Representation Does Matter

John Abela

Department of Artificial Intelligence, University of Malta
jabel@cs.um.edu.mt

Abstract. In Machine Learning the main problem is that of learning a 'description' of a class (possibly an infinite set) from a finite number of positive and negative training examples. For real world problems, however, one must distinguish between the actual instance of the class to be learned and the numeric or symbolic encoding of the instances of the same class. The question here is whether different encodings (or representations) of the instances of a real-world class can actually affect the performance of the learning algorithm. In artificial neural networks (ANNs), for example, it is required that the classes are always encoded as vectors over some field (usually the set of reals). In this paper it is argued that the representation of the class instances plays a very important role in machine learning since it has bearing on two very important issues — the *structural completeness* of the training set and also the *inductive bias* of the learning algorithm.

1 The Learning Problem

A learning program is a program that improves its performance with experience. To explain how this is done let us first give give an informal definition of the learning problem. Let O be a domain of discourse and let $\mathcal{C}$ be a, possibly infinite, set of related classes in O. Let C be a class in $\mathcal{C}$ and let C^+ be a finite subset of C and C^- be a finite subset of O whose members do not belong to C. We call C^+ the *positive* training set and C^- the *negative* training set. The learning problem is then to find, using C^+ and C^-, a class description for C. Of course, in practice, this might be, for all intents and purposes, impossible since if the number of classes in $\mathcal{C}$ is infinite, then C^+ may be a subset of infinitely many classes in $\mathcal{C}$. In other words, no finite subset, on its own, can characterize an infinite set. We therefore insist only on finding a class description for some class $C' \in \mathcal{C}$ such that C' *approximates* C. This depends, of course, on our having a satisfactory definition of what it means for a class to approximate another.

The above definition covers one of the most common learning tasks. This is the task of finding a function that maps objects that are considered 'similar' to the same class value. As Sammut explains in [Sam94], this is the so-called *categorization problem* referred to by Bruner, Goodnow, and Austin in their seminal 1956 work [BGA56]. To quote Sammut:

> *Learning experience may be in the form of examples from a trainer or the results of trial and error. In either case, the program must be able to represent its observations of the world, and it must also be able to represent the hypotheses about the patterns it may find in those observations. Thus, we shall often refer to the* observation language *and the* hypothesis language. *the observation language describes the inputs and outputs of the program and the hypothesis language describes the internal state of the learning program, which corresponds to its theory of the concepts and patterns that exist in the (training) data.*
> *The input to a learning program consists of descriptions of objects from the universe and, in the case of supervised learning, an output value associated with the example. The universe can be an abstract one, such as the set of all natural numbers, or the universe may be a subset of the real world. No matter which method of representation we choose, descriptions of objects in the real world must ultimately rely on measurements of some properties of those objects, for example the length of time a person has been employed for the purpose of approving a loan. The accuracy and reliability of a learned concept depends heavily on the accuracy and reliability of the measurements. A program is limited in the concepts that it can learn by the representational capabilities of both the observational and hypothesis languages.*

The observation language is therefore used to encode instance of the universe of discourse while the hypothesis language is used to describe the concepts that are learned (from the data). Learning is therefore reduced to the task of searching through the space of all sentences in the hypothesis language for the sentence (concept) that is *consistent* with the training data.

2 What is Representation?

The issue of representation deals with the encoding, into some mathematical structure, of the domain of discourse of some learning problem. We cannot overemphasize the fundamental, and very important, distinction between the objects themselves in some domain of discourse and their representation in the observation language, i.e. the numeric or symbolic encoding of the objects (as strings, numbers, vectors, graphs, etc.). Consider, for example, the set of all humans. A human can be encoded, or represented, in a number of ways;

1. **Bitmap encoding**, i.e. a picture (binary image) of a human,
2. **Genome**, a string containing the DNA sequence of a human,
3. **2-D vector**, e.g. (weight, height), and
4. **Attribute n-tuple**, e.g. race, complexion, colour of hair, etc.

Depending of what classes we want to consider in a given domain of discourse, an encoding may or may not be appropriate for the purpose of class description. It was shown in [Sch92], for instance, that a 2-D vector (weight, height) is,

in general, sufficient to distinguish the class of male humans from the class of female humans. In fact, when the vectors are plotted in the Cartesian plane it is easy to see that the two classes form two recognizable clusters. It is also conceivable that one can distinguish between males and females from bitmap representations. If, on the other hand, we want to consider the class of humans who have *minor thalassaemia*, a completely asymptomatic congenital condition of the blood, then such a 2-D vector will not do. Neither would a bitmap. The genome representation of the human would, in this case, be required. This is because people with minor thalassaemia are not externally distinguishable but their DNA contains the thalassaemia gene.

