The 'Mouse Bandit' Project

Shay Neufeld | Gil Mandelbaum

Progress Report: 11/28/2016

Table of contents

Progress

i. Processing the data into features

In the last milestone, we presented processed behavior data where each row was a trial that contained all the information available about that trial:

```
Elapsed Time (s)          6.64
Since last trial (s)     1.523
Trial Duration (s)        0.54
Port Poked               Right
Right Reward Prob          0.9
Left Reward Prob           0.1
Reward Given                 1
```

However, the main idea behind our project is that past experiences guide future decisions. If you go to the right port five times in a row and never get a reward, that is going to instruct you to perhaps try the left port, for example. So we want to provide our model with features that contain information about past decisions and decision outcomes.

This involved us making some judgement calls as to what the model is able to use to predict the future. Our judgements erred on the side of providing the model with more than less information (after all, we can always cut out predictors later), and was guided by our intuition for what a mouse is likely to remember and use when making a decision. We decided to split the features into two general types that very roughly correspond to ideas of '*short term*' and '*long term*' memory. The '*short term*' memory features contain precise information about the past few decisions made, whereas the '*long term*' memory features contain less precise information about decisions farther into the past. This is inspired from the idea that we generally remember things that happened most recently with the most amount of detail, but tend to sort of 'average together' events farther in the past to provide a more general 'feeling' rather than detailed account of events.

The *'long-term'* memory features are the following:

```
Previous_10_Left_Choice      : # left choices in last 10 trials
Previous_10_Right_Choice     : # right choices in last 10 trials
Previous_10_Left_Reward      : # rewards from left in last 10 trials
Previous_10_Right_Reward     : # rewards from right in last 10 trials
Streak                       : current reward streak
```

A *streak* of +2 means that animal currently has gotten 2 rewards in a row, whereas -2 means the last reward was 3 trials ago.

The '*short-term'* memory features are the following:

```
Port                         : Port chosen
Reward                       : 1: reward given 0: no reward
ITI                          : Time elapsed since last trial
trialDuration                : Time between trial initiation & decision
```

We decided to provide these features for the 5 most previous trials (with respect to the trial the model is predicting), as well as the *ITI* and *trialDuration* for the trial being predicted.

Therefore, we have a total of 27 predictors: 5 'long-term' memory features, and 22 'short-term' memory features. We wrote a python function that takes the trial data (shown above) and converts it into a dataframe where each row is a decision that contains all 27 predictors as well as the decision made by the animal.
(located in *datapreprocessing_code/bandit_preprocessing.py*)

ii. The goal of the model

Initially, we thought to build a model that predicted which way the mouse would go next (left or right). However, what we didn't appreciate was how seldom the animal actually switches ports. The data we are analyzing is from a behavior where the reward probabilities of the ports change every ~50 trials, and a mouse will do 300-500 trials per session. If the reward probabilities are 80-20, that results in ~10% 'switch' or 'explore' trials (ie. the animal make a different decision that his previous choice) and ~90% 'stay' or 'exploit' trials.

Therefore, a model that simply predicts whatever the mouse did last will score 90% (for 80-20 probabilities) and 95% for 90-10 reward probabilities. Indeed, this is what we saw. Even taking out the previous decision, the model still scored ~95% (now using whatever the mouse did 2 trials ago).

What we are really interested in is predicting when a mouse 'changes its mind' – when it switches. At some point, the mouse accumulates enough evidence (from the reward outcomes) that perhaps things have changed and it should change its behavior. It's precisely that point that we want to try and model. So instead of predicting which port the mouse chooses, we thought it might be more salient to instead just predict when the mouse executes an 'explore' trial.

