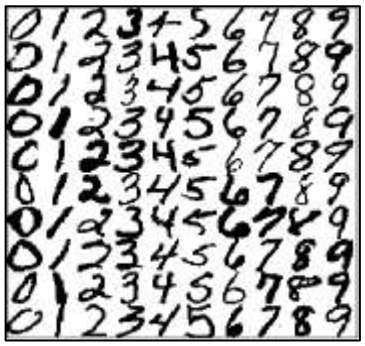
Recognizing Handwritten Digits Using the LLE Algorithm with Back Propagation

Lori Cillo, Attebury Honors Program
Dr. Rajan Alex, Mentor
West Texas A&M University
Canyon, Texas

ABSTRACT. This work is to explore a methodology for recognizing patterns such as handwritten digits as a computer program just as we humans recognize them. The Locally Linear Embedding (LLE) algorithm is a simple yet powerful algorithm to recognize handwritten characters by finding a set of linearly independent components for a given set of characters. Then the algorithm expresses an arbitrary character image as a linear combination of the linearly independent set. That is, the algorithm discovers mathematically meaningful features for a large input example pattern set and then uses these features to recognize a new input character pattern. We use simulation via Matlab with an input of handwritten digits to understand how the LLE algorithm mathematically reduces the dimensionality of the inputs, finds these features, applies the back-propagation algorithm to learn ,and then minimizes the errors in categorizing them into particular categories. The experiment demonstrates that the LLE combined with back-propagation algorithm can learn and adjust its results when it inaccurately categorizes an arbitrary input and eventually predict the correct category it belongs too as well as establishes its viability as a methodology to solving far more complex problems of pattern recognition.

Our goal in this experiment was to find some features that identify each character during the training phase and then to recognize the digit that represents an arbitrary input image from the previously identified features. We do a study of the LLE algorithm with back propagation using the training image set of handwritten digits from the benchmark MNIST database as an input within Matlab. The digits are each images of 28*28 pixels stored in a row of a matrix 784 cells long with 100 images for each digit from 0-9. A set of handwritten digits can be written in a multitude of ways, but each has certain recognizable features. A 0 is a circle, 1 is a straight line, and a 2 is a half circle and a line. Each digit is made up of many data points, but they can be defined by what makes them unique from each other. The LLE algorithm reduces the dimensionality of a series of inputs and defines them as features computed in a mathematically meaningful way and back-propagation to adjust the results when they inaccurately categorize an inputted digit and eventually predict the correct category to which it belongs.

Some of the Digits:



Reducing the dimensionality of an input is the act of taking each image which has many data points and simplifying it into attributes or features. Two of the most common ways to go about this are PCA (principal component analysis) and MDS (multidimensional scaling). MDS takes an input matrix giving dissimilarities between pairs of items and outputs a coordinate matrix whose configuration minimizes a loss function. PCA allows us to compute a linear transformation that maps data from a high dimensional space to a lower dimensional sub-space. Both methods are equivalent if the distances correspond to Euclidean distances, are both simple to implement, and neither involves local minima. However, neither is capable of generating highly nonlinear embeddings. With the LLE algorithm, we can do that as well as map data into a single global coordinate system.

A single global coordinate system means that every digit you input is defined as a linear combination of the same features. In our experiment, the min state of a digit is nine features because the dimensionality of the min state should be less by one than the number of classes, of which there are 10, the digits 0 through 9.

There are two ways of doing this. One way is to fix a radius for a hypersphere, the data within which are taken as neighbors. In this case, the number of neighbors may not be fixed. The other way is to define the number of neighbors (K) for each data point. The local geometry of each image is modeled as a linear combination of these K nearest neighbors. Using k=18, store references to each nearest neighbor in a K x N proximity matrix where every column has 18 references that will form an approximation of the image. You can represent each image as a linear combination of its neighbors using:

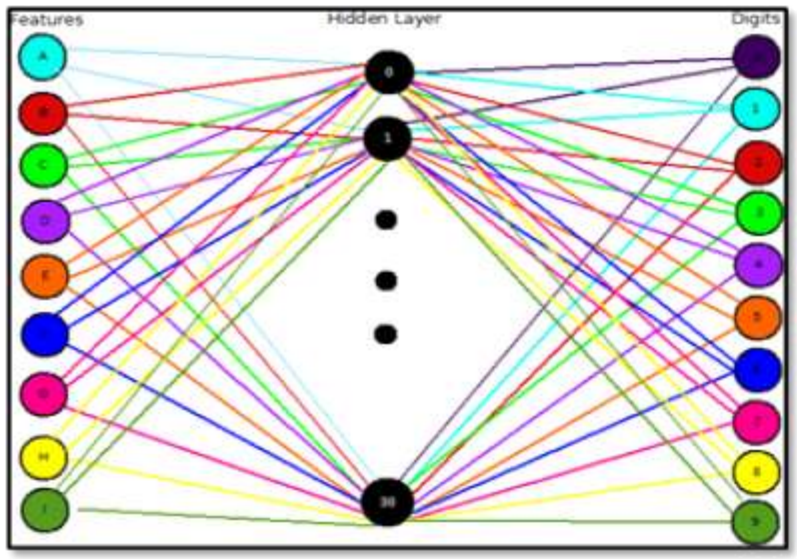
If x_j is not a neighbor of x_i, then w_{ij} is 0 and is not computed in the sum to find x_i. To solve the equation we must mathematically discover the value of w_{ij} for only our 18 neighbors.

We want to find the reconstruction weights so for each image, we create a matrix that takes the indexes of the neighbors from the proximity matrix and puts each neighbor into its own row of 784 cells, using the indexes of the proximity matrix. Next, create a 18×1 vector of the original image repeated, calling it Z, and subtract each neighbor from it. Compute the local covariance with C = Z' * Z and finally solve C *w = 1 for w. We create the following equation from the above:

$$\sum_{k=1}^{N} w_{ik}(x_i - x_j)(x_i - x_k) = 1 \qquad (5)$$

We use the above equation which states that all the sums will add to together to be one, noting that though the resultant matrix is 1000*1000 only the 18 active weights will actually contain values.

Next, map the coordinates for each image. To do this, create a sparse matrix of 1000×1000 that has 0s for all weights except for the neighbors. Find the covariance matrix using the sparse matrix as M in M = (I-W)'(I-W). Perform Eigen Analysis on M. Choose d Eigenvectors corresponding to the lowest d non-zero Eigenvalues. The matrix thus obtained is a d X N matrix. Each column is the corresponding d dimensional vector obtained by reducing the dimensionality of the original vector. Use these eigenvectors and the weights to create a 1000*9 matrix that shows each image as a linear combination of these components. The discovered weights times the neighbors added together will result in the original image. The closer the weight is to one the closer the neighbor is to the image.



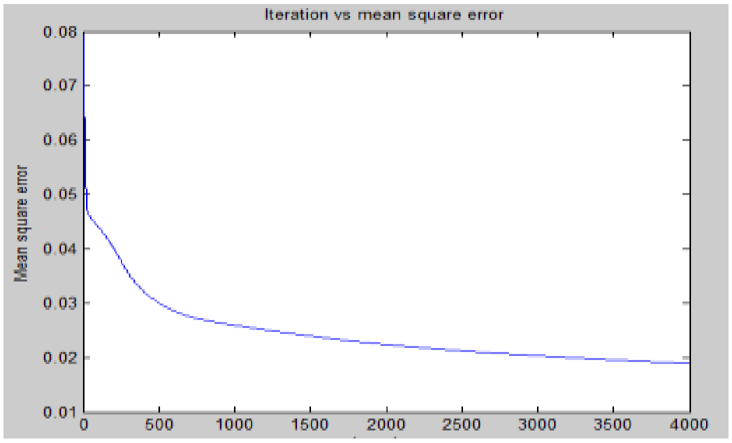
We use the dimensionality reduced vector of an image obtained from the LLE algorithm as input into the hidden layer of the back propagation algorithm. The results of the hidden layer are then used as input to find the digit represented by the nine features. The hidden layer in our experiment consists of 30 nodes which are each a linear combination of the nine features. The network layer represents each digit as a linear combination of the 10 digits. The goal is to use the training images to allow the neural network to learn and to accurately categorize an image as a digit.

To do this, we must first define two matrices, v and w, consisting of random numbers between -1 and 1. The matrix v will contain the end result of the weights multiplied by the initial 9 features to find the 30 hidden layer nodes, and w will contain the weights multiplied by the 30 hidden layer nodes that define each digit uniquely. The deltaV matrix is 30 * 784 and is used to hold the 784 bits that make each of the 30 hidden layer nodes and deltaW which is 10*30 contains the 9 features each of the 30 hidden layer nodes consists of.

