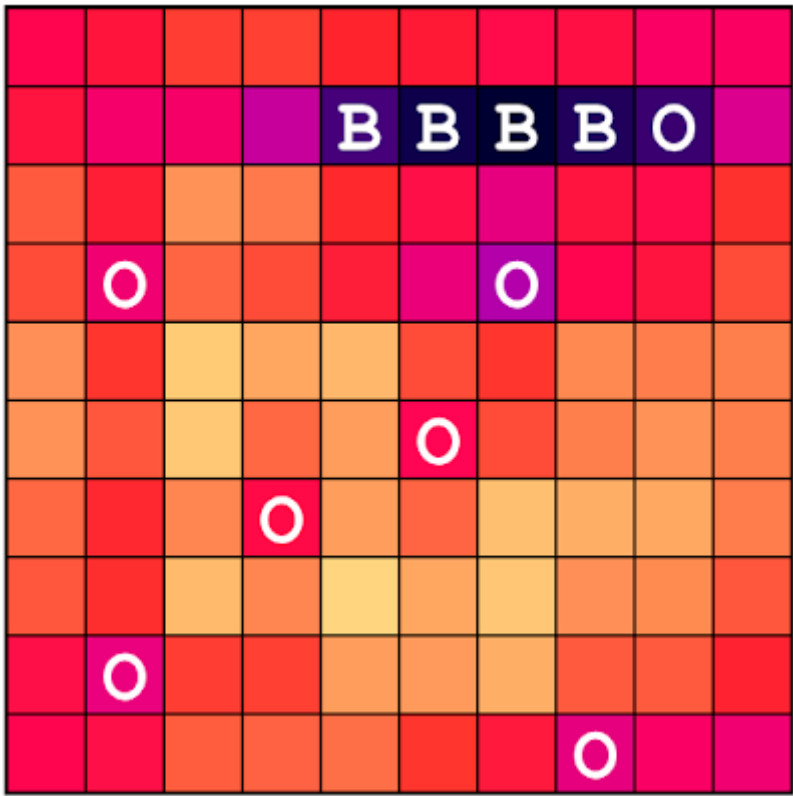


The Linear Theory of Battleship

Alemi — 2011-10-03 00:48 — 23 Comments



Recently I set out to hold a [Battleship](#) programming tournament here among some of the undergraduates. Naturally, I myself wanted to win. So, I got to thinking about the game, and developed what I like to call "the linear theory of battleship". A demonstration of the fruits of my efforts can be found [here](#). Below, my aim is to guide you through how I developed this theory, as an exercise in using physics to solve an interesting unknown problem. This is one of the things I really love about physics, the fact that obtaining an education in physics is essentially an education in reasoning and thinking through complicated problems, along with an honestly short list of tips and tricks that have proven successful for tackling a wide range of problems. So, how do we develop the linear theory of battleship? First we need to quantify what we know, and what we want to know.

The Goal

So, how does one win Battleship? Since the game is about sinking your opponents ships before they can sink yours, it would seem that a good strategy would be to try to maximize your probability of getting a hit every turn. Or, if we knew the probabilities of there being a hit on every square, we could guess each square with that probability, to keep things a little random. So, let's try to represent what we are after. We are after a whole set of numbers

$$P_{i,\alpha}$$

where i ranges from 0 to 99 and denotes a particular square on the board, and α can take the values C,B,S,D,P for carrier, battleship, submarine, destroyer, and patrol boat respectively. This matrix should tell us the probability of there being the given ship on the given square. E.g.

$$P_{53,B}$$

would be the probability of there being a battleship on the 53rd square. If we had such a matrix, we could figure out the probability of there being a hit on every square by summing over all of the ships we have left, i.e.

