Learning Time Series Classification by Clustering on the Manifold

Eugene Shvarts

September 27, 2013

A work in progress! Project is executed under the guidance of Prof. Naoki Saito, in collaboration with Prof. Hrushikesh Mhaskar (Claremont Graduate University), and supported through VIGRE stipend DMS0636297.

The ambitious project

Explore methods of change detection to develop machine learning applications with potential use on large unstructured datasets.

- Quantitative definition of change between datasets as a foundation for metalearning and semi-supervised learning.
- Natural place to start – time series models accommodate well-known relationships between data instances.
- Survey the area, compare many methods on several datasets.

The reality

- Available ideas require searching a large parameter space and expensive computations.
- Settled on one time-series problem with prior results available for comparison.
- Preliminary theoretical results for performance guarantees.
- Performed pre-processing on a medical dataset, and wrote an algorithm for change detection procedure (to be detailed in another presentation).

The reality

- Available ideas require searching a large parameter space and expensive computations.
- Settled on one time-series problem with prior results available for comparison.
- Preliminary theoretical results for performance guarantees.
- Performed pre-processing on a medical dataset, and wrote an algorithm for change detection procedure (to be detailed in another presentation).

Here we primarily explore relationships between the parameters, and ideas for improving prediction performance.

The lips dataset

- 50 greyscale (scalar) videos of 55x70 pixel frames, with varying length. Each visualizes a pair of lips speaking a number between 1-5.
- Naively, each frame is a 3850-element feature vector, and they evolve according to the function of time which generates the video.
- Need to capture good features for the image ...

The lips dataset

- 50 greyscale (scalar) videos of 55x70 pixel frames, with varying length. Each visualizes a pair of lips speaking a number between 1-5.
- Naively, each frame is a 3850-element feature vector, and they evolve according to the function of time which generates the video.
- Need to capture good features for the image ... but wait, what do we do once we have these features?

References and techniques

- Need some form of **ensemble classification** to treat the collections of frames in each video as one item. Lips dataset previously analyzed in "Signal Ensemble Classification using Low-Dimensional Embeddings and Earth Mover's Distance", Lieu & Saito.

References and techniques

- Need some form of **ensemble classification** to treat the collections of frames in each video as one item. Lips dataset previously analyzed in "Signal Ensemble Classification using Low-Dimensional Embeddings and Earth Mover's Distance", Lieu & Saito.
- ARMA model presumes a hidden state vector evolving by a linear operator, which evolves the time series by another linear operator. Derivation of closed-form approximation for ARMA parameters from observed features in "Dynamic Textures", Doretto et al.
- In "Statistical Computations on Grassmann and Stiefel Manifolds for Image and Video-Based Recognition", Chellappa et al. use the ARMA parameters to express the expected value of the features as a low-dimensional subspace of ambient Euclidean space.

Frame features

- A good selection of features will determine the image as a function of the features, in a sufficiently sparse manner so as not to replicate inherent noise.
- In all cases, the mean and variance of the pixel values were taken as two features of each frame. Then, I tried two different approaches I was recommended:

Frame features

- A good selection of features will determine the image as a function of the features, in a sufficiently sparse manner so as not to replicate inherent noise.
- In all cases, the mean and variance of the pixel values were taken as two features of each frame. Then, I tried two different approaches I was recommended:
- Taking the principal components of each frame by treating either the rows or columns as probability distributions. This yields min(frame dim) $= 55$ more features.

Frame features

- Taking the principal components of each frame by treating either the rows or columns as probability distributions. This yields min(frame dim) $= 55$ more features.
- Taking the discrete cosine transform of the image. This produces prod(frame dim) $= 3850$ features, which is unhelpful. However, linear truncation provides a good feature representation.

Frame features

- Taking the principal components of each frame by treating either the rows or columns as probability distributions. This yields min(frame dim) $= 55$ more features.
- Taking the discrete cosine transform of the image. This produces prod(frame dim) $= 3850$ features, which is unhelpful. However, linear truncation provides a good feature representation.
- Nonlinear truncation by keeping only several of the largest magnitude components proved to work as well.

