GameNetViz: Infovis Proposal

Neil Newman[*] Jason Hartford[†]

November 2015

1 Introduction

Behavioural game theory aims to predict the behaviour of people as they interact strategically. Researchers in the field typically have two goals: explaining how observed behaviour deviates from perfect rational behaviour, and maximising predictive accuracy.

The recent success of Deep Learning in prediction tasks [5] such as image recognition [4] and natural language processing [9] has shown that for many problems, one can maximise a model's predictive accuracy by optimising flexible models composed of multiple processing layers to observed data. However, adding flexibility to a model makes it both more difficult to optimise and more difficult to interpret.

We hope to show that careful use of visualisation is useful both in understanding how the model is achieving its performance and in diagnosing the cause of optimisation problems.

There are two sources of data that we have to consider in this project: the raw data describing the games and observed behaviour, and the parameters of a machine learning model which attempts to predict behaviour given a description of a game.

A game is described by a *payoff matrix*: $n \times m$ tables where n is the number of decisions available to the first player and m is the number of decisions available to a second player. Each (i, j) cell in the table contains a tuple of integer values that describes the payoff that each player will get if the first player chooses action i and the second player chooses action j. The raw data comes from behavioural economics studies involving real subjects playing a variety of one-shot games with each other. One-shot means that the games entirely consist of each user making a single decision from a given set of decisions, without observing any previous play from their opponent.

There are 164 observations, where each observation consists of a unique payoff matrix and the choices that each player made. Payoff matrices in the dataset range in size from 2×2 to 121×121, with the majority being 3×3 games.

The machine learning model can be viewed as a composition of functions that map the payoff matrices to a vector of probability distributions describing the predicted frequency with which a player will select a particular action. The function composition can be described by a tree where the root nodes are the function inputs, each internal node indicates the application of a function to its parent's output, and the functions are parametrised by the weights of the in-edges. Domain experts are primarily interested in the parameters of the function (the edges in the graph), and the intermediate function outputs. Of particular importance is the notion of *learnt features* which are intermediate predictions of probabilities that players select actions that the model uses to make its final prediction.

The goal of this line of research is not just to build a model that maximises prediction accuracy, but also to understand how the model is making predictions. Previous work [10] has identified a set of hand-crafted, intuitive features based on the structure of payoff matrices that can achieve good prediction accuracy. The primary task is therefore to assess the similarity of the features learned by the model with the intuitive, hand-crafted features, where we can define similarity to be in terms of some distance metric.

[*]newmanne@cs.ubc.ca
[†]jasonhar@cs.ubc.ca

2 Personal Expertise

Applying deep learning to behavioural game theory is the subject of Jason's research. Both of us have each taken courses in game theory, behavioural economics, and machine learning, and both of us are supervised by Kevin Leyton-Brown, who does research in this area, so the domain is familiar to us.

3 Proposed solution

3.1 Feature Similarity

The primary goal of this visualisation is to discover new *learnt features* and investigate their relationship with hand-crafted features. Each feature (either learnt or hand-crafted) is simply a mapping from a matrix of payoffs to a vector of probabilities. This suggests two ways one could compare features: either directly by considering the parameters of the functions, or empirically by considering a derived distance between the outputs of the two features. We favour the latter approach because, as a consequence of the optimisation, the learnt feature functions may use a complex parametrisation to approximate relatively simple functions, and hence may be difficult to compare with hand-crafted features directly.

To evaluate an empirical comparison, we plan to plot a derived distance (e.g. euclidean, KL-divergence) between each feature's output and a hand-crafted feature's output across a set of games. That is, we compute

$$distance(predictionFeature_j(g_i), predictionFeature_k(g_i))$$

for each game g_i for each j, k pair, and then show this data in a plot for each j, k pair. Ideally, if the learned feature exactly matched the handcrafted feature, then every distance value would evaluate to 0. Other distance metrics are also possible: we might be interested in comparing the most likely decision from the probability distribution, rather than the distribution itself.

3.2 Debugging and Tuning

A second task is to allow the user to discover problems with the optimization of their model. Complex machine learning models are often susceptible to poor performance of the optimisation subroutine. Diagnosing the cause of poor optimisation performance is challenging because it involves reasoning about how the optimisation procedure navigates the surface of a high dimensional landscape. By visualising slices of the high dimensional landscape using surface plots and allowing the user to interact with it by adjusting model parameters, we hope to allow the user to discover reasons for optimisation failure.

