Theoretical Issues of Mimetic Classifiers

José Hernández-Orallo, Vicent Estruch

Departament de Sistemes Informàtics i Computació
Technical University of Valencia, C. de Vera s/n, 46022 Valencia, Spain.
{jorallo,vestruch}@dsic.upv.es

Abstract. In this report we study the theoretical properties of the simple mimicking method devised for obtaining classifiers that are similar to an oracle. The method is based on generating a set of random examples (random dataset) jointly with (part of) the training dataset. This work complements the experimental results shown in [2]. First, we show that when all the arguments of the function to be learned are nominal, a sufficiently large random dataset allows a loyal mimetic classifier to capture exactly the semantics of the oracle, as expected. Secondly, and more interestingly, we show that if the function to be learned is probabilistically pure (i.e. not fractal) and the classifier is loyal and fence-and-fill then for a sufficiently large random dataset then the error made by the the mimetic classifier will approach zero. We particularise the results for unpruned decision trees, which are loyal and fence-and-fill. Then we study the behaviour of the mimicking method considering learning curve functions. We derive the conditions where the method is useful and we analyse the influence of adding (part of) the training dataset. Finally, we discuss new methods for generating the random dataset considering the first classifier as a soft classifier and ranking the random examples.

Keywords: Mimetic classifiers, learning with oracles, machine learning, unlabelled datasets, co-training, comprehensible models.

1 Introduction

The motivation of this work comes when we look for a new classifier that could be "semantically" similar to an accurate (and maybe complex) classifier (an oracle Ω) but "syntactically" or "structurally" simpler, and ultimately comprehensible. In other words, we look for a new classifier that "mimics" or imitates the behaviour of an oracle (a combined classifier, a neural network, etc.). But how can we do this in a simple and effective manner? In [2] we have presented a method that is based on an additional "random dataset". This dataset can be artificially generated as large as we want, and this is possible just because we want it *unlabelled*, i.e., without class values. Next, the unlabelled random dataset is "classified" or "labelled" by using the oracle. What we obtain now is a labelled random dataset that captures or distils (partially) the semantics of the oracle. And once here, the final stage is easy: we just join this labelled random dataset with (part of) the original training dataset and we can train a single comprehensible model, e.g. a decision tree. We call the final classifier the mimetic classifer, denoted by μ. Figure 1 shows this process:

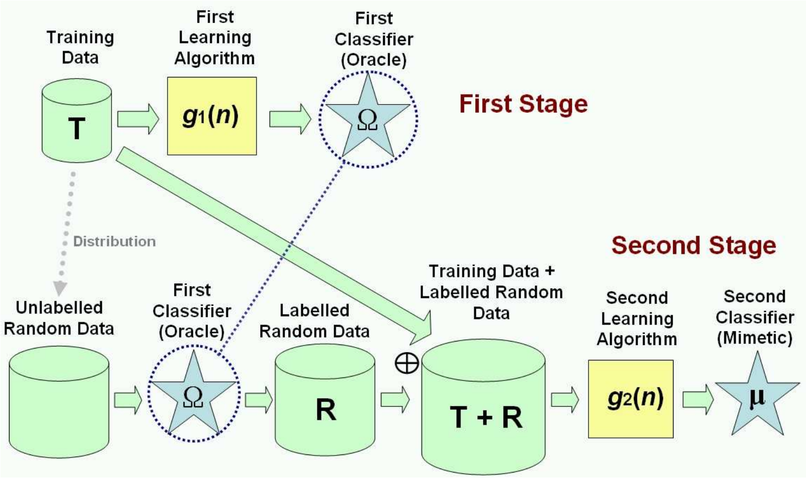

Fig. 1. Mimicking Process

In [2] we have performed an experimental study of different variants of the method, showing that it works in practice and that a prior distribution is better than a uniform distribution for generating the invented dataset. We have also shown that it is usually good to also use the training dataset in the invented dataset, and that results are increasingly better as larger the invented dataset is. However, all these experiments are not accompanied by theoretical results showing the rationale behind these experimental results. This report is devoted to obtain such results.

2 Notation and Settings

Next we give basic definitions for classifiers. An attribute domain X is any set of values, either nominal (finite) or numerical (real). The evidence space E is defined as a product $X_1 \times X_2 \times ... \times X_m$. The class domain C is defined as a finite set of c classes denoted by natural numbers: 0, 1, ..., $c-1$. From here, we define a classifier as a function $f : E \to C$. We usually refer to the function representing the oracle and mimetic classifier by f_Ω and f_μ respectively. Each element $e \in E$ is known as an unlabelled example. A pair (e, i) where $e \in E$ and $i \in C$ is known as a labelled example. A sample dataset is any finite subset of E. A sample dataset can be labelled or unlabelled. We use the notation D_f to designate a dataset D labelled with function f. The subscript can be dropped when f is clear from contexts. Note also that datasets are sets and not multisets. Additionally, and accordingly, we assume there are no equal examples labelled with different classes. We denote by T_τ (or simply T) the training dataset, a labelled dataset used for learning. We denote by R_Ω (or simply R) a random dataset labelled by oracle Ω. A test or validation dataset is a labelled sample dataset usually denoted by V.

If all the attributes in E are numeric, we can also call each $e \in E$ a point. For the sake of simplicity, we may consider (when needed) all numeric attributes $x_j \in X_j$ will be in the interval $[0..1]$. Given two points e_1 and e_2, we denote by $\overline{e_1 e_2}$ the segment connecting both points. The length of the segment is the Euclidean distance between e_1 and e_2, denoted by $d(e_1, e_2)$.

Given an evidence domain $E = X_1 \times X_2 \times ... \times X_m$, we denote by E_φ the subevidence domain constructed only with the nominal attributes, i.e., $E_\varphi = \prod X_j$ such that X_j is nominal. In a similar way, we denote by E_ρ the subevidence domain constructed only with the numeric attributes, i.e., $E_\rho = \prod X_j$ such that X_j is numeric. Obviously, E_φ is finite. We call $|E_\varphi|$ the "nominal dimension" of the dataset. Each element $\alpha \in E_\varphi$ is called an *anchor*. If we fix an anchor then we have that the whole evidence has only variable numeric attributes and we can extend the notion of segment and distance. For instance, we can define the Euclidean distance between two points e_1 and e_2, such that the nominal attributes match (i.e. equal to a certain α), as the Euclidean distance just considering the numeric attributes. If the nominal attributes of e_1 and e_2 are not equal then the distance is infinite. Without loss of generality we can sort the attributes such that every example e can be expressed in the following way $\langle e_f, e_r \rangle$ where $e_f \in E_\varphi$ and $e_r \in E_\rho$.

And, before starting up the definitions, we must clarify that we denote by $Prob(v)$ the probability of event q, and we denote by $Prob_w(v)$ the probability of event v if event w has happened, i.e. the conditional probability $Prob(v|w)$.

Definition 1. *Given an evidence domain E, we say that a method for generating a random sample R is* exhaustive *if and only if:*

when E has only nominal attributes:

$$\forall e \in E : Prob(\exists e' \in R, e = e') > 0$$

when E has only numeric attribues[1]*:*

$$\forall \epsilon > 0, \forall e_1, e_2 \in E, d(e_1, e_2) < \epsilon : Prob(\exists e' \in R, e' \in \overline{e_1 e_2}) > 0$$

when E has nominal and numerical attributes:

$$\forall e_f \in E_\varphi, \forall \epsilon > 0, \forall e_r \in E_\rho, e = \langle e_f, e_r \rangle, \forall e_2 \in E, d(e_1, e_2) < \epsilon : Prob(\exists e' \in R, e' \in \overline{e_1 e_2}) > 0$$

Given the previous definition we can consider several methods of generating random samples (given a training set):

- **Uniform Distribution**: for nominal attributes, one of the possible (seen in the training set) values is randomly chosen according to a uniform distribution. For numeric attributes, one random value is generated by using a uniform distribution on the interval $\{min_i, max_i\}$, where min_i is the lowest value observed for attribute i and max_i is the highest value observed for that attribute in the whole training dataset. This is independently uniform, since one attribute distribution does not affect the distribution of other attributes.

[1] An alternative definition $\forall \epsilon > 0, \forall e \in E : Prob[\exists e' \in R, d(e, e') < \epsilon$ is not valid because a method could be exhaustive in just one dimension or direction.

