Learning unknown additive normal form games

Mykel J. Kochenderfer
Stanford University, Stanford, California

Abstract

In this paper we introduce the class of additive normal form games, which is a subset of normal form games. In additive normal form games, the actions of each agent contribute some amount to the final payoff of all the agents. The contributions of the agents are assumed to be additive. We discuss the necessary and sufficient conditions for a normal form game to be an additive normal form game and show exactly how a normal form game may be converted to our additive representation. We observe that additive games always have either a dominant strategy equilibrium or weakly dominant strategy equilibria, although the equilibria may not always be Pareto optimal. Various learning techniques are applied to unknown repeated additive games, with Q-learning being the most successful.

Introduction

Frequently agents are placed in unfamiliar environments inhabited by other agents. Within these unknown circumstances, an agent wants to learn how to best respond to the other agents. Such a situation may be modelled by a repeated normal form game with a payoff matrix that is unknown to the agents.

Stochastic learning theory has been studied by Bush and Mosteller (1955) and applied by Arthur (1993) to the multi-armed bandit problem. Posh (1997) examined cycling properties of a stochastic learning algorithm for 2×2 normal form games where the agents know neither their own nor their opponents' payoff matrix.

In this paper we will generalize to n agents with a set of actions available to each agent, but we will make the rather strong assumption that the payoff matrix is *additive*. The formal definition of what it means for a normal form game to be additive is reserved for later in the paper, but it essentially means that if an agent takes an action, the action will contribute some amount to the payoff of all the other agents, including her own. After taking an action, the agent becomes aware of her payoff, which is the sum of all the contributions to her payoff by all agents. The details of what is common knowledge among the agents is reserved for later.

Although it is reasonable that the actions taken by agents contribute some amount to each other's payoff, it is not always the case that these contributions are additive. Many important games, such as Matching Pennies, cannot be represented with an additive payoff schema. Making this assumption restricts the sort of games we can describe, but we are able to say much more about the structure of the game and how the agents will learn.

Definitions

It is necessary to begin with a formal definition of what it means for a normal form game to be *additive* with n agents and m actions available to each agent.

Definition 1 *An additive normal form game is a tuple $G = (N, M, A^1, \ldots, A^n)$, where*

- *N is a finite set of n agents*
- *M is a finite set of m actions available to each agent*
- *A^i is an $n \times m$ real-valued matrix, where the kth component of the jth column of A^i, written $(A^i_j)_k$, is the contribution to the payoff to agent k of action j taken by the ith agent*

While the definition specifies that there are exactly m actions available to each agent, this need not always be the case. We assume that there are exactly m actions in order to simplify the representation of the actions as $n \times m$ matrices.

We shall now explicitly define the *payoff vector* of an additive normal form game. The payoff vector is a real-valued vector representing the payoff of all the agents, where the ith component of the vector represents the payoff to agent i.

Definition 2 *Let vector $\vec{s} = (s_1, \ldots, s_n) \in M^n$, called the action vector, represent the actions taken by each agent, with s_i being the action of agent i. The payoff vector in the normal form game $G = (N, M, A^1, \ldots, A^n)$ is given by*

$$\vec{u} = \sum_{i \in N} A^i_{s_i}$$

We have defined a compact representation of an additive normal form game. The number of elements in $A^1, \ldots, A^n$ is $n^2 m$. It is useful to define another matrix, A, which we will call the *sum-matrix*. The sum-matrix has dimension

$$\underbrace{m \times \cdots \times m}_{n \text{ times}} \times n$$

and has nm^n elements. The element $A_{s_1,\ldots,s_n,i}$ is the payoff for agent i when the agents take the action vector $\vec{s} = (s_1, \ldots, s_n)$.

Definition 3 *The sum-matrix A for a given additive normal form game $G = (N, M, A^1, \ldots, A^n)$ is an $m \times \cdots \times m \times n$ real-valued matrix where the $(s_1, \ldots, s_n, i)$ element is given by*

$$A_{s_1,\ldots,s_n,i} = \sum_{i' \in N} (A^{i'}_{s_{i'}})_i$$

To simplify our notation, we will define $A_{\vec{s},i} = A_{s_1,\ldots,s_n,i}$ when $\vec{s} = (s_1, \ldots, s_n)$. Note that a matrix with the dimensions of the sum-matrix may represent *any* arbitrary normal form game with n agents and m actions available to each agent.

In this paper, we will be looking specifically at repeated additive normal form games, so the payoff vector $\vec{u}$ and the action vector $\vec{s}$ is parameterized by t. In our repeated games, we will assume that the matrices $A^1, \ldots, A^n$ remain constant.

