Machine Learning, Special Issue on Unsupervised Learning, 1?? (to appear)

Supervised versus Unsupervised Binary-Learning by Feedforward Neural Networks

NATHALIE JAPKOWICZ nat@cs.dal.ca

Faculty of Computer Science, DalTech/Dalhousie University, 6050 University Avenue, Halifax, N.S., Canada, B3H 1W5

Editor: Douglas Fisher

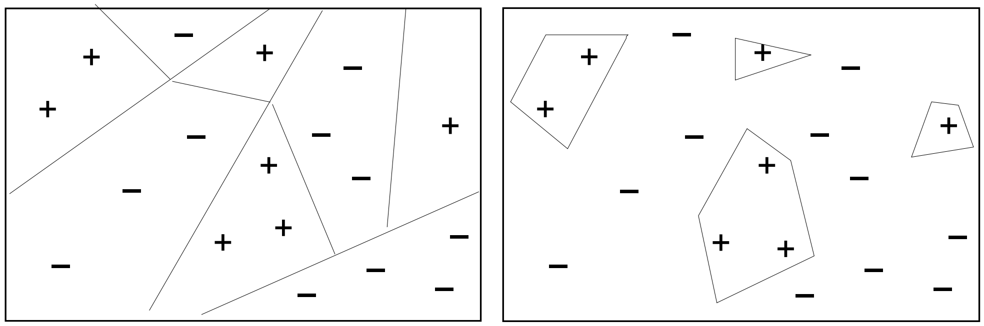
Abstract. Binary classification is typically achieved by supervised learning methods. Nevertheless, it is also possible using unsupervised schemes. This paper describes a connectionist unsupervised approach to binary classification and compares its performance to that of its supervised counterpart. The approach consists of training an autoassociator to reconstruct the positive class of a domain at the output layer. After training, the autoassociator is used for classification, relying on the idea that if the network generalizes to a novel instance, then this instance must be positive, but that if generalization fails, then the instance must be negative. When tested on three real-world domains, the autoassociator proved more accurate at classification than its supervised counterpart, MLP, on two of these domains and as accurate on the third. The paper seeks to generalize these results and concludes that, in addition to learning a concept in the absence of negative examples, 1) autoassociation is more efficient than MLP in multi-modal domains, and 2) it is more accurate than MLP in multi-modal domains for which the *negative class* creates a particularly *strong* need for specialization or the *positive class* creates a particularly *weak* need for specialization. In multi-modal domains for which the *positive class* creates a particularly *strong* need for specialization, on the other hand, MLP is more accurate than autoassociation.

Keywords: Unsupervised Learning, Supervised Learning, Discrimination, Recognition, Class Imbalance Problem, Feedforward Neural Networks, Autoassociation.

1. Introduction

Binary-learning can be approached in one of two ways. The first way considers both positive and negative examples of a concept and learns to *discriminate* between the two classes. The second way consists of considering only positive examples of the concept and learning how to *recognize* those examples. Discrimination-based learning falls in the category of *supervised learning* since it requires *labeled* examples of *both* the positive and the negative class, and derives a discrimination function based on this information. Recognition-based learning, on the other hand, falls in the category of *unsupervised learning* since *unlabeled* data are fed to the learning system which is left to choose an internal organization on its own. Supervised and unsupervised approaches to binary-learning are illustrated in Figure 1.

Although supervised methods are usually favored in the fields of Pattern Recognition, Machine Learning, Neural Networks, and Data Mining (e.g., Binary Hypothesis Testing (Fukunaga, 1990), Multi-Layer Perceptrons (MLP) (Rumelhart, Hinton, & Williams , 1986), C4.5 (Quinlan, 1993), CART (Breiman, Friedman, Olshen, & Stone, 1984), Nearest-Neighbors (Fukunaka, 1990) , and IB4 (Aha, Kibler, & Albert, 1991)), an unsupervised connectionist method was recently shown to be



(a) Supervised (b) Unsupervised

Figure 1. The two approaches to binary learning: Supervised versus Unsupervised Learning. In supervised learning, boundaries are drawn *between* the positive and the negative examples. In unsupervised learning, boundaries are drawn *around* the positive examples.

more accurate at classification than its supervised counterpart—MLP—on two out of three real-world domains and as accurate on the third (Japkowicz, Myers, & Gluck, 1995). This unsupervised approach to classification consists of training a one-hidden layer, nonlinear autoassociator to reconstruct the positive class of a domain at the output layer. Once trained, the autoassociator is used for classification, relying on the idea that if the network generalizes to a novel instance, then this instance must be positive, but that if generalization fails, then the instance must be negative.

The approach was originally introduced by (Hanson & Kegl, 1987) and used to learn a natural language grammar in the absence of negative (i.e., labeled non-grammatical) examples. The purpose of this study was to model the language acquisition process in its early stages, when the child is only exposed to grammatical sentences. At this stage, the child is not yet able utter sentences that could be ungrammatical (and thus corrected), or, in machine-learning terms, labeled as negative. The success of the approach on this cognitive task encouraged researchers to test it on practical engineering tasks for which positive examples are available, but for which negative examples are very expensive or difficult to obtain (such tasks are said to suffer from the *class imbalance problem* (Kubat, Holte, & Matwin, 1998)). The approach proved successful on tasks such as helicopter gearbox fault monitoring (Japkowicz et al., 1995) and motor fault monitoring (Petsche, Marcantonio, Darken, Hanson, Kuhn, & Santoso, 1996).[1]

This paper suggests the reasons for the apparent advantage of autoassociation-based classification over MLP in selected domains. Our methodology consists of, first, verifying the results obtained in the experiments on real-world domains of (Japkowicz et al., 1995), and, second, analyzing the flexibility of the autoassociator by testing it and comparing its performance to that of MLP on a series of synthetic domains with controlled characteristics. Consequently, the three real-world domains that motivated our study are analyzed in light of the synthetic domains in an attempt to validate our conclusions and suggest means of fine-tuning our results.

The study concludes that, in the context of feedforward neural networks, unsupervised techniques can be more appropriate than supervised ones as far as binary learning is concerned. Indeed, there exist a number of abstract domain characteristics for which unsupervised learning is more accurate than supervised learning.

The remainder of this paper is divided into six sections. Section 2 describes the two connectionist schemes compared in this study. Section 3 describes the results obtained from re-applying the autoassociator and the MLP network to the three real-world domains of (Japkowicz et al., 1995), using a more robust experimental framework. Section 4 states the particular hypotheses studied in this paper. Section 5 discusses the experiments carried out and the results obtained in view of these hypotheses. Section 6 extracts the domain characteristics of the three real-world domains and matches them to the results of Section 5, thus, justifying some of our conclusions and deriving new research directions designed to fine-tune these conclusions. Section 7 discusses additional future work. The research conducted in this paper is based on Chapters 4 and 7 of (Japkowicz, 1999a) and it constitutes an expanded version of (Japkowicz et al., 1995) and (Japkowicz, 1999b).

2. Supervised versus Unsupervised Binary-Learning in Feedforward Neural Networks

A formal definition for the task of learning by feedforward neural networks typically involves an input vector x^j and a response vector y^j which are such that the pair (x^j, y^j) belongs to some unknown joint probability distribution, P. The goal of the network is to induce a function $f(x)$ from a set $(x^1, y^1), (x^2, y^2), \ldots\ (x^N, y^N)$ of training examples of P, so that if (x, y) belongs to P, then $f(x)$ approximates y.

An important property of these networks is that they can be applied to binary-learning tasks—tasks in which a *single* concept is learned from training data that are either examples of this concept or counter-examples. Under the supervised paradigm, response vector, y^j, represents the class of its associated input vector, x^j, whereas under the unsupervised paradigm, $y^j = x^j$. This will be discussed in more detail below.

Throughout this study, we restrict our attention to Multi-Layer Perceptrons with a single hidden layer, i.e., with two layers of weights. Two implementations of the binary-learning task using feedforward neural networks are illustrated in Figure 2. In figure 2(a), the input layer contains 6 units, the hidden layer, 3 units, and the output layer, a single unit. Figure 2(b) is similar except for the output layer which contains 6 units. The network of figure 2(a) can be used for supervised binary-learning as described in section 2.1 while the network of figure 2(b) can be used for unsupervised binary-learning as described in Section 2.2.

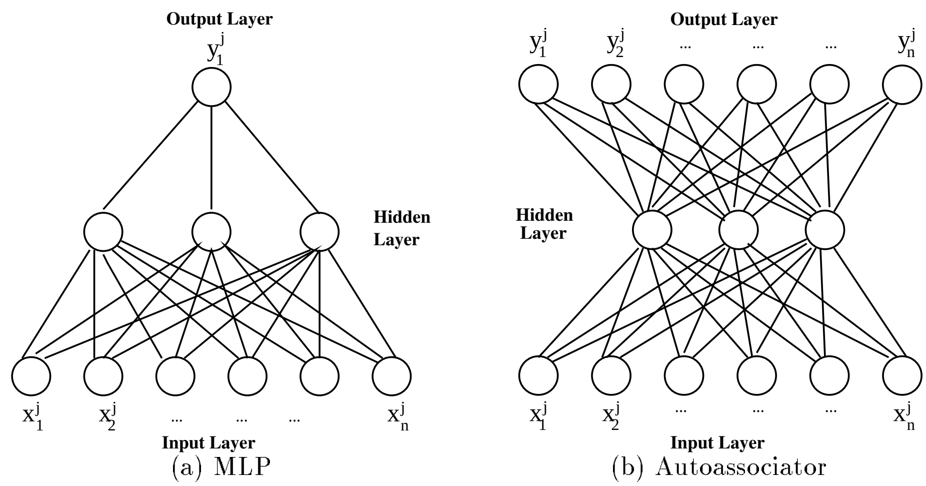


Figure 2. Examples of Feedforward Neural Networks: The network of figure (a) can be used in a supervised fashion while the network of figure (b) can be used in an unsupervised fashion for the task of binary-learning.