– **Prior distribution**: each attribute x_i of a new example is obtained as the value v_i in a different example $e \in T$, $e = (v_1, \ldots, v_i, \ldots, v_m)$ selected from the training set by using a uniform distribution. This procedure of generating instances also assumes that all the attributes are independent. Consequently, the method just maintains the probabilities of appearance of the different values observed in each attribute of the training dataset (prior).
– **Prior interpolating distribution**: it is similar to the previous distribution for nominal attributes. But for numeric attributes, we look one value $v_{i,1}$ in the training set as before but then we look for the closest values $v_{i,2}$ and $v_{i,3}$ from other examples such that $v_{i,2} < v_{i,1}$ and $v_{i,3} > v_{i,1}$. Then we use a uniform distribution from $v_{i,2}$ to $v_{i,3}$ to generate a new value v' which is used as the numeric value for the argument i of the example. If the value $v_{i,2}$ does not exist ($v_{i,1}$ has no value on the left), then we use a one-side normal distribution with mean at v_{i_1} and standard deviation half the range (min_i, max_i) of the attribute. For the right value in the alternative situation, we behave similarly.

It is easy to see that the uniform distribution and the prior interpolating distribution are exhaustive (providing we know the range of all the numeric values or we have them normalised from 0 to 1), while the prior distribution is not exhaustive if there are numeric attributes (values that have not appeared have 0 probability). Of course, there are other (exhaustive) methods for generating a dataset, e.g. using kernel density estimation methods.

And just to finish this section, let us define the notion of loyalty, applied to a classifier:

Definition 2. *Let $\mathcal{C}$ be a classifier, we say $\mathcal{C}$ is loyal if for every example e belonging to the training set used for learning the classifier, $\mathcal{C}$ classifies e correctly.*

For instance, an unpruned decision tree classifier is a loyal classifier.

3 General Results

Given the previous definitions we consider the situation where we have a domain E, an oracle Ω and an exhaustive method for generating random examples from E labelled by Ω. The random dataset created is denoted by R. Let us show some properties.

The first property is quite trivial and just shows that if all the attributes are nominal, then there is a finite number of examples and hence the semantics of the oracle can be captured exactly. Formally,

Proposition 1. *Given an evidence domain E with only nominal attributes, an oracle Ω, and a loyal mimetic classifier μ using a random dataset R generated by an exhaustive method labelled with Ω. Then:*

$$if \quad |R| \to \infty \quad then \quad \forall e \in E : f_\mu(e) = f_\Omega(e)$$

Proof. We know that every attribute X_i is nominal. Consequently, $\prod |X_i|$ will be the number of possible examples in E, where $|X_i|$ indicates the number of possible instances of the X_i attribute. If $|R| \to \infty$, since it is generated by an exhaustive method, than we will have necessarily that, at certain point[2], $R = E$ (since E is a finite set), and since μ is a loyal classifier that has used R for learning, we conclude that $f_\mu(e) = f_\Omega(e)$ for all $e \in E$. □

This has been fairly straightforward. It is also fairly obvious that the thing is different when real numbers appear in the domain.

Proposition 2. *Given an evidence domain E with at least one numeric attribute, and a loyal mimetic classifier μ using an random dataset R generated by an exhaustive method labelled with an oracle Ω. Then, there exists an oracle Ω such that*

$$if \quad |R| \to \infty \quad then \quad \exists e \in E : f_\mu(e) \neq f_\Omega(e)$$

Proof. Consider an oracle that represents the function $f_\Omega(X) = \mathit{true}$ if X is rational, otherwise $false$. Even an extremely large random dataset with an exhaustive method of generation may never generate all the rational numbers, so there will be an example such that has not been seen and the classifier may give an incorrect classification. □

This is related to the negative theorems about identification in the limit [3]. In fact we can find other less picturesque counterexamples.

Nonetheless, it seems that the greater the random dataset the mimetic classifier could *approximate* better the oracle, even when there are numeric attributes. However, this cannot be shown for whatever function and whatever classifier. We need to define some slight restrictions on the function and the classifier. Let us start with the function:

Definition 3. *Given a function f we say that a point $e \in E$ is in a pure region of radium ϵ, denoted by $Pure_\epsilon(e)$ if and only if*

$$\forall e' \in E(d(e, e') \leq \epsilon) \to f(e) = f(e')$$

That means that all points (at a distance less than ϵ) are of the same class that e. In other words, there is not a frontier closer than ϵ. It is important to note that ϵ is a paramter and may be different for each point.

Definition 4. *We say a function f only has pure regions (or simply it is pure) if and only if*

$$Prob_{e \in E}[\exists \epsilon > 0 \wedge Pure_\epsilon(e)] = 1$$

No additionally condition is imposed on nominal attributes apart from the fact that the previous condition has to be true for any anchor.

[2] In fact $|R|$ will be never greater than $|E|$.

From here we can allow an infinite number of points in boundaries, because the area of these boundaries is zero. However, we cannot have an infinite number of "spurious" points which are different from their surroundings, if these are spread around the space, even if this infinite number is neglectable in the whole space. For instance, natural numbers among real numbers. The reason is the strict definition of pure region.

In order to avoid this problem, we are going to extend the probability to the first definition.

Definition 5. *Given a function f we say that a point $e \in E$ is in a probabilistically pure region of radium ϵ, denoted by $PPure_\epsilon(e)$ if and only if*

$$Prob_{e' \in E \ | \ d(e,e') \leq \epsilon}[f(e) = f(e')] = 1$$

That means that the probability of selecting another point (at a distance less than ϵ) of the same class that e is equal to 1. In other words, there is not a region boundary closer than ϵ. The term probabilistically pure is a generalisation and includes cases where some finite or infinite sets of "spurious" points (with no area) of other class may be inside the region, but they can be neglected over the overwhelming majority of points which are of the same class as e. From here, now we give a more flexible (probabilistical) definition of pure function:

Definition 6. *We say a function f only has probabilistical pure regions (or simply it is probabilistically pure) if and only if*

$$Prob_{e \in E}[\exists \epsilon > 0 \wedge PPure_\epsilon(e)] = 1$$

No additionally condition is imposed on nominal attributes apart from the fact that the previous condition has to be true for any anchor.

The use of a second probability here is on one hand, to avoid the problems of frontiers as in the firs case, and, on the other hand, to avoid problems precisely with the "spurious" points, for which there does not exist such an ϵ. Again, it is important to note that the ϵ may be different for each point. That means that the function can have as much resolution and *any* form we may want. But, then, what kind of functions are ruled out by the previous definition? The answer is functions for which given a point we cannot even find a small region around it where the class is the same as the point for the overwhelming majority of points. This kind of functions are usually known as fractal functions.

For instance, the following function f is a fractal:

```
f(X) = g(X, 0, 1, true)
g(X, L, R, B) = if (X = (R - L)/2) then return B
                else if (X < (R - L)/2) then g(X, L, (R-L)/2, not(B))
                     else g(X, (R-L)/2, R, not(B))
```

In fact, the previous function f is *so* fractal that its appearance is "grey" (half the points are true and half the points are false) at whatever resolution. Curiously, it may be considered not fractal in a more informal sense! Nonetheless, the previous function is only defined for rational numbers. If we only consider rational numbers, then the previous function is not probabilistcally pure and, logically, is not pure, either.

However, if we make it complete for real numbers, i.e.

```
f'(X) = f(X) if X is rational
f'(X)= false otherwise
```

Now f' has fractal components but, since there are infinite more irrational numbers than rational, probabilistically, we have $f'(X) = false$ and hence, we can say that it is probabilistically pure (although not pure).

In general, most patterns we can think of which are usual in machine learning and data mining applications are probabilistically pure. Moreover, as we will see, with our mimicking method, oracles are usually implemented on a computer with limited resolution, which makes them probabilistically pure or even simply pure. Consequently, the previous restrictions can be considered pretty mild.

Once given these restrictions on the functions we are going to deal with, let us now give a definition of a fence-and-fill learning algorithm.

Definition 7. *A fence-and-fill learning algorithm is an algorithm that only generates pure classifiers.*

Note that it is defined with the no-probabilistic version. This is not an important restriction. If we consider that classifiers are algorithms that give an output for an input in a finite time then they are pure.

Now, given the previous definitions, we can assert a positive result:

Lemma 1. *Given an evidence domain E with only numeric attributes, an oracle Ω such that its characteristic function f_Ω is probabilistically pure, and a loyal fence-and-fill mimetic classifier μ using a random dataset R generated by an exhaustive method labelled with Ω. Then:*

$$if \quad |R| \to \infty \quad then \quad Prob_{e \in E}[f_\mu(e) = f_\Omega(e)] = 1$$

Proof. We know that every attribute X_i is numeric, so we can apply the definitions that use distances.