$$P_i = \sum_{\text{ships left}} P_{i,\alpha}$$

The Background

Alright, we seem to have a goal in mind, now we need to quantify what we have to work with. Minimally, we should try to measure the probabilities for the ships to be on each square given a random board configuration. Let's codify that information in another matrix

$$B_{i,\alpha}$$

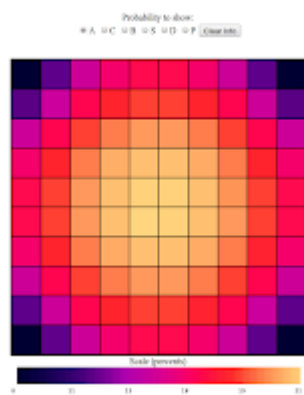
where B stands for 'background', i runs from 0 to 99, and α is either C,B,S,D, or P again, and stands for a ship. This matrix should tell us the probability of a particular ship being on a particular spot on the board assuming our opponent generated a completely random board. This is something we can measure. In fact, I wrote a little code to generate random Battleship boards, and counted where each of the ships appeared. I did this billions of times to get good statistics, and what I ended up with is a little interesting. You can see the results for yourself over at my [results exploration page](#) by changing the radio buttons for the ship you are interested in, but I have some screen caps below. Click on any of them to embiggen.

All

First of all, lets look at the sum of all of the ship probabilities, so that we have the probability of getting a hit on any square for any ship given a random board configuration, or in our new parlance

$$B_i = \sum_{\alpha=C,B,S,D,P} B_{i,\alpha}$$

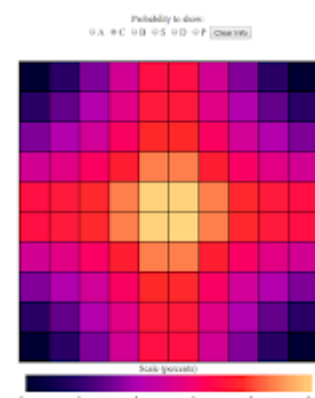
The results:



shouldn't be too surprising. Notice first that we can see that my statistics are fairly good, because our probabilities look more or less smooth, as they ought to be, and show nice left/right up/down symmetry, which it ought to have. But as you'll notice, on the whole there is greater probability to get a hit near the center of the board than near the edges, an especially low probability of getting a hit in the corners. Why is that? Well, there are a lot more ways to lay down a ship such that there is a hit in a center square than there are ways to lay a ship so that it gives a hit in a corner. In fact, for a particular ship there are only two ways to lay it so that it registers a hit in the corner. But, for a particular square in the center, for the Carrier for example there are 5 different ways to lay it horizontally to register a hit, and 5 ways to lay it vertically, or 10 ways total. Neat. We see entropy in action.

Carrier

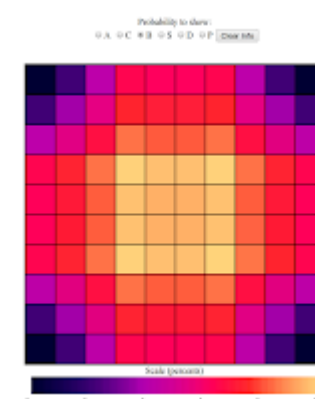
Next let's look just at the Carrier:



Woah. This time the center is very heavily favored versus the edges. This reflects the fact that the Carrier is a large ship, occupying 5 spaces, basically no matter how you lay it, it is going to have a part that lies near the center.

Battleship

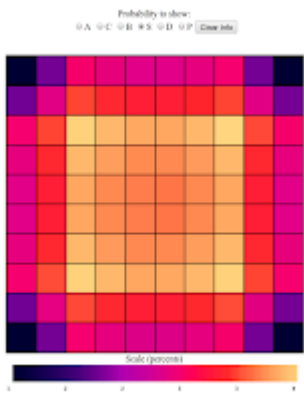
Now for the Battleship:



This is interesting. This time, the most probable squares are not the center ones, but the not quite center ones. Why is that? Well, we saw that for the Carrier, the probability of finding it in the center was very large, and so respectfully, our battleship cannot be in the center as often, as a lot of the time it would collide with the Carrier. Now, this is not because I lay down the Carrier first, my board generation algorithm assigns all of the boards at once, and just weeds out invalid ones, this is a real entropic effect. So here we begin to see some interesting Ship-Ship interactions in our probability distributions. But notice again that on the whole, the battleship should also be found near the center as it is also a large ship.

