ELSEVIER

Journal of Economic Behavior & Organization
Vol. 49 (2002) 345–361

JOURNAL OF
Economic Behavior
& Organization

www.elsevier.com/locate/econbase

A computational theory of the firm

Jason Barr, Francesco Saraceno

Department of Economics, Columbia University, 1022 International Affairs Building, 420 West 118th Street, New York, NY 10027, USA

Received 15 September 2000; received in revised form 6 July 2001; accepted 19 July 2001

Abstract

This paper proposes using computational learning theory (CLT) as a framework for analyzing the information processing behavior of firms; we argue that firms can be viewed as learning algorithms. The costs and benefits of processing information are linked to the structure of the firm and its relationship with the environment. We model the firm as a type of artificial neural network (ANN). By a simulation experiment, we show which types of networks maximize the net return to computation given different environments. © 2002 Elsevier Science B.V. All rights reserved.

JEL classification: C63; D21; D83; L20

Keywords: Firm learning; Information processing; Neural networks

1. Introduction

This paper is an exploration into the modeling of firm structure and learning given different environmental conditions. Our research objective is to study the question *what is the relationship between the structure of the firm and the level of complexity and instability of the environment*?

We argue that the nature of firm information processing and learning can be modeled with the tools of computational learning theory (CLT) as developed by computer scientists to study learning and problem solving by machines. CLT clarifies the *economic* features of information processing and learning, i.e. a process that requires resources that are costly to acquire and implement.

We provide a simple, general simulation model of the firm as a collection of information processing units–a type of artificial neural network (ANN). The optimal number of processing units is a function of the degree of complexity and instability of the environment. We

E-mail address: jb357@columbia.edu (J. Barr).

PII: S0167-2681(02)00002-1

specify the costs and benefits to computation, and show how the optimal firm size changes as the environment changes. Larger organizations have more computational power but with increasing costs. In particular, if the environment is very complex, larger firms have more difficulty adapting to frequent changes, and thus perform worse than smaller, more flexible ones. On the other hand, when the environment is stable, larger and more computationally powerful firms have an advantage vis a vis smaller ones.

The outline of the paper is as follows. Section 2 sketches the main concepts and findings of computational learning theory and provides a specific example of a learning machine, the ANN. In Section 3, we model the firm as a network of processing units which must learn economic environments that differ along the dimensions of stability and complexity. We establish, by means of this model, a relationship between environmental features and organizational structure. In Section 4, we discuss some real-world examples of firm adaptation given different environments. Finally, Section 5relates our paper to the existing literature and concludes.

2. Computational learning theory and the firm

In this section, we discuss the computational learning problem and its relevance to the firm.[1] Take a space X (such as the Euclidean space, $\mathbb{R}^N$), sometimes called *the instance space*, and a subset, $C \subseteq X$, called the *concept class*. The set of elements to be learned is referred to as the *target concept*, $c \in C$. In Section 3; for example, the target concept is a set of strings of 10 binary digits. Another example of a target concept is a particular demand function that a monopolist must learn.

The *learning environment* is composed of a set of examples $\{x_1, \ldots, x_t\} \in X$, drawn according to some fixed (and possibly unknown) probability distribution, and an associated set of indicator functions indicating whether each element of the example set is a member of the concept set. That is, for each x_i, $I_C(x_i) = 1$, if $x_i \in c$, and $I_C(x_i) = 0$, if $x_i \notin c$. The indicator function is said to be given by an "oracle" or "teacher." In other words, the oracle associates with the example x_i an indicator function $I_C(x_i)$. After drawing t samples, we have a *labeled multisample*, $[(x_1, I_C(x_1)) \cdots (x_t, I_C(x_t))] \in [X \times \{0, 1\}]^t$.

A *learner* uses the labeled multisample in order to attempt to learn the concept by choosing from a set of hypotheses (or functions) H the "best" one, in the sense that it is closest to the target. "The learner is an algorithm (or a partial recursive function) from data sets to hypothesis classes"(Niyogi, 1998, p. 5). Such an algorithm is a map $A : [X \times \{0, 1\}]^t \to C$. In our case, the learner is the firm. In particular, we view the firm as a neural network and the hypothesis as the firm's state of knowledge given the data its has processed (Section 2.1). We view the size of the hypothesis space as analogous to the size of the firm, or more specifically, as the amount of resources the firm devotes to learning the economic environment, i.e. the target concept.

The learner/firm tries to learn the concept by modifying its hypothesis after each example it processes. This updating process is the characteristic feature of learning. The specific

[1] We draw heavily on Kearns and Vazarani (1994), Vapnik (1995), Niyogi (1998), and Vidyasagar (1997) to which the reader is referred for an extensive treatment of these concepts.

hypothesis chosen after t examples is denoted by $h_t = A[(x_1, I_C(x_1)) \cdots (x_t, I_C(x_t))]$. The difference between h_t and the underlying concept c, is the *generalization error*, defined as the distance $d(h_t, c)$, where d is a specific metric.

Once it has selected the best hypothesis, the learner may use it to *generalize*, i.e. to process data that do not belong to the example space. It will be made clear later that the issue of generalization is at the heart of learning theory, and is one of the features that make it different, for example, from simple regression analysis.

For a concept to be learnable we need two distinct conditions. The first is that the representational capacity of the hypothesis class, H, has to have sufficient power to approximate the concept class. If this is not the case, it may be that *irrespective of the number of examples* the best hypothesis belonging to H will still be far away from the concept. This is called the *approximation error*, and it is defined as the distance between the best hypothesis in H and the concept, and is denoted as $d(h_\infty, c)$. This is the minimum error attainable by the learner, and if it is large, it is due to the insufficiency of the hypothesis class. A concept may be not learnable because the hypothesis space is not complex enough, i.e. has too few parameters to estimate the concept.