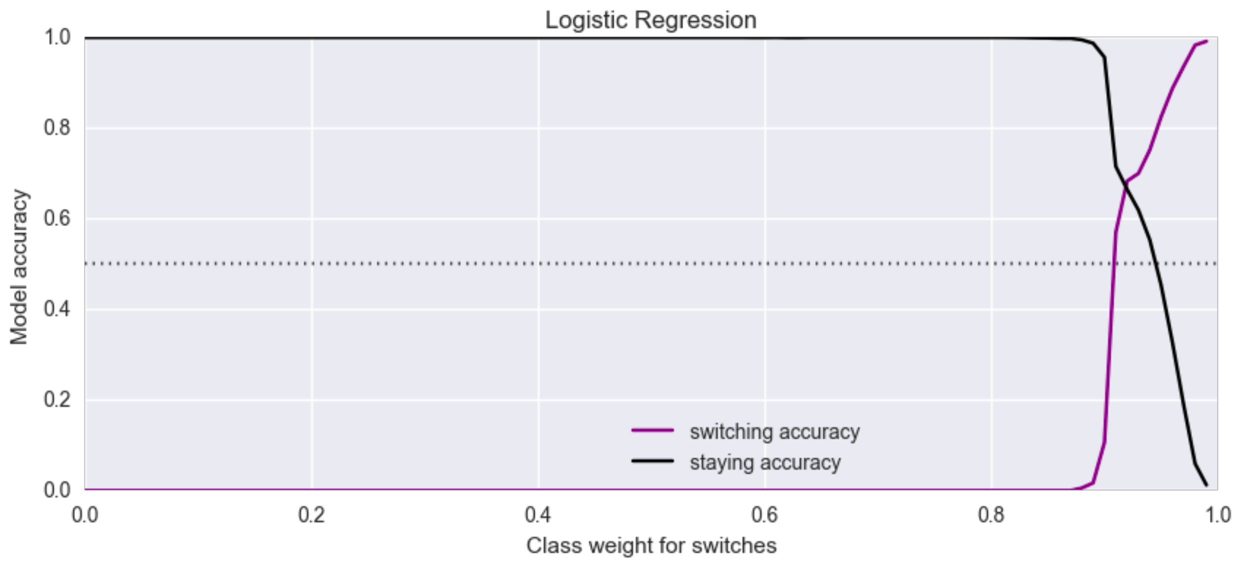
We are open to discussing this further and figuring out other potential goals for our model. For the rest of this progress report, we will discuss our attempts to model when the mouse decides to 'explore', which becomes a bit of an anomaly or low event detection problem akin to the 'diagnosing a rare disease' scenario we have encountered in class.

iii. Comparison of 3 simple classifiers: Logistic Regression, LDA, and a Decision Tree

(from LogReg_LDA_DecisionTree_Comparison.ipynb)

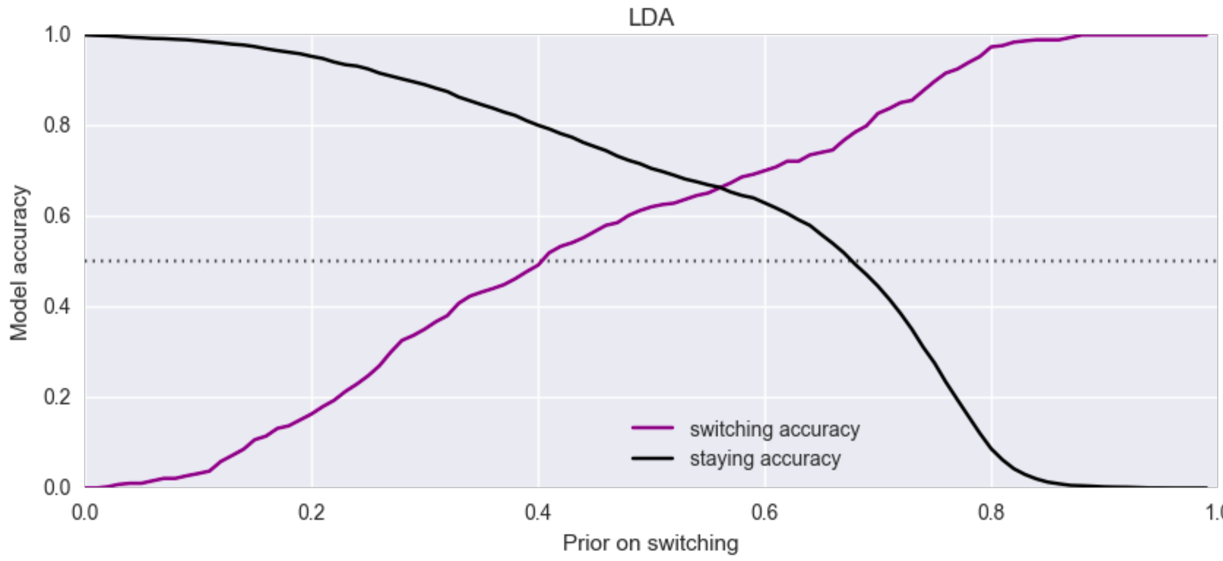
To begin, we used data from 1 well trained mouse performing the task with 80/20 % probability of reward from each port over 10 days. From this data, there was a total of **5861 trials** and **511 switch trials** (~9% of all trials).

