Linear Automaton Distance: A New Measure of Similarity Between Automata Representations of Repeated Games

Walter Stover
ICES at George Mason University
12/07/2023

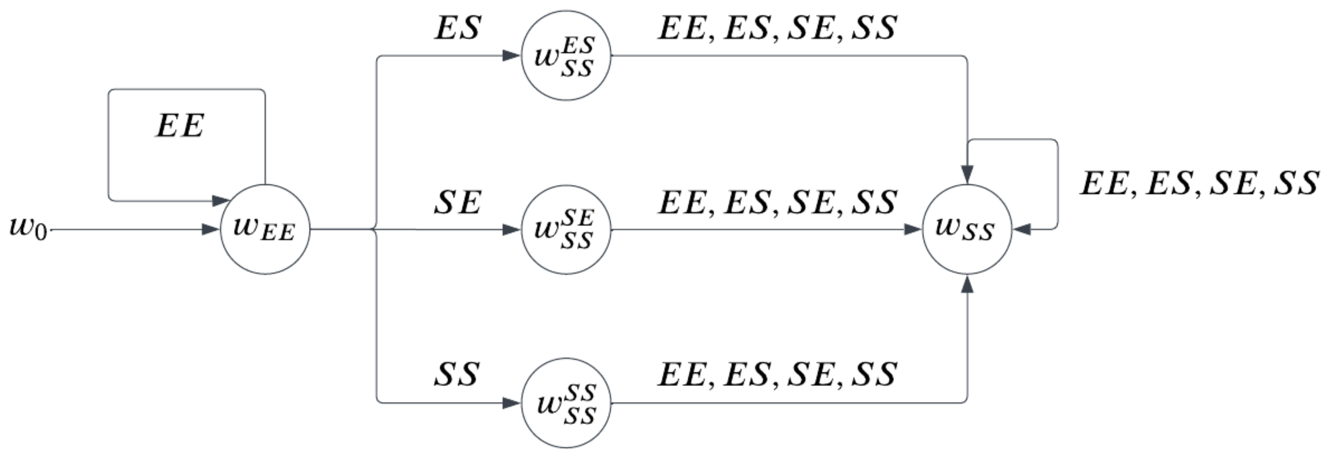
Introduction

Checking for subgame perfection often involves checking a large space of histories, even after simplifying using the one-shot deviation principle (Mailath & Samuelson p. 29). Deterministic finite state automata (DFAs) can be used to group these histories into equivalence classes, where an *automaton* (W, w_0, f, τ) consists of a set of states W, an initial state $w_0 \in W$, an output function $f : W \rightarrow \Pi_i \Delta(A_i)$, and a transition function $\tau : W * A \rightarrow W$.

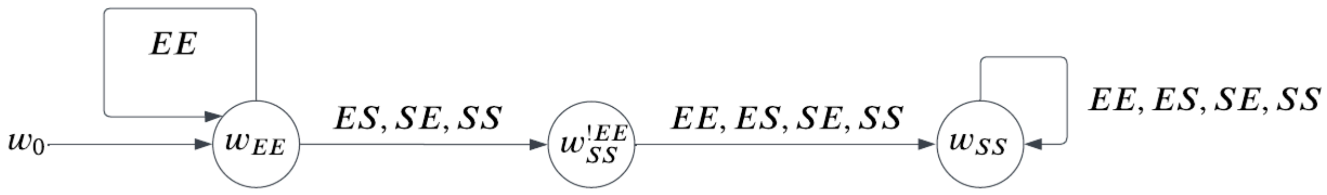
In game theory and elsewhere, DFAs can have multiple representations. There exist techniques for minimizing DFAs to check that they are in their simplest possible form, which is important for maximizing efficiency for any computational tasks involving DFA representations. In this paper, I propose going beyond binary DFA minimization techniques by constructing a similarity metric using normalized compression distance between the compressed forms of DFA languages. This continuous similarity metric would enable not only minimization tasks, but would also support classification and clustering analysis on theoretical game representations as well as real-world behavioral data. This could be a powerful tool for measuring game theoretic behavior in and outside of the lab.