The second source of problems comes from the finiteness of the sample available to the learner. If the sample is small, the hypothesis selected within the hypothesis class may be far from the best one, resulting in an another type of error. Such an *estimation error* may be denoted as $d(h_t, h_\infty)$. *Sample complexity*, for any given hypothesis class, measures the number of examples that the learner needs in order to obtain a low estimation error.

Hypothesis complexity generates a trade-off: simpler hypothesis classes (low hypothesis complexity) will imply larger approximation errors, but also lower sample complexity. On the other hand, if we try to reduce the approximation error by increasing the hypothesis complexity, we also increase the sample complexity, meaning that for the same number of examples fed to the learner, the estimation error will be larger.

This trade-off is equivalent to the one familiar to the economists faced in regression analysis. By increasing the number of regressors we can increase the R^2 (decrease the approximation error); but we also decrease the number of degrees of freedom, i.e the ability to generalize (a higher Standard Error and poorer out-of-sample forecasting). In the literature on learning this is referred to as *overfitting*: a learner using a complex hypothesis space may become very good in fitting the sample, but nevertheless may not be able to generalize. In other words, the learner would be learning the sample instead of the underlying concept. The tension between these two different effects will imply that there is an "optimal" hypothesis complexity, n^*, defined as the dimension that makes the bound to the generalization error minimum. Fig. 1 shows this point.

For a very low size of H, the gain in approximation obtained by increasing n outweighs the loss in estimation. After a certain point, instead, the gain in approximation is not enough to compensate for the loss in "degrees of freedom"; as a result total error will stop decreasing. This trade-off has a very natural translation in economics, as we discuss in the following Section 2.2.

The choice of an adequate hypothesis class is one of the main challenges of CLT. Computer scientists refer to it as to *structural risk minimization*. We define, for a given hypothesis complexity n, the average performance of the algorithm after t samples

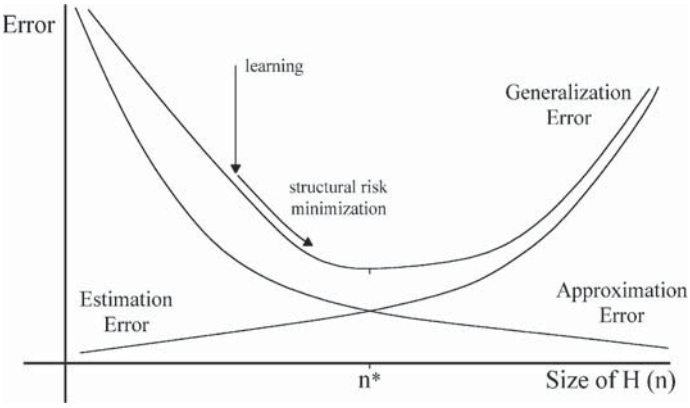

Fig. 1. Errors and hypothesis complexity.

are drawn as

$$J_n(h) = \frac{1}{t}\sum_{\tau=1}^{t} d(h_\tau(c, \boldsymbol{x}), c),$$

where $d(\cdot)$ is the distance measure.

Learning minimizes this value for a *given hypothesis complexity*, i.e. it finds, among the different hypotheses of size n, the one that has the lowest average generalization error $J_n^*(h) = \min_{h \in H_n} J_n(h)$.

With more complex hypothesis spaces (larger n), we reduce the estimation error, but we lose the ability to generalize (we increase the approximation error). The trade-off problem may be solved by looking for the best balance between approximation and estimation, i.e. by locating

$$J^*(h) = \min_n J_n^*(h).$$

The corresponding hypothesis complexity (n^*) is the one that minimizes the generalization error. In terms of Fig. 1, learning is the process, for a given value of n, of reducing the error to its lowest bound, represented by the generalization error curve. Structural risk minimization on the other hand, represents movements along the x-axis (i.e. comparison of algorithms of different complexity) in order to find the value of n that minimizes the lower bound.

Structural risk minimization is essentially the process of having different algorithms (hypothesis classes) learn the concept as well as possible, and then selecting the one with the best performance.[2]

A simple example may help to clarify the concepts discussed earlier. Suppose the *learner* is a monopolist firm; it faces a demand function $p = f(q, \Omega)$, where Ω is a vector of variables affecting the demand, such as GDP, real interest rates, prices of complements and

[2] Unlike computer scientists, we are also concerned with the economic costs of different hypotheses classes. We discuss them in Section 2.3.

substitutes, etc. The *instance space* X is the m-dimensional space $(p, q, \Omega_1, \ldots, \Omega_{m-2})$. The *concept space* is the set of all possible demand functions $p = f(q, \Omega)$.

Suppose that the firm assumes that the demand function is linear in q, and furthermore that its slope is fixed and known, whereas the position depends on Ω. In other words, the *hypothesis class* is the set of all functions such that $p = \alpha(\Omega) - \beta(q)$.

In each period τ, the firm draws an example $x \in X$, and processes it. The output $\hat{\alpha}_\tau$ is used to calculate the profit maximizing price, $\hat{p}_\tau$. The market plays the role of the *oracle*. If the profit obtained charging $\hat{p}_\tau$ is high, this means that the guess was good; if it is low, then the algorithm has to be modified, and another hypothesis has to be chosen. After t periods the firm has a *sample* of guesses about α and of the corresponding profit levels, and it makes its best estimate about the demand function. Note, however, in this example, the oracle returns the market-clearing quantity, which the firm then uses to update the value of α.

The hypothesis complexity may be insufficient, so that learning does not take place. Suppose that the real demand curve, the *concept* c, was $p = \alpha(\Omega) - \beta(q, \Omega)$, with the slope also dependent on Ω; then, a linear (in q) hypothesis class would never yield optimal profits. On the other hand, if the demand function was linear and the hypothesis class assumed a more complex relationship between q and Ω, then time and resources would be wasted only for the firm to learn that Ω has no effect on β.

