Exploring Data Dimensionality Reduction through Facial Recognition

Komel Merchant KOM218@LEHIGH.EDU
Lehigh University Department of Electrical and Computer Engineering

David Galaydick DAG217@LEHIGH.EDU
Lehigh University Department of Electrical and Computer Engineering

Gustavo Grinsteins GUG218@LEHIGH.EDU
Lehigh University Department of Mechanical Engineering

Abstract

In this project, we attempt to explore facial recognition using the ORL database of faces from the lens of computational time and accuracy trade-off. Since most applications require facial recognition to be fast, we attempt to use a solution which would support fast computational time when vast amounts of data need to be processed. Our solution is to compare State-of-the-art Data Dimensionality Reduction techniques (Principal Component Analysis and Fast Fourier Transforms) by using simple K-Nearest Neighboors and a fully connected Neural Network classifier to solve the problem.

Our results show that using Principal Component Analysis on a K-Nearest Neighboors classifier provides the best performance in terms of computational time and test accuracy.

1. Introduction

1.1. Problem

Facial Recognition has many applications in the real world. Areas like security, defence, and medicine have benefited from developments in this area. For this reason, it has been an extremely well-studied problem since the 70's. Our problem is two-fold; the first problem deals with facial recognition which is a multi-class classification problem where the data variance is usually fairly small. This is different than typical problems in classification because the deviations between certain labels usually comes down to small, local variations (e.g. shape of nose, small intensity variations between pigment of eyes, etc). Many solutions to this exists. Some use a purely data-driven approach, relying on Neural Network (NN) architectures to solve the problem. Investigations into a method using such techniques have proved to have an extremely high accuracy, but with a slower speed (Schroff Florian & james., 2015). The sub-problem we are investigating, though, is an attempt to solve both the issue of accuracy as well as time. Since most applications of facial recognition require a reasonably fast evaluation time. To solve this problem, we will attempt to use an array of different Data Dimensionality reduction (DDR) techniques to speed up the task. We also rely in our choice of model. Very much like (Adeel, 2011) and (Setiawan & Muttaqin, 2015), use a very simple KNN classifier to perform fast-simple classification. We also attempt to explore DDR in the context of a more complex model by using a multi-class 2-Layer NN.

1.2. Challenges

In our case, much of the difficulty finding a point where we get good accuracy as well as good computational run-time. When you reduce the data by too much, you tend to loose a lot of local features in the process. One of the challenges, as mentioned before, was to be able to preserve the minor variations from image to image such that those features can bee distinguished. The goal of our project is to find a point where this is minimized.

Proceedings of the 30th International Conference on Machine Learning, Atlanta, Georgia, USA, 2013. JMLR: W&CP volume 28.

2. Methodology

Our method for this experiment was to feed data into the different classifiers while varying some of their parameters to observe the effect on accuracy and testing time.

2.1. Dataset

For our dataset we used the ORL database of faces out of Cambridge University. This contains 400 pictures of faces from 40 different people, with 10 faces per person, each with some different kind of expression. All of the images featured the subject in front of a neutral background, and all of the expressions were still forward facing. This was to make data suited only to identifying the actual face rather than dealing with poses and trying to ignore background noise. There is an extension of the data provided by Zhejiang University that scaled the images to be 32x32 and 64x64 and provided a simple Matlab matrix file. We only used the 32x32 for training and testing our algorithms.

2.2. Model Architectures

K-Nearest Neighbor

The KNN used for testing was developed by calculating the distance of a target image to be classified to all the other images in the data set. After these distances are calculated, K-Nearest distances are picked, from these values a majority vote for the classification of the target image is done and the image is classified. TThis algorithm is reoeated for all the target images.

Fully Connected Neural Network

Neural Networks are classifying algorithms that can be used for facial recognition. NNs are composed of an input layer, hidden layers, and an output layer. For this project, the NN is a 2-layer (one Hidden layer) NN. The architecture used consists of a Cross-Entropy loss function, Softmax output layer, and Gradient Descent optimization with a learning rate of $\eta = 1$. This network set up allows for a each output node value to be distributed in terms of percents that add up to a value of one aiding multi-class NN problems (Hinton, 2013). The NN was built using Matlab's Neural Network toolbox. This module of Matlab facilitated the construction of the architecture, training, and testing of the NN.

