**Performance Measure**

Firstly, I have been re-thinking the performance measure. Especially,

where is number of observations, is number of states, is true state for observation , and is the probability of transitioning to state in observation , based on the transition matrix or neural network output.

A better alternative is to take the log of the probability to form the entropy:

This is similar to PTP, but has the advantage of being a standard penalty measure (lower values are better), well-known in the literature, and may represent the smaller transition states better.

**Propagation of transition probabilities / Forecasting**

If we want to forecast months ahead from an initial state vector , this is easy using transition matrices:

where M is the transition matrix.

However, as we discussed in our last meeting, it is not so straightforward with neural networks (NN). There are two approaches.

Firstly, represent a trained NN for state transition as the function

i.e. it takes a state vector as input, along with a vector of other predictor veariables (from ), and outputs a new transition probability vector.

Then, **method 1** (propagating the transition probabilities) is given be the algorithm:

I expect this can be coded fairly easily in Python, by passing the entire test data through the NN for each iteration. Note, however, that each loan (mortgage) should only be represented once in this test. Pick out maybe the first one.

Then, **method 2** (getting transition probabilities for all possible state transitions, then taking expected value based on previous transition probabilities), is given by the algorithm:

where such that each when and (i.e. one-hot encoding of state ).

This seems like it will be more difficult to implement in Python. Pass the whole test data in each iteration, and also for each possible state but maybe there is a smarter way?

Hope you can have a go at implementing methods 1 & 2.