K. Murthy, "Automatic construction of decision trees from data: A multi-disciplinary survey," *Data mining and knowledge discovery*, vol. 2, no. 4, pp. 345–389, 1998.

[7] J. Fürnkranz, "Separate-and-conquer rule learning," *Artificial Intelligence Review*, vol. 13, no. 1, pp. 3–54, 1999.

[8] F. V. Jensen, *An introduction to Bayesian networks*. UCL press London, 1996, vol. 210.

[9] M. Marchand and J. S. Taylor, "The set covering machine," *The Journal of Machine Learning Research*, vol. 3, pp. 723–746, 2003.

[10] S. B. Kotsiantis, I. D. Zaharakis, and P. E. Pintelas, "Machine learning: a review of classification and combining techniques," *Artificial Intelligence Review*, vol. 26, no. 3, pp. 159–190, 2006.

[11] L. Saitta and F. Neri, "Learning in the "real world"," *Machine Learning*, vol. 30, no. 2-3, pp. 133–163, 1998.

[12] I. H. Witten and E. Frank, *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2005.

[13] M. C. Campi, "Classification with guaranteed probability of error," *Mach. Learn.*, vol. 80, no. 1, pp. 63–84, Jul. 2010. [Online]. Available: http://dx.doi.org/10.1007/s10994-010-5183-x

[14] T. Alamo, R. Tempo, and A. Luque, "On the sample complexity of randomized approaches to the analysis and design under uncertainty," in *American Control Conference*, June 2010, pp. 4671–4676.

[15] F. Roli, G. Giacinto, and G. Vernazza, "Methods for designing multiple classifier systems," in *Multiple Classifier Systems*. Springer, 2001, pp. 78–87.

[16] L. Hall, K. Bowyer, W. Kegelmeyer, T. Moore, and C.-m. Chao, "Distributed learning on very large data sets," in *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Citeseer, 2000, pp. 79–84.

[17] A. Asuncion and D. J. Newman, "Uci machine learning repository [http://www. ics. uci. edu/˜ mlearn/mlrepository. html]. irvine, ca: University of california," *School of Information and Computer Science*, 2007.

[18] T. B. Bylander. Tom bylander's home page at the university of texas, san antonio. [Online]. Available: http://www.cs.utsa.edu/ bylander/

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

[20] G. Manganini, L. Piroddi, and M. Prandini, "A classification–based approach to the optimal control of affin switched systems," in 54^{th} *IEEE Conference on Decision and Control*, Osaka, Japan, Dec. 2015.