3.3 Scenario of use

The user begins by using the tool to evaluate the fitted parameters of a particular model. They can see the performance of the model on the training and validation subsets of the data and the fitted parameters of the model.

Associated with each fitted parameter is a spark-line showing the performance of the model in the neighbourhood of the fitted value. By manually adjusting the parameters, they discover parameters which have very little effect on the final performance and those that appear very important. Using this knowledge gained from the visualisation, the user may go back to their machine learning model and make adjustments to improve performance.

Once they are satisfied that they are reliably learning a model that performs well over their target datasets, they use the visualisation tool to explore how the model is achieving its performance. In particular, they look at the learnt features and compare them to the hand-crafted features used in the incumbent state-of-the-art models.

4 Implementation

We are considering using the D^3 [2] framework, as it seems extremely flexible and would be a useful skill to learn even outside of the scope of this course. However, both of us are very familiar with python and

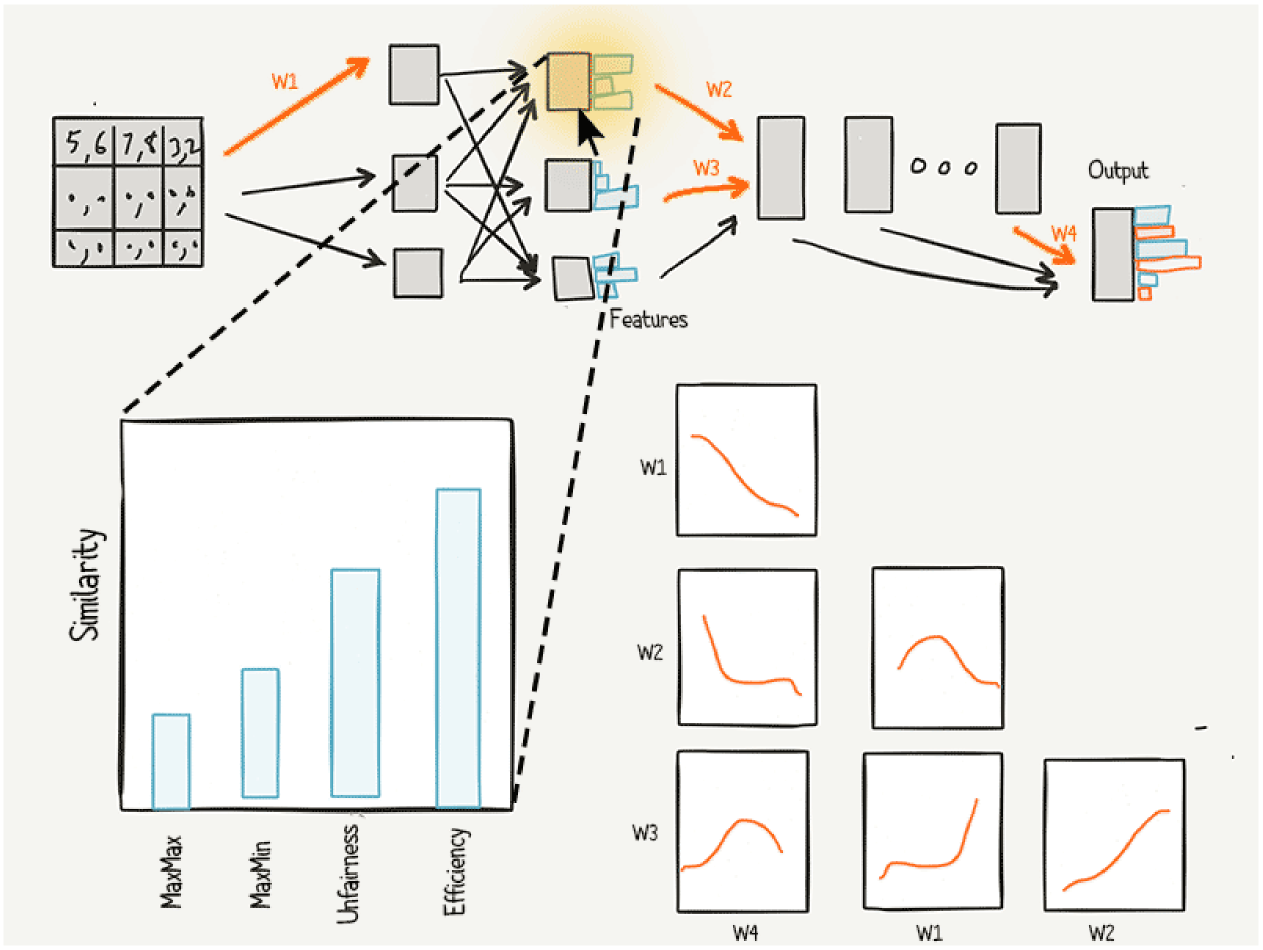

Figure 1: The above sketch of a potential interface shows the machine learning model applied to a particular game illustrated by the gray matrix in the top left. The mouse is currently hovering over a learnt feature which uses linked-highlighting to show the similarity to four hand-crafted features in the large plot on the bottom left. The user has also selected four parameters in the model (w_1, w_2, w_3 & w_4) to examine the objective function in the neighbourhood of their fitted value, shown in the plot matrix on the bottom right of the interface.

matplotlib [3], and have also considered the bokeh [1] framework with a potential lower learning curve.

5 Schedule

- Begin learning frameworks, tutorials (immediately)
- Read related work thoroughly, discover more related work (present - Nov 23rd)
- Data extraction from model and parsing code (present - Nov 13th)
- Feature distance viz initial prototype (week of Nov 16th)
- Update (Nov 23rd)
- Reiterate on feature distance viz (week of Nov 23rd)
- Parameter tuning interactive viz (week of Nov 23rd)
- Presentation (Dec 15th)
- Write final report (Dec 18th)

6 Previous Work

The closest prior work that we are aware of is [6], which develops an interactive system for validation of relatively complex regression models that allows researchers to understand the models' behaviour under different optimisation regimes. They demonstrate the usefulness of allowing engineers to visualise slices of the target function in order to get a feeling for the model's behaviour around a particular point in space. The deep learning literature has also attempted to visualise components of their models in order to better understand how they work. In [7], the authors build functions that simulate the loss surfaces of challenging non-convex optimisation problems. To support this work, they use random projections to visualise slices of their target function.