2.1. Artificial neural networks

In its most basic description an ANN can be thought of as a form of black box input–output system.[3] The network takes a set of inputs, applies a set of parameters (weights) and a transformation (squashing) function, and then produces an output. In addition, ANNs are learning machines: as they are presented with data, the weights are adjusted (trained) in order to learn the underlying patterns that generate the data.

The data are 'input' into the first (input) layer and are then processed by one or more intermediate (hidden) layers. The network then produces an output which is compared to the actual output. The difference is the network error. The network then adjusts its weights according to a learning algorithm that, if powerful enough, at each successive step reduces the error.[4]

Each layer consists of nodes that are the site of activity, where the data are transformed (this is not true for the input nodes). A graphical representation of the network is shown in Fig. 2. Notice the parallel nature of the data processing, which is an important feature of the network.

The input data is a vector $\boldsymbol{x}$ of length m. Each node of the hidden layer receives a weighted sum of the data from the input layer. That is, the input into the ith node in the hidden

[3] For a detailed treatment of neural networks in general see Croall and Mason (1992) and Freeman (1994). For statistical applications see Kuan and White (1994). For a game theoretic application see Cho (1994).

[4] The network we use here is referred to as a 'backward propagation network' (BPN), because during the learning stage, the weight adjustment is first done in the final (output) layer and is then propagated backwards to the first layer. BPNs are often used for pattern recognition and the learning of complex, non-linear functions. Most applications only use one hidden layer.

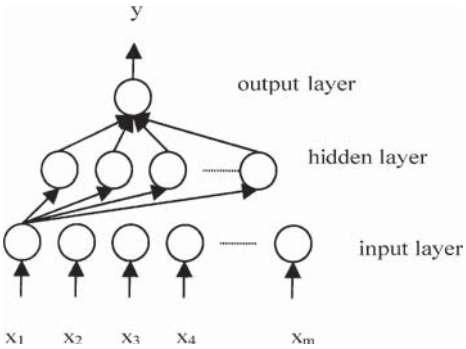

Fig. 2. Backward propagation network (note: not all connections shown).

layer is

$$\text{in}_i^{\text{h}} = \boldsymbol{w}_i^{\text{h}}\boldsymbol{x} = \left(w_{1i}^{\text{h}}x_1 + \cdots + w_{mi}^{\text{h}}x_m\right),$$

where $\boldsymbol{w}_i^{\text{h}}$ is an $1 \times m$ vector of weights. The set of inputs to the n nodes of the hidden layer is $\mathbf{in}^{\text{h}} = (\boldsymbol{w}_1^{\text{h}}\boldsymbol{x}, \ldots, \boldsymbol{w}_i^{\text{h}}\boldsymbol{x}, \ldots, \boldsymbol{w}_n^{\text{h}}\boldsymbol{x})$. Each input to a hidden node is transformed, via a squashing function, into an hidden-layer output (out_i^{h}). One of the most commonly used squashing functions is the sigmoid function, $\text{out}_i^{\text{h}} = g(\text{in}_i^{\text{h}}) = [1 + \text{e}^{(-\text{in}_i^{\text{h}})}]^{-1}$, which is a continuous approximation of the indicator function. Low values are squashed to zero ("no"); high values are squashed to one ("yes"). The vector of outputs from the hidden layer is

$$\mathbf{out}^{\text{h}} = g(\text{in}_1^{\text{h}}), \ldots, g(\text{in}_i^{\text{h}}), \ldots, g(\text{in}_n^{\text{h}}).$$

The input to the final layer is a weighted sum of all the outputs from the hidden layer: $\text{in}^{\text{o}} = \boldsymbol{w}^{\text{o}}\mathbf{out}^{\text{h}}$, where $\boldsymbol{w}^{\text{o}} = w_1^{\text{o}}, \ldots, w_i^{\text{o}}, \ldots, w_n^{\text{o}}$, is an $1 \times n$ vector of weights. Finally, the output of the network $\hat{y}$ of the network is determined by transforming in^{o} with the sigmoid function $\hat{y} = g(\text{in}^{\text{o}})$. In sum, we can express the network with one hidden layer as

$$\hat{y} = g\left[\sum_{i=1}^{n} w_i^{\text{o}}\, g(\boldsymbol{w}_i^{\text{h}}\boldsymbol{x})\right]. \tag{1}$$

2.1.1. The learning algorithm

The above process describes the input–output nature of the neural network. Here we show how the network learns. We begin with a completely untrained network by selecting random weight values. An example is selected from the example set (according to some probability law) and is fed to the network, which processes it as described earlier to obtain an output, $\hat{y}$. This output is compared to the ideal output y (given by the oracle), to determine the corresponding error: $\xi = (y - \hat{y})^2/2$. This information is then propagated backwards as the weights are adjusted according to the learning algorithm, which aims to minimize the

squared error, ξ. The gradient of ξ with respect to the output-layer weights is

$$\frac{\partial \xi}{\partial w_i^{\mathrm{o}}} = -(y - g(\mathrm{in}^{\mathrm{o}}))g'(\mathrm{in}^{\mathrm{o}})\mathrm{out}_i^{\mathrm{h}}, \quad i = 1, \ldots, n.$$

Given that $\partial(\mathrm{in}^{\mathrm{o}})/\partial(w_i^{\mathrm{o}}) = \mathrm{out}_i^{\mathrm{h}}$, and for the sigmoid function, $g'(\mathrm{in}^{\mathrm{o}}) = \hat{y}(1 - \hat{y})$. Similarly, we can find the gradient of the error surface with respect to the hidden layer weights:

$$\frac{\partial \xi}{\partial w_{ji}^{\mathrm{h}}} = -g'(\mathrm{in}_i^{\mathrm{h}})x_j[(y - g(\mathrm{in}^{\mathrm{o}}))g'(\mathrm{in}^{\mathrm{o}})w_i^{\mathrm{o}}], \qquad j = 1, \ldots, m; \quad i = 1, \ldots, n.$$

We see where the concept of backward propagation enters formally: we calculate the error of the output layer first, then bring the error back to the hidden layer to calculate the surface gradients there.