2.1. Supervised Learning

The network of figure 2(a), which we refer to as "MLP" or "the MLP network", is the most commonly used network in the field of connectionist systems. It is typically used to implement a supervised approach to binary learning. Using this scheme, single output y_1^j, the response vector associated with input vector x^j, is assigned the value of "1" or "0" according to whether or not x^j is an instance of the concept the network is trying to learn.

After training the network to approximate the function $f(x)$ from which the training examples are believed to have been generated, it is expected that, if the training set was appropriately designed, the network will be able to compute the appropriate label ("1" or "0") for any input vector of the size of the input layer, even if that vector did not appear in the training set. In all the experiments reported in this article, training was performed using the backpropagation procedure (Rumelhart et al., 1986).

2.2. Unsupervised Learning

The network of figure 2(b) is not as universal as the previous network, but it has been used previously to implement a compression scheme (Cottrell, Munro, & Zipser, 1987; Elman & Zipser, 1988). This network also can be used to implement an unsupervised approach to binary learning as proposed by (Hanson & Kegl, 1987). In this scheme, each response vector y^j is set to x^j, its corresponding input vector. In other words, the network is trained to reproduce the input at the output layer. Such a network is called an autoassociator (or autoencoder) and was originally introduced in (Rumelhart et al., 1986). For binary-learning to take place, the network is trained to reconstruct positive data only and classification is performed on a new vector x^{Test} by comparing its *reconstruction error*[2] to a threshold, and assigning it to the positive class if the reconstruction error is smaller than this threshold and to the negative class, otherwise. The idea behind this recognition-based unsupervised classification scheme is that since the autoassociator is trained to compress and decompress examples of the positive class only, when tested on a novel data point, it will compress and decompress it appropriately if this example belongs to the positive class, but it will not do so appropriately if the example does not belong to the positive class. As in the case of the MLP network, the autoassociator is trained using the backpropagation procedure (Rumelhart et al., 1986).

3. A Comparison of Supervised and Unsupervised Learning by Feed-forward Neural Networks on Three Practical Domains

In this section, MLP and autoassociation-based classification are compared and contrasted on a series of three real-world domains: helicopter gearbox fault monitoring, DNA promoter recognition, and sonar target recognition. Although the experiments reported here are heavily based on the work of (Japkowicz et al., 1995), this section extends its results in two ways. First it eliminates the bias created by the introduction in (Japkowicz et al., 1995) of a threshold-determination procedure. And, second, it justifies the choices made in (Japkowicz et al., 1995) of which class to consider as the "positive" class, i.e., the class to be recognized by the autoassociative process.

This section is divided into three subsections. Section 3.1 describes the three domains considered in our study, Section 3.2 discusses the experimental strategy used for our experiments and Section 3.3 lists their results.

3.1. Domain Description

To evaluate the performance of classification systems reliably, it is necessary to test them on several domains. The domains used in this section for such an evaluation are now described.

CH46 Helicopter Gearbox The CH46 Helicopter problem is a monitoring problem that consists of discriminating between faulty and non-faulty CH46 helicopter gearboxes, according to the whining sound they emit during their operation. These data were obtained from NRaD (Kolesar & NRaD, 1994). The sudden, unexpected failure of CH46 helicopter gearboxes can be very costly both in terms of lives and equipment. The development of a monitoring system that can identify imminent failures before takeoff or when in flight is of paramount importance. The data for this problem were obtained by pre-processing the vibration time signal of the gearboxes of various faulty and non-faulty helicopters. The complete data set is composed of 18 non-faulty instances and 46 faulty ones which come in the form of 256 long vectors of real numbers. In this particular problem, the non-faulty examples were chosen to represent the positive class.

Promoter The promoter recognition problem takes as input, segments of DNA, some of which represent promoters. A promoter is a sequence that signals to the chemical processes acting on the DNA where a gene begins. The goal of the problem is to train a classifier to recognize promoters, which are taken to be the positive class. The training set is composed of 100 examples (47 promoters and 53 non-promoters), each composed of a set of 51 nucleotides, where each nucleotide can take one of the four values {a, c, g, or t}. The promoter data were obtained from the U.C. Irvine Repository of Machine Learning and was modified in response to Norton's critique of the biological flaws underlying the original formulation of the data (Norton, 1994). In addition, as is usual for this problem when run on a connectionist system, each example was converted into a 204-bit long vector where each nucleotide was represented with 4 bits.

Sonar Target The sonar target detection problem takes as input the signals returned by a sonar system in the cases where mines and rocks were used as targets. The sonar data was obtained from the U.C. Irvine Repository of Machine Learning though only a subset of 100 instances (47 positive and 53 negative) from these data was used in this particular case study. The transmitted sonar signal is a frequency-modulated chirp, rising in frequency. The data set contains signals obtained from a variety of different aspect angles. Each instance of this data is represented as a 60-bit long vector spanning 90 degrees for the mine and 180 degrees for the rock. Each instance is a set of 60 numbers in the range 0.0 to 1.0. In this particular case study, the signals returned by the mine targets were the positive class.

3.2. Experimental Strategy

The technique used to assess the capabilities of the unsupervised versus the supervised scheme on the task of classification consisted of running both networks on a pre-defined space of possible capacity and stopping point values, evaluating the two methods over all possible threshold settings and reporting the best results obtained in both cases using a five-fold cross-validation testing scheme (Weiss & Kulikowski, 1991).

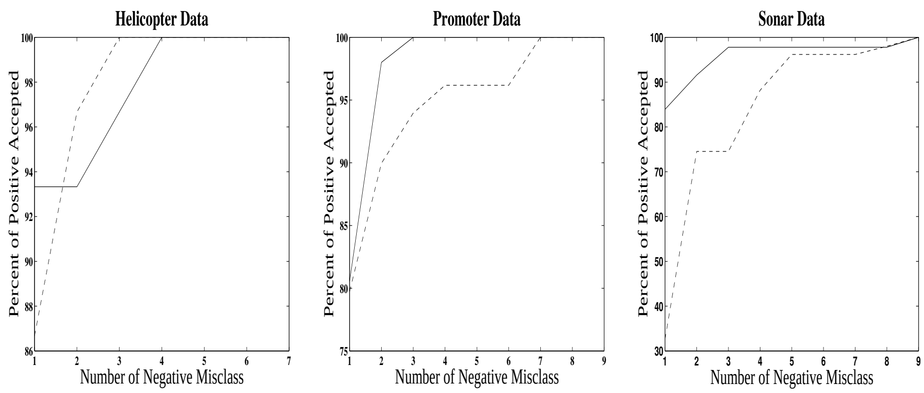
Our experiments divide the data from each domain into the same five-fold cross-validation sets as those used in (Japkowicz et al., 1995), and within each fold of each domain both networks were trained using backpropagation on the training partition—although the negative examples were removed in the case of the autoassociator—with 1, 2, 4, 8, 16, 32 and 64 hidden units for 500 epochs. Each network considered was then tested on the testing partition at epoch 10, 20, 30, etc. until epoch 500.

Within each paradigm (supervised or unsupervised) and within each fold of each domain, Receiving Operating Curve (ROC) Analyses were performed comparing all the combinations of network capacity and stopping point in order to select the optimal one. A ROC curve for a fold was built by successively recording the number of positive examples that would be correctly classified if the threshold between positive and negative data were such that only the n negative examples with highest output value (for the MLP network) and lowest reconstruction error (for the autoassociator) were misclassified. In these experiments, n was successively set to $1, 2, ..., MaxNeg$ where $MaxNeg$ represents the number of negative examples available in the testing set. The results obtained by each system and for each fold within a domain were averaged and these averaged results for both systems on that domain were then plotted on the same graph.[3] Within this graph, the curve located above the other represents the most accurate system.

It is worth noting that our methodology is slightly different from standard ones. Indeed, rather than determining, a-priori, a set of parameter and threshold values assumed to be optimal, and returning cross-validation error-rate results, we compared all the possible network parameter and threshold values of the MLP network to all the possible network parameter and threshold values of the autoassociator prior to reporting the optimal cross-validation error-rate results (note that we did, however, restrict the number of network parameters that could vary. In particular, only the network capacity and stopping point were allowed to vary. The learning rate and momentum were kept constant as is customary in the neural network literature). While it can be argued that our methodology does not guarantee that the results obtained can easily be mapped into practical situations, it has the advantage over the standard methodology of separating the issues of concept-learning, on the one hand, and capacity, stopping point and threshold determination, on the other. It thus eliminates many of the biases introduced by possibly arbitrary a-priori parameter and threshold selection. Given the small amount of data available for this study, these biases would have been particularly pronounced and, therefore, the results returned by our proposed a-posteriori method are much more reliable than those that would have been obtained using an alternate approach. In addition, this methodology has the advantage of providing us with a unified framework within which very different approaches such as a supervised and an unsupervised scheme can be compared fairly. The question of determining, a-priori, an optimal set of parameter and threshold values is, nonetheless, discussed in our section on Future Work. In particular, Section 7 will survey this issue in the case of the autoassociator when negative examples are missing.