Since R is generated by an exhaustive method, we have,

$$\forall \epsilon > 0, \forall e_1, e_2 \in E, d(e_1, e_2) < \epsilon : Prob(\exists e' \in R, e' \in \overline{e_1 e_2}) > 0$$

But we will have necessarily, since $|R| \to \infty$, that:

$$\forall \epsilon > 0, \forall e_1, e_2 \in E, d(e_1, e_2) < \epsilon : \exists e' \in R, e' \in \overline{e_1 e_2} \qquad (1)$$

This means that there are examples all around.

Since the mimetic classifier is loyal:

$$\forall e \in R : f_\mu(e) = f_\Omega(e)$$

Since the mimetic classifier is fence-and-fill:

$$Prob_{e \in E} \exists \epsilon > 0 \wedge Pure_\epsilon(e)$$

extending the definition of $Pure_\epsilon(e)$ we have:

$$Prob_{e \in E} \exists \epsilon > 0 \wedge (\forall e' \in E(d(e,e') \leq \epsilon) \rightarrow f_\mu(e) = f_\mu(e'))$$

Consequently, from being loyal and fence-and-fill, we have:

$$Prob_{e \in R}[\exists \epsilon > 0 \wedge (\forall e' \in E(d(e,e') \leq \epsilon) \rightarrow f_\mu(e) = f_\mu(e'))] = 1 \qquad (2)$$

That means that the mimetic classifier matches with the oracle in the overwhelming majority of the seen points, and their surrounding points have the same class assigned as the centre point.

And now, we know that f_Ω is probabilistically pure, that means:

$$Prob_{e \in E}[\exists \epsilon > 0 \wedge PPure_\epsilon(e)] = 1$$

extending the definition of $PPure_\epsilon(e)$ we have:

$$Prob_{e \in E}[\exists \epsilon > 0 \wedge Prob_{e' \in E \ | \ d(e,e') \leq \epsilon}[f_\Omega(e) = f_\Omega(e')] = 1] = 1 \qquad (3)$$

which means that for the overwhelming majority of the points, the class given by the oracle matches the surrounding points.

From (2) and (3) we have:

$$Prob_{e \in R}[\exists \epsilon > 0 \wedge Prob_{e' \in E \ | \ d(e,e') \leq \epsilon}[f_\mu(e') = f_\Omega(e')] = 1] = 1$$

That means that the mimetic classifier matches with the oracle in the overwhelming majority of the seen points ($d(e,e') = 0$) and their surrounding points ($0 < d(e,e') \leq \epsilon$).

And finally, from this and (1) we have:

$$Prob_{e \in E}[\exists \epsilon > 0 \wedge Prob_{e' \in E \ | \ d(e,e') \leq \epsilon}[f_\mu(e') = f_\Omega(e')] = 1] = 1$$

which means that the mimetic classifier matches with the oracle in the overwhelming majority points and their surrounding points, which logically entails that it matches on the overwhelming majority of points, i.e.:

$$Prob_{e \in E}[f_\mu(e) = f_\Omega(e)] = 1$$

□

The previous lemma means that with the conditions imposed the error will approach zero when we generate sufficiently many random examples for the mimetic classifier.

And from the previous lemma for evidences composed exclusively by numeric attributes, we can now establish the result for both numerical and nominal attributes.

Theorem 1. *Given an evidence domain E with either numeric or nominal attributes, an oracle Ω such that its characteristic function f_Ω is probabilistically pure, and a loyal fence-and-fill mimetic classifier μ using a random dataset R generated by an exhaustive method labelled with Ω. Then:*

$$if \quad |R| \rightarrow \infty \quad then \quad Prob_{e \in E}[f_\mu(e) = f_\Omega(e)] = 1$$

Proof. (sketch). If the domain E contains only nominal or numeric it is proven by proposition 1 or by lemma 1 respectively. If both nominal and numeric attributes exist in E, we can divide it into nominal E_φ and numeric E_ρ. Since we discussed in the beginning E_φ is finite and each element $\alpha \in E_\varphi$ is called an *anchor*. Since the way to generate examples is exhaustive, we have non-null probability for each anchor, and for each anchor we have the same conditions for making lemma 1 hold. □

From here, we have that for almost every function of interest (probabilistic pure) represented by an oracle we can approximate it as well as we want with fence-and-fill methods and a large enough dataset.

4 Results for Decision Trees

Since we are using decision trees for mimicking because one of the objectives of this technique is to obtain a comprehensible representation of any oracle, it would be interesting to particularise the previous results for decision trees. Here we go:

Proposition 3. *An unpruned decision tree learner is a loyal fence-and-fill learner.*

Proof. It is loyal because it is unpruned, and all the leaves are pure and hence all the training examples are classified correctly.

It is fence-and-fill because a decision tree is pure, since, for numeric attributes, partitions are of the form

$$X_i < v \quad | \quad X_i \geq v$$

where v is a rational number. That means that, in the end, the space formed by the numeric attributes of E will have a partition based on hypercube regions for each anchor. This kind of partitions (hypercubes) are pure for all the points except from the points at the boundaries. Since there is a finite number of conditions in a decision tree, the area of the boundaries is neglectable and, hence we have that:

$$Prob_{e \in E}[\exists \epsilon > 0 \wedge Pure_\epsilon(e)] = 1$$

which is the definition of a pure classifier.

Once seen they are loyal and pure then we have that decision tree learners are fence-and-fill learners. □

From here, we have the following trivial corollary of the previous theorem:

Corollary 1. of Theorem 1 *Given an evidence domain E with either numeric or nominal attributes, an oracle Ω such that its characteristic function f_Ω is probabilistically pure, and a unpruned decision tree learner μ that mimics Ω by using a random dataset R generated by an exhaustive method labelled with Ω. Then:*

$$if \quad |R| \to \infty \quad then \quad Prob_{e \in E}[f_\mu(e) = f_\Omega(e)] = 1$$

Proof. From the previous proposition. □

Of course it would be interesting to relate the error rate with the number of examples. However, this depends on the resolution of f_Ω, the dimension of the space and the way in which the random datasets are generated. We think no interesting bounds can be obtained from here. However, we can still do something interesting if we consider the whole picture and we are given the learning curves of the classifiers. This is addressed next.

5 Learning Curves and Influence of the Training Set

In this section, we are complicating things a little bit. Now we consider a real or target function f_τ (from which we have a training dataset and a test dataset). The target function is approximated by an oracle, and then the mimetic classifier wants to approximate the oracle. For the moment we do not use the training set for the second stage and we have a scenario as shown in Figure 2:

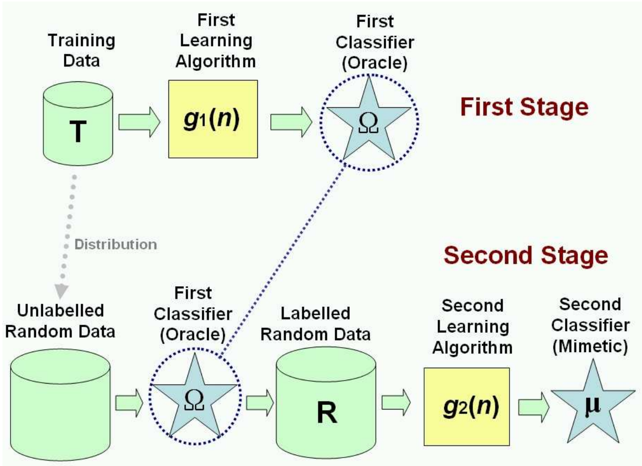

Fig. 2. Mimicking Process without Training Set used in the Second Stage

Seen in a more general way, we can say that we have a first learning stage with fidelity a_1 and a second learning stage with fidelity a_2 wrt. the results of the first stage. From here, we can establish the whole fidelity $a_{1:2}$ easily:

Theorem 2. *Given a two-stage learning scenario with c classes, where the first learner has fidelity a_1 for that problem and the second learner has fidelity a_2 for the problem representing the first stage, we have that:*

$$a_{1:2} = a_1 \cdot a_2 + (\frac{1}{c-1})(1-a_1) \cdot (1-a_2)$$

Proof. The proof is straightforward. For the first part of the sum we have to consider the cases where the target function and the function in the first stage match (i.e. a_1) and also the cases where the function in the first stage and the function in the second stage match (a_2). The probability that both things happen just gives the first part of the sum.