Binary Automata Minimization

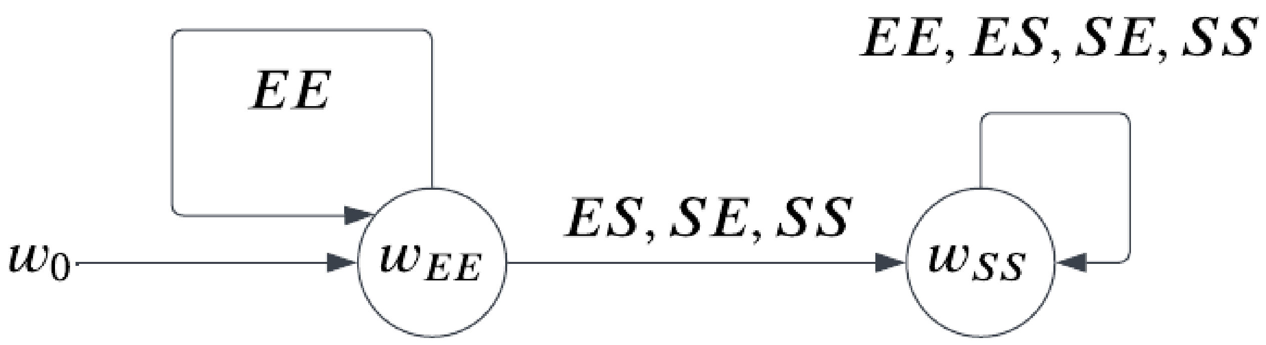
Starting with classical DFA minimization techniques, consider the grim trigger prisoner's dilemma game, where two players decide whether to exert effort or slack in some cooperative task. In this game, the set of states is $W = \{w_{EE}, w_{SS}\, w^{ES}_{SS}, w^{SE}_{SS}. w^{SS}_{SS}\}$, with output function $f(w_{EE}) = EE$ and $f(w_{SS}) = SS$, where E is for effort and S is for slacking. As in Mailath and Samuelson, circles are states, arrows are transitions, labeled by profiles leading to the transitions. The subscript indicates the action profile to be taken at any particular state, while the superscript of the three intermediate states denotes the action profile taken that led to the transition to this state.



Clearly some states in this first representation are redundant. Each of the three intermediate states collapses to the same output state w_{SS}, regardless of the action taken.



We can collapse the three intermediate states into one, $W_{SS}^{!EE}$, $!EE$ indicating any action profile other than EE. Most simply, we can simply look at Mailath & Samuelson's own diagram, which drops this intermediate state altogether:



In this case, visual inspection is enough to determine that Representation 1 contained redundant states that could be simplified, but how do we prove this?

We can formalize this through an equivalence relationship on states p, q:

$$p \approx q \iff \forall x \in \Sigma^*(\Delta(p, x) \in W_f \iff \Delta(q, x) \in W_f)$$

Where:

- p, q are the classes to be checked
- $x \in \Sigma^*$ are the elements x belonging to the input language Σ^* recognized by the automaton,
- $\Delta(p, x)$ is the extension of the transition function τ from symbols (single step) to strings (multiple steps).
- W_f is the set of final states of the automaton.
- $\approx$ is an equivalence relation which is reflexive, symmetric, and transitive.

This equivalence relationship can then be used to design a quotient automaton on equivalence classes $[p] = q|q \approx p$.

Then you just need an algorithm to construct equivalence classes:

1. Write down a table of all pairs $\{p, q\}$, and initialize all as unmarked.
2. Mark a pair $\{p, q\}$ if $p \in W_f$ and $q \notin W_f$ (or vice versa).
3. Repeat until a whole pass is made through the table of pairs without a new marking. For each unmarked p, q, if there is some $a \in \Sigma$ s.t. $\Delta(p, a), \Delta(q, a)$ are marked, then mark $\{p, q\}$.
4. If p, q is not marked, then $p \approx q$.

Leading to:

$\{p, q\}$ is marked $\iff \exists\, x \in \Sigma^*$ s.t. $\Delta(p, x) \in W_f$ and $\Delta(q, x) \notin W_f$ (or vice versa).

Three lemmas and an additional theorem are needed to reach this conclusion, which are omitted in this paper, but are shown in Kaznatcheev (2013).

Continuous Automata Distance Measure

The prior approach is a set of binary operations that construct quotient automata and rely on proving state equivalence classes. However, this approach is useful for pruning and minimizing automata to their simplest representation, which is beneficial for maximizing computational efficiency. It does not directly allow for measuring how similar two automata might be to each other.

However, these same features of automata - the set of possible states, the input alphabet, the transition function, and the set of final states - can also be used to generate a continuous measure of similarity between automaton representations, using the concept of *normalized compression distance*.