3.3. Results

The results obtained by comparing the performance of the autoassociator to that of the MLP network on each of the three real-world domains described in Section 3.1 are displayed in Figure 3. Figure 4 displays the results obtained when inverting the roles of the positive and the negative class. In particular, Figure 3(a) displays the results obtained on the helicopter gearbox domain when training the autoassociator on the non-faulty class; Figure 3(b) displays those obtained on the promoter data when training the autoassociator on the promoter class ; while Figure 3(c) displays the results obtained on the sonar data when training the autoassociator on the mine class. Conversely, Figure 4(a) displays the results obtained on the helicopter domain when training the autoassociator on the faulty class; Figure 4(b) displays



(a) Helicopter Domain (b) Promoter Domain (c) Sonar Domain

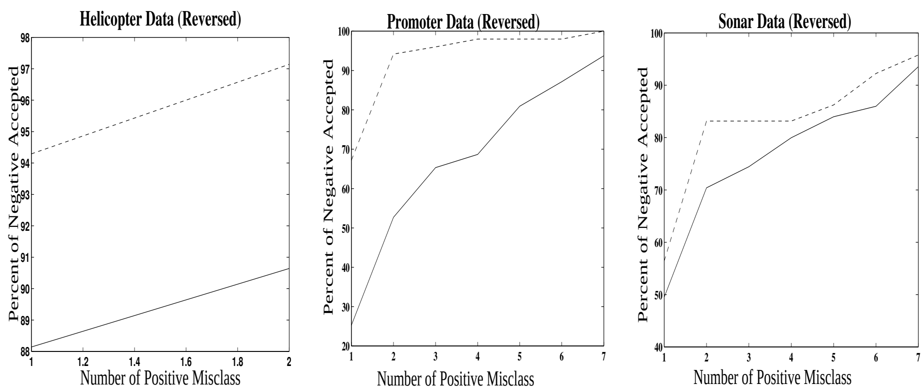
Figure 3. The result of a ROC analysis of MLP and the autoassociator on the three domains of (Japkowicz et al., 1995). MLP is represented by a broken curve while the autoassociator is represented by a full curve. The results show that MLP has the capability of being slightly more accurate than the autoassociator on the helicopter domain, but that the autoassociator has the capability of being more accurate than MLP on the other two domains.

the results obtained on the promoter domain when training the autoassociator on the non-promoter class; and Figure 4(c) displays the results obtained on the sonar domain when training the autoassociator on the rock class.[4]

The capacity and stopping criteria selected for each fold of each domain and by each network associated with Figure 3 are reported in Table 1 while those associated with Figure 4 are reported in Table 2. Each cell in these tables contains the parameters selected for each of the domains' five folds.

Table 1. Capacity and Stopping points selected for each of the experiments reported in Figure 3. The five values listed in each cell correspond to the figures for each of the five folds of each domain. The capacity values are expressed in terms of the number of hidden units while the stopping points correspond to number of epochs.

Type of Parameter	*Autoassociator*	*MLP*
Helicopter: capacity	1, 64, 1, 8, 1	1, 4, 1, 2, 1
Helicopter: stopping point	10, 310, 10, 70, 10	50, 370, 70, 40, 160
Promoter: capacity	2, 4, 8, 1, 2	1, 8, 1, 1, 8
Promoter: stopping point	30, 170, 100, 240, 150	20, 190, 20, 270, 100
Sonar: capacity	16, 32, 16, 32, 16	1, 32, 1, 1, 2
Sonar: stopping point	190, 40, 380, 140, 150	40, 30, 20, 200, 310



(a) Helicopter Domain (b) Promoter Domain (c) Sonar Domain

Figure 4. The result of a ROC analysis of MLP and the autoassociator on the three domains of (Japkowicz et al., 1995) where the autoassociator is trained on the negative rather than the positive class. MLP is represented by a broken curve while the autoassociator is represented by a full curve. The results show that MLP is more accurate than the autoassociator on all domains which demonstrates that the choice of the class on which the autoassociator is to be trained is extremely important.

Table 2. Capacity and Stopping points selected for each of the experiments reported in Figure 4. The five values listed in each cell correspond to the figures for each of the five folds of each domain. The capacity values are expressed in terms of the number of hidden units while the stopping points correspond to number of epochs.

Type of Parameter	*Autoassociator*	*MLP*
Helicopter: capacity	8, 16, 8, 8, 16	1, 4, 1, 2, 1
Helicopter: stopping point	110, 280, 350, 210, 150	50, 370, 70, 50, 160
Promoter: capacity	1, 1, 16, 8, 8	1, 8, 1, 1, 8
Promoter: stopping point	140, 30, 490, 20, 240	20, 190, 20, 270, 100
Sonar: capacity	4, 16, 4, 8, 8	1, 32, 1, 1, 2
Sonar: stopping point	20, 20, 310, 50, 60	40, 30, 20, 200, 310

For the results illustrated in Figure 3, the MLP network is capable of obtaining slightly better classification results than the autoassociator in the helicopter gearbox domain, but the autoassociator has the capability of outperforming the MLP network in the other two domains. This means that if a good capacity, a good stopping point and a good threshold can be established using only the training set (i.e., using only positive data in the case of the autoassociator), then the autoassociator is quite capable of learning a binary problem's classification in the absence of counter-examples. In addition, our experiment demonstrates that the autoassociator can even be more accurate than MLP which uses a supervised paradigm relying on both the positive and negative class. On the other hand, the results of Figure 4 show that when the faulty helicopter gearboxes, the non-promoters, and the rocks are considered to be the "positive" classes, MLP is capable of more accurate performance than the autoassociator in all three domains.

The results reported in Figure 3 are interesting and a bit surprising as they suggest that additional information brought in by the negative class may sometimes be more of a liability than an asset. As seen in the context of the experiments reported in Figure 4, however, this is not always the case: knowledge of the opposite class may sometimes be highly beneficial.[5] The question of when negative examples are an asset and when they are a liability is studied in Sections 4 and 5.

4. Hypotheses

To find out whether or not the results reported in Section 3.3 are meaningful in a more general context, we reflect on the nature of binary-learning when using a supervised or an unsupervised paradigm. A useful framework for understanding the induction process of a generic classifier is to decompose it into two opposing processes in search of an equilibrium. The first process seeks to *generalize* from the positive examples of the concept given to the classifier, while the second one seeks to *specialize* these examples. Generalization amounts to seeking a concept description which characterizes all the positive examples of the concept while specialization consists of seeking a concept description which excludes all its negative examples. This framework is illustrated in figure 5 and it is formalized in (Mitchell, 1982), using the notion of "Version Spaces".

In this generalization/specialization framework, supervised classifiers learn concepts by generalizing from positive examples while specializing using negative examples. While unsupervised learners are also able to generalize from positive examples, they do not, however, have access to negative examples for specialization. Their ability to specialize comes from a particular internal capability of theirs to bias the generalization process the way negative examples would, if considered.[6]

Given the nature of the supervised and the unsupervised paradigms, it is expected that the MLP network should be better suited than the autoassociator to learning domains that require a *strong* specialization bias caused by the *positive* class. Examples of such domains are those in which, when viewed separately from the negative class, a multi-modal positive class can be wrongly interpreted as uni-modal. In such domains, the MLP network's ability to rely on the negative examples during the inductive process allows it to discriminate between the unimodal and the multimodal interpretation. On the contrary, if domain requirements for a *strong* specialization come from the *negative* class, then the autoassociator should not be affected whereas the MLP network, should. Examples of such domains are those for which, although the positive class should rightly be interpreted as a multi-modal class, the negative class can be wrongly interpreted as a uni-modal one. There, knowledge of the negative class is, thus, a greater liability than its ignorance. Finally, in cases

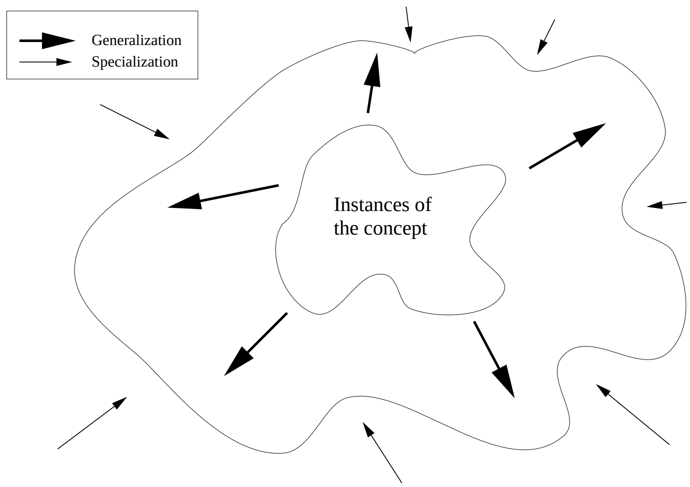


Figure 5. The Generalization/Specialization Framework: Inductive processes seek a balance between their generalization thrust and their specialization strive.

where a *weak* specialization is required, we expect that the autoassociator is more likely to learn this subconcept than the MLP network. Examples of such domains are those for which a multi-modal positive class should rightly be interpreted as multi-modal, but certain of its subcomponents are represented by very few examples and, thus, are very faint. The reason why the unsupervised scheme is expected to perform better than the supervised scheme on such domains is because, once again, the autoassociator focuses on characterizing the positive class independently of the negative class, whereas the MLP network may overlook a particularly small (or under-represented) subconcept, especially if the negative class is not under-represented. Illustrations of such domains will be presented in the next section and the remainder of this paper is devoted to the experimental testing of the hypotheses just described and to suggestions as to how they can be further refined.