Before we start our analysis, we must define explicitly what is known by the agents while playing this game repeatedly. Everything about the structure of the game is common knowledge, except for the payoff matrices $A^1, \ldots, A^n$. After each round, a given agent receives only her total payoff and has no knowledge of what actions the other agents took, what payoffs the other agents received, or what contributed to her total payoff.

Properties

In this section we will discuss some of the interesting properties of additive normal form games. We will discuss when it is possible to use our additive form to analyze general normal form games, and we will look at some equilibrium properties inherent in the structure of additive games and issues of optimality.

Conversion to additive form

As mentioned earlier, not all normal form games may be represented as an additive normal form game. At first glance, it is not obvious why Matching Pennies cannot be represented as an additive normal form game but Prisoner's Dilemma can. It is important to determine exactly when a normal form game can be represented as an additive normal form game, and so we introduce the property of *constant difference* in general normal form games.

Definition 4 *A normal form game represented by matrix A is defined to have the property of constant difference if and only if for all i and action vectors $(s_1, \ldots, s_n)$ and $(s'_1, \ldots, s'_n)$ the following equality holds,*

$$A_{s_1,\ldots,s_i,\ldots,s_n,i} - A_{s_1,\ldots,s'_i,\ldots,s_n,i}$$
$$= A_{s'_1,\ldots,s_i,\ldots,s'_n,i} - A_{s'_1,\ldots,s'_i,\ldots,s'_n,i}$$

Informally, constant difference means that no matter what actions the other agents take, the difference in payoff to an agent when taking two different actions is constant. This definition leads us to the following theorem.

Theorem 1 *A normal form game is an additive normal form game if and only if the payoff matrix for the game has the property of constant difference.*

It is easy to show the forward implication of this theorem, namely that the sum-matrix of an additive normal form game has the property of constant difference. Using the definitions above, we find that the constant difference in payoff for agent i when taking s_i and s'_i, after cancelling most of the terms in the two summations, is simply

$$(A^i_{s_i})_i - (A^i_{s'_i})_i$$

The implication in the other direction may be shown by defining the matrices $A^1, \ldots, A^n$ in terms of an arbitrary normal form game matrix A that has the constant difference property.

Theorem 2 *Any normal form game with a constant difference payoff matrix A may be represented as an additive normal form game $G = (N, M, A^1, \ldots, A^n)$ where*

- *N is the same set of n agents as in the original normal form game*
- *M is the set of m actions available to each agent*
- *$(A^i_j)_i = A_{1,\ldots,j,\ldots,1,i} - A_{1,\ldots,1,i}$, where the ith index in A of the first term on the right hand side is j*
- *$(A^i_j)_{(i+1) \bmod n} = A_{1,\ldots,j,\ldots,1,(i+1) \bmod n}$, where ith index of A is j*
- *$(A^i_j)_k = 0$, for all other values of i, j, and k*

While the proof for the above theorem is rather tedious, it is not difficult to see why the above construction is justified. For each agent, the contribution of the first action to her payoff is 0, which was selected arbitrarily to simplify the mathematics. It is up to the other agents to contribute to the sum specified in the normal form game matrix. One way of ensuring that agent i receives the proper payoff from the other agents is to simply make is so that agent $(i+1) \bmod n$ pays the full amount and the other agents contribute 0. We use the constant difference property for agent i with $A_{1,\ldots,1,i}$ as a reference point to find the amount agent i contributes towards her own payoff for each action.

We will demonstrate this construction with an example. Suppose we have a normal form game matrix A that has the constant difference property. We know, for instance, that the $(2, 3, 3, 2)$ element of this matrix is given by

$$A_{2,3,3,2} = (A^1_2)_2 + (A^2_3)_2 + (A^3_3)_2$$

Using the construction described in our theorem, we know that

$$(A^1_2)_2 + (A^2_3)_2 + (A^3_3)_2 = A_{1,2,1,2} + A_{1,3,1,2} - A_{1,1,1,2}$$

Using the constant difference property, we can also say that

$$A_{1,2,1,2} + A_{1,3,1,2} - A_{1,1,1,2} = A_{2,3,3,2} - A_{2,1,3,2} + A_{1,2,1,2}$$

We know that

$$A_{2,1,3,2} = (A^1_2)_2 + (A^2_1)_2 + (A^3_3)_2 = A_{1,2,1,2}$$

Substituting this back into the previous equation, we find that

$$\begin{aligned} A_{2,3,3,2} &= A_{2,3,3,2} - A_{2,1,3,2} + A_{1,2,1,2} \\ &= A_{2,3,3,2} - A_{1,2,1,2} + A_{1,2,1,2} \\ &= A_{2,3,3,2} \end{aligned}$$

So, we see that if we use the construction described in our theorem to convert to an additive structure, we can still get the value in the original matrix back with the constant difference property.