Sub / Destroyer

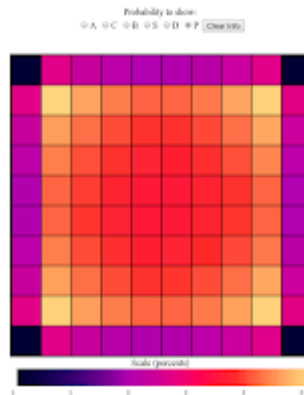
Next let's look at the sub / destroyer. First thing to note is that our plot should be the same for both of these ships as they are both the same length.



Here we see an even more pronounced effect near the center. The Subs and Destroyers are 'pushed' out of the center because the Carriers and Battleships like to be there. This is a sort of entropic repulsion.

Patrol Boat

Finally, let's look at the patrol boat:



The patrol boat is a tiny ship. At only two squares long, it can fit in just about anywhere, and so we see it being strongly affected by the affection the other ships have for the center. Neat stuff. So, we've experimentally measured where we are likely to find all of the battleship ships if we have a completely random board configuration. Already we could use this to make our game play a little more effective, but I think we can do better.

The Info

In fact, as a game of battleship unfolds, we learn a good deal of information about the board. In fact on every turn we get a great deal of information about a particular spot on the board, our guess. Can we incorporate this information into our theory of battleship? Of course we can, but first we need to come up with a good way to represent this information. I suggest we invent another matrix! Let's call this one

$$I_{j,\beta}$$

Where I is for 'information', j goes from 0 to 99 and beta marks the kind of information we have about a square, let's let it take the values M,H,C,B,S,D,P, where M means a miss, H means a hit, but we don't know which ship, and CBSDP mark a particular ship hit, which we would know once we sink a ship. This matrix will be a binary one, where for any particular value of j, the elements will all be 0 or 1, with only one 1 sitting at the spot marking our information about the square, if we have any. That was confusing. What do I mean? Well, let's say its the start of the game and we don't know a darn thing about spot 34 on the board, then I would set

$$I_{34,M} = I_{34,H} = I_{34,C} = I_{34,B} = I_{34,S} = I_{34,D} = I_{34,P} = 0$$

that is, all of the columns are zero because we don't have any information. Now let's say we guess spot 34 and are told we missed, now that row of our matrix would be

$$I_{34,M} = 1 \quad I_{34,H} = I_{34,C} = I_{34,B} = I_{34,S} = I_{34,D} = I_{34,P} = 0$$

so that we put a 1 in the column we know is right, instead, if we were told it was a hit, but don't know which ship it was:

$$I_{34,H} = 1 \quad I_{34,M} = I_{34,C} = I_{34,B} = I_{34,S} = I_{34,D} = I_{34,P} = 0$$

and finally, lets say a few turns later we sink our opponents sub, and we know that spot 34 was one of the spots the sub occupied, we would set:

$$I_{34,S} = 1 \quad I_{34,M} = I_{34,H} = I_{34,C} = I_{34,B} = I_{34,D} = I_{34,P} = 0$$

This may seem like a silly way to codify the information, but I promise it will pay off. As far as my [Battleship Data Explorer](#) goes, you don't have to worry about all this nonsense, instead you can just click on squares to set their information content. Note: shift-clicking will let you cycle through the particular ships, if you just regular click it will let you shuffle between no information, hit, and miss.

