Classifying Learner in Categorical Learning by Clustering Eye-tracking Data

Scott Cheng-Hsin Yang*
(Dated: April 29, 2010)

An eye-tracker recorded participants' gazes while they performed a categorizing task. Can we classify the participants based on these data of locations and times? I clustered a processed form of the time data using K-means and hierarchical clustering. I found two prominent learning classes with subclasses that exhibit different learning rates. I also clustered a processed form of the location data by K-medoids clustering. The results suggest yet another three independent learner classes based on the strategies adopted for the categorizing task.

I. INTRODUCTION

Going through school, we all have experienced that people learn at different rates, with different styles. An interesting question to ask is if there are distinct types of learners. In this report, I examine data from learners undergoing a categorizing task. The participants were asked to identify the images they saw as a member of a number of categories [1]. The locations of the participant's gaze were tracked by an eye-tracking machine, producing a time-series of locations for each learner. I aim to extract information about learner classes based on these time-series.

In more detail, the categorizing task consists of on average 179 trials. Each trial has four phases:

1. get-ready phase — participants get ready for the image categorization.
2. thinking phase — the image appears and participants examine it.
3. answering phase — participants identify the image as one of the four possible categories.
4. feedback phase — the correct answer is shown and participants can reexamine the image.

A preliminary analysis of eye-tracking data showed a fix-shift-fix pattern. In general, Participants spent a considerable amount of time "gazing" at a location, quickly shifted to another location, and gazed again [2]. Using this feature, one can represent the data more compactly with the locations and gaze (fixation) times. A further analysis of the gaze locations revealed that participants rarely gazed at locations that were unimportant. For the thinking phase, this means that the locations recorded fell primarily on one of the three locations where the image had distinct features [2]. The locations are then reduced from a series of real numbers to a sequence of four symbols: **b**, **c**, and **d** for the 3 distinct features, and **a** for all other locations. This report focuses on the sequences of gaze times and location symbols in the thinking phase.

For interpretability and computational feasibility, I further processed the data before classifying. Firstly, for each participant, the gaze times in the sequence were pooled together by trials, resulting in a curve of time versus trial index. Intuitively, one expects that the time a learner spend for each trial decreases as the task progresses; therefore, this simplification of the sequence makes the overall trend of learning rate more apparent. Secondly, for each participant, I searched for the most prominent L-motif in the corresponding sequence of location symbols. The most prominent L-motif is the subsequence of L symbols that appears most frequently in the sequence. The idea is that the motifs represent the special categorizing strategies the participants adopted during the categorization task. Contrasting the two representations, the first emphasizes the *overall* learning behaviour, whereas the second focuses on the *specific* learning patterns.

My goal of finding learner classes from the data falls under the field of unsupervised learning, since there is no information about the learner classes to begin with. An intuitive way to proceed is to first cluster the the trial times and motifs based on some measure of dissimilarity (or similarity). Inspired by an analysis that clusters fMRI time-series [3], I chose to use K-means clustering (and variation K-medoids) for its simplicity and hierarchical clustering for its visual presentation. From the properties of the clusters, I then introduce classes for learners. The rest of the report is organized as follows. In Sec. II, I explain how I clustered the trial times using K-means clustering and introduce the classes I found. In Sec. III, trial times are clustered by hierarchical clustering and results are compared with those from K-means clustering. In Sec. IV, the motifs are clustered via K-medoids. Lastly, I conclude with Sec. V.

*Electronic address: scotty@sfu.ca

II. K-MEANS CLUSTERING OF LEARNING BEHAVIOUR

Most clustering methods, including K-means, need some measure of distance so that closer objects can be clustered together. The most common distance measure for quantitative data is the Euclidean distance. To use this measure, the objects to be compared should have the same dimension. This is not the case for our trial time data, since each participant went through a different number of trials. Instead of finding another distance measure that can accommodate non-matching dimensions, I chose to normalize the data to the same dimension for simplicity. The normalization should also be seen as a further step to bring out the relative overall trends among participants by suppressing absolute differences. A simple normalization scheme is the following:

1. for the i^{th} participant's trial time sequence T_i, normalize T_i by subtracting the mean and dividing by $\max(T_i) - \min(T_i)$ (or the standard deviation). This step is not needed for normalizing the dimension; it is for suppressing absolute differences among participants.

