Bianca: Automated classification of simple (binary) and complex (multiclass) behavior in mice

Piper Keyes and Salvador Valdes
{pckeyes, svaldes}@stanford.edu

In the past decade, the field of systems neuroscience has increasingly relied on genetic tools to dissect complex neural circuits. Tools such as optogenetics and pharmacogenetics afford neuroscientists the ability to manipulate specific neural populations to investigate their role in various behavioral processes. A growing problem in the field has arisen from the quantification of such behavioral data. In simple behavioral assays, such as locomotor or feeding behavior, generating an unbiased, repeatable quantification is fairly simple. However, many experimental paradigms attempt to answer more interesting questions about animal behavior and thus output more complex behavioral data. With increased complexity comes increased variation and bias in quantifying this data, leading to poor repeatability across labs. Aiming to alleviate this problem, we developed a program to automate classification of complex behavioral data.

Introduction

The continuously expanding field of systems neuroscience focuses largely on investigating how specific neural pathways influence behavior. A ubiquitously utilized method (primarily in rodent research) for probing these sorts of questions involves the use of genetic tools that allow scientists to manipulate a specific subset of neurons using exogenous stimuli. Specifically, these tools utilize the machinery of adeno-associated viruses (i.e. AAVs) to insert engineered DNA into a mouse's genome within specific neurons in the brain. The engineered DNA codes for proteins that can be manipulated by an experimenter to either increase or decrease the activity of those neurons. One such tool, termed "pharmacogenetics", introduces a protein that has been biochemically engineered to bind to a single chemical, clozapine-N-oxide, that is not endogenously present in mice. These receptors, commonly referred to as DREADDs (designer receptors exclusively activated by designer drugs). By infusing the animal's brain with clozapine-N-oxide, scientists can either inhibit or enhance neuronal of the neurons that express the DREADDs. Another set of genetic tools, termed "optogenetics", works by expressing a light-sensitive protein into mouse neurons. When exposed to specific wavelengths of light, these proteins can also either activate or inhibit neuronal firing. Furthermore, the AAVs that express the DREADDs or light-sensitive proteins can be designed only to integrate into neurons with a specific genetic expression profile. These tools offer a powerful method for examining the roles of specific brain nuclei (and cell-types therein) and neural pathways, enabling sophisticated dissection of the roles of distinct circuits in the brain.

The primary data generated by pharmaco- and optogenetics comes from observable changes in mouse behavior. For example, one of the earliest optogenetic studies investigated how parallel neural pathways differentially impacted locomotor behavior[1]. More recently, pharmaco- and optogenetic experiments have been used to examine which brain nuclei send satiety signals that tell a mouse it is full and should stop eating[2]. Given the heavy reliance on behavioral output to interpret pharmaco- and optogenetic results, it is essential that quantification of behavioral assays is both accurate and reproducible. In the examples previously described, behavioral quantification is relatively simple; many programs exist that can track a mouse's locomotor behavior, and feeding behavior is most often quantified by the total amount of food eaten.

However, genetic tools are widely being utilized to study much more complex behavioral paradigms, such as social behavior behaviors". Quantification of these behaviors must be done by hand, which inevitably hinders both accuracy and reproducibility, as quantification across experimenters is always highly variable and can be biased. We therefore aim to develop software that applies machine learning techniques to quantify and label complex behavior in mice.

Related works

Groups have attempted to create behavioral tracking programs similar to the one we discuss here. A group at Janelia Farms has implemented behavior-tracking software that utilizes single frames as well as groups of frames as separate features[3], which could be useful for our project as well. We adapted this concept by subtracting preceding video frames from the current frame to provide information about the movement of the mouse that a static images alone cannot represent (discussed further below in "Image pre-processing"). Another paper applied machine learning to identify human behavior in movies[4]. This group used support vector machines, scale-invariant feature transformation, and histogram of gradients, which are common algorithms used in computer vision that came up repeatedly in our background research. We plan to apply these techniques in our automated behavior-tracking program.

Development of a programming environment

In order to create our automated behavior tracking software, we first needed to develop a programming environment that could facilitate human classification of behavioral data. To accomplish this, we utilized OpenCV and Python to create a free-standing hand classifier program (Figure 1A). The program's script takes in a directory as an input argument and finds videos in that directory. It then presents a text box to the human user asking which video she would like to score. After video selection, the video is displayed in a GUI that includes a guide to the functionality of the hand classifier. It includes descriptions of which keys code which behaviors, a displayed label of which behavior the program currently thinks is being recorded, and buttons to play, pause, slow-down and speed-up the video. The number of labels and their key assignments can easily be changed to suit any behavioral assay. The tool creates a text file (Figure 1B) at the beginning of each recording session with the name of the corresponding video. It records at which frame a keystroke was made and records that frame-label pair. The label "0" is reserved for data that the user wishes to discard, and is also used to mark the beginning and end of each text file. The user may quit the program at any point and remaining frames will be labeled "0".