For the second part of the sum we have to consider the cases where the target function and the function in the first stage do not match, i.e. $1 - a_1$. In this situation we have to consider that for all the cases where the first stage function and the second stage function do not match either, i.e. $1 - a_2$, some of them will, by chance, correct the former error. It is easy to see that this happens in just $1/(c-1)$ of the cases. Consequently, the probability that the first stage fails but the second stage corrects it by chance gives the second part of the sum. □

We can particularise the previous trivial result for the "oracle accuracy" (the similarity between the target function and the oracle), "fidelity" (the similarity between the oracle and the mimetic classifier) and "mimetic accuracy" (the similarity between the target function and the mimetic classifier)

We can define this more formally:

Definition 8. *Oracle accuracy is defined as:*

$$Acc_{\tau,\Omega} = Prob_{e \in E} f_\tau(e) = f_\Omega(e)$$

Definition 9. *Fidelity is defined as:*

$$Acc_{\Omega,\mu} = Prob_{e \in E} f_\Omega(e) = f_\mu(e)$$

Definition 10. *Mimetic accuracy is defined as:*

$$Acc_{\tau,\mu} = Prob_{e \in E} f_\tau(e) = f_\mu(e)$$

From here, we can obtain the following trivial result:

Theorem 3. *Given a target function of c classes, an oracle Ω and a mimetic classifier μ, we have that:*

$$Acc_{\tau,\mu} = Acc_{\tau,\Omega} \cdot Acc_{\Omega,\mu} + (\frac{1}{c-1})(1 - Acc_{\tau,\Omega}) \cdot (1 - Acc_{\Omega,\mu})$$

Proof. Trivial from Theorem 2. □

5.1 Learning Curves

Up to now, we have considered global error and we have given for granted that we know the error committed by each stage for a given size of training dataset T (for the first stage) and random dataset R (for the second stage). With the conditions assumed in previous sections, if we indefinitely increase the size of both, we would have that $a_1 \to 1$ and $a_2 \to 1$ and hence, $a_{1:2} \to 1$. However increasing indefinitely T is not possible since we are usually given a finite training set and increasing indefinitely R is not possible due to computationally restrictions (space and time).

Consequently, we need to generalise the constant "fidelities" to variable "fidelities". More precisely, from now, we consider a "learning curve" for each classifier, which tells us which specific value of fidelity is obtained for a given dataset size.

Definition 11. *Given a classifier, a fidelity learning curve function is defined as a function $g : \mathbb{N} \to \mathbb{R}$ and $\forall x : g(x) \in [0,1]$ such that given a dataset D extracted from target function or problem f, where D is used by the classifier, then the fidelity of the classifier wrt. f is given by $g(n)$ where $n = |D|$.*

Obviously, that function g may not exist even for a single classifier and problem, may be discontinuous and, obviously, may depend not only on the size of the dataset but also on the order of appearance of the examples. Consequently, we consider them to be "average-case" functions, i.e., either they are averages of all (or a representative sample of) the possible combinations of examples of each size or they also average the behaviour for several problems[3]. Nonetheless, the use of learning curve estimations or functions is not new, and has been used for more than a decade, theoretical bounds established (depending on the Vapnik-Chervonenkis dimension or other dimensions of the problem) and specific approximations found (see e.g. [5][4]).

Consequently, the function could be estimated and averaged over many subsets of E and even for many problems. Moreover, and what it is interesting, it can be used to model some behaviours and, as we will see next, to understand the reason why sometimes a particular technique does not work well or which options are better.

The performance of classifiers regarding the number of examples seen usually has a form of an inverse exponential growth curve, which usually begins from the accuracy given by a random classifier $1/c$ and rises first quickly and then more slowly to a maximum. This kind of behaviour can be modelled by the following learning curve:

Definition 12. *A "saturation" fidelity learning curve function is defined as the function of the form:*

$$s_{[a,\alpha]}(n) = (a - \frac{1}{c})(1 - e^{\alpha n}) + \frac{1}{c}$$

where a is the maximum fidelity or saturation point and α is the growth rate.

An example of a saturation learning curve ($s_{[0.9,0.05]}(n)$ for a 5-class problem is shown in Figure 3).

It is not difficult to consider a particular learning algorithm and learn one or more models for increasing sizes of the training set. In many cases, the previous function can fit well the behaviour of the algorithm and dataset, making it

[3] We cannot consider all the possible problems because of the free-lunch theorem [6] at least without further assumptions, but we can consider a representative sample of problems (e.g. the UCI dataset)

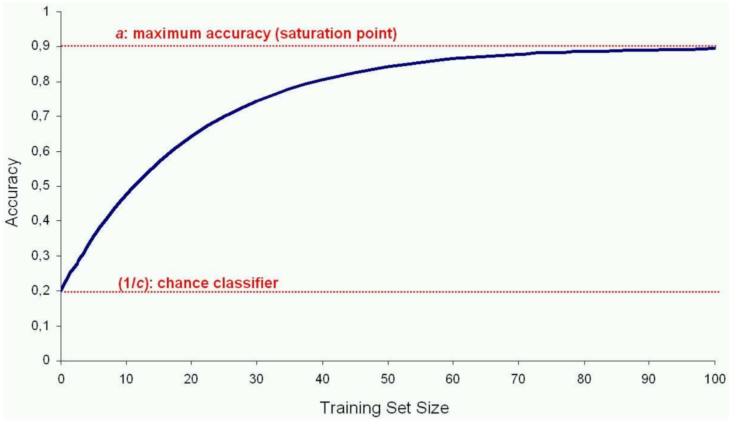

Fig. 3. Saturation Learning Curve

possible to obtain the parameters a and α in order to extrapolate for larger datasets.

In what follows we consider generalised curve functions for theoretical results and we just use saturation curves for examples.

And now, once presented the learning curve functions, we can give a characterisation of when it is useful to use the mimicking method:

Theorem 4. *Given a two-stage learning scenario with c classes, where the first learner has fidelity learning curve function g_1 for that problem and the second learner has fidelity g_2 for either the original problem and the problem representing the first stage, and a dataset T (for the first stage, which is considered correct wrt. the target function) of length $t = |T|$ and a dataset R (for the second stage) of length $r = |R|$ and accuracy $g_1(t)$, then the mimetic procedure is useful if and only if:*

$$g_2(t) < g_1(t) \cdot g_2(r) + (\frac{1}{c-1})(1 - g_1(t)) \cdot (1 - g_2(r))$$

Proof. If we do not use the mimicking procedure, then we can use the training set T with the second classifier and we have an accuracy of $g_2(t)$. If we use the mimetic procedure, we have the expression from Theorem 3. Consequently, the mimetic procedure is useful when the former expression is not greater or equal than the second expression. $\square$

Let us see with some examples whether we can determine the use or not of the mimimicking technique.

Example 1. Consider a neural network with fidelity learning curve function $g_1(n) = s_{[0.9,0.05]}(n)$ and a decision tree with fidelity learning curve function $g_2(n) = s_{[0.8,0.1]}(n)$. We have a training dataset with 50 examples. Is it reasonable to use the mimicking method?

The answer is no. It turns out that the use of mimicking (with infinite random dataset) will attain an accuracy of 0.69 while the direct use of the second classifier obtains almost 0.80 accuracy (the curve is rather steep). In fact, for these two curves there are no training size for which the mimicking method is useful. This is an example where the first classifier is good but slow and the second is bad but quick. Mimicking is useless.

Example 2. Consider a logistic regression classifier with fidelity learning curve function $g_1(n) = s_{[0.9,0.05]}(n)$ and a decision tree with fidelity learning curve function $g_2(n) = s_{[0.8,0.01]}(n)$. We have a training dataset with 100 examples. Is it reasonable to use the mimicking method? And with 200 examples? And with 300 examples?

For 100 examples the answer is yes. It turns out that the use of mimicking (with infinite random dataset) will attain an accuracy of 0.721 while the direct use of the second classifier obtains 0.579 accuracy.

With 200 training examples we have that mimicking (with an infinite random dataset) will attain an accuracy of 0.725 while the direct use of the second classifier obtains 0.719. The answer is yes.

With 300 training examples we have that mimicking (with infinite random dataset) will attain an accuracy of 0.725 while the direct use of the second classifier obtains 0.770. The answer is yes.