Once we have calculated the gradients, we adjust each weight value a small amount in the opposite (negative) direction of the gradient. We introduce a proportionality constant η, the learning-rate parameter, to smooth the updating process. Thus, if we define $\delta^{\mathrm{o}} = (y - \hat{y})g'(\mathrm{in}^{\mathrm{o}})$, we have the weight adjustment for the output layer as $w_i^{\mathrm{o}}(t+1) = w_i^{\mathrm{o}}(t) + \eta\delta^{\mathrm{o}}\mathrm{out}_i^{\mathrm{h}}$, for $i = 1, \ldots, n$. Similarly, for the hidden layer, $w_{ji}^{\mathrm{h}}(t+1) = w_{ji}^{\mathrm{h}}(t) + \eta\delta_i^{\mathrm{h}}x_j$, where $\delta_i^{\mathrm{h}} = g'(\mathrm{in}_i^{\mathrm{h}})\delta^{\mathrm{o}}w_i^{\mathrm{o}}$.

When the updating of weights is done, we present the next input pattern and repeat the weight-update process. The process normally continues until a pre-specified small error is achieved, and the network is said to have learned the data.

2.2. *The firm, CLT and artificial neural networks*

We believe it is useful to view the firm as a *learning algorithm* which produces a hypothesis based on the information from the environment. The organization consists of agents who follow a series of rules and procedures organized in both a parallel and serial manner. Firms learn and improve their performance by repeating their actions and recognizing patterns (i.e. learning by doing). As the firm processes information (iterates), it learns its particular environment and becomes proficient at recognizing new and related information.

Furthermore, the firm, like learning algorithms in general, faces a trade-off linked to the complexity of its organization. Small firms (hypothesis classes) are likely to be rather imprecise in understanding the environment they face; but on the other hand they act relatively quickly and are able to design decent strategies with small amounts of experience. Larger and more complex firms, on the other hand, produce more sophisticated analyses, but they need time and experience to do so. Translating in the language of CLT, small firms have a simple hypothesis class, and hence quickly find the best hypothesis; more complex organizations take more iterations but, once reached, the best hypothesis within their class is more likely to be close to the real concept. Finding the optimal structure of the firm (given its economic costs) may then be seen as a form of structural risk minimization. Unlike computer science, however, in economics the search for an optimal structure occurs given a competitive landscape, which imposes time and money constraints on the firm,

and consequently forces it to abandon the objective of theoretical learnability. In this paper we compare, for a given environment, different firm structures (hypothesis complexity), in order to identify the most profitable one (i.e. the one minimizing empirical risk, given the economic costs).

2.3. The cost of learning

We consider two different costs of information processing. The first is the 'time' it takes for the organization to learn; the second is the cost per information processing unit.

2.3.1. Time to learn

A general feature influencing firm behavior (especially when dealing with innovation) is that it faces *time constraints*. In this paper we define 'time' as the number of steps an algorithm needs to perform a task (Wilf, 1986). (Radner (1993) refers to it as the 'delay'). It is obtained by summing the number of steps performed at each layer.

In the case of a neural network with one hidden layer, we determine the total delay as follows. The input into each node in the hidden layer is the dot product of the m inputs and their weights. This makes m operations, which has a cost of $c_1 m$, if each multiplication takes c_1 units of time. Then, we add together the numbers, $(m-1)$ operations for a cost of $c_2(m-1)$. Each node then produces an output by applying the sigmoid function to the weighted sum, with a cost of c_3. From the hidden layer to the output we have a similar number of steps, but with n nodes. Thus, the total of forward steps is $c_1(m + n) + c_2(m + n - 2) + 2c_3$. The number of backward steps in calculated in a similar manner, so that we obtain total delay as a linear function of m and n: $c = \alpha_0 + \alpha_1(m + n)$.

2.3.2. Cost per node

We also assume that the firm must pay a fee (wage) per information processing unit. This gives a cost of $w = \alpha_2 n$.

Cost per iteration is then $c + w = \alpha_0 + \alpha_1 m + (\alpha_1 + \alpha_2)n$, and if the network makes t iterations, total cost is given by

$$C = t(c + w)$$

3. Complexity, stability and optimal structure: a simulation experiment

This section introduces a simple model of the firm as an ANN. The example of this section has the aim of illustrating the applicability of ANNs to the study of organizational learning; as such, it is aimed at emphasizing the general relationship between the firm's information processing and its environment and should not be seen as modeling a specific component of firm behavior.[5] Nevertheless, we provide two examples in Section 4 of how complexity and stability affect firm performance and structure.

[5] Our model builds on and extends Radner where learning, however, is not an issue; his network can be seen as a particular case of a feedforward network, with weights and squashing functions fixed beforehand and with constant 'environment.'

Our main goal is show how the optimal organizational size is inextricably linked to the environmental characteristics. We parametrize these characteristics along two dimensions—complexity and stability—which generate four possible types of environments: stable/simple (s, S), stable/complex (s, C), unstable/simple (u, S), and unstable/complex (u, C). For each of these configurations, we study the relationship between the firm/network size, and its performance; in other words, we investigate the trade-off discussed in Section 2, and how it changes with different environmental scenarios.

3.1. Description of the experiment

The firm is modeled as a backward propagation network (BPN) with a single hidden layer.[6] The number of input nodes is $m = 10$, while the number of hidden nodes n is variable, and is the measure of firm dimension, or equivalently of hypothesis complexity. Structural risk minimization hence, in this framework, is simply the search for the optimal number of nodes in the hidden layer. The task to be learned is a simple pattern recognition: given the binary inputs $\{0, 1\}^{10}$, the desired output is the equivalent number in base 10 (normalized to lie in the interval [0, 1]).