5. Experimental Hypothesis Testing

The hypotheses stated in the previous section are now turned into concrete questions by creating matching synthetic test domains. The resulting synthetic domains are then tested in accordance with the hypotheses.

5.1. Test Domains Generated

To test the hypotheses described in Section 4, MLP and the autoassociator were compared on a series of synthetic domains presenting the three types of characteristics deemed significant in the above discussion:

- Domains with *strong* specialization needs coming from the *positive* class.

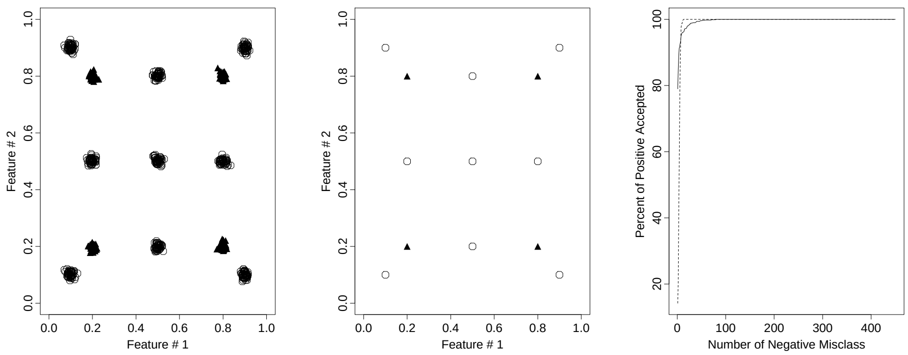
- Domains with *strong* specialization needs coming from the *negative* class.
- Domains with *weak* specialization needs coming from the *positive* class.

These domains were obtained by applying certain transformations to a *neutral* domain, labeled as such since it can be classified almost as accurately by the MLP network and the autoassociator.

5.1.1. The Neutral Domain The neutral domain is illustrated in Figures 6(a) and (b). More specifically, Figure 6(a) represents the complete domain which is composed of four positive and nine negative components corresponding to clusters of data points normally distributed around some given means, and of variance $\sigma^2 = 0.01$. The exact location of each component mean is specified in Figure 6(b) which shows that the means of the positive components are located at points: $(.2,.2)$, $(.2,.8)$, $(.8,.2)$, and $(.8,.8)$; and the means of the negative components are located at points: $(.5,.5)$, $(.1,.1)$, $(.1,.9)$, $(.9,.1)$, $(.9,.9)$, $(.5,.2)$, $(.2,.5)$, $(.5,.8)$, and $(.8,.5)$. This domain was inspired from the 2-D projection of the real-world domains used in section 3 and plotted in Figure 11 of Section 6, as described in (Japkowicz, 1999a).

The relative average classification performance of MLP and the autoassociator is reported in Figure 6(c) which represents the ROC curves obtained by the two systems on this domain. As described in Section 3.2, the ROC curve technique we used plots the percent of positive instances of the concept correctly characterized versus the actual number of false negatives for each classification method considered. Once again, the curves representing different systems are plotted on the same graph and the curve located above the others represents the most accurate classifier. In figure 6(c), the ROC curve for the MLP network is represented by a broken curve while the ROC curve for the autoassociator is represented by a full curve. Each curve represents the average performance obtained after repeating the experiment five times on different realizations of the domain for each system. The fact that the two curves are almost co-located indicates that the two systems are practically equally accurate on this neutral domain.

For the experiment reported in Figure 6(c), the momentum and learning rates of the two systems were set to the standard values of 0.9 and 0.05, respectively. As in the experiments of Section 3, a posteriori experimental methodology was followed which compares the average performance of the two networks over all possible capacities within the set of {1, 2, 4, 8, 16, 32, 64} hidden units and returns the optimal results obtained by both networks. Because we are now dealing with artificial domains, there was no need to run cross-validation experiments since any amount of data could be generated at will. Instead, 5 different pairs of training and testing sets were generated from the same backbone model of Figure 6(b) and the results obtained in each of these domains were averaged. The same capacity of 16 hidden units was found to be optimal for both networks and on all domains.[7] Since overfitting is a minor issue in large, well-behaved noiseless domains, there was no need to look for a specific optimal stopping point in every realization of



(a) Neutral Domain (b) Means Only (c) ROC Analysis

Figure 6. (a) and (b): The Original Artificial Domain and (c) the result of a ROC analysis of MLP and the autoassociator on this domain. MLP is represented by a broken curve while the autoassociator is represented by a full curve. Since the two curves are almost co-located, the two systems are about as accurate on this domain.

the domain: a single stopping point was selected beyond the average time of convergence observed for each system on all domains. This stopping point was set at 2000 epochs for the autoassociator and at 5000 epochs for the MLP network based on observing the average learning curves obtained by the two systems on the neutral domain and reported in Figure 7. These curves plot the average percentage error, over the same 5 domain realizations used in the experiments reported in Figure 6(c), obtained by each system as a function of the number of epochs for which it was trained.[8] Aside from indicating an adequate stopping point for use in the ROC analysis, these plots indicate that while the autoassociator converges almost immediately (as early as by epoch 500), the MLP network takes between 3000 and 3500 epochs to converge. This shows that, in addition to being as accurate or more accurate than MLP in certain domains, the autoassociator can also be significantly more efficient than MLP. This observation has been analyzed in more detail in (Japkowicz, 1999a; Japkowicz & Hanson, 1999). To rectify this discrepancy and improve MLP's efficiency on our artificial domains, its learning rate was increased to 0.2 in all the experiments reported in the remainder of this paper.

5.1.2. The Modified Domains The transformations applied to the neutral domain in view of the hypotheses of Section 4 are illustrated in Figures 8 and 9. Specifically,

- Modifications (a), (b) and (c) represent three domains in need of strong specialization coming from the positive class. These three modifications can be thought of as void reduction methods in that they all—in one way or another—decrease the void occurring between the four positive clusters of the original domain. Decreasing this void calls for a stronger specialization effect since it

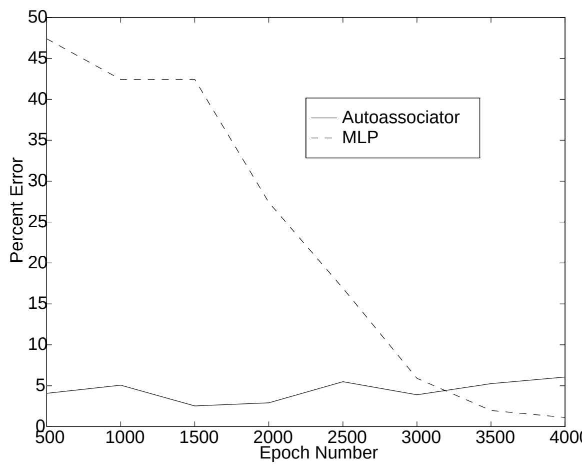


Figure 7. Average classification error rates (as a function of the number of epochs) obtained by MLP and the autoassociator on the neutral domain. Although MLP is slightly more accurate than the autoassociator on this domain, the autoassociator is much more efficient than MLP.

creates a tendency for classifiers to integrate the various positive clusters into a single larger cluster, a step that would cause a high classification error to occur since internal negative examples would be misclassified as positive rather than negative.

- Modifications (d) and (e) represent two domains in need of strong specialization coming from the negative class. More specifically, the purpose of modification (d) is to increase the amount of surface of each positive cluster surrounded by negative data. This modification calls for greater specialization strength because the negative data are not all located in a few localized spots, as in the neutral domain, but rather, they wrap around the positive clusters. The purpose of modification (e) is to decrease the density of each negative cluster, thereby decreasing the amount of external specialization provided to systems which take such information into consideration, and calling for increased internal specialization.

- Modification (f) represents a domain in need of weak specialization coming from the positive class. This modification was applied by decreasing the cardinality of one of the four positive clusters in order to test whether or not, in case of imbalance in the density of the positive clusters, the small cluster is correctly identified or misclassified as a noisy occurrence.

Modification (a) was implemented by creating a concave, but connected pattern. It consisted of turning the central internal negative cluster of figure 6(b) into a positive one. Modification (b) was implemented by moving the positive clusters to a distance of 0.2 units from each other (we recall that the original distance between

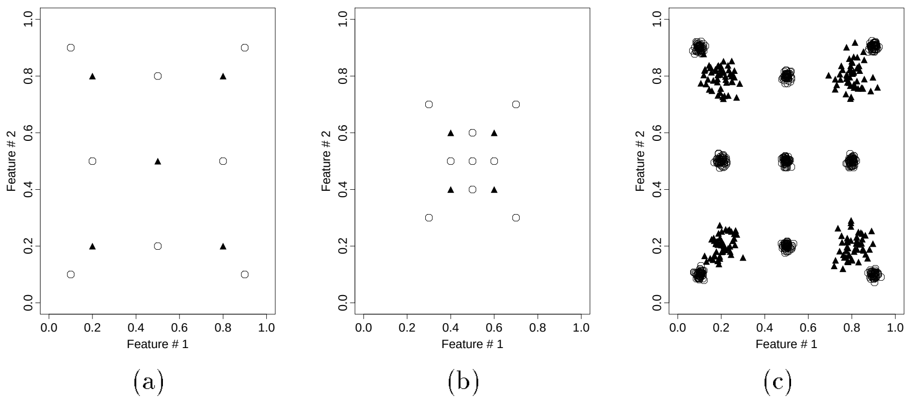


Figure 8. Modifications of the neutral domain that create strong specialization needs caused by the positive class. These domains are expected to be more accurately classified by MLP than by the autoassociator.