This is an example where the first classifier is good and quick and the second is bad and slow. In this situation we see that in some cases the mimicking method is useful while in others is not.

Example 3. Consider a Bayesian classifier with fidelity learning curve function $g_1(n) = s_{[0.8,0.1]}(n)$ and a decision tree with fidelity learning curve function $g_2(n) = s_{[0.9,0.01]}(n)$. We have a training dataset with 100 examples. Is it reasonable to use the mimicking method? And with 300 examples?

The answer is yes. It turns out that the use of mimicking (with infinite random dataset) will attain an accuracy of 0.725 while the direct use of the second classifier obtains 0.642 accuracy. With 300 training examples we have that mimicking (with infinite random dataset) will attain an accuracy of 0.725 while the direct use of the second classifier obtains 0.865. This is an example where the first classifier is bad and quick and the second is quick and slow. In some cases the mimicking method is useful.

Of course, there are cases where the mimicking method is useful because we have a black-box and we do not have training set.

In figures 4 and 5 we illustrate two typical cases in a graph that we call a "Mimetic Response Graph". This graph shows three curves: the learning curve of the first classifier, the learning curve of the second classifier if it were learned with different sizes of training data and, additionally, we show the learning curve of the mimetic classifier, considering the random data infinite. These graphs are quite illustrative, because we can see whether the method is useless (if the mimetic curve is below the curve of the second classifier) or in case it is useful, for which training sizes it is useful.

The first figure 4 shows a case where the mimicking method is useful for short training sets (no. examples ¡ 415).

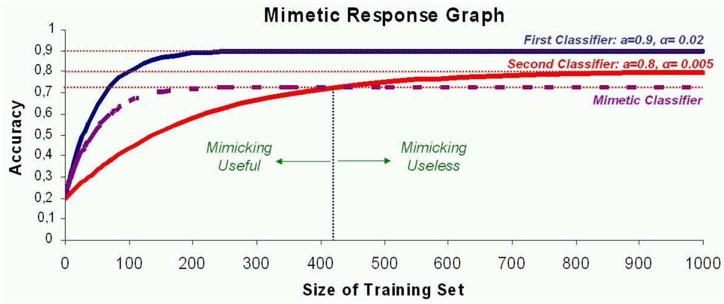

Fig. 4. Mimetic Response Graph when Mimicking is useful

The second figure 5 shows a case where the mimicking method is useless independently of the size of the training set.

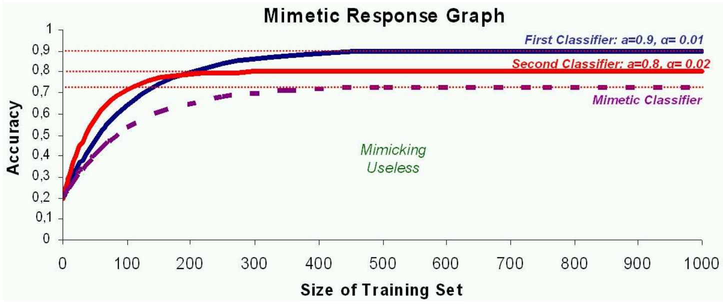

Fig. 5. Mimetic Response Graph when Mimicking is useless

According to the previous results and rationales, it seems that the most important thing to know of both classifiers is their learning rate. This is even more important than their maximum accuracies. There are cases where quick but bad classifiers can be used for the first stage and slow but good classifiers can be used for the second, especially when there are not many examples available.

Nonetheless, it must be stated that the mimicking procedure may be better than expected because g_2 may be better for the oracle than for the target function. This may happen because the oracle is usually an algorithmic device and hence is pure. By the results of the previous section, this means that fence-and-fill methods for the second stage will capture correctly the semantics of the oracle for a large random dataset. Let us formalise this rationale.

Lemma 2. *Given a mimicking scenario. If the first stage classifier (the oracle) is probabilistically pure and the second stage classifier is pure and loyal then the average-case learning curve of the second classifier follows the condition:*

$$if \quad |R| \to \infty \quad then \quad g_2'(|R|)) = 1$$

We use g_2' to denote that this good behaviour only happens wrt. the oracle. We reserve g_2 to the learning curve wrt. the target function.

Proof. Direct from theorem 1. □

And now, from this, we can assert the following theorem:

Theorem 5. *Given a mimicking scenario with training set T of size $t = |T|$. If the first stage classifier (the oracle) with learning curve g_1 is probabilistically pure and the second stage classifier with learning curve g_2 is pure and loyal then the mimetic classifier will be useful if and only if:*

$$g_2(t) < g_1(t)$$

Proof. From theorem 4 we have:

$$g_2(t) < g_1(t) \cdot g_2'(r) + (\frac{1}{c-1})(1 - g_1(t)) \cdot (1 - g_2'(r))$$

Considering we can generate as many random examples as we want then we have $|R| \to \infty$. Hence, from lemma 2 we have:

$$g_2(t) < g_1(t) \cdot 1 + (\frac{1}{c-1})(1 - g_1(t)) \cdot (1 - 1)$$

$$g_2(t) < g_1(t) + (\frac{1}{c-1})(1 - g_1(t)) \cdot 0$$

$$g_2(t) < g_1(t)$$

□

This is a very good result and can be seen in the mimetic response graph as that whenever there is a zone where the curve of the first classifier is over the curve of the second classifier, then the mimicking method is useful. Note that in the mimetic response graph we must show the behaviour of the second classifier wrt. the target function, not wrt. the oracle. This is precisely the point we have discussed, because the second tends to 1 whereas the first may not.

5.2 Influence of the Training Set

And given all this, now we can consider the situation where, additionally to the random dataset R, the mimetic classifier also uses the training set T. Consequently, we have a dataset used for the second stage $I = R \cup T$. This dataset will be referred as the "invented" dataset or the joint dataset. This is shown in figure 6.

For this new situation, we have to redefine our previous results.

First of all, we have to generalise our notion of learning curve to take two datasets of different distributions into account. Even taking into account averages, it is clear that the function $g(n)$ is just an idealisation that in many cases can only be approximated. But this is still possible because examples are drawn from a single distribution. In the case of joining R and T both datasets try to model the same problem but must be considered two different distributions. Consequently, we must extend the definition of the learning curve for the second stage. First we need to define an averager combined classifier:

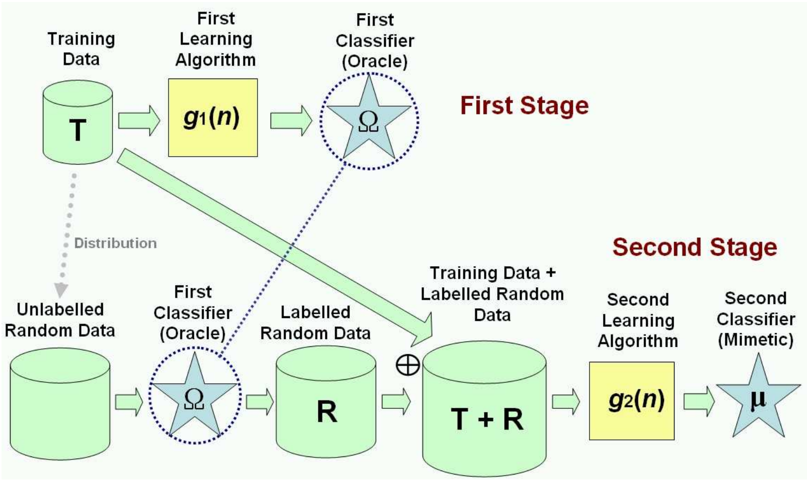

Fig. 6. Mimicking Process with Training Set used in the Second Stage

Definition 13. *Given two classifiers f_1 and f_2 and a ratio $r \in [0,1]$. An averager combined classifier is defined as $\mathcal{C}[r](f_1, f_2)$, where the predictions of $\mathcal{C}$ are averaged in such a way that given an example e:*

$$if \;\; f_1(e) = f_2(e) = i \;\; then \;\; \mathcal{C}(e) = i$$

$$otherwise \; : Prob(\mathcal{C}(e) = f_1(e)) = r \;\; and \;\; Prob(\mathcal{C}(e) = f_2(e)) = 1 - r$$

Definition 14. *Given a classifier input with data from two distributions, a proportional fidelity learning curve function is defined as a function $g : \mathbb{N} \times \mathbb{N} \to \mathbb{R}$ such that $\forall x, y : g_1(x, y) \in [0,1]$ and given a dataset D_1 extracted from target function or problem f_1 and a dataset D_2 extracted from target function or problem f_2, where $D_1 \cup D_2$ is used by the classifier, then the fidelity of the classifier wrt. the combined classifier $\mathcal{C}[n_1/(n_1+n_2)](f_1, f_2)$ is given by $g(n_1, n_2)$ where $n_1 = |D_1|$ and $n_2 = |D_2|$.*

Consider for instance a problem defined by function $f_1 = true$ and another problem defined by function $f_2 = false$. If we generate D_1 examples from f_1 and D_2 examples from f_2, a learner with good learning functions for f_1 and f_2 *separately* may have a very bad schizophrenic curve for an averager combined classifier $\mathcal{C}[r]$ when joining D_1 and D_2, especially if $r = n_1/(n_1 + n_2) \approx 0.5$.

