

## Homework #3: Negotiation and e-business

**Due Dates:** Checkpoint: May 1. On May 1, you will need to turn in 1) a snapshot of your code for task 2. This should include *completed* code for the Q-learning algorithm. 2) A 2-3 paragraph description of your business plan. This should describe the e-business you are proposing, how it will earn revenue, and the market it will serve. Folks with insufficient progress will get 5% taken off the final grade.

Project: Wednesday, May 15. This is the project itself. All three tasks must be completed by this date.

# 1 Game Theory - Nash Equilibria

## 1.1 Pure Strategy Nash Equilibria

For each of the following games, a) find the pure strategy Nash equilibria (there may be more than one) and b) prove that they are in fact Nash Equilibria. In some cases, you can eliminate dominated strategies to find the Nash equilibrium; in other cases, you'll need to evaluate the utility of different actions. Be sure to show your work when asked to prove something or find a solution.

### 1.1.1 Game 1

This game should look familiar. What's it called? Find its Nash equilibrium(a) and prove that they are (an) equilibrium/a.

	A	B
A	3,3	0,5
B	5,0	1,1

### 1.1.2 Game 2

Armaments: A fighter plane and its opponent, a bomber, are trying to decide on opposing strategies. The table here lists the probability that the fighter will destroy the bomber, given a choice in strategies. The fighter would like to maximize this probability, and the bomber minimize it. Find the pure strategy Nash equilibrium(a).

		Bomber		
		Full fire, low speed	Partial fire, medium speed	no fire, high speed
Fighter	Guns	0.30	0.25	0.15
	Rockets	0.18	0.14	0.16
	Bombs	0.35	0.22	0.17
	Ramming	0.21	0.16	0.10

### 1.1.3 Game 3

Meeting at the Mall. Harry and Sally have gotten separated at the mall and are trying to figure out where to meet each other. They're in a big hurry to get going, so they want to get to the car as soon as possible, but they need to find each other first. They can either wait at the car (faster) or wait at the food court (slower - they still need to walk to the car). Find the Nash equilibrium(a) and prove that it/they are equilibria/um.

	Wait at car	Wait at food court
Wait at Car	100,100	0,0
Wait at Food Court	0,0	25,25

## 1.2 Mixed Strategy Nash Equilibria

Matching Pennies. Players A and B are playing Matching Pennies. Each player simultaneously picks either "Heads" or "Tails". If both players choose the same thing, player A gets both pennies. If they choose differently, player B gets both pennies.

- Draw the normal form payoff matrix for this game.
- Show that there is no pure strategy Nash equilibrium for this game.
- find a mixed strategy Nash Equilibrium for this game. (that is, the probabilities with which each player should choose Heads or Tails.)

## 1.3 Auction Strategy

Consider the following Vickrey auction.

2 identical widgets are for sale. There are 3 buyers. Buyer A is willing to pay up to \$10 for a widget, buyer B \$8, and buyer C \$6. Each buyer wants at most one widget.

- What should each player bid?
- Which buyers will receive widgets?
- How much does each player pay?
- What if the widget seller imposes a *reserve price* of \$7. (That is, \$7 is the minimum bid)? Now who gets widgets and how much do they pay?

## 2 Learning in Games

Nash equilibria are a very useful solution concept because they tell us how rational players will behave in settings where everyone has complete information. Unfortunately, that's often not the case in the real world. For example, we often don't know the rewards our opponents will receive for a given strategy set.

In certain circumstances, agents without complete information are still able to learn Nash equilibrium strategies through repeated play. This scenario of players in a repeated game using learning to discover what to do has turned out to be a very useful one for studying agent learning strategies. I've placed a few papers studying this approach on the course's "Resources" page.

In this task, you'll write some code to allow two (or more) agents to repeatedly play a game against each other, with the goal of determining whether simple learning strategies do in fact lead to Nash equilibria.

Your code should be able to run from the command line, and should take as input a number of players (2 or more), a file containing the payoff matrix for the game, and a number of iterations for the game to run. For example:

```
game-learner 2 prisoners-dilemma-matrix 1000
```

Your program should then create the appropriate number of agents, each of which will learn what strategy to play (more on that below). The agents will only be able to see their own strategies; they should not know or use the reward each other player gets for its choice of strategy.