The algorithm will iterate 4000 times or until it reaches weights that find the correct digit for an input image. Each iteration slightly adjusts the weights, each time improving the accuracy of predicting a digit. Each iteration looks at all 1000 images in the training set and divides the by the maximum to normalize the input as between -1 and 1.

For each image within the iteration, the algorithm calls getOutput which returns the values of the hidden layer nodes found by using the random weights in v defined earlier and the normalized matrix V which contains the images defined as nine features. It then calls getOutput again this time with the first output and the random weights in w to define each digit as a linear combination of the 30 hidden layer nodes.

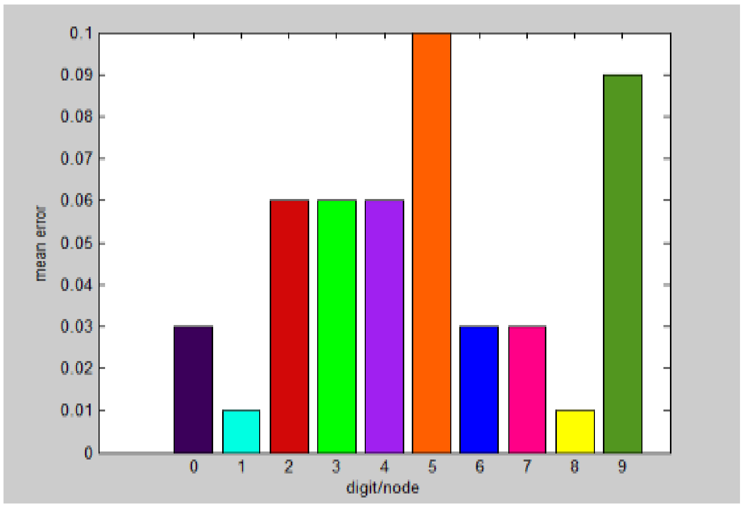
Each iteration adjusts the weights that incorrectly categorized images to increase accuracy. Each iteration reduces the mean error and as the iterations increase, the error tends to 0.

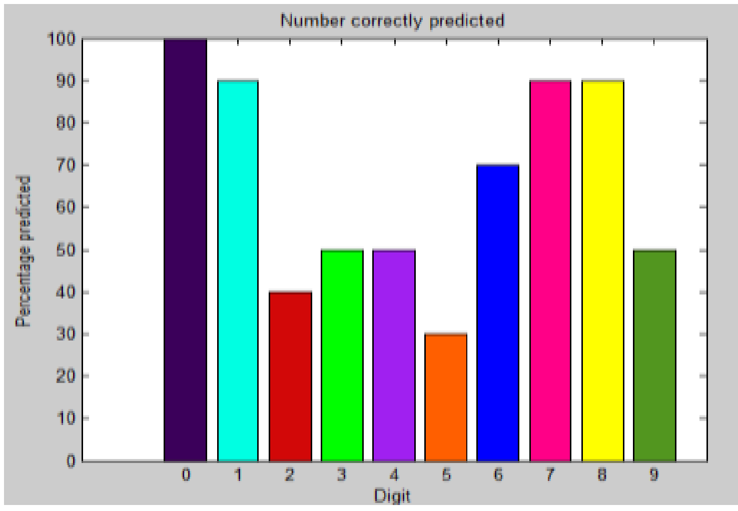


Each number has a variety of inputs with different thicknesses, orientations, shapes, and features that overlap with one another. Because the LLE algorithm classifies based on a small set of components that express differences between digits, automatic classification can be difficult

Numbers that have 90 percent or greater recognition (0, 1, 6, 7, 8) have less variety of ways to be drawn. 0 is a circle, 1 is a straight line, 7 is a horizontal and vertical line, and 8 is two circles on top of one another. Numbers that have 50 percent or less recognition (2, 3, 4, 5, 9) can be drawn in such a way that they closely resemble another digit. The following digits are very similar to one another:

2 and 3: 5 and 6: 4 and 9:





Works Cited

Saul, Lawrence K, Park Ave, Florham Park, and Sam T Roweis. 2001. An Introduction to Locally Linear Embedding.

Kouropteva, O, O Okun, and M Pietikäinen. 2003. Classification of handwritten digits using supervised locally linear embedding algorithm and support vector machine