We further demonstrate the use of our theorem by using our construction to convert Prisoner's Dilemma to additive form. Prisoner's Dilemma is represented by the payoff matrix

$$\begin{bmatrix} (1,1) & (4,0) \\ (0,4) & (3,3) \end{bmatrix}$$

First, we check that the matrix has the property of constant difference, and indeed it does. Now, we apply the described construction to get

$$A^1 = \begin{bmatrix} 0 & -1 \\ 1 & 4 \end{bmatrix}$$

and

$$A^2 = \begin{bmatrix} 1 & 4 \\ 0 & -1 \end{bmatrix}$$

The mapping from normal form games with constant difference is not unique, the construction defined in our theorem is perhaps the simplest.

Equilibrium properties

The standard definition of dominance applied to pure strategies in general normal form games is the following.

Definition 5 *Given a normal form game $G = (N, M, A)$[1] the pure strategy s_i^* dominates the pure strategy s_i for agent $i \in N$ if and only if*

$$A_{s_1,\ldots,s_i^*,\ldots,s_n,i} > A_{s_1,\ldots,s_i,\ldots,s_n,i}$$

A *dominant strategy* for agent i is a pure strategy that dominates all other strategies. We may also speak of strategies that are *weakly dominant* where we replace the $>$ in the definition with $\geq$, which allows there to be multiple weakly dominant strategies. We now apply these definitions to additive normal form games.

Theorem 3 *Additive normal form games have either one dominant strategy or multiple weakly dominant strategies for each agent. The dominant strategy (or weakly dominant strategies) for agent i is given by*

$$\arg\max_{j \in M} (A_j^i)_i$$

The theorem above follows directly from the constant difference property of the sum-matrix.

If we make the assumption that $(A_k^i)_i \neq (A_{k'}^i)_i$ for all i and $k \neq k'$, then each agent will have a dominant strategy. If all agents play their dominant strategy, it is said that they are at a *dominant strategy equilibrium*. For example, in the Prisoner's Dilemma game described earlier the action vector $(1,1)$ is the dominant strategy equilibrium. A dominant strategy equilibrium is also a pure strategy *Nash equilibrium*. In this equilibrium, no agent would receive a higher payoff by unilaterally deviating from her dominant strategy. There may also be mixed-strategy Nash equilibrium, but we will focus on dominant strategy equilibrium.

[1]Note that the definition uses the tuple (N, M, A) to represent a general normal form game, which is essentially equivalent to the representation $(N, A_1, \ldots, A_n, u_1, \ldots, u_n)$ used in other papers.

Optimality

We may look at the outcome of an additive normal form game in terms of optimality. We may apply the general definition of *Pareto optimality* to additive games.

Definition 6 *An action vector $\vec{s}^* \in M^n$ in an additive normal form game $G = (N, M, A^1, \ldots, A^n)$ is Pareto optimal if and only if for all action vectors $\vec{s} \neq \vec{s}^*$ and for all agents $i \in N$, $A_{\vec{s},i} > A_{\vec{s}^*,i}$ implies that there exists an agent $j \in N$ for whom $A_{\vec{s}^*,j} > A_{\vec{s},j}$.*

Informally, an action vector is Pareto optimal if there is no way to make any agent better off without making another agent worse off.

We have already seen an additive normal form game where the dominant strategy equilibrium is not Pareto optimal, namely the Prisoner's Dilemma game. The dominant strategy is $(1,1)$, but the Pareto optimal outcomes are $(2,2)$, $(1,2)$, and $(2,1)$.

In an arbitrary additive game with more agents and more actions, it is likely that the dominant strategy equilibrium is not Pareto optimal because the contributions of the agents are independent of each other. Therefore, Pareto optimality is an important issue to consider when agents are learning unknown additive games.

Learning

Now that we have an understanding of some of the significant properties of additive normal form games, we may discuss ways in which the set of agents may learn these games in a repeated setting. As specified earlier, the agents have no initial knowledge about their actions or their payoffs, just that they have m actions from which to select and that their final payoff depends on the sum of the contributions from the other agents.