Parameters of interest

- Dimension of the hidden state vector. Chellappa recommends 5-10 for large-dimensinoal (feature) data.
- Truncation of the observability operator. The expectation of the features $f(t)$ given initial condition $z(0) = z_0$ and the ARMA model

$$f(t) = Cz(t) + w(t)$$
$$z(t+1) = Az(t) + v(t) \quad ,$$

with v, w normally distributed noise is given by the observability operator

$$E \begin{bmatrix} f(0) \\ f(1) \\ f(2) \\ \vdots \end{bmatrix} = \begin{bmatrix} C \\ CA \\ CA^2 \\ \vdots \end{bmatrix} z_0 \equiv O_\infty z_0 \quad .$$

Parameters of interest

- DCT coefficient cutoff. The linear method requires two parameters; the horizontal frequency cutoff and the vertical. The nonlinear method requires one parameter; the number of highest-magnitude coefficients to retain.
- Machine learning consideration. Should the data be trained in a uniformly random way, or should training data be spread evenly between classes? How much training data is the right amount?
- Each analysis method, as shown later, introduces its own parameters for adjusting sensitivity e.g., to cluster size.

On the Grassman manifold

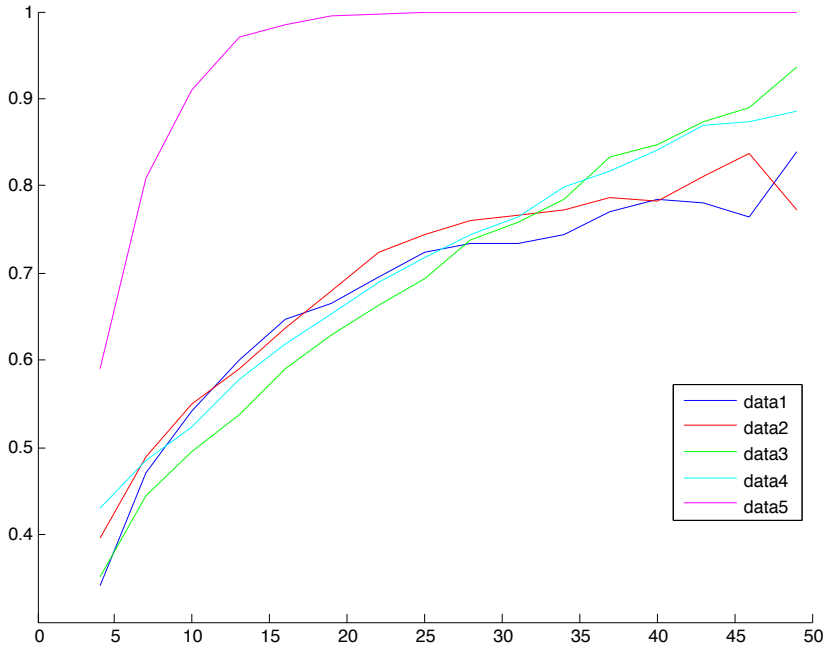
- Truncating O_∞ at the m^{th} term when the hidden variable has dimension d and the features are p-dimensional produces an $mp \times d$ matrix, interpreted as a d-dimensional subspace of mpd-dimensional space. This is a single point on the Grassman manifold $\mathcal{G}(d, mpd)$.
- Because we have landed on a known manifold, we may use its geodesic metric to compare points (which are entire videos), and so enlist any of our favorite point-cloud analysis and learning techniques.