2. choose a common dimension $p \leq \min(M_i)$, where M_i is the number of trials the i^{th} participants went through.

3. pool the M_i data into p segments so that each segment has M_i/p values.

4. take the mean of each segment.

This procedure yields p normalized trial times for the i^{th} participant. I did this for all N participants and stacked the normalized T_is as rows to produce the $N \times p$ matrix T.

Given that the distance measure is the Euclidean distance, the K-means clustering algorithm aims to cluster the N objects into K classes so as to minimize the within-cluster dispersion

$$W_K(C) = \sum_{k=1}^{K} N_k \sum_{C(i)=k} \left\| T_i - \bar{T}_k \right\|^2 , \tag{1}$$

where $C(i)$ is a many-to-one indexing system that records which object belongs to which cluster, and $\bar{T}_k$ is the coordinate of the center of the cluster k. The minimization is done by an enlarged optimization, where $\bar{T}_k$ is replaced by an arbitrary m_k (Section 14. 3. 6 in [4]). The algorithm iterates between two steps: 1) given a cluster indexing system $C(i)$, find the centers m_k that minimizes $W_K(C)$; and 2) given the centers m_k, change $C(i)$ to minimize $W_K(C)$. Usually, one initializes the centers first, and the algorithm iterates between step 2) and 1). I used the built-in K-means function in IGOR Pro 6.03 [5]. Several shortages should be noted. First, the resulting clusters are not guaranteed to be the global minimizer of the $W(C)$, since the algorithm is a greedy one. Second, the resulting clusters depend on the choices of initial centers, and multiple centers sometimes merge into one. Lastly, the number of clusters is user-defined and not directly dictated by the data.

The first problem is generally not solvable but not grossly wrong. The second problem can be avoided by running the algorithm with different initial conditions. To deal with the last problem, I start with the "gap statistic" proposed in [6] and mentioned in the text [4]. The basic idea of gap statistic is to compare the within-cluster dispersion W_k between the observed data and a proper, random reference data as a function of the cluster number k. In [6], the authors note that if the observed data has K well-separated clusters, the W_k of the observed data would decrease faster (slower) than that of the random reference data for $k \leq K$ ($k > K$). They propose that the reference data should have each feature distributed uniformly over the range of observed value for that feature. The gap statistic is defined to be

$$Gap(k) = \log(W_k^*) - \log(W_k), \tag{2}$$

where W_k^* (W_k) is the expected within-cluster dispersion of the random reference data (observed data). The number of clusters is chosen to to be the smallest k for which $Gap(k) \geq Gap(k+1) - s_{k+1}$, where s_{k+1} is an estimate of the standard deviation of $Gap(k+1)$. The advantage of gap statistics over other proposals is in its power to detect single cluster (k =1). In summary, the procedures for using gap statistics are

1. cluster the observed data using an algorithm (K-means here) for $k = 1$ to K.

2. compute the within-cluster dispersion W_k for each k.

3. for feature 1 to p, generate reference feature by sampling from a uniform distribution that covers the range of observed values for that feature.

4. compute the within-cluster dispersion for the reference data.

5. repeat steps 3 and 4 multiple times and obtain the expected within-cluster dispersion W_k^* and its standard deviation.

6. compute $Gap(k)$ using Eq. 2 and choose the number of clusters as mentioned above.