This abstract task may be interpreted in several different ways as modeling real-life behavior of firms; for instance, it could represent a marketing department trying to establish a link (a pattern) between a series of consumer characteristics (1: present; 0: absent) and a certain degree of consumer acceptance of the product (the corresponding base 10 value). Or, it might be seen as a production input–output pattern: certain inputs are linked by a particular rule to an output value, so that the task might be seen as a form of production function learning (in the spirit of Jovanovic and Nyarko (1996)).

The process works as follows. The network draws a vector of binary digits from the example set according to a given probability law (discussed later). The feed-forward process summarized by Eq. (1) produces an estimated output $\hat{y}$. The oracle (or marketplace) then gives the error (the distance between the estimated output and the equivalent in base 10 of the input vector): $\xi = d(y_\tau, \hat{y}_\tau) = 1/2(y_\tau - \hat{y}_\tau)^2$. The network uses this error to adjust the weights according to the gradient descent algorithm discussed in Section 2.1.

Through successive iterations (draws and estimations) the network updates the weights (whose initial values are randomly chosen) to produce an output close to the correct one. We limit the number of iterations to $t = 250$ as a way to simulate the limited time period that firms have to learn. In each period/iteration τ, we define the profit of the firm of dimension $n(\pi_\tau^n)$ as the inverse of the squared error (the lower the error, the better the job is done and thus the higher the "revenues") minus the cost of the network, as defined in Section 2.3:

$$\pi_\tau^n = \frac{\beta}{\xi_\tau} - [\alpha_0 + \alpha_1 m + (\alpha_1 + \alpha_2)n],$$

where β, α_0, α_1, α_2 are the parameters. (We can think of the firm as receiving a pay-off β each period that is affected by the inverse of the error $1/\xi_\tau$, so that the revenue is given by

[6] Note that by assuming only one hidden layer, we do not directly address the issue of hierarchy.

$(1/\xi_\tau)\beta$.) Notice that delay only depends on the dimension of the network (and the number of inputs), while if the task is learnable, ξ_τ decreases with the iterations. Larger firms have greater delay and wage costs. They will perform better than smaller ones only if their errors (dependent on the interaction between environmental characteristics and network structure) decrease by more and/or at a faster rate.

Total profit is calculated as the sum over the $t = 250$ periods (iterations) of the single period profits:

$$\Pi^n = \sum_{\tau=1}^{t} \pi_\tau^n = \sum_{\tau=1}^{t} \frac{\beta}{\xi_\tau} - [\alpha_0 + \alpha_1 m + (\alpha_1 + \alpha_2)n]t$$

Because of the stochastic outcome of the initial weight vector, we repeat the whole learning process 300 times, and take the average of the total profit.

3.1.1. Complexity

We measure the environmental complexity by the smoothness of the probability distribution used to produce examples. The simplest environment is the one in which one example arrives each period with certainty, while the most complex is one in which all the examples are equally probable. In other words, by altering the probability distribution, we can make the example class (and hence the concept space) more or less complex. A natural measure of the smoothness of the distribution is its entropy. Suppose the example set contains ν elements (in our simulations we have $\nu = 10$), then entropy will be

$$E = -\frac{\sum_{i=1}^{\nu} p_i \ln(p_i)}{\ln(\nu)},$$

where $p_i \in [0, 1]$, $\sum p_i = 1$. Entropy is increasing in complexity, and ranges between 0, for a degenerate distribution, and 1 when the distribution is uniform ($p_i = 1/\nu$, and the numerator is equal to $\ln(\nu)$).[7] In Section 4, we discuss a real-world example of how complexity can affect firm performance and structure.

3.1.2. Stability

We assume that at each period there is a probability ρ that the environment changes, in the sense that one element of the example set drops out, and is substituted by a different one, even if the new elements still belonging to the concept set (if follows the same rule as the one that it replaces, i.e. associate a vector of binary numbers of 10 digits to the equivalent number in base 10). By altering the value of ρ we can change the speed at which the environment changes, i.e. its stability. A very stable environment will be characterized by low values of ρ, while a fast changing one will have a very high ρ. In Section 4, we argue that changes in the stability of the environment were a factor in the evolution of firm performance and competition in the computer industry.

[7] We adopt the convention that $0 \ln(0) = 0$. Entropy is usually defined as the numerator of E, so that it varies in the interval $[0, \ln(\nu)]$. Thus, we have normalized it. Entropy is a crucial component of Information Theory (Mansuripur, 1987). See Arrow (1985) and Marshak (1954) for applications in economics.

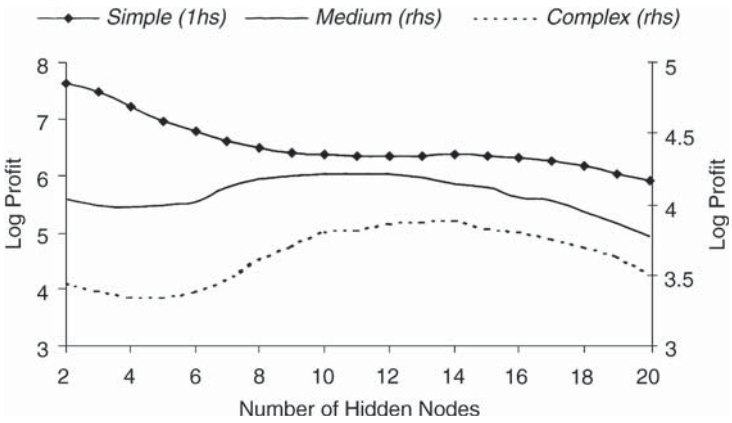

Fig. 3. Network size and (log) profit for different levels of environmental complexity.

3.2. The simulations

This section will investigate by means of simulations the relationship between firm size and environmental characteristics.[8] We will first study the environments of different complexity (keeping stability constant), and then focus on the stability dimension.