[11] visualise the layers in a convolutional network that is used for image classification, using a derived data technique that allows them to infer the input pixels that resulted in particular outputs in intermediate layers in the network. Unfortunately, this technique relies on the visual structure encoded in the parameters of the convolutional network for be useful, and thus doesn't generalise to our domain. This limitation motivates our interest in comparing learnt features to hand-crafted features to understand the intermediate representations in our model.

[8] describes *visual parameter space analysis*: systems where the inputs contain parameters that can be tuned that will affect the quality of the output. The paper distinguishes between different types of input: *control parameters*, which are tuned, and *environmental parameters* which come from measured data. The paper also codifies analysis tasks: using this paper's terminology, we have an *optimization* task, since we are interested in finding the best parameters given a loss function, as well as a *sensitivity* task, since we are interested in studying the impact on outputs of varying the model's input parameters. However, unlike this paper, our main task cannot be achieved by viewing the model as a black box, since we want to understand what it is that the black box is doing.

References

[1] Bokeh Development Team. *Bokeh: Python library for interactive visualization*, 2014.

[2] Michael Bostock, Vadim Ogievetsky, and Jeffrey Heer. D^3 data-driven documents. *Visualization and Computer Graphics, IEEE Transactions on*, 17(12):2301–2309, 2011.

[3] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing In Science & Engineering*, 9(3):90–95, 2007.

[4] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, page 2012.

[5] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 05 2015.

[6] Harald Piringer, Wolfgang Berger, and Jürgen Krasser. Hypermoval: Interactive visual validation of regression models for real-time simulation. In *Computer Graphics Forum*, volume 29, pages 983–992. Wiley Online Library, 2010.

[7] Tom Schaul, Ioannis Antonoglou, and David Silver. Unit tests for stochastic optimization. *CoRR*, abs/1312.6055, 2013.

[8] Michael Sedlmair, Christoph Heinzl, Stefan Bruckner, Harald Piringer, and Torsten Moller. Visual parameter space analysis: A conceptual framework. *Visualization and Computer Graphics, IEEE Transactions on*, 20(12):2161–2170, 2014.

[9] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 3104–3112, 2014.

[10] James R Wright and Kevin Leyton-Brown. Level-0 meta-models for predicting human behavior in games. In *Proceedings of the Fifteenth ACM Conference on Economics and Computation*, pages 857–874, 2014.

[11] Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, volume 8689 of *Lecture Notes in Computer Science*, pages 818–833. Springer International Publishing, 2014.