Following the above outline, I found that the trend of $Gap(k)$ is dominated by the trend of $\log(W_k^*)$, and the estimated number of cluster is 1. I suspect that this is because the observed data has a few outliers that cause the reference data to be very spread out compared to the observed data. Under this circumstance, W_k^* dominates as long as W_k is calculated without cluster centers near the outliers. Since the rule chooses the smallest k for which $Gap(k) > Gap(k+1)$ (ignore s_{k+1} here because it is small compare to $\Delta Gap(k)$), and since $W_1^* > W_2^*$ is generally true for the random reference, I will always get k=1 cluster. To get more interesting result, I modified the last step to compute

$$GapRatio(k) = \frac{\left(W_{k-1}^* - W_k^*\right)/W_{k-1}^*}{\left(W_{k-1} - W_k\right)/W_{k-1}} \qquad for\ \ k > 2 \tag{3}$$

and $GapRatio(1) = 1$. This expression is the ratio of the normalized differences in dispersion between the reference and observed data. The normalized difference is the percentage decrease gained by a new cluster given the dispersion at $k-1$. The number of clusters is then chosen to be the k that minimizes $GapRatio(k)$. According to the result in Fig. 1A, the observed data should have four clusters. The standard deviation of $GapRatio(4)$ suggests that the four clusters are only a little more separated than the random reference clusters. It is possible that higher k would give a smaller $GapRatio(k)$ with more convincing statistic. This can be investigated in the future.

To visualize the high-dimensional clusters, I performed principle component analysis on the data. Following Section 3. 4. 1 and 14. 5. 1 in [4], the singular value decomposition of the normalized data T with rows normalized T_i is

$$T = UDV^T, \tag{4}$$

where columns of UD (size $N \times p$) are the principle components, and V (size $p \times p$) is an orthogonal matrix. The first two principle components are the two orthogonal dimensions that capture the variance of the data the most. This 2-D manifold is useful for visualizing the clusters. Since it is hard to conclude if the number of cluster should be 1, 4, or 5 (because they have similar $GapRatio(k)$), I show the 5-means clustering in Fig. 1B, which contains more information. The curve that corresponds to the centers of the clusters are shown in Fig. 1C. As we shall see later with hierarchical clustering, single-mean clustering results in a center that is well represented by G4 in Fig. 1C. The fifth cluster is a single-member class that belongs to G2 if one uses four clusters.

The centers resulting from the four-means clustering have an interesting feature. Participants in G1 uses about the same amount of time in the thinking phase from start to finish. Participants in G3 and G4 start slow but end up thinking faster than G1. Participants in G2 starts with the slowest thinking but end up with the fastest thinking. Based on these properties, I introduce G1 as the *non-adaptive* class (because they do the job without learning more), G3 and G4 as the *progressive* class (because they make consistent progress throughout), and G2 as the *dark-horse* class (because they start very slow but end up very fast). The fifth cluster center in the inset of Fig. 1C is also interesting. For the first half, it behaves like a *dark horse*. At some point, it suddenly slows down a lot, then suddenly regains even more speed. I introduce this class as the *break-through* class.

III. HIERARCHICAL CLUSTERING OF LEARNING BEHAVIOUR

Hierarchical clustering is appealing for its visual representation of the data. The agglomerative (bottom-up) approach starts with the N objects in N clusters, merges the two most similar clusters iteratively (decreasing the number of clusters one at a time) until a single cluster is left. This approach exhibits a monotonicity property: the dissimilarity between the merged clusters increases as the iteration proceeds. The clustering can be displayed as a dendrogram (Fig. 2A), where nodes represent clusters and the height of each node corresponds to the inter-cluster dissimilarity of the node's two daughter clusters (see Section 14. 3. 12 in [4]). With the dendrogram, one can trace through the the hierarchy of clusters and gain insights into the structure of the data.

In more detail, the dissimilarity measure used is the Euclidean distance. The data is normalized as in Sec. II. The input to the algorithm is a $N \times N$ matrix D where element $d_{ii'}$ is the distance between object T_i and object $T_{i'}$ in T. Since the algorithm merges the two most similar clusters at each step, the results depend crucially on the measure of inter-cluster dissimilarity. The measure of choice is