Normalized compression distance (NCD) is a simple means of constructing a similarity index between two objects by measuring how difficult it is to transform one object into another, and has been used in a wide variety of clustering and classification applications, including anomaly detection, clustering heterogeneous data, network dynamics, and even music classification. The NCD itself is based on a more fundamental expression,

$$|p| = max\{K(x|y), K(y|x)\}$$

Where p is the shortest computer program that transforms two strings x, y into each other, and $K(x|y)$ is the algorithmic information measure of x given input y. This is the *information distance* between these two strings. To normalize this, the *normalized information distance* (NID) is measured instead:

$$NID(x,y) = \frac{max\{K(x|y),\ \ K(y|x)\}}{max\{K(x), K(y)\}}$$

In practice, the NID is not fully computable. However, Vitanyi and Cilibrasi (2005) rewrites NID to rely on a compressor C to construct the NCD as an approximation of the NID:

$$NCD_z(x,y) = \frac{C(xy) - min\{C(x)C(y)\}}{max\{C(x), C(y)\}}$$

The better the compressor C, the closer the NCD approximates the NID, and NCD takes on a value in a range between 0 and 1, where the higher the NCD score, the more dissimilar two strings are to each other, i.e. the more transformations are needed to turn one string into another.

Applying this procedure to DFAs such as is used in game theory is straightforward.

Given two DFAs, a given DFA can be represented as $A_i = \langle \Sigma_i, W_i, w_{0i}, \tau, W_{fi} \rangle, i \in [1,2]$, where again:

- Σ is the input alphabet,
- W is a finite, non-empty set of states,
- $w_0 \in W$ is the initial state,
- τ is the state-transition relation, and
- W_f is the set of final states.

Further, each DFA A_i will also have a *language*, $L(A_i)$, defined as the set of all strings $a_1, \ldots, a_n \in \Sigma^*$ recognized by A_i, where a string is recognized if:

$$\exists\, w_1, \ldots, w_n \in W : \langle w_{i-1}, a_i, w_i \rangle \in \tau, i = 1, \ldots, n, w_n \in W_f$$

Given a particular game, you can then write an algorithm that will construct the language $L(A_i)$ for this game. Given languages of two games, $L(A_i), i \in [1, 2]$, apply a compressor C to each language such that:

$$x, y = C(L(A_i)), i \in [1, 2]$$

Where x and y are the compressed forms of the languages of both DFAs.

Then we can apply the NCD definition above which will return a continuous [0,1] measurement of how similar the compressed form of these DFAs are to each other.

Example

Consider two variants on the prisoner's dilemma: the first being the grim trigger as in the example above for DFA minimization; the second being a prisoner's dilemma with forgiveness.

Then let A_{gt} be the DFA representation of the grim trigger variation, and A_f be the DFA representation of the forgiveness variation. Then:

- $L(A_{gt}) = \{EE, SS\}$, as only *EE* and *SS* are accepted final states,
- $L(A_f)$ = $\{EE, EEEE, EEES, EESE, EESS, ES, \ldots SSS\}$

Then $C_x, C_y = C(L(A_{gt}), C(L(A_f)))$, the compressed forms of these two DFAs, and C_{xy} is the union set of both compressed forms for combined compression, and the NCD would be:

$$\frac{C_{xy} - min(C_x, C_y)}{max(C_x, C_y)} \approx 0.84$$

Using $zlib$ in Python as the compressor C. This score indicates that the grim trigger and forgiveness variants of prisoner's dilemma are relatively unlike each other - a relatively large number of transformations are needed to get from one language to the other of these DFAs.

Conclusion

In this paper, I applied the concept of normalized compression distance to generate a continuous similarity metric to compare any two DFAs, and specifically applied it to the automaton representations of game theoretic situations such as the prisoner's dilemma.

Such a measure likely has a wide range of applications, including creating a more precise measurement of game theoretic behavior in human subject lab settings; optimizing algorithmic game theoretic treatments; and potentially clustering heterogeneous data on human behavior in large datasets such as financial data, by defining neighborhoods of users through similarity of their behavioral data in DFA form.

References

Cilibrasi, R., and P.M.B. Vitanyi. “Clustering by Compression.” *IEEE Transactions on Information Theory* 51, no. 4 (April 2005): 1523–45. https://doi.org/10.1109/TIT.2005.844059.

Kaznatcheev, Artem. “Minimizing Finite State Automata | Theory, Evolution, and Games Group.” Theory, Evolution, and Games Group, September 18, 2013. https://egtheory.wordpress.com/2013/09/18/minimizing-finite-state-automata/.

Li, Ming, Xin Chen, Xin Li, Bin Ma, and P.M.B. Vitanyi. “The Similarity Metric.” *IEEE Transactions on Information Theory* 50, no. 12 (December 2004): 3250–64. https://doi.org/10.1109/TIT.2004.838101.

Mailath, George J., and Larry Samuelson. *Repeated games and reputations: long-run relationships*. Oxford university press, 2006.

Terwijn, Sebastiaan A., Leen Torenvliet, and Paul MB Vitányi. "Nonapproximability of the normalized information distance." *Journal of Computer and System Sciences* 77, no. 4 (2011): 738-742.