Neural Networks for Encoding and Adapting in Dynamic Economies

In-Koo Cho
Department of Economics
University of Chicago

and

Thomas J. Sargent
Hoover Institution
Stanford University

and

Department of Economics
University of Chicago

April 19, 1995

Note: We thank Blake LeBaron, John Rust, and an anonymous referee for helpful comments. Parts of this paper draw heavily on materials in chapters 3 and 4 of Sargent's *Bounded Rationality in Macroeconomics*, Oxford University Press, 1993.

Neural Networks for Encoding and Adapting in Dynamic Economies

Introduction

During the last 15 years, neural networks have engaged the attention of researchers in diverse fields of science. This paper is an introduction to using neural networks in economic dynamics. For macroeconomics and repeated games, we are always looking for economical ways of formulating and characterizing *equilibria* – collections of strategies that satisfy our usual assumptions about individual rationality and common understandings. In this paper we adduce several examples that show neural networks to be fruitful sources of 'functional forms' for conveniently *representing* equilibria; for formulating ways of *computing* them; and for studying their *stability*.

We begin by describing feedforward neural networks as approximators, and relate them to statistical discriminant functions. Then we describe how neural nets of varying complexity can represent equilibria in two repeated games and one dynamic economic model. These three examples are designed to stretch neural networks in different directions. We start with a repeated prisoner's dilemma game, and use it to show the power of the perceptron as a device for encoding strategies. We define linear strategies, display perceptrons that encode them, and show that for the prisoner's dilemma game, these strategies are discriminating enough to support all subgame perfect payoff vectors. This outcome stems from the feature of the payoff matrix for the prisoner's dilemma game that *linear* strategies discriminate sufficiently among alternative histories. In the spirit of Minsky and Papert, we study whether linear strategies are enough in other two-player games. They are not, but for a general class of two-player games only a slightly more complex network *is* discriminating enough to support *all* equilibrium payoff vectors.

Inspired by work of Atkeson (1991) and Marcet and Marimon (1992), we then study a mechanism design game associated with repeated moral hazard in a stochastic growth model. The capital stock has dynamics chosen by a private sector assisted by a planner who has access to outside resources but who cannot monitor the investment level of the private sector and who must therefore rely on some sort of statistical test to infer the private sector's past behavior. [1] We show how simple neural networks can be applied to this example to sustain almost every average payoff vector beyond the autarchy level.

Each of our examples starts from the observation that the simplest neural network – the single unit perceptron – can implement a trigger strategy. This allows the neural network to encode enough history-dependent strategies to recover a large set of equilibria in repeated games and other dynamic economic settings. Our encoding of history-dependent strategies uses the insight

[1] The existence of the capital stock prevents application of the review strategies of Radner (1985).

that simple 'recurrent' neural networks can implement *accounting systems* for efficiently tracking the rewards associated with past behavior patterns. The *classifier* system of Holland, which we briefly recount, also centers around an accounting system.

Thus, we import two themes from the artificial intelligence literature: classification and accounting. We also touch briefly on a third theme, adaptation. Their thrift in encoding makes us optimistic about neural networks as devices in terms of which we shall eventually be able to formulate theories of learning and evolution over history-dependent strategies. We conclude this essay with some speculations about ways that these possibilities might soon be realized.

Discriminant functions

Players who repeatedly engage in a stage game with k actions must somehow *classify* histories into k categories to prompt one of the k actions. To prepare the way, we briefly describe statistical *discriminant functions*, a classic tool for studying dichotomous or polychotomous choices, and a bridge between classical linear statistical methods and neural networks. The simplest manifestations of these methods classify vectors of independent variables $x \in X$ for the purpose of predicting some associated characteristics $y \in Y$. In inspecting these methods, we should keep in the back of our minds that the X set that shall eventually interest us must somehow summarize the history of play.

Fisher (1936) used least squares regression to find a *linear discriminant function* for determining to which of two predetermined classes an individual belongs. [2] We are given two populations, $x_1 \in X_1$ and $x_2 \in X_2$, each of $k \times 1$ random vectors, each with common covariance matrix V, but with different mean vectors $Ex_1 = \mu_1, Ex_2 = \mu_2$. The means μ_1 and μ_2 and the common covariance matrix V are known. Vectors x are to be drawn from a mixture of the two distributions, with equal probability. We seek a rule for *classifying* an x randomly drawn from this mixture of populations X_1 and X_2, by which we mean saying that x belongs either to X_1 or X_2.

A solution of this problem is attained with the *linear discriminant function* $\beta' x - \beta_0$, where β is a $k \times 1$ vector and β_0 is a scalar, and which guides our decision according to the rule

$$\begin{aligned} &\text{if } \beta' x - \beta_0 \geq 0, \text{ then } x \text{ is a member of } X_1; \\ &\text{if } \beta' x - \beta_0 < 0, \text{ then } x \text{ is a member of } X_2. \end{aligned} \tag{1}$$

For a given within population variance $\beta' V \beta$, we select β to separate the two populations as much as possible, where the discrepancy between the two populations is measured by the criterion $\beta'(\mu_1 - \mu_2)$. We choose β to maximize $\beta'(\mu_1 - \mu_2)$, subject to $\beta' V \beta = c$, where $c > 0$ is a constant, by setting

$$\beta = \lambda^{-1} V^{-1}(\mu_1 - \mu_2), \tag{2}$$

[2] See Kendall (1957).

where λ is a Lagrange multiplier on the constraint $\beta' V \beta = c$. By setting the multiplier equal to one we pin down the variance c in the constraint, and so set β_0 (1) according to $\beta_0 = \beta'(\mu_1 + \mu_2)/2$.

When the means (μ_1, μ_2) and covariance matrix V are not known *a priori*, they are estimated by sample means and covariances, where the sample covariance is estimated by pooling observations across the X_1 and X_2 populations. Sample estimates are substituted into (2) to obtain the sample discriminant function. When X_1 and X_2 are each multivariate normal, the discriminant function (1) implements a likelihood ratio test.

The perceptron

Linear discriminant functions can be represented with a single-layer perceptron, the simplest type of neural network.[3] The perceptron models the interaction of k input neurons $x_{it}, i = 1, \ldots, k$, with one output neuron y_t. The neurons are elements $x_i \in X$ and $y \in Y$, where the spaces X, Y can be specified in various ways. For 'Ising neurons,' $X = \{-1, 1\}$; for classification perceptrons we take $Y = \{0, 1\}$. A perceptron is a function

$$y_t = A(w' x_t),$$

where w and x_t are each $(k \times 1)$ vectors, and A is a 'squasher' function, i.e. a monotonically nondecreasing function that maps $\mathbf{R}$ onto $[0, 1]$. Three popular squasher functions are:

1. The Heaviside step function:

$$A(z) = \begin{cases} 1, & \text{for } z \geq 0 \ ; \\ 0, & \text{for } z < 0. \end{cases}$$

2. The 'sigmoid function':

$$A(z) = \frac{1}{1 + e^{-z}}.$$

3. Any cumulative distribution function, e.g. the normal c.d.f.:

$$A(z) = \frac{1}{\sqrt{2\pi}\sigma} \int_{-\infty}^{z} \exp\left(\frac{-x^2}{2\sigma^2}\right) dx.$$

A perceptron first weights input x_{it} by w_i; then weighted inputs are summed across i and squashed to yield output y_t. With the Heaviside squasher function, the neuron is either 'on' ($y_t = 1$) or 'off' ($y_t = 0$). For non-negative inputs, a positive weight w_i from input i to the output neuron is called an 'exciting' connection, and a negative weight is called an 'inhibiting' connection, because such connections make it more or less likely that the output neuron will 'fire.'