$$d_{CL}(G, H) = \max_{i \in G,\, i' \in H} d_{ii'}, \tag{5}$$

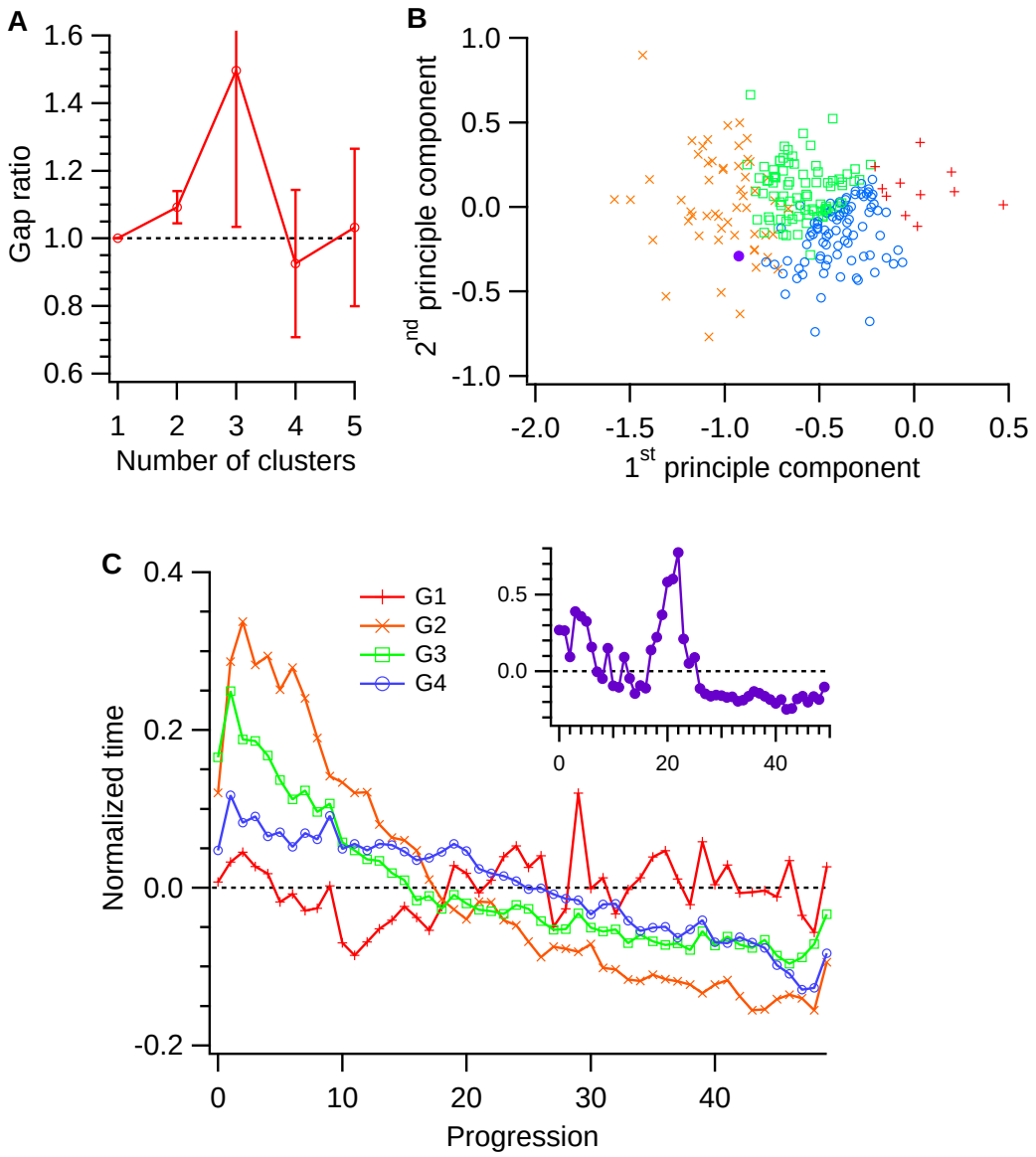

FIG. 1: **A**. The GapRatio test. Error bars are standard deviations of W_k^* calculated from 5 realizations of the reference data. Markers represent the mean of $GapRatio(k)$. The Gap Ratio test predicts four clusters. **B**. Clusters visualized on the first and second principle components of the observed data. Clusters are obtained by the K-means clustering algorithm. Different markers represent different classes. The fifth cluster with a single member (solid circle) is a member of the $G2$ cluster (orange $\times$) if only 4 clusters are allowed. **C**. Cluster centers of the clusters shown in **B**. The y-axis is the normalized trial time — trial time sequence T_c transformed by $[T_c - \bar{T}_c]/[\max(T_c) - \min(T_c)]$. The x-axis is the normalized dimension p with p =50 (see first paragraph in Sec. II point 2 and 3). The inset shows the center of the fifth cluster. Note that the inset's y-axis has a larger range.