3.2.1. The (s, S) and (s, C) environments

Beginning with the most stable environment ($\rho = 0$), we investigated the performance of networks of different sizes for increasing levels of complexity. Starting with environments that are very simple to learn (i.e. most of the distribution is concentrated on two elements), we gradually increased the complexity by smoothing the probability distribution. We examined three different levels of complexity: low, medium, high.[9] For each of them, we calculated the total profit for firms of different size (n goes from 2 to 20). The results, reported in Fig. 3, confirm our conjectures.

First, the three curves never intersect, meaning that profits are decreasing in environmental complexity. In other words, learning simpler environments is easier (and hence more profitable) *regardless* of firm dimension. Second, the trade-off between larger costs and better performance is evident from the hump shaped profit curves: small firms have poorer 'revenues,' but have lower costs, while large firms have high costs that offset the gain in performance. This does not hold if the environment is extremely simple; in that case, the small firm does better *and* has lower costs.

[8] The simulations were run in Mathematica 3.0. The code is available upon request. The parameters we used for the simulations are $\beta = 5, \alpha_0 = 0, \alpha_1 = 1, \alpha_2 = 1, \eta = 11$.

[9] The low complexity environment has all the probabilities concentrated on two values (one value would be trivial): $p_1 = p_2 = 0.5$, $p_i = 0$, $i = 3, \ldots, 10$. Entropy in this case is $E = 0.277$. The medium complexity environment has probabilities concentrated on the first five elements, with zeros in the other five ($p_j = 0.2$, $p_i = 0$, $j = 1, \ldots, 5$; $i = 6, \ldots, 10$). The entropy value is $E = 0.77$. Finally, the case of maximum complexity has a uniform distribution, with $p_i = 0.1$, and $E = 1$. Notice that the complexity of the environment boils down to the number of elements the network has to learn to distinguish: 2, 5, 10, respectively.

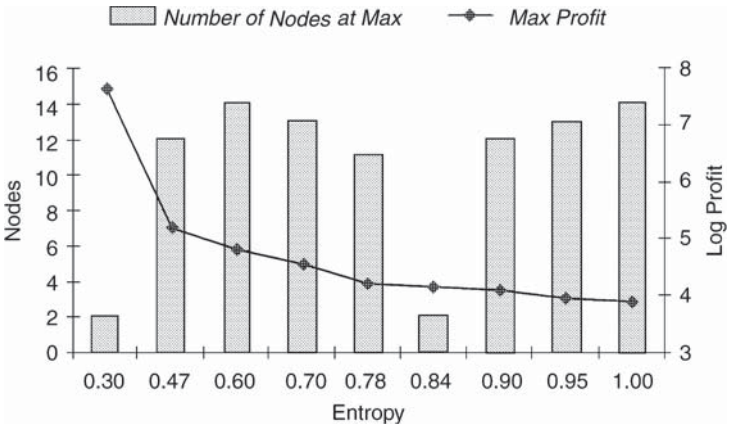

Fig. 4. Optimal dimension and maximum profit for increasing environmental complexity.

To investigate the relationship between environmental and organizational complexity, we increased the entropy values by making the probability distribution more and more uniform.[10] The results are shown in Fig. 4.

First, the figure confirms the results we had earlier for only three entropy levels: the maximum attainable profit (the continuous line) is decreasing in environmental complexity. The relationship between environmental and organizational complexity (the bar chart), nevertheless, is not monotone. There is no apparent relationship between entropy levels and the number of nodes for which profit is maximum.

In summary, the first set of simulations gives us some interesting insights. We can see that complex environments are unambiguously associated with lower profits, for any organizational dimension. Furthermore, the trade-off we described earlier between performance and flexibility is visible and robust for complex environments. Finally, there is no clear relationship between environmental complexity and firm dimension/complexity.

3.2.2. The (u, S) and (u, C) environments

The second set of simulations investigated the network behavior in environments characterized by various levels of stability. We took different values of ρ (the probability of changes in the example set), and studied how different networks perform. The experiment was run for a very complex environment ($E = 1$), and for a very simple one ($E = 0.27$). The case of a complex environment is depicted in Figs. 5 and 6.

Fig. 5 shows two things. First, with the exception of the first one, curves corresponding to increasing levels of ρ lie below each other. This means that more unstable environments yield lower profits irrespective of the network size. Second, we find here, as we had in Fig. 3, that for each level of environmental stability, profits are hump shaped. In other words, as

[10] Each time the number of elements of the example set with 0 probability was reduced by 1, and all the elements with positive probability had equal chance of being drawn.

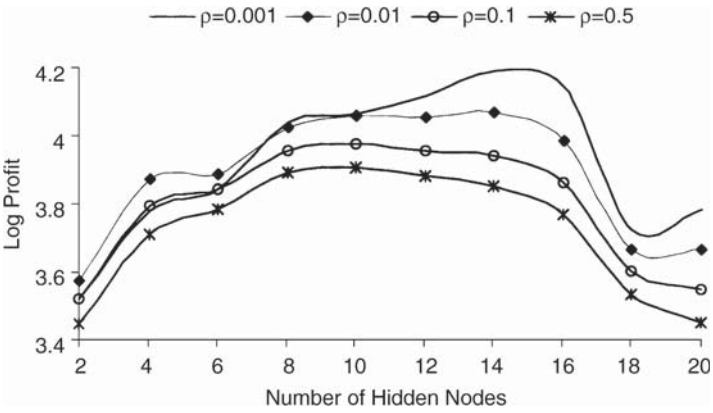

Fig. 5. Environmental complexity and network profit in a high complexity environment ($E = 1$).

we had earlier for complexity, along the stability dimension the trade-off of Section 2 is relevant, and an optimal size may be determined.