There are many issues involved in learning an unknown additive normal form game. At some point, an agent needs to explore her different actions. It would make no sense for an agent to select an action and simply stick with it for eternity. However, after a period of time, it is important that the selection of actions becomes less random so that the agent can focus on playing the action that seems to provide the best reward. Of course, the agent must take into account that the other agents are also learning, and so selecting the same action could potentially provide very different rewards in different rounds.

Urn learning scheme

Posch (1997) uses an urn scheme for learning unknown 2×2 matrix games, which seems to be well suited for n agents learning unknown additive games. The urn scheme assumes that the payoffs are positive integers. Even if it is not the case that the payoffs are restricted to positive integers, the analysis remains the same. The urn initially contains balls that correspond to each action. At every round, the agent selects a ball at random from the urn and takes the corresponding action. The agent then places in the urn the same number of balls of that type as was received as a payoff.

The urn learning rule is particularly appealing in learning unknown additive games because it is self-reinforcing, meaning good actions are played more frequently. Also, if all agents are using this learning process, the selection of actions becomes stable over time.

To see how well the urn scheme works for agents trying to learn an unknown additive game, simulations were run on two agents playing Prisoner's Dilemma. The matrices A^1 and A^2 (created using the construction described earlier) were altered by adding 1 to every element in the matrices so that it would work with the urn scheme.

The first set of Prisoner's Dilemma simulations, of one million iterations each, started with one ball for each action at the first iteration. It turns out that different simulations (with different random number seeds) had the agents' strategies converging to different action vectors. The inconsistency in what the agents learn is undesirable; we want them to learn to play the best action available to them. The inconsistency is due to the fact that the jar starts with one ball for each action. Whatever ball is randomly selected at the first round is given tremendous weight in being selected in the next round. For example, suppose we select a ball at the first round and we receive a payoff of 6. We then replace the ball with 6 balls of the same type. At the next round, we are 6 times more likely to choose the same action again than to explore the unknown action that might provide an even greater payoff.

In the second set of simulations for Prisoner's Dilemma, the urns were initialized with 50 balls of each type. The agents in all of the simulations converged on the unique dominant strategy equilibrium, namely $(1, 1)$. To summarize the results, the number of iterations it took until both agents were playing their dominant strategy 90% of the time varied widely but was generally in the 100,000s, but a few times it was as low as 120,000 in one simulation.

Simulations were also run on two player games with 6 actions available to each player. The matrices were the following,

$$A^1 = \left[\begin{array}{cccccc} 0 & 1 & 0 & 5 & 2 & 6 \\ 1 & 4 & 0 & 3 & 2 & 2 \end{array} \right]$$

and

$$A^2 = \left[\begin{array}{cccccc} 0 & 4 & 3 & 2 & 3 & 4 \\ 1 & 1 & 5 & 1 & 6 & 3 \end{array} \right]$$

We see that the dominant strategy equilibrium is given by the action vector $(6, 5)$. It took the agents on the order of one hundred million iterations to learn to play their dominant strategies 80% of the time.

While the urn learning scheme is easy to visualize and simple to implement, it would be beneficial to have some way to control how quickly agents learn and how much they explore.

Q-learning

Q-learning, which is a type of reinforcement learning, has been applied to general stochastic games with much success (Kaelbling, Littman, & Moore 1996). A simplified version of Q-learning is quite appropriate in the domain of unknown additive normal form games. Q-learning allows you to control both the rate of learning and exploration, which we had no control over in the urn scheme.

The action an agent selects at each round is based on an estimation of the payoff. The estimation of the payoff is given by the function $Q : M \to \mathbb{R}$, which is updated at each round. After selecting action $s \in M$ and receiving payoff r, the agent updates the expected payoff of taking that action according to the following rule,

$$Q(s) \leftarrow Q(s) + \alpha(r - Q(s))$$

where $\alpha \in [0, 1]$ is the learning rate, which may be parameterized by time, t. It is desirable that the learning rate decreases over time.

Much has been written on various methods of selecting an action based on $Q(s)$ (Kaelbling, Littman, & Moore 1996). One selection strategy that is well suited for learning additive games is called *Boltzmann exploration*. Agents select their action s according to the distribution

$$P(s) = \frac{e^{Q(s)/T}}{\sum_{s' \in M} e^{Q(s')/T}}$$

where T is the *temperature* parameter that controls the amount of randomness in the exploration. It is best when using Boltzmann exploration on additive games to reduce the temperature only gradually at first. The problem with the urn simulations was that the rate of exploration attenuated too quickly and the agents converged on actions that were not dominant. After a certain point, however, it is important that the temperature drops to a point where there is very little exploration; once an agent believes she has found her dominant strategy, she wants to play it as much as possible.