where $G(i)$ and $H(i')$ are indexing systems like $C(i)$ [4]. Agglomerative clustering based on d_{CL} is called complete linkage clustering. The use of most dissimilar members (furtherest neighbor) as the criteria for cluster merging results in a phenomenon where members across clusters can be much more similar than members within the cluster. Other choices of measure include the nearest-neighbor measure (change max in Eq. 5 to min) and average measure (change max in Eq. 5 to the the average of all the members in G and H). For the data at hand, out of the three methods, only the complete linkage clustering produces clusters with reasonable sizes. The other two measures produce K clusters with $K-1$ of them being single-member clusters up to K =8. This is the reason for my choosing of complete linkage.

The dendrogram resulting from complete linkage clustering is shown in Fig. 2. I coded for these in IGOR Pro 6.03. The little graphs in Fig. 2 are cluster centers computed by averaging over all the members in the corresponding

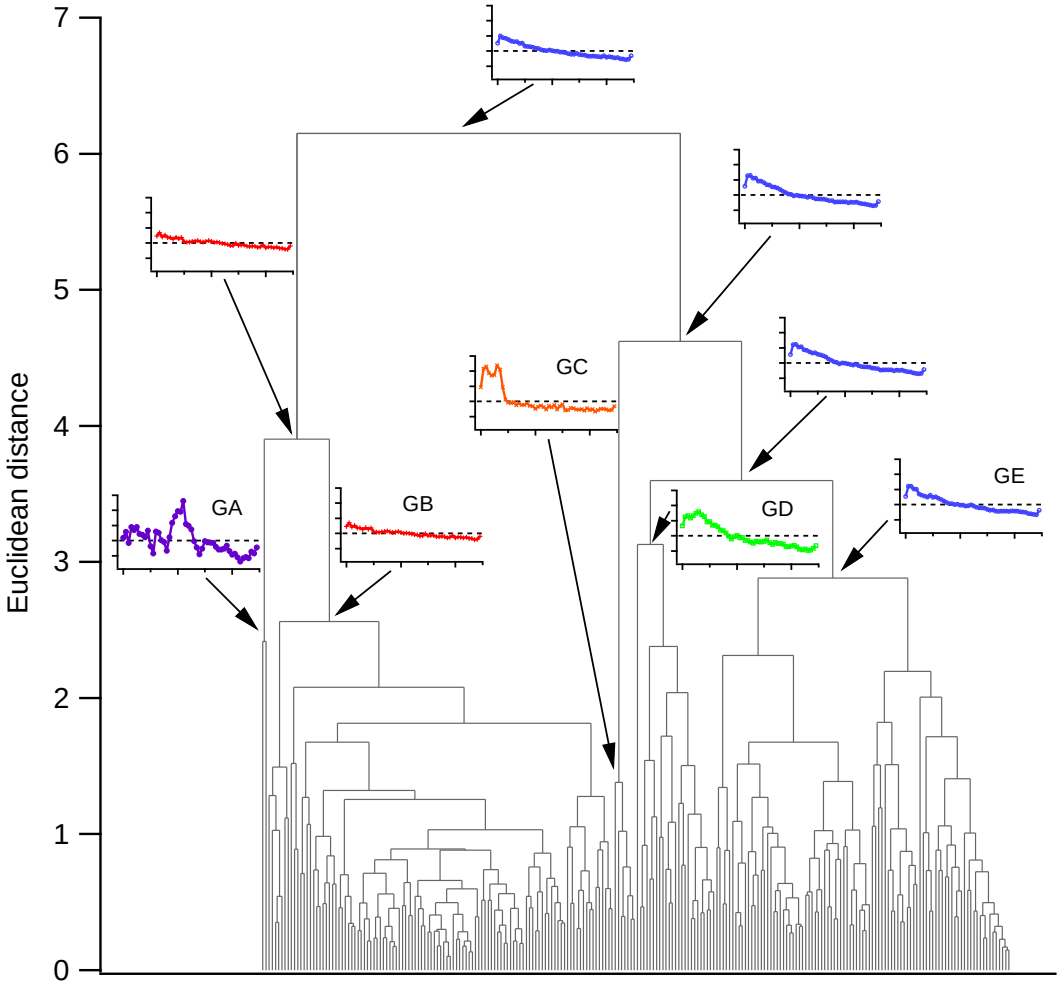

FIG. 2: Dendrogram of complete linkage clustering of T. Each node is a cluster center. The height of the node denotes the hierarchy of the cluster (higher nodes have more subclusters) and is computed with Eq. 5. The little graphs, with arrows pointing to nodes, are cluster centers like Fig. 1C. Cutting horizontally across Euclidean distance ≈ 3.3, one gets the 5 cluster centers labeled GA, GB, GC, GD, and GE.