2.3. Data Reduction techniques

PCA - Principal Component Analysis

Principal component analysis is a technique for projecting data onto lower dimensional linear space. More specifically, this is the orthogonal projection onto the principal subspace such that the variance of the projected data is maximized. This high variance is what indicates a principal component.

The process:

1. Take the mean of the data:
   $\mu = \frac{1}{N}\Sigma_{n=1}^{N} x_n$
2. Center the data around the mean and calculate the covariance of the data:
   $\sigma = \frac{1}{N}\Sigma_{n=1}^{N}(x_n - \mu)(x_n - \mu)^T$
3. Perform singular value decomposition (SVD) on σ to find the desired number of eigenvectors and their associated eigenvalues for the data. There are numerous ways of doing this, but for testing we did this through the eigs() function in Matlab. After doing this, multiply the original data by the matrix of eigenvectors to project the data onto the lower dimensional space.

Fourier Analysis

For this algorithm, we first must take in a two-dimentional image and transform into the Fourier Domain. This is done by using the following equation:

$\mathrm{Y}_{p+1,q+1} = \sum_{j=0}^{m-1}\sum_{k=0}^{n-1}\omega_m^{jp}\omega_n^{kq}X_{j+1,k+1}$

Where m and n span the dimensions of the image, and $\omega_a = \exp\{\frac{2\pi i}{a}\}$

This compresses the original signal. In figure 1 we see how this work in two dimensions. The original signal goes through a fast Fourier transform which we then shift to the centers and extract the frequencies at the center to use as our new data point (the third image from the left in Figure 1. The Problem with this however, is that the we do lose some salient features. When we reconstruct the image,we see that there is a slight blur to which might cause a loss in local features.

For our algorithm, we have one parameter, w, which corresponds to the side length of the window.

3. Results

3.1. Fourier Analysis

Test Accuracy

For Fourier window size we used the parameter $w \in \{10 \times 10, 20 \times 20, 30 \times 30\}$.

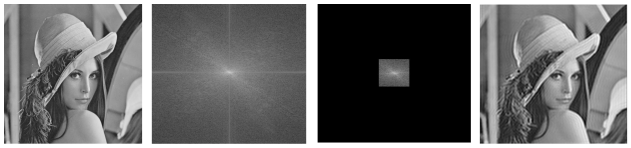

Figure 1. Converting Lena to FFT, Extracting a Window around Lena and reconstructing reduced image using padding and inverse FFT from left to right

In addition, we tested these DDR parameters with KNN with parameters, $k \in \{1, 2, 5, 10\}$ and NN hidden nodes $\in \{10, 36, 100\}$

In terms of accuracy for KNN, we found it to be fairly consistent results across all of the different values of w. The highest always seemed to be between the lowest 2 k-values, 1 and 2, at a percentage accuracy of around 85 to 95 percent.The lowest across all values of w seemed to come from using a K-value of 10 which consistently produced an accuracy of 65 percent. The results for this can be seen in Figure 9

For NN, the highest accuracy seems to occur at the hidden node value of 36 for w = 30x30 and 20x20. Applying FFT varies the accuracy slightly indicating that greater FFT reductions result in higher accuracy with w = 10x10 giving 80% accuracy. The lowest accuracy occurs at 10 hidden nodes at FFT of$w$ = 10x10.

Computation Time

In terms of time for the KNN model and the NN model, we achieved the results we had expected. As we decrease the window size of the FFT domain, we get lower computational times. We do observe, interestingly, that the 30x30 window yields a higher runtime than with no FFT for KNN. the reason for this most likely has to be the fact that the computations we are dealing with are in the complex domain. Since the window size closely resembles the 32x32 image size. We get the best computational performance when $k = 1$ and w = 10x10 for KNN and when Hidden Nodes = 1 and w = 10x10.