[3] See White (1992) and Weisbuch (1990) for extensive treatment of the material of the next several subsections. See Rubinstein (1993) for an economic application of perceptrons.

For a given set of weights w and the Heaviside step function as the squasher, a perceptron represents a particular linear discriminant function or *classifier*. The neuron 'fires' ($y_t = 1$) if and only if $w' x_t \geq 0$. [4]

Given a 'training sample' $\{y_t, x_t\}_{t=1}^T$, where t denotes a particular individual, the perceptron training problem is to choose w to minimize $\sum_{t=1}^T (y_t - A(w' x_t))^2$. This is a nonlinear least squares problem. The literature on perceptrons describes various algorithms of the iterative form

$$w_t = w_{t-1} + \gamma_t \nabla A(w_{t-1}, x_t)(y_t - A(w'_{t-1} x_t)),$$

where $\{\gamma_t\}$ is a nonincreasing sequence of positive numbers, and $\nabla A(w, x)$ is the gradient of S with respect to w. One scheme is to pick γ_t equal to a small positive constant, and repeatedly to run the sample $\{y_t, x_t\}_{t=1}^T$ through the algorithm until convergence occurs. This recursive algorithm has an interpretation as a *stochastic approximation* scheme, a class of learning algorithms described in Ljung, Pflug, and Walk (1992).[5] Implementations of nonlinear least squares found in the econometrics literature also apply directly.

Perceptrons and discriminant analysis

Early work on the perceptron focused on determining classes of objects which could or could not be separated by a perceptron. Results of Minsky and Papert (1969) caused a long period of disenchantment about the utility of the single unit perceptron because it can represent a linear discriminant function, but not a *nonlinear* discriminant function. Nonlinear discriminant functions are needed for many classification problems, including the famous XOR problem studied by Minsky and Papert. Enthusiasm for perceptrons revived in the early 1980s when it was recognized that networks of perceptrons can approximate any nonlinear discriminant function by including 'hidden units.'

Feedforward neural networks with hidden units

By arranging banks of perceptrons into rows and linking elements of successive rows via weighted summation operators, we construct a *feedforward neural network*. Halbert White and various co-workers[6] have shown that feedforward neural networks are well regarded as approximators of nonlinear functions g mapping vectors $x \in X$ into vectors $y \in Y$.

A feedforward neural network with one hidden layer is described by the

[4] With one of the other two squasher functions above, the value of y_t renders a probability $\in (0,1)$ that an individual is classified in one of the sets.

[5] Also see Ljung (1977), Beneveniste, Métivier, and Priouret (1990), and Ljung and Söderström (1983).

[6] See White (1992) and Gallant and White (1988).

two equations

$$\begin{aligned} y_t &= \theta_0 + \sum_{j=1}^{q} \theta_j a_{tj} \\ a_{tj} &= A\left(\sum_{i=1}^{k} w_{ji} x_{it}\right). \end{aligned} \tag{3}$$

The second equation describes the output a_{tj} of hidden unit j, which is simply a perceptron. The first equation generates the output y_t of the network by taking an affine function of the $(q \times 1)$ vector a_t of outputs of the hidden units. The parameters of the network are the weights w_{ji} and θ_j. The network can be represented compactly as

$$y_t = \theta_0 + \sum_{j=1}^{q} \theta_j A(w_j{}' x_t). \tag{4}$$

The literature has addressed two issues about models of this class: *representation* and *estimation*. The representation or approximation literature delineates the class of functions $g : X \rightarrow Y$ that can be arbitrarily well approximated by model (4). Hornik, Stinchcombe, and White (1989) have shown that a very wide class of functions can be approximated by (4). [7] The parameter that controls the proximity of approximation is q, the number of hidden units. For a given q, the best mean-square approximator to a function $g(x)$ is determined by the values of (θ, w) that minimize the squared norm

$$\| g(x) - \theta_0 + \sum_{j=1}^{q} \theta_j A(w_j{}' x) \|^2 . \tag{5}$$

Here $\| \cdot \|^2$ is the L_2 norm. For a given q, the best approximator can be found by versions of Newton's method. The approximation literature assures us that, if we select q large enough, we can find a (θ, w) that make this mean squared error as small as we want.

The *estimation* problem occurs when we are given a sample $\{y_t, x_t\}_{t=1}^{T}$ and a particular model of the form (4), with fixed q, and want to estimate the parameters. This is a version of the nonlinear regression problem. The model to be estimated can be written in the form

$$y_t = g(x_t, \beta) + \epsilon_t,$$

where we have stacked the parameters (θ, w) into the vector β. Estimation can proceed by utilizing one of a variety of algorithms based on the methods

[7] They show that, if enough hidden units are included, then any Borel measurable function from one finite dimensional space to another can be approximated arbitrarily well. Barron (1991) showed that to achieve the same approximation rate a feedforward network uses only linearly many parameters ($O(qn)$), while polynomial, spline, and trigonometric expansions use parameters that grow exponentially ($O(q^n)$).

of stochastic approximation. The 'training' algorithms discussed in the literature all use versions of these iterations. There exist 'on-line' algorithms that have asymptotic properties equivalent to 'off-line' algorithms. However, for small samples one can generally do better by using an 'off-line' algorithm.

Recurrent networks

The second equation of (3) can be modified in a way that lets us model dynamics. For example, we could specify

$$a_{tj} = A(w_j{}'x_t + \gamma_j{}'y_{t-1}),$$

where γ_j is a vector of parameters. In the special case that $w_j = 0$ for all j, this is an autonomous dynamic system, one that can be used to represent the systematic part of a nonlinear vector autoregression. Alternatively, we can specify

$$a_{tj} = A(w_j{}'x_t + \delta_j{}'a_{t-1}),$$

where a_{t-1} is the vector of values of $a_{t-1,j}$, and δ is a vector of additional parameters. This kind of network has been used by Elman (1988), and captures feedback to the hidden units from past values of hidden units.

A recurrent network can easily be set up to build up statistics (e.g., sums and sums of squares) of past inputs to or outputs from a network, thereby providing the flexibility to implement various accounting devices. Accounting devices play important roles in various applications of networks to decision problems and economic models.

Classifier systems

The brain as an accounting system

A good illustration of an accounting system is John Holland's 'classifier system.' This is a dynamic accounting system over pieces of decision rules. It consists of an evolving and more or less comprehensive list of 'if–then' statements, which Holland calls *classifiers* that map *conditions* into *actions*; a set of interpretations of these classifiers to enable them to make decisions through time; an *accounting system* that cumulates payoffs associated with past experiences; and an *auction* for making decisions in 'real time.' The system has a list with a fixed number of classifiers, which compete for survival.

When a message from the environment enters the classifier system, sometimes *several* classifiers have their 'if' parts satisfied, but their 'then' parts might differ, so that the classifiers are offering conflicting advice. An 'auction' resolves the conflict, with classifiers bidding to spend 'balances' accumulated via an accounting process that registers the consequences of past decisions. The accounting system is the vehicle by which the classifier system learns to alter its behavior over time.

Formally, a classifier system consists of the following objects.