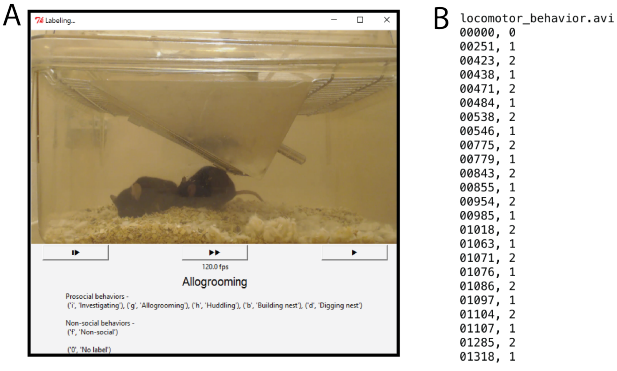

Figure 1: Software for hand classification of dataset. A) GUI that allows for interacting with the video file being classified. B) Sample classification text file

After the user has classified the videos of her choosing, a label parsing script takes in a text file to generate training data for future learning algorithms. To compensate for human delay in coding a mouse's changing behavior, the first and last 15 frames of any clumps of consecutive frames with the same label are discarded. Additionally, to correct for mistaken keystrokes, any clumps with < 31 frames are discarded.

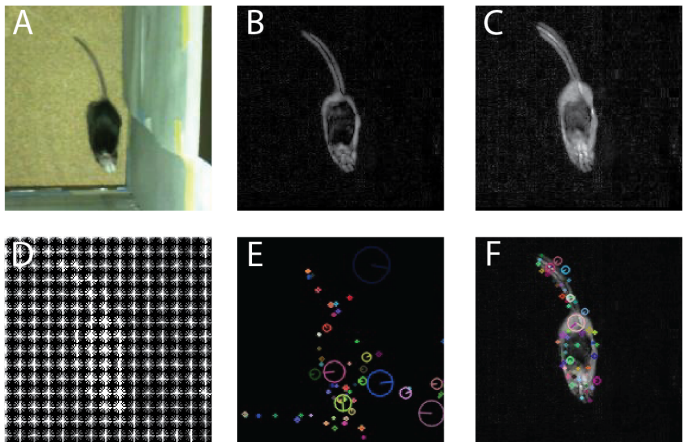

Figure 2: Image pre-processing. A) Raw frame from a video of mouse locomotor behavior. B) The same frame with the previous frame subtracted. C) The same frame with an average of the previous five frames subtracted. D) HoG features extracted from the averaged subtracted frame. E) SIFT features extracted from the non-averaged frame. F) SIFT features extracted from the averaged frame.

Image pre-processing

The videos of behaving mice inevitably include data that will not inform and may even introduce noise into our learning algorithms, such as the environment in which the mouse behaves. To increase our signal-to-noise ratio, we took steps to pre-process the videos before feeding them into our learning algorithms (Figure 2). First, we decided that subtracting each frame from its succeeding frame would decrease the noise of the data while preserving information about the behavior of the mouse. We therefore created a dataset of subtracted frames for our learners (Figure 2B). Furthermore, in order to increase the information about the mouse's behavior in each frame, we created a dataset where we sub-

tracted from each frame and average of the five preceding frames (Figure 2C). In this dataset, a moving mouse would leave a long trail behind it, contrasting greatly with a stationary mouse. Finally we created a mask for each frame centered on the behaving mouse that followed the mouse's movement, further decreasing unnecessary aspects of our dataset.

Classification of simple behavior

Though our ultimate aim in this project is to enable reliable automated classification of complex mouse behavior, we first wanted to test our project's design on a simpler behavioral dataset. Attempting to automate classification of a simple, binary behavior would provide insight as to 1) whether we can classify behavior at all, and 2) which combinations of feature extraction and learning algorithms perform the best. We therefore chose to initially classify simple locomotor behavior in mice. Here, we used videos of single mice in an open field, generated by Piper Keyes in Xiaoke Chen's lab (Figure 2A). After human classification of whether the mouse was moving or still, we extracted feature vectors in multiple ways. First, we used the averaged subtracted frames as feature vectors. From the OpenCV and Python libraries (including numpy and scikit-learn), we also utilized scale-invariant feature transformation (SIFT) and histogram of orientation gradients (HoG) to extract feature vectors from the averaged subtracted frames (Figure 2D, F). Finally, as SIFT is best at detecting hard edges, we also applied SIFT to the non-averaged subtracted frames which display sharper edges than the averaged subtracted frames (Figure 2E). As HoG produces a gradient of vectors across the entire image, we reasoned that the averaged subtracted frames would suffice.