3.2. Principal Component Analysis

Test Accuracy

We performed tests on the data by reducing the dimensionality from 1024 to various lower dimensions M and applying all of those results to multiple values of k for the KNN algorithm and multiple hidden nodes for NN. For our tests we used:

$M = 1024, 300, 200, 70, 50, 30$

$k = 1, 2, 5, 10$

Hidden Nodes $= 10, 36, 100$

Testing was performed using a randomly selected 20% of the data set for KNN. For NN, 10% of the data was selected randomly for testing.

For KNN, the trend we observed in regards to accuracy is that reducing the dimensionality didn't have a very large impact on the accuracy. Likely due to the fact that each set of data was split into testing and training data separately, the results seem unintuitive with some higher dimensional data being less accurate, but the results are still more or less within expectations.

Changing k had a more notable impact on the accuracy. Increasing k generally decreased the accuracy. Larger k values in the KNN algorithm tend to increase the chances of incorrect labels being introduced into the neighborhood. It appears that the data set we used is more sensitive to that issue, so accuracy was always highest when k was 1.The figures for K-NN with FFT dimentionality Reduction can be found in Figure 10.

For NN accuracy tends to increase as the number of hidden nodes increase as observed in figure 4 reaching the highest accuracy percentages at 100 hidden nodes.

As seen in figure 2 most of the variance in the data used can be achieved by using 30 to 100 principal components (M). The reduction of data makes the NN accuracy increase due to the fact that noise is reduce and the important features that make the images distinct from the other are highlighted. Therefore, the accuracy performance of the NN performs better at low values of M (70, 50, and 30) and performs poorly at high M values. It is important to note that the highest values for accuracy occur when no PCA DDR is implemented.

Computation Time

The tests we performed for computation time were exactly the same as for accuracy in terms of conditions they were run under. At the start of the KNN and NN algorithm, the time was recorded, and at the end the time was recorded again and the difference was taken.

The trend we observed in regards to computation time was that reducing the dimensionality had a very noticeable effect on the time. The unreduced data was taken at least thrice as long as the most reduced data to compute, which is within expectations.

Changing the value for k didn't seem to have a consistent effect on the computation time. There were some scenarios where k = 1 yielded the longest time and

some where k = 10 yielded the longest computation time. Across various test cases, we found this to be the case, so it's safe to assume that the cause is likely due to the machine the tests were run on behaving inconsistently in minor ways.

Changing the hidden nodes in the NN to higher values increases the computational time required to test the data. This increase in time seems to be proportional to the number of dimentions in the data. For example in figure 6 when no PCA is used the total dimensionality of the data is 1024 pixels and the highest computational time at that setting occurs at 100 hidden nodes for the NN.

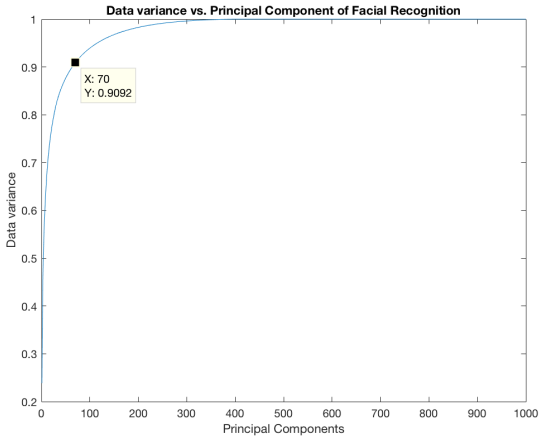

Figure 2. 90% of the data variance in the ORL database of faces can be achieve by using round 70 principal components

4. Conclusion

From what we've observed from our test results, we can conclude that of the methods we explored, PCA seems to be the most effective in terms of reducing computation time and maintaining accuracy. Of the algorithms we applied the reduced data to, this trend was most apparent with KNN. KNN not only fast, keeping the computation time below 0.05 seconds for reduced data, but also generally kept its accuracy above 80 percent for values of k that were low enough to not yield incorrect results. Factoring in the percent change of run time, FFT and PCA were comparable, as seen in the comparison of times for the two methods on the neural network, but the accuracy for the neural network was generally lower than it was for KNN, so we can conclude that with the algorithms we used, PCA and KNN is most optimal for this problem.