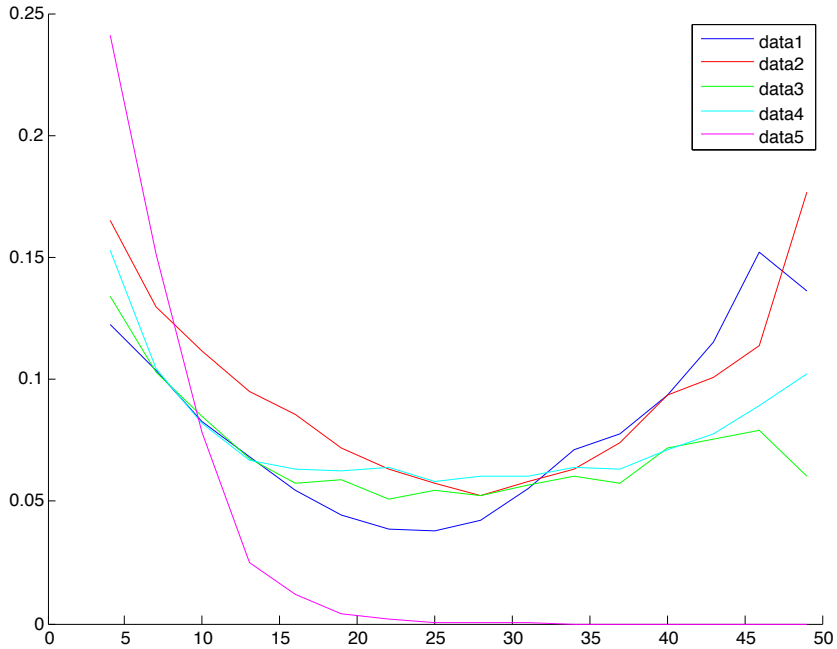
Methods and discussion

- First attempt was a straightforward k-nearest neighbor comparison, equipping the geodesic metric. Because 'points' here are subspaces, there was a lot of wheel-reinventing involved in writing code for each of the analysis methods; this was a setback.
- Early testing quickly showed that $k = 1$ is the best parameter choice (against PCA features), so this is essentially a nearest-neighbor classification. Larger k values rapidly eroded the prediction performance. Strangely, $d = m = 1$ were the only choices which yielded satisfactory results for PCA features as well.

PCA features, KNN classifier. d=1 m=1 k=1

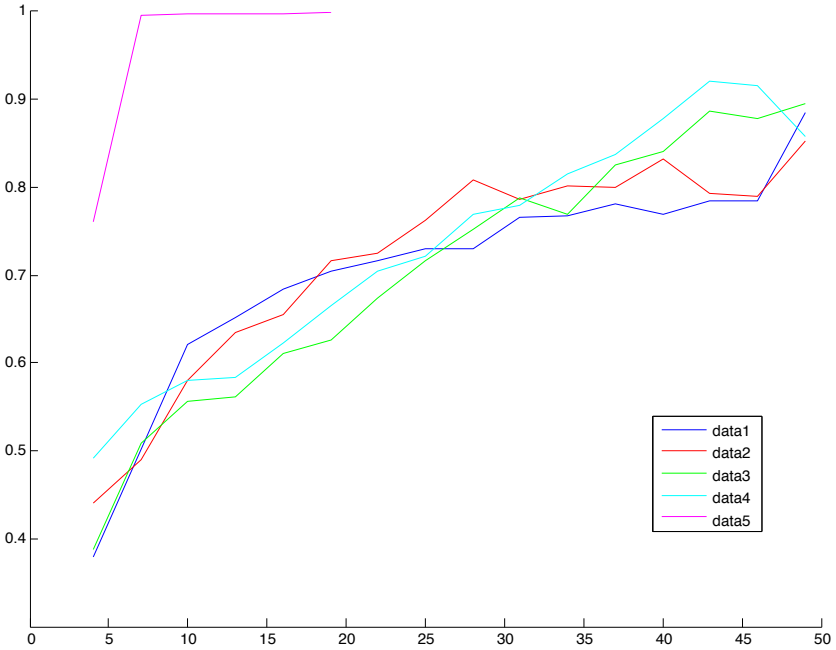
The mean and variance of prediction accuracy over 1000 trials against random training data.