1. *Bit strings (classifiers).* A classifier is a bit string of fixed length, written over the trinary alphabet $0, 1, \#$, where $\#$ is interpreted as 'either 0 or 1' or 'I don't care.' The first part of each bit string is interpreted as encoding a 'condition' statement, while the remaining bits encode an 'action' statement. The presence of the $\#$ sign accommodates generalization.
2. *A decoding device.* When a state occurs in the environment, this device identifies which of a fixed collection of classifiers match in their 'condition' parts the condition prevailing in the environment. The device thereby identifies a set of relevant classifiers, from which one is to be selected actually to make a decision at the moment.
3. *An accounting system.* A measure of value called *strength* is assigned to each classifier in the system at each point in time. Strengths for each classifier are updated over time in response to the utilities and costs that flow from the environment when the classifier acts. The accounting system computes cumulated averages of realized utilities net of costs. In sequential settings, the accounting system *taxes* classifiers operating at one stage, and awards the proceeds to the classifier at the immediately preceding stage of the decision tree whose decision moved the system to the position that gave the presently active classifier the opportunity to act. Setting up the accounting system in this way is important to induce decisions whose only rewards are that they facilitate subsequent decisions that will ultimately generate rewards.
4. *An auction system.* The auction system determines which of a set of matched classifiers is granted the right to act in any given situation. Two alternative auction principles are:
   (a) The highest strength rule gets to make the decision.
   (b) The right to make a decision is allocated probabilistically, with the probability of being granted the decision made equal to a rule's relative strength.
5. *A device for introducing new classifiers.* New classifiers are introduced in several situations.
   (a) *Uncovered situations.* The most obvious occurs when the environment produces a condition that matches no existing classifiers. In this situation, a new classifier is generated whose condition statement matches the existing environmental condition, and whose action part is randomly generated.
   (b) *Try something new.* New classifiers are generated and old ones are occasionally extinguished in order to provide room for experimenting with untried actions.
   (c) *Generalize and specialize.* New classifiers are synthesized to *generalize* (replace 0's and 1's with $\#$'s), or to *specialize* (replace $\#$'s with 0's or 1's in existing rules).

Generality versus discrimination

An author of a classifier system controls the list of states or conditions to encode, the scheme for encoding them, and the accounting system that distributes rewards and collects 'taxes.' [8] Thus, some 'hard wiring' goes into the construction of a classifier system, much of it being done with an eye to the particular problem at hand. In some environments, what is not hard-wired is the degree of generalization.

The problem of learning induces an incompletely understood tradeoff between 'general' rules (those with 'conditions' that are coarse and therefore are often met) and 'specific' rules (those with conditions that are fine and therefore less frequently met). An advantage of 'general' rules is that their conditions are frequently encountered, which means that their performance can be assessed frequently. A disadvantage is that they call for the same action for all states that satisfy the condition. In effect, general classifiers give the advice: 'use a piece-wise constant decision rule over the subset of the state space that I cover.' Specific decision rules have the opposite advantages and disadvantages. They potentially permit fine-tuning the action to fit the specific point in the condition space, but they pay for that advantage by requiring longer histories of experience to learn. [9]

An interesting property of classifier systems is that they can be set up in ways that permit the degree of generalization or specificity to emerge adaptively. The presence of the # (or 'I don't care') symbol in the alphabet, together with devices designed either to generalize or specialize, [10] provide this capacity. The literature contains intriguing simulation examples in which different degrees of generalization have emerged in classifier systems, but at the present time little is known about general principles that determine their propensity to generalize.

Economic Applications

We now illustrate how some of these ideas can be combined, varied, and applied to study economic equilibria. As promised, the following ideas will be exploited: the representation of strategies in terms of hyperplanes, implemented in terms of linear functions via perceptrons; the accounting for history by averaging past payoff information; and an attempt to get by with very *general* rules that make minimal distinctions among histories.

Arthur's artificial bandit

Brian Arthur's (1989) application of the Holland classifier system to a two armed bandit problem illustrates both the role of the accounting system in recording past payoff information, and the potential for the classifier system to 'learn.' The ith arm pays a random variable x_{it} drawn independently and identically from a distribution F_i with mean μ_i. Assume that $\mu_1 > \mu_2$. A

[8] In sequential problems, the author must also link classifier sub-systems 'intertemporally' in a ways that permit learning to experience the rewards of patience.

[9] There are similar tradeoffs between parametric and non-parametric estimation strategies in econometrics.

[10] See Marimon, McGrattan, and Sargent (1990).

player must pull one arm for each t with $1 \leq t \leq T$, with his reward being his total payoffs. The player knows neither F_1 nor F_2.

Arthur let the classifier system consist of the two classifiers,

$$\begin{array}{c} \#\ 0 \\ \#\ 1. \end{array}$$

Here the first entry encodes the 'condition' and the second entry encodes the 'action'; # means 'whatever the history of observations,' 1 means pull the first arm, and 0 means pull the second arm. The conditions of both classifiers are met all of the time. The first classifier plays arm 1 all of the time, and the second plays arm 2 all of the time. Let $\tau_i(t), i = 1, 2$, be a clock recording the cumulative number of times that arm i has been pulled as of time t, which equals the cumulative number of times that classifier i has acted. Arthur set up the accounting system

$$S_{i\tau_i} = S_{i\tau_i-1} + (1/\tau_i)(x_{i\tau_i} - S_{i\tau_i-1}),$$

where $S_{i\tau_i}$ is the 'strength' of classifier i after it has been rewarded with its payoff when its clock is at τ_i. It is easy to show that $S_{i\tau_i}$ records the average payoff associated with playing arm i.

Arthur used a random device based on relative strengths to determine which classifier to obey at time t. In particular, at time $t+1$, the system obeys classifier 1 with probability π_{1t+1} given by

$$\pi_{1t+1} = S_{1\tau_1(t)}/(S_{1\tau_1(t)} + S_{2\tau_2(t)}).$$

By using methods of stochastic approximation, Arthur showed that the classifier system eventually 'probability-matches,' that is, plays the arms in fractions-over-time that are proportional to their expected rewards. [11]

Prisoner's Dilemma

In Arthur's model, histories are summarized by the average revenue generated by each arm, and the probability of pulling an arm is determined by a sigmoid function. This section extends to repeated games the idea of using linear summaries past histories. We will focus on pure strategies and so instead of the sigmoid function, we shall use the Heaviside function to assign a particular action conditioned on the linear summary.

To illustrate the main idea, we shall study the infinitely repeated prisoner's dilemma game whose component game is

$$G: \quad \begin{array}{cc} & \begin{array}{cc} C & D \end{array} \\ \begin{array}{c} C \\ D \end{array} & \begin{bmatrix} 3,3 & 0,4 \\ 4,0 & 1,1 \end{bmatrix} \end{array} \tag{6}$$

[11] See Arthur (1991) for discussion of various ways of altering the classifier system to improve its performance.

In period t, player i chooses an action $s_i^t \in \{C, D\}$, and $s^t = (s_1^t, s_2^t)$ is the realized outcome. Let $h^T = (s^1, \ldots, s^T)$ (where $s^t \in \{C, D\}^2$ for $t = 1, \ldots, T$) be a history in period $T+1$, and let H^T be the set of all histories in period $T+1$.

A repeated game strategy f_i is a mapping from a history to an action:

$$f_i : H = \bigcup_{T \geq 0} H^T \rightarrow \{C, D\}$$

where h^o is the null history. Let $\{\sigma^t(f)\}_{t=1}^{\infty}$ be the sequence of outcomes induced by $f = (f_1, f_2)$. Player i's objective function is

$$\liminf_{T \rightarrow \infty} \frac{1}{T} \sum_{t=1}^{T} u_i(s^t).$$

The strategy f_i is complex because it takes a history $h^T \in H^T$ as an input, and because the number of elements in H^T is 2^{2T}, which increases exponentially as the game proceeds. The astonishing eventual size of the argument of f_i makes it implausible that f_i literally describes the strategic plan of a real human being. Yet game theory routinely assumes that a rational player implements f_i.