Fig. 6 shows the relationship between environmental stability and the profit maximizing network size (with the corresponding profit). The first fact to notice is that, as in the case of complexity, the maximum attainable profit is lower in more unstable environments: learning is more difficult, and errors decrease less. The second interesting feature of the simulation is that here *we can see a pattern in network/firm optimal size*: the more unstable the environment, the smaller the optimal size of the network. The continuous change of the samples presented to the network is in fact more disrupting when the network has to update many weights. Similarly, a large company has to revise a complex system of relations in response to changes of the environment; and when this change occurs too frequently performance is

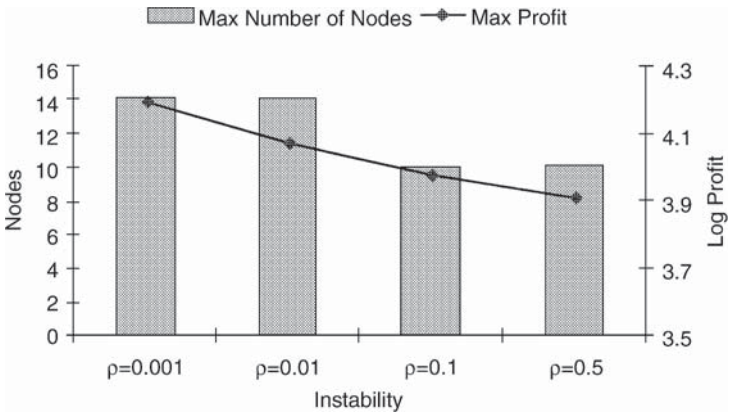

Fig. 6. Optimal size and maximum attainable profit for different levels of environmental instability (complex environment).

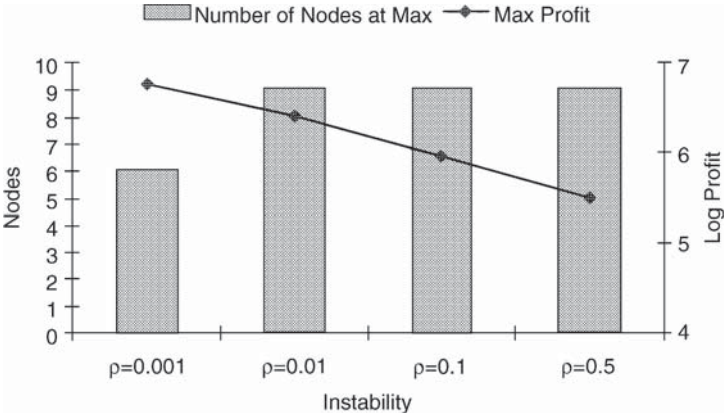

Fig. 7. Optimal size and maximum attainable profit for different levels of environmental instability (simple environment).

affected. We will argue later that a similar argument may be among the factors explaining the evolution of the personal computer industry.

Next, we ran the same experiment in a very simple environment. The behavior of the profit curve is very similar to the one of Fig. 5, and hence it is not reported. Notice that the hump shape of the profit function is an extremely robust feature, present in all the simulations we made. On the other hand, Fig. 7 shows that in a very simple environment, the inverse relationship between environmental instability and network/firm size, that we had in Fig. 6 disappears.[11] This can be explained by the fact that low entropy means that most of the probability is concentrated on few elements of the sample; this implies that most of the time, when an element is dropped and substituted by a new one this has no influence, given that the element itself had 0 probability of being drawn.

To conclude, the simulations performed in this section show some extremely robust results.

- The network/firm faces a trade-off, between performance and costs of the structure. This trade-off is apparent in the hump shaped profit curve, and has a direct parallel in economics, in the trade-off between coordination costs and computational power.
- Increasing complexity of the environment implies decreasing performance and hence profits.
- Increasing instability of the environment implies decreasing performance and hence profits.
- We also found a very interesting result that only appears in complex environments: the optimal size of the firm/network decreases as the instability of the environment increases. Large networks are not able to cope with a rapidly changing environment.

In the next section, we discuss some examples of industrial evolution and adaptation.

[11] Notice further that profits are much larger in size in Fig. 7 than in Fig. 6. This confirms the finding of Section 3.2.1, that increasing complexity implies worse performances.

4. Discussion

Here we discuss two examples that are meant simply to illustrate how complexity and stability can affect firm performance.

4.1. Stability and the personal computer market

By the late 1970s IBM was the world's largest computer firm and the number one producer of mainframes. IBM's success was due to both its strategic product introductions and its successful approach to customer service. The life-cycle for the mainframe was several years; and IBM had grown to be a large bureaucratic institution designed to sell and rent relatively few machines each year. The mainframe market was a relatively stable one.

With the introduction of the personal computer (PC) the product cycle rapidly increased. New machines were being introduced every 18 months—the length of time that Intel needed to produce a new generation of microprocessor. Even though IBM had, in many ways, spawned the PC market as we know it, it was ill prepared to compete in a world of rapid change and high instability. So much so that by the early 1990s the company was losing money and needed to undertake a tremendous and expensive restructuring in order to remain competitive.

In the sense outlined in Section 3.2.2, we see the mainframe market as a stable technological environment (a low ρ). IBM had had the time to adapt to the relatively slow technology cycles. The introduction of the microprocessor was an exogenous shock, a sudden increase in the stability parameter ρ. The environment suddenly became very unstable, with rapidly changing technological standards, consumer needs, market conditions and product cycles. IBM found itself with an inefficient structure that was too slow to adapt; it was unable to fight its smaller and more dynamic competitors, which quickly outperformed it.

4.2. Complexity and the conglomerate

In Section 3, we measure the complexity of the information processing task by the entropy of the dataset. In other words, the complexity measures the "quantity" of information that the firm is likely to process.

In the late 1960s, firms increased dramatically the rate of mergers and acquisitions. Such unprecedented diversification led to the separation of top management at the corporate office from the middle managers who were responsible for running the daily operations. This separation occurred for two reasons. One was that the top management had little knowledge or experience with the technological processes and markets of the divisions or subsidiaries they acquired. The second was simply that the large number of acquired businesses created an extraordinary overload in decision making at the corporate office. As a result, throughout the 1970s the number of divestitures increased dramatically. Over time, companies divested businesses unrelated to their core focus, and narrowed their product lines (Chandler, 1990).