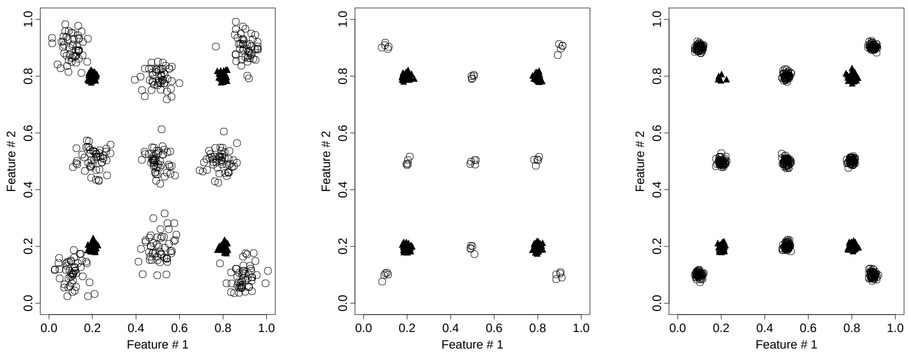
each positive cluster was of 0.6 units). Modification (c) was implemented by increasing the variances of each positive cluster to 0.04 (we recall that the variance was 0.01 in the original domain). Modification (d) was implemented by increasing the variance of each negative cluster to 0.04. Modification (e) was implemented by decreasing the size of the negative clusters to 5 examples each (we recall that each cluster contained 50 data points in the original domain). Finally, modification (f) was implemented by decreasing the cardinality of the left top positive cluster of Figures 6(a) and (b) from 50 to 5 in order to test whether or not, in case of imbalance in the density of the positive clusters, the small cluster gets correctly identified.

According to the hypotheses of Section 4, the domains of Figure 8 should get more accurately classified by the MLP network and the domains of Figure 9 should get more accurately classified by the autoassociator. The next section describes the experiments conducted on these domains in order to test these hypotheses, and then discusses their results.

5.2. Experiments and Results

In this section, the experiments are first described followed by a discussion of the results obtained.

5.2.1. Experiments The results of the comparisons of MLP with the autoassociator on each of the domains of Figures 8 and 9 are displayed in Figure 10. As in Figure 6(c), these results are reported in the form of ROC curves. Again, for each domain, the curves for the MLP network and autoassociator are plotted on the same graph and the curve located above the other represents the most accurate classifier on that domain. The autoassociator's performance is represented by a full



(d) (e) (f)

Figure 9. Modifications of the neutral domain that create strong specialization needs caused by the negative class (figures (d) and (e)) and weak specialization needs caused by the positive class (figure (f)). These domains are expected to be more accurately classified by the autoassociator than by MLP.

curve while MLP's is represented by a broken curve. As in previous experiments, all the experiments were repeated five times on domains emanating from the same backbone models of Figures 8 and 9 and the curves displayed represent the average results obtained on these five domains.

Optimal capacities were determined as in the experiments reported in Figure 6(c) (they are reported in Table 3) and the optimal stopping criteria were once again set to 2000 for the autoassociator and 5000 for the MLP network. The momentum of the networks was set to the standard value of 0.9 for each system, but, while the learning rate of the autoassociator was set to the standard value of 0.05, as mentioned previously, it was increased to 0.2 in the MLP network, in order to speed up learning.

5.2.2. Results The results of the experiments performed on the six domains described in Section 5.1 are reported in the graphs of Figure 10.

These graphs show that, according to the expectations expressed in Section 4, MLP outperforms the autoassociator in the domains in need of strong specialization caused by the positive class (Figures 10(a), (b) and (c)); the autoassociator outperforms MLP in the domains in need of strong specialization caused by the negative class (Figures 10(d)[9] and (e)); and the autoassociator outperforms MLP in the domain in need of weak specialization caused by the positive class (Figure (f)).[10]

The results obtained in this section thus demonstrate that, at least on synthetic domains, autoassociation-based classification is more accurate than MLP in multimodal domains presenting certain well-defined characteristics. The autoassociator's marked efficiency on the neutral domain relative to MLP also suggests that

Table 3. Optimal Capacities for the autoassociator and MLP network on the different domains considered in this study. The first column lists the optimal capacities for the autoassociator and the second column lists the optimal capacities for MLP.

Domain	*Autoassociator*	*MLP*
Neutral	16	16
(a)	4	64
(b)	16	16
(c)	8	64
(d)	64	16
(e)	64	16
(f)	16	4

autoassociation-based classification is better disposed towards multi-modal domains than MLP.

The question of whether these results are dependent upon overall problem sizes was also considered, though in the context of a different study focused on the *between-class* class imbalanced problem of the type illustrated by the domain of Figure 6(e).[11] This study suggested that no relationship between an increase in problem size (with a constant class imbalance ratio) and an increase in classification error could be established for either the autoassociator or the MLP network in such problems (Japkowicz, 2000b).

6. Suggested Framework for Fine-Tuning the Current Results

The observations made in the previous section will now be used to explain the results obtained on real-world domains in Section 3.3 and suggest new research directions to fine-tune our current results in case of a mismatch. This study is divided into three parts. In the first part, the domains of Section 3 are plotted and a number of statistics gathered. In the second part, the salient characteristics of these statistics are listed and they are analyzed and discussed in the last part.

6.1. Statistics Gathering

The real-world domains of Section 3 are first projected onto their first two principal components, thus reducing them to 2-Dimensional representations like the synthetic domains of Section 5. A simple graphical analysis method was used to cluster the

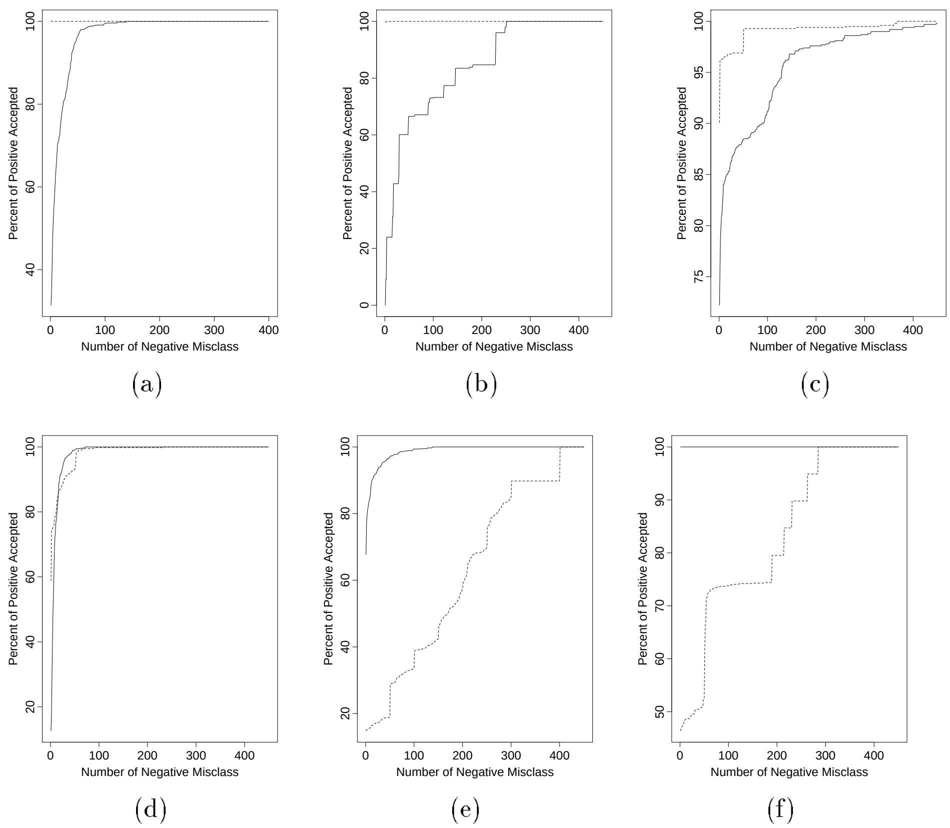


Figure 10. ROC Analysis for all the domains. The autoassociator's performance is represented by a full curve while MLP's is represented by a broken one. As expected from the hypotheses of Section 4, MLP is shown to be more accurate than the autoassociator in domains (a), (b) and (c), while the autoassociator is shown to be more accurate in domains (d), (e) and (f).

positive and negative data of these 2-D domains. This graphical analysis method—which falls beyond the scope of this paper—is described in (Japkowicz, 1999a). The resulting representations are illustrated in Figure 11. In this figure, black triangles correspond to non-faulty gearboxes, promoters and mines, respectively and white circles correspond to faulty gearboxes, non-promoters and rocks, respectively. The polygons drawn on the graphs represent the limits of the various clusters formed from these data. These 2-D projections account for 39.4% of all the helicopter gearbox monitoring domain information; for 23.4% of all the sonar target detection domain information and for only 4.22% of all the DNA promoter recognition domain information. Because of the extremely small amount of information carried by the 2-D projection in the DNA domain, the projections for this domain were deemed insignificant and no further analysis was conducted.