Logistic Regression – tuning class weights:

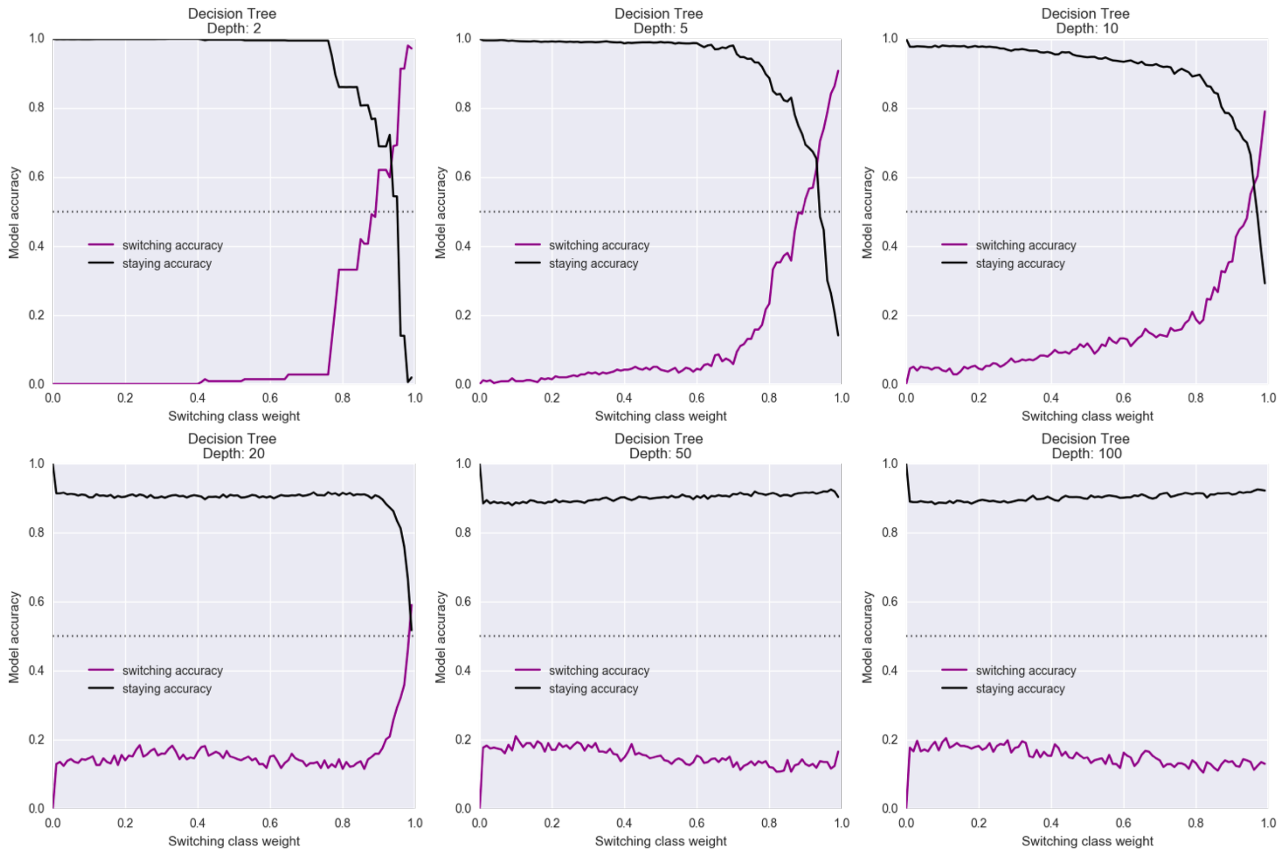


Here we used sklearn's LogisticRegressionCV – so for each class weight pair, the model also did cross validation against 10 different regularization parameters and chose the best one.

LDA – tuning class priors:



Decision Tree – tuning depths and class weights:



All 3 models had comparable maximum performance at ~60-70% accuracy for both switching and staying trials.