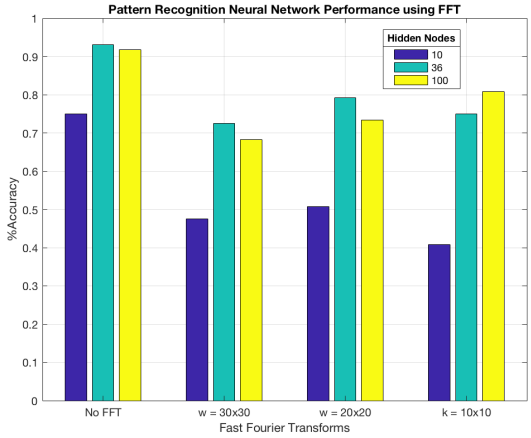

Figure 3. Accuracy performance of ORL database of faces using different magnitudes of FFT at different NN hidden node values

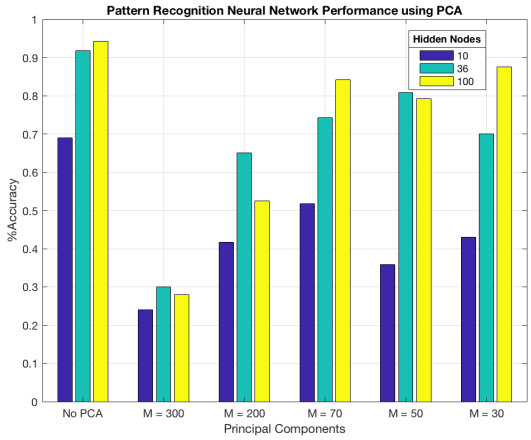

Figure 4. Accuracy performance of ORL database of faces using different magnitudes of PCA at different NN hidden node values

References

Adeel, Mohammed Abdul. Human face recognition based on multidimensional pca and extreme learning machine. *Pattern Recognition*, 44(10-11):211–229, 2011.

Hinton, Geoffrey. Nerual networks for machine learning. University Video Lecture through Coursera, 2013.

Schroff Florian, Kalenichenko Dmitry and james., Philbin. Facenet: A unified embedding for face

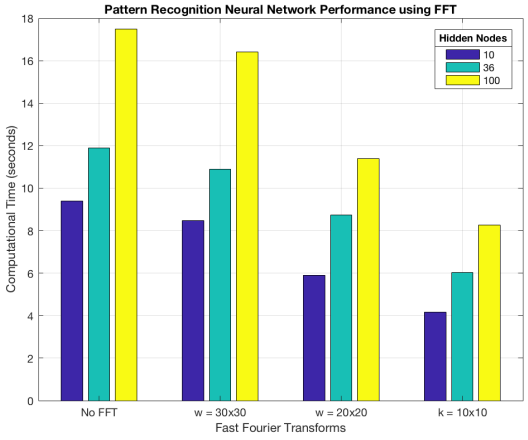

Figure 5. Time performance of ORL database of faces using different magnitudes of FFT at different NN hidden node values

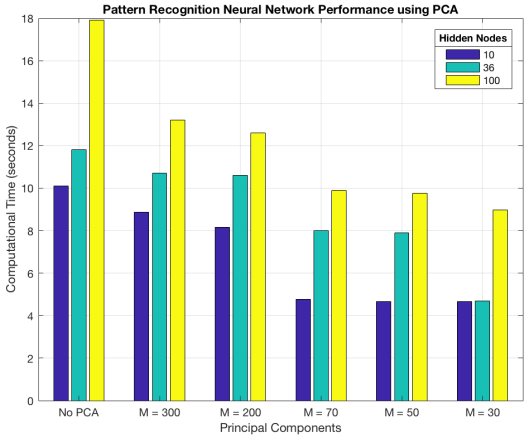

Figure 6. Time performance of ORL database of faces using different magnitudes of PCA at different NN hidden node values

recognition and clustering. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015.