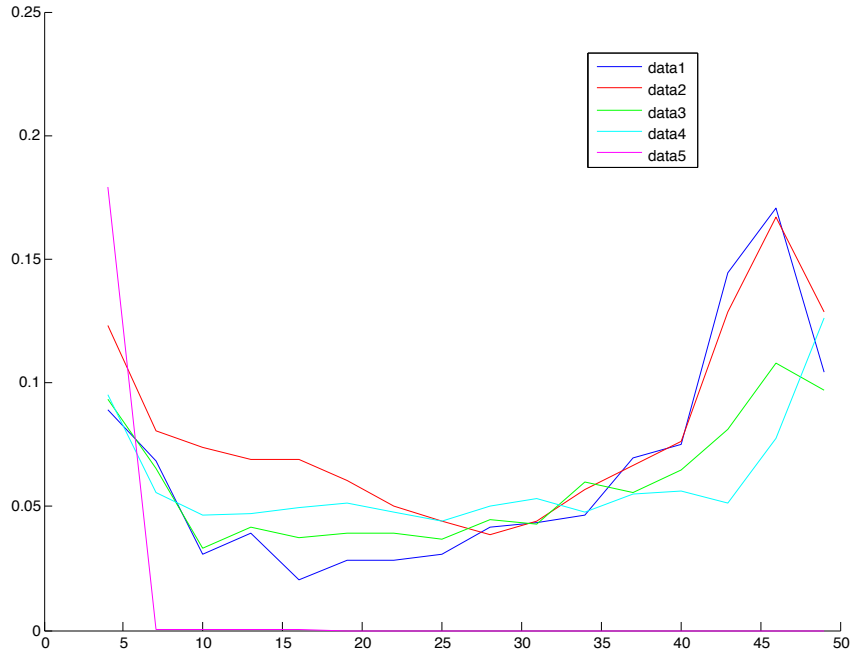




PCA features, KNN classifier. d=1 m=1 k=1

The mean and variance of prediction accuracy over 200 trials against equally distributed training data.