Comparing the best models for each:

Logistic Regression:
```
           Predicted NO        Predicted YES
True NO         1032.0          572.0
True YES          54.0          101.0

F1: 0.244

Accuracy on class 0: 0.64
Accuracy on class 1: 0.65
```

LDA:
```
            Predicted NO        Predicted YES
True NO         1046.0          563.0
True YES          50.0          100.0

F1: 0.246

Accuracy on class 0: 0.65
Accuracy on class 1: 0.67
```

Decision Tree:
```
            Predicted NO        Predicted YES
True NO         1122.0          482.0
True YES          59.0           96.0

F1: 0.262

Accuracy on class 0: 0.70
Accuracy on class 1: 0.62
```

The logistic regression and LDA classifiers performed quite similarly. The decision tree performed a little better on the switches, predicting fewer false positives than the other two models. For now, we decided to proceed with the LDA model, partly because the concept of priors on the decision of whether to switch fits nicely with computational neuroscience models where certain actions are modeled with specific priors. In addition, it would be cool to model how these priors change – or even how the model could estimate the change in the priors – given different reward probabilities or expertise in the task. However, that being said, we would be open to discussing the other two models (and more) as we move forward.

We decided against using an ensemble method like random forest or ADABoost, since part of our objective is to use the model to gain some insight into what information the mouse may be using to guide its decisions. Therefore, it is of particular importance to us to create an interpretable model.

iv. Comparison of different features

(*from comparing_different_feature_subsets.ipynb)*

We made some judgement calls for which features to include. But which features actually have the most predictive power? We compared the following different feature subsets with a LDA classifier:

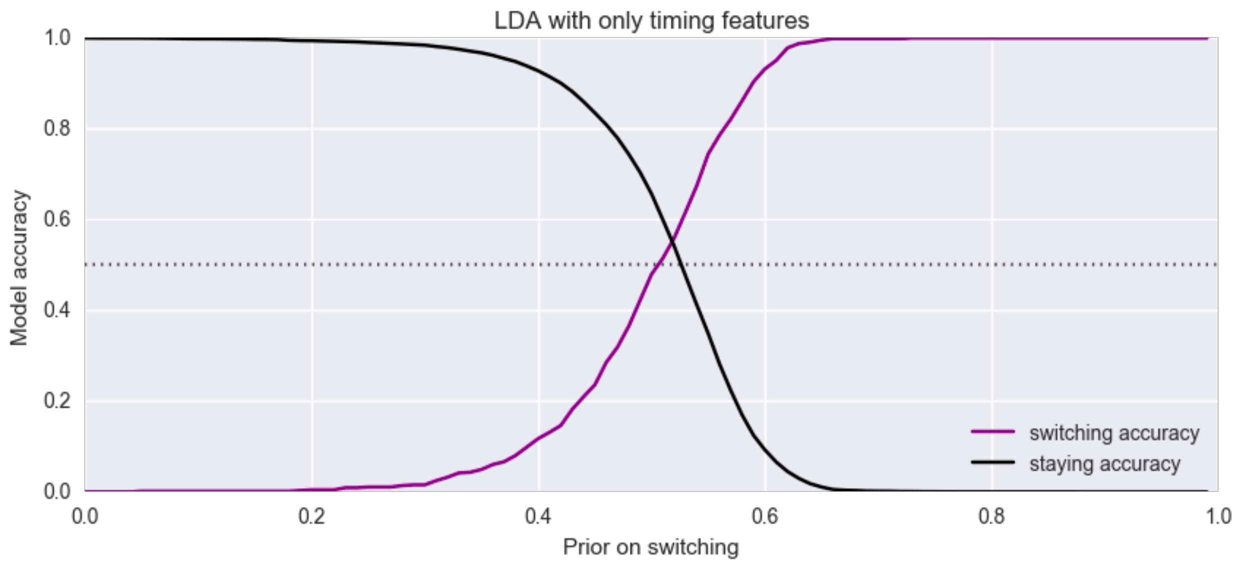
Only 'timing' features (i.e. inter-trial-interval and trial duration)
Only 'outcome' featuers (i.e. whether a reward was given)
Only 'long term memory' features
Only 'shot term memory' features