But with these generalisation of the learning curve, we can now address the issue of assessing the influence of the training set added to the random dataset:

Theorem 6. *Let us consider a two-stage learning scenario with c classes, a dataset T (for the first stage, which is considered correct wrt. the target function) of length $t = |T|$ and a dataset $I = R \cup T$ (for the second stage) of length $i = |I|$, where R is a random dataset generated from the first stage with an exhaustive method of length $r = |R|$. Consider as well that the first learner is pure and has fidelity learning curve function g_1 for the target function and the second learner has fidelity $g_2(r, t)$ for the problem representing the invented dataset.*

For a sufficiently large number of random examples and training examples, and considering R and T have no common examples, we have:

$$g_{1:2}(r,t) = \frac{r}{r+t} \cdot [g_1(t) \cdot g_2(r,t) + (\frac{1}{c-1})(1-g_1(t)) \cdot (1-g_2(r,t))]$$
$$+(1-\frac{r}{r+t}) \cdot g_2(r,t)$$

Proof. If we consider the "averager" combined classifier $C_1(r,t)$ just before the second stage, we have:

$$C_1(r,t) = g_1(t)\frac{r}{r+t} + \frac{t}{r+t}$$

The first part has the $g_1(t)$ because it comes from the first stage classifier through R. The second part comes from the target function directly through T, which is considered correct.

And now, we have a first-stage function $C_1(r,t)$ and a second-stage learning curve function $g_2(r,t)$. From theorem 3 we have:

$$C_1(r,t) \cdot g_2(r,t) + (\frac{1}{c-1})(1-g_1(r,t)) \cdot (1-g_2(r,t))$$

If we use the expression of $C_1(r,t)$ in the above formula, we have:

$$[g_1(t)\frac{r}{r+t} + \frac{t}{r+t}] \cdot g_2(r,t) + (\frac{1}{c-1})(1-[g_1(t)\frac{r}{r+t} + \frac{t}{r+t}]) \cdot (1-g_2(r,t)) =$$

$$\frac{r}{r+t}g_1(t) \cdot g_2(r,t) + \frac{t}{r+t} \cdot g_2(r,t) + (\frac{1}{c-1})(\frac{r}{r+t} - g_1(t)\frac{r}{r+t}) \cdot (1-g_2(r,t)) =$$

$$\frac{r}{r+t}g_1(t) \cdot g_2(r,t) + \frac{t}{r+t} \cdot g_2(r,t) + \frac{r}{r+t}\frac{1}{c-1}(1-g_1(t)) \cdot (1-g_2(r,t)) =$$

$$\frac{r}{r+t}[g_1(t) \cdot g_2(r,t) + \frac{1}{c-1}(1-g_1(t)) \cdot (1-g_2(r,t))]+$$
$$(1-\frac{r}{r+t}) \cdot g_2(r,t)$$

□

The previous theorem establishes a general model from this variant of the mimicking, where the training set is used in both stages. The problem of the previous model is that $g_2(r,t)$ may be difficult to be estimated for all r and t. The approach we are taking below is based on the definition of three specific models, according to several assumptions on learning curve $g_2(r,t)$. These models are:

- **Model 1 (Ideal)**: We assume that $\forall g_2(r,t) = g_2(r+t)$ where g_2 is the learning curve function of the second classifier. This assumption makes sense if R and T are highly similar.
- **Model 2 (Optimistic)**: We assume that $\forall g_2(r,t) = g_2(max(r,t))$ where g_2 is the learning curve function of the second classifier. Since r will usually be greater than r, we will simply consider $g_2(r,t) = g_2(r)$. This assumption makes sense if R and T are similar, but T adds complexity to R and slows the learning rate.
- **Model 3 (Pessimistic)**: We assume that $\forall g_2(r,t) = g_2(min(r,t))$ where g_2 is the learning curve function of the second classifier. Since r will usually be greater than r, we will simply consider $g_2(r,t) = g_2(t)$.

Of course we can consider even more pessimistic models than model 1, but we assume that at least R and T try to model the same function (i.e. the first-stage classifier is not very bad).

Let us start with model 1.

Proposition 4. *Under model 1, the mimicking procedure is useful if and only if:*

$$\frac{r}{r+t} \cdot [g_1(t) \cdot g_2(r+t) + (\frac{1}{c-1})(1-g_1(t)) \cdot (1-g_2(t+r))]$$
$$+(1-\frac{r}{r+t}) \cdot g_2(r+t)$$
$$>$$
$$g_2(t)$$

Proof. Just using the assumption of model 1 in theorem 6 we have the first expression. The second expression is just given with the second classifier learning from the training set only. □

Proposition 5. *Under model 1, for a given n, if $g_1(n) < 1$ and $g_2(n) > 1/c$ then it is beneficial to use the training set as well.*

Proof. From model 1, we can express the final accuracy of the mimicking structure using the training set as:

$$\frac{r}{r+t} \cdot [g_1(t) \cdot g_2(r+t) + (\frac{1}{c-1})(1-g_1(t)) \cdot (1-g_2(r+t))]$$
$$+(1-\frac{r}{r+t}) \cdot g_2(r+t) \qquad (1)$$

Let us concentrate on the second part of the first expression and let us proof first that:

$$(\frac{1}{c-1})(1-g_1(t)) \cdot (1-g_2(r+t)) < ((1-g_1(t)) \cdot g_2(t+r) \qquad (2)$$

This can be shown simplifying it to:

$$\begin{aligned}(\frac{1}{c-1}) \cdot (1 - g_2(t+r)) &< g_2(r+t) \\ (1 - g_2(t+r)) &< (c-1) \cdot g_2(t+r) \\ 1 &< c \cdot g_2(t+r) \\ 1/c &< g_2(t+r)\end{aligned}$$

which is true by assumption.

And now, considering (1) we have that:

$$\begin{aligned}&[g_1(t) \cdot g_2(r+t) + (\frac{1}{c-1})(1 - g_1(t)) \cdot (1 - g_2(r+t))] \\ &< [g_1(t) \cdot g_2(r+t) + ((1 - g_1(t)) \cdot g_2(r+t)] \\ &= [g_2(r+t)]\end{aligned}$$

Consequently, in (1), we have that the first part is lower than the second. Since both parts are proportional, it makes sense to make the second term higher by increasing the proportion of t wrt. r □

According to the previous propositions, we can see that in general the cases where the mimicking technique is useful increases.

The previous corollaries confirm intuition and, in this sense, are not much too informative. However, they can be used to clarify what to do in some examples:

Example 4. Consider a neural network with fidelity learning curve function $g_1(n) = s_{[0.9,0.05]}(n)$ and a decision tree with fidelity learning curve function $g_2(n) = s_{[0.8,0.1]}(n)$. We have a training dataset with 50 examples. Is it reasonable to use the mimicking method?

The answer is no. It turns out that the use of mimicking has a maximum with a random dataset size of 30 examples (with infinite random dataset it is lower) giving at most 0.75 while the direct use of the second classifier obtains 0.80 accuracy. In fact, for these two curves there are no training size for which the mimicking method is useful. This is an example where the first classifier is good but slow and the second is bad but quick. Mimicking is useless. The results are equal to those without the training set.

Example 5. Consider a logistic regression classifier with fidelity learning curve function $g_1(n) = s_{[0.9,0.05]}(n)$ and a decision tree with fidelity learning curve function $g_2(n) = s_{[0.8,0.01]}(n)$. We have a training dataset with 100 examples. Is it reasonable to use the mimicking method? And with 200 examples? And with 300 examples?