A representation may be better than another because it 'stores more information' relevant to the class in question. It has been argued that there is no form of representation that can 'store all the information' about a given object in the domain of discourse. The closest thing to such a perfect form of representation might be the hypothetical 'transporter' device in the *Startrek* TV series. The transporter device can, allegedly, 'dissolve' a human into the constituent atoms, encode the human is some form, transfer this information to a remote location, and then reconstruct the exact human, from different atoms, complete with identical DNA, thoughts, feelings, emotions, memory, experiences, etc.

Another argument why one particular representation may be better than another is that the regularities of a particular class might not be 'visible' under some representations [CT95]. We illustrate with an example.

String	**Gödel Number**
ab	4
aabb	900
aaabbb	889,350
aaaabbbb	8,687,348,670

Table 1: Some strings from $a^n b^n$ and their Gödel Numbers.

Table 1 shows, in the first column, four strings from the language $a^n b^n$. It should be easy to anyone with a basic knowledge of formal languages to guess from which language the strings are drawn. The second column shows the Gödel number of the same strings. All we have done is to re-encode the strings as natural numbers. We have used the Gödel number encoding of the strings. This mapping is deterministic and computable in polynomial time. The regularity that was previously visible in the strings now 'seems' to have disappeared. It is clearly more difficult for humans to 'guess' the correct class if the encoding is not 'right'. But why? Are learning algorithms also sensitive to the 'appropriate' choice of representation?

As Clark and Thornton explain in [CT95]:

> *Some regularities enjoy only an attenuated existence in a body of training data. These are regularities whose statistical visibility depends on some systematic re-coding of the data. The space of possible re-coding is, however, infinitely large — it is the space of applicable Turing machines.*

Many learning algorithms, such as BP ANNs and C4.5, accept only one form of representation of the input data. This means that they allow only one observation language and the implication of this is that the practitioner must always use the same encoding (representation) of the domain of discourse. It is well known that artificial neural networks (ANNs) accept only vectors over the field of reals as input. The questions we ask is: *Does the choice of representation affect learning*? In other words, are learning algorithms sensitive to re-encodings of the domain of discourse and do these re-encodings effect the learning process in any way?

3 Does Representation Matter?

In the early days of A.I. (i.e. 1960s to 1980s) most researchers agreed that representation was an important issue in cognitive science. Recently this assumption has been questioned. As Thornton explains in [Tho03b],

> *Now there is growing disagreement over the relevance of representation and a steadily deepening polarization of views with respect to the necessity of employing it in cognitive machinery.*

An argument very often used by those who do not see representation as a crucial issue is that if a class in a given domain of discourse can be learned in polynomial time under one encoding (observation language) then it should be learnable in polynomial time in another encoding as long as the mapping from one encoding is deterministic and is computable in polynomial time. This argument, *prima facie*, makes sense. Suppose we have a GI algorithm that learns the language $a^n b^n$ from a finite number of examples in time polynomial in the size of the training set. If we can re-encode the strings as natural numbers using, for instance, the Gödel encoding of the strings, then there must be a polynomial algorithm to learn the same class. Before we present our objections and reservations to this argument we must first clarify a number of points.

1. It is always assumed that, under any reasonable encoding, a class is always *computable*. If, for example, our domain of discourse is the set of all animals and we want to consider the class of cats, then if we encode the animals as strings in some observation language, the set of strings corresponding to cats must be a computable language. This assumption is fundamental since if a class is not computable (under some encoding) then it cannot have a finite description (a sentence in the hypothesis language).
2. If a class is computable and has a finite description under some encoding E then it will still be computable and have a finite description under encoding E' as long as there is a deterministic and computable mapping from E to E'.

> If the language $a^n b^n$ is a computable language and has a finite description then the set of natural numbers which are Gödel encodings of the strings of this language must also be a computable set with a finite description. The finite description of $a^n b^n$ is a formal grammar while the description of the associated set of Gödel numbers is a set of μ-recursive functions. It is well known from the theory of computation that graphs can be encoded as strings and string as numbers. If fact, for every computable language, the set of Gödel number encodings of the strings in the language is itself a computable set [Sud97].

The argument of those who do not see representation as an issue is that any representation can be used as long as the class to be learned has a finite description in the hypothesis language. This argument is used mostly by those in the connectionist community. This is probably because artificial neural networks, ANNs, only accept vectors as input. Although the above is, in theory, true we believe that researchers in the connectionist and machine learning communities who claim that representation is not an important issue are missing some important points.