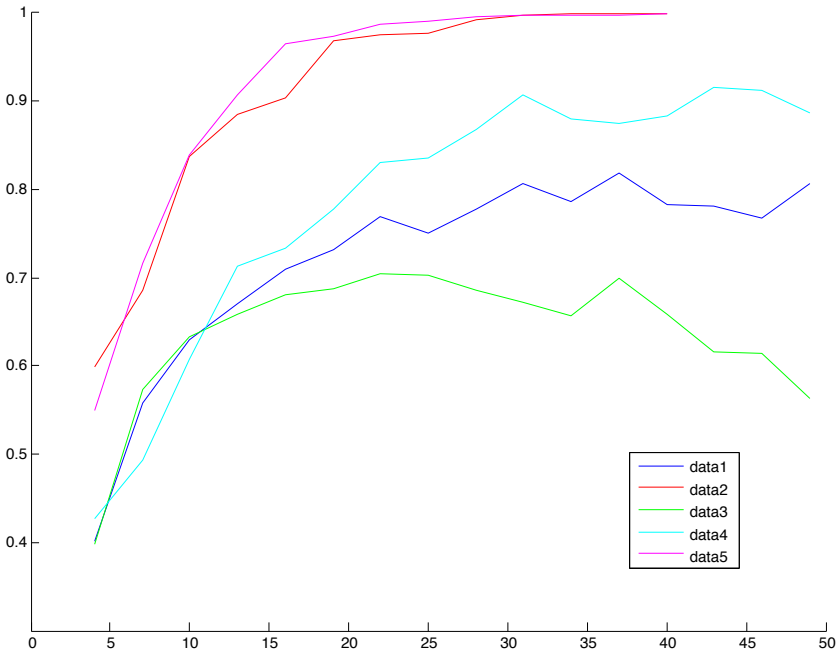


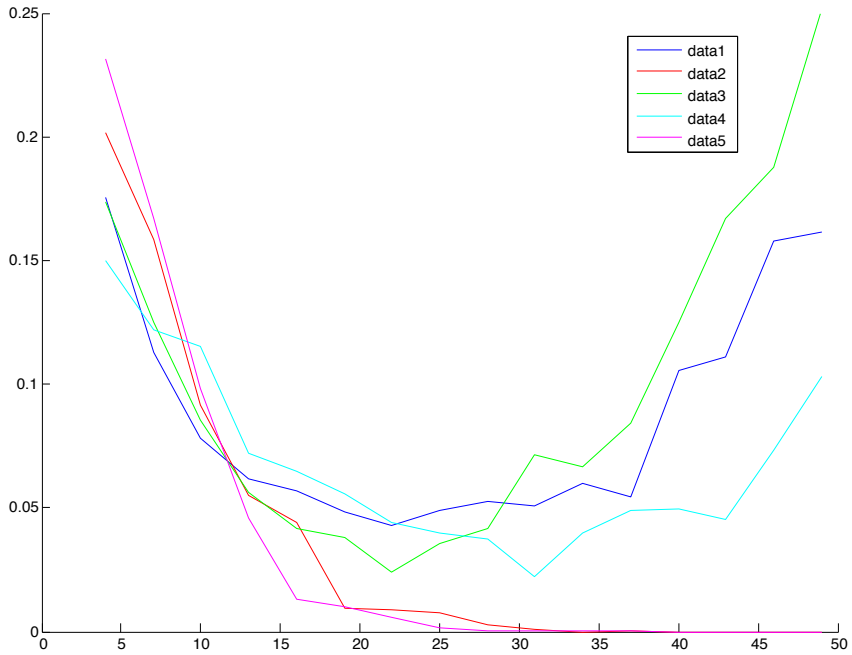
Methods and discussion

- Next, I tried DCT coefficients. Using an elbow method, I tried equal truncations in both dimensions and found that cut = [5,5] generated the best results (with 25 features). Observing the variance below for high training data, an underlying theme in this experiment is that using richer features doesn't even yield diminishing returns – the results are worse! However, the experiments at this stage all use the KNN classifier, so this phenomenon may be model-dependent.

DCT w/ linear cut, KNN classifier. d=1 m=1 cut=[5 5] k=1

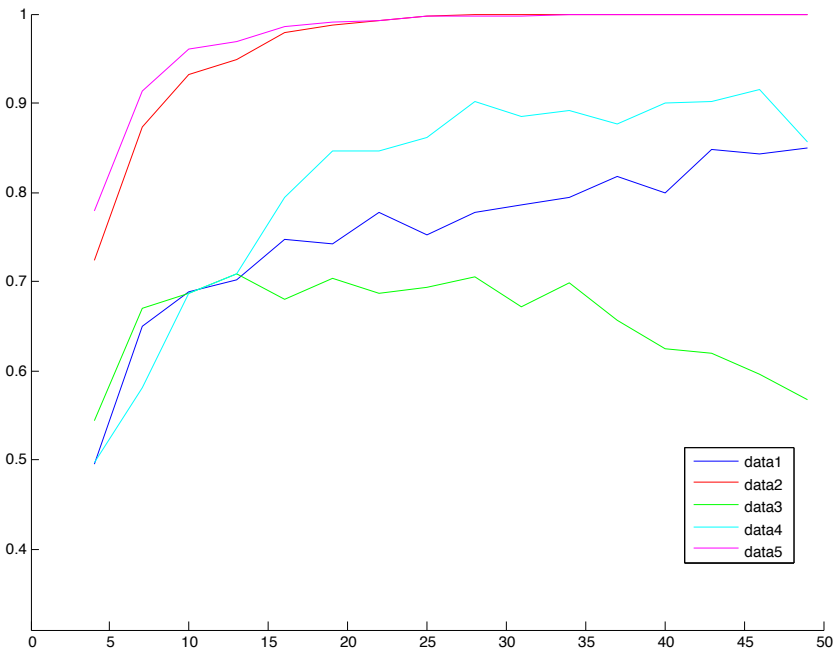
The mean and variance of prediction accuracy over 200 trials against random training data.