For 100 examples the answer is yes. It turns out that the use of mimicking has a maximum with a random dataset size of 500 examples (with an infinite random dataset it is lower) giving at most 0.735 (without the training it was 0.721) while the direct use of the second classifier obtains 0.579 accuracy.

With 200 training examples we have that mimicking has a maximum with a random dataset size of 500 examples (with an infinite random dataset it is lower) giving at most 0.745 (without the training it was 0.725) while the direct use of the second classifier obtains 0.719. The answer is yes. With 220 training examples we would have that it would be useful with the training set but useless without the training set.

With 300 training examples we have that mimicking has a maximum with a random dataset size of 500 examples (with an infinite random dataset it is lower) giving at most 0.752 (without the training it was 0.725) while the direct use of the second classifier obtains 0.770. The answer is no.

This is an example where the first classifier is good and quick and the second is bad and slow. In some cases the mimicking method is useful. In comparison with examples in previous sections, the use of the training set make more cases useful.

Example 6. Consider a Bayesian classifier with fidelity learning curve function $g_1(n) = s_{[0.8,0.1]}(n)$ and a decision tree with fidelity learning curve function $g_2(n) = s_{[0.9,0.01]}(n)$. We have a training dataset with 100 examples. Is it reasonable to use the mimicking method? And with 150 examples? And with 200 examples? And with 300 examples?

The answer is yes. It turns out that the use of mimicking has a maximum with a random dataset size of 500 examples (with an infinite random dataset it is lower) giving at most 0.751 (without the training it was 0.725) while the direct use of the second classifier obtains 0.642 accuracy.

With 150 training examples we have that mimicking has a maximum with a random dataset size of 500 examples (with an infinite random dataset it is lower) giving at most 0.763 (without the training it was 0.725) while the direct use of the second classifier obtains 0.744. The answer is yes. (Without training the answer was no).

With 200 training examples we have that mimicking has a maximum with a random dataset size of 500 examples (with an infinite random dataset it is lower) giving at most 0.773 (without the training it was 0.725) while the direct use of the second classifier obtains 0.805. The answer is no.

With 300 training examples we have that mimicking has a maximum with a random dataset size of 200 examples (with an infinite random dataset it is lower) giving at most 0.799 (without the training it was 0.725) while the direct use of the second classifier obtains 0.865. The answer is no.

This is an example where the first classifier is bad and quick and the second is quick and slow. In some cases the mimicking method is useful. In comparison with examples in previous sections, the use of the training set make more cases useful.

Summing up, there can be cases where increasing the number of random examples is always good. There are other cases where the mimetic technique is of no use. But, as shown in some previous examples and unlike what happened with the mimicking method without the training set, when we use the training set jointly with the random set there are some cases where the mimicking technique is good upto a certain amount of random data but further increasing the random data is not beneficial.

This is interesting to note that for some combinations of learning curves there may be three different zones. The following generalised response graph (Figure 7) shows in a third dimension the influence of the size of the random dataset (the first classifier has learning curve $s_{[0.8,0.1]}$ and the second $s_{[0.7,0.02]}$:

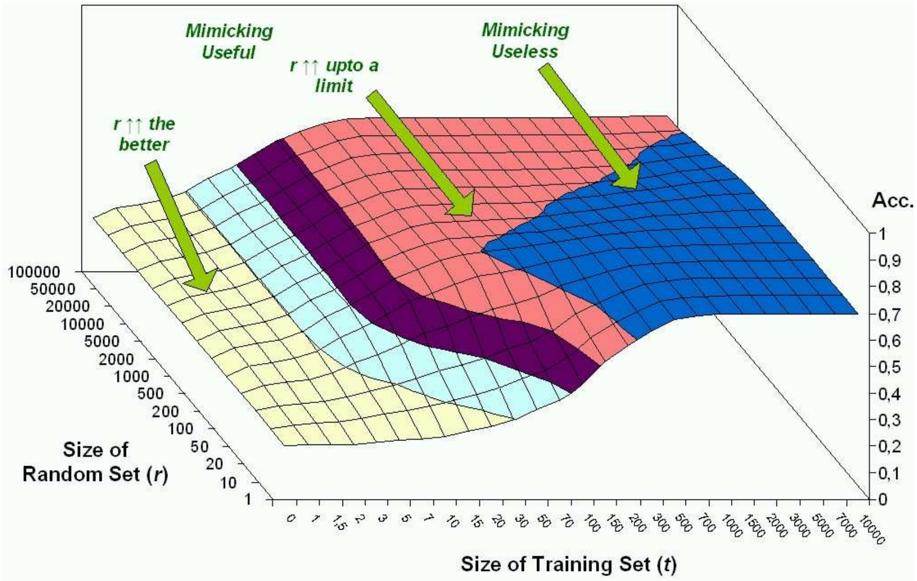

Fig. 7. Generalised Mimetic Response Graph when Training is used

It seems counterintuitive that there is a zone for which increasing the random dataset is not beneficial. This is a pity, because this is one of the easiest parameters that can be modified to increase the quality of the whole process.

Fortunately, it seems there may be a way-out in these strange zones. The idea is based on noting that the previous graph is always increasing for whatever (r, t) to $(r * k, t * k)$ where $k > 1$. Let us prove it:

Proposition 6. *Under model 1 and non-decreasing learning curve functions, if we increase r and t in the same proportion (i.e. $(r * k, t * k)$ where $k > 1$) then we have greater or equal overall accuracy.*

Proof. From model 1 we have:

$$\frac{r}{r+t} \cdot [g_1(t) \cdot g_2(r+t) + (\frac{1}{c-1})(1 - g_1(t)) \cdot (1 - g_2(r+t))]$$
$$+(1 - \frac{r}{r+t}) \cdot g_2(r+t) \qquad (1)$$

It is trivial to see that the ratios $\frac{r}{r+t}$ remain constant after the increase $(r*k, t*k)$, so it is just needed to prove that

$$g_1(t) \cdot g_2(r+t) + (\frac{1}{c-1})(1 - g_1(t)) \cdot (1 - g_2(r+t))$$

cannot decrease when r and t grow, which is fairly trivial because both g_1 and g_2 are non-decreasing. And we also have to show that:

$$g_2(r + t)$$

cannot decrease when r and t grow, which is even more trivial because both g_1 and g_2 are non-decreasing.

$\square$

That justifies the idea of *duplicating* examples from the training set. Provided the second classifier can be "teased" in this way, so forcing it to give more relevance to the training set. This means that we can have a new $t' = k \cdot t$ and a new $r' = k \cdot t$. Of course, repeating the training set may have not good effects (especially if we repeat the data several times), but at least there may be some situations where it can be beneficiial.

And we leave it here for this model. We leave as future work to obtain similar (or different) results with the other two models.

Of course there is a fourth model, which assumes that the first classifier is probabilistically pure and we still have the conditions of theorem 5, and hence we have that the mimicking method in much more cases. It is easy to see that the results can be adapted to the case where we have also the training set. Let us show the lemma also holds

Lemma 3. *Given a mimicking scenario where the training set is used for the second stage classifier. If the first stage classifier (the oracle) is probabilistically pure and the second stage classifier is pure and loyal then the average-case learning curve of the second classifier follows the condition:*

$$if \quad |R| \to \infty \quad then \quad g_2'(|R| + |T|)) = 1$$

We use g_2' to denote that this good behavious only happens wrt. the oracle. We reserve g_2 to the learning curve wrt. the target function.

Proof. From theorem 1 just considering that any probabilistically pure classifier is also probabilistically pure when we add a finite set of cases (represented by T).

$\square$

And now, from this, we can assert the following theorem similar to theorem 5:

Theorem 7. *Given a mimicking scenario with training set T of size $t = |T|$. If the first stage classifier (the oracle) is probabilistically pure and the second stage classifier is pure and loyal, and the second stage uses both R and T then the mimetic classifier will be useful if and only if:*

$$g_2(t) < g_1(t)$$

Proof. From theorem 6 we have:

$$g_2(t) < \frac{r}{r+t} \cdot [g_1(t) \cdot g_2'(r,t) + (\frac{1}{c-1})(1 - g_1(t)) \cdot (1 - g_2'(r,t))]$$
$$+(1 - \frac{r}{r+t}) \cdot g_2'(r,t)$$

Considering we can generate as many random examples as we want then we have $|R| \to \infty$. Hence, from lemma 3 we have:

$$g_2(t) < \frac{r}{r+t} \cdot [g_1(t) \cdot 1 + (\frac{1}{c-1})(1 - g_1(t)) \cdot (1 - 1)]$$
$$+(1 - \frac{r}{r+t}) \cdot 1$$

but also $r \to \infty$ and hence:

$$g_2(t) < 1 \cdot [g_1(t) \cdot 1 + (\frac{1}{c-1})(1 - g_1(t)) \cdot (1 - 1)]$$
$$+0 \cdot 1$$

which yields:

$$g_2(t) < g_1(t)$$

□

Note that in this case, we have the same result with and without the use of the training set, but the previous results tell that, even in this ideal situation, we would need less examples, in general, if we use the training set in the second stage th, than if we do not use them.