Models with bounded rationality are motivated by the suspicion that this theory attributes excessive computational power to players. Here we take a particular way of modeling a strategy in a repeated game. In each period, a player observes an outcome $s \in \{C, D\}^2$ and forms a perception $\alpha_i(s)$, where $\alpha_i : S = \{C, D\}^2 \rightarrow I\!R$. A history $h^T = (s^1, \ldots, s^T)$ is summarized by the aggregate $\sum_{t=1}^{T} \alpha_i(s^t)$. The player's decision is determined by a single threshold decision rule: if the aggregate $\sum_{t=1}^{T} \alpha_i(s^t)$ is non-negative, then he takes C, and otherwise, he chooses D.

Because the threshold for decision making is 0, one can consider

$$\sum_{s \in \{C,D\}^2} \alpha(s)\pi(s : h^T), \tag{7}$$

where $\pi(s : h^T)$ is the empirical frequency of s in h^T. This class of strategies has a simple geometric representation. Notice that the threshold of the decision rule is

$$\sum_{s \in \{C,D\}^2} \alpha(s)\pi(s : h^T) = 0, \tag{8}$$

which is a hyperplane in the unit simplex of $I\!R^4$. The player classifies histories into two sets: one above the hyperplane, and the other below the hyperplane, and assigns an action to each half space determined by (8). For this reason, we call it a linear strategy, and say that histories are classified according to a linear classifier.

Each linear strategy can be represented by a single Ising neuron, a perceptron of order 1, consisting of sensory, associative, and decision units. Here is how the perceptron 'plays.' In the first period, the decision unit assigns a particular action s_i^1 based on an 'empty' signal. At the beginning of period 2, the sensory unit observes $s^1 = (s_1^1, s_2^1)$, and emits $\alpha_i(s^1)$. The associative unit receives this signal, and reports to the decision unit 1 if $\alpha_i(s^1) \geq 0$, and 0, otherwise. Depending upon the report from the associative unit, the decision unit chooses an action. Thus, our perceptron of order 1 is defined formally in terms of the triple

$$\varphi_i = \langle B_i, \alpha_i, s_i^1 \rangle.$$

At the beginning of the game, the machine selects an initial action s_i^1. Once the game starts, each history $h^T = (s^1, \ldots, s^T)$ is processed into

$$\sum_{t=1}^{T} \alpha_i(s^t),$$

where α_i is the synaptic weight generated by the sensory unit for each outcome. Then the associative unit generates reports according to the Heaviside function

$$A\left(\sum_{t=1}^{T} \alpha_i(s^t)\right)$$

where $A(z) = 1$ if $z \geq 0$, and otherwise, 0. The decision unit chooses an action according to

$$B_i : \{0, 1\} \to \{C, D\}.$$

We have defined the perceptron in such a way that the input is the entire history h^T. Because the length of the input increases as the game continues, one may argue that we still impose excessive computational capability. But as our discussion of recurrent networks suggested, one can implement the same computing machine recursively by taking the input in each period to be the outcome instead of a history, and by adding a 'storage unit' of the summary statistics computed in the previous round.

One way to 'remember' a previous summary statistic is to introduce the 'threshold adjustment rule' invented by Rosenblatt (1956). Instead of fixing the threshold of the associative unit as 0 in the formal definition, we would construct a sequence $\{\theta^t\}_{t=0}^{\infty}$ of thresholds. Let $\theta^0 = 0$ be the threshold of the associative unit at the beginning of period 1. Once the new signal s^1 is processed in period 1, the threshold becomes $\theta^1 = \alpha_i(s^1)$ at the beginning of period 2. Following $T-1$ periods of play, $\theta^{T-1} = \sum_{t=1}^{T-1} \alpha_i(s^t)$, which is the threshold for the associative unit to process s^T, and after the information s^T is processed, it is changed to $\theta^T = \sum_{t=1}^{T} \alpha_i(s^t)$. It is not difficult to prove that our formal definition and the recursive definition describe identical computing machines.

Since only the sign of $\sum_{t=1}^{T} \alpha_i(s^t)$ influences actions, we can take $\sum_{s \in S} \alpha_i(s)\pi(s : h^T)$, where $\pi(s : h^T)$ is the empirical frequency of s in h^T, as the *state* of the machine.

Representation

Linear strategies in a repeated game presume little computational capability, but evidently form a small subset of all repeated game strategies because they force actions only to depend on the historical empirical frequency of outcomes. A natural question is whether one reduces the set of equilibrium *outcomes* of a game by restricting the feasible strategies to our class of linear strategies. For the prisoner's dilemma, the answer is no.

Theorem 1. *For any individually rational payoff vector $v^* = (v_1^*, v_2^*)$, construct a perceptron φ_i by assigning the synaptic weight α_i of player i as*

$$\alpha_i(s) = u_j(s) - v_j^* \quad i \neq j, \forall s \in \{C, D\}^2, \tag{9}$$

the decision rule so that player i chooses D if and only if $\sum_{s \in S} \alpha_i(s)\pi(s : h^T) > 0$, and the initial action as $s_i^1 = C$. The perceptron thus constructed forms a Nash equilibrium whose long run average payoff vector is v^*.

Notice that

$$\sum_{s \in S} \alpha_i(s)\pi(s : h^T) = \left[\sum_{s \in S} u_j(s)\pi(s : h^T)\right] - v_j^*.$$

In φ_i, player i is monitoring player j's average payoff from past plays. We have constructed the decision rule so that player i punishes player j by playing D whenever player j's average payoff exceeds v_j^*. This implies that player j cannot receive more than v_j^* against player i's linear strategy. On the other hand, whenever player j's average payoff falls below v_j^*, player i cooperates so that player j can improve his own payoff. [12] This suggests that the pair of linear strategies can sustain v^* as the equilibrium payoff vector.

Theorem 1 is a representation theorem: any individually rational payoff vector can be represented as an equilibrium outcome of linear strategies. Thus, although we have restricted computational capability, we have not limited the set of equilibrium outcomes.

When we have many equilibrium outcomes, the selection of 'good' equilibria becomes a central question. A useful way to identify 'good' equilibrium outcomes is to check their stability by seeking robustness against small mistakes. One can imagine several forms of mistakes in the operation of a perceptron, including errors in computing $\sum_{s \in S} \alpha_i(s)\pi(s : h^T)$ or in choosing a wrong action ('trembling hand'). Since any mistakes are captured in the state of the machine, one can incorporate 'robustness against mistakes' by

[12] The actual proof of this second part is rather complex.

requiring that the pair of perceptrons achieve a Nash equilibrium following any pair of states of the machine that can be revealed by some histories. This is in the spirit of requiring subgame perfection since we consider any state induced by all possible histories. An interesting result is that the pair of Nash equilibrium perceptrons constructed in Theorem 1 is stable in this sense.

Corollary 1. *Following any history, the pair of perceptrons constructed in Theorem 1 forms a Nash equilibrium whose long run average payoff vector is* v^*.

More general games

Theorem 1 and Corollary 1 demonstrate the potential benefit of neural networks as a modeling tool. By using the simplest form of neural networks, we recover the folk theorem for the repeated prisoner's dilemma game. However, in other games, some outcomes cannot be sustained by a pair of linear strategies. To show this, we consider another infinitely repeated prisoner's dilemma game where the component game is

$$G' : \quad \begin{array}{cc} & \begin{array}{cc} C & D \end{array} \\ \begin{array}{c} C \\ D \end{array} & \begin{bmatrix} 2,2 & 0,6 \\ 6,0 & 1,1 \end{bmatrix} \end{array} \tag{10}$$

Notice that the cooperation outcome is Pareto dominated by some convex combination of $u(C,D)$ and $u(D,C)$.