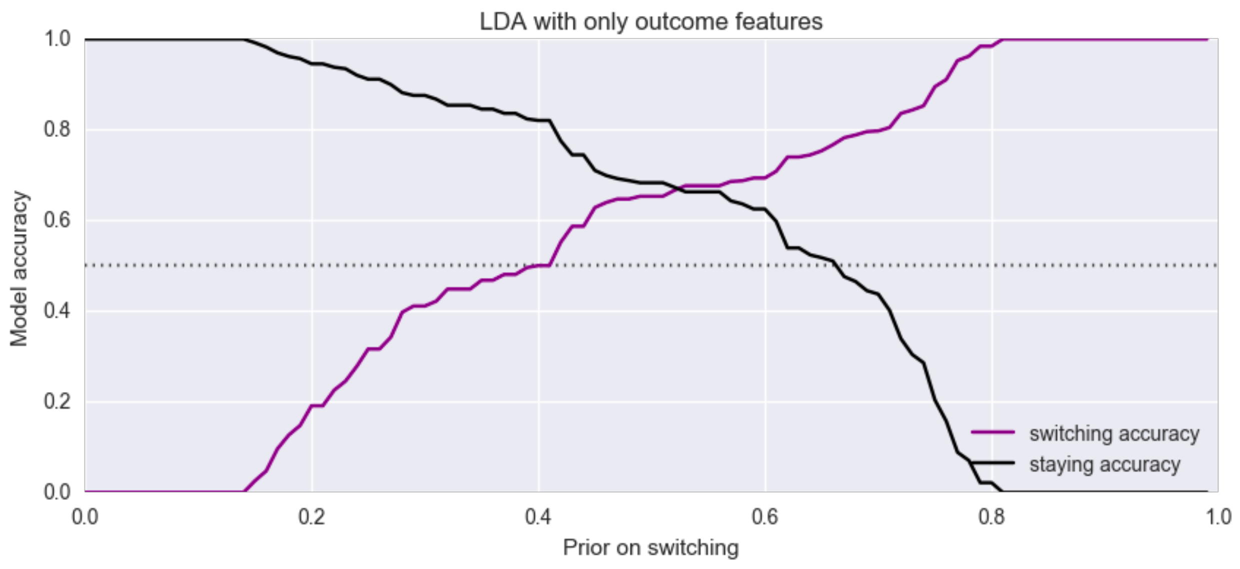
First, let's compare 'timing' features to 'outcome' features. Using the priors found to maximize accuracy previously, the LDA model behaved in the following way:

```
Outcome features:
          Predicted NO  Predicted YES
True NO         1965.0         1184.0
True YES          75.0          174.0

F1: 0.217

Accuracy on class 0: 0.62
Accuracy on class 1: 0.70

Timing features:
          Predicted NO  Predicted YES
True NO          243.0         2906.0
True YES          20.0          229.0

F1: 0.135

Accuracy on class 0: 0.08
Accuracy on class 1: 0.92
```

```
All features:
         Predicted NO  Predicted YES
True NO        1945.0         1204.0
True YES         72.0          177.0

F1: 0.217

Accuracy on class 0: 0.62
Accuracy on class 1: 0.71
```

The model using outcome features was nearly identical to the model using all features, and the model with only timing features was much worse. Perhaps, when using the timing features, different priors should be used? We tuned the priors with cross-validation to see if that made a big difference:





Best model with timing features:

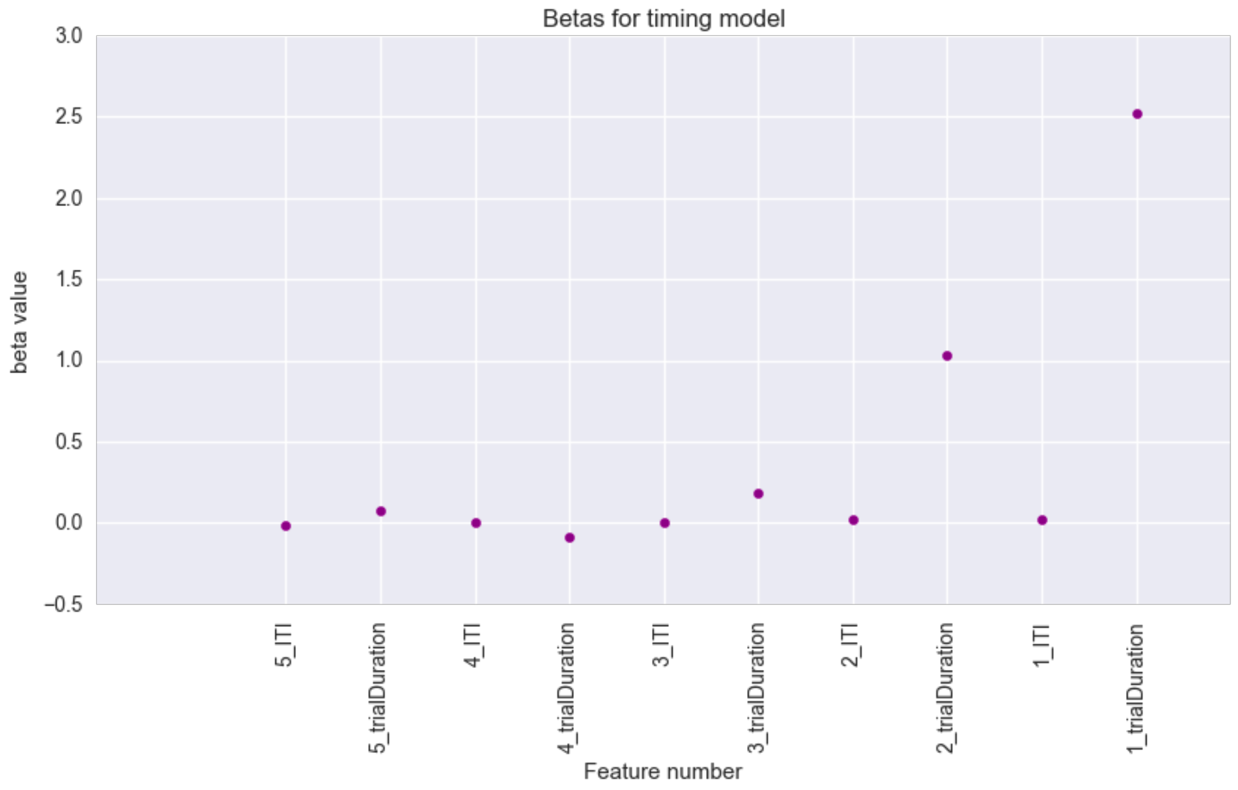
```
         Predicted NO         Predicted YES
True NO         1683.0         1466.0
True YES         114.0          135.0

F1: 0.146

Accuracy on class 0: 0.53
Accuracy on class 1: 0.54
```

