Learning to Play Hearts

Leo Kuvayev
Department of Computer Science
University of Massachusetts
Amherst, MA 01002
kuvayev@cs.umass.edu

The success of neural networks and temporal difference methods in complex tasks such as in (Tesauro 1992) provides the opportunity to apply these methods in other game playing domains. I compared two learning architectures: supervised learning and temporal difference learning for the game of hearts.

Supervised Learning Framework. This version employs a supervised learning algorithm. There are four evaluating networks, one for each suit. If a suit is legal to play, the corresponding network is evaluated on all legal plays in this suit. The card with the highest evaluation is returned as the best candidate in the suit. If several suits could be played then the card with the highest value across all suits is selected. Once the trick is completed, we calculate its value by summing all point-bearing cards in the trick. The neural network is updated with the target value being the trick value: $\varepsilon = Val_{trick} - Q(s_{cur}, a_{cur})$ where $Q(s_{cur}, a_{cur})$ is the output of the network for the current state and action. After we calculated the error, the standard back propagation procedure (Rumelhart, Hinton, & Williams 1986) is applied. It moves the weights of the network in the error minimizing direction.
1

Temporal Difference Framework. The TD framework based on TD(0) algorithm (Sutton 1988) is similar to Supervised Learning framework. One major difference is that the error now consists of two parts: the value of the trick (as in the previous case) and the next prediction. To make a decision we evaluate the network on all legal moves and select the one with the highest value. To update the network we wait until it is our turn again and calculate the value of the best move in the current position. When we update the network the error equals to the reward obtained after the last move plus the value of the current state minus the value of the last move: $\varepsilon = Val_{trick} + max_a Q(s_{old}, a) - Q(s_{old}, a_{old})$ where $Q(s, a)$ is the state-action pair as in (Watkins 1989).

In order to analyze the learning capabilities of both algorithms, I trained my player against two random players and displayed the results on Figure 1.

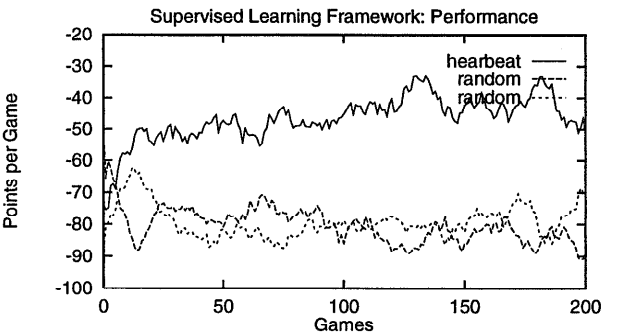

Figure 1: Results for Supervised Learning framework

The player employed random strategy for passing cards. In both architectures the player learned to beat random players after 200 trials. The learning occurred faster in the Supervised Learning case because the networks were four times smaller than in the TD case. The TD predictions are harder to learn as they contain more noise. The average square error reached the lower value in the SL experiment. The additional training did not seem to improve the performance.

Both supervised learning and TD learning architectures produced learning though not highly intelligent players. The first architecture achieved a higher level of play against random players. Neural networks proved to be an effective evaluation function approximator in the noisy and non-stable environment.

References

Rumelhart, D.; Hinton, G.; and Williams, R. 1986. Learning internal representations by error propagation. In *Parallel Distributed Processing*, volume 1. Cambridge, MA: Bradford Books/MIT Press.

Sutton, R. 1988. Learning to predict by the methods of temporal differences. *Machine Learning* 9–44.

Tesauro, G. 1992. Practical issues in temporal difference learning. *Machine Learning* 8:257–277.

Watkins, C. 1989. *Learning with delayed rewards.* Ph.D. Dissertation, Cambridge University.