Iterative face image feature extraction with Generalized Hebbian Algorithm and a Sanger-like BCM rule

Clayton Aldern (Clayton_Aldern@brown.edu)
Tyler Benster (Tyler_Benster@brown.edu)
Carl Olsson (Carl_Olsson@brown.edu)

Brown University Computational Neuroscience Final Project, Spring 2013
Professor Elie Bienenstock

Abstract—**Principal component analysis extracts dimensions of the greatest variance in a dataset and allows for dimensionality reduction via projection onto these dimensions. In the case of high-dimensional data, this process is computationally intensive, as it traditionally requires solving the eigenvalue problem for a very large covariance matrix. Recently, neural network-inspired learning rules have provided iterative methods for encoding features of dataset as synaptic weights. Here, we present the results of the implementation of the Generalized Hebbian Algorithm on a dataset of facial images and experiment with a Sanger's rule–like extension of the BCM rule, as well. We show GHA to quickly and iteratively converge to the first *p* principal components in order of descending eigenvalue and the modified BCM rule to converge to more complex features.**

Keywords—**feature extraction; image processing; principal component analysis; Generalized Hebbian Algorithm; Oja's rule; Sanger's rule; BCM rule**

Introduction

Principal component analysis

In image processing, it is often desirable to reduce the dimensionality of datasets. Such dimensionality reduction is necessary for interpreting quantitative differences between images and—if constituent algorithms are biologically inspired—speculating on neural mechanisms for encoding such data. Principal component analysis (PCA) is a linear orthogonal transform that provides a direct method for dimensionality reduction, in which a dataset is projected onto the first *p* eigenvectors of the covariance matrix of the data (zero mean correlation matrix). These projected dimensions are referred to as principal components. Specifically, for a given dataset of dimensionality D_1, PCA finds a subspace of dimensionality D_2 where $D_2 < D_1$ and, importantly, each principal dimension is orthogonal to all others. While many techniques for PCA exist, projection onto this linear space generally maximizes the variance of the projected data.

In image processing, PCA can be used for image compression, reconstruction, classification, and recognition problems (e.g, Wang, 2012).

Hebbian learning

Practically, it is both desirable and biologically plausible to evolve such an algorithm iteratively over time. Such a paradigm allows for fast feature learning, which is useful, for example, in facial recognition implementations (Ebied et al., 2012). Hebbian learning (Hebb, 1949) provides a neural framework for understand such a process. In standard Hebbian learning, a single time-dependent output neuron *y(t)* dynamically modifies its firing rate at time *t*

$$y(t) = \boldsymbol{w}^T(t)\boldsymbol{x}_t,$$

using

$$\boldsymbol{w}(t+1) = \boldsymbol{w}(t) + \eta y(t)\boldsymbol{x}_t,$$

where η is the learning rate, $\mathbf{w}$ is the synaptic weight vector from input to output, and $\mathbf{x}_t$ is an input vector at time *t.* Without bound, this paradigm causes $\mathbf{w}$ to increase to infinity. Oja (Oja, 1982) modified the Hebb rule by adding a multiplicative weight-decay term to the weight update rule:

$$\boldsymbol{w}(t+1) = \boldsymbol{w}(t) + \eta y(t)(\boldsymbol{x}_t - y(t)\boldsymbol{w}(t)),$$

or, with rearrangement,

$$\boldsymbol{w}(t+1) = \boldsymbol{w}(t) + \eta y(t)\boldsymbol{x}_t - \eta y^2(t)\boldsymbol{w}(t)$$

which normalizes the weight vector and causes it to converge to the principal eigenvector of the covariance matrix. Some implementations may require different learning rates for each term.

Oja's rule, however, only works for a single linear output neuron. In order to learn additional components, more outputs are necessary. In 1989, Sanger introduced the Generalized Hebbian Algorithm (GHA), which accommodates multiple outputs (Sanger, 1989). This approach is also known as Sanger's rule,

$$\boldsymbol{w}_i(t+1) = \boldsymbol{w}_i(t) + \eta_i y_j(t)(\boldsymbol{x}_t - \sum_{j=1}^{i} \boldsymbol{w}_j(t) y_i(t))$$

where i = 1, 2, …, p. The GHA learns the first p eigenvectors of the covariance matrix in order of decreasing eigenvalue. It can be shown that Sanger's rule is equivalent to performing Oja's rule for each component while normalizing using Graham-Schmidt orthonormalization (Qiu et al., 2012).