Best model with outcome features;

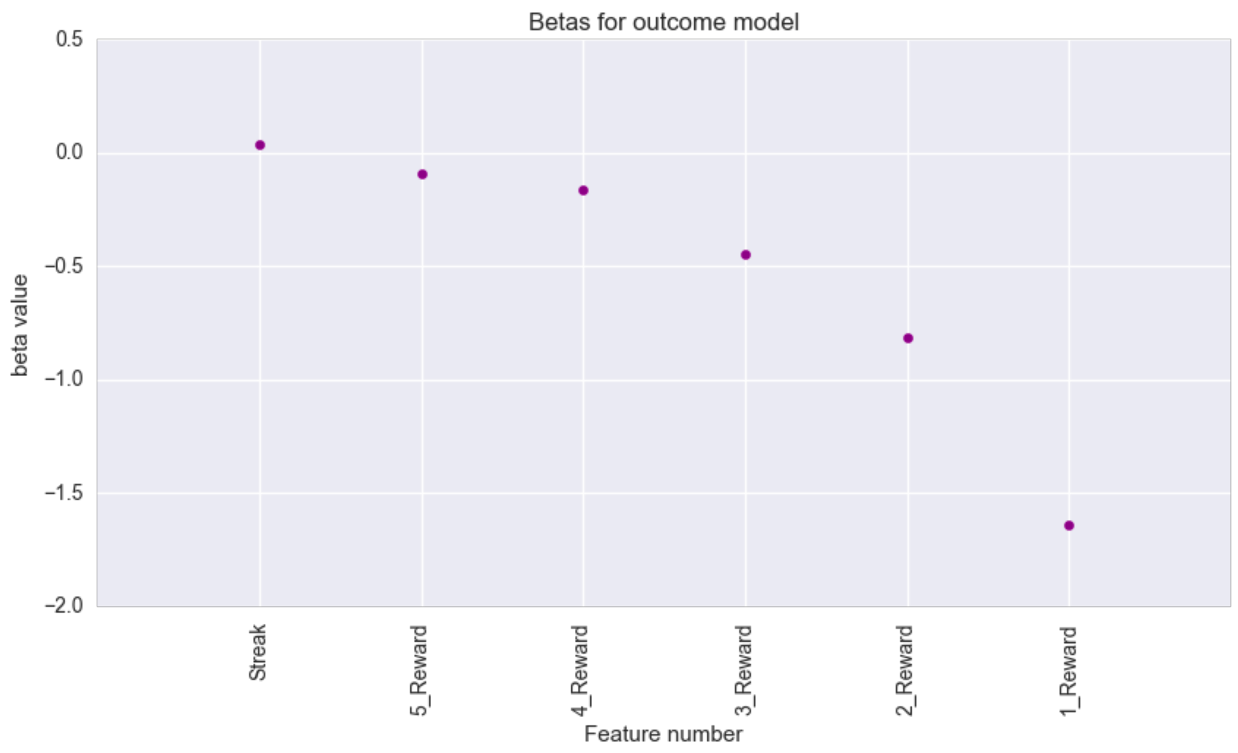
```
            Predicted NO        Predicted YES
True NO         2182.0          967.0
True YES          91.0          158.0

F1: 0.230

Accuracy on class 0: 0.69
Accuracy on class 1: 0.63
```