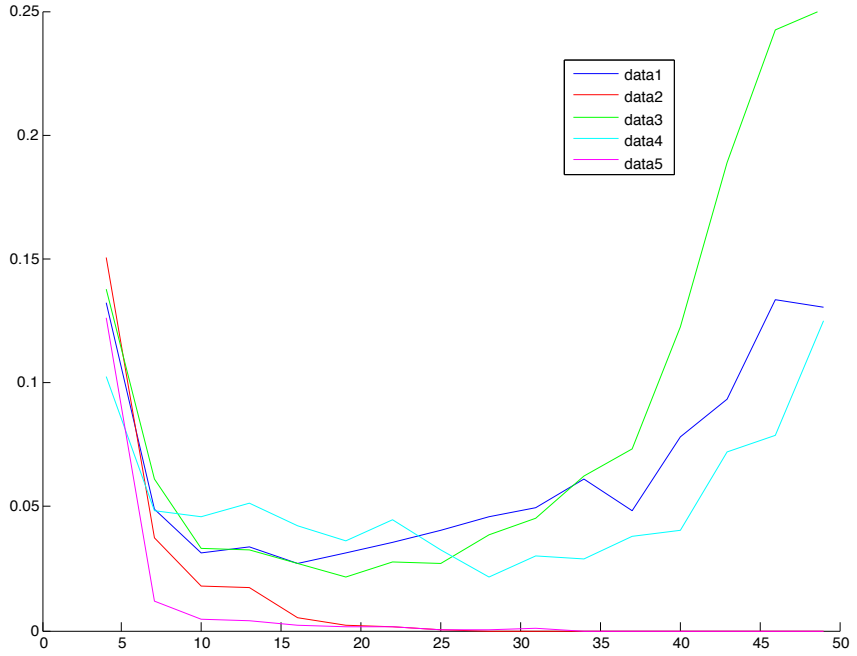




DCT w/ linear cut, KNN classifier. d=1 m=1 cut=[5 5] k=1

The mean and variance of prediction accuracy over 200 trials against equally distributed training data.