Here, we experiment with GHA and related techniques, including the Bienenstock-Cooper-Munro rule, in the case of two-dimensional inputs as well as in image space using a facial image database.

Methods

All computation was done using the MATLAB software package. A brief outline of the methodology follows:

A. Load/Preprocess images
    a. Convert to grayscale, resize, and vectorize
    b. Tile input images, permute
    c. Zero-mean the input images
B. Calculate components
    a. Run GHA algorithm
    b. Run BCM algorithm

Image loading/preprocessing

We experimented with the Generalized Hebbian Algorithm in the case of two dimensions and 10000 dimensions. In the two dimensional case, input vectors were random inputs iteratively drawn and presented from a covariance matrix. Higher dimensional data, in the form of facial images, came from the FGnet Aging Data Base (Face and Gesture Working Group, 2004), a database of 1002 grayscale and color face images at various lighting conditions and ages. We imported the faces into MATLAB and center-cropped the faces to 100 x 100 grayscale pixels. We then vectorized the images and concatenated the vectors into a final input matrix of size 10000 x 1002. Additionally, it was necessary to tile and permute the input images. This gave GHA more time to converge to the principle components.

We also zero-meaned the input by subtracting the "mean face" from each input image vector. Sanger's rule will not learn the principle components of an input without this normalization step.

GHA implementation

GHA was implemented locally using iterative Graham-Schmidt orthonormalization and subtraction of an Oja term. That is, whole images were presented as individual inputs and synaptic weights were computed serially for each image. Each consecutive synaptic weight was computed by performing Oja's rule on a modified input created by subtracting off the projected energy from all previously computed weights. Thus, all weights converged approximately simultaneously.

In running GHA, we used additional parameters to ensure convergence of the algorithm: $\tau_w = 15000$ (synaptic decay), $\alpha = 0.1$ (separate weight of the Oja term), and $\eta = 0.8$ (learning rule, in this case, constant). We attempted to learn the first four principal components.

BCM rule

In our third set of experiments, we attempted to extend the Bienenstock-Cooper-Munro (BCM) rule to higher dimensions and multiple, iterative outputs. The BCM rule is another synaptic modification rule, in which changes in the weight matrix are tracked by a normalizing, post-synaptic activation function $\theta(t)$ that changes sign at a threshold (Bienenstock et al.,

1982). Here, we implemented the higher-dimensional BCM rule as

$$\boldsymbol{w}(t+1) = \boldsymbol{w}(t) + \tanh\big(y(t) - \theta(t)\big)\eta_1 \boldsymbol{x}_t$$

where $\theta(t)$ is modified according to

$$\theta(t+1) = \theta(t) + \eta_2 |\mathrm{y(t)}|^p - \theta(t)$$

where $|\ .\ |$ is the absolute value function and p is a constant, in our simulations equal to 1.5. In order to extend the BCM model to multiple output units, we employed a Sanger-like strategy as above, subtracting the energy from all (i-1) previously computed 'components' when computing the ith 'component'. Component, here, is used relatively, as the BCM rule does not learn principal components.

Control

For the sake of comparison, we ran the built-in MATLAB PCA function on the input matrix, as well (i.e. on the 10000 x 10000 covariance matrix).

Results

In the 2D case, GHA performed as expected, as illustrated in Figure 1.

