Machine Learning for Vehicle Choice

The goal of the machine learning for vehicle choice project is to demonstrate several points:

1. The KTAB methodology is not limited to processing expert opinion. It can be used with hard data such as economic data, “big data” on vehicle choice, and so on.
2. Generic neural networks (NN) can be improved by adding domain-specific structures. For processing static images, performance can be improved by adding a custom layer to perform convolution. (The eyes of most animals have a hard-wired convolution.) For processing numeric time series, performance can be improved by adding a custom recurrent layer to remember previous values and track the implied “state” of the underlying process. We wish to test the hypothesis that adding a custom layer to model collective decision making can improve performance on data generated by collective choices, compared to a generic network..
   1. We do not hypothesize that a NN with a custom layer can produce performance better than that achieved by more standard, non-NN methods of analyzing consumer choice. Specifically, we do not attempt to surpass or even match the performance which Rubal Dua can achieve with his methods.

I setup several Python scripts in the ktab-priv/survey directory. It might make sense to move them into the tensorflow-sandbox repository. It's currently labeled as "generic scripts", but combining them would reduce the number of repos.

The script names indicate their function. They all assume Python 3, as that's what David Pugh, Andrew Howe, and I agreed on.

* [survey-synth.py](https://github.com/KAPSARC/ktab-priv/blob/master/survey/survey-synth.py): this generates a CSV file of synthetic survey-ish data. You can somewhat control the structure of the data by editing the script. Each row is derived from a multi-dimensional seed vector. Lower dimensions give simpler data, which is easier to learn accurately. The actual data in each row is either a floating point number or a binary 0/1 value.
  + The floating point numbers represent survey responses like income, number of family members, and other things represented by a number.
  + The binary values represent the "one hot" encoding of choice from a set, like the county of residence, previous model of car, new car chosen, etc. I happened to choose two different binary choice responses for input and (of course) one for choice, but you can edit that if you wish.
  + The front part of the row is the “input” part, with data describing the household. The back part of the row is the “choice” part, with data describing the household’s choice.
  + By editing the script, you can change the size of the seed, the amount of response data, and the number of choices. If you set a 2-dimensional seed, but provide 10 floating point responses, then there is more than enough data to learn the mapping. If you set a 10-dimensional seed, but provide only 2 floating point responses, then there is not enough.
  + I ran it simply by stepping through it with Spyder.
* survey-nn-XX.py: generic NN to guess the vehicle choice on each row from the other responses on that row. Each is written in the basic Tensorflow format as of the July version of TF, and will run with whatever processor you have. That is, whether you have a CPU-only system or have a GPU installed does not change the Python script at all. (It changes which TF library you install, but they have the same interface).
  + The XX part indicates how many layers are in the neural net.
  + Again, you should be able to just step through it with Spyder. It is hard-coded to read the format produced by synth-survey, so any changes to the CSV format must be done on both the generation and learning sides. The final output is a graph of how performance on validation data (not training) improved, according to several measures.
* survey-kt-XX.py: NN using the KTAB methodology to guess vehicle choice on each row from the other responses on that row. The custom output layer uses the Central Position Theorem (CPT) to guess what the collective-decision-making process for that household would choose. That is, the NN learns how to construct a different CDMP model for each household, then uses the estimated utility matrix and actor-strengths for that household in the CPT to guess their choice. It uses the discrete enumerated model of politics (EMP) over the seven-or-so vehicle options, not the continuous spatial model of politics (SMP).
  + The XX part indicates how many layers are in the neural net.
  + Again, you should be able to just step through it with Spyder. The CSV format must match that which was synthesized, and the final output is the same kind of graphs.
* [2017-07-29-tmp-survey-perf.tgz](https://github.com/KAPSARC/ktab-priv/blob/master/survey/2017-07-29-tmp-survey-perf.tgz) An archive of runs from that date, as well as an analysis of variance (ANOVA) to show that using KTAB-esq learning improved the result.

To apply these scripts to the actual survey data, several steps need to performed.

1. Convert the survey data into CSV data. As we discussed, it will be necessary to determine what each field in data means, whether it describes the household structure or the choice of new vehicle, whether it was actually used in the data set available, and so on. To use the CSV format described above will require representing every survey response as the following:
   1. Floating point numbers to represent survey responses like income, number of family members, and other things represented by a number.
   2. The binary values represent the "one hot" encoding of choice from a set, like the county of residence, previous model of car, new car chosen, etc. This will require determining how many categorical responses (input or choice) are in the data.
2. Alter the existing survey-kt-XX.py to read the fields of the actual CSV data. This should require only changing the parameter indicating how many fields are inputs (it does not distinguish between floating or binary) and how many are outputs. As I recall, we agreed on 7 as the number of output choices, but that could easily be changed.