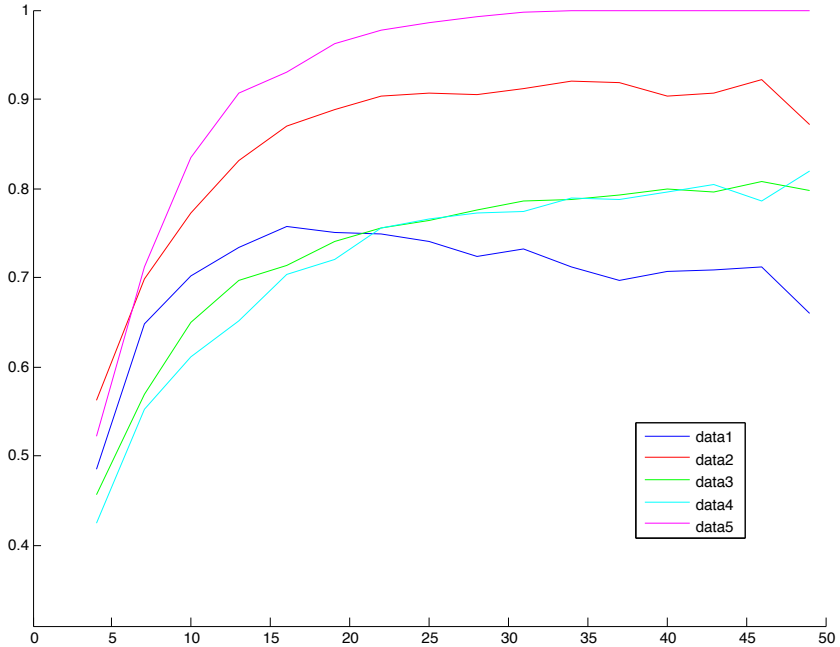


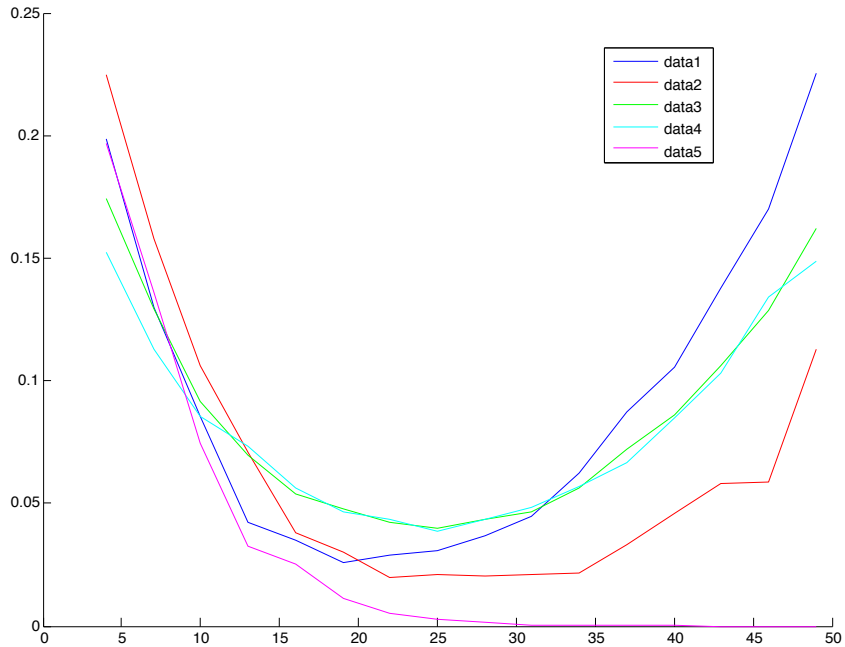
Methods and discussion

- The nonlinear method I used located the largest-magnitude DCT coefficients of each frame, and kept a sparse vector with their location and value. In computing the observability matrix however, a sparse SVD is necessary, again creating a full matrix. Inspection showed that typically the eigenvector matrix itself was sparse as well – eliminating all coordinates with magnitude below some error tolerance (I used 10^{-12}) typically leaves 10-20 features, significantly fewer than the other methods, speeding up computation without compromising accuracy. Elbow methods suggest that cut $= 6$ is optimal.

DCT w/ nonlinear cut, KNN classifier. d=1 m=1 cut=6 k=1

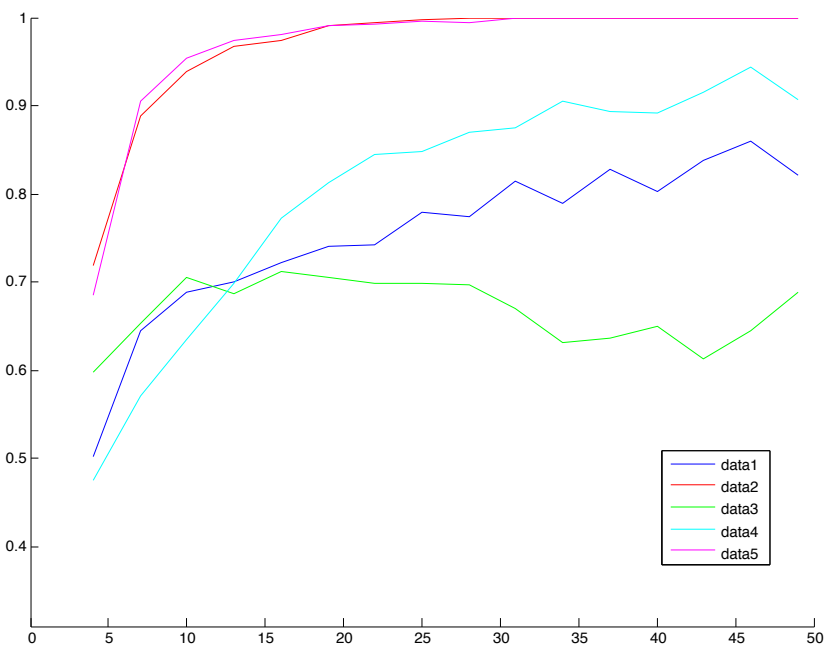
The mean and variance of prediction accuracy over 1000 trials against random training data.