Since the security level payoff of each player is 1, $(2,3)$ is an individually rational payoff vector. The folk theorem says that this payoff vector can be sustained by some subgame perfect equilibrium. However, no linear strategies can sustain this payoff vector as a Nash equilibrium. Instead of a formal proof, we will examine whether the perceptron ψ_i whose synaptic weights are determined by

$$\alpha_i(s) = u_j(s) - v_j^* \quad i \neq j \tag{11}$$

can support $v^* = (2,3)$ as a stable Nash equilibrium vector. Fix a history $\tilde{h}^T = (s^1, \ldots, s^T)$ so that

$$\sum_{t=1}^{T} \alpha_i(s^t) > 0$$

for each $i = 1,2$. In this case, following $\tilde{h}^T$, player i chooses C. By

$$\alpha_i(C,C) = u_j(C,C) - v_j^* \leq 0 \quad \forall i = 1,2, \tag{12}$$

the summary statistic $\sum \alpha_i(s^t)$ remains negative as long as both players choose C. As a result, once (C,C) is played, the continuation play is stuck at (C,C) and player 2's long run average payoff becomes $u_2(C,C)$, which is strictly less than $v_2^* = 3$.

To support $(2,3)$ as an equilibrium payoff in (10), two players must coordinate their actions not to play (C,C). That is because once (C,C) is played, both players get stuck with playing (C,C) for the rest of the game. In order to avoid playing (C,C), we need a more sophisticated way of classifying histories.

Notice that the target payoff vector can be generated by alternating (D,D), (C,D) and (D,C). A natural guess is that we can support $(2,3)$ as a Nash equilibrium payoff vector by forcing players to play only these three outcomes. To make players avoid (C,C), we need a more 'sophisticated' perceptron for at least one player, say player 2, which monitors not only the opponent's but also his own average payoff. That is, player 2 chooses D if

$$\sum_{t=1}^{T} \alpha_2(s^t) = T\left(\frac{1}{T}\sum_{t=1}^{T} u_1(s^t) - v_1^*\right) > 0.$$

But if

$$\sum_{t=1}^{T} \alpha_2(s^t) = T\left(\frac{1}{T}\sum_{t=1}^{T} u_1(s^t) - v_1^*\right) \leq 0, \tag{13}$$

player 2 further considers his own average payoff from the same history. If (13) holds and

$$\sum_{t=1}^{T} \alpha_1(s^t) = T\left(\frac{1}{T}\sum_{t=1}^{T} u_2(s^t) - v_2^*\right) \leq 0, \tag{14}$$

then player 2 chooses D. Otherwise, he chooses C. If player 1 uses the perceptron with a single associative unit built upon (9), then he will choose C if and only (14) holds. If player 2 considers (13) and (14) simultaneously, then the two players never play (C,C) following any history. We can show that this pair of perceptrons forms a Nash equilibrium that induces $v^* = (2,3)$ following every history, i.e., a stable Nash equilibrium.

This example shows that even for 2×2 games, some individually rational payoff vectors cannot be sustained by linear strategies. Instead, some outcomes require strategies with two linear classifiers, where one unit monitors the opponent's average payoff, and the other one monitors his own payoff. At this point, one might conjecture that we have to increase the number of linear classifiers as the number of actions in the component game increases. Fortunately, this conjecture is false. It can be shown that we require only strategies with at most *three* linear classifiers to sustain any individually rational payoff vector in *general* two person repeated games. This result is attained in Cho (1994a).

The construction of linear Nash equilibria suggests that it might be necessary for each party to monitor the opponent's payoff following any history. If so, one may wonder whether the same construction of 'simple' strategies applies to games with imperfect monitoring where some players cannot directly observe the action and the payoff of the opponent. In this case, each player must rely on public information to infer the activities of the other players. If the public outcome is completely uninformative about the underlying actions of the players, then no cooperation beyond the one shot Nash equilibrium is possible. A natural question is whether the 'simple' construction carries over to models where the public outcome is sufficiently informative.

Cho (1994a) examines a general two person repeated games with imperfect monitoring and with no discounting. Suppose that the public outcome is informative enough to construct a least squares estimator of the frequency of actions of the opponent. The sufficient condition to construct the least squares estimator is, roughly speaking, that the number of public outcomes be as large as the product of the number of actions available to each player. One can apply the same construction of strategies that have three linear classifiers over the least squares estimators. By exploiting the consistency of the least squares estimators, we can recover the folk theorem.

Capital and Moral Hazard without Discounting

To demonstrate that linear strategies are useful in constructing simple equilibria in models other than the repeated games, we now examine a stochastic growth model with moral hazard. As the previous examples demonstrate, through only a handful of linear classifiers, we can discriminate histories finely enough to sustain any individually rational payoff vector in a repeated game. By computing average payoff of each player, one can precisely figure out when to punish and when to cooperate.

In the presence of imperfect monitoring, however, the linear classifier needs to extract information about hidden actions of the opponent. Because some player cannot compute the average payoff of the opponent, one must be able to construct a linear 'proxy' of the hidden variables as the game continues. The central issue becomes how well the linear classifier can learn about the history of hidden actions.

The existence of capital in the growth model aggravates the problem. As the capital stock changes, the feasible set of strategies of the players also changes. In order to monitor the activities of an agent through a series of public data, one may have to accumulate substantial amounts of data. Unless the linear classifier can learn the hidden activities swiftly and accurately, one may not carry out the punishment and the cooperation available before the capital stock changes.

We shall show that linear strategies *can* overcome these problems. By exploiting the resiliency of linear strategies, we shall construct a linear strategy sequential equilibrium that can sustain any payoff vector in a small neighborhood of the efficient frontier.

A Model

We use an undiscounted version of a stochastic growth model studied by Marcet and Marimon (1992). Consider a production economy where the representative entrepreneur decides consumption (c^t) and investment (i^t) in period $t = 1, 2, \ldots$. Let k^t be the capital stock in period t which produces output according to $f(k^t)$ where f is a neoclassical production function satisfying Inada condition. Let τ^t be the subsidy by the planner. The budget constraint of the entrepreneur in period t is

$$c^t + i^t = f(k^t) + \tau^t.$$

The capital stock evolves according to

$$k^{t+1} = dk^t + i^t$$

where $d \in (0,1)$.

Since f satisfies the Inada condition,

$$\max_{k \in \mathbb{R}_+} f(k) - (1-d)k < \infty$$

and

$$k(d) = \arg\max_{k \in \mathbb{R}_+} f(k) - (1-d)k$$

is well-defined. Let $u(c^t)$ be the utility of the entrepreneur in period t. The entrepreneur's objective function is

$$\liminf_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T} u(c^t).$$

Define

$$\underline{v}_1 = u(f(k(d)) - (1-d)k(d))$$

as the autarchy level payoff. That is, given $k(d)$ as the level of capital, the entrepreneur can maintain $\underline{v}_1$ by investing $(1-d)k(d)$ in every period without any subsidy from the planner.

Let the planner's objective function be

$$\liminf_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T} \left[\lambda u(c^t) - \tau^t\right]$$

where $\lambda \in \mathbb{R}_+$ is the weight assigned to the entrepreneur's utility.

Suppose that the return from investment is stochastic. Given i^t, the new capital stock generated by i^t is $g(i^t)$. We assume that

$$Eg(i^t) = i^t,$$

$$g(0) = 0$$

and if $i > i'$, then $g(i)$ first order stochastically dominates $g(i')$. The evolution of the capital stock is now dictated by

$$k^{t+1} = dk^t + g(i^t)$$

and the planner only observes the capital stock in each period as well as the transfer τ^t.