clusters. The arrows show which graph corresponds to which node. Cutting horizontally across the dendrogram at any level, the cut verticle lines lead to nodes that represent the most prominent clusters at that level. For example, cutting horizontally across Fig. 2 at Euclidean distance = 4, one touches the three verticle lines that lead to the three most prominent clusters. Cutting across at 3.3, one gets the 5 most prominent clusters, whose centers are shown in Fig. 3A. Figure 3B shows the 5 clusters on the first and second principle components.

Comparing Fig. 2B with Fig. 1B, one sees that hierarchical clustering and K-means clustering produce different clusters. However, the two sets of clusters are not entirely different. Comparing Fig. 2A with Fig. 1C, one can still intuitively identify these cluster centers with the learner classes introduced at the end of Sec. II. GA is undoubtedly in the *break-through* class. In fact, one of the two members in the cluster is the fifth cluster center resulting from the 5-means clustering. GC is similar to G4 and should be in the *progressive* class. GD and GE are similar to G2 and can be classified as *black horses*. GB shares the features from both the *black-horse* and the *progressive* class.

Applying the GapRatio test described in Sec. II to hierarchical clustering, I found that the number of clusters is 2 with certainty ($GapRatio(2) = 0.1$). The two cluster centers are well represented by GB and GE. Inspired by this result, a better classification scheme may be to introduce two classes: *slow progressive* and *fast progressive* with

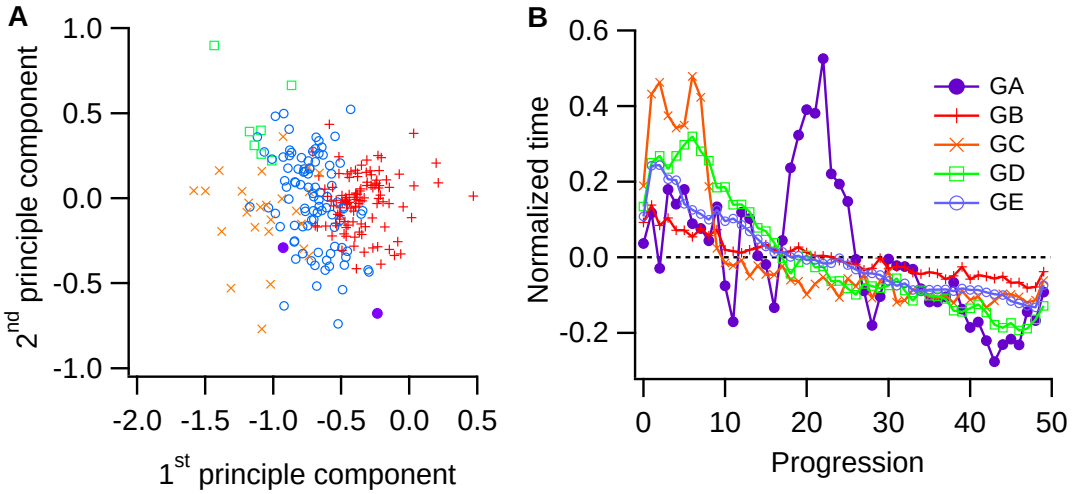

FIG. 3: **A**. The cluster centers of GA, GB, GC, GD, and GE corresponding to Fig. 2. **B**. The five clusters visualized on the first and second principle components.