The basic pseudocode is as follows:

```
for i = 1 to number_of_iterations
  for j = 1 to number_of_agents
    strategy_set.append(select_strategy(agent-j))
  end
  for j = 1 to number_of_agents
    agentj.payoff = assign_payoff(strategy set)
  end
  for j = 1 to number_of_agents
    agentj.update_strategies
  end
end
```

## 2.1 Learning

As in project 2, we'll use reinforcement learning here. Each agent will learn the payoff associated with each strategy (or action -I'll use the two terms interchangeably here).

In this case, we'll use a slightly more complex (and theoretically appealing) algorithm called *Q-learning*. Q-learning is an effective and widely used algorithm for learning the values of actions, even in the case where those values might change over time.

As in project 2, you'll want to keep the agent's possible actions and their estimated payoffs in a hashtable, often referred to as a Q-table.

So as in project 2, there are two problems to be dealt with: how to choose an action, and how to estimate the value of an action.

### 2.1.1 Choosing an action

In this project, we'll use a slightly more complex method of choosing an action than in project 2. If you remember, in that project, you normalized the scores for each action (category) and used those as probabilities. One problem with that approach is what to do with actions that have either very high or very low scores - they wind up getting oversampled. Another problem is that this doesn't directly map to the case where we want to know the *value* of an action, rather than our probability of taking it.

To avoid that problem, we'll use a method known as *Boltzmann exploration*. In Boltzmann exploration, the probability of picking an action  $a$  is:

$$\frac{e^{Q(a)/\tau}}{\sum_{a' \in A} e^{Q(a')/\tau}} \quad (1)$$

Where  $Q(a)$  is the estimated payoff of an action,  $A$  is the set of all possible actions, and  $\tau$  is called the temperature. When  $\tau$  is large, each action will have approximately the same probability; when  $\tau$  is small, actions will be selected proportionally according to their estimated payoff.

So Boltzmann exploration is not too different from the strategy you implemented in Project 2. The basic differences are: Values are not normalized to probabilities ahead of time (the formula does this),  $Q$  values are exponentiated (which lets us control the likelihood of selecting an action), and we've introduced "temperature", which is a way of controlling the amount of exploration our agent does.

So now we have a parameter in here (temperature) that needs some values. The general strategy is to start with temperature high, so that all strategies get explored, and then gradually 'cool' the temperature so that only high-payoff actions get selected. (The Boltzmann distribution was originally formulated to explain crystallization of cooling materials, hence the use of temperature).

A reasonable starting strategy for temperature is to start at 100 and decrease it by 1 every 100 iterations until you get to 1. This is a very slow cooling schedule, which will hopefully help the agents find a solution. Later in this task, you'll vary the rate of temperature change.

### 2.1.2 Updating action values

The second problem to be solved is how to update the value of an action  $a$ . Let's say that your agent plays action  $a$  and receives a payoff of  $p$ . Our learning rule is:

$$Q(a) = Q(a) + \alpha(p - Q(a)) \quad (2)$$

where  $\alpha$  is our learning rate. Let's think about this for a bit. If our  $Q$  value is exactly right, then it won't change at all, which is what we want. Now, if  $\alpha$  is 1, we'll update our  $Q$  value to be  $p$ . On the other hand, if  $\alpha$  is 0, we'll completely ignore our new evidence. When  $\alpha$  is between 0 and 1, we'll give the

evidence some credit, but we'll also consider the things we learned in the past, which we rolled up in our present  $Q$  value.

Once again, we need to select a value for  $\alpha$ . As a starting value, set  $\alpha$  to be  $\frac{\tau}{100}$ . So as temperature falls, so will  $\alpha$ . You'll also tune this later on in this task.

Notice that each agent is only going to see *its own* payoff when learning. This means that an action might receive different payoffs, depending upon what the agent's opponent does. One of the questions we'll ask is whether that's a problem, or whether this simple algorithm can learn when an agent's opponent is also learning.

## 2.2 The Task Itself

You should code a simulation engine that can create 2 Q-learning agents, in whatever language you choose. As stated above, your program should be able to take as input a game matrix.

For each of the games in problem 1, run your agents on the problem for 10000 iterations. (You will want to do multiple trials, just to make sure your results are accurate.) For each game, answer the following questions:

1. Did the agents converge to the Nash equilibrium solution? If not, why not? What could be changed to help them?
2. If there were multiple equilibria, did the agents coordinate on an equilibrium? If not, why not?
3. In games with mixed strategy equilibria, do the agents learn a mixed strategy? How can you tell?
4. Try varying the rate at which  $\tau$  and  $\alpha$  change. Do faster or slower changes influence the agents' ability to converge to an equilibrium? What about their total payoff over time? Is there a tradeoff between the rate at which agents learn and their cumulative rewards? Use graphs or tables to illustrate your points.