Both the learning rate and the exploration rate must be tuned in order to get good learning performance. Simulations of Q-learning with the temperature fixed at a constant value did extremely poorly compared to the urn scheme. Constant learning rates, however, did not do as poorly.

Simulations were run on Prisoner's Dilemma using this Q-learning scheme with Boltzmann exploration with $\alpha(t) = 0.1$ and $T(t) = t^{-1/4}$. Other definitions for $\alpha(t)$ and $T(t)$ may have been better, but tuning these functions remains somewhat of a black art. In most simulations, the agents either predominantly play $(1, 1)$ or $(3, 3)$. The first is the dominant strategy, and the other is the Pareto optimal outcome that provides a higher payoff to both agents than the dominant strategy equilibrium. However, in all simulations the agents eventually converged to the dominant strategy equilibrium. Frequently, the agents appear to have converged on $(2, 2)$ up until around the millionth move when the agents shift to consistently playing $(1, 1)$. Although $(2, 2)$ might be "better" for both agents than $(1, 1)$, it is unstable. Fluctuations from $(2, 2)$ cause the agents to fall into the dominant strategy equilibrium.

As expected, simulations run on the two player game with 6 actions available to each agent required the temperature to decrease much more slowly. The same learning rate was used, but the temperature was set to be $T(t) = t^{-1/8}$. The agents generally found the dominant strategy equilibrium $(6, 5)$ quickly, and when they temporarily settled on $(6, 3)$ they eventually switched to $(6, 5)$ after three million or so iterations.

It is not straightforward how to optimize the general Q-learning procedure so that it takes advantage of the additive structure of the game. All agents know that there is a constant difference between the payoff of their actions, provided that the other agents keep their actions constant. In the early iterations of the repeated game, the agents will not be keeping their actions constant. Suppose an agent measures the difference between the payoffs of various actions when the actions of the other agents seem to be stable. The agent would then select the action where her contribution to her payoff is greatest. Such a strategy is equivalent to switching to whatever action provides the greatest total payoff, which is what Q-learning already does. Hence, the constant difference property seems to make Q-learning even more successful on additive games than arbitrary non-additive games.

Discussion

Analysis of normal form games have, in general, been limited to studying 2×2 matrix games, is mostly due to the complexity of representing the payoffs of more than two players. In this paper, we introduce the idea of additive games where the payoffs may be represented simply by n rectangular matrices with a total of $n^2 m$ elements as opposed to the nm^n elements required to describe a general normal form game.

We showed that for a normal form game to be able to be represented as an additive game, the payoff matrix must have the property of constant difference. This is a rather strong assumption, but many important games have this property. It is conceivable that many real-life situations may be, at the very least, approximated by additive games.

Additive games have the property of having a dominant strategy equilibrium, provided that we assume that no two actions played by an agent contribute the same amount to the agent. Otherwise, there will be multiple weakly dominant strategy equilibria. We expect that intelligent agents will converge to such an equilibrium. We also discussed the issue where the dominant strategy equilibrium is not Pareto optimal, but we found experimentally that agents that use the learning techniques described in this paper always seem to leave their Pareo optimal play in favor of the dominant strategy equilibrium. Higher-order stategies, such as “Tit-for-Tat” may be used in games such as Prisoner’s Dilemma (Axelrod & Hamilton 1981; Axelrod 1984). Further research is needed to determine exactly how such higher-order strategies would work in an n player additive game where the payoffs are initially unknown.

The decomposition of payoffs to contributions from other agents allows us to analyze games much more easily. The simplicity of the representation will lead to an increased ability to model the interaction of many agents with many actions.

Acknowledgements

Thanks to Yoav Shoham and Rob Powers for their ideas and suggestions.

References

Arthur, W. 1993. On designing economic agents that behave like human agents. *Journal of Evolutionary Economics* 3(1):1–22.

Axelrod, R., and Hamilton, W. 1981. The evolution of cooperation. *Science* 221:1390–96.

Axelrod, R. 1984. *The Evolution of Cooperation*. New York: Basic Books, Inc.

Bush, R., and Mosteller, F. 1955. *Stochastic models for learning*. New York: Wiley.

Kaelbling, L. P.; Littman, M. L.; and Moore, A. P. 1996. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research* 4:237–285.

Posch, M. 1997. Cycling in a stochastic learning algorithm for normal form games. *Journal of Evolutionary Economics* 7(2):193–207.