Even with parameter tuning, the model with only timing features does much worse than the outcome features. This is not surprising from our understanding of behavior – clearly the reward outcomes are what is ultimately driving the animal to make its decision. However, we thought that some of the timing information may relate to how the animal is making its decision – from our last report, we showed that the distribution of inter-trial-intervals looked different across ‘same’ and ‘switch’ trials. So it was somewhat to our surprise that the adding the timing features to the model features did not add any additional predictive power in the data we analyzed.

If we look at the betas for the model using timing features, we can see that the largest beta is the trial duration of the most recent trial:



One interpretation that may explain this is when the animal switches, it must travel a (very small) additional distance compared to when it makes the same decision. This difference in distance could account for a small but consistent difference in the trial duration, which would explain why it’s being used by the model here.



The betas for the model using outcome features nicely decrease with time – the most salient outcomes are the most recent ones. One interpretation from this analysis is that the animal – when executing the 80-20 probability task – tends to use the previous ~3 reward outcomes when making a decision, however by the 4th and 5th trials the betas are pretty small. Surprisingly, the streak feature was not predictive (I thought it would be, after all its sort of analogous to a gambler on a 'hot streak' or a 'cold streak', but maybe mice don't think that way).

Comparing 'short-term' and 'long-term' features:

```
Short features:
          Predicted NO  Predicted YES
True NO         1940.0         1209.0
True YES          69.0          180.0

F1: 0.220

Accuracy on class 0: 0.62
Accuracy on class 1: 0.72
```

```
Long features:
          Predicted NO  Predicted YES
True NO         1330.0         1819.0
True YES          55.0          194.0

F1: 0.172

Accuracy on class 0: 0.42
Accuracy on class 1: 0.78

All features:
          Predicted NO  Predicted YES
True NO         1945.0         1204.0
True YES          72.0          177.0

F1: 0.217

Accuracy on class 0: 0.62
Accuracy on class 1: 0.71
```

Here we can see that short-term features outperformed long-term features. Again, from a behavioral point of view this fits nicely with the concept that the most immediate previous actions are going to be the most salient in making your next decision.

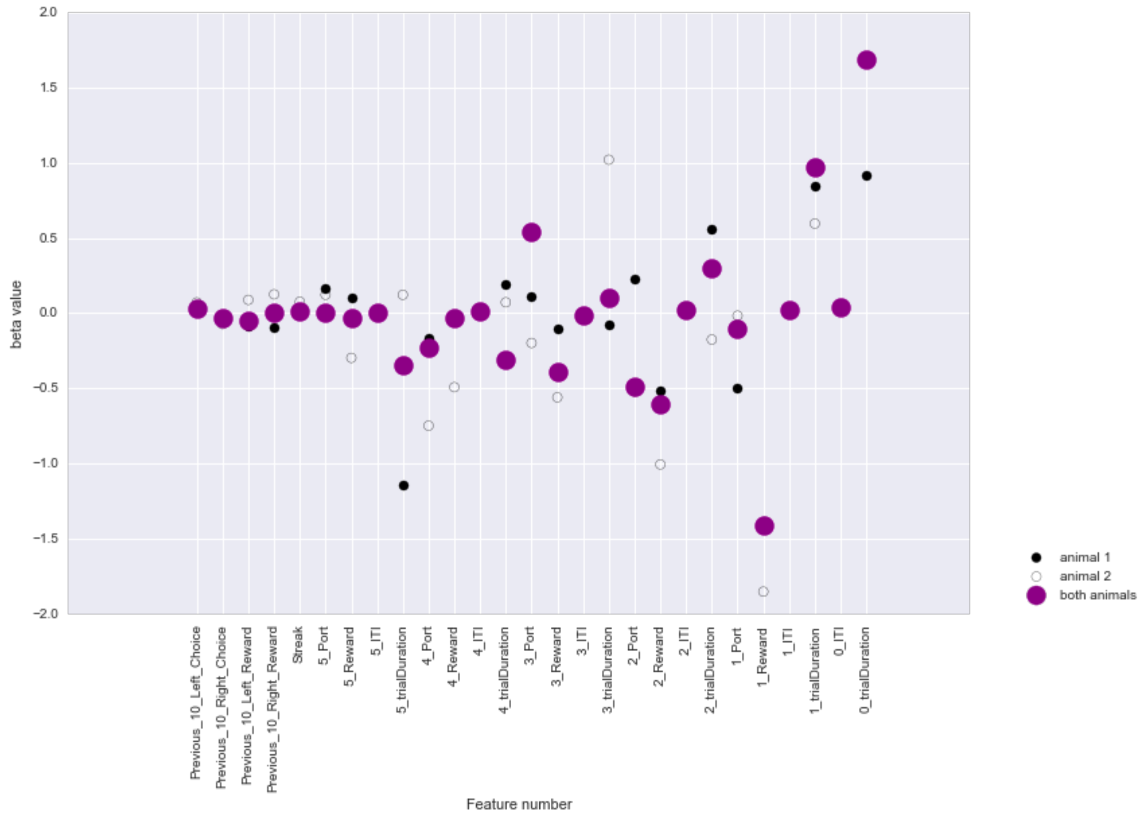
In conclusion, with the data analyzed here (which is just the 80-20 % reward probability task), it appears the most salient features are the 3-4 most recent reward outcomes.