The Theory

Alright if we decide to go with my silly way of codifying the information, at this point we have two pieces of data,

$$B_{i,\alpha}$$

our background probability matrix, and

$$I_{j,\beta}$$

our information matrix, where what we want is

$$P_{i,\alpha}$$

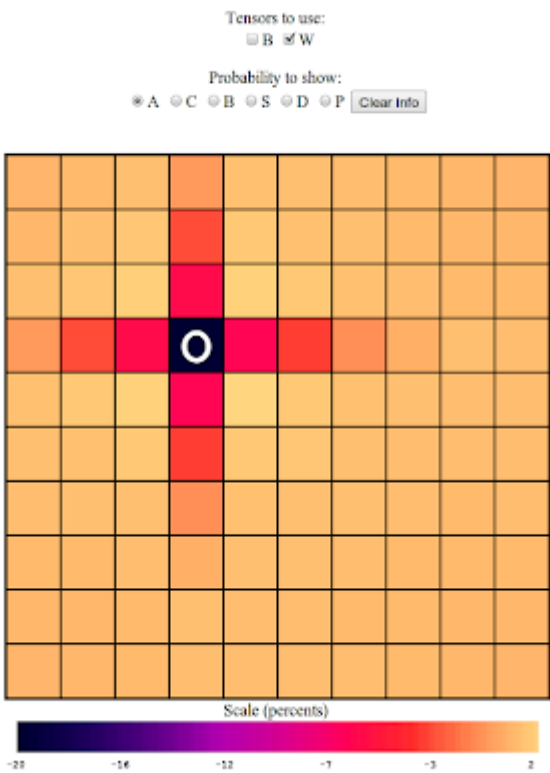
the probability matrix. Here is where the linear part comes in. Why don't we adopt the time honored tradition in science of saying that the relationship between all of these things is just a linear one? In matrix language that means we will choose our theory to be

$$P_{i,\alpha} = B_{i,\alpha} + \sum_{j=[0,...,99], \beta=M,H,C,B,S,D,P} W_{i,\alpha,j,\beta} I_{j,\beta}$$

Whoa! What the heck is that!? Well, that is my linear theory of battleship. What the equation is trying to say is that I will try to predict the probability of a particular ship being in a particular square by (1) noting the background probability of that being true, and (2) adding up all of the information I have, weighting it by the appropriate factor. So here, P is our probability matrix, B is our background info matrix, I is our information matrix, and W is our weight matrix, which is supposed to apply the appropriate weights. That W guy seems like quite the monster. It has four indexes! It does, so let's try to walk through what they all mean. Here:

$$W_{i,\alpha,j,\beta}$$

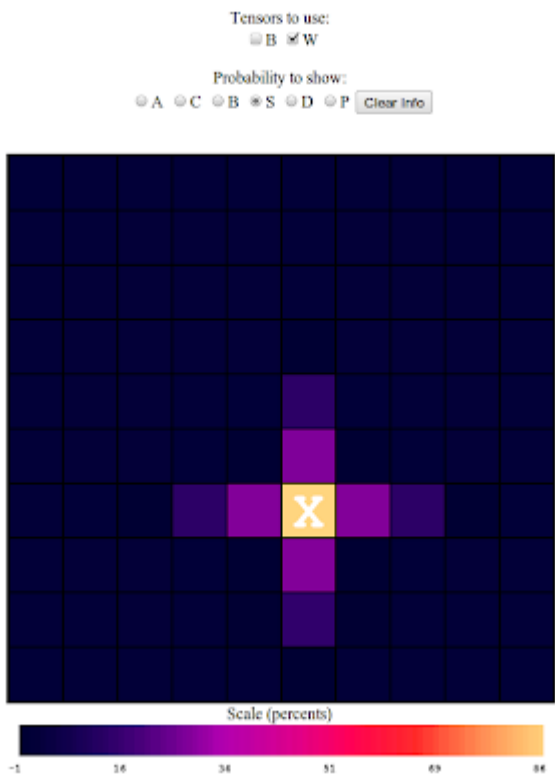
is supposed to tell us: "the extra probability of there being ship alpha at location i, given the fact that we have the situation beta going on at location j" Read that sentence a few times. I'm sorry its confusing, but it is the best way I could come up with explaining W in english. Perhaps a visual would help. Behold the following: (click to embiggen)