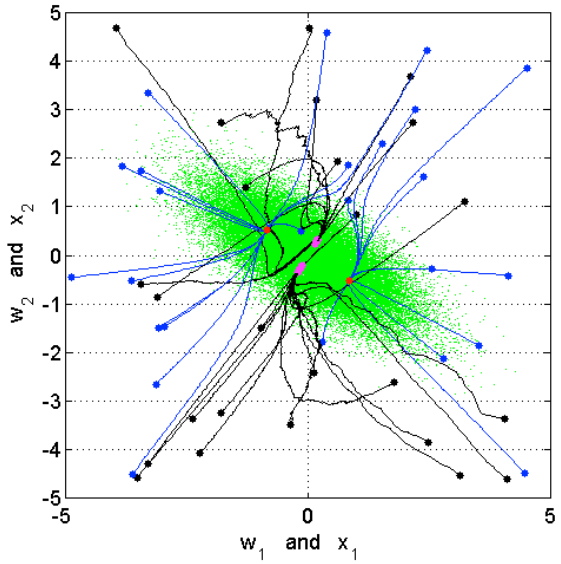

Figure 1. Generalized Hebbian Algorithm with two-dimensional inputs. Green points represent inputs, blue and black points represent random weight initializations for the first and second components, and red and pink points represent final stored weight vectors (equivalent to ± eigenvectors of the covariance matrix).

For our 10000 x 10000 image covariance matrix, PCA took approximately 45 minutes to run. In the case of facial images as inputs, GHA learned the first p eigenvectors of the covariance matrix of the image database in approximately five seconds. A screenshot of video of this convergence can be found in Figure 2. The squared error was best for the first component at approximately 0.01. As expected, the squared error increased with each further component, increasing from approximately 0.01 to 1.

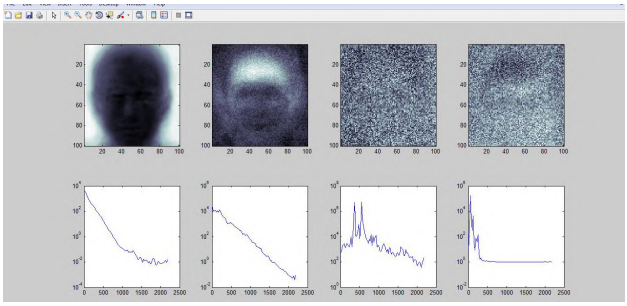

Figure 2. Evolution of Generalized Hebbian Algorithm for four output units. A screenshot of the first four synaptic weigh vectors converging (top row). Bottom: convergence of the squared error between the synaptic weights and the eigenvectors computed in PCA.

Final convergence of GHA is illustrated in Figure 3.

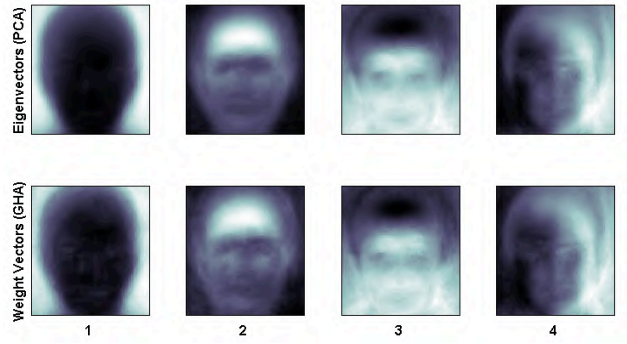

Figure 3. Convergence of Generalized Hebbian Algorithm for four output units. Top: eigenvectors computed via PCA. Bottom: synaptic weights computed via GHA.

The BCM rule presented more complex results. Without mean-centering, as illustrated in Figure 4, the synaptic weights appear to code for feature vectors that are not collinear with the principal components. Implementation of the BCM rule required careful parameter adjustment and, as compared to GHA, more iterations (i.e. image

presentations) in order to converge to potentially relevant features. With mean-centering, as illustrated in Figure 5, the features appear to eventually converge to the first eigenvector, while the first few components failed to converge.

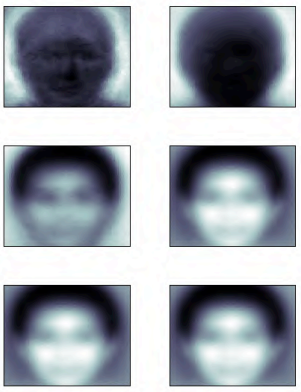

Figure 4. Convergence of BCM rule for six output units without mean-centering. Feature vectors are not collinear with the first six eigenvectors.