Since the entrepreneur is risk averse, in any Pareto efficient solution, the planner must bear all the risk by offering τ^t so as to make the consumption level of the entrepreneur independent of the outcome from the investment:

$$\lambda u'(c^t) = 1.$$

Without a proper mechanism to control the entrepreneur, however, the entrepreneur has an incentive to under invest so that he can increase consumption in each period.

Linear Strategies

We will show that the planner can construct a 'linear strategy' that implements 'almost' any payoff $v_1 > \underline{v}_1$ as a sequential equilibrium. The (static) optimal solution $(\hat{k}, \hat{\tau})$ of the planner is characterized by

$$\begin{aligned} \lambda u'(f(\hat{k}) - (1-d)\hat{k} + \hat{\tau}) &= 1 \\ f'(\hat{k}) - (1-d) &= 0. \end{aligned}$$

Choose $k^* < \hat{k}$, and $\tau^* > \hat{\tau}$ so that

$$\begin{aligned} \lambda u'(f(k^*) - (1-d)k^* + \tau^*) &< 1 \\ f'(k^*) - (1-d) &> 0. \end{aligned}$$

Let $i^* = (1-d)k^*$ be the level of investment needed to maintain k^* as a steady state, and

$$c^* = f(k^*) - i^* + \tau^*$$

be steady state consumption. Let

$$v_1^* = u(f(k^*) - i^* + \tau^*)$$

and

$$v_p^* = \lambda u(c^*) - \tau^*.$$

Obviously, (v_1^*, v_p^*) is not efficient. But, by choosing k^* and τ^* in a small neighborhood of $\hat{k}$ and $\hat{\tau}$, we can make the distance from the efficient frontier arbitrarily small.

Recall that

$$u : \mathbb{R}^3 \to \mathbb{R}$$

is a differentiable concave function that maps (k, i, τ) into $u(f(k) - i + \tau)$. Differentiate u at (k^*, i^*, τ^*):

$$u'(c^*)(f'(k^*)(k - k^*) - (i - i^*) + (\tau - \tau^*)).$$

We can write the tangent hyperplane as

$$\mathbf{H}'_1 = \{(k,i,\tau) : f'(k^*)(k-k^*) - (i-i^*) + (\tau - \tau^*) = 0\}.$$

By applying the same operation to the objective function of the planner, we obtain

$$\mathbf{H}'_p = \left\{(k,i,\tau) : f'(k^*)(k-k^*) - (i-i^*) + \left(1 - \frac{1}{\lambda u'(c^*)}\right)(\tau - \tau^*) = 0\right\}.$$

Notice that we have chosen k^* and τ^* so that

$$1 - \frac{1}{\lambda u'(c^*)} < 0.$$

By the concavity of the objective functions, $\mathbf{H}'_1$ and $\mathbf{H}'_p$ serve as inflated estimates for $u(f(k)-i+\tau)$ and $\lambda u(c)-\tau$, respectively: if $v_1^* = u(f(k)-i+\tau)$, then $f'(k^*)(k-k^*)-(i-i^*)+(\tau-\tau^*) < 0$, and if $v_p^* = \lambda u(f(k)-i+\tau)-\tau$, then $f'(k^*)(k-k^*)-(i-i^*)+(1-\frac{1}{\lambda u'(c^*)})(\tau-\tau^*) < 0$.

We cannot use $\mathbf{H}'_1$ and $\mathbf{H}'_p$ as the threshold to assign an action to the planner, because the investment level cannot be observed by the planner. Thus, we need to construct a proxy for i^t. Define

$$\hat{i}^t = (k^{t+1} - k^*) - d(k^t - k^*).$$

Notice that for $t \geq 1$,

$$E\left[\hat{i}^t : i^t\right] = i^t - i^*,$$

since $i^* = (1-d)k^*$. By replacing i^t with $\hat{i}^t$ into $\mathbf{H}'_1$, we have

$$f'(k^*)(k^t - k^*) - (k^{t+1} - k^* - d(k^t - k^*)) + (\tau^t - \tau^*).$$

By adding up with respect to $t \geq 1$, we construct a function

$$G_1^T = (f'(k^*) - (1-d))\sum_{t=1}^{T}(k^t - k^*) - (k^{T+1} - k^1) + \sum_{t=1}^{T}(\tau^t - \tau^*)$$

as a 'linear proxy' for the gross payoff of the entrepreneur above v_1^* so that if $G_1^T \geq 0$, then the planner assigns $\tau^{T+1} = 0$, and if $G_1^T < 0$, then he assigns $\tau^{T+1} = \tau^*$. Similarly, we construct

$$G_p^T = (f'(k^*) - (1-d))\sum_{t=1}^{T}(k^t - k^*) - (k^{T+1} - k^1) + \left(1 - \frac{1}{\lambda u'}\right)\sum_{t=1}^{T}(\tau^t - \tau^*)$$

so that the entrepreneur plays $i^{T+1} = 0$ if $G_p^T > 0$, and plays $i^{T+1} = i^*$, otherwise.

We confine the formal analysis of this model to Appendix A, where we establish:

Theorem 2. *Let φ_1 and φ_p be the pair of strategies for the entrepreneur and the planner constructed as above. Then (φ_1, φ_p) is a sequential equilibrium whose long run expected payoff vector is (v_1^*, v_p^*).*

Proof. See the appendix. ∎

The planner learns the entrepreneur's activity through G_1^T, which can be computed from public information. Since the entrepreneur is risk averse, it appears that the planner could potentially overestimate the entrepreneur's average payoff by using the linear proxy G_1^T. But because the average increment of the capital stock is a consistent estimator of the average investment by the entrepreneur, the planner eventually can accurately estimate the entrepreneur's investment level and average payoff.

Conclusions and Open Questions

The last example shows both the strengths and the weaknesses of perceptrons as a learning tool. Because linear strategies are simple to implement, we can represent a large class of equilibrium outcomes with simple perceptrons. However, in the presence of imperfect monitoring, the linear proxies can overestimate the hidden variables. We had to rely on a strong law of large numbers to avoid the problem of overestimation. As a result, our approach does not directly carry over to models with discounting and moral hazard. Without moral hazard, Cho (1994b) showed how to recover the folk theorem with a neural network with a finite number of associative units. Cho (1994b) used a feedback rule to adjust the state variable in order to obtain subgame perfect equilibrium, in contrast to Cho (1994a), where feedforward neural networks were sufficient. With imperfect monitoring *and* discounting, a more sophisticated feedback rule will be needed. So we are continuing to search for a neural network with finite processors capable of recovering the folk theorem. If successful, this endeavor promises to open efficient ways of computing equilibria.

Throughout this paper, we have regarded bounded rationality as a constraint on computational capabilities. Thus, the main exercise was to construct equilibria sustained by simple neural networks, and to study how much the restriction on strategies reduced the set of equilibrium outcomes. Rubinstein (1986) pioneered a different line of research by modelling bounded rationality as a cost to implementing equilibrium strategies. In the Rubenstein tradition, it is assumed that players are perfectly rational, but constrained to carry out the plan only through computing machines with finite capabilities, and that the cost of computation depends upon some complexity measure of the computing machine. Rubinstein (1986) demonstrated that small complexity costs may collapse the set of equilibria to a very small set.

The approach of Rubinstein (1986) presumes that players have access to finite automata that encode transition and outcome functions. We believe that neural networks are useful tools to study the strategic implications of complexity cost, especially when the feasible computing machines may have infinite states, and the computing machine can adapt itself as the game progresses. We are pursuing two classes of extensions, the first designed to build real-time adaptation into the neural networks (e.g., via adjustment of the 'target payoff' levels); and a second to impute some computational costs to employing more complex networks. This second endeavor promises to extend the notions of Rubenstein (1986) to computing machines with an infinite dimensional state.