v. Comparing across 2 different animals

(from comparing_across_animals.ipynb)

As we move to combine more data, look across learning and different reward probabilities, one important thing to keep in mind is whether it is fair to combine data from different animals. That is – is it reasonable to assume that different mice are using the same information to make decisions? Clearly, there is some shared information – the reward outcomes are going to matter for every mouse. But it's possible that there exist subtle differences in the strategies adopted by different mice. We have not yet addressed this thoroughly, but did a preliminary comparison of two mice performing the same task (80-20) with comparable accuracy. Below are the accuracies and beta coefficients for models trained on animal 1, animal 2, or both.

```
for animal number 1:

             Predicted NO  Predicted YES
True NO            1010.0          592.0
True YES             45.0          112.0

F1: 0.260
Accuracy on class 0: 0.63
Accuracy on class 1: 0.71

for animal number 2:

             Predicted NO  Predicted YES
True NO             951.0          577.0
True YES             31.0           80.0

F1: 0.208
Accuracy on class 0: 0.62
Accuracy on class 1: 0.72

for both animals:

             Predicted NO  Predicted YES
True NO            1907.0         1204.0
True YES             81.0          206.0

F1: 0.243
Accuracy on class 0: 0.61
Accuracy on class 1: 0.72
```



From this very preliminary examination, the two animals look very similar. This provides some evidence that combining data across animals is justified. However, it will be interesting to compare across animals with different behavior expertise, as well as across learning.

Proposal for future work

Modeling learning

So far in our analysis we grouped behavioral sessions together. For example, we grouped 10 sessions of a well-trained animal to then explore multiple regression models with the goal of predicting performance outcome such as spout choice and when the animal switched between spouts. Because we grouped all sessions (and the animal performance was consistent during those sessions) we did not ask questions about whether our models outcome, for example beta coefficients or predictive power of port choice and switch, change as the animal's performance improves over days, i.e., learns the task. We think that our current models capture the features that describe the well-trained animals strategy to solve the task. Applying the model to sessions in which the animal is learning will allow us to ask question of how the model output evolves over that time, ultimately giving us better understanding of how the animal formed the learned strategy. Also, it would be of great interest to compare how the model evolves over time across animals to look at whether they are learning in different ways.

Comparing different reward probabilities

In addition we plan use the analysis we have done until now (on trained animals) and future analysis (during learning as described above) to ask questions about animals that were trained in the same task structure but with different reward delivery probabilities. So far our analysis has been reserved to behavior tasks with 80/20% reward probabilities. However, we have data that spans most of the parameter space. Do mice change their strategy in different reward probability contexts, or do they learn a single strategy that they employ regardless of the probabilities?

Online/more complex models

Lastly, we hope to explore additional aspects of this animal behavior with tools not taught in the course. For example, we could implement a model that updates during a given session. This is of particular interest to us, since such a model could potentially be employed to predict decisions online and allow us to modulate the behavioral parameters accordingly. and thus test specific hypotheses about how the mouse is making its decision.