That is a picture of

$$W_{i,C,33,M}$$

that is, that is a picture of the extra probabilities for each square (i is all of them), of there being a carrier, (alpha=C) given that we got a miss (beta=M) on square 33, (j=33). You'll notice that the fact that we saw a miss affects some of the squares nearby. In fact, knowing that there was a miss on square 33 means that the probability that the carrier will be found on the adjacent squares is a little lower (notice on the scale that the nearby values are negative), because there are now fewer ways the carrier could be on those squares without it overlapping over into square 33. Let's try another:



That is a picture of

$$W_{i,S,65,H}$$

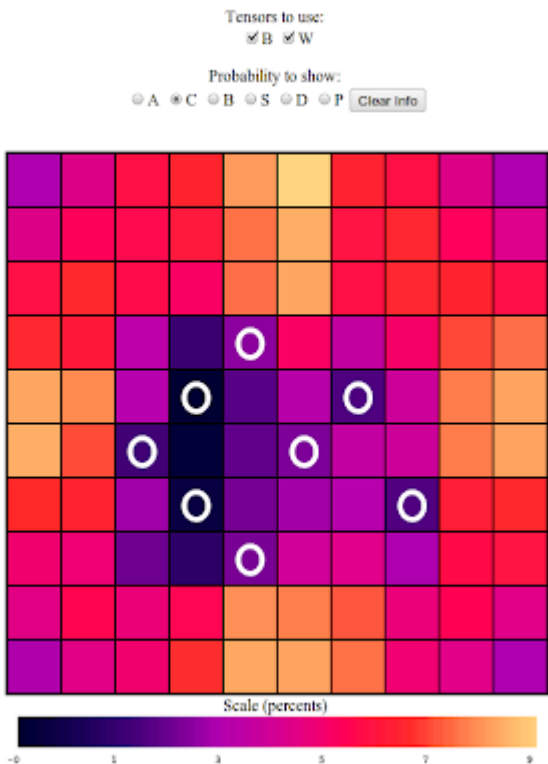
that is, it's showing the extra probability of there being a submarine (alpha=S), at each square (i is all of them, since its a picture with 100 squares), given that we registered a hit (beta=H) on square 65 (j=65). Here you'll notice that since we marked a hit on square 65, it is very likely that we will also get hits on the squares just next to this one, as we could have suspected. In the end, by assuming our theory has this linear form, the benefit we gain is that by doing the same sort of simulations I did to generate the background information, I can back out what the proper values should be for this W matrix. By doing billions and billions of simulations, I can ask, for any particular set of information, I, what the probabilities are P, and solve for W. Given that the problem is linear, this solving step is particularly easy for me to do.

The Results

In the end, this is exactly what I did. I had my computer create billions of different battleship boards, and figure out what the proper values of B and W should be for every square of the matrix. I put all of those results together in a way that I hope is easy to explore up at the [Fancy Battleship Results Page](#), where you are free to explore all of the results yourself. In fact, the way it's set up, you can even use the [Superduper Results Page](#) as a sort of Battleship Cheat Sheet. Have it open while you play a game of battleship, and it will show you the probabilities associated with all of the squares, helping you make your next guess. I've used the page while playing a few games of battleship online, and have had some success, winning 9 of the 10 games I played against the computer player. Of course, this linear theory isn't everything...

Why Linear isn't everything

But at the end of the day, we've made a pretty glaring assumption about the game of battleship, namely that all of the information on the board adds in a linear way. Another way to say that is that in our theory of battleship, we have a principle of superposition. Another way to say that is that in this theory, what you think is happening in a particular square is just the sum of the results from all of the squares, independent of one another. Another way to say that is to show it with another picture. Consider the following:



Here, I've specified a bunch of misses, and am asking for the probability of there being a Carrier on all of the positions of the board. If you look in the center of that cluster of misses, especially in the inner left of the bunch, you'll see that the linear theory tells me that there is a small but finite chance that the Carrier is located on those squares. But if you stop to look at the board a little bit, you'll notice that I've arranged the misses such that there is a large swatch of squares in the center of the cluster where the Carrier is strictly forbidden. There is no way it can fit such that it touches a lot of those

central squares. This is an example of the failure of the linear model. All the linear model knows is that in the spots nearby misses there is a lower probability of the ship being there, but what it doesn't know to do is look at the arrangement of misses and check to see whether there is any possible way the ship can fit. This is a nonlinear effect, involving information at more than one square at a time. It is these kinds of effects that this theory will miss, but as you'll notice, it still does pretty well. Even though it reports a finite positive probability of the Carrier being inside the cluster, the value it reports is a very small one, about 1 percent at most. So the linear theory will have corrections at the 1 percent level or so, but that's pretty good if you ask me.