Appendix

Analysis of Theorem 6.2

Define

$$W^T = \sum_{t=1}^{T}(k^t - k^*)$$

and

$$\mathcal{T}^T = \sum_{t=1}^{T}(\tau^t - \tau^*).$$

To simplify notation, assume without loss of generality that

$$k^1 = 0.$$

Let

$$\alpha = f'(k^*) - (1-d)$$

and

$$\beta = -\left[1 - \frac{1}{\lambda u'(c^*)}\right].$$

By construction,

$$\alpha, \beta > 0.$$

Define

$$\mathbf{H}_1 = \left\{(W^T, k^{T+1}, \mathcal{T}^T) : \alpha W^T - k^{T+1} + \mathcal{T}^T = 0\right\}$$

and

$$\mathbf{H}_p = \left\{(W^T, k^{T+1}, \mathcal{T}^T) : \alpha W^T - k^{T+1} - \beta \mathcal{T}^T = 0\right\}.$$

Given a hyperplane $\mathbf{H}$, let $\overline{\mathbf{H}}$ represent the open half space above $\mathbf{H}$.

For later reference, let us summarize some useful results. First, $\forall T \geq 1$,

$$\mathcal{T}^T \leq 0$$

which implies that

$$\limsup_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T}\tau^t \leq \tau^*. \tag{15}$$

By assumption, $\forall T \geq 1$, $k^{T+1} \leq \overline{k}$.

Note that $W^{T+1} > W^T$ if and only if $k^{t+1} > k^*$. Define

$$\overline{T} = \inf\{t : d^t \overline{k} \leq k^*\}.$$

Then,

$$W^T \leq \frac{\overline{k}}{\alpha} + \sum_{t=1}^{\overline{T}} d^t \overline{k} \equiv \overline{W} \quad \forall T \geq 1.$$

Thus,

$$\limsup_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} k^t \leq k^*. \tag{16}$$

By the construction of the strategy, $\tau^T = 0$ if and only if the state variable is in $\overline{\mathbf{H}}_1$. Thus, $\mathcal{T}^{T+1} < \mathcal{T}^T$ whenever the state variable is in $\overline{\mathbf{H}}_1$. By exploiting the fact that $\alpha, \beta > 0$, one can show that

$$\mathcal{T}^T \geq -\alpha\overline{W} - \tau^* = \underline{\mathcal{T}}.$$

Combining this result with (15),

$$\lim_{T\to\infty} \frac{1}{T}\mathcal{T}^T = 0.$$

Since $\{\mathcal{T}^T\}$ is bounded from below, one can compute how far W^T is pushed down while $i^t = 0$ is being played so that the capital stock evolves according to $k^{t+1} = dk^t$. Note that $i^{T+1} = 0$ if and only if the state variable is in $\overline{\mathbf{H}}_p$:

$$\alpha W^T - k^{T+1} - \beta\mathcal{T}^T \geq 0.$$

Thus, while $i^{T+1} = 0$ is played,

$$W^T \geq \frac{k^{T+1} - \beta\mathcal{T}^T}{\alpha} \geq \frac{\beta\underline{\mathcal{T}}}{\alpha}. \tag{17}$$

We warn, however, that the right hand side is *not* a lower bound of the random variable W^T, since W^T can be smaller than the right hand side while the state is in $\underline{\mathbf{H}}_p$.

By (16), we know that

$$\limsup_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} k^t \leq k^*.$$

We shall prove that

$$\liminf_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} k^t \geq k^*$$

by following the idea of Fudenberg, Kreps and Maskin (1990).

Define a new random variable

$$z^t = (1-d)W^t + k^{t+1} + \frac{\beta}{\alpha}\underline{\mathcal{I}}.$$

Notice that the last term is the 'lower bound' of W^T while $(W^T, k^{T+1}, \mathcal{I}^T) \in \overline{\mathbf{H}}_p$. Thus, if $z^t \leq 0$, then

$$(W^T, k^{T+1}, \mathcal{I}^T) \in \underline{\mathbf{H}}_p$$

and

$$i^t = i^*. \tag{18}$$

One can easily verify that

$$z^{t+1} - z^t = k^t - k^* + (1-d)g(i^t). \tag{19}$$

Define

$$\mathcal{R}^T = \{t \leq T-1 : z^t \leq 0\}$$

as the collection of periods before T when $z^t \leq 0$, and

$$x^T = \sum_{t\in\mathcal{R}^T} k^t - k^* + (1-d)g(i^t).$$

By (18) and (19),

$$E\left[x^{T+1} : x^T\right] = x^T.$$

Let $\underline{x}$ be the lowest realized value of $k - k^* + (1-d)g(i)$ for $\forall k \geq 0$ and $i \geq 0$. Define

$$X^T = \max\left[x^1, \ldots, x^T\right].$$

Lemma 1.

$$z^T \geq \underline{x} + \left(x^T - X^T\right) \quad \forall T \geq 1. \tag{20}$$

Proof. We prove the lemma by induction. For $T = 1$, the conclusion holds obviously. Suppose that the conclusion holds for T. If $z^T > 0$, then

$$z^{T+1} \geq \underline{x} + z^T \geq \underline{x} \geq \underline{x} + \left(x^T - X^T\right).$$

If $z^T \leq 0$, then

$$\begin{aligned} z^{T+1} =& z^T + \left(x^{T+1} - X^T\right)) \\ =& \left(\underline{x} + x^T - X^T\right) + \left(x^{T+1} - X^T\right)) \\ =& \underline{x} + \left(x^{T+1} - X^T\right) \\ \geq& \underline{x} + \left(x^{T+1} - X^{T+1}\right). \end{aligned}$$

∎

Since $\{x^T\}$ is a martingale,

$$\frac{x^T - X^T}{T} \to 0$$

almost surely. Thus,

$$\liminf_{T\to\infty} \frac{z^T}{T} \geq 0$$

which implies

$$\liminf_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T} k^t \geq k^*.$$

We conclude that

$$\lim_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T} k^t = k^*.$$

Since $i^t = i^*$ or 0, and $\tau^t = \tau^*$ or 0, we can conclude that almost surely,

$$\limsup_{T\to\infty} \frac{1}{T}\#\left\{t \leq T : i^t = 0 \quad \text{or} \quad \tau^t = 0\right\} = 0.$$

This implies that

$$\lim_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T} u(c^t) = \lim_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T} u(f(k^*) - i^* + \tau^*) = v_1^*$$

almost surely. Similarly, the long run average payoff of the planner is exactly v_p^*. This proves that the pair of linear strategies constructed supports (v_1^*, v_p^*).

Next, we have to show that no party has an incentive to deviate from the linear strategy. We only verify that the entrepreneur has no incentive to deviate. Similar logic applies to the planner.