Setiawan, Eko and Muttaqin, Adharul. Implementation of k-nearest neighbors face recognition on low-power processor. *TELKOMNIKA*, 13(3):949–954, 2015.

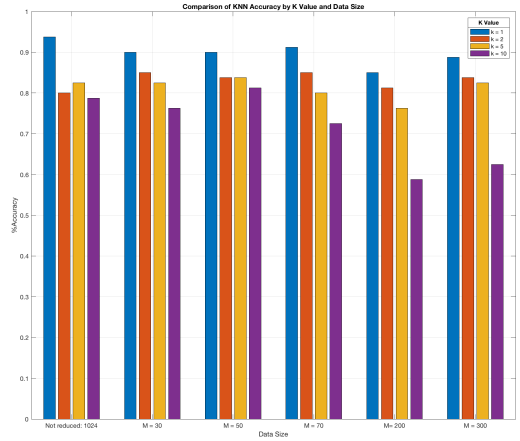

Figure 7. Accuracy performance of ORL database of faces using different magnitudes of PCA at different K values for the KNN algorithm

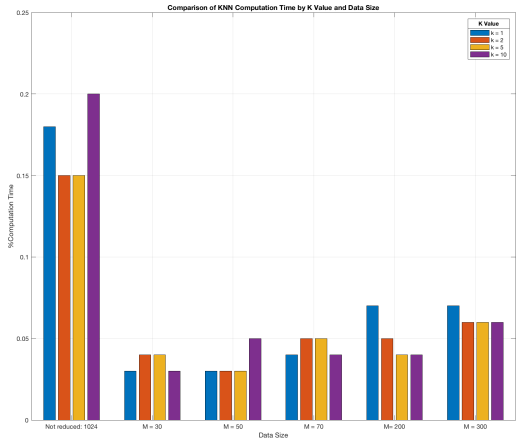

Figure 8. Time performance of ORL database of faces using different magnitudes of PCA at different K values for the KNN algorithm

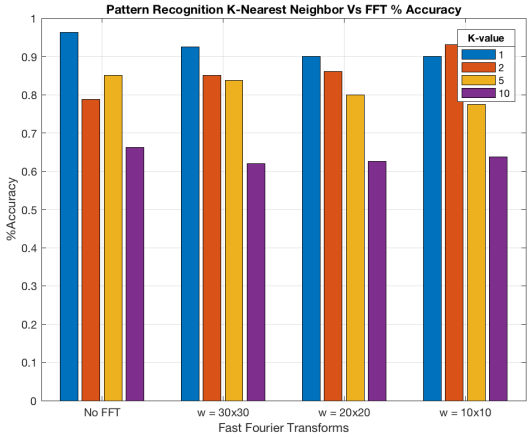

Figure 9. Accuracy performance of ORL database of faces using different FFR window sizes and KNN Algorithm

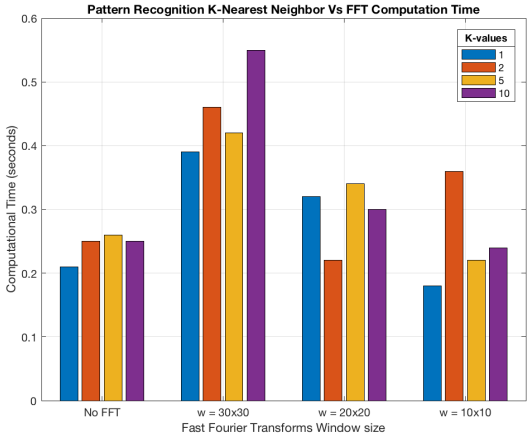

Figure 10. Time performance of ORL database of faces using different FFR window sizes and KNN Algorithm