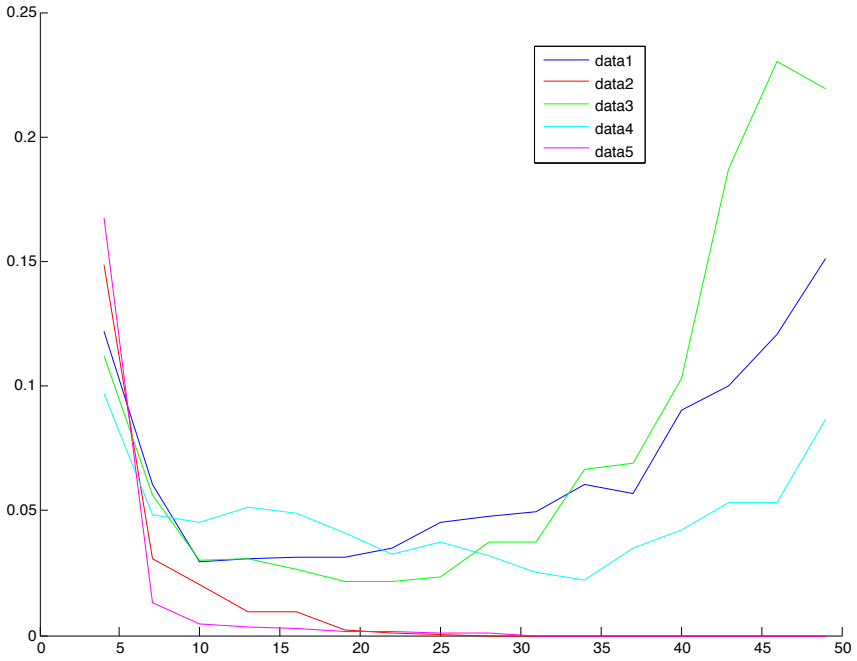




DCT w/ nonlinear cut, KNN classifier. d=1 m=1 cut=6 k=1

The mean and variance of prediction accuracy over 200 trials against equally distributed training data.





Conclusions

- In all cases, class 5 is precisely classified, with accuracy and variance converging to 1 and 0 respectively at various rates depending on the method, as the training data size increases. The DCT methods partition class 2 effectively as well.
- In each case, using equally distributed training data noticably reduces the variance in accuracy. However, the accuracy does not appear to be appreciably affected, suggesting that occasionally missing an entire class is not the reason for the strange behavior of the variance for large training data size.

Conclusions

- There appears to be a saturation in accuracy for each class; especially around 70-80% for classes 1, 3, and 4. This may be a fault of the KNN method in analyzing this particular dataset.
- The U-shape of the variance is entirely a mystery to me, as it seems to defy the logic of cross-validation. The optimal amount of training data to use for minimal variance (and seemingly maximum accuracy) lies between 22-25 of the 50 videos. Typical average accuracy of prediction for these parameters over all classes is 75-85%.

Ongoing and future work

- It is critical to expand to more varied classification methods. The first two on the mind are diffusion geometry (by approximating the classification function on the Grassman manifold), and Earth Mover's distance (EMD). For the former, I have working code (thanks to HNM), and for the latter I expect to have the same in the near future (thanks to NS).
- Using a diffusion map requires setting two further parameters – the continuous epsilon (neighborhood size) and discrete degree (for quadrature). A next task is automating that procedure.

Ongoing and future work

- The video data is certainly not dense on the entire Grassman manifold, especially for larger m, d. So, with HNM we are extending the Belkin-Niyogi result for convergence of the graph Laplacian to the manifold Laplacian in the case of mapping to a sub-manifold.
- The nonlinear DCT can be further improved by taking those coefficients for which the *trigonometric frame coefficients* have largest magnitude, as these describe the local features of the image. This was recommended by HNM.
- It is still necessary to confirm the success of the method on a different data set.

Ongoing and future work

- The behavior of the m, d parameters hasn't been fully explored yet – it appears in some cases (especially with DCT features) that increasing m, d leads to comparably accurate results. We would like d to remain relatively small (5-10), but it is not clear why increasing m would reduce performance.
- Should the ARMA model not provide an adequate description for the time series evolution, one potential direction involves using nonlinear operators (NARMA) and a function approximation model to generate the observability operator and the appropriate Grassman manifold.