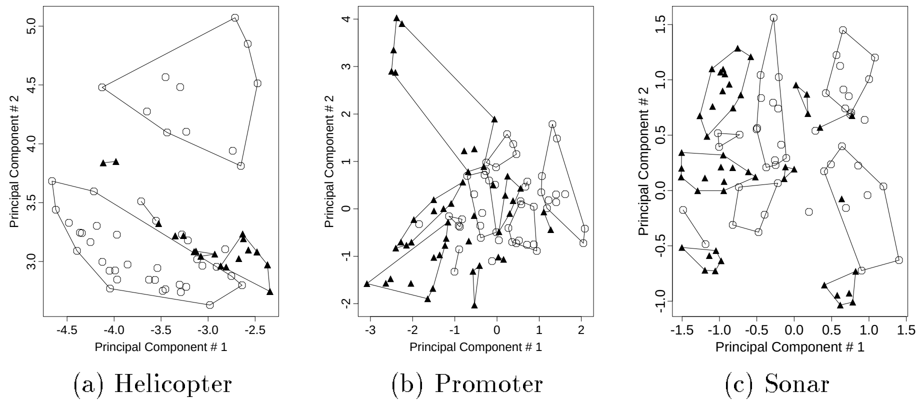


Figure 11. 2-D Plot and Analysis of the three real-world domains of Section 3. Black triangles represent non-faulty helicopter gearbox and mine data, respectively (Class1) and white circles represent faulty helicopter gearbox and rock data, respectively (Class2). The convex hulls surrounding various sets of data points represent the boundaries of the Class1 and Class2 clusters computed by the graphical clustering method described in (Japkowicz, 1999a).

Once the domain projections and analyses were completed, a certain number of domain characteristics, which correspond to the domain characteristics tested on the synthetic domains of the previous section, were then computed from the plots for the helicopter gearbox monitoring and the sonar detection domains. These characteristics are:

- $Class1AvgVar$, the average variance of the various clusters constituting Class1.
- $Class2AvgVar$, the average variance of the various clusters constituting Class2.
- $AvgMeanDist$, the average mean distance between the means of the clusters constituting Class 1 and the clusters constituting Class 2.
- $AvgDist$, the average actual distance between the clusters of Class1 and Class2, respectively.[12]
- $Class1AvgSampleSize$, the average sample size of the clusters constituting Class1.
- $Class2AvgSampleSize$, the average sample size of the clusters constituting Class2.
- $Class1AvgVarSize$, the variance in the sample size of the clusters constituting Class1.
- $Class2AvgVarSize$, the variance in the sample size of the clusters constituting Class2.

For both the helicopter and sonar domains depicted in Figure 11, the quantities just listed were evaluated assuming that Class1 corresponds to the black triangles and Class2 corresponds to the white circles. The values obtained by doing so are displayed in Table 4.

Table 4. Values for the domain characteristics of the three real-world domains of Section 3 where Class1 represents non-faulty gearboxes and mines, respectively and Class2 represents faulty gearboxes and rocks, respectively.

Characteristics	Helicopter	Sonar
$Class1AvgVar$	0.0151	0.027
$Class2AvgVar$	0.21	0.0718
$AvgMeanDist$	1.149	1.217
$AvgDist$	0.568	0.785
$Class1AvgSampleSize$	6.5	7.17
$Class2AvgSampleSize$	20.5	7.2
$Class1AvgVarSize$	12.5	18.97
$Class2AvgVarSize$	180.5	7.7

6.2. Salient Characteristics

The next part of this study consisted of listing, for each of the two domains considered, the most salient observations listed in Table 4.

For the Helicopter Gearbox Monitoring domain, we observed that:

1. The average variance of the Class1 clusters is about 14 times smaller than that of the Class2 clusters (see the values of $Class1AvgVar$ versus $Class2AvgVar$)
2. The average sample size of the Class1 clusters is about 3 times smaller than that of the Class2 clusters (see the values of $Class1AvgSampleSize$ versus $Class2AvgSampleSize$)
3. The variance of the Class1 sample size is between 14 and 15 times smaller than that of the Class2 sample size (see the values of $Class1AvgVarSize$ versus $Class2AvgVarSize$)
4. The domain is not strongly multi-modal: Class1 seems to belong roughly to a single cluster "sandwiched" between the only two Class2 clusters.

For the Sonar detection domain, we observe that:

1. The average variance of the Class1 clusters is between 2 and 3 times smaller than that of the Class2 clusters (see the values of $Class1AvgVar$ versus $Class2AvgVar$)

2. The variance of the Class1 sample size is between 2 and 3 times larger than that of the Class2 sample size (see the values of $Class1AvgVarSize$ versus $Class2AvgVarSize$)

3. The domain is very clearly multi-modal: several pairs of Class1 clusters are separated by Class2 clusters and a couple of pairs of Class1 clusters are separated by Class1 clusters.

6.3. Analysis and Discussion

We now analyze the observations regarding the real-world domains in light of Section 5's results.

6.3.1. Helicopter Domain In the case of the helicopter domain, the first observation can be related to the results obtained for modification (d) in Section 5 for the situation where Class1 represents the positive class and Class2, the negative one and for modification (c) in Section 5 for the situation where Class1 represents the negative class and Class2 represents the positive one. This suggests that, according to the results of Section 5, the autoassociator should be more accurate than MLP in the first situation, but that MLP should be more accurate than the autoassociator in the second. However, the results of Section 3 suggest that this is not the case in the first situation. Indeed, in the situation where Class1 (i.e., the non-faulty gearboxes) represents the positive class, MLP was shown to be slightly more accurate than the autoassociator (except if only a single negative misclassification can be tolerated). In the reverse case, both the analytical results of Section 5 and the practical results of Section 3 agree.

Similarly, the second observation for the helicopter gearbox domain can be related to the results obtained for modification (e) in Section 5. This suggests that when Class1 represents the positive class, the autoassociator should be more accurate than MLP. Once again, however, the results of Section 3 contradict the expectations raised in Section 5 in the case where Class1 represents the positive class (except, again, if only a single negative misclassification can be tolerated). Since no results were established in Section 5 for the case where the positive class is under-represented, no conclusion can be drawn for the situation where Class2 represents the positive class.

The third observation for the helicopter gearbox can again be related to some results from Section 5. This time, it can be related to modification (f) which suggests that the autoassociator should be more accurate than MLP when Class2 represents the positive class. When Class1 represents the positive class we also expect the autoassociator to outperform MLP since Figure 12(f) suggests that the autoassociator's superiority in this case is caused by a deterioration of MLP's accuracy rather than an improvement of the autoassociator's. In both cases, however, the results of Section 3 contradict the expectations raised in Section 5.

The last observation does not correspond to any of the experiments performed in Section 5. Rather, it corresponds to the implicit assumption, in that section, that the domain of interest is strongly multi-modal. The observation that this assumption is violated in the case of the helicopter domain suggests that the great number of mismatches observed between the results of Section 3.3 and those of Section 5 on this domain may be caused by this disagreement. Indeed, Figure 9 indicated MLP's difficulty in dealing with highly multimodal domains by its need for many more epochs of training than the autoassociator (at least, six times more) or a larger learning rate. Although MLP's performance on non highly multi-modal domains has not been documented, we suspect that it is very good and that the autoassociator shows classification superiority only in cases of high multi-modality. If this is truly the case, then the results obtained on the helicopter gearbox domain could be justified, as we just suggested, by the domain's low multi-modality.

6.3.2. Sonar Domain For the sonar domain, the first observation can again be related to the results obtained for modification (d) in Section 5 for the situation where Class1 represents the positive class and Class2 represents the negative one, and for modification (c) in Section 5 for the situation where Class1 represents the negative class and Class2 represents the positive one. This suggests that, according to the results of Section 5, the autoassociator should be more accurate than MLP in the first case, but that MLP should be more accurate than the autoassociator in the second. According to the results of Section 3 for the sonar domain, this is precisely the case.

The second observation for the sonar domain can be related to modification (f) which suggests that, once again, the autoassociator should be more accurate than MLP when Class1 represents the positive class. As previously, when Class2 represents the positive class we also expect the autoassociator to outperform MLP since Figure 10(f) suggests that the autoassociator's superiority in this case is caused by a deterioration of MLP's accuracy rather than an improvement of the autoassociator. In the situation where Class1 represents the positive class, the results of Section 3 agree with those of Section 5, but in the situation where Class2 represents the positive class, they do not.

The last observation for the sonar domain does not directly correspond to any of the experiments of Section 5, but it does suggest that, in the case of the Sonar domain, the assumption made in Section 5 regarding the multi-modality of the domain of interest does apply. This may, therefore, explain why the results of Section 5 match those obtained on the sonar domain better than those obtained on the helicopter gearbox domain.

6.3.3. Discussion Attempting to match the results of Section 5 to those of Section 3 is a useful exercise as it allows us to view the worthy aspects of our study on artificial domains while revealing its shortcomings. These worthy aspects are reflected by the fact that, in the sonar domain which is highly multi-modal (like the artificial domains of Section 5), the results of Section 5 apply almost perfectly: only one out of four expected results was not observed. On the other hand, there are many shortcomings to the study of Section 5 as revealed by the fact that one out of four expected results was not observed in the case of the sonar domain and, more notably, four out of five expected results did not occur in the case of the helicopter gearbox domain.