Domain	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
0 0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0 1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1 0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1 1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

Table 2: All the functions with two binary-valued parameters.

We are not claiming that classes become 'unlearnable' if the representation is changed. What we are claiming is that the learning problem *changes if the representation changes*. Let us consider again the language $a^n b^n$ and the set of strings drawn from this language shown in Table 1. It is conceivable that one can design a simple GI algorithm that learns languages of this type. Now what if one is given only the Gödel encodings of this language? Does learning become harder? We claim that these are two different learning problems. If one encodes the strings as natural numbers, the class then changes too. In other words, the specification of the class is tied to the form of representation. The set of natural numbers that are Gödel number encodings of the strings in $a^n b^n$ is no longer the class $a^n b^n$. This set of numbers is computable and does have a finite description (a set of μ-recursive functions) but might require a completely different learning algorithm with a different inductive bias and a different search strategy. This may go some way towards explaining why neural networks perform so badly in certain applications. It has been claimed that a neural network (of the appropriate size) can approximate an computable function in the Euclidean space $\mathbb{R}^n$. In other words, given an ANN of the appropriate size, a set of weights exists that

will compute the function. Even if this is true the problems remain. When one chooses a particular architecture and size of a neural network one is putting an implicit bound on the size of the class description. Suppose, for example, that we have an ANN with n weights to learn a class C. We are then implicitly assuming that C can be described (or parameterized) with n real numbers. This issue has recently received attention from the connectionist community [LGC96].

To demonstrate this we conducted a number of experiments with backpropagation (BP) ANNs. We tried to train a Backpropagation neural network with the complete data sets for 8, 10, 12, and 16-bit even parity mapping. By *complete* we mean that for each n-bit parity we labeled each of the 2^n strings with a T (True) or F (False). The reason we tried complete data sets was to find the optimal network size, i.e. number of weights, for each of the mappings. For the 8-bit parity data set we first tried an $8 \times 8 \times 2$ network. This did not converge after running for many hours. We then tried changing the various learning parameters but to no avail. We increased the number of hidden neurons to 16 and then to 32 and got very much the same results. We finally tried a network with 64 hidden neurons and the network converged within a few seconds. This network had $(8 \times 64) + (64 \times 2) = 640$ weights. The ANN for 10-bit parity converged in just over 3 mins and required 256 hidden neurons while the ANN for 12-Bit parity required 1024 hidden neurons and and converged in just under 12 minutes. We could not use Backpropagation to learn the 16-bit parity problem since the ANN software we used, Brainmaker®, was limited to 1024 hidden neurons. We then used a Cascade Correlation neural network to learn the 16-bit even parity mapping. The Cascade Correlation network changes its architecture during learning. We then tried to train the networks on incomplete data, sometimes with data sets that contained up to 85% of the 2^n possible strings. The networks never generalized correctly from incomplete data. This phenomenon has been observed many times [Tho03a]. We were not surprised at the neural network's inability to learn from incomplete parity data. There is absolutely no reason why a learning algorithm should generalize correctly from incomplete parity data *unless the algorithm has the right inductive bias*. This point is, most unfortunately, not always well understood. For any given training set of incomplete parity data, there might be thousands of mappings that are consistent with the training set. Why should then the neural network converge to the parity mapping? It will only do so if it has the correct inductive preference bias. Our experiments with the complete parity data sets demonstrated that a Backpropagation neural network can, in fact, represent the parity mapping. In other words, given an ANN of appropriate size, there exist a set of weights that can describe (or parameterize) the parity mapping. Generalization from incomplete training sets is a totally different issue. Neural networks do not learn the parity mapping precisely because they do not have the correct bias. Moreover, the inductive bias of a neural network is fixed and can only be changed slightly by modifying some learning parameters. It is therefore unlikely that anyone can convince a Backpropagation ANN to generalize the parity problem correctly from incomplete training ex-

amples. Some researchers, including Thornton [Tho03a], have proposed various reasons as to why this happens.
Table 2 shows the complete list of mappings with two binary input values. Suppose we would like to teach a learning algorithm the AND function (function 2) and, to do this, we prepare the following training set (0,0 → 0) and (0,1 → 0). This is clearly an incomplete training set for the AND function. There is absolutely no reason why a learning algorithm should find the correct function from this training set. The reader should note that functions 1, 2, 3, and 4 all are consistent with the training examples. Why would a learning algorithm choose function 2 (AND) over the others unless it had *exactly* the right inductive preference bias?