## 2.3 Extra Credit

1. Extend your system to handle 3 or more agents. Run your system on the following game, known as the Santa Fe Bar Problem, with at least 10 agents. How does the problem difficulty (how often the agents find an equilibrium) change as the number of agents increases?

Santa Fe Bar Problem: Each agent wants to decide whether it should go and visit the local bar in town. It likes to go if there are *some* people there, but not too many. In particular, an agent will attend if less than 60% of the other agents attend.

Agents will play the game repeatedly, choosing from their two strategies: go or stay home. If an agent makes the 'right' choice (goes to the bar if less than

60% of the other agents go, or stays home if more than 60% of the agents go to the bar) it gets a payoff of 1. If it makes the wrong choice (goes to the crowded bar, or stays away from the uncrowded bar) it gets a payoff of 0.

The question is: given our Q-learning strategy, will our agents all converge to a solution where 60% of the agents are at the bar on a given night? (The population might change, but the fraction *should* remain at 60%.) After initial learning, how wildly does attendance fluctuate around 60%? Does altering the learnign rate or temperature help? A graph (or multiple graphs) will help you illustrate this point.

## 2.4 Evaluation

I will evaluate this task based on three criteria, weighted equally. 1) Completeness. Have you done all of the things asked for in this task? 2) Correctness. Does your code work properly? Do your experiments seem to make sense? 3) Coverage. How extensive and well-presented are your experiments? Did you do the bare minimum, or did you really 'get at' what's going on in this problem? Are your expreiments cleanly and concisely presented?

## 3 Business Plans

Throughout the course, we've talked about successful and unsuccessful strategies for e-businesses. This task represents your opportunity to put forth your ideas for a successful online business.

This business can take any of a number of forms, including, but not limited to, B2B markets, C2C matchmaking, and B2C sales of retail items. Some possibilities include:

- targeted sales of niche items, like specialty books
- Online consulting services
- Business-to-business market making
- Online registration and usage tracking of content
- A file-sharing community
- etc.

The list is endless. The point of this is for you to be creative. Imagine the sort of business or endeavor you'd start if you had the opportunity. If you're unsure whether something is applicable, ask me! I'll probably say yes.

### 3.1 The Business Plan

Chapter 2 of the Laudon book talks about the eight components of a business model. They are:

- Value proposition: Why will people be interested in what you're offering?
- Revenue Model? How is your company going to generate income? (advertising? Sales? fees? etc)
- Marketplace Who are your expected customers, and how big is this market, in orders of magnitude?
- Competitors. Who will you be competing against?
- Competitive advantage. What skills, knowledge, resources, etc will make your firm stand out?
- Marketing. How do you plan to get the attention of your potential customers?
- Organization. What sort of organization will your business have? (will it be large or small? Will there be separate departments for content creation and programming? Will you contract out payroll and accounting to another firm, or do it yourself?)
- Management team. What are the skills that you'll need from the company leaders? How many will there be?

Prepare a document that discusses each of these points in detail. Feel free to use diagrams and flowcharts to help explain your points. This document should be professionally prepared (i.e. use a word-processing program) and explain what your business will do, why this is a good idea, and how you plan to succeed. Write this document as if I am a potential venture capitalist deciding whether or not to fund your business.

You may want to do some research on current businesses that have used a similar business model. Some resources include:

- Industry Standard: [www.thestandard.com](http://www.thestandard.com) - now defunct, but past issues are still available.
- Business 2.0 - [www.business2.com](http://www.business2.com) - Business-oriented industry news.
- OpenSource - [www.opensource.org](http://www.opensource.org) - Open-source home site. For businesses taking a less traditional approach.
- Red Herring - [www.redherring.com](http://www.redherring.com) - e-commerce industry news.
- - there's many more ...

## 3.2 Evaluation

I'll evaluate this task based on: 1) Completeness - is everything here that's supposed to be here? 2) Plausibility. Am I convinced by your business plan? Would I fund it if I was a zillionaire? 2) Creativity/Energy - Did you just do something cookie-cutter, or did you come up with your own original ideas?