We then used our feature vectors to train on various learning algorithms. These include linear support vector machines (SVM), decision tree of depth 5, random forest of depth five and SVM with radial basis function (RBF) kernel, with the exception of pairing subtracted averaged frames with SVM with RBF kernel due to high computational cost that our computers couldn't process. The first half of the classification text file was used for training data and the second half was used as holdout data. Error analysis (Table 1) revealed that, in general, our algorithms performed fairly poorly on this dataset. The majority of our results generated a test error worse than random classification (0.5). This could result from a number of factors. First, our dataset was not very large. Furthermore, the video was only hand classified once by a single observer, which likely means it includes a substantial amount of human error. Additionally, we did not spend a relatively large amount of time tuning the various parameters of the feature extraction methods and learning algorithms. However, patterns did emerge from our results that suggested which combinations classify the behavior most successfully. In particular, using SIFT without averaging frames to extract features and learning with either linear SVM or a decision tree resulted in somewhat high training error but a comparatively low test error. Additionally, HoG and the raw averaged subtracted frames performed well. We therefore chose to apply these four combinations to our complex behavioral dataset.

Classification of complex behavior

The complex behavioral assay from which

Table 1: Error analysis for classification of simple behavior. Errors generated by each combination of feature extracting method and learning algorithm when trained and tested on classifications of locomotor behavior in mice.

	Subtracted averaged frames		HoG with subtracted averaged frames		SIFT with subtracted averaged frames		SIFT with subtracted non-averaged frames	
	Train	Test	Train	Test	Train	Test	Train	Test
Linear SVM	0.000	0.413	0.125	0.469	0.396	0.655	0.441	0.272
Decision tree (depth = 5)	0.339	0.365	0.398	0.617	0.41	0.612	0.421	0.388
Random forest (depth = 5)	0.338	0.573	0.331	0.645	0.42	0.629	0.461	0.698
SVM with RBF kernel	N/A	N/A	0.497	0.764	0.290	0.620	0.154	0.554

our dataset arose was designed and carried out by Greg Nachtrab in Xiaoke Chen's lab. In this paradigm, the subject mouse has cohabitated with a cage mate for at least a year, resulting in a strong social bond between the two animals. During the experiment, the cage mate is anesthetized and placed into the home cage with the subject mouse. We have observed that, upon recognizing that its cage mate is still breathing but unable to be awoken, the subject mouse will generally display a constellation of prosocial, "compassionate" behaviors towards their anesthetized cage mate (Figure 3). These behaviors include investigation, allogrooming (i.e. grooming another mouse), digging out space for building a new nest around the anesthetized mouse, building the new nest, and huddling. The mouse also engages in non-social behaviors. In these videos, the mice tend to transition through the behaviors linearly. The mice initially (i.e. first 20-30 minutes) investigate and allogroom their cage mate, while later (i.e. last 20-30 minutes) the mice will dig/build the nest and huddle. We therefore divided the classification text files, generated on two different videos, into quarters. From the first video, the first and third quarters were designated as hold-out data. From the second video, the second and fourth quarters were designated as hold out data. This strategy ensures that both the training and holdout data contain a representative sample of the entire range of classified behaviors.

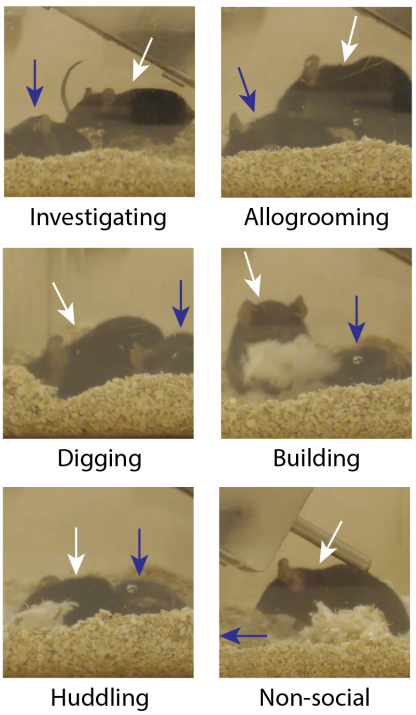

Figure 3: Complex social behaviors in mice. Representative images of the range of behaviors observed in our dataset. White arrows signify the subject mouse. Blue arrows represent the anesthetized cage mate.