We suggest that the mismatch we encountered when attempting to apply the results obtained on artificial domains to real-world domains had to do with the following factors, mostly related to the fact that our results are, to this point, qualitative rather than quantitative:

Lack of Multi-Modality Quantification: As suggested in our analysis, one of the reasons why a mismatch between our theoretical expectations based on artificial domains and the practical results may be related to the amount of multi-modality associated with the domain under study. A study based on artificial domains varying the amount of multi-modality present would be useful for fine-tuning our current results and reliably explaining the mismatches in the helicopter domain.

Lack of Internal Quantification: Some of the mismatch between the results obtained in the artificial domains and the real-world domains may also be caused by the fact that internal quantifications were not taken into consideration. For example, while in figures 8(c) the difference in variance between the positive and negative class is a factor of 4, it is a factor of 14 in the helicopter gearbox domain. Such differences may be a factor in the discrepancies between our results on artificial and real-world domains.

Lack of External Quantification: While the artificial domain experiments of section 5 studied the effect of different and separate domain characteristics upon the accuracy of the autoassociator relative to that of MLP, it did not combine the different characteristics it considered. Consequently, the study has not been able to describe the relative importance of the different characteristics it surveyed. This is of utmost importance since real-world domains have combined characteristics, which, if yielding theoretically opposite conclusions, must be weighted against one another.

Lack of Error Rate Quantification: In our study, we only considered qualitative rather than quantitative error rates. i.e., we only concluded that one system outperformed the other without mentioning by what amount. For example, it is important to note that MLP outperforms the autoassociator by only a few percent error in the helicopter gearbox domain whereas the autoassociator outperforms MLP by around 20% and even more in the sonar domain (See Figure 3). These ratio should also be taken into consideration in our analysis.

Limited Domain Knowledge: As mentioned earlier, our real-world domain projections in 2-D plots did not conserve all the domain information contained in the original domains. It would be useful to employ a different visualization technique that would allow us not to give up all this information. For example, we could use a visualization technique such as that of (Inselberg & Dimsdale, 1990).

Qualitative Restrictions: Our study was restricted to certain types of characteristics. For example, we did not consider the opposite case to that of Figure 9(e) in which the positive class would be under-represented while the negative one would be well-represented. As well, we did not consider cases where clusters of opposite classes overlap or where clusters are not normally distributed.

Altogether, the observations of this section suggest that the results obtained by the autoassociator and MLP on the helicopter gearbox and sonar target detection domains were not coincidental. Rather, especially in the case of the sonar domain, they seem rooted in a number of intrinsic differences between the two systems that favor the classes of domains to which the domains belong. Nevertheless, although the experiments of this paper have managed to isolate some of these characteristics, future work should be carried out in order to refine our study and fully explain the two system's performance on real-world data.

7. Future Work

We have shown that unsupervised learning by feedforward neural networks can be as accurate, if not more accurate, than supervised learning by feedforward networks on real world domains, but, to this point, we ignored the issue of a-priori parameter and threshold determination which are important aspects of recognition-based systems for practical settings. Such a question can be very delicate, especially when negative examples are not available. The purpose of this section is to illustrate the fact that although in some cases parameter and threshold determination in the absence of negative examples can be more complicated than in others, it is always possible provided that enough positive examples are available.

To illustrate this observation, the results of two case studies were plotted in Figures 12 and 13. The plots of Figure 12 correspond to the results obtained by the autoassociator on Fold 4 of the helicopter gearbox monitoring domain where the positive class is taken to be the non-faulty class. This case represents a situation where many different combinations of parameters will succeed and where no big risk is taken by not using negative examples in the parameter setting process. The plots of Figure 13, on the other hand, illustrate a case where parameter setting is not straightforward and in which lack of knowledge of the negative class can yield a suboptimal result. These plots correspond to the results obtained by the autoassociator on Fold 2 of the promoter domain in the case where the positive class corresponds to the promoter class.

In more detail, each plot in Figures 12 and 13 displays the reconstruction error curve obtained by the instances of the testing set as a function of the epoch number and for a given network capacity, after the network has been trained on the corresponding training set. The capacities considered in these figures are 1, 16

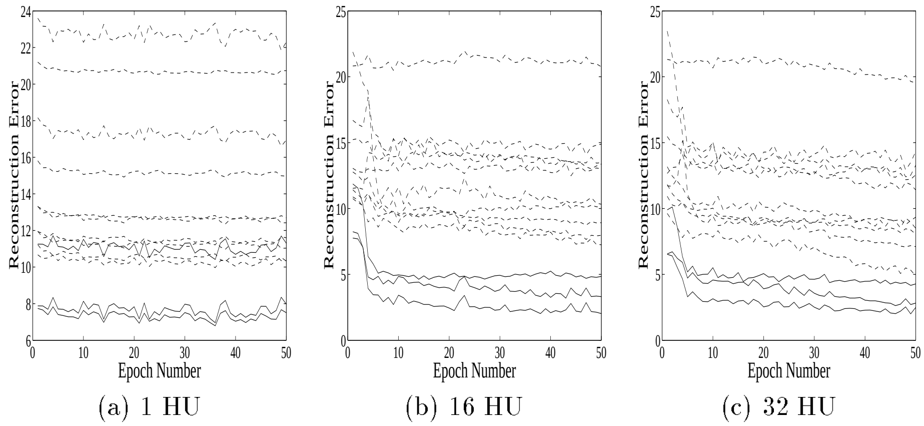


Figure 12. Reconstruction Errors as a function of the Epoch Number of the examples in the testing set of Fold 4 of the Helicopter Gearbox Domain. Positive examples are represented by full lines while negative examples are represented by broken lines. For presentation purposes, the epoch numbers displayed were divided by 10.

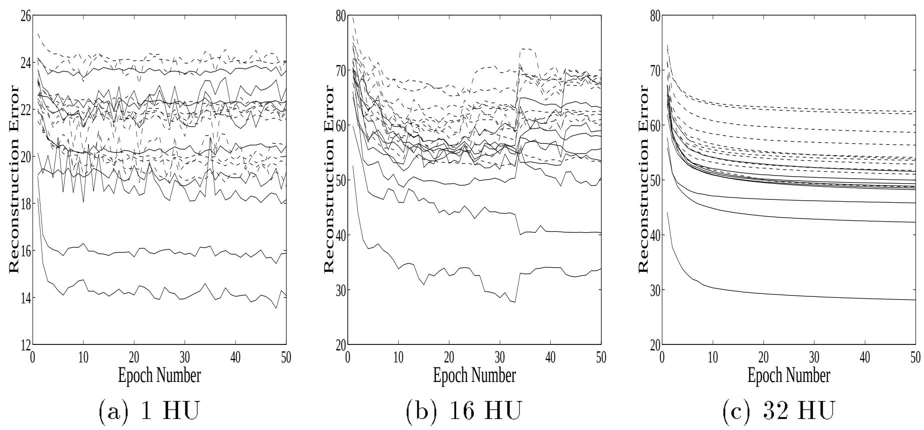


Figure 13. Reconstruction Errors as a function of the Epoch Number of the examples in the testing set of Fold 2 of the Promoter Domain. Positive examples are represented by full lines while negative examples are represented by broken lines. For presentation purposes, the epoch numbers displayed were divided by 10.

and 32 hidden units and each capacity corresponds to a single plot. Within each plot, positive examples are represented by a full-curve whereas negative examples are represented by a broken-curve. Ideally, all the full-curves should be located at the bottom of the graphs and separated by a gap from all the broken-curves which should be located at the top of the graphs.

The plots of Figure 12 suggest that, in certain cases, parameter setting can take place relatively safely in the absence of counter-examples. Indeed, these graphs suggest a policy by which a network capacity and stopping point are selected soon after the occurrence of a drop in the reconstruction error of the set of positive examples, as in Figure 12(b), but not too late after that. Once optimal parameters are established following this policy, selecting a threshold boundary slightly larger than the reconstruction error of the positive example with largest reconstruction error is a safe bet.

The plots of Figure 13, on the other hand, suggest that, in certain cases, parameter setting is a fairly difficult task and cannot be safely undertaken in the absence of counter-examples or, at least, of a very sophisticated clustering component. Indeed, in Figure 13(c), it can be argued that a good clustering strategy could detect that the positive example with highest reconstruction error is atypical and thus that the threshold boundary could be placed closer to the positive example with second highest reconstruction error.

If many positive examples are available, then a good clustering process able to discard atypical examples while recognizing where the bulk of the reconstruction errors lies, is likely to be successful. If, however, like in the case studies of this work, little positive data is available, then negative examples, sometimes, play a very important role in setting a threshold. In these cases, however, very few negative examples are typically sufficient as demonstrated by (Japkowicz et al., 1995).

8. Conclusion

This paper originated from the observation that autoassociation-based classification was reported to be more accurate than the MLP discrimination network on two out of three real-world domains and as accurate as MLP on the third, despite the fact that the autoassociator learns a concept in the absence of negative examples (Japkowicz et al., 1995). The purpose of this paper was to find out whether this result was purely coincidental or whether it could be expected in other domains. The results obtained suggest that the autoassociator is more accurate than MLP in domains that require particularly strong specialization coming from the negative class or particularly weak specialization coming from the positive class, in order to be accurately classified. The main analysis of the paper is conducted on synthetic domains, but a preliminary attempt at confirming its results is presented on two real-world domains.