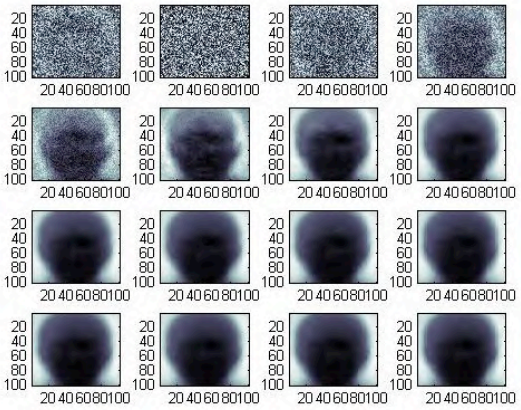

Figure 5. Convergence of BCM rule for sixteen output units with mean-centering. Feature vectors appear to eventually converge to the first eigenvector.

Discussion

Our GHA experiments confirmed several expectations. For constant learning rates, synaptic weights never precisely converged to the proper eigenvectors, though they were nearly collinear, as evidenced by the low mean squared error calculations. In order to ensure total convergence, it was necessary to implement time-dependent learning rates that approached zero over time. Without the time-dependency, weights tended to 'quiver' around a given eigenvector, as each presented input affected the weights in a constant manner. It was also necessary to either duplicate our input matrix several times or oversample a single input matrix in order to reach an appropriate number of iterations for convergence. In BCM implementation, this latter point was further exacerbated.

Challenges in implementation pertained to extending our model past two dimensions. Images sit in high-dimensional space (in this case, 10000-D space), and it was necessary to adjust parameters—especially synaptic decay terms—accordingly. Extending the BCM rule to image space, in particular, required squeezing the squared term through a hyperbolic tangent function in order to maintain relative stability. Our BCM implementation is highly sensitive to small changes in parameters, and further investigation is necessary to assure a more robust implementation.

The BCM results do not lend themselves to easy interpretation. While the first few weights appear to code for complex features, the sixth weight seems to code for the mean face. As the inputs were not zero-meaned in the case of the first BCM implementation, it was possible that we would achieve different results with zero-meaned data. However, in zero-meaning our data, we achieved the results in Figure 5, in which the latter features appeared to converge to the first eigenvector, while the first few 'components' failed to converge with the present number of iterations. It is difficult to speculate as to why this might be the case, and more investigation is necessary to determine the appropriate functional relationships therein. Future work in extending the BCM model to higher dimensions and multiple outputs could provide new insight into biologically plausible encoding of such stimuli.

Acknowledgements

The authors wish to thank Dr. Elie Bienenstock for his valuable input in implementing the BCM rule for higher dimensions.

References

Bienenstock, E. L., Cooper, L. N., & Munro, P. W. (1982). Theory for the development of neuron selectivity: orientation specificity and binocular

interaction in visual cortex. The Journal of Neuroscience, 2(1), 32-48.
Ebied, H. M., Revett, K., & Tolba, M. F. (2012). Evaluation of unsupervised feature extraction neural networks for face recognition. Neural Computing and Applications, 1-12.
Face and Gesture Working Group. "FG-Net Aging Data Base." http://www-prima.inrialpes.fr/FGnet/html/benchmarks.html
Hebb, D. O. (1949). The organization of behavior: A neuropsychological approach. John Wiley & Sons.
Oja, E. (1982). Simplified neuron model as a principal component analyzer. Journal of mathematical biology, 15(3), 267-273.
Qiu, J., Wang, H., Lu, J., Zhang, B., & Du, K. L. (2012). Neural Network Implementations for PCA and Its Extensions. ISRN Artificial Intelligence, 2012.
Sanger, T. D. (1989). Optimal unsupervised learning in a single-layer linear feedforward neural network. Neural networks, 2(6), 459-473.
Wang, C. W., & Jeng, J. H. (2012, November). Image compression using PCA with clustering. In Intelligent Signal Processing and Communications Systems (ISPACS), 2012 International Symposium on (pp. 458-462). IEEE.