Summary

And so it is. I've tried to develop a linear theory for the game Battleship, and display the results in a [Handy Dandy Data Explorer](#). I encourage you to play around with the website, use it to win games of Battleship, and in the comments, point out interesting effects, things you think I've missed, or ideas for how to come up with linear theories of other things.

battleship fun linear algebra

[Previous post](#)

[Next post](#)

Comments

ALSO ON THE PHYSICS VIRTUOSI

End of the Earth VII: The Big Freeze

8 years ago • 2 comments

http://tinyurl.com/7rdj996 It is traditional here at The Virtuosi to plot the ...

A Homemade Viscometer I

8 years ago • 3 comments

Stirring a bowl of honey is much more difficult than stirring a bowl of water. ...

Special Brain

7 years ago • 2 comments

A colleague (and friend) of mine (hereafter referred to as Katie Mack the ...

A Curio

8 years ago

Lasers! C In less th NASA's I

23 Comments

The Physics Virtuosi

Disqus' Privacy Policy


1 Login

Recommend

Tweet

Share

Sort by Best




Join the discussion...

LOG IN WITH


OR SIGN UP WITH DISQUS ?

Name

- 


Ryan • 6 years ago

This is cool, but seems irrelevant to the game of Battleship. When ships are placed optimally, they won't be randomly scattered: the player wants to make his opponent indifferent between striking any of the squares. This may entail a somewhat lower probability of locating a ship on the corner squares, but only because a successful corner strike yields an unusually high amount of information about the placement of the ship (and will thus be avoided by players), not because the set of *possible* ship configurations underweights the corner.

1 ^ | v • Reply • Share ›
- 


Alexander Myadzel • 6 years ago

Try this with <http://en.battleship-game.org> :-)

^ | v • Reply • Share ›
- 


USS Johnny Bravo • 6 years ago

The Battleship Data Explorer seems to have a bug. The moment I register a hit (X), the scale at the bottom goes to something like -7 to 415. That means pretty much all the squares look purple, rendering it useless.

^ | v • Reply • Share ›
- 

Alex Alemi Mod ➔ USS Johnny Bravo • 6 years ago

I've added a power law scaling of 0.5 to the colorbar. It helps resolves things a bit at the low end. The differences in the scale are real, in the sense that if you register a hit, it is phenomenally more likely that there are more hits just near by that hit. Once you know that you've hit a particular ship, it helps a lot if you shift-click to register the ship type.

^ | v • Reply • Share ›
- 

WebScrawler • 6 years ago

Ryan makes a good point. It is made worse by the meta game... I used to play my sister all the time. I got her used to me clustering ships together, then I ran a game where every ship except the tiny sub was in one of the four corners... I sunk all of her ships without taking a single strike. She swore I was cheating the whole game -- I remember her being upset all these years later.

^ | v · Reply · Share ›



Anonymous · 9 years ago

Your post is useless without a reference

^ | v · Reply · Share ›



Fernando Tenorio · 9 years ago

This is a very nice article. I need to read it again and play with the results page to see if I fully understand it.

Now, why don't you just calculate the B matrix instead of running billions of simulations. I mean, given a set of ships and square-Information, it's easy to get B.

^ | v · Reply · Share ›



Anonymous · 10 years ago

I think regression analysis might show there is more than a linear relationship

^ | v · Reply · Share ›



Doug S. · 10 years ago

Putting ships on the edge has a weakness.

If you get a "hit" near the center, then one of the four spaces adjacent to the hit must also be a hit. A "hit" is on the edge reveals more information: there are only three spaces adjacent to an edge space, so the probability of getting a hit on a randomly chosen adjacent space is higher. Corners are even worse, because there are only two spaces adjacent to each corner.