5.3 Additional Remarks

The previous theorem also shows that a bad classifier can perform better for small dataset and can be used as an oracle for a good classifier that performs well for larger samples. For instance, it seems that neural networks + decision tree learning may not be a good combination (because neural networks usually require a large number of examples), but that the combination ensemble + decision tree can be a good combination (since ensemble methods avoid overfitting and are good for small samples).

Finally, there are some details we have left out in the previous discussion. First we have assumed that a random dataset R and the training set T have no common examples. This is easy if we have numeric attributes and exhaustive random generator methods, but may not be a reasonable assumption in other cases. If there are common examples, it may happen that both classes are the same[4]. In this case, the previous results are practically identical to what have been shown. On the contrary, if the class of the example e in R is different from the class of e in T, then we can do three different things for constructing the "invented" dataset:

[4] This is the only possible case if the first stage algorithm is loyal.

- correct the errors in R with those of T and just use the good examples.
- maintain both datasets (now they will be inconsistent).
- maintain only the class of R.

It seems reasonable to consider the first option better. If we do this, the previous results still hold, since this correction is good, and the use of the training set (additionally to the random dataset) is reinforced, because the first term of the expression in theorem 6 will be increased slightly.

A second issue is that the examples in the training set are obtained from a real prior distribution, whereas the examples in R are generated, at best, with a simulated prior distribution. That means, that it is likely that examples similar to those in T are more probable (according to the target distribution) that examples similar to those in R. Once again, this will affect positively the use of the training set in the second stage.

6 Co-Training and Ranking

Other scenarios could also be considered, such the use of existing unlabelled data. This situation is obviously better, since the given unlabelled data complements the prior distribution given by the training set. Techniques from the field of learning from unlabelled labelled data and, especially from co-training, could be used here [1]. In fact, mimicking can be seen as an asymmetrical (i.e. one way) co-training where one of the two classifiers is considered much better than the other and where unlabelled data is generated rather than obtained from a pool.

In particular we are interested in the notion of ranking the examples in such a way that more confident predictions are chosen first for the random dataset. So the idea is to generate a random dataset where the more reliable predictions for the oracle have greater probability. A way to achieve this is to use the oracle as a probability estimator rather than a classifer. A probability estimator for classification or, simply, a soft classifier, is a model that accompanies each prediction with a confidence or even better it gives a probability for each class. If we consider oracles taht can provide with one of these modalities, we can *improve* the example generation. Let us see how:

First, we generate a large set of n examples given whatevever method seen before, for instance the prior interpolating distribution. Then we rank these examples according to the class probability estimation of the predicted class or the reliability given by the oracle. The rank i goes from 1 (the best ranked) to n (the worst ranked).

From here, two different methods for extracting the final random dataset of size r can be used:

- **Absolute Ranked Bootstrap**: Select the r best-ranked examples. The problem of this method is that it is not exhaustive (some zones in the space can be never chosen because the oracle behaves badly there). Moreover, if $n >> r$ then this is more likely to happen. Consequently, the relation between n and r is very important in this method.

- **Probabilistically Ranked Bootstrap**: Select r examples probabilistically with probability $Prob(e) = 2(n-i+1)/(n(n+1)))$ where i is the ranking of the example e. This method is exhaustive and the results will be more or less the same independently of the relation betwen n and r. Obviously there are other probabilities that can weigh the ranking more or less, but this is a first and easy approach.

And now, can we show under certain circumstances that the generation of examples in this way is better?

The idea is to think that the oracle has an overall fidelity or accuracy of a_1. However, if we select the best ranked with more probability, we could have a better fidelity $a_1' > a_1$. As long as this has no bad consequences for the accuracy of the second stage, i.e., for a_2, this can be considered a good idea.

However, according to the probabilistically ranked bootstrap, it may happen that as long r becomes higher, then there will be examples everywhere and then the mimetic classifier will capture so well the semantics of the whole oracle and not only the part that works well.

Consequently, a different approach could be just to establish a confidence threshold and select only the examples with threshold greater that this.

- **Threshold Bootstrap**: Select an example. If the reliability is lower than the threshold h than discard the example. Generate examples in this way until the number of examples is r. This method is not exhaustive.

From all the previous options, it seems difficult to assess theoretically which of them is best. And for the last one, the best threshold has to be established experimentally.

Nonetheless, any of the previous options could be used to simulate the behaviour, just considering that it is reasonable to think that using these variations the g_1 (as is seen from R) will be better (more precisely it will have a greater learning ratio and even at the beginning).

This can be model through a new $g_1(r,t)$ that "depends" on r, but in this case depends on t positively (as before) but negatively on r because the greater the random dataset we will have to include not only well-ranked predictions but also bad predictions. This adds an extra difficulty and more assumptions to the previous models and we think that it is not worthy being considered. For the moment.

7 Discussion and Conclusions

In this work we have analysed some theoretical issues concerning the mimicking method. The first part has been devoted to analyse the following questions:

- **If we have a good oracle, can we approximate to it with a different classifier?** The answer has been yes, under a few reasonable assumptions.
- **Do the assumptions hold for decision trees?** The answer is yes.

The second part has been devoted to analyse the following questions:

- **If I know the accuracies of the oracle and the fidelity of the mimetic classifier, can I compute the overall accuracy?** We have given a straightforward formula to do this.
- **Is it useful to add the training set to the random dataset for the second stage?** The answer is yes.
- **Which are the cases where the mimicking method is not good?** That depends on the model, but the only situation where mimicking can be desestimated is when the accuracy of the oracle, obtained from the training set only, is worse than the accuracy of the mimetic classifier, obtained from the training set only.
- **Given a training set and learning curves of two classifiers, how many random examples may I need to have certain accuracy?** We have given an equation to solve this kind of problems.
- **How should be the first learning algorithm and the second learning algorithm in a mimicking scenario?** The first one should be quick. The second should be highly accurate with large dataset. A loyal fence-and-fill algorithm for the second stage would be a good choice since the first one is usually pure.

Obviously, there are many other open questions. Some questions related to the methodology used in this work are:

- Question: if R is not a set but a multiset we have duplicate examples. How does it affect? We guess it doesn't matter for the previous results.
- Question: if T is not a set but a multiset we have duplicate examples. How does it affect? We guess it doesn't matter for the previous results.
- Question: if we consider numeric = rational instead of numeric=real, which conditions can be relaxed? Do we need the probabilistically pure condition?

But before addressing these issues, we think we have proposed some models and new random generation methods that should be examined experimentally, This come-and-go situation between theoretical and experimental results will hopefully give a more precise portrait on how mimetic classifiers work and what are their limits.

This work cannot be mimicked without the permission of the authors.

8 Acknowledgements

We would like to thank the help of Carlos Monserrat for some discussions about the convexity of the evidence space and the conditions for discretisation, and Cesar Ferri for several comments on a draft of this report.

References

1. A. Blum and T. Mitchell. Combining Labeled and Unlabeled Data with Co-Training. In *Proc. of the 1998 Conf. on Computational Learning Theory*, 1998.

2. V. Estruch, C. Ferri, J. Hernández, and M.J. Ramírez. Simple mimetic classifiers. In *3rd Machine Learning and Data Mining Conference, MLDM03*, volume to appear of *Lecture Notes in Computer Science*, 2003.
3. E.M. Gold. Language Identification in the Limit. *Information and Control*, 10:447–474, 1967.
4. H. Gu and H. Takahashi. How bad may learning curves be? *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(10):1155–1167, 2000.
5. D. Haussler, M. Kearns, M. Opper, and R. Schapire. Estimating average-case learning curves using bayesian, statistical physics and vc dimension methods. *Advances in Neural Information Processing Systems*, 4:855–862, 1992.
6. D. Walpert and W. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, pages 67–82, 1997.