The simulation in Section 3.2.1 shows that increasing entropy is associated with decreasing profit from information processing. We can relate this to the experience of conglomerates

in the 1960s and 1970s. Firms believed that expansion in unrelated product areas would not tax the information processing capabilities of the firm. However, expansion into these areas produced a large jump in the amount of information managers had to process and as a result they found that conglomerates were suffering lower profits. When firms divested they decreased the amount of information and focused on information and production in areas for which they had expertise, and, as a result, made the information load easier and increased their subsequent profits.

5. Conclusion

In this paper we have argued for the importance of viewing the firm not as a production function, but rather as an information processing and learning algorithm. In this way, we can use results from computational learning theory to help model the firm and its relationship to the environment. Our model is an attempt to begin to bridge two streams of literature. On one hand, the management literature looks at the relationship between firm structure, performance, and environmental characteristics. Management scholars have shown that the necessary amount of information processing by the firm is positively related to the degree of uncertainty in its environment. In order to avoid information processing overload, the firm creates managerial positions and divides workers into different specialized subunits, such as production, sales/marketing, and research. The optimal number of workers per manager depends on both the complexity and the uncertainty of the information flows (Burns and Stalker, 1994; Galbraith, 1973). The more the firm differentiates into sub-units to focus on a specific areas, the more costly it is to integrate these various sub-units. The firm thus tries to find an optimal balance between differentiation and integration (Lawrence and Lorsch, 1986).

On the economics side, our simulation model is in the same spirit as the recent agent-based models of the organization. These models borrow heavily from computer science models of data processing. Radner (1993), for example, proposes a model of the firm as a simple parallel processing algorithm, and shows that the network which maximize the speed of processing is hierarchical. Bolton and Dewatripont (1994) model the communication and coordination costs associated with decentralization. DeCanio and Watkins (1998) construct a simple contagion model of information transmission among agents. Decanio et al. (2000) model the firm as a graph; the optimal network is one that finds the best speed of information transmission given its cost. Finally, there is also a growing body of work on computational organization theory, which uses computer simulations to model the structure of organizations (Carley and Prietula, 1994). We seek to add to this literature by studying two additional components: *firm learning and pattern recognition; and the relationship between firm structure and environment*.

We believe our methodology leads to a large number of possible research opportunities. For example, Barr et al. (2001) explore more fully how computational learning theory can model the learning of the production function. Barr (2001) models decision making and organizational structure by blending rugged landscapes and artificial neural networks. Future research will also consider competition among networks of different sizes and capabilities.

Acknowledgements

We would like to thank Duncan Foley, who has provided us with inspiration and guidance. Oliver Holle was an active part of this project at its early stages; we gratefully acknowledge his role. We would also like to thank Eleazar Eskin for his help with the working of the networks. We appreciate the useful comments from Richard Ericson, Paul Sengmüller and two anonymous referees. Finally, we thank the participants in various seminars at Columbia University and New York University. The usual caveats apply.

References

Arrow, K.J., 1985. Informational structure of the firm. American Economic Review, PAP 75, 303–307.
Barr, J., 2001. Firm information processing, organization and adaptation. Mimeo, Columbia University.
Barr, J., Eskin, E., Saraceno, F., 2001. Environment, complexity and organizational structure: a model of firm learning. Mimeo, Columbia University.
Bolton, P., Dewatripont, M., 1994. The firm as a communication network. Quarterly Journal of Economics 109, 809–839.
Burns, T., Stalker, G.M., 1994. The management of innovation (Oxford University Press, New York).
Carley, Prietula, 1994. Computational organization theory. Lawerence Elrbaum Associates, Hillsdale.
Chandler Jr., A.D., 1990. The enduring logic of industrial success. Harvard Business Review, March–April, 130–140.
Cho, I., 1994. Bounded rationality, neural networks and folk theorem in repeated games with discounting. Economic Theory 4, 935–957.
Croall, Mason, 1992. Industrial Applications of Neural Networks: Project ANNIE Handbook, Springer, New York.
DeCanio, S.J., Watkins, W.E., 1998. Information processing and organizational structure. Journal of Economic Behavior and Organization 36, 275–294.
Decanio, S.J., Dibble, C., Amir-Atefi, K., 2000. The importance of organizational structure for adoption of innovations. Management Science 46 (10), 1285–1299.
Freeman, J.A., 1994. Simulating Neural Networks with Mathematica. Addision-Wesley, New York.
Galbraith, J., 1973. Designing Complex Organizations, Addison-Wesley, Reading.
Jovanovic, Nyarko, 1996. Learning by doing and the choice of technology. Econometrica 64 (6), 1299–1310.
Kearns, M.J., Vazirani, U.V., 1994. An Introduction to Computational Learning Theory. MIT Press, Cambridge.
Kuan, C.-M., White, H., 1994. Artificial neural networks: an econometric perspective. Econometric Reviews 13 (1), 1–91.
Lawrence, P.R., Lorsch, J.W., 1986. Organization and Environment. Harvard Business School Press, Boston.
Mansuripur, M., 1987. Introduction to Information Theory. Prentice Hall, Englewood Cliffs.
Marshak, J., 1954. Towards an economic theory of organization and information. In: Thrall, R.M. (Eds.), Decision Theory. Wiley, New York.
Niyogi, P., 1998. The informational Complexity of Learning. Kluwer Academic Publishers, Boston.
Radner, R., 1993. The organization of decentralized information processing. Econometrica 61 (5), 1109–1146.
Vapnik, V.N., 1995. The Nature of Statistical Learning Theory. Springer, New York.
Vidyasagar, M., 1997. A Theory of Learning and Generalization. Springer, New York.
Wilf. S., 1986. Algorithms and Complexity. Prentice Hall, Englewood Cliffs.