Neural Networks for opponent prediction in AI

Currently I am working on a project where I am creating an AI player for a game. I am using UCT algorithm and I plan to add a prediction of an opponent's move. For that I would like to use Neural Networks, but I have encountered some problems:

1. There is already some data on which I would like to build a basic neural net for a default player, i.e. a player I haven't seen before. The problem is I would like to update this net so that it adapts for each specific opponent. I have tried searching for an online neural net algorithm, but not very successfully so far. Can you give me some advice how to implement this? How can I update the parameters based on the new data without running the training completely from scratch?
2. The coding is done in Java and I have tried using Weka to process the data I have. However, the only neural net classifier I found in there is Multilayer Perceptron. I am familiar with a Single-layer perceptron and I know that its output are not probabilities (as opposed to loglinear or naive bayes classifiers). Is the Multilayer Perceptron also not generative? If so, how inaccurate it is to use these "scores" it outputs as probabilities? Should I use a different Neural Net algorithm? If so, is there a library for Java available for that?

Thank you very much.

java | artificial-intelligence | machine-learning | neural-network

asked Apr 3 '12 at 23:10

Laky
631 3 13

This has been investigated elsewhere, with good results. Sigmoids are a throwback from the original studies of biological cognition (synapse strength). Synapses are analog models of a non-linear transfer function (learning) that is inherently digital in nature. Use square waves, and don't be confused by those that believe them to be linear. They are the very definition of a non-linearity. stackoverflow.com/questions/10018821/... – Dominic Cerisano Feb 13 at 5:11

add comment

1 Answer

Ok, so I have though about this a bit more and done some more research and I decided to proceed as follows:

1. As I haven't found anything better, I will simply do 1 update step with the new training value using the usual backpropagation algorithm. As far as I know doing 1 update with a new value is the usual approach in some other algorithms, when people convert them to learn online. Finding the correct learning rate will probably be interesting though. As this approach will only do 1 update per move, the backpropagation doesn't have to be optimised very much, hence I will code my own solution and won't use any library.
2. Here I will probably abandon the Weka tool and I will try to use the FANN library instead. As I won't need to use it after calculating the default parameters, it won't matter that it is actually implemented in a different language.

Please let me know if you know about better solutions. Thanks.

answered Apr 4 '12 at 23:09

Laky
631 3 13

add comment

Not the answer you're looking for? Browse other questions tagged java | artificial-intelligence | machine-learning | neural-network **or ask your own question.**