It turns out that, for any given n, the set of binary strings that are odd or even parity form a *kernel* language [Abe02]. For example, 4-bit even parity can be described by the following evolving transformation system (ETS) [Gol90] description:

Kernels: ε, 11, and 1111.
Transformations: $0 \rightarrow \varepsilon$ (Weight 0.0) and $1 \rightarrow \varepsilon$ (Weight 1.0)

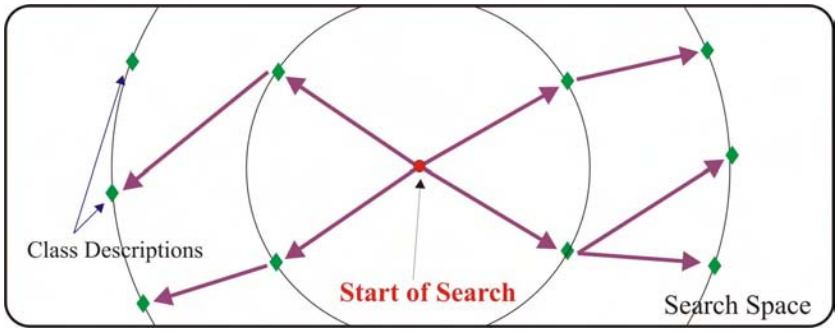

Fig. 1: Enumeration of the search space according to the inductive bias of the learning algorithm.

We ran an ETS grammatical inference (GI) algorithm on a number of incomplete 16-bit parity data sets — *par01* had no noise and *par02* had 2% misclassification noise. In each case the algorithm found the correct ETS description is less than 3 minutes on a training set of several thousand strings. The training examples were exactly the same that were used, unsuccessfully, to train the ANNs.

par01 16-bit parity-problem training set (even parity) used to train Backpropagation neural network. Binary alphabet, very large training set ($\approx 10,000$), multiple kernels, no noise, confluent.
Kernels: *11*, *1111*, *111111*, *11111111*, *1111111111*, *11111111111*,

1111111111111, and *111111111111111*.
Features: *0*, and *1*.

par02 16-bit parity-problem training set (even parity) used to train Backpropagation neural network. Binary alphabet, large training set ($\approx 8,000$), multiple kernels, 2% misclassification noise, confluent.
Kernels: *11*, *1111*, *111111*, *11111111*, *1111111111*, *11111111111*, *1111111111111*, and *111111111111111*.
Features: *0*, and *1*.

The ETS algorithm can learn the parity mapping even with very small, structurally complete, data sets of less than 20 strings. We must emphasize that this is because the ETS algorithms so happens to have the right inductive preference bias. When the algorithm learns, it first considers the 'simple' ETS descriptions, i.e. the ETS description with the least number of features. It so happens that the parity problem has a very simple ETS description and this was very close to the algorithm's starting point in the search space. This is depicted in Figure 1. It might turn out that under some other representation, the target class description will be 'far away' from the starting point in the search space. The phenomenon was evident when the author was considering ETS learning of chain-code picture languages [Abe96]. A picture such as a rectangle could be represented by its string contour encoding. It this case, the language of all strings that are contour encodings of rectangles (or any other figure) would be a context-sensitive language — sometimes with many productions. A GI learning algorithm that used Occam's bias would have a hard time learning these classes. This because the algorithm would first consider grammars with small number of productions before grammar with a large number of productions. The target grammar would therefore be 'far away' from the starting point in the search space. The same class of figures encoded as graphs could be represented by a very simple graph grammar. This would arguably be much closer to the starting point of the search if one used a learning algorithm with Occam's bias to search the space of graph grammars. We also strongly feel that the 'correct' inductive bias for a given learning problem depends on the choice of representation. If the representation is changed then a new, completely different bias might be required. Most learning algorithms search through the space of class descriptions by first considering the simple class descriptions and progressing to more complex class descriptions. A DFA learning algorithm might first consider DFAs with 1 state, then DFAs with 2 states, and so on. In other words, the algorithm exploits an ordering of the space of class descriptions (i.e the search space). This bias is another example of Occam's bias. In essence, a learning algorithm's inductive preference bias specifies the method of enumerating the space of class descriptions. The algorithm stops when it finds a class description consistent with the training set. If one changes the method of enumerating the space, a different class description may be found. This is because, in general, the search space may contain several class descriptions consistent with the training set. In the case of parity problem, a neural network finds the first set of weights consistent with the set of incomplete parity data. It so happens that the mapping

found by the network is not the correct (i.e. parity) mapping. This means that the network does not have the 'right' bias for the parity problem.
Wolpert [WM95] and many others have shown that no inductive bias can achieve a higher generalization accuracy than any other bias when when considered over all classes in a given domain. In spite of this, it has been documented that certain bias do perform better than average on many real-world problems [Tho99]. This strongly suggests that many real-world problems are homogenous in nature in that they require very similar inductive biases. This explains why certain learning algorithms such as ID3 do well on most applications. When learning algorithms do badly it is very often a case of incorrect inductive bias. The answer, of course, is to use a learning algorithm that has a variable inductive preference bias. With these algorithms the user can, to a certain extent, change the algorithm's bias by modifying a number of parameters.