By the same principle, it's bad to place your ships so that they touch each other; your opponent might accidentally discover one when trying to sink the other.

^ | v · Reply · Share ›



alexvy86 · 10 years ago

Great read =>. You explained everything so clearly that it felt almost like a regular piece of morning news, with a much more interesting content!

^ | v · Reply · Share ›



Yasha · 10 years ago

If I'm playing Battleship against your algorithm, I'll make sure to put my ships near the edges and corners rather than arranging them randomly.

Though I suppose after enough games against me you could modify your algorithm by basing B on the statistics of my particular board configurations.

^ | v · Reply · Share ›



Jeff Johnson · 10 years ago

I've been playing around with the idea that parliamentary constituencies in the UK violate Duverger's law (single-member-districts are likely to lead to two-party systems; there are clearly more than two viable parties in the UK) because of interactions between constituencies in which different parties would merge if left independent. I didn't realize that this was, in essence, a giant game of battleship. Thanks for the help.

^ | v · Reply · Share ›



Brian Maso · 10 years ago

Kudos to you for this very instructive application of linear probability. Love it!

Game theory holds a very surprising result that applies to your analysis: your pure probability-based strategy only remains maximally effective SO LONG AS YOU DON'T TELL YOUR OPPONENT YOU ARE USING IT! (Sorry for screaming)

Von Neumann developed the "minimax theorem" to describe how purely rational automata can maximize their chances of winning. According to his game theory, a rational player should always choose a game strategy that maximizes his own benefits *after* assuming his opponent will choose a game strategy minimizing the same. Once you make the mistake of informing your opponent of your strategy -- for example by publishing an intelligent and informed blog post on the interwebs about how you intend to automate your Battleship play -- an intelligent opponent will of course use your strategy against you by placing his pieces in the least-likely locations. (You even point out these places -- the corners!)

Publishing your strategy ahead of time is definitely a no-no according to the minimax theory.

(Of course you can take pure rationalism too far in real life games -- Jon von Neumann famously argued that the US should launch an all-out unprovoked nuclear attack on the USSR simply because, according to his minimax theorem, it was the most rational thing to do if you viewed the Cold War as a "game" between nations!)

The moral: pure strategy and math always need to be alloyed with guile and misdirection when applied to pure games of strategy.

^ | v · Reply · Share ›



SVL · 10 years ago



Funny thing is, here in Russia I had a math book for kids which explained how to play this game to win. You need to ensure your smallest ships have max space to hide in, hence you need to place all the big ships together in one part of your board leaving the rest for the small ones, smth like 40/60 for 10*10 board. Your theory would fail against educated Russian kid:)

^ | v · Reply · Share ›



Ced · 10 years ago

I don't have a strong mathematical background but I found this interesting because I arrived at a very similar result when I created my battleship game.

My algorithm would look at the board after each turn and would check to see if each ship would fit in each location. If the ship fit, it would add to the priority of each grid square at that location. At the end, the grid square with the highest priority value was selected. This method does not suffer from the problem you describe at the end of your article. On a side note, I found that playing against such an algorithm was not very fun because it started the game the same way every time. I 'seeded' the board randomly for a few turns before letting the algorithm do its magic. The game is available at <http://frossen.servegame.or...>

^ | v · Reply · Share ›



Alemi · 10 years ago

Anonymous:

The only part that assumes your opponent is selecting ships randomly is the B matrix, the W matrix, and the linear theory in general can be adapted for any background probabilities you like. W will remain unchanged, provided you are still restricting yourself to a linear theory.

But I agree, assuming the background for a human player is the same as for a randomly generated board is certainly not correct, but it is a reasonable assumption to make in the beginning. And, in particular, I was originally thinking in terms of a computer AI tournament, where random board configurations are a lot more meaningful.

^ | v · Reply · Share ›



Anonymous · 10 years ago

Doesn't this all assume that your opponent is selecting battleship positions uniformly? Anyone who plays battleship knows that this is never the case.