We used the four combinations of feature extractors and learning algorithms described above on our dataset. Both the SIFT with non-averaged frame subtraction/linear SVM and the raw subtracted frames/decision tree combinations produced similar training errors around ~0.2. The SIFT with non-averaged frame subtraction/decision tree combination performed more poorly, resulting in a training error of 0.352. The HoG/linear SVM combination produced 0.0 training error, suggesting an overfitting of the model. Though these training errors are still relatively large, compared to the simple behavior dataset they are significantly better. Given the six different possible classes, random chance would result in ~0.83 error as compared to the 0.5 error in the previous binary dataset. Moreover, each pairing produces ~0.6 test error, which is still better than random and also better than some test errors produced on the simple dataset.

Sources of error in this dataset are likely similar to those in the previous dataset. The behavior was again only hand classified by one observer with minimal duplicate classifications of specific frames. Additionally, as mentioned previously, in order to improve the speed at which the learning algorithms ran, we decrease the resolution of the videos. If we had maintained the videos at the original resolution, the regained data may have further aided the learning algorithm.

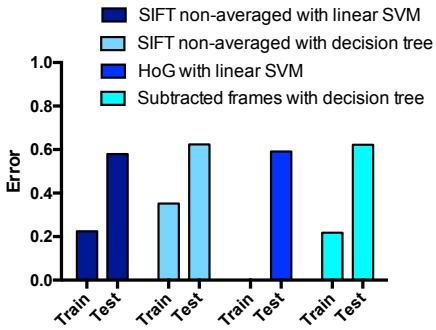

Figure 4: Error analysis for classification of complex behavior. Across the four combinations of feature extractors and learning algorithms, SIFT with linear SVM and the raw subtracted frames with decision tree performed the best.

Conclusion

Our project aimed to create an automated behavior classification software for mouse behaviors. We applied our selected feature extraction methods and learning algorithms to simple behaviors (i.e. locomotion) before moving on to complex behaviors (i.e. social interaction). From our learning on simple behaviors, we identified the most accurate combinations of feature extraction methods and learning algorithms: SIFT on non-averaged subtracted frames with linear SVM, SIFT on non-averaged subtracted frames with decision trees, HoG on averaged subtracted frames with linear SVM, and raw averaged subtracted frames with decision trees. We applied these four combinations to our complex behavioral dataset and found that (whatever we end up finding).

In the future, there are many optimizations we would like to apply to our program. First, the reliance of this program on human classification is a weakness that cannot be completely resolved. However, we plan to obtain hand classifications of video files from multiple observers in order to decrease the impact of individual observer bias currently present in the training data. Furthermore, our raw complex behavioral data can be optimized. The current method of data collection utilizes a single camera viewing the mice from one side of the cage. This leads to ambiguities when the subject mouse is behind the anesthetized mouse that are impossible for a human to classify with 100% accuracy and will therefore introduce noise into the training and test data. Including a second camera that records the mice from an orthogonal angle may alleviate this problem. For instance, if an observer comes to a point in the video where they cannot resolve with confidence what the mouse is doing, she can select the "no label" option until the behavior is clear again. Then, when watching the same behavior from the orthogonal camera angle, this behavior may become clear and classifiable. Thus, these ambiguous instances in the behavior are more likely to receive an accurate classification.

Additionally, though our errors generated currently are quite large, they are not completely interpretable without comparing them to between observer or even within observer error. In order to provide an improvement to current behavior classification methods, our program need only outperform human error. In order to address this, we would need to compare human classification of the same video by different observers or between two different classification sessions within the same observer. This would be a simple addition to our program and would validate its usefulness.

Finally, due to lack of computational power and speed, we were unable to test every method of feature extraction we wanted to. We discovered that, at least for SIFT, using frame subtraction caused issues during the simple behavior: when the mouse was completely still, the frame became entirely black and SIFT had no features to extract. We therefore reasoned that applying SIFT and HoG to the raw, unsubtracted frames might yield better results. Furthermore, vectorization of the pixels of each frame could also potentially serve as good feature vectors as they would contain many more sources of data (though likely also more noise). With a more powerful computer, these possibilities could be tested.

References

1. Benjamin S. Freeze, Alexxai V. Kravitz, Nora Hammack, Joshua D. Berke and Anatol C. Kreitzer. *Journal of Neuroscience*. 20 November 2013, **33** (47) 18531-18539
2. Haijiang Cai, Wulf Haubensak, Todd E. Anthony and David J. Anderson. *Nature Neuroscience*. 27 July 2014, **17** 1240-1248
3. Mayank Kabra, Alice A. Robie, Marta Rivera-Alba, Steven Branson, and Kristin Branson. *Nature Methods*. 2 December 2012, **10** (1) 64-67
4. Ivan Laptev, Marcin Marszalek, Cordelia Schmid, Benjamin Rozenfeld. Learning Realistic Human Actions from Movies. CVPR 2008 - IEEE Conference on Computer Vision & Pattern Recognition, Jun 2008, Anchorage, United States. IEEE Computer Society, pp.1-8, 2008