non-adaptive and *break-through* being subclasses of *slow progressive* and *black-horse* a subclass of *fast progressive*.

IV. K-MEDOIDS CLUSTERING OF LEARNING MOTIF

As mentioned in Sec. I, the idea here is to find motifs in the sequences of location symbols as special learning strategies adopted by participants. Then, cluster the motifs and identify clusters. Before the motif finding, I insert another symbol **z** into the sequence whenever the participant finished the thinking phase for a trial. There are now five possible symbols in the thinking phase: **a**, **b**, **c**, **d** for locations and **z** for going to the next trial.

I used a brute-force approach to find the motifs [7]. Choosing the length L for the motif, the algorithm uses all possible subsequences of length L as templates, slides each template through the entire sequence, and records the number of matches [7]. The motif recorded is the template that has the most number of matches. In my case, a match means perfect symbol matching. One can also allow M number of mismatches. The data to be clustered then becomes a $N \times L$ matrix S with the i^{th} row being the motif that corresponds to participant i. For this part, I chose L =6, which is long enough to show patterns but not too long to be improbable.

The K-means clustering algorithm does not work on categorical data, as there is no good way to compute an averaged categorical center. To deal with this, I used the K-medoids clustering algorithm which replaces each center with the member that minimizes the its own cluster's within-cluster dispersion. For later usage in the context of K-medoids, the word center refers to an existing object chosen to minimize within-cluster dispersion. The new distance measure is the Hamming distance. Similar to K-means clustering, the K-medoids algorithm also depends on the initial choice of centers. To find good cluster centers, I ran the algorithm 1000 times, each time with a set of randomly chosen K objects as initial centers. I took the K most frequent final centers as the resulting cluster centers. The results are:

Number of clusters	Cluster center
1	41
2	41 119
3	41 119 4
4	41 119 4 20
5	41 119 4 45 20

The number under "Cluster center" is the index of the participant. The order of the centers goes from most (left) to least (right) frequent. The motifs of these centers are:

Index	Sequence
41	**bbbbbb**
119	**cbcbcb**
4	**aaaaaa**
20	**cdczcd**
45	**zcbdcb**

The first motif corresponds to participants' gazing at an image feature position consecutively. The second motif corresponds to participants' looking back and forth between two image features. The two most popular centers imply two classes of learning strategies, *staring* and *switching*. The third motif is strange: participants kept on not looking at any of the features. The fourth motif also exhibits the switching pattern; whereas the fifth motif goes around all available image features.

The absolute locations of the eye-tracking data is actually not very important because the images are randomized over different participants. This means that **b** and **c** for different participants might correspond to the same image feature. To capture the invariance, for each pair of motif, I transformed the first motif via **b**→**c**, **c**→**d**, and **d**→**b**, and compute a hamming distance for each transformation. The same is done on the second pair. With the original untransformed distance, each pair of motifs has 7 distances attached to them. The minimum of the 7 is the new invariant distance. Using the invariant distance for K-medoids, I obtained:

Number of clusters	Cluster center
1	41
2	41 4
3	41 4 20
4	41 4 20 45
5	41 4 20 45 116

The new motif is:

Index	Sequence
116	**bbbcbb**

Comparing with the previous result, the motif with pure *switching* is now replaced by a *peaking* strategy, where attention is mostly spent on a single feature and only rarely goes off to other features. In summary, the most prominent class of special learning pattern is *staring*, followed by *switching* (includes partial switching), and the unknown (motif 4). Plotting these clusters on the first and second principle components does not show anything insightful. I thus conclude that the special learning strategy adopted does not correlated with the overall learning behavior.