^ | v · Reply · Share ›



Alemi · 10 years ago

mikedub:

Glad to hear you enjoyed it.

I'd like to make a nice web interface to allow human players to play against my final AI design, incorporating the higher order logic and the like. If and when I get around to it, I'll be sure to drop a link on this blog, so stay tuned!

^ | v · Reply · Share ›



Alemi · 10 years ago

Christopher Lord:

Granted, humans do not place their ships randomly. I agree that a database of human placed ships would improve things, but such a database is not easily obtained. If you know one I would love a link.

Second, again, granted. One could easily incorporate rules to catch all of the geometrically disallowed positions for a ship to be, but such a rule would be a nonlinear one by design.

I make no claims that this is a winning battleship player, in fact, I realize it is a rather poor one. The point of the post was to show that even a linear theory can be quite effective, generating a 'valid' theory of battleship correct to about the few percent level.

Rest assured, my entrant to the competition did not blindly follow the prescription above, and did include parts that calculated where ships were allowed to be, as well as some simple logic to attempt to learn not only where an opponent liked to guess (in order to avoid placing ships in those locations), but also where the opponent placed their ships (with respect to the background probabilities), in an attempt to learn how non random the player's ship placements were and take advantage of that.

^ | v · Reply · Share ›



mikedub · 10 years ago

Thanks the the entertaining article to go with my morning coffee. I would love to read about your AI strategy!

^ | v · Reply · Share ›



Christopher Lord · 10 years ago

Two comments.

First, you're assuming a random distribution. Humans tend not to follow such distributions. It'd be nice to have a database of human games and see how they distribute ships compared to a uniform distribution.

Regarding your last point, if you can prove that a set of misses disqualify a particular carrier in an area, you can actually encode a probability of zero for a carrier in all of those places.

probability of zero for a carrier in all of those places.

To do that, I suppose you have to break your matrix I up further, such that you can express miss-carrier, hit-battleship, and so on. A shot that misses all boats would set miss-carrier, miss-battleship, and so on. After a turn, you can run the board through an algorithm that removes any probability for boats from places that are no longer possible.

This would even let you refine your probabilities as hits occur, too. i.e., suppose you are working on sinking a ship. At some point, you can prove that your next shot will sink the only possible ship left.

^ | v · Reply · Share ›



Alemi · 10 years ago

Hey Andy,

As for GPD, perhaps, not sure if this alone is worthy, but I know you're looking for people, you could email me.

It's true that there are no 'right' answers, but that's what I love about physics. Even without 'right' answers, you know whether you are 'right' or not by testing your theories.

As for your questions:

- 1) Honestly, no good reason, except that I thought people might already be intimidated by the 4 index W, didn't want to bump it to 6 unnecessarily. Instead opted for separating the indexes conceptually.
- 2) I didn't store it, but noticed why I lost. I was too trusting of the predictions that round, instead of using the predictions to augment my human strategy, I just blindly clicked the highest probability square each time. This can lead to situations where you make what a human player would know to be a poor decision.
- 3) Yup, and I did as far as the tournament went, I had my player place ships randomly to lower the probability of the arrangement, i.e. they tended to cluster on the outside edges of the board.

Perhaps in a followup, I could talk more of the AI strategy I ended up implementing.

^ | v · Reply · Share ›



Andy Rundquist · 10 years ago

This is fantastic, Alemi, and perhaps GPD-worthy (interested?). I love the way you approached this problem. The funny thing is, I bet a lot of people won't believe you when you say this falls into the category of a complicated problem you don't know the solution to (at first). Reading through it, I can see where you made decisions that could have gone a different way, but you were likely satisfied with the results and moved on. I find my students struggle with doing that as they want to make the "right" decision at every step.

3) questions for you:

- 1) Why not make the board matrices 2D, seems more natural to me?
- 2) did you store the game you lost (out of the 10) to see why?
- 3) Can you use your data set to find low entropy states to set up your own board to win more?

^ | v · Reply · Share ›