The results of this study thus suggest that unsupervised learning techniques, which are often dismissed in favor of supervised ones in the context of binary classification, may present an interesting array of classification strengths. This observation comes in addition to the fact that unsupervised techniques learn concepts in the absence of negative data and may thus be useful—assuming that parameter and threshold setting can also be performed accurately in the absence of negative data—for a whole range of practical domains that cannot even be considered by supervised learning methods.

Acknowledgments

I would like to express my gratitude to Stephen J. Hanson and Casimir A. Kulikowski, my Ph.D. thesis advisors, for their helpful guidance and comments during the development of the ideas presented in this work. Mark Gluck and Catherine Myers' suggestions during the first phase of this project were also very helpful. In addition, I would like to thank Doug Fisher as well as two anonymous reviewers who provided me with extremely thorough comments on two previous drafts of this paper, John Josephson for his useful suggestions and Mike McAllister for his help with some coding issues that arose during my experiments.

Notes

1. Related to these studies, a *three-hidden layer* nonlinear autoassociator was also successfully applied to a *multi-class* character-recognition task, using one autoassociator per class (Schwenk & Milgram, 1995).
2. The reconstruction error corresponds to $\Sigma_{i=1}^{d}[x_i^{Test} - f_i(x^{Test})]^2$, where x_i^{Test} and $f_i(x^{Test})$ represent input and output unit i of the network, respectively, and d is the length of the input and output vectors.
3. Note that since, within every domain considered, 1) the number of negative testing examples available in each fold, N_{f_i} is different and 2) the length of the ROC graph's x-axis is equal to N_{f_i}, we reported the average results obtained over all folds by considering N_{Common}, the largest common number of negative examples available in all folds within each domain. i.e., $N_{Common} = min_{i=1..5} N_{f_i}$.
4. In the plots of Figure 3, the x-axis represents the number of negative examples (faulty gearboxes, non-promoters, and rocks) misclassified while the y-axis represents the number of positive examples (non-faulty gearboxes, promoters and mines) recognized. Conversely, in the plots of Figure 4, the x-axis represents the number of positive examples(non-faulty gearboxes, promoters and mines) misclassified while the y-axis represents the number of negative examples (faulty gearboxes, non-promoters, and rocks) recognized. As mentioned before, in order to be able to compute the average ROC results obtained over all folds within a domain, the largest common number of negative testing examples available in all the folds, N_{Common}, had to be selected. For example, since in one fold of the helicopter gearbox problem, the number of non-faulty testing patterns is 2, the x-axis of Figure 4(a) only spans interval [1, 2] and y values are only reported for x-values of 1 and 2, even though the number of non-faulty testing patterns in all the other folds is greater than 2.
5. Although the results of this paper are slightly different from those obtained in (Japkowicz et al., 1995)—in (Japkowicz et al., 1995), the autoassociator, trained on the configuration used in Figure 3, outperformed the MLP network on both the helicopter gearbox and sonar domain and the two systems were equally accurate on the promoter domain—the same qualitative conclusions hold. We believe that the difference between the two studies is due to the biases introduced in (Japkowicz et al., 1995) by means of an a-priori parameter and threshold setting strategies. These biases were eliminated in the experiments of this paper through the use of our posteriori methodology.
6. Note that different unsupervised learning systems use different internal biases. Japkowicz, Hanson & Gluck (2000) and Japkowicz (1999a) extract the particular internal bias used by the autoassociator.
7. In this set of experiments, because of the small variability from domain to domain (there was much more variability from one fold to the other in the real-world experiments), the a-posteriori optimal capacity was determined as a function of the best average results over the five runs, rather than on a trial-per-trial basis.

8. In figure 7, threshold boundaries for the MLP network were established by setting a boundary at $M = mean(NegativeTuningSet) + (mean(PositiveTuningSet) - mean(NegativeTuningSet))/2$ (where Positive and Negative Tuning Set are fresh sets of positive and negative data not used to train the network nor to test it). For the autoassociator, they were established by fitting the reconstruction errors obtained by the elements of PositiveTuningSet to a Gaussian distribution and determining its 97% confidence interval (97% was chosen arbitrarily).
9. Note that in the case of figure 10(d), the difference in accuracy of the two systems is very slight. However, since MLP was slightly more accurate than the autoassociator in the neutral domain, the difference in accuracy observed in domain (d) is relatively meaningful.
10. Given that our artificial domains are noiseless and that their training sets are generated using the same backbone model as their testing sets, MLP should eventually be able to converge on all these domains given long enough training times. The fact that it does not do so in domains (e) and (f) despite the already large training time imparted to it and its increased learning rate shows that it has a very difficult time doing so, especially in comparison to the autoassociator which does not require increased training times or learning rates.
11. Domains 5(e) and 5(f) can be cast as two different expressions of the class imbalance problem. Domain 5(e) illustrates the case of a *between-class* imbalance, where all the subclusters belonging to the same class are represented by the same number of examples but there is a difference in the size of the subclusters belonging to two different classes, while fomain 5(f) illustrates the case of a *within-class* imbalance where not all the subclusters within the same class are necessarily represented by the same number of examples.
12. The previous characteristic, $AvgMeanDist$, focuses on the distance between the means of the various clusters. This characteristic subtracts the average standard deviations of the Class1 and Class2 components from the results of the previous question.

References

Aha, D.W., Kibler, D., & Albert, M.K., (1991). Instance-Based Learning Algorithms. *Machine Learning*, 6(1).

Breiman, L., Friedman, J., Olshen, R., & Stone, C. (1984). *Classification and Regression Trees*, Wadsworth & Brooks Publishers.

Cottrell, G.W., Munro, P., & Zipser, D. (1987). Learning Internal Representations from Gray-Scale Images: An Example of Extensional Programming, *Proceedings of the Ninth Annual Conference of the Cognitive Science Society*, Seattle, WA, 462–473.

Elman, J. L. and Zipser, D. (1988). Learning the Hidden Structure of Speech, *Journal of the Acoustical Society of America*, 83.

Fukunaga, K. (1990). *Introduction to Statistical Pattern Recognition (2nd Edition)*, Academic Press, San Diego, CA.

Hanson, S.J. and Kegl, J. (1987). PARSNIP: A Connectionist Network that Learns Natural Language Grammar from Exposure to Natural Language Sentences, *Proceedings of the Ninth Annual Conference on Cognitive Science*, Seattle, WA.

Inselberg, A. and Dimsdale, B. (1990). Parallel Coordinates: A Tool for Visualizing Multi-Dimensional Geometry, *Proceedings of the IEEE Visualization Conference*, 361–370.

Japkowicz, N., Myers, C., & Gluck, M.A. (1995). A Novelty Detection Approach to Classification, *Proceedings of the Fourteenth Joint Conference on Artificial Intelligence*, Montreal, CA, 518–523.

Japkowicz, N. (1999a). *Concept-Learning in the Absence of Counter-Examples: An Autoassociation-Based Approach to Classification*, Ph.D. Dissertation, Technical Report DCS-TR-390, Rutgers University.

Japkowicz, N. (1999b). Are we better off without counter-examples?, *Proceedings of the First International ICSC Congress on Computational Intelligence Methods and Applications*, Rochester, NY, 242–248.

Japkowicz, N. & Hanson, S.J. (1999). Adaptability of the Backpropagation Procedure, *Proceedings of the International Joint Conference on Neural Networks*, Washington, DC, Paper 208.

Japkowicz, N., Hanson, S.J., & Gluck, M.A. (2000a). Nonlinear Autoassociation is not Equivalent to PCA, *Neural Computation*, 12(3), 531–545.

Japkowicz, N. (2000b). The Class Imbalance Problem: Significance and Strategies (to appear), *Proceedings of the International Conference on Artificial Intelligence (Inductive Learning Track)*, Las Vegas, NV.

Kolesar, B. and NRaD (1994). Helicopter Gearbox Fault Monitoring. *Presentation at the 1994 Neural Information Processing System Workshop on Monitoring and Novelty Detection.*

Kubat, M., Holte, R. & Matwin, S. (1998). Machine Learning for the Detection of Oil Spills in Satellite Radar Images, *Machine Learning*, 30, 195–215.

Norton, S.W. (1994). Learning to Recognize Promoter Sequences in E. coli by Modeling Uncertainty in the Training Data, *Proceedings of the Twelfth National Conference on Artificial Intelligence*, Seattle, WA.

Petsche, T., Marcantonio, A., Darken, C., Hanson, S.J., Kuhn, G.M., & Santoso, I. (1996). A Neural Network Autoassociator for Induction Motor Failure Prediction, *Proceedings of the Eigth Conference on Advances in Neural Information Processing Systems*, Denver, CO.

Quinlan, J.R. (1993). *C4.5: Programs for Machine Learning*, Morgan Kaufmann Publishers, San Mateo, CA.

Rumelhart, D.E., Hinton, G.E. & Williams, R.J. (1986). Learning Internal Representations by Error Propagation *Parallel Distributed Processing*, David E. Rumelhart & J. L. Mc Clelland (Eds), MIT Press, Cambridge, MA, 318–364.

Schwenk, H. & Milgram, M. (1995). Transformation Invariant Autoassociation with Application to Handwritten Character Recognition, *Proceedings of the Seventh Conference on Advances in Neural Information Processing Systems*, Denver, CO, 991–998.

Weiss, S.M. & Kulikowski, C.A. (1991). *Computer Systems That Learn*, Morgan Kaufmann Publishers, San Mateo, CA.