V. CONCLUSION

My goal is to identify learner classes based on a set of eye-movement time series. These time series quantify the participants' gaze while they performed a categorization task consisted of multiple trials. After processing the data, I obtained relatively short sequences of times that represent the overall learning behavior. Clustering these sequences of times via K-means and hierarchical clustering, I found two prominent learner classes: *slow progressive* and *fast progressive*. Typical learners of both classes become faster at the task as the experiment proceeds. The distinction is in that a typical *fast progressive* learner starts slower but ends up faster than a *slow progressive* learner. The former also has two subclasses: 1)the *non-adaptive* class with constant thinking speed throughout and 2) the *break-through* class with a sudden de-acceleration and re-acceleration in thinking speed midway through the experiment. The latter class has a subclass *black-horse* that starts extremely slow but end up being the fastest. These are the classes for overall learning behavior.

The eye gaze locations are processed into sequences of location symbols like DNA sequences. I searched for motifs in these sequences to represent special learning patterns. Using K-medoids to cluster the motifs, I found that most prominent way of learning adopted is *staring*, followed by *switching* (looking back and forth at two things). Not gazing at anything particular is also an outstanding motif. I found no correlation between these motif-inspired classes and the time-inspired classes mentioned above.

APPENDIX A: KOLMOGOROV-COMPLEXITY CLUSTERING

Something interesting that I ran out of time to do is clustering based on Kolmogorov complexity [8, 9]. The Kolmogorov complexity, defined as

$$d_k(x,y) = \frac{K(x|y) + K(y|x)}{K(xy)}, \tag{A1}$$

is actually a metric [9]. The complexity $K(x)$ is the fewest possible bits needed to code data x. The conditional complexity $K(y|x)$ is the number of bits needed to code data y using the coding strategy that is optimized for compressing data x. The conditional complexity gives information about how similar data x and y are. Even though the compression proposed by Kolmogorov is not computable, it is shown that one can estimate $K(x)$ by $C(x)$ where $C(x)$ is the number of bits needed to code data x using any of the compressors we have at hand (e.g. gzip). Further, $K(y|x) = K(xy) - K(x)$ and $C(xy)$ and $C(x)$ are both computable. The advantage of this metric over Eucledian distance, Hamming distance, and others is that it works for any pair of data, regardless of their dimensions and types. Using this metric, I can compare the sequences of location symbols (very different lengths) and thus cluster the entire data set without much processing. It is interesting to see what comes out of this.

---

[1] M. R. Blair, M. R. Watson, and K. M. Meier. Errors, efficiency, and the interplay between attention and categorical learning. *Congnition* **112**, 330–336 (2009).

[2] Communication with Dr. Mark Blair.

[3] C. Goutte, P. Toft, E. Rostrup, F. A. Nielsen, and L. K. Hansen. On clustering fMRI time series. *NeuroImage* **9**, 298–310 (1999).

[4] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* 2nd Ed. Springer, NY, 2009.

[5] WaveMetrics Inc. `http://www.wavemetrics.com/` (2010).

[6] R. Tibshirani, G. Walther, and T. Hastie. Estimating the number of clusters in a data set via the gap statistic. *J. R. Statist. Soc. B* **63** Part 2, 411–423 (2001).

[7] J. Lin, E. Keogh, S. Lonardi, P. Patel. Finding motifs in time series. In proceedings *SIGKDD'02* July 23–26 (2002). `http://www.cs.ucr.edu/~stelo/papers/motif_KDD.pdf`.

[8] C. A. Ratanamahatana, E. Keogh, A. J. Bagnall, S. Lonardi. A novel bit level time series representation with implications for similarity search and clustering. *Advances in knowledge discovering and data mining* **3518** 771–777 (2005).

[9] M. Li, X. Chen, X. Li, B. Ma, P. M. B. Vitanyi. The similarity metric. Proceedings of the 14th annual ACM-SIAM symposium on Discrete algorithms. 863–872 (2003).