First, since $\{\sum_{t=1}^{T}(k^{t+1} - dk^t) - i^t\}_{T=1}^{\infty}$ is a martingale with bounded increments,

$$\frac{1}{T}\left[\sum_{t=1}^{T}(k^{t+1} - dk^t) - i^t\right] \to 0 \tag{21}$$

almost surely. By (15) and (16), regardless of the entrepreneur's repeated game strategy,

$$\limsup_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T} k^t \leq k^*$$

and

$$\limsup_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T} \tau^t \leq \tau^*.$$

Since $k^* < k(d)$,

$$f\left(\frac{1}{T}\sum_{t=1}^{T}k^t\right) - (1-d)\left(\frac{1}{T}\sum_{t=1}^{T}k^t\right) \leq f(k^*) - (1-d)k^*$$

whenever

$$\frac{1}{T}\sum_{t=1}^{T}k^t \leq k^*.$$

Since $\{k^T\}$ is uniformly bounded from above, $\forall\delta > 0$, $\exists T(\delta) > 0$ so that $\forall T \geq T(\delta)$,

$$\left|(1-d)\left[\frac{1}{T}\sum_{t=1}^{T}k^t\right] - \frac{1}{T}\sum_{t=1}^{T}\left[k^{t+1} - dk^t\right]\right| < \delta. \tag{22}$$

Since u is continuous and $u' > 0$, $\forall\epsilon > 0$, $\exists\delta > 0$ and $T(\delta)$ so that $\forall T \geq T(\delta)$, (22) holds and

$$u\left(f\left(\frac{1}{T}\sum_{t=1}^{T}k^t\right) - \frac{1}{T}\sum_{t=1}^{T}\left[k^{t+1} - dk^t\right] + \frac{1}{T}\sum_{t=1}^{T}\tau^t\right) \leq v_1^* + \epsilon.$$

By (21), $\forall\eta > 0$, $\exists T(\eta) \geq T(\delta)$ so that $\forall T \geq T(\eta)$,

$$Pr\left(\frac{1}{T}\left[\sum_{t=1}^{T}(k^{t+1} - dk^t) - i^t\right] < \eta \quad \forall T \geq T(\eta)\right) \geq 1 - \eta.$$

Thus, since u and f are concave, for a sufficiently small $\eta > 0$,

$$Pr\left(\frac{1}{T}\sum_{t=1}^{T}u\left(f(k^t) - i^t + \tau^t\right) < v_1^* + \epsilon + \eta \quad \forall T \geq T(\eta)\right) \geq 1 - \eta$$

from which we conclude that the entrepreneur's long run expected payoff cannot exceed

$$(1-\eta)(v_1^* + \epsilon + \eta) + \eta M$$

for some positive constant M. Since $\epsilon, \eta > 0$ are arbitrary, we have proved that the pair of linear strategies sustains v^* as the long run expected payoff vector.

References

Anderson, T. W. (1958). *Introduction to Multivariate Statistical Analysis.* John Wiley, New York.

Arthur, W. Brian (1989a). 'The Dynamics of Classifier Competitions'. Mimeo. 7 March.

—— (1991). 'Designing Economic Agents that Act Like Human Agents: A Behavioral Approach to Bounded Rationality'. *American Economic Review, Papers and Proceedings*, 81:353–9.

Atkeson, Andrew (1991). 'International Lending with Moral Hazard and Risk of Repudiation'. *Econometrica*, Vol. 59: 1069–1089.

Barron, A. R. (1991). 'Universal Approximation Bounds for Superpositions of a Sigmoidal Function'. Mimeo. Technical Report no. 58, Department of Statistics, University of Illinois.

Chen, Xiaohong and White, Halbert (1992). 'Asymptotic Properties of Some Projection-based Robbins-Monro Procedures in a Hilbert Space'. Mimeo. Department of Economics, University of California at San Diego, November.

—— —— (1993). 'Convergence of Nonparametric Learning Models'. Mimeo. Department of Economics, University of California at San Diego, February.

Cho, In-Koo (1994a). 'Perceptrons Play Repeated Games with Imperfect Monitoring'. Mimeo. University of Chicago.

—— (1994b). 'Bounded Rationality, Neural Network and Repeated Games with Discounting'. *Economic Theory*, Vol. 4, 935–957.

—— (1993). 'Perceptrons Play Repeated Prisoner's Dilemma'. *Journal of Economic Theory*, forthcoming.

Elman, J. L. (1988). 'Finding Structure in Time'. Mimeo. CRL Report 8801, Center for Research in Language, University of California at San Diego.

Fisher, R. A. (1936). 'The Use of Multiple Measurements in Taxonomic Problems'. *Annals of Eugenics*, 7:179–88.

Fudenberg, Drew, David M. Kreps and Eric Maskin (1990). 'Repeated Games with Long-run and Short-run Players'. *Review of Economic Studies*, 57, 555–573.

Gallant, A. R., and White, H. (1988). 'There Exists a Neural Network That Does Not Make Avoidable Mistakes'. *IEEE Second International Conference on Neural Networks*, SOS Printing, San Diego, pp. 657–64.

Goldberg, D.E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley, Menlo Park, Calif.

Hebb, D.O. (1949). *The Organization of Behavior.* John Wiley, New York.

Hertz, John, Krogh, Anders, and Palmer, Richard (1991). *Introduction to the Theory of Neural Computation.* Addison-Wesley, Redwood City, Calif.

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems.* University of Michigan Press, Ann Arbor, Mich.

—— (1986). 'Escaping Brittleness: The Possibilities of General-Purpose Learning Algorithms Applied to Parallel Rule-Based Systems'. In R. S. Michalski, J. G. Carbonell, and T. M. Mitchell (eds.), *Machine Learning: An Artificial Intelligence Approach, ii.* Morgan Kaufmann, Los Altos, Calif.

Hornik, Kurt, Stinchcombe, Maxell, and White, Halbert (1989). 'Multi-layer Feedforward Networks are Universal Approximators'. Mimeo. Department of Economics, University of California at San Diego, February.

Kendall, M. G. (1957). *A Course in Multivariate Analysis.* Charles Griffin, London.

—— and White, Halbert (1991). 'Strong Convergence of Recursive M-Estimators for Models with Dynamic Latent Variables'. Mimeo. University of Illinois, March.

Kushner, H. J. and Clark, D. S. (1978). *Stochastic Approximation Methods for Constrained and Unconstrained Systems.* Springer-Verlag, New York and Berlin.

Ljung, Lennart (1977). 'Analysis of Recursive Stochastic Algorithms'. *IEEE Transactions on Automatic Control*, 22:551–75.

—— and Söderström, T. (1983). *Theory and Practice of Recursive Identification.* MIT Press, Cambridge, Mass.

—— Pflug, Georg, and Walk, Harro (1992). *Stochastic Approximation and Optimization of Random Systems.* Birkhäuser Verlag, Basel, Boston, and Berlin.

Marcet, Albert and Ramon Marimon (1992). 'Communication, Commitment, and Growth'. *Journal of Economic Theory*, Vol. 58, No. 2, 219–249.

Marimon, Ramon, McGrattan, E., and Sargent, T. (1990). 'Money as a Medium of Exchange in an Economy with Artificially Intelligent Agents'. *Journal of Economic Dynamics and Control*, 14:329–74.

Minsky, M. L., and Papert, S.A. (1969). *Perceptrons*. MIT Press, Cambridge, Mass.

Müller, Berndt and Reinhardt, Joachim (1990). *Neural Networks: An Introduction*. Springer-Verlag, Berlin and Heidelberg.

Radner, Roy (1985). 'Repeated Principal Agent Games with Discounting'. *Econometrica*, 53, 1173–1198.

Rosenblatt, Frank (1956). *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*. Spartan Books, Washington, D.C.

Rubinstein, Ariel (1986). 'Finite Automata Play the Repeated Prisoner's Dilemma'. *Journal of Economic Theory*, 39, 83-96. Rubinstein, Ariel (1993). 'On Price Recognition and Computational Complexity in a Monopolistic Model'. *Journal of Political Economy*, 101, 473–484.

Weisbuch, Gérard (1990). *Complex Systems Dynamics: An Introduction to Automata Networks*. Lecture Note Volume II, Santa Fe Institute Studies in the Sciences of Complexity, Addison-Wesley.

White, Halbert (1992). *Artificial Neural Networks: Approximation and Learning Theory*. Oxford: Basil Blackwell.