4 Conclusions

In this paper we have argued that the choice of representation (the observation and hypothesis languages) is an important one. The choice of the observation language is important since this determines which concepts can be learned [Sam94]. The choice of the hypothesis language is also of importance since, as Goldfarb explains in [Gol90], the class description learned must be *communicable*. As Sammut explains:

> *A representation that is opaque to the user may allow the program to learn, but a representation that is transparent also allows the user to learn*

Learning algorithms that fix the observation and hypothesis languages, such as BP ANNs, are limited in the concepts that they can learn. The choice of representation influences the inductive bias of the learning algorithm and also the structural completeness criteria of the training data. Given a real world problem therefore, the choice of the observation and hypothesis languages must be very carefully chosen.

References

[Abe02] Abela, John, *ETS Learning of Kernel Languages.* Ph.D. Thesis, University of New Brunswick, Canada, 2002.

[ABKG95] Abela, J., Bhavsar, V., Kamat. V.N., and Goldfarb, L., *Can a Vector Space Based Learning Model Discover Inductive Class Generalisation in a Symbolic Environment?* Pattern Recognition Letter 16(7), pp. 719–726, 1995.

[Abe96] Abela, John, *Learning Picture Languages*, Technical Report TR-CS-9605, Department of Computer Science and Artificial Intelligence, University of Malta, 1996.

[BGA56] Bruner, J.S., Goodnow, J., and Austin, G.A., *Study of Thinking* John Wiley and Sons, NY, 1956.

[CT95] Clarke, A. and Thornton, C., *Trading Spaces: Computation, Representation, and the Limits of Uniformed Learning*, Cognitive and Computing Sciences, University of Sussex, Brighton, UK, 1995.

[Gol90] Goldfarb, Lev. *On the Foundations of Intelligent Processes — 1: An Evolving Model for Pattern Learning.* Pattern Recognition, 23, pp. 595–616, 1990.

[LGC96] Lawrence S., Giles, L., Chung Tsoi, A., *What Size Neural Network Gives Optimal Generalization — Convergence Properties of Backpropagation*, Technical Report, UMIACS-TR-96–22 and CS-TR-3617, Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742, 1996.

[MCM83] Michalski, R., Carbonel, J., Mitchell, T., (eds). *Machine Learning — An Artificial Intelligence Approach.* Morgan Kaufmann Publishers Inc. 1983.

[Ros73] Rosch, Eleanor, H., *On the Internal of Perceptual and Semantic Categories.* in Timothy E. Morre, ed., *Cognitive Development and the Acquisition of Language*, Academic Press. 1973.

[Sam94] Claude Sammut *Knowledge Representation.* in D. Michie, D. J. Speigelhalter and C. C. Taylor (Eds.), *Machine Learning, Neural Nets and Statistical Classification* pp. 228–245. Ellis Horwood, 1994.

[Sch92] Schalkoff, R., *Pattern Recognition — Statistical, Structural, and Neural Approaches*, John Wiley and Sons, Inc., 1992.

[SA90] J. Stender, and T. Addis (eds). *Symbols vs Neurons.* IOS Press, Amsterdam, 1990.

[Sud97] Sudkamp, Thomas, *Languages and Machines*, Addison Wesley Longman, 1997.

[Tho99] Thornton, Chris *There is No Free Lunch but the Starter is Cheap: Generalisation from First Principles* Cognitive and Computing Sciences, University of Sussex, Brighton, UK, 1999.

[Tho03a] Thornton, Chris *Parity: The Problem that Won't Go Away* Cognitive and Computing Sciences, University of Sussex, Brighton, UK, 2003.

[Tho03b] Thornton, Chris *Re-presenting Representation* Cognitive and Computing Sciences, University of Sussex, Brighton, UK, 2003.

[WM95] Wolpert, D., and Macready, W. *No Free Lunch Theorems for Search*, Unpublished MS, 1995.